Universitat Oberta de Catalunya

Proyecto de Fin de Carrera - 2014

OfertaCinema

Para dispositivos móviles Android

Memoria del Proyecto	
Autor:	Marco Antonio Campos Rodríguez
Consultor:	Albert Grau Perisé
Profesor Responsable de la Asignatura:	Santi Caballe Llobet
Fecha de entrega:	**16/06/2014**

Resumen

Debido a la situación económica del país, mucha gente está dejando de ir al cine y las distribuidoras y jefes de salas exhibidoras se preguntan el por qué. Debido a este problema, algunas distribuidoras se están planteando en cerrar algunas salas, debido a la poca influencia de público.

Por otro punto, el gran problema de algunas de estas salas es el precio elevado de las entradas, ya que en algunos casos, el público ha dejado de consumir.

Por estos y otros motivos más, se ha pensado en realizar un proyecto para dispositivos móviles como teléfonos inteligentes, Tablets en la tecnología Android. Teniendo en cuenta el gran auge que está teniendo en la actualidad y la previsible tendencia a futuro de este tipo de tecnologías.

Hoy en día, Smartphones y Tablets están al alcance de cualquiera. Poco a poco están desplazando el uso de ordenadores convencionales y eso ha hecho que se haya convertido en un mercado realmente llamativo para cualquier empresa dedicada a las Tecnologías de la Información y la Comunicación (TIC). El desarrollar aplicaciones para plataformas móviles, como Android, hoy en día es una línea de negocio que muchas empresas, incluso desarrolladores por cuenta propia, están teniendo muy presente.

La idea inicial de este TFG es el desarrollo de una aplicación para dispositivos móviles sobre plataforma Android, que facilite al usuario en ver las ofertas de varias salas de cines, en donde interactuara con mucha facilidad, realizando funcionalidades como: ver las películas ofertadas de cada cine, seleccionarla butacas de la sala seleccionada, realizar la compra de los tickets.Esta aplicación estará sincronizada con un servidor que irá actualizando cada cierto tiempo las ofertas, ya sean nuevas ofertas o las mismas ofertas con diferente descuento.

La característica principal de esta aplicación será incentivar al público en ir a consumir a las salas de cine, que conllevara a obtener mayor público en los aforos de las salas.

TABLA DE CONTENIDO

Agradecimientos

A mis padres por su paciencia, su comprensión y su apoyo incondicional, que me han brindado durante todo el tiempo que me he dedicado a los estudios en la UOC.

A mis hermanas que me han ayudado mucho con el proyecto, en las labores de testing.

1. DESCRIPCION DEL PROYECTO

1.1. Introducción

El presente documento recoge toda la información del desarrollo llevado a cabo para el proyecto final de grado. El proyecto surge a partir de la baja influencia que tienen los cines, debido a la situación económica que está pasando en el país. Por ese motivo se ha pensado en el desarrollo de una aplicación en donde el usuario podrá ver las ofertas de varios cines de una manera fácil e intuitiva y realizar la compra de entradas.

Con esta introducción global, se ha pensado en desarrollar un aplicativo en la plataforma Android, debido a su mayor ratio de crecimiento en el mercado y por permitir mediante interfaces, crear todas las funcionalidades que eran necesarias para el desarrollo.

Este aplicativo llamado OfertaCinema, es una aplicación para dispositivos móviles sobre plataforma Android, que facilite al usuario en ver las ofertas de varias salas de cines, en donde interactuara con mucha facilidad, realizando funcionalidades como: ver las películas ofertadas de cada cine, seleccionar butacas de una sala seleccionada, realizar la compra de los tickets u obtener notificaciones de las nuevas ofertas.

Esta aplicación estará sincronizada con un servidor que irá actualizándose cada cierto tiempo las ofertas, ya sean nuevas ofertas o las mismas ofertas. Para ello utilizaremos las nuevas tecnologías que hay en el mercado

Más adelante analizaremos que motivaciones justifican el proyecto, así como, definiremos los objetivos generales del mismo.

1.2. Objetivo

El objetivo principal de este proyecto, es desarrollar una aplicación para dispositivos móviles Android, que facilite al usuario en poder ver todos los cines que han publicado sus ofertas para una determinada fecha, así como poder ver una breve descripción de las películas.

A estas funcionalidades se les suma la opción de realizar la compra de entradas para una película del cine seleccionado, como obtener notificaciones de las nuevas ofertas, cambios en las ofertas o la entrada de nuevos cines con sus respectivas ofertas.

Otro importante objetivo es hacer del proyecto un proyecto de software libre, en el que se pueda disfrutar de las ventajas que ofrece. Ventajas tanto en el momento del desarrollo como en el momento de la ampliación. El desarrollo debe enfocarse de forma que sea sencillo para que en un futuro se pueda ampliar con más funcionalidades.

1.2.1. Especificación de Requerimientos

Aparte de cumplir con el objetivo principal las funcionalidades mencionadas, el Trabajo de Fin de Grado debe cumplir con los siguientes requerimientos:

- El estudio del sistema operativo Android, así como sus características generales de los dispositivos móviles que lo incorporan.
- El desarrollo de un aplicativo en Java para Android que tenga las características más novedosas de los dispositivos móviles que lo incorporan.
- Ampliar conocimientos del lenguaje de programación Java, como las utilización de las herramientas SDK Android.
- La utilización de programas externos como SoapUI, que nos permite probar, simular y generar código de servicios web de forma ágil, partiendo del contrato de los mismos en formato WSDL.
- Aprender a realizar cada fase de un proyecto de esta envergadura.
- Cumplir con las fechas estimadas para cada fase de desarrollo del proyecto.
- Usar de librerías externas como ksoap2-android-assembly-3.0.0-jar-with-dependencies., android-support-v4, YouTubeAndroidPlayerApi.
- Usar proyectos externos para implementar funcionalidades como, google-play-services_lib.

1.3. Funcionalidades Principales

Esta aplicación utilizara un webservice (implementada con el mismo lenguaje de programación Java) para la obtención de datos de la base de datos. Cabe destacar que el proyecto se puede dividir en dos partes:

- Cliente: En este caso será la aplicación del dispositivo móvil. Desarrollado con el lenguaje de programación Java, utilizando el IDE Eclipse.
- Servidor: En este caso será el webservice que conectará el dispositivo móvil. Desarrollado con el lenguaje de programación Java, utilizando el IDE NetBeans.

A continuación mencionaremos las funcionalidades principales que presenta la aplicación desarrollado para dispositivos móviles Android:

- ✓ **Conexión con webservice**: Una vez realizado la descarga del aplicativo en el dispositivo móvil, como única vez se pedirá los datos del webservice para realizar la conexión con la base de datos (IP del webservice dinámica). En esta ocasión se realizará de esta manera, en caso contrario el webservice tendrá una IP estática, en la cual vendrá por defecto en el aplicativo, y se omitirá esta funcionalidad.

- ✓ **Obtención de datos**: Una vez realizada la conexión con el webservice, se procederá a la descarga de todos los datos de la base de datos (Postgres) hacia el aplicativo de dispositivo móvil.

- ✓ **Envió de datos:** Él envió de datos se realiza cuando el usuario desea realizar la acción de comprar entradas para una película proyectada de unos de los cines seleccionados. El aplicativo pedirá una serie de datos al usuario para realizar dicha funcionalidad.

- ✓ **Recibir notificaciones:** El usuario recibirá notificaciones en el dispositivo móvil, debido a las nuevas incorporaciones de cines, películas, proyecciones o bajas. Para ello, en este caso se utiliza un programa externo (SoapUI), para ejecutar las funcionalidades mencionadas.

- ✓ **Recibir email:** El usuario después de realizar una compra de entradas correctamente, el webservice notificará al usuario mediante un correo electrónico (dato obtenido en el aplicativo en la acción de comprar), la compra realizada y un fichero adjunto con los datos de la compra y un código QR, que servirá para notificarlo en el cine seleccionado.

2. PLANIFICACIÓN DEL PROYECTO

2.1. Metodología

A fin de seguir la metodología más utilizada en el mundo del desarrollo de software se ha utilizado un ciclo de vida que a continuación se describe:

- **Propuesta**: En esta fase se realiza una propuesta real en donde el consultor del aula lo estudia y valida dicha propuesta. En una propuesta establecida, el consultor puede aportar o modificar cambios sobre algunos puntos expuestos de dicha propuesta.

- **Planificación:** Para la planificación se ha tenido en cuenta las fechas de entrega de cada PAC, ya que es sumamente importante respetar las fechas entregas, a fin de evitar posibles retrasos en la elaboración del proyecto.

- **Análisis y Diseño:** En esta fase se analizará a fondo el desarrollo de la fase de Diseño del proyecto, mostrando para ello una serie de diagramas y esquemas que harán más sencillo su entendimiento.
  - ✓ Recogida de requisitos
  - ✓ Prototipo
  - ✓ Diagrama de clases y UML

- **Implementación:** Esta fase se inicia, como objetivo poner en práctica todas las características y reglas que se han ido estableciendo en las dos fases anteriores. Se describe brevemente las principales tecnologías que se utilizará en el proyecto, las librerías externas, el IDE, el servidor de la base de datos, etc. En este caso utilizamos las siguientes herramientas:
  - ✓ IDE Eclipse para el desarrollo del aplicativo
  - ✓ IDE NetBeans para el desarrollo del webservice.
  - ✓ Gestor de base de datos Postgres y SQLite para Android.
  - ✓ Librerías externas.
- **Testing:** En esta fase se realiza una serie de pruebas contra el aplicativo, siendo como perfil de un usuario.
- **Entrega:** En la entrega final del proyecto, se entrega el código fuente, la memoria correspondiente al proyecto y un video de presentación, que contendrá todas las funcionalidades que puede realizar un usuario con el dispositivo móvil.

2.2. Planificación del Proyecto

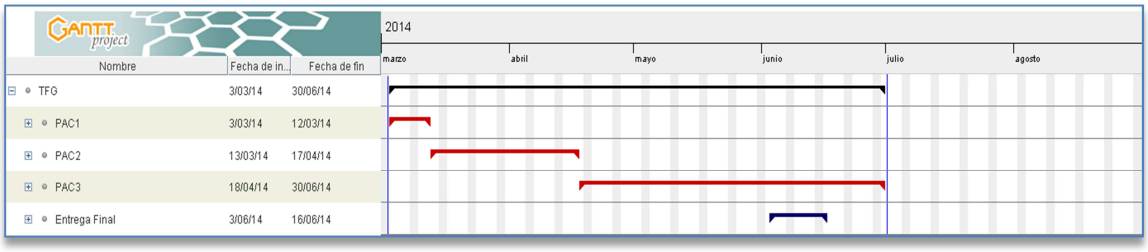
2.2.1. Hitos de Entrega

A continuación se mostrará los hitos que se entregan durante el desarrollo de este proyecto:

Documento		Fecha enunciado	Fecha de entrega
PAC1	Plan de trabajo del proyecto	03/03/2014	12/03/2014
PAC2	Análisis y diseño del proyecto.	13/03/2014	17/04/2014
PAC3	Implementación del proyecto	18/04/2014	02/06/2014
Entrega final	Memoria del proyecto. Video de presentación. Código fuente del proyecto	03/06/2014	16/06/2014

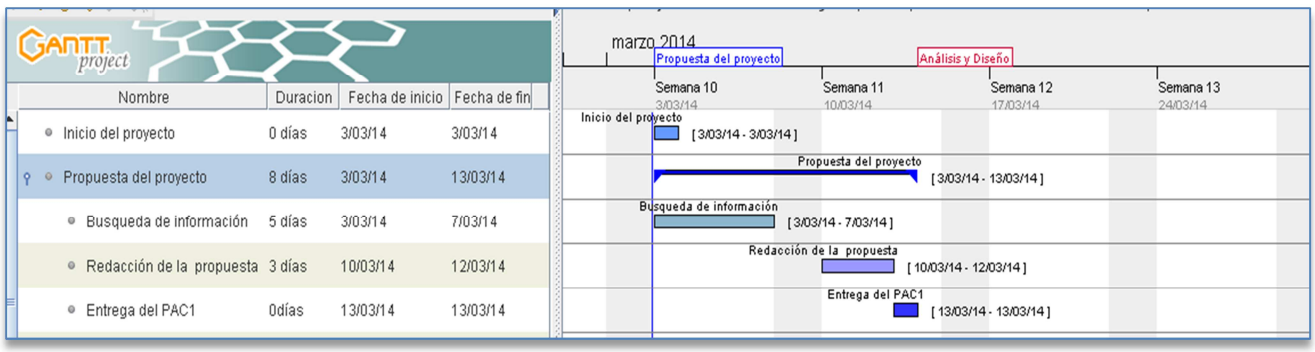
2.2.2. Diagrama de Gantt

Diagrama de Gantt con los tiempos de inicio y fin de cada PAC.

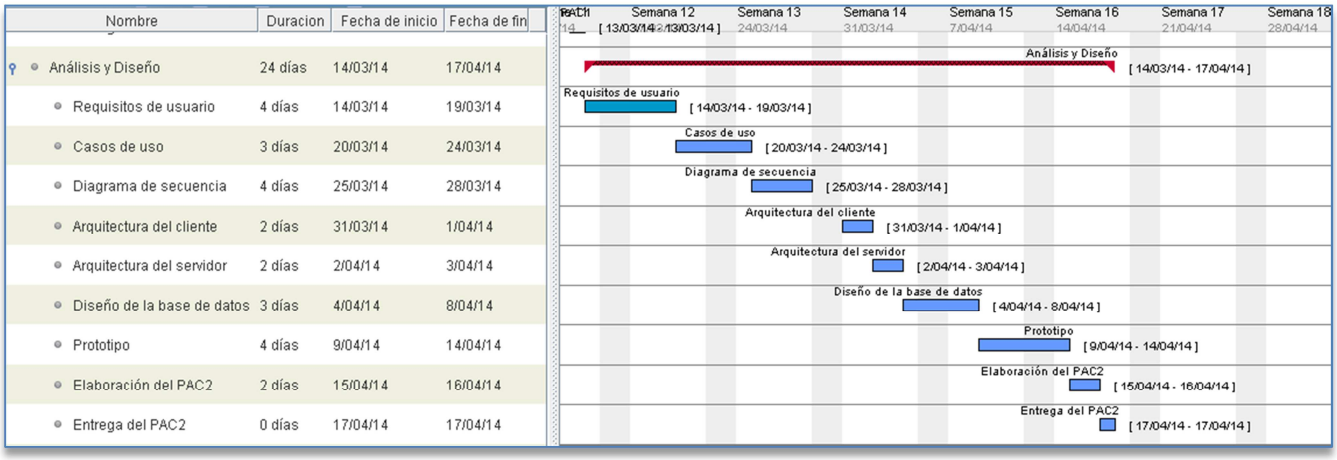


A continuación se mostrará una planificación más detallada para cada entrega:

Inicio del proyecto

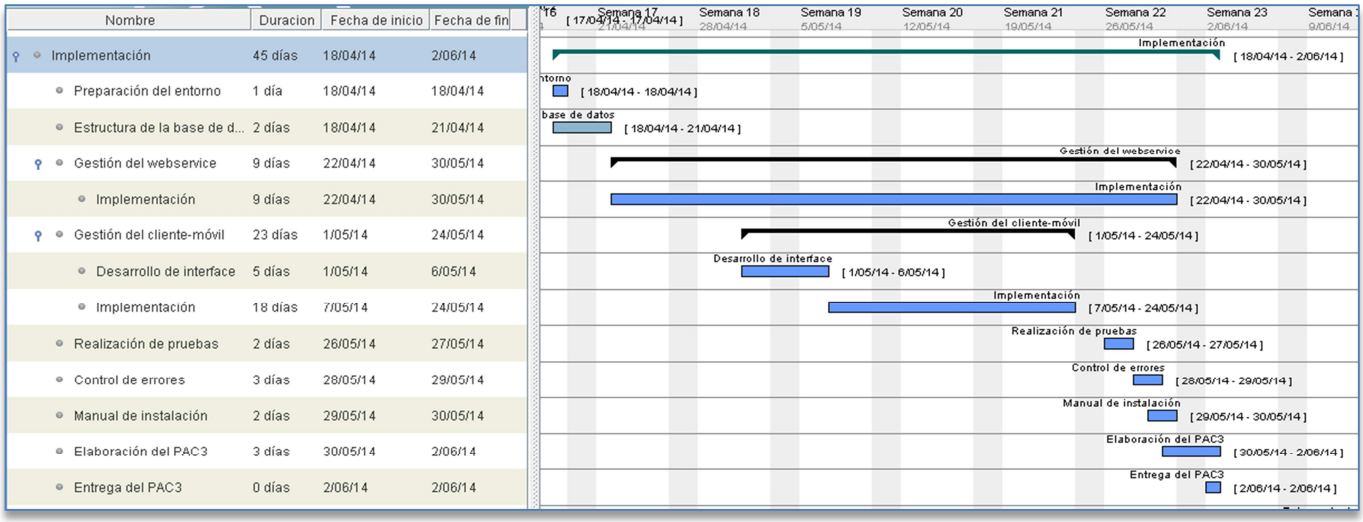


Análisis del proyecto



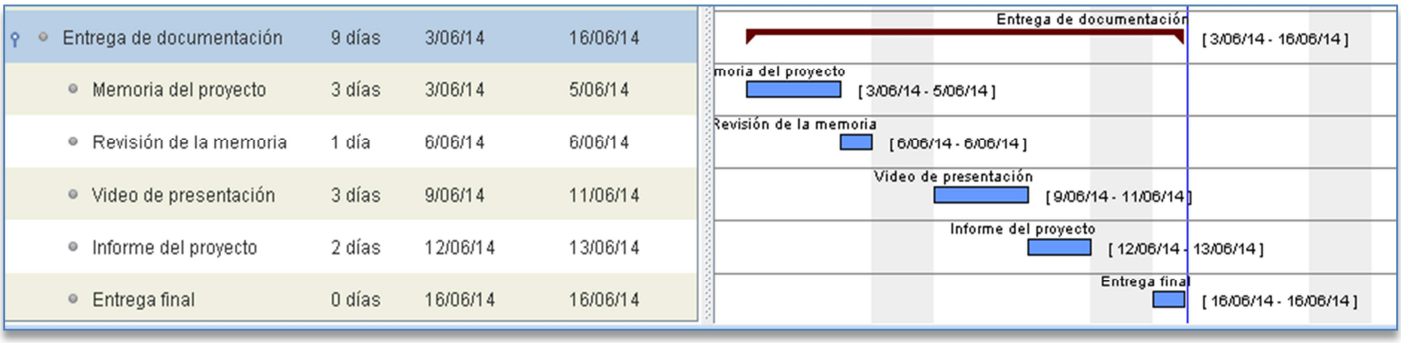
Nombre	Duracion	Fecha de inicio	Fecha de fin
Análisis y Diseño	24 días	14/03/14	17/04/14
Requisitos de usuario	4 días	14/03/14	19/03/14
Casos de uso	3 días	20/03/14	24/03/14
Diagrama de secuencia	4 días	25/03/14	28/03/14
Arquitectura del cliente	2 días	31/03/14	1/04/14
Arquitectura del servidor	2 días	2/04/14	3/04/14
Diseño de la base de datos	3 días	4/04/14	8/04/14
Prototipo	4 días	9/04/14	14/04/14
Elaboración del PAC2	2 días	15/04/14	16/04/14
Entrega del PAC2	0 días	17/04/14	17/04/14

Implementación del proyecto

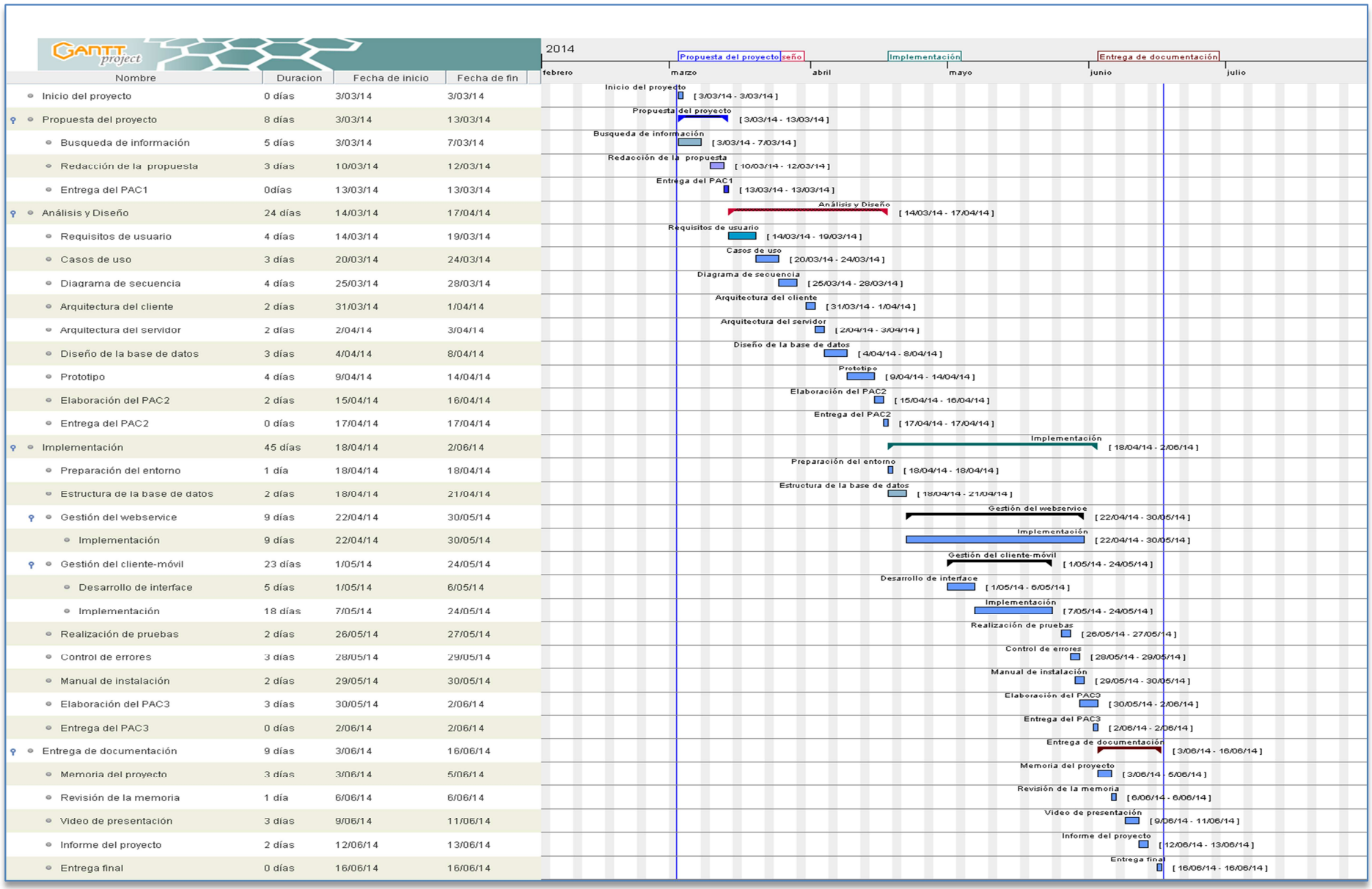


Nombre	Duracion	Fecha de inicio	Fecha de fin
Implementación	45 días	18/04/14	2/06/14
Preparación del entorno	1 día	18/04/14	18/04/14
Estructura de la base de d...	2 días	18/04/14	21/04/14
Gestión del webservice	9 días	22/04/14	30/05/14
Implementación	9 días	22/04/14	30/05/14
Gestión del cliente-móvil	23 días	1/05/14	24/05/14
Desarrollo de interface	5 días	1/05/14	6/05/14
Implementación	18 días	7/05/14	24/05/14
Realización de pruebas	2 días	26/05/14	27/05/14
Control de errores	3 días	28/05/14	29/05/14
Manual de instalación	2 días	29/05/14	30/05/14
Elaboración del PAC3	3 días	30/05/14	2/06/14
Entrega del PAC3	0 días	2/06/14	2/06/14

Entrega Final



Entrega de documentación	9 días	3/06/14	16/06/14
Memoria del proyecto	3 días	3/06/14	5/06/14
Revisión de la memoria	1 día	6/06/14	6/06/14
Video de presentación	3 días	9/06/14	11/06/14
Informe del proyecto	2 días	12/06/14	13/06/14
Entrega final	0 días	16/06/14	16/06/14


