Memoria Trabajo Final de Grado

MyTopRoom

Aplicación para Android:
“Buscador de ofertas de Hoteles"

Trabajo final de Grado de Ingeniería Informática
Ingeniería del Software – 1er semestre 2013

Consultor: Juan José Cuadrado Gallego
Alumno: Isabel Guerra Monclova

Fecha de entrega: 08 de Enero de 2014

A mis mejores amigas,
tanto las que me animaron
a empezar este proyecto
como las que me han apoyado
y empujado a finalizarlo,
y por supuesto a mi madre,
quien siempre me apoya
incondicionalmente
en todo lo que hago
"¡Mil gracias! ¡Os quiero!"

Ficha del Trabajo final de Grado

Título del Trabajo	My Top Room Aplicación para Android: “Buscador de ofertas de Hoteles"
Nombre del Consultor	Juan José Cuadrado Gallego
Nombre del Alumno	Isabel Guerra Monclova
Fecha de entrega (mm/aaaa)	01/2014
Área del Trabajo final	Ingeniería del Software
Titulación	Grado de Ingeniería Informática

Resumen del Trabajo

El objetivo de este proyecto es el desarrollo de una aplicación móvil para dispositivos con sistema operativo Android, que permita la búsqueda de ofertas de Hotel de una forma diferente a las actuales.

Para ello se ha empleado el modelo del ciclo de vida en cascada, con las fases de análisis, diseño, construcción y pruebas del sistema desarrollado. El software final sigue una arquitectura de tipo cliente/servidor y ha sido realizado con Java como lenguaje base de programación, haciendo uso de algunas librerías como Apache HTTP Request para las conexiones con el servidor remoto como las propias de Android, que facilitan la creación de interfaces gráficas y la gestión de los recursos de los dispositivos en el desarrollo de aplicaciones móviles.

Summary in English

The project's aim is to develop a mobile application for devices with Android as platform, which allows searching for Hotel offers in a different way than the current applications.

In order to do that, it has been applied the waterfall model as design process, performing the phases of analysis, design, construction and testing on the software developed. The final product follows a client/server architecture and it has been implemented with Java as the base programming language, using some of its libraries like Apache HTTP Request to manage the connections to the remote server and also the Android libraries and components, which facilitate to the programmer the creation of the user graphic interfaces and the management of the devices resources when developing mobile applications.

Palabras clave

Cliente/servidor, Java, Android, HTTP Request, SQLite, Thread

Tabla de Contenidos

Indice de Figuras

Indice de Tablas

1. Planificación del Proyecto

1.1 Contexto

La empresa “IBTOURS, S.L.” es una de las compañías europeas líder en reservas de alojamiento online porque tiene distintos sitios Web donde el consumidor final puede localizar y reservar hoteles. Las claves para dar un buen servicio son en primer lugar la velocidad de sus motores de disponibilidad, y en segundo contar con acuerdos con el mayor número de hoteles para obtener de forma directa la información de su disponibilidad y precios.

Estos sitios Web son distintas marcas de la propia empresa, cada una con objetivos distintos según al cliente que va enfocado. Por ejemplo, un sitio puede estar enfocado a hoteles familiares, otro puede ser de hoteles en una región específica, otro de hoteles de lujo, etc.

Actualmente cuando una persona quiere reservar un hotel, tiene que ir a uno de los sitios Web, según el tipo de alojamiento que busque, introducir los campos de búsqueda como número de personas, fechas o destino, buscar el Hotel que le guste entre la lista de disponibles y reservarlo en ese momento.

1.2 Objetivos

1.2.1 Objetivo general

El proyecto surge por la necesidad de ofrecer a los clientes una nueva forma de reservar hoteles que pueda atraer a más clientes y que ayude a entender mejor lo que demandan, dónde y cuándo.

El objetivo principal de este proyecto es dar un nuevo mecanismo de búsqueda a los clientes a través de una aplicación móvil “MyTopRoom”, para estar más cerca del cliente y poder aprovechar la interacción que te ofrecen los dispositivos móviles actuales.

En esta nueva forma de búsqueda, el cliente también enviará lo que busca: destino, hotel, fechas, pero podría informar también del precio, una vez envía la petición, será la aplicación móvil la que le avise cuando se encuentre una oferta que se adecúe a lo que buscaba en lugar de ser el cliente quién tiene que buscar cada vez.

De esta forma y con la implantación de este proyecto se pretende ofrecer un elemento diferenciador de los buscadores actuales para llegar a más clientes, que permite a la empresa tener una nueva fuente de información donde estudiar la demanda y así contribuir a que se puedan mejorar las ofertas que se muestran en los distintos sitios Web.

1.2.2 Objetivos específicos

Existen otras aplicaciones similares en el mercado –trivago, booking, rastreator (hoteles),…- pero en todas ellas el cliente tiene que buscar y reservar en ese momento, así que con esta nueva forma dónde el usuario recibe la oferta cuando el sistema la encuentra se persigue que la primera aplicación móvil que lanza la empresa tenga un elemento diferenciador y pueda ser competitiva.

Otro objetivo es poder contar con otro espacio de publicidad que la empresa pueda ofrecer tanto a hoteleros como empresas y así tener otra fuente de ingresos.

En último lugar se pretende actualizar tecnológicamente a la empresa con el desarrollo de una aplicación móvil que permita lanzar otras aplicaciones móviles para el resto de sitios Web.

El desarrollo e implantación de esta aplicación debe asegurar el cumplimiento de la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, y contar con las condiciones de calidad y seguridad que impone la tienda de software en línea “Google Play”, ya que desde ahí se distribuirá la aplicación.

1.3 Alcance del proyecto

1.3.1 Descripción de necesidades

El desarrollo de la aplicación móvil “MyTopRoom” tiene que contemplar inicialmente los siguientes requerimientos.

El proceso comienza cuando el usuario descarga de la tienda “Google Play” la aplicación y la instala en el dispositivo móvil. Solo puede haber una aplicación por dispositivo.

Al abrir la aplicación, si el usuario no se ha dado de alta con anterioridad no existirá cuenta asociada al dispositivo y por tanto no puede utilizar la aplicación. El sistema permitirá al usuario crear una cuenta y darse de alta.

Solo puede haber una cuenta por dispositivo, para evitar la creación de cuentas masivas de usuarios. Cuando el usuario crea la cuenta por primera vez, el sistema detectará si ya existe una en el dispositivo y si es el caso pedirá al usuario si desea eliminar la actual y asociar la nueva. Si el usuario acepta, la cuenta anterior se elimina y se crea la nueva.

Al crear la cuenta, el usuario accede y el sistema solicitará los datos de perfil, que son la moneda en la que quiere ver el resultado.

En los siguientes accesos, el sistema mostrará directamente la página principal de la aplicación.

En esta página principal, el sistema mostrará el listado de las últimas ofertas encontradas. Si no hay ninguna, el sistema mostrará la opción para que el usuario pueda proceder a realizar una búsqueda.

El sistema ofrecerá al usuario un menú de navegación rápida para poder acceder de forma sencilla a la página principal, la página para realizar búsquedas, la página para consultar las ofertas obtenidas y la página de las búsquedas enviadas.

El usuario podrá enviar hasta veinte búsquedas diferentes. Para cada búsqueda, el usuario deberá informar del destino, rango de precio aproximado, número de noches, número de adultos, número de niños y sus edades y el rango de estrellas de hotel.

El sistema almacenará las búsquedas en el motor de disponibilidad y las asociará a la Cuenta del usuario grabándolas en la base de datos del dispositivo.

La aplicación se conectará al motor de disponibilidad cuando el usuario accede y si se detecta que hay un hotel disponible que cumple las condiciones de búsqueda del usuario, descargará las ofertas y las mostrará en el dispositivo.

Las ofertas que se reciban serán agrupadas por destino, rango de precios y número de estrellas. El sistema ofrecerá al usuario distintos filtros que facilite la lectura de los hoteles encontrados.

Las búsquedas que no tengan resultados se almacenarán seis meses. Si pasado ese tiempo no se ha detectado ninguna oferta, se eliminarán del dispositivo y del motor de disponibilidad.

Por último el sistema ofrecerá por cada oferta de hotel el enlace al sitio web donde se encuentra. Si el usuario está interesado en una oferta, podrá pulsar el enlace, ir al sitio web y reservarla en ese momento. Las ofertas que lleven más de seis meses en el dispositivo, se eliminarán del dispositivo. El enlace incluirá el identificativo de la aplicación móvil y así en los casos en los que el usuario confirme la reserva en el sitio Web correspondiente, se podrá comprobar que ha llegado mediante esta nueva aplicación.

1.3.2 Requerimientos funcionales iniciales

En base al alcance definido la aplicación debe contemplar las siguientes funcionalidades divididas en los siguientes módulos:

- Módulo de conexión

Es el módulo que ejecuta todo usuario para acceder a la aplicación. Permite a los usuarios crear una cuenta o acceder a través de los datos de una cuenta existente.

También permite a los usuarios consultar o modificar los datos de acceso y se encarga de actualizarlos en el motor de disponibilidad y en el dispositivo.

- Módulo de usuario

Se encarga de la gestión de los datos de perfil del usuario, tanto su modificación como actualización en el motor de disponibilidad. En una primera versión solo constará de la moneda en la que quieren obtenerse los resultados, pero se prevé incorporar otros datos como el idioma y país de origen

- Módulo de Búsquedas

El objetivo principal de este módulo es el de ofrecer las funcionalidades relacionadas con las búsquedas de hoteles como enviar las búsquedas al motor de disponibilidad o eliminar las caducadas.

- Módulo de Ofertas

Por último este módulo se encarga de toda la gestión de ofertas. Es el encargado de solicitar al motor de disponibilidad las posibles ofertas del usuario, notificarlas al usuario, visualizarlas, ofrecer distintas opciones de búsqueda sobre las ofertas recibidas, redirigir a la página web original de la oferta cuando el usuario está interesado en reservar una de ellas y eliminar las ofertas caducadas.

1.3.3 Funcionalidades propuestas para nuevas versiones

Con el fin de aprovechar una fuente nueva de ingresos por publicidad, se propone incorporar una sección de publicidad en la parte superior de la aplicación, dónde se muestren distintos anuncios relacionados con los proveedores de hoteles y otros servicios de la empresa.

Para mejorar la relación con los clientes, para aquellos que reservan a través de esta aplicación y una vez confirmada la reserva, se propone que la aplicación descargue para el usuario cupones de servicios gratuitos o de descuentos, que serían asignados previamente por el motor de disponibilidad (spa gratis en hotel, 2x1 restaurantes de hotel, 2 x 1 en habitación de hotel, etc.) en base al número y tipo de ofertas reservadas.

Implementar la aplicación para el resto de sistemas operativos con el objetivo de que la aplicación sea instalable en más dispositivos.

1.4 Entorno tecnológico

Cuando se crea una aplicación Web se deben tomar decisiones técnicas previas como por ejemplo sobre qué sistema operativo se puede instalar, si es necesaria una versión concreta del navegador o si funciona sobre una máquina virtual. En el caso de aplicaciones móviles, es imprescindible decidir sobre qué sistemas operativos se puede instalar porque esto limita el número de marcas de dispositivos móviles que podrán descargar la aplicación y por tanto limita a los clientes a los que llega.

Entre los sistemas operativos más utilizados está Android, iphone OS, Symbian, Windows mobile y Blackberry OS, pero es sin duda Android desarrollado por Google el que tiene hoy en día la cuota de mercado más alta[1], gracias a sus características como sistema abierto y otras como su entorno de ejecución o el marco de trabajo de sus aplicaciones, siendo elegido por fabricantes como Samsung, HTC, Motorola y LG para sus terminales móviles. En consecuencia, esta primera aplicación móvil que lanzará la compañía será desarrollada para Android.

Una vez escogido el sistema operativo, se deciden las siguientes herramientas de desarrollo:

- *Android SDK,* que cuenta con un conjunto de librerías (API) necesarias para la construcción, testeo y depuración de aplicaciones Android.
- *ADT Plugin –Android Development Tools-,* plugin para el IDE Eclipse, que permite integrarse con este entorno de desarrollo
- *JDK (Java Development Kit),* necesario para el funcionamiento de Eclipse, que se utiliza como entorno de desarrollo para compilar y ejecutar aplicaciones.

Por otro lado, para la implementación de la aplicación móvil, se utilizarán las siguientes tecnologías:

- *JSON*, formato ligero para el intercambio de datos que no requiere el uso de XML y que será utilizado para devolver el contenido de las peticiones al motor de disponibilidad
- *SQLLite,* base de datos relacional Open Source que sólo necesita una pequeña cantidad de memoria en su ejecución (Benbourahala, 2013) y que se utilizará para crear base de datos en el dispositivo y guardar los datos que deben ser persistentes.

1.5 Organización del proyecto

A lo largo de los últimos treinta años se han definido distintos métodos de ingeniería del software para organizar los proyectos. Cada uno de ellos define uno o más procesos de desarrollo que definen qué tareas deben llevarse a cabo, en qué orden, qué roles hacen falta y cuál es su responsabilidad, y qué artefactos se usarán como punto de partida de cada tarea y cuáles se entregarán al finalizar cada una de ellas.

La elección del método de desarrollo del software depende de las características del producto que desarrollamos y el enfoque que queramos darle. Aunque existen muchos, podemos agruparlos en tres grandes familias:

[1] **Bicheno, Scott** (2013, 31 de octubre) "[…]Android Captures Record 81 Percent Share of Global Smartphone Shipments in Q3 2013" [artículo en línea] [Fecha de consulta: 1 de diciembre del 2013] < http://blogs.strategyanalytics.com/WSS/post/2013/10/31/Android-Captures-Record-81-Percent-Share-of-Global-Smartphone-Shipments-in-Q3-2013.aspx>

- Los que siguen el ciclo de vida en cascada, adecuada para proyectos con un objetivo claro y donde se conocen los detalles de cómo será la solución
- Los métodos iterativos o incrementales, para aquellos donde no sabemos a priori, cómo será el resultado final del desarrollo y necesitamos un método que facilite el descubrimiento de la solución mediante la obtención de información empírica lo mejor posible, y
- Los métodos ágiles, que son un conjunto de métodos de desarrollo iterativos donde se da menor importancia al trabajo previo de análisis y se potencia el cambio y la adaptación.

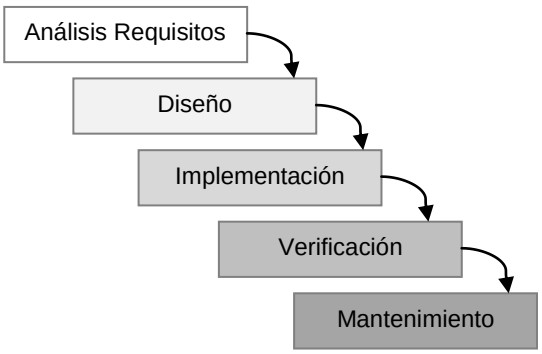
Analizando este proyecto donde tenemos un objetivo claro y conocemos el detalle de la solución que queremos implementar y con un modelo lineal secuencial, el enfoque metodológico que emplearemos será en cascada, cuyos procesos definimos en el siguiente apartado.

1.5.1 Ciclo de vida clásico o en cascada

La manera de organizar el desarrollo es a través de una secuencia definida de acontecimientos y de los resultados que debe proporcionar cada uno de ellos. Dichos acontecimientos quedan divididos en las siguientes etapas (Pradel, 2011):

1. Requisitos: define qué debe ser y hacer el producto que se quiere desarrollar
2. Análisis y Diseño: en esta etapa el resultado será definir cómo debe ser el producto tanto desde el punto de vista externo “qué hace el sistema” como el interno “de qué componentes está formado”
3. Implementación: es la etapa en la que se escribe el código, los manuales de instalación y se genera el producto ejecutable
4. Pruebas: esta etapa ayuda a verificar que el producto desarrollado se corresponde con los requisitos
5. Mantenimiento: etapa en la que el sistema se pone a disposición de todos los usuarios y se producirán cambios porque haya defectos u otros errores, o bien por mejoras de nuevas o de las actuales funcionalidades.

El método es “en cascada” porque el comienzo de cada una de las etapas mencionadas, no puede comenzar hasta la finalización de la inmediatamente anterior y el punto de partida de cada una, es el entregable de la anterior.



Para el desarrollo de este proyecto software se obviará la etapa de mantenimiento.

1.5.2 Descomposición en actividades (WBS)

La estructura de descomposición del proyecto (EDP) o "Work Breakdown Structure" (WBS) es una división natural del proyecto en las distintas actividades, para llegar al producto final y con la finalidad de facilitar la planificación, organización y control de los trabajos que se van completando.

En base a las fases del ciclo de vida en cascada escogido para el desarrollo y las entregas de las distintas PEC se propone el siguiente WBS definido con tres niveles:

- **Proyecto Fin de Grado "MyTopRoom"**
  - **Planificación**
    - Propuesta del proyecto
    - Aprobación de la propuesta
    - Introducción y objetivos del proyecto
    - Entorno tecnológico
    - Metodología y organización de tareas
    - Planificación temporal (diagrama Gantt)
    - Elaboración documentación
    - Entrega PEC1
  - **Análisis de requisitos y diagrama de clases**
    - Especificación requisitos funcionales por módulo
    - Especificación requisitos no funcionales
    - Actores y casos de uso
    - Diagramas de clases gestoras, entidades y jerarquías
    - Requisitos interfaz de usuario y prototipos de pantalla
    - Elaboración documentación
    - Entrega PEC2
  - **Diseño**
    - Clases del sistema (fichas CRC)
    - Diseño arquitectura de componentes
    - Diseño capas y clases
    - Diseño de persistencia
    - **Implementación**
      - Desarrollo de clases
      - Desarrollo de interfaz gráfica
      - Elaboración manuales de instalación
    - **Pruebas**
      - Pruebas unitarias por módulo
      - Pruebas de integración
    - Elaboración documentación
    - Entrega PEC3
  - **Memoria y presentación**
    - Elaboración memoria
    - Elaboración presentación
    - Elaboración competencia transversales
    - Entrega memoria y presentación
  - **Tribunal Virtual**
    - Inicio de preguntas
    - Entrega respuestas

1.5.3 Planificación del proyecto

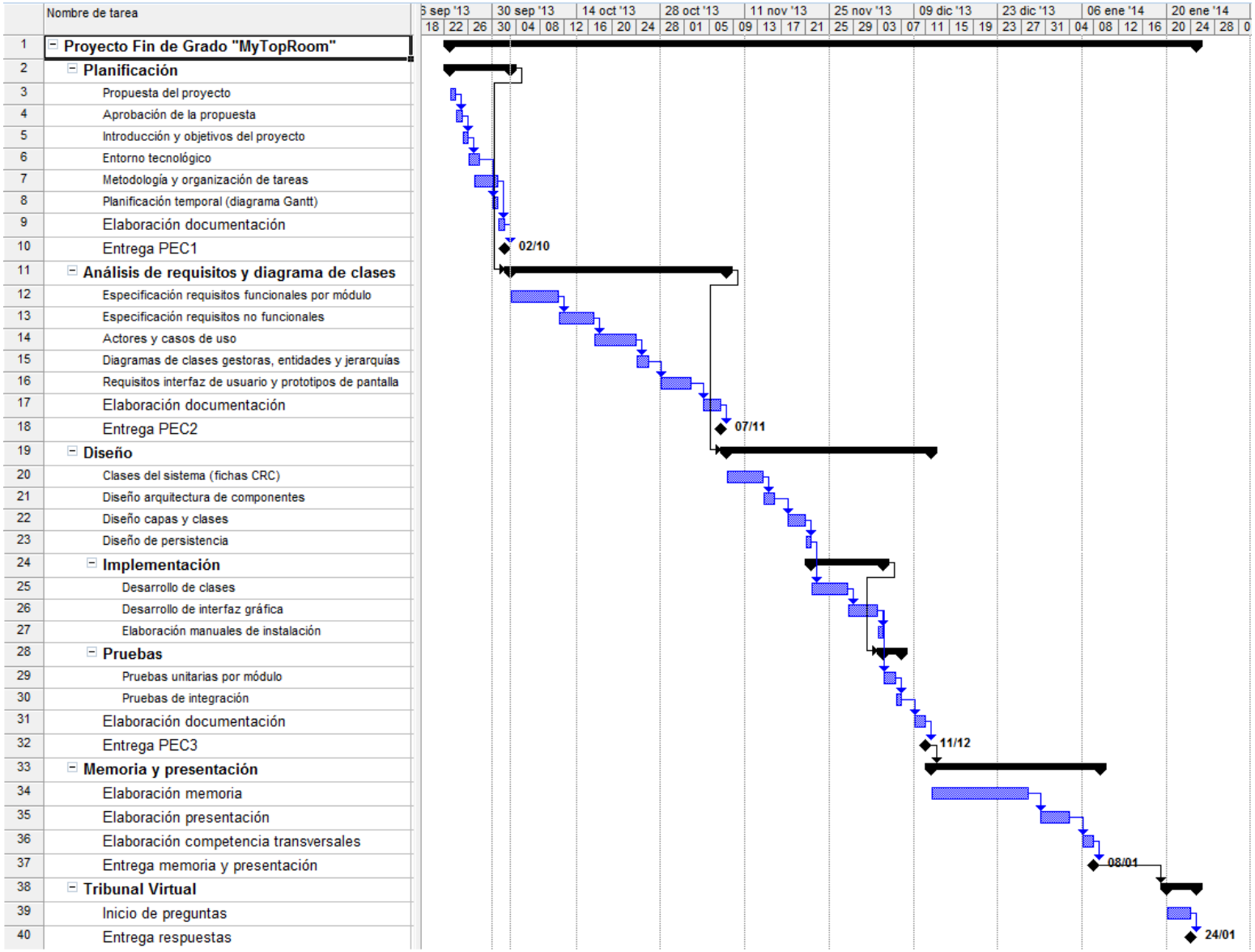
El objetivo de la planificación es obtener la distribución de actividades en el tiempo que garantice las condiciones de entregables exigidos.

1.5.3.1. Diagrama de Gantt

El diagrama de Gantt es una de las técnicas de planificación más comunes. En él están todas las actividades o tareas en las que el proyecto está dividido, junto con la duración de cada una de ellas y los hitos a cumplir para garantizar la finalización del proyecto.

En las siguientes figuras se muestra por un lado la planificación de las tareas y por otro el gráfico Gantt con la planificación y duración de las tareas.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Proyecto Fin de Grado "MyTopRoom"**	**90 días?**	**lun 23/09/13**	**vie 24/01/14**	
2	**Planificación**	**8 días?**	**lun 23/09/13**	**mié 02/10/13**	
3	Propuesta del proyecto	1 día	lun 23/09/13	lun 23/09/13	
4	Aprobación de la propuesta	1 día	mar 24/09/13	mar 24/09/13	3
5	Introducción y objetivos del proyecto	1 día	mié 25/09/13	mié 25/09/13	4
6	Entorno tecnológico	2 días	jue 26/09/13	vie 27/09/13	5
7	Metodología y organización de tareas	2 días	vie 27/09/13	lun 30/09/13	
8	Planificación temporal (diagrama Gantt)	1 día?	lun 30/09/13	lun 30/09/13	6
9	Elaboración documentación	1 día	mar 01/10/13	mar 01/10/13	7
10	Entrega PEC1	1 día	mié 02/10/13	mié 02/10/13	9
11	**Análisis de requisitos y diagrama de clases**	**26 días**	**jue 03/10/13**	**jue 07/11/13**	**2**
12	Especificación requisitos funcionales por módulo	6 días	jue 03/10/13	jue 10/10/13	
13	Especificación requisitos no funcionales	4 días	vie 11/10/13	mié 16/10/13	12
14	Actores y casos de uso	5 días	jue 17/10/13	mié 23/10/13	13
15	Diagramas de clases gestoras, entidades y jerarquías	2 días	jue 24/10/13	vie 25/10/13	14
16	Requisitos interfaz de usuario y prototipos de pantalla	5 días	lun 28/10/13	vie 01/11/13	15
17	Elaboración documentación	3 días	lun 04/11/13	mié 06/11/13	16
18	Entrega PEC2	1 día	jue 07/11/13	jue 07/11/13	17
19	**Diseño**	**24 días**	**vie 08/11/13**	**mié 11/12/13**	**11**
20	Clases del sistema (fichas CRC)	4 días	vie 08/11/13	mié 13/11/13	
21	Diseño arquitectura de componentes	2 días	jue 14/11/13	vie 15/11/13	20
22	Diseño capas y clases	3 días	lun 18/11/13	mié 20/11/13	21
23	Diseño de persistencia	1 día	jue 21/11/13	jue 21/11/13	22
24	**Implementación**	**8 días**	**vie 22/11/13**	**mar 03/12/13**	
25	Desarrollo de clases	4 días	vie 22/11/13	mié 27/11/13	23
26	Desarrollo de interfaz gráfica	3 días	jue 28/11/13	lun 02/12/13	25
27	Elaboración manuales de instalación	1 día	mar 03/12/13	mar 03/12/13	26
28	**Pruebas**	**3 días**	**mié 04/12/13**	**vie 06/12/13**	**24**
29	Pruebas unitarias por módulo	2 días	mié 04/12/13	jue 05/12/13	26
30	Pruebas de integración	1 día	vie 06/12/13	vie 06/12/13	29
31	Elaboración documentación	2 días	lun 09/12/13	mar 10/12/13	30
32	Entrega PEC3	1 día	mié 11/12/13	mié 11/12/13	31
33	**Memoria y presentación**	**20 días**	**jue 12/12/13**	**mié 08/01/14**	**32**
34	Elaboración memoria	12 días	jue 12/12/13	vie 27/12/13	
35	Elaboración presentación	5 días	lun 30/12/13	vie 03/01/14	34
36	Elaboración competencia transversales	2 días	lun 06/01/14	mar 07/01/14	35
37	Entrega memoria y presentación	1 día	mié 08/01/14	mié 08/01/14	36
38	**Tribunal Virtual**	**5 días**	**lun 20/01/14**	**vie 24/01/14**	**37**
39	Inicio de preguntas	4 días	lun 20/01/14	jue 23/01/14	
40	Entrega respuestas	1 día	vie 24/01/14	vie 24/01/14	39


Nombre de tarea
1 Proyecto Fin de Grado "MyTopRoom"
2 Planificación
3 Propuesta del proyecto
4 Aprobación de la propuesta
5 Introducción y objetivos del proyecto
6 Entorno tecnológico
7 Metodología y organización de tareas
8 Planificación temporal (diagrama Gantt)
9 Elaboración documentación
10 Entrega PEC1
11 Análisis de requisitos y diagrama de clases
12 Especificación requisitos funcionales por módulo
13 Especificación requisitos no funcionales
14 Actores y casos de uso
15 Diagramas de clases gestoras, entidades y jerarquías
16 Requisitos interfaz de usuario y prototipos de pantalla
17 Elaboración documentación
18 Entrega PEC2
19 Diseño
20 Clases del sistema (fichas CRC)
21 Diseño arquitectura de componentes
22 Diseño capas y clases
23 Diseño de persistencia
24 Implementación
25 Desarrollo de clases
26 Desarrollo de interfaz gráfica
27 Elaboración manuales de instalación
28 Pruebas
29 Pruebas unitarias por módulo
30 Pruebas de integración
31 Elaboración documentación
32 Entrega PEC3
33 Memoria y presentación
34 Elaboración memoria
35 Elaboración presentación
36 Elaboración competencia transversales
37 Entrega memoria y presentación
38 Tribunal Virtual
39 Inicio de preguntas
40 Entrega respuestas
6 sep '13
30 sep '13
14 oct '13
28 oct '13
11 nov '13
25 nov '13
09 dic '13
23 dic '13
06 ene '14
20 ene '14
02/10
07/11
11/12
08/01
24/01

2. Análisis de requisitos

Un requisito es una característica observable del sistema que satisface una necesidad o expresa una restricción que afecta al software que estamos desarrollando (Pradel, 2011) Por tanto es imprescindible entender correctamente todos los requisitos del sistema, de lo contrario desarrollaremos un producto que no hace exactamente lo que se requería.

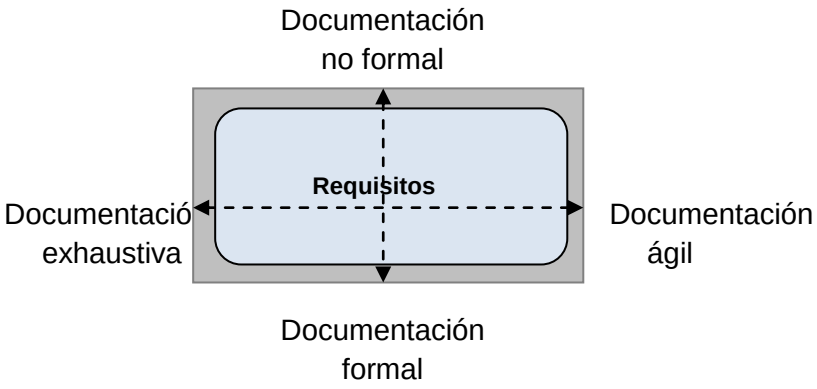
En el proceso de recopilación de requisitos, por un lado están los que expresan necesidades del cliente cuando solicita el software, pero por otro hay que obtener todos los requisitos de todas aquellas personas que puedan tener un impacto o interés en el sistema. Al conjunto de estas personas y el cliente se les conoce con el nombre de stakeholders. Por ejemplo un requisito que indique que la aplicación móvil debe ser fácil de usar, sin que el usuario tenga que leer un manual, no es un requisito del cliente, pero los usuarios finales son stakeholders de la aplicación y hay que tener en cuenta sus necesidades.

La técnica más básica para la obtención de requisitos es pedirlos a los stakeholders mediante alguna técnica creativa como brainstorming (tormenta de ideas) u otras como el modelado de roles de usuario para identificar los tipos de usuario del sistema, las listas predefinidas de requisitos o las entrevistas y cuestionarios (Pradel, 2012)
Una vez recogidos, como son necesidades de distintas personas involucradas, normalmente con roles o fines distintos, a veces ocurre que existen contradictorios entre lo que pide una u otra persona, o hay demasiados requisitos como para tenerlos todos en cuenta en una primera versión del desarrollo, entonces en este caso se pasa por un proceso de gestión de requisitos, en el que estima el coste de cada requisito, se priorizan según la importancia que tengan para los stakeholders y así seleccionar los que al final se tendrán en cuenta en la etapa de desarrollo (Pradel, 2012)

El siguiente paso es documentar los requisitos seleccionados y es un paso muy importante, ya que esta documentación sirve de base a los desarrolladores para implementar el sistema y además se utiliza como registro de lo que se ha acordado con los stakeholders.

