UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

Aplicación Android para la empresa Travelling-Service

Alfonso Gómez Matesanz

SEPTIEMBRE 2014

Aplicación Android para la empresa Travelling-Service

AUTOR : Alfonso Gómez Matesanz

TUTOR: Gonzalo Martínez Muñoz

Dpto. de Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Septiembre 2014

Resumen

Actualmente, resulta inconcebible un día a día sin teléfono móvil, y más en concreto, sin Smartphone. Debido al auge de estos dispositivos, cada vez más personas utilizan los múltiples servicios que ofrecen. Android es el sistema operativo para dispositivos móviles que ha experimentado mayor crecimiento en los últimos años. Por otro lado, la adaptación de los negocios al entorno web, es otro de los aspectos más presentes en la actualidad.

Por ello aunar estas fuerzas - internet y smartphones- en los negocios cobra cada vez más sentido. En esta dirección, este proyecto puede realizar una pequeña aportación. Travelling-Service es una agencia de contratación de viajes en línea, fundada en Octubre de 2012. Su página web permite reservar servicios turísticos como tours, hoteles, coches, restaurantes o vuelos.

El presente proyecto viene a cubrir un hueco vacío que se ha observado entre Travelling-Service y empresas similares en el mercado actual. La mayoría de estas empresas cuenta con una aplicación móvil aparte de un sitio web. Lo que diferencia a Travelling-Service de todas ellas es que un gran porcentaje de ventas llega directamente por llamada telefónica, como si de una agencia de viajes presencial se tratase. Por lo que una aplicación móvil se hace esencial con el objetivo de otorgar comodidad al usuario, y por consiguiente ingresos, a la empresa.

Este proyecto intenta ofrecer una aplicación potente y estable que permitirá tanto el contacto directo con la empresa de viajes, como la búsqueda y gestión de servicios relacionados con un posible viaje.

A lo largo del documento se presenta la aplicación realizada, exponiendo un estudio comparativo de aplicaciones similares existentes en el mercado, una descripción de características y requisitos principales, todas las tecnologías utilizadas en el desarrollo de la herramienta, tanto del lado cliente como del lado del servidor, así como un análisis que recorrerá todas las fases del proyecto desde el diseño hasta las pruebas. Por último, puede encontrarse un manual de uso para una mejor comprensión y más fácil utilización de la aplicación.

Abstract

At present times, it´s unbelievable day-to-day without mobile phone, and more to the point without a Smartphone. Due to the boom of these devices, more and more people use many facilities they offer. Android is the operating system for devices phones that has experienced a greater growth in recent years.

Furthermore, the adaptation of the web business environment is another of the most present aspects nowadays. For this reason joining these forces -internet and Smartphone- can gain more sense on business. Accordingly, this project can provide with a small contribution. Travelling-Service is a recruitment agency online travel, founded in October 2012. Their website allows people worldwide to book travel services like tours, hotels, cars, restaurants or flights.

This project fills a vacuum gap that has been observed between Travelling-Service and similar companies in the market today. Most of these companies has a separate mobile application beside their websites. The difference between Travelling-Service and the rest of similar applications is a large percentage of sales are mainly made through phone call, as a travel agency on site. So that a mobile application has become essential in order to provide comfort to the user, and therefore revenue for the company.

This actual project seeks to provide a powerful and stable mobile application which allows not only direct contact with the travel agency, but also searching and management services related to a potential trip.

The whole application is explained throughout the following document, providing a comparative study of similar applications found on the market, a description of key features and requirements, all the technologies that have been used in the development of the tool, both on client-side and server-side, and an analysis that will cover all the steps of the project, from design through tests. Finally a hand-book for better understanding and more friendly use of the application can be found.

Palabras clave

La siguiente lista contiene palabras descriptivas que podrán utilizarse para catalogar el proyecto.

- Android
- Aplicación
- Móvil
- Cliente
- Servidor
- Viajes
- Hoteles
- Coches
- Vuelos
- Turismo
- Travelling-Service

Keywords

The following list contains descriptive words which may be used to catalog the project.

- Android
- Application
- Mobile
- Client
- Server
- Trip
- Hotels
- Cars
- Flights
- Tourism
- Travelling-Service

Glosario

- **Smartphone:** Anglicismo. Teléfono móvil construido sobre una plataforma informática móvil, con capacidad para almacenar datos y realizar actividades asemejándose a una minicomputadora.
- **Android:** Sistema operativo basado en linux para dispositivos móviles.
- **Linux:** Combinación de kernel libe con sistema GNU.
- **GNU:** Sistema operativo desarrollado por GNU, formado en su totalidad por software libre.
- **Kernel:** Núcleo, software que constituye una parte fundamental un sistema operativo.
- **Actividad:** Cada una de las pantallas de la aplicación móvil.
- **Preferencia:** Mecanismo de almacenamiento más sencillo de Android en forma de partes clave-valor.
- **API:** Siglas de Interfaz de Programación de Aplicaciones. Es un conjunto de funciones, métodos o procedimientos que ofrece una biblioteca para ser utilizado por otro software como una capa de abstracción.
- **Servicio Web:** Tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
- **Recurso:** Datos (arrays, imágenes, cadenas, números, estilos) que se almacenan independientemente del código de aplicación para mejorar la organización del proyecto y permitir su adaptación a cambios.
- **Cliente:** Aplicación informática que consume un servicio remoto de otro ordenador conocido como servidor.
- **Servidor:** Nodo que forma parte de una red y provee de servicios a nodos denominados clientes.
- **Script:** Archivo de órdenes, archivo de procesamiento por lotes o guión es un programa usualmente simple, que por lo regular se almacena en archivos de texto plano.
- **Multiplataforma:** Válido para varios sistemas.

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1 | INTRODUCCIÓN

A lo largo de los últimos años, el avance de la tecnología para dispositivos móviles ha sido tal, que un gran número de acciones han sido absorbidas completamente por el uso de teléfonos inteligentes e internet. Desde la comunicación instantánea entre dos personas, hasta la compra de algún producto por internet. A diario salen a la luz nuevas aplicaciones, cada vez más fáciles y cómodas, que nos hacen posible la realización de tareas cotidianas desde la palma de nuestra de mano.

Es indudable que las ventajas superan los inconvenientes. Este tipo de tecnología ha ayudado a mejorar muchos aspectos de la vida cotidiana, por ejemplo, el acceso a la información, el almacenamiento de datos personales de forma segura o la compra de manera inmediata, sencilla y desde cualquier lugar.

1.1 OBJETIVOS

El desarrollo de una aplicación móvil para Travelling-Service viene motivado por la inminente necesidad que tiene la empresa de situarse en dicho mercado de aplicaciones, donde casi la totalidad de empresas del sector cuentan con una.

Atendiendo a las ventajas de las aplicaciones móviles centradas en el sector de viajes, parece casi obligatorio la creación de una aplicación para este negocio. Proporcionaría un nuevo canal de venta, un producto nuevo y diferente a modo de escaparate, además de generar una nueva imagen de marca aprovechando la tecnología, diferenciando y empujando el negocio a un bajo coste.

Para ello se ha realizado un estudio de las tecnologías móviles disponibles en la actualidad, donde se ha encontrado entre ellas una que encaja con el objetivo del proyecto debido principalmente a su universalidad. Según un estudio [1] el 80% de los teléfonos móviles son *smartphones.* De ese alto porcentaje, más de otro 80% lleva Android como sistema operativo, seguido por IOS con un 13%, y en tercer lugar, con un 3% se encuentra Windows Phone, como puede verse en la figura 1. Además, en España, donde reside Travelling-Service, la implantación de Android supera ya el 90%.

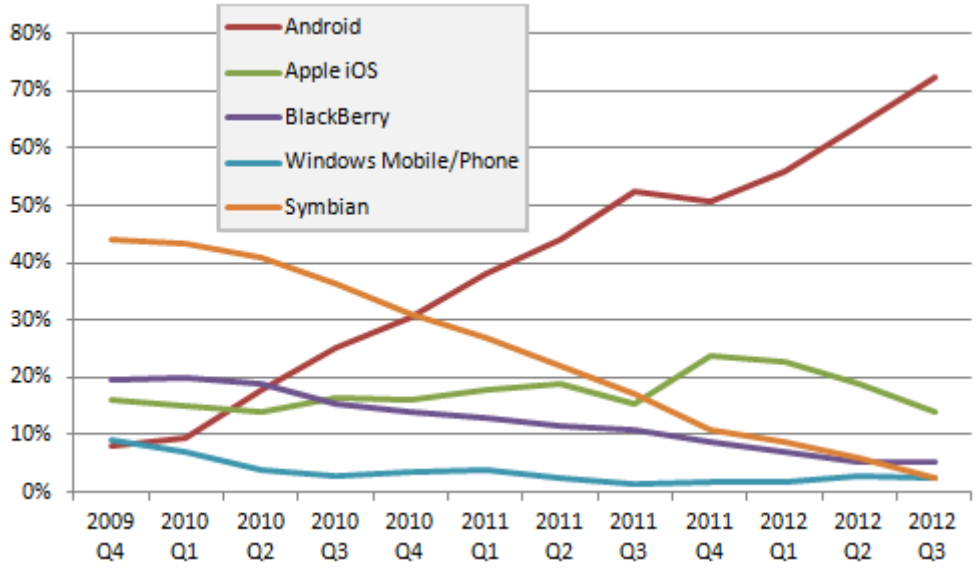

ILUSTRACIÓN 1 : MERCADO DE SISTEMAS OPERATIVOS EN SMARTPHONES [31]

Otro estudio [2] indica que Android también es líder en el número de aplicaciones descargadas por sistema operativo, donde se encuentra por encima del 85%, frente a un 34% de IOS, un 4% de Windows Phone, y un 3% de RIM, como puede observarse en la siguiente figura.

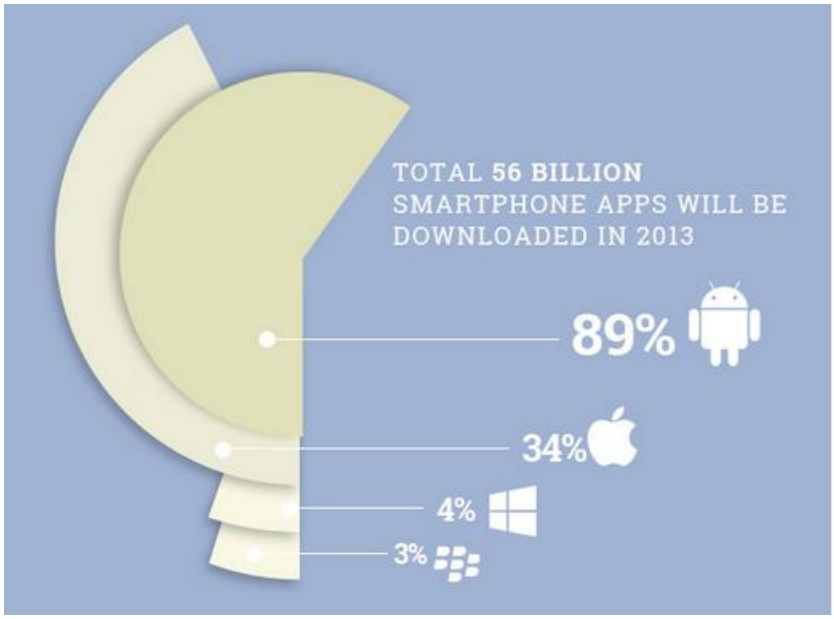

ILUSTRACIÓN 2 : APLICACIONES DESCARGADAS POR SISTEMA OPERATIVO [2]

Debido al gran apoyo que recibe por parte del mercado internacional, así como por ser de desarrollo libre, unido a todo lo recogido anteriormente en este apartado, Android parece el candidato perfecto para este proyecto.

1.2 OBJETIVOS DEL PROYECTO

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación para dispositivo móvil en Android para la gestión y reserva de servicios relacionados con viajes, donde los usuarios podrán registrarse, consultar y reservar aspectos relacionados con la gestión de uno o varios viajes, así como consultar datos relevantes sobre la empresa, con el fin de aumentar las ventas de la misma.

Otro de los principales objetivos de un producto de estas características es la forma de llegar al usuario, por lo que una interfaz amigable y la facilidad de uso se hacen indispensables en este proyecto.

La robustez es otro de los objetivos de cualquier sistema software, a través de la cual se mide, en su mayoría, la calidad del producto.

El último de los objetivos a destacar es el reto que supone llevar a cabo un proyecto software desde cero, planificando todas sus etapas, desde el estudio y análisis, hasta la codificación y las pruebas necesarias. Siendo necesaria asimismo una planificación detallada del tiempo y los recursos, así como la puesta en práctica de las capacidades y conocimientos adquiridos a lo largo de toda la carrera.

1.3 FASES DE REALIZACIÓN

En este apartado se da una visión general de las fases seguidas a lo largo de la realización del proyecto, que son las correspondientes a la mayoría de un desarrollo software.

- **Estudio:** Investigación sobre las posibles tecnologías a utilizar, posible estudio de mercado y de viabilidad. Indagación y pruebas sobre aplicaciones relacionadas, con el objetivo de extraer lo que se suele utilizar en proyectos similares.

- **Análisis:** Definición de requisitos con el objetivo de dilucidar exactamente qué es lo que se desea construir.

- **Diseño:** Tras finalizar las fases de estudio y análisis, y con el listado de requisitos bien definido, se procede a diseñar la arquitectura de la aplicación.

Esto incluye diversos diagramas que definirán el diseño de pantallas, interfaz, base de datos y seguridad.

- **Implementación:** Programación de la aplicación móvil en Java.

- **Pruebas:** Una vez finalizadas todas las fases anteriores, se hace necesario la realización de una fase de pruebas generales para comprobar la calidad y efectividad del software desarrollado.

- **Documentación:** Según una guía sobre desarrollo software [3], la documentación es algo indispensable para el correcto mantenimiento de cualquier producto software. Tanto a nivel de código como de documento, con el fin de dejar constancia por escrito de las fases llevadas a cabo en el ciclo de vida del producto software.

- **Mantenimiento:** La guía de principios de desarrollo software de Microsoft Ibérica [4] afirma que el mantenimiento de un producto software supone el 80% del ciclo de vida. En este proyecto puede no tener mucha cabida, pero se orienta a un trabajo futuro con bastante proyección con el fin de mejorar y ampliar la aplicación.

1.4 Estructura del documento

El presente documento recoge todas las fases mencionadas en el apartado anterior. Así como posibles anexos con información relevante sobre el proyecto. Recorrerá todas las fases del proyecto desde el diseño hasta las pruebas, y por último un manual de uso.

Además, todas las tecnologías, herramientas y lenguajes utilizados en el desarrollo de la aplicación, tanto del lado del cliente como del lado del servidor, se explican para una mejor comprensión del proyecto.

En el siguiente capítulo se realiza un estudio del estado del arte, que contiene una breve descripción del sistema operativo Android junto a sus aspectos más relevantes, y una comparación de aplicaciones similares existentes en el mercado.

En el capítulo 3 - Análisis se documenta la fase de análisis, definición del proyecto, metodologías de desarrollo, descripción de características y listado de requisitos.

A continuación, en el capítulo 4 - Diseño, se ilustra el diseño de la aplicación, incluyendo arquitectura del sistema, diseño de interfaces, diseño de base de datos, de servidor y de seguridad.

El capítulo 5 - Implementación explica la fase de implementación de aplicación, tanto el lado de cliente como el lado de servidor, en los lenguajes elegidos. Así como las librerías utilizadas y técnicas de desarrollo.

El capítulo 6 - Pruebas enumera las pruebas que se han realizado tras las fases anteriores.

Un penúltimo capítulo que manifiesta unas conclusiones sobre el trabajo realizado.

Finalmente, un capítulo que propone un posible trabajo futuro con el objetivo de plantear mejoras una vez finalizado este Trabajo de Fin de Grado.

2 | ESTADO DEL ARTE

2.1 ANDROID

Android es un sistema operativo basado en el *kernel* de Linux y diseñado principalmente para dispositivos móviles con pantalla táctil, tales como *smartphones* o tabletas. Inicialmente desarrollado por Android Inc, y respaldado económicamente por Google, que más tarde, en el año 2005 adquirió la empresa.

Uno de los aspectos fundamentales del sistema operativo de Android fue su orientación a la multiplataforma, algo realmente novedoso, debido a que hace unos años, un sistema operativo se asociaba a un único dispositivo. Rápidamente esta característica hizo que Android alcanzara sus objetivos, convirtiéndose en el sistema operativo más utilizado.

2.1.1 ESTRUCTURA DE ANDROID

Como puede apreciarse en la figura 3, Android se encuentra estructurado en varias capas.

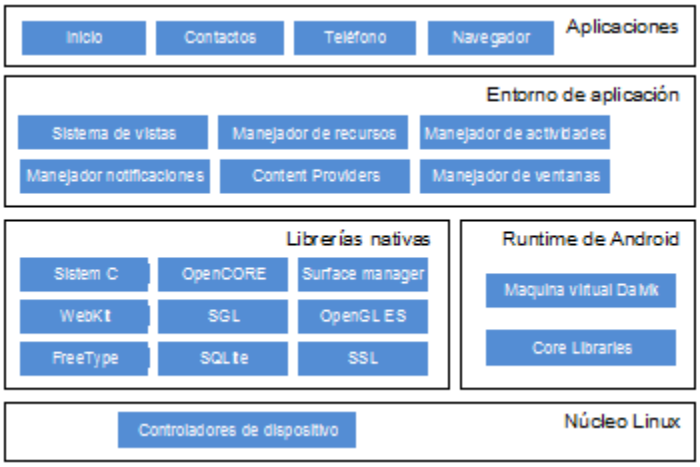

ILUSTRACIÓN 3 - ARQUITECTURA DE ANDROID [5]

Si se parte de una vista ascendente, lo primero que se encuentra es el **núcleo de Android**. Basado en el *kernel* del sistema operativo Linux versión 2.6. Esta capa proporciona servicios de seguridad, manejo de memoria, multiproceso y soporte para controladores de dispositivo.

A continuación se localiza el **entorno de ejecución** (*Android Runtime),* basado en la máquina virtual de Java. Dadas las limitaciones de los dispositivos Google decidió crear la máquina virtual *Dalvik*, la cual respondió mucho mejor a dichas limitaciones. Entre sus características destacan, la optimización de recursos debido a la ejecución de ficheros *Dalvik* ejecutables (.dex) y la delegación al *kernel* de Linux procesos como el *threading* y el manejo de memoria a bajo nivel.

Al mismo nivel se ubican las **librerías nativas**, escritas en C/C++ y compiladas en código nativo del procesador. Muchas librerías utilizan código abierto. Entre ellas se destacan:

- **System C Library:** adaptada para dispositivos embebidos en Linux.
- **WebKit:** Soporta el navegador web de Android y su vista webview. Misma librería que Google Chrome y Safari de Apple.
- **SQLite:** Ligero pero potente motor de bases de datos relacionales.
- **SSL:** Proporciona servicios de encriptación (*Secure Socket Layer*)

El **entorno de aplicación** (*Application Framework*) ofrece una plataforma de desarrollo libre para aplicaciones, donde su principal riqueza reside en la reutilización de componentes. Componentes desarrollados por Google, o por usuarios. Los servicios más importantes son:

- ***Views:*** conjunto de vistas.
- ***Location Manager:*** proporciona servicios de localización a aplicaciones.
- ***Activity Manager*****:** manejador del ciclo de vida de las actividades.
- ***Notification Manager:*** permite mostrar notificaciones a las actividades.
- ***Content Provider:*** concede accesos a datos de otras aplicaciones

Por último se encuentra **la capa de aplicaciones** (*Applications*) formada por el conjunto de aplicaciones del dispositivo, ya sean nativas o instaladas por el usuario. Todas ellas deben correr en la máquina virtual *Dalvik* para garantizar la seguridad del sistema. La mayoría de ellas escritas en Java.

