Universidad Carlos III de Madrid

Escuela politécnica Superior

Ingeniería Técnica en Informática de Gestión

PROYECTO FIN DE CARRERA

Aplicación Android para la gestión de Directorios Activos “Ldapp”

Alumno: Jorge Ángel Martín-Maestro Hermida.

Directora: Susana Montero Moreno.

Tutora: Ana Isabel González-Tablas Ferreres.

Octubre de 2012

“El esfuerzo solo proporciona plenamente su recompensa, después de que una persona se niega a darse por vencida.”

Napoleón Hill

Titulo: Apl Android para la gestión de Directorios Activos “Ldapp”
Autor: Jorge Ángel Martín-Maestro Hermida.
Director: Susana Montero Moreno.
Tutor: Ana Isabel González-Tablas Ferreres.

TRIBUNAL

Presidente: Benjamín Ramos Álvarez.

Vocal: Javier Fernández Muñoz.

Secretario: Jorge Blasco Alís

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 5 de Octubre de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid.

Vocal

Secretario **Presidente**

Agradecimientos

La consecución de este proyecto fin de carrera representa para mi, no solo la finalización de mi etapa universitaria, sino también, la afirmación de que con esfuerzo y trabajo, cualquier cosa es posible, y es por ello que me siento especialmente orgulloso. También ahora, soy consciente, de la importancia de todo el apoyo prestado por algunas personas, que en momentos de debilidad estuvieron a mi lado y que nunca permitieron que bajase los brazos animándome a continuar cuando yo pensaba que mi camino había terminado. Es por ello que quiero hacer especial mención a mi familia, en especial a mi madre, por su confianza en mi, y por estar siempre cuando más la necesitaba, y a Mariam, por su paciencia y comprensión, cuando a mi ya no me quedaba ninguna de ellas.

También quiero agradecer la ayuda prestada a mi tutora del proyecto, Susana Montero Moreno, por su inestimable ayuda y por haber continuado siendo mi tutora aún cuando ya había abandonado la universidad, así como a todo el profesorado y a la propia universidad Carlos III de Madrid, por haberme dado una formación de primera categoría.

A todos ellos, gracias.

Resumen

En la actualidad, vivimos en un mundo cada vez más globalizado, y consecuentemente, más intercomunicado, en el que el mundo de las telecomunicaciones nos abre un abanico de posibilidades prácticamente infinito en cuanto a comunicación y acceso a la información se refiere, y el presente proyecto busca integrar en una misma aplicación ambas cualidades.

Así pues, este proyecto consiste en la ejecución de una aplicación que permitirá al usuario de un dispositivo móvil con plataforma Android, guardar distintas configuraciones de acceso a servicios de directorio, para la realización de búsquedas sobre los mismos, de los que se obtendrán todas las entradas coincidentes con el criterio de búsqueda seleccionado para dicho filtrado (por nombre, apellidos, teléfono, email, …). Sobre este conjunto de resultados obtenidos en la búsqueda sobre el directorio, el usuario también podrá seleccionar una determinada entrada, para visualizar de una manera más amplia su información. Además de poder acceder a toda esta información, el usuario dispondrá de otras funcionalidades proporcionadas por la propia plataforma Android, como pueden ser, el salvado de dicha entrada como un registro de la agenda del dispositivo, el marcado del teléfono de dicha entrada, el envío de un email a su dirección de correo electrónico, el posicionamiento de su dirección mediante *Google Maps*, o la búsqueda por nombres y apellidos de la entrada sobre la red social *LinkedIn*, que nos permitirá ampliar su información.

Índice general

Índice de figuras

Índice de tablas

Índice de código fuente

1 Introducción

Esta introducción esta enfocada a acercar al lector, a las motivaciones, objetivos y contenidos que conforman las bases de este proyecto. A continuación se detallan cada una de los mencionados temas introductorios.

1.1 Visión General

En mundo de la telefonía móvil ha evolucionado enormemente en los últimos tiempos, de tal manera que hemos pasado en tan solo un par de décadas, de manejar dispositivos móviles con limitadas funcionalidades y poco prácticos en cuanto a usabilidad se refiere a maquinas con infinidad de recursos y de reducido tamaño, que abren un gran abanico de posibilidades de uso. Actualmente tanto la capacidad y rendimiento como las diferentes plataformas que existen para dispositivos móviles, posibilitan la creación de un ilimitado número de aplicaciones que cubren cualquier tipo de necesidad que pueda tener el usuario de un dispositivo móvil.

Los servicios de directorio son utilizados en la actualidad por infinidad de organizaciones para almacenar y organizar toda la información referente a los recursos de los que dispone la misma, y en consecuencia pueden llegar a ser una fuente de información muy importante para su actividad comercial durante el día a día, y por esta misma razón y en consonancia con el gran crecimiento que se ha dado en el mundo de los dispositivos móviles y las plataformas que implementan durante los últimos años, y a que se ha posibilitado el acceso a Internet prácticamente desde cualquiera de ellos, ha surgido la necesidad de poder tener acceso a estos servicios de directorio de manera remota. En ese contexto, se han desarrollado algunos aplicativos que permiten acceder a los servicios de directorio, algunos de ellos se describen brevemente en el *apartado 2.3.* La mayoría de estas aplicaciones, permiten al usuario, configurar el acceso a un determinado servicio de directorio, y realizar algunas operaciones de consulta sobre el mismo, mediante el filtrado por un determinado criterio de búsqueda. También integran estás operaciones de consulta con algunas de las funcionalidades propias de la plataforma que monta el dispositivo móvil, como por ejemplo, el almacenamiento de los resultados de búsqueda sobre el servidor de directorio en la agenda del dispositivo, la sincronización de los contactos del directorio con los del dispositivo, el marcado del campo teléfono o el envío de e-mails al campo e-mail perteneciente a una de las entradas obtenidas como resultado de una búsqueda sobre el directorio, etc…

1.2 Objetivos

En lo que a mi aprendizaje se refiere, los principales objetivos perseguidos han sido los siguientes:

- Obtención de conocimientos sobre la plataforma *Android*.
  Este fue mi primer objetivo, la obtención de información referente a la plataforma *Android* y la comprensión de la misma, así como el conocimiento de las diferentes formas de desarrollar una aplicación para dicha plataforma.
- Obtención de conocimientos sobre protocolo LDAP (Protocolo de Acceso a Directorios Activos).
  Puesto que la funcionalidad principal de la aplicación consiste en el acceso desde cualquier dispositivo móvil con plataforma *Android* a distintos "*directorios activos*" configurados por el usuario. Una de las tareas iniciales fue la de recabar información sobre los mismo, y sobre el protocolo utilizado para el acceso a ellos, denominado "*LDAP*". Toda esta información obtenida me ayudaría a posteriori a comprender en que consiste un "*servidor LDAP*" cual es su estructura, y la manera de acceder a ellos. No obstante, más adelante se expondrá un breve resumen que sintetiza e intentará explicar el funcionamiento de los mismos al lector.
- Obtención de conocimientos sobre diferentes Servicios de Internet.
  Otro de los requisitos que tenia que cumplir la aplicación, era la integración de la misma, con diferentes servicios de internet. Así pues en los primeros pasos del proyecto se decidió la inclusión en la aplicación de servicios tales como el de posicionamiento mediante "*Google Maps*" para las direcciones de las personas obtenidas de las búsquedas dentro del directorio activo y servicios relacionados con alguna red social, en este caso "*LinkedIn*", ya que por su orientación profesional era la más indicada para su integración con la funcionalidad de búsqueda en directorios activos. Tras decidir la inclusión en la aplicación de los servicios previamente mencionados, el siguiente objetivo era la documentación sobre las diferentes librerías disponibles para el acceso a los mismos, la arquitectura que seguían y los sistemas de seguridad que implementaban cada uno de ellos, para el posterior desarrollo de las funcionalidades que utilizaría estos servicios.

No obstante, el objetivo primordial en la ejecución de este proyecto ha sido el de la integración de la gestión para el acceso remoto a diferentes directorios activos, con algunos de los servicios que proporciona tanto la plataforma Android como pueden ser añadir un nuevo contacto a la agenda, enviar un email, marcar un determinado teléfono, como los servicios que proporcionan las redes sociales, en este caso la red profesional "*LinkedIn*" y los servicios de posicionamiento como "*Google Maps"*. De esta manera se pretende dotar al usuario de una forma de interconexión entre distintas fuentes de información, permitiéndole así ampliarla de una manera más integral.

Todas estas funcionalidades podrían ser de especial utilidad para aquellas organizaciones que quisieran dotar a sus empleados de una fuente de información distribuida, como por ejemplo de su cartera de clientes, de la información referente a los demás

empleados de la empresa, etc…, y ampliar dicha información con la funcionalidad de búsqueda en "*LinkedIn*". Podrían acceder a ella desde cualquier dispositivo móvil con plataforma Android en la que se encontrase instalada la aplicación "*Ldapp*".

1.3 Contenido de la memoria

La memoria de este proyecto está estructurada en 10 capítulos, que a su vez se subdividirán en otros apartados. A continuación se describe el contenido de cada uno de estos capítulos.

El *Capítulo 1*, "*Introducción*", contiene una visión general dentro del contexto en el que actualmente nos encontramos, así como un apartado enfocado a exponer al lector cada uno de los objetivos que se pretende alcanzar en este proyecto fin de carrera y por ultimo, el presente apartado, donde se expondrá una breve descripción del contenido de cada uno de los capítulos que componen este proyecto.

El *Capítulo 2, "Estado del Arte",* contiene el estado actual del arte. En este capítulo se expone al lector en que punto se encuentran las tecnologías empleadas durante la elaboración del presente proyecto hoy en día.

El *Capítulo 3*, "*Plataforma Android*", está enfocado a dar una idea al lector sobre los fundamentos de la plataforma *Android*, en que consiste y la estructura y arquitectura empleada por la misma.

El *capítulo 4*, "*Protocolo Ldap*", contiene una breve descripción sobre el funcionamiento de los protocolos de acceso a servicios de directorio, con el fin, de ilustrar al lector sobre una de las principales funcionalidades que implementará la aplicación "*Ldapp*".

El *capítulo 5*, "*Aplicación Ldap*", contiene toda la información referente al análisis, diseño e implementación de la aplicación "*Ldapp*", así como información referente a la arquitectura empleada en la aplicación.

El *capítulo 6*, "*Gestión del proyecto*", contiene toda la información relacionada con la gestión del proyecto, como la metodología utilizada, los recursos utilizados, o los tiempos empleados en la ejecución del proyecto.

El *capítulo 7*, "*Conclusión y Trabajos Futuros*", contiene una breve conclusión obtenida como consecuencia de la finalización del proyecto, así como los posibles trabajos que se podrían realizar sobre este proyecto fin de carrera como extensión al mismo.

El *capítulo 8*, "*Referencias*", contiene todas las referencias documentales, que han sido utilizadas durante la documentación del proyecto.

El *capítulo 9*, "*Glosario*", contiene una breve descripción para las palabras de difícil comprensión para el lector contenidas en el proyecto.

El *capítulo 10, "Aplicaciones Utilizadas",* contiene una lista con todas las herramientas utilizadas para la realización del presente proyecto.

2 Estado del Arte

En este capítulo se dará al lector una visión ampliada del estado actual de las tecnologías utilizadas para la realización de este proyecto.

2.1 Dispositivos móviles

Por dispositivo móvil se entiende cualquier computadora de tamaño reducido con capacidad de procesamiento, conexión a internet permanente o intermitente y un tamaño de memoria limitado. Algunos de los dispositivos móviles más comunes son:

- Smartphone (teléfonos inteligentes de última generación).
- PDA´s.
- Celulares.
- Videoconsola portátil.
- Reproductor de audio portátil.
- Cámara digital.
- Cámara de video.
- Laptop (portatil).

Los controles de dichos dispositivos pueden ser o bien pequeños botones o como ya sucede en la actualidad, pantallas táctiles o bien híbridos de ambos tipos, táctiles y con botones.

Figura 1. Ejemplos de dispositivos móviles.

De entre todos ellos, los que más relevancia tienen en la actualidad, son los denominados "*smartphones*" o teléfonos inteligentes. Estos dispositivos son teléfonos móviles que incorporan características propias de un ordenador personal. Permiten la instalación de aplicaciones, tienen conexión a internet, y otras muchas funcionalidades, todo ello gracias a que normalmente incorporan un potente sistema operativo como por ejemplo "*Android", "IOS", "Symbian OS", "BlackBerry OS",* etc…, Algunos de ellos será estudiados más extensamente en apartados sucesivos.

2.2 Sistemas Operativos para Dispositivos Móviles

Actualmente hay un gran abanico de sistemas operativos diseñados específicamente para funcionar en dispositivos móviles. Estos sistemas operativos han sido concebidos para obtener el mayor rendimiento a partir de una limitación de

recursos, como suele ocurrir en la mayoría de dispositivos móviles. Ya que estos suelen contar con funcionalidades propias de ordenadores personales pero no asi con la capacidad que ellos tienen, en cuanto a memoria y capacidad de procesamiento se refiere.

Los más conocidos del mercado son los siguientes:

- Android
- Symbian OS
- iOS
- BlackBerry OS
- Windows Phone
- Otros
- Bada
- MeeGo

A continuación se muestra en la Figura 2, un gráfico con el porcentaje de implantación de los diferentes sistemas operativos durante 2011. Como se puede observar, en la actualidad los sistemas operativos que acaparan el mercado son "*Android*" e "*IOS*", entre ambos aglutinan más del 74 % de la cuota de mercado.

En apartados sucesivos se expondrán más a fondo tanto el sistema operativo "*IOS*" propiedad de "*Apple inc*" como el sistema operativo "*Android*" de "*Google inc*".

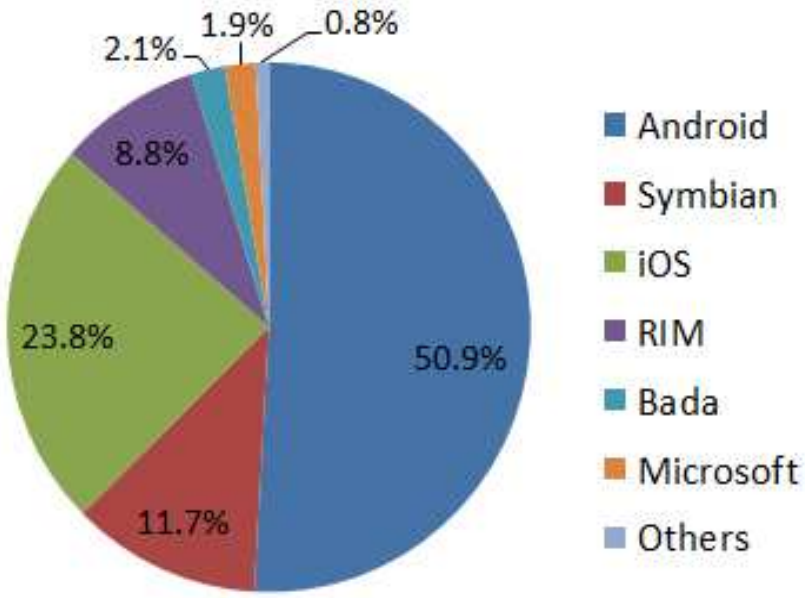
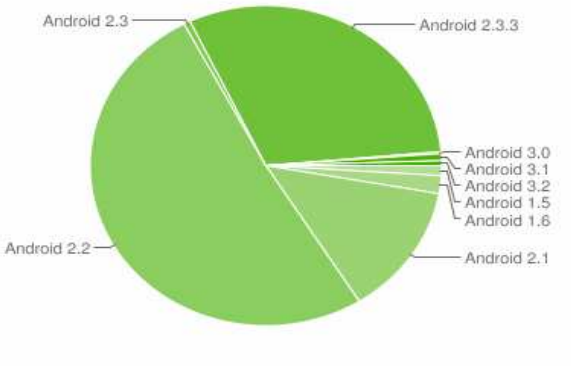


Figura 2. Cuota de mercado de smartphones por Sistema Operativo en 2011. (fuente: Gartner Research (Febrero 2011))

2.2.1 Sistema operativo Android

El sistema operativo "*Android*" propiedad de "*Google inc*", es en la actualidad el sistema operativo para smartphones que más cuota de mercado acapara, aunque también existen versiones del mismo para "*tablets*", estamos hablando de las versiones 3.X denominada "*HoneyComb*".

En la actualidad la versión de Android predominante en el mercado de los smartphones, es decir, la más implantada por las distintas compañías de telefonía móvil en sus dispositivos, es la versión 2.2 ó "*Froyo"*, según los datos estadísticos basados en los terminales que se conectan a "*Android Market*" (sitio web para descarga de aplicaciones para dispositivos móviles Android) publicados por Google, con un total del 51%, aunque ya le sigue muy de cerca la versión 2.3.X ó "*Gingerbread*" con un 31,4%. A continuación en la Figura 3 se muestra un gráfico con los datos publicados por Google, con respecto a la implantación de las distintas versiones de Android en dispositivos móviles.



1.5	Cupcake	1.0%
1.6	Donut	1.8%
2.1	Eclair	13.3%
2.2	Froyo	51.2%
2.3 2.3.2	Gingerbread	0.6%
2.3.3 2.3.4		30.7%
3.0	Honeycomb	0.2%
3.1		0.7%
3.2		0.5%

Figura 3. Implantación de las distintas versiones de Android (2011).
(basado en los terminales que se conectan a "Android Market")

La implantación de este sistema operativo en dispositivos móviles ha sido exponencial desde su irrupción en el mercado de la telefonía móvil allá por 2008 [17], su popularidad se debe en parte a la pertenencia de Google a la "*OHA*" ("*Open Handset Alliance*"), alianza comercial entre 84 compañías pertenecientes al mundo de las telecomunicaciones, entre las que se encuentran algunos de los mayores fabricantes de dispositivos móviles a nivel mundial, como pueden ser *HTC, Samsung, LG, Motorola, Sony Ericsson, etc…,* que han incluido Android en prácticamente todos los nuevos modelos que han ido sacando al mercado. Esto es debido en parte a una de las principales características de Android, es de código abierto, y cada una de las empresas mencionadas anteriormente, ha podido modelar el sistema operativo conforme a sus necesidades.

Figura 4. Miembros pertenecientes a la "Open HandSet Alliance".

2.2.2 Sistema operativo IOS

El sistema operativo "*IOS*" propiedad de la compañia "*Apple inc*" es con el sistema operativo "*Android*" uno de los más implantados en dispositivos móviles en la actualidad, con el 23,8% durante 2011 como se muestra en la Figura 2.

En un principio nació, como un sistema operativo dirigido exclusivamente para su implantación en "*iphone*" (smartphone), pero posteriormente fue remodelado para su utilización en otros dispositivos móviles de la casa, como "*iPod Touch*" (reproductor de audio portatil), "*iPad*" (tablet) ó "*Apple TV*". A diferencia de Google con su sistema operativo "*Android"*, Apple, no permite la instalación de "*iOs*" en dispositivos móviles de terceros.

Este sistema operativo esta basado en el sistema operativo "*Mac OS X*", que a su vez esta basado en "*Darwin BSD*" con lo cual, su kernel es "*Unix*". Actualmente siguen desarrollándose nuevas versiones del mismo, pero la versión estable más reciente es la 5.X.

En cuanto al desarrollo de aplicaciones para este sistema operativo, *Apple* pone a disposición de los desarrolladores su *SDK,* aunque esta solo funcionará en equipos *MAC,* de *Apple,* y a diferencia del sistema operativo Android, el programa de desarrolladores de aplicaciones de *iOs*, el "*iOs Developer Program*" requiere del pago de unas tasas para poder realizar pruebas en un dispositivo móvil de *Apple,* así como para la distribución de la aplicación. Estas tasas son de 99$ al año.

Todas estas trabas en cuanto al desarrollo de una aplicación para *iOs* han sido las que inclinaron la balanza a favor del desarrollo de una aplicación para *Android.*

2.3 Aplicaciones para la gestión de servicios Ldap

En la actualidad, ya hay disponibles otras aplicaciones que tienen como principal funcionalidad la gestión de servicios Ldap para plataformas Android, aunque su funcionalidad se limita exclusivamente a la gestión de dicho servicio. A continuación se describen algunas de las disponibles gratuitamente desde el *Market de Android*,

- "*Ldap Client*":
  Aplicación perteneciente a la compañía *UnboundId* [2], que permite la adición de nuevas configuraciones de servidores, así como la búsqueda sobre ellos, en función de distintos criterios de búsqueda. También permite el envío de emails a las entradas devueltas como resultado de una búsqueda. A diferencia de la aplicación desarrollada en el presente proyecto, esta, no permite la edición de los registros de configuración de servidores, tampoco permite resultados múltiples en las búsquedas sobre un determinado servidor (la coincidencia debe ser exacta al filtro de búsqueda). En definitiva, esta aplicación es una versión reducida de la aplicación *Ldapp.*

- "*Contacts inline*":
  Aplicación perteneciente a "*Philippe Prados*" y que contiene un complemento llamado "*LDAP inline*", que permite la sincronización de los contactos del dispositivo móvil y la búsqueda de contactos en diferentes servidores Ldap.

- "*LDAP To You*":
  Aplicación perteneciente a la empresa "*E-PLANIT SYSTEMS*" que permite la búsqueda en diferentes servicios de directorio. Permite visualizar de manera detallada una determinada entrada del directorio, obtenida de una búsqueda sobre el mismo. A diferencia de los dos anteriores, esta aplicación no es gratuita.

- "*LDAP sync*":
  Aplicación desarrollada por "*Daniel Weisser*" y que permite al usuario, sincronizar su agenda de contactos con las entradas de un determinado servicio de directorio. No existe la posibilidad de realizar búsquedas sobre el propio servicio de directorio.

Las principales diferencias entre la aplicación que se expone en este proyecto fin de carrera, con respecto a las anteriores, son principalmente, que la aplicación Ldapp además de implementar las funcionalidades propias de un gestor de servicios Ldap, se integra con la red social LinkedIn y amplia algunas de las funcionalidades propias de la gestión de servidores Ldap antes mencionadas.

3 Plataforma Android

Este capítulo esta enfocado a explicar al lector cada una de las características de la plataforma Android de manera pormenorizada, desde su arquitectura, hasta la metodología que sigue un desarrollo para dicha plataforma. También se detallará la manera en que se ha llevado acabo el desarrollo de este proyecto.

3.1 Arquitectura del sistema Android

La arquitectura "*Android*" esta basada en el kernel de Linux, es por ello, que tiene un núcleo que aporta a este sistema operativo rapidez, fiabilidad y seguridad probadas.

Internamente, Android utiliza Linux para la gestión de su memoria, de los procesos, para las operaciones de red y para otros servicios del sistema operativo.
No obstante, el usuario nunca interaccionará directamente contra Linux, tampoco lo harán las aplicaciones instaladas en el dispositivo móvil, será pues el propio "*Android*", el que interactúe con el. A continuación, se muestra en la Figura 5, la estructura que sigue un sistema "*Android*".

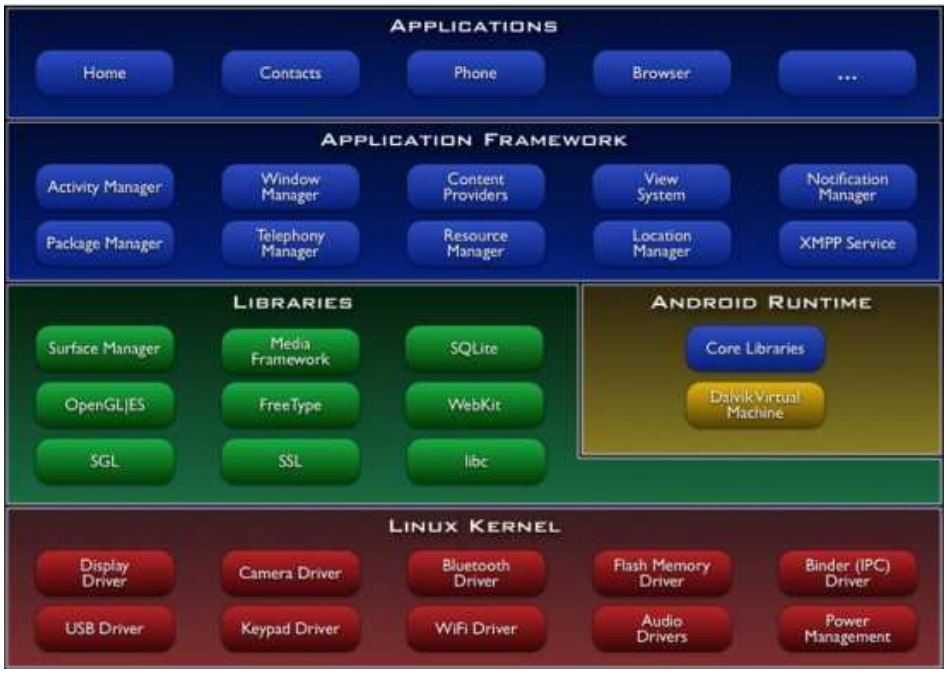


Figura 5. Arquitectura sistema Android.

La arquitectura "*Android*" implementa las capas que se describen a continuación:

- Bibliotecas nativas:

  Es la capa situada por encima del Kernel y contiene las bibliotecas nativas de "*Android*". Estas bibliotecas compartidas están implementadas en C ó C++, y compiladas para la arquitectura de hardware especifica utilizada por el dispositivo móvil y preinstaladas por el fabricante del mismo.

  Algunas de las bibliotecas nativas de Android de mayor trascendencia son las siguientes:

  - **Gestor de superficie**:

    Es el gestor de ventana de composición utilizado por "*Android*". Permitirá al sistema crear todo tipo de efectos como ventanas transparentes.
  - **Gestor de gráficos 2D y 3D**:

    En "*Android"*, es posible combinar en una sola interfaz de usuario elementos de 2 y 3 dimensiones, y esta biblioteca permitirá al sistema tomar la decisión de utilizar hardware 3D si es que el dispositivo móvil dispone de el, o un renderizador de software en caso contrario.
  - **Códecs multimedia**:

    "*Android*" puede reproducir video y grabar o reproducir audio, esta librería contiene los codecs necesarios para reproducir un determinado elemento multimedia en distintos formatos (AAC, AVC, MP3 y MP4).
  - **Base de datos SQL:**

    "*Android*" incluye un motor de bases de datos "*SQLite*" [18], la misma base de datos utilizada por el navegador "*Firefox*" y el "*iPhone*" de "*Apple*". La utilizarán las aplicaciones "*Android*" para almacenamiento persistente.
  - **Tecnología de navegador:**

    Utilizada por el sistema para la muestra rápida de contenido HTML, Android utiliza la biblioteca "*WebKit*" [19] para este cometido.

- Tiempo de ejecución:

  También sobre la capa del Kernel se encuentra la capa de tiempo de ejecución, que está provista de la maquina virtual "*Dalvik*".

  Esta maquina virtual tiene como especial peculiaridad, que el código se compila en ella por medio de instrucciones independientes denominadas "*bytecodes*", que son ejecutadas por la propia maquina virtual en el dispositivo móvil.

  Esta maquina virtual desarrollada por Google, realmente es una maquina virtual Java, optimizada para su utilización en dispositivos con restricciones de memoria.

  Todo el código escrito en Java para un sistema "*Android*" se ejecutará en esta maquina virtual.

- Estructura de aplicaciones

Esta capa esta situada por encima de las capas de "*bibliotecas nativas*" y de la capa de "*tiempo de ejecución*". Esta capa contiene los bloques de construcción de alto nivel en los que nos apoyaremos a la hora de crear aplicaciones para el sistema "*Android*". Los componentes más importantes de la estructura de aplicaciones son los siguientes:

- **Gestor de actividad:**
  Este gestor es el encargado de controlar el ciclo de vida de las aplicaciones y mantiene un orden entre las aplicaciones, para que el usuario pueda moverse de unas a otras.
- **Proveedor de contenido:**
  Estos objetos encapsulan datos que tienen que ser compartidos entre distintas aplicaciones.
- **Gestor de recursos:**
  Encargado de gestionar cualquier tipo de elemento de la aplicación que no sea código.
- **Gestor de ubicación:**
  Se encarga de posicionar el dispositivo móvil en todo momento.
- **Gestor de notificaciones:**
  Es el encargado de la gestión de cualquier tipo de notificación que tenga que mostrarse al usuario, como por ejemplo, citas, alertas, etc…

- Aplicaciones y widgets

  Esta es la capa superior en el diagrama de arquitectura de un sistema "*Android*", y será la que contenga todas las aplicaciones y widgets (aplicaciones que ocuparan una pequeña parte de la pantalla de inicio y toda su funcionalidad se podrá llevar acabo desde la misma) contenidos en nuestro dispositivo móvil, tanto los preinstalados en el como los instalados posteriormente por el usuario.

3.2 Ciclo de vida de un "Activity"

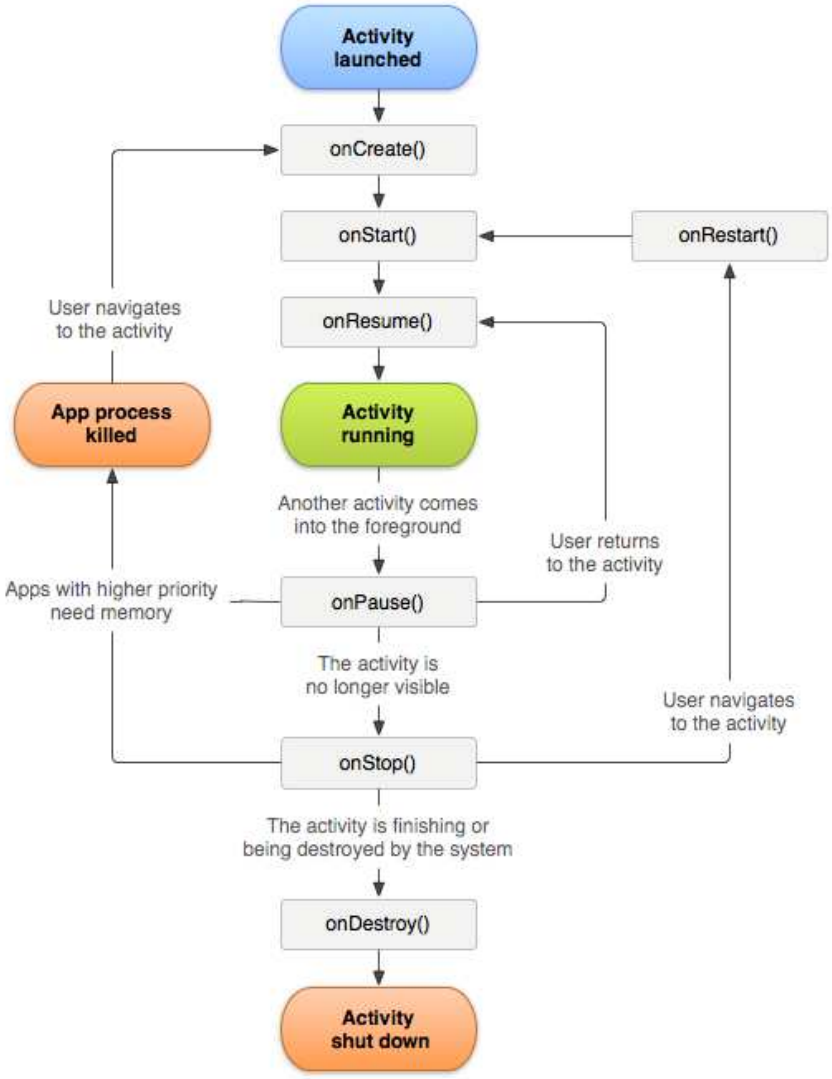


Figura 6. Diagrama Ciclo de vida de una actividad Android.

El ciclo de vida de una aplicación "*Android*" es algo diferente al de una aplicación convencional. Internamente, cada pantalla de la interfaz de usuario esta representada por una clase "*Activity*", y cada una de ellas tiene su propio ciclo de vida dentro de la plataforma "*Android*". A su vez, una aplicación "*Android*" esta compuesta por una o varias actividades y un proceso *Linux* que las contiene. El ciclo de vida del proceso contenedor no esta directamente relacionado con el ciclo de vida de las actividades de una aplicación, que puede estar "viva" aunque el proceso Linux haya dejado de ejecutarse. Por ejemplo en el caso de que la aplicación se quedase en un segundo plano de ejecución. Este mecanismo se utiliza para liberar recursos del sistema, cuando una determinada aplicación no está en uso.

El ciclo de vida de una actividad "*Android*" esta compuesto por diferentes estados de ejecución de la misma. El cambio entre unos u otros estados no está al alcance de los desarrolladores, será una función que solo podrá llevarse acabo por el sistema operativo. No obstante, este se encarga de notificar el cambio de un estado a otro de una determinada actividad mediante la ejecución de los métodos "*onXX*" que ya si que podrán ser implementados en cada actividad por el desarrollador, para tener un cierto control sobre los cambios de estado durante la ejecución de la actividad. A continuación se especifican los métodos de que dispone el desarrollador para manejar dichos cambios de estado:

- onCreate(Bundle):
  Este método es ejecutado cuando se inicia la actividad por primera vez.
- onStart():
  Este método es ejecutado cuando va a mostrarse la actividad al usuario.
- onResume():
  Es ejecutado cuando la actividad puede empezar a interactuar con el usuario.
- onPause():
  Se ejecuta cuando una actividad va a pasar a ser ejecutada en segundo plano.
- onStop():
  Se ejecuta cuando la actividad ya no esta siendo visualizada por el usuario y no va a ser ejecutada durante algún tiempo.
- onRestart():
  Se ejecuta cuando una actividad que estaba en estado "*detenido*", va a volver a mostrarse al usuario.
- onDestroy():
  Se ejecuta antes de que la actividad sea destruida.
- onSaveInstanceState(Bundle):
  Este método es ejecutado por "*Android*" para permitir a las actividades guardar el estado de la instancia, como puede ser la posición de un cursor en un campo de texto por ejemplo. La implementación predeterminada para una actividad ya guardan el estado de todos los controles de la misma por defecto, así que normalmente no será necesario implementar este método.

