Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e Implementación de una aplicación para la gestión de compra de viajes y excursiones en dispositivos móviles.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Víctor Manuel Velásquez Soplatz

Tutor: David De Andrés Martínez

2016/2017

Resumen

Este proyecto consiste en la creación de una aplicación móvil desarrollada para sistemas operativos Android, cuya principal funcionalidad, es que cada usuario pueda comprar de una lista de Viajes y Excursiones publicada, la actividad que más le interese.

El proceso de desarrollo de la aplicación se ha llevará a cabo en tres fases. Un primer proceso de análisis para especificar los requisitos de la aplicación mediante los casos de uso, un segundo proceso de diseño centrado en el usuario, atendiendo sobre todo al contexto de uso, y posteriormente, se ha realizará la implementación y desarrollo de la aplicación utilizando tecnologías como Java y XML a través de Android Studio, además de los servicios de Google Play Services necesarios para la autenticación de los usuarios.

Por último, la aplicación utilizará una base de datos NoSQL alojada en la nube llamada Firebase Realtime Database, donde se almacenan los usuarios, sus listas de compra, y sus amigos. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada usuario conectado.

Palabras clave: Aplicación móvil, Android, lista de compras, categorías, diseño centrado en el usuario, autenticación, nube.

Abstract

This project consists of the creation of a mobile application developed for Android operating systems, whose main functionality is that each user can buy from a list of trips and tours published, the activity that interests you.

The application development process will be carried out in three phases. A first process of analysis to specify the requirements of the application through the use cases, a second process of design centered on the user, taking into account the context of use, and subsequently, the implementation and development of the application will be made using Technologies such as Java and XML through Android Studio, in addition to Google Play services required for user authentication.

Finally, the application will use a cloud-hosted NoSQL database called Firebase Realtime Database, where users, their shopping lists, and friends are stored. The data is stored in JSON format and synchronized in real time with each connected user.

Keywords: Mobile application, Android, shopping list, categories, user-centered design, authentication, cloud.

Tabla de contenidos

1. Introducción

El futuro de las aplicaciones ya ha llegado. El sector de las aplicaciones móviles está en un claro auge y se pronostica que ésta tendencia seguirá a la alza durante los próximos años. El crecimiento de las apps parece no tener fin, así que las empresas ya están dando la bienvenida a este nuevo canal de comunicación.

Hay muchos motivos por los que es muy beneficioso tener una aplicación hoy en día, pero si hay que centrarse en en uno éste es su futuro prometedor. Se prevén más de 268 mil millones de descargas en el año 2017 y España encabeza el ranking europeo de descargas de apps móviles.

Las grandes guías telefónicas servían para conseguir un número de teléfono o la dirección de algún restaurante popular, para comprar boletos al concierto de su cantante favorito, se tenía que hacer fila y muy larga, poder estar en una ciudad y querer hacer un tour o simplemente querer viajar dentro del país de tus vacaciones….ahora con su móvil puede lograr estas actividades y otras más, con tan solo descargar una o varias aplicaciones en su móvil.

Es por esto que ahora más que nunca me sirve como base para plantearme el desarrollo y la creación una aplicación móvil para la compra de Viajes y Excursiones, para el colectivo de estudiantes de intercambio Erasmus, ya que al ser un publico de unos cinco mil estudiantes al año, abre una gran oportunidad de promocionar mi aplicación, y después de esto empezar a introducir nuevas funcionalidades para poder expandir el tipo de publico y llegar a más personas.

1.1 Motivaciones

Uno de los motivos por los que decidí desarrollar el proyecto en la plataforma Android está relacionado con el auge actual de la tecnología utilizada para el desarrollo de la aplicación, el éxito de los smartphones y las posibilidades que ofrecen. De esta forma, he podido adquirir conocimientos de desarrollo destinado a estas plataformas en oposición a otras tecnologías como los ordenadores de sobremesa.

Otro de los motivos consiste en el sistema operativo al que va dirigida la aplicación, en este caso Android, un sistema operativo que ya lleva años en el mercado y que cada vez está más extendido, tanto en smartphones como en tablets. Gran parte de su popularidad es debido a que es de código abierto lo que permite un sin fin de posibilidades nuevas. Debido a esto, hay que agradecer el gran apoyo que recibimos todos los desarrolladores de Google, con nuevos métodos para conectar toda la aplicación con todo el mundo.

Por último, la posibilidad de desarrollar una aplicación para dispositivos móviles en un ámbito que no se estudia ampliamente a lo largo de la carrera como materia obligatoria, por lo que era una buena oportunidad para obtener conocimiento en dicho ámbito y poder desarrollar una aplicación para el mercado Erasmus actual en Valencia, en donde trabajo hace un par de

años, ya que es un mercado bastante interesante, porque ellos al empezar su vida estudiantil no conocen Valencia ni España, así que la creación de la aplicación me servirá para brindarles este servicio de compra de excursiones y viajes.

1.2 Objetivos

El objetivo principal de este futuro Trabajo de Final de Grado es la creación de una aplicación móvil para la gestión de compras de viajes y excursiones.

La aplicación en un principio, está dirigida al colectivo de estudiantes de Erasmus en Valencia pero también puede estar dirigida a cualquier persona del planeta que desee administrar sus listas de compra de viajes y excursiones con un diseño novedoso y sencillo.

Desde el punto de vista académico tiene los siguientes objetivos:

- Diseñar, implementar y evaluar una aplicación móvil de gestión de compras de viajes y excursiones desde cero.
- Adquirir experiencia en la ejecución de aplicaciones en entornos de depuración, así como en el entorno de desarrollo Android Studio.
- Adquirir experiencia en la configuración de sistemas de gestión de bases de datos como SQLite.
- Adquirir nociones para diseñar la aplicación centrándose en el usuario, obteniendo como resultado un diseño sencillo e intuitivo.
- Adquirir nociones sobre la plataforma Firebase, para llevar un control sobre los usuarios, sobre los amigos y sobre las listas de viajes y excursiones compartidas.

1.3 Estructura

Este apartado está diseñado para describir con claridad las fases que ha atravesado la aplicación durante su ciclo de vida. Está estructurado en las siguientes partes:

- **Introducción**: En este apartado se introduce el proyecto, así como la motivación, objetivos y estructura.
- **Estado del arte**: En este punto se hará un análisis de dos aplicaciones similares en Google Play Store, resumiendo las características comunes.
- **Contexto tecnológico**: Este capítulo describirá los entornos de desarrollo utilizados para realizar la aplicación y las tecnologías empleadas como Java, Android y Firebase, una potente plataforma de desarrollo móvil en la nube de Google.
- **Especificación de Requisitos**: Durante este apartado se hará una descripción completa del comportamiento del sistema que se va a desarrollar. Por una parte, se hablará de la

funcionalidad que el cliente desea tener en su aplicación, y por otra, la funcionalidad que va a implementarse.

- **Análisis**: Este punto describirá las fases de análisis necesarias para obtener el modelo conceptual, concretamente se utilizaran los casos de uso.
- **Diseño**: Capítulo que describirá cuáles son los diseños a los que tendrá que ajustarse la aplicación, así como las metodologías que se han utilizado.
- **Implementación**: En este apartado se mencionarán las herramientas utilizadas durante la fase de implementación, y se tratarán temas como el almacenamiento externo, la gestión de bases de datos en SQLite, las clases necesarias para manejar correctamente los datos y el uso y manejo de firebase para autentificar en la nube.
- **Evaluación**: Punto en el que se describirán las técnicas de evaluación de la aplicación y en el que se resumirán los resultados obtenidos en varios móviles con pantallas de diferente tamaño.
- **Conclusiones**: En este capítulo se hará una reflexión sobre el trabajo realizado, analizando si se han cumplido los objetivos propuestos. Además, de una valoración personal sobre los conocimientos adquiridos y sobre el desarrollo del proyecto.

1.4 Siglas y Acrónimos

HTML: (*HyperText Markup Language*) Lenguaje de marcas de hipertexto, considerado el elemento de construcción más básico de una página web. Usado para crear y representar visualmente una página web determinando el contenido de la misma, pero no su funcionalidad.

CSS: (*Cascading Style Sheets*) Hojas de Estilo en Cascada, es un simple mecanismo o lenguaje utilizado para describir cómo debe ser mostrado un documento en la pantalla, a la hora de imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. URL: (*Uniform Resour- ce Locator*) Localizador de recursos uniforme, es un identificador recursos dentro de una red, como Internet, los cuales pueden cambiar con el tiempo mientras que la dirección es la misma. Muy utilizado en la *World Wide Web*.

JAR: (*Java ARchive*) Archivo java, es un paquete de archivos comprimido que contiene clases Java y recursos para ejecutar aplicaciones software o librerías de *Java platform*.

APK: (*Android Application Package*) Paquete de aplicación Android, es un paquete de archivos variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android.

UML: (*Unified Modeling Language*) Lenguaje unificado de modelado, es un lenguaje de modelado para construir y documentar un sistema.

JSON: (*JavaScript Object Notation*) Notación de objetos JavaScript, es un formato de texto ligero utilizado para el intercambio de mensajes y datos. Es el formato de mensajes más utilizado, por encima de XML, llegando a considerarse como un lenguaje indepen- diente.

XML: (*eXtensible Markup Language*) Lenguaje de marcas extensible, permite almace- nar datos de forma legible lo que le hace ser muy útil en el intercambio de datos entre aplicaciones.

API: (*Application Programming Interface*) Interfaz de programación de aplicaciones, es una biblioteca que ofrece un conjunto de métodos o funciones capaces de ser llamados o utilizados por otra aplicación como una capa de abstracción.

2 Estado del arte

En este apartado, expondré una visión general del estado en el que se encuentra la plataforma Android en el mercado de dispositivos móviles actual.

Actualmente en el mercado existen varias aplicaciones con una funcionalidad parecida a la de nuestra aplicación. Algunas de ellas comparten varios objetivos perseguidos en este proyecto como la posibilidad de guardar tus eventos personales y compartirlas con tus amigos. Debido al tiempo que llevan las aplicaciones en Google Play Store, muchas funcionalidades han quedado desfasadas y hoy en día, se pueden incluir de forma más fácil y con un coste menor.

De todas las aplicaciones que existen ahora mismo en el mercado se ha elegido una con más similitud a nuestros objetivos. Por lo tanto, vamos a realizar un análisis con las características más importantes y sobretodo, con las funcionalidades que se van a mejorar.

2.1 Tendencias en Dispositivos Móviles

Para el desarrollo de la aplicación se ha elegido la plataforma Android. Esta elección se fundamenta en el hecho de que el sistema operativo Android ha ido creciendo de forma vertiginosa sobre los demás desde el año 2010 como se puede observar en la siguiente figura.

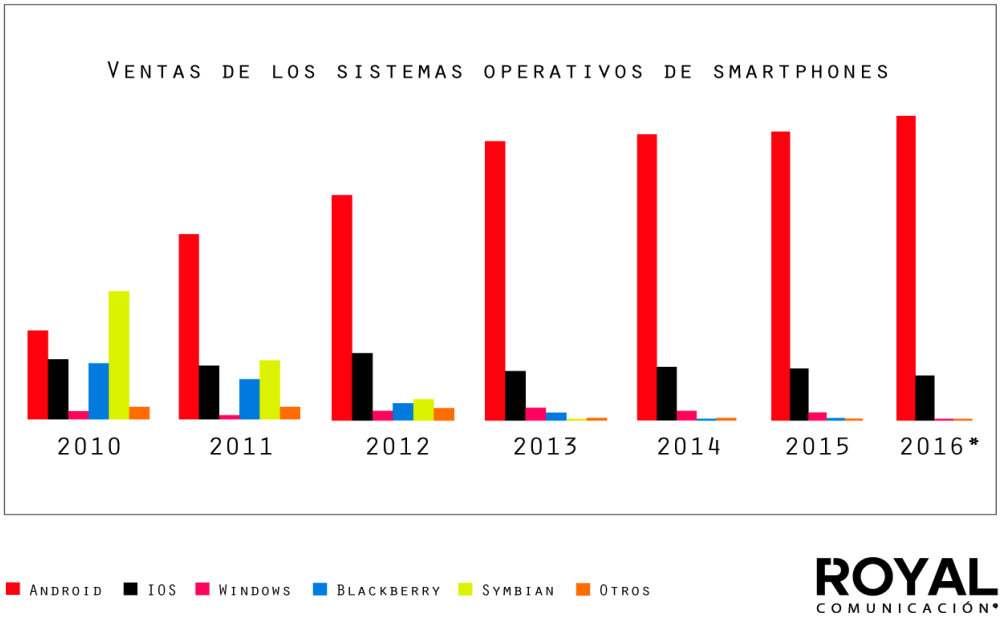

Figura 2.1: Crecimiento del S.O Android en los últimos años

En la actualidad, mas del 85% de los smartphones del Mercado español usan el sistema operativo de Google tal y como vemos en la siguiente figura.

