Departamento de Computación
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO DE FIN DE GRADO
GRADO EN ENXEÑERÍA INFORMÁTICA
Mención en Sistemas de Información

Aplicación móvil para actividades a campo abierto

Alumno: Alberto Ramil Fernández
Directores: Javier Parapar López
Óscar Pedreira Fernández

A Coruña, 10 de enero de 2018

Datos del trabajo

Título del Trabajo:	Aplicación móvil para actividades a campo abierto
Clase a la que pertenece:	Trabajo Clásico de Ingeniería
Alumno:	Alberto Ramil Fernández
Directores:	Javier Parapar López Óscar Pedreira Fernández

Miembros del tribunal:

Fecha de lectura y defensa: A Coruña, 10 de enero de 2018

Calificación:

JAVIER PARAPAR LÓPEZ
Profesor de Universidad
Departamento de Computación
Universidade da Coruña

y

ÓSCAR PEDREIRA FERNÁNDEZ
Profesor Predoctoral FPU
Departamento de Computación
Universidade da Coruña

CERTIFICAN: Que la memoria titulada *Aplicación móvil para actividades a campo abierto* ha sido realizada por ALBERTO RAMIL FERNÁNDEZ bajo su dirección y constituye su Traballo de Fin de Grado en el Grado de Enxeñería Informática.

En A Coruña, a 10 de enero de 2018

JAVIER PARAPAR LÓPEZ
Director

ÓSCAR PEDREIRA FERNÁNDEZ
Director

Dedicatoria

Agradecimientos

I would like to thank...

Author name
A Coruña, 10 de enero de 2018

Resumen

This is the summary of the project. No more than one page, please.

Palabras clave

First keyword, second keyword, etc.

Índice general

Capítulo 1

Introducción

Intro

1.1. Motivación

1.2. Objetivos

- Y
- X

1.3. Estructura de la memoria

La memoria del presente proyecto está estructurada del siguiente modo:

- **Introducción** Explica el contexto en el que se enmarca el proyecto, introduce la problemática a tratar y detalla el alcance y objetivos del mismo desde un punto de vista global. También muestra la estructura de la memoria y el plan de trabajo seguido.
- **Tecnología** Describe y justifica las principales tecnologías empleadas para desarrollar el objeto del proyecto atendiendo a los requisitos del mismo.
- **Conceptos**

- **Proceso de ingeniería** Detalla el proceso de ingeniería: la metodología, la planificación y la gestión del proyecto.
- **Desarrollo** Realiza una descripción detallada del análisis, diseño, implementación, pruebas y despliegue del sistema.
- **Conclusiones y trabajo futuro** Proporciona una evaluación global del producto obtenido así como futuras líneas de trabajo que se podrían explotar alrededor del proyecto.
- **Apéndices** Está compuesto por las siguientes secciones complementarias:
  - **Glosario** Define los términos y acrónimos técnicos empleados en la memoria del proyecto.
  - **Bibliografía** Recoge los documentación bibliográfica sobre la que se apoya el proyecto.

1.4. Plan de trabajo

Capítulo 2

Conceptos

Capítulo 3

Tecnología

La selección de la tecnología adecuada es una condición necesaria, aunque no suficiente, para llevar a cabo un proyecto con éxito. En este capítulo se detallarán las elecciones tecnológicas y se justificarán debidamente.

Este capítulo contiene una sección para explicar las tecnologías de cada uno de los componentes. Adicionalmente, se incluye un apartado donde se describen las herramientas utilizadas para dar soporte a la gestión y desarrollo del proyecto.

Con estas tecnologías se ha llevado a cabo la construcción de los distintos componentes del sistema siguiendo los pasos que comentaremos en el próximo capítulo.

Capítulo 4

Proceso de ingeniería

En este capítulo se justifica y se describe la metodología de desarrollo sobre la que se apoya el proceso de ingeniería. En el caso del presente proyecto se ha optado por emplear una aproximación ágil al proceso unificado de desarrollo. Se estudiarán también el resto de aspectos relacionados con el proceso de ingeniería como son el ciclo de vida, la descomposición en tareas, la organización, la estimación de costes y duración, la gestión de proyecto, la planificación y la gestión de riesgos.

Capítulo 5

Analisis

En este capítulo se detalla el proceso de desarrollo del proyecto. En primer lugar, se analizarán los requisitos del proyecto para definir la arquitectura general del sistema. Luego se describirán el análisis, el diseño y la complementación de los componentes. Por último, se mostrarán datos ofrecidos por las herramientas de soporte al desarrollo.

desarrollo.

5.1. Análisis de Requisitos

En esta aplicación móvil para actividades a campo abierto objetivo de este proyecto, se establecieron una seria de requisitos generales que debería cumplir la aplicación.

El registro del usuario para poder comenzar a usarla.

La creación de puntos de interés marcados en un mapa con nombre, descripción y un punto en el mapa con o sin señal GPS clasificados en tipo, caza o pesca.

Se podrán guardar las rutas seguidas por un usuario en sus caminatas por cualquier tipo de terrero.