2.1.2 VERSIONES DEL SISTEMA OPERATIVO

Android no ha parado de evolucionar desde el lanzamiento de su primera versión. En Febrero de 2009 Google lanza su primera actualización 1.1 que se diferenciaba de la primera en poder adjuntar archivos en los mensajes. Si echáramos la vista atrás y lo

comparásemos con una versión actual, comprenderíamos la inmensa evolución del sistema operativo. Todas las actualizaciones siguientes incorporan funcionalidades nuevas o mejoran las ya existentes.
Todas las versiones de Android reciben del inglés el nombre de diferentes postres, siguiendo, además, orden alfabético.

- **A: Apple Pie** (v1.0): tarta de manzana.
- **B: Banana Bread** (v1.1): pan de plátano.
- **C: Cupcake** (v1.5): panqué.
- **D: Donut** (v1.6): rosquilla.
- **E: Éclair** (v2.0/v2.1): pastel francés.
- **F: Froyo** (v2.2) (abreviatura de «*frozen yogurt*»): yogur helado.
- **G: Gingerbread** (v2.3): pan de jengibre.
- **H: Honeycomb** (v3.0/v3.1/v3.2): panal de miel.
- **I: Ice Cream Sandwich** (v4.0): emparedado de helado.
- **J: Jelly Bean** (v4.1/v4.2/v4.3): gominola.
- **K: KitKat** (v4.4): tableta de chocolate con leche**.**

Actualmente conviven muchas de las versiones mencionadas en el mercado, debido a que muchos terminales no se han actualizado a nuevas versiones. En la siguiente tabla puede observarse el porcentaje de terminales que usan las diferentes versiones.

Versión	Nombre	Distribución
2.2	**Froyo**	**1.3%**
2.3 - 2.3.7	**GingerBread**	**21.2%**
3.2	**Honeycomb**	**0.1%**
4.0-4.04	**Ice Cream Sandwich**	**16.9%**
4.1-4.3	**Jelly Bean**	**59.1%**
4.4	**Kit-Kat**	**1.4%**

TABLA 1 - TABLA DE VERSIONES

O a modo de gráfico [6] como puede verse en la siguiente figura.

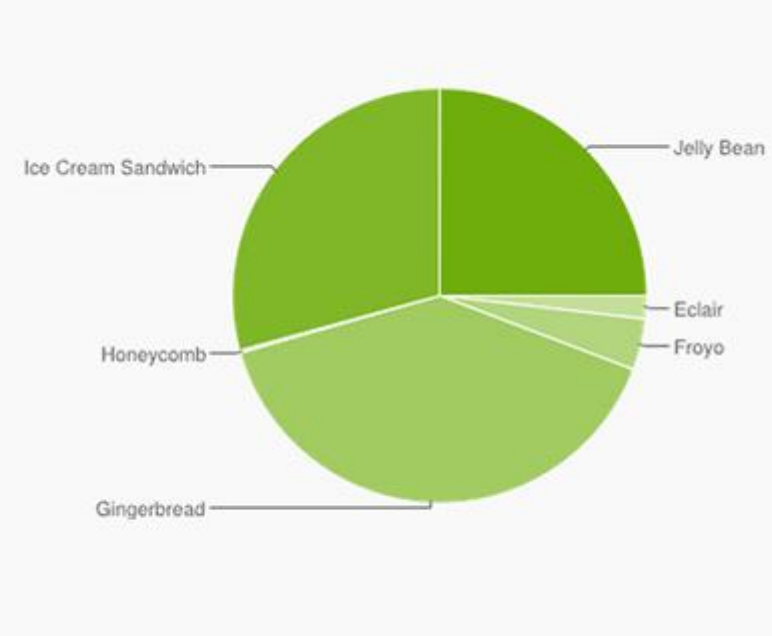

ILUSTRACIÓN 4- GRÁFICO DE REPARTO DE VERSIONES [6]

2.2 ENFOQUES EXISTENTES

En el ámbito de las aplicaciones dedicadas a gestión y búsqueda de viajes, pueden encontrarse dos vertientes diferenciadas, una de ellas dedicada a la búsqueda de información, opiniones de otros usuarios, con un detalle más amplio de los lugares buscados, y otro enfoque más orientado a la reserva directa de los servicios relacionados con viajes.

A continuación se muestran varias capturas de pantalla por cada aplicación, en concreto tres de ellas. Siendo la primera una captura de la pantalla principal o menú lateral principal, la segunda, un listado de servicios filtrados, y por último, un detalle de uno concreto.

Sobre el primer enfoque, el máximo exponente es ***TouristEye*** [7].

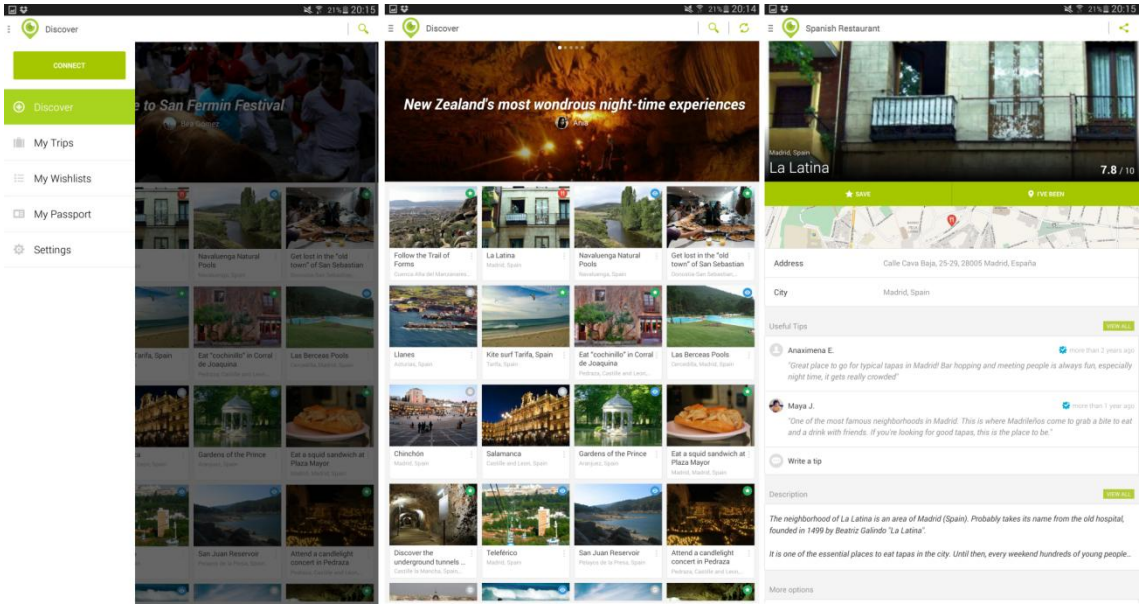

ILUSTRACIÓN 5 - TOURISTEYE

TouristEye es una aplicación multiplataforma cuyo objetivo principal es ofrecer información relevante al usuario acerca de todo lo relacionado sobre un destino turístico concreto. Ideal para coleccionar ideas y planificar posibles viajes.

En la otra vertiente podemos encontrar dos aplicaciones.

La primera de ellas es ***TripAdvisor*** [8].

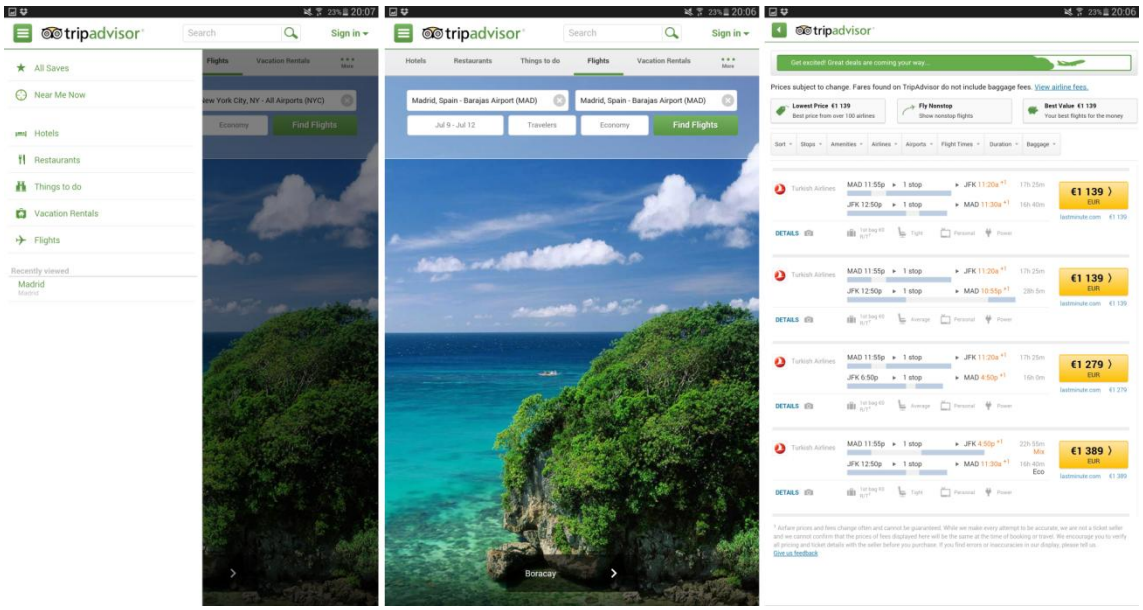

ILUSTRACIÓN 6 - TRIPADVISOR [8]

Por último, cabe destacar ***Kayak*** [9].

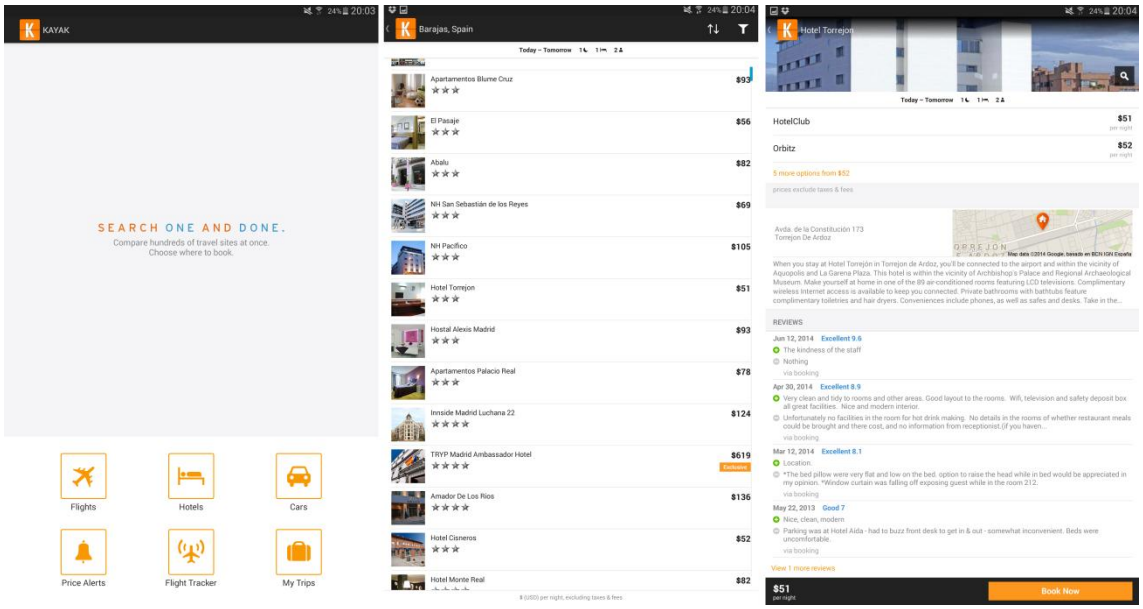

ILUSTRACIÓN 7 - KAYAK [9]

Estas dos últimas aplicaciones permiten, además de la búsqueda de información acerca de servicios turísticos, la reserva de los mismos, permitiendo al usuario introducir sus datos de facturación, o tenerlos registrados, para que el reservar un hotel, vuelo, u otro servicio, se convierta en un mero de trámite de suma facilidad.

3 | ANÁLISIS

En este capítulo se presenta la fase de análisis, parte inicial de todo proyecto software, se definen los requisitos y se muestra una vista global de la arquitectura pensada para el sistema. En el siguiente capítulo se tendrá en cuenta este catálogo de requisitos como base para el diseño de todos los aspectos de la aplicación. Es por ello por lo que la fase de análisis es de suma importancia para el devenir de todo producto software. Es en este capítulo donde se deben asentar las bases, a modo de cimientos, del proyecto, y, a partir de las cuales se construirá todo lo demás.

El proceso a seguir se basa en primer lugar en una buena definición de requisitos y en elegir una metodología de desarrollo acorde con el proyecto.

La arquitectura de la aplicación tendrá el formato de cliente-servidor, constituyendo el servidor la base de datos, y API's de acceso, y cliente, la aplicación móvil.

Éste capítulo es fundamental para entender el proyecto. A continuación se desglosan la funcionalidad y las características a modo de catálogo de requisitos, teniendo en cuenta tanto requisitos funcionales como no funcionales.

3.1 CATÁLOGO DE REQUISITOS

El catálogo de requisitos es la especificación del comportamiento que se espera de cualquier proyecto software. Estudiando otras aplicaciones similares, se ha predefinido una serie de requisitos que se consideran indispensables para el proyecto. A continuación, se muestra una enumeración y breve descripción de los requisitos establecidos para el diseño y desarrollo de la aplicación.

3.1.1 REQUISITOS FUNCIONALES

Los requisitos funcionales describen todas las interacciones que tendrán los usuarios con el software.

3.1.1.1 Gestión de usuarios

RF1: Registro

1. La aplicación debe permitir al usuario introducir sus datos en el formulario de registro.
2. El sistema se encargará de validar los datos.
3. El sistema mostrará un mensaje de error si alguno de los datos es incorrecto o no cumple las condiciones especificadas del formulario de registro.
4. En el caso de que la validación sea correcta, el sistema se encargará de guardar los datos del usuario en base de datos.
5. El sistema enviará un correo electrónico de confirmación al usuario en el caso de que el registro se haya efectuado correctamente.
6. La aplicación mostrará un mensaje de bienvenida al usuario y le redirigirá a la pantalla principal con su sesión ya iniciada.

RF2: Identificación

1. Para iniciar sesión el usuario deberá identificarse con su nombre de usuario y contraseña correspondiente. En el caso de ya haber iniciado sesión anteriormente el inicio de sesión será automático al iniciar la aplicación.
2. El sistema se encargará de validar y permitir o denegar el acceso a la aplicación.
3. El sistema mostrará un mensaje de error en el caso de que la validación no sea correcta.
4. En el caso de que la validación sea correcta, se mostrará un mensaje de bienvenida al usuario y le redirigirá a la pantalla principal

RF3: Cierre de sesión

1. Cualquier usuario de la aplicación debe poder finalizar sesión en la aplicación mediante un botón de que indique "Cierre de sesión".
2. En el caso de que el usuario pulse el botón de cierre de sesión, el sistema mostrará un mensaje, para asegurarse de que el usuario quiere cerrar sesión, permitiendo aceptar o cancelar.
3. El usuario será redirigido a la pantalla principal pero en este caso sin estar logado.

RF4: Solicitud de baja

1. El usuario puede solicitar darse de baja en el sistema.
2. El sistema preguntará si hay alguna razón por la que el usuario desea darse de baja.
3. El sistema mostrará un cuadro de texto que el usuario puede rellenar con sugerencias.

4. La aplicación mostrará un mensaje, para asegurarse de que el usuario quiere darse de baja, permitiendo aceptar o cancelar.
5. En el caso de que el usuario pulse aceptar y finalmente quiera darse de baja, se borrarán todos sus datos de la base de datos.
6. Se le notificará por correo electrónico que se ha dado de baja en la aplicación.

RF5: Cuenta de usuario

1. Debe existir una pantalla en el sistema que permita al usuario consultar o modificar los datos de su cuenta.
2. Si modifica algún dato el sistema lo validará.
3. En caso de que la validación sea correcta, se actualizarán sus datos en la base de datos.

RF6: Mis viajes

1. Debe existir una pantalla en la que el usuario pueda consultar los servicios contratados.

Además, el usuario podrá acceder al detalle de cada reserva y consultar datos de interés.

3.1.1.2 Aplicación

RF7: Pantalla principal

1. La pantalla principal de la aplicación constará de tres botones:
    a. **Búsqueda de vuelo**: La aplicación mostrará un buscador de vuelos.
    b. **Búsqueda de hotel**: La aplicación mostrará un buscador de hoteles.
    c. **Búsqueda de vehículo de alquiler:** Mostrará un buscador de vehículos de alquiler.

RF8: Menú lateral

1. El menú lateral mostrará la opción de acceder a:
    a. **Registrarse**: Formulario para registrar un nuevo usuario en la aplicación.
    b. **Iniciar sesión**: Pantalla para iniciar sesión en la aplicación.
    c. **Mi cuenta**: Datos de cuenta de usuario.
    d. **Mis viajes**: Viajes contratados por el usuario.
    e. **Afiliación**: Pantalla de información acerca de planes de afiliación.
    f. **FAQ's** : Preguntas frecuentes
    g. **Quiénes somos**: Información relevante sobre la empresa.
    h. **Contacto**: Formulario para contactar con la empresa.

RF9: Búsqueda vuelos, hoteles y coches

1. En todas las pantallas de búsqueda se requerirá que el usuario rellene los filtros correctamente.

RF10: Notificaciones

1. El sistema debe lanzar alarmas o notificaciones en los siguientes casos:
    a. Avisos importantes sobre viajes contratados.
    b. Avisos sobre descuentos o viajes nuevos.
    c. Retrasos o información relevante en vuelos.

3.1.2 Requisitos No funcionales

Requisitos complementarios o atributos de calidad. Especifican criterios que juzgan operaciones del sistema en lugar de su comportamiento (requisitos funcionales).

RNF1: Documentación

1. Página con servicio de ayuda online.
2. Manual de usuario de la aplicación.
3. La codificación del sistema deberá ser clara y estar documentada de manera que algún programador pueda agregar funcionalidad posteriormente, procurando seguir los estándares de programación Android.

RNF2: Seguridad

1. Para poder utilizar la aplicación hay que autentificarse.
2. Solo será necesario autentificarse una vez en el dispositivo.
3. Si el usuario no cierra su sesión, se mantendrá abierta para futuros usos de la aplicación.
4. Los datos personales serán cifrados.

RNF3: Mantenibilidad y portabilidad

1. Disponibilidad para todo tipo de dispositivo Android. *Smartphones* o *tablets*.
2. La aplicación Android estará soportada para la mayoría de versiones posibles.
3. Será necesario disponer de una conexión a internet, ya sea por Wifi o por tarifa de datos.

RNF4: Interfaz y usabilidad

1. La aplicación debe constar de una interfaz sencilla, atractiva e intuitiva. De tal forma que su uso no suponga un impedimento o esfuerzo al usuario a la hora hacer uso de la aplicación.
2. La introducción de datos debe estar estructurada procurando evitar errores.

RNF5: Rendimiento

1. Se esperan tiempos de respuesta no superiores a un segundo en las peticiones al servidor y menores en las consultas a la base de datos.
2. Tanto los accesos a base de datos como algún cálculo que se realice en la aplicación no supone demasiada carga para el dispositivo, por lo que el rendimiento será óptimo.

3.2 METODOLOGÍA DE DESARROLLO

Se trata de un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo del software. Existe una gran cantidad de métodos diferenciados por sus fortalezas y debilidades. Una metodología de desarrollo software se basa en:

- Herramientas, modelos y métodos para ejecutar dicho proceso de desarrollo.

Cada metodología de desarrollo tiene su propio enfoque. Entre ellos, para el proyecto Travelling-Service el enfoque de prototipado [8] parece ser el más apropiado, debido a que se centra en la experiencia con usuario y a que su construcción debe ser en poco tiempo.