Existen varios estilos de documentación de requisitos que dependen de las necesidades del proyecto en concreto, por ejemplo si lo más prioritario es la agilidad se documenta lo mínimo necesario y se confía en otros mecanismos de comunicación, o si lo es la exhaustividad, se documentaría el mayor número posible de requisitos con el mayor detalle posible (Pradel 2012)

Los distintos estilos se distinguen mediante dos ejes:



El eje horizontal define cómo de exhaustivo o de ágil, y el eje vertical clasifica las necesidades de documentación según si es necesario utilizar un lenguaje formal, como el lenguaje de especificación OCL –object constraint language- que garantice que la documentación es inambigua, consistente y verificable, o se puede usar el lenguaje natural de las personas que intervienen en el desarrollo (Pradel, 2012) En este proyecto para los requisitos se emplea una documentación ágil y poco formal mediante el uso del lenguaje natural.

Los requisitos se suelen clasificar en dos grandes grupos: los funcionales, que hacen referencia a la funcionalidad que debe proporcionar el sistema y los datos que debe conocer y almacenar, y los no funcionales que son aquellas cualidades o restricciones que debe cumplir el sistema, como por ejemplo la usabilidad, el rendimiento o la mantenibilidad.

2.1 Requisitos funcionales

Los requisitos funcionales se definen para cada uno de los módulos descritos en el capítulo de Introducción (apartado 1.3.1), con el objetivo de descomponer el desarrollo del software en partes más pequeñas que faciliten su construcción.

Como complemento a la descripción textual de los requisitos funcionales del sistema, se incluyen a modo de prototipo, las interfaces gráficas de usuario de cada una de ellas. El paradigma de desarrollo por prototipos es un medio de establecer los requisitos del software a partir de la visualización de un producto de software inicial, el cual se llama prototipo (Pressman, 2001)

En la siguiente figura se muestra el prototipo de la pantalla inicial de la aplicación, cuando el usuario la abre por primera vez y a continuación se describen las funcionalidades de cada módulo.

Figura 1. Prototipo de la pantalla inicial de la aplicación

2.1.1. Módulo de Conexión

Este módulo se encarga de gestionar la conexión de los usuarios y para ello el usuario debe acceder con su correo electrónico y contraseña. Si aún no dispone de Cuenta debe crearla primero y después acceder.

El sistema marcará por defecto la opción “Recordar” y así la siguiente vez que el usuario acceda no tendrá que volver a introducir las credenciales de nuevo y la aplicación se abrirá directamente en la pantalla principal Home. El usuario podrá en cualquier momento desconectar y salir de la Cuenta y en este caso la próxima vez que se ejecute la aplicación volverá a salir la página de Login.

Las funcionalidades de este módulo junto con sus prototipos de pantalla son las siguientes:

- **Crear una Cuenta**

Permite al usuario darse de alta para acceder a las funcionalidades de la aplicación y que el sistema pueda asociar la cuenta al dispositivo.
El usuario deberá informar del nombre de usuario, correo electrónico, contraseña y una pregunta y respuesta secreta que le servirá para recuperar la contraseña en caso de olvido.
Al crear la cuenta en el motor de disponibilidad se relaciona con el dispositivo y si el usuario intenta crear otra cuenta y se detecta que ya existe una relacionada con el dispositivo, el sistema dará de baja la cuenta anterior y relacionará esta nueva con la confirmación del usuario.

Para evitar que el usuario tenga que crear la cuenta y después introducir las credenciales para entrar, el sistema abrirá la pantalla principal de la aplicación una vez el usuario cree la cuenta.

A continuación se muestra el prototipo de pantalla con el formulario que el usuario debe cumplimentar para crear la cuenta:

Figura 2. Prototipo de la pantalla "Crear cuenta"

- **Acceder a la aplicación**

Al abrir la aplicación como podemos ver en la Figura 1, si el usuario no se ha logado o es la primera vez que se abre, se solicitará al usuario la dirección del correo electrónico y contraseña para acceder.
Por defecto se marcará la opción de recordar las credenciales para que el usuario no tenga que introducir los datos cada vez que abra la aplicación.

Esta funcionalidad permite que el sistema identifique al usuario como un usuario registrado y además al acceder se encargará de comprobar si hay ofertas en el motor de disponibilidad que están pendientes de descargar en el dispositivo móvil y si es así copiarlas en la base de datos de la aplicación y mostrarlas al inicio.

Figura 3. Prototipo de la pantalla "Home" al acceder con las credenciales

- **Recuperar la contraseña**

Esta funcionalidad permite al usuario recuperar la contraseña para acceder a la aplicación. Para ello el usuario deberá indicar el correo electrónico, la pregunta y respuesta secreta que seleccionó cuando creo la cuenta y si los datos son correctos el sistema logará al usuario y se mostrará la pantalla con los datos de acceso para que el usuario pueda cambiar la contraseña por una conocida.

Puede verse el enlace para recuperar la contraseña en el prototipo de la pantalla inicial de la aplicación en la Figura 1.

Figura 4. Prototipo de la pantalla "Recuperar contraseña"

❖ Consultar datos de acceso

Permite que el usuario pueda consultar los datos de acceso y si lo desea modificarlos, una vez ha accedido a la aplicación.
Esta funcionalidad y otras relacionadas con la configuración de la cuenta de usuario como la de consultar datos de perfil o desconectar, estarán disponibles desde la barra superior de la pantalla.

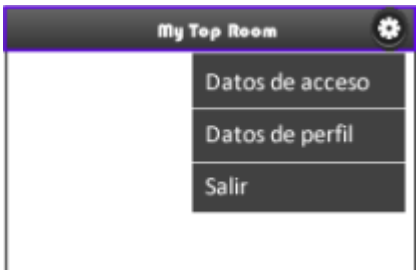


Figura 5. Prototipo de la pantalla del menú consultar datos de acceso, perfil y salir

❖ Modificar datos de acceso

Una vez que el usuario consulta sus datos de acceso puede modificarlos al cambiar su valor y pulsar el botón “Modificar” El sistema se encargará de actualizar la información en el motor de disponibilidad y en la base de datos del dispositivo.

El correo electrónico es el único dato de acceso que no se puede modificar dado que el motor relaciona la cuenta de correo con el dispositivo.

Figura 6. Prototipo de la pantalla “Modificar datos de acceso”

❖ Desconectar

Permite al usuario cerrar la sesión y salir de la aplicación de forma que la próxima vez que abra la aplicación tendrá que volver a introducir sus credenciales. De esta forma el usuario puede entrar con una cuenta distinta o crear una nueva.
La funcionalidad está disponible desde el botón “Salir” como se muestra en la Figura 5.

2.1.2. Módulo de Usuario

El módulo de Usuario se encarga de gestionar los datos de perfil asociados a cada cuenta, que no sean de acceso. En un primer momento solo incluye la moneda pero se prevé incorporar otros datos relevantes de usuario como el idioma.

Las funcionalidades son las siguientes:

- **Consultar datos del perfil**

Permite que el usuario pueda consultar los datos del perfil y si lo desea modificarlos. La funcionalidad está disponible desde el botón "Datos del perfil" que se muestra en la Figura 5.

- **Modificar datos del perfil**

Una vez que el usuario consulta sus datos de perfil puede modificarlos al cambiar su valor y pulsar el botón "Aplicar" El sistema se encargará de actualizar la información en la base de datos de la aplicación y el motor de disponibilidad.

El motor de disponibilidad se encargará de realizar la conversión de la moneda de las posibles ofertas a la moneda del perfil del usuario, con el tipo de cambio del día en que se obtengan. Si el usuario modifica la moneda después de haber obtenido ofertas, éstas no se cambiarán y seguirán con la misma moneda con la que se descargaron.

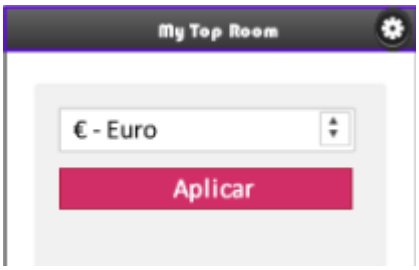


Figura 7. Prototipo de pantalla "Modificar datos del perfil"

2.1.3. Módulo de Búsquedas

Este módulo se encarga de facilitar las distintas opciones de búsquedas de ofertas de Hoteles y permite al usuario enviar estas búsquedas al motor de disponibilidad. También se encarga de eliminar aquellas búsquedas que no han recibido ninguna oferta y llevan más de seis meses en el motor.

Las funcionalidades del módulo son:

- **Seleccionar opciones de búsqueda**

Esta funcionalidad permite al usuario configurar las opciones personalizadas para enviar distintas búsquedas de hoteles.

Entre ellas se encuentra el destino, el rango de precio de la estancia, el número de noches, el número de adultos, número de niños y sus edades y el rango de estrellas de hotel.

Los rangos de precio podrán oscilar entre 10 y 2000. El número máximo de noches que se permitirá será de 31 y el rango de edades de los niños podrá variar entre 0 y 16 años.

Para el destino se facilitará un desplegable que ayude al usuario con los nombres de los destinos disponibles.

En todas las pantallas internas de la aplicación se incorpora un menú de acceso rápido a las principales funcionalidades y esta funcionalidad estará accesible desde el botón "Buscar"

Figura 8. Prototipo de la pantalla acceder a las opciones de búsqueda

- **Enviar una búsqueda**

Una vez que el usuario haya seleccionado las opciones de una búsqueda podrá enviarla al motor de disponibilidad a través de esta funcionalidad.
En la parte inferior de la pantalla de opciones de búsqueda tendrá el botón "Enviar" para proceder a su envío al motor.

Las búsquedas enviadas estarán sincronizadas tanto en el motor de disponibilidad como en la base de datos de la aplicación.
Si una búsqueda tiene más de seis meses de antigüedad en el motor y no se encontraron ofertas, el motor se encargará de eliminarlas y la próxima vez que el usuario inicie la aplicación, el sistema intentará sincronizar las búsquedas y también las eliminará de la base de datos.

Figura 9. Prototipo de pantalla de "Enviar una búsqueda"

❖ **Consultar búsquedas**

El usuario podrá consultar en todo momento las búsquedas que tiene pendientes y para las cuales no se han encontrado aún ofertas. Con esta funcionalidad el sistema le mostrará la lista de las búsquedas y también permitirá al usuario eliminarlas.

Esta funcionalidad estará accesible desde el botón "Búsquedas" del menú de acceso rápido.

Figura 10. Prototipo de la pantalla de "Búsquedas"

❖ **Eliminar una búsqueda**

El usuario deberá primero consultar la lista de búsquedas y desde la misma pantalla podrá eliminar las que desee pulsando el icono de papelera junto a cada oferta, como se muestra en la Figura 10. El sistema se encargará de eliminarla también del motor de disponibilidad.

2.1.4. Módulo de Ofertas

El módulo de ofertas se encargará de proporcionar todas las funcionalidades relacionadas con la gestión de ofertas recibidas, desde permitir su visualización como obtener las ofertas nuevas del motor de disponibilidad, redirigir al usuario a la página oficial de la misma o eliminar aquellas en las que el usuario ya no esté interesado.

Las funcionalidades de este módulo son:

- **Consultar ofertas**

Al acceder a la aplicación, el sistema comprueba si existen ofertas nuevas y en tal caso se descargan y se muestran al usuario en la pantalla principal, y con esta funcionalidad el usuario podrá acceder al listado de todas las ofertas recibidas. Además estará disponible desde el botón “Ofertas” en el menú de acceso rápido.

El listado de ofertas se mostrará ordenado por fecha desde la más reciente y desde esta pantalla el usuario también podrá eliminarlas.

Figura 11. Prototipo de la pantalla de “Ofertas”

- **Consultar los hoteles de una oferta**

Esta funcionalidad permite al usuario acceder al listado de hoteles que contiene una oferta y estará accesible a través de un icono junto a cada oferta.

Por defecto los hoteles se ordenarán por precio desde el más barato al más caro y desde esta funcionalidad el usuario podrá elegir visualizar el detalle de un hotel de la lista o ir a la página original de la oferta para reservarlo.

Por cada hotel se mostrará el nombre, número de estrellas, una imagen pequeña, el precio, el listado de los tres sitios webs con el precio más cercano y el enlace de su página original.

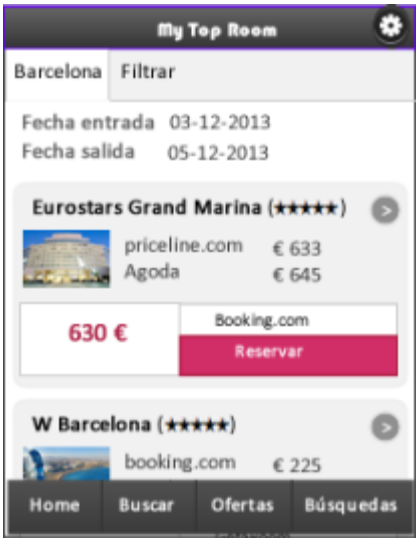


Figura 12. Prototipo de la pantalla "Hoteles de una oferta"

- **Filtrar los hoteles de una oferta**

Esta funcionalidad permitirá al usuario filtrar el listado de hoteles por otras opciones para facilitar la búsqueda del hotel deseado. Entre otras opciones se podrá buscar por tipo de alojamiento, instalaciones o el más recomendado.

Figura 13. Prototipo de la pantalla "Filtros de Hotel"

- **Consultar el detalle de un hotel**

Una vez el usuario consulta el listado de hoteles de una oferta concreta podrá acceder a esta funcionalidad a través del botón "Ver detalle" y el sistema mostrará toda la información del hotel como el resto de imágenes, detalles como la dirección web, teléfono y dirección.

Figura 14. Prototipo de la pantalla "Detalle de Hotel"

❖ **Reservar un hotel**

Esta funcionalidad permite redirigir al usuario a la página web original donde se encuentra la oferta del hotel para que pueda proceder a reservarlo. Para ello el usuario deberá consultar el listado de hoteles de una oferta y pulsar el correspondiente botón "Reservar" como puede verse en la Figura 12.

❖ **Eliminar una oferta**

El usuario deberá primero consultar la lista de ofertas y desde la misma pantalla podrá eliminar las que desee al pulsar el icono de la papelera correspondiente, como puede verse en la Figura 11. El sistema previa confirmación del usuario la borrará de la base de datos.

2.2 Requisitos no funcionales

Tal y como se ha mencionado, estos hacen referencia a las restricciones y cualidades sobre la solución de desarrollo del software pero dado que no es algo que suele expresar el cliente directamente como necesidades a veces resulta difícil recopilar estos tipos de requisitos.
Para ello resulta útil apoyarse en la técnica de obtención de requisitos por listas predefinidas y en este caso tanto el estándar IEEE-830 como la plantilla Volere ofrecen una lista predefinida de requisitos no funcionales.

En este proyecto se emplea la clasificación que propone la plantilla Volere y se incluye en cada sección los distintos requisitos de la aplicación:

1. *De presentación*: hacen referencia a la parte visual del sistema
   - En la cabecera de la aplicación se mostrará siempre el nombre "MyTopRoom"
   - El diseño de las pantallas debe incluir los colores corporativos
   - El diseño de la resolución de pantalla no podrá exceder de 200 x 250 píxeles

2. *De usabilidad y humanidad*: son los relacionados con un sistema usable
   - La aplicación no debe requerir formación o lectura de guía para su uso
   - El lenguaje debe ser claro

3. *De cumplimiento*: se trata de requisitos que indican la forma de cumplir las responsabilidades del sistema, incluyendo características como la velocidad, la fiabilidad, etc.
   - Al enviar las ofertas a los dispositivos móviles se almacenarán por completo en ellos, de modo que el usuario pueda consultarlas aunque no disponga de conexión a Internet.
   - La comunicación se realizará en línea y en lotes

4. *Operacionales y de entorno*: agrupan los requisitos relacionados con el entorno en el que se usará el software
   - La aplicación deberá estar disponible para dispositivos con SO Android

5. *De mantenimiento y de soporte*: se refiere a los requisitos relacionados con el mantenimiento correctivo y evolutivo de la aplicación
   - Las posibles actualizaciones de la aplicación no pueden eliminar los datos almacenados del usuario

6. *De seguridad*: aquellos sobre los requisitos de acceso, integridad, auditoría, etc.
   - La creación de cuentas y el acceso a la aplicación irán bajo una comunicación encriptada.

7. *Culturales y políticos*: aquellos que tienen en cuenta aspectos culturales o políticos
   - La aplicación detectará el idioma del usuario: si es español la aplicación se mostrará con ese idioma por defecto, si es catalán en catalán y cualquier otro idioma cargará la aplicación en inglés.
   - El mismo idioma de la aplicación se añadirá por defecto en la cuenta del usuario
   - La moneda por defecto será el Euro-€

8. *Legales*: son los que hay que satisfacer en virtud de la ley que se aplicará sobre la organización que sea responsable del sistema
   - El almacenamiento de los datos en el motor de disponibilidad y en el dispositivo móvil deben garantizar el cumplimiento de la LOPD.

2.3 Especificación de casos de uso

Como describe Cockburn (2001) un caso de uso captura un contrato entre un stakeholder del sistema sobre su comportamiento. El caso de uso describe dicho comportamiento del sistema bajo varias condiciones en respuesta a una petición de uno de los stakeholders llamado también “actor principal”

Los casos de uso se especifican fundamentalmente de forma textual, aunque también se pueden describir mediante diagramas de secuencia, diagramas de flujo o haciendo uso del diagrama de UML para casos de uso (Cockburn, 2001)

En nuestro caso utilizaremos el diagrama de casos de uso de UML, que se utiliza para mostrar visualmente cuáles son los actores que intervienen en el sistema y qué casos de uso pueden ejecutar. En cualquier caso, no sustituye a la descripción textual, ya que el diagrama no contiene información de la funcionalidad y esto es algo que debe quedar especificado en cada caso de uso.

2.3.1 *Actores*

Un actor es algo que comunica con el sistema o producto y que es externo al sistema en sí (Pressman, 2001) Por tanto los actores podrían ser tanto personas que interactúan con la aplicación como otros sistemas o aplicaciones que de alguna forma comunican con ella.

En nuestro caso se definen dos tipos de actores:

- **Usuario:** será toda persona que ejecuta la aplicación en un dispositivo móvil
- **Usuario registrado:** serán aquellas personas que se han registrado en el sistema creando una cuenta y además acceden a la aplicación mediante su correo electrónico y contraseña

En los siguientes apartados, al igual que con los requisitos funcionales, los casos de uso también se han agrupado en función del módulo que contendría la funcionalidad de dicho caso de uso. Por cada módulo se detalla:

- Diagrama UML de los casos de uso con los actores y los tipos de relaciones entre los casos de uso y éstos
- Descripción textual de cada caso de uso

2.3.2 *Casos de uso del módulo de conexión*

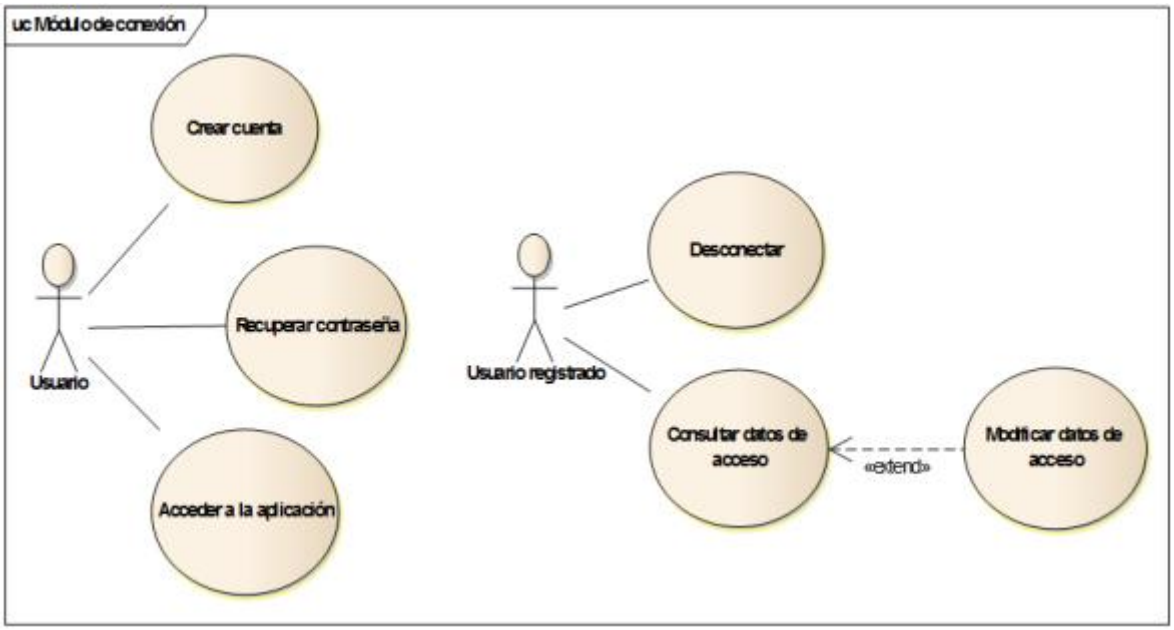


Figura 15. Casos de uso del módulo de conexión

Caso de uso 1	Crear una cuenta
Resumen de la funcionalidad	Permite al usuario crear una cuenta nueva
Casos de uso relacionados	
Actores	Usuario
Precondición	No puede existir una Cuenta con otro correo electrónico relacionada con el dispositivo móvil
Postcondición	Se creará la Cuenta y la aplicación redirigirá al usuario a la pantalla “Home” de la aplicación
Escenario de éxito	1. El sistema muestra la pantalla inicial 2. El usuario selecciona la opción “Cuenta nueva” 3. El sistema muestra la pantalla “Crear cuenta” con el formulario 4. El usuario introduce el nombre, correo electrónico, contraseña, la pregunta secreta, la respuesta secreta y pulsa “Crear” 5. El sistema crea la Cuenta en el motor de disponibilidad y en la base de datos del dispositivo y redirige al usuario a la página “Mi Perfil”

Extensiones	2a. El usuario elige salir de la aplicación 2a1. Se finaliza el caso de uso 4a. El actor introduce algún dato erróneo 4a1. El sistema muestra un mensaje de error 5a. El sistema no puede conectar con el motor de disponibilidad 5a1. El sistema muestra un mensaje indicando que no se puede crear la Cuenta en ese momento 6a. El sistema detecta que ya existe una Cuenta asociada al dispositivo 6a1. El sistema informa al usuario si desea asociar la nueva Cuenta al dispositivo y anular la anterior 6a2. El usuario confirma el mensaje 6a3. El sistema actualiza la Cuenta del dispositivo y redirige al usuario a la página "Mi Perfil"

Tabla 1. Caso de uso "Crear cuenta"

Caso de uso 2	**Recuperar la contraseña**
Resumen de la funcionalidad	Permite al usuario recuperar la contraseña en caso de extravío
Casos de uso relacionados	
Actores	Usuario
Precondición	El usuario debe haber creado una Cuenta
Postcondición	El sistema enviará al correo electrónico la nueva contraseña
Escenario de éxito	1. El sistema muestra la pantalla inicial 2. El usuario selecciona la opción "Recuperar contraseña" 3. El sistema muestra el formulario para recuperar la contraseña 4. El usuario introduce el correo electrónico, la pregunta secreta y la respuesta secreta y pulsa "Enviar" 5. El sistema comprueba que los datos son correctos con los guardados en la base de datos del dispositivo y redirige al usuario a la pantalla de los datos de acceso para que pueda modificarla por una conocida
Extensiones	3a. El usuario introduce una dirección de correo electrónico errónea 3a1. El sistema muestra un mensaje de error 4a. El sistema no puede acceder a la base de datos 4a1. El sistema muestra un mensaje indicando que no se puede recuperar la contraseña en ese momento

Tabla 2. Caso de uso "Recuperar la contraseña"

Caso de uso 3	Acceder a la aplicación
Resumen de la funcionalidad	Permite al usuario acceder a las funcionalidades privadas de la aplicación
Casos de uso relacionados	Crear una cuenta
Actores	Usuario registrado
Precondición	El usuario debe haber creado una cuenta y accedido con sus datos
Postcondición	Se mostrará la pantalla principal “Home” de la aplicación
Escenario de éxito	1. El sistema muestra la pantalla inicial 2. El usuario registrado introduce el correo electrónico, la contraseña y pulsa “Entrar” 3. El sistema comprueba que los datos son correctos, descarga las últimas ofertas y redirige al usuario a la página “Home” de la aplicación
Extensiones	2a. El usuario registrado introduce un correo electrónico no válido o no cumplimenta los campos 2a1. El sistema muestra un mensaje de error al usuario 3a. El sistema no puede conectar con el motor de disponibilidad y comprobar la identidad del usuario 3a1. El sistema muestra un mensaje indicando que no se puede acceder en ese momento 3b. El sistema detecta que el email no corresponde con la cuenta del dispositivo 3b1. El sistema muestra un mensaje indicando que el email no se corresponde con la cuenta del dispositivo

Tabla 3. Caso de uso “Acceder a la aplicación”

Caso de uso 4	Consultar datos de acceso
Resumen de la funcionalidad	Permite al usuario consultar los datos de acceso a la aplicación
Casos de uso relacionados	
Actores	Usuario registrado
Precondición	El usuario debe haber creado una cuenta y accedido con sus datos
Postcondición	Se muestra el formulario con los datos de acceso al usuario
Escenario de éxito	1. El usuario registrado selecciona “Datos de acceso” 2. El sistema muestra los datos de acceso
Extensiones	2a. El sistema no puede obtener la información de la base de datos 2a1. El sistema indica al usuario que no ha sido posible recuperar la información en ese momento

Tabla 4. Caso de uso “Consultar datos de acceso”

Caso de uso 5	Modificar datos de acceso
Resumen de la funcionalidad	Permite al usuario modificar los datos de acceso a la aplicación
Casos de uso relacionados	Consultar datos de acceso
Actores	Usuario registrado
Precondición	El usuario debe consultar los datos de acceso
Postcondición	Se actualizará la nueva información de acceso
Escenario de éxito	1. El usuario registrado modifica cualquiera de los datos de acceso (contraseña, pregunta secreta, respuesta secreta) y pulsa "Modificar" 2. El sistema actualiza los datos y redirige al usuario a la página "Home" de la aplicación
Extensiones	1a. El usuario registrado introduce algún dato incorrecto 1a1. El sistema muestra un mensaje de error al usuario 3a. El sistema no puede conectar con el motor de disponibilidad y actualizar los datos 3a1. El sistema muestra un mensaje indicando que no se puede modificar los datos en ese momento

Tabla 5. Caso de uso "Modificar datos de acceso"

Caso de uso 6	Desconectar de la aplicación
Resumen de la funcionalidad	Permite al usuario desconectar la sesión y cerrar la aplicación
Casos de uso relacionados	Acceder a la aplicación
Actores	Usuario registrado
Precondición	El usuario debe haber creado una cuenta y accedido con sus datos
Postcondición	Se cerrará la sesión y la aplicación y el usuario deberá introducir de nuevo las credenciales al abrir la aplicación
Escenario de éxito	1. El usuario pulsa "Salir" 2. El sistema cierra la sesión, actualiza el valor para recordar las credenciales y redirige al usuario a la pantalla de Login

Tabla 6. Caso de uso "Desconectar"

2.3.3 *Casos de uso del módulo de usuario*

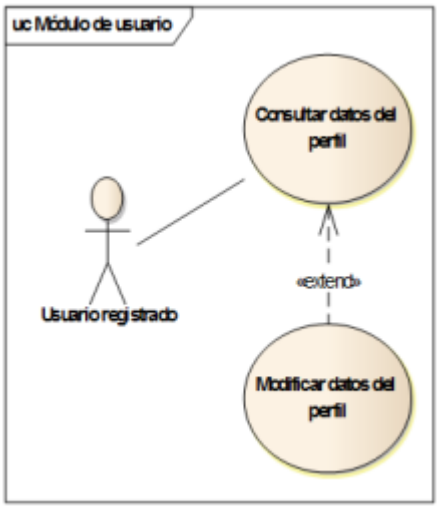


Figura 16. Casos de uso del módulo de usuario

Caso de uso 7	Consultar datos de perfil
Resumen de la funcionalidad	Permite al usuario consultar los datos de perfil a la aplicación
Casos de uso relacionados	
Actores	Usuario registrado
Precondición	El usuario debe haber creado una cuenta y accedido con sus datos
Postcondición	Se muestra el formulario con los datos de perfil al usuario
Escenario de éxito	1. El usuario registrado selecciona “Datos de perfil” 2. El sistema muestra los datos de perfil
Extensiones	2a. El sistema no puede obtener la información de la base de datos 2a1. El sistema indica al usuario que no ha sido posible recuperar la información en ese momento

Tabla 7. Caso de uso “Consultar datos de perfil”

Caso de uso 8	Modificar datos de perfil
Resumen de la funcionalidad	Permite al usuario modificar los datos de perfil a la aplicación
Casos de uso relacionados	Consultar datos de perfil
Actores	Usuario registrado
Precondición	El usuario debe consultar los datos de perfil
Postcondición	Se actualizará la nueva información de perfil

Escenario de éxito	1. El usuario registrado modifica la moneda y pulsa “Aplicar” 2. El sistema actualiza la información y redirige al usuario a la página “Home” de la aplicación
Extensiones	2a. El sistema no puede conectar con el motor de disponibilidad y actualizar los datos 2a1. El sistema muestra un mensaje indicando que no se puede modificar los datos en ese momento

Tabla 8. Caso de uso “Modificar datos de perfil”

2.3.4 *Casos de uso del módulo de búsquedas*

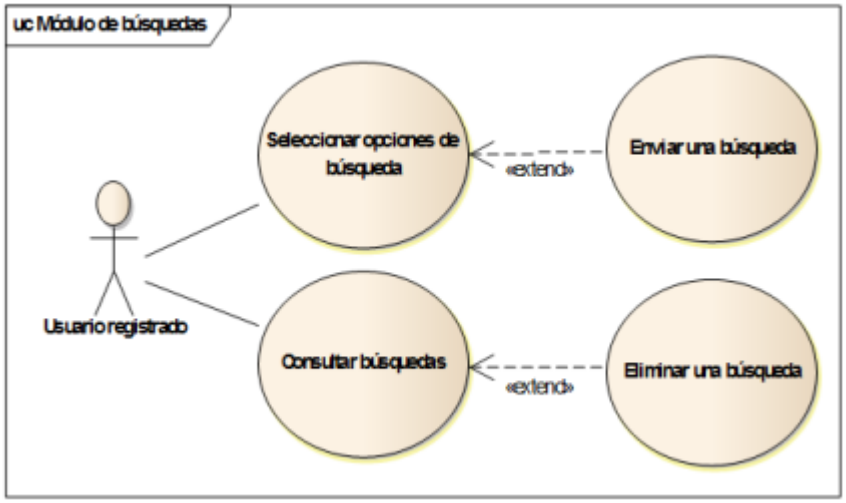


Figura 17. Casos de uso del módulo de búsquedas

Caso de uso 9	Seleccionar opciones de búsqueda
Resumen de la funcionalidad	Permite al usuario seleccionar distintas opciones de búsqueda
Casos de uso relacionados	
Actores	Usuario registrado
Precondición	El usuario debe haber creado una cuenta y accedido con sus datos
Postcondición	Se muestra la pantalla de búsqueda con las opciones seleccionadas
Escenario de éxito	1. El usuario registrado selecciona “Buscar” 2. El sistema muestra al usuario registrado el formulario con todas las opciones de búsqueda

