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

APLICACIÓN MÓVIL: CONSULTA DE ESTADO GENERAL DE FINCAS REGISTRO DE LA PROPIEDAD

Ángel Salvador Ayala Ochoa
Andrea Adriana Grimaldi Santos
Asesorado por el Ing. Ricardo Israel Mazariegos Castillo

Guatemala, agosto de 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

APLICACIÓN MÓVIL: CONSULTA DE ESTADO GENERAL DE FINCAS REGISTRO DE LA PROPIEDAD

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ÁNGEL SALVADOR AYALA OCHOA
ANDREA ADRIANA GRIMALDI SANTOS
ASESORADO POR EL ING. RICARDO ISRAEL MAZARIEGOS CASTILLO

AL CONFERÍRSELES EL TÍTULO DE

INGENIEROS EN CIENCIAS Y SISTEMAS

GUATEMALA, AGOSTO DE 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Raúl Eduardo Ticún Córdova
VOCAL V	Br. Henry Fernando Duarte García
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Angel Roberto Sic García (a. i.)
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. José Ricardo Morales Prado
EXAMINADOR	Ing. Roberto Estuardo Ruiz Cruz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

Ángel Salvador Ayala Ochoa

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Raúl Eduardo Ticún Córdova
VOCAL V	Br. Henry Fernando Duarte García
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Sergio Arnaldo Méndez Aguilar
EXAMINADOR	Ing. William Estuardo Escobar Argueta
SECRETARIA	Inga. Lesbia Magalí Herrera López

Andrea Adriana Grimaldi Santos

HONORABLE TRIBUNAL EXAMINADOR

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

APLICACIÓN MÓVIL: CONSULTA DE ESTADO GENERAL DE FINCAS REGISTRO DE LA PROPIEDAD

Tema que nos fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 11 de octubre de 2014.

Ángel Salvador Ayala Ochoa **Andrea Adriana Grimaldi Santos**

Guatemala, 24 de septiembre de 2014

Estimado Ing. Santos,

Deseándole éxitos en sus actividades diarias, confirmo por este medio que el trabajo de graduación titulado **"Aplicación móvil: Consulta de Estado General de Fincas Registro de la Propiedad"**, realizado por los alumnus **"Ángel Salvador Ayala Ochoa**, con carnet **200714403** y **Andrea Adriana Grimaldi Santos**, con carnet **200611206** ha sido revisado y aprobado por mí en el marco del curso Seminario de Investigación, llevado a cabo durante el primer semestre del año en curso.

Sin otro particular le saluda,

Ing. Ricardo Mazariegos

Ricardo Israel Mazariegos Castillo
Ingeniero en Ciencias y Sistemas
Colegiado No.: 12451

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

Guatemala, 15 de Octubre de 2014

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 los estudiantes **ÁNGEL SALVADOR AYALA OCHOA** con carné **2007-14403, y ANDREA ADRIANA GRIMALDI SANTOS** con carné **2006-11206,** titulado: **"APLICACIÓN MÓVIL: CONSULTA DE ESTADO GENERAL DE FINCAS REGISTRO DE LA PROPIEDAD",** 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

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24188000 Ext. 1534

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, del trabajo de graduación ***"APLICACIÓN MÓVIL: CONSULTA DE ESTADO GENERAL DE FINCAS REGISTRO DE LA PROPIEDAD"****, realizado por los estudiantes, ÁNGEL SALVADOR AYALA OCHOA y ANDREA ADRIANA GRIMALDI SANTOS, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

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

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 01 de agosto de 2016

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.357-2015

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: **APLICACIÓN MÓVIL: CONSULTA DE ESTADO GENERAL DE FINCAS REGISTRO DE LA PROPIEDAD,** presentado por los estudiantes universitarios: **Ángel Salvador Ayala Ochoa y Andrea Adriana Grimaldi Santos,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

FACULTAD DE INGENIERIA
DECANO

Guatemala, agosto de 2016

/cc

ACTO QUE DEDICO A:

Mis padres Edwin Salvador Ayala y Telma Maritza, por su apoyo, y mi formación durante mi niñez.

Mi hermano José Manuel Ayala, por haberme dado tanto cariño e inspirarme a ser un ejemplo para él.

Ángel Salvador Ayala Ochoa

AGRADECIMIENTOS A:

Mis padres	Por ayudarme con su ejemplo a ser la persona que ahora soy y enseñarme el valor del trabajo duro.
MI hermano	José Manuel Ayala, por darme esa alegría siempre que la necesite.
Mis amigos	Omar Vides, Juan Rodríguez, Luis Fernando Valdez, Luis Fernando Leal, Oscar Arévalo, José Hernández, Julio González y Gerson Estrada, por alegrar mis días durante mi formación en esta casa de estudios y siempre apoyarme durante mis estudios.
MIs amigos	Omar Morales, José Javier Cardona, Juan Andrés Cardona, Jorge Letrán, Alejandra Canahuí, Alex Muñoz, por enseñarme cada día a estar preparado a lo que sea que la vida me traiga.
Ingeniero Renato Ponciano	Por motivarme el primer día de clases a graduarme de esta universidad.

Ingeniero César Villela, Ingeniero René Orneliz y licenciado César A. Fernández F.

Por sus enseñanzas excelentes, ser un ejemplo con su docencia y en el ámbito laboral.

Ángel Salvador Ayala Ochoa

ACTO QUE DEDICO A:

Mis padres Carol Mario Grimaldi Herrera y Orquídea Janette Santos Cardona, por su apoyo incondicional, su guía y su luz.

Mi hermana Nina Fabella Grimaldi Santos, por aconsejarme y acompañarme en las noches de desvelo, en las decepciones, en las alegrías y en la vida.

Mi tia Adriana Lina Ethel Grimaldi Herrera, por ser un ejemplo de constancia y disciplina.

Andrea Adriana Grimaldi Santos

AGRADECIMIENTOS A:

Mis padres Mario Grimaldi y Jannette Santos, por darme la vida, confiar en mis capacidades y enseñarme a labrar mi camino.

Mik hermana Nina Grimaldi, por mostrarme la fuerza que ahora forma parte de mí.

Mi tia Adriana Grimaldi, por ser la mejor catedrática que conozco.

Mis amigas Ana Leslie Gálvez, Lucía Duarte, Martha Hernández y Linda Amézquita, por mostrarme que la esencia de las personas no es transgedida por el tiempo.

Mis amigos Luis Fernando López, Luis Felipe Méndez, Jacqui Hernández, Walter Cardona, Julio González y Gerson Estrada, por las alegrías e inspirarme con su trabajo y determinación.

MIs amigos Ángel Ayala, Omar Morales, Jorge Letrán, Alejandra Canahuí, José Javier Cardona, Alex Muñoz y, en especial, Juan Andrés Cardona, por aceptarme como soy y permitirme crecer junto a ellos.

Ing. Soraya Martinez	Por darme la oportunidad de experimentar por primera vez la belleza de la docencia.
Ing. Ricardo Morales, Ing. Rene Orneliz e Ing. Otto Rodriguez	Por sus excelentes cátedras.
Mis catedraticos y Maestros	por todas las experiencias vividas y por su tiempo.

Andrea Adriana Grimaldi Santos

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

LISTADO DE SIMBOLOS

Simbolo	Significado
$	Dolar

GLOSARIO

Acoplamiento Grado en el que un módulo depende de otro.

Alta utilidad Facultad de una herramienta de poder usarse para lograr un fin que agregue valor a quien la use.

Análisis sintáctico Análisis de un conjunto de lexemas en función de su concordancia y cumplimiento con una gramática de un lenguaje determinado.

Aplicación Programa escrito en un lenguaje de computación determinado.

Clase Conjunto de funciones y variables que describen el funcionamiento y características comunes de los objetos que pertenecen a ella.

Débilmente acoplado Tipo de acoplamiento que describe cómo múltiples sistemas pueden juntarse para realizar transacciones, sin importar los componentes de hardware y software utilizados. De esta forma, se reduce el riesgo que un cambio hecho en un elemento pueda provocar cambios inesperados en otro elemento.

Gramática Unidad mínima con significado léxico que no representa morfemas gramaticales.

Instancia	Realización específica de una clase.
Kernel	Parte esencial del sistema operativo que contiene los componentes básicos de un sistema. Gestiona recursos de software y acceso al hardware.
Lexema	Unidad mínima con significado léxico que no representa morfemas gramaticales.
Librería	Colección de rutinas precompiladas lista para su inmediato uso.
Módulo	Unidad separada de hardware o software que contiene funciones conceptualmente relacionadas.
Objeto	Instancia de una clase, siendo esta un grupo de variables, funciones o estructuras de datos.
Paquete	Agrupación de clases asociadas.
Patrón	Formalización de la relación entre un problema y una solución, usado para hacer una decisión orientada al diseño arquitectónico de un sistema.
Protocolo	Conjunto de reglas para el intercambio de datos entre dos componentes de software o hardware.
Servicio	Componente de aplicación que puede ejecutar operaciones y no cuenta con una interfaz gráfica.

Sistema operativo	Software básico que provee una interfaz entre el usuario, programas y hardware. Administra los recursos de software y hardware.

RESUMEN

La tecnología ha pasado de ser un punto de apoyo para la resolución de problemas, a formar una simbiosis con la cotidianidad del hombre. Debido a la alta disponibilidad tecnológica, en particular de teléfonos móviles, el nacimiento de aplicaciones que faciliten las actividades diarias es constante y casi abrumador.

Como parte de la responsabilidad social que debería practicar un desarrollador de aplicaciones, se describe en este documento una aplicación móvil que facilitará el acceso a la información a toda persona que desee realizar consultas sobre su propiedad con base en registros oficiales, de una forma fácil e intuitiva.

Para lograr este fin, se aprovecharán los servicios web ya disponibles, a través de los cuales se puede acceder a información sobre las propiedades registradas legalmente.

Este documento está dividido en cuatro capítulos: el primero describe la solución y la teoría bajo la cual se estudiará el problema y los resultados de la aplicación de la solución; en el segundo se encuentran los antecedentes, la especificación del mercado objetivo y un *benchmarking* de la aplicación; en el tercero se expone la descripción del prototipo, las respectivas validaciones y el cuarto capítulo contiene la documentación de la solución y las consideraciones de la aplicación.

OBJETIVOS

General

Desarrollar e implementar una herramienta tecnológica móvil para facilitar el acceso a la información sobre las propiedades, haciendo valer el derecho de las personas a acceder libremente a la información en poder de administración pública.

Específicos

1. Construir un canal de acceso a la información proveída por el Registro General de la Propiedad en el marco de la Ley de Libre Acceso a la Información.

2. Identificar los elementos arquitectónicos necesarios para implementar una solución accesible a través de teléfonos móviles.

3. Realizar una aplicación móvil de alta utilidad para dispositivos Android que facilite la consulta de información sobre propiedades.

4. Diseñar y construir tres módulos a través de los cuales se pueda verificar y visualizar el estado de la propiedad, saldo del propietario y documentos en trámite relacionados con una propiedad.

INTRODUCCIÓN

Como parte de las necesidades fundamentales que todo Gobierno tiene, el general Justo Rufino Barrios fundó, en 1877, el Registro General de la Propiedad. La sede del Registro General de la Propiedad se encuentra desde 1976 en la 9ª avenida 14-25 de la zona 1 de la capital de Guatemala, edificio que con anterioridad fuese la Corte Suprema de Justicia.

Desde entonces, el crecimiento de los registros ha correspondido al aumento de la población y de las adquisiciones. Sin embargo, la forma de llevar los registros de las propiedades era obsoleta hasta fechas relativamente recientes. Anteriormente, los registros eran trabajados de forma manual, las respuestas a los procesos eran lentas y el proceso de recepción, presentación, operación y respuesta a peticiones era un proceso largo e ineficiente.

El Registro le dio la bienvenida al 2004 con un profundo proceso de reestructuración de instalaciones, infraestructura, equipo tecnológico y procesos, implementando así sistemas electrónicos de almacenamiento, análisis y búsqueda.

Como parte del sistema tecnológico que soporta las operaciones del Registro General de la Propiedad, se desarrolló un sistema de búsqueda ágil de estatus de fincas y documentos relacionados a ellas, usando tecnologías basadas en estándares y de alta eficiencia, el cual se describe a detalle en el presente documento.

1. ESTUDIO DE LA TECNOLOGÍA Y SU IMPACTO EN GUATEMALA

El Registro General de la Propiedad de Guatemala es una institución pública encargada de administrar los registros de propiedad de los habitantes en apego con la Constitución Política de la República.

Según el decreto 57-2008, la Ley del Libre Acceso a la Información Pública, toda persona interesada, sin discriminación alguna, tiene el derecho a solicitar y a tener acceso a la información pública en posesión de las autoridades. En pos de esto, una persona que desee conocer datos sobre su propiedad, como el estado de un documento, su saldo o los documentos adjudicados a ella, puede dirigirse a las una de las oficinas del Registro General de la Propiedad y solicitar la información que desee.