3.3 Principales componentes de la SDK Android

La "*SDK Android*" consiste en un kit de desarrollo que proporciona "*Android*" para el desarrollo de aplicaciones para dicha plataforma de manera gratuita.

La "*SDK Android*" esta compuesta por una serie de objetos que servirán para realizar una aplicación "*Android*". Los más importantes son los que se mencionan a continuación:

- Actividades ("*Activity*"):
  Ya hemos hablado antes de ellas, pero en definitiva son las entidades que definen la interfaz de una pantalla de usuario. Una aplicación podrá contener varias y se utilizarán para llevar un control de las diferentes fases de la aplicación.

- Intenciones ("*Intent*"):
  Son las entidades que permitirán al desarrollador definir el paso de una actividad.
- Servicios ("*Service*"):
  Son tareas que se ejecutan en segundo plano, con independencia de que el usuario este visualizando nuestra aplicación o no. Como por ejemplo cuando estamos escuchando música, estemos o no visualizando la pantalla de nuestro reproductor, la música debe seguir sonando. En estos casos deberemos utilizar servicios.
- Proveedores de contenido ("*ContentProvider*"):
  Consiste en un conjunto de datos a los que se tiene acceso mediante una API personalizada. Estos objetos se utilizan cuando una aplicación quiere compartir sus datos de manera global, es decir, con otras aplicaciones.

3.4 Estructura de una aplicación Android

Para la implementación de la aplicación "*ldap*" se ha utilizado con IDE de desarrollo "*Eclipse*", y en consecuencia, la estructura que estudiaremos a continuación es la de un proyecto "*Android*" para esta herramienta de desarrollo.

Estructura de carpetas.

El primer paso para la implementación de una aplicación "*Android*" es la creación de un nuevo proyecto, este tendrá la estructura de carpetas similar a la que se muestra en la Figura 7, correspondiente a la estructura de carpetas de un nuevo proyecto Android creado en *Eclipse*.

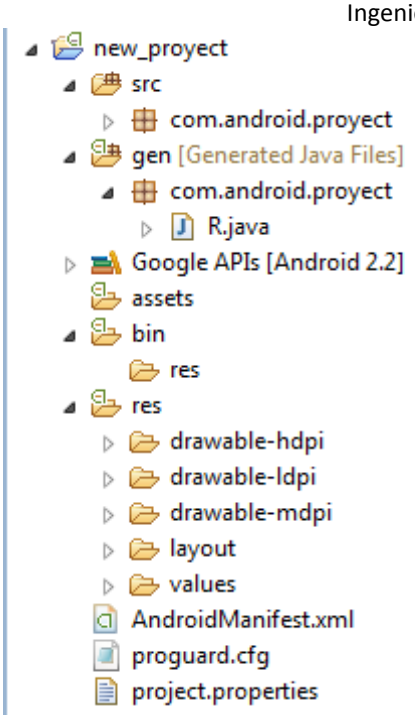


Figura 7. Estructura de carpetas aplicación Android.

A continuación se describe el fin de cada una de las carpetas que contiene un proyecto "*Android*":

- Carpeta "*src*":
  Contiene todos los paquetes que a su vez contendrán las clases Java que implementarán la funcionalidad de la aplicación.
- Carpeta "*gen*":
  Contiene la clase Java "*R.java*", que a su vez contiene identificadores para todos los resursos de la aplicación contenidos en la carpeta "*res*", como pueden ser imágenes, cadenas de texto, estilos ("*themes*"), etc... Las referencias a estos recursos se añadirán de manera automática por el complemento de "*Eclipse*" para "*Android*" según se vayan introduciendo en la carpeta "*res*". Es recomendable no modificar esta clase Java de manera manual.
- Carpeta "*bin*":
  Contiene los archivos de la aplicación compilada.
- Carpera "*libs*":
  No se crea de manera predeterminada en un nuevo proyecto "*Android*", pero es recomendable utilizarla para almacenar las librerías externas utilizadas por la aplicación.
- Carpeta "*res*":
  Contiene todos los recursos de la aplicación. Como se puede observar en la Figura 7, la carpeta "*res*" contiene a su vez otras subcarpetas:

- **drawable:**
  Contiene las imágenes de la aplicación
- **drawable-hdpi:**
  Contiene las imágenes que se mostrarán solo en dispositivos con pantallas de alta resolución.
- **drawable-ldpi:**
  Contiene las imágenes que se mostrarán solo en dispositivos con pantallas de baja resolución.
- **drawable-mdpi:**
  Contiene las imágenes que se mostrarán solo en dispositivos con pantallas de media resolución.
- **layout:**
  Contiene los archivos xml que representan las interfaces graficas para las actividades de la aplicación.
- **values:**
  Esta carpeta contiene inicialmente el archivo "*strings.xml*", que es donde declaramos las cadenas de texto que usará nuestra aplicación. No es obligatorio definir todas las cadenas de texto allí, pero es bastante recomendable hacerlo, de esta manera que si tenemos que cambiar alguna cadena que se utilice repetidamente a lo largo de la misma, lo podremos hacer de una sola vez.

Archivo "AndroidManifest.xml".

Hay que hacer especial mención al documento "*AndroidManifest.xml*" contenido en la carpeta raíz de cualquier proyecto "*Android*", ya que es la base para el funcionamiento de una aplicación de este tipo. Este documento contendrá las referencias a todos los componentes de la aplicación, como actividades, servicios, proveedores de contenido, etc… que utilice la misma, también contendrá muchos parámetros de configuración, como por ejemplo, su versión objetivo o el tipo de

instalación que se debe realizar en los dispositivos que la descarguen, el tipo de permisos que tendrá para el acceso al *GPS* del dispositivo, el acceso a internet y otros muchos. A continuación se realizará una descripción más detallada del archivo "*AndroidManifest.xml*" utilizado en la aplicación "*Ldapp*", del que se muestra un fragmento en el Código 1.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="es.uc3m.android.ldapp"
      android:versionCode="1"
      android:versionName="1.0"
      android:installLocation="auto">

    <!--
      Definicion de los permisos con los que contará la aplicacion
      en los dispositivos móviles que la contengan
    -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />

    <!--
      Declaracion de la aplicacion en si misma (su nombre) y de las
      actividades   que contendrá
    -->
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".SplashScreen">
            <intent-filter>
               <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Define la utilizacion de esta libreria dentro de la aplicacion -->
        <uses-library android:name="com.google.android.maps"/>
        <activity android:name="es.uc3m.android.ldapp.MapaActivity"></activity>
        <activity android:name="es.uc3m.android.ldapp.PopUp"
                  android:label="@string/app_name"
                        android:theme="@android:style/Theme.Dialog">
        </activity>
        <activity android:name="es.uc3m.android.ldapp.LinkedInActivity"
                  android:label="@string/app_name"
                  android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.EMBED" />
            </intent-filter>
        </activity>
        <activity
               android:name="es.uc3m.android.ldapp.LinkedInProviderActivity"
               android:label="@string/app_name">
            <intent-filter>
           <action android:name="android.intent.action.VIEW"></action>
           <category android:name="android.intent.category.DEFAULT"></category>
           <category android:name="android.intent.category.BROWSABLE"></category>
           <data android:scheme="uc3mmovil" android:host="linkedin"></data>
                       </intent-filter>
        </activity>
    <!--
         Definición de la version minima de la API de Android en la que podrá
         funcionar la aplicacion, y la versión para la que se ha diseñado desde
         un inicio.
     -->
    </application>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8"/>
</manifest>
```

Código 1. Archivo "AndroidManifest.xml".
(Aplicación "*Ldapp*")

Como se puede observar en la Figura 7, la primera línea tras la línea de código que define el inicio del documento xml de este archivo, corresponde a la etiqueta que marca el inicio del manifiesto. En esta etiqueta inicial se definen el numero de la version de la aplicación y la localización donde se realizará la instalación de la aplicación cuando sea descargada en un dispositivo móvil. En este caso la localización está definida como "*auto*" que permitirá al dispositivo realizar la instalación donde

considere en función de de la memoria disponible del dispositivo y de la tarjeta de memoria. Como se muestra en el Código 2.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="es.uc3m.android.ldapp"
      android:versionCode="1"
      android:versionName="1.0"
      android:installLocation="auto">
```

Código 2. Cabecera "AndroidManifest.xml".
(Aplicación "*Ldapp*")

Las siguientes etiquetas que nos encontramos en este documento xml son las etiquetas "*<uses-permission>*" que definirán los permisos de uso de la aplicación sobre el dispositivo móvil para funcionar de forma adecuada, como por ejemplo la funcionalidad de localización , mediante el permiso "*android.permission.ACCESS_FINE_LOCATION*", que permitirá a nuestra aplicación recibir actualizaciones sobre el posicionamiento del dispositivo móvil vía internet o vía *GPS,* ó los permisos "*android.permission.READ_CONTACTS*" *y "android.permission.WRITE_CONTACTS"* que darán permiso a la aplicación de lectura y escritura de contactos del dispositivo. Como se puede obsercar en el Código 3.

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
```

Código 3. Definición de permisos de la aplicación en "AndroidManifest.xml".
(Aplicación "*Ldapp*")

Las etiquetas que suceden a las de definición de los permisos de la aplicación, son las que definirán cada una de las actividades pertenecientes a la misma. Estas están contenidas a su vez, por la etiqueta "*<application>*", en la cual se definen parámetros como por ejemplo la cadena con el nombre de la aplicación, que se mostrará en la parte superior de cada una de las actividades de la misma "*app_name*". Como se puede observar en el Código 4.

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
```

Código 4. Etiqueta "<application>" de "AndroidManifest.xml".
(Aplicación "*Ldapp*")

Cada una de las actividades declaradas dentro de la etiqueta "*<application>*" se definirá mediante la etiqueta "*<activity>*" que como mínimo deberá tener una referencia a la clase Java que la implementa, como se muestra en el Código 5.

```
<activity android:name="es.uc3m.android.ldapp.MapaActivity"></activity>
```

Código 5. Etiqueta "<activity>" de "AndroidManifest.xml". (Aplicación "*Ldapp*")

No obstante, se pueden definir muchas más cualidades para una determinada actividad como parámetros de la etiqueta "*<activity>*", como por ejemplo los estilos que seguirá la propia actividad, los denominados "*themes*", equivalentes a los estilos de *HTML,* ó incluso añadir etiquetas hijas, como por ejemplo "*<intent-filter>*" que definirá el ámbito una determinada actividad. En el caso de nuestra aplicación, "*Ldapp*", utilizamos esta etiqueta en varias ocasiones un ejemplo es el caso que se muestra en el Código 6 donde se incluye para definir la actividad principal de la aplicación.

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Código 6. Etiqueta "<intent-filter>" de "AndroidManifest.xml". (Aplicación "*Ldapp*")

Por ultimo la etiqueta "*<uses-sdk>*" definirá la versión mínima de la sdk de "*Android*" que soportará la aplicación, y la versión objetivo para la que está diseñada la misma, como se muestra en el Código 7.

```
<uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8"/>
```

Código 7. Etiqueta "<uses-sdk>" de "AndroidManifest.xml". (Aplicación "*Ldapp*")

4 Protocolo Ldap

Este capítulo esta enfocado a explicar al lector en que consiste el protocolo Ldap, ya que este proyecto se basa principalmente en la gestión de directorios activos mediante dicho protocolo.

4.1 ¿En qué consiste un servicio de directorio?

Un servicio de directorio consiste fundamentalmente en una aplicación o un conjunto de ellas que almacenan y organizan la información referente a los usuarios de una red y sus recursos, y permiten a los administradores gestionar el acceso de los usuarios a los mismos.

Normalmente, los directorios almacenan sus datos de manera jerárquica, y tienen una morfología en árbol. Cada nodo o entrada de dicho árbol, se encuentra identificada de manera unívoca por su "*Distinguished Name*", en castellano "*Nombre Distinguido*" y que habitualmente se define mediante su forma abreviada "*DN*". Este *DN*, esta compuesto por alguno de los atributos de la propia entrada el denominado "Delative Distinguished Name" o *RDN* y el *DN* del nodo padre. Se puede apreciar un pequeño ejemplo en la Figura 8.

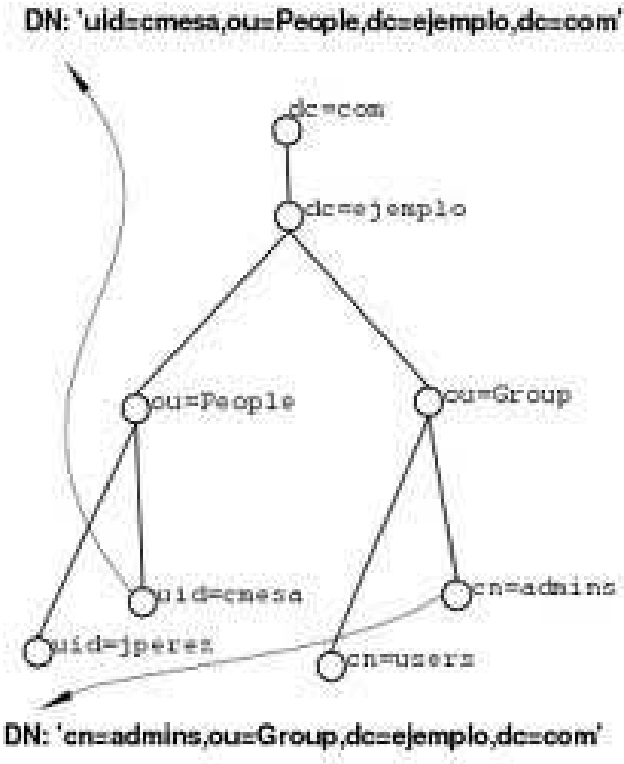


Figura 8. Estructura de un árbol Ldap.

Sobre un directorio se pueden realizar una serie de operaciones como pueden ser:

- Operaciones de búsqueda.
- Operaciones de adición de nuevos nodos.
- Operaciones de edición de nodos.
- Operaciones de borrado de nodos.
- Operaciones de comparación de nodos.
- Operaciones de autenticación y de cierre de la conexión con el directorio (también sobre conexiones seguras).
- Etc…

No obstante, en el siguiente apartado se explicará de una manera más pormenorizada como llevar acabo todas estas operaciones por medio del protocolo *Ldap.*

4.2 Definición de protocolo Ldap

El protocolo *LDAP* ("*Lightweight Directory Access Protocol ")* surgió como una alternativa al protocolo *DAP* ("*Directory Access Protocol*"). Tanto el protocolo *DAP* como *LDAP* cumplen las especificaciones para protocolos de acceso a servicios de directorio, denominado "*X.500*" [20]. Esta especificación fue diseñada por la "*Unión Internacional de las Telecomunicaciones*" (*ITU* por sus siglas en inglés) para sentar las bases en cuanto al acceso a los mismos se refiere. A continuación se muestra en la Figura 9, la estructura que debe seguir un sistema de directorio activo según la especificación *X.500.*

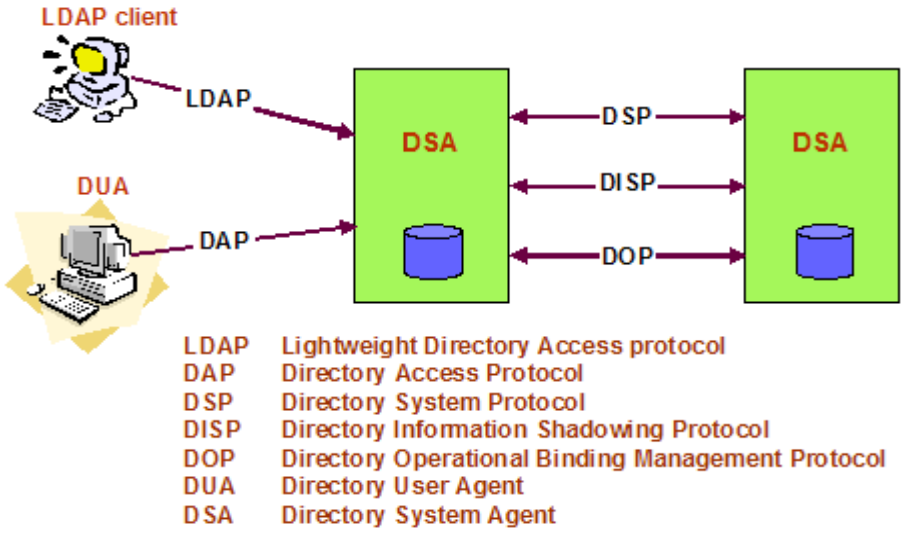


Figura 9. Estructura sistema de directorio.
(especificación *X.500*)

Como se puede observar en la figura, el protocolo *LDAP,* no es más que el lenguaje utilizado para llevar acabo la comunicación con el servicio de directorio que provee un *DSA* ("*Directory System Agent*" o "*Agente de sistema de directorio*").

El proceso que habitualmente se sigue para poder interactuar con un *DSA* mediante el uso del protocolo *LDAP* requiere los siguientes pasos:

1. Crear la conexión con el servidor *LDAP.* Por defecto en sobre el puerto TCP 389, mediante una conexión estándar o segura (conexión TLS en el puerto 389 o SSL en el puerto 636) que se llevará acabo mediante la operación "*Start TLS*".
2. Autenticarse en el servidor *LDAP* mediante la operación "*Bind*", con el usuario que va a acceder a los datos y su contraseña.
3. Ejecutar operaciones contra servidor *LDAP.* Algunas de las operaciones que se pueden llevar acabo contra un servidor *LDAP*, son las siguientes:

    a. **Search**. Buscar y obtener entradas de directorio.
    b. Compare. Probar si una entrada nombreda contiene un valor de atributo dado.
    c. **Add**. Añadir una nueva entrada.
    d. **Delete**. Borrar una entrada determinada.
    e. **Modify**. Modificar una entrada.
    f. **Modify Distinguished Name**. Modificar o renombrar el *DN* de una determinada entrada.
    g. **Abandon**. Cancelar una petición previa.
    h. **Unbind**. Cerrar la conexión.

5 Aplicación Ldapp

Durante el trascurso de este capítulo, se intentará mostrar al lector de una manera lo más precisa posible, como se han llevado acabo los procesos de Análisis, Diseñó e Implementación de la aplicación "*Ldapp*".

5.1 Análisis y Diseño

La aplicación estará enfocada a facilitar al usuario la configuración de conexiones a diferentes servidores LDAP y el filtrado de datos en los mismos. La aplicación integrará la funcionalidad de búsqueda disponible sobre dichos servidores y algunos de los servicios ofrecidos por la plataforma Android sobre dispositivos móviles, tales como:

- *Envío de emails* a las direcciones contenidas en alguna de las entradas filtradas sobre el servidor LDAP.
- *Geoposicionamiento* de una de las entradas seleccionada en el filtrado de datos sobre el LDAP, mediante la dirección postal contenido en dicha entrada, mediante la utilización de la API de Google Maps.
- *Adición a la agenda de contactos del dispositivo móvil* de una entrada seleccionada en el filtrado de datos sobre el servidor LDAP.
- *Realización de llamadas* al número de teléfono de una determinada entrada seleccionada en el filtrado de datos sobre el servidor LDAP.

  En definitiva, la aplicación aportará al usuario más facilidad de acceso a los servidores LDAP, y de gestión de los datos recuperados de ellos, mediante la integración con los servicios dispuestos por la plataforma Android.

Todo el proceso de Análisis y Diseño ha sido llevado a cabo siguiendo el estándar UML 2.0 [6].

5.1.1 Especificación de requisitos

A continuación se detallan las restricciones y requisitos obtenidos para el sistema. Cada uno de ellos tendrá la siguiente estructura:

Identificador de requisito	**Identificador único, que seguirá el siguiente formato en función del tipo de requisito que conforme:** o **RES – XX: tipo de requisito "restricción", seguido de el número identificativo "XX" de dos cifras.** o **RF – XX: tipo de requisito "funcional", seguido de el número identificativo "XX" de dos cifras.** o **RFN – XX: tipo de requisito "no funcional", seguido de el número identificativo "XX" de dos cifras.**
Nombre de requisito	Nombre descriptivo del requisito.
Tipo	Nombre descriptivo del tipo de requisito. o Restricción. o Funcional. o No funcional. o De usabilidad. o De rendimiento.
Fuente del requisito	Fuente de la que se recopilo el requisito, cliente, equipo de desarrollo, etc…
Prioridad del requisito	Define la prioridad con la que debe ser llevado a cabo el requisito.
Descripción requisito	Descripción del requisito.

Tabla 1. Metodología Especificación de Requisitos.

5.1.1.1 Restricciones del proyecto

A continuación se detallan cada una de las restricciones tenidas en cuenta a la hora de implementar la aplicación.

Identificador de requisito	**RES - 1**
Nombre de requisito	Lenguaje programación.
Tipo	Restricción.
Fuente del requisito	Cliente.
Prioridad del requisito	Alta.
Descripción requisito	El lenguaje utilizado para la implementación de la aplicación será "Java", en su versión reducida para plataformas Android.

Tabla 2. RES – 1 Lenguaje programación.

Identificador de requisito	RES - 2
Nombre de requisito	Compatibilidad con versiones plataforma Android.
Tipo	Restricción.
Fuente del requisito	Cliente.
Prioridad del requisito	Alta.
Descripción de requisito	La aplicación será compatible con versiones de Android iguales o superiores a la versión 1.5 “CupCake” y de rendimiento optimo para la versión 2.2 “Froyo”.

Tabla 3. RES – 2 Compatibilidad con versiones plataforma Android.

Identificador de requisito	RES - 3
Nombre de requisito	API comunicación servidor LDAP
Tipo	Restricción
Fuente del requisito	Equipo de desarrollo.
Prioridad del requisito	Alta
Descripción de requisito	Puesto que las librerías core de Java que vienen incluidas en Android no coinciden con la versión “Java Standard Edition” (J2SE), ni tampoco con la versión “Java Micro Edition” (J2ME), son más bien una mezcla de ambas versiones, no contienen los paquetes necesarios para la gestión de servidores LDAP mediante JNDI, se omiten estas librerías contenidas inicialmente en J2SE. Se utilizarán para suplir esta carencia las librerías proporcionadas por “UnboundId” para la comunicación con servidores de directorio LDAP. Dichas librerías son de código abierto y pueden ser distribuidas bajo licencia GPLv2 o LGPLv2.1 [2].

Tabla 4. RES – 3 API comunicación servidor LDAP.

5.1.1.2 Requisitos funcionales

Se entienden por requisitos funcionales aquellas funcionalidades específicas, que definirán como serán llevados los casos de uso a la práctica.

A continuación se detallan cada uno de los requisitos funcionales obtenidos del análisis del sistema.

Identificador de requisito	RF - 1
Nombre de requisito	Gestión de servidor LDAP
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación facilitará al usuario la gestión de Servidores LDAP.

Tabla 5. RF – 1 Gestión de servidor LDAP.

Identificador de requisito	RF - 1.1
Nombre de requisito	Añadir nueva configuración servidor LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación contendrá la funcionalidad de añadir un nuevo servidor LDAP (añadir los datos de configuración del mismo y guardarlo), tanto configuraciones de acceso seguro al servidor LDAP, en las cuales se exigirán las credenciales del usuario (usuario y contraseña), como configuraciones anónimas, en las que el usuario tendrá limitada la funcionalidad sobre el servidor LDAP.

Tabla 6. RF – 1.1 Añadir nueva configuración servidor LDAP.

Identificador de requisito	RF - 1.2
Nombre de requisito	Editar configuración servidor LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación contendrá la funcionalidad de editar un nuevo servidor LDAP (editar los datos de un servidor guardado previamente y guardarlos).

Tabla 7. RF – 1.2 Editar configuración servidor LDAP.

Identificador de requisito	RF - 1.3
Nombre de requisito	Eliminar configuración servidor LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación contendrá la funcionalidad de eliminar uno o varios servidores LDAP al mismo tiempo (borrar los datos de servidores añadidos previamente).

Tabla 8. RF – 1.3 Eliminar configuración servidor LDAP.

Identificador de requisito	RF - 2
Nombre de requisito	Búsqueda en servidor LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación proveerá al usuario de la funcionalidad de búsqueda en un determinado Servidor LDAP basándose en un determinado criterio de búsqueda y un filtro seleccionado por el usuario. Los criterios de búsqueda serán los siguientes: - Nombre - Email - Teléfono - Id

Tabla 9. RF – 2 Búsqueda en servidor LDAP.

Identificador de requisito	RF - 3
Nombre de requisito	Listado resultados búsqueda en servidor LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación listará los resultados de la búsqueda realizada sobre el Servidor LDAP, dando la posibilidad de seleccionar uno de ellos para la posterior visualización de los datos de esa entrada del LDAP.

Tabla 10. RF – 3 Listado resultados búsqueda en servidor LDAP.

Identificador de requisito	RF - 4
Nombre de requisito	Detalle entrada LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación mostrará al usuario los siguientes datos de la entrada del LDAP seleccionada de los resultados de la búsqueda: “Nombre”, “Apellidos”, “Email”, y “Teléfono” si tuviera, en caso de no tener teléfono, dicho campo no se mostrará en el detalle de la entrada.

Tabla 11. RF – 4 Detalle entrada LDAP.

Identificador de requisito	RF - 4.1
Nombre de requisito	Añadir atributo a detalle entrada LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario tendrá la posibilidad de visualizar algunos datos más de los mostrados por defecto para esa entrada, la colección de datos disponibles para ser mostrados dependerá del tipo de entrada que estemos visualizando, el número de los mismos variará en función de los tipos de objetos (persona, organización…) que implemente dicha entrada.

Tabla 12. RF – 4.1 Añadir atributo a detalle entrada LDAP.

Identificador de requisito	RF - 4.2
Nombre de requisito	Añadir entrada LDAP a agenda contactos del dispositivo.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario tendrá la posibilidad de añadir la entrada del servidor LDAP visualizada a la agenda de contactos del móvil.

Tabla 13. RF – 4.2 Añadir entrada LDAP a agenda contactos.

Identificador de requisito	RF - 4.3
Nombre de requisito	Marcar número de teléfono de la entrada LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario tendrá la posibilidad de llamar al teléfono obtenido para la entrada del servidor LDAP seleccionada en caso de tener dicho campo relleno, en caso contrario se mostrará un aviso al usuario indicándole que no se puede realizar la acción.

Tabla 14. RF – 4.3 Marcar número de teléfono de la entrada.

Identificador de requisito	RF - 4.4
Nombre de requisito	Redactar email para entrada LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario tendrá la posibilidad de redactar un nuevo email dirigido a la dirección de correo electrónico de la entrada del servidor ldap seleccionado en caso de tener dicho campo relleno, en caso contrario se mostrará un aviso al usuario indicándole que no se puede realizar la acción.

Tabla 15. RF – 4.4 Redactar email para entrada LDAP.

Identificador de requisito	RF - 4.5
Nombre de requisito	Geo-posicionar dirección entrada LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario tendrá la posibilidad de geo-posicionar por internet la ubicación de la “dirección” de la entrada seleccionada en los mapas provistos por Google Maps.

Tabla 16. RF – 4.5 Geo-posicionar dirección entrada LDAP.

Identificador de requisito	RF - 4.6
Nombre de requisito	Búsqueda en LinkedIn de entrada LDAP.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario tiene la posibilidad de realizar una búsqueda en LinkedIn sobre la entrada del servidor ldap seleccionada, el filtrado se realizará sobre el nombre y apellidos de la entrada en cuestión.

Tabla 17. RF – 4.6 Búsqueda en LinkedIn de entrada LDAP.

Identificador de requisito	RF - 4.6.1
Nombre de requisito	Autenticación Usuario en LinkedIn.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	Si el usuario selecciona la opción de "búsqueda en LinkedIn" y el usuario aun no ha concedido permiso a la aplicación para tener acceso a los datos de LinkedIn, se le mostrará la pantalla de log in de LinkedIn para que introduzca sus credenciales y habilite esta funcionalidad de la aplicación. Si el usuario ya había realizado esta operación en otro acceso a la aplicación, se realizará la búsqueda directamente, y se listarán los resultados de la misma.

Tabla 18. RF – 4.6.1 Autenticación usuario en LinkedIn.

Identificador de requisito	RF - 4.6.2
Nombre de requisito	Listado resultados búsqueda en LinkedIn.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El resultado se listará y se paginará si son demasiados registros para una sola página.

Tabla 19. RF – 4.6.2 Listado resultados búsqueda en LinkedIn.

Identificador de requisito	RF - 4.6.3
Nombre de requisito	Visualizar ficha contacto LinkedIn.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El usuario podrá seleccionar uno de los registros de LinkedIn listados como resultado de la búsqueda, para poder visualizar la ficha web del contacto en cuestión.

Tabla 20. RF – 4.6.3 Visualizar ficha contacto LinkedIn.

5.1.1.3 Requisitos no funcionales

Se entiende por requisitos no funcionales todos aquellos que no describen ningún tipo de información a guardar ni funcionalidades del sistema. Se incluirán por consiguiente todos los requisitos relacionados con apariencia ("look and feel"), usabilidad, estabilidad, coste, etc…

A continuación se detallan todos los requisitos no funcionales obtenidos para nuestro sistema.

Identificador de requisito	**RNF – 1**
Nombre de requisito	Idioma de la aplicación.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	El Idioma por defecto utilizado en la aplicación será el Español, dejando para futuras ampliaciones la posibilidad de añadir un segundo idioma.

Tabla 21. RNF – 1 Idioma de la aplicación.

Identificador de requisito	**RNF – 2**
Nombre de requisito	Combinación de colores de la aplicación.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	La aplicación combinará colores negros y verdes en todas sus pantallas, pudiendo utilizar varias tonalidades de los mismos en busca de un buen contraste que resalte las funcionalidades más importantes de cada pantalla, y facilite al usuario el reconocimiento de las mismas a simple vista. Asimismo, el color utilizado para el texto será el blanco.

Tabla 22. RNF – 2 Combinación de colores de la aplicación.

Identificador de requisito	**RNF – 3**
Nombre de requisito	Mensajes de aviso en demora de operaciones.
Tipo	Requisito
Fuente del requisito	Equipo de desarrollo.
Prioridad del requisito	Alta
Descripción de requisito	Cuando se lleve acabo cualquier operación que se demoré demasiado, el usuario será advertido de ello mediante una barra de progreso, que le informará del estado de la misma.

Tabla 23. RNF – 3 Mensajes de aviso en demora de operaciones.

Identificador de requisito	**RNF – 4**
Nombre de requisito	Paginación de resultados en las búsquedas.
Tipo	Requisito
Fuente del requisito	Cliente
Prioridad del requisito	Alta
Descripción de requisito	En caso de recibir una cantidad elevada de registros en una búsqueda sobre algún servidor LDAP, estos serán paginados, mejorando de esta manera el rendimiento de la aplicación y evitando un ralentizamiento excesivo de la misma.

Tabla 24. RNF – 4 Paginación de resultados en las búsquedas.

5.1.2 Casos de Uso

5.1.2.1 Diagrama de Casos de Uso

A continuación se detalla el diagrama de casos de uso perteneciente a toda la aplicación. En el que se pueden apreciar cada una de las funcionalidades que implementará la aplicación, así como todos los tipos de usuarios y sistemas que interactuarán con la misma.