El usuario también podrá crear grupos con los usuarios que quiera y los integrantes del mismo poder añadir a otros, el resultado de esta funcionalidad es la que permitirá posteriormente crear rutas conjuntas. Para crear una ruta conjunta primero se

elige el grupo del que se hará el seguimiento y se enviarán las invitaciones para participar en él a cada integrante del grupo. Estas invitaciones en el caso de ser aceptadas llevaran al usuario a un mapa y periódicamente se irán realizando actualizaciones de las posiciones del resto de integrantes del grupo anteriormente indicado. Finalmente se podrán ver las rutas conjuntas igual que las individuales.

5.1.1. Actores

Los únicos actores que se presentan en la aplicación son los siguientes:

- **Usuario no autenticado**.Usuario que no está autenticado en la aplicación y que se le permite registrarse en el sistema o iniciar sesión si la ya se registro en otro momento.

- **Usuario autenticado**.Usuario autenticado que puede acceder a todas as funcionalidades del sistema.

5.1.2. Casos de uso

A continuación, en esta sección, se exponen los requisitos funcionales que surgen de los requisitos generales planteados en el punto anterior.

• **Usuario no autenticado**

- ***R1* Registrarse en la aplicación.** El usuario podrá darse de alta en el sistema introduciendo sus datos en el formulario que se le indican. Una vez registrado se iniciará sesión automáticamente con el nuevo perfil.

- ***R2* Iniciar sesión en la aplicación.** El usuario ya registrado podrá, con sus credenciales, autenticarse en el sistema. Se pedirá o nombre del usuario la aplicación y su contraseña. Se guardará el estado en el terminal hasta que el usuario decida desconectarse.

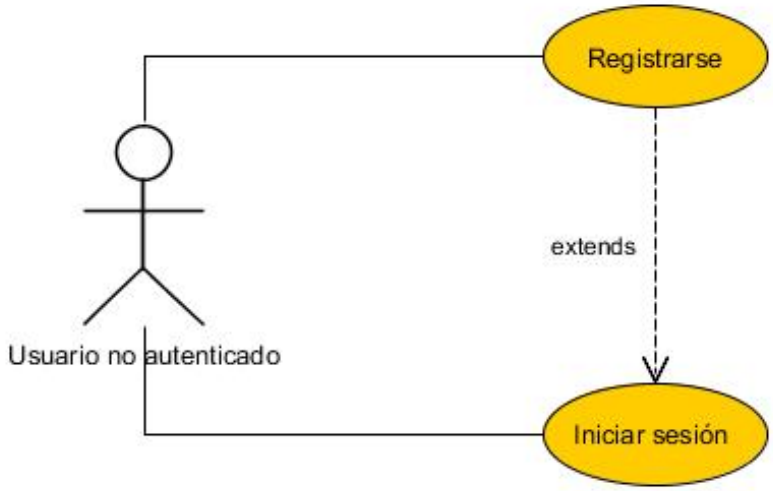

Figura 5.1: Casos de uso del actor Usuario No Autenticado

- **Usuario autenticado**

Para hacer un poco más comprensible dividiré los casos de uso del usuario autenticado por grupos funcionales.

- **Gestión de puntos de interés**
  Aquí se describirán los casos de uso relacionados con la gestión de la información del los puntos de interés

  - ***R-PDI-1 Guardar Punto De Interés caza***, el usuario podrá guardar un punto concreto, de caza, asociado a un par de coordenadas pudiendo añadirle un nombre y una descripción.
  - ***R-PDI-2 Guardar Punto De Interés pesca***, el caso de uno es similar al de anterior pero este es para el tipo de pesca.
  - ***R-PDI-3 Eliminar PDI***, el usuario podrá seleccionar un punto o una lista de puntos para ser borrados.
  - ***R-PDI-4 Buscar los PDI***, permite ver todos los puntos de interés de cada tipo por separado pintados en un mapa y pudiendo ciclar en ellos para conocer su nombre y descripción.

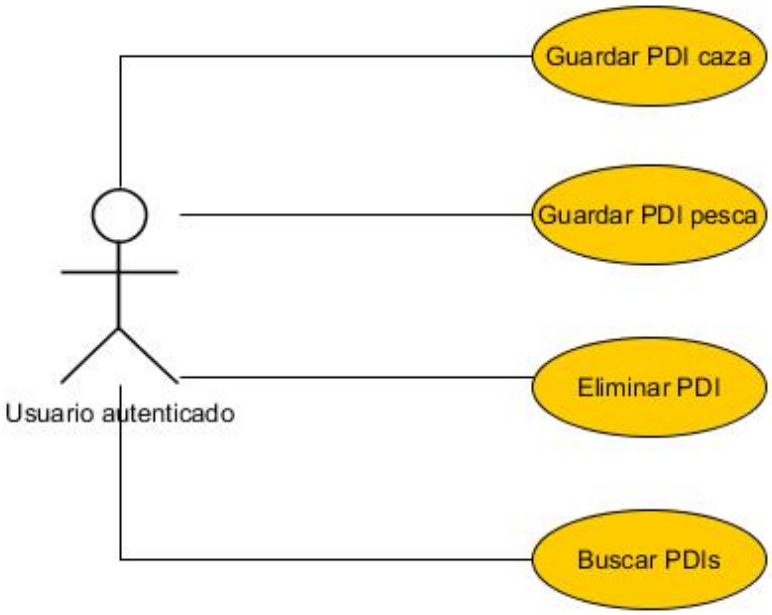

Figura 5.2: Casos de uso de gestión de puntos de interés

- **Gestión de grupos**

  - ***R-G-1 Crear grupo***, el usuario crea un grupo con nombre único.
  - ***R-G-2 Añadir integrantes***, el usuario busca en la base de datos los usuarios que quiere integrar en el grupo previamente creado.
  - ***R-G-3 Eliminar integrantes***, el usuario puede eliminar los integrantes que vea pertinentes.
  - ***R-G-4 Ver grupos***, el sistema listarla los grupos en los que el usuario está registrado.
  - ***R-G-5 Ver integrantes grupo***, el sistema permitirá ver los integrantes del grupo que el usuario indique , previo listado del caso de uso R-G-4.