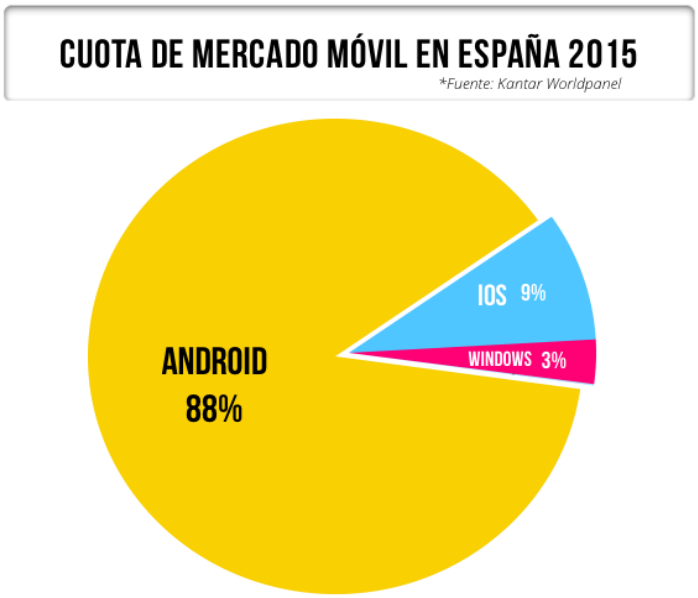

Figura 2.1.1: Cuota del Mercado en España en el 2015

Debido a que es el más utilizado, y la previsión es que seguirá siéndolo durante los próximos años, una aplicación desarrollada para este sistema operativo podrá ser instalada en más dispositivos y así permitir su uso a un mayor número de estudiantes Erasmus.

2.2 Otras Aplicaciones en el Mercado

2.2.1 Aplicación de eventos – EventBrite

Esta aplicación se puede descargar gratuitamente para sistemas operativos Android y cuenta con más de 5.000.000 de usuarios desde 2010. Su objetivo principal es proporcionar eventos locales populares y además de enterarte de los eventos que asisten tus amigos. Esta aplicación no solo se centra en el tema de viajes y excursiones, sino, también a la eventos culturales, conciertos, etc.

Entre las características más interesantes destacan:

- La similitud en el diseño con una lista de eventos, como se observa en la imagen, además de tener a mano las opciones de Menú en la parte inferior de la aplicación.

- La opción de compartir los eventos y por múltiples vías: WhatsApp, Messenger, Facebook, Correo, Mensajes, etc

- La facilidad a la hora de buscar eventos, ya que se pueden organizan y filtrar fácilmente por categorías y por ciudades.

- La falta de un sistema para autentificar a los usuarios de la aplicación, que no sea sólo por email, sino ya sea por Google o Facebook.

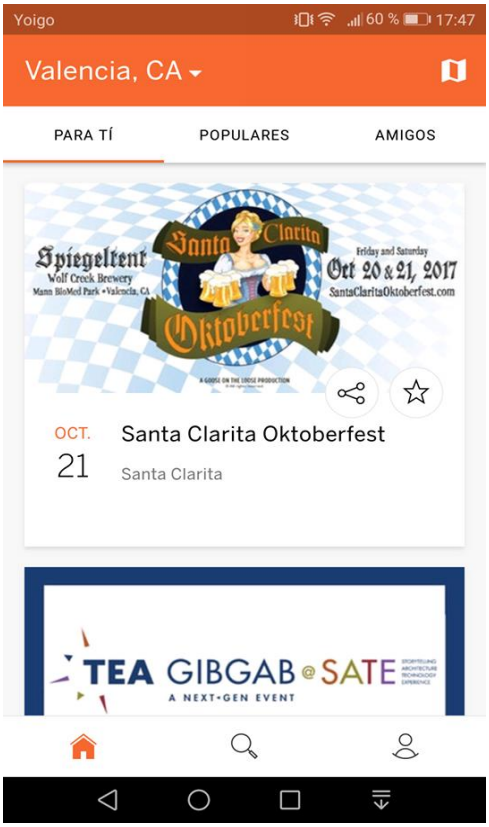

Figura 2.1: Pantalla de Eventos de EventBrite

2.2.2 Aplicación de eventos - All events in city

En este caso la aplicación también se puede descargar gratuitamente para sistemas operativos Android contando con más de 50.000 usuarios desde 2014. Esta aplicación cuenta con un diseño gráfico muy trabajado en comparación con la aplicación anterior, lo que es muy importante a la hora de realizar una aplicación de este tipo. Su objetivo principal es mostrar eventos, pero en este caso, al entrar a la aplicación te pide permisos de localización para poder enseñar los eventos que están cerca de tu localización. También añade la opción de compartirlas por muchas plataformas como puede ser Gmail o WhatsApp.

Las características más importantes serían:

- Su diseño novedoso e intuitivo, incluyendo además de categorías disponibles en las que puedes clasificar casi todos los eventos.

- En comparación a la otra aplicación, esta si dispone de un servicio de autentificar por

medio de Facebook, Gmail, etc.

En este caso, la aplicación nos permite compartir lasmediante otras plataformas y desde la misma aplicación. Esta opción es muy parecida la plataforma Firebase en nuestra aplicación.

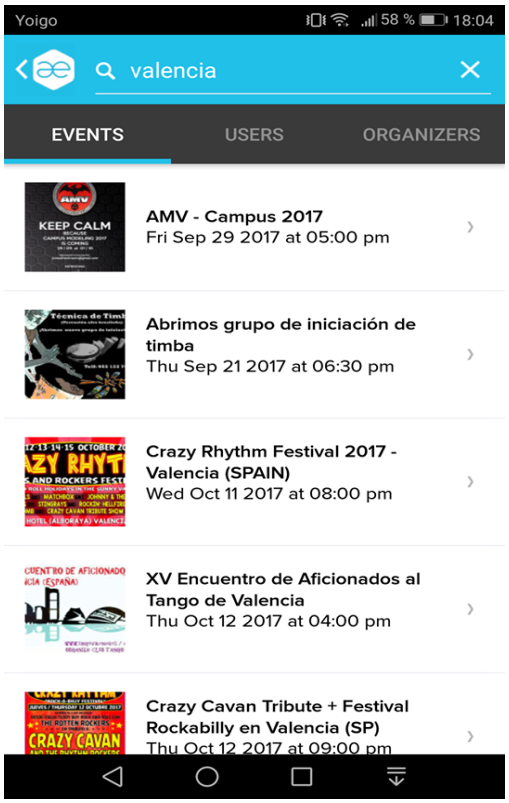

Figura 2.2: Pantalla de Eventos de All events in city

En la siguiente tabla podemos observar una comparativa entre las diversas características que aplicación posee.

	Erasmus	EventBrite	All Event
Login con otra herramienta			
Menús Sencillo			
Mapas de la ciudad			
Calendario integrado			
Intuitiva de usar			
Información de Amigos			
Compra por otros métodos pago			
Herramienta de traducción			

Tabla 1: Comparación de funcionalidades entre aplicaciones

3 Contexto Tecnológicos

En este punto se van a exponer los distintos entornos de programación, lenguajes, y tecnologías que se han utilizado para desarrollar la aplicación, sin entrar en detalles de implementación. También se van a describir las técnicas de debugging y el control de versiones que se ha llevado a lo largo del ciclo de vida de la aplicación.

Para acabar se introducirá la plataforma Firebase y se explicarán las diferentes opciones que se pueden habilitar al diseñar nuestra aplicación utilizando esta plataforma.

3.1 Entorno de desarrollo

En este primer apartado se van a mencionar cuáles han sido los entornos que se han utilizado para desarrollar la aplicación, tanto a nivel de sistema operativo como de entorno de programación.

3.1.1 Android Studio

Android Studio, es el IDE oficial para el desarrollo de aplicaciones en la platafor- ma Android. Partiendo de la herramienta IntelliJ IDEA, Android Studio ofrece aún más características que mejoran su productividad en la construcción de aplicaciones Android, tales como:

- Un sistema de construcción basado en Gradle flexibles.
- Construir variantes y generación de archivos APK múltiples.
- Plantillas de código para ayudar a desarrollar características comunes.
- Un rico editor de diseño con soporte para la edición de arrastrar y soltar.
- Herramientas para capturar excepciones, medir rendimiento, facilidad de uso, compatibilidad de versiones, y otros problemas.
- Reducción de código con ProGuard y menores recursos con Gradle.

Considerada la mejor herramienta para desarrollar aplicaciones móviles nativas, pero a su vez, también requiere una gran curva de aprendizaje y grandes conocimientos de Java. Esto último ha sido la razón que nos ha llevado a desarrolar la aplicación con esta gran herramienta.

Figura 3.1.1: Logotipo de Android Studio

3.2 Java

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para funcionar en otra. Desde 2012, es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web.

3.3 Android

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

El sistema permite programar aplicaciones en una variación de Java llamada Dalvik que veremos a continuación. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Esta sencillez, junto a la existencia de herramientas de programación gratuitas, hacen que una de las cosas más importantes de este sistema operativo sea la cantidad de aplicaciones disponibles, que extienden casi sin límites la experiencia del usuario. En la figura 3 se observan los componentes por los que está formado el sistema de Android.

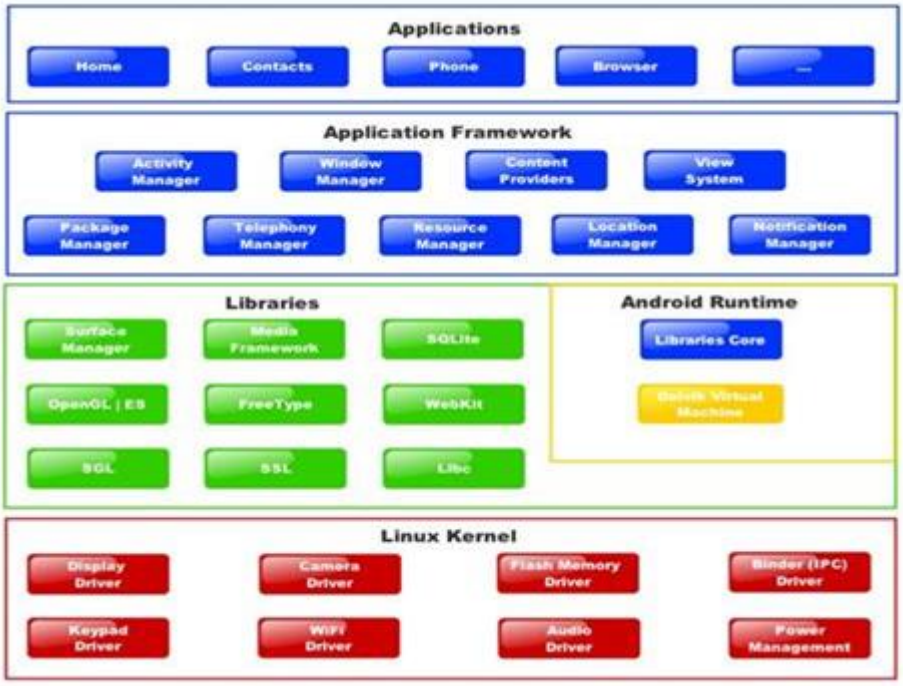

Figura 3.3: Ilustración de component Sistema Android

3.3.1 XML

XML proviene de Extensible Markup Language (Lenguaje de Marcas Extensible). Se trata de un metalenguaje (un lenguaje que se utiliza para decir algo acerca de otro) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium (W3C), una sociedad mercantil internacional que elabora recomendaciones para la World Wide Web.

Dado que en gran parte, la utilidad de una herramienta depende de la creatividad de quien la utiliza, resulta imposible resumir todas las aplicaciones de XML. En pocas palabras, se puede decir que ofrece la posibilidad de estructurar y representar datos. Pero además de facilitar la organización de los recursos y la configuración de un programa, cumple un papel muy importante que es, sin lugar a dudas, su punto fuerte: le permite comunicarse con otras aplicaciones, de diferentes plataformas y sin que importe el origen de la información en común.

3.3.2 SQLite

Es un motor de bases de datos muy popular en la actualidad por ofrecer características tan interesantes como su pequeño tamaño, no necesitar servidor, precisar poca configuración, ser transaccional y por supuesto ser de código libre. A diferencia de los sistema de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo.

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias.

3.3.3 Google Play Store

Es una librería que contiene las interfaces con los servicios individuales de Google y que permite obtener la autorización de los usuarios para obtener acceso a estos servicios con sus credenciales. También contiene las API que permiten resolver cualquier problema en tiempo de ejecución.

Utilizar los servicios de Google Play permite la libertad de usar las nuevas APIs para utilizar las herramientas populares sin preocuparse de la compatibilidad en los dispositivos. Los cambios se distribuyen automáticamente por Google Play Store, y las nuevas versiones de la librería se obtienen a través del SDK de Android, lo que hace que sea más fácil desarrollar la aplicación sin tener que preocuparse de estos servicios.

3.3.4 Técnicas de Debugging

Para desarrollar una aplicación Android de este tipo, hay que realizar muchas tareas de debugging (depuración). Es muy importante disponer de un entorno que nos muestre la mayor información posible acerca de los errores y los valores que tiene asignados cada variable en tiempo real. Android Studio ofrece un entorno muy bien pensado para la depuración de código Java, debido a esto, ha sido una de las técnicas que más me ha ayudado a lo largo del desarrollo.

3.3.5 Control de Versiones

Es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. Una versión o revisión de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación. Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas.

Ejemplos de este tipo de herramientas son entre otros: CVS, Subversion, GitHub, etc. Este último ha sido el que he utilizado para el control de versiones en mi aplicación ya que tiene muy buena integración en Android Studio.