Tabla 9. Caso de uso “Seleccionar opciones de búsqueda”

Caso de uso 10	Enviar una búsqueda
Resumen de la funcionalidad	Permite al usuario enviar una búsqueda al motor de disponibilidad
Casos de uso relacionados	Seleccionar opciones de búsqueda
Actores	Usuario registrado
Precondición	El usuario debe seleccionar las opciones de búsqueda
Postcondición	Se enviará la búsqueda al motor de disponibilidad y se actualiza en la base de datos del dispositivo
Escenario de éxito	1. El usuario registrado marca las opciones de búsqueda deseadas y pulsa "Enviar" 2. El sistema envía la búsqueda al motor disponibilidad y redirige de nuevo al usuario a la página de opciones de búsqueda
Extensiones	2a. El sistema no puede conectar con el motor de disponibilidad o la base de datos y enviar la búsqueda 2a1. El sistema muestra un mensaje indicando que no se puede enviar la búsqueda en ese momento 2b. Existe algún error en los datos del formulario 2b1. El sistema muestra el error 2c. El sistema detecta algún error en los datos de la búsqueda 2c1. El sistema muestra el error

Tabla 10. Caso de uso "Enviar una búsqueda"

Caso de uso 11	Consultar búsquedas
Resumen de la funcionalidad	Permite al usuario visualizar todas las búsquedas enviadas al motor de disponibilidad de las cuáles aún no se han encontrado ofertas
Casos de uso relacionados	
Actores	Usuario registrado
Precondición	El usuario elige consultar todas sus búsquedas
Postcondición	Se mostrará un listado de las búsquedas enviadas ordenadas por fecha
Escenario de éxito	1. El usuario registrado pulsa "Búsquedas" 2. El sistema muestra al usuario el listado de las búsquedas enviadas ordenadas por fecha desde la más reciente
Extensiones	2a. El sistema no puede obtener la información de la base de datos 2a1. El sistema indica al usuario que no ha sido posible recuperar la información en ese momento

Tabla 11. Caso de uso "Consultar búsquedas"

Caso de uso 12	Eliminar una búsqueda
Resumen de la funcionalidad	Permite al usuario modificar los datos de perfil a la aplicación
Casos de uso relacionados	Consultar búsquedas
Actores	Usuario registrado
Precondición	El usuario debe consultar la lista de búsquedas
Postcondición	Se eliminará la búsqueda seleccionada de la lista de búsquedas
Escenario de éxito	1. El usuario elige eliminar una búsqueda y pulsa el icono "Eliminar" de la búsqueda seleccionada 2. El sistema solicita la confirmación del usuario registrado 3. El usuario confirma la eliminación 4. El sistema elimina la búsqueda de la lista y vuelve a mostrar el listado de búsquedas
Extensiones	3a. El usuario registrado cancela la eliminación 3a1. El caso de uso finaliza 4a. El sistema no puede conectar con el motor de disponibilidad y eliminar la búsqueda 4a1. El sistema muestra un mensaje indicando que no se puede eliminar la búsqueda en ese momento

Tabla 12. Caso de uso "Eliminar una búsqueda"

2.3.5 *Casos de uso del módulo de ofertas*

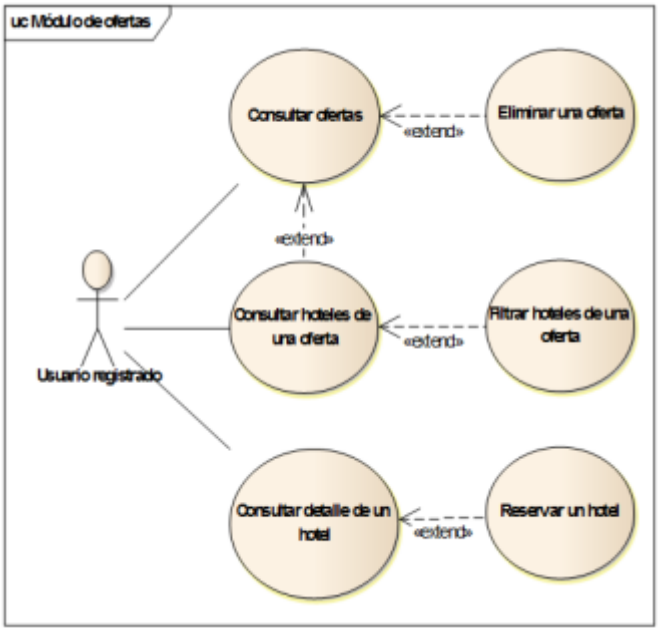


Figura 18. Casos de uso del módulo de ofertas

Caso de uso 13	Consultar ofertas
Resumen de la funcionalidad	Permite al usuario visualizar todas las ofertas recibidas de las distintas búsquedas enviadas al motor de disponibilidad
Casos de uso relacionados	
Actores	Usuario registrado
Precondición	El usuario elige consultar todas sus ofertas
Postcondición	Se mostrará un listado de las ofertas ordenadas por fecha
Escenario de éxito	1. El usuario registrado pulsa "Ofertas" 2. El sistema muestra al usuario el listado de las búsquedas enviadas ordenadas por fecha desde la más reciente
Extensiones	2a. El sistema no puede obtener la información de la base de datos 2a1. El sistema indica al usuario que no ha sido posible recuperar la información en ese momento

Tabla 13. Caso de uso "Consultar ofertas"

Caso de uso 14	Consultar los hoteles de una oferta
Resumen de la funcionalidad	Permite al usuario visualizar la lista de hoteles de una oferta concreta
Casos de uso relacionados	Consultar ofertas
Actores	Usuario registrado
Precondición	El usuario debe consultar la lista de ofertas
Postcondición	Se mostrará el listado de hoteles de la oferta
Escenario de éxito	1. El usuario selecciona una oferta y elige visualizar la lista de sus hoteles 2. El sistema muestra el listado de hoteles, cada uno de ellos con la opción de visualizar su detalle o reservar en la página oficial de la oferta
Extensiones	2a. El sistema no puede obtener la información de la base de datos 2a1. El sistema indica al usuario que no ha sido posible recuperar la lista de hoteles en ese momento

Tabla 14. Caso de uso "Consultar los hoteles de una oferta"

Caso de uso 15	Filtrar los hoteles de una oferta
Resumen de la funcionalidad	Con distintas opciones de búsqueda permite al usuario filtrar el listado de hoteles para obtener un subconjunto de ellos
Casos de uso relacionados	Consultar los hoteles de una oferta
Actores	Usuario registrado
Precondición	El usuario debe consultar el listado de hoteles de una oferta

Postcondición	Se mostrarán solo los hoteles del listado que cumplan las condiciones indicadas por el usuario
Escenario de éxito	1. El usuario elige filtrar el listado de hoteles 2. El sistema muestra las distintas opciones por las que puede filtrar (tipo de alojamiento, instalaciones y puntuación) 3. El usuario marca las opciones y pulsa "Usar filtros" 4. El sistema muestra solo los hoteles que cumplen las condiciones indicadas por el usuario registrado
Extensiones	4a. El sistema no puede obtener el listado de hoteles con los filtros 4a1. El sistema muestra un mensaje indicando que no se puede filtrar el listado de hoteles en ese momento

Tabla 15. Caso de uso "Filtrar los hoteles de una oferta"

Caso de uso 16	**Consultar el detalle de un hotel**
Resumen de la funcionalidad	Permite al usuario visualizar el detalle de un hotel concreto de la lista de hoteles de una oferta
Casos de uso relacionados	Consultar los hoteles de una oferta
Actores	Usuario registrado
Precondición	El usuario debe consultar el listado de hoteles de una oferta
Postcondición	Se mostrarán todas las características del hotel seleccionado
Escenario de éxito	1. El usuario elige ver el detalle de un hotel del listado 2. El sistema muestra todas las características y atributos del hotel seleccionado (fotografías, dirección y precios)
Extensiones	2a. El sistema no puede obtener la información del hotel de la base de datos 2a1. El sistema muestra un mensaje indicando que no se puede mostrar la información del hotel en ese momento

Tabla 16. Caso de uso "Consultar el detalle de un hotel"

Caso de uso 17	**Reservar un hotel**
Resumen de la funcionalidad	Permite al usuario ir a la dirección web con la oferta original a reservar el hotel
Casos de uso relacionados	Consultar los hoteles de una oferta
Actores	Usuario registrado
Precondición	El usuario debe consultar el listado de hoteles de una oferta
Postcondición	Se redirige al usuario a la página original dónde se encontró la oferta de hotel

Escenario de éxito	1. El usuario elige reservar un hotel del listado 2. El sistema actualiza el hotel como visto y redirige al usuario a la dirección web externa dónde se encontró la oferta para que el usuario pueda reservarlo
Extensiones	2a. El sistema no puede conectar con la base de datos para actualizar la información del hotel 2a1. El sistema muestra el mensaje de error

Tabla 17. Caso de uso "Reservar un hotel"

Caso de uso 18	**Eliminar una oferta**
Resumen de la funcionalidad	Permite al usuario modificar los datos de perfil a la aplicación
Casos de uso relacionados	Consultar ofertas
Actores	Usuario registrado
Precondición	El usuario debe consultar la lista de ofertas
Postcondición	Se eliminará la oferta seleccionada de la lista de ofertas
Escenario de éxito	1. El usuario elige eliminar una oferta y pulsa el icono "Eliminar" 2. El sistema solicita la confirmación del usuario registrado y elimina la oferta de la lista
Extensiones	2a. El usuario registrado cancela la eliminación 2a1. El caso de uso finaliza 3a. El sistema no puede conectar con la base de datos y eliminar la oferta 3a1. El sistema muestra un mensaje indicando que no se puede eliminar la oferta en ese momento

Tabla 18. Caso de uso "Eliminar una oferta"

3. Diseño

3.1 Introducción

“El diseño es la actividad que transforma la especificación de requisitos en una solución técnica viable…El resultado del diseño es una especificación de cómo construir la aplicación o sistema y de las restricciones en la implementación” (Albin, 2003)

Para llegar a esta especificación se empieza por el diseño de la arquitectura, que se enfoca en dividir el sistema en componentes que al interactuar entre ellos, satisfacen los requisitos funcionales y no funcionales, utilizando modelos.

“Un modelo es una representación, en cierto medio, de algo en el mismo u otro medio. El modelo, capta los aspectos importantes de lo que estamos modelando, desde ciertos puntos de vista, y simplifica u omita el resto.” (Booch, 2000)

Como indica Albin (2003) estos modelos pueden definirse mediante diagramas, descripciones textuales, fórmulas, etc. y después se agrupan en vistas del sistema, donde cada vista representa algún aspecto del sistema, como por ejemplo su comportamiento, información o rendimiento.

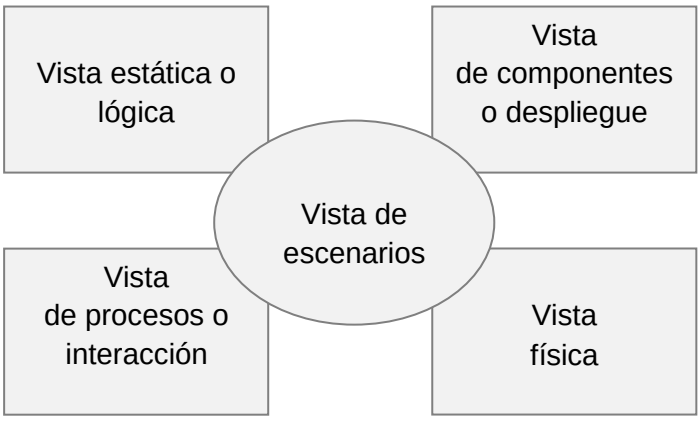
Con el objetivo de intentar establecer lo que debería incluir toda descripción de arquitectura software, se publicó en el año 2000 el estándar IEEE1471 “*IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*”, “que sienta las bases para la descripción de la arquitectura global de los sistemas software, utilizando una separación modular del diseño en diferentes perspectivas, denominadas puntos de vista” (Vallecillo, 2013)
Lo que comenta Albin (2003) es que existe una gran brecha entre los requisitos del cliente y los modelos que representan la implementación, lo que impide que muchos stakeholders puedan revisar o entender el diseño y al aplicar el IEEE1471, podemos reducir esta brecha semántica entre los requisitos y el diseño, y mejorar el proceso de la descripción de arquitectura en general.

“Actualmente hay varias propuestas para la especificación arquitectónica de los sistemas software basados en la norma IEEE 1471, como por ejemplo el modelo de "4+1" vistas de Philippe Kruchten (1995), el marco arquitectónico de John Zachman (1987), el marco arquitectónico TOGAF del Open Group y el modelo de referencia para el procesamiento abierto y distribuido RM-ODP de ISO/IEC e ITU-T (ISO/IEC 10746, 1998)” (Vallecillo, 2013) aunque ninguno de ellos conforma en su totalidad todas las recomendaciones de la norma.

En nuestro caso, se ha escogido el modelo de “4+1” vistas de Kruchten y como lenguaje de modelado se usará UML –*Unified Modeling Language-,* que surge en la segunda mitad de los noventa desde la empresa Rational por la necesidad de estandarizar los modelos y facilitar el entendimiento entre ingenieros de software.

Hoy en día, es un lenguaje aceptado a nivel mundial y mantenido por medio del consorcio sin ánimo de lucro OMG –Object Management Group- y que ayuda a especificar, visualizar y documentar modelos de un Software.

El modelo de Kruchten distingue cinco vistas:



- La vista lógica es el punto de vista para representar los requisitos funcionales y la actividad principal es la de identificar los objetos del modelo de dominio u objetos de negocio, es decir, debe especificar cómo los requisitos funcionales recogidos se mapean en clases y se realizan a través de sus inter-relaciones. Para ello definimos los diagramas de clases y objetos.

- La vista de procesos descompone dicha funcionalidad en procesos, dónde se define la colaboración, concurrencia y sincronización, y que representamos con los diagramas de secuencia entre las clases y objetos en la fase de diseño.

- La vista de desarrollo o componentes describe estáticamente cómo se organizan los programas software con respecto del entorno de desarrollo a través de la definición de componentes y paquetes, indicados con más detalle también en la fase de construcción.

- La vista física describe cómo se distribuye el software en los diferentes elementos hardware, tanto de procesamiento como de comunicación mediante los diagramas de despliegue, que pueden verse en el apartado de arquitectura del sistema (apartado 3.5)

- Por último, los escenarios vinculan todas las vistas anteriores juntas. Sirven para describir las funcionalidades principales del sistema mediante una serie de casos de uso encadenados a los que Kruchten denomina escenarios y se representan mediante diagramas de casos de uso. En este framework esta vista se considera redundante con respecto a las anteriores y de ahí el “+1” en el nombre.

Las cuatro primeras vistas se centran en los aspectos de desarrollo e implementación del sistema por lo que se definen en la etapa de diseño y construcción. La vista de escenarios en cambio, como sirve para ilustrar la funcionalidad principal del sistema se define en la etapa de análisis, tal y como está definida en el apartado de casos de uso anterior (apartado 2.3)

Como señala Vallecillo (2013) no todas las vistas de este modelo son necesarias en todos los sistemas software, y por tanto aquellas vistas que no sean relevantes para un sistema pueden ser omitidas de su descripción arquitectónica.

A continuación, para este proyecto se definen las vistas: lógica, de desarrollo, de proceso y física.

3.2 Vista lógica: diagramas de entidades, clases y fichas CRC

Una técnica muy usada para identificar los objetos del modelo de dominio en base a los requisitos, es a través de los análisis guiados por la responsabilidad con las tarjetas CRC -Class Responsibility Collaborator-
Un modelo CRC como indican Beck & Cunningham (1989) es una colección de tarjetas de índices estándar divididas en tres secciones: nombre de la clase, responsabilidades y colaboraciones. Cada clase representa una colección de objetos similares del dominio, las responsabilidades son algo que la clase conoce o realiza y que se indica a través de sus métodos y por último una colaboración es otra clase que interactúa con ella para llevar a cabo sus responsabilidades.

Las clases identificadas son las siguientes:

Usuario	
Descripción: Tipo: Características: Responsabilidades: Colaboraciones:	Representa a un usuario dentro de la aplicación Principal Concreta, persistente Mantiene los datos de un usuario Busqueda, Oferta, Hotel
Constructores: +Usuario() +Usuario(String email, String password) +Usuario(String name, String email, String password, Integer idPregunta, String respuesta, Timestamp fechaCreacion)	
Atributos: -idUsuario: Integer //Identificador único de usuario -nombre: String //Nombre del usuario -email: String //Dirección de correo electrónico del usuario -password: String //Contraseña de acceso -idPregunta: Integer //Hace referencia a la clave de la pregunta secreta escogida para recuperar la contraseña de acceso	

Usuario
-respuesta: String //Respuesta a la pregunta secreta a verificar en caso de pérdida de la contraseña -recordar: Boolean //Atributo de tipo "Si/No" que indica si la aplicación debe recordar las credenciales y no solicitarlas al usuario cada vez, o no -idMoneda: String //Valor clave de la moneda asociada al perfil del usuario -fechaCreacion: Timestamp //Fecha en la que se da de alta el usuario al crear la cuenta -fechaActualizacion: Timestamp //Fecha que se actualiza cada vez que se modifica alguno de los atributos del usuario
Métodos: //Métodos de acceso a los atributos privados -getIdUsuario(): Integer -setIdUsuario(Integer idUsuario) -getNombre(): String -setNombre(String nombre) -getEmail(): String -setEmail(String email) -getPassword(): String -setPassword(String password) -getIdPregunta(): Integer -setIdPregunta(Integer idPregunta) -getRespuesta(): String -setRespuesta(String respuesta) -getFechaCreacion(): Timestamp -setFechaCreacion(Timestamp fecha) -getFechaActualizacion(): Timestamp -setFechaActualizacion(Timestamp fecha) -getRecordar(): Boolean -setRecordar(Boolean recordar) -getIdMoneda(): String -setIdMoneda(String moneda)

Busqueda	
Descripción: Tipo: Características: Responsabilidades: Colaboraciones:	Representa una búsqueda realizada por un usuario Principal Concreta, persistente Almacena y proporciona los datos de una búsqueda realizada por el usuario Usuario
Constructores: +Busqueda() +Busqueda(Integer idUsuario, Integer idBusqueda, String codDestino, Integer estrellas, Date fechaInicio, Date fechaFin, Float precioMinimo, Float precioMaximo, String idMoneda, Date fechaEnvio)	
Atributos: -idUsuario: Integer //Identificador del usuario propietario de la búsqueda -idBusqueda: Integer //Valor clave que representa unívocamente la búsqueda	

Busqueda
-codDestino: String //Código de destino de la búsqueda -estrellas: Integer //Cantidad de estrellas que deben tener los hoteles de la búsqueda -fechaInicio: Date //Fecha aproximada de inicio de la estancia -fechaFin: Date //Fecha aproximada de fin de la estancia -precioMinimo: Float //Precio mínimo de la habitación de Hotel -precioMaximo: Float //Precio máximo de la habitación de Hotel -idMoneda: String //Clave de la moneda del usuario en la que se envía la búsqueda -fechaEnvio: Date //Fecha del envío de la búsqueda al motor de disponibilidad -encontrada: Boolean //Atributo de tipo “Si/No” que indica si se han recibido ofertas de la búsqueda o no
Métodos: //Métodos de acceso a los atributos privados -getIdUsuario(): Integer -setIdUsuario(Integer idUsuario) -getIdBusqueda(): Integer -setIdBusqueda(Integer idBusqueda) -getCodDestino(): String -setCodDestino(String codDestino) -getEstrellas(): Integer -setEstrellas(Integer estrellas) -getFechaInicio(): Date -setFechaInicio(Date fechaInicio) -getFechaFin(): Date -setFechaFin(Date fechaFin) -getPrecioMinimo(): Float -setPrecioMaximo(Float precioMaximo) -getIdMoneda(): String -setIdMoneda(String moneda) -getFechaEnvio(): Date -setFechaEnvio(Date fecha) -getEncontrada(): Boolean -setEncontrada(Boolean encontrada)

Oferta	
Descripción: Tipo: Características: Responsabilidades: Colaboraciones:	Representa una oferta recibida del motor de disponibilidad Principal Concreta, persistente Almacena y proporciona los datos de una oferta recibida Usuario, Busqueda
Constructores: +Oferta() +Oferta(Integer idUsuario, Integer idOferta, String codDestino, Integer estrellas, Date fechaInicio, Date fechaFin, Float precioMinimo, Float precioMaximo, String idMoneda, Integer idBusqueda, Date fechaBusqueda)	

Oferta
Atributos: -idUsuario: Integer //Identificador del usuario propietario de la oferta -idOferta: Integer //Valor clave que representa unívocamente la oferta -codDestino: String //Código de destino de la búsqueda -estrellas: Integer //Cantidad de estrellas que deben tener los hoteles de la búsqueda -fechaInicio: Date //Fecha aproximada de inicio de la estancia -fechaFin: Date //Fecha aproximada de fin de la estancia -precioMinimo: Float //Precio mínimo de la habitación de Hotel -precioMaximo: Float //Precio máximo de la habitación de Hotel -idMoneda: Integer //Clave de la moneda del usuario en la que se envía la búsqueda -idBusqueda: Integer //Referencia a la búsqueda a la que corresponde la oferta -fechaBusqueda: Date //Fecha en la que se envío la búsqueda -listaHoteles<Hotel> //Lista de Hoteles de los que consta la oferta
Métodos: //Métodos de acceso a los atributos privados -getIdUsuario(): Integer -setIdUsuario(Integer idUsuario) -getIdOferta(): Integer -setIdOferta(Integer idOferta) -getCodDestino(): String -setCodDestino(String codDestino) -getEstrellas(): Integer -setEstrellas(Integer estrellas) -getFechaInicio(): Date -setFechaInicio(Date fechaInicio) -getFechaFin(): Date -setFechaFin(Date fechaFin) -getPrecioMinimo(): Float -setPrecioMaximo(Float precioMaximo) -getIdMoneda(): String -setIdMoneda(String moneda) -getidBusqueda(): Integer -setIdBusqueda(Integer idBusqueda) -getFechaBusqueda(): Date -setFechaBusqueda(Date fecha) -getListaHoteles(): List<Hotel> -setListaHoteles(List<Hotel> lista)

Hotel	
Descripción:	Representa un Hotel incluido en una oferta
Tipo:	Principal
Características:	Concreta, persistente
Responsabilidades:	Almacena y proporciona los datos de un Hotel
Colaboraciones:	Oferta

Hotel
Constructores: +Hotel() +Hotel(Integer idUsuario, Integer idOferta, Integer idHotel)
Atributos: -idUsuario: Integer //Identificador del usuario propietario de la oferta -idOferta: Integer //Valor clave que representa unívocamente la oferta -idHotel: Integer //Valor clave que identifica unívocamente un Hotel -codDestino: String //Código de destino al que pertenece el Hotel -estrellas: Integer //Cantidad de estrellas del Hotel -nombreWeb: String //Nombre del sitio Web donde se encontró la oferta -urlWeb: String //Dirección URL que contiene la oferta del Hotel -consultado: Boolean //De tipo “Si/No” indica si el enlace para reservar el Hotel ha sido pulsado por el usuario al menos una vez
Métodos: //Métodos de acceso a los atributos privados -getIdUsuario(): Integer -setIdUsuario(Integer idUsuario) -getIdOferta(): Integer -setIdOferta(Integer idOferta) -getIdHotel(): Integer -setIdHotel(Integer idHotel) -getCodDestino(): String -setCodDestino(String codDestino) -getEstrellas(): Integer -setEstrellas(Integer estrellas) -getNombreWeb(): String -setNombreWeb(String nombreWeb) -getUrlWeb(): String -setUrlWeb(String urlWeb)

Estas clases representan el dominio de la aplicación y en el siguiente diagrama de entidades se muestra las relaciones entre ellas:

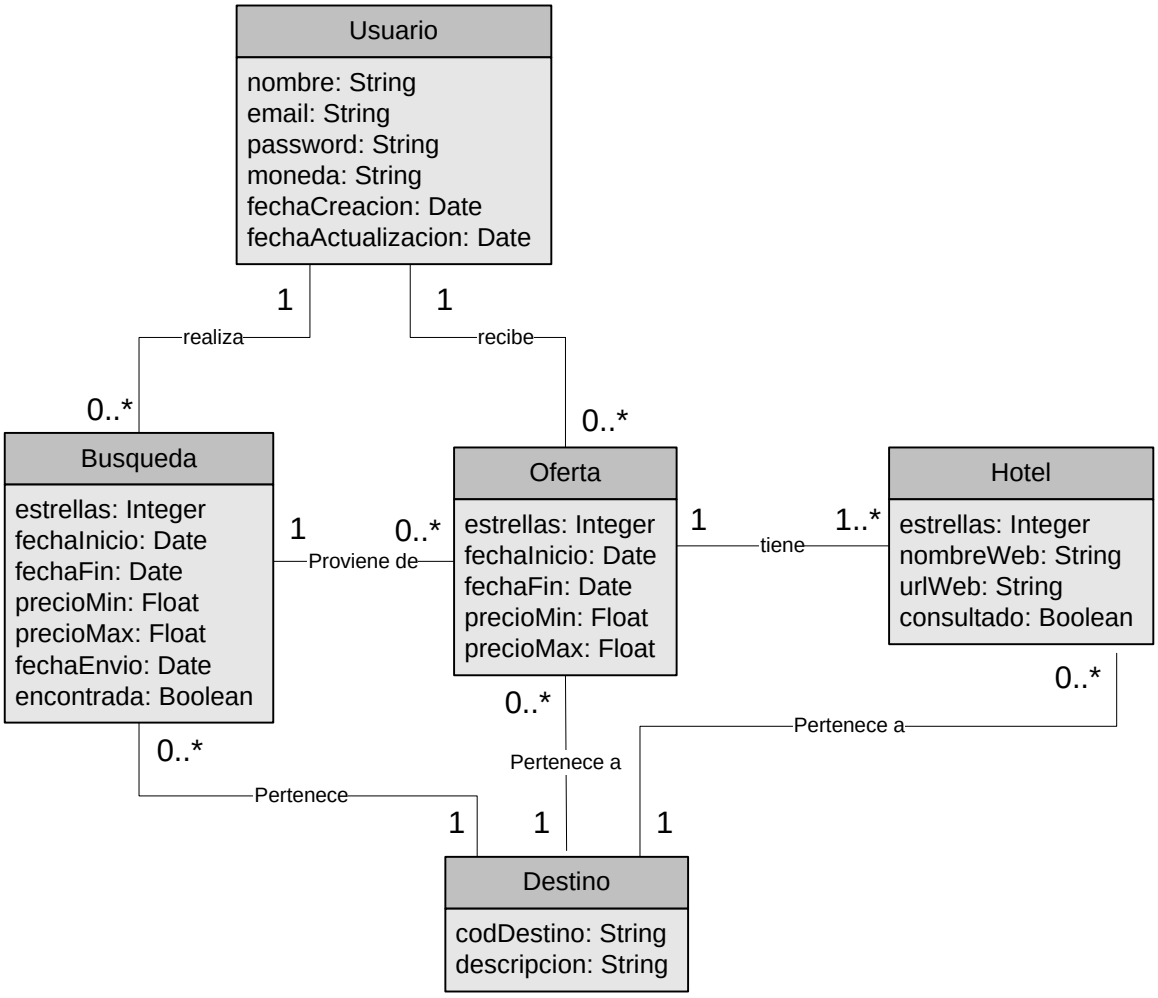


Figura 19. Diagrama de clases

Como se trata de clases persistentes, traducimos el mapa conceptual de clases al siguiente diagrama Entidad-Relación con estas decisiones de diseño:

- La asociación “Pertenece” de las clases “Busqueda”, “Oferta” y “Hotel” se representará como un atributo “codDestino” dentro de cada clase
- La asociación “Proviene de” entre Oferta y Búsqueda se indicará con un atributo dentro de la clase Oferta que identifique unívocamente a la búsqueda a la que pertenece

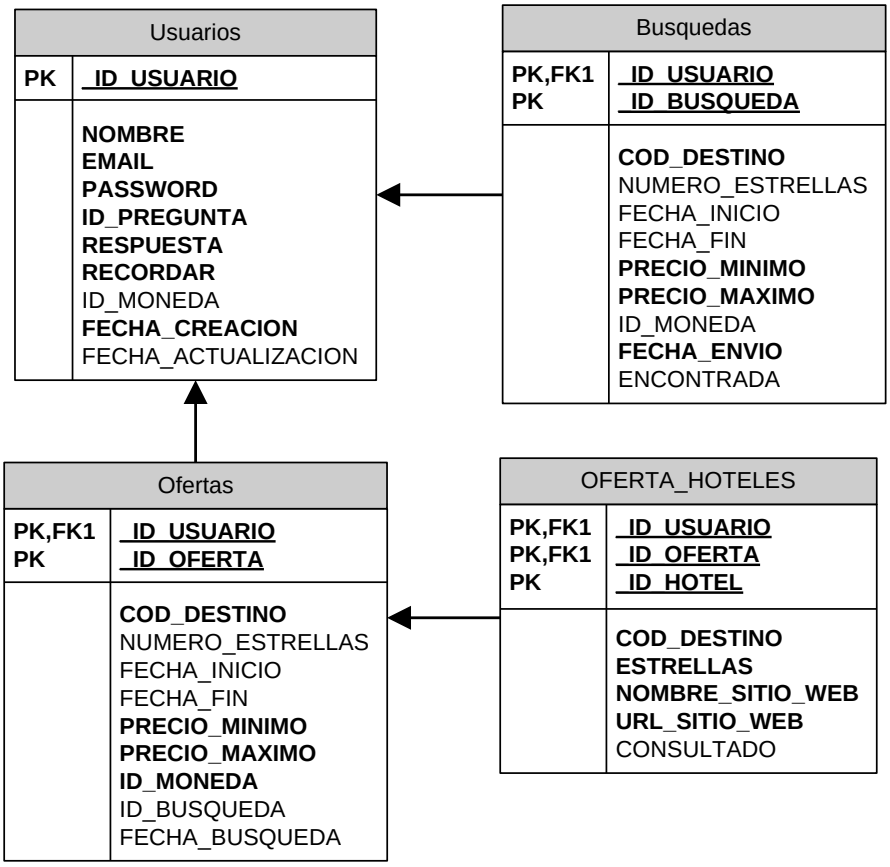


Figura 20. Diagrama Entidad-Relación

3.3 Vista de desarrollo o de componentes: arquitectura multi-capa

El siguiente paso en el diseño de la arquitectura lógica de la aplicación es la elección de la arquitectura software para definir la estructura de la aplicación.
Existen numerosos estilos arquitectónicos definidos para los sistemas software (sistemas de procesamiento por lotes, sistemas cliente-servidor, sistemas basados en eventos, sistemas orientados a servicios,…) y para este proyecto, por las características del desarrollo de aplicaciones Java, se ha escogido una arquitectura multi-capa donde se hace una división lógica de los componentes de la aplicación en las siguientes capas:

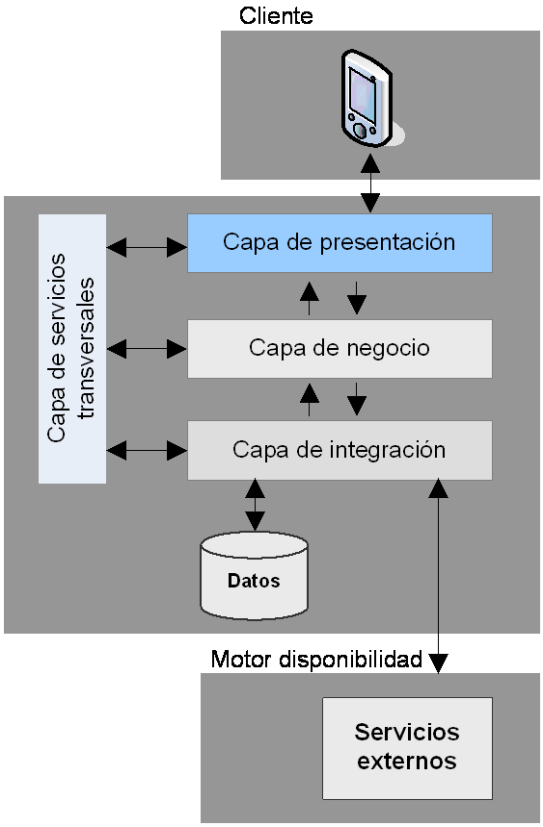


Figura 21. Arquitectura multi-capa de la aplicación móvil

"La buena distribución de una aplicación en capas incrementa la mantenibilidad, la robustez, la extensibilidad, la escalabilidad y la seguridad de la aplicación. (…) Un concepto clave en la distribución por capas de una aplicación es que los componentes de una capa solo pueden interactuar con componentes transversales, componentes de la misma capa o con componentes de las capas inferiores" (Vallecillo, 2013)

- En la capa de presentación se incluirán las clases que representan la interfaz gráfica de usuario. En Android cada pantalla está representada por una Actividad que es una clase de tipo *Activity* (apartado 4.2) o una subclase de ella, por tanto en base a los prototipos de la aplicación, tenemos las siguientes clases en la capa de presentación:
  - LoginActivity: pantalla inicial de la aplicación
  - CrearActivity: pantalla "Crear cuenta"
  - RecuperarActivity: pantalla "Recuperar contraseña"
  - PrivateActivity: pantalla genérica que muestra el menú inferior y superior de las pantallas privadas de la aplicación
  - HomeActivity: pantalla "Home"
  - AccesoActivity: pantalla para consultar y modificar los datos de acceso
  - PerfilActivity: pantalla para consultar y modificar los datos de perfil
  - BuscarActivity: pantalla para enviar una búsqueda
  - BusquedasActivity: pantalla de "Búsquedas"
  - OfertasActivity: pantalla de "Ofertas"

- ListaHotelesActivity: pantalla de “Hoteles de una oferta”
- FiltrarHotelesActivity: pantalla “Filtros de hotel”
- HotelActivity: pantalla “Detalle de un hotel”

- En la capa de negocio las clases que contienen la lógica de negocio (ServicioConexionBL, ServicioUsuarioBL, ServicioOfertasBL, ServicioBusquedasBL)
- En la capa de integración se incluirá una clase MyTopRoomBD que se encarga del acceso a datos en la base de datos integrada del dispositivo móvil y la clase HttpClientManager que se encargará de realizar las peticiones al motor de disponibilidad remoto.

