TFM-Desarrollo de Aplicaciones sobre Dispositivos Móvil

2014/15

MEMORIA

Master Universitario en Ingeniería Informática

Proyecto: App para promoción de artistas y grupos musicales

Tutores…: Ignasi Lorente Puchase
Jordi Almirall López

Fecha entrega: **09-01-2015**

CONTENIDO

FIGURAS

VISIÓN GENERAL

1 Introducción

1.1 Motivación

Este proyecto se engloba dentro del área de desarrollo de aplicaciones móviles, en concreto sobre la opción de Gestor de Mapas y GPS para la plataforma Android.

La idea general inicial se basa en desarrollar una app para dispositivos Android que permita promocionar a artistas y grupos musicales. Lo que se pretende es crear una red social especializada tipo consumidor/productor donde los artistas o grupos podrán publicar su proyecto musical y los fans o usuarios en general podrán seguir la trayectoria de cualquier grupo o artista. En consecuencia, esta app contemplará dos tipos de usuarios: activos, los que generan contenido y los pasivos que sólo consumirán servicios sin interactuar con el sistema.

Definiendo la app de una forma más concreta, los usuarios que sean artista o pertenezcan a un grupo musical podrán tener una red, una vez identificados, dónde: darse a conocer, publicar su discografía o proyecto musical, publicar su afiche (foto del artista o grupo), videos, bolos o actuaciones en próximas fechas (publicar una agenda de actuaciones o conciertos y los lugares en dónde se celebrarán). Por otro lado, cualquier usuario (identificado o no) puede: consultar la promoción de cualquier grupo, ver comentarios, fotos, videos y una agenda de actuaciones dónde podrá ver en un mapa la localización de los lugares dónde se celebrarán las actuaciones, conciertos o bolos de cada artista o grupo musical.

Precisamente en esta agenda de actuaciones se centrará la app de este TFM. Estos lugares, dónde se mostrarán las próximas actuaciones, se mostrarán como pines dentro del mapa, y al pulsar en los mismos, saldrá un cuadro de diálogo que permitirá realizar las acciones oportunas: ver que grupo o artista actúa, la dirección del evento, ver la ruta o rutas disponibles para acceder al punto determinado, …

1.2 Objetivos

Con este proyecto se pretende conseguir crear la aplicación descrita en el punto anterior, y además, ésta pueda ser de utilidad a muchos usuarios. Para llevarlo a buen término será imprescindible gestionar eficientemente el proyecto en sus diferentes fases, desarrollar las funcionalidades necesarias e investigar acerca de la tecnología a emplear.

Las funcionalidades a conseguir estarán divididas en dos grupos: funcionalidades para cualquier usuario sin necesidad de estar registrado y funcionalidades para usuarios registrados.

Funcionalidades para cualquier usuario que use la aplicación:

- Consultar datos de los artistas o grupos musicales.
- Consultar datos de promoción de los artistas o grupos musicales.
- Consultar próximos eventos que se realizarán en días concretos de los artistas o grupos musicales.
- Ver en un mapa eventos que se realizarán en una fecha concreta y su localización.
- Ver en un mapa los eventos y su localización que se realizarán en próximas fechas cerca de la localización del usuario en un radio determinado.
- Ver la ruta hacia el evento
- Ver en StreetView la zona dónde se realizará el evento.
- Compartir los eventos.
- Guardar los eventos en la agenda del dispositivo móvil.

Funcionalidades para los usuarios que se registran en el sistema:

- Dar de alta el artista o grupo musical
- Añadir y eliminar promociones
- Añadir y eliminar eventos

Respecto al estudio de las tecnologías a utilizar en este proyecto el objetivo es profundizar lo máximo posible sobre las tecnologías emergentes, como las tecnologías en la nube (BaaS/MBaaS), Servicios de Google Maps en su versión 2, SDK de Android, utilización de fragments, api support-android,…

Dentro del objetivo personal mi meta es poner en práctica los conocimientos adquiridos a lo largo del master, aplicando las mejores técnicas de desarrollo en la plataforma android y realizar una gestión correcta del proyecto con el fin de obtener un feedback con las lecciones aprendidas y asumirlos como guía en futuros proyectos.

1.3 Justificación

Uno de los motivos principales que me ha decantado realizar el TFM en el área de desarrollo de aplicaciones móviles ha sido el concepto de movilidad que existe en la actualidad. La gran difusión de este tipo de tecnologías poco a poco está cambiando, y siguen cambiando, la forma de vivir y relacionarse con los demás.

Elegir la plataforma Android en este TFM tiene su origen en mis conocimientos del lenguaje de programación Java y por la disponibilidad de dispositivos con este sistema operativo para poder realizar pruebas.

Otro aspecto que me ha impulsado a realizar este proyecto es el concepto de geolocalización. La Geolocalización está adquiriendo una importancia relevante en la actualidad debido a aplicaciones surgidas gracias a la combinación de: la tecnología de GPS, los dispositivos móviles y las redes sociales. Multitud de desarrollos incluyen mapas en los que los usuarios pueden localizar puntos de su interés (los mapas de Google son ampliamente utilizados para estos fines). Además, aplicaciones con gran éxito como FourSquare basadas en mapas combina el mismo concepto de geolocalización con el concepto de red social.

Todos estos factores han determinado la balanza en esta área de conocimiento para realizar este TFM.

1.4 Alcance

El alcance de este proyecto va a estar limitado a unas funcionalidades concretas para que pueda ser desarrollado en tiempo y forma a lo solicitado en este TFM. Dichas funcionalidades se describen en el apartado de **"***Especificación de requisitos***".** No obstante, este proyecto se podría complementar con una serie de servicios que lo podrían hacer atractivo y viable económicamente en un mundo empresarial: venta de entradas, merchandising, concursos, oferta de empleo, intercambio de material musical, mercadeo musical, publicidad,…En el apartado de geolocalización se podría ofertar más servicios: un servicio para poder ver en el mapa usuarios que van un día concreto a un mismo evento y ver que cerca están unos de los otros cerca del destino. Poder pasar la información del evento al navegador del coche…

Como se puede apreciar, la geolocalización y las redes sociales forman un binomio interesante que abren las puertas a la imaginación (tanto social como técnicas) y a diversas oportunidades de negocio.

1.5 Riesgos

Todo proyecto no está exento de riesgos y este no es menos. De todos los factores generales que pueden repercutir negativamente en el desarrollo de un proyecto, los más significativos que podrían afectar a éste serían:

- **Los relativos a la valoración incorrecta tanto del alcance como de la complejidad del proyecto:** Para reducir este riesgo, es conveniente ser metódico en todos los pasos a dar y estudiar detenidamente los conceptos a tratar en el proyecto, tanto a nivel teórico como a nivel práctico para intentar ajustar, en lo máximo posible, las funcionalidades a nivel cuantitativo y cualitativo, reduciendo de esta manera la complejidad global del proyecto sin perder usabilidad.

- **La gestión ineficiente del tiempo y del proyecto en sí:** Aparte del tiempo limitado del que se dispone, y que se pueda disponer al 100% del mismo, se tiene que contar con un tiempo extra para conocer la tecnología que se va a utilizar. Con el fin de prevenir este riesgo se realiza un planning marcando todas las entregas y trabajos a realizar marcando los hitos que permitirán llevar el proyecto a buen término. En esta gestión del tiempo también se tendrá en cuenta el tiempo necesario para el estudio e investigación de la tecnología a utilizar.

En cuanto a los riesgos específicos del PFM, añadiría otros dos factores:

- **Falta de compromiso y dedicación:** Aunque a primera vista este riesgo, personalmente, no parece muy significativo, sí hay que tenerlo muy en cuenta, ya que puede propiciar que la planificación no consiga los hitos en los tiempos establecidos y ello produzca problemas serios al final del proyecto. Por lo tanto, hay que evitar estancamientos innecesarios que puedan conducir a un desánimo peligroso donde el proyecto pueda verse afectado y de esta manera no cumplir con los hitos marcados.

- **Factores personales y profesionales**: Cuando cuentas con una familia y un trabajo, hay que priorizar las necesidades y anteponer unas tareas a otras. Las necesidades familiares van antes que las profesionales y las profesionales antes que cualquier otra. Esto implica reservar tiempo extra. Dicho de otro modo, hay que valorar el alcance con una pequeña holgura que nos permita poder hacer ciertos cambios en el planning (si se dieran las casuísticas antes mencionadas y estas sean compatibles) sin que ello pueda perjudicar alcanzar los hitos necesarios para culminar con éxito el proyecto.

2 Especificación de requisitos

A continuación se describen los requisitos que se tendrán en cuenta en el App del TFM.

2.1 Requisitos funcionales

Módulos principales	Funcionalidad requerida	Funcionalidad objetiva
Identificación y persistencia (BaaS)		
	Identificación en el sistema	Identificación de usuarios en el sistema para crear contenidos.
	Creación de usuarios	Creación de usuarios que acceden al sistema para producir contenido
	Creación de Comentarios promocionales	Creación de comentarios promocionales producidos por los artistas y grupos musicales creados en la red social
	Creación agenda de actuaciones	Relación de puntos de actuación de los grupos y artistas en un día concreto en un lugar determinado(fecha y lugar)
	Subida y bajada de recursos multimedia(foto)	Foto perteneciente al artista o grupo musical.
Gestión de Mapas (Google Maps)		
	Localizar Eventos (actuaciones, conciertos, bolos,..)	Localizar puntos de eventos o consulta de puntos cercanos. Compartir los eventos y guardarlos en la agenda del dispositivo
	Mostrar múltiples pines	Consultar puntos dados de alta por los artistas y grupos musicales
	Configurar diálogos de los pines	Proporcionar Información básica sobre los puntos (Nombre del grupo, fecha, local, dirección…)
	Aplicar acciones al diálogo de los pines	Ampliación de información
	Zoom	Realizar zoom sobre los mapas
	Distintas vistas de los Mapas	Cambiar las distintas formas de ver los mapas que ofrece Google
	Cómo llegar…	Obtener indicaciones de la ruta hasta un punto
	Cerca de mi…	Obtener puntos cerca de mi posición en base a una determinada distancia
Gestión de GPS		
	Gestión de conversiones entre coordenadas de geolocalización/direcciones y viceversa a partir de la API de Google Maps	Facilitar la ubicación de los puntos y su interpretación por parte de los usuarios

Figura 1: Requisitos funcionales

2.2 Requisitos no funcionales

Requisitos no funcionales	Descripción
Entorno de desarrollo	Como entorno de desarrollo de la App se utilizará Eclipse y se instalarán los plugins necesarios para el desarrollo de aplicaciones nativas Android.
Persistencia de datos	Los datos que utilice el App de la aplicación se almacenarán en un sistema Baas en la nube, para ello, se hará uso de un backend en el que se almacenarán los objetos necesarios.
Lenguaje de programación	Se desarrollará en la plataforma nativa de Android y se utilizará el lenguaje de programación Java.
Librerías	Se utilizará el framework de desarrollo nativo para Android, así como las librerías adicionales para el acceso a la API de Maps y al backend.
Entorno de ejecución	Se proporcionará un ejecutable en formato .apk para poder instalar el App en dispositivos Android.
Pruebas	Se realizarán las pruebas unitarias y los test de usuario que resulten necesarios para evaluar el buen funcionamiento del App.
Documentación	Se proporcionará una memoria final y un vídeo de presentación del proyecto.

Figura 2: Requisitos no funcionales

3 Metodología de desarrollo (DCU Ágil)

En este proyecto se usará DCU ágil que combina el proceso típico del Diseño Centrado en el Usuario y Agile Software Development.

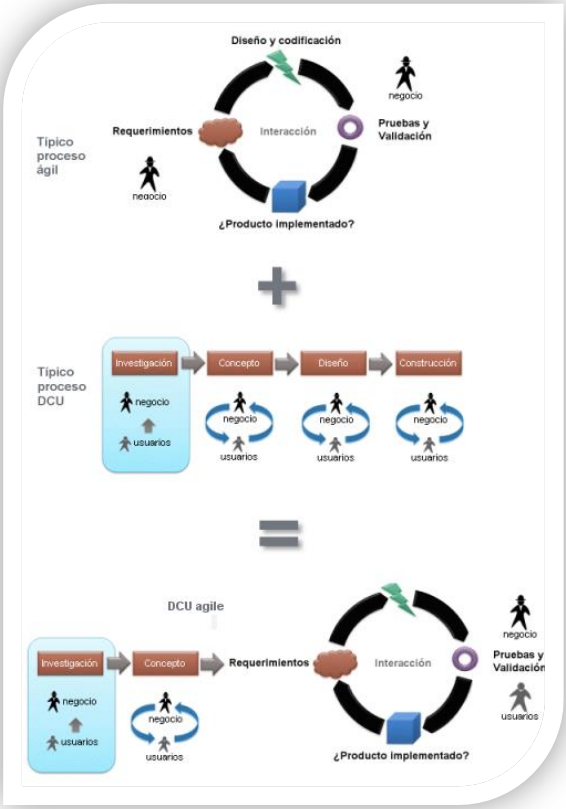


Figura 3: Proceso iterativo del DCU según ISO 13407; extraído de Sergeev (n.d.)

Actualmente la experiencia de usuario y la usabilidad son necesarias, por no decir imprescindibles, para el éxito de cualquier tipo de producto o software.

El diseño centrado en el usuario (DCU) es una aproximación al diseño de productos y aplicaciones que sitúa al usuario en el centro de todo el proceso. Se puede decir que el DCU es una filosofía cuya premisa se basa en que hay que tener en cuenta al usuario en todas las fases del diseño para garantizar el éxito de un producto, ya que no es posible entender el producto desvinculado de su uso, su contexto, o de las necesidades y motivaciones del usuario final. Por lo tanto, el DCU es un proceso cíclico, donde la usabilidad del diseño es evaluada de forma iterativa y mejorada incrementalmente.

Por otro lado, dada la evolución constante de la industria, se requieren ciclos de vida realmente cortos para poder dar salida a las aplicaciones a tiempo. Los métodos ágiles suelen ser muy adecuados para el desarrollo de aplicaciones móviles debido a la alta

volatidad del entorno dónde los cambios de entornos de desarrollo, nuevos terminales y nuevas tecnologías emergen a un ritmo más elevado. Este método se adapta bien en proyectos relativamente pequeños donde los equipos no suelen ser muy grandes y el tipo de aplicaciones desarrolladas no suelen ser de alta criticidad, dado que suelen ser aplicaciones para entretenimiento o de gestión empresarial no crítica.

Por ello, el agregar el modelo de desarrollo ágil se intenta conseguir, entre otras cosas, entregar algo lo más pronto posible y evitar problemas originados por cambios de requisitos y así mitigar los riesgos. Para conseguirlo hay que darle más prioridad a la entrega del producto que a la documentación y basarse en las pruebas de la aplicación, pruebas que a menudo se automatizan.

3.1 Equipo (Roles)

El proyecto contará con los siguientes roles como participantes del mismo.

Roles	Descripción
Director del proyecto	Encargado de la dirección del proyecto (alumno)
Diseñador	Encargado del diseño
Programador	Encargado de la programación e implementación del proyecto
Clientes	Usuarios finales de la aplicación: Testers (probadores) y consultores del proyecto

Figura 4: Roles

PLANIFICACIÓN Y VIABILIDAD

4 Planificación

En el siguiente diagrama de Gantt se muestra de forma resumida la planificación del proyecto, sus tareas, los participantes y las fechas clave.

4.1 Cronograma

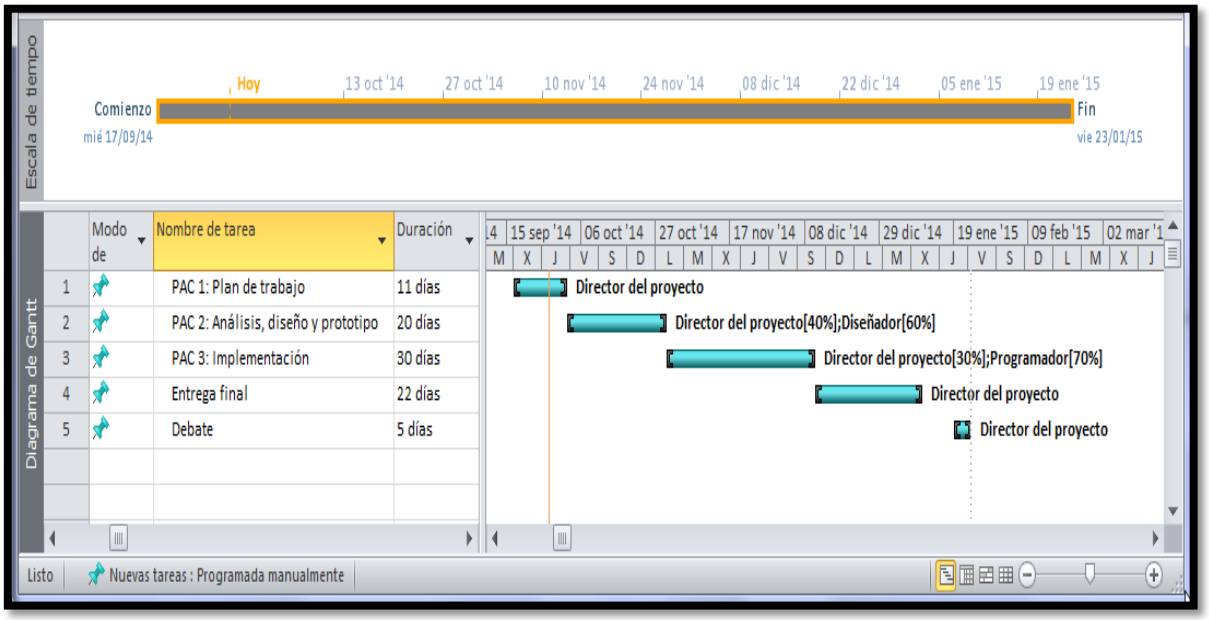


Figura 5: Cronograma

4.2 Tareas (Hitos)

Nombre de tarea	Duración	Inicio	Fin	Predecesoras	Recursos
PAC 1: Plan de trabajo	11 días	Mie 17/09/14	Mie 01/10/14	Director de Proyecto	PAC 1: Plan de trabajo
Tarea	Definir el proyecto y realizar su planificación inicial TFM.				
Hito	Plan de trabajo, PAC 1.				
Descripción	El plan de trabajo servirá como guía para el resto del desarrollo.				
PAC 2: Análisis, diseño y prototipo	20 días	Jue 02/10/14	Mie 29/10/14	1	Director de Proyecto [40%] Diseñador [60%]
Tarea	Realizar el análisis de tareas, diseño y prototipo acorde a la metodología DCU.				
Hito	Documentación que incluye perfiles de usuario, contexto de uso, análisis de tareas, escenarios de uso, diagramas de flujo de interacción y un prototipo de alto nivel.				
Descripción	La entrega de esta documentación facilitará cumplir los objetivos finales del proyecto, ya que se estudian los potenciales clientes del proyecto, se analizan todas las tareas detalladamente y se				

	obtiene un prototipo de alta fidelidad del proyecto. Todas estas tareas sirven como entrada de datos para la siguiente fase.				
PAC 3: Implementación	30 días	Jue 30/10/14	Mie 10/12/14	2	Director de Proyecto [30%] Programador [70%]
Tarea **Hito** **Descripción**	Implementar la solución del proyecto y creación de documentación complementaria. Código fuente, instalables y documentación complementaria. Fase final del proyecto en la que se obtiene el producto final. Puede requerir iterar con las fases anteriores en caso de detectar problemas en el producto.				
Entrega final	22 días	Jue 11/12/14	Vie 09/01/15	3	Director de Proyecto
Tarea **Hito** **Descripción**	Finalizar el proyecto y la documentación. Memoria y video de presentación del proyecto. En esta fase se finaliza la fase anterior (en caso de no haberlo hecho) y se presenta al público objetivo el producto.				
Debate	5 días	Lun 19/01/15	Vie 23/01/15	4	Director de Proyecto
Tarea **Hito** **Descripción**	Debatir sobre el proyecto y trabajos entregados. Sin entregables Se incluye en la planificación pero no se tiene en cuenta en el coste estimado del proyecto.				

Figura 6: Tareas (Hitos)

5 Viabilidad

Una vez identificado el proyecto se analiza la viabilidad conforme a tres niveles: técnico, operativo y económico.

A nivel técnico el proyecto es viable. Se utiliza software y aplicaciones que ofrece el mercado: SDK de Android, Api de Google Maps, IDE Eclipse con los plugins necesarios y el Software de Backend (BaaS). Para el hardware se dispone de un portátil y un dispositivo móvil.

A nivel operativo, y teniendo en cuenta los posibles riesgos, cabe la posibilidad de que se incremente el tiempo necesario para la finalización en el plazo estipulado, pudiendo de esta forma influir negativamente en la viabilidad del proyecto.

A nivel económico, y siempre dentro de un contexto académico, resulta viable. Al disponer del hardware necesario y hacer uso de software gratuito o de prueba, sólo existe el coste operacional. Sin embargo, en el contexto empresarial, hay que tener en cuenta otros factores que aquí no tenemos definidos y, que si realizamos el cálculo puro y duro de precio por horas invertidas sin tener en cuenta la proyección de negocio y la explotación del app, daría un resultado descompensado y poco viable. A pesar de ello, se realiza una aproximación del coste que supondría dentro del mundo empresarial, sin que con ello determine la viabilidad del proyecto económicamente.

5.1 Coste

Aproximación del coste operacional dentro del mundo empresarial:

Precio/Hora		
Director de proyecto	Diseñador	Programador
60€	30€	25€

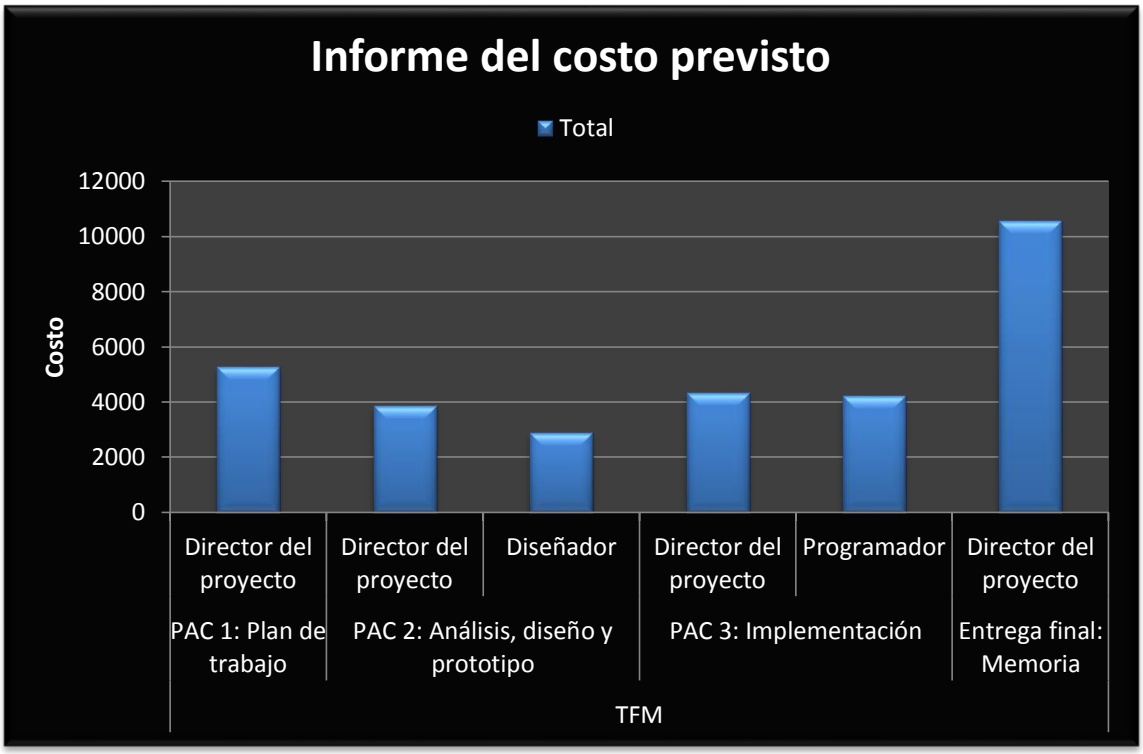


Figura 7: Tabla de costes

El coste total asciende 31.080 €.

ANÁLISIS, DISEÑO Y EVALUACIÓN

6 Análisis

Lo que se pretende en esta fase es conocer las características de los usuarios, sus necesidades y objetivos, así como el contexto de uso, con el fin de poder detectar las funcionalidades que debe tener la app para satisfacer a sus usuarios.

6.1 Usuarios y contexto de uso

Son muchas las preguntas que se nos puede ocurrir a la hora de analizar los usuarios que usarán la aplicación: ¿Quiénes son los usuarios? ¿Qué tareas tendrán que hacer? ¿Por qué utilizan el sistema? ¿Cómo utilizan el sistema? ¿Para qué utilizarán la aplicación? ¿Qué les crea satisfacción y qué no les agrada? ¿Qué dispositivos utilizan? ¿Utilizan alguna aplicación similar?...

Aunque estas preguntas son muy genéricas, sirven de punto de partida para determinar tres aspectos fundamentales:

- Conocer los usuarios potenciales.
- Conocer el contexto de utilización.
- Conocer las tareas que realizan.

6.1.1 Indagación: investigación y requisitos de usuario

Para investigar y conocer estos aspectos se ha usado varias técnicas de indagación: investigación contextual, entrevistas y análisis competitivo.

Estas técnicas permiten valorar y obtener, por una parte, la motivación del usuario mientras interactúa con un sistema similar al que se va a diseñar (contextual Enquiry), y por otro lado, una comprensión en profundidad sobre las necesidades, preferencias y experiencias de los usuarios con el sistema a desarrollar (entrevistas).

La justificación del uso de estas técnicas tiene su base en el contexto de uso de este aplicativo, ya que va dirigido a un público que tendrá un afín común dentro del sector espectáculos y entretenimiento. Por ello, las entrevistas están enfocadas a descubrir elementos que, diferentes perfiles de usuarios, pueden aportar en la definición de funcionalidades necesarias básicas a cubrir.

La investigación contextual permite observar distintos perfiles de usuario y valorar la aceptación y motivación que este producto puede aportar en su contexto de uso. El análisis competitivo

trata de analizar productos similares al que se está diseñando con el objetivo de conocer las expectativas de los usuarios, las funcionalidades básicas o comunes, estudiar las interfaces…

Por todo ello, a los usuarios se le ha solicitado analizar dos aplicaciones ya existentes de gran difusión: Google Maps y FourSquare para Android, con el objetivo de identificar y proponer funcionalidades que deberían integrarse en la aplicación.

Las tareas que se han solicitado ejecutar a los usuarios se centraron, exclusivamente, en averiguar el desenvolvimiento y aceptación que mostrarían ante una situación habitual en el uso de herramientas de localización y búsqueda de puntos de información en un mapa, ya que la aplicación que se desarrollará tiene un porcentaje alto de uso de este tipo de tecnología, y las observaciones que se obtengan de esta pruebas ayudará a detectar las funcionalidades y diseño más acorde a las necesidades del usuario.

Los dispositivos sobre el que se han realizado las pruebas han sido un Smartphone y una Tablet Android, ambas con una versión del sistema operativo Android 4.1.2 y 4.4.3 respectivamente. También disponen de GPS y conexión a Internet móvil activa.

Las tareas propuestas fueron las siguientes:

Prueba	Tarea	Descripción
Prueba 1	Ubicación actual	Realizar la localización de su ubicación actual
Prueba 2	Consultar evento	Consultar los datos de un punto cercano (evento musical). El punto es desvelado en el momento de realizar la prueba.
Prueba 3	Ruta al destino	Localizar la ruta más idónea desde la ubicación actual y el punto de destino

Figura 8: Pruebas y tareas

6.1.2 Perfiles de usuario

Se ha intentado reunir un variado grupo de usuarios con unas características de perfiles variados con el objetivo de agrupar y determinar los perfiles de usuarios potenciales que usarán la app.

Las diferentes tareas que se llevan a cabo, están orientados a la obtención de información para definir el producto que ofrecerá un servicio determinado. Los usuarios involucrados en las pruebas se muestran en "*Anexo I: Perfiles de usuario y resultados"* junto con el resultado obtenido.