GANTT project
2014
Propuesta del proyecto
seño
Implementación
Entrega de documentación
febrero
marzo
abril
mayo
junio
julio
Nombre
Duracion
Fecha de inicio
Fecha de fin
Inicio del proyecto 0 días 3/03/14 3/03/14
Propuesta del proyecto 8 días 3/03/14 13/03/14
Busqueda de información 5 días 3/03/14 7/03/14
Redacción de la propuesta 3 días 10/03/14 12/03/14
Entrega del PAC1 0días 13/03/14 13/03/14
Análisis y Diseño 24 días 14/03/14 17/04/14
Requisitos de usuario 4 días 14/03/14 19/03/14
Casos de uso 3 días 20/03/14 24/03/14
Diagrama de secuencia 4 días 25/03/14 28/03/14
Arquitectura del cliente 2 días 31/03/14 1/04/14
Arquitectura del servidor 2 días 2/04/14 3/04/14
Diseño de la base de datos 3 días 4/04/14 8/04/14
Prototipo 4 días 9/04/14 14/04/14
Elaboración del PAC2 2 días 15/04/14 16/04/14
Entrega del PAC2 0 días 17/04/14 17/04/14
Implementación 45 días 18/04/14 2/06/14
Preparación del entorno 1 día 18/04/14 18/04/14
Estructura de la base de datos 2 días 18/04/14 21/04/14
Gestión del webservice 9 días 22/04/14 30/05/14
Implementación 9 días 22/04/14 30/05/14
Gestión del cliente-móvil 23 días 1/05/14 24/05/14
Desarrollo de interface 5 días 1/05/14 6/05/14
Implementación 18 días 7/05/14 24/05/14
Realización de pruebas 2 días 26/05/14 27/05/14
Control de errores 3 días 28/05/14 29/05/14
Manual de instalación 2 días 29/05/14 30/05/14
Elaboración del PAC3 3 días 30/05/14 2/06/14
Entrega del PAC3 0 días 2/06/14 2/06/14
Entrega de documentación 9 días 3/06/14 16/06/14
Memoria del proyecto 3 días 3/06/14 5/06/14
Revisión de la memoria 1 día 6/06/14 6/06/14
Video de presentación 3 días 9/06/14 11/06/14
Informe del proyecto 2 días 12/06/14 13/06/14
Entrega final 0 días 16/06/14 16/06/14
Inicio del proyecto [3/03/14 - 3/03/14]
Propuesta del proyecto [3/03/14 - 13/03/14]
Busqueda de información [3/03/14 - 7/03/14]
Redacción de la propuesta [10/03/14 - 12/03/14]
Entrega del PAC1 [13/03/14 - 13/03/14]
Análisis y Diseño [14/03/14 - 17/04/14]
Requisitos de usuario [14/03/14 - 19/03/14]
Casos de uso [20/03/14 - 24/03/14]
Diagrama de secuencia [25/03/14 - 28/03/14]
Arquitectura del cliente [31/03/14 - 1/04/14]
Arquitectura del servidor [2/04/14 - 3/04/14]
Diseño de la base de datos [4/04/14 - 8/04/14]
Prototipo [9/04/14 - 14/04/14]
Elaboración del PAC2 [15/04/14 - 16/04/14]
Entrega del PAC2 [17/04/14 - 17/04/14]
Implementación [18/04/14 - 2/06/14]
Preparación del entorno [18/04/14 - 18/04/14]
Estructura de la base de datos [18/04/14 - 21/04/14]
Gestión del webservice [22/04/14 - 30/05/14]
Implementación [22/04/14 - 30/05/14]
Gestión del cliente-móvil [1/05/14 - 24/05/14]
Desarrollo de interface [1/05/14 - 6/05/14]
Implementación [7/05/14 - 24/05/14]
Realización de pruebas [26/05/14 - 27/05/14]
Control de errores [28/05/14 - 29/05/14]
Manual de instalación [29/05/14 - 30/05/14]
Elaboración del PAC3 [30/05/14 - 2/06/14]
Entrega del PAC3 [2/06/14 - 2/06/14]
Entrega de documentación [3/06/14 - 16/06/14]
Memoria del proyecto [3/06/14 - 5/06/14]
Revisión de la memoria [6/06/14 - 6/06/14]
Video de presentación [9/06/14 - 11/06/14]
Informe del proyecto [12/06/14 - 13/06/14]
Entrega final [16/06/14 - 16/06/14]

3. OBJETIVO DEL PROYECTO

3.1. Objetivos Generales

a) Hacer fácil e intuitivo las funcionalidades definidas para el usuario.
b) Incentivar al público en ir a las salas de cine, esto supondrá, que el aplicativo sea más conocido sobre otras aplicaciones.
c) Cumpla con las características más utilizadas en los aplicativos que ya existen.

3.2. Objetivos Específicos

a) Desarrollar en lenguaje Java, un aplicativo para dispositivos móviles Android, siguiendo la misma estructura e interface de las aplicaciones que existen en el mercado.
b) Desarrollar en el lenguaje Java un proyecto webservice para la comunicación con el aplicativo Android.

3.3. Objetivos Didácticos

Aprendizaje o profundización en las siguientes tecnologías de desarrollo de software:

a) Lenguaje de programación Java en el IDE Eclipse Indigo SR2.
b) Utilización de las herramientas proporcionadas por el SDK Android.
c) Utilización del servicio web SOAP para la comunicación entre el aplicativo y el webservice.
d) Lenguaje de programación Java utilizando el IDE NetBeans 7.4
e) Utilización de un servicio webservice en NetBeans.
f) Lenguajes de consultas para atacar a la base de datos en PostgreSQL 9.3
g) Utilización de la herramienta hibernate para el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación.
h) Utilización de algunos servicios externos como: servicio de YouTube, servicio de google maps.

3.4. Alcance del Proyecto

Para el desarrollo de este proyecto, se ha elegido como lenguaje de programación, Java y como IDEs, Eclipse y Netbeans, debido a los conocimientos adquiridos en la UOC, como en la experiencia profesional. El proyecto se desarrollará siguiendo las pautas establecidas por el consultor.

Cada fase del proyecto será remitida en los Pacs correspondientes y validados por el propio consultor. A medida que se vaya entregando los Pacs, el desarrollo del proyecto va evolucionando, a su vez, se va adquiriendo nuevos conocimiento logrando así el objetivo principal.

El proyecto se considera finalizado cuando se haya entregado el Pac4 o entrega final y en donde se haya obtenido una proyecto cliente – servidor, en donde el cual se podrá realizar todas las funcionalidades establecidas.

4. ANÁLISIS DE DISEÑO

En este punto se estudiará los requisitos de usuario de la aplicación así como los diferentes casos de uso que se pueden dar. Con ello se pasará a la fase de diseño del proyecto entrando en la parte de la arquitectura del mismo. Y por último se comentarán los aspectos más relevantes de la implementación de la aplicación.

4.1. Análisis

4.1.1. Requisitos de Usuario

A continuación se especificarán los requisitos de usuario, para ello se seguirá el siguiente patrón:

- **Identificador**: Se seguirá la siguiente nomenclatura RE_OFERTCINE_XXX, siendo "XXX" un número de serie,
- **Descripción**: Descripción del requisito
- **Necesidad**: Establece la necesidad del requisito dentro del proyecto. Tendrá como valores del 1 al 5, siendo el 1 la prioridad más baja y el 5 la prioridad más alta.
- **Prioridad**: Establece la prioridad del requisito dentro del proyecto. Tendrá como valores del 1 al 5, siendo el 1 la prioridad más baja y el 5 la prioridad más alta.
- **Estabilidad**: Especifica la sensibilidad del requisito a ser modificado, tomando valores entre "Si" y "No".

A continuación se muestra una serie de requisitos, siguiendo el mismo patrón mencionado anteriormente.

Conexión con Webservice	
Identificador	RE_OFERTCINE_001 - Conexión con Webservice
Descripción	El usuario deberá insertar unos parámetros para realizar la conexión con el webservice (esto se realizará la primera vez que ingrese al aplicativo).
Necesidad	5
Prioridad	5
Estabilidad	Si

En el requisito RE_OFERTCINE_001 - Conexión con Webservice, solo se utilizará hasta que el webservice este alojado en sitio web, así los parámetros de conexión vendrán por defecto en la aplicación.

Listado de Cine	
Identificador	RE_OFERTCINE_002 - Listado de Cine
Descripción	El usuario debe ser capaz de visualizar un listado de cines con sus respectivas ofertas.
Necesidad	5
Prioridad	5
Estabilidad	Si

Listado de películas	
Identificador	RE_OFERTCINE_003 - Listado de películas
Descripción	El usuario debe de poder a acceder al listado de películas de cualquier cine seleccionado.
Necesidad	5
Prioridad	5
Estabilidad	Si

Descripción de una película	
Identificador	RE_OFERTCINE_004 - Descripción de una película
Descripción	El usuario debe de poder a acceder a una breve descripción de la película seleccionada.
Necesidad	4
Prioridad	3
Estabilidad	Si

Tráiler de la película	
Identificador	RE_OFERTCINE_005 - Ver tráiler de una película
Descripción	El usuario debe de poder a acceder a ver el tráiler de la película seleccionada.
Necesidad	4
Prioridad	3
Estabilidad	Si

Localización del cine	
Identificador	RE_OFERTCINE_006 – Localización del cine
Descripción	El usuario debe de poder a acceder a la localización del cine seleccionado.
Necesidad	4
Prioridad	3
Estabilidad	Si

Ver sala	
Identificador	RE_OFERTCINE_007 – Visualizar sala
Descripción	El usuario debe de poder a acceder a visualizar la sala en donde se proyecta la película(existe una variedad de salas)
Necesidad	5
Prioridad	5
Estabilidad	Si

Ver datos seleccionados	
Identificador	RE_OFERTCINE_008 – Visualizar los datos seleccionados
Descripción	El usuario debe de poder a acceder a visualizar la todos los datos seleccionados, antes de realizar la compra de entradas.
Necesidad	5
Prioridad	5
Estabilidad	Si

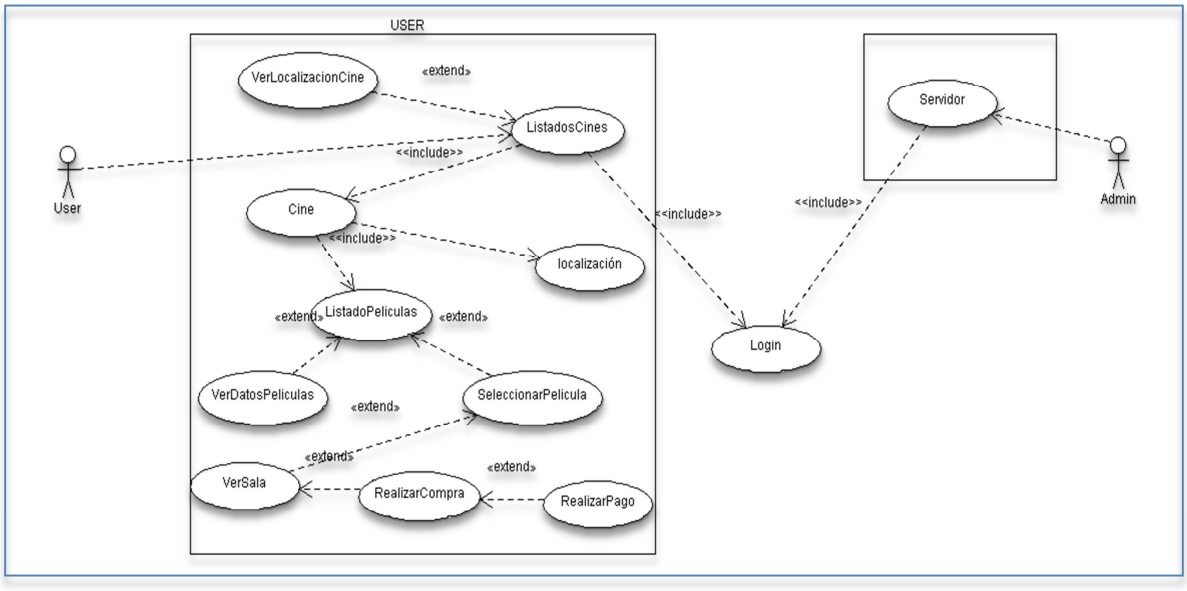
Realizar compra	
Identificador	RE_OFERTCINE_009 – Realizar la compra
Descripción	El usuario debe de poder realizar la compra de las entradas seleccionadas.
Necesidad	5
Prioridad	5
Estabilidad	Si

Recibir notificaciones	
Identificador	RE_OFERTCINE_010 – Recibir notificaciones
Descripción	El usuario debe de poder recibir notificaciones de las nuevas ofertas que se publican.(El aviso se dará como una notificación hacia el dispositivo móvil)
Necesidad	5
Prioridad	5
Estabilidad	Si

Recibir email	
Identificador	RE_OFERTCINE_011 – Recibir email
Descripción	El usuario debe de poder recibir un email a su correo electrónico (insertado en el aplicativo), con los datos relacionados a la compra. Este email tendrá un adjunto que servirá como constancia de la compra
Necesidad	5
Prioridad	5
Estabilidad	Si

4.1.2. Diagrama de Caso de Uso

En el diagrama de casos de uso, se puede observar, las diferentes funcionalidades que puede realizar un usuario en el aplicativo. En la aplicación OfertaCinema se podrá visualizar un listado de todos los cines que han publicado sus ofertas, ver la localización del cine, una breve descripción de la película seleccionada, ver el tráiler, ver diferentes tipos de sala, realizar la compra de entradas y realizar el pago de las butacas seleccionadas de una oferta seleccionada.



Caso de uso general

A continuación se especificarán los diagramas de cada caso de uso, para ello se seguirá el siguiente patrón:

- **Título**: Nombre del caso de uso
- **Actor**: Agente externo que interactúa con el sistema participando en el caso de uso.
- **Escenario**: Especificación detallada de los pasos que realiza el usuario dentro del escenario.
- **Post-condición**: Condiciones que son el resultado de la ejecución del caso de uso.
- **Pre-condición**: Condiciones que deben darse para la realización del caso de uso.

A continuación se muestra una serie de casos de usos, siguiendo el mismo patrón mencionado anteriormente.

- **Caso de uso para el administrador**, en este caso será el propio usuario.

Título:	Registrar
Actor:	Administrador
Pre-condición:	El usuario se descarga la aplicación en el dispositivo móvil
Escenario:	
1.-El usuario inserta la dirección del webservice y el password. 2.-El sistema verifica si la dirección es correcta. 2.1.- En caso de que los datos fueran incorrectos, la aplicación mostrará un aviso de error. 3.-El sistema confirma los datos insertados y descarga todos los datos del servidor.	
Post-condición:	El usuario ha insertado los datos del webservice correctamente.

- **Casos de uso para el usuario**

Título:	Lista de Cines
Actor:	Usuario
Pre-condición:	El usuario ingresa a la aplicación, que previamente ha obtenido todos los datos.
Escenario:	
1.-El usuario busca en el dispositivo móvil el icono que corresponde a la aplicación. 2.-El usuario pulsa sobre el icono para abrir la aplicación. 2.1.- En caso de que no muestre ningún listado, la aplicación mostrará un error de configuración. 3.-Se inicia la carga del listado de los cines.	
Post-condición:	El usuario visualiza un listado con todos los cines que han publicado ofertas.

Título:	Lista de películas por cada cine
Actor:	Usuario
Pre-condición:	El usuario ingresa a la aplicación, anteriormente ha obtenido todos los datos.
Escenario:	
1.-Caso de uso Lista de Cines 2.-El usuario pulsa sobre el cine que desea ver las películas que oferta. 2.1.- En caso de que no muestre ningún listado, la aplicación mostrará un mensaje de que no hay ofertas. 2.2.-En caso de que muestre películas en las que no se pueden seleccionar, será por el motivo de que se ha superado la fecha límite de su oferta. 3.-Se inicia la carga del listado de las películas.	
Post-condición:	El usuario visualiza un listado con todos las películas que ha publicado el cine seleccionado

Título:	Seleccionar Película
Actor:	Usuario
Pre-condición:	El usuario ingresa a la aplicación, que previamente ha obtenido todos los datos.
Escenario:	
1.-Caso de uso Lista de Cines 2.- Lista de películas por cada cine 3.-Seleccionamos la película a mostrar. 4.-La aplicación mostrará una breve descripción de la película seleccionada. 4.1.-Mostrará una pestaña de sesión, en donde se podrá seleccionar la sesión y la fecha de la película. 4.2.-Mostrará una pestaña de info con una breve descripción de la película. 4.3.-Mostrará una pestaña de lugar, en donde mostrara la geolocalización del cine seleccionado.	
Post-condición:	El usuario visualiza la película seleccionada que ha publicado el cine

Título:	Seleccionar Asiento
Actor:	Usuario
Pre-condición:	El usuario ingresa a la aplicación, anteriormente ha obtenido todos los datos.
Escenario:	
1.-Caso de uso Lista de Cines 2.- Caso de uso Lista de películas por cada cine 3.- Caso de uso Seleccionar Película. 4.-En la pestaña sesión: 4.1.- Seleccionamos la sesión del combobox. 4.1.1-Si no selecciona ninguna sesión, la aplicación mostrara un aviso. 4.2.-Selecionamos la fecha, pulsando el botón elegir fecha. 4.2.1.-Si no seleccionamos ninguna fecha, la aplicación mostrara un aviso. 4.2.1.-Si seleccionamos una fecha que no corresponde a la oferta válida, la aplicación mostrara un aviso. 5.-La aplicación muestra la sala con sus asientos correspondientes.	
Post-condición:	El usuario visualiza la sala en donde se proyecta la película seleccionada.

Título:	Realizar compra
Actor:	Usuario
Pre-condición:	El usuario ingresa a la aplicación, que previamente ha obtenido todos los datos.
Escenario Principal:	
1.-Caso de uso Lista de Cines 2.- Caso de uso Lista de películas por cada cine 3.- Caso de uso Seleccionar Película. 4.- Seleccionar Asiento 5.-Pulsamos el botón comprar 5.1.-Si no seleccionamos ningún asiento, la aplicación mostrará un aviso. 6.-La aplicación muestra una breve información de los datos seleccionados anteriormente.	
Post-condición:	El usuario visualiza la compra que ha realizado antes de realizar el caso de uso de pago.

Título:	Realizar pago
Actor:	Usuario
Pre-condición:	El usuario ingresa a la aplicación, que previamente ha obtenido todos los datos.
Escenario Principal:	
1.-Caso de uso Lista de Cines 2.- Caso de uso Lista de películas por cada cine 3.- Caso de uso Seleccionar Película. 4.- Caso de uso Seleccionar Asiento 5.- Caso de uso Realizar compra 6.-Pulsamos el botón siguiente. 6.1-La aplicación mostrará un aviso de información. 6.2.-Desea realizar la compra. 6.2.1.-Pulsamos si, nos enviará a la pantalla de realizar el pago 6.2.3.-Pulsamos no, nos cerrara el aviso. 7.-Se inserta los datos personales. 7.1.-Si no se inserta los datos correctos, la aplicación mostrará un aviso al pulsar el botón comprar. 8.-Se inserta los datos del tipo de pago 8.1.-Si no se inserta los datos correctos, la aplicación mostrará un aviso al pulsar el botón comprar. 9.-Pulsamos el botón comprar, la aplicación nos mostrará un aviso de información. 10.-.Acepta la compra 10.1.-Pulsamos si, nos enviará a la pantalla de casos de uso Lista de Cines 10.2.-Pulsamos no, nos cerrara el aviso.	
Post-condición:	El usuario realiza el pago de la compra realizada anteriormente

4.1.3. Diagrama de Clases

Diagrama de clases correspondiente al webservice

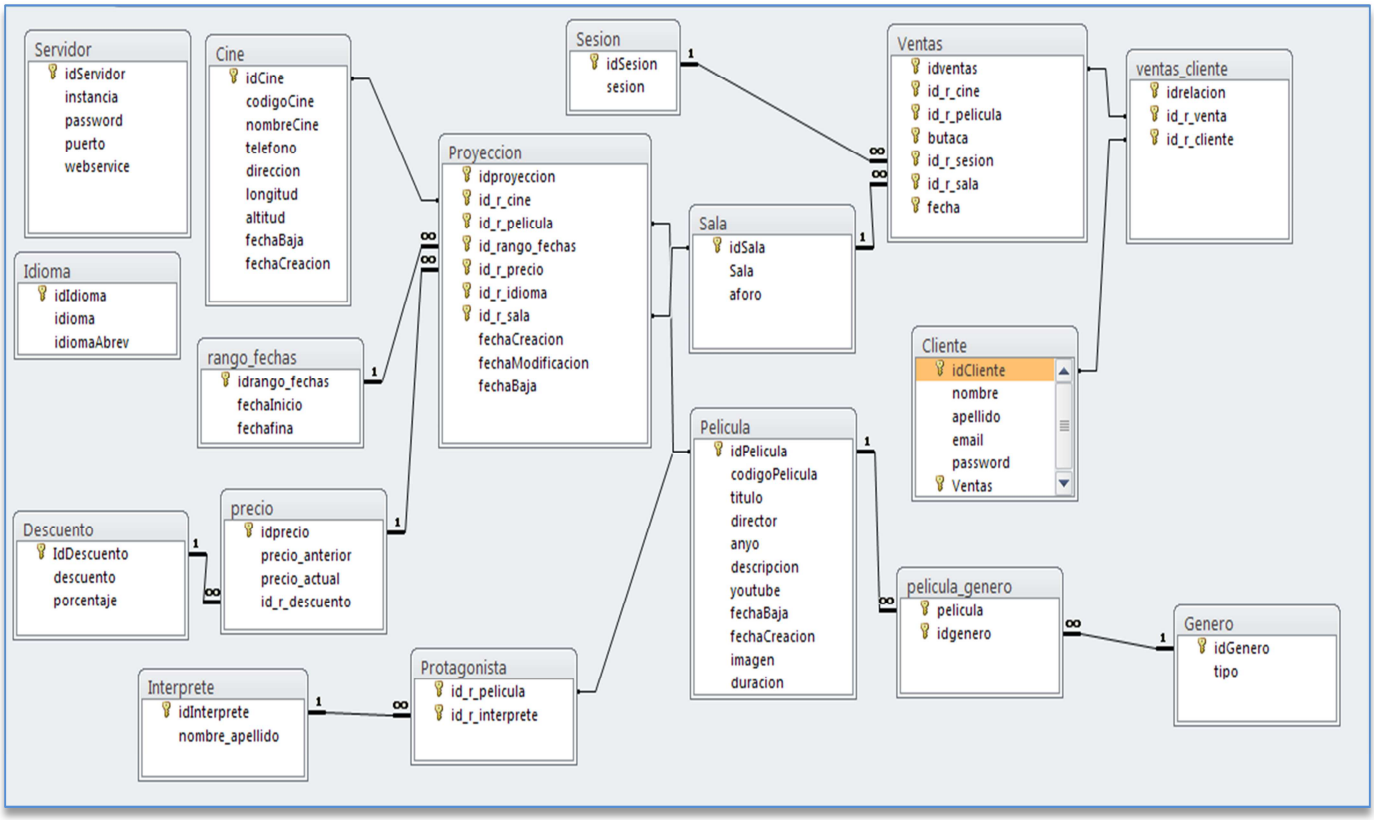
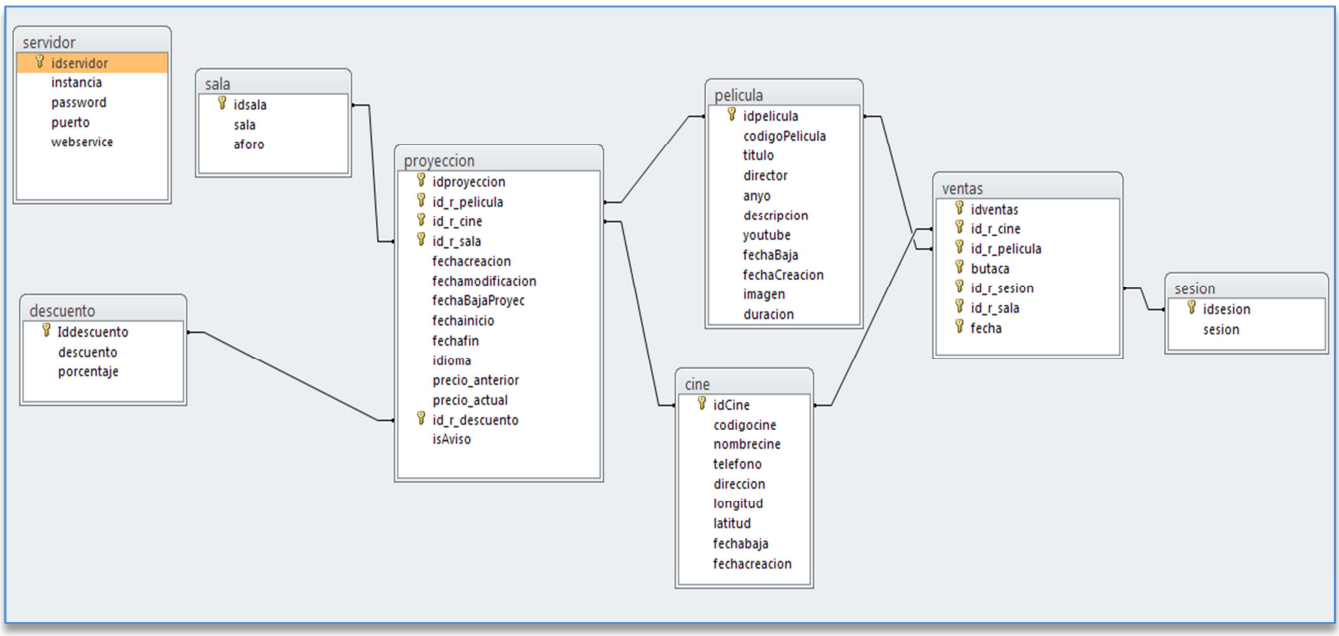


Diagrama de clases correspondiente al cliente.



