Aplicación Android Alquiler de Coches

Memoria Final

Alumno: David Lorenzo López

Consultor: Jordi Ceballos Villach

Agradecimientos:

A mi mujer Yoana, que siempre me ha apoyado y animado a seguir adelante. Gracias a ella he podido sacar este proyecto adelante combinando el trabajo con los estudios, cosa que no ha sido fácil. Siempre ha estado ahí cuando la he necesitado y en los momentos más difíciles.

A mi Padre, que ha hecho todo lo posible para que pueda estudiar esta carrera, y aunque no vivimos en la misma isla, no pasa un solo día sin llamarme para interesarse por mí.

A mi Madre, que siempre quiso que estudiara una carrera y me hiciera un hombre de provecho, pero no pudo quedarse en este mundo para verlo. Cada día que pasé estudiando la tuve presente, y el haber llegado hasta aquí es en gran parte gracias a ella.

A los consultores y compañeros de la UOC, que de una forma u otra me han apoyado cuando lo he necesitado y han hecho posible el resultado final.

Índice

1. Introducción

Android es el sistema operativo para dispositivos móviles más usado en el mundo, cientos de millones de dispositivos lo usan en más de 190 países, y cada día un nuevo millón de usuarios comienza a usar un dispositivo con Android.

En la imagen 1 se puede ver el gráfico de crecimiento de uso del SO de Google a lo largo de los años. Como se puede apreciar, en sus inicios fue bastante difícil su aceptación por parte de la comunidad, pero el gran trabajo de Google en sus constantes mejoras y actualizaciones, tanto de rendimiento como de diseño; y su compatibilidad con infinidad de dispositivos ha hecho que se haga el auténtico dominador del mercado actualmente con más del 80% de cuota de mercado global.

1. Crecimiento de Android

Exhibit 1: Global Smartphone OS Shipments and Market Share in Q2 2014

Global Smartphone Operating System Shipments (Millions of Units)	Q2 '13	Q2 '14
Android	186.8	249.6
Apple iOS	31.2	35.2
Microsoft	8.9	8.0
BlackBerry	5.7	1.9
Others	0.5	0.5
Total	233.0	295.2

Global Smartphone Operating System Marketshare %	Q2 '13	Q2 '14
Android	80.2%	84.6%
Apple iOS	13.4%	11.9%
Microsoft	3.8%	2.7%
BlackBerry	2.4%	0.6%
Others	0.2%	0.2%
Total	100.0%	100.0%

Total Growth Year-over-Year %	48.9%	26.7%

Source: Strategy Analytics

2. Cuotas de mercado de SO para smartphones del segundo cuarto del 2014

En esta tabla de la Imagen 2 se puede ver la cuota de mercado en el segundo cuarto de 2014, comparándolo con el segundo cuarto de 2013. Se puede apreciar un crecimiento del 4,4% con respecto al año anterior, mientras que el resto de Sistemas Operativos han experimentado un ligero descenso.

Este crecimiento de Android en estos últimos años, y su flexibilidad para crear aplicaciones que se adaptan a infinidad de dispositivos como móviles, tabletas, relojes, televisores, electrodomésticos, coches, etc. ha hecho que me decida a realizar este proyecto de creación de una aplicación de Android para profundizar en el desarrollo para este Sistema Operativo.

2. Definición General del Proyecto

En este proyecto se desarrollará una aplicación de alquiler de coches para el SO Android, en la que se podrán realizar, consultar, cancelar y modificar reservas de coches totalmente online, además de otras funcionalidades que veremos posteriormente. Con este desarrollo se investigará en el desarrollo de aplicaciones para este sistema operativo; desde su diseño y pasando por su implementación, hasta llegar a su publicación en la tienda de aplicaciones más importante de la plataforma, Google Play.

La aplicación se basará en el servicio web de disponibilidad y reservas de coches de la empresa Canarias.com, para la que actualmente trabajo como desarrollador y responsable de IT. Este servicio web pertenece al motor de reservas de coches de Canarias.com y está desarrollado en el lenguaje de programación ASP. Además, provee una interfaz XML que permite realizar todo el flujo de reserva de vehículos totalmente online, así como consultar las zonas y oficinas de reserva, y los vehículos disponibles para reservar.

Se cuidará mucho el diseño de la aplicación, siguiendo en la medida de lo posible las recomendaciones de diseño de Google para las aplicaciones de Android, que recientemente ha recibido el Premio de Oro por la mejor contribución a la experiencia de usuario en este 2014.

2.1 Objetivos

El objetivo del proyecto es profundizar en el desarrollo de aplicaciones para la plataforma Android. Para ello, se realizara el desarrollo de una aplicación de complejidad media-alta donde se pasará por las **tres fases principales del desarrollo**.

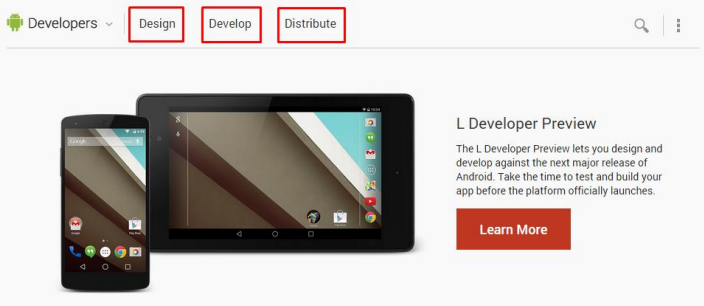
En primer lugar se tratará su **diseño y arquitectura**, preparando todos los componentes que formarán parte del sistema en sí, y diseñando las interacciones entre las diferentes pantallas, los casos de uso, el prototipo y la base de datos.

Seguidamente, se llevará a cabo la **implementación** de la aplicación. Partiendo del diseño realizado en el paso anterior, siguiendo las recomendaciones de Google para el desarrollo de

aplicaciones Android, así como usando diferentes patrones de desarrollo que ayudaran a obtener un código limpio y más fácil de mantener.

Finalmente, se llevarán a cabo las **pruebas** de funcionamiento y la **distribución** de la aplicación en Google Play, la tienda de aplicaciones más importante de Android. Este último paso, aunque pueda parecer menor, es muy importante, ya que una buena distribución y promoción en Google Play nos ayudará en mayor parte al éxito del proyecto.

Estos tres objetivos generales que hemos marcado, concuerdan con las secciones que Google indica en la web de desarrolladores de Android: http://developer.android.com



3. Secciones de desarrollo de Android

Son los tres fases que todo desarrollador de Android debe seguir para hacer más posible el éxito de su aplicación.

2.2 Funcionalidades principales

El proyecto, como se ha comentado anteriormente, consistirá en una aplicación de alquiler de coches a través de Internet, haciendo uso del servicio web XML de la empresa Canarias.com.

Este servicio web provee una interfaz con diferentes operaciones para consultar la disponibilidad de los vehículos, realizar, modificar y cancelar las reservas de los mismos, así como consultar los accesorios o extras disponibles y listar las zonas y los modelos disponibles.

La aplicación que se va a desarrollar tendrá, por tanto, las siguientes funcionalidades:

- Consultar las oficinas disponibles para reservar.
- Consultar los vehículos disponibles para reservar.
- Consultar la disponibilidad de vehículos por zona.
- Consultar la disponibilidad de vehículos por modelo.
- Reservar un coche con o sin accesorios o extras.

- Modificar una reserva.
- Cancelar una reserva.
- Listar las reservas realizadas.
- Consultar una reserva.

Además, se realizará un apartado de configuración donde se podrán definir diferentes datos de la aplicación, como la información del cliente para que se rellene automáticamente al realizar una reserva, o actualizar las zonas y modelos disponibles en la base de datos local de la aplicación. Incluirá también un apartado de ayuda donde se propondrán algunos tutoriales en formato "*slide*" que permitirán al usuario aprender a usar la aplicación y sacar el máximo partido de ella.

Se utilizarán diferentes patrones de diseño tanto para la interfaz gráfica como para la implementación del acceso a base de datos, entre otras funcionalidades.

3. Calendario del Proyecto

Para completar de manera satisfactoria el proyecto ideado, se seguirá el método de evaluación continua propuesto por la UOC. Es decir, se marcarán 4 hitos correspondientes a las 4 PEC propuestas. En las 3 primeras se hará una entrega parcial del proyecto, mientras que en la última se hará la entrega total, que incluye a su vez las 3 entregas anteriores.

Se seguirá el ciclo de vida clásico de desarrollo, siguiendo las fases de Planificación, Diseño, Desarrollo-Pruebas y Entrega. Las fechas de finalización de cada fase corresponderán con las fechas de Entrega de cada una de las PEC indicadas en el calendario de la asignatura.

3.1 Entregas establecidas

Las fechas de entrega establecidas coinciden con las fechas de las 4 PEC indicadas en el calendario, y son las siguientes:

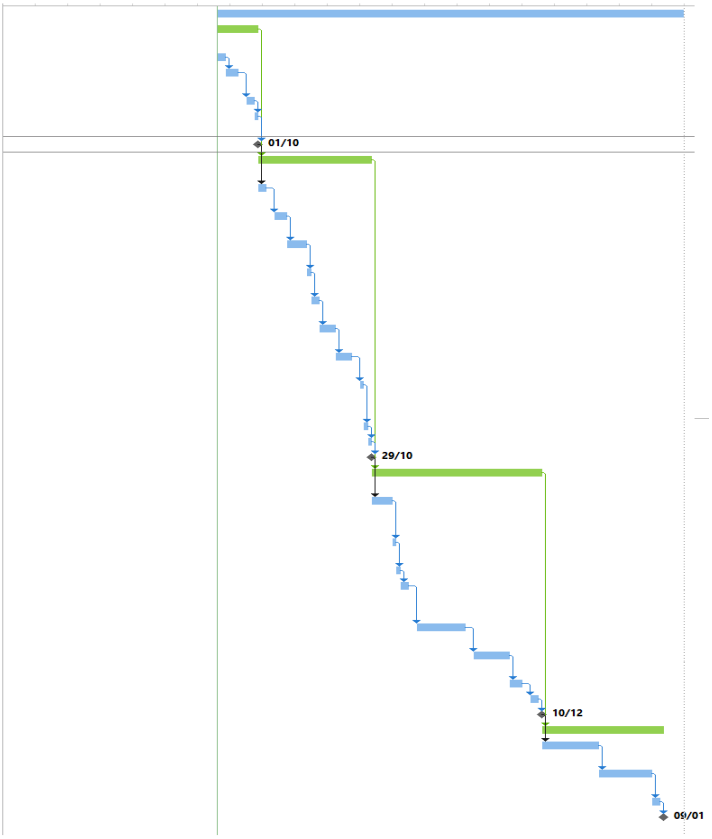
Fecha de entrega	Tarea	Entregables
1 de Octubre de 2014	PEC 1	Plan de trabajo
29 de Octubre de 2014	PEC 2	Análisis funcional, Diseño Técnico y Prototipo
10 de Diciembre de 2014	PEC 3	Implementación
9 de Enero de 2015	Entrega Final	Memoria y vídeo con la presentación del proyecto

3.2 Calendario

La fecha inicial del proyecto será el 17 de Septiembre de 2014 y la fecha final será el 9 de Enero de 2015, en la que realizará la entrega completa del proyecto.

Se ha elaborado el plan de proyecto y a continuación se muestra el diagrama de Gantt con las tareas a realizar para completar el proyecto:

1	TFC - ANDROID APP	83 días	lun 22/09/14	mié 14/01/1	
2	**PEC 1 - Plan de trabajo**	**8 días**	**lun 22/09/14**	**mié 01/10/14**	
3	Sesiones de estudio	2 días	lun 22/09/14	mar 23/09/1	
4	Preparación Plan Trabajo	3 días	mié 24/09/14	vie 26/09/14	3
5	Redacción PEC 1	2 días	lun 29/09/14	mar 30/09/1	4
6	Revisión PEC 1	1 día	mié 01/10/14	mié 01/10/14	5
7	Entrega PEC 1	0 días	mié 01/10/1	mié 01/10/1	6
8	**PEC 2 - Análisis Funcional y Diseño**	**20 días**	**jue 02/10/14**	**mié 29/10/14**	**2**
9	Recopilación de funcionalidades	2 días	jue 02/10/14	vie 03/10/14	7
10	Recopilación de casos de uso	3 días	lun 06/10/14	mié 08/10/14	9
11	Creación diagrama casos de uso	3 días	jue 09/10/14	lun 13/10/14	1
12	Creación diagrama estático	1 día	mar 14/10/14	mar 14/10/14	1
13	Creación diagrama arquitectura	2 días	mié 15/10/14	jue 16/10/14	1
14	Creación diseño BBDD	2 días	vie 17/10/14	lun 20/10/14	1
15	Creación prototipo de App	4 días	mar 21/10/14	vie 24/10/14	1
16	Recopilación información para PEC 2	1 día	lun 27/10/14	lun 27/10/14	1
17	Redacción PEC 2	1 día	mar 28/10/1	mar 28/10/14	
18	Revisión PEC 2	1 día	mié 29/10/1	mié 29/10/14	
19	Entrega PEC 2	0 días	mié 29/10/1	mié 29/10/14	
20	**PEC 3 - Implementación**	**30 días**	**jue 30/10/14**	**mié 10/12/14**	
21	Módulo de conexión con WebService de Canarias.com	3 días	jue 30/10/14	lun 03/11/14	
22	Creación Clases del Modelo de datos	1 día	mar 04/11/14	mar 04/11/14	
23	Creación BBDD	1 día	mié 05/11/1	mié 05/11/14	
24	Creación interfaz DAO de conexión con BBDD	2 días	jue 06/11/14	vie 07/11/14	
25	Desarrollo de la aplicación	10 días	lun 10/11/14	vie 21/11/14	
26	Desarrollo interfaz gráfica	7 días	lun 24/11/14	mar 02/12/14	
27	Pruebas	3 días	mié 03/12/1	vie 05/12/14	
28	Empaquetación	2 días	lun 08/12/14	mar 09/12/14	
29	Entrega PEC 3	1 día	mié 10/12/1	mié 10/12/14	
30	**Entrega Final**	**22 días**	**jue 11/12/14**	**vie 09/01/15**	
31	Preparación Memoria Final	10 días	jue 11/12/14	mié 24/12/14	
32	Preparación Video Explicativo	9 días	jue 25/12/14	mar 06/01/15	
33	Revisión General	2 días	mié 07/01/1	jue 08/01/15	
34	Entrega final TFG	1 día	vie 09/01/15	vie 09/01/15	



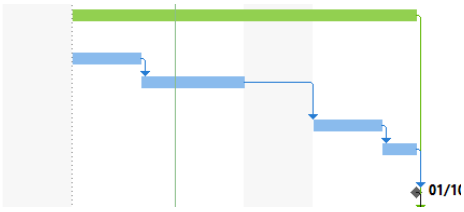
El tiempo total para realizar el proyecto son 81 días, dejando aparte los fines de semana, que se tomarán en principio como descanso. Esto permitirá tener días de sobra por si surge algún imprevisto que haga tener que replantear el plan de proyecto.

En el diagrama se ha establecido una tarea completa que engloba todo el proyecto. Además, se han establecido 4 hitos correspondientes a las 4 entregas establecidas, que a su vez corresponden con las 4 PEC, y que son dependientes unas de otras.

Dentro de cada hito, se ha dividido la planificación en tareas sucesivas, que se irán desarrollando una a una hasta completar el hito. Estas tareas más pequeñas permiten desglosar las entregas y centrarse en todos los puntos importantes que permitan completar la entrega con éxito.

3.3 Detalle de la entrega de la PEC 1

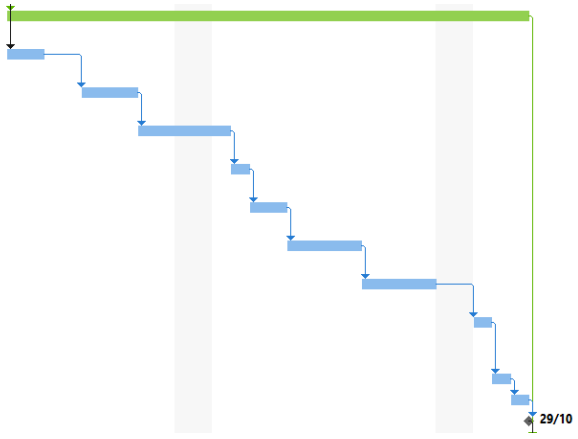
2		PEC 1 - Plan de trabajo	8 días	lun 22/09/14	mié 01/10/14
3		Sesiones de estudio	2 días	lun 22/09/14	mar 23/09/1
4		Preparación Plan Trabajo	3 días	mié 24/09/14	vie 26/09/14 3
5		Redacción PEC 1	2 días	lun 29/09/14	mar 30/09/1 4
6		Revisión PEC 1	1 día	mié 01/10/14	mié 01/10/14 5
7		Entrega PEC 1	0 días	mié 01/10/1	mié 01/10/1 6



Identificador tarea	Nombre tarea	Descripción tarea
3	Sesiones de estudio	Estudio de los materiales de apoyo de la asignatura.
4	Preparación Plan Trabajo	Definición de los objetivos y funcionalidades del proyecto.
5	Redacción PEC 1	Redacción del documento de entrega de la PEC 1.
6	Revisión PEC 1	Revisión del documento de entrega de la PEC 1.
7	Entrega PEC 1	Entrega de la PEC 1 en el REC.

3.4 Detalle de la entrega de la PEC 2

8		PEC 2 - Análisis Funcional y Diseño	20 días	jue 02/10/14	mié 29/10/14 2
9		Recopilación de funcionalidades	2 días	jue 02/10/14	vie 03/10/14 7
10		Recopilación de casos de uso	3 días	lun 06/10/14	mié 08/10/14 9
11		Creación diagrama casos de uso	3 días	jue 09/10/14	lun 13/10/14 1
12		Creación diagrama estático	1 día	mar 14/10/14	mar 14/10/14 1
13		Creación diagrama arquitectura	2 días	mié 15/10/14	jue 16/10/14 1
14		Creación diseño BBDD	2 días	vie 17/10/14	lun 20/10/14 1
15		Creación prototipo de App	4 días	mar 21/10/14	vie 24/10/14 1
16		Recopilación información para PEC 2	1 día	lun 27/10/14	lun 27/10/14 1
17		Redacción PEC 2	1 día	mar 28/10/1	mar 28/10/1 1
18		Revisión PEC 2	1 día	mié 29/10/1	mié 29/10/1 1
19		Entrega PEC 2	0 días	mié 29/10/1	mié 29/10/1 1

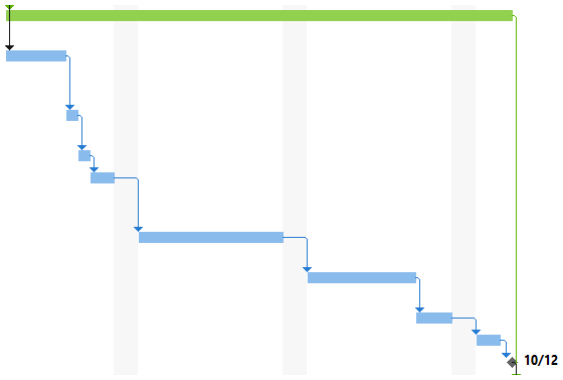


Identificador tarea	Nombre tarea	Descripción tarea
9	Recopilación de funcionalidades	Definición más precisa de las funcionalidades que tendrá la aplicación.
10	Recopilación de casos de uso	Recopilación de los casos de uso que se podrán llevar a cabo en la aplicación.

11	Creación diagrama casos de uso	Creación del diagrama de casos de uso así como la descripción formal de cada uno.
12	Creación diagrama estático	Creación del diagrama estático de clases.
13	Creación diagrama arquitectura	Creación del diagrama de arquitectura de la aplicación.
14	Creación diseño BBDD	Creación del diseño de la base de datos.
15	Creación prototipo de App	Creación del prototipo de la interfaz gráfica de la aplicación.
16	Recopilación información para PEC 2	Recopilación y ordenación de todos los documentos creados en las tareas anteriores.
17	Redacción PEC 2	Confección del documento de entrega de la PEC 2, que contendrá los diagramas creados en las tareas anteriores, además de sus correspondientes explicaciones.
18	Revisión PEC 2	Revisión del documento de entrega de la PEC 2.
19	Entrega PEC 2	Entrega de la PEC 2

3.5 Detalle de la entrega de la PEC 3

20	PEC 3 - Implementación	30 días	jue 30/10/14	mié 10/12/14	8
21	Módulo de conexión con WebService de Canarias.com	3 días	jue 30/10/14	lun 03/11/14	19
22	Creación Clases del Modelo de datos	1 día	mar 04/11/14	mar 04/11/14	21
23	Creación BBDD	1 día	mié 05/11/14	mié 05/11/14	22
24	Creación interfaz DAO de conexión con BBDD	2 días	jue 06/11/14	vie 07/11/14	23
25	Desarrollo de la aplicación	10 días	lun 10/11/14	vie 21/11/14	24
26	Desarrollo interfaz gráfica	7 días	lun 24/11/14	mar 02/12/14	25
27	Pruebas	3 días	mié 03/12/14	vie 05/12/14	26
28	Empaquetación	2 días	lun 08/12/14	mar 09/12/14	27
29	Entrega PEC 3	1 día	mié 10/12/14	mié 10/12/14	28

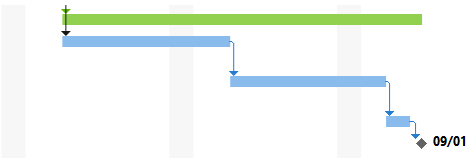


Identificador tarea	Nombre tarea	Descripción tarea
21	Módulo de conexión con Web Service de Canarias.com	Creación del módulo de conexión con el Web Service de Canarias.com
22	Creación Clases del Modelo de datos	Creación de las clases del modelo de dominio.
23	Creación BBDD	Creación de la base de datos SQLite que usará la aplicación.
24	Creación interfaz DAO de conexión con BBDD	Creación de los objetos de conexión a la base de datos siguiendo el patrón DAO (Data Access Object).

25	Desarrollo de la aplicación	Desarrollo de la aplicación Android, que estará compuesta por los módulos creados anteriormente junto a las partes propias de una aplicación Android: Activities, Layouts, Settings, Themes, etc.
26	Desarrollo interfaz gráfica	Desarrollo y perfeccionamiento de la interfaz gráfica de la aplicación, siguiendo en la medida de lo posible las recomendaciones de diseño de Google.
27	Pruebas	Batería de pruebas para comprobar todos los casos de uso y corregir errores si fuera necesario.
28	Empaquetación	Empaquetación de la aplicación para obtener los entregables.
29	Entrega PEC 3	Entrega del software de la aplicación: código y ejecutables.

3.6 Detalle de la entrega final

30		Entrega Final	22 días	jue 11/12/14	vie 09/01/15
31		Preparación Memoria Final	10 días	jue 11/12/14	mié 24/12/14
32		Preparación Video Explicativo	9 días	jue 25/12/14	mar 06/01/15
33		Revisión General	2 días	mié 07/01/1	jue 08/01/15
34		Entrega final TFG	1 día	vie 09/01/15	vie 09/01/15



Identificador tarea	Nombre tarea	Descripción tarea
31	Preparación Memoria Final	Redacción y revisión de la memoria completa del TFG.
32	Preparación Video Explicativo	Grabación y edición del vídeo de presentación del TFG.
33	Revisión General	Revisión completa de todos los entregables de esta fase: Memoria final y vídeo de presentación.
34	Entrega final TFG	Entrega final del proyecto.

4. Recursos e infraestructura

En este proyecto se utilizarán diferentes recursos para conseguir los objetivos propuestos. Toda la infraestructura usada es propia o se posee la licencia para usarla, ya sea por ser software libre o por disponer de las licencias por parte de la UOC o terceros.

Los recursos e infraestructura usada se pueden dividir en dos grupos, que se detallan a continuación.

4.1 Recursos de Hardware

Para desarrollar la aplicación y confeccionar los entregables se usará un ordenador portátil Dell Studio 1555, mientras que para realizar las pruebas de la aplicación se dispondrá de dos dispositivos móviles y un Tablet.

A continuación se detallan las características técnicas de cada dispositivo.