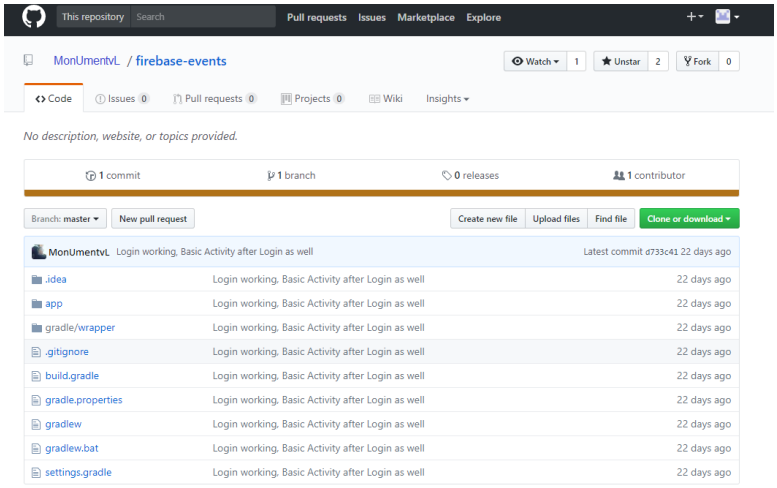

Figura 3.3: Aplicación web de GitHub

3.4 Firebase – Google

Firebase es la nueva y mejorada plataforma de desarrollo móvil en la nube de Google. Se trata de una plataforma disponible para diferentes plataformas (Android, iOS, web). En muchas ocasiones nos planteamos cómo acceder a un servicio web para tener nuestra aplicación trabajando con datos en la nube. Por ello surgió Firebase, para proveer una API donde guardar y sincronizar los datos en la nube en tiempo real. Uno de los aspectos que más hay que destacar es la asombrosa documentación que se puede consultar cuando accedemos a la plataforma. Hay un gran cantidad de información interesante y necesaria disponible para todo aquel desarrollador que quiera probar suerte en esta plataforma.

Sus características fundamentales están divididas en varios grupos, podemos agruparlas en:

- Analíticas: Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.

- Desarrollo: Permite construir mejores apps, permitiendo delegar determinadas operaciones en Firebase, para poder ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación, entre otras.

- Crecimiento: Permite gestionar los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello dispondremos de funcionalidades como las de invitaciones, indexación o notificaciones.

- Monetización: Permite ganar dinero gracias a AdMob.

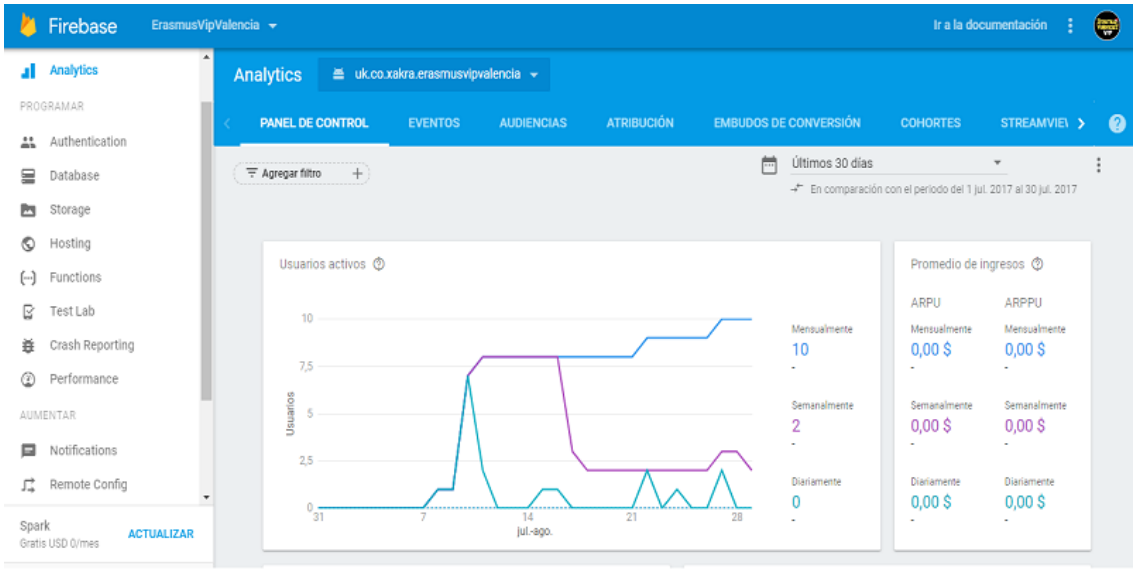

Figura 3.4: Aplicación web de Firebase

3.4.1 Autenticación

Muchas de la apps que existen actualmente reconocen la identidad de un usuario. El reconocimiento de la identidad de un usuario permite guardar los datos de este de manera segura en la nube, ya sea para consultar sus datos o para añadir nuevos.

Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya hechas para autenticar usuarios en tu app. Admite autenticación con contraseñas y proveedores de identidades federadas populares, como Google y Facebook. Se integra estrechamente con otros servicios de Firebase y aprovecha estándares industriales como OAuth 2.0 y OpenID Connect. Por lo tanto, se puede integrar fácilmente con tu backend personalizado.

Puedes iniciar sesiones usando FirebaseUI como una solución de autenticación directa completa, o bien usar el Firebase Authentication SDK para integrar manualmente uno o varios métodos de inicio de sesión a tu app. En nuestro caso, hemos elegido la segunda opción dado que tenemos varios métodos de inicio de sesión.

3.4.2 Base de datos en tiempo real

Firebase dispone de una base de datos NoSQL alojada en la nube donde se pueden almacenar y sincronizar los datos que vamos generando en nuestra aplicación. Estos datos son sincronizados con todos los clientes en tiempo real y siguen estando disponible cuando la aplicación pierde la conexión.

La Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. También proporciona un lenguaje de reglas flexibles basadas en expresiones llamado Security

Rules de Firebase Realtime Database, donde se define el modo en que los datos se deben estructurar y el momento en que se pueden someter a lectura y escritura. Es importante pensar en el modo en que los usuarios necesitan acceder a los datos y luego estructurarlos de forma adecuada.

3.4.2.1 Estructuración de los datos

La construcción de una base de datos estructurada de manera apropiada requiere un poco de previsión. Lo más importante es que necesitas planear cómo guardar los datos y luego recuperarlos para que ese proceso sea lo más sencillo posible.

Todos los datos de Firebase Realtime Database se almacenan como objetos JSON. Se plantea la base de datos como un árbol JSON alojado en la nube. A diferencia de la base de datos SQL, no existen tablas ni registros. Cuando se agregan datos al árbol JSON, estos se convierten en un nodo en la estructura JSON existente.

Los datos pueden anidarse hasta 32 niveles de profundidad, no obstante no hay que excederse.

Cuando obtienes datos en una ubicación de tu base de datos, también recuperas todos los nodos secundarios. En la práctica, es mejor mantener la estructura de datos con la mayor simpleza posible. Si los datos, en cambio, se dividen en rutas separadas, también llamadas no normalizadas, se pueden descargar de manera eficiente en llamadas separadas, a este tipo de estructura se le llama compactada.

3.5 Herramientas

3.5.1 Visual Paradigm for UML

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML como su nombre indica. Según su definición y comentarios aportados por los usuarios es ideal para Arquitectos de sistemas, Analistas de Sistemas e Ingenieros de Software que estén interesados en la construcción de sistemas a gran escala. Dispone de un entorno gráfico para visualizar, especificar, construir y documentar el sistema.

Figura 3.5.1: Logotipo de Visual Paradigm for UML

3.5.2 WireframeSketcher

WireframeSketcher, es una herramienta que ayuda a crear rápidamente diagramas,

maquetas y prototipos para aplicaciones de escritorio, web y móviles. Podemos encontrarla tanto como un *plug-in* para cualquier IDE basado en Eclipse como en su versión independiente. Con WireframeSketcher se puede reunir rápidamente la información útil de las partes interesadas, mostrar sus propuestas a los clientes, compartir sus ideas con otros desarrolladores y al final construir un software mejor.

Se ha optado por esta herramienta por haber sido utilizada anteriormente en Desarrollo centrado en el usuario, además de ser simple y fácil de usar, su versión de prueba es totalmente funcional para un proyecto.

Figura 3.5.2: Logotipo de WireframeSketcher

4 Especificación de Requisitos

En este apartado se va dar una descripción completa del comportamiento del sistema que se va a desarrollar. Más adelante definiremos el conjunto de casos de uso que describe todas las interacciones que van a tener los usuarios con el software. Los casos de uso también son conocidos como requisitos funcionales. Además de los casos de uso, también contiene requisitos no funcionales (o complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación.

Está dirigida tanto al cliente como al equipo de desarrollo. El lenguaje utilizado para la redacción debe ser informal, de forma que sea fácilmente comprensible para todas las partes involucradas en el desarrollo.

4.1 Introducción

En primer lugar es necesario definir algunas características del proyecto, tales como el propósito y el ámbito, requerimientos técnicos por los que ha de regirse el desarrollo de la aplicación.

4.1.1 Propósito

El propósito de este capítulo es definir los requerimientos que debe tener y cumplir la aplicación desarrollada. Esta especificación de requisitos tiene como objetivo formalizar las funcionalidades, de forma que haya una base con la que contrastar el desarrollo de la aplicación, y así poder realizar el desarrollo de una forma más sencilla y guiada.

También se harán varias entrevistas con los usuarios finales para obtener la descripción funcional de la aplicación. Estas entrevistas se programarán de tal forma que permitan a los analistas reflexionar sobre la información recibida, de esta manera se evitará recibir información de forma ambigua o contradictoria.

4.1.2 Ámbito

La aplicación está destinada a dispositivos móviles Android en la cual se realizarán todas y cada una de las distintas acciones posibles.

Dentro de las características que implementa la aplicación, el usuario puede registrarse o identificarse para habilitar las funciones de visualización y compra de las actividades.

La funcionalidad principal podría resumirse en permitir la compra de actividades y poder compartir esta información con tus amigos.

4.2 Descripción general

A continuación, se detallan una serie de apartados que describen los objetivos del producto, así como su funcionalidad y requisites.

4.2.1 Perspectiva del Producto

El acceso a la aplicación se puede realizar desde cualquier parte del planeta mediante un dispositivo móvil. Si se dispone de acceso a Internet se podrá visualizar y comprar productos de Viajes y Excursiones. La aplicación está dirigida a cualquier persona que desee compartir sus compras de viajes ó excursiones, ya sean a sus amigos, familiares o compañeros de un piso de estudiantes.

La aplicación está diseñada en Android Studio. Utiliza Java como lenguaje de programación y la API de Android, para habilitar los distintos componentes que usamos en la aplicación.

Debido a que la aplicación contiene una serie de características propias, es necesario usar el servicio de alojamiento web que proporciona Firebase.

4.2.2 Funciones del Producto

Este apartado describe cuáles son las funcionalidades que debe proporcionar la aplicación.

Las funciones principales que la aplicación debe permitir son las siguientes:

1. *Gestión de registro e identificación:*

a) Registrar email.

b) Registrar mediante Facebook.

c) Identificarse con un usuario registrado.

d) Cerrar sesión.

2. *Gestión de Producto:*

a) Consultar un producto

b) Escoger un producto con su precio.

c) Consultar precio total.

d) Comprar producto

e) Consultar producto comprado.

4.2.3 Características de usuario

La aplicación solo puede ser manejada por un usuario identificado. Este usuario debe ser capaz de realizar todas las operaciones y funciones comentadas en el apartado anterior, con la única restricción que requerira acceso a Internet.

4.2.4 Restricciones generals

En cuanto a las restricciones, la necesidad de trabajar con una base de datos que tiene una estructura preestablecida, puede hacer que funcione de manera incorrecta si se modifica dicha estructura o se elimina parcialmente. En caso de tener que modificar la estructura de la base de datos, será necesario realizar correcciones en la aplicación.

A la hora de usar la aplicación, hay que tener en cuenta que es necesario disponer de acceso a internet. Esto puede realizarse mediante conexión Wi-Fi o mediante conexiones móviles 4G. Hay que tener en cuenta que la velocidad con la que la aplicación responda a las peticiones dependerá tanto de la velocidad de la conexión

como del rendimiento del servidor de Google en ese momento, así pues, siempre será mejor disponer de una conexión de alta velocidad.

4.3 Requisitos Especificos

En este apartado se hace una descripción más exhaustiva de todos aquellos requerimientos que debe poseer la aplicación, tanto a nivel funcional como de interfaz y diseño. Para concluir, se mencionarán algunos atributos presentes en la aplicación.

4.3.1 Requisitos funcionales

4.3.1.1 Registrar mediante email ó Facebook

- Introducción: Los usuarios pueden elegir registrarse mediante dos métodos: email o mediante Facebook.
- Entrada: Botón registrarse en la pantalla de login.
- Proceso: El usuario accede a la pantalla de login, introduce sus datos y presiona el botón de registrarse o en la pantalla de bienvenida, presiona el botón de Facebook.
- Salida: Si no ocurre ningún error, la aplicación añade el email a la base de datos en la nube de Firebase.

4.3.1.2 Identificarse

- Introducción: Los usuarios pueden elegir identificarse en la aplicación mediante su email o mediante Facebook.
- Entrada: Botón login en la pantalla de login, para hacerlo seleccionar boton de email o con el botón de Facebook en la pantalla de bienvenida, para hacerlo con la cuenta de Facebook.
- Proceso: El usuario introduce sus datos y presiona el botón de login o en la pantalla de bienvenida, presiona el botón de Facebook.
- Salida: Si no ocurre ningún error la aplicación añade el email a la base de datos en la nube de Firebase.