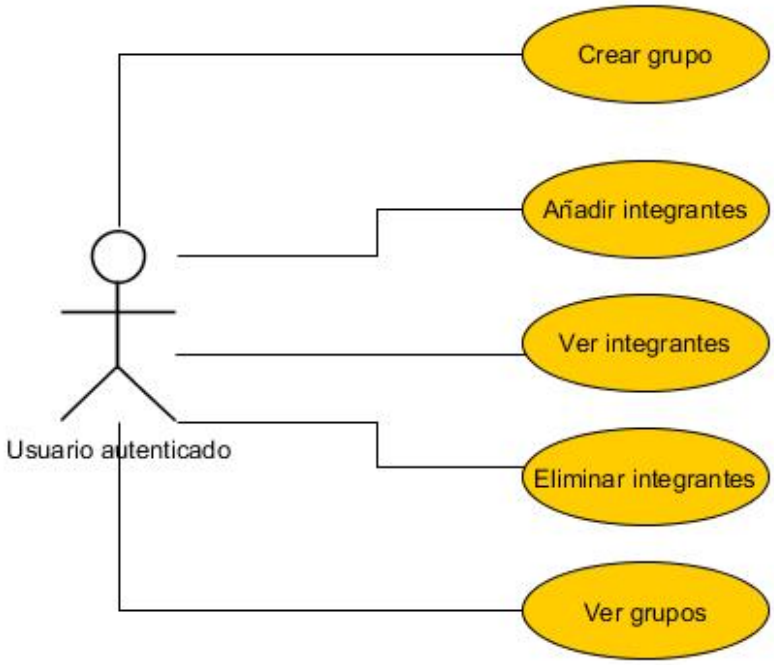

Figura 5.3: Casos de uso de gestión de grupos de usuarios

- **Gestión de rutas,**el usuario iniciará una navegación privada siendo guardada la ruta seguida.
  - R-R-1 Crear ruta privada, el usuario registra la ruta con un nombre.
    - ***R-R-1.1*** Iniciar ruta, el sistema comienza a guardar las coordenadas por la que el usuario esta navegando y dibujando la ruta en el mapa. Las coordenadas se irán guardando periódicamente.
    - ***R-R-1.2*** Parar ruta, permite parar la navegación, tanto de guardar las coordenadas como de pintar la ruta seguida.
    - ***R-R-1.3*** Guardar ruta, el sistema guarda los últimos puntos que quedaban sin actualizar y ejecuta el caso de uso ver ruta en el mapa(R-R-4).
  - ***R-R-2 Crear ruta compartida***, este caso de uso permite guardar la ruta seguida por el usuario y al mismo tiempo ver la posición del resto de integrantes de un grupo, anteriormente seleccionado, en tiempo real. Este caso de uso también enviaría a los integrantes del grupo una invitación a dicha ruta.

- ***R-R-2.1 Iniciar ruta***, se comienza a guardar y dibujar la ruta en el mapa. Por otra parte se comienza el seguimiento del resto de usuario que estén también navegando. Como también una actualización parcial de la ruta seguida en el servidor.
- ***R-R-2.2 Parar ruta***, se para la navegación y se deja de actualizar la posición al resto de usuario de la ruta compartida.
- ***R-R-2.3 Finalizar ruta***, se guardan los puntos que faltan de enviar al servidor y se deja de enviar datos al resto de integrantes.

- ***R-R-3 Listar rutas*** , permite al usuario ver todas las rutas realizadas tanto de manera privada como de manera compartida.
- ***R-R-4 Ver ruta en mapa***, el sistema dibuja en un mapa la ruta seguida y previamente seleccionada.
- ***R-R-5 Eliminar ruta***, permite al usuario borrar de la aplicación la ruta indicada.

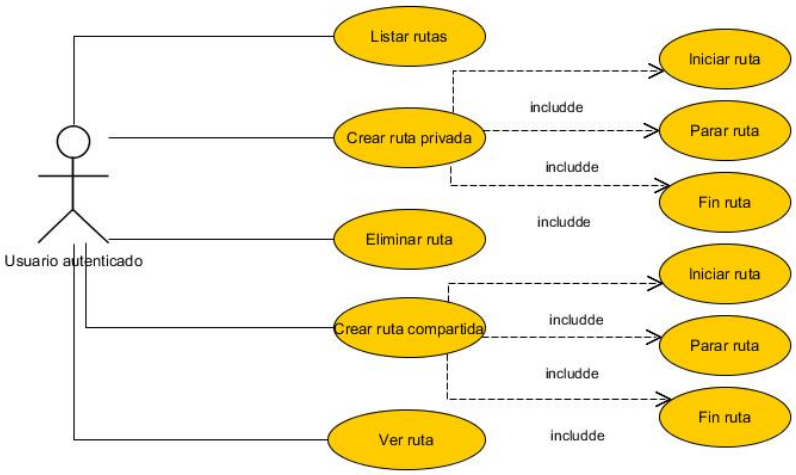

Figura 5.4: Casos de uso de gestión de rutas

5.2. Maquetas

Comenzamos creando las maquetas de como debería que ser a interface del usuario. Se busca que sea una interface simple y intuitiva. Para esto, se intentarán seguir las pautas y usar lo máximo posible los elementos de Material Design [6]. Material Design es un lenguaje de diseñoo para distintas plataformas y dispositivos, creada por el diseñador graco de Google Matas Duarte. Ofrece una guía para ofrecer a los usuarios una experiencia común y habitual entre distintas aplicaciones.

