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

UNA APLICACIÓN ENFOCADA EN LA BÚSQUEDA DE CÓDIGO FUENTE, UTILIZANDO LAS API DE BÚSQUEDA DE GOOGLE

Carlos Manuel Samayoa Ramos

Asesorado por el Ing. Victor Adolfo González García

Guatemala, junio de 2010

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

UNA APLICACIÓN ENFOCADA EN LA BÚSQUEDA DE CÓDIGO FUENTE, UTILIZANDO LAS API DE BÚSQUEDA DE GOOGLE

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR:
CARLOS MANUEL SAMAYOA RAMOS

ASESORADO POR EL ING. VICTOR ADOLFO GONZÁLEZ GARCÍA

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JUNIO DE 2010

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Inga. Glenda Patricia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Luis Pedro Ortíz de León
VOCAL V	Br. José Alfredo Ortíz Herincx
SECRETARIA	Inga. Mayra Grisela Corado

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADORA	Ing. Pedro Pablo Hernández Ramirez
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. Juan Álvaro Díaz Ardavin
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

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

UNA APLICACIÓN ENFOCADA EN LA BÚSQUEDA DE CÓDIGO FUENTE UTILIZANDO LAS API DE BÚSQUEDA DE GOOGLE,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, en marzo de 2009.

Carlos Manuel Samayoa Ramos

Guatemala, 01 de marzo de 2010

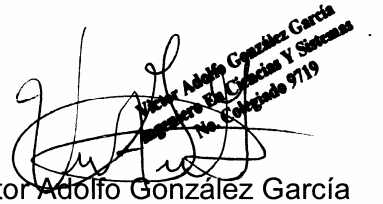
Ingeniero
Carlos Azurdia
Revisor de Trabajo de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería

Respetable Ing. Azurdia:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante CARLOS MANUEL SAMAYOA RAMOS, titulado: "UNA APLICACIÓN ENFOCADA EN LA BÚSQUEDA DE CÓDIGO FUENTE UTILIZANDO LAS API DE BÚSQUEDA DE GOOGLE", y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Sin otro particular, me suscribo de usted.

Atentamente,



Víctor Adolfo González García
Ingeniero en Ciencias y Sistemas
Colegiado No. 9719
Asesor de Trabajo de Graduación

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

Guatemala, 24 de Marzo de 2010

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 del estudiante **CARLOS MANUEL SAMAYOA RAMOS,** titulado: **"UNA APLICACIÒN ENFOCADA EN LA BÙSQUEDA DE CÒDIGO FUENTE UTILIZANDO LAS API DE BÙSQUEDA DE GOOGLE",** 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 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 ***"UNA APLICACIÓN ENFOCADA EN LA BÚSQUEDA DE CÓDIGO FUENTE, UTILIZANDO LAS API DE BÚSQUEDA DE GOOGLE"****, presentado por el estudiante CARLOS MANUEL SAMAYOA RAMOS, 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 Turk
Director, Escuela de Ingeniería Ciencias y Sistemas

Guatemala, 21 de junio 2010

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.236.2010

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; **UNA APLICACIÓN ENFOCADA EN LA BÚSQUEDA DE CÓDIGO FUENTE, UTILIZANDO LAS API DE BÚSQUEDA DE GOOGLE,** presentado por el estudiante universitario **Carlos Manuel Samayoa Ramos,** autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
DECANO

Guatemala, junio de 2010

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/cc
c.c. archivo.

AGRADECIMIENTOS A:

Mis padres	Por darme la vida, el cariño, afecto y valores que han fortalecido nuestro núcleo familiar a lo largo de nuestras vidas. Gracias a su tenaz y perseverante apoyo durante mi carrera universitaria.
La USAC	Por ser mi alma máter en donde aprendí a contemplar con otros ojos el extraordinario y fascinante universo del que somos parte.
Mis catedráticos	Por su grandiosa labor de compartir su conocimiento y ser formadores de profesionales. Gracias por enseñarme que las ciencias son sin duda una de las grandes maravillas y más preciados tesoros de la humanidad.
Mis asesores	Por brindarme sus invaluables conocimientos, consejos y apoyo profesional en la realización de este trabajo.
Mis amigos	Con quienes tuve la oportunidad de forjar una amistad tan estable y fuerte como un átomo de hierro. Y quienes me apoyaron y alentaron en los momentos difíciles.

ACTO QUE DEDICO A:

La memoria de

Sir Issac Newton
Johannes Kepler
Galileo Galilei
Albert Eistein
Niels Henrik David Bohr
James Clerk Maxwell
Werner Karl Heisenberg
Max Karl Ernest Ludwig Planck
Erwin Schrödinger
Richard Phillips Feynman

Todos estos científicos brillantes que han dejado su huella en la historia de la humanidad. Quienes en su viaje en este universo alrededor de nuestra estrella, realizaron sin duda notables descubrimientos que han permito revelar las intrigantes leyes que orquestan nuestro maravilloso y bello universo. Todas estas mentes geniales que han sido mi fuente de inspiración y de sabiduría en mi vida.

Mis padres.

Mis hermanas.

Mis amigos.

El pueblo guatemalteco, por darme el privilegio de haber estudiado en la reconocida Universidad de San Carlos de Guatemala.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

EJEMPLOS

LISTA DE SÍMBOLOS

Símbolo	Significado
%	Porcentaje
PR	Rankin de página (Pagerank)

GLOSARIO

AJAX Es un acrónimo de la palabra en inglés *Asynchronous JavaScript And XML*. Esta combina una técnica de comunicación asíncrona entre un explorador y el servidor. Y a su vez utiliza tecnología de *JavaScript* y XML, lo que provee un desarrollo fácil de aplicaciones Web con interfaces ricas en componentes visuales.

API Interfaz de programación de aplicaciones (*Application Programming Interface*, por sus siglas en inglés). Es un conjunto de funciones, métodos o procedimientos encapsulados en una librería que brinda servicios específicos.

Carbon Una de las principales API de los sistemas operativos *Mac OS*.

CVS *Concurrent Versioning System*, es un sistema de control de versiones de archivos en un repositorio de datos. Lo que permite que más de una persona pueda trabajar un mismo archivo y llevar un control de cambios e integración del mismo.

DOM Es un modelo de objetos para la representación de documentos. Estos permiten definir y acceder a la estructura de documentos HTML y XML.

Eclipse En desarrollo de software, es conocido como una herramienta de desarrollo integrado, multiplataforma y de código abierto, desarrollado nativamente en lenguaje Java.

ER En ingeniería de software una expresión regular es un patrón que se construye con caracteres especiales y operadores. La cual describe la secuencia de los elementos que una cadena puede contener como caracteres válidos.

Feed En ciencias de la computación, generalmente en Internet se refiere a fuente Web. Su estándar es el lenguaje XML. Y a la cual un usuario se suscribe para recibir actualizaciones sin necesidad de volver a visitar dicha fuente.

Framework Son un caso especial de librerías de código, en los cuales se encuentra un conjunto de código genérico abstraído el cual es reusable. Este provee funcionalidad genérica para el desarrollo de otros programas o software.

Google Empresa fundada el 4 de septiembre de 1998 en *Melon Park*, CA, Estados Unidos. Es propietaria del motor de búsqueda con el mismo nombre. Se puede definir a Google como el motor de búsqueda más grande de Internet.

Googleware	Conjunto de herramientas y servicios producidos por Google que pueden ser utilizados para buscar, localizar, publicar información. Se pueden mencionar algunas como *Google Maps*, *Google Video*, *Google Search*.
HTTP	Protocolo utilizando en toda transacción vía Internet para la transferencia de hipertexto.
IDE	Popularmente en desarrollo se refiere al Entorno de Desarrollo Integrado. Es un programa el cual provee un conjunto de herramientas que facilitan la programación y codificación de sistemas.
IEEE	Son las silgas con las que se identifica al Instituto de Ingenieros Electricistas y Electrónicos. Institución técnico-profesional encargada de administrar los estándares relacionados a las áreas de electrónica, como la ingeniería de sistemas.
Infoware	Término utilizado para relacionar información con el software. Es la administración de la información que un usuario provee a un sistema mediante la interacción con el mismo. Permitiendo descubrir y generar tendencias de comportamiento y preferencias de usuario.
JDK	Kit de desarrollo de Java (*Java Development Kit* por sus siglas en inglés). Es el compilador y conjunto de librerías para desarrollar software en lenguaje Java.

Lock Se refiere al mecanismo de cierre de exclusión mutua de un recurso. De tal forma que cuando se necesita un recurso, éste es otorgado únicamente si no se encuentra reservado por otro sistema, de lo contrario será denegado. Cuando se otorga, se reserva y se dice que tiene *lock*. El recurso puede ser liberado únicamente por el sistema al que fue otorgado.

Plugin Se puede traducir al español como complemento. Este es una aplicación que se agrega y se ejecuta sobre una aplicación principal la cual interactuar por medio de un conjunto de API.

SDK En ingeniería de software, se le conoce como Kit de Desarrollo de Software. Es un software que provee frameworks, librerías y API para el desarrollo e integración de sistemas.

Singleton Es un patrón de diseño de software que permite restringir las instancias de objetos que pueden ser creadas de una clase. Este es encargado de proveer únicamente una instancia global que permite que dicha instancia sea la misma para todos los recursos que la necesiten en un sistema.

SOA En ingeniería de software, es un patrón de arquitectura de sistemas distribuidos el cual sigue la filosofía de una arquitectura orientada a servicios. Ampliamente utilizado en sistemas distribuidos que proveen y consumen

servicios através de Internet.

SSH En sistemas operativos Linux y Unix principalmente se refiere al programa y protocolo que permite comunicaciones seguras, mediante la encriptación de datos.

Swing Es una librería que fue desarrollada por *Sun Microsystems*, que provee un conjunto de elementos gráficos para desarrollar complejas interfaces de tipo cliente-servidor en lenguaje Java.

SWT *Standard Widget Toolkit*, es el conjunto de componentes para la construcción de interfaces gráficas en Java sobre la plataforma de desarrollo Eclipse.

URL Del inglés *Uniform Resource Identifier*, o identificador uniforme de recurso. El cual indica el nombre de un recurso como una página, servicio, dirección de un servidor, dirección de correo el cual es accesible mediante una red o un sistema.

Win32 Es una de las versiones más modernas de la API de 32 bits los sistemas operativos Windows de Microsoft, las cuales se encuentran escritas en lenguaje C.

XML Es un metalenguaje basado en etiquetas extensibles ampliamente utilizando y adoptado como un estándar para compartir información en Internet.

RESUMEN

Google inició como un motor de búsqueda caracterizado por su elegante y complejo algoritmo *PageRank* desarrollador por Lawrence Page y Serger Brin, dos estudiantes de post-grado en la Universidad *Standford* en California. A Page por una parte le interesaba la estructura de la *World Wide Web* y como se entrelazaban las páginas para poder analizar su relevancia.

De forma que en marzo de 1996 decidió iniciar un proyecto al que llamó *BackRub*, el objetivo principal de este proyecto era determinar la cantidad de enlaces entrantes que una página tiene, es decir, la cantidad de enlaces que una página tiene desde otras páginas. Para realizar esto, Page diseñó y desarrolló un sistema.

Este sistema tenía como función principal analizar los enlaces de cada página visitada y alimentar la base de datos en donde se indexan todas las todas las páginas de Internet. De hecho este mismo sistema siguen funcionando e indexando páginas solo que sobre una arquitectura distribuida alrededor del mundo. De modo que una vez cumplido el objetivo de *BackRub* (el cual todavía se puede encontrar en http://www.archive.org/) surgió el problema de clasificar la importancia de cada una de las páginas.

A Brin le interesó mucho este problema de tal forma que junto con Page desarrollaron una función para asignar relevancia a las páginas, según la cantidad de enlaces entrantes. La idea básica de esta función era dividir el número de enlaces entrantes de una página entre el número de enlaces

saliente de esa misma página. Pero se debe que notar que cada enlace saliente de igual forma tiene su propia relevancia de forma que el cálculo requiere un elevado número de operaciones recursivas.

En cierta forma esta es la base del funcionamiento del algoritmo detrás de los sistemas de búsqueda de Google, claro actualmente tiene muchas modificaciones y adecuaciones que soportan el modelo en que funciona dicha empresa. El funcionamiento de este algoritmo y otros detalles se podrán encontrar en el primer capítulo de este documento de investigación. De igual forma se detallan otras funcionalidades y API de datos de que provee Google para realizar búsquedas utilizando la tecnología *Googleware*.

OBJETIVOS

General:

Integrar los API de datos de búsqueda de código de Google para elaborar una aplicación de búsqueda de código fuente sobre repositorios de código.

Específicos:

1. Explicar el funcionamiento de las API de búsqueda de código de Google para realizar búsquedas de código fuente.

2. Explicar y definir los conceptos fundamentales de los sistemas de versionamiento de código fuente relacionados con su integración con los repositorios de Google.

3. Explicar y definir los parámetros principales relacionados en las consultas de búsqueda de código fuente sobre las API de Google.

4. Investigar y establecer una conexión a los servicios de las API de datos, para realizar búsquedas utilizando los servicios de Google.

5. Explicar conceptos técnicos avanzados sobre la plataforma y arquitectura de Eclipse para el desarrollo de plug-ins.

6. Desarrollar un plug-in en la plataforma de Eclipse utilizando las librerías cliente Java de Google, para elaborar búsquedas de código fuente.

ALCANCE

Alcances

1. Investigación del funcionamiento de las API de búsqueda de código de Google, para búsquedas de código fuente.

2. Utilizar e integrar las API y servicios de búsqueda de código de Google, para elaborar un sistema con capacidad de búsquedas de código sobre un repositorio código fuente.

Límites

1. Se limita el estudio únicamente de las API de búsqueda de código de Google.

2. El desarrollo del caso práctico se limita a utilizar la arquitectura de plug-in de Eclipse y las API de búsqueda de código de Google, para el desarrollo de un componente que permite realizar búsquedas de código.

INTRODUCCIÓN

Se podría pensar que los motores de búsqueda son una tecnología que pronto pasará de moda. Pero recordemos exactamente porque es que los motores de búsqueda se han vuelto tan populares en las tecnologías de información, al punto que han cambiado la forma en que actualmente se accede la información. En un principio cuando Internet era un proyecto de investigación, ni si quiera existían los navegadores de Internet. No fue si no hasta el año de 1993 cuando Marc Andressen, mientras se graduaba en la Universidad de Illinois tuvo la genial idea de construir un software al que llamo *Mosaic*, primer navegador en el mundo. Luego *Mosaic* se convirtió en el principal activo de Netscape Communications Inc., empresa financiada e iniciada por James Clark, quien fue presidente de Sillicon Graphics, en Sillicon Valley.

Cuando emergió Netscape y Microsoft Internet Explorer, fue claramente el inicio de una nueva etapa en las tecnologías de información. Se definieron lenguajes y estándares para la publicación de contenido, y este se ha incrementado desde entonces. Actualmente en Internet se puede encontrar información en muchos formatos; páginas, documentos, imágenes, videos, archivos de código fuente, etc. Y para cada tipo de documento, tal y como lo ha hecho Google, se necesita una búsqueda especializada, pues cada tipo de información requiere diferentes parámetros de búsqueda. El presente trabajo de investigación se enfoca en la búsqueda de código fuente en repositorios de código, iniciando esta con la explicación de la columna vertebral de Google, el algoritmo *PageRank*.

1. MARCO CONCEPTUAL

1.1. Antecedentes del problema

- En desarrollo de software, el localizar piezas de código fuente o librerías puede ser una tarea ardua y difícil. El primer lugar de apoyo para localizar alguna información, sería la documentación de referencia del API, framework o librería en la versión correcta.

- Es por ello que los desarrolladores en grandes proyecto de software utilizan sistemas de versionamiento de código, para administrar los cambios y las versiones del código. Pero estos sistemas se enfocan en eso y nada más, en versionar el código. Qué pasa cuando se desea localizar una clase, función o procedimiento y no se recuerda el nombre del archivo o librería en la cual se encuentra. Y menos las versión.

- Se dificulta localizar funcionalidad específica en grandes repositorios de código fuente. Y lo que puede ser aún peor se puede hacer difícil encontrar vulnerabilidades de seguridad, errores o funciones obsoletas en código o librerías para los desarrolladores que no cuentan con herramientas de búsqueda de código en grandes repositorios de código.

1.2. Justificación del problema

El localizar de forma rápida y precisa una pieza de código fuente o librería para un desarrollador de software es muy útil debido a que ayuda a mejorar su productividad y evitar los errores de programación. A veces por no saber la sintaxis correcta, firma o los parámetros adecuados de una función se

cometen errores en programación. Los cuales después provocan revisiones del sistema, tiempo de depuración y correcciones, que al final se traducen retrasos y mayores costos en el desarrollo del software.

Gracias a los sistemas de versionamiento de código, ahora es posible administrar de mejor manera las grandes cantidades de líneas de código de un sistema. Pero como ya sabemos, los repositorios de grandes proyectos de software tienden a crecer mucho y pueden llegar a contener miles y miles de líneas de código y archivos. Por lo que buscar funcionalidad especifica en estos repositorios puede llegar a ser tediosa e infructuosa.