4.3.1.3 Cerrar Sesión

- Introducción: Solo los usuarios que se han identificado en la aplicación pueden elegir esta opción, sirve para salir de la session actual.

- Entrada: Opción cerrar sesión en la sección de *Mi Perfil* de la aplicación.

- Proceso: El usuario accede a *Mi Perfil*, y selecciona la opción de cerrar sesión.

4.3.1.4 Escoger un producto e incremetar precio

- Introducción: Con esta opción se puede añadir más número de productos comprados junto con su precio.
- Entrada: Botón de añadir en la pantalla, para modificar el número de productos.
- Proceso: El usuario accede a la pantalla de futuros eventos, elige la actividad y pulsa el botón ok del diálogo para entrar en él, elige número de participantes y pulsa el botón ok del diálogo para modificar el precio.
- Salida: La aplicación añade el producto junto con su precio a la lista y, a la base de datos de SQLite interna, concretamente a la tabla producto. Si solamente se modifica, busca el producto en la base de datos y sustituye el precio y el número por el nuevo elegido.

4.3.1.5 Compartir un producto

- Introducción: Esta acción solo puede realizarse cuando el usuario se ha identificado en la aplicación. Con esta opción se pueden compartir las listas de la compra con sus amigos, familiares, etc.
- Entrada: Botón de compartir en la barra superior de la pantalla compartir el product.
- Proceso: El usuario accede a la pantalla de compartir la lista, presiona el icono de compartir en la barra superior, selecciona el medio para compartir y pulsa el botón “ok” del diálogo para realizar la acción.
- Salida: Producto compartido.

4.3.1.6 Consultar el precio total

- Introducción: Con esta opción consultamos el precio total de la compra.
- Entrada: Se puede observar el precio total al seleccionar el número de productos a comprar.
- Proceso: El usuario accede a la pantalla de futuros eventos y presiona el botón para consultar la actividad, y así visualizer el precio por unidad.
- Salida: La aplicación suma todos los participantes añadidos en la lista y nos indica el total de nuestra compra.

4.3.2 Requisitos de interfaz

Con respecto a los requisitos de interfaz podemos diferenciar tres tipos: interfaz de usuario, interfaz software e interfaz hardware.

En cuanto a la interfaz de usuario, el principal objetivo es conseguir una interfaz simple y sencilla de manejar. Puesto que la aplicación está destinada a dispositivos móviles y con diversas funciones, no existe un patrón básico que compartan las distintas interfaces. Aún así, debido al carácter de la aplicación y su funcionalidad orientada al uso de compra, la interfaz más común es la empleada por las funciones de escoger productos y precios, la cual consiste en una lista de productos junto con sus precios.

En referencia a la interfaz software, el proyecto se desarrolla empleando el sistema operativo Windows 10, con el entorno de desarrollo Android Studio y la plataforma Firebase como servidor. Java es el lenguaje principal puesto que Android hace uso de dicho lenguaje.

Como requisito o interfaz hardware es importante mencionar la necesidad de emplear un dispositivo móvil con el sistema operativo Android y conexión a Internet, además de un servidor de base de datos propocionada por Google, Firebase.

4.3.3 Restricciones de diseño

A diferencia de una página web en la que sería adecuado seguir los estándares marcados por el W3C, a la hora de desarrollar una aplicación móvil no existen unos estándares básicos que seguir. Aún así, se han seguido en la medida de lo posible una serie de mejores prácticas sugeridas por la guía de desarrolladores Android ofrecida en su página web, mediante la cual se ofrece información por ejemplo, de qué hacer y cómo realizarlo para adaptar la interfaz a distintos tamaños de pantalla.

4.3.4 Requisitos no funcionales

Los requisitos no funcionales para la aplicación, es decir, los que no especifican el comportamiento del sistema, son:

- **Rendimiento**: La aplicación debe desempeñar su función de una manera fluida. Se debe buscar la experiencia de uso más agradable para el usuario.

- **Disponibilidad**: La aplicación debe estar disponible en la tienda Play Store. Debe funcionar sin conexión a Internet, ya que los alumnos que vienen a estudiar no tienen conexión inmediata.

- **Accesibilidad**: La aplicación debe ser legible y tiene que seguir los patrones de accesibilidad de Google.

- **Usabilidad**: Cualquier alumno extranjero debe ser capaz de utilizar la aplicación y

acceder a toda la funcionalidad sin ningún tipo de restricción.

- **Estabilidad**: La aplicación debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma y avisando a este de la naturalidad del error.

- **Mantenimiento**: La aplicación debe ser mantenida y actualizada, dando posibilidad a mejorar el rendimiento y la usabilidad en cualquier momento.

- **Interfaz**: Clara y concisa. No debe dar lugar a la confusión del usuario y debe seguir los estándares de diseño de interfaces de Google.

- **Integración**: La aplicación debe de integrarse con todo el sistema operativo de Android. Hacer uso de las aplicaciones nativas si se necesita y mantener un diseño acorde al sistema.

- **Optimización**: El consumo de batería y de datos debe ser adecuado, y nunca dejar procesos sueltos que consuman memoria y batería. El tiempo de ejecución debe ser mínimo, para mejorar los tiempos de respuesta y la experiencia de uso del usuario.

4.3.5 Otros requisitos

Dentro de los requisitos para la aplicación, es esencial que la aplicación funcione en la mayoría de dispositivos Android posibles pero debido al gran avance de la tecnología muchos de los recursos utilizados en la aplicación necesitan una alta versión de Android para mostrarse correctamente tanto en diseño como en implementación, en la actualidad esto no es un problema, ya que la mayoria de dispositivos Android poseen las últimas actualizaciones, permitiendo así el correcto funcionamiento.

5 Análisis

En este capítulo se va a proceder a realizar la especificación mediante los casos de uso, una técnica que permite expresar de forma simplificada el comportamiento de un sistema ante la interacción de los usuarios. Su utilización está indicada especialmente para sistemas interactivos debido a que reflejan las acciones de un usuario cuando hace uso de una funcionalidad en la aplicación.

5.1 Casos de Uso

Son una descripción de los pasos que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participan en un caso de uso se denominan actores. Un caso de uso es una secuencia de interacciones que se desarrollará entre un sistema y sus actores, en

respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema.

5.1.1 Tipos de relaciones

- <<communicates>>: Relación de asociación entre un actor y un caso de uso que denota la participación del actor en dicho caso de uso. En la práctica se suele omitir.

- <<include>>: Relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de un escenario en otro.

- <<extends>>: Relación de dependencia entre dos casos de uso que denota que un caso de uso es una especialización de otro.

Se utiliza una relación de tipo <<extends>> entre casos de uso cuando nos encontramos con un caso de uso similar a otro pero que hace algo más que éste. Por contra, utilizaremos una relación tipo <<include>> cuando nos encontramos con una parte de comportamiento similar en dos casos de uso y no queremos repetir la descripción de dicho comportamiento común.

5.1.2 Ventajas

Como técnica de extracción de requerimiento permite que el analista se centre en las necesidades del usuario, qué espera éste lograr al utilizar el sistema. Aunque comúnmente se asocian a la fase de Test de una aplicación, esta idea es errónea, y su uso se extiende mayormente a las primeras fases de un desarrollo.

De acuerdo al presente proyecto nos encontramos **un único actor**, el usuario potencial de la aplicación.

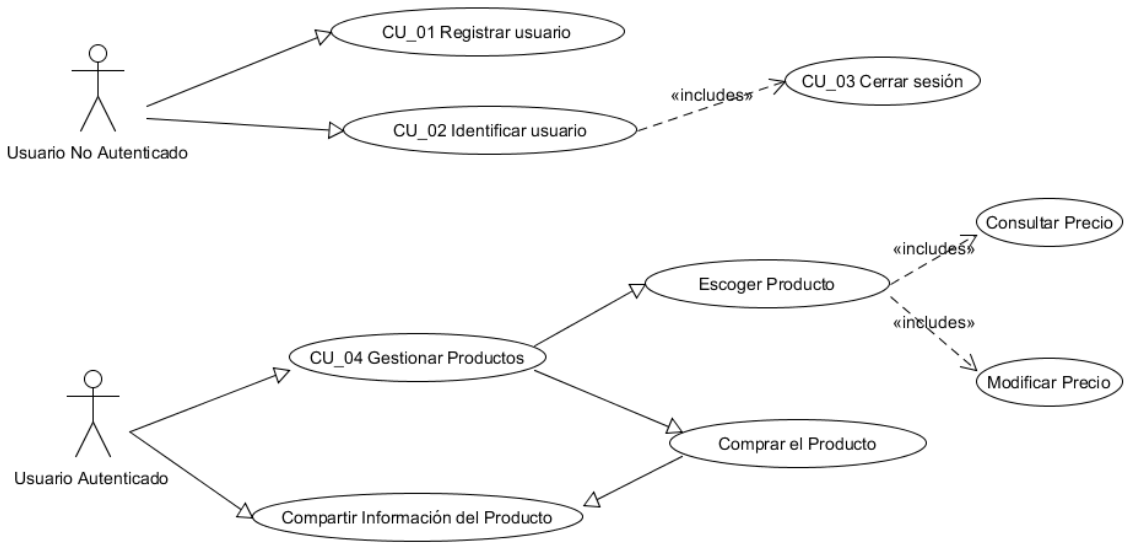

Figura 5.1.2: Diagrama de Caso de Uso

5.1.3 Base de Datos: Modelo Entidad – Relación

Se ha diseño mediante UML el modelo Entidad-Relación, de la base de datos, el cual representa de forma visual una abstracción de la tabla que formará nuestra base de datos.

Figura 5.1.3 Modelo E- R

erasmusvipvalencia-2e126-export.json


```
"users":{
"8etBLLCFf9fGeGVkC2ZhlXgBB6S2"
{
  "tickets" : {
    "2341341" : {
      "direction" : {
        "lat" : "39.47",
        "long" : "-0.34"
      },
      "info" : "Granada, Malaga, Sevilla y Córdoba",
      "name" : "Andalucia Trip",
      "price" : "11",
      "time" : {
        "day" : "11/08/2017",
        "hour" : "23:30 - 05:40"
      },
      "url" : "http://www.pubyfiesta.com/modules/event/getImage.php?params=0000359508.jpg&w=1000&h=800"
    },
    "134134123" : {
      "direction" : {
        "lat" : "42.6929641",
        "long" : "-1.7960056"
      },
      "info" : "Un día diferente con Erasmus Vip",
      "name" : "Viaje a la Nieve",
      "price" : "40",
      "time" : {
        "day" : "25/12/2017",
        "hour" : "10:00 - 22:00"
      },
      "url" : "http://www.erasmusvipvalencia.com/wp-content/uploads/2016/03/nochevieja.jpg"
    }
  }
```


Figura 5.3.1.1 Base de datos Firebase

6 Diseño

La etapa de diseño permite describir como el sistema va a satisfacer los requisitos. Esta etapa a menudo tiene diferentes niveles de detalle. Los niveles más altos de detalle, generalmente describen los componentes o módulos que formarán el software a ser producido. Los niveles más bajos describen con mucho detalle cada módulo que contendrá el sistema.

Es importante anotar que la fase de diseño no solo aporta a la definición del proyecto las especificaciones sobre los requisitos funcionales, sino que también provee las especificaciones sobre distintas áreas, tales como la escalabilidad, disponibilidad, portabilidad, flexibilidad, etc.

La arquitectura de la aplicación es una arquitectura típica cliente-servidor. Por un lado el cliente es la propia aplicación Android y por otro, el servidor se corresponde con una base de datos NoSQL alojada en la nube que ofrece soporte para la persistencia.

En el siguiente apartado se va a describir el software utilizado para desarrollar el proyecto.

6.1 API de Android

Android, puede ser considerado como un framework del lenguaje de programación Java, es decir, un esquema de desarrollo específico creado para la implementación de una aplicación sobre dicho lenguaje. De manera general, la ventaja de utilizar un framework radica en que el desarrollador no tiene que definir una estructura global de la aplicación, sino que tiene que ir añadiendo estructuras y servicios que le ofrece el propio framework. A continuación se van a describir los elementos y estructuras utilizados a la hora de programar en Android.

6.1.1 Android Manifest

Está situado en la raíz de nuestras aplicaciones como AndroidManifest.xml, es un archivo de configuración donde podemos aplicar las configuraciones básicas de nuestra app. Su configuración puede realizarse a través de una interfaz gráfica, pero es recomendable hacerlo desde el propio xml. En él se declaran todas las actividades que se han utilizado en la aplicación y en cada una de ella se declaran valores como el nombre, el estilo y el punto de entrada a la aplicación.

6.1.2 Views y Layouts

La interfaz de usuario se define en los archivos XML del directorio res/layout. Cada pantalla tiene un código XML diferente. Diseñar una pantalla usando Java puede resultar complejo y poco eficiente, sin embargo, Android soporta XML para diseñar pantallas y define elementos personalizados, cada uno representando a una "subclase" especifica de view.