3.4 Vista de procesos

“Los objetos obran recíprocamente para implementar un comportamiento” (Booch, 2000)
Mediante un diagrama de secuencia se puede representar la interacción entre objetos como un gráfico bidimensional, donde la dimensión vertical es el eje de tiempo que avanza hacia abajo y la dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración (Booch, 2000)

A continuación se definen los diagramas de secuencia asociados a cada caso de uso, para mostrar la interacción entre los objetos principales de nuestra aplicación.

3.4.1 *Módulo de conexión*

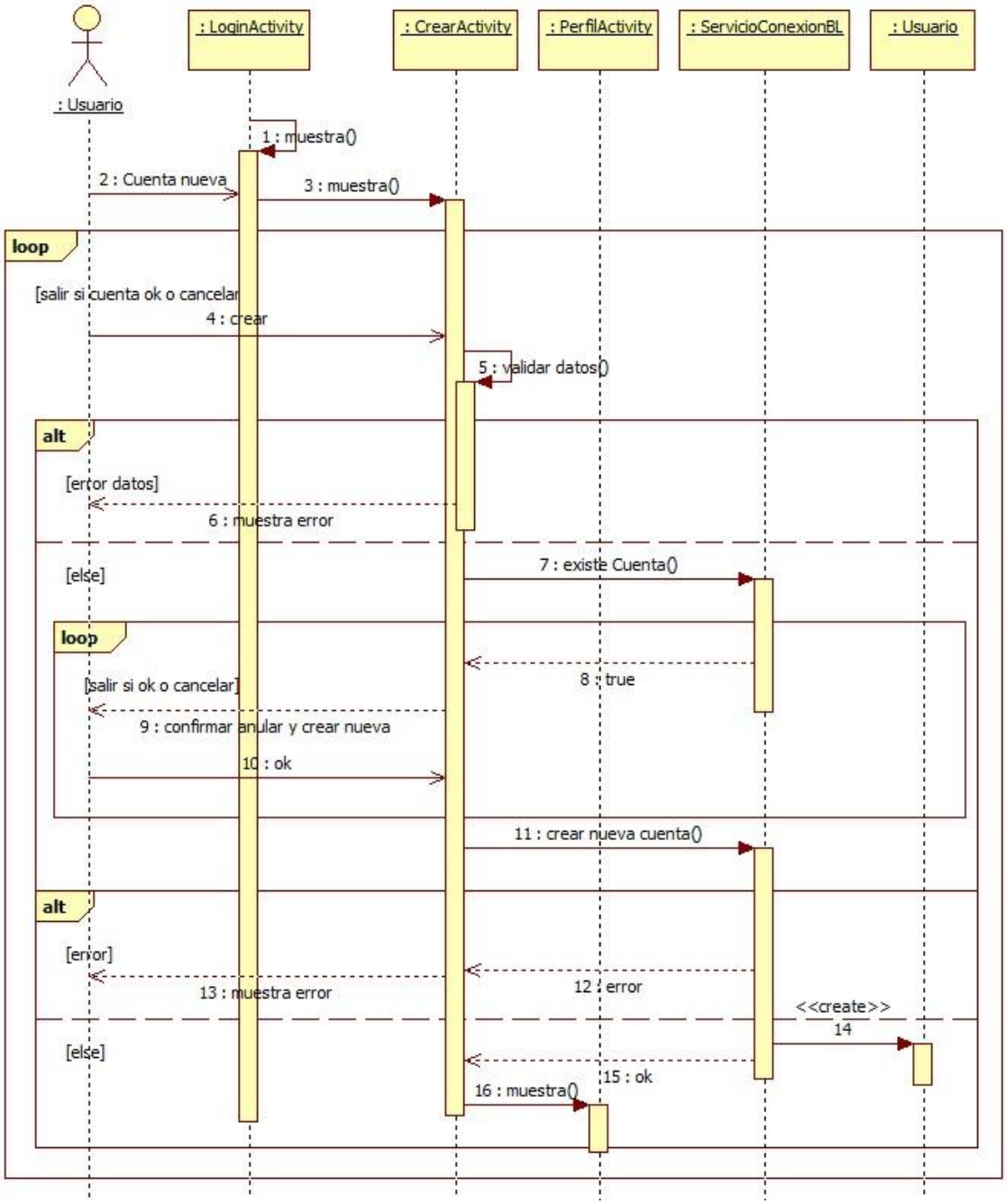


Figura 22. Diagrama de secuencia "Crear una cuenta"

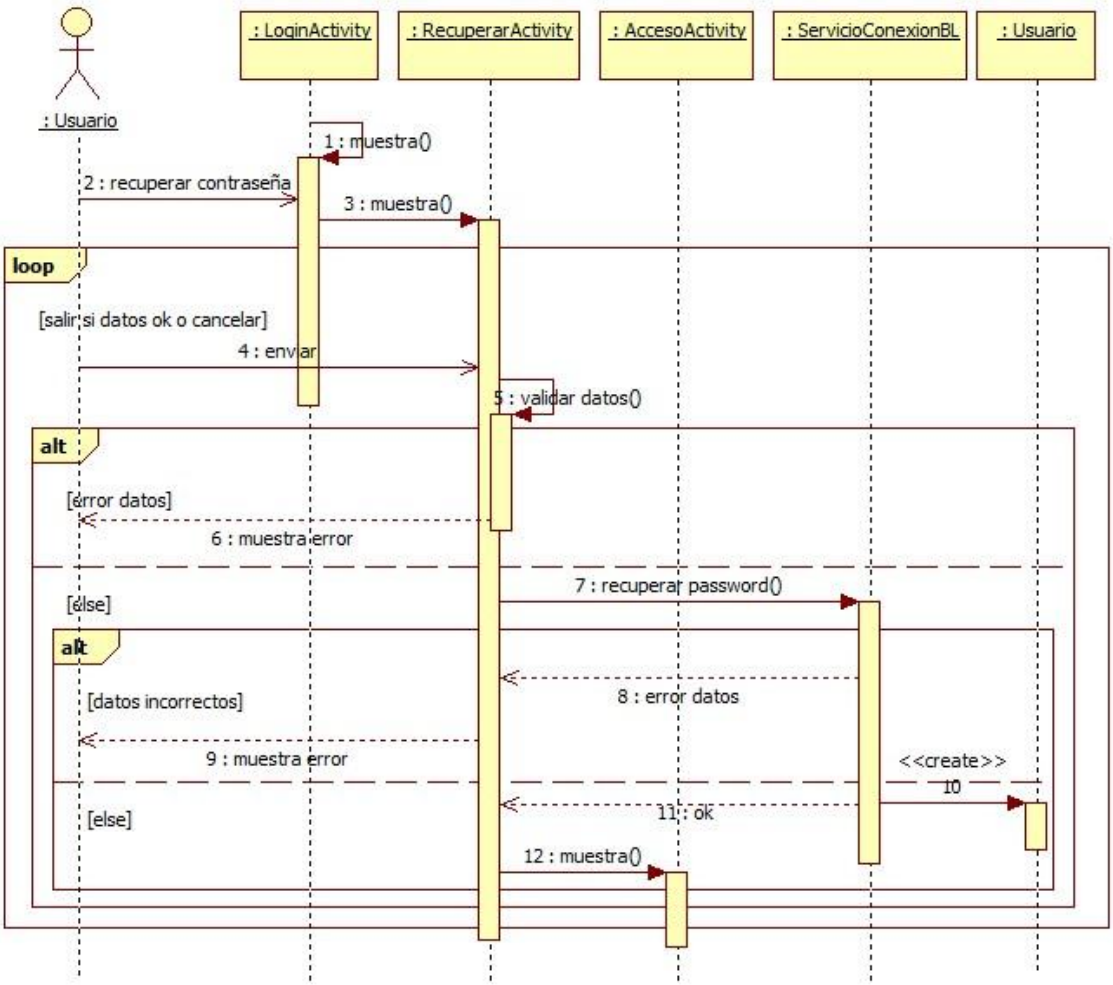


Figura 23. Diagrama de secuencia "Recuperar contraseña"

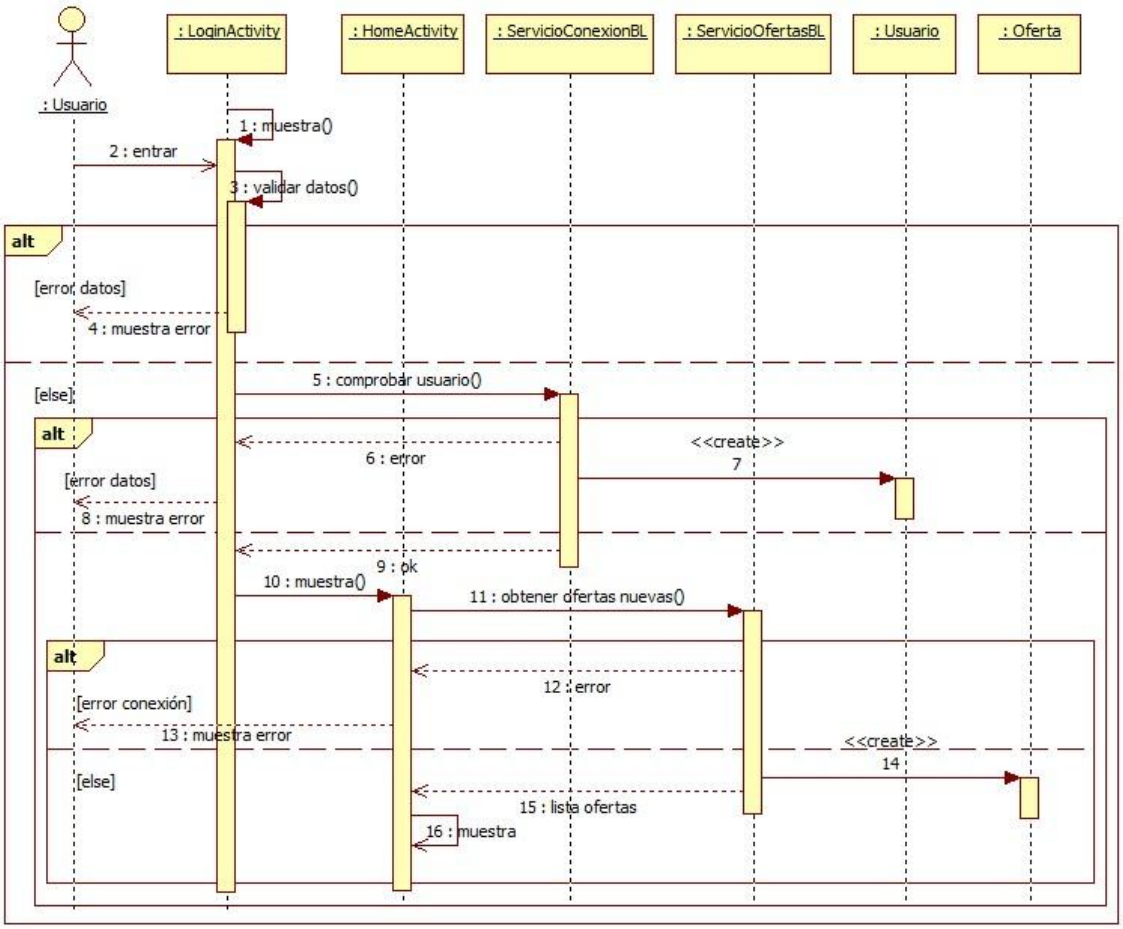


Figura 24. Diagrama de secuencia "Acceder a la aplicación"

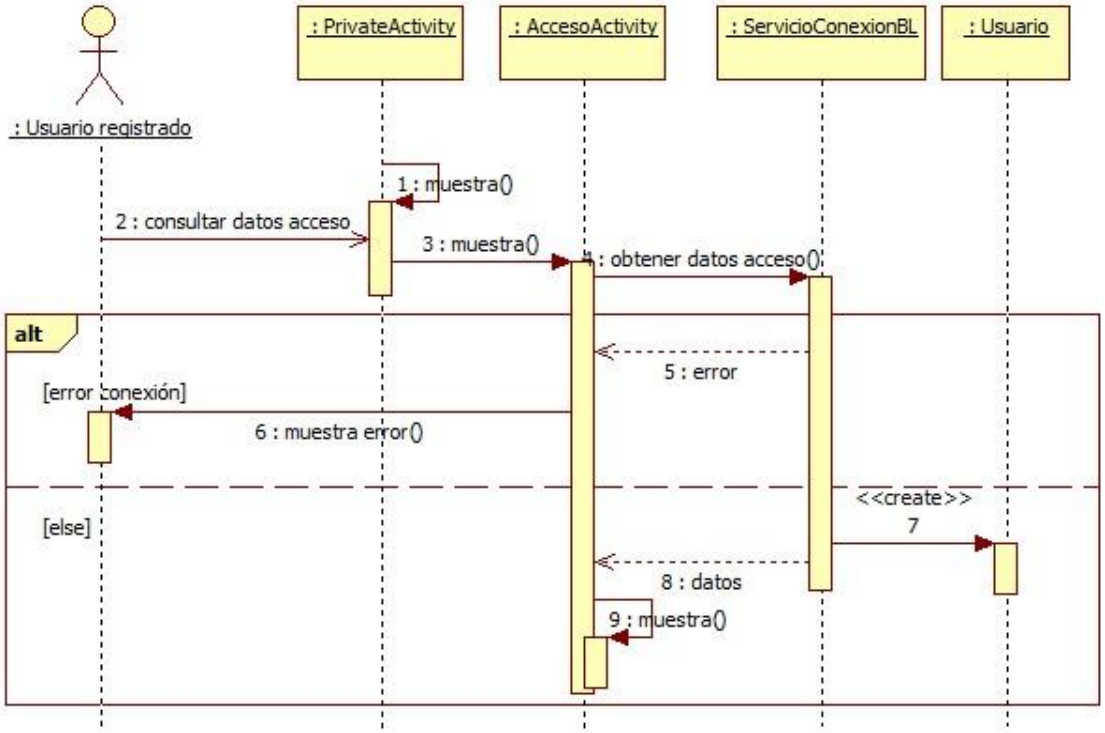


Figura 25. Diagrama de secuencia "Consultar datos de acceso"

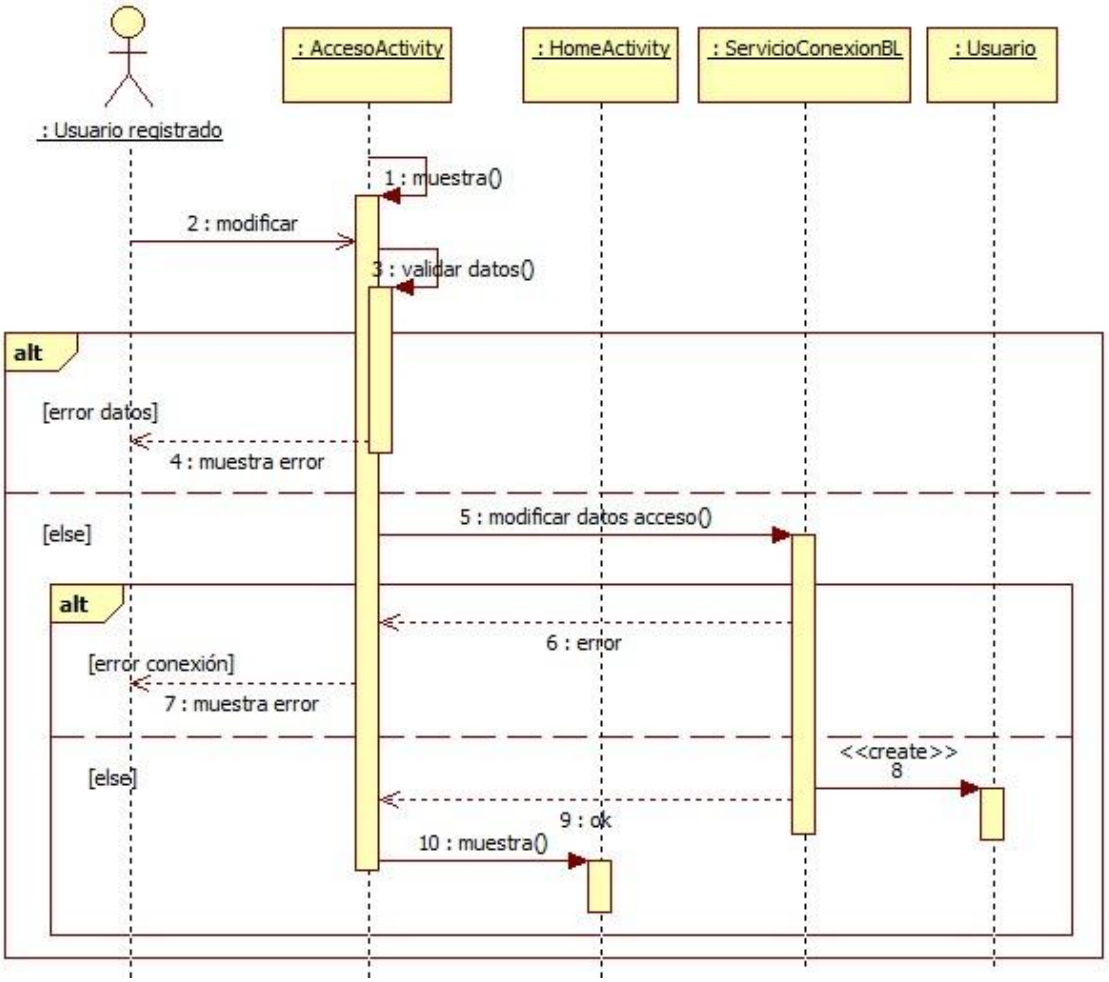


Figura 26. Diagrama de secuencia "Modificar datos de acceso"

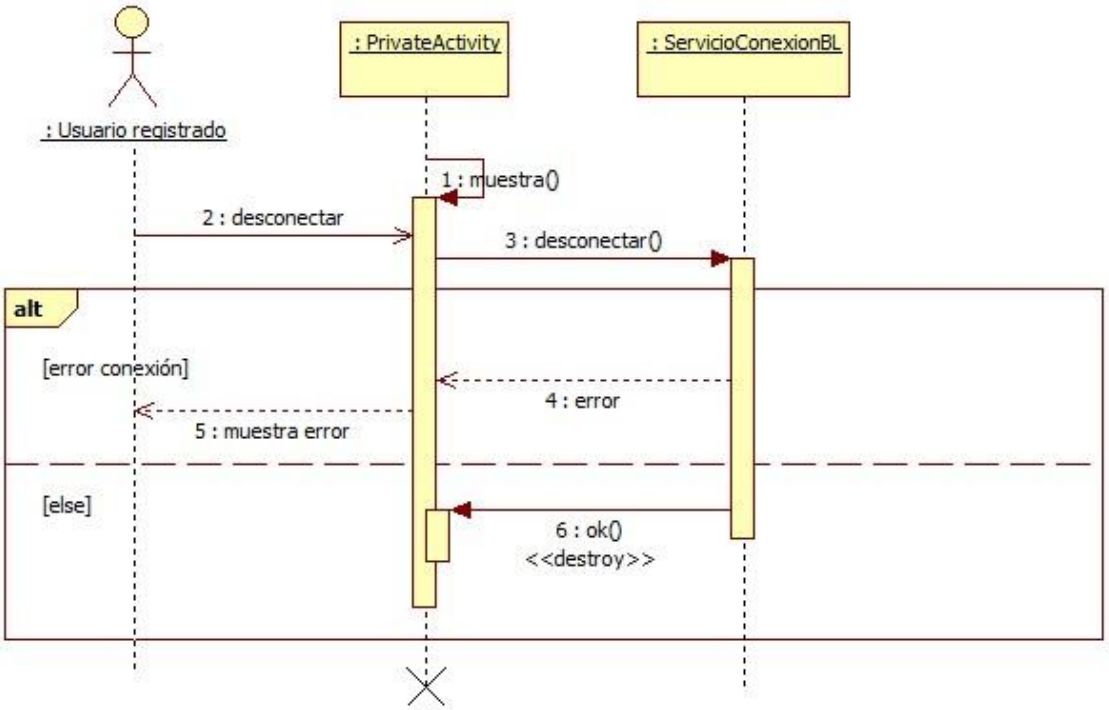


Figura 27. Diagrama de secuencia "Desconectar"

3.4.2 *Módulo de usuario*

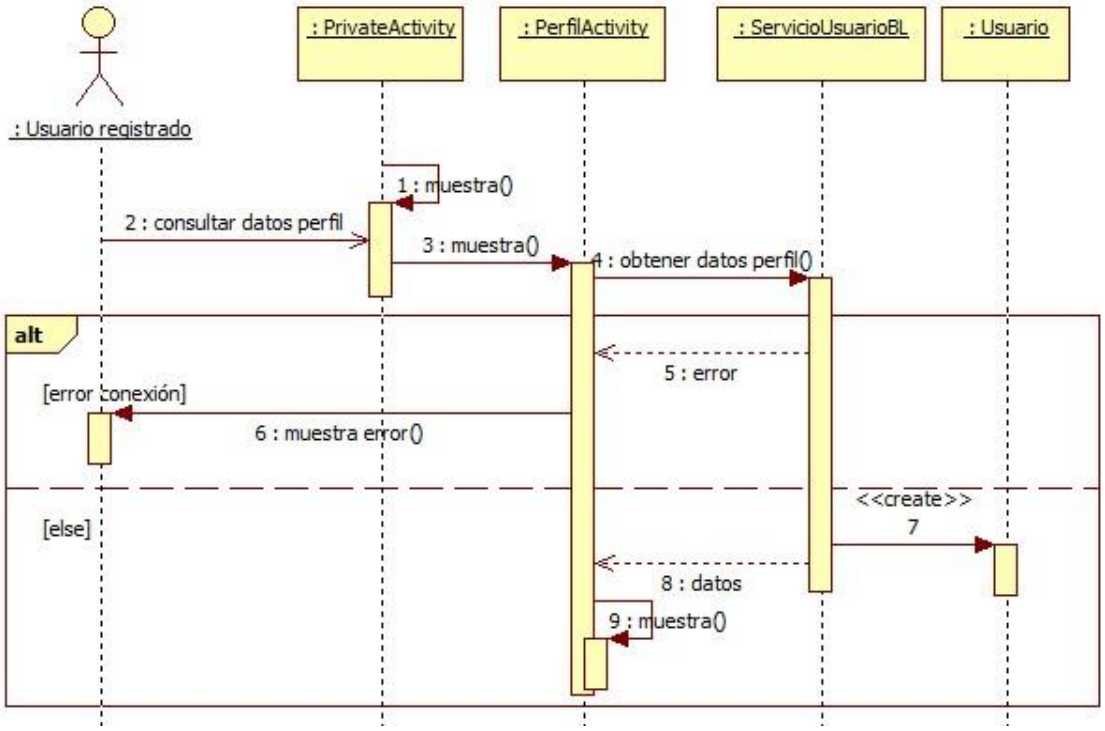


Figura 28. Diagrama de secuencia "Consultar datos de perfil"

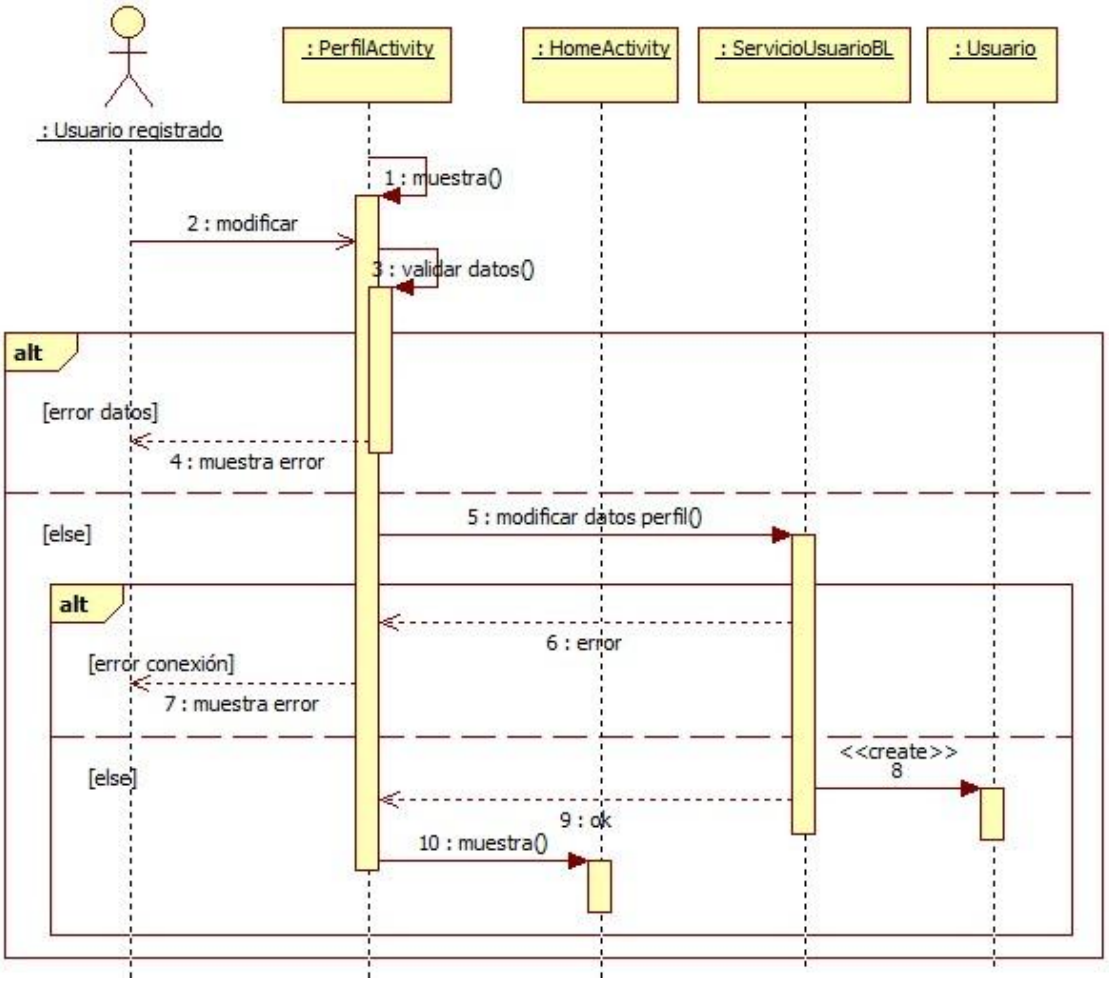


Figura 29. Diagrama de secuencia "Modificar datos de perfil"