El modelo de prototipos pertenece a los modelos de desarrollo evolutivo, el prototipo debe ser construido en poco tiempo y sin utilizar muchos recursos. El diseño rápido se centra en representación de aspectos software visibles para el usuario. Este rápido diseño conduce a la creación de un primer prototipo que será evaluado y retroalimentado por el cliente, gracias al cual el equipo de desarrollo podrá entender mejor lo que se debe hacer y permitiendo ver al cliente resultados progresivos a corto plazo.

Las diferentes etapas por la que pasa un software construido con este tipo de metodología son:

- Plan rápido
- Modelado, diseño rápido
- Desarrollo
- Entrega
- Retroalimentación
- Comunicación
- Entrega final

Este modelo es muy útil cuando el cliente conoce perfectamente los objetivos generales del software, pero no detalla los requisitos.

A continuación se ilustra la metodología de desarrollo utilizada en este proyecto mediante un gráfico sencillo.

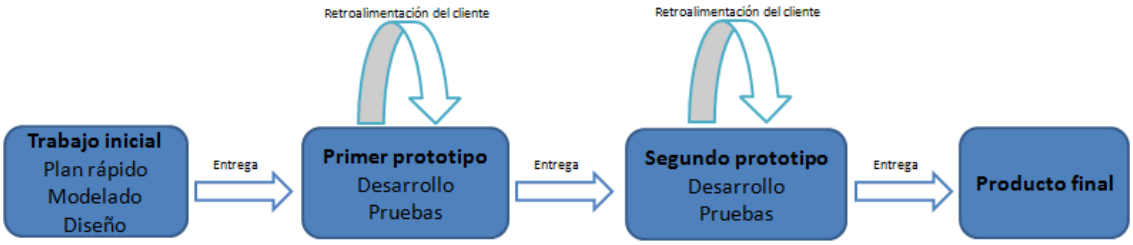

ILUSTRACIÓN 8 - REPRESENTACIÓN GRÁFICA DE MODELO DE PROTOTIPOS

3.2.1 CONTENIDO DE CADA PROTOTIPO

En el primer prototipo se crea la base de datos del servidor y la interfaz de usuario de todas las pantallas de la aplicación. Además, la funcionalidad de registrar usuario e iniciar sesión en la aplicación estarán disponibles.

En el segundo prototipo se intentan refinar las interfaces por si no quedaron lo suficientemente intuitivas. Las búsquedas serán funcionales y se mostrarán los resultados al usuario. Se implementa la funcionalidad del resto de pantallas que no necesitan ni servidor, ni API, ni base de datos, tales como, pantallas de información de la empresa, pantalla de herramientas, etcétera.

En el último prototipo se mejora lo anterior y se implementan las compras del usuario, pasando a ser viajes contratados en la pantalla de mis viajes.

Antes de cada entrega de prototipo se realizan una serie de pruebas como se indica en Ilustración 8 - Representación gráfica de modelo de prototipos.

3.3 BASE DE DATOS

Es obvio que la base de datos no puede ser local en el propio dispositivo, por lo que se implementa una base de datos externa situada en el servidor y a la que se accede a través del API del servidor.

La base de datos almacenará toda la información de usuarios, como la de viajes reservados por ellos, se explicará más adelante en el Capítulo 4 - Diseño.

3.4 SEGURIDAD

Existirá una ligera capa de seguridad para evitar cierto tipo de ataques. La aplicación se comunicará con el servidor por medio de llamadas HTTP para enviar y recibir información. Esta información puede verse en peligro ante ataques *Man in the Middle* [9], por ejemplo. Si el atacante consigue interceptar las cabeceras enviadas al servidor podría obtener datos muy valiosos como contraseñas y nombres de usuario.

La forma más rápida de evitar estos ataques es cifrar los datos que se intercambian entre servidor y cliente, así como cifrar las contraseñas, para que en el caso de que un atacante consiguiera los datos, no le sean de utilidad. En el caso de la aplicación los datos estarán cifrados en md5 con el fin de evitar semejantes ataques.

3.5 ESTUDIO DE API'S

Como se indica en el catálogo de requisitos, existen tres tipos de búsquedas diferenciadas en el proyecto. Búsqueda de hoteles, búsqueda de vuelos y búsqueda de vehículos de alquiler. Se han estudiado diversos servicios web para conseguir esta funcionalidad, quedando finalmente elegidas tres.

Todas proporcionan acceso a datos y servicios que se pueden utilizar como bloques de construcción para desarrollar aplicaciones. En el caso de esta aplicación se utilizarán concretamente:

- *Expedia [24]*: Búsqueda de hoteles. El servicio web de hoteles de Expedia brinda todo tipo de datos sobre hoteles al usuario. Se trata de un servicio web inmenso, con posibilidad de integrarlo no solo en dispositivos móviles.

- *Hotwire [25]*: Búsqueda de vehículos de alquiler. Este servicio web proporciona datos comerciales sobre alquiler de coches. Cada resultado devuelto incluye una dirección URL que se puede utilizar para ver los detalles adicionales y completar una compra.
  La API permite realizar búsquedas utilizando parámetros relacionados con el alquiler de coche, por ejemplo, lugar, fecha y hora de recogida y devolución del vehículo.

- *FlightStats [23]*: Búsqueda de vuelos. El servicio de *FlightStats* ofrece una amplia variedad de información relacionada con transporte aéreo, incluida planificación de vuelo y retrasos.

Todas ellas tienen condiciones de uso bastante similares. La cuenta de desarrollador es gratuita, obteniendo servicios limitados, como número de llamadas a los servicios web y número de resultados ofrecidos por las búsquedas. El segundo plan, de pago, ofrece llamadas ilimitadas a los servicios, pudiendo además utilizar números de cuenta y porcentajes de comisiones para dividir los beneficios que ofrecería nuestra aplicación.

Una vez registrado nuestro proyecto en fase de desarrollo, todas nos proporcionan una clave secreta y cifrada (*APIKey*) que se deberá incluir en todas las llamadas a los servicios de búsqueda desde nuestra aplicación, junto a otros parámetros que se especificarán en el capítulo 5.

4| DISEÑO

En el presente capítulo se describe todo el proceso de diseño de la aplicación Travelling-Service. Se ha realizado un diseño que abarca todos los requisitos descritos en el apartado 3.1 Catálogo de requisitos. El diseño proporciona una idea completa del software desarrollado en el proyecto. Además, se justifican las decisiones tomadas para el posterior desarrollo. Para ello, el capítulo se divide en 4.1 Arquitectura de la aplicación, 4.2 Tecnologías utilizadas, 4.3 Diseño de pantallas, 4.4 Diseño de la interfaz, 4.5 Diseño de la base de datos, 4.6 Diseño de scripts y procedimientos almacenados y 4.7 Diseño de la seguridad.

El diseño de la aplicación ha ido sufriendo algún cambio a lo largo del desarrollo. En consecuencia, se muestran en este documento varias versiones de algunos diagramas de diseño del mismo. A continuación se enumeran los diferentes motivos que han implicado dichas modificaciones:

- Pruebas no satisfactorias, errores, dificultades o no satisfacción de todos los requisitos.
- Ampliación o adición de algún requisito.
- Adquisición de nuevos conocimientos.

4.1 ARQUITECTURA DE LA APLICACIÓN

Como se menciona en el Capítulo 3 - Análisis, la arquitectura de la aplicación se basa en el modelo cliente servidor. En el caso de este proyecto, y como puede verse en la siguiente figura, la aplicación móvil constituirá la parte cliente, y la base de datos estará alojada en un servidor al que se accederá mediante llamadas http.

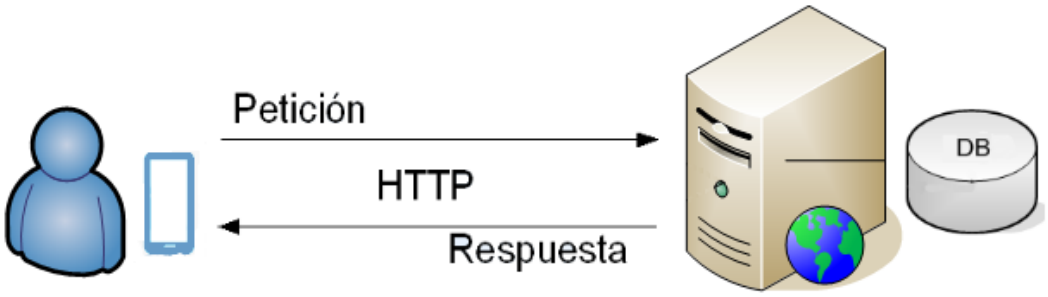

ILUSTRACIÓN 9 - ARQUITECTURA DEL SISTEMA

Para el proyecto Travelling-Service era necesario una base de datos centralizada, y no local en el terminal, por lo que la separación entre cliente y servidor se convierte en más que lógica en este caso.

El cliente es quien inicia las solicitudes, teniendo un papel activo en la comunicación, y espera la respuesta del servidor. En el caso de este proyecto, y como es común en modelos de este tipo, el cliente interactúa con el usuario a través de una interfaz gráfica.

Por otro lado, el receptor de la solicitud, el servidor, espera a que lleguen solicitudes de los clientes, desempeñando un papel pasivo en la comunicación. Tras la recepción, procesa y envía los datos al cliente.

Las ventajas [11] que supone dicha separación son notables, ventajas de tipo organizativo debidas a la centralización de la gestión de la información y separación de responsabilidades, lo que clarifica y facilita el sistema. La escalabilidad y encapsulación son otras de las ventajas más destacables de este modelo, permitiendo aumentar funcionalidades o recursos de ambos por separado.

En el apartado de Diseño de pantallas de este capítulo se ilustran accesos al servidor mediante un diagrama de pantallas.

4.2 TECNOLOGÍAS UTILIZADAS

En este pequeño apartado se enumeran las diferentes herramientas y tecnologías utilizadas durante el desarrollo del proyecto.

4.2.1 NINJA MOCKUPS

NinjaMocks [32] es un constructor gráfico de maquetas. Permite al diseñador organizar y diseñar las maquetas utilizando su potente herramienta *drag&drop,* con la que puede arrastrar elementos y moverlos a su antojo dentro de la maqueta. Dentro de los numerosos editores de maquetas existentes, *NinjaMocks* permitía la creación de un proyecto para Android, pudiendo hacer uso de la mayoría de *widgets* que posee este sistema operativo.

4.2.2 VISUAL PARADIGM

Visual Paradigm [33] es una herramienta de diseño para proyectos software. Soporta multitud de estándares de modelado como UM, SysML, ERD, DFD, BPMN. Es una herramienta que facilita y agiliza el desarrollo de los proyectos basados en experiencias de usuario apoyándose en la identificación de los casos, requisitos o flujos de acontecimientos.

4.2.3 ECLIPSE

Eclipse [12] es un entorno de desarrollo compuesto por herramientas de programación de código abierto multiplataforma. Se le añaden *plugins* para programar en Java por ejemplo (*Java Development Tools, JDT*), o, en el caso del presente proyecto, Android (*Android Development Tools, ADT*).

En concreto se utilizará el *Bundle ADT [13],* que contiene los componentes esenciales de desarrollo Android. Diseñado para ampliar las características de Eclipse y permitir configurar y codificar nuevos proyectos Android rápidamente.

4.2.4 Android SDK

El *Android SDK (Android Software Development Kit [14],* contiene herramientas necesarias para y gratuitas para el desarrollo Android en Eclipse. Combinado con el *Bundle ADT* antes mencionado forman la combinación perfecta para este tipo de desarrollo.

El *SDK* contiene emuladores y un depurador bastante potente. Además incorpora la herramienta *Manager,* que permite descargar e incorporar librerías de la versión deseada al proyecto, así como todo tipo de documentación deseada.

4.2.5 PHP

El lenguaje PHP [15] (*HyperText Preprocessor*) es un lenguaje de código abierto normalmente dedicado al desarrollo web y que puede ser incrustado en lenguaje HTML. Utilizado también en servidores para gestionar llamadas http, lo que es perfecto para alcanzar los objetivos de este proyecto.

4.2.6 POSTGRESQL

Se trata de un sistema de gestión de bases de datos relacionales de código abierto [16], muy común en entornos Linux. Un sistema muy potente que que dispone de gran parte de la funcionalidad básica de gestores comerciales de bases de datos. Todos los datos de la aplicación estarán almacenados en ella.

4.2.7 TORA

Tora [17] es un conjunto de herramientas mutiplataforma de software libre que asiste a desarrolladores de bases de datos Oracle, MySQL y PostgreSQL en su trabajo. Muy valorada, debido a la facilidad que ofrece respecto a la gestión de base de datos.

4.2.8 GENYMOTION

Genymotion[18] es un emulador de Android que aprovecha la arquitectura x86 para dar una forma fluida y rápida a los distintos dispositivos Android que es capaz de emular. Bastante más rápido que el emulador nativo de Android, además es compatible con Windows y Linux, lo que será necesario para el proyecto, pues habrá fase de codificación y pruebas en Linux, con el fin de probar los *scripts php* del servidor en localhost, como se explicará más adelante.

Posee una interfaz simple, capaz de soportar distintas funcionalidades, sin olvidar su principal objetivo, los desarrolladores.

4.3 DISEÑO DE PANTALLAS

Con un primer diagrama de pantallas se pretende dar una visión general de la aplicación, donde las transiciones estén definidas, así como los recursos utilizados por cada una de ellas. En este apartado todas los diagramas se han realizado con Visual Paradigm [33].

4.3.1 PRIMER PROTOTIPO DE PANTALLAS

En este diseño puede observarse cómo la aplicación está en su mayoría definida. Incluso haciendo referencia a las pantallas que van a utilizar APIs, ficheros php del servidor y/o base de datos.

Como puede verse en la **¡Error! No se encuentra el origen de la referencia.**, las pantallas Login y Account, para inciar sesión y registrar un nuevo usuario respectivamente, utilizarán un fichero php del servidor llamado account.php. Además, interactuarán con la base de datos, en concreto con la tabla de usuarios. Del mismo modo que las pantallas Mis Viajes, Mi Cuenta y Darse de baja.

Las tres pantallas de búsqueda de la aplicación, hotel, vuelo y coche utilizan una API para mostrar los datos al usuario según el criterio de búsqueda introducido.

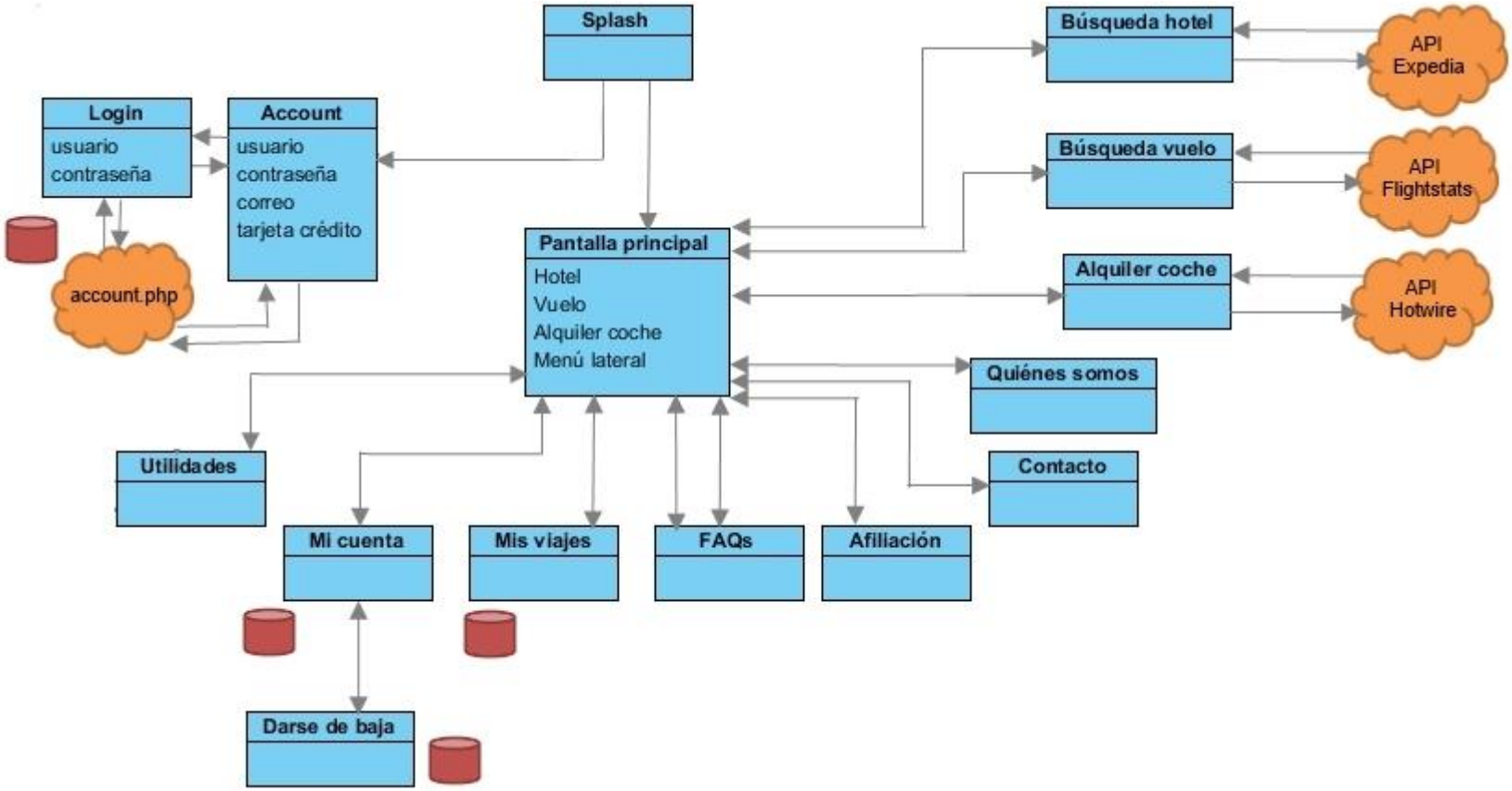

ILUSTRACIÓN 10- DIAGRAMA DE PANTALLAS VERSIÓN 1

4.3.2 Segundo prototipo de pantallas

En un segundo estudio se añaden tres pantallas nuevas al primer prototipo:

- **Detalle elemento:** Muestra el detalle del elemento de la lista filtrada que seleccione el usuario.
- **Detalle reserva:** Similara a la anterior, pero en este caso desde la pantalla Mis Viajes, son elementos ya reservados por el usuario.
- **Pago de reserva:** Pantalla para efectuar la reserva del elemento seleccionado por el usuario.

Las pantallas de detalle hacen peticiones a las diferentes APIs para recuperar los datos de detalle y mostrarlos al usuario.

Una vez se realiza una compra, la aplicación guarda en base de datos los diferentes identificadores de usuario y de elemento reservado mediante peticiones al sevidor utilizando los ficheros php que se describen más adelante en 4.6 Diseño de scripts y procedimientos almacenados, para, en cualquier otro momento, poder recuperar de nuevo el detalle de la reserva mediante una petición a la correspondiente API con los identificadores correspondientes.

Puede verse cómo la aplicación queda finalmente definida.

