Universidad Nacional Autónoma de México

Facultad de Ciencias

Diseño de una aplicación de búsqueda y recuperación de información para dispositivos móviles

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ISMAEL JONAS GALVÁN HERNÁNDEZ

TUTOR: DR. JORGE LUIS ORTEGA ARJONA

2010

A mis padres, Ismael y Odilia
A mi hermana, Judith
Y a mi Jazmín

Agradecimientos

A mi papá, por enseñarme a trabajar y hacer siempre las cosas bien desde un principio. Por su apoyo y ayuda en todo lo que hago.

A mi mamá, por apoyarme en el momento más difícil de la carrera. Por sus consejos y ayuda siempre.

A mi hermana, por ayudarme en lo que necesito.

Muy especialmente a mi novia Jazmín, por estar conmigo siempre en todo momento. Por su paciencia, comprensión y toda su ayuda.

Muy especialmente a mi asesor, Jorge L. Ortega Arjona, por su interés, por su tiempo, por todos sus consejos y la ayuda para mejorar éste trabajo. Por la paciencia y también por su apoyo.

A la maestra Guadalupe Ibargüengoitia, por su ayuda en la revisión del trabajo y por enseñarme la ingeniería de software y los beneficios que tiene en el ámbito profesional.

Al maestro Gustavo Márquez, por la revisión del trabajo y por orientarme en algunos puntos sobre el desarrollo de la aplicación.

Al Dr Héctor Benítez, por el tiempo para revisar el trabajo y sus comentarios respecto al enfoque de la tesis.

Al profesor Salvador López por la revisión del trabajo.

“ten, nine, eight, seven, six, five, four, three, two, one, freedom”
The Skatalites.

Hoja de Datos del Jurado

1. Datos del Alumno
   Galván
   Hernández
   Ismael Jonas
   56668050
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Ciencias de la Computación
   097279064

2. Datos del tutor
   Dr
   Jorge Luis
   Ortega
   Arjona

3. Datos del sinodal 1
   M. en C.
   Gustavo Arturo
   Márquez
   Flores

4. Datos del sinodal 2
   M. en C.
   María Guadalupe Elena
   Ibargüengoitia
   González

5. Datos del sinodal 3
   Dr
   Héctor
   Benítez
   Pérez

6. Datos del sinodal 4
   Mat
   Salvador
   López
   Mendoza

7. Datos del trabajo escrito
   Diseño de una aplicación de búsqueda y recuperación de información para dispositivos móviles
   180 p
   2010

Índice general

Índice de figuras

Resumen

El objetivo de esta tesis es diseñar e implementar una aplicación para un dispositivo móvil utilizando Solr para indexar y hacer búsquedas en una base de datos. La interfaz de usuario de la aplicación se desarrolla utilizando la plataforma Android.

Solr es un servidor de búsquedas basado en la biblioteca para búsquedas Lucene. Está escrito en lenguaje Java y tiene soporte para diversas características como categorías, replicación y diferentes formatos de respuesta para mostrar los resultados. Utiliza como formato de respuesta XML, binario y otros.

Android es una plataforma de software que incluye un sistema operativo para dispositivos móviles. Además, ofrece una plataforma de desarrollo que integra un emulador de dispositivo y perfiles de memoria, entre otras herramientas.

En la primera etapa de la tesis se define el diseño de la aplicación, la fuente de datos, la configuración del servicio de búsqueda, la indexación de la información y la lógica del motor de búsquedas y el tipo de respuesta que da el servicio web a la interfaz en el dispositivo móvil.

Para la segunda etapa se implementa la aplicación con el diseño que se realiza, se genera el código para la interfaz sobre el dispositivo móvil y se define la configuración para el emulador dentro del ambiente de desarrollo de Android.

Capítulo 1

Introducción

1.1. Contexto

Las personas han utilizado la computadora como una herramienta para facilitar sus tareas en el trabajo, en su casa o cuando salen de viaje. La posibilidad de tener una computadora en casa en los principios del desarrollo de estas herramientas era inimaginable, pues el hardware que se utilizaba era bastante grande y no se podía utilizar en cualquier ambiente. No se pensaba entonces en la posibilidad de trasladar la computadora a donde se necesitara.

Las computadoras han cambiado y han evolucionado. Ahora, además de tener computadoras de escritorio, es una realidad tener sistemas de cómputo móvil como PDA's[1] y algunos teléfonos celulares, que permiten acceso a nuestra información almacenada en una computadora de escritorio, haciendo cálculos, almacenando datos y pidiendo información tal vez a otros sistemas en otras partes del mundo.

Después de utilizar la computadora personal en las casas, escuelas , empresas y muchos sitios más, el uso que se le ha dado ha sido muy grande incluso sin tener conexión a una red. A partir de la aparición de Internet y la posibilidad de conectarse a otra máquina en otro lugar, la facilidad en la comunicación por medio de computadoras personales ha crecido. Sin embargo, se han tenido muchos problemas al tratar de establecer estándares para que toda la información almacenada en Internet pueda consultarse desde diferentes dispositivos y sistemas. Este problema es aún mayor en el caso de utilizar un dispositivo móvil.

Los usuarios han encontrado en los teléfonos celulares y los PDA's una forma sencilla y fácil de obtener información, almacenarla o intercambiarla. Esto es porque la cantidad de celulares hoy en día es mucho más grande que la cantidad de computadoras portátiles[2].

1.1.1. Aplicaciones para dispositivos móviles

Los dispositivos móviles (principalmente teléfonos celulares) actualmente tienen un poder de procesamiento necesario para realizar diversas funciones. Las empresas han visto en ellos una gran oportunidad para llegar a más personas, utilizando diversas aplicaciones que los usuarios necesitan.

[1] PDA significa asistente personal digital (Personal Digital Assistant).

[2] Son conocidas también como laptop, son pequeñas computadoras personales móviles.

A continuación podemos ver algunas de las categorías en las cuales podemos dividir a las aplicaciones para los dispositivos móviles[11]:

a. **Aplicaciones independientes** – Estas aplicaciones son conocidas como stand-alone[3] y se ejecutan solamente en el dispositivo utilizando sus propios recursos. Aplicaciones de entretenimiento, por ejemplo los juegos o las utilidades comunes como administradores de archivos y muchas más.

b. **Software de productividad personal** – Son programas que pueden o no utilizar una conexión a Internet, pero que básicamente se utilizan para nuestra vida cotidiana, como por ejemplo los procesadores de texto, presentaciones, notas o navegadores.

c. **Aplicaciones de conexión** – Son aplicaciones desarrolladas para conectarse a una red, principalmente a Internet. Este tipo de aplicaciones se enfocan en mandar datos y recibirlos. Algunas pueden utilizar los navegadores o conexiones a máquinas remotas, generalmente servidores.

d. **Aplicaciones para comercio** – Una de las mayores aplicaciones en los dispositivos móviles es la posibilidad de comprar productos en Internet usando los teléfonos celulares. Es una manera fácil y rápida para realizar compras desde cualquier lugar con cobertura.

e. **Aplicaciones especializadas** – En esta sección encontramos aplicaciones que son desarrolladas para fines específicos, por ejemplo, se pueden manejar inventarios de una forma remota o transferencias de cuentas, entre otros usos empresariales o personales.

f. **Aplicaciones de localización** – Esta es una categoría que ha surgido gracias a los dispositivos móviles. Facilita una localización sencilla y podemos saber la ruta que debe seguirse para encontrar un destino. Como ejemplo de estas aplicaciones podemos encontrar los mismos servicios de localización o guías interactivas en lugares como museos o centros comerciales.

La cantidad de aplicaciones que pueden realizarse en un dispositivo son muchas y gran parte de ellas aún están siendo desarrolladas, por lo que el campo para el desarrollo es todavía muy grande. En la actualidad, se debe desarrollar software específico para obtener la información de nuevos medios de comunicación, se debe iniciar la investigación de nuevas tecnologías, ya que son una herramienta muy importante para el futuro.

1.1.2. Búsqueda de información

Internet es actualmente la más grande fuente de información. Se utiliza en gran parte para hacer búsquedas sobre diferentes tipos de contenido, por ejemplo, buscar páginas de Internet que tienen la información necesaria.

Más del 80 % de la gente que utiliza Internet lo hace a través de motores de búsqueda para explorar páginas relacionadas a los temas que son de su interés. El tráfico que se genera en una página a partir del visitante que proviene de algún buscador es 85 % en promedio.[4]

[3] Cuando se habla de una aplicación stand-alone, significa que la aplicación no requiere de una conexión a Internet para funcionar.

[4] Es una aproximación que hay en diferentes páginas de Internet, por ejemplo la página de la Asociación Mexicana de Internet[14]

Un problema desde el inicio de Internet ha sido la falta de un estándar para la recuperación de información. No hay un orden para el gran número de páginas y sitios que existen en Internet, y no hay reglas generales que especifiquen procesos para la recuperación de datos para los diferentes dispositivos que existen.

Otro problema son los millones de páginas que están en Internet. Es una cantidad de información muy grande que hace imposible buscar con una simple revisión manual. Es por esto que ha surgido la necesidad de tener buscadores o programas que realicen la búsqueda automáticamente y encuentren los resultados que se necesitan.

Los buscadores se pueden dividir en tres categorías generales[53]:

a. *Web Crawler*

   Los buscadores tipo Web Crawler [5] utilizan programas para inspeccionar las páginas de Internet. Revisan datos que les sean útiles para almacenarlos y después incluirlos en la base de datos del buscador. Se crea un índice de todas las páginas y, de esta manera, se puede tener acceso más rápido a los sitios. Google, Bing y Lycos son ejemplos de este tipo de buscadores.

b. *Directorio*

   En los motores de búsqueda del tipo Directorio se mantiene un índice manual, al que los administradores del buscador le añaden las páginas a solicitud de los usuarios. Estos sistemas tienen una organización por categorías, aunque no tienen un algoritmo o un programa que defina la categoría para cada página, ya que esa tarea la realiza el encargado de indexar los contenidos. El buscador de Yahoo utiliza un directorio.

c. *Datos Específicos*

   En los motores de búsqueda de Datos Específicos se mantiene una base de datos en donde está toda la información disponible y, a través de una interfaz entre el programa cliente y el servidor de base de datos, se obtienen los resultados según la búsqueda sobre las tablas. La interfaz puede ser web, aplicaciones stand-alone o programas cliente-servidor. Estos buscadores forman parte de sistemas propietarios, son aplicaciones que se agregan a los sistemas existentes en las empresas.

1.2. Planteamiento del problema

Actualmente existen motores de búsqueda para Internet como Google, Yahoo Search, Bing, etc. Estos buscadores están enfocados a la páginas web. Buscan información a través de Internet y manejan sus resultados como enlaces hacia la página web que contenga la información que el usuario busca.

La mayoría de estos buscadores de páginas tienen su versión para dispositivos móviles. En algunos casos estas versiones son sólo una interfaz para su sistema de búsquedas general, es

[5] Araña Web o Web Crawler, son programas que indexan las páginas y su contenido en la Web.

decir, es un programa intermedio que hace la consulta al motor de búsquedas principal y trabaja de la misma manera, ya que solo cambia la vista de resultados para adaptarse a un dispositivo móvil.

Existen buscadores para la información que está en Internet, pero no se dispone de un buscador personalizado que pueda dar la información que está almacenada en una base de datos en un servidor por medio de un dispositivo móvil.

Las características que un buscador de este tipo necesita son:

a. Integrar un componente para recuperar la información almacenada en una base de datos existente, mostrando esa información en un dispositivo móvil.

b. Realizar búsquedas desde un dispositivo móvil, sin el uso de navegadores web.

c. Ofrecer la posibilidad de manejar los datos en el dispositivo móvil.

1.3. Hipótesis

Se puede desarrollar una aplicación de búsqueda y recuperación de información que consulte bases de datos que permitan conexiones JDBC[6], y que sus tablas estén indexadas utilizando el motor de búsquedas Solr. La interfaz se puede ejecutar en un dispositivo con plataforma Android y que permita conexiones a red.

1.4. Aproximación

Un servicio de búsqueda puede implementarse utilizando Solr en un servidor de aplicaciones web. Este servicio devuelve los resultados en un formato específico para que otra aplicación utilice esa información.

Una interfaz Android puede hacer una conexión al servidor Solr y puede obtener el resultado por medio de una petición HTTP al servidor que ofrece el servicio.

Si ya existe una fuente de datos, es necesario conectar el servicio de indexación y búsqueda al manejador de la base de datos a través de un conector JDBC, si no existe una fuente de datos se debe crear primero la base de datos que almacene esa información.

1.5. Contribuciones

Como contribuciones podemos mencionar las siguientes:

1. *Diseño de un buscador para bases de datos desde una aplicación móvil*:

   El diseño permite hacer una implementación con otras tecnologías. No se define en base a una en especial, y además, considera la escalabilidad para extender su funcionalidad.

[6]JDBC se refiere a Java Database Connectivity, una especificación de Java para realizar operaciones con una base de datos

2. *Interfaz de usuario del dispositivo móvil para el servidor de búsquedas Solr*:

   Se propone una interfaz para un dispositivo móvil con sistema operativo Android, para desplegar la información indexada con Solr. Esta es una forma de conectar aplicaciones ya definidas utilizando dispositivos móviles.

3. *Una forma de integración de Solr y Android*:

   La comunicación de Solr y Android se hace por medio de peticiones HTTP. El formato de resultado en JSON[7], para visualizar los resultados y la posibilidad de agregar funciones adicionales.

1.6. Estructura de la tesis

La tesis se divide en los capítulos que se describen a continuación:

- El capítulo 2 es una introducción al desarrollo de aplicaciones para dispositivos móviles. Se hace una breve descripción de las tecnologías que se usan en el desarrollo del proyecto *Tlatemoani*.
- En el capítulo 3 se revisan los trabajos relacionados con ésta tesis. Se mencionan los diferentes motores de búsqueda que existen y sus características. Se mencionan también las aplicaciones WAP y aplicaciones con buscadores integrados en el sistema móvil. Se analizan las bases de datos internas en un dispositivo móvil y la solución que dan a las búsquedas.
- El diseño de la aplicación se desarrolla en el capítulo 4. Se describen los aspectos generales de la aplicación de búsqueda y se divide el diseño en dos etapas: la primera es la parte del servicio de búsqueda; la segunda es la interfaz de usuario.
- En el capítulo 5 se muestra la implementación del proyecto *Tlatemoani* de una manera más detallada, mostrando la configuración de las herramientas para el desarrollo y algunos elementos principales dentro del código de la aplicación.
- En el capítulo 6 se muestran las conclusiones que se obtienen al finalizar el desarrollo de este trabajo.
- Los archivos de configuración necesarios para el servidor de búsquedas y el emulador del dispositivo con Android, se pueden consultar en el Apéndice A. El código completo de la aplicación se muestra en el Apéndice B. En el Apéndice C están las pruebas al sistema.

[7] JSON es un formato para el intercambio de información a través de objetos y arreglos o estructuras de datos simples

Capítulo 2

Antecedentes

En este capítulo se analizan las diferentes tecnologías utilizadas en la implementación de la aplicacíon de búsqueda. La primera parte de este capítulo introduce el concepto de dispositivo móvil y se dan ejemplos de los dispositivos que se consideran móviles. Se mencionan también las herramientas que se necesitan para el desarrollo de software para los dispositivos móviles. Se describe el campo de la recuperación de la información y algunos modelos de sistemas IR(Infromation Retrieval)[1]. Además se introduce el concepto de "Búsqueda de *Texto Completo*".

En la segunda parte del capítulo se muestra una introducción al desarrollo de aplicaciones Android. Se muestra el conjunto de software y las características del sistema operativo, así como algunas de las bibliotecas principales y la máquina virtual. Como parte del desarrollo, se analizan algunas características básicas del emulador, la interfaz de usuario y los componentes fundamentales en una aplicación Android.

En la parte final de este capítulo se describe el servidor de búsquedas Solr. Se comienza describiendo la biblioteca IR llamada Lucene, y se menciona la integración de Lucene con otros componentes para diseñar un motor de búsquedas básico. Se mencionan algunas clases básicas para el desarrollo de aplicaciones con Lucene. Además se describe el servidor de búsquedas Solr a través de las características principales de Lucene, y se hace una comparación entre la funcionalidad de Solr y la tecnología de bases de datos para la búsqueda de información.

2.1. Aplicaciones para dispositivos móviles

En este trabajo se mencionan los dispositivos móviles y se consideran como:

Aquellos dispositivos electrónicos digitales que los usuarios pueden llevar consigo y que se caracterizan por tener un tamaño reducido, que caben en el bolsillo o en la palma de la mano. Tienen una plataforma de software que permite la instalación de aplicaciones externas. Son asistentes personales, teléfonos celulares, que ahora se han convertido en pequeños sistemas de cómputo, aunque no llegan al tamaño y robustez de una notebook[11].

A continuación se describe en detalle las características de algunos dispositivos móviles.

[1] IR se refiere al campo de la Recuperación de la Información.

2.1.1. Dispositivos móviles

Existen muchos tipos de dispositivos móviles. Podemos encontrar una gran variedad de modelos y marcas de asistentes personales, teléfonos celulares y una clase particular son los *Smartphones*. Existen otros tipos de dispositivos como los set-boxes[2] y las smart cards[3]. Aunque estos dispositivos también pueden ser programados, los PDA y los smartphones son los dispositivos más comunes, sobre los cuales se desarrollan más aplicaciones en el mercado. Por esta razón, este trabajo se enfoca al desarrollo para dispositivos del tipo PDA y smartphone únicamente.

2.1.1.1. Teléfonos celulares

El teléfono celular es un dispositivo móvil que sirve para la comunicación. El tamaño de los teléfonos es muy pequeño, por lo que pueden transportarse a cualquier sitio y tienen movilidad completamente. La movilidad trae consigo la posibilidad de transferir datos a través de diferentes medios, por ejemplo, puertos infrarrojos IRDA[4] y tecnología Bluetooth[5]. Estas tecnologías han sido de gran utilidad para funciones generalmente utilizadas por las aplicaciones de productividad personal, juegos, administración de archivos y datos, entre otras.

2.1.1.2. PDA

En lo que se refiere a PDA, su nombre proviene de *Personal Digital Assistant*. Es un pequeño sistema cómputo móvil. En un principio funcionó como agenda electrónica para convertirse después en un importante dispositivo de administración de información personal.

Todos los PDA poseen un sistema operativo con diversas utilidades de administración o entretenimiento. Podemos instalar aplicaciones personalizadas. Además, tienen puertos para conectarse a periféricos de una computadora de escritorio. También cuentan con soporte para diferentes tecnologías como Bluetooth, comunicación por medio de puertos infrarojos, tecnología TouchScreen [6] (también conocida como pantalla táctil), un Stylus o lápiz[12].

Una característica importante es que no tienen teclado, pero puede incluirse como un accesorio.

Los PDA también nos permiten llevar una gran cantidad de información, ya que algunos de estos dispositivos cuentan con tarjetas de memoria extras además de la memoria principal. Estas tarjetas son usadas para almacenar información del usuario como documentos, imágenes o videos. Los PDA también permiten la conexión a Internet.

2.1.1.3. Smartphone

Son dispositivos electrónicos que integran la funcionalidad de un teléfono celular y que además cuenta con funciones similares a un asistente personal digital.

[2]Un set-box es un dispositivo que tiene diferentes funciones entre ellos, sirven para recibir una señal dígital, autenticación de derechos de acceso, entre otras.

[3]Una smartcard es un dispositivo de cómputo móvil que se utiliza para almacenar información de forma segura, es una alternativa a las tarjetas de crédito convencionales.

[4]IRDA Infrared Data Association, define un estándar para la transmisión de datos por medio de rayos infrarrojos.

[5]Bluetooth es una especificación para redes inalámbricas que permite la transferencia de datos.

[6]Es una pantalla que permite el toque físico sobre su superficie para realizar diversas funciones.

Una de sus características más importantes es que se pueden instalar programas desarrollados por las compañías fabricantes del dispositivo, software de terceros o por los operadores del servicio de telefonía. Otras características es que cuentan con un sistema operativo, acceso a internet, cámaras dígitales integradas y la posibilidad de consultar archivos de texto o presentaciones. Los smartphones tienen diferentes tipos de sistemas operativos, por ejemplo, Symbian, Windows Mobile, Palm OS, iPhone OS, Android y algunos más dependiendo de las compañías que los fabrican[11].

2.1.2. Desarrollo de aplicaciones móviles

Las tecnologías son importantes para el desarrollo de las aplicaciones móviles porque es necesario saber cuál es la que ofrece más ventajas para cumplir con los requerimientos del sistema de software que se quiere desarrollar. Además es importante tener una herramienta que permita desarrollar las aplicaciones de forma rápida y de manera más simple. También se deben considerar algunas características de los dispositivos para construir una aplicación.

2.1.2.1. Consideraciones para el desarrollo

Hay varios factores que se deben tener en cuenta cuando se escriben aplicaciones para dispositivos móviles o sistemas embebidos[7].

Por lo general los dispositivos móviles cuentan con las siguientes características[12]:

1. *B*ajo poder de procesamiento
2. *M*emoria RAM limitada
3. *C*apacidad limitada de almacenamiento permanente
4. *P*antallas pequeñas con baja resolución
5. *A*ltos costos asociados a la transferencia de datos
6. *B*aja velocidad de transferencia de datos con una latencia[8] alta.
7. *D*uración de la batería limitada
8. *M*ulti–procesamiento

En estos dispositivos el tamaño de la pantalla es pequeña, la memoria es limitada y la capacidad de procesamiento es reducida. Además, la capacidad de almacenamiento es menor comparada con computadoras de escritorio o laptops.

[7] Un sistema embebido es un sistema que puede ser incluído dentro de un sistema de cómputo más grande, es decir, se puede agregar como un componente más a otro sistema.

[8] Se le llama latencia a la suma de retardos dentro de una red.

2.1.2.2. Sistemas operativos

Para comprender los sistemas operativos es necesario conocer dos conceptos importantes. En primer lugar el *hardware*, que está formado por todos los componentes materiales de una computadora y constituye la base operacional del software. Y el *software*, que es el conjunto de aplicaciones, instrucciones que se ejecutan sobre el hardware. Un programa de software debe utilizar de alguna manera el hardware para la entrada o salida de datos o para operar los componentes del dispositivo.

Los sistemas operativos están diseñados para administrar los diferentes recursos del hardware de una computadora. El sistema operativo construye un modelo que permite el acceso y uso de esos recursos. También combina el hardware y el software de una computadora o un dispositivo[9].

2.1.2.3. Herramientas

Para el desarrollo de aplicaciones móviles es necesario hacer uso de un conjunto de herramientas. A continuación se explican conceptos básicos que se deben conocer para comenzar con el desarrollo para dispositivos móviles. Las aplicaciones se escriben utilizando lenguajes de programación. Algunas veces las propias plataformas móviles los ofrecen, es decir, con el dispositivo móvil se ofrece un sistema operativo y un conjunto de herramientas para que el desarrollador las utilice para crear sus aplicaciones.

Un Entorno de Desarrollo Integrado(conocido como *IDE*) es una aplicación de software que ayuda a los programadores a desarrollar sus aplicaciones para diversas plataformas y lenguajes de programación. Un *IDE* generalmente consta de los siguientes elementos[49]:

1. Un editor de código fuente.
2. Un compilador o un intérprete para el lenguaje de programación.
3. Herramientas gráficas para construir la estructura de la aplicación.
4. Un depurador de código.
5. Sistema para agregar plug-in[9].

Además de estas herramientas básicas, existen otros componentes adicionales cuando se trata del desarrollo para dispositivos móviles, por ejemplo:

1. Un emulador de dispositivos.
2. Un convertidor a código binario especial para ser ejecutado en el dispositivo.
3. Una herramienta para empaquetar la aplicación.
4. Herramientas visuales para el diseño de la interfaz de usuario.

Algunos IDE's conocidos son Microsoft Visual Studio, NetBeans de la compañía Sun Microsystems y Eclipse para el desarrollo de aplicaciones en Java principalmente, y que permite la integración con Android.

[9] Un plug-in es un complemento que puede agregarse a un sistema de cómputo para ampliar su funcionalidad.

El emulador es una parte muy importante en el desarrollo de aplicaciones móviles, ya que permite probar los desarrollos sin tener físicamente el dispositivo, aunque tiene algunas limitaciones en cuanto a la comunicación y simulación de algunos servicios.

Por lo general se pone a disposición del desarrollador la Application Programming Interface(API) que define la manera en que el programa puede solicitar servicios a las bibliotecas nativas. Por ejemplo, la plataforma Android proporciona diferentes API's para el desarrollo de aplicaciones. Otros ejemplos son las API de iPhone, Google Maps o Java.

Es muy común tener el apoyo de frameworks además de los API's. Los frameworks son diseños reusables para un sistema de software. Un framework de software puede incluir bibliotecas o un conjunto de programas que ayudan a desarrollar o combinar otros componentes de un proyecto de software[15].

Por último, para cada plataforma se ofrece un kit de desarrollo conocido como SDK. Normalmente es un conjunto de herramientas que permiten el desarrollo de aplicaciones específicamente para esa plataforma. Un SDK puede contener APIs, documentación y algunos ejemplos sencillos de aplicaciones. El IDE es una aplicación gráfica que integra diferentes componentes, uno de ellos puede ser un SDK.

2.2. Recuperación de información

A través del tiempo las personas han comprendido la necesidad de almacenar y buscar información. Con la ayuda de las computadoras se han recolectado grandes cantidades de información y la búsqueda de datos útiles en esas colecciones se ha convertido en una actividad indispensable.

Almacenar y recuperar información se convirtió en una necesidad y con la ayuda de las computadoras fue posible automatizar los procesos de búsqueda. En 1945 Vannevar Bush publicó un artículo llamado *As We May Think*[20] en el que surgió la idea de la recuperación de información de forma automática en los sistemas que contenían grandes cantidades de datos, y en 1950 esta idea se realizó a través de las descripciones más detalladas de los archivos de texto que podían ser buscados de forma automática.

La recuperación de información(IR) se encarga del estudio de la búsqueda de información en documentos de manera relevante, de la búsqueda de los mismos documentos o de búsquedas también en bases de datos. La búsqueda puede ser a través de Internet, de forma local o de manera remota. Los documentos pueden ser texto, imágenes, sonido o datos de otras características.

Existen buscadores de información como Google y Bing entre otros, que son algunas de las aplicaciones más populares para la búsqueda de información en Internet. Un buscador es una aplicación que permite realizar una consulta para obtener resultados que coincidan con un criterio de búsqueda. Algunos buscadores utilizan un vocabulario, es decir, una lista de términos en lenguaje natural además de un algoritmo de búsqueda y otro algoritmo para la valoración de los resultados.

2.2.1. Modelos

En un principio, los modelos de sistemas IR permitían a los usuarios especificar sus búsquedas usando una compleja combinación de operadores booleanos como AND, OR y NOT. Los sistemas basados en operadores booleanos tienen desventajas, por ejemplo, no se contempla la clasificación de documentos y es muy difícil para un usuario formar un criterio de búsqueda complejo[19].

Actualmente muchas aplicaciones de recuperación de información utilizan otros modelos que permiten la clasificación de resultados, asignando una puntuación numérica a cada documento para cada consulta del usuario. Varios modelos se han propuesto para realizar este proceso. Los más usados son el modelo de espacio vectorial y los modelos probabilísticos.

El modelo de Espacio Vectorial (Vector Space Model) es un modelo algebraico para representar documentos de texto(y cualquier objeto en general) como vectores de indentificación, por ejemplo, los términos de un índice. Este modelo se utiliza en el filtrado de información, en la recuperación de la información, para la indexación y clasificación de la relevancia de los resultados. Un *documento* se representa como un vector, y cada dimensión corresponde a un término independiente. La defínción de término depende de la aplicación en donde se utiliza, pero generalmente el término se refiere a una palabra o también a frases muy largas[58]. Por ejemplo un documento puede ser la frase "Recuperación de información" y un término es "información".

Los modelos probabilísticos (Probabilistic Models) tratan el proceso de recuperación de información como una inferencia probabilística. Las similitudes se calculan como la probabilidad de que un documento sea relevante para la consulta determinada[51].

2.2.2. Búsqueda de texto completo (Full-Text Search)

Es una técnica para buscar información que está almacenada en una base de datos. Un motor de búsqueda puede revisar todas las palabras en los documentos que son indexados para encontrar las coincidencias con las palabras que el usuario quiere encontrar.Cuando se realiza una consulta sobre un conjunto pequeño de documentos se puede realizar un escanéo de cada uno para encontrar las palabras que se buscan, pero cuando la cantidad de documentos es muy grande, se utiliza la búsqueda de texto completo. La tarea de buscar se divide en 2 etapas: la primera es la de indexar el contenido de los documentos y crear una lista de términos que generalmente se llama *índice*; la segunda etapa es buscar una o más palabras. Ya que se tiene un índice con términos asociados a los documentos en donde aparecen y otros datos llamados metadatos, la búsqueda se hace sobre este índice para que sea más rápida.

2.3. Desarrollo de aplicaciones móviles con Android

Actualmente, en el campo del desarrollo de aplicaciones para dispositivos móviles, encontramos en el mercado teléfonos celulares que utilizan una gran variedad de sistemas opera-

tivos, como Microsoft's Windows Mobile[10], Symbian OS[11], Mobile Linux[12], iPhone OS[13] y muchos otros sistemas propietarios. Google entró en el mercado con su plataforma Android, enfocándose a características como el desarrollo de la plataforma, ofreciendo código abierto y un framework altamente enfocado al desarrollo de interfaz de usuario.

En 2005 Google adquirió la compañía Android Inc, para comenzar con el desarrollo de la plataforma Android y se comienzó con el desarrollo de la máquina virtual *Dalvik*[16]. En 2007 varias empresas se unieron para formar la Open Handset Alliance. Entre sus miembros están Google, Motorola, Samsung, Sony Ericsson, Intel, Texas Instruments, entre otros, que pueden consultarse en su sitio web[14]. En octubre del 2008 Google hizo disponible el código fuente de la plataforma Android bajo la licencia de código abierto Apache. En abril de 2009 se liberó la versión 1.5 del SDK de Android, y en enero del 2010 la versión 2.1. Las características se revisan a detalle más adelante[21].

2.3.1. La plataforma Android

Android es una plataforma de software para el desarrollo de aplicaciones móviles. En esta sección hablaremos de Android, sus características, el entorno de desarrollo y lo que puede ofrecer.

La plataforma Android, está pensada para el desarrollo móvil específicamente, contiene los siguientes elementos:

1. Un kernel Linux que proporciona la interfaz de bajo nivel con el hardware, manejo de memoria y control de procesos, todo optimizado para dispositivos móviles.
2. Bibliotecas de código abierto para el desarrollo de aplicaciones que incluyen SQLite, WebKit, OpenGL ES y un gestor de multimedia.
3. Un entorno de ejecución que incluye una máquina virtual y bibliotecas nativas que proveen funcionalidad específica.
4. Un framework que ofrece un sistema de servicios, incluyendo el gestor de ventanas, los proveedores de contenido, gestor de localización, telefonía y servicios como P2P[15].
5. Un framework para la interfaz de usuario.
6. Aplicaciones preinstaladas.

El SDK de Android soporta la mayoría de la especificación J2SE, excepto algunos componentes como el de AWT y Swing. En su lugar, Android ofrece un amplio framework para la construcción de la interfaz de usuario.

El lenguaje de programación Java, tiene una máquina virtual que es la responsable de interpretar el bytecode del lenguaje en tiempo de ejecución. Una máquina virtual da la optimización

[10] `http://www.microsoft.com/windowsmobile`
[11] `http://www.symbian.org/`
[12] `http://www.linuxfoundation.org`
[13] `http://developer.apple.com/devcenter/ios/gettingstarted/docs/iphoneosoverview.action`
[14] `http://www.openhandsetalliance.com/oha_members.html`.
[15] Se refiere a una red Peer to Peer, o de par a par, en la que las computadoras funcionan como iguales entre si.

necesaria para que el rendimiento del lenguaje interpretado sea comparable a lenguajes compilados como C o C++. Android tiene su propia máquina virtual para ejecutar los archivos generados con Java. La máquina virtual se llama *Dalvik VM*.

2.3.1.1. La máquina virtual Dalvik

Los requisitos de desempeño en los teléfonos celulares son muy exigentes. Los diseñadores de los dispositivos necesitan optimizar el código lo más posible. Si revisamos los paquetes de Android veremos que son muchos y muy extensos en cuanto a cantidad. Según Google, las bibliotecas de sistema pueden utilizar hasta 10 Mb, incluso con una máquina virtual optimizada[4].

Se consideraron aspectos de rendimiento al desarrollar la Dalvik JVM[16]. En primer lugar la máquina virtual Dalvik toma los archivos de clase generados(.class) y los combina en uno o más archivos ejecutables Dalvik (.dex)[22]. Se reusa información duplicada desde múltiples archivos de clase, esto reduce la necesidad de espacio(sin comprimir) por medio de un archivo tradicional jar. Por ejemplo, el archivo .dex de la aplicación web-browser en Android es de 200 Kb apróximadamente, mientras que el equivalente sin comprimir es de 500 Kb. En segundo lugar se ha refinado la recolección de basura en Dalvik VM en la versión más reciente[17].

La máquina virtual Dalvik utiliza una clase diferente de código ensamblado, utiliza registros como unidades primarias de almacenamiento de datos en lugar de una pila como se utiliza en las máquinas virtuales en J2ME, la KVM y CVM[12].

Debemos señalar que el código ejecutable final en Android, como resultado de la máquina virtual Dalvik, no se basa en el bytecode de Java. En su lugar están los archivos: dex. Esto significa que no puede ejecutarse directamente el bytecode de Java. Se debe comenzar con los archivos de clase java(.class) y luego convertirlos para ser enlazados a los archivos .dex.

El Android SDK utiliza ampliamente XML para definir capas de interfaz de usuario, sin embargo, todo este XML es compilado en archivos binarios antes de que residan en el dispositivo. Android proporciona mecanismos especiales para usar estos datos XML[4].

[16] Java Virtual Machine, es un programa nativo encargado de ejecutar código binario, en este caso Java bytecode, que es generado por el compilador de Java.

2.3.1.2. Entorno de software Android

A continuación se puede ver en la siguiente figura el entorno de software de Android:

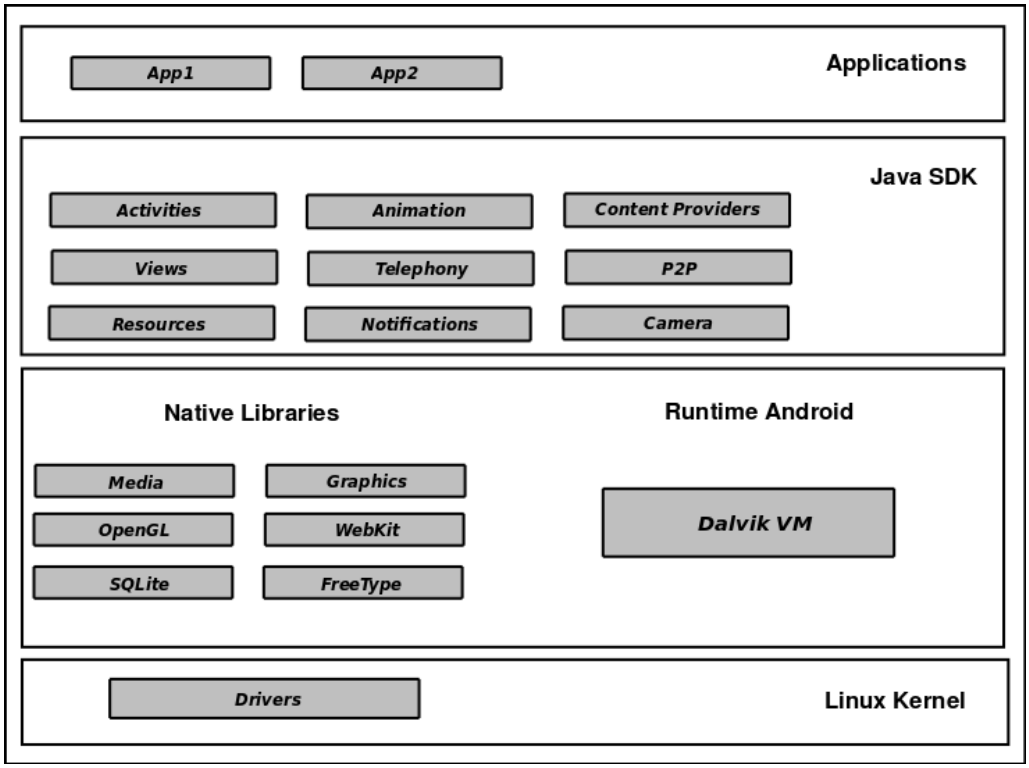

Figura 2.1: Entorno Android

En el primer nivel está el núcleo de la plataforma Android, el kernel Linux. La versión del kernel es la versión 2.6, y es el componente responsable de los controladores de dispositivo, el acceso a los recursos, la gestión de energía y otras funciones del sistema operativo. Los controles de dispositivo suministrados incluyen pantalla, cámara, teclado, wifi, memoria flash y audio. Aunque el núcleo es Linux, la mayoría de las aplicaciones en un dispositivo Android están desarrolladas en Java y se ejecutan a través de la máquina virtual Dalvik.

En el segundo nivel, en la parte superior del núcleo están una serie de bibliotecas en C y C++, como el OpenGL, WebKit, FreeType, Secure Socket Layer (SSL), la biblioteca runtime de C (libc), SQLite, entre otras. El sistema de bibliotecas C está basado en BSD[17]; se adaptó para dispositivos móviles basados en Linux.

La biblioteca WebKit es responsable de dar soporte al navegador web. Es la misma biblioteca que da soporte al navegador Chrome de Google y a Safari de Apple Inc. La biblioteca FreeType es responsable del soporte para las fuentes de texto. SQLite es un manejador de bases de datos relacionales de código abierto que está disponible en la plataforma Android.

Las bibliotecas para multimedia se basan en OpenCore PacketVideo[18], éstas bibliotecas son

[17] BSD, Berkeley Software Distribution una distribución basado en Unix.
[18] Se puede consultar más información en : `http://www.packetvideo.com/`.

responsables de la grabación y reproducción de audio y video. Una biblioteca llamada Surface Manager controla el acceso al sistema de visualización y da soporte para gráficos en 2D y 3D[1].

La mayoría de los accesos a las bibliotecas básicas se hace a través de la máquina virtual Dalvik, la puerta de entrada a la plataforma Android. Como se ha mencionado Dalvik está optimizada para ejecutar múltiples instancias, por ejemplo, los accesos de las aplicaciones Java a las bibliotecas principales. Cada aplicación obtiene su propia instancia de la máquina virtual. La Dalvik VM es compatible con el SDK 5 de Java pero optimizado para la plataforma Android, sin embargo, algunas características pueden diferir ya que la versión Java SE en Android es un subconjunto de la plataforma completa.[4]

En el tercer nivel están las bibliotecas principales del API Java. Incluyen la telefonía, recursos, localizaciones, interfaz de usuario, proveedores de contenido, paquetes de administración(instalación, seguridad, entre otros)[3]. Los programadores desarrollan aplicaciones para el usuario final en la parte superior de este nivel. Algunos ejemplos de aplicaciones en el dispositivo incluyen por ejemplo agenda, teléfono, navegador web, juegos, etc.

Las API's 3D en Android se basan en una implementación de OpenGL ES del grupo Khronos[19]. OpenGL ES es un subconjunto de OpenGL que está enfocado a sistemas embebidos.

En el último nivel están las aplicaciones de terceros, es decir, aplicaciones que son desarrolladas por empresas y usuarios independientes.

Desde la perspectiva de los medios de comunicación, Android soporta los formatos más comunes de audio, video e imágenes. Desde el punto de vista inalámbrico, se ofrecen API's para el soporte de Bluetooth, EDGE, 3G, WiFi y telefonía GSM. Esto es en función del hardware.

2.3.2. Desarrollo con Android

Para el desarrollo de aplicaciones con Android no es necesario un entorno de programación integrado(IDE). Sin embargo, para facilitar el trabajo se utiliza para este proyecto el IDE llamado *Eclipse*, ya que cuenta con un plug-in para el desarrollo en Android llamado Android Development Tools(ADT) que permite desarrollar, depurar y probar las aplicaciones.

Como se mencionó, no es necesario utilizar un IDE para desarrollar aplicaciones Android. Se puede utilizar simplemente las herramientas de línea de comandos que se proporcionan en el SDK. En ambos casos, es necesario el soporte para un emulador de dispositivo para ejecutar, depurar y probar las aplicaciones, no se necesita un dispositivo real en el 90 % del tiempo durante el desarrollo de las aplicaciones[21].

2.3.2.1. El emulador

Un emulador facilita el desarrollo ya que permite de forma más rápida el proceso de compilación, ejecución y depuración. Un dispositivo real es la mejor opción al probar la aplicación final. Durante el proceso de desarrollo la mejor opción es utilizar un emulador por el tiempo en el que se carga una aplicacíon en el dispositivo real y el número de veces que hay que hacerlo[5].

[19] Se puede consultar más información acerca de OpenGL ES en: `http://www.khronos.org`.

El emulador Android imita la mayoría de las funciones del dispositivo real, pero tiene sus limitaciones en cuanto a las conexiones USB, la captura de video o imágenes con la cámara, los audífonos, la simulación de la batería, el bluetooth entre otras. En caso de implementar estas funciones es encesario tener el dispositivo para hacer las pruebas directamente.

El procesador que se utiliza en el emulador está basado en un ARM(Advanced RISC Machine) que es un microprocesador de 32-bits basado en arquitectura RISC(Reduced Instruction Set Computer). Los procesadores ARM son ampliamente utilizados en los dispositivos móviles y otros sistemas embebidos donde el bajo consumo de energía es importante. Gran parte de los dispositivos en el mercado de telefonía móvil utilizan procesadores basados en esta arquitectura además de dispositivos como el iPod, Nintendo DS y Game Boy Advance que se ejecutan en procesador ARM versión 4[23].

Hay muchos tipos de dispositivos con diferentes características. Por esta razón es necesario emular los diferentes tipos de dispositivos y permitir que puedan adaptarse a distintas configuraciones en un entorno emulado. El emulador tiene diferentes skin[20] que representan diversos dispositivos así como orientaciones vertical y horizontal. Cuenta con un teclado QWERTY[21] que puede estar visible u oculto.

La simulación de la velocidad de la red es un elemento clave para el desarrollo de aplicaciones para dispositivos móviles. Esta característica es útil porque la experiencia de los usuarios puede variar durante su uso en el dispositivo real. Es importante que las aplicaciones se adapten en la ausencia de una conexión de red. El emulador prevé un amplio conjunto de herramientas para probar diferentes condiciones de la red y su velocidad.

Se debe tener en cuenta que puede haber cargos por el uso de conexión a la red y por esto la importancia de tener un emulador que nos permita hacer pruebas de conectividad y velocidad de la red sin tener que hacer esas pruebas en un dispositivo real y pagar los costos que se generan por los servicios de transmisión de datos[5].

Para utilizar el emulador es necesario crear configuraciones *AVD(Android Virtual Devices)*. En estas configuraciones se indica una plataforma Android a ejecutar en el emulador y un conjunto de opciones de hardware además del skin del emulador que se desea utilizar.

Cuando se carga el emulador se debe especificar la configuración que se desea utilizar. Android proporciona herramientas para crear estas configuraciones y para el manejo del emulador a través de la línea de comandos. Si se utiliza un IDE como Eclipse, el plug-in permite realizar estas configuraciones de un modo gráfico. Se pueden encontrar más detalles acerca del emulador en la página de desarrollo de Android[22].

2.3.2.2. La interfaz de usuario

Android utiliza un framework similar al utilizado en otros frameworks para computadoras de escritorio. El Android UI[23] framework es similar a JavaFX, Microsoft Silverligth y Mozilla

[20] Se refiere a la apariencias que muestra el dispositivo.
[21] Los teclados QWERTY son los teclados que comúnmente se utilizan en la mayoría de las computadoras.
[22] `http://developer.android.com/guide/developing/tools/emulator.html`.
[23] UI, User Interface o Interfaz de Usuario.