1.3. Descripción del problema

El motor de búsqueda de Google, tiene la capacidad de indexar código fuente, debido a que permite registrar direcciones de servidores en los cuales se alojan repositorio de código como Subversión o CVS.

De tal cuenta se propone la elaboración de una aplicación que mediante el registro de un repositorio de código como Subversión o CVS, se pueda elaborar un componente que utilice los servicios de las API de datos de Google, para proveer a los desarrolladores de software una herramienta que les permita localizar de forma más eficiente, funcionalidad especifica en repositorios de código fuente.

Esta aplicación tendrá que ser diseña siguiendo la arquitectura de plug-in, la cual permite crear extensiones sobre el entorno de desarrollo integrado Eclipse. De igual forma se deberá seguir el patrón de arquitectura SOA para tener un desacople bajo del servicio ya que este puede cambiar en el tiempo.

Uno de los principales retos será la integración de estos API como una extensión sobre Eclipse.

1.4. Delimitación del problema

El presente trabajo de investigación se limita al estudio y documentación de las API de datos de búsqueda de código únicamente.

Dado que las API de datos, se encuentran desarrollados en varios lenguajes, se seleccionarán las librerías de cliente escritas en lenguaje Java debido a la aplicación se integrará como un punto de extensión de la plataforma Eclipse, el cual se encuentra escrito en código nativo de este mismo lenguaje.

El desarrollo del caso práctico por ende, será desarrollado completamente el lenguaje Java, utilizando las API anteriormente descritas y otras librerías utilitarias requeridas por los servicios de Google para realizar conexiones remotas. Las cuales permitan el desarrollo de un servicio de búsqueda de código fuente en repositorios de código, el cual será integrado al IDE de desarrollo para proveer de una herramienta útil a desarrolladores de software.

1.5. Resultados esperados

Proveer al lector una referencia para la utilización de API de datos Google para búsquedas de código fuente.

Desarrollar un caso práctico el cual sirva como guía para desarrolladores que necesiten integrar o implementar las funcionalidades y servicios brindados por Google, así como desarrollar puntos de extensión de Eclipse.

2. MARCO TEÓRICO

2.1. El Big Bang de Google

2.1.1. Historia del algoritmo PageRank

Lawrence Page y Serger Brin desarrollaron un algoritmo que les permitiría calcular la relevancia de una página siguiendo la filosofía de 'citas'. Es decir basado en que la página más relevante, es la que tiene más enlaces entrantes o bien enlaces desde otras páginas.

Este algoritmo fue denominado *PageRank*. Para explicar su funcionamiento, se asume por un momento que se tiene una página, esta página tiene asociada un *URL*. Llamaremos a esta página 'A' y tiene de T1…Tn enlaces entrantes. Es decir, enlaces o citas desde otras páginas.

Existe un parámetro que se identifica con la letra 'd' (del inglés *damping*). Este es un parámetro de amortiguación que representa la probabilidad de que un usuario que navega sobre una página, pulse un link en una página para seguir navegando en vez de escribir la dirección URL en la barra de direcciones del navegador. Dado que es una probabilidad, puede tener un valores entre cero y uno. Y debido a que es una variable con distribución uniforme, es decir con un dominio definido. Según experimentos ya realizados anteriormente indican que este valor es de 0.85 o del 85% de probabilidad.

La función $C(T_i)$ representa la cantidad de enlaces saliente de la página T_i. De forma que el *PageRank* de la página A esta dado por la sumatoria del

pagerank de cada uno de los enlaces entrantes, tal y como se ilustra en la función de abajo.

Figura 1. Fórmula para calcular pagerank

$$PR(A) = (1-d) + d * (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

$$PR(A) = (1-d) + d * \left[\sum_{i=1}^{i=n} PR(Ti)/C(Ti) \right]$$

Pero fue claro para ambos, Page y Brin que este algoritmo aparte de permitir asignar una relevancia a una página, permitía realizar búsquedas de páginas por toda la *World Wide Web* con resultados mucho más precisos que otros "buscadores" como Yahoo y Excite que en esos momentos estaban luchando por el primer puesto en la emergente economía de información, pues recién estaba naciendo el nuevo modelo económico de vender publicidad por Internet.

Mientras Yahoo.com inicio como un proyecto también de *Standford* por los emprendedores estudiantes Jerry Yang y David Filo. El principal objetivo de *Yahoo* era categorizar enlaces de páginas. La categorización era totalmente manual mediante la búsqueda de links para posteriormente ser agregados a una categoría, esta era la parte fácil. La parte para nada difícil pero tediosa era buscar las páginas, pues Yang y Filo pasaban todo el día buscando páginas en toda la Internet para luego colocar estos enlaces en alguna de las categorías de su sistema. Y Excite que era la sombra de Yahoo, pues cada servicio nuevo que lanzaba Yahoo, Excite lo copiaba casi de igual forma.

Aunque en un principio Google no contaba con un definido y formal modelo de negocio su éxito se debido en parte a la introducción de publicidad sobre contenido y no publicidad invasiva.

Cuando Google junto con esto desarrollo los servicios *AdSense* y *AdWords* simplemente atrajo mucho la atención de los medios publicitarios. Debido a que la publicidad sobre contenido era sin duda más efectiva que la publicidad invasiva. Otra de las estrategias que dieron mucha ventaja a Google fue el correcto manejo y análisis de información mediante sus sistemas, conocido como "*infoware*". Y la escalabilidad de la arquitectura de sus sistemas distribuidos le permitido a Google convertirse en toda una plataforma como lo es actualmente.

2.1.2. Aspecto generales de Google

Se ha podido observar como Google ha evolucionado en los servicios que ha brindando. Inicio con su motor de búsqueda, correo electrónico y ha evolucionado al punto de llevar aplicaciones del escritorio a la *Web* con aplicaciones como *Google Docs*, *Google Calendar* y *Google Maps*. La arquitectura de clúster de Google en donde sus centros de datos se encuentran distribuidos alrededor del mundo le ha permitido tener una escalabilidad inmensa al punto de convertirse en una plataforma de servicios para usuarios y empresas.

Actualmente Google ha liberado una gran cantidad de código fuente y un conjunto muy diverso de librerías o interfaces de programación de aplicaciones (API) para que los desarrolladores puedan ejecutar peticiones a los servidores de Google desde sus propias aplicaciones. Estas aplicaciones son utilizadas tanto en el desarrollo de software web como aplicaciones para escritorio. Entre las API que provee Google actualmente se encuentran:

1. *Google Calendar API*
2. *Google Code Search API*
3. *Google Docs API*
4. *YouTube API*
5. *Google Maps API*
6. *Google Base API*
7. *Google Piccasa API*

2.2. Sindicación web y feeds

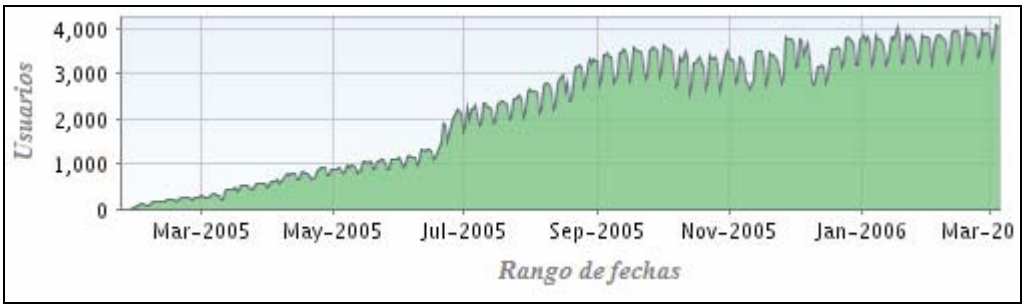
La sindicación o bien redifusión se refiere a la disposición de sitios o páginas para distribuir contenido informativo a los suscriptores de un feed o una fuente de alimentación. De forma que un proveedor de redifusión de contenido únicamente debe de proveer un URL que es la fuente del feed.

De forma que los suscriptores de una fuente deben de contar con una aplicación ya sea web o de escritorio para leer los nuevos contenidos publicados por la fuente de sindicación. Uno de los principales objetivos de este formato es evitar visitar un sitio para encontrar información actualizada. En vez, el sitio cuando tiene información actualizada la pública y llega a todos los suscriptores de esa fuente.

Estas fuentes por normalización son escritas o codificadas en lenguaje de etiquetas XML. Aunque puede ser cualquier otro que pueda transportarse por medio del protocolo HTTP. Hay algunos casos en lo que es un hibrido, XML con HTML y *JavaScript*, o simplemente solo XML. La redifusión de contenido se hizo necesaria y popular con los sitios de noticias y el surgimiento de blogs o bitácoras de contenido. Pero ahora es muy popular, pues existe gran cantidad de sitios web y con ellos gran cantidad de contenido que se actualiza

diariamente. Tal es el ejemplo de videocast en sitios como YouTube. Contenido de información como Wikipedía, etc. Abajo se podrá observar una gráfica que muestra la cantidad de usuarios que han utilizado RSS para recibir contenido de *WikiNews* entre los años del 2005 al 2006.

Figura 2. Utilización de sindicación y RSS sobre Internet



Fuente: http://es.wikipedia.org/wiki/Imagen:En_wikinews_rss_circulation.png

Los dos formatos principales para redifusión de contenido son el RSS y el Atom, dos formatos que veremos a continuación.

2.2.1. Detrás de RSS

Es un formato de sindicación desarrollado por la organización *RssBoard*[1]. *Really Simple Sindication*. Como su nombre lo dice es un estándar simple de sindicación, basado en XML para difundir contenido. Este como otros formatos que se presentarán a continuación se basan en el protocolo feed.

La sindicación web o también conocida como redifusión web, permite a un usuario suscrito a una fuente RSS obtener actualizaciones de la fuente sin necesidad de volver a visitar dicha fuente con tan solo tener un lector de feeds, es decir un lector de fuentes. La extensión del archivo puede ser .rss o bien .xml. El tipo de extensión de correo de Internet multipropósito MIME (*Multipurpose Internet Mail Extensions*) es 'application/rss+xml'. Como se

observará posteriormente los API de datos de Google se basan en la plataforma de formatos de sindicación Atom 1.0 y RSS 2.0 para la publicación de contenido.

Dado que los feeds de los API de datos de Google pueden ser no solo consultados, si no también pueden ser creados, modificados y eliminados el formato RSS tiene varias limitaciones. Pues RSS solo soporta formato de sindicación. Es decir, redifusión que se reduce a solamente lectura. Es por ello que Atom es la versión mejorada de RSS que veremos a continuación.

2.2.2. Detrás de Atom

Atom es un formato de sindicación desarrollado por el grupo '*atompub*' de IETF (del inglés *Internet Engineering Task Force*). El Grupo de Trabajo en Ingeniería de Internet es una organización con el principal objetivo de normalizar las tecnologías e ingeniería de protocolos de comunicación y arquitectura de Internet.

De modo que Atom es de igual forma un estándar XML utilizado para redifusión web. Ben Trott quien fue uno de los impulsadores también fue uno de los que pudo notar cierta incompatibilidad en versiones de RSS 0.9 y 1.0 debido a que no eran lo suficientemente ínter operables.

De forma que la IETF definió así un nuevo formato y un nuevo protocolo. Este se llamo *Atom Publishing Format and Protocol*. El formato es Atom y el protocolo APP del inglés *Atom Publishing Protocol*. Un protocolo basado en HTTP para crear y actualizar contenido web.

La extensión del archivo puede ser .atom o bien .xml. El tipo MIME es 'application/atom+xml'. Los estándares de este formato son el RFC 4287[2] y RFC 5023[3]. Una de las mejoras a este formato de sindicación web y al protocolo es que permiten actualizaciones de contenido

2.2.3. Detrás de Google Data

Google data o bien GData por abreviación es un protocolo basado en los formatos de sindicación que se explicaron anteriormente; Atom 1.0, RSS 2.0 y en el protocolo de publicación de contenido *Atom Publishing Protocol* (APP).

GData pretende ampliar estos estándares de varias maneras a través de un mecanismo de extensión creado dentro de ellos. Los feeds GData se ajustan al formato de sindicación Atom o RSS. El modelo de publicación GData se ajusta al protocolo de publicación de contenido *Atom Publishing Protocol*.

Todos los servicios pueden proporcionar feeds de GData, desde servicios públicos como feeds de noticias y feeds de sindicación de video hasta datos personalizados como eventos de calendario o de correo electrónico, o elementos de listas de tareas. Los modelos RSS y Atom son ampliables de forma que cada proveedor de feeds puede definir sus propias extensiones y semántica como se desee, siempre que se ajusten a los estándares.

Los proveedores de feeds pueden proporcionar tanto feeds de sólo lectura que son feeds de resultados de búsqueda. Como también feeds de escritura como por ejemplo para las aplicaciones de calendario, dada una invitación permitir la confirmación o denegación, de cualquier de las dos formas es una escritura.

Dado que Google Data se ha creado a partir de una tecnología básica con los modelos de sindicación estándar y HTTP, se puede enviar solicitudes Google Data y procesar los feeds resultantes de formas distintas:

1. Lectores de feeds o agregadores de sindicación tradicionales.
2. Clientes basados en *JavaScript* o AJAX en un navegador web.

El protocolo Google Data utiliza un lenguaje de programación neutro; se puede escribir en un cliente en cualquier lenguaje de programación que permita enviar solicitudes HTTP y procesar respuestas basadas en XML. El servicio de Google Data es el encargado de crear y eliminar feeds ya que el protocolo Google Data no lo permite de ninguna forma.

Cabe resaltar dos características sobresalientes de este protocolo con respecto a Atom 1.0 y RSS 2.0. Este protocolo permite autenticación y resuelve los problemas de control de concurrencia de forma optimista.

La autenticación que provee este protocolo es un enfoque útil cuando un usuario necesita acceder a un servicio, pero para ello debe de proporcionar credenciales para la utilización del servicio. Esto es necesario para las aplicaciones basadas en cliente de escritorio debido a que estas necesitan autenticación para las aplicaciones instaladas, también conocidas '*ClientLogin*'.

Los servicios de datos de Google utilizan autenticación de cuenta por proxy, también llamado '*AuthSub*'. Estas son utilizadas para aplicaciones basadas en un cliente web que corre sobre un navegador. La ventaja de utilizar el sistema *AuthSub* es que en lugar de preguntar al usuario sus credenciales, conecta al usuario a un servicio de Google que requiere credenciales o autenticación. El servicio devuelve una toquen que la aplicación web puede

utilizar. Otra ventaja de este enfoque es que Google se ocupa de forma segura y almacena las credenciales del usuario, en vez que de manejar y controlar en la interfaz web de una aplicación. El proceso de autenticación es exitoso cuando el sistema retorna un toquen de autorización, en caso de aplicaciones web, se retorna un encabezado HTTP.

El control concurrencia optimista (CCO), para el protocolo GData es importante debido a que permite garantizar la lectura de feeds integrales. También permite garantizar no sobrescribir cambios realizados por otros usuarios. El método optimista se basa en versionamiento de los feeds. Y antes de actualizar un feed, se verifica que la versión del feed a actualizar sea la misma que la versión del feed en el sistema. De esta forma si la versión es la misma, la modificación puede realizarse satisfactoriamente sin ningún problema. En caso contrario, se deberá de actualizar la versión del feed y reintentar modificar la entrada.

Esto es una facilidad que brindan los feeds de datos de Google. Al igual que otros marcos de trabajo que permite el manejo de concurrencia optimista como *Hibernate*. La manera más sencilla de manejarlo es mediante la asociación de un número de versión o identificador único de versión. Esto mejora mucho la semántica de cada feed pues se permite tener un total control sobre la evolución de las modificaciones y actualizaciones del feed en el tiempo.

Los feeds de datos de Google son tan variados como los servicios que brinda. Algunos feed no soportan CCO. Tal es el caso de feeds de actualizaciones, como feeds de noticias o entradas de blog. Feeds de datos de Google que si soportan CCO son feeds de *Google Docs* y *Google Calendar*. De forma que cuando se intenta utilizar CCO en feed no que no lo soporta, el

servicio detectará un conflicto de versión y responde con un error de conflicto con código 409.

De tal forma que en la tabla comparativa de abajo se puede visualizar las ventajas de GData con respecto a Atom 1.0 y RSS 2.0.

Tabla I. Comparación de funcionalidad de protocolos de sindicación

GData	Atom 1.0	RSS 2.0	Función
Sí	Sí	Sí	Sindicación web
Sí	No	No	Consulta
Sí	Sí	No	Actualización
Sí	No	No	Autenticación
Sí	No	No	Control concurrencia optimista

3. API DE DATOS DE BÚSQUEDA DE CÓDIGO

3.1. Nociones fundamentales

Las API de datos de Google para búsqueda de código son un conjunto de librerías que permiten desde otras aplicaciones utilizar el motor de búsqueda de Google para buscar definiciones de código fuente, tal como funciones, métodos, paquetes o definición de clases. Las aplicaciones pueden ser tanto clientes de escritorio como aplicaciones web.