4.1.4. Diagrama de secuencia

A continuación se especificarán los diagramas de secuencia de cada caso de uso:

Diagrama de secuencia: Conexión con el webservice

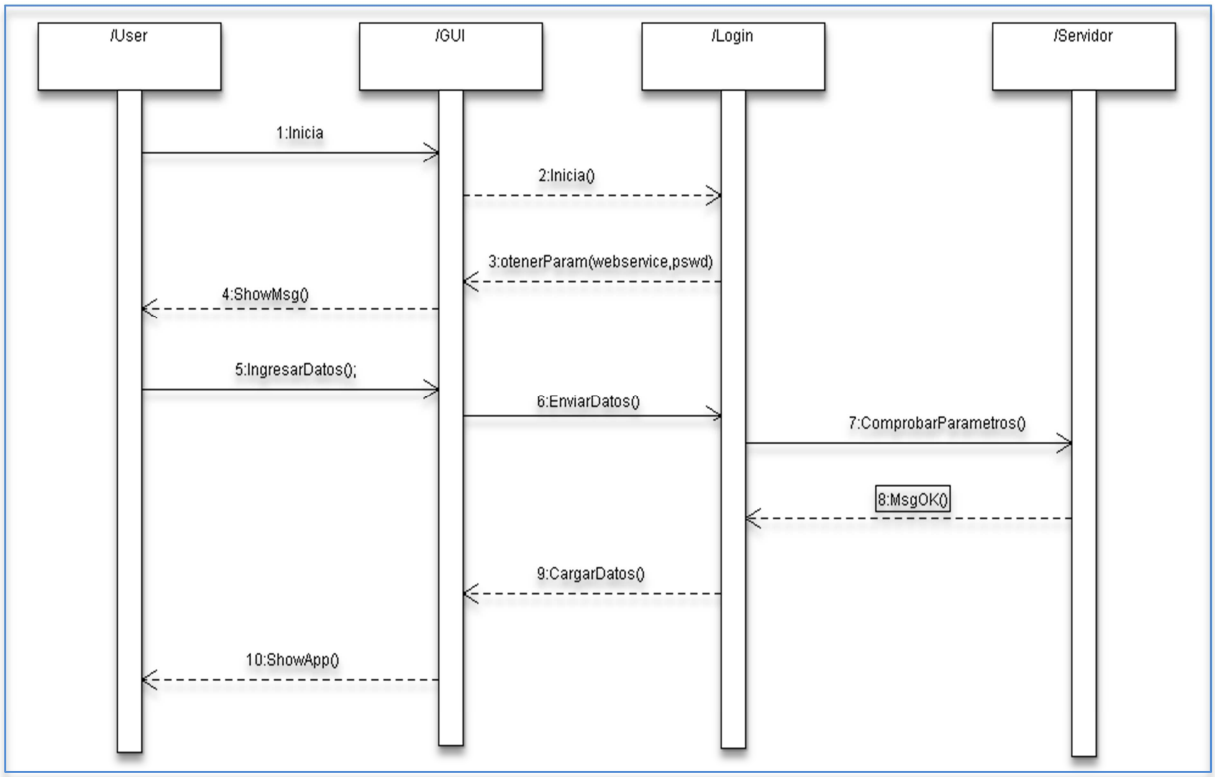


Diagrama de secuencia: Listar Cines

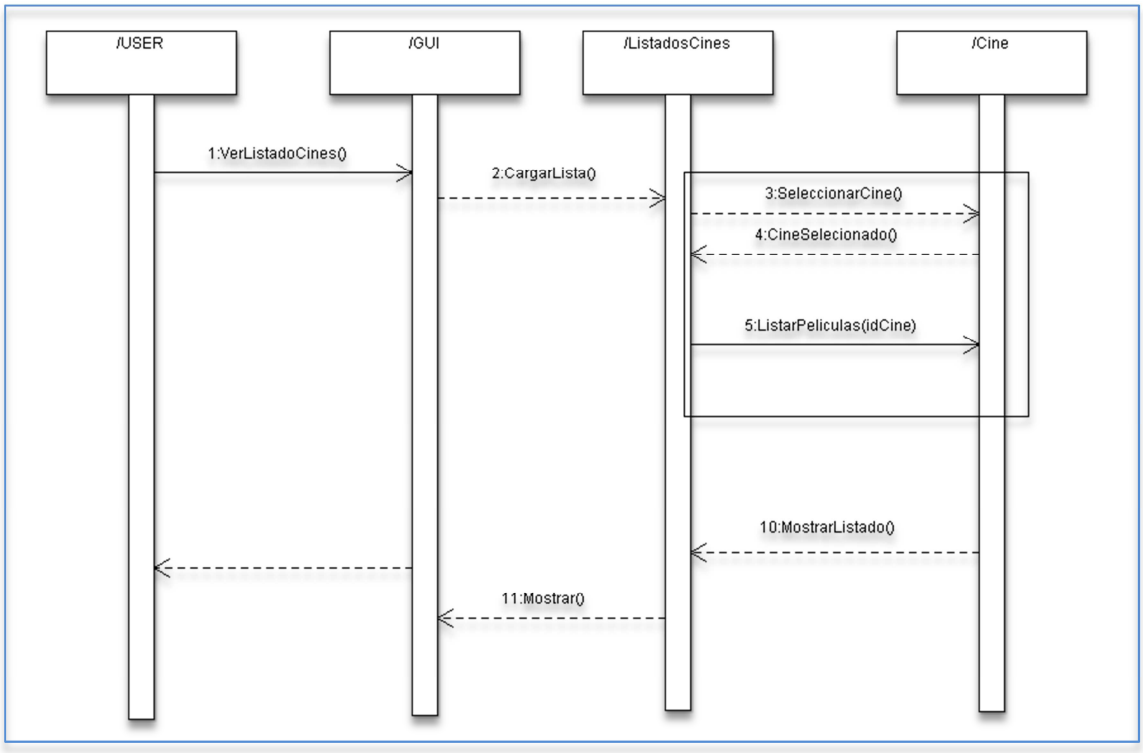


Diagrama de secuencia: Lista películas del cine seleccionado

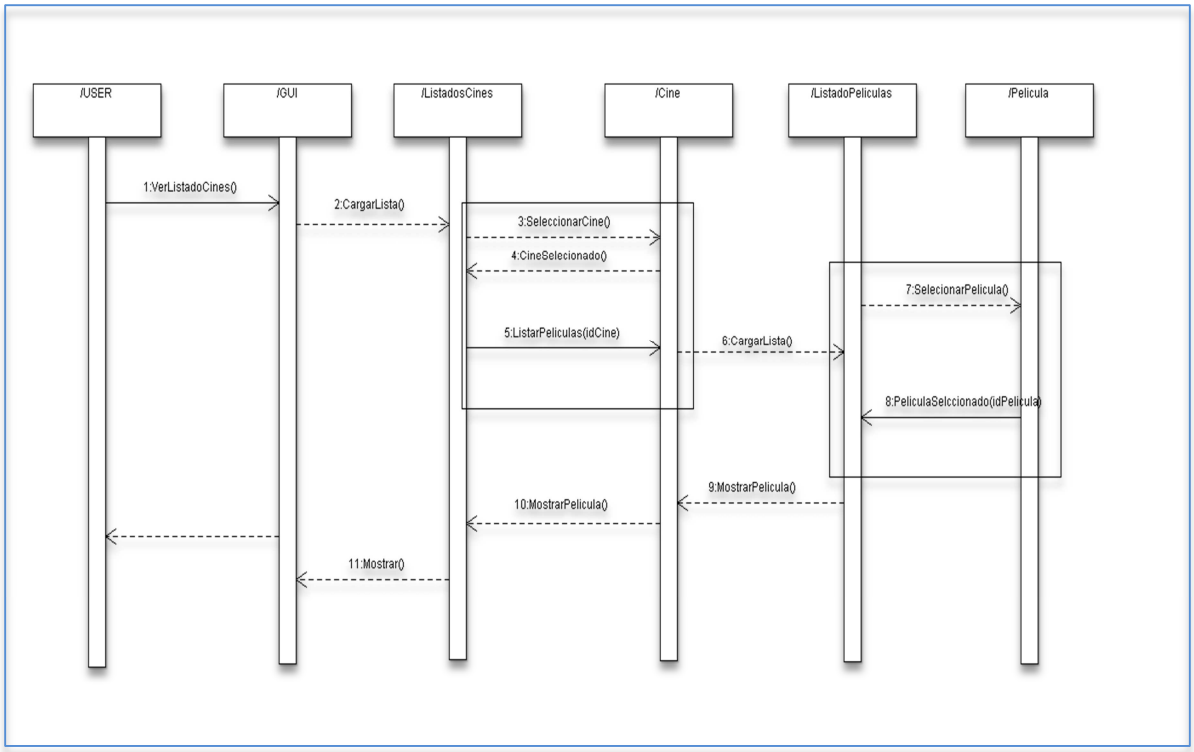


Diagrama de secuencia: Seleccionar Asiento

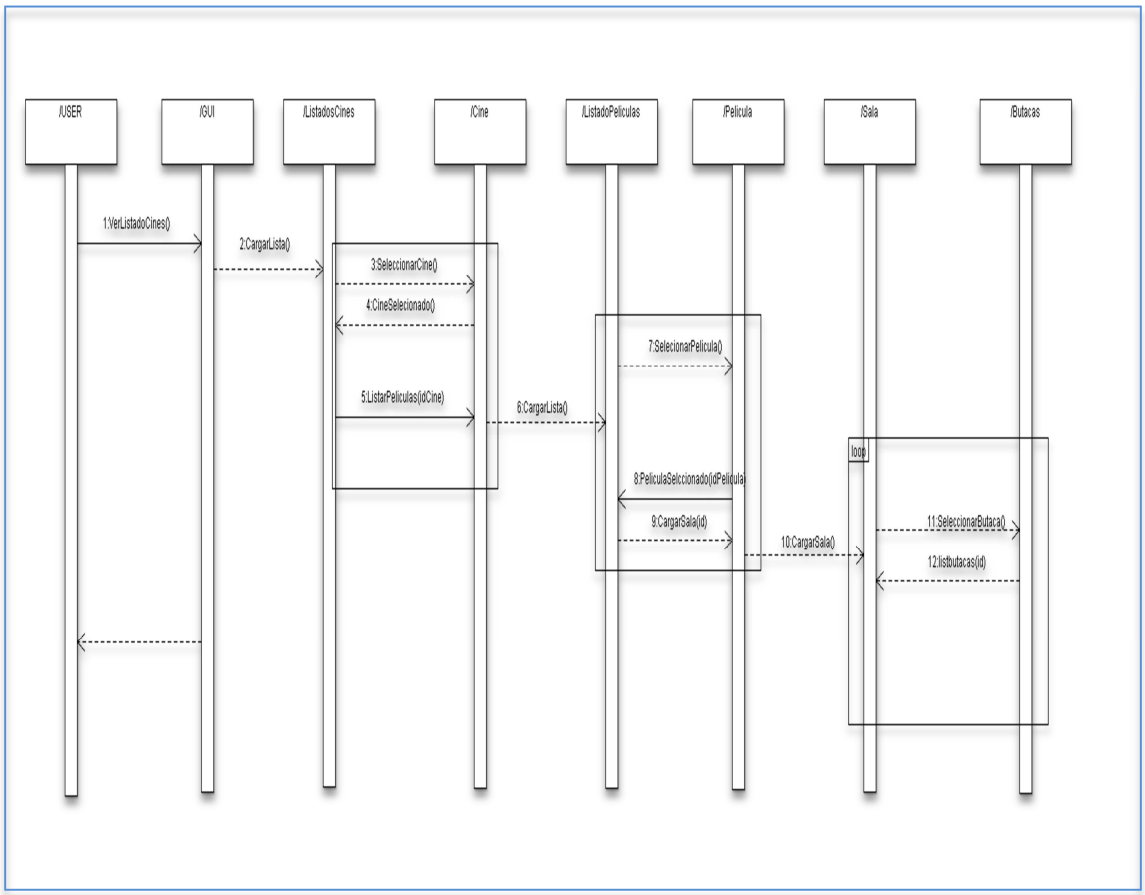
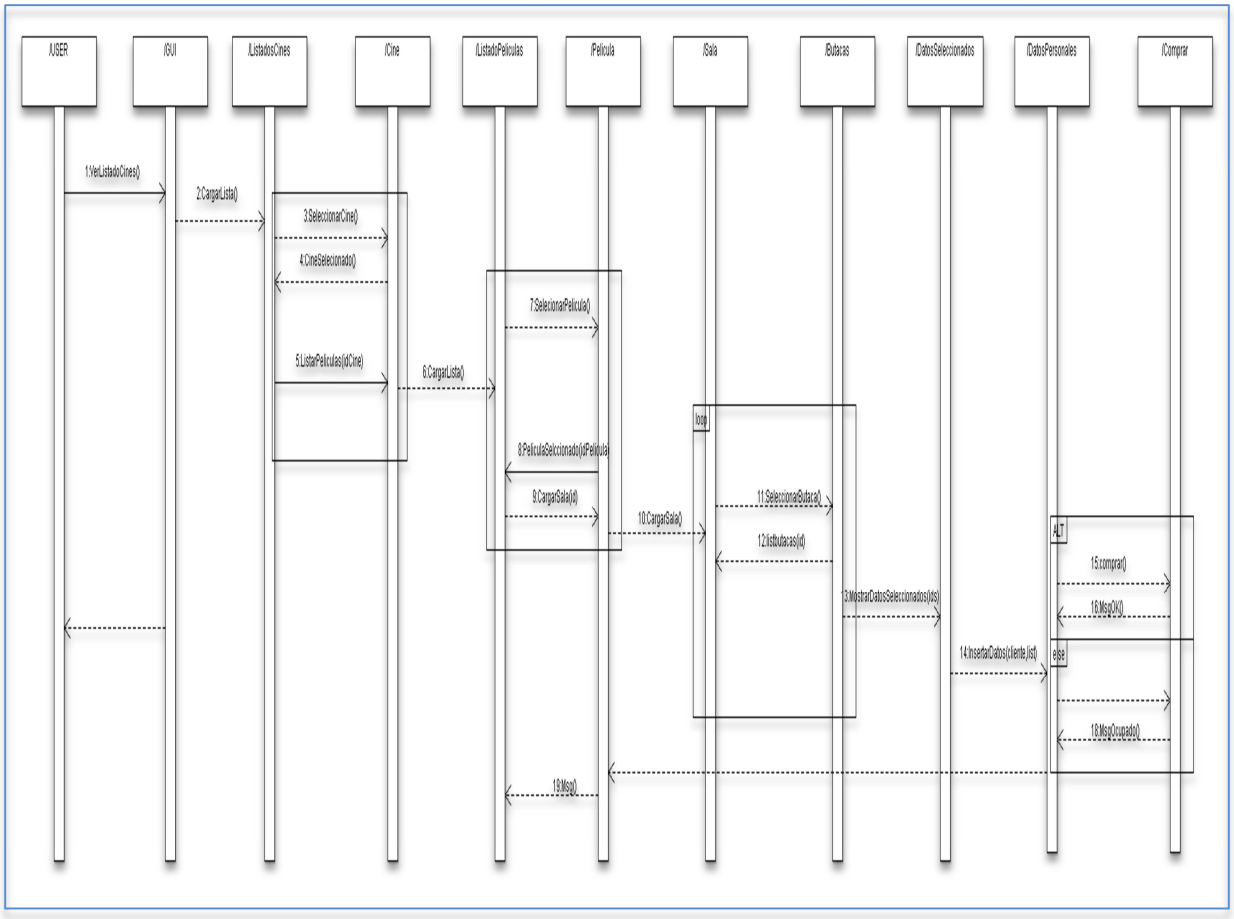


Diagrama de secuencia: Realizar compra



4.1.5. Arquitectura del Sistema

El sistema constará de dos partes que son cliente y servidor:

- **Cliente**: Los requerimientos que se ha utilizado para su implementación, es la tecnología JAVA con el SDK Android y como base de datos SQLlite.

- **Servidor**: Los requerimientos que se ha utilizado para su implementación, se limita a la utilización en el lenguaje de programación, utilizando el framework Hibernate y para la base de datos utilizamos el gestor PostgreSQL.

4.1.5.1. Arquitectura del hardware

- **Aplicación Servidor:** Se ha utilizado un equipo Intel Core 2 DUO o superior ya que la carga del IDE NetBeans consume muchos recursos. Con un mínimo de 3G de memoria RAM y como S.O. Windows XP Profesional.

- **Aplicación Cliente:** Para las pruebas iniciales, se ha utilizado el emulador por defecto del SDK Android y para las pruebas con más funcionalidades, se ha utilizado el dispositivo móvil Sony XPERIA Z1 y esporádicamente el SAMSUNG NOTE 3

4.1.5.2. Arquitectura del software

- **Aplicación Servidor:** Para la parte del servidor se utilizará el IDE NetBeans 7.4, utilizando el Framework Hibernate y como servidor de aplicaciones apache-tomcat-6.0.39.Ademas se incluirá algunas librerías externas como:
  - Directorio tomcat.
  - apache-collections-commons-collections-3.1.jar
  - barcode.jar
  - dom4j-1.6.1-sources.jar
  - hibernate3.jar
  - itextpdf-5.3.2.jar
  - jasypt-1.9.0-lite.jar
  - javassist-3.4.ga.jar
  - jta-1.1.jar
  - postgresql-9.0-802.jdbc4.jar

  Como anteriormente se ha comentado, el sistema de gestor de la base de datos será PostgreSQL 9.3.

- **Aplicación Cliente:** Para la parte del cliente se utilizará el IDE Eclipse, versión Indigo Service Release 2.

  El plugin Android Developer Tools (*ADT*), en este caso el plugin Android 4.2.2.

  Como base de datos se utilizará SQLite.

  Se utilizará librerías externas como:
  - ksoap2-android-assembly-3.0.0-jar-with-dependencies.jar
  - android-support-v4.jar
  - YouTubeAndroidPlayerApi.jar
  - Generar el KEY de google para activar las funcionalidades de video y googlemaps.
  - Importar el proyecto google-play-services_lib

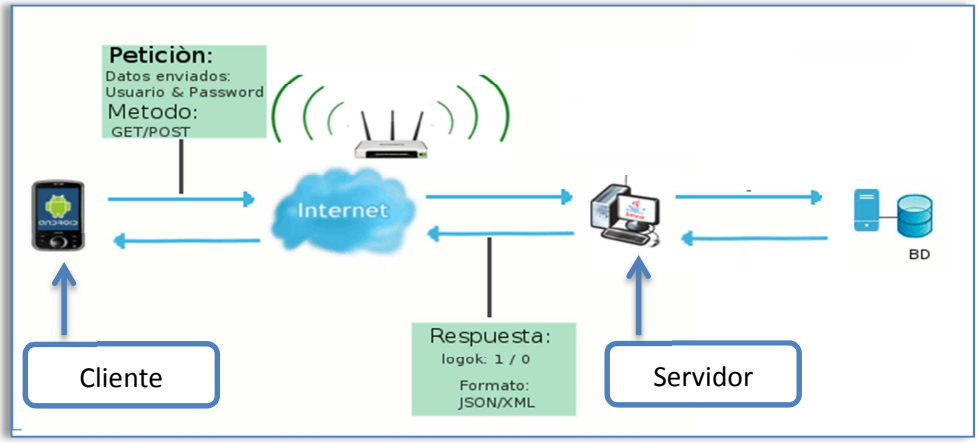
4.2. Diseño

Después de haber realizado un análisis de los requisitos y establecer las funcionalidades de la aplicación, se debe definir un diseño de la misma dentro del ciclo de desarrollo de software. Para ello se describirá de manera detallada un diseño de la arquitectura de la aplicación, un diseño de la base de datos y se comentarán los diferentes diseños realizados para la interfaz de usuario.

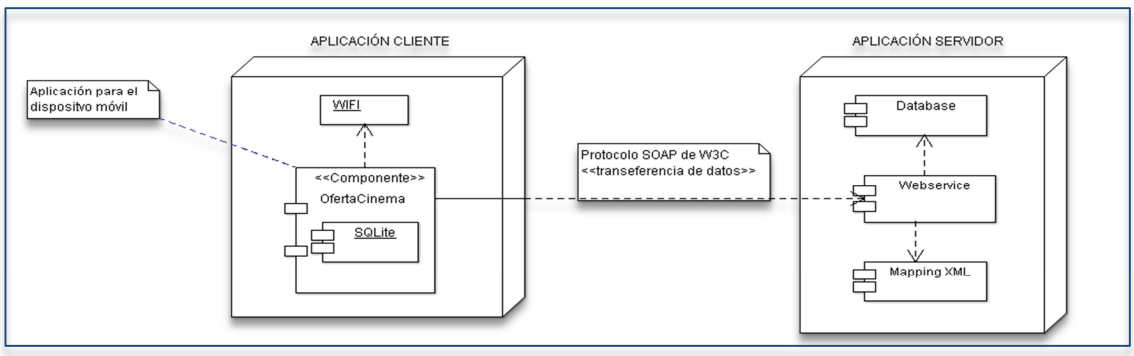
4.2.1. Diseño de la Arquitectura

En este apartado se analizará la arquitectura que debe tomar la aplicación así como un estudio de los componentes que la formarán. Esto permitirá trazar los requisitos descritos en la fase de análisis de tal manera que se pueda verificar que la implementación realizada cumple con todas las funcionalidades previamente descritas.

En la siguiente imagen se mostrara el aplicativo comunicándose con el webservice:

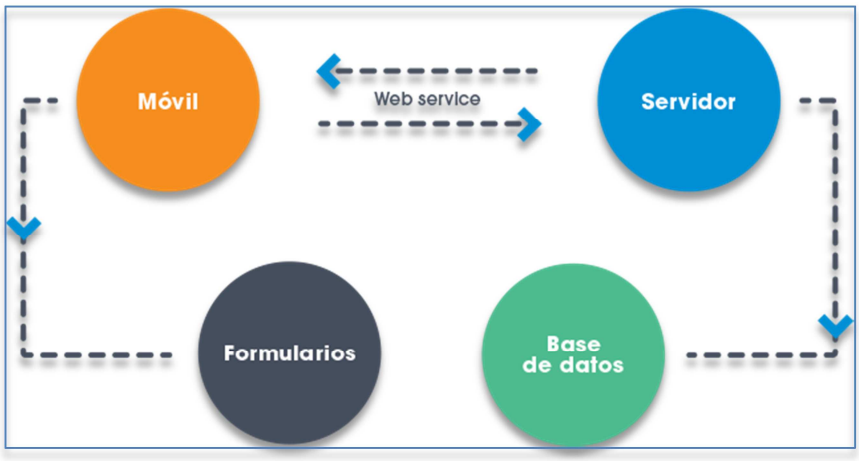


En el siguiente diagrama se mostrará una relación general de los componentes (cliente-servidor)

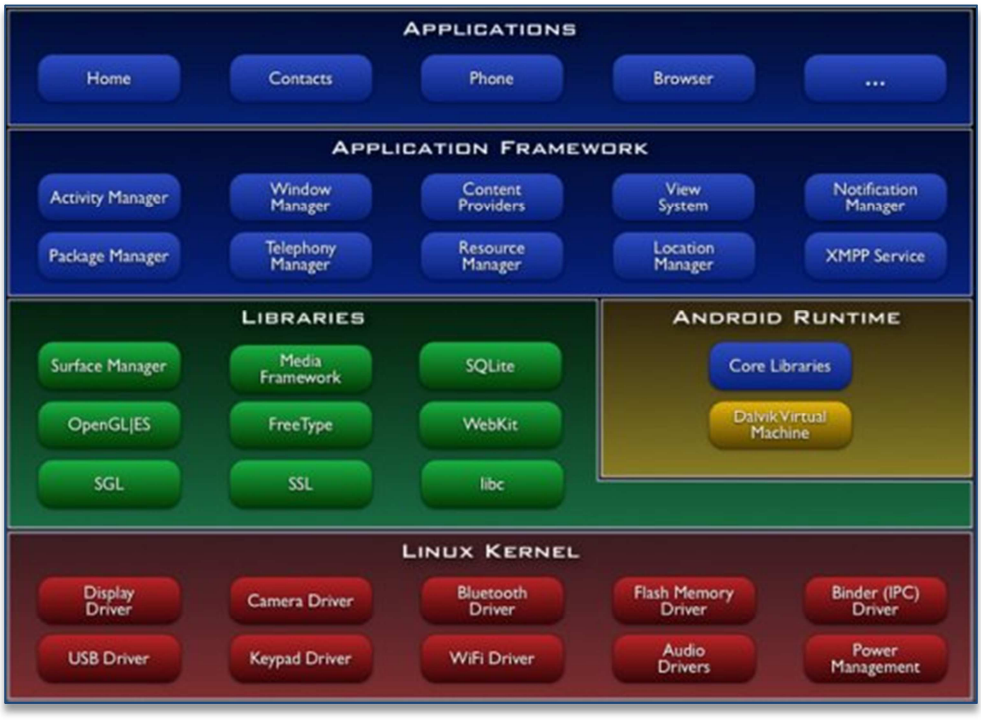


4.2.1.1. Arquitectura del cliente

En el siguiente gráfico se muestra como trabajará la aplicación del dispositivo móvil Android con el servidor.



