Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

PROPUESTA DE UNA METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE *SOFTWARE* EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002

Norma Magaly Chonay Vásquez
Asesorado por Inga. Zulma Karina Aguirre Ordoñez

Guatemala, marzo de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

PROPUESTA DE UNA METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE *SOFTWARE* EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

NORMA MAGALY CHONAY VÁSQUEZ
ASESORADO POR INGA. ZULMA KARINA AGUIRRE ORDOÑEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERA EN CIENCIAS Y SISTEMAS

GUATEMALA, MARZO DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADORA	Inga. Elizabeth Domínguez Alvarado
EXAMINADOR	Ing. Juan Alvaro Díaz Ardavín
EXAMINADOR	Ing. José Ricardo Morales Prado
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

PROPUESTA DE UNA METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE *SOFTWARE* EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha febrero de 2011.

Norma Magaly Chonay Vásquez

Guatemala, noviembre de 2011.

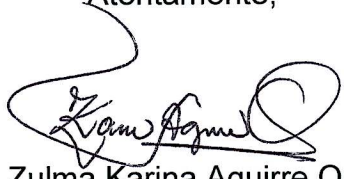
Ing. Carlos Alfredo Azurdia Morales
Coordinador Comisión de Trabajos de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero:

Por medio de la presente hago de su conocimiento que, he procedido a revisar el trabajo de graduación titulado **PROPUESTA DE UNA METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE SOFTWARE EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002**, elaborado por la estudiante Norma Magaly Chonay Vásquez, y de acuerdo a mi criterio, se encuentra concluido y cumple con los objetivos propuestos para su desarrollo.

Sin otro particular, me suscribo de usted.

Atentamente,



Inga. Zulma Karina Aguirre Ordoñez

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 18 de Noviembre de 2011

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de la estudiante **NORMA MAGALY CHONAY VÁSQUEZ** carné **2000-10569,** titulado: **"PROPUESTA DE UNA METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE SOFTWARE EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado ***"PROPUESTA DE UNA METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE SOFTWARE EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002"*** *presentado por la estudiante NORMA MAGALY CHONAY VÁSQUEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCIÓN DE
INGENIERÍA EN CIENCIAS
Y SISTEMAS

Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería Ciencias y Sistemas

Guatemala, 07 de marzo 2012

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.123-2012

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **PROPUESTA DE UNA METODOLOGIA PARA ESTÁNDARES DE PROGRAMACIÓN DE SOFTWARE EN GUATEMALA, TOMANDO EN CUENTA LAS MEJORES PRÁCTICAS PARA ESCRIBIR CÓDIGO, LLAMADA EXTILO REGULAR 9002,** presentado por la estudiante universitaria, **Norma Magaly Chonay Vásquez,** autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
DECANO

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, marzo de 2012

/cc
c.c. archivo.

ACTO QUE DEDICO A:

Dios	Por permitirme culminar esta etapa.
Mi mamá	María Isabel Vásquez Sotoj, por estar siempre conmigo.

AGRADECIMIENTOS A:

Dios	Por todo su amor y todo lo que me ha permitido vivir, en especial por las personas que he conocido.
Mi mamá	Por su amor y paciencia.
Mi papá José Chonay, mi tía Olga Chonay y mis hermanos: Francisco, Gaby, Jose, y José Francisco	Por sus muestras de cariño.
Mis amigas	Ana Lucía Cabrera, Emy Villatoro, Flor Rodas, Gaby Tobar, Julia Lemus, Karla Rascón, Lucía Orozco, Mar Girón y Marisol Amado, por su amistad incondicional.
Inga. Damaris de López e Inga. Zulma Aguirre	Por todo su cariño, consejos y apoyo; mil gracias.
Mis compañeros y amigos de la Universidad de San Carlos de Guatemala	Por su amistad, en especial a Henry Castañeda, Alan Morales (q.e.p.d.), Mario Milián y Gustavo Alvarado.

Las familias que me abrieron las puertas de su hogar en el transcurso de mi carrera

En especial a la familia Chacón Matamoros, infinitas gracias.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

DICOM	*Digital Imaging and Communication in Medicine*. Estándar usado para el intercambio de imágenes médicas: almacenamiento, impresión, transmisión y manejo de imágenes médicas.
GUID	*Globally unique identifier*. Número de 16 bytes, implementado por *Microsoft*® para identificar componentes de *software*, aunque tengan el mismo nombre.
Modelo OSI (*Open System Interconnection*)	Es el modelo de red creado por la Organización Internacional para la Estandarización (ISO), usado para definir la arquitectura de interconexión entre sistemas de comunicaciones. Consta de 7 capas.
Script	Es un archivo simple de texto plano, que contiene órdenes que se ejecutan para interactuar con el sistema operativo (archivo de procesamiento por lotes) o con las aplicaciones (en páginas *web* se utilizan *scripts* del lado del cliente y del lado del servidor).
Teletype	Tipo de letra que simula el tipo de letra de una máquina de escribir.

RESUMEN

Estándares publicados por organizaciones como ANSI (*American National Standards Institute*) e ISO (*International Organization for Standardization*) en la segunda mitad de los años noventa del siglo XIX, iniciaron la regulación de lenguajes de programación, siendo FORTRAN y ALGOL los primeros en ser normalizados; éstas y otras normas que fueron cobrando auge a medida que la producción de *software* fue aumentando, y las entidades a todo nivel obtuvieron ventajas como calidad, confiabilidad y competitividad, al utilizar guías para desarrollar e implementar programas computacionales.

Lo anterior sienta precedente para las certificaciones existentes a la fecha, como CMM (*Capability Maturity Model*), ISO 9001 o ISO 15504 - SPICE (*Software Process Improvement and Capability dEtermination*), algunas de ellas alcanzadas por empresas guatemaltecas, ya que reconocen la importancia y los beneficios que se obtienen.

En los años setenta fue escrito el libro *The Elements of Programming Style*, por el científico canadiense Brian Wilson Kernighan, quien dentro de las reglas de programación menciona que, se deben utilizar nombres de variables significativos y dar formato al código, para que se le facilite entenderlo a quien lo lea.

Lo anterior es apoyado por programadores de Pascal, Modula, Java y .NET, quienes promueven convenciones para nombrar a los identificadores de los elementos del *software*. Convenciones que algunas empresas no las recomiendan en los entornos de desarrollo integrados (*IDEs*), ya que a través

de estos *IDEs* se puede conocer el tipo de dato de las variables; pero para los lenguajes dinámicamente tipificados, no se cuenta con estos entornos de desarrollo.

Por ello se propone una metodología para estándares de programación de *software* en Guatemala, que se recomienda utilizar también para nombrar elementos de lenguajes estáticamente tipificados, ya que por el hecho de ser una guía, provee de ventajas para los involucrados en la producción de *software*.

La metodología Extilo Regular 9002, propone una nomenclatura para nombrar los elementos más comunes, implementados por lenguajes de programación y sistemas de gestión de base de datos, utilizando la notación camelCase, combinada con el Sistema Húngaro y la Aplicación Húngara. En ella, se indica por medio de diagramas la sintaxis sugerida para los nombres, además por medio de tablas se listan los prefijos para los tipos de datos primitivos, modificadores de las clases, atributos y métodos, prefijos de las etiquetas HTML, de los tipos de índices para las bases de datos y de otros objetos utilizados en ellas. También en figuras, se resalta la sangría que es importante aplicar al escribir estructuras selectivas y repetitivas en el programa.

Aunque la normalización esté basada en Visual Basic .Net, Visual C#, PHP, Java, C++, Oracle, *Microsoft* SQL, MySQL y PostgreSQL, ésta puede aplicarse a otros lenguajes de programación y sistemas de gestión de base de datos, indistintamente del idioma utilizado para desarrollar el *software*.

Extilo Regular 9002 se pone en práctica desarrollando dos prototipos de una aplicación que permite agregar, editar, eliminar y consultar datos de usuarios, así como recuperar la contraseña de determinado usuario. El primer

prototipo está elaborado en Visual C# y *Microsoft* SQL, y el segundo en Visual Basic .NET y Oracle.

El objetivo de este trabajo de graduación, es que esta metodología se incorpore en el contenido del laboratorio de los cursos de programación del pensum actual de la carrera de Ingeniería en Ciencias y Sistemas, de la Facultad de Ingeniería, de la Universidad de San Carlos de Guatemala, para que los futuros profesionales la conozcan y apliquen a lo largo de su carrera universitaria, y luego en el desarrollo profesional; es por ello que se elaboró un folleto de Extilo Regular 9002 basado en el presente trabajo de graduación, para su distribución.

OBJETIVOS

General

Proponer una metodología para estándares de programación de *software* en Guatemala, tomando en cuenta las mejores prácticas para escribir código, llamada Extilo Regular 9002.

Específicos

1. Proponer una metodología para programar tanto a nivel de base de datos como a nivel de aplicación, que pueda aplicarse independientemente del manejador de base de datos y del lenguaje utilizado.

2. Aplicar la metodología Extilo Regular 9002, desarrollando dos aplicaciones que permitan crear, eliminar y modificar usuarios, así como cambiarles la contraseña, utilizando las últimas versiones de prueba (*trial*) de algunos lenguajes de programación y sistemas de base de datos utilizados en el desarrollo de *software* en Guatemala.

3. Plasmar la metodología Extilo Regular 9002 en un folleto, para poder incorporarla en el contenido del laboratorio de los cursos de programación del pensum actual de la carrera de Ingeniería en Ciencias y Sistemas, de la Facultad de Ingeniería, de la Universidad de San Carlos de Guatemala.

INTRODUCCIÓN

Con la aparición de entes encomendados a regular internacionalmente la fabricación, comercio e interacción de productos y servicios, y a medida que fueron surgiendo los lenguajes de programación, se crearon estándares para especificar cada lenguaje de programación, y así también reglamentar su uso.

Para seguir promoviendo el uso de los estándares y así beneficiarse de las ventajas que proveen, el presente trabajo de investigación tiene como objetivo principal, formular una metodología que estandarice el formato y estructura del código fuente de las aplicaciones de *software* a desarrollar, independientemente del lenguaje utilizado y del idioma de la herramienta utilizada.

Extilo Regular 9002 propone una nomenclatura para nombrar elementos de lenguajes de programación y sistemas de base de datos, la propuesta está basada en Visual Basic .Net, Visual C#, PHP, Java, C++, Oracle, *Microsoft* SQL, MySQL y PostgreSQL. Además indica buenas prácticas para programar, como fomentar el paradigma modular o estructurado.

Este trabajo está organizado en cuatro capítulos, el primer capítulo trata la historia de los estándares, un paso a través del tiempo, cómo iniciaron, la situación actual y el futuro que tienen las regulaciones de programación.

En el capítulo dos, se aborda la importancia de la estandarización, haciendo énfasis en las grandes ventajas de ella, y desventajas que se deben tomar en cuenta, pero que son menos significativas que los beneficios que se

alcanzan. En el tercer capítulo se desarrolla toda la propuesta, por medio de diagramas, tablas, ejemplos y guías útiles e interesantes para escribir código fuente.

El capítulo cuatro presenta el código fuente del primer prototipo desarrollado en Visual C# y *Microsoft* SQL, cuyo fin es gestionar usuarios de un sistema y que se pueda recuperar la contraseña de los usuarios. El segundo prototipo elaborado en Visual Basic .NET y Oracle, se presenta en el apéndice. Para ambos prototipos además del código fuente, se muestran imágenes de los diversos formularios de la aplicación.

1. HISTORIA DE LOS ESTÁNDARES

El surgimiento de diversos lenguajes de programación desde 1949 con Short Code, A-0 (Math Matic) en 1951, FORTRAN (*FORmula TRANslating system*) en 1957, ALGOL y LISP un año más tarde, COBOL en 1960, hasta los más recientes como C# y los lenguajes de quinta generación, por ejemplo, ha dado lugar a la creación de organizaciones encargadas de regular su uso entre los programadores de cada lenguaje, indicando así especificaciones de sintaxis y semántica, es decir, definir la estructura y la manera de interpretar los programas escritos en cada lenguaje de programación; con el objeto de asegurar el mantenimiento, la portabilidad, fiabilidad y eficiente ejecución de los programas escritos en cada lenguaje, independientemente del sistema de cómputo sobre el cual se desarrollen y ejecuten.

Por lo anterior, varios lenguajes han sido normalizados por organizaciones como el Instituto de Normalización Nacional Americano (ANSI), fundado originalmente como AESC (*American Engineering Standards Committee*) en 1919; y la Organización Internacional de Normalización (ISO) creada en 1947. Siendo FORTRAN el primer lenguaje de programación estandarizado, en marzo de 1966 por ASA (*American Standards Association*), nombre que tuvo ANSI desde 1928 hasta 1969. Y en 1972, ISO publicó su primer estándar de lenguaje de programación para ALGOL: ISO 1538.

Así fueron surgiendo los estándares de programación, y no fue, si no hasta finales de los años 80, cuando las empresas dejaron de percibir los estándares no sólo como una guía para desarrollar productos de calidad y fiabilidad, sino como algo indispensable para competir exitosamente en el

mercado internacional. Afán que también conlleva a percibir la informatización como algo indispensable para el crecimiento y optimización de los procesos de cualquier tipo de empresa; ayudándose para ello de la tecnología, lo cual se traduce en sistemas de información, desarrollados por su propio departamento de informática o por una empresa subcontratada.

Los siguientes datos estadísticos, indican que el sector productivo de *software*, ya forma parte imprescindible de los rubros de producción de los países a nivel mundial; por lo que ya es tema de investigaciones y proyectos que tienen como fin promover su crecimiento y asegurar su calidad, para que las empresas dedicadas a este sector, estén aptas para competir local, regional e internacionalmente.

Según el análisis del sector productor de *software* en Guatemala, elaborado por Proexport Colombia y el Banco Interamericano de Desarrollo - Fondo Multilateral de Inversión (BID - FOMIN), desde 1999 hasta 2003, dicho sector experimentó un crecimiento de 28,4 por ciento. Este análisis indica que el mayor crecimiento del sector se dio en la década de los noventas, cuando la mayoría de las grandes empresas iniciaron la automatización de sus procesos. También en dicho documento, se menciona que existen en Guatemala muchos competidores en el desarrollo de *software* a la medida, en donde sobresalen Byte, Enter y GYSSA, quienes en conjunto abarcan el 70 por ciento del mercado local.

La publicación No. 18 de CEGESTI, de septiembre 2004, indica que India, Israel e Irlanda son los grandes proveedores de *software*, a nivel internacional, que logran billones de dólares en exportaciones; seguidos por Rusia y China con un promedio de $500 millones de dólares aproximadamente.

A continuación un paso por la historia, a través del tiempo, de los estándares relacionados con el desarrollo de *software*; ya que como se dijo anteriormente, algunos estándares surgieron para regularizar los diversos lenguajes de programación, y como se expone en los siguientes párrafos, otros han surgido con el objeto de regularizar los procesos para el desarrollo de programas de computación, alcanzando con ambos, cierto grado de calidad.

1.1. Antecedentes

Erran Carmel, reconocido Doctor en Administración de Sistemas de Información, en su publicación *The New Software Exporting Nations: Success Factors*, expone que el éxito de la industria del *software* en la India, ha demostrado que contar con estándares de calidad, reconocidos a nivel internacional, rompe el paradigma de que comprar *software* proveniente de un país lejano es un riesgo; ya que la industria del *software* hindú, tiene el CMM nivel 5 (*Capability Maturity Model*), por lo que es considerada, como una industria segura y atractiva.

En Guatemala, la empresa de desarrollo Byte, empresa sobresaliente en el mercado local, está haciendo importantes avances en su certificación CMMI, ya que actualmente está en el Nivel 3. Además fue la primera empresa de servicios en Guatemala en certificarse en ISO 9001:2000, y a la fecha tiene la certificación ISO 9001:2008.

CMM (*Capability Maturity Model*), es un modelo de evaluación de los procesos de una organización, elaborado inicialmente para los procesos de desarrollo e implementación de *software*, por la Universidad Carnegie Mellon para el SEI (*Software Engineering Institute*), centro de investigación y desarrollo

patrocinado por el Departamento de Defensa de los Estados Unidos de América.

El modelo CMM define procesos agrupados en Áreas Clave de Proceso (KPA - *Key Process Area*): gestión, organizacional e ingeniería. Estas áreas a su vez, están agrupadas en niveles de madurez, niveles que la organización alcanza, conforme cumple con las actividades definidas para cada nivel y para los niveles anteriores.

El modelo CMMI (*Capability Maturity Model Integration*) es la evolución de CMM, cuyo objetivo es mejorar la usabilidad de modelos de madurez, integrando varios modelos diferentes en un solo marco (*framework*). Este modelo es aplicable a organizaciones de diferentes sectores, no solamente a las áreas de tecnologías de información, para las que fue desarrollado el CMM. Actualmente hay 3 modelos del CMMI en su tercera versión (1.3), para optimizar los procesos:

- CMMI para el desarrollo (CMMI-DEV o *CMMI for Development*): para desarrollar mejores productos y servicios.
- CMMI para la adquisición (CMMI-ACQ o *CMMI for Acquisition*): para la adquisición de mejores productos y servicios.
- CMMI para servicios (CMMI-SVC o *CMMI for Services*): para proporcionar un servicio superior.

En la versión 3 de estos modelos, se tienen 5 niveles de madurez, que las organizaciones van alcanzando progresivamente:

- Nivel 1: inicial. Los procesos son muy variados y por lo tanto pobremente controlados.

- Nivel 2: gestionado. Existen métricas básicas para ejecutar el proceso y lograr los objetivos.
- Nivel 3: definido. Los estándares, procedimientos y descripciones de procesos, se aplican más rigurosamente.
- Nivel 4: cuantitativamente gestionado. Además de ser un proceso definido, éste se controla utilizando técnicas cuantitativas.
- Nivel 5: optimizado. Teniendo un proceso cuantitativamente gestionado, de forma sistemática se revisa y modifica para mejorarlo continuamente, adaptándolo a los objetivos del negocio

Por otro lado, las normas ISO 9000 son normas de calidad y gestión continua de la misma (métodos de auditoría), establecidas por la Organización Internacional para la Estandarización (ISO). Éstas se pueden aplicar a cualquier tipo de organización o actividad sistemática, orientada a la producción de bienes o servicios.

Una organización que cumple con la norma ISO 9001:2000, sigue los requisitos básicos en cuanto a normas de calidad; y para incrementar la excelencia existe la norma ISO 9004:2000, la cual provee directrices para la mejora del desempeño en sistemas de gestión de calidad.

Actualmente, la principal norma de la familia ISO 9000, es la ISO 9001:2008, norma muy parecida a la ISO 9001:2000, pero con importantes aclaraciones y modificaciones, que no serán descritas en este documento.

Tanto CMMI como ISO 9001:2000, son normas que promueven la gestión continua de la calidad de los procesos, relacionados con la producción de *software*, en el caso del CMMI, y con la producción de cualquier bien y servicio, en el caso de ISO 9001:2000. Sin embargo, estas normas no definen guías

para escribir los programas de *software*, tarea de la fase de implementación del proceso de desarrollo de *software*, que consiste en escribir en determinado lenguaje de programación una secuencia de instrucciones, que integradas forman el *software*, el cual es ejecutado en el *hardware* del ordenador.

El libro *The Elements of Programming Style*, es el primer trabajo relacionado a los estándares de programación, que contiene ejemplos de los lenguajes de esa época: FORTRAN y PL/I.

Fue escrito en la década de los años setenta, por el científico en computación Brian Wilson Kernighan, originario de Ontario, Canadá, y por el escritor y consultor en el campo de la programación James Peterson Plauger. Dicho trabajo sugiere algunas reglas de programación, como por ejemplo:

- Escribir claro, de forma simple y directa, y no sacrificar la claridad por eficiencia.
- Utilizar librerías de funciones lo más que se pueda.
- Evitar demasiadas variables temporales.
- Asegurarse que los comentarios concuerden con el código escrito.
- Hacer pruebas tomando en cuenta valores límites.
- No parchar el código malo, reescribirlo.
- Utilizar nombres de variables significativos.
- Siempre inicializar las variables.
- Dar formato al código, para que a quien lo lea, le sea fácil entenderlo.

Como indica el científico Wilson Kernighan, el nombre de las variables debe ser significativo, para lo cual cabe mencionar las notaciones que han surgido para nombrar elementos utilizados especialmente en la programación de *software*.

En 1813, el químico sueco Friherre Jöns Jacob Berzelius, inventó la notación para fórmulas químicas, llamada *medial capitals*, que consiste en simbolizar cada elemento químico, con una o dos letras, escribiendo en mayúscula la primera de ellas. Como por ejemplo: MgO, óxido de magnesio.

En los años noventa, a la intercalación de letras mayúsculas y minúsculas en palabras compuestas, se le llamó notación *CamelCase,* porque a lo largo de la palabra, las letras mayúsculas se asemejan a las jorobas de un camello (c*amel* significa camello en el idioma inglés; c*ase* no tiene traducción al español, pero se refiere a letras mayúsculas o minúsculas).

Desde la década de los 70 su uso se expandió a los lenguajes de programación, como una convención para nombrar a los identificadores de los elementos del *software*. Siendo actualmente recomendada esta práctica por desarrolladores de Pascal, Modula, Java y .NET, por ejemplo.

Luego de la explosión de la informática, en los años ochenta, se utilizó esta notación para nombrar marcas de productos y compañías, como iMac en 1998.

A raíz de esta notación, surge *PascalCase*, cuya diferencia con la notación *CamelCase* es que la primera letra se coloca en mayúscula (PosFila), mientras que en *CamelCase*, se escribe en minúscula (posFila), y en ambas notaciones, todas las primeras letras de las siguientes palabras se escriben en mayúscula.

Años más tarde, Charles Simonyi, ingeniero de *software*, originario de Hungría, inventó la Notación Húngara, ahora llamada Aplicación Húngara, estándar que formó parte de su tesis doctoral en ciencias de la computación, en 1977. La cual consiste en nombrar a las variables, anteponiéndoles como

prefijo el propósito de la misma (posFila: posición de una fila); no el tipo de dato de la variable, como lo sugiere la notación llamada Sistema Húngaro (iPos: variable de tipo entero que indica una posición).

En 1981, cuando Simonyi fue contratado directamente por Bill Gates para trabajar en *Microsoft*® y dejó de laborar en Xerox, se empezó a utilizar en la compañía del empresario Gates, la Aplicación Húngara, y luego el Sistema Húngaro. Este último fue utilizado en *Microsoft*® hasta la liberación del primer .Net framework (febrero 2002), ya que desde entonces, dicha compañía ha sugerido no utilizar el sistema húngaro, debido a lo poco popular del mismo a causa del surgimiento de los IDEs (*Integrated development environments*), los cuales muestran el tipo de variable y alertan al usuario de operaciones que usan tipos de datos incompatibles.

Entre los trabajos más recientes de estándares de programación, se puede mencionar que en 1997, Sun publica un documento de convenios de código para el lenguaje JAVA; por otra parte, *Microsoft*® ha publicado guías para programar utilizando el .Net framework, en sus diferentes versiones. También cabe mencionar que la actualización más reciente realizada a los estándares de programación del proyecto GNU (proyecto que promueve el *software* no propietario), fue hecha en junio 2009.

1.2. Situación actual

Actualmente, los modelos para aseguramiento de la calidad, reconocidos a nivel internacional son: CMMI, ISO 9001 e ISO 15504 - SPICE (*Software Process Improvement and Capability dEtermination*). Estos modelos se enfocan en la mejora y evaluación de procesos para desarrollo de *software*, cuya aplicación ha ido en aumento exponencial, debido a la visión que han

tenido las empresas de ampliar su mercado local, lo que las ha orillado a contar con un respaldo, como lo es una certificación reconocida internacionalmente, para ser aceptadas en países extranjeros.

Otro factor, que también ha influido es, contar con una guía práctica y eficiente que dicte la manera de realizar los procedimientos para alcanzar los objetivos de la organización, independientemente del personal que en ella se encuentre.

Aunque el resultado de aplicar estos modelos sea todo un éxito, desde el punto de vista sistémico, la sinergia resultante de las tareas propias de cada fase, no garantiza que cada subtarea, como lo es el caso de la programación, esté estandarizada, por lo que se hace necesario normalizar dentro de los procesos del ciclo de vida del *software*, la escritura de código.

Desde que la tecnología se volvió una necesidad para el eficiente procesamiento de la información, se ha librado una guerra de mercado, entre el *software* libre y el *software* propietario, que si bien es cierto, cada uno tiene su enfoque muy peculiar, pero los clientes por quienes compiten, son los mismos. Por lo que cada parte, está en la constante búsqueda de ofrecer no solamente un funcionamiento estable, sino un valor agregado como lo es la calidad, la cual en cualquier ámbito da la seguridad y la confianza, de que no fue producido empíricamente.

En el caso del *software* libre, seguir ciertas normas para escribir código, puede resultar contraproducente para algunas personas, que podrían decir que adoptar este tipo de metodologías, restringe la creatividad de cada individuo. Sin embargo, estas regularizaciones no pretenden reducir la autonomía de los desarrolladores, ya que su objetivo es trazar los límites que enmarcan las

prácticas ordenadas que aseguran la portabilidad y mantenimiento de los programas, alargando así la vida del *software*; ya que un sistema que no puede crecer, no durará mucho.

Por otro lado, la piratería es una mala práctica, que ataca directamente al *software* propietario, ocupando Guatemala el puesto 22 dentro de los países con más altos índices de piratería, según el sexto estudio anual de la BSA, publicado en mayo de 2009. Por ejemplo, si personas sin autorización reproducen exactamente determinada aplicación estandarizada, al aplicar técnicas de ingeniería inversa y obtener el código fuente que fue desarrollado utilizando estándares de programación, no sería tarea difícil apropiarse también del código fuente.

Claro está que para combatir esta mala práctica, los países deben promover leyes eficientes que protejan la propiedad intelectual, así como estrategias a nivel gubernamental que promuevan condiciones para que toda persona encuentre productos originales, los desee y pueda adquirir. Además, cada institución debe contar con profesionales que aporten soluciones para proteger las aplicaciones, de reproducciones no autorizadas.

1.3. Futuro de los estándares de programación

Políticas como la globalización, eliminan las barreras entre los países, lo cual afecta a las pequeñas y medianas empresas que generalmente son las que no cuentan con el nivel para competir internacionalmente; lo que promueve que éstas busquen y practiquen técnicas que les otorguen el reconocimiento global, siendo unas de estas técnicas, los estándares de programación.

Por la misma competencia global, todas las empresas, incluyendo las productoras de *software*, están presionadas a incrementar su eficiencia a bajo costo, por lo cual la rotación de personal no debe influir en el presupuesto establecido periódicamente. Es decir, que un producto de *software* pueda ser desarrollado por cualquier analista-programador, luego de tener claro el proceso, y que descifrar la manera en la que está escrito el programa no sea toda una odisea que implique costos como el retraso en la entrega o incumplimiento de los requerimientos.

Al hacer la analogía con la industria textil del siglo XXI, las empresas productoras de *software*, deben evitar convertirse en maquilas de *software*, en donde lo que importa es producir la mayor cantidad posible al más bajo costo, sin incorporar la calidad de la materia prima y del mismo producto final; sino más bien, desarrollar con un enfoque ingenieril, adoptando métodos formales para iniciar, desarrollar y administrar los proyectos.

Como se puede apreciar, las publicaciones y estándares descritos anteriormente, son reglas muy generales, o son específicas para un lenguaje de programación, o bien solamente indican cómo nombrar los objetos utilizados en los programas, por lo que en el capítulo tres del presente trabajo, se presenta la metodología Extilo Regular 9002, que toma en cuenta las mejores prácticas de programación y diversos lenguajes, para proponer una guía de estándares de escritura de código, aplicable en la mayoría de lenguajes.

2. IMPORTANCIA DE LA ESTANDARIZACIÓN

El que una organización cuente con estándares, ya sea los reconocidos mundialmente como ISO 9001:2000, CMMI, ó con estándares propios, genera confianza, tanto para incorporarse en el mercado internacional del *software* como para atraer inversión extranjera. Pero para lograrlo, se requiere del apoyo gerencial, para proporcionar los recursos físicos, financieros y humanos, requeridos para tal fin. De igual manera teniendo los recursos, también se requiere de concientizar y lograr que todo el personal comprenda la importancia de utilizar estándares y el beneficio obtenido.

En la publicación del 2003 titulada *A methodology for self-diagnosis for software quality assurance in small and medium-sized industries in Latin America*, del periódico electrónico de sistemas de información en países en desarrollo, EJSDC, los autores exponen que la mayoría de empresas de *software* alrededor del mundo, no ha implementado una metodología de aseguramiento de la calidad del *software*, lo cual les ha traído, altos costos de producción y de mantenimiento de sistemas, por lo que han sido desplazados del mercado internacional, al no tener el mismo nivel de competitividad, que las empresas que sí tienen un método de aseguramiento de la calidad.

Hay que tomar en cuenta que, la implementación de estándares de programación en las empresas, independientemente de su tamaño, puede encontrar obstáculos culturales, como la resistencia al cambio, si el personal no ve los beneficios de adoptar esta metodología, si no que por el contrario la percibe como trabajo extra y burocrático que no se logra apreciar de modo tangible en el producto final. En las empresas pequeñas, el bajo presupuesto y

los tiempos ajustados de desarrollo, también pueden ser un obstáculo, al momento de implementar estándares de programación.

Por lo que no hay que perder de vista, que los beneficios pueden no ser a corto plazo, pero la empresa se asegura que en cualquier momento, cualquier persona del equipo de desarrollo, pueda modificar el sistema desarrollado; que la administración del proyecto se simplifique; que el cliente esté satisfecho y tranquilo de tener un *software* que no solo cumple con sus requerimientos, si no que fue desarrollado con estándares, obteniendo así un valor agregado en el producto.

A continuación se exponen las ventajas que brinda la estandarización de los programas de *software*, así como un par de desventajas que se deben tomar en cuenta, pero incluso a pesar de ellas y del costo que implica implementar una metodología de aseguramiento de calidad, pesan menos que las grandes ventajas que comprende esta normalización.

2.1. Ventajas de la estandarización

Facilita el mantenimiento del sistema. Difícilmente el mantenimiento de toda la vida de cualquier *software*, es realizado por el autor original, así como es inevitable no darle mantenimiento al código del programa, ya sea para ampliar su funcionamiento o por cambios en los procesos del negocio; por lo que establecer apropiadamente un formato y escritura uniforme, facilitan el mantenimiento de cualquier aplicación.

Contribuye al ahorro de recursos destinados al mantenimiento del *software*: tiempo y personal. Si el código desarrollado cuenta con la debida documentación, aplicación de sangría y estandarización, se torna fácil la

interpretación y reutilización del código desarrollado; optimizando el tiempo de desarrollo y recurso humano, destinado para escribir código nuevo, modificar el existente o integrarlo con otros sistemas. Esto beneficia a toda empresa que cuente con un sistema informático, independientemente de si la organización cuenta con su propio Departamento de Desarrollo, o si subcontrata a otra empresa para que le desarrolle; ya que siempre tendrá un costo.

Coadyuva a la calidad del sistema. La calidad del producto final de *software*, depende de la calidad de los procesos utilizados para desarrollar y mantener el sistema; por lo que utilizar estándares en los lenguajes de programación, desde el inicio de la fase de Programación, contribuye y promueve la utilización de estándares y normas de calidad a lo largo del ciclo de vida del *software*, independientemente de la metodología empleada.

Contribuye a la elaboración de la documentación técnica. Los estándares de programación también facilitan el entendimiento de la aplicación, al personal que realice la documentación técnica, ya que el código es más entendible para documentar la lógica del programa y realizar diagramas de flujo, sin tener que destinar a uno o varios analistas-programadores, tiempo completo para apoyar a los documentadores.

Ayuda en la optimización del código sin alterar su funcionamiento. Facilitando la reingeniería para mejorar la estructura interna, con el objeto de optimizar características internas del programa: mejorar su desempeño y extensibilidad, simplificar su estructura, cambiar o agregar un nuevo paradigma, o bien perfeccionar la lectura del código.

Beneficia la migración del sistema a nuevas versiones en las herramientas de desarrollo y manejadores de base de datos. Al cambiar la herramienta de

desarrollo, la estructura interna estandarizada facilita el trabajo de las personas encargadas de migrar de una versión a otra, para que los sistemas se enriquezcan con las nuevas funcionalidades que incorpora cada versión y así no se queden obsoletos.

Simplifica el diseño de los casos de prueba de la estructura interna del programa. Si la metodología utilizada para el desarrollo del *software* incluye pruebas de caja blanca, el producto de *software* estandarizado permitirá examinar y revisar el *software* de manera completa y eficiente, porque el código será significativamente más legible, que uno que no cuenta con regulaciones.

Mejora la lectura del programa. Contar con convenciones de código, permite al personal del equipo de desarrollo que interactuará con él, comprender el código más rápidamente; evitando así que desde el inicio, el analista-programador se haga la idea de que entender el código escrito por otra persona, será una tarea ardua y más aún darle mantenimiento.

2.2. Desventajas de la estandarización

Compromete la propiedad intelectual del código fuente. La portabilidad que provee un programa estandarizado, para un *software* propietario que por vulnerabilidades en la seguridad perimetral, llegue a manos de personas no autorizadas, es un gran riesgo, ya que podrían leer, modificar y distribuir el código, por ser un programa fácil de entender. Para evitar llegar hasta este punto, se requieren medidas a nivel de *hardware* y/o *software*, que las organizaciones deben practicar, no solamente para proteger sus aplicaciones, sino para proteger la información que se maneja por medio de ellas; tema que queda fuera del alcance del contexto de este trabajo de graduación.

Tiende a ser obsoleta. Si los estándares permanecen vigentes durante largos períodos de tiempo, pueden llegar a conservar características obsoletas; por lo que se requiere hacer revisiones periódicas, para innovar y actualizar dichas regulaciones.

Restringe la creatividad del analista – programador. Algunas personas del medio informático, opinan que seguir estas guías, limitan la creatividad de los programadores, pero hay que recordar que son lineamientos que promueven un orden en cada proceso, dirigiendo así al producto de *software* hacia la calidad. Es decir que, se requiere que el programador cumpla las normas de estandarización al programar, y utilice su análisis y creatividad, para que el código escrito cumpla con los objetivos esperados.

3. METODOLOGÍA PARA ESTÁNDARES DE PROGRAMACIÓN DE *SOFTWARE* EN GUATEMALA: EXTILO REGULAR 9002

Los entornos de desarrollo integrados (IDEs) permiten consultar mediante un solo clic, la definición de los objetos de una aplicación, es decir, su tipo, valor, propiedades, métodos, parámetros, entre otras características propias de cada elemento.

Y como se indicaba en el primer capítulo, por esta razón es que ya no se recomienda el uso del sistema húngaro; pero no hay que perder de vista que los IDEs aplican principalmente para los lenguajes estáticamente tipificados, es decir para los lenguajes que chequean los tipos de datos del código escrito en tiempo de compilación, contrario a los lenguajes dinámicamente tipificados que hacen la verificación en tiempo de ejecución, y por lo tanto éstos últimos no facilitan obtener la información de los objetos utilizados.

Por lo cual, la metodología que aquí se sugiere ayuda enormemente al contar con un estándar para escribir código, teniendo ventajas muy positivas para el Equipo de Desarrollo, e incluso para la misma Gerencia por ser el departamento que distribuye el recurso dentro de la organización; destacando dentro de las ventajas expuestas en el capítulo anterior, la calidad y el fácil mantenimiento del sistema.

En este capítulo se presenta la metodología para escribir código, la cual promueve la incorporación de los tipos, en el nombre de cada objeto, ya que dentro de los lenguajes mencionados en Extilo Regular 9002, PHP y *JavaScript*

son dinámicamente tipificados. Además, si en una variable de tipo cadena se almacena código que posteriormente será ejecutado con la instrucción EXEC de *Microsoft* SQL, o bien código HTML construido del lado del servidor para que sea interpretado del lado del cliente, por medio de una página *web*; el *IDE* no mostrará información de los tipos de datos de los elementos utilizados, pero el nombre de ellos sí indicará esta información.

Si los Equipos de Desarrollo no cuentan con los recursos para instalar un *IDE* que indique toda la información de los objetos declarados en el proyecto, o bien si no existen estos entornos de desarrollo para los lenguajes a utilizar, la notación expuesta en Extilo Regular 9002, proveerá esta información además de mantener estandarizado el código.

Aunque la metodología está basada en los lenguajes de programación y sistemas de gestión de base de datos más utilizados para producir *software* en Guatemala: Visual Basic .Net, Visual C#, PHP, Java, C++, Oracle, *Microsoft* SQL, MySql y PostgreSQL, también es aplicable en el desarrollo de *software* empleando otros lenguajes de programación, independientemente del idioma y de la herramienta de desarrollo utilizada.

3.1. Archivos de la aplicación

Una aplicación interpretada o compilada, dependiendo del lenguaje de programación utilizado para desarrollarla, está conformada por varios archivos, que pueden ser DLL (*Dynamic Link Library*), imágenes, hojas de estilo (*CSS: Cascading Style Sheets*), clases, formas, servicios *web*, librerías, páginas HTML, ASP, ASPX, PHP, y los archivos con las extensiones propias de cada herramienta de desarrollo; la cantidad de archivos puede variar dependiendo del grado de modularidad del proyecto, por lo cual se hace necesario:

- Nombrarlos anteponiéndoles como prefijo el tipo de archivo que representa dentro de la aplicación.
- Organizarlos dentro del proyecto, con el objeto de ubicar los archivos dentro de carpetas como si se estuviera organizando un archivo de expedientes, lo cual facilita el mantenimiento del sistema y refleja orden.

Para este cometido se sugiere nombrar y organizar los archivos de la siguiente manera:

Figura 1. **Carpeta del proyecto**

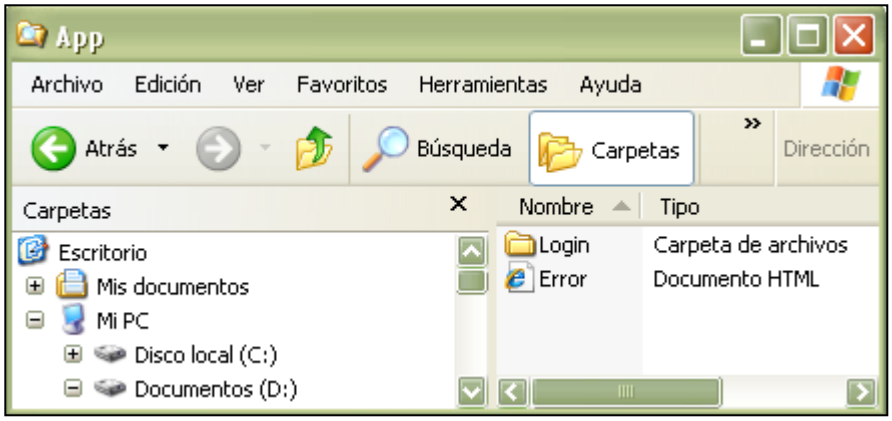
Fuente: elaboración propia.

Crear una carpeta para el proyecto con el nombre del mismo (ver figura 1), de preferencia en una partición diferente a la del sistema operativo; esto ayudará a que si éste se daña, los documentos de la partición que contiene el proyecto podrán recuperarse.

Dentro de la carpeta del proyecto, crear las carpetas que contendrán los diferentes archivos descritos anteriormente, con el objeto de separar en carpetas las imágenes de las librerías utilizadas, por ejemplo.

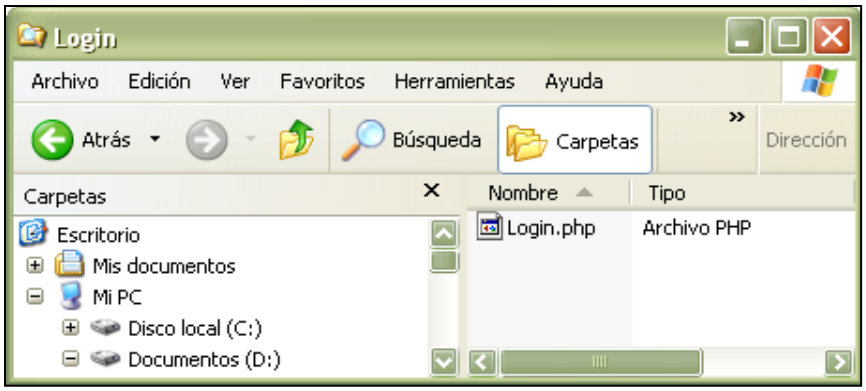
Agrupar los archivos principales de la aplicación por módulos, por ejemplo, un módulo de inicio de sesión, un módulo de seguridad, y los que apliquen dependiendo del tipo de aplicación a desarrollar, tal como se indica en las siguientes dos figuras.

Figura 2. **Archivos de la aplicación**



Fuente: elaboración propia.

Figura 3. **Archivo dentro de un módulo de la aplicación**



Fuente: elaboración propia.

A los archivos dentro de cada módulo, puede anteponérseles como prefijo la abreviatura del propósito de los mismos, para diferenciar un archivo que contiene un reporte de uno que es utilizado para darle mantenimiento a los datos, dentro del mismo módulo.

Siguiendo esta notación, dentro del módulo Login del ejemplo, aparte de la página Login.php, puede tenerse otro archivo repLogin.php; lo que ayuda a identificar rápidamente que repLogin.php contiene el código para generar el reporte del módulo, y que Login.php fue desarrollado para gestionar las credenciales de los usuarios.

Las hojas de estilo, imágenes, servicios *web*, clases, *scripts*, y librerías, son algunos de los archivos auxiliares que puede utilizar una aplicación, los cuales pueden nombrarse y agruparse como se indica en las figuras 4-7.

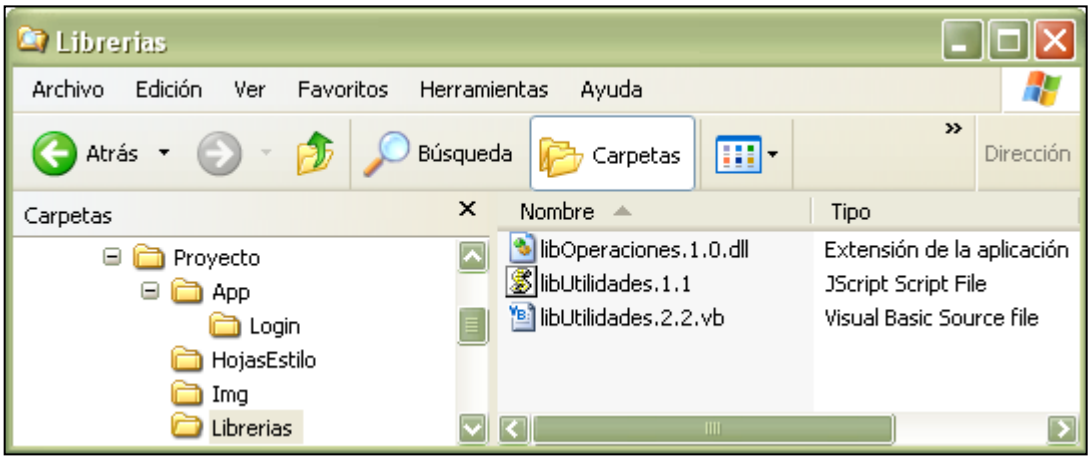
Figura 4. **Hojas de estilo**

Fuente: elaboración propia.

Figura 5. **Imágenes**

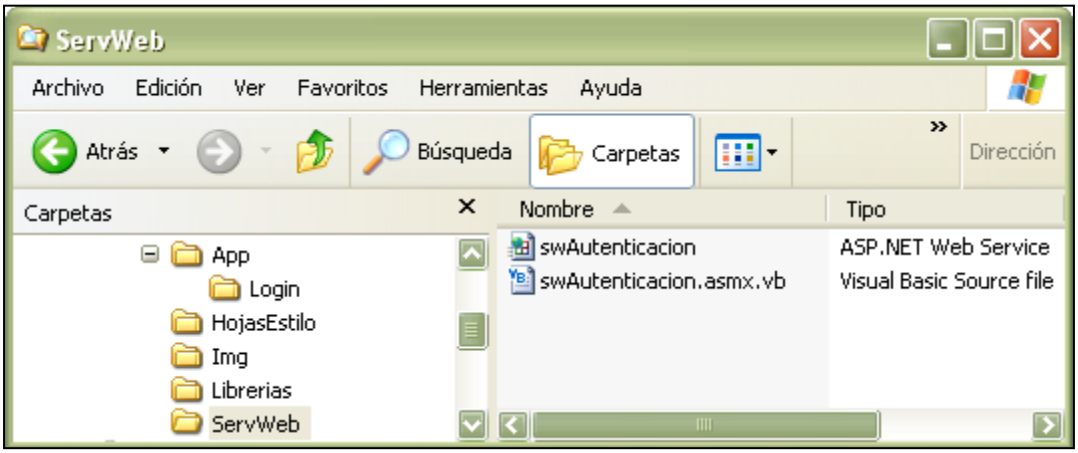
Fuente: elaboración propia.

Figura 6. **Librerías**



Fuente: elaboración propia.

Figura 7. **Servicios *web***



Fuente: elaboración propia.

Dentro de cada archivo se escribe el código siguiendo el esquema básico: inicialización, desarrollo y finalización; en donde primero se incluyen las librerías, y se declaran e inicializan todas las variables y constantes a utilizar

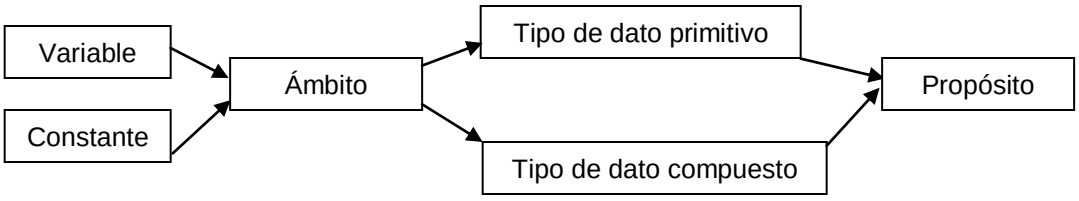
dentro del programa; luego se desarrolla el programa, por medio de subrutinas, que a su vez utilizan estructuras de control, clases y manejo de errores para indicarle al lenguaje lo que debe realizar. Por último, se liberan los recursos utilizados, por ejemplo liberación de memoria, cierre de archivos, de conexiones a la base de datos, entre otros.

A continuación se indica cómo nombrar a las variables, constantes, funciones y procedimientos escritos dentro del código fuente de la aplicación.

3.2. Nombres de variables y constantes

En un programa independientemente de su tamaño, siempre se utilizarán variables y regularmente constantes; cuya declaración implica darles un nombre para luego hacer referencia a ellas en diferentes partes del código. Por lo que al nombrar una variable o constante, se recomienda utilizar la notación *CamelCase*, combinada con el Sistema Húngaro y la Aplicación Húngara. Es decir, nombrarlas anteponiendo el prefijo que indique si es una variable o una constante, seguido del ámbito de la misma, luego el prefijo del tipo de dato, y por último el propósito de la misma:

Figura 8. **Notación para nombrar variables y constantes**



Fuente: elaboración propia.

Los prefijos para indicar si es una variable o una constante son:

Variable --> vr
Constante --> ct

Si el ámbito es local (la variable o constante es utilizada en una porción de código definida), emplear el prefijo L; si se accede a ella desde cualquier parte del programa (ámbito global), utilizar el prefijo G.

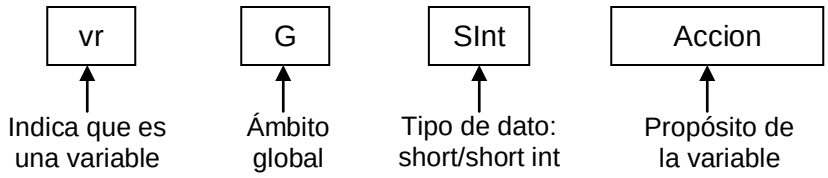
Si al almacenar los nombres de variables, constantes, subrutinas, valores de retorno, parámetros, clases, sus objetos, atributos y métodos, objetos de la base de datos, columnas de tablas y demás elementos que puedan nombrarse, utilizando el estándar expuesto en el presente trabajo, el lenguaje de programación solamente maneja mayúsculas ó minúsculas, ó bien permite la distinción entre mayúsculas y minúsculas utilizando determinada notación (usando “” para delimitar el nombre, como lo permite Oracle), se sugiere sustituir el uso de la notación *CamelCase* por el uso de minúsculas (*lower case*) y el carácter _, para que visualmente se distinga la abreviatura que denota cada parte dentro del nombre.

Por ejemplo, en Oracle para no utilizar las comillas (“”) al nombrar una tabla de usuarios, se nombraría de la siguiente manera: tbl_seg_usuario.

Cada herramienta de desarrollo tiene definidos sus tipos de datos, por lo que las tablas I y II recopilan los principales tipos de datos, y el prefijo a utilizar en cada uno de estos tipos, para los lenguajes de programación y sistemas de gestión de base de datos, en los que está basado esta metodología (Visual Basic .Net, Visual C#, PHP, Java, C++, Oracle, *Microsoft* SQL, MySql y PostgreSQL).

Por ejemplo, si se requiere de una variable global que cambie su valor dependiendo de la acción que realice el usuario desde la aplicación: ingresar un nuevo cliente o actualizar los datos de uno ya existente, y que dicho valor se evalúe posteriormente para aplicar un descuento si la acción es modificar, la misma se nombra de la siguiente manera:

Figura 9. **Ejemplo de nombre para variable global**



Fuente: elaboración propia.

Tabla I. **Prefijos de los tipos de datos primitivos utilizados en los lenguajes de programación: Visual Basic .Net, Visual C#, PHP, Java y C++**

Tipo de dato primitivo del lenguaje de programación	Prefijo	Descripción	Lenguaje de programación que implementa el tipo de dato				
			Visual Basic .Net	Visual C#	PHP	Java	C++
Array/Vector/Matriz	v	Vector de 1 o más dimensiones			x	x	
Boolean/Bool	bol	Valor de lógica binaria	x	x		x	x
Byte	bt	Número entero de 8 bits, sin signo	x	x		x	
Char	ch	Un carácter	x	x		x	x
Date	dt	Fecha	x				
Datetime/time_t	dtt	Fecha y hora	x	x			x
Decimal	dec	Decimal exacto	x	x			
Double/Punto flotante	db	Decimal aproximado de precisión doble con signo	x	x	x	x	x

Continuación de la tabla I.

Tipo de dato primitivo del lenguaje de programación	Prefijo	Descripción	Lenguaje de programación que implementa el tipo de dato				
			Visual Basic .Net	Visual C#	PHP	Java	C++
Enum	enm	Colección de valores		x		x	x
Integer/Int/Entero	int	Número entero de 32 bits, con signo	x	x	x	x	x
Long	lng	Número entero de 64 bits, con signo	x	x		x	x
Long double	lDb	Decimal aproximado largo, de precisión doble con signo					x
Object/Objeto	obj	Puede asumir cualquier tipo de dato	x		x		
Sbyte	sBt	Número entero de 8 bits, con signo	x	x			
Short/Short int	sInt	Número entero de 16 bits, con signo	x	x		x	x
Signed char	sCh	Número entero de 8 bits, con signo					x
Single/Float	flt	Decimal aproximado de precisión simple con signo	x	x		x	x
String/Cadena	str	Colección de carácteres	x	x	x	x	
UInteger/UInt	uInt	Número entero de 32 bits, sin signo	x	x			x
Ulong/Unsigned long	ulng	Número entero de 64 bits, sin signo	x	x			x
Unsigned char	uCh	Número entero de 8 bits, sin signo					x
User-Defined	uDef	Estructura definida por el usuario	x				
Ushort/Unsigned short	uSt	Número entero de 16 bits, sin signo	x	x			x
Void	ptr	Tipo de dato puntero					x

Fuente: elaboración propia.

Tabla II. **Prefijos de los tipos de datos primitivos utilizados en los sistemas de gestión de base de datos (SGBD): Oracle, *Microsoft* SQL, MySql y PostgreSQL**

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
Anydata/sql_variant	aDt	Colección de datos de cualquier tipo, con la descripción del tipo de cada dato	x	x		
Anydataset	aDst	Colección de datos de cualquier tipo	x			
Anytype	aTp	Colección de descripciones de tipos de datos	x			
BFile	bFl	Almacena la ubicación de un archivo que se encuentra fuera de la base de datos	x			
Bigint	bInt	Número entero de 64 bits		x	x	x
Bigserial	bSrl	Bigint autoincrementable				x
Binary/Bytea	bny	Cadena binaria		x	x	x
Binary_double/Double/Double precision	db	Decimal aproximado de precisión doble	x		x	x
Binary_float/Float	flt	Decimal aproximado de precisión simple	x	x	x	
Bit	bit	Entero que puede tener los valores 1, 0 o NULL; o cadena de bits de longitud fija		x	x	x
Bit varying (n)	vbit	Cadena de bits de longitud variable				x
BLob	bLob	Almacena datos binarios, se puede considerar como VARBINARY	x	x	x	
Boolean	bol	Tipo enumerado predefinido que puede tener valores TRUE o FALSE				x

Continuación de la tabla II.

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
Box	bx	Tipo de dato geométrico que define un rectángulo				x
Char/Character	ch	Cadena de carácteres no UNICODE de longitud fija	x	x	x	x
Cidr	nadr	Dirección de red, IPv4 o IPv6				x
Circle	cl	Tipo de dato geométrico que define un círculo				x
CLob	cLob	Almacena carácteres en el formato definido para la base de datos	x			
Cursor	crs	Tipo de dato cuyo contenido son los registros devueltos por una consulta	x	x	x	x
Date	dt	Fecha, sin zona horaria	x	x	x	x
Datetime	dtt	Fecha y hora, sin zona horaria		x	x	
Datetime2	dtt2	Extensión del tipo Datetime, incluye precisión en la fracción de segundos		x		
Datetimeoffset	dtOSet	Fecha y hora con zona horaria, incluye precisión en la fracción de segundos		x		
DBURIType	dbUri	Hace referencia a información dentro o fuera de la BDD, por medio de DBURIRef (XML Path)	x			
Decimal	dec	Decimal aproximado de precisión simple		x	x	x
Enum	enm	Cadena con un valor obtenido de una lista de valores definidos explícitamente			x	
GeometryCollection	geoc	Colección de objetos geométricos (datos en un modelo plano) o geográficos (datos en un modelo elíptico)		x	x	
Hierarchyid	hId	Representa la posición en una estructura jerárquica		x		

Continuación de la tabla II.

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
HTTPURIType	uri	Almacena direcciones URL de páginas web externas o archivos	x			
Image	img	Datos binarios de longitud variable		x		
Inet	hadr	Dirección de red o de host, IPv4 o IPv6				x
Int /Integer	int	Número entero de 32 bits		x	x	x
Interval	ivl	Lapso de tiempo				x
Interval day to second	ivlDS	Periodo de tiempo especificado en días, horas, minutos y segundos	x			
Interval year to month	ivlYM	Periodo de tiempo especificado en años y meses	x			
Line	ln	Línea infinita en el plano (en proceso de implementación)				x
LongBlob	lBLob	Tipo BLOB que almacena (2^32)-1 carácteres			x	
LongText	lTxt	Tipo TEXT que almacena (2^32)-1 carácteres			x	
Lseg/Linestring	lSeg	Tipo de dato geométrico que define una línea		x	x	x
Macaddr	madr	Dirección MAC				x
MediumBlob	mBLob	Tipo BLOB que almacena (2^24)-1 carácteres			x	
Mediumint	mInt	Número entero de 24 bits			x	
MediumText	mTxt	Tipo TEXT que almacena (2^24)-1 carácteres			x	
Money	mon	Valor monetario de 8 bytes		x		x
MultiLineString	mln	Colección de líneas (LINESTRING)		x	x	
MultiPoint	mpnt	Colección de puntos (POINT) no conectados		x	x	
MultiPolygon	mply	Colección de polígonos (POLYGON)		x	x	
Nchar	nCh	Cadena de carácteres UNICODE de longitud fija	x	x		

Continuación de la tabla II.

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
NClob	nCLob	Almacena carácteres en el formato UNICODE	x			
Nested table/table	vtbl	Tipo tabla, almacena registros	x	x		
NText	nTxt	Almacena carácteres UNICODE de longitud variable		x		
Number	nm	Número entero o decimal aproximado de precisión simple	x			
Numeric	nmc	Número entero o decimal aproximado de precisión simple		x	x	x
NVarchar2/Nvarchar	nVch2/nVch	Cadena de carácteres UNICODE de longitud variable	x	x		
ORDAudio	aud	Admite el almacenamiento y gestión de datos de audio	x			
ORDDicom	dicom	Admite el almacenamiento y gestión de imágenes médicas en formato DICOM	x			
ORDDoc	doc	Admite el almacenamiento y gestión de cualquier información multimedia: audio, imágenes o video	x			
ORDImage	dImg	Admite el almacenamiento y gestión de imágenes	x			
ORDVideo	vdo	Admite el almacenamiento y gestión de videos	x			
Path	pth	Tipo de dato geométrico que contiene varios puntos conectados (POINT); el primer y el último punto, pueden o no estar conectados				x
Point	pnt	Tipo de dato geométrico que define un punto		x	x	x

Continuación de la tabla II.

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
Polygon	ply	Tipo de dato geométrico que define un polígono (PATH cerrado)		x	x	x
Real	rl	Decimal aproximado de precisión simple, de 4 bytes		x	x	x
REF	ref	Apuntador a un objeto de la base de datos	x			
Rowversion	rVer	Lleva el control de la versión de las filas de una tabla		x		
SDO_Geometry	geom	Almacena la información geométrica de un objeto	x			
SDO_GeoRaster	geo	Almacena la imagen de un objeto geométrico	x			
SDO_Topo_Geometry	tGeom	Describe la topología de un objeto geométrico	x			
Serial	srl	Smallint autoincrementable				x
Set	st	Cadena con cero o más valores, obtenidos de una lista de valores definidos explícitamente			x	
SI_AverageColor	avgClr	Describe el color promedio que caracteriza a una imagen	x			
SI_Color	clr	Representa los valores de color de una imagen (valor RGB)	x			
SI_ColorHistogram	clrHist	Describe las frecuencias relativas de los colores de la porción de una imagen	x			
SI_FeatureList	fLst	Contiene hasta 4 características básicas de la imagen (SI_AverageColor, SI_ColorHistogram, SI_PositionalColor y SI_Texture), y su respectiva importancia, cuando se comparan imágenes	x			

Continuación de la tabla II.

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
SI_PositionalColor	posClr	Representa los colores más significantes en determinada posición de una imagen	x			
SI_StillImage	sImg	Representa imágenes digitales, con características como altura, ancho y formato	x			
SI_Texture	txtr	Describe la textura de una imagen	x			
Smalldatetime	sDt	Fecha y hora, de menor rango que el tipo Datetime		x		
Smallint	sInt	Número entero de 16 bits		x	x	x
Smallmoney	sMon	Valor monetario de 4 bytes		x		
Text	txt	Almacena carácteres no UNICODE, se puede considerar como VARCHAR		x	x	x
Time (without time zone)	t	Almacena una hora del día sin zona horaria		x	x	x
Time with time zone	tTZ	Almacena una hora del día con zona horaria				x
Timestamp (without time zone)	tStp	Extensión del tipo Datetime, incluye precisión en la fracción de segundos	x		x	x
Timestamp with local time zone	tStpLTZ	Fecha y hora con zona horaria de la base de datos, incluye precisión en la fracción de segundos	x			
Timestamp with time zone	tStpTZ	Fecha y hora con zona horaria, incluye precisión en la fracción de segundos	x			x
Tinyblob	tBLob	Tipo BLOB que almacena (2^8)-1 carácteres			x	
TinyInt	tInt	Número entero de 8 bits		x	x	
Tinytext	tTxt	Tipo TEXT que almacena (2^8)-1 carácteres			x	
Uniqueidentifier	uId	GUID de 16 bytes		x		
Varbinary	vBny	Cadena binaria de longitud variable		x	x	

Continuación de la tabla II.

Tipo de dato primitivo del sistema de gestión de base de datos	Prefijo	Descripción	SGBD que implementa el tipo de dato			
			Oracle	SQL	MySQL	PostgreSQL
Varchar/Character varying	vch	Cadena de carácteres no UNICODE de longitud variable		x	x	x
Varchar2	vch2	Cadena de carácteres de longitud variable	x			
Varrays/Arrays	v	Arreglo, conjunto ordenado de elementos del mismo tipo	x			x
XDBURIType	xDbUri	URL que contiene el nombre del documento XML, seguido de algún path que extrae un fragmento de información (XML Path)	x			
XMLType/xml	xml	Contiene información XML	x	x		
Year	y	Almacena los años 0, y 1901 hasta 2155			x	

Fuente: elaboración propia.

En *Microsoft* SQL no es posible declarar variables globales, debido a que las variables de este tipo, ya están previamente definidas con los carácteres @@ antepuestos al nombre de las mismas. Por ejemplo, @@ERROR, es la variable que devuelve el último número de error del sistema.

PostgreSQL tampoco provee el manejo de variables globales, pero esto podría hacerse por medio de los lenguajes procedurales que manejan almacenamiento global de información, disponibles en la distribución estándar de PostgreSQL, como por ejemplo: PL/Perl, PL/Tcl y PL/Python.

En Oracle tampoco hay manera de declarar variables globales; aunque se pueden declarar en la sección de declaración de un paquete, pero a menos que al acceder o modificar el valor de la variable, se haga referencia al paquete (NombrePaquete.NombreVariable), el ámbito de la variable solamente será dentro del paquete. Otra limitante es que, al modificar el valor de la supuesta variable global, el cambio aplicará solamente en la sesión actual; si se accede a la variable desde otra parte del código, el cambio no se verá reflejado, a menos que se envíe como parámetro.

Las variables globales que maneja MySQL se inicializan cada vez que se inicia el servidor, y solamente pueden ser modificadas si se cuenta con el privilegio SUPER. Se debe tomar en cuenta que los cambios en las variables globales serán visibles en las sesiones que se abran luego del cambio; estos no se verán reflejados en la sesión de los clientes conectados al momento del cambio, ni siquiera en la del cliente que modificó la variable global.

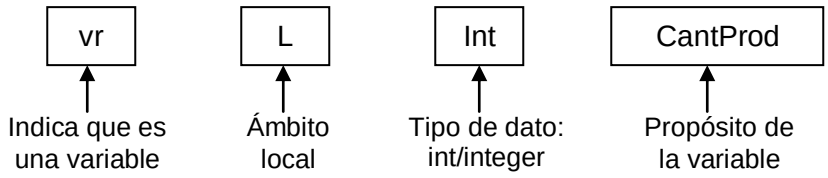
Tanto en *Microsoft* SQL, como en Oracle y PostgreSQL se puede implementar el uso de variables globales utilizando una tabla, que pueda ser modificada y consultada por cualquier usuario. El inconveniente es que para obtener el valor actual de la variable, se debe consultar constantemente el valor de la misma previo a utilizarla, para no perder los cambios entre la última vez que fue utilizada y el actual intento de uso.

El uso de variables globales no es recomendable, ya que su valor puede cambiarse desde cualquier parte del código, lo cual va en contra del principio de encapsulación de los datos; y contra los principios básicos de las pruebas unitarias, uno de los tipos de pruebas realizadas en las últimas fases del desarrollo de *software*, que consisten en aislar pequeñas partes del código para probar su correcto funcionamiento.

Al tener variables globales dentro de las pruebas unitarias, las porciones de código a probar aumentan, por lo que ya no se tendría una prueba unitaria sino una prueba de integración, porque se estarían probando paralelamente varias partes del código.

Para nombrar una variable local, que dentro de un procedimiento almacenado obtenga la cantidad de productos vendidos en el mes actual, por ejemplo, se haría de la siguiente manera:

Figura 10. **Ejemplo de nombre para variable local**

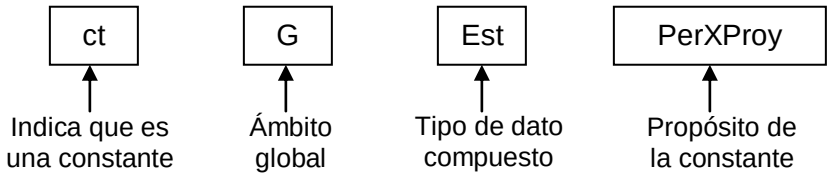


Fuente: elaboración propia.

Si la variable o constante no es de tipo de dato primitivo, sino de tipo de dato compuesto, es decir que es de un tipo creado a partir de tipos elementales y/o de otros tipos compuestos, utilizar el prefijo Est, luego de indicar el ámbito de la variable o constante.

Por ejemplo, si se requiere declarar una estructura en donde estén almacenados por proyecto, los datos personales de los miembros del equipo de desarrollo, el nombre de la misma sería:

Figura 11. **Ejemplo de nombre para constante global**



Fuente: elaboración propia.

El nombrar a las variables y constantes de esta manera, además de permitir identificar fácilmente si es una variable o una constante, el ámbito, el tipo de dato y el propósito de la misma, evitará que una variable o constante local, tenga el mismo nombre que una global; ya que cuando esto sucede, la local prevalece sobre la global dentro de la porción de código en donde fue definida, y si esto no se toma en cuenta al hacer referencia a dicha variable o constante, el sistema no funcionará como se espera porque realmente se estará utilizando el valor local y no el valor global.

Aparte de nombrar las variables y constantes con el estándar anterior, también se recomienda respetar el tipo de dato de la variable o constante, especialmente si el lenguaje de programación es débilmente tipado, como PHP, ya que al escribir el siguiente código:

Figura 12. **Ejemplo para nombrar variables en PHP**

```
0001  //Crear la variable de tipo entero que contiene el valor de un código
0002  $vrLIntCodigo = 1;
0003  …….
0004  //Se asigna una cadena de carácteres a la variable vrLIntCodigo
0005  $vrLIntCodigo = “Error”;
```

Fuente: elaboración propia.

no habrá error, porque se creará la variable $vrLIntCodigo, que primero será del tipo entero y luego de tipo cadena. Después si se hace referencia a la variable $vrLIntCodigo, pensando que su valor es 1, se cometerá un error que no será fácil de detectar.

Como otra buena práctica de programación, también siempre se debe inicializar las variables, evitando así que estas tengan valores nulos que provoquen errores en tiempo de ejecución.

Otra buena práctica es la programación *top-down*, estructurada o modular, la cual consiste en dividir el programa en bloques más pequeños de código, llamados subrutinas o subprogramas; que son independientes entre sí, resuelven una tarea específica y evitan la duplicación del código, logrando así:

- Reutilizar el código: los subprogramas solamente se escribirán una vez, aunque se utilicen en varias partes del programa.

- Facilitar la actualización del código: si se debe realizar alguna modificación, solamente se escribirá una vez, no se debe hacer en todas las partes donde aparezca el mismo código.

- Crear, probar y depurar las rutinas de manera independiente: cada subrutina se puede crear aisladamente y varios desarrolladores podrán trabajar simultáneamente en la elaboración de un programa, repartiéndose las distintas partes del mismo. De igual manera, se podrá modificar una subrutina sin afectar a las demás. Para lograrlo, es útil monitorizar si un subprograma ocupa varias páginas de código, y considerar si la subrutina debe subdividirse.

En el siguiente punto se indica cómo nombrar las subrutinas dentro de un programa, es decir cómo nombrar los procedimientos y funciones.

3.3. Nombres de procedimientos y funciones

Para declarar las subrutinas se requiere:

- Darles un nombre único, el cual debe contener un verbo en modo infinitivo. Si el lenguaje acepta sobrecarga o polimorfismo, no habrá error al declarar dos funciones con el mismo nombre que ejecuten diferentes sentencias, por ejemplo, porque se invocará la que corresponda; solamente hay que considerar que la depuración del programa no será totalmente transparente, porque la modificación del código tendrá que hacerse en todas las implementaciones que tenga una misma subrutina.

- Si es una función, se debe indicar un tipo de dato de retorno, que es el tipo de dato del valor que la subrutina devuelve al terminar su ejecución.

- La lista de parámetros, que es el conjunto de argumentos, opcionales, que la subrutina puede recibir para realizar su tarea.

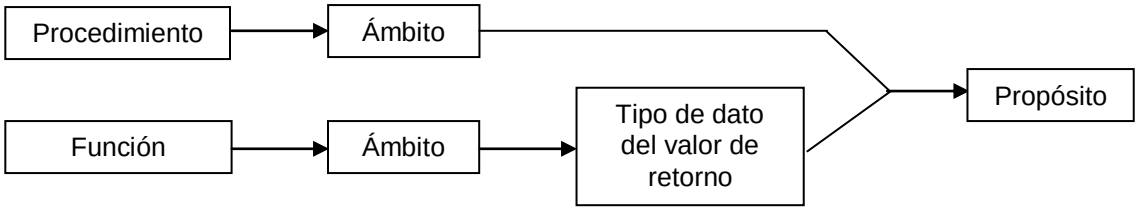
Entonces para nombrar a los procedimientos y funciones también se recomienda utilizar la notación *CamelCase* combinada con la Aplicación Húngara. Es decir que se antepone el prefijo que indique si es un procedimiento o una función, seguido del ámbito y luego el propósito, como se muestra en la figura 13. Si se trata de una función, después de indicar el ámbito de la misma, se debe indicar el tipo de dato del valor de retorno (Sistema Húngaro).

Los prefijos para indicar si el subprograma es un procedimiento o una función son:

Procedimiento --> pc

Función --> f

Figura 13. **Notación para nombrar procedimientos y funciones**



Fuente: elaboración propia.

Si el ámbito es público (la subrutina puede ser llamada desde cualquier parte del programa), emplear el prefijo Pb; si el ámbito es privado (se invoca solamente dentro del módulo en donde es declarada), utilizar el prefijo Pr.

En los SGBD, no se define el ámbito de una subrutina como público o privado, ya que por ser objetos de la base de datos, el acceso a ellos se restringe por los permisos otorgados a los usuarios (consulta, ejecución, modificación), o bien por el esquema (análogo a un módulo dentro de una aplicación) bajo el cual se crean.

Nombrar a las subrutinas de esta manera, permite a los desarrolladores diferenciar los procedimientos de las funciones, con sólo leer el nombre del subprograma, sin tener que ir a la definición del mismo.

A pesar que en lenguajes como C y C++ todas las subrutinas devuelven un valor, es aplicable nombrar de diferente manera a los procedimientos y funciones, porque cuando se declara un procedimiento el valor de retorno se indica con la palabra void, para señalar que no devuelve nada.

En cada subrutina, también se escribe el código siguiendo el esquema descrito al final del punto 3.1: inicialización, desarrollo y finalización; en donde primero se declaran e inicializan todas las variables y constantes a utilizar; luego se desarrolla cada subrutina, invocando a otras subrutinas, utilizando estructuras de control, clases, manejo de errores, etc. Por último, si es necesario se liberan los recursos utilizados: memoria, archivos, conexiones a la base de datos, por ejemplo.