Básicamente el funcionamiento de estas API se basa en la invocación de servicios remotos al servicio de búsqueda de datos de Google. Esto se realiza mediante las librerías cliente adecuadas según el lenguaje de desarrollo de la aplicación. Se podrá ver más adelante cuales son las librerías cliente disponibles por Google para integrarlas con diferentes lenguajes y plataformas.

Para interactuar con los datos de búsqueda de código de Google existen dos enfoques. Las aplicaciones pueden enviar una consulta manual. O bien se puede hacer utilizando una librería cliente. Si se sigue el enfoque de enviar una consulta manual, se deberá utilizar el protocolo HTTP mediante una petición de la dirección del servicio de búsqueda. Para descargar feeds de datos de código fuente de Google, se utiliza la dirección.

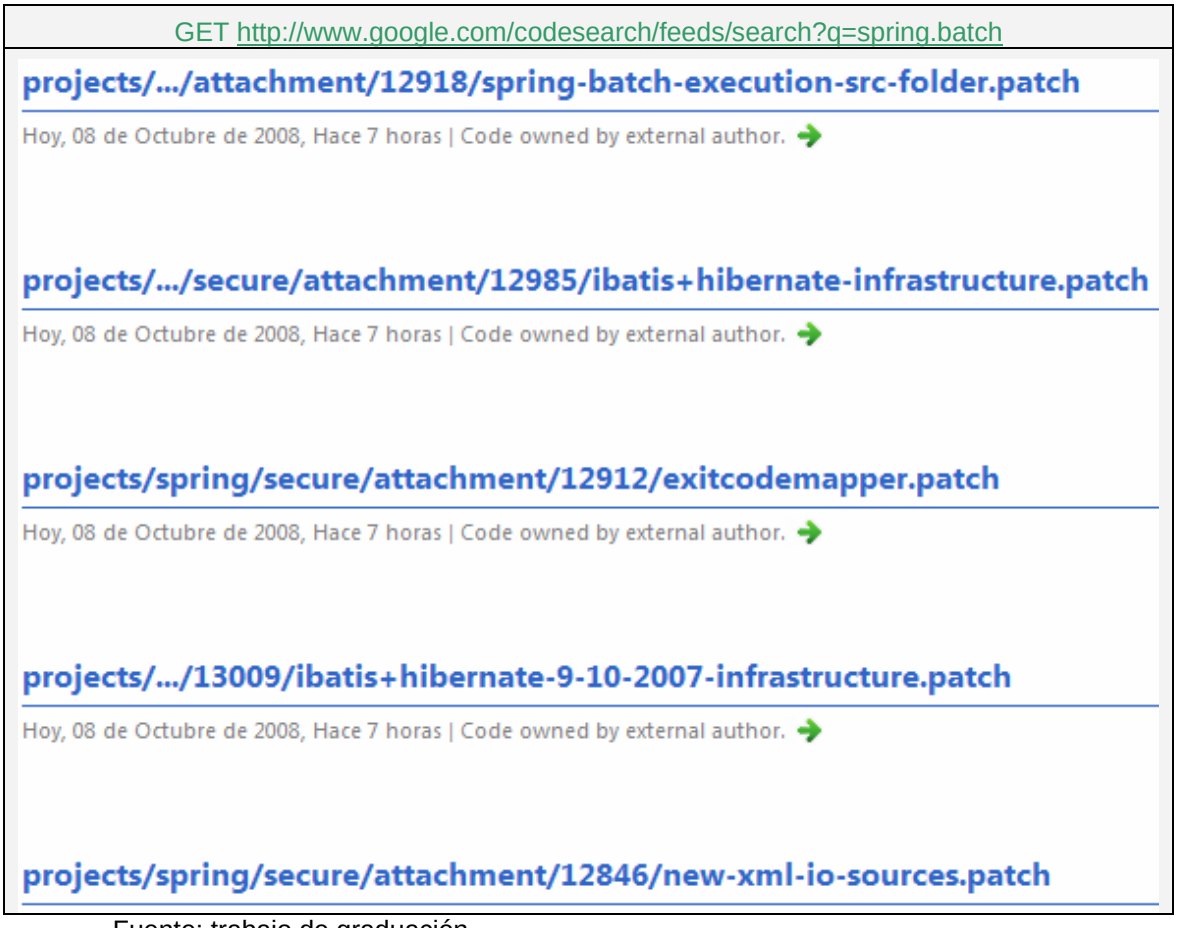
http://www.google.com/codesearch/feeds/search

Supongamos que se desea obtener una búsqueda sobre alguna actualización de un feed con alguna cadena ‘spring.batch’, la cadena enviada debería ser:

http://www.google.com/codesearch/feeds/search?q=spring.batch

Y el servidor devolverá todos los feeds de datos actualizados recientemente con las cadenas ‘spring’ y/o ‘batch’ que se encuentren en el código fuente. En un explorador se vería el resultado como se muestra en la siguiente figura.

Figura 3. Resultados de búsqueda obtenidos



Fuente: trabajo de graduación

Y sí se utiliza el enfoque de consultas mediante librerías de cliente en vez de enviar la consulta manualmente se deberá utilizar un servicio de búsqueda especifico. Esto debido a que Google brinda varios servicios de búsquedas. Como búsqueda en noticias, en bitácoras web, en calendarios. Para el presente caso se analizan búsquedas datos de código fuente. Si se trabaja con lenguaje Java, la clase *CodeSearchService* provee este servicio. Se explicará más a detalle esta clase pues es la base para realizar la conexión al servicio de búsquedas de código.

Luego de enviar la consulta al servidor, este retorna el resultado de los feeds actualizados en formato xml. A continuación se muestran los parámetros validos para realizar consultas más complejas y poder manipular los resultados enviados por el servidor.

3.2. Parámetros de consultas

Cuando se indica el primero parámetro para obtener resultados de feeds de datos de código de Google, se debe de utilizar el operador '?' después de la dirección *http://www.google.com/codesearch/feeds/search*.

Para agregar otro parámetro a la consulta se debe utilizar el operador '&' seguido del nombre del parámetro y el valor del mismo. Teniendo esto claro seguiremos a describir las características principales de los parámetros para los feeds de datos de código de Google.

3.2.1. Parámetro q

Este parámetro es una expresión regular de consulta. Este es igual que el parámetro q del estándar para los feeds de GData, pero para el caso del API

de datos búsqueda de código es interpretado como un conjunto de expresiones regulares extendidas de POSIX en lugar de solo cadenas de texto.

De forma que existen aspectos importantes a considerar para definir correctamente el parámetro q. Cuando se crea una lista de cadenas separadas por espacios, se toman como expresiones regulares individuales en donde la separación dada por el espacio significa la operación lógica *AND*. De la misma forma que funciona la búsqueda Web de Google.

Para obtener resultados de una expresión regular exacta se debe de encerrar entre comillas dobles. Por ejemplo q=”java.lang.*”. Para excluir resultados de una expresión se debe anteponer el operador -. Por ejemplo para excluir de una búsqueda todas la clases ‘Thread’ el parámetro deberá definirse de la forma q=-java.lang.Thread.

A diferente de los datos feeds de GData, este parámetro no distingue entre mayúsculas y minúsculas.

3.2.2. Parámetro start-index

Es el primer resultado del índice dado. Con este parámetro es posible indicarle al servicio de búsqueda obtener los resultado a partir del número n. Donde n debe ser un número entero mayor o igual a 1. Por ejemplo si se obtiene de una consulta diez resultados únicamente los resultados del seis en adelante, el parámetro deberá tener el valor start-index=6.

3.2.3. Parámetro max-results

Este parámetro indica el número máximo de resultados a recuperar del total devuelto por el servicio de búsqueda. Este al igual que el anterior el

parámetro n debe ser un número entero mayor o igual a 1. De forma que al utilizar este parámetro los feeds devueltos por el servicio de búsqueda no serán más que n. Por ejemplo si se quiere obtener únicamente cinco resultados de un total de cien, el parámetro deberá tener un valor max-results=5.

De forma que con estos tres parámetros básicos se pueden obtener resultados de actualizaciones de feeds de datos de búsqueda de código de Google. La aplicación principal de los parámetros start-index y max-results es principalmente para hacer paginación de los resultados. Y dado que el parámetro ‘q’ es tomado como una expresión regular este se puede extender con expresiones regulares de POSIX como se detallará más adelante.

3.3. Elementos de una entrada de código

Los elementos de una entrada de un feed de búsqueda de código se basan principalmente en los formatos de sindicación Atom y GData que vimos anteriormente. Estos elementos de referencia contienen importante información que permiten manipular los resultados obtenidos de una búsqueda. Se explicará a continuación cada uno de estos elementos.

El elemento <*atom:author*> es una cadena que indica el nombre del autor del código fuente. Debido a que Google ha publicado gran cantidad de código, se dar el caso en que el autor pueda ser Google. Pero en caso contrario este elemento tendrá otro valor. En caso de que el elemento no tenga ningún valor se aplica el valor del elemento de la etiqueta <*atom:source*>, esto debido a que así lo indica la especificación de Atom para el elemento *atom:author*.

Elemento <*atom:rights*> esta cadena tienen información relacionada a los derechos de autor. Este es un código que indica un tipo de licencia de la pieza de código fuente y la forma en que puede ser utilizada y redistribuida.

Elemento <*atom:title*> esta cadena contiene información relacionada a la fuente de publicación la cual es legible para un usuario. Esta cadena por lo regular contiene la dirección del archivo que fue localizado en el repositorio de control de versiones.

Para los elementos de Google Code Search que se basan en el formato GData utiliza un DOM que tiene la abreviatura gcs. Uno de los principales elementos es el <*gcs:match*> el cual contiene información sobre los segmentos exactos en donde el servicio de búsqueda ha encontrado una coincidencia con la expresión regular enviada por la consulta. Este elemento tiene el atributo type, que indica el tipo de texto con el cual se ha encontrado la coincidencia exacta. Regularmente tiene valores como ‘text’, ‘html’, ‘text/html’. Y el segundo atributo *lineNumber* indica el número de línea del archivo fuente donde fue localizado el código.

El elemento <*gcs:package*> tiene el atributo name que indica el nombre del paquete. Y el atributo uri que contiene información sobre la dirección especifica del paquete y también el control de versión del servidor o archivo.

El elemento <*gcs:file*> mediante el atributo name especifica el nombre del archivo con el control de versión en el repositorio o versión del archivo.

3.4. Extensión de los parámetros de búsqueda POSIX

Uno de los problemas que existe al utilizar expresiones regulares es que existen gran variedad de convenciones y herramientas para escribir expresiones regulares. De forma que las expresiones regulares (ER) de POSIX no es más que un estándar de la IEEE para regular y normalizar la sintaxis en los sistemas Unix. Es por ello que este mismo estándar es utilizado para las expresiones regulares que se utilizan en las consultas de datos de búsqueda de código de Google.

En la siguiente tabla de contenido se muestra la sintaxis, los caracteres y meta caracteres validos para las ER de POSIX, los cuales sirven para extender el parámetro q de las consultas de código fuente.

Tabla II. Caracteres de expresiones POSIX

Carácter	***Descripción***
.	Significa una coincidencia con cualquier carácter. Muchas aplicaciones excluyen los cambios de línea y exactamente los caracteres que son una nueva línea. En una expresión POSIX con corchete, el carácter '.' significa literalmente un punto dentro de la expresión. Por ejemplo la expresión a.c, coincide con expresiones como "abc" o "a1c". Pero la expresión [a.c] solo tendrá coincidencias con una cadena que sea "a", "." o "c".
[ ]	La expresión de corchetes, significa un solo carácter que se encuentre dentro de los corchetes. Por ejemplo para la expresión [abc], cadenas de texto como "a", "b" o "c" son coincidencias validas. Cuando se utilizan corchetes también es posible la definición de rangos de caracteres.

	Por ejemplo la expresión [a-z], significa que cualquier letra minúscula del abecedario es una cadena valida. También en la utilización de corchetes es posible mezclar rangos. Por ejemplo la expresión [abc-ex-z] acepta cualquier cadena con valor “a”, “b”, “c”, “d”, “e”, “x”, “y” o “z”. Y el carácter “-” será tomado como literal únicamente cuando se encuentre al inicio, al final o bien precedido de una contra diagonal. Por ejemplo [123-], [-123] o bien [12\-3] aceptará cualquier cadena con valor “1”, “2”, “3” o “-”.
[^]	Toma como coincidencia valida cualquier carácter que no se encuentre definido dentro de los corchetes. Por ejemplo para la expresión [^abc], se toma como caracteres validos cualquier que no sea “a”, “b” o “c”. Las reglas que vimos anteriormente se aplican a este caso, con la única diferencia es que esta expresión toma como validos lo caracteres que no se encuentren en los corchetes.
^	Indica la posición inicial de una cadena o bien la posición inicial de cada línea.
$	Indica la posición final de una cadena o la posición justo antes del inicio de una nueva línea.
\(\)	Permite definir una sub-expresión. La cadena que se encuentra dentro del paréntesis puede ser utilizada más adelante en la misma expresión. Este tipo de sub-

	expresión también llamada un sub-bloque.
\(\)	Permite definir una sub-expresión. La cadena que se encuentra dentro del paréntesis puede ser utilizada más adelante en la misma expresión. Este tipo de sub-expresión también llamada un sub-bloque.
\n	Encuentra la coincidencia para el n-ésimo sub-bloque o sub-expresión, en donde n es un digito de 1 a 9.
*	Esta expresión permite indicar que el elemento o grupo de elementos que preceden al símbolo '*' pueden existir en la cadena de 0 cero a muchas veces. Para la expresión ab*c, cadenas validas serían "ac", "abc", "abbc", etc. Este operador funciona también con expresión como corchetes y paréntesis. Por ejemplo la expresión [abc]*, acepta como entradas validas "a", "aa", "bb", "cc" incluso acepta una cadena vacía.
\{m,n\}	Esta expresión permite identificar cadenas en donde un carácter debe aparecer en una cadena de al menos m veces y un máximo de n. Este carácter precede a la expresión. Por ejemplo x\{3,5\}, encontrara como cadenas validas cadenas como "aaa", "aaaa" y "aaaaa".

Fuente: http://en.wikipedia.org/wiki/Regular_expression#Syntax

A continuación se muestran algunos ejemplos de aplicación que pueden ser útiles en la definición de una cadena para enviar al servicio de búsqueda de código.

1. “[^b]at” buscará todas las coincidencias con cualquier cadena que no sea “.bat”.
2. “^private” buscará todas las fuentes en donde la línea inicie con “private”.

Algunos metacaracteres de escape que se manejan con la diagonal invertida son reservados para ER de POSIX. Con ésta sintaxis la diagonal invertida causa que los metacaracteres sean tratados como literales, de forma que para eso existen los metacaracteres de agregación.

Tabla III. Metacaracteres de agregación

Metacarácter	***Descripción***
?	El elemento precedente es válido cuando se encuentra cero o una vez. Por ejemplo la expresión .html? acepta cadenas como “.htm” o bien “.html”.
+	El elemento precedente es válido cuando se encuentra una o varias veces. Por ejemplo la expresión ([a-z]+.)+Thread, acepta cadenas como “java.lang.Thread”, “edu.gt.usac.Thread” o bien “concurrent.Thread”.
|	Este operador permite definir una opción entre dos elementos. El primero elemento precede al operador y el segundo es el siguiente. Por ejemplo la expresión (static|dinamic)_cast permite las cadenas “static_cast” o “dinamic_cast”.

Fuente: http://en.wikipedia.org/wiki/Regular_expression

Dado que las ER son muy flexibles permiten la escritura de consultas bastante diversas y específicas a la vez. Uno de los problemas que surge con esto, es la configuración de la localidad del navegador. Depende de la localidad del navegador se encuentran organizadas el conjunto de letras que forman el alfabeto, por ejemplo abc…zABC…z pero también puede ser AaBbCc…Zz. De forma que el estándar POSIX ha definido algunas clases y categorías de caracteres como se muestra en la siguiente tabla.

Tabla IV. Expresiones regulares POSIX y su equivalente en ASCII

POSIX	***ASCII***	***Descripción***
[:alnum:]	[A-Za-z0-9]	Caracteres alfanuméricos
[:word:]	[A-Za-z0-9_]	Caracteres alfanuméricos y “_”
	[^\w]	Ningún carácter letra
[:alpha:]	[A-Za-z]	Caracteres del alfabeto
[:blank:]	[\t]	Espacio y tabulador
[:cntrl:]	[\x00-\x1F\x7F]	Caracteres de control
[:digit:]	[0-9]	Dígitos
	[^\d]	Ningún dígito

[:graph:]	[\x21-\x7E]	Caracteres visibles
[:lower:]	[a-z]	Letras minúsculas
[:print:]	[\x20-\x7E]	Caracteres visibles y espacios
[:punct:]	[-!"#$%&'()*+,./:;<=>?@[\\\]_`{|}~]	Caracteres de puntuación
[:space:]	[\t\r\n\v\f]	Caracteres de espacio en blanco
	[^\s]	Ningún carácter de espacio en blanco
[:upper:]	[A-Z]	Letras mayúsculas
[:xdigit:]	[A-Fa-f0-9]	Dígitos hexadecimales

Fuente: http://en.wikipedia.org/wiki/Regular_expression

Es importante notar que las expresiones POSIX tal y como lo indica la tabla anterior, son validas únicamente cuando se encierran entre corchetes. De forma que se pude tomar ventaja para la definición de estos elementos para la definición de consultas para las búsqueda de código de Google.