Figura 5.5: Inisiar sesión

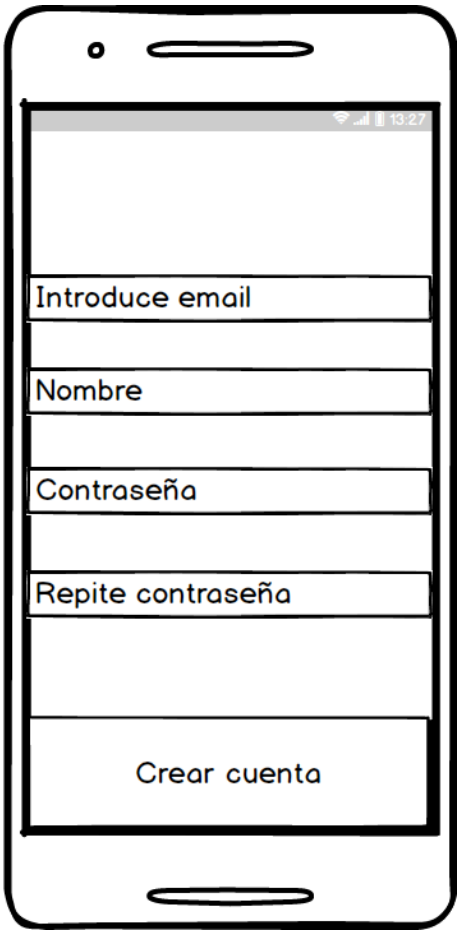

Figura 5.6: Registrar usuario

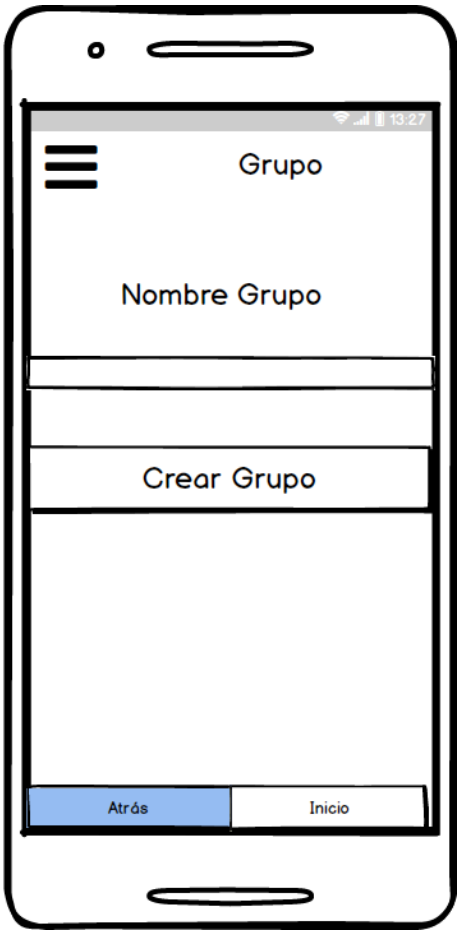

Figura 5.7: Crear grupo

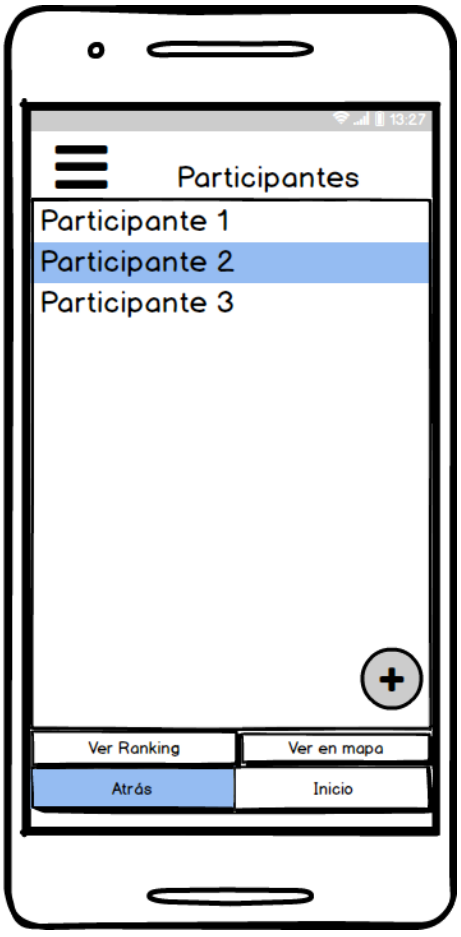

Figura 5.8: Añadir usuarios a grupo

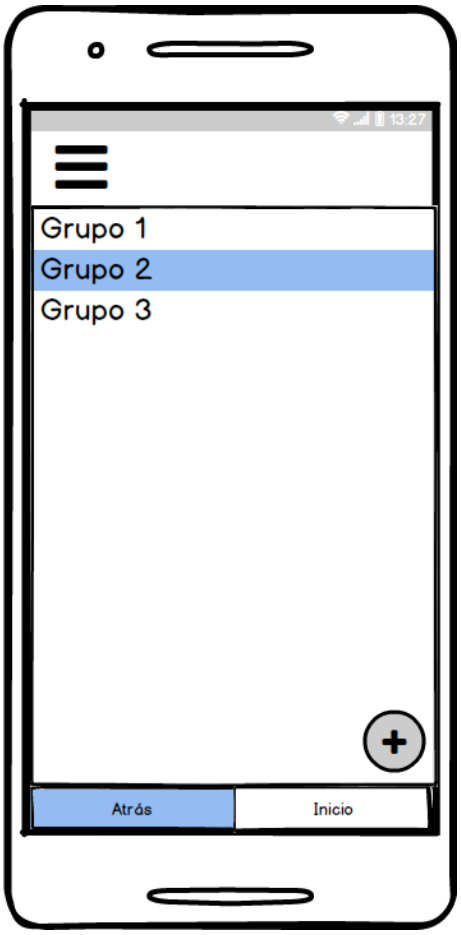

Figura 5.9: Listar grupos

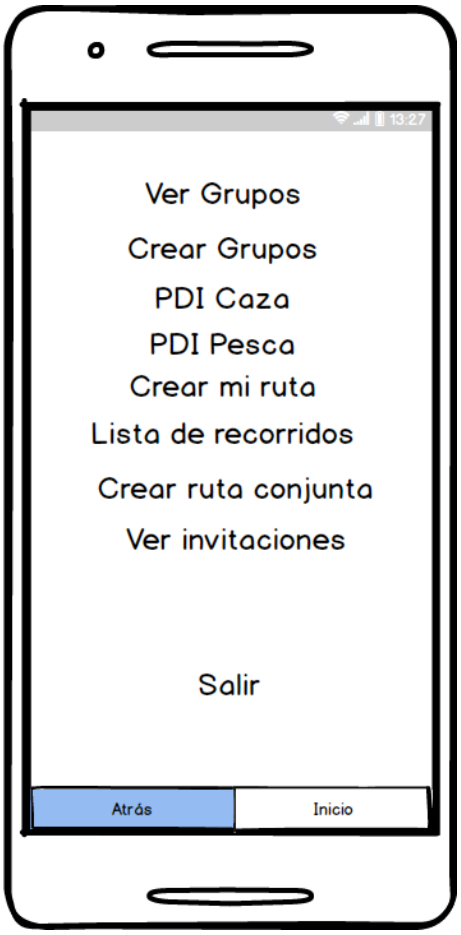

Figura 5.10: Opciones generales