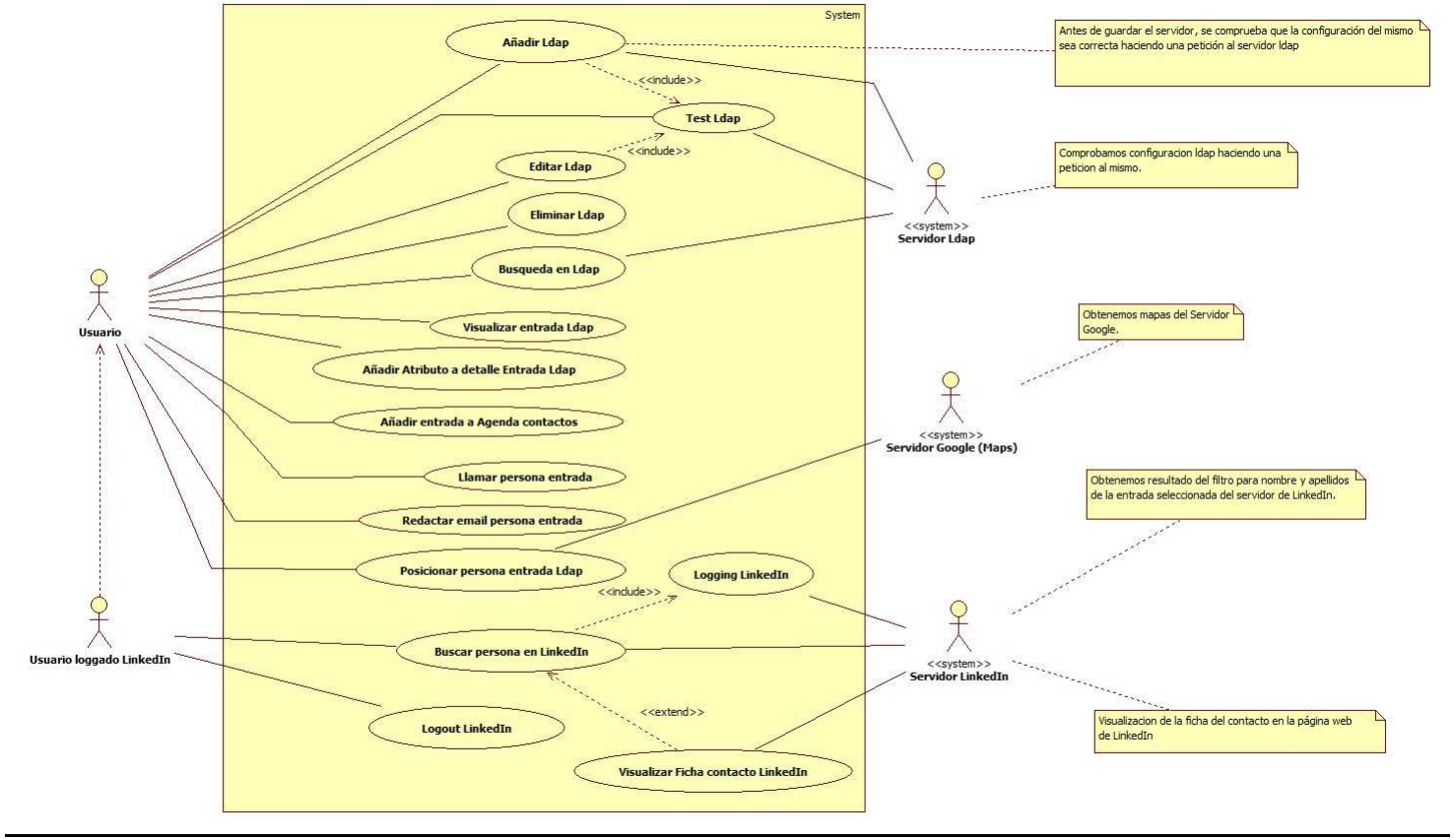


Figura 10. Diagrama de casos de uso de la aplicación.

5.1.2.2 Especificación de Casos de Uso

5.1.2.2.1 Especificación de Actores

A continuación se detalla cada uno de los actores que interactuarán con los casos de uso del sistema. Cada definición de Autor se estructurará de la siguiente manera:

Identificador	**AC – XX, las letras AC, indican que se trata de un Actor, y "XX" representa el número de 2 cifras identificativo del Actor.**
Actor	Nombre del Actor.
Casos de Uso	Casos de uso con los que interactúa el Actor.
Tipo	Primario, Secundario, según la importancia del mismo en el sistema.
Relaciones	Relaciones con otros Actores, si es que las tuviera.
Descripción	Descripción del Actor.

Tabla 25. Metodología Especificación de Actores.

Los actores pertenecientes al sistema son los siguientes:

Identificador	**AC - 01**
Actor	Usuario.
Casos de Uso	o Añadir Ldap. o Test Ldap. o Editar Ldap. o Eliminar Ldap. o Búsqueda Ldap. o Visualizar entrada Ldap. o Añadir entrada a Contactos. o Llamar persona entrada Ldap. o Redactar email persona entrada. o Posicionar persona entrada Ldap.
Tipo	Primario.
Relaciones	Ninguna.
Descripción	Usuario estándar aplicación.

Tabla 26. AC – 01 Usuario.

Identificador	**AC - 02**
Actor	Usuario Logado LinkedIn.
Casos de Uso	o Búsqueda personas LinkedIn. o Logout LinkedIn.
Tipo	Primario.
Relaciones	Usuario.
Descripción	Usuario logado en LinkedIn desde la aplicación. Este usuario habrá dado permiso a la aplicación para poder conectarse a su cuenta de LinkedIn.

Tabla 27. AC – 02 Usuario Logado LinkedIn.

Identificador	**AC - 03**
Actor	Servidor Ldap.
Casos de Uso	o Test Ldap. o Búsqueda Ldap.
Tipo	Secundario.
Relaciones	Ninguna.
Descripción	Servidor Ldap registrado previamente en la aplicación. Nos devolverá los resultados de los filtrados realizados sobre el mismo.

Tabla 28. AC – 03 Servidor Ldap.

Identificador	**AC – 04**
Actor	Servidor Google Maps.
Casos de Uso	o Posicionar persona Ldap.
Tipo	Secundario.
Relaciones	Ninguna.
Descripción	Servidor de Google, encargado de geo posicionar a la persona que hayamos seleccionado de una búsqueda en el ldap. Lo hará tomando como referencia la dirección de dicha entrada del ldap.

Tabla 29. AC – 04 Servidor Google Maps.

Identificador	**AC - 05**
Actor:	Servidor LinkedIn.
Casos de Uso:	o Login LinkedIn. o Visualizar ficha contacto LinkedIn.
Tipo:	Secundario.
Relaciones:	Ninguna.
Descripción:	Servidor de LinkedIn, devuelve a la aplicación los resultados para el filtrado por nombre y apellidos de la entrada seleccionada de la búsqueda en el ldap.

Tabla 30. AC – 05 Servidor LinkedIn.

5.1.2.2.2 Especificación de Casos de Uso

A continuación se detalla cada uno de los casos de uso del sistema. Cada definición de Caso de Uso se estructurará de la siguiente manera:

Identificador	**CU – XX, "CU" indica que se trata de un Caso de Uso, y "XX" será el número de 2 cifras identificativo del cada caso de uso.**	
Nombre	Nombre identificativo de cada caso de uso	
Descripción	Descripción del caso de uso.	
Actores	Actores que interactúan con el caso de uso.	
Precondición	Precondiciones que han de cumplirse para poder llevar acabo el caso de uso.	
Postcondición	Postcondiciones que se cumplirán una vez se haya llevado acabo el caso de uso.	
Secuencia normal	Secuencia normal del flujo de ejecución del caso de uso. Cada paso de la secuencia tendrá un identificador numérico y una descripción de la acción que se llevará acabo en este paso.	
	Paso	**Acción**
[Secuencia alternativa]	Secuencia alternativa del flujo de ejecución del caso de uso normal. Cada paso de esta secuencia tendrá un identificador numérico y una descripción de la acción que se llevará acabo en este paso. Sección opcional.	
	Paso	**Acción**
[Excepciones]	Conjunto de pasos que se llevarán acabo cuando durante la ejecución de la secuencia normal de ejecución del caso de uso haya una excepción. Cada paso de la excepción se identificará por un identificador numérico y una descripción de la acción que se llevará acabo en este paso.	
	Paso	**Acción**
Importancia	Alta, Media o Baja, en función de la prioridad de implementación del caso de uso.	
Comentarios	Comentarios extra sobre el caso de uso.	

Tabla 31. Metodología Especificación de Casos de Uso.

A continuación se detallan todos los casos de uso obtenidos para el sistema:

Identificador	**CU-1**	
Nombre	Añadir Ldap.	
Descripción	Configura parámetros para un nuevo ldap y lo guarda en la BBDD de la aplicación.	
Actores	Usuario.	
Precondición	El campo Id, debe ser único, no podrá ser igual a ninguno de los introducidos previamente.	
Postcondición	Si los datos introducidos son correctos, los cambios serán guardados en BBDD y el servidor se mostrará en el listado de servidores ldap (pantalla inicial).	
Secuencia normal	**Paso**	**Acción**
	1	Usuario inserta parámetros nuevo ldap.
	2	La aplicación envía una petición de conexión al ldap, para comprobar que los datos introducidos son correctos.
	3	La aplicación recibe una respuesta satisfactoria del servidor ldap.
	4	La aplicación guarda la nueva configuración ldap en su BBDD.
	5	La aplicación muestra al usuario una ventana de dialogo con el mensaje de confirmación de salvado del ldap.
Excepciones	**Comunicación servidor Ldap**	
	Paso	**Acción**
	1	Usuario inserta parámetros nuevo ldap.
	2	La aplicación envía una petición de conexión al ldap, para comprobar que los datos introducidos son correctos.
	3a	La aplicación recibe una respuesta de error (bien por que no se haya podido comunicar con el servidor, o bien por que alguna credencial sea incorrecta, en tal caso el error lo devolverá el propio servidor ldap).
	4a	La aplicación muestra al usuario el mensaje de error.
Importancia	Alta	
Comentarios		

Tabla 32. CU – 1 Añadir LDAP.

Identificador	**CU-2**	
Nombre	Editar Ldap.	
Descripción	Edición de una de las configuraciones ldap introducidas previamente.	
Actores	Usuario.	
Precondición	Tiene que existir alguna configuración ldap, añadida previamente.	
Postcondición	Si los datos introducidos son correctos, los cambios serán guardados en BBDD y el servidor con los cambios realizados se mostrará en el listado de servidores ldap (pantalla inicial).	
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona una configuración del listado de servidores ldap.
	2	Usuario selecciona acción a realizar sobre el servidor ldap seleccionado (en este caso “edición”).
	3	La aplicación muestra el detalle del servidor ldap, posibilitando al usuario modificar los datos del mismo.
	4	Usuario realiza modificaciones.
	5	La aplicación muestra al usuario una ventana de dialogo con el mensaje de confirmación de salvado del ldap.
Excepciones	**Comunicación servidor Ldap**	
	Paso	**Acción**
	1	Usuario selecciona una configuración del listado de servidores ldap.
	2	Usuario selecciona acción a realizar sobre el servidor ldap seleccionado (en este caso “edición”).
	3	La aplicación muestra el detalle del servidor ldap, posibilitando al usuario modificar los datos del mismo.
	4	Usuario realiza modificaciones.
	5a	La aplicación muestra al usuario una ventana de dialogo con el mensaje de error.
Importancia	Alta.	
Comentarios		

Tabla 33. CU – 2 Editar LDAP.

Identificador	**CU-3**	
Nombre	Eliminar Ldap.	
Descripción	Borrado de una de las configuraciones ldap introducidas previamente.	
Actores	Usuario.	
Precondición	1. La configuración ldap debe haber sido añadida previamente. 2. Para poder eliminar algún servidor ldap, previamente, tendremos que haberlo chequeado en el listado.	
Postcondición	La configuración ldap desaparecerá del listado de servidores ldap.	
Secuencia normal	**Paso**	**Acción**
	1	Usuario chequea una o varias configuraciones del listado de servidores Ldap que desea eliminar.
	2	La aplicación busca en BBDD la configuración del ldap por su id que fue chequeada y la elimina de BBDD.
	3	La aplicación actualiza listado servidores ldap. (Ya no aparecerán los que hayan sido borrados)
Excepciones	**Comunicación servidor Ldap**	
	Paso	**Acción**
	1	Usuario chequea una o varias configuraciones del listado de servidores Ldap que desea eliminar.
	2	La aplicación busca en BBDD la configuración del ldap por su id, pero se produce un error al eliminar el registro de BBDD.
	3a	La aplicación muestra el error al usuario mediante una ventana de dialogo.
Importancia	Alta.	
Comentarios		

Tabla 34. CU – 3 Eliminar LDAP.

Identificador	**CU-4**	
Nombre	Test Ldap.	
Descripción	Test de una de las configuraciones ldap, durante la acción de añadir o editar servidor ldap, desde la pantalla de detalle del servidor.	
Actores	Usuario.	
Precondición	Los campos del detalle del servidor deben estar cumplimentados.	
Postcondición		
Secuencia normal	**Paso**	**Acción**
	1	Usuario rellena o edita campos del detalle de servidor ldap.
	2	La aplicación realiza conexión de prueba con servidor ldap para los parámetros del ldap del detalle.
	3	La aplicación recibe conexión satisfactoria del servidor ldap.
	4	La aplicación muestra a el usuario el mensaje de “comunicación exitosa con ldap”
Excepciones	**Test devuelve error**	
	Paso	**Acción**
	1	Usuario rellena o edita campos del detalle de servidor ldap.
	2	La aplicación realiza conexión de prueba con servidor ldap para los parámetros del ldap del detalle.
	3a	La aplicación recibe error de conexión con el servidor ldap.
	4a	La aplicación muestra a el usuario el error recibido del servidor ldap.
Importancia	Alta.	
Comentarios	Este test lo podrá realizar manualmente el Usuario antes de guardar la configuración, pero se realizará de todas formas de manera automática cuando el usuario guarde los cambios al añadir o editar.	

Tabla 35. CU – 4 Test LDAP.

Identificador	**CU-5**	
Nombre	Búsqueda en Ldap.	
Descripción	Búsqueda, filtrando por: - Email. - Nombre. - Apellidos. - Teléfono. - Id. Para una de las configuraciones ldap guardadas previamente.	
Actores	Usuario.	
Precondición	El Usuario debe haber introducido una configuración ldap previamente.	
Postcondición		
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona el criterio de búsqueda.
	2	Usuario añade el filtro de búsqueda.
	3	La aplicación envía petición de búsqueda al servidor ldap.
	4	Servidor Ldap devuelve resultados de la búsqueda.
	5	La aplicación muestra listado resultados al Usuario.
Excepciones	**Error comunicación con el servidor Ldap**	
	Paso	**Acción**
	1	Usuario selecciona un criterio de búsqueda.
	2	Usuario añade el filtro de búsqueda.
	3a	La aplicación intenta realizar vínculo de conexión con el ldap, pero no se recibe respuesta del mismo.
	4a	La aplicación muestra ventana de dialogo con “error de comunicación con el ldap”.
Importancia	vital	
Comentarios		

Tabla 36. CU – 5 Búsqueda en LDAP.

Identificador	**CU-6**	
Nombre	Visualizar entrada Ldap.	
Descripción	Desde el listado de resultados de la búsqueda realizada en el ldap, podemos acceder al detalle de una de las entradas encontradas, pulsando en ella.	
Actores	Usuario.	
Precondición	El Usuario debe haber realizado una búsqueda en una de las configuraciones ldap añadidas previamente.	
Postcondición		
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona una de las entradas devueltas por la búsqueda en el servidor ldap.
	2	La aplicación muestra el detalle para la entrada seleccionada por el usuario.
Excepciones		
	Paso	**Acción**
Importancia	Alta.	
Comentarios		

Tabla 37. CU – 6 Visualizar entrada LDAP.

Identificador	**CU-7**	
Nombre	Añadir Atributo a detalle Entrada Ldap.	
Descripción	Desde el detalle de la entrada Ldap tenemos la posibilidad de añadir atributos no mostrados previamente en dicho detalle.	
Actores	Usuario.	
Precondición	La entrada seleccionada tiene que tener algún campo más de los que se muestran en el detalle.	
Postcondición	El atributo seleccionado se añadirá al listado de atributos del detalle de entrada.	
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona del listado de atributos, el nombre del atributo que desea visualizar en el detalle.
	2	La aplicación muestra el atributo en el listado de atributos del detalle de la entrada.
Excepciones	**Atributo ya existe en listado atributos del detalle**	
	Paso	**Acción**
	1	Usuario selecciona del listado de atributos, el nombre del atributo que desea visualizar en el detalle.
	2a	La aplicación muestra aviso “atributo ya se encuentra en listado de atributos de la entrada”.
Importancia	Alta.	
Comentarios		

Tabla 38. CU – 7 Añadir atributo a detalle entrada LDAP.

Identificador	**CU-8**	
Nombre	Añadir entrada Ldap a Agenda contactos.	
Descripción	Añadir entrada ldap a la agenda de contactos del teléfono, abrirá la pantalla de añadir contacto, con los campos rellenos por los atributos de la entrada del ldap seleccionada previamente.	
Actores	Usuario.	
Precondición		
Postcondición	Los datos del contacto se guardarán en la memoria del teléfono.	
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona opción de añadir a contactos del detalle de entrada ldap.
	2	La aplicación muestra la pantalla de añadir contacto a la agenda con los campos “teléfono”, “nombre” e “email” rellenos con los datos de la entrada ldap.
	3	Usuario cumplimenta campos vacíos de la pantalla de añadir contacto.
	4	Se guarda contacto en memoria del teléfono.
Excepciones	**Atributo ya existe contacto en la agenda del teléfono**	
	Paso	**Acción**
	1	Usuario selecciona opción de añadir a contactos del detalle de entrada ldap.
	2	La aplicación muestra la pantalla de añadir contacto a la agenda con los campos “teléfono”, “nombre” e “email” rellenos con los datos de la entrada ldap.
	3	Usuario cumplimenta campos vacíos de la pantalla de añadir contacto.
	4a	La aplicación muestra mensaje error, “El contacto ya se encuentra en la agenda de contactos”.
Importancia	Alta.	
Comentarios		

Tabla 39. CU – 8 Añadir entrada LDAP a agenda de contactos.

Identificador	**CU-9**	
Nombre	Llamar persona entrada LDAP.	
Descripción	Llamar persona entrada, abrirá la pantalla de realizar llamada, con el campo del teléfono a marcar relleno con el teléfono de la entrada del ldap .	
Actores	Usuario.	
Precondición	La entrada del ldap tendrá que contener el atributo “teléfono”.	
Postcondición		
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona la opción llamar de la pantalla de detalle de la entrada.
	2	La aplicación muestra la pantalla de marcado con el teléfono de la entrada del ldap.
Excepciones	**La entrada del ldap no tiene atributo “teléfono”**	
	Paso	**Acción**
	1	Usuario selecciona la opción llamar de la pantalla de detalle de la entrada.
	2a	La aplicación comprueba si la entrada que estamos mostrando en el detalle contiene el atributo “teléfono”, como no existe muestra aviso.
Importancia	Alta.	
Comentarios		

Tabla 40. CU – 9 Llamar persona entrada LDAP.

Identificador	**CU-10**	
Nombre	Redactar email persona entrada LDAP.	
Descripción	Redactar email persona entrada, abrirá la pantalla de nuevo email, con el campo “para” relleno con el email de la entrada del ldap.	
Actores	Usuario.	
Precondición	La entrada del ldap tendrá que contener el atributo “email”.	
Postcondición		
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona la opción “enviar email” de la pantalla de detalle de entrada.
	2	La aplicación muestra la pantalla de nuevo email con el campo “para” relleno con el email de la entrada ldap.
Excepciones	**La entrada del ldap no tiene atributo “email”**	
	Paso	**Acción**
	1	Usuario selecciona la opción “enviar email” de la pantalla de detalle de entrada.
	2a	La aplicación comprueba si la entrada que estamos mostrando en el detalle contiene el atributo “email”, como no existe muestra mensaje de aviso al Usuario.
Importancia	Alta.	
Comentarios		

Tabla 41. CU – 10 Redactar email persona entrada LDAP.

Identificador	**CU-11**	
Nombre	Posicionar persona entrada Ldap	
Descripción	Abrirá la pantalla de Google Maps, y con la dirección de la entrada, hará una petición al servidor de Maps, para posicionar en el mapa a la persona de la entrada ldap.	
Actores	Usuario.	
Precondición	1. La entrada del ldap deberá contener el atributo “dirección”, tendrá que existir y contener un valor de dirección en un formato correcto. 2. El Usuario tendrá que tener habilitada la conexión a internet del móvil (mediante wifi o 3G) .	
Postcondición	Se mostrará en la pantalla del Maps la posición de la dirección para la entrada ldap.	
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona la opción “localizar en mapa” de la pantalla de detalle de la entrada.
	2	La aplicación abre la pantalla de Google Maps.
	3	La aplicación envía petición de posicionamiento mandando la dirección de la entrada ldap a servidor Google Maps.
	4	Aplicación descarga mapas y posición de servidor Google Maps.
	5	Aplicación muestra posición en mapa de la dirección de la entrada ldap al usuario.
Excepciones	**La entrada del ldap no tiene atributo “dirección”**	
	Paso	**Acción**
	1	Usuario selecciona la opción “localizar en mapa” de la pantalla de detalle de la entrada.
	2a	La aplicación comprueba si la entrada que estamos mostrando en el detalle contiene el atributo “dirección”, como no existe muestra aviso al Usuario.
	Error conexión con Servidor Google Maps	
	Paso	**Acción**
	1	Usuario selecciona la opción “localizar en mapa” de la pantalla de detalle de la entrada.
	2b	La aplicación comprueba si la entrada que estamos mostrando en el detalle contiene el atributo

		"dirección", existe, así que realiza petición a servidor de Google Maps.
	3b	La aplicación no recibe respuesta del servidor de Google Maps y muestra error al Usuario.
Importancia	Alta.	
Comentarios		

Tabla 42. CU – 11 Posicionar persona entrada LDAP.

Identificador	**CU-12**	
Nombre	Buscar persona en LinkedIn	
Descripción	Buscará los contactos que coinciden en nombre y apellidos con los de la entrada del ldap y listará el resultado de la búsqueda.	
Actores	Usuario.	
Precondición	1. La entrada del ldap deberá contener los atributos "nombre" y "apellidos". 2. El Usuario tendrá que tener habilitada la conexión a internet del móvil (mediante wifi o 3G) . 3. El Usuario tendrá que logarse en LinkedIn y permitir el acceso a los datos a la aplicación (la aplicación redireccionará al usuario a la página web de login cuando seleccione esta opción de búsqueda antes de realizar la operación).	
Postcondición	Se listarán los resultados de la búsqueda.	
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona opción "búsqueda en LinkedIn" de la pantalla de detalle entrada del Ldap.
	2	La aplicación comprueba si el Usuario ha dado permiso a la aplicación para el acceso a datos de LinkedIn.
	3	La aplicación tiene acceso concedido a los datos de LinkedIn, así que realiza petición de búsqueda enviando "nombre" y "apellido" para el filtrado a el servidor de LinkedIn
	4	Servidor de LinkedIn devuelve resultado filtrado.
	5	Aplicación muestra listado paginado de los resultados al Usuario.
Secuencia alternativa	**El usuario no se ha logado y ha concedido**	

	permisos a la aplicación para el acceso a datos en LinkedIn	
	Paso	**Acción**
	1	Usuario selecciona opción “búsqueda en LinkedIn” de la pantalla de detalle entrada del Ldap.
	2a	La aplicación comprueba si el Usuario ha dado permiso a la aplicación para el acceso a datos de LinkedIn, en este caso no es así.
	3a	La aplicación abre navegador y carga pantalla de login en LinkedIn, desde la que el usuario tras logarse podrá dar permisos a la aplicación.
Importancia	Alta.	
Comentarios		

Tabla 43. CU – 12 Buscar persona en linkedIn.

Identificador	**CU-13**	
Nombre	Login LinkedIn	
Descripción	Permite al Usuario logarse en LinkedIn y conceder acceso a sus datos de LinkedIn a la aplicación.	
Actores	Usuario.	
Precondición	Usuario no tiene que estar logado previamente en LinkedIn.	
Postcondición	Aplicación tendrá permisos para la búsqueda en LinkedIn.	
Secuencia normal	**Paso**	**Acción**
	1	Usuario selecciona opción de “búsqueda en LinkedIn” en la pantalla de detalle de la entrada del Ldap.
	2	La aplicación comprueba si el Usuario ha dado permiso a la aplicación para el acceso a datos de LinkedIn.
	3	La aplicación abre navegador y carga pantalla de login en LinkedIn, desde la que el usuario tras logarse podrá dar permisos a la aplicación.
Excepciones	**Error de conexión.**	
	Paso	**Acción**
	1	Usuario selecciona opción de “búsqueda en LinkedIn” en la pantalla de detalle de la entrada del Ldap.
	2	La aplicación comprueba si el Usuario ha dado permiso a la

		aplicación para el acceso a datos de LinkedIn.
	3a	La aplicación intenta cargar la pagina de login de LinkedIn en el navegador, pero no tiene la conexión a internet habilitada.
	4a	La aplicación muestra mensaje de error de conexión.
Importancia	Alta.	
Comentarios		

Tabla 44. CU – 13 Login LinkedIn.

Identificador	**CU-14**	
Nombre	Visualizar Ficha contacto LinkedIn.	
Descripción	Permite al Usuario visualizar la ficha del contacto de LinkedIn seleccionada del listado de resultados de la búsqueda en el mismo (abrirá la ficha del contacto en la web de LinkedIn).	
Actores	Usuario logado LinkedIn.	
Precondición	Haber obtenido algún registro en el listado de la búsqueda en LinkedIn.	
Postcondición		
Secuencia normal	**Paso**	**Acción**
	1	Usuario clica un registro del listado resultado de la búsqueda en LinkedIn y selecciona la opción de “visualizar ficha contacto”.
	2	Aplicación abre el navegador, y carga ficha del contacto en la web de LinkedIn.
Excepciones	**Error de conexión.**	
	Paso	**Acción**
	1	Usuario clica un registro del listado resultado de la búsqueda en LinkedIn y selecciona la opción de “visualizar ficha contacto”.
	2a	La aplicación intenta cargar la página de login de LinkedIn en el navegador, pero no tiene la conexión a internet habilitada.
	3a	La aplicación muestra mensaje de error de conexión.
Importancia	Alta.	
Comentarios		

Tabla 45. CU – 14 Visualizar ficha contacto LinkedIn.

Identificador	**CU-15**	
Nombre	Logout LinkedIn.	
Descripción	Permite al Usuario realizar el logout de LinkedIn, retirando así los permisos de acceso a datos de la aplicación sobre la cuenta de LinkedIn del usuario.	
Actores	Usuario logado LinkedIn.	
Precondición	Usuario debe haberse logado en LinkedIn previamente.	
Postcondición	1- Logout del usuario en LinkedIn 2- La aplicación deja de tener acceso a los datos de la cuenta del Usuario en LinkedIn	
Secuencia normal	**Paso**	**Acción**
	1	Usuario clica opción de logout desde el menú contextual de cualquiera de las pantallas de la aplicación.
	2	Aplicación se desvincula de la cuenta del Usuario en LinkedIn
Excepciones	**Error de borrado del registro de LinkedIn.**	
	Paso	**Acción**
	1	Usuario clica opción de logout desde el menú contextual de cualquiera de las pantallas de la aplicación.
	2a	La aplicación intenta eliminar el registro en memoria que le permite acceso a LinkedIn, pero hay un error en el borrado.
	3a	La aplicación muestra mensaje de error al Usuario.
Importancia	Alta.	
Comentarios		

Tabla 46. CU – 15 Logout LinkedIn.

5.1.3 Diagramas de Actividad

Mediante la inclusión de estos diagramas de actividad se pretende profundizar en el flujo de control entre actividades que se llevarán acabo dentro de cada uno de los casos de uso expuestos previamente.

5.1.3.1 Diagrama de Actividad – "Añadir Ldap"

El diagrama de Actividad que se muestra a continuación muestra el flujo de control seguido por la aplicación cuando el usuario realiza la operación de añadir un nuevo servidor Ldap.

Una vez que el usuario ha terminado de completar el formulario de alta del servidor Ldap, y confirma que los datos son correctos, la aplicación realiza una prueba de comunicación de la nueva conexión. Si el servidor Ldap emite una respuesta satisfactoria, los datos del nuevo servidor Ldap serán persistidos en BBDD y se mostrará un dialogo al usuario informándole de ello. En caso de no obtener respuesta del servidor Ldap, directamente se mostrará un dialogo al usuario indicándole el error sucedido y no se persistirá el nuevo servidor Ldap en BBDD.

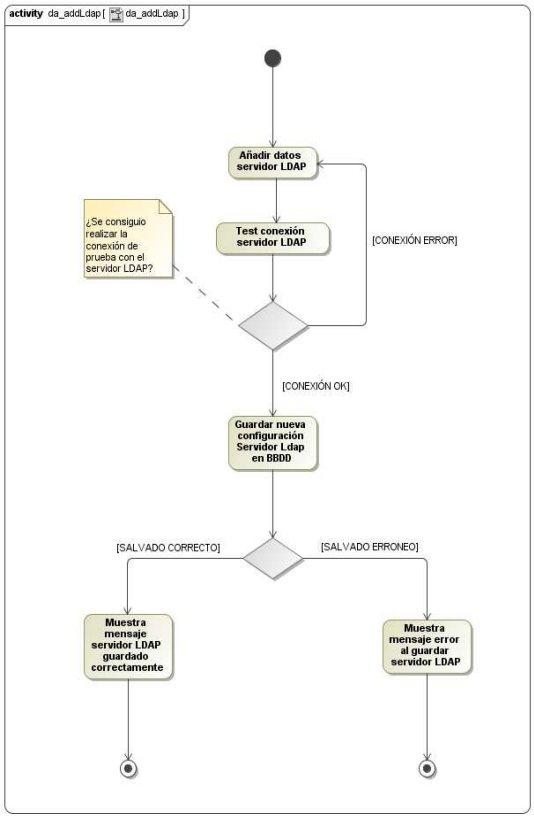


Figura 11. Diagrama de Actividad "Añadir LDAP".

5.1.3.2 Diagrama de Actividad – "Editar Ldap"

El flujo de control de este caso de uso es similar al del caso de uso de añadir un nuevo servidor Ldap, ya que el punto de partida es el mismo, el formulario de detalle de servidor Ldap. Al igual que en el caso de uso anterior, cuando el usuario termina de editar los datos del servidor Ldap, se realiza una prueba comunicación con el servidor Ldap, si es satisfactoria, se persistirá dicha configuración del servidor Ldap en BBDD, y se mostrará un dialogo al usuario indicándole la correcta finalización de la operación. En caso de recibir una respuesta errónea del servidor Ldap o de no recibirla, se mostrará un dialogo al usuario indicándole el error acontecido.

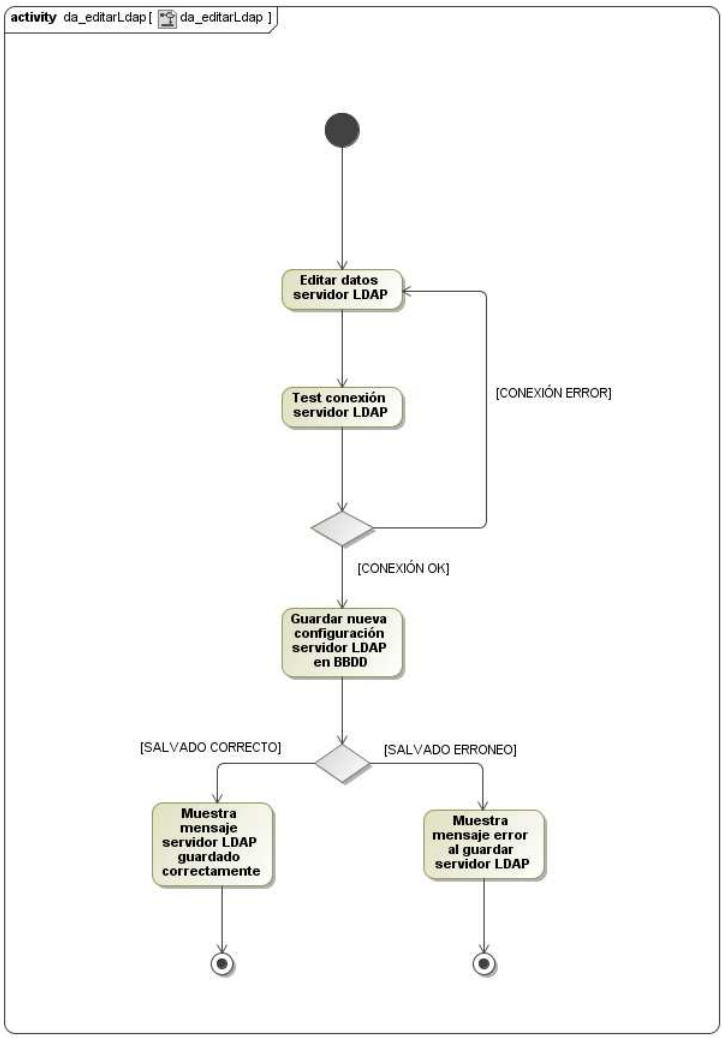


Figura 12. Diagrama de Actividad "Editar LDAP".

5.1.3.3 Diagrama de Actividad – "Eliminar Ldap"

El caso de uso correspondiente a la eliminación de uno o varios servidores Ldap seguirá el siguiente flujo de control (mirar apartado *5.1.2*). Cuando el usuario pulsa el botón de confirmación de borrado, en primer lugar se comprueba si se ha seleccionado alguna configuración de servidor Ldap para ser eliminada. En caso afirmativo (si el número de servidores Ldap seleccionados es >= 1) se procede a eliminar dichos servidores Ldap de BBDD y si el borrado se lleva a cabo de manera satisfactoria, se le informará de ello a el usuario por medio de un dialogo. Si ocurre un error en BBDD y no se puede llevar a cabo dicho borrado, también se informará a el usuario mediante un dialogo que avisará de tal suceso.