Cada fichero describe un layout y cada layout a su vez puede contener otros elementos. Estos elementos pueden estar formados por view. Un view es un objeto cuya clase es android.view.View. Es una estructura de datos cuyas propiedades contienen los datos de la capa, la información específica del área rectangular de la pantalla y permite establecer el layout. Es útil como clase base para los widgets, que son unas subclases ya implementadas que dibujan los elementos en la pantalla.

Los widgets contienen sus propias medidas, pero puedes usarlas para construir tu interfaz más rápidamente. La lista de widgets que puedes utilizar incluyen Text, EditText, Button, RadioButton, CheckBox, etc. A continuación, se puede observar de forma esquemática en la siguiente figura, una construcción de layouts con sus respectivas views.

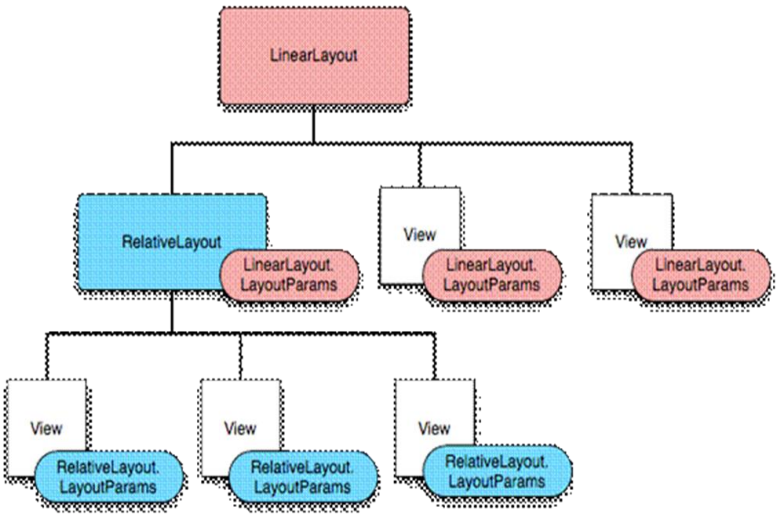

Figura 6.1.2: Views y Layouts

6.1.3 Activities

Se usa el termino de Activity para denominar a un tipo de clases java que heredan de Activity. Una actividad, como su propio nombre indica, es algo que el usuario puede hacer. En Android, una actividad es un conjunto de acciones (tocar la pantalla para apretar un botón, para escribir con el teclado, etc) que son una iteración directa con el usuario y que afectan a una parte de la aplicación. También representa la lógica de negocio de una pantalla de la aplicación visualizada gracias a una interfaz gráfica de usuario (layouts).

El método más importante en las activities es OnCreate, en él se inicializan todas las referencias a las views de nuestros layouts, además de añadir la funcionalidad de cada uno de ellos, los eventos de escucha, etc.

6.1.4 Fragments

Un fragmento podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de la interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades. Son como pequeñas actividades contenidas dentro de una actividad anfitriona, manejando su propio diseño y ciclo de vida. Los fragmentos facilitan el despliegue de las aplicaciones en cualquier tipo de tamaño de pantalla y orientación.

Otra ventaja de usarlos es que permiten crear diseños de interfaces de usuario de múltiples vistas ya que los fragmentos son imprescindibles para generar actividades con diseños dinámicos, como por ejemplo el uso de pestañas en un View Pager.

6.1.5 Fragments Pager Adapter

Representa cada vista como un fragmento que se mantiene persistente en el gestor de fragmentos para que el usuario siempre pueda volver a cada página. De esta forma, puede ser muy útil cuando existen varios fragmentos que van a utilizarse de manera similar. En nuestro caso se va a utilizar para cargar las listas en diferentes categorías junto con su nombre y precio total de la compra.

Como se aprecia en la figura 13, cada View Pager tiene asociado un Fragment Pager Adapter con los diferentes fragmentos a mostrar en él. Para que el adaptador funcione correctamente solo necesita utilizar las subclases getItem (int) y getCount (), donde la primera se utiliza para devolver el fragmento asociado a cada posición y la segundo para llevar la cuenta del número de fragmentos disponibles. También se puede utilizar la subclase getPagerTitle (int) para asociar un título a cada fragmento.

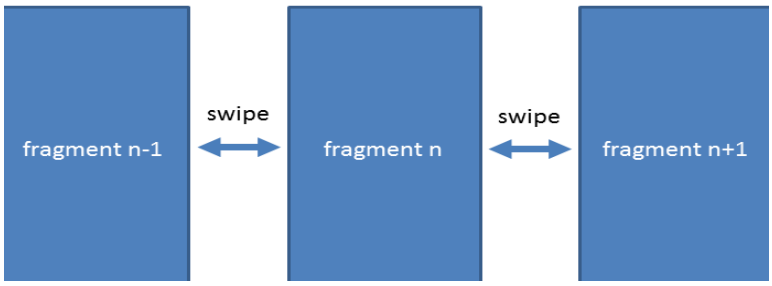

Figura 6.1.5: Fragments Pager Adapter

6.1.6 Intents

Es un mensaje asíncrono que solicita acciones de otros componentes. Básicamente, sirve para invocar componentes, en android entendemos por componentes las Activities, que representan una única pantalla con interfaz de usuario; los Services, que no disponen de interfaz gráfica, y realizan tareas costosas en segundo plano; los ContentProviders, que permiten compartir datos entre aplicaciones; y por último, los BroadcastReceiver que permiten responder a notificaciones del sistema.

Sus métodos principales son:

- startActivity() para lanzar una actividad.
- startService() para lanzar un servicio.
- sendBroadcast() para lanzar una notificación.
- bindService() para comunicar un servicio.

En este proyecto se van a utilizar para lanzar actividades y en algunas ocasiones, a la hora de crearlos, para añadir información adicional mediante la función extras, que utiliza la técnica pares (clave-valor). Con esto se va a conseguir pasar información entre las diferentes

pantallas de la aplicación.

6.1.7 Adapters

Un adaptador es un objeto que comunica a un ListView (lista de objetos) los datos necesarios para crear las filas de la lista. Es decir, conecta la lista con una fuente de información como si se tratase de un adaptador de corriente que alimenta a un televisor. Además de proveer la información, también genera los Views para cada elemento de la lista. Se puede apreciar el funcionamiento de un adapter en la figura 10.

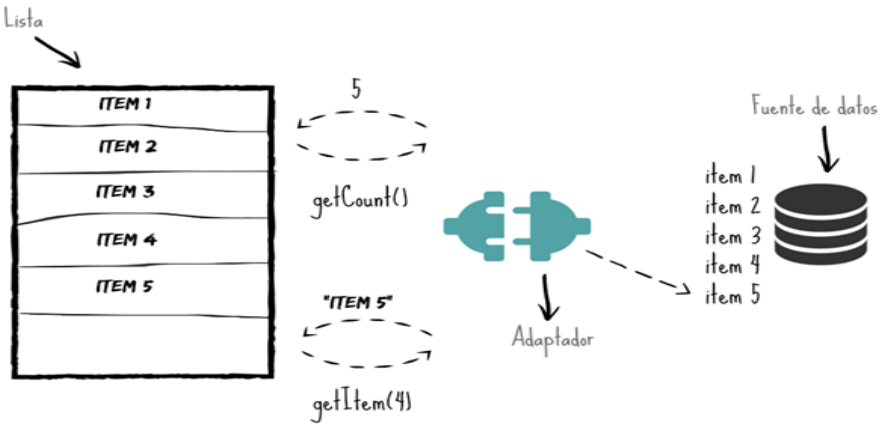

Figura 6.1.7: Gráfica de Adapters

Cuando se relaciona un adaptador a una lista, inmediatamente comienza un proceso de comunicación interno para poblar la con una fuente de datos. Dicha comunicación se basa principalmente en los siguientes métodos del adaptador:

- getCount(), el cual devuelve la cantidad de elementos que hay en la fuente de datos. Con este valor la lista ya puede establecer un límite para añadir items.

- getItem(), con el que se obtiene un elemento de la fuente de datos asignada al adaptador en una posición establecida. Normalmente la fuente de datos es una lista de objetos.

Aunque estos métodos no son los únicos que existen para establecer la relación, son los más significativos para entender el concepto de un adaptador.

6.1.8 Action Bar y Toolbar

Action Bar es la barra de título y herramientas que aparece en la parte superior de muchas de las aplicaciones actuales. Normalmente muestra un icono, el título de la actividad en la que nos encontramos, una serie de botones de acción, y un menú desplegable (menú de overflow)

donde se incluyen más acciones que no tienen espacio para mostrarse como botón o simplemente no se quieren mostrar como tal.

Puede implementarse de dos formas:

- Haciendo uso de la funcionalidad básica incluida por defecto en las actividades al utilizar uno de los temas de la librería de soporte y extenderlas de AppCompatActivity (librería de Android).
- Utilizar el nuevo componente Toolbar proporcionado por la librería appcompat. De esta forma se puede incluir de forma explícita la action bar en los layouts XML como si fuera cualquier otro control, y no sólo en la parte superior de la pantalla a modo de app bar, sino también en cualquier otro lugar de la aplicación donde se quiera utilizar esta funcionalidad de barra de acciones.

6.1.9 Navigation Drawer

Es un menú lateral deslizante, el cual aparece en muchas aplicaciones al deslizar el dedo desde el borde izquierdo de la pantalla hacia el lado opuesto (también puede aparecer en el lado derecho, pero es menos frecuente).

Para añadir el navigation drawer a una actividad hay que hacer que el elemento raíz del layout XML sea del tipo <android.support.v4.widget.DrawerLayout>. Y dentro de este elemento colocar únicamente 2 componentes principales: el layout real de la actividad y el layout del menú lateral, que entre otras cosas hace de contenedor de las distintas opciones del menú lateral.

El primero de estos elementos se añade en forma de <include>. Para el segundo se va a utilizar otro de los nuevos componentes incluidos con la nueva librería de diseño de Android (Design Support Library). El componente en cuestión es el llamado NavigationView, que nos ayuda bastante en la construcción del layout del menú lateral.

6.1.10 View Pager

Está constituida por una serie de páginas que contienen los elementos que se quieren mostrar. Para mostrar cada una de las páginas se van pasando hacia adelante o atrás con nuestro dedo, mostrando una animación cada vez que se pasa de una página a otra. Para usar la vista ViewPager hay que tener instalada la librería de soporte de Android (Support Library). También es necesario que tenga un Fragment Pager Adapter asociado para manejar los fragmentos correctamente como se puede observar en la figura 11.

6.1.11 Mensajes Toast

Son muy útiles para ofrecer información extra en forma de eventos o sucesos específicos. Tienen un tiempo determinado de duración que cuando expira desaparecen automáticamente. Estos mensajes se han utilizado para indicarle al usuario información sobre los siguientes

eventos:

- Errores de autenticación de los usuarios.
- Cuando se añaden productos y precios en las listas.
- Al cargar y compartir los productos.

Estos mensajes ayudan a que el usuario no se sienta perdido cuando ocurre algún error y le dan seguridad cuando se introducen datos correctamente. Este tipo de mensajes realimentan la comunicación entre la aplicación y los usuarios, además son sutiles, ocupan poco espacio y duran pocos segundos, con lo que a los usuarios avanzados no les incomoda. En la figura 16 se encuentra un ejemplo de un mensaje Toast que se utiliza en la pantalla de login.

6.2 Interfaz Gráfica de usuario

También conocida como GUI (graphical user interface), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo.

Uno de los aspectos a cuidar durante todo el desarrollo ha sido el diseño gráfico de la aplicación ya que todas las imágenes se han escalado a los diferentes tamaños de pantalla que hay en el mercado. Según las pulgadas de cada pantalla, los elementos visuales deben de estar en uno de los siguientes tamaños:

- xxxhdpi: 1280x1920 px.
- xxhdpi: 960x1600 px
- xhdpi: 640x960 px
- hdpi: 480x800 px
- mdpi: 320x480 px

6.2.1 Prototipos

En este apartado se van a describir y mostrar los diseños de los prototipos de la interfaz, los cuales deben facilitar el uso de la aplicación a los usuarios. Los prototipos que se van a mostrar a continuación tienen la finalidad de representar los aspectos interactivos de la aplicación con un cierto nivel de precisión, y así poder evaluar su usabilidad y funcionalidad, pero sin entrar en detalles de implementación.

6.2.1.1 Pantalla de Bienvenida

La pantalla de bienvenida se ha diseñado pensando en el usuario, para que sea sencilla e intuitiva. Puede tanto logearse y registrarse usando el botón de Login / Register. También se puede acceder ó registrarse pulsando el botón de Facebook.

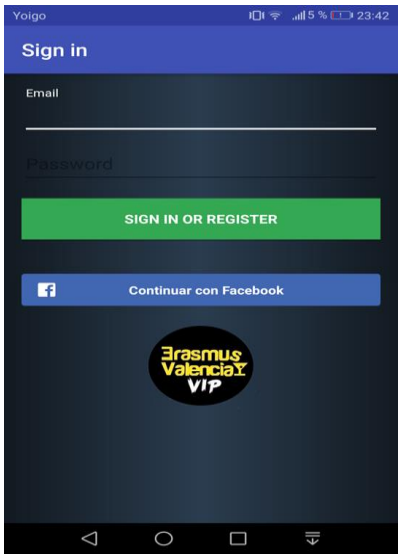

Figura 6.2.1.1: Pantalla de Bienvenida

6.2.1.2 Pantalla Principal

Esta pantalla se ha diseñado básicamente para que el usuario pueda escoger y seleccionar la actividad que más le interese. También actúa como punto de entrada a varias pantallas de la aplicación, gracias al Menú inferior.

