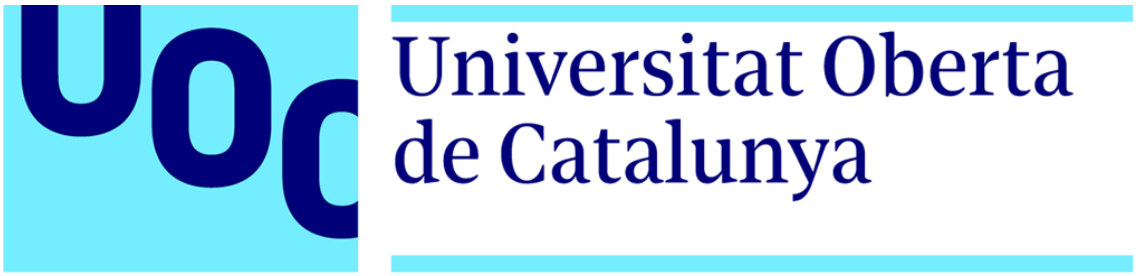


SocialPlan

Nombre Estudiante: Alejandro Resúa García
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Albert Mata Guerra
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

03/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*SocialPlan*
Nombre del autor:	*Alejandro Resúa García*
Nombre del consultor/a:	*Albert Mata Guerra*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Castellano*
Palabras clave	*Android, Red social, Evento*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El sector de la telefonía, es el sector que más ha evolucionado en los últimos años, debido a la aparición de los Smartphones y a las aplicaciones, que permiten personalizarlos al gusto del usuario. Estas aplicaciones pueden adquirirse mediante las tiendas virtuales como la Play Store de Android. Estas características atraen a usuarios y desarrolladores por igual.

Vista la importancia de este sector se decide crear una aplicación para Smartphone.

"*SocialPlan*" surge de la necesidad de que se permita de forma sencilla e intuitiva la publicación de eventos y la creación de grupos para su posterior adhesión de los usuarios a los mismos. De esta forma, asume el rol de red social en la que los usuarios pueden ponerse en contacto para realizar eventos conjuntos y el rol meramente informativo en la que los usuarios se pueden informar de los eventos cercanos a su localidad.

Para el desarrollo de la aplicación se elige la plataforma "Android" ya que a día de hoy es la plataforma que cuenta con más usuarios y el tipo de aplicación necesita la máxima difusión para poder subsistir. Además, se descartan otro tipo de desarrollos debido a las características de geolocalización de la aplicación y que se necesita la mayor ligereza al realizar peticiones contra la base de datos compartida por los usuarios, y desarrollada con Firebase.

En conclusión, la aplicación se puede resumir con la siguiente premisa: hacer posible que personas que quieran realizar un mismo plan o evento puedan organizarlo juntas.

Abstract (in English, 250 words or less):

The telephony sector is the sector that has envolved the most in recent years, due to the appearance of Smartphones. Applications allow you to customize them to the user's needs. These apps can be purchased through virtual stores such as the Android Play Store, Google Play. These features attract users and developers in the same way.

Given the importance of this sector, it is decided to create an application for Smartphone.

"SocialPlan" arises from the need to allow the publication of events in a simple and intuitive way and the creation of groups for their subsequent adhesion of users to them. In this way, it assumes the role of a social network in which users can get in touch to carry out joint events and the merely informative role in which users can report on events close to their location.

For the development of the application, the "Android" platform is chosen because today is the platform with the most users and the type of application needs the most diffusion to be able to survive. In addition, other types of developments are discarded due to the geolocation characteristics of the application and that the greatest lightness is needed when making requests against the database shared by users, and developed with Firebase.

To sum up, the application can be summarized with the following premise: make it possible for people who want to carry out the same plan or event to organize it together.

Índice

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN

El presente documento constituye la memoria del Trabajo Final de Máster (TFM) realizado por Alejandro Resúa García, alumno de la Universitat Oberta de Catalunya (UOC). El TFM desarrollado tiene como título "*SocialPlan*" y ha sido dirigido por Albert Mata Guerra, profesor colaborador de la universidad.

En este capítulo se presentarán los objetivos y el plan de trabajo.

1.1 Contexto y justificación del Trabajo

La aplicación a desarrollar pretende ser una herramienta útil para usuarios que buscan ocio y planes a su gusto cerca de su localización y la posibilidad de recurrir a la aplicación para poder conocer a otras personas con gustos similares y realizar planes creando grupos y adhiriéndose a los mismos.

Además, se parte de la base en que un usuario siempre lleva su Smartphone con él y por consiguiente la red 3G o 4G, por lo que siempre se está en contacto con la aplicación para poder estar informado de los grupos realizados y de los nuevos planes que se puedan adaptar al gusto del usuario.

La idea de desarrollar una aplicación de este tipo surge de la necesidad de que se permita de forma sencilla e intuitiva la publicación de eventos y la creación de grupos para su posterior adhesión por parte de otros usuarios a dichos eventos.

Hoy en día vivimos en un mundo conectado en el que encontrar planes de ocio es bastante sencillo. En ocasiones lo complicado no es encontrar planes si no encontrar las personas con las que realizarlos. Por ello, esta aplicación surge de esa necesidad, de hacer posible que personas que quieran realizar un mismo plan o evento puedan organizarse para realizarlo juntas.

Otra de las necesidades por las que surge esta aplicación es dejar a un lado la publicidad casi vírica que las grandes empresas tienen tanto en las aplicaciones web, como en las aplicaciones móviles. Por ello, los usuarios son los que promocionarán los eventos que hay en una ciudad y tendrá el mismo valor un concierto de gran formato que un micro-teatro en un pequeño local.

Así surge la realización de esta aplicación que tendrá por título "*SocialPlan*".

1.1.1 Análisis de antecedentes

Para poder conocer los requisitos funcionales que debe tener la aplicación se realizará en este apartado un estudio de mercado sobre las aplicaciones similares a la aplicación a desarrollar para definir las similitudes y las diferencias de la aplicación y qué características nuevas se aportarán.

Se han seleccionado y comparado dos aplicaciones móviles diferentes: la aplicación Fever, "*Figura 1: Fever*" y la aplicación Meetup, "*Figura 2: MeetUp*".

Figura 1: Fever

http://www.feverup.com/

Figura 2: Meetup

https://www.meetup.com/es-ES/

Por un lado, Fever es una aplicación que permite descubrir planes en una ciudad en concreto y crear listas personalizadas de planes basados en los gustos de los usuarios.

Por otro lado, Meetup es una aplicación que pone en contacto a personas de una ciudad para que hagan juntas lo que les gusta hacer.

Estas dos aplicaciones serán comparadas con la que se desarrollará en este trabajo: *SocialPlan*, "*Figura 3: SocialPlan"*.

Figura 3: SocialPlan

El estilo de aplicación de *SocialPlan* será similar al que se puede ver en *Fever*. Se propondrán planes cercanos y al gusto de los usuarios. Sin embargo, la principal novedad que propone la aplicación a desarrollar es la creación de los planes y los eventos por parte de los usuarios de la propia aplicación, dando así mucha más participación a los usuarios y teniendo a disposición de ellos eventos multitudinarios y conocidos como eventos alternativos y desconocidos para la gran mayoría.

Además, *SocialPlan* obtiene de *MeetUp* la idea de crear grupos en los que los usuarios puedan adherirse a los mismos. Aunque propone como novedad la creación de grupos cerrados en los que los usuarios puedan decidir el número de personas máximas para que un plan se realice poniendo en contacto a las mismas. De esta forma, se evita la creación de grupos abiertos que solamente contacten a las personas y realiza la función novedosa que propone la aplicación: la función de red social.

Entre las similitudes de las aplicaciones mencionadas, cabe destacar que son aplicaciones multiidioma, por lo que para que *SocialPlan* pueda llegar al máximo número de usuarios, también estará disponible en varios idiomas: inglés y castellano. Aunque esta lista se podrá aumentar a lo largo del tiempo.

1.1.2 Estado del arte

Una red social es la estructura social que dispone de una serie de vínculos que unen tanto a miembros individuales como a colectivos de una sociedad. Trasladando esta teoría al mundo de Internet y en concreto a las aplicaciones móviles, las redes sociales permiten a las personas conectarse a otras personas conocidas o nuevas, de una forma rápida para poder interactuar y crear comunidades sobre un interés común.

Cada red social tiene su objetivo específico. A continuación, se listan las redes sociales más conocidas:

- ***Twitter:*** Sirve para que los usuarios puedan enviar y recibir mensajes de texto de 280 caracteres (tweets) en tiempo real, explicando de forma breve lo que está haciendo justo en el momento en el que ocurre. Además, Twitter fue el encargado de la creación de los hashtags lo que permite a los usuarios a seguir tendencias, hilos de conversación y sucesos que están siendo comentados por varios usuarios.
- ***Facebook:*** Es una red social que permite a los usuarios estar en contacto con sus familiares y amigos, además, permite la presencia de empresas u organizaciones para una mayor comunicación entre éstas y sus seguidores. ¿Qué estás pensando? Es la pregunta que realiza Facebook a sus usuarios para escribir o publicar nuevos contenidos.
- ***Instagram:*** Es una red social orientada a las fotografías realizadas en un momento específico. Sobre todo, está orientada a los smartphones, lo que originó que su uso a nivel mundial haya vuelto popular el fenómeno de los autorretratos o selfies.

1.2 Objetivos del Trabajo

Una vez definido el contexto de la aplicación a desarrollar se listan los objetivos del trabajo indicando los requerimientos funcionales y no funcionales del trabajo.

Funcionales:

- Registro de usuario para poder acceder a la aplicación.
- Identificación de usuario a través de redes sociales.
- Listado de planes adaptados a los gustos de los usuarios y cercanos a su localización.
- Búsqueda de planes.
- Publicación de un nuevo plan o evento informativo para los usuarios.
- Crear grupo para la realización de un plan.
- Adherirse a un grupo creado por otro usuario.
- Comunicación con los usuarios de un grupo.

No funcionales:

- Desarrollo de una aplicación para dispositivos móviles Android.
- Desarrollo de una aplicación Android nativa con las funcionalidades necesarias para el correcto funcionamiento y para el desarrollo interno de la lógica que requiere la aplicación.

- Interfaz intuitiva y amigable para el usuario.
- Aplicación accesible para todos los usuarios.
- Disponibilidad en varios idiomas.
- Aprendizaje de la programación móvil nativa en dispositivos Android y de tecnologías nuevas.
- Aprendizaje de cada una de las tareas involucradas en la creación de una aplicación para dispositivos móviles y de las decisiones tomadas en esta, así como la resolución de los posibles problemas que puedan surgir.

1.3 Enfoque y método seguido

Los dispositivos móviles han evolucionado enormemente en los últimos tiempos, desde PDAs hasta Smartphones, estos son cada vez más asequibles y por tanto el número de usuarios que los utiliza cada vez mayor. Esta evolución ha permitido y provocado el crecimiento exponencial de las aplicaciones desarrolladas específicamente para este sector.

Según la empresa consultora y de investigación de las nuevas tecnologías de la información *Gartner Inc.* en 2016 se vendieron más de 352 millones de dispositivos Android, mientras que Apple vendió tan solo 77 millones de dispositivos.

Por tanto, la idea de desarrollar una aplicación para dispositivos Android se debe al gran número de ventas de estos dispositivos y su amplísima penetración en el mercado. Sin embargo, también se ha tenido en cuenta la opción de realizar la aplicación para usuarios que usen otro tipo de dispositivos como pueden ser los iPhones o los que usen Windows Phone (ésta última se descarta desde el inicio, debido a su desuso).

Por ello y debido a las funcionalidades que ofrece la aplicación como es la generación del listado de los planes y eventos por parte de los usuarios es conveniente que la red de dispositivos sea la mayor posible para que la difusión sea mayor y así poder realizar una aplicación exitosa que pueda ser exportable a otro tipo de plataformas, como los dispositivos iOS.

A continuación, se exponen las ventajas e inconvenientes de realizar una aplicación nativa, híbrida o web.

	NATIVA	HÍBRIDA	WEB
LENGUAJE	Java	HTML, CSS, JS	HTML, CSS, JS
COSTE DESARROLLO	Elevado	Bajo	Bajo
INTERFAZ USUARIO	Superior	Buena	Inconveniente
RENDIMIENTO	Superior	Buena	Inconveniente
MULTIPLATAFORMA	Inconveniente	Superior	Superior
TIEMPO DESARROLLO	Inconveniente	Buena	Superior
APP STORES	Superior	Superior	Inconveniente

Tabla 1: Comparativa de desarrollos

La aplicación nativa a diferencia de las otras dos opciones, está desarrollada y optimizada para un sistema operativo específico y se adapta al 100% con las funcionalidades y características del dispositivo obteniendo una mejor experiencia de uso. Aunque el coste y el tiempo de desarrollo es mucho mayor que para una aplicación híbrida multiplataforma o para una aplicación web. Esto no es un inconveniente ya que como se ha mencionado anteriormente la aplicación será para dispositivos Android.

Además, teniendo en cuenta la necesidad de un back-end externo al del propio dispositivo se requiere un rendimiento óptimo, por lo que se decide finalmente la realización de una aplicación nativa ya que el rendimiento es ligeramente superior al de la híbrida.

Por último, se ha realizado una valoración del uso de APIs externas para la creación de la aplicación. Para ello, se han comprobado las APIs de las aplicaciones estudiadas en "*1.1.1 Análisis de antecedentes*". Debido a las características y los requisitos funcionales de la aplicación, se ha descartado el uso de estas APIs, ya que no se corresponden con los objetivos que se quieren lograr con la creación de este trabajo.

1.4 Planificación del Trabajo

En este apartado se analizan las fases a las que el proyecto se verá sometido, así como el diagrama de Estructura de Descomposición del Trabajo o EDT. De esta forma, se presentará el calendario y el diagrama Gantt de la realización del trabajo.

1.4.1 Fases

A continuación, se especifican y enumeran las distintas fases que se llevarán a cabo en este TFM.

1. **Plan de trabajo:**

En esta primera fase se presentará una buena idea para un trabajo. Se sacarán ideas de trabajos similares y cuando se tenga todo claro se recopilará información que pueda ser útil a lo largo de todo el proceso de realización.

2. **Análisis:**

Se identificarán las características de los usuarios, sus necesidades y objetivos, así como el contexto de uso. Además, se describirán las funcionalidades de la aplicación y se realiza un análisis de los riesgos de la aplicación.

3. **Diseño:**

En esta fase se tendrá que construir el esqueleto de la aplicación y posteriormente tener la capacidad de interpretar la aplicación para su realización. Se realizará construyendo los escenarios de uso, los flujos de interacción, un prototipo, la definición de los casos de uso y la arquitectura de la aplicación.

4. Aprendizaje:

Es el momento de adquirir conocimientos de tecnologías que se usarán en la aplicación. Así como saber utilizar las herramientas elegidas para la elaboración de la aplicación.

5. Implementación:

En esta etapa habrá que llevar a código el diseño realizado en la fase de diseño.

6. Pruebas:

En ella se comprobará que la aplicación cumple con todos los requisitos. También se tratará de encontrar posibles errores que se solventarán, así como una revisión del código para hacerlo más eficiente.

7. Documentación:

En esta etapa es donde se va a ir generando toda la documentación. Según en la fase en la cual nos encontremos habrá que generar un tipo de documentación y unos diagramas o gráficos determinados.

Esta actividad incluye la parte de documentación especificada en cada fase anterior, además se realizará un manual de uso y una presentación de la aplicación.

8. Comercialización:

Una vez desarrollada y probada la aplicación, se distribuirá para ponerla en manos de los usuarios.

1.4.2 Diagrama EDT

Se cree adecuado la elección de un ciclo de vida que produzca resultados intermedios, por ello se ha optado por un ciclo de vida **incremental**.

El objetivo es crear una única iteración para satisfacer los requisitos especificados y las funcionalidades que satisfagan estos requisitos.

Este proceso se repite, hasta que se elabore la aplicación al completo.

Esto ayuda a no ver la aplicación como un todo, sino como pequeñas partes que se van incorporando, permitiendo controlar la calidad del software.

Además, por cada iteración tentemos un prototipo que podemos ver y probar; eso nos dará confianza y mitigará la sensación de incertidumbre.

En el caso de que se produzca un error importante, sólo la última iteración necesitaría ser descartada.

Esto se ve reflejado en el siguiente diagrama de estructura de descomposición del trabajo (EDT). Se quiere reflejar, la iteración en el proceso de construcción del desarrollo-pruebas de la aplicación. Por ello, se repite en dos ocasiones la descomposición de dichas tareas.