Sin embargo, se desea una aplicación móvil que facilite la realización de estos trámites. De esta forma, el propietario tendrá acceso a la información sobre su propiedad al alcance de su mano, sin tener que regirse a horarios administrativos y traslado físico a alguna de las oficinas del Registro General de la Propiedad.

1.1. Teoría que soporta la investigación

El proyecto se fundamentará en la teoría unificada de aceptación y uso de la tecnología por sus siglas en inglés, UTAUT. Esta teoría ayudará a comprender a los usuarios en cuanto a la intención de uso de la aplicación y su comportamiento subsecuente.

Se analizarán los siguientes factores clave:

- Expectativa de funcionamiento
- Expectativa de esfuerzo
- Influencia social
- Condiciones facilitadoras

Estos factores se presentan como hipótesis y tienen un rol significativo como determinantes directos de la aceptación y el comportamiento del usuario. Las hipótesis son:

- Expectativa del funcionamiento: el grado en el que el usuario cree que le ayudará en su desempeño está relacionado con su género y edad, de tal forma que el efecto será más fuerte en los usuarios jóvenes, sin importar su género.

- Expectativa de esfuerzo: el grado en el que el usuario considera que el sistema tiene facilidad de uso está relacionado con su género, edad y experiencia, de tal forma que el efecto será más significativo entre los usuarios jóvenes, sin importar su género y con mediana experiencia en el uso de teléfonos móviles.

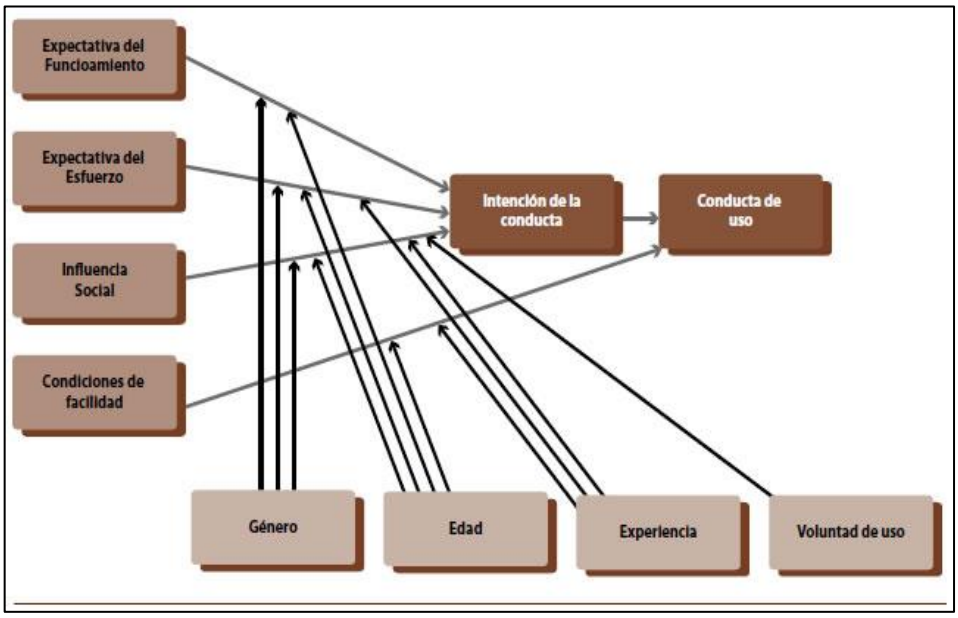
- Influencia social: el grado en el que el usuario percibe que las personas importantes en su vida creen que debería usar el sistema está relacionado con el género, edad, experiencia y voluntad de uso, de tal forma que el efecto será significativo en usuarios de todas las edades, sin importar su género, con mediana experiencia en teléfonos móviles y con mediana voluntad de uso.

- Condiciones de facilidad: el grado en el que el usuario percibe que la estructura técnica y organizacional es capaz de dar soporte al sistema se encuentra relacionado con la edad y la experiencia, de tal forma que el efecto será más fuerte en usuarios de todas las edades, y con poca o mediana experiencia.

Esta teoría ha sido seleccionada debido a que se ajusta las necesidades que el proyecto intenta cubrir. Para medir esos factores, se realizarán encuestas que los consideren, tomando en cuenta el género, edad, experiencia y voluntad de uso del encuestado. Estas encuestas se realizarán en línea.

La siguiente gráfica muestra la relación entre los factores considerados.

Figura 1. **Gráfico relacional UTAUT**



Fuente: ALCÁNTARA PILAR, Juan Miguel. *Modelización del comportamiento del consumidor online: el papel moderado de la cultura, el diseño wed y el idioma*. p. 62.

1.2. Teoría general sobre herramientas seleccionada

A lo largo de la historia, la metodología de elaborar los bienes y el concepto de calidad han ido evolucionando de una forma paralela.

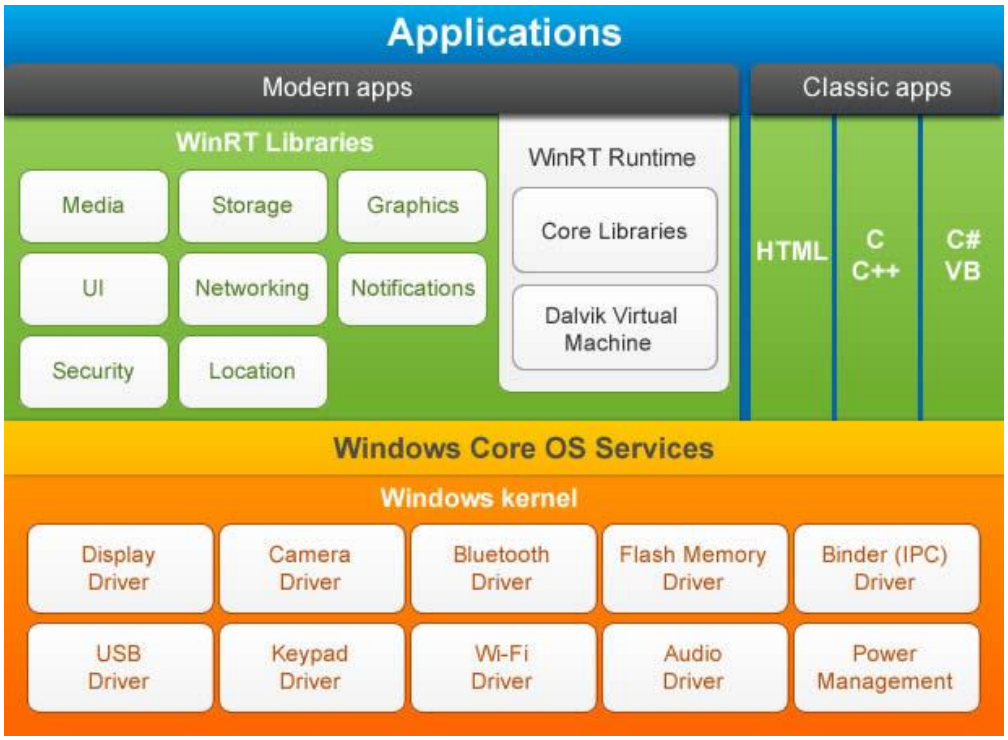
1.2.1. Aplicaciones móviles

Las aplicaciones móviles son herramientas que representan soluciones informáticas que caben en el bolsillo. La popularización de dispositivos móviles permite al usuario acceder a un enorme catálogo de aplicaciones que le facilitan sus tareas cotidianas.

1.2.2. Sistema operativo Android

La figura 2 describe la arquitectura de Android. Android OS puede ser analizado como una pila de diferentes capas de software, donde cada una de las capas es un grupo de diversos componentes de programa. Comportándose como una simbiosis, el sistema operativo, enlaces respectivos y aplicaciones importantes funcionan en conjunto. Cada capa provee de servicios diferentes a la capa que se encuentra justo arriba.

Figura 2. **Arquitectura de Android**



Fuente: *Aprendiendo sobre la arquitectura Android*. http://www.hermosaprogramacion.com/2014/08/aprendiendo-la-arquitectura-de-android/. Consulta: mayo de 2014.

La descripción de cada capa se resume de la siguiente forma:

- Aplicaciones: es la capa superior y en donde encajarán las aplicaciones. En este sentido, pueden utilizarse las diversas aplicaciones estándar que vienen preinstaladas con cada dispositivo, como el navegador, gestor de contactos, mensajería, entre otros. Con estas herramientas, el desarrollador puede escribir aplicaciones que utilicen o reemplacen las aplicaciones existentes.

- Entorno de aplicación: son bloques con los que interactúa una aplicación. Estos programas administran las funciones básicas como el manejo de los recursos del teléfono, gestión de llamadas, entre otros.

- Librerías: esta capa permite al dispositivo manejar diferentes tipos de datos. Estas librerías generalmente están escritas en C++ y son específicas para un hardware en particular. Algunas de las librerías más importantes son: SQLite, WebKit, OpenGL, entre otras.

- Linux *kernel*: es la capa más básica y representa el núcleo del sistema. Todo el sistema operativo Android está construido sobre un *kernel* con cambios arquitectónicos hechos finalmente por Google. Esta versión modificada es la que interactúa con el hardware y contiene controladores de hardware específicos, siendo estos los que se comunican directamente con el hardware. El *kernel* también actúa como una capa de abstracción entre el hardware y otras capas de software.

- Android Runtime: consiste en una máquina virtual Dalvik y de librerías Java. Dalvik es un tipo de máquina virtual Java que se utiliza en dispositivos Android para correr aplicaciones y es optimizada para bajo consumo y ambientes de pocos recursos de memoria.

1.2.3. Arquitectura orientada a servicios

La arquitectura orientada a servicios (SOA, por sus siglas en inglés), es un estilo arquitectónico y un paradigma cuya estructura se describe en términos de servicios. A través de ella, es posible enlazar el negocio completo como una serie de tareas repetitivas, también llamadas servicios.

Un servicio es una representación lógica de actividad de negocio que puede repetirse y que tiene una salida determinada (por ejemplo, consultar el estado de crédito de un cliente, información sobre el clima, entre otros). Está encapsulado y abstraído lógicamente y puede estar compuesto de otros servicios. Además, está construido de tal forma que representa una caja negra para quien consume el servicio.

Una arquitectura orientada a servicios es esencialmente una colección de servicios. Los servicios se comunican entre ellos. La comunicación puede involucrar el simple paso de datos o dos o más servicios coordinando alguna actividad. Esta interacción se establece de tal forma que una entidad realiza una unidad de trabajo en pos de la otra entidad. Cada interacción es autónoma y débilmente acoplada, de tal forma que las interacciones son independientes entre sí.

Este estilo arquitectónico está basado en el diseño de los servicios (los cuales reflejan actividades de negocio de la vida real), conteniendo procesos de negocio de la empresa. La representación de servicios usa descripciones de negocio para proveer contexto (por ejemplo: procesos de negocio, metas, reglas, políticas, interfaces y componentes de servicio) e implementa servicios usando orquestación de servicios. Requiere además una fuerte gobernanza de representación de servicio e implementación.

Algunas organizaciones notan beneficios significativos al usar SOA, como costos más bajos, más consistencia en las aplicaciones, mayor agilidad, encapsulamiento de tareas (lo que aísla los problemas y, por lo tanto, facilita su solución). La implementación de SOA reduce la redundancia e incrementa la usabilidad y el valor que se le aporta al cliente, produciendo sistemas interoperaciones y modulares más sencillos de usar y mantener.

1.2.4. Protocolo de acceso a objeto simple

El protocolo de acceso de objeto simple (o SOAP, por sus siglas en inglés) es un protocolo de comunicación usado por servicios web. La aplicación primaria de SOAP es comunicación entre aplicaciones. SOAP codifica y usa XML como un esquema de codificación parámetros para petición y respuesta que usa HTTP para fines de transporte.

SOAP cubre cuatro aspectos fundamentales:

- Un formato de mensaje para comunicación de una vía, describiendo cómo el mensaje puede ser empaquetado en un documento XML.

- Una descripción de como un mensaje SOAP puede ser transportado usando HTTP (para interacción web) o SMTP (para interacción basada en correo electrónico).

- Un conjunto de reglas que debe ser respetado cuando se procesa un mensaje SOAP y una clasificación simple de las entidades involucradas en el procesamiento de un mensaje SOAP.

- Un conjunto de convenciones que cómo volver una llamada RCP (llamada a procedimiento remoto, por sus siglas en inglés) en un mensaje SOAP y viceversa.

SOAP es un protocolo ligero que permite que las aplicaciones pasen mensajes y datos entre sistemas dispares y ambientes distribuidos, permitiendo invocaciones remotas. En este caso, SOAP es ligero debido a que posee únicamente dos propiedades fundamentales, a saber:

- Enviar y recibir paquetes con protocolo HTTP
- Procesar mensajes XML

2. MARCO METODOLÓGICO

2.1. Antecedentes

El Registro General de la Propiedad en Guatemala tiene sus orígenes gracias a la iniciativa del general Justo Rufino Barrios en 1877.