XML User Interface (XUL) que son los frameworks UI de cuarta generación[24], en los que la interfaz de usuario es declarativa y de forma independiente del resto de la aplicación.

La programación en Android UI consiste en declarar la interfaz en archivos XML. Después se cargan las definiciones de vistas XML como las ventanas de la aplicación UI. También los menús se pueden cargar desde archivos XML. Las pantallas o ventanas en Android son conocidas como actividades que comprenden múltiples vistas que un usuario necesita para completar una unidad lógica de acción. Las vistas son bloques básicos para la construcción de la interfaz de usuario y se pueden combinar para formar vistas compuestas llamadas "grupos de vistas". Una actividad contiene éstas vistas compuestas, que incluyen vistas y grupos de vistas. Esta es la lógica de componentes de UI en Android[4].

2.3.2.3. Componentes fundamentales

Cada framework de aplicación tiene algunos componentes clave que los desarrolladores necesitan entender antes de empezar a escribir aplicaciones basadas en el framework. Por ejemplo, es necesario entender los JSP y Servlets para escribir aplicaciones en J2EE[25]. Del mismo modo, se necesitan comprender los siguientes conceptos para desarrollar aplicaciones Android: actividades, vistas, intenciones, proveedores de contenido, servicios y el archivo manifiesto para comenzar con el desarrollo en Android. Por lo que examinamos de manera breve estos conceptos a continuación.

- **View**, las vistas son los bloques básicos de construcción de una interfaz de usuario. Las vistas son jerárquicas.
- **Activity**, una actividad por lo general representa una pantalla de la aplicación y puede contener una o más vistas, aunque también puede no tenerlas.
- **Intent**, generalmente define una "intención" de hacer algún trabajo. Un intent es una descripción abstracta de una operación que puede ser realizada[24]. Los intents encapsulan varios conceptos, por lo que la mejor manera de comprenderlos es ver ejemplos de su uso. Se puede usar los intents para realizar las siguientes tareas, por ejemplo:

  1. Emitir un mensaje.
  2. Iniciar un servicio.
  3. Carga de una actividad.
  4. Mostrar una página web o una lista de contactos.
  5. Marcar un número telefónico o contestar una llamada telefónica.

  Un intent se puede ver como un framework de paso de mensajes. Usando intents, se pueden transmitir mensajes del sistema a una actividad o servicio objetivo, indicando tu intención de tener una acción para realizar. El sistema entonces determinará el objetivo que llevará a cabo las acciones según corresponda.

[24] Frameworks basados en XUL, lenguaje basado en XML para la interfaz de usuario.

[25] J2EE, Java 2 Enterprise Edition, es una especificación para aplicaciones empresariales, que se ejecutan en un servidor de aplicaciones.

- **Content Provider**, el intercambio de datos en las aplicaciones dentro de un dispositivo es muy común, por eso Android define un mecanismo estándar para que las aplicaciones compartan datos(como la lista de contactos) a través de los proveedores de contenido, se pueden exponer sus datos y sus aplicaciones pueden utilizar datos de otras aplicaciones.

- **Service**, se asemejan aquéllos que pueden observarse en el sistema operativo Windows u otras plataformas. Son procesos que pueden funcionar durante un largo periodo de tiempo sin interactuar con el usuario. Android define dos tipos de servicio: locales y remotos. Los servicios locales son componentes que solo son accesibles por la aplicación que aloja el servicio. Los servicios remotos son servicios a distancia que están destinados para acceder a ellos remotamente por otras aplicaciones. Se pueden utilizar servicios existentes o también se pueden crear nuevos servicios.

- **AndroidManifest.xml**, es un archivo similar al archivo de configuración web.xml en J2EE. Define el contenido y el comportamiento de la aplicación. Por ejemplo, se definen las actividades del programa así como los servicios y los permisos que la aplicación necesita para ejecutarse.

2.3.3. Estructura de aplicaciones Android

El tamaño y la complejidad de las aplicaciones Android puede variar mucho, pero la estructura principal es siempre similar. Las aplicaciones Android tienen algunos elementos que son requeridos y otros no. En la siguiente tabla se resumen los elementos que conforman la estructura principal de la aplicación.

Elemento	Descripción	Requerido
AndroidManifest.xml	Archivo descriptor de aplicación de Android. En este archivo se definen las actividades, provedores de contenido, servicios, permisos, etc.	Sí
src/	Un directorio que contiene todo el código fuente de la aplicación	Sí
assets/	Es una colección arbitraria de directorios y archivos	No
res/	Es un directorio que contiene todos los recursos de la aplicación. Contiene otros directorios	Sí
drawable/	Un directorio que contiene imágenes usadas por la aplicación	No
layout/	Un directorio que contiene las vistas de la aplicación	No
values/	Es un directorio que define recursos en descriptores xml,por ejemplo, strings, estilos y colores	No

Cuadro 2.1: Elementos de Aplicación

Una aplicación Android está compuesta principalmente por tres partes: el descriptor de la aplicación, una colección de diversos recursos y el código fuente. El lenguaje XML es muy utilizado por Android, los archivos son compilados utilizando el Android Asset Packing Tool(AAPT). Cuando la aplicación se instala en el dispositivo, se almacena de forma binaria. Cuando el archivo se lee durante el proceso de ejecución se hace en su forma binaria y no se

transforma de nuevo en XML. Esto da la ventaja de trabajar de manera sencilla en XML y no tener que preocuparse de tomar los valiosos recursos del dispositivo[4].

2.4. Servidor de búsquedas Solr

Para realizar las búsquedas en una base de datos externa a nuestro programa en Android vamos a utilizar un servidor de búsquedas llamado *Solr*.

Solr es un servidor de búsquedas de código abierto. Está basado en una biblioteca escrita Java para realizar búsquedas llamada Lucene e incluye API's para el manejo del servidor como cliente java llamado SolrJ. Además tiene otras funciones, por ejemplo resaltar las palabras, replicación, facets y otras características que se mencionan más adelante en esta sección[25].

2.4.1. Lucene

El proyecto *Lucene* está soportado por Apache Software Foundation desde el 2001. Fue desarrollado en un principio por Doug Cutting en el año 2000. Es una biblioteca escalable para Recuperación de Información. Es adecuada para aplicaciones que requieren la búsqueda a texto completo y con soporte para múltiples plataformas[59].

En ocasiones se confunde a Lucene con una aplicación lista para usarse como lo son los programas de búsqueda de archivos, los crawlers o los motores de búsqueda web, y no es así. Lucene se puede utilizar, por ejemplo, si se quiere implementar un motor de búsquedas, ya que es un conjunto de instrumentos para la aplicación, pero no todos.

Se puede pensar en Lucene como una capa que permite de manera sencilla, capacidades de indexación y búsqueda a partir de diferentes fuentes de información, siempre y cuando los datos puedan ser convertidos a un formato de texto[26].

2.4.1.1. Conceptos básicos

Existen conceptos básicos que se deben conocer para comenzar el trabajo con Lucene, a continuación se definen los principales:

- **Campo** – Es un par nombre-valor. Contiene meta-datos que describen el contenido.
- **Documento** – Un documento es una colección de Campos.
- **Índice** – En Lucene, un índice es una colección de documentos. Es una estructura de datos que permite el acceso aleatorio rápido a palabras almacenadas en su interior.
- **Query** – Es una forma de preguntar dentro de una base de datos. Lucene tiene su propio lenguaje para realizar las consultas. Este lenguaje permite indicar sobre que campo se debe buscar, además tiene la habilidad para especificar operadores booleanos como OR, AND y NOT. Esto permite incluir un criterio de búsqueda más particular.
- **Indexación** – Es el proceso de preparar la información, si es necesario darle un formato textual y agregar la entrada de esos datos ya como documentos a un índice generado por Lucene, de manera que facilite una búsqueda rápida.

- **Búsqueda** – Se requiere un índice ya construido para realizar la consulta a través de un query. Existen diferentes tipos de consulta, algunos pueden ser por palabra clave, frase, comodines y algunos otros. Después de realizar este proceso de búsqueda en el índice, se regresa una lista de resultados que coinciden con los términos de la búsqueda.

2.4.1.2. Análisis de texto

En Lucene "el análisis de texto es el proceso de conversión de un campo de texto en la representación más fundamental indexada, llamada término"[7]. Durante el proceso de búsqueda estos términos se utilizan para encontrar los documentos que los contienen.

Podemos decir que un analizador es una encapsulación del proceso de análisis. Esto significa que el analizador separa el texto en términos para poder realizar diversas operaciones sobre él, por ejemplo, se pueden extraer palabras, descartar signos de puntuación y palabras, eliminar acentos en los caractéres, convertir en minúsculas, eliminar palabras comunes. A este proceso también se le conoce con el nombre de *tokenización* y a los trozos de texto que resultan de la aplicación de las operaciones anteriores se les llama *tokens*[18].

Elegir un analizador es importante al iniciar un desarrollo con Lucene. Uno de los factores para elegir un analizador es el lenguaje, porque cada uno tiene características propias que se deben considerar. Otro factor es el dominio del texto, ya que existen diferentes industrias que manejan diferentes terminologías como abreviaciones, acrónimos y otros. Sin embargo, no todas las soluciones que ofrece Lucene serán de utilidad para las diferentes situaciones. Por eso en Lucene existe la posibilidad de implementar una solución personalizada, un analizador adecuado para cada necesidad[26].

2.4.1.3. Integración con Lucene

En la figura2.2 podemos ver una manera de integrar Lucene con una aplicación de búsqueda muy simple.

En la figura se ve como a través de los dos servicios principales: recolección de datos e índice de búsqueda, podemos tener una aplicación básica. Utilizamos Lucene como medio de indexación de documentos que pueden obtenerse a través de diversas fuentes, como pueden ser páginas web que están en servidores remotos, documentos almacenados en el sistema de archivos en una máquina local, archivos de diferentes formatos como HTML, doc, PDF o cualquier formato del cual se pueda extraer información textual.

Además, Luce permite indexar información almacenada en una base de datos lo cual incluye la capacidad de hacer búsquedas del tipo texto completo. Esta característica no siempre está incluida en los manejadores de bases de datos[7].

Con el contenido indexado, es necesario obtener del usuario las palabras o frases que quiere buscar. Con estas palabras se revisa el índice de búsqueda y se obtiene la información que necesita el usuario. Posteriormente por medio de alguna interfaz se muestra la lista de resultados que coinciden con su consulta.

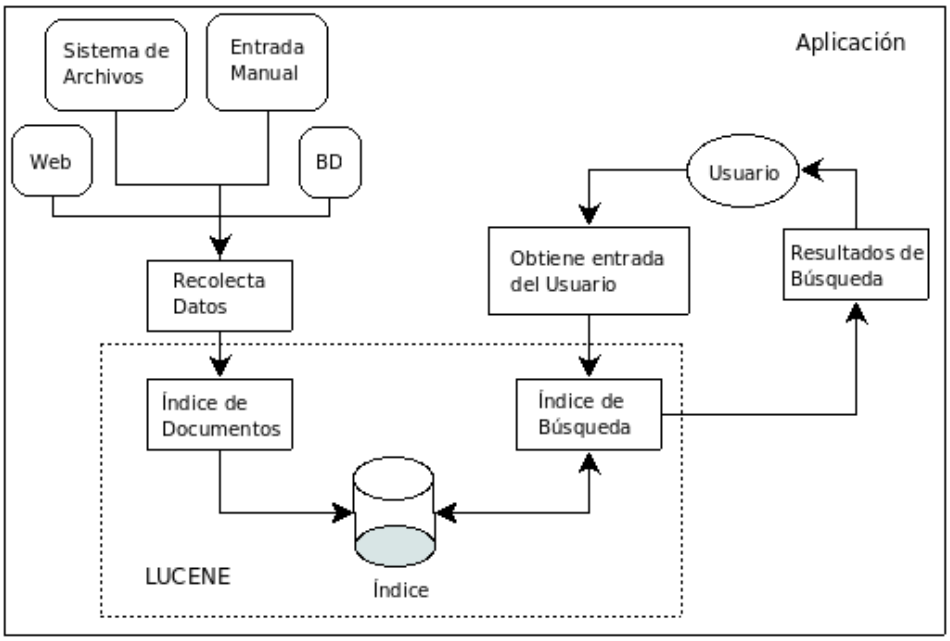

Figura 2.2: Aplicación básica con Lucene

2.4.1.4. Clases principales

Para el desarrollo con Lucene es necesario conocer las clases básicas para indexar y buscar la información. Para conocer más a detalle los métodos de cada clase se puede consultar el API de Lucene a través de su página oficial[27].

- **IndexWriter**, esta clase provee el funcionamiento principal en el proceso de indexación. Se utiliza para crear un índice y agregar documentos a un índice existente. Esta clase no es la única que puede modificar un índice. Se pueden tener otras alternativas explorando el API.

- La clase **Directory** se utiliza para representar la localización de un índice, por ejemplo se tiene una subclase llamada **FSDirectory** que permite almacenar un índice en el sistema de archivos. También se puede utilizar otra implementación de la clase **Directory** llamada **RAMDirectory** que almacena los datos en memoria. Esto es útil para índices pequeños y que necesitan borrar la información del índice al terminar la ejecución de la aplicación.

- Un objeto de la clase **Analyzer** debe especificarse en el constructor de la clase **IndexWriter**. Se utiliza un analizador antes de que el texto sea indexado. Es la clase encargada de extraer los tokens y descartar el resto de la información. Si el contenido no está en formato textual, es necesario darle formato antes de ser analizado. Esta es una clase abstracta pero Lucene tiene algunas implementaciones por ejemplo **WhitespaceAnalyzer**, **SimpleAnalyzer**, **StopAnalyzer**, entre otras.

- La clase **Document** es la representación de un documento que, como se mencionó, es una colección de campos que representan el contenido del documento o meta-datos asociados con ese documento. Para Lucene es irrelevante la fuente de datos de donde se extrae la información que compone el documento. Los meta-datos pueden contener información como: autor, tema, fecha de modificación y algunos otros.

- La clase **Field** define una representación para cada par nombre-valor. Dentro de Lucene se utiliza para almacenar los valores de cada campo de un documento. Se pueden utilizar 4 diferentes tipos de campo: Keyword, UnIndexed, UnStored y Text.

La interfaz de búsqueda básica de Lucene se compone de 5 clases principales, se utilizan para desempeñar operaciones relacionadas con el proceso de búsqueda. Las clases son:

- La clase **IndexSearcher** se utiliza para realizar las funciones de búsqueda. Es una clase que abre el índice en modo solo lectura y ofrece diferentes funciones para buscar los datos.
- La clase **Term** representa la unidad básica que se requiere para realizar una búsqueda. Se pueden utilizar los objetos **Term** a través de la implementación **TermQuery**.
- **Query** es una clase abstracta y es padre de un conjunto de clases para realizar búsquedas, la más básica es **TermQuery**. Además de **BooleanQuery**, **PhraseQuery**, **PrefixQuery**, **PhrasePrefixQuery**, **RangeQuery**, **FilteredQuery** y **SpanQuery**.
- La clase **TermQuery** permite utilizar el tipo de consulta más básica en Lucene.
- La clase **Hits** es un contenedor para referencias hacia las coincidencias que hayan resultado de realizar la consulta en el proceso de búsqueda.

2.4.2. Solr

Un uso común de Solr es el siguiente: si se tiene un conjunto de datos almacenados en una base de datos, Solr puede acceder a esta información para mostrarla a través de una página web, por ejemplo.

La tecnología que conforma el núcleo principal de Solr es *Lucene*. Como mencionamos anteriormente, es una biblioteca para recuperación de información de alto rendimiento. A continuación se muestra una lista de características de Lucene que son más utilizadas en Solr[8]:

1. Un índice invertido para almacenamiento persistente basado en texto para realizar de manera eficiente la recuperación de documentos por términos indexados.
2. Un conjunto de analizadores de texto para transformar una cadena de texto en una serie de términos(palabras) que pueden ser indexados y buscados, es la unidad más fundamental en el proceso de indexación.
3. Ofrece una sintáxis de consulta con la que se pueden realizar diferentes tipos de búsqueda como una búsqueda con términos simples, búsquedas por aproximación y otras más.
4. Algoritmo de puntuación para clasificar los primeros candidatos más probables para el resultado.
5. Una función para resaltar las palabras encontradas en el resultado.

Con la revisión anterior de la biblioteca para la recuperación de datos se puede describir a Solr como la "puesta en servidor" de Lucene[8]. La mayoría de características de Solr son diferentes a las de Lucene, por ejemplo el proceso de *Faceting*[26]. A continuación se presenta el conjunto de características diferentes de Solr con respecto a Lucene.

[26]Es una técnica de búsqueda en la que se clasifican los resultados permitiendo crear filtros.

1. Procesamiento de solicitudes por HTTP para la indexación y consulta de documentos.
2. Gran cantidad de Cache[27] para respuestas rápidas a consultas.
3. Una interfaz de administración basada en web que incluye lo siguiente:
   a) Estadísticas de desempeño en tiempo de ejecución.
   b) Un formulario de consulta para realizar búsquedas en el índice.
   c) Un navegador de esquemas con algunas estadísticas.
   d) Desglose detallado de fases de análisis de texto.
4. Archivos de configuración para el servidor y para los archivos de esquema(en XML).
   a) Solr se suma a la biblioteca de análisis de Lucene y permite la configuración a través de archivos XML.
   b) Introduce la noción de un tipo de campo. Los tipos están presentes en las fechas y son importantes en referencia al ordenamiento y clasificación de resultados.
5. Faceting para los resultados de las consultas.
6. Un plug-in de corrector ortográfico usado para hacer una consulta alternativa por ejemplo "te refieres a " [28].

2.4.2.1. El directorio *solr/home*

Una vez instalado el servidor Solr, es necesario tener un directorio de inicio. En este directorio es donde se encuentran los datos y la configuración para ejecutar una instancia de Solr. El directorio se llama **Solr home** y se puede identificar como el directorio de inicio porque contiene el archivo *solr.xml* o contiene los directorios **conf** y **data**[8].

El directorio *solr/home/* por lo general tiene la siguiente estructura:

1. **bin** – Este directorio se sugiere para colocar scripts de replicación si se tiene una configuración más compleja.
2. **conf** – En este directorio se encuentran los archivos de configuración. Los dos más importantes son los que se mencionan a continuación. Sin embargo, existen otros archivos que sirven para el proceso de análisis de texto.
3. **conf/schema.xml** – Este archivo define el esquema para el índice además incluye las definiciones de los tipos de campo para el proceso de anánilis.
4. **conf/solrconfig.xml** – Este es el archivo prncipal en la configuración de Solr.
5. **conf/xslt** – Este directorio contiene varios archivos xslt. Son usados para transformar las respuestas XML de Solr a formatos como RSS[29].

[27] Es información duplicada que se encuentra en memoria principal y su acceso es más rápido.
[28] Un corrector como las sugerencias de Google, cuando el usuario se equivoca generalmente al escribir una palabra.
[29] RSS(Really Simple Syndication) es una familia de formatos web codificados en XML, se utiliza para la redifusión web.

6. **data** – Contiene el índice actual Lucene. Son datos binarios, así que no se puede trabajar directamente con el contenido de este directorio.

7. **lib** – Es un lugar opcional para colocar archivos jar[30] extras.

Existen diferentes maneras de iniciar el servidor Solr indicando la configuración que se quiera utilizar. Una forma es por medio de una propiedad en el ambiente Java. Se define una propiedad llamada solr.solr.home y con esta propiedad al iniciar el servidor de aplicación se inicia con el **Solr home** que se especificó.

Utilizando el servidor de servlets Jetty[31] se define el directorio **Solr home** como sigue:

```
java -D solr.solr.home = MiSolr/ -jar start.jar
```

Otra manera es indicar el directorio en el archivo *web.xml* del archivo *war* de Solr. Este archivo está localizado en el directorio: *src/web-app/web/WEB-INF*.

```
<env-entry>
  <env-entry-name>solr/home</env-entry-name>
  <env-entry-value>MiSolr/</env-entry-value>
  <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

De esta manera al iniciar el servidor se tiene la configuración que se define en el directorio *solr/home/*. En los siguientes capítulos se muestra parte de la configuración del servidor Solr para la aplicación de ejemplo y los archivos de configuración completos en el Apéndice A.

2.4.2.2. Importación de datos

Utilizando *dataImportHandler(DIH)* se pude importar información desde una base de datos vía JDBC, sistema de archivos, RSS feed y otras fuentes de datos. DIH no es parte de Solr directamente, sin embargo, se distrubuye a través del archivo *apache–solr–dataimporthandler–1.4.jar*, el cual puede encontrarse ya incluído en el archivo *solr.war* o también puede encontrarse en el directorio lib de la instalación de Solr.

Conceptualmente el DIH se puede dividir en las siguientes partes[52]:

1. **DataSource** – Es la base de datos, RSS feed, archivo XML o la fuente de datos de donde se obtiene la información.

2. **Declaraciones Document/entity** – Especifica el mapeo entre el contenido del DataSource y el Solr Schema.

3. **EntityProcesor** – Es el código responsable del mapeo. Solr tiene cuatro implementaciones: FileListEntityProcessor, SqlEntityProcessor, CachedSqlEntityProcessor, XPathEntityProcessor.

[30] Un archivo jar permite ejecutar aplicaciones Java, así como comprimir y descomprimir las clases.
[31] Jetty es un servidor HTTP y un contenedor de servlet's escrito en java.

4. **Transformer** – Es un código definido por el usuario para transformar el contenido importado antes de agregarlo a Solr.

El DIH ofrece las siguientes funciones[8]:

1. *Importar datos desde una base de datos por medio de JDBC.*
2. *Importar datos desde archivos XML a través de URL(HTTP GET) o desde archivo.*
3. *Se puede combinar datos de diferentes tablas.*
4. *Extracción y transformación de datos.*
5. *Importar actualizaciones de datos suponiendo una última fecha de actuailzación.*

Para registrar el DIH con Solr es necesario hacerlo en el archivo **solrconfig.xml**. Hay diferentes formas de hacerlo según sea la fuente de datos. A continuación se muestra un ejemplo utilizando como fuente datos una base de datos en PostgreSQL:

```
<dataSource name="ratings" driver="org.postgresql.Driver"
 url="jdbc:postgresql://localhost:5432/solr_dw" user="solr_dw" />
```

2.4.2.3. Comparación con bases de datos

Una base de datos y un índice de búsqueda Lucene no son muy diferentes conceptualmente. A continuación se describe el índice de búsqueda y se mostrarán las diferencias con los conceptos básicos de una base de datos de manera breve. Para esto es necesario revisar la diferencia que existe entre utilizar un índice Lucene y una base de datos que no contiene la posibilidad de características de indexación de texto y búsqueda de texto completo.

La gran diferencia que hay entre un índice Lucene y una base de datos es que el índice en Lucene es solamente una tabla, a diferencia de las bases de datos que por lo general se componen de más de una tabla para almacenar los datos. La tabla que representa el índice no tiene soporte para consultas relacionales(operaciones JOIN). Hay que recordar que el índice solo es una herramienta para buscar la información y no es la fuente principal de los datos[7].

Otras diferencias son :

1. *Actualizaciones.* Documentos completos se pueden agregar o quitar pero no se actualizan en el índice.
2. *Búsqueda con subcadena contra Búsqueda con texto.* Usando una base de datos, la peor consulta utilizando la búsqueda por subcadena podría ser algo como “Select * from MyTable where name LIKE ’Books”’, esta consulta podría coincidir con frases como “My Books” o “CookBooks”. Lucene busca principalmente sobre los términos(palabras). Dependiendo de la configuración del análisis, esto puede significar que varias formas de la palabra pueden ser encontradas también, por ejemplo “book”, el singular.
3. *Scored Results.* Gran parte de la potencia de Lucene es la capacidad de calificar los documentos de acuerdo a que tan bien coinciden con el criterio de búsqueda. Por ejemplo si en una búsqueda se utiliza un operador booleano como OR para buscar varios términos

Lucene calificará con mayor puntuación a los documentos que coincidan con más términos y una calificación menor para los que coincidan solo en uno de sus términos. En una base de datos no se tiene un concepto como este, un registro coincide o no.

4. *Slow commits*. Solr está altamente optimizado para la velocidad de búsqueda. La velocidad es debido en gran parte a los cachés.

2.5. Resumen

En este capítulo se presentan las diferentes tecnologías que se utilizarán en los siguientes capítulos. Se mencionó el concepto de dispositivo móvil así como los tipos de aplicaciones que existen y las herramientas que se necesitan para desarrollarlas. Se revisan algunos conceptos sobre la recuperación de la información y un tipo de búsqueda particular llamada Búsqueda de Texto Completo.

También se da una introducción a la plataforma Android, se revisa el entorno de software y el sistema operativo y se mencionan algunas de las bibliotecas principales además de algunas características del emulador, el framework para el desarrollo de la interfaz de usuario y la estructura básica de una aplicación Android.

Se revisa el servidor de búsquedas Solr. Se muestra la base de Solr llamada Lucene, se mencionan algunas de las clases principales y las funciones de Lucene que utiliza Solr y también se mencionan algunas diferencias entre los dos componentes. Por último se señalan las diferencias entre el uso de Solr y las bases de datos.

Capítulo 3

Trabajo relacionado

En este capítulo se revisan los trabajos relacionados con la recuperación de la información y los motores de búsqueda. Se revisan los diferentes tipos de búsqueda que se utilizan en los disositivos móviles y también se mencionan los principales motores de búsqueda en dispositivos móviles. Una parte importante de este trabajo es la manera de accesar a la información desde un dispositivo móvil. En la sección 3.2 se mencionan cuatro formas de accesar a los datos desde un dispositivo móvil.

Se analizan diferentes opciones para almacenar los datos en el dispositivo. En el caso de las bases de datos la mayoría de los sistemas son comerciales, aunque se mencionan dos opciones con licencia de uso libre.

La manera de obtener los datos desde un dispositivo móvil es importante ya que es la función básica de una aplicación de recuperación de información. Por eso es necesario conocer los requerimientos y las funciones que se necesitan realizar en la aplicación para elegir un método de recuperación de datos adecuado.

3.1. Motores de búsqueda web

Un motor de búsqueda web es un servicio web que permite realizar una búsqueda en Internet y devuelve los resultados en diferentes formatos. Las búsquedas se realizan por medio de palabras, frases y operadores. Los resultados son generalmente páginas web, aunque también pueden ser documentos, imágenes, videos, audio y otros tipos de contenido.

Existen varios motores de búsqueda web. Algunos de los más utilizados son los siguientes:

- Google[1] – Es uno de los buscadores más utilizados de Internet. Una de sus principales características es que utiliza un algoritmo para la clasificación y el posicionamiento de resultados llamado **PageRank**[2]. Tiene una forma para que los usuarios puedan agregar un sitio web al índice del motor de búsqueda[28].

- Yahoo[3] – El servicio de búsqueda se llama **Yahoo Search**. Se puede agregar un sitio web al índice y también sitios especialmente para dispositivos móviles. Tiene una interfaz en

[1] `http://www.google.com`
[2] `http://www.google.com/intl/es/corporate/tech.html`
[3] `http://mx.yahoo.com/`

varios idiomas y mantiene un directorio con enlaces organizados por categorías[29].

- Bing[4] – Inició en enero del 2009 como reemplazo a otros buscadores de la empresa Microsoft como **Live Search**. Incluye un panel de sugerencias a las búsquedas realizadas y también tiene la posibilidad de compartir el historial de búsquedas a través de correo electrónico o de servicios como Facebook[30].

Los componentes principales de estas aplicaciones son los que se muestran a continuación[53]:

- **Interfaz Gráfica** – La Interfaz gráfica es generalmente una página web que se muestra a través de un navegador. Aunque existen otros tipos de interfaz, todos los motores de búsqueda utilizan una página para realizar las búsquedas. La interfaz se compone de una caja de texto donde se escriben las palabras para la búsqueda y otras opciones de personalización como el idioma o tipo de contenido a buscar.

  Algunas interfaces muestran la información organizada por categorías y subcategorías. Este tipo de interfaz es utilizada por los motores de búsqueda que son del tipo Directorio.

- **Índice** – Un índice es una estructura de datos en donde se almacenan los términos de búsquedas utilizados por el motor para encontrar los diferentes documentos. A diferencia de una base de datos el índice permite una mayor velocidad en operaciones de búsqueda sobre los datos. Existen diferentes métodos de indexación, por ejemplo, puede ser manual o de manera automática por medio de diferentes algoritmos.

- **Robot** – Un Robot web o Web bot es un programa que realiza las siguientes funciones:

  - Análisis Estadístico.
  - Verificación del estado de los enlaces.
  - Reúne informaciíon para construir el índice.

  Los robots web también son conocidos por las tareas más específicas que realizan. Entre los más utilizados están los Spiders, Knowbots, Wanderers, Webcrawlers, Worms y WebAnts.

Los motores de búsqueda web tienen algunas características en común. Para interactuar con el motor es necesario tener un navegador web. Las interfaces son páginas que deben ser escritas en lenguaje HTML o alguna variante. La resolución de los elementos que componen la interfaz están diseñados para visualizarse en pantallas amplias como las de las computadoras personales o de laptop. Las búsquedas se realizan en un índice que es construido por la aplicación del motor, es decir, se reúne la información de Internet y se genera el índice sobre el cual se va a buscar.

Los resultados principales son páginas web y elementos multimedia que también están disponibles en esas páginas. Estos motores no permiten una búsqueda en bases de datos propias o en información almacenada en un dispositivo.

[4] `http://www.bing.com/`

3.1.1. Búsqueda en dispositivos móviles

Los motores de búsqueda web también tienen una versión para dispositivos móviles. Es una interfaz que permite a los usuarios de teléfonos celulares y otros dispostivos móviles, acceder al motor de búsqueda conectandose a través de una página wap. El protocolo de aplicaciones inalámbricas (WAP) es un estándar para la comunicación en las aplicaciones para dispositivos móviles.

Algunos de los buscadores para dispositivos móviles se han especializado en buscar contenido para dispositivos móviles, por ejemplo, tonos, juegos, videos, audio, imágenes, aplicaciones específicas. En la actualidad el contenido puede presentarse en una interfaz que no ha sido diseñada específicamente para un dispositivo móvil. Esto hace que sea difícil de encontrar desde un teléfono donde el navegador no soporta este tipo de portales que contienen la información para dispositivos móviles[32].

Existen diferentes tipos de búsqueda en dispositivos móviles. Entre los más utilizados están los siguientes[31]:

- **Motores de Búsqueda Optimizados** – Son optimizados para dispositivos móviles. Son versiones diseñadas especialmente para estos dispositivos basados en los motores de búsqueda web. Algunos ejemplos son el buscador Google o la versión para dispositivos móviles de Bing. Los buscadores, al igual que en sus versiones web, se utilizan para buscar páginas wap o contenido para dispositivos móviles en sitios web.
- **Búsqueda por mensajes de texto** – A través de este servicio el usuario puede realizar diversas búsquedas sobre contenido, imágenes, videos y más, enviando un mensaje de texto SMS[5] a una central de datos. Una vez enviando el mensaje con una palabre clave y la consulta se recibe una respuesta en SMS con los resultados de la consulta. Un ejemplo de este servicio lo ofrece Google en su sección de aplicaciones y servicios para dispositivos móviles.[6]
- **Búsqueda en Directorio** – También es llmada búsqueda local. Es útil para realizar búsquedas sobre servicios en el lugar donde la persona se encuentra. Existen servicios para buscar comercios, servicios de taxi, o servicios locales que estén próximos al usuario que realiza la consulta. Son búsquedas especiales donde es necesario saber la ubicación del dispositivo. Es una aplicación especializada de los sistemas de localización.

Las características en general de la búsqueda en dispositivos móviles son las siguientes:

- El dispositivo necesita un navegador WAP.
- Las búsquedas se realizan en directorios manejados por el operador del servicio o de la aplicación.
- Los principales buscadores están enfocados en la búsqueda en Internet. Se pueden buscar sitios web o específicamente para dispositivos móviles.
- En la búsqueda para dispositivos móviles es importante el espacio, así que solo se muestra el contenido principal de una página o de un enlace. Esto hace más difícil la comparación entre los resultados por parte del usuario.

[5] Mensajes de Texto Corto, se utilizan en los teléfonos celulares

[6] `http://www.google.com/intl/es_ALL/mobile/default/search.html`

- La búsqueda en dispositivos móviles por lo regular está diseñada para buscar sitios para dispositivos móviles, es complicado encontrar sitios web que puedan visualizarse correctamente en estos dispositivos.

3.1.2. Los principales motores de búsqueda en dispositivos móviles

Los motores de búsqueda para dispositivos móviles más utilizados actualmente son:

- Google Mobile[7] – Este buscador es parte de una serie de servicios que se ofrecen en Internet por parte de Google para los dispositivos móviles. Tiene diferentes páginas de acceso para los diferentes tipos de especificaciones que existen en la web para dispositivos móviles como WAP 1.0/2.0 y i-mode[8] que es la especificación más utilizada en Japón. Se pueden buscar imágenes o solo páginas diseñadas para dispositivos móviles. También búsqueda de información local. Además se puede utilizar a través de mensajes de texto SMS[33].

- Yahoo Mobile[9] – El buscador de Yahoo para dispositivos móviles ofrece funciones como: enfoque a las necesidades del usuario de dispositivo móvil, esto es, respuestas en lugar de enlaces web, reconoce la ubicación del dispositivo, búsqueda de información local. Ofrece una aplicacion de búsqueda para instalarse en el dispositivo, tiene funciones para búsqueda por voz, búsqueda instantánea, asistente de búsqueda, localización automática, solo está disponible para muy pocos dispositivos y algunas plataformas [34].

- Bing Mobile[10] – Después de la versión para web del motor de búsquedas Bing, apareció la versión para dispositivos móviles. Entre las funciones que tiene son, por ejemplo, la búsqueda por localización como lo hace la aplicación de Google Maps, contiene una sección donde se pueden realizar búsquedas por categoría[35]. Además incluye búsquedas de información local, como vuelos, compras, servicios médicos entre otros[36]. Una de las características de la aplicación de Bing para Windows Mobile es que permite funciones para los teléfonos con pantalla táctil, conocidos como *touchscreen*.

Estos motores de búsqueda para dispositivos móviles trabajan con búsquedas sobre sus propias bases de datos o directorios. No es posible buscar información en una base de datos externa a sus sistemas de información.

3.2. Acceso a la información desde dispositivos móviles

Existen diferentes formas en las que se puede acceder a la información desde un dispositivo móvil, por ejemplo:

- Información en el dispositivo por medio de registros.

- Sincronización con un servidor de bases de datos en una computadora personal.

- Una base de datos dentro del dispositivo.

[7] `m.google.com/search`

[8] i-mode se refiere a un conjunto de tecnologías que permiten acceder a páginas a través de una red para dispositivos móviles, se utiliza en Japón principalmente.

[9] `m.yahoo.com/search`

[10] `m.bing.com/`

- Una base de datos a través de un servicio o página web.

Generalmente los motores de búsqueda en dispositivos móviles comerciales utilizan la misma información que la versión para web, es decir, tienen la misma fuente de datos, y la aplicación principal, el motor, está basado en el mismo que utilizan las aplicaciónes de búsqueda para computadoras personales.

3.2.1. Información almacenada en registros

A través de una aplicación en Java se puede tener un sistema simple de manejo de información dentro del dispositivo móvil. En la especificación J2ME de Java se puede encontrar RMS (Record Management System). Esto es un sistema de manejo de información. Dentro del sistema el elemento básico es el *Record* y *RecordStore* es una colección de registros.

Para trabajar con este sistema es necesario importar el paquete *javax.microedition.rms* dentro del midlet[11]. Existen algunos puntos importantes que se deben tomar en cuenta cuando se utiliza un sistema de almacenamiento de este tipo. A continuación se mencionan algunos de los más importantes[37]:

- **Límite de almacenamiento** – El espacio varía de un dispositivo a otro, sin embargo, en la especificación MIDP [12] se exige 8K para el almacenamiento de datos persistentes. La memoria persistente en un dispositivo es compartida.
- **Velocidad** – Las operaciones de lectura y escritura pueden tardar más que en los datos volátiles en memoria. Además no se recomienda realizar estas operaciones dentro de algún evento thread dentro del midlet.
- **Registros** – Un registro puede contener cualquier cosa que una secuencia de bytes pueda representar, por ejemplo un número, una cadena, una matriz o una imágen. La responsabilidad de interpretar el contenido del registro es exclusivamente de la aplicación que lo utiliza. RMS proporciona solamente el almacenamiento y un identificador único.

Existen diferentes funciones que se pueden realizar con los *Record* y *Record Stores*, entre otras, están la lectura y escritura de registros, la actualización, la eliminación, además la clase *RecordStore* permite obtener el tiempo de la última modificación, el nombre del registro, el número de registros en el record store y el tamaño total en bytes del *RecordStore*.

En cuanto a la búsqueda de información en el sistema de registros es necesario implementar los métodos encargados de realizar las búsqueda. Esto permite realizar el proceso de forma específica para cada aplicación. Sin embargo es una desventaja no tener un método que ya proporcione esta funcionalidad, por lo que es necesario implementar todo el sistema de búsquedas. Una de las clases importantes para realizar la búsqueda en registros es **RecordFilter**. Esta clase permite realizar una búsqeda con un criterio a través del método *matches()*.

Este sistema de registros que maneja Java es un ejemplo de sistemas de almacenamiento persistente en los dispositivos. Un ejemplo más puede ser el sistema de archivos no–volátil (NVFS)[13] en los dispositivos Palm.

[11] Son calses Java que se dedican a una sola actividad en J2ME.
[12] Mobile Information Device Profile,es un perfil para el desarrollo de aplicaciones para dispositivos móviles
[13] `http://kb.palmone.com/`

Aunque este tipo de almacenamiento permite tener información en el dispositivo, el espacio es muy poco. Además, es un uso compartido por las aplicaciones del teléfono, por lo que es más complicado el manejo de los recursos. La cantidad de información que puede manejarse de esta manera es muy limitada, por lo que solo sería factible para una solución que requiera búsqueda de datos dentro del dispositivo o como auxiliar para el almacenamiento de datos usados por la aplicación.

3.2.2. Sincronización de datos con el dispositivo móvil

En algunos tipos de dispositivo como los PocketPC (dispositivos que ejecutan alguna versión de Windows CE o también llamada Windows Mobile) es posible tener base de datos internas a través de una herramienta llamada SQL Server CE. Esta es una versión diseñada para los dispositivos móviles del sistema manejador de bases de datos SQL Server de Microsoft[38].

Al tener una versión para dispositivos móviles de un sistema de bases de datos es posible obtener información de los sistemas de datos de las aplicaciones para computadoras de escritorio. Esto permite tener acceso de una manera más sencilla a los servidores de bases de datos sin tener toda la información en el dispositivo móvil.

El desarrollo de este tipo de aplicaciones es una ventaja, ya que es posible escribir los programas para PocketPC en el IDE Visual Studio de la misma manera que se hace con las aplicaciones de escritorio. Para desarrollar una aplicación solo se debe indicar que se quiere desarrollar un proyecto SmartDevice e indicar el tipo PocketPC.

En una aplicación se debe indicar el origen de los datos, esto es, un archivo con extensión SDF y después es necesario crear la conexión a la fuente de datos.

Se pueden crear tablas y utilizar sentencias SQL como en un servidor de base de datos común. Para realizar todas estas funciones dentro de la aplicación es necesario utilizar algunas referencias[38], por ejemplo:

- System.Data
- System.Data.Common
- System.Data.SqlServerCE

Además de realizar operaciones en la base de datos se puede sincronizar el propio dispositivo con un servidor instalado en una computadora de escritorio o también de manera remota en un servidor SQL Server[39].

Aunque permite tener datos dentro del dispositivo, la cantidad de información que se puede almacenar en una base de datos en SQL Server CE es muy limitada. Además, aunque se puede realizar consultas(querys) sobre las tablas de la base de datos, el lenguaje SQL es muy básico y permite solo consultas muy simples.

3.2.3. Bases de datos para dispositivos móviles

Las bases de datos para dispositivos móviles son similares a las bases de datos convencionales pero que pueden ser utilizadas en redes para dispositivos móviles[40]. La utilización

de las bases de datos en dispositivos móviles ha aumentado debido a que muchas de las aplicaciones que actualmente se encuentran disponibles en las plataformas para dispositivos móviles requieren almacenar información para trabajar sin la conexión permanente a una red.

Las bases de datos son solo una herramienta para almacenar datos y las búsquedas sobre esos datos se debe hacer de la misma manera que como se hace en los manejadores de bases de datos para las computadoras de escritorio. Algunas de las bases de datos para dispositivos móviles permiten una sincronización con versiones instaladas en servidores.

Existen una gran variedad de productos comerciales que tienen versiones de sus manejadores de bases de datos para dispositivos móviles. A continuación se mencionan algunos:

- **Sybase's SQL Anywhere** – SQL Anywhere es un conjunto de tecnologías que proporcionan herramientas para el manejo de datos. A continuación se muestran las tecnologías SQL Anywhere[54]:
  - **SQL Anywhere Server** – Es un sistema de administración de bases de datos relacionales(RBDMS) que puede utilizarse para el desarrollo de aplicaciones para dispositivos móviles.
  - **UltraLite** – Es un sistema de admnistración de bases de datos diseñado para dispositivos móviles, por ejemplo, PDA's o Smartphones.
  - **Mobilink** – Es una tecnología para la sincronización entre bases de datos relacionales y otras fuentes de datos relacionales.
  - **QAnywhere** – Permite el desarrollo de aplicaciones basadas en mensajes así como también amplía la funcionalidad de web services para dispositivos móviles.
  - **SQL Remote** – Es una tecnología que permite sincronizar datos las bases de datos de SQL Anywhere por medio de archivo, servidor FTP o por correo electrónico de manera ocasional.

  Las bases de datos de SQL Anywhere para dispositivos móviles ofrecen las siguientes características [41]:
  - **Manejo de datos en dispositivos móviles** – Tiene las características de una base de datos empresarial como procedimientos almacenados, interfaz ANSI SQL, triggers, búsqueda full-text, OLAP, además, UltraLite tiene el procesos para la sincronización de datos en la aplicación. Tiene soporte para las plataformas Windows Mobile, Palm OS, Symbian OS y Blackberry.
  - **Sincronización con sistemas empresariales** – Se puede hacer una sincronización con Oracle, Microsoft SQL Server, MySQL, Sybase ASE, entre otras. Se pueden extender servicios de mensajería como JMS y MQSeries para dispositivos móviles con QAnywhere.
  - **Desarrollo de aplicaciones** – Un asistente ayuda en la configuración y sincronización de la base de datos. También cifra todos los archivos de la base de datos y la comunicación de datos con una encriptación de 128 bits.
- **SQL Server CE** – Es el motor de base de datos para dispositivos móviles de Microsoft y está diseado para PocketPC, es decir, dispositivos que funcionan con la plataforma Windows Mobile. Está orientado hacia el almacenamiento local de datos y funciona con

el .Net Compact Framework. Además está desarrollado con la tecnología .NET. Con SQL Server CE se pueden crear bases de datos en el dispositivo y se pueden realizar conexiones con bases de datos SQL Server en un servidor remoto [39].

La edición Compact Edition incluye un subconjunto de SQL Server 2005, como tipos de datos y elementos comunes de Transact-SQL(T-SQL)[14].

Algunas de las características de SQL Server CE son[55]:

- *Instalación*
  - Tamaño de instalación 1.8 MB.
  - Se ejecuta en plataforma Windows Mobile.
  - Se ejecuta como un servicio, con la aplicación.
  - Se instala centralmente con MSI(Microsoft Installer).
- *Archivos de Datos*
  - Un solo archivo.
  - Soporta diferentes extensiones de archivo.
  - Soporta bases de tamaño 4GB máximo.
  - Soporta almacenamiento XML.
- *Programación*
  - Soporta lenguaje T-SQL.
  - Soporta acceso a datos remotos.
  - Sincronización con ADO.Net Sync Framework.
  - Soporta procedimientos almacenados, vistas y triggers.
  - Seguridad por roles.
  - Conexiones concurrentes: 256.

- **DB2 Everyplace** – Es una pequeña base relacional de alto rendimiento para la sincronización de datos entre aplicaciones empresariales y apliciones de dispositivos móviles. Proporciona una base de datos local para el dispositivo. Está disponible en 3 ediciones[42] según las capacidades que se requieran:

  - DB2 Everyplace Enterprise Edition
  - DB2 Everyplace Express Edition
  - DB2 Everyplace Database Edition

  Esta última edición es la que permite tener una base de datos interna en diferentes plataformas para dispositivos móviles. Permite crear tablas o índices y borrar, insertar o actualizar registros de una tabla. A continuación se muestran las características principales[43]:

  - Está optimizada para aplicaciónes para dispositivos móviles SAP.
  - Las bases de datos son de 200 a 350 KB sin requerir de administración.
  - Funcionalidad de lenguaje estructurado de consultas SQL.

[14] T-SQL es una extensión de SQL propiedad de Microsoft y Sybase.

- Encriptación de datos en la base y seguridad en la sincronización de datos.
- Puede ejecutarse sobre plataformas Embedded Linux kernel 2.4 o superior, Palm OS 4.1,5.0 y 5.4, Symbian OS V7s, Windows CE y Windows Mobile.

- **Oracle Lite** – Como otros manejadores de bases de datos, Oracle tiene una versión de su sistema para dispositivos móviles. Tiene la posibilidad de trabajar sin conexiíon de red y también puede sincronizarse para compartir datos con un sevidor de bases de datos oracle. Oracle Lite tiene tres componentes principales[56], se muestran a continuación:

  - **Cliente** – Es un conjunto de aplicaciones que permiten manejar la base de datos internamente en el dispositivo. En primer lugar se incluye la base de datos y se proporciona una forma segura de almacenamiento de datos relacional sin necesidad de un DBA(administrador de base de datos). También incluye un agente de sincronización de datos y una aplicación GUI que permite la sincronización manual y API's para incluirlas en las aplicaciones.
  - **Servidor para dispositivos Móviles** – Es el núcleo de Oracle Lite. Puede instalarse en diferentes servidores con sistemas operativos como Windows, Linux, Sun SPARC, HP-UX y IBM AIX. Además, el sevidor ofrece un sistema de sincronización bidireccional con el cliente en el dispositivo móvil y una interfaz de administración.
  - **Herramientas de Desarrollo** – El Oracle Database Lite Mobile Development Kit(MDK) es un conjunto de herramientas, API's, manuales y ejemplos de código que permiten el rápido dearrollo de aplicaciones con Oracle Lite. El Mobile Database Workbench(MDW) es una herramienta visual para el diseño de replicación de bases de datos.

  La base de datos Oracle Lite soporta interfaces de acceso y estándares abiertos como ODBC, JDBC y ADO.Net lo que permite que se puedan utilizar diferentes IDE que soportan esos estándares. Las características principales de Oracle Lite son las siguientes:

  - Soporta integridad de datos incluyendo integridad referencial.
  - Tamaño de la base de datos es de 4GB máximo.
  - Soporte de Binary Large Object (BLOB).
  - Procedimientos almacenados y triggers.
  - API's de acceso a datos en JDBC, ODBC y ADO.Net .
  - Sincronización bidireccional.
  - Modelo de sincronización Publicación/Subscripción.
  - Llamada a sincronización personalizada.
  - Sincronización en segundo plano.
  - Compresión de datos y encriptación SSL.
  - Lenguaje Scripting para administración automatizada.
  - Extensión de API's.

- **HanDBase** – Es una base de datos originalmente diseñada para los dispositivos Palm aunque ahora se puede ejecutar en diferentes plataformas como el sistema operativo Windows Mobile.

  El sistema tiene las siguientes características[44]:

- Programa
  - Permite al usuario crear y almacenar datos con base en un conjunto de filtros a través de vistas.
  - Tiene soporte para memory cards, se almacenan las bases en la memoria y se abren desde HanDBase.
  - El límite de campos es de 100.
  - Permite tener 200 bases de datos, limitado al espacio en memoria.
  - Permite un número máximo de 65,000 registros, limitado al espacio en memoria.
  - Operaciones básicas para la base de datos y los registros, como borrar, agregar, copiar, mover entre otras.
- Seguridad
  - Encriptar campos de la base de datos, encriptar registros manualmente.
  - Encriptación de 128 bits.
  - La encriptación no está soportada en la platafoma BlackBerry.
  - Permisos de acceso a la bases de datos y de operaciones sobre registros.
- Plataformas
  - Palm OS – En versiones 4.1, 5.0, 5.2, 5.4 o superior.
  - Windows Mobile – Funciona en las versiones 5.x y 6.x. Es necesaro un procesador basado en ARM o X-Scale.
  - Symbian S60 – Dispositivos Symbian S60 con Symbian 9.1 o 9.4.
  - Iphone/Iphone Touch – Con versión del sistema operativo 2.0 o superior.
  - Windows - El programa para la sincronización con la PC necesita de las siguientes versiones de Windows: 2000, Server 2003, XP o Vista.
  - Macintosh - Se necesita de las siguientes versiones para la sincronización con el dispositivo móvil: Mac OS X 10.4, 10.5, 10.6.

Además de los productos comerciales existen bases de datos que son software libre[15], por ejemplo:

- **SQLite** – Es un sistema de base de datos relacional escrito en C. Es una biblioteca de apróximadamente 225 KB. Es un proyecto de software libre diseñado en un principio por D Richard Hipp. SQLite funciona como un componente del programa que lo utiliza. Esto se hace a través de llamadas a funciones como se hace con una biblioteca común en C. Los archivos que almacenan los datos de la base así como índices, tablas y registros se guardan en un solo archivo estándar en el dispositivo[45].

  SQLite se utiliza en sistemas diseñados para computadores personales. Sin embargo también existen proyectos que involucran dispositivos móviles como la plataforma Android y aplicaciones para Pocket PC.

  SQLite tiene las siguentes características[46]:

  - Sin Administración – SQLite no necesita ser instalado para utilizarlo, no necesita ser configurado.

[15] Se puede consultar en `http://es.wikipedia.org/wiki/Software_libre`.

- Sin Servidor – El proceso de interacción entre el cliente y servidor debe ser implementado por separado. SQLite solo funciona dentro de la aplicación y los datos que se encuentran en el dispositivo.
- Lenguaje SQL – Implementa la mayoría de SQL92[16].
- API's – Ofrece una API simple, la interfaz se compone de tres categorías: Objetos, Constantes y Funciones.
- Extensiones – Aunque está escrito en C, existen diferentes Wrappers[17] y Bindings[18] en otros lenguajes.
- Plataformas – El sistema es multi–plataforma, no tiene dependencias externas, está escrito en ANSI C.
- Licencia – El código fuente del sistema es de dominio público.
- Tamaño – Es compacto, no necesita más de 300 KB en el dispositivo para utilizarlo.
- Tipos –Soporte para tipos de datos BLOB[19]

- **Db4o** – Es una base de datos relacional orientada a objetos que fue creada por Versant Corporation. Tiene una licencia libre GNU[20]. Está disponible para sistemas .NET y Java[47]. Puede utilizarse en dispositivos móviles que tengan plataformas J2ME o en dispositivos PocketPC o con algún sistema operativo Windows Mobile.

  Algunas de las características más importantes de Db4o son[48]:

  - Plataformas – Puede trabajar con plataformas Java y .NET o con servidores desde clientes de esas plataformas.
  - Tamaño – El máximo de un archivo regular de base de datos puede ser hasta 254 GB, aunque está limitado a la capacidad del dispositivo. El espacio requerido por el sistema de base de datos es aproximadamente de 600 KB para instalación.
  - Operación – Puede trabajar de modo local o cliente/servidor, transacciones multiples, modo de solo lectura y otros.
  - Replicación – Cuenta con un sistema de replicación, db4o a db4o, puede trabajar con Hibernate [21]
  - Reportes – Para objetos Java puede utlizar JasperReport, JReport entre otros y en .NET ReportViewer.
  - Seguridad – Archivos de Base de datos con password, y encriptación sencilla.
  - Administración – Sin necesidad de un DBA, recuperación automática, seguridad en threads, manejo de espacio.

3.2.4. Acceso a bases de datos a través de un componente web

Se pueden realizar consultas a una base de datos a través de un componente web que puede ser una página o un servicio. Esto significa que la aplicación en el dispositivo móvil no

[16] `http://www.sqlite.org/omitted.html`
[17] Son capas de código que cubren código existente.
[18] Un binding es una adaptación de una biblioteca en un lenguaje de programación diferente.
[19] Binary Large OBjects, se utilizan en bases de datos para manejar imágenes u objetos multimedia.
[20] Es una licencia creada por la Free Software Foundation.
[21] Es un framework de mapeo objeto–relacional para java.

accede a la fuente de datos directamente. Lo que sucede es que el componente se conecta a la base de datos para obtener la información y después se pasa esa información a la aplicación del dispositivo móvil que la solicitó.

Se puede analizar el caso de las aplicaciones con tecnología Java, la especificación J2ME que está diseñada para los dispositivos móviles puede ser integrada con aplicaciones J2EE, que es la especificación para las aplicaciones empresariales y entornos distribuidos.

J2ME proporciona herramientas para crear aplicaciones distribuidas. Además, se pueden integrar apliaciones para dispositivos móviles con sistemas J2EE en el lado del servidor. En el lado del cliente, las aplicaciones pueden ejecutarse en el dispositivo que tenga las características requeridas por la plataforma Java. La comunicación se puede realizar a través del protocolo HTTP.

En general, la secuencia que sigue una aplicación que accede a una base de datos de manera distribuida y a través de un componente web es la siguiente[57]:

- La aplicación cliente envía los datos en una petición HTTP indicando en las cabeceras los datos requeridos para procesar de manera correcta la petición.
- En el caso de Java el servlet recibe la petición y la decodifica, procesando la información, realizando las operaciones definidas y conectándose a la base de datos para extraer los datos que se solicitan.
- Después de tener la información solicitada, el servlet codifica los datos y los envía en una respuesta HTTP a la aplicación cliente. En otros casos el servlet puede generar una página web, por ejemplo en XML para que el cliente tome los datos.
- Por último el cliente recibe los datos si fueron enviados y decodifica la respuesta para manejar la información según sus funciones.

Cuando se realiza la comunicación entre las aplicaciones el tipo de mensaje puede ser de diferentes tipos, por ejemplo puede ser texto plano, pueden ser objetos, XML y algunos otros.

3.3. Resumen

En la primera parte del capítulo se menciona la función de un motor de búsqueda web y sus componentes principales. Se explica en qué consiste la búsqueda en dispositivos móviles y los tipos de búsqueda que se pueden utilizar. También se mencionan sus principales características. Por último, se muestran algunos de los principales motores de búsqueda en dispositvos móviles que actualmente están funcionando.

En la segunda parte se muestran cuatro formas para el acceso a los datos desde un dispositivo móvil. Se revisa el método que permite almacenar los datos en el dispositivo y una forma de sincronizar los datos desde una computadora fija a un dispositivo móvil. Se muestran las características y funciones básicas de los sistemas de bases de datos más utilizados en los dispositivos móviles, por ejemplo, los sistemas comerciales Sybase SQL Anywhere, SQL Server CE, DB2 Everyplace, entre otros, y los sistemas libres como SQLite y Db4o. Por último se explica brevemente un método para acceder a la información por por medio de un componente web.

Capítulo 4

Diseño de una aplicación de búsquedas

Para este capítulo se ha tomado como base el método de diseño descrito en el artículo "The Practical Design Method: A Software Design Method for a First Object-Oriented Project"[50], siguiendo las fases para el sistema llamado *Tlatemoani*. Este método es la guía del sistema en todas sus formas, desde la fase de requerimientos hasta las pruebas del sistema[50]. En este capítulo se describe el diseño para cada una de esas formas. Inicia con los requerimientos del sistema y después el diseño de sistema además del diseño detallado a través de diagramas, programación y pruebas. La fase de programación se trata completamente en el capítulo 5, así como el resultado de las pruebas al sistema.

4.1. Análisis de requerimientos

En esta sección se describen los requerimientos que debe cumplir la aplicación:

- Requerimientos en tiempo de ejecución
  - **Caso de Uso 1: Ingreso a la pantalla de búsqueda.**

    Al ingresar a la aplicación en la primera pantalla el sistema debe pedir un nombre de usuario y una clave para poder acceder.
  - **Caso de Uso 2: Pantalla de ayuda.**

    En la aplicación debe mostrarse un texto que ayude al usuario a comprender el funcionamiento de la aplicación y que le permita utilizar las diferentes opciones del sistema.
  - **Caso de Uso 3: Selección de índices.**

    La aplicación debe tener una pantalla que permita al usuario seleccionar los índices sobre los que quiere hacer su búsqueda.
  - **Caso de Uso 4: Realizar una búsqueda.**

    En la aplicación el usuario debe escribir una o más palabras para realizar una búsqueda.

- **Caso de Uso 5: Lista de resultados por número de coincidencias.**

  Se debe presentar al usuario el número de coincidencias con su búsqueda en cada uno de los índices que seleccionó antes de mostrarle los resultados en detalle.

- **Caso de Uso 6: Lista de resultados en detalle.**

  La aplicación debe mostrar los resultados en detalle, es decir, toda la información con respecto a un resultado en particular, se debe mostrar a través de una lista que contenga un título para cada uno de los resultados de la búsqueda.

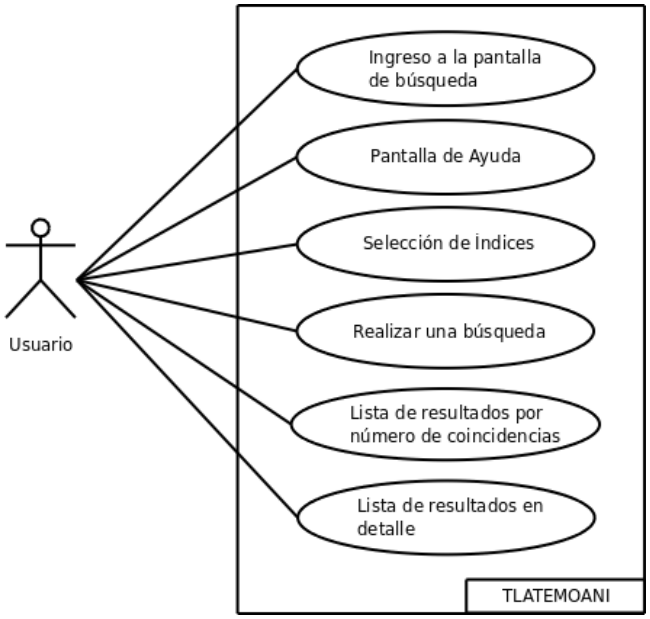

Figura 4.1: Diagrama Casos de Uso

- Requerimientos en tiempo de construcción

  1. La aplicación debe tener una pantalla de acceso.
  2. La aplicación debe hacer una búsqueda en una base de datos que contiene texto.
  3. La aplicación debe mostrar el número de resultados en cada índice seleccionado.
  4. Se debe permitir seleccionar los índices para realizar la búsqueda.
  5. Se deben enlistar los resultados mostrando un título para cada resultado.
  6. Se debe regresar un máximo de 30 resultados por consulta.
  7. Se debe mostrar una pantalla de ayuda para el uso de la aplicación.

4.2. Diseño de sistema

A continuación se muestra la **Lista de Operaciones** del proyecto *Tlatemoani*:

1. Verificar el acceso a la búsqueda.
2. Realizar una búsqueda.

3. Elegir los índices en los cuales se debe buscar.

4. Mostrar el número de resultados para cada índice elegido.

5. Mostrar los resultados.

6. Mostrar ayuda.

4.2.1. Arquitectura general

En la Figura 4.2 se muestra la arquitectura general del sistema *Tlatemoani*:

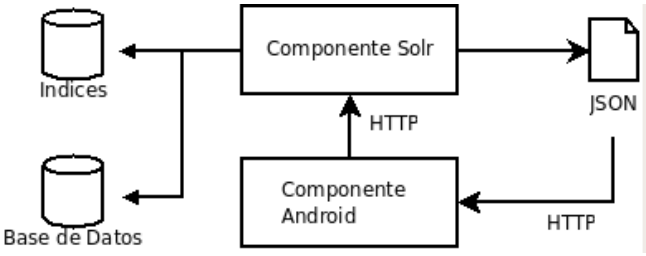

Figura 4.2: Arquitectura General

La arquitectura se basa en dos componentes, el componente Solr permite la conexión entre la base de datos y el dispositivo, también se encarga de realizar las búsquedas. Y el componente Android se conecta al componente Solr a través del protocolo HTTP y recibe respuestas con contenido JSON.

4.2.2. Arquitectura del componente Solr

Para realizar las búsquedas, el componente Solr recibe una petición a través de HTTP. Después utiliza clases del cliente Java para Solr (llamado *SolrJ*) y regresa los resultados en objetos JSON[1] por medio de servlets.

En la Figura 4.3 se puede ver el diseño de la arquitectura del componente Solr:

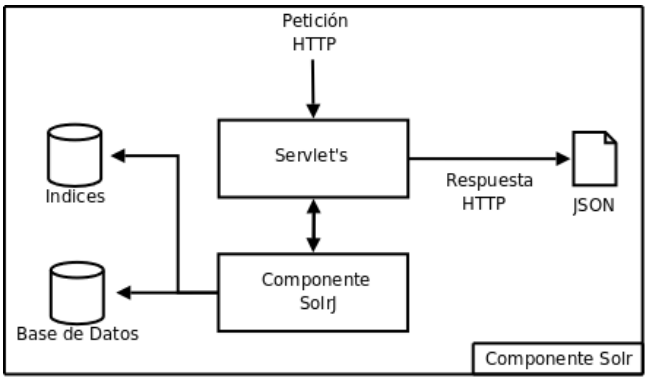

Figura 4.3: Arquitectura Componente Solr

[1]JSON es un formato para el intercambio de información a través de objetos y arreglos o estructuras de datos simples

4.2.3. Arquitectura del componente Android

La interfaz se debe implementar para el dispositivo móvil por medio de las herramientas de Android. El componente hace una petición al servlet por medio de HTTP y recibe una respuesta JSON. Del objeto JSON se obtiene la información y se almacena en un objeto del tipo Dato que podrá ser accesible a las clases "Actividad", que generan las pantallas para mostrar la información utilizando archivos XML para construir la interfaz de usuario.

En la Figura 4.4 se puede ver el diseño de arquitectura del componente Android.

4.3. Diseño detallado

Esta sección se divide en dos etapas, en la primera se muestra el diseño detallado del componente Solr y en la segunda el diseño en detalle del componente Android. En cada etapa se muestran los diagramas de clases y de interacción. Y se describe el funcionamiento principal de cada clase.

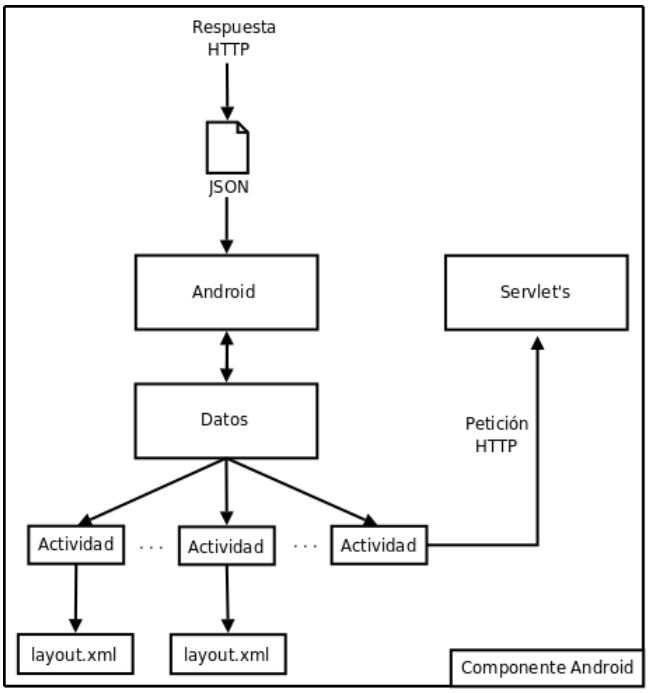

Figura 4.4: Arquitectura Componente Android

4.3.1. Componente Solr

En la Figura 4.5 se muestra el diagrama de clases. Se puede ver cómo se relacionan las clases. Los servlets utilizan las clases para realizar las tareas correspondientes. En el servlet accesoServlet se utiliza la clase ConexionSQL para verificar el acceso del usuario. En el servlet indicesServlet se utilizan las clases de conexión al servidor Solr. La clase EjecutorQuery sirve para enviar una consulta al servidor y la clase ResultadoQuery permite el manejo del resultado de la búsqueda. También se utiliza la clase ControladorIndices para revisar los índices disponibles para la búsqueda en el servidor.

Figura 4.5: Diagrama Clases Componente Solr

En las siguientes figuras se muestran cada una de las clases del componente Solr:

- **Campo**:
  Esta clase define el concepto de campo para los índices de búsqueda, contiene datos como el nombre, id, estado entre otros.

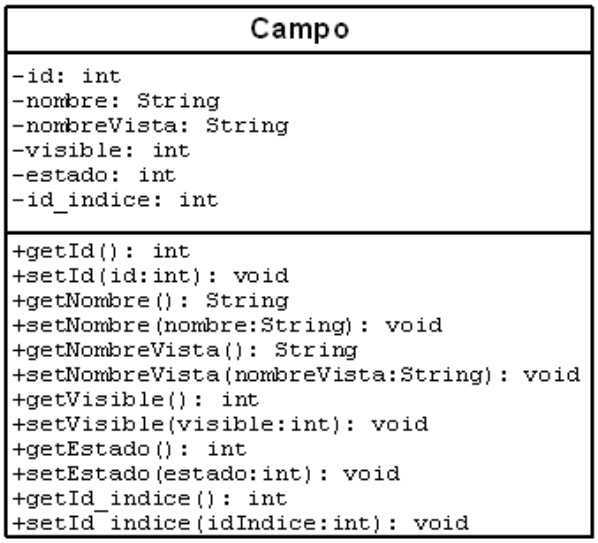

Figura 4.6: Clase Campo

- **Indice**:
  Esta clase define el concepto de índice, para utilizar los índices de búsqueda dentro del componente Solr.

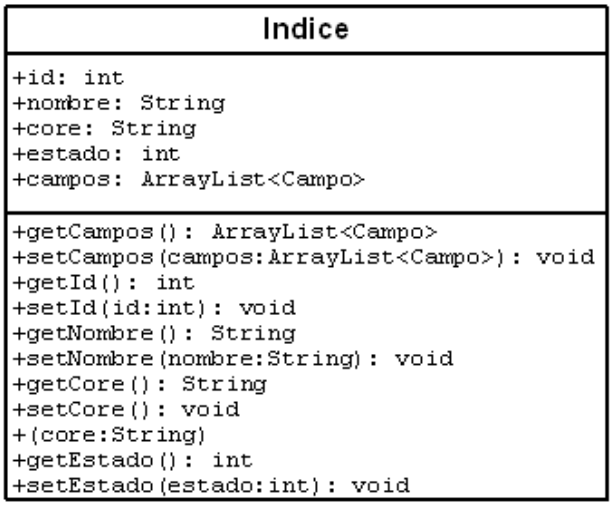

Figura 4.7: Clase Indice

- **ControladorIndices**:
  Esta clase se utiliza para construir la lista de índices y obtiene la información desde la base de datos. Utiliza también una lista de campos.

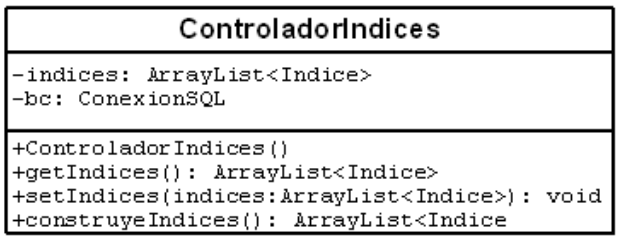

Figura 4.8: Clase ControladorIndices

- **EjecutorQuery**:
  Esta clase ejecuta la consulta en el servidor Solr, en el *core*(se puede consultar en el capítulo 5) que se indica en el método ejecutaQuery.

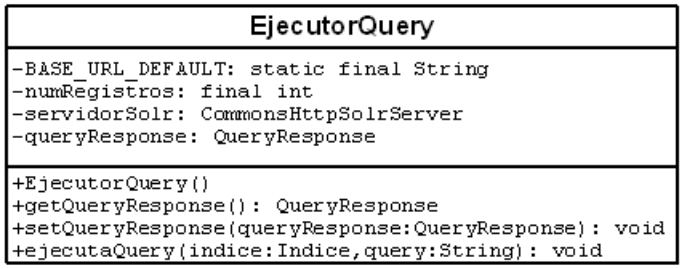

Figura 4.9: Clase EjecutorQuery

- **ResultadoQuery**:
  Esta clase maneja el resultado de una consulta al servidor Solr. La clase debe ser utilizada en conjunto con EjecutorQuery.

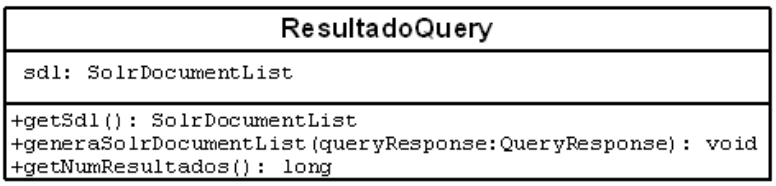

Figura 4.10: Clase ResultadoQuery

- **ConexionSQL**:
  Esta clase permite la conexión a la base de datos MySQL donde se encuentran los datos para la aplicación.

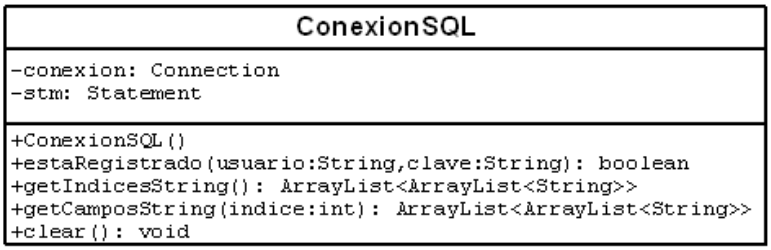

Figura 4.11: Clase ConexionSQL

- **accesoServlet**:
  Este servlet regresa una respuesta JSON, se utiliza para verificar que el usuario está registrado en la base de datos. El método es muy simple y solo es un ejemplo de como verificar los datos, un método que proporcione más seguridad para el acceso de los datos deberá ser implementado.

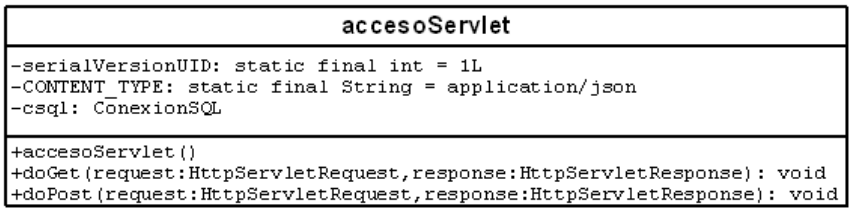

Figura 4.12: Clase accesoServlet

- **indicesServlet**:
  Este servlet regresa objetos JSON dependiendo del método que se pase como parámetro:

  1. Se regresa la información correspondiente a los índices disponibles
  2. Se regresa información sobre el número de resultados sobre los índices seleccionados
  3. Regresa los resultados de una consulta sobre un índice específico

  La respuesta de cada uno de los métodos es un objeto JSON enviado como respuesta.

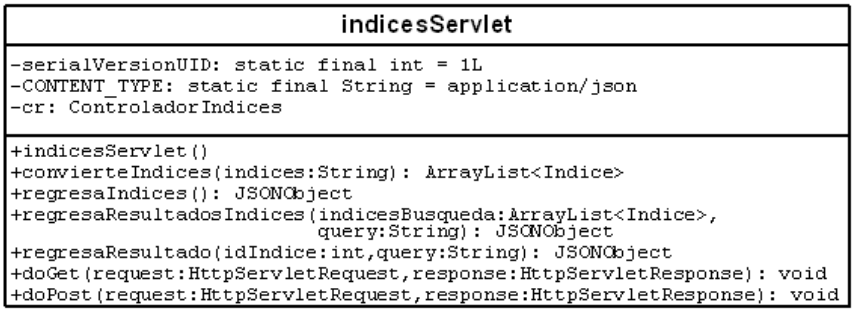

Figura 4.13: Clase indicesServlet

4.3.2. Componente Android

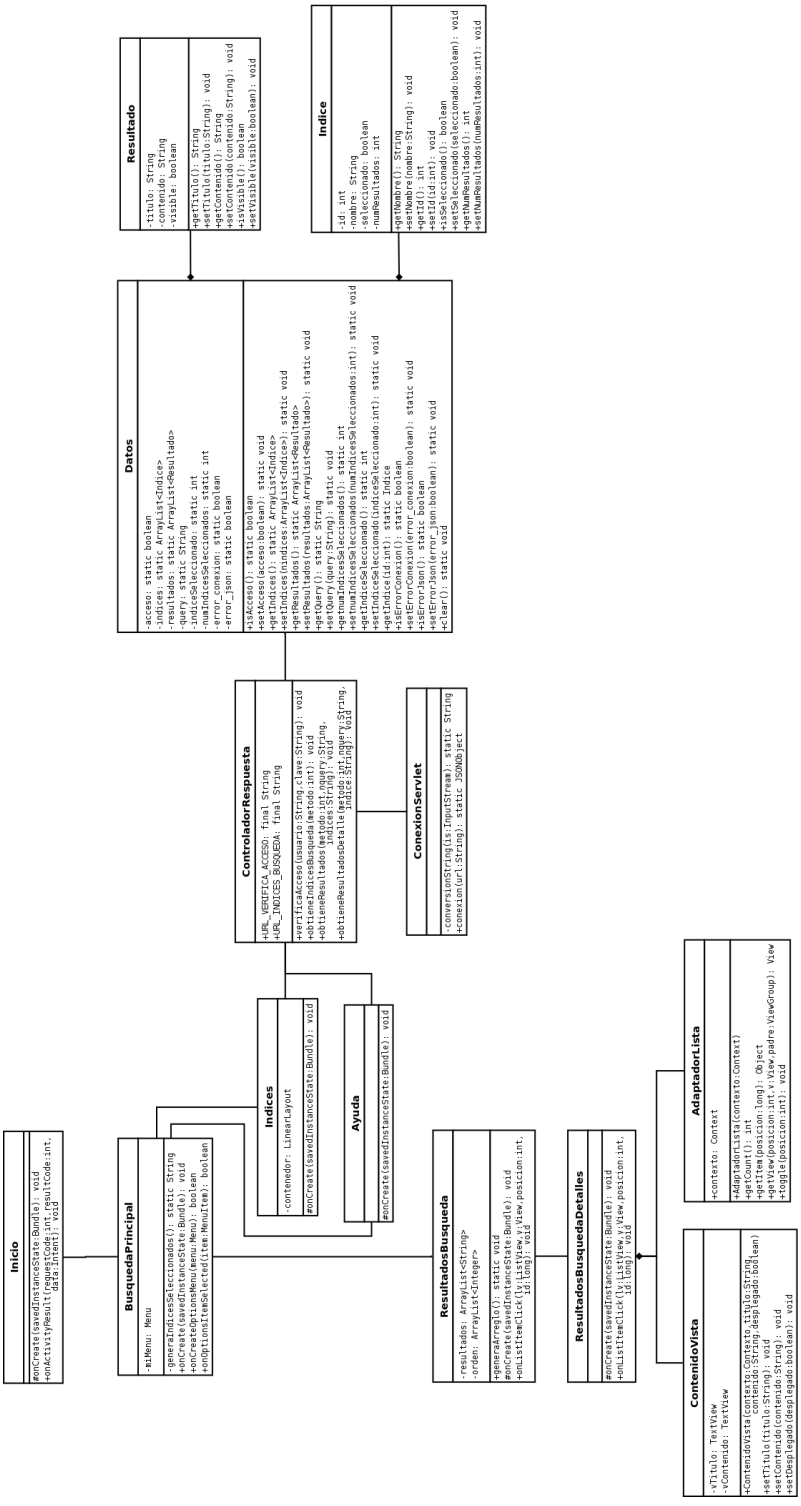

Figura 4.14: Diagrama clases del componente Android

En la Figura 4.14 se muestra el diagrama de clases. La clase ResultadoBusquedaDetalle tiene 2 clases internas. La clase ContenidoVista genera los elementos necesarios para construir la interfaz de usuario mientras AdaptadorLista realiza las funciones de la lista de resultados. Todas las pantallas de la aplicación se construyen con las clases del tipo Activity y estas clases acceden a los datos comunes por medio de la clase Datos. La clase Datos también es utilizada por la clase ControladorRespuesta para almacenar las respuestas que se obtienen con la clase ConexionServlet y almacenarlas en las clases Indice y Resultado.

En las siguientes figuras se muestran cada una de las clases del componente Android:

- **Indice**:
  Esta clase representa un índice de búsqueda dentro de la aplicación en el componente Android.

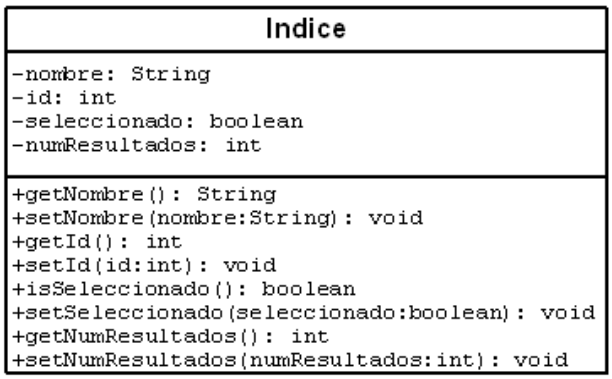

Figura 4.15: Clase Indice

- **Resultado**:
  Esta clase representa un resultado de una búsqueda para poder presentarlo en forma de lista en la actividad ResultadosBusquedaDetalle.

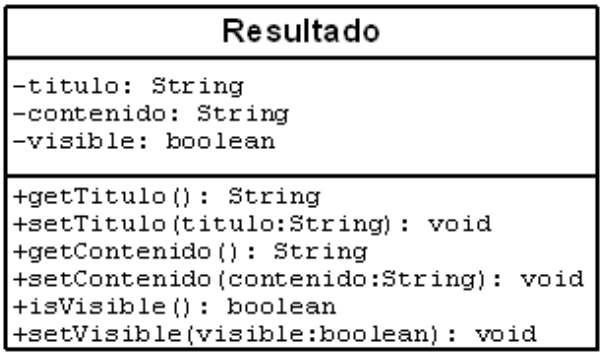

Figura 4.16: Clase Resultado

- **ControladorRespuesta**:
  Esta clase, se utiliza para obtener realizar las peticiones y poner la información en la clase Datos para que las actividades tengan acceso a los datos.

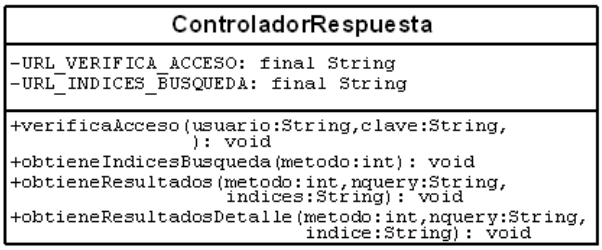

Figura 4.17: Clase ControladorRespuesta

- **ConexionServlet**:
  Esta clase crea la conexión entre la aplicación en Android y el servlet, por medio del método conexion obtiene el objeto JSON como respuesta.

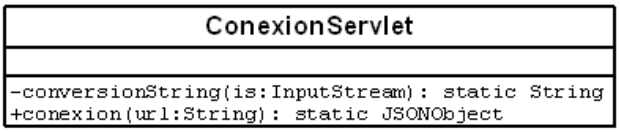

Figura 4.18: Clase ConexionServlet

- **Datos**:
  Esta clase se utiliza para almacenar los datos y estados de los elementos que se utilizan a través de toda la aplicación.

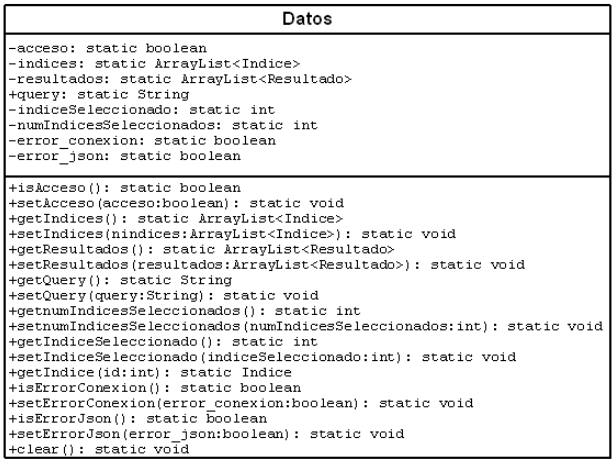

Figura 4.19: Clase Datos

- **Inicio**:
  Esta clase genera la pantalla de Inicio de la aplicación por medio del layout inicio, se definen cajas de texto para que el usuario escriba su nombre y clave para tener acceso a la aplicación.

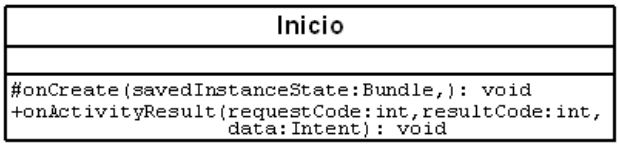

Figura 4.20: Clase Inicio

- **BusquedaPrincipal**:
  Esta clase genera la pantalla principal de búsqueda en la aplicación por medio del layout busqueda_1, crea un Intent(en sección 2.3.2.3) hacia la pantalla donde se muestran las coincidencias de cada índice de búsqueda.

  También crea un menú con las opciones: Índices, Ayuda y Salir.

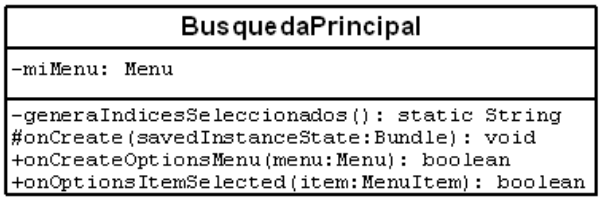

Figura 4.21: Clase BusquedaPrincipal

- **Indices**:
  Esta clase genera la pantalla de índices de la aplicación por medio de los métodos de la clase. No se utiliza layout, ya que los elementos deben ser creados dinámicamente según los índices que estén activos.

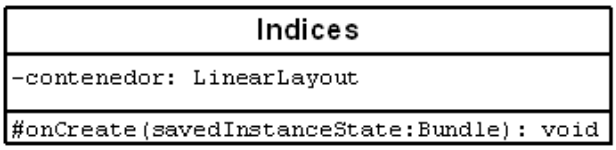

Figura 4.22: Clase Indices

- **Ayuda**:
  Esta clase genera la pantalla de ayuda de la aplicación por medio del layout ayuda.xml.

Figura 4.23: Clase Ayuda

- **ResultadosBusqueda**:
  Esta clase genera la pantalla de número de coincidencias para cada índice.

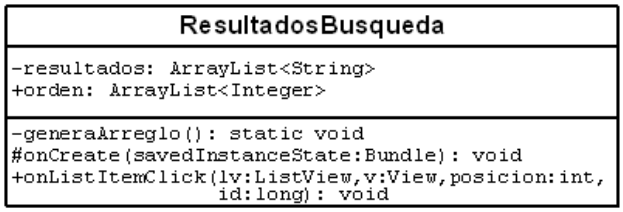

Figura 4.24: Clase ResultadosBusqueda

- **ResultadosBusquedaDetalle**:
  Esta clase genera la pantalla de número de coincidencias para cada índice, se despliega una lista con los elementos.

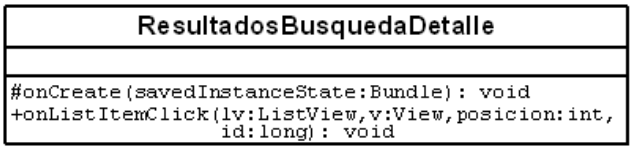

Figura 4.25: Clase ResultadosBusquedaDetalle

El diagrama de interacción describe cómo se realiza cada una de las operaciones descritas anteriormente según el caso de uso:

- **Caso de Uso 1: Ingreso a la pantalla de búsqueda**:

  En este diagrama se muestra como se realiza el proceso de verificación de los usuarios para ingresar en el sistema y los objetos que intervienen en este proceso, desde que se carga la aplicación por medio del método onCreate(), y hasta el paso final que es crear el nuevo Intent para cargar las siguiente actividad dentro de la aplicación. Figura 4.26.

- **Caso de Uso 2: Pantalla de Ayuda**:

  Cuando el usuario se encuentra en la pantalla de búsqueda principal puede acceder por medio del botón **menu** del dispositivo al menú de la aplicación. Entre las opciones está la Ayuda. Al elegir la opción Ayuda, se crea un nuevo Intent para mostrar la pantalla con la información del uso de la aplicación. Figura 4.27.

- **Caso de Uso 3: Selección de Indices**:

  Cuando se elige la opción Indices en el menú de la aplicación, se crea un nuevo Intent desde la pantalla de búsqueda principal para mostrar la pantalla de selección de índices. Antes de generar los índices se realiza la conexión al servlet para obtener los datos de los índices disponibles con el fin de construir la pantalla de selección. Figura 4.28.

- **Caso de Uso 4: Realizar una búsqueda**:

  Al realizar una búsqueda se realiza una petición HTTP al servlet por medio del método conexión del objeto ConexionServlet. La respuesta se maneja utilizando los métodos

de la clase ControladorRespuesta y utilizando el método setNumResultados de la clase Datos para saber el número de coincidencias en cada índice. Figura 4.29.

- **Caso de Uso 5: Lista de Resultados por número de coincidencias**:

  Después de realizar la consulta se muestra la pantalla de resultados con el nombre del índice y el número de coincidencias en forma de lista. Cuando se presiona un elemento de la lista se realiza la operación definida en el método OnListItemClick() y en caso de que el índice tenga un número de coincidencias mayor que cero se crea un nuevo Intent para cargar la actividad de resultados en detalle. Figura 4.30.

- **Caso de Uso 6: Lista de Resultados en detalle**:

  Para mostrar los resultados finales se obtiene la respuesta del servlet por medio de una conexión con la clase ConexionServlet. Los resultados se almacenan en la lista de tipo Resultado y se obtienen a través de la clase Datos. Se obtiene el título, el contenido y el estado de cada elemento de la lista de resultados y se agrega a la lista por medio de la clase AdaptadorLista con los métodos set para cada atributo. Figura 4.31.

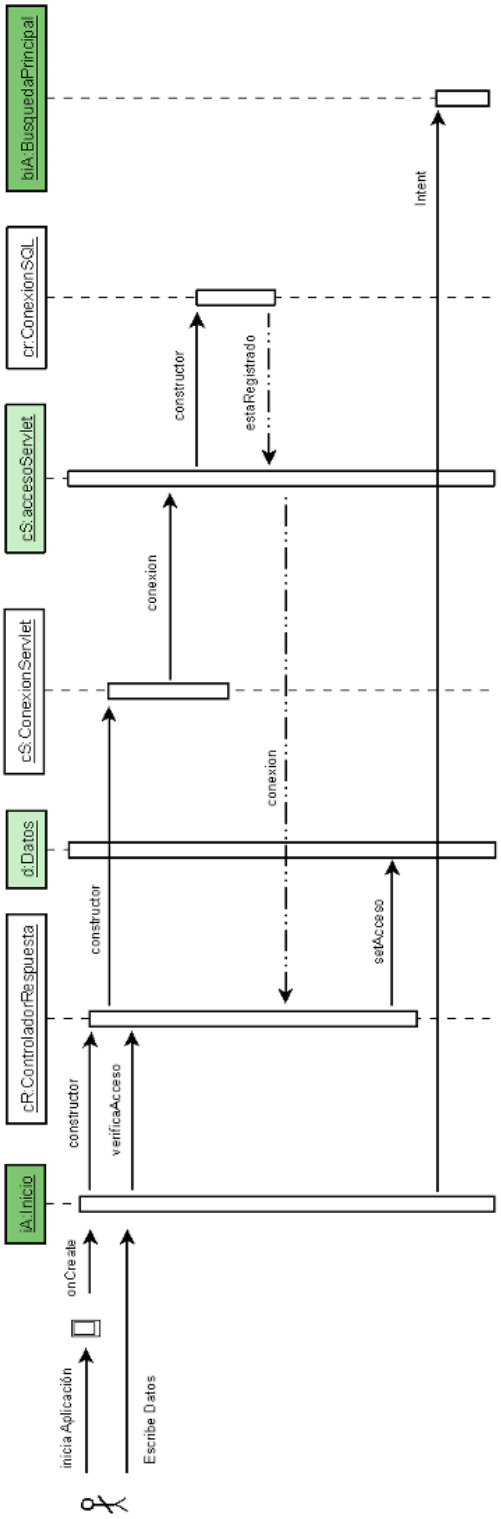

Figura 4.26: Diagrama de interacción para caso de uso 1

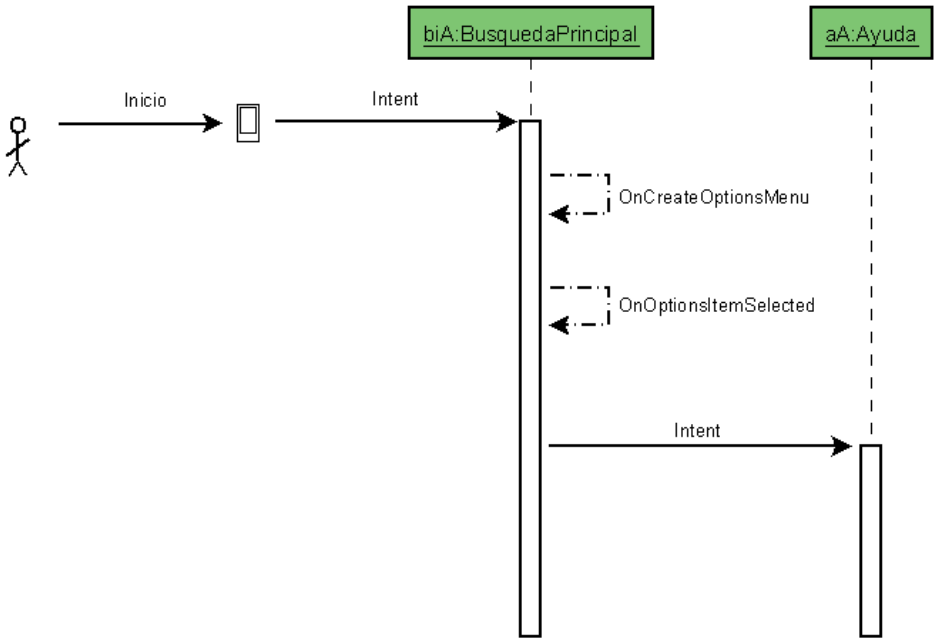

Figura 4.27: Diagrama de interacción para caso de uso 2

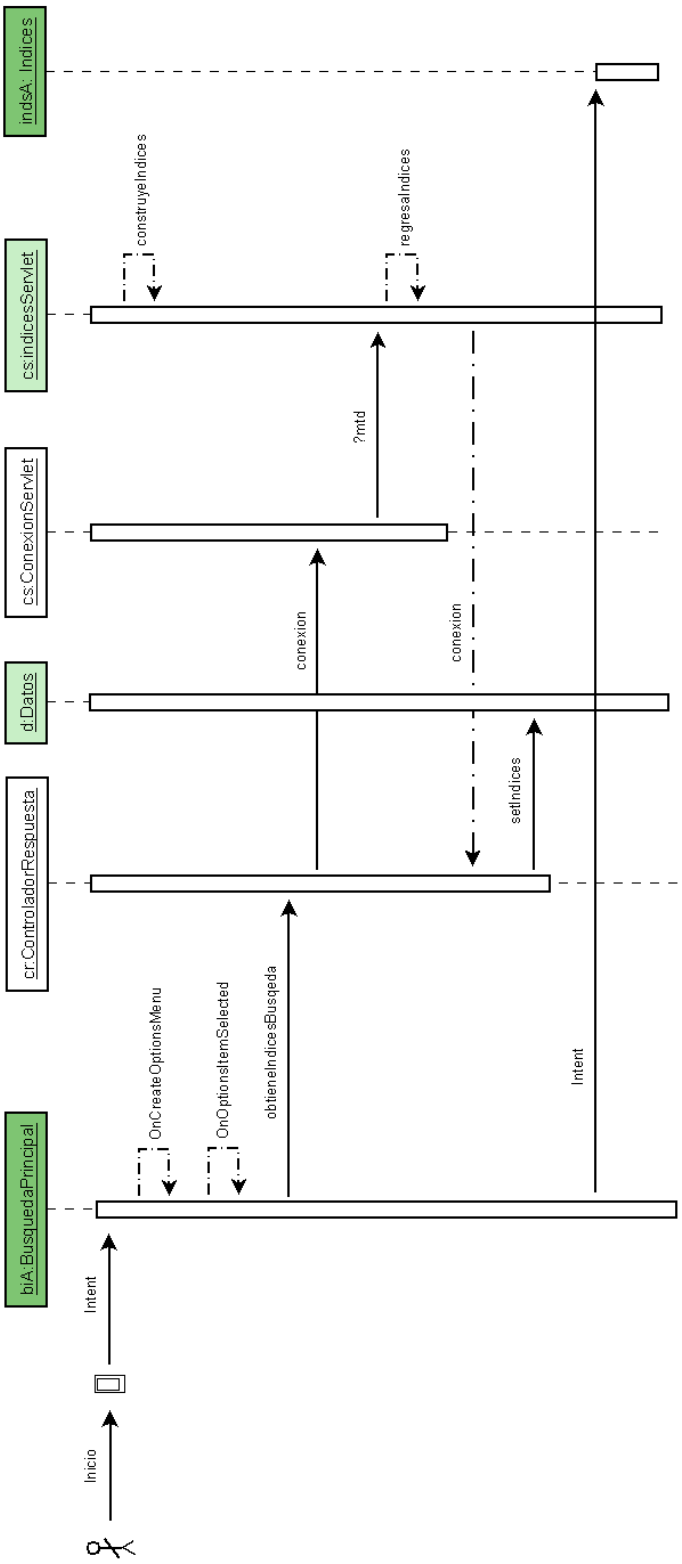

Figura 4.28: Diagrama de interacción para caso de uso 3

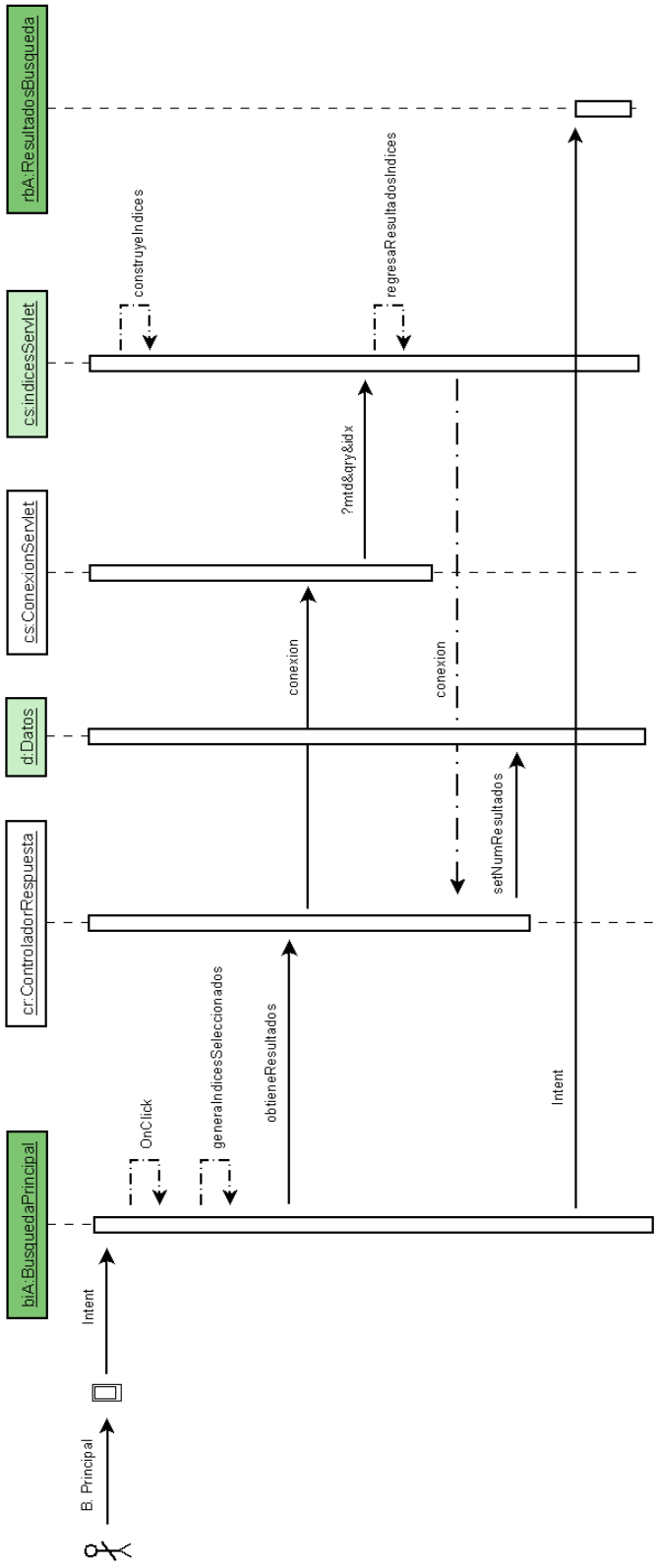

Figura 4.29: Diagrama de interacción para caso de uso 4

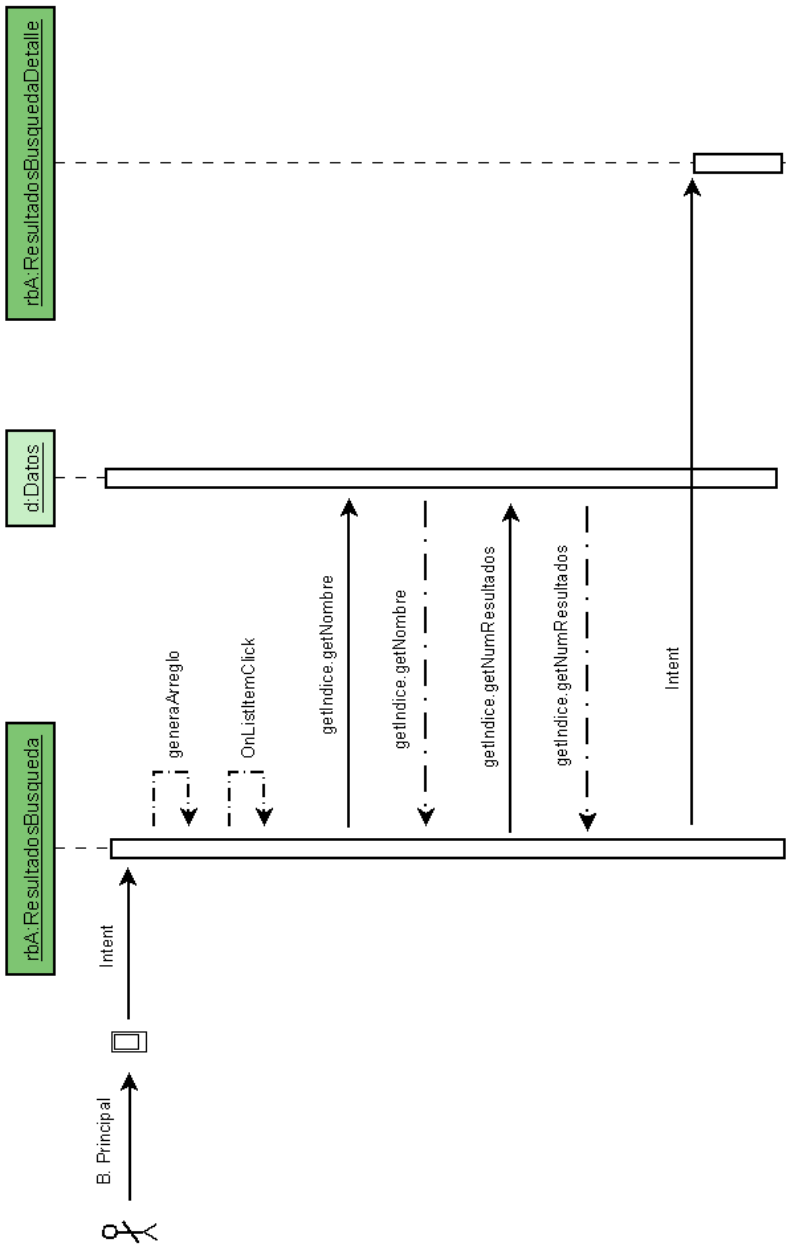

Figura 4.30: Diagrama de interacción para caso de uso 5

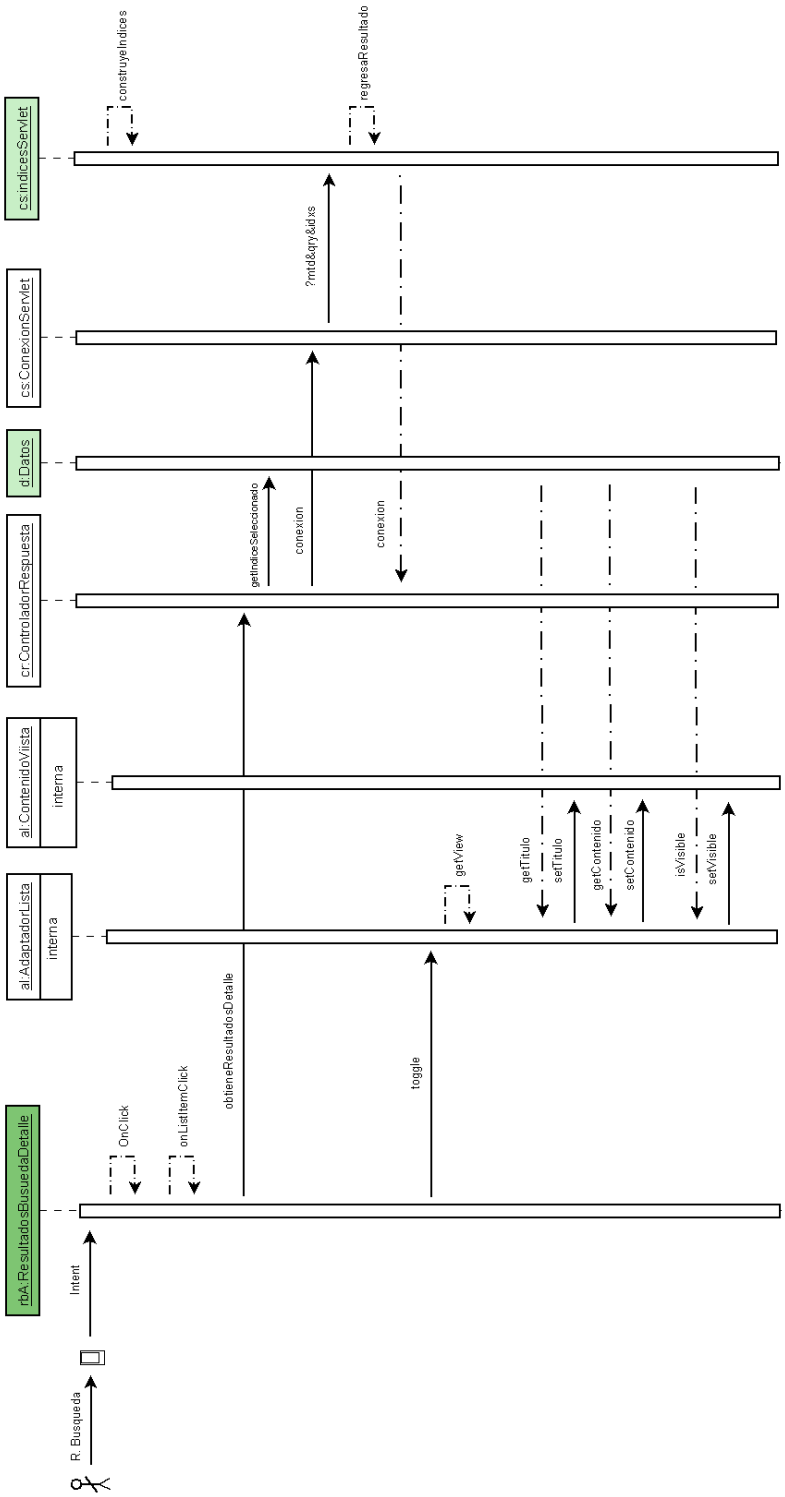

Figura 4.31: Diagrama de interacción para caso de uso 6

4.4. Plan de pruebas

En la etapa de pruebas se deben verificar los siguientes casos de prueba para la aplciación:

I. **Ingreso a la pantalla de búsqueda**

Nombre : Ingreso
Pre–Condición: Conexión Internet, Registro servlet, BD
Secuencia Principal:

1) El usuario escribe nombre de usuario.
2) El usuario escribe su clave.
3) El sistema se conecta al sevlet y verifica datos.
4) El sistema muestra la pantalla inicial de búsqueda.

Alternativas: En cualquier momento se puede presionar el botón **Home** para salir del sistema
Errores:

1) Si no hay conexión se muestra un mensaje.
2) Si el nombre de usuario es incorrecto se muestra un mensaje.
3) Si la clave es incorrecta se muestra un mensaje.

Post–Condición: Se muestra la pantalla inicial de búsqueda.
Notas: El usuario y clave deben ser agregados en la base de datos por el administrador.

II. **Pantalla de Ayuda**

Nombre : Ayuda
Pre–Condición: Encontrarse en la pantalla de búsqueda inicial, Servlet,Solr
Secuencia Principal:

1) El usuario presiona el botón **menu**.
2) El usuario selecciona opción Ayuda.

Alternativas:

1) En cualquier momento se puede presionar el botón

   `"<-"`

   para regresar a la pantalla inicial de búsqueda.

Errores:

1) Se debe revisar el sistema de actividades o el archivo manifiesto durante la fase de desarrollo.

Post–Condición: Se muestra la pantalla ayuda.
Notas: La pantalla de ayuda solo es texto.

III. **Selección de índices**

Nombre : Selección
Pre–Condición: Encontrarse en la pantalla de búsqueda inicial, Servlet,Solr
Secuencia Principal:

1) El usuario presiona el botón **menu**.
2) El usuario selecciona la opción "Indices".
3) El sistema muestra la pantalla de índices disponibles.
4) El usuario marca o desmarca los índices que elige.
5) El sistema registra los índices de selección para realizar la petición de búsqueda sobre esos índices únicamente.

Alternativas:

1) En cualquier momento se puede presionar el botón **Home** para salir del sistema.
2) En cualquier momento se puede presionar el botón **menu** para mostrar el menú del sistema nuevamente.
3) En cualquier momento se puede presionar el botón

   `"<-"`

   para salir del sistema.

Errores:

1) Se debe revisar el sistema de actividades o el archivo manifiesto durante la fase de desarrollo.

Post–Condición: Se asignan los índices de búsqueda en el sistema
Notas: Al iniciar la pantalla búsqueda inicial todos los índices están seleccionados

IV. **Realizar búsqueda**

Nombre : Búsqueda
Pre–Condición: Encontrarse en la pantalla de búsqueda inicial,Conexión, Servlet,Solr
Secuencia Principal:

1) El usuario escribe una consulta(palabra(s)).
2) El usuario presiona el botón **Buscar**.
3) El sistema se conecta al sevlet y realiza la consulta.
4) El sistema muestra la pantalla de Número Coincidencias.

Alternativas:

1) En cualquier momento se puede presionar el botón **Home** para salir del sistema.
2) En cualquier momento se puede presionar el botón **menu** para mostrar el menú del sistema.
3) En cualquier momento se puede presionar el botón

"<-"

para salir del sistema.

Errores:

1) Si no hay conexión se muestra un mensaje.
2) Si no se escribe una consulta se muestra un mensaje.
3) Si no está seleccionado un índice se muestra un mensaje.

Post–Condición: Se muestra la pantalla Número Coincidencias.
Notas: Se muestran los índices aunque no tengan resultados para que el usuario confirme que en ese índice no se encontró ninguna coincidencia.

V. **Lista de resultados por número de coincidencias**.

Nombre : Número Coincidencias.
Pre–Condición: Encontrarse en la pantalla Número Coincidencias, Conexión, Servlet,Solr.
Secuencia Principal:

1) El usuario selecciona un índice con resultados.
2) El sistema muestra la pantalla de resultados en detalle.

Alternativas:

1) El usuario selecciona un índice con 0 resultados, el sistema regresa a la pantalla inicial de búsqueda.
2) En cualquier momento se puede presionar el botón **Home** para salir del sistema.
3) En cualquier momento se puede presionar el botón

   "<-"

   para regresar a la pantalla inicial de búsqueda.

Errores:

1) Si no hay conexión se muestra un mensaje.

Post–Condición: Se muestra la pantalla de resultados en detalle.
Notas: Se muestran los índices aunque no tengan resultados para que el usuario confirme que en ese índice no se encontró ninguna coincidencia.

VI. **Lista de resultados en detalle**

Nombre : Detalles.
Pre–Condición: Encontrarse en la pantalla de resultados en detalle, Servlet,Solr
Secuencia Principal:

1) El usuario presiona un elemento de la lista de resultados.
2) El sistema muestra el contenido del resultado en detalle.

Alternativas:

1) El usuario selecciona un elemento de la lista, el sistema oculta el contenido del resultado.
2) En cualquier momento se puede presionar el botón **Home** para salir del sistema.
3) En cualquier momento se puede presionar el botón

   `"<-"`

   para regresar a la pantalla Número Coincidencias.

Errores:

1) Se debe revisar el sistema de actividades o el archivo manifiesto durante la fase de desarrollo.

Post–Condición: Se muestra la pantalla de resultados en detalle.
Notas: El contenido de cada resultado aparece oculto hasta que el usuario presiona el título.

4.5. Resumen

En este capítulo se muestran los diagramas de cada uno de los componentes del proyecto *Tlatemoani*, una aplicación de búsqueda y recuperación de información para un dispositivo móvil. En primer lugar se definen los requerimientos y después se diseñan los casos de uso para el sistema. En las secciones siguientes se muestran los diagramas de clases y de interacción de cada uno de los componentes del sistema.

Al final del capítulo se diseñan los casos de prueba para el sistema y los resultados de estas pruebas se muestran en el capítulo 5, así como la implementación de las clases y la configuración de las herramientas necesarias.

Capítulo 5

Implementación de una aplicación de búsquedas

En este capítulo se describe el proceso de implementación de la aplicación de búsqueda y recuperación de la información. Se inicia mostrando la instalación y configuración básica de las herramientas que se utilizan para el desarrollo del programa como se describe en la sección 2.1.2.3. Después de la instalación se explican algunos detalles importantes en la configuración de los componentes para el funcionamiento de la aplicación.

En la segunda parte se muestran los puntos principales de la implementación de cada uno de los componentes de la aplicación. En primer lugar se muestran algunas de las funciones principales en el componente Solr, así como los servlet que se utilizan para interactuar con el dispositivo Android.

En la parte final de este capítulo se muestra la implementación del componente Android. En primer lugar se revisa y explica la forma de crear la interfaz gráfica de la aplicación y después se revisa la conexión entre el dispositivo y el servidor. También se revisa la parte de la recuperación de objetos JSON y por último se describe la etapa de presentación de resultados de la búsqueda.

5.1. Configuración de herramientas

En esta sección se describe de manera breve la instalación de las herramientas necesarias para el desarrollo y prueba de la aplicación. La primera parte tiene que ver con el componente de búsquedas Solr y la segunda parte son las herramientas principales para el desarrollo en la plataforma Android para el sistema operativo Linux.

5.1.1. MySQL

MySQL es un sistema para el manejo de bases de datos. Es necesario para administrar algunos datos de la aplicación y para realizar la indexación de los datos para el servidor Solr. A continuación se muestra la instalación y configuración de MySQL.

- **Instalación**

  Para instalar MySQL se utiliza el siguiente comando:

```
bash_ismael$ sudo aptitude install mysql-server5.1
```

Se puede instalar una aplicación con una interfaz gráfica para la administración de MySQL y otra llamada *Query Browser* para las consultas. Se instalan con el siguiente comando:

```
bash_ismael$ sudo aptitude install mysql-admin
```

La interfaz del administrador se puede ver en la Figura 5.1 y el Query Browser en la Figura 5.2.

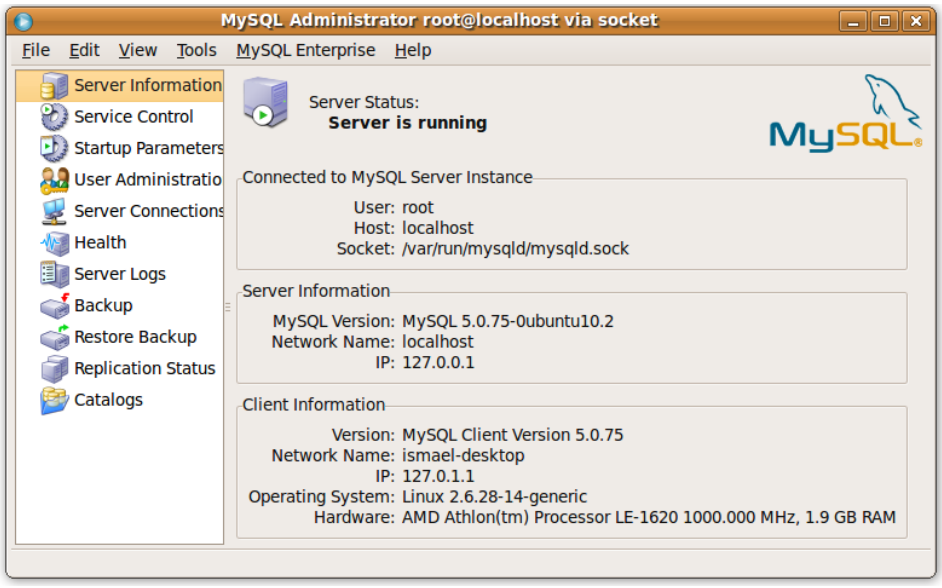

Figura 5.1: Administrador MySQL

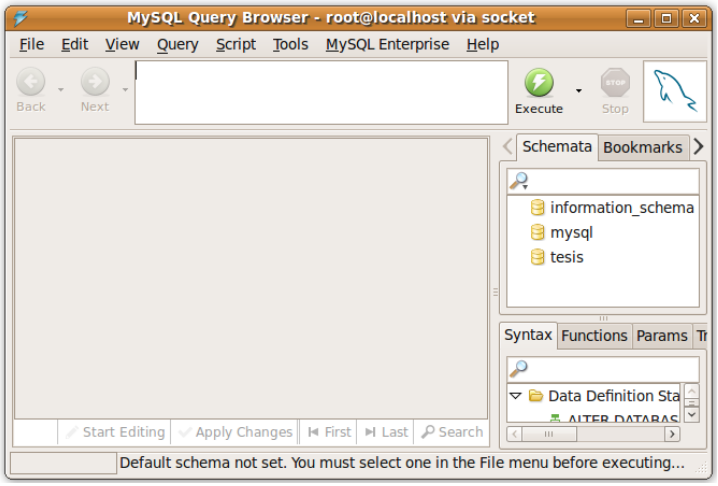

Figura 5.2: Query Browser

Se pueden ver más detalles de instalación en la página oficial de MySQL. [1]

[1] http://www.mysql.com/

- **Configuración**

  Una vez que se ha instalado el servidor MySQL, se debe crear la base de datos para la aplicación. Se utiliza el siguiente comando para iniciar la consola de MySQL escribiendo el password definido durante la instalación:

  ```
  bash_ismael$ mysql -p -u root
  ```

  Después de entrar se debe poner el comando para crear la base de datos. En este caso se crea una base con el nombre *tesis*:

  ```
  mysql> create database tesis;
  ```

  Para confirmar que se ha creado la base se utiliza:

  ```
  mysql> show databases;
  ```

  Es necesario crear un nuevo usuario y asignar los permisos en la base de datos creada. Esto se puede hacer de manera muy sencilla, a través de la herramienta de administración de MySQL en la vista *User Administration* con el botón *New User* y proporcionando los datos requeridos.

  En la Figura 5.3 se muestra la vista del administrador.

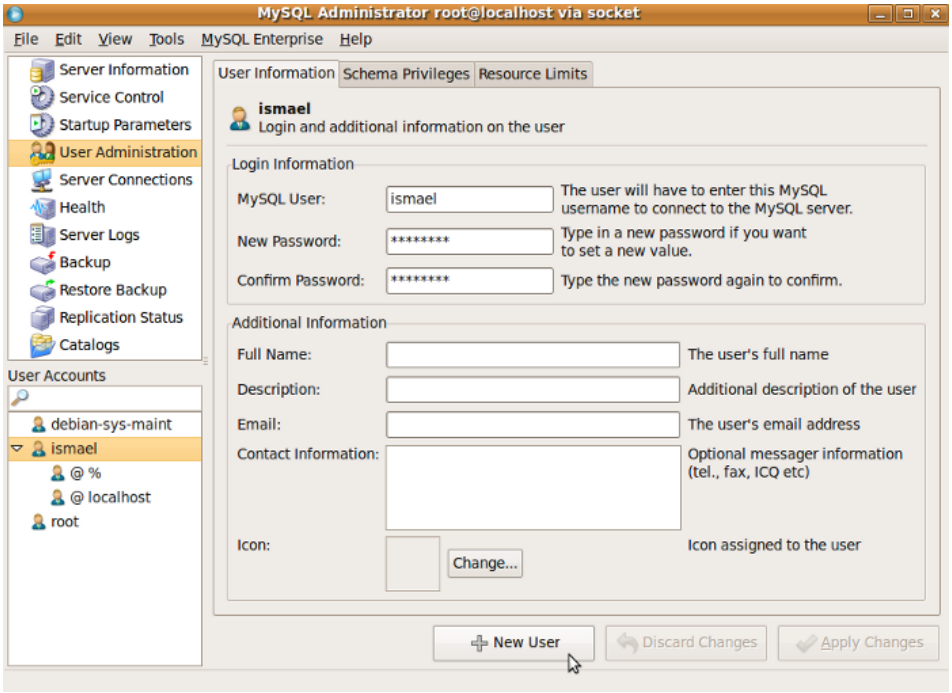

Figura 5.3: Nuevo usuario

- **Información**

  La información que se muestra en la aplicación fue obtenida de las páginas

`http://www.directorio.unam.mx/` y `http://www.pumabus.unam.mx/`. De la primera se obtuvo un subconjunto de los datos, los cuales son registros asociados a la *Facultad de Ciencias* y de la segunda página se tomaron los datos relativos a las referencias, paradas y rutas del transporte interno de la Universidad.

Se crearon dos script's[2] para MySQL: el primero contiene la información y los datos de cada registro, el segundo contiene las instrucciones para generar las tablas para la base de datos.

A continuación vemos un ejemplo de como se construye una de las tablas. Esto se define en el archivo **ScriptBaseDatos1.txt**:

```
CREATE TABLE rutas(
id_ruta MEDIUMINT NOT NULL AUTO_INCREMENT,
no_ruta VARCHAR(8) NOT NULL,
nombre_ruta VARCHAR(45) NOT NULL,
origen VARCHAR(20) NOT NULL,
distancia VARCHAR(10) NOT NULL,
no_paradas VARCHAR(2) NOT NULL,
PRIMARY KEY (id_ruta)
);
```

Para crear todas las tablas se debe utilizar el siguiente comando desde la consola:

```
bash_ismael$ mysql -p -u tesis < ScriptBaseDatos1.txt
```

Para cargar los datos en las tablas se debe utilizar el script que contiene la información. Esto se hace desde la consola de MySQL:

```
mysql> use tesis;
mysql> load data local infile "ScriptDatosRutas1.txt"
        into table rutas fields terminated by '*';
```

La información en los archivos de datos se define así:

```
*Ruta 1*Metro Universidad - Circuito Interior*Universidad*7.2 km*16
*Ruta 2*Metro Universidad - Circuito Exterior*Universidad*4.2 km*10
*Ruta 3*Metro Universidad - Zona Cultural*Universidad*7.2 km*15
.
.
.
```

Los datos son separados por * y se agregan en el orden de los campos definidos en la tabla correspondiente.

Con esto termina la configuración y la carga de información de prueba para la aplicación de búsqueda.

[2]Un script es un conjunto de instrucciones que permiten realizar tareas automatizadas

5.1.2. Java SDK

Para instalar el kit de desarrollo para Java se necesita descargar un archivo ejecutable de la página: `http://java.sun.com/javase/downloads/index.jsp`, se debe seleccionar el paquete *JDK+Java EE Bundle* y elegir la plataforma *Linux*.

- **Instalación**

  Al terminar la descarga del archivo se debe iniciar el instalador de la siguiente forma:

  ```
  bash_ismael$ ./java_ee_sdk_5_08-jdk-6u18-linux.bin
  ```

  En la Figura 5.4 se puede ver la interfaz gráfica del instalador.

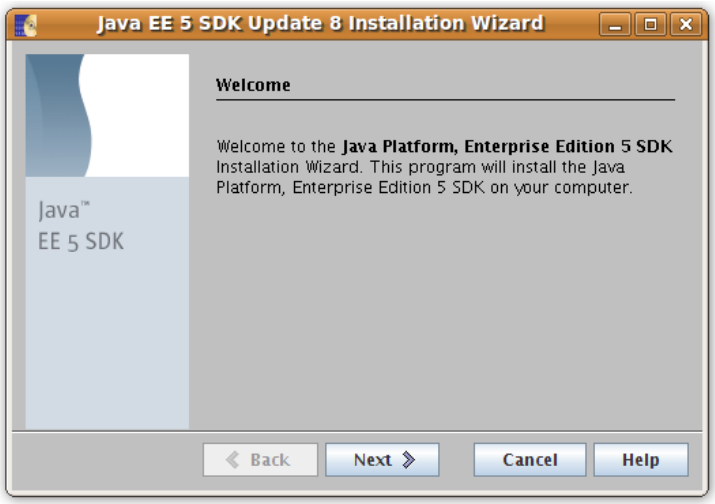

Figura 5.4: Instalación Java Sdk

Para consultar más detalles sobre la instalación se puede consultar la página oficial de Java. [3]

5.1.3. Tomcat

Para el componente Solr se necesita un contenedor de servlet's[4]. Tomcat implementa las especificaciones de Servlet y JSP y es utilizado para el desarrollo en esta tecnología.

- **Instalación**

  Para instalar Tomcat se necesita descargar un archivo empaquetado de la página del proyecto Apache Tomcat que es: `http://tomcat.apache.org/download-60.cgi`, se debe elegir la versión 6.x la distribución binaria. Una vez que ha finalizado la descarga se debe desempaquetar el archivo de la siguiente manera:

  ```
  bash_ismael$ unzip apache-tomcat-6.0.24.tar.gz
  bash_ismael$ tar xvf apache -tomcat-6.0.24.tar
  ```

[3] `http://java.sun.com/javase/6/webnotes/README.html`

[4] Los servlets son objetos que se ejecutan dentro del contexto de un servidor de aplicación que implementa la especificación J2EE.

Después de los comandos anteriores se genera un directorio llamado *apache -tomcat-6.0.24*. En este directiorio ya está lo necesario para utilizar Tomcat. Para más detalles sobre la instalación y configuración de Tomcat puede visitar su página oficial. [5]

5.1.4. Eclipse

Eclipse es el IDE que se utiliza para escribir el código de la aplicación y es la herramienta principal para la implementación del proyecto. En esta sección se muestra la instalación básica de Eclipse 3.5 Galileo y más adelante se explica la instalación del complemento para el desarrollo de aplicaciones en la plataforma Android.

- **Instalación**

  Para la instalación de Eclipse se debe descargar un archivo empaquetado, se debe elegir el paquete *Eclipse IDE for Java EE Developers* de la categoría *Enterprise* en la página: `http://www.eclipse.org/` y después seleccionar la plataforma *Linux 32-bit*.

  Al finalizar la descarga se utiliza el siguiente comando para extraer el contenido en el directorio por defecto de Eclipse:

```
bash_ismael$ unzip eclipse-jee-galileo-linux-gtk.tar.gz
bash_ismael$ tar xvf eclipse-jee-galileo-linux-gtk.tar
```

  Con este comando termina la instalación básica de eclipse. Para más detalles de la instalación se puede consultar la página oficial del proyecto Eclipse en `http://wiki.eclipse.org/Eclipse/Installation`.

- **Agregar Tomcat a Eclipse**

  Primero se debe activar la pestaña *Servers* en Eclipse, después ir al menú $Window \rightarrow ShowView \rightarrow Other \rightarrow Servers$.

  En la pestaña "Servers" se debe agregar un nuevo servidor con la opción *New Server*, indicar aquí *Tomcat 6.x* y el directorio de instalación.

  Una vez que aparezca Tomcat como servidor en la vista "Servers" se debe seleccionar y presionar la tecla *F3* para abrir la vista de propiedades, después poner en la pestaña *Modules* y agregar un nuevo módulo web con la siguiente información:

```
Document base:  tomcat_home/webapps/ROOT
Path: /
```

  Al terminar se debe guardar el archivo y estará listo para utilizar *Tomcat* desde Elicpse.

5.1.5. Solr

Como se comentó en la sección 4.2, Solr es el motor de búsqueda para poder realizar la indexación de los datos desde la base de datos que se ha creado para las pruebas.

- **Instalación**

  - Para instalar Solr es necesario descargar un archivo empaquetado de la siguiente página: `http://www.apache.org/dyn/closer.cgi/lucene/solr/`. Al terminar la descarga se utiliza el siguiente comando para desempaquetar el archivo:

[5] `http://tomcat.apache.org/tomcat-6.0-doc/index.html`

```
bash_ismael$ tar xvfz apache-solr-1.4.0.tgz
```

El comando anterior genera un directorio donde se encuentran los archivos necesarios para utilizar Solr.

Para más detalles de la instalación y configuración visite la página oficial: `http://lucene.apache.org/solr/tutorial.html`

Para la indexación es necesario tener el conector jdbc para MySQL. El conector se obtiene de la siguiente página: `http://dev.mysql.com/downloads/connector/j/`. Al finalizar la descarga se debe desempaquetar y colocar el archivo .jar dentro del directorio *apache-solr-1.0.4/lib/*

5.1.6. SDK Android

Como se explica en la sección 4.3, se ofrece un kit de desarrollo para la plataforma Android el cual contiene herramientas para el proceso de implementación de aplicaciónes para la plataforma. Por ejemplo el emulador del dispositivo o el *monitor de depuración dalvik* entre otras.

- **Instalación**

  Para la instalación del SDK se necesita elegir el paquete *Starter* para la plataforma linux y descargarlo de la siguiente página: `http://developer.android.com/sdk/index.html`

  Se debe desempaquetar el archivo al finalizar la descarga como se muestra a continuación:

  ```
  bash_ismael$ tar xvfz android-sdk_r04-linux_86.tgz
  ```

  Se crea un directorio donde se encuentran todas las herramientas necesarias para el desarrollo. Para más información acerca de la instalación y configuración visite la página: `http://developer.android.com/sdk/index.html`

- **Configuración**

  Después de instalar el SDK es necesario actualizar la especificación a una más reciente. En este caso a la 2.1, y para hacer esto se tiene que ejecutar la herramienta *Android SDK and AVD Manager* de la siguiente manera:

  ```
  bash_ismael$ cd android-sdk-linux_86/tools
  bash_ismael$ android
  ```

  Después de seleccionar y descargar los paquetes en la vista *Available Packages* el resultado se puede ver en la Figura 5.5:

Figura 5.5: Actualización Android Sdk

5.1.7. ADT

ADT (Android Development Tools) es el complemento de Android para Eclipse. Este complemento permite integrar las herramientas para la construcción de aplicaciones Android en el entorno de desarrollo Eclipse Galileo.

- **Instalación**

  Para comenzar con la instalación se debe iniciar Eclipse y seleccionar
  $Help \rightarrow InstallNewSoftware$, en la vista *Available Software* se debe presionar *Add...* y escribir los siguientes datos:

  ```
  Name:       Android Plugin
  Location:   https://dl-ssl.google.com/android/eclipse/
  ```

  Ahora se debe seleccionar *Available Software* y en la lista debe aparecer *Developer Tools*, se debe seleccionar y poner en *Next*. Después se acepta la licencia y se da click en *Finish*. Al terminar se debe reiniciar Eclipse.

  En las preferencias de Eclipse se debe indicar el directorio donde está el SDK de Android. Se debe seleccionar $Window \rightarrow Preferences...$ en el panel izquierdo seleccionar "Android", click en "Browse" para indicar el directorio, click en "Apply" y con esto está listo para trabajar el complemento de Android en Eclipse Galileo.

  Para más información acerca de la instalación se puede revisar la página oficial en: `http://developer.android.com/sdk/eclipse-adt.html`.

- **Configuración**

  Es necesario crear un *AVD(Android Virtual Device)* para poder utilizar el emulador. En Eclipse se debe elegir el botón "Opens the Android SDK and AVD Manager" como se muestra en la Figura5.6:

  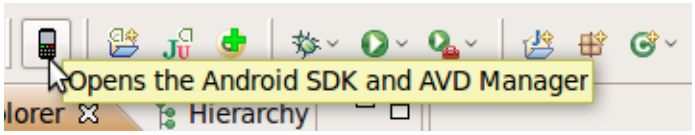

  Figura 5.6: Crear un AVD Paso 1

  En la vista "Virtual Devices" presionar el botón "New..." para crear un AVD. En la Figura 5.7 se muestra la pantalla para ingresar los datos como nombre. El target que debe ser "Google APIs(Google Inc) Level 7" y otras propiedades como la resolución de pantalla o el tamaño de la tarjeta SD.

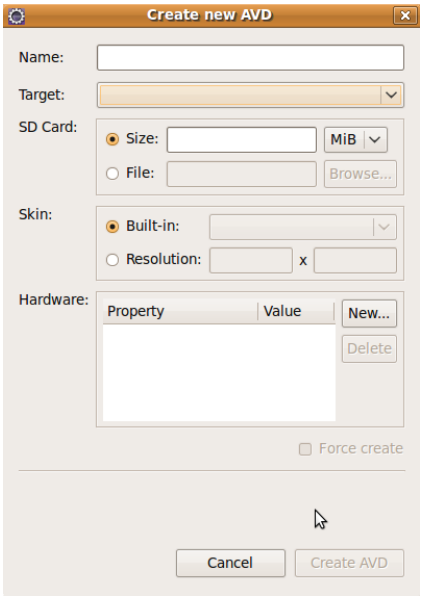

Figura 5.7: Crear un AVD Paso 2

La Figura 5.8 muestra el AVD listo para utilizarse:

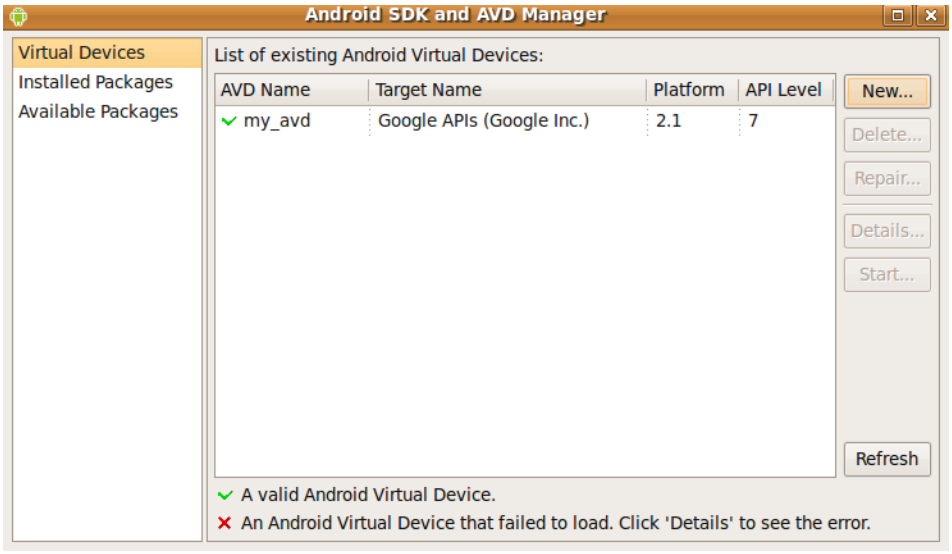

Figura 5.8: Crear un AVD Paso 3

5.2. Desarrollo del componente SolrJ

En esta sección se muestra el proceso de indexación de datos al motor de búsqueda y los archivos de configuración principales para realizar esta tarea. Además se revisan los métodos principales en el código fuente de la parte de búsqueda en Solr y, por último, se explica la manera en que se implementan los servlet para cada una de las tareas que se definieron en el diseño general4.

5.2.1. Indexación

Como se menciona en la sección 2.4.2.1, para trabajar en Solr es necesario tener un directorio *solr/home* en el que se deben especificar los *core* o índices con los que trabajá el motor de búsquedas. Para el proyecto "Tlatemoani" se tiene la siguiente estructura de directorios y archivos:

```
home/solr1.4.0/example/tesis
        solr.xml
        directorio/
                conf/
                        data-config.xml
                        schema.xml
                        solrconfig.xml
        paradas/
                conf/
                        data-config.xml
                        schema.xml
                        solrconfig.xml
        referencias/
                conf/
                        data-config.xml
                        schema.xml
                        solrconfig.xml
        rutas/
                conf/
                        data-config.xml
                        schema.xml
                        solrconfig.xml
```

Revisamos el contenido del archivo *solr.xml* que contiene las referencias a los *core's* en el directorio *tesis*:

```
<?xml version='1.0' encoding='UTF-8'?>
<solr persistent='true'>
        <cores adminPath='/admin/cores'>
                <core name='Directorio' instanceDir='directorio/'/>
                <core name='Rutas' instanceDir='rutas/'/>
                <core name='Paradas' instanceDir='paradas/'/>
                <core name='Referencias' instanceDir='referencias/'/>
        </cores>
</solr>
```

Ahora revisamos los archivos en el directorio config/ para el índice Rutas:

- **solrconfig.xml**

  En este archivo no se realizan modificaciones. Solo se toma la configuración por defecto. Es necesario que este archivo esté en el directorio *config/* del *core* correspondiente.

- **data-config.xml**

  En este archivo se define el *DataSource* que se refiere a la fuente de datos que se deberá indexar. En este caso se define el servidor de base de datos, el conector JDBC, un nombre de usuario y un password. Además se define una consulta para la información que será indexada. El archivo es el siguiente:

```
<dataConfig>
    <dataSource driver="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/tesis"
        user="ismael" password="123456789" />
    <document name="tabla_rutas">
        <entity name="rutas" query="select id_ruta,
                no_ruta as Numero_Ruta,
                nombre_ruta as Nombre_Ruta,
                origen as Origen,
                distancia as Distancia,
                no_paradas as Numero_Paradas
                from rutas;">
        </entity>
    </document>
</dataConfig>
```

- **schema.xml**

  En este archivo se definen los tipos de datos para cada valor de los campos por ejemplo se define un tipo de campo llamado "text_ws", que permite tener valores divididos por espacios en blanco. Para hacer esto se utiliza un Analyzer como se mencionó en la sección 2.4.1.2. Los analizadores se definen de la siguiente manera:

  ```
  <fieldType name="text_ws" class="solr.TextField"
      positionIncrementGap="100">
      <analyzer>
          <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
  </fieldType>
  ```

  Después de la definición de los tipos, se definen los campos indicando el nombre, el tipo, si se indexa, si se almacena y si es requerido. En el siguiente código se muestra un ejemplo de la definición de 3 campos:

  ```
  <field name="id_ruta" type="integer" indexed="true" stored="false"
      required="true" />
  <field name="Numero_Ruta" type="text_ws" indexed="true" stored="true" />
  <field name="Nombre_Ruta" type="text_ws" indexed="true" stored="true" />
  ```

  También se debe incluir una etiqueta para la unicidad de documentos, un campo de búsqueda por defecto y un operador por defecto para las consultas. Además para esta aplicación se requiere tener una copia de campos para que la búsqueda se realice entre los diferentes campos sin especificar uno explícitamente y otra copia para la interfaz de usuario en Android. El propósito de este campo se explicará más adelante.