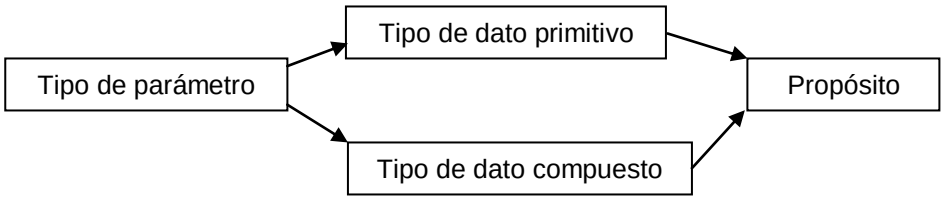
Si en una subrutina se requiere utilizar valores de variables o incluso modificar variables, que se encuentran en otra parte del programa y no están declaradas como variables globales (siguiendo la recomendación del punto 3.2); al invocar a un procedimiento o función se le pueden enviar estos datos por medio de parámetros, los cuales se describen en el siguiente punto.

3.3.1. Parámetros

Los parámetros se pueden concebir como variables que se declaran e inicializan al llamar a la subrutina, por lo que la transferencia de ellos a las funciones y procedimientos, puede ser por valor o por referencia. La diferencia entre una y otra es que, por valor el subprograma recibe una copia del valor que tiene la variable enviada por parámetro; por referencia, se recibe la dirección de memoria de la variable, lo que implica que cualquier modificación de la variable dentro del subprograma, se hace directamente en la posición de memoria de la misma.

Si la subrutina recibe parámetros, la siguiente figura indica cómo nombrarlos, para así diferenciar las variables de los parámetros recibidos dentro de la subrutina.

Figura 14. **Notación para nombrar parámetros**



Fuente: elaboración propia.

Los prefijos para indicar el tipo de parámetro son:

Por valor --> pV
Por referencia --> pR

Para los prefijos del tipo de dato, utilizar los indicados en las tablas I y II, siguiendo el mismo procedimiento definido para indicar el tipo de dato de las variables y constantes, en el apartado anterior.

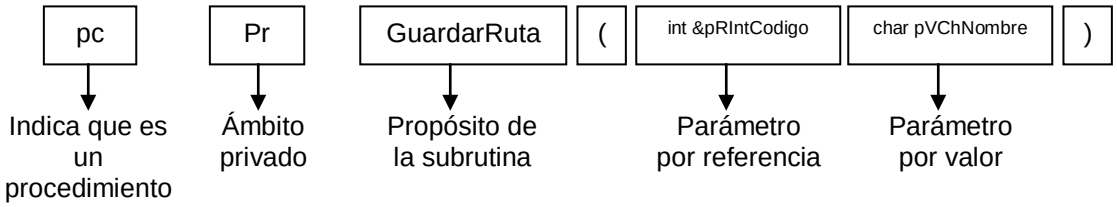
En esta metodología se indica como nombrar a los parámetros, pero depende del compilador del lenguaje de programación o del sistema de gestión de base de datos, la sintaxis requerida para indicar si el parámetro que recibirá la subrutina es por valor o por referencia.

En *Microsoft* SQL, a los parámetros por referencia se les pospone la palabra reservada OUTPUT.

Por ejemplo, en Visual Basic el mecanismo del argumento transferido se indica por medio de las palabras reservadas *ByVal* o *ByRef*; mientras que en C++, por defecto los parámetros se pasan por valor, si se requiere transferir un argumento por referencia se antepone el carácter & al nombre del mismo.

Por ejemplo, en un programa de C++ que administre recorridos de una empresa de servicios de entrega de paquetería, para nombrar un procedimiento que reciba los datos para almacenar las rutas, se realiza de la siguiente manera:

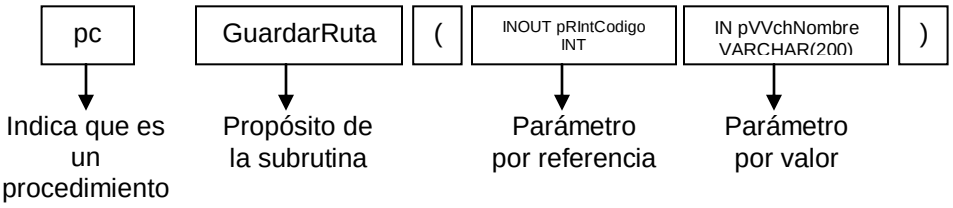
Figura 15. **Ejemplo de nombre para procedimiento en C++**



Fuente: elaboración propia.

El procedimiento almacenado de MySQL que se podría invocar desde la aplicación, para almacenar los datos de la ruta en la base de datos, se nombrará como sigue:

Figura 16. **Ejemplo de nombre para procedimiento en MySQL**



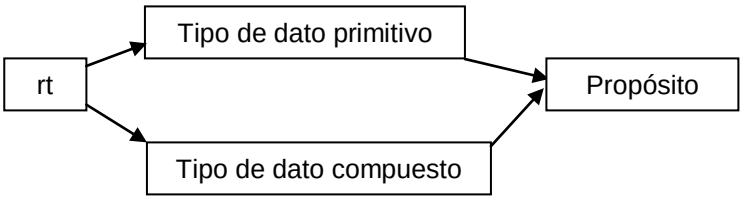
Fuente: elaboración propia.

Así como las subrutinas pueden recibir datos, la naturaleza de las funciones permite que por medio de éstas se devuelva un único resultado, llamado valor de retorno; por lo que a continuación se detalla cómo nombrar la variable que se devuelve. Nombrar de diferente manera al valor de retorno, permite diferenciarlo del resto de variables y parámetros, utilizados en la subrutina.

3.3.2. Valor de retorno

Los valores de retorno de las funciones, se sugiere nombrarlos como sigue:

Figura 17. **Notación para nombrar variables de retorno**

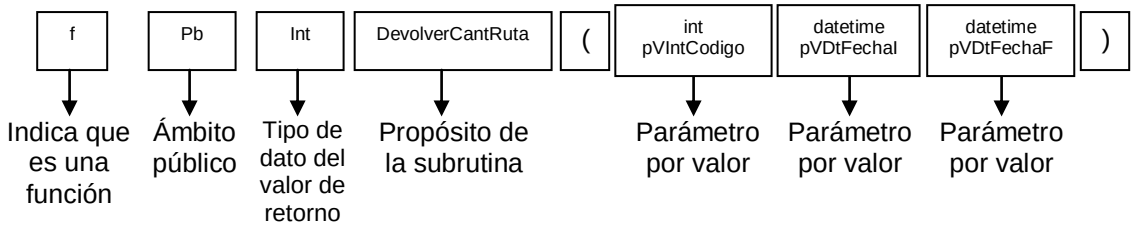


Fuente: elaboración propia.

Para nombrar los parámetros y los valores de retorno, al igual que para nombrar las variables y constantes, se sigue la notación *CamelCase*, combinada con el Sistema Húngaro y la Aplicación Húngara.

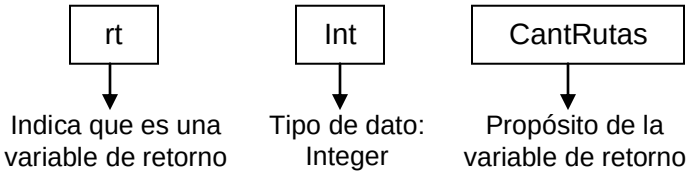
Por lo que una función en C# que devuelva la cantidad de rutas que fueron recorridas por un vehículo determinado en un rango de fechas y su valor de retorno, se nombrarían así:

Figura 18. **Ejemplo de nombre para función en Visual C#**



Fuente: elaboración propia.

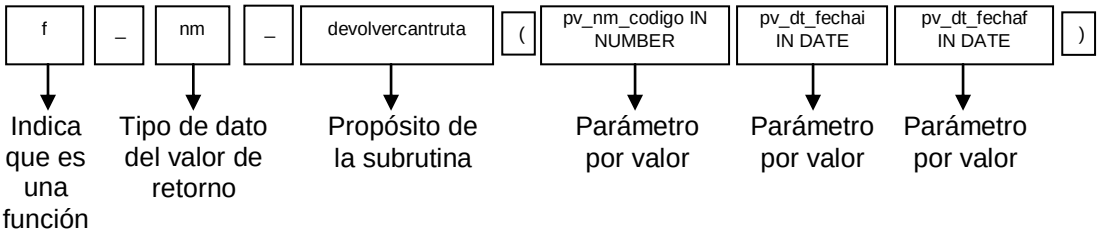
Figura 19. **Ejemplo de nombre para variable de retorno en Visual C#**



Fuente: elaboración propia.

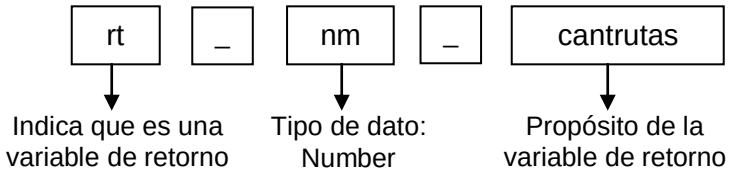
En Oracle la función que se podría invocar desde la aplicación de C#, para consultar en la base de datos la cantidad de rutas de determinado vehículo en un rango de fechas, y el valor de retorno se nombran como sigue:

Figura 20. **Ejemplo de nombre para función en Oracle**



Fuente: elaboración propia.

Figura 21. **Ejemplo de nombre para variable de retorno en Oracle**



Fuente: elaboración propia.

Se debe tener en cuenta que la declaración de las subrutinas debe realizarse antes de que sean llamadas dentro del programa principal. Esto quiere decir, que al igual que las variables o constantes, para utilizarlas, primero deben ser declaradas.

3.4. Estructuras de control de flujo

La estructura de un programa, podría decirse que está conformada por segmentos de código que pueden tener una estructura secuencial, selectiva o repetitiva. Las estructuras secuenciales se caracterizan, como su nombre lo indica, porque las instrucciones de código se ejecutan consecutivamente. Si un programa solamente tiene estructuras secuenciales, no es factible condicionar

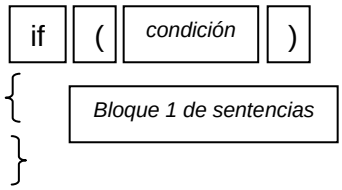
la ejecución de ciertas instrucciones, ni repetir una misma instrucción, sin tener que escribir n líneas con el mismo código; por lo cual existen las estructuras selectivas y repetitivas. Este tipo de estructuras permiten modificar el flujo de ejecuciones de las instrucciones del programa, de forma condicional (utilizando expresiones lógicas) o de forma repetitiva.

En los siguientes apartados, se indicarán los tipos de estructuras selectivas y repetitivas, y se explicará la mejor manera de escribirlas, resaltando que para estas instrucciones de código un aspecto muy importante al escribirlas es la sangría utilizada.

3.4.1. Estructuras selectivas o sentencias de salto

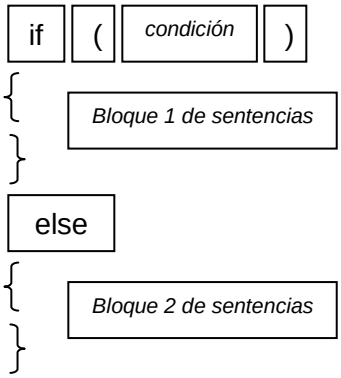
Las sentencias de salto, se identifican porque si la condición es verdadera se ejecuta un bloque de instrucciones, y si es falsa, se ejecuta otro bloque de código, o no se ejecuta nada, incluso estas estructuras pueden estar anidadas. En las condiciones se pueden utilizar tanto operadores relacionales como operadores lógicos, para comparar variables, constantes o bien los valores devueltos por una función, y pueden ser varias expresiones condicionales. La sentencia de selección *if* puede ser: simple, doble o compuesta, como se muestra en las siguientes figuras.

Figura 22. **Estructura simple de la sentencia de selección *if*, en Java**



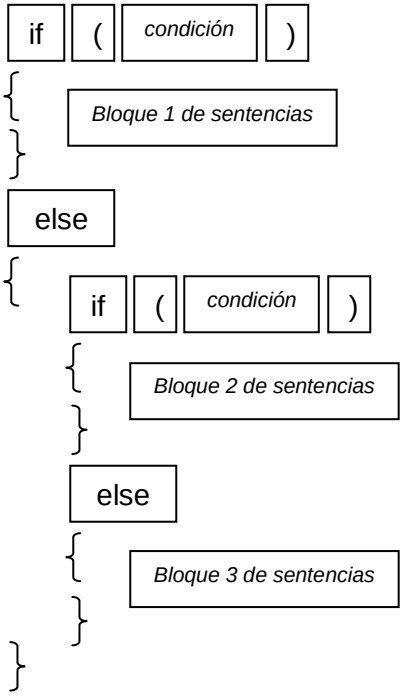
Fuente: elaboración propia.

Figura 23. **Estructura doble de la sentencia de selección *if*, en Java**



Fuente: elaboración propia.

Figura 24. **Estructura compuesta de la sentencia de selección *if*, en Java**



Fuente: elaboración propia.

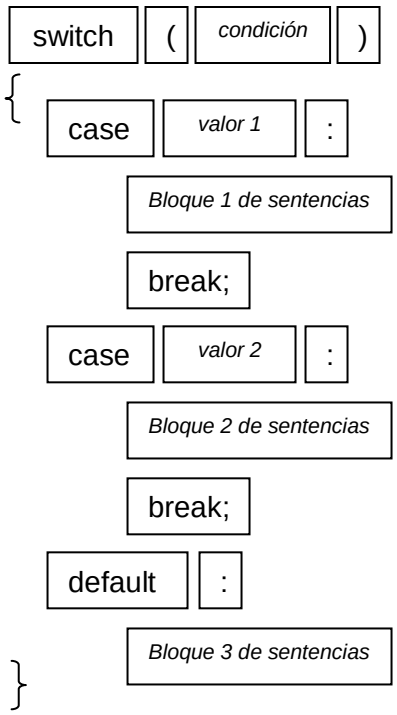
Hay que tener presente que para comparar dos expresiones, en algunos lenguajes, el operador de igualdad se escribe con dos signos igual (==), y el operador de asignación solamente con uno (=). Si en una condición se coloca el operador de asignación, el compilador de C++ sí advertirá de una incorrecta asignación; pero otros como el de Java aceptarán dicha asignación, y al ejecutarse el programa, definitivamente el resultado no será el esperado, porque lo que se ejecutará será una asignación y no una condición.

Cuando el bloque de sentencias a ejecutar solamente tiene una línea de código, los lenguajes de programación y los sistemas de gestión de base de datos permiten omitir los componentes léxicos (establecidos por la sintaxis de cada uno de ellos) para encerrar los bloques de sentencias, en algunos son las llaves ({}), en otros las palabras BEGIN y END, y en lenguajes como Visual Basic .Net, Oracle, MySql y PostgreSQL, no existen.

Como la sentencia *else* es opcional, se recomienda siempre colocar estos componentes léxicos, ya que al momento de añadir sentencias *if*, el flujo del código puede verse significativamente afectado si no se toma en cuenta la falta de estos separadores de bloque; provocando así que un *else* sea asociado por el compilador al *if* más interno sin *else*. Por ello, también se recomienda aplicar sangría a cada bloque de sentencias.

Cuando el número de alternativas es significativo, anidar varias sentencias *if* presentaría problemas de escritura, legibilidad y mantenimiento; por lo que se utiliza la estructura de selección múltiple *select*.

Figura 25. **Estructura de la sentencia de selección *switch*, en C++**



Fuente: elaboración propia.

La sentencia *select* consiste en que se evalúa una expresión, que puede dar como resultado n valores diferentes, y para cada uno de ellos se ejecutará un bloque de sentencias distinto; o bien si el resultado no coincide con los valores definidos en el código, se ejecuta el bloque de sentencias por defecto, como se muestra en la figura 25.

3.4.2. Estructuras repetitivas o sentencias de bucle

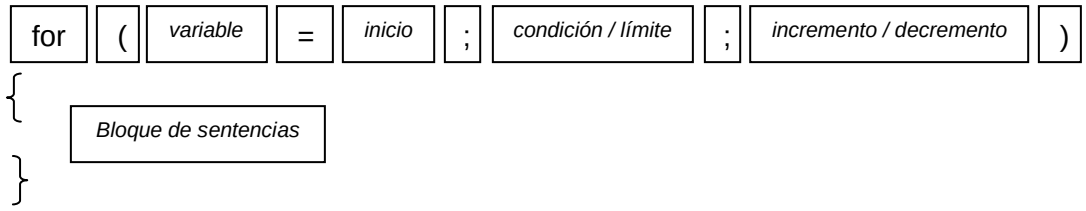
Estas sentencias ejecutan un conjunto de líneas de código, cierto número de veces, el cual puede definirse implícita o explícitamente, para no caer en un

ciclo infinito. Cuando el programador no indica explícitamente el número de repeticiones, el código se ejecuta mientras se cumpla la condición establecida.

Si se conoce la cantidad de veces que se ejecutarán determinadas líneas de código, se utiliza la sentencia desde/para. Esta sentencia requiere de una variable, que llevará la cuenta de las iteraciones, la cual inicia en un valor y dependiendo del lenguaje, termina cuando se cumple una condición o cuando se alcance el límite indicado. A esta variable se le llama contador, y se recomienda declararla en un ámbito local y llamarla con los nombres: h, i, j, k, l, m, n. En donde la variable h, será el contador del bucle más externo, la variable i del bucle siguiente, y así sucesivamente. Para esta estructura también es muy importante aplicar sangría en el bloque de sentencias dentro de ella.

El hábito de utilizar los nombres de la i a la n, para variables de tipo entero, tiene sus inicios con los programadores de FORTRAN; ya que en los programas ejecutados con el atributo *implicit*, el nombre de las variables y constantes cuyo nombre inicie con dichas letras, se declaran automáticamente de tipo entero; y las que inicien con otra letra, se declaran implícitamente de tipo real.

Figura 26. **Estructura de la sentencia de bucle *for*, en Visual C#**



Fuente: elaboración propia.

En Oracle, Visual C#, PHP, Java, PostgreSQL y C++, para el ciclo desde/para, se puede indicar si la variable disminuirá o incrementará su valor cada vez que se ejecute el bucle, lo cual deberá concordar con el valor inicial y final que se indiquen en la sentencia. En las otras herramientas de programación que implementan este tipo de ciclo, mencionadas en el presente trabajo, la variable siempre aumenta su valor.

Una variante de la sentencia *for* es la sentencia *for in*, en la cual no se requiere inicializar el contador, ni indicar el límite o condición, ni tampoco indicar si la variable debe incrementarse o debe disminuir; ya que como esta se emplea para recorrer objetos, siempre inicia en 0 y finaliza hasta terminar de recorrer el objeto, independientemente de la longitud del mismo. Por ejemplo, si se utiliza para recorrer los elementos dentro de un arreglo, en PHP se haría de la siguiente manera, notar que se sigue manteniendo la sangría en el código.

Figura 27. **Ejemplo de la sentencia *for in* en PHP**

```
0001  //Se declara el arreglo a recorrer
0002  $vrLVCodigos = array(1, 2, 3, 4, 5);
0003
0004  //Recorrido del vector que contiene números enteros
0005  foreach ($vrLVCodigos as $vrLIntCod)
0006  {
0007      //Se imprime cada valor recorrido en el arreglo
0008      echo "Valor de \$vrLVCodigos: $vrLIntCod.\n";
0009  }
```

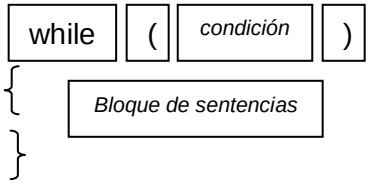
Fuente: elaboración propia.

Si no se sabe la cantidad exacta de veces que se ejecutarán los bloques de sentencias, se emplean las sentencias mientras o repetir. La sentencia mientras ejecuta ciertas líneas de código mientras se cumpla una condición, condición que es evaluada al inicio del bucle; y la sentencia repetir, las ejecuta

hasta que la condición es verdadera, la cual es evaluada al final de la estructura, lo que permite ejecutar el bucle al menos una vez.

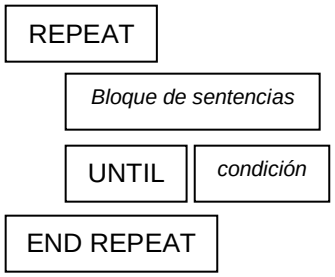
Para estas estructuras también es importante mantener la sangría, y así poder identificar cada bloque de estructuras repetitivas, especialmente cuando están anidadas.

Figura 28. **Estructura de la sentencia de bucle mientras, en PHP**



Fuente: elaboración propia.

Figura 29. **Estructura de la sentencia de bucle repetir, en MySql**



Fuente: elaboración propia.

Ni en *Microsoft* SQL ni en MySQL, existe la sentencia *for*; en el primero, solamente existe la sentencia iterativa mientras, y en MySQL, existen mientras y repetir.

Cuando los componentes léxicos utilizados para encerrar estos bloques de sentencias, son símbolos de puntuación como las llaves ({}), la llave de inicio puede escribirse en la misma línea que la sentencia de control de flujo, para así reducir el espaciado vertical. O también escribir la llave de apertura en la línea siguiente de la sentencia de control de flujo, pero seguida de código, para no dejar en una línea solamente la llave inicial.

Las estructuras cíclicas pueden ser independientes, es decir que se realiza una iteración, y hasta que ésta finalice se realiza otra iteración; o bien las estructuras pueden estar anidadas, iniciándose una iteración y dentro de ella una segunda iteración, y hasta que la más interna finalice, se retorna el control a la repetición más externa. Esta última modalidad se utiliza generalmente para recorrer estructuras de datos: arreglos de arreglos (matrices).

3.5. Manejo de errores y excepciones

Como indica la Real Academia Española, un error es una acción desacertada, equivocada o errada, y una excepción es algo que se aparta del rumbo general de su especie y que ocurre inesperadamente; ambos casos pueden presentarse en las aplicaciones, a consecuencia de deficientes pruebas realizadas a la aplicación, o bien de factores externos como un fallo en la conexión a la base de datos, por ejemplo. Estas razones son suficientes para que todos los programas requieran un manejo de errores y excepciones, porque todo Equipo de Desarrollo querrá saber por qué falló la aplicación y tener todos los datos de la falla para arreglarla.

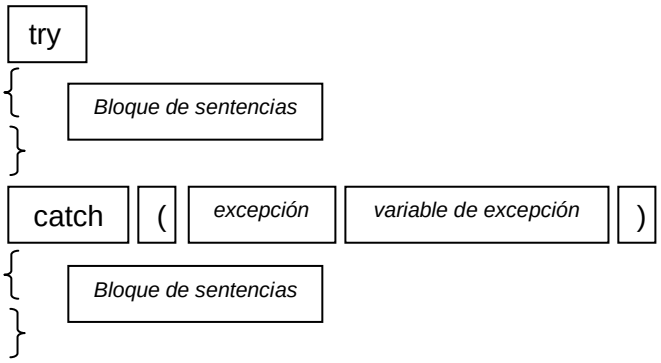
Tanto un error como una excepción interrumpen el flujo del programa, por lo que si estos no se manejan, el programa deja de ejecutarse, fomentando desconfianza y disgusto en el usuario hacia la aplicación. Para evitar lo

anterior, el manejo de estos sucesos inesperados no permite que la aplicación colapse y muestra de forma amigable los errores al usuario. Es importante que el manejo de errores y excepciones también incluya el almacenamiento de los datos asociados a ellos: fecha y hora, fuente en donde se originó (nombre de la página *web*, clase, librería, ventana, entre otros), ubicación de la excepción o del error (evento, procedimiento, función, etc.), número de error y descripción del mismo.

Datos que se recomienda almacenar en un archivo de texto, ubicado físicamente en el servidor de la aplicación en una carpeta diferente de la del proyecto o en la máquina cliente, ya que si es un error de conexión a la base de datos o al servidor, no será factible almacenarlo en la misma. Aunque dependiendo de la gravedad del error, incluso puede no llegar a almacenarse en ningún lugar, sino solamente mostrarse en pantalla.

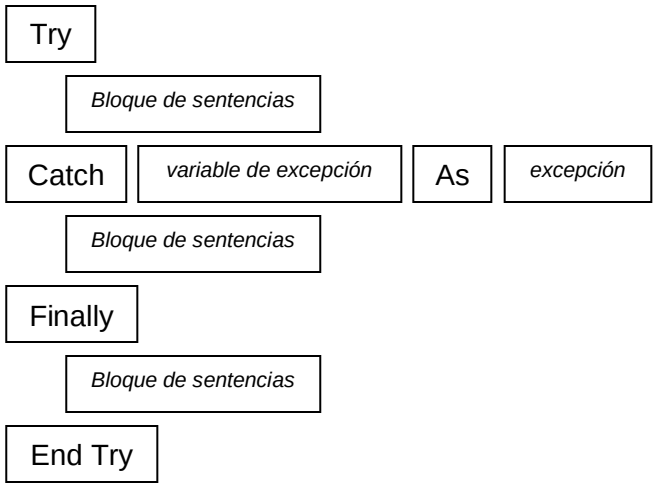
Al implementar el manejador de excepciones, se debe utilizar sangría en el cuerpo del código a monitorizar, para que por medio de dicha tabulación, se distinga este código del bloque que controla la excepción, y del bloque que siempre se ejecuta al finalizar independientemente de si ocurre o no una excepción, según la sintaxis de cada lenguaje. Aparte de aplicar la sangría en el código, no se deben omitir los componentes léxicos para encerrar cada uno de los bloques de sentencias ({} o BEGIN-END), recomendación que también se indicó para las estructuras selectivas y como se muestra en las siguientes figuras.

Figura 30. **Manejo de excepciones en Java**



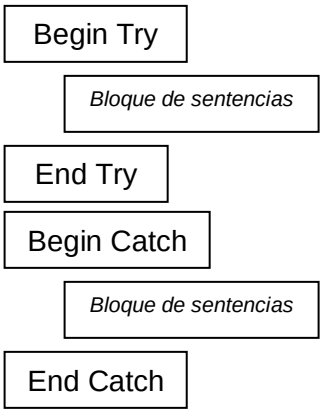
Fuente: elaboración propia.

Figura 31. **Manejo de excepciones en Visual Basic .Net**



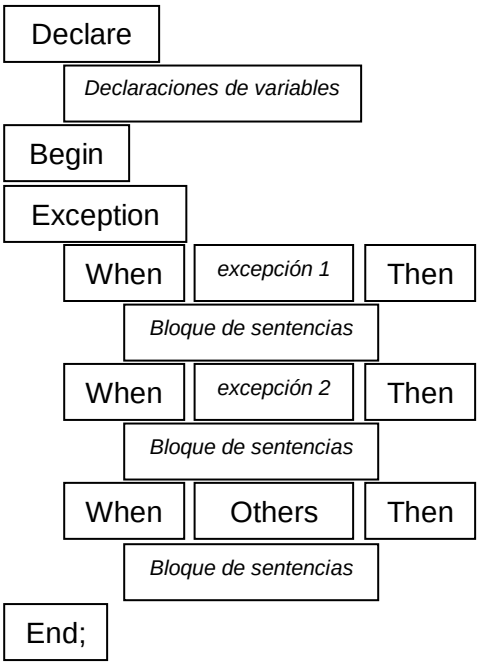
Fuente: elaboración propia.

Figura 32. **Manejo de excepciones en *Microsoft* SQL. Este manejo de excepciones aplica a partir de *Microsoft* SQL Server 2005**



Fuente: elaboración propia.

Figura 33. **Manejo de excepciones en Oracle**



Fuente: elaboración propia.

Los lenguajes de programación y sistemas de gestión de base de datos tratados en este trabajo, excepto *Microsoft* SQL, permiten capturar determinado tipo de excepción; lo cual ayuda a indicar qué debe realizar la aplicación cuando ocurra una división entre cero o cuando se viole una restricción de llave primaria; pero basándose en que una excepción ocurre inesperadamente, se recomienda utilizar manejadores de excepciones generales, que sean capaces de manejar cualquier tipo de excepción.

En el siguiente código se muestra un ejemplo en Java, de cómo indicar específicamente la excepción de arreglos y de cómo indicar una excepción general:

Figura 34. **Ejemplo de manejo de excepciones en Java**

```
0001  //Código a monitorizar
0002  try
0003   {
0004          …
0005  }
0006  //Capturar la excepción de arreglos manejados incorrectamente
0007  catch (ArrayException vrLVExExcepcion)
0008   {
0009          …
0010  }
0011  //Capturar cualquier tipo de excepción
0012  catch (Exception vrLExExcepcion)
0013   {
0014          …
0015  }
0016  /*Código que se ejecuta sin importar lo que ocurra en el bloque try o en el bloque catch*/
0017  finally
0018   {
0019          …
0020  }
```

Fuente: elaboración propia.

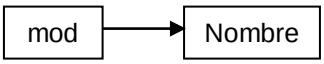
Como el manejo de excepciones también puede anidarse, se pueden manejar excepciones específicas, siempre y cuando el último manejador contenga la excepción general.

3.6. Creación de utilidades

Las librerías y módulos, conforman un repositorio de componentes, que ya están compilados y listos para utilizarse, sin llegar a ser un programa ejecutable, para utilizarlos basta con hacer referencia al archivo que contiene la utilidad: incluyendo el nombre del mismo al inicio de la aplicación, importando el propio componente a la aplicación o bien agregando una enlace a la librería, dependiendo de la herramienta de programación. Por ejemplo, tal como se realiza en C++ luego de incluir la librería *iostream.h*, ya se pueden utilizar instrucciones como *cout* (para desplegar información en pantalla) o *cin* (para obtener carácteres proporcionados por el usuario).

Para nombrar los objetos de las utilidades, declarados como módulos en Oracle, como módulos o clases en Visual Basic .Net, y como clases en Visual C#, PHP y Java, se sugiere seguir la siguiente sintaxis, empleando la notación *CamelCase*:

Figura 35. **Notación para nombrar los módulos de las utilidades**



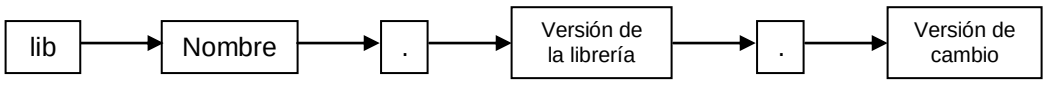
Fuente: elaboración propia.

Como las utilidades pueden ser empleadas por varias aplicaciones y las aplicaciones pueden incluir una o más de ellas, es importante invocar a las estructuras definidas por el usuario, funciones y procedimientos que las

componen, por medio del nombre del módulo; ya que utilizando referencias completas, aunque los nombres de los elementos dentro de las utilidades sean los mismos, no habrá colisión entre los nombres, evitando así errores de compilación o ejecución, dependiendo del tipo de librería (estática o dinámica).

Los archivos físicos que contienen las utilidades se nombrarán de la siguiente manera, también siguiendo la notación *CamelCase*:

Figura 36. **Forma de nombrar los archivos que contienen librerías o módulos**



Fuente: elaboración propia.

En donde el número de la versión de la librería indica que las versiones con el mismo número son compatibles; y el número de versión de cambio se altera cuando se soluciona algún error o se realiza una modificación interna que no afecta la interfaz de la librería.

El que no cambie la versión de la librería y solamente varíe el número de versión de cambio, ayuda a que en los sistemas que la utilizan, la actualización de la nueva librería sea transparente porque las diferencias no alterarán el funcionamiento de la aplicación. Además llevar el control de la versión de cambio, ayuda a determinar qué tan actualizadas están las utilidades de determinado proyecto.

El detalle de las modificaciones de cada versión puede llevarse como parte del manual técnico del proyecto, documentación que se explica en los últimos puntos de este capítulo. También pueden llevarse por medio de un programa de control de versiones, aprovechando los beneficios de estas aplicaciones para llevar el control de cambios y verificar la compatibilidad de los mismos al integrar las modificaciones.

Las constantes, variables, procedimientos, funciones, estructuras de control de flujo, manejo de errores y excepciones, utilizadas en las utilidades también deben seguir los estándares sugeridos en esta metodología, porque aunque no son una aplicación como tal, siguen siendo código que además de funcionar, debe estar escrito para facilitar la utilización por parte de cualquier persona del área de desarrollo de sistemas informáticos; sobre todo porque la propia naturaleza de las utilidades sugiere emplearlas en cualquier aplicación fuera o dentro de la organización que la desarrolló.

Por ejemplo si se quiere crear un módulo en Oracle, que contenga los procedimientos almacenados para crear, modificar y eliminar usuarios de una aplicación, se haría de la siguiente manera:

Figura 37. **Ejemplo para nombrar un módulo en Oracle**

```
0001  --Crear un módulo llamado modUsuarios
0002  create module mod_usuarios
0003        ….
0004  end module;
```

Fuente: elaboración propia.

Para crear una librería en Visual C# que valide una cuenta de correo recibida por parámetro, la declaración de la clase sería:

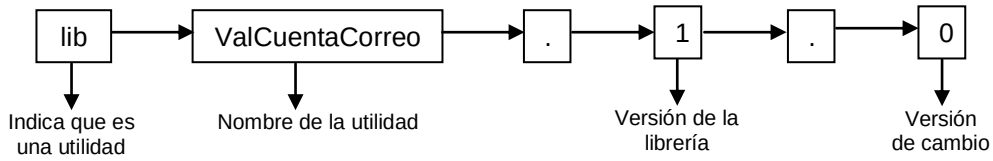
Figura 38. **Ejemplo para nombrar una librería en Visual C#**

```
0001  //Declarar la clase de la librería
0002  public class modValCuenta
0003  {     //Función que valida una cuenta de correo
0004        public int fPbIntValidarCta(string pVStrMail)
0005        {    ….
0006        }
0007  }
```

Fuente: elaboración propia.

Y el nombre del archivo que contiene modValCuenta sería:

Figura 39. **Ejemplo de nombre para un archivo que contiene una utilidad**

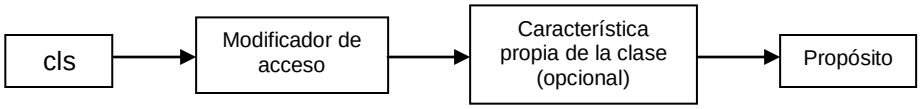


Fuente: elaboración propia.

3.7. Creación de clases

En el punto anterior se mencionó que en Visual C#, PHP y Java, se pueden crear módulos de utilidades por medio de clases; y siendo las clases una de las características principales de la programación orientada a objetos, para nombrarlas se sugiere la siguiente notación para distinguirlas de las clases de los módulos de utilidades, utilizando *CamelCase*, combinado con el Sistema Húngaro y la Aplicación Húngara:

Figura 40. **Notación para nombrar clases**



Fuente: elaboración propia.

Como se aprecia en la figura, luego del prefijo cls se indica el nivel de accesibilidad de la clase: si no se limitará el acceso utilizar el prefijo Pb, por ser una clase pública; y emplear el prefijo Pr sólo si la clase privada es parte de otra clase (clase anidada), porque solamente se podrá invocar por la clase a la que pertenece.

Los niveles descritos en el párrafo anterior son los más utilizados, sin embargo existen otros tipos específicos dentro de los lenguajes orientados a objetos como, *protected*, *internal* y *protected internal* en Visual C#. En la siguiente tabla se listan estos niveles de acceso y los prefijos sugeridos para cada uno de ellos.

Los lenguajes de programación permiten omitir el modificador de acceso al declarar las clases, pero se recomienda siempre agregarle el prefijo de este tipo de modificador al nombre de la clase, dependiendo del tipo que cada lenguaje asume por defecto cuando no se define esta propiedad, para conservar el estándar.

Tabla III. **Modificadores de acceso de las clases**

Modificador de acceso	Prefijo	Descripción
Público	Pb	Puede invocarse desde cualquier parte, el acceso no es restringido.
Privado	Pr	Se puede acceder solamente desde la clase a la que pertenece.
Protegido	Prt	Puede invocarse desde la misma clase a la que pertenece, o desde la clase que la hereda.
Interno	Intr	Puede accederse desde cualquier parte del código que esté dentro del mismo paquete en donde se haya declarado.
Protección interna	PIntr	Puede invocarse desde cualquier parte del código, dentro del mismo paquete en donde está declarada, o desde una clase que la hereda que se encuentre en otro paquete.

Fuente: elaboración propia.

En lenguajes como Java y Visual C#, además del modificador de acceso, a las clases también se les puede indicar, de manera opcional, la forma en que pueden implementarse; esto implica que al omitir este modificador, la clase se instancia, hereda y extiende según lo que dicta la teoría general de la programación orientada a objetos.

Entonces si en la declaración de la clase se indica este modificador, al nombre de la misma también se le debe agregar el prefijo sugerido para cada opción.

Por ejemplo, si la clase es estática, en Java este tipo de clase debe ser parte de otra clase, porque no pueden ser instanciadas y por lo tanto solamente pueden invocarse por la clase a la que pertenecen, al igual que se invocan las clases privadas. En Visual C# también se pueden crear clases estáticas, sin que éstas sean parte de otra clase; pero de igual manera no pueden ser instanciadas y sus métodos solamente se pueden invocar desde la clase estática a la que pertenecen.

En Visual C# existen otros modificadores, como *abstract* y *sealed*; en Java se puede utilizar *abstract*, *final* y *synchronizable*. Para identificarlos se sugiere utilizar los siguientes prefijos al nombrar la clase:

Tabla IV. **Modificadores que determinan características propias de las clases, utilizados en Java y Visual C#**

Modificador	Prefijo	Descripción
Estático	Stat	Puede invocarse solamente desde la clase a la que pertenece, no puede instanciarse.
Abstracto	Abs	Cuando se declara se implementa parcialmente, para que cada clase que la herede la termine de implementar. No se puede instanciar.
Final	Fnl	Este tipo termina una cadena de herencia, no se puede heredar.
Sincronizable	Snc	Asegura que la clase no sea accedida simultáneamente, permitiendo así, que los atributos se modifiquen sin tener problemas de sobrescritura.
Sellado	Seal	No se puede heredar.

Fuente: elaboración propia.

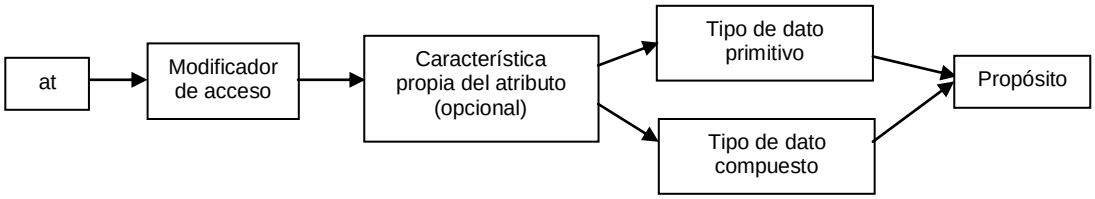
Para implementar una clase, también se requiere que ésta tenga atributos y métodos, elementos que se describen en los siguientes apartados.

3.7.1. Atributos

Los atributos son las características de la clase, que se definen como variables de algún tipo; y de igual manera que la clase, pueden tener determinado modificador de acceso para indicar cómo se puede acceder a ellos desde dentro o fuera de la clase a la que pertenecen (encapsulamiento). También son llamados, campos o propiedades; incluso variables de clase o de instancia según su tipo de modificador. Para nombrarlos se tiene la siguiente

notación, que de igual manera que la indicada para las variables, constantes, parámetros y valores de retorno, se emplea la notación *CamelCase*, combinada con el Sistema Húngaro y la Aplicación Húngara:

Figura 41. **Notación para nombrar los atributos de las clases**



Fuente: elaboración propia.

Los modificadores de acceso de las clases, listados en la tabla III, aplican también para los atributos.

Los lenguajes de programación también permiten omitir el modificador de acceso al declarar los atributos, pero de igual manera se recomienda siempre agregarle el prefijo de este tipo de modificador al nombre del atributo, tomando en cuenta el tipo que cada lenguaje asume por defecto cuando no se define este modificador, para seguir el estándar.

En PHP, Visual C#, C++ y Java, a los atributos también se les puede indicar, de forma opcional, la manera en que se pueden utilizar; si no se indica, el atributo se accede por medio de un objeto de dicha clase, según la teoría general de la programación orientada a objetos (POO). Si no se indica este modificador al declarar el atributo, al nombrarlo tampoco se debe indicar el prefijo que le corresponde. En la siguiente tabla se listan estas características opcionales, con sus prefijos y una breve descripción:

Tabla V. **Modificadores que determinan características propias de los atributos**

Modificador	Prefijo	Descripción
Estático	Stat	Puede accederse solamente desde la clase a la que pertenece, porque es un atributo que pertenece a la clase, no a los objetos instanciados.
Abstracto	Abs	Cuando se declara se implementa parcialmente, para que cada clase derivada lo termine de implementar.
Final	Fnl	Toma el valor fijado en el momento de la inicialización (cuando se declara), luego su valor no puede ser cambiado.
ReadOnly	ROly	Toma el valor fijado en el momento de la inicialización (cuando se declara) o en el constructor, luego su valor no puede ser cambiado.
Virtual	Vrl	Indica que la implementación del atributo puede modificarse en la clase derivada.
Overryde	Ory	Modifica la implementación del atributo virtual o abstracto, en la clase derivada.
Volátil	Vtl	Permite que el atributo sea accedido simultáneamente, con lo que el valor del atributo siempre estará actualizado.
Transitorio	Tns	Indica que el valor del atributo no debe almacenarse cuando se archive el objeto, porque no es una parte persistente del estado del objeto.

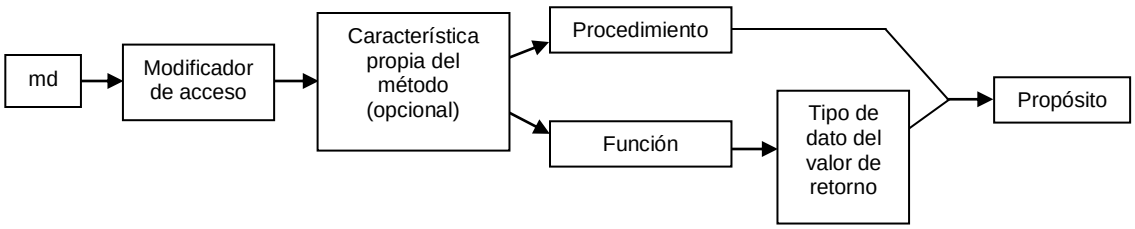
Fuente: elaboración propia.

Luego de definir los lineamientos para nombrar los atributos de una clase, se continúa en el siguiente apartado, con lo correspondiente a los métodos que determinan el comportamiento de una clase.

3.7.2. Métodos

Los métodos en una clase, no son más que subrutinas (procedimientos o funciones), por lo que se sugiere nombrarlos como se indica en la siguiente figura, en donde se emplea la notación *CamelCase*, combinada con la Aplicación Húngara.

Figura 42. **Notación para nombrar métodos de clases**



Fuente: elaboración propia.

Como se puede observar en el estándar propuesto para nombrar estas subrutinas, a estos miembros de la clase, también se les puede indicar, por medio de modificadores de acceso, cómo se puede acceder a ellos. Estos modificadores son los mismos indicados, anteriormente, para las clases y los atributos, por lo que la información indicada en la tabla III, también aplica para los métodos.

A los métodos también se les puede indicar, de manera opcional, la forma en que se pueden utilizar; si no se indica, el método se accede por medio de una instancia de la clase, según lo dictado por la teoría general de la POO. Si al declarar el método este modificador no se indica, al nombrarlo tampoco se antepone el prefijo.

En la siguiente tabla se listan estas características opcionales, utilizadas en PHP, Visual C#, C++ y Java:

Tabla VI. **Modificadores que determinan características propias de los métodos**

Modificador	Prefijo	Descripción
Estático	Stat	Puede accederse solamente desde la clase a la que pertenece, porque pertenece a la clase, no a los objetos instanciados.
Abstracto	Abs	Cuando se declara solamente se indica el tipo de retorno, el nombre y los parámetros, para que cada clase derivada lo implemente.
Final	Fnl	El método no puede redefinirse por medio de una clase derivada.
Virtual	Vrl	Indica que la implementación del método puede modificarse en la clase derivada.
Overryde	Ory	Modifica la implementación del método virtual o abstracto, en la clase derivada.
Nativo/Externo	Ext	Método que está escrito en otro lenguaje, y puede ser utilizado como un método escrito en el lenguaje en el que está definida la clase.
Sincronizable	Snc	Solamente puede ser ejecutado por un hilo a la vez, el método se bloquea hasta que termine de ejecutarse.

Fuente: elaboración propia.

La convención para nombrar los parámetros que reciben los métodos de las clases, y para los valores de retorno de las funciones, es la misma indicada en los puntos 3.3.1 y 3.3.2, respectivamente.

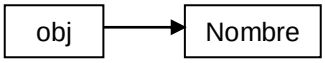
Cuando la clase ya tiene definidos sus miembros, está lista para ser instanciada, a menos que sea una clase estática o abstracta, por lo que en el siguiente segmento, se indica cómo deben nombrarse las instancias de las clases.

3.7.3. Instanciación de clases

La instanciación de una clase es lo homólogo a la declaración e inicialización de variables, en donde se indica el nombre de la misma, su tipo y su valor inicial. En este caso, se llama objeto a la instancia de la clase, el cual la representa, y tendrá los atributos y métodos que se hayan definido en ella.

Se sugiere utilizar la notación *CamelCase* y la siguiente nomenclatura para nombrar los objetos de las clases:

Figura 43. **Notación para nombrar objetos de clases**



Fuente: elaboración propia.

Después de definir la nomenclatura de las clases y sus miembros, se ejemplifica la misma con el siguiente código PHP, que ingresa y consulta el ISBN y el nombre de libros de una biblioteca:

Figura 44. **Ejemplo para nombrar clases y objetos en PHP**

```
0001  //Definición de la clase pública Libro
0002  class clsPbLibro
0003  {    private $atPrStrISBN = ‘’; //atributo privado que almacena el identificador del libro (ISBN)
0004       private $atPrStrNombre = ‘’; //atributo privado que almacena el nombre del libro
0005
0006       //Procedimiento que fija el ISBN y el nombre del libro
0007       public function mdPbPcFijaDatos($pVStrISBNLibro, $pVStrNombreLibro)
0008       {    $this->atPrStrISBN = $pVStrISBNLibro; //fija el ISBN del libro
0009            $this->atPrStrNombre = $pVStrNombreLibro; //fija el nombre del libro
0010       }
0011
0012       //Función que retorna el ISBN del libro
0013       public function mdPbFStrDevolverISBN()
0014       {    return $this->atPrStrISBN; //devuelve el ISBN del libro
0015       }
```

Continuación de la figura 44.

```
0016
0017        //Función que retorna el nombre del libro
0018        public function mdPbFStrDevolverNombre()
0019        {     return $this->atPrStrNombre; //devuelve el nombre del libro
0020        }
0021
0022  //Creando el libro “El regreso de Sherlock Holmes”
0023  $objRegresoSherlockH = new clsPbLibro();
0024
0025  //Desde la nueva instancia de la clase, se llama al método público que fija el ISBN y el nombre del
0026  libro
0027  $objRegresoSherlockH->mdPbPcFijaDatos(“9781420931648”, “EL REGRESO DE SHERLOCK
0028  HOLMES”);
0029
0030  //Se muestran en pantalla los datos de $objRegresoSherlockH
0031  echo("NUEVO LIBRO DE LA BIBLIOTECA: " . $objRegresoSherlockH->mdPbFStrDevolverISBN() .
0032  " - " . $objRegresoSherlockH->mdPbFStrDevolverNombre() . "\n");
0033  }
```

Fuente: elaboración propia.

En esta metodología se sugiere que, para los nombres de variables, constantes, parámetros, valores de retorno, clases y sus atributos, se utilice uno o más sustantivos para indicar su propósito. Y que los identificadores de procedimientos, funciones (declarados dentro o fuera de una utilidad) y métodos de clases, incluyan al menos un verbo en el nombre del propósito; esto se puede observar en los ejemplos de los apartados del presente capítulo.

Adicionalmente, para funciones que retornen un valor booleano (verdadero o falso), se puede indicar su propósito, utilizando el modo indicativo de la tercera persona del singular, de los verbos ser o estar, por ejemplo:

Figura 45. **Ejemplo para funciones que retornan un valor booleano**

```
0001  'Función en Visual Basic .Net que indica si una cadena es un número entero
0002  Public Function fPbBolEsEntero(ByVal pVStrCadena As String) As Boolean
0003  ……..
0004  End Function
0005
0006  'Función de Visual Basic .Net que indica si cierto producto está en 'determinada tienda
0007  Public Function fPbBolEstaEnTienda(ByVal pVLngProducto As Long, _
0008          ByVal pVSIntTienda As Short) As Boolean
0009  ……..
0010  End Function
```

Fuente: elaboración propia.

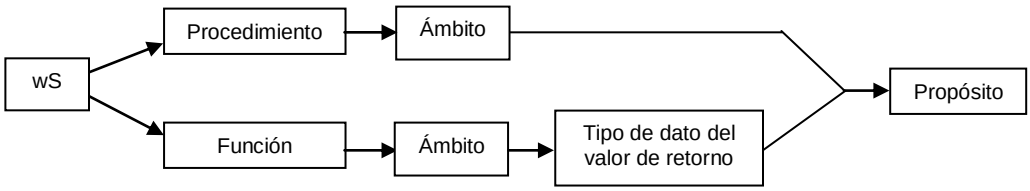
3.8. Servicios *Web*

Los procedimientos y funciones convencionales se declaran para ser utilizados por una misma aplicación; en el caso de las utilidades, las subrutinas se encuentran encapsuladas en las librerías y son utilizadas por una o más aplicaciones, pero en ningún caso hay interacción entre varias aplicaciones a través de ellas. Para ello se hace uso de los servicios *web*, los cuales son subrutinas remotas, que por medio de protocolos y estándares definidos, sirven para intercambiar información entre aplicaciones.

En los últimos años, su utilización ha aumentado, ya que se basan en HTTP sobre TCP, lo que facilita el acceso por la red y proveen flexibilidad, ya que su implementación es independiente de las aplicaciones que utilizan el servicio *web*.

Para nombrar los servicios *web*, al estándar mencionado en el punto 3.3 de este documento, se le hace el siguiente cambio, anteponiéndole el prefijo wS al nombre de la subrutina:

Figura 46. **Notación para nombrar servicios *web***



Fuente: elaboración propia.

Generalmente los servicios *web* invocados por las aplicaciones serán funciones, porque la aplicación que invoca al servicio, requerirá saber si hubo éxito o no, por medio de un código de mensaje, por ejemplo.

3.9. Código HTML

Al hablar de servicios *web* y HTTP, también cabe mencionar el lenguaje HTML (*Hyper Text Markup Language*), lenguaje que no es de programación, sino un lenguaje de marcas utilizado para la elaboración de páginas *web*. Dicho lenguaje se escribe por medio de etiquetas HTML, encerradas entre corchetes angulares (<>); para cada elemento hay una etiqueta inicial (<EtiquetaInicial>) y otra de cierre (</EtiquetaFinal>), que pueden o no encerrar algún contenido. Por ejemplo la etiqueta br no encierra ningún contenido, porque es un salto de línea; en cambio *title* sí contiene un texto que es el título del documento.

Los elementos HTML tienen atributos y eventos, los cuales pueden ser accedidos por medio del atributo *id* especificado al momento de escribir el elemento; es decir que se puede acceder a los atributos y eventos por medio del nombre del elemento, desde un *script* con la notación: objeto.evento u objeto.atributo.

Para nombrar los identificadores de los elementos HTML dentro de las páginas *web*, se sugiere utilizar la notación *CamelCase*, combinada con la Aplicación Húngara. Los prefijos para anteponerle al nombre de cada elemento, se describen en la tabla VII. Se debe tomar en cuenta que, el atributo *id*, no es soportado por todas las etiquetas HTML, por lo que en la tabla se indica el carácter -, para indicar que el elemento no puede identificarse, por no poseer atributo *id*. Si no se accederá a los atributos ni eventos de los elementos HTML, se puede omitir el atributo *id* al escribir cada elemento.

Algunos elementos HTML también cuentan con el atributo *name*, cuyas diferencias con *id* son:

- El nombre sí se incluye como parte de la información de la forma que se envía al servidor (*submit*), el identificador no.
- Con el identificador, el elemento puede ser referenciado utilizando la instrucción document.getElementById('identificadorElemento').
- Con el nombre, el elemento se puede referenciar utilizando la instrucción document.forms['NombreDeLaForma'].elements ['NombreElemento'].
- Los elementos con nombre, son accedidos indicando un índice numérico, ya que están en document.forms[índiceForma].elements[índiceElemento].
- Al utilizar la instrucción getElementsByName('NombreElemento'), se obtiene una colección de elementos que comparten el mismo nombre.

Aunque los valores del atributo *name* se puedan repetir dentro de los elementos HTML de la forma, se recomienda colocar el mismo valor en el atributo *id*, para un mismo elemento.

En la siguiente tabla no se indica prefijo, para el elemento *input*, porque éste puede ser de varios tipos; por lo que en la tabla VIII, se listan los prefijos para cada uno de los valores que puede tomar el atributo *type* en esta etiqueta.

Tabla VII. **Prefijos de las etiquetas HTML**

Etiqueta HTML	Prefijo	Descripción
<a>	bkm	Inserta vínculos o marcadores.
<abbr>	abr	Explica abreviaciones.
<acronym>	acrn	Explica acrónimos.
<address>	adr	Provee información de contacto.
<area>	area	Define sectores para mapas de imagen.
<b>	b	Texto en negrita.
<base>	-	URI base para resolver URIs relativas.
<bdo>	bdo	Suprime el algoritmo bidireccional.
<big>	bg	Texto en tamaño grande.
<blockquote>	bkq	Citar párrafos.
<body>	bdy	Contiene los elementos a mostrar.
 	br	Fuerza un quiebre de línea.
<button>	btn	Crea un botón.
<caption>	cptn	Establece un título para una tabla.
<cite>	cite	Inserta una cita o referencia.
<code>	cd	Representa texto de computadora.
<col>	col	Da atributos a columnas en una tabla.
<colgroup>	colgp	Agrupa columnas en una tabla.
<dd>	dd	Define descripciones en una lista.
<del>	del	Indica texto eliminado.
<dfn>	dfn	Asigna una definición a un término.
	div	Define un bloque de contenido.
<dl>	dl	Define una lista.
<dt>	dt	Inserta un término en una lista.
	em	Indica énfasis.
<fieldset>	fldst	Agrupa controles en un formulario.
<form>	frm	Inserta un formulario.
<frame>	frme	Inserta un marco.
<frameset>	frmst	Inserta un grupo de frames.

Continuación de la tabla VII.

Etiqueta HTML	Prefijo	Descripción
<h1>	h1	Encabezado de nivel 1.
<h2>	h2	Encabezado de nivel 2.
<h3>	h3	Encabezado de nivel 3.
<h4>	h4	Encabezado de nivel 4.
<h5>	h5	Encabezado de nivel 5.
<h6>	h6	Encabezado de nivel 6.
<head>	-	Define el bloque de encabezado.
<hr>	hr	Dibuja una línea o regla horizontal.
<html>	-	Contiene al documento.
<i>	i	Muestra texto en itálica.
<iframe>	ifrme	Inserta un marco dentro del documento.
<img>	img	Inserta una imagen.
<input>	-	Muestra controles de entrada de varios tipos, ver Tabla VIII.
<ins>	ins	Indica texto insertado.
<kbd>	kbd	Indica texto a ingresarse por el usuario.
<label>	lbl	Establece una etiqueta para un control.
<legend>	lgn	Asigna un título a un fieldset.
	li	Define un artículo en una lista.
<link>	lnk	Define la relación entre el documento HTML y un recurso externo.
<map>	mp	Define un mapa de imagen.
<meta>	-	Da información sobre el documento.
<noframes>	nfrme	Contenido alternativo para marcos.
<noscript>	nscpt	Contenido alternativo para scripts.
<object>	obj	Ejecuta aplicaciones externas.
	ol	Inserta una lista ordenada.
<optgroup>	optgp	Agrupa opciones en un control select.
<option>	opt	Define una opción en un control select.
<p>	p	Define un párrafo.
<param>	prm	Da un parámetro para un objeto.
<pre>	pre	Bloque de texto preformateado.
<q>	q	Inserta una cita en una línea.
<samp>	smp	Representa texto de programas.
<script>	-	Contiene scripts.

Continuación de la tabla VII.

Etiqueta HTML	Prefijo	Descripción
<select>	slct	Crea un control select.
<small>	sml	Muestra texto en letra pequeña.
	spn	Asigna atributos al texto en líneas.
	stg	Indica énfasis fuerte.
<style>	-	Define atributos visuales (hojas estilo).
<sub>	sb	Define texto en subíndice.
<sup>	sp	Define texto en super-índice.
<table>	tbl	Inserta una tabla.
<tbody>	tby	Define un cuerpo en una tabla.
<td>	td	Celda regular de una tabla.
<textarea>	txta	Entrada de texto de líneas múltiples.
<tfoot>	tft	Define un pie en una tabla.
<th>	th	Celda de encabezado de una tabla.
<thead>	thd	Define un encabezado de tabla.
<title>	-	Define el título del documento.
<tr>	tr	Inserta una fila en una tabla.
<tt>	tt	Muestra texto en teletype.
	ul	Inserta una lista sin orden.
<var>	vr	Indica una instancia de una variable.

Fuente: elaboración propia.

Los prefijos indicados en la tabla VII, pueden ser aplicados a los nombres de controles (botones, etiquetas, cajas de texto, etc.) utilizados en formularios para desarrollar aplicaciones *standalone*, también a nombres de controles *web* (controles que provee la caja de herramientas de Visual Studio .Net), a nombres de componentes de la librería Swing de Java, y también para nombrar cualquier componente GUI (*Graphical user interface*).

Tabla VIII. **Prefijos para los valores del atributo type, en el elemento INPUT**

Valor del atributo *type*	Prefijo	Descripción
button	btn	Define un botón
checkbox	ck	Define una casilla de selección, se pueden seleccionar varias a la vez
file	fl	Define un elemento INPUT y un botón de "Búsqueda …", para subir archivos
hidden	hd	Define un INPUT que no es visible para el usuario, pero puede almacenar valores
image	img	Define una imagen que actúa como un botón que envía la información de la forma al servidor
password	pw	Define un campo en donde el texto se muestra enmascarado
radio	rd	Define una casilla de selección, solamente se puede seleccionar una casilla
reset	rst	Define un botón que restablece todos los campos de la forma, a sus valores iniciales
submit	smt	Define un botón que envía la información de la forma al servidor
text	txt	Define un campo en donde el usuario puede ingresar texto

Fuente: elaboración propia.

A continuación se muestra el código de una página *web* en donde se pueden ingresar los nombres y apellidos de una persona, y muestra los datos ingresados al presionar un botón:

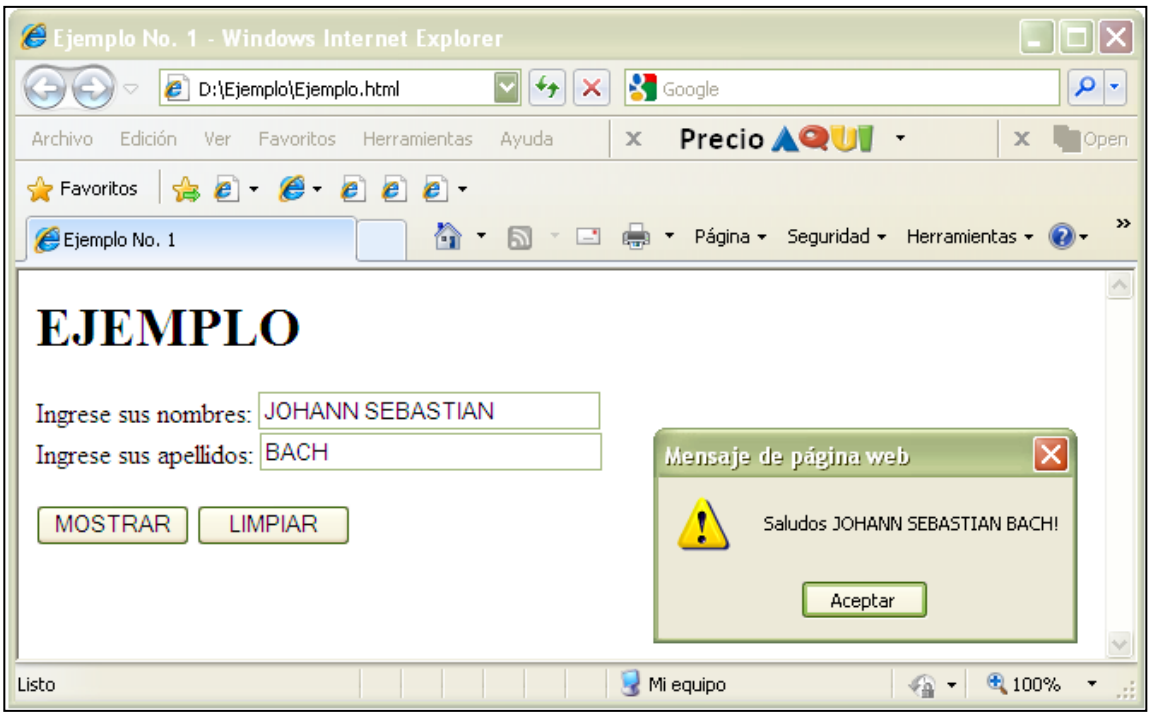
Figura 47. **Ejemplo de una página *web***

```
0001  <!-- Ejemplo No. 1
0002  Autor:                    Norma Magaly Chonay Vásquez
0003  Fecha creación:           02/01/2011
0004  Fecha última modificación: 02/01/2011
0005  Descripción:              Solicitar datos al usuario, y mostrarlos en pantalla al presionar un
0006  botón
0007  -->
0008  <html>
0009      <head>
0010          <!-- Título del documento -->
0011          <title>Ejemplo No. 1</title>
0012              <script language="jscript">
0013                  //Le da el enfoque al control recibido por parámetro
0014                  function pcPbDarEnfoque (pRTxtControl)
0015                  {   pRTxtControl.focus();
0016                  }
0017              </script>
0018      </head>
0019      <body id="bdyEjemplo" onload="pcPbDarEnfoque(frmPrincipal.txtNombre)">
0020          <!-- Encabezado principal (nivel 1) -->
0021          <h1>EJEMPLO</h1>
0022          <form id="frmPrincipal" name="frmPrincipal">
0023              <!-- Elementos que reciben los datos ingresados por el usuario -->
0024              Ingrese sus nombres:  <input type="text" name="txtNombre" id="txtNombre" value=""
0025                      tabindex="1" maxlength="70" style="text-transform: uppercase; width:200px"/><br />
0026              Ingrese sus apellidos: <input type="text" name="txtApellido" id="txtApellido" value=""
0027                      tabindex="2" maxlength="70" style="text-transform: uppercase; width:200px"/><br />
0028              <br />
0029              <!-- Botón que al presionarlo muestra los datos ingresados por el usuario -->
0030              <input type="button" name="btnMostrar" id="btnMostrar" value="MOSTRAR"
0031                      tabindex="3" alt="Presione aquí, para ver sus datos" style="width:90px"
0032                      onclick="alert('Saludos ' + txtNombre.value.toUpperCase() + ' ' +
0033                                  txtApellido.value.toUpperCase() + '!');"/>
0034              <input type="reset" name="btnLimpiar" id="btnLimpiar" value="LIMPIAR" tabindex="4"
0035                      style="width:90px" onclick="pcPbDarEnfoque(frmPrincipal.txtNombre)"/>
0036          </form>
0037      </body>
0038  </html>
```

Fuente: elaboración propia.

El código escrito anteriormente, da como resultado la página *web* que se muestra a continuación:

Figura 48. **Página *web* en lenguaje HTML y *JavaScript***



Fuente: elaboración propia.

3.10. Código *script*

Para proveer de dinamismo a las páginas *web*, surgieron los lenguajes *script*, utilizados para crear animaciones, realizar efectos visuales, proveer interactividad, procesar formularios, validar datos, y enviar correos electrónicos, por ejemplo.

Los *scripts* se pueden clasificar en *scripts* del lado del cliente y del lado del servidor; la principal diferencia entre uno y otro radica en el lugar en donde se procesan: en el servidor que aloja la aplicación o en la máquina cliente (máquina del usuario en donde se ejecuta el navegador).

El uso de determinado tipo de *script* depende de los objetivos de la aplicación; por ejemplo, si se desean reducir las peticiones hechas al servidor, los *scripts* del lado del cliente deberán realizar la mayor cantidad de procesos posibles en la máquina del usuario, para que las peticiones realizadas al servidor sean solamente las necesarias.

Cuando las aplicaciones utilizan este tipo de *scripts*, la máquina cliente debe contar con suficiente espacio en disco (porque los *scripts* se almacenan en la computadora), el navegador debe permitir la ejecución de *scripts* y de preferencia tener configurado que la actualización de los *scripts* sea cada vez que se visite el sitio *web*, para que los clientes siempre utilicen la última versión disponible en el servidor de estas librerías.

Es importante mencionar que el código del *script* del lado del cliente puede ser modificado sin necesidad de compilar la aplicación, pues es interpretado por el navegador; lo que provee flexibilidad al momento de una actualización; pero también hay que tomar en cuenta que esta característica, permite que el código sea alterado en la máquina cliente, lo cual puede ser minimizado si el navegador comprueba las nuevas versiones de las páginas cada vez que se visite cada sitio *web*, como se recomienda en el párrafo anterior.

La gran ventaja de los *scripts* del lado del servidor, es que el código no puede ser accedido desde la máquina cliente; las credenciales para la conexión a la base de datos, por ejemplo, quedan fuera del alcance de la máquina cliente.

Actualmente, realizar *scripts* del lado del servidor en lenguajes como Visual Basic .Net, C#, Java, PHP o Perl, es una práctica muy utilizada, que puede realizarse siguiendo la metodología descrita hasta este punto.

Dentro de los lenguajes *script* del lado del cliente, se encuentran *JavaScript* (o JScript) y VBScript, siendo el primero el más utilizado, ya que VBScript solamente puede ser interpretado por Internet Explorer. En esta clasificación, también cabe mencionar la técnica AJAX (*Asynchronous JavaScript And XML*), que es una combinación de *JavaScript* y XML, empleada en las aplicaciones *web*, para ejecutar código del lado del cliente (por medio de *JavaScript*), manteniendo una comunicación asíncrona con el servidor en segundo plano (a través de XML). Para esta clase de *scripts* también aplica la metodología descrita hasta el momento.

Los *scripts* del lado del cliente pueden estar dentro de un archivo, que será incluido en la cabecera de la página HTML, como si fuera una librería; práctica que se sugiere seguir para mantener la programación modular. Como en algunas ocasiones es imprescindible incluir código dentro de la misma página HTML, ya sea con lenguaje *script* del lado del cliente o del lado del servidor, también se sugiere seguir la metodología Extilo Regular 9002.

A continuación se muestra el *script* en lenguaje JScript, extraído del código utilizado en la página *web* de la figura número 48, en donde se puede apreciar que se sigue la metodología descrita en este trabajo:

Figura 49. **Ejemplo de código *script***

```
0001  <script language="jscript">
0002        //Le da el enfoque al control recibido por parámetro
0003        function pcPbDarEnfoque (pRTxtControl)
0004        {   pRTxtControl.focus();
0005        }
0006   </script>
```

Fuente: elaboración propia.

Luego de exponer la mayor parte de lineamientos para programar código en el desarrollo de aplicaciones, y para escribir instrucciones que son comunes entre los lenguajes de programación y gestores de bases de datos, en el siguiente apartado, se detallan las especificaciones para nombrar objetos propios de las bases de datos.

3.11. Objetos de base de datos

Una base de datos se refiere a cualquier conjunto de datos almacenados, y en informática, los programas para administrarla son los sistemas de gestión de base de datos; siendo Oracle, *Microsoft* SQL, MySql y PostgreSQL, los SGBD utilizados de muestra en esta metodología.

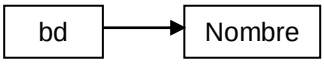
Para almacenar los datos, las bases de datos están formadas por objetos, los cuales son definidos por el lenguaje de definición de datos (DDL, *Data Definition Language*), gestionados por el lenguaje de manipulación de datos (DML, *Data Manipulation Language*), y asegurados (niveles de acceso) por el lenguaje de control de datos (DCL, *Data Control Language*).

Cabe mencionar que cada gestor tiene su propia sintaxis, pero todos ellos se basan en el lenguaje de consulta estructurado SQL (S*tructured Query Language*) para administrar las bases de datos; por lo que la sintaxis de cada SGBD mencionado en el presente trabajo, queda fuera del alcance de esta metodología, más no así, la notación para nombrar los objetos definidos por el DDL, y para definir la estructura de consultas empleando el DML.

3.11.1. Tablas

Lo primero es crear la base de datos, para lo cual se sugiere utilizar la notación *CamelCase* y la siguiente nomenclatura para nombrarla:

Figura 50. **Notación para nombrar bases de datos**

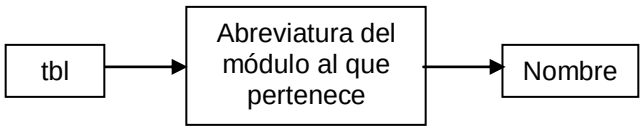


Fuente: elaboración propia.

Teniendo la base de datos, se pueden crear las tablas, las cuales son una colección de filas y columnas, en donde las filas son también llamadas tuplas o registros, que tienen el valor asignado a cada columna o atributo de la tabla. Y cada columna, es un campo de determinado tipo de dato, que define la estructura de la tabla.

Así como se sugirió agrupar los archivos principales de la aplicación por módulos, también se sugiere agrupar las tablas: las del módulo de seguridad, las del módulo de configuración, y los que apliquen para la aplicación a desarrollar. Para nombrar las tablas con la notación *CamelCase*, se hará de la siguiente manera, tomando en cuenta que los nombres de las tablas deben ir en singular, porque hacen referencia al tipo de información que representa la tabla (clase) y cada tupla es un objeto de dicha clase:

Figura 51. **Notación para nombrar tablas en una base de datos**

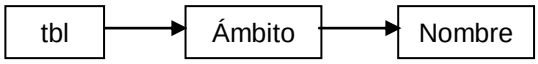


Fuente: elaboración propia.

Si hay esquemas (espacios de nombres) creados en la base de datos que definan los módulos de la aplicación, en donde las tablas pueden ser clasificadas, al nombrar la tabla, ya no será necesario indicar el módulo al que pertenece. Por ejemplo, si en la base de datos se crea un esquema con nombre seg, bajo el cual estará la tabla que almacenará los datos de usuarios, la misma se nombra tblUsuario en lugar de tblSegUsuario, ya que para hacer referencia a ella se hará como seg.tblUsuario y no seg.tblSegUsuario.