Componente	Características técnicas	Función
Ordenador de desarrollo	Dell Studio 1555. 1. Procesador Intel Core 2 Dúo P8600 2.4 GHz 2. Memoria 4096 MB, DDR2 PC2-6400, 2x2048MB, max. 4GB 3. Windows 8.0 64 bits	Desarrollo de la aplicación. Confección de documentos de entrega. Tareas de diseño. Creación y edición de vídeo.
Móvil para pruebas	LG Nexus 5 4. Procesador quad-core Qualcomm Snapdragon 800 2.26 GHz, GPU Adreno 330 5. Memoria 2GB RAM 6. Android OS, v4.4.4 KitKat	Realización de las pruebas al desarrollar la aplicación. Se usará este dispositivo como principal en lugar del simulador.
Móvil para pruebas	Samsung Galaxy S4 7. Procesador Exynos 5 Octa octa-core 1.6 GHz / Qualcomm Snapdragon 600 1.9GHz (según región), GPU PowerVR SGX 544MP / Adreno 320 8. Memoria 2GB RAM 9. Android OS, v4.3 Jelly Bean	Segundo móvil para realizar las pruebas de la aplicación. Al tener dos dispositivos se pueden tener más puntos de vista y afinar más el diseño.
Tablet para pruebas	ACER Iconia Tab A500 10. Procesador NVIDIA Tegra 2 dual-core 1GHz 11. Memoria 1GB RAM DDR2 12. Android OS, v4.0 ICS	Tablet para realizar las pruebas de la aplicación. Con este dispositivo se podrán realizar las pruebas de las interfaces gráficas para tabletas creadas en la aplicación.

4.2 Recursos Software

Para desarrollar el proyecto con éxito, se usarán distintos componentes y aplicaciones software, los cuales se detallan a continuación.

Componente	Función
Android Studio IDE	Entorno de desarrollo integrado proporcionado por Google para el desarrollo de aplicaciones para Android OS.

Microsoft Office 2013	Desarrollo de documentos de entrega y documentación.
Microsoft Project 2013	Desarrollo del plan de proyecto.
Google Drive	Copias de seguridad de documentos y documentación del TFG.
Git, GitHub	Repositorio en la nube para llevar el seguimiento del desarrollo del software.
Camtasia Studio	Grabación del vídeo de presentación del TFG.
Magic Draw	Creación de diagramas para el diseño y arquitectura de la aplicación.
Adobe Reader	Lectura de documentación en PDF.
Google Chrome	Búsqueda de información en Internet.
Adobe Photoshop CC	Edición y retoque gráfico.
Evolus Pencil 2.0.5	Creación del prototipo de la interfaz de la aplicación.
Adobe Premiere Pro CC	Edición del vídeo de presentación del TFG.

5. Tecnologías implicadas

En primer lugar está Java, el principal lenguaje de desarrollo para Android, en el que se codificará toda la base de la aplicación, utilizando algunas de las librerías de Java incluidas en Android, así como las propias de la plataforma.

Otra tecnología implicada en el desarrollo para Android es XML, puesto que todas las interfaces, textos, temas y configuraciones, entre otras muchas cosas, se especifican usando este lenguaje.

Por otro lado, se dispondrá de terminales con distintas versiones del sistema operativo Android instalado, con lo que se podrá comprobar el correcto funcionamiento y la apropiada visualización de la aplicación en las distintas versiones del SO.

Finalmente, se utilizarán también repositorios y almacenamiento en la nube para tareas de copias de seguridad y control de versiones del código fuente del proyecto, así como de la documentación y entregables. En este caso, el elegido es Git en combinación con GitHub. El primero nos permitirá generar y administrar nuestro repositorio local, mientras que el segundo nos permitirá sincronizar ese repositorio local con la web, para que este sea accesible a través del navegador y poderlo compartir con otros usuarios.

6. Riesgos del proyecto

Para garantizar el éxito del proyecto, será necesario anticipar los posibles riesgos que puedan suceder durante su desarrollo, y que impidan su exitoso final. Estos riesgos pueden ser generados por situaciones propias del proyecto, o por situaciones externas a él, como pueden ser situaciones familiares, de trabajo o de otra índole.

Por tanto habrá que establecer una relación de posibles riesgos que puedan suceder, una probabilidad estimada de aparición, el impacto que tendría la presencia de ese riesgo en el proyecto, así como una o varias acciones mitigadoras que se tomarían en caso de aparición del riesgo o para su prevención. A continuación se muestra una tabla con esta relación de riesgos, ordenados por el nivel de impacto que tendrían para el proyecto.

Riesgo	Descripción	Probabilidad de aparición	Impacto en el proyecto	Acciones mitigadoras
Fallo de Hardware	Avería del ordenador de desarrollo con la consiguiente posible pérdida de datos.	Media	Crítico	Realizar copias de seguridad diarias de los archivos del proyecto. Disponer de un segundo equipo con el entorno de desarrollo preparado para usarlo en caso de fallo del primero.
Planificación incorrecta	Errores en la estimación de las tareas que componen el plan de trabajo.	Media	Alto	Buscar información sobre proyectos similares y repasar el plan de trabajo. Dejar unos días de más para ser flexible a la hora de tener que modificar la planificación del proyecto.
Falta de conocimientos	Alguna de las funcionalidades de la aplicación podrían requerir el uso de componentes que no se han usado antes, y su aprendizaje puede suponer más tiempo del esperado.	Media	Alto	Tratar de esquematizar cómo se desarrollará cada funcionalidad, y qué componentes harán falta para su desarrollo. Si algún componente no se ha usado nunca, tenerlo en cuenta en la planificación de esa tarea.
Problemas de trabajo	El trabajo puede requerir en algún momento más tiempo del natural, y eso pueden ser días menos de desarrollo.	Baja	Medio	Intentar adelantar tareas siempre que se pueda, para tener días de más por si surge algún problema. Dejar aun así, si es posible, días de margen en la planificación para recomponerla en caso de problema en el trabajo.
Enfermedades	Puede suceder que algún familiar cercano o yo mismo nos	Media	Medio	Acudir a las revisiones médicas pertinentes. Seguir una vida saludable.

	viésemos afectados por una enfermedad.			Replantear la planificación del proyecto según se pueda.
Avería de los dispositivos de pruebas	Avería de los móviles usados para las pruebas o del Tablet.	Media	Bajo	Usar el simulador de Android proporcionado por Google en el Android SDK.

7. Documentación y análisis

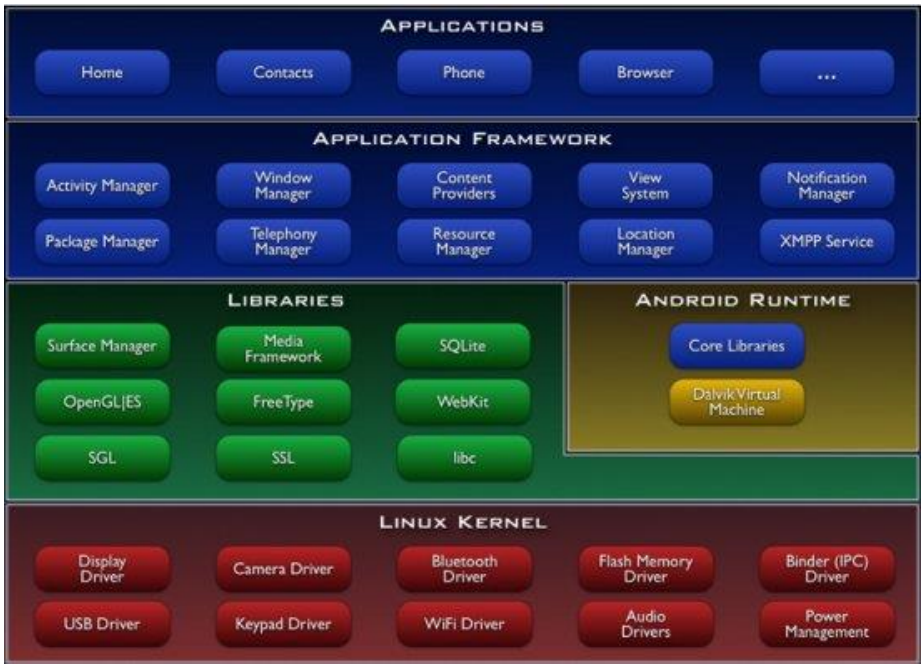
En este apartado se realizará un análisis del proyecto, desde los requerimientos funcionales y no funcionales, pasando por la descripción de los casos de uso y finalizando con el diseño técnico y el prototipo. Este análisis permitirá hacerse una idea más precisa del alcance del sistema, así como de su funcionamiento interno y externo.

Comenzamos destacando los aspectos generales de la arquitectura del sistema operativo Android, que nos permitirán tener una base para entender mejor el funcionamiento de la aplicación.

7.1 Arquitectura de Android

El Sistema Operativo Android está compuesto de una pila de software que contiene aplicaciones, sistema operativo, entorno de ejecución, software intermediario, librerías y servicios. Cada capa de la arquitectura está compuesta de múltiples componentes, y provee diferentes servicios a la capa situada justo encima en la pila de software.

A continuación se muestra un esquema de la arquitectura del SO Android, así como una breve explicación de cada capa.



4. Pila de Software del SO Android

- **Kernel Linux:** Provee un nivel de abstracción entre el hardware y las capas superiores de la pila de software. Está basado en la versión 2.6 del Kernel Linux y provee multitarea preventiva, servicios de sistema de bajo nivel como memoria, procesos y administración de energía; así como la pila de red y controladores del dispositivo como Wi-Fi, audio y pantalla.
- **Entorno de ejecución:** Provee un entorno de ejecución multitarea permitiendo a numerosos procesos ejecutarse de forma concurrente. Cada aplicación del sistema corre sobre una instancia de la Máquina Virtual Dalvik, la cual provee una capa de aislamiento entre las distintas aplicaciones y el sistema operativo. Esto permite que las aplicaciones no interfieran entre ellas, además provee una interfaz de acceso al hardware que las hace independientes del dispositivo que se use bajo el sistema operativo.

  El entorno de ejecución de Android provee numerosas librerías que pueden ser usadas por los desarrolladores y por otras partes del sistema para realizar distintas funciones. Entre ellas se pueden destacar las librerías específicas de la Máquina Virtual Dalvik, librerías de Java, librerías de Android y de C/C++, que son usadas para gestionar tareas a más bajo nivel.
- **Framework de aplicación:** Un conjunto de servicios que forman el entorno en el que corren las aplicaciones de Android. Este framework implementa el concepto de que las aplicaciones se construyen basándose en componentes que son reusables,

intercambiables y remplazables. Cada aplicación puede publicar sus capacidades junto con datos para que estos puedan ser reusados por otras aplicaciones.
Entre los principales servicios se encuentran el *Activity Manager*, que maneja el ciclo de vida de las aplicaciones; *Content Providers*, que permiten a las aplicaciones publicar y compartir datos con otras aplicaciones; *View System*, que permite crear interfaces gráficas, y *Notification Manager*, que permite gestionar las alertas y notificaciones al usuario por parte de las aplicaciones.

- **Aplicaciones:** Están localizadas en la parte superior de la pila de software de Android. Comprenden las aplicaciones nativas que vienen con el sistema operativo, como puede ser el Navegador o la aplicación de Email, y las aplicaciones de terceros que el usuario instala posteriormente en el sistema.

7.2 Requerimientos Funcionales

La aplicación que se va a desarrollar, permitirá interactuar con el servicio web de Canarias.com para realizar reservas de vehículos totalmente online. En un principio, únicamente disponibles en la isla de Tenerife, ya que es ahí donde Canarias.com posee su flota propia de vehículos; pero la aplicación se dejará abierta para, en un futuro próximo, incorporar nuevos proveedores de alquiler de coches que permitan realizar reservas de vehículos en todo el mundo.

7.2.1 Descripción básica del funcionamiento

Las funcionalidades que poseerá la aplicación derivan de las posibilidades que nos ofrece el servicio web contra el que vamos a integrar, y que se detallan a continuación:

- **Consultar las oficinas disponibles para reservar:** Se realiza una petición al servicio web para obtener un listado de las oficinas disponibles para reservar vehículos, así como su información detallada como la dirección, ciudad, coordenadas o instrucciones de recogida y devolución de vehículos.
- **Consultar los modelos de coches disponibles para reservar:** Se realiza una petición al servicio web para obtener un listado de los vehículos disponibles para reservar. Con esta petición se obtienen todos los detalles de cada vehículo, como sus características, código SIPP, imágenes, etc.
- **Consultar la disponibilidad de vehículos por zona:** Se realiza una petición al servicio web pasando los parámetros de búsqueda (fechas y oficinas de recogida y devolución) y se obtiene un listado de modelos disponibles para reservar con el precio de cada uno.
- **Consultar la disponibilidad de vehículos por modelo:** Esta funcionalidad no la permite realizar directamente el servicio web, lo que se hará es lanzar una petición por

zona, y filtrar el modelo seleccionado previamente por el usuario para mostrarle ese únicamente.

- **Reservar un coche con o sin extras.** Se permite la reserva de un vehículo en unas fechas determinadas, con la posibilidad de añadir extras o accesorios a la reserva (conductor adicional, silla de bebé, elevador, GPS, etc.).
- **Modificar una reserva.** Una vez hecha la reserva, se permitirá realizar ciertas modificaciones sobre ella, como cambiar algún dato del titular (excepto el e-mail), y añadir o eliminar algún extra o accesorio.
- **Cancelar una reserva.** En cualquier momento antes del día de la entrada, se permitirá cancelar la reserva sin coste alguno para el cliente.
- **Listar las reservas realizadas.** Se permitirá visualizar las reservas realizadas desde la aplicación móvil, para poder realizar acciones sobre ellas (visualizar, modificar o cancelar).
- **Consultar una reserva.** Se permitirá ver el detalle de una reserva.
- **Listado de oficinas.** Se podrá consultar el listado de oficinas disponibles en forma de listado y mapa, así como acceder a la ficha de cada una para conocer sus detalles y localización.
- **Listado de vehículos.** Se podrá consultar el listado de vehículos disponibles, así como acceder al detalle de cada uno para ver sus características.

Todas estas funcionalidades se podrán realizar directamente desde la aplicación móvil, lo cual beneficia al cliente que podrá reservar su vehículo desde cualquier lugar a través de su smartphone o tablet.

7.2.2 Seguridad

La seguridad debe estar presente en cualquier tipo de aplicación informática, ya sea web, móvil o de cualquier otro tipo. A continuación se especifican algunos mecanismos de seguridad que se usarán en el desarrollo para garantizar la confidencialidad e integridad de los datos.

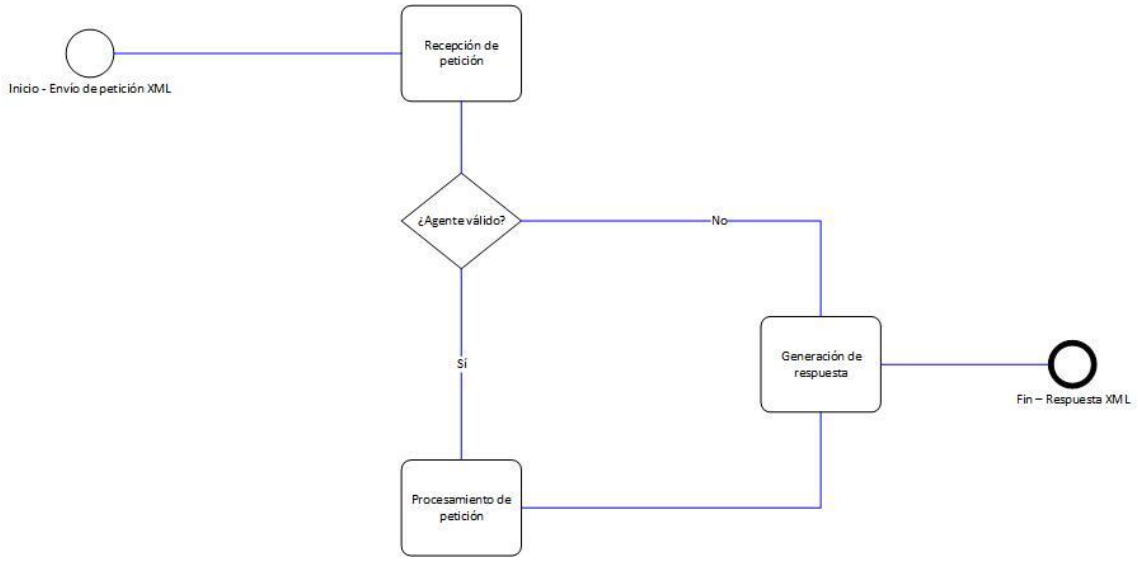
Sistema de autenticación

En este caso, se dispone de un sistema de autenticación de peticiones al servicio web, que funciona de la siguiente manera:

1. En el sistema web se crea un “Agente de Ventas”. Este agente dispone de un código y una contraseña para realizar ventas a través del servicio web de Canarias.com.
2. El código y la contraseña del agente de ventas se usarán en cada petición que se realice al servicio web.

3. Para cada petición recibida, el servicio web comprueba que el código y contraseña recibidos pertenecen a un agente de ventas válido.
4. Si la autenticación es correcta, se procesa la petición y se emite la respuesta; en caso contrario, se devuelve un error de autenticación al cliente.

A continuación se muestra un diagrama de flujo que simula una petición al servicio web.



5. Petición HTTP

Como se puede observar, para cada petición que se recibe, se realiza en primer lugar la autenticación del usuario y contraseña del agente de ventas, si esta se realiza correctamente, se procesa la petición y se genera la respuesta XML; en caso contrario, se genera directamente la respuesta XML con un mensaje de error.

Hay que indicar que el agente de ventas es único para la aplicación móvil, no es individual por cada usuario que use la aplicación, es decir, al instalar la aplicación ya vendrá pre-configurado (y sin posibilidad de cambiar) el código y contraseña del agente de ventas, por tanto el usuario final no tendrá que configurar esta información en ningún momento.

Protocolo SSL

Se usará el protocolo SSL para la comunicación con el servicio web de Canarias.com. Este protocolo proporciona autenticación y privacidad entre ambos extremos de una conexión TCP, lo cual nos garantiza que los datos no podrán ser leídos ni alterados durante su camino entre ambos extremos de la conexión.

Al usar peticiones HTTP para comunicar con el servicio web, y puesto que HTTP trabaja sobre TCP, se puede hacer uso del protocolo SSL para garantizar la privacidad de nuestras conexiones.

7.3 Requerimientos no funcionales

A continuación se detallan los requerimientos no funcionales de la aplicación, es decir, aquellos detalles que corresponden a aspectos que no describen la información a guardar ni las funciones a realizar, sino que pueden usarse para juzgar la operación del sistema en lugar de comportamientos específicos.

7.3.1 Disponibilidad

La disponibilidad de la aplicación se basa en el servicio web de Canarias.com. Siempre y cuando este servicio se encuentre disponible y en funcionamiento, desde la aplicación se podrán realizar reservas de vehículos. Ya que su funcionamiento se basa en el flujo de peticiones-respuestas entre la aplicación y el servicio web.

Cualquier caída del servicio web implica que desde la aplicación no se podrán hacer reservas. Únicamente se podrían realizar las operaciones que no requieran uso del servicio web, como por ejemplo consultar las oficinas o vehículos disponibles, así como las reservas realizadas desde la propia aplicación.

7.3.2 Interfaz gráfica

Al ser una aplicación para dispositivos móviles que pueden variar de tamaño y resolución de pantalla, se cuidará mucho el diseño de la aplicación, siguiendo en lo posible las recomendaciones de Google en sus nuevas directrices de diseño: Material Design[1].

Para conseguir una rápida adaptación del usuario a la aplicación, se seguirán una serie de patrones comunes a las principales aplicaciones Android, lo cual reducirá la curva de aprendizaje necesaria para que el usuario maneje con soltura la aplicación.

Debido a la gran cantidad de dispositivos en los que se usa Android hoy en día: móviles, tabletas, televisores, relojes, coches, electrodomésticos, etc. es imposible, dado el tiempo para realizar el proyecto, adaptarlo a todos estos dispositivos. Es por ello que se focalizarán los esfuerzos que adaptarlo a dispositivos móviles y tabletas, que son los dispositivos de los que se dispone para hacer pruebas.

[1] Material Design (en línea). http://www.google.com/design/. (fecha de consulta: 03/12/2014)

Esta adaptación requiere un esfuerzo adicional en la creación de las interfaces gráficas, así como en general en la capa de presentación de la aplicación. Será necesario crear en muchos casos dos interfaces XML para cada Actividad, así como usar Fragmentos en la creación de las Actividades para permitir la división dinámica de las pantallas en una o dos columnas, dependiendo del tamaño del dispositivo en que se visualice.

7.4 Usuarios

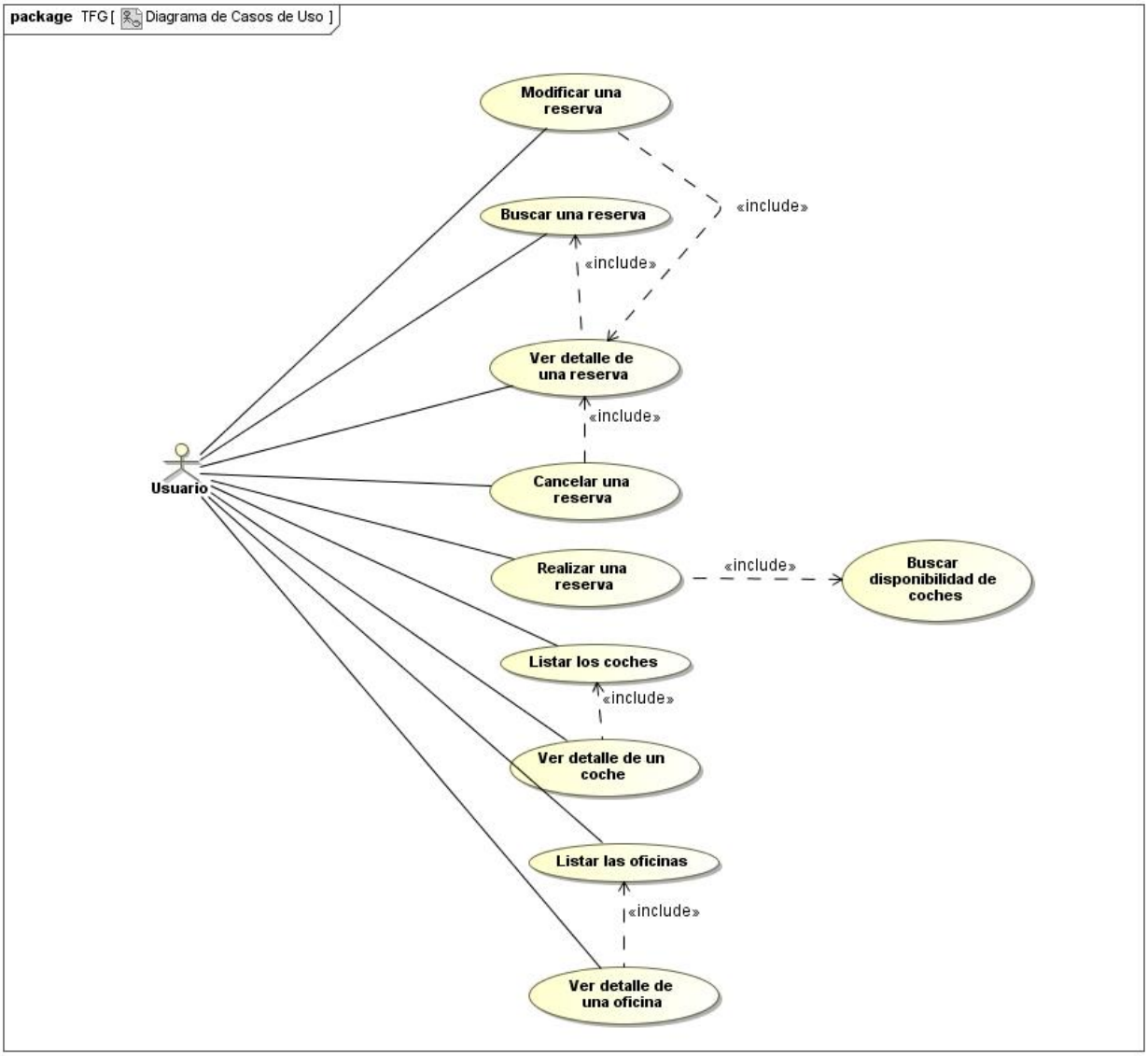
Los usuarios de la aplicación no necesitan identificarse con el sistema web, es decir, cada usuario puede descargarse la aplicación desde Google Play y realizar una reserva de un coche directamente sin necesidad de ningún tipo de registro previo.

Por tanto, tenemos un único actor en el sistema, que será el usuario que use la aplicación, y todas las funcionalidades de la aplicación podrán ser realizadas por él.

7.5 Casos de Uso

Los casos de uso identifican las acciones que se pueden llevar a cabo en la aplicación, y permiten estudiarlas aisladas del resto para centrarse en la funcionalidad que ofrece cada una y comprender sus relaciones.

Teniendo en cuenta los actores citados en el apartado anterior, presentamos a continuación el diagrama de casos de uso.



6. Casos de Uso

A continuación se realiza la descripción formal de cada caso de uso.

Caso de uso: Listar las oficinas.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere ver en qué oficinas puede alquilar un coche.
Propietario: quiere que los usuarios puedan ver las oficinas donde alquilar coches
Precondición: -
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el listado de oficinas.