Como resultado de la investigación, se puede concluir que la segmentación de usuarios por nivel de experiencia con las aplicaciones móviles no sería efectiva. Por otro lado, es necesaria una experiencia mínima con el uso de la tecnología para el uso de la aplicación a desarrollar, a pesar de ello, se ha podido apreciar que los que no tienen experiencia en este tipo de aplicaciones, rápidamente se hacen con el control y lo encuentran amigable.

Demográficamente la idea inicial no se ha definido por lo que en principio no se segmenta usuarios de acuerdo a este criterio. Sin embargo, sí se apreció una posible segmentación en el interés que han presentado diversos usuarios. La mitad de los usuarios encuentran las pruebas muy útiles para buscar puntos de interés, y sobre el contexto de la aplicación a desarrollar la

contemplan como una buena herramienta de búsqueda de eventos musicales; no se apreció interés en aportar información que fuera relevante para el sistema.

Otros en cambio se interesaron por la funcionalidad de poder publicar contenidos en la aplicación, ya sea porque formaban parte de un grupo musical o conocían a alguien que formaba parte de otro.

Por lo tanto, y en base a esta segmentación, los perfiles de usuario potenciales localizados se han clasificado de la siguiente manera:

Perfil de usuario	**USUARIO ACTIVO**
Contexto de uso	Usuarios habituales de la aplicación. Añaden todo tipo de información al sistema. Normalmente será un usuario que pertenezca a un grupo musical que necesite promocionar su grupo. Para sus fans, publicarán comentarios de sus actividades. Publicarán la agenda de sus actuaciones o eventos para que sus seguidores y público en general puedan conocer sus puntos de interés de sus actuaciones en un mapa, el cual les informará detalladamente de su ubicación.
Análisis de las tareas	Realizarán todas las tareas que permita la aplicación: consultar y añadir comentarios, grupos musicales y el calendario de eventos; también podrán consultar los eventos en un mapa dónde le indicará en qué lugar se realizará.
Funcionalidades adicionales	Tendrán que registrarse en el sistema para poder publicar cualquier contenido.

Figura 9: Usuario Activo

Perfil de usuario	**USUARIO PASIVO**
Contexto de uso	Hacen uso de la aplicación esporádicamente y consultan la información que está disponible. No comentan ni añaden nueva información. Los contextos de uso pueden ser: usuarios que quieren consultar eventos que tendrán lugar en un determinado periodo cerca de la ubicación del usuario. Fans, amigos que quieran seguir la información de sus grupos favoritos. Usuarios que quieren consultar el panorama musical activo en una determinada área de acción.
Análisis de las tareas en la aplicación	Las tareas que realizarán serán la de consultar las publicaciones creadas por los usuarios registrados en el sistema. Podrán consultar comentarios, grupos musicales y el calendario de eventos; también podrán consultar los eventos en un mapa dónde le indicará en qué lugar se realizará.
Funcionalidades adicionales	Este tipo de usuario no tendrá que registrarse en el sistema, ya que no publicará nada, sólo consultará.

Figura 10: Usuario Pasivo

7 Diseño

Los perfiles de usuario, los personajes o personas y los escenarios son técnicas que nos acercan a los usuarios y a sus motivaciones, objetivos y situaciones de uso. Son técnicas que sirven para entender y analizar los usuarios y el uso que hacen de los sistemas interactivos.

En la fase anterior de análisis de usuarios y la definición de requisitos se representaron los perfiles de usuario que sirven para elaborar los personajes y los escenarios necesarios para iniciar las actividades de diseño propiamente dichas.

7.1 Diseño conceptual

Con la información recopilada en la fase anterior se elabora los escenarios de uso que describirán, desde el punto de vista del usuario, como se utilizará la aplicación móvil en un contexto concreto. En esta fase se utilizan personajes y escenarios.

Los personajes son descripciones de un usuario arquetípico que puede servir como guía en el proceso de diseño. Los escenarios son descripciones de las acciones necesarias para realizar acciones específicas y pueden incluir la descripción del comportamiento en una situación dada; incluyen el contexto, los actores, los objetivos, la secuencia y el resultado. Con ello se pretende descubrir objetivos, funcionalidades y deseos de los usuarios en un contexto y situación determinada.

Una vez definidos y elaborados los escenarios de uso, se podrán determinar las necesidades de los usuarios y de diseño; principalmente, en la conceptualización de la estructura y los flujos de interacción de la aplicación.

7.1.1 Escenarios de uso

A continuación se identifican los escenarios propuestos.

ESCENARIO 1	**Registro de usuarios**
Perfil de usuario	Usuario activo
Personaje	Juan Pardo. Este personaje es un músico profesional de 44 años de edad. La tecnología móvil no es un problema para él. Tiene una trayectoria musical de más de 20 años.
Contexto	Los usuarios activos son normalmente artistas, músicos, gente del espectáculo que necesitan publicitar sus grupos y proyectos musicales. Este usuario pertenece a un grupo musical y necesita publicitar su grupo. Utiliza una app que permite promocionar a grupos musicales publicando información de los grupos y publicación de sus fechas
Objetivo	Publicitar su grupo
Tareas	Darse de alta en el sistema para autenticarse. Proporcionar datos del grupo. Comprobar su grupo en la lista
Necesidad de información	Localizar la opción de registro
Funcionalidades necesarias	Registrarse en el sistema: introducir un nombre de usuario y un password Introducir los datos del grupo: nombre, mail y página web.
Desarrollo de las tareas	Una vez instalada la aplicación deberá registrarse en el sistema. Para ello, necesitará localizar el botón de login. Si es nuevo y no se ha registrado antes deberá localizar la opción de nuevo usuario e introducir lo datos solicitados
Escenario con personaje	Hasta hace poco Juan Pardo utilizaba los canales tradicionales para promocionar los grupos en los que era componente. Desde hace un año ha creado su propio grupo y necesita una nueva manera rápida y económica de dar promoción a su grupo. Hoy ha quedado con su amigo de infancia y le ha comentado que hay una app que permite promocionar su grupo y publicar sus próximas actuaciones. A Juan le gusta lo que le cuentan y baja la app. Una vez bajada en su dispositivo móvil procede a ver cómo es el interface de la aplicación. Se decide a probarla con lo que lo primero que tiene que hacer es registrarse en el sistema para poder publicar su grupo y sus actuaciones. En el menú principal aparece la opción de Login que le solicita los datos de registro. Como Juan es la primera vez que accede se tiene que dar de alta. En la pantalla aparece un link que le permite registrarse como un usuario nuevo. Aparece otra pantalla que le solicita los datos de usuario, password, el nombre del grupo, su mail y la página web. En la misma pantalla aparece un botón que le permite enviar esos datos para completar el registro y poder ver su grupo publicado. Una vez el sistema ha registrado a Juan lo devuelve al menú principal

Figura 11: Escenario de uso 1

ESCENARIO 2	Anotación de actuaciones y su localización
Perfil de usuario	Usuario activo
Personaje	Miguel Lozano un chico universitario de 18 años que tiene un grupo Rock compuesto por 5 amigos. Cómo todos los jóvenes, tiene abundante conocimiento en el manejo de dispositivos móviles y de aplicaciones de geolocalización.
Contexto	El usuario pertenece a un grupo musical y pretende publicar las próximas actuaciones que realizarán indicando las fechas y su localización exacta para que las personas interesadas (fans, amigos,…) puedan localizar el evento sin dificultad.
Objetivo	Publicar una nueva actuación indicando el día que se realizará y su localización
Tareas	Identificarse en el sistema. Localizar los eventos Añadir un evento e introducir los datos Indicar la localización del evento en un mapa
Necesidad de información	Identificarse en el sistema y localizar como añadir los eventos
Funcionalidades necesarias	Localizar puntos mediante la navegación por el mapa, selección del punto, formulario de registro del evento
Desarrollo de las tareas	Hay que identificarse en el sistema para poder publicar un evento o actuación. Una vez identificados se localiza el botón de eventos y luego se localiza el botón de añadir evento que nos mostrará un formulario donde introduciremos los datos del evento: fecha y lugar
Escenario con personaje	Son las 8 de la tarde. Miguel, como otros días, va ensayar con su grupo las nuevas canciones compuesta por el guitarrista del grupo. A los diez minutos le llaman por teléfono para comunicarle que tiene una actuación en la sala Meirás de A Coruña dentro de dos semanas. Ilusionado, por ser su primera actuación, quiere publicarla en su aplicación para que sus amigos y seguidores puedan ver el día que actuarán y dónde se celebrará. Para ello selecciona la app de su dispositivo móvil, se identifica y localiza el botón de actuaciones. Una vez ha desplegado la lista de eventos procede a localizar el botón de añadir nuevo evento. Se despliega un nuevo formulario solicitando indicar la fecha y el lugar de la celebración de su actuación. Miguel aprecia que por la dirección indicada no aparece el lugar en el mapa. Miguel se da cuenta de que el formulario le permite indicar la localización exacta pulsando en el mapa, con lo cual el evento marca su localización correctamente.

Figura 12: Escenario de uso 2

ESCENARIO 3	Comentarios promocionales de grupos
Perfil de usuario	Usuario activo
Personaje	Alicia Pérez es una chica de 24 años y estudiante de 5 º de Piano. Ha formado un cuarteto de música con otros estudiantes. Tiene conocimientos con los dispositivos móviles y los utiliza para navegar por internet y conectarse en las redes sociales.
Contexto	Este personaje, como muchos otros, es de los que están conectados continuamente a las redes sociales. Son conscientes de que las redes sociales son el medio por excelencia y permiten rápidamente que todo el mundo se entere de lo que sucede rápidamente. Por ello, va a utilizar este fenómeno para promocionar su grupo al mayor número de usuarios.
Objetivo	Dar a conocer su cuarteto musical a los demás usuarios
Tareas	Autenticarse en la aplicación Localizar la opción de comentarios Añadir un comentario Complementar el formulario de entrada Publicar el comentario
Necesidad de información	Identificarse en el sistema, publicar información
Funcionalidades necesarias	Se necesita identificarse en el sistema para poder agregar y publicar información
Desarrollo de las tareas	Lo primero que se necesita es identificarse en el sistema para ello se tendrá que identificar el proceso de acceso al sistema dónde una vez accedido se le indicará que introduzca el usuario y password necesarios para identificarse como usuario registrado en el sistema. Luego tiene que localizar la opción de comentarios y localizar la opción de añadir comentarios. A continuación rellenar el formulario con los datos solicitados y enviar el comentario para ser publicado

Escenario con personaje	Son las 6 de la tarde. Alicia llama a su compañera Marta del grupo que necesitan dar un poco de publicidad al grupo para que sus compañeros y amigos puedan estar al día de los movimientos de su cuarteto. Ya ha publicado información en las redes sociales, pero le gustaría estar en otra red social más especializada en el ámbito musical para acceder a profesionales del sector musical. Alicia ya había bajado la app e incluso se había registrado. Entonces se anima a publicar ciertas curiosidades de su grupo.

Figura 13: Escenario de uso 3

ESCENARIO 4	**Localización de un evento en el mapa y compartir**
Perfil de usuario	Usuario pasivo
Personaje	Carmen Vidal 42 años. Ama de casa. Tiene un Smartphone que utiliza para conectarse a internet, utilizar el Whatsapp como mensajería y también utiliza Google Maps para localizar algún lugar.
Contexto	Carmen tiene dos hijos adolescentes que han formado un grupo musical con los amigos de la infancia. Como madre intenta estar atenta a los movimientos de sus hijos e intenta averiguar dónde actúan sus hijos para ver qué tal se desenvuelve su grupo y de paso divertirse con ellos. A veces informa a sus amigos dónde actuarán los críos para acompañarle en sus actuaciones
Objetivo	Buscar un evento y compartirlo con otras personas por diversos medios que permita el dispositivo móvil
Tareas	Localizar la acción de Mapa Buscar el evento solicitado dentro del mapa Localizar la acción de compartir Compartir el evento por el medio elegido
Necesidad de información	Disponer de la aplicación instalada en el dispositivo móvil, saber manejar un punto de identificación en un mapa
Funcionalidades necesarias	Localizar puntos mediante la navegación por el mapa, selección del punto, compartir el punto de información (evento)
Desarrollo de las tareas	Localizar el punto navegando por el mapa, seleccionarlo, pulsar el botón de compartir y seleccionar la forma de envío del evento
Escenario con personaje	El próximo mes actuarán los hijos de Carmen en la sala VRT de A Coruña. Sus hijos le informan que puede verlo en la app donde ellos han registrado sus actuaciones. Carmen descarga la app e intenta buscar la agenda de actuaciones del grupo de sus niños. Busca el evento del día en concreto y lo visualiza en el mapa. En el mapa puede ver la información del evento: fecha, lugar y su localización. Después comparte la información con los amigos por medio de Whatsapp.

Figura 14: Escenario de uso 4

ESCENARIO 5	**Actuaciones cerca de mi ubicación**
Perfil de usuario	Usuario pasivo
Personaje	María González es una chica de 24 años estudiante de medicina que le gusta salir de fiesta con sus amigos los fines de semana. Como la mayoría de personas de esta generación conocen y disponen de dispositivos móviles que le permiten estar conectados con sus amigos, familiares y demás personas. Están al día con las redes sociales.
Contexto	En el sector espectáculos, los grupos y artistas musicales promocionan sus actuaciones y eventos promocionales con antelación para que el público pueda acercarse a sus eventos. Por otro lado, muchas personas buscan por internet información sobre los eventos que suceden cerca de uno para acercarse a ellos con el fin de buscar divertimento. Los usuarios pasivos de esta app pueden acceder a estos eventos (sin necesidad de registrarse) y poder compartirlos con otras personas independientemente de que sean usuario o no de la app.
Objetivo	Localizar eventos cerca de la ubicación del usuario en un radio determinado
Tareas	Localizar la acción “Cerca de mí” Buscar un evento solicitado Marcar el evento Indicar la información detallada en el punto elegido

Necesidad de información	Localizar el punto en el mapa
Funcionalidades necesarias	Localizar puntos mediante la navegación por el mapa, selección del punto.
Desarrollo de las tareas	Localizar el punto navegando por el mapa, seleccionarlo, pulsar punto para ver más información
Escenario con personaje	Es viernes. Después de las clases de la facultad, María se dispone a ir a verse con sus amigas a un local de moda como todos los viernes. Hoy no tienen ningún plan en especial. Una vez reunidas en el local María dice: ¿qué hacemos hoy? (sacando su Smartphone). Sus amigas con cara de póker responden: pues no sabemos. Entonces María arranca su app y comprueba en un mapa los eventos que se celebran hoy cerca de su ubicación. Comprueban los eventos y se animan a ir a uno en concreto esta noche.

Figura 15: Escenario de uso 5

7.1.2 Flujos de interacción

En el siguiente diagrama de flujo se refleja el comportamiento interactivo de la aplicación.

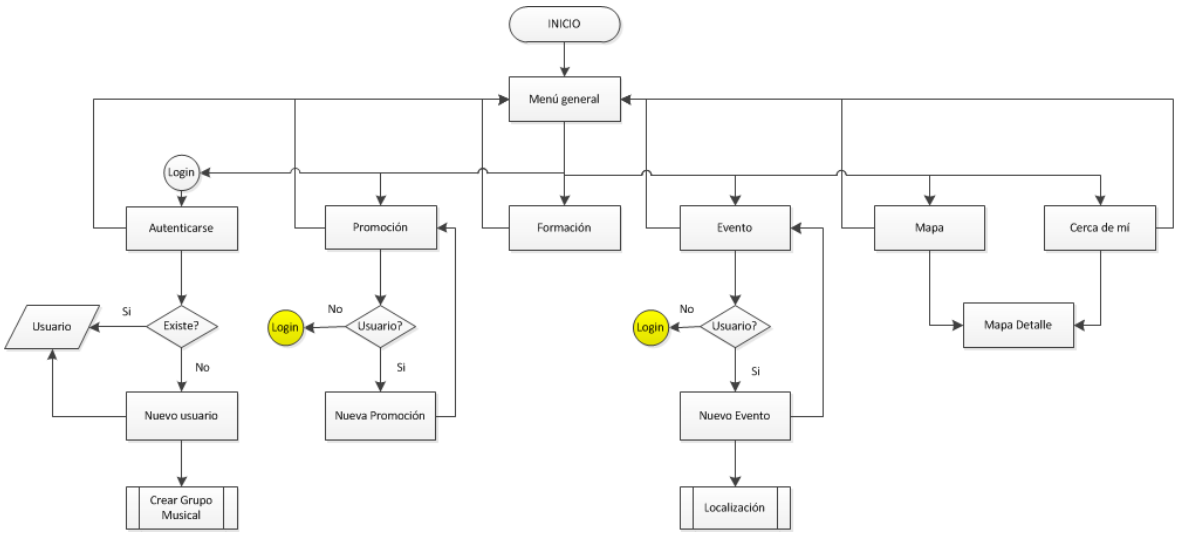


Figura 16: Diagrama de flujo

7.2 Prototipado

Un prototipo es una representación de la aplicación, que permite comunicar decisiones de diseño y evaluarlo antes de desarrollar el producto final.

Como punto de partida se realizan los primeros bocetos (Sketches) con el objetivo de encauzar los primeros argumentos de diseño. Después, con los flujos de interacción definidos en el diseño conceptual, se realiza un prototipo horizontal de alta fidelidad. Con ello se visualizan las diferentes pantallas de la aplicación con los elementos que contendrán, y se obtiene una representación más precisa y útil de cara a obtener una aplicación final más usable, y que proporcione una experiencia de usuario más satisfactoria.

7.2.1 Sketches

Con el prototipo de baja fidelidad se trata de modelar elementos generales del sistema, sin llegar al detalle. Se ha recogido una primera aproximación por medio de escaneos de bocetos realizados en papel.

Lo primero que se ha esbozado es el menú principal. Con ello se pretende que recoger las opciones principales que llevará implícitas la aplicación. Después se analizará las pantallas que podrán manejar los perfiles de usuarios que se han determinado en la etapa de indagación: usuarios activos y usuarios pasivos.

Se necesitarán pantallas que permitan ver una lista de datos cualquier perfil de usuario: comentarios, grupos musicales y eventos que los usuarios activos hayan publicado. Por otro lado se necesitará las pantallas que permitan alimentar esas listas, es decir, dónde se puedan dar de alta o publicar los datos relacionados a esas listas. Por otro lado, también se necesitará pantallas donde aparezca un mapa dónde se pueda ver los eventos que tendrá lugar en un tiempo igual o superior al actual en ese momento, y que nos dé información en detalle del evento seleccionado. Tampoco se puede olvidar las pantallas correspondientes a la identificación de los usuarios que publiquen información en el sistema (usuarios activos), ya que para que se pueda publicar información en el sistema tendrán que logearse, y para ello, tendrán que estar previamente registrados en el sistema.

Con toda esta información se han generado los siguientes bocetos:

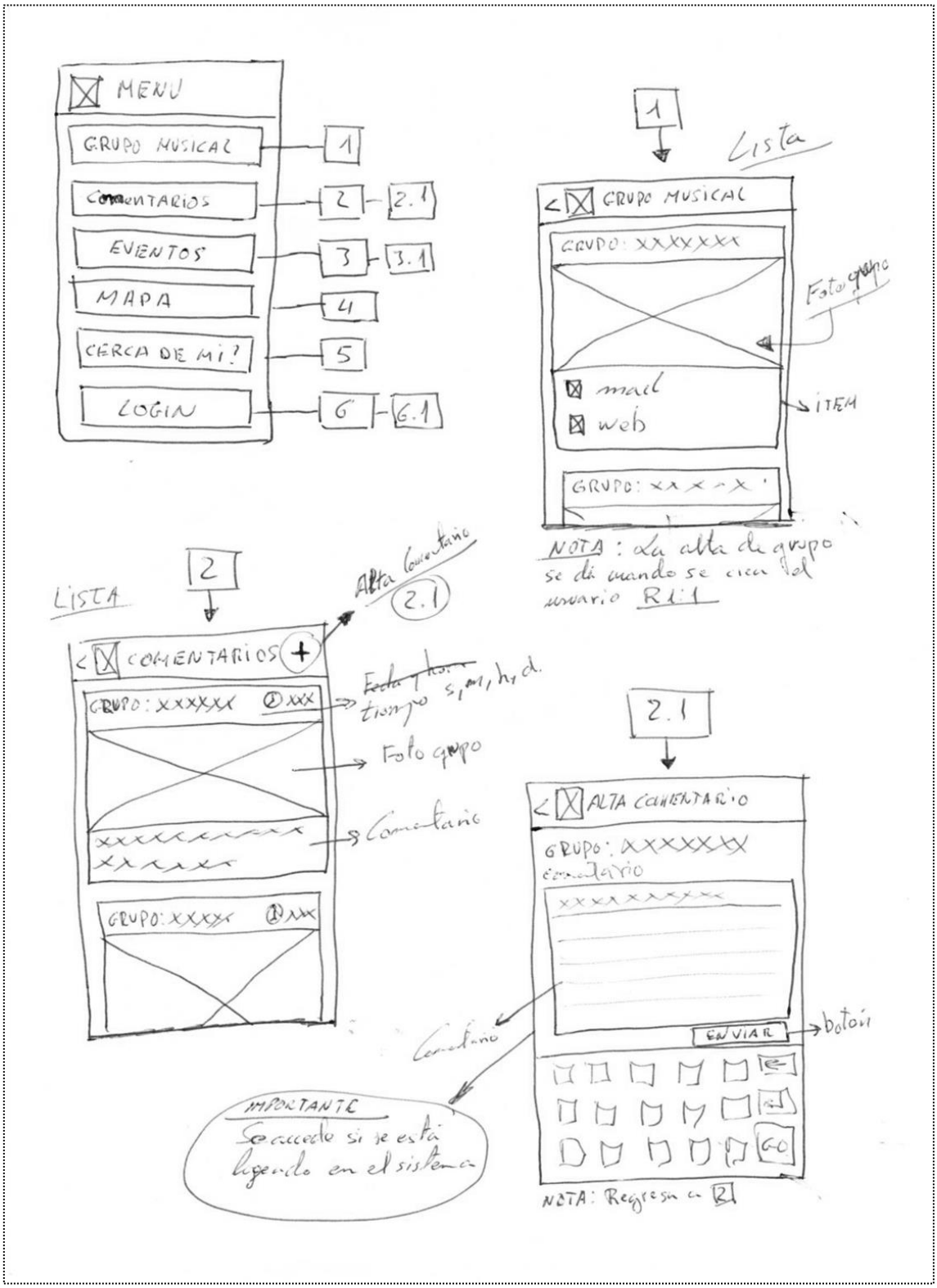


Figura 17: Boceto 1

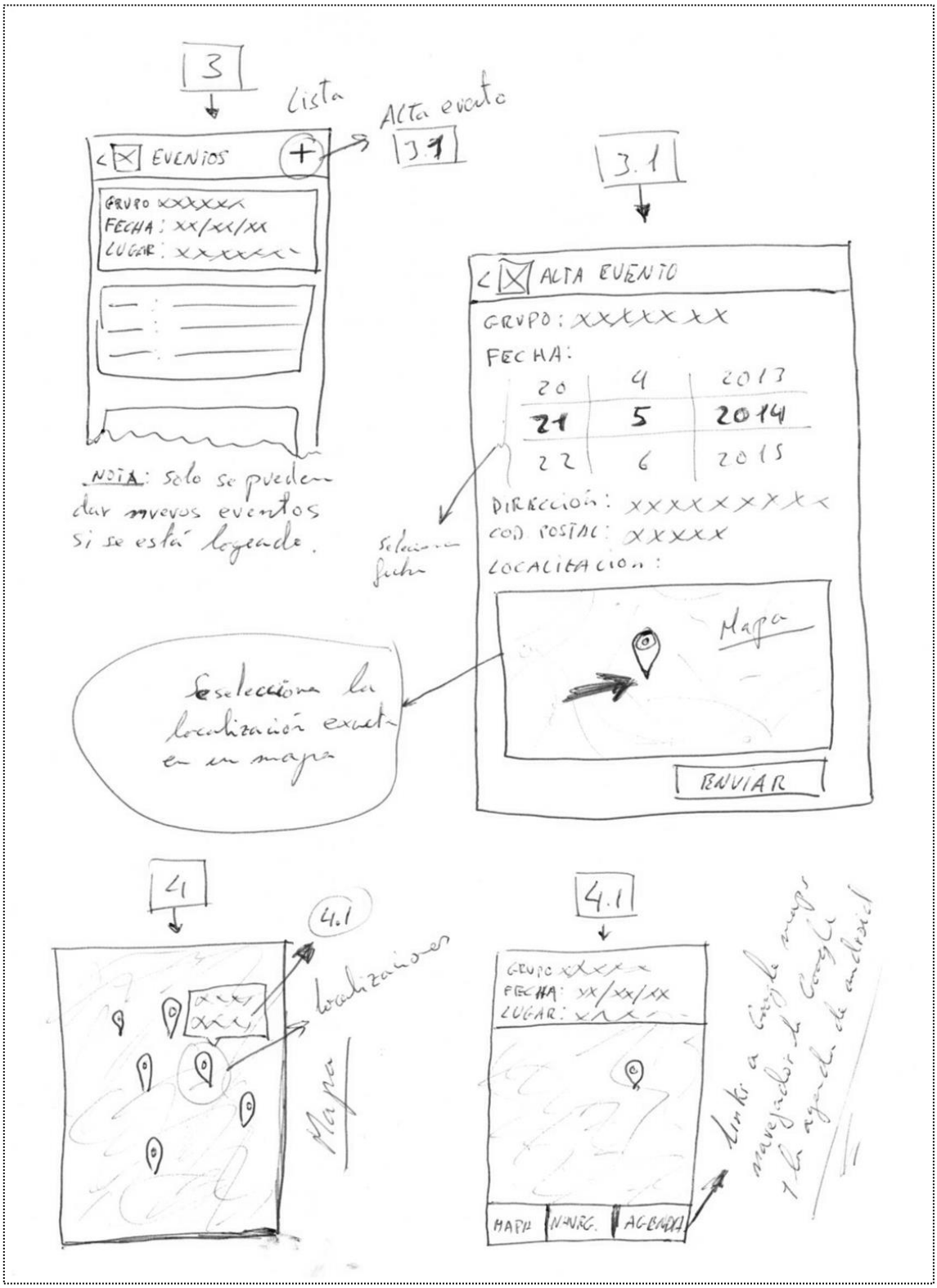


Figura 18: Boceto 2

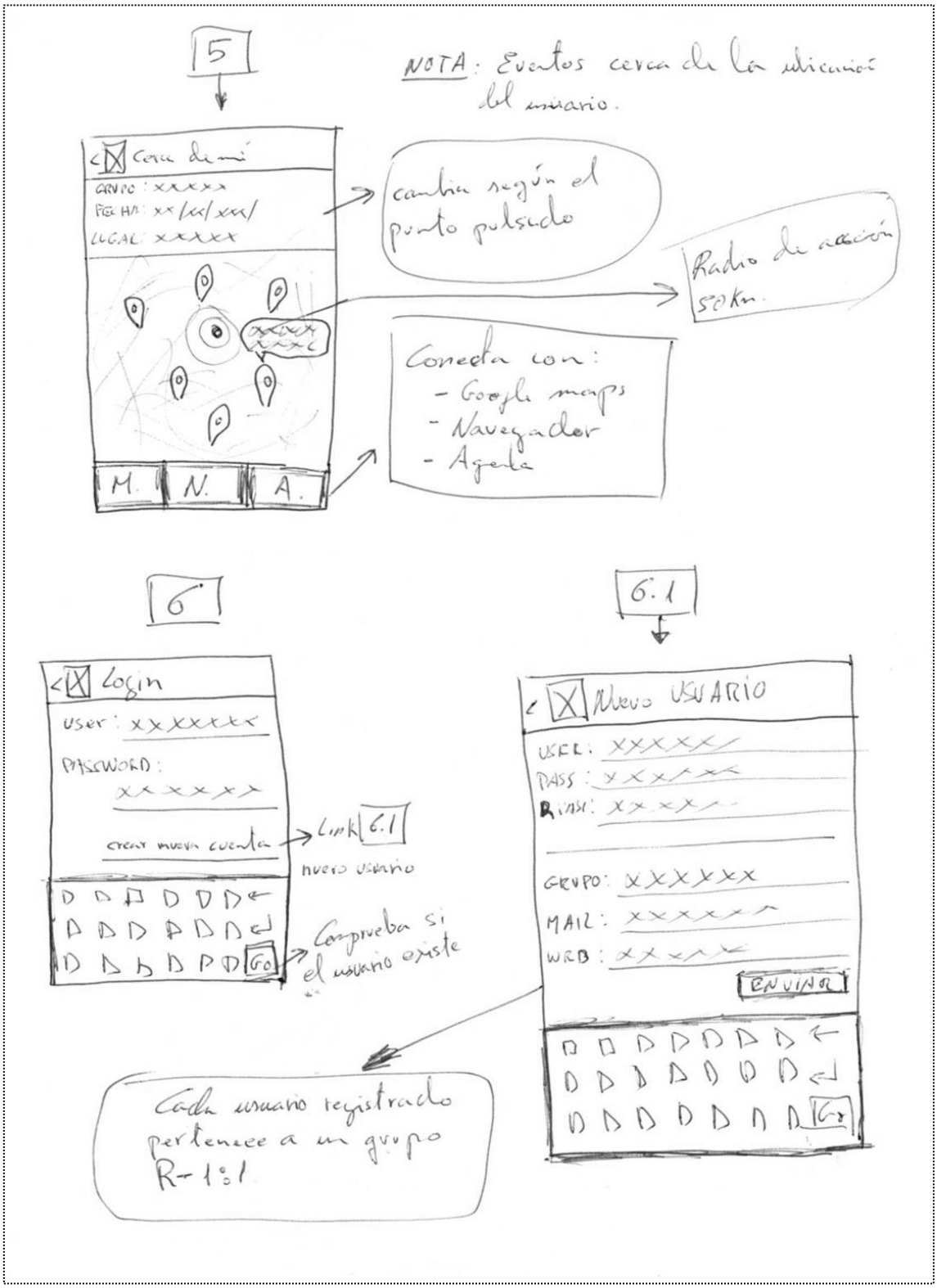


Figura 19: Boceto 3

7.2.2 Prototipo

Con el modelado obtenido en la fase anterior (prototipo de baja fidelidad) se puede definir el modelo de alta fidelidad.