En el siguiente gráfico muestra la arquitectura de Android. Como se puede ver está formada por cuatro capas. Una de las características más importantes es que todas las capas están basadas en software libre.



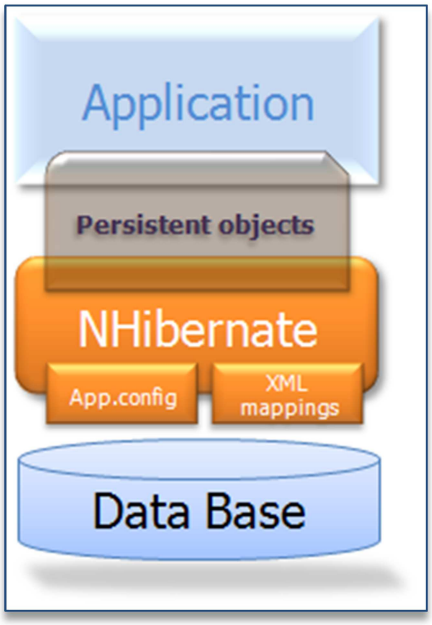
Para la aplicación Cliente, se utilizará el patrón MVC (Modelo –Vista -Controlador).

4.2.1.2. Arquitectura del servidor

Para la arquitectura del servidor, utilizaremos el Framework Hibernate que es una herramienta de Mapeo objeto-relacional (ORM) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

A continuación se detalla las capas que estará compuesta la aplicación web:

- **Vista**: funciones de webmethods que serán llamados por la aplicación Android.
- **Modelo**: ficheros *.hbm.xml, que se utilizan para mapear la relación de correspondencia entre la clase Java y la tabla de base de datos y declarado en el archivo de configuración de Hibernate "hibernate.cfg.xml".
- **Persistencia**: fichero de configuración para la conexión a la base de datos

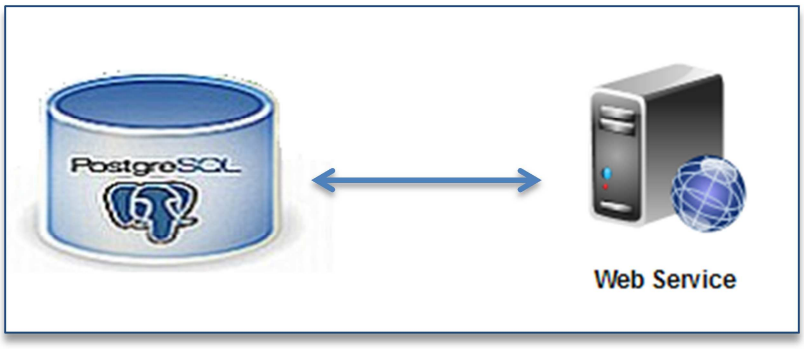


4.2.2. Diseño de la Base de Datos

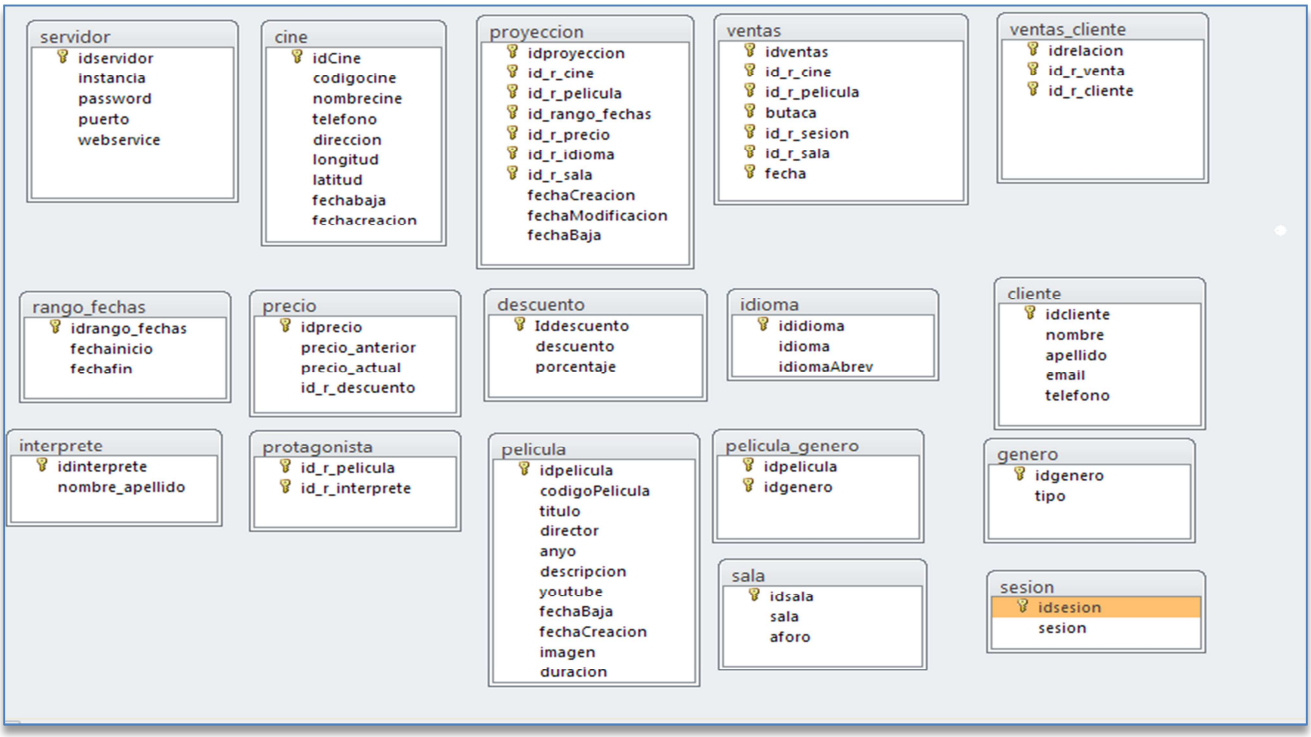
Debido a los requisitos obtenidos durante la fase de análisis, es necesario el uso de un sistema de almacenamiento para poder manejar toda la información con la que la aplicación va a trabajar. Se necesitara dos sistemas de almacenamiento; uno para el aplicativo Android y otro para el webservice.

- **Servidor – Webservice**

  Para el webservice, se utiliza PostgreSQL, que es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.



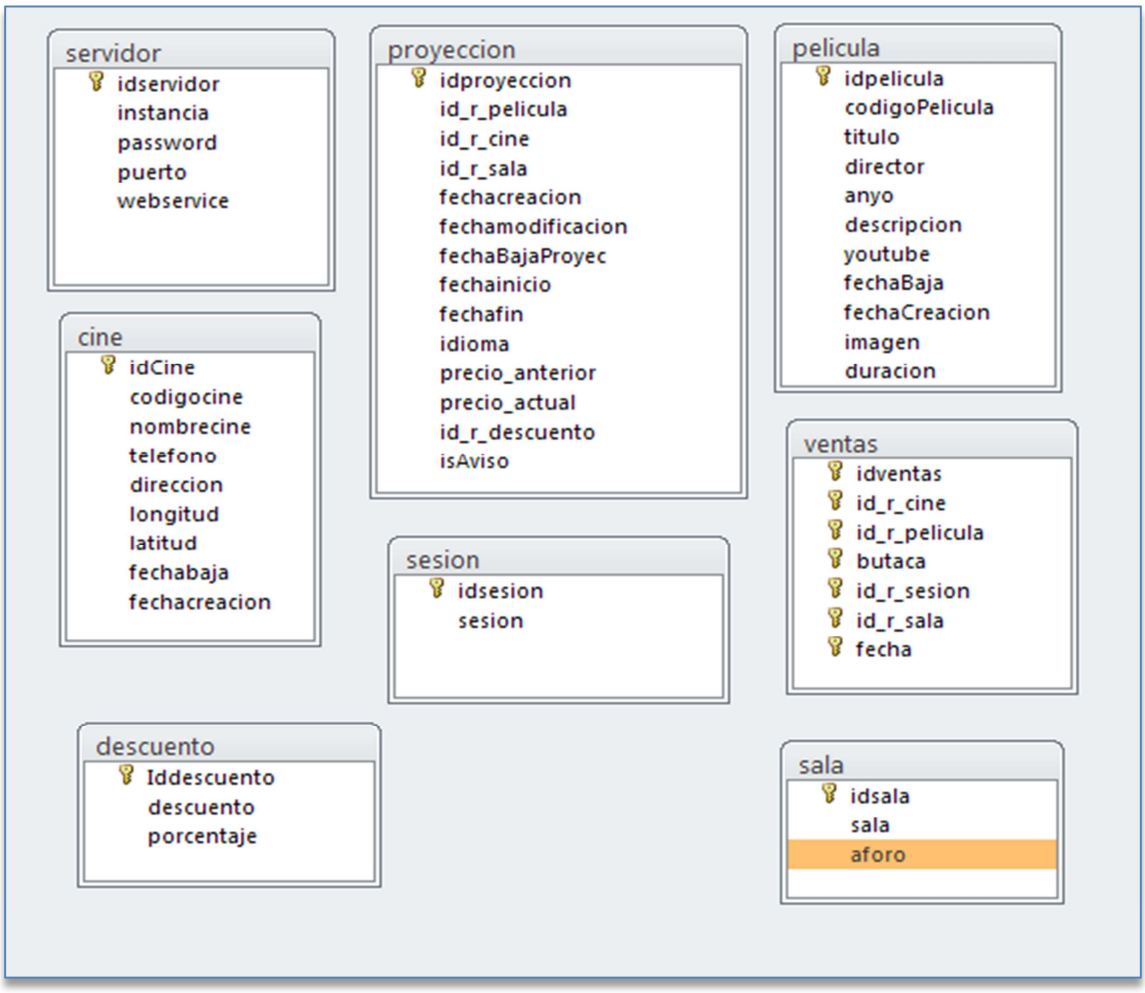
En el siguiente diagrama se muestra todas las tablas con sus correspondientes atributos que participan tanto en el proyecto webservice.



- **Cliente – Aplicativo en Android**

  Para el aplicativo en Android, se utiliza SQLite, que permite acceder a los datos mediante consultas de manera rápida y sencilla.

En el siguiente diagrama se muestra todas las tablas con sus correspondientes atributos que participan tanto en el aplicativo Android.



4.2.3. Diseño de la Interfaz

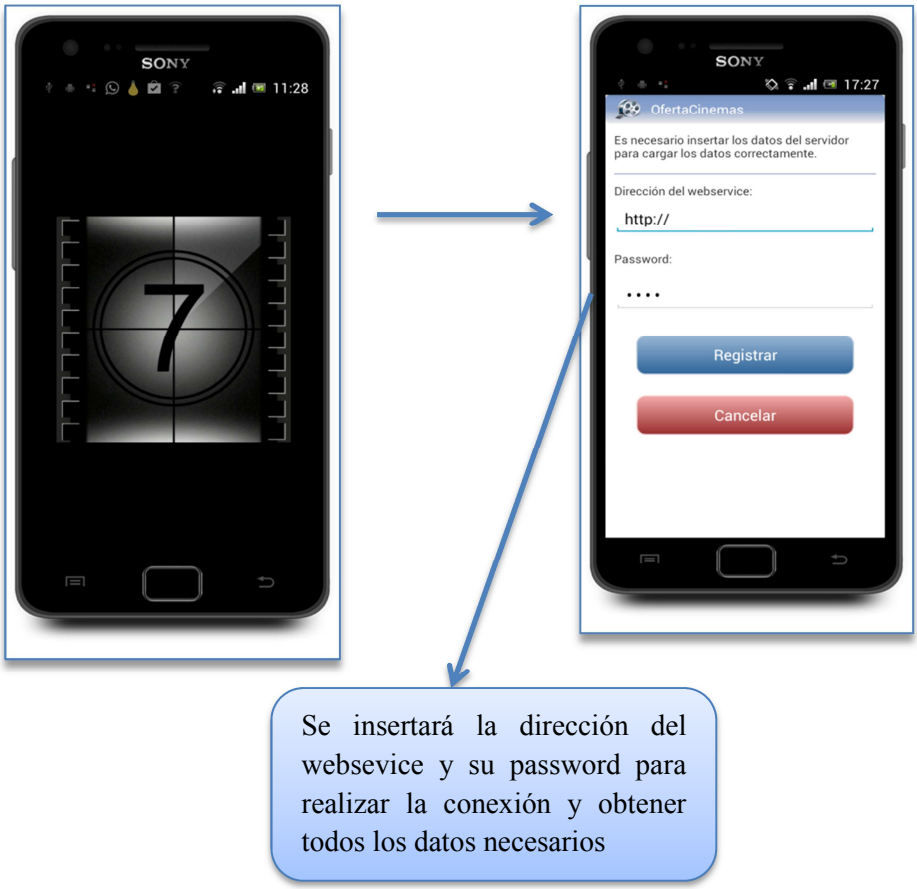
4.2.3.1. Icono de la Aplicación

El icono que mostrará en el dispositivo móvil, una vez realizado la instalación

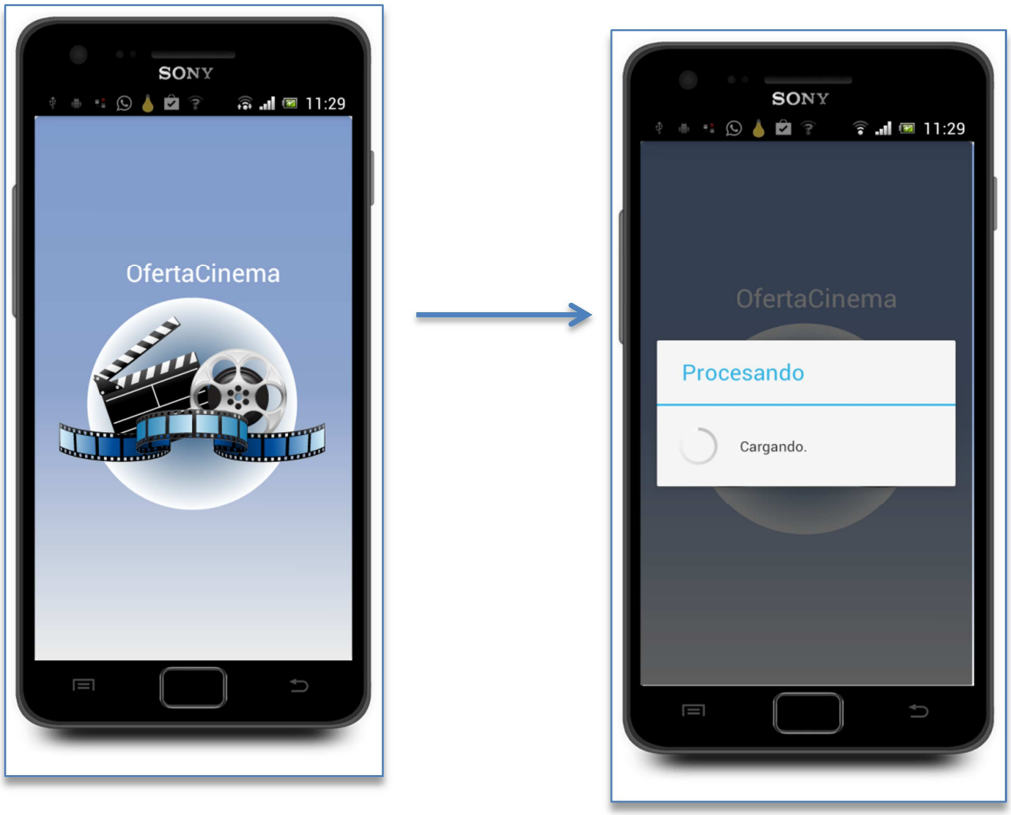
OfertaCinema

4.2.3.2. Pantallas

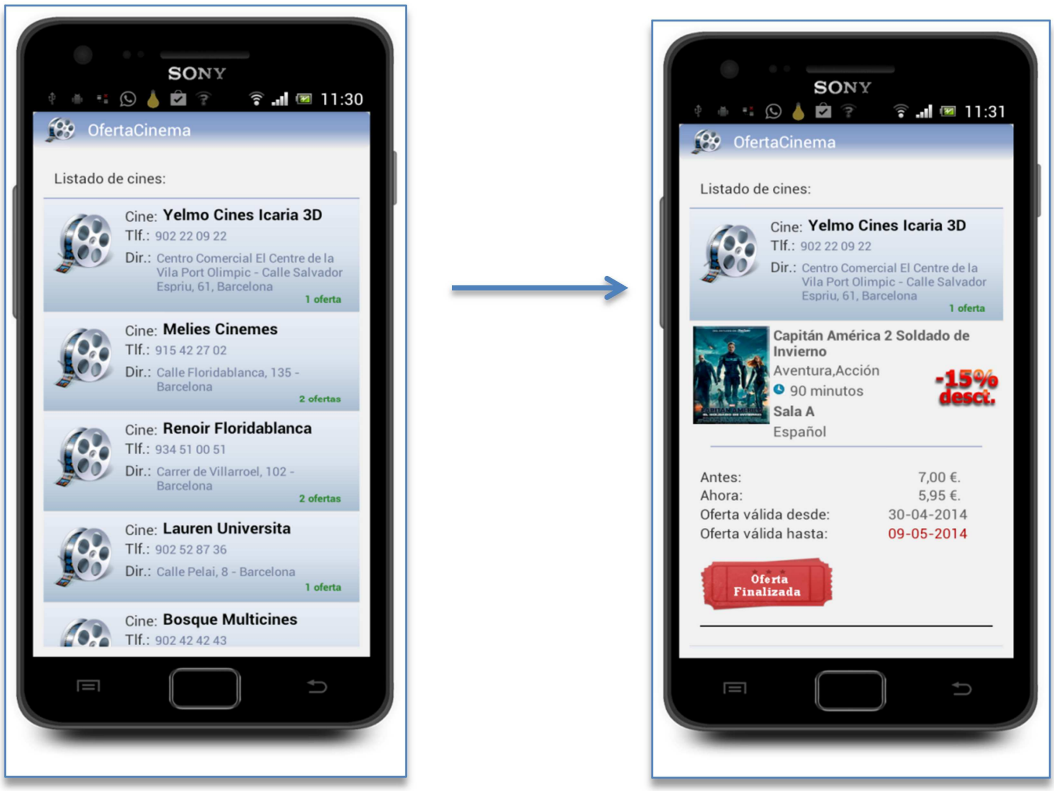
Pantallas de inicio de la aplicación



Pantallas de cargar de datos



Pantallas de listado de cines

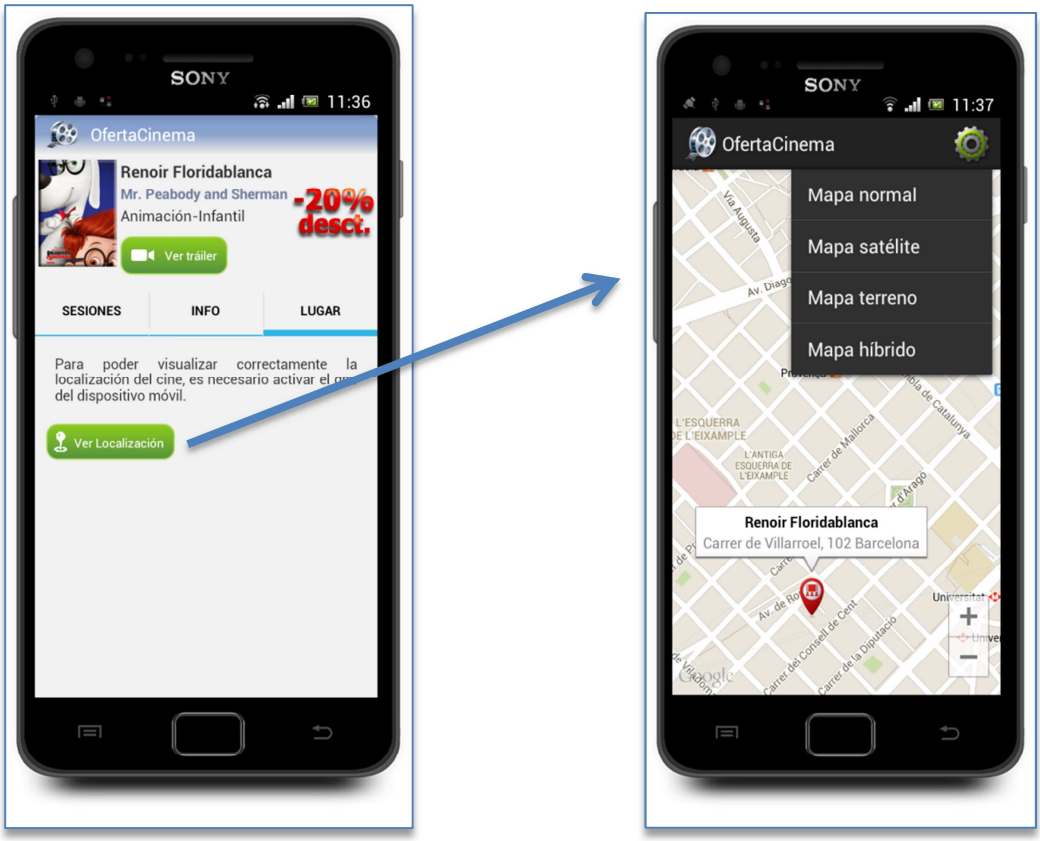


Mediante el aviso de una notificación se puede modificar los datos de una oferta, mostrando en el cine un icono "+".En este caso el tiempo estimado para obtener notificaciones es de 1 minuto (para realizar las pruebas)

Pantallas de una película seleccionada-1ºPestaña

Pantallas de una película seleccionada-2ºPestaña

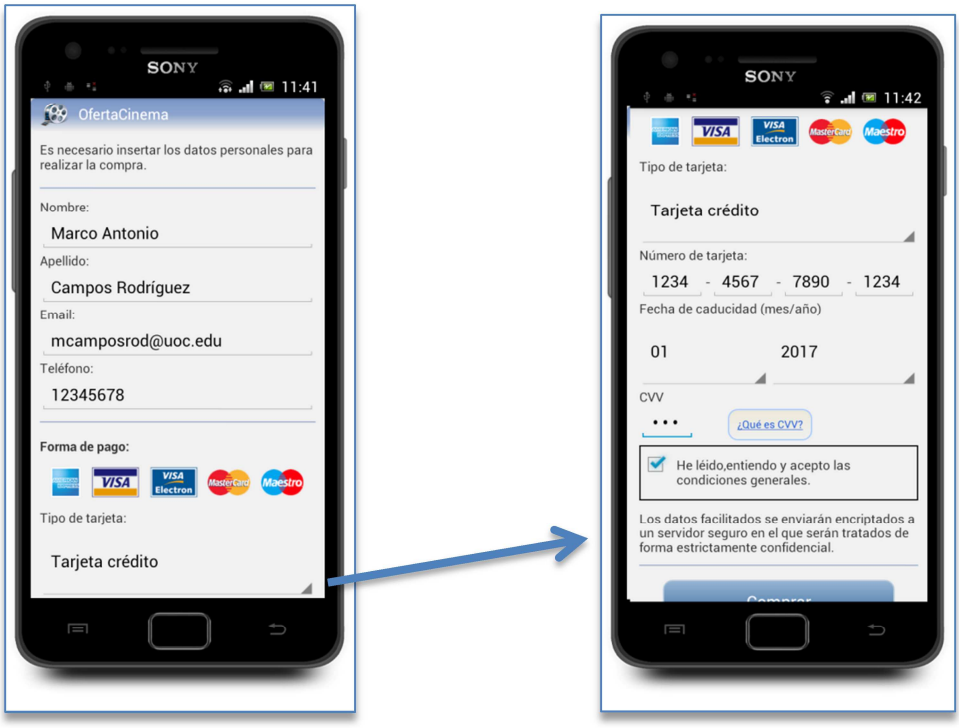
Pantallas de una película seleccionada-3ºPestaña