Cabe mencionar que para nombrar a las tablas temporales, ya sean locales o globales, se hará como sigue:

Figura 52. **Notación para nombrar tablas temporales en una base de datos**

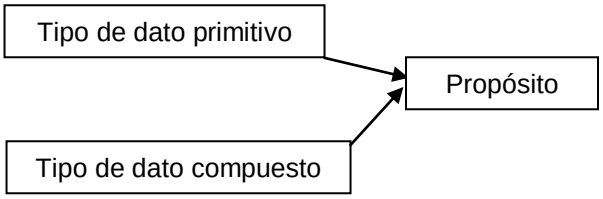


Fuente: elaboración propia.

Para indicar el ámbito, se utilizarán los mismos prefijos indicados para los ámbitos de las variables: L si la tabla es visible solamente en la sesión actual, y G, si es visible en todas las sesiones activas de la base de datos.

Al crear una tabla, ya sea una tabla permanente, temporal o una variable de tipo tabla, se deben indicar sus columnas, y para nombrarlas se sugiere la siguiente notación utilizando *CamelCase*:

Figura 53. **Notación para nombrar columnas de tablas**



Fuente: elaboración propia.

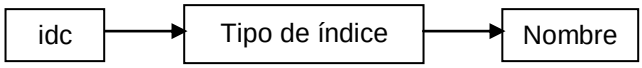
En donde el prefijo del tipo de dato, es el que se indica en la tabla número dos.

3.11.2. Índices

Para optimizar el acceso a las bases de datos, se emplean los índices, ya que por medio de ellos se pueden realizar menos accesos al disco ubicando más rápidamente dónde está la información, utilizando las estructuras de datos que éstos implementan, como los árboles B.

Por defecto, manejadores de base de datos como *Microsoft* SQL crean automáticamente índices para las llaves primarias especificadas en cada tabla; proceso en que el manejador nombra los índices utilizando su propia notación. Para los índices que se creen a lo largo del desarrollo, implementación y mantenimiento de una aplicación, se presenta la siguiente nomenclatura utilizando la notación *CamelCase*:

Figura 54. **Notación para nombrar índices de bases de datos**



Fuente: elaboración propia.

Para los tipos de índice se tienen los siguientes prefijos:

Tabla IX. **Prefijos para los tipos de índice**

Tipo de índice	Prefijo	Descripción	SGBD que implementa el tipo de índice			
			Oracle	SQL	MySQL	PostgreSQL
Árbol B	arbB	Implementa la estructura de datos árbol B.	x		x	x
Bitmap	bmp	En un arreglo de bits (bitmap) se indica por cada fila de la tabla, con 1 si la misma contiene el dato indexado, y con 0 si no lo contiene.	x			
Particionado	ptc	Cada partición indexa un valor de la columna o columnas que incluye el índice.	x			
Basado en función	fnc	Indexa expresiones basadas en funciones predefinidas o definidas por el usuario.	x			x
Dominio	dmn	Implementa un tipo de índice definido por el usuario, de acuerdo a la lógica del negocio.	x			
Agrupado (clúster)	agr	Junto con la llave del índice, se encuentran los datos de la fila.		x		
No agrupado (no clúster)	nagr	Junto con la llave del índice, se encuentra un puntero hacia la fila correspondiente.		x		
Único	unc	Garantiza que los valores del dato indexado no se repitan.		x	x	x
Con columnas incluidas	ccol	Índice no agrupado que almacena otras columnas de la fila, junto con la llave del índice.		x		

Continuación de la tabla IX.

Tipo de índice	Prefijo	Descripción	SGBD que implementa el tipo de índice			
			Oracle	SQL	MySQL	PostgreSQL
Texto completo	txc	Indexa componentes léxicos de la columna o columnas (tipo char, text, varbinary, en sus diferentes modalidades, image y xml), sobre la cual se creó el índice.		x	x	
Espacial	spl	Índice para columnas de tipo espacial (tipos geométricos y geográficos)		x	x	x
Filtrado	ftd	Índice no agrupado que por medio de un filtro, indexa ciertas filas de la tabla.		x		x
XML	xml	Índice para columnas de tipo xml.		x		
Hash	hsh	Implementa como estructura de datos, una tabla de Hash.			x	x
GiST	gst	Estructura de datos base, para implementar árboles de búsqueda: B+ o árboles R, por ejemplo.				x

Fuente: elaboración propia.

3.11.3. Otros objetos

Adicionalmente a los objetos de base de datos mencionados en los puntos anteriores, hay otras entidades de uso frecuente que pueden crearse para gestionar la información, como las que se listan en la siguiente tabla.

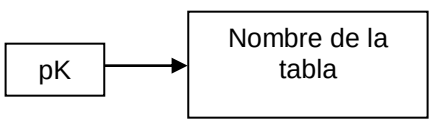
Tabla X. **Prefijos para otros objetos frecuentemente utilizados en las bases de datos**

Objeto de la base de datos	Prefijo	Descripción
Vista	view	Tabla lógica definida por una consulta, que está basada en una o más tablas o vistas.
Desencadenador (*trigger*)	tgr	Subrutina que se ejecuta automáticamente cuando una tabla o vista (en algunos SGBD) es modificada por eventos DML, DDL o DCL (conexión o inicio de sesión a la BDD).
Diagrama	diag	Representa gráficamente las tablas y sus relaciones.
Rol	rol	Conjunto de permisos para gestionar determinados objetos de la base de datos.
Usuario	usr	Usuario de la base de datos que tiene uno o más roles.
Inicio de sesión	lgin	En *Microsoft* SQL, este objeto es el que permite que un usuario pueda conectarse a la BDD.

Fuente: elaboración propia.

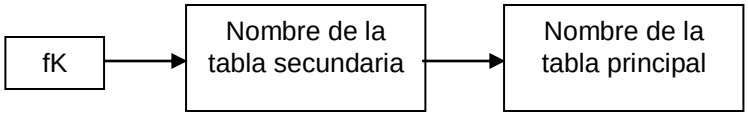
El nombre que los manejadores de base de datos relacionales asignan a las restricciones de las tablas, utilizando su propia notación, se puede modificar utilizando la siguiente nomenclatura:

Figura 55. **Notación para nombrar la llave primaria en una tabla**



Fuente: elaboración propia.

Figura 56. **Notación para nombrar la llave foránea en una tabla**

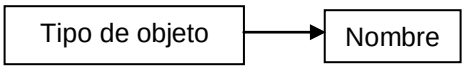


Fuente: elaboración propia.

Aunque los diagramas no son objetos creados por el DDL, son gráficos utilizados dentro del SGBD, principalmente para mostrar las relaciones entre las tablas de la base de datos.

En la tabla anterior, se indican los prefijos que se antepondrán al nombre de cada uno de estos objetos, utilizando la notación *CamelCase*:

Figura 57. **Notación para nombrar otros objetos de bases de datos**



Fuente: elaboración propia.

A continuación se describirá una metodología para definir la estructura de consultas empleando el DML.

3.11.4. Consultas

Para manipular los objetos de la base de datos, se le indica al gestor, seleccionar, insertar, eliminar o actualizar determinada entidad, lo cual lleva a recomendar que estas instrucciones se escriban de forma estructurada, es decir, escribir las consultas dentro de procedimientos o funciones, obteniendo

así las ventajas expuestas al final del punto dos del presente capítulo, ventajas que se obtienen cuando se aplica la programación top-down, estructurada o modular.

Los aspectos a considerar para escribir consultas son:

- Escribir las palabras reservadas del SGBD en letras mayúsculas (independientemente de si el *IDE* utilizado, resalta o no con determinado color de letra estos componentes léxicos): al leer el código desarrollado, esta práctica permite distinguir visualmente, estas palabras de las escritas por el usuario.

- Utilizar sangría: insertar tabulaciones al escribir sentencias de control de flujo, de manejo de errores o excepciones; y al listar los campos afectados, las tablas involucradas, las condiciones de la consulta y las expresiones de orden y/o agrupamiento de las filas afectadas, para distinguir las partes de la consulta.

- Dar un alias (nombre corto) a cada tabla: provee flexibilidad a la consulta al agregar una o más tablas en la misma (en la cláusula FROM o en subconsultas), ya que no tendrá que modificarse la consulta original para agregar el alias, porque ya lo tendrá; especialmente si las tablas agregadas tienen nombres de columnas iguales a la de la tabla inicial.

- Indicar explícitamente los nombres de los campos de cada tabla en las sentencias INSERT y SELECT: al agregar o eliminar columnas en el diseño de una tabla (tomando en cuenta las restricciones que aplique el SGBD, para tales acciones), las consultas existentes no se verán afectadas; siempre y cuando las columnas afectadas no formen parte de

las restricciones (llave primaria o foránea), los atributos agregados acepten valores nulos o tengan un valor establecido por defecto, por mencionar algunas consideraciones.

Con ello, en la sentencia SELECT solamente las columnas necesarias serán transportadas, especialmente si la tabla tiene un número significativo de ellas, reduciendo así el tamaño de la información transferida.

- Dar un alias a las columnas devueltas en la sentencia SELECT: si los registros devueltos son manipulados por una aplicación o se visualizan en forma de tabla dentro del gestor de BDD, se distinguirán los campos que tengan el mismo nombre, entre las tablas involucradas.

  También si el SGBD permite copiar y pegar los resultados de la consulta junto con sus encabezados, en una hoja de cálculo por ejemplo, para reportar los resultados ya no será necesario renombrar los títulos de las columnas cada vez que se ejecute la consulta.

- Utilizar variables en lugar de valores estáticos o múltiples llamadas a la misma subrutina: si en la consulta, una función o una subconsulta es llamada más de una vez con los mismos parámetros, o bien un valor es utilizado varias veces, es buena práctica emplear una constante conteniendo el valor estático, una variable recibiendo el valor devuelto por la función, o una variable tipo tabla que contenga los registros de la subconsulta, para optimizar el rendimiento de la misma, y que el cambio en el valor estático o en los parámetros de la función o subconsulta se realicen una única vez y no las veces que se utilice en el código.

3.12. Presentación de la aplicación

El usuario final no interactúa directamente con el código fuente de la aplicación, sino con el programa que se interrelaciona con la capa de aplicación (capa siete del modelo OSI), por lo que a continuación se indican algunos lineamientos muy generales a tomar en cuenta para presentarle al usuario una interfaz igual de estandarizada que el código que la conforma:

- Procurar que el tipo de letra de las páginas *web* o formularios sea el mismo en toda la aplicación, variando el tamaño y estilo (subrayado, formato de negrita o cursiva), para resaltar títulos de los subtítulos del menú, o para indicar el título de la opción, por ejemplo.

- La alineación del texto también hay que considerarla, procurando justificar tanto el texto como los objetos dentro de una tabla o panel; o bien alineando tanto vertical como horizontalmente los elementos de un mismo formulario.

- Seguir un mismo estilo para los colores y los íconos, por ejemplo utilizar colores pastel o combinar tonalidades de un mismo color, en toda la interfaz. Para los íconos, incluir en la API (*Application Programming Interface*), imágenes con colores similares y del mismo tema, por ejemplo en el sitio Purina Centroamérica, que vende productos para mascotas, los íconos utilizados son del mismo tipo, es decir, no se muestra un perro como dibujo animado, sino la foto de los animales son de un perro y de un gato reales.

Si por la naturaleza o por el objetivo de la aplicación, no es viable aplicar los lineamientos anteriores, se pueden omitir, ya que el objetivo de implementar los mismos es, mantener uniformidad a nivel visual.

Figura 58. **Página de categorías de los productos distribuidos por Purina Centroamérica**

Fuente: elaboración propia.

En las páginas *web*, se utilizan hojas de estilo (CSS, *Cascading Style Sheets*), para aplicar el mismo diseño a los elementos HTML de las páginas, y así no tener que aplicar el mismo formato en cada elemento dentro de cada página *web* del sitio. Esto se logra, incluyendo la hoja de estilo como una librería externa (archivo con extensión .css), la cual contiene los selectores para aplicarlos: por tipo de elemento del HTML, para un *id* determinado, para todos los elementos con el mismo atributo *class*, o combinándolos.

Cuando el selector conectado a un estilo sea una clase, para nombrarlo se sugiere anteponerle el prefijo cls, al nombre de la clase, utilizando la notación con *CamelCase*.

3.13. Reportes

La información almacenada se presenta al usuario en informes, los cuales deberían tener al menos, las siguientes características:

- Mostrar la fecha y hora de los datos reportados.
- Si el sistema tiene un logo, incluirlo en el encabezado del reporte.
- Indicar el número de página, de preferencia utilizando la frase Página n de m, mostrando así el número total de páginas.
- En todos los reportes de la aplicación, alinear el texto y procurar usar el mismo tipo de letra, variando el tamaño y estilo entre en títulos, subtítulos, agrupaciones, totales, subtotales, entre otros.
- Utilizar colores e íconos del mismo estilo.

Dentro de algunas herramientas para elaborar reportes en los lenguajes utilizados para esta metodología se pueden mencionar: *Crystal Reports* para Java y Visual Studio, .netCharting y XtraReports para Visual Studio, *Reporting Services* en *Microsoft* SQL, iReport para Java, PHPReports y Agata en PHP, *Oracle Reports Developer*, y QuickReport utilizando el *IDE* C++ Builder.

Puesto que toda aplicación debe ir acompañada de documentación, tanto técnica como de uso, en los siguientes puntos de este capítulo, se indica los elementos básicos que estos documentos deben contener.

3.14. Documentación interna

La documentación interna sirve de apoyo a las personas del equipo de Desarrollo que le dan mantenimiento a la aplicación, ya que además de contar con un estándar de programación, estos comentarios dentro del código explican las características significativas del programa. A continuación se listan los puntos que debe incluir:

- Al inicio de cada archivo, incluir el nombre del mismo, una breve descripción, nombre del autor, fecha de creación, fecha, autor y descripción del último cambio, número de versión actual y listar prerrequisitos para la correcta ejecución del programa.

- Para cada subrutina indicar, una descripción del propósito de la misma, el algoritmo utilizado para su funcionamiento (cuando son métodos complejos), los valores específicos y el significado que pueden tener los parámetros, así como indicar las partes críticas.

- Dentro de cada subrutina comentar lo que no es obvio, a manera de no repetir lo que realiza el código.

Como recomendación general, poner atención en la ortografía y en la gramática, que si bien estos aspectos no son tomados en cuenta por el compilador o intérprete, por lo tanto no afectan el funcionamiento del programa, sí complementan la buena interpretación y presentación del trabajo (sinónimo de calidad), en conjunto con el buen funcionamiento del mismo.

3.15. Documentación externa

Dentro de la documentación externa, se encuentra la documentación que exige cada etapa de la metodología para el desarrollo de *software* que se haya adoptado, el manual técnico y el manual de usuario.

Los lineamientos para la documentación de las diferentes etapas del proceso de desarrollo, son dictados por cada metodología, por lo que no se describen en el presente trabajo.

El manual técnico, completa la documentación interna, ya que también está enfocada a los desarrolladores del *software*, para darle mantenimiento técnico a la aplicación y disminuir la curva de aprendizaje de las personas que se incorporen al proyecto durante la vida útil del mismo. Por lo tanto, los lineamientos básicos a seguir para elaborarlo son:

- Listar los objetivos del programa, así como sus límites y alcances.
- Describir el funcionamiento de la aplicación, indicando los datos de entrada y de salida.
- Detallar los procesos críticos y complejos.
- Especificar los requisitos técnicos para la instalación y puesta en marcha del programa, tanto a nivel de *software* como a nivel de *hardware*.
- Indicar la estructura de la aplicación, es decir cómo están organizados los componentes como módulos, librerías, contraseñas, bitácoras, tabla con mensajes de error, reportes, archivos de configuración, entre otros.
- Explicar las modificaciones por cada cambio de versión en las utilidades, si el detalle de las modificaciones no se gestiona en otra herramienta para control de versiones (como Subversion, Git, Mercurial, Bazaar, Visual Studio Team System, entre otros).

El documento que indica cómo operar el sistema, es el manual de usuario; el cual describe los requisitos y procedimiento de instalación, y también cada opción del programa. Este documento enfocado al usuario, quien desconoce el *software*, tendrá los siguientes puntos:

- Lista de los objetivos y ventajas del programa, así como sus límites y alcances.
- Requerimientos mínimos para la instalación, a nivel de *software* y *hardware*.
- Proceso de instalación y desinstalación.
- Descripción detallada de cómo utilizar cada opción del programa.
- Posibles inconvenientes y cómo solucionarlos.

Especialmente en el manual de usuario, se deben incluir imágenes de la aplicación, lo cual facilita enormemente la explicación de cada punto.

Parte del manual de usuario, también puede incluirse en la llamada ayuda en línea, que consiste en asistir al operador en cada opción de la aplicación, mostrando en pantalla un texto que explique brevemente el funcionamiento de algún elemento, a solicitud del usuario (presionando una tecla o dando un clic sobre algún ícono de ayuda).

Este tipo de manual no debe confundirse con lo que se conoce como manual de procedimientos, ya que este documento indica cómo operar la aplicación, no la lógica del negocio específicamente.

En la metodología Extilo Regular 9002, se lista lo más utilizado y lo existente a la fecha en los lenguajes utilizados como base, si surgen nuevos objetos, tipos o paradigmas de programación, el desarrollador será capaz de

asignar el prefijo para nombrarlos y proponer los lineamientos más adecuados, tomando de referencia los estándares dispuestos en el presente trabajo.

Si se utilizan plantillas ya definidas por el lenguaje, como las que provee .Net para inicio de sesión (Login), registro de usuarios (CreateUserWizard), reinicio de contraseña (ChangePassword), entre otras, en donde a los controles no se les puede cambiar el nombre que ya tienen definido, porque de lo contrario se pierde el funcionamiento proveído por la plantilla, no se podrá aplicar este estándar, pero el código seguirá estando estandarizado, porque cumplirá con el estándar requerido por el *IDE*, el cual a su vez será conocido por los usuarios de la herramienta de programación.

A continuación se presenta el código fuente de los prototipos que se desarrollaron aplicando esta metodología, empleando los lenguajes de programación: Visual Basic .Net, Visual C#, y sistemas de gestión de base de datos: Oracle y *Microsoft* SQL.

4. CÓDIGO FUENTE QUE GESTIONA USUARIOS DE UN SISTEMA Y PERMITE RECUPERAR LA CONTRASEÑA DE UN USUARIO

Los siguientes dos prototipos permiten agregar, editar, eliminar y consultar datos de usuarios; la información que se gestiona para cada usuario es un nombre de usuario, una contraseña, una cuenta de correo electrónico, nombres, apellidos y una o más preguntas secretas con su respectiva respuesta. Cada usuario puede modificar sus propios datos, siempre y cuando el nuevo nombre de usuario que se indique no exista en la base de datos; solamente el usuario ADMIN puede agregar, eliminar, consultar y reiniciarles contraseña a todos los usuarios.

La aplicación también tiene la opción para recuperar la contraseña de determinado usuario, para lo cual se debe indicar el nombre de usuario, el *email* registrado en el sistema, y responder a una de las preguntas secretas almacenadas que se muestran aleatoriamente; los datos indicados deben coincidir con lo almacenado en el sistema para cada usuario.

Es importante mencionar que por confidencialidad, en la base de datos las contraseñas se encuentran cifradas.

En estos prototipos no se maneja bitácora, por ser una aplicación de ejemplo, pero los programas deben llevar algún tipo de registro para toda acción que se realice, especialmente para monitorizar la inserción, modificación y eliminación de datos.

Aunque en la aplicación, la llave para cifrar las credenciales de acceso a la base de datos y la cuenta de correo electrónico desde la que se envían los *emails*, no estén cifrados ni ocultos, se recomienda que en todo sistema estos datos estén cifrados o bien almacenados en algún lugar de acceso restringido.

Para implementar cada prototipo, además de tener instalados los sistemas de gestión de base de datos y la plataforma de los lenguajes de programación, se requiere tener una conexión a internet, la cuenta de correo electrónico notificacion.sistema.gt@gmail.com, que siempre exista en la base de datos el usuario ADMIN y 5 preguntas secretas.

Aunque en los prototipos se utilicen diferentes lenguajes de programación y distintos sistemas de gestión de base de datos, la lógica para gestionar los usuarios y recuperar la contraseña, es la misma en ambos ejemplos. Por ello, para el primer prototipo se describe el funcionamiento del programa, y para el otro solamente se muestra en el apéndice del presente trabajo, las pantallas y el código, que es lo que varía con respecto al primer modelo.

4.1. Prototipo # 1: Visual C# - *Microsoft* SQL

En la base de datos se tienen 3 tablas, una para almacenar los datos de los usuarios, otra que almacena las preguntas secretas, y una tercera en donde se guardan las respuestas por pregunta de cada usuario.

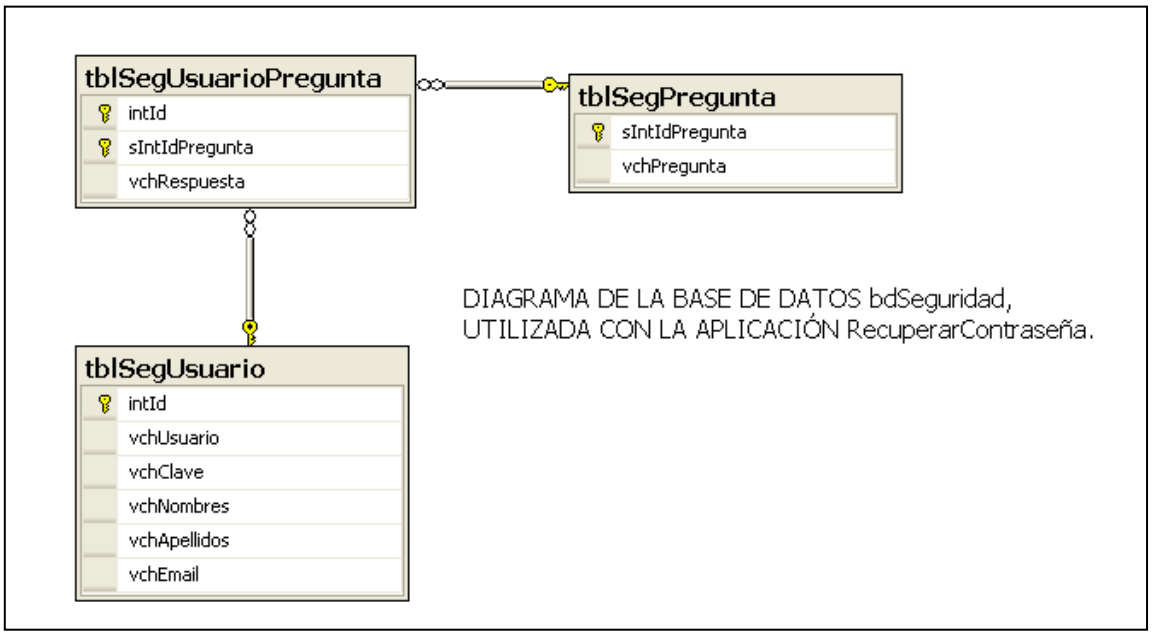
En las siguientes figuras se muestra el diagrama de la base de datos, y el *script* utilizado para crear el usuario ADMIN (contraseña superUs3r) y las preguntas secretas.

Figura 59. ***Script* para crear el usuario ADMIN**

```
0001  --Crear el usuario administrador
0002  INSERT INTO tblSegUsuario (intId, vchUsuario, vchClave,
0003              vchNombres, vchApellidos, vchEmail)
0004  VALUES (1, 'ADMIN', 'ozkyOTBVsyR+80kaeQnuBw==',
0005              'USUARIO', 'ADMINISTRADOR', UPPER('notificacion.sistema.gt@gmail.com'))
```

Fuente: elaboración propia.

Figura 60. **Diagrama almacenado en la base de datos bajo el nombre diagSegUsuario**



Fuente: elaboración propia.

Figura 61. ***Script* para crear las preguntas secretas**

```
0001  --Llenar la tabla con las preguntas
0002  INSERT INTO tblSegPregunta (sIntIdPregunta, vchPregunta)
0003        VALUES (1, 'MENCIONE A UN AMIGO(A):')
0004  INSERT INTO tblSegPregunta (sIntIdPregunta, vchPregunta)
0005        VALUES (2, '¿CUÁL ES SU PLATILLO FAVORITO?')
0006  INSERT INTO tblSegPregunta (sIntIdPregunta, vchPregunta)
0007        VALUES (3, '¿QUÉ ES LO QUE MÁS RECUERDA DE LOS 0-5 AÑOS?')
0008  INSERT INTO tblSegPregunta (sIntIdPregunta, vchPregunta)
0009        VALUES (4, '¿QUÉ LUGAR TURÍSTICO LE GUSTARÍA CONOCER?')
0010  INSERT INTO tblSegPregunta (sIntIdPregunta, vchPregunta)
0011        VALUES (5, '¿EN DÓNDE ESTÁ UBICADO SU RESTAURANTE FAVORITO?')
```

Fuente: elaboración propia.

Se crearon procedimientos almacenados para guardar, actualizar, eliminar y buscar usuarios, y otro más para buscar las preguntas secretas. En todos los procedimientos después de insertar, actualizar o eliminar registros, se consulta el valor de la función no determinista @@ERROR, para verificar si hubo error al ejecutar la instrucción de SQL.

El procedimiento pcGuardarUsuario recibe como parámetros de entrada todos los datos del nuevo usuario, y como parámetro de salida se tiene el *Id* asignado al usuario al momento de almacenar el registro en la tabla. Si el nombre de usuario enviado no existe en la tabla, se guardan los datos, de lo contrario se retorna 0, indicando que no se puede crear el usuario.

Figura 62. **Procedimiento pcGuardarUsuario (parte 1 de 2)**

```
0001  ALTER PROCEDURE pcGuardarUsuario
0002  --Crea un usuario
0003  (     @pRIntId As INT OUTPUT,              --Id del usuario
0004        @pVVchUsuario As VARCHAR(10),        --Nombre del usuario
0005        @pVVchClave As VARCHAR(1000),        --Clave cifrada
0006        @pVVchNombres As VARCHAR(50),        --Nombres del usuario
0007        @pVVchApellidos As VARCHAR(50),      --Apellidos del usuario
```

Fuente: elaboración propia.

Figura 63. **Procedimiento pcGuardarUsuario (parte 2 de 2)**

```
0008            @pVVchEmail As VARCHAR(50)              --Email del usuario
0009    )AS
0010    BEGIN
0011            --Consultar si el usuario existe
0012            IF NOT EXISTS (SELECT Usuario.intId As Id
0013                            FROM tblSegUsuario Usuario
0014                            WHERE Usuario.vchUsuario = LTRIM(RTRIM(@pVVchUsuario))
0015            )
0016            BEGIN
0017                    SELECT @pRIntId = MAX(intId) + 1 FROM tblSegUsuario
0018                    INSERT INTO tblSegUsuario (intId, vchUsuario, vchClave,
0019                            vchNombres, vchApellidos, vchEmail)
0020                    VALUES (@pRIntId, UPPER(LTRIM(RTRIM(@pVVchUsuario))),
0021                            LTRIM(RTRIM(@pVVchClave)),
0022                            UPPER(LTRIM(RTRIM(@pVVchNombres))),
0023                            UPPER(LTRIM(RTRIM(@pVVchApellidos))),
0024                            UPPER(LTRIM(RTRIM(@pVVchEmail))))
0025
0026                    IF @@ERROR <> 0
0027                    BEGIN
0028                            RETURN 3 --Error del SQL
0029                    END
0030                    ELSE
0031                    BEGIN
0032                            RETURN 1 --El usuario fue creado
0033                    END
0034            END
0035            ELSE
0036            BEGIN
0037                    RETURN 0 --El usuario ya existe
0038            END
0039    END
```

Fuente: elaboración propia.

pcModificarUsuario es el procedimiento para modificar datos de usuarios, recibe como parámetros de entrada todos los datos del usuario a modificar, y 5 cadenas con las respuestas indicadas por el usuario.

Si existe un usuario con el *Id* recibido como parámetro, se modifican los datos, de lo contrario se retorna 0, indicando que no se puede actualizar el usuario. Si hay que actualizar las respuestas secretas del usuario, se borra lo que exista en tblSegUsuarioPregunta para ese usuario, y dependiendo de la

respuesta indicada se guarda el dato asociándolo al código de la pregunta que le corresponda.

Figura 64. **Procedimiento pcModificarUsuario (parte 1 de 3)**

```
0001  ALTER PROCEDURE pcModificarUsuario
0002  --Actualiza los datos del usuario
0003  (       @pVIntId As INT,                        --Id del usuario
0004          @pVVchUsuario As VARCHAR(10),           --Nombre del usuario
0005          @pVVchClave As VARCHAR(1000),           --Clave cifrada
0006          @pVVchNombres As VARCHAR(50),           --Nombres del usuario
0007          @pVVchApellidos As VARCHAR(50),         --Apellidos del usuario
0008          @pVVchEmail As VARCHAR(50),             --Email del usuario
0009          @pVVchRespuesta1 As VARCHAR(300),       --Respuesta 1
0010          @pVVchRespuesta2 As VARCHAR(300),       --Respuesta 2
0011          @pVVchRespuesta3 As VARCHAR(300),       --Respuesta 3
0012          @pVVchRespuesta4 As VARCHAR(300),       --Respuesta 4
0013          @pVVchRespuesta5 As VARCHAR(300),       --Respuesta 5
0014          @pVsIntOpcion As SMALLINT               --0: No actualizar las respuestas
0015                                                  --1: Sí actualizar las respuestas
0016  )AS
0017  BEGIN
0018          --Consultar si el usuario existe
0019          IF EXISTS (SELECT Usuario.intId As Id
0020                  FROM tblSegUsuario Usuario
0021                  WHERE Usuario.intId = @pVIntId
0022          )
0023          BEGIN
```

Fuente: elaboración propia.

Figura 65. **Procedimiento pcModificarUsuario (parte 2 de 3)**

```
0024                  --Actualizar los datos del usuario
0025                  UPDATE Usuario
0026                  SET Usuario.vchUsuario = UPPER(LTRIM(RTRIM(@pVVchUsuario))),
0027                          Usuario.vchClave = LTRIM(RTRIM(@pVVchClave)),
0028                          Usuario.vchNombres = UPPER(LTRIM(RTRIM(@pVVchNombres))),
0029                          Usuario.vchApellidos = UPPER(LTRIM(RTRIM(@pVVchApellidos))),
0030                          Usuario.vchEmail = UPPER(LTRIM(RTRIM(@pVVchEmail)))
0031                  FROM tblSegUsuario Usuario
0032                  WHERE Usuario.intId = @pVIntId
0033
```

Fuente: elaboración propia.

Figura 66. **Procedimiento pcModificarUsuario (parte 3 de 3)**

```
0034                    --Actualizar las respuestas del usuario
0035                    IF @pVsIntOpcion = 1
0036                    BEGIN
0037                            DELETE FROM tblSegUsuarioPregunta WHERE intId = @pVIntId
0038                            IF (@pVVchRespuesta1 <> '') --Respuesta 1
0039                            BEGIN
0040                                    INSERT INTO tblSegUsuarioPregunta (intId, sIntIdPregunta,
0041                                            vchRespuesta)
0042                                    VALUES (@pVIntId, 1, @pVVchRespuesta1)
0043                            END
0044                            IF (@pVVchRespuesta2 <> '') --Respuesta 2
0045                            BEGIN
0046                                    INSERT INTO tblSegUsuarioPregunta (intId, sIntIdPregunta,
0047                                            vchRespuesta)
0048                                    VALUES (@pVIntId, 2, @pVVchRespuesta2)
0049                            END
0050                            IF (@pVVchRespuesta3 <> '') --Respuesta 3
0051                            BEGIN
0052                                    INSERT INTO tblSegUsuarioPregunta (intId, sIntIdPregunta,
0053                                            vchRespuesta)
0054                                    VALUES (@pVIntId, 3, @pVVchRespuesta3)
0055                            END
0056                            IF (@pVVchRespuesta4 <> '') --Respuesta 4
0057                            BEGIN
0058                                    INSERT INTO tblSegUsuarioPregunta (intId, sIntIdPregunta,
0059                                            vchRespuesta)
0060                                    VALUES (@pVIntId, 4, @pVVchRespuesta4)
0061                            END
0062                            IF (@pVVchRespuesta5 <> '') --Respuesta 5
0063                            BEGIN
0064                                    INSERT INTO tblSegUsuarioPregunta (intId, sIntIdPregunta,
0065                                            vchRespuesta)
0066                                    VALUES (@pVIntId, 5, @pVVchRespuesta5)
0067                            END
0068                    END
0069
0070                    IF @@ERROR <> 0
0071                    BEGIN
0072                            RETURN 3 --Error del SQL
0073                    END
0074                    ELSE
0075                    BEGIN
0076                            RETURN 1 --Los datos del usuario se actualizaron correctamente
0077                    END
0078            END
0079            ELSE
0080            BEGIN
0081                    RETURN 0 --El usuario no existe
0082            END
0083    END
```

Fuente: elaboración propia.

Para eliminar un usuario se utiliza el procedimiento pcEliminarUsuario, el cual primero elimina las preguntas que pudiera tener asociadas el usuario, y luego elimina el registro en tblSegUsuario, cuyo *Id* sea igual al valor recibido por parámetro, siempre y cuando no sea el usuario ADMIN.

Figura 67. **Procedimiento pcEliminarUsuario**

```
0001  ALTER PROCEDURE pcEliminarUsuario
0002  --Elimina un usuario
0003  (     @pVIntId As INT          --Id del usuario a eliminar
0004  )AS
0005  BEGIN
0006        --Consultar si el usuario existe
0007        IF EXISTS (SELECT Usuario.intId As Id
0008              FROM tblSegUsuario Usuario
0009              WHERE Usuario.intId = @pVIntId
0010                    AND @pVIntId <> 1 --El usuario ADMIN no se puede eliminar
0011        )
0012        BEGIN
0013              DELETE PreguntaUsr
0014              FROM tblSegUsuarioPregunta PreguntaUsr
0015              WHERE PreguntaUsr.intId = @pVIntId
0016
0017              DELETE Usuario
0018              FROM tblSegUsuario Usuario
0019              WHERE Usuario.intId = @pVIntId
0020
0021              IF @@ERROR <> 0
0022              BEGIN
0023                    RETURN 3 --Error del SQL
0024              END
0025              ELSE
0026              BEGIN
0027                    RETURN 1 --El usuario se eliminó exitosamente
0028              END
0029        END
0030        ELSE
0031        BEGIN
0032              RETURN 0 --El usuario no existe
0033        END
0034  END
```

Fuente: elaboración propia.

En las búsquedas de datos se emplean dos procedimientos almacenados, pcBuscarPreguntas (figura 68) devuelve las preguntas almacenadas en la tabla

tblSegPregunta, y pcBuscarUsuario (figuras 69 - 74) devuelve datos de la tabla tblSegUsuario con base al valor del parámetro de entrada @pVsIntOpcion. @pVsIntOpcion puede tomar cinco valores, de 0 a 4, si el parámetro tiene otro valor se retorna 2, indicando que la opción no es válida. Este procedimiento retorna 1 cuando sí se encuentran datos para la opción solicitada.

Figura 68. **Procedimiento pcBuscarPreguntas**

```
0001  ALTER PROCEDURE pcBuscarPreguntas
0002  --Busca las preguntas a mostrarle al usuario
0003  AS
0004  BEGIN
0005          IF EXISTS (SELECT * FROM tblSegPregunta Pregunta)
0006          BEGIN
0007                  SELECT Pregunta.sIntIdPregunta As ID,
0008                          Pregunta.vchPregunta As PREGUNTA
0009                  FROM tblSegPregunta Pregunta
0010                  ORDER BY ID
0011
0012                  IF @@ERROR <> 0
0013                  BEGIN
0014                          RETURN 3 --Error del SQL
0015                  END
0016                  ELSE
0017                  BEGIN
0018                          RETURN 1 --Sí hay datos
0019                  END
0020          END
0021          ELSE
0022          BEGIN
0023                  RETURN 0 --No hay datos
0024          END
0025  END
```

Fuente: elaboración propia.

Figura 69. **Procedimiento pcBuscarUsuario (parte 1 de 6)**

```
0001  ALTER PROCEDURE pcBuscarUsuario
0002  --Busca el usuario, el usuario y el email, el usuario y clave indicados, o bien los datos de todos los
0003  --usuarios
0004  (     @pVIntId As INT,                    --Id del usuario
0005        @pVVchUsuario As VARCHAR(10),       --Nombre del usuario
0006        @pVVchClave As VARCHAR(1000),       --Clave cifrada
0007        @pVVchEmail AS VARCHAR(50),         --Email
0008        @pVsIntOpcion SMALLINT              --0: para consultar si el usuario y el mail
0009                                                --corresponden a la misma persona
0010                                            --1: para consultar si el nombre de usuario
0011                                                --ya existe
0012                                            --2: si el usuario y la clave son correctos,
0013                                                --se devuelven todos los datos
0014                                            --3: buscar el usuario por ID
0015                                            --4: buscar los datos de todos los usuarios
0016
0017  )AS
0018  BEGIN
```

Fuente: elaboración propia.

Figura 70. **Procedimiento pcBuscarUsuario (parte 2 de 6)**

```
0019        IF @pVsIntOpcion = 0 --Consultar si el usuario y el email coinciden
0020        BEGIN
0021              IF EXISTS (SELECT Usuario.intId As Id
0022                    FROM tblSegUsuario Usuario
0023                    WHERE Usuario.vchUsuario = LTRIM(RTRIM(@pVVchUsuario))
0024                          AND Usuario.vchEmail = LTRIM(RTRIM(@pVVchEmail))
0025              )
0026              BEGIN
0027                    SELECT TOP 1 Usuario.intId As Id,
0028                          Usuario.vchUsuario As Usuario,
0029                          Usuario.vchNombres As Nombres,
0030                          Usuario.vchApellidos As Apellidos,
0031                          Usuario.vchEmail As EMail,
0032                          COALESCE(Pregunta.vchPregunta, '') As Pregunta,
0033                          COALESCE(Resp.vchRespuesta, '') As Respuesta
0034                    FROM tblSegUsuario Usuario
0035                    LEFT JOIN tblSegUsuarioPregunta Resp ON (Resp.intId = Usuario.intId)
0036                    LEFT JOIN tblSegPregunta Pregunta
0037                          ON (Pregunta.sIntIdPregunta = Resp.sIntIdPregunta)
0038                    WHERE Usuario.vchUsuario = LTRIM(RTRIM(@pVVchUsuario))
0039                          AND Usuario.vchEmail = LTRIM(RTRIM(@pVVchEmail))
0040                    ORDER BY NEWID()
```

Fuente: elaboración propia.

En la figura anterior, se muestra el código que se ejecuta cuando @pVsIntOpcion tiene valor de 0: si existe un usuario cuyo nombre y correo electrónico sean los indicados por parámetro, se devuelven los datos del usuario (excepto la clave) junto con una de las preguntas que el usuario tenga en tblSegUsuarioPregunta, elegida aleatoriamente.

Cuando @pVsIntOpcion tiene valor de 1, si ya existe un usuario con el mismo nombre que indica @pVVchUsuario, no se retornan datos, solamente 1 indicando que ya existe ese nombre de usuario.

Figura 71. **Procedimiento pcBuscarUsuario (parte 3 de 6)**

```
0041
0042                        RETURN 1 --El usuario y el email sí son correctos
0043                END
0044        END
0045        ELSE
0046        BEGIN
0047                --Consultar si el usuario existe
0048                IF @pVsIntOpcion = 1
0049                BEGIN
0050                        IF EXISTS (SELECT Usuario.intId As Id
0051                                FROM tblSegUsuario Usuario
0052                                WHERE Usuario.vchUsuario =
0053                                                LTRIM(RTRIM(@pVVchUsuario))
0054                                        AND Usuario.intId <> @pVIntId
0055                        )
0056                        BEGIN
0057                                RETURN 1 --El usuario sí existe
0058                        END
0059                END
0060                ELSE
```

Fuente: elaboración propia.

Si @pVsIntOpcion tiene valor de 2, se devuelven todos los datos del usuario cuyo nombre y clave sean los indicados por parámetro.

Figura 72. **Procedimiento pcBuscarUsuario (parte 4 de 6)**

```
0061                    BEGIN
0062                        --Consultar si el usuario y la clave son correctos
0063                        IF @pVsIntOpcion = 2
0064                        BEGIN
0065                            IF EXISTS (SELECT Usuario.intId As Id
0066                                FROM tblSegUsuario Usuario
0067                                WHERE Usuario.vchUsuario =
0068                                        LTRIM(RTRIM(@pVVchUsuario))
0069                                    AND Usuario.vchClave =
0070                                        LTRIM(RTRIM(@pVVchClave))
0071                            )
0072                            BEGIN
0073                                SELECT Usuario.intId As Id,
0074                                    Usuario.vchUsuario As Usuario,
0075                                    Usuario.vchClave As Clave,
0076                                    Usuario.vchNombres As Nombres,
0077                                    Usuario.vchApellidos As Apellidos,
0078                                    Usuario.vchEmail As EMail
0079                                FROM tblSegUsuario Usuario
0080                                WHERE Usuario.vchUsuario =
0081                                        LTRIM(RTRIM(@pVVchUsuario))
0082                                    AND Usuario.vchClave =
0083                                        LTRIM(RTRIM(@pVVchClave))
0084
0085                                RETURN 1 --El usuario y la clave sí son correctos
0086                            END
0087                        END
0088                        ELSE
```

Fuente: elaboración propia.

El valor 3 también es válido para @pVsIntOpcion, ya que si existe el usuario cuyo identificador sea el que indica @pVIntId, se devuelve el *id*, usuario, nombres, apellidos y correo electrónico.

Figura 73. **Procedimiento pcBuscarUsuario (parte 5 de 6)**

```
0089                    BEGIN
0090                        --Buscar el usuario por Id
0091                        IF @pVsIntOpcion = 3
0092                        BEGIN
0093                            IF EXISTS (SELECT Usuario.intId As Id
0094                                FROM tblSegUsuario Usuario
0095                                WHERE Usuario.intId = @pVIntId
0096                            )
0097                            BEGIN
0098                                SELECT Usuario.intId As Id,
0099                                    Usuario.vchUsuario As Usuario,
0100                                    Usuario.vchNombres As Nombres,
0101                                    Usuario.vchApellidos As Apellidos,
0102                                    Usuario.vchEmail As EMail
0103                                FROM tblSegUsuario Usuario
0104                                WHERE Usuario.intId = @pVIntId
0105
0106                                RETURN 1 --El usuario sí existe
0107                            END
0108                        END
0109                        ELSE
```

Fuente: elaboración propia.

Se retornan los datos de todos los usuarios (excepto la clave), junto con el campo TIENE PREGUNTA SECRETA, cuyo valor depende de si en tblSegUsuarioPregunta existe más de alguna pregunta secreta para cada usuario, cuando @pVsIntOpcion tiene valor de 4; los datos de esta consulta se devuelven ordenados por *Id*.

Figura 74. **Procedimiento pcBuscarUsuario (parte 6 de 6)**

```
0110                        BEGIN
0111                            --Buscar los datos de todos los usuarios
0112                            IF @pVsIntOpcion = 4
0113                            BEGIN
0114                              SELECT Usuario.intId As ID,
0115                                    Usuario.vchUsuario As USUARIO,
0116                                    Usuario.vchNombres As NOMBRES,
0117                                    Usuario.vchApellidos As
0118                                        APELLIDOS,
0119                                    Usuario.vchEmail As [E-MAIL],
```

Continuación de la figura 74.

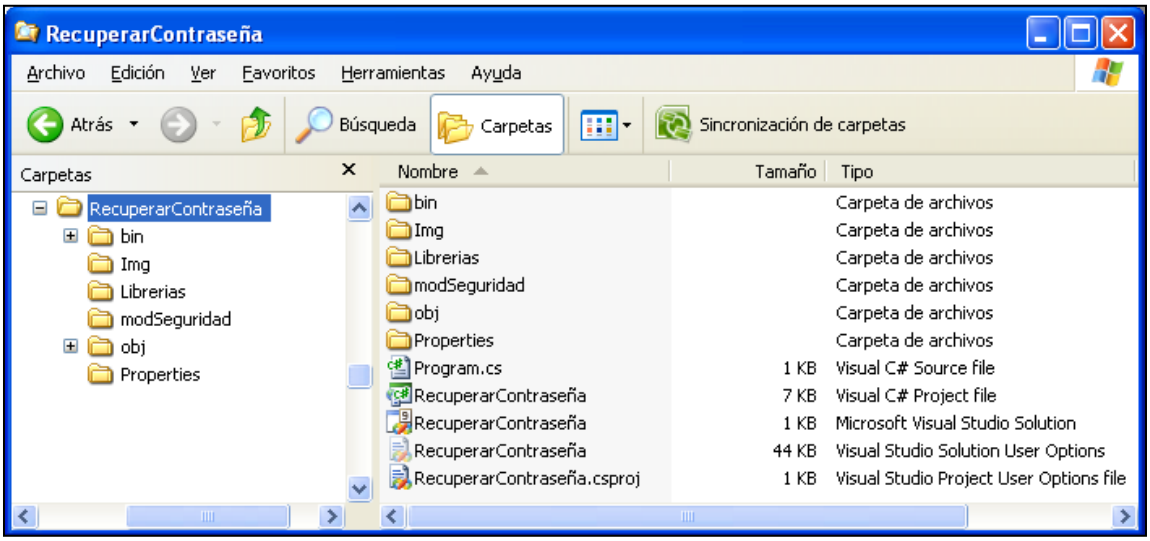
```
0120                                                    CASE
0121                                                      WHEN EXISTS (SELECT
0122                                                              Resp.*
0123                                                              FROM tblSegUsuarioPregunta
0124                                                                      Resp
0125                                                              WHERE
0126                                                                      Resp.intId = Usuario.intId)
0127                                                      THEN 'SÍ'
0128                                                      ELSE 'NO'
0129                                                    END As
0130                                                              [TIENE PREGUNTA SECRETA]
0131                                            FROM tblSegUsuario Usuario
0132                                            ORDER BY ID
0133
0134                                            RETURN 1 --Se consultaron los datos de usuarios
0135                                          END
0136                                          ELSE
0137                                          BEGIN
0138                                                    RETURN 2 --No es una opción válida
0139                                          END
0140                                  END
0141                          END
0142                  END
0143          END
0144
0145          IF @@ERROR <> 0
0146          BEGIN
0147                  RETURN 3 --Error del SQL
0148          END
0149          ELSE
0150          BEGIN
0151                  RETURN 0 --No hay datos de usuarios
0152          END
0153  END
0154
```

Fuente: elaboración propia.

Mostrado el código desarrollado en *Microsoft* SQL Server 2008 edición Express, en los siguientes párrafos se muestra el código de la aplicación realizada en el lenguaje Visual C#, utilizando el *IDE* Microsoft Visual Studio 2008 edición *Professional* de evaluación, la cual se conecta a la base de datos bdSeguridad para enviar y obtener la información de los usuarios, a través de los procedimientos almacenados descritos anteriormente.

Dentro del folder del proyecto, se distribuyen por carpeta, las imágenes utilizadas dentro del programa, una librería y los formularios, como se muestra en las siguientes figuras.

Figura 75. **Carpeta del proyecto RecuperarContraseña**

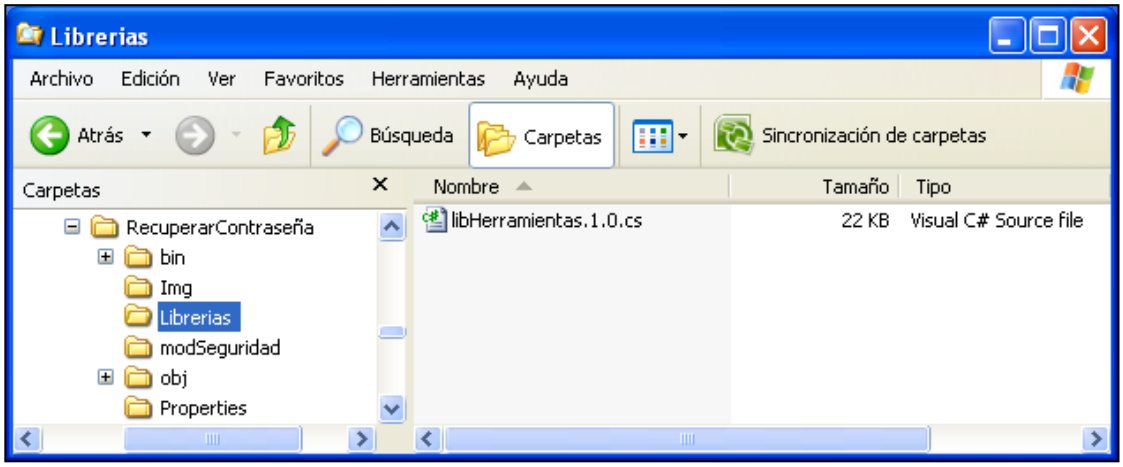


Fuente: elaboración propia.

Figura 76. **Carpeta del proyecto RecuperarContraseña, que contiene las imágenes utilizadas en los formularios**

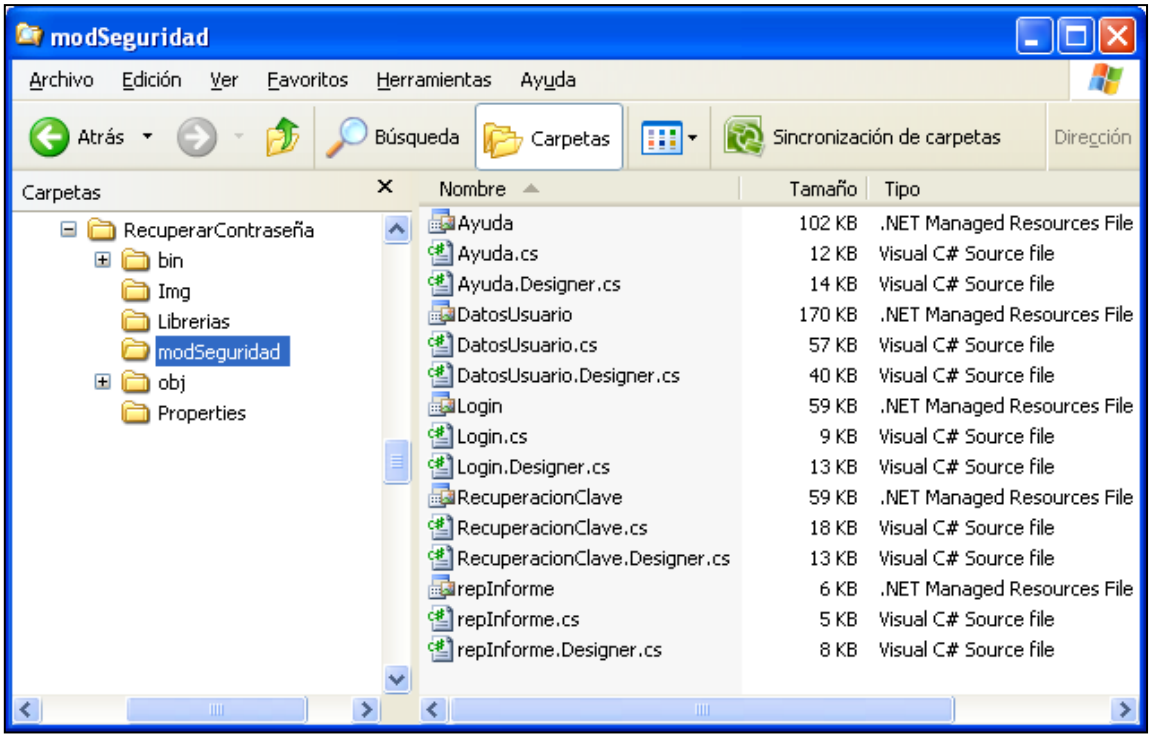
Fuente: elaboración propia.

Figura 77. **Carpeta del proyecto RecuperarContraseña, que contiene la librería utilizada en los formularios**



Fuente: elaboración propia.

Figura 78. **Carpeta del proyecto RecuperarContraseña, que contiene los formularios de la aplicación**



Fuente: elaboración propia.

El hecho que sean varios formularios con verificaciones en común, promovió a tener la clase modHerramientas, en donde se tiene un atributo privado tipo *string*, que almacena la llave utilizada para cifrar las contraseñas, y los siguientes métodos y funciones públicos que son invocados en los formularios:

- fPbBolEsLetra, fPbBolEsVocalTildada, fPbBolEsNumero: reciben la tecla presionada, y cada función retorna verdadero si el valor de la misma era el esperado.

Figura 79. **Definición de las funciones fPbBolEsLetra, fPbBolEsVocalTildada, fPbBolEsNumero, de libHerramientas.1.0**

```
0032  //Indica si la tecla recibida es una letra minúscula o mayúscula, no incluye vocales tildadas
0033  public bool fPbBolEsLetra(ref KeyPressEventArgs pRKpTecla) {
0034        //Letras mayúsculas
0035     if ((pRKpTecla.KeyChar >= 65 && pRKpTecla.KeyChar <= 90) || pRKpTecla.KeyChar == 'Ñ' ||
0036        //Letras minúsculas
0037        (pRKpTecla.KeyChar >= 97 && pRKpTecla.KeyChar <= 122) || pRKpTecla.KeyChar == 'ñ') {
0038        return true;
0039     }
0040     else {
0041        return false;
0042     }
0043  }
0044  //Indica si la tecla recibida es una vocal tildada minúscula o mayúscula
0045  public bool fPbBolEsVocalTildada(ref KeyPressEventArgs pRKpTecla) {
0046              //Vocales minúsculas tildadas
0047     if (pRKpTecla.KeyChar == 225 || pRKpTecla.KeyChar == 233 || pRKpTecla.KeyChar == 237 ||
0048         pRKpTecla.KeyChar == 243 || pRKpTecla.KeyChar == 250 ||
0049              //Vocales mayúsculas tildadas
0050        pRKpTecla.KeyChar == 193 || pRKpTecla.KeyChar == 201 || pRKpTecla.KeyChar == 205 ||
0051        pRKpTecla.KeyChar == 211 || pRKpTecla.KeyChar == 218) {
0052        return true;
0053     }
0054     else {
0055        return false;
0056     }
0057  }
0058  //Indica si la tecla recibida es un número
0059  public bool fPbBolEsNumero(ref KeyPressEventArgs pRKpTecla) {
0060     //números
0061     if (pRKpTecla.KeyChar >= 48 && pRKpTecla.KeyChar <= 57) {
0062        return true;
0063     }
0064     else {
0065        return false;
0066     }
0067  }
```

Fuente: elaboración propia.

- pcPbValidarCaracteres: dependiendo del valor de pVsIntOpcion, valida si la tecla presionada es un carácter alfanumérico, alfabético, carácter válido para una cuenta de correo, número, espacio o retorno de carro.

Figura 80. **Definición del procedimiento pcPbValidarCaracteres, de libHerramientas.1.0 (parte 1 de 2)**

```
0069  //Restringe el ingreso de números, letras y los carácteres válidos
0070  public void pcPbValidarCaracteres(ref KeyPressEventArgs pRKpTecla,
0071  Control pRCtrlEnfoque,
0072  Int16 pVsIntOpcion) {
0073  //Variable con el mensaje a desplegar
0074  string vrLStrMensaje = "";
0075
0076  switch (pVsIntOpcion) {
0077     case 1: //Carácteres alfanuméricos
0078        if ( fPbBolEsLetra(ref pRKpTecla) || //Letras mayúsculas y minúsculas
0079           fPbBolEsNumero(ref pRKpTecla) || //números
0080           pRKpTecla.KeyChar == 45 || //carácter -
0081           pRKpTecla.KeyChar == 95 || //carácter _
0082           pRKpTecla.KeyChar == 8) { //retroceso (backspace)
0083           //Carácter válido
0084           pVsIntOpcion = 0;
0085        }
0086        vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS, NÚMEROS " +
0087              "Y LOS CARÁCTERES - _";
0088        break;
0089     case 2: //Carácteres alfabéticos
0090        if (fPbBolEsLetra(ref pRKpTecla) || //Letras mayúsculas y minúsculas
0091           pRKpTecla.KeyChar == 32 || //espacio
0092           fPbBolEsVocalTildada(ref pRKpTecla) || //Vocales minúsculas y mayúsculas tildadas
0093           pRKpTecla.KeyChar == 8) {  //retroceso (backspace)
0094           //Carácter válido
0095           pVsIntOpcion = 0;
0096        }
0097        vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS Y ESPACIOS " +
0098              "EN BLANCO";
0099        break;
0100     case 3: //Carácteres para cuenta de correo
0101        if (fPbBolEsLetra(ref pRKpTecla) || //Letras mayúsculas y minúsculas
0102           fPbBolEsNumero(ref pRKpTecla) || //números
0103           pRKpTecla.KeyChar == 95 || //carácter _
0104           pRKpTecla.KeyChar == 46 || //punto
0105           pRKpTecla.KeyChar == 64 || //arroba
0106           pRKpTecla.KeyChar == 8) {  //retroceso (backspace)
0107           //Carácter válido
0108           pVsIntOpcion = 0;
0109        }
0110        vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS, NÚMEROS " +
0111              "Y LOS CARÁCTERES . _ @";
0112        break;
0113     case 4: //Números
0114        if (fPbBolEsNumero(ref pRKpTecla) || //números
0115           pRKpTecla.KeyChar == 8) { //retroceso (backspace)
```

Fuente: elaboración propia.

Figura 81. **Definición del procedimiento pcPbValidarCaracteres, de libHerramientas.1.0 (parte 2 de 2)**

```
0116                pVsIntOpcion = 0;  //Carácter válido
0117             }
0118             vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE NÚMEROS";
0119             break;
0120          case 5: //Carácteres alfanuméricos y espacio
0121             if (fPbBolEsLetra(ref pRKpTecla) || //Letras mayúsculas y minúsculas
0122                fPbBolEsNumero(ref pRKpTecla) || //números
0123                pRKpTecla.KeyChar == 45 || //carácter -
0124                pRKpTecla.KeyChar == 95 || //carácter _
0125                pRKpTecla.KeyChar == 8 || //retroceso (backspace)
0126                pRKpTecla.KeyChar == 32 || //espacio
0127                fPbBolEsVocalTildada(ref pRKpTecla)) {  //Vocales minúsculas y mayúsculas tildadas
0128                pVsIntOpcion = 0;  //Carácter válido
0129             }
0130             vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS, NÚMEROS, " +
0131                   "ESPACIOS EN BLANCO Y LOS CARÁCTERES - _";
0132             break;
0133          case 6: //espacio
0134             if (pRKpTecla.KeyChar == 32) {   //espacio
0135                pVsIntOpcion = 0; //Carácter válido
0136             }
0137             vrLStrMensaje = "SOLAMENTE SE PERMITE PRESIONAR LA BARRA " +
0138                   "ESPACIADORA Y LA TECLA ENTER";
0139             break;
0140          default:
0141             pVsIntOpcion = -1; //Carácter no válido
0142             break;
0143       }
0144       //Si es un carácter válido
0145       if (pVsIntOpcion == 0) {
0146          pRKpTecla.Handled = false;
0147       }
0148       else
0149       {   //Si se presiona la tecla ENTER
0150          if (pRKpTecla.KeyChar == '\r') {
0151             //Si el control tiene el método Focus
0152             if (pRCtrlEnfoque.CanFocus) {
0153                pRCtrlEnfoque.Focus(); //se pasa el cursor al control indicado por parámetro
0154             }
0155          }
0156          else {
0157             //Si la tecla presionada no es válida, se muestra el mensaje correspondiente
0158             MessageBox.Show(vrLStrMensaje, "ERROR", MessageBoxButtons.OK,
0159                   MessageBoxIcon.Warning);
0160             pRKpTecla.Handled = true;
0161          }
0162       }
0163    }
```

Fuente: elaboración propia.

- fPbBolValidarEmail: valida que la sintaxis de la cuenta de correo indicada, sea correcta, utilizando una expresión regular.

Figura 82. **Definición de la función fPbBolValidarEmail, de libHerramientas.1.0**

```
0169  //Función que valida si la cuenta de correo electrónico es válida
0170  public bool fPbBolValidarEmail(string pVStrEmail) {
0171      return Regex.IsMatch(pVStrEmail,
0172          @"^(?("")(""".+?""@)|(([0-9a-zA-Z]((\.(?!\.))|[-!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-zA-Z])@))"
0173          +
0174          @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-zA-Z][-\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,6}))$");
0175  }
```

Fuente: elaboración propia.

- fPbSqlConectar: es la función que se invoca para conectar a la base de datos. El usuario usrModSeguridad está asociado al inicio de sesión lginModSeguridad, el cual tiene asignado el rol rolModSeguridad. Este rol puede ejecutar todos los procedimientos almacenados de la base de datos.

Figura 83. **Definición de la función fPbSqlConectar, de libHerramientas.1.0**

```
0177  //Función que devuelve la conexión a la BDD
0178  public SqlConnection fPbSqlConectar() {
0179      System.Data.SqlClient.SqlConnection vrLSqlConexion =
0180          new System.Data.SqlClient.SqlConnection();
0181
0182      vrLSqlConexion.ConnectionString = "Data source=nchonay\\sqlexpress; " +
0183          "Initial Catalog=bdSeguridad; " +
0184          "User id=lginModSeguridad; Password=@ñonuev0;";
0185      vrLSqlConexion.Open();
0186      return vrLSqlConexion;
0187  }
```

Fuente: elaboración propia.

- fPbStrCifrar: por seguridad se cifra la contraseña de cada usuario, y para validar si la clave ingresada en la aplicación coincide con la registrada en el sistema, estos valores se comparan cifrados; además la aplicación no cuenta con el procedimiento para descifrar.

Figura 84. **Definición de la función fPbStrCifrar, de libHerramientas.1.0**

```
0189  //Función para cifrar
0190  public string fPbStrCifrar(string pVStrTexto) {
0191      byte[] vrLBtLlave; //Para almacenar la llave
0192      //Para almacenar el texto a cifrar
0193      byte[] vrLBtTexto = UTF8Encoding.UTF8.GetBytes(pVStrTexto);
0194      //Para calcular el valor hash MD5 de la llave
0195      MD5CryptoServiceProvider vrLMD5Hash = new MD5CryptoServiceProvider();
0196      //Para cifrar con el algoritmo triple DES
0197      TripleDESCryptoServiceProvider vrL3DESAlgoritmo =
0198            new TripleDESCryptoServiceProvider();
0199
0200      //Almacenar la llave con el valor hash
0201      vrLBtLlave =
0202          vrLMD5Hash.ComputeHash(UTF8Encoding.UTF8.GetBytes(atPrStrLlave));
0203      vrLMD5Hash.Clear();
0204
0205      //Características para aplicar el cifrado
0206      vrL3DESAlgoritmo.Key = vrLBtLlave;
0207      vrL3DESAlgoritmo.Mode = CipherMode.ECB;
0208      vrL3DESAlgoritmo.Padding = PaddingMode.PKCS7;
0209
0210      //Cifrar
0211      //Para cifrar
0212      ICryptoTransform vrLCptoCifrador = vrL3DESAlgoritmo.CreateEncryptor();
0213      byte[] vrLBtTextoCifrado =
0214          vrLCptoCifrador.TransformFinalBlock(vrLBtTexto, 0, vrLBtTexto.Length);
0215      vrL3DESAlgoritmo.Clear();
0216
0217      return Convert.ToBase64String(vrLBtTextoCifrado, 0, vrLBtTextoCifrado.Length);
0218  }
```

Fuente: elaboración propia.

- fPbTblBuscarUsuario: este procedimiento es utilizado para llamar al procedimiento almacenado pcBuscarUsuario, y captura en una tabla la información enviada desde la base de datos.

Figura 85. **Definición de la función fPbTblBuscarUsuario, de libHerramientas.1.0 (parte 1 de 2)**

```
0220  //Función para validar si el usuario y la contraseña son correctos,
0221  //si el usuario y el email son correctos,
0222  //buscar datos de todos los usuarios, buscar un usuario por ID,
0223  //o bien validar si el usuario ya existe
0224  public DataTable fPbTblBuscarUsuario(int pVIntId, string pVStrUsuario,
0225      string pVStrClave, string pVStrEmail, Int16 pVsIntOpcion,
0226      ref SqlConnection pRSQLConexion, ref Int16 pRsIntResultado) {
0227      //Valor de retorno
0228      DataTable rtTblDatos = new DataTable();
0229      //Inicializar el código del mensaje resultado
0230      pRsIntResultado = 10;
0231
0232      try {
0233          //Validar si se pudo establecer la conexión con SQL
0234          if (pRSQLConexion != null) {
0235              SqlCommand vrLSQLComando = new SqlCommand();
0236              SqlDataAdapter vrLSQLAdaptador;
0237
0238              //Indicar el nombre del procedimiento almacenado
0239              vrLSQLComando.CommandType = CommandType.StoredProcedure;
0240              vrLSQLComando.CommandText = "pcBuscarUsuario";
0241
0242              //Pasar los parámetros
0243              vrLSQLComando.Parameters.Clear();
0244              vrLSQLComando.Parameters.AddWithValue("@pVIntId", pVIntId);
0245              vrLSQLComando.Parameters.AddWithValue("@pVVchUsuario",
0246                      pVStrUsuario);
0247              vrLSQLComando.Parameters.AddWithValue("@pVVchClave", pVStrClave);
0248              vrLSQLComando.Parameters.AddWithValue("@pVVchEmail", pVStrEmail);
0249              vrLSQLComando.Parameters.AddWithValue("@pVsIntOpcion",
0250                      pVsIntOpcion);
0251              SqlParameter vrLSQLRetorno = vrLSQLComando.Parameters.Add(
0252                      new SqlParameter ("@pVsIntRetorno", SqlDbType.SmallInt));
0253              vrLSQLRetorno.Direction = ParameterDirection.ReturnValue;
0254
0255              //Ejecutar la consulta
0256              vrLSQLComando.Connection = pRSQLConexion;
0257              vrLSQLAdaptador = new SqlDataAdapter(vrLSQLComando);
0258              vrLSQLAdaptador.Fill(rtTblDatos);
0259              pRsIntResultado = Convert.ToInt16(
0260                      vrLSQLComando.Parameters["@pVsIntRetorno"].Value);
0261          }
0262          else {
0263              MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0264                  "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0265          }
0266      }
```

Fuente: elaboración propia.

Figura 86. **Definición de la función fPbTblBuscarUsuario, de libHerramientas.1.0 (parte 2 de 2)**

```
0267      catch (Exception pVExExcepcion) {
0268          pRsIntResultado = 10;
0269          MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0270              MessageBoxIcon.Error);
0271      }
0272      return rtTblDatos;
0273  }
```

Fuente: elaboración propia.

- pcPbModificarUsuario: para modificar los datos de los usuarios registrados en la base de datos se invoca a este procedimiento, que a su vez invoca al procedimiento almacenado pcModificarUsuario.

Figura 87. **Definición del procedimiento pcPbModificarUsuario, de libHerramientas.1.0**

```
0275  //Para actualizar los datos del usuario
0276  public void pcPbModificarUsuario(Int32 pVIntId, string pVStrUsuario,
0277      string pVStrClave, string pVStrNombres, string pVStrApellidos, string pVStrEmail,
0278      string pVStrRespuesta1, string pVStrRespuesta2, string pVStrRespuesta3,
0279      string pVStrRespuesta4, string pVStrRespuesta5, Int16 pVsIntOpcion,
0280      ref SqlConnection pRSQLConexion, ref Int16 pRsIntResultado) {
0281      //Inicializar el código del mensaje resultado
0282      pRsIntResultado = 10;
0283
0284      try {
0285          //Validar si se pudo establecer la conexión con SQL
0286          if (pRSQLConexion != null) {
0287              SqlCommand vrLSQLComando = new SqlCommand();
0288
0289              //Indicar el nombre del procedimiento almacenado
0290              vrLSQLComando.CommandType = CommandType.StoredProcedure;
0291              vrLSQLComando.CommandText = "pcModificarUsuario";
0292
0293              //Pasar los parámetros
0294              vrLSQLComando.Parameters.Clear();
0295              vrLSQLComando.Parameters.AddWithValue("@pVIntId", pVIntId);
0296              vrLSQLComando.Parameters.AddWithValue("@pVVchUsuario", pVStrUsuario);
0297              vrLSQLComando.Parameters.AddWithValue("@pVVchClave", pVStrClave);
```

Continuación de la figura 87.

```
0298            vrLSQLComando.Parameters.AddWithValue("@pVVchNombres",
0299                pVStrNombres);
0300            vrLSQLComando.Parameters.AddWithValue("@pVVchApellidos",
0301                pVStrApellidos);
0302            vrLSQLComando.Parameters.AddWithValue("@pVVchEmail", pVStrEmail);
0303            vrLSQLComando.Parameters.AddWithValue("@pVVchRespuesta1",
0304                pVStrRespuesta1);
0305            vrLSQLComando.Parameters.AddWithValue("@pVVchRespuesta2",
0306                pVStrRespuesta2);
0307            vrLSQLComando.Parameters.AddWithValue("@pVVchRespuesta3",
0308                pVStrRespuesta3);
0309            vrLSQLComando.Parameters.AddWithValue("@pVVchRespuesta4",
0310                pVStrRespuesta4);
0311            vrLSQLComando.Parameters.AddWithValue("@pVVchRespuesta5",
0312                pVStrRespuesta5);
0313            vrLSQLComando.Parameters.AddWithValue("@pVsIntOpcion", pVsIntOpcion);
0314            SqlParameter vrLSQLRetorno = vrLSQLComando.Parameters.Add(
0315                new SqlParameter("@pVsIntRetorno", SqlDbType.SmallInt));
0316            vrLSQLRetorno.Direction = ParameterDirection.ReturnValue;
0317
0318            //Ejecutar la consulta
0319            vrLSQLComando.Connection = pRSQLConexion;
0320            vrLSQLComando.ExecuteNonQuery();
0321            pRsIntResultado = Convert.ToInt16(
0322                vrLSQLComando.Parameters["@pVsIntRetorno"].Value);
0323        }
0324        else {
0325            MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0326                "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0327        }
0328    }
0329    catch (Exception pVExExcepcion) {
0330        pRsIntResultado = 10;
0331        MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0332            MessageBoxIcon.Error);
0333    }
0334 }
```

Fuente: elaboración propia.

- pcPbGuardarUsuario: este procedimiento de la librería se utiliza para llamar a pcGuardarUsuario.