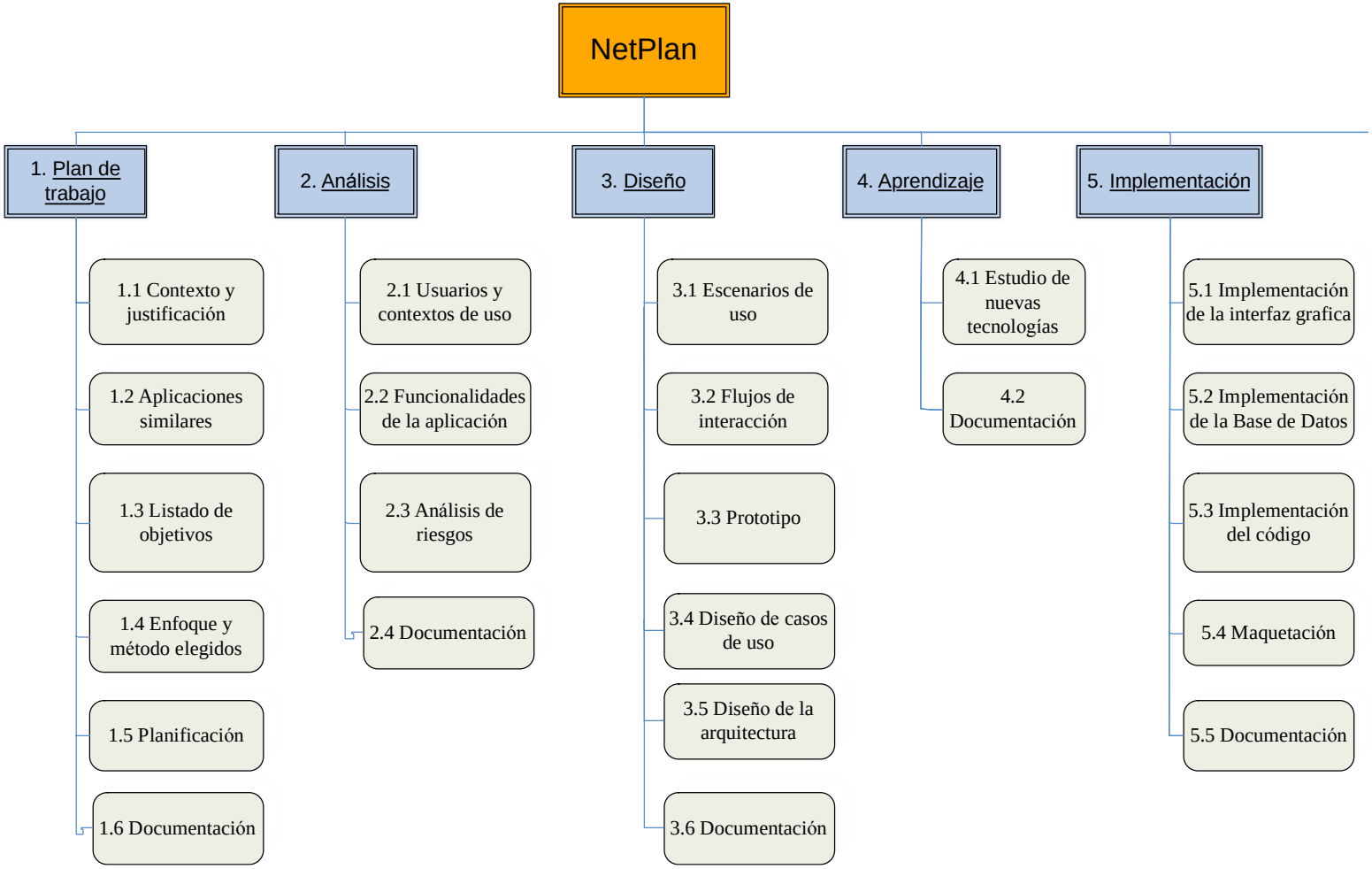


Figura 4: EDT primera parte

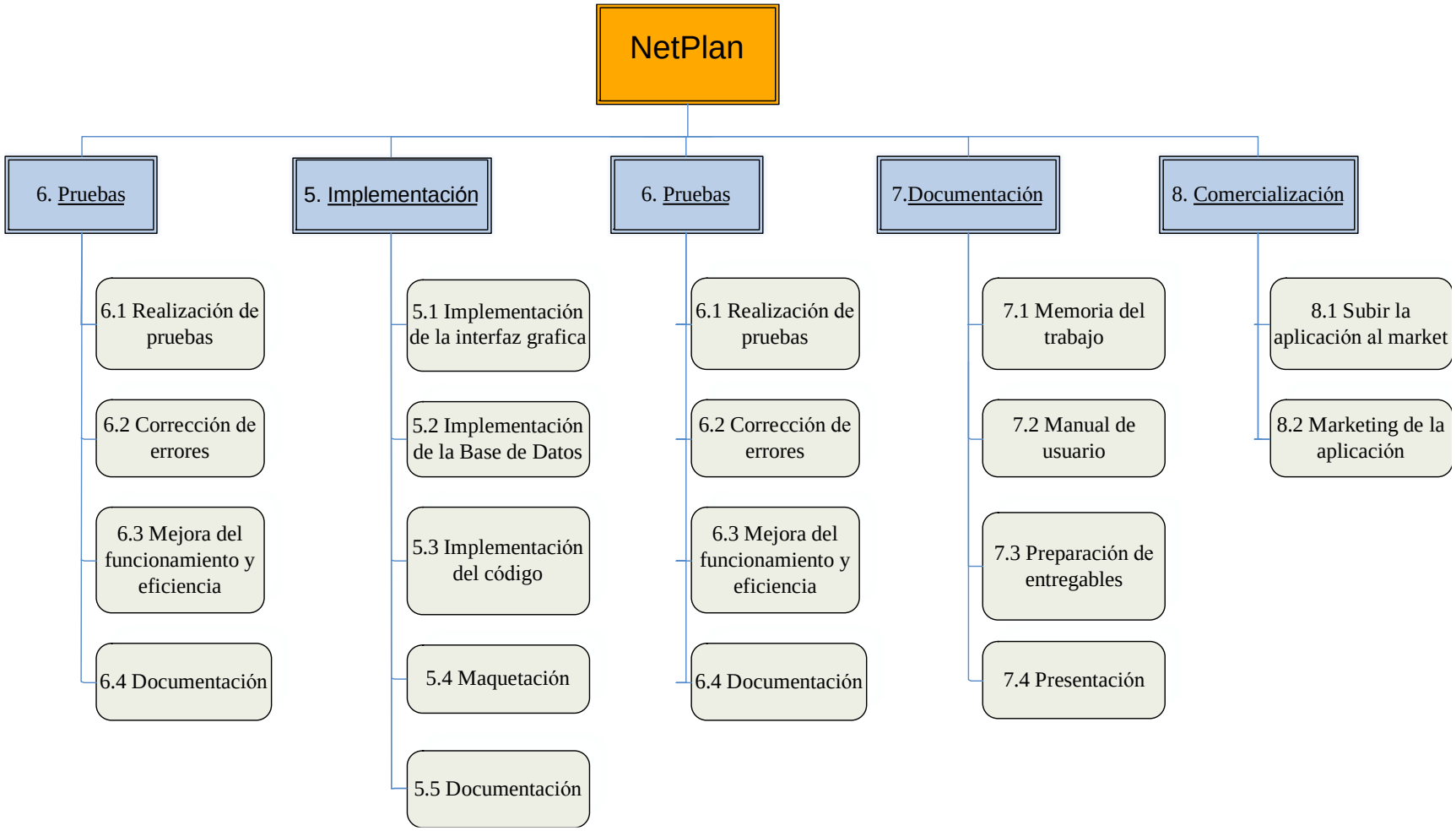


Figura 5: EDT segunda parte

t

1.4.3 Calendario y estructuración del trabajo

A continuación, se muestra el calendario con la distribución temporal y la distribución de horas del trabajo.

Calendario:

septiembre 2017

lu	ma	mi	ju	vi	sa	do
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

octubre 2017

lu	ma	mi	ju	vi	sa	do
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

noviembre 2017

lu	ma	mi	ju	vi	sa	do
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

diciembre 2017

lu	ma	mi	ju	vi	sa	do
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

enero 2018

lu	ma	mi	ju	vi	sa	do
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Tabla 2: Calendario

Leyenda:

Días no laborable	Diseño	Pruebas
Plan de trabajo	Aprendizaje	Documentación
Análisis	Implementación	Comercialización

Tabla 3: Leyenda de colores

Después de haber estimado los tiempos de cada tarea y de realizar un calendario, se puede observar que la fecha de inicio es el **20 de septiembre de 2017** y la fecha de finalización prevista es el **3 de enero de 2018**.

La distribución de las horas se realizará de la siguiente forma:

- De lunes a jueves, 2 horas y media al día.
- Viernes, 3 horas al día.
- Sábados y domingo, 4 horas al día.
- Días no laborables: 4 horas al día.
    - 12 de octubre de 2017.
    - 1 de noviembre de 2017.
    - 6 de diciembre de 2017.
    - 8 de diciembre de 2017.
- Días festivos, no se realiza trabajo.
    - 25 de diciembre de 2017.
    - 1 de enero de 2018.

El total de horas estimadas serán de **318 horas y 30 minutos**.

1.4.4 Diagrama Gantt

A continuación, se muestra el diagrama Gantt de la planificación del trabajo con cada tarea desglosada y las horas por tareas.

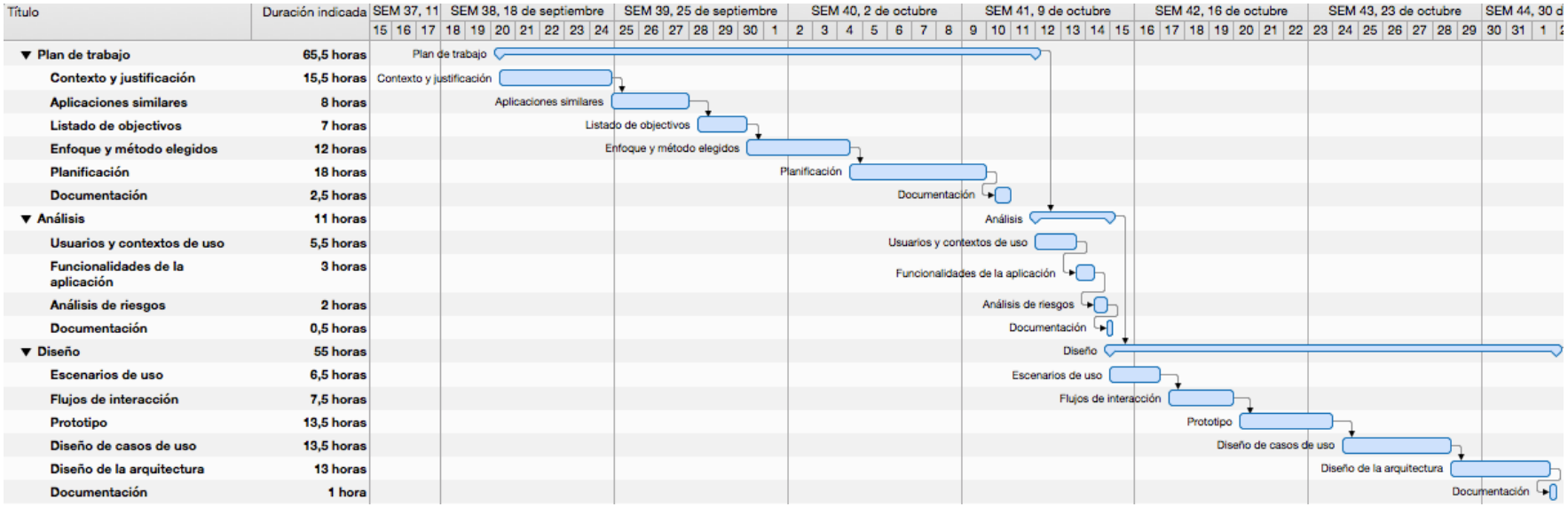


Figura 6: Diagrama Gantt primera parte

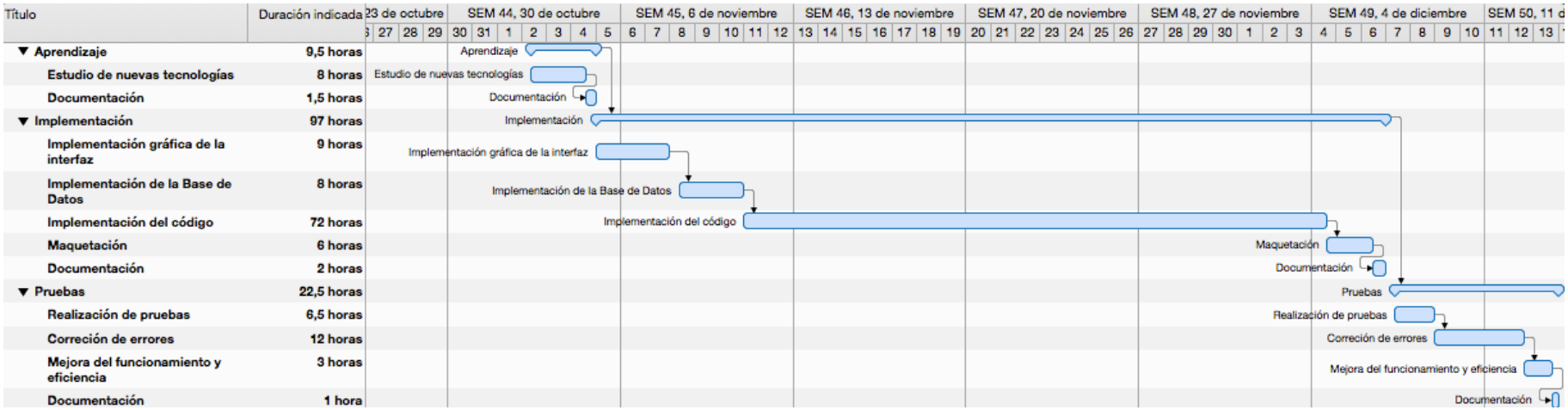


Figura 7: Diagrama Gantt segunda parte

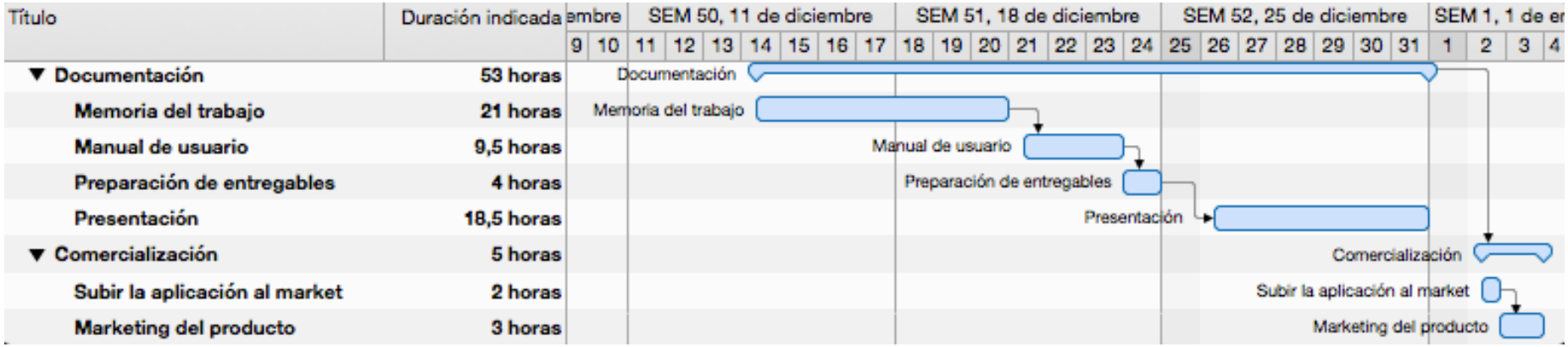


Figura 8: Diagrama Gantt tercera parte

1.5 Breve sumario de productos obtenidos

Los entregables al finalizar el trabajo serán:

- .zip con el código del proyecto.
- .apk instalable para dispositivos Android.
- .js de función realizada para notificaciones
- Memoria del trabajo final de máster en PDF.
- Vídeo de la presentación.

1.6 Breve descripción de los otros capítulos de la memoria

El resto de este documento estará compuesto por los siguientes capítulos:

- **Capítulo 2 "ANÁLISIS":** se identifican las características de los usuarios, sus necesidades y objetivos, así como el contexto de uso. Además, se describen las funcionalidades de la aplicación y se realiza un análisis de los riesgos de la aplicación.

- **Capítulo 3 "DISEÑO":** se realiza el diseño conceptual de la aplicación con los escenarios de uso y los flujos de interacción en el sistema, así como el prototipo de la misma y el diseño técnico con la definición de los casos de uso y el diseño de la arquitectura de la aplicación.

- **Capítulo 4 "DESARROLLO E IMPLEMENTACIÓN":** se explica la arquitectura, las tecnologías, las herramientas y la implementación necesaria para llevar a cabo las especificaciones de la aplicación.

- **Capítulo 5 "PRUEBAS":** se verifica el correcto funcionamiento de la aplicación realizando una serie de pruebas.

- **Capítulo 6 "VALORACIÓN ECONÓMICA":** se indicarán los gastos asociados al desarrollo y al mantenimiento del trabajo, así como los beneficios económicos obtenidos.

- **Capítulo 7 "CONCLUSIONES":** se describen las conclusiones del TFM y se detallan las posibles ampliaciones y mejoras de la aplicación.

- **Capítulo 8 "GLOSARIO":** se definen los términos y acrónimos más relevantes utilizados.

- **Capítulo 9 "BIBLIOGRAFÍA":** se detalla la bibliografía utilizada.

- **Capítulo 10 "ANEXOS":** se encuentran diferentes anexos del proyecto como son los diagramas de casos de uso extendidos, un manual de usuario y un manual de instrucciones.

2. ANÁLISIS

En este capítulo se definirán y especificarán las características de los usuarios y sus necesidades, así como los contextos de uso. Además, se describirán las funcionalidades que debe tener la aplicación y se realizará un análisis de riesgos.

2.1 Usuarios y contextos de uso

Para realizar los perfiles de usuario se ha realizado un formulario específico y completo para conocer las necesidades de los posibles usuarios.

A continuación, se exponen los datos más significativos de dicho formulario, que son los que permitirán conocer los perfiles de usuario de la aplicación y las necesidades asociadas a los mismos.