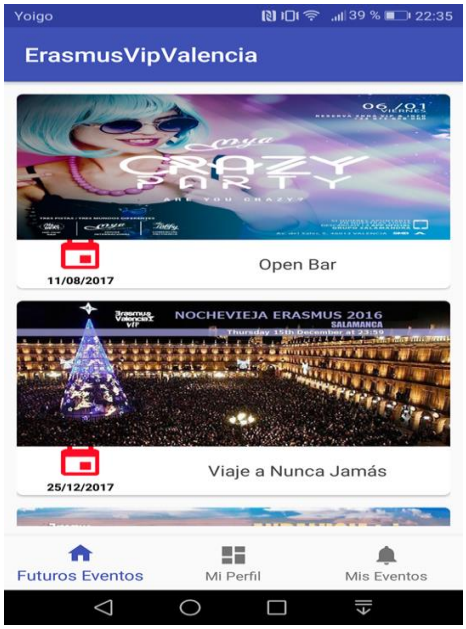

Figura 6.2.1.2: Pantalla de Futuros Eventos

6.2.1.3 Pantalla Comprar Producto

Esta pantalla está formada por un conjunto de botones que se utilizarán para seleccionar la cantidad de tickets a comprar con su respective información de fecha y nombre de la Actividad.

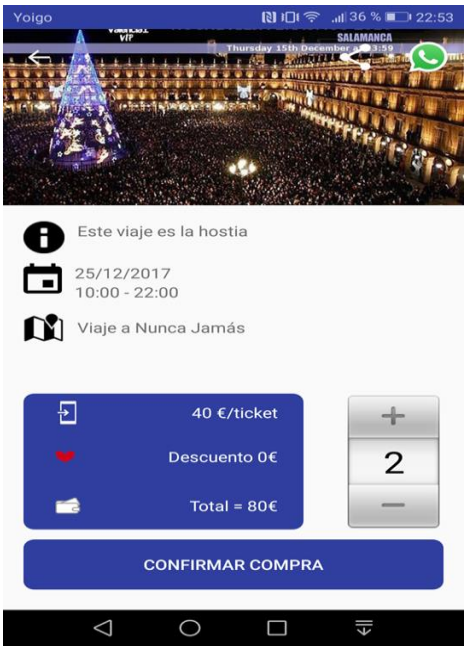

Figura 6.2.1.3: Pantalla de Compra Prouducto

6.2.1.4 Pantalla de Compatir Productos

Esta pantalla se ha diseñado para que el usuario pueda compartir los productos con sus amigos y familiares.

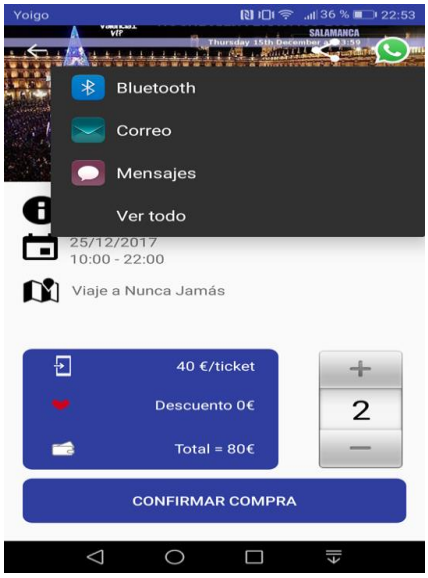

Figura 6.2.1.4: Pantalla de Compartir Productos

6.2.1.5 Pantalla Productos Comprados

En esta pantalla podremos comprobar los productos comprados por el usuario.

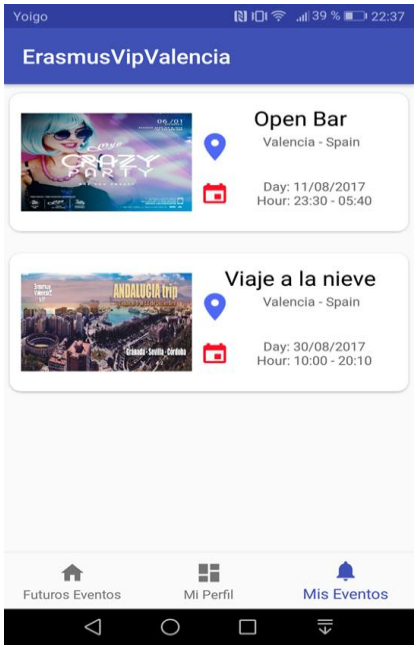

Figura 6.2.1.5: Pantalla de Productos Comprados

6.2.1.6 Pantalla de Mi Perfil

La pantalla de mi perfil, se ha diseñado para mantener información del actual usuario que está conectado en la aplicación. Además que permite modificar sus datos personales.

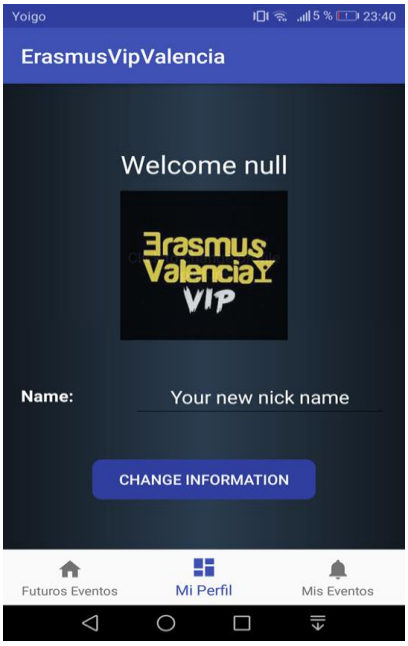

Figura 6.2.1.6: Pantalla de Mi perfil

6.3 Arquitectura Cliente – Servidor

Para que sea posible el acceso de múltiples usuarios a una información común es necesario usar una arquitectura cliente/servidor.

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (servidor) que le da respuesta.

Aplicado al presente caso de estudio, nuestra aplicación Android haría la función de cliente haciendo peticiones a nuestro servidor para obtener o modificar datos communes que sería nuestra base de datos Firebase, que posteriormente pueden ser requeridos por otras aplicaciones clientes.

Cabe destacar también que es una **arquitectura de 3 capas**, quiere decir que es un conjunto de subsistemas cada uno de los cuales depende del que se encuentra en la capa inferior y proporciona los cimientos del que se encuentra inmediatamente por encima de él.

La arquitectura de tres capas es de las más habituales en sistemas informáticos y podemos distinguir los siguientes niveles:

- **Presentación**: Hace referencia a la parte visual que se utilizará para mostrar la información y los datos. Normalmente, es la parte con la que el usuario final interaccionará.

- **Negocio o Lógica**: Es la capa intermedia encargada de interactuar con la interfaz y los datos. Su función es recopilar los datos y procesarlos para que se muestren o almacenen según se desee.

- **Persistencia**: Es la capa de datos. Se encarga de las tareas que realizamos habitualmente con los datos: insertar, modificar, consultar o borrar. Por norma general, la información persiste en esta capa.

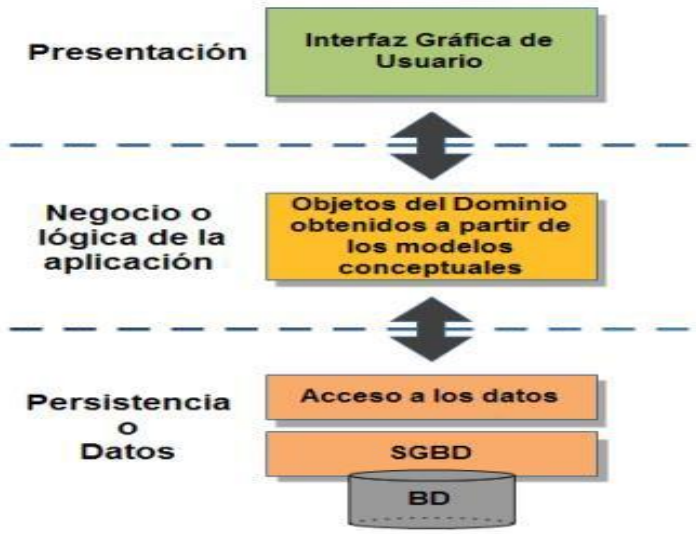

Figura 6.3: Figura Arquitectura 3 capas

6.3 Pruebas de Usabilidad

Se entiende por usabilidad la cualidad que posee la página web o la aplicación móvil que son sencillos de usar porque facilitan la lectura de los textos y presentan funciones y menús sencillos, por lo que el usuario encuentra satisfechas sus consultas y cómodo su uso.

Para analizar la usabilidad de los diseños mostrados anteriormente, se ha analizado si cumplen con los principios de usabilidad de Jakob Nielsen.

6.3.1 Visibilidad del estado del sistema

Este principio de usabilidad indica que siempre se debe tener informado al usuario de lo que está pasando en la aplicación y ofrecerle una respuesta en el menor tiempo posible.

Para cumplir este principio se ha añadido un mensaje tras la compra satisfactoria de un producto.

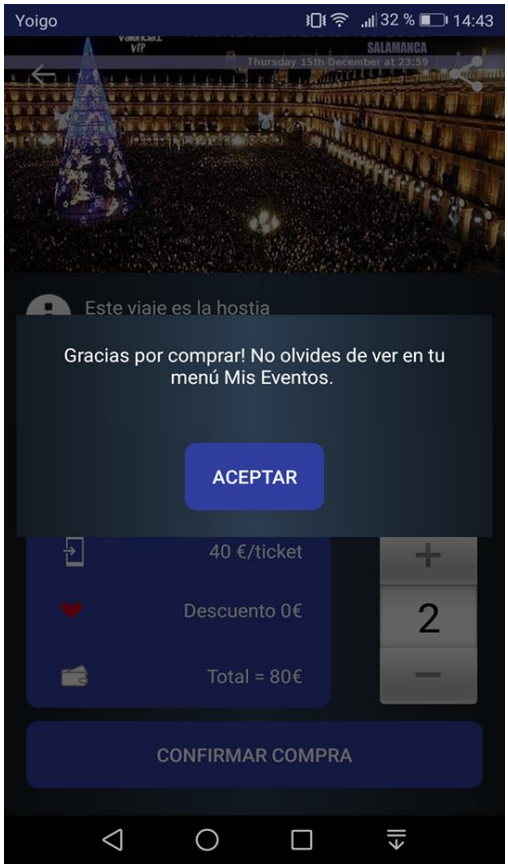

Figura 6.3.1: Mensaje de Compra Satisfactoria

6.3.2 Relación entre el mundo real y el sistema

La información tiene que mostrarse con un orden lógico y las imágenes o iconos usados tienen que ser claros, sin darle la posibilidad al usuario de equivocarse.

En este caso se han colocado iconos representativos en cada apartado del menú, seguido del nombre identificativo.

6.3.3 Control y libertad del usuario

Darle al usuario la posibilidad de subsanar un error y no sentirse frustrado por no poder realizar algo.

Se implementará en la aplicación un apartado para dar la posibilidad de editar un perfil personal.

6.3.4 Consistencia y estándares

Presentar un diseño de manera consistente, es decir, que la información que es similar aparezca siempre de la misma manera (mismas palabras, iconos y posición en la pantalla) y que la información que es diferente se exprese siempre diferente.

Se ha colocado el botón de menú de navegación en la esquina superior izquierda y se mantiene en esa posición en toda la aplicación, además siempre se mantiene el mismo estilo y forma de menú y cabecera.

6.3.5 Prevención de errores

Prevenir cualquier error que pueda cometer el usuario. Y dado el caso de que este cometa uno, tenemos que poner a su alcance todas las opciones posibles para poder corregirlo.

Dado esto se realizarán comprobaciones de campos en tiempo real, de esta forma se evita el error al escribir mal el email.

6.3.6 Reconocer antes que recordar

Siempre es mejor reconocer que obligar al usuario a memorizar acciones u objetos para que pueda cumplir su objetivo.

Todos los botones y menús de la aplicación cumplen los estándares de iconos y textos para que sean familiares a los usuarios.

6.3.7 Flexibilidad y eficiencia de uso

Disponer de una aplicación preparada para todo tipo de usuarios, desde los más experimentados hasta los más novatos. Si cualquiera navega por la aplicación se consigue flexibilidad y si además dispone de opciones avanzadas se obtiene eficiencia.

Cualquier usuario puede comprar su producto, sin necesidad de tener un alto nivel de usar aplicaciones.

6.3.8 Diseño estético y minimalista

Las aplicaciones no deben contener información innecesaria, distrae al usuario y puede llegar a molestar en la navegación. Elimina todo lo que consideres innecesario y que no aporta nada a lo que quieres decir.

El diseño utilizado para la aplicación es minimalista y se muestra solo y exclusivamente la información necesaria.

6.3.9 Ayuda a los usuarios a corregir errores

Intentar que todos los errores que puedan ocurrir en la aplicación estén expresados en un lenguaje entendible por todos, no solo por códigos. De esta forma se indica al usuario qué es lo que pasa en ese momento y que tiene que hacer para salir de ahí.

Cada error que se produzca en la aplicación mostrará una alerta con la descripción del problema.