3.4.3 *Módulo de búsquedas*

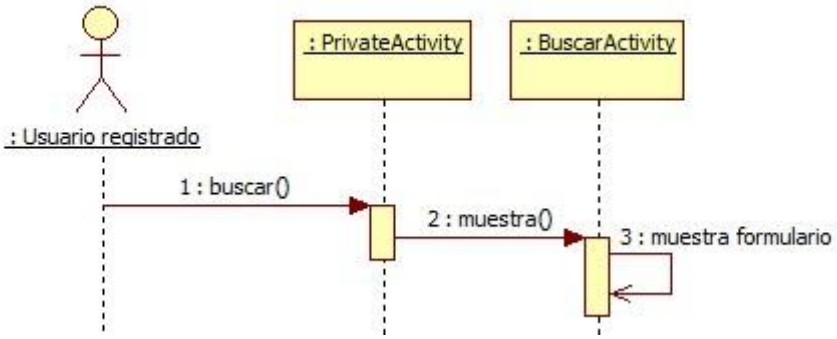


Figura 30. Diagrama de secuencia "Seleccionar opciones de búsqueda"

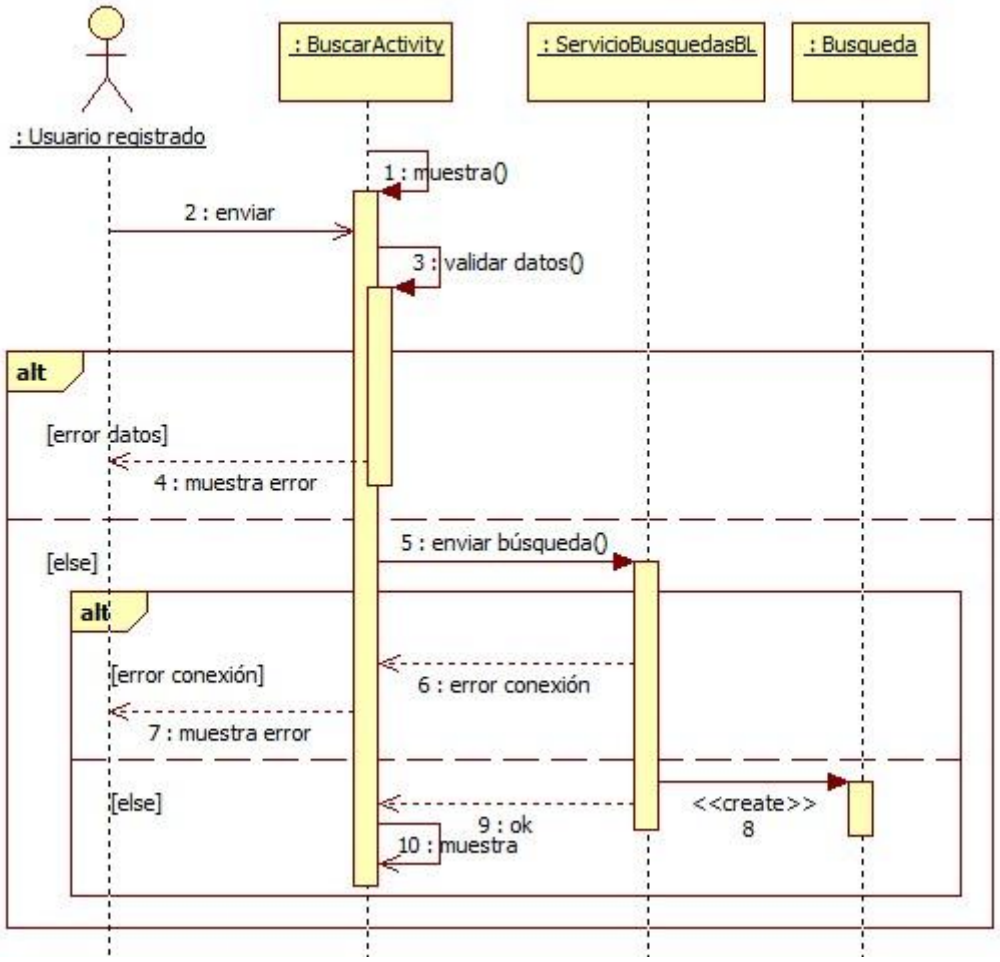


Figura 31. Diagrama de secuencia "Enviar una búsqueda"

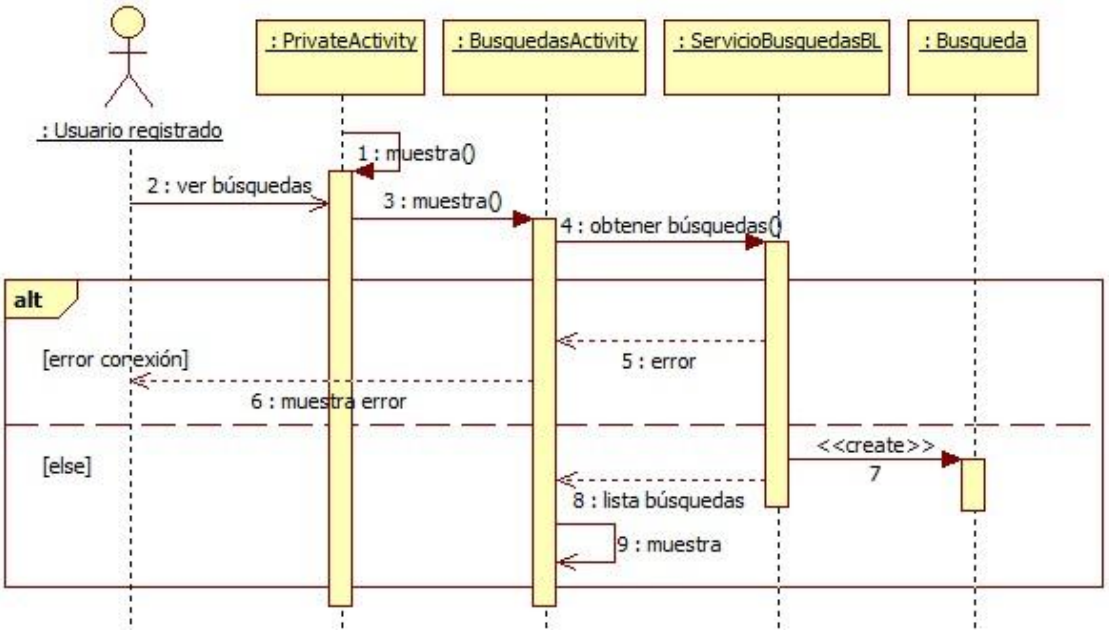


Figura 32. Diagrama de secuencia "Consultar búsquedas"

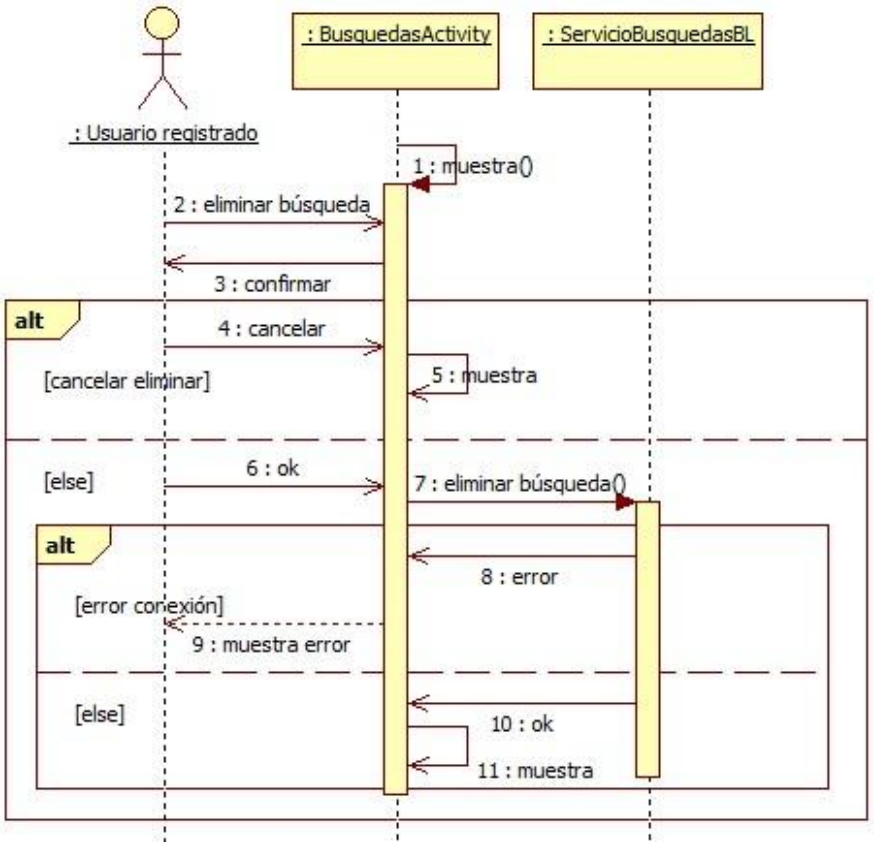


Figura 33. Diagrama de secuencia "Eliminar una búsqueda"

3.4.4 *Módulo de ofertas*

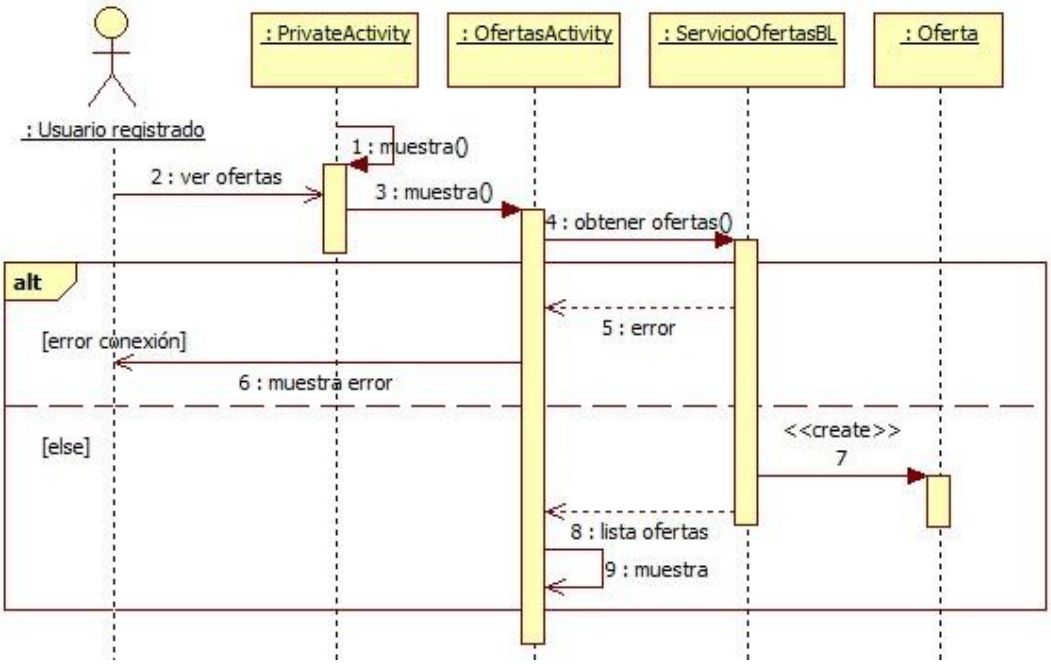


Figura 34. Diagrama de secuencia "Consultar ofertas"

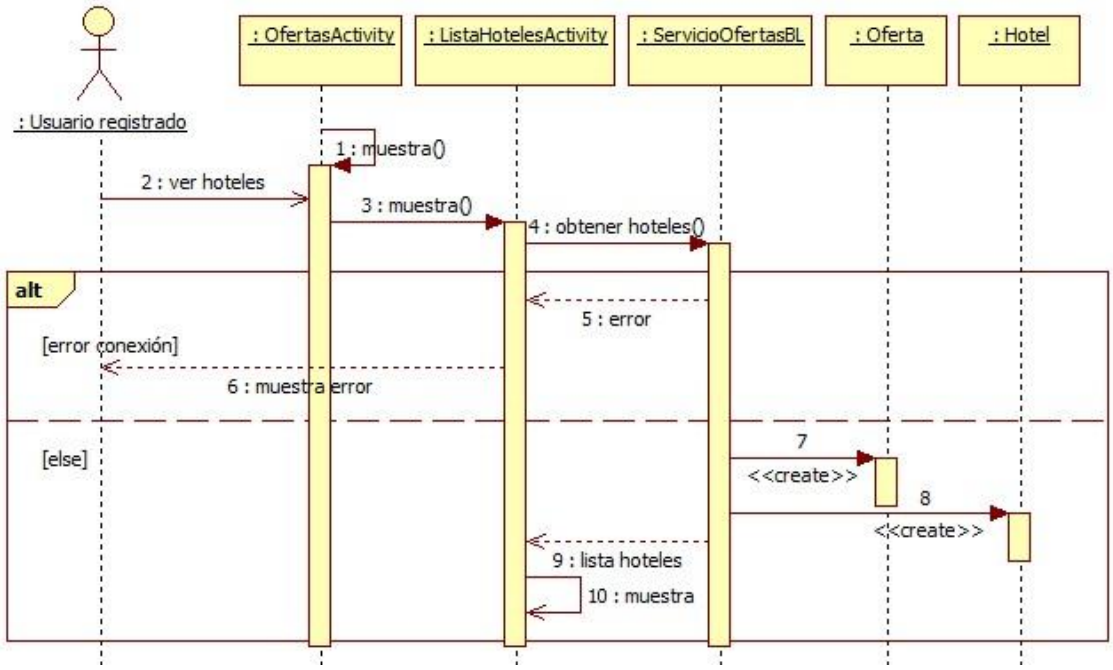


Figura 35. Diagrama de secuencia "Consultar hoteles de una oferta"

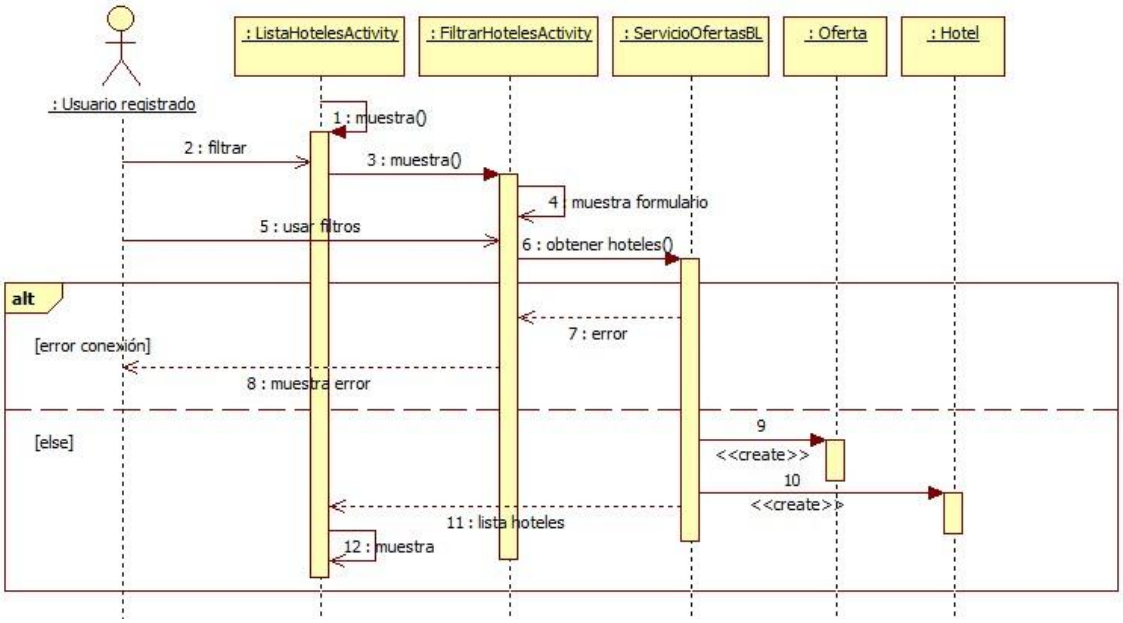


Figura 36. Diagrama de secuencia "Filtrar los hoteles de una oferta"

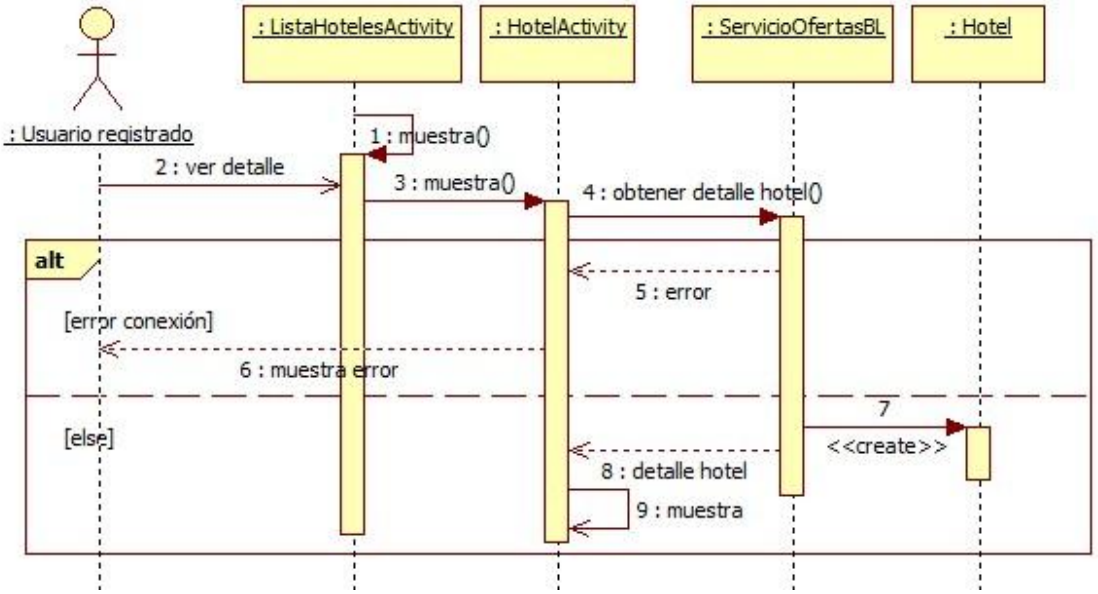


Figura 37. Diagrama de secuencia "Consultar el detalle de un hotel"

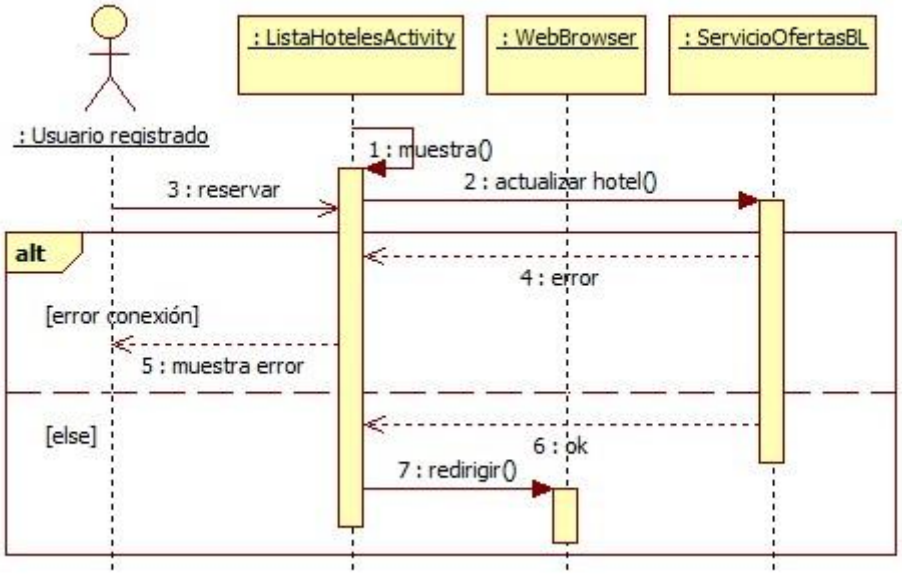


Figura 38. Diagrama de secuencia "Reservar un hotel"

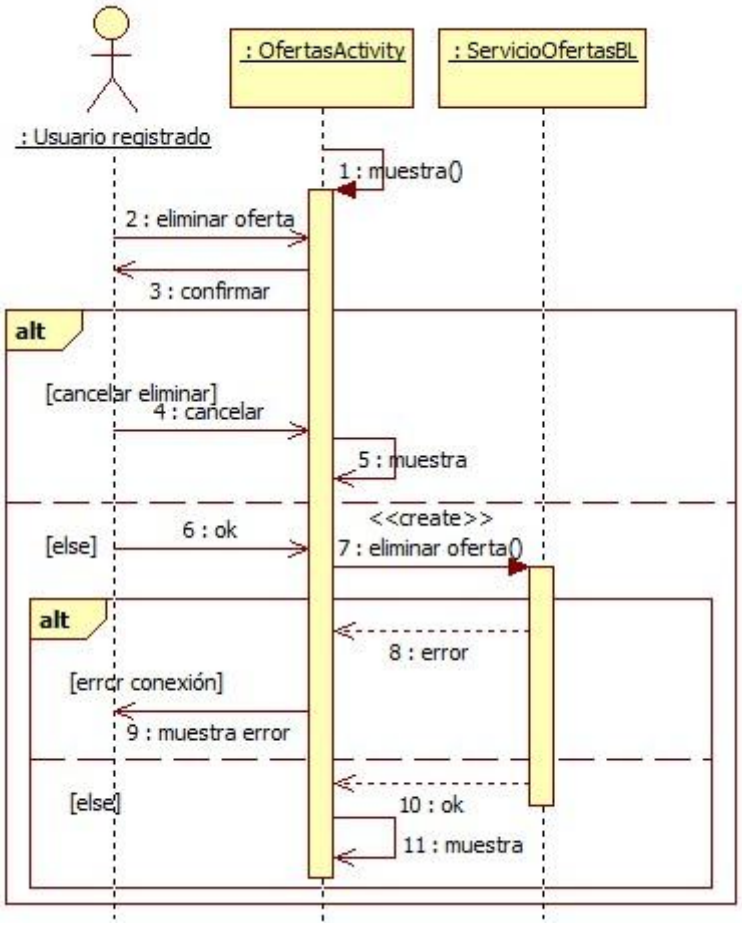


Figura 39. Diagrama de secuencia "Eliminar una oferta"

3.5 Vista física: arquitectura del sistema

Este apartado incluye de entrada la definición global de la arquitectura del sistema desde el punto de vista de los componentes de alto nivel, después en la fase de construcción se define la arquitectura multicapas de la que está formada la plataforma Android y por último se describen los componentes principales que forman una aplicación para Android como MyTopRoom.

La arquitectura del sistema es una arquitectura cliente-servidor, donde la aplicación se divide en un componente servidor, encargado de ofrecer los servicios y un conjunto de clientes que usan dichos servicios.

El componente cliente viene representado por la aplicación Android que se instala en cada dispositivo y el componente servidor es donde se instala el motor de disponibilidad y búsqueda de ofertas, que proporciona los servicios de autenticación, obtención de ofertas y petición de búsqueda de ofertas a las aplicaciones cliente.

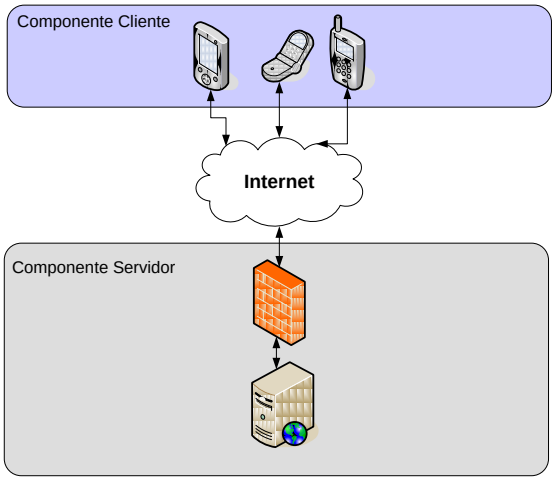


Figura 40. Arquitectura cliente-servidor

Una de las ventajas de utilizar este modelo es que se permite la distribución de sus componentes de forma natural y son más fáciles de escalar, lo que permite que se puedan incorporar nuevos servidores, por ejemplo si el número de usuarios que instalan la aplicación creciera, de forma transparente a dicha aplicación.

Los servicios que ofrecen los servidores se llaman servicios web y pueden ser de dos tipos:

- Servicios web basados en SOAP, es un software identificado por una URI, cuyas interfaces públicas y enlaces se definen y describen en ficheros XML. Se pueden implementar en varios mensajes y siguen las especificaciones SOAP, WSDL y UDDI
- Servicios web basados en REST, es un software diseñado para funcionar de forma más eficiente sobre la web. REST se basa en la arquitectura orientada a recursos (ROA) que está basada en acciones sobre dichos recursos para permitir la interacción entre el cliente y el servidor, mediante métodos *get, post, put y delete* ya incluidos en las llamadas HTTP. No requieren que los mensajes sea XML, lo que hace que sean más eficientes que los basados en SOAP, sobre todo para las aplicaciones móviles donde las respuestas al usuario deben ser siempre lo más rápidas posibles.

Hoy en día muchas de las aplicaciones que ofrecen servicios que se utilizan en webs o aplicaciones móviles, se implementan basados en REST y para el formato de los datos se utilizan siempre formatos ligeros como JSON (comentado en el apartado 2.2) Por ejemplo, aplicaciones de red social como facebook o twitter, disponen de APIs REST que se pueden utilizar para acceder a sus servicios desde una aplicación móvil.

Para este proyecto, por simplificación se crearán páginas “PHP” que estarán disponibles en un servidor Web para atender las peticiones del dispositivo móvil y cuyas respuestas serán enviadas en formato JSON.

4. Construcción

4.1 Arquitectura multicapa de Android

Antes de comenzar a desarrollar en Android, es importante conocer la arquitectura detrás de la plataforma y los conceptos claves que ayudarán después en la implementación de aplicaciones para este sistema operativo de dispositivos móviles.

La siguiente figura muestra el diagrama de la arquitectura Android con las capas que la forman:

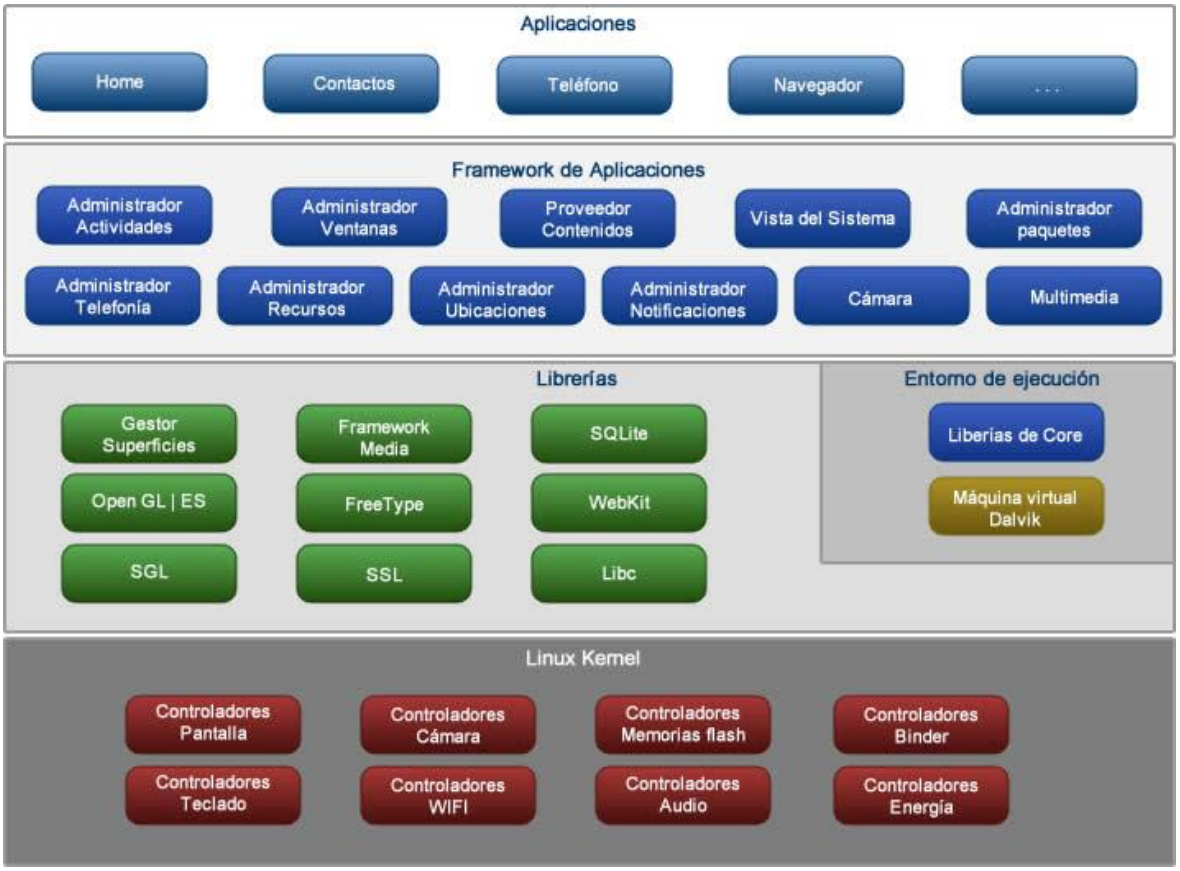


Figura 41. Arquitectura de Android

Cada una de las capas está formada a su vez por varios componentes y ofrece diferentes servicios a la capa superior.

- *Linux Kernel,* es la capa más básica y representa el núcleo del SO Android. Se trata de un Kernel de Linux 2.6, similar al que está en cualquier distribución de Linux, pero algunos cambios añadidos por Google para adaptarse al hardware que se ejecuta en Android.
  Actúa como capa de abstracción para los elementos de hardware a los que acceden las aplicaciones, así estas no tienen que preocuparse del modelo o características del hardware concreto del dispositivo y serán los drivers y controladores los que se encargan de eso.

Por último, Android también utiliza el kernel de Linux para las funciones principales de bajo nivel como la gestión de la memoria, gestión de los procesos, conexiones, etc.