Figura 88. **Procedimiento pcPbGuardarUsuario, de libHerramientas.1.0**

```
0336  //Para crear un nuevo usuario
0337  public void pcPbGuardarUsuario(ref int pRIntId, string pVStrUsuario,
0338      string pVStrClave, string pVStrNombres, string pVStrApellidos, string pVStrEmail,
0339      ref SqlConnection pRSQLConexion, ref Int16 pRsIntResultado) {
0340      //Inicializar el código del mensaje resultado
0341      pRsIntResultado = 10;
0342
0343      try {
0344          //Validar si se pudo establecer la conexión con SQL
0345          if (pRSQLConexion != null) {
0346              SqlCommand vrLSQLComando = new SqlCommand();
0347
0348              //Indicar el nombre del procedimiento almacenado
0349              vrLSQLComando.CommandType = CommandType.StoredProcedure;
0350              vrLSQLComando.CommandText = "pcGuardarUsuario";
0351
0352              //Pasar los parámetros
0353              vrLSQLComando.Parameters.Clear();
0354              vrLSQLComando.Parameters.AddWithValue("@pVVchUsuario", pVStrUsuario);
0355              vrLSQLComando.Parameters.AddWithValue("@pVVchClave", pVStrClave);
0356              vrLSQLComando.Parameters.AddWithValue("@pVVchNombres",
0357                  pVStrNombres);
0358              vrLSQLComando.Parameters.AddWithValue("@pVVchApellidos",
0359                  pVStrApellidos);
0360              vrLSQLComando.Parameters.AddWithValue("@pVVchEmail", pVStrEmail);
0361              SqlParameter vrLSQLRetorno = vrLSQLComando.Parameters.Add(
0362                  new SqlParameter("@pVsIntRetorno", SqlDbType.SmallInt));
0363              vrLSQLRetorno.Direction = ParameterDirection.ReturnValue;
0364              SqlParameter vrLSQLId = vrLSQLComando.Parameters.Add(
0365                  new SqlParameter("@pRIntId", SqlDbType.Int));
0366              vrLSQLId.Direction = ParameterDirection.Output;
0367
0368              //Ejecutar la consulta
0369              vrLSQLComando.Connection = pRSQLConexion;
0370              vrLSQLComando.ExecuteNonQuery();
0371              pRsIntResultado = Convert.ToInt16(
0372                  vrLSQLComando.Parameters["@pVsIntRetorno"].Value);
0373              pRIntId = Convert.ToInt32(vrLSQLComando.Parameters["@pRIntId"].Value);
0374          }
0375          else {
0376              MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0377                  "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0378          }
0379      }
0380      catch (Exception pVExExcepcion) {
0381          pRsIntResultado = 10;
0382          MessageBox.Show(pVExExcepcion.ToString(), "ERROR",
0383              MessageBoxButtons.OK, MessageBoxIcon.Error);
0384      }
0385  }
```

Fuente: elaboración propia.

- pcPbEliminarUsuario: se elimina un usuario por medio de esta subrutina, invocando a pcEliminarUsuario.

Figura 89. **Definición del procedimiento pcPbEliminarUsuario, de libHerramientas.1.0**

```
0387  //Procedimiento para eliminar un usuario
0388  public void pcPbEliminarUsuario(int pVIntId, ref SqlConnection pRSQLConexion,
0389      ref Int16 pRsIntResultado) {
0390      //Inicializar el código del mensaje resultado
0391      pRsIntResultado = 10;
0392
0393      try {
0394          //Validar si se pudo establecer la conexión con SQL
0395          if (pRSQLConexion != null) {
0396              SqlCommand vrLSQLComando = new SqlCommand();
0397
0398              //Indicar el nombre del procedimiento almacenado
0399              vrLSQLComando.CommandType = CommandType.StoredProcedure;
0400              vrLSQLComando.CommandText = "pcEliminarUsuario";
0401
0402              //Pasar los parámetros
0403              vrLSQLComando.Parameters.Clear();
0404              vrLSQLComando.Parameters.AddWithValue("@pVIntId", pVIntId);
0405              SqlParameter vrLSQLRetorno = vrLSQLComando.Parameters.Add(
0406                      new SqlParameter("@pVsIntRetorno", SqlDbType.SmallInt));
0407              vrLSQLRetorno.Direction = ParameterDirection.ReturnValue;
0408
0409              //Ejecutar la consulta
0410              vrLSQLComando.Connection = pRSQLConexion;
0411              vrLSQLComando.ExecuteNonQuery();
0412              pRsIntResultado = Convert.ToInt16(
0413                      vrLSQLComando.Parameters["@pVsIntRetorno"].Value);
0414          }
0415          else {
0416              MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0417                      "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0418          }
0419      }
0420      catch (Exception pVExExcepcion) {
0421          pRsIntResultado = 10;
0422          MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0423                  MessageBoxIcon.Error);
0424      }
0425  }
```

Fuente: elaboración propia.

- fPbTblBuscarPreguntas: para ejecutar pcBuscarPreguntas se invoca a este procedimiento, y así poder mostrarle al usuario el catálogo de preguntas secretas disponibles.

Figura 90. **Definición de la función fPbTblBuscarPreguntas, de libHerramientas.1.0**

```
0427  //Función para buscar las preguntas
0428  public DataTable fPbTblBuscarPreguntas(ref SqlConnection pRSQLConexion,
0429        ref Int16 pRsIntResultado) {
0430     DataTable rtTblDatos = new DataTable(); //Valor de retorno
0431     //Inicializar el código del mensaje resultado
0432     pRsIntResultado = 10;
0433     try {
0434        //Validar si se pudo establecer la conexión con SQL
0435        if (pRSQLConexion != null) {
0436           SqlCommand vrLSQLComando = new SqlCommand();
0437           SqlDataAdapter vrLSQLAdaptador;
0438           //Indicar el nombre del procedimiento almacenado
0439           vrLSQLComando.CommandType = CommandType.StoredProcedure;
0440           vrLSQLComando.CommandText = "pcBuscarPreguntas";
0441           //Pasar los parámetros
0442           vrLSQLComando.Parameters.Clear();
0443           SqlParameter vrLSQLRetorno = vrLSQLComando.Parameters.Add(
0444                 new SqlParameter("@pVsIntRetorno", SqlDbType.SmallInt));
0445           vrLSQLRetorno.Direction = ParameterDirection.ReturnValue;
0446           //Ejecutar la consulta
0447           vrLSQLComando.Connection = pRSQLConexion;
0448           vrLSQLAdaptador = new SqlDataAdapter(vrLSQLComando);
0449           vrLSQLAdaptador.Fill(rtTblDatos);
0450           pRsIntResultado = Convert.ToInt16(
0451                 vrLSQLComando.Parameters["@pVsIntRetorno"].Value);
0452        }
0453        else {
0454           MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0455                 "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0456        }
0457     }
0458     catch (Exception pVExExcepcion) {
0459        pRsIntResultado = 10;
0460        MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0461              MessageBoxIcon.Error);
0462     }
0463     return rtTblDatos;
0464  }
```

Fuente: elaboración propia.

- fPbBolEnviarEmail: cuando se crea un nuevo usuario, se recupera o se reinicia la contraseña, se envían las credenciales de acceso vía correo electrónico a la cuenta registrada del usuario en el sistema, utilizando como remitente la cuenta notificacion.sistema.gt@gmail.com.

Figura 91. **Definición de la función fPbBolEnviarEmail, de libHerramientas.1.0 (parte 1 de 2)**

```
0471  //Enviar correo electrónico
0472  public bool fPbBolEnviarEmail(string pVStrEmail, string pVStrUsuario,
0473      string pVStrClave, Int16 pVsIntOpcion) {
0474      //Configuración del correo
0475      MailAddress vrLAdrsRemitente =
0476          new MailAddress("notificacion.sistema.gt@gmail.com");
0477      MailAddress vrLAdrsDestinatario = new MailAddress(pVStrEmail);
0478      MailMessage vrLMsgMensaje =
0479          new MailMessage(vrLAdrsRemitente, vrLAdrsDestinatario);
0480      vrLMsgMensaje.Subject = "SISTEMA.GT";
0481      vrLMsgMensaje.IsBodyHtml = true;
0482
0483      switch (pVsIntOpcion) {
0484          //Mensaje de bienvenida
0485          case 1:
0486              vrLMsgMensaje.Body = "<b>BIENVENIDO</b> al SISTEMA.GT <br><br>" +
0487                  "Su usuario es: " + pVStrUsuario + "<br>" +
0488                  "La contraseña es: " + pVStrClave + "<br><br><br>" +
0489                  "Debe ingresar al sistema para elegir sus preguntas secretas, " +
0490                  "que servirán para recuperar la contraseña al momento de olvidarla.";
0491              break;
0492          //Mensaje cambio de contraseña
0493          case 2:
0494              vrLMsgMensaje.Body = "Sus credenciales para ingresar al SISTEMA.GT " +
0495                  "han cambiado. <br><br>" + "Su usuario es: <b>" + pVStrUsuario +
0496                  "</b><br>" + "La contraseña es: <b>" + pVStrClave + "</b>";
0497              break;
0498          default:
0499              vrLMsgMensaje.Body = "";
0500              break;
0501      }
0502
0503      SmtpClient vrLSmtpCorreoSaliente = new SmtpClient("smtp.gmail.com");
0504      vrLSmtpCorreoSaliente.Port = 587;
0505      vrLSmtpCorreoSaliente.EnableSsl = true;
0506      vrLSmtpCorreoSaliente.UseDefaultCredentials = false;
```

Fuente: elaboración propia.

Figura 92. **Definición de la función fPbBolEnviarEmail, de libHerramientas.1.0 (parte 2 de 2)**

```
0507        vrLSmtpCorreoSaliente.Credentials =
0508              new NetworkCredential("notificacion.sistema.gt@gmail.com", "notific@c1on");
0509
0510        try {
0511           //Enviar el mensaje
0512           vrLSmtpCorreoSaliente.Send(vrLMsgMensaje);
0513        }
0514        catch (SmtpFailedRecipientException pVSmtpExExcepcion) {
0515           MessageBox.Show(pVSmtpExExcepcion.Message, "ERROR",
0516                 MessageBoxButtons.OK, MessageBoxIcon.Error);
0517           return false;
0518        }
0519        catch (Exception pVExExcepcion) {
0520           MessageBox.Show(pVExExcepcion.Message, "ERROR",
0521                 MessageBoxButtons.OK, MessageBoxIcon.Error);
0522           return false;
0523        }
0524        return true;
0525     }
```

Fuente: elaboración propia.

- fPbVGenerarClave: para generar una clave aleatoria se utiliza esta función, que devuelve un arreglo de 2 posiciones, en una devuelve la clave no cifrada y en la otra posición la contraseña cifrada. Se convierte en tipo byte tanto la hora en que se genera la nueva clave, como el nombre del usuario, y luego se le extraen 5 dígitos al resultado de sumar 3 bytes del usuario y 6 de la hora.

Figura 93. **Definición de la función fPbVGenerarClave, de libHerramientas.1.0 (parte 1 de 2)**

```
0527  //Generar clave aleatoria
0528  public string[] fPbVGenerarClave(string pVStrUsuario) {
0529      string[] vrLvClave;
0530      string vrLStrTemp = "";
0531      vrLvClave = new string[2];
0532      vrLvClave[0] = "";
0533      vrLvClave[1] = "";
0534
0535      //Generar la clave
0536      //Tomando la hora en formato completo
0537      byte[] vrLBtClave = UTF8Encoding.UTF8.GetBytes(DateTime.Now.ToString("T"));
0538      foreach (byte vrLBtByte in vrLBtClave) {
0539          vrLvClave[0] += vrLBtByte.ToString();
0540      }
```

Fuente: elaboración propia.

Figura 94. **Definición de la función fPbVGenerarClave, de libHerramientas.1.0 (parte 2 de 2)**

```
0541      //Tomando el nombre del usuario
0542      vrLBtClave = UTF8Encoding.UTF8.GetBytes(pVStrUsuario + pVStrUsuario +
0543              pVStrUsuario);
0544      foreach (byte vrLBtByte in vrLBtClave) {
0545          vrLStrTemp += vrLBtByte.ToString();
0546      }
0547      //Tomando 5 dígitos de la suma de 3 bytes del usuario y 6 bytes de la hora
0548      vrLvClave[0] = Convert.ToString(Convert.ToInt64(vrLvClave[0].Substring(8, 13)) +
0549          Convert.ToInt32(vrLStrTemp.Substring(0, 2))).Substring(1, 6);
0550      //Cifrar la clave
0551      vrLvClave[1] = fPbStrCifrar(vrLvClave[0]);
0552
0553      return vrLvClave;
0554  }
```

Fuente: elaboración propia.

En la siguiente figura se muestra el atributo de la clase, que almacena la clave con la que se cifra en la función fPbStrCifrar.

Figura 95. **Atributo de la clase modHerramientas**

```
0029  //Llave para cifrar
0030  private string atPrStrLlave = "DESPUÉS DE LA TORMENTA, SIEMPRE LLEGA LA C@1MA";
```

Fuente: elaboración propia.

Todos los formularios tienen un ícono de ayuda, que al presionar abre una ventana que provee ayuda en línea. Para ello se tiene una única forma Ayuda, que muestra determinado texto con base al valor de sus atributos privados (figuras 98 y 99), los cuales son fijados por cada formulario que la invoca:

- atPrStrNomPadre: nombre del formulario que invoca a frmAyuda.
- atPrStrNomOpcion: indica la acción que se está realizando: reiniciando contraseñas, eliminando, agregando nuevos usuarios o editando.
- atPrStrNomModalidad: puede tener los valores ADMIN o vacío, para diferenciar si el que inició sesión fue el usuario administrador.

Figura 96. **Atributos privados de la clase frmAyuda**

```
0001        //Atributos privados
0002        private string atPrStrNomPadre;
0003        private string atPrStrNomOpcion;
0004        private string atPrStrNomModalidad;
0005        //Métodos para acceder a los atributos privados
0006        public string mdPbStrNomPadre {
0007            get { return atPrStrNomPadre; }
0008            set { atPrStrNomPadre = value; }
0009        }
0010        public string mdPbStrNomOpcion {
0011            get { return atPrStrNomOpcion; }
0012            set { atPrStrNomOpcion = value; }
0013        }
0014        public string mdPbStrNomModalidad {
0015            get { return atPrStrNomModalidad; }
0016            set { atPrStrNomModalidad = value; }
0017        }
```

Fuente: elaboración propia.

Como frmAyuda es un elemento de tipo Cuadro Acerca de, que es una plantilla incluida en Visual Studio, a continuación se muestra el código añadido al archivo Ayuda.cs:

Figura 97. **Procedimiento pcPrLoad de Ayuda.cs**

```
0001  //Procedimiento Load
0002  private void pcPrLoad(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0003     //Dependiendo de la página que lo invoque se muestra la ayuda
0004     switch (this.mdPbStrNomPadre) {
0005        case "frmInicio":
0006           this.Text = "Ayuda Login";
0007           txtBxDescripcion.Text = "Formulario de ingreso: \r\n" +
0008              "--> Ingrese su usuario y contraseña.  No deben contener más de 10 carácteres; " +
0009                    "solamente se aceptan números, letras y los carácteres - _ \r\n\r\n" +
0010              "--> En la contraseña se hace distinción entre mayúsculas y minúsculas. \r\n\r\n" +
0011              "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del botón.";
0012           break;
0013        case "frmDatosUsuario":
0014           this.Text = "Ayuda Datos Usuario";
0015           switch (this.mdPbStrNomOpcion) {
0016              case "AGREGANDO ...":
0017                 txtBxDescripcion.Text = "Agregar usuarios: \r\n" +
0018                    "--> El usuario no debe contener más de 10 carácteres; solamente se aceptan " +
0019                          "números, letras y los carácteres - _ \r\n\r\n" +
0020                    "--> Para los nombres y apellidos, solamente se aceptan letras y espacios en " +
0021                          "blanco. \r\n\r\n" +
0022                    "--> Para el E-mail, se aceptan números, letras y los carácteres @ . _ \r\n\r\n" +
0023                    "--> Las credenciales de acceso se envían al correo indicado.\r\n\r\n" +
0024                    "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del " +
0025                          "botón.";
0026                 break;
0027              case "EDITANDO ...":
0028                 txtBxDescripcion.Text = "Formulario de datos: \r\n" +
0029                    "--> El usuario y contraseña, no deben contener más de 10 carácteres; " +
0030                          "solamente se aceptan números, letras y los carácteres - _ \r\n\r\n" +
0031                    "--> En la contraseña se hace distinción entre mayúsculas y minúsculas. \r\n\r\n" +
0032                    "--> Para los nombres y apellidos, solamente se aceptan letras y espacios en " +
0033                          "blanco. \r\n\r\n" +
0034                    "--> Para el E-mail, se aceptan números, letras y los carácteres @ . _ \r\n\r\n" +
0035                    "--> Debe seleccionar al menos una pregunta secreta. \r\n\r\n" +
0036                    "--> Cada respuesta no debe contener más de 100 carácteres; solamente se " +
0037                          "aceptan números, letras, espacios en blanco y los carácteres - _ \r\n\r\n" +
0038                    "--> Si no desea modificar la contraseña, deje en blanco los campos " +
0039                          "<<Contraseña anterior>>, <<Nueva contraseña>> y <<Confirmar " +
0040                          "contraseña>>. \r\n\r\n" +
0041                    "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del " +
0042                          "botón.";
0043                 break;
```

Continuación de la figura 97.

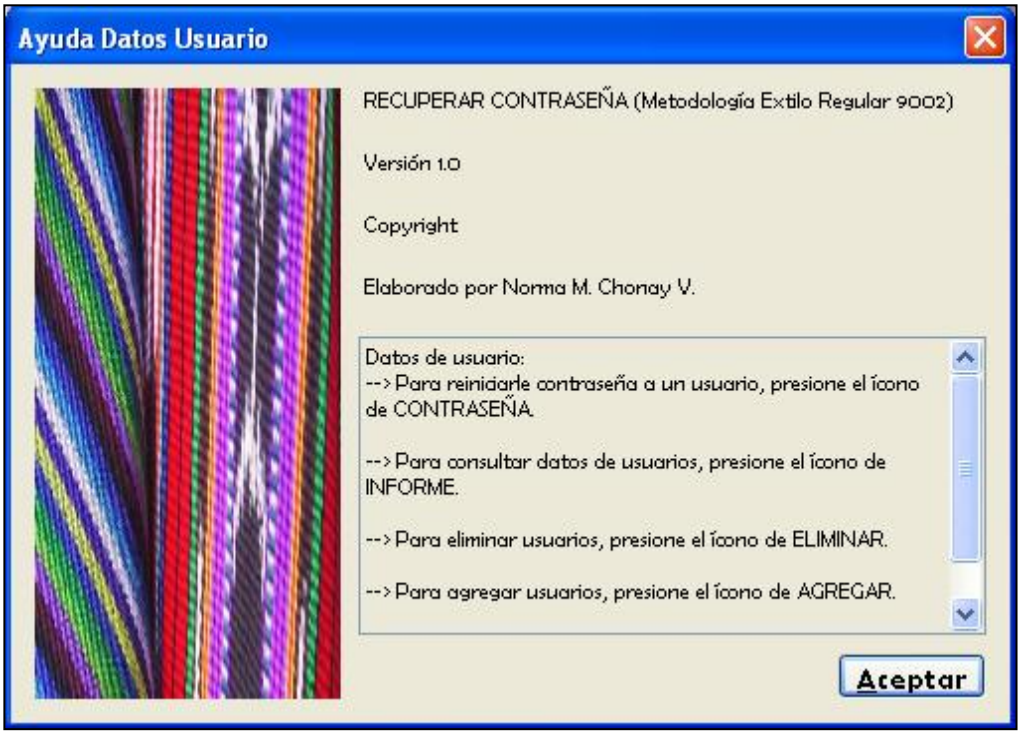
```
0044            case "ELIMINANDO ...":
0045                txtBxDescripcion.Text = "Eliminar usuarios: \r\n" +
0046                    "--> Primero debe ingresar el ID del usuario, luego presionar el botón BUSCAR, " +
0047                        "para visualizar los datos del usuario a eliminar. \r\n\r\n" +
0048                    "--> Después de verificar que el usuario es el que se desea eliminar, presionar " +
0049                        "el botón ACEPTAR; se pedirá una confirmación antes de eliminar el " +
0050                        "usuario permanentemente. \r\n\r\n" +
0051                    "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del " +
0052                        "botón.";
0053                break;
0054            case "REINICIANDO ...":
0055                txtBxDescripcion.Text = "Reinicio de contraseñas de usuarios: \r\n" +
0056                    "--> Primero debe ingresar el ID del usuario, luego presionar el botón BUSCAR, " +
0057                        "para visualizar los datos del usuario a quien se le reiniciará la contraseña. " +
0058                        "\r\n\r\n" +
0059                    "--> Después de verificar que el usuario es el que busca, presionar el botón " +
0060                        "ACEPTAR; se pedirá una confirmación antes de cambiar la contraseña. " +
0061                        "\r\n\r\n" +
0062                    "--> Las credenciales de acceso se envían al correo indicado.\r\n\r\n" +
0063                    "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del " +
0064                        "botón.";
0065                break;
0066            case "":
0067                txtBxDescripcion.Text = "Datos de usuario: \r\n";
0068                //Si es el usuario administrador
0069                if (mdPbStrNomModalidad == "ADMIN") {
0070                    txtBxDescripcion.Text = txtBxDescripcion.Text +
0071                      "--> Para reiniciarle contraseña a un usuario, presione el ícono de " +
0072                          "CONTRASEÑA. \r\n\r\n" +
0073                      "--> Para consultar datos de usuarios, presione el ícono de INFORME. \r\n\r\n" +
0074                      "--> Para eliminar usuarios, presione el ícono de ELIMINAR. \r\n\r\n" +
0075                      "--> Para agregar usuarios, presione el ícono de AGREGAR. \r\n\r\n";
0076                }
0077                txtBxDescripcion.Text = txtBxDescripcion.Text +
0078                      "--> Para modificar sus propios datos, presione el ícono de EDITAR.";
0079                break;
0080            default:
0081                txtBxDescripcion.Text = "";
0082                break;
0083        }
0084        break;
0085    case "frmRecuperacionClave":
0086        this.Text = "Ayuda Recuperación de Clave";
0087        txtBxDescripcion.Text = "Recuperación de clave: \r\n" +
0088            "--> El usuario y el E-mail deben coincidir con los registrados en el sistema.  La " +
0089                "nueva contraseña se enviará a la dirección de correo electrónica registrada. " +
0090                "\r\n\r\n" +
0091            "--> El usuario no debe contener más de 10 carácteres; solamente se aceptan " +
0092                "números, letras y los carácteres - _ \r\n\r\n" +
0093            "--> Para el E-mail, se aceptan números, letras y los carácteres @ . _ \r\n\r\n" +
0094            "--> Primero debe validar los datos, y luego se mostrará alguna de las preguntas " +
0095                "secretas que haya registrado, para que indique la respuesta. \r\n\r\n" +
```

Continuación de la figura 97.

```
0096                "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del botón.";
0097            break;
0098        default:
0099            break;
0100    }
0101 }
```

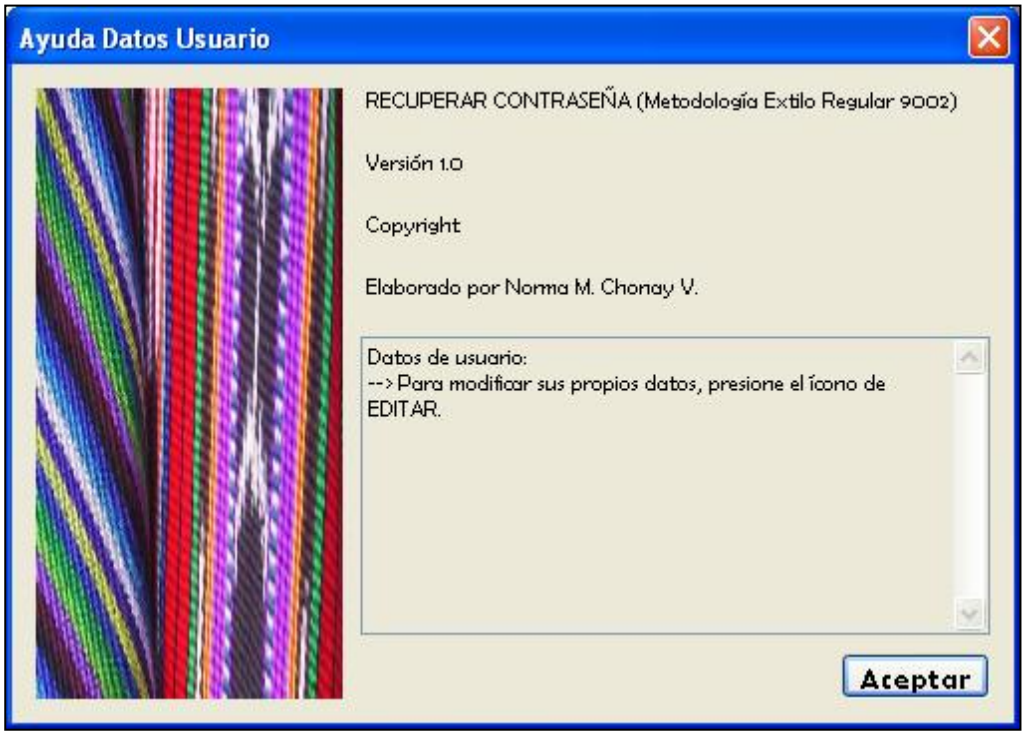
Fuente: elaboración propia.

Figura 98. **Ayuda en línea, cuando inicia sesión el usuario ADMIN**



Fuente: elaboración propia.

Figura 99. **Ayuda en línea, cuando inicia sesión cualquier usuario que no sea ADMIN**



Fuente: elaboración propia.

A continuación se describe cada uno de los formularios, iniciando con la ventana principal de la aplicación, que es un formulario en donde se solicita el ingreso de un usuario y contraseña, y se tiene opción a presionar 3 botones:

- ENTRAR: al presionarlo se valida que el usuario haya ingresado un nombre de usuario y una contraseña; con los datos ingresados, se busca si existe un usuario con ese nombre y esa clave, para permitir el ingreso al sistema.
- CANCELAR: quita los datos que se hubiesen escrito en las cajas de texto, y coloca el cursor en el control del usuario.

- RECUPERAR CLAVE: si el usuario no tiene su contraseña para ingresar, puede presionar este botón en donde se muestra otra ventana que solicita el ingreso del usuario, correo electrónico y la respuesta exacta de una de las preguntas secretas que configuró el usuario.

Figura 100. **Diseño de la forma Login**

Fuente: elaboración propia.

En toda la aplicación se validan los carácteres permitidos en cada control, por lo que en esta forma, para mostrar el mismo mensaje, y realizar cualquier cambio en el mismo de manera centralizada, se maneja una constante global implementada como atributo estático de la clase frmInicio.

Figura 101. **Constante global de la clase frmInicio**

```
0041        //Administra las constantes
0042        static class clsPbStatCteGlobal {
0043            //Mensaje que muestra el tooltip en los controles txtUsuario y txtClave
0044            public static string atPbStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE " +
0045                "LETRAS, NÚMEROS Y LOS CARÁCTERES - _";
0046
0047            //Devuelve el valor de atPbStrMensaje
0048            public static string mdPbStatStrDevolverMensaje {
0049                get { return atPbStrMensaje; }
0050            }
0051        }
```

Fuente: elaboración propia.

El código del formulario de inicio de sesión, es el siguiente:

Figura 102. **Código de Login.cs**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                          Login.cs                                                      */
0003  /*   Descripción:                      Forma para ingresar credenciales de acceso.                   */
0004  /*   Autor:                            Norma M. Chonay V.                                            */
0005  /*   Fecha de creación:                23/02/2011                                                    */
0006  /*   Fecha de última modificación:     23/02/2011                                                    */
0007  /*   Autor de última modificación:     Norma M. Chonay V.                                            */
0008  /*   Versión:                          1.1                                                           */
0009  /*   Prerrequisitos:                   Ninguno                                                       */
0010  /*****************************************************************************************************/
0011
0012  using System;
0013  using System.Collections.Generic;
0014  using System.ComponentModel;
0015  using System.Data;
0016  using System.Drawing;
0017  using System.Linq;
0018  using System.Text;
0019  using System.Windows.Forms;
0020  using RecuperarContraseña.Librerias;
0021  using RecuperarContraseña.modSeguridad;
0022
0023  namespace RecuperarContraseña {
0024      public partial class frmInicio : Form {
0025          public frmInicio() {
0026              InitializeComponent();
0027          }
0028
```

Continuación de la figura 102.

```
0029        //Instanciar la librería de utilidades
0030        modHerramientas modHerr = new modHerramientas();
0031
0032        //Procedimiento Load
0033        private void pcPrLoad(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0034            txtUsuario.Focus();
0035        }
0036
0037        /*************************************************************************************************/
0038        /********************** VALIDACIÓN DE DATOS EN LOS CONTROLES *****************/
0039        /*************************************************************************************************/
0040        //Valida los carácteres ingresados en el control txtUsuario
0041        private void pcPrValidarTxtUsuario(object pVObjTipoObjeto,
0042            KeyPressEventArgs pVKpTecla) {
0043            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtClave, 1);
0044        }
0045
0046        //Valida los carácteres ingresados en el control txtClave
0047        private void pcPrValidarTxtClave(object pVObjTipoObjeto,
0048            KeyPressEventArgs pVKpTecla) {
0049            modHerr.pcPbValidarCaracteres(ref pVKpTecla, btnEntrar, 1);
0050        }
0051
0052        /*************************************************************************************************/
0053        /***************************** EVENTOS DE LOS CONTROLES ***************************/
0054        /*************************************************************************************************/
0055        //Se validan las credenciales
0056        private void pcPrClicEntrar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0057            try {
0058                //Validar que se haya ingresado usuario y clave
0059                if (txtUsuario.Text != "" & txtClave.Text != "") {
0060                    //Conectarse a SQL
0061                    System.Data.SqlClient.SqlConnection vrLSqlConexion;
0062                    vrLSqlConexion = modHerr.fPbSqlConectar();
0063
0064                    //Validar si se pudo establecer la conexión con SQL
0065                    if (vrLSqlConexion != null) {
0066                        DataTable vrLTblDatos = new DataTable();
0067                        Int16 vrLsIntResultado = 0;
0068                        vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text,
0069                              modHerr.fPbStrCifrar(txtClave.Text),
0070                              "", 2, ref vrLSqlConexion, ref vrLsIntResultado);
0071                        switch (vrLsIntResultado) {
0072                            case 0:
0073                                MessageBox.Show("USUARIO Y CONTRASEÑA NO VÁLIDOS", "ERROR",
0074                                      MessageBoxButtons.OK, MessageBoxIcon.Warning);
0075                                txtUsuario.Focus();
0076                                break;
0077                            case 1:
0078                                //Validar si la consulta devolvió datos
0079                                if (vrLTblDatos.Rows.Count == 1) {
0080                                    txtUsuario.Text = "";
0081                                    txtClave.Text = "";
```

Continuación de la figura 102.

```
0082
0083                        //Pasar los datos del usuario a frmDatosUsuario
0084                        frmDatosUsuario vrLfrmDatosUsuario = new frmDatosUsuario();
0085                        vrLfrmDatosUsuario.mdPbIntId =
0086                              Convert.ToInt32(vrLTblDatos.Rows[0]["Id"]);
0087                        vrLfrmDatosUsuario.mdPbStrUsuario =
0088                              vrLTblDatos.Rows[0]["Usuario"].ToString();
0089                        vrLfrmDatosUsuario.mdPbStrClave =
0090                              vrLTblDatos.Rows[0]["Clave"].ToString();
0091                        vrLfrmDatosUsuario.mdPbStrNombres =
0092                           vrLTblDatos.Rows[0]["Nombres"].ToString();
0093                        vrLfrmDatosUsuario.mdPbStrApellidos =
0094                           vrLTblDatos.Rows[0]["Apellidos"].ToString();
0095                        vrLfrmDatosUsuario.mdPbStrEmail =
0096                              vrLTblDatos.Rows[0]["EMail"].ToString();
0097
0098                        vrLSqlConexion.Close();
0099                        vrLfrmDatosUsuario.ShowDialog(this);
0100                        txtUsuario.Focus();
0101                     }
0102                     else {
0103                        MessageBox.Show("NO HAY DATOS DEL USUARIO", "ERROR",
0104                              MessageBoxButtons.OK,
0105                              MessageBoxIcon.Error);
0106                        txtUsuario.Focus();
0107                     }
0108                     break;
0109                  case 2:
0110                     MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " +
0111                           "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0112                     txtUsuario.Focus();
0113                     break;
0114                  case 3:
0115                     MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS", "ERROR",
0116                           MessageBoxButtons.OK,
0117                           MessageBoxIcon.Error);
0118                     txtUsuario.Focus();
0119                     break;
0120                  default:
0121                     //Error mostrado en la librería
0122                     txtUsuario.Focus();
0123                     break;
0124               }
0125            }
0126            else {
0127               MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0128                     "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0129               txtUsuario.Focus();
0130            }
0131         }
0132         else {
0133            MessageBox.Show("DEBE INGRESAR USUARIO Y CONTRASEÑA", "ERROR",
0134                  MessageBoxButtons.OK,
```

Continuación de la figura 102.

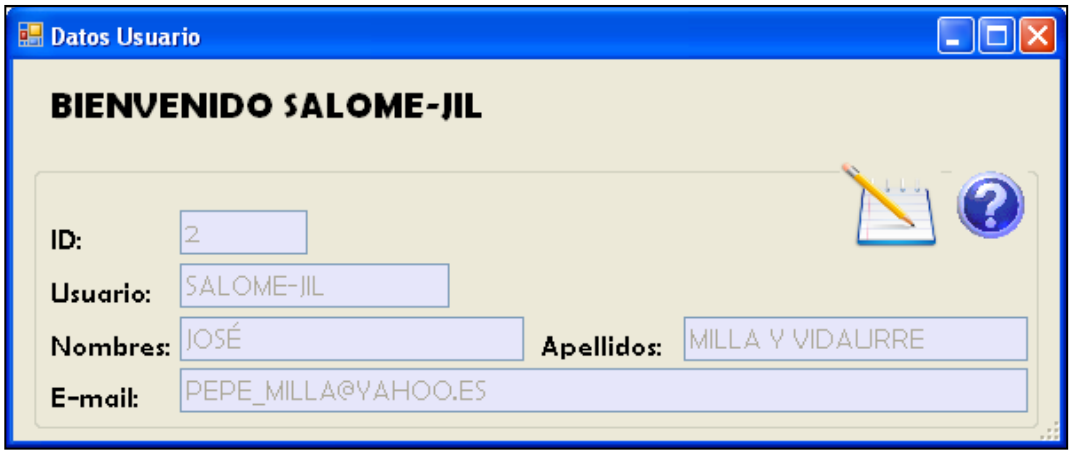
```
0135                        MessageBoxIcon.Warning);
0136                    txtUsuario.Focus();
0137                }
0138            }
0139            catch (Exception pVExExcepcion) {
0140                MessageBox.Show(pVExExcepcion.ToString(), "ERROR",
0141                    MessageBoxButtons.OK, MessageBoxIcon.Error);
0142                txtUsuario.Focus();
0143            }
0144        }
0145        //Si se presiona el botón CANCELAR, se limpia el texto ingresado en los controles
0146        private void pcPrClicCancelar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0147            txtUsuario.Text = "";
0148            txtClave.Text = "";
0149            txtUsuario.Focus();
0150        }
0151        //Al presionar el botón RECUPERAR CLAVE se abre frmRecuperacionClave
0152        private void pcPrClicRecuperarClave(object pVObjTipoObjeto,
0153            EventArgs pVEvtEvento) {
0154            //Llamar a la ventana para recuperar clave
0155            txtUsuario.Text = "";
0156            txtClave.Text = "";
0157            frmRecuperacionClave vrLfrmRecuperacionClave = new frmRecuperacionClave();
0158            vrLfrmRecuperacionClave.ShowDialog(this);
0159        }
0160        //Al dar clic en el ícono de AYUDA
0161        private void pcPrAyudaLogin(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0162            //Llamar a la ventana de ayuda en línea
0163            frmAyuda vrLfrmAyudaLogin = new frmAyuda();
0164            vrLfrmAyudaLogin.mdPbStrNomPadre = this.Name;
0165            vrLfrmAyudaLogin.mdPbStrNomOpcion = "";
0166            vrLfrmAyudaLogin.mdPbStrNomModalidad = "";
0167            vrLfrmAyudaLogin.ShowDialog(this);
0168        }
0169    }
0170 }
```

Fuente: elaboración propia.

Cuando un usuario inicia sesión se muestra el formulario DatosUsuario, el cual añade a la leyenda Bienvenido, el nombre del usuario que ingresó a la aplicación, y muestra los datos del usuario registrados en la base de datos. Si es el usuario ADMIN se muestran todos los íconos por medio de los cuales puede realizar cualquier acción: reiniciar contraseñas, generar el informe de usuarios, eliminar, agregar nuevos usuarios o editar sus propios datos. Si es

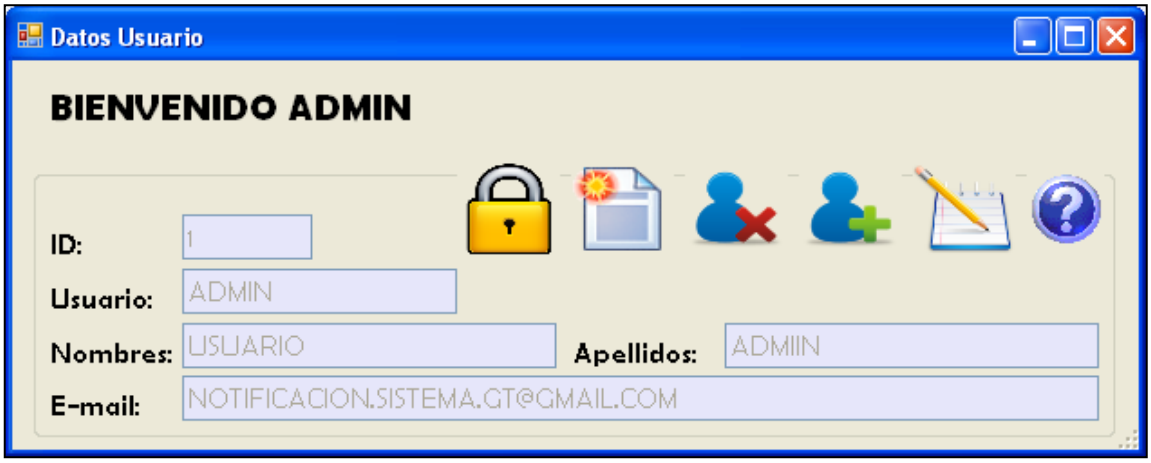
cualquier otro usuario, solamente podrá modificar sus propios datos: usuario, nombres, apellidos, dirección de correo electrónico, preguntas secretas y contraseña.

Figura 103. **Opciones de cualquier usuario que no es ADMIN**



Fuente: elaboración propia.

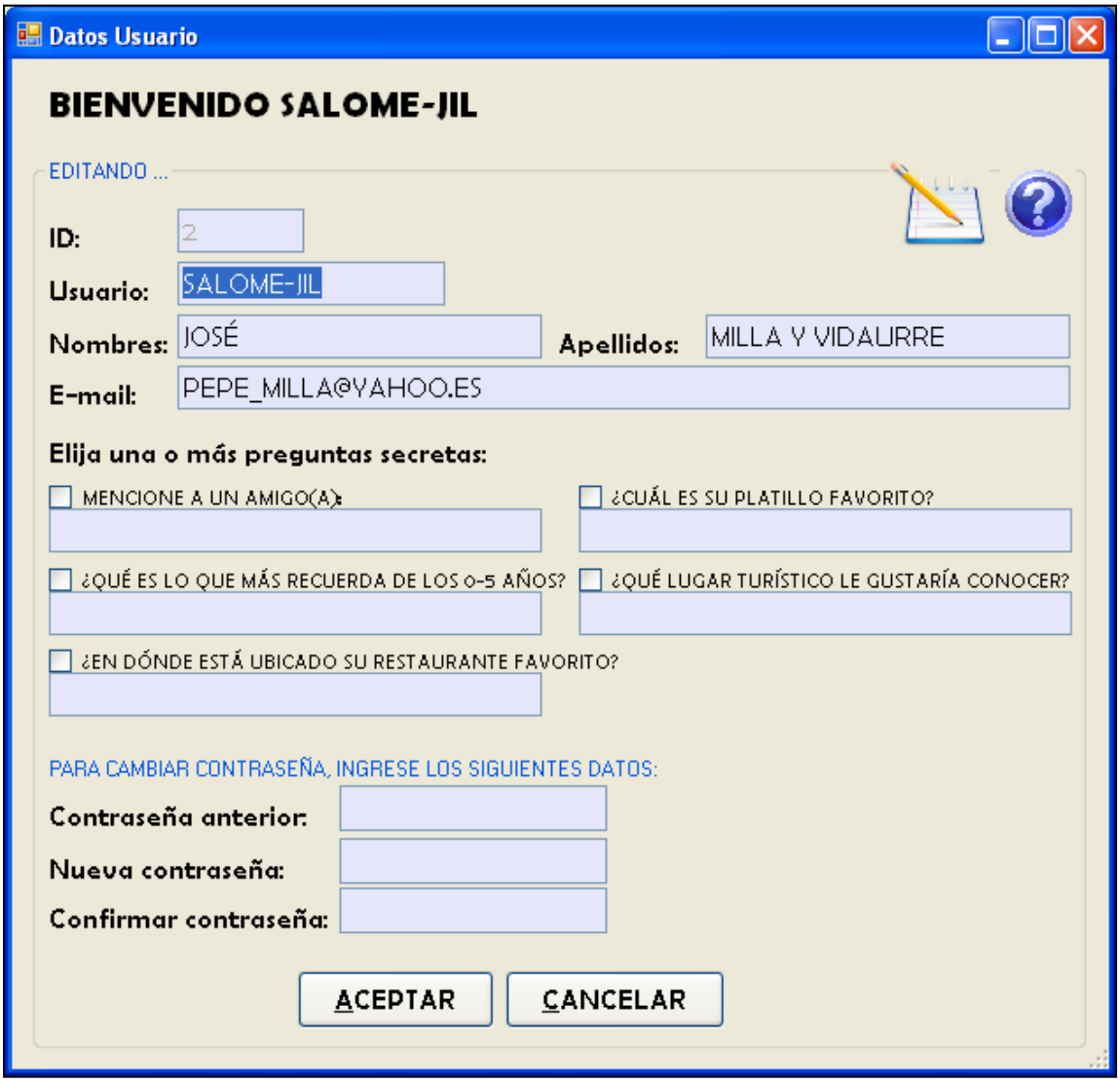
Figura 104. **Opciones del usuario ADMIN**



Fuente: elaboración propia.

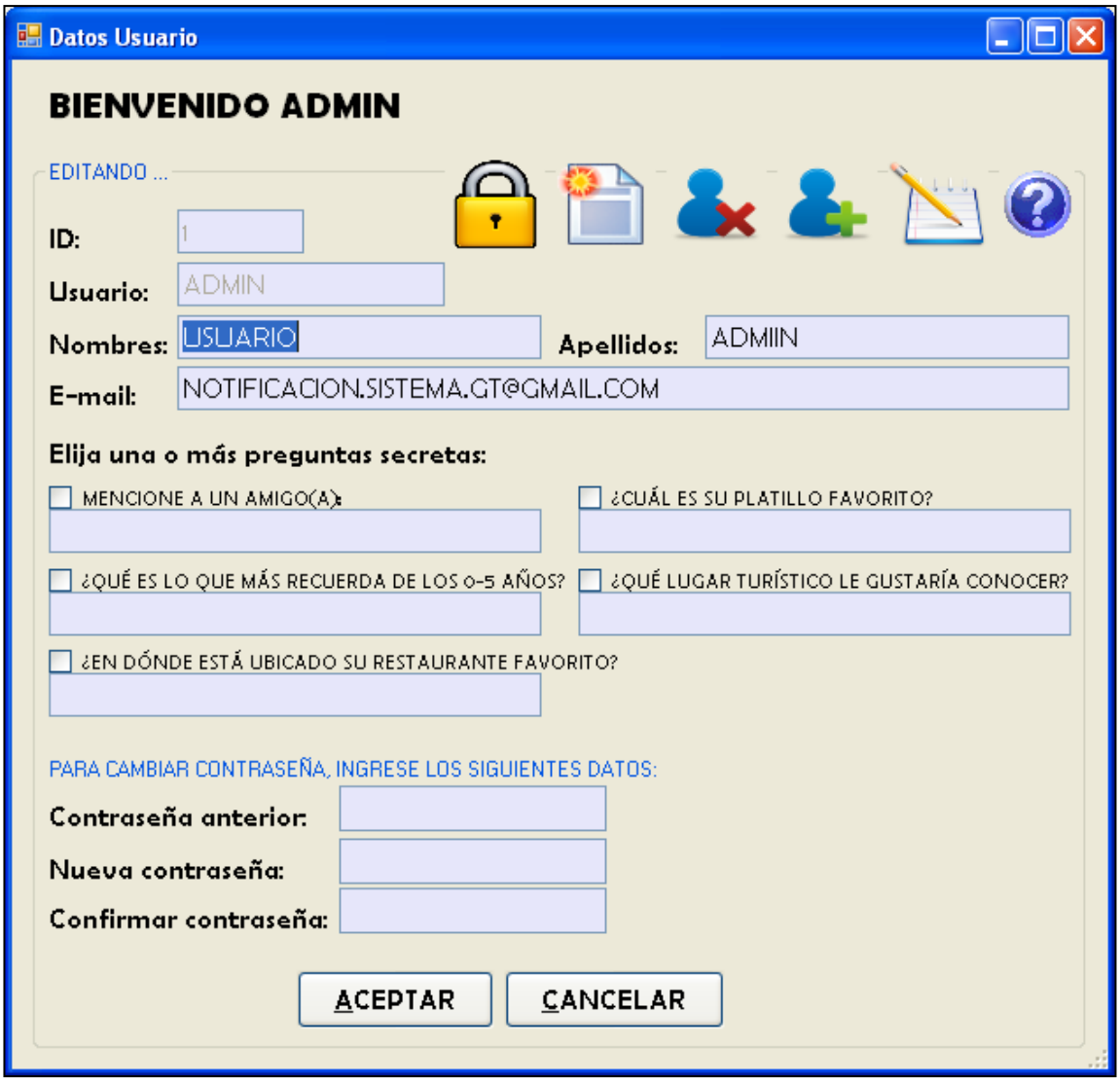
La diferencia entre la opción EDITAR del usuario ADMIN y de los demás usuarios, es que el usuario administrador no puede modificar su nombre de usuario, ya que dentro de toda la aplicación, con este valor es que se reconoce cuando el usuario que inició sesión es el administrador.

Figura 105. **Opción EDITAR de cualquier usuario que no es ADMIN**



Fuente: elaboración propia.

Figura 106. **Opción EDITAR del usuario ADMIN**



Fuente: elaboración propia.

Solamente el usuario ADMIN puede agregar usuarios al sistema, para ello basta con dar clic en el ícono de agregar, luego ingresar el nombre de usuario, nombres, apellidos y correo electrónico. Si se indica toda la información en el formulario y en la base de datos no existe un usuario con ese nombre, se

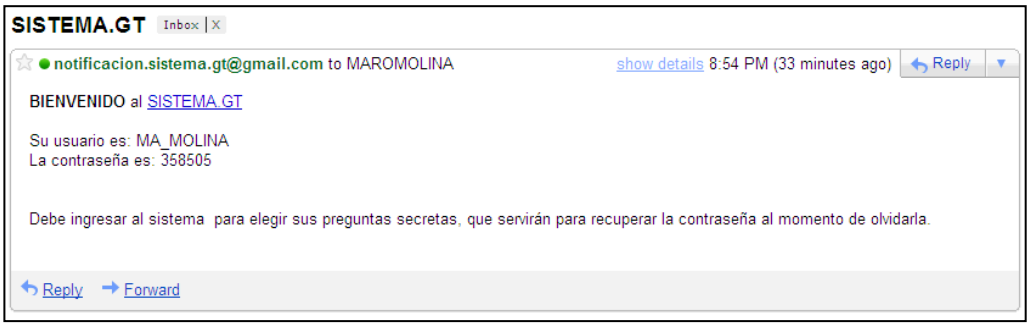
genera una contraseña, y se envían las credenciales de acceso al *email* indicado.

Figura 107. **Opción AGREGAR del usuario ADMIN**

Fuente: elaboración propia.

En el cuerpo del mensaje que se envía al nuevo usuario, se le indica que debe ingresar al sistema y configurar sus preguntas secretas, lo cual le servirá para recuperar la contraseña al momento de extraviarla. La siguiente figura muestra el correo electrónico que se envía desde la aplicación.

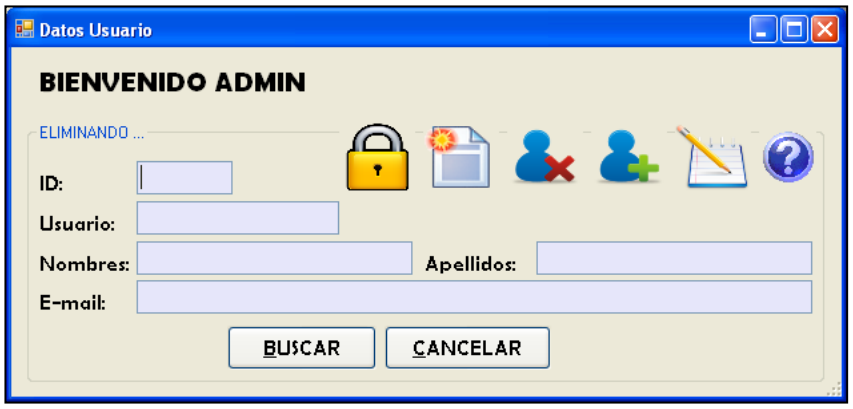
Figura 108. **Mensaje de bienvenida enviado por correo electrónico a cada nuevo usuario**



Fuente: elaboración propia.

Tanto para eliminar como para reiniciar la contraseña de cualquier usuario, se requiere ingresar el *Id* del usuario, para buscar la información con base a este valor, permitiendo así que el usuario ADMIN verifique los datos del usuario a eliminar o a reiniciarle la clave. Si se presiona el botón ACEPTAR, se solicita una confirmación para proceder a ejecutar la acción.

Figura 109. **Opción ELIMINAR del usuario ADMIN**



Fuente: elaboración propia.

Figura 110. **Opción REINICIAR CONTRASEÑA del usuario ADMIN**

Fuente: elaboración propia.

El mensaje enviado al usuario cuando se reinicia la contraseña desde la opción RECUPERAR CLAVE o desde REINICIAR CONTRASEÑA, es el indicado en la siguiente figura.

Figura 111. **Mensaje enviado por correo electrónico a cada usuario cuando cambia la contraseña**

Fuente: elaboración propia.

Al presionar el ícono de informe, se abre la siguiente ventana, mostrando los datos de todos los usuarios registrados en el sistema.

Figura 112. **Informe**

Fuente: elaboración propia.

El código fuente de este otro formulario (frmInforme), es el que se muestra a continuación:

Figura 113. **Código de repInforme.cs**

```
0001  /*************************************************************************************************/
0002  /*   Archivo:                      repInforme.cs                                                 */
0003  /*   Descripción:                  Forma para mostrar el reporte.                                */
0004  /*   Autor:                        Norma M. Chonay V.                                            */
0005  /*   Fecha de creación:            06/03/2011                                                    */
0006  /*   Fecha de última modificación: 06/03/2011                                                    */
0007  /*   Autor de última modificación: Norma M. Chonay V.                                            */
0008  /*   Versión:                      1.1                                                           */
0009  /*   Prerrequisitos:               Ninguno                                                       */
0010  /*************************************************************************************************/
0011
0012  using System;
```

Continuación de la figura 113.

```
0013  using System.Collections.Generic;
0014  using System.ComponentModel;
0015  using System.Data;
0016  using System.Drawing;
0017  using System.Linq;
0018  using System.Text;
0019  using System.Windows.Forms;
0020  using RecuperarContraseña.Librerias;
0021
0022  namespace RecuperarContraseña.modSeguridad {
0023      public partial class frmInforme : Form {
0024          public frmInforme() {
0025              InitializeComponent();
0026          }
0027
0028          //Instanciar la librería de utilidades
0029          modHerramientas modHerr = new modHerramientas();
0030
0031          //Procedimiento Load
0032          private void pcPrLoad(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0033              bool vrLBolCerrar = false; //indica si se cierra la ventana
0034              try {
0035                  //Mostrar la fecha
0036                  lblFecha.Text = DateTime.Now.ToString();
0037
0038                  //Conectarse a SQL
0039                  System.Data.SqlClient.SqlConnection vrLSqlConexion;
0040                  vrLSqlConexion = modHerr.fPbSqlConectar();
0041
0042                  //Validar si se pudo establecer la conexión con SQL
0043                  if (vrLSqlConexion != null) {
0044                      DataTable vrLTblDatos = new DataTable();
0045                      Int16 vrLsIntResultado = 0;
0046                      vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, "", "", "", 4, ref vrLSqlConexion,
0047                              ref vrLsIntResultado);
0048                      switch (vrLsIntResultado) {
0049                          case 0:
0050                              MessageBox.Show("NO HAY DATOS", "ERROR", MessageBoxButtons.OK,
0051                                      MessageBoxIcon.Warning);
0052                              vrLBolCerrar = true;
0053                              break;
0054                          case 1:
0055                              //Validar si la consulta devolvió datos
0056                              if (vrLTblDatos.Rows.Count >= 1) {
0057                                  //Mostrar los datos de los usuarios
0058                                  dGridVisor.DataSource = vrLTblDatos;
0059                              }
0060                              else {
0061                                  MessageBox.Show("NO HAY DATOS", "ERROR", MessageBoxButtons.OK,
0062                                          MessageBoxIcon.Error);
0063                                  vrLBolCerrar = true;
0064                              }
0065                              break;
```

Continuación de la figura 113.

```
0066                        case 2:
0067                            MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES VÁLIDA",
0068                                "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0069                            vrLBolCerrar = true;
0070                            break;
0071                        case 3:
0072                            MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS", "ERROR",
0073                                MessageBoxButtons.OK, MessageBoxIcon.Error);
0074                            vrLBolCerrar = true;
0075                            break;
0076                        default:
0077                            //Error mostrado en la librería
0078                            vrLBolCerrar = true;
0079                            break;
0080                    }
0081                }
0082                else {
0083                    MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE DE " +
0084                        "DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0085                    vrLBolCerrar = true;
0086                }
0087                vrLSqlConexion.Close();
0088            }
0089            catch (Exception pVExExcepcion) {
0090                MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0091                    MessageBoxIcon.Error);
0092                vrLBolCerrar = true;
0093            }
0094            if (vrLBolCerrar == true) {
0095                this.Dispose();
0096                this.Close();
0097            }
0098        }
0099    }
0100 }
```

Fuente: elaboración propia.

El formulario nombrado frmDatosUsuario, tiene el siguiente código fuente:

Figura 114. **Código de DatosUsuario.cs**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                      DatosUsuario.cs                                                   */
0003  /*   Descripción:                  Forma para consultar datos y modificarlos.                        */
0004  /*   Autor:                        Norma M. Chonay V.                                                */
0005  /*   Fecha de creación:            27/02/2011                                                        */
0006  /*   Fecha de última modificación: 27/02/2011                                                        */
0007  /*   Autor de última modificación: Norma M. Chonay V.                                                */
0008  /*   Versión:                      1.1                                                               */
0009  /*   Prerrequisitos:               Ninguno                                                           */
0010  /*****************************************************************************************************/
0011
0012  using System;
0013  using System.Collections.Generic;
0014  using System.ComponentModel;
0015  using System.Data;
0016  using System.Drawing;
0017  using System.Linq;
0018  using System.Text;
0019  using System.Windows.Forms;
0020  using RecuperarContraseña.Librerias;
0021
0022  namespace RecuperarContraseña.modSeguridad {
0023      public partial class frmDatosUsuario : Form {
0024          public frmDatosUsuario() {
0025              InitializeComponent();
0026          }
0027
0028          //Instanciar la librería de utilidades
0029          modHerramientas modHerr = new modHerramientas();
0030
0031          //Procedimiento Load
0032          private void pcPrLoad(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0033              grBxPrincipal.Size = new System.Drawing.Size(598, 151);
0034              this.Size = new System.Drawing.Size(631, 249);
0035              //Mostrar los datos del usuario
0036              pcPrMostrarDatos();
0037
0038              //Si es el usuario administrador
0039              if (mdPbStrUsuario == "ADMIN") {
0040                  picAgregar.Visible = true;
0041                  picQuitar.Visible = true;
0042                  picInforme.Visible = true;
0043                  picClave.Visible = true;
0044              }
0045
0046              //Colocar las preguntas en los controles
0047              pcPrObtenerPreguntas();
0048          }
0049
0050          /*****************************************************************************************************/
0051          /********************** VALIDACIÓN DE DATOS EN LOS CONTROLES *****************/
0052          /*****************************************************************************************************/
```

Continuación de la figura 114.

```
0053        //Valida los carácteres ingresados en el control txtId
0054        private void pcPrValidarTxtId(object pVObjTipoObjeto, KeyPressEventArgs pVKpTecla) {
0055            modHerr.pcPbValidarCaracteres(ref pVKpTecla, btnAceptar, 4);
0056        }
0057
0058        //Valida los carácteres ingresados en el control txtUsuario
0059        private void pcPrValidarTxtUsuario(object pVObjTipoObjeto,
0060            KeyPressEventArgs pVKpTecla) {
0061            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtNombres, 1);
0062        }
0063
0064        //Valida los carácteres ingresados en el control txtNombres
0065        private void pcPrValidarTxtNombres(object pVObjTipoObjeto,
0066                KeyPressEventArgs pVKpTecla) {
0067            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtApellidos, 2);
0068        }
0069
0070        //Valida los carácteres ingresados en el control txtApellidos
0071        private void pcPrValidarTxtApellidos(object pVObjTipoObjeto,
0072                KeyPressEventArgs pVKpTecla) {
0073            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtEmail, 2);
0074        }
0075
0076        //Valida los carácteres ingresados en el control txtEmail
0077        private void pcPrValidarTxtEmail(object pVObjTipoObjeto, KeyPressEventArgs pVKpTecla) {
0078            if (grBxPrincipal.Text == "EDITANDO ...") {
0079                modHerr.pcPbValidarCaracteres(ref pVKpTecla, chkPregunta1, 3);
0080            }
0081            else {
0082                modHerr.pcPbValidarCaracteres(ref pVKpTecla, btnAceptar, 3);
0083            }
0084        }
0085
0086        //En los checkboxs no se validan los carácteres ingresados, solamente si se presiona
0087        //la tecla ENTER
0088        private void pcPrValidarChkPregunta1(object pVObjTipoObjeto,
0089                KeyPressEventArgs pVKpTecla) {
0090            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtRespuesta1, 6);
0091        }
0092
0093        //En los checkboxs no se validan los carácteres ingresados, solamente si se presiona
0094        //la tecla ENTER
0095        private void pcPrValidarChkPregunta2(object pVObjTipoObjeto,
0096                KeyPressEventArgs pVKpTecla) {
0097            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtRespuesta2, 6);
0098        }
0099
0100        //En los checkboxs no se validan los carácteres ingresados, solamente si se presiona
0101        //la tecla ENTER
0102        private void pcPrValidarChkPregunta3(object pVObjTipoObjeto,
0103                KeyPressEventArgs pVKpTecla) {
0104            modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtRespuesta3, 6);
0105        }
```

Continuación de la figura 114.

```
0106
0107        //En los checkboxs no se validan los carácteres ingresados, solamente si se presiona
0108        //la tecla ENTER
0109        private void pcPrValidarChkPregunta4(object pVObjTipoObjeto,
0110              KeyPressEventArgs pVKpTecla) {
0111           modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtRespuesta4, 6);
0112        }
0113
0114        //En los checkboxs no se validan los carácteres ingresados, solamente si se presiona
0115        //la tecla ENTER
0116        private void pcPrValidarChkPregunta5(object pVObjTipoObjeto,
0117              KeyPressEventArgs pVKpTecla) {
0118           modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtRespuesta5, 6);
0119        }
0120
0121        //Valida los carácteres ingresados en el control txtRespuesta1
0122        private void pcPrValidarTxtRespuesta1(object pVObjTipoObjeto,
0123              KeyPressEventArgs pVKpTecla) {
0124           modHerr.pcPbValidarCaracteres(ref pVKpTecla, chkPregunta2, 5);
0125        }
0126
0127        //Valida los carácteres ingresados en el control txtRespuesta2
0128        private void pcPrValidarTxtRespuesta2(object pVObjTipoObjeto,
0129              KeyPressEventArgs pVKpTecla) {
0130           modHerr.pcPbValidarCaracteres(ref pVKpTecla, chkPregunta3, 5);
0131        }
0132
0133        //Valida los carácteres ingresados en el control txtRespuesta3
0134        private void pcPrValidarTxtRespuesta3(object pVObjTipoObjeto,
0135              KeyPressEventArgs pVKpTecla) {
0136           modHerr.pcPbValidarCaracteres(ref pVKpTecla, chkPregunta4, 5);
0137        }
0138
0139        //Valida los carácteres ingresados en el control txtRespuesta4
0140        private void pcPrValidarTxtRespuesta4(object pVObjTipoObjeto,
0141              KeyPressEventArgs pVKpTecla) {
0142           modHerr.pcPbValidarCaracteres(ref pVKpTecla, chkPregunta5, 5);
0143        }
0144
0145        //Valida los carácteres ingresados en el control txtRespuesta5
0146        private void pcPrValidarTxtRespuesta5(object pVObjTipoObjeto,
0147              KeyPressEventArgs pVKpTecla) {
0148           modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtClave, 5);
0149        }
0150
0151        //Valida los carácteres ingresados en el control txtClave
0152        private void pcPrValidarTxtClaveAnt(object pVObjTipoObjeto,
0153              KeyPressEventArgs pVKpTecla) {
0154           modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtClaveNueva, 1);
0155        }
0156
0157        //Valida los carácteres ingresados en el control txtClaveNueva
0158        private void pcPrValidarTxtClaveNueva(object pVObjTipoObjeto,
```

Continuación de la figura 114.

```
0159            KeyPressEventArgs pVKpTecla) {
0160        modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtConfClave, 1);
0161    }
0162
0163    //Valida los carácteres ingresados en el control txtConfClave
0164    private void pcPrValidarTxtConfClave(object pVObjTipoObjeto,
0165            KeyPressEventArgs pVKpTecla) {
0166        modHerr.pcPbValidarCaracteres(ref pVKpTecla, btnAceptar, 1);
0167    }
0168
0169    /*****************************************************************************************/
0170    /************************************ PROCEDIMIENTOS *****************************/
0171    /*****************************************************************************************/
0172    //Habilita o deshabilita los controles para editar los datos del usuario
0173    private void pcPrDeshabilitarControles(bool pVBolOpcion) {
0174        //Si es el usuario administrador
0175        if (mdPbStrUsuario == "ADMIN" & grBxPrincipal.Text == "EDITANDO ...") {
0176            txtUsuario.Enabled = false;
0177        }
0178        else {
0179            txtUsuario.Enabled = pVBolOpcion;
0180        }
0181        txtId.Enabled = false;
0182        txtNombres.Enabled = pVBolOpcion;
0183        txtApellidos.Enabled = pVBolOpcion;
0184        txtEmail.Enabled = pVBolOpcion;
0185    }
0186
0187    //Habilita o deshabilita los botones
0188    private void pcPrDeshabilitarBotones(bool pVBolOpcion, int pVIntPosY) {
0189        btnAceptar.Text = "&ACEPTAR";
0190        btnAceptar.Visible = pVBolOpcion;
0191        //Si el botón ACEPTAR está visible, habilitarlo
0192        if (pVBolOpcion == true)  {
0193            btnAceptar.Enabled = true;
0194        }
0195        btnCancelar.Visible = pVBolOpcion;
0196        pnlClave.Visible = pVBolOpcion;
0197        pnlPreguntas.Visible = pVBolOpcion;
0198        btnAceptar.Location = new Point(150, pVIntPosY);
0199        btnCancelar.Location = new Point(268, pVIntPosY);
0200    }
0201
0202    //Colocar los datos originales del usuario en los controles
0203    private void pcPrMostrarDatos() {
0204        //Mostrar los datos del usuario
0205        if (lblNomUsuario.Text.Contains(mdPbStrUsuario) == false) {
0206            lblNomUsuario.Text = lblNomUsuario.Text + mdPbStrUsuario;
0207        }
0208        txtId.Text = mdPbIntId.ToString();
0209        txtUsuario.Text = mdPbStrUsuario;
0210        txtNombres.Text = mdPbStrNombres;
0211        txtApellidos.Text = mdPbStrApellidos;
```

Continuación de la figura 114.

```
0212            txtEmail.Text = mdPbStrEmail;
0213            txtClave.Text = "";
0214            txtClaveNueva.Text = "";
0215            txtConfClave.Text = "";
0216
0217            //Deshabilitar los controles
0218            pcPrDeshabilitarControles(false);
0219            pcPrDeshabilitarBotones(false, 276);
0220        }
0221
0222        //Limpiar los controles, para ingresar nuevos usuarios
0223        private void pcPrInicializarDatos() {
0224            //Limpiar los datos de los controles
0225            txtId.Text = "";
0226            txtUsuario.Text = "";
0227            txtNombres.Text = "";
0228            txtApellidos.Text = "";
0229            txtEmail.Text = "";
0230            txtClave.Text = "";
0231            txtClaveNueva.Text = "";
0232            txtConfClave.Text = "";
0233        }
0234
0235        //Inicializar los datos de los controles
0236        private void pcPrCancelar(bool pVBolMostrarDatos) {
0237            grBxPrincipal.Text = "";
0238            grBxPrincipal.Size = new System.Drawing.Size(598, 151);
0239            this.Size = new System.Drawing.Size(631, 249);
0240            if (pVBolMostrarDatos == true) {
0241                //Mostrar los datos del usuario
0242                pcPrMostrarDatos();
0243            }
0244        }
0245
0246        //Se validan y almacenan los datos ingresados en la opción AGREGAR
0247        private void pcPrGuardar() {
0248            try {
0249                //Quitar espacios
0250                txtNombres.Text = txtNombres.Text.ToUpper().Trim();
0251                txtApellidos.Text = txtApellidos.Text.ToUpper().Trim();
0252                txtUsuario.Text = txtUsuario.Text.ToUpper().Trim();
0253                txtEmail.Text = txtEmail.Text.ToUpper().Trim();
0254
0255                //Validar que no estén vacíos
0256                if (txtUsuario.Text != "" &
0257                    txtNombres.Text != "" & txtApellidos.Text != "" &
0258                    txtEmail.Text != "") {
0259                    //Validar la dirección de correo electrónico
0260                    if (modHerr.fPbBolValidarEmail(txtEmail.Text) == true) {
0261                        //Generar la clave
0262                        Int16 vrLsIntResultado = 0;
0263                        string[] vrLvNuevaClave;
0264                        vrLvNuevaClave = modHerr.fPbVGenerarClave(txtUsuario.Text);
```

Continuación de la figura 114.

```
0265                    if (vrLvNuevaClave[0] == "" || vrLvNuevaClave[1] == "") {
0266                        MessageBox.Show("NO SE PUDO GENERAR LA CONTRASEÑA, " +
0267                            "EL USUARIO NO FUE CREADO", "ERROR", MessageBoxButtons.OK,
0268                            MessageBoxIcon.Error);
0269                        pcPrCancelar(true);
0270                    }
0271                    else {
0272                        //Conectarse a SQL
0273                        System.Data.SqlClient.SqlConnection vrLSqlConexion;
0274                        vrLSqlConexion = modHerr.fPbSqlConectar();
0275
0276                        //Verificar si el usuario ya existe
0277                        modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text, "", "", 1,
0278                            ref vrLSqlConexion, ref vrLsIntResultado);
0279                        switch (vrLsIntResultado) {
0280                            case 0:
0281                                int vrLIntId = -1;
0282                                vrLsIntResultado = 0;
0283
0284                                //Crear el usuario
0285                                modHerr.pcPbGuardarUsuario(ref vrLIntId,
0286                                    txtUsuario.Text,
0287                                    vrLvNuevaClave[1], txtNombres.Text, txtApellidos.Text,
0288                                    txtEmail.Text,
0289                                    ref vrLSqlConexion, ref vrLsIntResultado);
0290                                switch (vrLsIntResultado) {
0291                                    case 0:
0292                                        MessageBox.Show("EL USUARIO NO FUE CREADO, EL " +
0293                                            "NOMBRE DE USUARIO INDICADO YA EXISTE", "ERROR",
0294                                            MessageBoxButtons.OK, MessageBoxIcon.Error);
0295                                        break;
0296                                    case 1:
0297                                        //Mostrar el código asignado al usuario y deshabilitar los controles
0298                                        txtId.Text = vrLIntId.ToString();
0299                                        pcPrDeshabilitarControles(false);
0300                                        pcPrDeshabilitarBotones(true, 151);
0301                                        btnAceptar.Enabled = false;
0302                                        pnlClave.Visible = false;
0303                                        pnlPreguntas.Visible = false;
0304
0305                                        //Enviar el correo con la nueva contraseña
0306                                        if (modHerr.fPbBolEnviarEmail(txtEmail.Text,
0307                                            txtUsuario.Text,
0308                                            vrLvNuevaClave[0], 1) == true) {
0309                                            MessageBox.Show("USUARIO CREADO.  LA CONTRASEÑA " +
0310                                                "FUE ENVIADA A LA DIRECCIÓN DE CORREO INDICADA " +
0311                                                "PARA ESTE USUARIO.", "NOTIFICACIÓN",
0312                                                MessageBoxButtons.OK, MessageBoxIcon.Information);
0313                                        }
0314                                        else {
0315                                            MessageBox.Show("SE CREÓ EL USUARIO, PERO HUBO " +
0316                                                "INCONVENIENTE AL ENVIAR LAS CREDENCIALES A LA " +
0317                                                "CUENTA DE CORREO INDICADA", "ERROR",
```

Continuación de la figura 114.

```
0318                                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
0319                            }
0320                            break;
0321                        case 3:
0322                            MessageBox.Show("EL USUARIO NO FUE CREADO, HUBO " +
0323                                "UN ERROR EN LA BASE DE DATOS", "ERROR",
0324                                MessageBoxButtons.OK, MessageBoxIcon.Error);
0325                            break;
0326                        default:
0327                            //Error mostrado en la librería
0328                            break;
0329                    }
0330                    break;
0331                case 1:
0332                    MessageBox.Show("EL USUARIO INDICADO YA EXISTE", "ERROR",
0333                        MessageBoxButtons.OK, MessageBoxIcon.Error);
0334                    txtUsuario.Focus();
0335                    break;
0336                case 2:
0337                    MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " +
0338                        "VÁLIDA", "ERROR", MessageBoxButtons.OK,
0339                        MessageBoxIcon.Error);
0340                    txtUsuario.Focus();
0341                    break;
0342                case 3:
0343                    MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS",
0344                        "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0345                    txtUsuario.Focus();
0346                    break;
0347                default:
0348                    //Error mostrado en la librería
0349                    txtUsuario.Focus();
0350                    break;
0351            }
0352            vrLSqlConexion.Close();
0353        }
0354    }
0355    else {
0356        MessageBox.Show("LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " +
0357            "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0358        txtEmail.Focus();
0359    }
0360  }
0361  else {
0362    MessageBox.Show("TODOS LOS CAMPOS SON OBLIGATORIOS", "ERROR",
0363        MessageBoxButtons.OK, MessageBoxIcon.Warning);
0364    txtUsuario.Focus();
0365  }
0366 }
0367 catch (Exception pVExExcepcion) {
0368   MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0369       MessageBoxIcon.Error);
0370   txtUsuario.Focus();
```