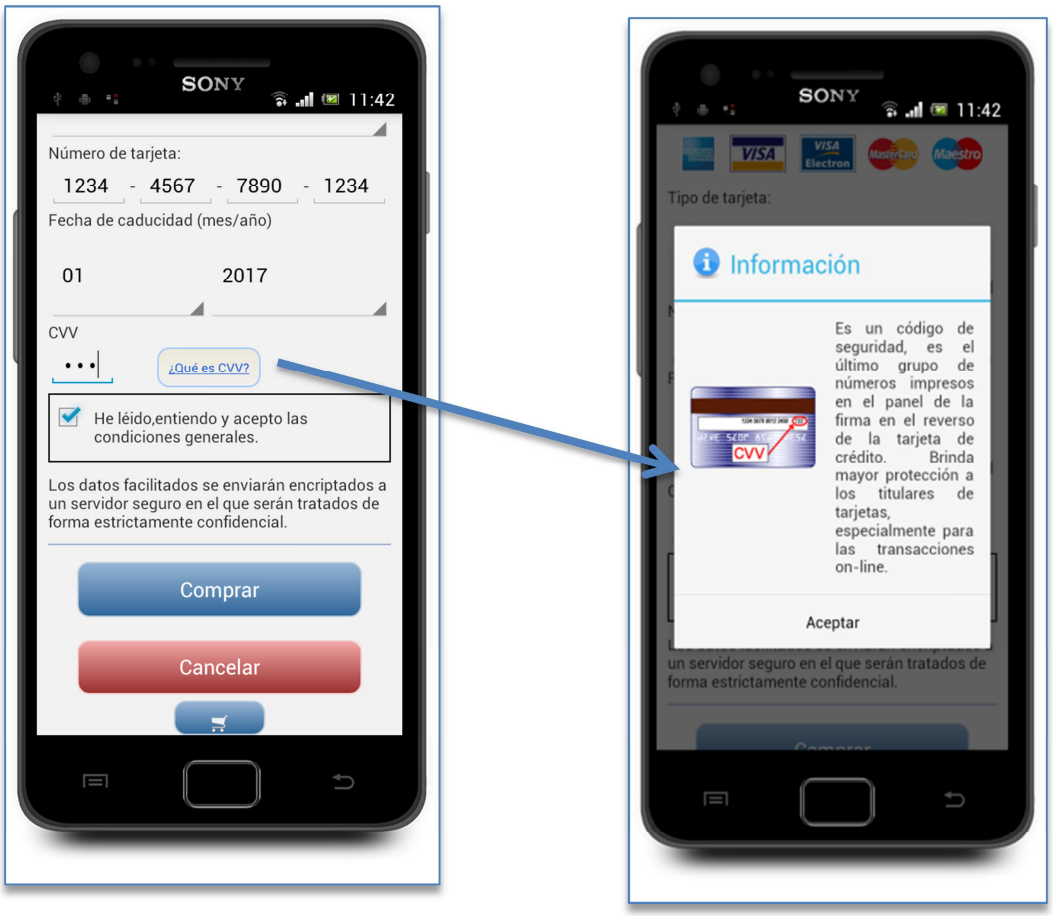


Pantalla de sala en donde se proyecta la película

Pantalla de información de lo seleccionado

Pantalla para realizar la compra

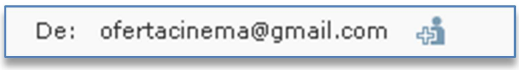





SONY
11:42
Número de tarjeta:
1234 - 4567 - 7890 - 1234
Fecha de caducidad (mes/año)
01 2017
CVV
¿Qué es CVV?
He léido,entiendo y acepto las condiciones generales.
Los datos facilitados se enviarán encriptados a un servidor seguro en el que serán tratados de forma estrictamente confidencial.
Comprar
Cancelar
OfertaCinema
Renoir Floridablanca
Carrer de Villarroel, 102
Barcelona - 934 51 00 51
Mr. Peabody and Sherman
Fecha: 03-06-2014 Sesión: 20:30h.
Asiento: 1A, 4A, 10A
PVP: 23,23€ (IVA incluído)
Sala B
Atención
¿Desea realizar la compra de las entradas seleccionadas?
No
Si
11:43
Información
La compra se ha realizado correctamente.En breve recibirá un email con los datos de la compra.
Aceptar

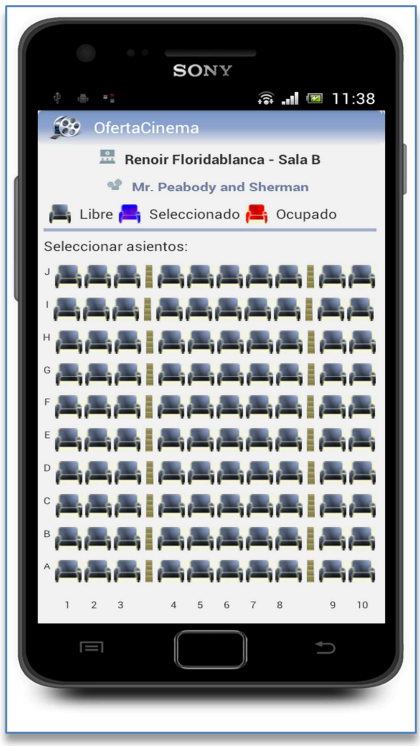
Una vez realizado la compra, en el correo que hemos insertado (en este caso utilizamos el dela UOC), aparecerá un email enviado por el aplicativo, adjuntando un fichero PDF con los datos de la compra. Como se muestra en este ejemplo:

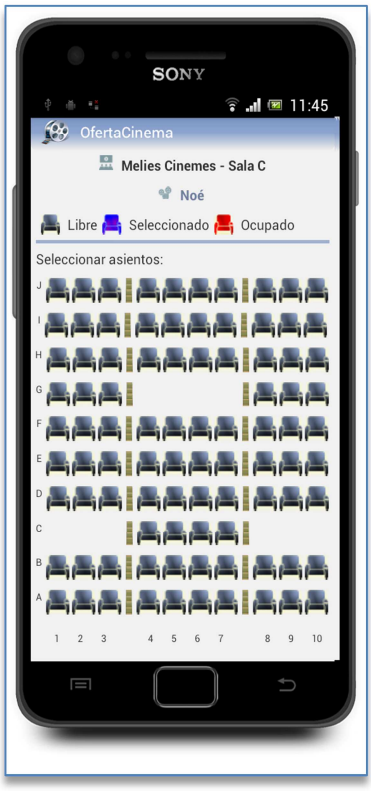
Enviado por:

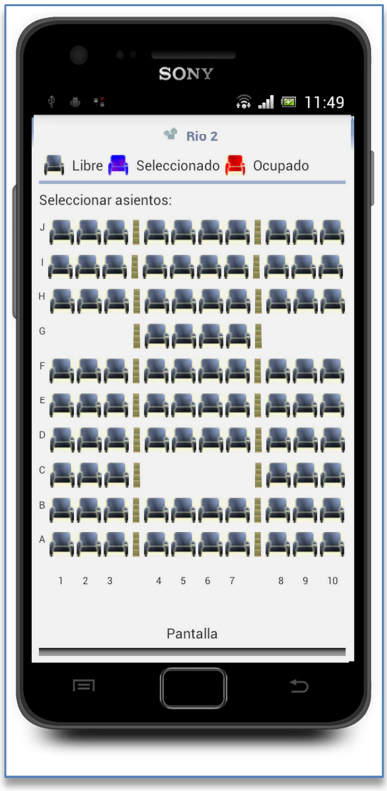


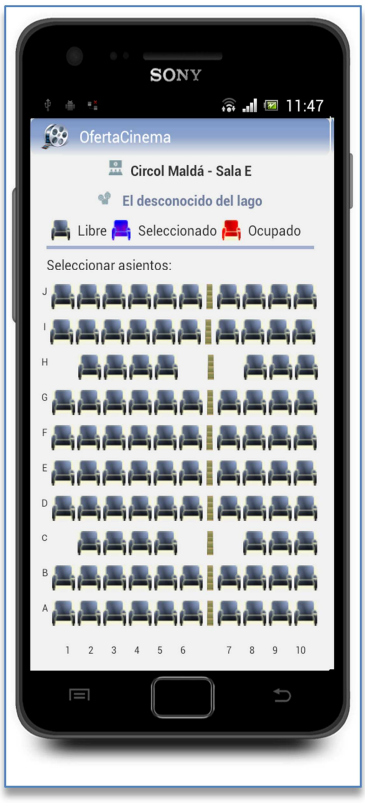
Archivo adjunto:

En este proyecto existen **5 variedades de salas**



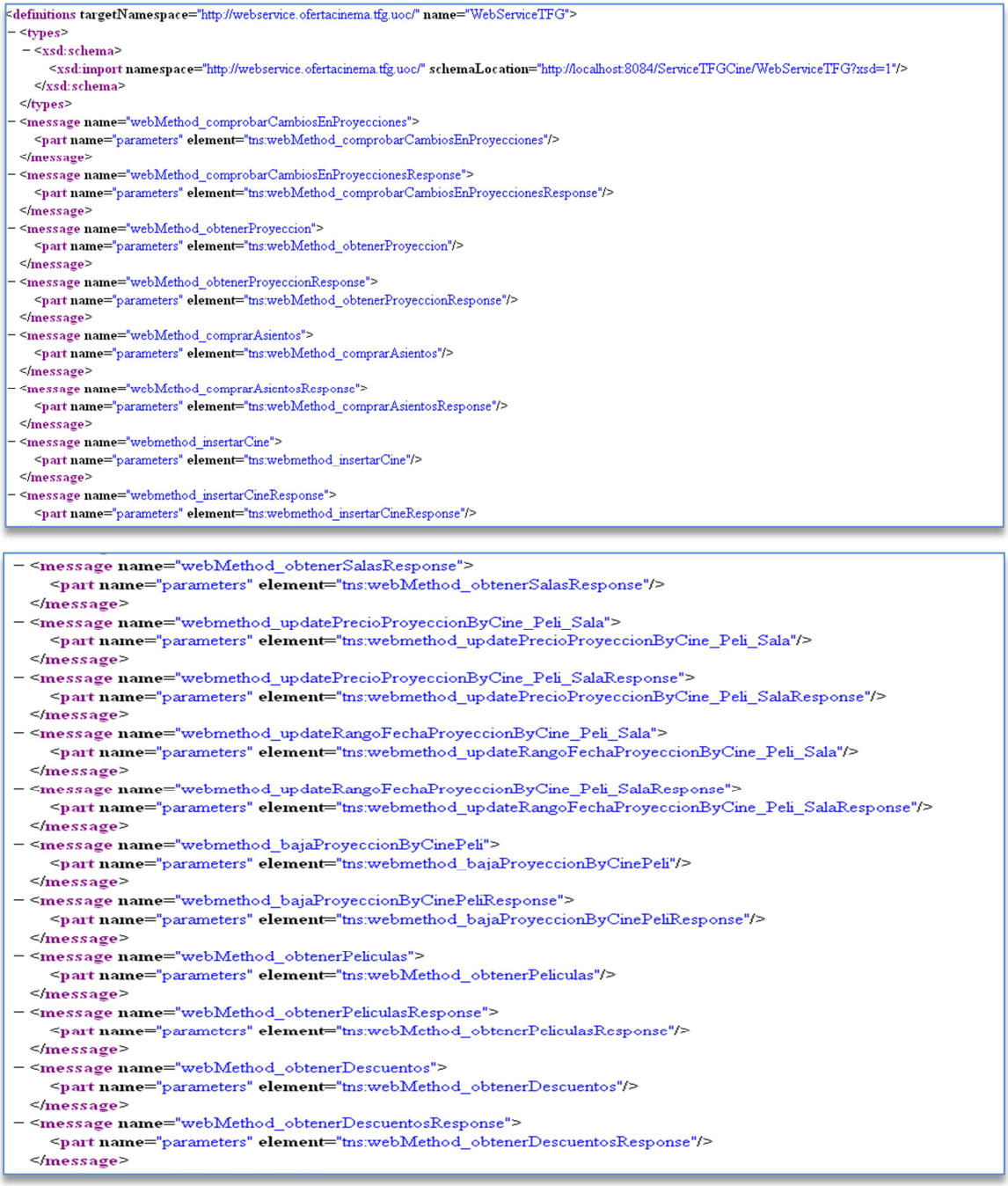






Para la parte del Servidor, se omitirá cualquier interfaz gráfico, ya que no provee de estos recursos para mostrar al usuario, además solo se trata de funcionalidades (webmethods) que es diseñado para ser invocado por la aplicación. Esta independencia se logra a través del formato de intercambio de los datos el cual es al final de cuentas XML, con ciertos campos que están definidos por el protocolo SOAP.

En la siguiente imagen, se muestra la información (en formato XML) después de realizar la ejecución del webservice.

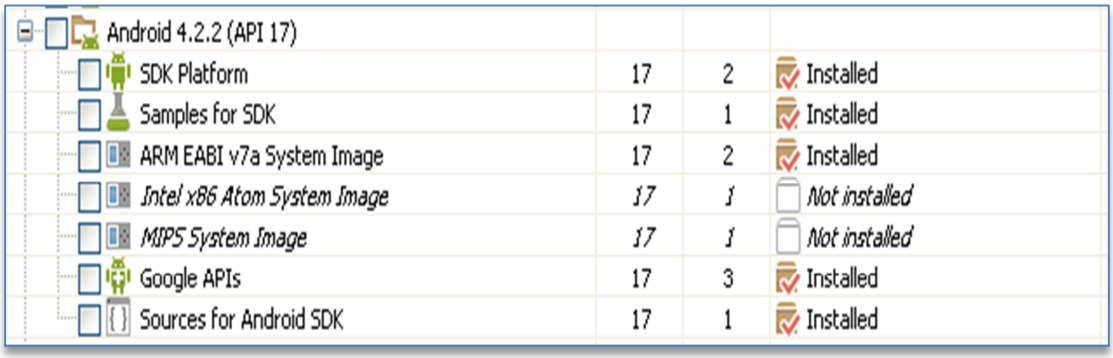


5.IMPLEMENTACIÓN

5.1. Cliente

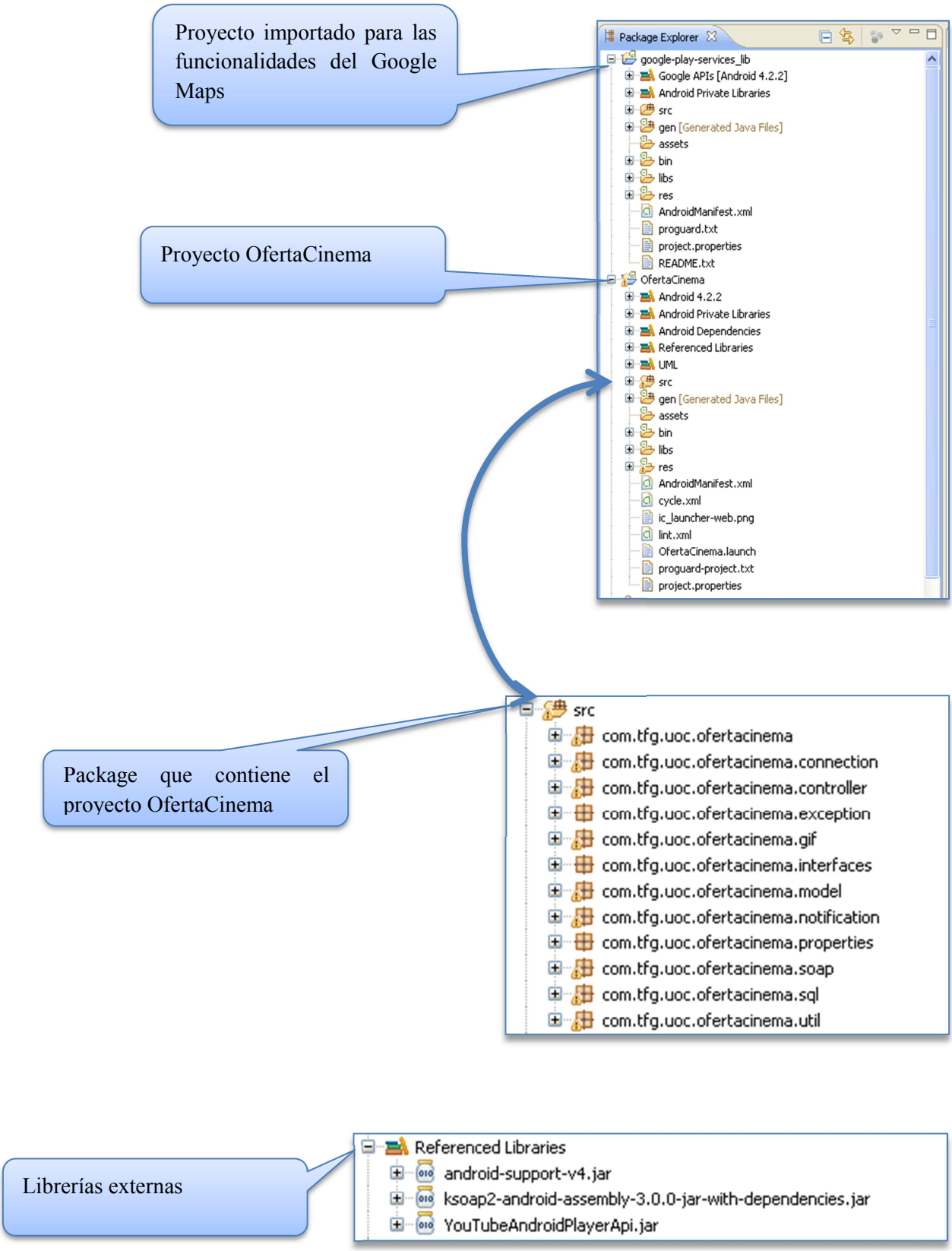
Como se ha comentado anteriormente, para el aplicativo cliente, se ha utilizado el IDE Eclipse.

El SDK Manager Android que se utiliza en este proyecto es Android 4.2.2.

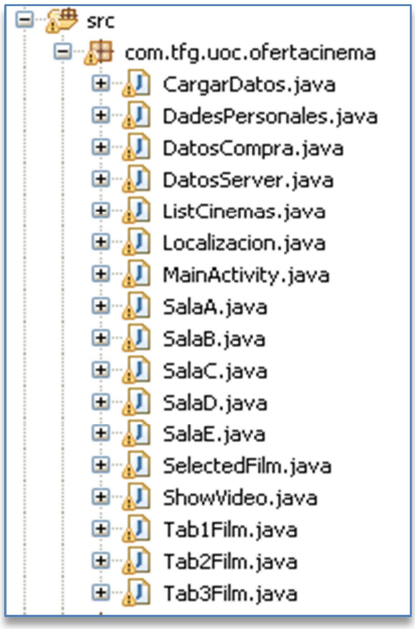


5.1.1. Descripción de Package

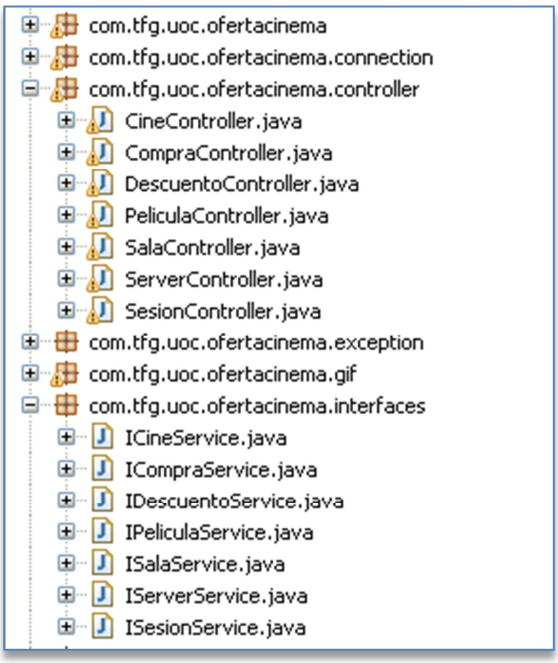
A continuación mostramos los componentes que contiene el proyecto



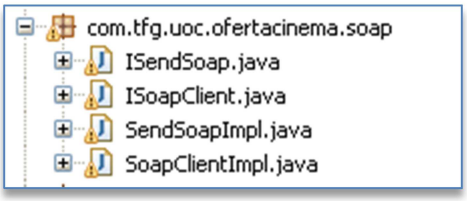
Package para las clases **MainActivity.**



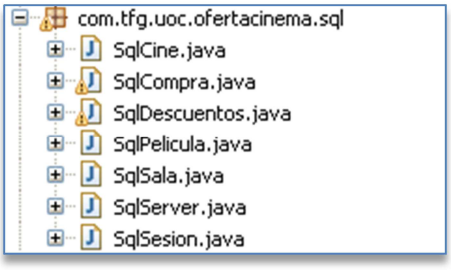
Package de la **lógica de negocio**.



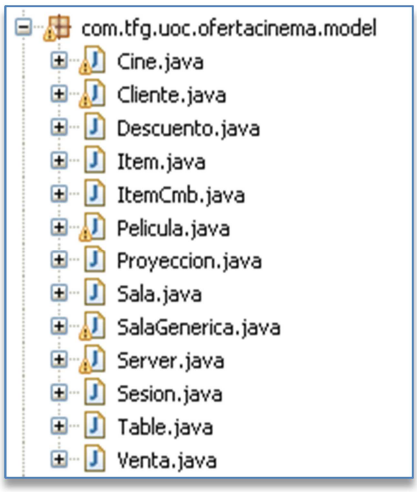
Package de que contiene las clases del servicio **SOAP**, para la comunicación con el webservice.



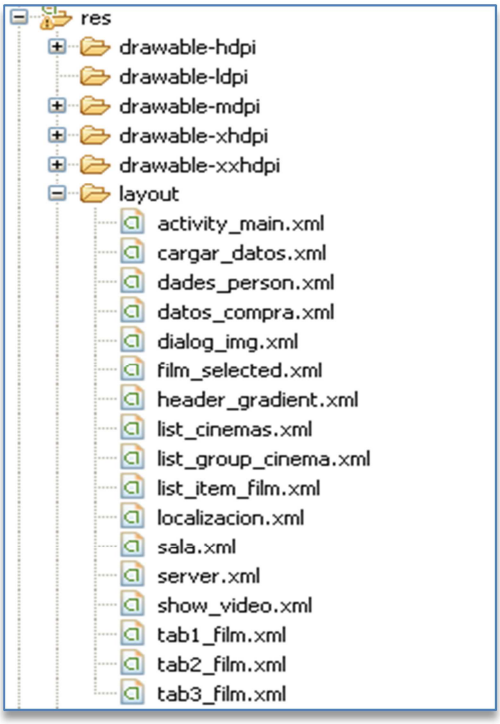
Package que contiene las clases para las consultas **SQL**.



Package que contiene las clases Entidades, que nos facilita enormemente el transporte de la información.



Directorio res/ contiene todos los grupos de recursos tales como drawable/, layout/, menu /, valúes/, XML/.

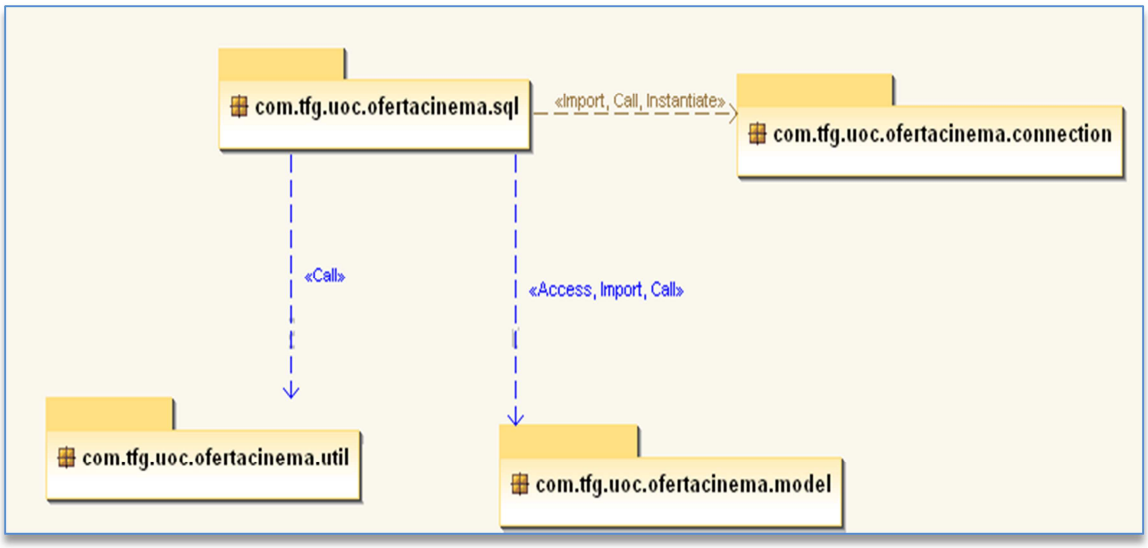


5.1.2. Relación de Package

A continuación mostramos los Package de mayor importancia y los que tienen mayor relación.