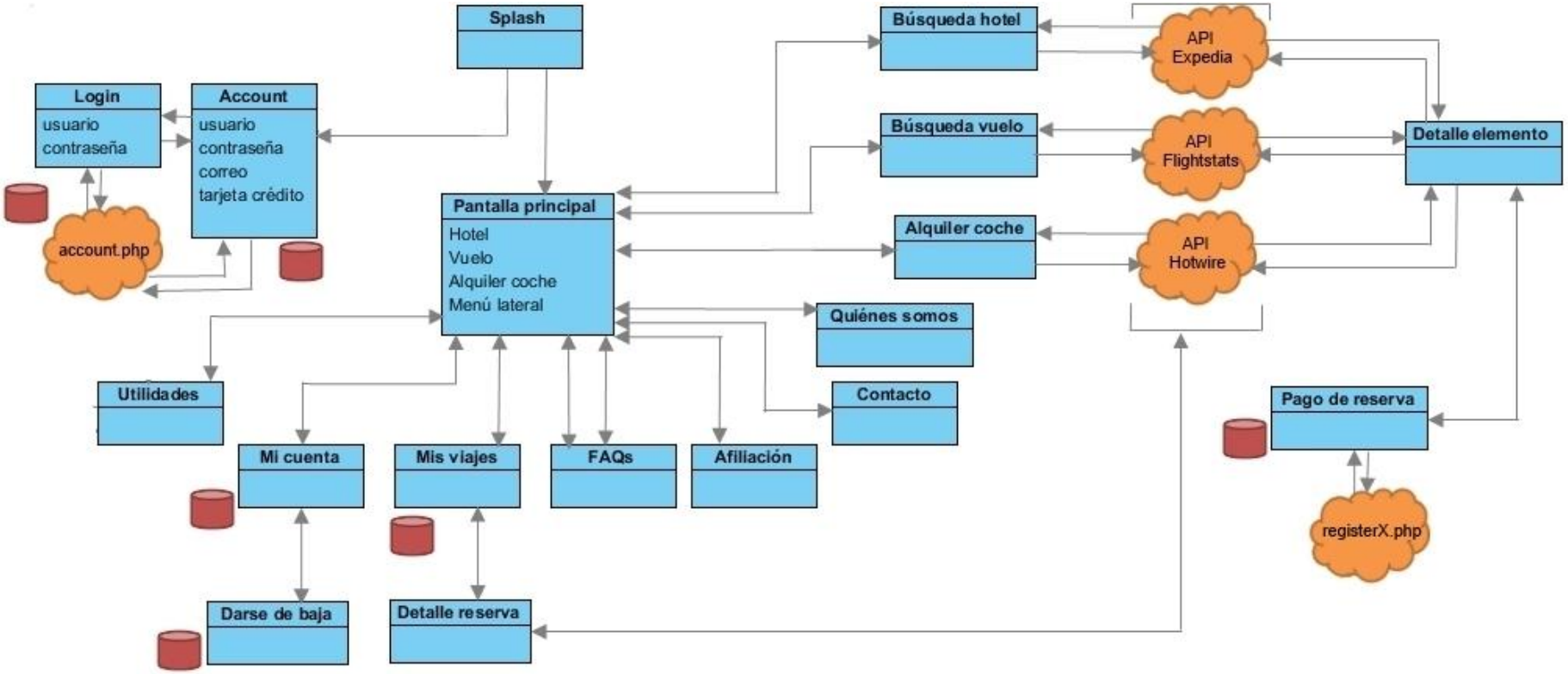

ILUSTRACIÓN 11 - DISEÑO DE PANTALLAS FINAL

4.4 Diseño de la interfaz

La interfaz se ha diseñado con el propósito de que sea intuitiva y simple, pero a la vez potente y capaz de cubrir todos los requisitos enumerados en el apartado 3.1 Catálogo de requisitos.

En este apartado todas las maquetas han sido diseñadas con NinjaMocks [32]. En ellas se puede observar un diseño de maquetas de la aplicación. En primer lugar, y si el usuario no se encuentra con una sesión activa, la aplicación pide un inicio de sesión, dando opción del mismo modo a registrarse en el sistema, en el caso de que no tuviera un usuario creado.

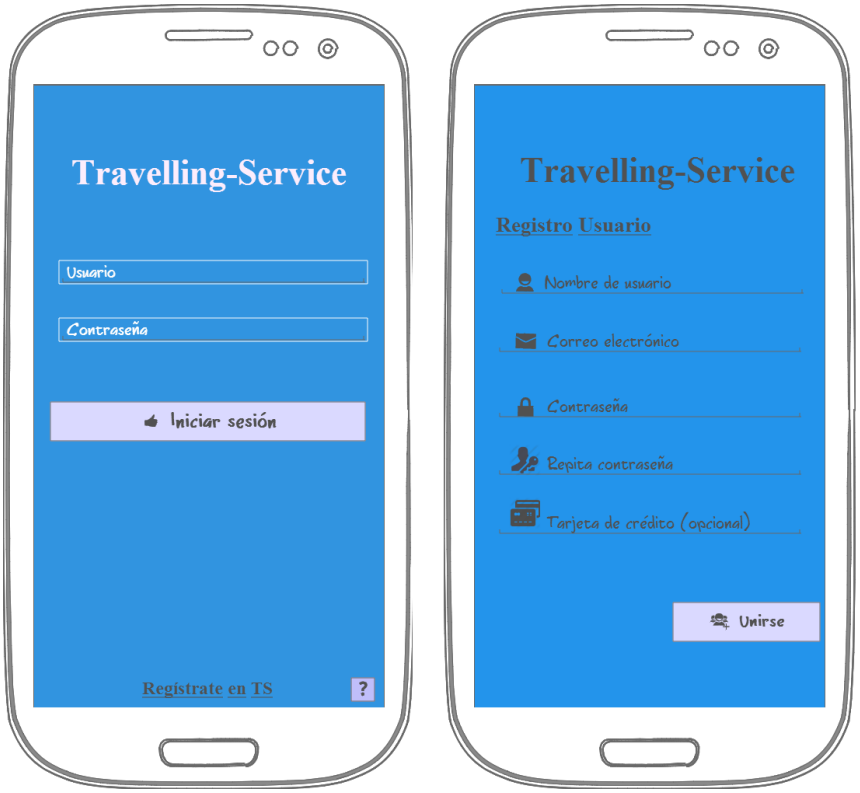

Ilustración 12 - Inicio de sesión y Registro [32]

La actividad principal consta de tres botones que dirigen a cada una de las pantallas de búsqueda, vuelos, hoteles y alquiler de coches.

ILUSTRACIÓN 13 - MENÚ PRINCIPAL [32]

Dependiendo de la pantalla de búsqueda que el usuario desee, deberá rellenar unos filtros de búsqueda diferentes, acordes con el tipo de elementos que desea buscar. Como puede verse en la ilustración de Búsqueda de hoteles puede verse cómo la aplicación requiere que el usuario introduzca la ciudad del hotel que desea, así como número de personas y fecha de entrada y salida del hotel. En la ilustración de búsqueda de vuelos, en cambio, se indica cómo el usuario debe especificar el aeropuerto de origen y el de destino, del mismo modo que el día de salida del vuelo. Finalmente en la ilustración de búsqueda de coche es necesario que el usuario fije la ciudad, la fecha de recogida y entrega del vehículo, y el número de plazas que desea.

ILUSTRACIÓN 14 - PANTALLAS DE BÚSQUEDA [32]

Una vez realizada una búsqueda, los resultados se muestran a modo de lista como puede verse en la siguiente ilustración.

ILUSTRACIÓN 15 - LISTA DE RESULTADOS [32]

En la siguiente figura se muestra el menú lateral que podrá desplegarse en la pantalla principal, tanto arrastrando el desde el borde izquierdo de la pantalla hacia la derecha, como pulsando el icono de rayas que hay en la barra superior.

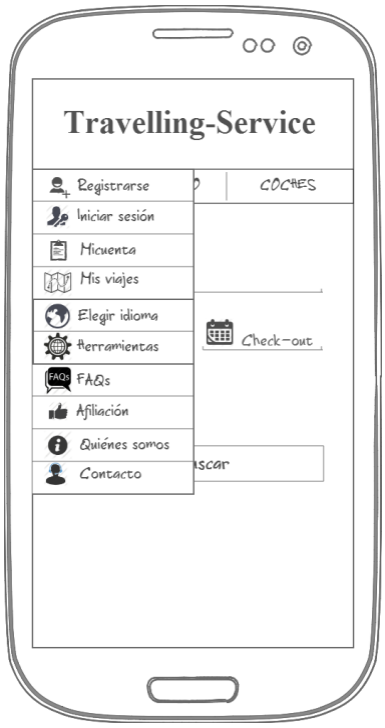

ILUSTRACIÓN 16 - MENÚ LATERAL [32]

En el menú lateral se puede seleccionar "Registrarse" e "Iniciar sesión", cuyas maquetas se pueden ver en las Ilustración 12 - Inicio de sesión y Registro.

También puede accederse a los datos de "Cuenta" del usuario activo, como puede observarse en la siguiente figura.

ILUSTRACIÓN 17 - MI CUENTA [32]

La siguiente opción del menú lateral es "Mis viajes", en la que se mostrará a modo de lista las reservas que ha efectuado un usuario. Además, puede accederse al detalle de cada reserva pulsando el botón de "Detalle de la reserva".

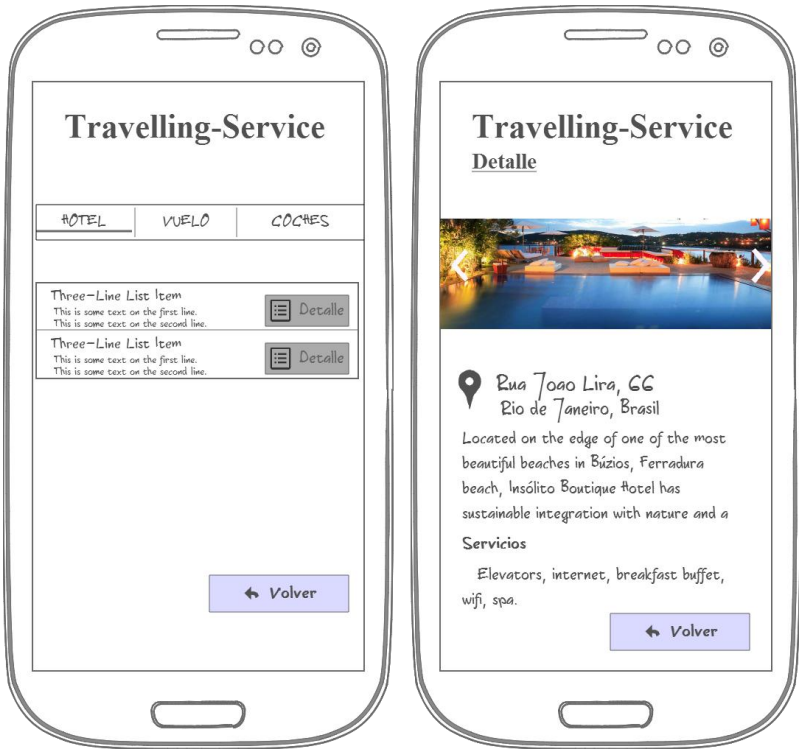

ILUSTRACIÓN 18 - MIS VIAJES Y DETALLE [32]

Las siguientes tres opciones del menú lateral mostrarán información y datos relevantes sobre la empresa, por ejemplo, información sobre planes de afiliación, "quiénes somos" o preguntas frecuentes. Como se muestran en las siguientes figuras respectivamente.

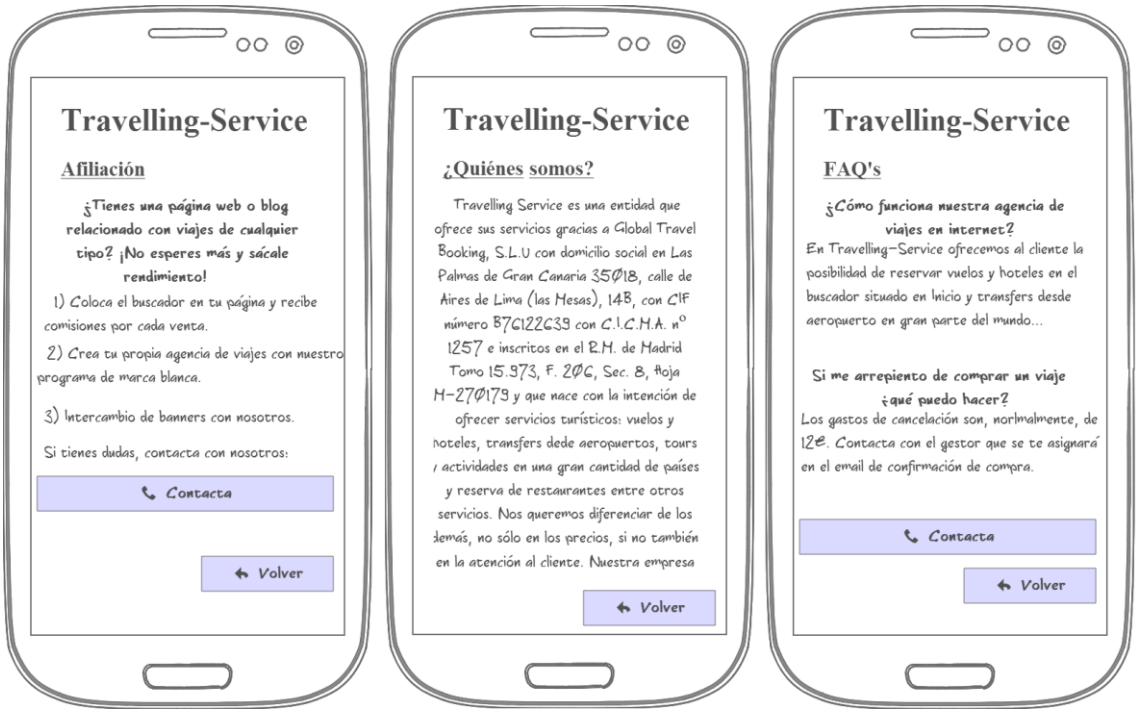

ILUSTRACIÓN 19 - DATOS DE LA EMPRESA [32]

Por último, la figura representa la pantalla del formulario de contacto, desde la cual se podrá contactar con la empresa vía correo electrónico.

ILUSTRACIÓN 20 - FORMULARIO DE CONTACTO [32]

4.5 Diseño de la base de datos

En este apartado se tratan los aspectos de diseño relacionados con la base de datos del servidor. Como se menciona en el apartado anterior, los diagramas están realizados con Visual Paradigm [33]. En un principio se muestra un diseño modelo entidad-relación, al final del apartado se muestra el detalle de tablas.

4.5.1 Modelo entidad-relación

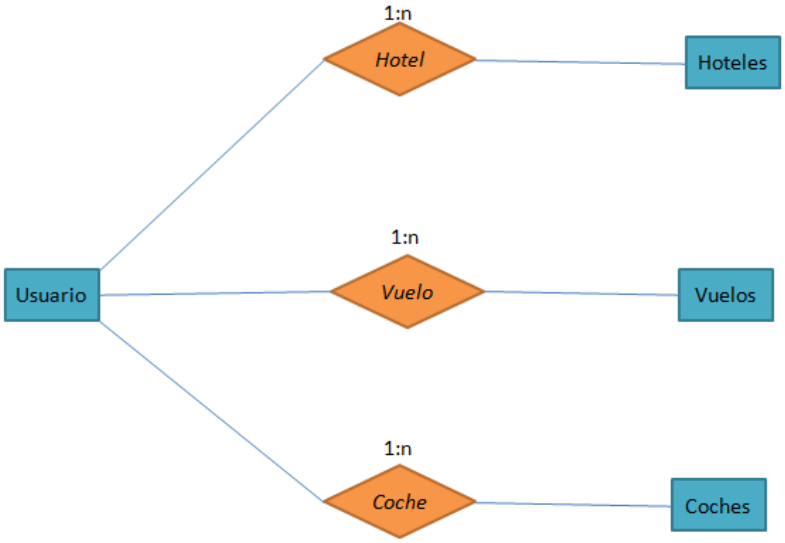

Ilustración 21 - Modelo entidad relación primera versión [33]

La ilustración muestra un primer esbozo del esquema entidad relación de la base de datos. Como puede verse, un usuario podrá contratar un número indeterminado de hoteles, vuelos o coches. Este primer diseño se realizó en la fase de análisis del proyecto, más adelante se enumeran cada uno de los atributos de las tablas de la base de datos.

Además se decidió que las tablas Hoteles, Vuelos y Coches, no iban a estar almacenadas en la base de datos de Travelling-Service, debido a su gran tamaño, alto coste de mantenimiento y sobre todo, porque toda esta información es accesible a través de los servicios web utilizados. Por ello, la base de datos se reduce a la tabla usuarios y las tablas-relación Hotel, Vuelo, Coche. Estando las otras tres tablas anteriormente mencionadas alojadas, por así decirlo, en la nube, como ilustra la siguiente figura.

Para relacionar las tablas de los servicios web con las de la aplicación, se utilizarán los identificadores devueltos por ellas. Por ejemplo, si un usuario reserva el hotel con identificador 15403, ese es el identificador de hotel que se almacenará junto a su identificador de usuario.

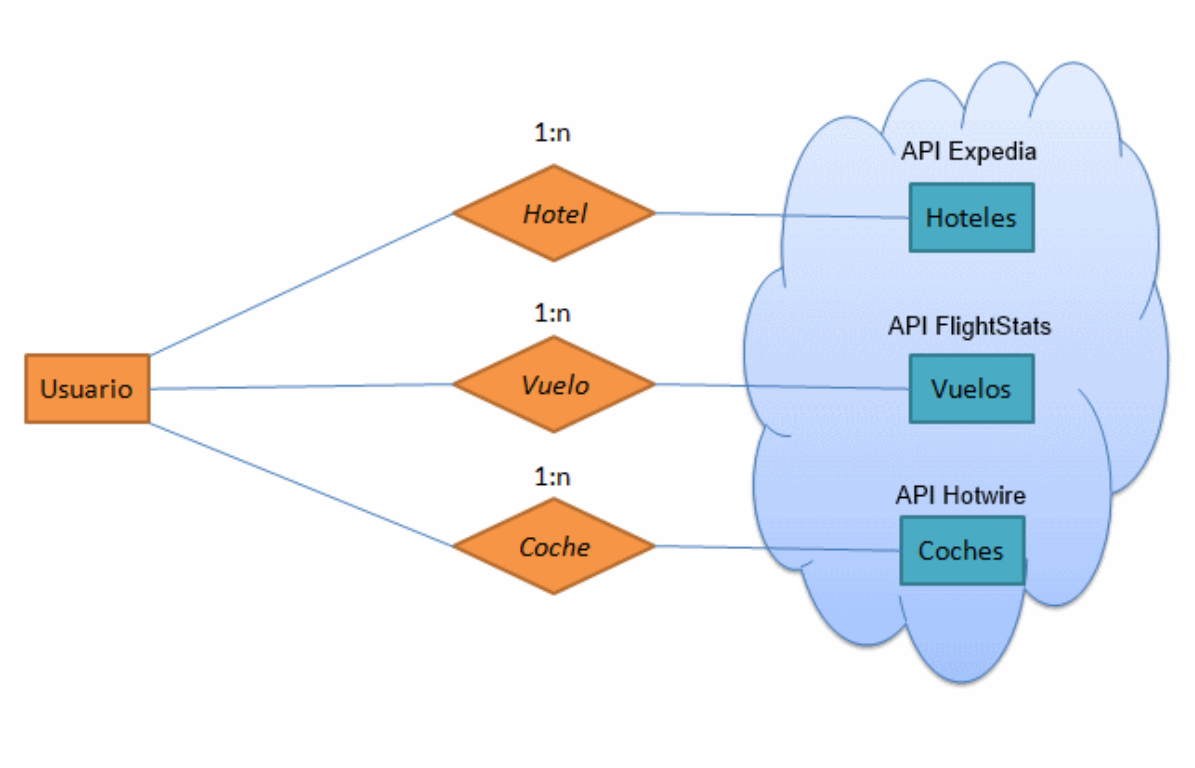

ILUSTRACIÓN 22 - MODELO ENTIDAD RELACIÓN FINAL [33]

4.5.2 DISEÑO FINAL BASE DE DATOS

Las cuatro tablas que constituirán la base de datos del servidor de Travelling Service se muestran en la siguiente figura.