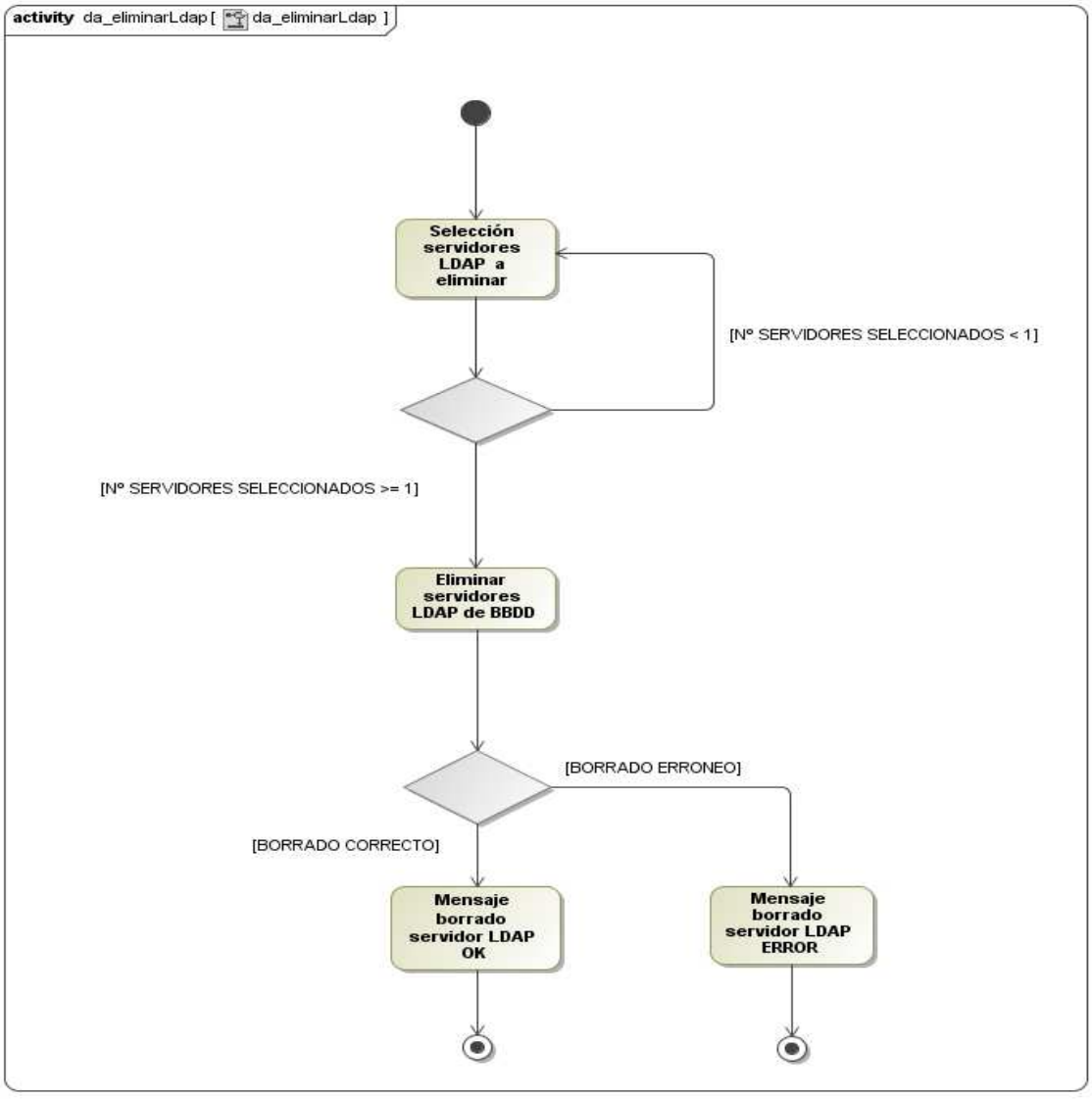


Figura 13. Diagrama de Actividad "Eliminar LDAP".

5.1.3.4 Diagrama de Actividad - "Búsqueda en Ldap"

Este caso de uso (mirar apartado *5.1.2*), sigue seguirá la siguiente sucesión de actividades.

Cuando el usuario termina de cumplimentar el formulario de búsqueda en un determinado servidor Ldap, y confirma la petición de la misma, la aplicación comprobará en primer lugar, que todos los campos de dicho formulario estén cumplimentados. Si hay campos con error, se mostrará un dialogo con el mensaje de error a el usuario, y se volverá a mostrar la pantalla de búsqueda para que vuelva a cumplimentar todos los campos correctamente. Si los datos para la petición de búsqueda introducidos por el usuario son correctos, se envía dicha petición al servidor Ldap y se espera la respuesta del mismo. Una vez recibida la respuesta del servidor Ldap, se comprueba que esta sea correcta, y si es así, se procede a listar los resultados obtenidos de la misma, en caso contrario se muestra un dialogo a el usuario informando del error sucedido.

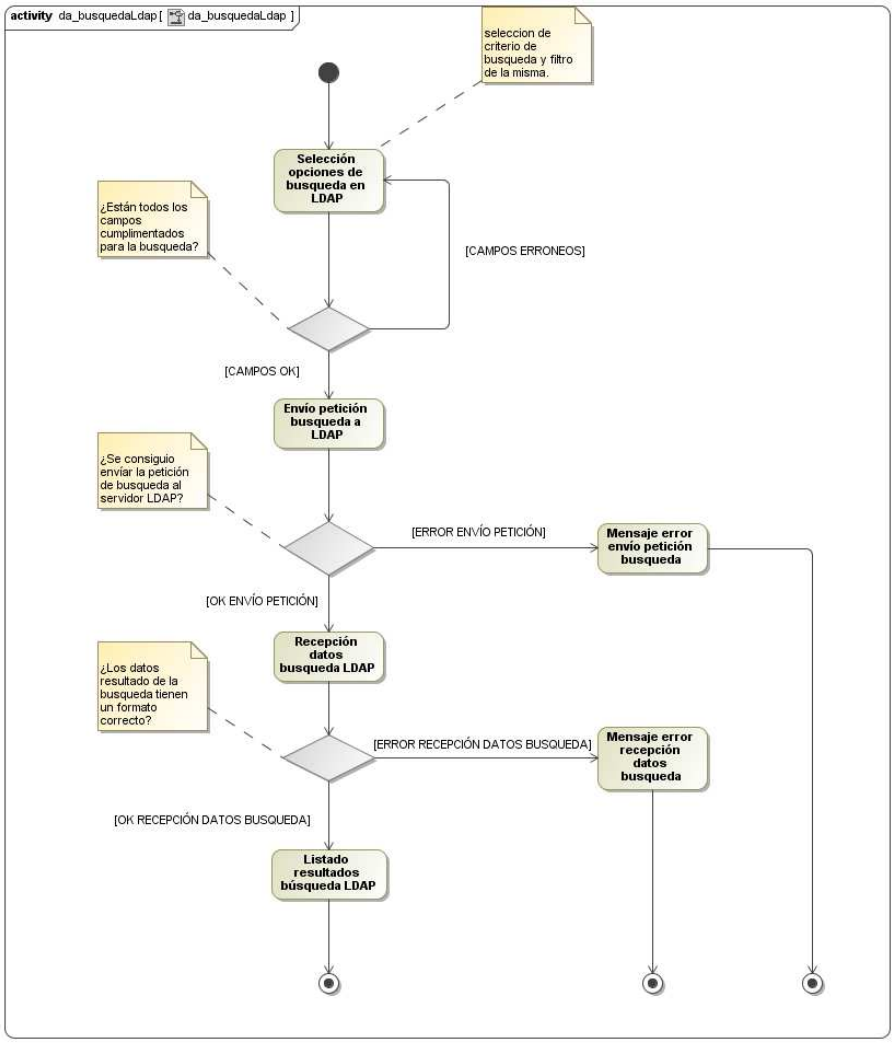


Figura 14. Diagrama de Actividad "Búsqueda en LDAP".

5.1.3.5 Diagrama de Actividad – "Visualizar entrada Ldap"

Este caso de uso (mirar apartado *5.1.2*) se activa cuando el usuario selecciona una determinada entrada de las obtenidas como resultado de una búsqueda en un determinado servidor Ldap. Al seleccionar dicha entrada del listado de resultados, en primer lugar se muestran en la pantalla de detalle de entrada Ldap los atributos denominados "*obligatorios*" (siempre aparecerán en este detalle) y a posteriori la aplicación comprobará si tiene o no el atributo teléfono (puede mostrarse o no), si lo tiene también lo mostrará en el detalle de la entrada bajo los atributos "*obligatorios*".

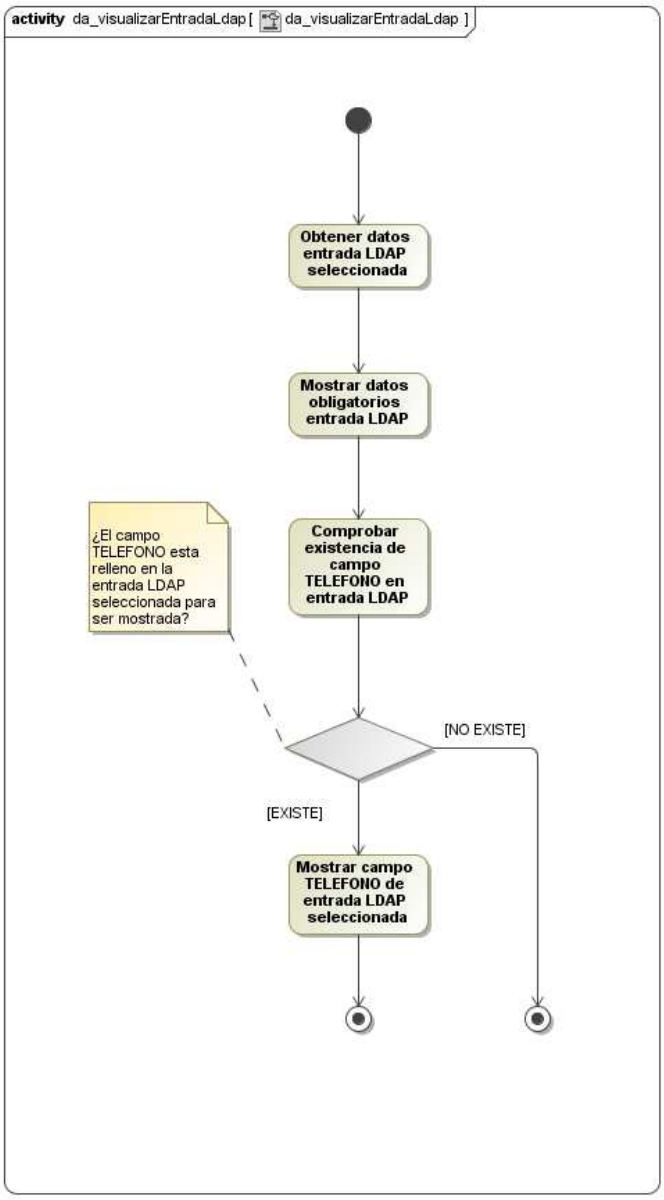


Figura 15. Diagrama de Actividad "Visualizar entrada LDAP".

5.1.3.6 Diagrama de Actividad – "Añadir atributo a entrada Ldap"

Este caso de uso (mirar apartado *5.1.2*) consistirá en la adición de atributos al detalle de la entrada, que permitirá visualizar al usuario atributos de la entrada Ldap que no se mostraron inicialmente. Una vez el usuario ha seleccionado el atributo que desea añadir al detalle, la aplicación comprueba que dicho atributo no se encuentre ya listado en el detalle, y en tal caso lo añadirá al mismo. Si por el contrario, el atributo ya se encontrase listado en el detalle de la entrada Ldap, se mostrará un dialogo con el mensaje de error al usuario.

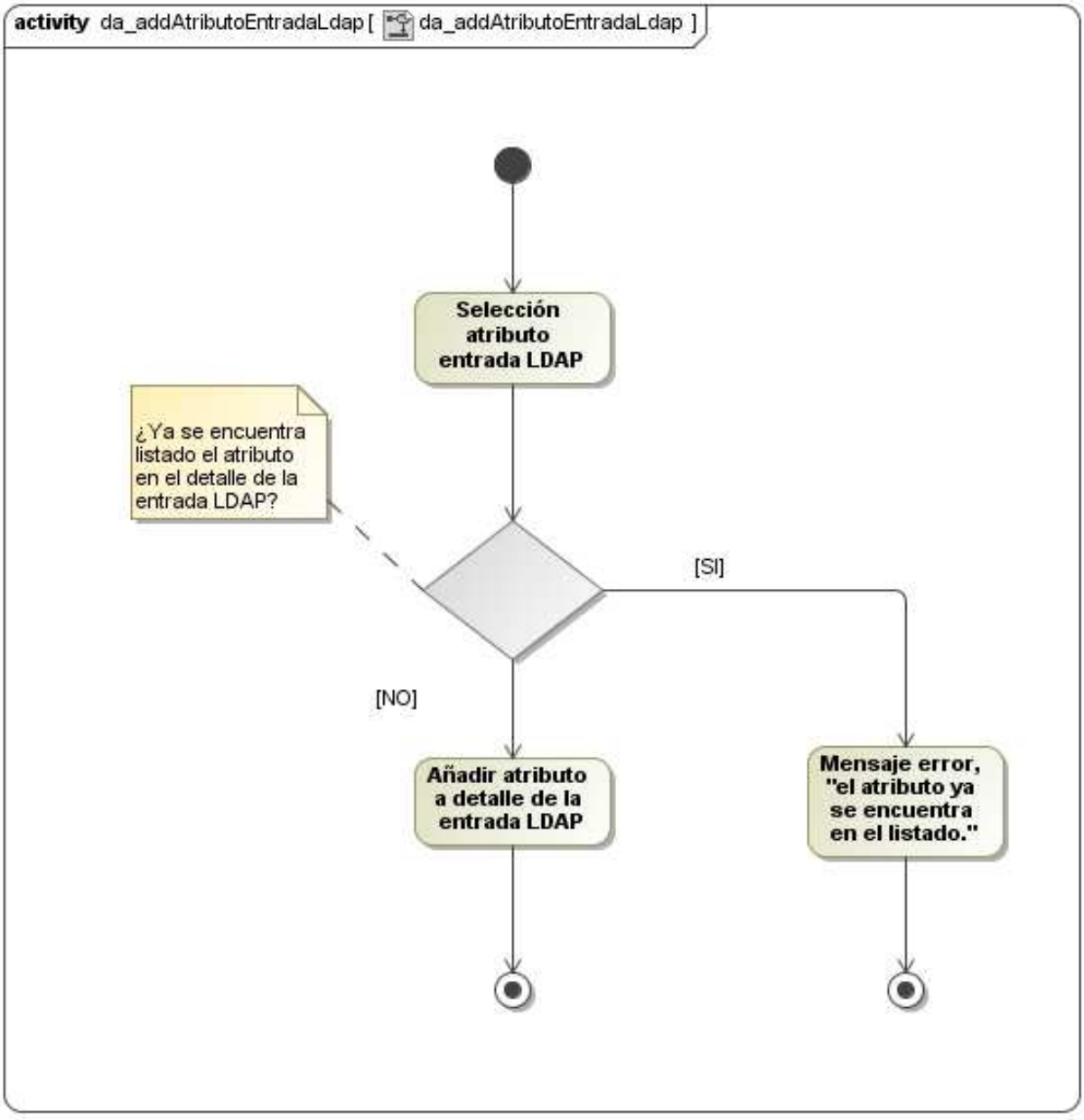


Figura 16. Diagrama de Actividad "Añadir atributo a entrada LDAP".

5.1.3.7 Diagrama de Actividad – "Añadir entrada a agenda contactos"

Este caso de uso (mirar apartado *5.1.2*) consiste en la adición de los datos de una determinada entrada del Ldap a la agenda de contactos del dispositivo móvil. Cuando el usuario realiza esta operación, la aplicación comprueba en primer lugar los datos de estado de la agenda, y si realmente cabe la posibilidad de realizar el alta en la misma (por que no esta llena, por ejemplo), realiza una lectura de los datos de la entrada y añade un nuevo contacto cumplimentado con los datos de la misma.

Si se comprobase algún tipo de error en el estado de la agenda antes de añadir el nuevo contacto, se informaría a el usuario por medio de una ventana de dialogo, del error ocurrido.

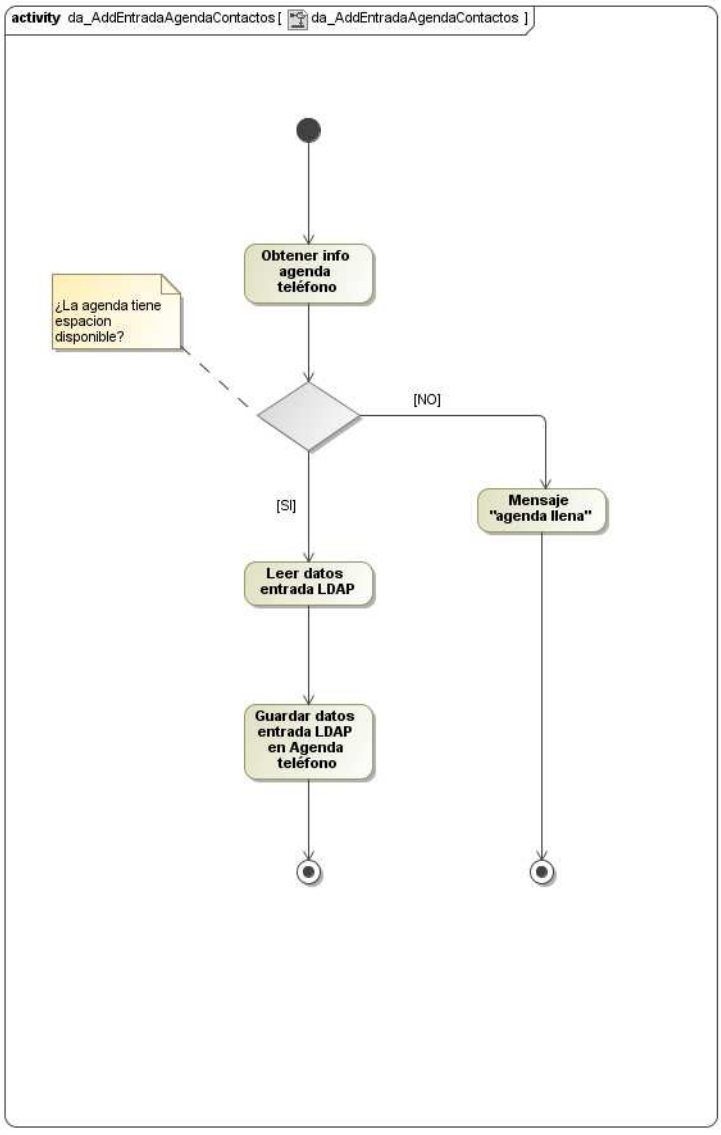


Figura 17. Diagrama de Actividad "Añadir entrada a agenda de contactos".

5.1.3.8 Diagrama de Actividad – "Llamar persona entrada"

Este diagrama de actividad ilustra el caso de uso de "llamar persona entrada" (mirar apartado *5.1.2*), en el cual se obtendrá el número de teléfono de la entrada Ldap que este visualizando en ese momento el usuario por medio del detalle de la propia entrada, y cargará la pantalla de marcado del dispositivo móvil cumplimentada con dicho número de teléfono.

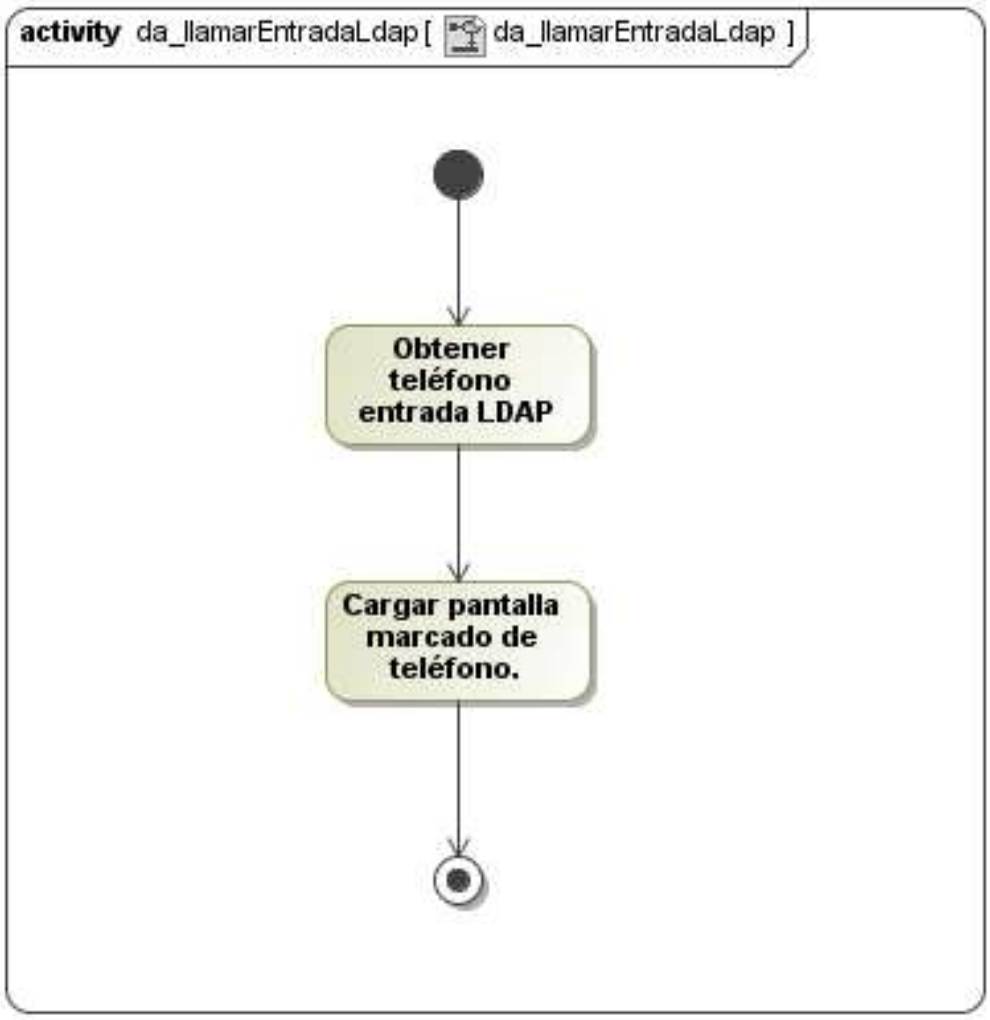


Figura 18. Diagrama de Actividad "Llamar persona entrada LDAP".

5.1.3.9 Diagrama de Actividad – "Redactar email persona entrada"

En este caso el diagrama de actividad ilustra el flujo de control que seguirá el caso de uso de "Redactar email persona entrada" (mirar apartado *5.1.2*), que consistirá en obtener la dirección de email correspondiente a la entrada que esté visualizando el usuario desde el detalle de la misma y comprobará que este no sea vacío o nulo, en cuyo caso procederá a cargar la pantalla de redacción de email del dispositivo móvil. En caso contrario, mostrará una ventana de dialogo con el siguiente mensaje "La dirección email está vacía".

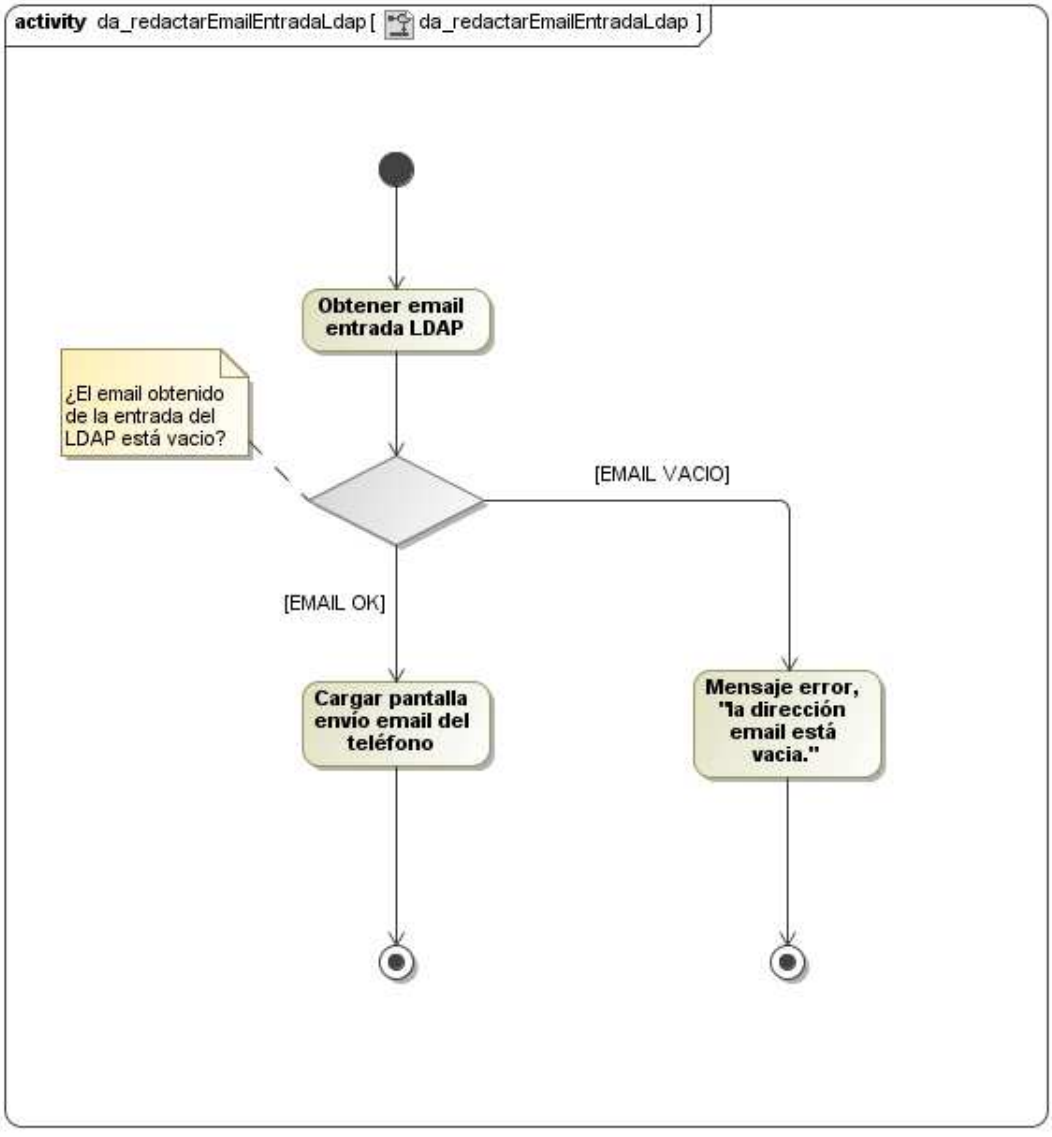


Figura 19. Diagrama de Actividad "Redactar email persona entrada LDAP".

5.1.3.10 Diagrama de Actividad – "Posicionar persona entrada"

El siguiente diagrama de actividad es el correspondiente al caso de uso "Posicionar persona entrada" (mirar apartado *5.1.2*) y consistirá en el geoposicionamiento para el atributo dirección de la entrada que este visualizando el usuario desde la pantalla de detalle de entrada Ldap.

El proceso constará de los siguientes pasos, en primer lugar se obtendrá la dirección de dicha entrada Ldap y se comprobará si tiene un formato de dirección correcto, en segundo lugar se enviará la petición de búsqueda de dicha dirección al servidor de posicionamiento y se esperará una respuesta del mismo. Si la recepción de los datos de posicionamiento es correcta, se procederá a ubicar las coordenadas para dicha dirección en la vista de tipo mapa, en caso contrario se mostrará el mensaje de error al usuario mediante una ventana de dialogo.

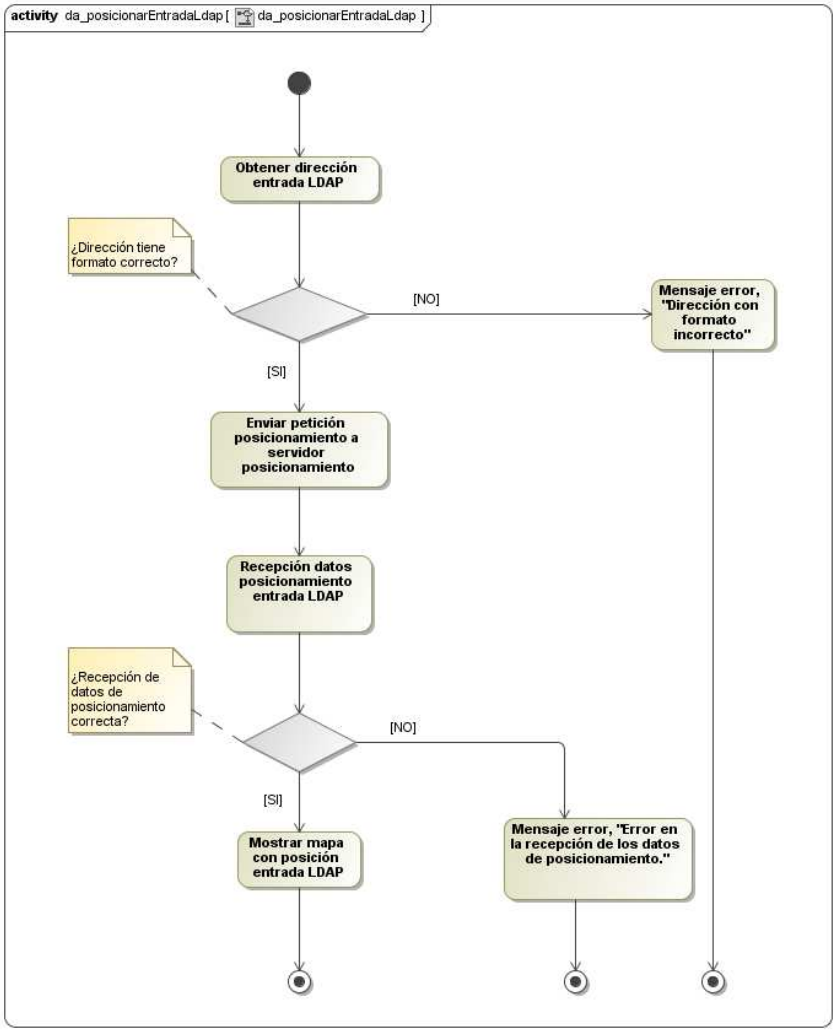


Figura 20. Diagrama de Actividad "Posicionar persona entrada LDAP".

5.1.3.11 Diagrama de Actividad – "Buscar persona en LinkedIn"

En este caso, el diagrama de actividad corresponde al caso de uso "Buscar persona en *LinkedIn*" (mirar apartado *5.1.2*) y el proceso de ejecución que seguirá se explicará en las siguientes líneas.

En primer lugar se comprobará si el usuario se había logado previamente y había concedido los permisos necesarios a la aplicación para acceder a los datos de *LinkedIn*, en caso de no ser así, redirigirá al usuario a la pantalla de acceso de *LinkedIn* para que conceda dichos permisos, si por el contrario, si que había concedido previamente estos permisos, se procederá a obtener los atributos "*nombre*" y "*apellidos*" de la entrada que el usuario estuviese visualizando desde el detalle de entrada Ldap, y se comprobará que ninguno de ellos se encuentre con valor vacío o nulo, en cuyo caso, se procederá a enviar la petición de búsqueda de contactos que coincidan en nombre y apellido a el servidor de *LinkedIn*, en caso contrario se mostrará una ventana de dialogo indicando al usuario dicho acontecimiento.

Una vez enviada la petición de búsqueda, la aplicación quedará a la espera de la recepción de los resultados enviados por el servidor de *LinkedIn*, y si la recepción es correcta y hay coincidencias en la búsqueda, la aplicación procederá a listarlas para el usuario. En caso de no obtener ninguna coincidencia, o no haber recibido una respuesta correcta se procederá a mostrar mediante una ventana de dialogo el error al usuario. A continuación en la Figura 21 se muestra el diagrama de Actividad correspondiente a la actividad "*Buscar persona en LinkedIn*".