Con el prototipo de alta fidelidad se trata de construir un modelo lo más próximo posible al sistema que se diseña y desarrolla. Este tipo de prototipo, con diferencia al apartado anterior, sirve para evaluar de manera más precisa aspectos funcionales y de usabilidad. También servirá como evaluación mediante un test con usuarios. A continuación se puede ver el prototipo dinámico generado con el programa JustInMind Prototyper Free. El prototipo está disponible online para probarlo en la siguiente url:

https://www.justinmind.com/usernote/tests/12989260/12989271/12991351/index.html

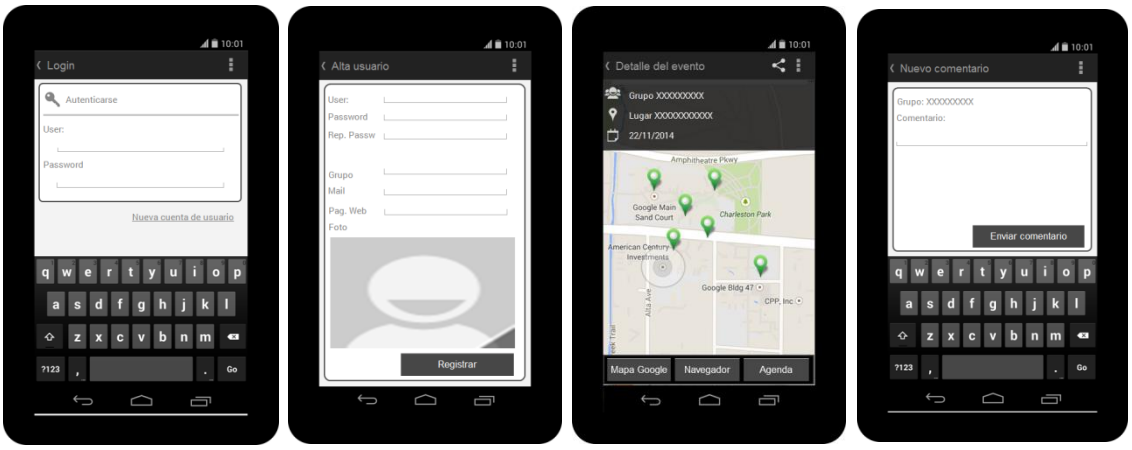


Figura 20: Prototipos realizados con JustInMind Prototyper

8 Evaluación

El objetivo de la última fase es planificar la evaluación del prototipo. El proceso de DCU es un proceso iterativo y, por tanto, hay que ir evaluando los diseños y corrigiendo los errores de manera iterativa.

8.1 Test de usuarios

El test de usuarios es la prueba que representa la mejor forma de evaluar la usabilidad de un diseño. Esta fase se basa en la observación de cómo un grupo de usuarios llevan a cabo una serie de tareas encomendadas por el evaluador, analizando, de esta manera, los problemas de usabilidad con los que se encuentran.

8.1.1 Perfiles de usuario (testeadores)

Para la realización del test de usuarios se necesitan usuarios que se presten a realizar la prueba. Estos usuarios deberán ser de diferentes perfiles para obtener una evaluación lo más heterogénea posible.

Con el fin de tener más información relativa al test se definen unos formularios para que los usuarios de la prueba los cubran antes y después del test.

El formulario que se realiza antes del test obtiene datos del usuario para determinar su perfil en algunas cuestiones de contexto. El cuestionario alberga cuestiones como:

Formulario	
Nº	Preguntas
1	¿Tiene alguna experiencia con los dispositivos móviles (SmartPhones, Tablets,…?
2	¿Cuántas horas al día dedica o utiliza este tipo de dispositivos?
3	¿Qué tipo de aplicaciones utiliza habitualmente?
4	¿Qué tipo de conexión utiliza (Wifi, 3g, 4g,…)?

5	¿Qué tipo de sistema operativo utiliza en su dispositivo (Android, IOS,…)?
6	¿Qué resolución de pantalla tiene su dispositivo?

Figura 21: Cuestionario

8.1.2 Tareas a realizar

Se solicitará al participante una serie de tareas a realizar sobre el prototipo, analizando los errores que cometa, el tiempo empleado y su satisfacción final una vez finalice la tarea.

La primera información que se quiere obtener mediante la prueba es el grado de entendimiento. Por ello, se le indica al usuario que no haga nada, que únicamente observe el interfaz y diga qué cree que está viendo, de qué cree trata el app, para qué cree que sirve, y todas aquellas impresiones que tenga (expresiones y gestos).

Seguidamente se le indica que durante 5 minutos explore sobre los escenarios que más adelante se especificarán junto con las tareas correspondientes, para ver si de forma intuitiva y cognitiva van identificando su objetivo. Los resultados serán recogidos y valorados de la misma forma que las tareas correspondientes.

Una vez se ha obtenido una primera impresión de los escenarios y del grado de comprensión del usuario acerca de la función, objetivos y opciones que ofrece la app, analizamos la facilidad de uso con las siguientes tareas ejecutadas en sus diversos escenarios.

Escenario	**Registro de usuarios**
Tareas	Localizar la acción de registro Proporcionar datos del formulario. Enviar los datos de registro Comprobar su grupo en la lista

Escenario	**Anotación de actuaciones y su localización**
Tareas	Localizar la acción de Login Complementar el formulario de identificación al sistema. Localizar los eventos Añadir un evento e introducir los datos Indicar la localización del evento en un mapa Publicar el evento

Escenario	**Añadir comentarios promocionales de grupos**
Tareas	Localizar la acción de Login Autenticarse en la aplicación Localizar la opción de comentarios Añadir un comentario Complementar el formulario de entrada Publicar el comentario

Escenario	**Localización de un evento en el mapa y compartir**
Tareas	Localizar la acción de Mapa Buscar el evento solicitado dentro del mapa Localizar la acción de compartir Compartir el evento por el medio elegido

Escenario	**Actuaciones cerca de mi ubicación**
Tareas	Localizar la acción “Cerca de mí” Buscar un evento solicitado Marcar el evento Indicar la información detallada en el punto elegido

Figura 22: Tareas

8.1.3 Preguntas sobre las tareas

Una vez realizado el test se le hace al usuario una serie de preguntas resumidas en un formulario con el objetivo de obtener el grado de satisfacción o frustración del interface observadas por el usuario en el prototipo. El segundo formulario es para que puntúe sobre proceso del test en general.

Formulario	
Nº	Preguntas
1	¿Qué opina del prototipo que acaba de testear?
2	¿Cree que están correctas las opciones del menú?
3	¿Cómo valora el proceso de registro?
4	¿Qué dificultades ha encontrado para completar las tareas?
5	¿Qué defectos ha detectado en el uso del prototipo?
6	¿Hay alguna opción que hayas echado de menos?
7	¿Cómo valora los colores y contrastes utilizados?
8	¿Cómo se podría mejorar el prototipo?

Figura 23: Formulario para obtener el grado de satisfacción o frustación

Cuadro para puntuar con una X del 1 al 5

	1	2	3	4	5
Grado de satisfacción que has obtenido del test					
Duración del test					
Dificultades en las tareas propuestas					
Equipamiento técnico del laboratorio					
El trato dado recibido del monitor					

Figura 24: Formulario para puntuar sobre el test general

8.1.4 Realización del test con usuarios y análisis de sus resultados

El control y dirección del test es controlado por un monitor. El monitor se encarga de recibir al usuario, explicar cómo va el laboratorio, es decir el escenario en dónde se realizarán las pruebas, y en qué consisten las pruebas.

Una vez empezado el test, el monitor es quién dirá al usuario lo que tiene que hacer en el testeo y también recordará al usuario que tiene que ir informando de todos los pasos que va

haciendo. Recoge todas las observaciones y va anotando los problemas o comentarios del usuario. Será, también, el encargado de analizar los datos obtenidos en el test.

El objetivo de estas pruebas es mejorar de forma iterativa la usabilidad de la aplicación, por lo que cada prueba nos ofrecerá suficiente información de carácter cualitativo para mejorar la solución de diseño, aun cuando no nos permita detectar el 100% de los problemas de usabilidad. Es decir, esta es una prueba destinada a medir tanto la usabilidad objetiva (qué y cómo actúa el usuario), como la usabilidad subjetiva (cómo de fácil ha percibido la tarea). Con ello, no sólo se pretende detectar en qué momentos el usuario se equivoca o se detiene durante la realización de la tarea, sino también el porqué; qué es aquello que no entiende o qué le ha llevado a tomar decisiones equivocadas.

Una vez los participantes finalizan la prueba y se haya registrado toda la información pertinente, se procede a analizar los resultados y sintetizarlos en un informe final, concluyendo qué mejoras necesita el diseño en base a estos resultados.

En base a los resultados obenidos, en este test se pudieron constatar varios puntos a tener en cuenta:

- **Facilidad de acceso al menú:** La mayoría de los usuarios están familiarizados con aplicaciones de google donde el menú se accede con un pequeño arrastre desde el lado izquierdo donde aparecen las opciones principales de la aplicación. Los usuarios echaban de menos esta opción ya que les resulta muy útil.

- **Promoción de las formaciones en primer plano:** Otra gran porcentaje opinaron que, ya que se trata de una aplicación de promoción, fuera esa la primera pantalla a mostrar con los comentarios promocionales ordenados de manera descendente, es decir, que se vieran los últimos comentarios como los primeros de la lista.

- **Marcar un radio de eventos próximos en el mapa:** Dentro de la funcionalidad de indicar los eventos próximos a la posición del usuario. Algunos de ellos solicitaron la necesidad de marcar con una circunferencia el radio máximo de proximidad de los eventos.

- **Contemplar diversas versiones de Android:** Los usuarios disponen de diferentes dispositivos con diferentes versiones del sistema operativo. Esto es un grave problema conocido de Android como es la fragmentación (se considera que Android es un sistema operativo fragmentado debido a que varias versiones del mismo se encuentran disponibles) que desde hace algunas versiones intenta solucionar con más o menos acierto desarrollando unas librerías de soporte llamadas android-support. Los usuarios disponen de versiones desde la 2.2 a la 5.0.

- **Contemplar diversos dispositivos (Smarphone, Tablets):** Los distintos tamaños, resoluciones y densidades de los dispositivos hacen que el diseño pueda verse alterado según en qué dispositivo se ejecute la aplicación. Los usuarios dispone de diversos dispositivos (Smartphone de 3,4, (4,5), 5 pulgadas, Tablets de 5, 7, 8, 9, 10.1 pulgadas).

Una vez determinadas estas propuestas se procedió a su estudio para contemplar si se podrían implementar en el proyecto sin que ello afectase a la planificación inicial, ya que, en

caso de detectar desviaciones respecto al calendario previsto, habría que establecer las medidas apropiadas para garantizar la finalización del trabajo con éxito.

Después del estudio el resultado fue positivo, ya que la mayoría de propuestas corresponden al diseño y Android proporciona soluciones para todas ellas, sin embargo, se ha limitado la propuesta de "**Contemplar diversos dispositivos…"** a los rangos de hasta 5'', 7'' y 10.1'' con el objetivo de acotar la carga de trabajo, pero sin que ello obstaculice conocer la forma de proceder para alcanzar el objetivo en un trabajo posterior.

IMPLEMENTACIÓN

En la etapa anterior se realizó un prototipo de la aplicación y un test de usuarios con el fin de apreciar defectos y necesidades susceptibles de mejoras (dentro de los límites planteados al inicio del proyecto) para proporcionar un diseño actual, funcional y con un alto grado de experiencia de usuario que diera un valor añadido a la aplicación.

Al realizar la prueba y valoración del prototipo se apreciaron necesidades susceptibles de mejoras que propiciaron cambios en el diseño a favor de aplicar una experiencia de usuario más acorde al tipo de aplicaciones que el usuario está más habituado a manejar.

Estos cambios no afectan al diseño técnico, sino más bien al desarrollo de la interface de usuario, y ya que se está utilizando el método de desarrollo ágil, se realizan los cambios necesarios para acometer las nuevas propuestas.

Las tareas realizadas para conseguir el objetivo propuesto se han dividido en tres etapas: **diseño técnico, desarrollo y pruebas.**

9 Diseño Técnico

En esta etapa se concreta con más detalle la arquitectura de la aplicación, el paradigma a seguir, los patrones de diseño a aplicar, el modelo de datos, donde residirá la persistencia de los datos, la comunicación entre los dispositivos y los servicios que consumirán…

9.1 Arquitectura de la aplicación

La arquitectura a seguir es la de cliente/servidor que consta de dos capas diferenciadas: el cliente que se ejecuta en un dispositivo Android y la parte servidora, dónde se alojan los datos de la aplicación y pone a disposición los servicios de comunicación entre el dispositivo y el servidor.

Para la parte servidora se utiliza un tipo de servicio en la nube denominado Backend as a Service (BaaS), ya que este servicio proporciona todo lo necesario para el funcionamiento de la aplicación y, de ese modo, se puede abstraer de la tarea tediosa de construir y mantener un backend, ya que no es objetivo principal en este proyecto.

En este caso se ha optado por el BaaS Parse.com (*www.parse.com*). Parse.com un servicio en la nube que nos permite, entre otras funciones, guardar objetos de la aplicación y posteriormente recuperarlos. En el apartado "*Arquitectura Servidor*" se hablará más en detalle de este servicio.

También hay que tener en cuenta los servicios de Google Maps en la nube que serán consumidos por las Apis de Servicios de Google Play Services. Este componente proporciona funciones principales, como la autenticación para servicios de Google, servicios basados en la ubicación de mayor calidad y menor potencia, y naturalmente, proporciona mapas más envolventes y que mejoran la experiencia de uso.

Estos son los elementos que compondrán nuestra arquitectura de la aplicación.

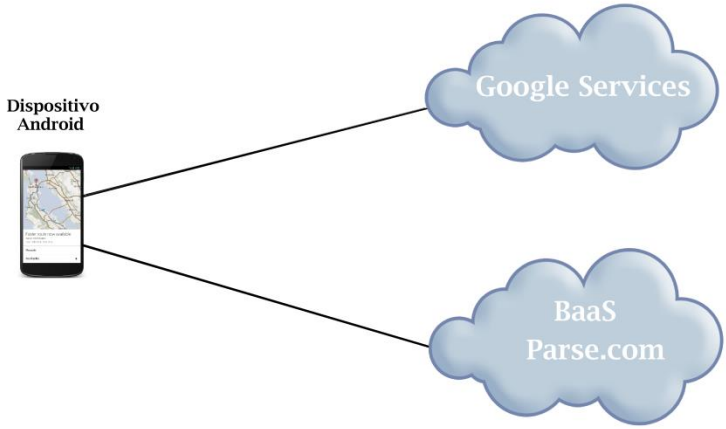


Figura 25: Arquitectura de la aplicación

En la parte cliente, es decir en el dispositivo móvil, se encargaría de soportar el SDK de Android, las Apis de Google Services para la generación de Mapas y servicios de geolocalización, las Apis de Android-support para la compatibilidad con diferentes versiones de Android, las Apis de Parse.com para la comunicación con el Backend y la persistencia de datos, y otras librerías para diversas funciones que conoceremos a lo largo de la memoria. Se puede ver de una manera gráfica todos los componentes de la arquitectura de la aplicación.

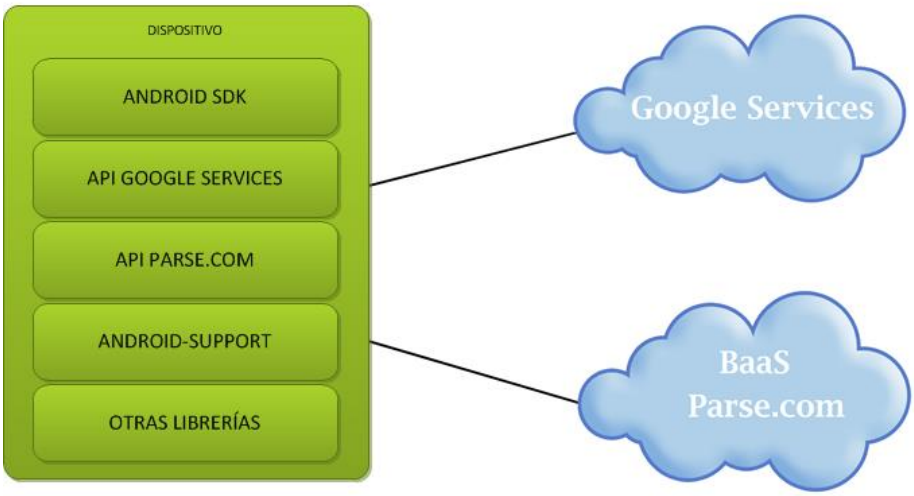


Figura 26: Arquitectura de la aplicación y sus Componentes

9.2 Arquitectura cliente

Como ya se ha dicho en la planificación, la aplicación se va a desarrollar en el entorno nativo de Android. En Android es habitual aplicar un modelo MVC (Modelo, Vista, Controlador) y la estructura que nos impone, a la hora de crear un proyecto Android, sigue ese paradigma de programación aplicando un patrón muy definido y estructurado.

Siguiendo este modelo la aplicación se divide en diversas capas como se puede apreciar en la siguiente figura:

Figura 27: Arquitectura cliente

- La capa de presentación: formada por los recursos textuales y la interfaz gráfica.
- La capa de permisos: permiten a la aplicación acceder a diferentes características del dispositivo.
- La capa de la lógica de la aplicación: formada por las diferentes clases que dan forma a la aplicación.
- Librerías: que necesitará la aplicación.

Estas capas se explicarán más en detalle en el apartado "*Programación de las funcionalidades*".

9.3 Arquitectura servidor

En el apartado "*Arquitectura de la aplicación*" ya se ha comentado que se usaría un BaaS. Este BaaS se utiliza para la persistencia de los datos de la. Este es un servicio en la nube que permite guardar los objetos de la aplicación y posteriormente recuperarlos. La ventaja de usar un sistema de este tipo, aparte del ahorro de tiempo en el desarrollo de una solución propia, es la facilidad para migrar entre diferentes plataformas, ya que dispone de Apis para Android, IOS, Windows, OS X, JavaScript, Windows Phone Y Rest.

También ofrece una serie de funcionalidades para la gestión de usuarios, conexión con redes sociales, integración con aplicaciones tipo cloud y ejecución de código propio en la nube.

9.4 Diagramas de casos de uso

En este diagrama se puede apreciar los actores y casos de usos que contendrá la aplicación. El usuario pasivo solo accederá a los recursos disponibles pero no podrá ni añadir ni borrar ningún recurso. Por lo contrario, el usuario activo tendrá las mismas funciones que el usuario pasivo pero además podrá añadir y borrar recursos. Para ello el usuario activo tiene que registrarse en el sistema.

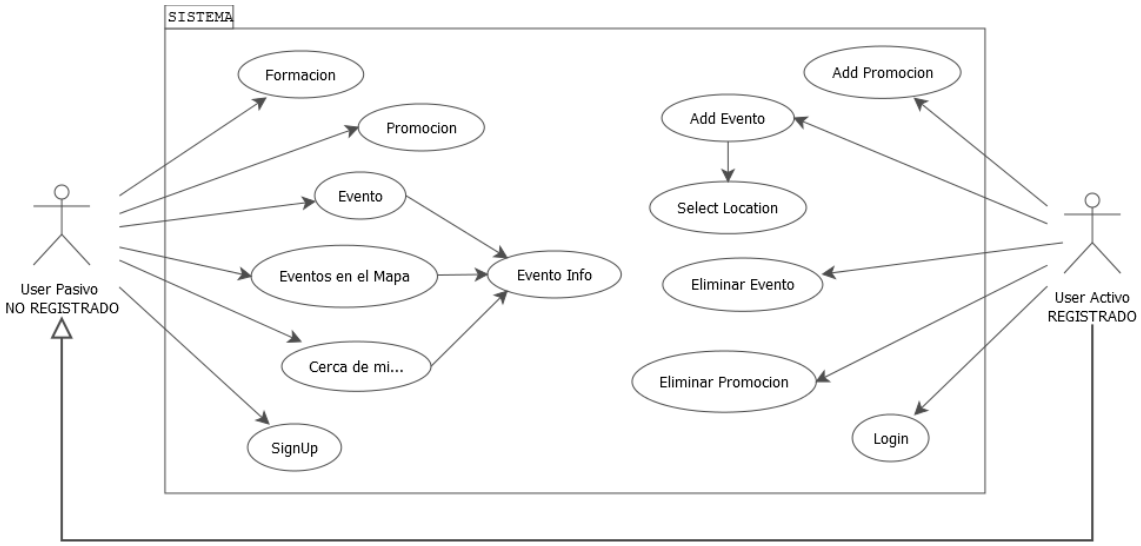


Figura 28: Diagrama de casos de uso

A continuación se exponen las especificaciones detalladas de los casos de uso con representaciones textuales que servirán como conjunto base para las pruebas que se realizarán más adelante.

Nombre	Consultar formaciones (1) (Formacion)
Actores	Usuario pasivo
Objetivo	Consultar las formaciones (artista y grupos musicales) que existen en el sistema
Precondiciones	No haber iniciado una sesión en el sistema
Postcondiciones	Ninguna
Ecenario básico	1. Acceder al menú 2. Seleccionar la opción de “Formaciones”
Escenario alternativo	

Nombre	Consultar formaciones (2) (Formacion)
Actores	Usuario activo
Objetivo	Consultar la ficha de su formación
Precondiciones	Haber accedido al sistema haciendo logín
Postcondiciones	Ninguna
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Formaciones”
Escenario alternativo	

Nombre	Consultar Promociones (1) (Promocion)
Actores	Usuario pasivo
Objetivo	Consultar los últimos comentarios promocionales de los artista y grupos musicales que existen en el sistema

Precondiciones	No haber iniciado una sesión en el sistema
Postcondiciones	Ninguna
Escenario básico	1. Arrancar la aplicación
Escenario alternativo	1. Acceder al menú 2. Seleccionar la opción de “Promociones”

Nombre	Consultar Promocion (2) (Promocion)
Actores	Usuario activo
Objetivo	Consultar los últimos comentarios promocionales publicados por un usuario
Precondiciones	Haber iniciado una sesión en el sistema y no haberla cerrado antes de salir
Postcondiciones	Ninguna
Escenario básico	1. Arrancar la aplicación
Escenario alternativo	1. Acceder al menú 2. Seleccionar la opción de “Promociones”

Nombre	Consultar eventos (1) (Evento)
Actores	Usuario pasivo
Objetivo	Consultar todos los eventos que se celebrarán desde el día que el usuario hace la consulta en adelante
Precondiciones	No haber iniciado una sesión en el sistema
Postcondiciones	Ninguna
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Eventos”
Escenario alternativo	

Nombre	Consultar eventos (2) (Evento)
Actores	Usuario activo
Objetivo	Consultar los eventos publicados por un usuario que se celebrarán desde el día que el usuario hace la consulta en adelante
Precondiciones	Haber iniciado una sesión en el sistema
Postcondiciones	Ninguna
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Eventos”
Escenario alternativo	

Nombre	Consultar eventos en el mapa (Eventos en el Mapa)
Actores	Usuario activo y pasivo
Objetivo	Consultar los eventos que se celebrarán un día determinado
Precondiciones	Ninguna
Postcondiciones	Eventos encontrados según la fecha determinada
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Eventos en el mapa” 3. Seleccionar una fecha 4. Buscar los eventos de la fecha seleccionada
Escenario alternativo	

Nombre	Cerca de mí (Cerca de mí)
Actores	Usuario activo y pasivo
Objetivo	Consultar todos los eventos que se celebrarán en diferentes fechas desde el día que realiza la consulta y que estén dentro de un radio determinado de proximidad de la localización del usuario
Precondiciones	Ninguna
Postcondiciones	Eventos encontrados dentro del radio de proximidad de la localización del usuario
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Eventos cerca de mí…”
Escenario alternativo	

Nombre	Evento Info
Actores	Usuario activo y pasivo
Objetivo	Mostrar el evento en detalle con opciones para compartir con otros usuarios y aplicaciones
Precondiciones	Seleccionar un evento
Postcondiciones	Ninguna
Escenario básico	1. Seleccionar un evento: bien desde la lista de eventos, desde “Eventos en el mapa” o desde “Eventos cerca de mi…”
Escenario alternativo	

Nombre	SignUp
Actores	Usuario pasivo
Objetivo	Registrarse y dar de alta una formación
Precondiciones	No estar registrado en el sistema
Postcondiciones	El usuario está registrado en el sistema y se ha publicado su formación
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Login” 3. Pulsar la opción de “¿No tienes cuenta?” 4. Rellenar los campos solicitados 5. Agregar una foto seleccionada de la galería o tomar una foto de la cámara 6. Una vez comprobado los datos pulsar el botón de “Sign Up” para crear la cuenta de usuario y publicar la formación
Escenario alternativo	

Nombre	SignUp
Actores	Usuario pasivo
Objetivo	Registrarse y dar de alta una formación
Precondiciones	No estar registrado en el sistema
Postcondiciones	El usuario está registrado en el sistema e iniciada una sesión (usuario activo) y se ha publicado su formación
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Login” 3. Pulsar la opción de “¿No tienes cuenta?” 4. Rellenar los campos solicitados 5. Agregar una foto seleccionada de la galería, o tomar una foto con la cámara del dispositivo 6. Una vez comprobado los datos pulsar el botón de “Sign Up” para crear la cuenta de usuario y publicar la formación
Escenario alternativo	

Nombre	Login
Actores	Usuario activo
Objetivo	El usuario accede al sistema e inicia una sesión
Precondiciones	Estar registrado en el sistema
Postcondiciones	El usuario accede al sistema iniciando una sesión
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Login” 3. Rellenar los campos solicitados: username y password 4. Pulsar el botón de “Log in” para acceder al sistema
Escenario alternativo	4. Pulsar el botón de “Log in” para acceder al sistema 5. El sistema no reconoce los datos introducidos 6. Volver al punto 3

Nombre	Add Promocion
Actores	Usuario activo
Objetivo	El usuario añade al sistema una nueva promoción
Precondiciones	Estar registrado en el sistema y haber iniciado una sesión
Postcondiciones	Se crea una nueva promoción
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Promociones” 3. Pulsar el botón de “+” en la cabecera de la pantalla 4. Rellenar los datos 5. Pulsar el botón de “GUARDAR”
Escenario alternativo	

Nombre	Add Evento
Actores	Usuario activo
Objetivo	El usuario añade al sistema un nuevo evento
Precondiciones	Estar registrado en el sistema y haber iniciado una sesión
Postcondiciones	Se crea un nuevo evento
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Eventos” 3. Pulsar el botón de “+” en la cabecera de la pantalla 4. Elegir la fecha del evento 5. Rellenar los datos del punto a buscar 6. Pulsar “MAPA” para situar la dirección en el mapa 7. Si la localización no es correcta proporcionar buscar otra localización o proporcionar una localización diferente manualmente sobre el mapa. 8. Proporcionar un nombre diferente si se requiere

	9. Pulsar “ACEPTAR” para confirmar la nueva información 10. Rellenar el resto de los datos 11. Pulsar “GUARDAR”
Escenario alternativo	
Nombre	Eliminar Promoción
Actores	Usuario activo
Objetivo	El usuario añade al sistema una nueva promoción
Precondiciones	Estar registrado en el sistema y haber iniciado una sesión
Postcondiciones	Se elimina del sistema la promoción seleccionada
Escenario básico	1. Acceder al menú 2. Seleccionar la opción de “Promociones” 3. Pulsar continuamente hasta que aparezca un formulario que nos solicita la eliminación o la cancelación de la operación de eliminar el registro 4. Pulsar la opción de “Eliminar”
Escenario alternativo	1. Acceder al menú 2. Seleccionar la opción de “Promociones” 3. Pulsar continuamente hasta que aparezca un formulario que nos solicita la eliminación o la cancelación de la operación de eliminar la promoción 4. Pulsar la opción de “Cancelar”

Nombre	Eliminar Evento
Actores	Usuario activo
Objetivo	El usuario añade al sistema un nuevo evento
Precondiciones	Estar registrado en el sistema y haber iniciado una sesión
Postcondiciones	Se elimina del sistema el evento seleccionado
Escenario básico	5. Acceder al menú 6. Seleccionar la opción de “Eventos” 7. Pulsar continuamente hasta que aparezca un formulario que nos solicita la eliminación o la cancelación de la operación de eliminar el registro 8. Pulsar la opción de “Eliminar”
Escenario alternativo	4. Acceder al menú 5. Seleccionar la opción de “Eventos” 6. Pulsar continuamente hasta que aparezca un formulario que nos solicita la eliminación o la cancelación de la operación de eliminar el evento 4. Pulsar la opción de “Cancelar”

Figura 29: Tabla de representaciones textuales de casos de uso

9.5 Modelo de datos

El modelo de datos que se utiliza en la aplicación consta de las siguientes entidades implicadas:

- User
- Formacion
- Promocion
- Evento

La relación existente entre ellas se puede apreciar en la figura indicada abajo. La representación se ha hecho con la estructura definida en Parse.com con la finalidad de mostrar el diseño construido en Parse.com.