Desde 1976 el edificio del Registro se encuentra en la 9ª avenida 14-25 de la zona 1 de la ciudad de Guatemala. El Registro contaba con una infraestructura y sistema de operaciones que databa del siglo XIX. Hasta hace poco, el equipo de trabajo era obsoleto y las operaciones relacionadas con las fincas se realizaban de forma manual, haciendo el proceso de consulta bastante lento y la entrega de la información podía darse meses después de ser solicitada.

Luego de una serie de reformas estructurales e implementación de tecnologías de la información en 1996, el Registro General de la Propiedad cuenta con servicios más ágiles y eficientes. Se digitalizaron los libros y a partir de 2004 se revisan los procesos de forma integral permitiendo la agilización de los servicios.

2.2. Descripción del problema

Como parte del proceso de modernización de consulta de información sobre fincas y en conformidad con la Ley del Libre Acceso a la Información, se necesita una aplicación móvil que permita la verificación de información sobre las fincas. En la actualidad se cuenta únicamente con una página web para consultar ciertos datos.

2.3. Propuesta de valor

En la actualidad se cuenta con una página web en la cual se pueden realizar consultas sobre propiedades. Sin embargo, en esta aún no se puede verificar el estado de los documentos relacionados con la propiedad, ni el saldo del propietario. Es por ello que la propuesta de valor es una aplicación en la cual esta información esté al alcance a un par de toques, de forma inmediata.

2.4. Mercado objetivo

Ciudadano dueño de una propiedad con acceso a internet por medio de un dispositivo móvil.

2.5. Solución planteada

Una aplicación móvil para Android de alta utilidad, manejabilidad y de fácil mantenimiento. La aplicación consumirá los servicios web puestos a disponibilidad por el Registro General de la Propiedad.

2.5.1. Sistema operativo

Considerando los sistemas operativos más utilizados en el mercado guatemalteco, el diseño, desarrollo y pruebas de la aplicación Android se harán con la intención de garantizar un correcto funcionamiento en los siguientes sistemas operativos:

- Android 2.3.x (Gingerbread)
- Android 4.0 (Ice Cream Sandwich)
- Android 4.1 (Jelly Bean)

Considerando el sistema operativo soportado en la actualidad por Apple, el diseño, desarrollo y pruebas para la aplicación iOS se hará para iOS 7.

2.5.2. Módulos

- Bitácora del documento: módulo para verificar en qué estado se encuentra un documento relacionado con una finca.
- Consulta de recibos: módulo para verificar los datos correspondientes a un recibo.
- Documentos en trámite: módulo que muestra todos los documentos correspondientes a una finca.

2.5.3. *Benchmarking*

Se consideraron aplicaciones que basaban su modelo de negocio en servicios web que ponían información pública a disponibilidad de la población.

2.5.3.1. Public Record App

Aplicación destinada lograr un fácil acceso de los registros de la Corte Suprema de Justicia de EEUU, bajo el marco de la Ley de Libre Acceso a la Información de ese país.

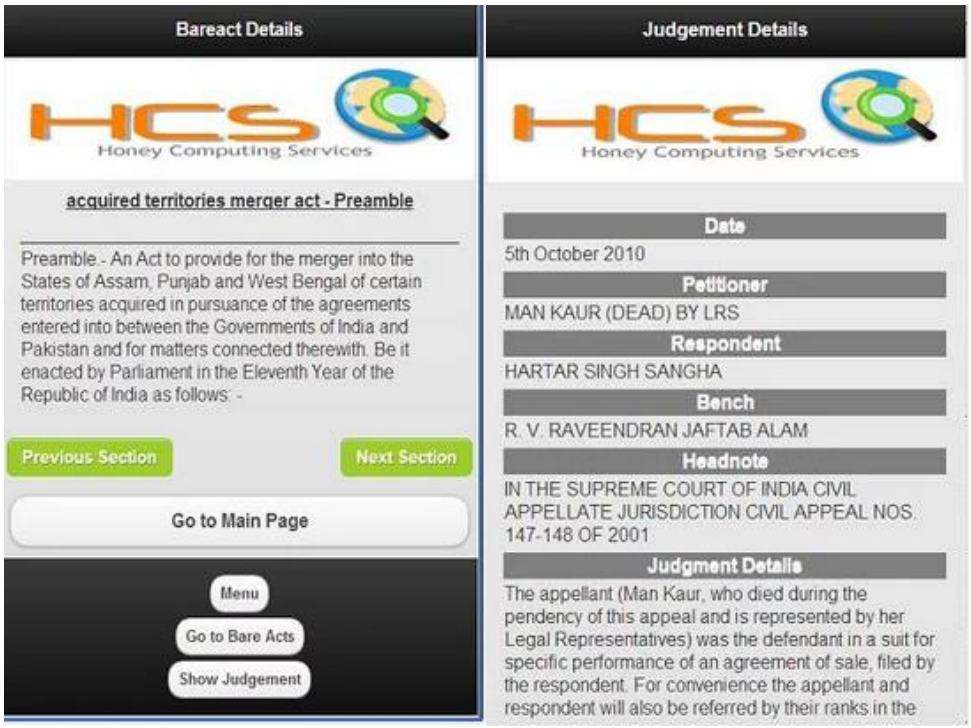
La aplicación es gratuita. Se puede acceder a un archivo de la Corte con el número de caso. Es posible acceder a aproximadamente un billón de registros que incluyen registros de propiedad, de nacimiento, matrimonios, divorcios, declaraciones de bancarrota, entre otros. Funciona para el sistema operativo Android desde su versión 2.1.

2.5.3.2. Supreme Court Cases

Esta aplicación está orientada a usuarios en la India. A través de ella se puede acceder a información sobre juicios en curso. La información es actualizada cada quince días. Los datos que la aplicación ofrece son, entre otros, día del juicio, nombre del demandante, nombre del demandado, citación e historial del curso de eventos del juicio.

Es posible realizar búsquedas por rango de años, mes, en orden de reciente a antiguo y viceversa. Es una aplicación de pago y tiene un costo de $ 11,99 hasta la fecha de hoy. Disponible para Android 2.2.

Figura 3. ***Benchmarking* - Supreme Court Cases**



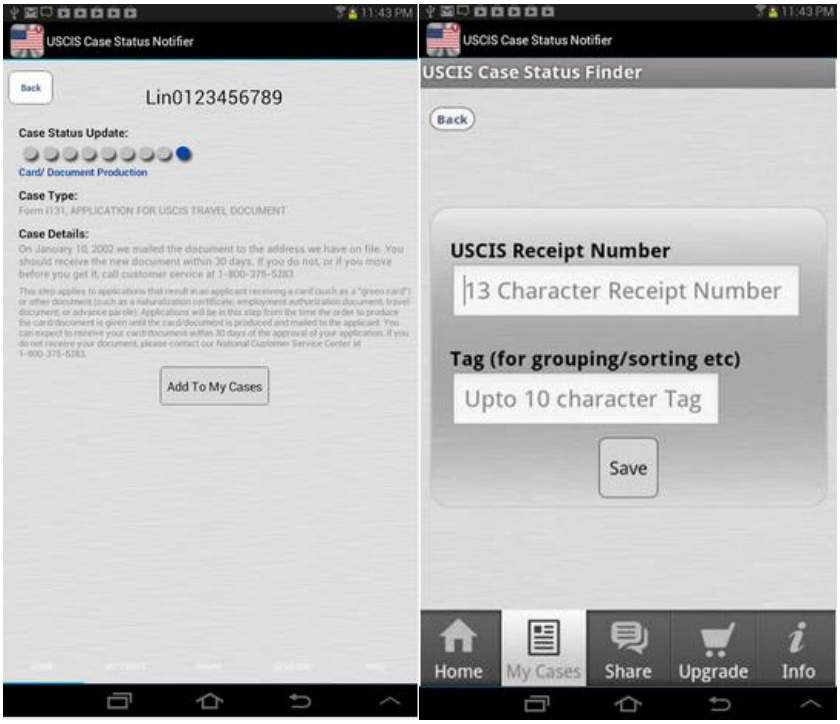
Fuente: elaboración propia, empleando HTML5.

2.5.3.3. USCIS Case Status Notifier

Aplicación a través de la cual puede accederse a casos de servicios de ciudadanía e immigración de Estados Unidos (USCIS, por sus siglas en inglés). La información puede ser accedida libremente en el marco de la Ley del Libre Acceso a la Información de Estados Unidos.

La aplicación puede configurarse para recibir alertas sobre un caso determinado, también es posible compartir información del caso con otra persona. Es una aplicación gratuita, disponible para dispositivos Android a partir de su versión 1.6.

Figura 4. ***Benchmarking* - USCIS Case Status Notifier**



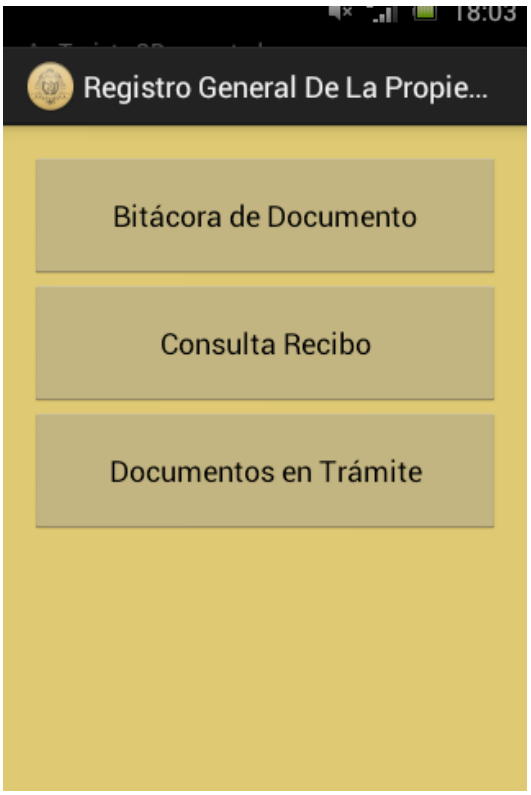
Fuente: elaboración propia, empleando HTML5.

2.5.4. Descripción de prototipo

Cumpliendo con uno de los objetivos planteados, la aplicación debe tener una navegación sumamente sencilla e intuitiva. Se tuvo en cuenta que es probable que esta sea la primer aplicación móvil que algunos de los usuarios utilicen, por lo que fue de suma prioridad la alta usabilidad del producto.

Inmediatamente después de la descarga, la primera pantalla consta únicamente de tres puntos botones, de la siguiente forma:

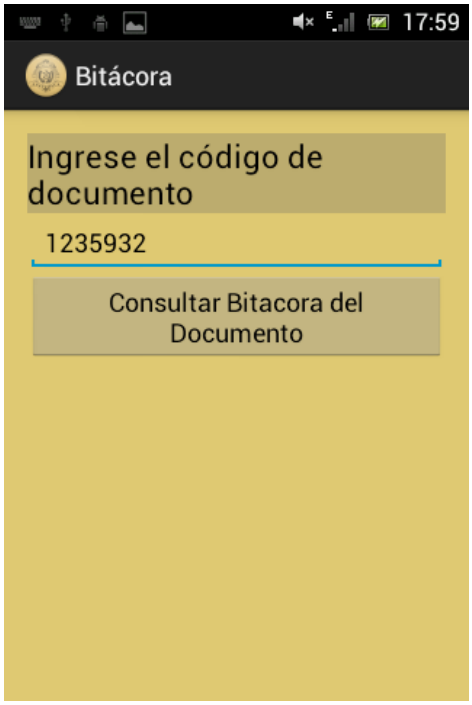
Figura 5. **Pantalla principal**



Fuente: elaboración propia, empleando HTML5.

Haciendo uso del módulo Bitácora de documento, el propietario puede conocer el historial de un documento relacionado con una finca. De esta forma, puede consultar en cualquier momento el curso del caso relacionado con ese documento. La pantalla correspondiente se muestra a continuación.

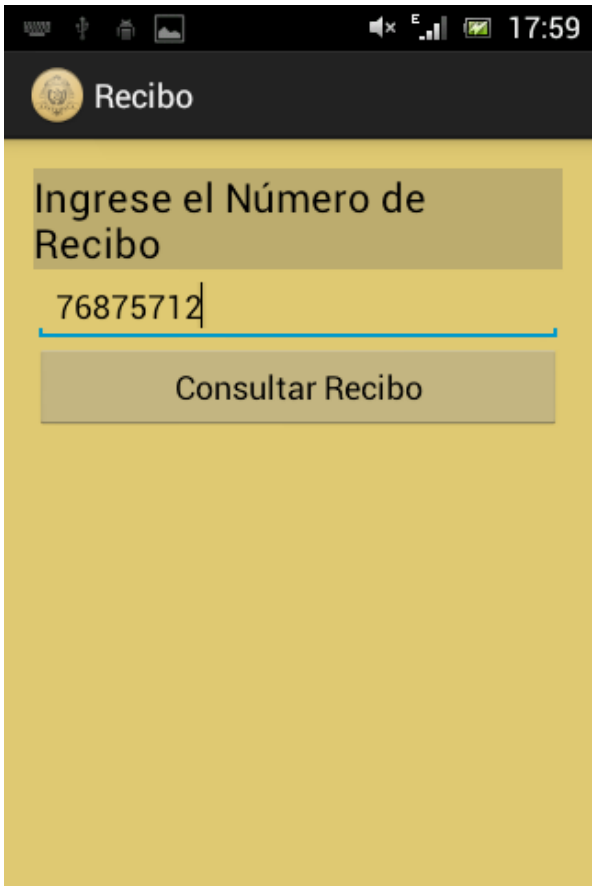
Figura 6. **Módulo bitácora del documento**



Fuente: elaboración propia, empleando HTML5.