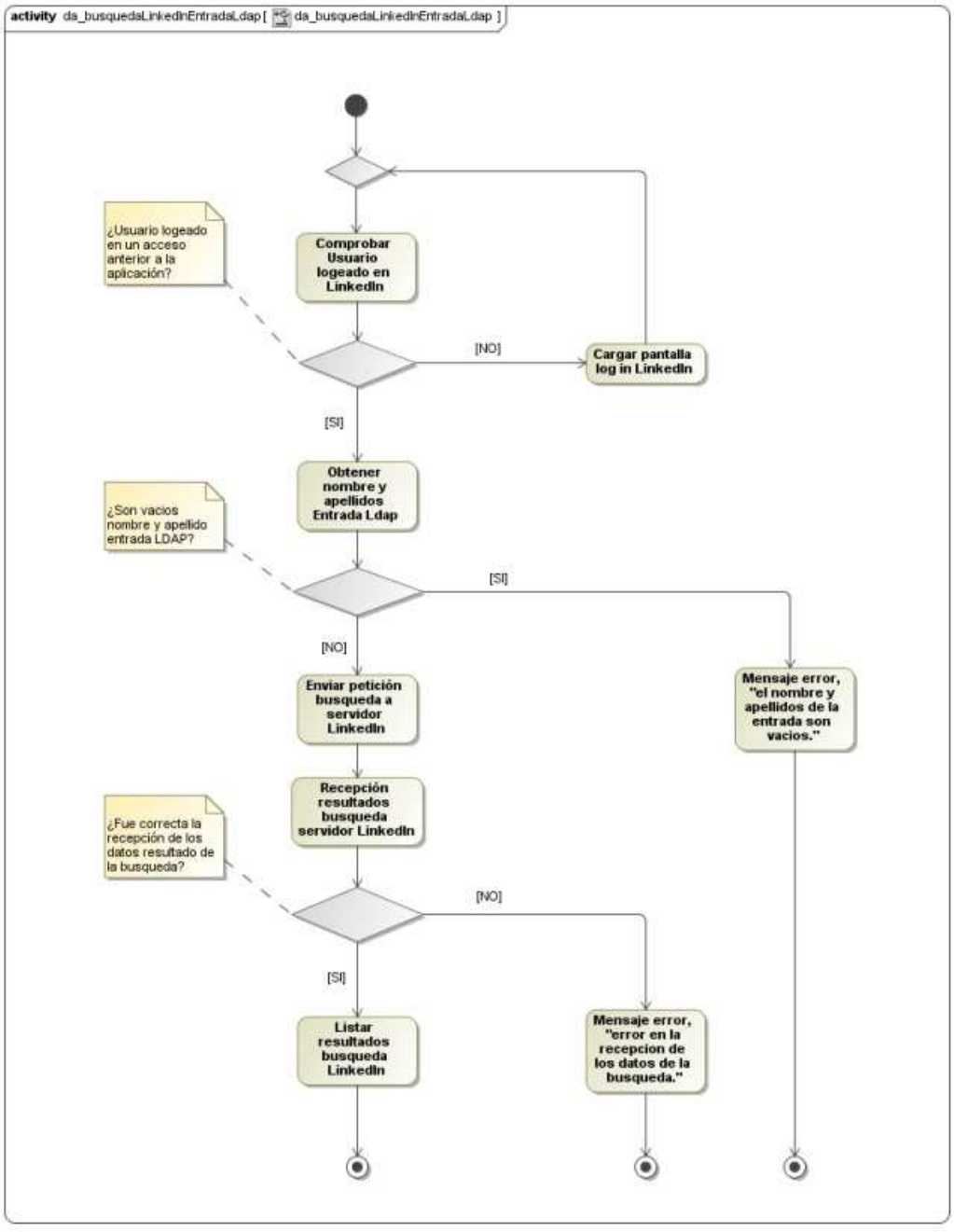


Figura 21. Diagrama de Actividad "Buscar persona en LinkedIn".

5.1.3.12 Diagrama de Actividad – "Log out LinkedIn"

Este diagrama de actividad representa el caso de uso "Log out LinkedIn" (mirar apartado *5.1.2*) consistente en la acción de desvincular la aplicación de LinkedIn. Esta acción implicará la revocación de todos los permisos que el usuario hubiese concedido a la misma previamente. La aplicación solo realizará la comprobación de si la aplicación se encontraba ya logada previamente o no. En caso de no estar logada con anterioridad, simplemente mostrará una ventana de dialogo con un mensaje indicando al usuario dicha situación.

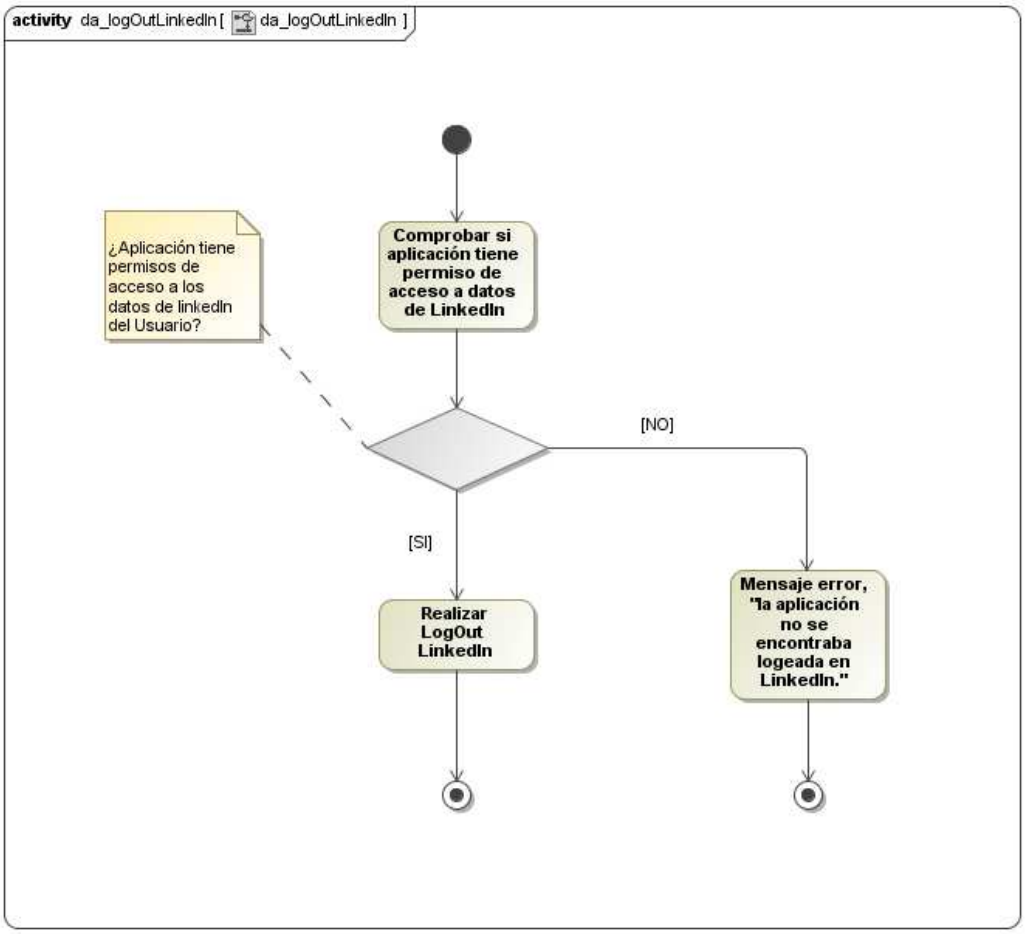


Figura 22. Diagrama de Actividad "Log out LinkedIn".

5.1.4 Arquitectura

A continuación se va a proceder a realizar una descripción del diseño a más alto nivel de la estructura del sistema mediante su diagrama Arquitectura del sistema y cada uno de los elementos que componen la misma.

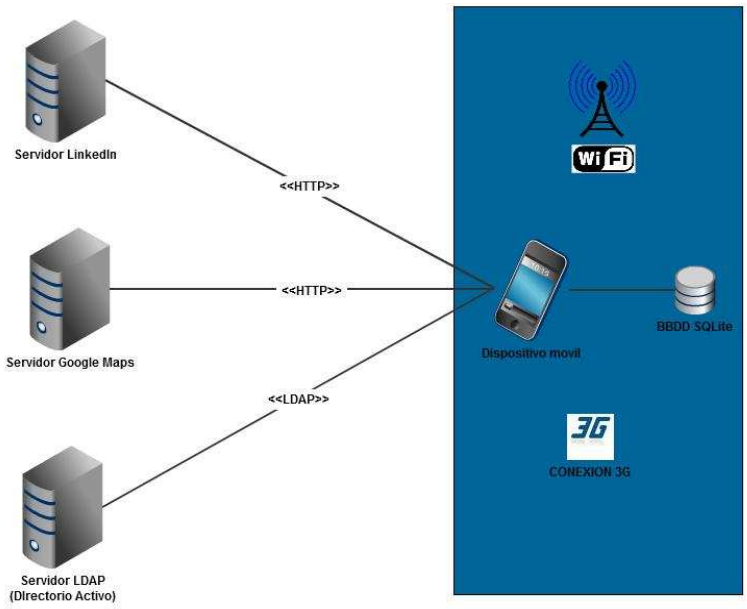


Figura 23. Arquitectura del sistema.

A continuación se describe cada un de los elementos que componen la arquitectura del sistema:

- Servidor *Ldap*:
  Servidor LDAP sobre el que realizaremos consultas bajo el protocolo de aplicación LDAP desde nuestro dispositivo móvil. Las conexiones con dichos servidores se realizarán sobre el protocolo de transporte TCP directamente. Este nos permitirá crear conexiones con el mismo de manera anónima (sobre el puerto 389 de la maquina servidora) o de manera segura mediante SSL o TLS (sobre los puertos 389 en el caso de TLS, aunque una vez establecida la comunicación cambie al puerto 636 y 636 en el caso de SSL) que cifrarán las comunicaciones entre el cliente y el servidor LDAP.
- Servidor *LinkedIn*:
  Servidor de *LinkedIn* sobre el que realizaremos consultas bajo la API REST que proporciona *LinkedIn*. Para poder realizar estas consultas, el usuario se tendrá que autenticar previamente bajo el método de autenticación *OAuth*. Todo este procedimiento se describe más detalladamente en el apartado *5.2.10.*

- Servidor *Google Maps*:
  Servidor de Google que proveerá a la aplicación de los servicios de posicionamiento sobre la entrada que hayamos seleccionado en la búsqueda realizada sobre el servidor LDAP. La aplicación utiliza la API de *Google Maps*, y esta en función de los métodos

que invoquemos, llamará a uno de los servicios de los que dispone. Sobre todo para las funcionalidades de posicionamiento y zoom sobre los *MapView* de *Android*, todas estas funcionalidades se detallará más en el apartado 3.2 de este documento.

- Dispositivo móvil:
  Dispositivo móvil con una versión de Google *Android* superior a la 1.5 "*CupCake*" (nivel de API 3). La versión objetivo de la aplicación será la 2.2 "*Froyo*" (nivel de API 8).
  También deberá contener una instalación de la aplicación.
- BBDD móvil:
  Base de datos *SQLite* compatible con la plataforma *Android* en la cual se guardará toda la información referente a las configuraciones de servidores LDAP introducidas por el usuario en nuestra aplicación. La descripción de dicha base de datos se encuentra de forma más detallada en el apartado 3.1.5.

5.1.5 Diagrama de Clases

5.1.5.1 Diagrama de Paquetes

Mediante el diagrama de paquetes se pretende aclarar la manera en que se estructurarán todas las clases del sistema, y las relaciones que habrá entre unos paquetes y otros.

En la figura 24 se muestra la estructura que se seguirá durante la implementación de la aplicación.

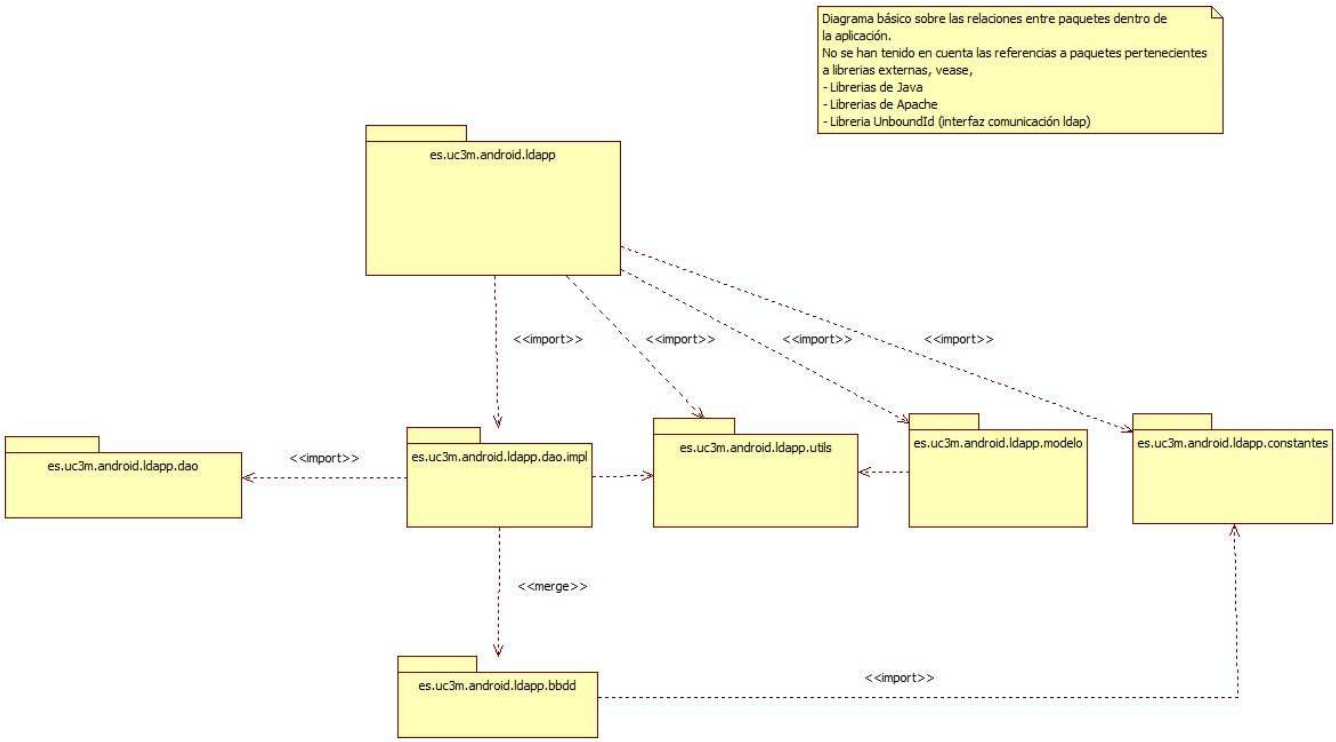


Figura 24. Diagrama de Paquetes.

5.1.5.2 Diagramas de Clases

El modelo de clases se encuentra formado por todos los atributos, métodos y la visibilidad de los mismos, así como por las relaciones entre dichas clases. El objetivo de representar mediante este diagrama todos estos elementos, es obtener un mayor nivel de detalle, sobre lo que será la aplicación, y de esta manera facilitar su futura implementación.

En este apartado se mostrarán al lector todas las clases que conforman cada uno de los paquetes representados en la Figura 23 correspondiente al diagrama de paquetes de la aplicación.

5.1.5.2.1 Paquete "es.uc3m.android.ldapp"

Este es el principal paquete de la aplicación, y desde él, se hace referencia al resto de paquetes de la misma. A continuación se describe la finalidad de cada una de las clases contenidas en este paquete.

- *ListadoServidores*:
  Clase que extiende de "*ListActivity*" y que representa la pantalla encargada de listar todas las configuraciones de servidores Ldap introducidas hasta el momento.
- *Búsqueda*:
  Clase que proporciona la funcionalidad de búsqueda de manera asíncrona.
- *EditServidor*:
  Clase que implementa los métodos que se ejecutarán tras la edición de una determinada configuración de servidor Ldap de manera asíncrona.
- *SaveServidor*:
  Clase que contendrá los métodos encargados de guardar una nueva configuración de servidor Ldap de manera asíncrona.
- *TestServidor*:
  Clase que contendrá los métodos encargados de realizar un testeo de la conexión con configuración de servidor Ldap introducida por el usuario de manera asíncrona.
- *SimpleCursorAdapterListadoServidoresAdapter*:
  Clase encargada de listar cada instancia de servidor Ldap introducida previamente por el usuario en el listado de servidores Ldap disponibles.
- *BusquedaLdap*:
  Clase que representa la pantalla de búsqueda en un determinado servidor Ldap.

- *DetalleServidor*:
  Clase que representa la pantalla de detalle para la configuración de servidor Ldap. Esta clase se utilizará tanto para la inserción como para la edición de una configuración de servidor Ldap.
- *SplashScreen*:
  Clase que representa la pantalla de presentación de la aplicación.
- *ListadoResultados*:
  Clase que representa la pantalla que lista los resultados de una determinada búsqueda sobre un determinado servidor Ldap.
- *LdapObjectAdapter*:
  Clase encargada de listar cada ítem del listado de resultados de una búsqueda sobre un determinado Ldap. La clase anteriormente mencionada "*ListadoResultados*" la instanciará para mostrar todos los resultados de la búsqueda al usuario.
- *DetalleEntradaLdap*:
  Clase encargada de mostrar el detalle de la entrada seleccionada por el usuario del listado de resultados de una búsqueda sobre un determinado servidor Ldap.
- *MapaActivity*:
  Clase encargada de mostrar el mapa sobre el que se posicionará la dirección de una determinada entrada seleccionada por el usuario del listado de resultados de una búsqueda en un servidor Ldap.
- *MapOverlay*:
  Clase auxiliar utilizada por la clase "*MapaActivity*", y encargada de mostrar la capa superpuesta a la pantalla representada por la clase "*MapaActivity*" y que contendrá el punto exacto de la dirección de una entrada seleccionada previamente por el usuario.
- *PopUp*:
  Clase que representa un popUp, que se utilizará para mostrar mensajes al usuario.
- *LinkedInActivity*:
  Clase que realiza la búsqueda por nombre y apellidos de la entrada seleccionada por el usuario sobre la red social *LinkedIn* y pinta el listado de resultados obtenidos de la búsqueda. También implementará las funcionalidades de paginación del listado de resultados.
- *UpdateAdapter*:
  Clase encargada de mostrar cada uno de los ítems resultado de la búsqueda en linkedIn. Esta clase será instanciada por la clase "*LinkedInActivity*".
- *LinkedInProviderActivity*:
  Clase encargada de verificar que el usuario que va a realizar la búsqueda en *LinkedIn* a dado su consentimiento a la aplicación llamante para tener acceso a sus datos de *LinkedIn*.
- *BuscarContactosLinkedIn*:

Clase instanciada por la clase "*LinkedInActivity*" encargada de hacer la petición de búsqueda a *LinkedIn,* de manera asíncrona.

- *LinkedInProvider*:
  Clase que contiene todos los métodos que implementan la funcionalidad de autenticación y de petición y recepción de los datos de *LinkedIn.* Esta clase será instanciada por la clase "*LinkedInActivity*", para realizar la verificación de que el usuario estaba autenticado previamente en *LinkedIn,* y también por la clase "*BuscarContactosLinkedIn*" para realizar la petición de búsqueda sobre *LinkedIn* y la recepción de los resultados de la misma.

El diagrama de clases correspondiente a este paquete se muestra en las Figuras 26 y 27.

5.1.5.2.2 Paquete "es.uc3m.android.ldapp.bbdd"

Este paquete contiene todas las clases de mantenimiento de la BBDD del dispositivo móvil, encargadas de generar las tablas iniciales cuando se instala la aplicación en el dispositivo y también de realizar las actualizaciones sobre la BBDD si es que las hubiera. A continuación se describe cada clase más detalladamente.

- *LdapData*:
  En la primera carga de la aplicación "*Ldapp*", generará su base de datos en el dispositivo móvil, y si se realiza alguna actualización de la aplicación y también de la base de datos, se encargará de realizar la actualización de la misma.

En la Figura 25 Se muestra el diagrama de clases de dicho paquete.

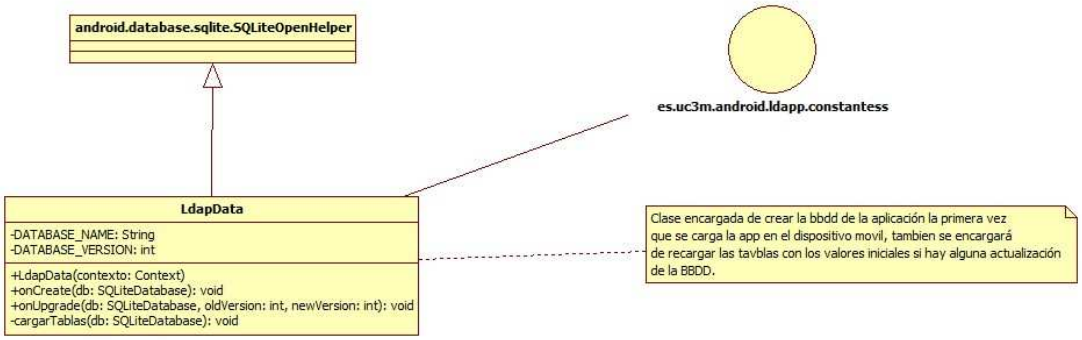


Figura 25. Diagrama de Clases "es.uc3m.android.ldapp.bbdd".

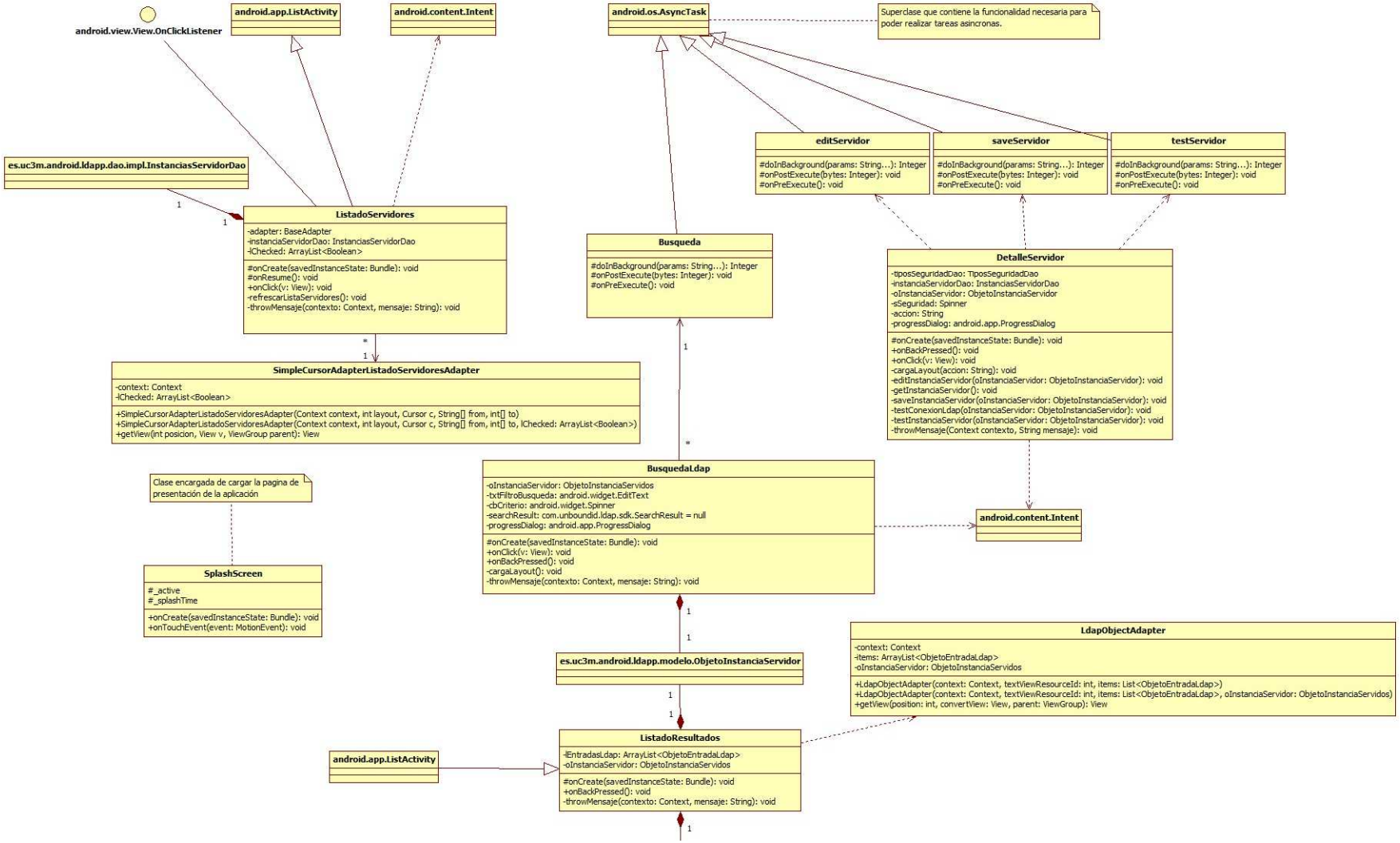


Figura 26. Diagrama de Clases “es.uc3m.android.ldapp".

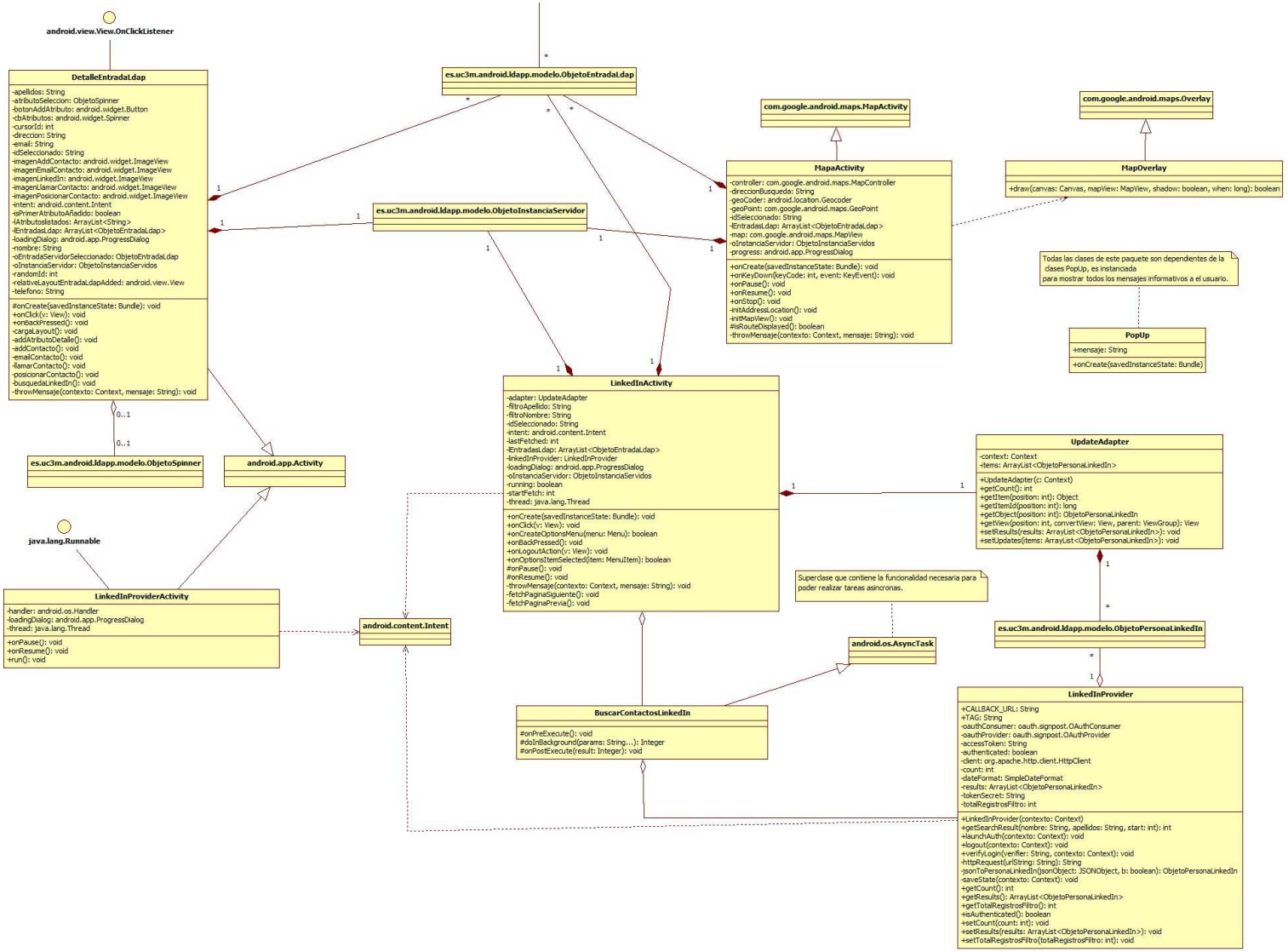


Figura 27. Diagrama de Clases “es.uc3m.android.ldapp".

5.1.5.2.3 Paquete “es.uc3m.android.ldapp.constantes”

Este paquete contendrá todas las constantes de carácter global a la aplicación. A continuación se muestra en la Figura 28 el diagrama de clases para este paquete.

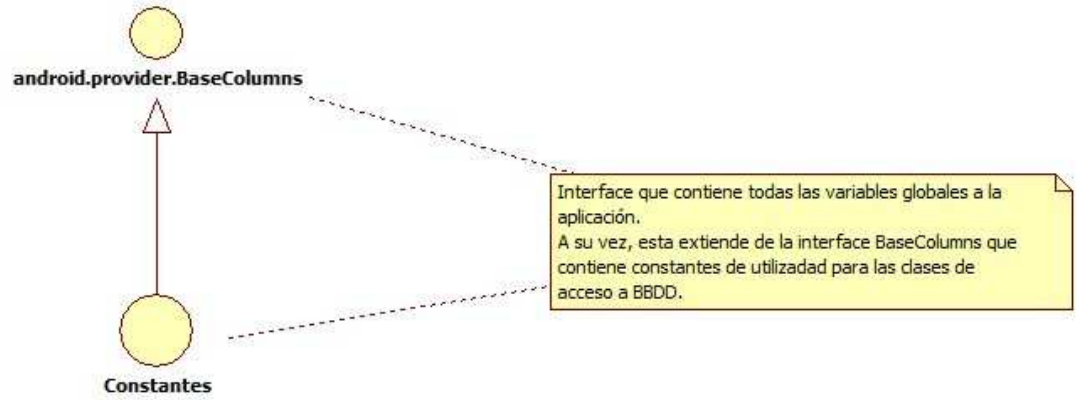


Figura 28. Diagrama de Clases “es.uc3m.android.ldapp.constantes”.

5.1.5.2.4 Paquete “es.uc3m.android.ldapp.dao”

Este paquete contendrá todas las interfaces que definirán la operatividad de cada uno de las clases DAO de la aplicación, es decir, contendrán todos los métodos comunes a las clases que implementen dicho interface.

A continuación se muestra en la Figura 29 el diagrama de clases para este paquete.

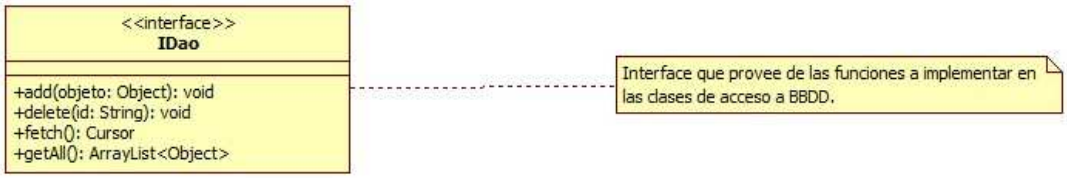


Figura 29. Diagrama de Clases “es.uc3m.android.ldapp.dao”.

5.1.5.2.5 Paquete “es.uc3m.android.ldapp.dao.impl”

Este paquete contendrá cada una de las clases DAO, es decir, las encargadas del acceso a datos. Realizarán las operaciones de inserción, edición, borrado y consulta sobre la BBDD e implementará alguna de las interfaces incluidas en el paquete “es.uc3m.android.dao”. A continuación se describen más en profundidad, cada una de las clases contenidas en este paquete.

- *InstanciasServidorDao*:
  Clase de acceso a datos, que implementa todos los métodos de manipulación sobre los datos de la tabla “*instancias_servidor*”, es decir, la inserción, modificación y borrado de los mismos, asi como otros métodos como “*getAll()*” que obtiene para la obtención de los mismos mediante cursores (método “*fetch*”)
- *TiposSeguridadDao*:
  Al igual que la clase expuesta previamente, esta contiene todos los métodos de manipulación de datos, pero de la tabla “*tipos_seguridad*”, y también implementará todos los métodos de la interfaz “*IDao*”.

Tanto la clase “*InstanciasServidorDao*” como la clase “*TiposSeguridadDao*” extenderán de la clase “*LdapData*”, que será la encargada de proveer a ambas de una conexión a la base de datos sobre la que operarán. A continuación se muestra en la Figura 30 el diagrama de clases para este paquete.

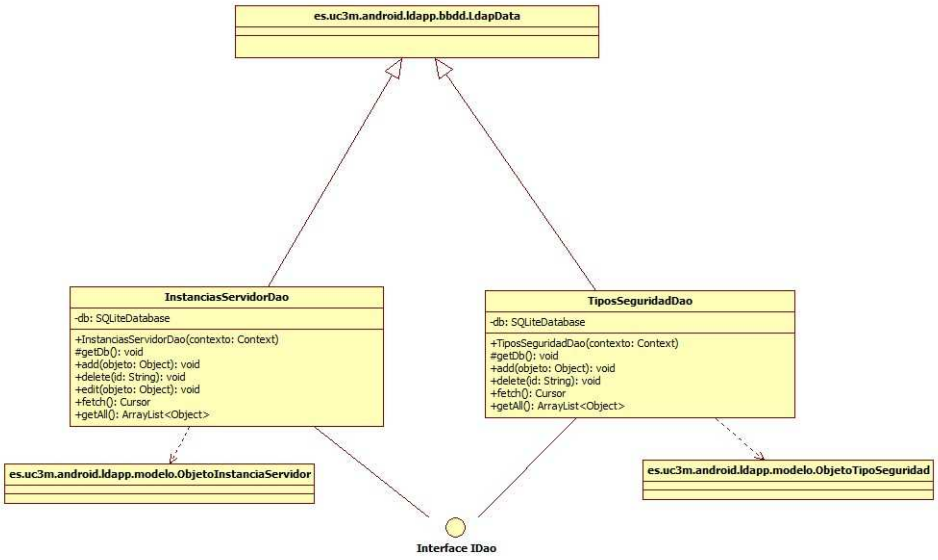


Figura 30. Diagrama de Clases “es.uc3m.android.ldapp.dao.impl”.

