UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Aplicación para dispositivos móviles basados en Android para la consulta y gestión de reservas de vuelos de la aerolínea Canaryfly.

Trabajo de Fin de Grado

Las Palmas de Gran Canaria
Noviembre de 2014

Autor:

Marlon Demís Fernández Berga

Grado en Ingeniería Informática (Tecnologías de la Información)

Tutor:

Abraham Rodríguez Rodríguez

Ciencias de la Computación e Inteligencia Artificial

Agradecimientos

Con la finalización de este proyecto culmina mi etapa universitaria, la cual recordaré con mucha añoranza en el futuro por todos los grandes momentos que me ha dado, momentos felices como cuando apruebas una asignatura "de las difíciles" o momentos complicados, como cuando tienes que estudiar para aprobar dicha asignatura, el estrés y desesperación se apoderan de nosotros en esos momentos, pero aún así, son parte de las experiencias que con tanto satisfacción me llevo.

Los años que han durado mis estudios universitarios, con alguno que otro de más, no ha sido provechoso únicamente desde el punto de vista académico, las personas a las que he tenido la oportunidad de conocer son gente que recordaré por siempre y con quienes, ojalá, puede seguir compartiendo infinitos momentos en el futuro.

En estos momentos uno se pone a recordar y a tener en cuenta muchas cosas y soy consciente del increíble apoyo con el que he contado, el cual, en momentos de flaqueza, han hecho que siga adelante. Mi familia ha sido la principal fuente de ayuda y no tengo más que palabras de gratitud, especialmente gracias a mis padres por demostrándome la confianza y el orgullo que tienen hacia mí, por la educación y valores que me han inculcado y que hayan hecho que hoy esté donde estoy, y por supuesto gracias a mis dos hermanas pequeñas por su cariño incondicional y la felicidad que me proporcionan. Gracias a los cuatro, la mejor familia del mundo, los amo.

También quiero agradecer a mis dos amigos de toda la vida, que a lo largo de estos años los he tenido que descuidar en más de una ocasión a causa de la carrera pero que siempre lo han entendido y hoy siguen ahí como el primer día.

A mis "super colegas del infierno", que son lo mejor que me llevo de la universidad y quienes han hecho que estudiar ingeniería informática haya valido la pena. Me siento realmente afortunado de haberles conocido y de poder decir que soy amigo de ustedes.

Al resto de compañeros que he tenido a lo largo de los años, especialmente a los de "Colegueo", gracias por todos los momentos que hemos vivido juntos.

Quiero agradecer a aquellas personas que, en mayor o menos medida, han invertido algo de su tiempo en proporcionarme su ayuda, sin ustedes me hubiera costado mucho más.

Por supuesto, tengo que agradecer al profesor Abraham Rodríguez Rodríguez por acogerme ante la búsqueda de tutor para este Trabajo de Fin de Grado.

Y por último, gracias a la empresa Canaryfly, por confiar en mí y permitirme dar mis primeros pasos en el mundo profesional. Especialmente me gustaría agradecerle a mi compañero Ricardo Tamarán Núnez Monzón, webmaster del Departamento de Informática, toda su dedicación en la realización de este proyecto, sus consejos, todas las dudas que me ha resuelto y la claridad con la que lo ha hecho han sido vitales.

Muchas gracias a todos por su granito de arena, el cual ha permitido que hoy esté aquí.

Índice

1. Introducción

1.1. Resumen

El presente documento describe el desarrollo de la aplicación basada en dispositivos Android que la empresa Canaryfly, una aerolínea canaria, ofrecerá a sus clientes con el objetivo de facilitarles información de sus vuelos y la posibilidad de gestionar sus reservas.

Actualmente Canaryfly no cuenta con ninguna plataforma que permita una interacción eficaz con dispositivos móviles, motivo por el cual nace la propuesta de este proyecto.

De entre todas las posibilidades que existían, se ha seleccionado desarrollar dicha aplicación en Android por dos motivos principales. El primero de ellos, el gran cupo de mercado que abarcan los dispositivos móviles con este sistema operativo [1]. El segundo motivo fue por preferencia personal, dado que el Grado en Ingeniería Informática proporciona conocimientos sobre el lenguaje de programación Java, para el cual existe un soporte muy amplio y completo por parte de Android.

La aplicación final pretende ser una extensión del sistema actual de Canaryfly, dado que el objetivo que se persigue es el de ofrecer las mismas funcionalidades que ofrece la página web. De esta manera, se busca una unificación de los procesos y la administración de los contenidos.

Canaryfly, como cualquier aerolínea, utiliza un Sistemas de Distribución Global, también conocido como GDS [2]. Estos sistemas externos se dedican especialmente a proveer soluciones informáticas de distribución para la industria del viaje y el turismo. En el caso de Canaryfly, el sistema externo contratado es Kiu Systems Solutions [3], la cual provee toda la información necesaria, tanto en la web como en la aplicación móvil, mediante su web service [4].

Con todo lo anterior mencionado, el trabajo implementado conforma un primer prototipo que abarca las principales funcionalidades de una compañía aérea, como es la obtención de información de vuelos, posibilidad de acceder a la información de una reserva y realizar la facturación online. Además se han desarrollado formularios propios de Canaryfly que permiten la solicitud de presupuesto para vuelos especiales ofertados.

El diseño y la arquitectura utilizados buscan que el resultado final de este Trabajo de Fin de Grado (de aquí en adelante TFG) sea escalable, ya que en el futuro se seguirá desarrollando el resto de funcionalidades necesarias hasta alcanzar una versión estable que pueda ser explotada.

1.2. Objetivos principales

Desde un punto de vista académico, se ha buscado adquirir los fundamentos de la programación orientada a dispositivos móviles, especialmente aquellos que trabajan sobre la arquitectura del sistema operativo Android. El estudio de la gestión de recursos y procesos que ofrece este sistema ha sido vital para el buen desarrollo del proyecto.

Dado que este software puede ser integrado en muchísimos tipos de dispositivos distintos, con versiones de sistema y tamaños de pantalla diferentes, se hace un verdadero reto implementar una aplicación que funcione correctamente en todos los móviles o tabletas. Por ello, se ha hecho un estudio sobre las mejores prácticas a la hora de programar y diseñar una aplicación Android, especialmente sobre la personalización de los elementos visuales del dispositivo. Este trabajo ha proporcionado ser capaces de de tener un producto 100% personalizado sin ningún elemento por defecto.

Para el desarrollo de esta aplicación ha sido necesario que varios sistemas diferentes trabajen conjuntamente, por lo que se ha construido una arquitectura lo más robusta posible capaz de modular al máximo todo el diseño del proyecto.

Desde el punto de vista de las funcionalidades que puede llegar a ofrecer la aplicación de cara a los clientes de la empresa, se ha buscado satisfacer las necesidades de los usuarios casuales, pero especialmente se le quiere brindar un mejor manejo de sus reservas a aquellos clientes fieles y que tienen por costumbre viajar con Canaryfly.

La aplicación quiere ser una extensión de la actual página web, por lo que se quiere centralizar la administración de los contenidos de ambos sistemas. Para ello se ha llevado a cabo una reestructuración interna, especialmente en lo que atañe a las bases de datos. Como es obvio, desde los teléfonos se quiere ofrecer los mismos servicios facilitados en la página web, en este proyecto se han desarrollado una serie de funcionalidades que pretenden ser optimizadas y ampliadas en el futuro.

Formularios para solicitar presupuesto de vuelos en grupo o charter, recuperar información de una reserva y realizar la facturación online son elementos que la aplicación debe satisfacer, pero especialmente se busca un sistema eficaz para obtener de una manera muy sencilla toda la información relativa a los horarios, las tarifas y los precios, con las diferentes opciones que existen al haber varios tipos de descuento, de los vuelos que ofrece la aerolínea.

Con todo esto, se pretende inflar la repercusión de la empresa Canaryfly en el sector de las aerolíneas y llegar a ser una opción preferente a la hora de elegir compañía para realizar nuestros desplazamientos.

1.3. Estado del arte

En lo referente al estado del arte de este proyecto, se ha investigado sobre las aplicaciones Android que ofrecen servicio de reservas de vuelos. Para esto hemos hecho uso de la plataforma de distribución de aplicaciones para móviles con sistema operativo Android, Google Play Store [5], donde se puede encontrar varias categorías y diferentes tops basados en el número de descargas y en la puntuación que la comunidad les concede.

A continuación mostraremos las 5 aplicaciones destacadas en la categoría "Viajes y guías" a noviembre del año 2014, dando una breve descripción de cada una de ellas:

- Skyscanner todos los vuelos [6]

10 millones de descargas - 4,5 de puntuación con 151.446 votos.

En el número 1, la aplicación más descargada y mejor valorada por los usuarios, Skyscanner, un buscador de viajes que compara millones de vuelos en todo el mundo entre cientos de compañías aéreas. Encuentra las mejores ofertas en vuelos baratos del mercado.

- KAYAK [7]

5 millones de descargas - 4,5 de puntuación con 132.360 votos.

En segundo lugar, con menos descargas que el primer puesto pero con una puntuación muy similar, KAYAK, un buscador de viajes inteligente que realiza búsquedas y reservas de hoteles, vuelos y coches; rastreo de vuelos y gestión de errores.

- EasyJet [8, 9]

1 millón de descargas - 4,3 de puntuación con 49.164 votos.

La siguiente aplicación pertenece a una aerolínea británica de bajo coste, con EasyJet se pueden buscar, reservar y gestionar vuelos de entre 580 rutas con las que trabajan por toda Europa y 104 entre Europa y el norte de África, vendiendo por internet o teléfono.

- Vueling [10, 11]

500.000 descargas - 3,6 de puntuación con 2.787 votos.

Primera aplicación perteneciente a una aerolínea española, también de bajo coste. Vueling vuela a más de 130 destinos con unas tarifas muy asequible. Gracias a esta aplicación podemos aplicar los descuentos de residente y familia numerosa en trayectos nacionales.

- Ryanair [12, 13]

1 millón de descargas - 2,8 de puntuación con 11.221 votos.

En último lugar, la aerolínea irlandesa lowcost con más trayectos en Europa y la más rentable del mundo, Ryanair. Sin embargo, en lo que a su aplicación en Android se refiere, vemos que es la que peor puntuación tiene de este estudio y está bastante lejos del número de descargas de los primeros puestos. Su lentitud es su gran problema.

Por último, es importante mencionar que todas estas aplicaciones deben interactuar con un Sistema de Distribución Global (GDS). La empresa líder mundial en proporcionar este servicio es Amadeus [14], la cual trabaja con 490 aerolíneas en más de 217 países por todo el mundo.

Como conclusión, vemos que la principal tendencia en este tipo de aplicaciones es la de comparadores de vuelos más que las de una aerolínea en concreto. De esto podemos deducir que una aerolínea que está sopesando la posibilidad de crear una aplicación para dispositivos móviles debe tener en consideración el público al que va dirigido y cubrir necesidades mucho más específicas, pudiendo obviar algunos factores más generales, sabiendo que la repercusión de su producto difícilmente alcance a la de los buscadores de vuelos, pero logrando una mayor aceptación entre su público más fiel.

2. Competencias

Toda la información referente a las competencias del TFG se han obtenido directamente de la web oficial de la Universidad de Las Palmas de Gran Canaria [15].

- CII01.

  Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

A lo largo del proyecto se han realizado una serie de fases que dan por validada los requerimientos de esta competencia, los cuales serán explicados en capítulos posteriores del presente documento.

  - Análisis: Obtención, análisis, especificación y validación de requisitos.
  - Diseño: Diseño de arquitectura, estructura de datos e interfaces de usuario.
  - Implementación: Se pone de manifiesto los conocimientos adquiridos durante los estudios, aplicándolos de manera correcta para un desarrollo eficiente.

- CII02.

  Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

Al igual que la competencia anterior, las fases de análisis, diseño e implementación que conforman el desarrollo de este proyecto dan por satisfecha esta competencia. Como complementación, en el siguiente capítulo de este documento (Aportaciones) comentamos el impacto económico y social que este producto tiene.

- CII04.

  Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.

Para el desarrollo e instalación de este TFG se han tenido en cuenta los requisitos software y hardware fundamentales, los cuales se expondrán en capítulos posteriores del presente documento.

- *CII18.*

  Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Esta competencia se ve satisfecha gracias al análisis realizado sobre la legislación y normativa vigente que afecta a este TFG, el cual puede consultarse en el capítulo 4, "Normativas y Legislación".

- TFG01.

  Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

La realización de este Trabajo de Fin de Grado, la escritura del presente documento a modo de memoria del proyecto y la posterior presentación y defensa del mismo ante el tribunal universitario correspondiente cumplimenta esta competencia.

3. Aportaciones

3.1. Aportación al entorno socio-económico y técnico

Es incuestionable la gran repercusión que han tenido los smartphones en los últimos años y como se han convertido en parte de la cultura tecnológica que nos rodea. Esto abre la posibilidad y la necesidad de desarrollar aplicaciones para dispositivos móviles para casi cualquier entorno. La facilidad de acceso a nuestro teléfono móvil crea un enlace directo entre el público potencial y el producto que queremos ofertar.

Con este tipo de aplicaciones también conseguimos entrar en la tendencia moderna de eliminar el papel físico, y tener toda nuestra documentación en nuestro dispositivo.

En el contexto de este TFG tenemos que analizar a donde queremos llegar con la implementación de una aplicación móvil para una aerolínea. Lo primero a tener en cuenta son sus ventas. Este tipo de organizaciones suelen tener varios puntos de venta de billetes, tales como las agencias de viaje, los terminales en los aeropuertos o una página web corporativa. Este último, al menos en el caso de Canaryfly, es el punto de venta con mayores ingresos. Teniendo esto en cuenta es lógico pensar que si a los usuarios que compran por web se les proporciona una manera sencilla y eficaz de

consultar información de vuelos desde su teléfono móvil, estas ventas sufrirían un aumento considerable, además de conseguir el prestigio deseado.

Al ser una aerolínea canaria el mercado al que se enfoca es el transporte a algunos puntos del oeste de África pero principalmente se dedica a los viajes entre islas. Existen diferentes alternativas pero la empresa que copa este mercado por encima del resto es la aerolínea Binter Canarias [16]. Actualmente Canaryfly tiene que trabajar mucho para llegar a ser la primera opción por parte de los canarios a la hora de elegir compañía para viales insulares, ya que cuenta con muchos menos años de experiencia y desarrollo que Binter, pero se ha detectado que esta organización no cuenta con una aplicación nativa para dispositivos móviles, por lo que el desarrollo de este proyecto pretende aportar un pequeño grano de arena para conseguir una ventaja competitiva y distintiva.

Con este proyecto y muchos más que tiene pendiente la empresa, Canaryfly pretende de aquí a unos años estar a un nivel similar al de su competidor directo y que la sociedad canaria, en su mayoría, la conozca y la tenga en cuenta del mismo modo.

3.2. Aportación a nivel personal

En mi realización de Prácticas Externas en la empresa Canaryfly a finales del año 2013 pude demostrar mis aptitudes adquiridas en el Grado en Ingeniería Informática, gracias a las cuales pude optar a un puesto de trabajo una vez finalicé las prácticas.

Desde entonces he aprendido muchísimo en el campo del desarrollo web, además de experiencia que sólo se puede conseguir en un entorno real de trabajo. En todo este tiempo he participado en diversos proyectos y he desarrollado diferentes aplicaciones para uso interno de la empresa, pero nunca había tenido la posibilidad de introducirme en el mundo de los dispositivos móviles, el cual desde hace ya unos años me llama muchísimo la atención y hoy por hoy puedo decir que es algo en lo que quiero especializarme.

Mis conocimientos en el lenguaje de programación Java hizo que el jefe del Departamento de Informática de la empresa pensara en mí para la realización de este proyecto, el cual acepte con mucha ilusión.

Teniendo en cuenta la dimensión de esta aplicación se propuso como Trabajo de Fin de Grado, lo cual fue, en mi opinión, una decisión acertadísima. He disfrutado mucho en la realización de este proyecto lo cual ha hecho que le ponga gran empeño y dedicación. He adquirido una gran cantidad de experiencia en una de las ramas de la informática que considero es el futuro y estoy impaciente por seguir estudiándola.

Profesionalmente hablando, este ha sido el primer proyecto del que se puede decir que soy el líder absoluto y el que controla su desarrollo completamente. La confianza que Canaryfly ha depositado en mí en la agradezco y aprecio enormemente. Esa toma de responsabilidad me ha hecho crecer mucho y sólo me da ánimos para seguir desarrollándome como profesional.

4. Normativa y legislación

4.1. Normativa y regulación de la informática en el ámbito nacional.

Las normativas y legislación expuestas en este apartado hacen referencia al ámbito nacional donde se encuentra el servidor en el que se desplegará este sistema, que sería España.

Licencia de Software [17]

Una licencia de software establece un contrato entre licenciante, el autor o titular de los derechos de distribución y explotación, y licenciatario; que especifica los términos y condiciones a seguir para el uso del software. No existe un único tipo de licencia de software, éstas varían según si el código es abierto, cerrado, de dominio público o según su destinatario.

- Licencia Campus de Microsoft (Paquete de Office de Microsoft) gracias al convenio entre la ULPGC y Microsoft:

Esta licencia representa un simple acuerdo de alquiler de software permitiendo a la Universidad que tenga el programa utilizar los productos Microsoft en todos los ordenadores durante uno o tres años. Este contrato permite el uso de aplicaciones Microsoft Office, de escritorio, en los equipos domésticos de los empleados de la Universidad. Su ha sido necesario para redactar este documento con la herramienta Word.