Figura 5.11: Crear PDI

Figura 5.12: Visualizar PDIs

Figura 5.13: Ruta individual

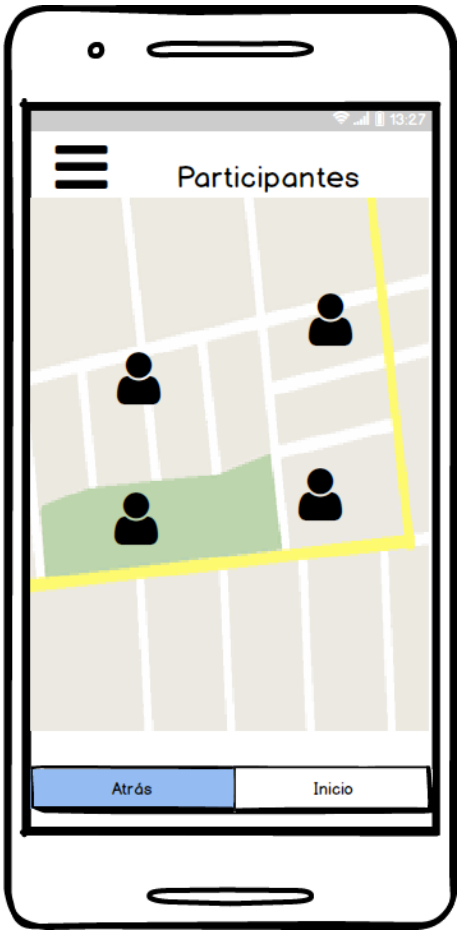

Figura 5.14: Ruta compartida

Capítulo 6

Diseño

En este capítulo se comentará el diseño de la aplicación de manera superficial, la arquitectura general, el modelo de datos y también aspectos más concretos del diseño del servidor y da aplicación móvil.