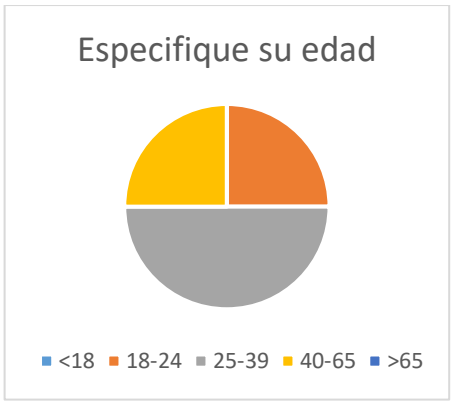


Figura 9: Encuesta 1

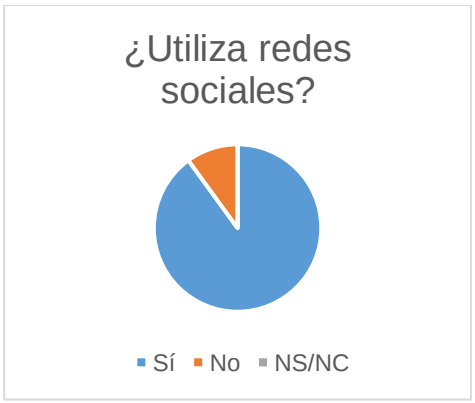


Figura 10: Encuesta 2

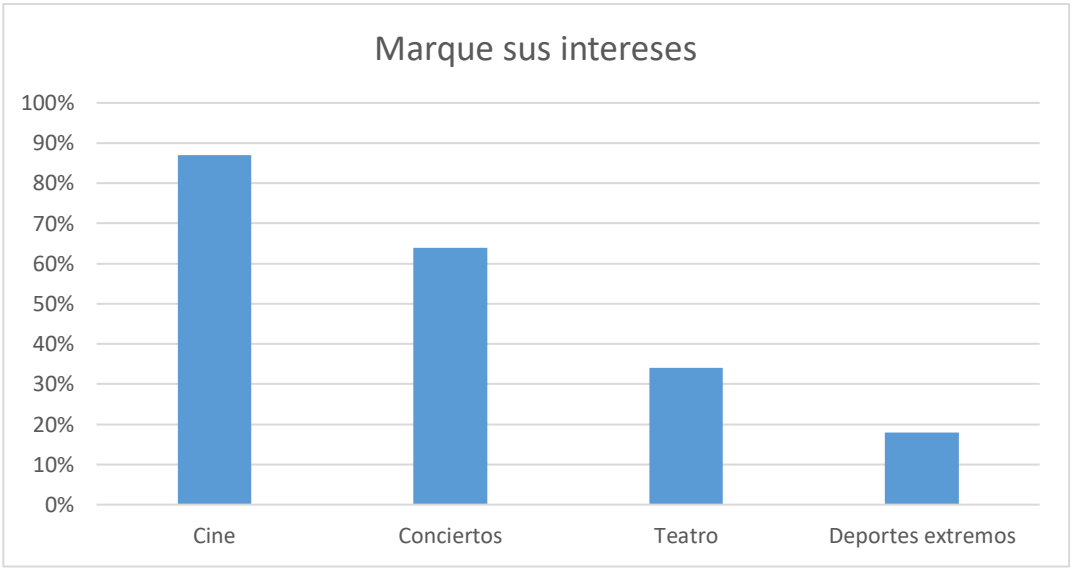


Figura 11: Encuesta 3

Figura 12: Encuesta 4

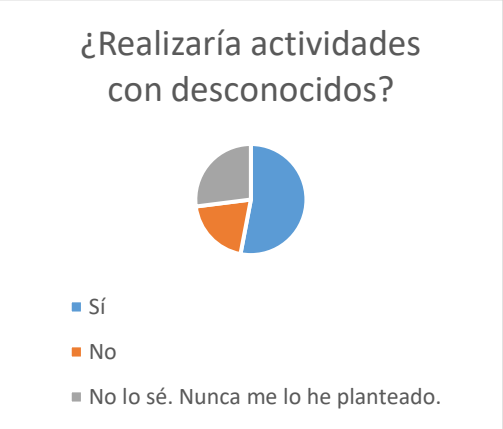


Figura 13: Encuesta 5

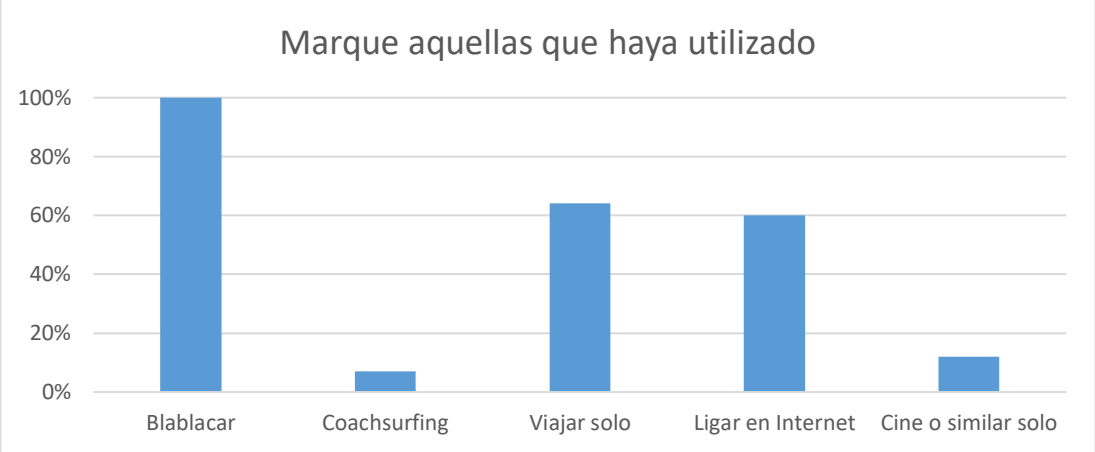


Figura 14: Encuesta 6

La encuesta la han realizado 53 personas de entre 18 y 65 años, aunque el grupo más activo ha sido el comprendido por personas de entre 25 y 39 años. Además, el 88% de las personas encuestadas afirma usar las redes sociales.

2.2 Necesidades detectadas

En las cuestiones planteadas a posibles usuarios, se ha comprobado que existen varias generalizaciones en las necesidades:

- Es necesaria una herramienta que sea accesible para todo tipo de usuarios independientemente de su edad.
- Una herramienta que haga las funciones de red social y permita la conexión con otros usuarios.
- Una herramienta que permita la inclusión de la mayor parte de eventos.

- La necesidad de que los usuarios puedan ser activos o pasivos, pudiendo participar de forma activa en la aplicación, creando nuevos eventos o comprobando qué eventos existen cercanos a una localidad a modo informativo.

Otra de las necesidades más importantes que los usuarios demandaron libremente en el formulario de estudio fue la necesidad de crear una aplicación que permitiese la búsqueda de eventos.

De esta forma, es como se han podido realizar los perfiles de usuario más importantes que usaría la aplicación.

2.3 Perfiles de usuario

Tras estudiar las necesidades de los usuarios, se pueden diferenciar 3 claros perfiles, que se exponen a continuación: 2 perfiles de usuario focales y uno secundario.

Los usuarios focales son los que usarán la aplicación de forma activa. Se dividen en dos perfiles de usuario diferenciados: un perfil que usará la aplicación para subir eventos y otro perfil que usará la aplicación para adherirse a grupos y crear los suyos propios.

El perfil de usuario que añadirá nuevos eventos o usuario colaborativo será el encargado de hacer la aplicación cada vez más grande y difundir en redes sociales los eventos generados.

El otro perfil focal será activo ya que se trata de un perfil de usuario aventurero que creará grupos para poder asistir a eventos en concreto con gente que se adhiera al mismo.

El tercer perfil que surge de las necesidades detectadas es el de un usuario que usará la aplicación de forma ocasional, para informarse de eventos cercanos a su localidad o para informarse de eventos en una localidad en concreto cuando vaya a visitar dicha localidad.

Estos tres perfiles engloban a los usuarios más importantes que utilizarán la aplicación.

2.4 Fichas de personas

[1]Para poder entender mejor los perfiles de usuario encontrados en las necesidades detectadas, se define una ficha de usuario por cada uno de los perfiles encontrados:

[1] Las fotos utilizadas para las fichas de usuario cuentan con el permiso de los mismos.

1. Ficha Borja Pérez: Perfil de usuario colaborativo. Usuario focal.
2. Ficha Iker Sesma: Perfil de usuario de aventurero. Usuario focal.
3. Ficha Arkaitz Carbajo: Perfil de usuario informativo. Usuario secundario.

2.4.1 Perfil focal: usuario colaborativo

Figura 15: Usuario colaborativo

Nombre: Borja Pérez

Edad: 40

Profesión: Marketing y atención al público

Descripción de la persona:

Borja tiene 40 años y está soltero. Vive solo en un pequeño piso en el centro de Bilbao y aunque es licenciado en económicas, trabaja en un centro cultural atendiendo y dando información al público que se acerca al mismo. Además, es el encargado de promocionar los eventos que se van a realizar en el propio centro.

Borja es un apasionado de la vida y lo que más le gusta es viajar.

El tiempo libre lo pasa con sus amistades y siempre buscando nuevas cosas que hacer para poder salir de la rutina. Es siempre el encargado de planificarlo todo ya que es el centro de unión de sus amigos.

Descripción del escenario:

Es primero de mes, y Borja comprueba los eventos que se van a realizar ese mes en el centro cultural donde trabaja y envía los boletines a los socios del centro cultural.

Borja entra en la aplicación y crea los nuevos eventos que se van a realizar ese mes en el centro para así poder promocionarlos.

2.4.2 Perfil focal: usuario aventurero

Figura 16: Usuario aventurero

Nombre: Iker Sesma

Edad: 34

Profesión: Diseñador

Descripción de la persona:

Iker vive con su novia en Gernika. Es graduado en informática, aunque no ha ejercido como tal, ya que siempre le ha apasionado la fotografía y el diseño. Ahora mismo está estudiando grado en diseño y trabaja como diseñador gráfico en una pequeña empresa de Bilbao.

A Iker le encantan los animales y tiene un perro desde hace 6 meses al que le encanta amaestrar y enseñar nuevos trucos.

Le encantan los deportes de riesgo y siempre que puede los practica.

Descripción del escenario:

Iker entra en la aplicación cuando va a trabajar en el tren y comprueba que se va a realizar goming desde el Puente Colgante de Bizkaia ese mismo domingo y se necesitan grupos de 4 personas para saltar.

Intenta convencer a sus amigos e incluso a sus compañeros de trabajo, pero no logra que nadie se anime.

Al volver a casa, entra en la aplicación y crea un grupo nuevo para realizar el evento con un número de 4 personas.

Al levantarse de la siesta, Iker comprueba que el grupo se ha completado y se pone en contacto con los usuarios para quedar para ese domingo y desayunar todos juntos y luego ir a realizar el salto.

2.4.3 Perfil secundario: usuario informativo

Figura 17: Usuario informativo

Nombre: Arkaitz Carbajo

Edad: 35

Profesión: Arquitecto de software

Descripción de la persona:

Arkaitz está casado y es padre de un niño de 2 años. Vive en Santutxu, el barrio más poblado de Bilbao. Todos los días trabaja en el centro de Bilbao como Arquitecto de Software y responsable del departamento de I+D de su empresa.

Es fan de Android, y siempre utiliza su Samsung Galaxy S7. Además, es un gran fan de las series norteamericanas que intenta ver siempre que puede, aunque desde que es padre solo le queda tiempo para verlas en el metro yendo a trabajar.

Como responsable de I+D, suele tener que viajar en varias ocasiones a Madrid y Barcelona para asistir a conferencias y reuniones de trabajo para proyectos internacionales.

Descripción del escenario:

Es un viernes a las 8.30 de la mañana. Arkaitz sale rumbo a Madrid junto con dos compañeros para asistir a una conferencia de Big Data.

Cuando coge el taxi en Madrid para dirigirse al hotel, comprueba en la aplicación si hay algún monólogo interesante esa noche por el centro de Madrid para poder desconectar de la conferencia y así poder disfrutar de la noche madrileña y conocer mejor a sus compañeros de trabajo.

Finalmente, hay un monólogo que le interesa ver y se lo comunica a sus compañeros que se apuntan sin pensárselo.

2.5 Funcionalidades

Tras comprobar las necesidades detectadas y los perfiles de usuario, las funcionalidades más importantes de las que dispondrá la aplicación son las siguientes:

- Registrar o identificar a un usuario a través de un formulario o a través de una red social.
- Crear/modificar/eliminar un evento informativo.
- Crear/modificar/eliminar un grupo de actividad.
- Adherirse a un grupo o cancelar esa adhesión.
- Buscar eventos/grupos por localidad y fecha.
- Contactar con usuarios.

2.6 Análisis de riesgos

En este apartado se identificarán los riesgos que pueden presentarse a lo largo de la creación del proyecto. Todo proyecto tiene sus riesgos y han de tenerse en cuenta. Por tanto, conocerlos de antemano puede evitar que sucedan o tratar de minimizarlos al máximo posible.

Por ello se estudiará la probabilidad de que ocurra, el impacto que causaría, así como las posibles medidas correctoras o preventivas que se podrían llevar a cabo.

Los riesgos que se presentan son los siguientes:

- *Errores en la planificación temporal debido a la falta de experiencia en cuándo a la planificación de proyectos se refiere*.
  - *Probabilidad*: Muy probable.
  - *Consecuencia*: No llegar a tiempo a la fecha acordada para la finalización del proyecto.
  - *Impacto*: Alto.
  - *Medida correctora*: Reestructuración de la planificación.
  - *Medida preventiva*: Hacer una planificación temporal lo más real posible.
- *Problemas de salud o de tipo personal*.
  - *Probabilidad*: Probable.
  - *Consecuencia*: Reducción o paralización del tiempo para realizar labores durante un tiempo determinado.
  - Impacto: Alto.
  - *Medida correctora*: Realizar horas no fijadas en el calendario para recuperar el tiempo perdido.
  - *Medida preventiva*: No hay posibilidad de evitarlo.
- *Pérdida de información del proyecto*.
  - *Probabilidad*: Improbable.
  - *Consecuencia*: Pérdida de datos.
  - *Impacto*: Variable. Depende de la pérdida.
  - *Medida correctora*: Rehacer las partes del proyecto pérdidas.
  - *Medida preventiva*: Realizar copias de seguridad.

- *Pérdida o inutilidad de los instrumentos de trabajo como puede ser la entrada de virus en el sistema operativo, descarga eléctrica en caso de tormenta*.
    - *Probabilidad*: Improbable.
    - *Consecuencia*: Inutilidad del equipo.
    - *Impacto*: Alto.
    - *Medida correctora*: En el peor de los casos reinstalar el sistema operativo y todas las herramientas necesarias.
    - *Medida preventiva*: Tener bien protegido el equipo.
- *Errores en la obtención de los requisitos funcionales del sistema*.
    - *Probabilidad*: Improbable.
    - *Consecuencia*: Retrasos en los plazos del proyecto.
    - *Impacto*: Alto (Si se detecta en las primeras fases sería menor).
    - *Medida correctora*: Se realizará de nuevo el análisis de funcionalidades.
    - *Medida preventiva*: Realizar un buen análisis.
- *Funcionamiento de la aplicación en un simulador diferente que en el dispositivo físico*.
    - *Probabilidad*: Probable.
    - *Consecuencia*: Ralentización de la aplicación, funcionamiento de la cámara de fotos, GPS diferente al programado o interacción simulada con los demás usuarios.
    - Impacto: Medio.
    - *Medida correctora*: Implementar los métodos para que se ajusten al dispositivo deseado.
    - *Medida preventiva*: En cuanto a la ralentización se puede evitar realizando un código eficiente. Pero el funcionamiento de la cámara o el GPS no se puede prevenir. Para la interacción con otros usuarios se podrá simular usuarios.
- *Estancamiento en el desarrollo de la aplicación debido al desconocimiento de la plataforma*.
    - *Probabilidad*: Probable.
    - *Consecuencia*: Pérdida de tiempo.
    - *Impacto*: Medio.
    - *Medida correctora*: Realizar buena información y sintetizarla. Acudir a expertos en el campo.
    - *Medida preventiva*: Búsqueda de una buena bibliografía, analizando las cuestiones desconocidas.
- *Funcionamiento inesperado de la aplicación*.
    - *Probabilidad*: Probable.
    - *Consecuencia*: Pérdida de tiempo.
    - *Impacto*: Medio.
    - *Medida correctora*: Corrección del funcionamiento
    - Medida preventiva: Realizar programas de prueba y añadir un sistema de trazas para posibles errores.

3. DISEÑO

3.1 Árbol de navegación

Una vez definidas las funcionalidades que debe tener la aplicación en la práctica anterior, se presenta el árbol de navegación que indica claramente el recorrido que hará el usuario pantalla a pantalla. En este capítulo, se presenta dicho árbol, además de las explicaciones necesarias para poder entenderlo.

3.1.1 Leyenda

Para poder entender el árbol de navegación se define un pequeño esquema que explica en detalle todas y cada una de las figuras y colores del mismo.