En el módulo de Consulta de recibo, el usuario puede verificar el monto y estado de un recibo. Luego de ingresar el número de documento, el usuario puede realizar la consulta presionando el botón Consultar bitácora del documento.

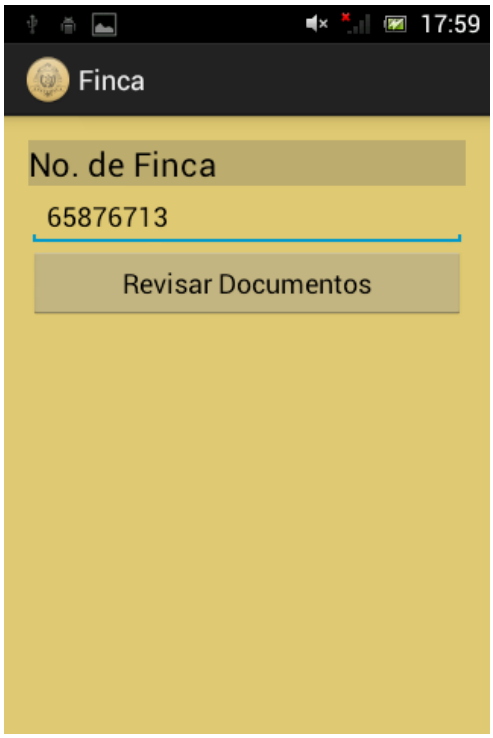
Figura 7. **Módulo de consulta de recibo**



Fuente: elaboración propia, empleando HTML5.

En el módulo Documentos en trámite, el usuario puede verificar qué documentos relacionados con sus propiedad se encuentran en proceso. Para ello, debe ingresar el código correspondiente y luego presionar Revisar documentos.

Figura 8. **Módulo de revisión de documentos**



Fuente: elaboración propia, empleando HTML5.

3. DOCUMENTACIÓN DE LA SOLUCIÓN Y CONSIDERACIONES

3.1. Patrón modelo vista controlador (MVC)

El modelo vista controlador es un patrón de diseño arquitectónico. El principal objetivo del uso de este modelo es encapsular los datos, su representación visual y la interfaz entre ambos. En este sentido, el primero corresponde al modelo, el segundo a la vista y el tercero al controlador.

A pesar de que a primera vista el uso de este modelo puede parecer que aumenta la dificultad de implementación, sus bondades se ven reflejadas en la facilidad de mantenimiento. También facilita la distribución de tareas dentro de un equipo de trabajo, teniendo, por ejemplo, un equipo encargado de un buen diseño para la vista, otro con fuerte conocimiento sobre los datos encargado del modelo y un equipo que se encargue de la visión global de la meta y manejo de peticiones encargado del controlador. De esta forma, el modelo colabora de forma importante con el control de calidad del programa.

Debido a las bondades del modelo en cuanto a organización y facilidad de mantenimiento, y teniendo en mente el objetivo de crear una aplicación de alta utilidad, este modelo resultó idóneo. A continuación se encuentra la descripción de cada una de las capas.

3.2. Modelo

Generalmente es la primera capa en la que se trabaja. En esta capa se trabaja directamente con datos desde el punto de vista de su estado. Así, el modelo corresponde a los datos y a las reglas que se le aplican a los datos, las cuales reflejan los conceptos que maneja la aplicación. El modelo le da al controlador una representación de los datos de cualquier cosa que el usuario requiera (una lista de ítems, una cadena de texto, entre otros) La forma cómo el modelo de datos le sea presentado al usuario no modifica el modelo de datos. El modelo contiene la parte más importante de la lógica de la aplicación. Las buenas prácticas indican que deben usarse métodos para obtener y definir el valor de las variables (también denominados *getters* y *setters*).

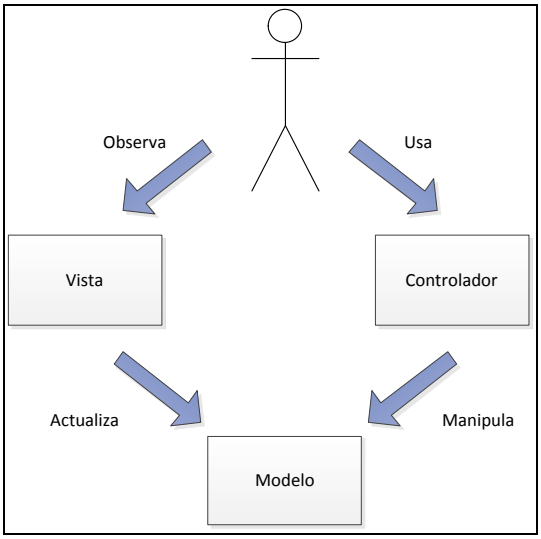
3.3. Vista

La vista es responsable de generar la salida que se envía para su despliegue. Llama al modelo para cualquier información que necesite y la coloca en un formato adecuado. Por ejemplo, una lista de datos obtenidos desde el modelo puede ser desplegada usando HTML por la vista. Está conformada por una colección de clases que dibujan y actualizan la interfaz de usuario conforme a se necesite.

3.4. Controlador

Contiene las clases que enlazan la vista y el modelo y las comunican entre sí. Puede enviar comandos al modelo y actualizar su estado, por ejemplo, editando un documento. También envía comandos a la vista asociada para cambiar la presentación del modelo.

Figura 9. **Gráfica de relación entre las capas de MVC**



Fuente: elaboración propia, empleando HTML5.

3.5. Extensible Markup Language (XML)

Es un metalenguaje que puede ser utilizado para documentos autodescriptivos y modulares, programas e incluso otros lenguajes. Estos documentos son usados con frecuencia para intercambio de datos entre sistemas, que de otra forma serían incompatibles.

La palabra en inglés *markup* se refiere a un conjunto de palabras y símbolos para describir la identidad o función de un las partes de un documento (por ejemplo: “esto es un párrafo”, “este es un pie de página”, “esta es una lista”). XML cumple esta función al tener elementos descriptivos que representan el contenido que albergan.

XML es muy versátil. El Worldwide Web Consortium (W3C), un cuerpo internacional de estándares, comenzó a trabajar XML a mediados de 1996 y la versión 1.0 fue lanzada en 1998. El valor de XML yace no tanto en su carácter de innovación, sino en la aceptación general que tiene en la industria como una forma común de describir e intercambiar data.

XML utiliza etiquetas para describir la información, como en el siguiente ejemplo:

Figura 10. **Ejemplo de la estructura básica de un documento XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<Orden>
        <Cliente>
            <nombre>Juan Perez </nombre>
            <calle>Calle 9</calle>
            <ciudad>Ciudad Aire</ciudad>
            <estado>AZ</estado>
            <zip>10000</zip>
            </Cliente>
</Orden>
```

Fuente: elaboración propia, empleando HTML5.

Una ventaja fundamental de XML es que, dado que es un lenguaje basado en texto, se explica por sí mismo y es muy sencillo de entender a primera vista. Sin embargo, esta ventaja puede ser también una desventaja: debido que se

explica muy bien a sí mismo, los documentos XML pueden convertirse rápidamente en conjuntos de datos bastante extensos. Otras cosas interesantes de XML son las siguientes:

- Jerarquía: cada elemento puede tener un elemento subordinado justo debajo de él. En el ejemplo anterior, “Cliente” contiene varios elementos “hijos”.

- Extensible: XML, a diferencia de HTML, no tiene un número limitado de etiquetas. Es posible crear tantas etiquetas como sea necesario. En el ejemplo anterior, el documento representa una abstracción de un cliente e incluye campos para describirle.

- Modular: XML provee de diseño modular y promueve la reutilización de código al admitir referencias a otros documentos XML.

- Independiente del lenguaje de programación: debido a que XML no es un lenguaje de programación *per se*, puede ser usado como un mecanismo común para intercambio de datos entre lenguajes de programación y una forma común de conectar aplicaciones, usando el protocolo orientado a objetos simples.

- No se preocupa de despliegue: XML, a diferencia de HTML, no tiene relación con mecanismos de despliegue y tiene que unirse a alguna otra tecnología como hojas de estilo en cascada (CSS, por sus siglas en inglés) para ser desplegado. Esta separación nace de una de las metas primordiales de intercambio de datos. En muchos casos, los datos se intercambian entre sistemas sin necesidad de desplegarlos.

3.6. Servicios web

Son aplicaciones para cliente y servidor que se comunican usando el protocolo HTTP en la web. También pueden verse como componentes de aplicación que están disponibles externamente. Proporcionan una forma estándar de interactuar con aplicaciones de software con distintas características, como entornos o plataformas. Su principal uso es integrar aplicaciones que estén escritas en diferentes lenguajes y corran en diferentes plataformas. Se caracterizan por su alta extensibilidad, compatibilidad e interoperabilidad. Los servicios web son débilmente acoplados de tal forma que sean de fácil mantenimiento. Los programas que proveen servicios simples pueden interactuar con otros para otorgar servicios de alto valor.

Existen diferentes modelos de servicios web para desarrolladores. Estos modelos se clasifican en dos categorías, a saber:

- REST: transferencia de estado representacional (REST, por sus siglas en inglés) es una forma de crear y comunicar servicios web en donde los recursos tienen URL y son manipuladas por operaciones especificadas en un encabezado HTTP. Cada modelo y cada proceso es modelado como un recurso con un URL único. Es una tecnología adecuada para aplicaciones que no requieren seguridad más allá de lo que está disponible en la infraestructura intrínseca a HTTP y donde HTTP es un protocolo conveniente. Google Maps y Amazon son ejemplos de sistemas que usan REST.

- SOAP/basadas en WSDL: este tipo de servicios es útil en aplicaciones de mayor peso, que requieran seguridad sofisticada y disponibilidad.

Para más información sobre SOAP, por favor leer el capítulo 4.3. de este documento.

Los clientes (programas que desean obtener un conjunto de datos determinado) contactarán a un servicio web en el servidor y enviarán una petición de servicio, pidiendo la información de interés. El servidor responderá usando una respuesta de servicio.

3.7. Descripción, descubrimiento e integración universal (UDDI)

Es un protocolo que define un método estándar para publicación, descubrimiento. El propósito principal de UDDI es indiciar datos y metadatos sobre servicios web. Por otra parte, un registro, ya sea utilizado en una red pública o destinado para uso interno de una organización, contiene un mecanismo construido con estándares para clasificar, catalogar y manejar servicios web, orientado al mismo fin: permitir que un servicio web encontrado y consumido.

Así, UDDI especifica protocolos de acceso al registro para servicios web, métodos para controlar el acceso al registro y a mecanismos para distribuir y delegar los registros hacia otros registros.

UDDI incluye:

- Un modelo de datos: define tipos de datos que incluyen una descripción sobre la función del servicio, información sobre quien lo publique y otros metadatos importante. Estos tipos de datos están definidos en diferentes esquemas XML.

- Definición de nodos y registros: incluye una definición específica de la relación entre una instancia de una implementación UDDI y otra.

    - Nodos: un nodo es un servidor UDDI que soporta al menos un conjunto de funcionalidad mínimo definida en la especificación. Es miembro de un y solo un registro UDDI.

    - Registro: está compuesto por uno o más nodos. Ejecuta un conjunto completo de funcionalidades.

    - Registros relacionados: comparten un espacio de nombres común. Cuentan con registros UDDI individuales que implementan políticas de compartición de información.

3.8. Lenguaje de descripción de servicios web (WSDL)

Cuando se crea un servicio y se desea hacerlo disponible para su consumo, es necesario saber qué información necesita el servicio, qué información devuelve y dónde encontrar el servicio en sí. El lenguaje de descripción de servicios web (WSDL, por sus siglas en inglés), es en esencia un documento XML usado para describir servicios web. WSDL es normalmente una combinación de SOAP y un esquema XML. Un programa cliente que se conecta a un servicio web puede leer el WSDL y determinar qué funciones están disponibles en el servidor. Al usar el servicio web, el cliente en realidad está llamando a una de las funciones listadas en el WSDL.

Al utilizar XML como formato estándar, es posible compartir la información sobre servicios de forma transparente. El primer paso es decidir qué funciones cumplirá el servicio en realidad. Luego, identificar las salidas del servicio

(nótese que en ningún momento se indica que deba entenderse el funcionamiento del servicio web). Por último, se crea un documento WSDL correspondiente al servicio.

La descripción del servicio detalla la interfaz abstracta a través de la cual un cliente podría acceder a un servicio en caso de utilizarlo. Se identifica qué puertos, qué tipo de puertos y las direcciones que implementan el servicio; los lazos (*bindings*) que indican los protocolos usados por cada puerto y el mensaje que llevará el servicio.

De esta forma, un documento WSDL puede verse de la siguiente forma:

Figura 11. **Estructura general de un WSDL**