Escenario principal de éxito:

1. El usuario pide listar las oficinas.
2. La aplicación muestra el listado de oficinas disponibles.

Extensiones:
2a. No hay oficinas disponibles.
2a1. La aplicación muestra un mensaje indicando que no hay oficinas disponibles.

Caso de uso: Listar los modelos.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere ver qué coches puede alquilar.
Propietario: quiere que los usuarios puedan ver los coches que pueden alquilar.
Precondición: -
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el listado de modelos disponibles para alquilar.
Escenario principal de éxito:

1. El usuario pide listar los modelos.
2. La aplicación muestra el listado de modelos disponibles.

Extensiones:
2a. No hay modelos disponibles.
2a1. La aplicación muestra un mensaje indicando que no hay modelos disponibles.

Caso de uso: Ver detalle de oficinas.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere ver el detalle de una oficina.
Propietario: quiere que los usuarios puedan ver todos los datos de las oficinas con detalle.
Precondición: Hay oficinas disponibles.
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el detalle de la oficina seleccionada.
Escenario principal de éxito:

1. El usuario *lista las oficinas*.
2. La aplicación muestra el listado de oficinas disponibles.
3. El usuario selecciona la oficina de la que quiere ver más detalles.
4. El sistema muestra los detalles de la oficina seleccionada.

Extensiones:
2a. No hay oficinas disponibles.
2a1. La aplicación muestra un mensaje indicando que no hay oficinas disponibles.
4a. El detalle de la oficina no está disponible.

4a1. La aplicación muestra un mensaje indicando que el detalle de la oficina no está disponible.
4a2. Se vuelve al paso 2.

Caso de uso: Ver detalle de un coche.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere ver el detalle de un coche.
Propietario: quiere que los usuarios puedan ver todos los datos de los coches disponibles para alquilar.
Precondición: Hay coches disponibles.
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el detalle del modelo seleccionado.
Escenario principal de éxito:

1. El usuario *lista los modelos*.
2. La aplicación muestra el listado de modelos disponibles.
3. El usuario selecciona el modelo del que quiere ver más detalles.
4. El sistema muestra los detalles del modelo seleccionado.

Extensiones:
2a. No hay modelos disponibles.
2a1. La aplicación muestra un mensaje indicando que no hay modelos disponibles.
4a. El detalle de un modelo no está disponible.
4a1. La aplicación muestra un mensaje indicando que el detalle del modelo no está disponible.
4a2. Se vuelve al paso 2.

Caso de uso: Buscar disponibilidad de coches.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere buscar disponibilidad y precios de los coches.
Propietario: quiere que los usuarios puedan consultar disponibilidad y precios de sus coches.
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el listado de coches disponibles según los parámetros de búsqueda junto con su precio de alquiler.
Escenario principal de éxito:

1. El usuario selecciona las zonas y fechas de recogida y devolución e indica consultar disponibilidad.
2. La aplicación consulta el Web Service con los parámetros indicados por el usuario y muestra los coches disponibles al usuario junto con su precio.

Extensiones:
2a. No hay modelos disponibles.
2a1. La aplicación muestra un mensaje indicando que no hay modelos disponibles.

Caso de uso: Realizar una reserva.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere realizar una reserva de un coche.
Propietario: quiere que los usuarios puedan reservas coches a través de la aplicación.
Garantías mínimas: -
Garantías en caso de éxito: Se realizará la reserva del modelo elegido y se mostrará el detalle de la reserva.
Escenario principal de éxito:

1. El usuario *busca disponibilidad de coches.*
2. El usuario selecciona el modelo que va a reservar.
3. El sistema muestra la lista de extras y accesorios disponibles para el modelo, fechas y oficinas seleccionadas.
4. El usuario selecciona los extras que desea.
5. El sistema muestra el formulario para introducir los datos del titular de la reserva.
6. El usuario introduce los datos e indica que quiere realizar la reserva.
7. El sistema envía la petición de reserva al Web Service y muestra el detalle de la reserva al usuario.

Extensiones:
1a. No hay modelos disponibles.
1a1. La aplicación muestra un mensaje indicando que no hay modelos disponibles.
4a. El usuario indica que no quiere ningún extra o accesorio.
4a1. Se continúa en el punto 5.
6a Hay datos obligatorios que faltan por introducir.
6a1. El sistema muestra un mensaje al usuario indicándole los campos que faltan por rellenar.
6a2. Se vuelve al paso 5.
7a. Hay un error al realizar la reserva.
7a1. El sistema muestra el error producido al usuario.
7a2. Se vuelve al paso 5.

Caso de uso: Buscar una reserva.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere buscar una reserva realizada anteriormente.
Propietario: quiere que los usuarios puedan consultar las reservas realizadas anteriormente.
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el listado de reservas que coinciden con los parámetros de búsqueda.
Escenario principal de éxito:

1. El usuario indica que quiere buscar una reserva.

2. El sistema muestra el listado de reservas realizadas y un filtro para especificar los parámetros de búsqueda.
3. El usuario rellena los datos del filtro e indica que quiere buscar las reservas.
4. El sistema muestra el listado de reservas que coinciden con los parámetros de búsqueda.

Extensiones:
2a. No hay reservas realizadas.
2a1. El sistema muestra un mensaje indicando que no hay reservas realizadas aún.
4a. No hay reservas que coincidan con los criterios de búsqueda.
4a1. El sistema muestra un mensaje indicando que no hay reservas que coincidan con los criterios de búsqueda.

Caso de uso: Ver detalle de una reserva.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere ver el detalle de una reserva realizada.
Propietario: quiere que los usuarios puedan consultar las reservas realizadas anteriormente y ver todos sus detalles.
Garantías mínimas: -
Garantías en caso de éxito: La aplicación mostrará el detalle de la reserva seleccionada.
Escenario principal de éxito:
1. El usuario *busca una reserva.*
2. El usuario indica que quiere ver el detalle de una reserva.
3. El sistema muestra el detalle de la reserva.

Extensiones:
1a. No hay reservas que coincidan con los criterios de búsqueda.
1a1. El sistema muestra un mensaje indicando que no hay reservas que coincidan con los criterios de búsqueda.

Caso de uso: Cancelar una reserva.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere cancelar una reserva realizada anteriormente.
Propietario: quiere que los usuarios puedan cancelar las reservas realizadas anteriormente si lo desean.
Garantías mínimas: -
Garantías en caso de éxito: La reserva será cancelada.
Escenario principal de éxito:
1. El usuario indica *ver el detalle de una reserva*.
2. El usuario indica que quiere cancelar la reserva.
3. El sistema pide la confirmación del usuario para cancelar la reserva.

4. El usuario confirma la acción.
5. El sistema envía la petición de cancelación al Web Service y muestra el mensaje de confirmación de reserva cancelada al usuario.

Extensiones:
1a. No se encuentra la reserva.
1a1. Se vuelve al paso 1.
4a. El usuario cancela la acción.
4a1. Se vuelve al paso 1.
5a. Se produce un error al cancelar la reserva.
5a1. El sistema muestra un mensaje al usuario indicando el error producido.
5a2. Se vuelve al paso 1.

Caso de uso: Modificar una reserva.
Actor principal: Usuario de la aplicación.
Ámbito: Aplicación Android de alquiler de coches.
Nivel de objetivo: Usuario.
Stakeholders e intereses:
Usuario de la aplicación: quiere modificar una reserva realizada anteriormente.
Propietario: quiere que los usuarios puedan modificar las reservas realizadas anteriormente.
Garantías mínimas: -
Garantías en caso de éxito: La aplicación modificará la reserva según los parámetros indicados.
Escenario principal de éxito:

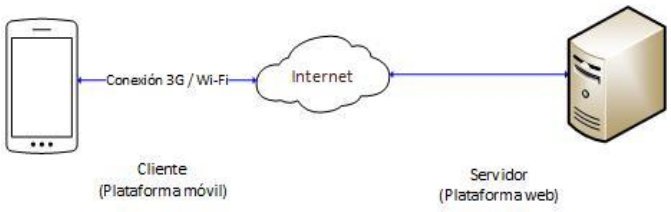
1. El usuario indica *ver el detalle de una reserva*.
2. El usuario indica que quiere modificar la reserva.
3. El sistema muestra la pantalla de modificación de reserva.
4. El usuario modifica los datos que desee y confirma los cambios.
5. El sistema envía la petición de modificación de reserva al Web Service y muestra el *detalle de la reserva* al usuario con un mensaje indicando que se ha modificado la reserva correctamente.

Extensiones:
1a. No se encuentra la reserva.
1a1. Se vuelve al paso 1.
4a. El usuario cancela la acción.
4a1. Se vuelve al paso 1.
5a. Se produce un error al modificar la reserva.
5a1. El sistema muestra un mensaje al usuario indicando el error producido.
5a2. Se vuelve al paso 3.

7.6 Diseño Técnico

La arquitectura general del proyecto se basa en un modelo cliente-servidor, donde la parte cliente será el dispositivo móvil con la aplicación instalada, y la parte servidora será el servicio web de Canarias.com, encargado de procesar las peticiones del cliente.

A continuación se muestra un diagrama general de este modelo:



7. Modelo Cliente-Servidor

El dispositivo móvil se conectará a Internet a través de una conexión 3G/4G o Wi-Fi, y de esta manera tendrá comunicación directa con el servicio web, que se encuentra en un servidor ubicado en un centro de datos.

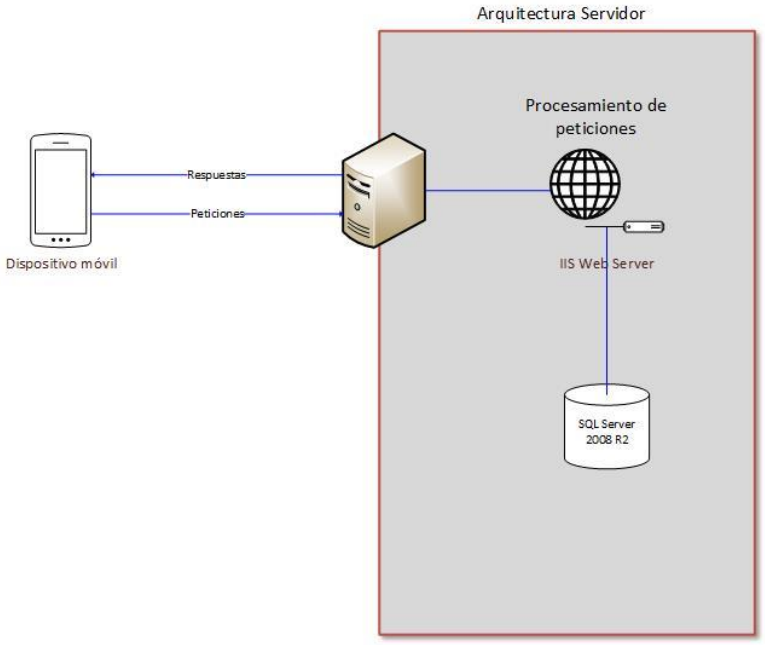
7.6.1 Arquitectura del servicio web

El servicio web se basa en un motor de reservas de coches desarrollado en el lenguaje de programación ASP, que expone una interfaz XML para recibir peticiones externas que permitan realizar reservas de coches a los clientes externos al sistema.

Esta API consta de unos métodos especificados en la documentación del servicio. Cada método está diseñado para realizar una acción específica (obtener disponibilidad, realizar una reserva, modificar una reserva, cancelar una reserva, etc.).

Al recibir una petición XML, el sistema primero valida los datos de la misma, la procesa, y devuelve una respuesta en formato XML al cliente con el resultado.

A continuación se muestra el diagrama de arquitectura general de la parte servidora:



8. Arquitectura del servicio web

13. El servicio web se encuentra constantemente en ejecución, esperando peticiones de los clientes.
14. Se encarga de procesar dichas peticiones y generar la respuesta adecuada.
15. Puede procesar múltiples peticiones al mismo tiempo, gracias al uso de múltiples hilos de ejecución por parte del servidor web IIS.
16. La base de datos SQL Server 2008 R2 se encuentra en la misma máquina física que el servicio web, por tanto la latencia de comunicación entre ambos servicios es mínima.

7.6.2 Arquitectura de la plataforma móvil

La plataforma móvil consta de la aplicación móvil que se va a desarrollar, y que se podrá instalar en cualquier dispositivo que use el sistema operativo Android y que sea compatible con la versión mínima del SDK que se usará, y que en nuestro caso se trata de la API 11 que corresponde a la versión de Android 3.0 Honeycomb.

La comunicación con el servicio web se realizará de la siguiente forma:

17. El usuario de la aplicación realizará alguna acción en la misma que implique el envío de una solicitud al servicio web.
18. La aplicación se queda a la espera de la respuesta.
19. El servicio web procesa la petición y elabora una respuesta XML que envía de vuelta a la aplicación cliente.

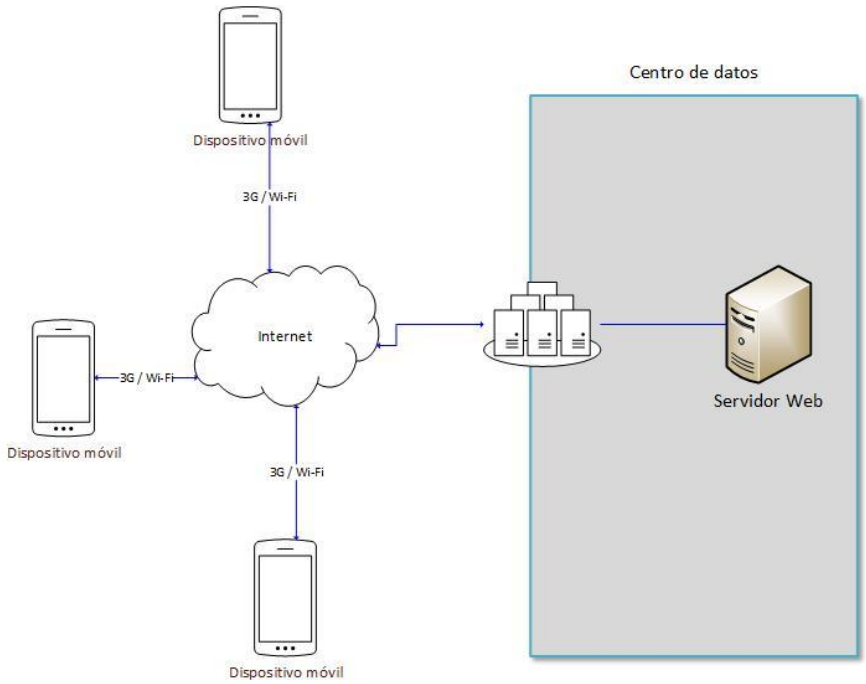
20. Una vez recibida la respuesta por parte de la aplicación, se analiza y procesa el resultado y se muestra la información al cliente en la interfaz gráfica.

7.6.3 Arquitectura Física

La arquitectura física se corresponde con la mayoría de entornos de aplicaciones móviles hoy en día. Los dispositivos móviles se conectarán a Internet a través de una conexión 3G/4G o Wi-Fi, por otro lado, el servidor que aloja el servicio web se encuentra en un centro de datos en Francia con conexión permanente a Internet. Por tanto toda la transmisión de información entre ambas partes se realiza a través de Internet.

Tanto en la fase de desarrollo como en la fase de producción el esquema de comunicación será el mismo, ya que en todo momento se hará uso del servicio web de Canarias.com para realizar las pruebas de la aplicación.

A continuación se muestra un diagrama de red con los dispositivos que componen la estructura física del sistema.



9. Arquitectura Física

7.6.4 Arquitectura Lógica

La aplicación se diseñará siguiendo una arquitectura por capas, con tres capas bien diferenciadas: capa de presentación, capa de negocio y capa de datos. Dentro de cada capa se

dividirán las funcionalidades en componentes individuales. Esto permitirá desacoplar las partes de la aplicación que realizan una misma funcionalidad, y por tanto se obtendrá una mayor cohesión y capacidad de reutilización en el código.

La **capa de presentación** estará ligada a las funciones propias del sistema operativo Android, y contendrá las partes que forman la interfaz gráfica, así como las clases de apoyo a la misma (Activities, Dialogs, Widgets, etc.).

La **capa de negocio** será la encargada de la comunicación entre la aplicación y el servicio web de Canarias.com. Además, contendrá los modelos del dominio que permitirán transformar los datos XML en objetos de negocio para su posterior manipulación.

La **capa de datos** será la encargada de la persistencia de los datos en la base de datos local de la aplicación. Se comunica directamente con la capa de negocio e implementa el patrón Data Access Object (DAO) para realizar la persistencia de los datos. Esta capa sigue las recomendaciones de Google para el acceso a datos en Android, implementando la clase abstracta SQLiteOpenHelper que permite crear, actualizar e interactuar con la base de datos SQLite de la aplicación.

Las ventajas de una arquitectura por capas son muchas, pero de entre ellas podemos destacar las siguientes:

- Permite la reutilización de componentes en una capa, sin afectar al resto de capas. Por ejemplo, si queremos modificar el sistema gestor de la base de datos, solo tendremos que modificar la capa de datos, y el resto de capas permanecen inalteradas.
- Permite probar los componentes por separado.
- Permite crear un sistema con un bajo acoplamiento entre sus componentes.

La arquitectura por capas también tiene algunas desventajas:

- Generalmente requiere realizar un mayor esfuerzo de desarrollo.
- Podría afectar al rendimiento el hecho de tener que pasar los datos a través de múltiples capas para realizar las operaciones.

A continuación se muestra el diagrama general de arquitectura de componentes de la aplicación:

Universitat Oberta de Catalunya

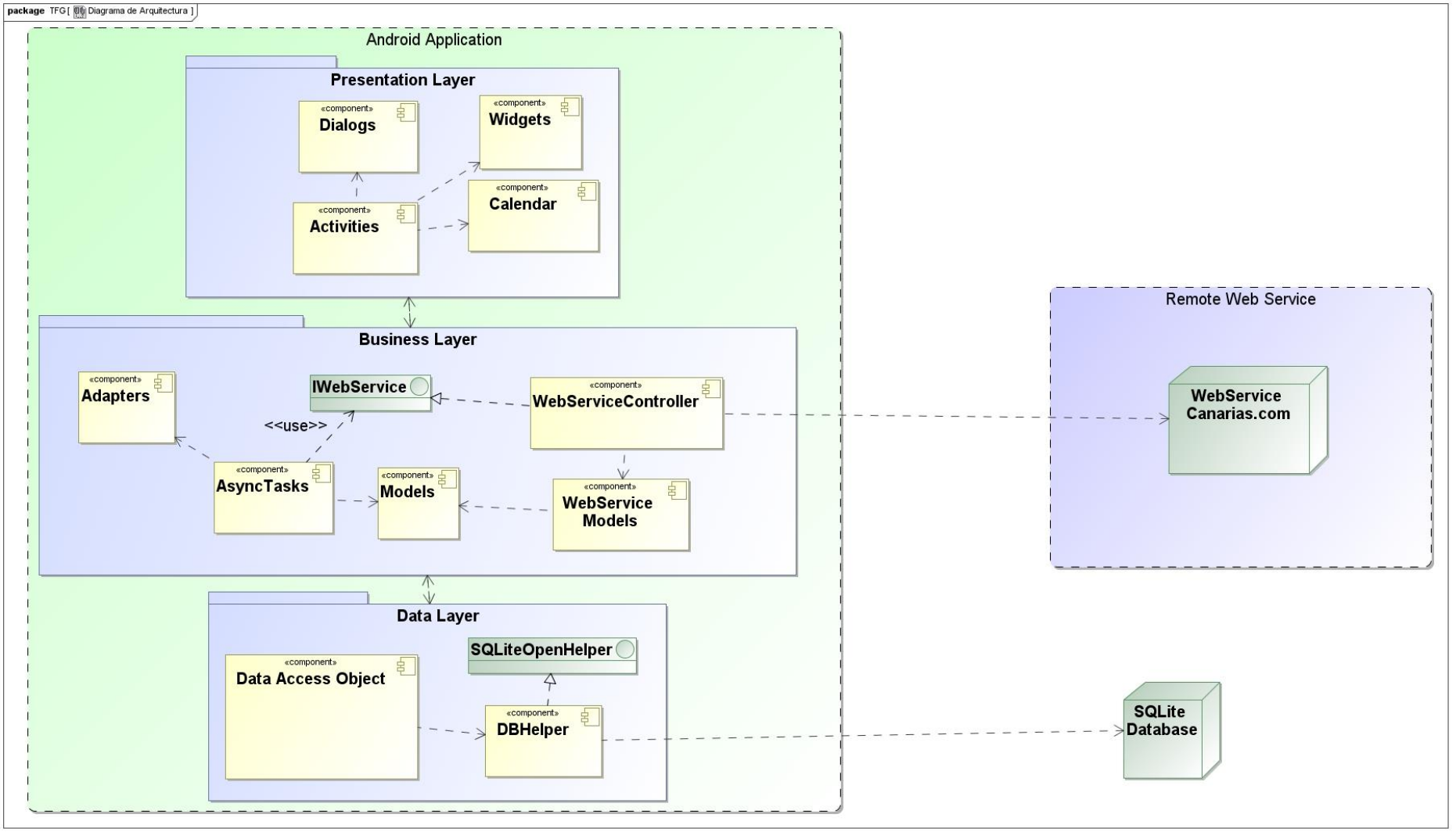
Alumno: David Lorenzo López

Consultor: Jordi Ceballos Villach

TRABAJO DE FIN DE GRADO

Aplicación Android Alquiler de Coches

Entrega (09/01/2015)



10. Arquitectura - General

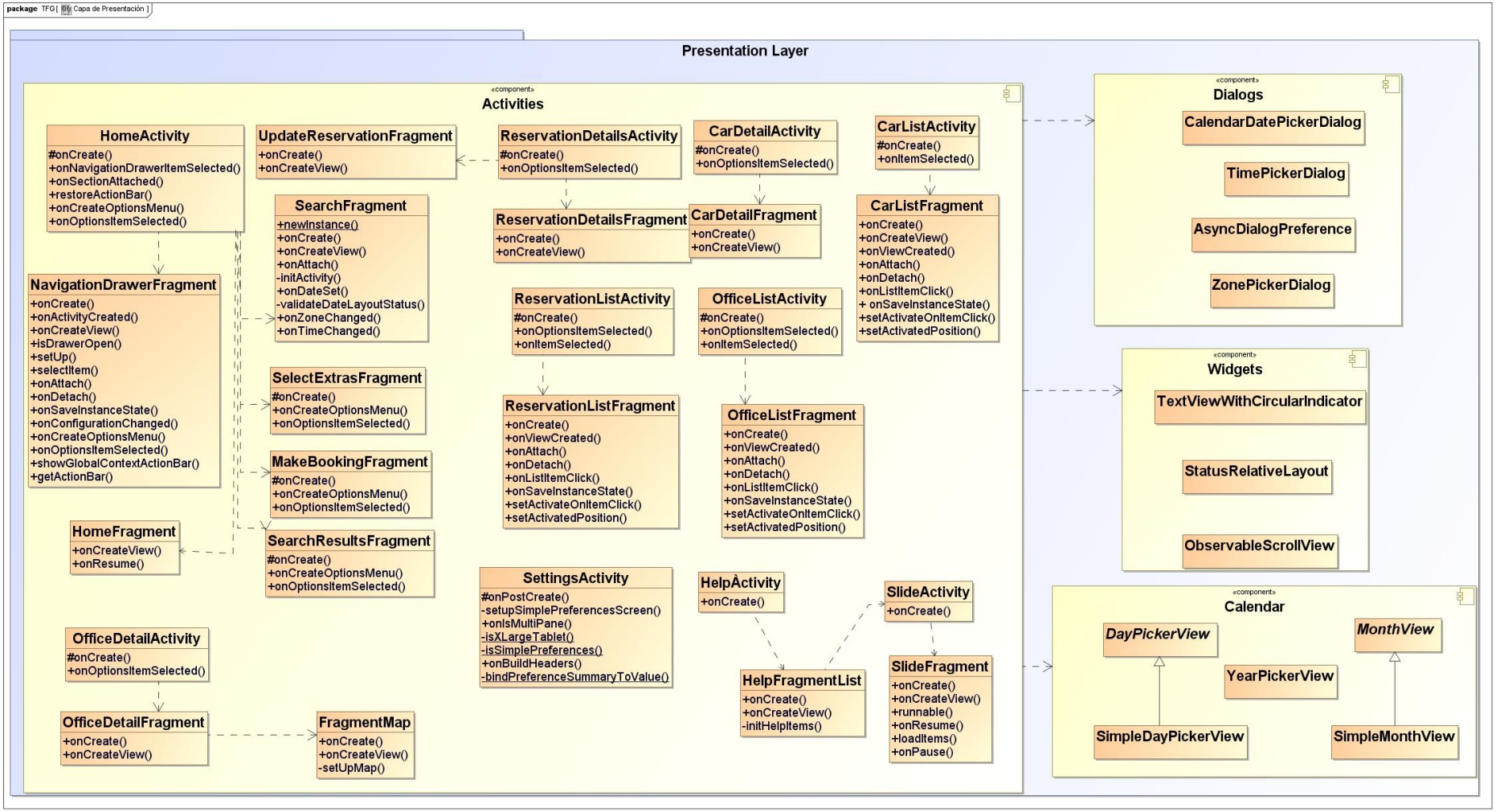
En los siguientes apartados iremos avanzando por cada una de las capas, indicando con más detalle la composición de cada componente. Para cada capa, en primer lugar se muestra un diagrama con las clases de cada componente y a continuación se realiza una breve explicación de cada uno.