5.1.5.2.6 Paquete "es.uc3m.android.ldapp.utils"

Este paquete contendrá todas las clases que contengan los métodos auxiliares utilizados en la aplicación. Solo cuenta con la clase "*Utils*", que será la que englobe todos estos métodos de carácter secundario.

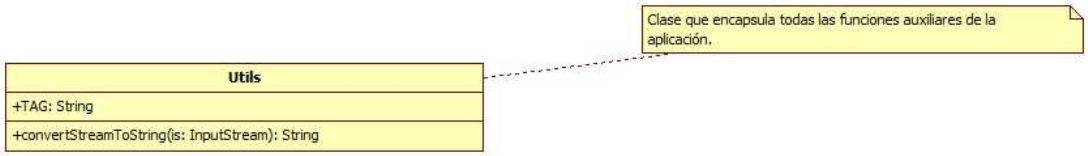


Figura 31. Diagrama de Clases "es.uc3m.android.ldapp.utils".

5.1.5.2.7 Paquete "es.uc3m.android.ldapp.modelo"

Este paquete contendrá todas las clases "entidad" que componen el sistema, es decir, las clases que definirán el modelo de la aplicación o lo que es lo mismo, la base lógica de la misma. A continuación se detallan más en profundidad las clases que contiene este paquete.

- *ObjetoAtributoLdap:*
  Se instanciará este objeto por cada atributo que contenga una determinada entrada recuperada del servidor Ldap.
- *ObjetoEntradaLdap:*
  Se instanciará este objeto por cada entrada recuperada del servidor Ldap, y contendrá un "*id*" identificador univoco de la entrada, y una lista de objetos de la clase "*ObjetoAtributoLdap*", con todos los atributos que contiene dicha entrada.
- *ObjetoInstanciaServidor:*
  Esta clase representa, un objeto instancia servidor, es decir, una configuración de servidor Ldap, y se instanciará tantas veces como nuevas configuraciones genere el usuario, para ser persistidas en base de datos, o para ser recuperadas de la misma.
- *ObjetoPersonaLinkedIn:*
  Esta clase representa un registro de *persona* obtenido como resultado de una petición de búsqueda sobre la red social *LinkedIn.* Sus atributos serán los datos que se mostrarán por cada *persona* obtenida de dicha búsqueda en el listado de resultados de la misma.
- *ObjetoSpinner:*
  En la jerga *Android,* un Spinner representa un combo, y esta clase en particular, hace referencia a cada elemento listado en un determinado

combo. Así pues, se creará una nueva instancia de este objeto por cada elemento que se quiera cargar en un determinado *Spinner*.

- *ObjetoTipoSeguridad:*
  Este objeto representa un determinado tipo de seguridad y se instanciará por cada tipo de seguridad que queramos cargar en el *Spinner* (combo), de la pantalla de "*Detalle de Configuración de Servidor Ldap*", que se detalla más adelante, en el apartado *5.2.4.*

A continuación se muestra en la Figura 32 el diagrama de clases para este paquete.

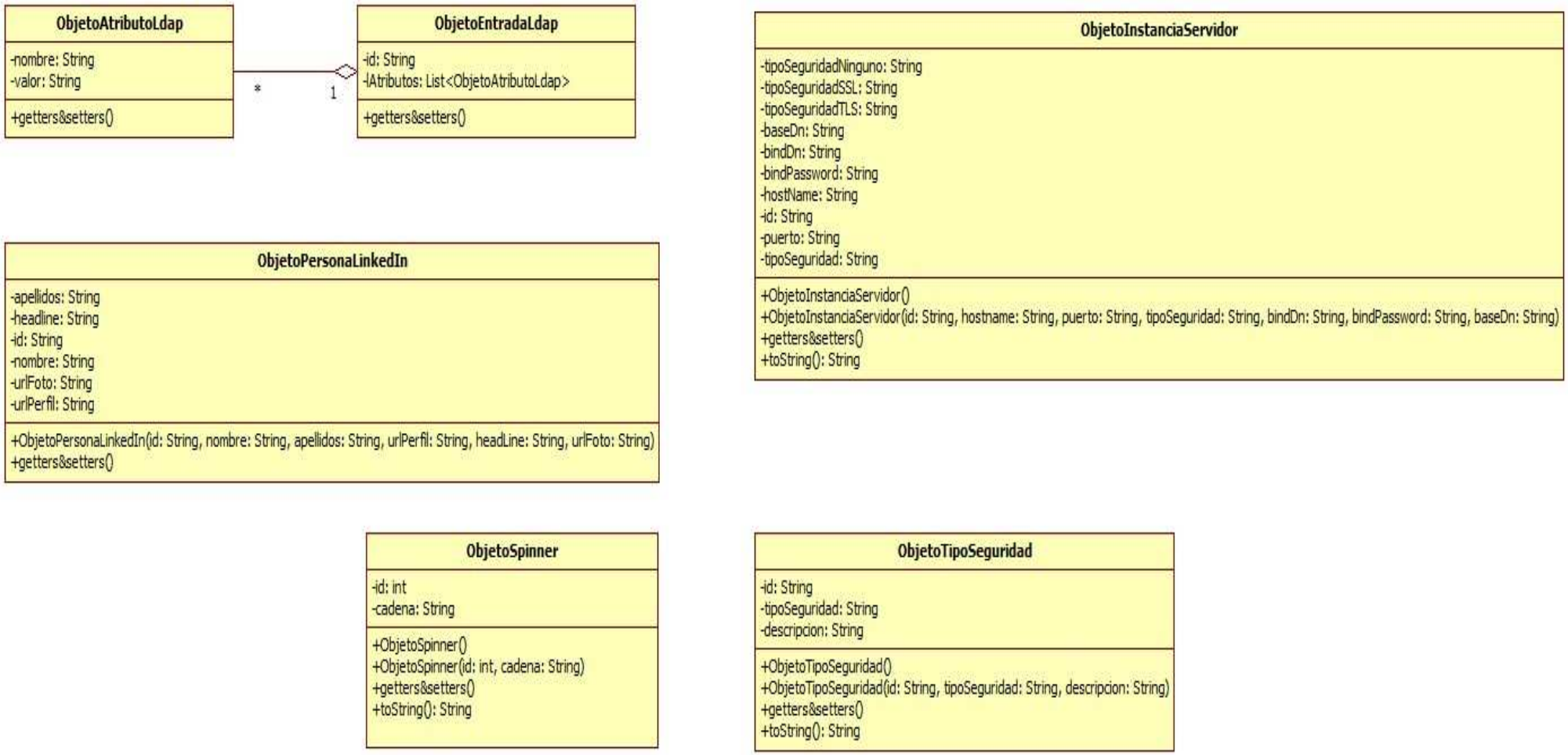


Figura 32. Diagrama de Clases "es.uc3m.android.ldapp.modelo".

5.1.6 Diagrama Entidad-Relación

La aplicación también implementa una capa de bases de datos. La base de datos que integrará el sistema almacenará toda la información referente a las configuraciones de servidores LDAP generadas por el usuario en la aplicación así como también los tipos de seguridad que implementará cada una de ellas. A continuación se muestra en la Figura 33 el diagrama Entidad-Relación para la misma.

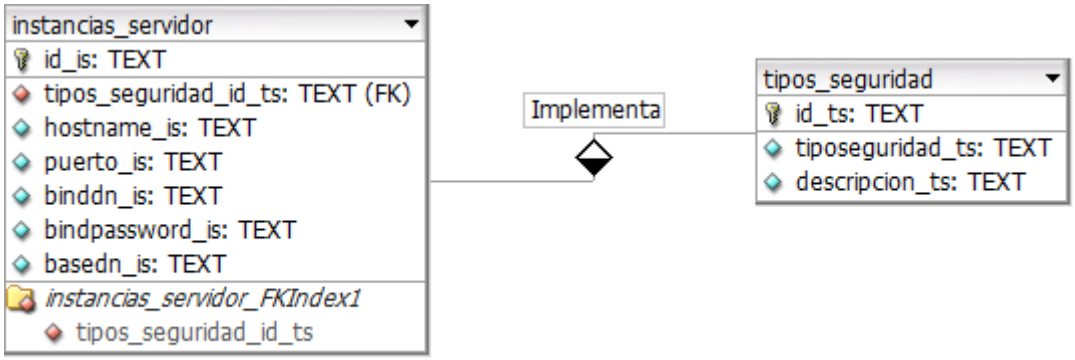


Figura 33. Diagrama Entidad - Relación.

5.2 Implementación

Una vez detallado todo lo referente al análisis y diseño de la aplicación, a continuación se expondrán de manera pormenorizada cada uno de los detalles en cuanto a la implementación se refiere.

La definición de "*implementación*" según la Real Academia Española de la Lengua, es la acción de "implementar", es decir, *"de poner en funcionamiento, aplicar métodos, medidas, etc., para llevar algo a cabo"*, y en eso exactamente consistirá este apartado, en definir de una forma más concreta, como se ha llevado acabo la aplicación tomando como referencia, eso si, el análisis y diseño previo de la misma.

A continuación se expone una descripción de cada uno de los módulos implementados para la aplicación.

5.2.1 Lenguaje de programación y entorno de desarrollo

Todas las aplicaciones implementadas para la plataforma *Android* utilizan el lenguaje de programación "*Java*" y como no puede ser de otra manera la que se expone a continuación también. Para la implementación de la aplicación "*Ldapp*" se hizo uso de la JDK 6 ("*Java Development Kit*", kit de desarrollo java) sobre el JRE ("*Java Runtime Enviroment*", Entorno de ejecución de Java) de Java.

El entorno de desarrollo utilizado durante todo el proceso de implementación de la aplicación ha sido "Eclipse" en su versión "*Helios Service Release 2*".
Para la implementación de aplicaciones *Android* fue necesaria la descarga e instalación de su SDK ("*Software Development Kit*"), consistente en una serie de herramientas para el desarrollo de aplicaciones para plataformas *Android*.

Una vez instalados, tanto Eclipse, como el SDK6 de *Android*, se instalo el complemento para *Eclipse*, *ADT* ("*Android Development Toolkit*"), que permite al desarrollador funcionalidades tales como probar su aplicación en un "*dispositivo virtual Android*" (AVD, "*Android Virtual Device*") o lo que es lo mismo, el emulador de un dispositivo móvil real. Este complemento nos permitirá configurar un dispositivo virtual, con las características que creamos adecuadas para el testeo de la aplicación.

Como versión de la plataforma *Android* "objetivo" de la aplicación "Ldapp", entendiendo como "objetivo", la versión para la que la aplicación tendrá un rendimiento óptimo, se ha optado por la versión 2.2 de *Android*, denominada "*Froyo*", ya que en la actualidad es la más popular y la que actualmente están instalando la mayoría de las compañías de telefonía en los terminales que tienen a la venta. No obstante, la aplicación es compatible con cualquiera de las demás versiones de *Android*.

5.2.2 Base de datos

La base de datos utilizada en la aplicación, será "*SQLite*" esta, está especialmente diseñada para ser utilizada en dispositivos con restricciones de memoria, como por ejemplo dispositivos móviles, en nuestro caso bajo la plataforma *Android*. Las bases de datos "*SQLite*", se integran con la aplicación que hará uso de ellas para evitar la latencia de acceso a la misma como ocurre en los sistemas de gestión de bases de datos cliente-servidor, en los que la base de datos es un proceso independiente con el que se comunica el programa principal.
En resumen, las bases de datos "*SQLite*" son ideales para ser utilizadas en plataformas *Android* por 3 motivos:

- Son gratuitas.
- De tamaño reducido (la versión actual ocupa cerca de 150 KB).
- No requiere configuración o administración.

5.2.3 Pantalla "Presentación Aplicación Ldapp"

Esta pantalla es la encargada de presentar la aplicación, solo se mostrará durante un periodo de 5 segundos (5000 milisegundos) al inicio de la misma, para informar al usuario de la aplicación que esta a punto de ejecutarse, y la versión de la misma, en la jerga *Android* es la denominada "*Splash screen*".

Figura 34. Pantalla de Presentación de aplicación "Ldapp".

Esta pantalla tiene como base una clase de tipo "*Activity*", es decir, extiende de ella. Desde esta clase "*Activity*" se define la interfaz de usuario de forma "declarativa"

es decir, se hace referencia a un archivo xml que es el que contiene la declaración de todos los elementos que forman la pantalla. En el caso de la aplicación "*Ldapp*" la interfaz de usuario, siempre será definida *declarativamente.*

A continuación en el Código 8 se muestra el fragmento de código de la actividad, contenedora de la funcionalidad y apariencia de esta pantalla.

```
package es.uc3m.android.ldapp;

public class SplashScreen extends Activity {

	protected boolean _active = true;
	protected int _splashTime = 9000;

	@Override
	public void onCreate(Bundle savedInstanceState)
	{
	    super.onCreate(savedInstanceState);
	    setContentView(R.layout.splash);

	    // Hilo de ejecución para mostrar splash screen
	    Thread splashTread = new Thread() {
	        @Override
	        public void run() {
	            try {
	                int waited = 0;
	                while(_active && (waited < _splashTime)) {
	                    sleep(100);
	                    if(_active) {
	                        waited += 100;
	                    }
	                }
	            } catch(InterruptedException e) {
	                // do nothing
	            } finally {
	                finish();
	                startActivity(new Intent(getApplicationContext(),
	                                ListadoServidores.class));
	                stop();
	            }
	        }
	    };
	    splashTread.start();
	}
}
```

Código 8. Activity "SplashScreen".

El método "SetContentView" será el encargado de hacer referencia al archivo xml que contiene todos los elementos de la interfaz gráfica de esta pantalla. "R.layout.splash" simplemente define la ruta al archivo xml dentro del contexto de la aplicación. En este caso hace referencia a la siguiente ruta, "*/ldapp/res/layout/splash.xml*". A continuación en el Código 9 podrá observar el archivo xml con la declaración de los elementos de la interfaz grafica de esta pantalla.

Como se puede observar en dicho código xml, la pantalla esta formada por un "*layout*" que ocupará el total del ancho y del alto de la pantalla del dispositivo móvil, tendrá orientación vertical, y tendrá como fondo la imagen "fondo_splash" (todas las imágenes de la aplicación se guardarán por defecto en la ruta "*/ldapp/res/drawable*" del proyecto, y se referenciarán tal y como se muestra en el código "*@drawable/fondo_splash*").

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/fondo_splash"
    android:orientation="vertical" >
</LinearLayout>
```

Código 9. Xml interfaz gráfica Activity "SplashScreen".

Como se puede observar, en el método "*onCreate*" de la Actividad se crea un hilo de ejecución "*splashTread*" y se ejecuta tras una espera de *n* milisegundos, esta cantidad de tiempo se define en la variable global "*_splashTime*" y tras ella, se invocará la actividad "*ListadoServidores*" mediante la instanciación de un nuevo "*Intent*", como se muestra a continuación en el Código 10.

```
startActivity(new Intent(getApplicationContext(),ListadoServidores.class));
```

Código 10. Llamada Activity "ListadoServidores".

5.2.4 Pantalla "Listado de Servidores Ldap"

La pantalla que se describe a continuación, es la primera que visualizará el usuario tras la pantalla de presentación de la aplicación, en ella se listarán todas las configuraciones de servidores ldap que el mismo haya ido guardando previamente. También tendrá la posibilidad de añadir nuevas "configuraciones ldap", eliminarlas o editarlas.

Cada uno de los registros listados corresponderá a una "configuración de servidor *Ldap*", y contendrá los siguientes campos descriptivos de la misma:

- *Id:* texto identificativo del servidor *Ldap*.
- *Hostname:* hostname al que estamos accediendo.
- *BaseDn:* define la base de búsqueda dentro de la estructura de árbol del servidor *Ldap*, mediante el "*DN*" o "*Distinguished Name*", nombre identificativo para cada entrada del mismo.

La opción de "añadir configuración *Ldap*" redirigirá al usuario a la pantalla de detalle de servidor *Ldap*, en la cual tendrá que rellenar los datos para la nueva configuración y pulsar el botón de guardar. Antes de guardar en BBDD la nueva configuración, la aplicación realizará una comprobación sobre la comunicación con el servidor *Ldap* para los datos introducidos, y si la comunicación es correcta procederá a salvar los mismos en BBDD y mostrará la nueva configuración en el listado de servidores *Ldap*. En caso contrario mostrará un mensaje al usuario con el error encontrado. En la figura…. podrá observar los pantallazos de "*Listado de Servidores Ldap*" y en la figura…. de "*Detalle de Configuración de Servidor Ldap*".

La opción de "*eliminar configuración Ldap*" desde el listado de servidores *Ldap* permitirá realizar un borrado simple, de un solo registro o múltiple mediante el chequeo de las configuraciones que se desee eliminar y tras clicar el botón de "Borrar Servidor" estas serán eliminadas de la BBDD y desaparecerán del listado.

Tanto la opción de "*edición de configuración Ldap*" como la de "*búsqueda en servidor Ldap*" podrán seleccionarse cuando cliquemos encima de una de las configuraciones listadas, desde el dialogo que se muestra en el pantallazo de la izquierda de la Figura 35.

Figura 35. Pantallazos "Listado de Servidores Ldap".

Cuando se seleccione la opción de editar, se cargará la pantalla de "*Detalle de Configuración de Servidor Ldap*" con todos los campos en estado editable y rellenos con los datos de dicha configuración. Una vez el usuario termine de editar la configuración del *Ldap*, y confirme los cambios, la aplicación realizará una comprobación de la comunicación con el servidor, y en caso de obtener una respuesta correcta del mismo procederá a guardarla en BBDD, en caso contrario mostrará un dialogo informando del error ocurrido al usuario.

Si el usuario selecciona la opción de "*búsqueda en servidor Ldap*", se le redirigirá a la pantalla de búsqueda, donde podrá realizar un filtrado sobre el mismo, esta funcionalidad se detallará más en profundidad en el apartado….

Sobre la implementación de esta pantalla cabe destacar, que la clase que implementa esta pantalla, extenderá del tipo "*ListActivity*" que proveerá a la clase de la funcionalidad necesaria para listar "*items*" (por ítem se entiende cualquier tipo de vista o interfaz gráfica implementada programáticamente o declarativamente, mediante un archivo xml) mediante un objeto de la clase "*SimpleCursorAdapter*" que en nuestro caso se denominará "*SimpleCursorAdapterListadoServidoresAdapter*"y que se encargará de mostrar cada registro obtenido de la tabla "*instancias_servidor*" de la BBDD por el cursor "*cursor*" que se le pasa como parámetro, con el formato definido para el ítem dentro del adaptador.

A continuación se muestra en Código 11 la definición de la clase "*ListadoServidores*" y de la asignación del adaptador a la misma.

```
public class ListadoServidores extends ListActivity implements
OnClickListener {
	private BaseAdapter adapter;

	protected void onCreate(Bundle savedInstanceState) {

		// ...
		refrescarListaServidores();
	}

	private void refrescarListaServidores(){
		// ...
		Cursor cursor = instanciaServidorDao.fetch();
		adapter = new SimpleCursorAdapterListadoServidoresAdapter(
				this,
				R.layout.rowservidor,
				cursor,
				new String[] {"id_is","hostname_is","basedn_is"},
				new int[] { R.id.txt_id_servidor,
				R.id.txt_hostname_servidor,
				R.id.txt_basedn_servidor},
				lChecked
			);

		setListAdapter(adapter);
		// ...
	}

}
```

Código 11. ListActivity "ListadoServidores".

5.2.5 Pantalla "Detalle de Configuración de Servidor Ldap"

La pantalla de detalle de configuración, es la que utilizará el usuario para introducir los datos de las nuevas configuraciones de servidores Ldap que desee guardar en la aplicación, también será la que permitirá al usuario editar una determinada configuración en caso de que lo desee. Cuando la opción elegida sea la de edición, aparecerán todos los campos rellenos con los datos de la configuración *Ldap* que se desea editar.

Los campos que aparecen en el detalle de configuración de servidor *Ldap*, son los siguientes:

- *Id*: campo obligatorio. Nombre identificativo de la configuración *Ldap*.
- *Host name*: campo obligatorio. Nombre del host del servidor *Ldap* al que conectaremos.
- *Puerto servidor*: campo opcional. Obligatorio cuando el tipo de seguridad es "*TLS*" ó "*SSL*". Número del puerto de comunicación con el servidor.
- *Tipo de seguridad*: campo obligatorio. Podrá ser una conexión de tipo "*anónimo*", "*TLS*" ó "*SSL*".

- *Bind DN*: solo será necesario si el tipo de seguridad es del tipo TLS o SSL, y representa el DN del usuario que realiza la conexión al servidor *Ldap*.
- *Bind Password*: solo será necesario si el tipo de seguridad es del tipo TLS o SSL, y representa la contraseña definida para el usuario (Bind DN) que realiza la conexión al servidor *Ldap*.
- *Base DN*: campo obligatorio. Representa el nodo "*base*" dentro del árbol de entradas del servidor *Ldap* a partir del cual el usuario podrá realizar las operaciones que desee (sobre el base DN y todos sus nodos hijos).

Alta y edición de configuración servidor Ldap.

Tanto en el caso del alta de una nueva configuración como en el de edición de la misma, el usuario podrá realizar un testeo de la comunicación con el servidor Ldap recién configurado. No obstante, cuando el usuario clique el botón de "*Guardar Servidor LDAP*" se volverá a realizará el testeo de comunicación, y si la comunicación es correcta, y recibimos respuesta del servidor, la nueva configuración Ldap será guardada en BBDD, en caso contrario se mostrará un dialogo al usuario con el error recibido.

A continuación se muestran en la Figura 36 los pantallazos de la pantalla de detalle.

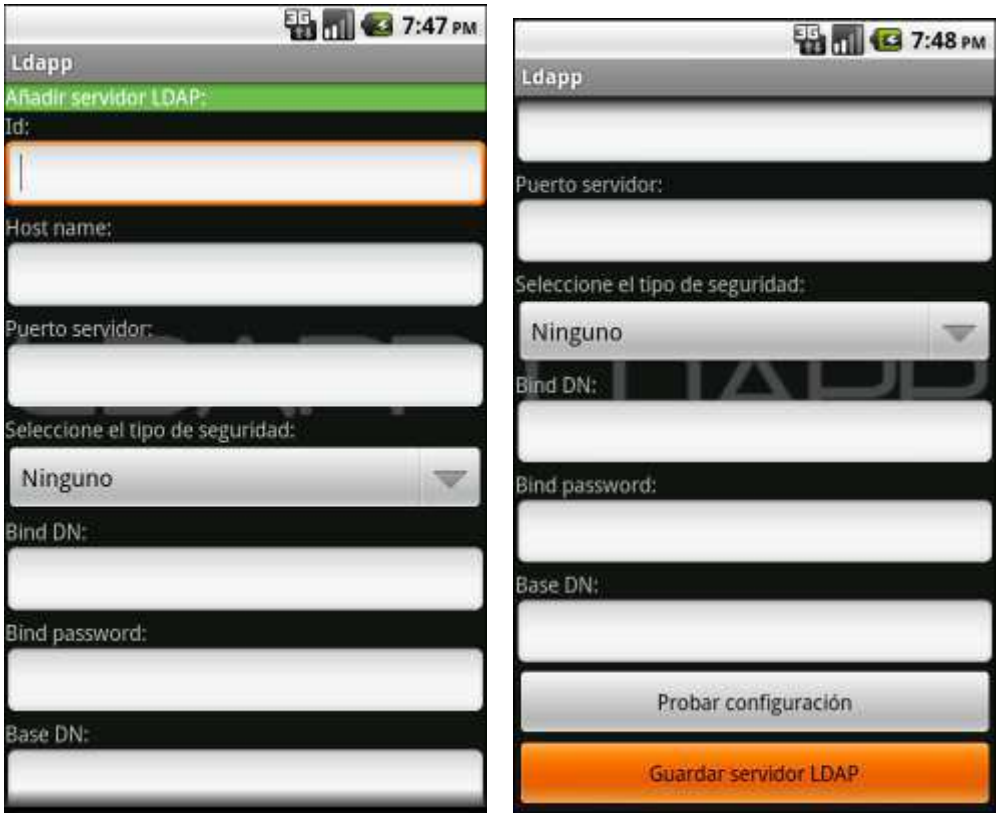


Figura 36. Pantallazos "Listado de Servidores Ldap".

Carga del combo de tipos de seguridad.

De esta pantalla cabe destacar que en cada carga de la misma, se obtendrá de BBDD un cursor con el que se procederá a la carga del combo ("*Spinner*" en Android) de tipos de seguridad. Se ha decidido la carga desde BBDD por una cuestión de dinamismo, si en algún momento los tipos de seguridad cambian, o se amplían, solo habría que añadirlos a la tabla "*tipos_seguridad*" y automáticamente se añadirán en la próxima carga de la pantalla.

A continuación se muestra en el fragmento de código, Código 12, la carga del combo de tipos de seguridad. En dicho código se puede observar que en primer lugar se comprueba si el cursor tiene algún registro y en caso afirmativo, se recorre por medio de la sentencia "*while*" añadiendo el "*id*" del tipo de seguridad del campo "*id_ts*" del cursor a cada nuevo elemento del combo, asi como el texto de tipo de seguridad del campo "*tiposeguridad_ts*". Cada nuevo elemento del combo estará representado por un objeto de la clase "ObjetoSpinner" que se rellenará con los datos de un registro del cursor y se añadirá al combo ("*Spinner*") mediante el método "*add*" del objeto "*tiposSeguridadAdapter*", el adaptador definido para el combo.

```
// ...
// Definimos Spinner
sSeguridad = (Spinner) findViewById(R.id.cb_add_seguridad);
tiposSeguridadAdapter = new ArrayAdapter<ObjetoSpinner> (
            this,
            android.R.layout.simple_spinner_item);

tiposSeguridadAdapter.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
sSeguridad.setAdapter(tiposSeguridadAdapter);
// Cargamos spinner
Cursor cursor = tiposSeguridadDao.fetch();
startManagingCursor(cursor);
if(cursor.moveToFirst()){
     do {
            ObjetoSpinner oSpinner = new ObjetoSpinner();

            oSpinner.setId(Integer.valueOf(cursor.getString(
                  tiposSeguridadDao.COL_ID_TS)));
            oSpinner.setCadena(cursor.getString(
                  tiposSeguridadDao.COL_TIPOSEGURIDAD_TS));
            tiposSeguridadAdapter.add(oSpinner);

     }while (cursor.moveToNext());
}

// ...
```

Código 12. Definición y carga Spinner "tipos de seguridad".

Test comunicación con servidor Ldap.

Otro de los puntos importantes en lo que a la implementación se refiere de esta pantalla, es la funcionalidad de testeo de la comunicación con el servidor, que procederemos a detallar de una manera más minuciosa a continuación.

```
private void testConexionLdap(ObjetoInstanciaServidor
                              oInstanciaServidor) throws Exception
{
            LDAPConnection conexion = null;
            try{
                  conexion =  oInstanciaServidor.getConexionLdap();

                  final Entry e =
                  conexion.getEntry(oInstanciaServidor.getBaseDN());

                  if (e == null){
                        throw new Exception("error al recuperar
                                             entradas.");
                  }
            }catch (Exception e){
                  Log.e("uc3mAppException", "testConexionLdap", e);
                  throw e;
            }finally{
                  if (conexion != null){
                        conexion.close();
                  }
            }
      }
```

Código 13. Test de comunicación con servidor Ldap.

Para realizar el testeo de la comunicación, lo primero que se lleva a cabo es la obtención de todos los datos introducidos por el usuario en la pantalla de detalle y se crea una instancia de la clase "*ObjetoInstanciaServidor*" que será la que contenga dichos datos como atributos de la misma ("*id*", "*hostname*", "*puerto*", "*tipoSeguridad*", "*bindDn*", "*bindPassword*", "*baseDn*"). Esta es pasada como parámetro a la función "*testConexionLdap*" que invocará la función "*getConexionLdap*" perteneciente a la propia instancia "*ObjetoInstanciaServidor*" recibida como parámetro, para obtener la conexión al ldap. Si la conexión se recupera de manera correcta, se procederá a obtener la entrada del "*base DN*" de la instancia del servidor, y si finalmente, la búsqueda devuelve un resultado satisfactorio, se dará la prueba de comunicación por exitosa. En caso de fallar el establecimiento de conexión con el servidor, o la obtención de la entrada perteneciente a la base DN de la instancia del servidor, se dará la prueba de comunicación por errónea, y se mostrará un dialogo con el mensaje de error al usuario. En el Código 13 se puede observar el fragmento de código perteneciente a la función "*testConexionLdap*".

Para realizar la conexión con el Ldap el método "*getConexionLdap*", llama a su vez a una serie de métodos pertenecientes a las librerías de código abierto de "*UnboundId*" (licencia LGPL), ya que por motivos de rendimiento y para evitar problemas de versionado, los dispositivos Android no contienen algunas librerías nativas de Java, como es el caso de las destinadas al acceso a servidores Ldap (el paquete "*javax.naming.**"). Se llevara a cabo un tipo de conexión u otra en función del

tipo de seguridad que tenga la configuración Ldap que va a ser testeada. A continuación se muestra una breve descripción para cada uno de estos tipos de conexión implementados en el método "*getConexiónLdap*".

- Sea cual sea el tipo de seguridad definida para la conexión, se configuran unos parámetros de conexión comunes a todas ellas:

  - setAutoReconnect(true), habilita la opción de intentar la reconexión con el servidor ldap en caso de error en la comunicación con el mismo.
  - setTimeoutMillis(30000), establece el tiempo de espera máximo antes de la desconexión del servidor ldap. Está indicado en milisegundos.
  - setFollowReferrals(false), establece el comportamiento en caso de obtener derivaciones de una entrada a otra en una búsqueda en el ldap, en este caso se ha deshabilitado esta posibilidad.
  - setMaxMessageSize(1024 * 1024), establece el tamaño máximo en bytes de los mensajes que atenderá la conexión del servidor Ldap.

- Si el tipo de Seguridad es "*SSL*", se genera un "*socketFactory*" que se pasará junto con los parámetros de conexión comunes, el "hostname" y el "*puerto*" para el que queremos obtener la conexión. A continuación se muestra en Código 14, el fragmento de código donde se lleva a cabo este tipo de conexión.

```
SocketFactory socketFactory = null;
if (this.tipoSeguridad.equals(tipoSeguridadSSL)){
            final SSLUtil sslUtil = new SSLUtil(new

      TrustAllTrustManager());
            try{
                  socketFactory = sslUtil.createSSLSocketFactory();
            }catch (Exception e){
                  Log.e("ldappException", "getConexionLdap",e);
                  throw new LDAPException(ResultCode.LOCAL_ERROR,
            "Error al realizar la conexión segura SSL.",e);
            }
}
final LDAPConnectionOptions options = new LDAPConnectionOptions();
options.setAutoReconnect(true);
options.setConnectTimeoutMillis(30000);
options.setFollowReferrals(false);
options.setMaxMessageSize(1024*1024);

final LDAPConnection conn =new LDAPConnection(socketFactory, options,
                                    hostName,
Integer.valueOf(puerto));
// Si tenemos datos para ligar la conexion, comprobamos que
// las credenciales sean correctas.
if ((bindDn != "") && (bindPassword != "")){
      try{
            conn.bind(bindDn, bindPassword);
      }catch (LDAPException le){
            Log.e("uc3mAppException", "getConexionLdap", le);
            conn.close();
            throw le;
      }
```

Código 14. Conexión Servidor Ldap con tipo seguridad "SSL".

- Si el tipo de seguridad es "*TLS*", primero se genera la conexión con los parámetros de conexión comunes, y el "*socketFactory*" con valor nulo y luego se comprueba si realmente dicha conexión es del tipo "*TLS*".