- *Librerías,* esta capa está formada por las librerías nativas de Android. Están escritas en C o C++ y se compilan para la arquitectura hardware específica de cada dispositivo y suele ser una tarea que realiza el fabricante junto con su instalación antes de ponerlo a la venta. Su objetivo es proporcionar funcionalidades a las aplicaciones de aquellas tareas que se repiten con frecuencia. Para ver algunos ejemplos a continuación se detallan algunas de estas librerías:
    - *Gestor de superficies* (*Surface Manager),* se encarga de componer las imágenes que se muestran en la pantalla. Cada vez que una aplicación intenta *dibujar* en la pantalla, la librería no pasa directamente el mapa de bits a la pantalla, en su lugar, realiza los cambios en los mapas de bits que guarda en memoria y después los combina para formar la imagen final que envía a la pantalla. Esto hace que se puedan realizar con facilidad efectos como las transparencias, animaciones, transiciones, etc.
    - *Open GL | ES (OpenGL for Embedded Systems),* es el motor gráfico 3D basados en las APIs de OpenGL ES 1.0, 1.1. y 2.0.
    - *SGL (Scalable Graphics Library),* librería también utilizada en Chrome, es el motor gráfico de 2D de Android y se encarga de representar elementos en dos dimensiones
    - *Framework Media,* estas librerías permiten visualizar, reproducir e incluso grabar numerosos formatos de imagen, vídeo y audio (JPG, GIF, PNG, MPEG4, AVC, MP3, AAC O AMR)
    - *FreeType,* es la librería que permite mostrar las fuentes tipográficas, tanto las basadas en vectores como las que se basan en mapas de bits.
    - *SSL,* librería que proporciona seguridad al acceder a Internet mediante encriptado
    - SQLite, es un motor de base de datos relacionales, disponible para todas las aplicaciones.
    - *WebKit,* es un motor web que utiliza el navegador para renderizar páginas web
    - *Libc,* son librerías C basadas en la implementación de Berkeley Software Distribution, pero que han sido optimizadas para sistemas Linux embebidos. Estas librerías proporcionan una funcionalidad básica para la ejecución de las aplicaciones.

- *Entorno de ejecución Android,* está formado también por librerías pero no se considera una capa en sí mismo, sino una agrupación de ellas que ofrecen el entorno de ejecución de las aplicaciones.
  Está formado por:

- *Máquina virtual Dalvik*, el componente principal que ejecuta todas las aplicaciones no nativas de Android. Las aplicaciones se codifican casi siempre en Java y se compilan en un formato específico para esta máquina virtual, que es la que las ejecuta. Esto permite compilar una sola vez la aplicación y poder distribuirla compilada con la garantía de que podrá ejecutarse en cualquier dispositivo Android que tenga la versión mínima del SO que requiera la aplicación.

  Aunque escribamos las aplicaciones en Java, el código compilado no es compatible con el bytecode Java, porque Dalvik no es una máquina virtual Java, así que una aplicación Android no puede ejecutarse en las máquinas virtuales comunes. Durante el proceso de compilación en Android, sí que se generan los .class de los ficheros Java con su bytecode, pero después estos archivos se convierten al formato mejorado y comprimido de Dalvik, que dan como resultado archivos .dex

- *Librerías Core,* son diferentes librerías de clases de Java SE (*Standard Edition*) y Java ME (*Micro Edition*)

- *Framework de aplicaciones,*
  - *Administrador de Actividades (Activity Manager),* se encarga de controlar el ciclo de vida de las actividades (detallado en el apartado 1.5.1.) y gestionar la pila de actividades
  - *Administrador de ventanas (Window Manager),* gestiona las ventanas de las aplicaciones y crea superficies que pueden ser rellenadas por las actividades. Para crear esas superficies utiliza las librerías del *Gestor de Superficies*
  - *Proveedor de contenido (Content Provider),* son los encargados de encapsular los conjuntos de datos que puedan ser compartidos entre aplicaciones, como por ejemplo los proveedores de contenido que permiten a las aplicaciones acceder a los contactos guardados en el teléfono.
  - *Vista del Sistema (View System),* proporciona controles que permiten construir las interfaces de usuario en las distintas ventanas, como botones, casillas de verificación, listas, cuadros de texto, etc.
  - *Administrador de paquetes (Package Manager)*, las aplicaciones Android que se distribuyen en paquetes (ficheros .apk, apartado 2.1.1.) contienen tanto los archivos .dex como el resto de recursos que necesita la aplicación. Esta aplicación gestiona la instalación de los nuevos paquetes y permite obtener información de los que ya están instalados
  - *Administrador de Telefonía (Telephony Manager),* se encarga de las funciones asociadas con las llamadas de voz, SMS y MMS
  - *Administrador de Recursos (Resource Manager),* proporciona acceso a los elementos que se incluyen en una aplicación: imágenes, sonidos, cadenas de texto, etc.

- *Administrador de Ubicaciones (Location Manager),* es el que permite obtener la posición geográfica del dispositivo Android mediante el GPS o las redes disponibles.
- *Administrador de Notificaciones (Notification Manager),* es el encargado de proporcionar servicios para notificar al usuario cuando la aplicación quiera llamar su atención, por ejemplo un mensaje de alerta, un sonido, activar el vibrador, etc.
- *Cámara,* se encarga de proporcionar acceso a la cámara del dispositivo, tanto para fotografías como para grabar vídeo (si el dispositivo lo incluye)
- *Multimedia,* se encarga de proporcionar funcionalidades que permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo. Hacen uso de las librerías de bajo nivel *framework media*.

- *Aplicaciones,* es la capa superior de la arquitectura e incluye todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, como por ejemplo la aplicación de contactos, el navegador, el calendario, la calculadora, etc.
  Lo más importante de esta capa, es que todas las aplicaciones que se instalan en el dispositivo utilizan el mismo framework de aplicación (capa siguiente) para acceder a los servicios que proporciona el sistema operativo.

4.2 Componentes en una aplicación Android

Una vez entendidas las capas en las que se divide la plataforma Android, el siguiente paso es conocer los tipos de componentes software que forman una aplicación Android, los más importantes y habituales son:

- *Actividad (Activity),* es el componente principal de la interfaz gráfica y sirve para crear las distintas pantallas de la aplicación. Cada pantalla estaría representada por su Actividad, que siempre será una clase de tipo *Activity* o una subclase de ella.
- *Vista (View),* las vistas son el componente que se utiliza para construir los controles que van en las distintas pantallas (cuadros de texto, botones, listas desplegables, etc) y también es posible extender su funcionalidad o crear nuestros propios controles, con la creación de subclases de la clase *View.* Estas vistas también pueden ser definidas en archivos XML y hacer que el sistema genere los controles a partir de dicho archivo.
- *Widget,* son vistas de aplicaciones en miniatura, la mayoría interactivos, que pueden incluirse en otras aplicaciones, como por ejemplo añadirse en la Home o pantalla de inicio de un dispositivo, y recibir actualizaciones periódicas.
- *Layout,* es el componente que define la estructura visual para una parte gráfica, ya sea una *Activity* o un *Widget*, para organizar las vistas (en cuadrícula, de forma líneal,…) y pueden definirse en un archivo XML o bien directamente en el código.

- *Servicio (Service),* son componentes que no tienen interfaz gráfica y que se utilizan para ejecutar tareas pesadas en un segundo plano, como por ejemplo actualizar los datos en una base de datos, realizar una petición a un servicio web, etc. Es importante saber que aunque está en segundo plano, no crea un hilo (thread) de ejecución nuevo y forma parte del hilo principal de la aplicación, de modo que si la tarea es demasiado pesada, podría hacer que el resto de la aplicación se quedara colgada, así que en esos casos se recomienda crear un hilo de ejecución separado.
- *Intención (Intent),* es un componente básico para la comunicación entre los distintos componentes Android, y se utilizan para enviar mensajes o peticiones a otros componentes, por ejemplo para lanzar una actividad, un servicio, un mensaje broadcast o iniciar otra aplicación.
- *Receptor de mensajes Broadcast (Broadcast receiver),* componente que sirve para detectar determinados mensajes o eventos globales, para poder realizar las acciones oportunas, por ejemplo eventos como “batería baja” o “SMS recibido”
- *Proveedor de contenido (Content Provider),* componente que sirve para compartir datos entre distintas aplicaciones, sin necesidad de conocer detalles como su almacenamiento o estructura y sin comprometer a la seguridad, de esta forma otra aplicación podría acceder a los datos de nuestra aplicación mediante el *Content Provider* que definamos, sin que se conozca la implementación.
- *Procesos o hilos de ejecución (Processes and Threads),* si un componente de la aplicación arranca y la aplicación no tiene ningún otro componente en ejecución, el sistema Android arranca un nuevo proceso Linux para la aplicación con un solo thread de ejecución. Todos los componentes que arranquen a partir de aquí, serán iniciado por el mismo proceso y usará el mismo thread. No obstante, para no afectar al proceso e hilo principal, Android permite que los componentes se puedan iniciar en procesos separados que corran en un hilo diferente, algo muy útil cuando se trata de componentes que deben realizar operaciones de larga duración o que requieren un nivel muy alto de CPU.

El que una aplicación incluya uno o más de estos componentes, dependerá de la complejidad de la aplicación.

4.2.1. Ciclo de vida de una aplicación Android y métodos de la clase Activity

Cada aplicación se ejecuta en su propio proceso y es el SO Android quien se encarga de gestionarlos, lanzarlos o pararlos y decidir qué hacer en función de los recursos que tenga disponible o las órdenes que realice el usuario.

Cada proceso se encarga de una o varias actividades (Activity) y aunque el SO toma sus decisiones en cuanto a la gestión del proceso, es imprescindible saber cierta información sobre el funcionamiento de dicha gestión y el ciclo de vida por el que pasa el componente Activity para no perder información. Por ejemplo, si el usuario navega de una actividad en una aplicación a otra (por ejemplo al pulsar el botón “atrás” del móvil), el sistema operativo guarda el estado del proceso, lo mata y cuando el usuario vuelve de nuevo a la

aplicación lo crea y usa el estado guardado para poder restaurar una copia del mismo, si no se hubieran guardado bien los datos del usuario, estos se habrían perdido.

El diagrama siguiente contiene un resumen de los estados por los que pasa una actividad, y los métodos de la clase Activity que se ejecutan al pasar de uno a otro:

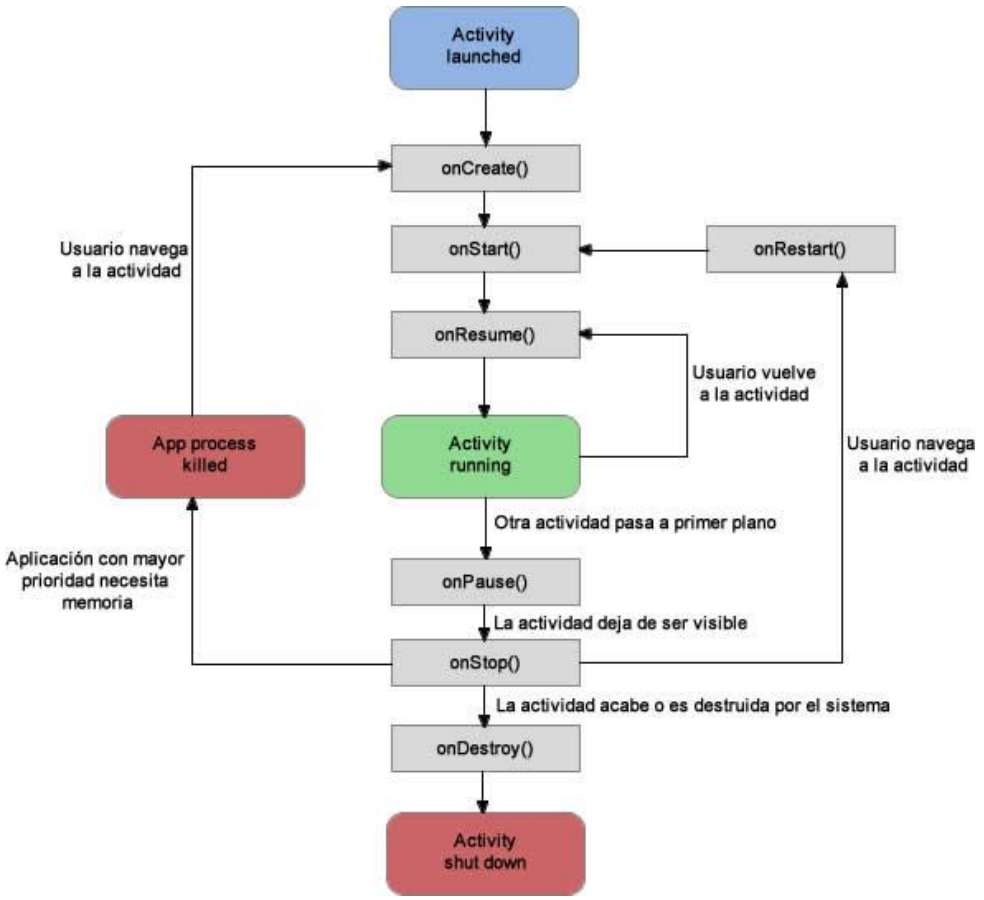


Figura 42. Ciclo de vida de una Actividad

Al empezar la Activity, los métodos que se ejecutan son onCreate(), onStart() y onResume(), y la actividad queda corriendo en el SO. Si el usuario se va a mirar otra aplicación o el SO necesita recursos, primero se ejecuta el método onPause() y se quita de la pantalla, después los métodos onStop() o el método onDestroy() podrían llegar a ejecutarse, por ejemplo en el caso de que el SO necesita liberar memoriao no ejecutarse.

Después del método Stop(), si el usuario vuelve a la aplicación y no se había ejecutado el método Destroy(), para iniciarla de nuevo el sistema ejecutará los métodos onRestart(), onStart() y onResume()

Por tanto al desarrollar, es importante guardar la información antes de que pueda ejecutarse el método onStop() y obtener dicha información cuando se ejecute un método onRestart()

4.3 Tipos de aplicaciones móviles: nativas, web o híbridas

Una vez familiarizados con la arquitectura de Android, para el diseño de una aplicación móvil, es importante conocer los distintos tipos que existen, para qué dispositivos están destinados y en base a las características de nuestra aplicación, elegir el más adecuado.

En el siguiente apartado se describen cada uno de estos tipos y qué diferencias existen entre ellos, y así escoger el apropiado para la aplicación MyTopRoom.

4.3.1. Aplicaciones nativas

Son aplicaciones comprimidas en un archivo binario, que se descarga y almacena en el dispositivo móvil. Para acceder a este archivo, los usuarios utilizan los conocidos "app stores" u otras funcionalidades que incluyen los propios dispositivos para descargarlas.

Una vez se ha instalado, el SO la ejecuta cuando el usuario la selecciona en la pantalla de inicio, pero estos tipos no requieren ninguna otra aplicación o contenedor para ejecutarse.

Ya en ejecución, la aplicación hace uso explícito de la API del sistema operativo del dispositivo, para acceder a todas sus funcionalidades (GPS, almacenamiento, radio,…) y por último la parte visual, se construye con el uso de las especificaciones de cada tipo de sistema operativo.

El proceso de construcción y distribución de aplicaciones nativas es similar en los distintos sistemas operativos móviles que tenemos (iOS, Android, Blackberry OS, Symbian OS, Windows Mobile…), pero los lenguajes y herramientas que se utilizan son distintos. La siguiente figura muestra un resumen de dichos procesos:

Figura 43. Proceso desarrollo y distribución de aplicaciones móviles

El desarrollador escribe el código fuente de la aplicación, unido a los recursos estáticos necesarios (imágenes, archivos de música,…) y con el uso de las herramientas de desarrollo (SDK) que tienen los distintos proveedores de SO móviles, lo compila para obtener el fichero binario ejecutable, y por último lo empaqueta con los recursos estáticos para obtener el paquete final, que será distribuido mediante los distintos "app stores" u otros mecanismos.

Todo este conjunto de herramientas (compilador, empaquetador y distribuidor) se conoce como el "SDK del sistema operativo móvil" y lo más importante es que es específico de cada plataforma y sistema operativo, aunque muchos de los pasos sean similares. La siguiente figura muestra las diferencias entre algunos de sistemas operativos móviles del mercado, en cada uno de los pasos:

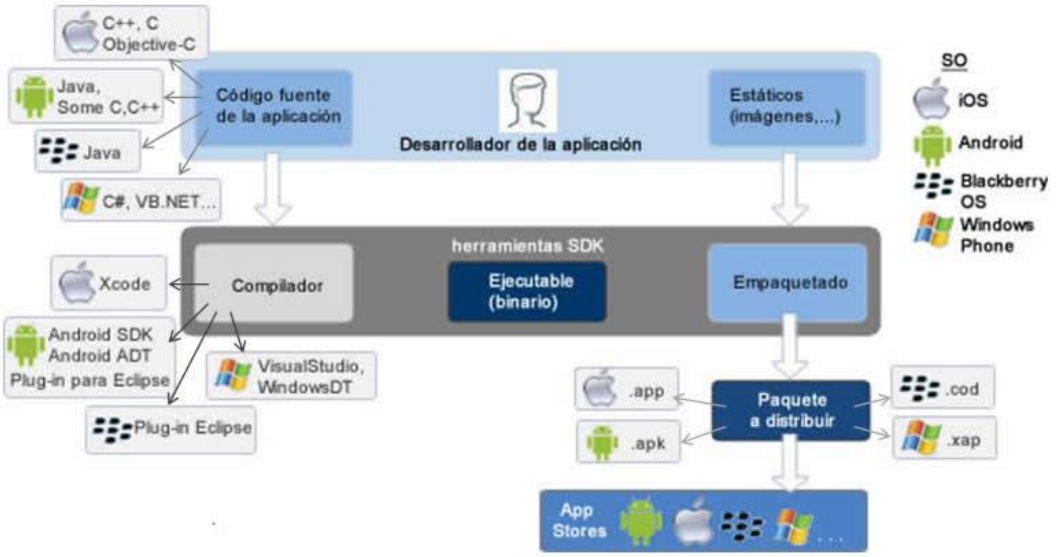


Figura 44. Lenguajes y herramientas de desarrollo en SO móviles

Como podemos ver, si decidimos construir una aplicación nativa, el programarla en un lenguaje no significa que sirva para todos los dispositivos, por lo que desarrollarla para el mayor número de dispositivos requiere el dominio de varios lenguajes y resulta bastante costoso.

Con esta información lo más obvio sería desechar el hacer una aplicación nativa, sin embargo, hay que tener en cuenta sus ventajas:

- Se ejecutan directamente por el SO, sin necesidad de contenedor o aplicaciones extras
- Hace uso explícito de la API del SO

Figura 45. Interacción entre aplicación nativa y dispositivo móvil

En la figura anterior, vemos que la aplicación con llamadas directas a la API del sistema operativo, puede hacer uso de todos los componentes hardware del dispositivo (pantalla táctil, redes, GPS,…) Estas API son muy rápidas y potentes, y permiten explotar al máximo la potencia del móvil. No obstante como cada API es distinta en función del SO, el desarrollo sigue siendo costoso.

Los proveedores de los distintos SO móviles, para mejorar el coste de este proceso y facilitar el desarrollo de aplicaciones nativas, junto con la API del SO, incorporan una serie de aplicaciones con distintas funcionalidades que pueden ser utilizadas mediante una API de alto nivel. Por ejemplo la aplicación de contactos, un navegador web, una aplicación de correo electrónico, el calendario, etc. De esta forma, una aplicación nativa, con esta API de alto nivel podría obtener los eventos del calendario, obtener la lista de contactos, enviar un correo electrónico,…, sin necesidad de tener que implementar estas aplicaciones desde cero. Para la parte visual, hacen lo mismo, como los controles, componentes y look & feel son distintos entre fabricantes, lo que hacen es ofrecer un kit de herramientas “GUI Toolkit” que facilita el desarrollo de la interface para cada tipo de SO, aunque el diseñador deberá conocer las diferencias entre cada uno de ellos si quiere que la aplicación sea multi-plataforma.

Como ejemplos de aplicaciones nativas podemos citar Skype o Twitter para Android, o la famosa angry birds.

4.3.2. Aplicaciones web

Unas de las aplicaciones incorporadas por los proveedores de SO, mencionada en el apartado anterior, es el navegador web. Las últimas tecnologías en el desarrollo web como HTML5, CSS y JavaScript han obligado a los proveedores de SO a enfocarse en mejorar el motor de renderización de las páginas en sus navegadores, para mostrar las páginas desarrolladas con ellas, en lugar de crear distintas especificaciones de implementación web.

Los usuarios pueden abrir el navegador del móvil y visualizar la aplicación web, disponible en un servidor web, o bien pueden distribuirse en un archivo ejecutable como las nativas, de forma que al instalarse en el dispositivo se crea un icono en la home igual que las nativas, y cuando el usuario la selecciona la página se abre en el navegador, pero sin mostrar el marco del navegador (Chrome o IE Explorer) o la barra de direcciones, de forma que el usuario no percibe que la aplicación se está ejecutando en el navegador. Además si se utiliza HTML5, con sus últimos elementos de audio, video, canvas, etc. se puede conseguir una interacción del usuario con la pantalla.

Por tanto distinguimos dos tipos:

- Las aplicaciones móviles puramente web (se instalan como una nativa, proporciona una UI –*interfaz gráfica de usuario*- interactiva, se puede ejecutar por completo en el cliente con Ajax y pueden estar disponibles sin conexión)

- Las aplicaciones móviles Web Sites (están disponibles en un servidor web, el usuario las ejecuta cuando escribe su dirección en el navegador, tienen una navegación estática y requieren tener conexión, de la misma forma que al usar el navegador en un PC, hacen llamadas al servidor y cargan los elementos de la página)

Otro detalle a tener en cuenta, es la cantidad de toolkits y frameworks que tenemos disponibles para desarrollar la parte web para los navegadores de móviles, que hace que sea muy rápido y sencillo, sobre todo para personas que están familiarizadas con el desarrollo web tradicional. Por ejemplo el framework JQuery Mobile, HTML5, el lenguaje JavaScript o el último nivel de hojas de estilos CSS3.

Con el potencial de estas nuevas tecnologías y la posibilidad de instalar la aplicación en el dispositivo, parece muy recomendado utilizar este tipo para la implementación de la aplicación móvil, a menos que sea necesario acceder a ciertas funciones del dispositivo.

En la siguiente figura se muestra la interacción entre una aplicación móvil web y el dispositivo:

Figura 46. Interacción entre aplicación web y dispositivo móvil

Algunos de los componentes del dispositivo son accesibles y puede interactuarse con ellos mediante el navegador, como la pantalla táctil (html5+css3) o el teclado, pero otros como la cámara o micrófono no son accesibles, y otros lo son de forma limitada, como algunos de los datos de GPS mediante las últimas funciones de html5. No obstante, HTML5 sigue en evolución y puede que un día todas las funciones del dispositivo sean accesibles mediante el navegador.

Al motor de renderización o del navegador se le conoce como Webkit, es un proyecto open source que sirve como base para los distintos navegadores de dispositivos móviles y que tiene implementadas la mayoría de las funciones de HTML5, y aunque cada proveedor de SO móvil realiza sus cambios en el Webkit, implementar en html5 nos da una amplia gama de compatibilidad entre los distintos dispositivos, a pesar de las pequeñas diferencias que puede haber entre unas versiones y otras de SO móvil.

Como resumen, elegir entre una aplicación nativa o web dependerá de las condiciones:

- *Acceso a las funciones del dispositivo,* en las nativas completo y en las web parcial
- *Velocidad,* mucho más rápidas las nativas
- *Coste de desarrollo,* más razonables las web ya que las nativas hay que conocer varios lenguajes y utilizar distintos SDK para llegar al mayor número posible de dispositivos
- *App Store,* las nativas se descargan desde el app store instalado en el dispositivo, mientras que las web se consiguen a través de otros canales, como páginas web, lo que implica un esfuerzo de marketing para que llegue al conocimiento de los usuarios
- *Proceso de aprobación,* mientras que las aplicaciones de un app store tienen que cumplir con muchos requisitos, tanto si es nueva como si es una actualización, las web no tienen ninguno.

Entre algunas de las aplicaciones web podemos citar wordreference para android o youtube.

4.3.3. Aplicaciones híbridas

Por último, para obtener las ventajas de una aplicación web, sin renunciar a poder interactuar con las funciones del dispositivo nacen las aplicaciones híbridas.

Estas consisten en una aplicación nativa, donde el código html está embebido. Por un lado, hay porciones de código nativo que accede a las funciones del dispositivo y por otro, hay porciones escritas con tecnologías web: html, css y js.

Estas aplicaciones pueden tanto descargarse de páginas web como instalarse en el dispositivo como paquete.

Si vemos la misma figura de interacción entre la aplicación móvil híbrida y el dispositivo, los beneficios son notables:

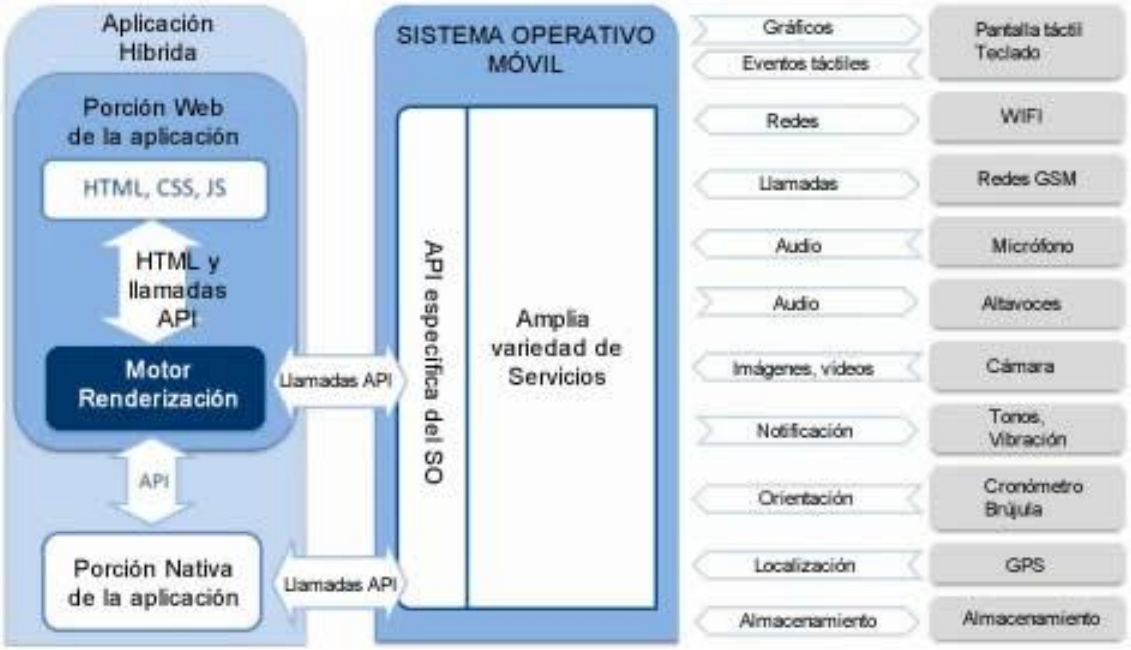


Figura 47. Interacción entre aplicación híbrida y dispositivo móvil

La porción nativa de la aplicación, puede acceder a todas las funciones del dispositivo (calendario, contacto, cámara,…) mediante llamadas a la API del SO; la porción web, mediante las tecnologías web puede acceder también a aquellas que están disponibles por el motor de renderización, como las vistas en el apartado anterior con HTML5. Además para disponer de la información de los componentes o funciones del dispositivo desde el navegador, podemos implementar el puente entre las llamadas del motor y las llamadas de la porción nativa, o bien utilizar uno de los frameworks existentes creados para ello; uno bastante popular es PhoneGap que queda representado con la interacción anterior en la figura 48.

Para implementar la parte nativa, hay que tener en cuenta lo detallado en el apartado 4.3.1 de aplicaciones nativas, en cuanto al lenguaje de programación y herramientas de desarrollo que debemos utilizar y para la parte web, tenemos dos opciones:

- Dejar los recursos web y estáticos (páginas html, css, js,…) en un servidor web y el resto en la aplicación móvil, de forma que al ejecutarla la aplicación se conectará al servidor para obtener las páginas y demás recursos. El inconveniente es que el usuario, si hay una conexión lenta, tendrá que esperar a que se descarguen los recursos para ver el contenido, pero de esta forma si tenemos que hacer actualizaciones, se hacen una vez en el servidor y estarán disponibles para los usuarios cada vez que abran la aplicación.

- Por el contrario, incluir los recursos web y estáticos dentro de la aplicación móvil y empaquetarla en el ejecutable. La ventaja es que los recursos quedan almacenados en el dispositivo del usuario y por tanto no tiene que esperar para ver el contenido, está todo almacenado en el mismo, pero la desventaja es que para actualizar las páginas habrá que empaquetar de nuevo la aplicación y hacer que el usuario se actualice a la nueva versión.

Figura 48. Interacción entre aplicación híbrida y dispositivo móvil con PhoneGap

Como ejemplo de aplicaciones híbridas podemos citar facebook, bookingPro.

Teniendo en cuenta todo lo comentado, para la aplicación MyTopRoom optamos por implementar una aplicación nativa, donde todos los recursos estén empaquetados dentro del ejecutable de la aplicación, y para los datos dinámicos (datos de acceso u obtención de nuevas ofertas por ejemplo) se realizarán llamadas al servidor web externo (motor de disponibilidad) que los proporciona y se grabarán en la base de datos del dispositivo.

4.4 Tecnologías empleadas

En este apartado se resume las tecnologías empleadas en la aplicación:

- *Java,* la aplicación nativa es para Android y aunque en algunas versiones puede desarrollarse en C++, esto requiere de herramientas extras y puede complicar el proceso de compilación y empaquetado, por lo que la mayoría de los desarrolladores opta por hacerlo en Java. En nuestro caso utilizamos Java.

- *Apache HTTPComponents,* conjunto de herramientas de componentes Java de bajo nivel enfocados para trabajar con el protocolo HTTP y asociados

- *JSON,* las respuestas de los servicios REST se intercambian con este formato ligero de datos. La aplicación parsea las respuestas en este formato para convertirlas en clases Java (POJOs) que representen el modelo de dominio. En este proyecto las respuestas del servidor web seguirán este formato.

- *SQLLite,* gestor de bases de datos de Android se utiliza para crear una base de datos en el dispositivo y guardar los datos que deben ser persistentes.

4.5 Herramientas de software y recursos hardware

4.5.1. Herramientas de software

- *ADT Bundle para Windows*

Como entorno de desarrollo –IDE- para escribir la aplicación móvil en Java, si se opta por utilizar un IDE ya instalado, como puede ser Eclipse, es necesario después incorporar el plug-in de Android Developer Tools –ADT- para la depuración y empaquetado, y los paquetes del Android SDK, que incluyen la plataforma Android.

En nuestro caso, utilizaremos otra opción para desarrolladores Android, que es el paquete completo ADT para Windows, llamado ADT Bundle. En él se incluye un eclipse, el plug-in de ADT, herramientas y la plataforma Android, y una serie de emuladores predefinidos para simular dispositivos móviles en el PC.