7.6.4.1 Capa de Presentación

La capa de presentación es la encargada de recibir los datos desde la capa de negocio y mostrarlos al usuario. Esta capa usa todas las librerías y procesos propios de Android para gestionar el ciclo de vida de las Activities, manejar los eventos adecuadamente y mostrar las pantallas al usuario. Contiene también las implementaciones de los diálogos y calendarios usados por las Activities para interactuar con el usuario.

A continuación se muestra el diagrama detallado de la capa de presentación. Las flechas que conectan las clases y componentes indican las relaciones de uso que se establecen entre ellos. Se ha ocultado la definición de algunos métodos y atributos menos importantes para hacer más legible el diagrama.

Alumno: David Lorenzo López
Consultor: Jordi Ceballos Villach



11. Arquitectura - Capa de Presentación

Como se puede apreciar, la capa de presentación contiene cuatro componentes principales: Dialogs, Widgets, Calendar y Activities, que detallamos a continuación.

21. **Dialogs**: Son clases que heredan de la clase Dialog de Android, la cual permite crear un diálogo con una interfaz personalizada para mostrar algún dato al usuario, alertar sobre algún evento del sistema, e incluso permitirle interactuar mediante formularios. En este caso, se usa para los selectores de fechas y zonas, así como para la descarga de zonas y vehículos que se realiza desde la pantalla de Configuración.
22. **Widgets**: Son controles personalizados que generalmente heredan de un control básico de Android (TextView, EditText, ScrollView, etc.). Se usan para añadir nuevas funcionalidades a los controles básicos sin tener que crear uno completo desde cero, gracias a la propiedad de herencia de los lenguajes orientados a objetos, como es el caso de Java.
23. **Calendar**: Contiene los widgets específicos para el calendario que se usa a lo largo de la aplicación para seleccionar las fechas en los formularios de búsqueda. Se trata de widgets igual que los anteriores, pero que se han encapsulado dentro de otro componente ya que hacen referencia a una misma funcionalidad, a diferencia del resto.
24. **Activities**: Es la parte principal de una aplicación Android. Se encargan de gestionar el ciclo de vida de la misma y de guiar al usuario a través de los diferentes casos de uso que se pueden realizar.

    Como se puede apreciar, dentro del componente Activities existen además otras clases cuyo nombre termina en *Fragment*. Los Fragments permiten realizar un diseño adaptativo y dinámico que tenga una mayor flexibilidad a la hora de reusar partes de la interfaz y del código de aplicación.

    Un Fragment representa una porción de la interfaz de una Activity, y una Activity puede usar varios fragmentos e ir intercambiándolos dinámicamente para construir una interfaz multi-panel, incluso pueden ser reusados desde distintas Activities.

    Los fragments por tanto nos permitirán tener una mayor flexibilidad a la hora de decidir el formato de visualización de los datos, ya que cada Fragment puede tener asociada una interfaz u otra, y se pueden intercambiar o cargar dinámicamente dependiendo, por ejemplo, del tamaño de la pantalla del usuario, o de la versión del SO del dispositivo.

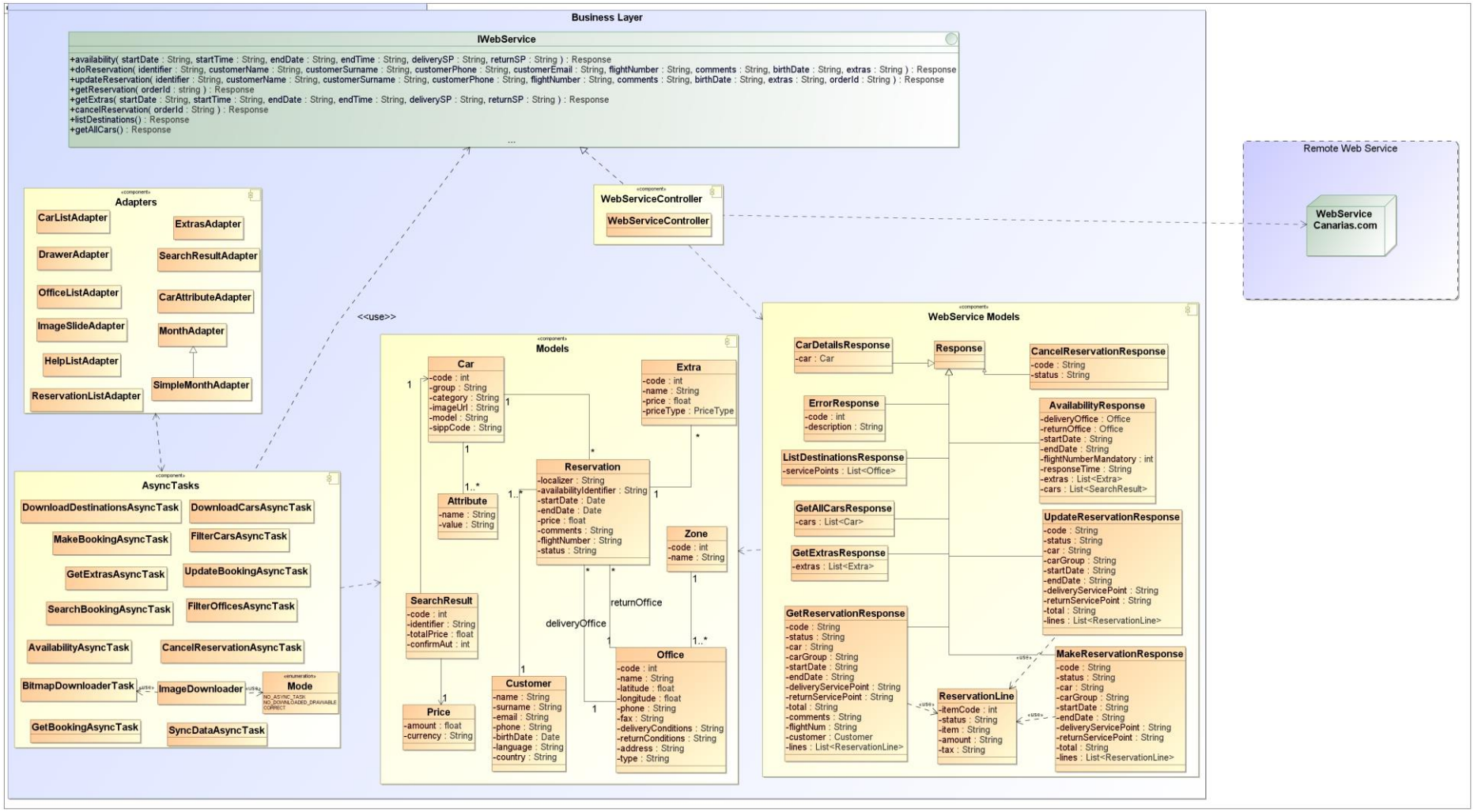
7.6.4.2 Capa de Negocio

La capa de negocio realiza las acciones propias del dominio de la aplicación, en este caso, el alquiler de coches online.

Contiene por tanto las clases que componen el modelo del dominio (Models), clases de apoyo a la capa de presentación pero que contienen lógica de negocio (Adapters), clases que realizan tareas en segundo plano (AsyncTasks), así como los modelos y el controlador del web Service que serán los encargados de interactuar con el servicio web de Canarias.com (Webservice Models, Controllers).

En esta capa, se realiza la comunicación con el servicio web de Canarias.com, es decir, es la puerta de entrada y salida a nuestra aplicación. En las peticiones de salida, los datos presentes en los modelos del dominio son convertidos en una URL que se usará para realizar una petición HTTP GET al servicio web, en dicha URL van los parámetros que especifican la función a ejecutar y los valores que se pasarán como argumentos. Por otro lado, en las respuestas de entrada, los datos que llegan en la respuesta HTTP en formato XML son *parseados* a clases del modelo de dominio, para su posterior utilización por parte del resto de módulos.

A continuación se muestra el diagrama específico de esta capa, así como una explicación detallada de cada componente. Igual que en el caso anterior, se han ocultado algunos atributos y métodos menos relevantes para hacer más legible el diagrama.


Business Layer
IWebService
+availability(startDate : String, startTime : String, endDate : String, endTime : String, deliverySP : String, returnSP : String) : Response
+doReservation(identifier : String, customerName : String, customerSurname : String, customerPhone : String, customerEmail : String, flightNumber : String, comments : String, birthDate : String, extras : String) : Response
+updateReservation(identifier : String, customerName : String, customerSurname : String, customerPhone : String, flightNumber : String, comments : String, birthDate : String, extras : String, orderId : String) : Response
+getReservation(orderId : string) : Response
+getExtras(startDate : String, startTime : String, endDate : String, endTime : String, deliverySP : String, returnSP : String) : Response
+cancelReservation(orderId : String) : Response
+listDestinations() : Response
+getAllCars() : Response
Remote Web Service
WebService Canarias.com
WebServiceController
Adapters
CarListAdapter
ExtrasAdapter
DrawerAdapter
SearchResultAdapter
OfficeListAdapter
CarAttributeAdapter
ImageSlideAdapter
MonthAdapter
HelpListAdapter
SimpleMonthAdapter
ReservationListAdapter
<<use>>
AsyncTasks
DownloadDestinationsAsyncTask
DownloadCarsAsyncTask
MakeBookingAsyncTask
FilterCarsAsyncTask
GetExtrasAsyncTask
UpdateBookingAsyncTask
SearchBookingAsyncTask
FilterOfficesAsyncTask
AvailabilityAsyncTask
CancelReservationAsyncTask
BitmapDownloaderTask
ImageDownloader
Mode
GetBookingAsyncTask
SyncDataAsyncTask
Models
Car
-code : int
-group : String
-category : String
-imageUrl : String
-model : String
-sippCode : String
Extra
-code : int
-name : String
-price : float
-priceType : PriceType
Reservation
-localizer : String
-availabilityIdentifier : String
-startDate : Date
-endDate : Date
-price : float
-comments : String
-flightNumber : String
-status : String
Attribute
-name : String
-value : String
Zone
-code : int
-name : String
SearchResult
-code : int
-identifier : String
-totalPrice : float
-confirmAut : int
returnOffice
deliveryOffice
Office
-code : int
-name : String
-latitude : float
-longitude : float
-phone : String
-fax : String
-deliveryConditions : String
-returnConditions : String
-address : String
-type : String
Customer
-name : String
-surname : String
-email : String
-phone : String
-birthDate : Date
-language : String
-country : String
Price
-amount : float
-currency : String
WebService Models
CarDetailsResponse
-car : Car
Response
CancelReservationResponse
-code : String
-status : String
ErrorResponse
-code : int
-description : String
AvailabilityResponse
-deliveryOffice : Office
-returnOffice : Office
-startDate : String
-endDate : String
-flightNumberMandatory : int
-responseTime : String
-extras : List<Extra>
-cars : List<SearchResult>
ListDestinationsResponse
-servicePoints : List<Office>
GetAllCarsResponse
-cars : List<Car>
UpdateReservationResponse
-code : String
-status : String
-car : String
-carGroup : String
-startDate : String
-endDate : String
-deliveryServicePoint : String
-returnServicePoint : String
-total : String
-lines : List<ReservationLine>
GetExtrasResponse
-extras : List<Extra>
GetReservationResponse
-code : String
-status : String
-car : String
-carGroup : String
-startDate : String
-endDate : String
-deliveryServicePoint : String
-returnServicePoint : String
-total : String
-comments : String
-flightNum : String
-customer : Customer
-lines : List<ReservationLine>
ReservationLine
-itemCode : int
-status : String
-item : String
-amount : String
-tax : String
MakeReservationResponse
-code : String
-status : String
-car : String
-carGroup : String
-startDate : String
-endDate : String
-deliveryServicePoint : String
-returnServicePoint : String
-total : String
-lines : List<ReservationLine>

12. Arquitectura - Capa de Negocio

25. **Adapters**: Los adapters son usados para popular con datos una lista de ítems, que normalmente va dentro de un ListView y se puede avanzar sobre ella deslizando con el dedo hacia arriba y abajo.
    Existen ciertos tipos de adapters predefinidos en Android como la clase BaseAdapter o ArrayAdapter, pero que normalmente no son suficientes para cubrir las necesidades de las aplicaciones. Para solucionarlo, es posible crear una clase que hereda de una de esas clases predefinidas, por ejemplo de la clase ArrayAdapter, y que permite modificar el comportamiento por defecto de dicha clase para mostrar, por ejemplo, un layout personalizado para cada ítem de la lista.
    Con esta simple acción, las posibilidades de personalización que se abren son enormes, ya que incluso cada ítem de la lista puede tener un layout o un comportamiento distinto.
26. **AsyncTasks**: Son clases que contienen lógica de negocio y se ejecutan en segundo plano para evitar bloquear la interfaz gráfica durante su ejecución. Heredan de la clase parametrizable AsyncTask e implementan sus 3 métodos básicos:
    - onPreExecute(): En este método se realizan las acciones previas a la ejecución en segundo plano, por ejemplo actualizar la interfaz o mostrar un diálogo para indicar al usuario que comienza a ejecutarse cierta acción. Este método es libre para actualizar la interfaz de usuario sin ninguna restricción ya que se ejecuta en el mismo hilo de ejecución.
    - doInBackground(): Es el método que realmente realiza la tarea en segundo plano. En nuestro caso se realizan principalmente conexiones al web Service o tareas que conlleven un tiempo de ejecución que pueda llegar a bloquear la interfaz gráfica por su elevado tiempo de ejecución. Desde este método no se puede realizar ningún cambio en la interfaz gráfica ya que se ejecuta en un hilo de ejecución distinto.
    - onPostExecute(): Una vez finaliza la tarea, se ejecuta este método que ya es libre de volver a actualizar la interfaz gráfica. Generalmente se usa para indicar al usuario que ha finalizado la tarea, y el mostrar resultado de la misma.

    En el proyecto se han usado las AsyncTask para implementar la lógica de negocio de la aplicación, desde ellas se conectará con el controlador del web Service para enviar las peticiones y recibir las respuestas, se conectará con los objetos DAO para el acceso a base de datos y se realizaran otras tareas que puedan llegar a bloquear la interfaz de usuario por su elevado tiempo de ejecución.
27. **Models:** Son clases que forman el modelo del dominio. Sirven para almacenar temporalmente la información y transferirla entre las diferentes capas de la aplicación.

28. **Webservice Models:** Al igual que los anteriores, sirven para transferir información, pero esta vez entre el Web Service y la aplicación. Estas clases se usan para parsear las respuestas XML recibidas desde el Web Service y transformarlas en objetos que luego puedan ser utilizados más fácilmente en la aplicación.
29. **Webservice Controller:** El Webservice Controller implementa la interfaz IWebservice que define las operaciones permitidas por el web Service de Canarias.com.
    Cada operación recibe unos parámetros de entrada y genera una URL a la que se realiza una petición HTTP. Al recibir la respuesta se parsea el contenido XML en objetos del modelo del webservice (clases del componente WebserviceModels) y se devuelve el resultado al componente que inició la llamada.

7.6.4.3 Capa de Datos

La capa de datos es la encargada de la persistencia de los datos en la base de datos local de la aplicación, en este caso, una base de datos SQLite cuyo esquema se explicará más adelante.

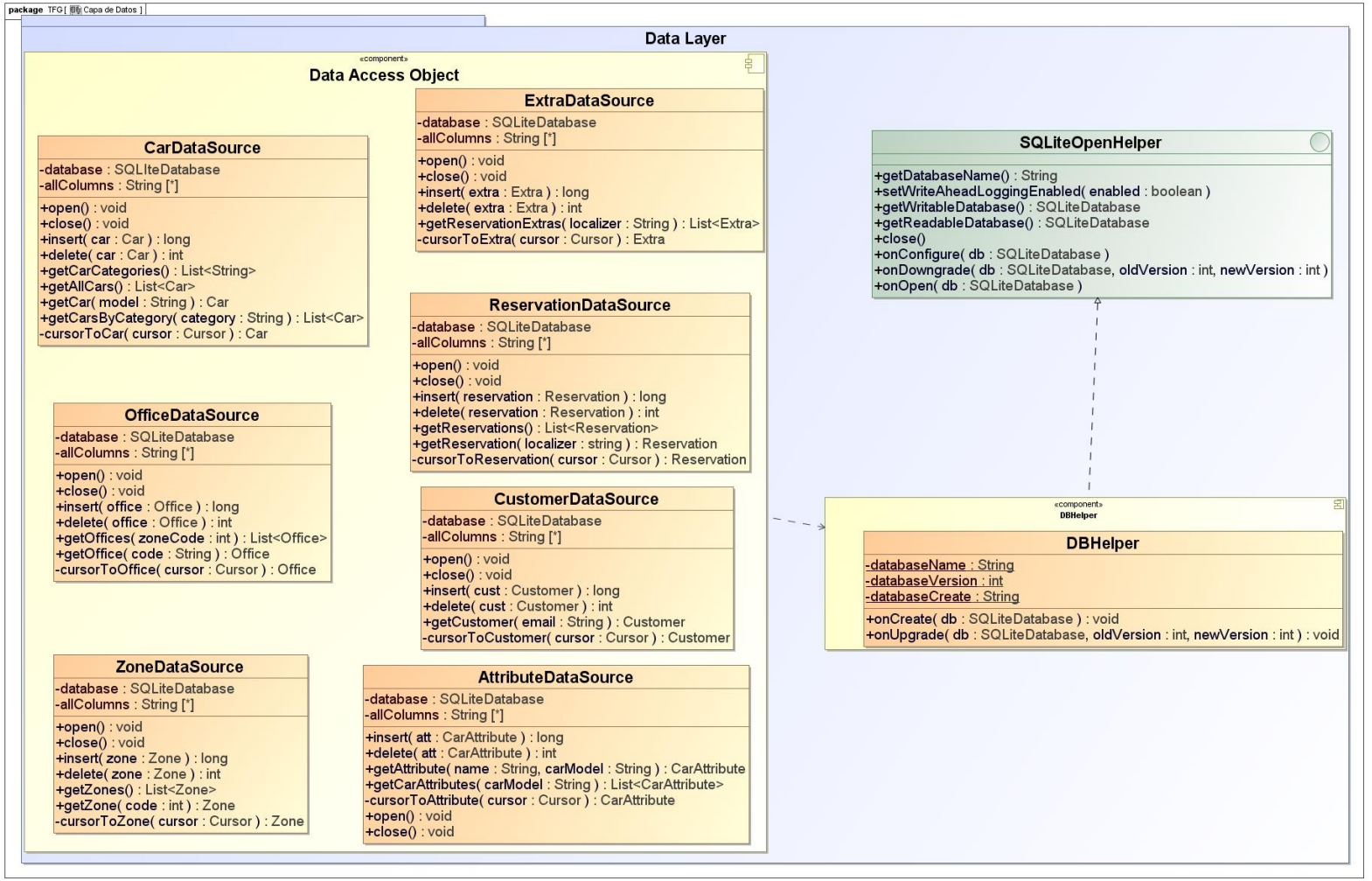
Para realizar la persistencia de datos, se seguirá el **patrón DAO** (Data Access Object). Este patrón provee una interfaz clara y bien diferenciada entre la aplicación y el acceso a datos, lo cual hace que se pueda intercambiar el motor gestor de bases de datos sin tocar la lógica de negocio de la aplicación.

La forma de implementar el patrón consiste en crear una clase por cada objeto que se quiera persistir en base de datos, e implementar los métodos necesarios para almacenar, actualizar, obtener y borrar ese objeto de la base de datos. Con esto, si se cambiara el motor de BD, solo habría que venir a esta clase y cambiar la parte en que se accede a base de datos, pero el resto del código de la aplicación quedaría igual.

Además, en esta capa se encuentra la clase **DBHelper**, que implementa la clase abstracta **SQLiteOpenHelper** proporcionada en el SDK de Android para acceder a la base de datos de la aplicación. Esta clase es la encargada de crear la Base de Datos la primera vez que se inicia la aplicación, así como realizar las actualizaciones del esquema cuando se actualice la aplicación si es necesario. También es necesaria para obtener el objeto SQLiteDatabase necesario para interactuar con la base de datos por parte de los objetos DAO.

A continuación se muestra el diagrama específico de esta capa, así como una explicación detallada de cada componente.

Alumno: David Lorenzo López
Consultor: Jordi Ceballos Villach



13. Arquitectura - Capa de datos

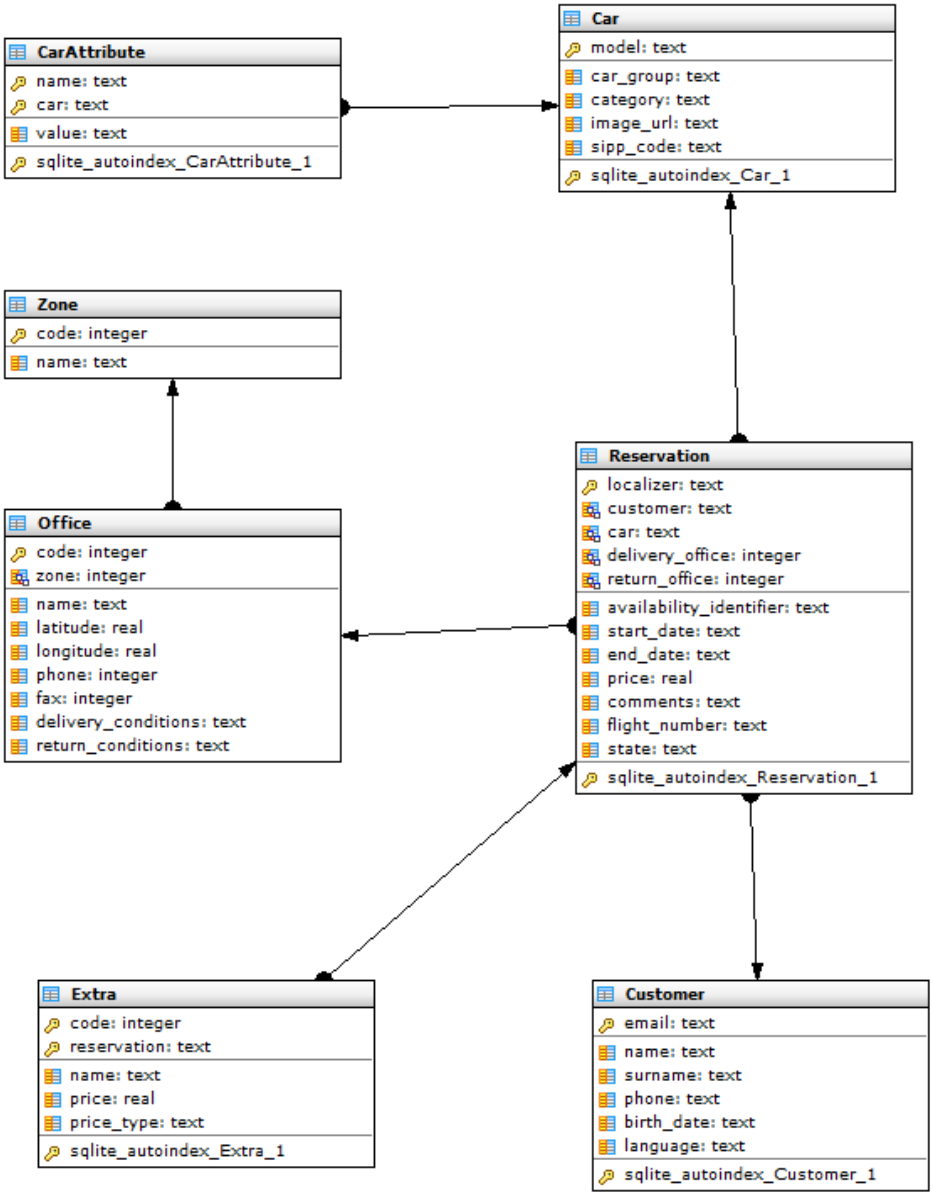
30. **Data Access Object:** Contiene las clases que implementan el patrón DAO. Una clase por cada entidad de la base de datos que encapsulará la lógica de acceso a datos de esa entidad, y de esta manera aislará esa lógica del resto de la aplicación creando una interfaz clara entre la capa de negocio y la base de datos.
31. **DBHelper:** Contiene la clase DBHelper que extiende la clase abstracta SQLiteOpenHelper proporcionada en el SDK de Android. Permite la creación y actualización del esquema de la base de datos, así como la obtención del objeto SQLiteDatabase para contactar con la base de datos y realizar operaciones sobre ella.