```
<?xml version="1.0" enconding="UTF-8"?>
<wsdl:definitions>

    <wsdl:types>
     ...
    </wsdl:types>

    <wsdl:message>
        <part name="parametro" type="xsd:string"/>
    </wsdl:message>

    <wsdl:portType>
        <wsdl:operation name="funcion">
         ...
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding>
     ...
    </wsdl:binding>

    <wsdl:service name="ServicioEjemplo">
     ...
    </wsdl:service>

</wsdl:definitions>
```

Fuente: elaboración propia, empleando HTML5.

3.9. Protocolo de acceso simple a objetos (SOAP)

El protocolo de acceso simple a objetos (SOAP, por sus siglas en inglés) es un protocolo estándar de intercambio de información estructurada, utilizado ampliamente en la implementación de servicios web.

La principal aplicación de SOAP es la comunicación entre aplicaciones. Es independiente de la plataforma y lenguaje. Usa HTTP como protocolo de transporte genérico.

La utilización de SOAP implica varias ventajas, a saber:

- Protocolo “de bajo peso”: algunos protocolos ya existentes contienen características avanzadas, como localizar objetos y realizar registros. En su esencia, SOAP define cómo conectar sistemas y se basa en tecnologías adicionales para proveer de registro y locación, como UDDI y WSDL, respectivamente.

- Compatible con múltiples protocolos de transporte: estos incluyen HTTP, protocolo de transporte de correo simple (SMTP), protocolo de transferencia de archivos (FTP) y servicio de mensaje de Java (JMS).

- Ampliamente soportado: la mayor parte de empresas competitivas manejan SOAP, como Apache, IBM y Microsoft.

- Extensible: en el encabezado de un documento XML pueden implementarse características adicionales como autenticación, versionamiento y optimización.

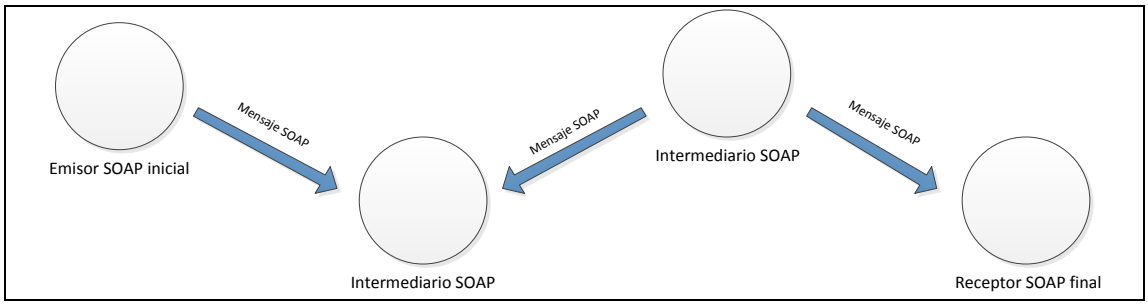
- Independiente del lenguaje y sistema operativo.

3.9.1. Ruta de un mensaje SOAP

No es más que un conjunto de nodos SOAP enlazados que son recorridos por un mensaje SOAP. Esta ruta se resume a un emisor SOAP, cero o más intermediarios SOAP y un receptor SOAP final.

El caso más sencillo consiste en el paso de un mensaje SOAP entre únicamente dos nodos: el emisor y receptor. En casos más complejos, el mensaje es procesado por nodos intermediarios, que reciben un mensaje SOAP y lo envían al siguiente, tal y como muestra la figura.

Figura 12. **Representación de la ruta de un mensaje SOAP**



Fuente: elaboración propia, empleando Visio.

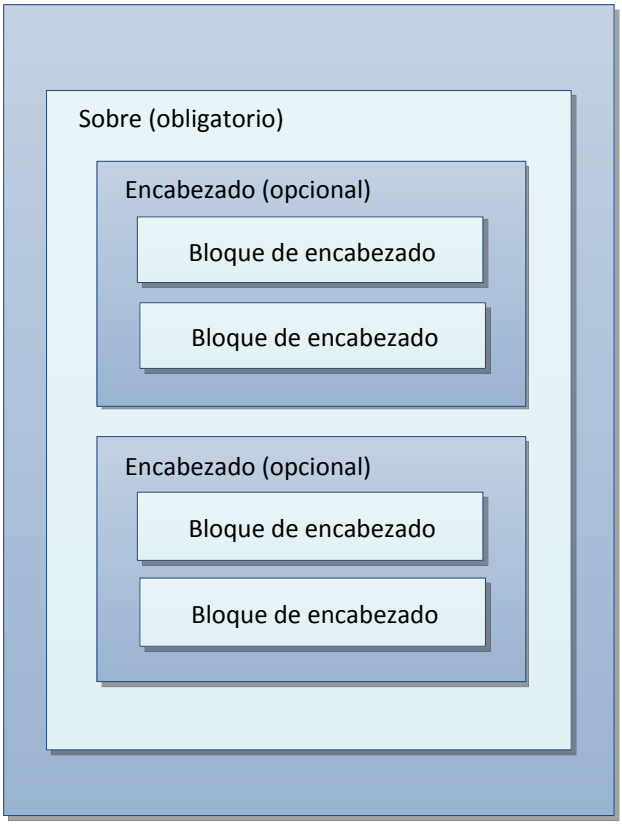
Un intermediario SOAP es tanto emisor como receptor, provee bloques de encabezado en un mensaje SOAP y redirige el mensaje hasta su receptor final.

El receptor final es el destino final del mensaje. En algunas circunstancias, el mensaje puede no alcanzar el receptor y eso se debe a algún error en alguno de los intermediarios SOAP.

3.9.2. Estructura de un mensaje SOAP

SOAP está estructurado bajo el concepto de envío de documentos XML de un emisor a un receptor. En este punto, un documento XML se convierte en un documento SOAP y está compuesto por un encabezado (<Header>) opcional y un cuerpo (<Body>) opcional. Se usa un elemento de falla (<Fault>) dentro del cuerpo para reportar errores. Los elementos se describen a continuación y se representan en la gráfica siguiente.

Figura 13. **Estructura de un documento SOAP**



Fuente: elaboración propia, empleando HTML5.

- Sobre (<Envelope>): es el elemento raíz de todo mensaje SOAP. Contiene dos elementos hijos, un encabezado (<header>) opcional y un cuerpo (<body>) obligatorio.

- Encabezado (<Header>): es un subelemento opcional del sobre y se usa para pasar información relacionada con la aplicación. La información se procesa por los nodos SOAP a lo largo del camino recorrido por el mensaje. Los elementos hijos inmediatos del elemento encabezado son llamados bloques de encabezado. Un bloque de encabezado es un elemento de un grupo lógico de datos que se dirigen a un grupo de nodos SOAP que pueden ser encontrados en el camino de un mensaje desde su emisor hasta su receptor final. Los bloques pueden ser analizados por nodos intermediarios; sin embargo, en una aplicación real, no todos los nodos procesan cada bloque de encabezado: cada nodo está diseñado para procesar ciertos bloques de encabezado y cada bloque de encabezado está pensado para ser procesado por ciertos nodos en particular. El encabezado permite agregar características a un mensaje SOAP de una forma descentralizada, sin acuerdo previo entre emisor y receptor. SOAP define algunos atributos destinados a indicar quién debe lidiar con una falla, en el caso que exista, y si es opcional u obligatorio. Esta información incluye el paso de directivas e información contextual relacionada al procesamiento del mensaje. Esto permite que los mensajes SOAP se construyan acorde a la aplicación.

- Cuerpo (<Body>): es un elemento obligatorio dentro sobre en el cual es transportada la información principal que contiene un mensaje SOAP. Este elemento y sus elementos asociados son usados para intercambiar información entre el emisor y receptor SOAP. Contiene un elemento de falla y los demás son definidos por el servicio web que los utiliza.

- Falla (<Fault>): elemento del cuerpo usado para llevar información de error e información de estado. Si el elemento falla, se agrega al mensaje, debe aparecer solamente una vez en el cuerpo del mensaje.

Con excepción del elemento falla, los elementos XML dentro del encabezado y el cuerpo son definidos por las aplicaciones que los usan, aunque la especificación SOAP define de la estructura de algunas constantes.

Un esqueleto de un mensaje SOAP es el siguiente:

Figura 14. **Estructura básica de un mensaje SOAP**

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

Fuente: elaboración propia, empleando HTML5.

3.10. Librería kSOAP2

Es una librería de código abierto muy popular entre los desarrolladores de aplicaciones móviles para el sistema operativo Android. Es especialmente útil cuando se necesita analizar sintácticamente un archivo WSDL y mensajes SOAP.

El sistema operativo Android no incluye de forma predeterminada ningún soporte para acceso a servicios web SOAP. Debido a que el servicio web del Registro de la Propiedad es un servicio web SOAP, se hizo uso de la librería kSOAP2 para poder comunicarse con el servicio desde Android.

3.11. Notación de objeto JavaScript (JSON)

La notación de objeto JavaScript (JSON, por sus siglas en inglés) es un formato de intercambio de datos. Su estructura hace que sea fácilmente comprensible a primera vista. Es un conjunto de reglas gramaticales que serializa objetos, arreglos, nombres, booleanos y nulos. Está basado en un subconjunto de la sintaxis JavaScript pero difiere de él: JSON no es JavaScript.

Es completamente independiente del lenguaje y utiliza una serie de convenciones que le parecerían familiares a programadores con experiencia en C, C++, C#, Java, Python o JavaScript.

JSON es capaz de representar números, booleanos, cadenas de texto, nulos, arreglos organizados en una secuencia de valores y objetos (en forma de mapeo de valores de cadenas de texto), compuestos por estos valores (u otros arreglos y objetos). No representa nativamente valores más complejos como funciones, expresiones regulares o fechas.

Está conformado por dos estructuras:

- Una colección de parejas cuyos componentes son nombre y valor. Se puede pensar en estas parejas como un objeto, una estructura, un diccionario, una tabla *hash* o un arreglo asociativo.

- Una lista ordenada de valores. Se puede pensar en esta lista como un vector, una secuencia, una lista o un arreglo.

Estos son datos estructurados universales. Prácticamente todos los lenguajes de programación moderno soportan esta forma de organizar datos.

Un conjunto de nombre-valor es denominado objeto. Un objeto comienza con una llave que abre (“{“) y termina con una llave que cierra (“}”). Cada nombre es seguido de dos puntos (“:”) y las parejas nombre-valor están separadas por una coma (“,”).

Puede lo anterior verse de forma clara con el siguiente ejemplo:

Figura 15. **Ejemplo de notación JSON**