En el anexo 1 se describe la instalación de este paquete de desarrollo y otros detalles de configuración del entorno.

4.5.2. Recursos Harware

- *Portátil*

Ordenador personal portátil con Windows como Sistema Operativo.

- *Nexus 4*

Dispositivo móvil empleado para la instalación de la aplicación en un dispositivo real, con Android como Sistema Operativo.

4.6 Estructura y componentes de la aplicación “MyTopRoom”

Al crear una aplicación Android (ver en Anexo 1) se crea una estructura de carpetas por defecto, que ampliaremos para incluir las clases necesarias de la aplicación:

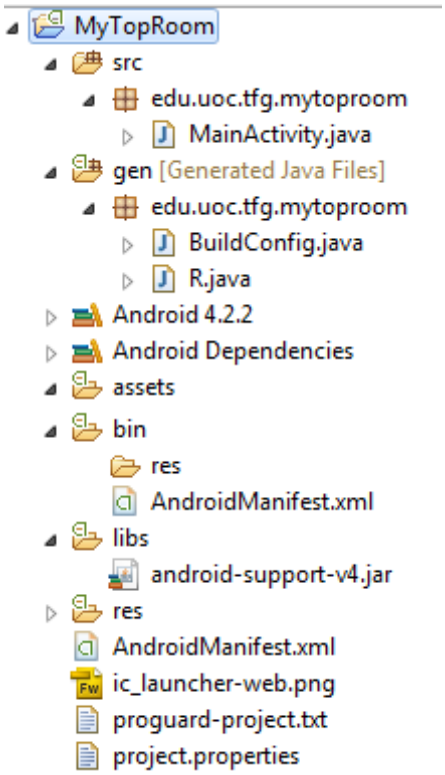


Figura 49. Estructura de una aplicación Android

- *Carpeta /src/,* en ella debe incluirse todo el código fuente de la aplicación, clases auxiliares. Por defecto siempre incluye una primera Activity que representa la primera pantalla
- *Carpeta /gen/,* contiene unos elementos generados automáticamente al compilar el proyecto. Uno de ellos “R.java”, contiene una serie de constantes con los ID de todos los recursos que están en la carpeta /res/, para que podamos acceder a ellos desde el código.
- *Carpeta /assets/,* contiene todos los demás ficheros necesarios para la aplicación, como ficheros de configuración, de datos, etc. y se diferencian de los recursos anteriores en que no se genera un ID para acceder a ellos, sino que se accede directamente con la ruta. Para aplicaciones híbridas, los ficheros de la parte web (html, js, css) se colocan bajo esta carpeta.
- *Carpeta /bin/,* es donde se colocan todos los elementos compilados, entre ellos el archivo ejecutable “.apk” de la aplicación para instalarlo en el dispositivo.
- *Carpeta /libs/,*es la que contendrá las librerías auxiliares que necesita la aplicación, la mayoría de los casos en formato “.jar”

- *Carpeta /res/,* es donde se incluyen los recursos del proyecto como imágenes, vídeos, cadenas de texto, etc. Entre los recursos creados por defecto el fichero "*activity_main.xml"* contiene la definición de la interfaz gráfica de la aplicación.
- *Fichero "AndroidManifest.xml"*, es el que contiene la definición en XML de los aspectos más importantes de la aplicación, como por ejemplo su identificación (el nombre, versión, icono…), sus componentes (pantallas, mensajes,…), las librerías auxiliares utilizadas, o los permisos que necesita la aplicación para su ejecución.

La estructura de clases de la aplicación, dentro de la carpeta *SRC,* dónde se incluyen tantos las clases Activity para representar las pantallas como las clases de negocio, dominio e integración, queda de esta forma:

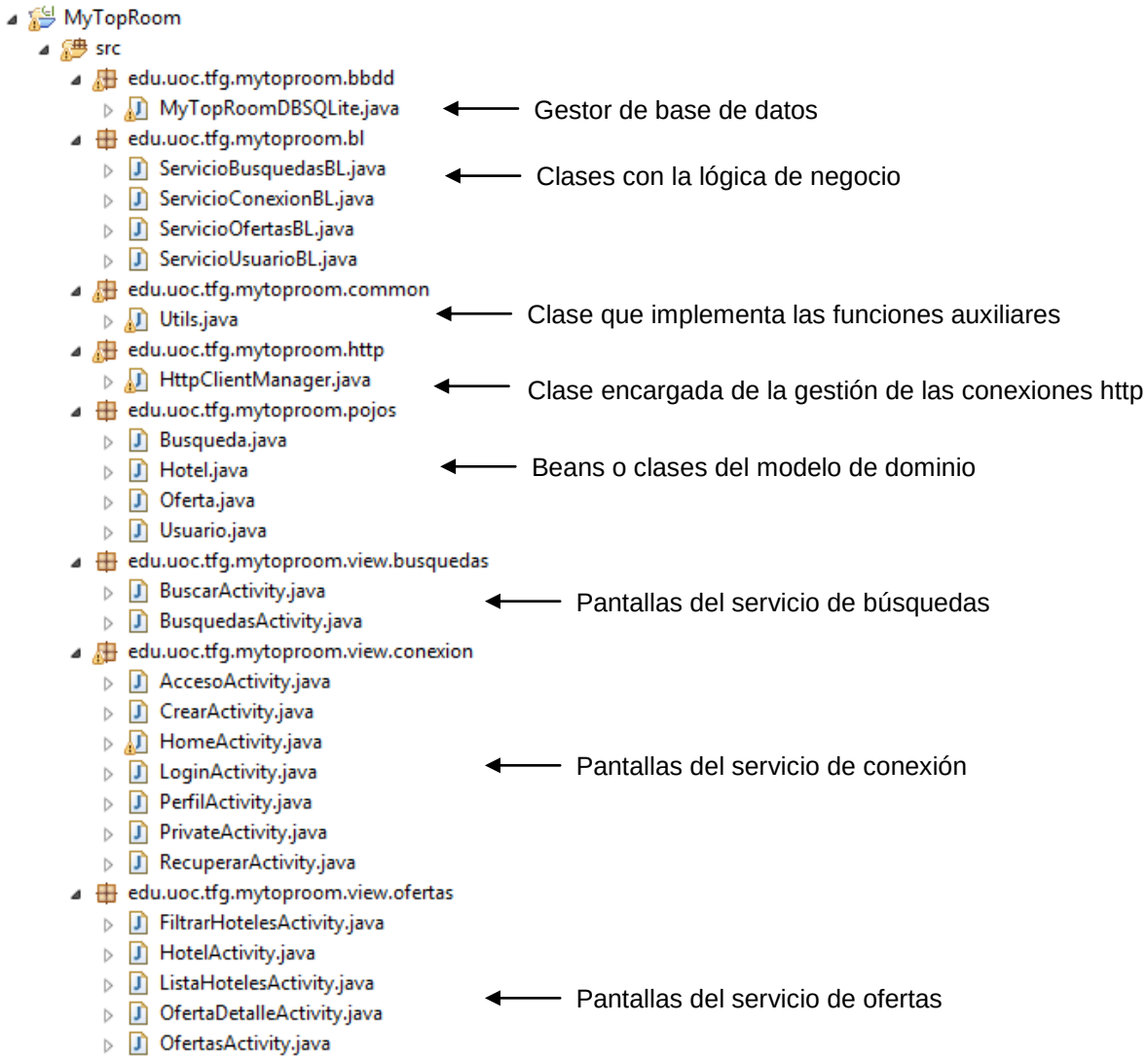


Figura 50. Estructura de clases de la aplicación

En cuanto a la carpeta de recursos (RES) el contenido queda:

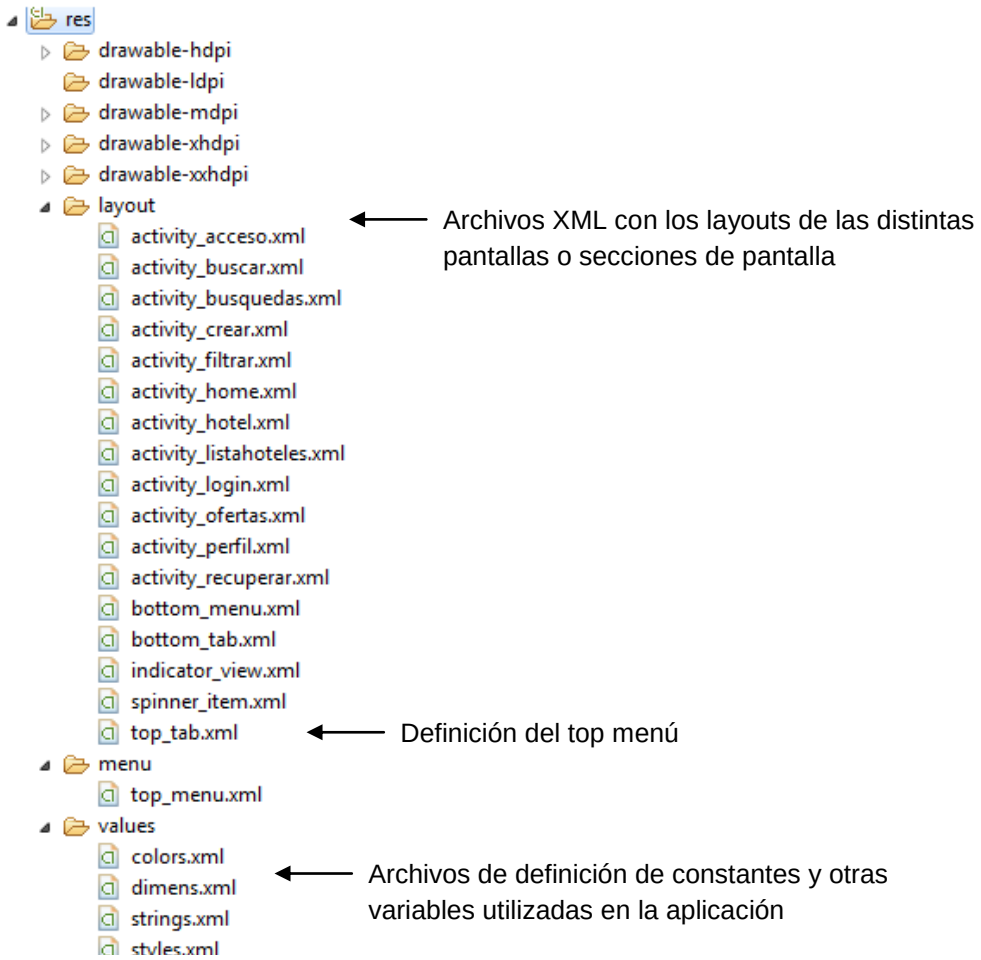


Figura 51. Carpeta de los recursos utilizados por la aplicación

No todos los componentes de Android citados en el apartado 4.2. han sido necesarios para esta aplicación y solo se han utilizado los siguientes:

- *Actividad (Activity),* cada pantalla está representada por su actividad, excepto en los casos en los que se utilizan pestañas o botones como por ejemplo al entrar a la aplicación con el usuario y contraseña. A partir de ahí se muestra siempre un menú inferior que se define en la actividad "PrivateActivity.java"; esta actividad es de clase "FragmentActivity" para que pueda mostrar siempre este menú en la parte inferior y en la parte central el fragmento que corresponda: la pantalla home del usuario, el formulario de búsquedas, pantalla de ofertas, búsquedas, etc. Por eso todas estas secciones de pantalla no son clases que hereden de "Activity" sino que heredan de "Fragment"

- *Vista (View),* para construir los controles que van en las distintas pantallas. Se definen en sus respectivos archivos XML dentro de la carpeta "Layout" y después en el método de creación de la actividad se establece la vista:

```
setContentView(R.layout.activity_login);
```
← Archivo "activity_login.xml"

- *Layout,* para definir la estructura visual de las pantallas siendo el más utilizado el "RelativeLayout" que permite colocar los componentes en base a la posición de otros. Por ejemplo en el caso del Layout para la pantalla "Crear Cuenta", la primera caja de texto se coloca en base a la posición de la capa superior y todos los demás debajo del control superior:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".CrearActivity" >

    <EditText
        android:id="@+id/txtNombre"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="@string/nombre"
        android:maxLength="150">
        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/txtEmail"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/txtNombre"
        android:layout_below="@+id/txtNombre"
        android:ems="10"
        android:inputType="textEmailAddress"
        android:hint="@string/email"
        android:maxLength="150"/>

    <EditText
        android:id="@+id/txtPass"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/txtEmail"
        android:layout_below="@+id/txtEmail"
        android:ems="10"
        android:inputType="textPassword"
        android:hint="@string/password"
        android:maxLength="100"/>

    <EditText
        android:id="@+id/txtConfPass"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/txtPass"
```

Posición con respecto a la capa superior

El campo de texto para el email se colocará debajo del campo de texto nombre

```
            android:layout_below="@+id/txtPass"
            android:ems="10"
            android:inputType="textPassword"
            android:hint="@string/confirmarpassw"
            android:maxLength="100"/>

        <Spinner
            android:id="@+id/listPreguntas"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/txtConfPass"
            android:layout_below="@+id/txtConfPass"
            android:prompt="@string/pregunta"
            android:textColor="@color/mediumgrey"/>

        <EditText
            android:id="@+id/txtRespuesta"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/txtPass"
            android:layout_below="@+id/listPreguntas"
            android:ems="10"
            android:hint="@string/respuesta"
            android:maxLength="100"/>

        <Button
                  android:id="@+id/btnCrear"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_centerHorizontal="true"
                  android:layout_marginBottom="14dp"
                  android:layout_below="@+id/txtRespuesta"
                  android:paddingLeft="15dip"
                  android:paddingRight="15dip"
                  android:text="@string/crear"
                  style="@style/ButtonPink"
android:background="@drawable/btn_pink" />

       <TextView
                  android:id="@+id/volverInicio"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_centerHorizontal="true"
                  android:layout_alignParentBottom="true"
                  android:text="@string/volver"
                  android:textColor="@color/pink"
                  android:textColorLink="@color/darkgrey" />

</RelativeLayout>
```

- *Intención (Intent),* utilizado para pasar entre las distintas activities y lanzar la que sea necesaria. Por ejemplo si el usuario pulsa el enlace “Recuperar contraseña”, la siguiente Activity que se debe mostrar es la que permite recuperar contraseña, así que dentro de la actividad de Login, se configura el listener sobre el enlace llamado “recuperarPassw” de modo que cuando se pulse lance la otra actividad:

```
recuperarPassw.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
              Intent itemIntent = new Intent(LoginActivity.this,
RecuperarActivity.class);
              startActivity(itemIntent);

        }
    });
```

En este caso no ha habido que informar de otros objetos o valores en el "Intent" y tan solo sirve para lanzar la actividad de RecuperarActivity desde LoginActivity, otros casos como por ejemplo al acceder por primera vez, la aplicación debe mostrar la pantalla con los "Datos de Perfil" en lugar de la Home, así que en ese componente "Intent" sí que se informará de este valor y así la clase "PrivateActivity" que se encarga de mostrar el fragmento adecuado en la parte central mostrará los datos de perfil en lugar de la home:

```
              Intent itemintent = new Intent(CrearActivity.this,
PrivateActivity.class);
              itemintent.putExtra("perfil", "si");
                    startActivity(itemintent);
```

Y después la actividad "PrivateActivity" podrá leer el contenido:

```
String valuePerfil = getIntent().getExtras().getString("perfil");
```

- *Procesos o hilos de ejecución (Processes and Threads),* en el caso de esta aplicación la tarea que puede llevar más tiempo y como consecuencia afectar al hilo principal es la de acceder al servidor web para descargar los datos, por eso se ha utilizado la clase "Thread", que permitirá lanzar otro hilo para esa operación y al finalizar dará el aviso al hilo principal:

```
runnable = new Runnable(){
public void run() {
    try {
         executeHttpPostAsync(activity, getUrlService());
    } catch (Exception e) {}
  }
};

Thread thread =  new Thread(null, runnable, "Background");
thread.start();

progressDialog = ProgressDialog.show(activity,
getStrmessageheadloading(), getStrmessagebodyloading(), true);
```

El método que lanza la petición POST HTTP es la que ejecutamos en el nuevo hilo y además se utiliza el componente "ProgressDialog" para mostrar el icono de cargando. Una vez que acabe o se produzca algún error, devolvemos el control al hilo principal mediante:

```
activity.runOnUiThread(returnRes);
```

Siendo “returnRes” nuestra clase “Runnable” que además dejará de mostrar el “ProcessDialog”:

```
private Runnable returnRes = new Runnable() {
        public void run() {
              progressDialog.dismiss();