6.1. Esquema general de la arquitectura

La aplicación que estamos desarrollando seguirá la arquitectura en tres capas. Este patrón arquitectónico cliente/servidor se diferencian tres capas, una capa de interface de usuario , una capa de persistencia y una capa intermedia llamada de servicios que permite la llamada de forma remota a la capa modelo(capa que contiene la lógica) por parte del cliente. Este esquema esta compuesto por:

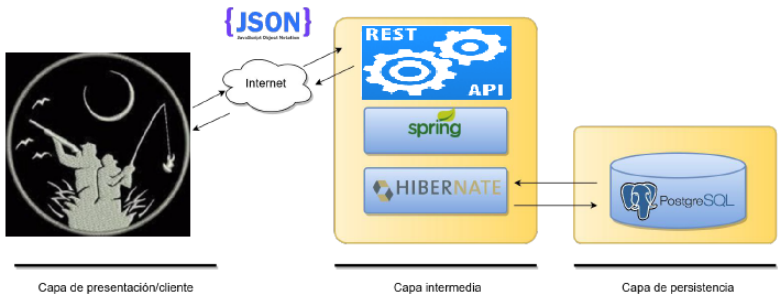

Figura 6.1: Esquema general de la arquitectura del sistema

- **Capa de presentación/cliente**:
  Capa que nos proporciona la información relacionada con los servicios que puede invocar el cliente. Es la encargada de comunicarse con las otras capas para guardar la información de cada usuario.En si es la capa con la que va a interactuar el usuario cada vez trabaje con esta aplicación.

- **Capa intermedia:**
  Esta capa es la encargada de enlazar la capa de persistencia con la capa de presentación/cliente. Lo que hace es recoger los datos que provienen de la base de datos, necesarios para satisfacer el servicio invocado y enviárselos al cliente.

- **Capa de persistencia:**
  Esta capa es la encargada de almacenar los datos del sistemas en la base de datos, además contiene todos los mecanismos de acceso a datos necesarios para poder hacer persistentes los datos. Esta capa debe ofrecer una interface que ayude a la comunicación con la capa intermedia, de manera que se abstraiga de la tecnología usada en el sistema de almacenamiento y no crea una dependencia con ella. Esta abstracción permitirá hacer cambios o actualizaciones en la tecnología sin afectar a otras capas con las que pudiera interaccionar en un futuro.

Un vez comentada la arquitectura general del sistema, pasaremos a comentar la capa intermedia un poco más a fondo ya que en ciertas ocasiones, la capa intermedia puede estar compuesta de N-capas(Arquitectura en N-capas). Ésta es una de esas ocasiones.

Los servidores que siguen el modelo Modelo-Vista-Controlador consiguen separar los datos de la aplicación, la lógica de negocio y el envío de información por la red.

Ésta separación ayuda al desarrollo de la aplicación tanto a la hora de crear la como a la hora de hacer su mantenimiento ya que marca al desarrollador a colocar el código en una capa concreta.

Los componentes capas que componen al patrón MVC son:

- **Modelo:**
  Está compuesta por clases que tienen acceso a los datos ofreciendo unos métodos que pueden ser usados de manera sencilla por las capas superiores. Éstos métodos son los encargados de acceder a la base de datos y proporcionar los datos persistentes.
- **Vista:**
  Esta formada por el interfaz donde se realizan las llamadas entre el servicio web, peticiones HTTP en las que la información va en formato JSON, y la aplicación cliente.
- **Controlador:**
  Esta capa será la encargada de implementar la lógica del interface llamando a las operaciones que ofrece el modelo y seleccionando la vista asociada a cada petición.

6.2. Modelo de datos

6.3. Servidor

Para que pudiera haber una comunicación entre la aplicación del cliente y la capa de persistencia hemos desarrollado una solución que se desplegará a través de un interface, al que se podrá acceder de manera remota y que permitirá tener acceso a las funcionalidades de la capa persistente.

La solución mencionada anteriormente será una aplicación, en java, que usará Spring ya que facilita la creación de aplicaciones de forma cómoda y rápida.

Durante el desarrollo de este proyecto el servidor ha sido desplegado en un proveedor de servidores virtuales llamado DigitalOcean,ya que ofrece distintos lugares donde poder ubicar lo hemos decidido hacerlo en uno concreto de Alemania para que el ping fuera mas cercano y rápido.