```
{
“colaboradores”: [
{ “primerNombre: “Juan”, “primerApellido: ”Pérez”},
{ “primerNombre: “Ana”, “primerApellido: ”González”},
{ “primerNombre: “Alejandro”, “primerApellido:
”Polanco”},
]
}
```

Fuente: elaboración propia, empleando HTML5.

3.12. Interfaz de programación de aplicaciones

La interfaz de programación de aplicaciones (API, por sus siglas en inglés) es una interfaz que sirve de puente de comunicación entre una aplicación con el sistema operativo, motor de bases de datos, protocolo de comunicación o algún otro tipo de componente. Es importante recalcar que una API es una interfaz a través de la cual se comunicarán dos componentes de software, de tal forma que las aplicaciones o componentes de software se comunican entre sí por medio de la API sin intervención del usuario.

Detrás de una interfaz gráfica, puede haber varias aplicaciones comunicándose entre sí a través de una API. Esto sucede, por ejemplo, en el sitio web de venta de accesorios deportivos en línea: el sitio utiliza una API para enviar la información de la tarjeta de crédito a una aplicación remota que verifica los datos; una vez se tenga una respuesta, la aplicación remota enviará la confirmación y se completará la transacción.

Google, Flickr, Amazon y Twitter son ejemplos de empresas que cuentan con API que permite que los desarrolladores integren funcionalidad a sus sitios web y aplicaciones. En el caso particular de Google, existen API como YouTube Analytics API, YouTube Streaming API y Google Maps API. La API de Flickr consiste en un conjunto de métodos que se pueden invocar. La API Amazon Product Advertising otorga a los desarrolladores la oportunidad de acceder a una selección de productos que vende Amazon y de publicitar productos Amazon con el fin de monetizar un sitio.

3.13. WebSphere Application Server

Proporciona un rango de entornos de ejecución Java EE 7 flexibles y seguros, con capacidad para gestionar desde proyectos de producción ligeros hasta grandes despliegues empresariales.

3.13.1. Antecedentes

La *world wide web* es relativamente nueva. Su popularidad entre tanto usuarios individuales como negocios ha crecido de forma exponencial. Aunque los usuarios usen la web para una serie de propósitos, los negocios usan la web primordialmente para proveer productos, servicios e información a sus clientes, empleados y proveedores.

Cuando los negocios comenzaron a mostrarse en la web, era suficiente una página web estática con información general de sus servicios y productos, adjuntando información de contacto como teléfonos o dirección. Los negocios que proveían servicios de información, como compañías de software, fueron los primeros en pasar esa frontera y poner sus servicios directamente para la descarga.

A medida que la web ha madurado y las nuevas tecnologías se han desarrollado, las páginas web estáticas no han sido suficientes. En respuesta, los negocios han construido sitios web que permitan mayor interacción y que admitan comunicación con otros negocios, permitiendo incluso que los empleados se comunicasen entre sí.

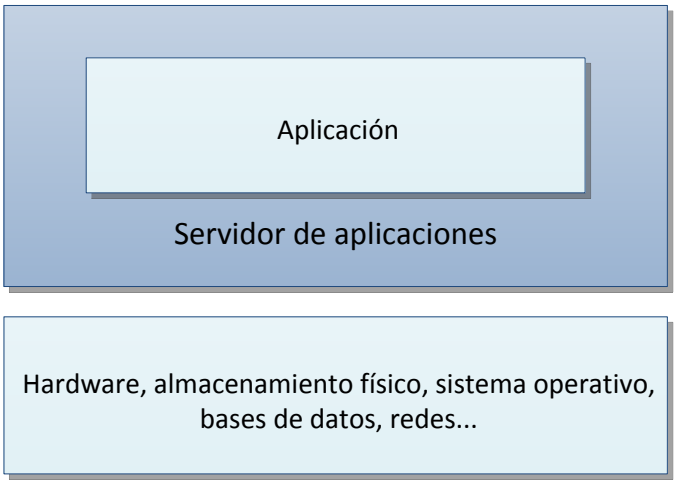
La plataforma IBM WebSphere es una plataforma de software para sitios con arquitectura orientada a servicios. WebSphere provee un amplio rango de productos que ayudan a desarrollar y soportar aplicaciones de negocio. Está diseñado para que a los clientes se les facilite la tarea de construir, desplegar y mantener sitios web dinámicas, aplicaciones móviles y otras soluciones complejas de forma productiva y efectiva.

3.13.2. Vista general del servidor de aplicaciones WebSphere

El servidor de aplicaciones WebSphere es una plataforma para aplicaciones bajadas en Java, provee una infraestructura para ejecutar aplicaciones para manejar el negocio. Aísla la infraestructura del hardware, sistema operativo y red. Un servidor de aplicaciones también sirve como plataforma para desarrollar y desplegar servicios web y Enterprise JavaBeans, también como un motor transaccional y de mensajes, mientras pone a disposición del usuario final la lógica del negocio y a una variedad de dispositivos cliente.

La siguiente figura muestra una representación básica de un servidor de aplicaciones.

Figura 16. **Representación básica de un servidor de aplicaciones**



Fuente: elaboración propia, empleando HTML5.

El servidor de aplicaciones actúa como intermediario entre los sistemas *back-end* y los clientes. Provee un modelo de programación, un entorno de infraestructura y un set de estándares para un enlace coherente entre ellos.

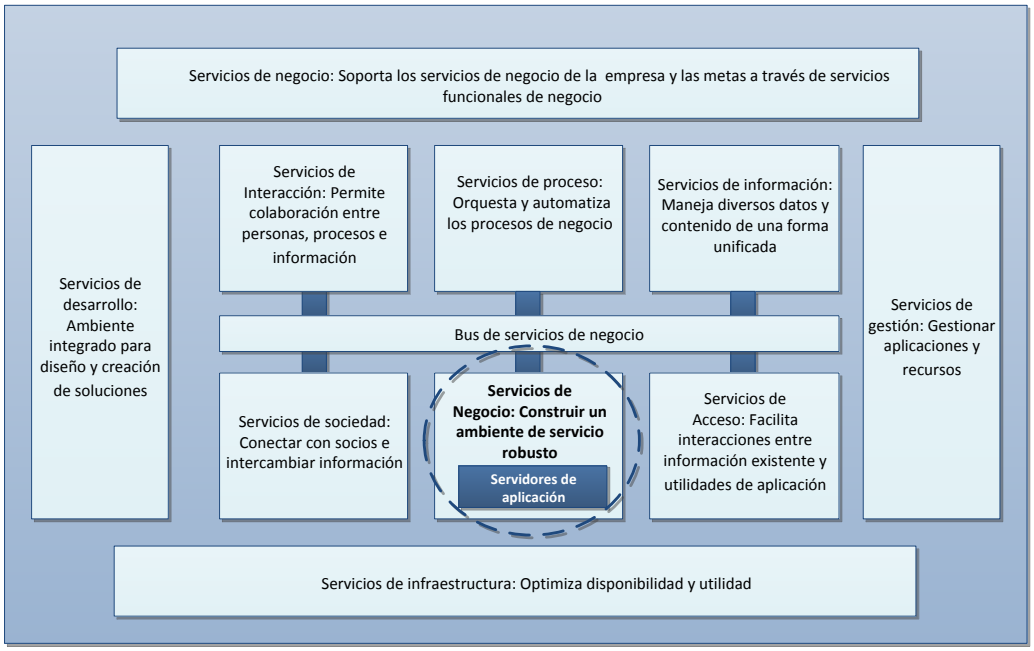
WebSphere provee un ambiente para ejecutar las soluciones e integrarlas en cualquier plataforma y sistema como servicios de aplicación de negocio, conforme a la arquitectura orientada a servicios.

3.13.3. Posición de WebSphere dentro de la arquitectura orientada a servicios

Desde el punto de vista de la arquitectura orientada a servicios, es posible ejecutar y desplegar aplicaciones reutilizables rápida y fácilmente; correr servicios en un ambiente seguro, escalable y de alta disponibilidad; conectar funcionalidades de diferentes aplicaciones y extender su alcance; manejar

aplicaciones sin demasiado esfuerzo y crecer a medida que las necesidades de negocio evolucionan.

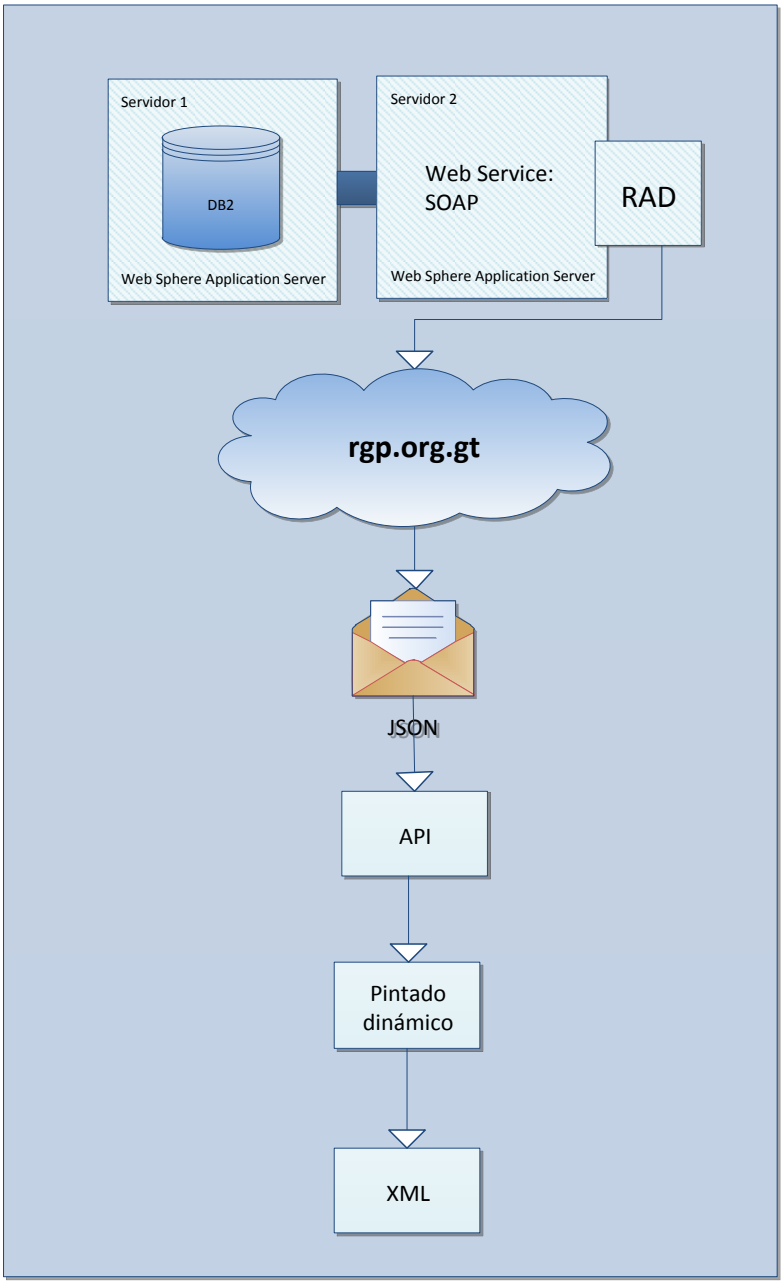
Figura 17. **Posición de WebSphere dentro de SOA**



Fuente: *Migración del recopilador de datos SOA*. http://www.ibm.com/support/knowledgecenter/es/SS3JRN_7.2.1/com.ibm.itcamfapps_soa.doc_7201/soa_install_guide/silent_migrate_was_dc_soa.html. Consulta: mayo de 2014.

WebSphere provee un ambiente para ejecutar soluciones e integrarlas con cualquier plataforma y sistema como un servicio de negocio conforme a la arquitectura de negocio, como indica la figura 17.

Figura 18. **Diagrama general del funcionamiento de la aplicación**



Fuente: elaboración propia, empleando HTML5.

La información sobre las fincas, usuarios y documentos está alojada en una base de datos DB2, que corresponde al primer servidor. En otro servidor se encuentran los servicios web SOAP que proporcionan una forma de acceder a esos datos. Haciendo uso de Web Sphere Application Server y Rational Application Development, se empaqueta la información en un JSON. Este JSON llevará la información solicitada a la API, quien pintará dinámicamente la pantalla correspondiente al estado de navegación del usuario.

3.13.4. Descripción de funcionamiento de módulo

La arquitectura del código de la aplicación se divide en cinco paquetes diferentes. Estos paquetes son:

- Utilerías
- Lógica de la consulta a distancia
- Administración objetos de datos para su administración
- Carpeta con la lógica validaciones para el servicio web

Los paquetes se ilustran en la siguiente figura:

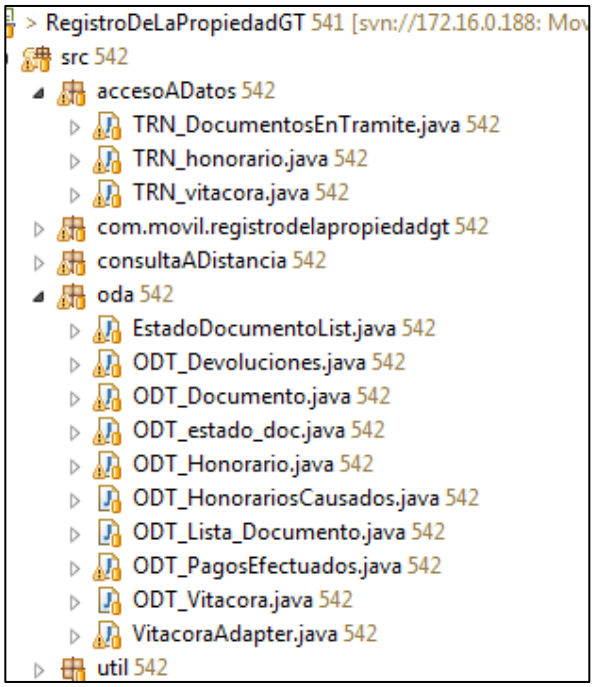
Figura 19. **Arquitectura del código de aplicación**

Fuente: elaboración propia, empleando HTML5.

3.13.4.1. Paquete de acceso a datos

El paquete “accesoADatos” contiene clases para manipulación de ODT, los cuales mapean con objetos la información que proviene del servicio web. El paquete “ODA” contiene estos ODT:

Figura 20. **Paquete de acceso a datos**

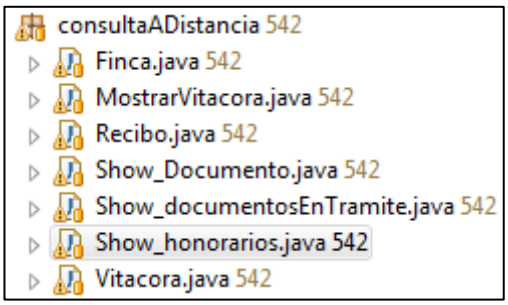


Fuente: elaboración propia, empleando HTML5.

3.13.4.2. Paquete de consultas a distancia

Por otro lado, las clases albergadas por el paquete ”consultaADistancia” son clases destinadas a definir objetos compatibles tanto con la lógica como con la interfaz de la aplicación. Por ejemplo, a través de estas clases se generan objetos que luego serán pintados en la interfaz gráfica en el teléfono.

Figura 21. **Paquete de consultas a distancia**



Fuente: elaboración propia, empleando HTML5.

Considerando que la capa de programación es independiente de la capa de comunicación con el servicio web, se incluyeron métodos destinados a realizar pruebas. Uno de estos métodos es el siguiente:

```
Private void llenarReciboTest(){
    //método de prueba
}
```

Este método llena el recibo con valores predeterminados y se usa para verificar que la interfaz se pinte dinámicamente en función de la información que reciba.

3.13.4.3. Paquete de utilidades

La librería “Util” contiene varias clases con fines de implementación. Entre ellas se encuentra librería de constantes que sirven para hacer referencia a valores fijos en la aplicación. De esta forma, en caso que una o más de las variables cambie en algún momento, es posible actualizar la estructura de constantes en lugar de modificar cada una de las ocurrencias. Es importante

mencionar que, por razones de seguridad, la librería no está representada en este documento. Sin embargo, la lógica de la clase se muestra a continuación:

Figura 22. **Lógica de clase correspondiente a librería "Util"**

```
static public String _Honotario_ListaHonorario = "lista honorarios";
       static public String _honorario_listaHonorario_monto = "monto";
static public String _honorario_listaHonorario_NombreDocumento = "IdDoc";

static public String _Honorario_ListaDevoluciones = "lista devoluciones";
       static public String _Honorario_ListaDevoluciones_fecha = "fecha";
       static public String _Honorario_ListaDevoluciones_monot = "monto";

static public String _Honorario_ListaPagosEfectuados = "listapagos
efecturados";
       static public String _Honorario_ListaPagosEfectuados_fecha = "fecha";
       static public String _Honorario_ListaPagosEfectuados_monto = "monto";
//final estructura recibo

//Estructura bitácora
         static public String _Bitacora_estado = "estado";
         static public String _Bitacora_fecha = "fecha";
         static public String _Bitacora_hora = "hora";
         static public String _Bitacora_clave = "clave";
//final estructura bitácora
         static public String _DocEnTramite_IdDoc = "IDDoc";

////////////////////////////// Fin JSON ////////////////////////////
```

Fuente: elaboración propia, empleando HTML5.

3.13.4.4. Flujo de consulta al servicio web

La comunicación hacia el servicio web SOAP corre en el *background* de la aplicación. A continuación se muestra un bloque de código que describe como sucede lo anterior.

Figura 23. **Bloque de código describiendo la comunicación hacia servicio web SOAP**

```
protected String doInBackground(String... arg0)
       try{
                String STR_NoRecibo=ET_NoRecibo.getText().toString();

                SoapObject Request = new SoapObjetc(NAMESPACE, METHOD_NAME);
                HttpTrasnportSE transporte = new HttpTransportSE(URL);
                Request.addProperty(Cons._WS_InputName_ConsultaRecibo, STR_NoRecibo);

             SoapSearilizationEnvelope soapEnvelope = new
             SoapSerializationEnvelope(SoapEnvelope.VER11);
                SoapEnvelope.setOutputSoapObject(Request);
                transporte.call(SOAP_ACTION, soapEnvelope);
                resultString = (SoapPrimitive) soapEnvelope.getResponse();
                System.out.println(resultStirng.toString());

                String respuesta_WS = resultStirng.toString();
                if(!respuesta_WS.equals("-1")&&!respuesta_WS.equals("-2"){
                    Show_honorarios.honorarios =
             traductor.getHonoraio(traductor.StringToJson(respuesta_WS));

        Show_honorarios.honorarios.setID_recibo(ET_NoRecibo.getText().toString().toUpperCase());
                Show_horararios.NoRecibo = ET_NoRecibo.getText().toString().toUpperCase();
                }
                else{
                        //asignacion de estados de error
                        if(respuesta_WS.equals("-1"){
                                Existe = false;
                        }
                                if(respuesta_WS.equals("-2")){
//...
```

Fuente: elaboración propia, empleando HTML5.

El proceso anteriormente mostrado es invocado de la siguiente manera.

Figura 24. **Invocación de proceso background**

```
public SoapPrimitive resultString = null;

protected void onPostExecute(){
        if(pd != null){
                ter = true;
                pd.dismiss();
                pd.hide();
        }
}
```

Fuente: elaboración propia, empleando HTML5.

Posteriormente, se comprueba la instancia y se espera una respuesta. Este tiempo de espera tiene una longitud establecida en la librería de constantes. Esta comprobación se hace de la siguiente forma.

Figura 25. **Comprobación de instancia**

```
    //...
    }
    else
    {
         objcon  = nrew conasync();
         objcon.execute("");
         Show_honorarios.NoRecibo = STR_noRecibo = STR_noRecibo.toUpperCase();

         int timeOut = Cons._TTL_Movil;
         while (Show_honorarios.Respuesta.equals("")&&timeOut!=0){
                  espera();
                  timeOut = timeOut-1;
         }
         if(!errorWS){
                  if(Existe){
                           if(timeOut != 0){
                                    Intent it_MLO = new Intent(this, Show_honorarios.class);
                                    startActivity(it_MLO);
                           }else{
                                    pd.dismiss();
                                    pd.hide();
                                    Toast.makeText(
                                             getBaseContext(),
                                             "la comunicación no se consiguió por favor prueba nuevamente, o
su número de recibo es incorrecto",
                                    Toast.LENGTH_LONG*Cons._tiempo_MSJ_Espera
                                    ).show();
                           }
                  }else{
                           pd.dismiss();
                           pd.hide();
                           Toast.makeText(
                                    getBaseContext(),
                                    "Su numero de recibo es incorrecto",
                                    Toast.LENGHT_LONG*Const._tiempo_MSJ_Espera
                                    ).show();
//...
```

Fuente: elaboración propia, empleando HTML5.

De esta forma, se recibe una cadena de texto con sintaxis JSON. Esta cadena es transformada a un objeto de tipo “honorario”. En este caso, este objeto es el que contiene la estructura correspondiente a un recibo.

Figura 26. **Cadena de texto con sintaxis JSON**

```
/**
*
* @param NoRecibo numero de recibo del cual se desea obtener datos
*
*/
private void poblarhonorario(String Respuesta){
        honorarios = traductor.getHonorario(traductor.StringToJson(Respuesta));
}
```

Fuente: elaboración propia, empleando HTML5.

Un objeto que reúne los datos del objeto “honorarios” es preparado para su envío como parámetro al objeto lo manejará a nivel de interfaz. Esto ocurre de la siguiente forma:

Figura 27. **Preparación de objeto parámetro hacia interfaz**

```
//Definición de longitud del String[]
pagos_efectuados = new String[ArrPagos.size()];
Honorarios = new String[Arrhonorario.size()];
Devoluciones = new String[ArrDevoluciones.size()];

//casteo de arrayList a String[]
pagos_efectuados = ArrPagos.toArray(pagos_efectuados);
Honorarios = Arrhonorario.toArray(Honorarios);
Devoluciones = ArrDevoluciones.toArray(Devoluciones);