3.5. Bibliotecas cliente de los API de datos de Google

Existen servicios de los API de datos de Google que pueden utilizarse mediante el protocolo HTTP. Pero también Google ha brindado una serie de

bibliotecas clientes en varios lenguajes de programación que ayudan el desarrollo y la integración de nuevos sistemas con estas API.

Para cada uno de los lenguajes de programación se proporciona un marco de trabajo o kit de desarrollo. Esta capa de abstracción permite al desarrollador trabajar sin necesidad de realizar peticiones HTTP.

Los lenguajes de programación que tienen bibliotecas compatibles con los API de datos de Google son:

- Java
- JavaScript
- .NET
- Phyton
- Objective-C

Estas cinco bibliotecas fueron escritas por Google. Pero Zend, un socio de Google ha escrito las bibliotecas en lenguaje PHP, de forma que se encuentran disponibles en este lenguaje también.

4. LIBRERÍA CLIENTE DE JAVA

4.1. Aspectos técnicos

La utilización de las librerías cliente de Java para enviar datos al API de Google se encuentran disponibles para descargar en la dirección http://code.google.com/p/gdata-java-client/downloads/list.

Aquí se podrá encontrar la lista de librerías que Google tiene disponible para los diferentes servicios. Incluso se podrá encontrar versiones de estas API con las fuentes. Pero para la utilización de los API de datos de búsqueda de código únicamente es necesaria una versión estable del archivo 'gdata-java-x.y.z.zip'. La que se utiliza para la presente investigación es la versión 'gdata-java-1.13.0.zip'. Las librerías que contenidas en esta versión son las siguientes.

Tabla V. Conjunto de librerías cliente java

gdata-appsforyourdomain-1.0.jar
gdata-base-1.0.jar
gdata-calendar-1.0.jar
gdata-client-1.0.jar
gdata-codesearch-1.0.jar
gdata-docs-1.0.jar
gdata-photos-1.0.jar
gdata-spreadsheet-1.0.jar

Para utilizar estas librerías se debe de disponer de al menos del JDK versión 1.5. Las librerías necesarias para la utilización de los API de datos de búsqueda de código únicamente son las siguientes tres.

1. gdata-base-1.0.jar
2. gdata-client-1.0.jar
3. gdata-codesearch-1.0.jar

4.2. Modelo de datos

La biblioteca de Java utiliza un conjunto de clases para representar los elementos utilizados por los API de datos. En la sección 3.3 vimos algunos de los elementos del formato GData. De tal forma que existe una clase que correspondiente a cada elemento visto anteriormente y otros más.

Esta biblioteca también puede analizar automáticamente contenido de elementos del formato Atom. Y permite manipular los atributos de cada uno de estos elementos mediante sus métodos y atributos correspondientes.

4.3. Clases para crear una conexión al servicio GData

Para realizar una conexión al servicio de búsqueda de datos de código fuente es necesario utilizar la clase *CodeSearchService*. Esta es una sub-clase de *GoogleService* como se muestra la jerarquía de herencia de clases abajo.

Figura 4. Jerarquía de la clase CodeSearchService

```
java.lang.Object
  └com.google.gdata.client.Service
      └com.google.gdata.client.GoogleService
          └com.google.gdata.client.codesearch.CodeSearchService
```

Para crear una instancia y establecer una conexión al servicio de búsqueda de código se debe de utilizar el constructor el cual requiere únicamente un nombre para el servicio. Este parámetro no tiene nada que ver con algún identificador o nombre del lado del servidor, es únicamente un identificador que el usuario del API asigna, al igual que lo hace con el nombre de una variable.

Figura 5. Constructor de instancia de clase

```
CodeSearchService(java.lang.String applicationName)
```

SearchService, es necesario asignarle una instancia de un localizador de recursos uniforme (URL). Para esto es necesario crea una instancia de la clase URL que se encuentra en el paquete java.net.

Figura 6. Constructor de instancia de clase URL

```
java.net.URL(String java.lang.String spec)
```

Al constructor de esta clase se le asigna como parámetro el feed de datos de búsqueda de código fuente del servicio de Google. Se debe de tener en cuenta que el feed requiere los parámetros q, start-index y max-results, los mismos de la sección 3.2.

Luego de tener estas dos instancias de clases, se puede proceder a realizar un servicio de búsqueda con el método getFeed de la clase *CodeSearchService*. Este método recibe dos parámetros. El primero de ellos es la instancia del URL creada anteriormente. Y el segundo es un *TypeClass*, el cual indica el tipo de feed a localizar. Para el caso de búsqueda de código se utiliza la clase *CodeSearchFeed.class*. Este retorna un objeto

CodeSearchFeed, el cual contiene todos los feeds y sus elementos y atributos que se obtuvieron de la búsqueda. Abajo se muestra como se podría realiza una conexión simple al servicio de búsqueda.

Figura 7. Ejemplo de conexión al servicio de búsqueda

```
CodesearchService codesearchService = new CodeSearchService("gcs-test");
URL privateFeedUrl = new URL(http://www.google.com/codesearch/feeds/search?q=hibernate);
CodeSearchFeed myFeed = codesearchService.getFeed(privateFeedUrl, CodeSearchFeed.class);
```

Fuente: trabajo de graduación

5. SISTEMA DE CONTROL DE VERSIONES

5.1. Aspectos generales

En ingeniería de software y en proyectos lo suficientemente grande eventualmente se puede llegar a un punto en el que la comunicación entre el los participantes es vital para éxito o fracaso de todo un proyecto. Asegurarse de que todos tienen la versión más reciente del último bit de información es crucial para evitar la duplicación de esfuerzos y garantizar que una persona inadvertidamente sobrescriba el trabajo de otra.

Regularmente el alcance de un proyecto crece y el número de personas implicadas con él. Lo que una vez parecía simple se convierte en más y más complejo, y como resultado se requiere más tiempo y esfuerzo. En el mundo de desarrollo de software, una de las maneras de gestionar esta complejidad es a través de la utilización de sistemas de control de versiones.

Estos sistemas proporcionan una forma de gestionar la complejidad de trabajar con un equipo de personas sobre un proyecto. Permite almacenar todos los resultados del proyecto en curso, como lo sería código fuente y la documentación del sistema. Y todo centralizado en un repositorio en el que los distintos miembros del equipo siempre tienen acceso a las últimas versiones de los archivos.

Actualizar cambios locales para que otros usuarios los puedan acceder. Automatización en el proceso de actualización de la copia local de trabajo del

proyecto para incorporar los últimos cambios de otros usuarios. Y quizás lo más importante, los datos históricos sobre el progreso del proyecto, el seguimiento de todos y cada uno de los cambios que se realizaron en el proceso de su creación. Son aspectos que a primera vista podría parecer menos importante, pero con la evolución en el ciclo de vida de un proyecto se puede requerir volver atrás en cierto punto del proyecto por muchas razones. Ya sea para detectar el origen de un error, para recuperar algo de algunos trabajos que se eliminan por error de la versión actual, para realizar una prueba, o tal vez para extraer un cambio particular a fin de que pueda ser aplicado a otra versión del proyecto.

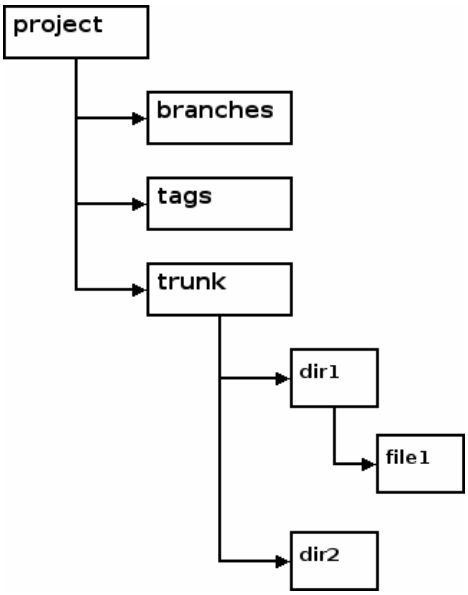
Puede ser considerado como un sistema de control de versiones cualquier sistema que provea las características.

- Gestión por medio de versiones cada cambio que ha sufrido un archivo. Esto cuando el archivo ha sufrido alguna modificación parcial.
- Pero también se pueden eliminar, renombrar o mover elementos del repositorio.
- Capacidad para llevar un histórico de todas las operaciones y cambios a cada elemento.
- Y por supuesto la capacidad de almacenar cualquier clase de archivo y directorios.

5.2. Funcionamiento

Básicamente su funcionamiento consiste en la disposición de un repositorio o conjunto de archivos y directorios que se requieren gestionar. Regularmente la estructura básica de este tipo de repositorio es de la forma que se muestra en la siguiente figura.

Figura 8. Estructura de un repositorio de código



En donde ‘Project’ es el nombre y directorio raíz del repositorio. Se puede notar claramente la existencia de tres subdirectorios principales. El principal de ellos es el denominado “trunk”. En este directorio se almacenan todos los archivos y documentos desarrollados durante toda la vida del proyecto.

Cuando se hace necesario el desarrollo o utilización en paralelo sobre una misma versión de un componente de software. O bien simplemente se realizan prueba unitarias o de sistema, se utiliza la estrategia de branches, o ramas. La cual permite modificar y guardar cambios pero sin afectar la versión que se encuentra en el directorio trunk. Solo afecta a la rama en que se realiza el trabajo. Lo que evita el problema de conflicto entre diferentes versiones.

Y el subdirectorio “tags” que es utilizado para guardar copias del proyecto en un estado específico. Es posible la creación etiquetas sobre etiquetas.

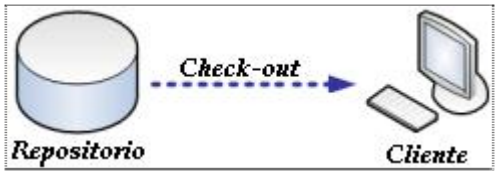
Regularmente es utilizado para llevar el control de versiones estables, como versiones alfa, beta o versión de compilación, etc.

5.3. Operaciones básicas

Todo sistema de control de versiones debe permitir operaciones básicas para la gestión de los archivos en el repositorio. Veamos la primera de ellas.

La operación de descarga de archivos del repositorio. Conocida popularmente como "*check out*". Esta operación solo se realiza una vez, cuando se realiza una copia local del repositorio. O bien cuando existen nuevos archivos en el repositorio que son necesario descargar a la copia local. Cuando se hace un *check-out* existen dos formas en que los sistemas de control de versiones lo pueden manejar, a continuación se presenta la primera.

Figura 9. Descarga y copia local desde un repositorio de datos



5.3.1. Manejo exclusivo

En donde se concede la lectura y modificación del archivo requerido al primer usuario que lo solicite, si y solo si este archivo no está siendo utilizado. El sistema se encarga de colocar un *lock* sobre el archivo para que ningún otro usuario lo pueda modificar. Cuando el primer usuario lo regresa al repositorio, el sistema quita el *lock* al archivo y este se libera para que otros usuarios puedan trabajar sobre él. Este enfoque lo es utilizado en repositorios de documentos como *Microsoft Office Project Server*.

5.3.2. Manejo colaborativo

En donde cada usuario descarga una copia local en la cual trabaja. Y publica al repositorio los cambios que realizó a la copia local. Uno de los problemas de este enfoque es que regularmente se dan muchos conflictos en el momento de sincronizar los archivos locales al repositorio y se deben de solucionar de forma manual. Herramientas que utilizan este enfoque son CVS y Subversión que es la herramienta que veremos en detalle más adelante.

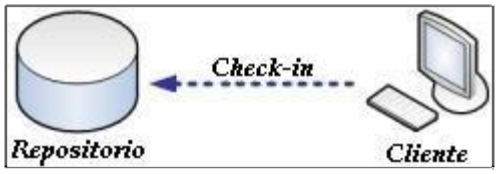
La operación de actualización o update. Como su nombre lo indica actualiza y modifica todos los archivos locales con los cambios que se encuentren con el repositorio. Es decir sincroniza la copia local con el repositorio. Esta actualización se realiza únicamente si no se detectan cambios y conflictos en los archivos locales. Si existen conflictos, los cambios deberán de actualizarse de forma manual.

Figura 10. Actualización de datos y dependencias desde el repositorio principal

La actualización del repositorio o bien "*check-in*". Consiste actualizar los archivos locales con el repositorio, o sincronización. Para esto el sistema de control de versiones antes de actualizar, verifica que la versión a sincronizar sea coincidente con la que se encuentra en el repositorio. Esto es debido a que otro usuario pudo actualizar antes el mismo archivo y la copia local ya no se encuentra sincronizada. Cuando se presenta dicho escenario se debe de

realizar un “*merge*” o integración de las fuentes de código. Esta no es más que la fusión de los cambios del repositorio con los cambios locales y subir la nueva versión al repositorio.

Figura 11. Confirmación de cambios y actualización de datos hacia el repositorio

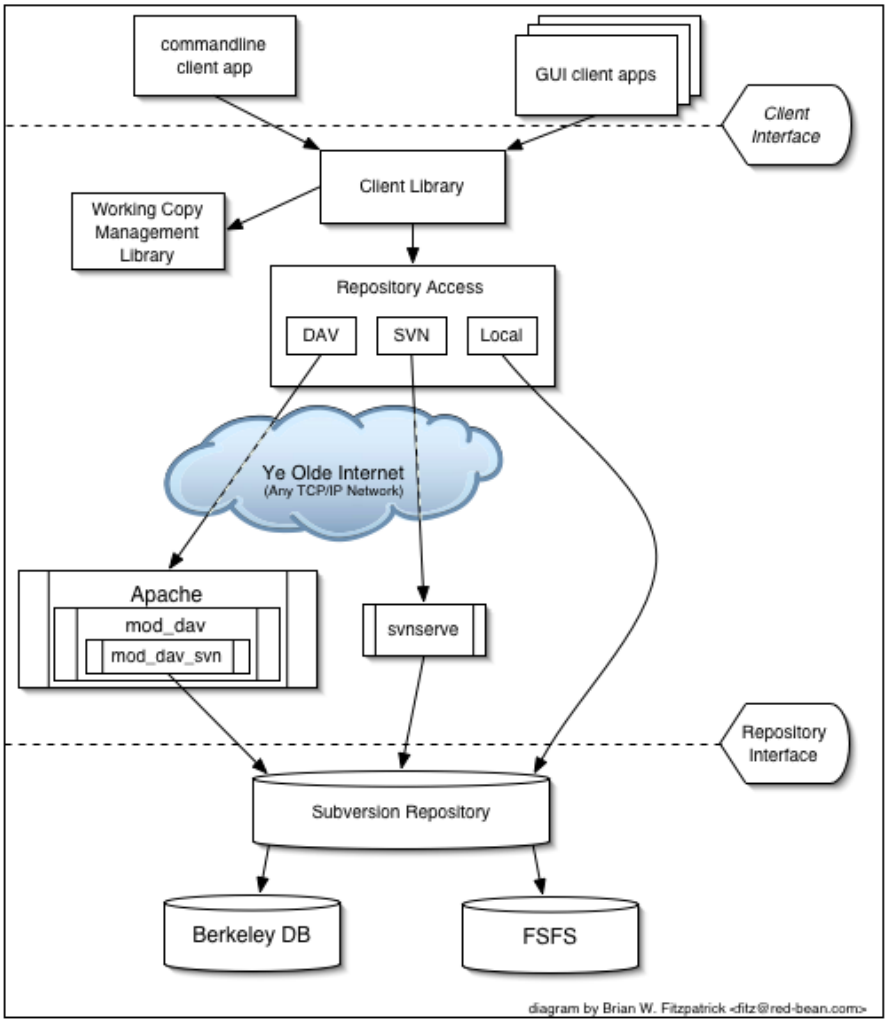


5.4. Subversion

Es un sistema de código abierto de control de versiones que tiene los aspectos y funciones básicas que mencionamos en las dos secciones anteriores. Permite la creación de repositorio de datos y llevar el control de cambios a archivos y directorios. Esto permite recuperar versiones antiguas y observar la evolución de los datos. También permite la administración de usuarios y permisos de lectura y escritura sobre los repositorios.

CollabNet, Inc. (http://www.collab.net) a principios del año 2000 empieza un proyecto para desarrollar un sistema que reemplazaría a CVS. Los objetivos del equipo de diseño eran simplemente, la corrección de errores de CVS. Luego de catorce meses de codificación liberan Subversión para el mes de agosto del 2001. Este al igual que CVS es un sistema que se basa totalmente sobre línea de comandos para la gestión de servicios. Esto no es coincidencia del azar, se hizo de esta forma para evitar un alto costo de cambio y el con el objetivo que los usuarios de CVS no tuvieran dificultades para cambiarse a Subversión. En el diagrama siguiente, se muestra la arquitectura de este sistema.