Se han utilizado dos entidades con relación 1:1 entre User y Formacion. En este caso podría utilizarse una única entidad que albergara los datos de las dos entidades (que sería lo recomendable). Sin embargo, se han dividido en dos intencionadamente porque en un desarrollo futuro, componentes de la misma formación, también podrán realizar promociones y añadir eventos, con lo cual, la relación pasaría de 1:1 a n:1, es decir, varios usuarios pertenecerán a un mismo grupo.

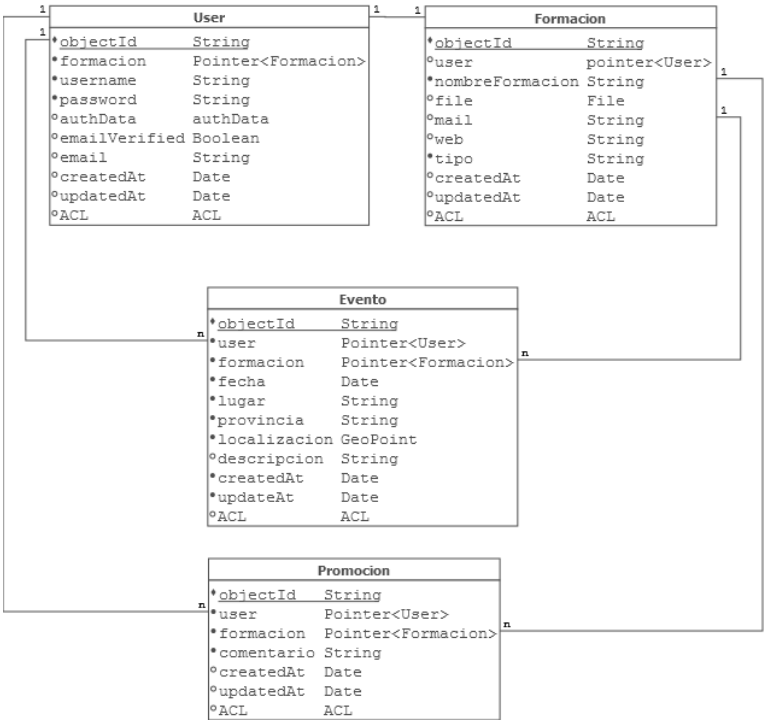


Figura 30: Modelo de datos

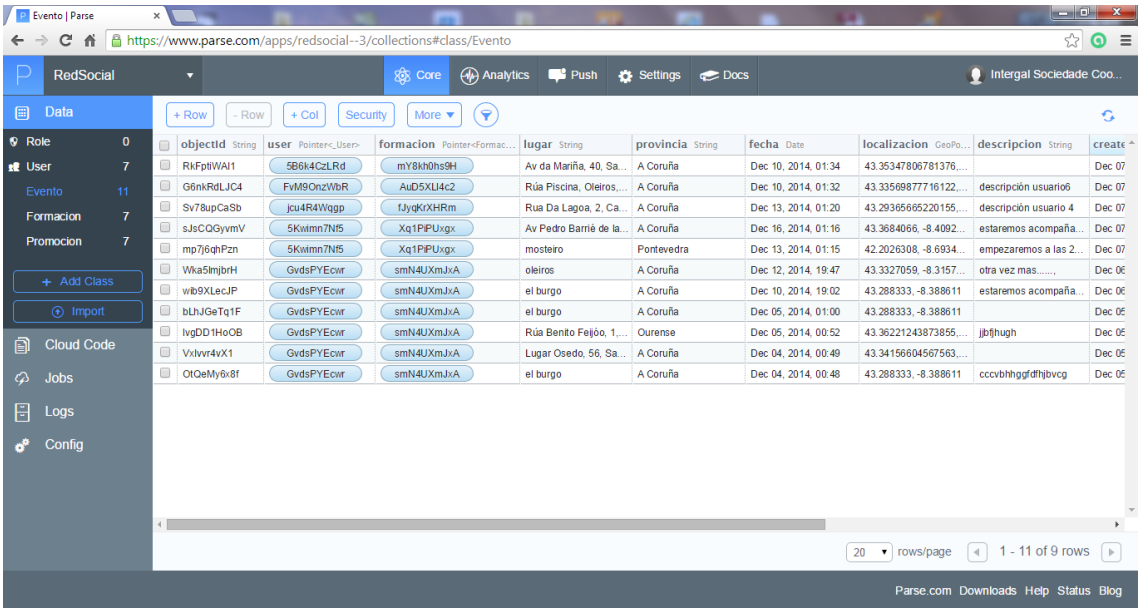


Figura 31: Pantalla del panel de control de Parse.com

9.6 Diagrama de clases

Este diagrama de clases muestra la relación de las principales clases utilizadas y desarrolladas en este proyecto.

En el apartado de "*Programación de las funcionalidades*" se describen un poco más en detalle las clases utilizadas en cada caso de uso.

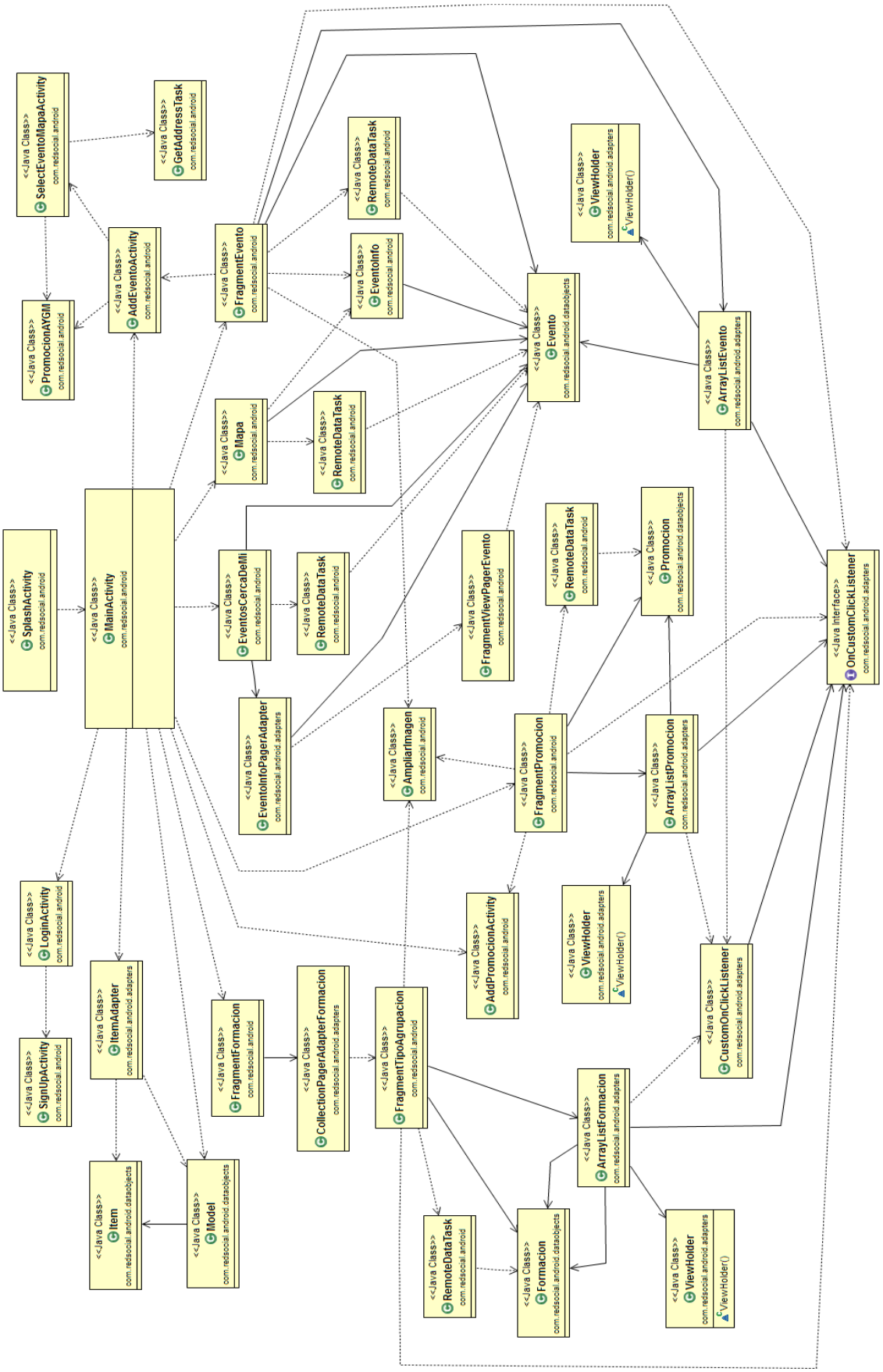


Figura 32: Diagrama de clases

10 Desarrollo

10.1 Entorno de trabajo

Para desarrollar esta aplicación se ha utilizado el siguiente entorno de hardware:

- Un portátil con W7
- Dispositivos móviles (SmartPhone, Tablet)

En el apartado de software se ha utilizado las siguientes herramientas:

- Java Runtime Environment 6.0.
- Eclipse (Eclipse Juno IDE for Java Developers).
- Android SDK (Google).
- Eclipse Plug-in (Android Development Tools - ADT).

Ya dentro del entorno de eclipse se necesitan varias librerías para manejar diversos recursos necesarios para esta aplicación:

- **Google Play Services**: necesarias para manejar los mapas y los servicios de geolocalización que se implementarán en esta aplicación.
- **Android-support-v7-appcompat**: necesarios para la compatibilidad de versiones antiguas de Android y de esta forma pueda ser ejecutada en diversos sistemas operativos desde la versión 2.2 (v8)
- **Joda time:** dispone de utilidades para controlar la duración del tiempo de publicación de las promociones.
- **Api Android Parse.com**: para conectarse con el Baas de Parse,com
- **ImageLoader**: para manejar de una manera eficiente la gestión de las imágenes.
- **TouchImageView**: para poder manejar las fotos haciendo zoom

10.2 Estructura de un proyecto Android

Cuando se crea un nuevo proyecto Android en Eclipse se genera automáticamente la estructura de carpetas necesaria para poder generar posteriormente la aplicación. Esta estructura será común a cualquier aplicación, independientemente de su tamaño y complejidad, y sigue una misma estructura básica, que se compone del código fuente (lógica de la aplicación), archivos de recursos y vistas (capa de presentación), librerías de código (librerías) y el Android Manifest (permisos).

En la siguiente imagen se puede ver los elementos creados para este proyecto:

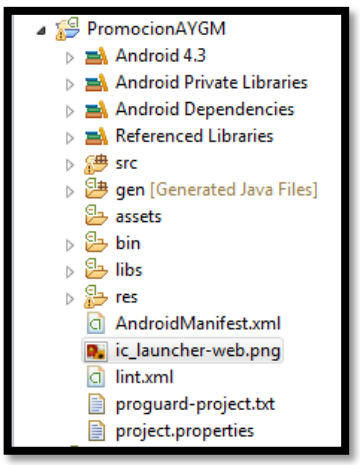


Figura 33: Estructura de un proyecto android

Carpeta /src/

Se encuentra toda la lógica de aplicación y todas las clases programadas en JAVA. Dentro de ella se han definido distintos paquetes con el objetivo de organizar mejor las clases que comprenden nuestra capa de reglas de negocio.

Figura 34: Carpeta /src/

Android Library

Aquí se encuentran todas las librerías propias del SDK de Android. Dependiendo de la versión elegida al crear el proyecto tendrá una versión u otra. En este caso se trabaja con la versión 4.3.

Carpeta /libs/

Aquí están las librerías externas importados que se utilizan en este proyecto.

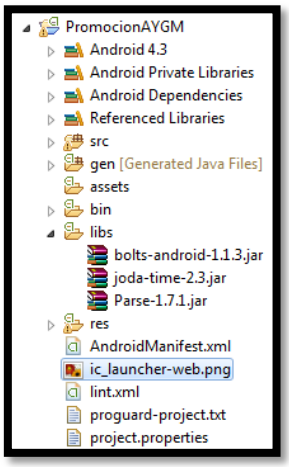


Figura 35: Carpeta /libs/

Carpeta /res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:

- /res/drawable/. Contienen las imágenes de la aplicación. Se puede dividir en /drawable-ldpi, /drawable-mdpi y /drawable-hdpi para utilizar diferentes recursos dependiendo de la resolución del dispositivo.
- /res/layout/. Contienen los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Se puede dividir en /layout y /layout-land para definir distintos layouts dependiendo de la orientación del dispositivo.

- /res/anim/. Contiene la definición de las animaciones utilizadas por la aplicación.
- /res/menu/. Contiene la definición de los menús de la aplicación.
- /res/values/. Contiene otros recursos de la aplicación: cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), ...
- /res/xml/. Contiene los ficheros XML utilizados por la aplicación, es decir, los ficheros que configuran el diseño gráfico de la interface de la aplicación.
- /res/raw/. Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

Carpeta /gen/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que se genera el proyecto, la maquinaria de compilación de Android genera una serie de ficheros fuente en java dirigido al control de los recursos de la aplicación.

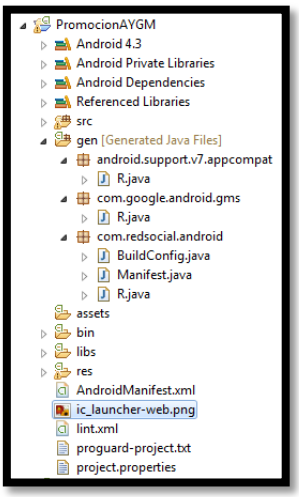


Figura 36: Carpeta /gen/

Carpeta /assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación como por ejemplo ficheros de configuración, de datos, etc.

La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/ es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Sin embargo, para los segundos, no se generarán ID y se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Se usa uno u otro según las necesidades de nuestra aplicación.

Fichero AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, …), sus componentes (pantallas, mensajes, …), o los permisos necesarios para su ejecución. En nuestro proyecto el fichero está definido de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.redsocial.android"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
```

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

<uses-feature android:name="android.hardware.camera" />

<!-- Para Google Maps V2. -->
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

<permission
    android:name="com.redsocial.android.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />

<uses-permission android:name="com.redsocial.android.permission.MAPS_RECEIVE" />

<application
    android:name="com.redsocial.android.PromocionAYGM"
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar" >
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyD_Bs0sH9gTGdDstqjZwFn3YPGWa0K_BWE" />

    <activity
        android:name="com.redsocial.android.SplashActivity"
        android:configChanges="keyboardHidden|orientation|screenLayout"
        android:label="@string/app_name"
        android:noHistory="true"
        android:screenOrientation="portrait" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="com.redsocial.android.MainActivity"
        android:label="@string/app_name"
        android:screenOrientation="portrait"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >
    </activity>
    <activity
        android:name="com.redsocial.android.AmpliarImagen"
        android:label="@string/title_activity_ampliar_imagen"
        android:theme="@android:style/Theme.Black.NoTitleBar" >
    </activity>
    <activity
        android:name="com.redsocial.android.Mapa"
        android:configChanges="keyboardHidden|orientation"
        android:label="@string/title_menu_eventos_mapa"
        android:screenOrientation="portrait"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
        android:windowSoftInputMode="stateHidden" >

        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.redsocial.android.MainActivity" />
    </activity>
    <activity
        android:name="com.redsocial.android.EventoInfo"
        android:label="@string/title_activity_evento_info"
        android:screenOrientation="portrait"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >

        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.redsocial.android.Mapa" />
    </activity>
    <activity
        android:name="com.redsocial.android.EventosCercaDeMi"
        android:label="@string/title_menu_cerca_de_mi"
        android:screenOrientation="portrait"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >

        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.redsocial.android.MainActivity" />
    </activity>
    <activity
        android:name="com.redsocial.android.LoginActivity"
```

```
            android:label="@string/title_activity_login"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
            android:windowSoftInputMode="adjustResize|stateVisible" >
        </activity>
        <activity
            android:name="com.redsocial.android.SignUpActivity"
            android:label="@string/title_activity_signup"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >
        </activity>
        <activity
            android:name="com.redsocial.android.AddEventoActivity"
            android:label="@string/title_activity_addEvento"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >
        </activity>
        <activity
            android:name="com.redsocial.android.AddPromocionActivity"
            android:label="@string/title_activity_addPromocion"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >

        </activity>
        <activity
            android:name="com.redsocial.android.SelectEventoMapaActivity"
            android:label="@string/title_activity_select_evento"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >
        </activity>
    </application>

</manifest>
```

Figura 37: Código del archivo Manifest.xml

Carpeta /bin/

Aquí se encuentran todos los archivos generados por la propia app. También se encuentra el ejecutable .apk. Es el archivo que se instalará en cualquier dispositivo para ejecutar la aplicación.

10.3 Diseño de las pantallas

Los componentes principales que forman la interface gráfica en Android son las Activities y los Fragments. Estos últimos aparecieron a partir de la versión 3.0 debido a que los dispositivos crecían rápidamente de tamaño y ello obligaba a reorganizar la información en la pantalla eficientemente. Los fragments dieron solución a este problema. Hoy en día son un elemento necesario para la programación sobre tablets.

El objetivo principal de una activity es interactuar con el usuario. Una actividad está conformada por la parte lógica (un archivo Java) y la parte gráfica (un archivo XML). La parte lógica es un archivo .java, que es la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad. La parte gráfica es un archivo XML que contiene todos los elementos que estamos viendo de una pantalla declarados con etiquetas parecidas a las del HTML.

Por otro lado, un fragment es como una porción de interface de usuario o vista que se integra en una activity. Con ello, se tiene la posibilidad de combinar múltiples fragmentos en una sola actividad o incluso reutilizar fragmentos en otras actividades. De esta manera, cada fragmento tendrá su propio ciclo de vida, recibirá sus propios eventos de entrada y se podrá agregar o eliminar mientras la activity de acogida esté en marcha.

Desde el momento que una activity aparece en la pantalla hasta el momento que se oculta, pasa por varias etapas, conocido como el ciclo de vida de una activity. La clase activity define una serie de eventos que controlan el ciclo de vida.

En este proyecto se utilizan las activities y los fragments que se encuentran en la librería Android-support-v7-appcompat para que versiones inferiores a la versión 3.0 puedan ejecutar aplicaciones que implementan fragments.

A continuación se muestran los ciclos de vida de las activities y los fragments.

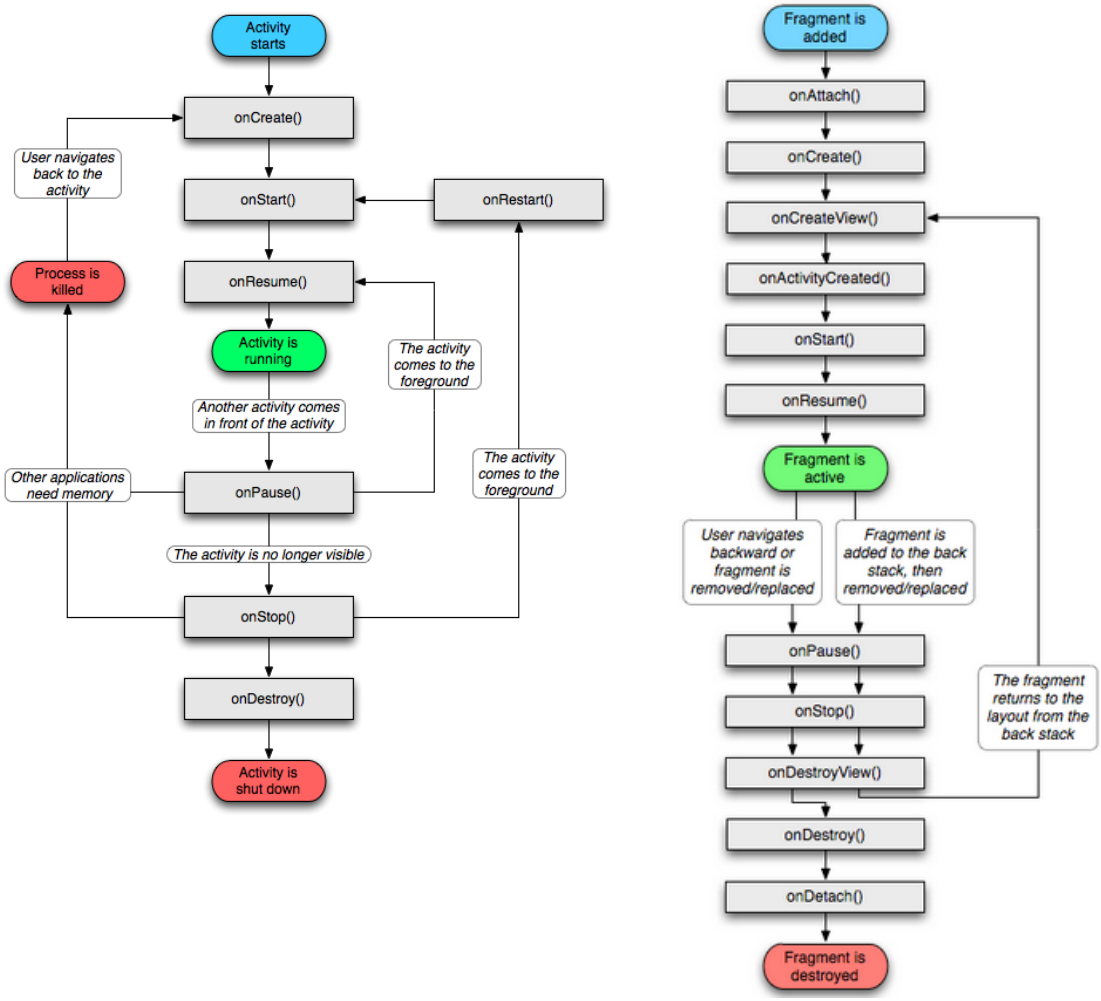


Figura 38: Ciclos de vida de las activities y los fragments

10.4 API Google Maps V2.

En este proyecto se utiliza la segunda versión de la API de Google Maps para Android. Esta nueva versión presenta muchas novedades interesantes, de las que cabe destacar las siguientes:

- Integración con los Servicios de Google Play (Google Play Services) y la Consola de APIs.
- Utilización a través de un nuevo tipo específico de fragment (MapFragment).
- Utilización de mapas vectoriales, lo que repercute en una mayor velocidad de carga y una mayor eficiencia en cuanto a uso de ancho de banda.
- Mejoras en el sistema de caché, lo que influye en la reducción de las famosas áreas en blanco que tardan en cargar.

- Los mapas son ahora 3D, es decir, podemos mover nuestro punto de vista de forma que lo veamos en perspectiva.

Para que se puedan mostrar los datos de Google Maps en un MapFragment, hay dos operaciones preliminares que son necesarias realizar:

1. Registrarse en el Servicio de Mapas de Google y obtener unos 40 caracteres de clave para la API de Mapas (*https://code.google.com/apis/console*).
2. Agregar al SDK el paquete Android Google Play Services (utilizando el Android SDK manager de Eclipse). Los archivos de soporte se instalarán en el <android-sdk> / extras / carpeta de google.
3. Importar el proyecto en nuestro workspace que está situado en <android-sdk-folder>/extras/google/google_play_services/ libproject/google-play-services_lib.
4. Establecer una dependencia entre el proyecto y Google Play Services. Para ello hay que ir a la barra de herramientas del Eclipse: Proyecto> Propiedades> Android> Biblioteca> Agregar> google play-services_lib.

10.5 Programación de las funcionalidades

En base al diagrama de uso y a las especificaciones adquiridas en todo el proceso de diseño se programan las diferentes clases que dan funcionalidad a la aplicación.

A continuación se explica una a una su función, acompañado de su diagrama de clases asociadas más importantes y un ejemplo gráfico del diseño de la pantalla generada:

10.5.1 SplashActivity

Esta clase extiende de la clase Activity y es la clase encargada de generar una pantalla de inicio de la aplicación. Esta clase realiza diversas funciones internas como la de recabar información del dispositivo: características del dispositivo (id, modelo,…), cuentas, el punto de localización y guardar todos estos datos en un fichero de preferencias.

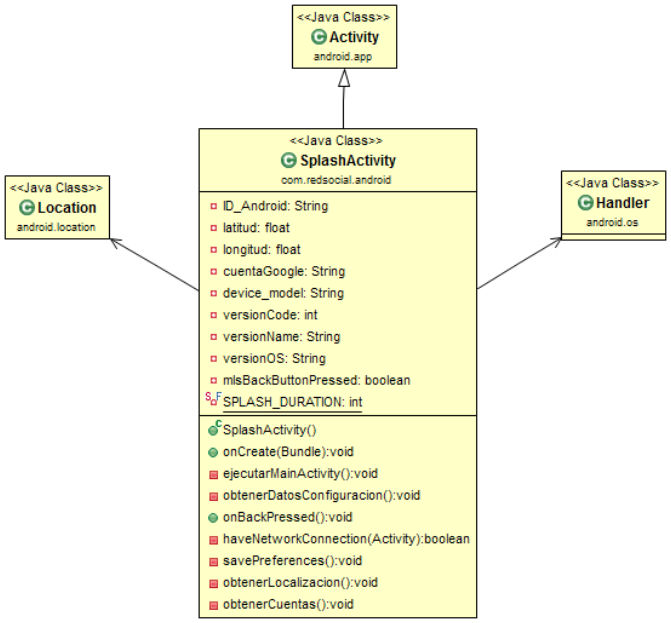


Figura 39: Relación de clases de SplashActivity.java

10.5.2 MainActivity

Esta es la clase principal de la aplicación o clase base que hereda de ActionBarActivity de la librería android.support.v7.app y contiene el menú lateral y el FragmentLayout que contendrá los fragments que se manejarán en la aplicación.

Las clases que permiten implementar el menú lateral son ActionBarrDrawerToggle y DraverLayout de la librería android.support.v4.widge y android.support.v4.app. Estas librerías permiten que funciones de versiones superiores puedan funcionar con la misma apariencia en dispositivos con versiones antiguas.