- Licencia de código abierto permisiva (Apache, PHP, MySQL Workbench Community Edition, Android Studio)

Este tipo de licencia establece que se puede crear una obra derivada a partir de la fuente, sin que tenga obligación de protección.

- Licencia dual de General Public License [18] (GNU, que es la licencia de tipo código abierto robusta fuerte) y licencia de código cerrado cuando aplique (MySQL database)

MySQL database (sistema de gestión de base de datos relacional) dispone de una licencia GNU que, según sus cláusulas, obliga a que la distribución de cualquier producto derivado de MySQL conlleve las mismas cláusulas que la licencia. Ahora bien, si un usuario/desarrollador desease distribuir un producto que funcione con MySQL, y hacerlo bajo una licencia diferente a la de GNU, puede adquirir una licencia comercial que se lo permita, de ahí la "licencia dual".

- Licencia dual GNU y Common Development And Distribution License [19] (NetBeans IDE)

La licencia CDDL es producida por Sun Microsystems y basada en la Licencia Pública de Mozilla, que se categoriza como licencia de código abierto robusta débil. Esto significa que, al igual que la robusta fuerte, contiene una cláusula que obliga que cualquier modificación que se lleve a cabo en el software original deberá ser licenciada bajo los mismos términos y condiciones que la licencia original. La diferencia yace en que las obras derivadas bajo esta licencia pueden ser distribuidas bajo licencias con otros términos y condiciones.

4.2. Normativa y regulación de la informática en el ámbito europeo.

El "Convenio sobre la Ciberdelincuencia" [20], de 21 de noviembre de 2001, firmado en Budapest y ratificado por varios países, surge de la necesidad de aplicar una política penal común para proteger a la sociedad de la ciberdelincuencia. Así es como cada representante de un país miembro del Consejo de Europa firmó dicho acuerdo. La meta del "Convenio sobre la Ciberdelincuencia" implicaba la definición de los delitos informáticos, junto con otros elementos relacionados con dichos delitos.

Este convenio es el único acuerdo internacional que aborda la seguridad de la información, tratando a los delitos contra la Confidencialidad, Integridad y Disponibilidad de la misma y los sistemas informáticos. A continuación se expondrá la categorización de delitos que presenta este convenio.

- Delitos contra la confidencialidad, la integridad y la disponibilidad de los datos y los sistemas informáticos
- Delitos informáticos
- Delitos informáticos relacionados con el contenido
- Delitos relacionados con infracciones de la propiedad intelectual y derechos afines

4.3. Normativa y regulación de la informática en el ámbito internacional.

Por lo que respecta al contexto internacional, son pocos los países que cuentan con una legislación adecuada; Estados Unidos, Alemania, Austria, Gran Bretaña, Holanda, Francia, España, Argentina y Chile son los que destacan.

- Estados Unidos - Este país adoptó en 1994 el Acta Federal de Abuso Computacional que modificó al Acta de Fraude y Abuso Computacional de 1986.

- Alemania - Sanciona, en 1986, la Ley contra la Criminalidad Económica, que contempla los siguientes delitos:
    - Estafa informática.
    - Espionaje de datos.
    - Alteración de datos.
    - Sabotaje informático.

- Austria - La Ley de reforma del Código Penal, sancionada el 22 de Diciembre de 1987, sanciona a aquellos que con dolo causen un perjuicio patrimonial a un tercero influyendo en el resultado de una elaboración de datos automática a través de la confección del programa, por la introducción, cancelación o alteración de datos o por actuar sobre el curso del procesamiento de datos. Además contempla sanciones para quienes comenten este hecho utilizando su profesión de especialistas en sistemas.

- Holanda - El 10 de Marzo de 1993 entró en vigencia la Ley de Delitos Informáticos, en la cual se penalizan los siguientes delitos:
    - El hacking.
    - El preacking (utilización de servicios de telecomunicaciones evitando el pago total o parcial de dicho servicio).
    - La ingeniería social (arte de convencer a la gente de entregar información que en circunstancias normales no entregaría).
    - La distribución de virus.

- Francia - En enero de 1988, este país dictó la Ley relativa al fraude informático, en la que se consideran aspectos como:
    - Intromisión fraudulenta que suprima o modifique datos
    - Conducta intencional en la violación de derechos a terceros que haya impedido o alterado el funcionamiento de un sistema de procesamiento automatizado de datos.
    - Conducta intencional en la violación de derechos a terceros, en forma directa o indirecta, en la introducción de datos en un sistema de procesamiento automatizado o la supresión o modificación de los datos que éste contiene, o sus modos de procesamiento o de transmisión.
    - Supresión o modificación de datos contenidos en el sistema, o bien en la alteración del funcionamiento del sistema (sabotaje).

- Chile - Fue el primer país latinoamericano en sancionar una Ley contra delitos informáticos, la cual entró en vigencia el 7 de junio de 1993. Esta ley se refiere a los siguientes delitos:
    - La destrucción o inutilización de los de los datos contenidos dentro de una computadora es castigada con penas de prisión. Asimismo, dentro de esas consideraciones se encuentran los virus.
    - Conducta maliciosa tendiente a la destrucción o inutilización de un sistema de tratamiento de información o de sus partes componentes o que dicha conducta impida, obstaculice o modifique su funcionamiento.
    - Conducta maliciosa que altere, dañe o destruya los datos contenidos en un sistema de tratamiento de información.

- Gran Bretaña - Debido a un caso de hacking en 1991, comenzó a regir en este país la Computer Misuse Act (Ley de Abusos Informáticos). Mediante esta ley el intento, exitoso o no, de alterar datos informáticos es penado con hasta cinco años de prisión o multas. Esta ley tiene un apartado que específica la modificación de datos sin autorización.

5. Plan de trabajo

A continuación se expondrán las tareas realizadas para este TFG y su estimación en tiempo invertido.

- Formación (80 horas): Aprender a utilizar el nuevo IDE Android Studio [22], entrenamiento en la programación en Java para Android y estudio de la interacción con el web service del sistema Kiu.

- Análisis (120 horas): Requisitos del sistema.

- Diseño (100 horas): Arquitectura del sistema, modelaje de la base de datos y desarrollo de las interfaces de usuario.

- Implementación (200 horas): Instalación y configuración de las herramientas de desarrollo, programación de las clases de modelo, vista y controlador en el cliente Android, programación de los ficheros del servidor destinados a atender las peticiones por parte del dispositivo móvil y programación de las conexiones y parseado de información por parte del servidor con el web service de Kiu y la base de datos MySQL.

- Pruebas (160 horas): Comprobación de todas las conexiones que intervienen en la arquitectura del sistema y testeo de los casos de uso. Modificación en aquellos casos en los que no se cumplía algún caso de uso

- Documentación (40 horas): Redacción de la memoria del TFG.

6. Recursos empleados

A continuación se expondrán los recursos, tanto hardware como software, empleados en el desarrollo y en el proceso de pruebas de este TFG. Como información complementaria, comentar que la inmensa mayoría del desarrollo del proyecto fue realizado en las oficinas del Departamento de Informática de Canaryfly.

6.1. Hardware

- PC con arquitectura de 64 bits, 8 GB de memoria RAM, procesador Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz y sistema operativo con núcleo Linux (Ubuntu 14.04.1 LTS).

- Teléfono móvil Samsung GALAXY Note 3, con pantalla de 5,7" y versión de Android 4.4.2 (KitKat).

- Teléfono móvil Samsung GALAXY ACE 2, con pantalla de 3,8" y versión de Android 2.3.6 (Gingerbread).

6.2. Software

- NetBeans IDE 8.0.1 [23]: Entorno de desarrollo integrado de código abierto y fundado por Sun MicroSystems en junio del año 2000. Es una herramienta para programadores con la que se pueden escribir, compilar, depurar y ejecutar programas.

- MySQL Workbench 6.2 [24]: Herramienta visual de diseño de bases de datos MySQL. Se utiliza para crear, diseñar, mantener y administrar dichas bases de datos.

- Mozilla Firefox 33.1.1 [25]: Navegador web de código abierto y libre.

- Apache 2.2 [26]: Servidor Web de código abierto.

- Android Studio 0.8.14: Entorno de desarrollo integrado (IDE) para la plataforma Android.

- Android SDK 23.0.5 [27]: Kit de desarrollo que incluye un conjunto de herramientas, tales como un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales.

- Draw.io [28]: Aplicación web para dibujar todo tipo de diagramas.

- Intérprete de órdenes del sistema Ubuntu [29] : Un terminal es una forma de acceder al sistema sin utilizar la interfaz gráfica, es decir, realizar todo tipo de tareas en formato texto. La forma de utilizar el sistema de este modo es mediante órdenes.

6.3. Otras herramientas

- PHP 5.4 [30]: Lenguaje de programación diseñado originalmente para el desarrollo web, y cuyo código se ejecuta del lado del servidor.

- MySQL 5.4 [31]: Sistema de gestión de bases de datos.

- XML (Extensible Markup Language) [32]: Lenguaje de marcas desarrollado para poder almacenar información de forma fácilmente legible.

- JSON (JavaScript Object Notation) [33]: Formato ligero para el intercambio de datos.

- cURL [34]: Herramienta para realizar conexiones entre servidor para intercambiar información.

7. Análisis

En este capítulo se estudiará los requerimientos que el proyecto debe satisfacer, tanto los actores que intervienen con el sistema como sus casos de uso.

El análisis se centrará, principalmente, en el estudio de la actual página web de Canaryfly (www.canaryfly.es) y de adquirir de ella todas las funcionalidades posibles, y las que se podrían mejorar, para implantarlas en una aplicación Android.

7.1. Obtención de requisitos

El primer paso para comenzar con el desarrollo es conocer lo siguiente:

- ¿Cuál es el problema? ¿Tiene solución?
- ¿Cuál es el contexto?
- ¿Qué se necesita para solucionar el problema?

Los requisitos son quienes dan respuesta a esta pregunta, por lo que es de vital importancia obtenerlos antes que nada. De entre las diferentes técnicas que hay, para este proyecto se utilizaron las entrevistas entre los desarrolladores de la empresa.

En sus inicios, hace alrededor de dos años, Canaryfly contrato los servicios de una empresa externa para que les construyeran una primera página web. Ésta era bastante limitada y tenía numerosos fallos, por lo que meses más tarde se decidió contratar un grupo de desarrolladores web para obtener una página corporativa propia y 100% personalizable.

En su momento, debido a los plazos que tuvieron que cumplirse, no hubo tiempo de analizar, diseñar y documentar el desarrollo de la web de una manera eficiente por lo que se empezó directamente por la implementación, lo que en el futuro acarreo abundantes problemas. A lo largo de este año se han ido desarrollando varias funcionalidades nuevas pero siempre sin un estudio previo que convierta el producto en un sistema robusto.

Actualmente la empresa ha logrado una estabilidad en la prestación de sus servicios y el departamento de informática está en proceso de iniciar un proyecto de reestructuración de todo el sistema informático. En esta reestructuración ya se tiene en cuenta la creación de aplicaciones para dispositivos móviles.

La empresa tiene tres grandes grupos de servicios, la web general para los clientes, la web con funcionalidades especiales para que las agencias de viaje puedan llevar un control de sus ventas, y una serie de aplicaciones web para uso interno del resto de departamentos de la organización (facturación, programación de vuelos, etc). Tras discutirlo, llegamos a la conclusión de que la aplicación móvil sólo debía proporcionar las funcionalidades de la web general, ya que no tenía mucho sentido en el resto de categorías.

Una vez decidido el objetivo y el público al que nos dirigimos, es más sencillo escarbar sobre las funcionalidades a desarrollar. La gestión del contenido de la web debe ser el mismo que el de la aplicación Android, por lo que se centralizará la administración del sistema y se unificarán las bases de datos.

7.2. Análisis de requisitos

En lo que al análisis de requisitos se refiere, ha habido que darle muchas vueltas a todos los procesos ya que en muchas ocasiones ocurrió tener decidido algo y que esto tuviera que cambiarse debido a diferentes factores.

De entre todos estos cambios, el más importante y al que más tiempo hubo que dedicarle fue a la implantación de nuevas tarifas en el mes de Octubre de 2014. Antes a esta modificación, las tarifas ofrecidas por la página web era estáticas, o dicho de otra forma, no variaban con el tiempo. Estas nuevas tarifas ofrecen un precio u otro en relación al número de horas que falten para un vuelo concreto. Todo esto afecta a la

información que se obtiene al seleccionar un itinerario, que es, junto con el proceso de compra, una de las funcionalidades más importantes.

Más allá de esto, hubieron más modificaciones en lo referente al proceso de reserva de vuelos por requerimiento del Ministerio de Fomento, pero no afectan al primer prototipo que compone este TFG. Todo esto se comentará en el capítulo "Futuro del proyecto".

Una de las principales mejoras que se quiere implantar en la página web es la creación de un sistema de usuarios para sus clientes, ya que actualmente no posee ninguno.

Aún se está decidiendo que funcionalidades se les va a ofrecer a todos aquellos usuarios registrados. Algunas posibilidades son un histórico de sus reservas o un autocompletado en los formularios de datos de pasajeros a la hora de comprar un billete, pero desde la empresa le han dado especial importancia al sistema de puntos relacionado a un pasajero. Acumular puntos por cada compra y ofrecer diferentes descuentos en base a los puntos acumulados es un objetivo prioritario. El problema radica en los trámites administrativos que esto conlleva, en la empresa aún no se ha llegado a un consenso. No se han decidido qué ofertas pueden entrar este sistema y además hay muchas promociones que aún no se han cerrado con diferentes empresas.

Todo esto hace que actualmente el desarrollo del sistema de usuarios esté paralizado y por lo tanto, no se haya podido desarrollar para este TFG, ya que como es obvio, todo usuario registrado por la web debe poder usar la aplicación móvil con su usuario y viceversa, y hasta que nos se nos proporcione información detallada sobre cómo la empresa desea este sistema de puntos, no podemos proporcionar ningún diseño.

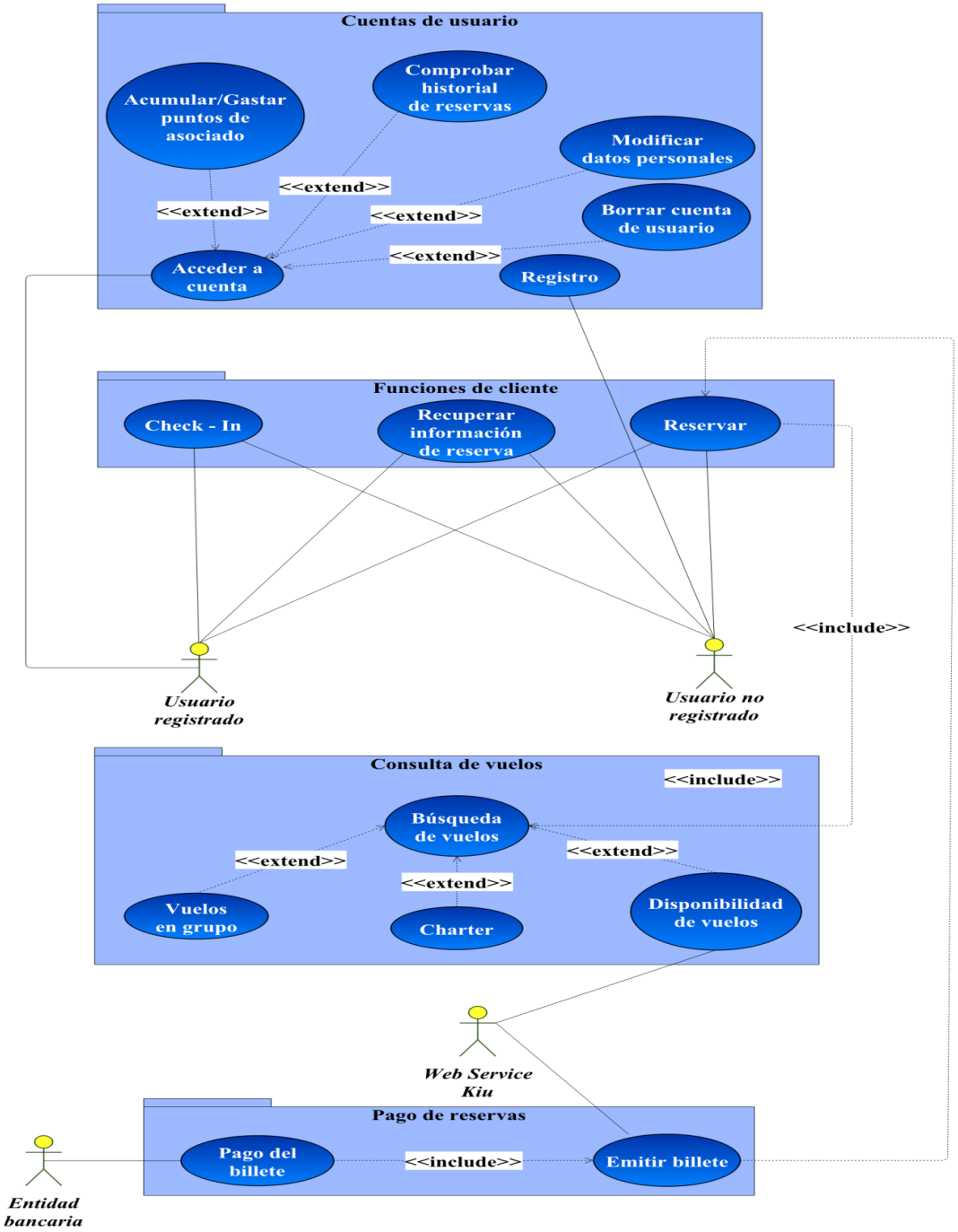
La posibilidad de realizar la facturación online es algo que se ha implantado en la web junto con las nuevas tarifas, dígase el mes pasado. En este TFG también se ha dado la posibilidad de hacerlo pero se especificará más sobre su desarrollo en el capítulo de "Implementación" y también sobre los cambios que se pretenden hacer en el capítulo "Futuro del proyecto".