7.6.5 Arquitectura de Base de Datos

La base de datos de la aplicación es la encargada de almacenar los vehículos disponibles para reservar, las zonas y oficinas disponibles para realizar la reserva de vehículos, así como las reservas realizadas por el usuario desde la aplicación con toda su información incluyendo los datos del cliente, extras, etc.

7.6.5.1 Modelo relacional de la base de datos

En primer lugar se muestra el diagrama del modelo relacional de la base de datos, en el que se pueden visualizar las entidades y las relaciones entre ellas. También se muestra la leyenda con la explicación de cada tipo de icono del diagrama.



14. Arquitectura - Diagrama de Base de Datos

Leyenda:

	Campo de la tabla
	Clave foránea
	Clave primaria

A continuación realizamos una explicación de cada entidad del diagrama, así como de sus relaciones con el resto de entidades.

Entidad	Descripción	Cardinalidad	Entidad relacionada
CarAttribute	Almacena los atributos de cada coche.	N:1	Car
Car	Almacena los datos de los coches disponibles para reservar.	1:N	CarAttribute
		1:N	Reservation
Customer	Almacena los datos del cliente que ha realizado la reserva.	1:N	Reservation
Extra	Almacena los datos de los extras de las reservas.	N:1	Reservation
Zone	Almacena las zonas en las que se encuentran las oficinas.	1:N	Zone
Office	Almacena los datos de las oficinas donde se pueden alquilar coches.	N:1	Zone
		2:N	Reservation
Reservation	Almacena la información de las reservas realizadas por el usuario de la aplicación.	N:2	Zone
		1:N	Extra
		N:1	Car
		N:1	Customer

CarAttribute:

Definición de la entidad CarAttribute.

Atributo	Descripción	Tipo	Clave
name	Nombre del atributo	Text	Sí: PK
car	Código del coche al que pertenece el atributo. FK: → Car	Text	Sí: PK
value	Valor del atributo	Text	No

Car:

Definición de la entidad Car.

Atributo	Descripción	Tipo	Clave
model	Modelo del coche	Text	Sí: PK
car_group	Grupo al que pertenece el coche.	Text	No
category	Categoría del coche.	Text	No
Image_url:	URL de la imagen del coche.	Text	No
sipp_code	Código SIPP que identifica las características del coche.	Text	No

Zone:

Definición de la entidad Zone.

Atributo	Descripción	Tipo	Clave
Code	Código de la zona	Integer	Sí: PK
Name	Nombre de la zona	Text	No

Office:

Definición de la entidad Office.

Atributo	Descripción	Tipo	Clave
code	Código de la oficina	Integer	Sí: PK
zone	Código de la zona a la que pertenece la oficina. FK: → Zone	Integer	Si: FK References Zone(code)
Name	Nombre de la oficina	Text	No
Latitude	Latitud de la oficina.	Real	No
Longitude	Longitud de la oficina	Real	No
phone	Teléfono de la oficina	Text	No
Fax	Fax de la oficina	Text	No

deliveryConditions	Texto indicando las condiciones e indicaciones para la recogida del coche.	Text	No
returnConditions	Texto indicando las condiciones e indicaciones para la devolución del coche.	Text	No

Customer:

Definición de la entidad Customer.

Atributo	Descripción	Tipo	Clave
Email	E-mail del cliente	Text	Sí: PK
Name	Nombre del cliente	Text	No
Surname	Apellidos del cliente	Text	No
phone	Teléfono del cliente	Text	No
Birth_date	Fecha de nacimiento del cliente	Text	No
Language	Código de idioma del cliente: es, en, fr, it, ru, de.	Text	No

Extra:

Definición de la entidad Extra.

Atributo	Descripción	Tipo	Clave
Code	Código del extra	Text	Sí: PK
Reservation	Localizador de la reserva a la que pertenece el extra	Text	Si: PK y FK References Reservation(localizar)
Name	Nombre del extra	Text	No
Price	Precio del extra	Real	No
PriceType	Tipo de precio del extra. Hay dos posibles valores: 32. DAILY: Precio por día 33. TOTAL: Precio para toda la reserva.	Text	No

Reservation:

Definición de la entidad Reservation.

Atributo	Descripción	Tipo	Clave
Localizer	Localizador de la reserva	Text	Sí: PK
Customer	Email del cliente que ha realizado la reserva	Text	Si: FK References Customer(email)
Car	Modelo del coche que se ha reservado	Text	Si: FK References Car(model)
deliveryOffice	Código de la oficina de recogida del coche	Integer	Si: FK References Office(code)
returnOffice	Código de la oficina de devolución del coche	Integer	Si: FK References Office(code)
Availability_identifier	Código que se recibe en la petición de disponibilidad y se envía al confirmar la reserva. Está asociado en el lado del web Service a las oficinas y fechas de recogida y devolución.	Text	No
Start_date	Fecha y hora de inicio de la reserva	Text	No
End_date	Fecha y hora de devolución de la reserva.	Text	No
Price	Precio total de la reserva	Real	No
Comments	Comentarios que el cliente ha hecho en la reserva	Text	No
Flight_number	Número de vuelo, que se pide al cliente en el caso de que alguna de las oficinas sea un aeropuerto.	Text	No
State	Estado de la reserva: 34. Confirmada 35. Pendiente de confirmación 36. Cancelada	Text	No

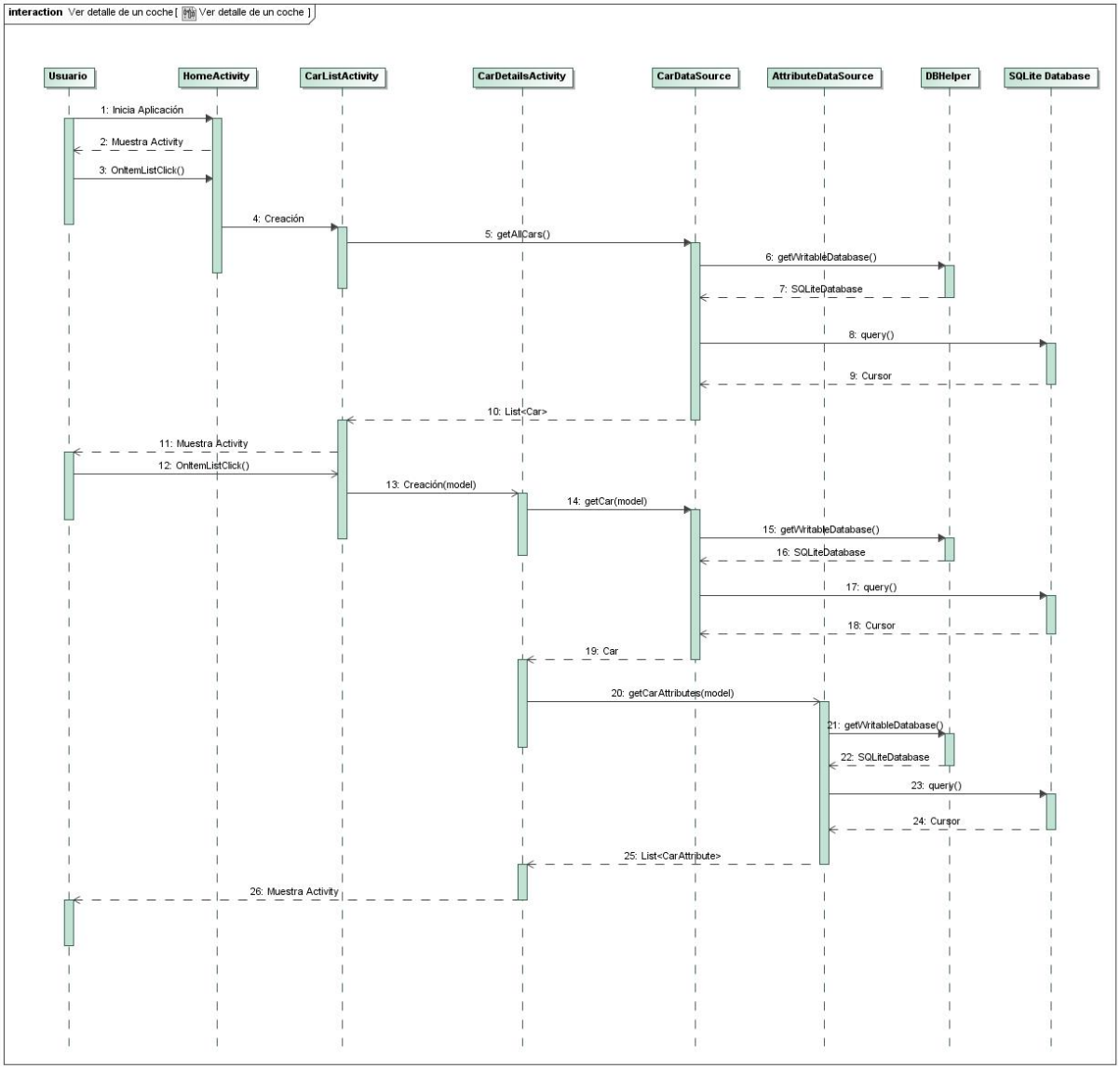
7.6.6 Diagramas de Secuencia

Los diagramas de secuencia permiten representar la interacción entre los objetos y componentes de la aplicación durante la ejecución de un caso de uso.

Existen numerosos casos de uso en el proyecto, pero la mayoría siguen una interacción similar, por tanto se presentan a continuación dos de los ejemplos más ilustrativos de casos de uso. En primer lugar se presenta la interacción de componentes en el caso de uso **Ver el detalle de un coche**, mientras que en el segundo caso se representa el proceso de **Creación de una Reserva de Coche**, el caso de uso principal y más complejo de la aplicación.

7.6.6.1 Diagrama de Secuencia – Ver el detalle de un coche

A continuación se presenta el diagrama de secuencia del caso de uso Ver el detalle de un coche.



15. Secuencia - Ver detalle de un coche

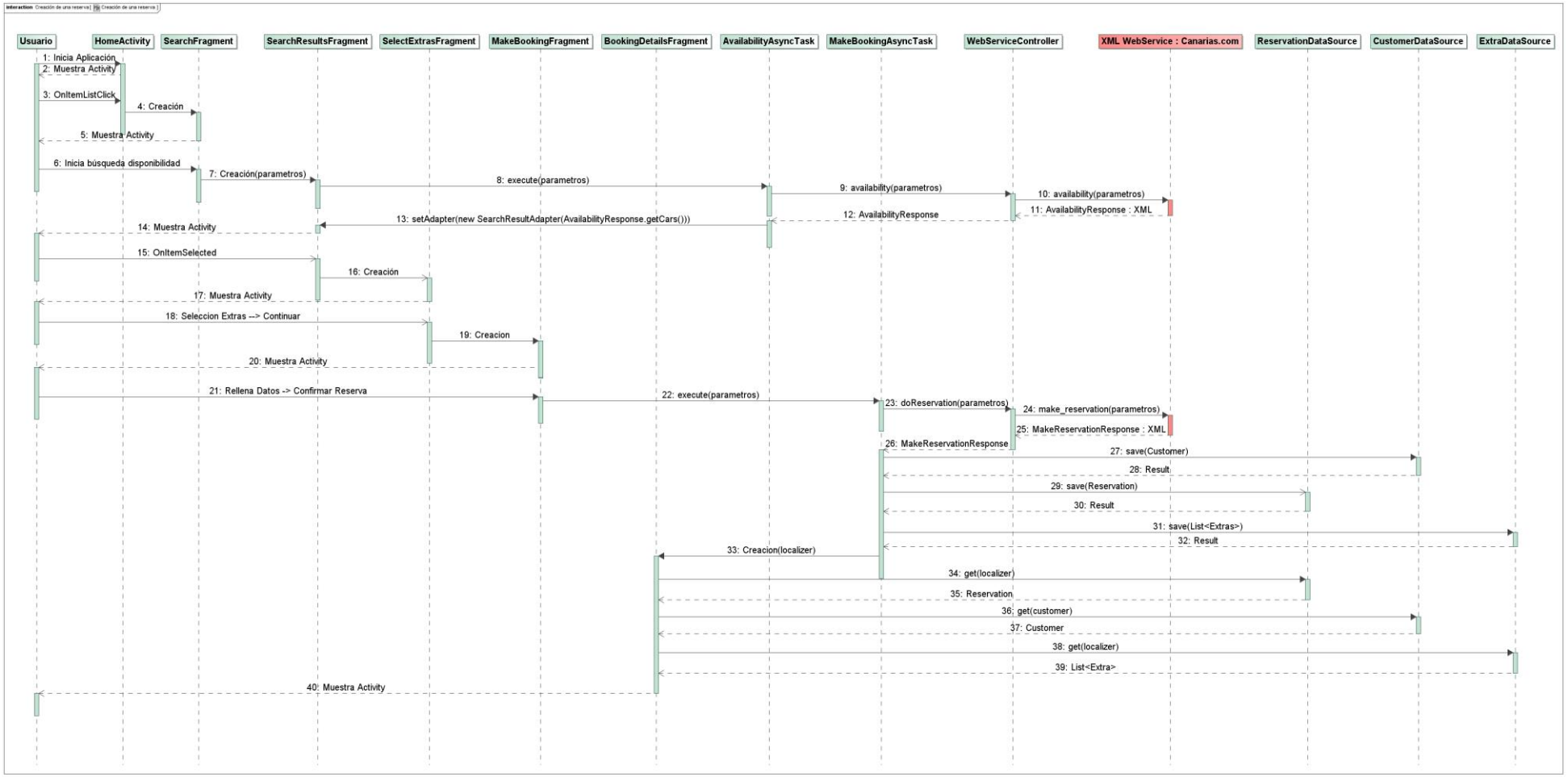
Seguidamente se indica la explicación de cada mensaje del diagrama.

ID	Nombre del mensaje	Descripción
1	Inicia Aplicación	El usuario pulsa sobre el icono de la aplicación en su móvil y se inicia la misma. El ActivityManager de Android crea la HomeActivity, especificada como Activity principal en el manifest de la aplicación.
2	Muestra Activity	Se muestra la Activity al usuario.
3	OnItemListClick()	El usuario selecciona la opción Ver Coches del menú de la aplicación.
4	Creación	Desde la HomeActivity se inicia un Intent para lanzar la CarListActivity que mostrará la lista de coches disponibles.
5	getAllCars()	Durante la creación de CarListActivity, se crea una instancia de CarDataSource y se llama a su método getAllCars() para obtener la lista de coches.
6	getWritableDatabase()	El data source crea una instancia de DBHelper para obtener la base de datos.
7	SQLiteDatabase	DBHelper devuelve una instancia de la base de datos con permiso de escritura.
8	query()	El data source realiza una consulta contra la base de datos para obtener los coches disponibles.
9	Cursor	La base de datos devuelve un cursor a través del cual se puede iterar para obtener los registros devueltos por la consulta.
10	List<Car>	El data source transforma el cursor en una lista de objetos Car y los devuelve a la CarListActivity.
11	Muestra Activity	Se muestra CarListActivity al usuario con la lista de coches.
12	OnItemListClick()	El usuario ha pulsado sobre el modelo del que quiere ver más detalles.
13	Creación(model)	Se crea la Activity CarDetailsActivity.
14	getCar(model)	Durante su creación, se crea una instancia de CarDataSource y se llama a su método getCar para obtener el detalle del coche.
15	getWritableDatabase()	El data source crea una instancia de DBHelper para obtener la base de datos.
16	SQLiteDatabase	DBHelper devuelve una instancia de la base de datos con permiso de escritura.
17	query()	El data source realiza una consulta contra la base de datos para obtener el detalle del coche indicado.
18	Cursor	La base de datos devuelve un cursor a través del cual se puede iterar para obtener los registros devueltos por la consulta.
19	Car	El data source crea una instancia de Car y la devuelve a CarDetailsActivity.
20	getCarAttributes()	A continuación se crea una instancia de AttributeDataSource y se llama a su método getCarAttributes para obtener los atributos del coche, a partir de su modelo.
21	getWritableDatabase()	El data source crea una instancia de DBHelper para obtener la base de datos.

22	SQLiteDatabase	DBHelper devuelve una instancia de la base de datos con permiso de escritura.
23	query()	El data source realiza una consulta contra la base de datos para obtener los atributos del coche indicado.
24	Cursor	La base de datos devuelve un cursor a través del cual se puede iterar para obtener los registros devueltos por la consulta.
25	List<CarAttribute>	El data source crea una lista de CarAttribute, que contiene los atributos del coche, y la devuelve a CarDetailsActivity.
26	Muestra Activity	Se muestra CarDetailsActivity al usuario, con el detalle del coche especificado.

7.6.6.2 Diagrama de Secuencia – Realizar una reserva

A continuación se muestra el diagrama de secuencia para el caso de uso Realizar una reserva. Al ser un caso de uso bastante grande en el que intervienen muchos componentes, y para hacer más legible el diagrama, se ha omitido la interacción de los Data Sources con DBHelper y la base de datos, ya que el protocolo es siempre el mismo, y se ha visto en el diagrama anterior en los mensajes 6, 7, 8 y 9; 15, 16, 17 y 18, y 21, 22, 23 y 24.


Usuario
HomeActivity
SearchFragment
SearchResultsFragment
SelectExtrasFragment
MakeBookingFragment
BookingDetailsFragment
AvailabilityAsyncTask
MakeBookingAsyncTask
WebServiceController
XML WebService : Canarias.com
ReservationDataSource
CustomerDataSource
ExtraDataSource
1: Inicia Aplicación
2: Muestra Activity
3: OnItemListClick
4: Creación
5: Muestra Activity
6: Inicia búsqueda disponibilidad
7: Creación(parametros)
8: execute(parametros)
9: availability(parametros)
10: availability(parametros)
11: AvailabilityResponse : XML
12: AvailabilityResponse
13: setAdapter(new SearchResultAdapter(AvailabilityResponse.getCars()))
14: Muestra Activity
15: OnItemSelected
16: Creación
17: Muestra Activity
18: Seleccion Extras --> Continuar
19: Creacion
20: Muestra Activity
21: Rellena Datos -> Confirmar Reserva
22: execute(parametros)
23: doReservation(parametros)
24: make_reservation(parametros)
25: MakeReservationResponse : XML
26: MakeReservationResponse
27: save(Customer)
28: Result
29: save(Reservation)
30: Result
31: save(List<Extras>)
32: Result
33: Creacion(localizer)
34: get(localizer)
35: Reservation
36: get(customer)
37: Customer
38: get(localizer)
39: List<Extra>
40: Muestra Activity

16. Secuencia - Realizar una reserva

A continuación se realiza una explicación de cada mensaje del diagrama.

ID	Nombre del mensaje	Descripción
1	Inicia Aplicación	El usuario pulsa sobre el icono de la aplicación en su móvil y se inicia la misma. El ActivityManager de Android crea la HomeActivity, especificada como Activity principal en el manifest de la aplicación.
2	Muestra Activity	Se muestra la HomeActivity al usuario.
3	OnItemListClick()	El usuario selecciona la opción Nueva Reserva del menú de la aplicación.
4	Creación	Se crea el SearchFragment, que muestra el formulario de búsqueda de precios, que el usuario deberá rellenar para comenzar la búsqueda.
5	Muestra Activity	Se muestra el SearchFragment al usuario.
6	Inicia búsqueda de disponibilidad	El usuario ha rellenado el formulario y ha pulsado en Consultar Disponibilidad, para ver los vehículos disponibles.
7	Creación(parámetros)	Se crea el SearchResultFragment pasando los parámetros que el usuario ha indicado en el formulario.
8	execute(parámetros)	El SearchResultFragment crea una instancia de AvailabilityAsyncTask, y ejecuta su método execute pasándole los parámetros que ha indicado el usuario en el formulario.
9	availability(parámetros)	La instancia de AvailabilityAsyncTask crea una instancia del WebServiceController para conectar con el web Service de Canarias.com. Llama a su método availability pasando los parámetros indicados por el usuario.
10	availability(parámetros)	El controlador conecta mediante una petición HTTP con el web Service de Canarias.com y llama a su función availability, con los parámetros indicados por el usuario.
11	AvailabilityResponse : XML	El web Service de Canarias.com responde con una respuesta XML a la petición del controlador. La respuesta contiene los extras disponibles y la lista de vehículos disponibles.
12	AvailabilityResponse	El controlador parsea la respuesta XML en un objeto del tipo AvailabilityResponse, gracias a las anotaciones que se hacen sobre los atributos de la clase y a la librería SimpleXML que comentaremos más adelante.
13	setAdapter(new SearchResultAdapter (AvailabilityResponse.getCars()))	Desde AvailabilityAsyncTask se establece el adaptador de la lista que muestra los resultados de disponibilidad de vehículos, mediante la llamada al método setAdapter y pasándole los resultados de búsqueda.
14	Muestra Activity	Se muestra el SearchResultFragment al usuario con la lista de vehículos disponibles y su precio.
15	OnItemSelected	El usuario ha seleccionado un vehículo para reservar de la lista de resultados.
16	Creación(AvailabilityResponse.getExtras(),	Desde SearchResultFragment, se inicia el SelectExtrasFragment, pasándole el coche seleccionado mediante la propiedad availabilityIdentifier que se usará más tarde para realizar la reserva, y la lista de extras disponibles para reservar.

	availabilityIdentifier)	
17	Muestra Activity	Se muestra SelectExtrasFragment al usuario.
18	Seleccion Extras --> Continuar	El usuario selecciona los extras que desea reservar (si desea alguno) y pulsa en Continuar para avanzar en la reserva.
19	Creación(extras Codes, availabilityIdentifier)	Desde SelectExtrasFragment se crea una instancia de MakeBookingFragment, pasándole los extras seleccionados y el identificador de la disponibilidad seleccionada, necesario para finalizar la reserva.
20	Muestra Activity	Se muestra MakeBookingFragment al usuario.
21	Rellena Datos -> Confirmar Reserva	El usuario rellena los datos del formulario y pulsa en Confirmar Reserva.
22	execute(parámetros)	Se crea una instancia de MakeBookingAsyncTask, una tarea en segundo plano que llevará a cabo el proceso de reserva, y ejecuta su método execute, pasándole los parámetros indicados en el formulario así como los códigos de extras si los hay y el identificador de disponibilidad.
23	doReservation(parámetros)	Desde MakeBookingAsyncTask se crea una instancia del WebServiceController y se llama a su método doReservation pasándole los parámetros de la reserva.
24	make_reservation(parámetros)	El controlador envía una petición HTTP al web Service de Canarias.com ejecutando la función make_reservation y pasándole los parámetros de la reserva.
25	MakeReservationResponse : XML	El web Service responde con una respuesta XML que contiene los datos de confirmación de la reserva.
26	MakeReservationResponse	El controlador parsea la respuesta XML a un objeto MakeReservationResponse y lo devuelve a la tarea MakeBookingAsyncTask.
27	save(Customer)	A partir de los datos indicados por el usuario en el formulario, que se han almacenado temporalmente en la instancia de MakeBookingAsyncTask, se crea una instancia de Customer que se almacenará en base de datos. Se crea por tanto una instancia de CustomerDataSource, y se llama a su método save para almacenarlo en base de datos.
28	Result	El data source conecta con el DBHelper para almacenar el cliente en base de datos. Como se ha indicado antes, se omiten estos mensajes, ya explicados en el diagrama anterior, para mejorar la legibilidad del diagrama actual. Se devuelve el resultado de la operación a la instancia de MakeBookingAsyncTask.
29	save(Reservation)	A partir del objeto MakeReservationResponse, se crea el objeto Reservation que se pasará al data source para almacenarlo en base de datos. Se crea por tanto una instancia de ReservationDataSource y se llama a su método save pasándole el objeto Reservation.
30	Result	El data source conecta con el DBHelper para almacenar la reserva en base de datos. Se devuelve el resultado de la operación a la instancia de MakeBookingAsyncTask.
31	save(List<Extras>)	A partir del objeto MakeReservationResponse, se crea una lista de Extras, si los hay, que se pasará al data source para almacenarlo en base de datos. Se crea por tanto una instancia de ExtraDataSource y se llama a su método save pasándole la lista de objetos Extra.
32	Result	El data source conecta con el DBHelper para almacenar los extras en base de datos.