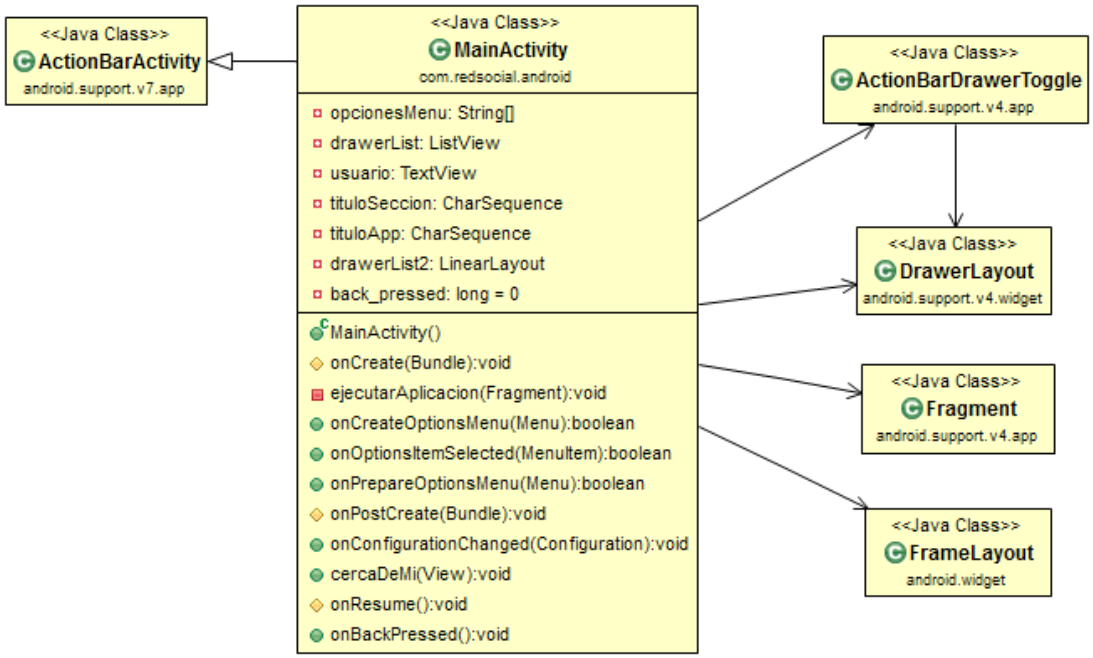


Figura 40: Relación de clases de MainActivity.java

Desde el menú lateral se accede a todas las pantallas que se manejan en la aplicación. Aquí se puede apreciar un diagrama que relaciona las clases que implementan los casos de uso diseñados anteriormente y las dependencias correspondientes. A continuación muestro el código que implementa la funcionalidad el menú lateral.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);

          //UI Componentes

          opcionesMenu = Model.getArray(getApplicationContext());
          drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
          drawerList = (ListView) findViewById(R.id.left_drawer2);
          drawerList2 = (LinearLayout) findViewById(R.id.left_drawer);
          usuario = (TextView) findViewById(R.id.usuario);

          ItemAdapter adapter = new ItemAdapter(this,R.layout.row, opcionesMenu);
          drawerList.setAdapter(adapter);

          drawerList.setOnItemClickListener(new OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> parent, View view,
                              int position, long id) {

                    drawerLayout.setBackgroundColor(Color.rgb(208, 208, 208));
                    fragment = null;
                    switch (position) {
                    case 0:
```

```
                    fragment = new FragmentFormacion();
                              tituloSeccion = getString(R.string.title_menu_formaciones);
                    break;
                    case 1:
                              fragment = new FragmentPromocion();
                              tituloSeccion = getString(R.string.title_menu_comentarios);
                    break;
                    case 2:
                              fragment = new FragmentEvento();
                              tituloSeccion = getString(R.string.title_menu_eventos);
                    break;
                    case 3:
                              startActivity(new Intent(getApplicationContext(), Mapa.class));
                    break;
                    case 4:
                              startActivity(new Intent(getApplicationContext(), LoginActivity.class));
                    break;
                    case 5:
                              startActivity(new Intent(getApplicationContext(), AddEventoActivity.class));
                    break;

                    case 6:
                              startActivity(new Intent(getApplicationContext(), AddPromocionActivity.class));
                    break;
                    }

          drawerList.setItemChecked(position, false);
          drawerLayout.closeDrawer(drawerList2);

          if (fragment!=null){
                    ejecutarAplicacion(fragment);
          }

          }});

          tituloApp = getTitle();

          drawerToggle = new ActionBarDrawerToggle(this,
                              drawerLayout,
                              R.drawable.ic_navigation_drawer,
                              R.string.drawer_open,
                              R.string.drawer_close) {

                    public void onDrawerClosed(View view) {
                              ActivityCompat.invalidateOptionsMenu(MainActivity.this);
                    }

                    public void onDrawerOpened(View drawerView) {
                              ActivityCompat.invalidateOptionsMenu(MainActivity.this);
                    }
          };

          drawerLayout.setDrawerListener(drawerToggle);

          getSupportActionBar().setDisplayHomeAsUpEnabled(true);
          getSupportActionBar().setHomeButtonEnabled(true);


          fragment = new FragmentPromocion();
          tituloSeccion = getString(R.string.title_menu_comentarios);
          ejecutarAplicacion(fragment);

          }
```

Figura 41: Código de la funcionalidad del menú lateral.

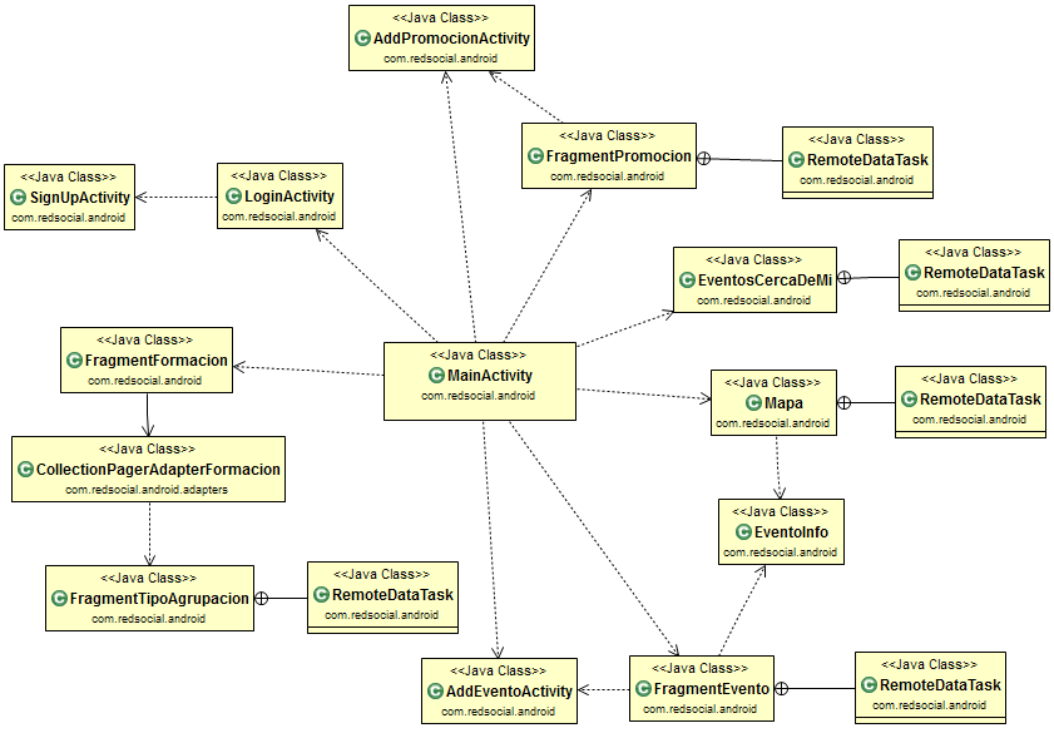


Figura 42: Relación de clases desde el MainActivity.java

Se puede ver el resultado de esta clase en las siguientes figuras. Se puede apreciar que la pantalla principal son las promociones. Estas promociones indican el nombre de la formación, la foto, el comentario promocional y el tiempo transcurrido desde que se ha publicado.

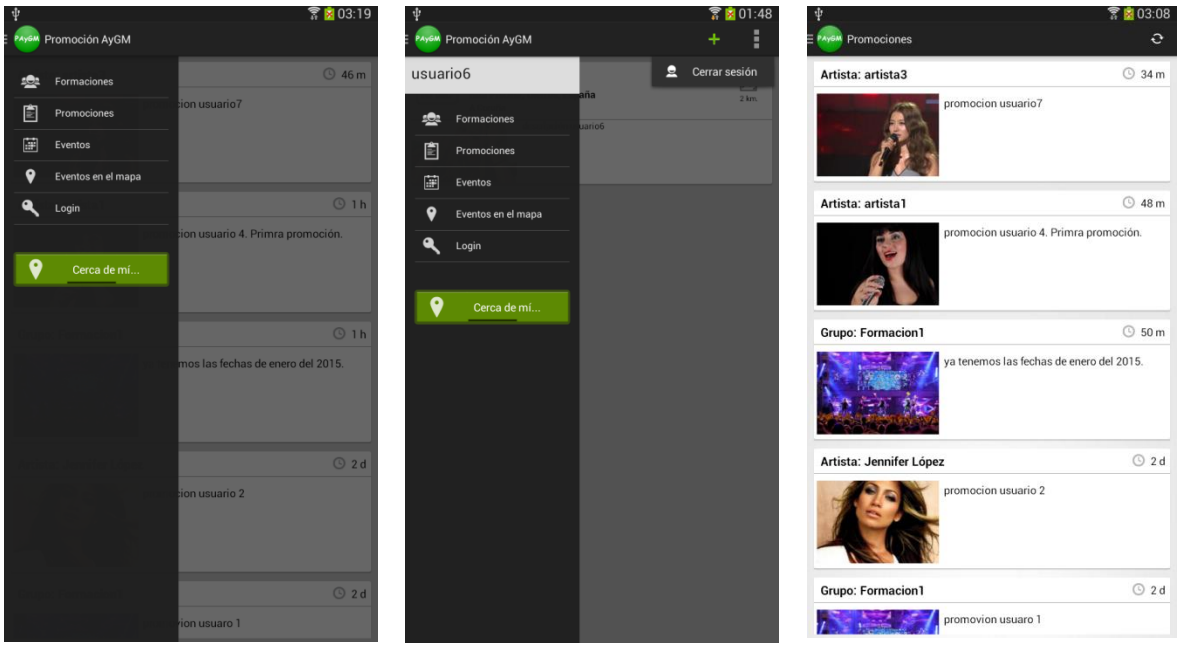


Figura 43: Pantallas del menú lateral y de la activity de promociones

El menú es desplegable lateralmente, con un diseño actual que se puede encontrar en la mayoría de las aplicaciones Google.

10.5.3 FragmentPromocion

Esta clase se encarga de mostrarnos las promociones. Para ello creamos un RemoteDataTask que se conecta con parse.com mediante su api y obtiene el resultado de promociones. A continuación mapea en un objeto de datos Promocion creando una lista de objetos promoción que se envía al ListAdapterPromocion para que nos mapee los datos con los objetos de la vista que serán mostrados en la pantalla.

Esta clase hereda de la clase ListFragment e implementa dos interfaces OnItemLongClickListener y OnCustomClickListener. Estas interfaces se utilizan para controlar la pulsación larga en un ítem del listView (que se utiliza para eliminar una promoción) y las pulsaciones de los elementos que se encuentran en el ítem del listView.

Cuando el usuario hace Login en el sistema esta clase muestra las promociones de este usuario y permite añadir y eliminar promociones, en caso contrario, muestra las promociones publicados por todos los usuarios sin posibilidad de añadir y eliminar promociones.

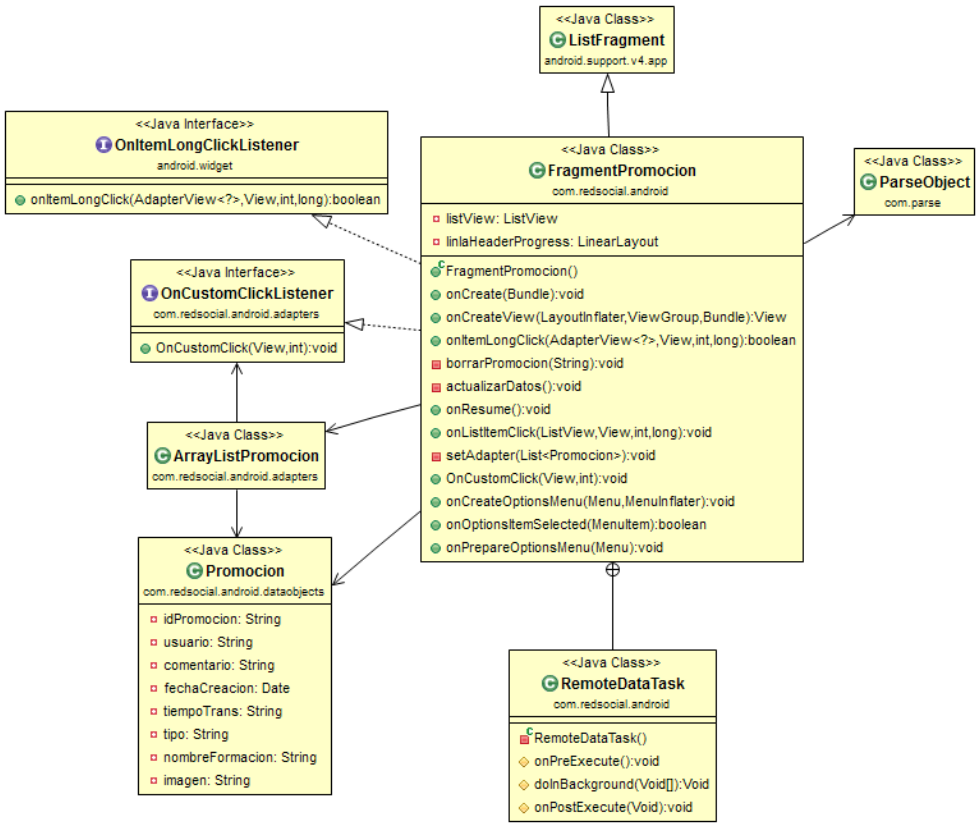
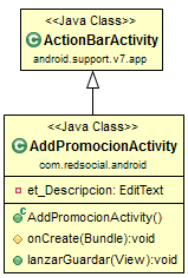


Figura 44: Relación de clases de la clase FragmentPromocion.java

En las siguientes figuras se puede ver las pantallas que representan las funcionalidades de esta clase, como la inserción de comentarios y la eliminación de los mismos, esta última operación pulsando en el ítem a eliminar durante un tiempo hasta que salga el dialogo que nos preguntará si queremos eliminarlo o no. Hay que decir que para realizar este tipo de operaciones se tiene que acceder al sistema como usuario registrado. Una vez registrado se accede a los comentarios que ese usuario ha publicado. A partir de ahí puede realizar las funciones antes citadas de añadir y



eliminar. Una vez que haga logout podrá ver todos los comentarios promocionales de todos los usuarios de la red.

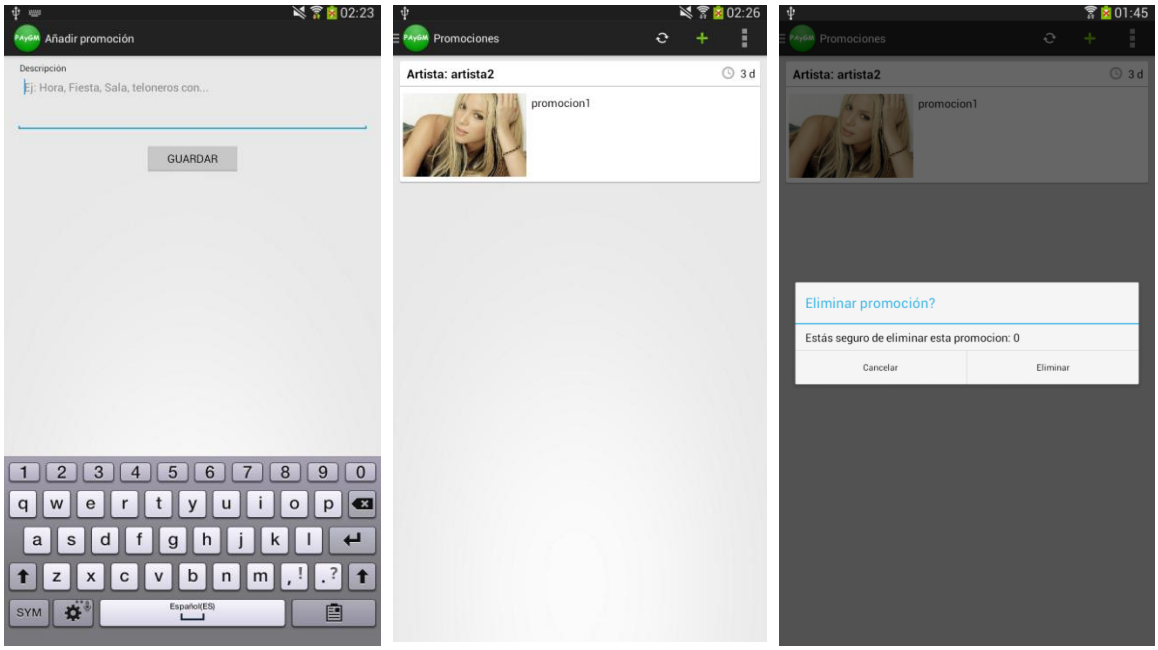


Figura 45: Pantallas promoción: añadir, lista y eliminar

10.5.4 FragmentEvento, AddEventoActivity y SelectEventoMapActivity

Esta clase gestiona los eventos. Igual que la clase de promociones, consigue los datos de eventos de Parse.com y los mapea con el objeto de datos Evento. Genera una lista de eventos que los manda al adapter para luego mapear con los objetos de fichero XML de eventos y mostrarlos en la vista del fragment.

Desde esta clase se accede a la Activity AddEventoActivity que es la encargada de localizarnos el lugar en el mapa o indicar la posición de localización manualmente. Esta clase hereda de ListFragment y al igual que la clase FragmentEvento implementa dos interfaces que controlan las pulsaciones realizadas en los ítems del listView.

En las siguientes figuras se puede ver las relaciones de clases y pantalla de eventos con las operaciones de añadir, eliminar eventos y selección de la localización del evento. Esta última funcionalidad recae en la clase SelectEventoMapaActivity. Esta clase se encarga de mostrarnos la localización en base a la dirección dada. En caso de que quisiéramos proporcionar una localización más determinada a la obtenida por la dirección dada, podemos indicarlo mediante posición en el mapa. En este caso los indicadores pin cambian de color para identificar la posición proporcionada manualmente en el mapa.

La función de añadir un evento la realiza la clase AddEventActivity. Esta clase y la clase SelectEventoMapActivity que hereda de ActionBarActivity.

A continuación se representan la relación de clases de las clases comentadas los códigos más representativos de las funciones de estas clases.

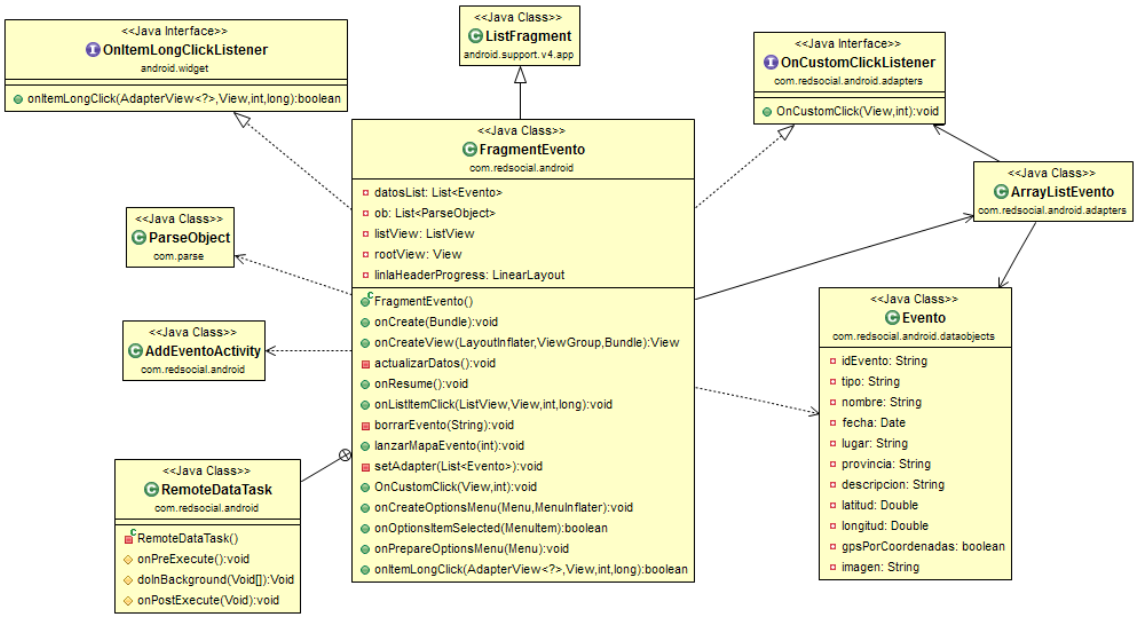


Figura 46: Relación de clases de la clase FragmentEvento.java

Muestra del código para obtener datos de Evento de Parse.

```
// Tarea remota asíncrona para obtener los datos
        private class RemoteDataTask extends AsyncTask<Void, Void, Void> {
            @Override
            protected void onPreExecute() {
                super.onPreExecute();

                linlaHeaderProgress.setVisibility(View.VISIBLE);
                }

                @Override
                protected Void doInBackground(Void... params) {
                    // Create the array
                   datosList = new ArrayList<Evento>();
                try {


                   Calendar calendar =Calendar.getInstance();
                             calendar.set(Calendar.HOUR,0);
                             calendar.set(Calendar.MINUTE,0);
                             calendar.set(Calendar.SECOND,0);
                             calendar.set(Calendar.MILLISECOND,0);

                    // Localiza la table "Eventos" en Parse.com
                    ParseQuery<ParseObject> query = new ParseQuery<ParseObject>("Evento");
                    query.setLimit(100);
                    query.orderByAscending("fecha");
                    query.whereGreaterThanOrEqualTo("fecha",calendar.getTime());

                    // Comprueba si se existe un usuario logeado. En caso afirmativo selecciona
                    // solo los eventos del usuario, de lo contrario, los selecciona todos
                    if (ParseUser.getCurrentUser()!=null)
                             query.whereEqualTo("user",ParseUser.getCurrentUser());

                    query.include("formacion");
                    ob= query.find();


                    for (ParseObject evento : ob) {

                        ParseObject formacion = evento.getParseObject("formacion");
                        ParseFile image = (ParseFile) formacion.get("file");
                        ParseGeoPoint location = new ParseGeoPoint();
                        location= evento.getParseGeoPoint("localizacion");

                        // Mapea los datos de parse con el dataobject Evento
                        Evento map = new Evento();
                        map.setIdEvento((String) evento.getObjectId());
                        map.setNombre((String) formacion.get("nombreFormacion"));
                        map.setTipo((String) formacion.get("tipo"));
                        map.setLugar((String) evento.get("lugar"));
                        map.setProvincia((String) evento.get("provincia"));
```

```
                    map.setFecha( ((Date) evento.get("fecha")));
                    map.setDescripcion((String) evento.get("descripcion"));
                    map.setLatitud(Double.valueOf(location.getLatitude()));
                    map.setLongitud(Double.valueOf(location.getLongitude()));
                    map.setImagen(image.getUrl());
                    datosList.add(map);
                }
            } catch (ParseException e) {
                Log.e("Error", e.getMessage());
                    e.printStackTrace();
                }
                return null;
            }

            @Override
            protected void onPostExecute(Void result) {
              setAdapter(datosList);

        }
    }
```

Figura 47: Muestra del código para obtener datos de Evento de Parse

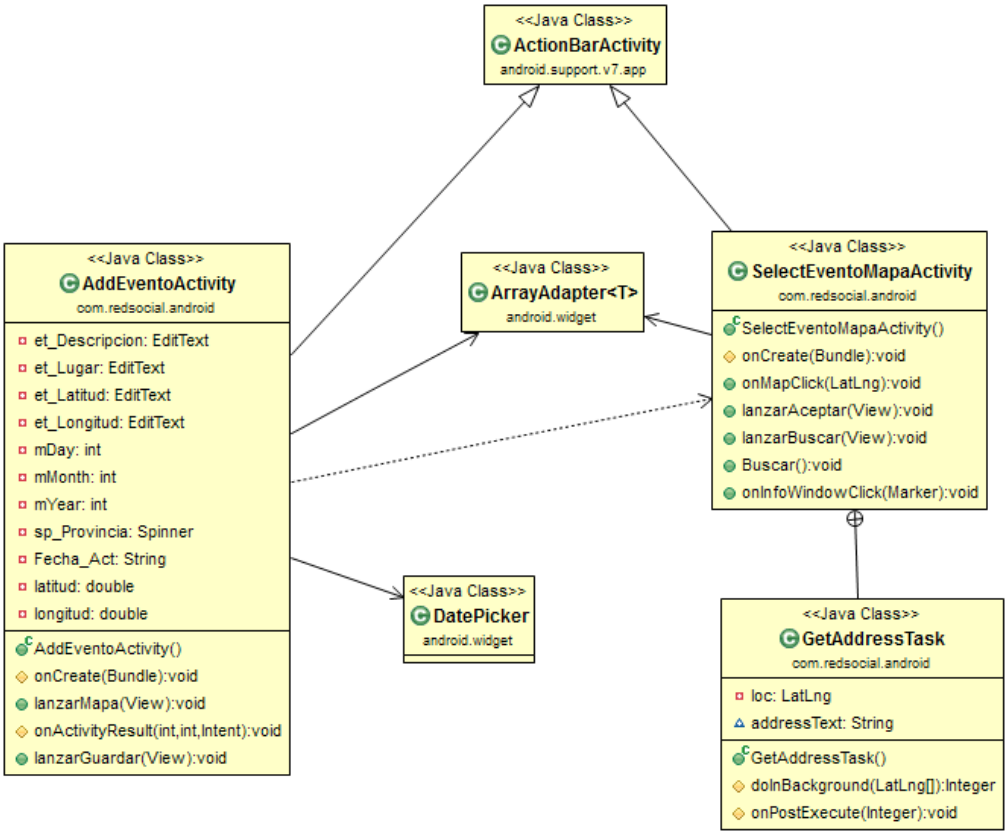


Figura 48: Relación de clases de la clase AddEventoActivity.java

Este es el parte del código que se utiliza en AddEventoActivity para grabar un evento en Parse.

```
        // Crea una barra de progreso
        final ProgressDialog dialog = new ProgressDialog(this);
        dialog.setMessage(getString(R.string.progress_addEvento));
        dialog.show();


        latitud = Double.valueOf(et_Latitud.getText().toString());
        longitud = Double.valueOf(et_Longitud.getText().toString());

        //Se crea un parse ParseGeoPoint
        ParseGeoPoint pgp = new ParseGeoPoint();
        pgp.setLatitude(latitud);
        pgp.setLongitude(longitud);
```

```
            // Se crea un objeto Parse
          ParseObject evento = new ParseObject("Evento");
          evento.put("fecha", date.getTime());
          evento.put("lugar", et_Lugar.getText().toString());
          evento.put("provincia", sp_Provincia.getSelectedItem().toString());
          evento.put("localizacion",pgp);
          evento.put("descripcion", et_Descripcion.getText().toString());
          evento.put("formacion", ParseUser.getCurrentUser().get("formacion"));
          evento.put("user", ParseUser.getCurrentUser());

          // Llama al método de SignUp de Parse
          evento.saveInBackground();

            // cierra la pantalla
            finish();
```

Figura 49: Parte del código para grabar un evento en Parse.com

El siguiente código se utiliza para eliminar un evento.