El pasado 1 de Noviembre se impuso como condición obligatoria a todas las compañías de transporte, tanto marítimas como aéreas, validar la condición de residencia de sus clientes mediante el sistema SARA [35] (Sistema de Acreditación de Residencia Automático) que el Ministerio de Fomento ofrece a todas estas compañías y así evitar que los pasajeros tengan que poseer un certificado impreso a la hora de viajar. Actualmente la página web realiza dicha verificación pero desde fomento nos han informado que el proceso de verificación no se hace correctamente. Canaryfly había decido realizar dichas consultas al sistema SARA antes de la emisión y cobro de un billete debido a unas limitaciones de Kiu, pero nos exigen que se haga después, lo cual implica que Kiu debe subsanar dichas carencias. Esto, al igual que en el caso del sistema de usuarios, imposibilita desarrollar dicha funcionalidad correctamente hasta que se resuelvan unos factores ajenos a Canaryfly.

La pasarela de pago actual también sufre de bastantes incidencias y la empresa se encuentra en proceso de cambio, lo cual imposibilita su desarrollo hasta nuevo aviso.

El resto de funcionalidades que ofrece la web, como la posibilidad de pedir presupuesto para vuelos especiales (charter y vuelos en grupo) o recuperar información de una reserva han podido ser desarrolladas sin mayores inconvenientes y se explicará más al detalle en el capítulo de "Implementación".

7.3. Especificación de requisitos



Modelo de casos de uso

7.3.1. Actores

Una vez hemos analizado todos los requisitos detalladamente, podemos definir nuestros actores [36] y sus funcionalidades, conformando nuestro modelo de casos de uso [37].

Los actores representan un rol jugado por un usuario o por cualquier otro sistema externo que interactúe con el sistema. Los que se han identificado son los siguiente:

- Usuario registrado: Aunque aún no se ha desarrollado, este actor cobrará un papel importante en el futuro de la aplicación de ahí que se incluya en este modelo de casos de uso.

- Usuario no registrado: Actualmente, al no haber sistema de usuarios, todos los usuarios que interactúen con la aplicación serán usuarios no registrados, los cuales tienen acceso a las funcionalidades generales de la aplicación.

- Web Service Kiu: El sistema de Kiu es el encargado de devolver información y realizar procesos conforme a los parámetros que se les pase. Su interacción con la aplicación y su correcto tratamiento de datos es vital para dar un buen soporte.

- Entidad bancaria: Al igual que el actor "Usuario registrado", este actor no se ha desarrollado pero en el futuro será quien se encargue de todo el proceso de pago.

7.3.2. Casos de uso

Por último, explicaremos nuestro modelo de casos de uso del UML [38] (Unified Modelling Language), el cual no es más que un diagrama de comportamiento, que define la naturaleza del caso de uso, pudiéndole asociar un actor.

- Facturación online: Esta funcionalidad tiene que ver con ese servicio tan común en las compañías aéreas que consiste en poder imprimir la tarjeta de embarque sin necesidad de pasar por la facturación del aeropuerto, ahorrando tiempo y teniendo siempre de una manera sencilla el código necesario para pasar el control de seguridad del aeropuerto.

  Lo único que necesitamos es el código de reserva, también conocido como localizador o PNR, que se nos proporciona una vez compramos un billete, y el/los apellido/s asociados a dicho código de reserva.

- Consulta de información de una reserva: A un usuario registrado se le mostraría una lista de sus reservas, pudiendo seleccionar la que prefiera. A los no registrados se les presenta un formulario en el que deben disponer, su código de reserva, el documento de identidad asociado a esa reserva y el origen de su viaje. Aplicando estos datos se obtendría una serie de datos que explicaremos más en detalle en el capítulo de implementación.

- Solicitud de presupuesto para vuelos especiales: Canaryfly ofrece la posibilidad de pedir presupuesto para diferentes tipos de vuelos fuera de los canales habituales. La vía para solicitar estos presupuestos es rellenar una serie de formularios proporcionando información relativa al vuelo que se desea realizar. El sistema envía un correo electrónico con todos estos datos a la persona designada para negociar un presupuesto con el interesado. Actualmente hay dos tipos de vuelos especiales:

    - Vuelos en grupo [39]: Por política de la empresa, una reserva puede contener un máximo de 9 pasajeros. Para aquellos clientes que estén interesados en viajar con un grupo mayor de 9 personas han de solicitarlo.

    - Chárter [40] : Además de los vuelos en grupo, pueden solicitarse vuelos chárter, que es un servicio de alquiler de un avión fuera del horario habitual o para llevar a un grupo de personas exclusivo, como puede ser un equipo de fútbol [41].

- Reservas: En este primer prototipo trabajamos con la pedida de disponibilidad de vuelos, dejando para el futuro el desarrollo de recogida de información de los pasajeros, validación de la residencia de éstos, pago y emisión del billete.

  En este punto hay numerosas consideraciones a tener en cuenta, en el capítulo de "Implementación" se explicarán al detalle uno por uno pero. Los mencionamos:

    1. Posibilidad de pedir vuelo de "Ida y Vuelta" o "Sólo Ida".
    2. Seleccionar un origen de entre nuestras 9 opciones.
    3. Proporcionar un aeropuerto de destino en relación a las rutas asociadas al origen seleccionado por el usuario.
    4. Dar la opción de elegir fecha de vuelo, deshabilitando aquellas fechas en las que la ruta seleccionada no tenga vuelos.
    5. No permitir que las fecha de vuelta sea anterior a la de ida.
    6. Controlar que como mínimo haya un pasajero seleccionado.
    7. Tener en cuenta si la ruta es un viaje nacional o internacional para ofrecer o no, la opción de aplicar descuentos de residencia y familia numerosa, además de las tarifas sociales disponibles.
    8. No permitir más de 9 pasajeros.
    9. Cumplir que haya al menos un adulto por varios niños y un adulto por cada bebé, no permitiendo más de dos bebés por reserva.
    10. Si hay niños o bebés en la reserva se ofrecerán ofrecer las tarifas sociales correspondientes.
    11. Obtener horarios, tarifas, precios y números de plaza de los vuelos.

8. Diseño

A continuación se van a exponer los tres campos relacionados con el diseño de la aplicación, la arquitectura del sistema, las interfaces gráficas y el modelaje de la base de datos.

8.1. Arquitectura del sistema

La arquitectura ha sido dividida en tres capas [42], buscando la separación de la lógica de negocios de la lógica de diseño.

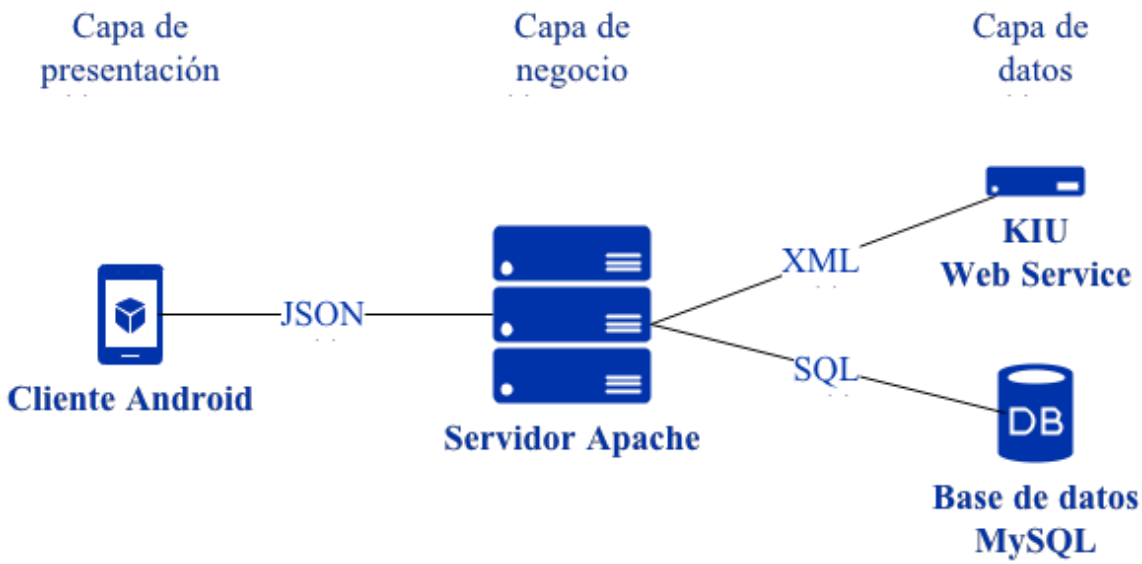


Imagen de la arquitectura del sistema

8.1.1. Capa de presentación

Representación gráfica del programa frente al usuario, la cual debe ser amigable y fácil de usar. En nuestro caso, las interfaces de usuario dispuestas por los dispositivos móviles con sistema operativo Android serán las encargadas de acceder y controlar los datos y los servicios de los objetos.

Como se mencionó en el primer capítulo de este documento, uno de los objetivos era adquirir conocimientos sobre el diseño gráfico en Android para ser capaces de adquirir un trabajo 100% personalizable. Esto quiere decir que todo elemento gráfico, dígase botones, formularios o diálogos pueden ser modificados y no es necesario utilizar los que vienen por defecto. Se hace especial hincapié en este punto dado que, en algunos casos concretos, la personalización de un elemento no es nada trivial.

Se han usado la misma fuente (Montserrat) y el mismo color (#0033A9) de la página web corporativa de Canaryfly buscando un diseño distintivo.

Se hace necesario explicar que el desarrollo gráfico de Android, aunque se puede hacer programáticamente, preferentemente se realiza mediante ficheros XML, por lo que podemos crear nuestros propios estilos para ser utilizados en varios elementos. A continuación mostramos, a modo de ejemplo, el estilo empleado para los botones de la aplicación:

```
<style name="button_cnf" parent="@android:style/Widget.Button">
    <item name="android:textSize">16sp</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textColor">#FFFFFF</item>
    <item name="android:gravity">center</item>
    <item name="android:layout_centerHorizontal">true</item>
    <item name="android:layout_gravity">center_horizontal</item>
    <item name="android:shadowColor">#000000</item>
    <item name="android:shadowDx">1</item>
    <item name="android:shadowDy">1</item>
    <item name="android:shadowRadius">0.6</item>
    <item name="android:background">@drawable/button_cnf</item>
    <item name="android:padding">10dp</item>
</style>
```

Imagen del contenido del fichero values/styles.xml

Los elementos que componen este fichero son bastante intuitivos, pero vamos a ser especial mención al elemento “background”, el cual utiliza otro fichero XML encargado de darle la forma que nos interesa, tanto el estado estático como el correspondiente efecto cuando lo pulsamos.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true">
        <shape android:shape="rectangle">
            <corners android:radius="3dp" />
            <stroke android:width="1dp" android:color="#00267F" />
            <gradient android:angle="-90" android:endColor="#0142D8" android:startColor="#060D30" />
        </shape>
    </item>
    <item>
        <shape android:shape="rectangle">
            <corners android:radius="3dp" />
            <stroke android:width="1dp" android:color="#00267F" />
            <gradient android:angle="-90" android:endColor="#1B52D2" android:startColor="@color/canaryfly" />
        </shape>
    </item>
</selector>
```

Imagen del contenido del fichero drawables/button_cnf.xml

En este punto le damos forma rectangular, redondeamos las esquinas, aplicamos unos bordes y le añadimos un degradado. El resultado es el siguiente:

De esta forma, ya tenemos un estilo común para todos los botones de la aplicación. Lo único que habría que hacer es declarar qué estilo se quiere utilizar en la definición de un botón en su respectivo fichero XML.

```
<Button
    android:id="@+id/button_booking_submit"
    style="@style/button_cnf"
    android:text="Buscar vuelos" />
```

Imagen de la declaración de un botón en un fichero XML aplicando un estilo

Esta es la filosofía a la hora de diseñar gráficamente aplicaciones Android. Elementos no tan básicos, como pueden ser los diálogos, requieren de programación en Java para su correcto manipulado. Del mismo modo ocurre cuando queremos utilizar una fuente externa a las que Android ofrece por defecto, no es posible aplicarlas desde XML.