//Definición de encabezados con sumatorias de honorarois, pagos y devoluciones
groups = new String[] {"Pagos Efectuados\n Q. + "honorios.getTotalPagoEfectuados(), "Honorarios causados \n
Q. -"+ honorarios.getTotalHonorarios(), "devoluciones \n Q. "+ honorarios.getTotalDevoluciones()};
```

Fuente: elaboración propia, empleando HTML5.

Este método pinta el objeto en la interfaz gráfica y lo muestra en la pantalla del teléfono, realizando los cálculos necesarios conforme al contenido del objeto “honorario”.

3.14. Acceso a servicios web desde Android

En las aplicaciones móviles es común el uso de servicios web para intercambio de datos e interacción con otros subsistemas. Para poder desarrollar aplicaciones Android usando el kit de desarrollo Android.

Usando RAD, se instaló la extensión de Android para Eclipse a través del menú Instalar nuevo software ubicado en el menú de ayuda. Se descargó a través de la dirección https://dl-ssl.google.com/android/eclipse/.

Luego se procedió a instalar un dispositivo virtual Android (AVD, por sus siglas en inglés), el cual es una configuración de dispositivo para el emulador que permite modelar dispositivos del mundo real.

3.15. Publicación de aplicación en la Google Play Store

A continuación se describe la publicación en la Google Play Store.

- Luego se deben subir los recursos. Esto consiste en un archivo de extensión APK de tamaño máximo de 50 Mb. Es posible subir más de un archivo APK si no es suficiente para todos los dispositivos deseados.

- Es importante tener en mente que los nombres de los paquetes no pueden modificarse, por lo que deben ser seleccionados cuidadosamente.

- Capturas de pantalla: es necesario que existan dos capturas de pantalla de la aplicación. Es posible subir otras seis, en caso que el desarrollador lo considere conveniente. Las capturas de pantalla pueden estar en formato JPEG o PNG sin canales alfa. El tamaño mínimo es de 320 píxeles y el máximo es 3 840 píxeles. La imagen debe estar con la resolución suficiente para que la plataforma pueda redimensionarla conforme sea necesario.

- Borrador de archivo APK: es posible guardar la aplicación como borrador mientras se edita el resto de aspectos de la lista. Esto es muy útil para verificar la integridad del archivo con respecto a la plataforma, sin necesidad de publicar la aplicación.

Algunos detalles deben especificarse, estos son:

- Nombre: nombre de la aplicación en Google Play Store.

- Descripción: descripción de la aplicación en un máximo de 4 000 caracteres.

- Idioma: de forma predeterminada, es inglés de Estados Unidos. El desarrollador puede incluir traducciones del nombre y descripción en función de facilitar la promoción.

- Tipo de aplicación: se debe seleccionar Aplicaciones o Juegos. Se seleccionó la primera opción en este caso.

- Categoría: entre las categorías disponibles están: educación, economía, finanzas, comunicaciones, salud y bienestar, estilo de vida y cómics.

- Ícono de alta resolución: este ícono se utilizará en las diferentes ubicaciones de Google Play. Debe ser un archivo PNG de 32 bits (con canal alfa), de un tamaño máximo de 1 024 KB y 512 x 512 píxeles.

Opcionalmente, puede agregarse una imagen destacada. Se utiliza para las diversas promociones de Google Play Store. Aunque no es obligatorio para publicar y guardar la aplicación en Google Play Store, es necesario para aparecer en el contenido destacado de la plataforma. Debe ser un archivo con formato JPEG o PNG de 24 bits (sin canal alfa) y de 1 024 x 500 píxeles.

CONCLUSIONES

1. A pesar de que la Ley del Libre Acceso a la Información corresponde al Decreto número 57-2008, es decir, tiene ocho años de tener vigencia, aún no existía una aplicación móvil que consumiera el servicio web proporcionado por el Registro General de la Propiedad. Con la implementación del sistema se hace valer el derecho de las personas a acceder a la información en el ámbito de información sobre propiedades, representando una forma de acceder a la información que ya estaba puesta a disposición por el Registro General de la Propiedad.

2. El sistema resulta siendo entonces una forma de agregarle valor a la sociedad, al ayudar a ejercer su derecho a acceder a información pública de forma gratuita.

3. Por otro lado, la tecnología utilizada favoreció el fácil acoplamiento de cada elemento en el sistema. En este sentido, se utilizaron tecnologías IBM como Web Sphere Application Server y Rapid Application Developer, que están diseñadas para integrarse fácilmente entre sí. Asimismo, se usó JSON como formato de intercambio de datos y SOAP como protocolo estándar de comunicación. Esta sinergia conllevó a una aplicación de alta abstracción, fácil mantenimiento y alta usabilidad, cumpliendo con los objetivos del proyecto.

RECOMENDACIONES

1. Al desarrollar, se deben seleccionar tecnologías que puedan integrarse fácilmente y de débil acoplamiento. Además, se recomienda el uso de protocolos estándares de comunicación. De esta manera, el sistema producido será fácilmente escalable y adaptable a otros sistemas.

2. Se debe tener especial cuidado en la selección del modelo arquitectónico. Esto facilitará mantenimiento del sistema, la distribución de trabajo en el equipo de trabajo y reducirá los tiempos de entrega.

3. Al subir una aplicación a Google Play Store, se debe dedicar el tiempo necesario para completar el formulario de características de la aplicación. Esto aumentará las posibilidades de que la aplicación sea encontrada por potenciales usuarios, además de facilitar la rápida comprensión del objetivo de la aplicación.

BIBLIOGRAFÍA

1. ¿Cómo subir aplicaciones?. [en línea]. <https://support.google.com/googleplay/android-developer/answer/113469?hl=es>. [Consulta: mayo de 2014].

2. ALCÁNTARA PILAR, Juan Miguel. *Modelización del comportamiento del consumidor online: el papel moderado de la cultura, el diseño web el idioma*. España: Universidad de Granada. 436 p.

3. *Comprender los servicios web, parte 2: Web Services Despcription Language*. [en línea]. <http://www.ibm.com/developerworks/ssa/webservices/tutorials/ws-understand-web-services2/>. [Consulta: mayo de 2014].

4. *Consumiendo web services SOAP en Android con ksoap2*. [en línea]. <http: //androcode. es /2012 /05 /consumiento-web-services-soap-en - android-con-ksoap2/>. [Consulta: abril de 2014].

5. *Estructura de las aplicaciones orientadas a objetos: El patrón modelo-vista-controlador (MVC)*. Facultad de Informática, Universidad Complutense de Madrid. [en línea]. <https://www.fdi. ucm. es/profesor/jpavon/poo/2.14.MVC.pdf>. [Consulta: mayo de 2014].

6. *Guía breve de tecnologías XML*. [en línea]. <http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML>. [Consulta: abril de 2014].

7. *Introduction to web services Ioannis G. Baltopoulos Departamento de Ciencias de la Computación, Universidad Imperial de Londres*. [en línea]. <https://www.cl.cam.ac.uk/~ib249/teaching/Lecture1.handout.pdf>. [Consulta: abril de 2014].

8. *IT Glossary: Service-Oriented Architecture*. [en línea]. <http://www. gartner.com/it-glossary/service-oriented-architecture-soa>. [Consulta: marzo de 2014].

9. *Ley de libre acceso a la información pública*. [en línea]. <http://www2.gwu.edu/~nsarchiv/evidence/Guatemala%20ley.pdf>. [Consulta: marzo de 2014].

10. PAPAZOGLOU, Mike P. *SOAP: Simple object access protocol*. [en línea]. <http://www.cs.colorado.edu/~kena/classes/7818/f08/lectures/lecture_3_soap.pdf>. [Consulta: mayo de 2014].

11. *Registro de la Propiedad en Guatemala*. [en línea]. <http://www.rgp.org.gt/>. [Consulta: mayo de 2014].

12. *QuickStudy: Application Programming Interface (API)*. [en línea]. <http://www.computerworld.com/s/article/43487/Application_Programming_Interface?pageNumber=1>. [Consulta: mayo de 2014].

13. *Recursos gráficos, capturas de pantalla y video*. [en línea]. [https://support.google.com/googleplay/android-developer/answer/1078870>. [Consulta: mayo de 2014].

14. *Service oriented architecture (SOA): Simply good design*. [en línea]. <http: // www-01. ibm. Com software/ solutions/ soa />. [Consulta: marzo de 2014].

15. *Step by step method to access web services from Android*. [en línea]. http: // www.codeproject.com / Articles / 112381 / Step-by-Step-Method-to-Access-Webservice-from-Andr>. [Consulta: abril de 2014].

16. *The Java EE6 Tutorial: What are web services?.* [en línea]. <http://docs. oracle.com/javaee/6/tutorial/doc/gijvh.html>. [Consulta: abril de 2014].

17. *The XML FAQ - Frequently-Asked-Questions*. [en línea]. <http://xml.silmaril.ie/>. [Consulta: mayo de 2014].

18. *Tutorials Point: UDDI*. [en línea]. <http://www.tutorialspoint.com/uddi/>. [Consulta: abril de 2014].

19. *Understanding MVC Architecture*. [en línea]. <https://www. youtube.com/watch?v=eTdVkgF_Slo>. [Consulta: mayo de 2014].

20. *Understanding SOAP Microsoft Development Network*. [en línea]. <http: //msdn.microsoft.com/en-us/library/ms995800.aspx>. [Consulta: mayo de 2014].

21. *Web Services Description Language (WSDL) Wanasanan Thongsongkrit*. [en línea]. <http://www.cs.colorado.edu/~kena/ classes/7818/f06/lectures/WSDL.pdf>. [Consulta: mayo de 2014].

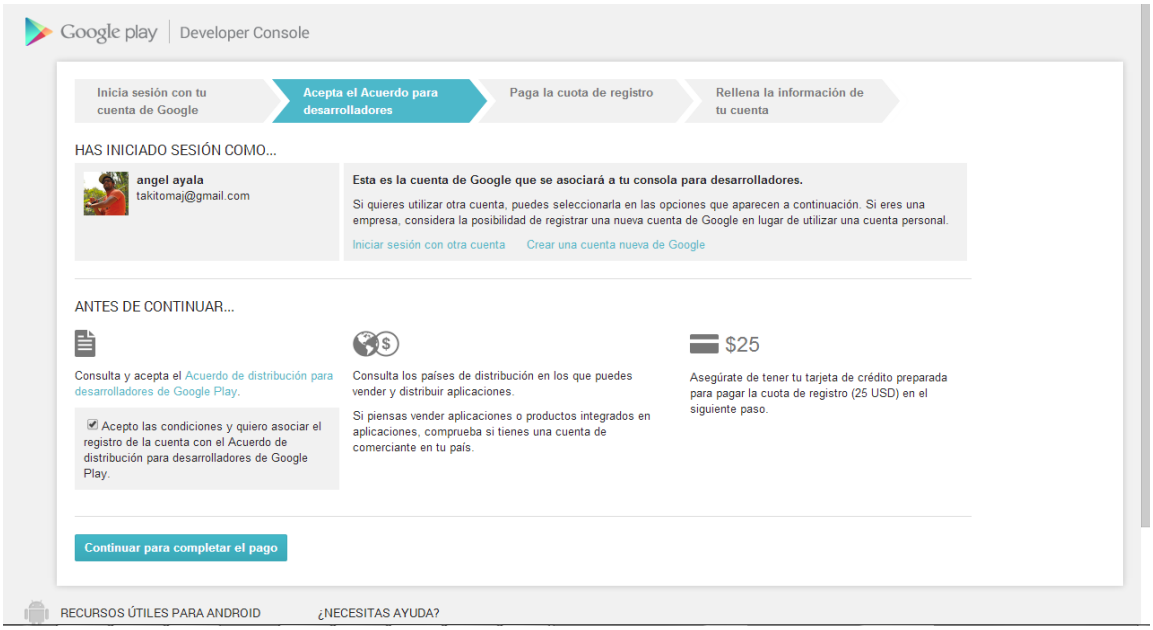
22. *Whats an API?*. [en línea]. <https://www. youtube. com/watch?v=UcHhwsTIK_o>. [Consulta: mayo de 2014].

APÉNDICES

Apéndice 1. **Creación de una cuenta de desarrollador en Google Play Store**

Los pasos para crear una cuenta de desarrollador son los siguientes :

- Ingresar a: https://play.google.com/apps/publish/
- Acceder con credenciales de Gmail.
- Aceptar las políticas del sitio.

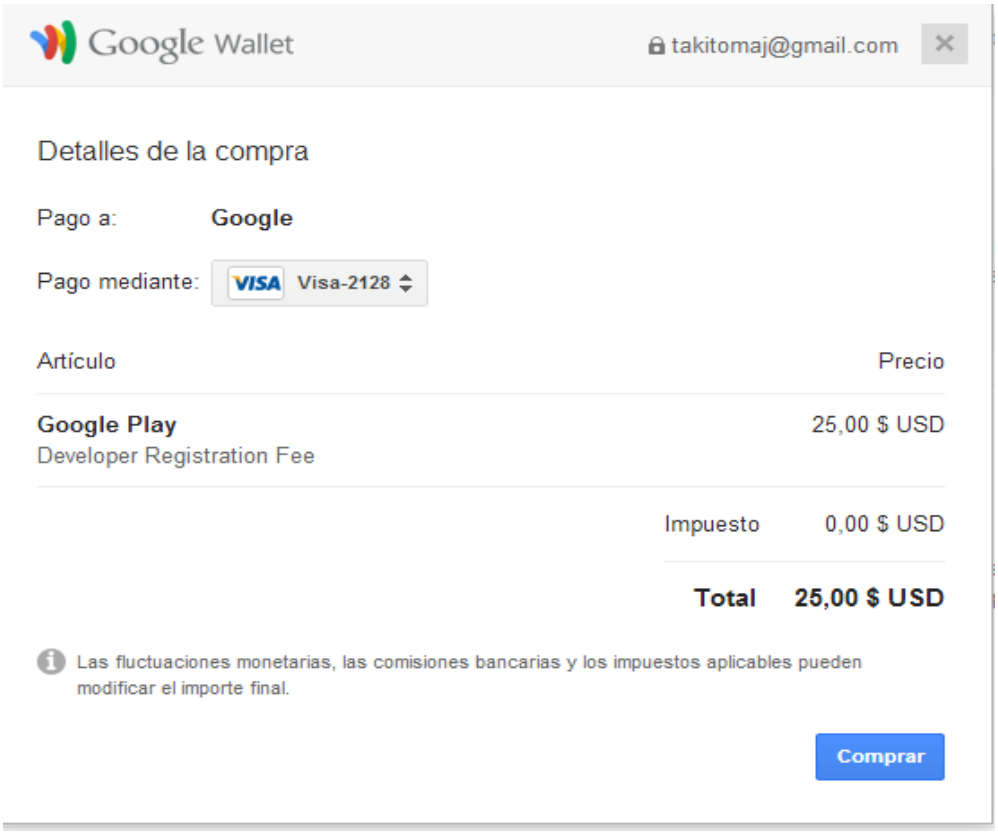


Continuación de apéndice 1.

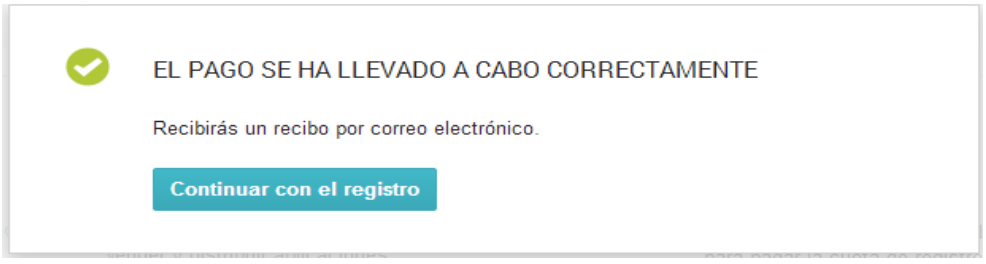
- Realizar un pago de $ 25,00. Esta transacción puede hacerse por varias vías:
    - PayPal (tarjeta electrónica)
    - Cargo a tarjeta de crédito (Visa, Master Card)

Fuente: elaboración propia, empleando HTML5.

Apéndice 2. **Ingreso de modo de pago**



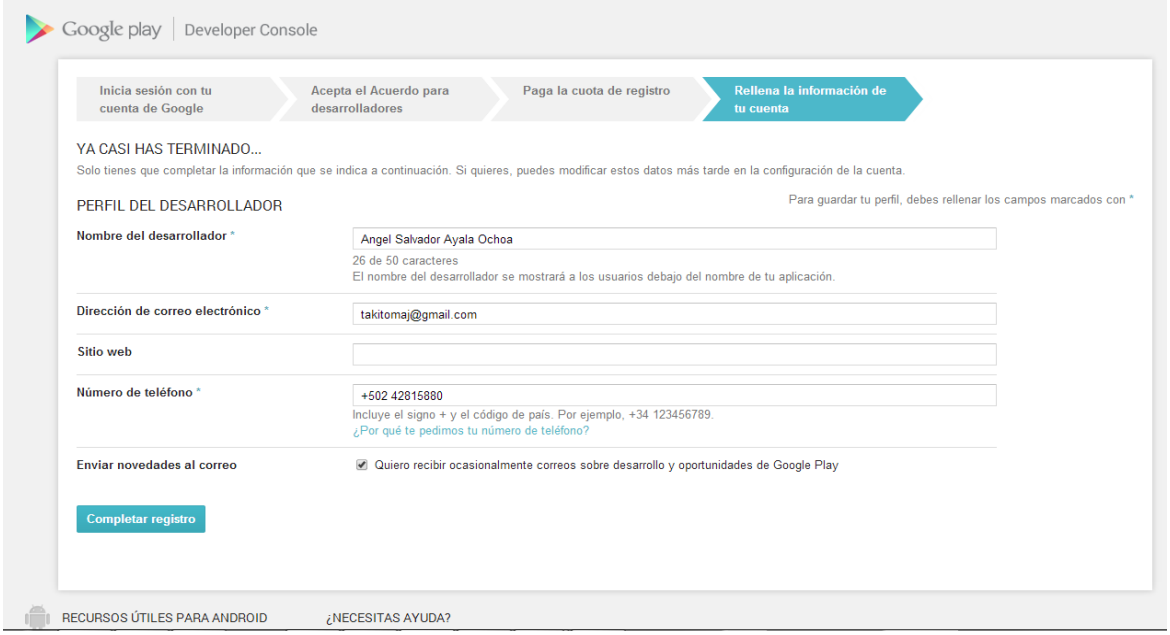
- Si la transacción es exitosa, se mostrará el siguiente mensaje de confirmación:



- Presionar el botón continuar.

Continuación de apéndice 2.

- Completar los datos en el formulario correspondiente al perfil de desarrollador.



Fuente: elaboración propia, empleando HTML5.