  ```
      <uniqueKey>id_ruta</uniqueKey>
      <defaultSearchField>todos</defaultSearchField>
      <solrQueryParser defaultOperator="OR"/>
      <copyField source="Nombre_Ruta" dest="todos"/>
      <copyField source="Numero_Ruta" dest="todos"/>
      <copyField source="Numero_Paradas" dest="todos"/>
      <copyField source="Nombre_Ruta" dest="zztitulo"/>
  ```

Es encesario iniciar Solr para indexar los datos de la base. Utilizamos un comando para iniciar el servidor utilizando el siguiente comando:

```
bash_ismael$ cd apache-solr-1.0.4/example
bash_ismael$ java -Dsolr.solr.home="tesis" -jar start.jar
```

Después de tener la definición de todos los archivos de configuración en los *core's* es necesario utilizar un servlet que permite importar los datos desde la base de datos e indexar el contenido según la configuración. Para hacer esto se debe poner la siguiente dirección en un navegador web:

```
http://localhost:8983/solr/Rutas/dataimport?command=full-import
```

Esto se hace para cada uno de los *core's* definidos en el archivo solr.xml en el directorio *tesis/* dentro del directorio principal de Solr. Al terminar la indexación de cada tabla de la base de datos, se crea un directorio data/ en el que se encuentra la información que ha sido indexada. Este directorio se encuentra en el directorio principal de cada *core*.

De esta manera se termina la indexación de los datos de prueba para el componente Solr.

5.2.2. Índices

Como se describe en el Capítulo 4, se implementa la clase *Indice* y *Campo*. Estas clases permiten manejar el concepto de índice en los servlets del componente Solr. La implementación de estas clases es muy simple, ya que solo contienen atributos de clase y los métodos de acceso *get y set* para el manejo del estado de los objetos.

La clase ConexionSQL permite la conexión a la base de datos por medio del conector jdbc de MySQL. La clase define los siguientes métodos:

- estaRegistrado(String usuario, String clave) – Verifica que un usuario este registrado en la base de datos y regresa el valor "true" en caso de que esté registrado y en otro caso regresa "false".
- getIndicesString() – Regresa una lista de objetos Indice con la información en la base de datos.
- getCamposString(int indice) – Regresa una lista de campos asociados al índice indicado con la información de la base de datos.

La clase *ControladorIndices* define un método que sirve para construir los índices con la información de la base de datos. El método es el siguiente:

```
public ArrayList<Indice> construyeIndices(){
  ArrayList<ArrayList<String>> listaIndices = bc.getIndicesString();
    for(int i=0;i<listaIndices.size();i++){
      ArrayList<String> camposIndice = listaIndices.get(i);
      Indice auxIndice = new Indice();
      auxIndice.setId(Integer.parseInt(camposIndice.get(0)));
      auxIndice.setNombre(camposIndice.get(1));
      auxIndice.setCore(camposIndice.get(2));
      auxIndice.setEstado(Integer.parseInt(camposIndice.get(3)));

      ArrayList<ArrayList<String>> listaCampos =
        bc.getCamposString(auxIndice.getId());
      ArrayList<Campo> nuevaListaCampos = new ArrayList<Campo>();
      for(int j=0;j<listaCampos.size();j++){
        ArrayList<String> campos = listaCampos.get(j);
        Campo campo = new Campo();
        campo.setId(Integer.parseInt(campos.get(0)));
        campo.setNombre(campos.get(1));
        campo.setNombreVista(campos.get(2));
        campo.setVisible(Integer.parseInt(campos.get(3)));
```

```
        campo.setEstado(Integer.parseInt(campos.get(4)));
        campo.setId_indice(Integer.parseInt(campos.get(5)));
        nuevaListaCampos.add(campo);
      }
      auxIndice.setCampos(nuevaListaCampos);
      this.indices.add(auxIndice);
    }
    return this.indices;
}
```

Esta clase tiene como atributo un objeto de la clase ConexionSQL que permite la conexión a la base de datos y una lista de objetos *Indice*.

5.2.3. Ejecutor

La clase *EjecutorQuery* se utiliza para ejecutar la consulta en el servidor Solr en cada uno de los índices y regresa una respuesta por medio del objeto *QueryResponse*. También define una constante para regresar el número máximo de coincidencias para un conjunto de resultados.

El siguiente método realiza la consulta al servidor Solr y recibe como parámetro un objeto *Indice* del cual se obtiene su core para poder realizar la consulta.

```
public void ejecutaQuery(Indice indice,String query){
        String core = indice.getCore();
        SolrQuery solrQuery = new SolrQuery(query);
        solrQuery.setRows(this.numRegistros);
        this.servidorSolr.setBaseURL(core);
        this.queryResponse = new QueryResponse();
        try{
                queryResponse = this.servidorSolr.query(solrQuery);
        } catch(SolrServerException e){
                e.printStackTrace();
        }
}
```

5.2.4. Resultados

Para obtener los resultados se implementa la clase *ResultadoQuery* para manejar los resultados. En este caso solo regresa un objeto *SolrDocumentList* que contiene el resultado de la consulta y otro método que obtiene información acerca del número de coincidencias.

```
public void generaSolrDocumentList(QueryResponse queryResponse){
        this.sdl = queryResponse.getResults();
}

y

public long getNumResultados(){
        return this.sdl.getNumFound();
}
```

5.2.5. Servlet

Se implementan dos servlets: el primero se llama “accesoServlet” y el segundo es “indicesServlet”. El primer servlet se utiliza para verificar el acceso del usuario desde el dispositivo móvil y recibe como

parámetro el nombre de usuario y una clave. Con estos datos verifica el registro en la tabla "acceso" de la base de datos y regresa un objeto *JSON* que representa un valor booleano. Esto se hace con el siguiente método:

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException{
String usuario = "";
String clave = "";
JSONObject json = new JSONObject();
try{
        usuario = request.getParameter("usr");
        clave = request.getParameter("clv");
}
catch(Exception e){
        e.printStackTrace();
}
//se define el tipo de respuesta
response.setContentType(CONTENT_TYPE);
PrintWriter out = response.getWriter();
if(usuario != null && clave !=null){
        boolean respuesta= false;
        //se utiliza un metodo de la clase ConexionSQL
        respuesta = (csql.estaRegistrado(usuario, clave))?true:false;
        //se regresa el resultado JSON como una cadena
        json.put("acceso", respuesta);
        out.println(json);
}
out.close();
}
```

En el segundo servlet se implementa el manejo de la información obtenida de Solr. Para esto se definen tres métodos principales para cada conjunto de datos que se necesita para la interfaz en Android. Los métodos son los siguientes:

- **regresaIndices()**

  Regresa un objeto *JSON* que contiene el nombre del índice y su id, por ejemplo, { "Rutas":1 }

- **regresaResultadosIndices(indicesBusqueda,query)**

  Regresa un objeto JSON que contiene el nombre del índice y el número de coincidencias de la búsqueda, por ejemplo { "Rutas":17, "Directorio":2 }

- **regresaResultado(idIndice,query)**

  Regresa un objeto *JSON* que contiene los datos de los resultados. Es una pareja con el título "resultados" como llave y un *JSONArray* como valor, en el que se definen los objetos *JSON* para cada registro, por ejemplo { "Nombre_Ruta":"Ruta 1","Paradas":21, ... }

El siguiente método maneja los parámetros para regresar los objetos *JSON* con la informacón que se requiere según el parámetro "metodo":

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException{
  int metodo = 0;
  String query = "";
```

```
  String indices = "";
  String i = "";
  int indice = 0;
  try{
    //se recuperan los datos de la solicitud
    String m = request.getParameter("mtd");
    metodo = Integer.parseInt(m);
    query = request.getParameter("qry");
    indices = request.getParameter("idxs");
    i = request.getParameter("idx");
    if(i != null){
      indice = Integer.parseInt(i);
    }
  }
  catch(Exception e){
    e.printStackTrace();
  }
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();
  //segn el mtodo se realizan las funciones
  switch(metodo){
    case 1:
      //regresa la informacin de ndices disponibles
      JSONObject json1 = this.regresaIndices();
      out.println(json1);
      break;
    case 2:
      //regresa la informacin de coincidencias
      JSONObject json2 = new JSONObject();
      if(!query.isEmpty() && !indices.isEmpty()){
        //pone los indices de busqueda segn el
        //string que se pasa como parametro
        ArrayList<Indice> indicesBusqueda = convierteIndices(indices);
        json2 = this.regresaResultadosIndices(
        indicesBusqueda,query);
         }
      out.println(json2);
      break;
    case 3:
      //se regresan los resultados de la bsqueda
      JSONObject json3 = new JSONObject();
      if(!query.isEmpty() && !i.isEmpty()){
        json3 = this.regresaResultado(indice,query);
      }
      out.println(json3);
      break;
    default:
      break;
  }
  out.close();
}
```

Para recibir el número de índices seleccionados desde el dispositivo se genera una lista de objetos *Indice* a partir de una cadena de texto que contiene el id del índice seleccionado y un separador. En este caso

es el carácter “a”. Esto se hace con el siguiente método:

```
public ArrayList<Indice> convierteIndices(String indices){
       //divide la cadena utilizando el caracter 'a' como separador
  StringTokenizer st = new StringTokenizer(indices,"a");
  ArrayList<Indice> indicesBusqueda = new ArrayList<Indice>();
  while(st.hasMoreTokens()){
            int indice = Integer.parseInt(st.nextToken());
    //se revisa que sea el indice
    for(int k=0;k<this.cr.getIndices().size();k++){
              if(cr.getIndices().get(k).getId() == indice){
                indicesBusqueda.add(cr.getIndices().get(k));
              }
            }
  }
  return indicesBusqueda;
}
```

5.3. Desarrollo del componente Android

Como se mencionó anteriormente, en esta sección se describe la implementación de la interfaz de usuario para el sistema. En primer lugar se describe el proceso para iniciar un proyecto Android con Eclipse y como generar una pantalla en Android por medio de una actividad y un recurso. Después se muestra la clase principal para realizar la conexión al componente Solr y por último se describe el manejo de los objetos *JSON* y la clase que permite mostrar los resultados en detalle al usuario.

5.3.1. Interfaz de usuario

Para el desarrollo con Android es necesario crear un proyecto como plantilla. En Eclipse se elige *File* → *New* → *Other* → *Android* → *Android* Project, en la Figura5.9 se puede ver la interfaz para crear un nuevo proyecto Android:

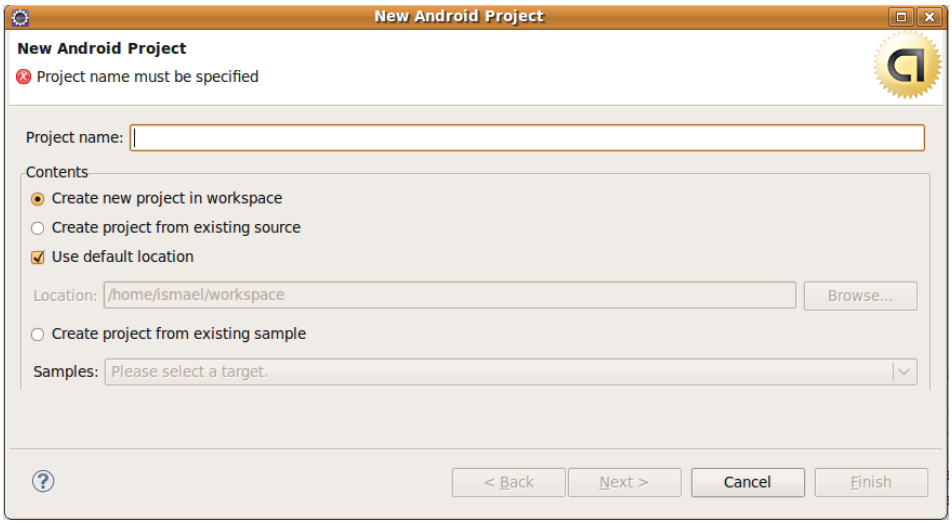

Figura 5.9: Nuevo proyecto Android

La estructura que se menciona en la sección 2.3.3 se genera al crear un nuevo proyecto, se puede ver a continuación:

```
tesisAndroid/
        src/
        gen/
                R.java
        assets/
        res/
                drawable-hdpi/
                drawable-ldpi/
                drawable-mdpi/
                layout/
                values/
AndroidManifest.xml
default.properties
```

Para construir la interfaz gráfica se utilizan dos técnicas diferentes: una es la construcción de la interfaz por medio de archivos XML en la sección 2.3.2.2 y la otra es como se hace comúnmente, es decir, a través de clases en el lenguaje de programación.

A continuación se muestra el archivo XML que define la interfaz en la pantalla de inicio:

```
<AbsoluteLayout android:id="@+id/AbsoluteLayout01" android:layout_width="fill_parent"
android:layout_height="fill_parent" xmlns:android=
"http://schemas.android.com/apk/res/android">

<ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/iv_logotipo" android:src="@drawable/logo_principal"
android:layout_y="50px" android:layout_x="62px">
</ImageView>

<TextView android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_x="40px" android:layout_y="150px" android:id="@+id/tv_login"
  android:text="Usuario:">
  </TextView>

<EditText android:layout_width="wrap_content" android:id="@+id/et_login"
android:layout_x="125px" android:width="150px" android:layout_height="35px"
android:layout_y="143px" android:textSize="6pt" android:text="u">
</EditText>

<TextView android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_x="40px" android:id="@+id/tv_pass" android:text="Clave:"
android:layout_y="210px">
</TextView>

<EditText android:layout_width="wrap_content" android:layout_x="125px"
android:layout_height="35px" android:width="150px" android:id="@+id/et_pass"
android:layout_y="202px" android:typeface="normal" android:visibility="visible"
android:inputType="textPassword" android:textSize="6pt" android:text="c">
</EditText>

<Button android:layout_width="wrap_content" android:id="@+id/boton" android:text=
"Aceptar"
android:layout_x="120px" android:layout_height="35px" android:layout_y="260px">
</Button>
</AbsoluteLayout>
```

En primer lugar se debe definir un contenedor para los demás elementos llamado "Layout". En este caso tenemos definido un layout que permite definir la posición exacta de los elementos que contiene y el identificador del layout se indica a través de su propiedad id.

Los elementos en la pantalla se agregan como parte del Layout. Se especifica el identificador, la posición y en algunos casos otras propiedades, por ejemplo, en el caso del elemento *ImagewView* se indica el recurso de la imágen y en el *EditText* se puede definir el tamaño de letra o el tipo de texto de entrada.

Una vez que se ha definido el recurso *Layout* es necesario asignarlo como parte de la actividad que maneje la lógica de la pantalla. A continuación vemos un ejemplo del uso de un recurso *Layout* en una clase *Activity* en Android. La clase *Inicio* en la aplicación *Tlatemoani* permite al usuario escribir su nombre de usuario y clave para acceder a la función de búsqueda. La interfaz se genera a partir del *Layout* definido en el archivo *inicio.xml*.

```
public void onCreate(Bundle bundle){
 super.onCreate(bundle);
 //se carga el layout
          setContentView(R.layout.inicio);
          // obtenemos el boton
          Button boton = (Button)findViewById(R.id.boton);
          //se define un nuevo listener para los eventos del boton
          boton.setOnClickListener( new OnClickListener(){
           public void onClick(View view){
            //se obtienen las cajas de texto
                 EditText usr = (EditText) findViewById(R.id.et_login);
                 EditText pass = (EditText) findViewById(R.id.et_pass);
                 . . .
```

Todos los datos que son creados o definidos en archivos xml, imágenes, strings y otros, son generados automáticamente como recursos en la clase *R.java*. A continuación vemos un fragmento del código de la clase:

```
package unam.fciencias.isma.tesis;

public final class R {
     public static final class attr {
     }
     public static final class drawable {
             public static final int icon=0x7f020000;
             public static final int logo_principal=0x7f020001;
     }
     public static final class id {
             public static final int AbsoluteLayout01=0x7f050006;
                 public static final int ButtonBusqueda1=0x7f050005;
                 public static final int EditTextBusqueda1=0x7f050004;
                 public static final int ImageView01=0x7f05000d;
                 public static final int LinearLayout01=0x7f050000;
                public static final int absoluteLayoutBusqueda1=0x7f050002;
                public static final int boton=0x7f05000c;
. . .
```

Cuando es necesario crear la interfaz de manera dinámica se utilizan las clases que proporciona el lenguaje de programación. Son clases que implementan el funcionamiento de los elementos visuales

que pueden manejarse en la pantalla, por ejemplo texto, imágenes, cajas de texto, botones y otros elementos.

En la pantalla de selección de índices se deben generar los elementos checkbox para cada índice de búsqueda disponible. Esto debe generarse desde código según el número de índices. En el siguiente fragmento de código se puede ver como se construye la interfaz gráfica de la pantalla de índices de selección con la clase *Indices* utilizando un objeto *LinearLayout* y el número necesario de objetos *CheckBox*.

```
//se genera el layout
contenedor = new LinearLayout(this);
contenedor.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.FILL_PARENT));
contenedor.setOrientation(LinearLayout.VERTICAL);
. . .
//genera los checkbox de los indices que esten activos
for(int i=0;i<Datos.getIndices().size();i++){
        CheckBox cb1 = new CheckBox(this);
        cb1.setText(Datos.getIndices().get(i).getNombre());
        if(Datos.getIndices().get(i).isSeleccionado()){
        cb1.setChecked(true);
}
. . .
contenedor.addView(cb1);
}
setContentView(contenedor);
. . .
```

5.3.2. Conexión HTTP

Antes de crear la conexión con el servlet dentro de la clase *ConexionServlet* se define un método que permite convertir un *InputStream* a una cadena de texto. Los objetos *InputStream* son bytes de los que se puede leer información. En este caso es una representación de datos en formato *JSON*.

El siguiente método define la conversión en una cadena la cual representa el objeto *JSON* que se obtiene desde el servlet.

```
private static String conversionString(InputStream is){
                if(is != null){
                BufferedReader br = new BufferedReader(
                new InputStreamReader(is));
                StringBuilder sb = new StringBuilder();
                String linea = null;
                try{
                        while ((linea = br.readLine()) != null){
                                sb.append(linea).append("\n");
                        }
                } catch (IOException e){
                e.printStackTrace();
                } finally {
                        try {
                                is.close();
                        } catch (IOException e){
                                e.printStackTrace();
                        }
```

```
                }
                return sb.toString();
                }
                else{
                return "";
                }
    }
```

En el método de conexión se recibe un *URL* que define la dirección del servlet y los parámetros que se pasan a través del método *GET*. Este URL se forma en la clase *ControladorRespuesta*.

En el siguiente método se muestra la forma de realizar la petición al servlet a través de un objeto *HttpClient* y un objeto *HttpGet*. Un objeto de la clase *HttpEntity* permite recuperar el contenido de la respuesta en un objeto de la clase *InputStream* y utilizando el método *conversionString* se obtiene un objeto del tipo String para crear el objeto *JSON* que se utilizará posteriormente.

```
public static JSONObject conexion(String url){
     JSONObject json = new JSONObject();
     try{
         HttpClient httpclient = new DefaultHttpClient();
        //crea un objeto para la peticin GET
        HttpGet httpget = new HttpGet(url);
        //para la respuesta
        HttpResponse response;
        response = httpclient.execute(httpget);
            //obtiene la entidad de respuesta
            HttpEntity entity = response.getEntity();
            if(entity != null){
             //se obtiene un objeto JSON como String
                InputStream instream = entity.getContent();
                String resultado = conversionString(instream);
                //se crea un JSONObject
                json = new JSONObject(resultado);
                //cierra el InputStream
                instream.close();
            }
        } catch(ClientProtocolException e){
            e.printStackTrace();
        } catch(IOException e){
            e.printStackTrace();
        } catch(JSONException e){
            e.printStackTrace();
        }
        return json;
    }
```

5.3.3. JSON

El motor de búsquedas Solr permite obtener como formatos de respuesta diferentes tipos, por ejemplo XML, PHP, Ruby entre otros. El lenguaje XML es más complejo que el formato JSON, es por eso que no se utiliza XML como formato para el transporte de los datos. Los formatos de salida en PHP y Ruby no se utilizan por que se tiene que implementar algun componente que conecte los dos lenguajes, por ejemplo para PHP y Java.

Los objetos *JSON* que se obtienen por medio de la clase *ConexionServlet* son manejados con la clase

ControladorRespuesta. Además esta clase es la que define los URL de petición para los servlets, como se muestra en el siguiente fragmento de código:

```
private final String URL_VERIFICA_ACCESO =
"http://10.0.2.2:8080/tesisServlet/accesoServlet";
private final String URL_INDICES_BUSQUEDA =
"http://10.0.2.2:8080/tesisServlet/indicesServlet";
. . .
```

En el caso de los *URL* se puede ver que la dirección IP no es **localhost**, esto es porque Android define un alias para referirse a la máquina local cuando se realiza una comunicación HTTP, este alias es la dirección **10.0.2.2** que aparece en el código. Se utiliza para las aplicaciones en Android que se conectan a localhost.

En cada uno de los cuatro métodos de esta clase se completa la construcción del *URL* con los parámetros como se muestra en el siguiente fragmento del método *obtieneIndicesBusqueda(int metodo)*:

```
. . .
String params = "?mtd="+metodo;
JSONObject json = ConexionServlet.conexion(this.URL_INDICES_BUSQUEDA+params);
. . .
```

Después de tener el *URL* se utiliza el método *conexion(String url)* de la clase *ConexionSQL* que regresa un objeto *JSON*. Para cada uno de los métodos siguientes se obtienen los valores y llaves, después se asignan a los atributos de la clase estática *Datos*.

- **verificaAcceso(String usuario, String clave)**

  En este método se verifica el acceso a la aplicación, por medio del servlet "accesoServlet", se recibe el objeto *JSON* y se asigna el valor booleano al atributo *acceso* a través del método *setAcceso* de la clase *Datos*.

- **obtieneIndicesBusqueda(int metodo)**

  Este método se utiliza para obtener los índices de búsqueda disponibles, se obtienen por medio de una petición al servlet. Al recibir el objeto *JSON* se obtiene una lista de objetos *Indice* que se asigna al atributo *indices* de la clase *Datos*.

- **obtieneResultados(int metodo,String nquery,String indices)**

  En este método se obtine el número de coincidencias de cada índice revisando el parámetro con el nombre del índice en la clase *Datos*.

- **obtieneResultadosDetalle(int metodo,String nquery,String indice)**

  En este método genera una lista de objetos *Resultado* utilizando el objeto *JSON*, se obtiene un *JSONArray* con la llave "resultados" y se recorre el arreglo *JSON* para obtener cada uno de los *JSONObject* que representan una coincidencia dentro del conjunto de resultados. Después de obtener los objetos *Resultado* se asignan a la clase *Datos*.

A continuación se muestra el método completo que se utiliza para obtener la lista de resultados en detalle:

```
public void obtieneResultadosDetalle(int metodo,String nquery,String indice){
  String query = URLEncoder.encode(nquery);
  //se definen los parametros para el servlet
  String params = "?mtd="+metodo+"&qry="+query+"&idx="+indice;
  //se obtiene el objeto JSON
```

```
  JSONObject json = ConexionServlet.conexion(this.URL_INDICES_BUSQUEDA+params);
  //se genera un arreglo de objetos de tipo Resultado
  ArrayList<Resultado> listaResultados = new ArrayList<Resultado>();
  try{
    JSONArray jsona = json.getJSONArray("resultados");
    for(int i=0;i<jsona.length();i++){
      JSONObject jsonAux = jsona.getJSONObject(i);
      Iterator<?> it = jsonAux.keys();
      Resultado resultado = new Resultado();
      String contenido="";

      while(it.hasNext()){
        try{

          String llave = (String)it.next();
          String valor= (String)jsonAux.get(llave);
          //se pone el titulo con el valor de la llave
          //que se indic en la iindexacin de la tabla
          if(llave.equals("zztitulo")){
            resultado.setTitulo(valor+" (ver mas)");
          }
          else{
            //se genera el contenido como una lista de
            //parejas, llave:valor
            contenido += llave+" :\n   "+valor+"\n";
          }
        }catch (JSONException e){
        e.printStackTrace();
        }
      }
      //se asignan los valores en Datos para las actividades
      resultado.setVisible(false);
      resultado.setContenido(contenido);
      listaResultados.add(i,resultado);
    }
    Datos.setResultados(listaResultados);
  }
  catch(JSONException e){
  e.printStackTrace();
}
}
```

En el código anterior se puede ver que se revisa si la llave del objeto *JSON* es igual a la cadena "zztitulo" como se mencionó en la sección 5.2.1. Este campo es una copia de otro y se utiliza para asignar ese otro campo como título para la interfaz. Cuando los datos son asignados el campo "zztitulo" se utiliza como título y los otros como el contenido del objeto *Resultado*.

5.3.4. Presentación de resultados

Como se mencionó en la sección anterior, para guardar los datos de los resultados obtenidos se utiliza una clase llamada *Resultado*. En esta clase se definen tres atributos: titulo, contenido y visible.

- **titulo** – Permite asignar un título al resultado.
- **contenido** – Permite asignar el contenido como una lista de parejas llave–valor.

- **visible** – Permite manejar el estado en la lista desplegable.

La clase que presenta los resultados finales se llama *ResultadosBusquedaDetalle* y hereda de *ListActivity* lo que permite crear la lista de resultados. Se debe crear un adaptador por medio de la clase *BaseAdapter*. Esta clase implementa las funciones de un adaptador.

A continuación se muestra la clase interna *AdaptadorLista*:

```
private class AdaptadorLista extends BaseAdapter{

  private Context contexto;

  /**
  * Constructor de Clase
  * @param contexto objeto Context
  */
  public AdaptadorLista(Context contexto){
    this.contexto = contexto;
  }

  /**
  * @see android.widget.Adapter#getCount()
  */
  @Override
  public int getCount(){
    return Datos.getResultados().size();
  }

  /**
  * @see android.widget.Adapter#getItem(int)
  */
  @Override
  public Object getItem(int posicion){
    return posicion;
  }

  /**
  * @see android.widget.Adapter#getItemId(int)
  */
  @Override
  public long getItemId(int posicion){
    return posicion;
  }

  /**
  * @see android.widget.Adapter#getView(int, android.view.View, android.view.ViewGroup)
  */
  @Override
  public View getView(int posicion, View v, ViewGroup padre){
    ContenidoVista cV;
    if (v == null){
      //se crea la vista por medio de la clase privada
      //con los datos estticos de la clase Datos
      cV = new ContenidoVista(contexto, Datos.getResultados().get(posicion).getTitulo(),
      Datos.getResultados().get(posicion).getContenido(),
```

```
      Datos.getResultados().get(posicion).isVisible());
    } else{
      //si no se puede obtener el contexto
      cV = (ContenidoVista)v;
      cV.setTitulo(Datos.getResultados().get(posicion).getTitulo());
      cV.setContenido(Datos.getResultados().get(posicion).getContenido());
      cV.setDesplegado(Datos.getResultados().get(posicion).isVisible());
     }
    return cV;
  }

  /**
  * Este metodo cambia el estado del
  * contenido. lo que permite quse se
  * haga visible.
  * @param posicion posicin en la lista
  */
  public void toggle(int posicion){
    Datos.getResultados().get(posicion).
    setVisible(!Datos.getResultados().
    get(posicion).isVisible());
    notifyDataSetChanged();
  }
}
```

Se implementan algunos de los métodos de la clase *Adapter* como *getView()* que define la vista utilizando un objeto de la clase *ContenidoVista*. En el método *toggle()* se implementa la funcionalidad para desplegar el contenido de cada resultado en la lista. Después de asignar el nuevo estado del elemento se llama al método *notifyDataSetChanged()* para que la lista sea actualizada con el nuevo estado y así se refleje el cambio en la lista de resultados al ocultar o desplegar un resultado.

Ahora se presenta la clase que define la interfaz gráfica por medio de las clases de Android. Para consultar la construcción de interfaz gráfica por medio de código consulte la sección 5.3.1.

```
private class ContenidoVista extends LinearLayout{

  private TextView vTitulo;
  private TextView vContenido;

  /**
  * Costructor de Clase
  * @param contexto objeto Context
  * @param titulo titulo del item
  * @param contenido contenido del item
  * @param desplegado true si esta desplegado
  *   false en otro caso
  */
  public ContenidoVista(Context contexto, String titulo,
    String contenido, boolean desplegado){
    super(contexto);
    //se define de manera vertical el orden de los elementos en
    //el layout
    this.setOrientation(VERTICAL);
    //se ponen propiedades del texto de titulo
    vTitulo = new TextView(contexto);
```

```
    vTitulo.setTextSize(16);
    vTitulo.setBackgroundColor(Color.BLACK);
    vTitulo.setText(titulo);
    addView(vTitulo,
    new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT,
    LayoutParams.WRAP_CONTENT));
    //se ponen propoiedades del texto del contenido
    vContenido = new TextView(contexto);
    vContenido.setText(contenido);
    vContenido.setTextColor(Color.BLACK);
    vContenido.setBackgroundColor(Color.LTGRAY);
    vContenido.setTextSize(13);
    addView(vContenido,
    new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT,
    LayoutParams.WRAP_CONTENT));
    //se define la visibilidad de acuerdo al atributo
    //desplegado
    vContenido.setVisibility(desplegado ? VISIBLE : GONE);
  }

  /**
  * Asigna el ttulo del item
  * @param titulo el titulo del item
  */
  public void setTitulo(String titulo){
    vTitulo.setText(titulo);
  }

  /**
  * Asigna el contenido del item
  * @param contenido el contenido del item
  */
  public void setContenido(String contenido){
    vContenido.setText(contenido);
  }

  /**
  * Asigna true si es visible
  * @param desplegado true si es visible
  *                   false en otro caso
  */
  public void setDesplegado(boolean desplegado){
    vContenido.setVisibility(desplegado ? VISIBLE : GONE);
  }
}
```

5.4. Resultado de pruebas

Los resultados de las pruebas diseñadas en la sección 4.4 se muestran a continuación:

I. Caso de prueba **Ingreso**:

- *El usuario introduce nombre y clave*:

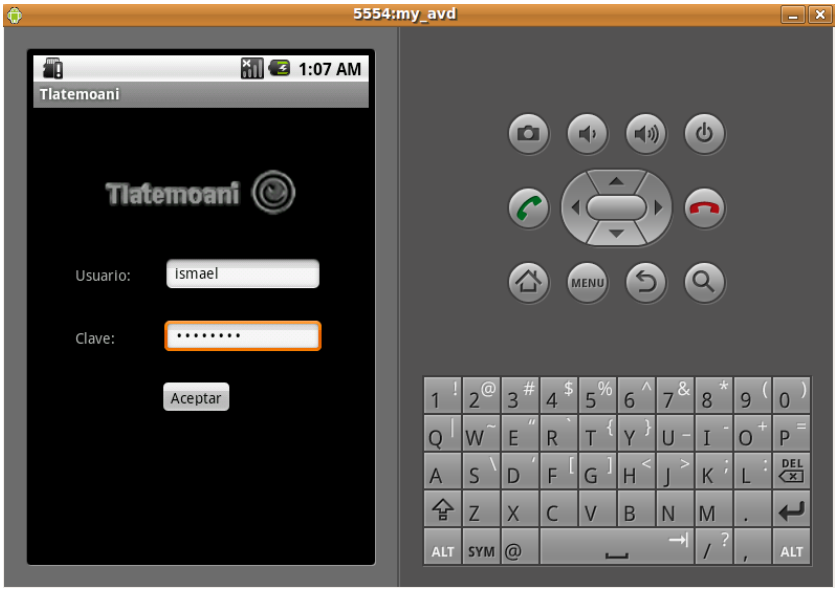

Figura 5.10: Caso de Prueba 1-1

- *Se muestra la pantalla inicial de búsqueda*:

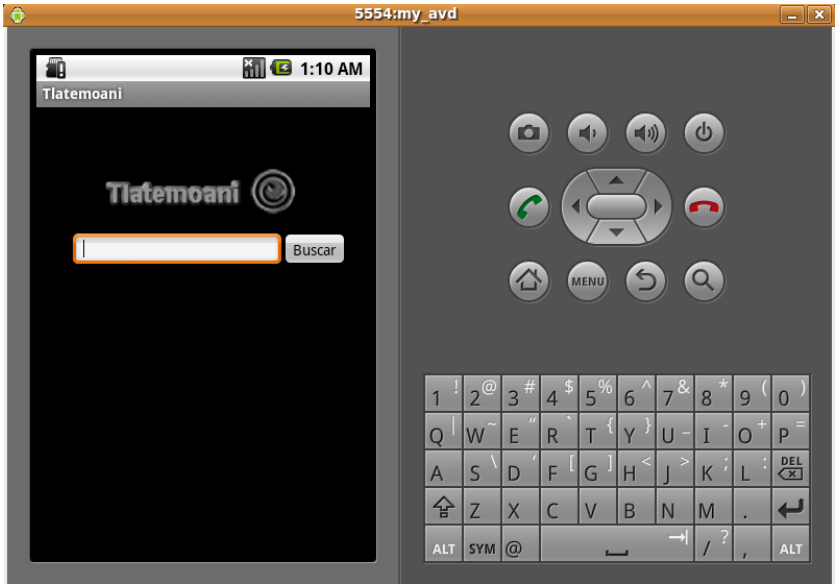

Figura 5.11: Caso de Prueba 1-2

- *Alternativa – Salir con Home*:

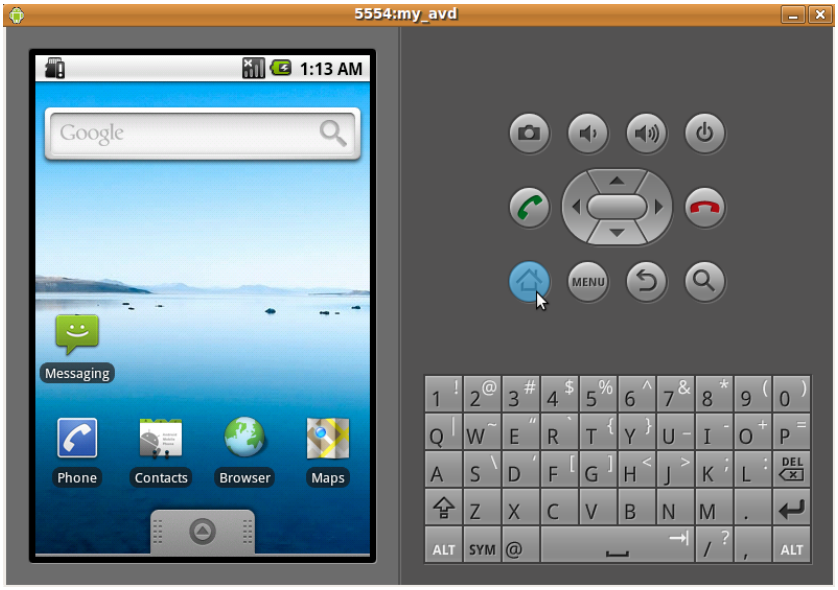

Figura 5.12: Caso de Prueba 1-3

- *Error – Usuario No válido*:

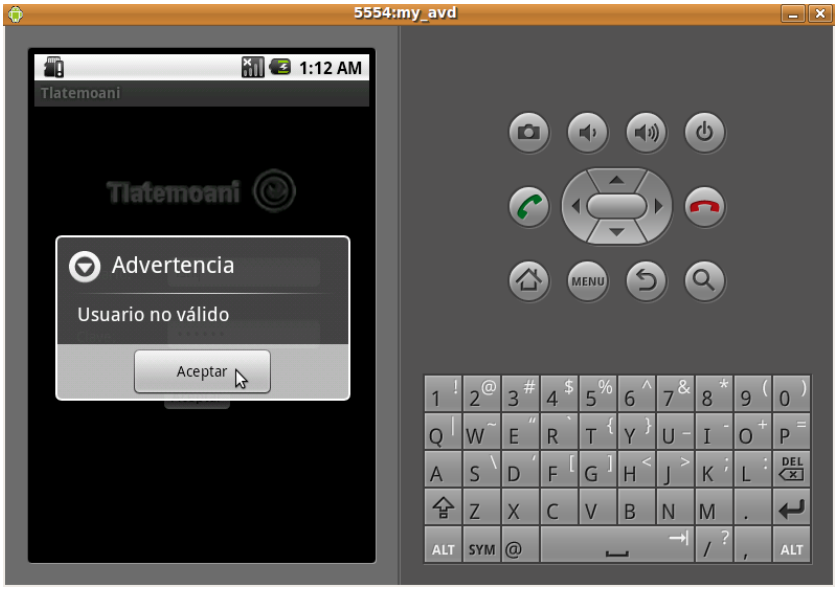

Figura 5.13: Caso de Prueba 1-4

- *Error – Sin Conexión*:

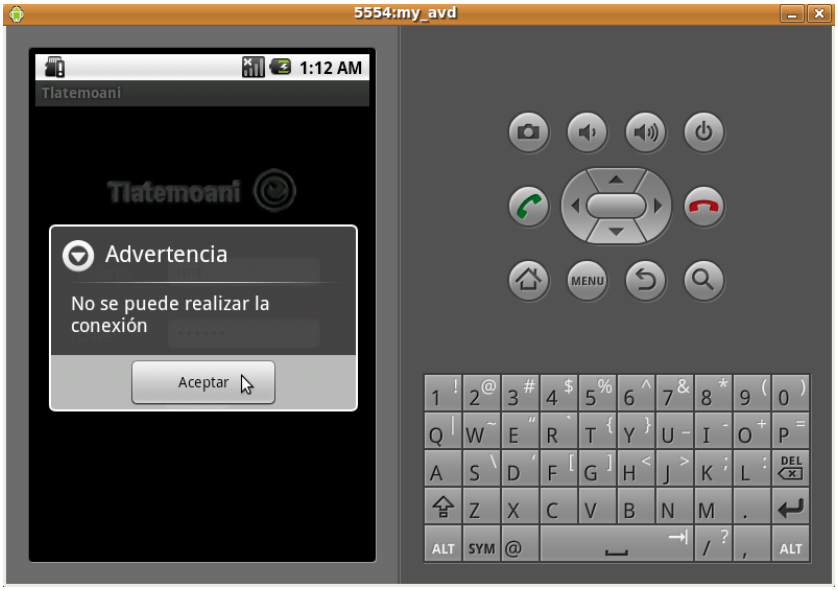

Figura 5.14: Caso de Prueba 1-5

Todas las pruebas de cada uno de los casos de uso se pueden consultar en el Apéndice C.

Por lo tanto el sistema *Tlatemoani* cumple con los casos de uso y de esta manera se cumplen los requerimientos establecidos en la sección4.1.

5.5. Resumen

En este capítulo se muesta la implementación del diseño que se define en el capítulo 4. Primero se describen los procedimientos para instalar y configurar las herramientas necesarias como la base de datos, el servidor Solr, el kit de desarrollo para aplicaciones Android y el IDE Eclipse. Después se explican las partes más importantes de las clases de los dos componentes, Solr y Android, y por último se presentan los resultados de las pruebas definidas en el capítulo de Diseño en la sección 4.4.

Capítulo 6

Conclusiones

El desarrollo de aplicaciones para dispositivos móviles son una gran oportunidad para la investigación, hace algunos años no se pensaba en tener un teléfono personal, realizar llamadas en cualquier momento a otra persona en otro país y hace algunos años más no se imaginaba que a través del teléfono celular se pudiera revisar un mapa de una ciudad de otro continente. La localización es una de las funciones más importantes de la tecnología para dispositivos móviles hoy en día. Sin embargo, para comprender su funcionamiento es necesario comenzar con las bases y ésta tesis puede ser el inicio de ese conocimiento para las personas que se interesen por la tecnología para dispositivos móviles.

El uso de diferentes tecnologías [1] permite alcanzar los objetivos desde diferentes puntos de vista, el servidor Solr es una gran herramienta para implementar un motor de búsqueda, permite utilizar muchas funciones que por tiempo no se implementan en este trabajo. Sin embargo, con el ejemplo que se desarrolla en esta tesis es posible agregar funciones de gran utilidad como las búsquedas con facet[2] o resultados highligth[3], entre muchas otras funciones.

Desde el punto de vista visual, Android y los layouts [4], hacen posible el desarrollo de la interfaz gráfica de una manera bastante rápida. Existen ahora muchos frameworks para aplicaciones en Internet que permiten construir sus interfaces gráficas de esta manera, sin embargo, en la tecnología para dispositivos móviles, la plataforma Android es una de las primeras que ofrece este tipo de frameworks integrados. Esto permite definir la interfaz de usuario de manera sencilla.

Es de gran utilidad la clase que maneja los recursos llamada *R.java*. De una manera muy fácil se pueden utilizar recursos en la aplicación ya sean imágenes, archivos, archivos de configuración como los layouts. Utilizar una clase con las referencias a los recursos es una forma que no era muy conocida en los primeros desarrollos para dispositivos móviles. Esta clase permite centrarse en la lógica del programa y no en como organizar y utilizar los demás recursos.

En los programas basados en JSP's y servlet's es muy común utilizar llamadas *AJAX*. Las llamadas asíncronas a un servidor permiten cargar elementos y datos sin tener que recargar la página por completo. En el proyecto Tlatemoani se utiliza *JSON* como medio de comunicación. Es una buena opción ya que es un formato ligero y simple además cumple con la función de envío de mensajes y datos en aplicaciones básicas.

Desarrollar la interfaz para la plataforma Android fue la mejor elección[5]. Tiene un gran futuro al ser una plataforma libre y de código abierto. Esto permitirá que se desarrolle y se mejore cada vez

[1] En la sección 2
[2] Clasificación de resultados.
[3] Resultados resaltados.
[4] En la sección 2.3.2.2
[5] En la sección 2.3

más, de esta manera los usuarios de los dispositivos móviles no estarán limitados a los sistemas y aplicaciones que los fabricantes de esos dispositivos les ofrecen. En el futuro es posible que el usuario pueda instalar su propio sistema operativo, sus aplicaciones o incluso la plataforma completa para su dispositivo móvil.

El proyecto *Tlatemoani*, puede compararse con los actuales motores de búsqueda que se describen en la sección 3.1.2, en sus funciones más básicas, como la de buscar una palabra o frase en su índice o índices, así como presentar los resultados. Aunque los buscadores más populares se utilizan para realizar una búsqueda de páginas web, en el caso de *Tlatemoani* está pensado para recuparar información desde bases de datos. Es por esto que a diferencia de los motores como Google, Yahoo o Bing no es necesario el uso de un navegador web para dispositivos móviles. De ésta manera se da la opción de tener un buscador que no solo búsque páginas en Internet. Esto permite al usuario consultar otro tipo de información que no está necesariamente en una página web.

Por lo tanto después de realizar el proyecto *Tlatemoani*, se puede realizar lo siguiente:

1. Instalación y configuración de las herramientas para el desarrollo.
2. Configuración de Solr para importar datos desde una base de datos.
3. Configuración de Solr para realizar la búsqueda en todos los campos de la tabla.
4. Realizar conexión a bases de datos por medio de un servlet.
5. Manejar información por medio de objetos JSON.
6. Construir aplicaciones para dispositivos Android.
7. Construir interfaces gráficas a través de archivos xml en Android.
8. Realizar una conexión desde una aplicación Android a un componente externo a través de HTTP.
9. Manejar formato JSON en aplicaciones Android.

Lo anterior permite en muchos casos realizar funciones que pueden integrarse en diferentes sistemas y no solo para las búsquedas. La conexión a través del protocolo HTTP es la más usada para intercambiar datos o enviar peticiones a servidores. Así cada una de las tareas puede ser implementada posteriormente en diferentes tipos de proyecto.

6.1. Contribuciones finales

Se plantearon tres contribuciones al realizar esta tesis en la sección 1.5.

1. *Diseño de un buscador para bases de datos desde una aplicación móvil:*

   Durante la implementación surgieron ideas para funciones extras, con el diseño para la aplicación es posible agregar clases o métodos que permiten modificar de manera más sencilla las funciones de la aplicación o agregar nuevas.

   Por ejemplo, al tener los datos en objetos se permite realizar cualquier operación con ellos. Una funcionalidad extra podriá ser: almacenar los datos en una base de datos interna, y para esto solo se necesita implementar una clase más.

   Además el componente Solr puede ser utilizado con interfaces desde otros dispositivos por ejemplo los iPhone, Pocket PC o sistemas con J2ME, implementando solo el componente de interfaz gráfica.

2. *Interfaz de usuario para dispositivo móvil para el servidor de búsquedas Solr:*

   Con el proyecto *Tlatemoani* se presenta una alternativa de interfaz gráfica para algunas funciones básicas del servidor Solr. Se da un ejemplo de como se puede implementar una vista gráfica de

los resultados que proporciona el motor de búsquedas en un dispositivo móvil con plataforma Android.

3. *Una forma de integración de Solr y Android:*

   Utilizando el protocolo *HTTP* se proporcionó un método de intercambio de información a través de un formato llamado JSON y que permitió que el desarrollo de esta parte de la aplicación se realizá de forma más simple al no definir archivos XML para el intercambio de información entre los componentes.

6.2. Trabajo futuro

En esta sección se describen otras alternativas a la aplicación, y además una lista de funciones que pueden mejorarse en el programa *Tlatemoani*.

- Otras Aplicaciones:

  1. Una alternativa puede ser una aplicación para sugerencia de rutas y caminos para llegar a un lugar o edificio específico dentro del campus Universitario. Podría utilizar la funcionalidad de Google Maps por ejemplo.
  2. Una aplicación cliente para el proceso de inscripción y selección de cursos en la facultad de Ciencias, utilizando contactos y diferentes medios de comunicación, por ejemplo de manera telefónica, por mensajes de texto, correo electrónico o listas de correo.

- Mejoras al proyecto Tlatemoani:

  1. Utilizar sesiones en el servlet.
  2. Encriptar los datos enviados al servlet.
  3. Implementar un método de verificación más seguro.
  4. Opción de guardar los datos en una base interna en el dispositivo.
  5. Incluir una imágen del lugar en el índice Referencias.
  6. Incluir una lista de búsquedas favoritas.
  7. Función de auto completar según las búsquedas más utilizadas.

  Con estas mejoras se ofrece una mayor protección a la información que se consulta, al encriptar los datos del usuario se evita que sean copiados por otras personas o programas. Se facilita el acceso a los datos desde la aplicación al manejar sesiones en el servidor. Con la mejora a la verificación de datos la seguridad es mayor.

  El manejo de los datos puede ser más sencillo si se almacenan en el dispositivo. Con la posibilidad de guardar datos se puede ayudar al usuario ofreciendo una lista de búsquedas favoritas o la función de autocompletar. Por último, en el caso particular de los datos en el índice Referencias, es posible agregar una imágen del sitio para facilitar el reconocimiento del lugar.

Apéndice A

Archivos de configuración

- **web.xml**
  Este archivo sirve para configurar los servlets dentro del servidor de aplicación.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation=
"http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
  <display-name>tesisServlet</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>accesoServlet</display-name>
    <servlet-name>accesoServlet</servlet-name>
    <servlet-class>unam.fciencias.isma.tesis.accesoServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>accesoServlet</servlet-name>
    <url-pattern>/accesoServlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <description></description>
    <display-name>indicesServlet</display-name>
    <servlet-name>indicesServlet</servlet-name>
    <servlet-class>unam.fciencias.isma.tesis.indicesServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>indicesServlet</servlet-name>
```

```
    <url-pattern>/indicesServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

- **AndroidManifest.xml**
  En este archivo se definen las actividades y los premisos de laaplicación entre otras cosas.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
      package="unam.fciencias.isma.tesis"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:label="@string/app_name"
    android:icon="@drawable/logo">
        <activity android:name=".Inicio"
                  android:label="@string/app_name">
            <intent-filter>
                <action
                android:name="android.intent.action.MAIN" />
                <category
                android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
<activity android:name="BusquedaPrincipal"></activity>
<activity android:name="Ayuda"></activity>
<activity android:name="Indices"></activity>
<activity android:name="ResultadosBusqueda"></activity>
<activity android:name="ResultadosBusquedaDetalle"></activity>
</application>
    <uses-sdk android:minSdkVersion="7" />

<uses-permission android:name="android.permission.INTERNET">
</uses-permission>
</manifest>
```