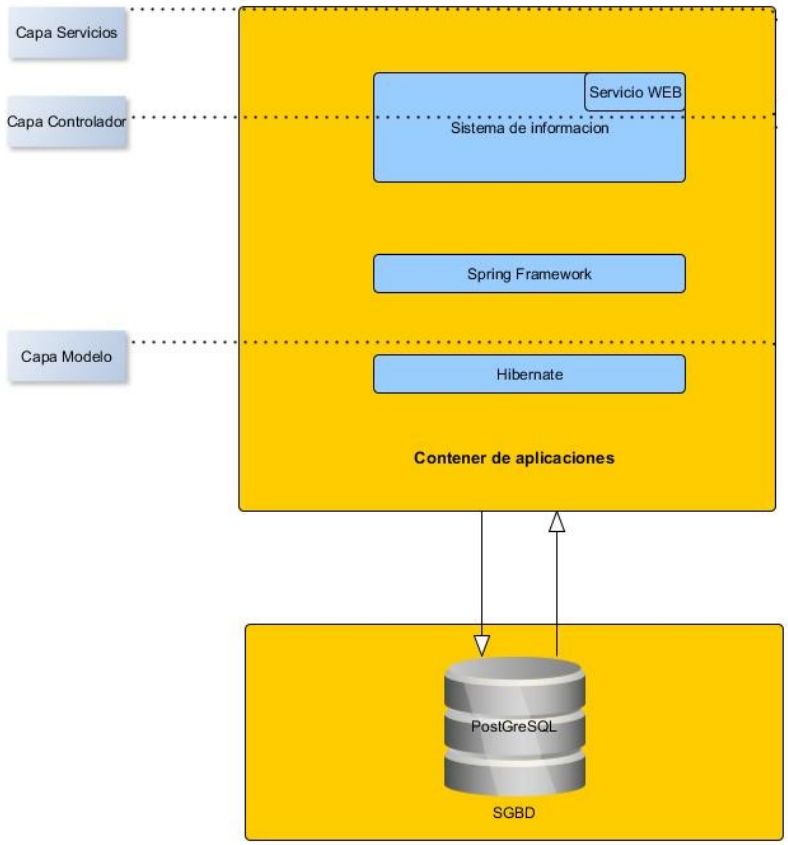

Figura 6.2: Arquitectura del servidor

6.3.1. Servicio web

La comunicación en esta capa se hará mediante una API. Una manera de describir la forma en que los programas o los sitios webs intercambian datos, cuyo formato de intercambio de datos normalmente es JSON. En este servicio hemos decidido usar una API REST que es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP para la obtención de datos. A través de los siguientes peticiones podremos acceder a los distintos recursos.

- Los metodos de acceso indican la accion se queremos realizar sobre los recursos del sistema. Siguiendo esta linea tenemos el GET para solicitar un recurso pedido. POST para crear el recurso en el sistema. Por otro lado para eliminar recursos

usaremos DELETE.

- Dependiendo del método usado usado iremos cambiando la ruta en la petición HTTP. Tanto POST como DELETE tendrán unas rutas con una forma muy similar ya que estos métodos acceden a recursos concretos, ejemplo */usuario/idusuario*. Mientras que GET tanto se puede usar para recursos concretos como para colecciones.

6.3.2. Organizacion dos paquetes

A aplicación del servidor sigue un arquetipo en 5 paquetes distintos de clases Java y uno para los test:

java

- **es.udc.fi.dc.config** Contiene las clases Java de configuración de Spring.
- **es.udc.fi.dc.model** Contiene todos los modelos de datos que estarán almacenados en la base de datos.
- **es.udc.fi.dc.daos** Contiene los interfaces (JpaRepository) que aceden a los datos del sistema.
- **es.udc.fi.dc.controller** Contiene as clases relacionadas con las peticiones remotas.
- **es.udc.fi.dc.services** Contiene tanto los interfaces de los servicios como la implementación de los mismos.

test

es.udc.fi.dc.test Contiene las clases que realizan los test.

6.3.3. Transmisión de la información

Como se mencionó anteriormente para el servicio web usaremos la API REST, esta necesita un lenguaje de intercambio para la transmisión de información. El formato que seguiremos para la transmisión será el JSON y como parseador Jackson. JSON,

siglas JavaScript Object Notation, es un formato estándar de transmisión de la información muy cómodo de usar que emplea pares de clave-valor.

Figura 6.3: JSON

6.3.4. Gestión de las clases persistentes

Para gestionar las clases persistentes de nuestro aplicación usaremos el patrón de diseño basado en DAO(Data Access Object).

Un DAO define un interfaz que contiene conjunto de operaciones de persistencia y a su vez una implementacion permitiendo la gestión de las entidades en la base de datos. Es decir oculta la gestión de la base de datos y a su vez la tecnología que usamos en ella, dotando a cada clase persistente de un DAO asociado a ella. En éste proyecto hemos usado Spring Data JPA que nos proporciona una interfaz para la gestión de los objetos del dominio sin tener que escribir nosotros la implementacion de los métodos. Esto nos permite ganar agilidad a la hora de crear los DAOs ya que permite extender esta interfaz a nuestra clase DAO proporcionando todos los métodos CRUD, métodos para modificar o eliminar objetos de ese tipo. Éste interfaz se llama JpaRepository.

Otro motivo por el cual usamos JpaRepository es que permite extender la creación de una buena seria de métodos de búsqueda por el nombre de los atributos de nuestras clases persistentes simplemente definiendo los como un método de nuestro interfaz del DAO.

```
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {

    public Usuario findByCorreo(String correo);

    public Usuario findByNombre(String nombre);

    public List<Usuario> findByNombreContaining(String nombre);

    @Override
    public <S extends Usuario> S save(S usuario);

    @Override
    public void delete(Long idUsuario);

    @Override
    public boolean exists(Long idUsuario);

    @Override
    public <S extends Usuario> S saveAndFlush(S usuario);

    @Override
    public Usuario findOne(Long idusuario);
```

Figura 6.4: Ejemplo de interfaz con JpaRepository

6.4. Aplicación Móvil

La capa de presentación será accesible por el usuario a través de una aplicación móvil desarrollada en Android. Los aspectos mas relevantes serán expuestos a continuación.

6.4.1. Servicios

- **Google Maps API**. Es un servicio web proporcionado por google que nos permite usar los mapas en nuestras aplicaciones y también conocer la ubicación del usuario en todo momento. Esta ubicación la marca con un punto azul pero no suministra coordenadas(latitud y longitud).

- **Google Location**. Es un servicio de google que nos suministra las coordinadas del usuario, según varia la posición ciertos metros o cada cierto tiempo.

- **Rest SystemService** Este punto seria el que ejecuta nuestro propio servidor para guardar o devolver lo datos de los usuario, puntos de interés, rutas o grupos del usuario.