ILUSTRACIÓN 23 - DISEÑO FINAL DE LAS TABLAS DE LA BASE DE DATOS [33]

4.6 DISEÑO DE SCRIPTS Y PROCEDIMIENTOS ALMACENADOS

Como se menciona en el Capítulo 3 - Análisis, la arquitectura del proyecto se basa en cliente servidor. Este apartado da una primera perspectiva sobre la comunicación de la aplicación con el servidor mediante ficheros php.

Existe un script PHP por cada acción importante que requiera la aplicación. Respecto a la gestión de usuarios se codificarán los ficheros **Account.php** y **MyAccount.php**. El primero de ellos accederá a los datos de la tabla usuarios, con el fin de llevar a cabo la consulta de usuario y contraseña, y poder así dar acceso al usuario a la aplicación. El segundo fichero php, en cambio, tendrá la capacidad de modificar los datos del usuario en el caso de que quisiera modificar alguno de ellos.

Otros ficheros a mencionar son los encargados de insertar en la base de datos los datos relacionados con las reservas que efectúe el usuario, que serán tres, **RegisterHotel.php, RegisterCar.php** y **RegisterFlight.php.**

Estos ficheros php devolverán un código de retorno diferente en función de si la operación se ha llevado con éxito, o por el contrario, a surgido algún error.

Por otro lado, se necesitarán ficheros que devuelvan datos que se tengan almacenados en la base de datos, tales como reservas de un usuario. Para ello se implementarán una serie de procedimientos almacenados, con el fin de que el servidor devuelva alguna estructura serializable de la que poder leer dichos datos y mostrarlos en la aplicación, por ejemplo, un estructura de XML. Los procedimientos almacenados estarán codificados en sql, y se incluirán, una vez finalizados, en la base de datos. Los ficheros sql serán **GetFlights.sql, GetCars.sql, GetHotels.sql.**

Por último, se necesitarán unos ficheros php encargados de hacer las llamadas a los procedimientos almacenados de la base de datos. Serán, respectivamente, **FlightsBD.php, CarsBD.php** y **HotelsBD.php.**

Tanto la base de datos como los ficheros php, estarán localizados en un servidor, al cual se accederá mediante peticiones http. Se trabajará de la misma forma con todos ellos, recibirán una serie de parámetros y devolverán un código de retorno o una serie de datos serializados para su posterior tratamiento en la aplicación.

Todos ellos se explican más adelante en el Capítulo 5 - Implementación de la Solución.

4.7 Diseño de la seguridad

Con el fin de evitar ataques y brechas en la seguridad, se decide cifrar datos de usuarios como las contraseñas con md5 [10].

Con esto se consigue que, aunque alguien consiguiera acceso a la base de datos y extrajera datos de los usuarios, dichos datos estarían cifrados, teniendo así que recurrir al uso de una fuerza bruta muy costosa.

Por ahora, la seguridad que incluye la aplicación es muy básica, se deberá ampliar para el producto final, siendo recomendable cifrar por ejemplo los parámetros de las peticiones http.

4.8 Conclusiones

Se presenta este apartado al final del extenso capítulo de diseño, a modo de resumen, y con el objetivo de hacer una síntesis de las decisiones tomadas, que afectarán en gran medida en el siguiente capítulo, implementación de la aplicación. Se destaca por tanto, del capítulo de diseño, los siguientes puntos:

- La arquitectura del proyecto se basa en el modelo cliente-servidor, donde el cliente será la aplicación móvil, y el servidor una base de datos basada en *PostgreSQL*.

- Se ha elaborado un diagrama de pantallas específico con transiciones entre ellas y recursos que utilizarán.
- Se crean unas maquetas que intentarán acercarse lo máximo posible al producto final.
- Asimismo, se diseña la base de datos SQL con un diagrama entidad relación, especificando del mismo modo los campos de las tablas.
- Se hace una primera enumeración de los ficheros PHP y SQL que compondrán la parte de servidor del producto.
- Los ficheros PHP y los procedimientos almacenados se utilizarán para accesos y modificaciones de la base de datos del proyecto, alojada en un servidor.
- Los datos que devuelva el servidor deben estar serializados en XML, con el fin de que su tratamiento por la aplicación sea más sencillo.

5 | IMPLEMENTACIÓN

En este quinto capítulo se aborda la fase de implementación del proyecto. Se explicarán algunos detalles relevantes así como las diferentes dificultades que se encontraron en el proceso.

En esta fase se implementa el diseño realizado en el capítulo anterior de diseño, basado en el catálogo de requisitos del Capítulo 3 - Análisis. Como se ha especificado en el capítulo anterior, se utilizará el *Bundle ADT* de Eclipse, junto al emulador *Genymotion* para la codificación de la parte cliente. Por otro lado, se utilizará el sistema operativo Linux, para trabajar con los *scripts* php y *PostgreSQL*.

A lo largo del capítulo se tratarán los siguientes aspectos:

- Elementos básicos de la parte cliente, módulos, librerías utilizadas e interfaz.
- Aspectos relevantes de la parte servidor, así como parte de su implementación.

5.1 PARTE CLIENTE

Como se ha especificado en capítulos anteriores, la parte de cliente la constituye la aplicación Android. En los siguientes apartados se intenta dar una visión de los elementos básicos que la componen.

5.1.1 ELEMENTOS BÁSICOS

Los primeros elementos a tratar, y con el fin de dar una perspectiva general de la parte cliente, serán los diferentes módulos implementados. Con módulos se intenta representar funcionalidades relacionadas y agrupadas dentro de un mismo paquete Java.

5.1.1.1 Módulos

La mayoría de los módulos realizan llamadas al servidor, todas ellas se realizan mediante tareas asíncronas, utilizando *AsyncTask [19],* con el objetivo de que la aplicación no se quede bloqueada. En los siguientes sub-apartados se explicará con más detalle el uso de tareas asíncronas, para, más adelante, en el apartado 5.2 Parte Servidor, explicar la actuación y devolución de datos por parte del servidor a dicha llamada del cliente.

A lo largo de este apartado se explica la implementación de las diferentes clases, a modo de camino, que un usuario recorrería por la aplicación la primera vez. Por ejemplo, registro, inicio de sesión, búsqueda de datos, reserva de servicios, incluso una posible baja de nuestro sistema.

5.1.1.1.1 Gestión de usuarios

De la gestión de usuarios se encargan las clases **Login.java, Logout.java**, **Register.java** y **Baja.java**. Todas ellas extienden la clase *Activity [20]*, con el fin de interactuar con el usuario.

La actividad **Register.java** es la encargada de dar de alta a un usuario en el sistema. Por eso necesitará una formulario de registro, recoger los datos del usuario, comprobarlos, y realizar una llamada de registro al servidor mediante una tarea asíncrona.

Cabe destacar de esta actividad la comprobación de datos introducidos por el usuario, antes de realizar la llamada al servidor, tales como el correo electrónico, dado que el usuario recibirá correos de la empresa confirmando su registro, o sus distintas reservas más adelante.

```
email = emailEditText.getText().toString();

if(!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()){
    emailEditText.setError("Email no válido");
    return false;
}
```

Ilustración 24 - Cmprobación de Email en el registor de usuario

La actividad **Login.java** es la encargada de realizar la llamada de inicio de sesión en el servidor para un usuario ya registrado en la aplicación. Para ello recoge los datos "nombre de usuario" y "contraseña" del formulario de inicio de sesión. Una vez comprobado que ambos campos poseen caracteres, realiza una llamada al servidor para comprobar dichos datos.

Las actividades **Logout.java y Baja.java** son las encargadas de borrar las preferencias de usuario en la aplicación. Una vez cerrada la sesión, si se vuelve a abrir la aplicación, pedirá un nuevo usuario.

```
/**
 * Hace Logout del usuario de la aplicación borrando sus preferencias
 */
private void LogoutMethod() {
    Preferences.setName(this, "");
    Preferences.setEmail(this, "");
    Preferences.setPassword(this, "");
    Preferences.setCard(this, "");
}
```

ILUSTRACIÓN 25 - BORRADO DE PREFERENCIAS DE USUARIO

La principal diferencia entre las dos reside en que la clase **Baja.java**, además, realiza una llamada asíncrona al servidor para borrar de la base de datos los datos del usuario que en ese momento esta logado en la aplicación.

5.1.1.1.2 INFORMACIÓN DE LA EMPRESA

Las clases About**.java, FAQS.java, Afilliation.java y AfilliationMoreInfo.java**, se encargan de mostrar al usuario información relevante sobre la empresa.

La clase **About.java** presenta al usuario a qué se dedica la empresa, desde cuándo, y qué personas la forman, indicando cuál es el cargo que desempeña cada una de ellas.

La clase **FAQS.java** enumera una serie de preguntas frecuentes sobre la empresa y los servicios que ofrece.

De este grupo de actividades cabe destacar las dos últimas mencionadas, ambas exponen la información a modo de lista desplegable, debido a la cantidad de información que se desea mostrar al usuario, de este modo queda más recogida e inteligible. Para ello se ha implementado una *ExpandableListView [21]* , haciendo uso de padres, hijos, listas de ítems y agrupaciones.

5.1.1.1.3 PREFERENCIAS

Las preferencias (*Shared Preferences*) [22] son un mecanismo sencillo que permite la gestión de diferentes opciones de usuario de una aplicación móvil. Almacenándolas en XML resulta bastante cómodo para el programador por su facilidad a la hora de integrarlas en el sistema. Además, pueden ser usadas como mecanismo para que el usuario pueda modificar alguno de los parámetros de configuración de la aplicación. Por otro lado, pueden ser utilizadas para almacenar ciertos datos que necesite la

aplicación de forma permanente, pues se necesitan almacenar datos del usuario y del servidor, a los que se pueda acceder rápidamente.

En este proyecto las clases Preferences.java y PreferenceFragment.java serán las encargadas de gestionar el listado de preferencias, el cual se almacenará en el fichero settings.xml. Este último fichero es de fácil manejo, debido a estar serializado en XML, donde pueden añadirse y modificarse rápidamente las diferentes preferencias que se desean utilizar. En la siguiente figura puede verse el fichero settings.xml de la aplicación.

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory android:title="Datos personales" >
        <EditTextPreference
            android:key="password"
            android:summary="Pulse para cambiar su contraseña"
            android:title="Contraseña" />
        <EditTextPreference
            android:key="email"
            android:summary="Pulse para cambiar su dirección de correo electrónico"
            android:title="Correo electrónico" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Datos de facturación" >
        <EditTextPreference
            android:key="tarjeta"
            android:summary="Pulse para cambiar o añadir su tarjeta de crédito"
            android:title="Tarjeta de crédito" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Opciones de cuenta">
        <Preference android:title="Darse de baja en Travelling-Service" android:key="BajaKey" android:summary="Deseo eliminar
        <Preference android:title="Cerrar sesión en Travelling-Service" android:key="LogoutKey" android:summary="Deseo cerrar
    </PreferenceCategory>
```

ILUSTRACIÓN 26 - ESQUEMA Y XML DE PREFERENCIAS

Como puede verse, se desea almacenar datos relacionados con el usuario, tales como el correo electrónico o la contraseña, con el fin de que pueda modificarlos desde la aplicación.

Además, se almacenan otros datos de interés en las preferencias, datos necesarios para la comunicación con los servicios web, o con el servidor. En la siguiente figura se muestran algunas de ellas.

```
/*******EXPEDIA PARAMETERS*******/
public final static String API_KEY_KEY = "apikey";
public final static String API_KEY_VALUE = "93ruanueye79tzsyfbrs2smu";
public final static String MINOR_REV_KEY = "minorRev";
public final static String MINOR_REV_VALUE = "4";
public final static String CID_KEY = "cid";
public final static String CID_VALUE = "55505";
public final static String CUSTOMER_USERAGENT_KEY = "customerUserAgent";
public final static String CUSTOMER_USERAGENT_VALUE = "MOBILE_APP";
public final static String CUSTOMER_IPADDRESS_KEY = "customerIpAddress";
public final static String CUSTOMER_IPADDRESS_VALUE = "10.0.3.15";
public final static String CUSTOMER_SESSION_ID_KEY = "customerSessionId";
public final static String CUSTOMER_SESSION_ID_VALUE = "0ABAAA3E-D2B7-5914-7022-64C34F902BBA";
public final static String LOCALE_KEY = "locale";
public final static String LOCALE_VALUE = "en_ES";
public final static String CURRENCY_CODE_KEY = "currencyCode";
public final static String CURRENCY_CODE_VALUE = "EUR";
public final static String DESTINATIONSTRING_KEY = "DestinationString";
/************************************/
```

ILUSTRACIÓN 27 - PARÁMETROS DE COMUNICACIÓN CON LA API DE EXPEDIA

Otro de los aspectos a destacar sobre las preferencias son las funciones codificadas para fijar dichos valores relacionados con el usuario. Una vez iniciada la sesión, por ejemplo, la aplicación guarda los datos del usuario en preferencias, como se ha indicado anteriormente, por medio de funciones codificadas en la clase **Preferences.java.**

```
/*
 * Función que fija el nombre del usuario en preferencias
 */
public static void setName(Context context, String username) {

    SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(context);
    SharedPreferences.Editor editor = settings.edit();
    editor.putString(Preferences.USER_NAME_KEY, username);
    editor.commit();
}
```

ILUSTRACIÓN 28 - FUNCIÓN PARA FIJAR UNA PREFERENCIA

```
Preferences.setId(Login.this, responseBody);
Preferences.setName(Login.this, nombre);
Preferences.setPassword(Login.this, password);
Preferences.setEmail(Login.this, email);
```

ILUSTRACIÓN 29 - ALMACENAMIENTO DE PREFERENCIAS AL INICIAR SESIÓN

5.1.1.1.4 LLAMADA A SERVICIOS WEB

En concreto, las clases de búsqueda de servicios (todas las comenzadas con la palabra Search) y la clase de detalleServicio.java, utilizan llamadas a los servicios web. Como se detalla en el punto 4.3 Diseño de pantallas, donde puede observarse que la aplicación hará uso de tres de ellas. El servicio web de Expedia [24], para la búsqueda de hoteles, el servicio web de HotWire [25], para la búsqueda de vehículos de alquiler y el servicio web de FlightStats [23], para la búsqueda de vuelos. Con el fin de simplificar este apartado, se explica el proceso de llamadas para el servicio web de Expedia sobre búsqueda de aspectos relacionados con hoteles. Las llamadas a los servicios web de HotWire y FlightStats son similares a las llamadas de Expedia mostradas en este apartado. Para realizar una llamada a un servicio web, en primer lugar se necesita una construir el enlace con los parámetros necesarios, tales como, id de usuario o clave secreta de la API proporcionada al desarrollador. Para construir la parte de los parámetros del enlace, se define un vector de pares de valores, con los valores guardados en preferencias, como puede verse en la siguiente figura.

```
Vector<NameValuePair> vars = new Vector<NameValuePair>();
vars.add(new BasicNameValuePair(Preferences.MINOR_REV_KEY,
        Preferences.MINOR_REV_VALUE));
vars.add(new BasicNameValuePair(Preferences.CID_KEY,
        Preferences.CID_VALUE));
vars.add(new BasicNameValuePair(Preferences.API_KEY_KEY,
        Preferences.API_KEY_VALUE));
vars.add(new BasicNameValuePair(Preferences.CUSTOMER_USERAGENT_KEY,
        Preferences.CUSTOMER_USERAGENT_VALUE));
vars.add(new BasicNameValuePair(Preferences.CUSTOMER_IPADDRESS_KEY,
        Preferences.CUSTOMER_IPADDRESS_VALUE));
vars.add(new BasicNameValuePair(
        Preferences.CUSTOMER_SESSION_ID_KEY,
        Preferences.CUSTOMER_SESSION_ID_VALUE));
vars.add(new BasicNameValuePair(Preferences.LOCALE_KEY,
        Preferences.LOCALE_VALUE));
vars.add(new BasicNameValuePair(Preferences.CURRENCY_CODE_KEY,
        Preferences.CURRENCY_CODE_VALUE));
vars.add(new BasicNameValuePair(Preferences.HOTEL_ID_KEY,
        Preferences.getHotelId(HotelDetail.this)));
```

ILUSTRACIÓN 30 - CONSTRUCCIÓN DE VECTOR DE PARES DE VALORES

Una vez construida la parte de parámetros del enlace, se concatena para construir finalmente lo que será la cadena completa con la que se realizará la llamada al servicio web.

```
String DETAIL_PAGE = "http://dev.api.ean.com/ean-services/rs/hotel/v3/info";
String path = DETAIL_PAGE + "?" + URLEncodedUtils.format(vars, null);
```

ILUSTRACIÓN 31 - CONSTRUCCIÓN DEL ENLACE PARA LA LLAMADA AL SERVICIO WEB

Finalmente, con el enlace ya completo, se realiza la llamada al servicio web.

```
HttpGet request = new HttpGet(url);

try {
    ResponseHandler<String> responseHandler = new BasicResponseHandler();
    HttpClient client = new DefaultHttpClient();
    String responseBody = client.execute(request, responseHandler);
```

ILUSTRACIÓN 32 - LLAMADA A UN SERVICIO WEB

El procedimiento a seguir para conseguir los datos de un hotel, por ejemplo, es el que sigue. En primer lugar se debe obtener el identificador del lugar donde el usuario quiere buscar hoteles, para ello se utiliza un servicio llamado *geoSearch*, que al ser llamado junto con una serie de parámetros concretos, devuelve el identificador del lugar deseado, para utilizarlo en llamadas posteriores. Dichos parámetros son los requeridos por Expedia para utilizar sus servicios, los cuales se almacenan en

preferencias. Se pueden observar en la Ilustración 27 - Parámetros de comunicación con la API de Expedia.

Cabe destacar que la respuesta puede ser en XML o JSon, con el fin de que se observe mejor, las ilustraciones están en formato XML, aunque en la aplicación las respuestas son analizadas en JSon.

En la siguiente imagen se muestra un ejemplo del resultado devuelto tras la llamada a *geoSearch* de Expedia para buscar el identificador de Budapest.

```
▼<LocationInfo>
   <destinationId>D0ACF2B1-08C9-4810-BB0E-BA714AC156FD</destinationId>
   <active>true</active>
   <type>1</type>
   <city>Budapest</city>
   <stateProvinceCode/>
   <countryCode>HU</countryCode>
   <countryName>Hungary</countryName>
   <code>Budapest,,HU</code>
   <description>Budapest</description>
   <geoAccuracy>0</geoAccuracy>
   <locationInDestination>false</locationInDestination>
   <latitude>0.0</latitude>
   <longitude>0.0</longitude>
   <refLocationMileage>0.0</refLocationMileage>
   <activePropertyCount>175</activePropertyCount>
 </LocationInfo>
```

ILUSTRACIÓN 33 - GEOSEARCH DE EXPEDIA

Una vez obtenido el identificador de Budapest, utilizamos la función *HoteList* de la API de Expedia con el identificador obtenido. La función devuelve un listado de hoteles de la ciudad. En la siguiente figura se muestran los datos de un solo hotel de la lista devuelta.