- **ayuda.xml**
  Es archivo define la interfaz de la pantalla ayuda.

```
 <LinearLayout android:id="@+id/LinearLayout01"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical">

 <TextView android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/tv_1" android:text="Ayuda">
 </TextView>

</LinearLayout>
```

- **busqueda_1.xml**
  Es archivo define la interfaz de la pantalla búsqueda principal.

```
<AbsoluteLayout android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/absoluteLayoutBusqueda1">
        <ImageView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
                android:id="@+id/imagenBusqueda1"
                android:src="@drawable/logo_principal"
                android:layout_y="50px" android:layout_x="60px">
        </ImageView>

        <EditText android:layout_width="wrap_content"
        android:id="@+id/EditTextBusqueda1"
                android:width="200px" android:layout_height="35px"
                android:layout_y="120px"
                android:layout_x="35px" android:textSize="6pt"
                android:maxLength="35" android:maxLines="1"
                android:minLines="1"
                android:lines="1">
        </EditText>

        <Button android:layout_width="wrap_content"
        android:id="@+id/ButtonBusqueda1"
                android:text="Buscar"
                android:layout_height="35px"
                android:layout_y="120px"
                android:layout_x="235px">
        </Button>
</AbsoluteLayout>
```

- **inicio.xml**

  Este archivo define la interfaz de la pantalla de inicio.

```
<AbsoluteLayout android:id="@+id/AbsoluteLayout01"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android">

        <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
                android:id="@+id/iv_logotipo"
                android:src="@drawable/logo_principal"
                android:layout_y="50px"
                android:layout_x="62px">
        </ImageView>

        <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
                android:layout_x="40px"
                android:layout_y="150px"
                android:id="@+id/tv_login"
                android:text="Usuario:">
        </TextView>
```

```
        <EditText android:layout_width="wrap_content"
        android:id="@+id/et_login"
                android:layout_x="125px"
                android:width="150px"
                android:layout_height="35px"
                android:layout_y="143px"
                android:textSize="6pt">
        </EditText>

        <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
                android:layout_x="40px"
                android:id="@+id/tv_pass"
                android:text="Clave:"
                android:layout_y="210px">
        </TextView>

        <EditText android:layout_width="wrap_content"
        android:layout_x="125px"
                android:layout_height="35px"
                android:width="150px"
                android:id="@+id/et_pass"
                android:layout_y="202px" android:typeface="normal"
                android:visibility="visible"
                android:inputType="textPassword"
                android:textSize="6pt">
        </EditText>

        <Button android:layout_width="wrap_content"
        android:id="@+id/boton" android:text="Aceptar"
                android:layout_x="120px"
                android:layout_height="35px"
                android:layout_y="260px">
        </Button>
</AbsoluteLayout>
```

- **strings.xml**
  En este archivo se pueden definir los Strings usados en la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Tlatemoani</string>
</resources>
```

- **ScriptBaseDatos1.txt**
  Con este archivo se general las tablas para la base de datos.

```
/* SCRIPT PARA LA BASE DE DATOS 1 */


/*CREATE DATABASE tesis1;
USE tesis1; */


CREATE TABLE rutas(
```

```
        id_ruta MEDIUMINT NOT NULL AUTO_INCREMENT,
        no_ruta VARCHAR(8) NOT NULL,
        nombre_ruta VARCHAR(45) NOT NULL,
        origen VARCHAR(20) NOT NULL,
        distancia VARCHAR(10) NOT NULL,
        no_paradas VARCHAR(2) NOT NULL,
        PRIMARY KEY (id_ruta)
);

CREATE TABLE paradas(
        id_paradas MEDIUMINT NOT NULL AUTO_INCREMENT,
        ruta VARCHAR(8) NOT NULL,
        nombre VARCHAR(50) NOT NULL,
        anterior VARCHAR(50) NOT NULL,
        proxima VARCHAR(50) NOT NULL,
        PRIMARY KEY (id_paradas)
);

CREATE TABLE referencias(
        id_referencias MEDIUMINT NOT NULL AUTO_INCREMENT,
        ruta VARCHAR(8) NOT NULL,
        lugar VARCHAR(60) NOT NULL,
        PRIMARY KEY (id_referencias)
);

CREATE TABLE directorio(
        id_directorio MEDIUMINT NOT NULL AUTO_INCREMENT,
        nombre VARCHAR(200) NOT NULL,
        cargo VARCHAR(200) NOT NULL,
        dependencia VARCHAR(150) NOT NULL,
        telefono VARCHAR(50) NOT NULL,
        extension VARCHAR(50) NOT NULL,
        fax VARCHAR(30) NOT NULL,
        correo VARCHAR(37) NOT NULL,
        PRIMARY KEY (id_directorio)
);

CREATE TABLE acceso(
        id_acceso MEDIUMINT NOT NULL AUTO_INCREMENT,
        usuario VARCHAR(50) NOT NULL,
        clave VARCHAR(8) NOT NULL,
        grupo VARCHAR(50) NOT NULL,
        PRIMARY KEY (id_acceso)
);

CREATE TABLE indices(
        id_indice MEDIUMINT NOT NULL AUTO_INCREMENT,
        nombre VARCHAR(50) NOT NULL,
        core VARCHAR(150) NOT NULL,
        estado MEDIUMINT NOT NULL,
        PRIMARY KEY (id_indice)
);

CREATE TABLE campos(
```

```
        id_campo MEDIUMINT NOT NULL AUTO_INCREMENT,
        nombre VARCHAR(50) NOT NULL,
        nombreVisible VARCHAR(50) NOT NULL,
        visible MEDIUMINT NOT NULL,
        estado MEDIUMINT NOT NULL,
        id_indice MEDIUMINT NOT NULL,
        PRIMARY KEY (id_campo)
);
```

- **solr.xml**
  En este archivo se definen los cores para Solr.

```
<?xml version='1.0' encoding='UTF-8'?>
<solr persistent='true'>
        <!-- Declaracion de los cores, -->
        <cores adminPath='/admin/cores'>
          <core name='Directorio' instanceDir='directorio/'/>
          <core name='Rutas' instanceDir='rutas/'/>
          <core name='Paradas' instanceDir='paradas/'/>
          <core name='Referencias' instanceDir='referencias/'/>
        </cores>
</solr>
```

- **solrconfig.xml**
  Este archivo se utiliza pra cada core en el diriectorio solr home.

```
   <?xml version="1.0" encoding="UTF-8" ?>

<config>

    <jmx />

    <abortOnConfigurationError>${solr.abortOnConfigurationError:true}
</abortOnConfigurationError>

    <indexDefaults>
        <useCompoundFile>false</useCompoundFile>
        <mergeFactor>10</mergeFactor>
        <ramBufferSizeMB>32</ramBufferSizeMB>
        <maxMergeDocs>2147483647</maxMergeDocs>
        <maxFieldLength>10000</maxFieldLength>
        <writeLockTimeout>1000</writeLockTimeout>
        <commitLockTimeout>10000</commitLockTimeout>
        <lockType>single</lockType>
    </indexDefaults>

    <mainIndex>
        <useCompoundFile>false</useCompoundFile>
        <ramBufferSizeMB>32</ramBufferSizeMB>
        <mergeFactor>10</mergeFactor>
        <maxMergeDocs>2147483647</maxMergeDocs>
        <maxFieldLength>10000</maxFieldLength>

        <unlockOnStartup>false</unlockOnStartup>
    </mainIndex>
```

```
    <updateHandler class="solr.DirectUpdateHandler2">
        <maxPendingDeletes>100000</maxPendingDeletes>
    </updateHandler>

    <query>
        <maxBooleanClauses>1024</maxBooleanClauses>

        <filterCache class="solr.LRUCache"
        size="512" initialSize="512" autowarmCount="256"/>

        <queryResultCache class="solr.LRUCache" size="512"
        initialSize="512" autowarmCount="256"/>

        <documentCache class="solr.LRUCache" size="512"
        initialSize="512" autowarmCount="0"/>

        <enableLazyFieldLoading>true</enableLazyFieldLoading>

        <queryResultWindowSize>50</queryResultWindowSize>

        <queryResultMaxDocsCached>200</queryResultMaxDocsCached>

        <HashDocSet maxSize="3000" loadFactor="0.75"/>

        <listener event="newSearcher"
                class="solr.QuerySenderListener">
            <arr name="queries">
                <lst> <str name="q">solr</str>
                <str name="start">0</str>
                <str name="rows">10</str> </lst>
                <lst> <str name="q">rocks</str>
                <str name="start">0</str>
                <str name="rows">10</str> </lst>
                <lst><str name="q">static newSearcher
                warming query from solrconfig.xml</str></lst>
            </arr>
        </listener>

        <listener event="firstSearcher"
        class="solr.QuerySenderListener">
            <arr name="queries">
            </arr>
        </listener>

        <useColdSearcher>false</useColdSearcher>

        <maxWarmingSearchers>4</maxWarmingSearchers>
    </query>
    <requestDispatcher handleSelect="true" >
        <requestParsers enableRemoteStreaming="false"
        multipartUploadLimitInKB="2048" />
<httpCaching never304="true">
        </httpCaching>
```

```
</requestDispatcher>

<requestHandler name="standard" class="solr.SearchHandler"
    default="true">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
            <str name="fl">
            *
            </str>
    </lst>
</requestHandler>

<requestHandler name="dismax" class="solr.SearchHandler" >
    <lst name="defaults">
        <str name="defType">dismax</str>
        <str name="echoParams">explicit</str>
        <str name="fl">
            nombre cargo
        </str>
    </lst>
</requestHandler>

<requestHandler name="spellchecker"
    class="solr.SpellCheckerRequestHandler" startup="lazy">
    <lst name="defaults">
        <int name="suggestionCount">1</int>
        <float name="accuracy">0.5</float>
    </lst>

    <str name="spellcheckerIndexDir">spell</str>

    <str name="termSourceField">word</str>
</requestHandler>

<requestHandler name="/dataimport"
    class="org.apache.solr.handler.dataimport.DataImportHandler">
    <lst name="defaults">
        <str name="config">data-config.xml</str>
    </lst>
</requestHandler>
<requestHandler name="/search"
    class="org.apache.solr.handler.component.SearchHandler">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
    </lst>

</requestHandler>
<requestHandler name="/update"
    class="solr.XmlUpdateRequestHandler" >
</requestHandler>

<requestHandler name="/analysis"
    class="solr.AnalysisRequestHandler" >
</requestHandler>
```

```
    <requestHandler name="/admin/"
        class="org.apache.solr.handler.admin.AdminHandlers" />
    <requestHandler name="/admin/ping"
        class="PingRequestHandler">
        <lst name="defaults">
            <str name="qt">standard</str>
            <str name="q">solrpingquery</str>
            <str name="echoParams">all</str>
        </lst>
    </requestHandler>

    <requestHandler name="/debug/dump"
        class="solr.DumpRequestHandler" >
        <lst name="defaults">
            <str name="echoParams">explicit</str>
            <str name="echoHandler">true</str>
        </lst>
    </requestHandler>
    <admin>
        <defaultQuery>*:*</defaultQuery>
</admin>

</config>
```

- **directorio/data-config.xml**
  En este archivo se define el documento y la entidad además del datasource.

```
<dataConfig>
    <!-- Definicion del datasource, puede haber varios -->
    <dataSource driver="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/tesis"
        user="ismael" password="123456789" />
    <!-- Se crea un documento nuevo con la informacion de la
  tabla directorio -->
    <document name="tabla_directorio">
        <entity name="directorio" query="select id_directorio,
  nombre as Nombre, cargo as Cargo, dependencia as Dependencia,
  telefono as Telefono, extension as Extension, fax as Fax,
  correo as Correo_Electronico from directorio">
        </entity>
    </document>
</dataConfig>
```

- **directorio/schema.xml**
  En este archivo se definen los campos para la indexación.

```
   <?xml version="1.0" encoding="UTF-8" ?>

<!-- Definicion del esquema-->
<schema name="tesis_directorio" version="1.1">

<!-- Definicion de tipos -->
  <types>
    <fieldType name="integer" class="solr.IntField"
```

```
    omitNorms="true"/>
    <!-- Se utiliza un tipo de texto con un filtro para
    separar por espacios en blanco -->
    <fieldType name="text_ws" class="solr.TextField"
    positionIncrementGap="100">
      <analyzer>
              <tokenizer
        class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
    <!-- Se utiliza un tipo de texto para separar por palabras,
    ignorar mayusculas y minusculas, y quitar palabras duplicadas-->
    <fieldType name="text" class="solr.TextField"
    positionIncrementGap="100">
            <analyzer type="index">
              <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
              <filter
        class="solr.RemoveDuplicatesTokenFilterFactory"/>
            </analyzer>
         </fieldType>
 </types>

<!-- Definicion de Campos -->
  <fields>
    <!-- Se definen los campos principales de la tabla,
    el id no se muestra -->
    <field name="id_directorio" type="integer" indexed="true"
    stored="false" required="true" />
    <field name="Nombre" type="text_ws" indexed="true"
    stored="true" />
    <field name="Cargo" type="text_ws" indexed="true"
    stored="true" />
    <field name="Dependencia" type="text_ws" indexed="true"
    stored="true" />
    <field name="Telefono" type="text_ws" indexed="true"
    stored="true" />
    <field name="Extension" type="text_ws" indexed="true"
    stored="true" />
    <field name="Fax" type="text_ws" indexed="true"
    stored="true" />
    <field name="Correo_Electronico" type="text_ws"
    indexed="true" stored="true" />

    <!-- Se implementa con un copyfield mas abajo,
se utiliza para que todos
    los campos se utilicen si se omite la especificacion
del nombre del campo -->
       <field name="todos" type="text" indexed="true"
stored="false" multiValued="true"/>
    <field name="zztitulo" type="text" indexed="true"
stored="true"/>
  </fields>
```

```
<!-- Para la unicidad de documentos, se utiliza uno de los campos,
debe ser un campo reuqerido -->
<uniqueKey>id_directorio</uniqueKey>

<!-- Campo para el QueryParser, para usarlo cuando un nombre de
campo explicito esta ausente -->
<defaultSearchField>todos</defaultSearchField>

<!-- Configuracion de SolrQueryParser: defaultOperator="AND|OR" -->
 <solrQueryParser defaultOperator="OR"/>

<!-- Se copian los campos para diferentes propositos, en este
caso para utilizar la busqueda sin especificar
explicitamente el nombre de cada campo.-->
   <copyField source="Nombre" dest="todos"/>
   <copyField source="Cargo" dest="todos"/>
   <copyField source="Dependencia" dest="todos"/>
   <copyField source="Telefono" dest="todos"/>
   <copyField source="Extension" dest="todos"/>
   <copyField source="Fax" dest="todos"/>
   <copyField source="Correo_Electronico" dest="todos"/>
   <copyField source="Nombre" dest="zztitulo"/>
</schema>
```

- **paradas/data-config.xml**
  En este archivo se define el documento y la entidad además del datasource.

```
<dataConfig>
    <!-- Definicion del datasource, puede haber varios -->
    <dataSource driver="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/tesis"
        user="ismael" password="123456789" />
    <!-- Se crea un documento nuevo con la informacion
de la tabla paradas -->
    <document name="tabla_paradas">
        <entity name="paradas" query="select id_paradas,
ruta as Ruta, nombre as Nombre, anterior as Parada_Anterior,
proxima as Parada_Proxima from paradas">
        </entity>
    </document>
</dataConfig>
```

- **paradas/schema.xml**
  En este archivo se definen los campos para la indexación.

```
   <?xml version="1.0" encoding="UTF-8" ?>

<!-- Definicion del esquema-->
<schema name="tesis_paradas" version="1.1">

<!-- Definicion de tipos -->
  <types>
    <fieldType name="integer" class="solr.IntField" omitNorms="true"/>
    <!-- Se utiliza un tipo de texto con un filtro para separar por
espacios en blanco -->
```

```
    <fieldType name="text_ws" class="solr.TextField"
positionIncrementGap="100">
      <analyzer>
              <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
    <!-- Se utiliza un tipo de texto para separar por palabras, ignorar
mayusculas y minusculas, y quitar palabras duplicadas -->
    <fieldType name="text" class="solr.TextField"
        positionIncrementGap="100">
            <analyzer type="index">
              <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
              <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
            </analyzer>
         </fieldType>
 </types>

<!-- Definicion de Campos -->
  <fields>
    <!-- Se definen los campos principales de la tabla, el id no
se muestra -->
    <field name="id_paradas" type="integer" indexed="true"
stored="false" required="true" />
    <field name="Ruta" type="text_ws" indexed="true"
stored="true" />
    <field name="Nombre" type="text_ws" indexed="true"
stored="true" />
    <field name="Parada_Anterior" type="text_ws"
indexed="true" stored="true" />
    <field name="Parada_Proxima" type="text_ws" indexed="true"
stored="true" />

    <!-- Se implementa con un copyfield mas abajo, se utiliza
para que todos
     los campos se utilicen si se omite la especificacion del
nombre del campo -->
       <field name="todos" type="text" indexed="true"
stored="false" multiValued="true"/>
    <field name="zztitulo" type="text" indexed="true"
stored="true"/>
  </fields>

<!-- Para la unicidad de documentos, se utiliza uno de los campos,
debe ser un campo reuqerido -->
<uniqueKey>id_paradas</uniqueKey>

<!-- Campo para el QueryParser, para usarlo cuando un nombre de
campo explicito esta ausente -->
<defaultSearchField>todos</defaultSearchField>

<!-- Configuracion de SolrQueryParser: defaultOperator="AND|OR" -->
 <solrQueryParser defaultOperator="OR"/>
```

```
<!-- Se copian los campos para diferentes propositos,
en este caso para utilizar la busqueda sin especificar
explicitamente el nombre de cada campo.-->
   <copyField source="Ruta" dest="todos"/>
   <copyField source="Nombre" dest="todos"/>
   <copyField source="Parada_Anterior" dest="todos"/>
   <copyField source="Parada_Proxima" dest="todos"/>
   <copyField source="Nombre" dest="zztitulo"/>
</schema>
```

- **referencias/data-config.xml**
  En este archivo se define el documento y la entidad además del datasource.

```
<dataConfig>
    <!-- Definicion del datasource, puede haber varios -->
    <dataSource driver="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/tesis"
        user="ismael" password="123456789" />
    <!-- Se crea un documento nuevo con la informacion
de la tabla referencias -->
    <document name="tabla_referencias">
        <entity name="referencias" query="select id_referencias,
ruta as Ruta, lugar as Lugar from referencias">
        </entity>
    </document>
</dataConfig>
```

- **referencias/schema.xml**
  En este archivo se definen los campos para la indexación.

```
   <?xml version="1.0" encoding="UTF-8" ?>

<!-- Definicion del esquema-->
<schema name="tesis_referencias" version="1.1">

<!-- Definicion de tipos -->
  <types>
    <fieldType name="integer" class="solr.IntField" omitNorms="true"/>
    <!-- Se utiliza un tipo de texto con un filtro para separar por
espacios en blanco -->
    <fieldType name="text_ws" class="solr.TextField"
positionIncrementGap="100">
      <analyzer>
            <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
    <!-- Se utiliza un tipo de texto para separar por palabras,
ignorar mayusculas y minusculas, y quitar palabras duplicadas-->
    <fieldType name="text" class="solr.TextField"
        positionIncrementGap="100">
            <analyzer type="index">
              <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
              <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
            </analyzer>
```

```
        </fieldType>
 </types>

<!-- Definicion de Campos -->
  <fields>
    <!-- Se definen los campos principales de la tabla, el id no se
muestra -->
    <field name="id_referencias" type="integer" indexed="true"
stored="false" required="true" />
    <field name="Ruta" type="text_ws" indexed="true"
stored="true" />
    <field name="Lugar" type="text_ws" indexed="true"
stored="true" />

    <!-- Se implementa con un copyfield mas abajo, se utiliza para
que todos
    los campos se utilicen si se omite la especificacion del nombre
del campo -->
       <field name="todos" type="text" indexed="true" stored="false"
multiValued="true"/>
    <field name="zztitulo" type="text" indexed="true" stored="true"/>
  </fields>

<!-- Para la unicidad de documentos, se utiliza uno de los campos,
debe ser un campo reuqerido -->
<uniqueKey>id_referencias</uniqueKey>

<!-- Campo para el QueryParser, para usarlo cuando un nombre de campo
explicito esta ausente -->
<defaultSearchField>todos</defaultSearchField>

<!-- Configuracion de SolrQueryParser: defaultOperator="AND|OR" -->
 <solrQueryParser defaultOperator="OR"/>

<!-- Se copian los campos para diferentes propositos, en este caso
para utilizar la busqueda sin especificar
explicitamente el nombre de cada campo.-->
   <copyField source="Ruta" dest="todos"/>
   <copyField source="Lugar" dest="todos"/>
   <copyField source="Lugar" dest="zztitulo"/>
</schema>
```

- **rutas/data-config.xml**

  En este archivo se define el documento y la entidad además del datasource.

```
<dataConfig>
    <!-- Definicion del datasource, puede haber varios -->
    <dataSource driver="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/tesis"
        user="ismael" password="123456789" />
    <!-- Se crea un documento nuevo con la informacion
de la tabla rutas -->
    <document name="tabla_rutas">
        <entity name="rutas" query="select id_ruta, no_ruta
 as Numero_Ruta, nombre_ruta as Nombre_Ruta, origen as Origen,
```

```
distancia as Distancia,no_paradas as Numero_Paradas from rutas;">
        </entity>
    </document>
</dataConfig>
```

- **rutas/schema.xml**
  En este archivo se definen los campos para la indexación.

```
   <?xml version="1.0" encoding="UTF-8" ?>

<!-- Definicion del esquema-->
<schema name="tesis" version="1.1">

<!-- Definicion de tipos -->
  <types>
    <fieldType name="integer" class="solr.IntField"
omitNorms="true"/>
    <!-- Se utiliza un tipo de texto con un filtro para
separar por espacios en blanco -->
    <fieldType name="text_ws" class="solr.TextField"
positionIncrementGap="100">
      <analyzer>
            <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
    <!-- Se utiliza un tipo de texto para separar por palabras,
ignorar mayusculas y minusculas, y quitar palabras duplicadas-->
    <fieldType name="text" class="solr.TextField"
positionIncrementGap="100">
          <analyzer type="index">
            <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
            <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
          </analyzer>
        </fieldType>
 </types>

<!-- Definicion de Campos -->
  <fields>
    <!-- Se definen los campos principales de la tabla, el id no
se muestra -->
    <field name="id_ruta" type="integer" indexed="true"
stored="false" required="true" />
    <field name="Numero_Ruta" type="text_ws" indexed="true"
stored="true" />
    <field name="Nombre_Ruta" type="text_ws" indexed="true"
stored="true" />
    <field name="Origen" type="text_ws" indexed="true"
stored="true" />
    <field name="Distancia" type="text_ws" indexed="true"
stored="true" />
    <field name="Numero_Paradas" type="text_ws" indexed="true"
stored="true" />

    <!-- Se implementa con un copyfield mas abajo, se utiliza
```

```
para que todos
    los campos se utilicen si se omite la especificacion del
nombre del campo -->
       <field name="todos" type="text" indexed="true" stored="false"
multiValued="true"/>
    <field name="zztitulo" type="text" indexed="true" stored="true"/>
  </fields>

<!-- Para la unicidad de documentos, se utiliza uno de los campos,
debe ser un campo reuqerido -->
<uniqueKey>id_ruta</uniqueKey>

<!-- Campo para el QueryParser, para usarlo cuando un nombre de
campo explicito esta ausente -->
<defaultSearchField>todos</defaultSearchField>

<!-- Configuracion de SolrQueryParser: defaultOperator="AND|OR" -->
 <solrQueryParser defaultOperator="OR"/>

<!-- Se copian los campos para diferentes propositos, en este caso
para utilizar la busqueda sin especificar
explicitamente el nombre de cada campo.-->
   <copyField source="Nombre_Ruta" dest="todos"/>
   <copyField source="Numero_Ruta" dest="todos"/>
   <copyField source="Numero_Paradas" dest="todos"/>
   <copyField source="Origen" dest="todos"/>
   <copyField source="Distancia" dest="todos"/>
   <copyField source="Nombre_Ruta" dest="zztitulo"/>
</schema>
```

Apéndice B

Código fuente completo

Ayuda.java

```
package unam.fciencias.isma.tesis;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase genera la pantalla de ayuda de la
 * aplicacion,
 * por medio del layout ayuda.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class Ayuda extends Activity{

  /**
   * @see android.app.Activity#onCreate(android.os.Bundle)
   */
  @Override
  protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.ayuda);

    TextView tv = (TextView) findViewById(R.id.tv_1);
    String ayuda =
    "[Acceso] \n\n" +
    "Para utiliza la aplicaci n es necesario contar "+
    "con un nombre de usuario y una clave. \n\n" +
    "[Uso B sico] \n\n" +
    "Se debe introducir una palabra para buscar."+
    "Se debe seleccionar al menos un  ndice, por defecto"+
    "todos los  ndices disponibles est n seleccionados.\n"+
    "Para ingresar al men  se utiliza la tecla \"menu\" \n"+
    "Para salir de la aplicaci n se utiliza la tecla "+
```

```
    "(<-) o en el men  la opci n \"Salir\" \n\n"+
    "[Indices] \n\n"+
    "En la pantalla se enlistan los indices disponibles para "+
    "la b squeda , se pueden seleccionar o quitar la selecci n. \n"+
    "Se debe seleccionar al menos un  ndice de b squeda \n\n"+
    "[Resultados] \n\n"+
    "Para visualizar los detalles de cada resultado se debe hacer "+
    "click sobre el t tulo de cada resultado para desplegar la "+
    "informaci n , se da click de nuevo para ocultarla \n\n"+
    "[Acerca de] \n\n"+
    "Desarrollado por Ismael J Galv n Hern ndez \n"+
    "2010 \n"+
    "fharem@gmail.com";
    tv.setText(ayuda);
    tv.setTextSize(11);
  }

}
```

BusquedaPrincipal.java

```
package unam.fciencias.isma.tesis;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, genera la pantalla principal de busqueda
 * en la aplicacion
 * por medio del layout busqueda_1, crea un Intent
 * hacia la pantalla
 * donde se mestran las coincidencias de cada indice
 * de busqueda.
 *
 * Tambien crea un menu, con las opciones, Indices,
 * Ayuda y Salir.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class BusquedaPrincipal extends Activity{

  Menu miMenu = null;
```

```
/**
 * Este metodo sirve para generar los indices
 * seleccionados
 * para enviarlos en la petici n http al servlet.
 * Se genera con el
 * id del indice y con el caracter 'a' como
 * separador
 * @return resultado cadena de indices seleccionados
 */
private static String generaIndicesSeleccionados(){
  String resultado = "";
  for(int k=0;k<Datos.getIndices().size();k++){
    if(Datos.getIndices().get(k).isSeleccionado()){
      resultado += Datos.getIndices().get(k).getId()+"a";
    }
  }
  return resultado;
}

/**
 * @see android.app.Activity#onCreate(android.os.Bundle)
 */
@Override
public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.busqueda_1);
      //se crea un objeto para obtener los indices de b squeda
      final ControladorRespuesta cr = new ControladorRespuesta();
      cr.obtieneIndicesBusqueda(1);
      //se obtiene la caja de texto
      final EditText tv_query =
        (EditText)findViewById(R.id.EditTextBusqueda1);
      //se obtiene el boton para implementar el listener
      Button buscar = (Button)findViewById(R.id.ButtonBusqueda1);
      buscar.setOnClickListener(new View.OnClickListener(){
          public void onClick(View view){
            String query = tv_query.getText().toString();
            Datos.setQuery(query);
            String indicesSeleccionados =
              BusquedaPrincipal.generaIndicesSeleccionados();
            //se obtienen las coincidencias
            cr.obtieneResultados(2,query,indicesSeleccionados);

            AlertDialog alertDialog =
              new AlertDialog.Builder(BusquedaPrincipal.this).create();
        alertDialog.setTitle("Advertencia");
        alertDialog.setButton("Aceptar",
            new DialogInterface.OnClickListener() {
                 public void onClick(
                     DialogInterface dialog, int which) {
                         return;
                 }
             });
        if(Datos.isErrorConexion() ||
```

```
            Datos.isErrorConexion()){
          alertDialog.setMessage(
              "No se puede realizar la conexi n");
            alertDialog.show();
        }
            //se muestra una alerta si no se escribe nada
        else if(query.length()<=0){
              alertDialog.setMessage("Escriba al menos una pablabra");
            alertDialog.show();
              }
            //se muestra una alerta si no esta
            //seleccionado ningun indice
            else if(Datos.getnumIndicesSeleccionados()<=0){
              alertDialog.setMessage("Seleccione al menos un  ndice");
            alertDialog.show();
            }
            //se inicia la otra actividad
            else{
              Intent myIntent =
                new Intent(view.getContext(),ResultadosBusqueda.class);
                startActivityForResult(myIntent, 0);
            }
          }
      });
}

/**
 * @see android.app.Activity#onCreateOptionsMenu(android.view.Menu)
 */
@Override
public boolean onCreateOptionsMenu(Menu menu){
   super.onCreateOptionsMenu(menu);
   this.miMenu = menu;
   //id grupo, id, orden, titulo
   menu.add(0,1,0," ndices");
   menu.add(0,2,1,"Ayuda");
   menu.add(0,3,2,"Salir");
   //se crean los items y se asigna el icono
   MenuItem m1 = menu.getItem(0);
   m1.setIcon(R.drawable.indices);
   MenuItem m2 = menu.getItem(1);
   m2.setIcon(R.drawable.ayuda);
   MenuItem m3 = menu.getItem(2);
   m3.setIcon(R.drawable.salir);
   return true;
}

/**
 * @see android.app.Activity#onOptionsItemSelected(android.view.MenuItem)
 */
@Override
public boolean onOptionsItemSelected(MenuItem item){
  Intent intent;
  //se obtiene el item seleccionado y se inicia la actividad
```

```
    //seg n sea el caso.
    switch(item.getItemId()){
      case 1:
        intent = new Intent(this.getBaseContext(),
            Indices.class);
                startActivityForResult(intent,0);
        break;
      case 2:
        intent = new Intent(this.getBaseContext(),
            Ayuda.class);
                startActivityForResult(intent,0);
        break;
      case 3:
        //termina esta actividad y regresa a la actividad anterior
        Datos.clear();
        finish();
        break;
      default:
        break;
    }
    return true;
  }

}
```

ConexionServlet.java

```
package unam.fciencias.isma.tesis;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONObject;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, crea la conexion entre la aplicacion
 * en Android y el
 * servlet, por medio del metodo conexion(url)
 * obtiene el objeto JSON
 * como respuesta.
 *
 * @author              Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class ConexionServlet{
```

```
    /**
     * Este metodo genera un String utilizando un
     * objeto InputStream.
     * @param is objeto InputStream
     * @return cadena cadena que representara un Objeto JSON
     */
    private static String conversionString(InputStream is){
    if(is != null){
            BufferedReader br = new BufferedReader(
                                new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String linea = null;
    try{
        while ((linea = br.readLine()) != null){
            sb.append(linea).append("\n");
        }
    } catch (IOException e){
        e.printStackTrace();
    } finally {
        try {
            is.close();
        } catch (IOException e){
            e.printStackTrace();
        }
    }
    return sb.toString();
    }
    else{
            return "";
    }
}

/**
 * Este metodo se conecta a la url que se pasa
 * como parametro
 * y regresa un objeto JSON desde el servlet
 */
public static JSONObject conexion(String url){
    JSONObject json = new JSONObject();
    try{
            HttpClient httpclient = new DefaultHttpClient();
            //crea un objeto para la petici n GET
            HttpGet httpget = new HttpGet(url);
            //para la respuesta
            HttpResponse response;
            response = httpclient.execute(httpget);
            //obtiene la entidad de respuesta
        HttpEntity entity = response.getEntity();
        if(entity != null){
            //se obtiene un objeto JSON como String
            InputStream instream = entity.getContent();
            String resultado = conversionString(instream);
            //se crea un JSONObject
```

```
                json = new JSONObject(resultado);
                //cierra el InputStream
                instream.close();
            }
            Datos.setErrorConexion(false);
        } catch(Exception ae){
                Datos.setErrorConexion(true);
        }
        return json;
    }

}
```

ControladorRespuesta.java

```
package unam.fciencias.isma.tesis;

import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.Iterator;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, se utiliza para obtener realizar
 * las peticiones y poner
 * la informacion en la clase Datos para que las
 * actividades tengan acceso
 * a los datos.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class ControladorRespuesta{

  //el alias para el localhost del s.o. es 10.0.2.2
  private final String URL_VERIFICA_ACCESO =
    "http://10.0.2.2:8080/tesisServlet/accesoServlet";
  private final String URL_INDICES_BUSQUEDA =
    "http://10.0.2.2:8080/tesisServlet/indicesServlet";

  /**
   * Este metodo verifica que el usuario este
   * registrado
   * @param usuario nombre del usuario
   * @param clave contrase a del usuario
   */
  public void verificaAcceso(String usuario, String clave){
    //se indican los parametros para el servlet
    String params = "?usr="+usuario+"&clv="+clave;
    //se obtiene la respuesta como objeto JSON
```

```
  JSONObject json = ConexionServlet.conexion(
      this.URL_VERIFICA_ACCESO+params);
  boolean respuesta;
  try{
    respuesta = json.getBoolean("acceso");
    //se actualiza el valor
    Datos.setAcceso(respuesta);
    Datos.setErrorJson(false);
  } catch(JSONException e){
    Datos.setErrorJson(true);
  }
}

/**
 * Este metodo sirve para obtener los indices de
 * busqueda.
 * @param metodo indica el m todo para el servlet
 */
public void obtieneIndicesBusqueda(int metodo){
  if(Datos.getIndices().size()==0){
  //se indican los parametros para el servlet
  String params = "?mtd="+metodo;
  JSONObject json = ConexionServlet.conexion(
      this.URL_INDICES_BUSQUEDA+params);
  //se obtienen los elementos llave del objeto JSON
  Iterator<?> it = json.keys();
  ArrayList<Indice> indices = new ArrayList<Indice>();
  //contador para indices seleccionados
  int k=0;
  while(it.hasNext()){
    try{
      //se asignan los datos
      String nombre = (String)it.next();
      int llave = json.getInt(nombre);
      Indice indice = new Indice();
      indice.setId(llave);
      indice.setNombre(nombre);
      indice.setSeleccionado(true);
      indices.add(indice);
      k++;
      Datos.setErrorJson(false);
    } catch (JSONException e) {
      Datos.setErrorJson(true);
    }
  }
  //se asigna los indices y el numero de indices
  //seleccionados
  //para las actividades
  Datos.setIndices(indices);
  Datos.setnumIndicesSeleccionados(k);
  }
}