- **Firebase**

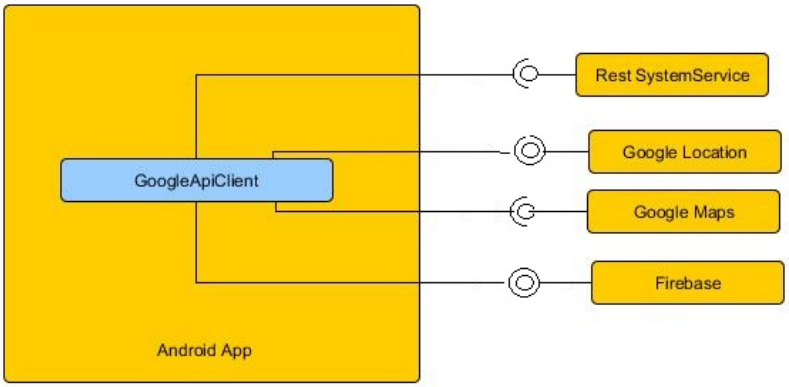

Figura 6.5: Servicios de la aplicación móvil

6.4.2. Organización de los paquetes

- Organizar paquetes por categoría
- Organizar paquetes por funcionalidades

6.4.3. Arquitectura de la aplicación móvil

- Arquitectura Orientada a Fragments

Capítulo 7

Conclusiones y trabajo futuro

Bla, bla, bla...

7.1. Investigación realizada

-
-

7.2. Características del proyecto

El uso de los paradigmas adecuados junto con herramientas consolidadas nos ha permitido construir una plataforma en la que servir una aplicación web donde los usuarios puedan buscar, valorar y obtener recomendaciones de contenidos audiovisuales.

A continuación se muestran las principales características del producto construido:

-
-

7.3. Trabajo futuro

Una vez finalizado este proyecto, se abren nuevas vías de trabajo, tanto de explotación comercial como de investigación científica:

-
-
-

Índice de Tablas

Índice de Figuras

Apéndice A

Glosario

ACID *Atomicity, Consistency, Isolation and Durability*: es un conjunto de propiedades que garantizan la fiabilidad de las transacciones de una base de datos. La atomicidad implica que cada transacción se ejecuta de forma completa o no se ejecuta. La consistencia garantiza que cada transacción llevará la base de datos de un estado consistente a otro consistente. El aislamiento permite que parezca que las transacciones se ejecutan de forma serializada cuando en realidad existe concurrencia. Por último, la durabilidad garantiza que una vez finalizada con éxito una transacción sus cambios se guardarán de forma persistente.

AM *Agile Modeling*: una metodología complementaria para modelar y documentar sistemas software mediante una serie de buenas prácticas basadas en el desarrollo ágil.

Big Data Es el término que se utiliza para llamar a los sistemas que procesan grandes colecciones de datos que son inmanejables bajo las herramientas tradicionales.

CF *Collaborative Filtering*: los algoritmos de filtrado colaborativo son una familia de algoritmos de recomendación cuyo funcionamiento se basa en generar sugerencias basándose en las preferencias de otros usuarios.

HDFS *Hadoop Distributed File System*: es un sistema de fichero distribuido, escalable y portable escrito en java y diseñado para servir de forma de almacenamiento a Hadoop.

IR *Information Retrieval*: la recuperación de información es la ciencia que estudia la obtención de información relevante a partir de una colección de documentos.

ML *Machine Learning*: el aprendizaje automático es una rama de la inteligencia artificial que busca construir y estudiar sistemas que sean capaces de aprender de los datos.

MVC *Model-view-controller*: es un patrón arquitéctonico para implementar interfaces de usuario. Divide el software en tres componentes. El modelo es el encargado de gestionar los datos y la lógica de negocio. La vista se encarga de la representación gráfica de la información. Por último, el controlador es el encargado de poner en comunicación los otros componentes.

NoSQL *Not Only SQL*: clase de sistemas de gestión de bases de datos que difieren del modelo relacional y cuyo principal objetivo es proporcional escalabilidad horizontal.

ORM *Object-Relational Mapping*: El mapeo objeto-relacional es una herramienta de programación que establece una correspondencia entre clases y objetos de un lenguaje de programación orientado a objetos con tablas y filas de una base de datos relacional siguiendo el patrón arquitectónico *active record*.

RDBMS *Relational DataBase Management System*: un sistema de gestión de base de datos relacional es un software diseñado para gestionar la creación, modificación y consulta de bases de datos bajo el esquema relacional.

RecSys *Recommender Systems*: los sistemas de recomendación tienen como objetivo predecir la preferencia de un usuario hacia un producto.

RM *Relevance Models*: forma abreviada empleada en esta memoria referirse a los *Relevance-based Language Models*, una técnica de IR que introduce el concepto de relevancia en los modelos de lenguaje estadísticos.

RPC *Remote Procedure Call*: es un tipo de comunicación entre procesos que permite la ejecución de una rutina en un equipo remoto.

RUP *Rational Unified Process*: se trata de la propuesta de IBM para implementar una metodología de desarrollo basada en el marco que establece el Proceso Unificado.

SQALE *Software Quality Assessment based on Lifecycle Expectations*: es un método genérico para evaluar la calidad de un código mediante unos índices y unos indicadores.

WSGI *Web Server Gateway Interface*: es una interfaz estándar de Python para la comunicación entre servidores y aplicaciones web.

Apéndice B

Bibliografía