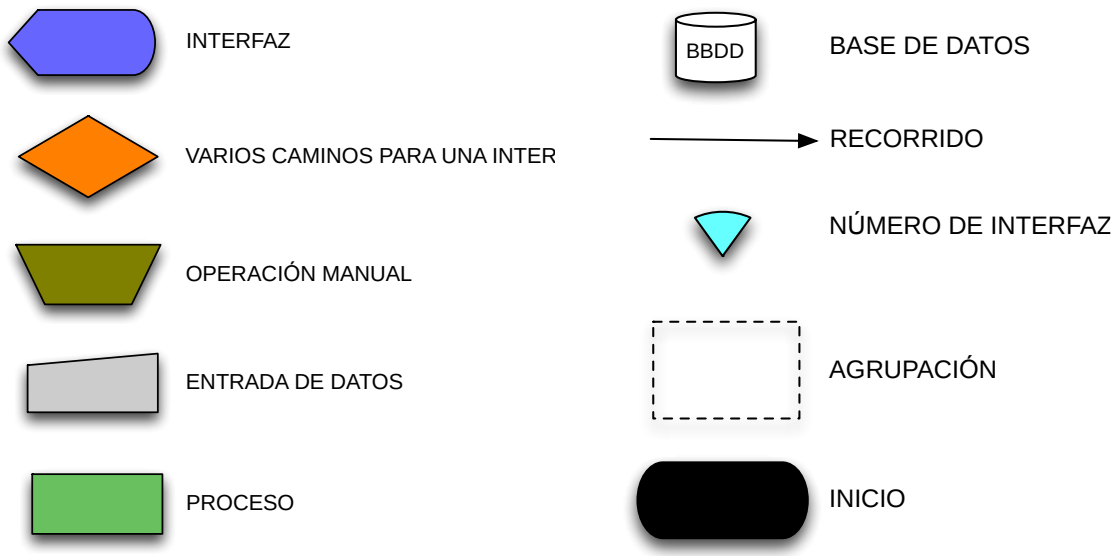


Figura 18: Leyenda diagrama de flujo

3.1.2 Diagrama de flujo

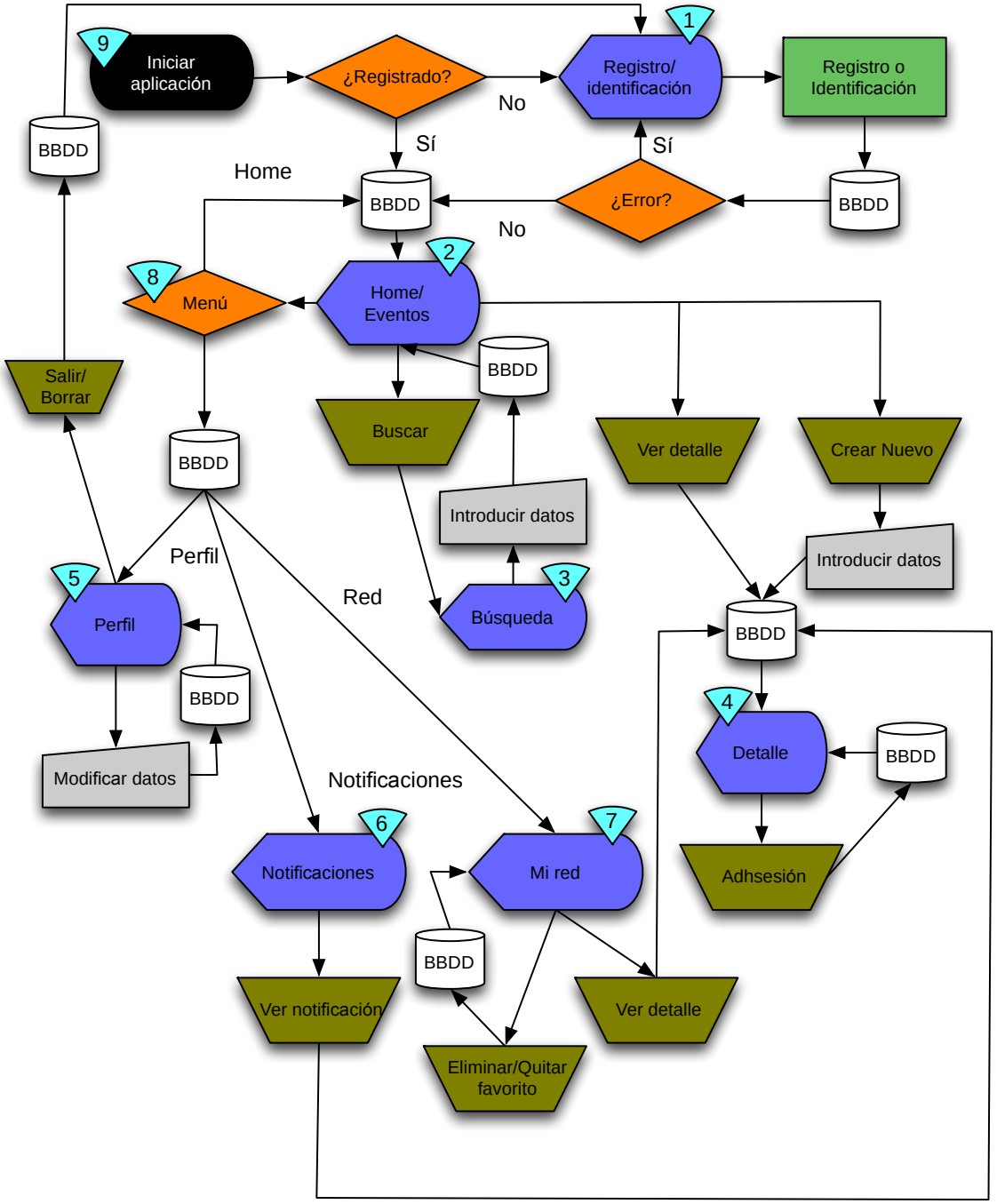


Figura 19: Diagrama de flujo

3.1.3 Memoria explicativa

Para poder entender mejor la interacción del usuario con la aplicación, en este capítulo se detalla el flujo de interacción. Teniendo en cuenta siempre la "*Figura 19: Diagrama de flujo*".

1. La aplicación se abre y muestra la splashscreen mientras se cargan los datos iniciales (número de interfaz – *NI*[2] – 9).
2. Si el usuario es nuevo se presenta la interfaz de registro/identificación (NI 1). Si ya ha sido registrado se presenta la interfaz de home (NI 2).
3. En la interfaz de registro (NI 1), el usuario puede elegir 3 acciones que se representaran en la aplicación como 3 interfaces diferentes: registrar un nuevo usuario en la aplicación a través de un formulario de registro, identificación de un usuario ya creado con anterioridad a través de su nombre de usuario y contraseña y la aceptación de requisitos para utilizar la aplicación con un usuario creado ya en una red social.
4. En NI 2 el usuario dispone de un listado con los eventos ordenados de forma ascendente por la fecha en la que se van a producir. Este listado muestra la foto del evento, y la información más relevante: el nombre, el lugar dónde se realiza y la fecha de realización del mismo. Además, tiene la opción de intercambiar el listado por el de los grupos con la misma ordenación que la de eventos. Este listado muestra la información más relevante de cada grupo: una foto, si está o no completo, el número de usuarios adheridos, el nombre, la fecha y el lugar de ejecución.
5. Desde NI 2 se puede acceder pulsando en el botón de búsqueda a la interfaz que permite realizar un filtrado (NI 3). En esta interfaz se puede elegir una localidad diferente a la geolocalización del dispositivo, fecha de ejecución, nombre del evento y por palabra clave. Al realizar la búsqueda, se vuelve a ver NI 2 con los resultados filtrados.
6. En NI 2 se puede seleccionar un elemento del listado. Esta acción presenta la interfaz de detalle del evento o grupo (NI 4) dónde se presentan los detalles del mismo. Si el usuario que ha entrado es el mismo que el creador del grupo o el evento, puede modificar la información.
7. Desde NI 2 se puede crear un nuevo evento o grupo pulsando en el botón de agregar. Mostrará una interfaz (NI 4) con los datos más relevantes para crear un evento o grupo. Éstos datos se verán modificados dependiendo el tipo de plan.
8. En NI 4 si el tipo de plan es un evento y no ha sido creado por el usuario conectado, podrá adherirse al grupo.
9. En cualquier interfaz (menos en NI 1 y NI 2), se puede ir a una interfaz intermedia de acceso a las opciones más relevantes de la aplicación (NI 8). En esta opción se podrá acceder directamente a la interfaz de la Home (NI 2), a la interfaz del perfil (NI 5), a la interfaz de notificaciones (NI 6) y a la interfaz de red (NI 7).
10. En NI 5 se muestra la información del usuario conectado: nombre de usuario, correo electrónico, contraseña, foto. Dependiendo del tipo de

[2] NI: Número de interfaz.

adhesión a la aplicación, se muestran unos datos u otros. Desde esta interfaz, se pueden modificar dichos datos y cerrar sesión o eliminar el perfil.

11. En NI 6 se ven las notificaciones de la aplicación a un usuario en concreto. En esta interfaz se ven 2 listados: notificaciones leídas y notificaciones no leídas. En el listado de notificaciones no leídas se puede interactuar con las mismas dependiendo el tipo de interacción. Entre las más importantes: un usuario ha comentado un grupo que está en la red del usuario, un usuario se adherido a un grupo que afecta al usuario, un usuario ha borrado o ha modificado un evento o un grupo que está en la red del usuario. Si se pulsa en un elemento de este listado se accede a NI 4.
12. En NI 7 se ven los planes que están en la red del usuario: eventos y grupos creados por el usuario, grupos en los que está el usuario, eventos favoritos y grupos cerrados realizados por el usuario con menos de un mes desde que se han realizado. Al pulsar en uno de estos elementos de los listados presentes se va a NI 4.
13. Desde NI 7 se puede borrar o modificar el plan presentado.

Una vez visto los flujos de interacción que puede realizar un usuario cabe destacar funcionalidades propias de los dispositivos móviles que no se especifican en este flujo.

1. En la NI 1, 4 y 5 se presenta la interfaz genérica del dispositivo para realizar foto o recuperarla de la galería.
2. Las interfaces solo estarán visibles en modo portrait.
3. En todas las interfaces se presenta un botón "*menú*" para poder ir a las acciones principales del sistema mediante una interfaz (NI 8).

3.2 Diseño de la aplicación

En este capítulo, se presentará el diseño de la aplicación a través de un prototipo de alta resolución[3] de las pantallas más importantes de la aplicación. Además, se explicarán la toma de decisiones en el diseño del prototipo.

3.2.1 Prototipo

En este apartado se presentan de forma ordenada las interfaces más importantes de la aplicación y que sirven de esquema para las demás interfaces:

[3] Puede haber variaciones sobre las interfaces finales de la aplicación.

Figura 20: Registro

Figura 21: Perfil

Figura 22: Home 1

Figura 23: Home 2

Figura 24: Detalle grupo general

Figura 25: Detalle grupo comentarios

Figura 26: Nuevo plan

Figura 27: Notificaciones

Todas las interfaces presentadas cuentan con la misma organización y servirán además como base y plantilla para otro tipo de pantallas como la búsqueda: un menú lateral que permite acceder a las partes más importantes de la aplicación en cualquier momento, una barra superior que especifica en dónde se encuentra el usuario, listados para mostrar de forma intuitiva la información al usuario y que permite la iteración con el usuario para poder ver detalles, un botón flotante en la esquina inferior derecha que permite realizar una acción importante sobre la interfaz en la que se encuentre el usuario (por ejemplo, añadir un nuevo evento) y una barra de navegación en la parte superior que permite al usuario moverse entre diferentes opciones de una misma interfaz.

3.2.2 Especificaciones generales

Todas las partes de esta fase se desarrollan para un dispositivo Android. Cabe destacar, que a mayor tamaño mejor será la experiencia del usuario.

La aplicación correrá sobre dispositivos móviles Android. En el diseño prima el contenido sobre el contenedor, es decir, en la simpleza está el gusto.

En este diseño, se ha intentado seguir (dentro de las posibilidades) esta regla y se ha intentado realizar las interfaces lo más simples posible, pensando además en la interacción intuitiva que pueda tener un usuario.

Los puntos básicos que sigue el diseño de la aplicación y que se puede ver en los siguientes apartados son los siguientes:

1. *Tipografía*: tamaños de letra amplios para poder ser legibles por cualquier usuario. Etiquetas cortas y claras.
2. *Navegación*: menos número de interacciones posibles para realizar una acción. Sistema de pestañas para acceder a las partes fundamentales de la aplicación y que tienen lógica por separado desde cualquier pantalla de la aplicación.
3. *Continuidad*: se informa al usuario en todo momento en dónde se encuentra a través del menú de opciones.
4. *Botonera*: botones simples e intuitivos y de suficiente tamaño para poder acceder a todas las funciones.
5. *Integración*: aplicación de logueo rápido a través de redes sociales.
6. *Intuición*: interfaces con gestos intuitivos para el usuario: tap, drag, pinch, spread…
7. *Agilidad de datos*: botones propios de Android para agilizar el proceso de inserción de imágenes.

3.2.3 Arquitectura

Tras detallar la paleta de colores seleccionados, en este punto se explica la arquitectura que sigue la aplicación, la navegación entre pantallas, y la presentación de datos al usuario.

Se ha seguido en la medida de lo posible la guía que ofrece Android para el desarrollo de interfaces de usuario: "https://developer.android.com/guide/practices/ui_guidelines/index.html".

Se buscan las siguientes directrices que han de seguir el diseño de las interfaces:

1. Claridad y simplicidad
2. Concisión y consistencia
3. Ligereza visual
4. Navegación intuitiva

En referencia a la navegación e interacción con la aplicación, el prototipo intenta seguir los principios básicos de "*la ley de Fitts*"[1] aplicada al diseño para móviles.

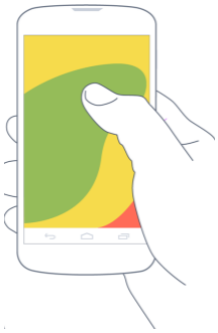


Figura 28: Ley de Fitts

La zona verde es dónde se presenta el contenido de fácil accesibilidad y en la zona roja los controles que no deban tocarse por error.

Se ha diseñado de esta forma, para poder tener una jerarquía entre todas las interfaces del prototipo. Así los listados son de fácil acceso y las opciones de menú y las acciones sobre las interfaces requieren total atención por parte del usuario.

Debido a la arquitectura seguida en la aplicación para disminuir y evitar al máximo el número de interacciones del usuario la navegación entre interfaces de la aplicación es del tipo "*experience-driven navigation*".

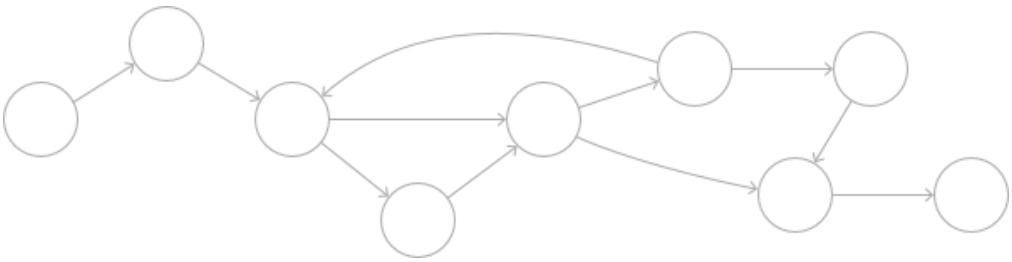


Figura 29: Experience-driven navigation

Este tipo de navegación es mucho más libre que el resto de navegaciones jerarquizadas. De esta forma, en cualquier interfaz se dispone de un icono de menú que permite la vuelta a cualquier interfaz raíz.

Así se evita el botón "*volver*" en este tipo de interfaces, que lo que proporcionaría al usuario sería una menor interacción con el sistema y mayor confusión.

3.3 Diagrama de Casos de Uso

En este apartado se representará gráficamente cómo debería interactuar la aplicación con los usuarios. Para ello dentro de los diagramas que nos ofrece el lenguaje UML, se utilizará el diagrama de casos de uso. Los diagramas de casos de uso extendidos se presentan en el "*9.1 Anexo I: Diagramas de casos de uso extendidos*".

A continuación, se expondrá el diagrama de casos de uso en el que se podrá observar que solo existe un actor en la aplicación. Éste será el usuario que se descargue la aplicación en su Smartphone e interactúe con ella.

Tras el diagrama se explicará brevemente que es lo que hace cada caso de uso.

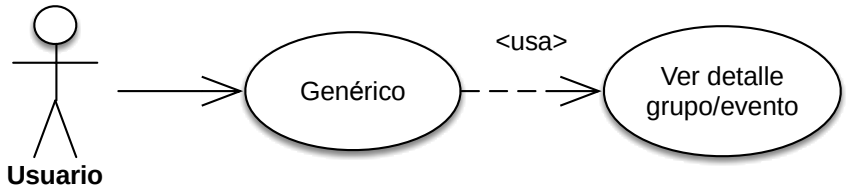


Figura 30: Casos de uso 1

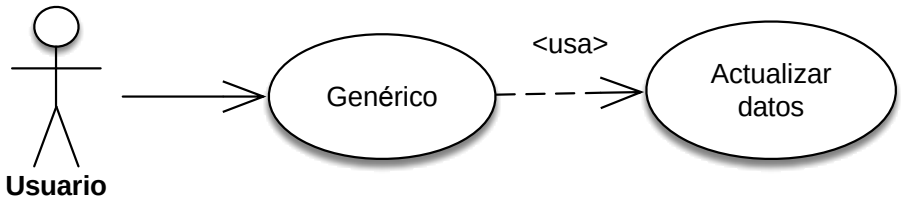


Figura 31: Casos de uso 2

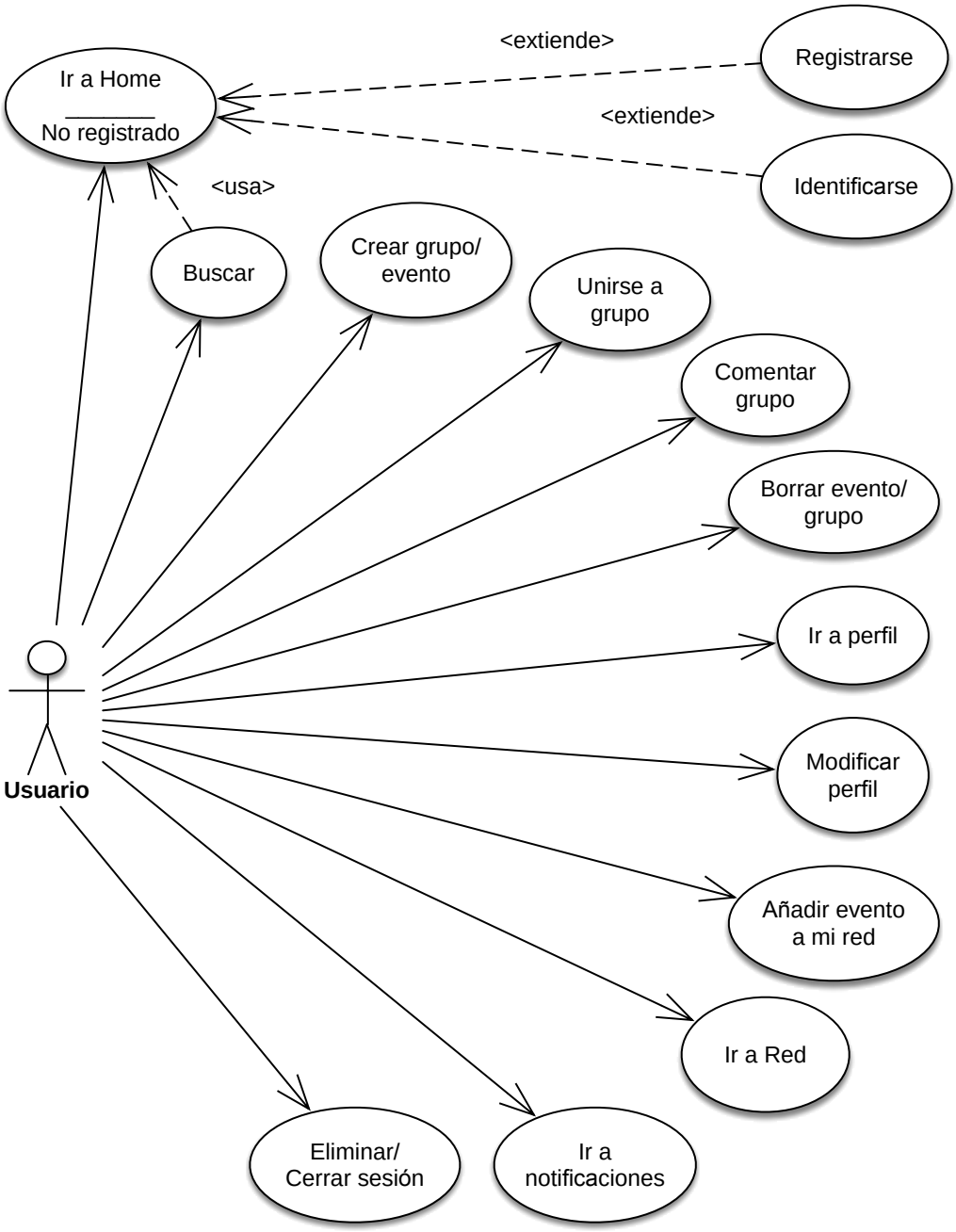


Figura 32: Casos de uso 3

3.3.1 C.U. Registrarse

Cuándo un usuario inicia la aplicación, se comprueba si existe o no ese usuario. Si no existe la aplicación cargará la vista de registro.