/**
```

```
 * Este metodo sirve para obtener el numero de
 * coincidencias
 * de cada indice
 * @param metodo indica el metodo para el servlet
 * @param nquery la consulta
 * @param indices una cadena que representa el numero
 *           de indices sobre el cual se debe buscar
 */
public void obtieneResultados(int metodo,String nquery,
    String indices){
  String query = URLEncoder.encode(nquery);
  //se definen los parametros para el servlet
  String params = "?mtd="+metodo+"&qry="+
          query+"&idxs="+indices;
  //se obtiene el objeto JSON
  JSONObject json =
    ConexionServlet.conexion(
        this.URL_INDICES_BUSQUEDA+params);
  Iterator<?> it = json.keys();
  while(it.hasNext()){
    try{
      //se asignan los valores a los  ndices
      String nombre = (String)it.next();
      int numeroResultados = json.getInt(nombre);
      for(int k=0;k<Datos.getIndices().size();k++){
        if(Datos.getIndices().get(k).
            getNombre().equals(nombre)){
          Datos.getIndices().get(k).
          setNumResultados(numeroResultados);
        }
      }
      Datos.setErrorJson(false);
    } catch(JSONException e){
      Datos.setErrorJson(true);
    }
  }
}

/**
 * Este m todo sirve para obtener los resultados de
 * la b squeda sobre el  ndice indicado
 * @param metodo indica el m todo para el servlet
 * @param nquery la consulta
 * @param indice id del  ndice
 */
public void obtieneResultadosDetalle(int metodo,
    String nquery,String indice){
  String query = URLEncoder.encode(nquery);
  //se definen los parametros para el servlet
  String params = "?mtd="+metodo+"&qry="+
          query+"&idx="+indice;
  //se obtiene el objeto JSON
  JSONObject json = ConexionServlet.conexion(
      this.URL_INDICES_BUSQUEDA+params);
```

```
    //se genera un arreglo de objetos de tipo Resultado
    ArrayList<Resultado> listaResultados =
      new ArrayList<Resultado>();
    try{
      JSONArray jsona = json.getJSONArray("resultados");
      for(int i=0;i<jsona.length();i++){
        JSONObject jsonAux = jsona.getJSONObject(i);
        Iterator<?> it = jsonAux.keys();
        Resultado resultado = new Resultado();
        String contenido="";

        while(it.hasNext()){
          try{

            String llave = (String)it.next();
            String valor= (String)jsonAux.get(llave);
            //se pone el titulo con el valor de la llave
            //que se indic  en la iindexaci n de la tabla
            if(llave.equals("zztitulo")){
              resultado.setTitulo(valor+" (ver m s)");
            }
            else{
              //se genera el contenido como una lista de
              //parejas, llave:valor
              contenido += llave+" :\n   "+valor+"\n";
            }
            Datos.setErrorJson(false);
          }catch (JSONException e){
            Datos.setErrorJson(true);
          }
        }
        //se asignan los valores en Datos para las actividades
        resultado.setVisible(false);
        resultado.setContenido(contenido);
        listaResultados.add(i,resultado);
      }
      Datos.setResultados(listaResultados);
    }
    catch(JSONException e){
    }
  }

}
```

Datos.java

```
package unam.fciencias.isma.tesis;

import java.util.ArrayList;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, se utiliza para almacenar los datos y estados de los
 * elementos que se utilizan a traves de toda la aplicacion.
```

```
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class Datos{

  private static boolean acceso = false;
  private static ArrayList<Indice> indices = new ArrayList<Indice>();
  private static ArrayList<Resultado> resultados = new ArrayList<Resultado>();
  private static String query;
  private static int indiceSeleccionado;
  private static int numIndicesSeleccionados=0;
  private static boolean error_conexion = false;
  private static boolean error_json = false;


  /**
   * Regresa el estado de acceso
   * @return acceso true si tiene acceso,
   *           false en otro caso
   */
  public static boolean isAcceso(){
    return acceso;
  }

  /**
   * Asigna el estado de acceso
   * @param acceso true si tiene acceso,
   *           false en otro caso
   */
  public static void setAcceso(boolean acceso){
    Datos.acceso = acceso;
  }

  /**
   * Regresa la lista de indices
   * @return indices lista de indices
   */
  public static ArrayList<Indice> getIndices(){
    return indices;
  }

  /**
   * Asigna la lista de indices
   * @param nindices lista de indices
   */
  public static void setIndices(
      ArrayList<Indice> nindices){
    indices = nindices;
  }

  /**
   * Regresa la lista de resultados
```

```
 * @return resultados lista de resultados
 */
public static ArrayList<Resultado> getResultados(){
  return resultados;
}

/**
 * Asigna la lista de resultados
 * @param resultados lista de resultados
 */
public static void setResultados(ArrayList<Resultado> resultados){
  Datos.resultados = resultados;
}

/**
 * Regresa el query
 * @return query el query
 */
public static String getQuery(){
  return query;
}

/**
 * Asigna el query
 * @param query el query
 */
public static void setQuery(String query){
  Datos.query = query;
}

/**
 * Regresa el numero de indices seleccionados
 * @return numIndicesSeleccionados el numero de
 * indices seleccionados
 */
public static int getnumIndicesSeleccionados(){
  return numIndicesSeleccionados;
}

/**
 * Asigna el numero de indices seleccionados
 * @param numIndicesSeleccionados el numero de
 * indices seleccionados
 */
public static void setnumIndicesSeleccionados(
    int numIndicesSeleccionados){
  Datos.numIndicesSeleccionados =
    numIndicesSeleccionados;
}

/**
 * Regresa el id del indice seleccionado
 * @return indiceSeleccionado el id del indice seleccionado
 */
```

```
public static int getIndiceSeleccionado(){
  return indiceSeleccionado;
}

/**
 * Asigna el id del indice seleccionado
 * @param indiceSeleccionado el id del indice seleccionado
 */
public static void setIndiceSeleccionado(
    int indiceSeleccionado){
  Datos.indiceSeleccionado = indiceSeleccionado;
}

/**
 * Este metodo regresa el indice indicando su id
 * @param id id del indice
 * @return indice objeto Indice con el id indicado
 */
public static Indice getIndice(int id){
  for(int k=0;k<Datos.indices.size();k++){
    if(Datos.indices.get(k).getId() == id){
      return Datos.indices.get(k);
    }
  }
  return null;
}


/**
 * Regresa el true si hay conexion
 * @return conexion true si hay conexion
 *                  en otro caso false
 */
public static boolean isErrorConexion(){
  return error_conexion;
}

/**
 * Asigna true si hay conexion
 * @return conexion true si hay conexion
 *                  en otro caso false
 */
public static void setErrorConexion(boolean error_conexion){
  Datos.error_conexion = error_conexion;
}

/**
 * Regresa el true si hay conexion
 * @return conexion true si hay conexion
 *                  en otro caso false
 */
public static boolean isErrorJson(){
  return error_json;
}
```

```
  /**
   * Asigna true si hay conexion
   * @return conexion true si hay conexion
   *                  en otro caso false
   */
  public static void setErrorJson(boolean error_json){
    Datos.error_json = error_json;
  }

  /**
   * Limpia los objetos indices y resultados
   */
  public static void clear(){
    Datos.indices.clear();
    Datos.resultados.clear();
  }

}
```

Indice.java

```
package unam.fciencias.isma.tesis;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, representa un indice de busqueda
 * dentro de la aplicacion
 * en el componente Android.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class Indice{

  private String nombre = "";
  private int id = 0;
  private boolean seleccionado = true;
  private int numResultados = 0;

  /**
   * Regresa el nombre del indice
   * @return nombre el nombre del indice
   */
  public String getNombre(){
    return nombre;
  }

  /**
   * Asigna el nombre del indice
   * @param nombre el nombre del indice
   */
  public void setNombre(String nombre){
```

```
  this.nombre = nombre;
}

/**
 * Regresa el id del indice
 * @return id id del indice
 */
public int getId(){
  return id;
}

/**
 * Asigna el id del indice
 * @param id id del indice
 */
public void setId(int id){
    this.id = id;
}

/**
 * Regresa true si el indice esta seleccionado
 * @return seleccionado true si el indice esta seleccionado
 *               false en otro caso
 */
public boolean isSeleccionado(){
  return seleccionado;
}

/**
 * Asigna true si el indice esta seleccionado
 * @param seleccionado true si el indice
 *             esta seleccionado
 *               false en otro caso
 */
public void setSeleccionado(boolean seleccionado){
  this.seleccionado = seleccionado;
}

/**
 * Regresa el numero de coincidencias de este
 * indice
 * @return numResultados el numero de coincidencias
 *               de este indice
 */
public int getNumResultados(){
  return numResultados;
}

/**
 * Asigna el n mero de coincidencias de este
 * indice
 * @param numResultados el numero de
 *             coincidencias de este indice
 */
```

```
  public void setNumResultados(int numResultados){
    this.numResultados = numResultados;
  }

}
```

Indices.java

```
package unam.fciencias.isma.tesis;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.LinearLayout;
import android.widget.CompoundButton.OnCheckedChangeListener;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, genera la pantalla de Indices de
 * la aplicacion
 * por medio de programacion, no se utiliza
 * layout ya que los
 * elementos deben ser creados dinamicamente segun
 * los indices
 * que esten activos.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class Indices extends Activity{

  private LinearLayout contenedor;

  /**
   * @see android.app.Activity#onCreate(android.os.Bundle)
   */
  @Override
  public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        //se genera el layout
        contenedor = new LinearLayout(this);
        contenedor.setLayoutParams(
            new LayoutParams(LayoutParams.FILL_PARENT,
            LayoutParams.FILL_PARENT));
        contenedor.setOrientation(LinearLayout.VERTICAL);

        //genera los checkbox de los indices que esten activos
        for(int i=0;i<Datos.getIndices().size();i++){
          CheckBox cb1 = new CheckBox(this);
            cb1.setText(Datos.getIndices().get(i).getNombre());
```

```
            if(Datos.getIndices().get(i).isSeleccionado()){
              cb1.setChecked(true);
            }
            final int k = i;
            //se genera el listener para el evento de seleccionar
            cb1.setOnCheckedChangeListener(
                new OnCheckedChangeListener(){
              public void onCheckedChanged(
                  CompoundButton buttonView,
                  boolean isChecked){
                    if (isChecked){
                      Datos.getIndices().get(k).
                      setSeleccionado(true);
                      //se incrementa el numero de
                      //indices seleccionados
                      Datos.setnumIndicesSeleccionados(
                        Datos.getnumIndicesSeleccionados()+1);
                    }
                    else{
                      Datos.setnumIndicesSeleccionados(
                        Datos.getnumIndicesSeleccionados()-1);
                      //se asigna el estado no
                      //seleccionado al indice
                      Datos.getIndices().get(k).
                      setSeleccionado(false);
                    }
                }
            });
            contenedor.addView(cb1);
        }
        setContentView(contenedor);
    }

}
```

Inicio.java

```
package unam.fciencias.isma.tesis;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, genera la pantalla de Inicio de
 * la aplicacion
 * por medio del layout inicio, se definen cajas
```

```
 * de texto para que
 * el usuario escriba su nombre y clave para tener
 * acceso a la
 * aplicacion
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class Inicio extends Activity{

    /**
     * @see android.app.Activity#onCreate(android.os.Bundle)
     */
  @Override
    public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    //se carga el layout
        setContentView(R.layout.inicio);
        // obtenemos el boton
        Button boton = (Button)findViewById(R.id.boton);
        //se define un nuevo listener para los
        //eventos del boton
        boton.setOnClickListener(
            new OnClickListener(){
          public void onClick(View view){
              //se obtienen las cajas de texto
                EditText usr =
                  (EditText) findViewById(R.id.et_login);
                EditText pass =
                  (EditText) findViewById(R.id.et_pass);
                //se crea un nuevo objeto para
                //verificar el acceso
                ControladorRespuesta cr =
                  new ControladorRespuesta();
                cr.verificaAcceso(usr.getText().toString(),
                    pass.getText().toString());
                boolean acceso = Datos.isAcceso();
                //se define una nueva alerta
                AlertDialog alertDialog =
                  new AlertDialog.Builder(Inicio.this).create();
          alertDialog.setTitle("Advertencia");
          alertDialog.setButton("Aceptar",
              new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog,
                       int which) {
                           return;
                   }
               });
          if(Datos.isErrorConexion()){
            alertDialog.setMessage(
                "No se puede realizar la conexi n");
            alertDialog.show();
          }
```

```
            //se define un intent para iniciar la actividad
            else if(acceso == true){
                    Intent myIntent = new Intent(view.getContext(),
                            BusquedaPrincipal.class);
                      startActivityForResult(myIntent, 0);
                   }
                   else{
                     alertDialog.setMessage("Usuario no v lido");
                     alertDialog.show();
                   }
                 }
          }); // fin del setOnclickListener
      }

    /**
     * @see android.app.Activity#onActivityResult(int,
     * int, android.content.Intent)
     */
    @Override
    protected void onActivityResult(int requestCode,
        int resultCode, Intent data) {
      super.onActivityResult(requestCode, resultCode, data);
      //cuando se regresa a esta actividad desde
      //una actividad anteior,
      //la aplicacion termina.
      finish();
    }
}
```

Resultado.java

```
package unam.fciencias.isma.tesis;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, representa un resultado de busqueda,
 * para poder
 * presentarlo en forma de lista en la actividad
 * ResultadosBusquedaDetalle
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class Resultado{

  private String titulo;
  private String contenido;
  private boolean visible;

  /**
   * Regresa el titulo del resultado
   * @return titulo el titulo del resultado
   */
```

```
  public String getTitulo(){
    return titulo;
  }

  /**
   * Asigna el titulo del resultado
   * @param titulo el titulo del resultado
   */
  public void setTitulo(String titulo){
    this.titulo = titulo;
  }

  /**
   * Regresa el contenido del resultado
   * @return contenido el contenido del
   *            resultado
   */
  public String getContenido(){
    return contenido;
  }

  /**
   * Asigna el contenido del resultado
   * @param contenido el contenido del
   *                  resultado
   */
  public void setContenido(String contenido){
    this.contenido = contenido;
  }

  /**
   * Regresa true si el contenido es visible
   * @return visible true si el contenido es
   *            visible false en otro caso
   */
  public boolean isVisible(){
    return visible;
  }

  /**
   * Asigna true si el contenido es visible
   * @param visible true si el contenido es visible
   *            false en otro caso
   */
  public void setVisible(boolean visible){
    this.visible = visible;
  }

}
```

ResultadosBusqueda.java

```
package unam.fciencias.isma.tesis;

import java.util.ArrayList;
```

```
import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase, genera la pantalla de numero de
 * coincidencias para cada indice
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class ResultadosBusqueda extends ListActivity{

  private static ArrayList<String> resultados;
  private static ArrayList<Integer> orden;


  /**
   * Este metodo genera un arreglo de resultados
   * para presentarlos en la lista, ademas de una
   * lista
   * con el orden de cada indice
   */
  private static void generaArreglo(){
    resultados = new ArrayList<String>();
    orden = new ArrayList<Integer>();
    int j=0;
    for(int k=0;k<Datos.getIndices().size();k++){
      if(Datos.getIndices().get(k).
          isSeleccionado()){
        String s = Datos.getIndices().get(k).
        getNombre()+" ("+
        Datos.getIndices().get(k).
        getNumResultados()+")";
        resultados.add(j,s);
        orden.add(j,new Integer(
            Datos.getIndices().get(k).getId()));
        j++;
      }
    }
  }


  /**
   * @see android.app.Activity#onCreate(android.os.Bundle)
   */
  @Override
    public void onCreate(Bundle savedInstanceState){
```

```
        super.onCreate(savedInstanceState);
        ResultadosBusqueda.generaArreglo();
        //se usa un ListAdapter predefinido que pone los
        //elementos de la lista
        //en TextViews
        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1,
                ResultadosBusqueda.resultados));
        getListView().setTextFilterEnabled(true);
    }

    /**
     * @see android.app.ListActivity#onListItemClick(
     * android.widget.ListView, android.view.View, int, long)
     */
    @Override
    protected void onListItemClick(ListView lv, View v,
        int posicion,long id){
      //se obtiene el item seleccionado
      int iid = orden.get(posicion).intValue();
      Datos.setIndiceSeleccionado(Datos.getIndice(iid).
          getId());
      //si el n mero de resultados es 0, se regresa a
      //la actividad anteior
      //si es diferente de 0 , se inicia la actividad
      if(Datos.getIndice(iid).getNumResultados() != 0){
        Intent myIntent = new Intent(v.getContext(),
            ResultadosBusquedaDetalle.class);
            startActivityForResult(myIntent, 0);
      }
      else{
        finish();
      }
     }

}
```

ResultadosBusquedaDetalle.java

```
package unam.fciencias.isma.tesis;

import android.app.AlertDialog;
import android.app.ListActivity;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

/**
```

```
 * PROYECTO TLATEMOANI
 *
 * Esta clase, genera la pantalla de numero de
 * coincidencias para cada indice
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class ResultadosBusquedaDetalle extends ListActivity{

  /**
   * @see android.app.Activity#onCreate(android.os.Bundle)
   */
  @Override
  public void onCreate(Bundle savedInstanceState){
          super.onCreate(savedInstanceState);
        //se define una nueva alerta
            AlertDialog alertDialog =
              new AlertDialog.Builder(
                  ResultadosBusquedaDetalle.this).create();
      alertDialog.setTitle("Advertencia");
      alertDialog.setButton("Aceptar",
          new DialogInterface.OnClickListener() {
               public void onClick(
                   DialogInterface dialog, int which){
                      finish();
               }
           });
          //se obtienen los resultados
          final ControladorRespuesta cr =
            new ControladorRespuesta();
          cr.obtieneResultadosDetalle(3, Datos.getQuery(),
              Datos.getIndiceSeleccionado()+"");
          if(Datos.isErrorConexion() || Datos.isErrorJson()){
            alertDialog.setMessage("No se realiz  la conexi n");
            alertDialog.show();
          } else{
          // se usa el adaptador propio
          setListAdapter(new AdaptadorLista(this));
          }
      }

      /**
       * @see android.app.ListActivity#onListItemClick(
       * android.widget.ListView, android.view.View, int, long)
       */
      @Override
      protected void onListItemClick(ListView lv, View v,
          int posicion, long id){
         ((AdaptadorLista)getListAdapter()).toggle(posicion);
      }

      /**
```

```
 * Clase BaseAdapter que presenta el contenido
 *
 */
private class AdaptadorLista extends BaseAdapter{

  private Context contexto;

  /**
   * Constructor de Clase
   * @param contexto objeto Context
   */
  public AdaptadorLista(Context contexto){
        this.contexto = contexto;
    }

    /**
     * @see android.widget.Adapter#getCount()
     */
  @Override
    public int getCount(){
      return Datos.getResultados().size();
    }

    /**
     * @see android.widget.Adapter#getItem(int)
     */
  @Override
    public Object getItem(int posicion){
        return posicion;
    }

    /**
     * @see android.widget.Adapter#getItemId(int)
     */
  @Override
    public long getItemId(int posicion){
        return posicion;
    }

    /**
     * @see android.widget.Adapter#getView(int,
     * android.view.View, android.view.ViewGroup)
     */
  @Override
    public View getView(int posicion, View v,
        ViewGroup padre){
        ContenidoVista cV;
        if (v == null){
          //se crea la vista por medio de la clase privada
          //con los datos estticos de la clase Datos
          cV = new ContenidoVista(contexto,
              Datos.getResultados().get(posicion).
              getTitulo(),
              Datos.getResultados().get(posicion).
```

```
                getContenido(),
                Datos.getResultados().get(posicion).
                isVisible());
        } else{
          //si no se puede obtener el contexto
          cV = (ContenidoVista)v;
          cV.setTitulo(Datos.getResultados().
              get(posicion).getTitulo());
          cV.setContenido(Datos.getResultados().
              get(posicion).getContenido());
          cV.setDesplegado(Datos.getResultados().
              get(posicion).isVisible());
        }
        return cV;
    }

    /**
     * Este metodo cambia el estado del
     * contenido. lo que permite quse se
    * haga visible.
     * @param posicion posicion en la lista
     */
    public void toggle(int posicion){
      Datos.getResultados().get(posicion).
      setVisible(!Datos.getResultados().
          get(posicion).isVisible());
        notifyDataSetChanged();
    }
}

/**
 * Esta clase se utiliza para desplegar la
 * informacion. Se compone
 * de un layout y dos campos de texto
 */
private class ContenidoVista extends LinearLayout{

  private TextView vTitulo;
    private TextView vContenido;

    /**
     * Costructor de Clase
     * @param contexto objeto Context
     * @param titulo titulo del item
     * @param contenido contenido del item
     * @param desplegado true si esta desplegado
     *            false en otro caso
     */
    public ContenidoVista(Context contexto,
        String titulo,
        String contenido,boolean desplegado){
        super(contexto);
        //se define de manera vertical el orden de
        //los elementos en el layout
```

```
        this.setOrientation(VERTICAL);
        //se ponen propiedades del texto de titulo
        vTitulo = new TextView(contexto);
        vTitulo.setTextSize(16);
        vTitulo.setBackgroundColor(Color.BLACK);
        vTitulo.setText(titulo);
        addView(vTitulo,
            new LinearLayout.LayoutParams(
              LayoutParams.FILL_PARENT,
              LayoutParams.WRAP_CONTENT));
        //se ponen propoiedades del
        //texto del contenido
        vContenido = new TextView(contexto);
        vContenido.setText(contenido);
        vContenido.setTextColor(
            Color.BLACK);
        vContenido.setBackgroundColor(
            Color.LTGRAY);
        vContenido.setTextSize(13);
        addView(vContenido,
            new LinearLayout.LayoutParams(
              LayoutParams.FILL_PARENT,
              LayoutParams.WRAP_CONTENT));
        //se define la visibilidad de acuerdo
        //al atributo desplegado
        vContenido.
        setVisibility(desplegado ? VISIBLE : GONE);
    }

    /**
     * Asigna el titulo del item
     * @param titulo el titulo del item
     */
    public void setTitulo(String titulo){
        vTitulo.setText(titulo);
    }

    /**
     * Asigna el contenido del item
     * @param contenido el contenido del item
     */
    public void setContenido(String contenido){
      vContenido.setText(contenido);
    }

    /**
     * Asigna true si es visible
     * @param desplegado true si es visible
     *               false en otro caso
     */
    public void setDesplegado(boolean desplegado){
      vContenido.
      setVisibility(desplegado ? VISIBLE : GONE);
    }
```

```
    }

  }
```

R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package unam.fciencias.isma.tesis;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ayuda=0x7f020000;
        public static final int icon=0x7f020001;
        public static final int icon2=0x7f020002;
        public static final int icons=0x7f020003;
        public static final int indices=0x7f020004;
        public static final int logo=0x7f020005;
        public static final int logo_principal=0x7f020006;
        public static final int salir=0x7f020007;
    }
    public static final class id {
        public static final int AbsoluteLayout01=0x7f050006;
        public static final int ButtonBusqueda1=0x7f050005;
        public static final int EditTextBusqueda1=0x7f050004;
        public static final int ImageView01=0x7f05000d;
        public static final int LinearLayout01=0x7f050000;
        public static final int absoluteLayoutBusqueda1=0x7f050002;
        public static final int boton=0x7f05000c;
        public static final int et_login=0x7f050009;
        public static final int et_pass=0x7f05000b;
        public static final int imagenBusqueda1=0x7f050003;
        public static final int iv_logotipo=0x7f050007;
        public static final int login_button=0x7f050013;
        public static final int password_text=0x7f050011;
        public static final int tv_1=0x7f050001;
        public static final int tv_login=0x7f050008;
        public static final int tv_pass=0x7f05000a;
        public static final int txt_password=0x7f050012;
        public static final int txt_username=0x7f050010;
        public static final int username_text=0x7f05000e;
        public static final int welcome_text=0x7f05000f;
    }
    public static final class layout {
        public static final int ayuda=0x7f030000;
        public static final int busqueda_1=0x7f030001;
        public static final int inicio=0x7f030002;
        public static final int main=0x7f030003;
```

```
    }
    public static final class string {
        public static final int app_name=0x7f040000;
    }
}
```

ConexionSQL.java

```
package unam.fciencias.isma.tesis;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import com.mysql.jdbc.jdbc2.optional.MysqlDataSource;

/**
 * PROYECTO TLATEMOANI
 *
 * Esta clase permite la conexion a la base de datos MySQL donde
 * se encuentran los datos para la aplicacion.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 *
 */
public class ConexionSQL{

  private Connection conexion;
  private Statement stm;

  /**
   * Constructor de Clase
   */
  public ConexionSQL(){
    MysqlDataSource mds = new MysqlDataSource();
    mds.setUser("ismael");
    mds.setPassword("123456789");
    mds.setDatabaseName("tesis");
    mds.setServerName("localhost");
    try{
      this.conexion = mds.getConnection();
      stm = conexion.createStatement();
    } catch(SQLException e){
      e.printStackTrace();
    }
  }

  /**
   * Este metodo regresa el valor booleano true si el
   * usuario esta registrado,
   * en otro caso regresa false
   * @param usuario el usuario
```

```
 * @param clave la clave de acceso
 * @return respuesta un boolean, true si esta registrado,
 *                  en otro caso false
 */
public boolean estaRegistrado(String usuario, String clave){
  boolean resultado = false;
  try{
    //se ejecuta la consulta sobre la tabla acceso
    ResultSet rs = stm.executeQuery(
        "select id_acceso from acceso where " +
        "usuario='"+usuario+"' and clave='"+clave+"' ");
    //si esta el usuario, resultado es true, si no es false.
    resultado = (rs.first()) ? true : false;
    rs.close();
  } catch(SQLException e){
    e.printStackTrace();
  }
  return resultado;
}

/**
 * Este metodo regresa una lista con los datos de cada
 * indice definido en la base de datos.
 * @return lista lista de datos de los  ndices
 */
public ArrayList<ArrayList<String>> getIndicesString(){
  ArrayList<ArrayList<String>> resultado =
    new ArrayList<ArrayList<String>>();
  try{
    ResultSet rs = stm.executeQuery("select * from indices");
    while (rs.next()){
      ArrayList<String> aux = new ArrayList<String>();
      aux.add(rs.getInt("id_indice")+"");
      aux.add((String)rs.getObject("nombre"));
      aux.add((String)rs.getObject("core"));
      aux.add(rs.getInt("estado")+"");
      resultado.add(aux);
    }
    rs.close();
  } catch(SQLException e){
    e.printStackTrace();
  }
  return resultado;
}

/**
 * Este metodo regresa una lista con los datos de cada campo
 * definido en la base de datos.
 * @param indice id del indice
 * @return lista lista lista de datos de los campos
 */
public ArrayList<ArrayList<String>> getCamposString(int indice){
  ArrayList<ArrayList<String>> resultado =
    new ArrayList<ArrayList<String>>();
```

```
    try{
      ResultSet rs = stm.executeQuery(
          "select * from campos "+
          "where  id_indice='"+indice+"'");
      while (rs.next()){
        ArrayList<String> aux = new ArrayList<String>();
        aux.add(rs.getInt("id_campo")+"");
        aux.add((String)rs.getObject("nombre"));
        aux.add((String)rs.getObject("nombreVisible"));
        aux.add(rs.getInt("visible")+"");
        aux.add(rs.getInt("estado")+"");
        aux.add(rs.getInt("id_indice")+"");
        resultado.add(aux);
      }
      rs.close();
    } catch (SQLException e){
      e.printStackTrace();
    }
    return resultado;
  }

  /**
   * Metodo para cerrar la conexion
   */
  public void clear(){
    try{
      stm.close();
      conexion.close();
    } catch(SQLException e){
      e.printStackTrace();
    }
  }

}
```

accesoServlet.java

```
package unam.fciencias.isma.tesis;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import net.sf.json.JSONObject;

/**
 * PROYECTO TLATEMOANI
 *
 * Este servlet regresa una respuesta JSON, se utiliza para verificar
 * que el usuarioeste registrado en la base de datos.
 *
 * El metodo es muy simple y solo es un ejemplo de como verificar
 * los datos,un metodo que proporcione mas seguridad para el acceso de
```

```
 * los datos debera ser implementado si se utilizan datos para una
 * solucion real.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version     %I%, %G%
 * @since       1.0
 */
public class accesoServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  private static final String CONTENT_TYPE = "application/json";
    private ConexionSQL csql;

    /**
     * Constructor de Clase
     */
    public accesoServlet() {
        super();
        csql = new ConexionSQL();
    }

  /**
   * @see javax.servlet.http.HttpServlet#doGet(
   * javax.servlet.http.HttpServletRequest,
   * javax.servlet.http.HttpServletResponse)
   */
  protected void doGet(HttpServletRequest request,
      HttpServletResponse response)
    throws ServletException, IOException{
    String usuario = "";
    String clave = "";
    JSONObject json = new JSONObject();
    try{
      usuario = request.getParameter("usr");
      clave = request.getParameter("clv");
    }
    catch(Exception e){
      e.printStackTrace();
    }
    //se define el tipo de respuesta
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    if(usuario != null && clave !=null){
      boolean respuesta= false;
      //se utiliza un metodo de la clase ConexionSQL
      respuesta = (csql.estaRegistrado(usuario, clave))?true:false;
      //se regresa el resultado JSON como una cadena
      json.put("acceso", respuesta);
      out.println(json);
    }
    out.close();
  }


  /**
```

```
   * @see javax.servlet.http.HttpServlet#doPost(
   * javax.servlet.http.HttpServletRequest,
   * javax.servlet.http.HttpServletResponse)
   */
  protected void doPost(HttpServletRequest request,
      HttpServletResponse response)
  throws ServletException, IOException {

  }

}
```

indicesServlet.java

```
package unam.fciencias.isma.tesis;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.StringTokenizer;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import net.sf.json.JSONObject;
import unam.fciencias.isma.tesis.solr.ControladorIndices;
import unam.fciencias.isma.tesis.solr.EjecutorQuery;
import unam.fciencias.isma.tesis.solr.Indice;
import unam.fciencias.isma.tesis.solr.ResultadoQuery;

/**
 * PROYECTO TLATEMOANI
 *
 * Este servlet regresa objetos JSON dependiendo del metodo que se
 * pase como parametro:
 *
 *  1 - Se regresa la informacion correspondiente a los indices disponibles
 *  2 - Se regresa informacion sobre el numero de resultados sobre los
 *      indices seleccionados
 *  3 - Regresa los resultados de una consulta sobre un indice especifico
 *
 *  La respuesta de cada uno de los metodos es un objeto JSON.
 *
 * @author Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class indicesServlet extends HttpServlet{

  private static final long serialVersionUID = 1L;
  private static final String CONTENT_TYPE = "application/json";
    private ControladorIndices ci;

    /**
     * Constructor de Clase
```

```
     */
    public indicesServlet(){
        super();
        ci = new ControladorIndices();
        ci.construyeIndices();
    }

    /**
     * Metodo utilizado para convertir una cadena del tipo
     * 1a3a2
     * a objetos Indice, donde cada numero es el id del indice
     * @param indices cadena codificada de  ndices
     * @return lista lista de indices
     */
    public ArrayList<Indice> convierteIndices(String indices){
      //divide la cadena utilizando el caracter 'a' como separador
    StringTokenizer st = new StringTokenizer(indices,"a");
    ArrayList<Indice> indicesBusqueda = new ArrayList<Indice>();
    while(st.hasMoreTokens()){
            int indice = Integer.parseInt(st.nextToken());
      //se revisa que sea el  ndice
            for(int k=0;k<this.ci.getIndices().size();k++){
              if(ci.getIndices().get(k).getId() == indice){
                indicesBusqueda.add(ci.getIndices().get(k));
              }
            }
    }
    return indicesBusqueda;
}

    /**
     * Este metodo regresa la informacion acerca de los indices
     * disponibles para la busqueda
     *
     * @return json objeto JSON
     */
    public JSONObject regresaIndices(){
      JSONObject jsona = new JSONObject();
      ArrayList<Indice> indices = ci.getIndices();
      for(int i=0;i<indices.size();i++){
        //si esta activo
        if(indices.get(i).getEstado() == 1){
          jsona.put(indices.get(i).getNombre(),indices.get(i).getId());
        }
      }
      return jsona;
    }

    /**
     * Este metodo se utiliza para regresar el objeto JSON
     * que contiene la informacion acerca del numero de coincidencias
     *
     * @param indicesBusqueda lista de indices de b squeda
     * @param query la consulta
```

```
   * @return json objeto JSON
   */
  public JSONObject regresaResultadosIndices(
      ArrayList<Indice> indicesBusqueda,
      String query){
    JSONObject json = new JSONObject();
    EjecutorQuery ejecutor = new EjecutorQuery();
    ResultadoQuery resultados = new ResultadoQuery();
    for(int i=0;i<indicesBusqueda.size();i++){
      String index = indicesBusqueda.get(i).getNombre();
      ejecutor.ejecutaQuery(indicesBusqueda.get(i), query);
      resultados.generaSolrDocumentList(ejecutor.getQueryResponse());
      long numResultados = resultados.getNumResultados();
      json.put(index, numResultados);
    }
    return json;
  }

  /**
   * Este metodo se utiliza para regresar el objeto JSON
   * que contiene la informacion de los resultados
   *
   * @param idIndice id del indice
   * @param query la consulta
   * @return json objeto JSON
   */
  public JSONObject regresaResultado(int idIndice,String query){
    JSONObject json = new JSONObject();
    EjecutorQuery ejecutor = new EjecutorQuery();
    ResultadoQuery resultados = new ResultadoQuery();
    for(int i=0;i<ci.getIndices().size();i++){
      if(ci.getIndices().get(i).getId() == idIndice){
        ejecutor.ejecutaQuery(ci.getIndices().get(i), query);
        resultados.generaSolrDocumentList(ejecutor.getQueryResponse());
        json.put("resultados",resultados.getSdl());
      }
    }
    return json;
  }

  /**
 * @see HttpServlet#doGet(HttpServletRequest request,
 * HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
  throws ServletException, IOException{
  int metodo = 0;
  String query = "";
  String indices = "";
  String i = "";
  int indice = 0;
  try{
    //se recuperan los datos de la solicitud
```

```
    String m = request.getParameter("mtd");
    metodo = Integer.parseInt(m);
    query = request.getParameter("qry");
    indices = request.getParameter("idxs");
    i = request.getParameter("idx");
    if(i != null){
      indice = Integer.parseInt(i);
    }
  }
  catch(Exception e){
    e.printStackTrace();
  }
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();
  //seg n el metodo se realizan las funciones
  switch(metodo){
    case 1:
      //regresa la informacion de indices disponibles
      JSONObject json1 = this.regresaIndices();
      out.println(json1);
      break;
    case 2:
      //regresa la informaci n de coincidencias
      JSONObject json2 = new JSONObject();
      if(!query.isEmpty() && !indices.isEmpty()){
        //pone los indices de busqueda seg n el
        //string que se pasa como parametro
        ArrayList<Indice> indicesBusqueda =
          convierteIndices(indices);
        json2 = this.regresaResultadosIndices(
            indicesBusqueda,query);
      }
      out.println(json2);
      break;
    case 3:
      //se regresan los resultados de la busqueda
      JSONObject json3 = new JSONObject();
      if(!query.isEmpty() && !i.isEmpty()){
        json3 = this.regresaResultado(indice,query);
      }
      out.println(json3);
      break;
    default:
      break;
  }
  out.close();
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request,
 * HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request,
    HttpServletResponse response)
```

```
    throws ServletException, IOException{

  }

}
```

Campo.java

```
package unam.fciencias.isma.tesis.solr;

/**
 * PROYECTO TLATEMAONI
 *
 * Esta clase define el concepto de Campo, para los indices de busqueda
 * en Tlatemoani dentro del componente Solr.
 *
 * @author Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class Campo{

  private int id;
  private String nombre;
  private String nombreVista;
  private int visible;
  private int estado;
  private int id_indice;


  /**
   * Regresa identificador del campo
   * @return id id del campo
   */
  public int getId(){
    return id;
  }

  /**
   * Asigna identificador del campo
   * @param id id del campo
   */
  public void setId(int id){
    this.id = id;
  }

  /**
   * Regresa nombre del campo
   * @return nombre nombre del campo
   */
  public String getNombre(){
    return nombre;
  }

  /**
```

```
 * Asigna nombre del campo
 * @param nombre nombre del campo
 */
public void setNombre(String nombre){
  this.nombre = nombre;
}

/**
 * Regresa el nombre para la vista del campo
 * @return nombreVista Nombre de la vista
 */
public String getNombreVista(){
  return nombreVista;
}

/**
 * Asigna el nombre de la vista del campo
 * @param nombreVista Nombre de la vista
 */
public void setNombreVista(String nombreVista){
  this.nombreVista = nombreVista;
}

/**
 * Regresa entero indica si es visible
 * @return visible 1 si es visible
 */
public int getVisible(){
  return visible;
}

/**
 * Asigna entero indica si es visible
 * @param visible 1 si es visible
 */
public void setVisible(int visible){
  this.visible = visible;
}

/**
 * Regresa estado del campo
 * @return estado 1 si esta activo
 */
public int getEstado(){
  return estado;
}

/**
 * Asigna estado del campo
 * @param estado 1 si est  activo
 */
public void setEstado(int estado){
  this.estado = estado;
}
```

```
  /**
   * Regresa id del indice al que pertenece el campo
   * @return id id del  ndice
   */
  public int getId_indice(){
    return id_indice;
  }

  /**
   * Asigna id del indice al que pertenece el campo
   * @param idIndice id del indice
   */
  public void setId_indice(int idIndice){
    id_indice = idIndice;
  }

}
```

ControladorIndices.java

```
package unam.fciencias.isma.tesis.solr;

import java.util.ArrayList;
import unam.fciencias.isma.tesis.ConexionSQL;

/**
 * PROYECTO TLATEMAONI
 *
 * Esta clase se utiliza para construir la lista
 * de indices
 * obtiene la informacion desde la base de datos.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class ControladorIndices{

  private ArrayList<Indice> indices;
  private ConexionSQL bc;

  /**
   * Constructor de Clase
   */
  public ControladorIndices(){
    this.bc = new ConexionSQL();
    this.indices = new ArrayList<Indice>();
  }

  /**
   * Regresa la lista de indices
   * @return indices lista de indices
   */
  public ArrayList<Indice> getIndices() {
```

```
    return indices;
  }

  /**
   * Asigna la lista de indices
   * @param indices lista de indices
   */
  public void setIndices(ArrayList<Indice> indices) {
    this.indices = indices;
  }

  /**
   * Construye los indices desde la base de datos
   * utilizando
   * un objeto de la clase ConexionSQL
   * @return indices lista de indices
   */
  public ArrayList<Indice> construyeIndices(){
    ArrayList<ArrayList<String>> listaIndices =
      bc.getIndicesString();
    for(int i=0;i<listaIndices.size();i++){
      ArrayList<String> camposIndice =
        listaIndices.get(i);
      Indice auxIndice = new Indice();
      auxIndice.setId(
          Integer.parseInt(camposIndice.get(0)));
      auxIndice.setNombre(camposIndice.get(1));
      auxIndice.setCore(camposIndice.get(2));
      auxIndice.setEstado(
          Integer.parseInt(camposIndice.get(3)));

      ArrayList<ArrayList<String>> listaCampos =
        bc.getCamposString(auxIndice.getId());
      ArrayList<Campo> nuevaListaCampos =
        new ArrayList<Campo>();
      for(int j=0;j<listaCampos.size();j++){
        ArrayList<String> campos = listaCampos.get(j);
        Campo campo = new Campo();
        campo.setId(Integer.parseInt(campos.get(0)));
        campo.setNombre(campos.get(1));
        campo.setNombreVista(campos.get(2));
        campo.setVisible(Integer.parseInt(campos.get(3)));
        campo.setEstado(Integer.parseInt(campos.get(4)));
        campo.setId_indice(
            Integer.parseInt(campos.get(5)));
        nuevaListaCampos.add(campo);
      }
      auxIndice.setCampos(nuevaListaCampos);
      this.indices.add(auxIndice);
    }
    return this.indices;
  }

}
```

EjecutorQuery.java

```
package unam.fciencias.isma.tesis.solr;

import java.net.MalformedURLException;

import org.apache.solr.client.solrj.SolrQuery;
import org.apache.solr.client.solrj.SolrServerException;
import org.apache.solr.client.solrj.impl.CommonsHttpSolrServer;
import org.apache.solr.client.solrj.response.QueryResponse;


/**
 * PROYECTO TLATEMAONI
 *
 * Esta clase ejecuta la consulta en el servidor Solr,
 * en el
 * core que se indica en el metodo ejecutaQuery.
 *
 * @author      Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class EjecutorQuery{

  private static final String BASE_URL_DEFAULT =
    "http://localhost:8983/solr/Directorio";
  private final int numRegistros = 30;
  private CommonsHttpSolrServer servidorSolr;
  private QueryResponse queryResponse;

  /**
   * Constructor de Clase
   */
  public EjecutorQuery(){
    this.queryResponse = new QueryResponse();
        try{
          this.servidorSolr =
            new CommonsHttpSolrServer(
              EjecutorQuery.BASE_URL_DEFAULT);
            } catch(MalformedURLException ex){
              ex.printStackTrace();
          }
  }


  /**
   * Regresa un objeto QueryResponse
   * @return queryResponse objeto QueryResponse
   */
  public QueryResponse getQueryResponse(){
    return queryResponse;
  }
```

```
  /**
   * Asigna un objeto QueryResponse
   * @param queryResponse objeto QueryResponse
   */
  public void setQueryResponse(QueryResponse
      queryResponse){
    this.queryResponse = queryResponse;
  }


  /**
   * Metodo ejecuta una consulta en el servidor
   * Solr, en el
   * indice que se indica
   * @param indice objeto Indice
   * @param query cadena que representa la consulta
   */
  public void ejecutaQuery(Indice indice,String query){
     String core = indice.getCore();
     SolrQuery solrQuery = new SolrQuery(query);
     solrQuery.setRows(this.numRegistros);
     this.servidorSolr.setBaseURL(core);
     this.queryResponse = new QueryResponse();
     try{
      queryResponse =
        this.servidorSolr.query(solrQuery);
    } catch(SolrServerException e){
      e.printStackTrace();
    }
   }

}
```

Indice.java

```
package unam.fciencias.isma.tesis.solr;

import java.util.ArrayList;


/**
 * PROYECTO TLATEMAONI
 *
 * Esta clase define el concepto de Idice, para los
 * indices de busqueda
 * en Tlatemoani dentro del componente Solr.
 *
 * @author Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class Indice{

  private int id;
  private String nombre;
```

```
private String core;
private int estado;
private ArrayList<Campo> campos;


/**
 * Regresa la lista de campos
 * @return campos lista de campos
 */
public ArrayList<Campo> getCampos(){
  return campos;
}

/**
 * Asigna la lista de campos
 * @param campos lista de campos
 */
public void setCampos(ArrayList<Campo> campos){
  this.campos = campos;
}

/**
 * Regresa id del indice
 * @return id id del indice
 */
public int getId(){
  return id;
}

/**
 * Asigna el id del indice
 * @param id id del indice
 */
public void setId(int id){
  this.id = id;
}

/**
 * Regresa el nombre del indice
 * @return nombre nombre del indice
 */
public String getNombre(){
  return nombre;
}

/**Asigna el nombre del indice
 * @param nombre nombre del indice
 */
public void setNombre(String nombre){
  this.nombre = nombre;
}

/**
 * Regresa el core del indice
```

```
   * @return core core del indice
   */
  public String getCore(){
    return core;
  }

  /**
   * Asigna el core del indice
   * @param core core del indice
   */
  public void setCore(String core){
    this.core = core;
  }

  /**
   * Regresa el estado del indice
   * @return estado 1 si esta activo
   */
  public int getEstado(){
    return estado;
  }

  /**
   * Asigna el estado del indice
   * @param estado 1 si esta activo
   */
  public void setEstado(int estado){
    this.estado = estado;
  }
}
```