```
// Submit Button //
Button submitBooking = (Button) findViewById(R.id.button_booking_submit);
submitBooking.setTypeface(Typeface.createFromAsset(getAssets(), "font/Montserrat-Regular.ttf"));
```

Imagen de cómo aplicar una fuente externa a un elemento visual programáticamente

Como podemos apreciar en este ejemplo, creamos una instancia del objeto “Button” y lo referenciamos a un botón visual existe en un fichero XML mediante la función “findViewById()” y su correspondiente id. Una vez hecho esto le aplicamos nuestra fuente, la cual está localizada en la carpeta “asserts”, usando la función “setTypeface()”.

Con estos ejemplos abarcamos el diseño gráfico por las dos vías existentes, XML y programación Java.

El mayor problema con el que nos topamos es la gran cantidad de dispositivos que existen con diferentes tamaños de pantalla, lo cual hace muy difícil conseguir que la misma aplicación se vea igual en todas sus pantallas. Android resuelve esto creando varias categorías distintas de tamaño de pantallas [43], pequeño, mediano, grande, extra grande y extra extra grande, para el caso de los tables. Por lo que un diseño profesional tendría que realizar más de un diseño distinto e implantarlos de la siguiente manera para que el propio dispositivo, al conocer el tamaño de su pantalla, sepa cual tiene que elegir.

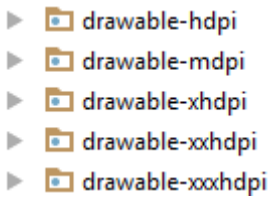


Imagen de la estructura del proyecto Android. Recursos visuales para varias pantallas

Dado que el objetivo era adquirir los conocimientos y no realizar el diseño final de la aplicación, sólo se ha usado un único diseño. La creación de un diseño más amigable y que funcione bien en todos los dispositivos es una tarea que se deja para el futuro.

Lo mismo ocurre para el caso en el que el móvil está en posición vertical u horizontal, la mejor práctica sería, además de tener en cuenta lo anterior, realizar un diseño personalizado para cuando el móvil está en una posición o en otra. Aunque aquí nos topamos con otro inconveniente. Cuando giramos el móvil estamos finalizando el ciclo de vida de la pantalla actual e iniciando otro nuevo, lo cual provoca que podamos perder información. Para evitar esto hay que tener en cuenta una serie de consideraciones en Java que, al igual que la creación de un mejor diseño, se deja como tarea futura, actualmente la aplicación se ha capado para que no pueda girarse y únicamente se vea en posición vertical, la cual es la más común en los teléfonos móviles.

Una vez comentados todos los aspectos importantes referentes al diseño de la aplicación, se procederá a exponer las diferentes pantallas de la interfaz de usuario, las cuales son funcionales pero no definitivas.

- Icono de la aplicación: El icono actual, al igual que el diseño entero, no es definitivo, pero fue una primera opción creada por el mismo asistente de Android Studio cuando se crea el proyecto.

Imagen del icono de la aplicación Canaryfly

- Pantalla de bienvenida: Al seleccionar la aplicación lo primer que aparece es un "splash screen" con el logo de Canaryfly durante un par de segundos.

Imagen de la pantalla de bienvenida

- Pantalla inicial: Una vez la pantalla de bienvenida desaparece, tenemos el principal menú desde el cual podemos realizar diferentes tareas. La opción "Conectar" hace referencia al sistema de usuarios, por lo que está inutilizada.

Imagen de la pantalla inicial de la aplicación

- Pantalla chárter: Formulario para solicitar presupuesto de vuelos chárter.

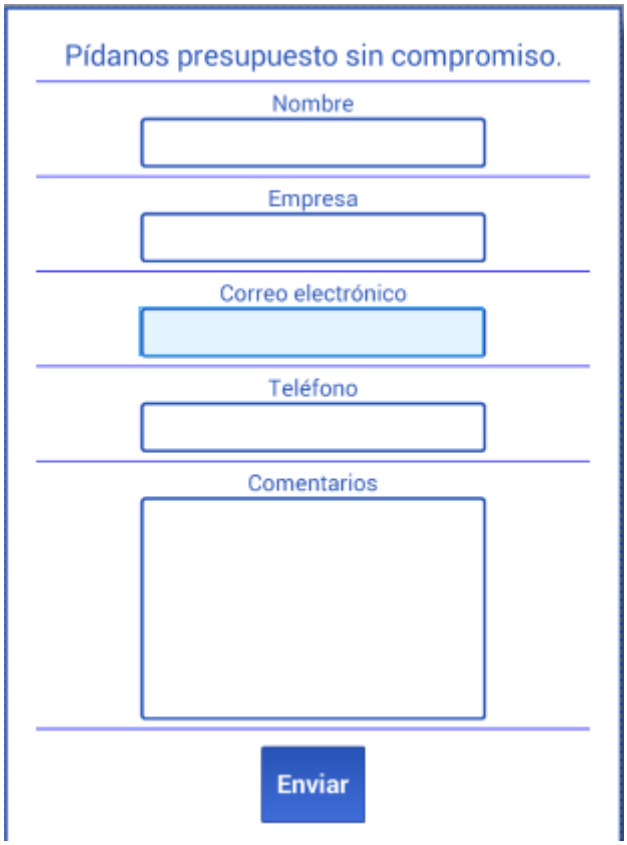


Imagen de la pantalla para solicitud de vuelos chárter

- Pantalla vuelos en grupo: Formulario para solicitar presupuesto de vuelos en grupo. Este formulario es más extenso que el anterior, por lo que se ha usado un scroll vertical para poder acceder a todos los campos.

Imagen de la pantalla de solicitud de vuelos en grupo

- Pantallas mis reservas: Formulario de recuperación de datos de reserva. Una vez los datos introducidos son validados y enviados, se recupera la información de la reserva, la cual mostramos en un dialogo a modo de tabla.

Imagen de la pantalla del formulario "Mis reservas"

Imagen de la tabla con los datos de la reserva

- Pantalla facturación online: Formulario para realizar la facturación online.

Imagen del formulario para realizar la facturación online

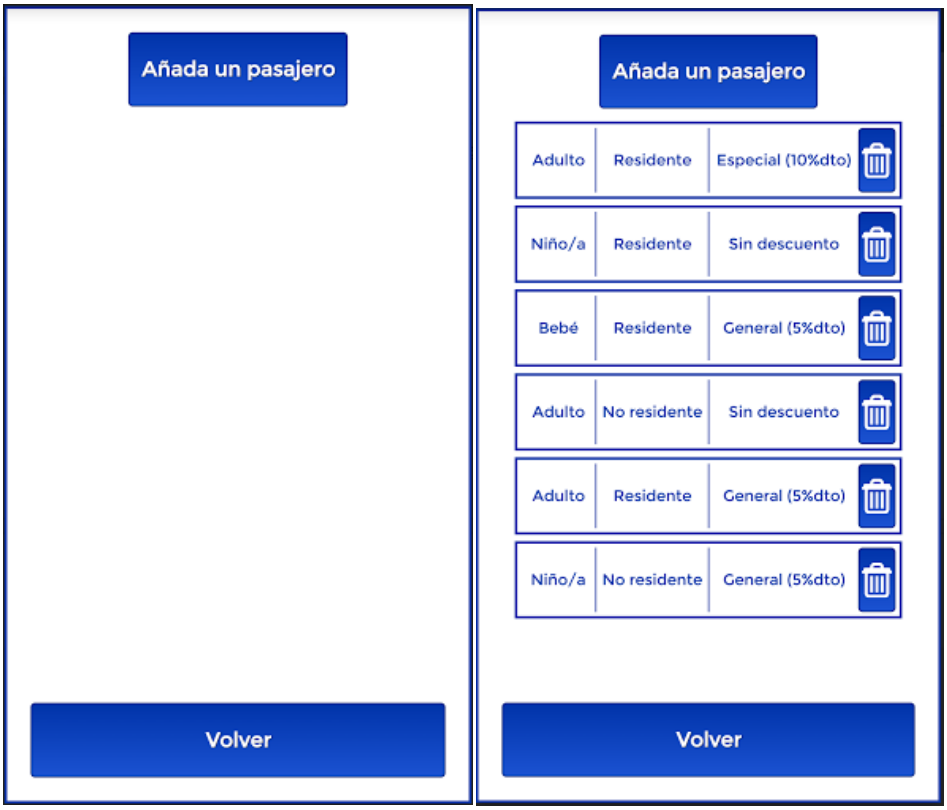
- Pantalla reservas: Formulario para introducir los datos necesarios para buscar disponibilidad de vuelos.

Imagen de la pantalla del formulario de reservas

- Pantalla calendario: Para seleccionar el día de vuelo se utiliza un calendario personalizado y no el que viene por defecto.

Imagen del calendario usado para seleccionar días de vuelo

- Pantalla selección de pasajeros: Los pasajeros pueden ser de varios tipos y pueden aplicar diferentes descuentos, por lo que se optó por hacerlo en una pantalla aparte. Al pulsar el botón “Añada un pasajero” se abre un dialogo donde podemos seleccionar los datos del pasajero. En este caso también usamos un scroll vertical por si el número de pasajeros fuera alto.



Imágenes de las pantallas de la selección de pasajeros

Imagen del dialogo para introducir datos de un pasajero

- Pantalla de disponibilidad de vuelos: Nuevamente utilizamos un scroll vertical para poder acceder a toda la información de los vuelos. Dependiendo de lo itinerario seleccionado puede haber más de una opción, por lo que se escogió un sistema de "tabs" para seleccionar las diferentes opciones, tanto pulsado en el botón superior o arrastrando con el dedo de izquierda a derecha.

Imagen de la pantalla de información de disponibilidad de vuelo

8.1.2. Capa de negocio

Esta capa sirve de intermediaria entre la interfaz de usuario y la capa de datos.

En nuestro contexto, la aplicación envía unos parámetros al servidor Apache en formato JSON, este los recibe y decide qué hacer. Las comunicaciones con la base de datos MySQL se llevan a cabo sin mayor complicación, ya que ambos sistemas están en el mismo servidor. Cuando requiere conectarse con el sistema Kiu, lo hace a través de ficheros XML que explicaremos a continuación en la capa de datos. Una vez el servidor obtiene una respuesta, la parsea y general un mensaje en formato JSON que se envía al dispositivo móvil, donde este la recibe y utiliza.

8.1.3. Capa de datos

8.1.3.1. KIU Web Service

Los WebServices son la interfaz que permite acceder en forma programática al inventario del sistema de reservas desde la disponibilidad hasta la emisión.

KIU deja del lado de la compañía aérea la responsabilidad de implementar los requerimientos para las conexiones mediante REST, el cual es un modelo de arquitectura que utiliza XML y HTTP. Por lo tanto es responsabilidad del implementador crear mensajes XML válidos e interpretar adecuadamente los mensajes XML que recibe en respuesta. A continuación se procederá a explicar la metodología de las conexiones con el servicio. Antes que nada mencionar que al trabajar con PHP hemos utilizado cURL para establecer las conexiones.

La comunicación se iniciará siempre del lado de la línea aérea o agencia de viaje y consiste en una petición POST vía HTTPS conteniendo 3 campos, a saber:

- user: Usuario asignado para acceso a los webservices.
- password: Clave de acceso asignada para los webservices.
- request: Texto XML con el mensaje cuyo procesamiento se solicita.

Las validaciones sobre los datos enviados se implementan en la forma más restrictiva posible, por lo que es importante tenerlas en cuenta a fin de evitar rechazos por cuestiones de formatos.

Los formatos reconocidos son:

- Entero: es un entero positivo o negativo sin espacios ni ningún otro símbolo especial que el de signo menos (-) al inicio de ser negativo. Ej. 123, 0, -44.

- Decimal: sigue el formato del entero pero admite un punto decimal (.) y una segunda sucesión de dígitos. Ej. -44, 0.00038, -456.55.

- Alfabético: es una cadena de caracteres de la A a la Z mayúsculas sin eñes, acentos, espacios ni ningún símbolo de puntuación. Ej. LIM, BASED.

- Alfanumérico: es una cadena de caracteres de la A a la Z y del 0 al 9. Ej. 29FEB, JFK0XX01.

- Alfanumérico con caracteres especiales: es una cadena que acepta los caracteres de la definición de Alfanumérico y anexa los caracteres especiales definidos en el standard DISH 20.1, a saber: espacio en blanco, punto decimal, guión medio y barra de fecha. Ej. PEREZ/JUAN.

- Fecha: Definición Calendar Date según ISO 8601 (YYYY-MM-DD) . Ej. 2013-03-27.

- Fecha y Hora: Definición Date and Time según ISO 8601 (YYYY-MM-DDThh:mm:ss). Ej. 2013-03-27T22:45:36.

- Lista: En los casos de listas definidas se enumeraran entre corchetes separadas por el carácter pipe. Ej. [Production|Testing].

En los casos de campos opcionales estos aceptan además de su formato reconocido el recibir una cadena vacía.

Para cada funcionalidad existe un par de mensajes, uno que identifica la petición y uno que le corresponde en respuesta. El identificador del mensaje corresponde al elemento raíz del XML. Este par se halla normalmente asociado por nombre. La única funcionalidad con la que hemos trabajado en este TFG es la función AirAvail, la cual es la que gestiona la información de disponibilidad de vuelo y se explicará en detalle en el Anexo 1.

Todo mensaje de petición incluye en su cabecera un set de atributos que facilita su identificación y procesamiento. Estos elementos son:

- EchoToken: Alfanumérico [0...128] (Opcional). Sirve como identificación y se envía junto con la respuesta sin modificación alguna.

- TimeStamp: Fecha y Hora ISO. Momento de generación del mensaje.

- Target: [Production|Testing]. Partición sobre la cual va a trabajarse.

- Version: Decimal (Opcional). Versión implementada.

- SequenceNmbr: Entero positivo. Sirve para identificar un número de secuencia en un set de mensajes relacionados. Se envía junto con la respuesta sin modificación alguna.

Ejemplo de inicio de petición:

```
    <KIU_AirAvailRQ EchoToken="12345" TimeStamp="2010-07-
17T09:30:47-02:00" Target="Production" Version="3.0"
SequenceNmbr="1">
```

Imagen de una petición al web service de Kiu

Además de los elementos comunes en las peticiones, las respuestas incluyen elementos para informar el éxito o fracaso de las operaciones realizadas.

En caso de éxito se incluye el elemento Success junto con los datos de la respuesta. De esta forma el encabezado de la respuesta para una petición seria:

```
    <KIU_AirAvailRQ EchoToken="12345" TimeStamp="2010-07-
17T09:30:47-02:00" Target="Production" Version="3.0"
SequenceNmbr="1">
    <Success/>
```

Imagen de una respuesta exitosa por parte del web service de Kiu

Mientras que en caso de fracaso se incluye un elemento Error el cual incluye un ErrorCode con el código numérico y un ErrorMsg con una descripción.

```
<KIU_AirAvailRQ EchoToken="12345" TimeStamp="2010-07-17T09:30:47-02:00"
Target="Production" Version="3.0" SequenceNmbr="1">
<Error>
      <ErrorCode>55023</ErrorCode>
      <ErrorMsg>Test error message</ErrorMsg>
</Error>
```

Imagen de una respuesta errónea por parte del web service de Kiu

A continuación procederemos a explicar las dos funciones utilizadas para este TFG:

KIU_AirAvailRQ/RS

Mensaje de consulta de disponibilidad. Obtiene la lista de opciones disponibles desde un origen hacia un destino en una fecha determinada.

Modelo XML para la solicitud:

```
<?xml version="1.0" encoding="UTF-8"?>
<KIU_AirAvailRQ EchoToken="1" TimeStamp="2012-04-20T15:45:07-03:00"
Target="Testing" Version="3.0" SequenceNmbr="1" PrimaryLangID="en-us"
DirectFlightsOnly="false">
      <POS>
            <Source AgentSine="NET00XXWW" TerminalID="NET00XX000">
            </Source>
      </POS>
      <SpecificFlightInfo>
            <Airline Code="XX"/>
      </SpecificFlightInfo>
      <OriginDestinationInformation>
            <DepartureDateTime>2012-05-07</DepartureDateTime>
            <OriginLocation LocationCode="AEP"/>
            <DestinationLocation LocationCode="COR"/>
      </OriginDestinationInformation>
      <TravelPreferences >
            <CabinPref Cabin="Economy"/>
      </TravelPreferences>
      <TravelerInfoSummary>
            <AirTravelerAvail>
                  <PassengerTypeQuantity Code="ADT" Quantity="1"/>
            </AirTravelerAvail>
      </TravelerInfoSummary>
</KIU_AirAvailRQ>
```

Imagen del modelo XML de solicitud de disponibilidad de vuelo

En la imagen anterior podemos apreciar un ejemplo de solicitud. Los campos a destacar son el Target, donde especificamos si esa consulta pertenece a un entorno real o de testeo. AgentSine y TerminalID son datos identificativos, al igual que Airline Code, que en el caso de Canaryfly es "PM".

Hay más elementos pero los realmente importantes son DepartureDateTime donde declaramos la fecha de búsqueda de disponibilidad, OriginLocation donde especificamos el origen del vuelo y DestinationLocation, donde especificamos el destino del vuelo. Cabe destacar que éstas consultas trabajan con los códigos IATA [45] de los aeropuertos, que son códigos distintivos entre todos los aeropuertos del mundo.

En el caso de Canaryfly, los aeropuertos con los que trabaja a fecha de la realización de este TFG son: Gran Canaria (LPA), Fuerteventura (FUE), Lanzarote (ACE), La Palma (SPC), Tenerife Norte (TFN), El Aiun (AUN), Guelmin (GLN), Nouadhibou (NDB) y Dakhla (VIL)

Modelo XML para la respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<KIU_AirAvailRS EchoToken="" TimeStamp="2012-04-20T18:45:10+00:00" Target=""
Version="3.0" SequenceNmbr=""><Success/>
      <OriginDestinationInformation>
            <DepartureDateTime>2012-05-07</DepartureDateTime>
            <OriginLocation>AEP</OriginLocation>
            <DestinationLocation>COR</DestinationLocation>
            <OriginDestinationOptions>
                  <OriginDestinationOption>
                        <FlightSegment DepartureDateTime="2012-05-07 18:20:00"
ArrivalDateTime="2012-05-07 19:50:00" StopQuantity="0" FlightNumber="3136"
JourneyDuration="01:30:00">
                              <DepartureAirport LocationCode="AEP"/>
                              <ArrivalAirport LocationCode="COR"/>
                              <Equipment AirEquipType="737" />
                              <MarketingAirline CompanyShortName="XX"/>
                              <Meal MealCode="S"/>
                              <BookingClassAvail ResBookDesigCode="Y"
ResBookDesigQuantity="9"/>
                              <BookingClassAvail ResBookDesigCode="B"
ResBookDesigQuantity="9"/>
                              <BookingClassAvail ResBookDesigCode="M"
ResBookDesigQuantity="9"/>
                        </FlightSegment>
                  </OriginDestinationOption>
                  <OriginDestinationOption>
                        <FlightSegment DepartureDateTime="2012-05-07 18:30:00"
ArrivalDateTime="2012-05-07 19:35:00" StopQuantity="0" FlightNumber="3144"
JourneyDuration="01:05:00">
                              <DepartureAirport LocationCode="AEP"/>
                              <ArrivalAirport LocationCode="COR"/>
                              <Equipment AirEquipType="SF3" />
                              <MarketingAirline CompanyShortName="XX"/>
                              <Meal MealCode="S"/>
                              <BookingClassAvail ResBookDesigCode="Y"
ResBookDesigQuantity="9"/>
                              <BookingClassAvail ResBookDesigCode="B"
ResBookDesigQuantity="9"/>
                              <BookingClassAvail ResBookDesigCode="H"
ResBookDesigQuantity="9"/>
                              <BookingClassAvail ResBookDesigCode="N"
ResBookDesigQuantity="9"/>
                              <BookingClassAvail ResBookDesigCode="X"
ResBookDesigQuantity="9"/>
                        </FlightSegment>
                  </OriginDestinationOption>
            </OriginDestinationOptions>
      </OriginDestinationInformation>
</KIU_AirAvailRS>
```

Imagen del modelo XML de respuesta de disponibilidad de vuelo

En la imagen anterior podemos ver un ejemplo de mensaje de respuesta por parte de KIU. Al igual que en la solicitud, la fecha de disponibilidad, el origen y el destino vienen adjunto como campos al principio del XML.

Los elementos que merecen la pena ser destacados son los siguientes:

- OriginDestinationOption, que contiene varios elementos FlightSegment, tantos como vuelos haya en el día seleccionado para recuperar una disponibilidad. Es necesario aclarar que la imagen anterior pertenece a las solicitudes que se hacían en la versión 2 del web service, actualmente se trabaja con la 3 y existe un único elemento OriginDestinationOption, no varios como se muestra en la imagen.

- FlightSegment guarda toda la información de todos los vuelos que puede haber en una ruta, incluyendo las escalas, por lo que para cada vuelo incluido en esta etiqueta existe la siguiente información:
    - Número de vuelo (FlightNumber): 3136 y 3144
    - Hora de salida (DepartureDateTime): 18:20 y 18:30
    - Hora de llegada (ArrivalDateTime): 19:50 y 19:35
    - Duración del vuelo (JourneyDuration): 01:30 y 01:50 *(horas:minutos)
    - Si trabajamos con vuelos con escala, el origen cambiará en algún vuelo con respecto al origen de nuestra ruta (DepartureAirport)
    - Lo mismo ocurre con el destino, si nuestro vuelo tiene escalas algún destino no será exactamente el de nuestra ruta (ArrivalAirport)
    - Tipo de avión (Equipment): 737 y SF3
    - La compañía de vuelo (MarketingAirline)
    - El tipo de merienda que se servirá en el avión:
    - Las tarifas disponibles (BookingClassAvail). Este elemento devolverá los códigos de las tarifas disponibles (ResBookDesigCode) y el número de plazas libres en el momento de la consulta (ResBookDesigQuantity)

8.1.3.2. Base de datos relacional MySQL

Actualmente las bases de datos de Canaryfly no cumplen con el modelo relacional. Entra dentro de la restructuración del sistema informático de la empresa el desarrollar una base de datos mucho más eficiente y robusta que la que está implantada actualmente.