listenerExecuteHttpPostAsync.onExecuteHttpPostAsyncListener(getResponseBody());
            }
        };
```

5. Testing

Al desarrollar aplicaciones, es común que se cometan errores, no solo por parte del programador, a veces es porque se olvidan requisitos, o estos se implementan de forma incorrecta o el cliente los expresó mal, y es el proceso de Testing con sus pruebas el que intenta detectar estos defectos para solucionarlos y mejorar la calidad del producto.

5.1 Niveles de Testing

El Testing se ejecuta en varios niveles durante el desarrollo y mantenimiento del Software, y las pruebas que forman parte de él se pueden distinguir por dos tipos de clasificación:

- En función del ámbito del sistema: es distinto probar una sola funcionalidad, que un grupo de funcionalidades relacionadas; se distinguen tres tipos de pruebas:
  - *Pruebas unitarias,* que verifican la funcionalidad de los componentes de forma aislada al resto.
  - *Pruebas de integración,* para verificar la interacción entre varios de los componentes que forman parte del sistema
  - *Pruebas de sistema,* que verifican el comportamiento de todo el sistema, todos sus componentes a la vez

- En función del objetivo de la prueba (probar la funcionalidad, el rendimiento, la usabilidad,…) podemos distinguir estos tipos:
  - *De aceptación,* son las que comprueban el comportamiento del sistema frente a los requisitos del cliente e implica definir los escenarios de las posibles entradas y salidas que deben producirse. A menudo son denominadas “UAT” (*User Acceptance Test*) test de aceptación de usuario, porque son ejecutadas por los usuarios finales antes de “aceptar” la aplicación y son uno de los tipos fundamentales que deben aparecer en todo proceso de testeo
  - *De instalación,* una vez desarrollado y validado las pruebas de aceptación, estas pueden comprobar el software sobre una instalación en un entorno concreto
  - *Alpha (desarrolladores) y Beta (usuarios),* son ejecutadas por un conjunto potencial de usuarios sobre el software antes de liberarlo, casi siempre siguiendo un plan de pruebas marcado, y estos usuarios pueden ser de la empresa (Testing *alpha*) o personal externo (Testing *beta*)
  - *De conformidad y fiabilidad,* son las que verifican que el software cumpla con la especificación

- *De regresión,* que consisten en probar que los test que superaba previamente el software, lo hace también después de haber introducido modificaciones, sobre todo en los desarrollos incrementales, donde a menos que haya cambiado la funcionalidad, el software debe superar los mismos test que antes de los cambios
- *De rendimiento,* que verifican los requisitos sobre el rendimiento del sistema, como la capacidad o el tiempo de respuesta
- *De seguridad,* enfocadas a verificar que el software está protegido de ataques externos y a certificar que se protege la confidencialidad, integridad y disponibilidad del sistema y sus datos
- *De estrés,* pruebas diseñadas para confrontar los programas a situaciones límite, como un volumen anormal de usuarios
- *De recuperación,* para verificar que el programa se reinicia correctamente ante un desastre
- *De usabilidad e interacción persona-ordenador (u otro dispositivo),* que pretenden evaluar la facilidad con la que los usuarios finales aprenden a utilizar el software. Esto es especialmente crítico en aplicaciones móviles, dónde los usuarios esperan poder utilizarlas sin tener que pasar por una formación previa.

Al definir estos tipos de pruebas, hay que adaptarlos al tipo de aplicación y en este caso, para una aplicación móvil se deben tener en cuenta otros detalles como la multitud de tipos de dispositivos donde puede instalarse, las pérdidas de cobertura temporales, la gestión que la aplicación hace de los recursos como la memoria o la batería.

5.2 Testing en aplicaciones móviles

5.2.1 *Pruebas de instalación*

En el caso de aplicaciones nativas o híbridas disponibles desde un app store u otro canal, hay que realizar pruebas de instalación en el mayor número posible de dispositivos.

En el caso de aplicaciones para Android el reto es mucho mayor porque hoy día existen cientos de dispositivos con hardware y versiones de software diferentes. Se puede invertir en la compra de algunos dispositivos, pero para la mayoría de empresas supondría un coste inasumible.

Como soluciones alternativas hoy en día se puede recurrir a empresas que se dedican a realizar Testing de aplicaciones móviles y disponen de un número importante de dispositivos, o bien, se pueden utilizar los emuladores que se incluyen en las herramientas de software de Android. Estos emuladores permiten que el proceso de Testing sea mucho más barato y además con ellos se pueden realizar simulaciones de pérdida de conexión, incrementar o disminuir la memoria del dispositivo, validar la pantalla en distintas resoluciones acorde con los tamaños de los dispositivos, etc.

5.2.2 *Pruebas de aceptación*

Para aplicaciones móviles son las que realiza el usuario desde la pantalla del dispositivo e incluye la validación de los controles de formulario, comprobación de los mensajes, verificación del flujo de navegación entre las pantallas o pruebas de cambios de orientación de la pantalla, que valida tanto la pérdida posible de datos como el muestreo de los componentes gráficos.

5.2.3 *Pruebas de rendimiento y estrés*

Existen más de 400 operadores de redes móvil en el mundo y cada uno soporta múltiples tecnologías de red (LTE, CDMA, GSM,…) Cada red tiene su propia infraestructura para trasladar los paquetes de la red móvil en paquetes TCP utilizados por los servidores Web, algunos operadores incluso cuentan con un proxy Web que decide cómo, cuándo y si estás autorizado para conectar con un sitio Web particular. Además algunos de estos "proxies" eliminan información de las cabeceras HTTP, las cuales pudiera necesitar nuestra aplicación.

Cada red está asociada a una localización y si probar todos los dispositivos no es viable, tampoco lo es viajar por las distintas partes del mundo para probar los distintos operadores de red móvil.

No obstante, con un operador de red a nuestro alcance, se pueden realizar pruebas de rendimiento importantes como comprobar la aplicación con una red WIFI, otra 3G y otra 2G y ver cómo se comportan los accesos a Internet, si nuestra aplicación utiliza servicios externos, comprobar si se liberan los recursos (GPS, cámara) que reserve la aplicación, qué sucede si intenta acceder a un servicio sin conexión o si se cae la conexión durante la transferencia de datos o como se comporta la memoria y si se libera de forma correcta.

5.2.4 *Pruebas de usabilidad*

Estas pruebas determinan la facilidad con la que los usuarios pueden utilizar una aplicación. En el caso de aplicaciones móviles, estas pueden interrumpirse mientras son ejecutadas por el usuario, por una pérdida de cobertura o una llamada entrante, y el diseño de la aplicación debe ser capaz de volver al proceso en curso antes de la interrupción.

Además de realizar pruebas provocando interrupciones sobre la aplicación, es importante testear primero la aplicación con usuarios reales para comprobar otros detalles como:

- Sencillez de la navegación, si el usuario ha necesitado ayuda para volver a una pantalla o encontrar alguna información
- Facilidad de uso de la aplicación, los usuarios deben ser capaces de entender el funcionamiento sin necesidad de tener que leer ningún manual o guía.

5.2.5 *Pruebas de seguridad*

La parte de envíos de datos sensibles deben ir codificados, al igual que en las aplicaciones web (HMAC-SHA), pero en aplicaciones móviles también debemos tener en cuenta los archivos que la aplicación guarda en el dispositivo, ya que éste podría llegar a manos de terceros.

5.2.6 *Pruebas de recuperación*

Para comprobar que la aplicación cuenta con un plan de almacenamiento y recuperación, ante situaciones como por ejemplo si se agota la batería, o llega un SMS o llamada. Con estas pruebas se comprueba en qué momentos hay backup y qué datos contiene.

5.2.7 *Pruebas de conformidad y fiabilidad*

Consisten en comprobar que se ha validado en los dispositivos, versiones de SO y modelos acordados y que se muestra bien en las distintas pantallas y resoluciones.

5.3 Plan de pruebas para la aplicación “MyTopRoom”

Los tipos de pruebas realizados son:

- Pruebas unitarias: realizadas durante el desarrollo en el IDE utilizado, definido en el Anexo 1.
- Pruebas de sistema: se ha instalado y probado la aplicación en un dispositivo real –Nexus 4- y otro virtual “DispositivoVirtual” definidos en el Anexo 1.
- Pruebas de aceptación: en el siguiente apartado se han definido las pruebas de los casos de usos analizados antes con los distintos escenarios.

5.3.1 *Pruebas de aceptación (User Acceptance Tesing)*

Prueba 1	
Caso de uso	Crear una Cuenta nueva
Objetivo	Probar el funcionamiento del flujo básico para crear una cuenta
Precondición	La aplicación está instalada No existe ninguna otra Cuenta relacionada con el dispositivo móvil
Descripción de la prueba	El usuario introduce los datos válidos en el formulario y crea una Cuenta
Resultados esperados	Se crea la Cuenta en el servidor, se almacenan estos datos en el dispositivo y se redirige al usuario a la página con los datos de perfil para configurar la moneda

Tabla 19. Prueba de aceptación 1

Prueba 2	
Caso de uso	Crear una Cuenta nueva
Objetivo	Probar el funcionamiento del flujo alternativo al crear una cuenta cuando ya existe una en el dispositivo
Precondición	La aplicación está instalada Existe otra Cuenta relacionada con el dispositivo móvil
Descripción de la prueba	El usuario introduce los datos válidos en el formulario y pulsa crear cuenta
Resultados esperados	El sistema detecta que hay otra cuenta y solicita la confirmación al usuario para eliminarla y crear la nueva. El usuario confirma y se elimina la anterior y se crea la nueva

Tabla 20. Prueba de aceptación 2

Prueba 3	
Caso de uso	Iniciar sesión
Objetivo	Probar el acceso a la aplicación mediante la validación de usuario
Precondición	La aplicación está instalada El usuario ha creado una Cuenta
Descripción de la prueba	El usuario introduce la dirección de correo y contraseña, pulsa acceder y la aplicación muestra la home de la parte privada
Resultados esperados	Se muestra la home de la parte privada

Tabla 21. Prueba de aceptación 3

Prueba 4	
Caso de uso	Iniciar sesión
Objetivo	Probar el acceso a la aplicación mediante datos de usuario no válidos
Precondición	La aplicación está instalada El usuario ha creado una Cuenta
Descripción de la prueba	El usuario introduce la dirección de correo y/o contraseña incorrectos, pulsa acceder y la aplicación muestra el mensaje de error
Resultados esperados	Se muestra el error informando que los datos no son correctos

Tabla 22. Prueba de aceptación 4

Prueba 5	
Caso de uso	Enviar una búsqueda
Objetivo	Probar el funcionamiento de envío de búsquedas al servidor
Precondición	La aplicación está instalada El usuario está logado
Descripción de la prueba	El usuario introduce los datos en el formulario y pulsa buscar

Resultados esperados	Se muestra un mensaje al usuario indicando que la búsqueda ha sido enviada con éxito

Tabla 23. Prueba de aceptación 5

Prueba 6	
Caso de uso	Ver Ofertas recibidas
Objetivo	Probar el funcionamiento de recepción de ofertas recibidas del servidor al acceder a la aplicación
Precondición	La aplicación está instalada El usuario está logado En la home se ha descargado alguna oferta nueva para el usuario
Descripción de la prueba	El usuario accede a la aplicación y en la home se muestra una oferta nueva que selecciona y la aplicación redirige a la lista de hoteles relacionados con dicha oferta.
Resultados esperados	Se muestra la lista de hoteles relacionados con la oferta

Tabla 24. Prueba de aceptación 6

Prueba 7	
Caso de uso	Ver todas las ofertas
Objetivo	Probar el funcionamiento de recepción de ofertas recibidas del servidor y la descarga de ofertas listas en el servidor
Precondición	La aplicación está instalada El usuario está logado La aplicación ha descargado y almacenado alguna oferta para el usuario
Descripción de la prueba	El usuario selecciona ver las ofertas y la aplicación muestra por un lado las almacenadas de otras peticiones y por otro realiza de nuevo una petición al servidor por si hay más
Resultados esperados	Se muestra la lista de ofertas, tanto las que estaban ya almacenadas como las recién descargadas

Tabla 25. Prueba de aceptación 7

Prueba 8	
Caso de uso	Ver detalle de un hotel
Objetivo	Probar el funcionamiento de la ficha de hotel
Precondición	La aplicación está instalada El usuario está logado El usuario tiene alguna oferta con hoteles almacenada
Descripción de la prueba	El usuario seleccionará una de las ofertas de la home o bien una de las almacenadas y el sistema mostrará la lista de hoteles, el usuario seleccionará uno de ellos y el sistema mostrará su ficha detalle
Resultados esperados	Se muestran los datos del hotel seleccionado

Tabla 26. Prueba de aceptación 8

Prueba 9	
Caso de uso	Ver perfil
Objetivo	Probar la obtención de datos del perfil
Precondición	La aplicación está instalada El usuario está logado
Descripción de la prueba	El usuario selecciona Mi Perfil y el sistema recupera los datos del dispositivo y los muestra por pantalla
Resultados esperados	Los datos del usuario se muestran por pantalla

Tabla 27. Prueba de aceptación 9

Prueba 10	
Caso de uso	Modificar moneda
Objetivo	Probar el funcionamiento del cambio de moneda desde el perfil
Precondición	La aplicación está instalada El usuario está logado
Descripción de la prueba	El usuario modifica la moneda en el perfil y el sistema envía al servidor la nueva moneda para las sucesivas ofertas
Resultados esperados	Se muestra un mensaje al usuario confirmando el cambio

Tabla 28. Prueba de aceptación 10

Prueba 11	
Caso de uso	Modificar datos de acceso
Objetivo	Probar el funcionamiento de la modificación de datos de acceso
Precondición	La aplicación está instalada El usuario está logado
Descripción de la prueba	El usuario modifica alguno o todos los datos de acceso (excepto el email que no es posible) y el sistema actualiza la información, tanto en el dispositivo como el servidor
Resultados esperados	Se modifican los datos

Tabla 29. Prueba de aceptación 11

Prueba 12	
Caso de uso	Ver la página de la oferta
Objetivo	Probar la redirección a una página de oferta
Precondición	La aplicación está instalada El usuario está logado. El usuario tiene alguna oferta
Descripción de la prueba	El usuario selecciona reservar alguna de sus ofertas y el sistema redirige al usuario a la página original con la oferta
Resultados esperados	Se muestra un mensaje de aviso que se va a salir de la aplicación y se muestra la página original en la pantalla

Tabla 30. Prueba de aceptación 12

6. Anexos

6.1. Anexo 1: Configuración del entorno

Desde el siguiente enlace http://developer.android.com/sdk/index.html descargamos el fichero ZIP con el ADT Bundle para Windows. Para ello se solicita aceptar el contenido de la licencia una vez elegida la plataforma de nuestro ordenador, 32-bit o 64-bit según corresponda.

Una vez descargado, descomprimimos el fichero y ejecutamos el IDE Eclipse contenido, que ya tiene preparado el plug-in con las herramientas de desarrollo Android y otros elementos de la plataforma.

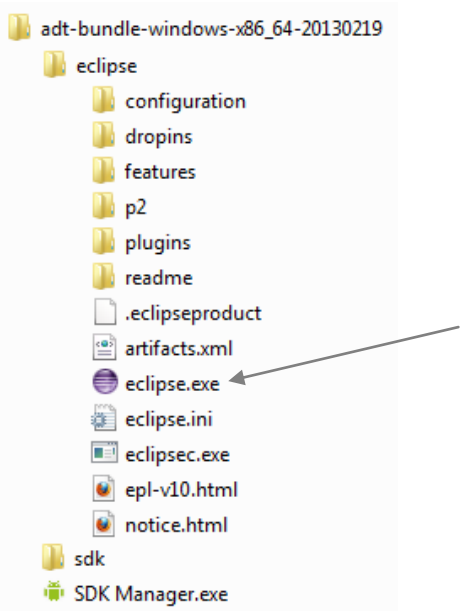


Figura 52. Estructura directorios ADT Bundle para Windows

La primera vez que se abre, el eclipse solicita el directorio del workspace, en el cual se crearán las carpetas de cada aplicación que se cree.

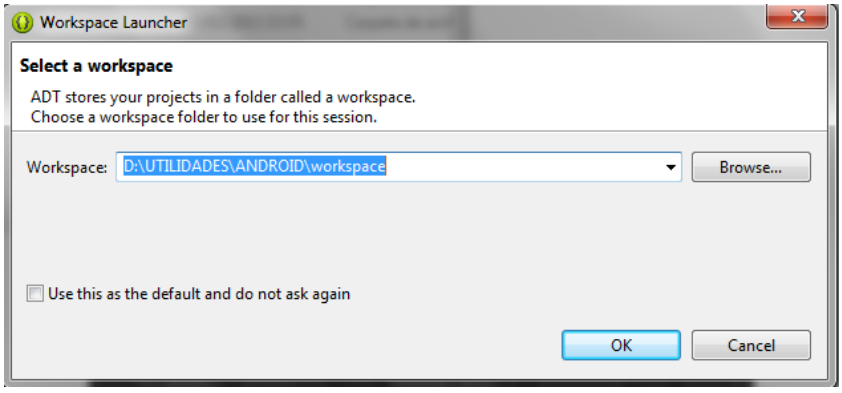


Figura 53. Eclipse – Elección del workspace

El plug-in de Android está contenido, junto a otras herramientas como *Android Virtual Device Manager,* que permite crear y simular distintos dispositivos para compilar y ejecutar nuestras aplicaciones en ellos, en lugar de tener que utilizar un dispositivo real. Esto es útil sobre todo durante el desarrollo y sobre todo para probar la aplicación en distintos dispositivos sin la necesidad de tenerlos físicamente.

Para crear una aplicación Android, bastará con añadir con el botón derecho en la pestaña "Package Explorar", *New / Android Application Project*

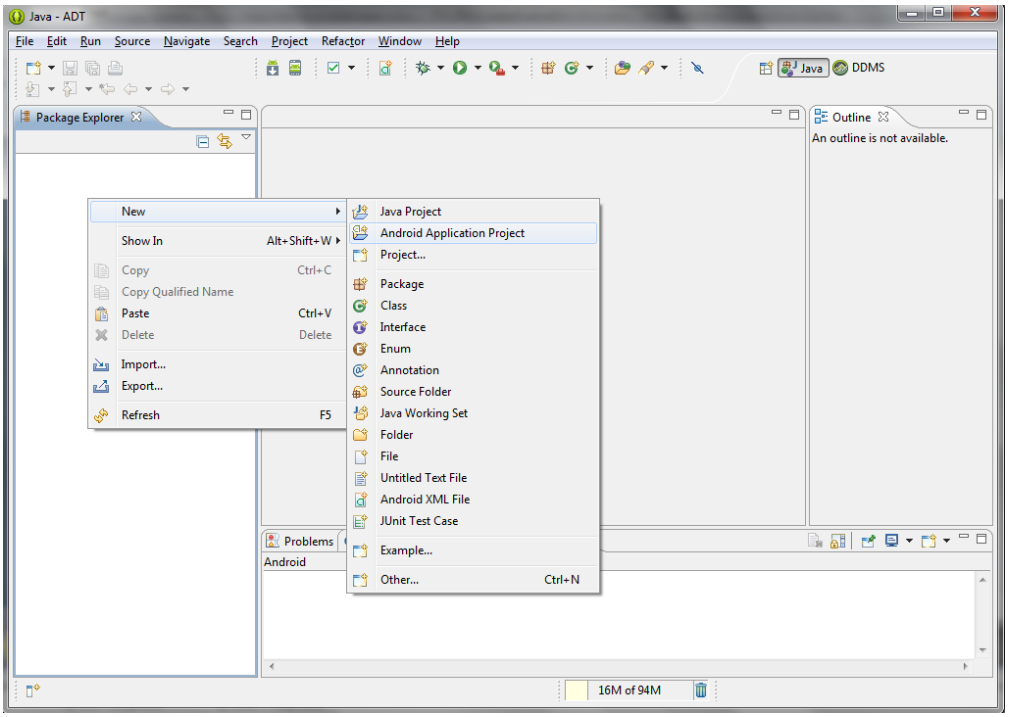


Figura 54. Eclipse – Crear nueva aplicación Android

En el siguiente pop-up se indica el nombre de la aplicación y otra serie de detalles importantes a la hora de desarrollar aplicaciones Android:

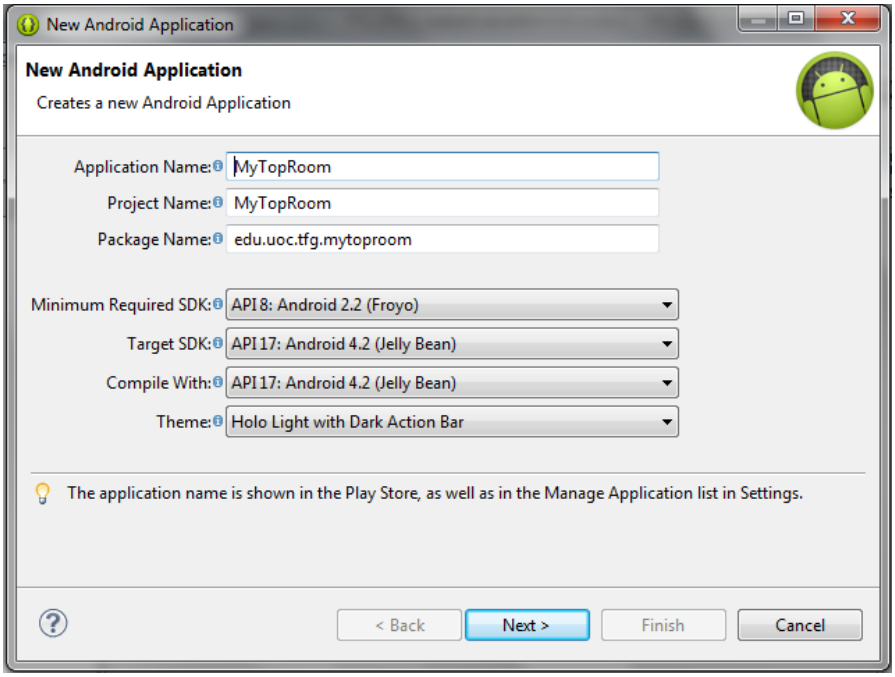


Figura 55. Eclipse – Especificaciones para una nueva aplicación

- *Application Name,* es el nombre de la aplicación que le aparecerá a los usuarios, en este caso MyTopRoom
- *Project Name,* es el nombre del directorio del proyecto y el nombre que visualiza Eclipse
- *Package Name,* es el namespace de nuestra aplicación, con las mismas reglas que se utilizan para cualquier aplicación Java, en este caso hemos empleado el dominio de la UOC añadiendo la asignatura y nombre de la aplicación.
- *Mínimo SDK requerido,* es la versión mínima de Android que soportará la aplicación. Para que sea distribuible en el mayor número de dispositivos posible, se debería indicar la versión más baja disponible para las características que vayamos a incluir. Si por ejemplo estamos incluyendo funciones que solo son soportadas por versiones recientes, la aplicación solo será ejecutable en esas versiones.

  Según un artículo publicado en enero de 2013 en el blog MAJS's (http://mjimenez.porexpertos.es/sdk-android-vs-mi-proyecto/), con información de cifras del crecimiento de Smartphones con SO Android, para llegar al mayor número de usuarios se recomienda desarrollar para versiones SDK 9 y superiores, y en caso ir dirigida tanto a Tablets como Smartphones lo recomendable es usar SDK 14 o superiores. En nuestro caso hemos dejado el API 8 que sale por defecto.

- *Target SDK,* indica la versión más alta de Android en la que hemos testeado la aplicación. Según van saliendo nuevas versiones, deberíamos testear la aplicación en ellas, para obtener el API con las últimas funciones de la plataforma. Se ha dejado la más reciente, la API 17

- *Compile with,* es la versión de la plataforma contra la que se compilará la aplicación. Por defecto, se selecciona la última versión disponible de Android.

- *Theme,* especifica el estilo de interface de usuario que se aplicará a la aplicación.

En las siguientes pantallas, el wizard para crear la aplicación móvil nos dará la opción de crear un icono para la aplicación y una primera actividad –clase Activity- que actuará de actividad principal la primera vez que se ejecuta la aplicación.

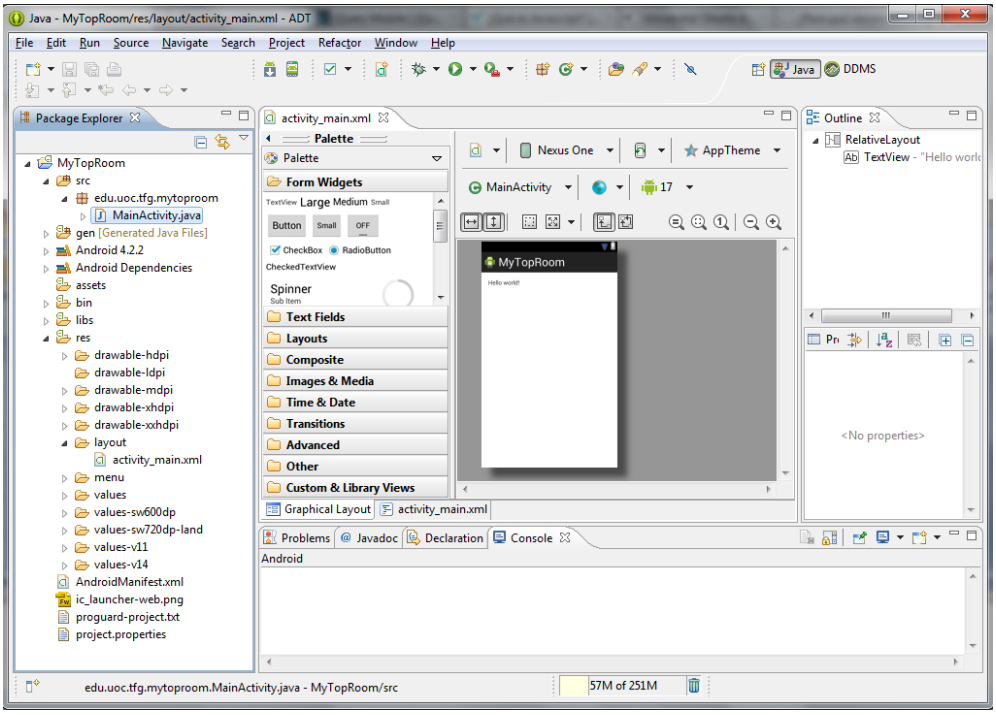


Figura 56. Eclipse – Crear la aplicación MyTopRoom

Con la primera actividad creada por defecto configuramos el dispositivo virtual donde se realizan los despliegues de la aplicación durante el desarrollo.

Pulsamos el icono  del *Android Virtual Device Manager,* pulsamos “New” e incluimos los datos para el dispositivo virtual.

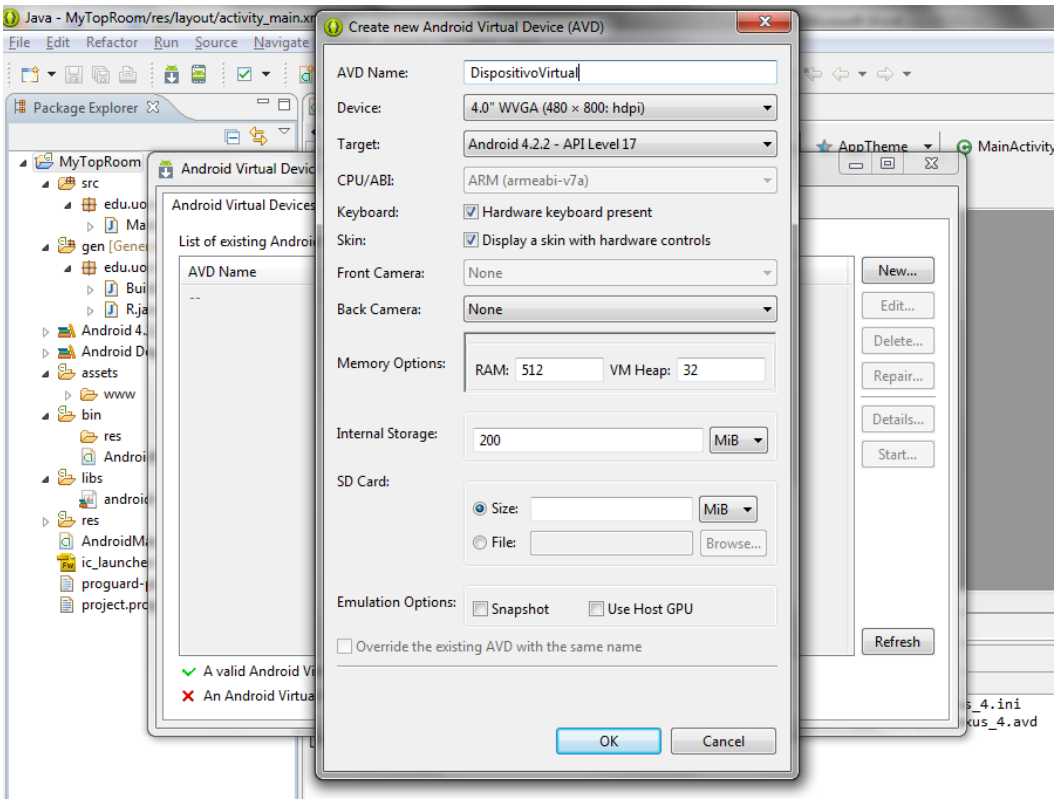


Figura 57. Eclipse – Configuración dispositivo virtual

En este caso elegimos las pulgadas y la misma resolución que tiene el Nexus 4, dispositivo real en el que se instala la aplicación. Si ejecutamos la aplicación, la aplicación se desplegará en este dispositivo virtual.

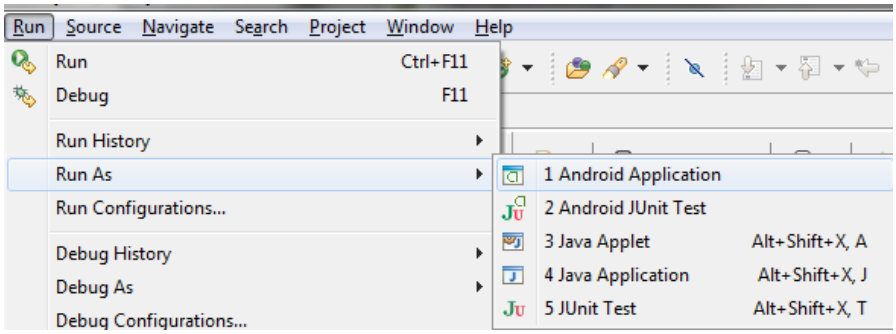


Figura 58. Eclipse – Ejecutar la aplicación Android

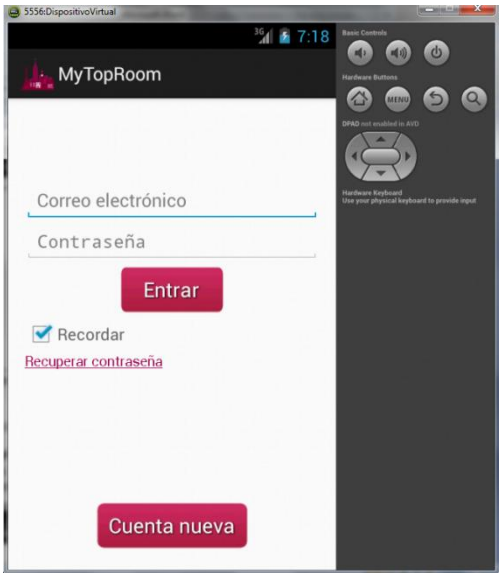


Figura 59. Eclipse – Aplicación ejecutada en dispositivo virtual

6.2. Anexo 2: Manual de instalación

Instalación en el dispositivo virtual

Dentro del fichero zip "**MyTopRoom.zip**" que contiene el proyecto Eclipse, una vez configurado el entorno como se ha definido con anterioridad, lo importamos y desplegamos en el dispositivo virtual:

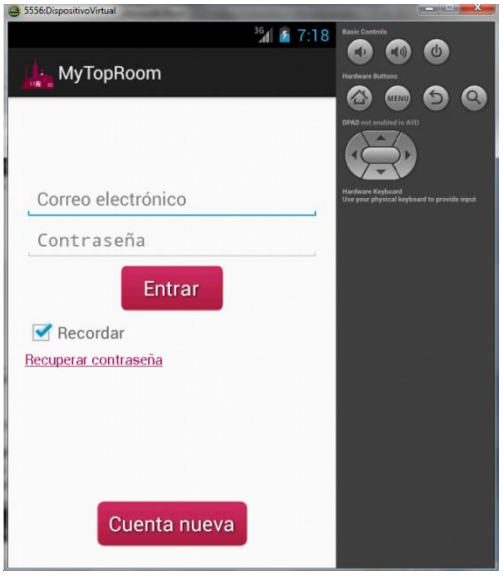


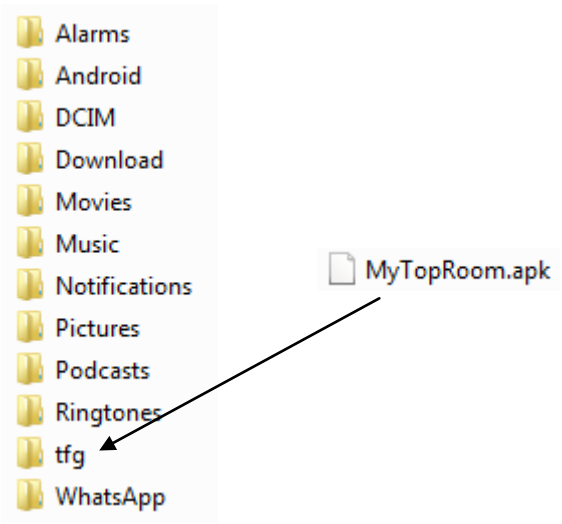
Figura 60. Instalación de aplicación Android en dispositivo

En el siguiente anexo se explican los pasos para instalar el prototipo del motor de disponibilidad en un servidor Web, para que el dispositivo pueda conectarse.

Instalación en el dispositivo físico

Dentro del fichero zip "**MyTopRoom.zip**" que contiene el proyecto Eclipse, tenemos la carpeta "bin" que es el directorio donde el IDE deja el archivo ejecutable Android "**MyTopRoom.apk**" al compilar la aplicación.

El proceso de instalación de esta aplicación es similar al método que se emplea en un pc con Windows, se copia en un directorio y con pulsarla la aplicación se instala. Para transferirla al Smartphone o tablet Android donde se quiera probar, mediante un cable USB conectado al ordenador, al mostrarse la unidad copiamos la aplicación a un directorio conocido dentro del dispositivo.



Después con cualquier aplicación para visualizar archivos y carpetas, por ejemplo File Manager, navegamos al directorio y seleccionamos la aplicación.

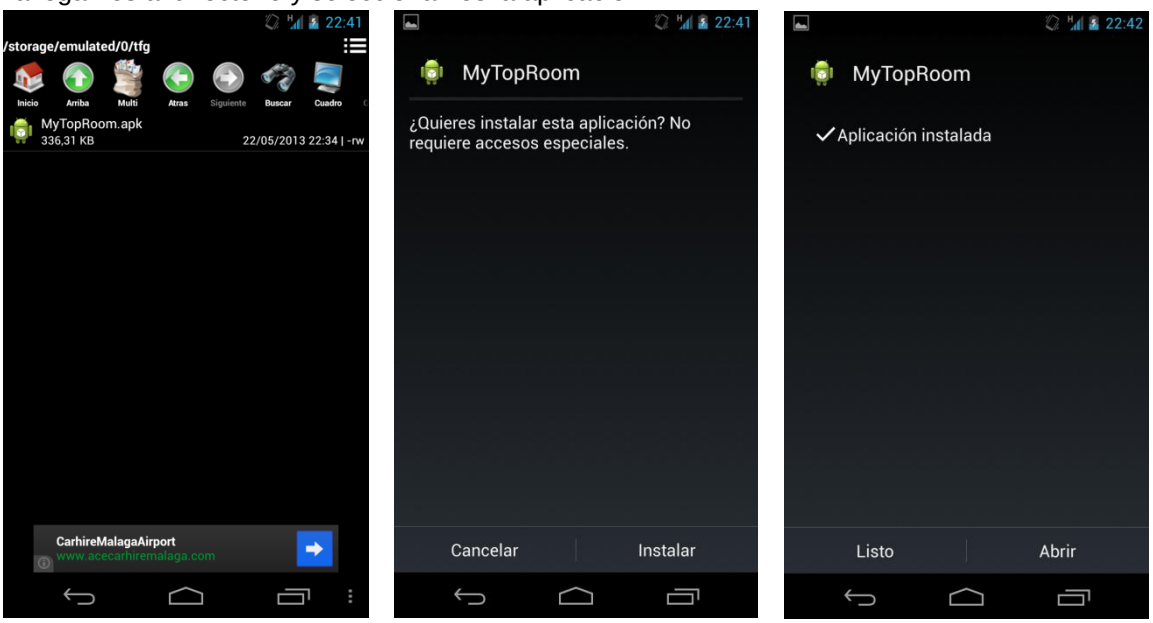


Figura 61. Instalación de aplicación Android en dispositivo

Una vez instalada se puede abrir y probar en el dispositivo, pero dado que no podrá acceder al motor simulado, solo se podrán comprobar las pantallas públicas, es decir las que pueden verse cuando el usuario no está logado. Una alternativa sería instalar el motor demo en un servidor web con dirección pública y sustituir dentro de la aplicación las direcciones URL a localhost (http://10.0.2.2/) por dicha dirección pública.

Figura 62. MyTopRoom

6.3. Anexo 3: Manual de instalación del prototipo de motor de disponibilidad

En el mismo equipo donde se vaya a desplegar el dispositivo virtual, se debe instalar un servidor Web, con PHP5 y la base de datos MySQL.

En el servidor Web, se creará la carpeta “mytoproomserver” y en ella se copiarán los archivos PHP incluidos en el ZIP “**MyTopRoom.ZIP**” en la carpeta “Servidor”, y así al ejecutar la siguiente dirección, las páginas PHP estén disponibles:

Figura 63. Página principal del motor de disponibilidad

Esta dirección es la que emplea la aplicación para conectar con el servidor remoto que actúa de prototipo de motor de disponibilidad.

Se creará la base de datos prototipo del motor de disponibilidad con el nombre de "mytoproomserver", usuario "adminmytoproom" y contraseña "password", y en ella se crearán las tablas contenidas en el fichero "servidor.sql", también dentro de la carpeta "Servidor" en el archivo comprimido.

Para comprobar que el dispositivo virtual accede al servidor remoto disponible en "localhost", podemos abrir el navegador en el dispositivo y consultar la dirección http://10.0.2.2/mytoproomserver/ Si podemos ver la página, el servidor remoto ya es accesible desde la aplicación:

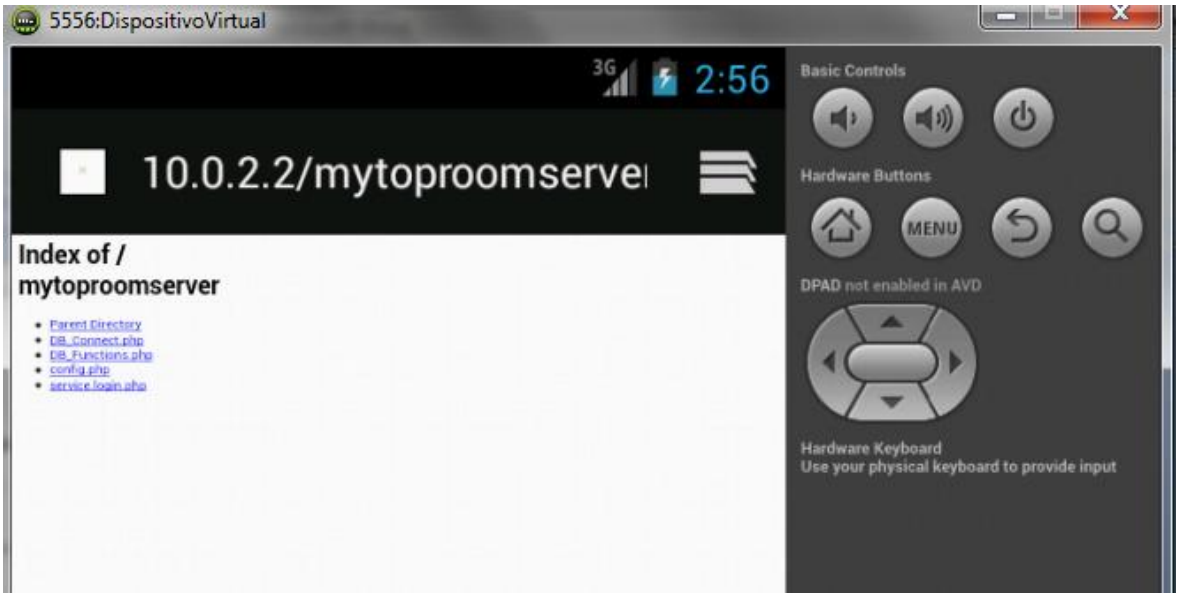


Figura 64. Comprobación de disponibilidad del servidor remoto en el dispositivo virtual

Desde el dispositivo virtual y siempre y cuando esté el servidor web disponible en la dirección “localhost” podemos ejecutar los diferentes casos de uso.

7. Bibliografía

(Albin, 2003) **Albin, Stephen T.** (2003) "The Software Product Life Cycle", "Using Architecture Frameworks" En: *The Art of Software Architecture* (1ª ed.) Canadá: Wiley Publishing, Inc.

(Beck, 1989) **Beck, K.; Ward, W.;** (1993) *A Laboratory For Teaching Object-Oriented Thinking.* Disponible en línea: http://c2.com/doc/oopsla89/paper.html#cards

(Benbourahala, 2013) **Benbourahala, N.; (2013)** En: *Android 4 – Principio del desarrollo de aplicaciones Java* (1ª ed) Barcelona: Ediciones ENI

(Booch, 2000) **Booch, G.; Rumbaugh, J.; Jacobson, I.** (2000) "La naturaleza y propósito de los modelos", "La vista estática", "La vista de casos de uso", "La vista de interacción" En: *El lenguaje unificado de modelado. Manual de referencia* (1ª ed.) Madrid: Pearson Educación, S.A..

(Cockburn, 2001) **Cockburn, Alistair.** (2001) "Introduction" En: *Writing Effective Use Cases (1ª ed.)* Canada: Editorial Addison-Wesley

(Gogolla, 2001) **Gogolla, M.; (2001)** En: *«UML»2001-The Unified Modeling Language. Modeling Languages, Concepts, and Tools* (1ª ed) Berlín: Editorial Springer

(Keynote, 2013) **Keynote.** *Testing Strategies and Tactics for Mobile Applications.* Versión en PDF disponible en: http://www.keynote.com/docs/whitepapers/WP_Testing_Strategies.pdf

(Pello, 2011) **Pello, J.** (2011) *Slash Mobile presenta "Testing en aplicaciones móviles iOS, Android"* Disponible en línea <http://www.softqanetwork.com/slash-mobile-presenta-testing-en-aplicaciones-moviles-ios-android>

(Pradel, 2011) **Pradel, J.; Raya, J.** (2011) "Introducción a la Ingeniería del software", "Requisitos", "Análisis UML" En: *Ingeniería del software* (1ª ed.) Barcelona: FUOC

(Pradel, 2012) **Pradel, J.; Raya, J.** (2012) "Introducción a la Ingeniería de requisitos", "Obtención de requisitos" En: *Ingeniería de requisitos. (1ª ed.)* Material docente de la UOC

(Pressman, 2001) **Pressman, Roger S.** (2001) "Conceptos y principios del análisis" En: *Ingeniería del software. Un enfoque práctico (5ª ed.)* Madrid: Editorial McGraw-Hill/Interamericana de España, SAU

(Ríos, 2011) **Ríos, A.; (2011)** Movibery. *Apps Nativas Vs. Web Apps. ¿Cuál elegir?* Disponible en línea <http://www.mobivery.com/es/blog/tecnologia-es/apps-nativas-vs-web-apps-cual-elegir>

(SWEBOK, 2013) **SWEBOK** *Software Testing.* Versión en PDF disponible en: https://computer.centraldesktop.com/home/viewfile?guid=36116515206BF043A33CB08E7EA5717ED2D5C8C87&id=21812001

(Vallecillo, 2013) **Vallecillo, A.; Romero, J.R.; Moreno, Nathalie; Durán, F.J (2013)** "Diseño de aplicaciones distribuidas". En: *Ingeniería del software de componentes y sistemas distribuidos* (1ª ed) Barcelona: FUOC

(Webtaller, 2011) **Webtaller.** *Usabilidad en aplicaciones para teléfonos móviles.* Disponible en línea: http://www.webtaller.com/maletin/articulos/usabilidad_en_aplicaciones_para_telefonos_moviles.php

(360logica**,** 2011) **360logica.** *Testing mobile applications.* Versión en PDF disponible en: http://www.tutorialspoint.com/white-papers/335.pdf