Continuación de la figura 114.

```
0371            }
0372        }
0373
0374        //Se validan y actualizan los datos ingresados en la opción EDITAR
0375        private void pcPrModificar() {
0376            try {
0377                //Quitar espacios
0378                txtNombres.Text = txtNombres.Text.ToUpper().Trim();
0379                txtApellidos.Text = txtApellidos.Text.ToUpper().Trim();
0380                txtUsuario.Text = txtUsuario.Text.ToUpper().Trim();
0381                txtEmail.Text = txtEmail.Text.ToUpper().Trim();
0382                txtRespuesta1.Text = txtRespuesta1.Text.Trim();
0383                txtRespuesta2.Text = txtRespuesta2.Text.Trim();
0384                txtRespuesta3.Text = txtRespuesta3.Text.Trim();
0385                txtRespuesta4.Text = txtRespuesta4.Text.Trim();
0386                txtRespuesta5.Text = txtRespuesta5.Text.Trim();
0387
0388                //Validar que no estén vacíos
0389                if (txtId.Text != "" & txtUsuario.Text != "" &
0390                    txtNombres.Text != "" & txtApellidos.Text != "" &
0391                    txtEmail.Text != "") {
0392                    //Validar la dirección de correo electrónico
0393                    if (modHerr.fPbBolValidarEmail(txtEmail.Text) == true) {
0394                        //Validar las contraseñas
0395                        bool vrLBolSeguir = true;
0396                        //Si más de algún campo tiene datos
0397                        if (txtClave.Text != "" || txtClaveNueva.Text != "" || txtConfClave.Text != "") {
0398                            //Validar si la clave del control txtClave es la misma que la que está registrada
0399                            if (mdPbStrClave == modHerr.fPbStrCifrar(txtClave.Text)) {
0400                                //Validar que la nueva clave y la confirmación tengan datos
0401                                if (txtClaveNueva.Text != "" & txtConfClave.Text != "") {
0402                                    //Validar que la nueva clave, sea la misma en los 2 controles
0403                                    if (txtClaveNueva.Text != txtConfClave.Text) {
0404                                        MessageBox.Show("LA NUEVA CONTRASEÑA NO COINCIDE " +
0405                                            "CON LA CONFIRMACIÓN DE LA MISMA", "ERROR",
0406                                            MessageBoxButtons.OK, MessageBoxIcon.Warning);
0407                                        txtClaveNueva.Focus();
0408                                        vrLBolSeguir = false;
0409                                    }
0410                                }
0411                                else {
0412                                    MessageBox.Show("DEBE INGRESAR LA NUEVA CONTRASEÑA Y " +
0413                                        "LA CONFIRMACIÓN DE LA MISMA", "ERROR",
0414                                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
0415                                    txtClaveNueva.Focus();
0416                                    vrLBolSeguir = false;
0417                                }
0418                            }
0419                            else {
0420                                MessageBox.Show("LA <<CONTRASEÑA ANTERIOR>> INGRESADA " +
0421                                    "EN EL FORMULARIO NO COINCIDE CON LA ALMACENADA EN " +
0422                                    "EL SISTEMA ACTUALMENTE", "ERROR", MessageBoxButtons.OK,
0423                                    MessageBoxIcon.Warning);
```

Continuación de la figura 114.

```
0424                        txtClave.Focus();
0425                        vrLBolSeguir = false;
0426                    }
0427                }
0428
0429                //Si pasó las validaciones de las contraseñas
0430                if (vrLBolSeguir == true) {
0431                    vrLBolSeguir = fPrBolValidarPreguntas();
0432                    //Si pasó las validaciones de las preguntas
0433                    if (vrLBolSeguir == true) {
0434                        Int16 vrLsIntResultado = 0;
0435                        string vrLStrClave = mdPbStrClave;
0436                        //Conectarse a SQL
0437                        System.Data.SqlClient.SqlConnection vrLSqlConexion;
0438                        vrLSqlConexion = modHerr.fPbSqlConectar();
0439
0440                        //Verificar si el usuario ya existe
0441                        modHerr.fPbTblBuscarUsuario(Convert.ToInt32(txtId.Text), txtUsuario.Text,
0442                            "", "", 1, ref vrLSqlConexion, ref vrLsIntResultado);
0443                        switch (vrLsIntResultado) {
0444                            case 0:
0445                                //Hacer las modificaciones en la BDD
0446                                if (txtClaveNueva.Text != "") {
0447                                    vrLStrClave = modHerr.fPbStrCifrar(txtClaveNueva.Text);
0448                                }
0449
0450                                modHerr.pcPbModificarUsuario(Convert.ToInt32(txtId.Text),
0451                                    txtUsuario.Text, vrLStrClave, txtNombres.Text, txtApellidos.Text,
0452                                    txtEmail.Text, txtRespuesta1.Text, txtRespuesta2.Text,
0453                                    txtRespuesta3.Text, txtRespuesta4.Text, txtRespuesta5.Text, 1,
0454                                    ref vrLSqlConexion, ref vrLsIntResultado);
0455                                switch (vrLsIntResultado) {
0456                                    case 0:
0457                                        MessageBox.Show("NO SE REALIZÓ LA ACTUALIZACIÓN DE " +
0458                                            "DATOS, EL USUARIO NO EXISTE", "ERROR",
0459                                            MessageBoxButtons.OK, MessageBoxIcon.Error);
0460                                        txtUsuario.Focus();
0461                                        break;
0462                                    case 1:
0463                                        MessageBox.Show("DATOS ACTUALIZADOS.", "NOTIFICACIÓN",
0464                                            MessageBoxButtons.OK, MessageBoxIcon.Information);
0465                                        break;
0466                                    case 3:
0467                                        MessageBox.Show("NO SE REALIZÓ LA ACTUALIZACIÓN DE " +
0468                                            "DATOS, HUBO UN ERROR EN LA BASE DE DATOS",
0469                                            "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0470                                        txtUsuario.Focus();
0471                                        break;
0472                                    default:
0473                                        //Error mostrado en la librería
0474                                        txtUsuario.Focus();
0475                                        break;
0476                                }
```

Continuación de la figura 114.

```
0477
0478                            vrLSqlConexion.Close();
0479                            this.Dispose();
0480                            this.Close();
0481                            break;
0482                        case 1:
0483                            MessageBox.Show("EL USUARIO INDICADO YA EXISTE", "ERROR",
0484                                MessageBoxButtons.OK, MessageBoxIcon.Error);
0485                            txtUsuario.Focus();
0486                            break;
0487                        case 2:
0488                            MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO " +
0489                                "ES VÁLIDA", "ERROR", MessageBoxButtons.OK,
0490                                MessageBoxIcon.Error);
0491                            txtUsuario.Focus();
0492                            break;
0493                        case 3:
0494                            MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS",
0495                                "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0496                            txtUsuario.Focus();
0497                            break;
0498                        default:
0499                            //Error mostrado en la librería
0500                            txtUsuario.Focus();
0501                            break;
0502                    }
0503                }
0504            }
0505        }
0506        else {
0507            MessageBox.Show("LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " +
0508                "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0509            txtEmail.Focus();
0510        }
0511    }
0512    else {
0513        MessageBox.Show("TODOS LOS CAMPOS SON OBLIGATORIOS", "ERROR",
0514            MessageBoxButtons.OK, MessageBoxIcon.Warning);
0515        txtUsuario.Focus();
0516    }
0517  }
0518  catch (Exception pVExExcepcion) {
0519    MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0520        MessageBoxIcon.Error);
0521    txtUsuario.Focus();
0522  }
0523 }
0524
0525 //Se validan los datos y se actualiza la contraseña en la opción REINICIAR
0526 private void pcPrReiniciarClave() {
0527    try {
0528       //Validar que no estén vacíos
0529       if (txtUsuario.Text != "" & txtEmail.Text != "") {
```

Continuación de la figura 114.

```
0530                //Validar la dirección de correo electrónico
0531                if (modHerr.fPbBolValidarEmail(txtEmail.Text) == true) {
0532                    DataTable vrLTblDatos = new DataTable();
0533                    Int16 vrLsIntResultado = 0;
0534                    string[] vrLvNuevaClave;
0535
0536                    //Conectarse a SQL
0537                    System.Data.SqlClient.SqlConnection vrLSqlConexion;
0538                    vrLSqlConexion = modHerr.fPbSqlConectar();
0539
0540                    //Consultar en la BDD si el usuario y el email coinciden
0541                    vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text.ToUpper().Trim(),
0542                        "", txtEmail.Text.ToUpper().Trim(), 0, ref vrLSqlConexion,
0543                        ref vrLsIntResultado);
0544                    switch (vrLsIntResultado) {
0545                        case 0:
0546                            MessageBox.Show("NO SE PUEDE RECUPERAR LA CLAVE, LA " +
0547                                "INFORMACIÓN INDICADA NO ESTÁ REGISTRADA EN EL SISTEMA",
0548                                "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0549                            break;
0550                        case 1:
0551                            //Validar si la consulta devolvió datos
0552                            if (vrLTblDatos.Rows.Count == 1) {
0553                                //Obtener la nueva clave
0554                                vrLvNuevaClave = modHerr.fPbVGenerarClave(
0555                                    txtUsuario.Text.ToUpper().Trim());
0556                                if (vrLvNuevaClave[0] != "" & vrLvNuevaClave[1] != "") {
0557                                    //Modificar los datos en la BDD
0558                                    vrLsIntResultado = 0;
0559                                    modHerr.pcPbModificarUsuario(
0560                                        Convert.ToInt32(vrLTblDatos.Rows[0]["Id"]),
0561                                        vrLTblDatos.Rows[0]["Usuario"].ToString(),
0562                                        vrLvNuevaClave[1],
0563                                        vrLTblDatos.Rows[0]["Nombres"].ToString(),
0564                                        vrLTblDatos.Rows[0]["Apellidos"].ToString(),
0565                                        vrLTblDatos.Rows[0]["EMail"].ToString(),
0566                                        "", "", "", "", "", 0,
0567                                        ref vrLSqlConexion, ref vrLsIntResultado);
0568
0569                                    switch (vrLsIntResultado) {
0570                                        case 0:
0571                                            MessageBox.Show("NO SE REALIZÓ LA ACTUALIZACIÓN DE " +
0572                                                "LA CONTRASEÑA, EL USUARIO NO EXISTE", "ERROR",
0573                                                MessageBoxButtons.OK, MessageBoxIcon.Error);
0574                                            break;
0575                                        case 1:
0576                                            //Enviar el correo con la nueva contraseña
0577                                            if (modHerr.fPbBolEnviarEmail(
0578                                                vrLTblDatos.Rows[0]["EMail"].ToString(),
0579                                                vrLTblDatos.Rows[0]["Usuario"].ToString(),
0580                                                vrLvNuevaClave[0], 2) == true) {
0581                                                MessageBox.Show("LA NUEVA CONTRASEÑA FUE " +
0582                                                    "ENVIADA A LA DIRECCIÓN DE CORREO " +
```

Continuación de la figura 114.

```
0583                                    "REGISTRADA EN EL SISTEMA.", "NOTIFICACIÓN",
0584                                    MessageBoxButtons.OK, MessageBoxIcon.Information);
0585                                }
0586                                else {
0587                                    MessageBox.Show("SE ACTUALIZÓ LA CONTRASEÑA, " +
0588                                        "PERO HUBO INCONVENIENTE AL ENVIAR LAS " +
0589                                        "NUEVAS CREDENCIALES A LA CUENTA DE CORREO " +
0590                                        "REGISTRADA", "ERROR", MessageBoxButtons.OK,
0591                                        MessageBoxIcon.Error);
0592                                }
0593                                break;
0594                            case 3:
0595                                MessageBox.Show("NO SE REALIZÓ LA ACTUALIZACIÓN DE " +
0596                                    "LA CONTRASEÑA, HUBO UN ERROR EN LA BASE DE " +
0597                                    "DATOS", "ERROR", MessageBoxButtons.OK,
0598                                    MessageBoxIcon.Error);
0599                                break;
0600                            default:
0601                                //Error mostrado en la librería
0602                                break;
0603                        }
0604                    }
0605                    else {
0606                        MessageBox.Show("NO SE PUDO GENERAR UNA NUEVA " +
0607                            "CONTRASEÑA", "ERROR", MessageBoxButtons.OK,
0608                            MessageBoxIcon.Error);
0609                    }
0610                }
0611                else {
0612                    MessageBox.Show("LA INFORMACIÓN INDICADA, NO ESTÁ " +
0613                        "REGISTRADA EN EL SISTEMA", "ERROR",
0614                        MessageBoxButtons.OK, MessageBoxIcon.Error);
0615                }
0616                break;
0617            case 2:
0618                MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " +
0619                    "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0620                break;
0621            case 3:
0622                MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS", "ERROR",
0623                    MessageBoxButtons.OK, MessageBoxIcon.Error);
0624                break;
0625            default:
0626                //Error mostrado en la librería
0627                break;
0628        }
0629
0630        vrLSqlConexion.Close();
0631        this.Dispose();
0632        this.Close();
0633    }
0634    else {
0635        MessageBox.Show("LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " +
```

Continuación de la figura 114.

```
0636                            "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0637                        txtEmail.Focus();
0638                    }
0639                }
0640                else {
0641                    MessageBox.Show("TODOS LOS CAMPOS SON OBLIGATORIOS", "ERROR",
0642                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
0643                    txtUsuario.Focus();
0644                }
0645            }
0646            catch (Exception pVExExcepcion) {
0647                MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0648                    MessageBoxIcon.Error);
0649                txtUsuario.Focus();
0650            }
0651        }
0652
0653        //Se validan y actualizan los datos ingresados en la opción ELIMINAR
0654        private void pcPrEliminar() {
0655            try {
0656                //Validar que no estén vacíos
0657                if (txtId.Text != "" & txtUsuario.Text != "" &
0658                    txtNombres.Text != "" & txtApellidos.Text != "" &
0659                    txtEmail.Text != "") {
0660                    Int16 vrLsIntResultado = 0;
0661
0662                    //Conectarse a SQL
0663                    System.Data.SqlClient.SqlConnection vrLSqlConexion;
0664                    vrLSqlConexion = modHerr.fPbSqlConectar();
0665
0666                    modHerr.pcPbEliminarUsuario(Convert.ToInt32(txtId.Text), ref vrLSqlConexion,
0667                        ref vrLsIntResultado);
0668                    switch (vrLsIntResultado) {
0669                        case 0:
0670                            MessageBox.Show("NO SE ELIMINÓ EL USUARIO, EL USUARIO NO EXISTE",
0671                                "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0672                            txtUsuario.Focus();
0673                            break;
0674                        case 1:
0675                            MessageBox.Show("USUARIO ELIMINADO.", "NOTIFICACIÓN",
0676                                MessageBoxButtons.OK, MessageBoxIcon.Information);
0677                            break;
0678                        case 3:
0679                            MessageBox.Show("NO SE ELIMINÓ EL USUARIO, HUBO UN ERROR EN " +
0680                                "LA BASE DE DATOS", "ERROR", MessageBoxButtons.OK,
0681                                MessageBoxIcon.Error);
0682                            txtUsuario.Focus();
0683                            break;
0684                        default:
0685                            //Error mostrado en la librería
0686                            txtUsuario.Focus();
0687                            break;
0688                    }
```

Continuación de la figura 114.

```
0689
0690                vrLSqlConexion.Close();
0691                pcPrCancelar(true);
0692            }
0693            else {
0694                MessageBox.Show("NO SE HA SELECCIONADO EL USUARIO A ELIMINAR",
0695                    "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0696                btnCancelar.Focus();
0697            }
0698        }
0699        catch (Exception pVExExcepcion) {
0700            MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0701                MessageBoxIcon.Error);
0702            btnCancelar.Focus();
0703        }
0704    }
0705
0706    //Se busca al usuario por el Id y se muestran los datos en los controles
0707    private void pcPrBuscarUsuario() {
0708        try {
0709            //Validar que se haya ingresado el id
0710            if (txtId.Text != "") {
0711                //Validar que no sea el usuario ADMINISTRADOR
0712                if ((grBxPrincipal.Text == "REINICIANDO ...") ||
0713                    ((txtId.Text != "1") && (grBxPrincipal.Text == "ELIMINANDO ..."))) {
0714                    //Conectarse a SQL
0715                    System.Data.SqlClient.SqlConnection vrLSqlConexion;
0716                    vrLSqlConexion = modHerr.fPbSqlConectar();
0717
0718                    //Validar si se pudo establecer la conexión con SQL
0719                    if (vrLSqlConexion != null) {
0720                        DataTable vrLTblDatos = new DataTable();
0721                        Int16 vrLsIntResultado = 0;
0722                        vrLTblDatos = modHerr.fPbTblBuscarUsuario(Convert.ToInt32(txtId.Text), "", "",
0723                            "", 3, ref vrLSqlConexion, ref vrLsIntResultado);
0724                        switch (vrLsIntResultado) {
0725                            case 0:
0726                                MessageBox.Show("EL USUARIO NO EXISTE", "ERROR",
0727                                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
0728                                txtId.Focus();
0729                                break;
0730                            case 1:
0731                                //Validar si la consulta devolvió datos
0732                                if (vrLTblDatos.Rows.Count == 1) {
0733                                    //Mostrar los datos del usuario
0734                                    txtUsuario.Text = vrLTblDatos.Rows[0]["Usuario"].ToString();
0735                                    txtNombres.Text = vrLTblDatos.Rows[0]["Nombres"].ToString();
0736                                    txtApellidos.Text = vrLTblDatos.Rows[0]["Apellidos"].ToString();
0737                                    txtEmail.Text = vrLTblDatos.Rows[0]["EMail"].ToString();
0738
0739                                    txtUsuario.Enabled = false;
0740                                    btnAceptar.Text = "&ACEPTAR";
0741                                    btnAceptar.Focus();
```

Continuación de la figura 114.

```
0742                            }
0743                            else {
0744                               MessageBox.Show("NO HAY DATOS DEL USUARIO", "ERROR",
0745                                     MessageBoxButtons.OK, MessageBoxIcon.Error);
0746                               txtId.Focus();
0747                            }
0748                            break;
0749                         case 2:
0750                            MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " +
0751                                  "VÁLIDA", "ERROR", MessageBoxButtons.OK,
0752                                  MessageBoxIcon.Error);
0753                            txtId.Focus();
0754                            break;
0755                         case 3:
0756                            MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS",
0757                                  "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0758                            txtId.Focus();
0759                            break;
0760                         default:
0761                            //Error mostrado en la librería
0762                            txtId.Focus();
0763                            break;
0764                      }
0765                   }
0766                   else {
0767                      MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " +
0768                            "DE DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0769                      txtId.Focus();
0770                   }
0771                   vrLSqlConexion.Close();
0772                }
0773                else {
0774                   MessageBox.Show("EL USUARIO ADMINISTRADOR NO SE PUEDE ELIMINAR",
0775                         "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0776                   txtId.Text = "";
0777                   txtId.Focus();
0778                }
0779             }
0780             else {
0781                MessageBox.Show("DEBE INGRESAR EL CÓDIGO DE USUARIO", "ERROR",
0782                      MessageBoxButtons.OK, MessageBoxIcon.Warning);
0783                txtId.Focus();
0784             }
0785          }
0786          catch (Exception pVExExcepcion) {
0787             MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0788                   MessageBoxIcon.Error);
0789             txtId.Focus();
0790          }
0791       }
0792
0793       //Se buscan las preguntas y se pasan los datos a los controles
0794       private void pcPrObtenerPreguntas() {
```

Continuación de la figura 114.

```
0795            try {
0796               //Conectarse a SQL
0797               System.Data.SqlClient.SqlConnection vrLSqlConexion;
0798               vrLSqlConexion = modHerr.fPbSqlConectar();
0799
0800               //Validar si se pudo establecer la conexión con SQL
0801               if (vrLSqlConexion != null) {
0802                  DataTable vrLTblDatos = new DataTable();
0803                  Int16 vrLsIntResultado = 0;
0804                  vrLTblDatos = modHerr.fPbTblBuscarPreguntas(ref vrLSqlConexion,
0805                        ref vrLsIntResultado);
0806                  switch (vrLsIntResultado) {
0807                     case 0:
0808                        MessageBox.Show("NO HAY PREGUNTAS PARA MOSTRAR", "ERROR",
0809                              MessageBoxButtons.OK, MessageBoxIcon.Warning);
0810                        vrLSqlConexion.Close();
0811                        this.Dispose();
0812                        this.Close();
0813                        break;
0814                     case 1:
0815                        //Validar si la consulta devolvió datos
0816                        if (vrLTblDatos.Rows.Count == 5) {
0817                           //Pasar los datos a los controles
0818                           for (int i = 0; i <= 4; i++) {
0819                              Control ctrPregunta = null;
0820
0821                              //Apuntar al checkbox correspondiente
0822                              switch (i) {
0823                                 case 0:
0824                                    ctrPregunta = chkPregunta1;
0825                                    break;
0826                                 case 1:
0827                                    ctrPregunta = chkPregunta2;
0828                                    break;
0829                                 case 2:
0830                                    ctrPregunta = chkPregunta3;
0831                                    break;
0832                                 case 3:
0833                                    ctrPregunta = chkPregunta4;
0834                                    break;
0835                                 case 4:
0836                                    ctrPregunta = chkPregunta5;
0837                                    break;
0838                              }
0839
0840                              //Si el control apunta a un checkbox válido
0841                              if (ctrPregunta != null) {
0842                                 ctrPregunta.Text = vrLTblDatos.Rows[i]["PREGUNTA"].ToString();
0843                                 ctrPregunta = null;
0844                              }
0845                           }
0846                        }
0847                        else {
```

Continuación de la figura 114.

```
0848                        MessageBox.Show("HACEN FALTA PREGUNTAS PARA MOSTRAR",
0849                            "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0850                        vrLSqlConexion.Close();
0851                        this.Dispose();
0852                        this.Close();
0853                    }
0854                    break;
0855                case 3:
0856                    MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS", "ERROR",
0857                        MessageBoxButtons.OK, MessageBoxIcon.Error);
0858                    vrLSqlConexion.Close();
0859                    this.Dispose();
0860                    this.Close();
0861                    break;
0862                default:
0863                    //Error mostrado en la librería
0864                    vrLSqlConexion.Close();
0865                    this.Dispose();
0866                    this.Close();
0867                    break;
0868            }
0869        }
0870        else {
0871            MessageBox.Show("NO SE PUDO ESTABLECER CONEXIÓN A LA BASE DE " +
0872                "DATOS", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0873            txtId.Focus();
0874        }
0875        vrLSqlConexion.Close();
0876    }
0877    catch (Exception pVExExcepcion) {
0878        MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0879            MessageBoxIcon.Error);
0880        txtId.Focus();
0881    }
0882  }
0883
0884  //Valida que si la pregunta está seleccionada, tenga una respuesta
0885  private void pcPrValidarCadaPregunta(bool pVBolSeleccionado,
0886      Control pRCtrlRespuesta, ref Int16 pRsIntContador) {
0887      if (pVBolSeleccionado == true) {
0888          //Si la pregunta está seleccionada
0889          if (pRCtrlRespuesta.Text.Trim() == "") {
0890              pRsIntContador += 1;
0891          }
0892      }
0893  }
0894
0895  //Tanto en la opción REINICIAR CONTRASEÑA, como en la opción ELIMINAR,
0896  //primero se debe indicar el ID del usuario
0897  private void pcPrParaBuscarDatosUsuario(string pVStrNomOpcion,
0898      string pVStrMensaje) {
0899      //Si se está en alguna opción
0900      if (grBxPrincipal.Text != "") {
```

Continuación de la figura 114.

```
0901            pcPrCancelar(false);
0902        }
0903
0904        grBxPrincipal.Text = pVStrNomOpcion;
0905        grBxPrincipal.Size = new System.Drawing.Size(598, 193);
0906        this.Size = new System.Drawing.Size(631, 295);
0907
0908        //Habilitar los controles
0909        pcPrDeshabilitarControles(false);
0910        pcPrDeshabilitarBotones(true, 151);
0911        pnlClave.Visible = false;
0912        pnlPreguntas.Visible = false;
0913        txtId.Enabled = true;
0914        btnAceptar.Text = "&BUSCAR";
0915
0916        //Limpiar los controles
0917        pcPrInicializarDatos();
0918
0919        MessageBox.Show(pVStrMensaje + " Y LUEGO PRESIONE EL BOTÓN <<BUSCAR>>",
0920            "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
0921        txtId.Focus();
0922    }
0923
0924    /*************************************************************************************************/
0925    /************************************** FUNCIONES ************************************************/
0926    /*************************************************************************************************/
0927    //Se valida que el usuario haya seleccionado al menos una pregunta
0928    //e ingresado la respuesta
0929    private bool fPrBolValidarPreguntas() {
0930        bool vrLBolCorrecto = false;
0931        Int16 vrLsIntContador = 0;
0932        //Validar que mas de alguna pregunta esté seleccionada
0933        if (chkPregunta1.Checked == false & chkPregunta2.Checked == false
0934            & chkPregunta3.Checked == false & chkPregunta4.Checked == false
0935            & chkPregunta5.Checked == false) {
0936            MessageBox.Show("DEBE ELEGIR AL MENOS UNA PREGUNTA SECRETA E " +
0937                "INDICAR SU RESPUESTA", "ERROR", MessageBoxButtons.OK,
0938                MessageBoxIcon.Warning);
0939            chkPregunta1.Focus();
0940        }
0941        else {
0942            //Validar que cada pregunta tenga su respuesta
0943            pcPrValidarCadaPregunta(chkPregunta1.Checked, txtRespuesta1, ref vrLsIntContador);
0944            pcPrValidarCadaPregunta(chkPregunta2.Checked, txtRespuesta2, ref vrLsIntContador);
0945            pcPrValidarCadaPregunta(chkPregunta3.Checked, txtRespuesta3, ref vrLsIntContador);
0946            pcPrValidarCadaPregunta(chkPregunta4.Checked, txtRespuesta4, ref vrLsIntContador);
0947            pcPrValidarCadaPregunta(chkPregunta5.Checked, txtRespuesta5, ref vrLsIntContador);
0948
0949            if (vrLsIntContador == 0) {
0950                //Si no hubo error al validar las preguntas
0951                vrLBolCorrecto = true;
0952            }
0953            else {
```

Continuación de la figura 114.

```
0954                MessageBox.Show("PARA CADA PREGUNTA SELECCIONADA DEBE INDICAR " +
0955                    "UNA RESPUESTA", "ERROR", MessageBoxButtons.OK,
0956                    MessageBoxIcon.Warning);
0957                chkPregunta1.Focus();
0958            }
0959        }
0960        return vrLBolCorrecto;
0961    }
0962
0963    /*************************************************************************************************/
0964    /***************************** EVENTOS DE LOS CONTROLES ***************************/
0965    /*************************************************************************************************/
0966    //Se validan los datos ingresados
0967    private void pcPrClicAceptar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0968        switch (grBxPrincipal.Text) {
0969            //Editar
0970            case "EDITANDO ...":
0971                pcPrModificar();
0972                break;
0973            //Agregar
0974            case "AGREGANDO ...":
0975                pcPrGuardar();
0976                break;
0977            //Eliminar
0978            //Reiniciar
0979            case "ELIMINANDO ...":
0980            case "REINICIANDO ...":
0981                if (btnAceptar.Text == "&BUSCAR") {
0982                    pcPrBuscarUsuario();
0983                }
0984                else {
0985                    //Opción ELIMINAR
0986                    if (grBxPrincipal.Text == "ELIMINANDO ...") {
0987                        //Solicitar confirmación antes de eliminar el usuario
0988                        if (MessageBox.Show("¿ESTÁ SEGURO DE ELIMINAR EL USUARIO?",
0989                            "CONFIRMACIÓN", MessageBoxButtons.OKCancel,
0990                            MessageBoxIcon.Question) == DialogResult.Cancel) {
0991                            //Se cancela la opción
0992                            pcPrCancelar(true);
0993                        }
0994                        else {
0995                            //Se hacen las validaciones para eliminar el usuario indicado
0996                            pcPrEliminar();
0997                        }
0998                    }
0999                    //Opción REINICIAR CLAVE
1000                    if (grBxPrincipal.Text == "REINICIANDO ...") {
1001                        //Solicitar confirmación antes de reiniciar la clave del usuario
1002                        if (MessageBox.Show("¿ESTÁ SEGURO DE REINICIARLE CONTRASEÑA " +
1003                            "AL USUARIO?", "CONFIRMACIÓN", MessageBoxButtons.OKCancel,
1004                            MessageBoxIcon.Question) == DialogResult.Cancel) {
1005                            //Se cancela la opción
1006                            pcPrCancelar(true);
```

Continuación de la figura 114.

```
1007                        }
1008                        else {
1009                            //Se hacen las validaciones para reiniciarle contraseña al usuario indicado
1010                            pcPrReiniciarClave();
1011                        }
1012                    }
1013                }
1014                break;
1015            default:
1016                break;
1017        }
1018    }
1019
1020    //Si se presiona el botón CANCELAR
1021    private void pcPrClicCancelar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
1022        //Se llama al procedimiento pcPrCancelar
1023        pcPrCancelar(true);
1024    }
1025
1026    //Al dar clic en la imagen AGREGAR, se habilitan los controles
1027    private void pcPrClicAgregar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
1028        //Si se está en alguna opción
1029        if (grBxPrincipal.Text != "") {
1030            pcPrCancelar(false);
1031        }
1032
1033        grBxPrincipal.Text = "AGREGANDO ...";
1034        grBxPrincipal.Size = new System.Drawing.Size(598, 193);
1035        this.Size = new System.Drawing.Size(631, 295);
1036
1037        //Habilitar los controles
1038        pcPrDeshabilitarControles(true);
1039        pcPrDeshabilitarBotones(true, 151);
1040        pnlClave.Visible = false;
1041        pnlPreguntas.Visible = false;
1042
1043        //Limpiar los controles
1044        pcPrInicializarDatos();
1045        txtUsuario.Focus();
1046    }
1047
1048    //Al dar clic en la imagen REINICIAR CONTRASEÑA, se habilitan los controles
1049    private void pcPrClicReiniciarClave(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
1050        //Indicarle al usuario que primero debe ingresar el ID del usuario a quien se le
1051        //reiniciará la clave
1052        pcPrParaBuscarDatosUsuario("REINICIANDO ...", "INGRESE EL ID DEL USUARIO A " +
1053            "QUIEN LE REINICIARÁ LA CONTRASEÑA");
1054    }
1055
1056    //Al dar clic en la imagen EDITAR, se habilitan los controles
1057    private void pcPrClicEditar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
1058        //Si se está en alguna opción
1059        if (grBxPrincipal.Text != "") {
```

Continuación de la figura 114.

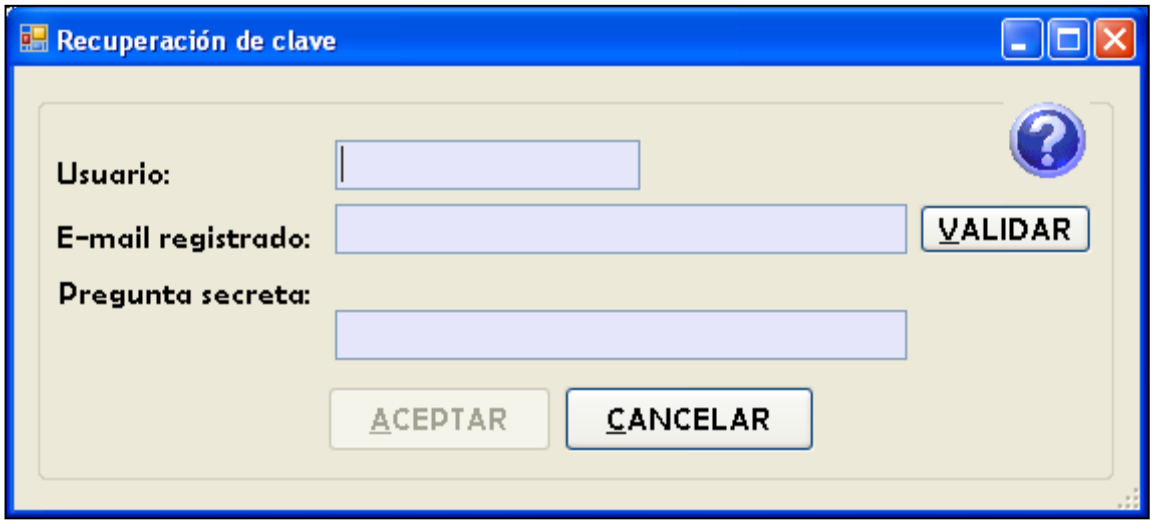
```
1060                pcPrCancelar(true);
1061            }
1062
1063            grBxPrincipal.Text = "EDITANDO ...";
1064            grBxPrincipal.Size = new System.Drawing.Size(598, 504);
1065            this.Size = new System.Drawing.Size(631, 607);
1066
1067            //Habilitar los controles
1068            pcPrDeshabilitarControles(true);
1069            pcPrDeshabilitarBotones(true, 460);
1070            //Si es el usuario administrador
1071            if (mdPbStrUsuario == "ADMIN") {
1072                txtNombres.Focus();
1073            }
1074            else {
1075                txtUsuario.Focus();
1076            }
1077        }
1078
1079        //Al dar clic en la imagen ELIMINAR, se habilitan los controles
1080        private void pcPrClicEliminar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
1081            //Indicarle al usuario que primero debe ingresar el ID del usuario a eliminar
1082            pcPrParaBuscarDatosUsuario("ELIMINANDO ...", "INGRESE EL ID DEL USUARIO " +
1083                "A ELIMINAR");
1084        }
1085
1086        //Al dar clic en el ícono de INFORME
1087        private void pcPrClicInforme(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
1088            //Llamar a la ventana del informe
1089            frmInforme vrLfrmInforme = new frmInforme();
1090            vrLfrmInforme.ShowDialog(this);
1091        }
1092
1093        //Al dar clic en el ícono de AYUDA
1094        private void pcPrClicAyudaDatosUsuario(object pVObjTipoObjeto,
1095            EventArgs pVEvtEvento) {
1096            //Llamar a la ventana de ayuda en línea
1097            frmAyuda vrLfrmAyudaDatosUsuario = new frmAyuda();
1098            vrLfrmAyudaDatosUsuario.mdPbStrNomPadre = this.Name;
1099            vrLfrmAyudaDatosUsuario.mdPbStrNomOpcion = grBxPrincipal.Text;
1100            vrLfrmAyudaDatosUsuario.mdPbStrNomModalidad = mdPbStrUsuario;
1101            vrLfrmAyudaDatosUsuario.ShowDialog(this);
1102        }
1103    }
1104 }
```

Fuente: elaboración propia.

Si se presiona el botón RECUPERAR CLAVE, se debe ingresar el usuario y correo electrónico registrados en el sistema, ya que al presiona el botón VALIDAR, con base a estos datos se ubica al usuario; si no coinciden los datos, no se podrá recuperar la clave.

Cuando se da clic en VALIDAR, se muestra la pregunta secreta retornada aleatoriamente por el procedimiento pcBuscarUsuario, y se habilita el control para que el usuario indique la respuesta. Si la respuesta ingresada en el formulario coincide exactamente con la registrada en la base de datos, se envían las nuevas credenciales de acceso (figura 111).

Figura 115. **Opción RECUPERAR CLAVE**



Fuente: elaboración propia.

El siguiente código corresponde a frmRecuperacionClave:

Figura 116. **Código de RecuperacionClave.cs**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                      RecuperacionClave.cs                                              */
0003  /*   Descripción:                  Forma para cambiar la clave.                                      */
0004  /*   Autor:                        Norma M. Chonay V.                                                */
0005  /*   Fecha de creación:            27/02/2011                                                        */
0006  /*   Fecha de última modificación: 27/02/2011                                                        */
0007  /*   Autor de última modificación: Norma M. Chonay V.                                                */
0008  /*   Versión:                      1.1                                                               */
0009  /*   Prerrequisitos:               Ninguno                                                           */
0010  /*****************************************************************************************************/
0011
0012  using System;
0013  using System.Collections.Generic;
0014  using System.ComponentModel;
0015  using System.Data;
0016  using System.Drawing;
0017  using System.Linq;
0018  using System.Text;
0019  using System.Windows.Forms;
0020  using RecuperarContraseña.Librerias;
0021
0022  namespace RecuperarContraseña.modSeguridad {
0023      public partial class frmRecuperacionClave : Form {
0024          public frmRecuperacionClave() {
0025              InitializeComponent();
0026          }
0027
0028          //Instanciar la librería de utilidades
0029          modHerramientas modHerr = new modHerramientas();
0030
0031          /*****************************************************************************************/
0032          /********************** VALIDACIÓN DE DATOS EN LOS CONTROLES *****************/
0033          /*****************************************************************************************/
0034          //Valida los carácteres ingresados en el control txtEmail
0035          private void pcPrValidarTxtEmail(object pVObjTipoObjeto, KeyPressEventArgs pVKpTecla) {
0036              modHerr.pcPbValidarCaracteres(ref pVKpTecla, btnValidar, 3);
0037          }
0038
0039          //Valida los carácteres ingresados en el control txtUsuario
0040          private void pcPrValidarTxtUsuario(object pVObjTipoObjeto,
0041              KeyPressEventArgs pVKpTecla) {
0042              modHerr.pcPbValidarCaracteres(ref pVKpTecla, txtEmail, 1);
0043          }
0044
0045          //Valida los carácteres ingresados en el control txtRespuesta
0046          private void pcPrValidarTxtRespuesta(object pVObjTipoObjeto,
0047                  KeyPressEventArgs pVKpTecla) {
0048              modHerr.pcPbValidarCaracteres(ref pVKpTecla, btnAceptar, 5);
0049          }
0050
0051          /*****************************************************************************************/
0052          /****************************** EVENTOS DE LOS CONTROLES *************************/
0053          /*****************************************************************************************/
```

Continuación de la figura 116.

```
0054        //Si se presiona el botón VALIDAR, se validan los datos ingresados y se muestra
0055        //una de las preguntas secretas registradas para el usuario
0056        private void pcPrClicValidar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0057            try {
0058                //Validar que no estén vacíos los controles
0059                if (txtUsuario.Text != "" & txtEmail.Text != "") {
0060                    //Validar la dirección de correo electrónico
0061                    if (modHerr.fPbBolValidarEmail(txtEmail.Text) == true) {
0062                        DataTable vrLTblDatos = new DataTable();
0063                        Int16 vrLsIntResultado = 0;
0064
0065                        //Conectarse a SQL
0066                        System.Data.SqlClient.SqlConnection vrLSqlConexion;
0067                        vrLSqlConexion = modHerr.fPbSqlConectar();
0068
0069                        //Consultar en la BDD si el usuario y el email coinciden
0070                        vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text.ToUpper().Trim(),
0071                            "", txtEmail.Text.ToUpper().Trim(), 0, ref vrLSqlConexion,
0072                            ref vrLsIntResultado);
0073                        switch (vrLsIntResultado) {
0074                            case 0:
0075                                MessageBox.Show("NO SE PUEDE RECUPERAR LA CLAVE, LA " +
0076                                    "INFORMACIÓN INDICADA NO ESTÁ REGISTRADA EN EL SISTEMA",
0077                                    "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0078                                break;
0079                            case 1:
0080                                //Validar si la consulta devolvió datos
0081                                if (vrLTblDatos.Rows.Count == 1) {
0082                                    //Mostrar la pregunta secreta
0083                                    if (vrLTblDatos.Rows[0]["Pregunta"].ToString() != "") {
0084                                        lblPregunta.Text = vrLTblDatos.Rows[0]["Pregunta"].ToString();
0085                                        mdPbStrRespuesta = vrLTblDatos.Rows[0]["Respuesta"].ToString();
0086                                        btnAceptar.Enabled = true;
0087                                        txtRespuesta.Enabled = true;
0088                                        txtRespuesta.Focus();
0089                                    }
0090                                    else {
0091                                        MessageBox.Show("NO SE PUEDE RECUPERAR LA CLAVE, EL " +
0092                                            "USUARIO NO TIENE REGISTRADA NINGUNA PREGUNTA " +
0093                                            "SECRETA", "ERROR", MessageBoxButtons.OK,
0094                                            MessageBoxIcon.Error);
0095                                        vrLSqlConexion.Close();
0096                                        this.Dispose();
0097                                        this.Close();
0098                                    }
0099                                }
0100                                else {
0101                                    MessageBox.Show("LA INFORMACIÓN INDICADA, NO ESTÁ " +
0102                                        "REGISTRADA EN EL SISTEMA", "ERROR", MessageBoxButtons.OK,
0103                                        MessageBoxIcon.Error);
0104                                    txtUsuario.Focus();
0105                                }
0106                                break;
```

Continuación de la figura 116.

```
0107                        case 2:
0108                            MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " +
0109                                "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0110                            vrLSqlConexion.Close();
0111                            this.Dispose();
0112                            this.Close();
0113                            break;
0114                        case 3:
0115                            MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS", "ERROR",
0116                                MessageBoxButtons.OK, MessageBoxIcon.Error);
0117                            vrLSqlConexion.Close();
0118                            this.Dispose();
0119                            this.Close();
0120                            break;
0121                        default:
0122                            //Error mostrado en la librería
0123                            break;
0124                    }
0125                    vrLSqlConexion.Close();
0126                }
0127                else {
0128                    MessageBox.Show("LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " +
0129                        "VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0130                    txtEmail.Focus();
0131                }
0132            }
0133            else {
0134                MessageBox.Show("TODOS LOS CAMPOS SON OBLIGATORIOS", "ERROR",
0135                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
0136                txtUsuario.Focus();
0137            }
0138        }
0139        catch (Exception pVExExcepcion) {
0140            MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0141                MessageBoxIcon.Error);
0142            txtUsuario.Focus();
0143        }
0144    }
0145
0146    //Si se presiona el botón ACEPTAR y si los datos ingresados son válidos,
0147    //se reinicia la contraseña
0148    private void pcPrClicAceptar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0149        try {
0150            txtRespuesta.Text = txtRespuesta.Text.Trim();
0151            txtUsuario.Text = txtUsuario.Text.ToUpper().Trim();
0152            txtEmail.Text = txtEmail.Text.ToUpper().Trim();
0153
0154            //Validar que no estén vacíos
0155            if (txtUsuario.Text != "" & txtEmail.Text != "" & txtRespuesta.Text != "") {
0156                //Validar la dirección de correo electrónico
0157                if (modHerr.fPbBolValidarEmail(txtEmail.Text) == true) {
0158                    //Validar que la respuesta sea la que está registrada
0159                    if (mdPbStrRespuesta == txtRespuesta.Text) {
```

Continuación de la figura 116.

```
0160                    DataTable vrLTblDatos = new DataTable();
0161                    Int16 vrLsIntResultado = 0;
0162                    string[] vrLvNuevaClave;
0163
0164                    //Conectarse a SQL
0165                    System.Data.SqlClient.SqlConnection vrLSqlConexion;
0166                    vrLSqlConexion = modHerr.fPbSqlConectar();
0167
0168                    //Consultar en la BDD si el usuario y el email coinciden
0169                    vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text,
0170                       "", txtEmail.Text, 0, ref vrLSqlConexion,
0171                       ref vrLsIntResultado);
0172                    switch (vrLsIntResultado) {
0173                       case 0:
0174                          MessageBox.Show("NO SE PUEDE RECUPERAR LA CLAVE, LA " +
0175                                "INFORMACIÓN INDICADA NO ESTÁ REGISTRADA EN EL " +
0176                                "SISTEMA", "ERROR", MessageBoxButtons.OK,
0177                                MessageBoxIcon.Error);
0178                          break;
0179                       case 1:
0180                          //Validar si la consulta devolvió datos
0181                          if (vrLTblDatos.Rows.Count == 1) {
0182                             //Obtener la nueva clave
0183                             vrLvNuevaClave = modHerr.fPbVGenerarClave(txtUsuario.Text);
0184                             if (vrLvNuevaClave[0] != "" & vrLvNuevaClave[1] != "") {
0185                                //Modificar los datos en la BDD
0186                                vrLsIntResultado = 0;
0187                                modHerr.pcPbModificarUsuario(
0188                                   Convert.ToInt32(vrLTblDatos.Rows[0]["Id"]),
0189                                   vrLTblDatos.Rows[0]["Usuario"].ToString(),
0190                                   vrLvNuevaClave[1],
0191                                   vrLTblDatos.Rows[0]["Nombres"].ToString(),
0192                                   vrLTblDatos.Rows[0]["Apellidos"].ToString(),
0193                                   vrLTblDatos.Rows[0]["EMail"].ToString(),
0194                                   "", "", "", "", "", 0,
0195                                   ref vrLSqlConexion, ref vrLsIntResultado);
0196
0197                                //Si no hubo error al actualizar la clave en la BDD
0198                                switch (vrLsIntResultado) {
0199                                   case 0:
0200                                      MessageBox.Show("NO SE REALIZÓ LA ACTUALIZACIÓN " +
0201                                            "DE LA CONTRASEÑA, EL USUARIO NO EXISTE",
0202                                            "ERROR", MessageBoxButtons.OK,
0203                                            MessageBoxIcon.Error);
0204                                      break;
0205                                   case 1:
0206                                      //Enviar el correo con la nueva contraseña
0207                                      if (modHerr.fPbBolEnviarEmail(
0208                                         vrLTblDatos.Rows[0]["EMail"].ToString(),
0209                                         vrLTblDatos.Rows[0]["Usuario"].ToString(),
0210                                         vrLvNuevaClave[0], 2) == true) {
0211                                         MessageBox.Show("LA NUEVA CONTRASEÑA FUE " +
0212                                               "ENVIADA A LA DIRECCIÓN DE CORREO " +
```

Continuación de la figura 116.

```
0213                                    "REGISTRADA EN EL SISTEMA.", "NOTIFICACIÓN",
0214                                    MessageBoxButtons.OK,
0215                                    MessageBoxIcon.Information);
0216                                }
0217                                else {
0218                                    MessageBox.Show("SE ACTUALIZÓ LA CONTRASEÑA, " +
0219                                        "PERO HUBO INCONVENIENTE AL ENVIAR LAS " +
0220                                        "NUEVAS CREDENCIALES A LA CUENTA DE " +
0221                                        "CORREO REGISTRADA", "ERROR",
0222                                        MessageBoxButtons.OK, MessageBoxIcon.Error);
0223                                }
0224                                break;
0225                            case 3:
0226                                MessageBox.Show("NO SE REALIZÓ LA ACTUALIZACIÓN " +
0227                                    "DE LA CONTRASEÑA, HUBO UN ERROR EN LA " +
0228                                    "BASE DE DATOS", "ERROR", MessageBoxButtons.OK,
0229                                    MessageBoxIcon.Error);
0230                                break;
0231                            default:
0232                                //Error mostrado en la librería
0233                                break;
0234                        }
0235                    }
0236                    else {
0237                        MessageBox.Show("NO SE PUDO GENERAR UNA NUEVA " +
0238                            "CONTRASEÑA", "ERROR", MessageBoxButtons.OK,
0239                            MessageBoxIcon.Error);
0240                    }
0241                }
0242                else {
0243                    MessageBox.Show("LA INFORMACIÓN INDICADA, NO ESTÁ " +
0244                        "REGISTRADA EN EL SISTEMA", "ERROR",
0245                        MessageBoxButtons.OK, MessageBoxIcon.Error);
0246                }
0247                break;
0248            case 2:
0249                MessageBox.Show("LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " +
0250                    "VÁLIDA", "ERROR", MessageBoxButtons.OK,
0251                    MessageBoxIcon.Error);
0252                break;
0253            case 3:
0254                MessageBox.Show("HUBO UN ERROR EN LA BASE DE DATOS",
0255                    "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
0256                break;
0257            default:
0258                //Error mostrado en la librería
0259                break;
0260        }
0261
0262        vrLSqlConexion.Close();
0263        this.Dispose();
0264        this.Close();
0265    }
```

Continuación de la figura 116.

```
0266                    else {
0267                        MessageBox.Show("LA RESPUESTA NO ES VÁLIDA", "ERROR",
0268                            MessageBoxButtons.OK, MessageBoxIcon.Warning);
0269                        txtRespuesta.Focus();
0270                    }
0271                }
0272                else {
0273                    MessageBox.Show("LA DIRECCIÓN DE CORREO ELECTRÓNICO NO " +
0274                        "ES VÁLIDA", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
0275                    txtEmail.Focus();
0276                }
0277            }
0278            else {
0279                MessageBox.Show("TODOS LOS CAMPOS SON OBLIGATORIOS", "ERROR",
0280                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
0281                txtUsuario.Focus();
0282            }
0283        }
0284        catch (Exception pVExExcepcion) {
0285            MessageBox.Show(pVExExcepcion.ToString(), "ERROR", MessageBoxButtons.OK,
0286                MessageBoxIcon.Error);
0287            txtUsuario.Focus();
0288        }
0289    }
0290
0291    //Si se presiona el botón CANCELAR, se limpia el texto ingresado en los controles
0292    private void pcPrClicCancelar(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0293        txtUsuario.Text = "";
0294        txtEmail.Text = "";
0295        txtRespuesta.Text = "";
0296        lblPregunta.Text = "Pregunta secreta:";
0297        btnAceptar.Enabled = false;
0298        txtRespuesta.Enabled = false;
0299        txtUsuario.Focus();
0300    }
0301
0302    //Al dar clic en el ícono de AYUDA
0303    private void pcPrAyudaRecupClave(object pVObjTipoObjeto, EventArgs pVEvtEvento) {
0304        //Llamar a la ventana de ayuda en línea
0305        frmAyuda vrLfrmAyudaRecupClave = new frmAyuda();
0306        vrLfrmAyudaRecupClave.mdPbStrNomPadre = this.Name;
0307        vrLfrmAyudaRecupClave.mdPbStrNomOpcion = "";
0308        vrLfrmAyudaRecupClave.mdPbStrNomModalidad = "";
0309        vrLfrmAyudaRecupClave.ShowDialog(this);
0310    }
0311  }
0312 }
```

Fuente: elaboración propia.

Como se puede apreciar en el código expuesto anteriormente, se aplicó la metodología EXTILO REGULAR 9002 al escribir cada línea, modularizando en subrutinas los procesos para utilizarlos desde diferentes partes de la aplicación, variando los valores enviados por parámetro para obtener el comportamiento esperado.

Los colores, tipo de letra e íconos son uniformes en todos los formularios; se manejan excepciones en los procedimientos y funciones solamente en los que llaman a otras subrutinas y en los que se comunican con la base de datos. Las recomendaciones con respecto al uso de sangría también fueron aplicadas; y por supuesto las indicaciones para nombrar a los elementos de la base de datos y de la aplicación. Permitiendo así que el código de cualquiera de los prototipos, pueda modificarse por un analista programador que sepa programar. Si la persona conoce el lenguaje de programación utilizado, invertirá menos tiempo en entender el código, que alguien que desconozca el lenguaje.

CONCLUSIONES

1. La metodología Extilo Regular 9002 propone estándares para programar *software* en Guatemala, tomando de referencia como lenguajes de programación Visual Basic .Net, Visual C#, PHP, Java y C++, y los sistemas gestores de base de datos Oracle, *Microsoft* SQL, MySQL y PostgreSQL.

2. Analizando la sintaxis y semántica de los lenguajes de programación, base de la metodología, se propuso la nomenclatura para nombrar los elementos que implementan y la sangría para escribir estructuras selectivas y repetitivas, para adoptar buenas prácticas de programación.

3. Los prototipos desarrollados utilizan la metodología propuesta, poniendo de manifiesto que sí es factible aplicar estos estándares, indistintamente del objetivo del sistema programado y del lenguaje de programación utilizado.

4. Al ser distribuido el folleto con la metodología, entre los estudiantes que cursen el laboratorio de los cursos de programación de la red curricular actual de la carrera de Ingeniería en Ciencias y Sistemas, de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, ellos la podrán poner en práctica para escribir código estandarizado que contará con todos los beneficios que conlleva la normalización.

RECOMENDACIONES

1. Por el ritmo tan acelerado al que va avanzando la tecnología, se espera que surjan nuevos paradigmas o herramientas de programación, y para que ésta metodología se mantenga actualizada, se sugiere revisar Extilo Regular 9002 para cambiar lo que corresponda, y así ésta pueda seguir siendo aplicada a cualquier lenguaje de programación.

2. Como Extilo Regular 9002 estandariza la escritura del código en los programas de *software*, se sugiere, diseñar una metodología para documentar internamente el código fuente, que pueda ser aplicada a diversos lenguajes de programación para desarrollar cualquier aplicación, y así cualquier persona del equipo de Desarrollo pueda interpretar la misma fácilmente y actualizarla; logrando con ello la normalización en este tipo de documentación.

3. Que el personal docente de la carrera de Ingeniería en Ciencias y Sistemas, de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, fomente la práctica de Extilo Regular 9002 entre los estudiantes, asistiéndolos con resolución de dudas y promoviendo la importancia de la estandarización, estimulando así el compromiso e interés de los alumnos para ponerla en práctica, lo cual les ayudará en su desarrollo profesional.

BIBLIOGRAFÍA

1. AGEXPORT. *Empresas Socias en Proceso de Certificación CMMI* (*Capability Maturity Model Integration*) [en línea]. AGEXPORT 2011. Disponible en Web: <http://www.export.com.gt/Portal/Entities/ShowContent.aspx?Eid=6788&Iid=2713&Path=Documents/News/2009-06/6788/Noticia-Empresas%20Socias%20en%20Proceso%20de%20Certificaci%C3%B3n%20CMMI.doc&ContentType=application/msword>. [Consulta: 30 de agosto de 2009].

2. ANSI. *American National Standards Institute* [en línea]. Disponible en Web: <http://www.ansi.org>. [Consulta: 01 de julio de 2009].

3. BOISE State University. *The elements of programming style* [en línea]. Abril de 1999. Disponible en Web: <http://cs.boisestate.edu/~amit/teaching/handouts/style.pdf>. [Consulta: 27 de agosto de 2009].

4. BSA. *Business Software Alliance* [en línea]. Disponible en Web: <http://www.bsa.org>. [Consulta: 02 de agosto de 2009].

5. CARMEL, Erran. *The New Software Exporting Nations: Success Factors* [en línea]. *American University*, Washington D.C., USA. 2003. Disponible en Web: <http://www.ejisdc.org/ojs2/index.php/ejisdc/article/viewFile/78/78>. [Consulta: 20 de junio de 2009].

6. CONCHA HURTADO, Nancy Elizabet. *Propuesta para implantar CMMI en una empresa con múltiples unidades desarrolladores de software* [en línea]. Tesis digitales UNMSM. Disponible en Web: <http://sisbib.unmsm.edu.pe/bibvirtualdata/tesis/Basic/concha_hn/cap1.pdf>. [Consulta: 30 de agosto de 2009].

7. GARCÍA PEREZ-SCHOFIELD, J. Baltasar. *Guía de estilo de programación en C++* [en línea]. Escuela Superior de Ingeniería Informática, Universidad de Vigo. Disponible en Web: <http://trevinca.ei.uvigo.es/~jgarcia/TO/guiaestilocpp.pdf>. [Consulta: 01 de enero de 2011].

8. GÓMEZ, Daira. *La industria del software costarricense y los mercados internacionales* [en línea]. CEGESTI. Septiembre de 2004. Disponible en Web: <http://www.cegesti.org/exitoempresarial/publications/resumido%20Software.pdf>. [Consulta: 11 de junio de 2009].

9. HTMLPOINT.com. *PHP teórico: Tipos de datos* [en línea]. Disponible en Web: <http://www.htmlpoint.com/php/guida/php_10.htm>. [Consulta: 23 de noviembre de 2009].

10. HTMLQuik.com. *Tag HTML* [en línea]. Disponible en Web: <http://www.htmlquick.com/es/reference/tags.html>. [Consulta: 01 de diciembre de 2009].

11. ISO. *International Organization for Standardization* [en línea]. Disponible en Web: <http://www.iso.org>. [Consulta: 01 de julio de 2009].

12. Junta de Castilla y León. *Guía de iniciación al lenguaje JAVA* [en línea]. Universidad de Burgos. Octubre de 1999. Disponible en Web: <http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/II_2.htm>. [Consulta: 24 de noviembre de 2009].

13. MAILXMAIL.COM. *Tipos de estructuras de programación: Estructuras básicas y secuencial* [en línea]. Marzo de 2005. Disponible en Web: <http://www.mailxmail.com/curso-aprende-programar/tipos-estructuras-programacion-estructuras-basicas-secuencial>. [Consulta: 12 de marzo de 2010].

14. MSDN. *Data Types (Transact-SQL)* [en línea]. Microsoft®. Disponible en Web: <http://msdn.microsoft.com/en-us/library/ms187752.aspx>. [Consulta: 28 de noviembre de 2009].

15. _____. *How to: verify that strings are in valid e-mail format* [en línea]. Microsoft®. Disponible en Web: <http://msdn.microsoft.com/en-us/library/01escwtf.aspx>. [Consulta: 27 de febrero de 2011].

16. _____. *Resumen de tipos de datos (Visual Basic)* [en línea]. Microsoft®. Disponible en Web: <http://msdn.microsoft.com/es-es/library/47zcea w7(VS.80).aspx>. [Consulta: 22 de noviembre de 2009].

17. _____. *Tipos de datos comparados en diferentes lenguajes* [en línea]. Microsoft®. Disponible en Web: <http://msdn.microsoft.com/es-es/library/4xwz0t37(VS.80).aspx>. [Consulta: 22 de noviembre de 2009].

18. MUNIVE HERRERA, Elizabeth; TREJO RAMÍREZ, Raúl. *A methodology for self-diagnosis for software quality assurance in small and medium-sized industries in Latin America* [en línea]. Instituto Tecnológico y de Estudios Superiores de Monterrey. 2003. Disponible en Web: <http://www.ejisdc.org/ojs2/index.php/ejisdc/article/viewFile/94/94>. [Consulta: 28 de junio de 2009].

19. MYSQL. *Data Types* [en línea]. Oracle®. Disponible en Web: <http://dev.mysql.com/doc/refman/5.0/en/data-types.html>. [Consulta: 28 de noviembre de 2009].

20. ORACLE. *Database concepts*. *Oracle Data Types* [en línea]. Disponible en Web: <http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/datatype.htm>. [Consulta: 25 de noviembre de 2009].

21. _____. *Database SQL Language Reference*. *Data Types* [en línea]. Disponible en Web: <http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/sql_elements001.htm>. [Consulta: 25 de noviembre de 2009].

22. ORÉ BRAVO, Alexander. *Introducción al CMMI – CMM* [en línea]. Calidad y *software*.com. Abril de 2008. Disponible en Web: <http://www.calidadysoftware.com/otros/introduccion_cmmi.php>. [Consulta: 30 de agosto de 2009].

23. PROEXPORT Colombia y Banco Interamericano de Desarrollo – Fondo Multilateral de Inversión (BID-FOMIN). *Análisis del sector software* [en línea]. Proexport Colombia. 2004. Disponible en Web: <http://www.proexport.com.co/VBeContent/library/documents/DocNewsNo8683DocumentNo7147.PDF>. [Consulta: 02 de junio de 2009].

24. *Software Engineering Institute, Carnegie Mellon*. *Published Appraisal Results* [en línea]. Disponible en Web: <http://sas.sei.cmu.edu/pars/pars_detail.aspx?a=13553>. [Consulta: 01 de octubre de 2011].

25. UMC. Universidad Nacional Experimental Marítima del Caribe, República Bolivariana de Venezuela. *Tipos de datos estándar de C y C++* [en línea]. Disponible en Web: <http://www.scribd.com/doc/2551829/Tipos-de-datos-en-C>. [Consulta: 24 de noviembre de 2009].

26. W3C. *The global structure of an HTML document* [en línea]. Disponible en Web: <http://www.w3.org/TR/html4/struct/global.html>. [Consulta: 02 de enero de 2011].

27. W3SCHOOLS. *HTML Tag List* [en línea]. Disponible en Web: <http://www.w3schools.com/tags/default.asp>. [Consulta: 03 de diciembre de 2009].

28. WIKIPEDIA. La enciclopedia libre. *CamelCase* [en línea]. Wikipedia®. Disponible en Web: <http://en.wikipedia.org/wiki/CamelCase>. [Consulta: 23 de agosto 2009].

29. _____. *Capability Maturity Model Integration* [en línea]. Wikipedia®. Disponible en Web: <http://es.wikipedia.org/wiki/CMMI>. [Consulta: 30 de agosto de 2009].

30. _____. *Charles Simonyi* [en línea]. Wikipedia®. Disponible en Web: <http://en.wikipedia.org/wiki/Charles_Simonyi>. [Consulta: 23 de agosto de 2009].

31. _____. *Code refactoring* [en línea]. Wikipedia®. Disponible en Web: <http://en.wikipedia.org/wiki/Code_refactoring>. [Consulta: 23 de agosto de 2009].

32. _____. *DICOM* [en línea]. Wikipedia®. Disponible en Web: <http://es.wikipedia.org/wiki/DICOM>. [Consulta: 22 de diciembre de 2010].

33. _____. *Globally Unique Identifier* [en línea]. Wikipedia®. Disponible en Web: <http://es.wikipedia.org/wiki/Globally_Unique_Identifier>. [Consulta: 22 de diciembre de 2010].

34. _____. *Hungarian notation* [en línea]. Wikipedia®. Disponible en Web: <http://en.wikipedia.org/wiki/Hungarian_notation>. [Consulta: 23 de agosto de 2009].

35. _____. *Modelo de capacidad y madurez* [en línea]. Wikipedia®. Disponible en Web: <http://es.wikipedia.org/wiki/Modelo_de_Capacidad_y_Madurez>. [Consulta: 30 de agosto de 2009].

36. _____. *Script (informática)* [en línea]. Disponible en Web: <http://es.wikipedia.org/wiki/Script_(inform%C3%A1tica)>. [Consulta: 02 de enero de 2011].

37. *WINDOWSCLIENT.NET*. *Como cifrar y descifrar texto usando C#* [en línea]. Mayo de 2009. Disponible en Web: <http://windowsclient.net /blogs/linkecubeko/archive/2009/05/01/como-cifrar-y-descifrar-texto-usando-c.aspx>. [Consulta: 27 de febrero de 2011].

APÉNDICES

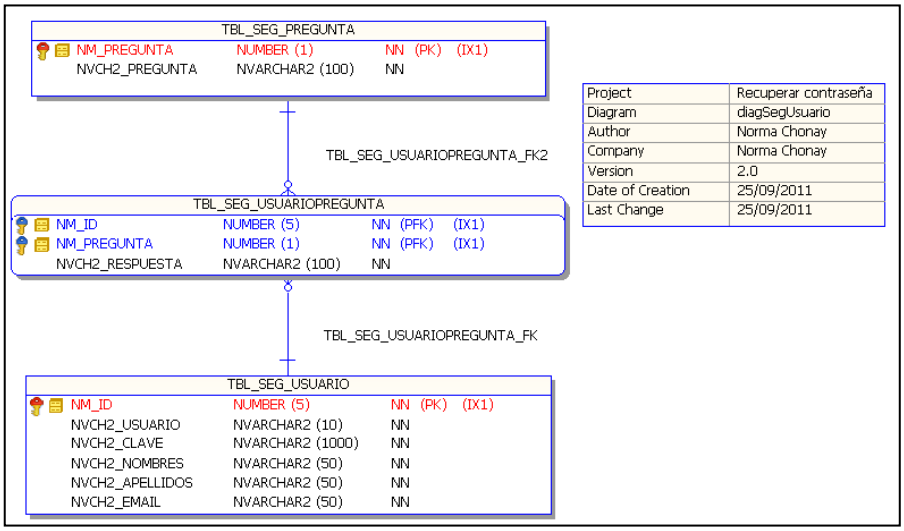
1. PROTOTIPO # 2: VISUAL BASIC .NET - ORACLE

Además de los lenguajes utilizados, este segundo ejemplo se diferencia del primer prototipo, en que es un proyecto *web* que invoca un servicio *web*, este servicio devuelve las 5 preguntas secretas almacenadas en la base de datos. Primero se muestran el diagrama y las subrutinas de la base de datos, luego se presentan las páginas y el código de las mismas; y por último el código asociado al servicio *web*.

1.1. Base de datos

En la siguiente figura se aprecian las tablas de la base de datos y las relaciones entre sí.

Figura A-1. **Diagrama de la base de datos**



Fuente: elaboración propia.

El siguiente código corresponde al procedimiento que crea un nuevo usuario en tbl_seg_usuario.

Figura A-2. **Procedimiento pc_guardarusuario**

```
0001  CREATE OR REPLACE PROCEDURE pc_guardarusuario (pr_nm_id OUT NUMBER,
0002          pv_nvch2_usuario IN NVARCHAR2, pv_nvch2_clave IN NVARCHAR2,
0003          pv_nvch2_nombres IN NVARCHAR2, pv_nvch2_apellidos IN NVARCHAR2,
0004          pv_nvch2_email IN NVARCHAR2, pr_nm_resultado OUT NUMBER)
0005  AS vr_l_nm_resultado NUMBER := 0;
0006  --Crea un usuario
0007  BEGIN
0008           --Consultar si el usuario existe
0009           SELECT COUNT(*)
0010           INTO vr_l_nm_resultado
0011           FROM tbl_seg_usuario
0012           WHERE nvch2_usuario = TRIM(pv_nvch2_usuario);
0013
0014           SELECT MAX(nm_id) + 1
0015           INTO pr_nm_id
0016           FROM tbl_seg_usuario;
0017
0018           IF (vr_l_nm_resultado = 0) THEN
0019                  INSERT INTO tbl_seg_usuario (nm_id, nvch2_usuario, nVch2_Clave,
0020                         nvch2_nombres, nvch2_apellidos, nvch2_email)
0021                  VALUES (pr_nm_id, UPPER(TRIM(pv_nvch2_usuario)), TRIM(pv_nvch2_clave),
0022                         UPPER(TRIM(pv_nvch2_nombres)),
0023                         UPPER(TRIM(pv_nvch2_apellidos)),
0024                         UPPER(TRIM(pv_nvch2_email)));
0025
0026                  pr_nm_resultado := 1; --El usuario fue creado
0027           ELSE
0028                  pr_nm_resultado := 0; --El usuario ya existe
0029           END IF;
0030  EXCEPTION
0031          WHEN OTHERS THEN
0032                  pr_nm_resultado := 3;
0033  END;
```

Fuente: elaboración propia.

Para modificar los datos de usuarios, se implementó la subrutina pc_modificarusuario.

Figura A-3. **Procedimiento pc_modificarusuario**

```
0001  CREATE OR REPLACE PROCEDURE pc_modificarusuario (pv_nm_id IN NUMBER,
0002          pv_nvch2_usuario IN NVARCHAR2, pv_nvch2_clave IN NVARCHAR2,
0003          pv_nvch2_nombres IN NVARCHAR2, pv_nvch2_apellidos IN NVARCHAR2,
0004          pv_nvch2_email IN NVARCHAR2, pv_nvch2_respuesta1 IN NVARCHAR2,
0005          pv_nvch2_respuesta2 IN NVARCHAR2, pv_nvch2_respuesta3 IN NVARCHAR2,
0006          pv_nvch2_respuesta4 IN NVARCHAR2, pv_nvch2_respuesta5 IN NVARCHAR2,
0007          pv_nm_opcion IN NUMBER, pr_nm_resultado OUT NUMBER)
0008  AS vr_l_nm_resultado NUMBER := 0;
0009  --Modifica un usuario
0010  BEGIN
0011          --Consultar si el usuario existe
0012          SELECT COUNT(*) INTO vr_l_nm_resultado
0013          FROM tbl_seg_usuario WHERE nm_id = pv_nm_id;
0014
0015          IF (vr_l_nm_resultado = 1) THEN
0016                  UPDATE tbl_seg_usuario
0017                  SET nvch2_usuario = UPPER(TRIM(pv_nvch2_usuario)),
0018                          nvch2_clave = TRIM(pv_nvch2_clave),
0019                          nvch2_nombres = UPPER(TRIM(pv_nvch2_nombres)),
0020                          nvch2_apellidos = UPPER(TRIM(pv_nvch2_apellidos)),
0021                          nvch2_email = UPPER(TRIM(pv_nvch2_email))
0022                  WHERE nm_id = pv_nm_id;
0023
0024                  IF (pv_nm_opcion = 1) THEN
0025                          DELETE FROM tbl_seg_usuariopregunta WHERE nm_id = pv_nm_id;
0026
0027                          IF (pv_nvch2_respuesta1 IS NOT NULL) THEN --Respuesta 1
0028                                  INSERT INTO tbl_seg_usuariopregunta (nm_id, nm_pregunta,
0029                                          nvch2_respuesta)
0030                                  VALUES (pv_nm_id, 1, pv_nvch2_respuesta1);
0031                          END IF;
0032                          IF (pv_nvch2_respuesta2 IS NOT NULL) THEN --Respuesta 2
0033                                  INSERT INTO tbl_seg_usuarioPregunta (nm_Id,
0034                                          nm_pregunta, nvch2_respuesta)
0035                                  VALUES (pv_nm_id, 2, pv_nvch2_respuesta2);
0036                          END IF;
0037                          IF (pv_nvch2_respuesta3 IS NOT NULL) THEN --Respuesta 3
0038                                  INSERT INTO tbl_seg_usuarioPregunta (nm_Id,
0039                                          nm_pregunta, nvch2_respuesta)
0040                                  VALUES (pv_nm_id, 3, pv_nvch2_respuesta3);
0041                          END IF;
0042                          IF (pv_nvch2_respuesta4 IS NOT NULL) THEN --Respuesta 4
0043                                  INSERT INTO tbl_seg_usuarioPregunta (nm_Id,
0044                                          nm_pregunta, nvch2_respuesta)
0045                                  VALUES (pv_nm_id, 4, pv_nvch2_respuesta4);
0046                          END IF;
0047                          IF (pv_nvch2_respuesta5 IS NOT NULL) THEN --Respuesta 5
0048                                  INSERT INTO tbl_seg_usuarioPregunta (nm_Id,
0049                                          nm_pregunta, nvch2_respuesta)
0050                                  VALUES (pv_nm_id, 5, pv_nvch2_respuesta5);
0051                          END IF;
0052                  END IF;
0053
```

Continuación de la figura A-3.