En este proyecto se ha credo una base de datos relacional que pretende cubrir las carencias presentes.

En el Anexo 2 se especifican todas las sentencias que han sido necesarias para la creación de esta base de datos relacional MySQL.

A continuación se presenta el diagrama con todas las entidades creadas y todos los enlaces que las relacionan entre sí.

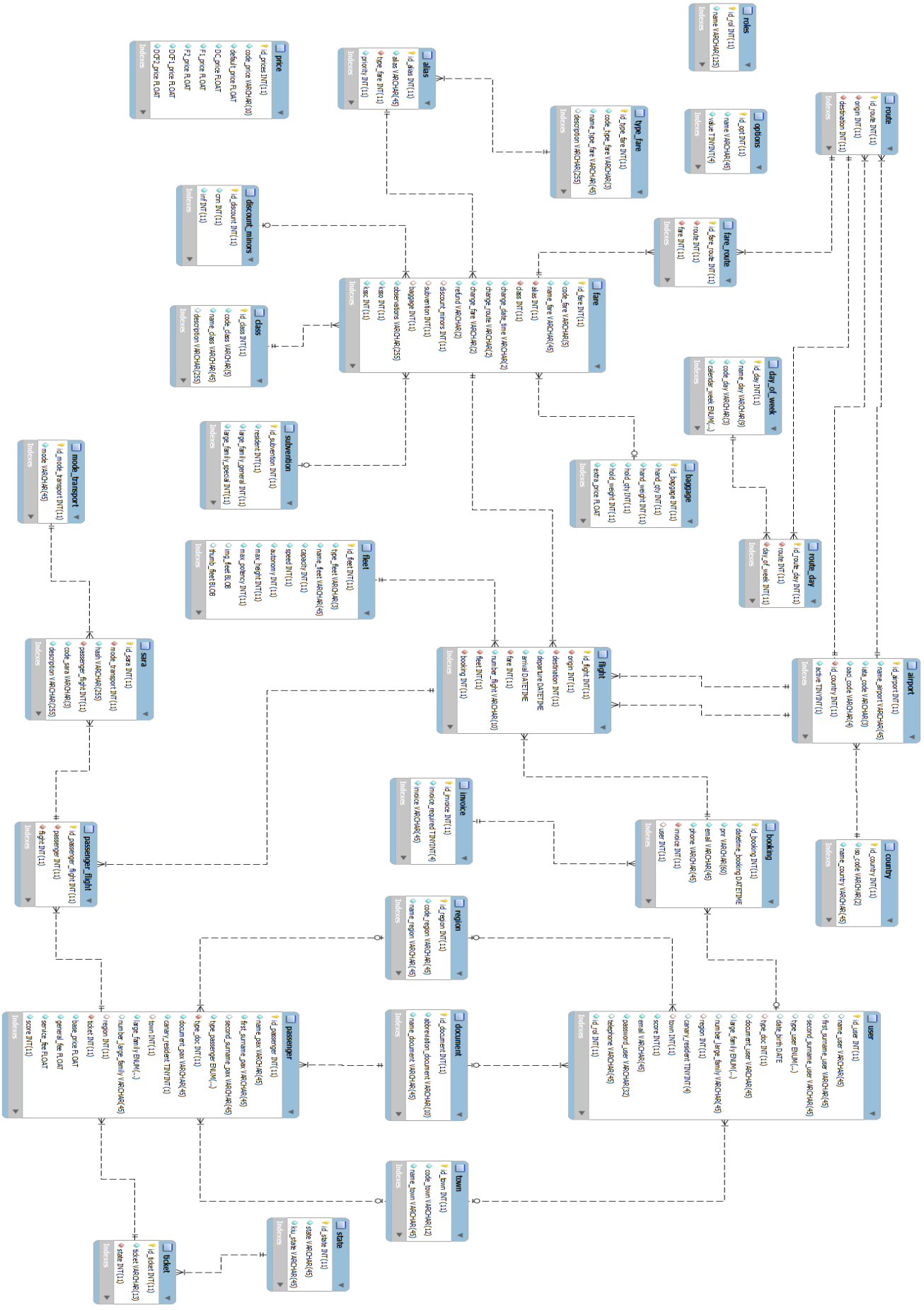


Imagen del diagrama de base de datos relacional MySQL.

9. Implementación

En este capítulo veremos más en detalle las funcionalidades ofrecidas por la aplicación y el procedimiento seguido para su implementación. A continuación se expondrá uno a uno todos los servicios desarrollados, ordenados de menos a más complejidad.

9.1. Facturación online

Para hablar de este apartado tenemos que hablar del sistema Kiu y de sus limitaciones si lo comparamos con otro con más experiencia en el sector como es Amadeus.

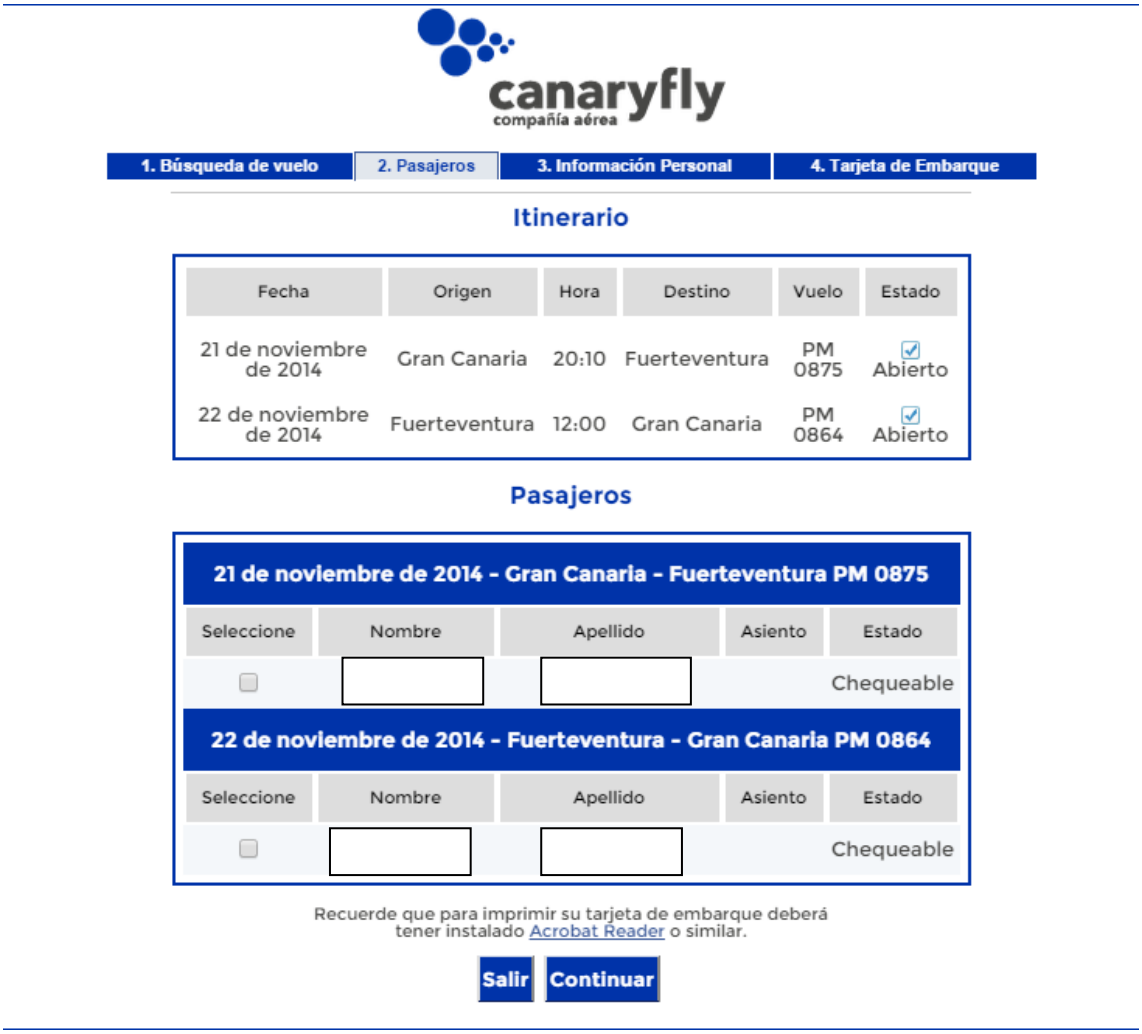
Debemos partir de la base de que Kiu no ofrece todas las funcionalidades que son deseables en un GDS. Toda la información y gestión de los datos relativos a la facturación online, de aquí en adelante webcheckin, no está disponible.

Como otras muchas funciones, Kiu se encuentra en fase de desarrollo para poder proveer esta información pero actualmente la solución que dan es una aplicación estándar propia, la cual funciona correctamente y los pasajeros está utilizando en la actualidad, pero Canaryfly no tiene margen para implantarlo en sus sistemas a gusto.

Lo que se ha hecho en la web es proporcionar el siguiente formulario:

Imagen del formulario para la realización del webckeckin por web

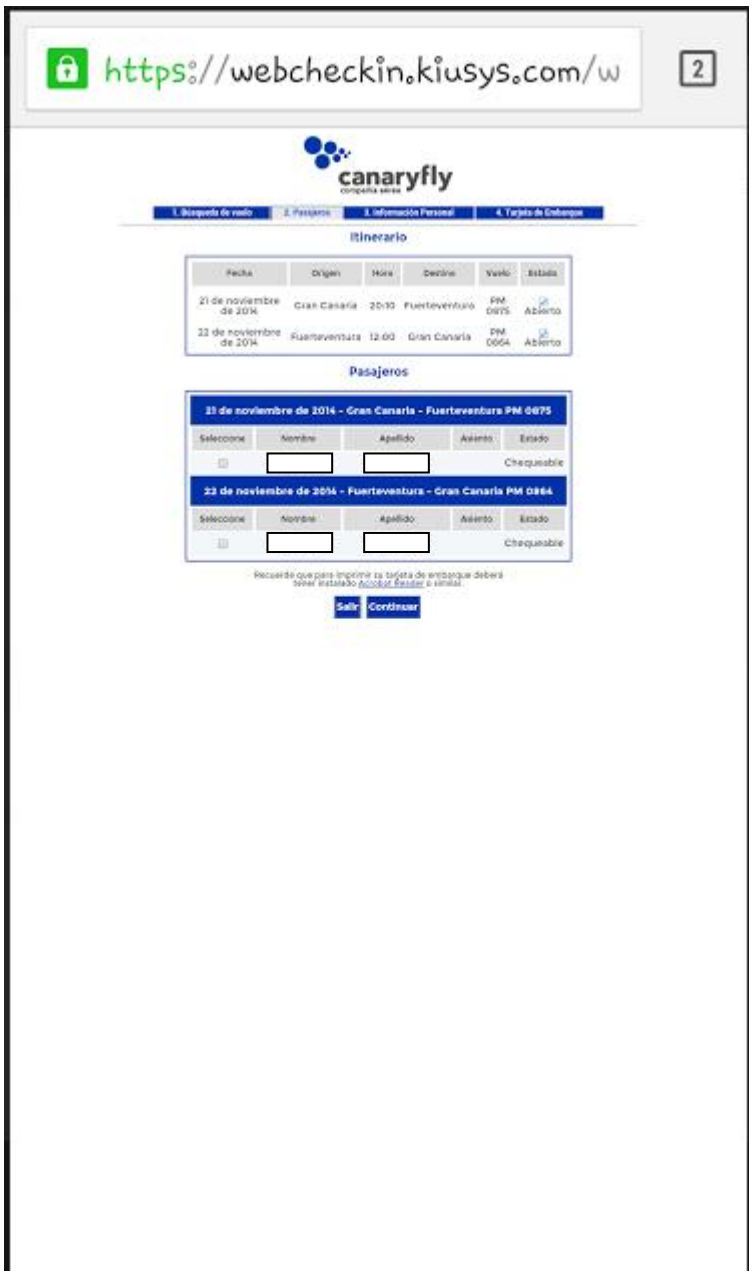
Una vez le damos a "Continuar" se envía una petición POST a la aplicación de Kiu, obteniendo el siguiente en un iframe.



Tras varias especificaciones conseguimos que Kiu cambiara el estilo CSS por defecto para que aparentar que Canaryfly tiene cierta repercusión en la aplicación pero este CSS no está adaptado a dispositivos móviles y no tenemos acceso a su modificación.

Todo esto acarrea, que hasta que no desarrollen su web service para trabajar con el webcheckin, desde una aplicación móvil lo único que se puede hacer es lo mismo que en la web, redirigir a su aplicación con los datos del pasajero.

Desde el formulario del móvil, comprobando que los datos con validos, se envía la petición y esto es lo que obtenemos:



Como se puede apreciar, la facturación online se puede realizar, pero deja mucho que desear su comodidad a la hora de trabajar con la aplicación de Kiu.

9.2. Chárter y vuelos en grupo

Tanto la solicitud de presupuesto de vuelos chárter como la de vuelos en grupo funcionan enviándole un correo con los datos recogidos en los formularios. Hasta que llegue el momento en el que la aplicación se ponga de cara al público y se explote, estas dos funcionalidades envían correos a mi cuenta de dirección electrónica de Canaryfly.

El proceso que se sigue es el de recoger los datos de los formularios, comprobarlos para testear que no se envía nada erróneo, como un correo inválido o algún campo vacío, pasarlos a un formato JSON, y hacer que éstos llegue a servidor Apache de la empresa. Una vez allí, por medio de PHP y la librería PHPMailer [44], enviamos los correos correspondientes. A continuación vemos un par de pruebas.

Imagen del formulario de vuelos en grupo realizando una prueba

Imagen del formulario de vuelos chárter realizando una prueba

En ambos casos aparece un mensaje informando al cliente de que la empresa se pondrá en contacto con él a la mayor brevedad posible.

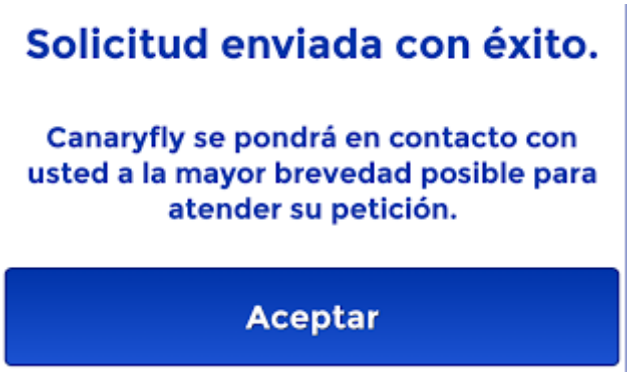


Imagen del mensaje de confirmación

A los pocos segundos, comprobamos el correo y vemos como los dos correos han llegado satisfactoriamente con los datos íntegros.

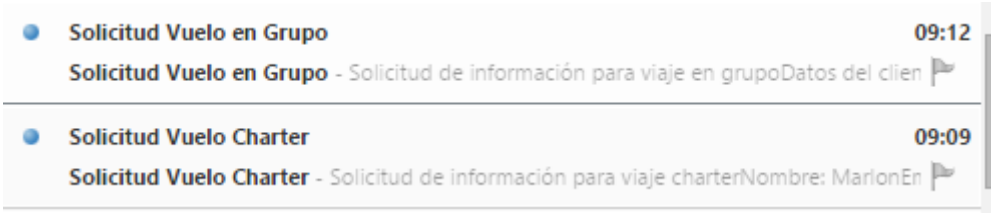


Imagen del email, confirmación del recibo del correo

Si abrimos el primero correo, el del vuelo en grupo, comprobamos los datos:

Solicitud de información para viaje en grupo

Datos del cliente

- **Nombre**: Marlon
- **Apellidos**: Fernández
- **Teléfono**: 666666666
- **Email**: otro_correo@inventado.com

Datos del vuelo

- **Ida**: Gran Canaria - La Palma 21/11/2014
- **Vuelta**: La Palma - Gran Canaria 24/11/2014

Datos del grupo

- **Nº de pasajeros**: 18
- **Tipo de grupo**: Vacacional
- **Observaciones**: Esto es una prueba para viajes en grupo.

Imagen del correo vuelos en grupo

Para terminar con la comprobación, abrimos el segundo correo y vemos que todo ha funcionado correctamente:

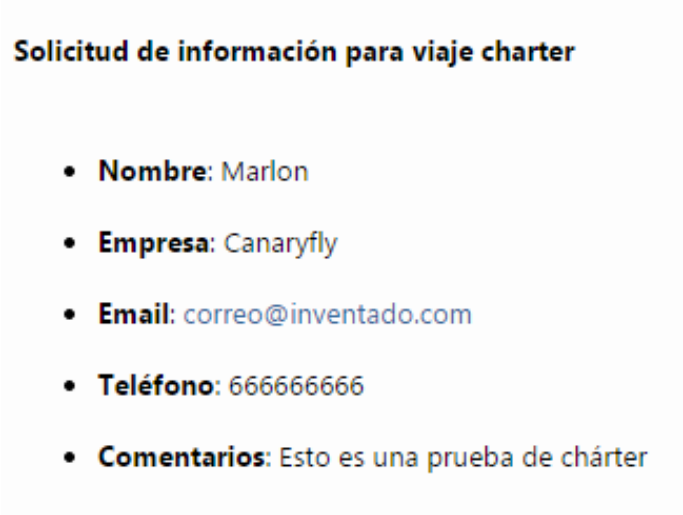


Imagen del correo chárter

9.3. Mis reservas

Esta funcionalidad trabaja directamente con la información que se almacena en base de datos al comprar un billete pero en el futuro se espera poder recuperar dicha información de web service.