Figura 12. Arquitectura y distribución de capas de Subversion



Referencia: svn_book, página No.22.

Los servicios en la capa de 'línea de comandos' se detallan a continuación describiendo los parámetros más importantes para cada uno. La mayoría de estos comando son utilizados por herramientas clientes que permiten administrar los archivos en los repositorios tal es el caso de TortoiseSVN o WebDav.

5.4.1. svn

Provee los principales servicios para la administración de los archivos y directorios de un repositorio de datos.

Tabla VI. Servicios para svn

Parámetro	***Descripción***
add ARG	Permite agregar archivos y directorios bajo el sistema de control de versiones.
blame	Especifica el contenido de un archivo con información de la revisión y del autor.
checkout	Especificando el nombre y contraseña permite realizar una copia local de todos los archivos y directorios del repositorio.
commit	Permite enviar cambios de la copia local al repositorio.
diff	Permite visualizar las diferencias que existen entre dos revisiones de archivos.
import	Importa un archivo o directorio al repositorio para ser versionado.
merge	Permite fusionar las diferencias entre dos versiones de un mismo archivo.
mkdir	Crea un directorio sobre el repositorio de control de

	versiones.
revert	Permite restaurar una copia de trabajo deshaciendo los cambios locales editados.
unlock	Permite desbloquear copias de trabajo especificando la dirección de un repositorio.
Update	Permite actualizar desde el repositorio, la copia actual de trabajo

5.4.2. svnadmin

Servicio que provee la funcionalidad básica para la administración de repositorios bajo el control de versiones. Dado los parámetros que se listan a continuación.

Tabla VII. Servicios para svnadmin

Parámetro	***Descripción***
create	Permite la creación de un nuevo repositorio de datos.
gump	Permite volcar el contenido del sistema de archivos de un repositorio a un solo archivo portable ‘dumpfile.dpf’.
load	Permite la lectura y carga de un archivo ‘dumpfile.dpf’, actualizando las revisiones en un repositorio del sistema de control de versiones.

hotcopy	Permite realizar una copia en caliente de un repositorio específico.
recovery	Permite correr el procedimiento de recuperación de un repositorio. Para esto se debe de bajar el servicio de Subversión pues la base de datos Berkeley requiere acceso exclusivo a los datos del repositorio.
setlo	Permite configurar las bitácoras de mensajes de una revisión o versión al contenido de un archivo.
setuuid	Permite restablecer el identificador de un repositorio. Si se especifica un nombre, este será asignado de lo contrario se generará uno nuevo. Este es un identificador único para cada repositorio.
verify	Permite la verificación de los datos de un repositorio. Se puede la verificación de una versión o revisión especifica. Y también la visualización del detalle de la verificación o bien únicamente la especificación de errores a un archivo.

5.4.3. svnserve

Es un pequeño y ligero servicio basado en línea de comando. Este servicio cliente-servidor provee operaciones con alto rendimiento a través de una red y se puede integrar con SSH para establecer conexiones de datos encriptados.

Tabla VIII. Servicios para svnserve

Parámetro	***Descripción***
D	Permite correr Subversión como un demonio o daemon (*Disk And Execution Monitor*). Cuando corre en modo demonio acepta conexiones y servicios TCP/IP a través del puerto 3690 por defecto, pero puede ser configurado en otro puerto.
version	Permite visualizar la versión de los componentes de Subversión, tal como los módulos de trabajo y autenticación de comunicaciones seguras.
inet mode	Permite correr el servicio en modo seguro. Cuando el servicio realiza una petición a Subversión, éste se comunica vía inetd. Si se trabaja sobre Unix con el demonio inted, este se puede configurar en el archivo inted.conf.
service	Permite correr el servidor como un servicio. En caso de que se ejecute sobre plataformas de Windows.

5.4.4. svnsync

Es una herramienta que permite crear copias espejo de repositorios. Esta permite recrear las revisiones de un repositorio en otro. Esto se logra teniendo dos repositorios sincronizados. Existe una fuente y un espejo en donde todo cambio realizado a la fuente es replicado al espejo. Cabe mencionar que el repositorio espejo puede ser local o bien puede ser remoto.

Tabla IX. Servicios para svnsync

Parámetro	***Descripción***
config-dir DIR	Le indica a Subversión que debe de leer la configuración especificada en vez de la localidad por defecto.
initialize MIRROR_URL SOURCE_URL	Realiza una verificación del repositorio espejo y si no existe ninguna información se genera información para administrar el espejo a partir de la fuente indicada.
synchronize DEST_URL	Permite sincronizar y actualizar todas las revisiones a un espejo. Se debe de indicar el destino del espejo a sincronizar.
Service	Permite correr el servidor como un servicio. En caso de que se ejecute sobre plataformas de Windows.

5.5. Registro de repositorio de código sobre la plataforma Google

Google provee gran cantidad de servicios gratuitos, por mencionar algunos; *Gmail*, *GoogleMaps*, *Picasa*, *GoogleDocs*, *YouTube*, etc. Esto debido a que posee una gran infraestructura de centro de datos distribuidos geográficamente alrededor del mundo.

Como se habrá podido observar los servicios de búsqueda han evolucionado. Estos se han vuelto cada vez más específicos y se enfocan en

cierto contenido. Búsqueda de imágenes, de documentos, de libros, grupos de personas, etc. Existe un enfoque asociada a cada tipo de información en publicada en Internet.

Y debido a que Google ha liberado una gran cantidad de API y código fuente, los servicios de búsqueda para código fuente no podían ser una excepción. Para buscar código fuente en repositorios de Internet se puede utilizar el URL http://www.google.com/codesearch. Este servicio al igual que todos los servicios de búsqueda es gratuito, no tiene ningún costo. De forma que localizar código fuente para los desarrolladores nunca pudo haber sido más fácil.

Y no se podía esperar menos de Google. Provee un servicio de registro de repositorios de código. Es decir, permite indexar repositorios de código publicados en Internet para que sean tomados en cuenta en las búsquedas y de esta manera localizar código fuente contenido en dichos repositorios. El URL para registrar repositorios de código es:

http://www.google.com/codesearch/addcode?hl=es

Se pueden registrar archivos simples o bien paquetes completos que contiene un conjunto de fuentes de código. También es posible registrar repositorios de código CVS. Para este último caso se deberá de indicar el nombre del repositorio público y el módulo.

Y para el caso de registrar un repositorio de tipo Subversión, únicamente se debe de indicar el URL del repositorio.

6. PLATAFORMA ECLIPSE

6.1. ¿Qué es Eclipse?

Eclipse es entorno de desarrollo integrado, el cual tiene una arquitectura de plug-in concebida para facilitar agregar herramientas según se necesiten, dependiendo de las tareas que se deban de realizar. El conjunto de librerías que conforman su núcleo, están escritas en código nativo de Java. Aunque no de la misma forma el sub-sistema de ventanas que proveen la interfaz de usuario. Es por ello que existen varias versiones de Eclipse, dependiente de la plataforma sobre la que corre. Es decir, no es lo mismo Eclipse para Windows, que Eclipse para Macintosh. Esto debido que para Windows se deberá cargar el sub-sistema de ventas Win32 y para MacOS se deberá cargar Carbon.

Aunque Java, es ampliamente utilizado en muchas aplicaciones empresariales y en sistemas que requieren tener un alto rendimiento computacional y transaccional no era muy utilizado para el desarrollo de interfaces hasta que surgió Swing.

Swing es framework de código nativo Java que permite el desarrollo de interfaces de usuario cliente-servidor. Pero Swing, tiene el inconveniente que puede variar su aspecto dependiendo de la plataforma sobre la que se encuentre corriendo. Y Eclipse en un inicio fue desarrollado con esta tecnología. De esta forma IBM se intereso en el proyecto Eclipse, pero notó este inconveniente y lo abordo desarrollando un nuevo framework al que llamo SWT.

Con esta biblioteca se pudo mejorar el aspecto de Eclipse, de tal forma que fuera independiente de plataforma. Es por ello que se podrá notar que Eclipse tiene un aspecto igual independientemente de plataforma. Dado que Eclipse Workbench utiliza los API de SWT para el sub-sistema de ventanas e interfaces de usuario.

6.2. Arquitectura de SWT

El conjunto de API que proporciona AWT y otro conjunto de librerías escritas en lenguaje C, sirven para proporcionar independencia de plataforma. No existe ninguna llamada directa por parte de este API al sub-sistema de ventanas de la plataforma que sirve como anfitrión. Lo que claramente proporciona un nivel de independencia de la plataforma.

Pero como ya se ha mencionado anteriormente, AWT tenía inconvenientes en cuanto a su aspecto según la plataforma en la que corría. Otra desventaja de esta librería es que no era muy flexible en cuantos algunos componentes de interfaz de usuario. Fue por ello que se desarrollo sobre Swing. Que provee más flexibilidad y mejores componentes. Por otra parte el rendimiento y aspecto de Swing no convenció a IBM. A causa de esto desarrolladores e investigadores empezaron a escribir una biblioteca gráfica de usuario llamada SWT, del inglés Standard Widget Toolkit. O bien traducido al español, Kit estándar de componentes.

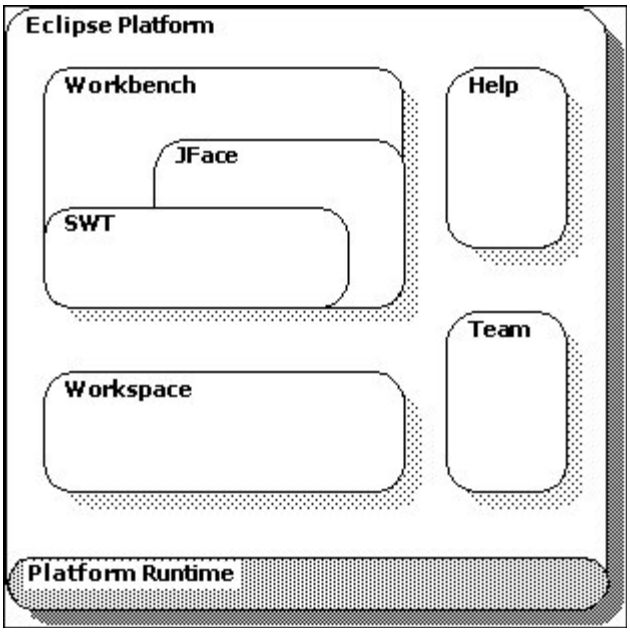
Las ventajas de utilizar la biblioteca SWT son:

- El aspecto no difiere de la plataforma sobre la que corre.
- API más simple, más rápido.

Pero no todo es alegría con SWT, también tiene desventajas:

- Requiere bibliotecas SWT nativas de plataforma destino.
- No es tan flexible como Swing en la codificación.
- Las API de AWT como Java2D, Java3D y JAI no puede ser utilizadas por SWT.

Figura 13. Estructura de la plataforma SWT



Fuente: Plataform Plug-in Developer Guide, estructura de plataforma, página No.3.

Se presenta a continuación de forma de breve y resumida que son cada una de las partes que conforma la estructura de la imagen anterior.

Platform Runtime, define el punto de extensión del modelo del plug-in. Descubre de forma dinámica los plug-ins y mantiene información acerca de ellos y sus puntos de extensión. Cabe mencionar que un punto de extensión es aquel componente que se encuentra en el workbench. Tal puede ser un menú, una barra de herramientas, un editor de texto o bien una ventana de depuración.

Administración de recursos (*Workspace*), define el conjunto de API para administrar recursos. Es decir, crear, modificar guardar archivos, folders, proyectos y otras operaciones a nivel del sistema de archivos.

Workbench UI, implementa todos los componentes que permite navegar al usuario en la plataforma. Define puntos de extensión mediante la agregación de componentes de interfaz de usuario como menús, acciones, diálogos, eventos, etc.

Help system, el plug-in sistema de ayuda implementa un sistema web interno para desplegar documentación en formato html y de fácil integración. Este define un punto de extensión a partir del cual puede ser utilizado por otro plug-in como un libro en digital. La documentación del servidor web incluye especiales características que permitan facilitar referencias a archivos utilizando enlaces web o bien direcciones del sistema de archivos.

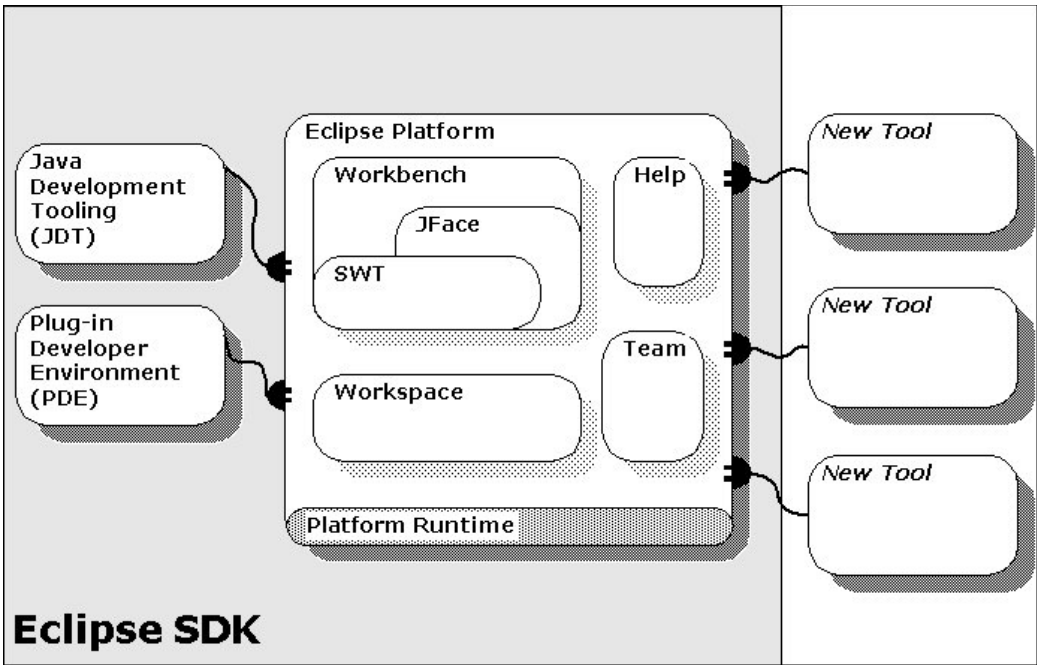
6.3. Arquitectura de Eclipse

La plataforma Eclipse tiene un micro-kernel interno. A excepción de este micro núcleo, todos los demás componentes están escritos como plug-ins. Y cada módulo es cargado por este pequeño núcleo.

Eclipse tiene una arquitectura de plug-in. De esta forma los desarrolladores pueden reutilizar funcionalidad y crear extensiones (plug-ins) que contribuirán en alguna funcionalidad de todo el sistema. La interfaz de usuario de Eclipse puede ser visualizada en tres grupos principales, Workbench (plataforma), JFace y SWT.

Desde el punto de vista del usuario, el workbench es la ventana principal de Eclipse. La que está compuesta por las barras de menús, barra de herramientas, navegadores y editores. El objetivo de esta es facilitar la integración de las herramientas. Por otro lado JFace se encuentra diseñado para trabajar con SWT. Y SWT define el conjunto de componentes que forman la base para la construcción de la interfaz de usuario de Eclipse.

Figura 14. Arquitectura de plug-in



Fuente: Plataform Plug-in Developer Guide, arquitectura de plataforma, página No.5

Algunos de estos elementos fueron mencionados anteriormente. Aquí se podrá observar como la arquitectura de plug-in entra en acción. El JDT es un plug-in que extiende la plataforma, proveyendo características especificas para editar, visualizar, depurar y compilar código fuente. Este es un plug-in que es instalado como un elemento del SDK de Eclipse.

Otro plug-in importante es el PDE, el cual ofrece un conjunto de herramientas que automatizan la creación, manipulación, depuración y despliegue de plug-ins. Este también es un plug-in instalado como un elemento del SDK. Un componente que sin duda es parte fundamente de Eclipse y su plataforma.

Permite desde la creación de proyectos de plug-ins, administrar propiedades importantes del plug-in como su versión, identificador, nombre, librerías. Generando las clases que controlan el plug-in, así como la administración de los archivos plugin.xml, creación y gestión de puntos de extensión para crear perspectivas, vistas, acciones editores y otros componentes visuales dentro del workbench.

Existen varias formas de hacer pruebas al plug-in desarrollado. Se puede utilizar marcos de trabajo dedicados a pruebas como JUnit. Pero este componente de igual forma permite correr el plug-in como parte de la plataforma, para esto se creará una nueva instancia de Eclipse en donde el plug-in se exporta y se instala en el workbench. Este es un aspecto importante de saberlo para un desarrollador pues facilita las pruebas y depuraciones a un componente

6.4. Conceptos avanzados del Workbench

Será necesario tener el conocimiento de conceptos más avanzados para el desarrollo de un plug-in. A pesar de que existen muchos puntos de extensión disponles para la contribución de la plataforma, con el fin de tener los conocimientos básicos para el desarrollo de un plug-in se deberá estar familiarizado con los siguientes tópicos.