```
SocketFactory socketFactory = null;

final LDAPConnectionOptions options = new LDAPConnectionOptions();
options.setAutoReconnect(true);
options.setConnectTimeoutMillis(30000);
options.setFollowReferrals(false);
options.setMaxMessageSize(1024*1024);

final LDAPConnection conn = new LDAPConnection(socketFactory, options,
                                hostName, Integer.valueOf(puerto));

// Comprobamos conexion TLS.
if (this.tipoSeguridad.equals(tipoSeguridadTLS)){
      final SSLUtil sslUtil = new SSLUtil(new
                                TrustAllTrustManager());
      try{
            final ExtendedResult r =conn.processExtendedOperation(
                                new StartTLSExtendedRequest(
                                sslUtil.createSSLContext()));
            if (r.getResultCode() != ResultCode.SUCCESS){
                                  throw new LDAPException(r);
            }
      }catch (LDAPException le){
            Log.e("uc3mAppException", "getConexionLdap", le);
            conn.close();
            throw le;
      }catch (Exception e){
            Log.e("uc3mAppException", "getConexionLdap", e);
            conn.close();
            throw new LDAPException(ResultCode.CONNECT_ERROR,
                        "Error al realizar la conexión segura TLS.", e);
      }
}

// Si tenemos datos para ligar la conexion, comprobamos que
// las credenciales sean correctas.
if (!bindDn.equals("")) && (!bindPassword.equals("")){
      try{
            conn.bind(bindDn, bindPassword);
      }catch (LDAPException le){
            Log.e("uc3mAppException", "getConexionLdap", le);
            conn.close();
            throw le;
      }
}
return conn;
}
```

Código 15. Conexión Servidor Ldap con tipo seguridad "TLS".

- Si el tipo de conexión es anónima, el flujo de ejecución nunca pasará por los bloques "*if*" para los tipos de conexión con seguridad "*SSL*" ni "*TLS*" solo creará la conexión con la variable "*socketFactory*" con valor *null*, los parámetros comunes de conexión "*options*", y el "*hostname*" y "*puerto*" definidos como atributos de la propia instancia del servidor Ldap.
- Como se puede observar en el Código 16, al final del mismo, se comprueba si "*bindDn*" y "*bindPassword*" tienen valor. Solo entrará en este bloque "*if*" cuando la conexión no es anónima. El "*bindDn*" es el "Distingished name" de la entrada del usuario que esta solicitando la conexión al servidor *Ldap*, y "*bindPassword*" es la contraseña del mismo.

5.2.6 Pantalla "Búsqueda en Servidor Ldap"

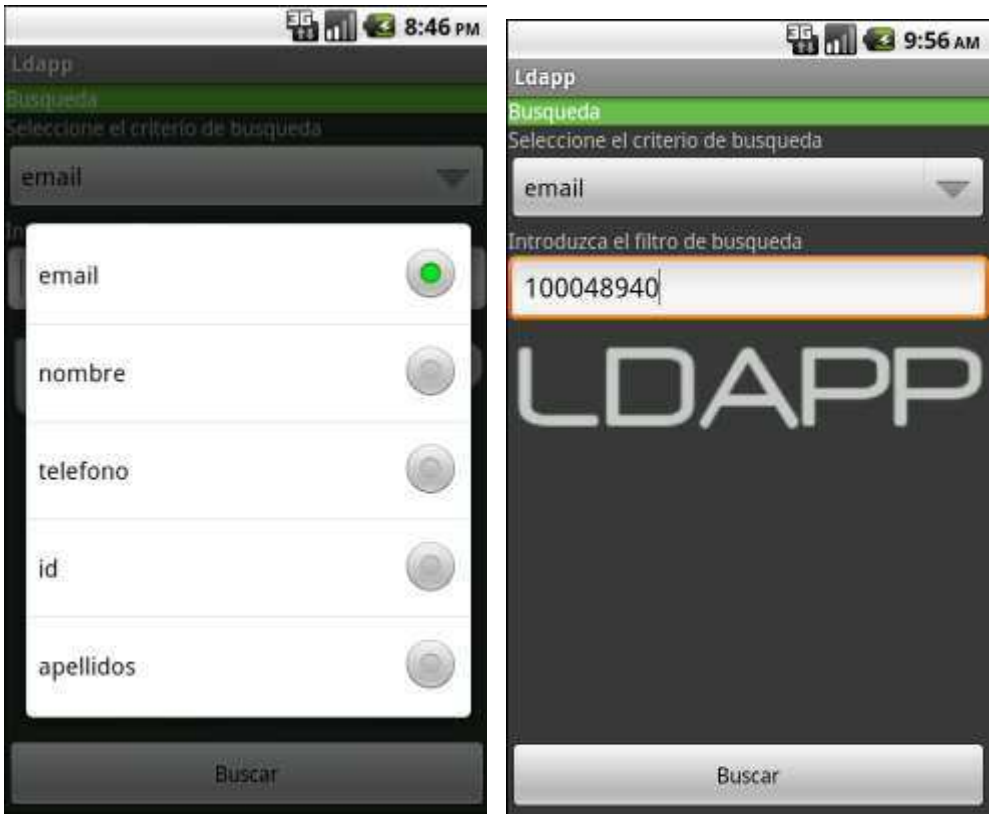


Figura 37. Pantallazos "Búsqueda en Servidor Ldap".

Esta pantalla será la que provea de la funcionalidad de búsqueda en una determinada configuración *Ldap* seleccionada al usuario, permitiéndole realizar un filtrado en función de alguno de los siguientes criterios de búsqueda:

- *Email:* la búsqueda devolverá todas las entradas encontradas en el servidor *Ldap* seleccionado que contengan en el campo "*mail*" de la entrada la cadena de texto de filtrado introducida por el usuario.
- *Nombre:* la búsqueda devolverá todas las entradas encontradas en el servidor *Ldap* seleccionado que contengan en el campo "*cn*" de la entrada, la cadena de texto de filtrado introducida por el usuario.
- *Teléfono:* la búsqueda devolverá todas las entradas encontradas en el servidor *Ldap* seleccionado que contengan en el campo "*telephoneNumber*" de la entrada, la cadena de texto de filtrado introducida por el usuario.
- *Id:* la búsqueda devolverá todas las entradas encontradas en el servidor *Ldap* seleccionado que contengan en el campo "*uid*" de la entrada, la cadena de texto de filtrado introducida por el usuario.

La implementación de la funcionalidad de búsqueda se ha llevado acabo por medio de una tarea asíncrona, que ejecuta dicha acción en "*background*", es decir, como un hilo de ejecución secundario, que corre paralelo al hilo de ejecución principal. Se realiza la petición de búsqueda a el servidor *Ldap* y desde un segundo plano se espera la respuesta del mismo. Esta tarea asíncrona es implementada en una clase denominada "*búsqueda*" que extiende de "*AsyncTask*", clase que contiene todos los métodos necesarios para realizar dicha llamada asíncrona. El método perteneciente a la clase "*búsqueda*" heredado de "*AsyncTask*" es el denominado "doInBackground", y el fragmento de código que lo implementa se muestra en el Código….

En primer lugar, para poder llevar a cabo la búsqueda sobre el servidor *Ldap*, es necesario establecer la conexión con el mismo, este punto se explica más extendidamente en el apartado *3.2.5.* Y dicha conexión para la búsqueda es representada por la variable "*conexión*" como podrá observar en el Código….

En segundo lugar se define la petición de búsqueda, que se realiza creando una nueva instancia de la clase "*SeachRequest*" (clase perteneciente a las librerías de "*unboundId*"), a la que le pasamos como parámetros, el "*base DN*" de búsqueda, es decir, el nodo raíz a partir del cual realizaremos la búsqueda dentro del árbol del servidor *Ldap*. El segundo parámetro es el objetivo de búsqueda, que definirá el ámbito de la misma, en nuestro caso será del tipo "*SUB*" que nos permitirá realizar la búsqueda sobre el propio "base DN" y sobre todas las entradas subordinadas a el, sin limite de profundidad dentro del árbol del servidor *Ldap*. El tercer parámetro corresponderá al filtro de búsqueda, en el Código…. se puede observar como se forma dicho filtro (variable "*filter*") en función del criterio de búsqueda seleccionado por el usuario.

Y por ultimo realizamos la búsqueda sobre la conexión al servidor Ldap generada previamente, como se muestra en el Código 16, esta devolverá un objeto del tipo "*SearchResult*" también perteneciente a las librerías de "*UnboundId*", que parsearemos en el método "*onPostExecute*" de la clase "*búsqueda*" y del cual

obtendremos todas las entradas obtenidos de la búsqueda y que a posteriori listaremos para el usuario.

```
protected Integer doInBackground(String... params) {
       Filter filter = null;
       ObjetoSpinner seleccion;
       String filtroBusqueda;
       LDAPConnection conexion;
       SearchRequest searchRequest = null;

       try{
           seleccion = (ObjetoSpinner) cbCriterio.getSelectedItem();
           filtroBusqueda = txtFiltroBusqueda.getText().toString();
           conexion = oInstanciaServidor.getConexionLdap();

           // Definimos filtro de búsqueda
           if(seleccion.getId() == IConstantes.opc_email){
              if (filtroBusqueda != null &&
                           !filtroBusqueda.equals("")){
                     filter = Filter.create("(mail=*" + filtroBusqueda +
                                                 "*)");
              }else{
                     filter = Filter.create("(mail=*)");
              }

           }else if(seleccion.getId() == IConstantes.opc_id) {
              if (filtroBusqueda != null &&
                           !filtroBusqueda.equals("")){
                     filter = Filter.create("(uid=*" + filtroBusqueda +
                                                 "*)");
              }else{
                     filter = Filter.create("(uid=*)");
              }
           }
              // ...

           // Definimos petición de busqueda
           searchRequest = new
                     SearchRequest(oInstanciaServidor.getBaseDN(),
                                        SearchScope.SUB, filter);
           searchRequest.setSizeLimit(0);

           // Lanzamos búsqueda sobre el servidor Ldap
           searchResult = conexion.search(searchRequest);
           conexion.close();

       }catch(Exception e){
           // ...
       }
       return 250;
```

Código 16. Búsqueda en Servidor Ldap.

5.2.7 Pantalla "Listado Resultado Búsqueda en Servidor Ldap"

Figura 38. Pantallazo "Listado Resultado Búsqueda en Servidor Ldap".

Esta pantalla es la encargada de mostrar todas las entradas encontradas como resultado de una búsqueda realizada por el usuario sobre el servidor *Ldap*, en la Figura 38 se muestra una captura de pantalla de la misma.

Al igual que en la pantalla de listado de servidores *Ldap* (apartado 3.2.4), la clase de "*ListadoResultados*" extiende de la clase "*ListActivity*", y es en si misma una lista. Que a su vez, también tiene definido un adaptador del tipo "*ArrayAdapter<ObjetoEntradaLdap>*" encargado de asignar a cada "*ítem*" del listado, en este caso del tipo "*ObjetoEntradaLdap*" una determinada apariencia. A continuación en el Código 17 se muestra la instrucción de asignación del adaptador a la clase "*ListadoResultados*".

```
setListAdapter(new LdapObjectAdapter(this, R.layout.rowldap,
       lEntradasLdap,
       oInstanciaServidor));
```

Código 17. Asignación ArrayAdapter a "ListadoResultados".

Los parámetros pasados a la nueva instancia de la clase "*LdapObjectAdapter*" (que a su vez extiende de "*ArrayAdapter<ObjetoEntradaLdap>*") representan lo siguiente:

- *this*: representa el contexto de la clase "ListadoResultados".
- *R.layout.rowldap*: es la referencia al xml donde se definen los elementos gráficos que compondrán la interfaz para cada "*ítem*" del listado de resultados es decir, la apariencia de cada elemento de la lista.
- *lEntradasLdap*: es la lista que contiene todas las entradas recuperadas de la búsqueda en el servidor Ldap.
- *oInstanciaServidor*: instancia del servidor donde se ha llevado a cabo la búsqueda.

Como se muestra a continuación en la declaración de los elementos de la interfaz, cada elemento del listado (en el Código….) estará compuesto a su vez por 4 campos, 3 de ellos visibles y uno de ellos oculto al usuario:

- Nombre: nombre de la entrada para ese ítem de la lista. Solo se mostrará si no es vacío en "*ObjetoEntradaLdap*".
- Email: email de la entrada para ese ítem de la lista. Solo se mostrará si no es vacío en "*ObjetoEntradaLdap*".
- Telefono: teléfono de la entrada para ese ítem de la lista. Solo se mostrará si no es vacío en "*ObjetoEntradaLdap*".
- Uid: representa el id para un determinado ítem de la lista de resultados. Siempre estará oculto al usuario.

Cada ítem de la lista estará formado por un layout del tipo "LinearLayout" que contiene los cuatro campos mencionados previamente, cada uno de ellos representado por un elemento "*TestView*", como se muestra en el Codigo 18.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:background="@drawable/list_selector">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_weight="1"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/txt_nombre"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center_vertical"
            android:singleLine="true"
        />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:id="@+id/txt_email"
            android:singleLine="true"
            android:ellipsize="marquee"

        />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:id="@+id/txt_telefono"
            android:singleLine="true"
            android:ellipsize="marquee"
        />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:id="@+id/txt_uid"
            android:singleLine="true"
            android:ellipsize="marquee"
        />
    </LinearLayout>
</LinearLayout>
```

Código 18. Xml "rowldap.xml".

5.2.8 Pantalla "Vista Entrada de Servidor Ldap"

La pantalla que se describe a continuación es la encargada de mostrar en detalle una determinada entrada seleccionada por el usuario del resultado de la búsqueda sobre un determinado servidor Ldap. En ella se visualizarán los siguientes campos siempre y cuando su valor no sea vacío:

- Nombre.
- Apellidos.
- Teléfono.
- Email.

También se permitirá al usuario añadir al detalle cualquiera de los demás campos pertenecientes a la entrada seleccionada. Para ello, la pantalla de detalle cuenta con un combo, que listará todos ellos y dará la posibilidad al usuario de seleccionar y añadirlo clicando en el botón "*Añadir Atributo*". En la Figura 39 expuesta a continuación se muestra una captura de pantalla de dicho detalle.

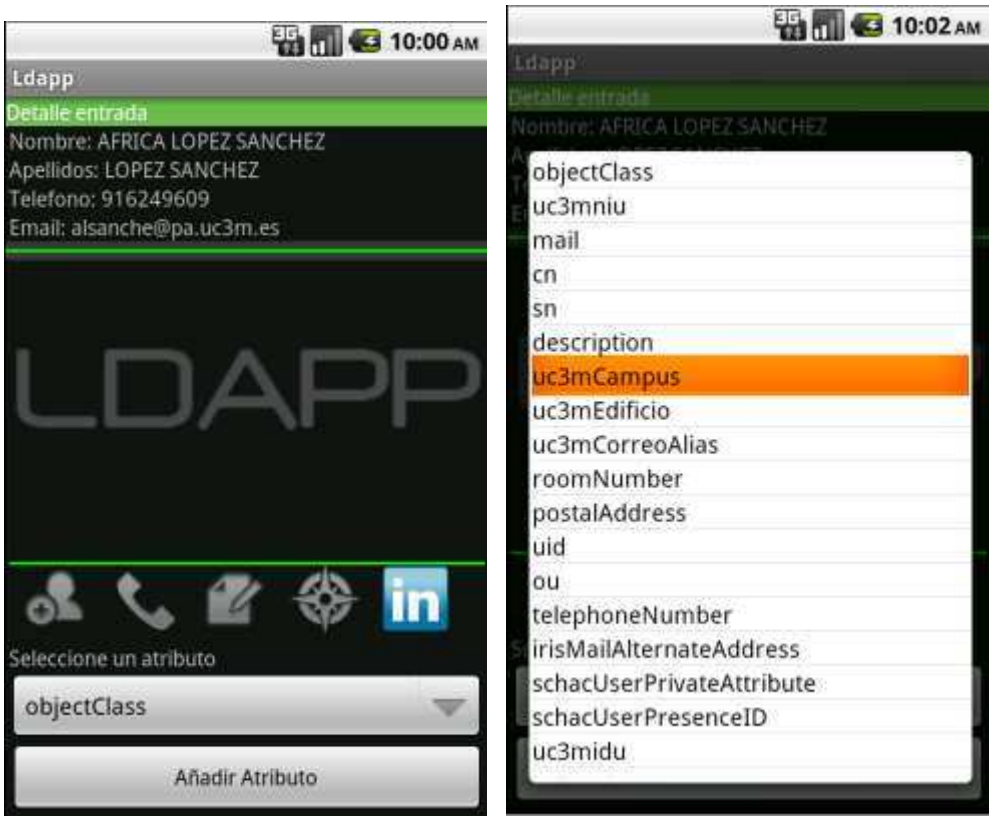


Figura 39. Pantallazos "Vista Entrada Servidor Ldap".

Esta pantalla además de mostrar los atributos de una determinada entrada del servidor *Ldap*, tiene una serie de funcionalidades que se exponen a continuación.

Añadir Entrada Ldap a Agenda de Contactos.

Esta funcionalidad esta representada en la pantalla mediante el icono , y nos permitirá añadir la entrada *Ldap* que estamos visualizando, a la lista de contactos de nuestro dispositivo móvil.

Cuando seleccionemos esta opción, se abrirá la pantalla de nuevo contacto con los campos "Nombre", "Teléfono", y "Correo electrónico" rellenos, siempre y cuando estos no estén vacíos en la entrada *Ldap*. A continuación en la Figura 40 se muestra una captura de pantalla para esta opción.

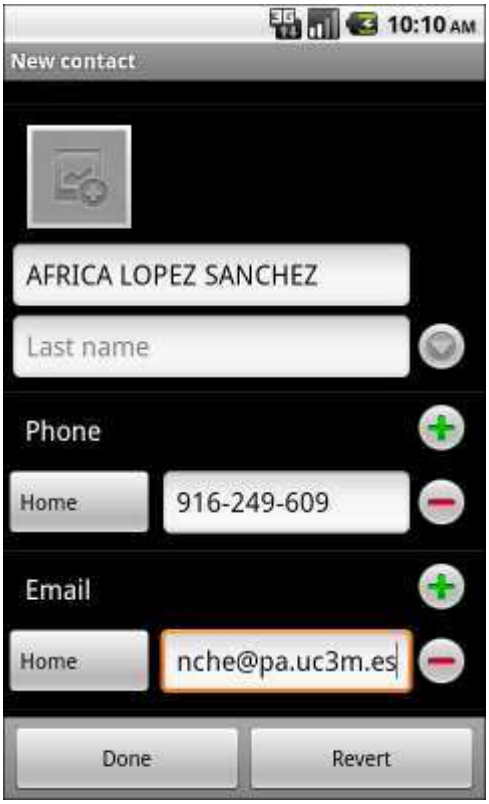


Figura 40. Pantallazo "Añadir Entrada Ldap como Nuevo contacto de agenda".

Sobre la implementación de esta funcionalidad, cabe destacar, la llamada mediante un "*Intent*" (clase encargada de crear la actividad que realizará una determinada funcionalidad) a la acción nativa de *Android*, que se encargará de mostrar la pantalla de nuevo contacto, cumplimentada con los datos de nuevo contacto, que le pasamos como parámetro. A continuación en el Código 19 se muestra el fragmento de código que realiza esta operación.

```
private void addContacto(){
	// Añadimos contacto a la agenda.
	Intent intent = new Intent(Intent.ACTION_INSERT,
			ContactsContract.Contacts.CONTENT_URI);

	if (nombre != null && !nombre.equals("")){
		intent.putExtra(ContactsContract.Intents.Insert.NAME, nombre);
	}
	if (email != null && !email.equals("")){
		intent.putExtra(ContactsContract.Intents.Insert.EMAIL, email);
	}
	if (telefono != null && !telefono.equals("")){
		intent.putExtra(ContactsContract.Intents.Insert.PHONE,
		telefono.replace("-","").trim());
	}
	startActivityForResult(intent,1);
}
```

Código 19. Añadir Entrada Ldap como Nuevo contacto de agenda.

Como puede observarse en el código, "*Intent.ACTION_INSERT*" es la acción nativa de Android, que nos permitirá cargar la pantalla de "*nuevo contacto*", y mediante "intent.putExtra" se cargarán en el "*intent*" los valores que rellenarán la ficha de "*nuevo*

contacto". Por ultimo, realizamos la llamada al "*intent*" por medio del método "*startActivityForResult*" que carga la pantalla de "*nuevo contacto*".

Marcar Teléfono de Entrada Ldap.

Esta funcionalidad esta representada en esta pantalla mediante el icono  . Cuando el usuario seleccione esta opción, se cargará la pantalla de marcado del dispositivo móvil, cumplimentada con el número de teléfono de la entrada que estaba visualizando en ese momento, siempre y cuando, el número de teléfono, no estuviese vacío, en ese caso, se mostraría un dialogo al usuario indicándole que dicha entrada no contiene un número de teléfono valido.

A continuación se muestra en la Figura 41 una captura de la pantalla de marcado de un dispositivo móvil Android.

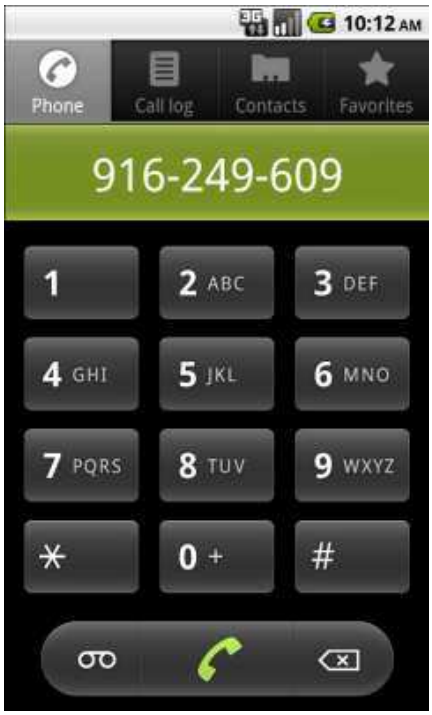


Figura 41. Pantallazo "Marcar número de teléfono de Entrada Ldap".

Al igual que la funcionalidad de "*añadir entrada Ldap a agenda de contactos*", la pantalla de marcado, se carga mediante una acción propia de la plataforma Android. A continuación en el Código 20 se muestra la función llamada para la carga de dicha pantalla.

```
private void llamarContacto(){
      if (telefono != null && !telefono.equals("")){
            intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" +
                        telefono.replace("-","").trim()));
            startActivity(intent);
      }// ...}
```

Código 20. Añadir Entrada Ldap como Nuevo contacto de agenda.

Como se puede observar en el Código….., el tipo de la nueva instancia "*Intent*", "*Intent.ACTION_DIAL*" que tiene por primer parámetro, es la acción propia de *Android* que indicará que lo que se desea es la carga de la pantalla de marcado del dispositivo. Como segundo parámetro se le pasa la cadena de texto que representa el teléfono que se va a mostrar en dicha pantalla.

Redactar Email Entrada Ldap.

Esta funcionalidad esta representada por el icono  , y consistirá en la carga de la pantalla de redactado de emails, cumplimentada con el email de la entrada que el usuario este visualizando en ese momento. Si se da el caso de que la entrada no tiene el atributo "email" o lo tiene vacío, se mostrará un dialogo a el usuario indicándole dicha incidencia.

Al igual que en las anteriores funcionalidades expuestas en este apartado, la carga de la pantalla de redactado de email, se realizará mediante la llamada a una acción propia de *Android* como se muestra en el Código 21.

```
private void emailContacto(){
      if (email != null && !email.equals("")){
            intent = new Intent(Intent.ACTION_SENDTO,
                              Uri.fromParts("mailto", email, null));
            startActivity(intent);
      }else{
            throwMensaje(this,"El contacto no tiene ningún email
                                    asociado.");
      }
}
```

Código 21. Redactar email a Entrada Ldap.

Como se puede observar en el Código…., como primer parámetro de la nueva instancia "*Intent*", pasamos el tipo de acción a la que queremos llamar "Intent.ACTION_SENDTO", como puede observarse en este caso se cargará la pantalla de redactado de email. Como segundo parámetros se le pasa el campo "*mailto*" de dicha pantalla, que es el email de la entrada que esta visualizando el usuario en ese momento.

Geoposicionar Entrada Ldap.

Esta funcionalidad esta representada en la pantalla de detalle de la entrada Ldap por medio del icono  . Consiste en posicionar la entrada del Ldap que este visualizando en cada momento el usuario. Esta operación se lleva a cabo utilizando el atributo "*postalAddress*" de la entrada Ldap para posicionarla en un mapa utilizando la API de "*Google Maps*". Para realizar dicha operación, se llama a otra "*activity*" por medio del fragmento de código que se muestra a continuación. La implementación de esta funcionalidad se desarrollará con más profundidad en el apartado 3.2.8.

```
private void posicionarContacto(){
	if (direccion != null && !direccion.equals("")){
		intent = new Intent(this,MapaActivity.class);
		// ...
		intent.putExtra("direccionBusqueda", direccion);
		 startActivity(intent);
	}else{
		throwMensaje(this,"El contacto no tiene ninguna dirección");
	}
}
```

Código 22. GeoPosicionar entrada Ldap.

Búsqueda en LinkedIn de la Entrada Ldap.

Esta operativa esta representada en esta pantalla por el icono  , y permitirá al usuario realizar una búsqueda de contactos de "*LinkedIn*" que coincidan en nombre y apellidos con los de la entrada del Ldap que en ese momento este visualizando el usuario. A continuación se muestra en el Código 23 la llamada a la "*activity*" que llevará acabo esta funcionalidad.

```
private void busquedaLinkedIn(){
	intent = new Intent(this,LinkedInActivity.class);
	// ...
	startActivity(intent);
}
```

Código 23. Búsqueda en LinkedIn.

5.2.9 Pantalla "GeoPosicionamiento de entrada Ldap"

Gracias a la API que nos proporciona Google Maps, en el paquete "*com.google.android.maps*", se ha podido llevar a cabo toda la funcionalidad de geoposicionamiento de la dirección que se explicará en profundidad en este apartado. Las Clases utilizadas de las librerías de Google Maps se describen brevemente a continuación.

- `MapView`: obtiene el mapa solicitado y lo muestra en pantalla.
- `MapController`: gestiona todas las instrucciones que deberán ejecutarse sobre el mapa, cambios de posición, zoom, etc…
- `GeoPoint`: se encarga de la georeferenciación de una determinada dirección, es decir, calcula latitud y longitud para dicha dirección.
- `Overlay`: nos permite dibujar objetos en una capa superpuesta a el mapa, como puede ser el puntero que indica nuestra posición, una brújula, etc…
- `MapActivity`: clase de la que extenderá la clase Activity y de la cual tendrá que extender cualquier clase que quiera realizar una gestión de mapas.

Para la implementación de la funcionalidad de geoposicionamiento de la entrada Ldap que esta visualizando el usuario desde la pantalla de "*Vista entrada servidor Ldap*" por medio de la API de "*Google Maps*" se han llevado a cabo una serie de pasos que se describen en detalle a continuación.

- Obtención de "API Key" de "*Google Maps*"

  Como paso previo a la utilización de la API de "*Google Maps*", ha sido necesaria la obtención de una "*API Key*" de aplicación para poder utilizar los servicios que dicha API provee. Google utiliza este mecanismo para asegurar que el desarrollador que accede a ella hace un uso correcto de los datos obtenidos de cualquiera de sus servicios. Es por ellos que el desarrollador tiene que registrarse y aceptar las condiciones de uso de la misma.

  Para llevar a cabo este registro hay que seguir los siguientes pasos:

  - Obtención del resumen MD5 del certificado de la aplicación Android.
    Nótese que todas las aplicaciones Android contienen un certificado que asegura su autoría y las liga a su desarrollador. Si se utiliza el plugin de Eclipse para Android, todas las aplicaciones Android irán firmadas por un certificado de prueba, que permite valga la redundancia la prueba de las mismas en dispositivos móviles. Este certificado de prueba se obtiene a partir del fichero de claves de prueba "*debug.keystore*" incluido en la propia SDK de Android (Si el desarrollador desease distribuir su aplicación por medio de los servicios de descarga, por ejemplo Android Market, tendría que crear su propio fichero de claves "*keystore*" con el que generar un certificado valido para su aplicación y poder firmarla). Para la obtención del resumen MD5 se ha utilizado la herramienta "*keytool*" incluida en la SDK de Java. A continuación se muestra en el Código 24 las instrucciones ejecutadas para la obtención del resumen MD5 a partir del fichero de claves de pruebas "*debug.keystore*".

```
keytool -list -alias androiddebugkey -keystore
C:\Users\jorge\.android\debug.keystore -storepass android -keypass
android

certificado de huella dactilar:

28:84:50:8F:96:15:00:E7:CB:F0:8A:62:79:98:0E:AF
```

Código 24. Obtención resumen MD5 mediante "Keytool".

  - Registro de resumen MD5 en Google Maps.
    Este registro deberá realizarse desde la página web de registro de Google Maps [8]. Para llevar a cabo este paso será necesario que el desarrollador disponga de una cuenta de Google. En caso de cumplir este requisito,

tendrá que aceptar los términos de uso y enviar el resumen obtenido en el punto anterior.

- Obtención de "*API Key*".
  Finalmente Google Maps nos facilita la clave alfanumérica, la denominada "API Key", que tendremos que enviar adjunta en cada una de las operaciones que realicemos contra los servicios de "*Google Maps*". A continuación se muestra la API Key obtenida para la aplicación "*Ldapp*".

```
API Key: 0Ob53IuedV_SvcQ2vUo3gBVkdSSAO7LpRpGvx8w
```

Código 25. API Key para aplicación "Ldapp" de Google Maps.

- Tras la obtención de la API Key, se incluye en la declaración del elemento "*com.google.android.maps.MapView*" dentro de la definición de la interfaz gráfica de la pantalla que mostrará la ubicación de la entrada del Ldap en el mapa de Google Maps, como se muestra en el Código 26.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
       <TextView
       android:id="@+id/myLocationText"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       />
       <com.google.android.maps.MapView
       android:id="@+id/map_view"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:enabled="true"
       android:clickable="true"
       android:apiKey="0Ob53IuedV_SvcQ2vUo3gBVkdSSAO7LpRpGvx8w"
       />
</LinearLayout>
```

Código 26. Declaración Interfaz usuario pantalla geoposicionamiento entrada Ldap.

- Por ultimo solo queda explicar la implementación de la funcionalidad de posicionamiento de la entrada Ldap. Dicha funcionalidad es llevada acabo por la clase "*MapaActivity*" que a su vez extiende de la clase "*com.google.android.maps.MapActivity*" que proveerá de los servicios disponibles en la API de Google Maps.

  En primer lugar en el método "*onCreate*" de la clase "*MapaActivity*" llamamos al método "*initMapView*" encargado de inicializar la vista del mapa. En el se

inicializan los atributos de la propia clase "*MapaActivity*" que más tarde se utilizarán para el posicionamiento de una determinada dirección.

A continuación se muestra en el Código 27 el fragmento de código para la función "*initMapView*".

```
private void initMapView(){
       // objeto que tiene la función de georeferenciar un
       // determinada dirección textual.
       geoCoder = new Geocoder(this);
       map = (MapView) findViewById(R.id.map_view);
       // controller se utilizará para posicionarnos en el
       // mapa y hacer zooms.
       controller = map.getController();
       // convierte el mapa en modo satelite.
       map.setSatellite(true);
       // activa los controles de zoom estandar.
       map.setBuiltInZoomControls(true);
}
```

Código 27. Método "initMapView".

Tras la inicialización de la vista "*MapView*", se procede el posicionamiento de la dirección de la entrada Ldap por medio de la llamada al método "*initAddressLocation*" también desde el método "*onCreate*" de la clase "*MapaActivity*". A continuación en el Código 28 se muestra el fragmento de código que realiza esta función.