El usuario elegirá la opción de "*Registrarse*" e introducirá el nombre de usuario, el correo electrónico, la contraseña y una foto y pulsará el botón *"Siguiente"*. Si el usuario ha introducido correctamente los datos correctamente se crea un nuevo usuario. Si no, se mostrará al usuario una alerta.

3.3.2 C.U. Identificarse

Cuándo un usuario inicia la aplicación, se comprueba si existe o no ese usuario. Si no existe la aplicación cargará la vista de registro.

El usuario elegirá la opción de "*Iniciar sesión*" e introducirá el nombre de usuario y la contraseña y pulsará el botón *"Siguiente"*. Si el usuario ha introducido correctamente los datos se muestra la pantalla de "*Home*". Si no, se mostrará al usuario una alerta.

3.3.3 C.U. Ir a Home

El usuario pulsará la hamburguesa del menú y seleccionará la opción "Home". Se le mostrará al usuario los datos relacionados con los eventos y grupos cerca de la localidad en la que se encuentra.

3.3.4 C.U. Buscar

El usuario pulsará el botón "Buscar" e introducirá los datos para filtrar. Se mostrará a continuación los grupos y eventos filtrados por los criterios de búsqueda introducidos por el usuario.

3.3.5 C.U. Crear Grupo/Evento

El usuario pulsará el botón **"+"** e introducirá los datos necesarios para crear un grupo o un evento: la foto, el nombre, una descripción, lugar, día y hora y el tipo. El usuario pulsará el botón **"+"** y si todos los datos son correctos el nuevo grupo/evento se habrá creado.

3.3.6 C.U. Unirse a grupo

El usuario pulsará el botón "*Unirse y confirmar asistencia*". Se le mostrará una alerta de confirmación que el usuario aceptará.
Si el usuario pulsa de nuevo en el botón dejará de estar en el grupo.

3.3.7 C.U. Comentar grupo

Situado en el detalle de un grupo, el usuario pulsará en el espacio disponible para poder realizar un comentario. Escribirá el comentario y confirmará la publicación del mismo.

3.3.8 C.U. Borrar evento/grupo

El usuario pulsará el icono de borrado que únicamente aparecerá en los grupos o eventos creados por él. Una vez pulsado el botón, deberá confirmar el borrado y se le redirigirá al listado oportuno.

3.3.9 C.U. Ir a Perfil

El usuario pulsará la hamburguesa del menú y seleccionará la opción "*Perfil*". Se le mostrará al usuario sus datos: el nombre de usuario, el correo electrónico, la contraseña y si está o no vinculado con Facebook.

3.3.10 C.U. Modificar Perfil

El usuario pulsará en un dato de usuario que quiera modificar e introducirá los datos nuevos. Al pulsar "*ok*" se guardan los datos introducidos si son correctos.

3.3.11 C.U. Añadir/quitar evento a mi red

El usuario pulsará el botón de "*me gusta*" y el evento quedará registrado en la red del usuario.
Si el usuario vuelve a pulsar el botón, el evento se eliminará de la red del usuario.

3.3.12 C.U. Ir a red

El usuario pulsará la hamburguesa del menú y seleccionará la opción "*Red*". Se le mostrará al usuario la información de los grupos y eventos creados por él, los grupos y eventos favoritos que no hayan expirado y los grupos cerrados que se hayan realizado y que tengan menos de un mes de antigüedad.

Si un usuario pulsa en un elemento se le mostrará su detalle.

3.3.13 C.U. Ir a notificaciones

El usuario pulsará la hamburguesa del menú y seleccionará la opción "*Notificaciones*". Se le mostrará al usuario la información de las notificaciones registradas en el sistema.

3.3.14 C.U. Eliminar/Cerrar sesión

El usuario pulsará la hamburguesa del menú y seleccionará el icono de desconexión del menú que aparece al pulsar el botón de la hamburguesa. Se redirigirá al usuario a la pantalla de registro para que pueda identificarse o registrarse de nuevo.

Aunque son dos casos de uso por separado que cada uno tiene una finalidad, el proceso a seguir y la redirección que se hace en el sistema también.

3.3.15 C.U. Ver detalle grupo/evento

Este es un caso de uso generalizado en el que se recuperan los datos de un grupo o un evento para poder ver su detalle.

3.3.16 C.U. Actualizar datos

Este es un caso de uso generalizado en el que se actualizan los datos generales del usuario.

3.4 Modelo de dominio

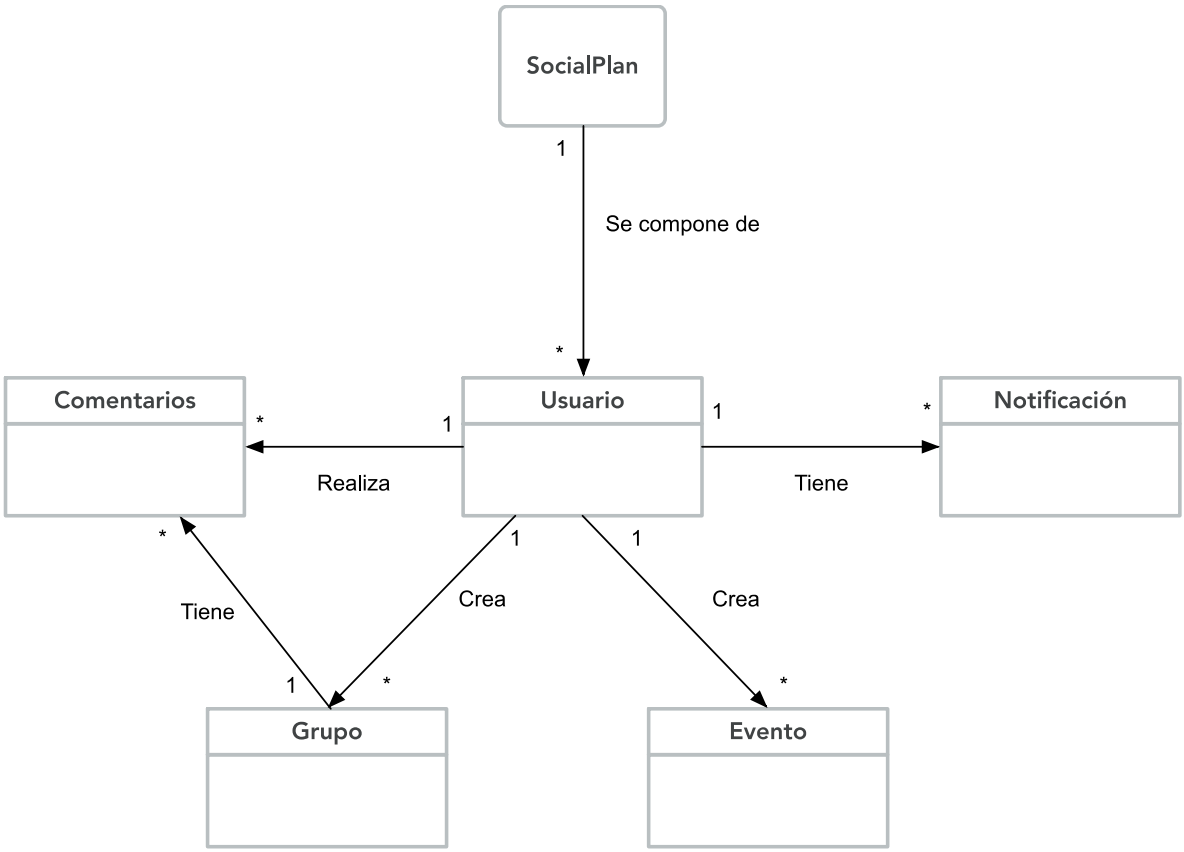


Figura 33: Modelo de dominio

3.5 Arquitectura del sistema

Durante este apartado se detalle el funcionamiento de la aplicación y se explica cómo están organizadas las clases que se usan y como están relacionadas entre ellas.

Es importante resaltar que para visualizar la relación existente entre las clases se utilizará un diagrama que representa el patrón MVC en vez del diagrama de clases. Se ha optado por este diagrama por que las aplicaciones móviles se rigen mejor por este patrón.

3.5.1 Especificaciones generales

La nomenclatura de flechas para los diagramas MVC de este capítulo es la siguiente: las flechas con línea continua (rojo) indican una asociación directa entre la clase de la que procede y la clase a la que apunta mediante la llamada de sus métodos; las flechas con línea discontinua (naranja) indican asociaciones indirectas, como pueden ser eventos. Además, para seguir este patrón MVC, se diferenciará con los siguientes colores: Azul-Controlador, Verde-Vista, Naranja-Modelo.

1. **Azul:** *activities y fragments*. Clases controladoras de las vistas, son de tipo *AppCompatActivity* y *Fragment* respectivamente recibiendo los eventos procedentes de la clase de la que extienden y gestionándolos correctamente para el correcto uso de la aplicación.
2. **Verde**: *layouts.* Contienen la definición de cómo son y de la estructura que tiene la vista a la que corresponde.
3. **Naranja:** *modelo.* Clases pertenecientes al modelo y contienen la información necesaria para la correcta gestión de base de datos y las vistas.

Dada la gran cantidad de atributos y métodos para las distintas clases, se ha optado por ocultar estos para evitar extensos y confusos diagramas.

3.5.2 Diagrama de clases y patrón MVC listas

A continuación, se presenta el diagrama de clases genérico para los listados existentes en la aplicación:

Un "*fragment"* que contendrá su "*adapter"* correspondiente para poblar los datos de las listas a través del "*modelo"* y presentará el "*ítem"* correspondiente con el que tendrá interacción el usuario.

Este diagrama se repetirá para todos los fragmentos que contenga listados.

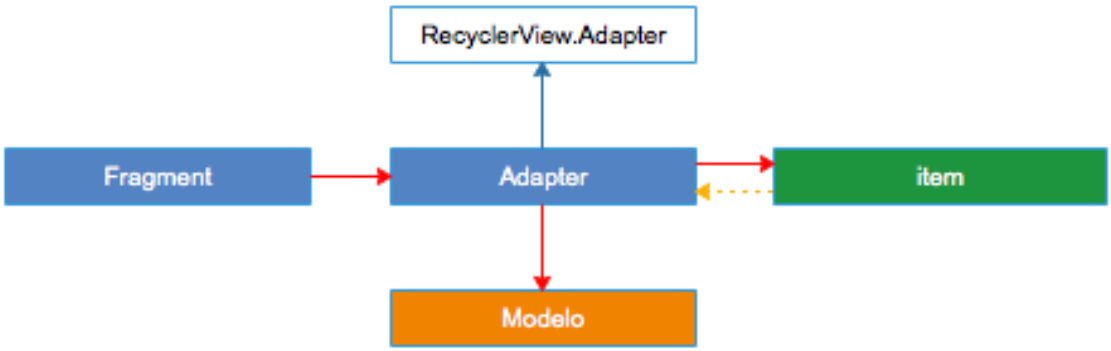


Figura 34: Diagrama de clases y patrón MVC listas

3.5.3 Diagrama de clases y patrón MVC general

A continuación, se presenta el diagrama de clases general de la aplicación. No están algunas clases para evitar repeticiones en el diagrama. Por ejemplo, el registro de usuario tiene una activity "*RegistroFormActivity*" y un layout "*activity_form_registro*" y la identificación del usuario está estructurada de la misma forma con una activity "*LoginActivity*" y un layout "*activity_login*".

En *"MainActivity"* se hace la referencia al *modelo* para no repetir la estructura en el *fragmento* correspondiente y evitar duplicidades.

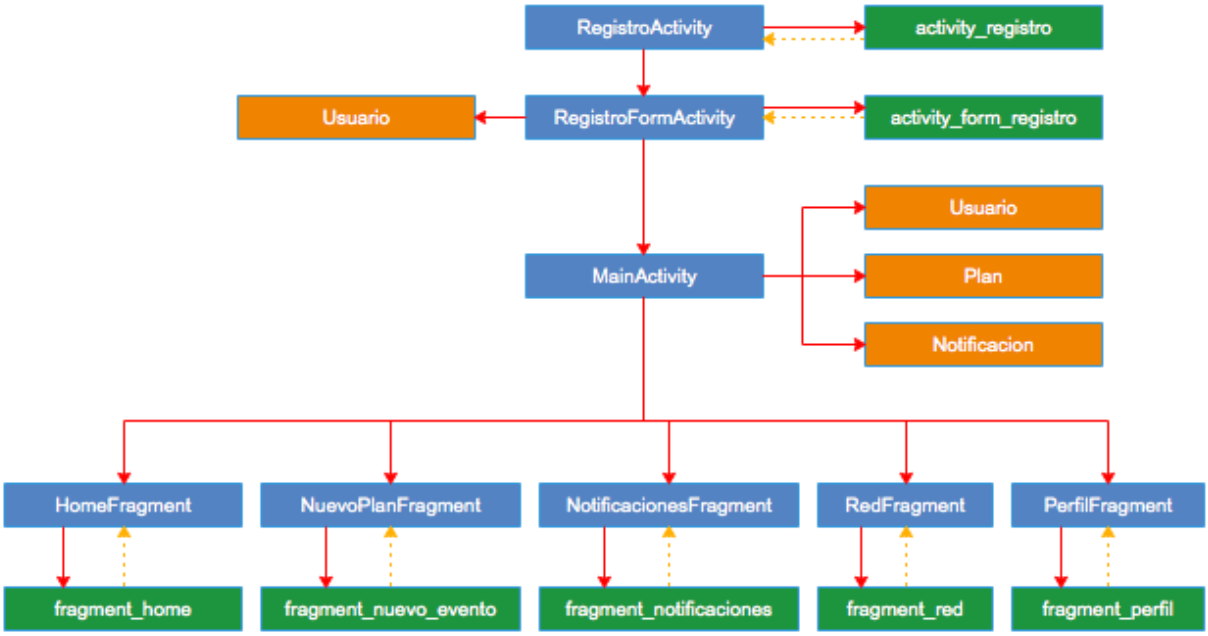


Figura 35: Diagrama de clases y patrón MVC general

3.6 Modelo de base de datos Firebase

La aplicación no cuenta con una base de datos relacional propiamente dicha, por lo que a continuación de presenta un diagrama con el modelo de base de datos utilizada en *Firebase*.
Además la aplicación, dispone de una base de datos similar a la de Firebase para guardar todos los identificadores de cada una de las acciones que se vayan insertando para hacer más sencillo el borrado si el usuario se da de baja.

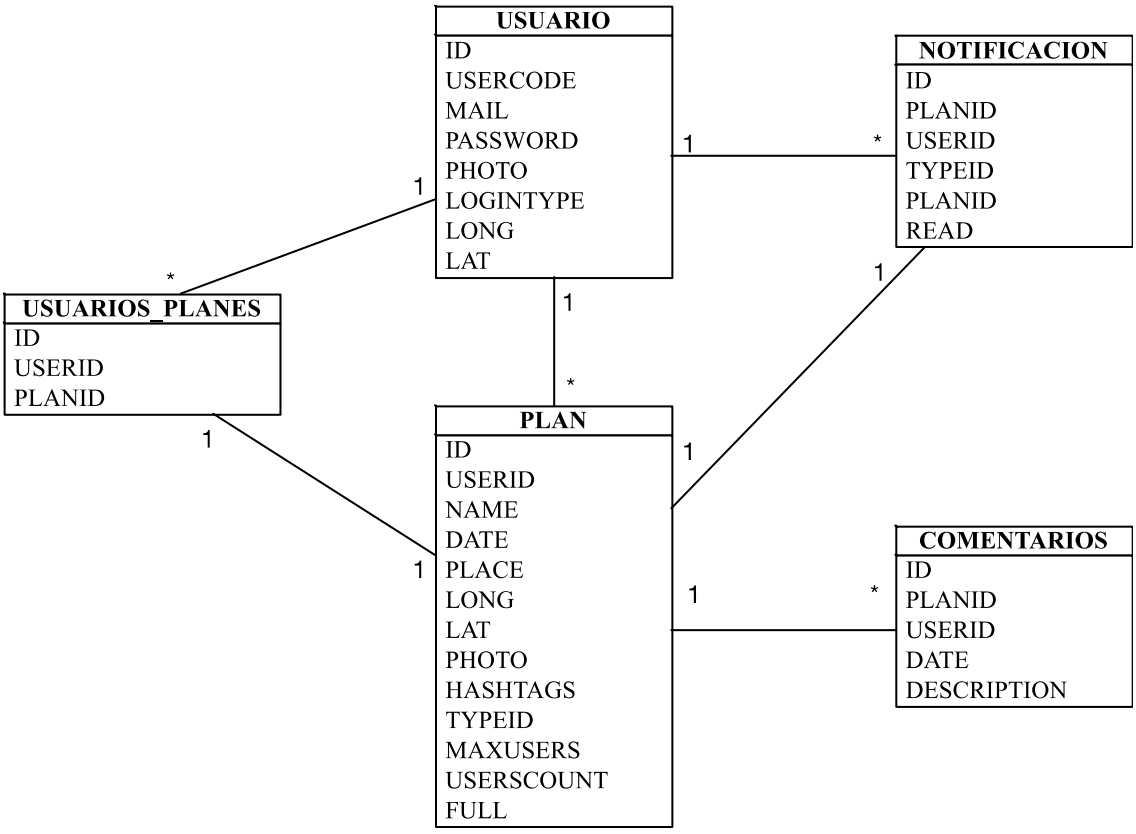


Figura 36: Modelo de base de datos Firebase

4. DESARROLLO E IMPLEMENTACIÓN

En este capítulo, se detallará cómo ha sido el proceso de desarrollo de la aplicación Android del TFM y se mostrará el código fuente de las operaciones más importantes de las que dispone la aplicación, obviando aquellas funcionalidades típicas del desarrollo de aplicaciones Android.

Además, se expondrán algunos de los problemas encontrados durante la implementación y cómo se han solventado.

Cabe destacar que esta es una aplicación Android, y tal y como se ha mencionado en capítulos anteriores debe ser accesible por la mayor parte de dispositivos que corren bajo esta plataforma. Por ello, la aplicación se ha desarrollado para la versión mínima 19 de Android. Aunque teniendo en cuenta que para aprovechar al máximo las funcionalidades que se ofrece en la aplicación, cuanto mayor sea el sistema operativo y el tamaño de pantalla del dispositivo, mejor será la experiencia de usuario.