7 Implementación

Las aplicaciones creadas con Android tienen un archivo común denominado «AndroidManisfest.xml» el cual proporciona información acerca de la aplicación y especifica los parámetros que afectan a cómo funciona.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="uk.co.xakra.erasmusvipvalencia">

    <!-- To auto-complete the email text field in the login form with the user's emails -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.READ_PROFILE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/logoponer"
        android:label="@string/app_name"
        android:roundIcon="@drawable/logoponer"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <meta-data android:name="com.facebook.sdk.ApplicationId"
        android:value="@string/facebook_app_id"/>
        <data android:scheme="@string/fb_login_protocol_scheme" />

        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="uk.co.xakra.erasmusvipvalencia.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/files" />
```

A continuación describiré algunos de los métodos que he utilizado para la fase de integración:

7.1 Login

En este apartado se podrá observar cuales han sido las dos métodos que se utilizan cuando un usuario se identifica en la aplicación.

7.1.1 Email y Password

En este punto se va puede observarr el método que se utiliza en la pantalla de login con email y contraseña cuando nos identificamos en la aplicación. Primero, hay que declarar las variables encargadas de la conexión con Firebase y de permitir la comunicación entre nuestra aplicación y el servidor de Google.

```
3     private void attemptLogin() {
4         if (mAuthTask != null) {
5             // IT SHOULD JUST CALL MAIN ACTIVITY
6             return;
7         }
8
9         // Reset errors.
10        mEmailView.setError(null);
11        mPasswordView.setError(null);
12
13        // Store values at the time of the login attempt.
14        String email = mEmailView.getText().toString();
15        String password = mPasswordView.getText().toString();
16
17        boolean cancel = false;
18        View focusView = null;
19
20        // Check for a valid password, if the user entered one.
21        if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
22            mPasswordView.setError(getString(R.string.error_invalid_password));
23            focusView = mPasswordView;
24            cancel = true;
25        }
26
27        // Check for a valid email address.
28        if (TextUtils.isEmpty(email)) {
29            mEmailView.setError(getString(R.string.error_field_required));
30            focusView = mEmailView;
31            cancel = true;
32        } else if (!isEmailValid(email)) {
33            mEmailView.setError(getString(R.string.error_invalid_email));
34            focusView = mEmailView;
35            cancel = true;
36        }
```

7.1.2 Facebook

En este punto se observa el método que se utiliza en la pantalla de login cuando nos identificamos en la aplicación a través de Facebook. También se puede apreciar el método que se encargada de la conexión con Firebase y de permitir la comunicación entre nuestra aplicación y el servidor de Google.

```java
public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        // Pass the activity result back to the Facebook SDK
        mCallbackManager.onActivityResult(requestCode, resultCode, data);

    }

    @Override
    public void onStop() {
        super.onStop();


    }

    private void handleFacebookAccessToken(AccessToken token) {
        Log.d("FACEBOOK", "handleFacebookAccessToken:" + token);

        AuthCredential credential = FacebookAuthProvider.getCredential(token.getToken());
        DataService.getInstance().getmAuth().signInWithCredential(credential)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            // Sign in success, update UI with the signed-in user's information
                            Log.d("FACEBOOK", "signInWithCredential:success");
                            FirebaseUser user =  DataService.getInstance().getmAuth().getCurrentUser();
                            updateUI(user);
                        } else {
                            // If sign in fails, display a message to the user.
                            Log.w("FACEBOOK", "signInWithCredential:failure", task.getException());
                            Toast.makeText(LoginActivity.this, "Authentication failed.",
                                    Toast.LENGTH_SHORT).show();
                            updateUI(null);
                        }
                    }
                });
    }
```

7.2 Fragments

En este apartado se podrá observar cuales han sido los Fragments que se utilizan para crear las diferentes pantallas de la aplicación.

```java
private BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelectedListener
            = new BottomNavigationView.OnNavigationItemSelectedListener() {

        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {

            Fragment newFragment = null;

            switch (item.getItemId()) {
                case R.id.navigation_home:

                    newFragment = new FutureEvents();
                    break;

                case R.id.navigation_dashboard:

                    newFragment = new MyProfile();
                    break;

                case R.id.navigation_notifications:

                    newFragment = new MyEvents();
                    break;

            }

            final FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
            transaction.replace(R.id.content, newFragment);
            transaction.addToBackStack(null);
            transaction.commit();
            return true;
        }

    };
```

7.2.1 Futuros Eventos

En este código se instancia al método padre “Fragments” para desarrollar la página **Futuros Eventos**.

Se hereda todas las propiedades de Fragments con la palabra reservada “extends”

```
    public FutureEvents() {
        // Required empty public constructor
    }

    public static FutureEvents newInstance() {
        FutureEvents fragment = new FutureEvents();
        Bundle args = new Bundle();

        return fragment;
    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View viewFragment = inflater.inflate(R.layout.fragment_future_events, container, false);

        RecyclerView recyclerView = (RecyclerView) viewFragment.findViewById(R.id.reciclerView);
        recyclerView.setHasFixedSize(true);
        //          WE SET ORIENTATION OF THE RECYCLERVIEW AS VERTICAL
        LinearLayoutManager layoutManager = new LinearLayoutManager(getContext());
        layoutManager.setOrientation(LinearLayout.VERTICAL);
        recyclerView.setLayoutManager(layoutManager);
        //          WE ADD SOME SEPARATION AMONG THE TICKETS
        recyclerView.addItemDecoration(new SpaceItems(30));
        //          GET THE TICKETS TO PUT IN THE ADAPTER
        final Tickets tickets [];
        tickets = DataService.getInstance().getTickets();
        //          CREATE AN ADAPTER
        TicketsAdapter ticketsAdapter = new TicketsAdapter(tickets);
        recyclerView.setAdapter(ticketsAdapter);
        return viewFragment;
    }
```

7.2.2 Mis Eventos

En este código se instancia al método padre “Fragments” para desarrollar la página **Mis Eventos**.

Se hereda todas las propiedades de Fragments con la palabra reservada “extends”

```java
public class MyEvents extends Fragment {

    private View viewFragment;
    private OnFragmentInteractionListener mListener;

    public MyEvents() {
        // Required empty public constructor
    }
    public static MyEvents newInstance() {
        MyEvents fragment = new MyEvents();
        return fragment;
    }
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    }
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        viewFragment=inflater.inflate(R.layout.fragment_my_events, container, false);
        RecyclerView recyclerView = (RecyclerView) viewFragment.findViewById(R.id.reciclerView);
        recyclerView.setHasFixedSize(true);
        //          WE SET ORIENTATION OF THE RECYCLERVIEW AS VERTICAL
        LinearLayoutManager layoutManager = new LinearLayoutManager(getContext());
        layoutManager.setOrientation(LinearLayout.VERTICAL);
        recyclerView.setLayoutManager(layoutManager);
        //          WE ADD SOME SEPARATION AMONG THE TICKETS
        recyclerView.addItemDecoration(new SpaceItems(20));
        //          GET THE TICKETS TO PUT IN THE ADAPTER
        DataService dataService = DataService.getInstance();
        final MyTickets tickets [];
        tickets = dataService.getDataBase().getMyTickets();
        //          CREATE AN ADAPTER
        TicketsBoughtAdapter ticketsBoughtAdapter = new TicketsBoughtAdapter(tickets);
        recyclerView.setAdapter(ticketsBoughtAdapter);
```

7.2.3 Mi Perfil

En este código se instancia al método padre “Fragments” para desarrollar la página **Mi Perfil**.

Se hereda todas las propiedades de Fragments con la palabra reservada “extends”

```java
public class MyProfile extends Fragment {


    private EditText newName;
    private TextView nameV;
    private ImageView newImage;
    private ProgressBar progressBar;
    private boolean changedImage;
    private int PICK_IMAGE;
    private Bitmap bm;
    private Button button;
    private String name;
    private static FirebaseAuth mAuth;


    public MyProfile() {
        // Required empty public constructor
    }


    public static MyProfile newInstance() {
        MyProfile fragment = new MyProfile();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        changedImage = false;
        bm = null;
        mAuth = FirebaseAuth.getInstance();
        super.onCreate(savedInstanceState);

    }
```

7.3 Base de Datos

En este apartado se podrá observar la inicialización de nuestra base de datos: Firebase, a través de la configuración de Tickets, para así poder tener los detalles de la compra de los productos comprados por el usuario.

También es de concretar que la Base de Datos, trabaja en tiempo real, por lo cuál los eventos e información se actualizan en un instante si existe alguna modificación por parte del desarrollar, en agregar nuevos eventos o tickets de compra.

```
public class DataBase {

    private String userId;
    private MyTickets tickets[];
    private DatabaseReference fbDataBaseRef;

    public DataBase(String id){

        this.userId = id;
        fbDataBaseRef = FirebaseDatabase.getInstance().getReferenceFromUrl("gs://erasmusvipvalencia-2e126.firebaseio.com");
        getMyTickets();
    }

    public MyTickets[] getMyTickets() {
        fbDataBaseRef.child("users").child(userId).child("tickets").addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                tickets = new MyTickets[(int)dataSnapshot.getChildrenCount()];

                int i= 0;
                for (DataSnapshot election : dataSnapshot.getChildren()) {
                    String img,quantity,day,hour,info,name,id;

                    id = (election.getKey());
                    day = ((String) election.child("time").child("day").getValue());
                    hour = ((String) election.child("time").child("hour").getValue());
                    img = ((String) election.child("img").getValue());
                    quantity = ((String) election.child("quantity").getValue());
                    info = ((String) election.child("info").getValue());
                    name = ((String) election.child("name").getValue());

                    tickets[i]= new MyTickets(img,quantity,day,hour,info,name,id);