- **Conexión**

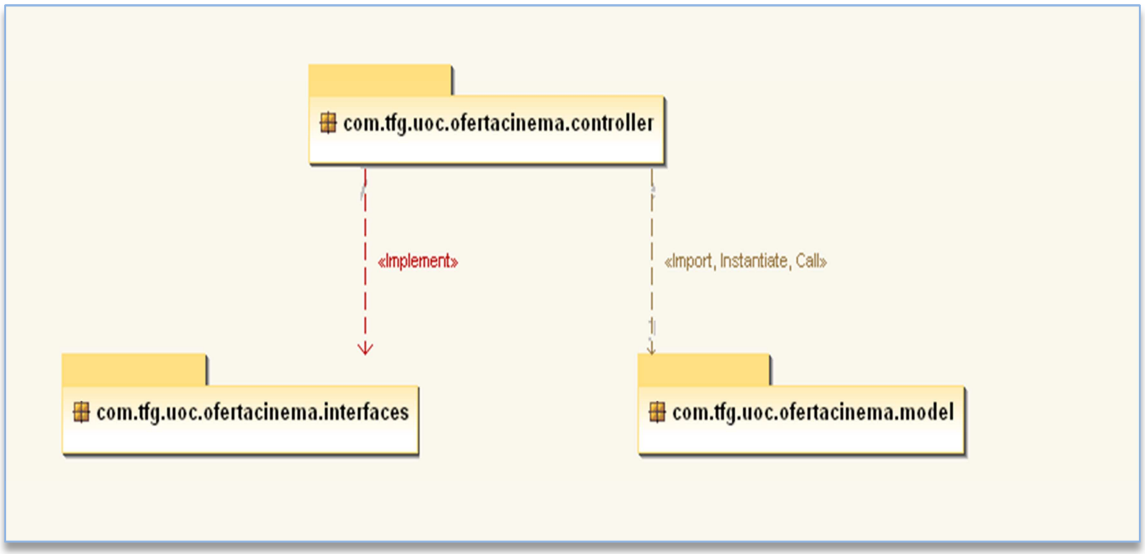
  Estos Package tienen relación para realizar la conexión con la base de datos (SQLite) y mostrarlo en a través de la aplicación en el dispositivo móvil.



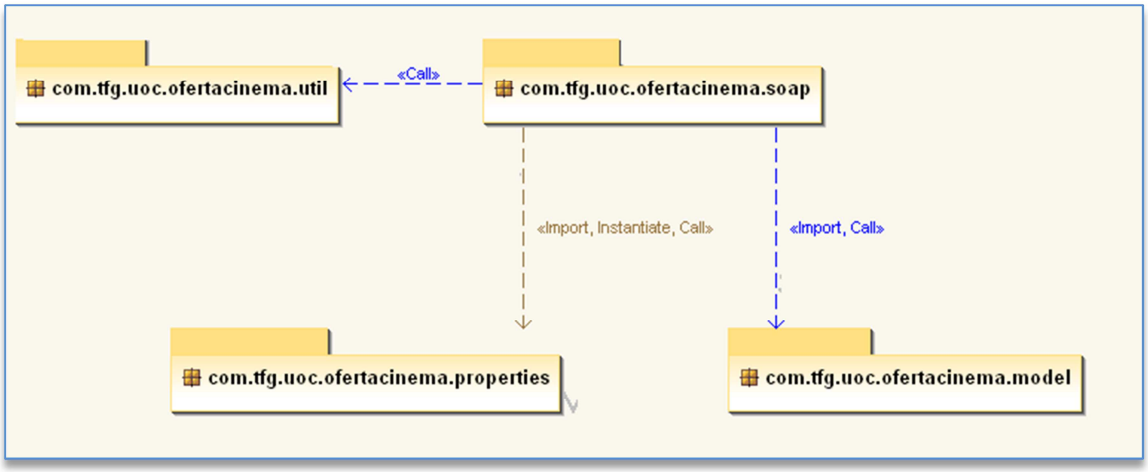
- **Interface**

  Estos Package tienen relación para realizar las funcionalidades de la capa de negocio (Controller), tanto como para obtener datos y como enviar los datos al webservice.

  Es el encargado de procesar los mandatos del usuario y programar eventos.

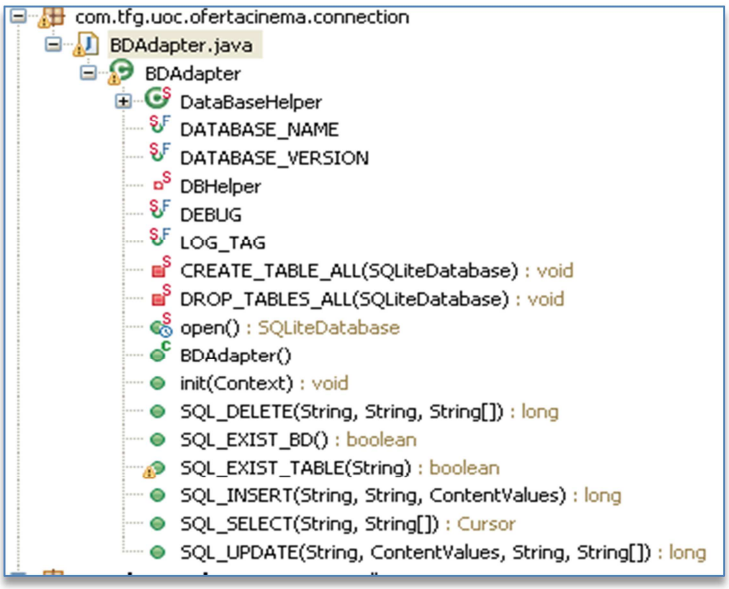


- **SOAP**
  Los siguientes Package participan en la comunicación con el protocolo SOAP, tanto para la carga de las clases Entidades, como tratar los datos en XML, que utiliza el protocolo SOAP para el intercambio de datos.



5.1.3. Descripción de Clases

Se crea una **clase BDAdapter**, para implementar la creación de las tablas que utilizará la base de datos SQLlite



Atributos que se utilizaran para la creación de la base de datos.

```
public static final String DATABASE_NAME = "TFG_CINE";
/******* if debug is set true then it will show all Logcat message ***/
public static final boolean DEBUG = true;
public static final String LOG_TAG = "TFG_CINE-DBAdapter";
/**** Database Version (Increase one if want to also upgrade your database) ****/
public static final int DATABASE_VERSION = 1;// started at 1
```

Se crea un subclase que heredará de **SQLiteOpenHelper**., que nos permite crear la base de datos y actualizar la estructura de tablas y datos iniciales.

```
/********** Main Database creation INNER class ********/
public static class DataBaseHelper extends SQLiteOpenHelper {

     public DataBaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

     @Override
    public void onCreate(SQLiteDatabase db) {
        if (DEBUG)
            Log.i(LOG_TAG, "new create");
        try {
            CREATE_TABLE_ALL(db);

        } catch (Exception exception) {
            if (DEBUG)
                Log.i(LOG_TAG, "Exception onCreate() exception");
        }
    }
     @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        if (DEBUG)
            Log.w(LOG_TAG, "Upgrading database from version" + oldVersion
                    + "to" + newVersion + "...");

        DROP_TABLES_ALL(db);
        onCreate(db);
    }
```

Función para crear las tablas

Clase para la creación de las tablas que contendrá la base de datos SQLite.

```
    public static String strTable = "";
    public static String T_cine = "cine";
    public static String T_pelicula = "pelicula";
    public static String T_proyeccion = "proyeccion";
    public static String T_descuento = "descuento";
    public static String T_sala = "sala";
    public static String T_ventas = "ventas";
    public static String T_sesion = "sesion";
    public static String T_cliente = "cliente";
    public static String T_server="server";

    /**
     * ********************************************
     * CREATE DE TABLAS
     * @return
     * ********************************************
     */

    public static String CREATE_T_cine() {
        try {
            strTable=" CREATE TABLE   IF NOT EXISTS  " + T_cine
            + " ( idcine INTEGER PRIMARY KEY AUTOINCREMENT, "
            + "  nombreCine TEXT, "
            + " telefono TEXT, "
            + " direccion TEXT, "
            + " longitud TEXT, "
            + " latitud TEXT, "
            + " fechaCreacion TEXT, "
            + " fechaBajaCine TEXT, "
            + " codigoCine TEXT ) " ;

            Log.v(" CREATE_T_cine = OK", strTable);
        } catch (Exception ex) {
            ex.toString();
            Log.v(" CREATE_T_cine = ERROR", ex.toString());
        }
```

Siguiendo la misma nomenclatura se crearán todas las tablas

En la siguiente clase se detalla el **MainActivity,** clase con la que se inicia la aplicación. MainActivity métodos que utiliza Android, en donde se inicia una actividad invocando los métodos específicos que corresponden a las etapas del ciclo de vida.

```
public class MainActivity extends Activity {

    private Handler handler = new Handler();
    private Runnable r = null;
    private GifMovieView gif1;
    private CineController cineController = new CineController(this);
    private ServerController serverController = new ServerController(this);


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getBaseContext().getResources().updateConfiguration(GBL.setLocale("language"),
                getBaseContext().getResources().getDisplayMetrics());

        setContentView(R.layout.activity_main);

        r = null;
        handler = null;
        handler = new Handler();
        gif1 = (GifMovieView) findViewById(R.id.gif1);
        gif1.setMovieResource(R.drawable.gifcinema);

        ejecutarHandler();

    }
```

Fichero Properties.

*config.properties

```
5
6 NAMESPACE=http://webservice.ofertacinema.tfg.uoc/
7 #URL=http://192.168.1.33:8084/ServiceTFGCine/WebService
8 #String s = "http://192.168.1.34:8084/ServiceTFGCine/
9 ACTION =http://webservice.ofertacinema.tfg.uoc/
10
11
12
13 # ***********************************************
14 # ***********************************************
15 # WEBMETHODS DEL WEBSERVICE
16 # ***********************************************
17 comprobarServer =webMethod_ComprobarServer
18 cargarCines =webMethod_obtenerCines
19 cargarCine =webMethod_obtenerCine
20 obtenerPelis =webMethod_obtenerPeliculas
21 obtenerDescuentos =webMethod_obtenerDescuentos
22 obtenerSalas =webMethod_obtenerSalas
23 obtenerSesiones =webMethod_obtenerSesiones
24 obtenerIdiomas =webMethod_obtenerIdiomas
25 obtenerProyecciones =webMethod_obtenerProyecciones
26 obtenerProyeccion =webMethod_obtenerProyeccion
27 comprarAsientos=webMethod_comprarAsientos
28 obtenerVentas=webMethod_obtenerVentas
29 comprobarCambiosEnCines =webMethod_comprobarCambiosEnCines
30 comprobarCambiosEnProyecciones =webMethod_comprobarCambiosEnProyecciones
31
32 # ***********************************************
33 # ***********************************************
34 ##NOTIFICACION
35 ##tiempo en minutos 1=60segundos
36 # ***********************************************
37 #intervaloNotificacion=1
38 intervaloTimer=1
39
```

Parámetros para la comunicacionSOAP

Webmethods del webservice

En la siguiente clase se detalla una de las clases que contiene funcionalidades para la comunicación SOAP.

```
public class SoapClientImpl implements ISoapClient {
    private     java.util.Vector<SoapObject> resultSoap;
    private SoapSerializationEnvelope soapEnvelope = null;
    private String URL="";
private Context context;

    @SuppressLint("NewApi")
    public SoapClientImpl(Context ctx) {
        context = ctx;
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder()
                .permitAll().build();

        StrictMode.setThreadPolicy(policy);

        soapEnvelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);

    }
```

```
// ----------SERVER WebService_ComprobarServer------
@Override
public String comprobarServer(String strWebservice, String strPswd)
        throws IOException, XmlPullParserException {
    String stResult = "";

    SoapObject req = new SoapObject(GBL.read("NAMESPACE"), GBL.read("comprobarServer"));
    SoapObject resul = null;
    soapEnvelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);

    req.addProperty(GBL.Prop("strWebservice", strWebservice,
            String.class));
    req.addProperty(GBL.Prop("strPswd", strPswd, String.class));

    soapEnvelope.setOutputSoapObject(req);

    try {
        URL=strWebservice;
        HttpTransportSE transporte = new HttpTransportSE(URL);
        String SOAP_ACTION = GBL.read("ACTION") + GBL.read("comprobarServer");
        transporte.call(SOAP_ACTION, soapEnvelope);
        try {
            resul = (SoapObject) soapEnvelope.bodyIn;
            String obj2 = String.valueOf(resul.getProperty(0).toString());
            stResult = obj2;// .getProperty(0).toString();
        } catch (Exception ex) {
            ex.toString();Log.v("comprobarServer", String.valueOf(ex.toString()));
            error("ERR_SERVER");
        }

    } catch (SoapFault ex) {
        ex.toString();
        error("ERR_SOAP");Log.v("comprobarServer", String.valueOf(ex.toString()));
    }
    return stResult;
```

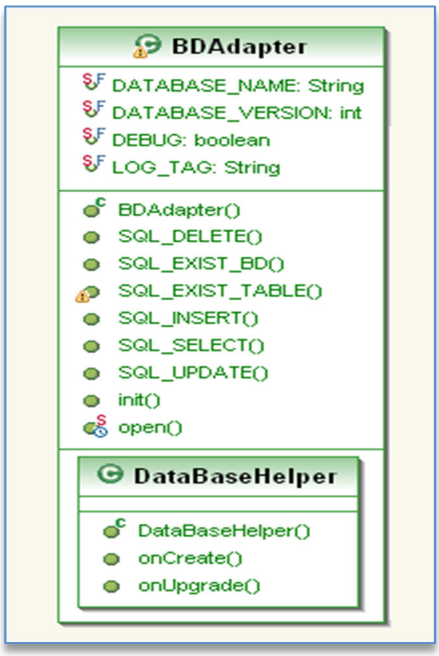
Fichero Manifest.xml, que contiene información esencial acerca de la aplicación para el sistema Android. Es la información que el sistema debe tener antes de que pueda ejecutar cualquiera de código de la aplicación

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tfg.uoc.ofertacinema"
    android:installLocation="preferExternal"
    android:label="@string/app_name"
    android:versionCode="1"
    android:versionName="1.0"
    >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17"/>

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"
         />

    <permission
        android:name="com.tfg.uoc.ofertacinema.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="com.tfg.uoc.ofertacinema.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

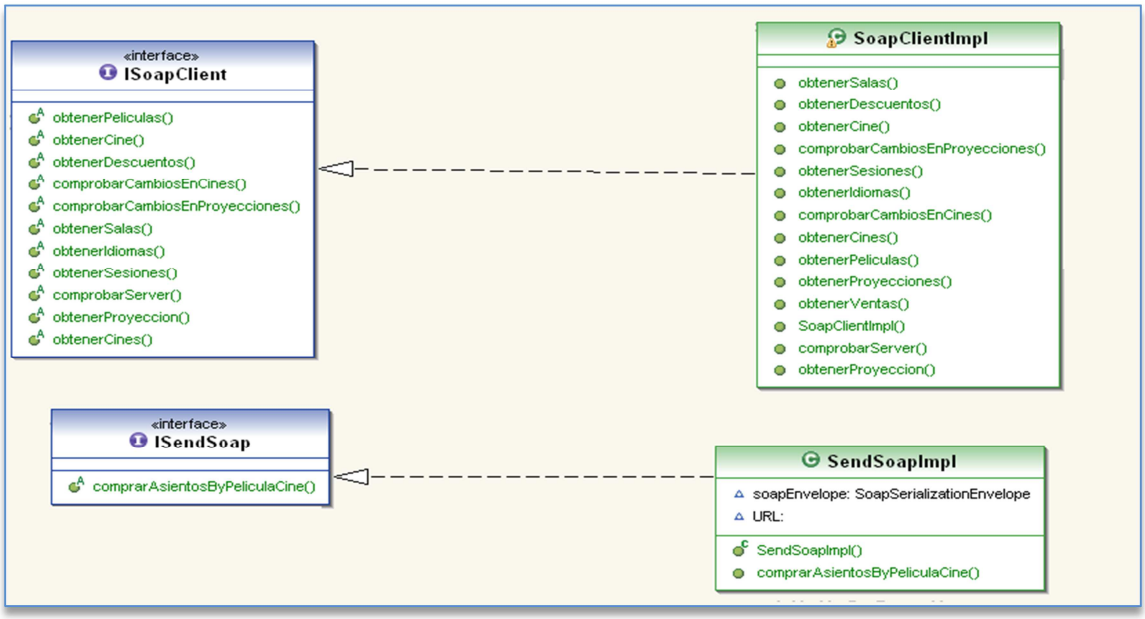
```
<application
    android:allowBackup="true"
    android:hardwareAccelerated="false"
    android:icon="@drawable/ic_ofertacinema"
    android:label="@string/app_name2"
    android:theme="@style/AppTheme" >
    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyBMwyELK8hvQlGq3AJSv5hyIm28zdGt7Ic" />
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />

    <uses-library android:name="com.google.android.maps" />

    <activity
        android:name="com.tfg.uoc.ofertacinema.MainActivity"
        android:hardwareAccelerated="false"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name="com.tfg.uoc.ofertacinema.DatosServer" >
    </activity>
```

5.1.4. Relación de Clases

- **BDAdapter**
  En esta clase tenemos la cadena de conexión con la que trabajará nuestra aplicación.
  En ella se define la base de datos y driver de conexión, si por ejemplo en un futuro se nos pide conectarnos a una base de datos diferente o establecer un sistema gestor distinto (pero con la misma estructura de tablas y campos), tan solo modificaremos esta clase y dicho cambio será transparente para el resto del sistema.



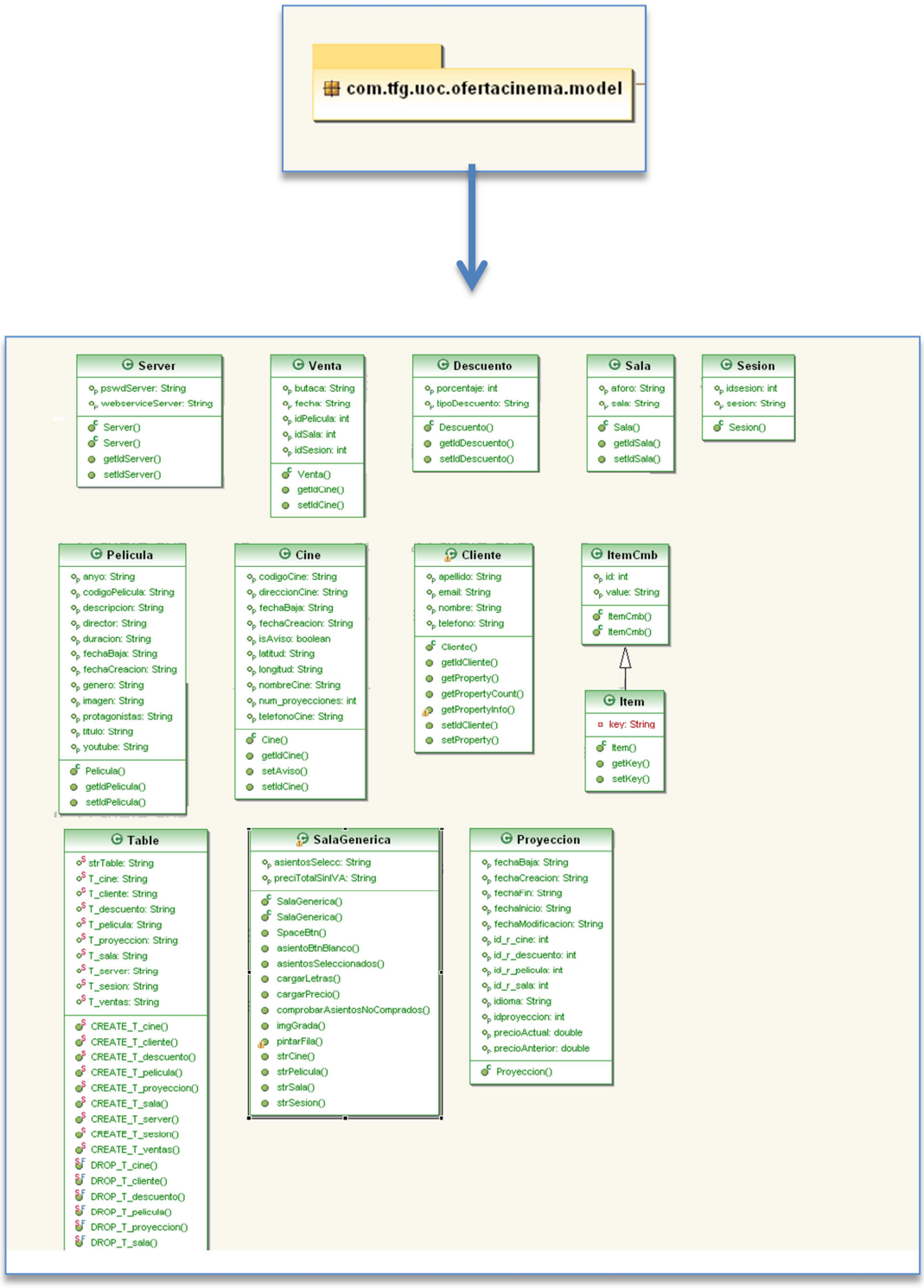
- **SOAP**
  A continuación mostramos las clases que permiten realizar las operaciones del protocolo de comunicación SOAP.Este protocolo nos ayudará a consumir las funcionalidades de un webservice.



- **Entidades**

  A continuación mostramos las clases que representamos las entidades (Tablas) de la base de datos.

  Nos facilita enormemente el transporte de la información, evitando que se envíen gran cantidad de parámetros a un método cuando queremos hacer un registro o actualización



- **Controller**

  A continuación mostramos las clases que permiten realizar las operaciones asociadas a la lógica de negocio como tal, desde ella realizamos las validaciones y llamados a las operaciones del sistema. Cada clase tendrá su propia clase interface para proporcionar un mecanismo de encapsulación de los protocolos de los métodos sin forzar al usuario a utilizar la herencia.