```
0054                    pr_nm_resultado := 1; --Los datos del usuario se actualizaron correctamente
0055            ELSE
0056                    pr_nm_resultado := 0; --El usuario no existe
0057            END IF;
0058    EXCEPTION
0059            WHEN OTHERS THEN pr_nm_resultado := 3;
0060    END;
```

Fuente: elaboración propia.

El procedimiento pc_eliminarusuario, elimina el usuario indicado por parámetro, siempre y cuando no sea el usuario ADMIN, cuyo *ID* tiene valor de 1.

Figura A-4. **Procedimiento pc_eliminarusuario**

```
0001    CREATE OR REPLACE PROCEDURE pc_eliminarusuario (pv_nm_id IN NUMBER,
0002            pr_nm_resultado OUT NUMBER)
0003    AS vr_l_nm_resultado NUMBER := 0;
0004    --Elimina un usuario
0005    BEGIN
0006            --Consultar si el usuario existe
0007            SELECT COUNT(*)
0008            INTO vr_l_nm_resultado
0009            FROM tbl_seg_usuario
0010            WHERE nm_id = pv_nm_id
0011                    AND pv_nm_id <> 1; --El usuario ADMIN no se puede eliminar
0012
0013            IF (vr_l_nm_resultado = 1) THEN
0014                    DELETE
0015                    FROM tbl_seg_usuarioPregunta
0016                    WHERE nm_id = pv_nm_id;
0017
0018                    DELETE
0019                    FROM tbl_seg_usuario
0020                    WHERE nm_id = pv_nm_id;
0021
0022                    pr_nm_resultado := 1; --El usuario se eliminó exitosamente
0023            ELSE
0024                    pr_nm_resultado := 0; --El usuario no existe
0025            END IF;
```

Continuación de la figura A-4.

```
0026  EXCEPTION
0027        WHEN OTHERS THEN
0028              pr_nm_resultado := 3;
0029  END;
```

Fuente: elaboración propia.

La siguiente subrutina devuelve determinados datos de los usuarios, dependiendo del valor del parámetro de entrada pv_nm_opcion:

- 0: devuelve todos los datos de tbl_seg_usuario y tbl_seg_usuariopregunta, buscando por nombre de usuario y correo electrónico.
- 1: consulta si el nombre de usuario ya existe, para un *ID* que no sea el recibido por el parámetro pv_nm_id.
- 2: devuelve todos los datos de tbl_seg_usuario, buscando por nombre de usuario y clave.
- 3: devuelve datos de tbl_seg_usuario, buscando por *ID*.
- 4: devuelve datos de tbl_seg_usuario para todos los usuarios, indicando para cada uno si tiene o no una pregunta secreta en tbl_seg_usuariopregunta.

Figura A-5. **Procedimiento pc_buscarusuario**

```
0001  CREATE OR REPLACE PROCEDURE pc_buscarusuario (pv_nm_id IN NUMBER,
0002        pv_nvch2_usuario IN NVARCHAR2, pv_nvch2_clave IN NVARCHAR2,
0003        pv_nvch2_email IN NVARCHAR2, pv_nm_opcion IN NUMBER,
0004        pr_nm_resultado OUT NUMBER, pr_crs_datos OUT SYS_REFCURSOR)
0005  AS vr_l_nm_resultado NUMBER := 0;
0006  --Busca datos de usuarios
0007  BEGIN
0008        --Consultar si el usuario y el email coinciden
0009        IF pv_nm_opcion = 0 THEN
0010              SELECT COUNT(*)
0011              INTO vr_l_nm_resultado
0012              FROM tbl_seg_usuario
0013              WHERE nvch2_usuario = TRIM(pv_nvch2_usuario)
0014                    AND nvch2_email = TRIM(pv_nvch2_email);
```

Continuación de la figura A-5.

```
0015
0016                    IF (vr_l_nm_resultado = 1) THEN
0017                            OPEN pr_crs_datos FOR
0018                            SELECT * FROM
0019                              (SELECT Usuario.nm_id As Id,
0020                                      Usuario.nvch2_usuario As Usuario,
0021                                      Usuario.nvch2_nombres As Nombres,
0022                                      Usuario.nvch2_apellidos As Apellidos,
0023                                      Usuario.nvch2_email As EMail,
0024                                      Pregunta.nvch2_pregunta As Pregunta,
0025                                      Resp.nvch2_respuesta As Respuesta
0026                              FROM tbl_seg_usuario Usuario
0027                              LEFT JOIN tbl_seg_usuariopregunta Resp
0028                                      ON (Resp.nm_id = Usuario.nm_id)
0029                              LEFT JOIN tbl_seg_pregunta Pregunta
0030                                      ON (Pregunta.nm_Pregunta = Resp.nm_Pregunta)
0031                              WHERE Usuario.nvch2_usuario = TRIM(pv_nvch2_usuario)
0032                                      AND Usuario.nvch2_email = TRIM(pv_nvch2_email)
0033                              ORDER BY DBMS_RANDOM.VALUE)
0034                            WHERE ROWNUM = 1;
0035
0036                            pr_nm_resultado := 1; --El usuario y el email sí son correctos
0037                    ELSE
0038                            pr_nm_resultado := 0; --Los datos del usuario no son correctos
0039                    END IF;
0040            ELSE
0041                    --Consultar si el usuario existe
0042                    IF pv_nm_opcion = 1 THEN
0043                            SELECT COUNT(*)
0044                            INTO vr_l_nm_resultado
0045                            FROM tbl_seg_usuario
0046                            WHERE nvch2_usuario = TRIM(pv_nvch2_usuario)
0047                                    AND nm_id <> pv_nm_id;
0048
0049                            IF (vr_l_nm_resultado = 1) THEN
0050                                    pr_nm_resultado := 1; --El usuario sí existe
0051                            ELSE
0052                                    pr_nm_resultado := 0; --No existe el usuario
0053                            END IF;
0054                    ELSE
0055                            --Consultar si el usuario y la clave son correctos
0056                            IF pv_nm_opcion = 2 THEN
0057                              SELECT COUNT(*)
0058                              INTO vr_l_nm_resultado
0059                              FROM tbl_seg_usuario
0060                              WHERE nvch2_usuario = TRIM(pv_nvch2_usuario)
0061                                      AND nvch2_clave = TRIM(pv_nvch2_clave);
0062
0063                              IF (vr_l_nm_resultado = 1) THEN
0064                                      OPEN pr_crs_datos FOR
0065                                      SELECT Usuario.nm_id As Id,
0066                                              Usuario.nvch2_usuario As Usuario,
0067                                              Usuario.nvch2_clave As Clave,
```

Continuación de la figura A-5.

```
0068                                    Usuario.nvch2_nombres As Nombres,
0069                                    Usuario.nvch2_apellidos As Apellidos,
0070                                    Usuario.nvch2_email As EMail
0071                            FROM tbl_seg_usuario Usuario
0072                            WHERE Usuario.nvch2_usuario = TRIM(pv_nvch2_usuario)
0073                                    AND Usuario.nvch2_clave = TRIM(pv_nvch2_clave);
0074
0075                            pr_nm_resultado := 1; --El usuario y la clave sí son correctos
0076                    ELSE
0077                            pr_nm_resultado := 0; --Los datos del usuario no son correctos
0078                    END IF;
0079            ELSE
0080                    --Buscar el usuario por Id
0081                    IF pv_nm_opcion = 3 THEN
0082                            SELECT COUNT(*)
0083                            INTO vr_l_nm_resultado
0084                            FROM tbl_seg_usuario
0085                            WHERE nm_id = pv_nm_id;
0086
0087                            IF (vr_l_nm_resultado = 1) THEN
0088                              OPEN pr_crs_datos FOR
0089                              SELECT Usuario.nm_id As Id,
0090                                      Usuario.nvch2_usuario As Usuario,
0091                                      Usuario.nvch2_nombres As Nombres,
0092                                      Usuario.nvch2_apellidos As Apellidos,
0093                                      Usuario.nvch2_email As EMail
0094                              FROM tbl_seg_usuario Usuario
0095                              WHERE nm_id = pv_nm_id;
0096
0097                              pr_nm_resultado := 1; --El usuario sí existe
0098                            ELSE
0099                              pr_nm_resultado := 0; --No existe el usuario
0100                            END IF;
0101                    ELSE
0102                            --Buscar los datos de todos los usuarios
0103                            IF pv_nm_opcion = 4 THEN
0104                              OPEN pr_crs_datos FOR
0105                              SELECT Usuario.nm_id As ID,
0106                                      Usuario.nvch2_usuario As USUARIO,
0107                                      Usuario.nvch2_nombres As NOMBRES,
0108                                      Usuario.nvch2_apellidos As APELLIDOS,
0109                                      Usuario.nvch2_email As "E-MAIL",
0110                                      CASE
0111                                        WHEN EXISTS (SELECT Resp.*
0112                                          FROM tbl_seg_usuarioPregunta Resp
0113                                          WHERE Resp.nm_id = Usuario.nm_id) THEN 'SÍ'
0114                                        ELSE 'NO'
0115                                      END As "TIENE PREGUNTA SECRETA"
0116                              FROM tbl_seg_usuario Usuario
0117                              ORDER BY ID;
0118
0119                              --Se consultaron los datos de los usuarios
0120                              pr_nm_resultado := 1;
```

Continuación de la figura A-5.

```
0121                                        ELSE
0122                                          pr_nm_resultado := 2; --No es una opción válida
0123                                        END IF;
0124                                    END IF;
0125                                  END IF;
0126                          END IF;
0127                END IF;
0128        EXCEPTION
0129                WHEN OTHERS THEN
0130                        pr_nm_resultado := 3;
0131        END;
```

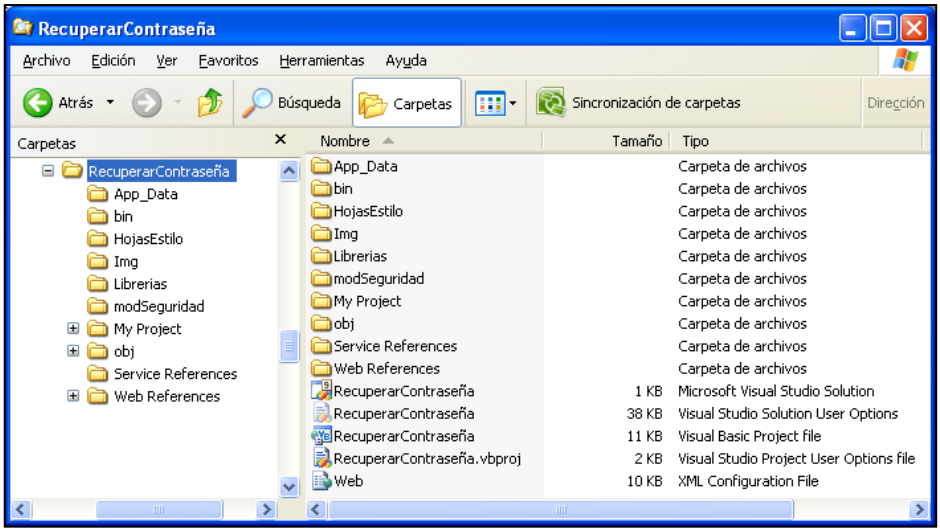
Fuente: elaboración propia.

Todas estas subrutinas se invocan desde la aplicación, cuyo código se describe en el subsiguiente apartado.

1.2. Aplicación

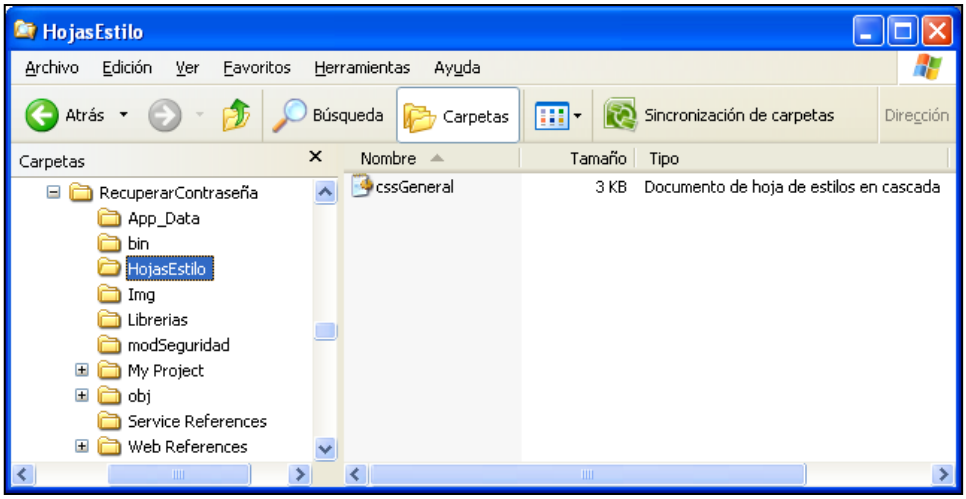
Los archivos que conforman la aplicación están dispuestos como se muestra en las siguientes figuras: una hoja de estilo, imágenes, tres librerías, cuatro formularios, una página HTML y la referencia al servicio *web*.

Figura A-6. **Carpeta del proyecto RecuperarContraseña**



Fuente: elaboración propia.

Figura A-7. **Carpeta del proyecto RecuperarContraseña, que contiene la hoja de estilo utilizada en los formularios**

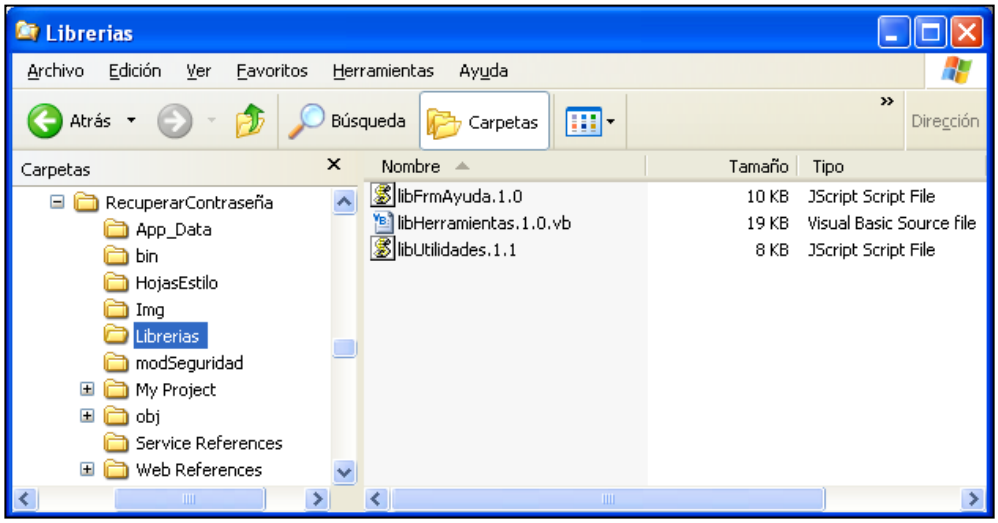


Fuente: elaboración propia.

Figura A-8. **Carpeta del proyecto RecuperarContraseña, que contiene las imágenes utilizadas en los formularios**

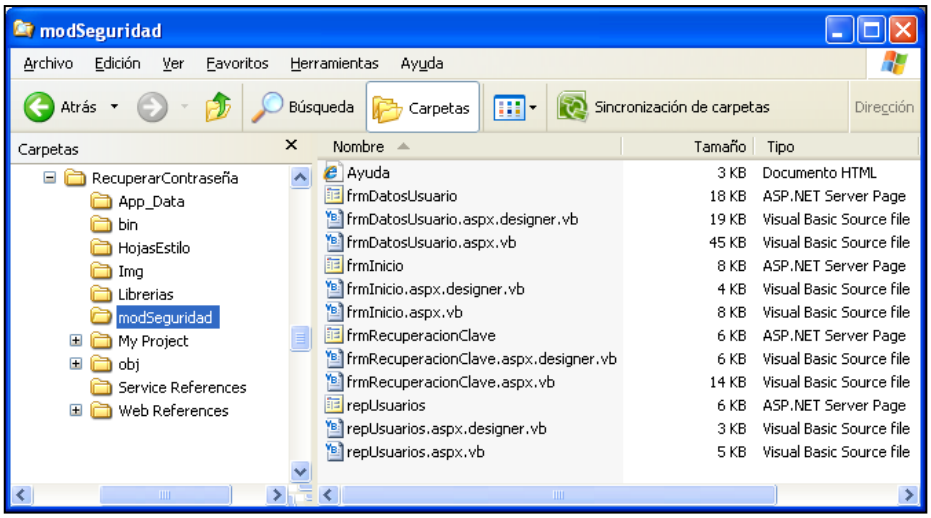
Fuente: elaboración propia.

Figura A-9. **Carpeta del proyecto RecuperarContraseña, que contiene las librerías utilizadas en los formularios**



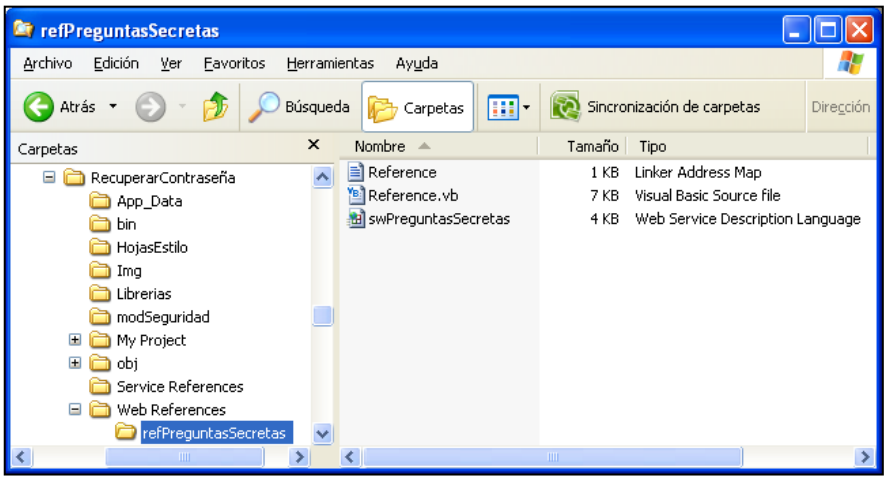
Fuente: elaboración propia.

Figura A-10. **Carpeta del proyecto RecuperarContraseña, que contiene los formularios de la aplicación y la página HTML**



Fuente: elaboración propia.

Figura A-11. **Carpeta del proyecto RecuperarContraseña, que contiene la referencia al servicio *web***



Fuente: elaboración propia.

El código que se describe a continuación, corresponde a las librerías utilizadas en el prototipo.

Figura A-12. **Código de libUtilidades.1.1.js, cuyas subrutinas son invocadas desde los formularios**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                        libUtilidades.1.1.js                                            */
0003  /*   Descripción:                    Librería con utilidades.                                        */
0004  /*   Autor:                          Norma M. Chonay V.                                              */
0005  /*   Fecha de creación:              21/04/2011                                                      */
0006  /*   Fecha de última modificación:   21/04/2011                                                      */
0007  /*   Autor de última modificación:   Norma M. Chonay V.                                              */
0008  /*   Versión:                        1.1                                                             */
0009  /*   Prerrequisitos:                 Ninguno                                                         */
0010  /*****************************************************************************************************/
0011
0012  //Indica si la tecla recibida es una letra minúscula o mayúscula, no incluye vocales tildadas
0013  function fPbBolEsLetra(pRKpTecla)
0014  {   if ((pRKpTecla.keyCode >= 65 && pRKpTecla.keyCode <= 90) || pRKpTecla.keyCode == 209 ||
0015          //Letras mayúsculas
0016          (pRKpTecla.keyCode >= 97 && pRKpTecla.keyCode <= 122) || pRKpTecla.keyCode == 241)
0017          //Letras minúsculas
0018      {   return true;
0019      }
0020      else
0021      {   return false;
0022      }
0023  }
0024  //Indica si la tecla recibida es una vocal tildada minúscula o mayúscula
0025  function fPbBolEsVocalTildada(pRKpTecla)
0026  {   if (pRKpTecla.keyCode == 225 || pRKpTecla.keyCode == 233 || pRKpTecla.keyCode == 237 ||
0027              pRKpTecla.keyCode == 243 || pRKpTecla.keyCode == 250 ||
0028              //Vocales minúsculas tildadas
0029          pRKpTecla.keyCode == 193 || pRKpTecla.keyCode == 201 || pRKpTecla.keyCode == 205 ||
0030              pRKpTecla.keyCode == 211 || pRKpTecla.keyCode == 218)
0031              //Vocales mayúsculas tildadas
0032      {   return true;
0033      }
0034      else
0035      {   return false;
0036      }
0037  }
0038  //Indica si la tecla recibida es un número
0039  function fPbBolEsNumero(pRKpTecla)
0040  {   if (pRKpTecla.keyCode >= 48 && pRKpTecla.keyCode <= 57) //números
0041      {   return true;
0042      }
0043      else
0044      {   return false;
0045      }
```

Continuación de la figura A-12.

```
0046  }
0047  //Restringe el ingreso de números, letras y los carácteres válidos
0048  function pcPbValidarCaracteres(pRKpTecla, pRCtrlEnfoque, pVsIntOpcion)
0049  {   //Variable con el mensaje a desplegar
0050      var vrLStrMensaje = "";
0051
0052      switch (pVsIntOpcion)
0053      {   case 1: //Carácteres alfanuméricos
0054            if ( fPbBolEsLetra(pRKpTecla) || //Letras mayúsculas y minúsculas
0055               fPbBolEsNumero(pRKpTecla) || //números
0056               pRKpTecla.keyCode == 45 || //carácter -
0057               pRKpTecla.keyCode == 95 || //carácter _
0058               pRKpTecla.keyCode == 8) //retroceso (backspace)
0059            {   pVsIntOpcion = 0; //Carácter válido
0060            }
0061            vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS, NÚMEROS " +
0062               "Y LOS CARÁCTERES - _";
0063            break;
0064         case 2: //Carácteres alfabéticos
0065            if (fPbBolEsLetra(pRKpTecla) || //Letras mayúsculas y minúsculas
0066               pRKpTecla.keyCode == 32 || //espacio
0067               fPbBolEsVocalTildada(pRKpTecla) || //Vocales minúsculas y mayúsculas tildadas
0068               pRKpTecla.keyCode == 8) //retroceso (backspace)
0069            {   pVsIntOpcion = 0; //Carácter válido
0070            }
0071            vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS Y " +
0072               "ESPACIOS EN BLANCO";
0073            break;
0074         case 3: //Carácteres para cuenta de correo
0075            if (fPbBolEsLetra(pRKpTecla) || //Letras mayúsculas y minúsculas
0076               fPbBolEsNumero(pRKpTecla) || //números
0077               pRKpTecla.keyCode == 95 || //carácter _
0078               pRKpTecla.keyCode == 46 || //punto
0079               pRKpTecla.keyCode == 64 || //arroba
0080               pRKpTecla.keyCode == 8) //retroceso (backspace)
0081            {   pVsIntOpcion = 0; //Carácter válido
0082            }
0083            vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS, NÚMEROS " +
0084               "Y LOS CARÁCTERES . _ @";
0085            break;
0086         case 4: //Números
0087            if (fPbBolEsNumero(pRKpTecla) || //números
0088               pRKpTecla.keyCode == 8) //retroceso (backspace)
0089            {   pVsIntOpcion = 0; //Carácter válido
0090            }
0091            vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE NÚMEROS";
0092            break;
0093         case 5: //Carácteres alfanuméricos y espacio
0094            if (fPbBolEsLetra(pRKpTecla) || //Letras mayúsculas y minúsculas
0095               fPbBolEsNumero(pRKpTecla) || //números
0096               pRKpTecla.keyCode == 45 || //carácter -
0097               pRKpTecla.keyCode == 95 || //carácter _
0098               pRKpTecla.keyCode == 8 || //retroceso (backspace)
```

Continuación de la figura A-12.

```
0099                pRKpTecla.keyCode == 32 || //espacio
0100                fPbBolEsVocalTildada(pRKpTecla)) //Vocales minúsculas y mayúsculas tildadas
0101             {  pVsIntOpcion = 0; //Carácter válido
0102             }
0103             vrLStrMensaje = "SOLAMENTE SE PERMITE EL INGRESO DE LETRAS, " +
0104                "NÚMEROS, ESPACIOS EN BLANCO Y LOS CARÁCTERES - _";
0105             break;
0106          case 6: //espacio
0107             if (pRKpTecla.keyCode == 32) //espacio
0108             {  pVsIntOpcion = 0; //Carácter válido
0109             }
0110             vrLStrMensaje = "SOLAMENTE SE PERMITE PRESIONAR LA BARRA " +
0111                "ESPACIADORA Y LA TECLA ENTER";
0112             break;
0113          default:
0114             pVsIntOpcion = -1; //Carácter no válido
0115             break;
0116       }
0117       if (pVsIntOpcion != 0) //Si no es un carácter válido
0118       {  //Si se presiona la tecla ENTER
0119          if (pRKpTecla.keyCode == 13)
0120          {  //Si el control tiene el método Focus
0121             if (pRCtrlEnfoque)
0122             {
0123                pRKpTecla.keyCode = 0;
0124                pRCtrlEnfoque.focus(); //se pasa el cursor al control indicado por parámetro
0125             }
0126          }
0127          else
0128          {  //Si la tecla presionada no es válida, se muestra el mensaje correspondiente
0129             alert(vrLStrMensaje);
0130             pRKpTecla.keyCode=0;
0131          }
0132       }
0133    }
0134    //Al dar clik en el ícono de AYUDA
0135    function pcPrAyudaLogin(pVStrNomPadre, pVStrNomOpcion, pVStrNomModalidad)
0136    {  //Llamar a la ventana de ayuda en línea
0137       var vrLStrURL = 'http://localhost/RecuperarContraseña/modSeguridad/Ayuda.htm' +
0138          '?NomPadre=' + pVStrNomPadre + '&NomOpcion=' + pVStrNomOpcion +
0139          '&NomModalidad=' + pVStrNomModalidad
0140       window.open(vrLStrURL,'','menubar=no,toolbar=no,height=329,width=503,resizable=no');
0141    }
0142    //Si se presiona el botón CANCELAR de frmInicio, se limpia el texto ingresado en los controles
0143    function pcPrClicCancelarLogin(pVTxtUsuario, pVTxtPassword)
0144    {
0145       pVTxtUsuario.value = "";
0146       pVTxtPassword.value = "";
0147       pVTxtUsuario.focus();
0148    }
```

Fuente: elaboración propia.

Figura A-13. **Código de libHerramientas.1.0.vb, cuyas subrutinas son invocadas desde los formularios**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                        libHerramientas.1.0.vb                                          */
0003  /*   Descripción:                    Librería con utilidades.                                        */
0004  /*   Autor:                          Norma M. Chonay V.                                              */
0005  /*   Fecha de creación:              24/04/2011                                                      */
0006  /*   Fecha de última modificación:   24/04/2011                                                      */
0007  /*   Autor de última modificación:   Norma M. Chonay V.                                              */
0008  /*   Versión:                        1.0                                                             */
0009  /*   Prerrequisitos:                 Ninguno                                                         */
0010  /*****************************************************************************************************/
0011
0012  Imports Oracle.DataAccess.Client
0013  Imports System.Security.Cryptography
0014  Imports System.Net.Mail
0015  Imports System.Net
0016
0017  Public Class modHerramientas
0018      'Llave para cifrar
0019      Private atPrStrLlave As String = "DESPUÉS DE LA TORMENTA, SIEMPRE LLEGA LA " + _
0020          "C@1MA"
0021
0022      'Función que valida si la cuenta de correo electrónico es válida
0023      Public Function fPbBolValidarEmail(ByVal pVStrEmail As String) As Boolean
0024          Return Regex.IsMatch(pVStrEmail, _
0025                "^(?("")(""".+?""@)|(([0-9a-zA-Z]((\.(?!\.))|[-!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=" + _
0026                "[0-9a-zA-Z])@))" + _
0027                "(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-zA-Z][-\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,6}))$")
0028      End Function
0029
0030      'Función que devuelve la conexión a la BDD
0031      Public Function fPbOcleConectar() As OracleConnection
0032          Dim vrLOcleConexion As New OracleConnection
0033
0034          vrLOcleConexion.ConnectionString = "Data source=nchonay; User id=usr_seg; " + _
0035              "Password=anionuev0;"
0036          vrLOcleConexion.Open()
0037          Return vrLOcleConexion
0038      End Function
0039
0040      'Función para cifrar
0041      Public Function fPbStrCifrar(ByVal pVStrTexto As String) As String
0042          Dim vrLBtLlave As Byte() 'Para almacenar la llave
0043          'Para almacenar el texto a cifrar
0044          Dim vrLBtTexto As Byte() = UTF8Encoding.UTF8.GetBytes(pVStrTexto)
0045          'Para calcular el valor hash MD5 de la llave
0046          Dim vrLMD5Hash As MD5CryptoServiceProvider = New MD5CryptoServiceProvider()
0047          'Para cifrar con el algoritmo triple DES
0048          Dim vrL3DESAlgoritmo As TripleDESCryptoServiceProvider
0049          vrL3DESAlgoritmo = New TripleDESCryptoServiceProvider()
0050
0051          'Almacenar la llave con el valor hash
```

Continuación de la figura A-13.

```
0052        vrLBtLlave = vrLMD5Hash.ComputeHash(UTF8Encoding.UTF8.GetBytes(atPrStrLlave))
0053        vrLMD5Hash.Clear()
0054
0055        'Características para aplicar el cifrado
0056        vrL3DESAlgoritmo.Key = vrLBtLlave
0057        vrL3DESAlgoritmo.Mode = CipherMode.ECB
0058        vrL3DESAlgoritmo.Padding = PaddingMode.PKCS7
0059
0060        'Cifrar
0061        Dim vrLCptoCifrador As ICryptoTransform = vrL3DESAlgoritmo.CreateEncryptor() 'Para cifrar
0062        Dim vrLBtTextoCifrado As Byte() = vrLCptoCifrador.TransformFinalBlock(vrLBtTexto, 0, _
0063                                                    vrLBtTexto.Length)
0064        vrL3DESAlgoritmo.Clear()
0065
0066        Return Convert.ToBase64String(vrLBtTextoCifrado, 0, vrLBtTextoCifrado.Length)
0067    End Function
0068
0069    'Para actualizar los datos del usuario
0070    Public Sub pcPbModificarUsuario(ByVal pVIntId As Int32, ByVal pVStrUsuario As String, _
0071            ByVal pVStrClave As String, ByVal pVStrNombres As String, _
0072            ByVal pVStrApellidos As String, ByVal pVStrEmail As String, _
0073            ByVal pVStrRespuesta1 As String, ByVal pVStrRespuesta2 As String, _
0074            ByVal pVStrRespuesta3 As String, ByVal pVStrRespuesta4 As String, _
0075            ByVal pVStrRespuesta5 As String, ByVal pVsIntOpcion As Int16, _
0076            ByRef pROcleConexion As OracleConnection, ByRef pRsIntResultado As Int16, _
0077            ByRef pRHdnMensaje As HiddenField)
0078        'Inicializar el código del mensaje resultado
0079        pRsIntResultado = 10
0080
0081        Try
0082            'Validar si se pudo establecer la conexión con Oracle
0083            If Not (pROcleConexion Is Nothing) Then
0084                Dim vrLOcleComando As OracleCommand = New OracleCommand()
0085
0086                'Indicar el nombre del procedimiento almacenado
0087                vrLOcleComando.CommandType = CommandType.StoredProcedure
0088                vrLOcleComando.CommandText = "pc_modificarusuario"
0089
0090                'Pasar los parámetros
0091                vrLOcleComando.Parameters.Clear()
0092                vrLOcleComando.Parameters.Add("pv_nm_id", OracleDbType.Int32, pVIntId, _
0093                                    ParameterDirection.Input)
0094                vrLOcleComando.Parameters.Add("pv_nvch2_usuario", OracleDbType.NVarchar2, _
0095                                    10, pVStrUsuario, ParameterDirection.Input)
0096                vrLOcleComando.Parameters.Add("pv_nvch2_clave", OracleDbType.NVarchar2, _
0097                                    1000, pVStrClave, ParameterDirection.Input)
0098                vrLOcleComando.Parameters.Add("pv_nvch2_nombres", OracleDbType.NVarchar2, _
0099                                    50, pVStrNombres, ParameterDirection.Input)
0100                vrLOcleComando.Parameters.Add("pv_nvch2_apellidos", OracleDbType.NVarchar2, _
0101                                    50, pVStrApellidos, ParameterDirection.Input)
0102                vrLOcleComando.Parameters.Add("pv_nvch2_email", OracleDbType.NVarchar2, 50, _
0103                                    pVStrEmail, ParameterDirection.Input)
0104                vrLOcleComando.Parameters.Add("pv_nvch2_respuesta1", _
```

Continuación de la figura A-13.

```
0105                                    OracleDbType.NVarchar2, 100, pVStrRespuesta1, _
0106                                    ParameterDirection.Input)
0107            vrLOcleComando.Parameters.Add("pv_nvch2_respuesta2", _
0108                                    OracleDbType.NVarchar2, 100, pVStrRespuesta2, _
0109                                    ParameterDirection.Input)
0110            vrLOcleComando.Parameters.Add("pv_nvch2_respuesta3", _
0111                                    OracleDbType.NVarchar2, 100, pVStrRespuesta3, _
0112                                    ParameterDirection.Input)
0113            vrLOcleComando.Parameters.Add("pv_nvch2_respuesta4", _
0114                                    OracleDbType.NVarchar2, 100, pVStrRespuesta4, _
0115                                    ParameterDirection.Input)
0116            vrLOcleComando.Parameters.Add("pv_nvch2_respuesta5", _
0117                                    OracleDbType.NVarchar2, 100, pVStrRespuesta5, _
0118                                    ParameterDirection.Input)
0119            vrLOcleComando.Parameters.Add("pv_nm_opcion", OracleDbType.Int16, _
0120                                    pVsIntOpcion, ParameterDirection.Input)
0121            vrLOcleComando.Parameters.Add("pr_nm_resultado", OracleDbType.Int16, _
0122                                    ParameterDirection.Output)
0123
0124            'Ejecutar la consulta
0125            vrLOcleComando.Connection = pROcleConexion
0126            vrLOcleComando.ExecuteNonQuery()
0127            pRsIntResultado = Convert.ToInt16(vrLOcleComando.Parameters("pr_nm_" + _
0128                                                    "resultado").Value)
0129        Else
0130            pRHdnMensaje.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0131                "DE DATOS"
0132        End If
0133    Catch pVExExcepcion As Exception
0134        pRsIntResultado = 10
0135        pRHdnMensaje.Value = pVExExcepcion.ToString()
0136    End Try
0137End Sub
0138
0139'Para crear un nuevo usuario
0140Public Sub pcPbGuardarUsuario(ByRef pRIntId As Int32, ByVal pVStrUsuario As String, _
0141        ByVal pVStrClave As String, ByVal pVStrNombres As String, _
0142        ByVal pVStrApellidos As String, ByVal pVStrEmail As String, _
0143        ByRef pROcleConexion As OracleConnection, ByRef pRsIntResultado As Int16, _
0144        ByRef pRHdnMensaje As HiddenField)
0145    'Inicializar el código del mensaje resultado
0146    pRsIntResultado = 10
0147
0148    Try
0149        'Validar si se pudo establecer la conexión con Oracle
0150        If Not (pROcleConexion Is Nothing) Then
0151            Dim vrLOcleComando As OracleCommand = New OracleCommand()
0152
0153            'Indicar el nombre del procedimiento almacenado
0154            vrLOcleComando.CommandType = CommandType.StoredProcedure
0155            vrLOcleComando.CommandText = "pc_guardarusuario"
0156
0157            'Pasar los parámetros
```

Continuación de la figura A-13.

```
0158            vrLOcleComando.Parameters.Clear()
0159            vrLOcleComando.Parameters.Add("pr_nm_id", OracleDbType.Int32, _
0160                                ParameterDirection.Output)
0161            vrLOcleComando.Parameters.Add("pv_nvch2_usuario", OracleDbType.NVarchar2, _
0162                                10, pVStrUsuario, ParameterDirection.Input)
0163            vrLOcleComando.Parameters.Add("pv_nvch2_clave", OracleDbType.NVarchar2, _
0164                                1000, pVStrClave, ParameterDirection.Input)
0165            vrLOcleComando.Parameters.Add("pv_nvch2_nombres", OracleDbType.NVarchar2, _
0166                                50, pVStrNombres, ParameterDirection.Input)
0167            vrLOcleComando.Parameters.Add("pv_nvch2_apellidos", OracleDbType.NVarchar2, _
0168                                50, pVStrApellidos, ParameterDirection.Input)
0169            vrLOcleComando.Parameters.Add("pv_nvch2_email", OracleDbType.NVarchar2, _
0170                                50, pVStrEmail, ParameterDirection.Input)
0171            vrLOcleComando.Parameters.Add("pr_nm_resultado", OracleDbType.Int16, _
0172                                ParameterDirection.Output)
0173
0174            'Ejecutar la consulta
0175            vrLOcleComando.Connection = pROcleConexion
0176            vrLOcleComando.ExecuteNonQuery()
0177            pRsIntResultado = Convert.ToInt16(vrLOcleComando.Parameters("pr_nm_" + _
0178                                                "resultado").Value)
0179            pRIntId = Convert.ToInt16(vrLOcleComando.Parameters("pr_nm_id").Value)
0180          Else
0181            pRHdnMensaje.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0182              "DE DATOS"
0183          End If
0184        Catch pVExExcepcion As Exception
0185          pRsIntResultado = 10
0186          pRHdnMensaje.Value = pVExExcepcion.ToString()
0187        End Try
0188      End Sub
0189
0190      'Procedimiento para eliminar un usuario
0191      Public Sub pcPbEliminarUsuario(ByVal pVIntId As Int32, _
0192            ByRef pROcleConexion As OracleConnection, ByRef pRsIntResultado As Int16, _
0193            ByRef pRHdnMensaje As HiddenField)
0194        'Inicializar el código del mensaje resultado
0195        pRsIntResultado = 10
0196
0197        Try
0198          'Validar si se pudo establecer la conexión con Oracle
0199          If Not (pROcleConexion Is Nothing) Then
0200            Dim vrLOcleComando As OracleCommand = New OracleCommand()
0201
0202            'Indicar el nombre del procedimiento almacenado
0203            vrLOcleComando.CommandType = CommandType.StoredProcedure
0204            vrLOcleComando.CommandText = "pc_eliminarusuario"
0205
0206            'Pasar los parámetros
0207            vrLOcleComando.Parameters.Clear()
0208            vrLOcleComando.Parameters.Add("pv_nm_id", OracleDbType.Int32, pVIntId, _
0209                                ParameterDirection.Input)
0210            vrLOcleComando.Parameters.Add("pr_nm_resultado", OracleDbType.Int16, _
```

Continuación de la figura A-13.

```
0211                                    ParameterDirection.Output)
0212
0213            'Ejecutar la consulta
0214            vrLOcleComando.Connection = pROcleConexion
0215            vrLOcleComando.ExecuteNonQuery()
0216            pRsIntResultado = Convert.ToInt16(vrLOcleComando.Parameters("pr_nm_" + _
0217                                                    "resultado").Value)
0218
0219         Else
0220            pRHdnMensaje.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0221               "DE DATOS"
0222         End If
0223
0224      Catch pVExExcepcion As Exception
0225         pRsIntResultado = 10
0226         pRHdnMensaje.Value = pVExExcepcion.ToString()
0227      End Try
0228   End Sub
0229
0230   'Función para validar si el usuario y la contraseña son correctos,
0231   'si el usuario y el email son correctos,
0232   'buscar datos de todos los usuarios, buscar un usuario por ID,
0233   'o bien validar si el usuario ya existe
0234   Public Function fPbTblBuscarUsuario(ByVal pVIntId As Int32, ByVal pVStrUsuario As String, _
0235         ByVal pVStrClave As String, ByVal pVStrEmail As String, ByVal pVsIntOpcion As Int16, _
0236         ByRef pROcleConexion As OracleConnection, ByRef pRsIntResultado As Int16, _
0237         ByRef pRHdnMensaje As HiddenField) As DataTable
0238      'Valor de retorno
0239      Dim rtTblDatos As DataTable = New DataTable()
0240      'Inicializar el código del mensaje resultado
0241      pRsIntResultado = 10
0242
0243      Try
0244         'Validar si se pudo establecer la conexión con Oracle
0245         If Not (pROcleConexion Is Nothing) Then
0246            Dim vrLOcleComando As OracleCommand = New OracleCommand()
0247            Dim vrLOcleAdaptador As OracleDataAdapter
0248
0249            'Indicar el nombre del procedimiento almacenado
0250            vrLOcleComando.CommandType = CommandType.StoredProcedure
0251            vrLOcleComando.CommandText = "pc_buscarusuario"
0252
0253            'Pasar los parámetros
0254            vrLOcleComando.Parameters.Clear()
0255            vrLOcleComando.Parameters.Add("pv_nm_id", OracleDbType.Int32, pVIntId, _
0256                                    ParameterDirection.Input)
0257            vrLOcleComando.Parameters.Add("pv_nvch2_usuario", OracleDbType.NVarchar2, _
0258                                    10, pVStrUsuario, ParameterDirection.Input)
0259            vrLOcleComando.Parameters.Add("pv_nvch2_clave", OracleDbType.NVarchar2, _
0260                                    1000, pVStrClave, ParameterDirection.Input)
0261            vrLOcleComando.Parameters.Add("pv_nvch2_email", OracleDbType.NVarchar2, _
0262                                    50, pVStrEmail, ParameterDirection.Input)
0263            vrLOcleComando.Parameters.Add("pv_nm_opcion", OracleDbType.Int16, _
```

Continuación de la figura A-13.

```
0264                                    pVsIntOpcion, ParameterDirection.Input)
0265            vrLOcleComando.Parameters.Add("pr_nm_resultado", OracleDbType.Int16, _
0266                                    ParameterDirection.Output)
0267            vrLOcleComando.Parameters.Add("pr_crs_datos", OracleDbType.RefCursor, _
0268                                    ParameterDirection.Output)
0269
0270            'Ejecutar la consulta
0271            vrLOcleComando.Connection = pROcleConexion
0272            vrLOcleAdaptador = New OracleDataAdapter(vrLOcleComando)
0273            vrLOcleAdaptador.Fill(rtTblDatos)
0274            pRsIntResultado = Convert.ToInt16(vrLOcleComando.Parameters("pr_nm_" + _
0275                                                    "resultado").Value)
0276
0277         Else
0278            pRHdnMensaje.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0279               "DE DATOS"
0280         End If
0281
0282      Catch pVExExcepcion As Exception
0283         pRsIntResultado = 10
0284         pRHdnMensaje.Value = pVExExcepcion.ToString()
0285      End Try
0286      Return rtTblDatos
0287   End Function
0288
0289   'Función para buscar las preguntas
0290   Public Function fPbTblBuscarPreguntas(ByRef pRsIntResultado As Int16, _
0291         ByRef pRHdnMensaje As HiddenField) As DataTable
0292
0293      Dim vrLTblsDatos As DataSet = New DataSet()
0294      'Valor de retorno
0295      Dim rtTblDatos As DataTable = New DataTable()
0296      'Inicializar el código del mensaje resultado
0297      pRsIntResultado = 10
0298
0299      Try
0300         'Invocar al servicio web que obtiene las preguntas secretas de la BDD de Oracle
0301         Dim vrLSwPreguntasSecretas As New refPreguntasSecretas.swPreguntasSecretas
0302         vrLTblsDatos = vrLSwPreguntasSecretas.wSfPbTblBuscarPreguntas()
0303
0304         'Si hay datos
0305         If Not (vrLTblsDatos Is Nothing) Then
0306            If vrLTblsDatos.Tables.Count = 1 Then
0307               'Obtener el código y cadena del mensaje
0308               rtTblDatos = vrLTblsDatos.Tables(0)
0309               pRsIntResultado = rtTblDatos.Rows(0)("ID")
0310               pRHdnMensaje.Value = rtTblDatos.Rows(0)("PREGUNTA")
0311               'Obtener las preguntas
0312               rtTblDatos.Rows.RemoveAt(0)
0313            Else
0314               pRsIntResultado = 3
0315            End If
0316         Else
```

Continuación de la figura A-13.

```
0317                pRsIntResultado = 3
0318            End If
0319
0320        Catch pVExExcepcion As Exception
0321            pRsIntResultado = 10
0322            pRHdnMensaje.Value = pVExExcepcion.ToString()
0323        End Try
0324        Return rtTblDatos
0325    End Function
0326
0327    'Enviar correo electrónico
0328    Public Function fPbBolEnviarEmail(ByVal pVStrEmail As String, ByVal pVStrUsuario As String, _
0329            ByVal pVStrClave As String, ByVal pVsIntOpcion As Int16, _
0330            ByRef pRHdnMensaje As HiddenField) As Boolean
0331        'Configuración del correo
0332        Dim vrLAdrsRemitente As MailAddress = New MailAddress("notificacion.sistema.gt" + _
0333                                                "@gmail.com")
0334        Dim vrLAdrsDestinatario As MailAddress = New MailAddress(pVStrEmail)
0335        Dim vrLMsgMensaje As MailMessage = New MailMessage(vrLAdrsRemitente, _
0336                                                vrLAdrsDestinatario)
0337        vrLMsgMensaje.Subject = "SISTEMA.GT"
0338        vrLMsgMensaje.IsBodyHtml = True
0339
0340        Select Case pVsIntOpcion
0341            'Mensaje de bienvenida
0342            Case 1
0343                vrLMsgMensaje.Body = "<b>BIENVENIDO</b> al SISTEMA.GT <br><br>" + _
0344                    "Su usuario es: " + pVStrUsuario + "<br>" + _
0345                    "La contraseña es: " + pVStrClave + "<br><br><br>" + _
0346                    "Debe ingresar al sistema para elegir sus preguntas secretas, " + _
0347                    "que servirán para recuperar la contraseña al momento de olvidarla."
0348            Case 2
0349                'Mensaje cambio de contraseña
0350                vrLMsgMensaje.Body = "Sus credenciales para ingresar al SISTEMA.GT han " + _
0351                    "cambiado. <br><br>" + "Su usuario es: <b>" + pVStrUsuario + "</b><br>" + _
0352                    "La contraseña es: <b>" + pVStrClave + "</b>"
0353            Case Else
0354                vrLMsgMensaje.Body = ""
0355        End Select
0356
0357        Dim vrLSmtpCorreoSaliente As SmtpClient = New SmtpClient("smtp.gmail.com")
0358        vrLSmtpCorreoSaliente.Port = 587
0359        vrLSmtpCorreoSaliente.EnableSsl = True
0360        vrLSmtpCorreoSaliente.UseDefaultCredentials = False
0361        vrLSmtpCorreoSaliente.Credentials = New NetworkCredential("notificacion.sistema.gt" + _
0362                                                "@gmail.com", "notific@c1on")
0363
0364        Try
0365            'Enviar el mensaje
0366            vrLSmtpCorreoSaliente.Send(vrLMsgMensaje)
0367        Catch pVSmtpExExcepcion As SmtpFailedRecipientException
0368            pRHdnMensaje.Value = pVSmtpExExcepcion.Message
0369            Return False
```

Continuación de la figura A-13.

```
0370        Catch pVExExcepcion As Exception
0371            pRHdnMensaje.Value = pVExExcepcion.Message
0372            Return False
0373        End Try
0374        Return True
0375    End Function
0376
0377    'Generar clave aleatoria
0378    Public Function fPbVGenerarClave(ByVal pVStrUsuario As String) As Object
0379        Dim vrLvClave(1) As Object
0380        Dim vrLStrTemp As String = ""
0381        vrLvClave(0) = ""
0382        vrLvClave(1) = ""
0383
0384        'Generar la clave
0385        'Tomando la hora en formato completo
0386        Dim vrLBtClave As Byte() = UTF8Encoding.UTF8.GetBytes(DateTime.Now.ToString("T"))
0387        For Each vrLBtByte As Byte In vrLBtClave
0388            vrLvClave(0) += vrLBtByte.ToString()
0389        Next
0390        'Tomando el nombre del usuario
0391        vrLBtClave = UTF8Encoding.UTF8.GetBytes(pVStrUsuario + pVStrUsuario + pVStrUsuario)
0392        For Each vrLBtByte As Byte In vrLBtClave
0393            vrLStrTemp += vrLBtByte.ToString()
0394        Next
0395        'Tomando 5 dígitos de la suma de 3 bytes del usuario y 6 bytes de la hora
0396        vrLvClave(0) = Convert.ToString(Convert.ToInt64(vrLvClave(0).Substring(8, 13)) + _
0397                Convert.ToInt32(vrLStrTemp.Substring(0, 2))).Substring(1, 6)
0398        'Cifrar la clave
0399        vrLvClave(1) = fPbStrCifrar(vrLvClave(0))
0400
0401        Return vrLvClave
0402    End Function
0403 End Class
```

Fuente: elaboración propia.

Figura A-14. **Código de libFrmAyuda.1.0.js, cuyas subrutinas son invocadas desde la página HTML**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                        libFrmAyuda.1.0.js                                          */
0003  /*   Descripción:                    Librería con utilidades para la forma Ayuda.                */
0004  /*   Autor:                          Norma M. Chonay V.                                          */
0005  /*   Fecha de creación:              24/04/2011                                                  */
0006  /*   Fecha de última modificación:   24/04/2011                                                  */
0007  /*   Autor de última modificación:   Norma M. Chonay V.                                          */
0008  /*   Versión:                        1.0                                                         */
0009  /*   Prerrequisitos:                 Ninguno                                                     */
0010  /*****************************************************************************************************/
0011
0012  //Procedimiento que se ejecuta al cargar la página
0013  function pcPbLoad()
0014  {   frmAyuda.btnCerrar.focus();
0015
0016      pcPbEscribirDescripcion();
0017  }
0018
0019  //Procedimiento que cierra la página
0020  function pcPbCerrar()
0021  {   window.close();
0022  }
0023
0024  //Dependiendo de la página que lo invoque se muestra la ayuda
0025  function pcPbEscribirDescripcion()
0026  {   //obtener la parte query del URL ?NomPadre=pVStrNomPadre
0027      //&NomOpcion=pVStrNomOpcion&NomModalidad=pVStrNomModalidad
0028      var vrLStrQueryURL = window.location.search.substr(1);
0029      var vrLStrNomPadre = '';
0030      var vrLStrNomOpcion = '';
0031      var vrLStrNomModalidad = '';
0032      var vrLVQuery;
0033      var vrLStrOpcion = '';
0034
0035      if (vrLStrQueryURL != '')
0036      {   vrLVQuery = vrLStrQueryURL.split('&', 3);
0037          for (var i = 0; i < vrLVQuery.length; i++)
0038          {   if (vrLVQuery[i] != '')
0039              {   vrLStrOpcion = vrLVQuery[i].split('=', 2);
0040                  switch (vrLStrOpcion[0])
0041                  {   case 'NomPadre':
0042                          vrLStrNomPadre = vrLStrOpcion[1];
0043                          break;
0044                      case 'NomOpcion':
0045                          vrLStrNomOpcion = vrLStrOpcion[1];
0046                          break;
0047                      case 'NomModalidad':
0048                          vrLStrNomModalidad = vrLStrOpcion[1];
0049                          break;
0050                  }
0051              }
```

Continuación de la figura A-14.

```
0052        }
0053    }
0054
0055    //Personalizar la ayuda
0056    switch (vrLStrNomPadre)
0057    {   case "frmInicio":
0058            document.title = "Ayuda Login";
0059            frmAyuda.txtBxDescripcion.value = "Formulario de ingreso: \r\n" +
0060                    "--> Ingrese su usuario y contraseña.  No deben contener más de 10 " +
0061                        "carácteres; solamente se aceptan números, letras y los carácteres - " +
0062                        "_ \r\n\r\n" +
0063                    "--> En la contraseña se hace distinción entre mayúsculas y minúsculas. " +
0064                        "\r\n\r\n" +
0065                    "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda del " +
0066                        "botón.";
0067            break;
0068        case "frmDatosUsuario":
0069            document.title = "Ayuda Datos Usuario";
0070            switch (vrLStrNomOpcion)
0071            {   case "AGREGANDO ...":
0072                    frmAyuda.txtBxDescripcion.value = "Agregar usuarios: \r\n" +
0073                            "--> El usuario no debe contener más de 10 carácteres; solamente se " +
0074                                "aceptan números, letras y los carácteres - _ \r\n\r\n" +
0075                            "--> Para los nombres y apellidos, solamente se aceptan letras y espacios " +
0076                                "en blanco. \r\n\r\n" +
0077                            "--> Para el E-mail, se aceptan números, letras y los carácteres @ . _ " +
0078                                "\r\n\r\n" +
0079                            "--> Las credenciales de acceso se envían al correo indicado.\r\n\r\n" +
0080                            "--> Los botones tienen acceso directo: Alt + la primera letra de la " +
0081                                "leyenda del botón.";
0082                    break;
0083                case "EDITANDO ...":
0084                    frmAyuda.txtBxDescripcion.value = "Formulario de datos: \r\n" +
0085                            "--> El usuario y contraseña, no deben contener más de 10 carácteres; " +
0086                                "solamente se aceptan números, letras y los carácteres - _ \r\n\r\n" +
0087                            "--> En la contraseña se hace distinción entre mayúsculas y minúsculas. " +
0088                                "\r\n\r\n" +
0089                            "--> Para los nombres y apellidos, solamente se aceptan letras y espacios " +
0090                                "en blanco. \r\n\r\n" +
0091                            "--> Para el E-mail, se aceptan números, letras y los carácteres @ . _ " +
0092                                "\r\n\r\n" +
0093                            "--> Debe seleccionar al menos una pregunta secreta. \r\n\r\n" +
0094                            "--> Cada respuesta no debe contener más de 100 carácteres; solamente " +
0095                                "se aceptan números, letras, espacios en blanco y los carácteres - _ " +
0096                                "\r\n\r\n" +
0097                            "--> Si no desea modificar la contraseña, deje en blanco los campos " +
0098                                "<<Contraseña anterior>>, <<Nueva contraseña>> y <<Confirmar " +
0099                                "contraseña>>. \r\n\r\n" +
0100                            "--> Los botones tienen acceso directo: Alt + la primera letra de la " +
0101                                "leyenda del botón.";
0102                    break;
0103                case "ELIMINANDO ...":
0104                    frmAyuda.txtBxDescripcion.value = "Eliminar usuarios: \r\n" +
```

Continuación de la figura A-14.

```
0105                        "--> Primero debe ingresar el ID del usuario, luego presionar el botón " +
0106                            "BUSCAR, para visualizar los datos del usuario a eliminar. \r\n\r\n" +
0107                        "--> Después de verificar que el usuario es el que se desea eliminar, " +
0108                            "presionar el botón ACEPTAR; se pedirá una confirmación antes de " +
0109                            "eliminar el usuario permanentemente. \r\n\r\n" +
0110                        "--> Los botones tienen acceso directo: Alt + la primera letra de la " +
0111                            "leyenda del botón.";
0112                 break;
0113              case "REINICIANDO ...":
0114                 frmAyuda.txtBxDescripcion.value = "Reinicio de contraseñas de usuarios: \r\n" +
0115                        "--> Primero debe ingresar el ID del usuario, luego presionar el botón " +
0116                            "BUSCAR, para visualizar los datos del usuario a quien se le " +
0117                            "reiniciará la contraseña. \r\n\r\n" +
0118                        "--> Después de verificar que el usuario es el que busca, presionar el " +
0119                            "botón ACEPTAR; se pedirá una confirmación antes de cambiar la " +
0120                            "contraseña. \r\n\r\n" +
0121                        "--> Las credenciales de acceso se envían al correo indicado.\r\n\r\n" +
0122                        "--> Los botones tienen acceso directo: Alt + la primera letra de la " +
0123                            "leyenda del botón.";
0124                 break;
0125              case "":
0126                 frmAyuda.txtBxDescripcion.value = "Datos de usuario: \r\n";
0127                 //Si es el usuario administrador
0128                 if (vrLStrNomModalidad == "ADMIN")
0129                 {   frmAyuda.txtBxDescripcion.value = frmAyuda.txtBxDescripcion.value +
0130                          "--> Para reiniciarle contraseña a un usuario, presione el ícono de " +
0131                              "CONTRASEÑA. \r\n\r\n" +
0132                          "--> Para consultar datos de usuarios, presione el ícono de INFORME. " +
0133                              "\r\n\r\n" +
0134                          "--> Para eliminar usuarios, presione el ícono de ELIMINAR. \r\n\r\n" +
0135                          "--> Para agregar usuarios, presione el ícono de AGREGAR. \r\n\r\n";
0136                 }
0137                 frmAyuda.txtBxDescripcion.value = frmAyuda.txtBxDescripcion.value +
0138                           "--> Para modificar sus propios datos, presione el ícono de EDITAR.";
0139                 break;
0140              default:
0141                 frmAyuda.txtBxDescripcion.value = "";
0142                 break;
0143           }
0144           break;
0145        case "frmRecuperacionClave":
0146           document.title = "Ayuda Recuperación de Clave";
0147           frmAyuda.txtBxDescripcion.value = "Recuperación de clave: \r\n" +
0148                    "--> El usuario y el E-mail deben coincidir con los registrados en el sistema.  " +
0149                        "La nueva contraseña se enviará a la dirección de correo electrónica " +
0150                        "registrada. \r\n\r\n" +
0151                    "--> El usuario no debe contener más de 10 carácteres; solamente se aceptan " +
0152                        "números, letras y los carácteres - _ \r\n\r\n" +
0153                    "--> Para el E-mail, se aceptan números, letras y los carácteres @ . _ \r\n\r\n" +
0154                    "--> Primero debe validar los datos, y luego se mostrará alguna de las " +
0155                        "preguntas secretas que haya registrado, para que indique la respuesta. " +
0156                        "\r\n\r\n" +
0157                    "--> Los botones tienen acceso directo: Alt + la primera letra de la leyenda " +
```

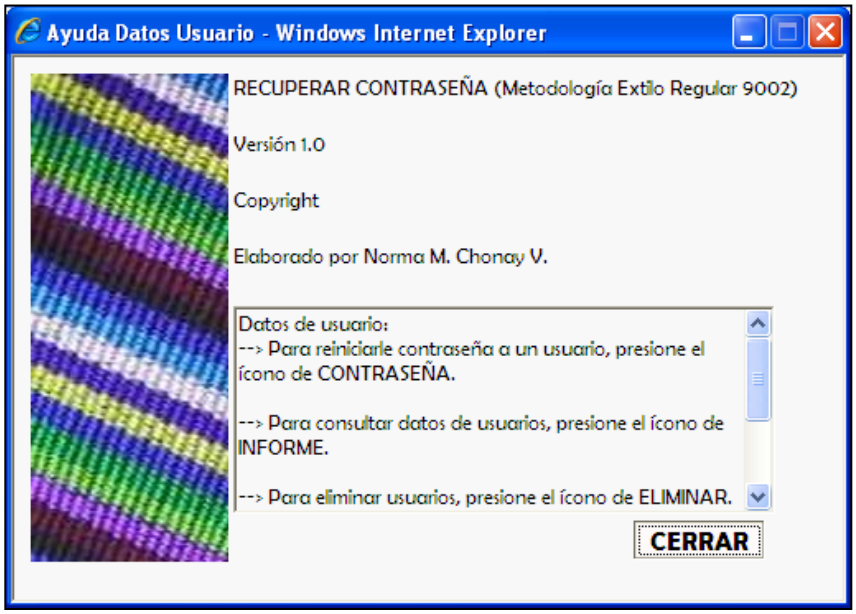
Continuación de la figura A-14.

```
0158                            "del botón.";
0159            break;
0160        default:
0161            document.title = 'Ayuda';
0162            frmAyuda.txtBxDescripcion.value = '';
0163            break;
0164    }
0165 }
```

Fuente: elaboración propia.

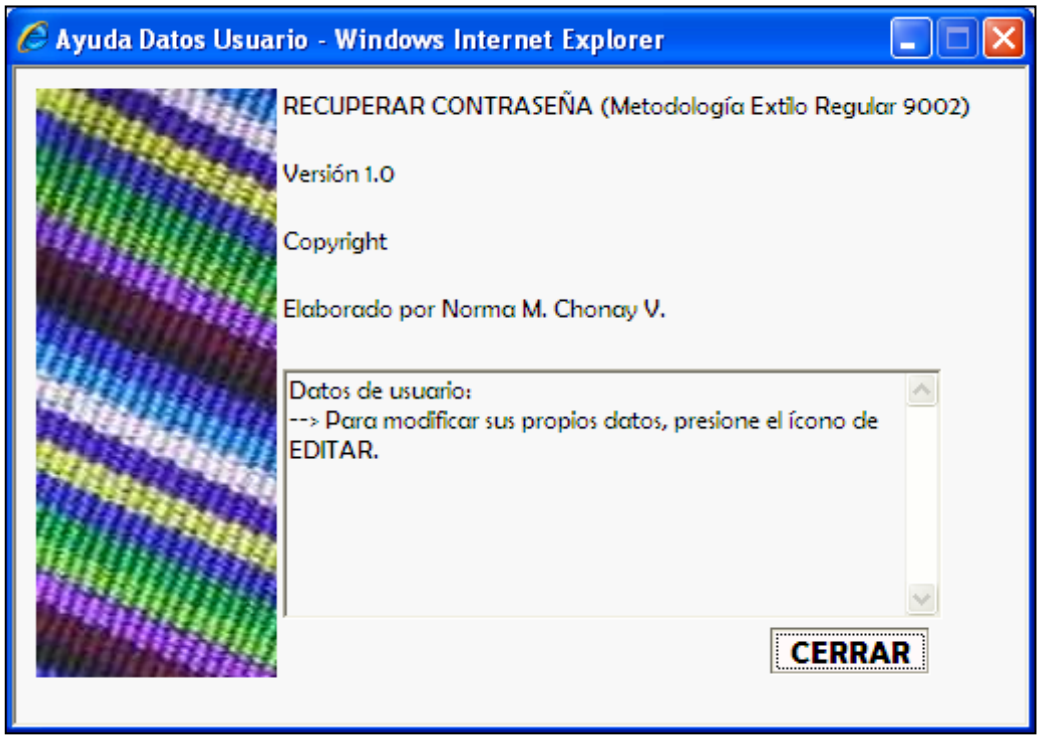
La ayuda en línea está implementada como una página HTML que muestra determinado texto, con base a los valores indicados por cada formulario que la invoca, en la dirección URL. El código de ésta página es el que se indica en la figura A-17.

Figura A-15. **Ayuda en línea, cuando inicia sesión el usuario ADMIN**



Fuente: elaboración propia.

Figura A-16. **Ayuda en línea, cuando inicia sesión cualquier usuario que no sea ADMIN**



Fuente: elaboración propia.

Figura A-17. **Código de la página Ayuda.htm**

```
0001  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
0002     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
0003  <!--
0004  /*****************************************************************************************************/
0005  /*    Archivo:                        Ayuda.htm                                                      */
0006  /*    Descripción:                    Forma que muestra ayuda en línea.                              */
0007  /*    Autor:                          Norma M. Chonay V.                                             */
0008  /*    Fecha de creación:              24/04/2011                                                     */
0009  /*    Fecha de última modificación:   24/04/2011                                                     */
0010  /*    Autor de última modificación:   Norma M. Chonay V.                                             */
0011  /*    Versión:                        1.1                                                            */
0012  /*    Prerrequisitos:                 Ninguno                                                        */
0013  /*****************************************************************************************************/
0014  -->
```

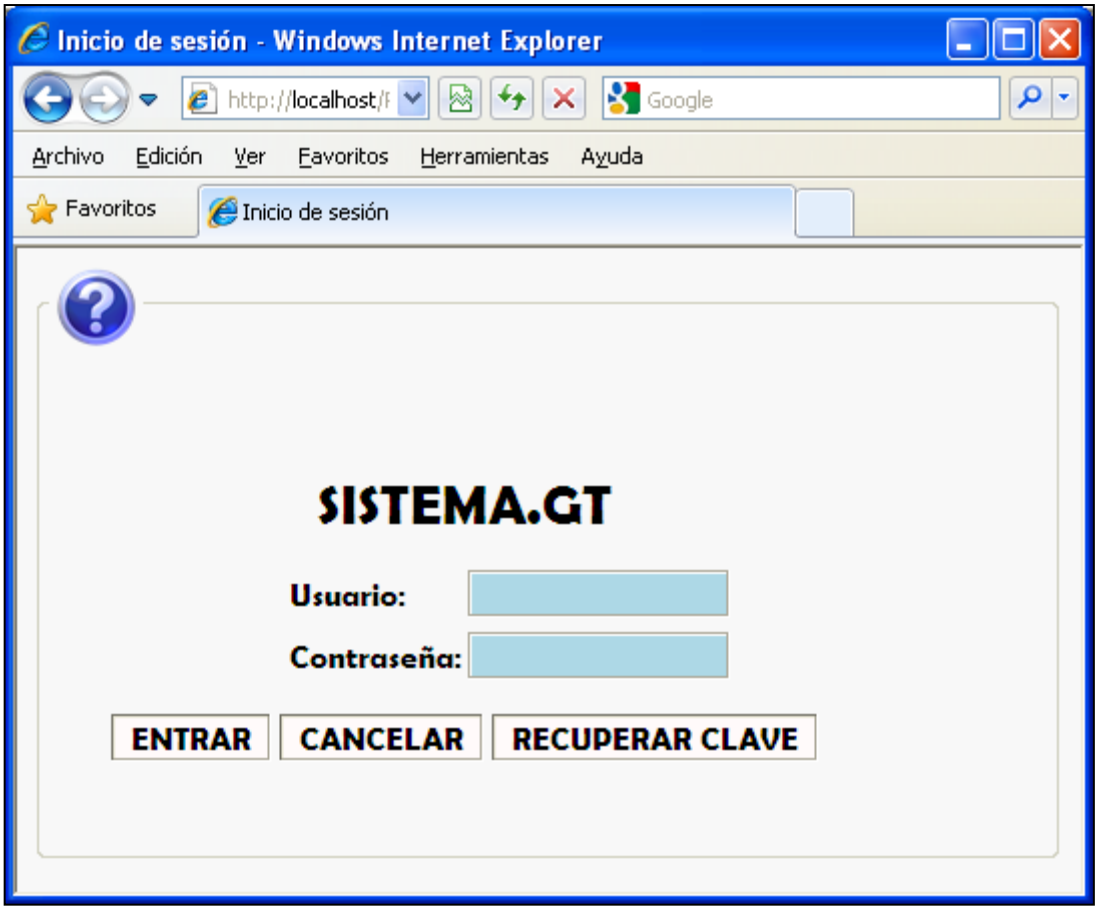
Continuación de la figura A-17.

```
0015
0016  <html xmlns="http://www.w3.org/1999/xhtml" >
0017  <head>
0018     <title></title>
0019     <link href="../HojasEstilo/cssGeneral.css" rel="stylesheet" type="text/css" />
0020     <script type ="text/javascript" language="jscript" src="../Librerias/libFrmAyuda.1.0.js">
0021             </script>
0022  </head>
0023  <body onload="pcPbLoad();">
0024     <form id="frmAyuda" action="">
0025        <table style="width:100%">
0026           <tr>
0027              <td rowspan="3" class="clsImagen"></td>
0028              <td align="justify" style="width:75%; height:20%">
0029                 <div class="clsTextoAyuda">RECUPERAR CONTRASEÑA (Metodología Extilo
0030                    Regular 9002)</div>
0031                 <br />
0032                 <div class="clsTextoAyuda">Versión 1.0</div>
0033                 <br />
0034                 <div class="clsTextoAyuda">Copyright</div>
0035                 <br />
0036                 <div class="clsTextoAyuda">Elaborado por Norma M. Chonay V.</div>
0037                 <br />
0038              </td>
0039           </tr>
0040           <tr>
0041              <td colspan="2" style="width:75%; height:70%">
0042                 <textarea id="txtADescripcion" cols="20" name="txtBxDescripcion" rows="8"
0043                    style="width:90%; height:90%; background-color:#F8F8F8" class="clsTextoAyuda">
0044                 </textarea></td>
0045           </tr>
0046           <tr>
0047              <td colspan="2" style="width:75%; height:10%" align="right">
0048                 <input id="btnCerrar" type="button" value="CERRAR" accesskey="C" class="clsBoton"
0049                    onclick="pcPbCerrar();"/>
0050                             
0051              </td>
0052           </tr>
0053        </table>
0054     </form>
0055  </body>
0056  </html>
```

Fuente: elaboración propia.

La figura A-18 muestra la ventana principal de la aplicación, que es donde se indica el usuario y contraseña para ingresar al sistema, también desde este formulario se ingresa a la opción para recuperar la clave.

Figura A-18. **Diseño del formulario de inicio de sesión**



Fuente: elaboración propia.

