Desarrollo de una aplicación para búsqueda y mantenimiento de rutas de senderismo

Juan Miguel Sánchez Begines
Plan de Estudios del Estudiante
Área del trabajo final

Francesc d'Assís Giralt Queralt
Carles Garrigues Olivella.

Junio de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	Desarrollo de una aplicación para búsqueda y mantenimiento de rutas de senderismo
Nombre del autor:	Juan Miguel Sánchez Begines
Nombre del consultor/a:	Francesc d'Assís Giralt Queralt
Nombre del PRA:	Carles Garrigues Olivella
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	Máster en Desarrollo de aplicaciones para dispositivos móviles
Área del Trabajo Final:	Trabajo final de Máster
Idioma del trabajo:	Español
Palabras clave	Senderismo, Aplicación móvil, Medio ambiente.

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

Este trabajo está orientado al desarrollo de una aplicación que permita al usuario buscar rutas para senderismo así como participar en el mantenimiento de las mismas.

Dada la creciente afición por esta actividad, es necesario no solo dar soporte a los usuarios para la búsqueda de estos espacios sino colaborar en su protección y cuidado.

Para ello, se pretende desarrollar una aplicación para dispositivos Android en la que se seguirá una metodología iterativa para ello.

Con esto, se espera conseguir no solo la aplicación en sí, sino una mayor concienciación de las personas. Además, en el aspecto personal del estudiante, se desea adquirir los conocimientos necesarios para desarrollar una aplicación para dispositivos móviles que abarque todo el ciclo de vida del software.

Abstract (in English, 250 words or less):

This work is oriented to the development of an application that allows the user to search for hiking routes as well as participate in the maintenance of them.

Given the growing fondness for this activity, it is necessary not only to support users to search for these spaces, but also to collaborate in their protection and care.

For this, it is intended to develop an application for Android devices in which an

iterative methodology will be followed for this.

With this, it is expected to achieve not only the application itself, but also a greater awareness of the people. In addition, in the personal aspect of the student, you want to acquire the necessary knowledge to develop an application for mobile devices that covers the entire life cycle of the software.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Hacer senderismo es una de las actividades cotidianas más realizadas hoy día. Incluso, aunque no sean personas aficionadas a ello, siempre puede surgir como un plan esporádico entre amigos o familiares.

Para ello, existen algunas aplicaciones que son muy útiles para buscar y conocer las rutas que se pueden realizar. Estas aplicaciones ayudan tanto a los usuarios a buscar las rutas como a contactar entre ellos para compartir esta experiencia.

Pero estas aplicaciones centradas en el usuario olvidan otro importante factor y es el mantenimiento y cuidado de dichas rutas. Es ahí donde entra nuestra aportación. En ninguna de las aplicaciones existentes se ha visto un apartado para reportar el mal estado o la existencia de productos contaminantes de estas rutas, alguna herramienta o forma de mantener estos sitios que al final son responsabilidad de todos.

Actualmente, para informar sobre el mal estado de alguna de estas rutas, se realiza mediante quejas a los ayuntamientos de la zona. Para realizar este trámite el ciudadano carece de total información de cómo poner en conocimiento del mismo el estado de estos espacios. Además, para realizar estos trámites, se suele requerir mucha documentación e información. Todo para que muchas de las quejas finalmente no lleguen a buen puerto.

Es por ello que se va a desarrollar una aplicación, no solo centrada en el usuario sino en la conservación de estos lugares necesarios para la continuidad de estas actividades y preservar los parajes naturales.

Esta necesidad surge del cada vez mayor deterioro de estos lugares y que en muchos casos se ve ocasionada por desinformación de a quién acudir para realizar las reparaciones o limpiezas pertinentes.

Con esta aplicación no solamente se pretende poder contactar con los ayuntamientos u otras organizaciones del medio ambiente sino además que por iniciativa de los propios senderistas puedan formar grupos de limpieza o se haga una concienciación sobre este tema.