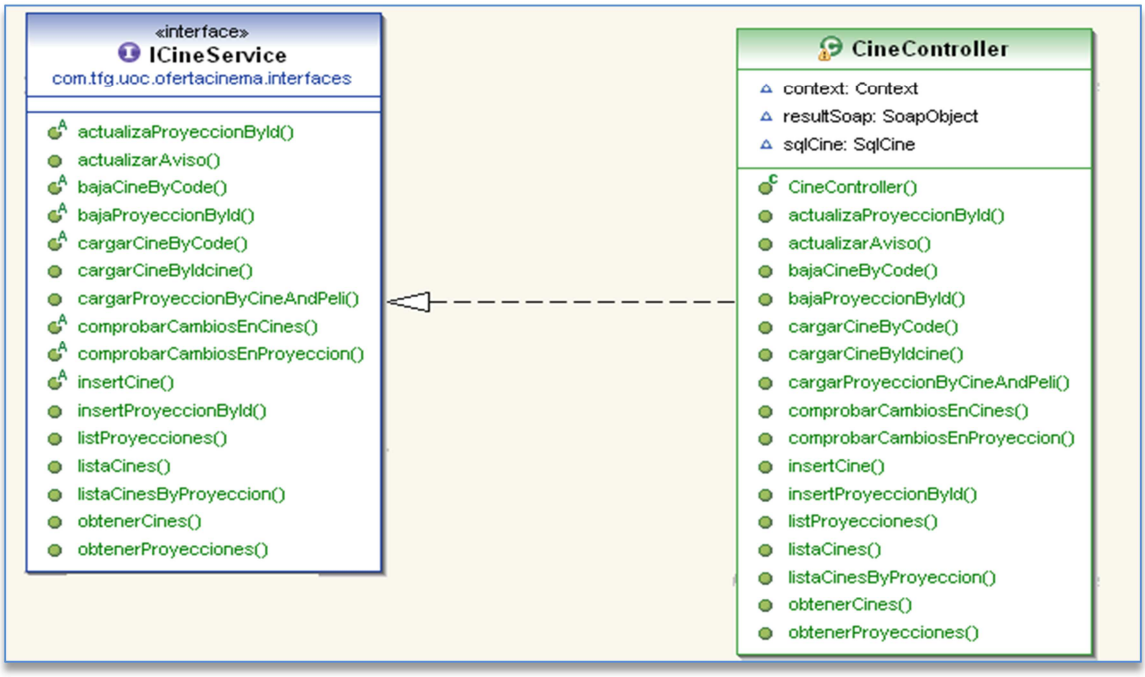


Diagrama de la clase Cine

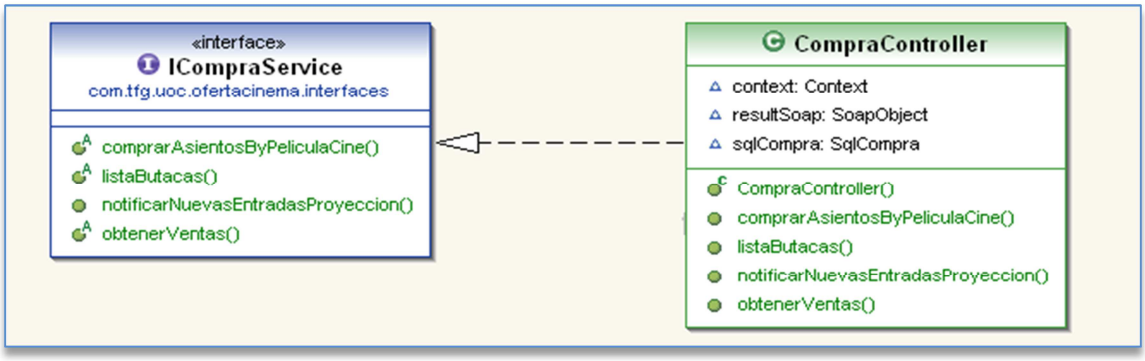


Diagrama de la clase Compra

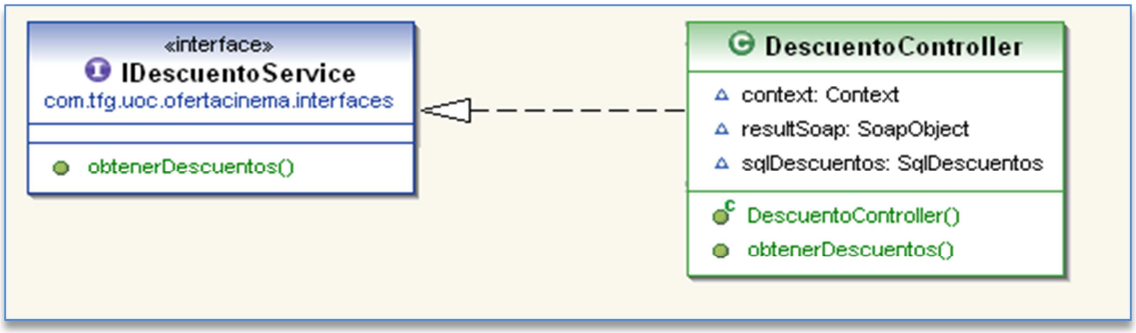


Diagrama de la clase Descuento

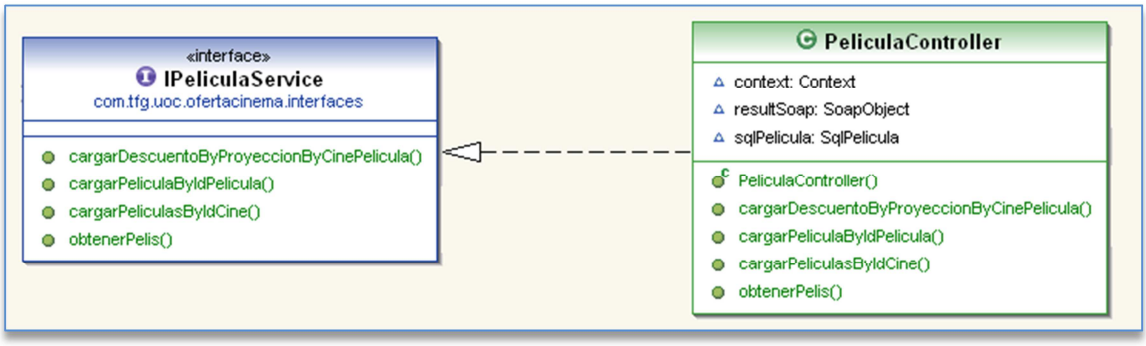


Diagrama de la clase Película

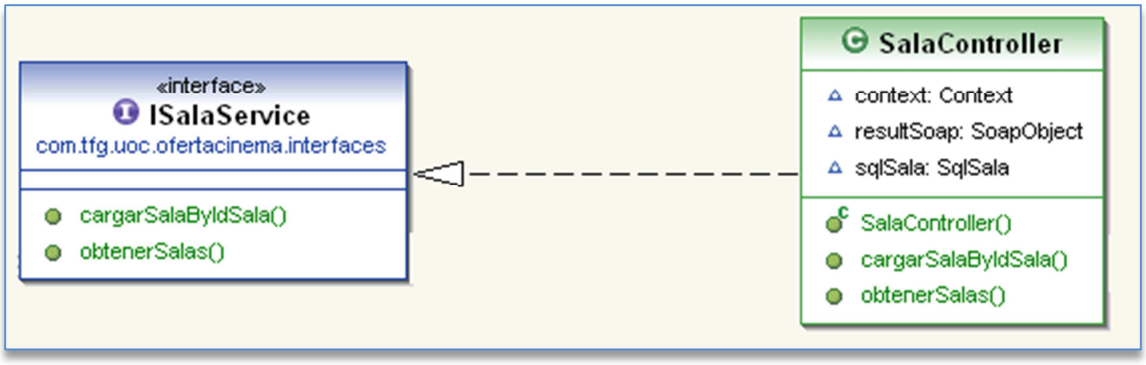


Diagrama de la clase Sala

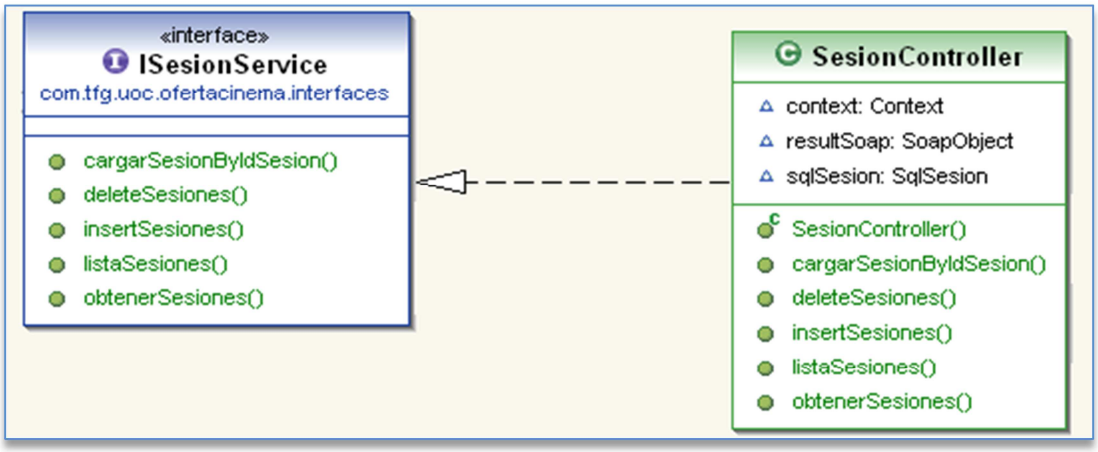


Diagrama de la clase Sesión

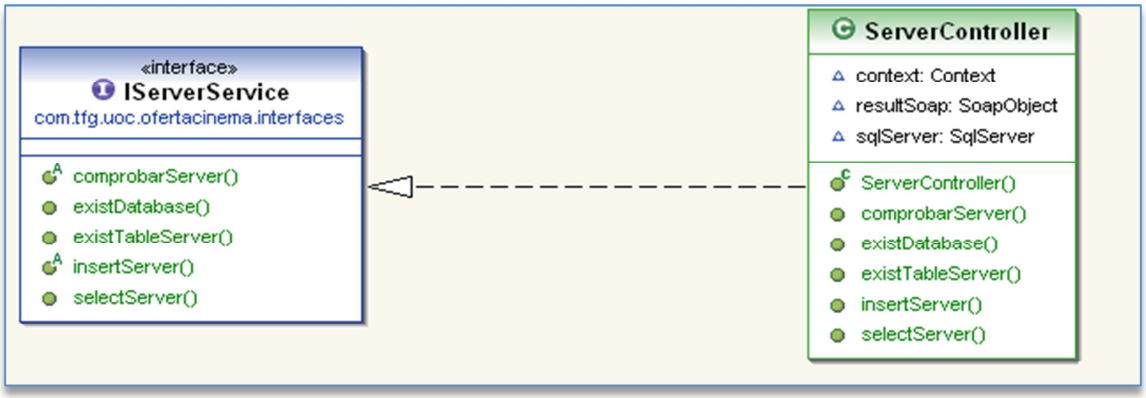
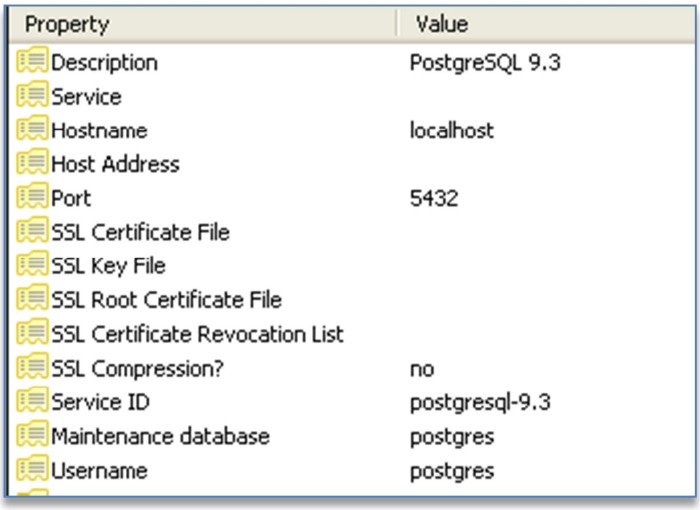


Diagrama de la clase Server

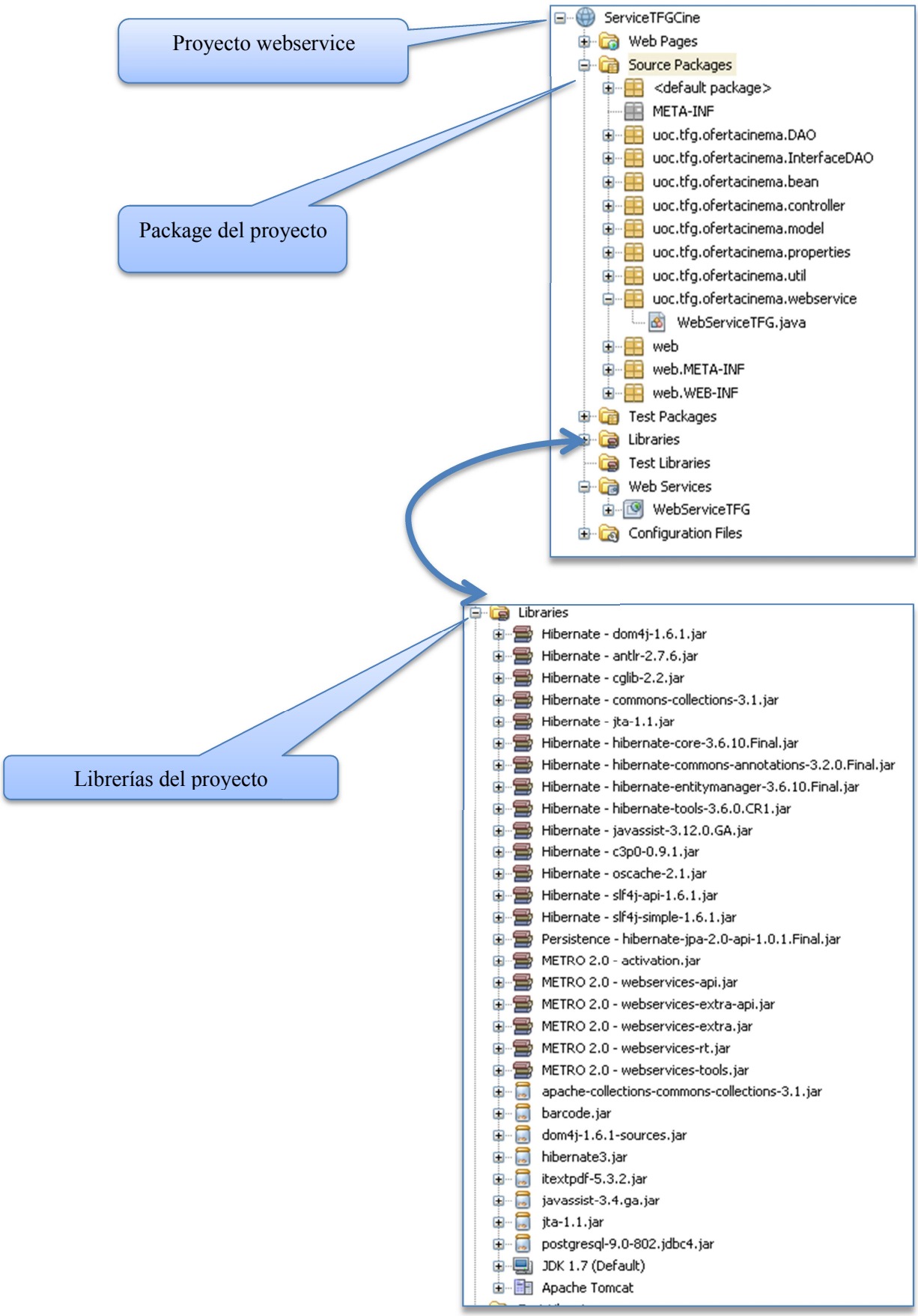
5.2. Servidor

Como se ha comentado anteriormente, para el aplicativo cliente, se ha utilizado el IDE NetBeans.

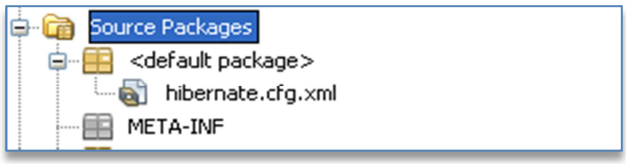
Como gestor de base de datos usamos PostgreSQL 9.3.



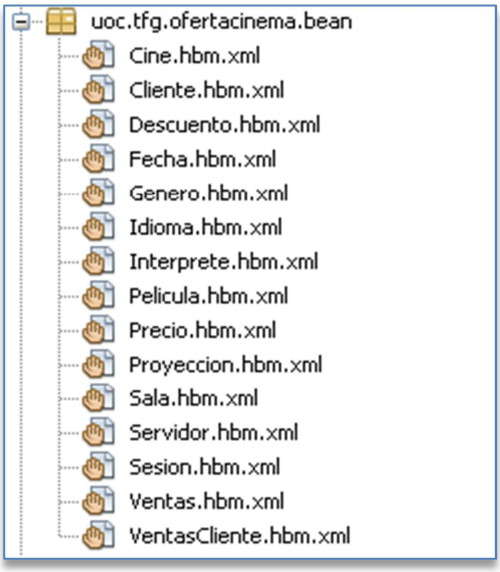
5.2.1. Descripción de Package



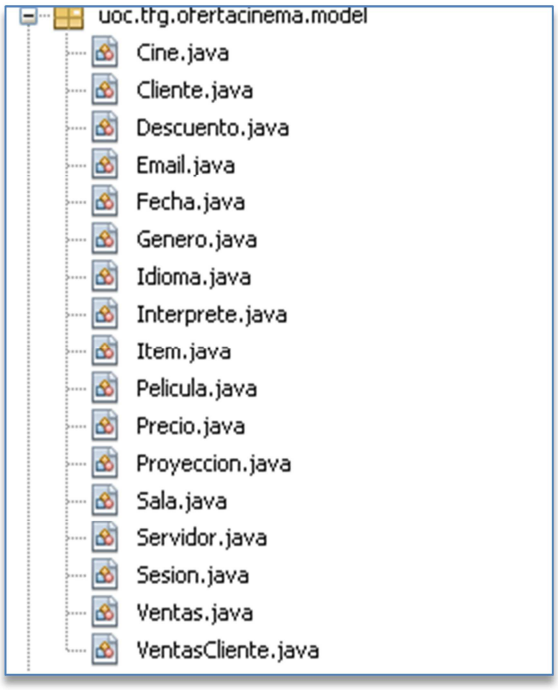
Package que contiene el fichero de conexión con la base de datos.



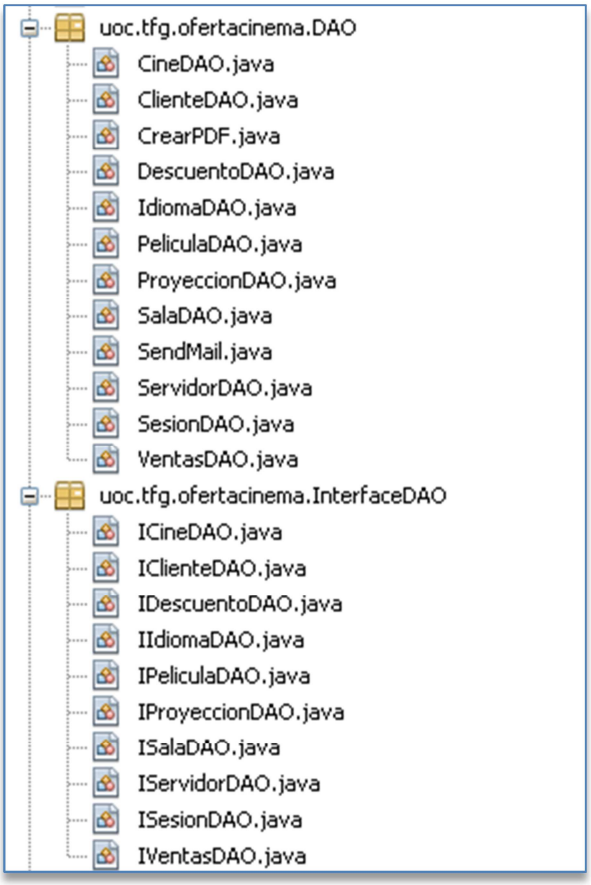
Package que contiene el fichero vean.xml que define cada tabla de la base de datos (mapeo de cada campo)



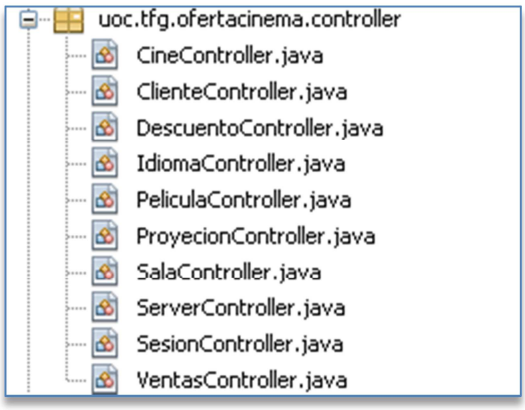
Package que contiene las clases Entidades, que nos facilita enormemente el transporte de la información



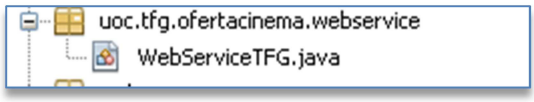
Package que contiene las clases DAO



Package que contiene la lógica de negocio

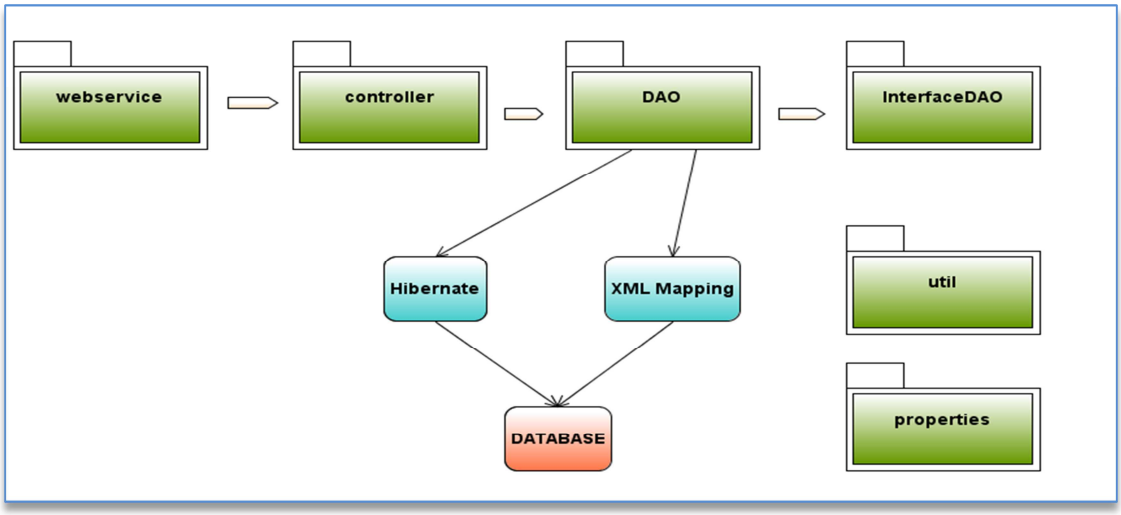


Package que contiene la clase webservice, que contendrá todos los webmethods.



5.2.2. Relación de Package

A continuación mostramos los Package generales de la aplicación, que son muy comunes en el IDE NetBeans y que representa las configuraciones generales del proyecto.



5.2.3. Descripción de Clases

Fichero de la capa de persistencia.

Parámetros para la conexión con la base de datos

```
<hibernate-configuration>
  <session-factory>
    <!-- Database connection settings -->
    <!-- nombre de la base de datos TFC_Cine-->
    <property name="dialect">org.hibernate.dialect.PostgreSQL82Dialect</property>
    <property name="connection.driver_class">org.postgresql.Driver</property>
    <property name="connection.url">jdbc:postgresql://localhost:5432/TFG_Cine</property>
    <property name="connection.username">postgres</property>
    <property name="connection.password">1234</property>
    <property name="connection.pool_size">100</property>
    <property name="transaction.factory_class">org.hibernate.transaction.JDBCTransactionFactory</property>
    <!-- <  SQL dialect - generate SQL for a particular database -->
    <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
    <!-- Echo all executed SQL statements -->
    <property name="show_sql">true</property>
    <property name="hibernate.format_sql">true</property>
    <!--    <property name="hibernate.jdbc.batch_size">0</property>
       <property name="hibernate.hbm2ddl.auto">update</property>-->
    <property name="hibernate.current_session_context_class">thread</property>
```

Fichero XML, para el mapeo de la información

```
    <!--Archivos de mapeo beans -->
    <mapping resource="uoc/tfg/ofertacinema/bean/Cine.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Descuento.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Sala.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Servidor.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Sesion.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Pelicula.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Genero.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Proyeccion.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Precio.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Fecha.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Idioma.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Interprete.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Ventas.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/Cliente.hbm.xml"/>
    <mapping resource="uoc/tfg/ofertacinema/bean/VentasCliente.hbm.xml"/>
```

A continuación se detalla el fichero Cine.hbm.xml.

```
<hibernate-mapping>
    <class name="uoc.tfg.ofertacinema.model.Cine" table="cine">
        <id column="idcine" name="idcine">
            <generator class="increment">
                <param name="sequence">idcine_seq</param>
            </generator>
        </id>
        <property column="codigocine" name="codigoCine" type="string"/>
        <property column="nombrecine" name="nombreCine" type="string"/>
        <property column="telefono" name="telefono" type="string"/>
        <property column="direccion" name="direccion" type="string"/>
        <property column="longitud" name="longitud" type="double"/>
        <property column="latitud" name="latitud" type="double"/>
        <property column="fechabaja" name="fechaBaja" type="date"/>
        <property column="fechacreacion" name="fechaCreacion" type="date"/>
    </class>
</hibernate-mapping>
```

A continuación se detalla la clase DAO de CineDAO.

```
public class CineDAO  implements  ICineDAO{

    private Session sesion;
    private Transaction tx;

    private void iniciaOperacion() throws HibernateException {
        try{
        sesion = CineController.getSessionFactory().openSession();
        tx = sesion.beginTransaction();
        }catch(Exception ex){
            ex.toString();
        }
    }

    private void manejaExcepcion(HibernateException he) throws HibernateException {
        tx.rollback();
        throw new HibernateException("Ocurrió un error en la capa de acceso a datos", he);
    }
```

```
public interface ICineDAO {

 //obtiene lista de cines
    public List<Cine> obtenerListaCines() throws HibernateException;

//obtiene  cine  por el codigo
    public Cine obtenerCineByCode(String codigoCine) throws HibernateException;

//inserta un cine nuevo a traves de un objeto cine
    public int insertarCine(Cine cine) throws HibernateException;

//comprueba si hay modificaciones , bajas o nuevos cines  a traves de sus codigos
    public List<Item> comprobarCambiosEnCines(String codigosCine) throws HibernateException;

//baja de un cine - actualiza la fecha baja a traves del codigo cine
    public boolean bajaCineByCode(String codigoCine) throws HibernateException;

}
```

A continuación se detalla la clase webservice, como se implementa los webmethods que son llamados desde el aplicativo Android.