El código de este formulario, consta del código de la página aspx (figura A-19), y del código en Visual Basic (figura A-20):

Figura A-19. **Código de frmInicio.aspx**

```
0001  <%@ Page Language="vb" AutoEventWireup="false" CodeBehind="frmInicio.aspx.vb"
0002      Inherits="RecuperarContraseña.frmInicio" %>
0003  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
0004      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
0005  <!--
0006  /*****************************************************************************************************/
0007  /*    Archivo:                          fmrInicio.aspx                                           */
0008  /*    Descripción:                      Forma para ingresar credenciales de acceso.              */
0009  /*    Autor:                            Norma M. Chonay V.                                       */
0010  /*    Fecha de creación:                19/04/2011                                               */
0011  /*    Fecha de última modificación:     19/04/2011                                               */
0012  /*    Autor de última modificación:     Norma M. Chonay V.                                       */
0013  /*    Versión:                          1.1                                                      */
0014  /*    Prerrequisitos:                   Ninguno                                                  */
0015  /*****************************************************************************************************/
0016  -->
0017  <html xmlns="http://www.w3.org/1999/xhtml" >
0018      <head runat="server">
0019          <title>Inicio de sesión</title>
0020          <link href="../HojasEstilo/cssGeneral.css" rel="stylesheet" type="text/css" />
0021          <script type ="text/javascript" language="jscript" src="../Librerias/libUtilidades.1.1.js">
0022                          </script>
0023      </head>
0024      <body>
0025          <form id="frmInicio" runat="server">
0026              <div style="width: 514px">
0027                  <fieldset >
0028                      <legend >
0029                          <img src="../Img/imgAyuda.gif" style="width: 43px; height: 40px; margin-right: 0px;"
0030                              alt="Ayuda" id="imgAyuda"
0031                              onclick="pcPrAyudaLogin(document.forms[0].name,'')" />
0032                      </legend>
0033                      <asp:Table ID="tblPrincipal" runat="server" style="margin-top: 0px" Width="514px">
0034                          <asp:TableRow >
0035                              <asp:TableCell Width="5%"></asp:TableCell>
0036                              <asp:TableCell Width="90%" >
0037                                  <asp:Login ID="ctrLogin" runat="server" DisplayRememberMe="False"
0038                                      LoginButtonText="Entrar" UserNameLabelText="Usuario:" TitleText="">
0039                                  <LayoutTemplate>
0040                                      <table border="0" cellpadding="1" cellspacing="0" style="width:100%">
0041                                          <tr>
0042                                              <td align="center"  colspan="3">
0043                                                  <asp:Label ID="lblNombre" runat="server" Text="SISTEMA.GT"
0044                                                      CssClass="clsTitulo">
0045                                                  </asp:Label>
0046                                              </td>
0047                                          </tr>
0048                                           <tr style="height:15px">
0049                                              <br />
0050                                          </tr>
0051                                          <tr>
0052                                              <td align="left" style="width:25%">
0053                                              </td>
```

Continuación de la figura A-19.

```
0054                                <td align="left" style="width:25%">
0055                                   <asp:Label ID="lblUsuario" runat="server"
0056                                      AssociatedControlID="UserName"
0057                                      CssClass="clsEtiqueta">Usuario:</asp:Label>
0058                                </td>
0059                                <td align="left"   style="width:50%">
0060                                   <asp:TextBox ID="UserName" runat="server" TabIndex="1"
0061                                      MaxLength="10"
0062                                      CssClass="clsTexto" Width="125px"
0063                                      onkeypress="pcPbValidarCaracteres(event,
0064                                         frmInicio['ctrLogin_Password'], 1);">
0065                                   </asp:TextBox>
0066                                </td>
0067                             </tr>
0068                             <tr style="height:5px">
0069                                <br />
0070                             </tr>
0071                             <tr>
0072                                <td align="left" style="width:25%">
0073                                </td>
0074                                <td align="left"  style="width:25%">
0075                                   <asp:Label ID="lblClave" runat="server"
0076                                      AssociatedControlID="Password"
0077                                      CssClass="clsEtiqueta">Contraseña:</asp:Label>
0078                                </td>
0079                                <td align="left"  style="width:50%">
0080                                   <asp:TextBox ID="Password" runat="server" TextMode="Password"
0081                                      TabIndex="2" MaxLength="10"
0082                                      Width="125px" CssClass="clsClave"
0083                                      onkeypress="pcPbValidarCaracteres(event,
0084                                         frmInicio['ctrLogin_btnEntrar'], 1);">
0085                                   </asp:TextBox>
0086                                </td>
0087                             </tr>
0088                             <tr style="height:15px">
0089                                <br />
0090                             </tr>
0091                             <tr>
0092                                <td align="center"  colspan="3">
0093                                   <asp:Button ID="btnEntrar" runat="server" CommandName="Login"
0094                                      Text="ENTRAR"
0095                                      TabIndex="3" AccessKey="E" CssClass="clsBoton"
0096                                      OnClick="pcPrClicEntrar"/>
0097                                   <asp:Button ID="btnCancelar" runat="server" Text="CANCELAR"
0098                                      TabIndex="4" AccessKey="C" CssClass="clsBoton"/>
0099                                   <asp:Button ID="btnCambiarClave" runat="server"
0100                                      Text="RECUPERAR CLAVE" OnClick="pcPrClicRecuperarClave"
0101                                      TabIndex="5" AccessKey="R" CssClass="clsBoton"/>
0102                                </td>
0103                             </tr>
0104                          </table>
0105                       </LayoutTemplate>
0106                       </asp:Login>
```

Continuación de la figura A-19.

```
0107                        </asp:TableCell>
0108                        <asp:TableCell Width="5%"></asp:TableCell>
0109                    </asp:TableRow>
0110                </asp:Table>
0111                <br />
0112                <br />
0113            </fieldset >
0114        </div>
0115        <asp:HiddenField ID="hdnMsjError" runat="server"/>
0116        <asp:HiddenField ID="hdnId" runat="server"/>
0117        <asp:HiddenField ID="hdnUsuario" runat="server"/>
0118        <asp:HiddenField ID="hdnClave" runat="server"/>
0119        <asp:HiddenField ID="hdnNombres" runat="server"/>
0120        <asp:HiddenField ID="hdnApellidos" runat="server"/>
0121        <asp:HiddenField ID="hdnEmail" runat="server"/>
0122    </form>
0123  </body>
0124 </html>
```

Fuente: elaboración propia.

Figura A-20. **Código de frmInicio.aspx.vb**

```
0001  /*************************************************************************************************/
0002  /*   Archivo:                      frmInicio.aspx.vb                                         */
0003  /*   Descripción:                  Forma para ingresar credenciales de acceso.               */
0004  /*   Autor:                        Norma M. Chonay V.                                        */
0005  /*   Fecha de creación:            21/04/2011                                                */
0006  /*   Fecha de última modificación: 21/04/2011                                                */
0007  /*   Autor de última modificación: Norma M. Chonay V.                                        */
0008  /*   Versión:                      1.1                                                       */
0009  /*   Prerrequisitos:               Ninguno                                                   */
0010  /*************************************************************************************************/
0011
0012  Imports System.Web.UI.WebControls
0013  Imports Oracle.DataAccess.Client
0014  Imports Oracle.DataAccess.Types
0015
0016  Partial Public Class frmInicio
0017      Inherits System.Web.UI.Page
0018
0019      'Instanciar la librería de utilidades
0020      Public modHerr As New modHerramientas
0021
0022      'Procedimiento Load
0023      Protected Sub pcPrLoad(ByVal pVObjTipoObjeto As Object, _
0024                             ByVal pVEvtEvento As System.EventArgs) Handles Me.Load
0025          If IsPostBack Then
```

Continuación de la figura A-20.

```
0026            'Script para mostrar los mensajes
0027            If (Not ClientScript.IsStartupScriptRegistered(Me.GetType(), "alert")) Then
0028                Dim vrLStrScript As New StringBuilder()
0029                vrLStrScript.Append("<script language='javascript' type='text/javascript'>")
0030                vrLStrScript.Append(" if (frmInicio.hdnMsjError.value != '') " + _
0031                                    "{alert(frmInicio.hdnMsjError.value);}")
0032                vrLStrScript.Append(" frmInicio.hdnMsjError.value = '';")
0033                vrLStrScript.Append(" frmInicio['ctrLogin_Password'].focus();")
0034                vrLStrScript.Append(" </script>")
0035                ClientScript.RegisterStartupScript(Me.GetType(), "alert", vrLStrScript.ToString())
0036            End If
0037            CType(ctrLogin.FindControl("btnCancelar"), Button).Attributes.Add("onclick", _
0038                "javascript:pcPrClicCancelarLogin(frmInicio['ctrLogin_UserName'], " + _
0039                "frmInicio['ctrLogin_Password']); return false;")
0040        End If
0041        SetFocus(ctrLogin)
0042    End Sub
0043
0044    '/*****************************************************************************************/
0045    '/************************** EVENTOS DE LOS CONTROLES ************************/
0046    '/*****************************************************************************************/
0047    'Se validan las credenciales
0048    Protected Sub pcPrClicEntrar()
0049        Try
0050            'Validar que se haya ingresado usuario y clave
0051            If (ctrLogin.UserName <> "" And ctrLogin.Password <> "") Then
0052                'Conectarse a Oracle
0053                Dim vrLOcleConexion As New OracleConnection
0054                vrLOcleConexion = modHerr.fPbOcleConectar()
0055
0056                'Validar si se pudo establecer la conexión con Oracle
0057                If Not (vrLOcleConexion Is Nothing) Then
0058                    Dim vrLTblDatos As New DataTable
0059                    Dim vrLsIntResultado As Int16 = 0
0060                    vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, _
0061                          ctrLogin.UserName.ToUpper, _
0062                          modHerr.fPbStrCifrar(ctrLogin.Password), _
0063                          "", 2, vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0064                    Select Case (vrLsIntResultado)
0065                        Case 0
0066                            hdnMsjError.Value = "USUARIO Y CONTRASEÑA NO VÁLIDOS"
0067                            ctrLogin.Focus()
0068                        Case 1
0069                            'Validar si la consulta devolvió datos
0070                            If (vrLTblDatos.Rows.Count = 1) Then
0071                                vrLOcleConexion.Close()
0072
0073                                'Pasar los datos del usuario a frmDatosUsuario
0074                                hdnId.Value = Convert.ToInt32(vrLTblDatos.Rows(0)("Id"))
0075                                hdnUsuario.Value = vrLTblDatos.Rows(0)("Usuario").ToString()
0076                                hdnClave.Value = vrLTblDatos.Rows(0)("Clave").ToString()
0077                                hdnNombres.Value = vrLTblDatos.Rows(0)("Nombres").ToString()
0078                                hdnApellidos.Value = vrLTblDatos.Rows(0)("Apellidos").ToString()
```

Continuación de la figura A-20.

```
0079                        hdnEmail.Value = vrLTblDatos.Rows(0)("EMail").ToString()
0080                        Server.Transfer("/RecuperarContraseña/modSeguridad/" + _
0081                                "frmDatosUsuario.aspx")
0082                    Else
0083                        hdnMsjError.Value = "NO HAY DATOS DEL USUARIO"
0084                        ctrLogin.Focus()
0085                    End If
0086                Case 2
0087                    hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES VÁLIDA"
0088                    ctrLogin.Focus()
0089                Case 3
0090                    hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0091                    ctrLogin.Focus()
0092                Case Else
0093                    'Error que viene de la librería
0094                    ctrLogin.Focus()
0095            End Select
0096        Else
0097            hdnMsjError.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0098                "DE DATOS"
0099            ctrLogin.Focus()
0100        End If
0101
0102    Else
0103        hdnMsjError.Value = "DEBE INGRESAR USUARIO Y CONTRASEÑA"
0104        ctrLogin.Focus()
0105    End If
0106
0107    Catch pVExExcepcion As Exception
0108        hdnMsjError.Value = pVExExcepcion.ToString()
0109        ctrLogin.Focus()
0110    End Try
0111 End Sub
0112
0113 '/*****************************************************************************************/
0114 '/************************************ PROCEDIMIENTOS *************************************/
0115 '/*****************************************************************************************/
0116 'Al presionar el botón RECUPERAR CLAVE se abre frmRecuperacionClave
0117 Protected Sub pcPrClicRecuperarClave()
0118     'Llamar a la ventana para recuperar clave
0119     Response.Redirect("/RecuperarContraseña/modSeguridad/" + _
0120                 "frmRecuperacionClave.aspx")
0121 End Sub
0122
0123 '/*****************************************************************************************/
0124 '/************************************** FUNCIONES ****************************************/
0125 '/*****************************************************************************************/
0126 'Devuelve el Id del usuario
0127 Public ReadOnly Property fPbDevolverId() As String
0128     Get
0129         Return hdnId.Value
0130     End Get
0131 End Property
```

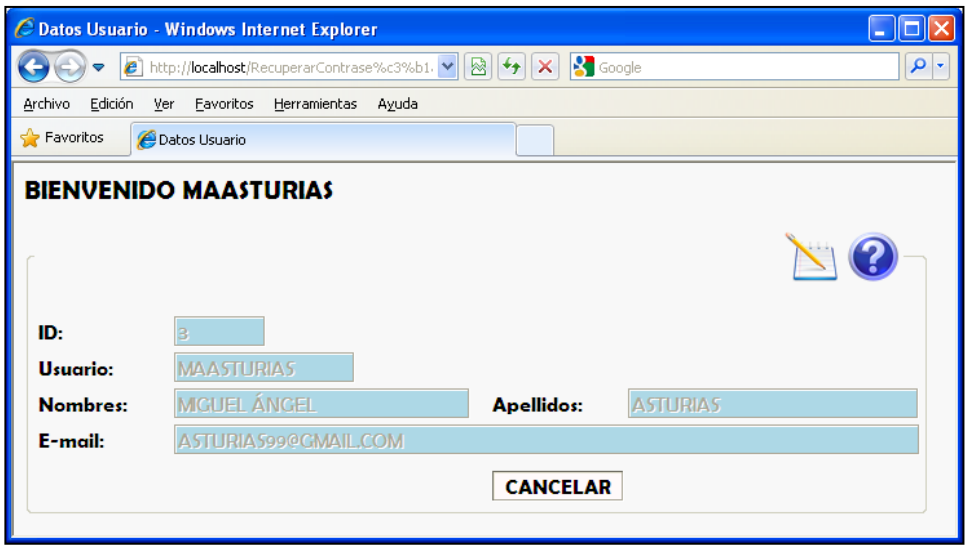
Continuación de la figura A-20.

```
0132
0133        'Devuelve el nombre del usuario
0134        Public ReadOnly Property fPbDevolverUsuario() As String
0135            Get
0136                Return hdnUsuario.Value
0137            End Get
0138        End Property
0139
0140        'Devuelve la clave cifrada del usuario
0141        Public ReadOnly Property fPbDevolverClave() As String
0142            Get
0143                Return hdnClave.Value
0144            End Get
0145        End Property
0146
0147        'Devuelve los nombres del usuario
0148        Public ReadOnly Property fPbDevolverNombres() As String
0149            Get
0150                Return hdnNombres.Value
0151            End Get
0152        End Property
0153
0154        'Devuelve los apellidos del usuario
0155        Public ReadOnly Property fPbDevolverApellidos() As String
0156            Get
0157                Return hdnApellidos.Value
0158            End Get
0159        End Property
0160
0161        'Devuelve el E-mail del usuario
0162        Public ReadOnly Property fPbDevolverEmail() As String
0163            Get
0164                Return hdnEmail.Value
0165            End Get
0166        End Property
0167    End Class
```

Fuente: elaboración propia.

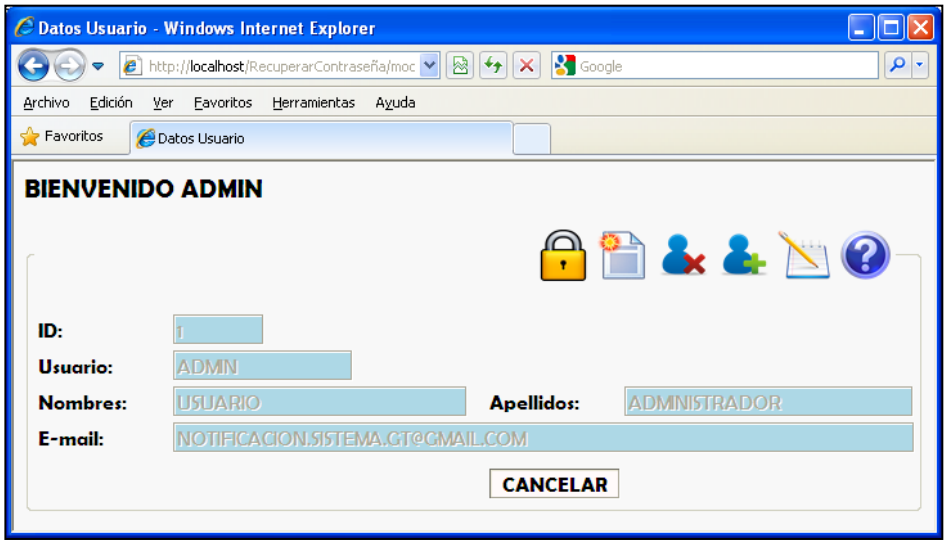
Una vez ingresadas las credenciales válidas, se muestra la página con los datos del usuario que inició sesión, página que varía dependiendo si es el usuario ADMIN o si es otro usuario. Las figuras que siguen, muestran el diseño de la página y el código de frmDatosUsuario.

Figura A-21. **Opciones de cualquier usuario que no es ADMIN**



Fuente: elaboración propia.

Figura A-22. **Opciones del usuario ADMIN**



Fuente: elaboración propia.

El código de las siguientes figuras implementa las opciones para editar datos de usuarios, agregar y eliminar usuarios, reiniciarle contraseña a determinado usuario e invocar al reporte.

Figura A-23. **Código de frmDatosUsuario.aspx**

```
0001  <%@ Page Language="vb" AutoEventWireup="false"
0002      CodeBehind="frmDatosUsuario.aspx.vb"
0003      Inherits="RecuperarContraseña.frmDatosUsuario" %>
0004  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
0005      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
0006  <!--
0007  /*****************************************************************************************************/
0008  /*    Archivo:                          frmDatosUsuario.aspx                                         */
0009  /*    Descripción:                      Forma para consultar datos y modificarlos.                   */
0010  /*    Autor:                            Norma M. Chonay V.                                           */
0011  /*    Fecha de creación:                26/04/2011                                                   */
0012  /*    Fecha de última modificación:     26/04/2011                                                   */
0013  /*    Autor de última modificación:     Norma M. Chonay V.                                           */
0014  /*    Versión:                          1.1                                                          */
0015  /*    Prerrequisitos:                   Ninguno                                                      */
0016  /*****************************************************************************************************/
0017  -->
0018  <html xmlns="http://www.w3.org/1999/xhtml" >
0019  <head runat="server">
0020      <title>Datos Usuario</title>
0021      <link href="../HojasEstilo/cssGeneral.css" rel="stylesheet" type="text/css" />
0022      <script type ="text/javascript" language="jscript" src="../Librerias/libUtilidades.1.1.js">
0023              </script>
0024              <script type ="text/javascript" language="jscript">
0025                  //Verificar si se ha iniciado sesión
0026                  function pcPbLoad()
0027                  {   if (document.all['lblNomUsuario'].innerText == 'BIENVENIDO ') {
0028                          alert('DEBE INICIAR SESIÓN');
0029                          window.location.href = 'http://localhost/RecuperarContraseña/modSeguridad' +
0030                              '/frmInicio.aspx';
0031                      }
0032                  }
0033              </script>
0034  </head>
0035  <body onload="pcPbLoad()">
0036      <form id="frmDatosUsuario" runat="server">
0037          <asp:Label ID="lblNomUsuario" runat="server" Text="BIENVENIDO "
0038              CssClass="clsEtqNomUsuario">
0039          </asp:Label>
0040          <br />
0041          <br />
0042          <div style="width: 700px">
0043              <fieldset >
0044                  <legend >
0045                      <asp:Label ID="lblOpcion" runat="server" Text="" CssClass="clsEtqOpcion"
```

Continuación de la figura A-23.

```
0046                    Width="20px">
0047                 </asp:Label>
0048                 <asp:Label ID="lblEspacio" runat="server" Text="" CssClass="clsEtqOpcion"
0049                    Width="360px">
0050                 </asp:Label>
0051                 <asp:ImageButton ImageUrl="~/Img/imgClave" style="width: 43px; height: 40px;
0052                    margin-right: 0px;" ToolTip="Reiniciar clave" id="imgClave" visible="false"
0053                    runat="server"/>
0054                 <asp:ImageButton ImageUrl="~/Img/imgReporte.gif" style="width: 43px;
0055                    height: 40px; margin-right: 0px;" ToolTip="Informe" id="imgInforme"
0056                    visible="false" runat="server"/>
0057                 <asp:ImageButton ImageUrl="~/Img/imgQuitar.png" style="width: 43px;
0058                    height: 40px; margin-right: 0px;" ToolTip="Eliminar usuario" id="imgQuitar"
0059                    visible="false" runat="server"/>
0060                 <asp:ImageButton ImageUrl="~/Img/imgAgregar.png" style="width: 43px;
0061                    height: 40px; margin-right: 0px;" ToolTip="Agregar usuario" id="imgAgregar"
0062                    visible="false" runat="server"/>
0063                 <asp:ImageButton ImageUrl="~/Img/imgEditar.png" style="width: 43px;
0064                    height: 40px; margin-right: 0px;" ToolTip="Editar datos" id="imgEditar"
0065                    runat="server" />
0066                 <img src="../Img/imgAyuda.gif" style="width: 43px; height: 40px;
0067                    margin-right: 0px;" alt="Ayuda" id="imgAyuda"
0068                    onclick="pcPrAyudaLogin(document.forms[0].name,
0069                       document.all['lblOpcion'].innerText, frmDatosUsuario['hdnUsuario'].value)" />
0070              </legend>
0071              <br />
0072              <asp:Table ID="tblPrincipal" runat="server" style="margin-top: 0px" Width="700px">
0073                 <asp:TableRow >
0074                    <asp:TableCell Width="15%">
0075                       <asp:Label ID="lblId" runat="server" Text="ID:"
0076                          CssClass="clsEtiqueta">
0077                       </asp:Label>
0078                    </asp:TableCell>
0079                    <asp:TableCell Width="10%">
0080                       <asp:TextBox ID="txtId" runat="server" CssClass="clsTexto" Width="98%"
0081                          MaxLength="10" TabIndex="1"
0082                          onkeypress="pcPbValidarCaracteres(event,
0083                             frmDatosUsuario['btnAceptar'], 4);">
0084                       </asp:TextBox>
0085                    </asp:TableCell>
0086                    <asp:TableCell Width="10%"></asp:TableCell>
0087                    <asp:TableCell Width="15%"></asp:TableCell>
0088                    <asp:TableCell Width="15%"></asp:TableCell>
0089                    <asp:TableCell Width="35%"></asp:TableCell>
0090                 </asp:TableRow>
0091                 <asp:TableRow >
0092                    <asp:TableCell Width="15%">
0093                       <asp:Label ID="lblUsuario" runat="server" Text="Usuario:"
0094                          CssClass="clsEtiqueta">
0095                       </asp:Label>
0096                    </asp:TableCell>
0097                    <asp:TableCell ColumnSpan="2">
0098                       <asp:TextBox ID="txtUsuario" runat="server" CssClass="clsTexto"
```

Continuación de la figura A-23.

```
0099                        Width="98%" MaxLength="10" TabIndex="2"
0100                        onkeypress="pcPbValidarCaracteres(event,
0101                           frmDatosUsuario['txtNombres'], 1);">
0102                     </asp:TextBox>
0103                  </asp:TableCell>
0104                  <asp:TableCell Width="15%"></asp:TableCell>
0105                  <asp:TableCell Width="15%"></asp:TableCell>
0106                  <asp:TableCell Width="35%"></asp:TableCell>
0107               </asp:TableRow>
0108               <asp:TableRow >
0109                  <asp:TableCell Width="15%">
0110                     <asp:Label ID="lblNombres" runat="server" Text="Nombres:"
0111                        CssClass="clsEtiqueta">
0112                     </asp:Label>
0113                  </asp:TableCell>
0114                  <asp:TableCell ColumnSpan="3">
0115                     <asp:TextBox ID="txtNombres" runat="server" CssClass="clsTexto"
0116                        Width="92%" MaxLength="50" TabIndex="3"
0117                        onkeypress="pcPbValidarCaracteres(event,
0118                           frmDatosUsuario['txtApellidos'], 2);">
0119                     </asp:TextBox>
0120                  </asp:TableCell>
0121                  <asp:TableCell Width="15%">
0122                     <asp:Label ID="lblApellidos" runat="server" Text="Apellidos:"
0123                        CssClass="clsEtiqueta">
0124                     </asp:Label>
0125                  </asp:TableCell>
0126                  <asp:TableCell Width="35%">
0127                     <asp:TextBox ID="txtApellidos" runat="server" CssClass="clsTexto"
0128                        Width="92%" MaxLength="50" TabIndex="4"
0129                        onkeypress="pcPbValidarCaracteres(event,
0130                           frmDatosUsuario['txtEmail'], 2);">
0131                     </asp:TextBox>
0132                  </asp:TableCell>
0133               </asp:TableRow>
0134               <asp:TableRow >
0135                  <asp:TableCell Width="15%">
0136                     <asp:Label ID="lblEmail" runat="server" Text="E-mail:"
0137                        CssClass="clsEtiqueta">
0138                     </asp:Label>
0139                  </asp:TableCell>
0140                  <asp:TableCell ColumnSpan="5">
0141                     <asp:TextBox ID="txtEmail" runat="server" CssClass="clsTexto"
0142                        Width="97%" MaxLength="50" TabIndex="5"
0143                        onkeypress="if (document.all['lblOpcion'].innerText == 'EDITANDO ...')
0144                           {pcPbValidarCaracteres(event, frmDatosUsuario['chkPregunta1'], 3);}
0145                           else
0146                           {pcPbValidarCaracteres(event, frmDatosUsuario['btnAceptar'], 3);}">
0147                     </asp:TextBox>
0148                  </asp:TableCell>
0149               </asp:TableRow>
0150               <asp:TableRow >
0151                  <asp:TableCell ColumnSpan="6">
```

Continuación de la figura A-23.

```
0152                    <table style="width:100%; margin:0px 0px 0px 0px" id="tblPreguntas"
0153                       border="0" cellpadding="1" cellspacing="0" runat="server">
0154                       <tr>
0155                          <td colspan="4">
0156                             <div class="clsEtiqueta">Elija una o más preguntas secretas</div>
0157                          </td>
0158                          <td colspan="1"></td>
0159                       </tr>
0160                       <tr>
0161                          <td style="width:5%">
0162                             <input id="chkPregunta1" type="checkbox" runat="server"
0163                                tabindex="6" onkeypress="pcPbValidarCaracteres(event,
0164                                   frmDatosUsuario['txtRespuesta1'], 6);"/>
0165                          </td>
0166                          <td style="width:44%">
0167                             <div id="divPregunta1" class="clsEtqPregunta"
0168                                runat="server">Pregunta 1</div>
0169                          </td>
0170                          <td style="width:2%"></td>
0171                          <td style="width:5%">
0172                             <input id="chkPregunta2" type="checkbox" runat="server"
0173                                tabindex="8" onkeypress="pcPbValidarCaracteres(event,
0174                                   frmDatosUsuario['txtRespuesta2'], 6);"/>
0175                          </td>
0176                          <td style="width:44%">
0177                             <div id="divPregunta2" class="clsEtqPregunta"
0178                                runat="server">Pregunta 2</div>
0179                          </td>
0180                       </tr>
0181                       <tr>
0182                          <td colspan="2">
0183                             <input id="txtRespuesta1" type="text" runat="server"
0184                                class="clsEtqRespuesta" style="width:95%" maxlength="100"
0185                                tabindex="7" onkeypress="pcPbValidarCaracteres(event,
0186                                   frmDatosUsuario['chkPregunta2'], 5);"/>
0187                          </td>
0188                          <td colspan="1"></td>
0189                          <td colspan="2">
0190                             <input id="txtRespuesta2" type="text" runat="server"
0191                                class="clsEtqRespuesta" style="width:95%" maxlength="100"
0192                                tabindex="9" onkeypress="pcPbValidarCaracteres(event,
0193                                   frmDatosUsuario['chkPregunta3'], 5);"/>
0194                          </td>
0195                       </tr>
0196                       <tr>
0197                          <td colspan="1">
0198                             <input id="chkPregunta3" type="checkbox" runat="server"
0199                                tabindex="10" onkeypress="pcPbValidarCaracteres(event,
0200                                   frmDatosUsuario['txtRespuesta3'], 6);"/>
0201                          </td>
0202                          <td colspan="1">
0203                             <div id="divPregunta3" class="clsEtqPregunta"
0204                                runat="server">Pregunta 3</div>
```

Continuación de la figura A-23.

```
0205                            </td>
0206                            <td colspan="1"></td>
0207                            <td colspan="1">
0208                               <input id="chkPregunta4" type="checkbox" runat="server"
0209                                  tabindex="12" onkeypress="pcPbValidarCaracteres(event,
0210                                     frmDatosUsuario['txtRespuesta4'], 6);"/>
0211                            </td>
0212                            <td colspan="1">
0213                               <div id="divPregunta4" class="clsEtqPregunta"
0214                                  runat="server">Pregunta 4</div>
0215                            </td>
0216                         </tr>
0217                         <tr>
0218                            <td colspan="2">
0219                               <input id="txtRespuesta3" type="text" runat="server"
0220                                  class="clsEtqRespuesta" style="width:95%" maxlength="100"
0221                                  tabindex="11" onkeypress="pcPbValidarCaracteres(event,
0222                                     frmDatosUsuario['chkPregunta4'], 5);"/>
0223                            </td>
0224                            <td colspan="1"></td>
0225                            <td colspan="2">
0226                               <input id="txtRespuesta4" type="text" runat="server"
0227                                  class="clsEtqRespuesta" style="width:95%" maxlength="100"
0228                                  tabindex="13" onkeypress="pcPbValidarCaracteres(event,
0229                                     frmDatosUsuario['chkPregunta5'], 5);"/>
0230                            </td>
0231                         </tr>
0232                         <tr>
0233                            <td colspan="1">
0234                               <input id="chkPregunta5" type="checkbox" runat="server"
0235                                  tabindex="14" onkeypress="pcPbValidarCaracteres(event,
0236                                     frmDatosUsuario['txtRespuesta5'], 6);"/>
0237                            </td>
0238                            <td colspan="4">
0239                               <div id="divPregunta5" class="clsEtqPregunta"
0240                                  runat="server">Pregunta 5</div>
0241                            </td>
0242                         </tr>
0243                         <tr>
0244                            <td colspan="2">
0245                               <input id="txtRespuesta5" type="text" runat="server"
0246                                  class="clsEtqRespuesta" style="width:95%" maxlength="100"
0247                                  tabindex="15" onkeypress="pcPbValidarCaracteres(event,
0248                                     frmDatosUsuario['pwClave'], 5);"/>
0249                            </td>
0250                            <td colspan="1"></td>
0251                            <td colspan="2"></td>
0252                         </tr>
0253                      </table>
0254                   </asp:TableCell>
0255                </asp:TableRow>
```

Continuación de la figura A-23.

```
0256                <asp:TableRow >
0257                   <asp:TableCell ColumnSpan="6">
0258                      <table style="width:100%; margin:0px 0px 0px 0px" id="tblClave"
0259                         border="0" cellpadding="1" cellspacing="0" runat="server">
0260                         <tr>
0261                            <td colspan="3">
0262                               <div class="clsEtqOpcion">PARA CAMBIAR CONTRASEÑA,
0263                                  INGRESE LOS SIGUIENTES DATOS:</div>
0264                            </td>
0265                         </tr>
0266                         <tr>
0267                            <td style="width:30%">
0268                               <div id="divClaveAnt" class="clsEtiqueta">Contraseña anterior:</div>
0269                            </td>
0270                            <td style="width:30%">
0271                               <input id="pwClave" type="password" class="clsClave"
0272                                  style="width:90%" maxlength="10" tabindex="16" runat="server"
0273                                  onkeypress="pcPbValidarCaracteres(event,
0274                                     frmDatosUsuario['pwClaveNueva'], 1);"/>
0275                            </td>
0276                            <td style="width:40%"></td>
0277                         </tr>
0278                          <tr>
0279                            <td colspan="1">
0280                               <div id="divClaveNueva" class="clsEtiqueta">Nueva contraseña:</div>
0281                            </td>
0282                            <td colspan="1">
0283                               <input id="pwClaveNueva" type="password" class="clsClave"
0284                                  style="width:90%" maxlength="10" tabindex="17" runat="server"
0285                                  onkeypress="pcPbValidarCaracteres(event,
0286                                     frmDatosUsuario['pwConfClave'], 1);"/>
0287                            </td>
0288                            <td colspan="1"></td>
0289                         </tr>
0290                         <tr>
0291                            <td colspan="1">
0292                               <div id="divConfClave"
0293                                  class="clsEtiqueta">Confirmar contraseña:</div>
0294                            </td>
0295                            <td colspan="1">
0296                               <input id="pwConfClave" type="password" class="clsClave"
0297                                  style="width:90%" maxlength="10" tabindex="18" runat="server"
0298                                  onkeypress="pcPbValidarCaracteres(event,
0299                                     frmDatosUsuario['btnAceptar'], 1);"/>
0300                            </td>
0301                            <td colspan="1"></td>
0302                         </tr>
0303                      </table>
0304                   </asp:TableCell>
0305                </asp:TableRow>
0306                <asp:TableRow >
0307                   <asp:TableCell ColumnSpan="4" HorizontalAlign="Right">
0308                      <asp:Repeater Id="repRepetidor" runat="server"
```

Continuación de la figura A-23.

```
0309                        OnItemCommand="pcPrClicRepetidor" Visible="false">
0310                        <ItemTemplate>
0311                           <asp:Button AccessKey="A" CssClass="clsBoton" TabIndex="19"
0312                              CommandName="btnOk" runat="server" Text="ACEPTAR"
0313                              ID="btnOk"/>
0314                        </ItemTemplate>
0315                     </asp:Repeater>
0316                     <asp:Button ID="btnAceptar" runat="server" Text="ACEPTAR"
0317                        AccessKey="A" CssClass="clsBoton" TabIndex="19"/>
0318                  </asp:TableCell>
0319                  <asp:TableCell ColumnSpan="2" HorizontalAlign="Left">
0320                     <asp:Button ID="btnCancelar" runat="server" Text="CANCELAR"
0321                        AccessKey="C" CssClass="clsBoton" tabIndex="20"/>
0322                  </asp:TableCell>
0323               </asp:TableRow>
0324            </asp:Table>
0325         </fieldset>
0326      </div>
0327      <asp:HiddenField ID="hdnMsjError" runat="server"/>
0328      <asp:HiddenField ID="hdnId" runat="server"/>
0329      <asp:HiddenField ID="hdnUsuario" runat="server"/>
0330      <asp:HiddenField ID="hdnClave" runat="server"/>
0331      <asp:HiddenField ID="hdnNombres" runat="server"/>
0332      <asp:HiddenField ID="hdnApellidos" runat="server"/>
0333      <asp:HiddenField ID="hdnEmail" runat="server"/>
0334      <asp:HiddenField ID="hdnRedireccionar" runat="server" Value="0"/>
0335   </form>
0336 </body>
0337 </html>
```

Fuente: elaboración propia.

Figura A-24. **Código de frmDatosUsuario.aspx.vb**

```
0001 /*****************************************************************************************************/
0002 /*   Archivo:                    frmDatosUsuario.aspx.vb                                             */
0003 /*   Descripción:                Forma para consultar datos y modificarlos.                          */
0004 /*   Autor:                      Norma M. Chonay V.                                                  */
0005 /*   Fecha de creación:          26/04/2011                                                          */
0006 /*   Fecha de última modificación: 26/04/2011                                                        */
0007 /*   Autor de última modificación: Norma M. Chonay V.                                                */
0008 /*   Versión:                    1.1                                                                 */
0009 /*   Prerrequisitos:             Ninguno                                                             */
0010 /*****************************************************************************************************/
0011
0012 Imports System.Web.UI.WebControls
0013 Imports Oracle.DataAccess.Client
0014 Imports Oracle.DataAccess.Types
```

Continuación de la figura A-24.

```
0015
0016  Partial Public Class frmDatosUsuario
0017      Inherits System.Web.UI.Page
0018
0019      'Instanciar la librería de utilidades
0020      Public modHerr As New modHerramientas
0021
0022      'Procedimiento Load
0023      Protected Sub pcPrLoad(ByVal pVObjTipoObjeto As Object, _
0024                             ByVal pVEvtEvento As System.EventArgs) Handles Me.Load
0025          Try
0026              If IsPostBack Then
0027                  'Script para mostrar los mensajes
0028                  If (Not ClientScript.IsStartupScriptRegistered(Me.GetType(), "alert")) Then
0029                      Dim vrLStrScript As New StringBuilder()
0030                      vrLStrScript.Append("<script language='javascript' type='text/javascript'>")
0031                      vrLStrScript.Append(" if (frmDatosUsuario.hdnMsjError.value != '')")
0032                      vrLStrScript.Append(" {   alert(frmDatosUsuario.hdnMsjError.value);")
0033                      vrLStrScript.Append(" }")
0034                      vrLStrScript.Append(" frmDatosUsuario.hdnMsjError.value = '';")
0035                      vrLStrScript.Append(" if (frmDatosUsuario['txtId'].disabled == false)")
0036                      vrLStrScript.Append(" {   frmDatosUsuario['txtId'].focus();")
0037                      vrLStrScript.Append(" }")
0038                      vrLStrScript.Append(" else")
0039                      vrLStrScript.Append(" {")
0040                      vrLStrScript.Append("      if (frmDatosUsuario['txtUsuario'].disabled == false)")
0041                      vrLStrScript.Append("      {  frmDatosUsuario['txtUsuario'].focus();")
0042                      vrLStrScript.Append("      }")
0043                      vrLStrScript.Append("      else")
0044                      vrLStrScript.Append("      {  if (frmDatosUsuario['txtNombres'].disabled " + _
0045                              "== false)")
0046                      vrLStrScript.Append("         {  frmDatosUsuario['txtNombres'].focus();")
0047                      vrLStrScript.Append("         }")
0048                      vrLStrScript.Append("         else")
0049                      vrLStrScript.Append("         {   if (frmDatosUsuario['btnCancelar'].disabled " + _
0050                              "== false)")
0051                      vrLStrScript.Append("            {  frmDatosUsuario['btnCancelar'].focus();")
0052                      vrLStrScript.Append("            }")
0053                      vrLStrScript.Append("         }")
0054                      vrLStrScript.Append("      }")
0055                      vrLStrScript.Append(" }")
0056                      vrLStrScript.Append(" if (frmDatosUsuario.hdnRedireccionar.value == '1') " + _
0057                              "{frmDatosUsuario.hdnRedireccionar.value = '0'; window.location.href " + _
0058                              "= 'http://localhost/RecuperarContraseña/modSeguridad/frmInicio.aspx';}")
0059                      vrLStrScript.Append(" </script>")
0060                      ClientScript.RegisterStartupScript(Me.GetType(), "alert", vrLStrScript.ToString())
0061                  End If
0062              End If
0063
0064              If Not IsPostBack Then
0065                  'Obtener los datos del usuario
0066                  Dim vrLfrmLogin As frmInicio
0067                  vrLfrmLogin = CType(Context.Handler, frmInicio)
```

Continuación de la figura A-24.

```
0068            hdnId.Value = vrLfrmLogin.fPbDevolverId
0069            hdnUsuario.Value = vrLfrmLogin.fPbDevolverUsuario
0070            hdnNombres.Value = vrLfrmLogin.fPbDevolverNombres
0071            hdnApellidos.Value = vrLfrmLogin.fPbDevolverApellidos
0072            hdnEmail.Value = vrLfrmLogin.fPbDevolverEmail
0073            hdnClave.Value = vrLfrmLogin.fPbDevolverClave
0074
0075            Dim vrLvTmpBotones As New ArrayList()
0076            vrLvTmpBotones.Add("Dato1")
0077            repRepetidor.DataSource = vrLvTmpBotones
0078            repRepetidor.DataBind()
0079
0080            Dim btnTemp As Button = repRepetidor.Items(0).FindControl("btnOk")
0081            btnTemp.Attributes.Add("onclick", "var vrLStrMsj = ''; " + _
0082                        " if (document.all['lblOpcion'].innerText == 'REINICIANDO ...') " + _
0083                        " { vrLStrMsj = '¿ESTÁ SEGURO DE REINICIARLE CONTRASEÑA " + _
0084                        "AL USUARIO?';" + _
0085                        " } " + _
0086                        " else " + _
0087                        " { vrLStrMsj = '¿ESTÁ SEGURO DE ELIMINAR EL USUARIO?';" + _
0088                        " } " + _
0089                        "return confirm(vrLStrMsj);")
0090        End If
0091
0092        'Mostrar los datos del usuario, si no se está en ninguna opción
0093        If (lblOpcion.Text = "") Then
0094            pcPrMostrarDatos()
0095        End If
0096
0097        'Si es el usuario administrador
0098        If (hdnUsuario.Value = "ADMIN") Then
0099            imgAgregar.Visible = True
0100            imgQuitar.Visible = True
0101            imgInforme.Visible = True
0102            imgClave.Visible = True
0103        End If
0104
0105        'Colocar las preguntas en los controles
0106        pcPrObtenerPreguntas()
0107    Catch pVExExcepcion As Exception
0108        hdnMsjError.Value = pVExExcepcion.ToString()
0109    End Try
0110 End Sub
0111
0112 '/*****************************************************************************************/
0113 '/************************************ PROCEDIMIENTOS *************************************/
0114 '/*****************************************************************************************/
0115 'Habilita o deshabilita los controles para editar los datos del usuario
0116 Private Sub pcPrDeshabilitarControles(ByVal pVBolOpcion As Boolean)
0117    'Si es el usuario administrador
0118    If (hdnUsuario.Value = "ADMIN" And lblOpcion.Text = "EDITANDO ...") Then
0119        txtUsuario.Enabled = False
0120    Else
```

Continuación de la figura A-24.

```
0121            txtUsuario.Enabled = pVBolOpcion
0122        End If
0123        txtId.Enabled = False
0124        txtNombres.Enabled = pVBolOpcion
0125        txtApellidos.Enabled = pVBolOpcion
0126        txtEmail.Enabled = pVBolOpcion
0127    End Sub
0128
0129    'Habilita o deshabilita los botones
0130    Private Sub pcPrDeshabilitarBotones(ByVal pVBolOpcion As Boolean)
0131        btnAceptar.Text = "ACEPTAR"
0132        btnAceptar.AccessKey = "A"
0133        btnAceptar.Visible = pVBolOpcion
0134        If (pVBolOpcion = True) Then 'Si el botón ACEPTAR está visible, habilitarlo
0135            btnAceptar.Enabled = True
0136        End If
0137        'Ocultar el botón ACEPTAR utilizado en las opciones REINICIAR y ELIMINAR
0138        repRepetidor.Visible = False
0139        'btnCancelar.Visible = pVBolOpcion
0140        tblClave.Visible = pVBolOpcion
0141        tblPreguntas.Visible = pVBolOpcion
0142    End Sub
0143
0144    'Colocar los datos originales del usuario en los controles
0145    Private Sub pcPrMostrarDatos()
0146        'Mostrar los datos del usuario
0147        If (lblNomUsuario.Text.Contains(hdnUsuario.Value) = False) Then
0148            lblNomUsuario.Text = lblNomUsuario.Text + hdnUsuario.Value
0149        End If
0150
0151        If (hdnUsuario.Value = "ADMIN") Then
0152            lblEspacio.Width = 360
0153        Else
0154            lblEspacio.Width = 550
0155        End If
0156
0157        'Deben haber datos del usuario, de lo contrario indica que no se ha iniciado sesión
0158        If (hdnId.Value <> "" And hdnUsuario.Value <> "" And hdnNombres.Value <> "" _
0159            And hdnApellidos.Value <> "" And hdnEmail.Value <> "") Then
0160            txtId.Text = hdnId.Value
0161            txtUsuario.Text = hdnUsuario.Value
0162            txtNombres.Text = hdnNombres.Value
0163            txtApellidos.Text = hdnApellidos.Value
0164            txtEmail.Text = hdnEmail.Value
0165        Else
0166            hdnMsjError.Value = "DEBE INICIAR SESIÓN"
0167        End If
0168
0169        pwClave.Value = ""
0170        pwClaveNueva.Value = ""
0171        pwConfClave.Value = ""
0172
0173        'Deshabilitar los controles
```

Continuación de la figura A-24.

```
0174        pcPrDeshabilitarControles(False)
0175        pcPrDeshabilitarBotones(False)
0176    End Sub
0177
0178    'Limpiar los controles, para ingresar nuevos usuarios
0179    Private Sub pcPrInicializarDatos()
0180        'Limpiar los datos de los controles
0181        txtId.Text = ""
0182        txtUsuario.Text = ""
0183        txtNombres.Text = ""
0184        txtApellidos.Text = ""
0185        txtEmail.Text = ""
0186        pwClave.Value = ""
0187        pwClaveNueva.Value = ""
0188        pwConfClave.Value = ""
0189    End Sub
0190
0191    'Inicializar los datos de los controles
0192    Private Sub pcPrCancelar(ByVal pVBolMostrarDatos As Boolean)
0193        'Indica que no se debe redireccionar a la página principal
0194        hdnRedireccionar.Value = "0"
0195
0196        If lblOpcion.Text = "" Then
0197            Response.Redirect("/RecuperarContraseña/modSeguridad/frmInicio.aspx")
0198        Else
0199            lblOpcion.Text = ""
0200
0201            If (pVBolMostrarDatos = True) Then
0202                'Mostrar los datos del usuario
0203                pcPrMostrarDatos()
0204            End If
0205        End If
0206    End Sub
0207
0208    'Se validan y almacenan los datos ingresados en la opción AGREGAR
0209    Private Sub pcPrGuardar()
0210        Try
0211            'Quitar espacios
0212            txtNombres.Text = txtNombres.Text.ToUpper().Trim()
0213            txtApellidos.Text = txtApellidos.Text.ToUpper().Trim()
0214            txtUsuario.Text = txtUsuario.Text.ToUpper().Trim()
0215            txtEmail.Text = txtEmail.Text.ToUpper().Trim()
0216
0217            'Validar que no estén vacíos
0218            If (txtUsuario.Text <> "" And _
0219                    txtNombres.Text <> "" And txtApellidos.Text <> "" And _
0220                    txtEmail.Text <> "") Then
0221                'Validar la dirección de correo electrónico
0222                If (modHerr.fPbBolValidarEmail(txtEmail.Text) = True) Then
0223                    'Generar la clave
0224                    Dim vrLsIntResultado As Int16 = 0
0225                    Dim vrLvNuevaClave(1) As Object
0226                    vrLvNuevaClave = modHerr.fPbVGenerarClave(txtUsuario.Text)
```

Continuación de la figura A-24.

```
0227                If (vrLvNuevaClave(0) = "" Or vrLvNuevaClave(1) = "") Then
0228                   hdnMsjError.Value = "NO SE PUDO GENERAR LA CONTRASEÑA, EL " + _
0229                         "USUARIO NO FUE CREADO"
0230                   pcPrCancelar(True)
0231                Else
0232                   'Conectarse a Oracle
0233                   Dim vrLOcleConexion As OracleConnection
0234                   vrLOcleConexion = modHerr.fPbOcleConectar()
0235
0236                   'Verificar si el usuario ya existe
0237                   modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text, "", "", 1, _
0238                      vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0239                   Select Case vrLsIntResultado
0240                      Case 0
0241                         Dim vrLIntId As Int32 = -1
0242                         vrLsIntResultado = 0
0243
0244                         'Crear el usuario
0245                         modHerr.pcPbGuardarUsuario(vrLIntId, _
0246                            txtUsuario.Text, _
0247                            vrLvNuevaClave(1), txtNombres.Text, txtApellidos.Text, _
0248                            txtEmail.Text, _
0249                            vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0250                         Select Case vrLsIntResultado
0251                            Case 0
0252                               hdnMsjError.Value = "EL USUARIO NO FUE CREADO, EL " + _
0253                                     "NOMBRE DE USUARIO INDICADO YA EXISTE"
0254                            Case 1
0255                               'Mostrar el código asignado al usuario y deshabilitar los controles
0256                               txtId.Text = vrLIntId.ToString()
0257                               pcPrDeshabilitarControles(False)
0258                               pcPrDeshabilitarBotones(True)
0259                               btnAceptar.Enabled = False
0260                               tblClave.Visible = False
0261                               tblPreguntas.Visible = False
0262
0263                               'Enviar el correo con la nueva contraseña
0264                               If (modHerr.fPbBolEnviarEmail(txtEmail.Text, _
0265                                  txtUsuario.Text, _
0266                                  vrLvNuevaClave(0), 1, hdnMsjError) = True) Then
0267                                  hdnMsjError.Value = "USUARIO CREADO.  LA CONTRASEÑA " + _
0268                                        "FUE ENVIADA A LA DIRECCIÓN DE CORREO INDICADA " + _
0269                                        "PARA ESTE USUARIO."
0270                               Else
0271                                  hdnMsjError.Value = "SE CREÓ EL USUARIO, PERO HUBO " + _
0272                                        "INCONVENIENTE AL ENVIAR LAS CREDENCIALES A LA " + _
0273                                        "CUENTA DE CORREO INDICADA"
0274                               End If
0275                            Case 3
0276                               hdnMsjError.Value = "EL USUARIO NO FUE CREADO, HUBO UN " + _
0277                                     "ERROR EN LA BASE DE DATOS"
0278                            Case Else
0279                               'Error que viene de la librería
```

Continuación de la figura A-24.

```
0280                        End Select
0281                    Case 1
0282                        hdnMsjError.Value = "EL USUARIO INDICADO YA EXISTE"
0283                        txtUsuario.Focus()
0284                    Case 2
0285                        hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " + _
0286                            "VÁLIDA"
0287                        txtUsuario.Focus()
0288                    Case 3
0289                        hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0290                        txtUsuario.Focus()
0291                    Case Else
0292                        '/Error que viene de la librería
0293                        txtUsuario.Focus()
0294                End Select
0295                vrLOcleConexion.Close()
0296            End If
0297        Else
0298            hdnMsjError.Value = "LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " + _
0299                "VÁLIDA"
0300            txtEmail.Focus()
0301        End If
0302    Else
0303        hdnMsjError.Value = "TODOS LOS CAMPOS SON OBLIGATORIOS"
0304        txtUsuario.Focus()
0305    End If
0306    Catch pVExExcepcion As Exception
0307        hdnMsjError.Value = pVExExcepcion.ToString()
0308        txtUsuario.Focus()
0309    End Try
0310 End Sub
0311
0312 'Se validan y actualizan los datos ingresados en la opción EDITAR
0313 Private Sub pcPrModificar()
0314    Try
0315        'Quitar espacios
0316        txtNombres.Text = txtNombres.Text.ToUpper().Trim()
0317        txtApellidos.Text = txtApellidos.Text.ToUpper().Trim()
0318        txtUsuario.Text = txtUsuario.Text.ToUpper().Trim()
0319        txtEmail.Text = txtEmail.Text.ToUpper().Trim()
0320        txtRespuesta1.Value = txtRespuesta1.Value.Trim()
0321        txtRespuesta2.Value = txtRespuesta2.Value.Trim()
0322        txtRespuesta3.Value = txtRespuesta3.Value.Trim()
0323        txtRespuesta4.Value = txtRespuesta4.Value.Trim()
0324        txtRespuesta5.Value = txtRespuesta5.Value.Trim()
0325
0326        'Validar que no estén vacíos
0327        If (txtId.Text <> "" And txtUsuario.Text <> "" And _
0328            txtNombres.Text <> "" And txtApellidos.Text <> "" And _
0329            txtEmail.Text <> "") Then
0330            'Validar la dirección de correo electrónico
0331            If (modHerr.fPbBolValidarEmail(txtEmail.Text) = True) Then
0332                'Validar las contraseñas
```

Continuación de la figura A-24.

```
0333            Dim vrLBolSeguir As Boolean = True
0334            'Si más de algún campo tiene datos
0335            If (pwClave.Value <> "" Or pwClaveNueva.Value <> "" _
0336               Or pwConfClave.Value <> "") Then
0337               'Validar si la clave del control txtClave es la misma que la que está registrada
0338               If (hdnClave.Value = modHerr.fPbStrCifrar(pwClave.Value)) Then
0339                  'Validar que la nueva clave y la confirmación tengan datos
0340                  If (pwClaveNueva.Value <> "" And pwConfClave.Value <> "") Then
0341                     'Validar que la nueva clave, sea la misma en los 2 controles
0342                     If (pwClaveNueva.Value <> pwConfClave.Value) Then
0343                        hdnMsjError.Value = "LA NUEVA CONTRASEÑA NO COINCIDE " + _
0344                              "CON LA CONFIRMACIÓN DE LA MISMA"
0345                        pwClaveNueva.Focus()
0346                        vrLBolSeguir = False
0347                     End If
0348                  Else
0349                     hdnMsjError.Value = "DEBE INGRESAR LA NUEVA CONTRASEÑA Y " + _
0350                           "LA CONFIRMACIÓN DE LA MISMA"
0351                     pwClaveNueva.Focus()
0352                     vrLBolSeguir = False
0353                  End If
0354               Else
0355                  hdnMsjError.Value = "LA <<CONTRASEÑA ANTERIOR>> INGRESADA " + _
0356                     "EN EL FORMULARIO NO COINCIDE CON LA ALMACENADA EN EL " + _
0357                     "SISTEMA ACTUALMENTE"
0358                  pwClave.Focus()
0359                  vrLBolSeguir = False
0360               End If
0361            End If
0362
0363            'Si pasó las validaciones de las contraseñas
0364            If (vrLBolSeguir = True) Then
0365               vrLBolSeguir = fPrBolValidarPreguntas()
0366               'Si pasó las validaciones de las preguntas
0367               If (vrLBolSeguir = True) Then
0368                  Dim vrLsIntResultado As Int16 = 0
0369                  Dim vrLStrClave As String = hdnClave.Value
0370                  'Conectarse a Oracle
0371                  Dim vrLOcleConexion As OracleConnection
0372                  vrLOcleConexion = modHerr.fPbOcleConectar()
0373
0374                  'Verificar si el usuario ya existe
0375                  modHerr.fPbTblBuscarUsuario(Convert.ToInt32(txtId.Text), txtUsuario.Text, _
0376                     "", "", 1, vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0377                  Select vrLsIntResultado
0378                     Case 0
0379                        'Hacer las modificaciones en la BDD
0380                        If (pwClaveNueva.Value <> "") Then
0381                           vrLStrClave = modHerr.fPbStrCifrar(pwClaveNueva.Value)
0382                        End If
0383
0384                        modHerr.pcPbModificarUsuario(Convert.ToInt32(txtId.Text), _
0385                           txtUsuario.Text, vrLStrClave, txtNombres.Text, txtApellidos.Text, _
```

Continuación de la figura A-24.

```
0386                                txtEmail.Text, txtRespuesta1.Value, txtRespuesta2.Value, _
0387                                txtRespuesta3.Value, txtRespuesta4.Value, txtRespuesta5.Value, _
0388                                1, vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0389                            Select Case vrLsIntResultado
0390                                Case 0
0391                                    hdnMsjError.Value = "NO SE REALIZÓ LA ACTUALIZACIÓN DE " + _
0392                                        "DATOS, EL USUARIO NO EXISTE"
0393                                    txtUsuario.Focus()
0394                                Case 1
0395                                    hdnMsjError.Value = "DATOS ACTUALIZADOS."
0396                                Case 3
0397                                    hdnMsjError.Value = "NO SE REALIZÓ LA ACTUALIZACIÓN DE " + _
0398                                        "DATOS, HUBO UN ERROR EN LA BASE DE DATOS"
0399                                    txtUsuario.Focus()
0400                                Case Else
0401                                    'Error que viene de la librería
0402                                    txtUsuario.Focus()
0403                            End Select
0404
0405                            vrLOcleConexion.Close()
0406                            'Indica que se debe redireccionar a la página principal
0407                            hdnRedireccionar.Value = "1"
0408                        Case 1
0409                            hdnMsjError.Value = "EL USUARIO INDICADO YA EXISTE"
0410                            txtUsuario.Focus()
0411                        Case 2
0412                            hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO " + _
0413                                "ES VÁLIDA"
0414                            txtUsuario.Focus()
0415                        Case 3
0416                            hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0417                            txtUsuario.Focus()
0418                        Case Else
0419                            'Error que viene de la librería
0420                            txtUsuario.Focus()
0421                    End Select
0422                End If
0423            End If
0424        Else
0425            hdnMsjError.Value = "LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " + _
0426                "VÁLIDA"
0427            txtEmail.Focus()
0428        End If
0429    Else
0430        hdnMsjError.Value = "TODOS LOS CAMPOS SON OBLIGATORIOS"
0431        txtUsuario.Focus()
0432    End If
0433  Catch pVExExcepcion As Exception
0434    hdnMsjError.Value = pVExExcepcion.ToString()
0435    txtUsuario.Focus()
0436  End Try
0437 End Sub
0438
```

Continuación de la figura A-24.

```
0439    'Se validan los datos y se actualiza la contraseña en la opción REINICIAR
0440    Private Sub pcPrReiniciarClave()
0441        Try
0442            'Validar que no estén vacíos
0443            If (txtUsuario.Text <> "" And txtEmail.Text <> "") Then
0444                'Validar la dirección de correo electrónico
0445                If (modHerr.fPbBolValidarEmail(txtEmail.Text) = True) Then
0446                    Dim vrLTblDatos As DataTable = New DataTable()
0447                    Dim vrLsIntResultado As Int16 = 0
0448                    Dim vrLvNuevaClave(1) As Object
0449
0450                    'Conectarse a Oracle
0451                    Dim vrLOcleConexion As OracleConnection
0452                    vrLOcleConexion = modHerr.fPbOcleConectar()
0453
0454                    'Consultar en la BDD si el usuario y el email coinciden
0455                    vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, _
0456                        txtUsuario.Text.ToUpper().Trim(), "", txtEmail.Text.ToUpper().Trim(), 0, _
0457                        vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0458                    Select Case vrLsIntResultado
0459                        Case 0
0460                            hdnMsjError.Value = "NO SE PUEDE RECUPERAR LA CLAVE, LA " + _
0461                                "INFORMACIÓN INDICADA NO ESTÁ REGISTRADA EN EL SISTEMA"
0462                        Case 1
0463                            'Validar si la consulta devolvió datos
0464                            If (vrLTblDatos.Rows.Count = 1) Then
0465                                'Obtener la nueva clave
0466                                vrLvNuevaClave = modHerr.fPbVGenerarClave( _
0467                                    txtUsuario.Text.ToUpper().Trim())
0468                                If (vrLvNuevaClave(0) <> "" And vrLvNuevaClave(1) <> "") Then
0469                                    'Modificar los datos en la BDD
0470                                    vrLsIntResultado = 0
0471                                    modHerr.pcPbModificarUsuario( _
0472                                        Convert.ToInt32(vrLTblDatos.Rows(0)("Id")), _
0473                                        vrLTblDatos.Rows(0)("Usuario").ToString(), _
0474                                        vrLvNuevaClave(1), _
0475                                        vrLTblDatos.Rows(0)("Nombres").ToString(), _
0476                                        vrLTblDatos.Rows(0)("Apellidos").ToString(), _
0477                                        vrLTblDatos.Rows(0)("EMail").ToString(), _
0478                                        "", "", "", "", "", 0, _
0479                                        vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0480
0481                                    Select Case vrLsIntResultado
0482                                        Case 0
0483                                            hdnMsjError.Value = "NO SE REALIZÓ LA ACTUALIZACIÓN DE " + _
0484                                                "LA CONTRASEÑA, EL USUARIO NO EXISTE"
0485                                        Case 1
0486                                            'Enviar el correo con la nueva contraseña
0487                                            If (modHerr.fPbBolEnviarEmail( _
0488                                                vrLTblDatos.Rows(0)("EMail").ToString(), _
0489                                                vrLTblDatos.Rows(0)("Usuario").ToString(), _
0490                                                vrLvNuevaClave(0), 2, hdnMsjError) = True) Then
0491                                                hdnMsjError.Value = "LA NUEVA CONTRASEÑA FUE " + _
```

Continuación de la figura A-24.

```
0492                                    "ENVIADA A LA DIRECCIÓN DE CORREO " + _
0493                                    "REGISTRADA EN EL SISTEMA."
0494                                Else
0495                                    hdnMsjError.Value = "SE ACTUALIZÓ LA CONTRASEÑA, " + _
0496                                        "PERO HUBO INCONVENIENTE AL ENVIAR LAS " + _
0497                                        "NUEVAS CREDENCIALES A LA CUENTA DE CORREO " + _
0498                                        "REGISTRADA"
0499                                End If
0500                            Case 3
0501                                hdnMsjError.Value = "NO SE REALIZÓ LA ACTUALIZACIÓN DE " + _
0502                                    "LA CONTRASEÑA, HUBO UN ERROR EN LA BASE DE " + _
0503                                    "DATOS"
0504                            Case Else
0505                                'Error que viene de la librería
0506                        End Select
0507                    Else
0508                        hdnMsjError.Value = "NO SE PUDO GENERAR UNA NUEVA " + _
0509                            "CONTRASEÑA"
0510                    End If
0511                Else
0512                    hdnMsjError.Value = "LA INFORMACIÓN INDICADA, NO ESTÁ " + _
0513                        "REGISTRADA EN EL SISTEMA"
0514                End If
0515            Case 2
0516                hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " + _
0517                    "VÁLIDA"
0518            Case 3
0519                hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0520            Case Else
0521                'Error que viene de la librería
0522        End Select
0523
0524        vrLOcleConexion.Close()
0525        'Indica que se debe redireccionar a la página principal
0526        hdnRedireccionar.Value = "1"
0527    Else
0528        hdnMsjError.Value = "LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " + _
0529            "VÁLIDA"
0530        txtEmail.Focus()
0531    End If
0532Else
0533    hdnMsjError.Value = "TODOS LOS CAMPOS SON OBLIGATORIOS"
0534    txtUsuario.Focus()
0535End If
0536Catch pVExExcepcion As Exception
0537    hdnMsjError.Value = pVExExcepcion.ToString()
0538    txtUsuario.Focus()
0539End Try
0540End Sub
0541
0542'Se validan y actualizan los datos ingresados en la opción ELIMINAR
0543Private Sub pcPrEliminar()
0544    Try
```

Continuación de la figura A-24.

```
0545            'Validar que no estén vacíos
0546            If (txtId.Text <> "" And txtUsuario.Text <> "" And txtNombres.Text <> "" And _
0547               txtApellidos.Text <> "" And txtEmail.Text <> "") Then
0548               Dim vrLsIntResultado As Int16 = 0
0549
0550               'Conectarse a Oracle
0551               Dim vrLOcleConexion As OracleConnection
0552               vrLOcleConexion = modHerr.fPbOcleConectar()
0553
0554               modHerr.pcPbEliminarUsuario(Convert.ToInt32(txtId.Text), vrLOcleConexion, _
0555                                           vrLsIntResultado, hdnMsjError)
0556               Select Case vrLsIntResultado
0557                  Case 0
0558                     hdnMsjError.Value = "NO SE ELIMINÓ EL USUARIO, EL USUARIO NO EXISTE"
0559                     txtUsuario.Focus()
0560                  Case 1
0561                     hdnMsjError.Value = "USUARIO ELIMINADO."
0562                  Case 3
0563                     hdnMsjError.Value = "NO SE ELIMINÓ EL USUARIO, HUBO UN ERROR " + _
0564                           "EN LA BASE DE DATOS"
0565                     txtUsuario.Focus()
0566                  Case Else
0567                     'Error que viene de la librería
0568                     txtUsuario.Focus()
0569               End Select
0570
0571               vrLOcleConexion.Close()
0572               pcPrCancelar(True)
0573            Else
0574               hdnMsjError.Value = "NO SE HA SELECCIONADO EL USUARIO A ELIMINAR"
0575               btnCancelar.Focus()
0576            End If
0577         Catch pVExExcepcion As Exception
0578            hdnMsjError.Value = pVExExcepcion.ToString()
0579            btnCancelar.Focus()
0580         End Try
0581      End Sub
0582
0583      'Se busca al usuario por el Id y se muestran los datos en los controles
0584      Private Sub pcPrBuscarUsuario()
0585         Try
0586            'Validar que se haya ingresado el id
0587            If (txtId.Text <> "") Then
0588               'Validar que no sea el usuario ADMINISTRADOR
0589               If ((lblOpcion.Text = "REINICIANDO ...") Or _
0590                  ((txtId.Text <> "1") And (lblOpcion.Text = "ELIMINANDO ..."))) Then
0591                  'Conectarse a Oracle
0592                  Dim vrLOcleConexion As OracleConnection
0593                  vrLOcleConexion = modHerr.fPbOcleConectar()
0594
0595                  'Validar si se pudo establecer la conexión con Oracle
0596                  If Not (vrLOcleConexion Is Nothing) Then
0597                     Dim vrLTblDatos As DataTable = New DataTable()
```

Continuación de la figura A-24.

```
0598                    Dim vrLsIntResultado As Int16 = 0
0599                    vrLTblDatos = modHerr.fPbTblBuscarUsuario(Convert.ToInt32(txtId.Text), "", _
0600                            "", "", 3, vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0601                    Select Case vrLsIntResultado
0602                       Case 0
0603                          hdnMsjError.Value = "EL USUARIO NO EXISTE"
0604                          txtId.Focus()
0605                       Case 1
0606                          'Validar si la consulta devolvió datos
0607                          If (vrLTblDatos.Rows.Count = 1) Then
0608                             'Mostrar los datos del usuario
0609                             txtUsuario.Text = vrLTblDatos.Rows(0)("Usuario").ToString()
0610                             txtNombres.Text = vrLTblDatos.Rows(0)("Nombres").ToString()
0611                             txtApellidos.Text = vrLTblDatos.Rows(0)("Apellidos").ToString()
0612                             txtEmail.Text = vrLTblDatos.Rows(0)("EMail").ToString()
0613
0614                             txtUsuario.Enabled = False
0615                             btnAceptar.Visible = False
0616                             repRepetidor.Visible = True
0617                             repRepetidor.Focus()
0618                          Else
0619                             hdnMsjError.Value = "NO HAY DATOS DEL USUARIO"
0620                             txtId.Focus()
0621                          End If
0622                       Case 2
0623                          hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " + _
0624                                "VÁLIDA"
0625                          txtId.Focus()
0626                       Case 3
0627                          hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0628                          txtId.Focus()
0629                       Case Else
0630                          'Error que viene de la librería
0631                          txtId.Focus()
0632                    End Select
0633                 Else
0634                    hdnMsjError.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0635                          "DE DATOS"
0636                    txtId.Focus()
0637                 End If
0638                 vrLOcleConexion.Close()
0639              Else
0640                 hdnMsjError.Value = "EL USUARIO ADMINISTRADOR NO SE PUEDE ELIMINAR"
0641                 txtId.Text = ""
0642                 txtId.Focus()
0643              End If
0644           Else
0645              hdnMsjError.Value = "DEBE INGRESAR EL CÓDIGO DE USUARIO"
0646              txtId.Focus()
0647           End If
0648        Catch pVExExcepcion As Exception
0649           hdnMsjError.Value = pVExExcepcion.ToString()
0650           txtId.Focus()
```

Continuación de la figura A-24.

```
0651        End Try
0652    End Sub
0653
0654    'Se buscan las preguntas y se pasan los datos a los controles
0655    Private Sub pcPrObtenerPreguntas()
0656        Try
0657            Dim vrLTblDatos As DataTable = New DataTable()
0658            Dim vrLsIntResultado As Int16 = 0
0659            vrLTblDatos = modHerr.fPbTblBuscarPreguntas(vrLsIntResultado, hdnMsjError)
0660
0661            Select Case vrLsIntResultado
0662                Case 0
0663                    hdnMsjError.Value = "NO HAY PREGUNTAS PARA MOSTRAR"
0664                    'Indica que se debe redireccionar a la página principal
0665                    hdnRedireccionar.Value = "1"
0666                Case 1
0667                    'Validar si la consulta devolvió datos
0668                    If (vrLTblDatos.Rows.Count = 5) Then
0669                        'Pasar los datos a los controles
0670                        Dim ctrPregunta As Global.System.Web.UI.HtmlControls.HtmlGenericControl
0671                        ctrPregunta = Nothing
0672                        For i As Int16 = 0 To 4
0673                            'Apuntar al checkbox correspondiente
0674                            Select Case i
0675                                Case 0
0676                                    ctrPregunta = divPregunta1
0677                                Case 1
0678                                    ctrPregunta = divPregunta2
0679                                Case 2
0680                                    ctrPregunta = divPregunta3
0681                                Case 3
0682                                    ctrPregunta = divPregunta4
0683                                Case 4
0684                                    ctrPregunta = divPregunta5
0685                            End Select
0686
0687                            'Si el control apunta a un checkbox válido
0688                            If Not (ctrPregunta Is Nothing) Then
0689                                ctrPregunta.InnerText = vrLTblDatos.Rows(i)("PREGUNTA").ToString()
0690                                ctrPregunta = Nothing
0691                            End If
0692                        Next i
0693                    Else
0694                        hdnMsjError.Value = "HACEN FALTA PREGUNTAS PARA MOSTRAR"
0695                        'Indica que se debe redireccionar a la página principal
0696                        hdnRedireccionar.Value = "1"
0697                    End If
0698                Case 3
0699                    hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0700                    'Indica que se debe redireccionar a la página principal
0701                    hdnRedireccionar.Value = "1"
0702                Case Else
0703                    'Error que viene de la librería
```

Continuación de la figura A-24.

```
0704                'Indica que se debe redireccionar a la página principal
0705                hdnRedireccionar.Value = "1"
0706            End Select
0707        Catch pVExExcepcion As Exception
0708            hdnMsjError.Value = pVExExcepcion.ToString()
0709            txtId.Focus()
0710        End Try
0711    End Sub
0712
0713    'Valida que si la pregunta está seleccionada, tenga una respuesta
0714    Private Sub pcPrValidarCadaPregunta(ByVal pVBolSeleccionado As Boolean, _
0715            ByRef pRCtrlRespuesta As HtmlInputText, _
0716            ByRef pRsIntContador As Int16)
0717        If (pVBolSeleccionado = True) Then 'Si la pregunta está seleccionada
0718            If (pRCtrlRespuesta.Value.Trim() = "") Then
0719                pRsIntContador += 1
0720            End If
0721        End If
0722    End Sub
0723
0724    'Tanto en la opción REINICIAR CONTRASEÑA, como en la opción ELIMINAR,
0725    'primero se debe indicar el ID del usuario
0726    Private Sub pcPrParaBuscarDatosUsuario(ByVal pVStrNomOpcion As String, _
0727            ByVal pVStrMensaje As String)
0728        '/Si se está en alguna opción
0729        If (lblOpcion.Text <> "") Then
0730            pcPrCancelar(False)
0731        End If
0732
0733        lblOpcion.Text = pVStrNomOpcion
0734
0735        'Habilitar los controles
0736        pcPrDeshabilitarControles(False)
0737        pcPrDeshabilitarBotones(True)
0738        tblClave.Visible = False
0739        tblPreguntas.Visible = False
0740        txtId.Enabled = True
0741        btnAceptar.Text = "BUSCAR"
0742        btnAceptar.AccessKey = "B"
0743
0744        'Limpiar los controles
0745        pcPrInicializarDatos()
0746
0747        hdnMsjError.Value = pVStrMensaje + " Y LUEGO PRESIONE EL BOTÓN <<BUSCAR>>"
0748        txtId.Focus()
0749    End Sub
0750
0751    '/*****************************************************************************************/
0752    '/************************************ FUNCIONES ******************************************/
0753    '/*****************************************************************************************/
0754    'Se valida que el usuario haya seleccionado al menos una pregunta
0755    'e ingresado la respuesta
0756    Private Function fPrBolValidarPreguntas() As Boolean
```

Continuación de la figura A-24.

```
0757        Dim vrLBolCorrecto As Boolean = False
0758        Dim vrLsIntContador As Int16 = 0
0759        'Validar que más de alguna pregunta esté seleccionada
0760        If (chkPregunta1.Checked = False And chkPregunta2.Checked = False _
0761              And chkPregunta3.Checked = False And chkPregunta4.Checked = False _
0762              And chkPregunta5.Checked = False) Then
0763           hdnMsjError.Value = "DEBE ELEGIR AL MENOS UNA PREGUNTA SECRETA E " + _
0764                 "INDICAR SU RESPUESTA"
0765           chkPregunta1.Focus()
0766        Else
0767           'Validar que cada pregunta tenga su respuesta
0768           pcPrValidarCadaPregunta(chkPregunta1.Checked, txtRespuesta1, vrLsIntContador)
0769           pcPrValidarCadaPregunta(chkPregunta2.Checked, txtRespuesta2, vrLsIntContador)
0770           pcPrValidarCadaPregunta(chkPregunta3.Checked, txtRespuesta3, vrLsIntContador)
0771           pcPrValidarCadaPregunta(chkPregunta4.Checked, txtRespuesta4, vrLsIntContador)
0772           pcPrValidarCadaPregunta(chkPregunta5.Checked, txtRespuesta5, vrLsIntContador)
0773
0774           If (vrLsIntContador = 0) Then 'Si no hubo error al validar las preguntas
0775              vrLBolCorrecto = True
0776           Else
0777              hdnMsjError.Value = "PARA CADA PREGUNTA SELECCIONADA DEBE " + _
0778                    "INDICAR UNA RESPUESTA"
0779              chkPregunta1.Focus()
0780           End If
0781        End If
0782        Return vrLBolCorrecto
0783     End Function
0784
0785     'Devuelve el título del reporte
0786     Public ReadOnly Property fPbDevolverTituloReporte() As String
0787        Get
0788           Return "LISTADO DE USUARIOS"
0789        End Get
0790     End Property
0791
0792     '/*********************************************************************************************/
0793     '/************************** EVENTOS DE LOS CONTROLES ************************/
0794     '/*********************************************************************************************/
0795     'Si se presiona el botón CANCELAR
0796     Private Sub pcPrClicCancelar(ByVal pVObjTipoObjeto As Object, ByVal _
0797           pVEvtEvento As System.EventArgs) Handles btnCancelar.Click
0798        'Se llama al procedimiento pcPrCancelar
0799        pcPrCancelar(True)
0800     End Sub
0801
0802     'Al dar clic en la imagen REINICIAR CONTRASEÑA, se habilitan los controles
0803     Private Sub pcPrClicReiniciarClave(ByVal pVObjTipoObjeto As Object, _
0804           ByVal pVEvtEvento As System.Web.UI.ImageClickEventArgs) Handles imgClave.Click
0805        'Indicarle al usuario que primero debe ingresar el ID del usuario a quien se le
0806        'reiniciará la clave
0807        pcPrParaBuscarDatosUsuario("REINICIANDO ...", "INGRESE EL ID DEL USUARIO " + _
0808              "A QUIEN LE REINICIARÁ LA CONTRASEÑA")
0809     End Sub
```

Continuación de la figura A-24.

```
0810
0811        'Al dar clic en la imagen ELIMINAR, se habilitan los controles
0812        Private Sub pcPrClicEliminar(ByVal pVObjTipoObjeto As Object, _
0813            ByVal pVEvtEvento As System.Web.UI.ImageClickEventArgs) Handles imgQuitar.Click
0814            'Indicarle al usuario que primero debe ingresar el ID del usuario a eliminar
0815            pcPrParaBuscarDatosUsuario("ELIMINANDO ...", "INGRESE EL ID DEL USUARIO " + _
0816                "A ELIMINAR")
0817        End Sub
0818
0819        'Al dar clic en la imagen EDITAR, se habilitan los controles
0820        Private Sub pcPrClicEditar(ByVal pVObjTipoObjeto As Object, _
0821            ByVal pVEvtEvento As System.Web.UI.ImageClickEventArgs) Handles imgEditar.Click
0822            'Si se está en alguna opción
0823            If (lblOpcion.Text <> "") Then
0824                pcPrCancelar(True)
0825            End If
0826
0827            lblOpcion.Text = "EDITANDO ..."
0828
0829            'Habilitar los controles
0830            pcPrDeshabilitarControles(True)
0831            pcPrDeshabilitarBotones(True)
0832            'Si es el usuario administrador
0833            If (hdnUsuario.Value = "ADMIN") Then
0834                txtNombres.Focus()
0835            Else
0836                txtUsuario.Focus()
0837            End If
0838        End Sub
0839
0840        'Al dar clic en la imagen AGREGAR, se habilitan los controles
0841        Private Sub pcPrClicAgregar(ByVal pVObjTipoObjeto As Object, _
0842            ByVal pVEvtEvento As System.Web.UI.ImageClickEventArgs) _
0843            Handles imgAgregar.Click
0844            'Si se está en alguna opción
0845            If (lblOpcion.Text <> "") Then
0846                pcPrCancelar(False)
0847            End If
0848
0849            lblOpcion.Text = "AGREGANDO ..."
0850
0851            'Habilitar los controles
0852            pcPrDeshabilitarControles(True)
0853            pcPrDeshabilitarBotones(True)
0854            tblClave.Visible = False
0855            tblPreguntas.Visible = False
0856
0857            'Limpiar los controles
0858            pcPrInicializarDatos()
0859            txtUsuario.Focus()
0860        End Sub
0861
0862        'Se validan los datos ingresados
```

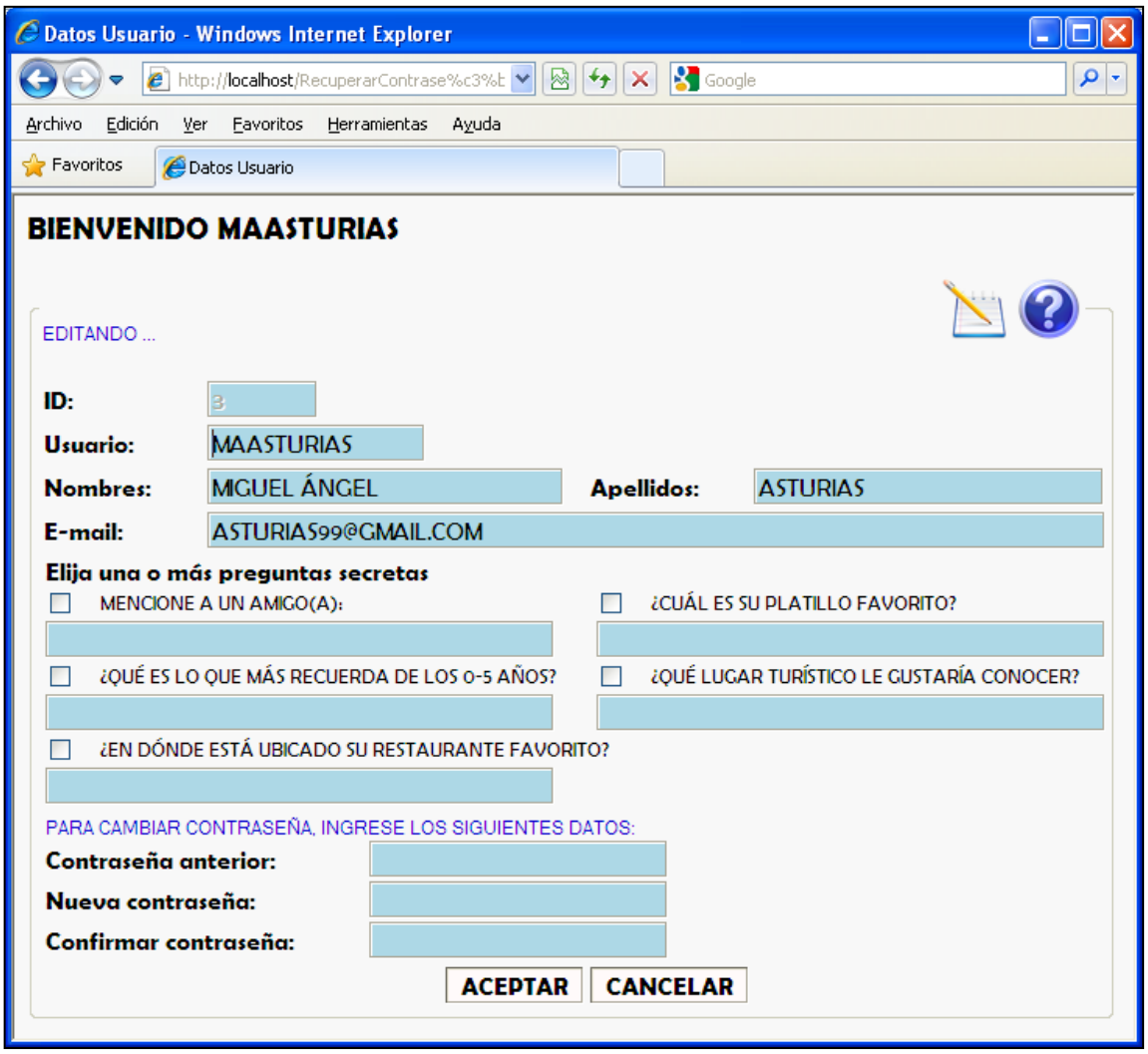
Continuación de la figura A-24.

```
0863    Private Sub pcPrClicAceptar(ByVal pVObjTipoObjeto As Object, _
0864            ByVal pVEvtEvento As System.EventArgs) Handles btnAceptar.Click
0865        Select Case (lblOpcion.Text)
0866            'Editar
0867            Case "EDITANDO ..."
0868                pcPrModificar()
0869            Case "AGREGANDO ..."
0870                'Agregar
0871                pcPrGuardar()
0872            Case "ELIMINANDO ...", "REINICIANDO ..."
0873                'Eliminar
0874                'Reiniciar
0875                If (btnAceptar.Text = "BUSCAR") Then
0876                    pcPrBuscarUsuario()
0877                End If
0878            Case Else
0879                'Default
0880        End Select
0881    End Sub
0882
0883    'Si  el usuario confirmó REINICIAR contraseña a determiando usuario,
0884    'o ELIMINAR un usuario
0885    Protected Sub pcPrClicRepetidor(ByVal pVObjTipoObjeto As Object, _
0886            ByVal pVEvtEvento As RepeaterCommandEventArgs)
0887        If lblOpcion.Text = "REINICIANDO ..." Then
0888            'Se hacen las validaciones para reiniciarle contraseña al usuario indicado
0889            pcPrReiniciarClave()
0890        End If
0891        If lblOpcion.Text = "ELIMINANDO ..." Then
0892            'Se hacen las validaciones para eliminar el usuario indicado
0893            pcPrEliminar()
0894        End If
0895    End Sub
0896
0897    'Al dar clik en el ícono de INFORME
0898    Private Sub pcPrClicInforme(ByVal pVObjTipoObjeto As Object, _
0899            ByVal pVEvtEvento As System.Web.UI.ImageClickEventArgs) _
0900            Handles imgInforme.Click
0901        'Llamar a la ventana de ayuda en línea
0902        Server.Transfer("/RecuperarContraseña/modSeguridad/repUsuarios.aspx")
0903    End Sub
0904 End Class
```

Fuente: elaboración propia.