```
 /*
  * Borrar un evento seleccionado
  */
 private void borrarEvento(String evento){

                    ParseQuery<ParseObject> query = ParseQuery.getQuery("Evento");
                    query.getInBackground(evento, new GetCallback<ParseObject>() {
                      public void done(ParseObject object, ParseException e) {
                        if (e == null) {
                              object.deleteInBackground(new DeleteCallback() {

                                  @Override
                                  public void done(ParseException e) {
                                if (e == null) {
                                Toast.makeText(getActivity()," deleted",
                                            Toast.LENGTH_SHORT).show();
                                onResume();
                              } else {
                                Toast.makeText(getActivity()," not deleted",
                                           Toast.LENGTH_SHORT).show();
                                e.printStackTrace();
                              }

                                                  }
                                          });
                          } else {
                            // algún error producido
                          }
                      }
                    });

          }
```

Figura 50: Parte del código para eliminar un evento en Parse.com

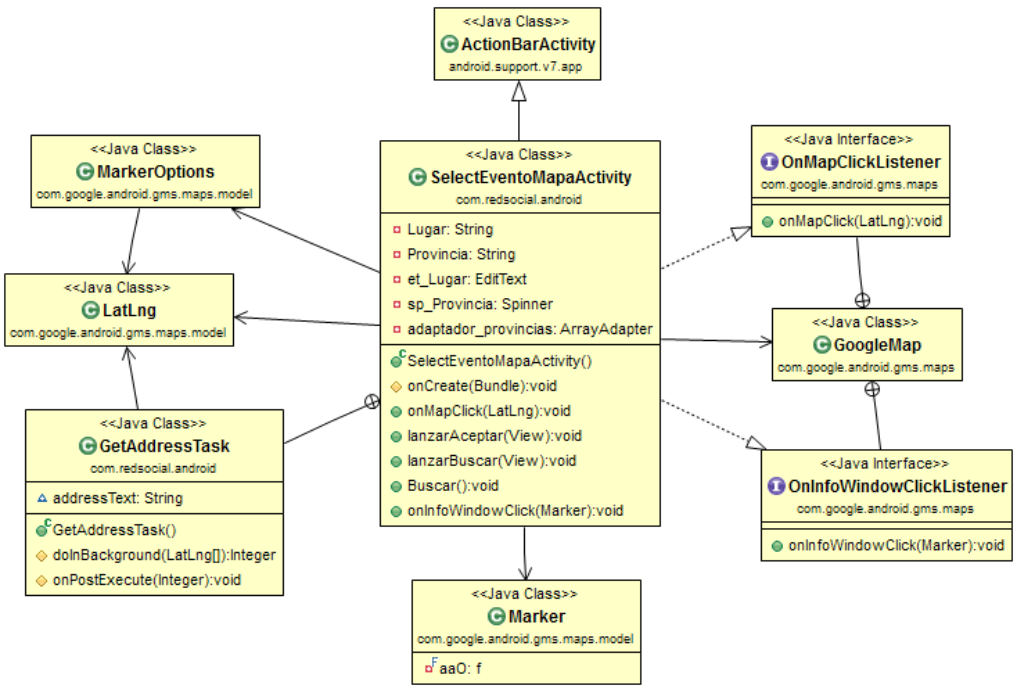


Figura 51: Relación de clases de la clase SelectEventoMapa.java

Este código es el que se implementa en SelectEventoMapActivity para reliazar el Geocoder, es decir, obtener la latitud y longitud en base a una dirección y también el Geocoder inverso que hace la función contraria.

```
  /*
   *  Busca una localizacion en funcion del nombre dado: Geocoder
   */
  public void Buscar()
  {
      Double double1;
      Double double2;
      Geocoder geocoder;
      double1 = Double.valueOf(42.771855000000002D);
      double2 = Double.valueOf(-7.4385579999999996D);
      geocoder = new Geocoder(this, Locale.getDefault());
      mapa.clear();
      List<Address> list=null;

            try {
                      list = geocoder.getFromLocationName((new   StringBuilder(
String.valueOf(et_Lugar.getText().toString()))).append(", ").append( sp_Provincia.getSelectedItem().toString()).toString(), 1);
            } catch (IOException e1) {
                      // TODO Auto-generated catch block
                      e1.printStackTrace();
            }

            try {

                       for (Address address:list)
          {
              double1 = Double.valueOf(address.getLatitude());
              double2 = Double.valueOf(address.getLongitude());

              marker=mapa.addMarker((new MarkerOptions()).position(new LatLng(double1.doubleValue(), double2.doubleValue()))
                                          .draggable(true)
                                  .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN))
                                  .title(et_Lugar.getText().toString()));
          }
          latLng = new LatLng(double1.doubleValue(), double2.doubleValue());
          CameraPosition cameraposition = (new com.google.android.gms.maps.model.CameraPosition.Builder())
                       .target(latLng).zoom(18F).bearing(0.0F).tilt(60F).build();
          mapa.setMapType(4);
          CameraUpdate cameraupdate = CameraUpdateFactory.newCameraPosition(cameraposition);
          mapa.animateCamera(cameraupdate);
          return;
      }
      catch(NullPointerException nullpointerexception)
      {
          nullpointerexception.printStackTrace();
      }

}
```

Figura 52: Busca una localización en función del nombre dado: Geocoder

```
          /*
           *  Tarea Geocoder inverso: se obtiene la dirección con la localización
           */
          public class GetAddressTask extends AsyncTask<LatLng, Void, Integer>{
            private LatLng loc;
            String addressText;
            @Override
            protected Integer doInBackground(LatLng... params) {
                int mFinalFlag=0;
                loc=params[0];
                String filterAddress = "";

                // Geocoder inverso
                Geocoder geoCoder = new Geocoder(getApplicationContext(), Locale.getDefault());
                try {
                    List<Address> addresses = geoCoder.getFromLocation(loc.latitude,
                            loc.longitude, 1);

                    if (addresses!=null&&addresses.size() > 0) {
                        Address address = addresses.get(0);
                        addressText = String.format(
                            "%s, %s, %s",
                            address.getMaxAddressLineIndex() > 0 ? address.getAddressLine(0) : "",
                            address.getLocality(),
                            address.getCountryName());
                    }
                } catch (IOException ex) {
                } catch (Exception e2) {
                    e2.printStackTrace();
                }
                return mFinalFlag;
            }
```

```
        @Override
        protected void onPostExecute(Integer result) {
           mapa
            .addMarker(
                new MarkerOptions()
                        .position(loc)
                        .draggable(true)
                        .icon(BitmapDescriptorFactory
                                .defaultMarker(BitmapDescriptorFactory.HUE_AZURE))
                        .title(addressText))
           .showInfoWindow();
           super.onPostExecute(result);
         }
 }
```

Figura 53: Tarea Geocoder inverso: se obtiene la dirección con la localización

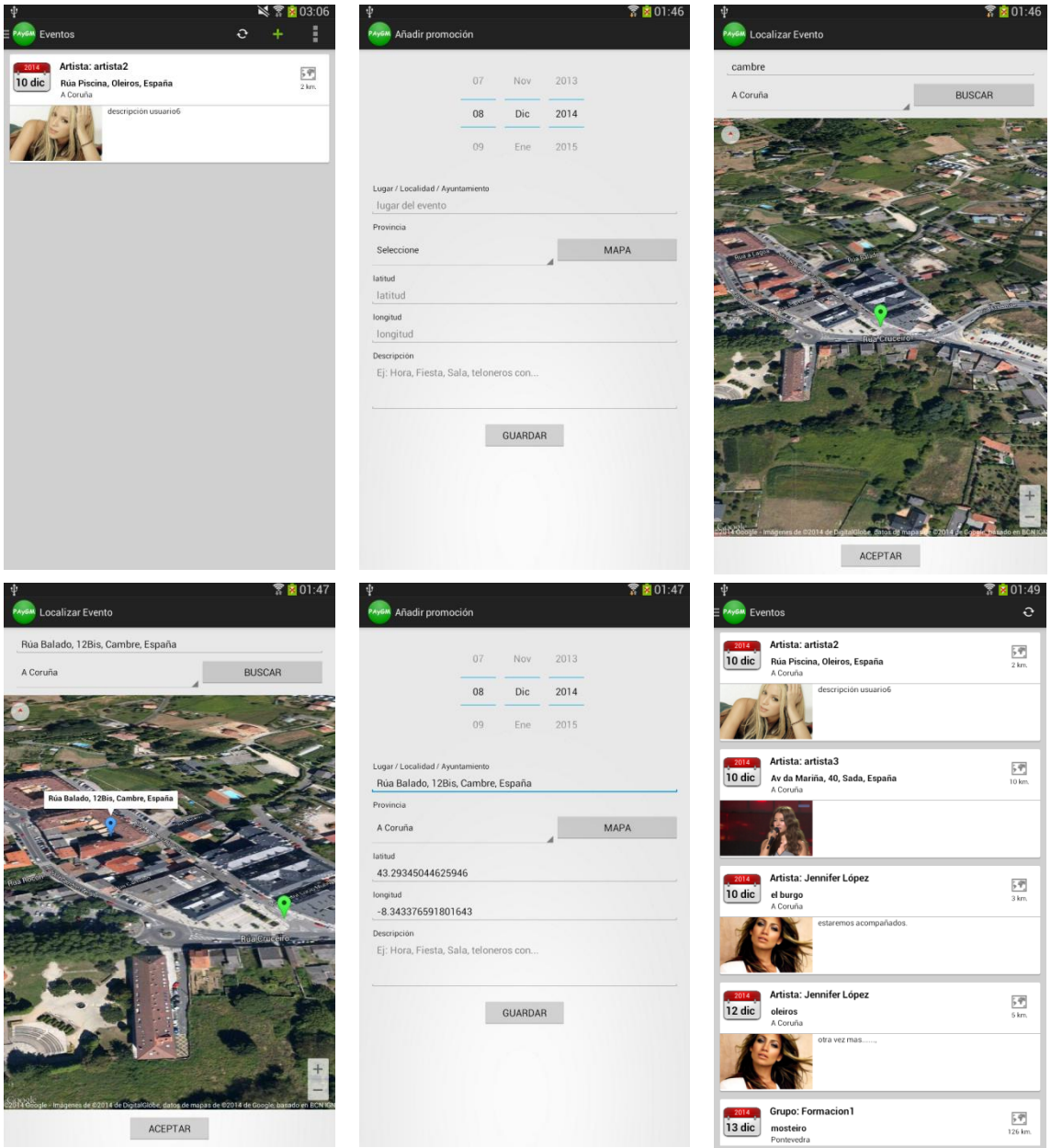


Figura 54: Pantallas de Evento: localizar y añadir un evento

10.5.5 FragmentFormacion y FragmentTipoAgrupacion

Esta clase se encarga de mostrarnos las formaciones añadidas al sistema mediante el registro al sistema y hereda de Fragment. Depende de otra clase FragmentTipoAgrupacion que es realmente quien obtiene los datos de Parse.com.

FragmentFormacion implementa un interface TabListener que mediante el adapter CollectionPagerAdapterFormacion permite que se muestren las formaciones según el tipo de agrupación definida (Grupo, Artista). A continuación se puede ver la relación de clases que se relacionan con FrgamentFormacion y con FragmentTipoAgrupacion.

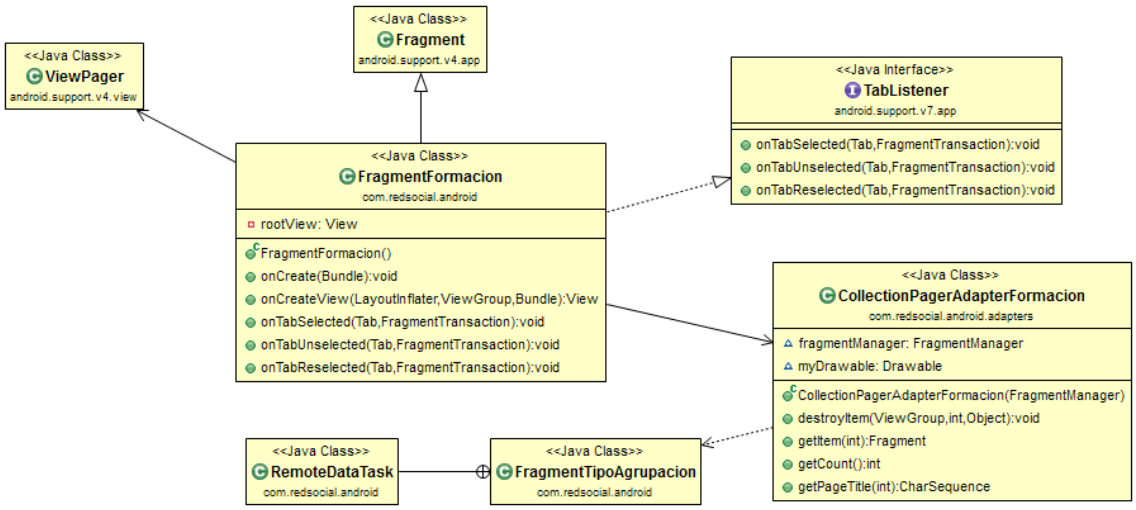


Figura 55: Relación de clases de la clase FragmentFormacion.java

Este es el código de FragmentFormacion.java.

```
package com.redsocial.android;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.view.ViewPager;
import android.support.v7.app.ActionBar.Tab;
import android.support.v7.app.ActionBar.TabListener;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.redsocial.android.adapters.CollectionPagerAdapterFormacion;



/*
 * Presenta las formaciones en un tab segun la agrupacion: Grupo, Artista
 */
public class FragmentFormacion extends Fragment implements TabListener{


          // representing an object in the collection.
          CollectionPagerAdapterFormacion mCollectionPagerAdapterFormacion;
          ViewPager mViewPager;

    private View rootView;

          @Override
          public void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);

                    setRetainInstance(true);
}
```

```
          @Override
          public View onCreateView(LayoutInflater inflater, ViewGroup container,
                  Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);

                  if (container == null) {
                     // Currently in a layout without a container, so no
                     // reason to create our view.
                     return null;
                  }

                     rootView = inflater.inflate(R.layout.swipe_tabs, container, false);

                        // ViewPager and its adapters use support library
                              mCollectionPagerAdapterFormacion =
                         new CollectionPagerAdapterFormacion(getActivity().getSupportFragmentManager());

                                    mViewPager = (ViewPager) rootView.findViewById(R.id.pager);
                                      mViewPager.setAdapter(mCollectionPagerAdapterFormacion);

        return rootView;
    }

          @Override
          public void onTabSelected(Tab tab,
                    FragmentTransaction fragmentTransaction) {
                    // When the given tab is selected, switch to the corresponding page in
                    // the ViewPager.
                    mViewPager.setCurrentItem(tab.getPosition());
          }

          @Override
          public void onTabUnselected(Tab tab,
                    FragmentTransaction fragmentTransaction) {
          }

          @Override
          public void onTabReselected(Tab tab,
                    FragmentTransaction fragmentTransaction) {
          }

}
```

Figura 56: Código FragmentFormacion.java

Aquí se puede ver el código de CollectionPagerAdapterFormacion.java

```
package com.redsocial.android.adapters;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.view.ViewGroup;

import com.redsocial.android.FragmentTipoAgrupacion;

public class CollectionPagerAdapterFormacion extends FragmentStatePagerAdapter {
          FragmentManager fragmentManager;

public CollectionPagerAdapterFormacion(FragmentManager fm) {
    super(fm);

    this.fragmentManager =fm;
}

@Override
public void destroyItem(ViewGroup container, int position, Object object) {
          super.destroyItem(container, position, object);
}


@Override
public Fragment getItem(int i) {

          Fragment fragment = new FragmentTipoAgrupacion();

          String tipo = "";
          switch (i){

          case 0:
                    tipo ="Grupo";
                    break;
          case 1:
                    tipo ="Artista";
                    break;

          }

    Bundle args = new Bundle();
    args.putString("Tipo", tipo);
    fragment.setArguments(args);
```

```
    return fragment;
}

@Override
public int getCount() {
    return 2;
}

@Override
public CharSequence getPageTitle(int position) {

          String name = "";
          switch (position){

          case 0:
                    name ="Grupo";
                    break;
          case 1:
                    name ="Artista";
                    break;

          }

    return name;
}

}
```

Figura 57: Código de CollectionPagerAdapterFormacion.java

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"

    android:layout_width="match_parent"
    android:layout_height="match_parent" >

     <android.support.v4.view.PagerTabStrip
        android:id="@+id/pager_tab_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#33b5e5"
        android:textColor="#fff" />

</android.support.v4.view.ViewPager>
```

Figura 58: Código xml de swipes_tabs.xml

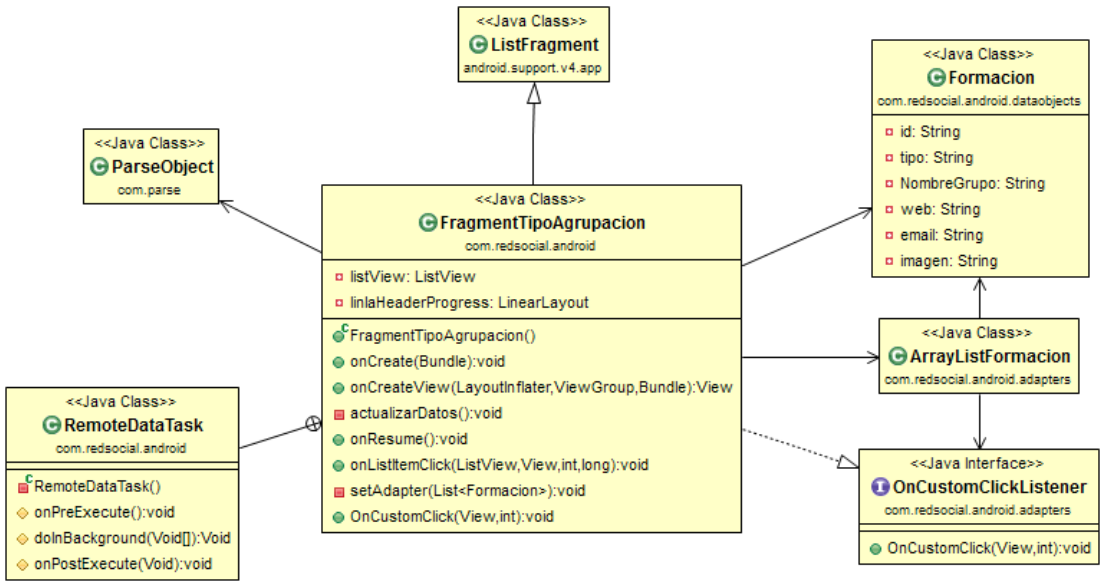


Figura 59: Relación de clases de la clase FragmentTipoAgrupación.java

Estas son las pantallas que se obtienen mediante la implementación de la clase FragmentFormacion y FragmentTipoAgrupacion.

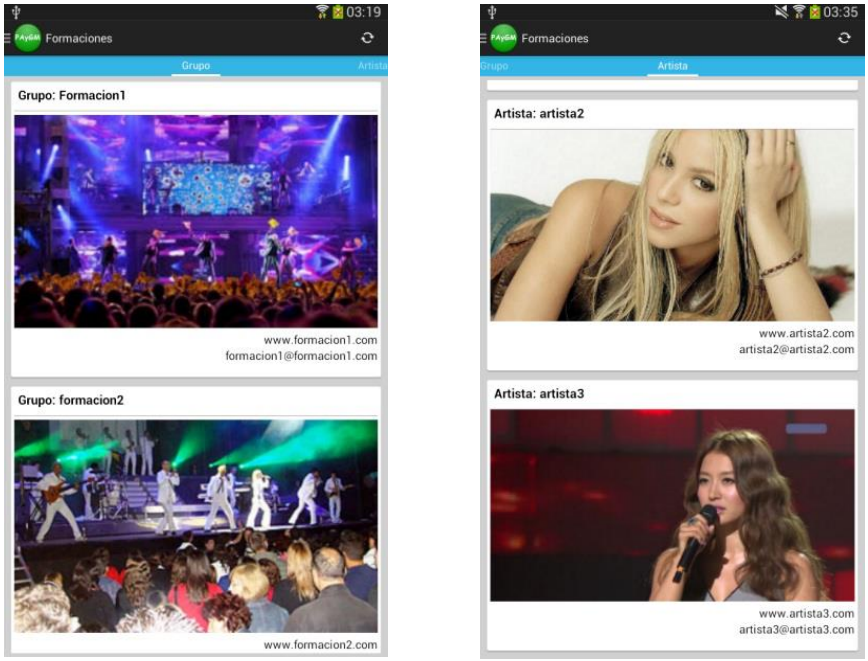


Figura 60: Pantallas de formaciones

10.5.6 Mapa

Con esta clase se consigue la funcionalidad de mostrar en un mapa los eventos que se produce en un día determinado. Esta clase extiende de la clase ActionBarActivity e implementa varias clases para el control del manejo del mapa. Estas clases de control son listeners encargados de registrar los eventos producidos sobre el mapa cuando se interactúa con él.
Una vez que tenemos desplegados los eventos en el mapa podemos pulsar en uno de ellos que nos mostrará información del evento. Si pulsamos en el infoWindow del punto nos lleva a otra actividad EventoInfo que nos dará más información del evento.

En el "*Anexo III: Código completo de Mapa.java"* se puede ver el código completo de esta clase.

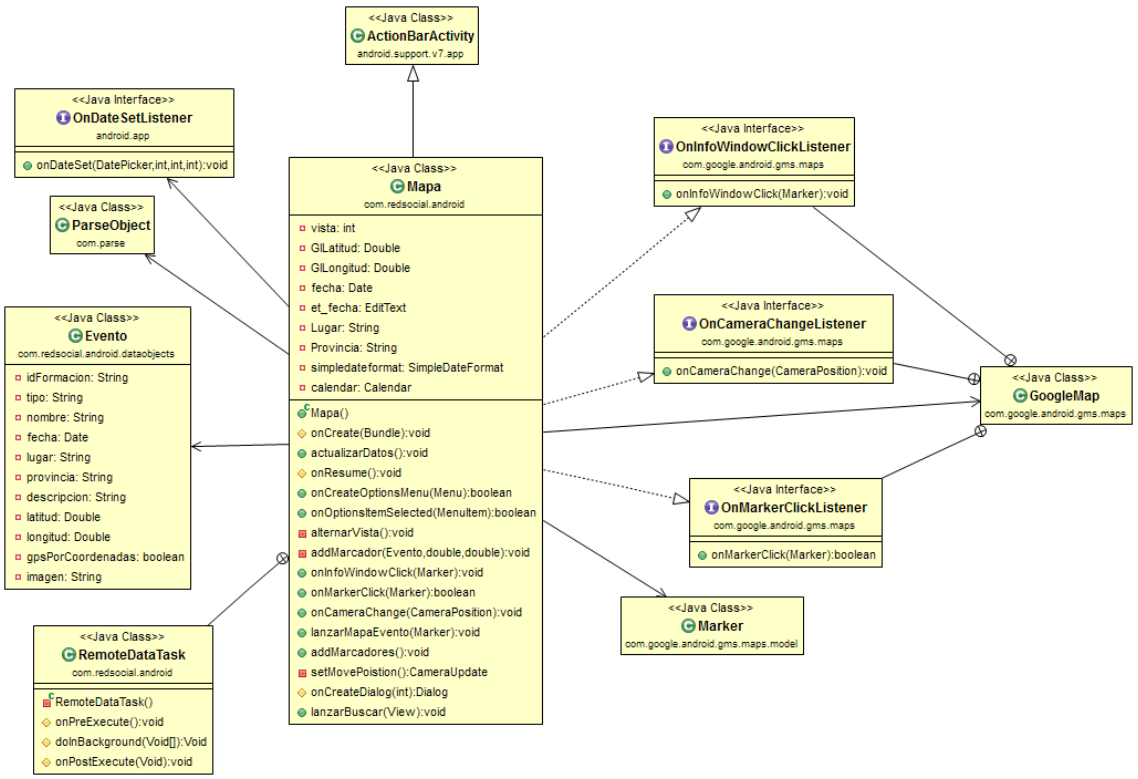
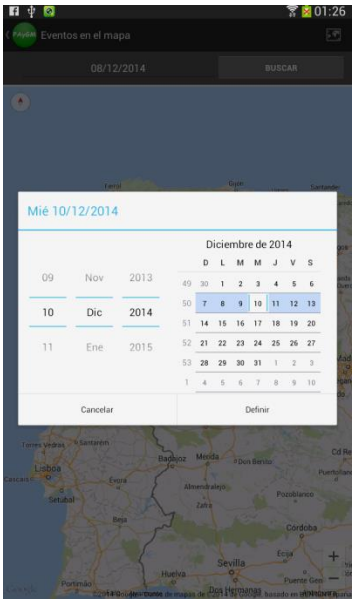
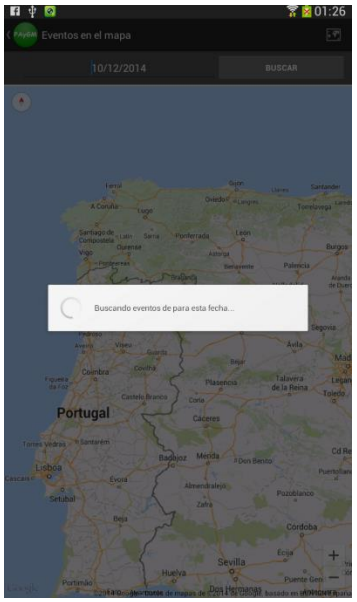


Figura 61: Relación de clases de la clase Mapa.java





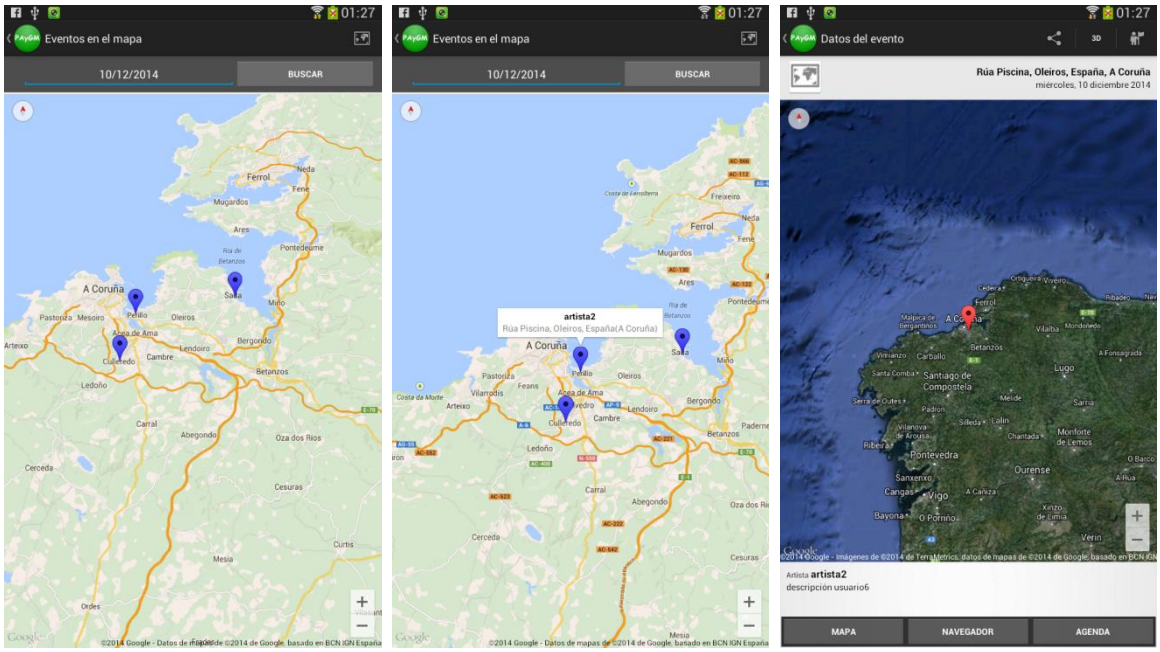


Figura 62: Pantallas relacionadas con la clase Mapa y EventoInfo

10.5.7 EventoInfo

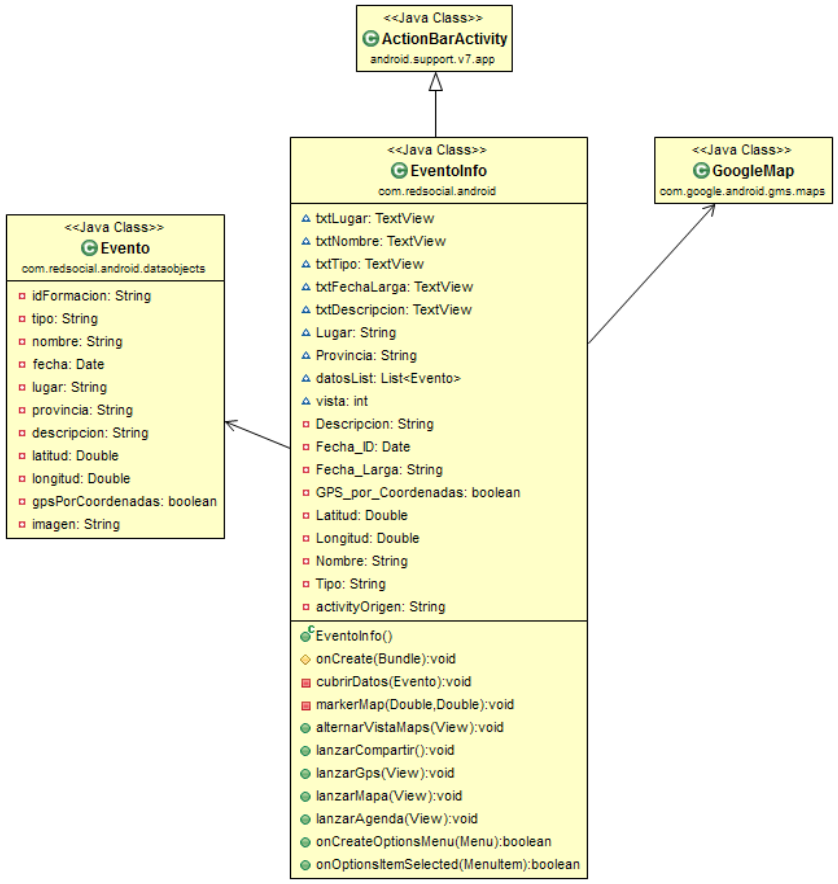


Figura 63: Relación de clases de la classe EventoInfo.java

Esta clase se encarga de dar información más exhaustiva de la localización de los eventos. Dispone de métodos que nos permite ejecutar otras aplicaciones y pasarle los datos de localización, como son: maps, navigation, StreetView. Ver distintos tipos de mapas, en 3D y compartir datos con otras aplicaciones.