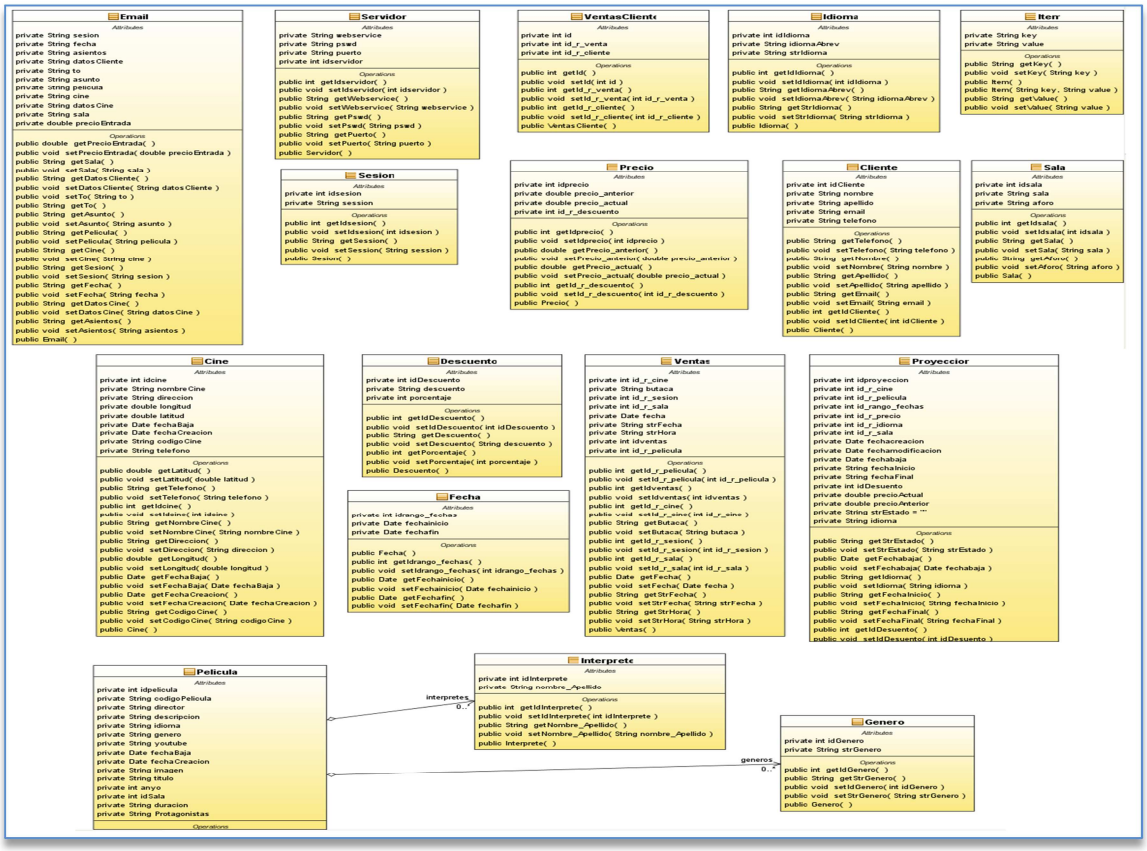
```
/**
 * comprueba los parametros que envia el cliente mobile con el servidor
 *
 * @param strWebservice
 * @param strPswd
 * @return
 */
@WebMethod(operationName = "webMethod_ComprobarServer")
public String webMethod_ComprobarServer(@WebParam(name = "strWebservice") String strWebservice, @WebParam(name = "strPswd")
    int ivalue = -2;
    String strValue = "";
    try {
        ivalue = servidorDAO.ComprobarServer(strWebservice, strPswd);
        if (ivalue > 0) {
            strValue = "OK_SERVER";
        } else {
            strValue = "ERROR_SERVER";

        }
    } catch (Exception ex) {
        strValue = ex.toString();
        ex.toString();
    }
```

5.2.4. Relación de Clases

.

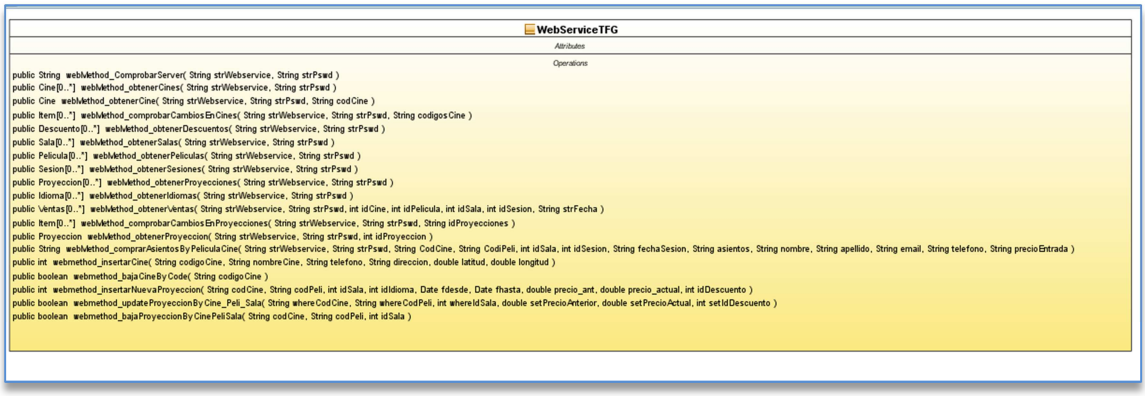
- **Entidades**

  Tal como comentamos en el apartado del cliente, son clases que nos facilita enormemente el transporte de la información, evitando que se envíen gran cantidad de parámetros a un método cuando queremos hacer un registro o actualización



- **Webservice**

En esta clase es donde se implementa las funcionalidades de los webmethods para el intercambio de datos con la aplicación en Android.



- **DAO**

Las clases DAO (Data Access Object), son clases donde reside la lógica de manejo de Hibernate, de esta forma conseguimos que nuestra lógica de negocio no sepa nada de Hibernate, y siempre que quiera acceder a los datos lo hará usando esta clase. Adicionalmente se van a trabajar con archivos de mapeo XML.

A continuación mostramos todas las clases DAO del Package DAO:

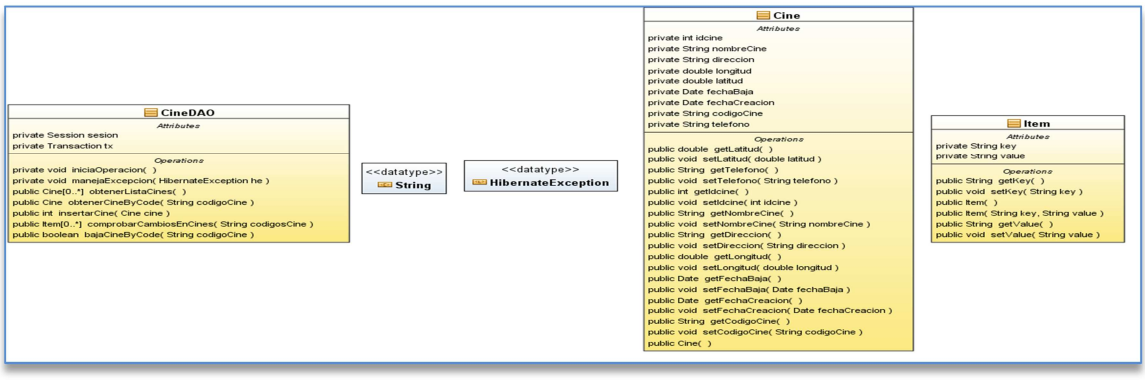


Diagrama de la clase Cine

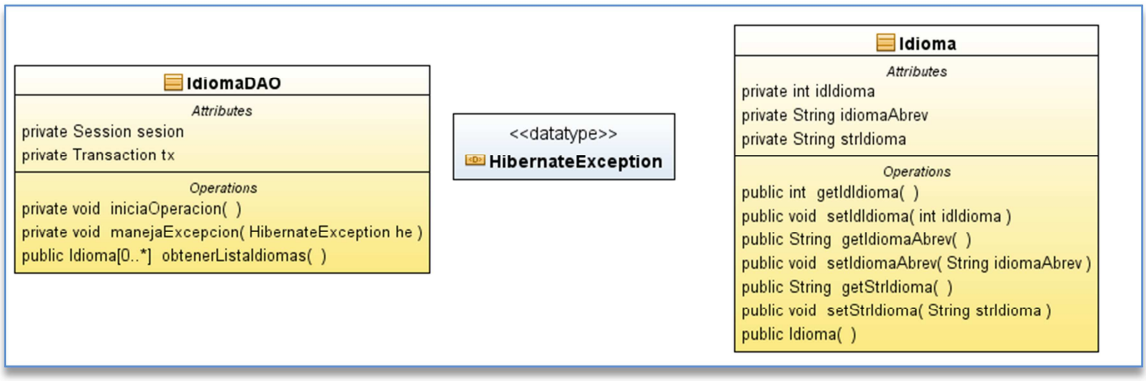


Diagrama de la clase Idioma

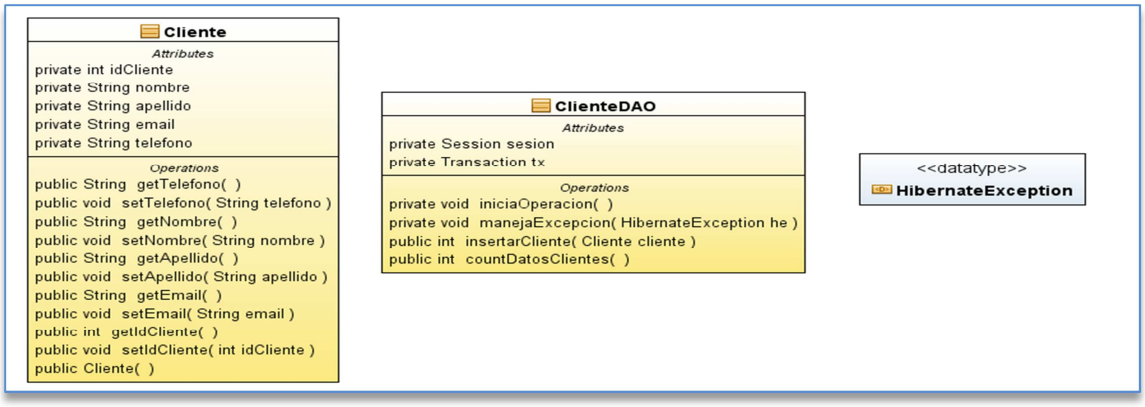


Diagrama de la clase Cliente

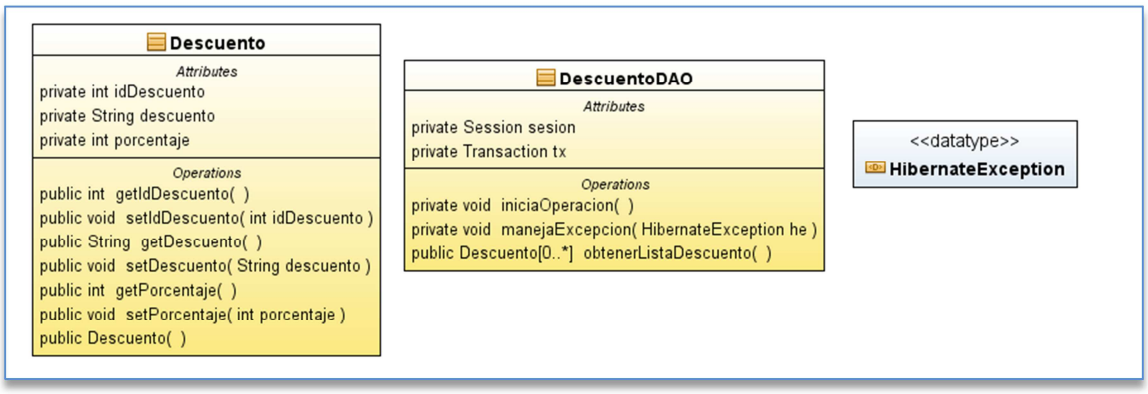


Diagrama de la clase Descuento

Diagrama de la clase Película

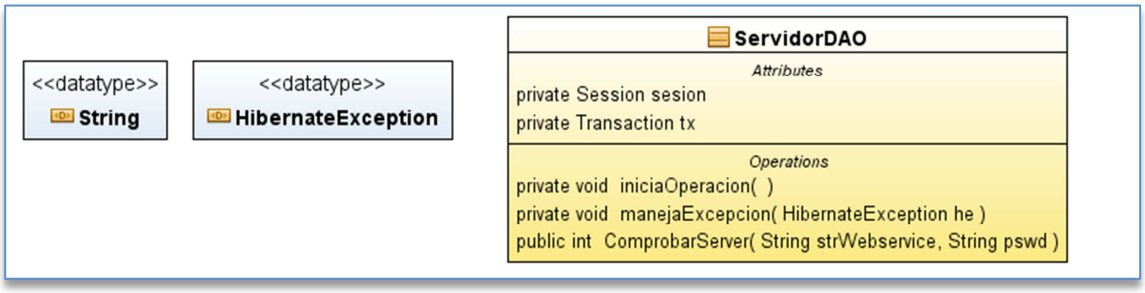


Diagrama de la clase Servidor

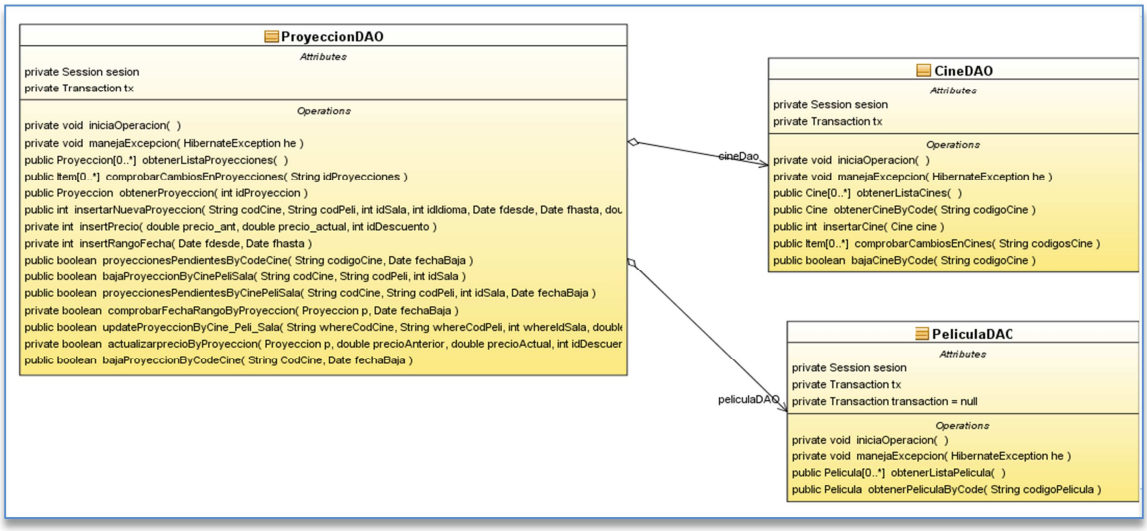


Diagrama de la clase Proyección

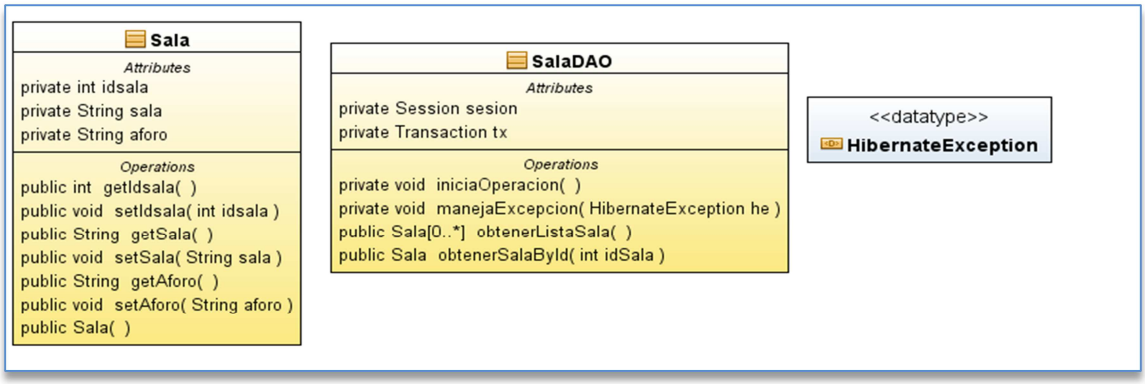


Diagrama de la clase Sala

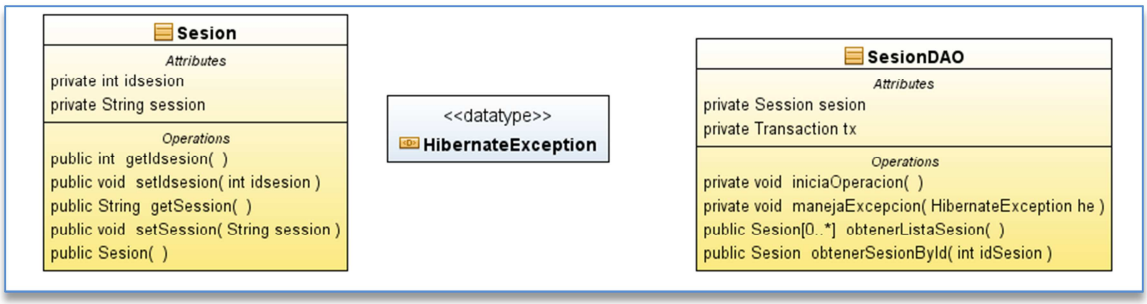


Diagrama de la clase Sala

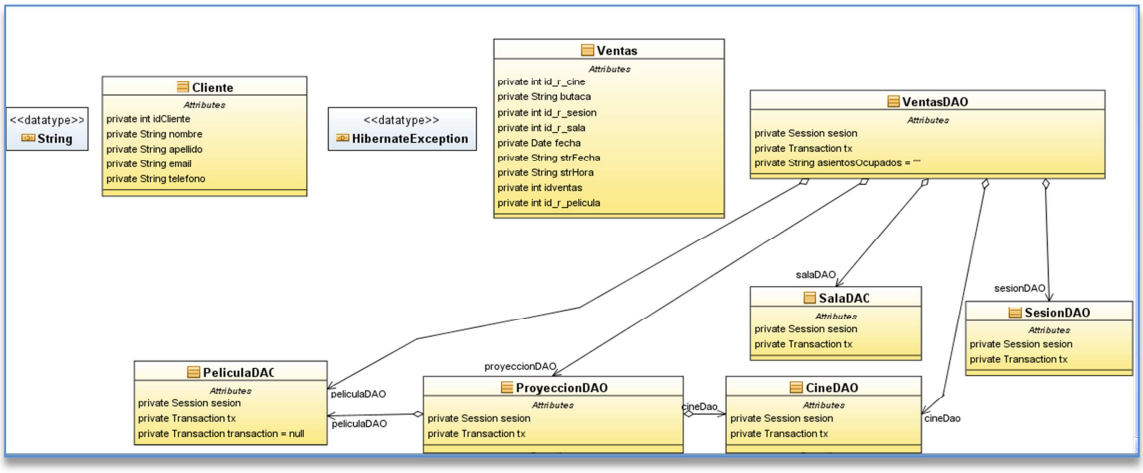


Diagrama de la clase Ventas

6. POSIBLES MEJORAS

A continuación se detalla una posibles mejoras que podría tener el aplicativo, a fin de realizarlo más competitivo en el mercado.

- **Gestionar los datos de la tarjeta**: Cuando se realiza una compra, el aplicativo requiere de los datos de tarjeta (crédito o débito) del titular, estos datos serán enviados a través de una pasarela a una empresa externa, así se obtendrá mayor seguridad en con estos datos.
- **Compatibilidad con todos los dispositivos**: Se podría implementar las interfaces para todos los dispositivos móviles Android (Samsung, Sony, LG, etc.) e incluyendo las tabletas.
- **Añadir opción de Idioma**: Esta funcionalidad es fácil en implementarlo en el aplicativo (añadiendo ficheros XML para cada idioma), lo más delicado será en poder obtener los datos de las base de datos, diferenciándolos por el idioma seleccionado en el aplicativo.
- **Añadir solo reservas**: Se podría añadir una funcionalidad de reservas de butacas por un tiempo determinado y con previo aviso de que la reserva se finalizará.
- **Compatibilidad con cualquier base de datos**: Se podría modificar la capa de persistencia para poder conectarse a diferentes gestores de base de datos, esto requeriría modificar algunas sintaxis de ciertas consultas SQL.

7.CONCLUSIONES

A nivel personal, el proyecto me ha permitido mejorar mis conocimientos generales sobre el entorno de desarrollo para dispositivos móviles, que hoy en día es una de las tecnologías que están evolucionando el mercado. El desarrollar un proyecto desde sus inicios, me ha llevado organizarme y a ocupar varios roles en el campo de la informática, lo cual ha sido muy fructífero a nivel de conocimiento y organización.

Los principales escoyos con los que me he encontrado a lo largo del proyecto, han estado localizados principalmente en dos fases:

La fase de diseño: Es donde he tenido muchas dudas, en como plantear algunas interfaces que sean muy intuitivos para el usuario común y muy fácil de saber cuál será su funcionalidad.

La fase de implementación: En algunos casos he tenido que volver a replantear algunas funcionalidades, debido que no era muy apropiados para su funcionamiento (optimización de código).

Como se trataba de dos proyectos (Cliente - Servidor), el escoyo de mayor envergadura era en la comunicación y envió de datos, para ello era necesario utilizar un dispositivo móvil para realizar las pruebas (Móvil- PC).

Otro aspecto en el que se ha empleado bastante tiempo es en la estructura de la base de datos, aspectos muy importantes en la lógica de negocio, que si bien no tiene dificultad técnica entrañan bastante dificultad lógica. En este caso, la estructura de la base datos eran muy diferentes tanto para la parte Cliente como para el Servidor.

En resumen, las conclusiones generales de la implementación son que se ha consumido mayor tiempo debido que se trataba de un proyecto Cliente- Servidor, y esto correspondía a implementar dos proyectos conjuntados, teniendo en cuenta que la comunicación entre Cliente-Servidor debía ser ligero.

8. GLOSARIO

Android. Sistema operativo móvil basado en Linux, diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas

Aplicación. Programa informático diseñado como herramienta para realizar uno o varios trabajos determinados.

Webservice. Es una colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones.

Webmethod. Es un método el cual se expone usando el atributo webMethod para exponerlo vía XML (en un webservice).

IDE. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica

Google Maps. Servicio de Google que ofrece imágenes de mapas.

Eclipse. Entorno de desarrollo integrado de código abierto.

NetBeans. Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

Framework. Es un marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a muy gran escala de componentes software para el desarrollo rápido de aplicaciones

Hibernate. Es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java.

Persistencia. Es un Framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).

DAO. Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos

XML. Es un sistema estándar de codificación de información.

SOAP. Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

9. BIOGRAFÍA

9.1. Publicaciones

Dave Minter; Jeff Linwood (2005). *Hibernate - Beginning Hibernate From Novice to Professional.*

Patrick Peak; Nick Heudecker (2006). *Hibernate Quickly*.

Christian Bauer; Gavin King (2005). *Hibernate in Action*.

O'Reilly Media (2012). *Head First Android Development.*

9.2. Artículos

Consumiendo Web Service SOAP: http://androideity.com/2011/11/16/consumiendo-web-service-soap-json-con-android-ii/

Llamando a un servicio Web: http://expertnotfound.wordpress.com/2011/11/05/llamando-a-un-servicio-web-desde-android/

Acceso a Servicios Web SOAP: http://www.sgoliver.net/blog/?p=2594

Google Maps API: http://www.elandroidelibre.com/2013/10/desarrollando-en-android-2-google-maps-api.html

9.3. Referencias Consultadas

Google Maps API. https://developers.google.com/maps/

Google YouTube API. https://developers.google.com/youtube/v3/

Stack Overflow. http://stackoverflow.com/

Developer Android. http://developer.android.com