- Estar habituando al Workbench
- Conocimiento básico del *farmework* JFace
- Conocimiento básico del framework SWT

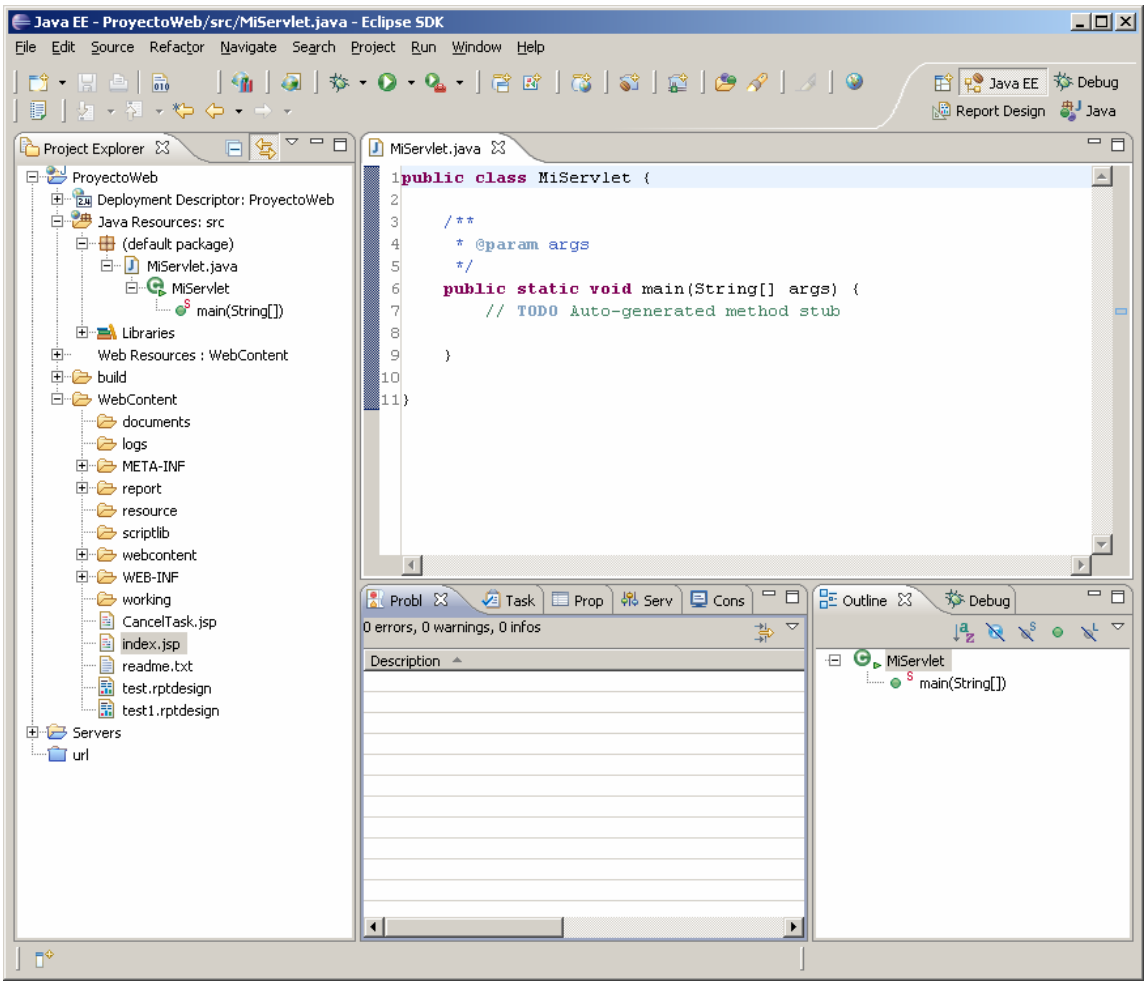
Cuando al menos se utiliza alguna de estas características en un plug-in, es buena idea tener en mente las buenas prácticas que se deben de seguir para la programación concurrente.

6.4.1. Habituándose con el Workbench

Se deberá estar familiarizado con la forma de operación de la plataforma. Como se utilizan las vistas, los editores, ventanas de información. Si no es así, no es problema siempre hay una primera oportunidad. A continuación se dará un vistazo la interfaz de usuario de la plataforma, se mostrará como un plug-in puede contribuir a la interfaz de la misma.

La plataforma se puede ver como una cabina de mando la cual sirve para navegar sobre toda la funcionalidad que proveen los plug-ins. Se puede navegar en un árbol de archivos de un proyecto, editar archivos y propiedades. Cuando un proyecto se encuentra abierto se verá algo similar como la imagen que se presenta a continuación.

Figura 15. Eclipse en acción



Esta vista es simplemente un ventana la cual presenta varias partes visuales. Estas partes encajan dentro de dos categorías: vistas y editores.

Las vistas proveen información sobre algún objeto en el cual el usuario se encuentra trabajando. Estas cambian constantemente según el contenido seleccionado por el usuario. Las vistas regularmente apoyan a los editores proveyendo información sobre el contenido del editor activo.

Los editores permiten al usuario editar todo tipo de archivos. Al igual que un común editor de texto, permite el ciclo de crear-abrir-guardar-cerrar. A diferencia de otros editores, este se encuentra totalmente integrado en la plataforma.

6.4.2. El framework JFace

Como se ha podido observar la plataforma define los puntos de extensión para los plug-ins que contribuirán con las funciones de interfaz de usuario. Muchos puntos de extensión, particularmente los asistentes de extensiones son implementados usando clases del paquete org.eclipse.jface.*.

JFace es un conjunto de herramientas de interfaz de usuario que provee clases ayudantes para el desarrollo de de características de interfaz de usuario que pueden ser tediosas de implementar. Este opera en un nivel superior de los componentes del sistema. Provee clases para el manejo y administración de tareas y eventos de programación. Su principal objetivo es despreocupar al programador para que este se enfoque en la implementación específica del plug-in, en vez de centrarse en solucionar problemas que son comunes en cualquier aplicación que involucre una interfaz de usuario, eventos, acciones, validaciones, etc.

Las líneas de código de JFace se mezclan con las de SWT, pero sin embargo estas son fácilmente distinguibles y claras. SWT no depende de ningún componente de JFace y este último está diseñado para proveer funcionalidad común en un nivel superior de la librería SWT. La plataforma utiliza JFace con el objetivo de reducir las dependencias en lo posible, y de mantenerse de forma neutral permitiendo a los programadores utilizar partes de JFace que se encuentren útiles.

6.4.3. El framework SWT

Del inglés *Standard Widget Toolkit*, que traducido al español sería el Kit de Herramientas Estándar de Componentes. Para desarrolladores en lenguaje Java, este les provee un API portable que tiene una estrecha integración con el sistema operativo nativo y la interfaz gráfica de usuario.

Muchas tareas de bajo nivel que se encuentran en la interfaz de usuario se manejan en las capas superiores de la plataforma de Eclipse. Por ejemplo, editores y acciones proveen implementaciones para interacciones comunes entre la aplicación y los componentes de la interfaz de usuario. Sin embargo el conocimiento técnico de SWT es importante para entender el resto del funcionamiento de la plataforma.

Este marco de trabajo define API común portable en todas las plataformas compatibles e implementa un API para cada plataforma utilizando componentes nativos en la mayoría de los casos y siempre que sea posible. Esto permite que los componentes de la interfaz gráfica de usuario adopten una apariencia igual a la plataforma y por ende consistente en el modelo de programación de dichos componentes. Es importante mencionar que dentro de los componentes SWT existe uno llamado "*Browser*" o Explorador en español. Este componente es un explorador que se encuentran disponible en las plataformas:

- Windows (Internet Explorer 5 o superior)
- Mac (OS X 10.4 o superior)
- Linux (Mozilla Firefox 1.0 o superior)

Esta información será útil posteriormente debido a que el plug-in que a continuación se desarrollará utiliza este componente para mostrar resultados.

7. DESARROLLANDO UN PLUG-IN

7.1. Desarrollo de un plug-in con dependencias externas

Para el desarrollo de un plug-in que utilice los API de datos de búsqueda que provee Google es necesario utilizar el conjunto de librerías base, client y codesearch que se mencionan en la sección 4.1.

Es importante mencionar que estas librerías realizan conexiones a los servicios de búsqueda de Google y por ende tiene dependencias de otras librerías necesarias para establecer una conexión. Estas librerías son las siguientes:

- activation.jar
- mail.jar
- servlet-api-2.4.jar

Si no se incluyen estas últimas tres librerías no se podrá establecer una conexión con el servicio de búsqueda código de Google, es por ello que son indispensables. Ahora bien, para desarrollar un plug-in con dependencias externas tal y como lo requiere el plug-in de búsqueda de código, es necesario incluir las librerías que vimos en la sección 4.1 y estas últimas tres dependencias. Pero el desarrollo de un proyecto de plug-in es diferente al desarrollo de una aplicación Java convencional o Web. En una aplicación Java

convencional al requerir librerías externas simplemente se agregan al proyecto mediante la correcta configuración del "*Java Build Path*" del proyecto.

El desarrollo de un proyect de tipo plug-in, que requiere librerías externas es un caso diferente. Se debe como primer paso desarrollar un nuevo proyecto por aparte del tipo plug-in desde archivos JAR existentes o bien *Plug-in from existing JAR files*. Este tipo de proyecto solo permite agregar librerías con extensión *.jar* en el proyecto y generar un plugin el cual contiene información sobre dichas librerías.

Para este caso entonces se crea un proyecto de plug-in desde archivos JAR existentes en donde se incluyen las librerías:

- gdata-base-1.0.jar
- gdata-client-1.0.jar
- gdata-codesearch-1.0.jar
- activaction.jar
- mail.jar
- servlet-api-2.4.jar

Una vez creado el plug-in con estas librerías, se procede a exportar como un Plug-in instalable o bien *Deployable plug-in and fragments*. Al realizar esto se generará un archivo con extensión JAR el cual es un plug-in instalable sobre cualquiera de las versiones de Eclipse. Solo basta con copiar el archivo al directorio de plugi-ns de Eclipse.

Figura 16. Directorio de instalación de plugi-ns

```
Directorio_instalacion_eclipse\eclipse\pulgins\
```

Fuente: elaboración propia

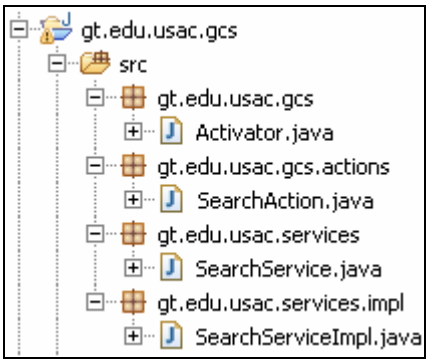
Una vez realizado esto, el plug-in de dependencias externas se encuentra listo para ser utilizado por otros proyectos tal y como veremos a continuación.

7.2. Integración de un plug-in con SWT

Para el desarrollo de un plug-in que provea el servicio de búsqueda de código fuente integrado en el ambiente de desarrollo de Eclipse, es necesaria la utilización del framework SWT. Este marco de trabajo provee un conjunto de componentes para construir interfaces gráficas en lenguaje de programación Java como se ha mencionado anteriormente.

El primer paso para elaborar el plug-in es crear un nuevo proyecto en Eclipse y seleccionar la categoría de "*Plug-in Development*". Luego seleccionar el tipo de proyecto "*Plug-in Proyect*". Para este proyecto se debe de seleccionar una estructura de directorios en donde el directorio "src" contendrá todas las fuentes, paquetes y recursos del proyecto. Y el directorio de salida deberá ser "bin". El segundo paso será crear la estructura de paquetes en el directorio "src" necesarios para organizar las clases que contendrán los servicios e interfaz gráfica del proyecto. Siguiendo el estándar de nombramiento de dominio inverso para nombrar los paquetes se nombrarán cuatro paquetes principales. Las siglas 'gcs' son para la abreviación de *Google Code Search*.

Figura 17. Estructura de paquetes del plugin de búsqueda de código



Fuente: trabajo de graduación

El paquete gt.edu.usac.gcs contiene la clase *Activator.java*. Esta es una sub-clase de *AbstractUIPlugin*. Esta última es parte de la infraestructura de Eclipse la cual que permite la administración del ciclo de vida del plug-in mediante los métodos *start()* y *stop()*. Esta se encuentra en el paquete org.eclipse.ui.plugin. Cualquier acción adicional como inicialización o finalización de valores de variables puede realizarse en estos métodos.

Esta clase será automáticamente cargada, pues al extender de *AbstractUIPlugin* el Run-Time de Eclipse logra detectar que este es una Interfaz de usuario de Eclipse. El método llamado por el *Run-Time* es el getDefault(), el cual devuelve la variable estática plug-in. Este es un singleton dado que esta compartido apara todas las instancias de esta clase. En la siguiente figura se detalla las líneas de código de cada clase.

Figura 18. Código de la clase Activador del plug-in

```
package gt.edu.usac.gcs;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.ui.plugin.AbstractUIPlugin;
import org.osgi.framework.BundleContext;
/**
 * Activador de clase que controla el ciclo de vida del plugin
```

```
 */
public class Activator extends AbstractUIPlugin {



// ID del plug-in
public static final String PLUGIN_ID = "gt.edu.usac.gcs";

// Instancia compartida (Singleton)
private static Activator plugin;

/**
 * constructor
 */
public Activator() {
plugin = this;
}

/*
 * (non-Javadoc)
 * @see org.eclipse.ui.plugin.AbstractUIPlugin#start(org.osgi.framework.BundleContext)
 */
public void start(BundleContext context) throws Exception {
super.start(context);
}

/*
 * (non-Javadoc)
 * @see org.eclipse.ui.plugin.AbstractUIPlugin#stop(org.osgi.framework.BundleContext)
 */
public void stop(BundleContext context) throws Exception {
plugin = null;
super.stop(context);
}
/**
 * @return  Devuelve la instancia compartida
 */
public static Activator getDefault() {
```

```
return plugin;
}
/**
 * Devuelve la imagen del descriptor para la imagen del archivo dado
 * @param path
 * @return el descriptor de imagen
 */
public static ImageDescriptor getImageDescriptor(String path) {
return imageDescriptorFromPlugin(PLUGIN_ID, path);
}
}
```

Fuente: trabajo de graduación

El paquete gt.edu.usac.gcs.actions contiene la clase *SearchAction.java*. Esta es la clase que contiene los componentes visuales del plug-in. Debido a que la interfaz es una ventana de diálogo, esta necesita implementar la clase *IWorkbenchWindowActionDelegate*, la cual se encuentra en al paquete de infraestructura org.eclipse.ui.

Esta requiere que en su método de inicialización un parámetro especial. En la figura siguiente se muestra la firma de dicho método.

Figura 19. Método de inicialización de la clase SearchAction

```
init(IWorkbenchWindow window)
```

Fuente: trabajo de graduación

Este parámetro de tipo *IWorkbenchWindow* el cual es la venta principal del entorno de Eclipse. Este parámetro dado cuando la clase es cargada. La clase *SearchAction,* es la interfaz de usuario. Esta fue diseñada para leer una consulta y desplegar un conjunto de resultados de enlaces a páginas y repositorios de código. Es por ello que se utilizó un objeto *Browser* para mostrar

los resultados de búsquedas. Este último objeto y otros como cajas de texto, botontes, etc., se encuentran en el paquete org.eclipse.swt.

Básicamente el diseño seguido para su elaboración consiste en que dada una entrada de consulta o búsqueda mediante el servicio de búsqueda se elabora un conjunto enlaces mediante un buffer string en formato HTML, los cuales son generados a partir de los feeds obtenidos. Posteriormente generado el buffer, este es dado al *Browser* para que este los muestre y puedan ser accedidos mediante su selección.