En una reserva puede haber más de una persona, de ahí que el formulario pida el código de reserva y el número de documento, para que cada pasajero pueda acceder a su información.

Al mismo tiempo, en una reserva puede haber más de un vuelo, motivo por el cual, a parte del código de reserva y el número de documento, se pide el origen de dicha reserva.

El procedimiento es muy similar a los casos anteriores, se envía la información desde el cliente Android al servidor PHP en formato JSON, éste obtiene los datos y busca en la base de datos MySQL los resultados asociados, si existen los obtiene y los devuelve al cliente.

Los códigos de billetes suelen ser de 6 caracteres, pero existe la posibilidad de que su longitud sea mayor. Normalmente en las compañías aéreas, dos personas con dos descuentos diferentes aplicados no pueden realizar una reserva juntos. En Canaryfly lo permitimos pero sabiendo que se generarán códigos de billete concatenados, tantos como diferentes descuentos se hayan aplicado. Esto se hace así ya que la emisión en Kiu si que debe hacerse por separado, lo que conlleva a obtener más de un localizador, pero de cara a las gestiones internas y el cobro conjunto de los billetes, la empresa lo permite

y lo controla. Teniendo en cuenta las posibilidades existentes, No Residente, Residente, No Residente y Familia Numerosa General, Residente y Familia Numerosa General, No Residente y Familia Numerosa Especial, y Residente y Familia Numerosa Especial, la longitud máxima que podría llegar a alcanzar un código de billete es de 36 dígitos.

Bien es cierto que es algo que nunca ha pasado y que en una única ocasión ocurrió que hubieron 3 descuentos diferentes aplicados.

Lo anteriormente comentado sirve para explicar el motivo por el cual se hace una comprobación en el formulario sobre el código de billete y que éste sea múltiplo de 6 y no exactamente 6.

En el caso del webcheckin sí que se comprueba que sean exactamente 6, ya que desconocemos el proceso interno de su aplicación y no podemos garantizar que un localizador mayor de 6 dígitos funcione, aunque hemos comprobado que aplicando uno de los localizadores concatenados la aplicación sí que funciona, la única incidencia es que aparece sólo en la reserva en lugar de acompañado, pero no se conlleva ningún problema a la hora de realizar la facturación online.

En el caso del documento de identidad no podemos aplicar ningún filtro más que comprobar que el campo no esté vacío, ya que alguien puede hacer reservas con un pasaporte y no existe validador para este tipo de documento.

La base de datos utilizada en este TFG no está explotada, o lo que es lo mismo, no consta con datos reales, pero para realizar una prueba se ha introducido manualmente unos datos ficticios que comprobaremos a continuación:

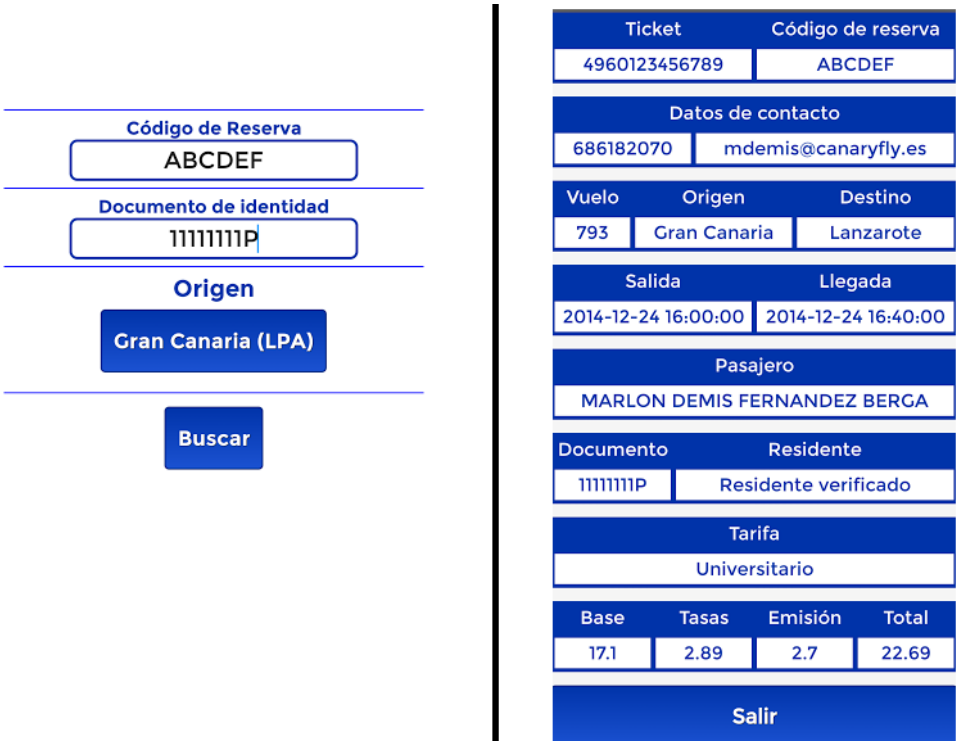


Imagen de la recuperación de una reserva de prueba

9.4. Reserva de vuelo

Por último, se explicará el proceso seguido para conseguir información real de la disponibilidad de vuelos, empezamos por el principio, el formulario que recoge los datos sobre los que un usuario quiere recuperar información.

- Ida y vuelta: La aplicación ofrece la posibilidad de elegir que un vuelo sea únicamente de ida o ida y vuelta, imposibilitando y desechando la opción de la fecha de vuelta cuando es un viaje de sólo ida.

Imagen del formulario de búsqueda de vuelos

- Selección de itinerario: Dependiendo del itinerario escogido, se podrá hacer una u otra cosa, por ello, "cambiar de fecha", "seleccionar pasajeros" y "buscar vuelos" se inhabilitan hasta que se detecte que tenemos un origen y un destino. El principal motivo de esto es determinar el vuelo como nacional o internacional, además de calcular los días en los que esa ruta vuela y darle una asistencia visual a los clientes deshabilitando los días en los que esa ruta no vuela.

Imagen de la lista de aeropuertos ofertados

Canaryfly cuenta con 9 destinos distintos y 26 rutas disponibles. Los destinos nacionales son Gran Canaria, Tenerife Norte, Lanzarote, La Palma y Fuerteventura. Los destinos internacionales son Dakhla (Sáhara Occidental), El Aiun (Marruecos), Guelmin (Marruecos) y Nouadhibou (Mauritania), todas ciudades del oeste de África.

Por motivos de falta de ocupación de vuelos, actualmente Canaryfly ha dejado de volar a Tenerife los sábados desde Gran Canaria, lo cual imposibilita todas aquellas rutas que tenía como escala a Tenerife. Esta es la única excepción, el resto de días todos los vuelos insulares están disponibles. Son los vuelos a África donde más podemos apreciar que no se viaja todo los días, veamos por ejemplo, Gran Canaria - Nouadhibou:

Imagen del calendario de la ruta Gran Canaria - Nouadhibou

Como podemos apreciar a simple vista, esta ruta sólo está disponible los miércoles y domingos. Si no se deshabilitaran el resto de días los clientes perderían mucho tiempo buscando disponibilidad de vuelos.

Comentar que los calendarios ofrecidos por Android daban problemas de compatibilidad entre versiones más antiguas y más modernas, por ello se hizo uso de la librería caldroid [46], la cual proporciona un calendario mucho más aceptado por todas las versiones de Android y fácilmente personalizable.

Aeropuertos, rutas, y días de vuelo son información obtenida desde la base de datos del servidor por el mismo proceso anteriormente explicado.

- Selección de pasajeros: Debido a que la mayoría de aerolíneas no dan la posibilidad de reservar con distintos descuentos aplicados, es muy común el uso de un "picker" numérico para seleccionar un número de pasajeros, ya sean adultos, niños o bebés.

  En el caso de Canaryfly, hemos decidido hacerlo en una ventana aparte ya que sí permitimos tal caso y es importante aplicar los diferentes tipos de descuentos por pasajero que se vaya añadiendo. Los tipos de pasajeros existen derivan de las combinaciones que hay entre, tipo de pasajero (Adulto, Niño, Bebé), Residencia (Sí, No) y Descuento de Familia Numerosa (General, Especial, Sin Descuento).

  Es aquí donde juega un papel importante haber determinado anteriormente si el vuelo era nacional o no. En el caso de ser nacional son aplicables todos los descuentos, de lo contrario, lo único que puede variar es el tipo de pasajero, dejando No Residente y Sin Descuento de Familia Numerosa como únicas opciones.

  En esta sección también controlamos los casos asociados a los menores de edad, únicamente las agencias de viaje pueden vender billetes a niños menores de edad que viajen solos, desde la web, y por supuesto desde la aplicación, no se debe permitir. Además, en el caso de los niños siempre tienen que ir asociados a un adulto con el mismo descuento asociado, lo que implica que un niño residente no puede viajar con un adulto no residente. Lo mismo ocurre con los bebés pero además, es política de empresa que no puedan añadirse más de dos bebés por reserva, lo cual también se controla y no se permite.

- Tarifas sociales: Son unas tarifas especiales destinadas a una clientela específica.

  - Tarifa Senior: Tarifa aplicable a mayores de 61 años y 364 días. Se solicita y comprobará a la hora de facturar el documento acreditativo.

  - Tarifa Universitario: Tarifa aplicable a estudiantes universitarios acreditado con carnet de una universidad vigente y menores de 30 años. Se solicitará y comprobará a la hora de facturar el documento acreditativo.

  - Tarifa Joven: Tarifa aplicable a menores de 25 años y mayores de 12 años. Se solicitará y comprobará a la hora de facturar el documento acreditativo.

Nuevamente, estas tarifas y la información asociada a ellas se recupera de la base de datos del servidor Apache.

Estas tarifas sólo deben ofrecerse si todos los pasajeros seleccionados cumplen las condiciones de cada una. De esta norma sacamos las siguientes restricciones:

- Si existe un bebé, ninguna de las tarifas ha de ser ofertada.
- Si existe un niño, la tarifa Senior no ha de ser ofertada.
- Si el vuelo es internacional, ninguna de las tarifas ha de ser ofertada.

* Únicamente se ofertan todas las tarifas sociales cuando viajan sólo adultos.

Esta oferta se hace efectiva justo cuando le damos a "Buscar Vuelos":

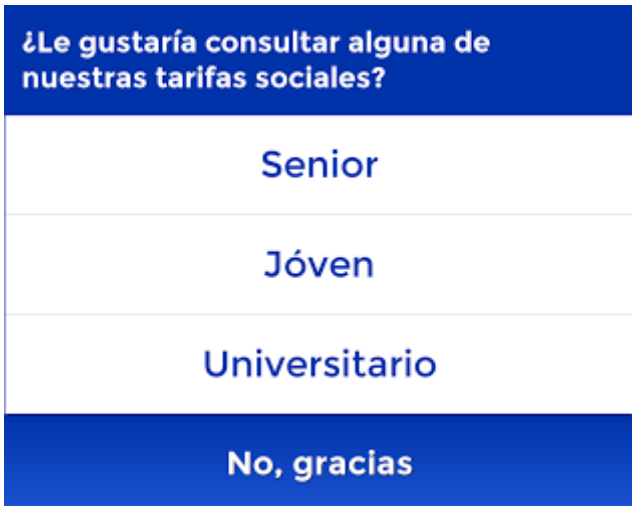


Imagen del dialogo que ofrece las tarifas sociales

- Muestra de precios: Hace cerca de un año, la página web no mostraba los precios de todas las tarifas, si no que había que seleccionar una en concreto para ver su precio asociado. Esto es debido a qué Kiu no devuelve los precios junto a la información de las tarifas. Tiene una función concreta para recuperar precios, de ahí que sólo se mostraran cuando se seleccionaba una tarifa. Cuando se nos encargó mostrar todos los precios nada más cargar la disponibilidad, surgió un gran problema, ya que Canaryfly cuenta con muchas más tarifas que la media, y tener que hacer una petición de precio para todas las tarifas en tiempo real, hacía que la solicitud de disponibilidad fuera muy lenta.

  Tras estudiar diferentes alternativas, optamos por desarrollar un script en PHP, que mediante el crontab del servidor CentOS, todas las mas madrugadas hiciera la petición de precios para todas las tarifas para todas las rutas disponibles en el día, cargándolas en una base de datos independiente. De esta manera resultaba muy sencillo y eficaz obtener todos los precios de las tarifas ofertadas.

- Búsqueda de vuelos: Para finalizar, explicaremos el proceso de recuperación de disponibilidad de vuelos.

  Hasta este punto hemos obtenido todo lo necesario, origen, destino, pasajeros con sus descuentos aplicados, fecha de vuelo y la posibilidad de haber seleccionado una tarifa especial. Como siempre, estos datos son enviados en formato JSON al servidor Apache, pero es en este punto donde cambia el proceso. Esta vez no sólo interactuaremos con la base de datos, si no que hacemos uso de las funciones desarrolladas por el segundo sistema de la capa de datos, el web service de Kiu.

  A partir de aquí la lógica de negocio tiene mayor complejidad que los casos anteriores, ya que hay que hacer trabajar conjuntamente a la base de datos con el web service y conseguir obtener de ambos lados toda la información correcta y necesaria.

  Se ha creado una clase en PHP encargada de obtener los datos del servidor:

```php
$kiu = new kiu();

$origin_booking = filter_input(INPUT_POST, 'origin');
$destination_booking = filter_input(INPUT_POST, 'destination');
$datetime_booking = filter_input(INPUT_POST, 'datetime'); // yyyy-MM-dd //
$adults = filter_input(INPUT_POST, 'ADT');
$children = filter_input(INPUT_POST, 'CNN');
$babies = filter_input(INPUT_POST, 'INF');
$social = filter_input(INPUT_POST, 'social');

$data = array(
    "type" => "AirAvail",
    "origin" => $origin_booking,
    "destination" => $destination_booking,
    "datetime" => $datetime_booking,
    "passengers" => array(
        "ADT" => $adults,
        "CNN" => $children,
        "INF" => $babies),
    "directonly" => false);

$result = array();

$air_avail = [$kiu->airAvail(
            $data["origin"], $data["destination"], $data["datetime"],
        $data["passengers"], $data["directonly"])];
```

  La función airAvail recibe un array como parámetro y se encarga de transformarlo al formato XML correspondiente, recuperando más tarde el XML de respuesta y parseandolo de tal manera que devuelve toda la información

proveniente del web service en un array bastante manejable, del que podremos extraer nuestra información de disponibilidad. Cuando tenemos nuestro array completo, lo convertimos en formato JSON y lo devolvemos a nuestro cliente Android, donde se encargará de crear las instancias de las clases modelo correspondiente con la información obtenida.

Es importante destacar que el origen y el destino utilizan sus códigos IATA (LPA, TFN, FUE, ...).

En este punto pueden ocurrir varias cosas, la primera de ellas es que en el día seleccionado ya no queden más vuelos, lo cual mostraría el siguiente mensaje.