4.1 Patrones de diseño

Para el desarrollo de la aplicación se han utilizado varios patrones de diseño que permiten diseñar la aplicación de una forma sencilla y segura siguiendo unas reglas establecidas para tener un código más accesible. Los patrones de diseño que se han utilizado son:

1. Inyección de vistas
2. Adapter
3. Modelo Vista Controlador

4.1.1 Inyección de vistas

Se ha utilizado "*ButterKnife"* que permite de una forma sencilla inyectar una vista directamente mediante su identificador sin tener que llamar a "*findViewById*", se utiliza la anotación "*@BindView*".

```
@BindView(R.id.toolbar) public Toolbar toolbar;
@BindView(R.id.drawer_layout) public DrawerLayout drawer;
@BindView(R.id.nav_view) public NavigationView navigationView;
```

Figura 37: Inyección de vistas

4.1.2 Adapter

Este patrón permite encajar una serie de datos con una vista Android, es decir, es la parte lógica que adapta estos datos para que sean visualizados.

Este patrón es muy utilizado en la aplicación ya que la mayoría de interfaces de la misma están formadas por vistas que cargan una serie de datos (p.ej.: *RecyclerViews* para las listas).

Este tipo de vistas no saben manejar la lista de objetos que se muestran dentro de ellas, por lo que será el "*Adapter*" el encargado de manejar los datos de manera que llenen la lista.

A continuación, un fragmento de código de la clase "*ComentariosRecyclerViewAdapter*" en la que se introducen los datos de los comentarios de un plan, rellenando la lista con su comentario, el nombre de usuario y la foto del mismo.

```
@Override
public void onBindViewHolder(final ComentariosHolder holder, int pos) {
    final Comentario comentario = items.get(pos);
    holder.getComent().setText(comentario.getComentario());
    holder.getUserComentName().setText(comentario.getUserName());

    if(comentario.getUserImage() != null){
        if(comentario.getLoginType() != null
                && comentario.getLoginType() == TipoLoginUserEnum.FACEBOOK.getId()) {
            Picasso.with(mContext).load(comentario.getUserImage()).into(holder.getUserComentImage());
        }else{
            final Bitmap bitmap = ImageUtils.convert(comentario.getUserImage());
            holder.getUserComentImage().setImageBitmap(bitmap);
        }
    }
}
```

Figura 38: Adapter

4.1.3 Modelo Vista Controlador

Es el patrón de diseño más utilizado en Android, en cuanto a patrones estructurales. Este patrón se utiliza para dividir el código de la aplicación en 3 partes fundamentales:

1. *Modelo*: las clases de datos.
    a. *model*: paquete en el que se guardan las clases del modelo de la aplicación.
2. *Vista*: las clases que hacen referencia a toda la parte visual del proyecto.
    a. *activities*: paquete en el que se guardan todas las actividades de la aplicación.
    b. *fragments*: paquete en el que se guardan todos los fragmentos de la aplicación.
3. *Controlador*: las clases de lógica.
    a. *adapters*: paquete que guarda todos los adapters utilizados para mostrar los datos de la aplicación.

A continuación, se muestra la estructura completa que sigue el TFM, en el que se puede ver la estructura mencionada:

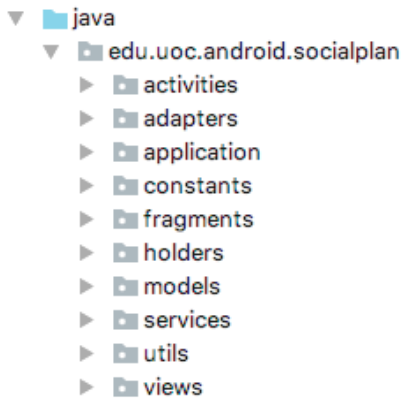


Figura 39: Patrón MVC

4.2 Implementación detallada

En este apartado, se detalla las funcionalidades más relevantes de las que dispone la aplicación y cómo se han resuelto.

4.2.1 Firebase

La mayor parte de las implementaciones hace uso de "*Firebase*" que es una plataforma para el desarrollo de aplicaciones web y móviles.

Se ha creado en "*Firebase*" el proyecto "*socialPlan*" para poder hacer uso de los servicios que proporciona esta plataforma: autentificación de los usuarios, bases de datos, funciones, reporte de errores, etc.

Uno de los servicios más utilizados a lo largo de la aplicación es el servicio de base de datos en tiempo real. Este servicio se utiliza para casi todas las operaciones que se realizan en la aplicación: crear un usuario, crear eventos, unirse a un evento, añadir un comentario, borrar eventos, etc.

La forma de uso en la aplicación es la siguiente (los ejemplos a continuación son para un "*plan*", el resto de llamadas a base de datos se hace de forma similar):

1. Obtener datos de la Base de datos de Firebase:

```
}else{
    queryPlans = realtimeDB
            .child(type)
            .orderByChild("userId")
            .equalTo(userId);
}

queryPlans.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        final GenericTypeIndicator<HashMap<String, Plan>> generic = new GenericTypeIndicator<HashMap<String, Plan>>() {};
        map = dataSnapshot.getValue(generic);
```

Figura 40: Firebase - Obtener datos

Se obtiene en un objeto "*Query*" la referencia de los objetos que se quiere recuperar. Esto sirve para realizar consultas con un parámetro en la base de datos (p.ej.: los eventos creados por un usuario). Llamando al método "*addListenerForSingleValueEvent*" se obtienen los datos de la query generada sin necesidad de escucha, es decir, la aplicación no se queda escuchando si ha habido cambios en la base de datos para actualizar debido a que solo se necesitan los datos en el momento que se realiza la búsqueda.

Si todo ha ido bien, se obtienen los datos a través de un objeto "*DataSnapShot*" y se parsean a un objeto conocido, en este caso al modelo "*Plan*".

2. Guardar un nuevo registro en la base de datos:

```
String planId = realtimeDB.child(tipoPlan).push().getKey();
plan.setId(planId);
realtimeDB.child(tipoPlan).child(planId).setValue(plan);
```

Figura 41: Firebase - Insertar

Para crear un nuevo registro, se obtiene el id que proporciona la base de datos para el registro nuevo a insertar. Esto se hace, si más adelante se necesita ese id para hacer referencia con otros objetos a guardar en la base de datos. Llamando al método "*setValue"* e informándolo con el objeto a guardar se realiza el insertado en base de datos.

3. Actualizar un registro en la base de datos:

```
realtimeDB.child(type).child(plan.getId()).updateChildren(plan.toMap());
```

Figura 42: Firebase - Updates

Para actualizar un objeto existente en base de datos, se obtiene la referencia en la que se encuentra en misma a través del nombre clave de la base de datos y a través del id del objeto que se quiere modificar. Una vez se tiene esa referencia, se hace una llamada a "*updateChildren*" con el objeto a modificar.

4. Eliminar un registro en la base de datos:

```
realtimeDB.child(FirebaseDBEnum.GRUPOS.getType()).child(plan.getId()).removeValue();
```

Figura 43: Firebase - Delete

El borrado funciona igual que la actualización pero llamando al método "*removeValue()*".

4.2.2 Registro

El registro en la aplicación consta de varias partes: el registro, la identificación de un usuario, el registro a través de Facebook y el olvido de contraseña en la identificación.

Para poder hacer uso de estas funcionalidades que proporciona "*Firebase*" se deben habilitar los proveedores de inicio de sesión correspondientes: correo electrónico/contraseña y Facebook.

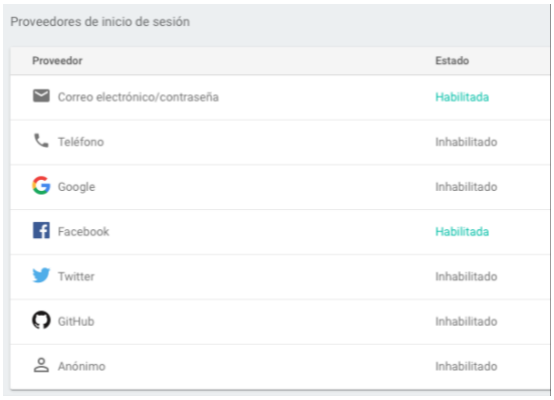


Figura 44: Registro - Dashboard

Para el registro de usuarios se debe crear una nueva instancia del objeto que permite la autenticación y a continuación llamar al método "*createUserWithEmailAndPassword"* con las credenciales que el usuario introduce en pantalla:

```
mAuth.createUserWithEmailAndPassword(userMail.getText().toString(), userPassword.getText().toString())
        .addOnCompleteListener( activity: RegistroFormActivity.this, (task) → {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d(TAG,  msg: "createUserWithEmail:success");
                    FirebaseUser user = mAuth.getCurrentUser();
                    addUser(user);
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG,  msg: "createUserWithEmail:failure", task.getException());
                    Toast.makeText( context: RegistroFormActivity.this, "en_Se ha producido un error",
                            Toast.LENGTH_SHORT).show();
                    progressBarDialog.hide();
                }
        });
```

Figura 45: Registro

Una vez se realiza la petición de crear un nuevo usuario, se comprueba el éxito de dicha operación. Si todo ha ido correcto, el objeto "*mAuth*" tendrá la información del usuario conectado y procederá a crear el usuario en la base de datos en tiempo real con los demás datos introducidos por pantalla.

El resto de funcionalidades del registro son similares en funcionamiento a la de un registro nuevo, por lo que no se explican en este apartado.

4.2.3 Geolocalización

La funcionalidad de la geolocalización es una parte fundamental de la aplicación y es la encargada de mostrar los planes según la localización del usuario. La localización de usuario hace uso de la implementación nativa de Android por lo que no se explica en este apartado.

Para la localización de los planes se guarda un elemento nuevo en la base de datos a través de la librería denominada "*GeoFire*" proporcionada por "*Firebase*", creando una instancia de este objeto y llamando al método "*setLocation*" con la referencia del id del plan a guardar y la latitud y longitud de la ubicación seleccionada.

```
geoFire = new GeoFire(realtimeDB.child(tipoPlanLocation));
geoFire.setLocation(planId, new GeoLocation(longitud, latitud));
```

Figura 46: Geolocalización - Insertar

Además, para obtener los planes según la distancia de la que se encuentran de un usuario se realiza una instancia de "*GeoQuery*" a la que se le pasan como parámetros la latitud, la longitud y la distancia máxima de los planes que se quieran recuperar.

Este listener tiene varios métodos ya que se utiliza para localizaciones en tiempo real y monitorizar los movimientos que se realicen sobre el rango de búsqueda realizado. En el caso de la aplicación, solo interesa el método "*onKeyEntered*" que se llamará por cada plan obtenido en el rango de búsqueda.

```
final GeoQuery geoQuery = geoFire.queryAtLocation(new GeoLocation(usuario.getLatitude(),
        usuario.getLongitude()), distance);
geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
    @Override
    public void onKeyEntered(String key, GeoLocation location) {
        if(map.containsKey(key)){
            if (FirebaseDBEnum.GRUPOS.getType().equals(type)) {
                gruposAdapter.addItem(map.get(key), type);
            } else {
                eventosAdapter.addItem(map.get(key), type);
            }
        }
    }
```

Figura 47: Geolocalización - Obtener datos

4.2.4 Notificaciones

La aplicación cuenta con un sistema de notificaciones al usuario en tiempo real cuando se produce una acción sobre algún elemento generado por el usuario. Entre las notificaciones destacadas se encuentran: los likes sobre un evento, la confirmación de asistencia a un grupo o el envío de un nuevo comentario en un plan.

"Firebase" tiene una herramienta de envío de notificaciones muy potente, pero pensada para el envío de notificaciones masivas a todos los usuarios o a usuarios subscritos a un tópico. Esto no tiene sentido en la aplicación.

Para implementar esta funcionalidad se ha hecho uso de los servicios de funciones de "*Firebase*" que con "*Node.js*" se crean funciones que luego se suben al servidor de "*Firebase*".

La función implementada dispone de un agente de escucha sobre la referencia "*messages*" de la base de datos. Cada vez que se realiza una inserción en ese grupo de registros el listener lo detecta y añade un nuevo evento a un tópico que está relacionado con un usuario en concreto.

```
// Listens for new messages added to messages/:pushId
exports.pushNotification = functions.database.ref('/messages/{pushId}').onWrite( event => {

    console.log('Push notification event triggered');

    //  Grab the current value of what was written to the Realtime Database.
    var valueObject = event.data.val();

  // Create a notification
    const payload = {
        notification: {
            title: "socialPlan",
            body: valueObject.text,
            sound: "default"
        },
    };

  //Create an options object that contains the time to live for the notification and the priority
    const options = {
        priority: "high",
        timeToLive: 60 * 60 * 24
    };

    return admin.messaging().sendToTopic("pushNotifications_"+valueObject.name, payload, options);
});
```

Figura 48: Notificaciones - Node.js

Para que esto tenga efecto en la aplicación el usuario debe subscribirse a un tópico en concreto. Esto se realiza a través del id de usuario que se proporciona siempre que se registra o se loguea un nuevo usuario:

```
FirebaseMessaging.getInstance().subscribeToTopic( s: "pushNotifications_"+prefs.getString( s: "userId",  s1: ""));
```

Figura 49: Notificaciones - Subscribir a tópico

Cuando un usuario cierra la sesión o desea eliminar sus credenciales, se debe tener en cuenta este tópico, y eliminarlo.

```
FirebaseMessaging.getInstance()
        .unsubscribeFromTopic( s: "pushNotifications_"+prefs.getString( s: "userId",  s1: ""));
```

Figura 50: Notificaciones - Salir del tópico

4.2.5 Crashlytics

Al tratarse de una aplicación móvil se ve necesario el uso de una herramienta para tener un reporte de los errores que se puedan producir en la aplicación. Se hace uso de otra herramienta de "*Firebase*" denominada "*Crashlytics*" que detecta todos los errores bloqueantes que se produzcan en la aplicación e informa el lugar en el que se ha producido dicho error.

```
Fabric.with( context: this, new Crashlytics());
```

Figura 51: Crashlytics

Todos estos errores aparecen en la consola de "*Firebase*", y es una forma de tener constancia de los mismos al no tener un servidor de trazas y poder detectarlos y su posterior corrección en versiones posteriores.

5. PRUEBAS

Durante este capítulo se exponen las diferentes pruebas a las que ha sido sometida la aplicación.

Las pruebas son consideradas satisfactorias si cumplen con los requisitos previstos e insatisfactorios si no los cumple. En caso de que la prueba sea insatisfactoria se explicará el motivo y las soluciones para tratar de corregirlo.

Esta es una de las etapas más importantes dentro del desarrollo de una aplicación y es donde se muestra la parte de la calidad de ésta.

5.1 Pruebas de interfaz

Estas pruebas validan todo lo referente al aspecto visual de la aplicación: se han hecho pruebas de la interfaz visual con el prototipo establecido en el apartado "*3.2.1 Prototipo*".

Los usuarios estaban confusos por el aspecto tan oscuro que presentaba la aplicación por lo que finalmente se ha decidido realizar las interfaces con un diseño mucho más claro que la versión definida en el prototipo.

Además, se ha verificado la ortografía y la correcta visualización de las imágenes, así como la internacionalización de la aplicación a los diferentes idiomas previstos: castellano e inglés.

5.2 Pruebas de navegación

Estas pruebas permiten comprobar la accesibilidad de los diferentes menús y la navegación por las diferentes vistas de la aplicación.

Para comprobar que la navegación por los menús funciona correctamente se ha navegado por cada uno de ellos de todas las formas posibles, obteniendo un resultado satisfactorio.

Sin embargo, se encontró un problema al entrar al detalle de un plan desde el menú de "*Mi red*". Al volver, al listado siempre aparecía el menú "*Home*". Esto se daba porque el botón de cerrar la interfaz siempre volvía a esta interfaz lo que permitía su refresco. Para solucionarlo, se añadió un control de pestañas a través del *fragment* con el método *setArguments()*, y pasándole como parámetro un objeto *Bundle* con la pestaña padre.

5.3 Pruebas de funcionalidad

Estas pruebas permiten determinar si se han cumplido los requisitos de la aplicación en cuanto al funcionamiento esperado.

Se ha creado un checklist (una lista de pruebas a validar) que comprobará que cuando se hagan cambios en la aplicación, los diferentes módulos funcionan correctamente.

La lista de pruebas a validar que se presenta a continuación está basada en los casos de uso del apartado "*3.3 Diagrama de casos de uso*".

Evento	Resultado
Registro de un nuevo usuario.	**OK (*1)**
Identificación del usuario.	OK
Registro a través de Facebook.	OK
Envío de correo electrónico al olvidar contraseña.	OK
Entrar solo cuándo el usuario no esté registrado o no tenga sus datos obligatorios cumplimentados.	OK
Permisos de ubicación y cámara.	OK
Carga de los eventos al entrar en la Home	OK
Carga de los eventos al aceptar los permisos de ubicación:	**OK (*2)**
Validación de los datos obligatorios para crear un plan.	OK
Modificar un plan.	OK
Pulsar botón de Like/Unlike en un evento.	OK
Unirse a un grupo/Salirse de un grupo.	OK
Enviar un comentario.	**OK (*3)**
Filtro de búsqueda por rango de distancia.	OK
Cargar notificaciones.	OK
Envío de notificaciones a un usuario.	**INFO (*4)**
Carga de eventos/grupos/favoritos de un usuario.	OK
Eliminar un evento/grupo de un usuario.	OK
Modificar los datos de un usuario.	**OK (*5)**
Cerrar sesión o borrar usuario.	OK
No conectividad	**INFO (*6)**
Búsqueda por nombre de eventos/grupos y localidad	**INFO (*7)**

Tabla 4: Checklist

1: Se crea un nuevo usuario siempre que se introduzca correctamente los datos. Sin embargo, hay un problema que se corregirá en siguientes versiones y es la validación de la existencia del correo electrónico (trabajo futuro).

2: Se producía un error a la hora de cargar los eventos al confirmar los permisos de ubicación y era necesario volver a cargar la "*Home*". Ocurría lo mismo al aceptar los permisos de la cámara, ya que había que volver a pulsar el botón de añadir foto para poder seleccionarla. Se arregló con el método "*onRequestPermissionsResult*" que comprueba la aceptación o denegación de los permisos y si se han aceptado se realiza la carga de los eventos o la apertura de la cámara.