```
▼<HotelSummary order="0">
   <hotelId>141122</hotelId>
   <name>Kempinski Hotel Corvinus Budapest</name>
   <address1>Erzsebet Ter 7-8</address1>
   <city>Budapest</city>
   <postalCode>1051</postalCode>
   <countryCode>HU</countryCode>
   <airportCode>BUD</airportCode>
   <propertyCategory>1</propertyCategory>
   <hotelRating>5.0</hotelRating>
   <confidenceRating>75</confidenceRating>
   <amenityMask>18317443</amenityMask>
   <tripAdvisorRating>4.5</tripAdvisorRating>
   <locationDescription>Near St. Stephen&apos;s Basilica</locationDescription>
  ▼<shortDescription>
     &lt;p&gt;&lt;b&gt;Property Location&lt;/b&gt; &lt;br /&gt;With a stay at Ke
     steps from Vorosmarty Square and minutes from Ferenc Deak Square. This
   </shortDescription>
   <highRate>654.7601</highRate>
   <lowRate>252.3586</lowRate>
   <rateCurrencyCode>USD</rateCurrencyCode>
   <latitude>47.49831</latitude>
   <longitude>19.05147</longitude>
   <proximityDistance>1.9597142</proximityDistance>
   <proximityUnit>MI</proximityUnit>
   <hotelInDestination>true</hotelInDestination>
   <thumbNailUrl>/hotels/1000000/30000/21200/21146/21146_151_t.jpg</thumbNailUrl
  ▼<deepLink>
     http://travel.ian.com/index.jsp?pageName=hotAvail&amp;cid=55505&amp;hotelID
     total=2&amp;showInfo=true&amp;locale=en_US&amp;currencyCode=USD
   </deepLink>
 </HotelSummary>
```


ILUSTRACIÓN 34 - MUESTRA DEL LISTADO DE HOTELES

Por último, se puede obtener un detalle del hotel a través de la función *HotelDetail*. Utilizando el identificador del hotel concreto. A continuación se expone una pequeña muestra del detalle del hotel.


```
▼<HotelDetails>
   <numberOfRooms>349</numberOfRooms>
   <numberOfFloors>9</numberOfFloors>
   <checkInTime>2:00 PM</checkInTime>
   <checkOutTime>12:00 PM</checkOutTime>
  ▼<propertyInformation>
     Pets allowed Check-in time starts at 2 PM Check-out time is noon
   </propertyInformation>
  ▼<areaInformation>
     Distances are calculated in a straight line from the property&apos;s location to the point of int
     distance. &lt;br /&gt;&lt;br /&gt; Distances are displayed to the nearest 0.1 mile and kilometre.
     /&gt;Ferenc Deak Square - 0.3 km / 0.2 mi &lt;br /&gt;St. Stephen&apos;s Basilica - 0.3 km / 0.2 r
     /&gt;Vigado Concert Hall - 0.3 km / 0.2 mi &lt;br /&gt;Szechenyi Istvan Square - 0.4 km / 0.2 mi
     /&gt;Hungarian Jewish Museum and Archives - 0.7 km / 0.4 mi &lt;br /&gt;Vaci utca - 0.7 km / 0.4 r
     /&gt;Inner City Parish Church - 0.7 km / 0.4 mi &lt;br /&gt;Hungarian State Opera House - 0.7 km ,
     /&gt;Adam Clark Square - 0.8 km / 0.5 mi &lt;br /&gt;Zero Kilometre Stone - 0.8 km / 0.5 mi &lt;b
     Hotel Corvinus Budapest is Budapest (BUD-Ferenc Liszt Intl.) - 17.7 km / 11 mi. &lt;/p&gt;
   </areaInformation>
```


ILUSTRACIÓN 35 - DETALLE DEL HOTEL

5.1.1.1.5 JSON PARSER

Como se menciona en el apartado anterior, las respuestas por parte de los servicios web son en *Json* [26]. La clase encargada de procesar la respuesta es ***JSONParser.java***. Esta clase se encarga de leer toda la respuesta y construir un objeto *JSon* mediante la función *JSonObject.* También es capaz de construir un *array* de objetos *Json* a través de la función *JSonArray.*

Esta clase es utilizada en cada llamada a una API que devuelve un JSon, por lo que debe estar muy modulada para que su uso sea fácil y rápido. En el proyecto tiene tres métodos.

- *InitializeJson:* Se encarga de hacer la petición http y leer la respuesta.
-

```
public void inicializeJson(String url){
    // Making HTTP request
    try {
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        //HttpPost httpPost = new HttpPost(url);
        HttpGet httpGet = new HttpGet(url);

        //HttpResponse httpResponse = httpClient.execute(httpPost);
        HttpResponse httpResponse = httpClient.execute(httpGet);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
```

ILUSTRACIÓN 36 - INICIALIZAR JSON

- GetJsonObject: Utiliza la función *JSonObject* para crear un objeto *JSon*.

```
public JSONObject getJSONObject(String url) {
    inicializeJson(url);

    try {
        jObj = new JSONObject(json);
    } catch (JSONException e) {
        Log.e("JSON Parser", "Error parsing data " + e.toString());
    }

    // return JSON String
    return jObj;

}
```

ILUSTRACIÓN 37 - FUNCIÓN GETJSONOBJECT

- GetJSonArray: Utiliza la función *JSonArray* para crear un *array* de objetos *JSon*.

```
public JSONArray getJSONArray(String url) {
    inicializeJson(url);
    Log.d("INFO3","URL: "+url);
    Log.d("INFO3","JSON: "+json);
    try {
        jObj2 = new JSONArray(json);
    } catch (JSONException e) {
        Log.e("JSON Parser2", "Error parsing data " + e.toString());
    }
    return jObj2;
}
```

ILUSTRACIÓN 38 - CÓDIGO DE LA FUNCIÓN GETJSONARRAY

Como puede verse, la clase está modulada de tal forma que una llamada a *GetJsonObject* o *GetJsonArray* una vez conseguida la respuesta del servicio web con el enlace correspondiente basta para obtener un objeto *Json*.

Una vez conseguido, se pueden obtener los datos de una forma sencilla, haciendo uso de la función *getString* de un nodo del objeto *Json.* Un ejemplo de las funciones puede verse en la siguiente figura.

```
JSONObject JSONObjMetaData = json.getJSONObject("MetaData");
JSONObject JSONObjCarMetaData = JSONObjMetaData.getJSONObject("CarMetaData");
JSONArray JSONObjCarTypes = JSONObjCarMetaData.getJSONArray("CarTypes");

for (int i = 0; i <= JSONObjCarTypes.length() - 1 ; i++) {

    car = JSONObjCarTypes.getJSONObject(i);

    CarModel = car.getString("PossibleModels");
    CarSize = car.getString("TypicalSeating");
    CarPrice = car.getString("Price");

}
```

ILUSTRACIÓN 39 - FUNCIONES JSON

5.1.1.1.6 XMLPULL PARSER

Por otro lado, y como se explica más adelante en el apartado 5.2.2 Scripts Php, la aplicación debe poder procesar ficheros XML. Para ello, se hace uso del objeto *XmlPullParser [34]*, como puede verse en la siguiente figura.

```
XmlPullParser XMLViajes;
try {

    URL xmlUrl = new URL(path);
    XMLViajes = XmlPullParserFactory.newInstance().newPullParser();
    XMLViajes.setInput(xmlUrl.openStream(), null);
} catch (XmlPullParserException e) {
    XMLViajes = null;
} catch (IOException e) {
    XMLViajes = null;
}
if (XMLViajes != null) {
    try {
        processXML(XMLViajes);
```

ILUSTRACIÓN 40 - XMLPULLPARSER

A continuación, en la función processXML, se procesa el objeto XML para obtener los datos. En este ejemplo, se muestra la función que procesa el XML de hoteles contratados por un usuario concreto.

```
private void processXML(XmlPullParser viajes)
        throws XmlPullParserException, IOException {
    int eventType = -1;
    boolean complete = false;
    String str = null, idHOTEL = null,
    nombreHotel = null, direccionHotel = null;

    while (eventType != XmlResourceParser.END_DOCUMENT) {
        if (eventType == XmlResourceParser.START_TAG) {
            str = viajes.getName();
        }
        if (eventType == XmlResourceParser.TEXT) {
            if (str.equals("id_hotel")) {
                idHOTEL = viajes.getText();
            }
            if (str.equals("nombre_hotel")) {
                nombreHotel = viajes.getText();
            }
            if (str.equals("direccion_hotel")) {
                direccionHotel = viajes.getText();
                complete = true;
            }
            if (complete){
                processTable(idHOTEL, nombreHotel, direccionHotel);
```

ILUSTRACIÓN 41 - FUNCIÓN QUE PROCESA UN XML

5.1.1.1.7 TAREAS ASÍNCRONAS (ASYNCTASKS)

Uno de los aspectos y funcionalidades más importantes del proyecto, es el poder realizar peticiones http a un servidor externo sin que el rendimiento de la aplicación se vea afectado. Es decir, sin que el usuario experimente bloqueos.

Para ellos se utilizará la clase *AsyncTask* que además, posee funciones que permiten realizar acciones antes y después de que finalice la petición. Lo cual es muy útil para poder sincronizar la información recibida y mostrarla en la interfaz de usuario.

Por ejemplo, en el caso de registrar un nuevo usuario, en la función *PreExecute,* justo antes de la petición al servidor, se arranca un diálogo de progreso con un mensaje para el usuario. Una vez finalizada la petición, se libera ese diálogo. En la siguiente figura puede observarse un ejemplo.

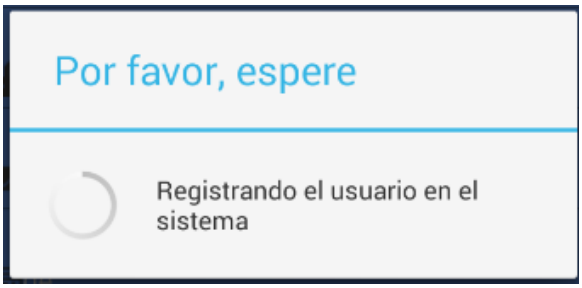

ILUSTRACIÓN 42 - EJEMPLO DE PROGRESSDIALOG

La función a destacar de esta clase es la función *doInBackground,* donde se debe colocar la llamada a realizar en segundo plano, normalmente, la más costosa del proceso. En el caso de esta aplicación, dentro de esta función se coloca la llamada a *PostSettingsToServer,* encargada de realizar la petición al servicio web o al servidor.

5.1.2 LIBRERÍAS

Para cumplir con el requisito de que la aplicación funcione en la mayor parte de los dispositivos móviles Android, se han utilizado las librerías de soporte de Google. Estas librerías dotan de elementos nuevos a las versiones más antiguas de Android. Una de ellas es la librería ***android-support-v4.jar*** que permite a dispositivos con versiones anteriores utilizar funciones de las bibliotecas más modernas.

Otra de las librerías a destacar es ***mail.jar*** que se utiliza para el envío de correos electrónicos desde la aplicación. Esta librería junto con las librerías ***activation.jar*** y ***additional.jar***, otorga a la aplicación la funcionalidad de envío de correos electrónicos. Estas librerías se usan en los casos de olvido de contraseña o registro de usuario, notificándole los datos o la información pertinente. Así como en el formulario de contacto con la empresa, donde Travelling-Service recibirá un correo electrónico con el mensaje del usuario.

5.1.3 INTERFAZ

En este apartado se enumeran los aspectos más relevantes respecto a la construcción de la interfaz de usuario.

Una de las dificultades destacables de la interfaz fue la codificación de la pantalla principal de la aplicación, en concreto del menú desplegable lateral. Hubo que utilizar elementos como *Navigation Drawer, Action Bar Drawer y Navigation Adapter [27].* Con el fin de seguir los estándares de codificación Android, se declararon los elementos que forman la barra lateral en las carpetas de recursos. En concreto, en el fichero *strings.xml* se declara un *array* de *strings,* con los títulos que aparecerán en el menú, así como un *array* de ítems que hacían referencia a recursos de las carpetas *drawable* (imágenes). En la siguiente figura pueden verse ambos *arrays* de recursos.

```
<!--SLIDING MENU-->
 <string-array name="nav_options">
    <item>Registrarse</item>
    <item>Iniciar sesión</item>
    <item>Mi cuenta</item>
    <item>Mis viajes</item>
    <item>Elegir idioma</item>
    <item>Herramientas</item>
    <item>FAQs</item>
    <item>Afiliación</item>
    <item>Quienes somos</item>
    <item>Contacto</item>
    <item>Darse de baja</item>
</string-array>
<!--                -->
```

```
<!-- Lista de iconos de navegacion-->
<array name="navigation_iconos">
    <item>@drawable/ic_action_register</item>
    <item>@drawable/ic_action_user</item>
    <item>@drawable/ic_action_myaccount</item>
    <item>@drawable/ic_action_travels</item>
    <item>@drawable/ic_action_world</item>
    <item>@drawable/ic_action_calculator</item>
    <item>@drawable/ic_action_questions</item>
    <item>@drawable/ic_action_afiliation</item>
    <item>@drawable/ic_action_info</item>
    <item>@drawable/ic_action_contact</item>
    <item>@drawable/ic_darse_baja</item>
</array>
<!--                -->
```

ILUSTRACIÓN 43 - RECURSOS PARA EL MENÚ LATERAL

Una muestra del menú lateral puede observarse en la siguiente figura.

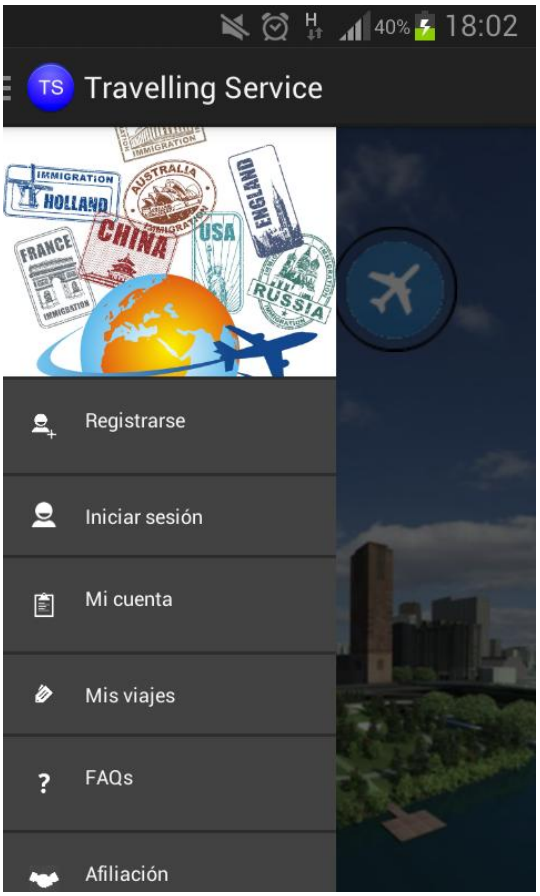

ILUSTRACIÓN 44 - MENÚ LATERAL

Otro de los aspectos recalcables respecto a la interfaz de usuario fue el uso de tablas para mostrar los resultados en la búsqueda de servicios turísticos a través de las *API's* externas anteriormente mencionadas. La tarea asíncrona irá devolviendo resultados que se irán añadiendo a la tabla progresivamente, de este modo el usuario puede ir viendo resultados mientras se descargan, sin experimentar bloqueos en la aplicación.

Para ello las pantallas de búsqueda de servicios turísticos están constituidas por un *ListView.* Utilizando las funciones *publishProgress* y on*ProgressUpdate* de la clase *AsyncTask,* se van añadiendo los distintos registros a la *ListView.*

```
@Override
protected void onProgressUpdate(String... values) {
    if (values.length == 3) {
        ArrayList<String> lst = new ArrayList<String>();
        lst.add(values[0]);
        lst.add(values[1]);
        lst.add(values[2]);
        this.values.add(lst);
    } else {
        Log.e("DEBUG_TAG",
                "onProgressUpdate() inside: Error downloading data.");
    }
}
```

ILUSTRACIÓN 45 - EJEMPLO DE ONPROGRESSUPDATE DE UNA TAREA ASÍNCRONA

Además de lo mencionado anteriormente, se ha utilizado un *ArrayAdapter[28]* con el que luego poder implementar la interfaz *OnClickListener,* permitiendo de este modo al usuario pinchar en la lista, a la aplicación recoger el identificador del elemento, y poder mostrar el detalle cómo se describe a continuación.

```
final HotelList_ArrayAdapter adapter = new HotelList_ArrayAdapter(this, list);
listview.setAdapter(adapter);
listview.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
```

ILUSTRACIÓN 46 - FUNCIÓN ONITEMCLICK DE LISTVIEW

El resultado de la lista puede verse en la siguiente imagen, mostrando un listado de hoteles disponibles con su dirección.

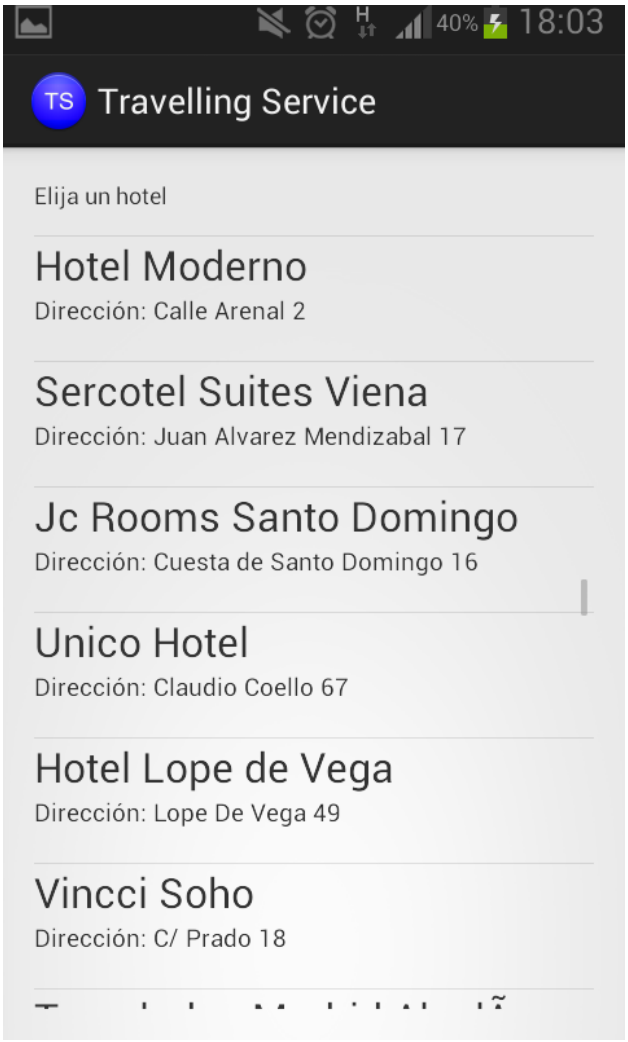

ILUSTRACIÓN 47 - LISTADO HOTELES

Una vez mostrada la lista de servicios turísticos disponibles para esa zona, con el fin de facilitar la usabilidad, el usuario puede pinchar en alguna de las filas de la tabla, lo que le llevará a la pantalla de detalle del elemento concreto.

Por último, cabe destacar la implementación de la pantalla de detalle de un servicio turístico. Como se menciona anteriormente en el apartado 5.1.1.1.4 Llamada a , la función Detail devuelve un fichero XML o JSon con el detalle del elemento concreto. Dentro de ese detalle pueden encontrarse varios enlaces a imágenes. El último de los aspectos a destacar de la fase de implementación es esa descarga de imágenes para mostrarla al usuario.

Esto se consigue mediante otra tarea asíncrona, que procesa el objeto Json devuelto por la *API,* recogiendo un enlace a una imagen y haciendo uso de la clase *BitmapFactory* [29] como puede verse en la siguiente figura, se consigue la imagen para mostrarla al usuario.