Imagen del mensaje que indica que no se ha encontrado vuelo

Si finalmente encuentra disponibilidad, puede ser que nos devuelva información de un único vuelo o de varios. Veamos un ejemplo de cada caso, comparándolo con los resultados que se obtiene desde la web de Canaryfly:

Gran Canaria - Tenerife Norte (24 de Diciembre de 2014) - Un único vuelo:

Imagen de la disponibilidad por web para un sólo vuelo

Imagen de la disponibilidad por la aplicación de Canaryfly para un sólo vuelo

Gran Canaria - Tenerife Norte es una ruta poco demandada en Canaryfly, pero si seleccionamos otra más popular, como por ejemplo Gran Canaria - Fuerteventira, vemos que el resultado cambia:

Gran Canaria - Fuerteventura (26 de Noviembre de 2014) - Varios vuelos:

Gran Canaria (**LPA**) - Fuerteventura (**FUE**)

Dom 23 Nov | Lun 24 Nov | Mar 25 Nov | **Mie 26 Nov** | Jue 27 Nov | Vie 28 Nov | Sab 29 Nov

SALIDA	LLEGADA	VUELO	TIPO	Amigo	Cómoda	Flex	Fly	Fly +
06:55	07:35	PM861	SW4	No Disponible	23,00€ 3 plazas	27,00€ 2 plazas	33,00€ 4 plazas	No Disponible
09:15	09:55	PM863	SW4	No Disponible	23,00€ 3 plazas	27,00€ 2 plazas	33,00€ 4 plazas	37,00€
17:50	18:30	PM873	SW4	No Disponible	No Disponible	27,00€ 2 plazas	33,00€ 4 plazas	37,00€
20:10	20:50	PM875	SW4	No Disponible	23,00€ 3 plazas	27,00€ 2 plazas	33,00€ 4 plazas	37,00€

Imagen de la disponibilidad por web para varios vuelos

Imagen de la disponibilidad por la aplicación de Canaryfly para varios vuelos

En este caso, los usuarios pueden ir desplazando por los diferentes vuelos, tanto moviéndose de una pantalla a otra con los dedos o pulsando en los tabs superiores

Por último, puede ser que seleccionemos una ruta con varias escalas, ya que existen rutas con origen y destino que no poseen un vuelo directo. Algunos ejemplos son Gran Canaria - La Palma (Escala en Tenerife Norte), Fuerteventura - Tenerife Norte (Escala en Gran Canaria) o la opción con más escalas, Lanzarote - La Palma (Escala en Gran Canaria y Tenerife Norte). Veamos un ejemplo de este último caso:

Imagen de la disponibilidad por web para un vuelos con escala

Vuelo 1 | Vuelo 2 | Vuelo 3

IDA: ACE - SPC
23/11/2014

Información del vuelo

Lanzarote (ACE) - Gran Canaria (LPA)			
Salida	Llegada	Vuelo	Tipo
09:10	09:50	PM782	AT4
Gran Canaria (LPA) - Tenerife Norte (TFN)			
Salida	Llegada	Vuelo	Tipo
14:35	15:05	PM621	AT7
Tenerife Norte (TFN) - La Palma (SPC)			
Salida	Llegada	Vuelo	Tipo
15:35	16:10	PM425	AT7

Vuelo 1 | Vuelo 2 | Vuelo 3

IDA: ACE - SPC
23/11/2014

Información del vuelo

Lanzarote (ACE) - Gran Canaria (LPA)			
Salida	Llegada	Vuelo	Tipo
09:10	09:50	PM782	AT4
Gran Canaria (LPA) - Tenerife Norte (TFN)			
Salida	Llegada	Vuelo	Tipo
14:35	15:05	PM621	AT7
Tenerife Norte (TFN) - La Palma (SPC)			
Salida	Llegada	Vuelo	Tipo
17:35	18:10	PM427	AT7

Vuelo 1 | Vuelo 2 | Vuelo 3

IDA: ACE - SPC
23/11/2014

Información del vuelo

Lanzarote (ACE) - Gran Canaria (LPA)			
Salida	Llegada	Vuelo	Tipo
09:10	09:50	PM782	AT4
Gran Canaria (LPA) - Tenerife Norte (TFN)			
Salida	Llegada	Vuelo	Tipo
14:35	15:05	PM621	AT7
Tenerife Norte (TFN) - La Palma (SPC)			
Salida	Llegada	Vuelo	Tipo
19:45	20:15	PM429	AT7

Imagen de la disponibilidad por la aplicación de Canaryfly para vuelos con escala

En este último caso, hay que tener una cosa en cuenta. Supongamos que quieren viajar 3 pasajeros de Fuerteventura a Tenerife con escala en Gran Canaria. Para una tarifa concreta, de Gran Canaria a Tenerife hay 4 plazas pero para la misma tarifa en la ruta de Fuerteventura a Gran Canaria sólo hay 2. Esto generaría conflictos ya que podrían darse casos de overbooking. Para evitar este problema, cuando una ruta tiene escala, las tarifas con menor número de plazas son las que se ofertan, evitando que puedan darse casos como el planteado anteriormente.

Por último y para terminar, comentar algunas consideraciones generales que afectan a todas las disponibilidades de vuelo. Relacionado con el tema del número de asientos, si en una tarifa cualquiera, tenga escala o no, quedan un número de plazas determinado y el número de pasajeros, sumando adultos y niños ya que los bebés no pagan, supera el número de plazas disponible, está debe aparecer como no disponible.

Siempre que el número de plazas sea inferior a 4, se ha de notificar, y en los casos de 2 plazas y última plaza, el mensaje debe aparecer en color rojo.

Como último aclaratorio, el precio que aparece en las tarifas no es el precio por pasajero, es el precio total de la reserva, con los descuentos aplicados a todos sus pasajeros.

10. Futuro del proyecto

Hay mucho que mejorar y desarrollar para que la aplicación pueda llegar a ser presentada al público, pero con este proyecto hemos creado la base y sabemos qué puntos hay que mejorar y qué nos falta por implementar.

Desde un punto de vista gráfico, como se explicó, un objetivo principal era adquirir los conocimientos necesarios para personalizar una aplicación Android. Para este TFG se ha hecho un diseño preliminar que pretende dar la mayor comodidad posible a la hora de interactuar con la aplicación y que mínimamente se identifique con Canaryfly. En el futuro se implantará un diseño mucho más trabajado, además de tener en cuenta que funcione perfectamente en todas las pantallas Android del mercado, incluyendo los tablets, por lo que también habrá que desarrollar que la aplicación pueda ser utilizada en posición horizontal.

Por supuesto, se trabajará para optimizar, si cabe, las funcionalidades ya creadas, además de implementar todas aquellas que nos faltan, dígase recolecta de datos personales de los pasajeros, validación de residencia con el sistema SARA, proceso de pago de reserva y emisión de billete.

A parte de lo anterior mencionado, se trabajará para otorgar un mejor sistema de facturación online. En cuanto Kiu desarrolle las funciones necesarias en su web service, se creará un proceso propio y mejor orientado a dispositivos móviles, no teniendo que utilizar la aplicación web que Kiu ofrece por defecto. Incluso, Kiu está trabajando con otras organizaciones destinadas a gestionar reservas, como es el caso de PassWallet [47], para poder obtener las tarjetas de embarque de manera más cómoda, por lo que se decidirá si se da la opción de elegir entre ambas opciones o únicamente se utiliza una de ellas.

En lo referente a los usuarios registrados, se creará un sistema sincronizado con la página web para poder gestionar los datos personales y facilitar la reserva de vuelos, además del sistema por puntos para posibles descuentos o promociones.

Dado que se llegará a recoger datos personales de los usuarios, se incluirán todos los avisos legales e información necesarias para cumplir con lo establecido en las Leyes LOPD y LSSI-CE [48].

Por suerte Android cuenta con un buen sistema para la traducción de contenido de una aplicación, ya que el texto puede ser almacenado en un fichero y mediante ids se enlaza con algún elemento visual. Copiando dicho fichero y cambiando el texto, pero no los ids, conseguimos traducir de una manera eficiente y sencilla. Inglés y francés son los idiomas que actualmente se quieren implantar tanto en la web como en la aplicación.

Por último, se creará una sección de información (FAQ, contacto, servicios, tarifas, etc).

11. Conclusiones

Una vez finalizado este TFG puedo hacer un repaso de todo lo aprendido y valorar la capacidad con la que he reaccionado ante los problemas que han ido surgiendo.

A parte de todos los conocimientos en programación para dispositivos móviles Android adquiridos, he de destacar la experiencia que he ganado a la hora de diseñar e implementar un proyecto más grande del que estábamos acostumbrados en la carrera. He podido comprobar, como tantos profesores nos han enseñado y como tantas veces nos han repetido, lo importantísimo que es sentarse a analizar, diseñar, reanalizar y rediseñar el producto tantas veces como sea necesario antes de teclear la primera línea de código. Por suerte puedo decir que para este TFG hubo un buen estudio previo de lo que se deseaba obtener y en todas las ocasiones que me topé con algún cambio de requisitos, que fueron varias veces, pude afrontar el cambio, con no mucha dificultad.

También he comprobado lo difícil que es tener que retomar un trabajo anterior, ya que a lo largo de estos meses, por motivos de prioridad, se me asignaba realizar otras labores ajenas al proyecto, lo cual podía tenerme varias semanas alejado de la implementación del TFG. Digamos que tuve que retomar el proyecto unas 3 o 4 veces, lo cual se hacía bastante complicado en algunas ocasiones, sobre todo en etapas más avanzadas.

He podido experimentar de primera mano los problemas que acarrea tener separado el software del hardware. Que muchas empresas distintas fabriquen dispositivos móviles para Android es un verdadero problema y es muy complicado, por no decir imposible, llegar a satisfacer al 100% del mercado. Estos problemas de los que hablo no son únicamente visuales, dado que existen muchas pantallas distintas, sino que además las versiones soportadas por los teléfonos varían y no paran de actualizarse. En el futuro Android seguramente se estandarizará y estos problemas se resolverán, pero actualmente es una lástima que usuarios que poseen este sistema puedan verse ante la situación de no poder usar una buena aplicación en su dispositivo móvil.

Por suerte, hay algunos tips que ayudan a desarrollar para la mayoría de los móviles y los hemos adquirido en este proyecto. Para las pruebas hemos utilizado una de las pantallas más grandes del mercado (Samsung Note 3) con una de las versiones más actuales (Kitkat) y en comparación uno de los móviles con pantalla más pequeña (Samsung Ace 2) con una de las versiones más antiguas (Gingerbread) y podemos decir que lo implementado hasta el momento funciona en ambos dispositivos.

Pese a los problemas encontrados en el camino de este TFG, me siento bastante satisfecho con el resultado final y deseo profundamente especializarme y aumentar mis conocimientos en el mundo de la programación para dispositivos móviles, dado que son las nuevas tendencias y he podido comprender lo importante que es exportar las tecnologías existentes a un entorno móvil.

12. Anexo 1 - Formato de mensajes XML (Web Service)

12.1. Elementos del mensaje KIU_AirAvailRQ:

- DirectFlightsOnly: [true|false]. Indica si debe mostrar solo vuelos directos.

- POS: Elemento que agrupa la información del Punto de Venta desde el cual se realiza la petición.

    - Source: Contenedor de los datos del Punto de Venta o POS.
        - AgentSine: Alfanumérico [9]. Identificador del agente que realiza la petición.
        - TerminalID: Alfanumérico [10]. Código identificador del dispositivo desde el que se realiza la petición.

- SpecificFlightInfo: Elemento optativo para indicar que los resultados devueltos deben corresponder a un único carrier.

    - Airline: Elemento que contiene a la aerolínea por la cual filtrar los resultados.
    - Code: Código IATA de la aerolínea/carrier.
- OriginDestinationInformation: Origen y destino para el que se pide la disponibilidad.
    - DepartureDateTime: Fecha ISO. Fecha de partida.
    - OriginLocation: Lugar de origen.
        - LocationCode: Alfabético [3]. Código IATA de aeropuerto.
    - DestinationLocation: Lugar de destino.
        - LocationCode: Alfabético [3]. Código IATA de aeropuerto.
- TravelPreferences: Indica las preferencias del usuario sobre las opciones a mostrar.
    - CabinPref: Indica las preferencias del usuario sobre la cabina.
        - Cabin: Cabina seleccionada para el vuelo.
            - Business
            - Cockpit
            - Economy
            - First
            - PremiumBusiness
            - PremiumEconomy
            - Suite
- TravelerInfoSummary: Información de pasajeros
    - AirTravelerAvail: Tipos y cantidad de pasajeros.
        - PassengerTypeQuantity: Cantidad de pasajeros para un tipo determinado.
            - Quantity: Entero [0...9]. Cantidad de pasajeros.
            - Code: [ADT|CNN|INF]. Tipo de pasajero.

12.2. Elementos del mensaje KIU_AirAvailRS:

- OriginDestinationInformation: Origen y destino para el que se devuelve la disponibilidad.

    - DepartureDateTime: Fecha ISO. Fecha de partida.
    - OriginLocation: Lugar de origen.
        - LocationCode: Alfabético [3]. Código IATA de aeropuerto.
    - DestinationLocation: Lugar de destino.
        - LocationCode Alfabético [3]. Código IATA de aeropuerto.

    - OriginDestinationOptions: Contenedor para las opciones de cada segmento de vuelo.
        - OriginDestinationOption: Opciones para el segmento de vuelo específico.
            - FlightSegment: Información acerca del segmento de vuelo. En los casos de vuelos no directos, este elemento se repite para cada tramo.
                - DepartureDateTime: Fecha ISO de partida del vuelo.
                - ArrivalDateTime: Fecha ISO de llegada del vuelo.
                - StopQuantity: Cantidad de paradas del vuelo.
                - FlightNumber: Alfanumérico [4]. Número identificatorio del vuelo.
                - JourneyDuration: Duración del vuelo.
                - DepartureAirport: Aeropuerto de origen.
                    - LocationCode: Alfabético [3]. Código IATA de aeropuerto.
                - ArrivalAirport: Aeropuerto de destino.
                    - LocationCode: Alfabético [3]. Código IATA de aeropuerto.
                - Equipment: Tipo de equipamiento aeronáutico.
                    - AirEquipType: Alfanumérico [3]. Código IATA del modelo de la aeronave.
                - MarketingAirline: Aerolínea de venta.
                    - CompanyShortName: Alfabético [1...32]. Nombre de la aerolínea.
                - Meal: Servicio de comida [*].

* - MealCode: Alfabético con caracteres especiales [1...3]. [BSDHLRCMPFO-]. Un código de hasta tres caracteres o un guión medio que simboliza los servicios de comida disponibles para la clase de reserva: Desayuno(B,Breakfast), Refrigerio(S,Snack), Cena(D,Dinner), Comida Caliente (H,HotMeal), Almuerzo(L,Lunch), Refrescos (R, Refreshments), Alcohol de Cortesía (C, Complimentary liquor), Comida (M, Meal), Alcohol disponible para la venta (P, Liquor for Purchase), Comida disponible para la venta (F, Food for Purchase), Comida Fría (O, Cold Meal), Sin servicio de comida (-).

                - BookingClassAvail: Códigos de reserva disponibles para el segmento de vuelo.
                    - ResBookDesigCode: Alfabético [1]. Código Designador de Reserva.
                    - ResBookDesigQuantity: Entero [0...9]. Cantidad de asientos disponibles para el Código Designador de Reserva.

13. Anexo 2 - Sentencias SQL

CREATE TABLE `airport` (

`id_airport` int(11) NOT NULL AUTO_INCREMENT,

`name_airport` varchar(45) NOT NULL,

`iata_code` varchar(3) NOT NULL,

`oaci_code` varchar(4) NOT NULL,

`id_country` int(11) NOT NULL,

`active` tinyint(1) NOT NULL,

PRIMARY KEY (`id_airport`),

UNIQUE KEY `name_airport_UNIQUE` (`name_airport`),

UNIQUE KEY `iata_code_UNIQUE` (`iata_code`),

UNIQUE KEY `oaci_code_UNIQUE` (`oaci_code`),

KEY `fk_airport_1_idx` (`id_country`),

CONSTRAINT `fk_country_airport` FOREIGN KEY (`id_country`) REFERENCES `country` (`id_country`) ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;

CREATE TABLE `baggage` (

`id_baggage` int(11) NOT NULL AUTO_INCREMENT,

`hand_qty` int(11) NOT NULL,

`hand_weight` int(11) NOT NULL,

`hold_qty` int(11) NOT NULL,

`hold_weight` int(11) NOT NULL,

`extra_price` float NOT NULL,

PRIMARY KEY (`id_baggage`)

) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

```
CREATE TABLE `alias` (
  `id_alias` int(11) NOT NULL AUTO_INCREMENT,
  `alias` varchar(45) NOT NULL,
  `type_fare` int(11) NOT NULL,
  `priority` int(11) NOT NULL,
  PRIMARY KEY (`id_alias`),
  UNIQUE KEY `alias_UNIQUE` (`alias`),
  KEY `fk_type_fare_alias_idx` (`type_fare`),
  CONSTRAINT `fk_type_fare_alias` FOREIGN KEY (`type_fare`) REFERENCES
`type_fare` (`id_type_fare`) ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;

CREATE TABLE `booking` (
  `id_booking` int(11) NOT NULL AUTO_INCREMENT,
  `datetime_booking` datetime NOT NULL,
  `pnr` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `phone` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `invoice` int(11) NOT NULL,
  `user` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_booking`),
  KEY `fk_user_booking_idx` (`user`),
  KEY `fk_invoice_booking_idx` (`invoice`),
  CONSTRAINT `fk_invoice_booking` FOREIGN KEY (`invoice`) REFERENCES `invoice`
(`id_invoice`) ON UPDATE CASCADE,
  CONSTRAINT `fk_user_booking` FOREIGN KEY (`user`) REFERENCES `user` (`id_user`)
ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
```

```
CREATE TABLE `class` (
  `id_class` int(11) NOT NULL AUTO_INCREMENT,
  `code_class` varchar(5) NOT NULL,
  `name_class` varchar(45) NOT NULL,
  `description` varchar(255) DEFAULT '',
  PRIMARY KEY (`id_class`),
  UNIQUE KEY `code_class_UNIQUE` (`code_class`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

CREATE TABLE `country` (
  `id_country` int(11) NOT NULL AUTO_INCREMENT,
  `iso_code` varchar(2) NOT NULL,
  `name_country` varchar(45) NOT NULL,
  PRIMARY KEY (`id_country`),
  UNIQUE KEY `iso_code_UNIQUE` (`iso_code`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;

CREATE TABLE `day_of_week` (
  `id_day` int(11) NOT NULL AUTO_INCREMENT,
  `name_day` varchar(9) NOT NULL,
  `code_day` varchar(3) NOT NULL,
  `calendar_week`
enum('MONDAY','TUESDAY','WEDNESDAY','THURSDAY','FRIDAY','SATURDAY','SUN
DAY') NOT NULL,
  PRIMARY KEY (`id_day`),
  UNIQUE KEY `code_day_UNIQUE` (`code_day`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `discount_minors` (
  `id_discount` int(11) NOT NULL AUTO_INCREMENT,
  `cnn` int(11) NOT NULL,
  `inf` int(11) NOT NULL,
  PRIMARY KEY (`id_discount`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

CREATE TABLE `document` (
  `id_document` int(11) NOT NULL AUTO_INCREMENT,
  `abbreviation_document` varchar(10) NOT NULL,
  `name_document` varchar(45) NOT NULL,
  PRIMARY KEY (`id_document`),
  UNIQUE KEY `abbreviation_document_UNIQUE` (`abbreviation_document`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;

CREATE TABLE `fare_route` (
  `id_fare_route` int(11) NOT NULL AUTO_INCREMENT,
  `route` int(11) NOT NULL,
  `fare` int(11) NOT NULL,
  PRIMARY KEY (`id_fare_route`),
  KEY `fk_f_fare_route_idx` (`fare`),
  KEY `fk_r_fare_route_idx` (`route`),
  CONSTRAINT `fk_f_fare_route` FOREIGN KEY (`fare`) REFERENCES `fare` (`id_fare`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_r_fare_route` FOREIGN KEY (`route`) REFERENCES `route`
(`id_route`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=245 DEFAULT CHARSET=utf8;
```