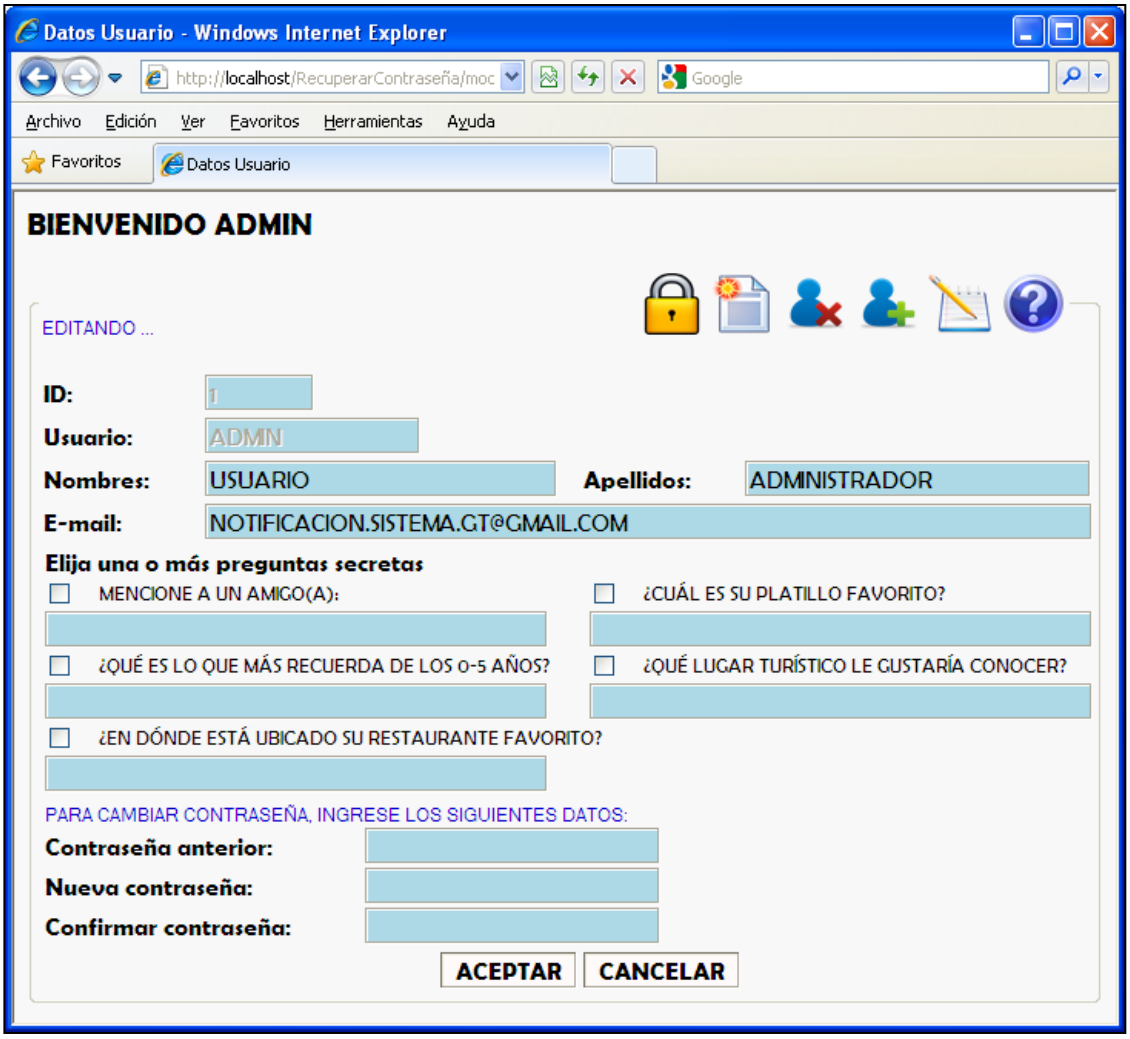
Las opciones del formulario frmDatosUsuario se describen a continuación por medio de imágenes.

Figura A-25. **Opción EDITAR de cualquier usuario que no es ADMIN**



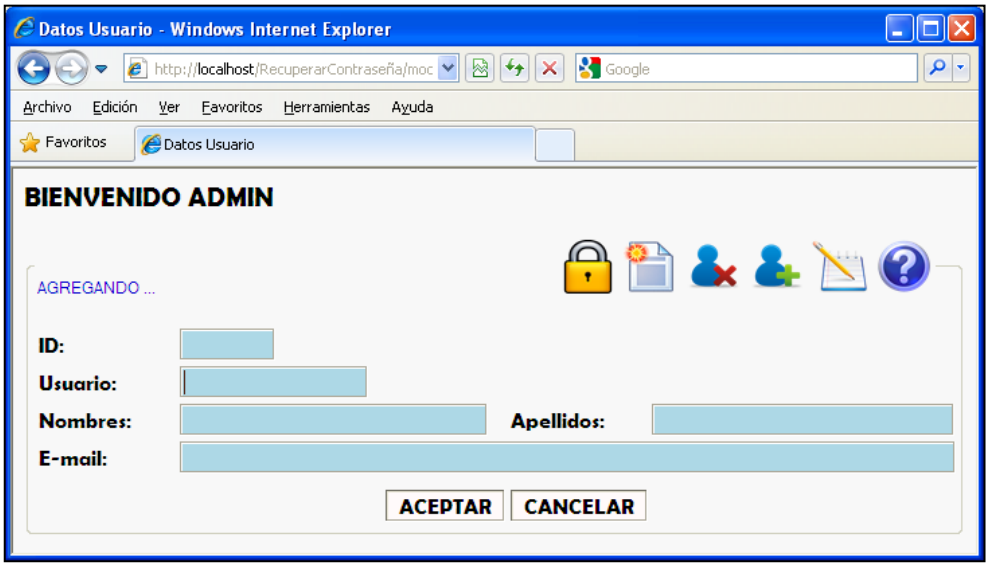
Fuente: elaboración propia.

Figura A-26. **Opción EDITAR del usuario ADMIN**



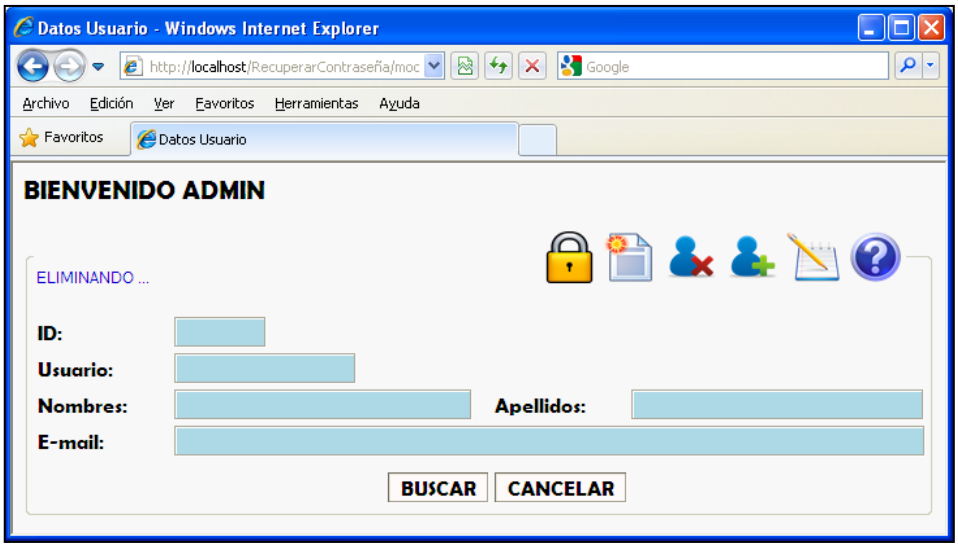
Fuente: elaboración propia.

Figura A-27. **Opción AGREGAR del usuario ADMIN**



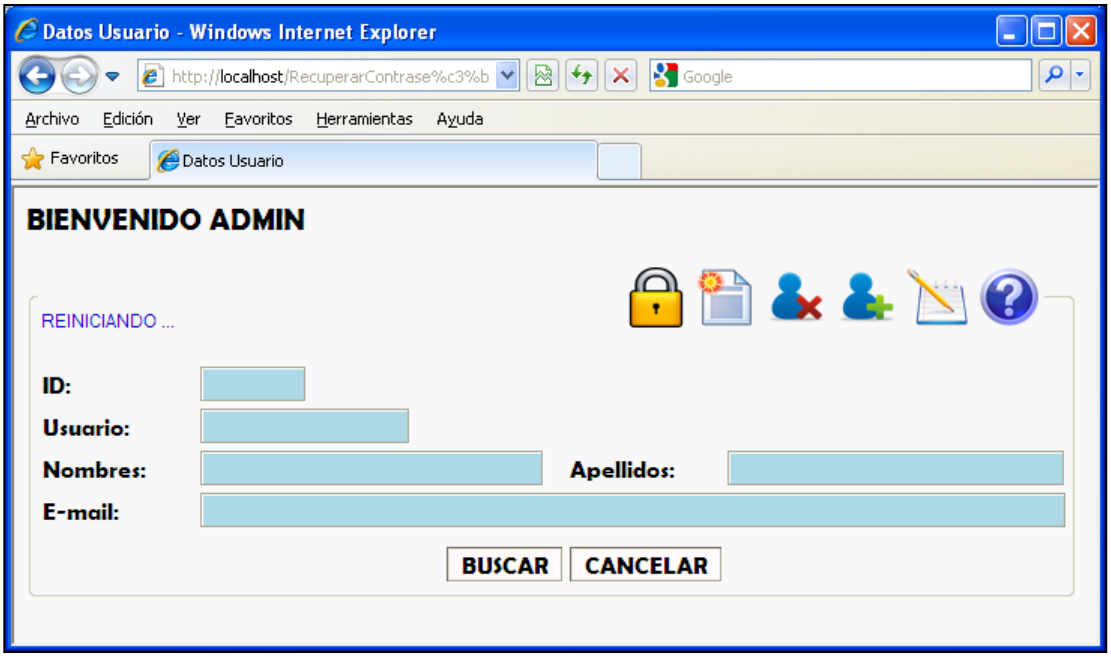
Fuente: elaboración propia.

Figura A-28. **Opción ELIMINAR del usuario ADMIN**



Fuente: elaboración propia.

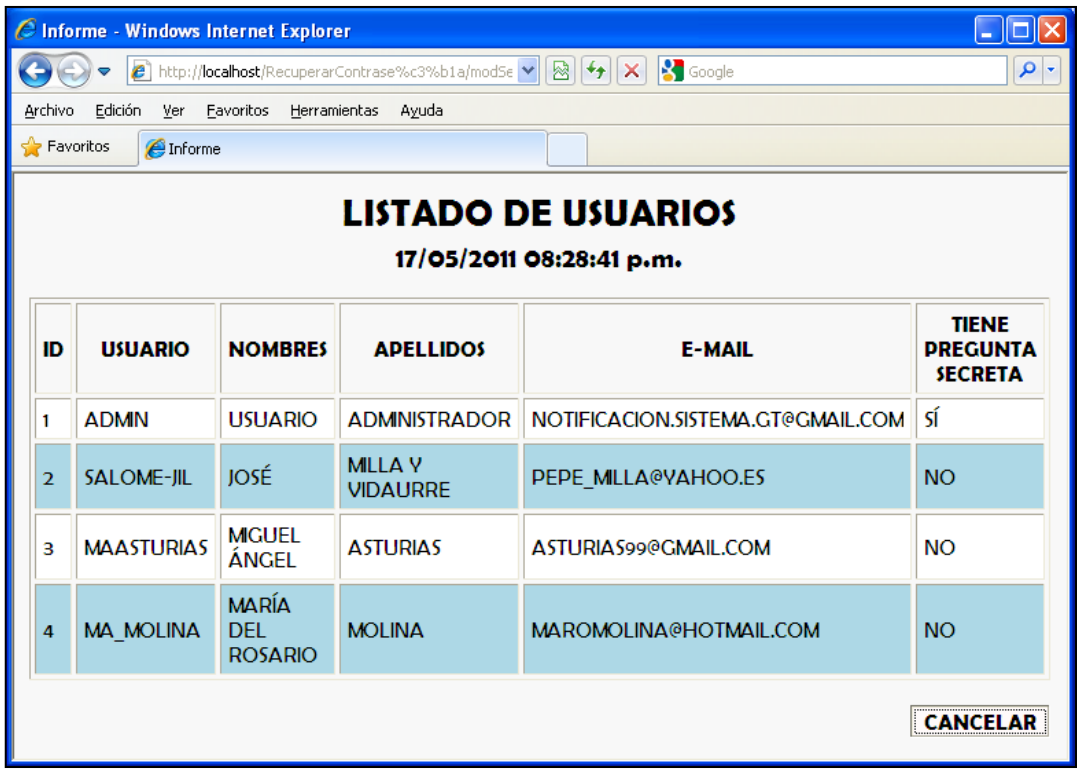
Figura A-29. **Opción REINICIAR CONTRASEÑA del usuario ADMIN**



Fuente: elaboración propia.

Al dar clic en la imagen identificada como imgInforme en frmDatosUsuario, se invoca a repUsuarios, formulario que implementa el reporte de los usuarios registrados en la base de datos. El informe de la aplicación se muestra en la siguiente imagen, seguido del código del formulario.

Figura A-30. **Informe**



Fuente: elaboración propia.

Figura A-31. **Código de repUsuarios.aspx**

```
0001  <%@ Page Language="vb" AutoEventWireup="false" CodeBehind="repUsuarios.aspx.vb"
0002      Inherits="RecuperarContraseña.frmInforme" %>
0003  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
0004      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
0005  <!--
0006  /*****************************************************************************************************/
0007  /*    Archivo:                        repUsuarios.aspx                                               */
0008  /*    Descripción:                    Forma para mostrar el reporte.                                 */
0009  /*    Autor:                          Norma M. Chonay V.                                             */
0010  /*    Fecha de creación:              03/05/2011                                                     */
0011  /*    Fecha de última modificación:   03/05/2011                                                     */
0012  /*    Autor de última modificación:   Norma M. Chonay V.                                             */
0013  /*    Versión:                        1.1                                                            */
0014  /*    Prerrequisitos:                 Ninguno                                                        */
0015  /*****************************************************************************************************/
0016  -->
```

Continuación de la figura A-31.

```
0017  <html xmlns="http://www.w3.org/1999/xhtml" >
0018  <head runat="server">
0019      <title>Informe</title>
0020      <link href="../HojasEstilo/cssGeneral.css" rel="stylesheet" type="text/css" />
0021      <script type ="text/javascript" language="jscript">
0022        //Verificar si se ha iniciado sesión
0023         function pcPbLoad()
0024         {   if (document.all['lblTitulo'].innerText == 'NINGUNO')
0025            {   alert('DEBE INICIAR SESIÓN');
0026                        window.location.href = 'http://localhost/RecuperarContraseña/modSeguridad/' +
0027                            'frmInicio.aspx';
0028                    }
0029                }
0030              </script>
0031  </head>
0032  <body onload="pcPbLoad()">
0033      <form id="frmInforme" runat="server">
0034          <table width="75%">
0035              <tr>
0036                  <td align="center">
0037                      <asp:Label ID="lblTitulo" runat="server" Text="Label"
0038                          CssClass="clsTitulo">NINGUNO</asp:Label>
0039                  </td>
0040              </tr>
0041              <tr>
0042                  <td align="center">
0043                      <asp:Label ID="lblFecha" runat="server" Text="Label"
0044                          CssClass="clsEtqNomUsuario">NINGUNO</asp:Label>
0045                  </td>
0046              </tr>
0047              <tr>
0048                  <td style="height:10px">
0049                  </td>
0050              </tr>
0051              <tr>
0052                  <td>
0053                      <asp:Repeater ID="repDatos" runat="server">
0054                          <HeaderTemplate>
0055                              <table border="1" cellpadding="5" cellspacing="3"
0056                                  style="border-style:groove">
0057                                  <tr class="clsEtiqueta" style="background-color:#F8F8F8">
0058                                      <th>ID</th>
0059                                      <th>USUARIO</th>
0060                                      <th>NOMBRES</th>
0061                                      <th>APELLIDOS</th>
0062                                      <th>E-MAIL</th>
0063                                      <th>TIENE PREGUNTA SECRETA</th>
0064                                  </tr>
0065                          </HeaderTemplate>
0066                          <ItemTemplate>
0067                              <tr class="clsEtqRespuesta" style="background-color:#FFFFFF">
0068                                  <td><%#DataBinder.Eval(Container.DataItem, "ID")%></td>
0069                                  <td><%#DataBinder.Eval(Container.DataItem, "USUARIO")%></td>
```

Continuación de la figura A-31.

```
0070                        <td><%#DataBinder.Eval(Container.DataItem, "NOMBRES")%></td>
0071                        <td><%#DataBinder.Eval(Container.DataItem, "APELLIDOS")%></td>
0072                        <td><%#DataBinder.Eval(Container.DataItem, "E-MAIL")%></td>
0073                        <td><%#DataBinder.Eval(Container.DataItem, _
0074                                                "TIENE PREGUNTA SECRETA")%></td>
0075                     </tr>
0076                  </ItemTemplate>
0077                  <AlternatingItemTemplate>
0078                     <tr class="clsEtqRespuesta">
0079                        <td><%#DataBinder.Eval(Container.DataItem, "ID")%></td>
0080                        <td><%#DataBinder.Eval(Container.DataItem, "USUARIO")%></td>
0081                        <td><%#DataBinder.Eval(Container.DataItem, "NOMBRES")%></td>
0082                        <td><%#DataBinder.Eval(Container.DataItem, "APELLIDOS")%></td>
0083                        <td><%#DataBinder.Eval(Container.DataItem, "E-MAIL")%></td>
0084                        <td><%#DataBinder.Eval(Container.DataItem, _
0085                                                "TIENE PREGUNTA SECRETA")%></td>
0086                     </tr>
0087                  </AlternatingItemTemplate>
0088                  <FooterTemplate>
0089                     </table>
0090                  </FooterTemplate>
0091               </asp:Repeater>
0092            </td>
0093         </tr>
0094         <tr>
0095            <td style="height:10px">
0096            </td>
0097         </tr>
0098         <tr>
0099            <td align="right">
0100               <asp:Button ID="btnCancelar" runat="server" Text="CANCELAR" AccessKey="C"
0101                  CssClass="clsBoton" tabIndex="1"/>
0102            </td>
0103         </tr>
0104      </table>
0105      <asp:HiddenField ID="hdnMsjError" runat="server"/>
0106   </form>
0107 </body>
0108 </html>
```

Fuente: elaboración propia.

Figura A-32. **Código de repUsuarios.aspx.vb**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                        repUsuarios.aspx.vb                                             */
0003  /*   Descripción:                    Forma para mostrar el reporte.                                  */
0004  /*   Autor:                          Norma M. Chonay V.                                              */
0005  /*   Fecha de creación:              03/05/2011                                                      */
0006  /*   Fecha de última modificación:   03/05/2011                                                      */
0007  /*   Autor de última modificación:   Norma M. Chonay V.                                              */
0008  /*   Versión:                        1.1                                                             */
0009  /*   Prerrequisitos:                 Ninguno                                                         */
0010  /*****************************************************************************************************/
0011
0012  Imports System.Web.UI.WebControls
0013  Imports Oracle.DataAccess.Client
0014  Imports Oracle.DataAccess.Types
0015
0016  Partial Public Class frmInforme
0017      Inherits System.Web.UI.Page
0018
0019      'Instanciar la librería de utilidades
0020      Public modHerr As New modHerramientas
0021
0022      'Procedimiento Load
0023      Protected Sub pcPrLoad(ByVal pVObjTipoObjeto As Object, _
0024          ByVal pVEvtEvento As System.EventArgs) Handles Me.Load
0025          Try
0026              If IsPostBack Then
0027                  'Script para mostrar los mensajes
0028                  If (Not ClientScript.IsStartupScriptRegistered(Me.GetType(), "alert")) Then
0029                      Dim vrLStrScript As New StringBuilder()
0030                      vrLStrScript.Append("<script language='javascript' type='text/javascript'>")
0031                      vrLStrScript.Append(" if (frmInforme.hdnMsjError.value != '') " + _
0032                                  "{alert(frmInforme.hdnMsjError.value);}")
0033                      vrLStrScript.Append(" frmInforme.hdnMsjError.value = '';")
0034                      vrLStrScript.Append(" frmInforme['btnCancelar'].focus();")
0035                      vrLStrScript.Append(" </script>")
0036                      ClientScript.RegisterStartupScript(Me.GetType(), "alert", vrLStrScript.ToString())
0037                  End If
0038              End If
0039
0040              If Not IsPostBack Then
0041                  'Obtener el título del reporte
0042                  Dim vrLfrmDatosUsuario As frmDatosUsuario
0043                  vrLfrmDatosUsuario = CType(Context.Handler, frmDatosUsuario)
0044                  lblTitulo.Text = vrLfrmDatosUsuario.fPbDevolverTituloReporte
0045              End If
0046
0047              'Mostrar la fecha
0048              lblFecha.Text = DateTime.Now.ToString()
0049
0050              'Conectarse a Oracle
0051              Dim vrLOcleConexion As OracleConnection
0052              vrLOcleConexion = modHerr.fPbOcleConectar()
0053
```

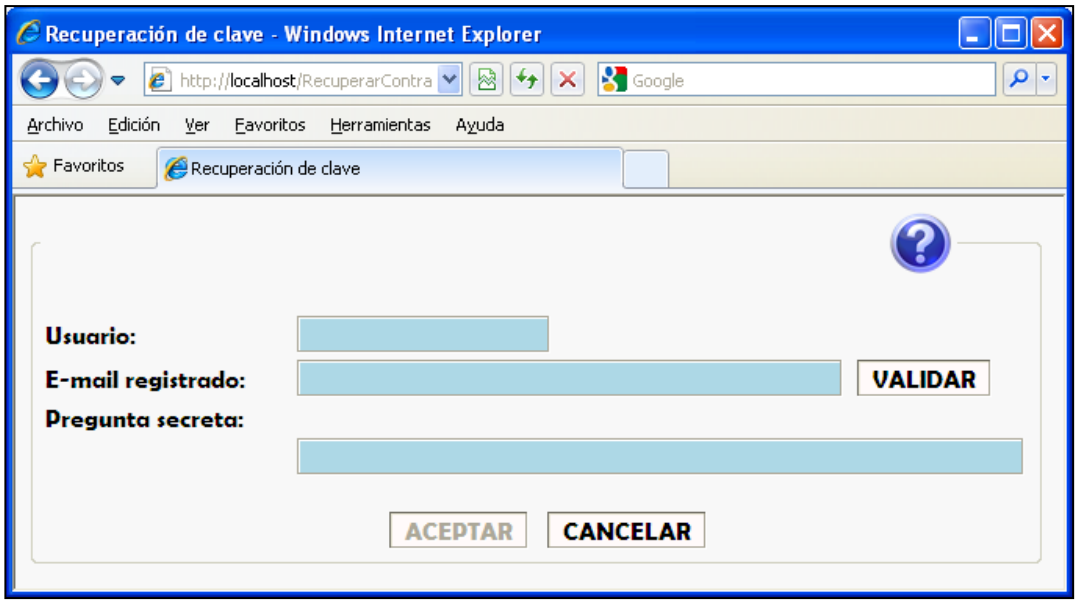
Continuación de la figura A-32.

```
0054            'Validar si se pudo establecer la conexión con Oracle
0055            If Not (vrLOcleConexion Is Nothing) Then
0056               Dim vrLTblDatos As DataTable = New DataTable()
0057               Dim vrLsIntResultado As Int16 = 0
0058               vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, "", "", "", 4, vrLOcleConexion, _
0059                     vrLsIntResultado, hdnMsjError)
0060
0061               Select Case vrLsIntResultado
0062                  Case 0
0063                     hdnMsjError.Value = "NO HAY DATOS"
0064                  Case 1
0065                     'Validar si la consulta devolvió datos
0066                     If (vrLTblDatos.Rows.Count >= 1) Then
0067                        'Mostrar los datos de los usuarios
0068                        repDatos.DataSource = vrLTblDatos
0069                        repDatos.DataBind()
0070                     Else
0071                        hdnMsjError.Value = "NO HAY DATOS"
0072                     End If
0073                  Case 2
0074                     hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES VÁLIDA"
0075                  Case 3
0076                     hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0077                  Case Else
0078                     'Error que viene de la librería
0079               End Select
0080            Else
0081               hdnMsjError.Value = "NO SE PUDO ESTABLECER CONEXIÓN A LA BASE " + _
0082                     "DE DATOS"
0083            End If
0084            vrLOcleConexion.Close()
0085            btnCancelar.Focus()
0086         Catch pVExExcepcion As Exception
0087            hdnMsjError.Value = pVExExcepcion.ToString()
0088         End Try
0089      End Sub
0090
0091      '/*****************************************************************************************/
0092      '/************************** EVENTOS DE LOS CONTROLES ************************/
0093      '/*****************************************************************************************/
0094      'Si se presiona el botón CANCELAR
0095      Private Sub pcPrClicCancelar(ByVal pVObjTipoObjeto As Object, _
0096            ByVal pVEvtEvento As System.EventArgs) Handles btnCancelar.Click
0097         Response.Redirect("/RecuperarContraseña/modSeguridad/frmInicio.aspx")
0098      End Sub
0099   End Class
```

Fuente: elaboración propia.

En el formulario frmInicio, al presionar el botón btnCambiarClave, se llama a frmRecuperacionClave, página que permite recuperar la clave, indicando el usuario, correo electrónico y la respuesta de alguna pregunta secreta, datos que aseguran de cierta forma que el usuario solicitante es quien dice ser, como se muestra en la siguiente figura.

Figura A-33. **Opción RECUPERAR CLAVE**



Fuente: elaboración propia.

Figura A-34. **Código de frmRecuperacionClave.aspx**

```
0001  <%@ Page Language="vb" AutoEventWireup="false"
0002      CodeBehind="frmRecuperacionClave.aspx.vb"
0003      Inherits="RecuperarContraseña.frmRecuperacionClave" %>
0004  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
0005      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
0006  <!--
0007  /*****************************************************************************************************/
0008  /*    Archivo:                          frmRecuperacionClave.aspx                                    */
0009  /*    Descripción:                      Forma para cambiar la clave.                                 */
0010  /*    Autor:                            Norma M. Chonay V.                                           */
0011  /*    Fecha de creación:                04/05/2011                                                   */
0012  /*    Fecha de última modificación:     04/05/2011                                                   */
0013  /*    Autor de última modificación:     Norma M. Chonay V.                                           */
0014  /*    Versión:                          1.1                                                          */
0015  /*    Prerrequisitos:                   Ninguno                                                      */
0016  /*****************************************************************************************************/
0017  -->
0018
0019  <html xmlns="http://www.w3.org/1999/xhtml" >
0020  <head runat="server">
0021      <title>Recuperación de clave</title>
0022      <link href="../HojasEstilo/cssGeneral.css" rel="stylesheet" type="text/css" />
0023      <script type ="text/javascript" language="jscript" src="../Librerias/libUtilidades.1.1.js">
0024      </script>
0025  </head>
0026  <body>
0027      <form id="frmRecuperacionClave" runat="server">
0028      <div style="width: 650px">
0029          <fieldset >
0030              <legend >
0031                  <asp:Label ID="lblEspacio" runat="server" Text="" CssClass="clsEtqOpcion"
0032                      Width="535px"></asp:Label>
0033                  <img src="../Img/imgAyuda.gif" style="width: 43px; height: 40px; margin-right: 0px;"
0034                      alt="Ayuda" id="imgAyuda"
0035                      onclick="pcPrAyudaLogin(document.forms[0].name, '', '')" />
0036              </legend>
0037              <br />
0038              <asp:Table ID="tblPrincipal" runat="server" style="margin-top: 0px" Width="650px">
0039                  <asp:TableRow >
0040                      <asp:TableCell Width="25%">
0041                          <asp:Label ID="lblUsuario" runat="server" Text="Usuario:"
0042                              CssClass="clsEtiqueta">
0043                          </asp:Label>
0044                      </asp:TableCell>
0045                      <asp:TableCell ColumnSpan="2">
0046                          <asp:TextBox ID="txtUsuario" runat="server" CssClass="clsTexto"
0047                              Width="98%" MaxLength="10" TabIndex="1"
0048                              onkeypress="pcPbValidarCaracteres(event,
0049                                  frmRecuperacionClave['txtEmail'], 1);"></asp:TextBox>
0050                      </asp:TableCell>
0051                      <asp:TableCell Width="20%"></asp:TableCell>
0052                      <asp:TableCell Width="10%"></asp:TableCell>
0053                      <asp:TableCell Width="20%"></asp:TableCell>
```

Continuación de la figura A-34.

```
0054                </asp:TableRow>
0055                <asp:TableRow >
0056                   <asp:TableCell Width="25%">
0057                      <asp:Label ID="lblEmail" runat="server" Text="E-mail registrado:"
0058                         CssClass="clsEtiqueta">
0059                      </asp:Label>
0060                   </asp:TableCell>
0061                   <asp:TableCell ColumnSpan="4">
0062                      <asp:TextBox ID="txtEmail" runat="server" CssClass="clsTexto"
0063                         Width="97%" MaxLength="50" TabIndex="2"
0064                         onkeypress="pcPbValidarCaracteres(event,
0065                            frmRecuperacionClave['btnValidar'], 3)"></asp:TextBox>
0066                   </asp:TableCell>
0067                   <asp:TableCell Width="20%">
0068                      <asp:Button ID="btnValidar" runat="server" Text="VALIDAR" AccessKey="V"
0069                         CssClass="clsBoton" TabIndex="3"/>
0070                   </asp:TableCell>
0071                </asp:TableRow>
0072                <asp:TableRow >
0073                   <asp:TableCell ColumnSpan="6">
0074                      <asp:Label ID="lblPregunta" runat="server" Text="Pregunta secreta:"
0075                         CssClass="clsEtiqueta">
0076                      </asp:Label>
0077                   </asp:TableCell>
0078                </asp:TableRow>
0079                <asp:TableRow >
0080                   <asp:TableCell Width="25%"></asp:TableCell>
0081                   <asp:TableCell ColumnSpan="5">
0082                      <asp:TextBox ID="txtRespuesta" runat="server" CssClass="clsEtqRespuesta"
0083                         Width="95%" MaxLength="100" TabIndex="4"
0084                         onkeypress="pcPbValidarCaracteres(event,
0085                            frmRecuperacionClave['btnAceptar'], 5)"></asp:TextBox>
0086                   </asp:TableCell>
0087                </asp:TableRow>
0088                <asp:TableRow >
0089                   <asp:TableCell >
0090                      <div style="height:15px"></div>
0091                   </asp:TableCell>
0092                </asp:TableRow>
0093                <asp:TableRow >
0094                   <asp:TableCell ColumnSpan="6" HorizontalAlign="Center">
0095                      <asp:Button ID="btnAceptar" runat="server" Text="ACEPTAR"
0096                         AccessKey="A" CssClass="clsBoton" TabIndex="5" Enabled="false"/>
0097                       
0098                      <asp:Button ID="btnCancelar" runat="server" Text="CANCELAR"
0099                         AccessKey="C" CssClass="clsBoton" tabIndex="6"/>
0100                   </asp:TableCell>
0101                </asp:TableRow>
0102             </asp:Table>
0103          </fieldset>
0104       </div>
0105       <asp:HiddenField ID="hdnMsjError" runat="server"/>
0106       <asp:HiddenField ID="hdnRespuesta" runat="server"/>
```

Continuación de la figura A-34.

```
0107        <asp:HiddenField ID="hdnRedireccionar" runat="server" Value="0"/>
0108    </form>
0109 </body>
0110 </html>
```

Fuente: elaboración propia.

Figura A-35. **Código de frmRecuperacionClave.aspx.vb**

```
0001 /*****************************************************************************************************/
0002 /*   Archivo:                       frmRecuperacionClave.aspx.vb                                     */
0003 /*   Descripción:                   Forma para cambiar la clave.                                     */
0004 /*   Autor:                         Norma M. Chonay V.                                               */
0005 /*   Fecha de creación:             04/05/2011                                                       */
0006 /*   Fecha de última modificación:  04/05/2011                                                       */
0007 /*   Autor de última modificación:  Norma M. Chonay V.                                               */
0008 /*   Versión:                       1.1                                                              */
0009 /*   Prerrequisitos:                Ninguno                                                          */
0010 /*****************************************************************************************************/
0011
0012 Imports Oracle.DataAccess.Client
0013 Imports Oracle.DataAccess.Types
0014
0015 Partial Public Class frmRecuperacionClave
0016     Inherits System.Web.UI.Page
0017
0018     'Instanciar la librería de utilidades
0019     Public modHerr As New modHerramientas
0020
0021     'Procedimiento Load
0022     Protected Sub pcPrLoad(ByVal pVObjTipoObjeto As Object, _
0023         ByVal pVEvtEvento As System.EventArgs) Handles Me.Load
0024         If IsPostBack Then
0025             'Script para mostrar los mensajes
0026             If (Not ClientScript.IsStartupScriptRegistered(Me.GetType(), "alert")) Then
0027                 Dim vrLStrScript As New StringBuilder()
0028                 vrLStrScript.Append("<script language='javascript' type='text/javascript'>")
0029                 vrLStrScript.Append(" if (frmRecuperacionClave.hdnMsjError.value != '') " + _
0030                     "{alert(frmRecuperacionClave.hdnMsjError.value);}")
0031                 vrLStrScript.Append(" frmRecuperacionClave.hdnMsjError.value = '';")
0032                 vrLStrScript.Append(" frmRecuperacionClave['txtUsuario'].focus();")
0033                 vrLStrScript.Append(" if (frmRecuperacionClave.hdnRedireccionar.value == '1') " + _
0034                     "{frmRecuperacionClave.hdnRedireccionar.value = '0'; window.location.href " + _
0035                     "= 'http://localhost/RecuperarContraseña/modSeguridad/frmInicio.aspx';}")
0036                 vrLStrScript.Append(" </script>")
0037                 ClientScript.RegisterStartupScript(Me.GetType(), "alert", vrLStrScript.ToString())
0038             End If
0039         End If
```

Continuación de la figura A-35.

```
0040        txtUsuario.Focus()
0041    End Sub
0042
0043    '/*************************************************************************************************/
0044    '/************************** EVENTOS DE LOS CONTROLES ************************/
0045    '/*************************************************************************************************/
0046
0047    'Si se presiona el botón VALIDAR, se validan los datos ingresados y se muestra
0048    'una de las preguntas secretas registradas para el usuario
0049    Private Sub pcPrClicValidar(ByVal pVObjTipoObjeto As Object, _
0050            ByVal pVEvtEvento As System.EventArgs) Handles btnValidar.Click
0051        Try
0052            'Validar que no estén vacíos los controles
0053            If (txtUsuario.Text <> "" And txtEmail.Text <> "") Then
0054                'Validar la dirección de correo electrónico
0055                If (modHerr.fPbBolValidarEmail(txtEmail.Text) = True) Then
0056                    Dim vrLTblDatos As DataTable = New DataTable()
0057                    Dim vrLsIntResultado As Int16 = 0
0058
0059                    'Conectarse a Oracle
0060                    Dim vrLOcleConexion As OracleConnection
0061                    vrLOcleConexion = modHerr.fPbOcleConectar()
0062
0063                    'Consultar en la BDD si el usuario y el email coinciden
0064                    vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, _
0065                        txtUsuario.Text.ToUpper().Trim(), "", txtEmail.Text.ToUpper().Trim(), 0, _
0066                        vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0067
0068                    Select Case vrLsIntResultado
0069                        Case 0
0070                            hdnMsjError.Value = "NO SE PUEDE RECUPERAR LA CLAVE, LA " + _
0071                                "INFORMACIÓN INDICADA NO ESTÁ REGISTRADA EN EL SISTEMA"
0072                        Case 1
0073                            'Validar si la consulta devolvió datos
0074                            If (vrLTblDatos.Rows.Count = 1) Then
0075                                'Mostrar la pregunta secreta
0076                                If (vrLTblDatos.Rows(0)("Pregunta").ToString() <> "") Then
0077                                    lblPregunta.Text = vrLTblDatos.Rows(0)("Pregunta").ToString()
0078                                    hdnRespuesta.Value = vrLTblDatos.Rows(0)("Respuesta").ToString()
0079                                    btnAceptar.Enabled = True
0080                                    txtRespuesta.Enabled = True
0081                                    txtRespuesta.Focus()
0082                                Else
0083                                    hdnMsjError.Value = "NO SE PUEDE RECUPERAR LA CLAVE, EL " + _
0084                                        "USUARIO NO TIENE REGISTRADA NINGUNA PREGUNTA " + _
0085                                        "SECRETA"
0086                                    vrLOcleConexion.Close()
0087                                End If
0088                            Else
0089                                hdnMsjError.Value = "LA INFORMACIÓN INDICADA, NO ESTÁ " + _
0090                                    "REGISTRADA EN EL SISTEMA"
0091                                txtUsuario.Focus()
0092                            End If
```

Continuación de la figura A-35.

```
0093                    Case 2
0094                        hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES VÁLIDA"
0095                        vrLOcleConexion.Close()
0096                    Case 3
0097                        hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0098                        vrLOcleConexion.Close()
0099                    Case Else
0100                        'Error que viene de la librería
0101                End Select
0102                vrLOcleConexion.Close()
0103            Else
0104                hdnMsjError.Value = "LA DIRECCIÓN DE CORREO ELECTRÓNICO NO ES " + _
0105                     "VÁLIDA"
0106                txtEmail.Focus()
0107            End If
0108        Else
0109            hdnMsjError.Value = "TODOS LOS CAMPOS SON OBLIGATORIOS"
0110            txtUsuario.Focus()
0111        End If
0112    Catch pVExExcepcion As Exception
0113        hdnMsjError.Value = pVExExcepcion.ToString()
0114        txtUsuario.Focus()
0115    End Try
0116 End Sub
0117
0118 'Si se presiona el botón ACEPTAR, si los datos ingresados son válidos
0119 'se reinicia la contraseña
0120 Private Sub pcPrClicAceptar(ByVal pVObjTipoObjeto As Object, _
0121         ByVal pVEvtEvento As System.EventArgs) Handles btnAceptar.Click
0122     Try
0123         txtRespuesta.Text = txtRespuesta.Text.Trim()
0124         txtUsuario.Text = txtUsuario.Text.ToUpper().Trim()
0125         txtEmail.Text = txtEmail.Text.ToUpper().Trim()
0126
0127         'Validar que no estén vacíos
0128         If (txtUsuario.Text <> "" And txtEmail.Text <> "" And txtRespuesta.Text <> "") Then
0129             'Validar la dirección de correo electrónico
0130             If (modHerr.fPbBolValidarEmail(txtEmail.Text) = True) Then
0131                 'Validar que la respuesta sea la que está registrada
0132                 If (hdnRespuesta.Value = txtRespuesta.Text) Then
0133                     Dim vrLTblDatos As DataTable = New DataTable()
0134                     Dim vrLsIntResultado As Int16 = 0
0135                     Dim vrLvNuevaClave(1) As Object
0136
0137                     'Conectarse a Oracle
0138                     Dim vrLOcleConexion As OracleConnection
0139                     vrLOcleConexion = modHerr.fPbOcleConectar()
0140
0141                     'Consultar en la BDD si el usuario y el email coinciden
0142                     vrLTblDatos = modHerr.fPbTblBuscarUsuario(-1, txtUsuario.Text, _
0143                         "", txtEmail.Text, 0, vrLOcleConexion, _
0144                         vrLsIntResultado, hdnMsjError)
0145
```

Continuación de la figura A-35.

```
0146                Select Case vrLsIntResultado
0147                   Case 0
0148                      hdnMsjError.Value = "NO SE PUEDE RECUPERAR LA CLAVE, LA " + _
0149                            "INFORMACIÓN INDICADA NO ESTÁ REGISTRADA EN EL SISTEMA"
0150                   Case 1
0151                      'Validar si la consulta devolvió datos
0152                      If (vrLTblDatos.Rows.Count = 1) Then
0153                         'Obtener la nueva clave
0154                         vrLvNuevaClave = modHerr.fPbVGenerarClave(txtUsuario.Text)
0155                         If (vrLvNuevaClave(0) <> "" And vrLvNuevaClave(1) <> "") Then
0156                            'Modificar los datos en la BDD
0157                            vrLsIntResultado = 0
0158                            modHerr.pcPbModificarUsuario(Convert.ToInt32( _
0159                               vrLTblDatos.Rows(0)("Id")), _
0160                               vrLTblDatos.Rows(0)("Usuario").ToString(), _
0161                               vrLvNuevaClave(1), _
0162                               vrLTblDatos.Rows(0)("Nombres").ToString(), _
0163                               vrLTblDatos.Rows(0)("Apellidos").ToString(), _
0164                               vrLTblDatos.Rows(0)("EMail").ToString(), _
0165                               "", "", "", "", "", 0, _
0166                               vrLOcleConexion, vrLsIntResultado, hdnMsjError)
0167
0168                            'Si no hubo error al actualizar la clave en la BDD
0169                            Select Case vrLsIntResultado
0170                               Case 0
0171                                  hdnMsjError.Value = "NO SE REALIZÓ LA ACTUALIZACIÓN " + _
0172                                        "DE LA CONTRASEÑA, EL USUARIO NO EXISTE"
0173                               Case 1
0174                                  'Enviar el correo con la nueva contraseña
0175                                  If (modHerr.fPbBolEnviarEmail( _
0176                                     vrLTblDatos.Rows(0)("EMail").ToString(), _
0177                                     vrLTblDatos.Rows(0)("Usuario").ToString(), _
0178                                     vrLvNuevaClave(0), 2, hdnMsjError) = True) Then
0179                                     hdnMsjError.Value = "LA NUEVA CONTRASEÑA FUE " + _
0180                                           "ENVIADA A LA DIRECCIÓN DE CORREO " + _
0181                                           "REGISTRADA EN EL SISTEMA."
0182                                  Else
0183                                     hdnMsjError.Value = "SE ACTUALIZÓ LA CONTRASEÑA, " + _
0184                                           "PERO HUBO INCONVENIENTE AL ENVIAR LAS " + _
0185                                           "NUEVAS CREDENCIALES A LA CUENTA DE " + _
0186                                           "CORREO REGISTRADA"
0187                                  End If
0188                                  'Indica que se debe redireccionar a la página de inicio
0189                                  hdnRedireccionar.Value = "1"
0190                               Case 3
0191                                  hdnMsjError.Value = "NO SE REALIZÓ LA ACTUALIZACIÓN " + _
0192                                        "DE LA CONTRASEÑA, HUBO UN ERROR EN LA BASE " + _
0193                                        "DE DATOS"
0194                               Case Else
0195                                  'Error que viene de la librería
0196                            End Select
0197                         Else
0198                            hdnMsjError.Value = "NO SE PUDO GENERAR UNA NUEVA " + _
```

Continuación de la figura A-35.

```
0199                                    "CONTRASEÑA"
0200                                End If
0201                            Else
0202                                hdnMsjError.Value = "LA INFORMACIÓN INDICADA, NO ESTÁ " + _
0203                                    "REGISTRADA EN EL SISTEMA"
0204                                End If
0205                            Case 2
0206                                hdnMsjError.Value = "LA OPCIÓN DE BÚSQUEDA INDICADA NO ES " + _
0207                                    "VÁLIDA"
0208                            Case 3
0209                                hdnMsjError.Value = "HUBO UN ERROR EN LA BASE DE DATOS"
0210                            Case Else
0211                                'Error que viene de la librería
0212                        End Select
0213
0214                        vrLOcleConexion.Close()
0215                    Else
0216                        hdnMsjError.Value = "LA RESPUESTA NO ES VÁLIDA"
0217                        txtRespuesta.Focus()
0218                    End If
0219                Else
0220                    hdnMsjError.Value = "LA DIRECCIÓN DE CORREO ELECTRÓNICO NO " + _
0221                        "ES VÁLIDA"
0222                End If
0223            Else
0224                hdnMsjError.Value = "TODOS LOS CAMPOS SON OBLIGATORIOS"
0225                txtUsuario.Focus()
0226            End If
0227        Catch pVExExcepcion As Exception
0228            hdnMsjError.Value = pVExExcepcion.ToString()
0229            txtUsuario.Focus()
0230        End Try
0231    End Sub
0232
0233    'Si se presiona el botón CANCELAR, se limpia el texto ingresado en los controles
0234    Private Sub pcPrClicCancelar(ByVal pVObjTipoObjeto As Object, _
0235            ByVal pVEvtEvento As System.EventArgs) Handles btnCancelar.Click
0236        'Indica que no se debe redireccionar a la página principal
0237        hdnRedireccionar.Value = "0"
0238
0239        txtUsuario.Text = ""
0240        txtEmail.Text = ""
0241        txtRespuesta.Text = ""
0242        lblPregunta.Text = "Pregunta secreta:"
0243        btnAceptar.Enabled = False
0244        txtRespuesta.Enabled = False
0245        txtUsuario.Focus()
0246    End Sub
0247 End Class
```

Fuente: elaboración propia.

En el siguiente apartado se muestra la implementación del servicio *web*, el cual es utilizado en frmDatosUsuario, por medio de la función nombrada fPbTblBuscarPreguntas que está definida en libHerramientas.1.0.vb.

1.3. Servicio *web*

Este servicio *web* se conecta a la base de datos implementada en el SGBD Oracle, y consulta los registros almacenados en tbl_seg_pregunta, invocando al procedimiento almacenado pc_buscarpreguntas.

Las preguntas secretas se devuelven en la tabla 0 de un dataset, en donde la primera fila de la tabla contiene el código y cadena del mensaje de error o de éxito al buscar las preguntas en la base de datos.

Figura A-36. **Procedimiento pc_buscarpreguntas**

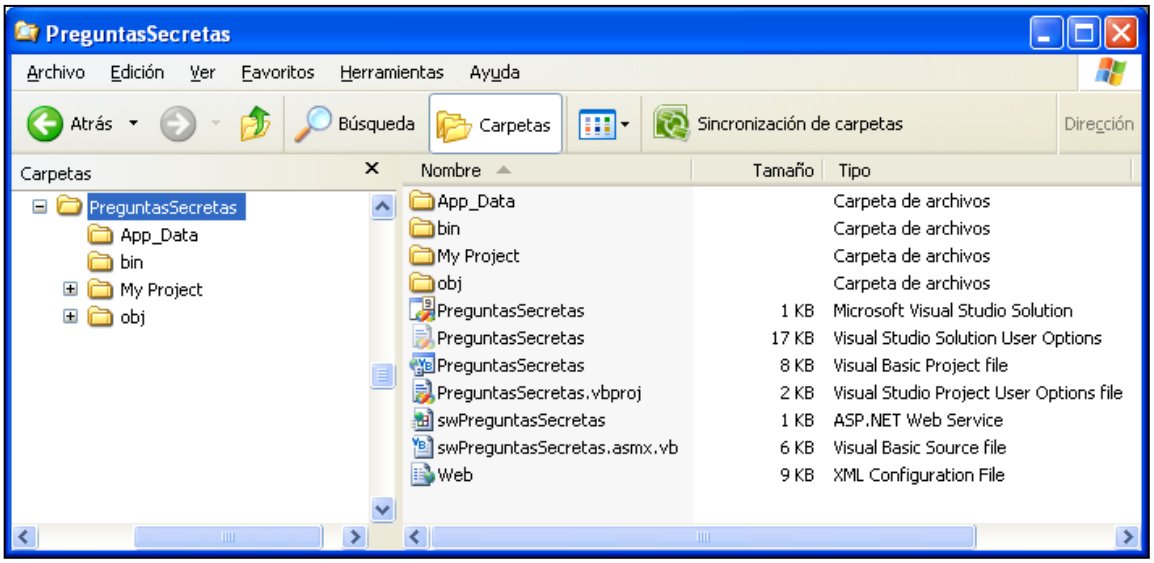
```
0001  CREATE OR REPLACE PROCEDURE pc_buscarpreguntas (pr_nm_resultado OUT NUMBER,
0002              pr_crs_preguntas OUT SYS_REFCURSOR)
0003  AS vr_l_nm_resultado NUMBER := 0;
0004  BEGIN
0005    --Busca las preguntas a mostrarle al usuario
0006    SELECT COUNT(*)
0007    INTO vr_l_nm_resultado
0008    FROM tbl_seg_pregunta;
0009
0010    IF (vr_l_nm_resultado > 0) THEN
0011      OPEN pr_crs_preguntas FOR
0012      SELECT Pregunta.nm_pregunta As ID,
0013             Pregunta.nvch2_pregunta As PREGUNTA
0014      FROM tbl_seg_pregunta Pregunta
0015      ORDER BY ID;
0016
0017      pr_nm_resultado := 1; --Sí hay datos
0018    ELSE
0019      pr_nm_resultado := 0; --No hay datos
0020    END IF;
0021  EXCEPTION
0022    WHEN OTHERS THEN
0023      pr_nm_resultado := 3;
0024  END;
```

Fuente: elaboración propia.

En el servicio *web* las credenciales de acceso a la base de datos tampoco están cifradas ni ocultas, pero se recomienda que en todo sistema estos datos estén cifrados o bien almacenados en algún lugar de acceso restringido.

Dentro del folder de este proyecto, se tiene solamente el código del servicio *web*, como se muestra en la siguiente figura.

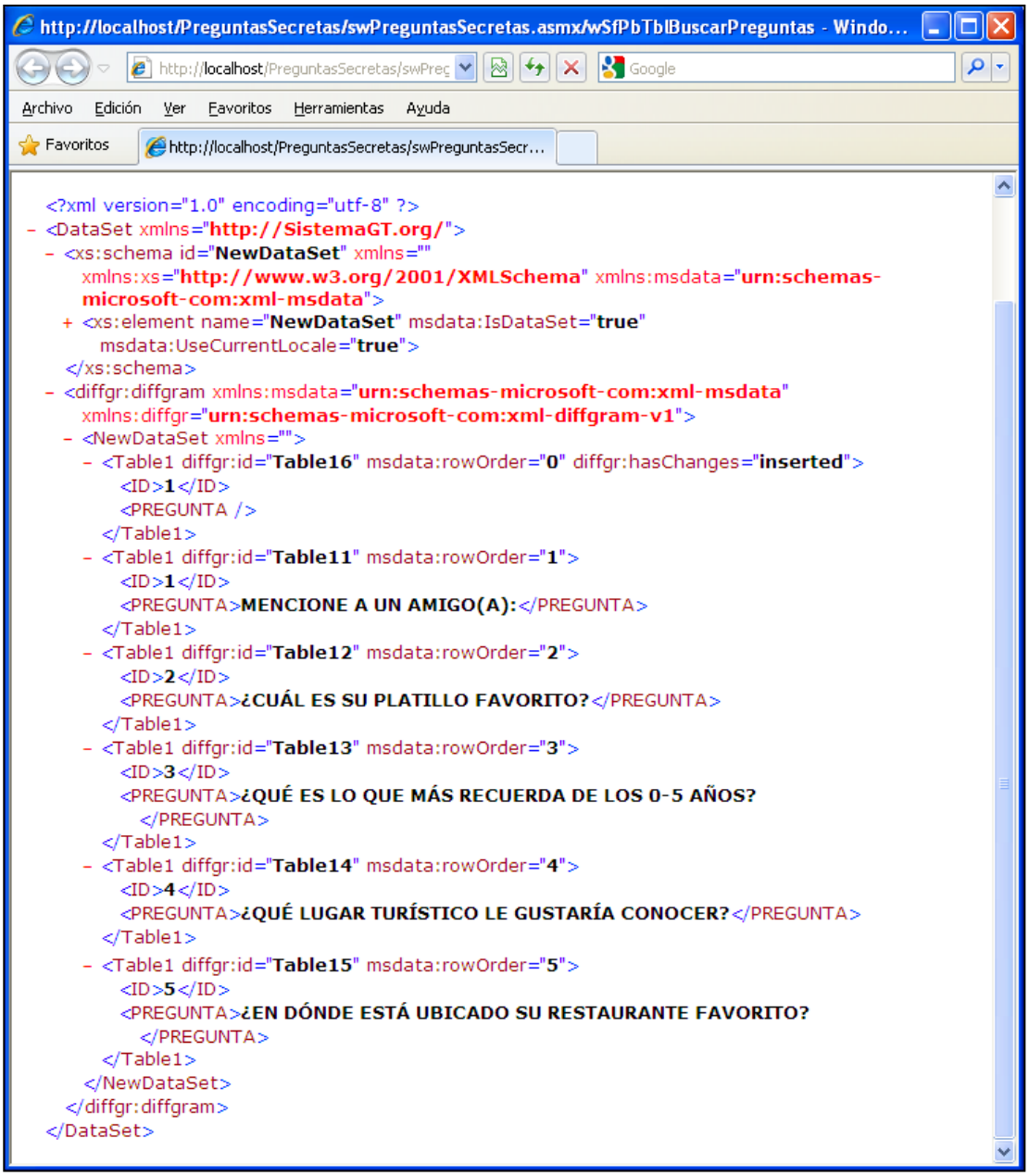
Figura A-37. **Carpeta del proyecto PreguntasSecretas**



Fuente: elaboración propia.

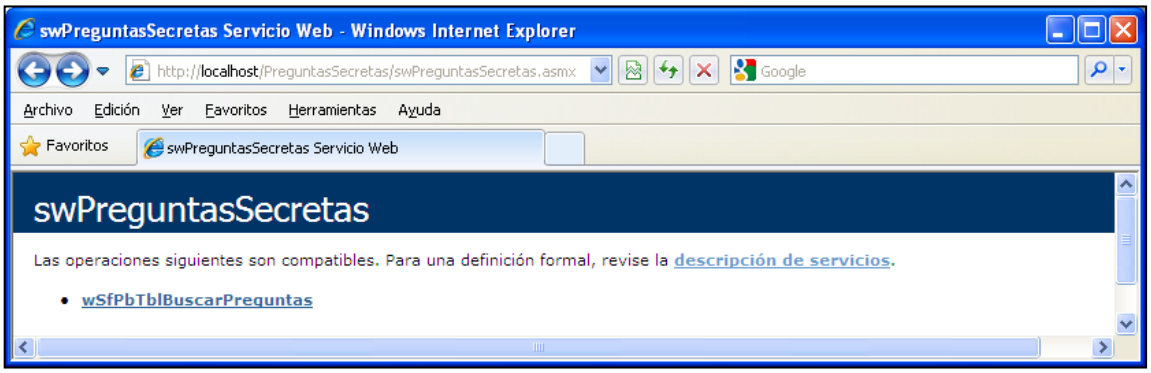
Al invocar el servicio *web* desde un navegador, se visualiza lo siguiente:

Figura A-38. **Definición y datos que retorna el servicio *web* PreguntasSecretas, al ser invocado desde el navegador *web* Internet Explorer 8.0**



Fuente: elaboración propia.

Figura A-39. **Link para invocar al servicio *web* PreguntasSecretas desde el navegador *web* Internet Explorer 8.0**



Fuente: elaboración propia.

Por último, se muestra en la figura A-40 el código del servicio *web*.

Figura A-40. **Código de swPreguntasSecretas.asmx.vb**

```
0001  /*****************************************************************************************************/
0002  /*   Archivo:                      swPreguntasSecretas.asmx.vb                                       */
0003  /*   Descripción:                  Servicio web para gestionar las preguntas                         */
0004  /*                                 secretas.                                                         */
0005  /*   Autor:                        Norma M. Chonay V.                                                */
0006  /*   Fecha de creación:            21/04/2011                                                        */
0007  /*   Fecha de última modificación: 21/04/2011                                                        */
0008  /*   Autor de última modificación: Norma M. Chonay V.                                                */
0009  /*   Versión:                      1.1                                                               */
0010  /*   Prerrequisitos:               Ninguno                                                           */
0011  /*****************************************************************************************************/
0012
0013  Imports Oracle.DataAccess.Client
0014  Imports Oracle.DataAccess.Types
0015
0016  Imports System.Web.Services
0017  Imports System.Web.Services.Protocols
0018  Imports System.ComponentModel
0019
0020  ' Para permitir que se llame a este servicio web desde un script, usando ASP.NET AJAX,
0021  'quite la marca de comentario de la siguiente línea.
0022  ' <System.Web.Script.Services.ScriptService()> _
0023  <System.Web.Services.WebService(Namespace:="http://SistemaGT.org/")> _
0024  <System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
```

Continuación de la figura A-40.

```
0025  <ToolboxItem(False)> _
0026  Public Class swPreguntasSecretas
0027      Inherits System.Web.Services.WebService
0028
0029      '/*************************************************************************************************/
0030      '/************************************** FUNCIONES ************************************************/
0031      '/*************************************************************************************************/
0032      'Función que devuelve la conexión a la BDD
0033      Private Function fPrOcleConectar() As OracleConnection
0034          Dim vrLOcleConexion As New OracleConnection
0035
0036          vrLOcleConexion.ConnectionString = "Data source=nchonay; User id=usr_seg; " + _
0037              "Password=anionuev0;"
0038          vrLOcleConexion.Open()
0039          Return vrLOcleConexion
0040      End Function
0041
0042      '/*************************************************************************************************/
0043      '/************************************** SERVICIO WEB *********************************************/
0044      '/*************************************************************************************************/
0045      'Función para buscar las preguntas
0046      <WebMethod()> _
0047      Public Function wSfPbTblBuscarPreguntas() As DataSet
0048          'Valor de retorno
0049          Dim rtTblsDatos As DataSet = New DataSet()
0050
0051          'Tabla con resultados
0052          Dim vrLTblDatos As DataTable = New DataTable()
0053          Dim vrLColColumna As DataColumn
0054          'Columna para el ID de la pregunta
0055          vrLColColumna = New DataColumn()
0056          vrLColColumna.DataType = System.Type.GetType("System.Int16")
0057          vrLColColumna.ColumnName = "ID"
0058          vrLTblDatos.Columns.Add(vrLColColumna)
0059          'Columna para el TEXTO de la pregunta
0060          vrLColColumna = New DataColumn()
0061          vrLColColumna.DataType = Type.GetType("System.String")
0062          vrLColColumna.ColumnName = "PREGUNTA"
0063          vrLTblDatos.Columns.Add(vrLColColumna)
0064
0065          'Inicializar el código y mensaje de resultado
0066          Dim vrLFilaMensaje As DataRow = vrLTblDatos.NewRow()
0067          vrLFilaMensaje("ID") = 10
0068          vrLFilaMensaje("PREGUNTA") = ""
0069
0070          Try
0071              'Conectarse a Oracle
0072              Dim vrLOcleConexion As OracleConnection
0073              vrLOcleConexion = fPrOcleConectar()
0074
0075              'Validar si se pudo establecer la conexión con Oracle
0076              If Not (vrLOcleConexion Is Nothing) Then
0077                  Dim vrLOcleComando As OracleCommand = New OracleCommand()
```

Continuación de la figura A-40.

```
0078            Dim vrLOcleAdaptador As OracleDataAdapter
0079
0080            'Indicar el nombre del procedimiento almacenado
0081            vrLOcleComando.CommandType = CommandType.StoredProcedure
0082            vrLOcleComando.CommandText = "pc_buscarpreguntas"
0083
0084            'Pasar los parámetros
0085            vrLOcleComando.Parameters.Clear()
0086            vrLOcleComando.Parameters.Add("pr_nm_resultado", OracleDbType.Int16, _
0087                                ParameterDirection.Output)
0088            vrLOcleComando.Parameters.Add("pr_crs_preguntas", OracleDbType.RefCursor, _
0089                                ParameterDirection.Output)
0090
0091            'Ejecutar la consulta
0092            vrLOcleComando.Connection = vrLOcleConexion
0093            vrLOcleAdaptador = New OracleDataAdapter(vrLOcleComando)
0094            vrLOcleAdaptador.Fill(vrLTblDatos)
0095            vrLFilaMensaje("ID") = Convert.ToInt16( _
0096                vrLOcleComando.Parameters("pr_nm_resultado").Value)
0097          Else
0098            vrLFilaMensaje("PREGUNTA") = "NO SE PUDO ESTABLECER CONEXIÓN A LA " + _
0099                "BASE DE DATOS"
0100          End If
0101          vrLOcleConexion.Close()
0102          vrLOcleConexion.Dispose()
0103        Catch pVExExcepcion As Exception
0104          vrLFilaMensaje("ID") = 10
0105          vrLFilaMensaje("PREGUNTA") = pVExExcepcion.ToString()
0106        End Try
0107
0108        'Devolver los datos del error
0109        vrLTblDatos.Rows.InsertAt(vrLFilaMensaje, 0)
0110
0111        rtTblsDatos.Tables.Add(vrLTblDatos)
0112        Return rtTblsDatos
0113      End Function
0114    End Class
```

Fuente: elaboración propia.