		Se devuelve el resultado de la operación a la instancia de MakeBookingAsyncTask.
33	Creacion(localizer)	Desde MakeBookingFragment, una vez terminada la operación de confirmación de reserva, se lanza el fragmento BookingDetailsFragment, pasándole el localizador de la reserva para obtenerla de la base de datos.
34	get(localizer)	Durante su creación, desde BookingDetailsFragment se crea una instancia de ReservationDataSource y se llama a su método get para obtener la reserva a partir de su localizador.
35	Reservation	El data source conecta con el DBHelper para obtener la reserva de la base de datos. Se devuelve la reserva a la instancia de BookingDetailsFragment.
36	get(customer)	Se crea una instancia de CustomerDataSource y se llama a su método get para obtener el cliente a partir de su email.
37	Customer	El data source conecta con el DBHelper para obtener el cliente de la base de datos. Se devuelve el cliente a la instancia de BookingDetailsFragment.
38	get(localizer)	Se crea una instancia de ExtraDataSource y se llama a su método get pasándole el localizador de la reserva para obtener los extras.
39	List<Extra>	El data source conecta con el DBHelper para obtener el listado de extras de la base de datos. Se devuelve la lista de extras a la instancia de BookingDetailsFragment.
40	Muestra Activity	Con toda esta información, se muestra BookingDetailsFragment al usuario con los detalles de la reserva, junto a un mensaje de confirmación de la realización de la reserva.

7.6.7 Prototipo

El Prototipo es una representación limitada de una aplicación de software, que permite a los Stakeholders probarlo y hacerse una idea de su forma y funcionalidad antes de comenzar su desarrollo. Esto permite poder detectar, en fases tempranas del ciclo de desarrollo, posibles fallos o problemas que puedan surgir con la idea que tienen los desarrolladores del producto, y de esta manera corregirlos antes de comenzar su desarrollo, ya que el coste de hacerlo después puede ser mucho mayor.

En este caso, hemos creado unos modelos de las pantallas de la aplicación, donde se podrá ver la estructura que tendrán así como la información que se muestra en cada una, todo de una manera aproximada, ya que en la fase posterior de desarrollo se pueden introducir algunos cambios o variaciones con respecto al prototipo. De esta forma, se podrán detectar algunos fallos iniciales así como hacerse una idea sobre qué información mostrará cada pantalla, la forma en que lo hará y la combinación de colores más adecuada para mejorar lo más posible la experiencia del usuario.

Para estructurar mejor la información, seguiremos los casos de uso indicados en el apartado 7.4 de esta memoria y agruparemos los prototipos correspondientes a cada caso de uso, además, se incluirán en primer lugar los prototipos de la pantalla principal.

7.6.7.1 Pantalla principal

En primer lugar, se presenta la pantalla principal de la aplicación, que es el punto de entrada a la misma.

17. Pantalla Principal

18. Navigation Drawer

La pantalla principal contará con un formato de parrilla (grid), en el que se situarán los puntos de entrada a los principales casos de uso de la aplicación.

Además, contará con una implementación del patrón de diseño de Android NavigationDrawer, que es un panel que entra desde el extremo izquierdo de la aplicación y ocupa aproximadamente dos tercios de la pantalla, dejando oscurecido el resto. Se usa generalmente para mostrar el menú principal, tal y como se puede ver en las capturas 17 y 18. Desde este menú, se puede acceder a todos los puntos de la aplicación.

7.6.7.2 Realizar una reserva

A continuación se presenta el prototipo para el caso de uso Realizar una reserva, separando las pantallas que lo conforman.

19. Formulario de búsqueda

20. Selector de oficinas

21. Selector de horas

22. Selector de fechas

Como se puede apreciar, en la captura 19 se muestra el formulario para realizar la búsqueda de precios. Se ha intentado dejar lo más simple y minimalista posible, con 4 pasos claros para realizar la búsqueda.

Al pulsar sobre cada sección se abren los diálogos de las capturas 20 a 22. La captura 20 muestra el diálogo selector de oficinas, que se lanza al pulsar sobre los botones Punto de Recogida y Punto de Devolución; la captura 21 muestra el selector de hora, que se muestra al pulsar sobre los botones Hora de Recogida y Hora de Devolución, y la captura 22 muestra el selector de fechas, que se lanza al pulsar sobre los botones Fecha de Recogida y Fecha de Devolución.

El botón Consultar Disponibilidad en la parte inferior es la llamada a la acción, que lanzará la búsqueda de disponibilidad de acuerdo a los parámetros introducidos.

23. Resultados de búsqueda

La pantalla 23 muestra los resultados de búsqueda, en los que se puede ver un resumen de la búsqueda realizada en la parte superior, así como un botón para volver al formulario anterior a modificar los parámetros de búsqueda.

En la parte inferior se muestra la lista de vehículos disponibles, junto con el precio del alquiler y las características de cada uno en forma de iconos representativos.

24. Selección de extras

25. Selección de extras - Colapsado

Seguidamente, tenemos la pantalla de selección de extras donde el usuario podrá añadir extras a su reserva.

En la parte superior se vuelve a mostrar un resumen del estado de la reserva hasta el momento. Al ocupar más espacio esta parte, se realizará dentro de un panel que ocultará o mostrará el resumen

de la reserva alternativamente al pulsar sobre él. Se puede ver expandido en la captura 24 y colapsado en la captura 25.

En la parte inferior se muestran los extras disponibles para reservar, su precio diario y un selector numérico para indicar la cantidad de cada extra que se desea.

26. Completar datos titular

27. Completar datos titular - Colapsado

A continuación pasamos a la pantalla en la que se rellenan los datos del titular de la reserva. Contiene, como las anteriores, un resumen de la reserva en la parte superior, dentro de un panel que se expandirá o colapsará al pulsar sobre él.

En la captura 26 se puede ver expandido, mostrando todo el resumen de la reserva hasta el momento. Mientras que en la captura 27 se puede ver colapsado, mostrando el formulario para rellenar los datos del titular.

El botón Confirmar Reserva en la parte inferior lanza la petición de confirmación de reserva al web Service para finalizar la misma.

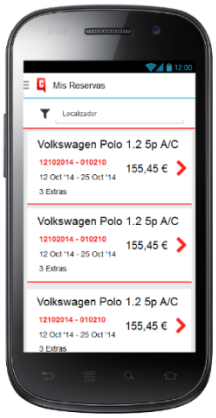
28. Resumen Reserva

Finalmente, se muestra el resumen de la reserva, donde se puede visualizar el vehículo reservado, los extras seleccionados, así como los puntos y fechas de recogida y devolución, junto con el resto de información de la reserva.

Además, se muestra un Tooltip (llamado Toast en Android) en la parte inferior para indicar que la reserva ha sido confirmada.

7.6.7.3 Ver detalle de una reserva, Cancelar una reserva, Modificar una reserva

Colocamos estos 3 casos de uso juntos porque las pantallas que los conforman son prácticamente las mismas en todos ellos.



29. Listado de reservas

30. Detalle de reserva

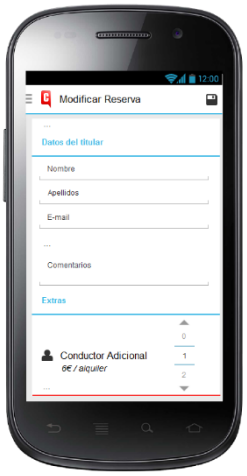
En primer lugar, en la captura 29 se puede ver el listado de reservas realizadas. Se muestra un resumen de la reserva así como un filtro en la parte superior para buscar una reserva por su localizador.

Al pulsar sobre una reserva, se accede al detalle de la misma, que se puede ver en la captura 30. Ahí se muestra la información completa de la reserva, así como los botones para cancelar y modificar la reserva situados en la parte superior derecha, en la barra de navegación.

31. Cancelar reserva

La acción de cancelar reserva se realiza en un diálogo que se muestra al pulsar el botón de cancelar en la barra de navegación. Se dan dos opciones, una para cancelar la reserva y otra para volver atrás y no hacer nada.

32. Actualizar Reserva Parte 1



33. Actualizar Reserva Parte 2

La acción de actualizar reserva se muestra en dos partes, ya que es una vista bastante alta y no cabe en una pantalla.

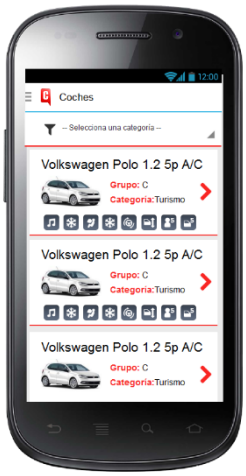
En la parte superior (captura 32) se muestran los puntos de recogida y devolución, así como las fechas y el vehículo reservado. Estos datos no podrán ser modificados en esta primera versión de la aplicación.

En la parte inferior (captura 33) se muestra un formulario donde se pueden modificar los datos del titular, excepto el email, así como los comentarios de la reserva. Además, se muestra la lista de extras reservados donde se podrá modificar la cantidad de cada uno.

Una vez modificados los datos, habrá que pulsar en el botón de Guardar situado en la barra de navegación para confirmar los cambios con el servicio web de Canarias.com.

7.6.7.4 Ver detalle de un coche

A continuación se muestran las pantallas para visualizar los coches disponibles y el detalle de cada uno.



34. Listado de coches

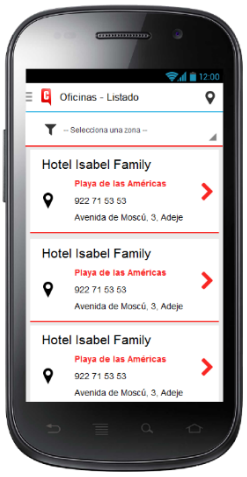
35. Detalle de un coche

En la pantalla 34 se puede apreciar el listado de coches disponibles. Para cada uno se muestra el modelo, una imagen, el grupo y la categoría, así como los extras disponibles en formato icono. Además, en la parte superior hay un filtro para ajustar la búsqueda por categoría.

Finalmente, en la pantalla 35 se puede apreciar el detalle de un coche, donde se puede ver toda la información del mismo y una imagen de mayor tamaño.

7.6.7.5 Ver detalle de una oficina

A continuación se muestran las capturas de pantalla correspondientes al caso de uso Ver el detalle de una oficina.



36. Listado de oficinas

37. Listado de oficinas en mapa

38. Detalle de una oficina

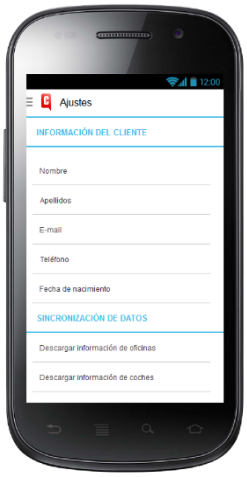
En la pantalla 36 se puede ver el listado de oficinas disponibles. Para cada una se muestra su nombre, la zona donde se encuentra así como su teléfono y dirección. Además, se incorpora un filtro en la parte superior para ajustar la búsqueda por zona.

En la pantalla 37 se muestran las oficinas en el mapa, desde donde se podrá visualizar la situación de las mismas así como acceder a su ficha de detalle pulsando en el icono de cada una.

Finalmente, en la captura 38 se muestra el detalle de la oficina. En la parte superior se puede ver el mapa con la localización, así como su nombre y el resto de información de la oficina. El mapa se podrá ampliar a una Activity dedicada donde se mostrará la localización de la oficina con mayor detalle.

7.6.7.6 Otras pantallas

A continuación se muestran otras pantallas de la aplicación, que no entran en ningún caso de uso específico del dominio.



39. Ajustes

40. Contacto

41. Ayuda

En la pantalla 39 se muestra la Activity de ajustes, que sigue las indicaciones de Google para generar configuraciones en las aplicaciones Android. En ella se pueden modificar los datos del titular de la reserva, que serán usados para precargar el formulario al realizar una reserva. En la parte inferior hay dos opciones para sincronizar los datos de coches y oficinas desde el servicio web.

En la pantalla 40 se muestra la Activity de contacto, desde donde se podrá contactar con Canarias.com directamente rellenando el formulario, a través de una llamada telefónica o visitando cualquiera de sus oficinas.

Finalmente, en la pantalla 41 se muestra la Activity de ayuda, donde se indicarán algunos consejos para sacar el máximo partido a la aplicación.

8. Implementación

A continuación se presentan las decisiones tomadas durante la fase de implementación y se detallan las características y técnicas más importantes que se han usado.

8.1 Premisas de la implementación

Cuando se desarrolla un proyecto para un dispositivo móvil, hay que tener en cuenta ciertos aspectos que difieren de los desarrollos para otras plataformas quizá más conocidas, como puede ser la web o los sistemas operativos para PC. A continuación se presentan algunas de las premisas que se han tomado en el proyecto para mejorar su funcionamiento en dispositivos móviles.

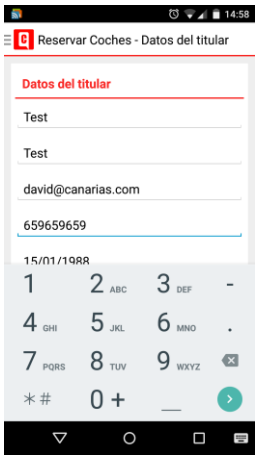
8.1.1 Minimizar los accesos a Internet

Se ha tratado de minimizar el acceso a Internet para el funcionamiento de la aplicación, con el motivo principal de mejorar el consumo de batería y minimizar el consumo de datos al usuario. En un paradigma Cliente-Servidor es inevitable tener que realizar alguna conexión a Internet cuando el servidor no se encuentra en la misma red local que el cliente. Pero existen técnicas como la caché de datos tanto en memoria como en base de datos que reducen estas conexiones necesarias y mejoran el rendimiento de la aplicación.

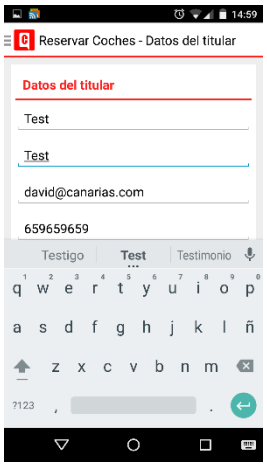
En nuestro caso, se ha usado la técnica de cacheo de información en una base de datos local de la aplicación para evitar las constantes conexiones al servicio web. Toda la información de los vehículos y las zonas de reserva se almacena en la base de datos local al iniciar la aplicación por primera vez, en lugar de consultar al servicio web cada vez que se necesita, luego se actualiza automáticamente cada cierto tiempo (por ejemplo cada 15 días). Cuando se necesita consultar la información, se va directamente a la base de datos local en lugar de consultar al servicio web, con el correspondiente ahorro de batería y consumo de datos.

8.1.2 Adecuación del teclado al contenido del campo del formulario

Para mejorar la usabilidad de la aplicación y facilitar al usuario la tarea de introducción de datos, se ha optimizado el tipo de teclado que aparece para cada campo, atendiendo a los caracteres que se espera que el usuario introduzca en dicho campo. Por ejemplo, para un campo numérico, aparece un teclado numérico, mientras que para un campo de texto, aparece el teclado de texto.



42. Teclado Numérico



43. Teclado Texto

8.1.3 Encriptación de datos

Los móviles están generalmente expuestos a comunicaciones inseguras, como pueden ser redes Wi-Fi públicas o con encriptaciones débiles (por ejemplo la encriptación WEP). Esto hace que los datos que se vayan a transmitir desde un dispositivo móvil puedan verse afectados si no se utilizan técnicas de encriptación para su transmisión.

En este caso, los datos a transmitir entre el dispositivo móvil y el servicio web no son excesivamente sensibles, salvo los datos del titular de la reserva. En cualquier caso, se usará el protocolo SSL para encriptar todas las comunicaciones entre el dispositivo móvil y el servicio web, ya que este último posee un certificado SSL que hace posible usar el protocolo HTTPS para realizar las peticiones, en lugar de HTTP.

Con esta simple mejora se consigue mejorar notablemente la seguridad en las comunicaciones, garantizando así la integridad y confidencialidad de los datos de extremo a extremo de la conexión.

8.1.4 Aplicación Multi-idioma

El SDK de Android permite fácilmente desarrollar una aplicación en múltiples idiomas. Para ello se hace uso de un fichero XML donde se almacenan las cadenas de texto que se usan en las interfaces gráficas.

En lugar de escribir el texto directamente en el control, lo que se hace es referenciar un string que se define en el fichero **strings.xml** situado en la carpeta **res/values** del proyecto. Ese fichero se traduce a múltiples idiomas, y se coloca en los diferentes directorios preparados para ello (**res/values-en para el inglés, res/values-fr para el francés**, etc.). De esta manera, el sistema operativo selecciona en tiempo de ejecución el fichero strings.xml correspondiente según el idioma que el usuario tenga configurado en el dispositivo móvil. Si el idioma no está disponible, se selecciona el fichero situado en la carpeta **res/values por defecto**.

8.1.5 Código comentado

Para facilitar la lectura y comprensión del código, se han añadido comentarios sobre cada método de las clases, explicando su funcionamiento. En algunos casos se han añadido pequeñas aclaraciones a nivel de líneas de código que ayudan a entender mejor esa sección.

Esto permitirá entender el funcionamiento del código desarrollado cuando se vuelva sobre él para realizar alguna modificación, además de facilitar su mantenimiento y la creación de nuevas versiones de la aplicación.

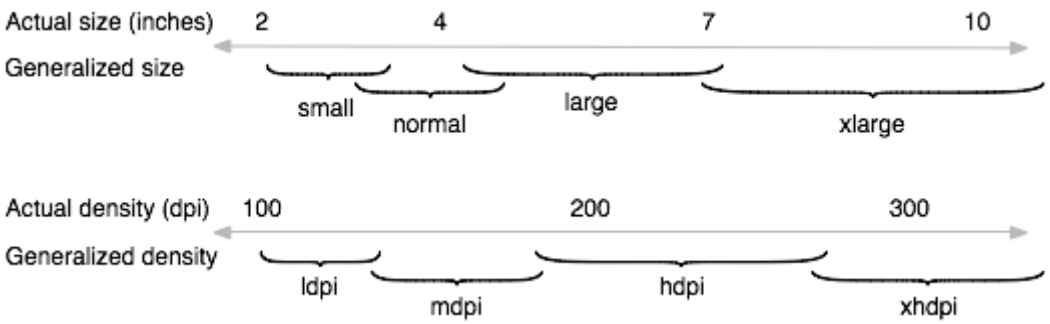
8.1.6 Optimización para distintos tamaños de pantalla

Android está presente en una amplia variedad de dispositivos, cada uno con un tamaño de pantalla y resolución diferente, es por eso que se dispone de un entorno de desarrollo de interfaces gráficas bastante flexible, que permite integrar en una misma aplicación diferentes versiones de una misma interfaz. El sistema operativo selecciona en tiempo de ejecución la versión de la interfaz más adecuada, en base a la configuración definida mediante la estructura de ficheros.

Al igual que se hacía con los strings multi-idioma, la fórmula para adecuar la aplicación a múltiples dispositivos viene dada por la estructura de ficheros que se cree dentro del directorio /res del proyecto. En este caso, los directorios a tener en cuenta serían res/layout, que almacena la definición de las interfaces, y res/drawable, que almacena las definiciones de objetos que se pueden usar en las interfaces, por ejemplo fondos, iconos, imágenes, etc.

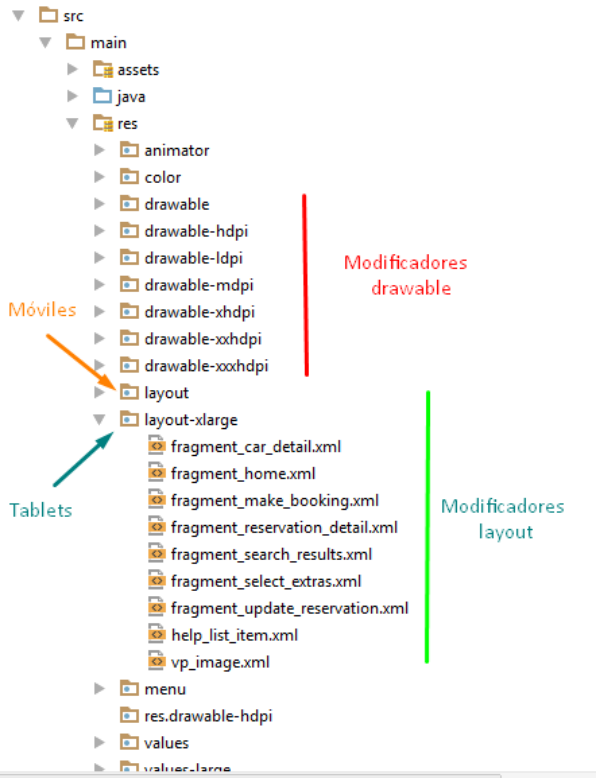
Los principales modificadores que se pueden aplicar a los directorios res/layout y res/drawable vienen definidos en base al tamaño de pantalla *(small, normal, large, xlarge)*, la densidad de píxeles por pulgada *(ldpi, mdpi, hdpi, xhdpi, xxhdpi…)*, la orientación *(land, port)* y el ratio de aspecto *(long, notlong)*.

A continuación se muestra una imagen que indica la relación entre las pulgadas de la pantalla de un dispositivo y el modificador a usar para que Android seleccione la interfaz correspondiente. En el segundo caso se muestra la relación entre las densidades de píxeles por pulgada y el modificador a usar.



44. Imagen de cómo Android mapea tamaños y densidades del dispositivo a tamaños y densidades generales (los datos son aproximados).

En este proyecto se ha optimizado la interfaz gráfica para adaptarla a smartphones y tablets, creando distintas versiones de algunas de las interfaces que se han colocado en los directorios correspondientes mediante el uso de los modificadores indicados anteriormente, de manera que Android selecciona una interfaz u otra dependiendo del dispositivo usado.



A continuación se muestra la parte del árbol del proyecto que lleva a cabo esta acción.

45. Modificadores para soportar múltiples pantallas

Como se puede apreciar, se añaden los modificadores a los directorios res/drawable y res/layout para soportar múltiples pantallas. De esta manera se optimizan las interfaces para múltiples dispositivos, y todo esto sin tener que modificar el código de la aplicación, lo cual permite maximizar la eficiencia y el trabajo a realizar, además de desacoplar completamente la interfaz de la implementación.

8.2 Implementación de la base de datos

El script de generación de la base de datos se ha escrito manualmente a partir del esquema relacional desarrollado en la fase de diseño del proyecto. Se han escrito los comandos de generación de las tablas con sus correspondientes restricciones, y se ha incluido en la implementación de la clase DBHelper en el proyecto.

Para cada tabla, primero se definen los campos que contiene, y a continuación se crea el comando de generación de la tabla.

```
//Definición Tabla Car
public static final String TABLE_CAR = "Car";
public static final String COLUMN_MODEL = "model";
public static final String COLUMN_GROUP = "car_group";
public static final String COLUMN_CATEGORY = "category";
public static final String COLUMN_IMAGEURL = "image_url";
public static final String COLUMN_SIPPCODE = "sipp_code";
//Script de creación Tabla Car
private static final String TABLE_CAR_CREATE = "create table "
            + TABLE_CAR + "(" + COLUMN_MODEL + " text primary key, "
            + COLUMN_GROUP + " text not null,"
            + COLUMN_CATEGORY + " text not null,"
            + COLUMN_IMAGEURL + " text not null,"
            + COLUMN_SIPPCODE + " text"
            + ");";
```

Se realiza lo mismo para el resto de tablas. Luego, en el método onCreate y onUpgrade se crea y actualiza la base de datos respectivamente, ejecutando los comandos definidos anteriormente.

```
//Crea la base de datos. Ejecutado la primera vez que se lanza la aplicación
@Override
public void onCreate(SQLiteDatabase db) {
      db.execSQL(TABLE_CAR_CREATE);
      db.execSQL(TABLE_ATTRIBUTE_CREATE);
      db.execSQL(TABLE_ZONE_CREATE);
      db.execSQL(TABLE_OFFICE_CREATE);
      db.execSQL(TABLE_CUSTOMER_CREATE);
      db.execSQL(TABLE_RESERVATION_CREATE);
      db.execSQL(TABLE_EXTRA_CREATE);
}

//Actualiza la base de datos. Ejecutado cada vez que se incrementa la versión.
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_CAR);
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_ATTRIBUTE);
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_ZONE);
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_OFFICE);
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_CUSTOMER);
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_RESERVATION);
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_EXTRA);
      onCreate(db);
}
```

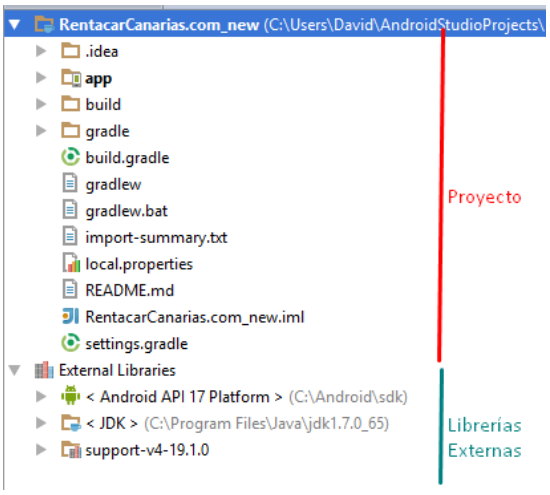
8.3 Implementación de la aplicación móvil

La implementación de la aplicación se ha seguido atendiendo a las recomendaciones de Google para el desarrollo Android, manteniendo el modelo por capas que se introdujo en la fase de diseño que ha permitido separar funcionalidades y mantener un código cohesionado y fácil de mantener.