Extiende de la clase ActionBarActivity e incluye Google Maps como un fragment dentro del fichero XML.

Aquí se puede ver algunas pantallas con las funciones implementadas en esta clase:

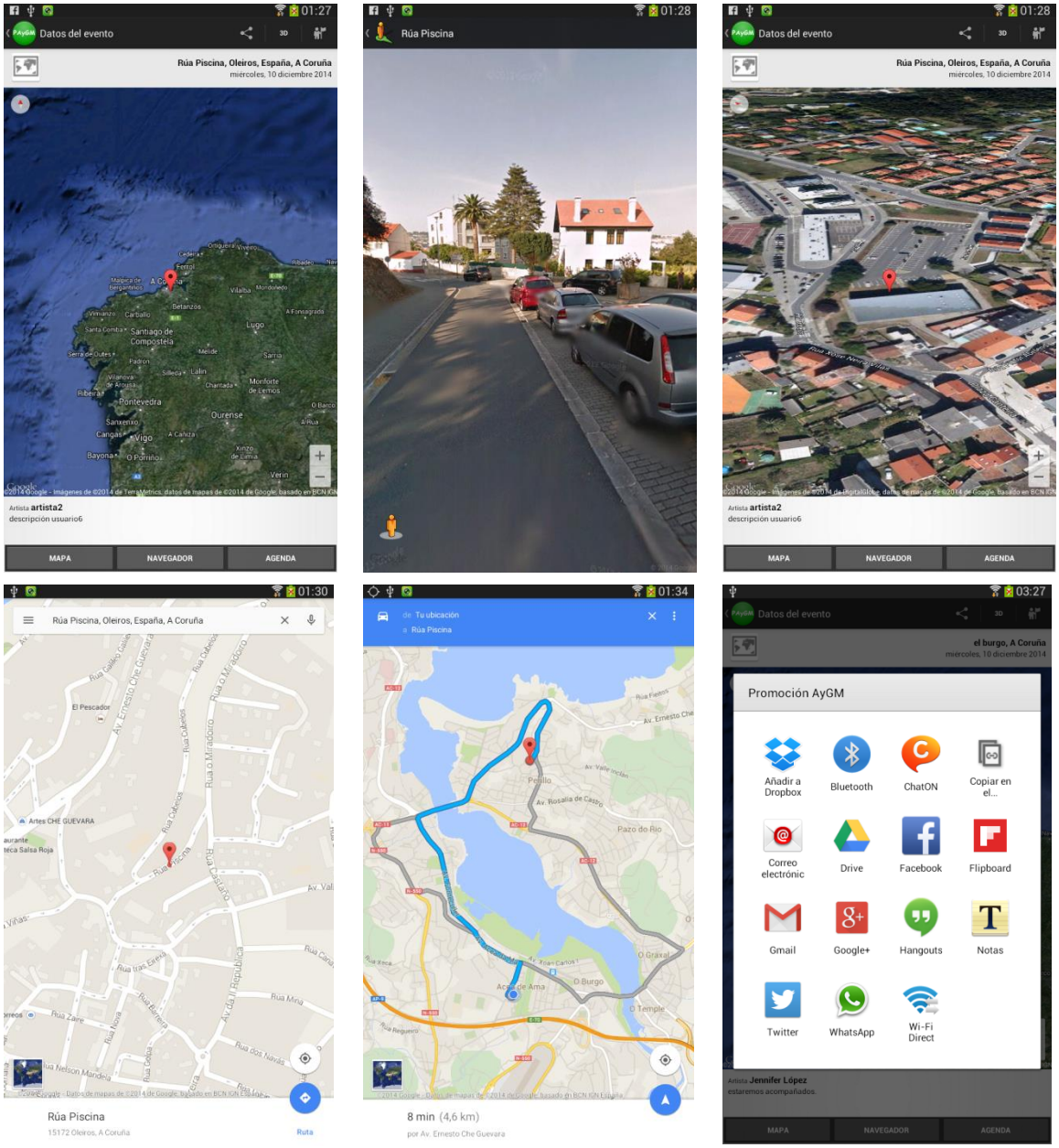


Figura 64: Pantallas relacionadas con la clase EventoInfo

10.5.8 EventosCercaDeMi

La funcionalidad de esta clase cubre mostrar todos los eventos que se producirán cerca del usuario dentro de un radio concreto que se marca con una circunferencia con el fin de que se pueda observar el área que ocupa. También se le ha dotado de una funcionalidad de un ViewPager que muestra la información del evento y la posición en el mapa según se van desplazando en el ViewPager, y viceversa, si se pulsa en el marcador dentro del mapa el ViewPager se desplaza al registro correspondiente.

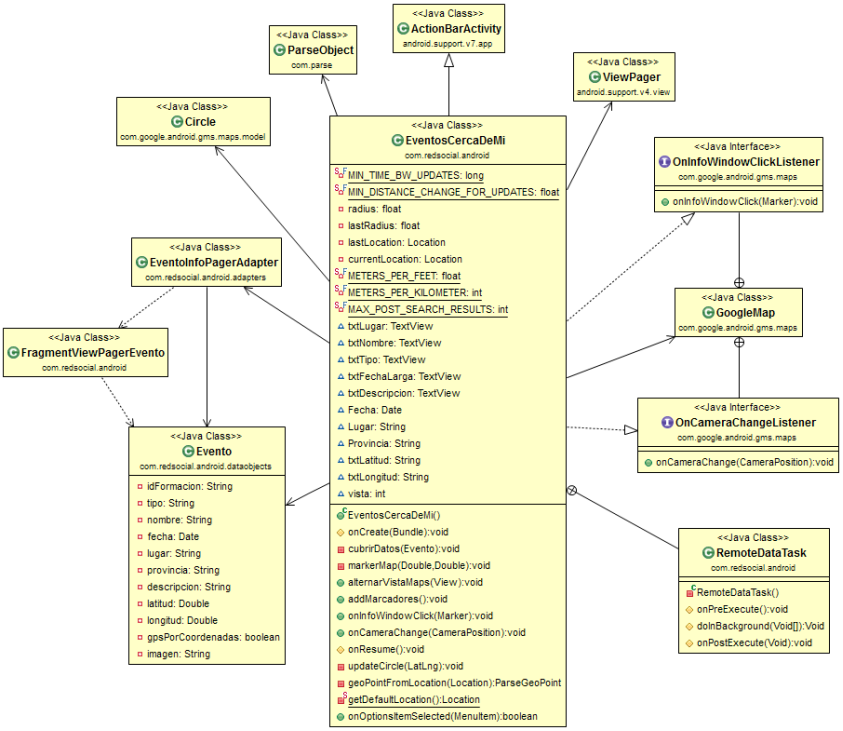


Figura 65: Relación de clases de la clase EventosCercaDeMi.java

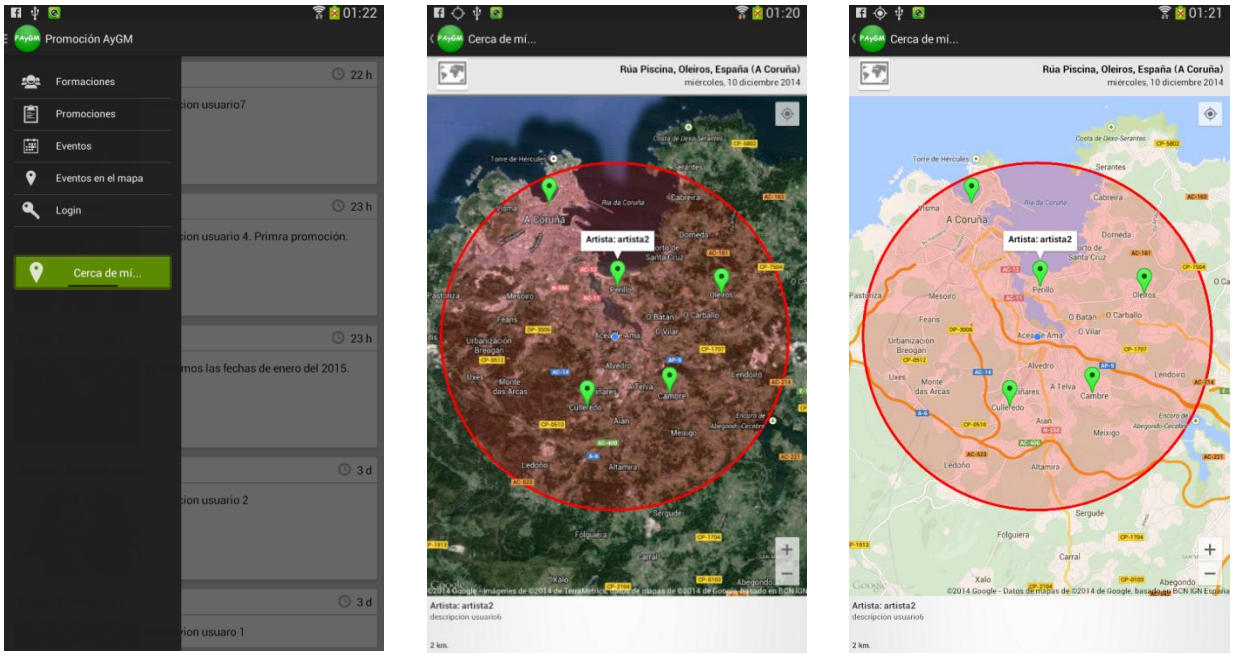


Figura 66: Pantallas relacionadas con la clase EventosCercaDeMi

10.5.9 LoginActivity y SignUpActivity

Estas son las clases que nos permiten identificarnos en el sistema o crear una nueva cuenta en caso de no estar registrado. Cuando creamos una nueva cuenta también creamos una formación, ya que la relación entre User y Formacion es de 1:1. También se ha implementado una función para elegir una foto tanto de la galería como de la cámara. Esta foto es subida a Parse.com por medio de su API.

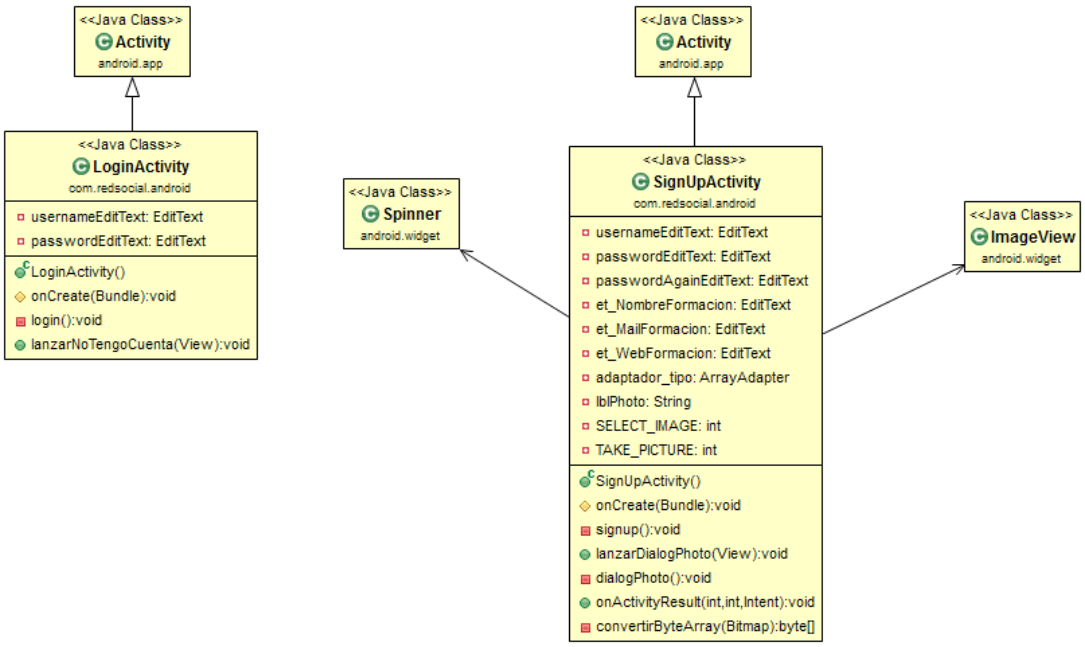
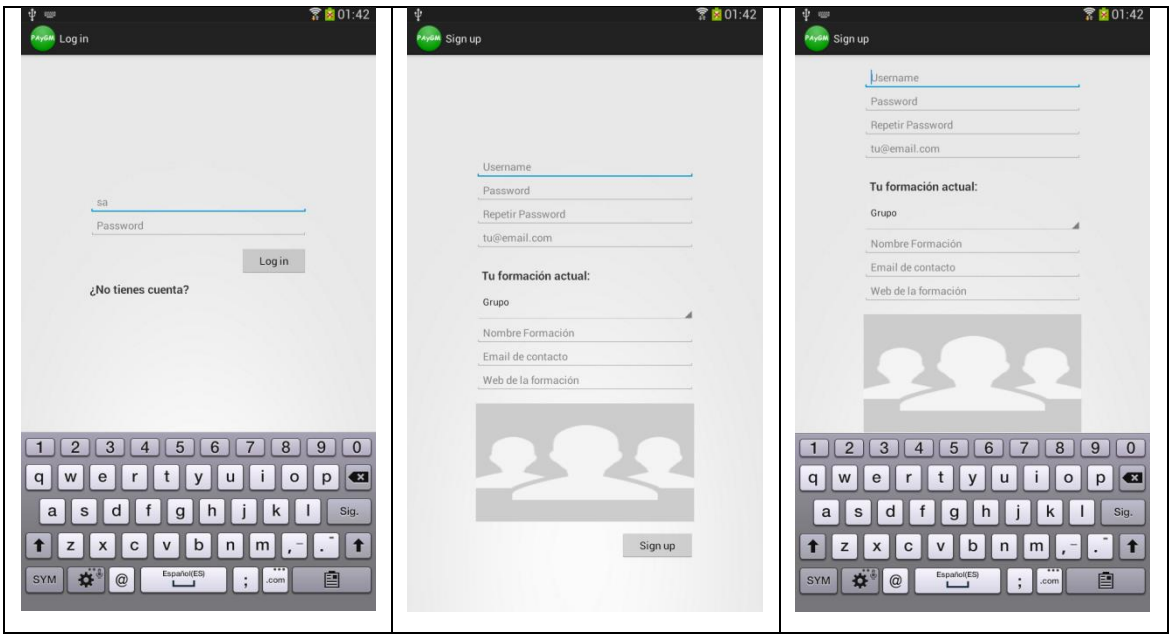


Figura 67: Relación de clase de LoginActivity.java y SignUpActivity.java



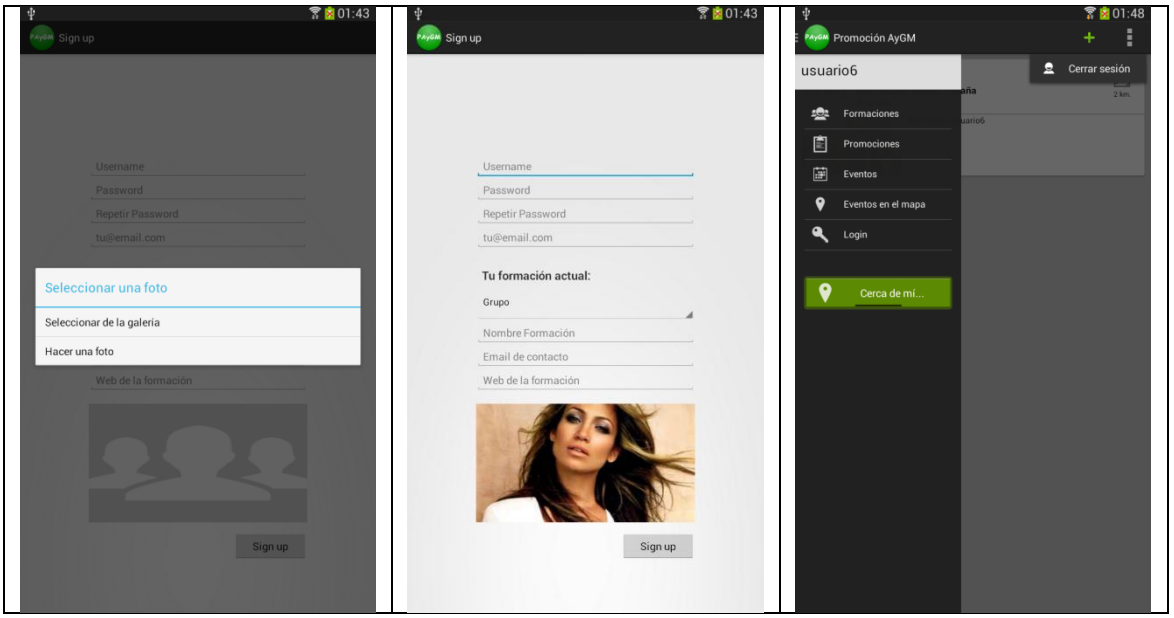


Figura 68: Pantallas relacionadas con las clases LoginActivity y SignUpActivity

10.5.10 PromocionAyGM

Esta clase extiende de la clase Application. La clase Application es el pegamento que une todas las actividades, servicios y recibidores en una entidad unificada. Se utiliza como contenedor de datos comunes utilizados por una gran mayoría de activities de la aplicación.

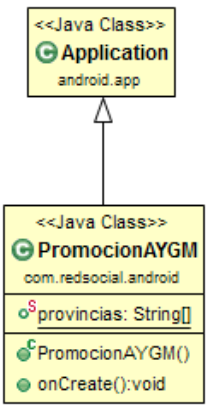


Figura 69: Relación de clases de la clase PromocionAYGM.java

10.5.11 AmpliarImagen

Esta clase permite manipular el tamaño de la imagen. Se utiliza la clase TouchImageView para facilitar esa función.

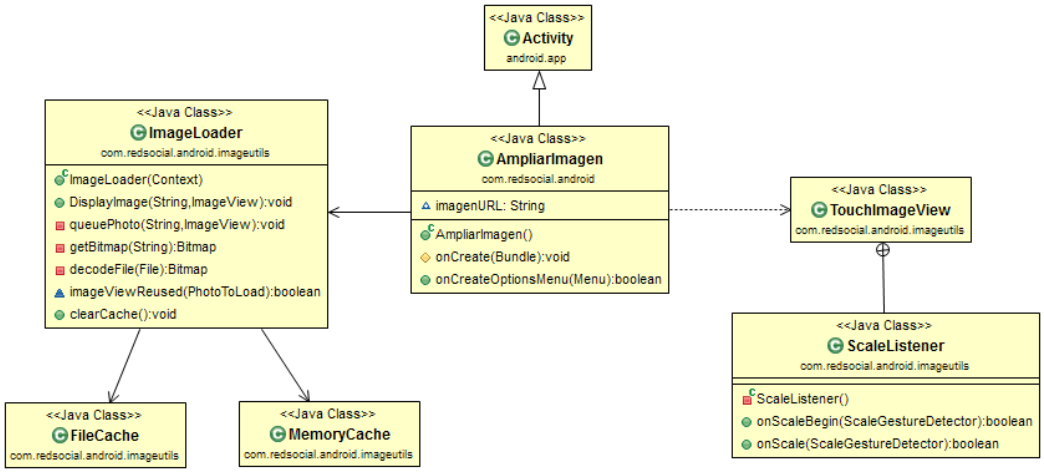


Figura 70: Relación de clases de la clase AmpliarImagen.java

Figura 71: Muestra de las pantallas relacionadas con la clase AmpliarImagen

11 Pruebas

En esta etapa se definirán un conjunto de pruebas para definir el correcto funcionamiento de la aplicación. Cabe decir que cuando se trata de pruebas automatizadas disponemos de muchos tipos de herramientas y frameworks orientados a asegurar la calidad del desarrollo de aplicaciones Android. Los enfoques más comunes que podemos encontrar son:

- **Google Testing Android:** Este es el framework incluido como parte de la plataforma Android (JUnit).
- **Robotium**: Black box de pruebas de integración para Android.
- **Robolectric**: Pruebas unitarias que se ejecutan fuera del emulador haciendo que las pruebas se ejecuten mucho más rápidas.

El uso de unos tipos u otros dependerá de las características y circunstancias del proyecto a testear.

En este proyecto se procede a utilizar el framework de testing de Android que no es más que una adaptación específica para Android del famoso framework JUnit estandar.

Cuando creamos un proyecto de Android con el asistente de Eclipse tenemos la opción de incluir casos de prueba en el propio proyecto. Otra manera muy común es la de separar las pruebas en un proyecto aparte. En este caso es la manera a seguir. Para ello se ha creado un nuevo proyecto Android Test Project y se seleccionó el proyecto PromocionAYGM. El nombre que se le ha asignado al proyecto de test es el mismo pero con la palabra "_Test" al final (**PromocionAYGM_Test**)

El código fuente de los tests se programa de forma manual. Las subclases principales sobre las que se suelen programar las pruebas unitarias son **ActivityInstrumentationTestCase2**, diseñada para realizar pruebas funcionales, y **ActivityUnitTestCase**, diseñada para realizar test de activities Android de forma aislada.

11.1 Pruebas con dispositivos

La aplicación se ha probado mediante dispositivos Android en las versiones 2.3 hasta la versión 4.3 tanto en SmartPhones de 4'' y 5'' como en tabletas de 7'' y de 10.1".

Para las pruebas se han tomado las especificaciones de los casos de uso. Todas las tareas definidas en la fase de análisis se han superado con éxito. A continuación se representa una tabla recogiendo las pruebas y su resultado.

Prueba	Descripción	Observaciones	Exito
Login	El usuario accede al sistema e inicia una sesión		
SignUp	El usuario se registra en el sistema, publica su formación e inicia una sesión		
Consultar Promociones (1)	Consultar los últimos comentarios promocionales publicados por cualquier usuario.	Se ha realizado con usuarios pasivos, es decir, usuarios que no necesitan iniciar sesión en el sistema.	
Consultar Promociones (2)	Consultar los últimos comentarios promocionales publicados por un usuario	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	
Consultar Eventos (1)	Consultar todos los eventos que se celebrarán desde el día que el usuario hace la consulta en adelante.	Se ha realizado con usuarios pasivos, es decir, usuarios que no necesitan iniciar sesión en el sistema.	
Consultar Eventos (2)	Consultar los eventos publicados por un usuario que se celebrarán desde el día que el usuario hace la consulta en adelante.	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	
Consultar Formación (1)	Consultar las formaciones (artista y grupos musicales) que existen en el sistema	Se ha realizado con usuarios pasivos, es decir, usuarios que no necesitan iniciar sesión en el sistema.	
Consultar Formación (2)	Consultar la ficha de una formación.	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	
Eventos en el Mapa	Consultar los eventos que se celebrarán un día determinado.		
Cerca de mi…	Consultar todos los eventos que se celebrarán en diferentes fechas desde el día que realiza la consulta, y además, que estén dentro de un radio deternminado de proximidad de la localización del usuario	Se ha realizado tanto con usuarios activos como con usuarios pasivos	

Evento Info	Mostrar el evento en detalle con opciones para compartir con otros usuarios y aplicaciones.	Se ha realizado las pruebas realizando las llamadas desde: "Eventos", "Eventos en el Mapa" y "Cerca de mí…"	
Add Promocion	El usuario que inicia una sesión añade al sistema una nueva promoción.	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	
Add Evento	El usuario que inicia una sesión añade al sistema un nuevo evento	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	
Eliminar Evento	El usuario que inicia una sesión elimina un evento.	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	
Eliminar Promocion	El usuario que inicia una sesión elimina una promoción.	Se ha realizado con usuarios activos, es decir con usuarios registrados en el sistema y que iniciaron una sesión.	

Figura 72: Tabla de pruebas de la aplicación realizadas en los dispositivos

11.2 Pruebas unitarias

Con las pruebas unitarias se trata de probar un componente en particular (es decir, la actividad) en forma aislada de otros componentes. Estas pruebas tienden a ser más simples y más rápidas que las pruebas de integración.

En este caso he desarrollado una prueba unitaria en la activity LoginActivity. La clase creada para el test es LoginActivityUnitTest.

Lo que intenta testear esta clase es que cuando se accede a la pantalla de login aparece por defecto un nombre de usuario. Este usuario se envía por medio de un intent a la pantalla de SignUpActivity, en el caso de querer crear una nueva cuenta.

Para ello se han definido tres métodos testeadores. Estos métodos comienzan por la palabra "test".

Este método comprueba los valores iniciales de los objetos clave a testear en la aplicación para garantizar las condiciones iniciales.

```
public void testPreconditions() {
                assertNotNull(buttonId);
}
```

Este comprueba la existencia del botón Log in y de que el texto que incluye sea el correcto.

```
public void testLayout() {
    buttonId = com.redsocial.android.R.id.action_button;
    assertNotNull(activity.findViewById(buttonId));
    Button view = (Button) activity.findViewById(buttonId);
    assertEquals("Incorrecta la etiqueta del botón", "Log in", view.getText());
}
```

Por último este método comprueba el el intent iniciado

```
  public void testIntentTriggerViaOnClick() {
    buttonId = com.redsocial.android.R.id.action_no_tienes_cuenta;
    TextView view = (TextView) activity.findViewById(buttonId);
    EditText username = (EditText) activity.findViewById(com.redsocial.android.R.id.username);
    username.setText("usuario1");
```

```
    assertNotNull("El boton no puede ser nulo", view);

    view.performClick();

    // Comprueba el intent iniciado
    Intent triggeredIntent = getStartedActivityIntent();
    assertNotNull("Intent es nulo", triggeredIntent);
    String data = triggeredIntent.getExtras().getString("username");

    assertEquals("Datos incorrectos pasados via intent",
        "usuario1", data);
}
```

11.3 Resultado de las pruebas unitarias

El resultado de las pruebas unitarias sobre la clase LoginActivityUnitTest es satisfactorio, como muestra la siguiente figura.

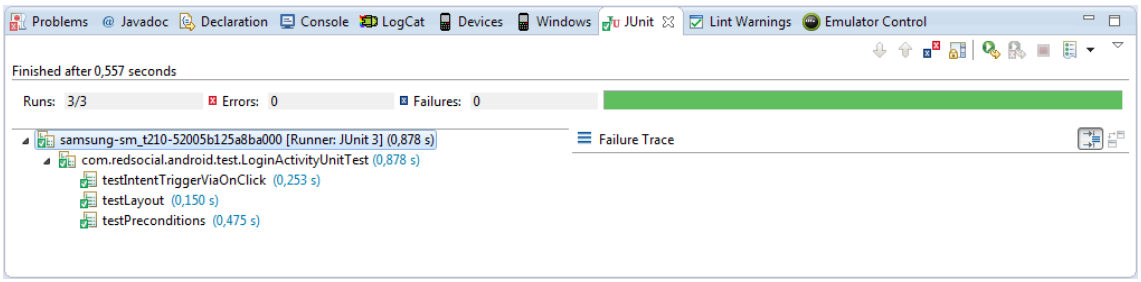


Figura 73: Resultado de las pruebas unitarias

11.4 Pruebas funcionales

Una clase de prueba más compleja es android.test.ActivityInstrumentationTestCase2. Con esta clase se pueden probar Activities con su entorno gráfico. La propia clase se encarga de crear el objeto Activity y de inicializarlo adecuadamente pasando por todos sus estados.