Figura 20. Código de la clase de interfaz de usuario SearchAction

```
package gt.edu.usac.gcs.actions;

import gt.edu.usac.services.SearchService;
import gt.edu.usac.services.impl.SearchServiceImpl;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.swt.SWT;
import org.eclipse.swt.browser.Browser;
import org.eclipse.swt.events.KeyAdapter;
import org.eclipse.swt.events.KeyEvent;

import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.FormAttachment;
import org.eclipse.swt.layout.FormData;
import org.eclipse.swt.layout.FormLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
```

```
import com.google.gdata.data.codesearch.CodeSearchEntry;
import com.google.gdata.data.codesearch.CodeSearchFeed;

/**
 * La accion del proxy sera creada por Eclipse y mostrara
 * la interfaz. Cuando el usuario intente usar el comando
 * este delegara y creara la ejecucion para delegarlo.
 * @see IWorkbenchWindowActionDelegate
 */
public class SearchAction implements IWorkbenchWindowActionDelegate {
private IWorkbenchWindow window;

private  static final int ESC = 27;          //tecla esc
private  static final int ENTER = 13;        //tecla enter
private static final int INTRO = 16777296;

private static final String HTML_HEADER = "<html><head><title>Resultados para
GCS</title></head>";
private static final String HTML_INIT_BODY = "<body>sistemas.ingenieria-
usac.edu.gt<br/><hr/>";
private static final String HTML_END_BODY = "</body></html>";

Browser browser;
SearchService searchService;

/**
 * Constructor
 */
public SearchAction() {
}



/**
 * La accion ha sido activada. El argumento del
 * metodo representa la accion 'real' en la interfaz
 * del entorno de Eclipse.
 * @see IWorkbenchWindowActionDelegate#run
```

```
 */
public void run(IAction action) {

try{
searchService = new SearchServiceImpl();
}catch(Exception e){
System.err.println(e.getMessage());
}

final Shell dialog = new Shell (window.getShell(), SWT.DIALOG_TRIM |
SWT.APPLICATION_MODAL);
dialog.setText("Sistema de busqueda de codigo usando GData");
FormLayout formLayout = new FormLayout ();
formLayout.marginWidth = 10;
formLayout.marginHeight = 10;
formLayout.spacing = 10;
dialog.setLayout (formLayout);

Label label = new Label (dialog, SWT.NONE);
label.setText ("Google Code Search:");
FormData data = new FormData ();
label.setLayoutData (data);

final Text text = new Text (dialog, SWT.BORDER);
data = new FormData ();
data.width = 200;
data.left = new FormAttachment (label, 0, SWT.DEFAULT);
data.top = new FormAttachment (label, 0, SWT.CENTER);
       text.setLayoutData (data);
       text.addKeyListener(new KeyAdapter() {
       public void keyPressed(KeyEvent event) {
       System.out.println(event.keyCode);
       switch (event.keyCode) {
       case ENTER :
       case INTRO:
       Text t = (Text)event.widget;
       searchAction(t.getText());
       break;
```

```
        case ESC:
        dialog.close ();
        break;
        }
        }
        });

Button ok = new Button (dialog, SWT.PUSH);
ok.setText ("Búscar código");
data = new FormData ();
data.top = new FormAttachment(text,0,SWT.CENTER);
data.left = new FormAttachment(text,0,SWT.DEFAULT);
ok.setLayoutData (data);
ok.addSelectionListener (new SelectionAdapter () {
public void widgetSelected (SelectionEvent e) {
searchAction(text.getText());
}
});

browser = new Browser(dialog, SWT.NONE);
data = new FormData();
data.width = 500;
data.height = 500;
data.top = new FormAttachment(ok,0, SWT.DEFAULT);
browser.setLayoutData(data);

dialog.setDefaultButton (ok);
dialog.pack ();
int x = window.getShell().getSize().x/2 - dialog.getSize().x/2;
int y = window.getShell().getSize().y/2 - dialog.getSize().y/2;
dialog.setLocation(x, y);
dialog.open ();
text.setFocus();
}

/**
 * Seleccion cuando sucede un cambio en la
```

```
 * plataforma.
 * @see IWorkbenchWindowActionDelegate#selectionChanged
 */
public void selectionChanged(IAction action, ISelection selection) {
}

/**
 * Meotodo que puede ser uilizado para
 * liberar cualquier tipo de recurso
 * previamente alojado en el sistema.
 * @see IWorkbenchWindowActionDelegate#dispose
 */
public void dispose() {
}

/**
 * Pondra en cache una instancia de la ventana
 * para proveer el mensaje de dialogo.
 * @see IWorkbenchWindowActionDelegate#init
 */
public void init(IWorkbenchWindow window) {
this.window = window;
}

/**
 * metodo principal encargado de realizar
 * la busqueda segun el paremtro de entrada q.
*/
private void searchAction(String q){
if(searchService!=null){
if(q!=null && q.length()>0){
searchService.setQuery(q);
try {
CodeSearchFeed csf = searchService.retrieveFeed();
if(csf!=null){
StringBuffer htmlBuffer = new StringBuffer();
htmlBuffer.append(HTML_HEADER);
```

```
  htmlBuffer.append(HTML_INIT_BODY);


  for(CodeSearchEntry entry: csf.getEntries()){
  htmlBuffer.append("<p>");
  htmlBuffer.append("<a
  href=\""+entry.getHtmlLink().getHref()+"\">"+entry.getTitle().getPlainText()+"</a><br/>
  ");
  htmlBuffer.append(entry.getHtmlLink().getHref());
  htmlBuffer.append("</p>");
  }
  htmlBuffer.append(HTML_END_BODY);
  browser.setText(htmlBuffer.toString());
  }
  }catch (Exception e) {
  e.printStackTrace();
  }
  }
  }
  }
}
```

Fuente: trabajo de graduación

El paquete gt.edu.usac.services contiene la interfaz del servicio de búsqueda. El diseño de este servicio está basado en la filosofía de las arquitecturas orientadas a servicios, SOA. Es por ello que en la clase anterior SearchAction, existe únicamente la definición de la interfaz del servicio la cual recibe una implementación del mismo.

Esto permite tener un desacople bajo en lo que se refiere a la dependencia del servicio de búsqueda de Google. Si en algún momento en el tiempo cambiará el API del servicio. Se puede elaborar otra implementación sin necesidad de afectar ningún otro componente del plug-in, como por ejemplo, la capa de interfaz de usuario. A continuación se detalla esta interfaz de servicio.

Figura 21. Código de la interfaz del servicio de búsqueda

```
package gt.edu.usac.services;
import com.google.gdata.data.codesearch.CodeSearchFeed;
/**
 * @author msamayoa
 */
public interface SearchService {

public CodeSearchFeed retrieveFeed() throws Exception;

public void setQuery(String q);

public void setAdvanceQuery(String q, String startIndex, String maxResults);
}
```

Fuente: trabajo de graduación

Y por último el paquete gt.edu.usac.services.impl el cual contiene la implementación del servicio de búsqueda, con el nombre de *SearchServiceImpl*. Este servicio es el encargado de establece la conexión con el servicio de búsqueda mediante la clase *CodeSearchService* que se detallo en la sección 9.3. Básicamente el servicio obtiene un conjunto de entradas y actualizaciones de diferentes repositorios. El servicio únicamente recibe como parámetro la cadena de consulta. A continuación se muestra la implementación del mismo.

Figura 22. Código de la implementación del servicio de búsqueda

```
package gt.edu.usac.services.impl;

import java.net.URL;

import com.google.gdata.client.codesearch.CodeSearchService;
import com.google.gdata.data.codesearch.CodeSearchFeed;
import gt.edu.usac.services.SearchService;
/**
 * @author msamayoa
```

```
*/
public class SearchServiceImpl implements SearchService {

private static final String CODESEARCH_FEEDS_URL =
"http://www.google.com/codesearch/feeds/search?";

private CodeSearchService codesearchService;
private URL privateFeedUrl;
private String q;
private String startIndex, maxResults;

public SearchServiceImpl() throws Exception {
codesearchService = new CodeSearchService("plugin-gt.edu.sistemas.gcs.pluing");
q = null;
startIndex = null;
maxResults = null;
}



public CodeSearchFeed retrieveFeed()  throws Exception{
String feedUrl = builStringUrl();
if(feedUrl!=null){
privateFeedUrl = new URL(feedUrl);
return codesearchService.getFeed(privateFeedUrl, CodeSearchFeed.class);
}
return null;
}

public void setAdvanceQuery(String q, String startIndex, String maxResults) {
this.q = q;
this.startIndex = startIndex;
this.maxResults = maxResults;
}

public void setQuery(String q) {
this.q = q;
}
```

```
  private String builStringUrl(){
  String feedUrl = null;
  if(q!=null && q.length()>0){
  feedUrl = CODESEARCH_FEEDS_URL + "q=" + q;
  }else{
  return null;
  }
  if(startIndex!=null && startIndex.length()>0){
  feedUrl += "&start-index=" + startIndex;
  }
  if(maxResults!=null && maxResults.length()>0){
  feedUrl += "&max-results="+ maxResults;
  }
  return feedUrl;
  }
}
```

Fuente: trabajo de graduación

En el constructor de esta implementación se crea la instancia del servicio de búsqueda con la clase *CodeSearchService*. La cadena que recibe es únicamente un identificador interno, no tiene ninguna relación con algún tipo de identificación con el servicio de Google. Este servicio es gratuito y no es necesario tener una llave o licencia para su utilización.

Se podrán observar también dos métodos en los que se asigna el parámetro ‘q’. Este parámetro es la consulta que se hará al servicio. Es posible también especificar el índice mínimo y máximo de los resultados que se desean obtener.

Su integración con la clase *SearchAction*, que es la que administra la entrada de la consulta es importante de mencionar. En el método run de esta clase se crea la instancia de la implementación del servicio, se realizó en esta sección debido a que esta sección de código es la que se ejecuta cuando se

acciona el plug-in. Hay que notar también que la sección de código está controlada mediante un excepción en caso ocurriera un error, o no se pudiera obtener una conexión al servicio de Google.

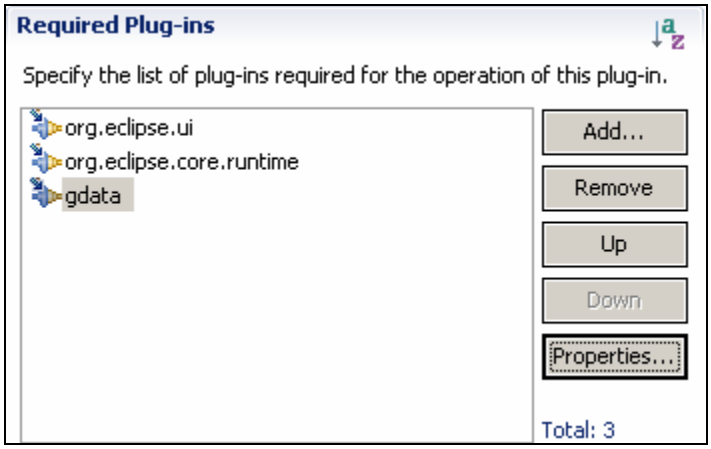
Un paso importante en el desarrollo de este plug-in es la correcta configuración de las dependencias. Es aquí cuando se utiliza el plug-in de dependencias externas elaborado anteriormente con las librerías requeridas por este plug-in, las API de Google. El plug-in fue nombrado gdata. Para su configuración se deberá ir al archivo del proyecto llamado 'plugin.xml'. Este archivo contiene varias secciones.

Se deberá seleccionar la pestaña de '*Dependencies*' o Dependencias. Luego se deberá agregar el plug-in desarrollado anteriormente, el cual contiene las librerías necesarias para realizar la conexión al servicio de búsqueda de Google.

Existen otros dos plug-ins de librerías que requiere este plug-in debido a la naturaleza del proyecto. El plug-in org.eclipse.ui y org.eclipse.core.runtime. Estos dos plug-in de librerías son automáticamente agregados al proyecto dado que utiliza componentes gráficos de interfaz de usuario.

Cuando se finalice de realizar las configuraciones de las dependencias del plug-in, se debería de obtener una configuración como la que se muestra en la siguiente figura.

Figura 23. Configuración de las dependencias requeridas por el plug-in



Fuente: trabajo de graduación

7.2.1. Archivo build.properties

Otro aspecto que vale la pena mencionar es que en el proyecto existe el archivo 'build.properties'. Este archivo contiene información sobre los directorios en donde se encuentra las fuentes del proyecto, es decir las clases. También contiene el directorio en el cual se generarán los archivos con extensión jar cuando sean compilados. Y por último tiene unas líneas que indican otro archivo de configuraciones como el META-INF. Posterior a esta línea se podrá encontrar un directorio en donde se especifican los recursos gráficos como iconos e imágenes que puedan ser utilizados por el plug-in.

Para este caso se tiene una configuración en donde hay un directorio llamado 'icons', el cual contiene una imagen que será el ícono para identificar el plugin.

Ejemplo 1. Archivo de configuración build.properties

```
source.. = src/
output.. = bin/
bin.includes = plugin.xml,\
               META-INF/,\
               .,\
               icons/
```

Fuente: trabajo de graduación

7.2.2. Archivo plugin.xml

Y para finalizar el archivo plugin.xml, contiene información que vale la pena mencionar que es importante para el correcto acceso del plugin. Este archivo dado que se encuentra en formato xml, el nodo raíz es llamado <plugin>.

Dentro de este existe otra etiqueta llamada <menu>. La cual permite configurar el acceso desde el menú de Eclipse. Es decir crear un menú desplegable sobre la barras de menús. Para definir la etiqueta que se mostrará el menú se deberá de asignar en el atributo 'label' de dicha etiqueta. Para crear un acceso directo mediante una combinación de teclas 'ctrl + letra', se deberá de anteponer a la letra los caracteres '&'. Esto le indicará que la combinación del acceso directo, será la tecla control más la tecla posterior.

Ejemplo 2. Configuración de menú de plug-in

```
<menu
    label="Búscar &amp;amp;Código"
    id="codeSearch">
</menu>
```

Fuente: trabajo de graduación

Y por último el nodo 'action'. Permite configurar la forma en que se tendrá acceso al plug-in. Es decir, desde menú contextual como el anterior. Permite definir también botones en la barra de herramientas y etiquetas informativas.

Ejemplo 3. Configuración de acciones del plug-in

```
<action
    label="Google &amp;amp;Code Search"
    icon="icons/gcs_icon.gif"
    class="gt.edu.usac.gcs.actions.SearchAction"
    tooltip="Code Search"
    menubarPath="sampleMenu/codeSearch"
    toolbarPath="codeSearch"
    id="gt.edu.usac.gcs.actions.SearchAction">
</action>
```

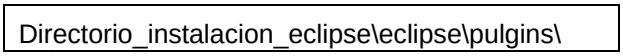
Fuente: trabajo de graduación

7.3. Instalación del plug-in y utilización en el ambiente de Eclipse

Ahora que ya se encuentra finalizado el plug-in, se procede a exportar de la misma forma que se exportó el plug-in de dependencias externas. Este de igual forma generará un archivo con extensión JAR. De forma que para instalar este plug-in se debe de copiar los dos plug-in. El de dependencias externas con nombre gdata_1.0.0.jar. Y el plug-in de búsqueda de código con nombre gt.edu.usac.gcs_1.0.0.jar.

Antes de instalar el pulg-in se debe tener en cuenta que Eclipse carga al inició todos sus componentes y plugi-ns. De esta forma se debe asegurar que cuando se copie los archivos de plug-in, Eclipse se deberá encontrar cerrado. O en caso estuviera abierto editando algún proyecto, se deberá reiniciar para que los cambios tengan efecto. Los archivos deben ser copiados al directorio donde se encuentra instalado el IDE, en el directorio de plugi-ns.

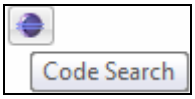
Figura 24. Directorio de plugi-ns de Eclipse



Fuente: elaboración propia

Una vez instalado el plug-in, en la próxima sesión iniciada de Eclipse se podrá acceder al plugin mediante acceso la barra de herramientas seleccionado el símbolo que se presenta en la siguiente figura.

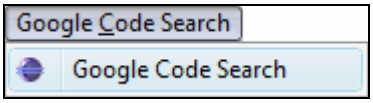
Figura 25. Icono en la barra de herramientas para acceso al plug-in



Fuente: trabajo de graduación

O bien mediante el acceso rápido “Atrl+C” al menú contextual “Google Code Search”.