```
@Override
protected Bitmap doInBackground(String... URL) {

    String imageURL = URL[0];

    Bitmap bitmap = null;
    try {
        // Download Image from URL
        InputStream input = new java.net.URL(imageURL).openStream();
        // Decode Bitmap
        bitmap = BitmapFactory.decodeStream(input);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return bitmap;
}
```

ILUSTRACIÓN 48 - EJEMPLO DE USO DE BITMAPFACTORY PARA CONSEGUIR UNA IMAGEN

Puede verse en la siguiente figura el detalle de un hotel.

ILUSTRACIÓN 49 - DETALLE DE UN HOTEL

Las pantallas con información sobre la empresa se han implementado con una *ExpandableListView,* como se menciona anteriormente en este capítulo. Se observa una ejemplo en la siguiente imagen.

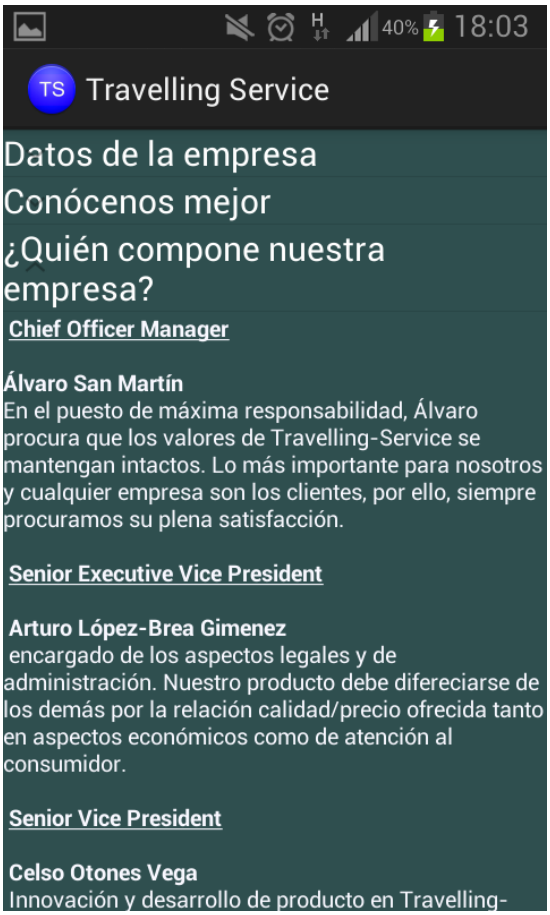

ILUSTRACIÓN 50 - EXPANDABLE LIST

5.2 PARTE SERVIDOR

A continuación se describe parte relevante de la codificación del lado del servidor del proyecto.

Para poder realizar esta fase, fue necesaria la instalación de Ubuntu 12.04 con los siguientes paquetes:

- PostgreSQL
- Netbeans
- Apache2
- Php

- Php5 - pgSQL
- Php5 -xdebug
- Tora
- PhpgAdmin

El objetivo de esta instalación era poder trabajar y realizar las pruebas en *localhost*, con el fin de poder implantar más adelante los ficheros y la base de datos en un servidor real.

5.2.1 Base de datos

Como se describe en el Capítulo 4 - Diseño, se crea una base de datos relacional con las tablas descritas en ese capítulo, teniendo en cuenta claves primarias y claves externas. Se lleva a cabo desde la aplicación Tora.

5.2.2 Scripts Php

Dentro de la parte relacionada con los *scripts php,* cabe destacar dos tipos. El primero de los tipos sirve para insertar o modificar datos de la base de datos, mientras que los *scripts* del segundo tipo son utilizados para consultar datos.

La funcionalidad de los *scripts* se encuentra descrita en el 4.6 Diseño de scripts y procedimientos almacenados, por lo que en este capítulo se ofrecerá una muestra de alguno de ellos.

Todos ellos, en primer lugar, comprueban los parámetros mediante la instrucción PHP *isset($_REQUEST["nombreParametro"]*. Como puede verse en la siguiente ilustración.

```
$salida = basename(__FILE__).":     ";
/*Comprobación de parámetros obligatorios*/
if ((!isset($_REQUEST["nombre"]) || !isset($_REQUEST["password"]))
    $salida = $salida."faltan parametros\n";
    file_put_contents (LOGFILE, $salida, FILE_APPEND);
    echo -1;
    return;
}
```

Ilustración 51 - Comprobación parámetros PHP

A continuación, se recogen los parámetros y se guardan en variables mediante la instrucción *($_REQUEST["nombreParametro"]*. La siguiente figura ilustra el proceso.

```php
/*REQUESTS de parámetros*/
$nombre = $_REQUEST["nombre"];
$password   = $_REQUEST["password"];
$email = $_REQUEST["email"];
$tarjeta = $_REQUEST["tarjeta"];
$login      = isset($_REQUEST["login"]);
```

ILUSTRACIÓN 52 - REQUESTS PHP DE PARÁMETROS

Una vez realizadas las comprobaciones y almacenado los valores, los *scripts* abren una conexión con la base datos. Mediante la función nueva_conexion().

```php
define("PGUSER", "alumnodb");
define("PGPASSWORD", "alumnodb");
define("DSN", "pgsql:host=127.0.0.1;dbname=TFG;options='--client_encoding=UTF8'");
function nueva_conexion_db(){
  $conexion =  new PDO(DSN, PGUSER, PGPASSWORD);
  $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  return $conexion;
}
```

ILUSTRACIÓN 53 - CONEXIÓN PHP A BASE DE DATOS

La función nueva_conexión la utilizarán todos los *scripts* para realizar cambios en la base de datos. Como ejemplo se muestran los *scripts* Account.php y MyAccount.php, que tras realizar los pasos recién nombrados, se encargarán de la gestión de usuarios, modificando o insertando nuevos datos de usuario en la base de datos. Como puede verse en la siguiente figura.

```php
$db = nueva_conexion_db();
$stmt = $db->prepare("SELECT id, password FROM usuario ".
                     "WHERE nombre = :nombre");
$stmt->bindParam(':nombre', $nombre, PDO::PARAM_STR);
$stmt->execute();
/*Si no existe usuario y no es login -> Se inserta usuario*/
if ($stmt->rowCount()==0 && !$login) {
    if (!isset($_REQUEST["tarjeta"])){
        $ok = $db->exec("INSERT INTO usuario (id,nombre,password,email) ".
                        "VALUES (uuid_generate_v4(),'".$nombre."', '".md5($password)."', '".$email."');");
```

ILUSTRACIÓN 54 - EJEMPLO DE SCRIPT INTERACTUANDO CON BASE DE DATOS

El segundo tipo de *scripts* son los que devuelven datos almacenados, con el fin de poder mostrarlos al usuario en la aplicación tras ser procesados. Estos datos se devuelven en formato XML, con el objetivo de que su procesado sea más sencillo y organizado.

Por ejemplo, si un usuario desea ver los servicios que tiene reservados en la pantalla de "Mis Viajes", la aplicación necesita los servicios que tiene guardados ese usuario en la base de datos. Para ello, se hace uso de procedimientos almacenados, que devuelven una tabla con los resultados filtrados de la base de datos, y un *script php*

que construya el XML a devolver con los datos de la tabla que retorna el procedimiento almacenado.

En las siguientes figuras, puede verse el procedimiento almacenado que devuelve los hoteles que tiene reservado un usuario, así como el fichero php que construye el XML, y la devolución de ese fichero XML por parte del servidor.

```
CREATE OR REPLACE FUNCTION getHoteles(idusuario varchar(255))
            RETURNS TABLE(id_usuario varchar(255),
                          id_hotel integer,
                          nombre_hotel varchar(255),
                          direccion_hotel varchar(255),
                          ciudad_hotel varchar(255),
                          link_foto varchar(1000)) AS
$$
DECLARE

BEGIN

    RETURN QUERY (SELECT h.id_usuario,
                         h.id_hotel,
                         h.nombre_hotel,
                         h.direccion_hotel,
                         h.ciudad_hotel,
                         h.link_foto
                  FROM HOTEL AS h
                  WHERE h.id_usuario = $1);

END;
$$ LANGUAGE 'plpgsql';
```

ILUSTRACIÓN 55 - EJEMPLO DE PROCEDIMIENTO ALMACENADO DE LA BASE DE DATOS

```php
$stmt = $db->query("SELECT * FROM getHoteles('$idUsuario');");

$xml = new SimpleXMLElement('<xml/>');
$i = 1;

while (FALSE!=($row=$stmt->fetch())) {

    $round = $xml->addChild('Hotel');
    $round->addAttribute('Numero',       $i);

    $round->addChild('id_hotel',           $row['id_hotel']);
    $round->addChild('nombre_hotel',           $row['nombre_hotel']);
    $round->addChild('direccion_hotel',           $row['direccion_hotel']);
    $round->addChild('ciudad_hotel',           $row['ciudad_hotel']);
    $round->addChild('link_foto',           $row['link_foto']);
    $i++;
}

Header('Content-type: text/xml');
print($xml->asXML());
```

ILUSTRACIÓN 56 - FICHERO PHP QUE CONSTRUYE UN XML