ResultadoQuery.java

```
package unam.fciencias.isma.tesis.solr;

import org.apache.solr.client.solrj.response.QueryResponse;
import org.apache.solr.common.SolrDocumentList;

/**
 * PROYECTO TLATEMAONI
 *
 * Esta clase maneja el resultado de una consulta al
 * servidor Solr.
 * La clase debe ser utilizada en conjunto con EjecutorQuery.
 *
 * @author Ismael Jonas Galvan Hernandez
 * @version      %I%, %G%
 * @since        1.0
 */
public class ResultadoQuery{

  private SolrDocumentList sdl;

  /**
   * Regresa un objeto SolrDocumentList
```

```
 * @return sdl objeto SolrDocumentList
 */
public SolrDocumentList getSdl(){
  return sdl;
}

/**
 * Este metodo obtiene los resultados de un objeto
 * QueryResponse de Solr
 * @param queryResponse objeto SolrDocumentList
 */
public void generaSolrDocumentList(QueryResponse
    queryResponse){
  this.sdl = queryResponse.getResults();
}

/**
 * Obtiene el numero de coincidencias del resultado
 * @return numero el numero de resultados
 */
public long getNumResultados(){
  return this.sdl.getNumFound();
}


}
```

Apéndice C

Pruebas del Sistema

Los resultados de las pruebas diseñadas en la sección 4.4 se muestran a continuación:

I. Caso de prueba **Ingreso**:

- *El usuario introduce nombre y clave*:

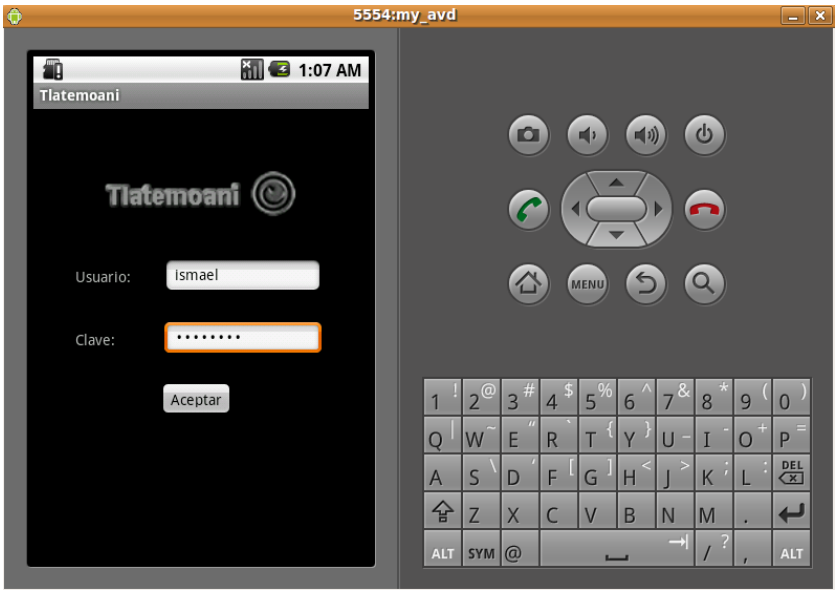

Figura C.1: Caso de Prueba 1-1

- *Se muestra la pantalla inicial de búsqueda*:

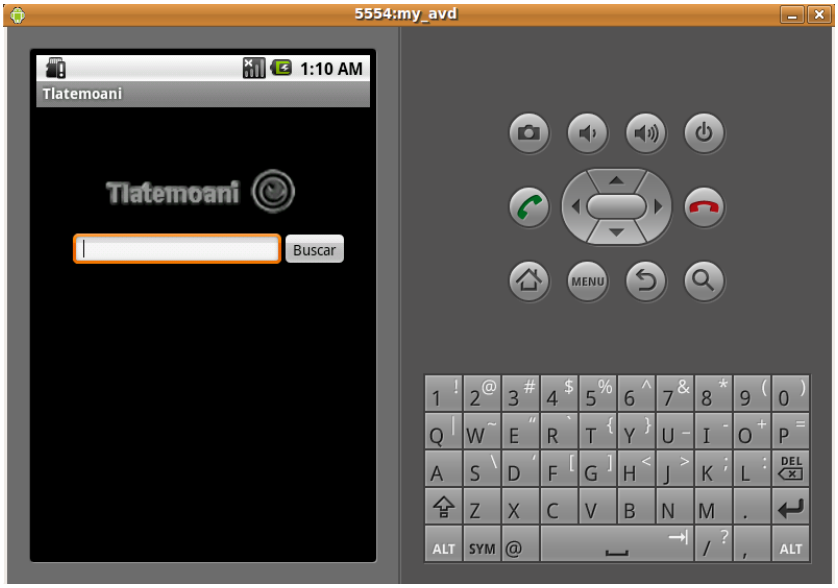

Figura C.2: Caso de Prueba 1-2

- *Alternativa – Salir con Home*:

Figura C.3: Caso de Prueba 1-3

- *Error – Usuario No válido*:

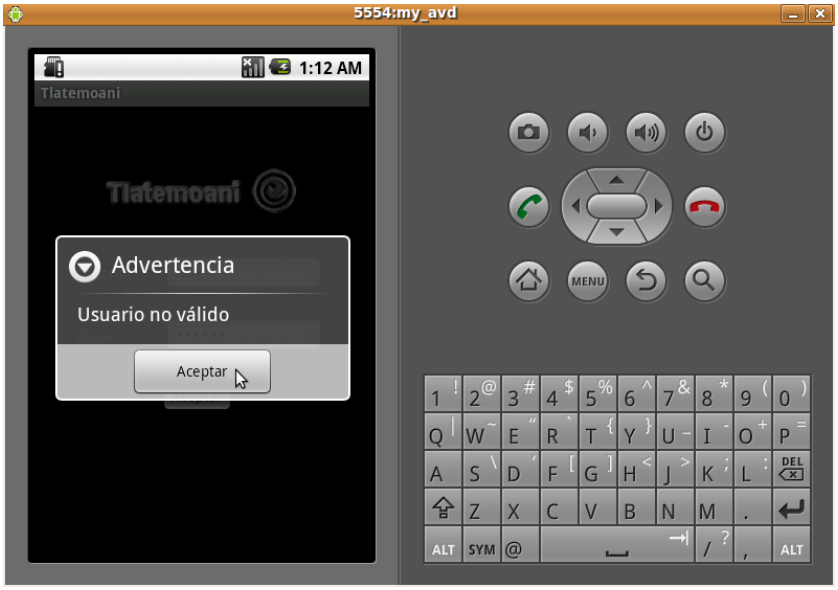

Figura C.4: Caso de Prueba 1-4

- *Error – Sin Conexión*:

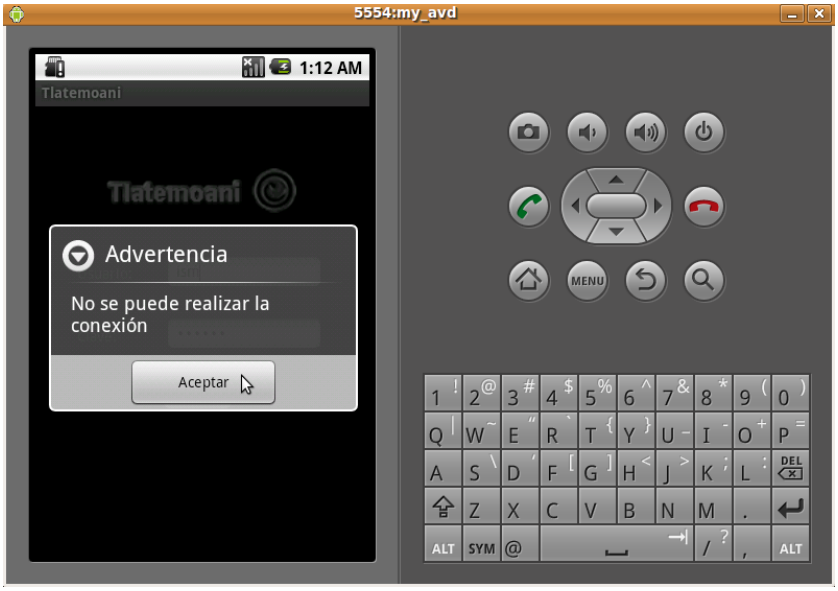

Figura C.5: Caso de Prueba 1-5

II. Caso de prueba **Ayuda**:

- *El usuario presiona el botón* ***menu***:

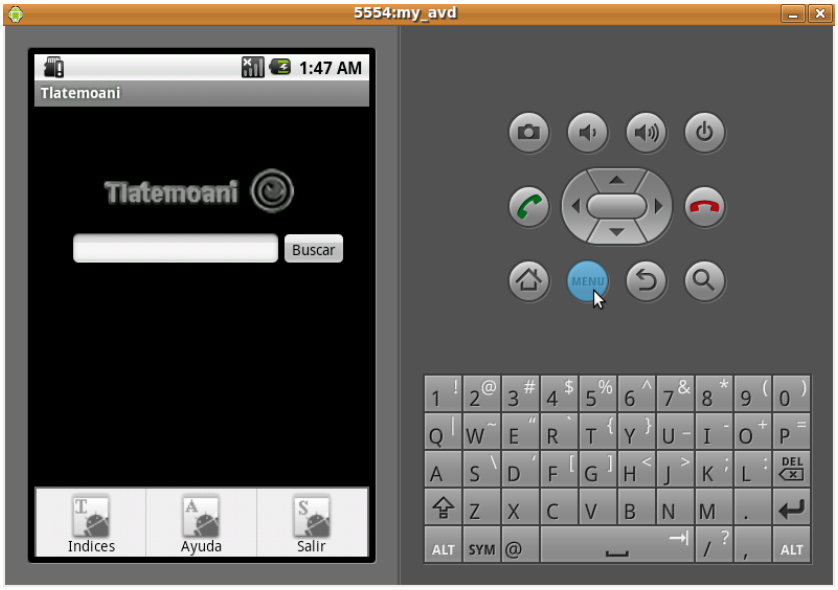

Figura C.6: Caso de Prueba 2-1

- *El usuario elige la opción "Ayuda"*:

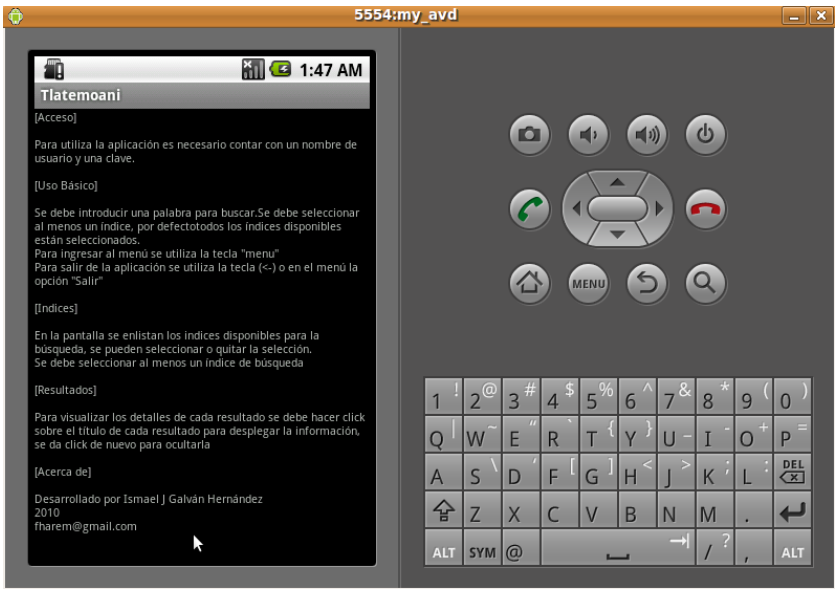

Figura C.7: Caso de Prueba 2-2

- *Alternativa – El usuario presiona el botón para regresar*:

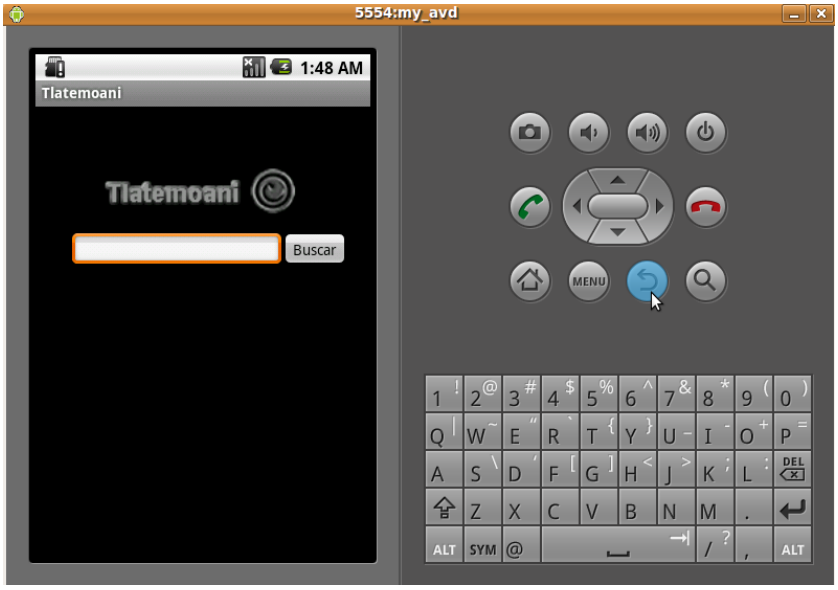

Figura C.8: Caso de Prueba 2-3

III. Caso de prueba **Selección**:

- *El usuario selecciona la opción Indices del "menu"*:

Figura C.9: Caso de Prueba 3-1

- *El sistema muestra los índices*:

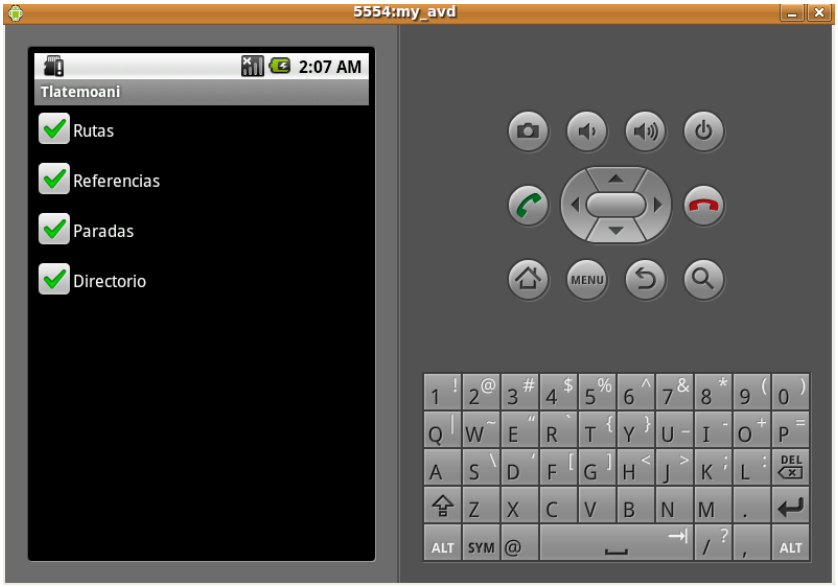

Figura C.10: Caso de Prueba 3-2

- *El usuario marca o desmarca los índices*:

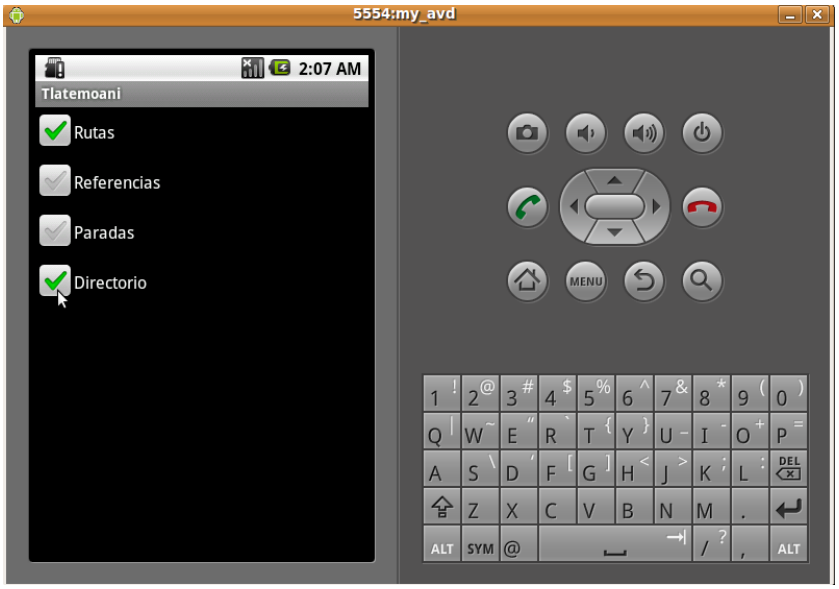

Figura C.11: Caso de Prueba 3-3

- *Alternativa – Salir con Home*: Ver Figura C.3
- *Alternativa – El usuario presiona el botón* ***menu***: Ver Figura C.6
- *Alternativa – El usuario presiona el botón para regresar*: Ver Figura C.8

IV. Caso de prueba **Búsqueda**:

- *El usuario escribe una palabra y presiona el botón "Buscar"*:

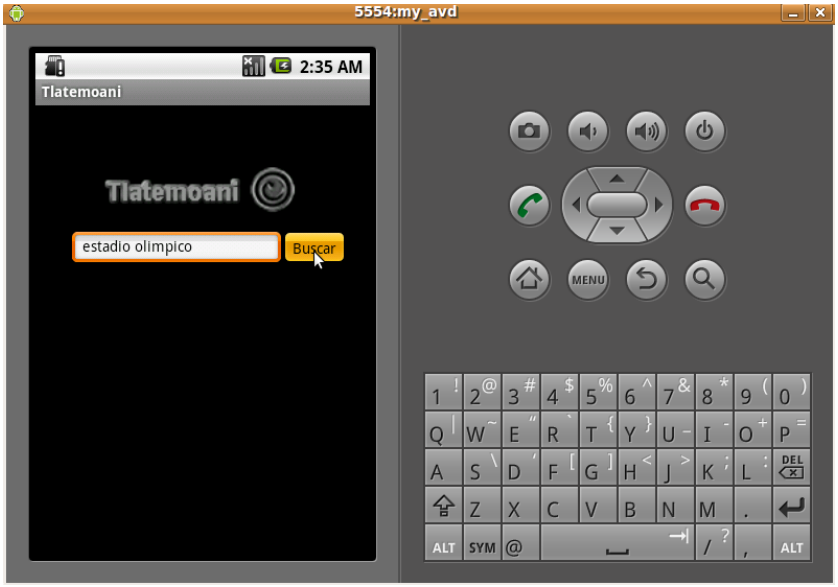

Figura C.12: Caso de Prueba 4-1

- *El sistema muestra el número de coincidencias de la búsqueda*:

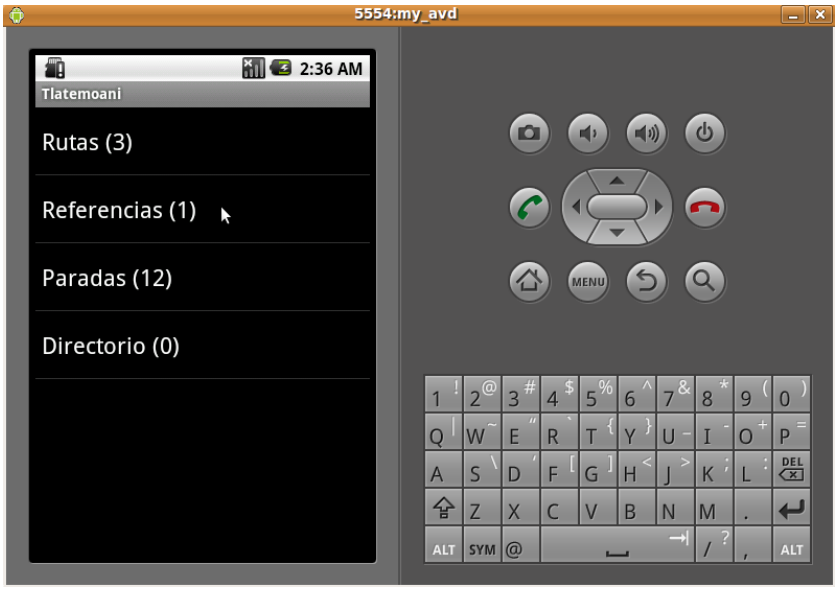

Figura C.13: Caso de Prueba 4-2

- *Alternativa – Salir con Home*: Ver Figura C.3
- *Alternativa – El usuario presiona el botón "menu"*: Ver Figura C.6
- *Alternativa – El usuario presiona el botón para regresar*: Ver Figura C.8
- *Error – Sin conexión*: Ver Figura C.5
- *Error – Si no se escribe una consulta se muestra un mensaje*:

Figura C.14: Caso de Prueba 4-7

- *Error – Si no está seleccionado ningún índice se escribe una consulta se muestra un mensaje*:

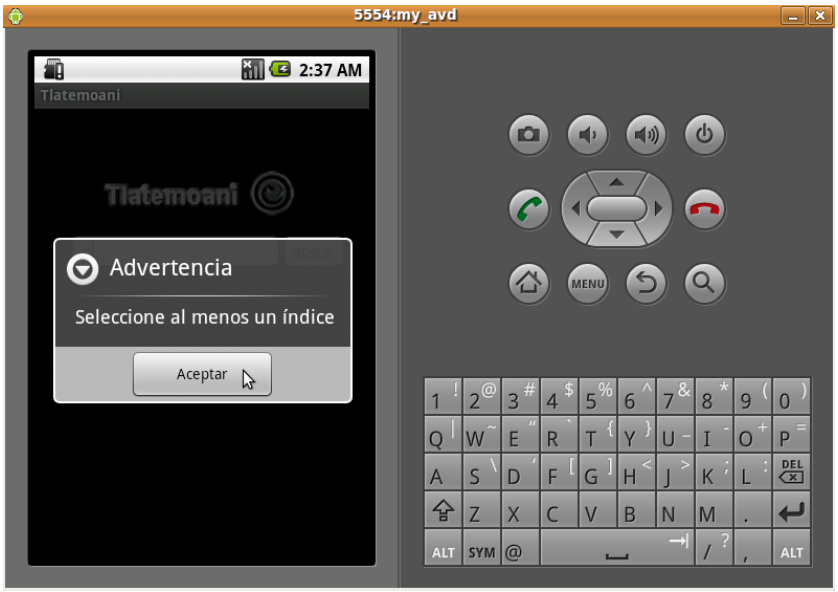

Figura C.15: Caso de Prueba 4-8

V. Caso de prueba **Número Coincidencias**:

- *El usuario selecciona un índice con resultados*:

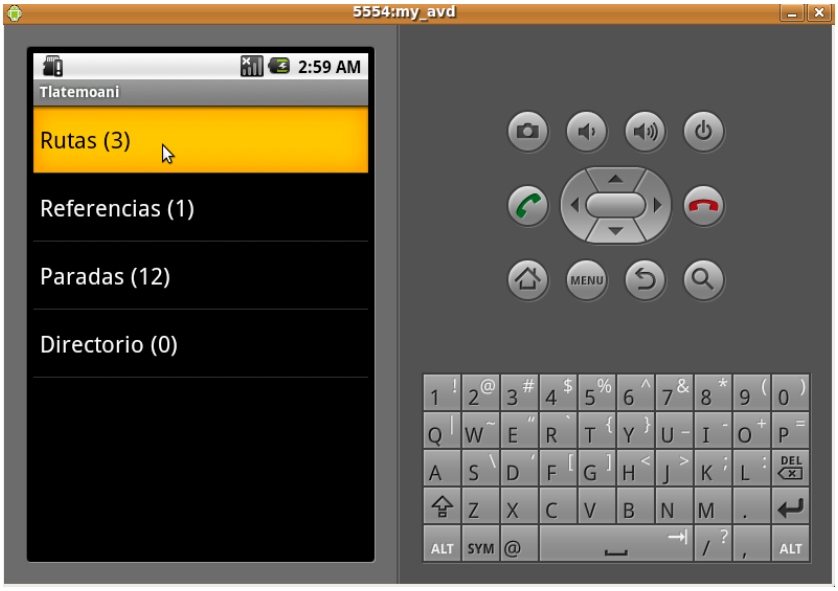

Figura C.16: Caso de Prueba 5-1

- *El sistema muestra la lista de resultados en detalle*:

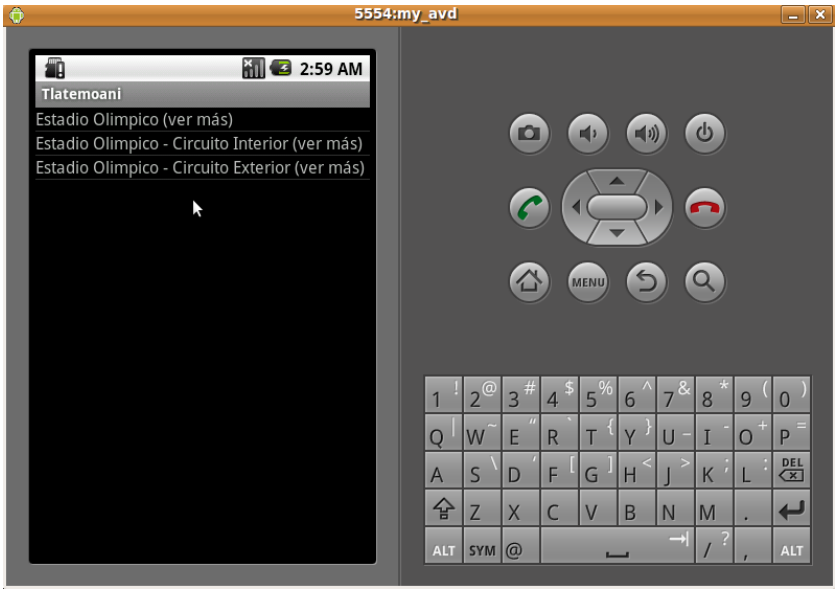

Figura C.17: Caso de Prueba 5-2

- *Alternativa – El usuario elige un índice sin resultados*:

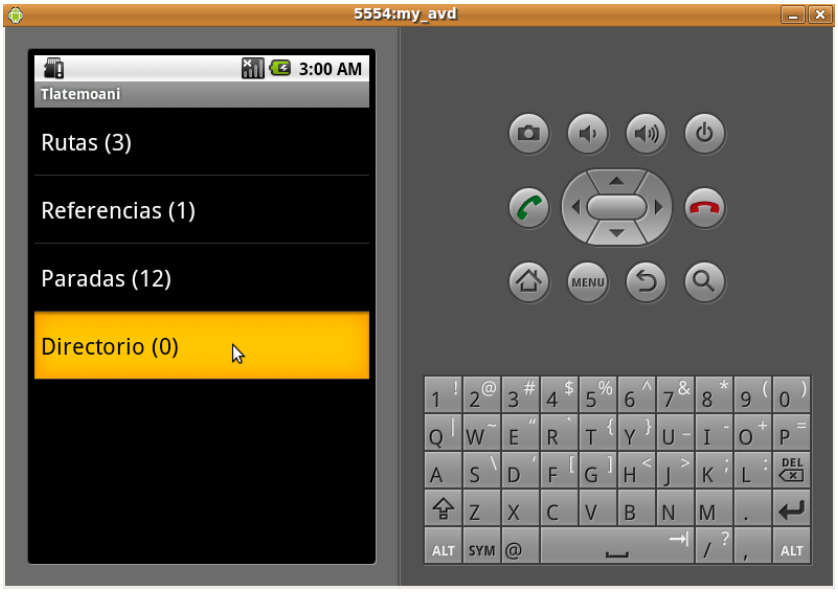

Figura C.18: Caso de Prueba 5-3

- *Alternativa – Salir con Home*: Ver Figura C.3
- *Error – Sin conexión*:

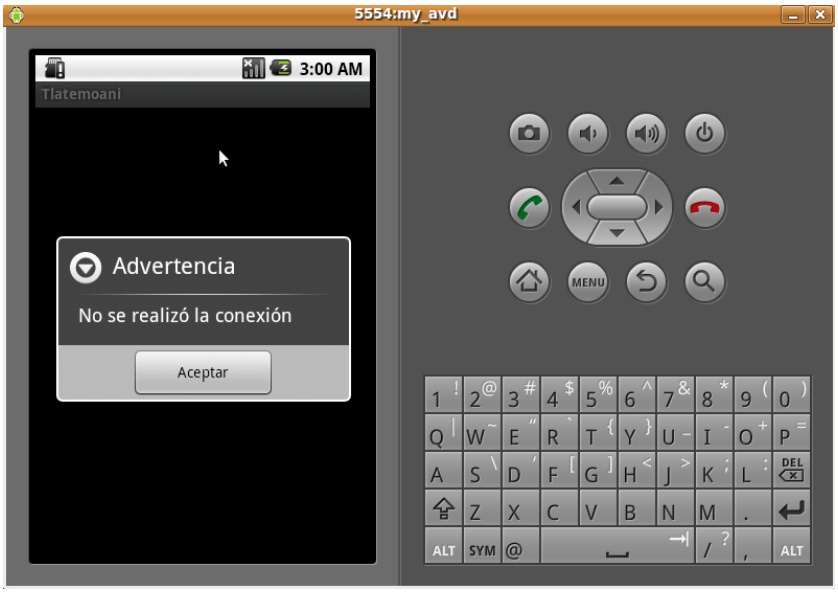

Figura C.19: Caso de Prueba 5-4

VI. Caso de prueba **Detalles**:

- *El usuario selecciona un elemento de la lista*:

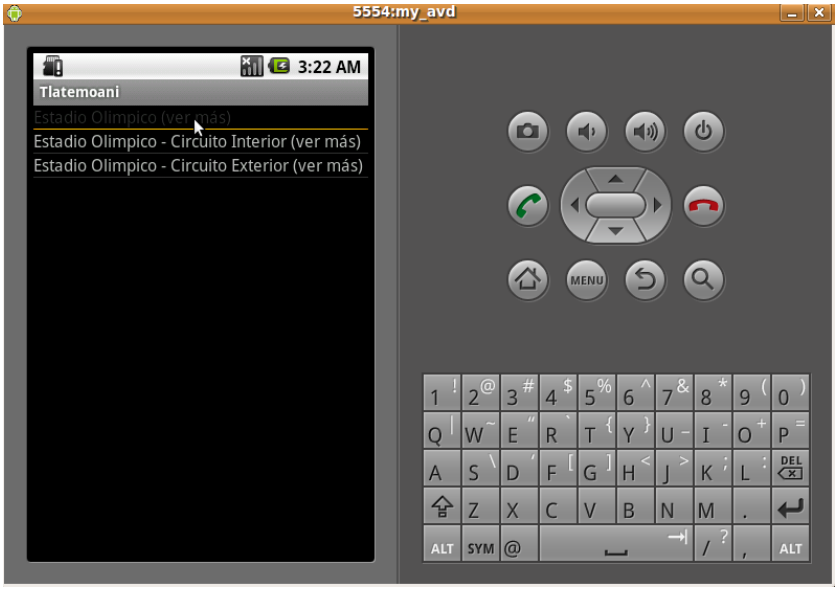

Figura C.20: Caso de Prueba 6-1

- *El sistema muestra el contenido del elemento*:

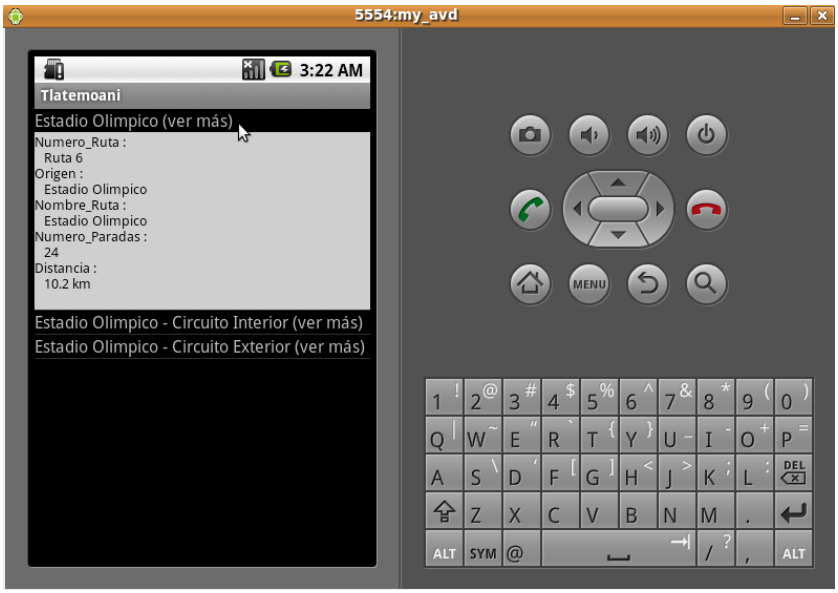

Figura C.21: Caso de Prueba 6-2

- *Alternativa – El sistema oculta el contenido del elemento*:

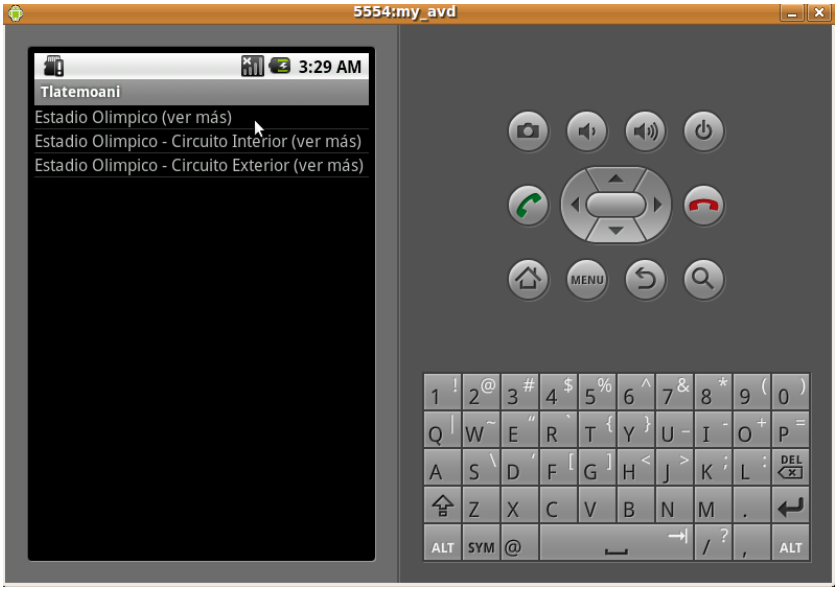

Figura C.22: Caso de Prueba 6-3

- *Alternativa – Salir con Home*: Ver Figura C.3
- *Alternativa – El usuario presiona el botón para regresar*: Ver Figura C.8
- *Error – Sin conexión*: Ver Figura C.5

Bibliografía

[1] *Professional Android Application Development*, Autor: Reto Meier, Editorial Wiley Publishing, Inc., año 2009.

[2] *Android Essentials*, Autor: Chris Haseman, Editorial Apress, SPRINGER, año 2008.

[3] *Beginning Android*, Autor: Mark L. Murphy, Editorial Apress, SPRINGER, año 2009.

[4] *Pro Android*, Autor: Sayed Y. Hashimi,Satya Komatineni, Editorial Apress, SPRINGER, año 2009.

[5] *Unlocking Android*,Autor: W. Frank Ableson, Charlie Collins, Robi Sen, Editorial Manning Publications Co, año 2009.

[6] *Android A Programmer's Guide*, Autor: J.F DiMarzio, Editorial McGraw Hill Companies, año 2008.

[7] *Lucene in Action*, Autor: Otis Gospodneti'c, Erick Hatcher, Doug Cutting, Editorial Manning Publications Co, año 2005.

[8] *Solr 1.4 Enterprise Search Server*, Autor: David Smiley, Eric Pugh, Editorial Packt Publishing, año 2009.

[9] *Operating Systems Design and Implementation, 3/E* , Autor: Andrew S. Tanenbaum, Editorial Prentice Hall, año 2006.

[10] *Smarthphone Operating System Concepts with Symbian OS*, Autor: Michael J. Jipping, Editorial Wiley Publishing, Inc., año 2007.

[11] *Desarrollos Móviles con .NET*, Autor: Maximiliano Firtman, Editorial MP Ediciones, Manuales USERS, año 2005.

[12] *J2ME: The Complete Reference*, Autor: James Keogh, Editorial McGraw-Hill Companies, año 2003.

[13] *Mobile Database Systems*, Autor: Vijay Kumar, Editorial Wiley-Interscience, año 2006.

[14] *AMIPCI*, Autor: AMIPCI, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: `http://www.amipci.org.mx/`.

[15] *Framework*, Autor: DocForge, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: `http://docforge.com/wiki/Framework`.

[16] *Historia Android*, Autor: Pocketnow.com, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: `http://pocketnow.com/thought/android-10-to-21-what-has-changed`.

[17] *Designing for Performance*, Autor: Android.com, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: `http://developer.android.com/guide/practices/design/performance.html`.

[18] *Token*, Autor: Lucene.org, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: `http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/analysis/Token.html`.

[19] *Information Retrieval*, Autor: Wiki Community, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: http://en.wikipedia.org/wiki/Information_retrieval.

[20] *As We May Think*, Autor: Vannevar Bush, Fecha de Consulta: 21-30 Octubre de 2009, Disponible en: http://www.theatlantic.com/magazine/archive/1969/12/as-we-may-think/3881/.

[21] *Android*, Autor: Wiki Community, Fecha de Consulta: 23 Octubre de 2009, Disponible en: http://es.wikipedia.org/wiki/Android.

[22] *Dalvik Virtual Machine*, Autor: Wiki Community, Fecha de Consulta: 23 Octubre de 2009, Disponible en: http://www.dalvikvm.com/.

[23] *Procesadores ARM*, Autor: Wiki Community, Fecha de Consulta: 26 Octubre de 2009, Disponible en: http://es.wikipedia.org/wiki/ARM.

[24] *Reference Section*, Autor: Android Community, Fecha de Consulta: 27 Octubre de 2009, Disponible en: http://developer.android.com/reference/packages.html.

[25] *Introduction to The Solr Enterprise Search Server*, Autor: Solr Project, Fecha de Consulta: 27 Octubre de 2009, Disponible en: http://wiki.apache.org/solr/.

[26] *Lucene Site*, Autor: Lucene Project, Fecha de Consulta: 27 Octubre de 2009, Disponible en: http://lucene.apache.org/.

[27] *Lucene API*, Autor: Lucene Project, Fecha de Consulta: 27 Octubre de 2009, Disponible en: http://lucene.apache.org/java/2_9_0/api/all/index.html.

[28] *Página Google*, Autor: Google Inc, Fecha de Consulta: 11 Noviembre de 2009, Disponible en: http://www.google.com/corporate/features.html.

[29] *Página Yahoo*, Autor: Yahoo Inc, Fecha de Consulta: 11 Noviembre de 2009, Disponible en: http://search.yahoo.com/info/.

[30] *Bing motor de Búsqueda*, Autor: Wiki Community, Fecha de Consulta: 13 Noviembre de 2009, Disponible en: http://es.wikipedia.org/wiki/Bing_(motor_de_b%C3%BAsqueda).

[31] *Mobile Search*, Autor: Wiki Community, Fecha de Consulta: 15 Noviembre de 2009, Disponible en: http://en.wikipedia.org/wiki/Mobile_search.

[32] *Mobile Content*, Autor: Wiki Community, Fecha de Consulta: 15 Noviembre de 2009, Disponible en: http://en.wikipedia.org/wiki/Mobile_content.

[33] *Servicios de Google para móviles*, Autor: Google Inc, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://google.dirson.com/o.a/google-movil.

[34] *Yahoo para móviles*, Autor: Yahoo Inc, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://es.mobile.yahoo.com/search/application.

[35] *Microsoft Bing for Mobile (Windows Mobile)*, Autor: Sean Ludwig - PCMag.com, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://www.pcmag.com/article2/0,2817,2352238,00.asp.

[36] *Discover Bing*, Autor: Microsoft, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://www.pcmag.com/article2/0,2817,2352238,00.asp.

[37] *Databases and MIDP, Part 1: Understanding the Record Management System*, Autor: Eric Giguere, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://developers.sun.com/mobility/midp/articles/databaserms/.

[38] *Acceso a base de datos de SQL Server CE mediante dispositivos móviles*, Autor: Sergio José Villaneda Ávila, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://www.netveloper.com/contenido2.aspx?IDC=300_0.

[39] *SQL Server Compact 2005*, Autor: Microsoft, Fecha de Consulta: 29 Noviembre de 2009, Disponible en: http://msdn.microsoft.com/es-es/library/ms173053(SQL.90).aspx.

[40] *Mobile Database*, Autor: Wiki Community, Fecha de Consulta: 4 Diciembre de 2009, Disponible en: `http://en.wikipedia.org/wiki/Mobile_database`.

[41] *SQL Anywhere for mobile applications*, Autor: Sybase Inc, Fecha de Consulta: 12 Diciembre de 2009, Disponible en: `http://www.sybase.com/products/databasemanagement/sqlanywhere/mobileapplications`.

[42] *DB2 Everyplace*, Autor: Grupo GSInnova, Fecha de Consulta: 12 Diciembre de 2009, Disponible en: `http://www.rational.com.ar/db2spanish/db2everyplace.html`.

[43] *DB2 Everyplace*, Autor: Grupo GSInnova, Fecha de Consulta: 12 Diciembre de 2009, Disponible en: `http://www-142.ibm.com/software/dre/hmc/compare.wss?HMC02=Y727597T60152D73`.

[44] *HanDBase*, Autor: DDH Software, Fecha de Consulta: 21 Diciembre de 2009, Disponible en: `http://www.ddhsoftware.com/handbase_specs.html`.

[45] *SQLite*, Autor: Wiki Community, Fecha de Consulta: 21 Diciembre de 2009, Disponible en: `http://en.wikipedia.org/wiki/SQLite`.

[46] *Features Of SQLite*, Autor: SQLite Org, Fecha de Consulta: 21 Diciembre de 2009, Disponible en: `http://www.sqlite.org/features.html`.

[47] *Db4o*, Autor: Wiki Community, Fecha de Consulta: 11 Enero de 2010, Disponible en: `http://en.wikipedia.org/wiki/Db4o`.

[48] *Db4o Specifications*, Autor: Versant Corp, Fecha de Consulta: 11 Enero de 2010, Disponible en: `http://www.db4o.com/about/productinformation/datasheet/`.

[49] *Getting Started with an Integrated Development Environment*, Autor: Dana Nourie, Sun Developer Network, Marzo 24, 2005.

[50] *The Practical Design Method A Software Design Method for a First Object-Oriented Project*, Autor: Jorge L. Ortega Arjona, Departamento de Matemáticas, Facultad de Ciencias, UNAM, año 2005.

[51] *Probabilistic models in information retrieval*, Autor: Norbert Fuhr, Editorial Oxford University Press, The Computer Journal vol 35, año 1992.

[52] *Whats new with Apache Solr*, Autor: Grant Ingersoll, IBM DeveloperWorks, año 2008.

[53] *Recuperación de la información en Internet: motores y otros agentes de búsqueda*, Autor: José Raúl Vaquero Pulido, Scire: representación y organización del conocimiento Vol.3, N.2 pag 85-100, año 1997.

[54] *SQL Anywhere Product Datasheet*, Autor: iAnywhere Solutions Inc, año 2006-2009.

[55] *Choosing Between SQL Server CE and SQL Server Express*, Autor: Steve Lasker, Microsoft, año 2006.

[56] *Oracle Database Lite 10g*, Autor: Oracle Corporation, año 2008.

[57] *Designing Wireless Clients for Enterprise Applications with Java Technology*, Autor: Sun Microsystems, año 2003.

[58] *A Vector Space Model for Automatic Indexing*, Autor: G. Salton, A. Wong, and C. S. Yang, Communications of the ACM, vol. 18, nr. 11, año 1975.

[59] *Lucene in Action*, Autor: Erik Hatcher, Conferencia: Java One Sun's Worldwide Java Developer Conference, Session TS-2994, año 2004.