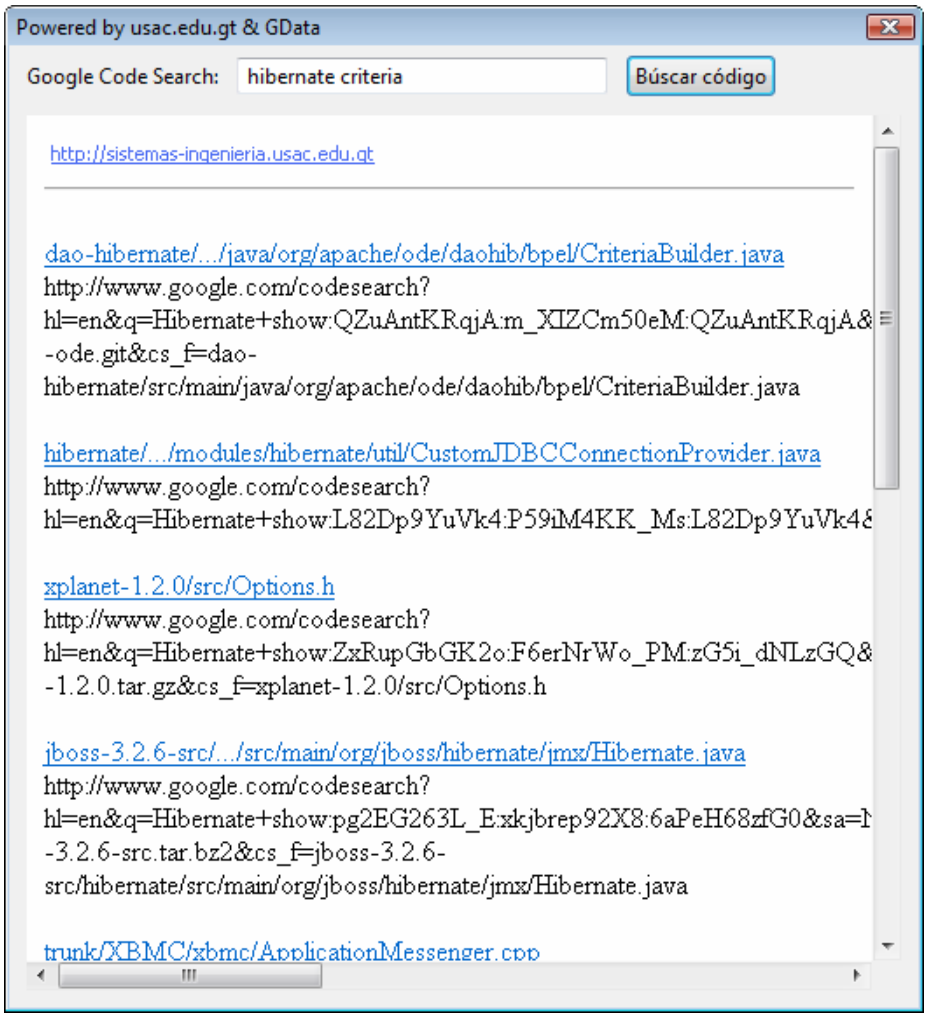
Figura 26. Menú contextual para acceso al plug-in



Fuente: trabajo de graduación

Se asume un ejemplo hipotético en donde un desarrollador con habilidades en el framework *Hibernate* desea realizar una consulta compleja sobre una entidad, para esto utilizará la clase *Criteria*, pero no recuerda exactamente la sintaxis. Puede hacer dos cosas, ir a Internet a buscar soporte en el API o bien consultar un manual de referencia. En ambos casos tiene que abandonar el entorno de desarrollo. Pero si en vez utiliza este plug-in podrá realizar la búsqueda desde Eclipse, en una ventana como la que se muestra a continuación.

Ejemplo 4. Resultados obtenidos de una consulta



Fuente: trabajo de graduación

Desde esta venta se puede navegar de igual forma que es un explorador. Claro existen ciertas limitaciones, pero la necesidad básica de buscar y encontrar información puede ser solventada sin perder mucho tiempo y productividad.

Veamos un ejemplo donde se puedan aplicar las expresiones regulares posix que se describieron en la sección 3.4. Se supone una consulta en donde

se desea buscar archivos de código fuente en donde únicamente busque en un repositorio CVS de código especificado.

Y es requerido que solo busque dentro un paquete específico pues se busca una clase con cierta funcionalidad. Dado que las expresiones posix, permiten ampliar y refinar los resultados se puede definir esta búsqueda con la siguiente entrada de consulta.

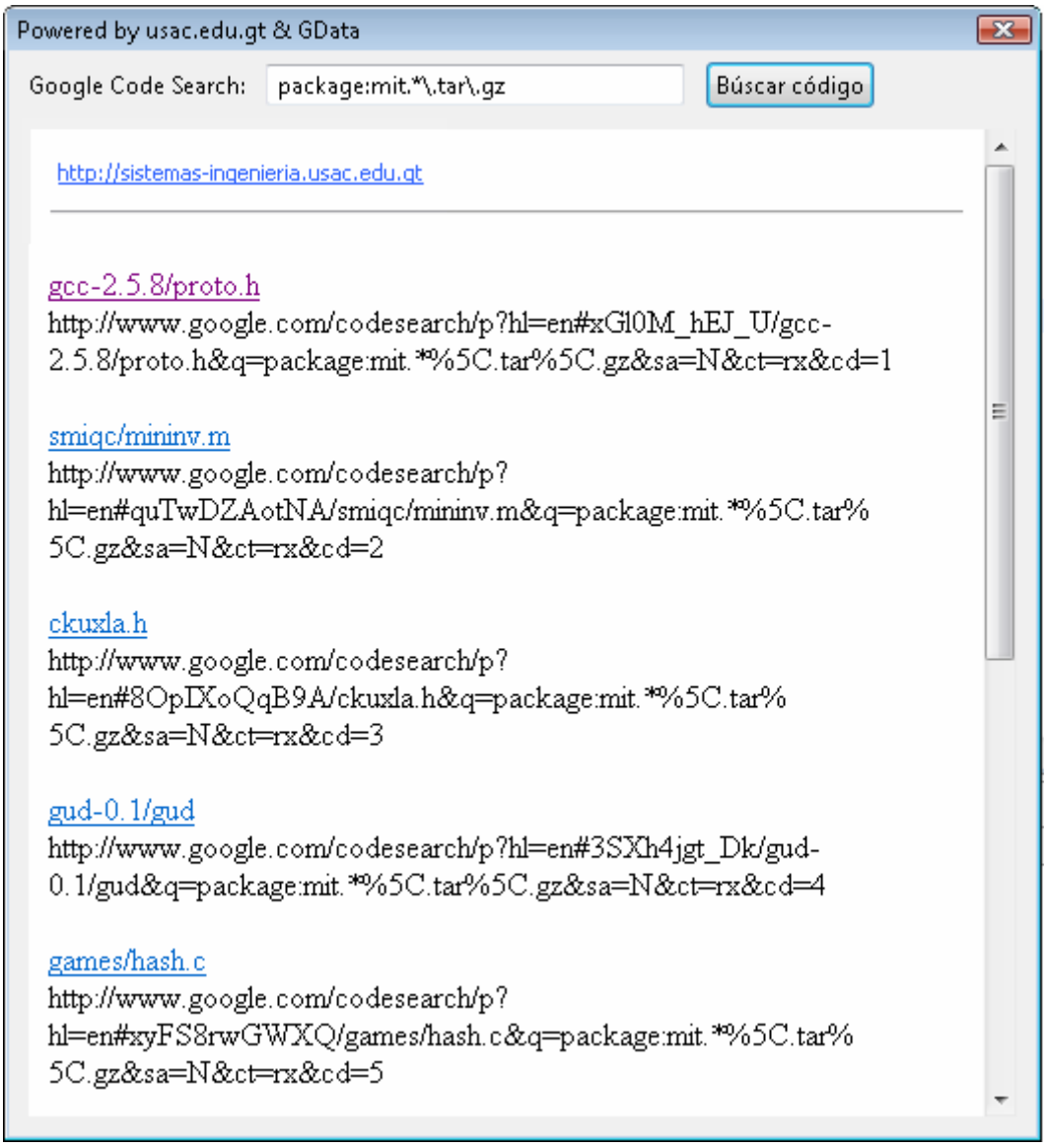
Ejemplo 5. Consulta usando ER Posix

```
package: url-cvs-servidor\paquete
```

Siguiendo como guía el ejemplo anterior, se puede escribir una expresión regular que busque en paquetes especificados según una expresión regular. Esta expresión regular puede ser el nombre de un servidor cvs que se encuentre publicado en algún dominio de un url.

Para el presente ejemplo se utilizará el dominio del Instituto Tecnología de Massachusetts, MIT. Se especificará que busque únicamente fuentes de código que se encuentren en paquetes tar.gz.*. En la siguiente figura se muestra la expresión utilizada y el resultado obtenido.

Ejemplo 6. Búsqueda de fuentes en el MIT



Fuente: trabajo de graduación

Como se observa en la imagen anterior, los resultados obtenidos fueron de la búsqueda en el repositorio de fuentes del MIT, dentro los paquetes tar.gz. Este tipo de búsquedas son útiles para localizar fuentes específicas. Por ejemplo, si quiere buscar el algoritmo para construir el fractal del conjunto de

Benoît Mandelbrot. Seguramente se podrá encontrar en repositorios de fuente como éste, en donde hemos obtenido resultados de fuente de código.

7.4. Desventajas

Una de las principales desventajas del desarrollo de este plug-in, es la complejidad que existe en el diseño de una interfaz gráfica utilizando los componentes del Framework SWT de Eclipse. Debido a que se tiene que respetar la filosofía de componentes e interfaces instalables. Esto requiere que el desarrollador de cualquier componente de un plug-in tenga nociones básicas sobre la arquitectura del SDK de Eclipse. Además se deberá de tener muy buena abstracción sobre la interfaz gráfica, pues no existe un diseñador que permita fácilmente pintar una pantalla. Toda la definición de las interfaces se debe realizar programáticamente.

El desarrollo del plug-in tiene dependencia de dos plataformas. En primera instancia corre sobre la plataforma Eclipse. Y por otra parte se tiene una dependencia del servicio de búsqueda de Google, dado que se utilizan las API de Datos de Google para buscar código fuente en repositorios de código.

Cuando se desea indexar un conjunto de archivos o librerías para poder ser localizadas con este servicio, se deben de publicar en Internet. Esto requiere un dominio y una dirección pública de Internet. Esto debido a que Google necesita la dirección del archivo o repositorio para poder indexarla en su base de datos cada nueva entrada en los archivos y repositorios de código.

Muchas veces la complejidad en el desarrollo de un componente de software puede variar dependiendo de la plataforma. En este caso se puede decir, que la complejidad para el desarrollo de un plug-in con dependencias

externas es media. Pues las librerías o dependencias, deberán ser primero transformadas en un plug-in, antes de ser utilizadas en otro plug-in. Puede ser una parte compleja si no se tiene la documentación adecuada y conocimiento técnico requeridos de la plataforma de desarrollo.

7.5. Ventajas

Uno de los mayores beneficios de utilizar este plug-in para un desarrollador sobre la plataforma de Eclipse, es la utilización de este servicio integrado al ambiente. Esto permite realizar búsquedas en el mismo ambiente de desarrollo sin necesidad de ir a un explorador y por ende perder tiempo.

Y dado que este es un componente que utiliza los API de Datos de búsqueda de Google, los resultados de una consulta serán específicos de código fuente. Llevándolo específicamente al archivo y línea específica en donde se localicen los resultados de la consulta. Esta herramienta puede ser muy útil cuando se desee localizar una función, procedimiento o extracto de código fuente de un repositorio que pueda tener varias versiones de un mismo archivo.

El desarrollar este plug-in utilizando las librerías cliente de Java de Google, permite una integración muy limpia debido a que Eclipse y su plataforma están escritos en código nativo de Java también. De forma que el rendimiento no se ve afectado. Ni tampoco la incompatibilidad con otras plataformas. Esa es una de las grandes ventajas de desarrollarlo utilizando tecnología como SWT.

Otra de las ventajas de seguir esta arquitectura y aprovechando los servicios de Google. Es que se pueden indexar varios repositorios de código fuente. Para proyectos de código abierto puede ser una herramienta muy útil y

de colaboración puesto que puede ayudar a integrar sistemas que pueden ser trabajados por varios grupos de personas al mismo tiempo.

CONCLUSIONES

1. Debido a la evolución de los protocolos de sindicación web como el GData que se encuentra basado en Atom y RSS, permitió ampliar los estándares de servicios públicos como feeds de sindicación de videos, noticias, elementos de código fuente, tareas, etc.

2. Este tipo de protocolo permitió que servicios como la búsqueda de código fuente y otros servicios de Google sean tan accesibles y fáciles de utilizar. Y una de las notables ventajas es que puede ser integrados con otras aplicaciones como se hizo con el plug-in desarrollado en la plataforma Eclipse.

3. Debido a que Google permite indexar repositorios de fuentes. Componentes como el buscador de código fuente integrado a Eclipse, pueden llegar a ser una herramienta que facilite la localización de piezas o componentes de código dentro de grandes repositorios de versionamiento.

4. Se pudo observar que para diferentes consultas realizadas con el componente de búsqueda utilizando expresiones regulares posix, permite no solo refinar resultados. Sino incluso realizar búsquedas más especificas dentro de grandes repositorios específicos, teniendo la capacidad de indicar en qué paquetes precisamente se debe de buscar e incluso indicando el lenguaje de búsqueda. Simplemente una característica realmente notable.

5. Aunque el Framework de desarrollo de interfaces SWT sea un poco más complejo al momento de escribir interfaces de usuario. Permite una independencia en cuanto al aspecto de plataforma. Lo que consiente que pueda ser utilizado en sistemas que necesiten interfaces cliente-servidor en donde sea requerido correr en varias plataformas sin que su aspecto se vea afectado.

6. Para el desarrollo de plug-in que necesite librerías de dependencias externas, es necesario crear un plug-in instalable el cual contenga estas librerías para posteriormente ser agregado como un plug-in al proyecto que se encuentre desarrollando.

7. Una relación entre complejidad y simpleza pudo ser detectada en la arquitectura basada en plug-ins de la plataforma Eclipse. Se puede pensar que muchos objetos simples de la naturaleza emergen de la complejidad. Pero estudios reciente demuestran que un simple conjunto de reglas muy básicas tiene la capacidad de generar patrones y objetos complejos, con una simple retroalimentación.

8. Que la filosofía de la plataforma de Eclipse se encuentra basada en pocas reglas simples; plug-ins, que a su vez pueden estar formados de otros plug-ins. Y estos mismos conforman la plataforma, simples reglas permite emerger sistemas más complejos. Lo que lleva replantear la relación entre complejidad y simpleza de los sistemas y las arquitecturas que los soportan.

9. Lo que puede lleva a decir que algunos sistemas complejos puede estar basados en reglas simples.

RECOMENDACIONES

1. Para la integración de los servicios que ofrece Google sobre el protocolo GData, se recomienda elaborar antes de su utilización una investigación sobre el lenguaje y el soporte de las librerías cliente para su posterior integración.

2. Se deberá de utilizar expresiones regulares posix para extender y refinar búsquedas de código fuente. Ya que estas permiten definir consultas más precisas y con mayores filtros.

3. Cuando se deba desarrollar un plug-in que requiera librerías externas, se debe recordar que la forma de hacerlo es mediante la creación de plug-in instalable que contiene a las dependencias. Y posteriormente agregarlo como un componente del plug-in en desarrollo.

4. El desarrollo de sistemas empresariales regularmente siguen una arquitectura distribuida dado que puede estar conformado por varios sistemas remotos. SOA, cliente-servidor son arquitectura muy utilizadas. Pero se recomienda el utilizar la arquitectura de componentes para el desarrollo de sistemas que puedan ser altamente transaccionales, tal y como lo hace la plataforma Eclipse.

5. Cuando se requiera el desarrollo de interfaces independiente de plataforma se recomienda la utilización de marcos de trabajo como Swing. En caso de requerir un cliente con independencia de plataforma

en cuanto a su aspecto se deberá de utilizar SWT, claro está para aplicaciones cliente-servidor basadas en lenguaje Java.

BIBLIOGRAFÍA

1. Algoritmo PageRank. Recuperado el 5 de septiembre de 2008, de http://es.wikipedia.org/wiki/PageRank.

2. Atributos y Elementos del formato de sindicación Atom, sección 4. Recuperado el 3 de septiembre de 2008, de http://www.atompub.org/2005/07/11/draft-ietf-atompub-format-10.html#rfc.section.4.2.1.

3. BackRub, la versión inicial del motor de búsqueda Google. Recuperado el 27 de agosto de 2008, de http://www.archive.org/.

4. Expresiones regulares básicas POSIX. Recuperado el 15 de octubre de 2008, de http://en.wikipedia.org/wiki/Regular_expression#Syntax.

5. Feeds de fuentes de código de Google. Recuperado el 20 de octubre de 2008, de http://www.google.com/codesearch/feeds/search.

6. Formato de sindicación Atom. Recuperado el 3 de septiembre de 2008, de http://tools.ietf.org/html/rfc4287.

7. Google code search Labs, Búsqueda de código fuente público. Recuperado el 25 de agosto de 2008, de http://www.google.com/codesearch.

8. Google code, librerías GData para cliente Java. Recuperado el 10 de octubre de 2008, de http://code.google.com/p/gdata-java-client/downloads/list.

9. Guía de referencia del programador de Eclipse (2005). “org.eclipse.platform.doc.isv.3.1.pdf”, páginas 2-20.

10. Guía de referencia para administrar Subversion 1.1 (2009). “svn-book.pdf”. Páginas 1-65.

11. Motor de búsqueda. Recuperado el 28 de agosto de 2008, de http://es.wikipedia.org/wiki/Motor_de_b%C3%BAsqueda.

12. Sindicación RSS y XML, lecturas en formato RSS. Recuperado el 29 de agosto de 2008, de http://es.wikipedia.org/wiki/RSS.

13. Sergey Brin and Lawrence Page (1998). "The anatomy of a large-scale hypertextual Web search engine". Recuperado el 25 de agosto de 2008, de http://infolab.stanford.edu/~backrub/google.html

14. Sindicación RSS. Recuperado el 3 de septiembre de 2008, de http://www.rssboard.org/.

15. Sistema de versionamiento Subversion. Recuperado el 25 de octubre de 2008, de http://www.open.collab.net/products/subversion/.

16. Sistemas distribuidos de versionamiento de código (WEBDAV). Recuperado el 10 de octubre de 2008, de http://www.ietf.org/rfc/rfc2518.txt.