8.3.1 Árbol del proyecto

Se ha estructurado el proyecto siguiendo la estructura básica de un proyecto creado mediante Android Studio, la cual ha cambiado respecto al desarrollo con Eclipse, ya que ahora se usa el Plugin Gradle[2] como sistema de compilación *(Build System)* del proyecto.

Gradle presenta numerosas ventajas como el uso del lenguaje DSL para describir y manipular la lógica de compilación, lo cual permite crear lógicas de construcción con mayor facilidad; una mayor flexibilidad con respecto a la integración de nuevos plugins, ya que estos pueden exponer su propia API para ser usada por los ficheros de compilación, y otras muchas mejoras.

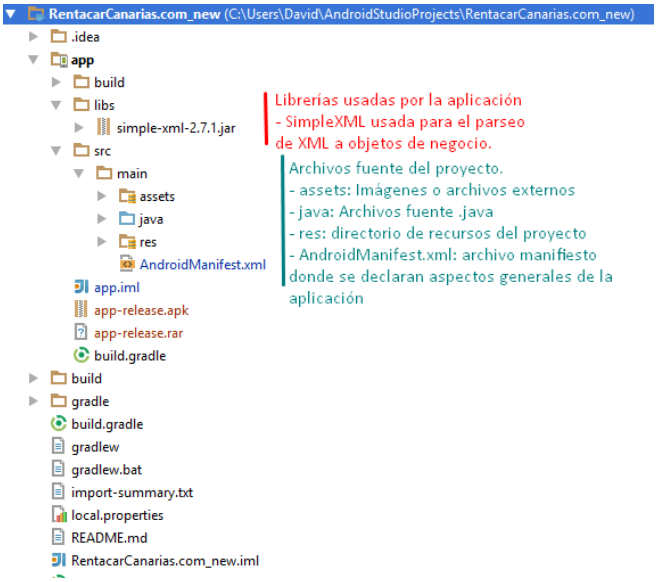


46. Estructura General

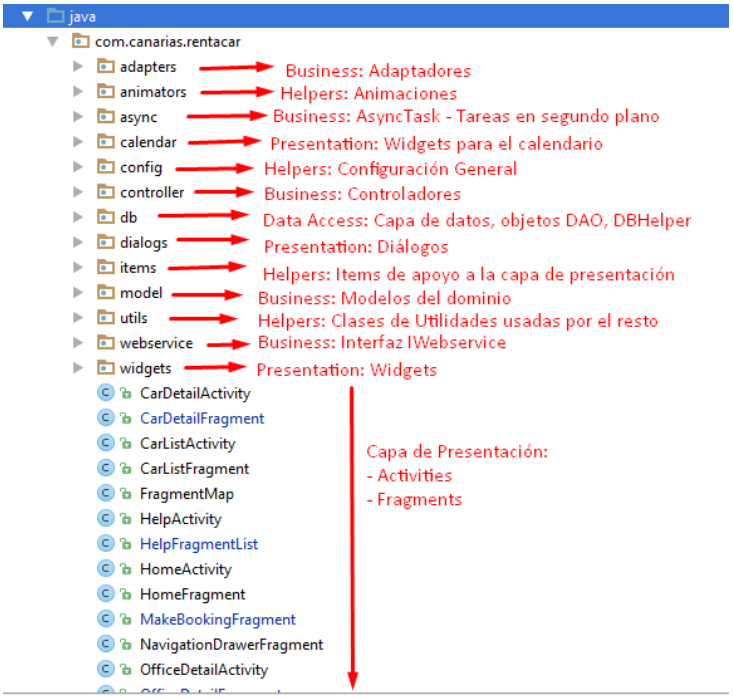
La estructura presente en la imagen 46 es ahora la estructura básica de los proyectos Android. Se divide en dos principales secciones, el Proyecto, que contiene los archivos del mismo, y las Librerías Externas usadas por el proyecto, en este caso el JDK de Java, el SDK de Android, y la librería de soporte usada para maximizar la compatibilidad con versiones anteriores de Android.

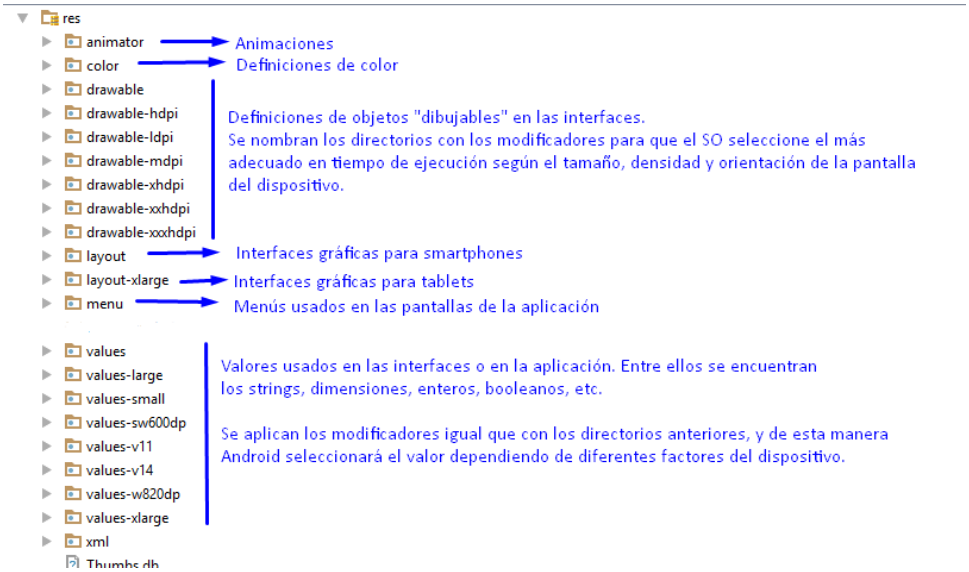
La carpeta principal, y donde se encuentra todo nuestro proyecto, es la carpeta app. A continuación se muestra su estructura y la explicación de cada sección.

[2] Plugin Gradle. (En línea). http://tools.android.com/tech-docs/new-build-system/user-guide#TOC-Introduction. (fecha de consulta: 03/12/2014)



A continuación mostramos el contenido de las carpetas java y res, las 2 principales del proyecto, para cada *package* se indica una breve explicación del contenido y la utilidad del mismo sobre la misma imagen. En primer lugar se indica la capa de arquitectura a la que pertenece, y luego se explica brevemente su utilidad. Hay algunos *packages* que se indican como *Helpers* porque contienen clases de apoyo que son usadas por el resto de capas libremente.





Finalmente, pasamos a explicar el contenido de la carpeta res, otra de las carpetas principales del proyecto:

8.3.2 Archivo AndroidManifest.xml

El archivo AndroidManifest.xml define los permisos que necesita la aplicación para ejecutarse en el dispositivo. También registra todas las Activities que se usan en la aplicación, así como algunos parámetros de configuración como puede ser el tema visual usado en cada Activity o la relación jerárquica entre las diferentes Activities.

Con respecto a los permisos, se usan los siguientes:

```
<!-- Acceso a Internet -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- Vibración del Teléfono -->
<uses-permission android:name="android.permission.VIBRATE" />
<!-- Localización -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<!-- OpenGL ES 2.0. para la API de Google Maps -->
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
<!-- Escribir en el almacenamiento externo -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!-- Leer el estado del teléfono -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<!-- Leer desde el almacenamiento externo -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<!-- Acceder al estado de la red -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- Leer de Google Services -->
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

8.3.3 Comunicación con el servicio web

Para gestionar la comunicación con el servicio web desde la aplicación, se ha hecho uso de la librería SimpleXML[3], que permite serializar y deserializar XML con un alto rendimiento y crear sistemas Java que se comuniquen con otros sistemas de cualquier tipo mediante XML.

El uso de esta librería es bastante sencillo, ya que en dos simples pasos se puede crear un sistema de serialización / deserialización de XML que nos permite obtener un código limpio y sencillo a la vez que útil para la tarea que queremos desarrollar.

En primer lugar, se etiquetan las propiedades de los modelos del dominio con anotaciones propias de la librería SimpleXML. Estas anotaciones indican a la librería cómo debe tratar cada propiedad:

```
@Element(name = "image")
private String imageUrl;
@Attribute(name = "Group")
private String group;
@ElementList(inline = true, entry = "attribute")
private ArrayList<CarAttribute> attributes;
```

En el ejemplo anterior, vemos como se anota el atributo imageUrl con la anotación *@Element* y el parámetro *name* con el valor *image*. Esto le indica a la librería que debe generar un elemento XML <image></image> a la hora de serializar el objeto. En el caso de la deserialización, deberá buscar el elemento <image></image> en el XML y colocar su valor en el atributo imageUrl del objeto.

En el segundo caso, la propiedad group se trata como un atributo, lo cual hará que al serializar se coloque como atributo, y al deserializar se busque el valor en los atributos del elemento XML.

En el último caso, @ElementList buscará una lista de elementos <attribute> en el XML y lo transformará en una Lista de objetos CarAttribute, que ya tienen también sus anotaciones correspondientes.

Como se puede ver, con unas simples anotaciones se ahorran muchas líneas de código y muchos errores que se producirían en caso de tener que serializar o deserializar el XML manualmente, a la vez que nos permite mantener un código limpio y entendible.

Para deserializar la respuesta XML que nos llega del servicio web, basta con usar unas simples líneas de código:

[3] SimpleXML. (En línea). http://simple.sourceforge.net/ (fecha de consulta: 03/12/2014)

```
Serializer serializer = new Persister();
try {
	CancelReservationResponse resp =
serializer.read(CancelReservationResponse.class, result);
	return resp;
} catch (Exception ex) {
	ErrorResponse error = serializer.read(ErrorResponse.class, result);
	return error;
}
```

Como vemos en el fragmento de código anterior, con estas simples líneas conseguimos deserializar la respuesta XML que nos llega del servicio web. En primer lugar se crea un objeto de tipo *Serializer*, y seguidamente se llama a su método *read* pasándole en primer lugar el tipo de objeto donde se van a deserializar los datos, y en segundo lugar el InputStream que contiene los datos.

Con estas simples líneas tendremos el objeto *resp* con todas sus propiedades rellenas con los datos de la respuesta XML, con lo cual ya podremos trabajar con él igual que con cualquier otro objeto java.

Por tanto ya tenemos completada la fase de comunicación con el servicio web, en la que se usará la librería SimpleXML para deserializar las respuestas XML en objetos Java y poder tratar con ellos.

8.3.4 Tareas en segundo plano

Las tareas en segundo plano tienen el objetivo de realizar operaciones de larga duración en un hilo de ejecución diferente al de la interfaz gráfica. Con esto se consigue que no se bloquee la interfaz durante la ejecución de dicha tarea, y se ofrece una mejor experiencia al usuario.

En este proyecto se hace un uso bastante importante de esta técnica, utilizando la clase parametrizable de Android AsyncTask. Esta clase ejecuta algunos de sus métodos en el hilo de la interfaz, y otros en un hilo aparte, que son los que realizan las operaciones de larga duración. A continuación vemos un ejemplo en la operación de búsqueda de disponibilidad de vehículos.

```
protected void onPreExecute() {
	rootView.findViewById(R.id.loadingLayout).setVisibility(
			View.VISIBLE);

}
```

El primer método ejecutado es onPreExecute(). Este método se ejecuta en el hilo de la interfaz gráfica, y es usado generalmente para indicar al usuario que se va a comenzar a realizar la tarea en segundo plano. En este caso, hacemos visible una parte de la interfaz que avisa al usuario que se está consultando la disponibilidad de vehículos.

```
@Override
protected List<SearchResult> doInBackground(Void... params) {
	// TODO Auto-generated method stub
	WebServiceController wsc = new WebServiceController();
	Response result = wsc.availability(
			this.params.get(Config.ARG_PICKUP_DATE),
			this.params.get(Config.ARG_PICKUP_TIME),
			this.params.get(Config.ARG_DROPOFF_DATE),
			this.params.get(Config.ARG_DROPOFF_TIME),
			this.params.get(Config.ARG_PICKUP_POINT),
			this.params.get(Config.ARG_DROPOFF_POINT));

	...
	...
}
```

El método doInBackground se ejecuta en un hilo distinto al de la interfaz gráfica, y es el encargado de realizar la tarea "pesada". En este caso como vemos, se crea una instancia del controlador del web Service y se llama a su método availability, pasando los parámetros de búsqueda. El resultado se almacena en la variable result y luego se opera con él para procesarlo (esta parte se ha omitido).

```
@Override
protected void onPostExecute(List<SearchResult> result) {

	if (result != null && result.size() > 0) {
		//Hay resultados
		...
	} else {
		//No hay resultados, mostrar mensaje
		...
	}
}
```

Finalmente, el método onPostExecute() recibe el resultado del doInBackground y actualiza la interfaz para informar al usuario del resultado de la operación en segundo plano. En este caso como vemos, se comprueba que hayan resultados y se muestran al usuario, y en caso de que no haya se muestra algún mensaje. Este método ya es libre de volver a actualizar la interfaz puesto que se ejecuta en su mismo hilo.

Este mismo paradigma es utilizado en todas las tareas en segundo plano que se usan en la aplicación.

8.3.5 Adaptadores Personalizados (Custom Adapters)

Los adaptadores son usados para popular los datos de una lista (ListView) en Android. Existen algunos tipos predefinidos en el SDK como BaseAdapter o ArrayAdapter, aunque en algunos casos no son suficientes para las necesidades requeridas por los casos de uso. En ese caso, se puede extender cualquiera de esas clases y modificar su comportamiento para adecuarse a las necesidades.

En el proyecto se hace uso de esta técnica extendiendo la clase parametrizable ArrayAdapter y sobrescribiendo su método getView(). Este método es llamado por el sistema operativo para obtener la interfaz que se aplicará a cada ítem de la lista, por tanto es el lugar para cargar nuestra interfaz personalizada y rellenar sus vistas con los valores de los objetos de negocio. A continuación se muestra un ejemplo de uno de estos adapters, en concreto el usado para mostrar el listado de vehículos. Se han omitido algunas partes menos importantes.

```
public class CarListAdapter extends ArrayAdapter<Car> {

    //Definición de propiedades
       ...

    //Constructor
       ...

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
         CarItemHolder drawerHolder;
        View view = convertView;
        if (view == null) {
             //Cargamos la interfaz por primera vez
            LayoutInflater inflater = ((Activity) context).getLayoutInflater();
            view = inflater.inflate(layoutResID, parent, false);
            ...
        } else {
             //Reciclamos la vista de otro item que ya no es visible
             //para evitar los errores OutOfMemory
            drawerHolder = (CarItemHolder) view.getTag();
        }
             //Obtenemos el objeto de la lista, según la posición
        Car dItem = this.filteredData.get(position);
             //Rellenamos las vistas de la interfaz con los valores del objeto
             ...
             //Retornamos la vista
        return view;
    }

    private static class CarItemHolder {
        TextView model;
        ImageView image;
        TextView category;
        TextView group;
        LinearLayout wrap;
    }
}
```

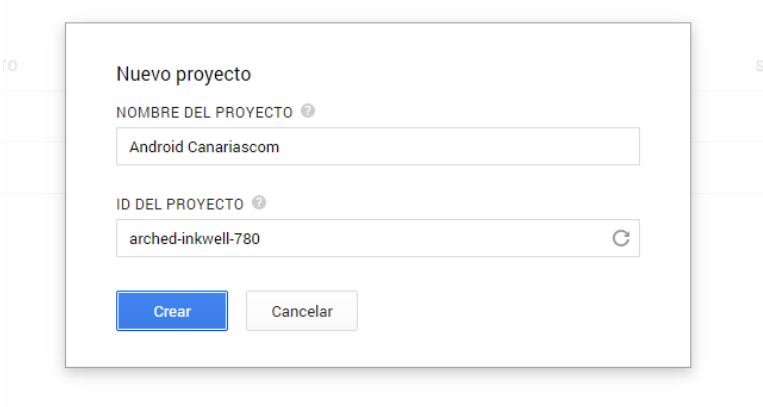
8.3.6 Utilización de Google Maps

El primer paso para poder usar Google Maps en Android es obtener una clave de la API de Google Maps en la consola de desarrolladores de Google. Para ello hay que seguir los siguientes pasos:

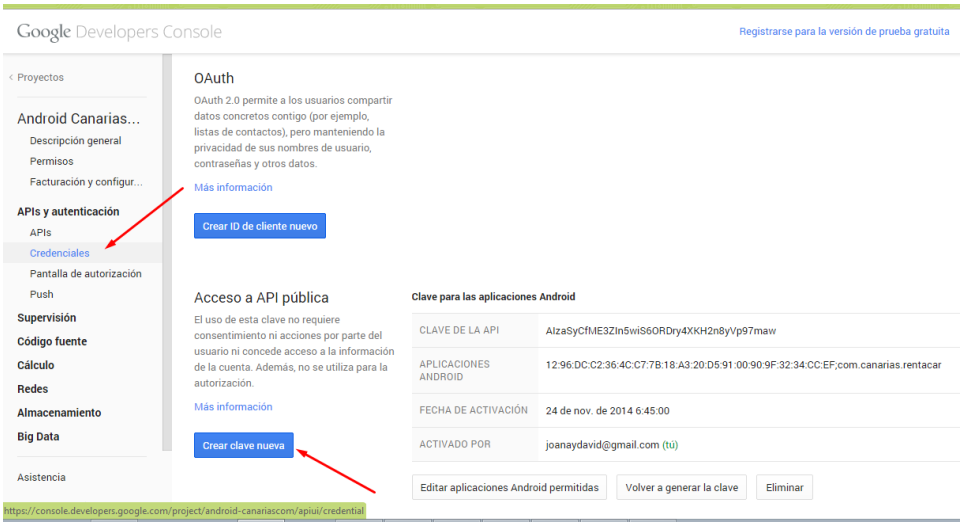
37. En primer lugar hay que crear una firma SHA-1 del almacén de llaves que se usará para generar el proyecto, en nuestro caso, como estamos aún en fase de desarrollo, usaremos el almacén de llaves debug o de depuración. Ejecutamos para ello el siguiente comando:
`keytool -exportcert -alias androiddebugkey -keystore debug.keystore -list -v`
Esto nos mostrará por pantalla la clave generada.

```
C:\Users\David\.android>keytool -exportcert -alias androiddebugkey -keystore deb
ug.keystore -list -v
Introduzca la contrase±a del almacún de claves:
Nombre de Alias: androiddebugkey
Fecha de Creaci¾n: 28-jul-2014
Tipo de Entrada: PrivateKeyEntry
Longitud de la Cadena de Certificado: 1
Certificado[1]:
Propietario: CN=Android Debug, O=Android, C=US
Emisor: CN=Android Debug, O=Android, C=US
N·mero de serie: 6e4c51c8
Vßlido desde: Mon Jul 28 22:02:01 BST 2014 hasta: Wed Jul 20 22:02:01 BST 2044
Huellas digitales del Certificado:
         MD5: A3:B6:4F:D3:41:18:FD:84:8D:0E:2C:FB:16:CC:2D:08
         SHA1: 47:A8:05:24:30:42:20:99:19:B6:A8:16:B8:C9:6A:55:BD:C8:C5:E4
         SHA256: 26:38:60:AC:23:EB:38:1C:20:0D:1F:7D:B3:E8:08:A3:93:6C:81:31:67:
2C:39:84:F4:06:E3:70:86:0C:B2:C6
         Nombre del Algoritmo de Firma: SHA256withRSA
         Versi¾n: 3

Extensiones:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: B8 15 90 1B 11 CF 9B CC   AC 9A CE 45 DD 54 20 A8  ...........E.T .
0010: 38 3A 2D 3F                                        8:-?
]
]
```

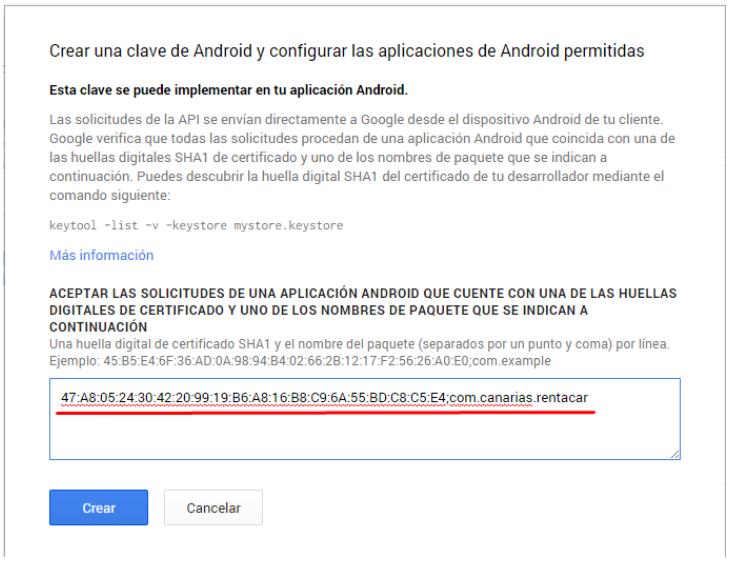
38. El siguiente paso es dirigirse a la consola de desarrolladores de Google https://console.developers.google.com y crear un Nuevo Proyecto.



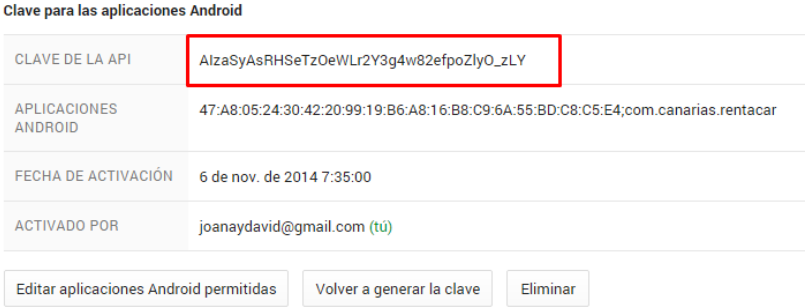
39. Seguidamente vamos al apartado Credentials y creamos una nueva clave de API pública.



40. Concatenamos la clave SHA-1 generada anteriormente y el nombre del package de nuestra aplicación, en este caso com.canarias.rentacar, con un punto y coma.



41. Pulsamos en Crear y tendremos generada nuestra clave de la API para usar en la aplicación.



Una vez generada la clave de la API, la colocamos en el AndroidManifest.xml para que el sistema la reconozca al cargar los mapas.

```
<meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyAsRHSeTzOeWLr2Y3g4w82efpoZlyO_zLY" />
```

Ahora estamos listos para usar los mapas en nuestra aplicación, lo cual se realiza mediante la vista MapView proporcionada en el SDK:

```
<com.google.android.gms.maps.MapView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/map" />
```

O también puede hacerse mediante el uso de MapFragment[4], un fragmento que carga un mapa directamente y te permite operar con él mediante el objeto GoogleMap.

8.3.7 Creando animaciones

Las animaciones son un nuevo concepto introducido en las últimas versiones de Android, pero que es compatible también con versiones anteriores gracias a las librerías de compatibilidad[5] desarrolladas por Google.

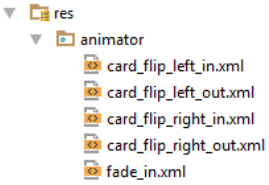
Se pueden implementar en XML y luego asignarlas en código, al igual que se hace con una interfaz a una Activity, o también se pueden implementar directamente en código. En nuestro caso, se ha implementado una en XML y algunas otras en código.

Las implementadas en código las he agrupado en una clase llamada AnimationHelper y se han creado como métodos estáticos que reciben por parámetro la vista a la que hay que aplicar la animación. Se muestra a continuación la animación *collapse*, que colapsa una vista modificando su altura hasta llegar a cero.