                    if (tickets[i] == null) {
                        Log.d("MYDATABASE", "User-GET" + userId + " is unexpectedly null");
```

8 Testeo y Pruebas

Para comprobar que la aplicación cumple los objetivos expuestos en la planificación, se desarrollaron varias pruebas para evaluar el rendimiento en distintos dispositivos móviles. También se realizaron pruebas de satisfacción para saber la opinión de los usuarios.

8.1 Pruebas de Rendimiento

Las primeras pruebas ejecutadas sobre la aplicación corresponden a las de rendimiento. Para que estas pruebas sean más objetivas, se han ejecutado sobre diferentes dispositivos Android, se han utilizado dispositivos de distintas gamas. El HTC One m10 corresponde a una gama alta de móviles Android, el Samsung Galaxy S5 a una gama media-alta y el Huawei P8 a la gama más discreta de los dispositivos Android.

	HTC One m10	Samsung Galaxy S4	Huawei P6
Procesador	Quad-core 1.6GHz	Quad-core 1.4 GHz	Dual Core 1.3 GHz
Memoria RAM	4GB RAM	2GB RAM	1GB RAM
Sistema Operativo	Android 7 Nougat	Android 5 Lollipop	Android 4.4 KitKat
Pantalla	6 pulgadas	4.8 pulgadas	4 pulgadas

Tabla 2. Dispositivos utilizados en las pruebas

Se han realizado diversas pruebas de tiempo de respuesta en los diferentes móviles.

Para esta prueba se ha medido el tiempo de respuesta de distintas funcionalidades de la aplicación. Los resultados se pueden observar en la Tabla 2.

- Por una parte se ha medido el tiempo de cargar de la aplicación, tanto el primer arranque como los demás. Para el primer arranque tenemos que tener en cuenta que la aplicación obtiene los recursos e introduce la información en la base de datos. Esto aumenta el tiempo de espera de arranque, pero una vez completado el primer arranque se disminuyen los tiempos de arranque. Si se mira desde esta perspectiva, aumentando el tiempo necesario para el primer arranque, se disminuye los tiempos posteriores de respuesta.

- La segunda prueba medida es la descarga de todos los eventos del Calendario. Realmente esta prueba depende de la velocidad de internet que se disponga, ya que solo se requiere descargar los eventos la primera vez que accedes a la aplicación. Una vez completado el primer acceso, los eventos son obtenidos de la base de datos, donde se almacenaron en el primer arranque de la aplicación.

- La tercera prueba corresponde al tiempo de acceso a los detalles de los Eventos. Esta funcionalidad, también depende de la velocidad de la conexión de internet. Como se puede aprecia, los tiempos de espera son mínimos y muy rápidos en móviles de gama alta.

- Por último, se ha medido el tiempo en comprar un producto. Esta funcionalidad, al requerir conexión a internet, también depende de la velocidad de la conexión. Como se puede aprecia, los tiempos de espera son mínimos y muy rápidos en móviles de gama alta con una buena conexión de internet.

	HTC One m10	Samsung Galaxy S5	Huawei P6
Inicio y carga (primera vez)	6.2 segundos	6.8 segundos	8.1 segundos
Inicio y carga	1.1 segundos	1.3 segundos	1.5 segundos
Descarga de eventos del calendario	14 segundos	14.2 segundos	16 segundos
Detalles de los Eventos	0.8 segundos	0.9 segundos	1 segundo
Comprar un producto	1.2 segundos	1.1 segundos	1.4 segundos

Tabla 3. Resultados de las pruebas de rendimiento

8.2 Pruebas y Satisfacción de Usuarios

Otro factor a analizar para saber si la aplicación cumple con los requisitos necesarios de cara al cliente es realizar pruebas para determinar el agrado del usuario.

La población utilizada para estas pruebas constaba de un equipo de 6 personas que enviaban comentarios y sugerencias para la aplicación. Para este fin se creó un grupo de Google+ cuya intención era poder probar la beta de la aplicación.

Para este cometido se ha realizado una encuesta a estos usuarios, donde se preguntaban sobre las siguientes cuestiones:

- **La aplicación es fácil de entender para el usuario**

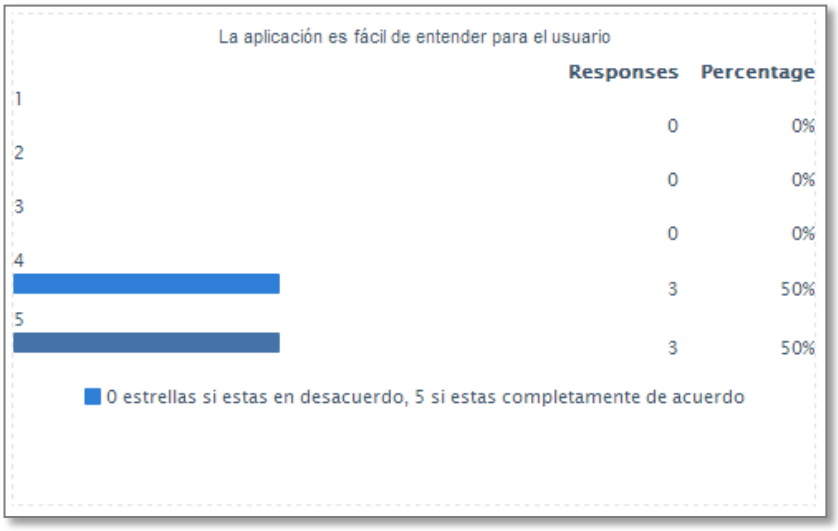

Figura 8.2 Pregunta 1 de las pruebas de aceptación

Como se puede observar la mayoría de los usuarios que probaron la aplicación quedaron satisfechos con la disposición de sus funcionalidades y su facilidad de uso.

- **La interfaz de la aplicación es atractiva y amigable**

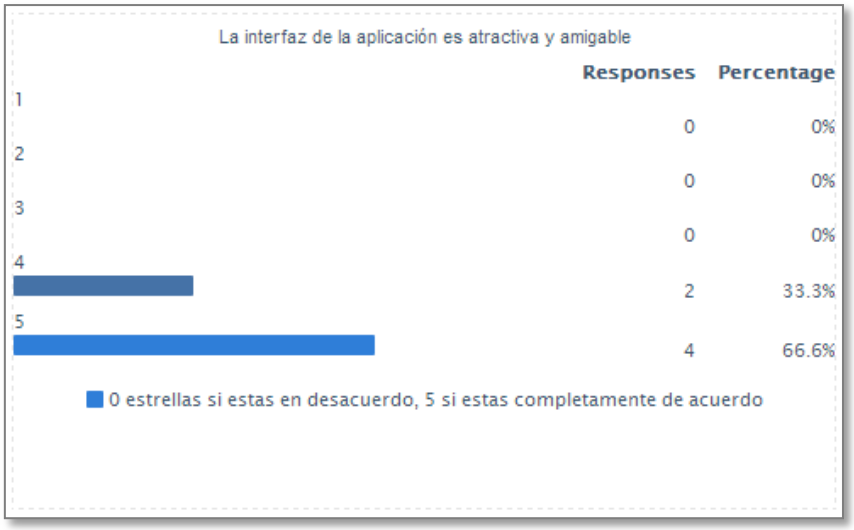

Figura 8.2.1: Pregunta 2 de las pruebas de aceptación

Como se puede observar, los usuarios consideran el diseño de la interfaz de la aplicación atractiva y amigable. Se puede considerar que el patrón de diseño ofrecido en la aplicación ha gustado entre los usuarios ya que la aplicación parece atractiva y fácil de usar.

- **El tiempo de respuesta es correcto bajo ciertas condiciones**

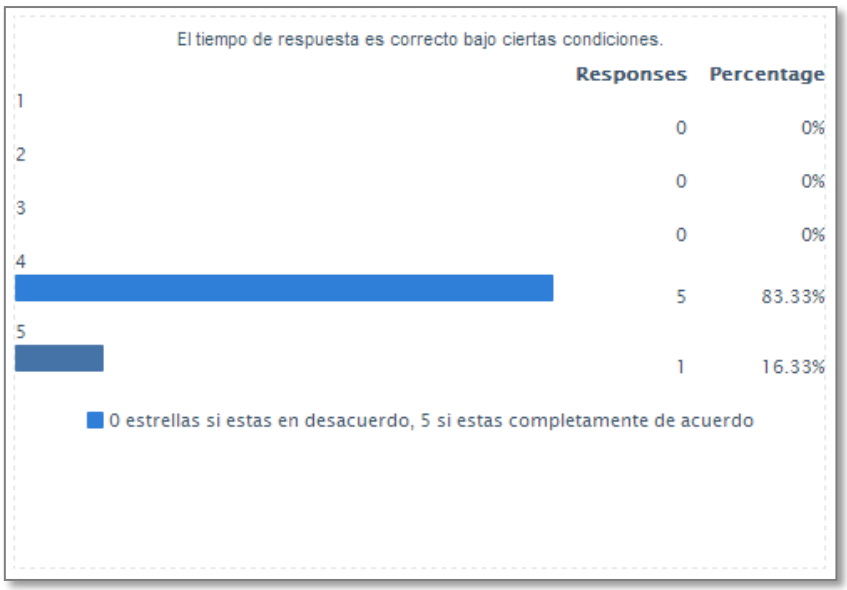

Figura 8.2.2: Pregunta 3 de las pruebas de aceptación

En esta pregunta pudimos comprobar como los usuarios consideraban notable el tiempo de respuesta.

- **En términos generales, la aplicación cumple su cometido y con buen rendimiento**

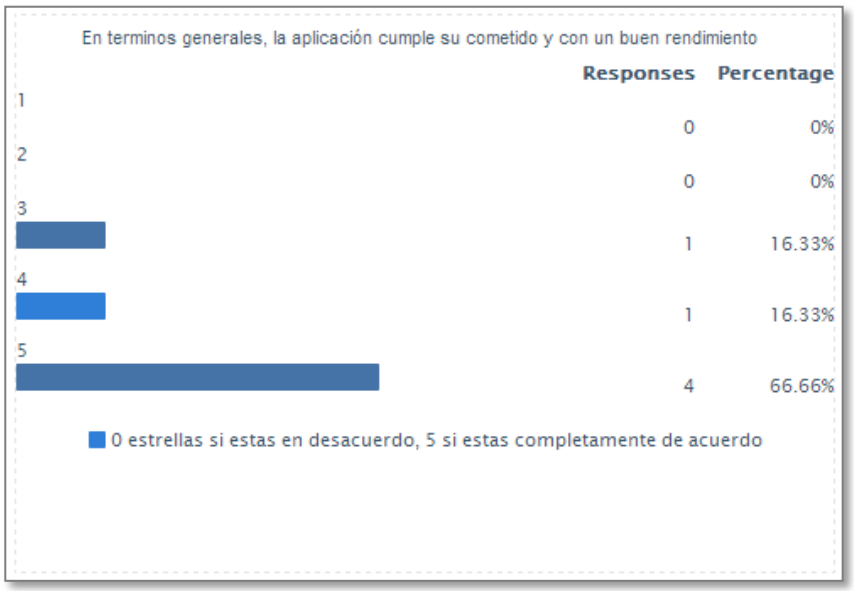

Figura 8.2.3: Pregunta 4 de las pruebas de aceptación

Observando esta pregunta puedo ver que los usuarios están contentos con la funcionalidad obtenida en la aplicación. Puedo ver que algún usuario ha valorado con un 3 el rendimiento de la aplicación. Esto es debido a no tener otro idioma que no sea el español.

Se podría decir que se ha obtenido un rendimiento y una cantidad de funcionalidades que todos los usuarios esperaban. Han valorado positivamente el tiempo de respuesta y el rendimiento por lo tanto podemos concluir que la aplicación cumple con los requisitos no funcionales expuestos en la fase de especificación.

9 Conclusiones

En este último apartado a tratar es el relacionado con las conclusiones del proyecto. En cuanto a los objetivos planteados al comienzo del proyecto, se puede afirmar que se han cumplido la mayoría e incluso se ha avanzado un poco más en el desarrollo, dejando así el camino libre para futuros proyectos o mejoras.

He podido realizar el desarrollo de una aplicación destinada a dispositivos móviles desde cero hasta lograr una aplicación que funciona correctamente y sin errores. De esta forma, he adquirido más conocimientos en Android así como en Firebase. Por otro lado, el hecho de tener que desarrollar la aplicación de forma individual, sin un código inicial o referencia ha hecho que consultar diferentes fuentes tales como libros o páginas web haya cobrado gran relevancia a la hora de abordar y solventar diversos problemas surgidos a lo largo del desarrollo. Sin embargo, el hecho de haber cursado la asignatura de Soluciones Informáticas para Dispositivos Móviles ha hecho partir con una gran base de conocimientos tanto en Android como en el manejo de bases de datos.

Una de las etapas más costosas fue el inicio del desarrollo y la creación de las interfaces, tanto a nivel de diseño como de implementación. Al tener un conocimiento bastante amplio del lenguaje Java, no ha habido graves problemas que no se hayan podido solventar consultando información en la web. Por otro lado, el diseño ha sido una de las etapas más importantes ya que todas las interfaces siguen una temática muy parecida y se ha intentado cuidar al máximo los detalles en todas las pantallas. También hay que mencionar la alta optimización a la hora de emplear los dispositivos virtuales gracias a las continuas actualizaciones de Android Studio, lo que ha hecho que las pruebas hayan sido un proceso rápido y no tan tedioso como habría resultado hace unos años.

También hay que añadir que el proceso más tedioso y que más tiempo he tenido que dedicar ha sido a la plataforma de Firebase, al ser unos servicios que no estaban tan destacados a la hora de construir un backend flexible y seguro, frente a otras tecnologías como pueden ser PHP y MySQL. Sin duda, hay que agradecerle a Google la inmensa cantidad de servicios que nos proporciona cuando hay que implementar un servidor para la aplicación en desarrollo. No solo incluyen clases, métodos y servicios para usar en Android; sino que también proporcionan sus servidores para contener a todos los usuarios que se registren con sus respectivos datos generados al usar la aplicación.

Cuando parecían que todo iba a pedir de boca, surgió el mayor problema de todos: el ordenador portátil donde se estaba desarrollando el proyecto dejó de funcionar (murió la placa base), por lo que tuve que comprarme un ordenador nuevo, pude recuperar los archivos haciendo una copia de disco duro, pero se perdió varios días de trabajo.

Como conclusion, a pesar de las dificultades, mencionar que el proyecto me ha servido para aprender aún más el uso de tecnologías que tienen mucho futuro como es la programación para dispositivos Android, así como para poner en práctica conocimientos adquiridos durante la carrera, como por ejemplo la especificación de

requisitos en un proyecto software y la construcción de casos de uso que asientan los requisitos funcionales.

9.1 Posibles mejoras y amplicaciones

Partiendo de la base en que se ha creado este proyecto se pueden realizar nuevas mejoras para ampliar la funcionalidad, obteniendo un producto mucho más completo, con la finalidad de que pueda cubrir las necesidades de casi cualquier tipo de actividades y eventos.

- Aplicación dinámica: Para que nuestros futuros clientes puedan subir sus eventos y actividades a la aplicación desde un panel de administración independiente dándoles autonomía y flexibilidad.
- Ampliación de las funcionalidades: No solo centarse en el tema de Viajes y Excursiones…sino también brindar opción de Alojamiento, Información de Restaurantes, etc.
- Diseño de la Aplicación: Mejorar todo el diseño visual de la Aplicación.
- Geolocalización de mapa: Para poder encontrar las actividades más interesantes por nuestra zona.
- Amistades: Poder saber que eventos les interesa a nuestros amigos o a cuales asistirán.
- Traducción a diferentes idiomas: esta sería la primera mejora para poder aprovechar para el colectivo Erasmus que no habla español.
- Sistema QR, para descargar los comprobantes de pago de un evento.

10 Bibliografía

1 La Evolución de las Aplicaciones en España. Disponible en: http://www.theappdate.es/blog/informe-sobre-las-apps-en-espana-2015-la-era-appcommerce

2 Marcombo, El Gran Libro de Android, 2ª Edición, 2011

3 Bruce Eckel, Thinking in Java, Edición Online, 1998

4 Android desde cero: https://www.youtube.com/watch?v=L2M8mi7W_fo&list=PLAzlSdU-KYwX0FHqTOEtKU37ALxbUvhky

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

6 Android developers, Fragment. 2013: https://developer.android.com

7 Firebase: https://firebase.google.com

8 Simplified Coding: https://www.simplifiedcoding.net

9 Hermosa Programación: https://www.hermosaprogramacion.com

10 Tutorials Point: https://www.tutorialspoint.com

11 Android Hive: https://www.androidhive.info

12 El Androide Libre: https://www.elandroidelibre.com

13 Adapter Android: https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter.html

14 Facebook login en Android: https://developers.facebook.com/docs/facebook-login/android

15 Facebook login con Firebase: https://firebase.google.com/docs/auth/android/facebook-login

16 Paypal developer: https://developer.paypal.com/developer/accounts