En el proyecto se han realizado dos test: uno sobre LoginActivityTest y otro sobre AddPromociónActivityTest.

En LoginActivityTest se valida los datos introducidos de username y password al pulsar el botón de Login.

En AddPromociónActivityTest se comprueba que al añadir una promoción los datos que se introducen son los mismos que se manejan.

11.5 Resultado de las pruebas funcionales

El resultado de las pruebas funcionales sobre las clases LoginActivityTest y AddPromocionActivityTest ha sido satisfactorio, como muestra las siguientes figuras.

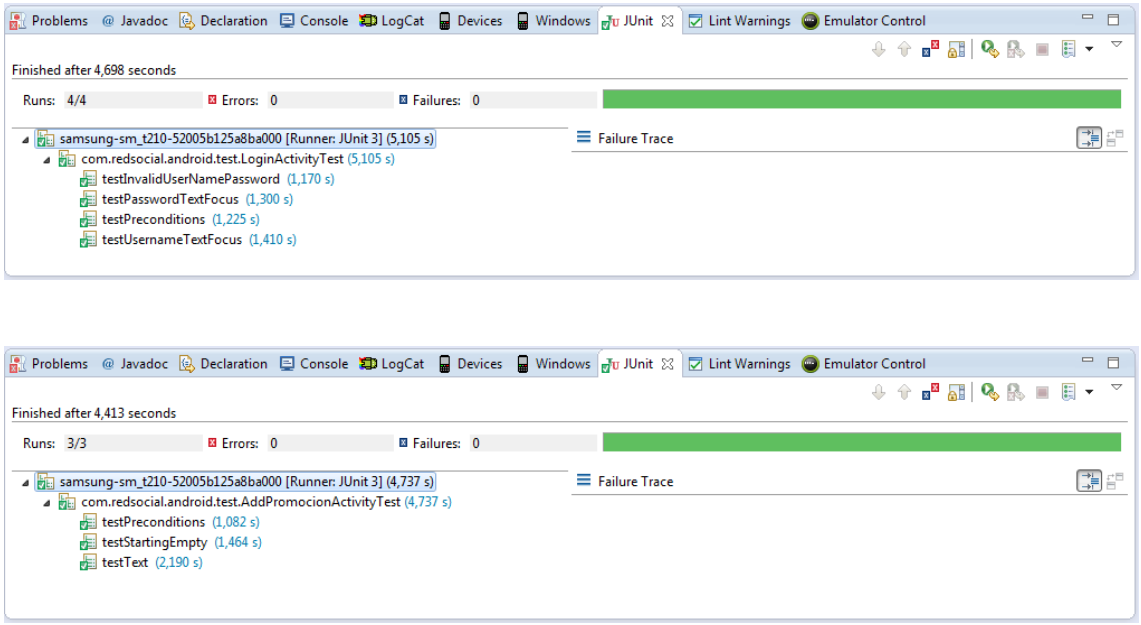


Figura 74: Resultados de las pruebas funcionales

CONCLUSIONES

Al término de este proyecto de ***Fin de Master*** se puede concluir que el resultado ha sido positivo y muy satisfactorio, tanto en la definición y planificación del mismo como en la elaboración y posterior desarrollo.

Los requisitos iniciales que se presentaron tras la elección del proyecto se han cumplido en su totalidad, incluso se han contemplado ciertas funcionalidades que no estaban previstas y definidas, pero que por el interés y necesidad que han suscitado en las pruebas del prototipo, fueron implementadas, no sin antes haber estudiado que su desarrollo no influyera negativamente en el estado del proyecto general en tiempo y forma.

Tampoco fue necesario tomar ninguna medida correctiva ya que el proyecto no se ha desviado de la ruta marcada en la planificación con lo que puede considerarse un éxito en base a la gestión del proyecto.

12 Lecciones aprendidas

Utilizar nuevas tecnologías emergentes y actuales como el uso de un software tipo BaaS, herramientas de test de usuarios o el propio desarrollo en Android que evoluciona constantemente: apis de android-support, fragments, Google Maps,… han supuesto un reto y una fuente de aprendizaje útil para futuros proyectos.

Trabajar con la metodología de trabajo de diseño centrado en el usuario me ha permitido reflexionar y valorar positivamente este tipo de técnicas para incluirlas en futuros proyectos con la intención de aportar la mejor experiencia de usuario posible.

La experiencia de aplicar el modelo de desarrollo ágil ha permitido facilitar la tarea de recibir los cambios o ajustes (como en este caso cuando se ha presentado el prototipo a los usuarios) ya que al priorizar la entrega del producto se ha permitido agilizar la entrega de nuevos avances de la aplicación.

13 Posibles ampliaciones

Durante el transcurso de este trabajo han ido surgiendo nuevas ideas y posibles mejoras, algunas de las cuales han sido incluidas en la aplicación (como ya se ha comentado anteriormente gracias al método DCU y al resultado obtenido mediante el test de usuarios). Sin embargo, para hacer la aplicación más atractiva y comercial se deberían añadir más funcionalidades que aporten un valor añadido a la aplicación y que marque una diferencia con respecto a otras aplicaciones que pueda haber similares en el mercado.

Como resulta evidente la aplicación no está finalizada y posiblemente contenga bugs que se deberían corregir. Por ello, lo primero que se tendría que hacer es complementar y depurar los servicios básicos necesarios para poder publicar una primera versión.

Estas ampliaciones básicas podrían ser:

- Permitir la modificación de promociones, eventos y de los datos identificativos de los artistas y grupos musicales ya que este proyecto se han omitido para reducir la carga de trabajo y cumplir con el planning propuesto.
- Ampliar los datos identificativos de los artistas y grupos musicales: discografía, videos, datos de contacto, componentes, bibliografía, …
- Permitir seleccionar distintos tipos de opciones de distancia en la funcionalidad "*Cerca de mi…*". Ejemplo: 1, 5, 10, 25, 50 Km.
- Permitir seleccionar distintos días de celebración de eventos en la funcionalidad "*Cerca de mi…*". Ejemplo: hoy, mañana, de hoy a mañana, de hoy a tres días, de hoy a una semana, de hoy a dos semanas,…
- Permitir comentarios a las promociones de los artistas y grupos musicales
- Permitir valuar las promociones y comentarios con un "me gusta"
- Permitir una lista de favoritos para tener un fácil acceso a los artistas y grupos musicales favoritos.
- Manejo de varios idiomas. Dar soporte a varios idiomas incluyendo todo lo relacionado al idioma, medidas, cantidades,...
- Poner un botón con las normas de uso y citar las leyes vigentes a las que está sujeta la aplicación en cuanto a la ley de protección de datos y al uso al que va a hacer con ellos,…
- Poder hacer login con otras cuentas como facebook, twiter, google,…

Naturalmente una vez implementadas estas ampliaciones habría que publicarla en Google Play.

Además de las ampliaciones básicas se le podría agregar otros servicios como:

- Oferta y demanda de empleo en el sector espectáculos: músicos (guitarristas, batería, bajo, guitarrista, cantantes,...), arreglistas, transportistas para trasladar el material y/o al grupo…
- Mercadeo musical. Compra y venta de instrumentos musicales y otros artículos.
- Merchandising. Venta de material promoción de los artistas y grupos musicales.
- Ventas de entradas.
- Viaje en grupo. Sería un servicio en el cual un grupo de usuarios pueden ir conjuntamente a un evento desde diferentes lugares y pueden ser monitorizados en un mapa indicando en qué lugar y a que distancia se encuentran del evento y de los restantes usuarios que forman el grupo. Para ello los usuarios que forman el grupo deben permitir conocer su ubicación geográfica en cada momento.
- Servicios de publicidad. Una forma más de monetizar la aplicación, ya que en los demás servicios se podrían hacer servicios especiales que por un coste específico puede dársele la opción al usuario de salir en las primeras posiciones de otros servicios como: oferta y demanda, mercadeo musical, merchandising, promociones,…

Por otro lado se podría contemplar usar otros servicios BaaS con la intención de ver las características técnicas y rendimiento que ofrecen y los costes económicos que puede acarrear si el servicio alcanzara un volumen de tráfico considerable.

La lista puede ser muy larga pero lo mejor en este caso sería completar las opciones básicas y poco a poco añadir nuevas funcionalidades que aporten valor añadido y que a los usuarios le resulten de ayuda.

También hay que tener en cuenta que implementar algunos de estos servicios necesitaría de una administración y gestión especializada con el propósito de dar soporte a la plataforma y un buen servicio comercial, logístico y administrativo.

Como reflexión final se puede apreciar que si se deja fluir la imaginación al desarrollador la aplicación puede llegar a ser muy amplia, pero quizás corra el peligro de no ser de interés para los usuarios a los cuales se quiere llegar.

No hay que perder de la vista de que el diseño es centrado en el usuario, no en el desarollador y las definiciones de las necesidades deben de salir de los usuarios, ya que ellos son los que al final solicitarán y disfrutarán de la experiencia de usuario que ellos mismos nos descubren. De nada sirve hacer investigación de usuarios si en el proceso esto se diluye.

Por eso, jamás se debería perder la visión de que se está satisfaciendo las necesidades de una persona, y se debería tener muy presente en todo momento, ya que esa persona será la que use, reclame y promocione nuestro producto.

ANEXOS

14 Anexo I: Perfiles de usuario y resultados

Perfiles de usuario y resultado de las pruebas.

Usuario 1	
Nombre	Antonio Blanco
Edad	41 años
Profesión	Técnico, mantenimiento de sistemas
Características	Dispone de Smartphone y de una Tablet. Tiene experiencia con estos dispositivos y conoce las aplicaciones de las pruebas. Tiene un grupo musical que lleva dos años formado y muestra interés por la aplicación.
Resultado de las pruebas	
Prueba 1	Éxito, sin problemas aparentes
Prueba 2	Éxito, sin problemas aparentes
Prueba 3	Éxito, sin problemas aparentes
Conclusiones	Conoce las aplicaciones y no tiene problemas en el uso. En la entrevista se ha mostrado muy interesado en las posibles funcionalidades.

Usuario 2	
Nombre	María Torres
Edad	27 años
Profesión	Administrativa
Características	Dispone de Smartphone y de una Tablet. Tiene experiencia con estos dispositivos y conoce las aplicaciones de las pruebas.
Resultado de las pruebas	
Prueba 1	Éxito, sin problemas aparentes
Prueba 2	Éxito, sin problemas aparentes
Prueba 3	Éxito, sin problemas aparentes
Conclusiones	Conoce las aplicaciones y no tiene problemas en el uso. En la entrevista ha sacado el tema de si podría ser posible puntuar o dar un me gusta a los grupos que le fueran de su agrado.

Usuario 3	
Nombre	Elena Bergara
Edad	24 años
Profesión	Estudiante de medicina
Características	Dispone de Smartphone. Conoce Google Maps pero no FourSquare. Sin embargo, tiene experiencia en otras aplicaciones de características similares
Resultado de las pruebas	
Prueba 1	Éxito, sin problemas aparentes
Prueba 2	Éxito, sin problemas aparentes
Prueba 3	Éxito, pero le ha llevado un poco de tiempo realizarla
Conclusiones	Conoce las aplicaciones y no tiene problemas en el uso

Usuario 4	
Nombre	Juan Hernández
Edad	32 años
Profesión	Músico profesional

Características	Tiene 10 años de experiencia en el mundo de los espectáculos. Domina la tecnología y tiene dispositivos móviles.
Resultado de las pruebas	
Prueba 1	Éxito, sin problemas aparentes
Prueba 2	Éxito, sin problemas aparentes
Prueba 3	Éxito, sin problemas aparentes
Conclusiones	Conoce las aplicaciones y no tiene problemas en el uso. En la entrevista ha puesto mucho énfasis en las funcionalidades de la aplicación ya que a él le interesa personalmente, por su profesión, el aplicativo a desarrollar.

Usuario 5	
Nombre	Sergio Pérez
Edad	19 años
Profesión	Estudiante
Características	Le gusta ir a fiestas y eventos dónde se organiza música en vivo. Tiene un Smartphone y una Tablet y domina con soltura Google Maps.
Resultado de las pruebas	
Prueba 1	Éxito, sin problemas aparentes
Prueba 2	Éxito, sin problemas aparentes
Prueba 3	Éxito, sin problemas aparentes
Conclusiones	Conoce las aplicaciones y no tiene problemas en el uso. En la entrevista ha mostrado interés por buscar eventos dónde algún tipo de grupo participara cerca de su ubicación

Usuario 6	
Nombre	Fernando Souto
Edad	55 años
Profesión	Arquitecto
Características	Tiene un Smartphone. Conoce Google Maps pero no lo usa muy a menudo. No conoce FourSquare.
Resultado de las pruebas	
Prueba 1	Éxito, pero con demasiado tiempo en el desarrollo
Prueba 2	Éxito, pero con demasiado tiempo en el desarrollo
Prueba 3	Éxito, pero con demasiado tiempo en el desarrollo
Conclusiones	Conoce Google Maps pero no lo usa muy a menudo. No conoce FourSquare. En algunas de las pruebas ha tenido complicaciones. En la entrevista mostró interés por las funcionalidades de búsqueda de eventos pero las demás funciones no mostraban interés.

15 Anexo II: Código de pruebas unitarias y funcionales

Prueba unitaria LoginActivityUnitTest.java

```
package com.redsocial.android.test;

import android.content.Intent;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.redsocial.android.LoginActivity;


public class LoginActivityUnitTest extends
    android.test.ActivityUnitTestCase<LoginActivity> {

  private int buttonId;
  private LoginActivity activity;

  public LoginActivityUnitTest() {
    super(LoginActivity.class);
  }

  @Override
  protected void setUp() throws Exception {
    super.setUp();

    Intent intent = new Intent(getInstrumentation().getTargetContext(),
                    LoginActivity.class);
    startActivity(intent, null, null);
    activity = getActivity();
  }

 public void testPreconditions() {
                    assertNotNull(buttonId);
          }

  public void testLayout() {
    buttonId = com.redsocial.android.R.id.action_button;
    assertNotNull(activity.findViewById(buttonId));
    Button view = (Button) activity.findViewById(buttonId);
    assertEquals("Incorrecta la etiqueta del botón", "Log in", view.getText());

  }


  public void testIntentTriggerViaOnClick() {
    buttonId = com.redsocial.android.R.id.action_no_tienes_cuenta;
    TextView view = (TextView) activity.findViewById(buttonId);
    EditText username = (EditText) activity.findViewById(com.redsocial.android.R.id.username);
    username.setText("usuario1");
    assertNotNull("El boton no puede ser nulo", view);

    view.performClick();



    // Comprueba el intent iniciado
    Intent triggeredIntent = getStartedActivityIntent();
    assertNotNull("Intent es nulo", triggeredIntent);
    String data = triggeredIntent.getExtras().getString("username");

    assertEquals("Datos incorrectos pasados via intent",
        "usuario1", data);
  }

}
```

Prueba funcional AddPromocionActivityTest.java

```
package com.redsocial.android.test;

import android.test.ActivityInstrumentationTestCase2;
import android.test.TouchUtils;
import android.test.suitebuilder.annotation.SmallTest;
import android.widget.EditText;

import com.redsocial.android.AddPromocionActivity;

public class AddPromocionActivityTest extends ActivityInstrumentationTestCase2<AddPromocionActivity> {

            private AddPromocionActivity mActivity;
            private EditText mView;
            private String resourceString;
```

```
	public AddPromocionActivityTest(String name) {
		super("com.redsocial.android", AddPromocionActivity.class);
		// TODO Auto-generated constructor stub
		setName(name);
	}


	protected void setUp() throws Exception {
		super.setUp();
		setActivityInitialTouchMode(true);
		mActivity = getActivity();
		mView = (EditText)	mActivity.findViewById(com.redsocial.android.R.id.et_Descripcion);

}

	@SmallTest
	public void testPreconditions() {
		assertNotNull(mView);
	}

	@SmallTest
	  public void testStartingEmpty() {
	    assertTrue("field is empty", "".equals(mView.getText().toString()));
	  }


	@SmallTest
	public void testText() {
		mView.clearComposingText();

		mView.requestFocus();
		TouchUtils.tapView(this, mView);
		sendKeys("1");

		resourceString = mView.getText().toString();
		assertEquals("1",resourceString);
	}

	protected void tearDown() throws Exception {
		super.tearDown();
	}

}
```

16 Anexo III: Código completo de Mapa.java

```
package com.redsocial.android;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Hashtable;
import java.util.List;
import java.util.Map;

import android.app.DatePickerDialog;
import android.app.ProgressDialog;
import android.app.DatePickerDialog.OnDateSetListener;
import android.app.Dialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.NavUtils;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.GoogleMap.OnCameraChangeListener;
import com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListener;
import com.google.android.gms.maps.GoogleMap.OnMarkerClickListener;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.parse.ParseException;
import com.parse.ParseGeoPoint;
import com.parse.ParseObject;
import com.parse.ParseQuery;
import com.redsocial.android.dataobjects.Evento;


/*
 * Situa los eventos en un mapa
 */
public class Mapa extends ActionBarActivity  implements OnInfoWindowClickListener,

                    OnMarkerClickListener,

                    OnCameraChangeListener,

                    OnTouchListener{

    private OnDateSetListener mDateSetListener;
    private GoogleMap mapa = null;
           private int vista = 0;
    private Marker marker;
    private Map<Marker,Evento> listMarker;
           private List<Evento> datosList;
           private Double Latitud;
    private Double Longitud;
    private EditText et_fecha;
    private List<ParseObject> ob;
    private Calendar calendar;
           private boolean moverCamara=true;

            @Override
               protected void onCreate(Bundle savedInstanceState) {
               super.onCreate(savedInstanceState);
               setContentView(R.layout.mapa);

                      ActionBar bar = getSupportActionBar();
                      bar.setDisplayHomeAsUpEnabled(true);

               et_fecha = (EditText)findViewById(R.id.et_Fecha);

               calendar = Calendar.getInstance();
                      calendar.setTime(new Date());
                      calendar.set(Calendar.HOUR_OF_DAY,0);
                      calendar.set(Calendar.MINUTE,0);
                      calendar.set(Calendar.SECOND,0);
                      calendar.set(Calendar.MILLISECOND,0);

       String s = (new SimpleDateFormat("dd/M/yyyy")).format(calendar.getTime());
       et_fecha.setText(s);
```

```
        et_fecha.setOnTouchListener(this);

        mDateSetListener = new OnDateSetListener(){
           @Override
           public void onDateSet(DatePicker view, int year, int monthOfYear,
                                 int dayOfMonth) {

                      calendar.set(year,monthOfYear,dayOfMonth);
                calendar.set(Calendar.HOUR_OF_DAY,0);
                      calendar.set(Calendar.MINUTE,0);
                      calendar.set(Calendar.SECOND,0);
                      calendar.set(Calendar.MILLISECOND,0);

                String s = (new SimpleDateFormat("dd/M/yyyy")).format(calendar.getTime());
                et_fecha.setText(s);

           }

        };


               mapa =((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
               mapa.setOnInfoWindowClickListener(this);

               actualizarDatos();


               }


               /*
                * Tarea conseguir datos de Parse
                */
               public void actualizarDatos(){
                      new RemoteDataTask().execute();
               }




               /*
                * Tarea remota asíncrona para obtener datos de parse
                */
                      private class RemoteDataTask extends AsyncTask<Void, Void, Void> {

                                 // Crea un dialogo de progreso
                           final ProgressDialog dialog = new ProgressDialog(Mapa.this);

                                 @Override
                                 protected void onPreExecute() {
                                     super.onPreExecute();

                                     dialog.setMessage(getString(R.string.progress_mapa));
                                     dialog.show();

                                     mapa.clear();

                                     }

                                     @Override
                                     protected Void doInBackground(Void... params) {

                                     // Create el array del conjunto eventos
                                     datosList=null;
                                     datosList = new ArrayList<Evento>();

                                     // Crea un map de marcadores con su evento
                                     listMarker=null;
                                     listMarker= new Hashtable<Marker,Evento>();


                                     try {
                                         // Localiza la clase llamada "Eventos" en Parse.com
                                         ParseQuery<ParseObject> query = new ParseQuery<ParseObject>("Evento");
                                         query.setLimit(100);
                                         query.orderByAscending("fecha");
                                         query.whereGreaterThanOrEqualTo("fecha",calendar.getTime());

                                         Calendar calendar2 = Calendar.getInstance();
                                         calendar2.setTimeInMillis(calendar.getTimeInMillis());
                                         calendar2.add(Calendar.DATE,1);

                                         Log.d("fecha desde:", calendar.getTime().toString());
                                         Log.d("fecha hasta:", calendar2.getTime().toString());
                                         query.whereLessThan("fecha", calendar2.getTime());
                                         query.include("formacion");

                                         if (ob!=null) ob.clear();
                                         ob= query.find();

                                         for (ParseObject evento : ob) {

                                             ParseObject formacion = evento.getParseObject("formacion");
                                             ParseGeoPoint location = new ParseGeoPoint();
                                             location= evento.getParseGeoPoint("localizacion");
```

```
                                // Mapea los datos de parse con el dataobject Evento
                                Evento map = new Evento();
                                map.setIdEvento((String) evento.getObjectId());
                                map.setNombre((String) formacion.get("nombreFormacion"));
                                map.setTipo((String) formacion.get("tipo"));
                                map.setLugar((String) evento.get("lugar"));
                                map.setProvincia((String) evento.get("provincia"));
                                map.setFecha( ((Date) evento.get("fecha")));
                                map.setDescripcion((String) evento.get("descripcion"));
                                map.setLatitud(Double.valueOf(location.getLatitude()));
                                map.setLongitud(Double.valueOf(location.getLongitude()));
                                datosList.add(map);
                            }
                        } catch (ParseException e) {
                            Log.e("Error", e.getMessage());
                                e.printStackTrace();
                            }
                            return null;
                        }

                        @Override
                        protected void onPostExecute(Void result) {
                           dialog.dismiss();
                           addMarcadores();

                       }
                   }


        @Override
                   public boolean onCreateOptionsMenu(Menu menu) {
                              getMenuInflater().inflate(R.menu.menu_mapa, menu);
                              return true;
                   }


                   @Override
                   public boolean onOptionsItemSelected(MenuItem item)
                   {
                              switch(item.getItemId())
                              {
                                        // Respond to the action bar's Up/Home button
                                  case android.R.id.home:
                                      NavUtils.navigateUpFromSameTask(this);
                                      break;
                                  case R.id.menu_vista:
                                                    alternarVista();
                                                    break;
                              }

                              return super.onOptionsItemSelected(item);
                   }


                   private void alternarVista()
                   {
                              vista = (vista + 1) % 4;


                              switch(vista)
                              {
                                        case 0:
                                                    mapa.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                                                    break;
                                        case 1:
                                                    mapa.setMapType(GoogleMap.MAP_TYPE_HYBRID);
                                                    break;
                                        case 2:
                                                    mapa.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
                                                    break;
                                        case 3:
                                                    mapa.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
                                                    break;
                              }
                   }

        private void addMarcador(Evento act, double lat, double lng)
        {
                 marker = mapa.addMarker(new MarkerOptions()
                .position(new LatLng(lat, lng))
                .title(act.getNombre())
                .snippet(act.getLugar()+"("+act.getProvincia()+")")
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE))); //.anchor(0.0f, 1f));

                 listMarker.put(marker, act);
        }


        @Override
        public void onInfoWindowClick(Marker marker) {

                   lanzarMapaEvento(marker);

        }
```

```
        @Override
        public boolean onMarkerClick(Marker marker) {
                // TODO Auto-generated method stub
                return false;
        }


        @Override
        public void onCameraChange(CameraPosition position) {
                // TODO Auto-generated method stub

        }


         public void lanzarMapaEvento(Marker marker)
            {
                Evento evento = (Evento)listMarker.get(marker);
                Intent intent = new Intent(getApplicationContext(), EventoInfo.class);
                intent.putExtra("evento", evento);
                startActivity(intent);

            }

        public void addMarcadores() {

             for (Evento act: datosList) {
                  try{
                              addMarcador(act,Double.valueOf(act.getLatitud()),
                                              Double.valueOf(act.getLongitud()));

                  }catch(NullPointerException nullpointerexception){}
             }

                  if (moverCamara){
                            CameraUpdate cameraupdate = setMovePoistion();
                            mapa.animateCamera(cameraupdate);
                  }

                  if (datosList.isEmpty())
                        Toast.makeText(this,"No se han obtenido datos para esta petición",
                        Toast.LENGTH_SHORT).show();
        }


        private CameraUpdate setMovePoistion() {


                  Longitud = (double) this.getSharedPreferences("Preferencias", 0)
                                .getFloat("longitud",0);
            Latitud = (double) this.getSharedPreferences("Preferencias", 0)
                            .getFloat("latitud",0);

                  LatLng latlng = new LatLng(Latitud, Longitud);
                  CameraUpdate cameraupdate = CameraUpdateFactory.newCameraPosition((new CameraPosition.Builder())
                                        .target(latlng)
                                        .zoom(7F)
                                        .bearing(0.0F)
                                        .tilt(60F)
                                        .build());
                  return cameraupdate;
        }

         protected Dialog onCreateDialog(int i)
            {
                Calendar calendar = Calendar.getInstance();
                int year = calendar.get(Calendar.YEAR);
                int mes = calendar.get(Calendar.MONTH);
                int diaMes = calendar.get(Calendar.DAY_OF_MONTH);
                switch(i)
                {
                default:
                    return null;

                case 0:
                    return new DatePickerDialog(this, mDateSetListener, year, mes, diaMes);
                }
            }

         public void lanzarBuscar(View view)
            {
                    moverCamara = false;
                actualizarDatos();
            }

        @Override
        public boolean onTouch(View v, MotionEvent event) {
                  if(v == et_fecha)
           showDialog(0);
       return false;

        }
}
```

mapa.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >



<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#AA000000"
    android:gravity="center_vertical"
    android:orientation="horizontal"
    android:paddingBottom="5.0dip"
    android:paddingLeft="10.0dip"
    android:paddingRight="10.0dip"
    android:paddingTop="5.0dip" >



    <EditText
        android:id="@+id/et_Fecha"
        android:layout_width="0.0dip"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20.0dip"
        android:layout_weight="3.0"
        android:gravity="center_horizontal"
        android:ems="10"
        android:hint="DD/MM/YYYY"
        android:inputType="text"
        android:textColor="#FFFFFFFF"
        android:maxLength="10"
        android:singleLine="true" />

    <Button
        android:layout_width="0.0dip"
        android:layout_height="wrap_content"
        android:layout_weight="2.0"
        android:onClick="lanzarBuscar"
        android:textColor="#FFFFFFFF"
        android:text="@string/buscarMayus"
        android:textSize="14.0sp"
        android:textStyle="bold" />

</LinearLayout>

 <fragment
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

BIOGRAFÍA Y REFERENCIAS

Material UOC

- Diseño centrado en el usuario
- Redacción de textos científico-técnicos
- Exposición de contenidos en vídeo
- Tecnología y desarrollo en dispositivos móviles
- Presentación de documentos y elaboración de presentacions
- Trabajo final de màster
    - *Módulo 1. Introducción al trabajo final*
    - *Módulo 2. El trabajo final como proyecto*
    - *Módulo 3. La gestión del proyecto a lo largo del trabajo final*

Android Design
http://developer.android.com/design/index.html

Prototyper justinmind
http://www.justinmind.com

Android developers
http://developer.android.com/develop/index.html

Support libraries
http://developer.android.com/tools/support-library/index.html

Joda Time
http://www.joda.org/joda-time/

Google Maps Android API v2
https://developers.google.com/maps/documentation/android/start?hl=es

Parse
https://www.parse.com

Android Developer Guide | Parse
https://parse.com/docs/android_guide

Stackoverflow
http://stackoverflow.com

Android}egin.com
http://www.androidbegin.com

Android JUnit Test
http://www.bogotobogo.com/Android/android11JUnitTest.php