CREATE TABLE `fare` (

`id_fare` int(11) NOT NULL AUTO_INCREMENT,

`code_fare` varchar(5) NOT NULL,

`name_fare` varchar(45) NOT NULL,

`alias` int(11) NOT NULL,

`class` int(11) NOT NULL,

`change_date_time` varchar(2) NOT NULL,

`change_route` varchar(2) NOT NULL,

`change_fare` varchar(2) NOT NULL,

`refund` varchar(2) NOT NULL,

`discount_minors` int(11) DEFAULT NULL,

`subvention` int(11) DEFAULT NULL,

`baggage` int(11) DEFAULT NULL,

`observations` varchar(255) NOT NULL,

`ksso` int(11) NOT NULL,

`kssc` int(11) NOT NULL,

PRIMARY KEY (`id_fare`),

KEY `fk_class_fare_idx` (`class`), KEY `fk_discount_fare_idx` (`discount_minors`),

KEY `fk_subvention_fare_idx` (`subvention`), KEY `fk_baggage_fare_idx` (`baggage`),

KEY `fk_alias_fare_idx` (`alias`),

CONSTRAINT `fk_baggage_fare` FOREIGN KEY (`baggage`) REFERENCES `baggage` (`id_baggage`) ON UPDATE CASCADE,

CONSTRAINT `fk_discount_fare` FOREIGN KEY (`discount_minors`) REFERENCES `discount_minors` (`id_discount`) ON UPDATE CASCADE,

CONSTRAINT `fk_subvention_fare` FOREIGN KEY (`subvention`) REFERENCES `subvention` (`id_subvention`) ON UPDATE CASCADE,

CONSTRAINT `fk_alias_fare` FOREIGN KEY (`alias`) REFERENCES `alias` (`id_alias`) ON UPDATE CASCADE,

CONSTRAINT `fk_class_fare` FOREIGN KEY (`class`) REFERENCES `class` (`id_class`) ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

```
CREATE TABLE `fleet` (
  `id_fleet` int(11) NOT NULL AUTO_INCREMENT,
  `type_fleet` varchar(3) NOT NULL,
  `name_fleet` varchar(45) NOT NULL,
  `capacity` int(11) NOT NULL,
  `speed` int(11) NOT NULL,
  `autonomy` int(11) NOT NULL,
  `max_height` int(11) NOT NULL,
  `max_potency` int(11) NOT NULL,
  `img_fleet` blob,
  `thumb_fleet` blob,
  PRIMARY KEY (`id_fleet`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

CREATE TABLE `invoice` (
  `id_invoice` int(11) NOT NULL AUTO_INCREMENT,
  `invoice_required` tinyint(4) NOT NULL DEFAULT '0',
  `invoice` varchar(45) NOT NULL,
  PRIMARY KEY (`id_invoice`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

CREATE TABLE `mode_transport` (
  `id_mode_transport` int(11) NOT NULL AUTO_INCREMENT,
  `mode` varchar(45) NOT NULL,
  PRIMARY KEY (`id_mode_transport`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

CREATE TABLE `flight` (

`id_flight` int(11) NOT NULL AUTO_INCREMENT,

`origin` int(11) NOT NULL,

`destination` int(11) NOT NULL,

`departure` datetime NOT NULL,

`arrival` datetime NOT NULL,

`fare` int(11) NOT NULL,

`number_flight` varchar(10) NOT NULL,

`fleet` int(11) NOT NULL,

`booking` int(11) NOT NULL,

PRIMARY KEY (`id_flight`), KEY `fk_departure_flight_idx` (`origin`),

KEY `fk_destination_flight_idx` (`destination`), KEY `fk_fleet_flight_idx` (`fleet`),

KEY `fk_booking_flight_idx` (`booking`), KEY `fk_fare_flight_idx` (`fare`),

CONSTRAINT `fk_booking_flight` FOREIGN KEY (`booking`) REFERENCES `booking` (`id_booking`) ON UPDATE CASCADE,

CONSTRAINT `fk_destination_flight` FOREIGN KEY (`destination`) REFERENCES `airport` (`id_airport`) ON UPDATE CASCADE,

CONSTRAINT `fk_fare_flight` FOREIGN KEY (`fare`) REFERENCES `fare` (`id_fare`) ON UPDATE CASCADE,

CONSTRAINT `fk_fleet_flight` FOREIGN KEY (`fleet`) REFERENCES `fleet` (`id_fleet`) ON UPDATE CASCADE,

CONSTRAINT `fk_origin_flight` FOREIGN KEY (`origin`) REFERENCES `airport` (`id_airport`) ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

CREATE TABLE `options` (

`id_opt` int(11) NOT NULL AUTO_INCREMENT,

`name` varchar(45) NOT NULL,

`value` tinyint(4) NOT NULL,

PRIMARY KEY (`id_opt`)

) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;

CREATE TABLE `passenger` (

`id_passenger` int(11) NOT NULL AUTO_INCREMENT,

`name_pax` varchar(45) NOT NULL,

`first_surname_pax` varchar(45) NOT NULL,

`second_surname_pax` varchar(45) DEFAULT NULL,

`type_passenger` enum('ADULT','CHILD','INFANT') NOT NULL,

`type_doc` int(11) NOT NULL,

`document_pax` varchar(45) NOT NULL,

`canary_resident` tinyint(1) NOT NULL,

`town` int(11) DEFAULT NULL,

`large_family` enum('NONE','GENERAL','SPECIAL') NOT NULL,

`number_large_family` varchar(45) DEFAULT NULL,

`region` int(11) DEFAULT NULL,

`ticket` int(11) NOT NULL,

`base_price` float NOT NULL,

`general_fee` float NOT NULL,

`service_fee` float NOT NULL,

`score` int(11) NOT NULL DEFAULT '0',

PRIMARY KEY (`id_passenger`), UNIQUE KEY `ticket_UNIQUE` (`ticket`),

KEY `fk_type_doc_passenger_idx` (`type_doc`), KEY `fk_town_passenger_idx` (`town`),

KEY `fk_region_passenger_idx` (`region`),

CONSTRAINT `fk_region_passenger` FOREIGN KEY (`region`) REFERENCES `region` (`id_region`) ON UPDATE CASCADE,

CONSTRAINT `fk_ticket_passenger` FOREIGN KEY (`ticket`) REFERENCES `ticket` (`id_ticket`) ON UPDATE CASCADE,

CONSTRAINT `fk_town_passenger` FOREIGN KEY (`town`) REFERENCES `town` (`id_town`) ON UPDATE CASCADE,
CONSTRAINT `fk_type_doc_passenger` FOREIGN KEY (`type_doc`) REFERENCES `document` (`id_document`) ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

```
CREATE TABLE `passenger_flight` (
  `id_passenger_flight` int(11) NOT NULL,
  `passenger` int(11) NOT NULL,
  `flight` int(11) NOT NULL,
  PRIMARY KEY (`id_passenger_flight`),
  KEY `fk_p_passenger_flight_idx` (`passenger`),
  KEY `fk_f_passenger_flight_idx` (`flight`),
  CONSTRAINT `fk_f_passenger_flight` FOREIGN KEY (`flight`) REFERENCES `flight` (`id_flight`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_p_passenger_flight` FOREIGN KEY (`passenger`) REFERENCES `passenger` (`id_passenger`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `price` (
  `id_prices` int(11) NOT NULL AUTO_INCREMENT,
  `code_price` varchar(10) NOT NULL,
  `default_price` float NOT NULL,
  `DC_price` float NOT NULL,
  `F1_price` float NOT NULL,
  `F2_price` float NOT NULL,
  `DCF1_price` float NOT NULL,
  `DCF2_price` float NOT NULL,
  PRIMARY KEY (`id_prices`)
) ENGINE=InnoDB AUTO_INCREMENT=1039 DEFAULT CHARSET=utf8;

CREATE TABLE `roles` (
  `id_rol` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(125) NOT NULL,
  PRIMARY KEY (`id_rol`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

CREATE TABLE `region` (

`id_region` int(11) NOT NULL AUTO_INCREMENT,

`code_region` varchar(45) NOT NULL,

`name_region` varchar(45) NOT NULL,

PRIMARY KEY (`id_region`),

UNIQUE KEY `code_region_UNIQUE` (`code_region`)

) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8;

CREATE TABLE `route` (

`id_route` int(11) NOT NULL AUTO_INCREMENT,

`origin` int(11) NOT NULL,

`destination` int(11) NOT NULL,

PRIMARY KEY (`id_route`),

KEY `fk_destination_route_idx` (`destination`),

KEY `fk_origin_route_idx` (`origin`),

CONSTRAINT `fk_destination_route` FOREIGN KEY (`destination`) REFERENCES `airport` (`id_airport`) ON UPDATE CASCADE,

CONSTRAINT `fk_origin_route` FOREIGN KEY (`origin`) REFERENCES `airport` (`id_airport`) ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=utf8;

CREATE TABLE `state` (

`id_state` int(11) NOT NULL AUTO_INCREMENT,

`state` varchar(45) NOT NULL,

`kiu_state` varchar(45) NOT NULL,

PRIMARY KEY (`id_state`)

) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

CREATE TABLE `route_day` (

`id_route_day` int(11) unsigned NOT NULL AUTO_INCREMENT,

`route` int(11) NOT NULL,

`day_of_week` int(11) NOT NULL,

PRIMARY KEY (`id_route_day`),

KEY `fk_r_route_day_idx` (`route`),

KEY `fk_d_route_day_idx` (`day_of_week`),

CONSTRAINT `fk_d_route_day` FOREIGN KEY (`day_of_week`) REFERENCES `day_of_week` (`id_day`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `fk_r_route_day` FOREIGN KEY (`route`) REFERENCES `route` (`id_route`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO_INCREMENT=137 DEFAULT CHARSET=utf8;

CREATE TABLE `sara` (

`id_sara` int(11) NOT NULL,

`mode_transport` int(11) NOT NULL,

`hash` varchar(255) NOT NULL,

`passenger_flight` int(11) NOT NULL,

`code_sara` varchar(3) NOT NULL,

`description` varchar(255) NOT NULL,

PRIMARY KEY (`id_sara`),

UNIQUE KEY `hash_UNIQUE` (`hash`),

UNIQUE KEY `passenger_flight_UNIQUE` (`passenger_flight`),

KEY `fk_mode_transport_sara_idx` (`mode_transport`),

CONSTRAINT `fk_mode_transport_sara` FOREIGN KEY (`mode_transport`) REFERENCES `mode_transport` (`id_mode_transport`) ON UPDATE CASCADE,

CONSTRAINT `fk_passenger_flight_sara` FOREIGN KEY (`passenger_flight`) REFERENCES `passenger_flight` (`id_passenger_flight`) ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```
CREATE TABLE `subvention` (
  `id_subvention` int(11) NOT NULL AUTO_INCREMENT,
  `resident` int(11) NOT NULL,
  `large_family_general` int(11) NOT NULL,
  `large_family_special` int(11) NOT NULL,
  PRIMARY KEY (`id_subvention`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

CREATE TABLE `ticket` (
  `id_ticket` int(11) NOT NULL AUTO_INCREMENT,
  `ticket` varchar(13) NOT NULL,
  `state` int(11) NOT NULL,
  PRIMARY KEY (`id_ticket`), KEY `fk_state_ticket` (`state`),
  CONSTRAINT `fk_state_ticket` FOREIGN KEY (`state`) REFERENCES `state` (`id_state`)
ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

CREATE TABLE `town` (
  `id_town` int(11) NOT NULL AUTO_INCREMENT,
  `code_town` varchar(12) NOT NULL,
  `name_town` varchar(45) NOT NULL,
  PRIMARY KEY (`id_town`), UNIQUE KEY `code_town_UNIQUE` (`code_town`)
) ENGINE=InnoDB AUTO_INCREMENT=90 DEFAULT CHARSET=utf8;

CREATE TABLE `type_fare` (
  `id_type_fare` int(11) NOT NULL AUTO_INCREMENT,
  `code_type_fare` varchar(3) NOT NULL,
  `name_type_fare` varchar(45) NOT NULL,
  `description` varchar(255) DEFAULT '',
  PRIMARY KEY (`id_type_fare`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `user` (
  `id_user` int(11) NOT NULL,
  `name_user` varchar(45) DEFAULT NULL,
  `first_surname_user` varchar(45) DEFAULT NULL,
  `second_surname_user` varchar(45) DEFAULT NULL,
  `type_user` enum('ADULT','CHILD','INFANT') DEFAULT NULL,
  `date_birth` date DEFAULT NULL,
  `type_doc` int(11) DEFAULT NULL,
  `document_user` varchar(45) DEFAULT NULL,
  `large_family` enum('0','GENERAL','SPECIAL') DEFAULT NULL,
  `number_large_family` varchar(45) DEFAULT NULL,
  `region` int(11) DEFAULT NULL,
  `canary_resident` tinyint(4) DEFAULT '0',
  `town` int(11) DEFAULT NULL,
  `score` int(11) NOT NULL DEFAULT '0',
  `email` varchar(45) NOT NULL,
  `password_user` varchar(32) NOT NULL,
  `telephone` varchar(45) DEFAULT NULL,
  `id_rol` int(11) NOT NULL,
  PRIMARY KEY (`id_user`),
  UNIQUE KEY `email_UNIQUE` (`email`), KEY `fk_type_doc_user_idx` (`type_doc`),
  KEY `fk_town_user_idx` (`town`), KEY `fk_region_user_idx` (`region`),
  CONSTRAINT `fk_region_user` FOREIGN KEY (`region`) REFERENCES `region`
(`id_region`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_town_user` FOREIGN KEY (`town`) REFERENCES `town` (`id_town`)
ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_type_doc_user` FOREIGN KEY (`type_doc`) REFERENCES `document`
(`id_document`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

14. Referencias

[1] - Estudio por parte de Gartner que informa sobre cómo Android abarca el mercado.

[2] - Artículo de la Wikipedia sobre los GDS.

[3] - Web corporativa de Kiu Systems Solutions.

[4] - Información referente al Web Service de Kiu Systems Solutions.

[5] - Artículo de la Wikipedia sobre Google Play Store.

[6] - Información de Google Play sobre Skyscanner.

[7] - Información de Google Play sobre KAYAK.

[8] - Información de Google Play sobre EasyJet.

[9] - Artículo de la Wikipedia sobre EasyJet.

[10] - Información de Google Play sobre Vueling.

[11] - Artículo de la Wikipedia sobre Vueling.

[12] - Información de Google Play sobre Ryanair.

[13] - Artículo de la Wikipedia sobre Ryanair.

[14] - Artículo de la Wikipedia sobre Amadeus.

[15] - Documento informativo sobre el proyecto docente del TFG (Curso 2014/2015).

[16] - Web corporativa de Binter Canarias.

[17] - Artículo de la Wikipedia sobre Licencias de software.

[18] - Licencia GNU.

[19] - Artículo de la Wikipedia sobre CDDL.

[20] - Convenio sobre la ciberdelincuencia.

[21] - Delitos informáticos.

[22] - Información sobre Android Studio.

[23] - Artículo de la Wikipedia sobre NetBeans.

[24] - Artículo de la Wikipedia sobre Workbench.

[25] - Artículo de la Wikipedia sobre Mozilla Firefox.

[26] - Artículo de la Wikipedia sobre Apache.

[27] - Información sobre Android SDK.

[28] - Información sobre la herramienta Draw.io.

[29] - Información sobre terminal de Ubuntu.

[30] - Artículo de la Wikipedia sobre PHP.

[31] - Artículo de la Wikipedia sobre MySQL.

[32] - Artículo de la Wikipedia sobre XML.

[33] - Artículo de la Wikipedia sobre JSON.

[34] - Artículo de la Wikipedia sobre cURL.

[35] - Información relativa a la implantación del sistema SARA de forma obligatoria.

[36] - Artículo de la Wikipedia sobre Actores (UML).

[37] - Artículo de la Wikipedia sobre Diagrama de casos de uso.

[38] - Artículo de la Wikipedia sobre UML.

[39] - Información sobre vuelos en grupo.

[40] - Información de vuelos chárter.

[41] - Noticia del alquiler por parte del CD Tenerife de un avión chárter de Canaryfly.

[42] - Artículo de la Wikipedia sobre la Programación por capas.

[43] - Información sobre el diseño Android para diferentes tipos y tamaños de pantalla.

[44] - Página de PHPMailer.

[45] - Artículo de la Wikipedia sobre los códigos IATA.

[46] - Proyecto caldroid.

[47] - Información de Google Play sobre PassWallet.

[48] - Información sobre la cumplimentación de las leyes LOPD y LSSI-CE.

Otras fuentes de información externas han sido la página oficial de Android, la web de AndroidCurso, el blog Androideity, el blog Hello Green Rules, los tutoriales de las listas de reproducción de los canales de Youtube xymind, codigofacilito y videotutoriales.com de Jesús Conde, además por supuesto de stackoverflow para buscar respuesta a los problemas que iban surgiendo.