```xml
▼<xml>
  ▼<Hotel Numero="1">
    <id_hotel>15584</id_hotel>
    <nombre_hotel>NH Principe de Vergara</nombre_hotel>
    <direccion_hotel>Principe de Vergara 292</direccion_hotel>
    <ciudad_hotel>Madrid</ciudad_hotel>
    <link_foto>www.nh-hoteles.es/NH-Principe-Vergara.html</link_foto
  </Hotel>
</xml>
```

ILUSTRACIÓN 57 - EJEMPLO DE XML DEVUELTO POR EL SERVIDOR

5.3 CONCLUSIONES

Este apartado se redacta a modo de resumen al final del capítulo de implementación con el fin de condensar los aspectos más importantes y algunas de las dificultades encontradas.

Se han desarrollado los módulos previstos en la fase de diseño, las interfaces y las tareas asíncronas que completan la funcionalidad de la aplicación.

Dado el manejo del lenguaje adquirido en la carrera, tanto en las asignaturas de Java como en la asignatura de Desarrollo de aplicaciones móviles, no han surgido problemas que no hayan podido ser solucionados con una búsqueda en internet o consultando las librerías necesarias.

Por otro lado, sí surgió alguna complicación en la instalación del entorno Ubuntu para el trabajo en *localhost*. Así como alguna tarea que duró un poco más de lo previsto, donde hubo que refrescar conocimientos sobre php y procedimientos almacenados adquiridos en las asignaturas de Sistemas informáticos.

Tras la fase de pruebas en *localhost* de todos estos ficheros, mi tutor, el profesor Gonzalo Martínez Muñoz, alojó dichos ficheros en un servidor de la universidad utilizado para la asignatura de Aplicaciones Móviles, con el objetivo de dar un poco de realismo a la aplicación, y para probar el rendimiento en un servidor real.

La dirección es: ptha.ii.uam.es/viajes

6| PRUEBAS

Se deben separar los dos sectores de pruebas que engloban la totalidad del proyecto. En un primer lugar, la fase de pruebas de la parte cliente (aplicación), y por otro lado, la parte servidor.

6.1 PRUEBAS EN EL CLIENTE

En el primero de los casos mencionados, las pruebas han sido sencillas, tras terminar la codificación de cada prototipo se ha realizado una serie de pruebas con el objetivo de intentar detectar posibles errores. Se realizan pruebas tanto de caja blanca como de caja negra [30] y con usuarios finales.

En todos los prototipos, el procedimiento de pruebas ha sido el que sigue.

En primer lugar, se han realizado pruebas de caja negra, escogiendo funciones que podían dar problemas debido a su complejidad, o a su conexión con el servidor. Se han probado paso a paso intentando dar solución a los problemas encontrados, con el objetivo de evitar errores futuros mayores.

En segundo lugar se realizan pruebas de caja blanca, con conocimiento del código, y examinando las partes o módulos aparentemente más problemáticos, que normalmente coincidían con los módulos marcados como erróneos en las pruebas de caja negra.

Por último, y una vez solucionados los errores de cada prototipo, se llevan a cabo pruebas con usuarios finales, dando como resultado algún cambio en la interfaz, por no ser del todo intuitiva. Cambiando por ejemplo huecos de texto por desplegables, o elementos de Android como calendarios o listas. También algunos huecos de texto se cambian a tipo numérico para que el teclado no muestre letras, por ejemplo, el hueco para introducir la tarjeta de crédito.

A continuación se enumeran las diferentes pruebas en cada uno de los prototipos.

- **Prototipo 1:** Se prueba el registro de usuario y el inicio de sesión desde la aplicación. Asímismo las pantallas de información sobre la empresa.
- **Prototipo 2:** Pruebas sobre búsquedas de servicios turísticos.

- **Prototipo 3:** Se prueban las reservas por parte del usuario.

En todas las fases de pruebas se va comprobando la correcta adición de los datos a la base de datos del proyecto.

En la siguiente tabla se observa el listado de pruebas realizadas. La mayoría de las pruebas que se intentaron fueron operaciones no permitidas para comprobar que el sistema lo detecta y toma medidas al respecto.

Entrada	Salida esperada	Prototipo
Registrarse	Registro de usuario en la BD	1
Registrarse sin indicar algún campo	No permite el registro	1
Registrarse con un correo electrónico que no tiene formato Email	No permite el registro	1
Intento de registro de un usuario ya existente	No permite el registro	1
Dar de baja el usuario	Se da de baja el usuario y se elimina de la BD	1
Inicio de sesión	Permite el inicio de sesión en la aplicación	1
Inicio de sesión sin indicar algún campo	No permite el inicio de sesión	1
Inicio de sesión con usuario y contraseña que no coinciden	No permite el inicio de sesión	1
Se intenta editar el usuario con algún dato vacío	No edita los datos del usuario	1
Se intenta hacer una búsqueda	La búsqueda es satisfactoria	2
Se intenta hacer una búsqueda con campos incompletos	No permite la búsqueda	2
Se intenta acceder al detalle de un servicio que no existe en la lista	No permite visualizar el detalle	2
Se intenta acceder al detalle de un servicio	Permite ver el detalle	2

Intento de reserva de un servicio	Permite la reserva y se registra en BD	3
Intento de reserva sin tarjeta de crédito	No permite la reserva	3
Intento de contactar con la empresa a través de formulario	Se envía un correo electrónico a la empresa	3
Intento de contactar con la empresa con algún campo vacío	No permite el envío del formulario	3

TABLA 2- TABLA DE PRUEBAS EN LA APLICACIÓN

6.2 PRUEBAS EN EL SERVIDOR

Como se ha descrito en apartados anteriores, la parte del servidor se codificó en Ubuntu 12.04, utilizando Apache y trabajando en *localhost.*

Para ir comprobando los cambios en la base de datos se utilizaba el software Tora.

Para depurar errores en los ficheros php se utilizaron varios métodos.

- Instalar el plugin de PHP a Netbeans, para poder depurar paso a paso los *scripts.*
- Revisar el fichero error.log, situado en la ruta con la instrucción $ cat /var/log/apache2/error.log
- Ejecutar el intérprete PHP en la línea de comandos. $ php5 miScript.php

Una de las pruebas más importantes se basaba en registrar un usuario en el servidor y realizar una reserva con él. Todo ello a través de llamadas en el explorador de internet a los scripts PHP, trabajando el *localhost.*

En primer lugar, se muestra en la imagen como se registra el usuario Prueba con constraseña Prueba1. El correo electrónico y la tarjeta de crédito también son de prueba.

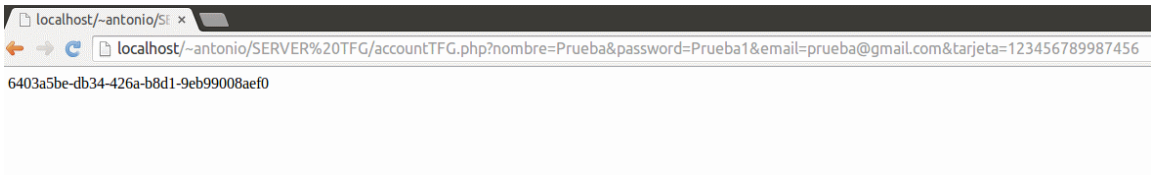

ILUSTRACIÓN 58 - PRUEBA DE REGISTRO DE USUARIO EN EL SERVIDOR

El script devuelve el identificador el usuario.

A continuación, mediante Tora, o mediante el procedimiento almacenado *UsusBD.php* se puede comprobar el registro del usuario en la base de datos.

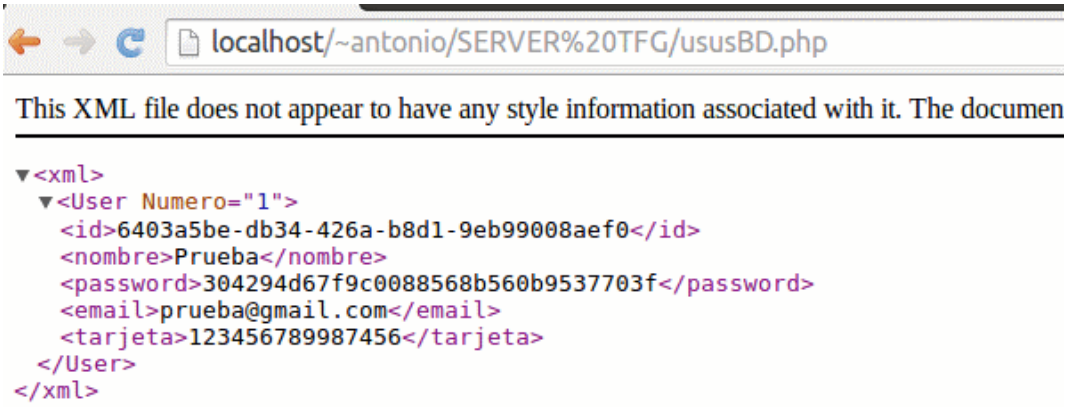

ILUSTRACIÓN 59 - PROCEDIMIENTO DE PRUEBA PARA COMPROBAR USUARIOS DEL SERVIDOR

Para comprobar que la funcionalidad de inicio de sesión funciona, se utiliza el mismo script que para el registro, indicando únicamente usuario y contraseña como parámetros, y en vez de correo electrónico y tarjeta de crédito, en este caso, se indica el parámetro *login* al final de la llamada.

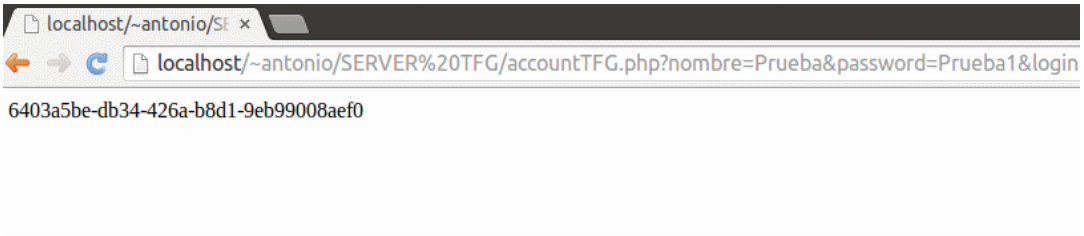

ILUSTRACIÓN 60 - PRUEBA SCRIPT DE INICIO DE SESIÓN

Como puede observarse, el script devuelve el mismo identificador de usuario que se había registrado. En caso de error en alguno de estos scripts, se devuelve un -1.

Por último, se comprueba el *script* que inserta en la base de datos una contratación de un servicio por parte de un usuario, en este caso un hotel.

La llamada al script de reserva de hotel es la siguiente:

http://localhost/~alfonso/SERVER%20TFG/registerHotel.php?idHotel=15584&idUsuario=6403abe-db34-426a-b8d19eb99008aef0&nombreHotel=NH%20Principe%20de%20Vergara&direccionHotel=Principe%20de%20Vergara%20292&ciudadHotel=Madrid&linkHotel=www.nh-hoteles.es/NH-Principe-Vergara.html

Comprobamos que el registro ha sido añadido correctamente a la tabla hoteles, relacionando el identificador de usuario con el identificador del hotel mediante el script hotelesBD.php, que recibe por parámetro el identificador del usuario, devolviendo un fichero XML con los hoteles de ese usuario.

ILUSTRACIÓN 61 - EJEMPLO SCRIPT HOTELESBD

Se puede observar como el mismo identificador de usuario registrado, ha registrado correctamente el hotel en la base de datos.

7 | CONCLUSIONES

Este capítulo se dedica a conclusiones finales y personales del proyecto, extraídas a lo largo de todo el proceso de desarrollo del mismo.

Se han cumplido todos los objetivos y requisitos mencionados en los capítulos. El principal de ellos era crear una aplicación móvil para Android capaz de ofrecer los servicios de la empresa a usuarios por medio de este canal.

La aplicación desarrollada ofrece la búsqueda y posible reserva de servicios turísticos tales como vuelos, hoteles y coches de alquiler, además permite la consulta de datos de interés de la empresa, así como el contacto con ella vía correo electrónico.

Atendiendo al desarrollo técnico del proyecto, fue difícil sintetizar todas las ideas y requisitos que había sobre la mesa. Una vez realizadas las fases de análisis y diseño, las fases posteriores fueron encauzadas rápidamente. Por lo que las primeras fases de planificación, análisis y diseño resultaron ser de suma importancia.

Además de haber aprendido a desarrollar un proyecto de principio a fin, pasando por todas sus fases, he adquirido nuevos conocimientos, o ampliado muchos de ellos sobre Java, Android, modelo cliente-servidor, PHP, SQL y otros conocimientos necesarios que se han requerido durante todas las fases.

8 | TRABAJO FUTURO

Dentro del trabajo futuro en este proyecto pueden destacarse varias vertientes.

La primera de ellas es la mejora de la interfaz gráfica, con ayuda de una persona especialista puede hacerse que la aplicación tenga un acabado visual más profesional.

Otro de los aspectos a mejorar es la seguridad, en este proyecto solo se han cifrado datos como las contraseñas o las tarjetas de crédito. Cuando la empresa comercialice la aplicación sería necesario implementar una capa de seguridad más exhaustiva. Relacionado con este tema, el incluir la API de Paypal podría ser de gran interés con el fin de reforzar la seguridad y la confianza de los usuarios.

En las pantallas de detalle de los servicios turísticos se muestra información general acerca de ellos, tales como imágenes o descripción general. Sin embargo los servicios web devuelven mucha información en los servicios de detalle, por lo que sería interesante poder sintetizarla y mostrarla al usuario.

Por otro lado, Android es un sistema operativo que se actualiza continuamente, por lo que debería programarse una tarea de mantenimiento para estudiar nuevas versiones y adaptar el proyecto, evitando de este modo comportamientos erróneos de la aplicación en futuras versiones de Android.

Puede hacerse necesario el poder actualizar los datos de la empresa por parte de alguien no especializado en desarrollo Android. Por lo que una actualización de los recursos de la aplicación vía web podría resultar interesante.

REFERENCIAS

[1] https://www.noellevitz.com/documents/shared/Papers_and_Research /2013/2013_E-Expectations_mobile.pdf

[2] http://devs.ticbeat.com/cuantas-personas-usan-apps/

[3] http://di002.edv.uniovi.es/~cueva/asignaturas/doctorado/2001/ complejidad.pdf

[4] http://confluence.wkcols.com/display/DEV/Quality+on+Software+ Development

[5] http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision- general-y-entorno-de-desarrollo/99-arquitectura-de-android

[6] http://andro4all.com/2013/04/fragmentacion-marzo-2013

[7] http://www.touristeye.es/

[8] http://es.wikipedia.org/wiki/Modelo_de_prototipos

[9] http://www.veracode.com/security/man-in-the-middle-attack

[10] http://www.md5.net/

[11] http://alegsa.com/Respuesta/ventajas_y_desventajas_del_modelo_ clienteservidor.ht

[12] http://es.wikipedia.org/wiki/Eclipse_(software)

[13] http://developer.android.com/tools/sdk/eclipse-adt.html

[14] http://developer.android.com/sdk/index.html

[15] https://php.net/manual/es/getting-started.php

[16] http://www.postgresql.org.es/sobre_postgresql

[17] http://es.wikipedia.org/wiki/Tora_(Bases_de_Datos_Oracle)

[18] http://www.genymotion.com/

[19] http://developer.android.com/reference/android/app/ AsyncTask.html

[20] http://developer.android.com/reference/android/os/Activity.html

[21] http://developer.android.com/reference/android/widget/ExpandableListView.htm

[22] http://www.androidcurso.com/index.php/tutoriales-android/42-unidad-9-almacenamiento-de-datos/299-preferencias

[23] http://api-portal.anypoint.mulesoft.com/flightstats/api/flight-status-and-track-api

[24] http://developer.ean.com/docs/getting-started/

[25] http://developer.hotwire.com/

[26] http://json.org/

[27] https://developer.android.com/design/patterns/navigation-drawer.html

[28] http://developer.android.com/reference/android/widget/ArrayAdapter.html

[29] http://developer.android.com/reference/android/graphics/BitmapFactory.html

[30] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm

[31] http://wordp.relatividad.org/ciencia/cuota-de-mercado-android-apple-blackberry-symbianwindows/

[32] http://ninjamock.com/

[33] http://www.visual-paradigm.com/

[34] http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html

ANEXO A

MANUAL DE USUARIO

Al arrancar la aplicación se mostrará una pantalla de inicio.

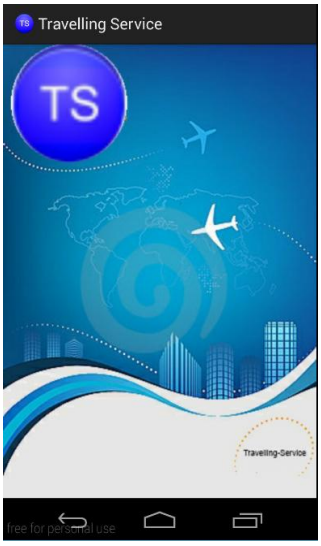

ILUSTRACIÓN 62 - PANTALLA DE INICIO

A continuación, la aplicación pedirá que nos registremos, o en su defecto que iniciemos sesión.

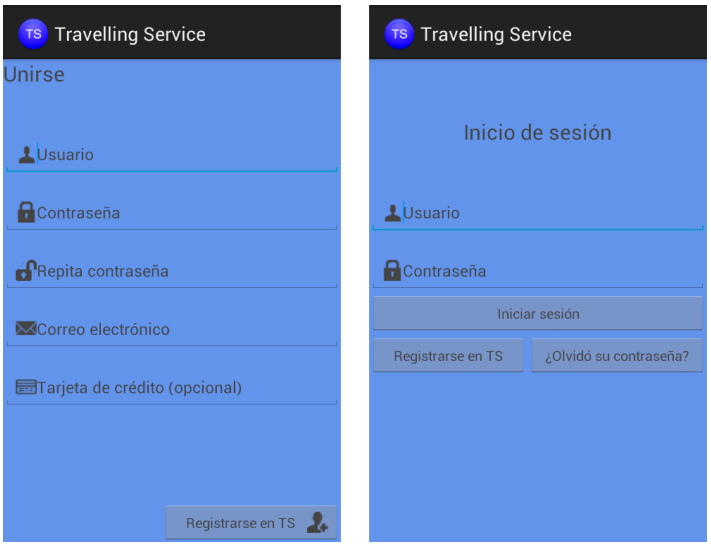

ILUSTRACIÓN 63 - REGISTRO E INICIO DE SESIÓN

Una vez iniciada la sesión, se accede a la pantalla principal, donde podemos buscar vuelos, hoteles o coches de alquiler.

ILUSTRACIÓN 64 - MENÚ PRINCIPAL

Si el usuario dese buscar un hotel, por ejemplo, debe pulsar en el botón del medio, en el del icono que lo representa. Mostrando la aplicación la siguiente pantalla. Donde deberá introducir el lugar deseado.

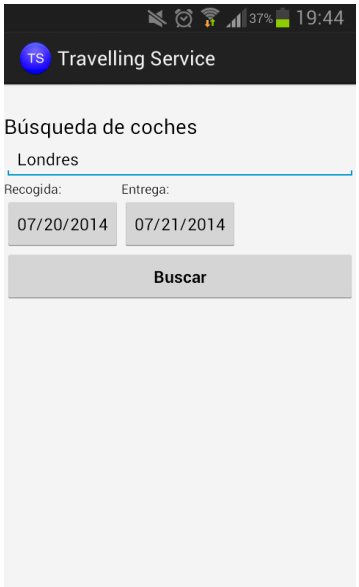

ILUSTRACIÓN 65 - PANTALLA DE BÚSQUEDA DE HOTELES

Además, el usuario podrá elegir la fecha de entrada y salida del hotel, así como el número de huéspedes.

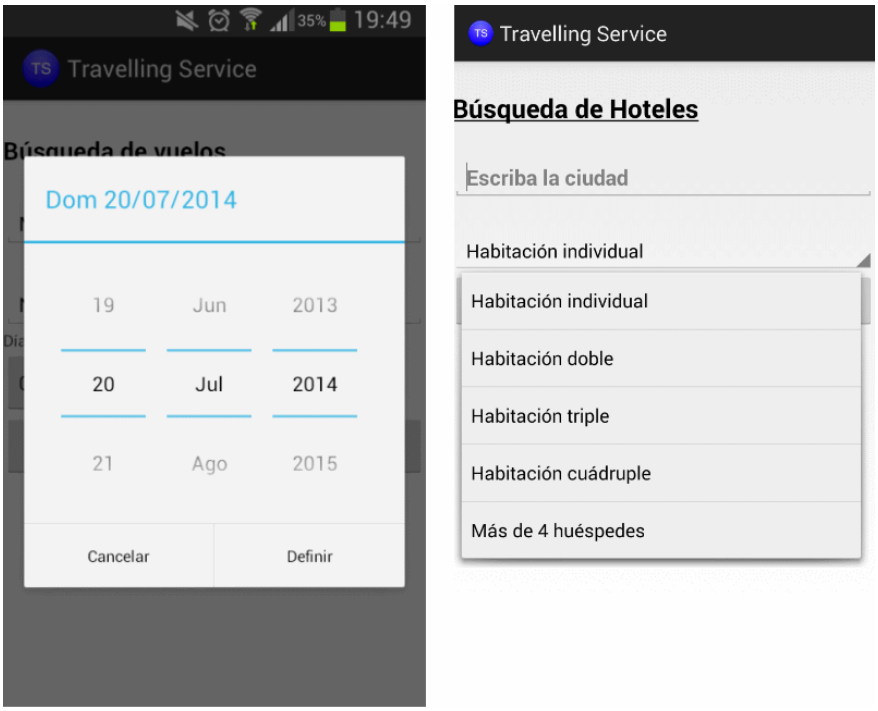

ILUSTRACIÓN 66 - ELECCIÓN DE FECHAS Y HUÉSPEDES

A continuación, tras pulsar en buscar, se mostrará una lista de hoteles del lugar elegido.

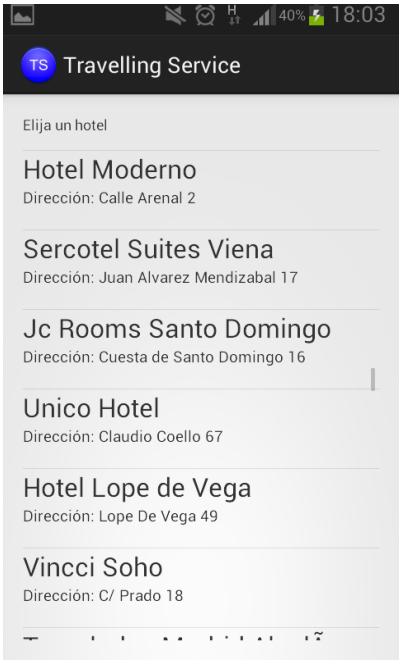

ILUSTRACIÓN 67 - LISTADO DE HOTELES

Para acceder a un hotel, basta con pinchar en uno de la lista. Mostrándose así un detalle sobre el mismo, con dirección, precio, descripción y una imagen.

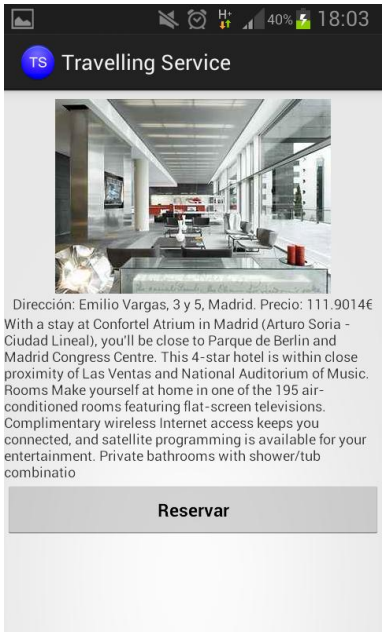

ILUSTRACIÓN 68 - DETALLE DEL HOTEL

Volviendo a la pantalla principal, si se arrastra el dedo de izquierda a derecha, se despliega un menú lateral con diversas opciones. También se puede desplegar el menú pulsando en el icono de tres rayas junto al icono superior de la empresa.

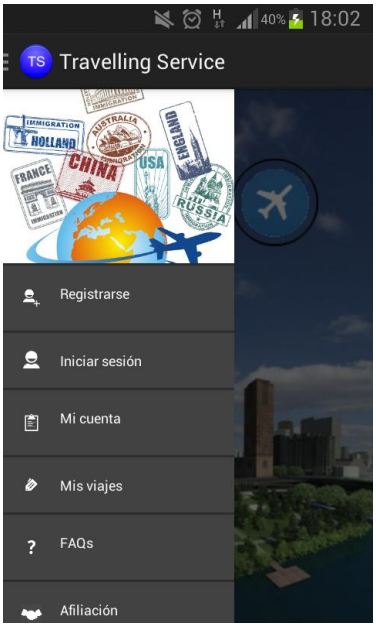

ILUSTRACIÓN 69 - MENÚ LATERAL DESPLEGADO

Pueden encontrarse opciones como registrarse o iniciar sesión, mencionadas más arriba. Si el usuario elige la opción "Mi cuenta", podrá modificar sus datos de usuario, así como cerrar la sesión o incluso darse de baja.

ILUSTRACIÓN 70 - PANTALLA MI CUENTA

Otra de las pantallas accesibles desde el menú lateral es la de "Mis viajes", dónde el usuario puede revisar los servicios contratados mediante la aplicación, pudiendo acceder también al detalle de los mismos. Las pantallas son similares a las de listado de hoteles y detalle de hotel mostradas anteriormente.

Otras pantallas son las relacionadas con información de la empresa. El usuario puede acceder a "Quiénes somos", "Faqs" o "Afiliación" desde el menú desplegable.

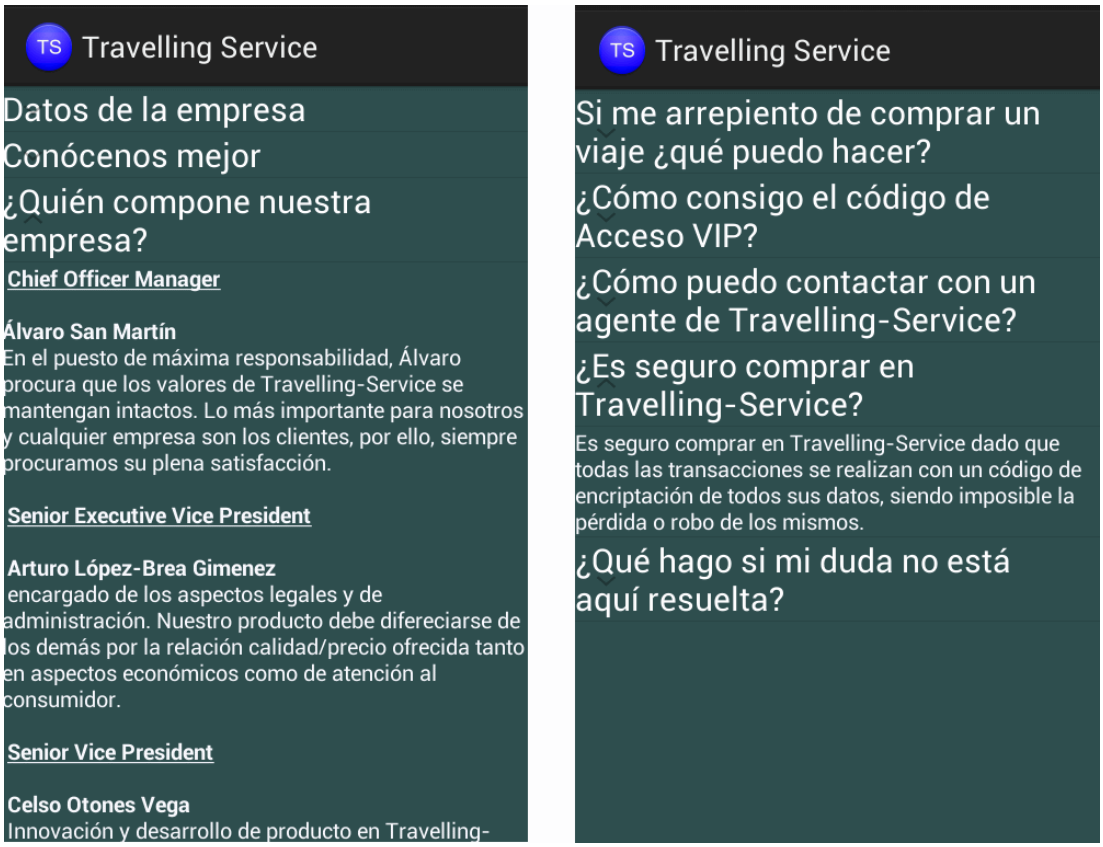

ILUSTRACIÓN 71 - INFORMACIÓN SOBRE LA EMPRESA Y FAQS

ILUSTRACIÓN 72 - PROGRAMAS DE AFILIACIÓN

Por último, se puede encontrar un apartado de contacto en el menú desplegable, que permitirá al usuario ponerse en contacto con la empresa vía correo electrónico.

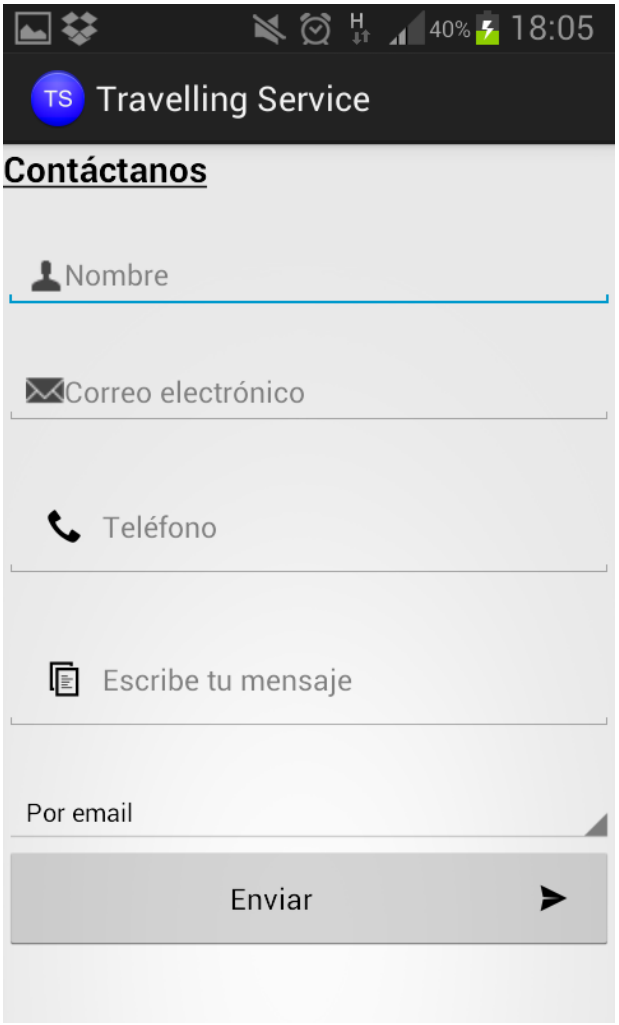

ILUSTRACIÓN 73 - CONTACTA