3: Al enviar un comentario, se producía confusión para el usuario. Debido a que se daba a enviar, pero en la aplicación parecía que no pasaba nada. Esto se solucionó añadiendo un *SnackBar* de confirmación, limpiando el texto y lo más importante, ocultando el teclado de la siguiente forma:

```
if (view != null) {
    InputMethodManager imm = (InputMethodManager)getActivity().getSystemService(Context.INPUT_METHOD_SERVICE);
    imm.hideSoftInputFromWindow(view.getWindowToken(), flags: 0);
}
```

Figura 52: Error comentarios

4: Cuando se envía notificación de un usuario a otro usuario, el usuario receptor la recibe según el idioma del dispositivo del usuario emisor. Se debe a que se forma el mensaje en el dispositivo emisor. Ocurre lo mismo con la hora que se muestra en la notificación y también en los comentarios de un evento. Se considera trabajo futuro y se arreglará en siguientes versiones. Además, la aplicación no responde abriendo el evento/grupo que genera la notificación y también se considera trabajo futuro.

5: Al modificar los datos del usuario, tanto la foto como el nombre de usuario, no se modificaban los mismos en el menú lateral y parecía que no se había hecho ningún cambio. Esto se corrigió añadiendo un *SnackBar* de confirmación y modificando los datos del menú lateral sobre la activity principal "*MainActivity*". Esto es posible, ya que esta activity es la que tiene el control del menú lateral y de la barra superior y carga los fragmentos de cada menú.

```
Snackbar.make(getView(), "en_Los datos del usuario se han actualizado correctamente.", Snackbar.LENGTH_LONG).show();
((MainActivity)getActivity()).userName.setText(usuario.getName());
```

Figura 53: Error menú lateral

***6:** Al no haber conectividad había fallos en el sistema, se ha solucionado mostrando un SnackBar de que no hay conectividad y evitando los errores. Sin embargo, la aplicación no mostrará nada.

***7:** No se ha implementado en esta versión. Solo se ha dispuesto el filtro por distancia de un usuario.

5.4 Pruebas de integración

Estas pruebas permiten determinar cómo se comporta la aplicación dentro de la plataforma Android.

Evento	Resultado
Simular llamada entrante mientras se ejecuta la aplicación.	OK
Simular una alerta de memoria.	OK
Almacenamiento y persistencia de datos al entrar y salir de la aplicación.	OK
Instalación y desinstalación de la aplicación.	OK

Tabla 5: Pruebas de integración

6. VIABILIDAD ECONÓMICA

En este punto se realizará un análisis sobre la inversión que habrá que realizar en el proyecto, así como la forma con la que se intentará recuperar dicha inversión y obtener además beneficios.

Se ha supuesto que, para realizar el proyecto, formo yo la empresa y que yo también soy el programador. Por tanto, el modelo de negocio se basa en que todos los costes de la inversión, hardware, software y costes de marketing, los asume la propia empresa sin financiación externa.

6.1 Inversión Inicial

A continuación, se analiza la inversión inicial que se aportará en el proyecto para poder llevarlo a cabo.

6.1.1 Mano de obra

Los gastos de desarrollo de la aplicación tratan dos elementos: el precio por hora del programador y el número de horas para realizar la aplicación.

Actualmente el salario bruto medio de un ingeniero técnico informático es de 20.000€/año. Sumando a esta cantidad el 30% que la empresa paga a la Seguridad Social (6.000€/año), el coste anual del trabajador para la empresa es:

20.000 + 6.000 = 26.000 €

Esta cantidad se basa en las tablas salariales del 2.011 del Convenio de Oficinas y Despachos de Bizkaia, para un analista/programador[4].

El número de horas por trabajador al año según este convenio son 1.742. Con estos datos, el coste por hora del trabajador para la empresa es de:

26.000€ / 1.742h = 15 €/h

La duración del proyecto es de 318,5 horas en total. Por lo tanto, el coste de la mano de obra será:

Mano de obra = 318,5h (duración del proyecto en horas) x 15 €/hora (salario) = **4.777,5€**

[4] El salario real es de 24.482,94, pero se han cogido esos número para tener un redondeo aproximado.

6.1.2 Hardware

Para la realización de este proyecto se utilizará un Mac valorado en 1.200 € y un dispositivo Android valorado en 199 €.

Debido a que se realizará una intensiva utilización de estos componentes se debe realizar una amortización de los mismos.

Considerando una vida útil para el Mac de 4 años y de 3 años para el dispositivo Android:

- Amortización anual del Mac 1200/4 = 300 €
- Amortización anual dispositivo Android 199/3 = 66,3 €
- Con una duración del proyecto de 15 semanas aproximadamente y un uso para el Mac de un 40% del tiempo y un 10% del tiempo para el iPhone 4S se tiene:
- Gasto amortización del Mac: (300 € x 15 semanas x 0,4 (porcentaje de uso)) / 52 semanas del año = 34,61 €
- Gasto amortización del dispositivo: (66,3 € x 15 semanas x 0,1 (porcentaje de uso)) / 52 semanas del año = 1,91 €

En resumen:

Gasto total amortización hardware = Mac + dispositivo = 34,61 + 1,91 = **66,10€.**

6.1.3 Software

Todo el software utilizado en la elaboración del trabajo será de libre licencia por lo que no habrá gastos de amortización para el mismo.

6.1.4 Otros gastos

Se incluirán los distintos gastos que se pueden asociar al proyecto:

Luz = 40 €	Agua = 5 €
Teléfono = 5 €	Internet = 40 €

El total de estos gastos se estiman en **90 €.**

6.1.5 Costes de marketing

Se incluirán en estos gastos el coste de adquirir la licencia de desarrollador de "*Google Play*" el cual permitirá distribuir y comercializar la aplicación.

La licencia cuesta $25, suponiendo un cambio de 1$-1€, 25€**.**

En cuanto al marketing de la aplicación sería suficiente con presentarla en la "*Google Play*".

También se puede ofrecer una versión gratuita de pruebas de la aplicación, pero con funciones limitadas, así la gente usará la aplicación y hablarán de ella, para posteriormente dar el salto a la aplicación completa de pago.

También se buscará otro tipo de promoción (Facebook, Twitter), pero con coste 0.

6.1.6 Coste total del proyecto

PARTIDA	COSTES
Mano de Obra	4.777,5 €
Hardware	66,10 €
Software	0 €
Otros gastos	90 €
Costes de marketing	25 €
TOTAL	**4.958,60 €**

Tabla 6: Coste total

El coste total del proyecto asciende a **4.958,60 €**.

6.2 Recuperación de la inversión

En este apartado, se analizará la forma de recuperar la inversión calculada en el punto "*2.1.6 Coste total del proyecto*".

6.1.3 Modelo de negocio de Google Play

Para poder distribuir la aplicación en la tienda virtual de *Google*, "*Google Play"*, se necesita la licencia de desarrollador correspondiente.

Esta licencia tiene una duración de 1 año, pasado ese plazo no se pueden subir más aplicaciones a la tienda, y las aplicaciones ya subidas no estarán visibles para nuevas adquisiciones por parte de los usuarios hasta que no se renueve la licencia.

Como *Google* se lleva parte de los ingresos, el precio de venta de la aplicación no serán íntegramente ingresos, sino que se percibirá el 85% de estos por cada aplicación vendida.

6.1.2.1 Análisis de escenarios y rentabilidad del proyecto

Teniendo en cuenta el modelo de negocio de "*Google Play*", se estudiará la rentabilidad del proyecto.

Según App Ani el 98% de los ingresos de "*Google Play*" vienen por las conocidas aplicaciones "*Freemium*", es decir, aplicaciones gratuitas que venden productos o subscripciones dentro de ella.

Teniendo en cuenta esto, y realizando unas subscripciones de 1€ al mes o de 5€ al mes para la creación de más planes, o poder acceder a todas las funcionalidades de la aplicación:

Estudio A: Precio de la subscripción a 1€/mes

Número de subscripciones en un año	Precio	Ingresos anuales	15% de ingresos para Google	85% de ingresos
100	1 €	1.200 €	180 €	1.020 €
500	1 €	6.000 €	900 €	5.100 €
1.000	1 €	12.000 €	1.800 €	10.200 €

Tabla 7: Estudio A

Estudio B: Precio de la subscripción a 5 €

Número de subscripciones en un año	Precio	Ingresos anuales	15% de ingresos para Google	85% de ingresos
100	5 €	6.000 €	900 €	5.100 €
500	5 €	30.000 €	4.500 €	25.500 €
1000	5 €	60.000 €	9.000 €	51.000 €

Tabla 8: Estudio B

La rentabilidad del proyecto vendrá dada por: el número de subscripciones x 12 meses x 85 % del precio de la aplicación debe ser igual al coste del proyecto, 4.958,60 €.

Por tanto, tendrían que producirse al menos:

Número de descargas	Precio
476	1 €
96	5 €

Tabla 9: Rentabilidad proyecto

6.1.2.2 Precio final para la aplicación

Para el estudio B se necesitan menos subscripciones para recuperar la inversión, pero será mucho más difícil.

Se podría realizar una oferta inicial gratuita de la aplicación para que llegue al mayor número de usuarios, y una vez se tenga una cartera amplia de clientes y pequeñas salas sigan queriendo promocionarse en la aplicación, se podría realizar el sistema de subscripciones a 5€ para recuperar la inversión e incluso conseguir beneficios.

Como táctica de precio psicológico se utilizará el **precio** de **4,99 €**.

7. CONCLUSIONES

Una vez finalizado el proyecto, se pueden sacar una serie de conclusiones, tanto personales como objetivas del desarrollo del proyecto, que se expondrán a continuación. Además, se realizará un análisis del seguimiento del proyecto y las líneas de trabajo futuro que se deberán abordar para tener un proyecto exitoso.

7.1 Análisis y reflexión

El proyecto tenía como objetivo la realización de una aplicación nativa Android y el aprendizaje del proceso de creación de una nueva aplicación en una plataforma de desarrollo ya conocida, pero con un servidor en la nube del que no tenía conocimientos previos.

Se puede decir que ambos objetivos han sido cumplidos. Además, se ha intentado en todo momento crear una aplicación atractiva y fácil de usar por el mayor número de dispositivos que usen la plataforma Android y que estén en uso hoy en día. También se dedicó bastante tiempo al aspecto visual que debería tener, consiguiendo una aplicación funcional y con una interfaz intuitiva y atractiva para los usuarios.

Por otra parte, el nivel de conocimiento obtenido durante la realización del proyecto ha sido muy elevado. El proceso ha sido costoso, sobre todo en la parte de búsqueda y aprendizaje de la información, ya que los conocimientos del servicio "*Firebase*" y de "*Node.js*" eran nulos antes de comenzar el trabajo.

Se ha intentado cumplir con la planificación inicial, aunque se ha producido una variación considerable en las horas utilizadas para llevar a cabo la tarea de *desarrollo e implementación.* Se ha debido al desconocimiento de los servicios propuestos por "*Firebase*" que han sido necesarias más horas para poder aprender correctamente a usar los servicios. Hay que tener en cuenta que la mayor parte de las funcionalidades de la aplicación usan dichos servicios (bases de datos en tiempo real, geolocalización, autentificación de los usuarios, notificaciones, etc.), por lo que en cada paso que se daba en la implementación se ha tenido que buscar nueva información, ejemplos y documentación.

Además, se ha invertido tiempo de la implementación para intentar realizar un nuevo diseño de la aplicación, debido a que, en las primeras pruebas con usuarios reales, los colores tan oscuros y algún elemento de la interfaz daba lugar a errores y confusiones. Entre estos problemas se encontraban errores de navegación explicados en el apartado "*5.2 Pruebas de navegación*".

Otra de las confusiones que se producían en la aplicación y se debía a errores del prototipo, era el botón de cerrar sesión, que inicialmente se encontraba al lado del botón de aceptar los cambios en la opción de menú de "*Perfil*" y

finalmente se quitó esta funcionalidad y se añadió una nueva opción de menú "*Cerrar sesión*".

En la siguiente tabla, se muestran la desviación en las horas previstas y las horas empleadas en el desarrollo e implementación de la aplicación:

Tarea	Horas previstas	Horas empleadas
Aprendizaje	9,5 horas	25 horas
Desarrollo e implementación	97 horas	107 horas
Pruebas	22,5 horas	28 horas

A pesar de que el calendario se ha mantenido en fechas, la desviación de horas total es de **31 horas** por lo que es una desviación considerable. Aunque solo se presenten las horas empleadas y no un calendario final, éstas han sido posibles gracias al puente de diciembre y a los fines de semana.

Finalmente, lo que más se podría destacar como resultado satisfactorio, es el esfuerzo personal aplicado para que, partiendo de un desconocimiento casi absoluto del sistema en el que se apoya la aplicación "*Firebase*" se haya conseguido una aplicación con un acabado bastante notable. Además, el potencial que ofrece la API y las constantes actualizaciones, motivan al desarrollador a pensar constantemente en nuevas ideas para desarrollar.

Este proyecto ha sido una clara puesta en práctica de la aplicación de las aptitudes adquiridas a lo largo del máster y el reto de enfrentarse a un nuevo problema y saber solucionarlo.

7.2 Trabajos futuros

Una vez finalizada la aplicación "*socialWar*", tenemos una aplicación completa y funcional. Pero aun así puede verse mejorada con nuevas funcionalidades. En este capítulo se explicarán qué partes de la aplicación pueden ser mejoradas y cómo hacerlo.

En un futuro inmediato se debería publicar la aplicación en la tienda de aplicaciones más conocida de Android, "*Google Play*", para permitir a los usuarios descargarla. También sería una tarea imprescindible hacer la aplicación "*socialPlan*" compatible con los dispositivos más pequeños presentando una interfaz más atractiva y sin cortes, para poder abarcar satisfactoriamente el mayor número de usuarios potenciales que puedan acceder a la aplicación. Ésta es una tarea sencilla, ya que "socialPlan" se ha hecho con los layouts de Android con lo que creando los propios para una determinada resolución se podría resolver el problema. Además, se ha hecho uso de pesos de layouts, es decir,

se divide la pantalla del dispositivo en porcentajes por lo que visualmente está bien, pero puede haber problemas a la hora de mostrar algunos layouts.

A medio plazo, sería interesante añadir algunas de las siguientes funcionalidades al proyecto:

- *Añadir amigos, y poder contactar directamente con ellos*. El chat se haría de forma similar a los mensajes enviados en los grupos y eventos.
- *Buscar por localidad, fecha, nombre del plan y/o tags*. Aunque forme parte de los objetivos previstos, por tiempo no se ha podido desarrollar. La implementación es sencilla, ya que la interfaz de búsqueda está hecha, y solo habría que añadir nuevos parámetros.
- *Mejorar la librería de búsqueda de lugares*. Se ha utilizado una librería propia de "*Firebase*" que es a nivel mundial, sin embargo, es un poco limitada a la hora de buscar y seleccionar lugares que sean recintos, como pueden ser teatros, salas de conciertos, etc.
- *Enviar notificaciones a usuarios añadidos a un grupo o evento*. Ahora mismo solo se envían notificaciones al usuario que ha creado el plan, de la misma forma se recuperaría la lista de los usuarios añadidos y por cada comentario se enviaría a los usuarios añadidos una notificación.
- *Funcionalidad de las notificaciones.* Aunque se han dividido las notificaciones en diferentes tipos (se han puesto diferentes colores en el menú de "*Mis notificaciones*") cuando se pulsa una notificación tan solo abre la aplicación y hay que buscar el plan para poder ver los cambios en el mismo. Debería añadirse que cuando se abra una notificación se acceda a la interfaz que ha desembocado esa notificación.
- *Añadir fecha desde/hasta en la creación de eventos*.
- *Añadir verificación de correo electrónico al crear un nuevo usuario*. "*Firebase*" proporciona este servicio.
- *Añadir más idiomas a la aplicación con los "strings.xml".*
- *Mejorar de forma progresiva las interfaces de usuario.*
- *Tutorial en la aplicación.* Se realizará con diferentes utilidades que permiten la primera vez que entras en la aplicación resaltar determinados aspectos de la misma para poder conocer su uso.
- *Realizar mantenimientos periódicos para tener adaptada la aplicación a nuevas versiones del sistema operativo.*
- *Promoción y marketing de la aplicación.* Se hará a través de redes sociales.

Estas mejoras no han sido implementadas en este proyecto por la falta de tiempo o debido a la gran dificultad que suponían. Todas estas nuevas funcionalidades y mejoras podrían proporcionarse al usuario de forma progresiva en el tiempo, impidiendo que la aplicación caiga en el olvido, manteniendo a los usuarios enganchados y consiguiendo así aumentar el valor que estos tienen sobre la aplicación.

8. GLOSARIO

- ***Activities***: Componente de la aplicación Android que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción.
- ***.apk***: Instalable de una aplicación para Android.
- ***API***: Application Programming Interface. Es un conjunto de funciones y procedimientos ofrecidos por diferentes plataformas para ser usados por otros softwares.
- ***EDP***: Estructura de descomposición del proyecto.
- ***Fragments***: Representación de una parte de la interfaz de usuario en una *Activity*.
- ***FK***: Foreign Key
- ***ID***: Identificador
- ***iOS***: Iphone OS (sistema operativo de Apple).
- ***Layouts***: archivos de Android que sirven para especificar la interfaz visual que ve el usuario en una aplicación.
- ***Listener***: Objeto que se utiliza para detectar eventos o acciones. Coloquialmente conocido como "*escuchador*".
- ***MVC***: Modelo-Vista-Controlador. Es un patrón de diseño de software que se utiliza para la implementación de sistemas dónde se requiera el uso de interfaces de usuario. Sirve para separar de una manera clara las 3 partes más importantes del software: los modelos o capa de datos, las vistas que permitirán la visualización de las interfaces y los controladores que contiene el código necesario para responder a las acciones o eventos que se solicitan en una aplicación.
- ***NI***: Número de interfaz.
- ***Node.js***: entorno de ejecución de Javascript.
- ***PK***: Primary Key
- ***TFM***: Trabajo final de máster.
- ***UML***: Unified Modeling Language. Es un lenguaje de modelado unificado y visual que sirve para la creación de diagramas que presenten la arquitectura, el diseño y la implementación de diferentes softwares.
- ***.zip***: Extensión asociada a archivos comprimidos.

9. BIBLIOGRAFÍA

9.1 Libros

Requena Santos, Félix (2003). *Análisis de redes sociales: Orígenes, teorías y aplicaciones*. Centro de investigaciones sociológicos.