```
private void initAddressLocation()throws Exception{
       List<Address> addresses =
              geoCoder.getFromLocationName(direccionBusqueda,5);

      if(addresses.size() > 0){
              // Obtiene latitud y longitud para la direccion
              // pasada como parametro
              geoPoint = new GeoPoint(
                     (int)(addresses.get(0).getLatitude() * 1E6),
                    (int) (addresses.get(0).getLongitude() * 1E6));
              // Posiciona puntero en el mapa para la latitud
              // y longitud que representa geoPoint
              controller.animateTo(geoPoint);
              // Configuramos el zoom que deseamos para el mapa.
              controller.setZoom(12);
              // ...
      }else{
              throwMensaje(this,"Por favor, introduzca una
              dirección valida.");
      }
}
```

Código 28. Método "initAddressLocation".

La primera instrucción del método "*initAddressLocation*", esta enfocada a obtener la posición exacta para una determinada dirección textual, esto se realiza por medio del método de Google Maps "*geoCoder.getFromLocationName*" que asignará a la lista "*addresses*" del tipo "*List<Address>*" todas las referencias que se encuentren en los servidores de Google Maps para la dirección textual pasada

como parámetro a dicho método, en el caso que nos ocupa la variable "*direccionBusqueda*".

Tras obtener la localización para "*direccionesBusqueda*" contenida en el primer elemento de la lista "*addresses*" convertirá esta a un punto en formato legible para el "*controller*" del mapa (el "controller" es el encargado de enviar las instruccones de posicionamiento a la vista MapView) por medio de la instanciación de la clase "*GeoPoint*", y guardará el resultado en el atributo de la clase "*MapaActivity*" llamado "*geoPoint*".

Finalmente posicionará la variable obtenida "*geoPoint*" en el mapa, mediante el método del controlador "*controller.animateTo(geoPoint)*". El método "*controler.setZoom(12)*" simplemente especifica un determinado zoom para el mapa.

A continuación se muestra en la Figura 42 una captura de pantalla de esta pantalla.

Figura 42. Pantallazo "Geoposicionamiento de entrada Ldap".

5.2.10 Pantalla "Listado Resultados de Búsqueda en LinkedIn"

La implementación de la funcionalidad de búsqueda sobre el servidor de *LinkedIn* se a llevado a cabo utilizando la propia API de *LinkedIn*. Para poder utilizar dicha API desde la aplicación "*Ldapp*" se han seguido los siguientes pasos:

- Obtención de la API Key de *LinkedIn*

  Al igual que pasaba con la API de *Google Maps*, para la utilización de la API de *LinkedIn* desde cualquier aplicación es necesaria la obtención de una "*API Key*".
  Para conseguirla es necesario tener una cuenta en *LinkedIn* y acceder al registro de aplicaciones, desde el portal para desarrolladores de *LinkedIn* [10]. Una vez finalicemos dicho registro obtendremos una "*API Key*" y una "*secret API Key*" de la cual explicaremos su utilidad más adelante.

- Autenticación mediante *OAuth*

  El objetivo de este tipo de autenticación es poder acceder a recursos protegidos sin la necesidad de enviar nuestro usuario y contraseña para obtener los mismos. En este enlace [11] se explica más detalladamente en que consiste este tipo de autenticación. A continuación se muestra en la Figura 43 un diagrama con los pasos que sigue este proceso de autenticación.

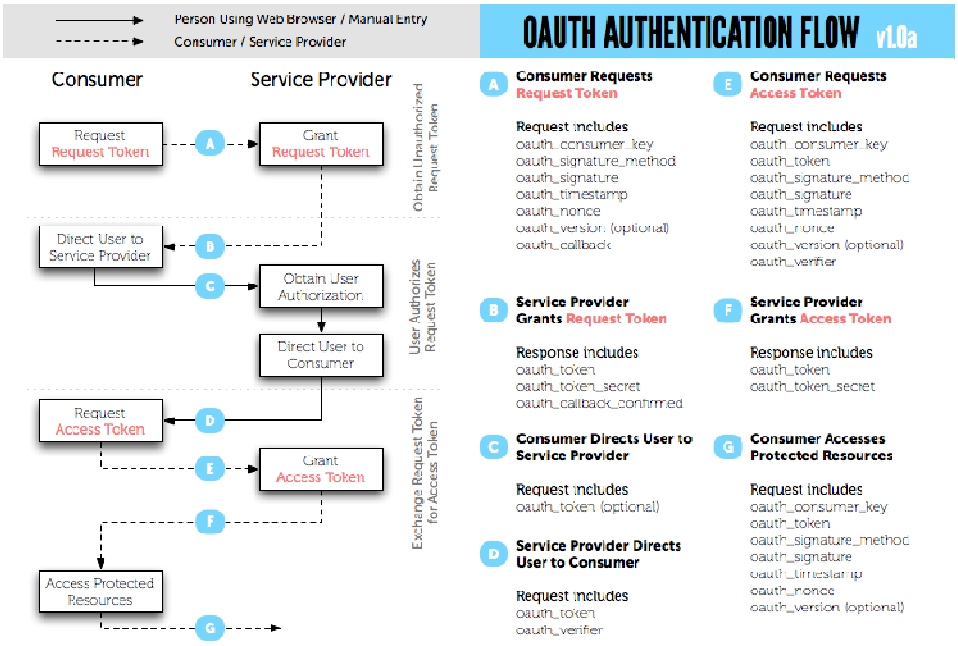


Figura 43. Diagrama del flujo de autenticación OAuth.

En nuestro caso, si el usuario no se ha autenticado previamente desde la pagina de login de *LinkedIn* y a concedido permisos a la aplicación para tener acceso a los datos del servicio, se le redirecciona hasta la misma. Una vez el usuario concede dichos permisos a la aplicación, el flujo de control vuelve del servicio de *LinkedIn* a la aplicación, y esta procede a realizar la búsqueda sobre el propio servicio de los

usuarios que coincidan en nombre y apellidos con los de la entrada Ldap que esta visualizando previamente el usuario, y que más tarde se listarán.

- Búsqueda en servicio *LinkedIn*

  Tras comprobar que la aplicación tiene los permisos necesarios concedidos por el usuario de acceso a los servicios de *LinkedIn*, se procederá a realizar la búsqueda de los contactos que coinciden en nombre y apellidos con los de la entrada *Ldap* que estaba visualizando el usuario. Esta operación está implementada mediante una petición *HTTP* a los servicios de *LinkedIn*, como se muestra en el Código 29

```
String response =
httpRequest("http://api.linkedin.com/v1/people-
search:(people:(id,first-name,last-name,picture-
url,headline,public-profile-url),num-results)?" +

            "keywords=" +
            "&first-name=" + URLEncoder.encode(nombre)  +
            "&last-name=" + URLEncoder.encode(apellidos)  +
            "&current-company=true" +
            "&current-title=true" +
            "&start=" + String.valueOf(start) +
            "&count=" + String.valueOf(MAX_COUNT) +
            "&format=json");
```

Código 29. Petición de búsqueda en servicios LinkedIn.

La respuesta a dicha petición de búsqueda estará en formato *JSON*. Este será parseado para obtener cada uno de los contactos obtenidos, y cada contacto obtenido se almacenará en una lista de objetos del tipo "*ArrayList<ObjetoPersonaLinkedIn>()*". En el Código 30 se muestra un fragmento de uno de los resultados obtenidos en formato *JSON*.

```
{
 "numResults": 598,
 "people": {
  "_count": 10,
  "_start": 0,
  "_total": 110,
  "values": [
   {
    "firstName": "Jorge Ángel",
    "id": "woopLlgt8J",
    "lastName": "Martín-Maestro   Hermida"
   },
   {
    "firstName": "Jorge",
    "id": "wUWrYrsa5E",
    "lastName": "Matías Martín"
   }
```

Código 30. Resultado formato JSON LinkedIn.

- Listado resultados búsqueda en servicio LinkedIn

Tras obtener los resultados de la petición de búsqueda en LinkedIn, se procede a listarlos. La Actividad "*LinkedInActivity*" es la encargada realizar el listado de los contactos encontrados. Esta Actividad extiende de la clase "*ListActivity*" y por tanto contiene toda la funcionalidad de las pantalla de listado de Android.
Como ya hemos venido viendo en pantallas anteriores, este tipo de actividades "*ListActivity*" tienen la necesidad de tener asociado un adaptador, que será el encargado de mostrar cada ítem del listado con un formato previamente definido.
En este caso el adaptador será del tipo "*UpdateAdapter*" que a su vez extiende de "*BaseAdapter*". Este adaptador de encargará de mostrar en cada ítem del listado los siguientes campos obtenidos de cada elemento de la lista de objetos del tipo "*ObjetoPersonaLinkedIn*".

- Nombre contacto.
- Apellidos contacto.
- Perfil contacto.

A continuación se muestra en la Figura 44 una captura de pantalla del listado de resultados de LinkedIn.

Figura 44. Pantallazo "Listado Resultados de Búsqueda en LinkedIn".

6 Gestión del proyecto

Este capítulo describe todos los aspectos relacionados con la gestión llevada a cabo sobre el presente proyecto fin de carrera tales y como metodología, ciclo de vida, recursos y una breve descripción de la historia del proyecto.

6.1 Metodología

La metodología de Ingeniería de Software hace referencia a como han de obtenerse los distintos productos parciales y finales durante el ciclo de vida de una aplicación, es decir, proponen un proceso disciplinado en cuanto a la ejecución de la misma se refiere, realizando especial énfasis en su planificación y control.

En el caso que nos atañe, se tomo la decisión de utilizar una metodología orientada a objetos basada en *UML 2.0* [6]. Se tomo la decisión de utilizar la especificación *UML 2.0* ya que era la más adecuada a las necesidades del proyecto, dotándome del lenguaje de modelado necesario, para especificar o describir los métodos y/o procesos que debía llevar acabo la aplicación.

6.2 Ciclo de vida

El desarrollo del proyecto ha seguido un modelo de "*ciclo de vida en Cascada*". Este consiste en dividir el desarrollo de la aplicación en una serie de etapas que se llevarán acabo de manera consecutiva, estas se describen más detalladamente a continuación.

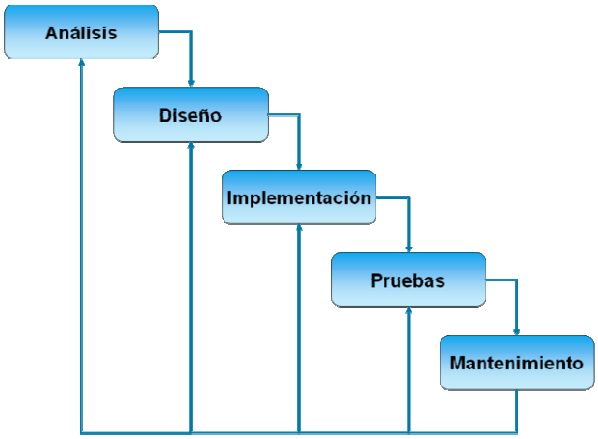


Figura 45. Ciclo de vida en cascada.

- Obtención de requisitos (Análisis)
  Proceso de obtención de las necesidades del cliente, las especificaciones que deberá satisfacer la aplicación.
- Diseño de la aplicación
  Consistente en el proceso de abstracción de todas las necesidades obtenidas del proceso de obtención de requisitos.
- Implementación de la aplicación
  Proceso de implementación del código fuente de la aplicación en un determinado lenguaje de programación.
- Pruebas de la aplicación
  Etapa en la que el equipo de desarrollo se encargará de testear el correcto funcionamiento de cada uno de los módulos de la aplicación.
- Mantenimiento de la aplicación
  Es una de las etapas más criticas, ya que en ella, el usuario entra en contacto directo con la aplicación, y pueden detectarse errores en el funcionamiento de la misma, así como nuevos requisitos a desarrollar.

El ciclo de vida en Cascada, además de seguir cada una de las etapas que se mencionaron previamente de manera secuencial, admite la posibilidad de iterar en cada una de ellas, es decir, si por ejemplo una vez en la etapa de mantenimiento de la aplicación surge la necesidad de cambiar algo en el diseño de la misma, se podrá volver a realizar dicho diseño y consecuentemente, se volverán a llevar a cabo las etapas de implementación y pruebas, posteriores a ella.

6.3 Recursos

Este apartado se encargará de detallar todos los recursos utilizados para llevar acabo este proyecto, tanto humanos como materiales, mediante la utilización de diagramas *RBS.*

En la Figura 46 Se puede observar el diagrama *RBS* para los recursos humanos utilizados en este proyecto. Que son fundamentalmente, un jefe de proyecto encargado de la gestión del proyecto, y la toma de decisiones, un analista, encargado de la toma de requisitos, y el diseño de la aplicación y por ultimo, un programador, encargado de la implementación de los diseños aportados por el analista.

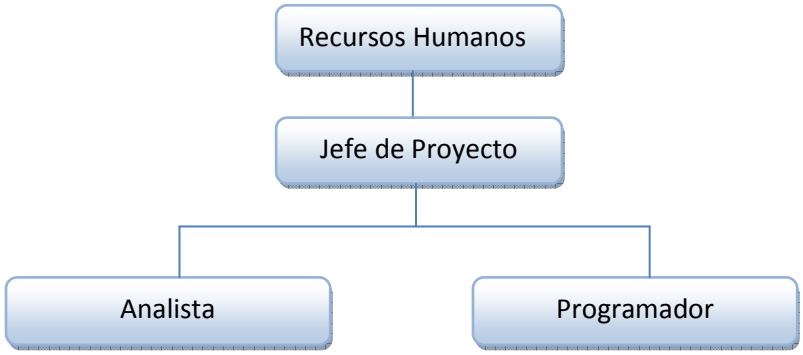


Figura 46. Diagrama RBS de Recursos Humanos.

En cuanto a los recursos materiales utilizados durante el desarrollo del proyecto, podemos desglosarlos en dos categorías, Hardware y Software, como se muestran a continuación en el diagrama *RBS* de la Figura 46

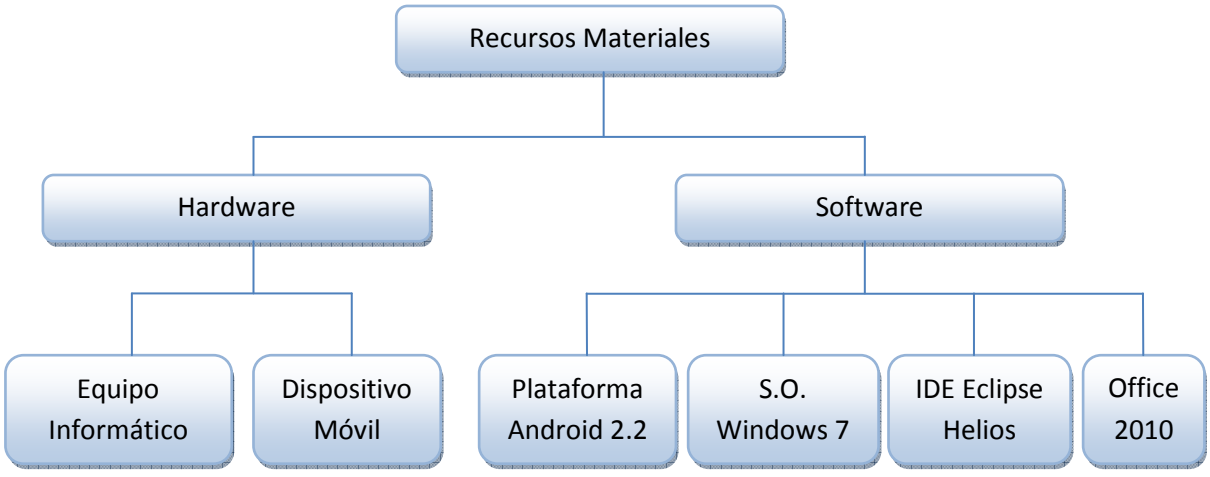


Figura 47. Diagrama RBS de Recursos Materiales.

Dentro de la categoría de recursos materiales dedicada al Hardware, se han hecho uso de los siguientes elementos:

- Equipo Informático
  Dell XPS 8300, Intel Core i5 2500 de 3.30 GHz de velocidad de procesado.
- Dispositivo Móvil
  Smartphone HTC Wildfire.

En la categoría de Software se han utilizado las herramientas que se describen a continuación:

- Plataforma Android 2.2
  El Sistema Operativo del dispositivo móvil utilizado para las pruebas realizadas sobre la aplicación "*Ldapp*", ha sido la versión 2.2 de Android, la denominada "*Froyo*" y con el nivel de API 8.
- S.O. Windows 7
  El sistema operativo instalado en el equipo informático, era Microsoft Windows 7, *Home Edition.*
- IDE Eclipse Helios
  La versión del IDE de desarrollo utilizado para implementar la aplicación ha sido *Eclipse*, en su versión "*Helios Service Release 2, Build id: 20110301-1815*".
- Office 2010
  La versión de la suite de Microsoft utilizada para la documentación de la presente memoria ha sido Microsoft Office "*2010 Starter*".

6.4 Presupuesto

En este apartado se expone de manera detallada el presupuesto estimado para la realización del presente proyecto. Los costes derivados de la realización del mismo, los podemos tipificar de la siguiente manera:

- **Costes directos:**
  - Recursos Humanos: Coste en "horas hombre" empleadas durante la ejecución de cada una de las fases del proyecto, donde ya se incluyen un 20% adicional en conceptos de gastos de Seguridad Social.
  - Materiales fungibles: Coste en materiales consumibles utilizados durante el proyecto.
  - Herramientas software: Coste proporcional al tiempo de ejecución del proyecto en licencias de Software (amortización de software).
  - Herramientas hardware: Coste proporcional al tiempo de ejecución del proyecto en equipos informáticos, dispositivos móviles, etc… (amortización del hardware).
- **Costes indirectos:**
  - Costes en suministros, alquileres de inmuebles, etc…, durante la realización del proyecto. Se han estimado en un 20% del total de costes directos imputados sobre el proyecto.

A continuación se detalla cada uno de los costes asociados al proyecto, desglosados anteriormente.

Costes de Recursos Humanos					
Apellidos y nombre	***N.I.F.***	***Categoría***	***Dedicación (hombres mes)****	***Coste hombre mes***	***Coste (Euro)***
Persona 1		Jefe de Proyecto	1,2	3.500,00	4.200,00
Persona 2		Analista-Programador	1,2	2.500,00	3.000,00
				Total RRHH	**7.200,00**

Tabla 47. Costes de Recursos Humanos

*1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas).

Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

Costes de Hardware					
Descripción	***Coste (Euro)***	***% Uso dedicado proyecto***	***Dedicación (meses)***	***Periodo de depreciación***	***Coste imputable ****
Dell XPS 8300, Intel Core i5 2500 de 3.30 GHz.	1.200,00	100	6	60	120,00
Smartphone HTC Wildfire.	120,00	100	3	60	6,00
				Total Hardware	**126,00**

Tabla 48. Costes de Hardware

* Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado.

B = periodo de depreciación (60 meses).

C = coste del equipo (sin IVA).

D = % del uso que se dedica al proyecto (habitualmente 100%).

Costes de Software				
Descripción	***Coste licencia***	***Duración licencia***	***Tiempo en uso en el proyecto (meses)***	***Coste imputable d)***
S.O Windows 7 Professional 64 bit	150,00	Indefinida	6	15,00*
Paquete Office 2010 professional	500,00	Indefinida	6	50,00*
Plataforma Android 2.2 (SDK)	0,00	Indefinida	4	0,00
IDE Eclipse Helios	0,00	Indefinida	4	0,00
			Total Software	**65,00**

Tabla 49. Costes de Software

* Puesto que las licencias adquiridas son de duración ilimitada, se ha estimado que el tiempo en que una aplicación de estas características suele quedarse obsoleta es de 5 años, y por lo tanto el porcentaje de coste imputable al proyecto seria de un 10% (correspondiente a los 6 meses empleados en la realización del mismo).

Coste Materiales Fungibles
Se estima que los costes asociados a consumo de materiales fungibles (papelería, cartuchos de tinta o toner de impresora, etc..) será de 200 Euros durante la ejecución del proyecto.

Tabla 50. Costes de materiales fungibles.

Costes Totales	
Concepto	***Presupuesto Costes Totales***
Coste Personal	7200,00
Costes Fungibles	200,00
Costes Amortización (Hw, Sw)	191,00
Costes indirectos	1518,20
Total	**9109,20**

Tabla 51. Costes totales del proyecto

6.5 Historia del proyecto

En este apartado se expondrán todas las etapas transcurridas hasta llegar a la consecución del presente proyecto, así como los tiempos empleados para la finalización de cada una de ellas.

El proyecto fue definiéndose durante el mes de Septiembre de 2011, por medio de varias reuniones con la tutora del mismo. Una vez tomada la decisión de realizar una aplicación para dispositivos móviles con plataforma *Android*, fuimos fijando los principales requisitos que conformarían la aplicación a posteriori. Durante este proceso de toma de requisitos y posteriormente, también fue llevado a cabo un proceso de documentación que comprendió los meses, entre Septiembre y Diciembre de 2011.

Tras las navidades del mismo año, ya había adquirido los conocimientos necesarios sobre la plataforma *Android* y sobre la arquitectura y los protocolos de acceso a los servicios de directorio, para comenzar con el diseño de la aplicación. Esta etapa de diseño duró aproximadamente un mes, hasta finales de Enero de 2012, y durante la misma, fueron elaborados los diagramas necesarios en el diseño de una aplicación de este tipo siguiendo la especificación *UML 2.0* [6]. Finalmente procedí a la implementación de la misma, basándome tanto en el análisis como diseño realizado en meses previos. Este proceso de implementación se extendió hasta mediados de Abril de 2012, cuando dí por finalizada la aplicación. A continuación se describe más detalladamente el trabajo realizado durante cada uno de los meses empleados en la ejecución del proyecto.

- Septiembre:
  Durante el mes de Septiembre, tuve varias reuniones con mi tutora, intentando definir el rumbo que finalmente seguiría el proyecto que tendría que realizar. En un principio, el proyecto iba a estar basado en una aplicación de directorio sobre la plataforma *Android,* para la universidad, aunque posteriormente tomamos la decisión de hacer la aplicación más dinámica, de manera que pudiese servir para el acceso a los directorios de cualquier organización. También se decidió integrar el acceso a alguna red social con el acceso a directorios, y finalmente nos decantamos por la red social y profesional *LinkedIn* ya que tiene un marcado carácter empresarial, que ofrecería la posibilidad de ampliar la búsqueda para una determinada entrada en un filtrado sobre un determinado directorio.
- Octubre y Noviembre:
  Empecé a documentarme sobre la plataforma *Android* y sobre la mejor manera de realizar un diseño para este tipo de aplicaciones móviles, así como de los patrones de diseño utilizados en las mismas. También recabé información sobre las herramientas utilizadas para su implementación, (en este caso el IDE elegido fue "*Eclipse",* por su carácter gratuito) y sobre la SDK

facilitada por *Android*, para la implementación de aplicaciones para dicha plataforma.

- Diciembre:
  El mes de Diciembre lo dedique a instalar en mi equipo de trabajo todas las herramientas necesarias para llevar a cabo la documentación del proyecto en lo referente al análisis como al diseño de la aplicación, así como el IDE de desarrollo utilizado, *Eclipse*, el SDK de *Android* y los complementos utilizados por eclipse para el desarrollo de este tipo de proyectos.
- Enero:
  En Enero, tras el fin de las navidades, y una vez preparadas todas las herramientas necesarias para la documentación del proyecto, empecé a realizar el diseño de la aplicación, tomando como referencia la especificación UML 2.0 [6] e incluyendo todos los diagramas obtenidos del mismo en el documento del proyecto.
- Febrero y Marzo:
  A finales del mes de Enero, ya había definido las pantallas de las que constaría la aplicación y el diseño de la misma estaba finalizado, así que durante los meses de Febrero y Marzo, mi dedicación casi en exclusiva fue a la implementación de estas pantallas, y a finales de Marzo la aplicación estaba acabada, a falta de realizar las pruebas pertinentes sobre cada una de las funcionalidades de la misma.
- Abril:
  Desde finales del mes de Marzo hasta mediados de Abril que sería cuando finalizaría el proyecto, me dedique a la realización de pruebas sobre la aplicación y a completar el documento del proyecto con toda la información referente a la implementación de la aplicación.

En la tabla 47 se muestra una estimación de las horas invertidas desde el comienzo del proyecto hasta su finalización en cada una de las etapas que se llevaron a cabo.

	Documentación	Análisis	Diseño	Implementación	Pruebas	Total Mensual
Septiembre	10h	0h	0h	0h	0h	**10h**
Octubre	10h	10h	0h	0h	0h	**20h**
Noviembre	40h	10h	0h	0h	0h	**50h**
Diciembre	50h	20h	0h	0h	0h	**70h**
Enero	10h	0h	30h	5h	0h	**45h**
Febrero	5h	0h	10h	50h	2h	**67h**
Marzo	0h	0h	0h	30h	2h	**32h**

Abril	20h	0h	0h	0h	5h	**25h**
Total Etapa	**145h**	**40h**	**40h**	**85h**	**9h**	**319h**

Tabla 52. Estimación coste de ejecución del PFC (horas).

7 Conclusiones y Trabajos Futuros

Este capítulo contiene las conclusiones obtenidas del desarrollo de este proyecto, tomando como referencia, el objetivo inicial que tenia el mismo, y el resultado final obtenido, a través de su proceso de desarrollo, así como los posibles trabajos futuros que se podrían llevar a cabo sobre el mismo.

7.1 Conclusiones del proyecto

Llegados a este punto, es el momento de exponer las conclusiones obtenidas, durante el desarrollo de este proyecto fin de carrera.

Los primeros pasos en la realización del proyecto consistieron en un proceso de documentación sobre las bases de la plataforma *Android*, tales como su arquitectura interna, la estructura que seguía una aplicación para este tipo de plataforma, así como las peculiaridades de una aplicación *Android* con respecto a una aplicación estándar, y otros muchos conceptos que he ido llevando a la practica durante el desarrollo de este proyecto, ya que a fin de cuentas, el objetivo del mismo era la implementación de una aplicación para dispositivos móviles. Para ello me serví de la documentación ofrecida por *Google* sobre la plataforma *Android* en su página oficial [7], así como de la amplia documentación aportada por la comunidad *Android* en internet y de algunos libros cuyas referencias se detallan en el *Anexo A*.

Durante este proceso de documentación también recopilé información sobre todo lo concerniente a los servicios de directorio y a los protocolos disponibles para el acceso a los mismos. Toda esta documentación recabada, me sirvió para consolidar la base de conocimiento que a posteriori me ayudaría a llevar a cabo el proyecto que nos atañe.

Tras la etapa de documentación, comencé un proceso de análisis sobre los requerimientos que surgieron en un primer momento, intentando abstraer todos los puntos que después definirían cada una de las funcionalidades de la aplicación y en base a dicho análisis llevé a cabo el diseño de la misma tomando como referencia la especificación *UML 2.0* [6], que me daría el soporte necesario para documentar el funcionamiento que debía implementar la aplicación *Ldapp* más adelante.

Por ultimo solo faltaba llevar a la práctica todo lo puesto en papel, así que finalmente procedí a implementar la aplicación. Este quizá resulto ser para mi el paso más sencillo en todo el proceso de elaboración del proyecto, ya que tenía algunos conocimientos sobre el lenguaje de programación de la plataforma *Android,* es decir *Java.* No obstante, la organización de una aplicación de este tipo en lo referente a la implementación de la misma difiere en algunas cosas de la de una aplicación *Java* estándar, por lo que también fue un aporte muy positivo para mí.

Como contrapunto, destacar la dificultad que entraño el proceso de Análisis, y Diseño de la aplicación dada la peculiaridad de la misma, al tratarse de una aplicación enfocada a dispositivos móviles. No obstante, esta parte del proyecto me ha aportado una experiencia de gran valor, y la posibilidad de poner en práctica muchos de los conocimientos adquiridos a lo largo de la carrera.

Así pues, la conclusión obtenida de todo el proceso de elaboración del presente proyecto fin de carrera, es la de haber obtenido la experiencia de un enfoque práctico sobre muchos de los conocimientos adquiridos a lo largo de mi etapa universitaria, que ahora finaliza, como colofón a todo el esfuerzo realizado durante estos años.

7.2 Trabajos Futuros

Este apartado, expone algunos de los trabajos que se podrían llevar a cabo en un futuro, siguiendo la línea de este proyecto fin de carrera. A continuación se exponen algunas de las tareas que podrían realizarse sobre el mismo.

- Envío masivo de email para una determinada búsqueda sobre un determinado servidor *Ldap*, dotando de esta manera al usuario de una funcionalidad de envío de circulares, publicidad, etc...
- Implementación de un *Widget* para la aplicación que permita la realización de búsquedas sobre un determinado *Ldap* desde la pantalla de inicio del móvil, permitiendo al usuario tener un acceso directo a las funcionalidades básicas de la aplicación.
- Implementación de la aplicación para otros idiomas, permitiendo así una cierta internacionalización de la misma. De esta manera, se permitiría al usuario la elección del idioma de la aplicación desde la pantalla de presentación de la misma.
- Funcionalidad que permita exportar los resultados de una determinada búsqueda sobre un determinado servidor *Ldap* a archivos del tipo ".xls", ".txt", "csv.", etc…, de esta manera, se daría la posibilidad de portar los datos obtenidos al usuario desde el dispositivo móvil a cualquier otro dispositivo, móvil o no, mediante el envío de dichos documentos vía email, *bluetooth*, etc…

8 Anexo A: Referencias

[1]"Android", Ed Brunette. Editorial Anaya, 2011.

[2]Página corporativa de "UnboundId", documentación "LDAP SDK for Java".
http://www.unboundid.com/products/ldap-sdk/docs/

[3]Tipos de licencias GNU. Último acceso Marzo 2012.
http://www.gnu.org/licenses/

[4]Información general, definiciones. Último acceso Marzo 2012
http://es.wikipedia.org/wiki/Wikipedia:Portada

[5]"System Analysis and Design", Alan Dennis, Barbara Haley Wixom, Roberta M. Roth. Editorial John Wiley & Sons.

[6]Documentación de referencia de UML 2.0,
http://www.omg.org/spec/UML/2.4.1/

[7]Web oficial de Android para desarrolladores. Ultimo acceso Marzo 2012.
http://developer.android.com/index.html

[8]Android Maps API Key Signup. Último acceso en Febrero de 2012.
http://code.google.com/intl/es-ES/android/maps-api-signup.html

[9]Social Media Integration. Ultimo acceso Enero 2012.
http://www.mobialia.com/learning/social_media_demo/

[10]LinkedIn Developers. Ultimo acceso Enero 2012.
http://developer.linkedin.com/apis

[11]Autenticación OAuth, Ultimo acceso Enero 2012.
http://www.returngis.net/2010/05/autenticacion-oauth/

[12]Historia de la plataforma Android, Ultimo acceso Abril 2012
http://www.ohmyphone.com/android/sistema-operativo-android/historia-android/

[13]Android y la fragmentación, cifras actuales emitidas por Google, Ultimo acceso Abril 2012
http://www.ohmyphone.com/android/sistema-operativo-android/historia-android/

[14]Android API Levels, Ultimo acceso Abril 2012
http://developer.android.com/guide/appendix/api-levels.html

[15]Diccionario de Informática, Ultimo acceso Abril 2012
http://www.alegsa.com.ar/Dic/dispositivo%20movil.php

[16]Gartner Research, estadísticas de venta de smartphones en 2011, Último acceso Abril 2012
http://www.gartner.com/it/page.jsp?id=1848514

[17]Cuota de mercado de Sistemas Operativos para móviles entre 2007 y 2011
http://blog.seattlepi.com/microsoft/chart-mobile-os-market-shares/

[18]Página oficial del motor de bases de datos SQLite, Último acceso Marzo 2012
http://www.sqlite.org

[19]Página oficial de WebKit, Último acceso Marzo 2012
http://www.webkit.org

[20]Página oficial sobre estándar *X.500*, Último acceso Abril 2012
http://www.x500standard.com/index.php?n=Main.HomePage

[21]Proyecto fin de carrera de *Jaime Aranaz Tudela*, de Enero 2009.

[22]Proyecto fin de carrera de *Francisco Jordán Teruel*, de Octubre 2010.

9 Anexo B: Glosario

- L.D.A.P.:
  Siglas en inglés para "*Protocolo Ligero de Acceso a Directorios*"
  (Lightweight Directory Access Protocol).
- J.2.S.E.:
  Siglas en inglés para "*Java 2 Standard Edition*", versión Java estándar para desarrollo de aplicaciones de escritorio y web.
- J.2.M.E.:
  Siglas en inglés para "*Java 2 Micro Edition*", es un subconjunto de la plataforma Java, enfocado al desarrollo de aplicaciones Java dirigidas a dispositivos con recursos restringidos, fundamentalmente a dispositivos móviles.
- A.P.I.:
  Siglas de "*Application Programming Interface*", en español Interfaz de Programación de Aplicaciones. Consiste en un conjunto de llamadas que ofrecen acceso a funciones y procedimientos, representando una capa de abstracción para el desarrollador.
- G.P.L.:
  Siglas en inglés para "*General Public License*", licencia de distribución de software que permite modificar y distribuir un programa gratuitamente, siempre y cuando se distribuya además del programa binario, el código fuente de la aplicación, permitiendo de esta manera acceder a el a otros programadores.
- L.G.P.L:
  Siglas en inglés para "*Lesser General Public License*", una versión menos restrictiva que G.P.L. Permite la inclusión de librerías de código abierto en programas de pago.
- LinkedIn:
  Sitio web orientado a negocios. Esta concebida como una red social de carácter profesional.
- Android:
  sistema operativo basado en Linux, enfocado para su uso en dispositivos móviles.
- J.N.D.I:
  Siglas en inglés para "*Java Naming and Directory Interface*" cuyo significado en español es "Interfaz de Nombrado y Directorio Java", es el API de Java para el acceso a Directorios. Proporciona una interface de búsqueda en directorios.
- S.S.L.:
  De las siglas en inglés "*Secure Sockets Layer*" o lo que es lo mismo, "capa de conexión segura", es un protocolo de cifrado que proporciona comunicaciones seguras bajo una red, comúnmente internet.
- T.L.S.:
  De las siglas en inglés "*Trasport Layer Security*" o lo que es lo mismo "seguridad de la capa de transporte", es el sucesor del protocolo de cifrado SSL, y al igual que su sucesor proporciona comunicaciones seguras bajo una red.
- H.T.T.P.:
  De las siglas en inglés "*Hypertext Transfer Protocol*" o lo que es lo mismo "protocolo de transferencia de hipertexto", es el protocolo de aplicación utilizado en internet. Define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies).
- B.B.D.D.: iniciales para "base de datos".

- D.A.O.: siglas en inglés "*Data Access Object*" o lo que es lo mismo "objeto de aceso a datos". Son las clases que contienen todos los métodos, que interactúan contra una determinada base de datos.
- R.E.S.T.: siglas en inglés "*Representational State Transfer*" o lo que es lo mismo "Transferencia de Estado Representacional", es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.
  Si bien en un principio este termino se refería a un conjunto de principios de arquitectura, en la actualidad se utiliza en el sentido más amplio para describir cualquier interfaz web simple que utiliza HTTP y XML.
- OAUTH: su nombre hace referencia a "*Open Authorization*" y es un protocolo abierto que permite autorización segura de un API de modo estándar y simple para aplicaciones de escritorio, móviles y web.
- J.S.O.N.: acrónimo de "*JavaScript Object Notation*" es un formato ligero para el intercambio de datos, su uso es similar al de XML.
- SQLite: es un sistema gestor de bases de datos, pero a diferencia de los sistemas gestores de bases de datos "cliente-servidor" convencionales, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica, sino que se enlaza con el programa, pasando a formar parte del mismo, lo que reduce la latencia de acceso a la base de datos. Los datos de dicha BBDD se guardan en un fichero estándar en la maquina host.

10 Anexo C: Aplicaciones utilizadas

- Eclipse Java EE IDE for Web Developers, versión “Helios Service Release 2”.
- SQLite Database Browser, version “2.0b1”.
- StarUML, version “5.0.2.1570”.
- MagicDraw, versión, Evaluation “16.8”.
- Paint.net, version “3.5.10”.