```
public static void collapse(final View v) {
        final int initialHeight = v.getMeasuredHeight();

        Animation a = new Animation() {
                @Override
                protected void applyTransformation(float interpolatedTime, Transformation t)
{
                        if (interpolatedTime == 1) {
                                v.setVisibility(View.GONE);
                        } else {
                                v.getLayoutParams().height = initialHeight - (int)
(initialHeight * interpolatedTime);
                                v.requestLayout();
                        }
```

[4] Android Reference (en línea) http://developer.android.com/reference/com/google/android/gms/maps/MapFragment.html (fecha de consulta (30/11/2014)
[5] Android Reference (en línea) https://developer.android.com/tools/support-library/features.html (fecha de consulta 30/11/2014)

```
                }

                @Override
                public boolean willChangeBounds() {
                        return true;
                }
        };

        // 1dp/ms
        a.setDuration((int) (initialHeight /
v.getContext().getResources().getDisplayMetrics().density));
        v.startAnimation(a);
}
```



Las implementadas mediante XML se localizan en la carpeta res/animator del proyecto. Son definidas ahí y referenciadas desde el código para asociarlas, por ejemplo, a una transición entre fragmentos, como es nuestro caso.

A continuación una de ellas, por ejemplo el **card_flip_left_in.xml.** Primero la definimos en XML:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Before rotating, immediately set the alpha to 0. -->
    <objectAnimator
        android:valueFrom="1.0"
        android:valueTo="0.0"
        android:propertyName="alpha"
        android:duration="0" />

    <!-- Rotate. -->
    <objectAnimator
        android:valueFrom="-180"
        android:valueTo="0"
        android:propertyName="rotationY"
        android:interpolator="@android:interpolator/accelerate_decelerate"
        android:duration="@integer/card_flip_time_full" />

    <!-- Half-way through the rotation (see startOffset), set the alpha to 1. -->
    <objectAnimator
        android:valueFrom="0.0"
        android:valueTo="1.0"
        android:propertyName="alpha"
        android:startOffset="@integer/card_flip_time_half"
        android:duration="1" />
</set>
```

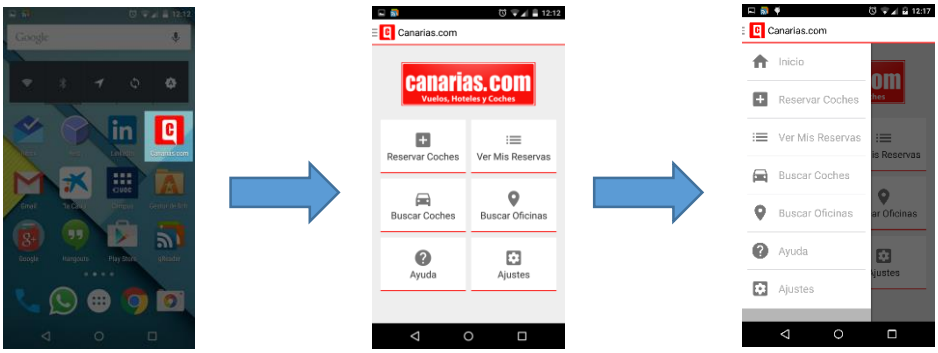
Y luego la referenciamos desde código para asignarla a una transición entre Fragments.

```
getFragmentManager().beginTransaction().setCustomAnimations(R.animator.card_flip_right_in,
R.animator.card_flip_right_out,R.animator.card_flip_left_in, R.animator.card_flip_left_out)
        .replace(R.id.reservation_detail_container,  fragment,UPDATE_FRAGMENT_TAG)
        .addToBackStack("Reservation_Detail").commit();
```

Con estas animaciones conseguimos un efecto más fluido en nuestra interfaz y una agradable experiencia de usuario.

8.4 Funcionamiento de la aplicación

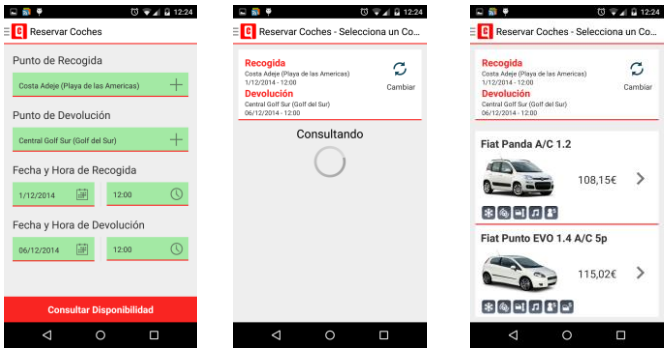
A continuación se mostrarán las capturas de la ejecución de la aplicación y algunos comentarios sobre ellas.



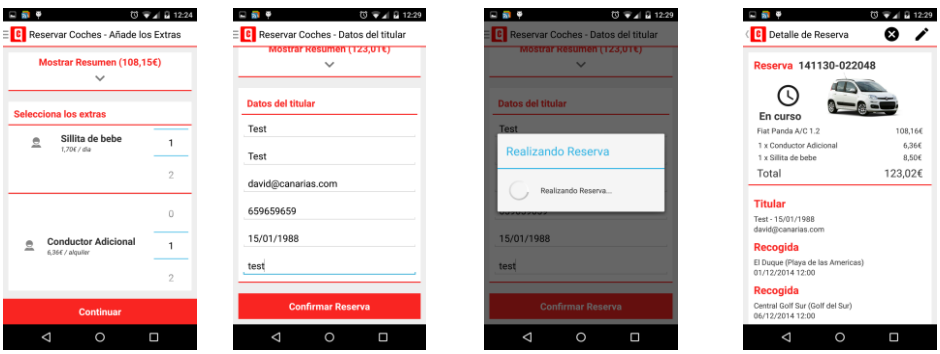
La pantalla principal muestra los puntos de entrada a todas las posibilidades que ofrece la aplicación, tanto desde la parrilla como desde el menú de navegación lateral.

8.4.1 Reservar un coche

A continuación vemos el proceso de reserva de un vehículo con todos sus pasos.



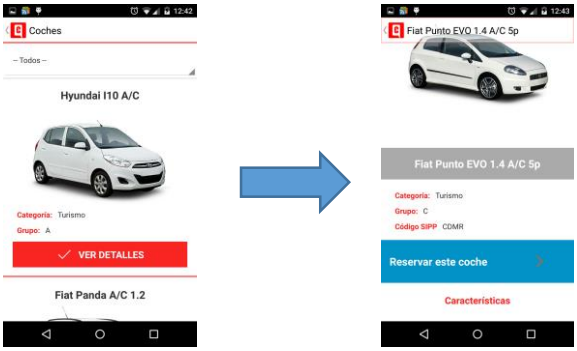
Seleccionamos los parámetros de búsqueda y el vehículo que vamos a reservar.



Añadimos los extras, rellenamos los datos del titular y confirmamos la reserva. Finalmente se muestra el detalle de la reserva con todos sus datos.

8.4.2 Buscar un coche

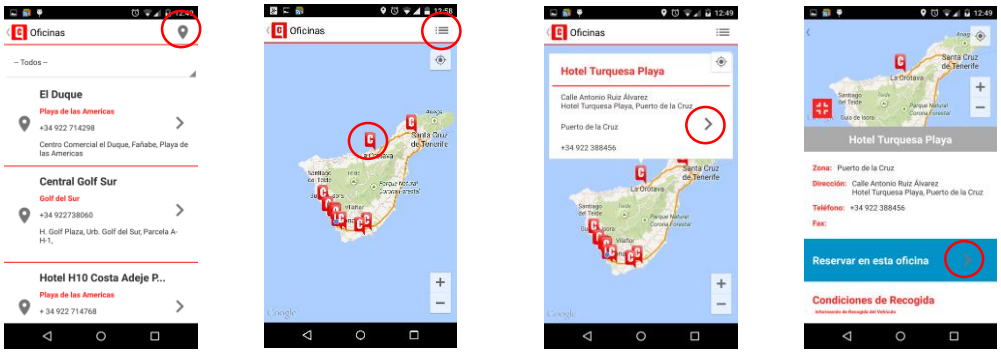
Para buscar un coche, pulsamos en la opción Buscar Coches de la pantalla principal.



Seleccionamos el vehículo de la lista de vehículos disponibles, pudiendo filtrar mediante el Spinner de la parte superior. Al seleccionar un vehículo se accede a su ficha de detalle donde se pueden ver sus características y reservarlo directamente.

8.4.3 Buscar una oficina

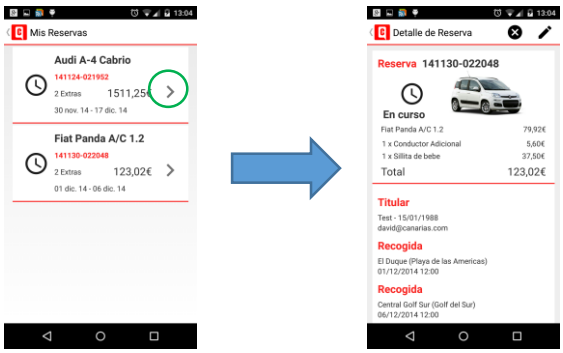
Accedemos al listado de oficinas pulsando en la opción Buscar Oficinas de la pantalla principal.



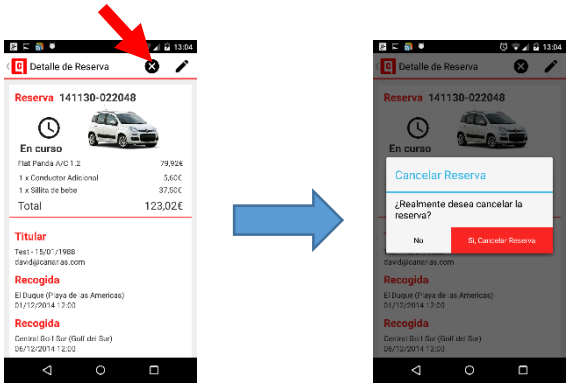
Podemos visualizar el listado de oficinas en el mapa o en formato lista, pulsando alternativamente en el botón situado en la barra superior. Si pulsamos sobre un marcador del mapa se nos abre la información de esa oficina. Pulsando sobre la flecha en el diálogo pasamos al detalle de la oficina, donde podemos ver el mapa en la parte superior y la información de la oficina a continuación.

8.4.4 Buscar, cancelar y modificar una reserva

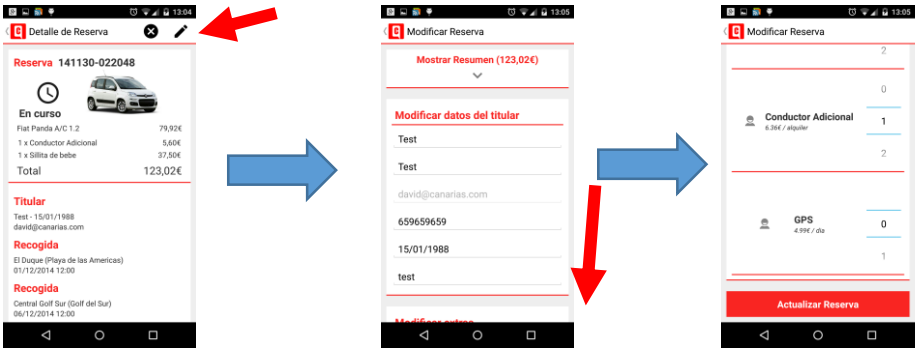
Agrupamos estas tres funcionalidades ya que la forma de llegar hasta ellas es muy parecida, y se utilizan las mismas pantallas. Para acceder pulsamos sobre el botón Ver Mis Reservas de la pantalla principal y se nos muestra el listado de reservas disponibles.



Desde el listado de reservas accedemos al detalle de cada una pulsando sobre la flecha.



Desde el detalle de reserva podemos cancelar la reserva pulsando el botón Cancelar situado en la barra superior. En el diálogo, confirmamos la cancelación pulsando en el botón rojo.

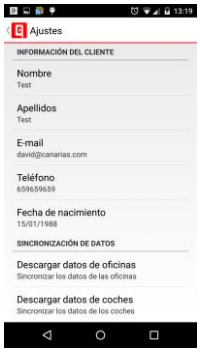


Para modificar la reserva pulsamos sobre el botón Modificar en la barra superior. Con esto accedemos a la pantalla de modificación, donde podemos modificar los datos del titular y la cantidad

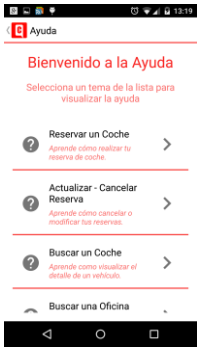
de extras reservados. Finalmente habrá que pulsar el botón Actualizar Reserva para confirmar los cambios.

8.4.5 Ajustes y Ayuda

Las secciones de Ajustes y Ayuda son accesibles desde la pantalla principal.



La sección de Ajustes permite configurar los datos del titular para cargarlos automáticamente al realizar una reserva, así como Actualizar la base de datos local con los vehículos y oficinas descargados desde el servicio web.



La sección de Ayuda contiene la explicación de cómo se debe realizar cada acción en la aplicación. Cada sección se presenta en formato de slide, en cada pantalla se muestra una imagen y un texto explicativo.

9. Conclusión y posibles mejoras

El desarrollo de esta aplicación para Android ha sido una tarea realmente excitante y apasionante. Llevaba ya tiempo deseando profundizar en el desarrollo para esta plataforma, pero por motivos de trabajo y estudios no había tenido el tiempo para ello. Soy desarrollador web en mi trabajo, y aunque algunos conceptos son parecidos, el desarrollo para Android tiene muchos aspectos que difieren bastante del desarrollo web, es una experiencia diferente que he adquirido con este proyecto y que me será muy útil para el futuro.

Se ha conseguido completar todos los objetivos propuestos en la planificación inicial, incluso se han podido tratar algunos puntos extras que no estaban previstos, como el trabajo con animaciones y efectos en las interfaces gráficas.

Una vez finalizada la exposición y análisis técnico del proyecto, paso a comentar las conclusiones extraídas de esta experiencia.

9.1 Consecución de objetivos

Los objetivos y requisitos propuestos para el proyecto se han completado en su totalidad. La aplicación permite realizar reservas de vehículos de forma totalmente nativa en la compañía Canarias.com localizada en Tenerife, usando su servicio web. Además, permite modificar y cancelar las reservas así como localizar oficinas en el mapa, o listar los vehículos disponibles para alquilar.

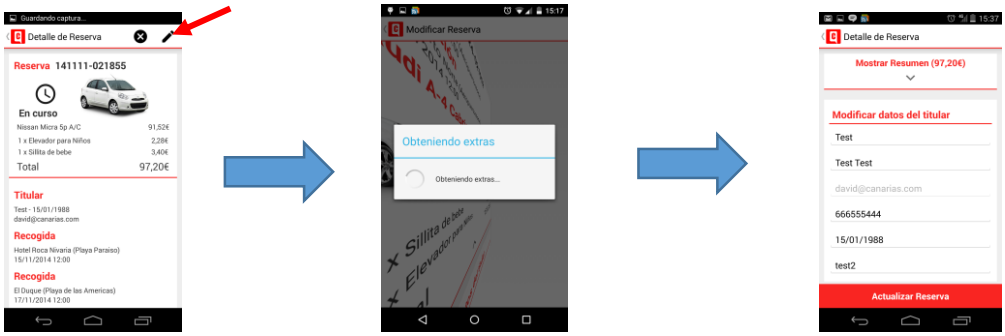
Se ha obtenido una aplicación con una interfaz de usuario agradable, y que avanza suavemente entre las distintas pantallas. Se ha intentado dejar lo más minimalista posible para no saturarla con muchos detalles. Solo se muestra lo necesario en cada pantalla, y se intenta guiar al usuario por el proceso de consecución de cada caso de uso.

Por tanto se puede afirmar que el producto está terminado según los requerimientos iniciales, y prácticamente listo para entregar a los usuarios. Quedan algunas ideas que se podrían implementar para darle un salto de calidad (no incluidas en los requisitos) pero que por falta de tiempo no se han podido implementar, y que comentaremos más adelante para tratar de implementar en un futuro.

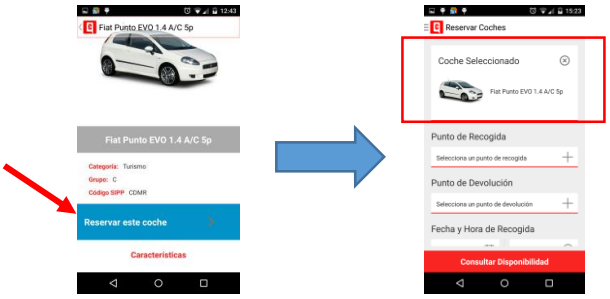
9.2 Variaciones del producto final respecto al diseño inicial previsto

El producto final tiene algunas variaciones con respecto a lo que se planificó en un principio, principalmente debido a que se han podido adelantar algunas tareas para las que se había asignado más tiempo, y en consecuencia se han podido añadir algunas funcionalidades adicionales:

1. **Sincronización periódica de vehículos y zonas**: Se ha añadido una funcionalidad que permite sincronizar automáticamente los vehículos y zonas de la base de datos local con los del servicio web cada 'X' días, siendo 'X' una cantidad configurada en código (actualmente 30 días). Esto permitirá que si se añade un nuevo vehículo u oficina en el sistema de Canarias.com, ésta sea descargada automáticamente en la siguiente sincronización.
   Esta acción también se puede realizar manualmente desde el apartado de Ajustes.
2. **Uso de animaciones en transiciones**: Se ha podido investigar un poco en el uso de animaciones tanto en las vistas como en las transiciones entre Fragments. Se ha creado un panel articulado para mostrar el resumen de la reserva durante el proceso de Nueva Reserva. Además, se ha añadido una transición entre Fragments al pasar de la pantalla de detalle de reserva a la pantalla de actualización de reserva, tal y como se muestra en las siguientes capturas:



3. **Accesos a reservar desde pantallas de detalle de vehículo y oficina:** Se ha preparado el fragmento del formulario de búsqueda (SearchFragment) para que reciba una oficina y/o vehículo predeterminado. Con esto se ha podido enlazar las pantallas de Detalle de Vehículo y Detalle de Oficina con la búsqueda, permitiendo realizar una búsqueda desde esa oficina o para ese vehículo en concreto.



9.3 Valoración personal

La experiencia extraída del desarrollo del proyecto es muy gratificante y enriquecedora. Me ha permitido profundizar en el desarrollo para Android realizando una aplicación de complejidad media-alta y usando algunas técnicas novedosas como el uso de animaciones.

Android es una plataforma en constante crecimiento y desarrollo, cada pocos meses hay nuevos cambios en el sistema, nuevas APIs para utilizar y nuevas formas de realizar las cosas. Esto hace que un desarrollador de Android tenga que estar en constante contacto con la comunidad, y atento a las últimas noticias sobre la plataforma, que le permitan mantener sus aplicaciones actualizadas con las últimas novedades del momento.

Precisamente, gracias a esa gran comunidad que hay en Internet sobre el mundo Android es posible que muchos desarrolladores, entre los que me incluyo, hayan podido realizar este tipo de aplicaciones y sobre todo superar los problemas que te vas encontrando en el camino y, que al buscarlos en los foros, siempre hay alguien al que ya le ha pasado y que ha comentado la solución. Creo que este aspecto es igual o más de importante que la propia documentación del SDK, y

agradezco realmente a toda esta gente que se preocupa de mantener y contribuir a este tipo de foros y comunidades.

Todo lo aprendido en este proyecto lo pondré en práctica en futuros proyectos, así como en las mejoras que tengo en mente para este. En el mundo de las reservas de vacaciones online, en el que ahora mismo trabajo, se están desarrollando muchas aplicaciones para dispositivos móviles, y esto hace que tengamos que estar al día en estos aspectos para poder ser competitivos. Por tanto estoy contento con el resultado del proyecto, y sobre todo con todos los conocimientos y experiencia que he podido adquirir.

9.4 Futuras mejoras

Todo proyecto tiene una fecha de límite, y por tanto hay que decidir hasta qué punto se llega para poder realizar correctamente la planificación del mismo. Esto hace que siempre queden cosas que mejorar o desarrollos para ampliar las capacidades del producto final. En este caso se han cumplido todos los objetivos propuestos, incluso se ha podido ampliar con algunos añadidos. Pero también han quedado algunos aspectos y nuevos desarrollos que podrían dar un plus a la aplicación.

1. **Impresión del comprobante de reserva**: Se podría implementar una conexión a una impresora de tickets (por ejemplo mediante Bluetooth), para permitir al usuario, si dispone de una, imprimir un comprobante de la reserva para entregar en la oficina al recoger el vehículo.
2. **Añadir nuevos proveedores de alquiler de coches:** La estructura de la aplicación nos permite añadir nuevos proveedores de alquiler de vehículos para permitir al usuario alquilarlos en el resto del mundo. Esto sería un desarrollo de complejidad media pero que daría un plus muy importante a la aplicación. Es un desarrollo muy probable ya que Canarias.com colabora con varios brokers de vehículos de nivel mundial, y por tanto se puede conectar con sus servicios web para ofrecer su flota a través de la aplicación.
3. **Añadir reservas de hoteles:** Al igual que para los vehículos, Canarias.com dispone de otro servicio web de reserva de hoteles totalmente online. Uno de los objetivos a medio plazo es realizar la conexión con este servicio web, e implementar un sistema de reserva de Hotel + Coche para permitir al usuario reservar una parte de sus vacaciones en forma de paquete.
4. **Sistema de notificaciones:** Debido al modelo de negocio de Canarias.com, algunas reservas de vehículos entran como pendientes de confirmación, y se confirman o cancelan en menos de 24 horas por un empleado, que previamente chequea la disponibilidad de flota. Se podría añadir un sistema de notificaciones que avise al usuario sobre los cambios de estado de las reservas que tiene pendientes de confirmación.

5. **Recogida de opiniones:** Otra opción interesante consistiría en añadir un sistema que permita recoger la opinión y valoración del servicio una vez finalizada la reserva de un vehículo. Se podría implementar a través del envío de un e-mail, o mejor aún, a través de una notificación (lo cual implicaría tener el punto 4 también) que le llevaría a una pantalla en la aplicación para introducir la valoración.
6. **Rediseño con Material Design:** En el proyecto actual se ha intentado aplicar algunas técnicas de diseño propuestas por Google en sus guías de diseño Material Design, pero por el corto tiempo de desarrollo que he tenido, no he podido aplicar todas las especificaciones que indica Google. Una buena mejora sería rediseñar las interfaces gráficas para seguir estas nuevas guías de diseño que le dan un aspecto mucho más elegante a la aplicación.

7. Glosario

ANDROID	Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, y también para relojes inteligentes, televisores y automóviles, inicialmente desarrollado por Android Inc., que Google respaldó económicamente y más tarde compró esta empresa en 2005.
GOOGLE PLAY	Google Play (anteriormente Android Market) es una plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android, así como una tienda en línea desarrollada y operada por Google. Ésta plataforma permite a los usuarios navegar y descargar aplicaciones (desarrolladas mediante Android SDK), música, libros, revistas y películas. También se pueden adquirir dispositivos móviles como ordenadores Chromebook, teléfonos inteligentes Nexus, Google Chromecast, entre otros.1
Patrón DAO (Data Access Object)	Un Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.
Evolus Pencil	Una herramienta para el diseño de prototipos de código libre y disponible para todas las plataformas. Fue construida con el propósito de proveer una herramienta sencilla y gratuita que los usuarios puedan instalar y usar fácilmente para crear prototipos sin necesidad de tener conocimientos de diseño gráfico ni programación.
Código SIPP	Código que resume las características de un vehículo. Son estándares de la industria para describir vehículos. Normalmente tienen 4 caracteres de longitud, por ejemplo: **CDMR**: Compact 4 door manual car with air conditioning Este código puede corresponder a varios modelos, como pueden ser: Ford Focus, Fiat Stilo, Hyundai Matrix.

8. Bibliografía

Android Developers. [En línea]. [Fecha de consulta: 20 de Septiembre de 2014] http://developer.android.com

Android reached record 85% smartphone market share in Q2 2014. [En línea]. [Fecha de consulta: 20 de Septiembre de 2014]. http://thenextweb.com/google/2014/07/31/android-reached-record-85-smartphone-market-share-q2-2014-report/

An Overview of the Android Architecture. [En línea]. [Fecha de consulta: 20 de Septiembre de 2014]. http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

Android Developers – Source Code. [En línea]. [Fecha de consulta: 20 de Septiembre de 2014]. http://source.android.com

Foros de Stackoverflow. [En línea] http://www.stackoverflow.com

Generating a Signed Release APK File in Android Studio. [En línea]. [Fecha de consulta: 2 de Diciembre de 2014]. http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio

Android working with Google Maps V2 and Custom Markers. [En línea]. [Fecha de consulta: 2 de Diciembre de 2014]. http://www.rogcg.com/blog/2014/04/20/android-working-with-google-maps-v2-and-custom-markers

Google Material Design Icons. [En línea]. [Fecha de consulta: 2 de Diciembre de 2014]. https://github.com/google/material-design-icons

Pushing the ActionBar to the Next Level. [En línea]. [Fecha de consulta: 2 de Diciembre de 2014]. http://cyrilmottier.com/2013/05/24/pushing-the-actionbar-to-the-next-level/

QuickReturnHeader. [En línea]. [Fecha de consulta: 9 Diciembre 2014]. https://github.com/ManuelPeinado/QuickReturnHeader