9.2 Sitios Web

Gartner. [En línea]. [Consulta: 2 de octubre de 2017].
<http://www.gartner.com/newsroom/id/3725117>

Fever. [En línea]. [Consulta: 5 de octubre de 2017].
<https://play.google.com/store/apps/details?id=com.feverup.fever&hl=es>

MeetUp. [En línea]. [Consulta: 5 de octubre de 2017].
<https://www.meetup.com/es-ES/>

Facebook. [En línea]. [Consulta: 7 de octubre de 2017].
<https://www.facebook.com>

Twitter. [En línea]. [Consulta: 7 de octubre de 2017]. <https://twitter.com>

Instagram. [En línea]. [Consulta: 7 de octubre de 2017].
<https://www.instagram.com>

Instagram. [En línea]. [Consulta: 7 de octubre de 2017].
<https://www.linkedin.com>

GoogleSheet. [En línea]. [Consulta: 12 de octubre de 2017].
<https://docs.google.com/spreadsheets/u/0/>

User Interface Guidelines. [En línea]. [Consulta: 15 de octubre de 2017].
<https://developer.android.com/guide/practices/ui_guidelines/index.html>

Ley de Fitts. [En línea]. [Consulta: 15 de octubre de 2017].
<http://emmallensa.com/la-ley-fitts-aplicada-dispositivos-moviles/>

JustInMind. [En línea]. [Consulta: 15 de octubre de 2017].
<https://www.justinmind.com/>

BIME Live. [En línea]. [Consulta: 22 de octubre de 2017].
<http://bime.net/live/es-es/inicio>

Guggenheim. [En línea]. [Consulta: 22 de octubre de 2017].
<https://www.guggenheim-bilbao.eus>

Firebase. [En línea]. [Consulta: 2 de noviembre de 2017].
<https://firebase.google.com/>

10. ANEXOS

A continuación, se desarrollará el anexo mencionado a lo largo de la documentación que se corresponderá con los casos de uso extendidos. Además de dos anexos con el manual de usuario y el manual de instrucciones para usar la aplicación.

10.1 Anexo I: Casos de uso extendidos

En este anexo se expondrán los casos de uso extendidos de la aplicación.

10.1.1 C.U. Registrarse

Descripción: El usuario se registra en la aplicación.

Precondición: No estar registrado en el sistema o haberse conectado por primera vez. Estar en la pantalla de registro.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón “Registrarse”.

2. El usuario introduce los datos obligatorios para poder registrarse.

3. El usuario pulsa en el botón “Guardar”.

4. EXTENDS C.U. Ir a Home.

Postcondición: Se crea un nuevo usuario y se guarda en la base de datos del dispositivo.

10.1.2 C.U. Identificarse

Descripción: El usuario se loguea en la aplicación.

Precondición: No estar registrado en el sistema o haberse conectado por primera vez. Estar en la pantalla de registro.

Requisitos no funcionales:

Flujo de eventos:

[Si el usuario elige “Iniciar sesión”]

1a. El usuario introduce los datos necesarios para iniciar sesión.

2a. El usuario pulsa el botón “Guardar”

3a. C.U. Ir a Home.

[Si el usuario elige “Iniciar sesión con Facebook”]

1b. El usuario aceptar las condiciones de Facebook para iniciar sesión.

2b. C.U. Ir a Home.

Postcondición: Se guarda el usuario en la base de datos del dispositivo.

10.1.3 C.U. Ir a Home

Descripción: El usuario consulta los eventos y grupos cercanos a su localidad.

Precondición: Estar identificado en la aplicación.

Requisitos no funcionales: Los eventos y los grupos están ordenados por la fecha en la que se van a realizar.

Flujo de eventos:

1. El usuario pulsa sobre el botón de "*Menú*".
2. El usuario pulsa sobre la opción "*Home*".
3. INCLUDE C.U. Actualizar datos.
4. Se muestra al usuario los eventos cercanos a su localidad.
5. El usuario pulsa sobre un evento o grupo determinado.
6. INCLUDE C.U. Ver detalle grupo/evento
7. Se muestra el detalle de un evento o grupo.

Postcondición:

10.1.4 C.U. Buscar

Descripción: El usuario busca y filtra los eventos y grupos según unos criterios.

Precondición: Estar identificado en la aplicación. Estar en la vista de Home.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón de "*Buscar*".
2. Se muestra la vista de filtrado.
3. El usuario introduce los datos por los que quiere filtrar.
4. El usuario pulsa en el botón "Buscar".
5. INCLUDE C.U. Ir a home
6. El usuario pulsa sobre un evento o grupo determinado.
7. INCLUDE C.U. Ver detalle grupo/evento
8. Se muestra el detalle de un evento o grupo.

Postcondición:

10.1.5 C.U. Crear Grupo/Evento

Descripción: El usuario crea un nuevo grupo o evento.

Precondición: Estar identificado en el sistema. Estar en una interfaz que permita crear un nuevo grupo o evento ("*Home*", "*Red*")

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón de "+".
2. El usuario introduce los datos obligatorios para crear un nuevo evento o grupo.
3. Se crea un nuevo grupo o evento.
4. INCLUDE C.U. Actualizar datos.
5. INCLUDE C.U. Ver detalle grupo/evento
5. Se muestra al usuario el detalle del grupo o evento creado

Postcondición: Se registra un nuevo evento o grupo en el sistema.

10.1.6 C.U. Unirse a grupo

Descripción: El usuario se une a un grupo creado.

Precondición: Estar identificado en el sistema. Estar en la vista de detalle de un grupo.

Requisitos no funcionales: El grupo no puede estar completo.

Flujo de eventos:

1. El usuario pulsa sobre el botón "Unirse y confirmar asistencia" de un grupo.
2. INCLUDE C.U. Actualizar datos
3. Se muestra el detalle del grupo con el usuario añadido al mismo.

Postcondición: Se actualiza el grupo.

10.1.7 C.U. Comentar grupo

Descripción: El usuario comenta un grupo.

Precondición: Estar identificado en el sistema. Estar en la vista de detalle de un grupo en la zona de comentarios y haber confirmado asistencia.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el apartado de comentarios.
2. El usuario introduce el comentario.

3. INCLUDE C.U. Actualizar datos

3. Se muestran los comentarios del grupo.

Postcondición: Se actualiza el grupo con un nuevo comentario.

10.1.8 C.U. Borrar evento/grupo

Descripción: El usuario borra un evento o un grupo.

Precondición: Estar identificado en el sistema. Estar en la vista de detalle de un grupo o evento o estar en la interfaz “Mi red”.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón “*Borrar*”.

2. El usuario confirma que desea borrar el evento o grupo.

3. INCLUDE C.U. Actualizar datos

[Si es un evento]

4a. Se envía una notificación a los usuarios que tuviesen el evento en su red.

[Si es un grupo]

4b. Se envía una notificación a los usuarios que estuviesen unidos al grupo.

Postcondición: Se borra el grupo o evento del sistema.

10.1.9 C.U. Ir a perfil

Descripción: El usuario consulta sus datos de perfil.

Precondición: Estar identificado en el sistema.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón de “*Menú*”.

2. El usuario pulsa sobre la opción “*Perfil*”.

3. Se muestra al usuario la información de su perfil.

Postcondición:

10.1.10 C.U. Modificar perfil

Descripción: El usuario modifica sus datos de perfil.

Precondición: Estar identificado en el sistema. Estar en la interfaz “*Perfil*”

Requisitos no funcionales:

Flujo de eventos:

1. El usuario introduce los datos a modificar.

2. INCLUDE C.U. Actualizar datos

3. Se modifican y muestran los datos introducidos por el usuario.

Postcondición: Se modifica el perfil del usuario.

10.1.11 C.U. Añadir/Quitar evento a mi red

Descripción: El usuario añade/quita un evento a/de su red.

Precondición: Estar identificado en el sistema. Estar en la "*Home*" o en el detalle de un evento.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón "*Me gusta*".

2. INCLUDE C.U. Actualizar datos

3. Se actualizan los eventos del usuario.

Postcondición: Se añade/elimina un nuevo evento a/de la red de un usuario.

10.1.12 C.U. Ir a red

Descripción: El usuario consulta su red.

Precondición: Estar identificado en el sistema.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón de "*Menú*".

2. El usuario pulsa sobre la opción "*Red*".

3. INCLUDE C.U. Actualizar datos

4. Se muestra al usuario los grupos creados por él mismo.

[Si pulsa en el botón eventos]

5a. Se muestra al usuario los eventos creados por él mismo.

[Si pulsa en el botón favoritos]

5b. Se muestra al usuario los eventos añadidos a su red.

[Si pulsa en el botón cerrados]

5c. Se muestra al usuario los grupos creados por él mismo que ya hayan pasado.

6. El usuario consulta el detalle de un grupo o evento.

[Si el evento o grupo ha sido creado por el usuario]

7a. INCLUDE C.U. Ver detalle grupo/evento

8a. El usuario realiza modificaciones en el grupo/evento

9a. INCLUDE C.U. Actualizar datos

[Si el evento o grupo NO ha sido creado por el usuario]

7b. INCLUDE C.U. Ver detalle grupo/evento

Postcondición: Se actualizan los datos del grupo/evento

10.1.13 C.U. Ir a notificaciones

Descripción: El usuario consulta sus notificaciones.

Precondición: Estar identificado en el sistema.

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón de "*Menú*".
2. El usuario pulsa sobre la opción "*Mis notificaciones*".
3. INCLUDE C.U. Actualizar datos
4. Se muestra al usuario las notificaciones pendientes y ya leídas.
6. El usuario consulta el detalle de una notificación.
7. INCLUDE C.U. Ver detalle grupo/evento

Postcondición: La notificación leída actualiza su estado.

10.1.14 C.U. Eliminar/Cerrar sesión

Descripción: El usuario cierra sesión o elimina su usuario.

Precondición: Estar identificado en el sistema. Estar en la interfaz de "*Mi perfil*".

Requisitos no funcionales:

Flujo de eventos:

1. El usuario pulsa sobre el botón de "Cerrar".

[Si el usuario elige cerrar sesión]

2a. El usuario confirma que desea cerrar sesión.

3a. INCLUDE C.U. Actualizar datos

4a. Se borra del dispositivo los datos del usuario.

5a. Se le muestra al usuario la pantalla de registro.

[Si el usuario elige eliminar el perfil]

2a. El usuario confirma que desea eliminar el perfil.

3a. INCLUDE C.U. Actualizar datos

4a. Se borra del dispositivo y del sistema los datos del usuario.

5a. Se avisa a los usuarios registrados en algún evento del usuario que el evento o grupo ha sido borrado.

6a. Se le muestra al usuario la pantalla de registro.

Postcondición: Se borra el usuario y los datos asociados al mismo.

10.1.15 C.U. Ver detalle grupo/evento

Descripción: Ver detalle grupo/evento es un caso de uso interno y generalizado, el cual ayuda a no realizar repeticiones en el futuro. El usuario ve el detalle de un grupo o evento.

Precondición: Precondiciones del caso de uso inicial.

Requisitos no funcionales: Se parte del caso de uso padre.

Flujo de eventos:

1. Se ve el detalle de un grupo o evento.

Postcondición:

10.1.16 C.U. Actualizar datos

Descripción: Actualizar Datos es un caso de uso interno y generalizado, el cual ayuda a no realizar repeticiones en el futuro. Actualiza los datos.

Precondición: Precondiciones del caso de uso inicial.

Requisitos no funcionales: Se parte del caso de uso padre.

Flujo de eventos:

1. Se actualizan los datos para el usuario.

Postcondición: Se actualiza en la base de datos y en la aplicación los datos a mostrar y guardados por el usuario.

10.2 Anexo II: Manual de usuario

Para comenzar a utilizar la aplicación es necesario tener la conexión de Internet activada.

Al entrar en la aplicación por primera vez el usuario debe seleccionar la forma de registro con la que desea acceder a la aplicación:

- *Crear un nuevo usuario*: el usuario deberá introducir como mínimo los datos obligatorios necesarios para crear un nuevo perfil: el nombre de usuario, el correo electrónico y una contraseña de 6 caracteres. Además, podrá seleccionar una foto de perfil o introducirla más adelante.
- *Identificarse*: si el usuario ya se ha registrado previamente en la aplicación, podrá volver a acceder introduciendo sus credenciales.
- *Identificarse con Facebook*: el usuario puede utilizar la red social Facebook para crear un nuevo usuario o identificarse en la aplicación.

Este registro solo es necesario la primera vez, a partir de haber creado de forma exitosa un usuario, el usuario accederá a la pantalla de "*Home*" automáticamente.

En esta pantalla, podrá ver los eventos informativos más cercanos a su localidad (en un rango de 100 kilómetros por defecto, que podrá cambiar pulsando en el botón búsqueda del menú superior). Por lo que nada más entrar se le pedirá al usuario que acepte los permisos de localización. Si no lo hiciese, no se mostraría ningún evento o grupo.

Para poder empezar a usar la aplicación, el usuario puede dar "*me gusta*" a los eventos y ver más detalles pulsando en ellos. Esto es común para todos los listados que hay en la aplicación, se pueden ver los detalles pulsando en un evento o grupo en concreto.

Para adherirse a un grupo, el usuario debe entrar al detalle del mismo, y pulsar el botón "*Unirse y confirmar asistencia*" el cual solo será visible si el usuario no es el que ha creado el grupo y si el grupo no está completo.

Los usuarios pueden crear planes pulsando el botón ***"+"*** en la parte inferior derecha de la interfaz y añadiendo los datos obligatorios del mismo. La diferencia entre el grupo y el evento es el botón de selección de "*Tipo de plan*" en la parte inferior de la interfaz. Si el usuario lo pulsa estará creando un grupo, si no, un evento informativo.

Si un usuario quiere borrar un grupo o un evento creado por él mismo, deberá acceder a la opción de menú "*Mi red*", identificar el plan a borrar y con un gesto del dedo llevando el plan hacia la izquierda, el grupo quedará eliminado.

10.3 Anexo III: Manual de instrucciones

La aplicación se puede instalar de varias formas: con la .apk que se adjunta en el proyecto o a través de algún simulador en Android Studio.

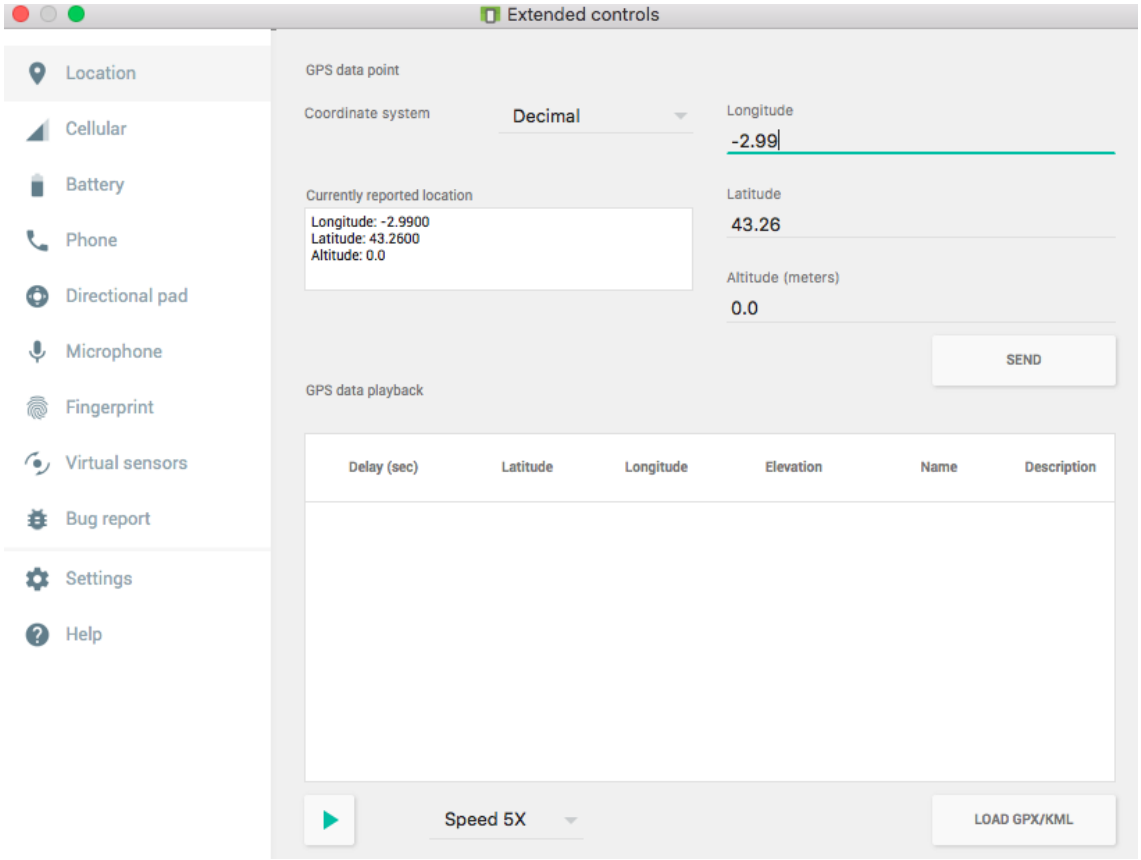
Por un lado, para instalar la aplicación a través de la .apk, se debe abrir la misma desde un dispositivo físico y pulsar el botón de instalar.

Por otro lado, si se desea probar la aplicación a través de Android studio:

- Se importa la aplicación con Android Studio: File – Open – socialPlan (carpeta contenedora del proyecto) – Open.
- Se configura un nuevo AVD (Android Virtual Device)  , y a continuación se pulsa en “Create new virtual device”.
- Se selecciona un dispositivo (por ejemplo, Nexus 5X).
- Se asocia al dispositivo una imagen de sistema (Por ejemplo, Nougat. Target 7.1.1 Google Play). Tener en cuenta en seleccionar siempre una imagen con Google Play, y que sea superior al API 19 de Android.
- Se pulsa en “Next”, y en la siguiente interfaz se puede configurar el dispositivo. Se dejan las opciones por defecto.
- Por último, se pulsa en el botón para ejecutar la aplicación  . Y se selecciona el dispositivo que se acaba de crear y se pulsa en “Ok”. A partir de aquí, se abrirá el simulador y se instalará la aplicación en el mismo abriéndola por defecto.

Se debe tener en cuenta que un simulador no cuenta con las mismas posibilidades que un dispositivo físico. Hay que simular la ubicación del dispositivo en el emulador, y lo mismo ocurre que la cámara del dispositivo.

Para simular la ubicación, habrá que acceder a más opciones en el simulador. Y simular la ubicación, enviándola al dispositivo.



Para simular la cámara, al registrar el emulador, se puede elegir en opciones avanzadas, que la cámara principal o la frontal sea una cámara simulada o la web cam del ordenador.