1.2. Estudio del mercado

Para llevar a cabo el desarrollo de esta aplicación se ha realizado un estudio del mercado actual sobre aplicaciones similares a la que planteamos en este Trabajo Fin de Máster.

Es necesario realizar este estudio para asegurarnos que nuestras aportaciones son novedosas e importantes para este ámbito así como para poder buscar sinergias entre las mismas o poder alimentar nuestra aplicación.

Entre las aplicaciones más importantes se encuentran (1), (2):

1. Wikiloc: Esta aplicación está considerada la mejor aplicación para senderismo actualmente. Disponible para IOS y Android, esta aplicación ofrece una gran cantidad de rutas por todo el mundo así como diferentes modalidades como rutas en bicicleta, a caballo, esquí, etc. Esta aplicación permite la búsqueda de rutas por niveles de dificultad y por comunidades autónomas.
2. Strava: Esta aplicación ofrece rutas para senderismo, cicloturismo o trekking similar a Wikiloc. Administra las rutas realizadas mediante la web y ofrece seguimiento GPS lo que resulta muy práctico. Además de esto, ofrece una red social para los deportistas y permite llevar un seguimiento muy exhaustivo del rendimiento a lo largo de toda la ruta. Disponible en Android e IOS
3. Endomondo: Esta aplicación es diferente al resto, es más un complemento a las anteriores dado que su funcionalidad está más orientada al seguimiento de la actividad física o realizar desafíos personales o en grupo. Finalmente, también dispone de una red social para deportistas. Disponible en Android e IOS
4. MapMyHike: Esta aplicación permite consultar rutas, diseñar recorridos, realizar un listado de rutas favoritas y ponerse en contacto con otros senderistas. Esta app permite conectar varios dispositivos a la vez, incluso algunos modelos de zapatillas de deporte. Disponible en Android e IOS
5. ViewRanger: Esta aplicación ofrece una serie de mapas especializados en senderismo y ciclismo. Permite el uso offline de la aplicación. Ofrece una guía de las rutas y grabaciones de las rutas. Finalmente, ofrece características de realidad aumentada para ayudarte en el trayecto de la ruta. Disponible en Android e IOS

Como se puede ver en todas las aplicaciones aquí presentadas, son aplicaciones centradas en el usuario, en la búsqueda de rutas de senderismo y en el aspecto colaborativo y rendimiento de la realización de las rutas pero carecen de un espacio centrado en las propias rutas de senderismo para su cuidado o para informar a los responsables del mismo.

1.3. Objetivos del Trabajo

- Adquirir los conocimientos necesarios para realizar las distintas fases del ciclo de vida de desarrollo de una aplicación móvil
- Desarrollar una aplicación en el lenguaje de programación Android.
- Facilitar al usuario la búsqueda de rutas de senderismo.
- Facilitar al usuario compartir las diferentes rutas de senderismo que realizan.

- Ayudar a mantener estos parajes naturales limpios de basuras.
- Promover la relación entre los usuarios que comparten esta afición.

1.4. Enfoque y método seguido

Para el desarrollo del Trabajo Fin de Máster, se pueden plantear las siguientes estrategias:

- Desarrollo de un producto nuevo: Realizar una aplicación desde cero añadiendo nuestro propio modelo de datos, filtrados y módulos de nuestra aplicación.
- Desarrollo de un producto a partir de uno existente: Realizar una aplicación a partir de una ya existente, añadiendo las funcionalidades diferenciales que se quiere aportar a lo que hay en el mercado.
- Desarrollo de un producto con fuente de datos externas: Realizar una aplicación ya sea a partir de una existente o no que recoja la información de alguna aplicación existente.

Partiendo de estas posibles estrategias que se plantean se ha decidido realizar un Desarrollo de un producto nuevo por los siguientes motivos:

- Al realizar un producto nuevo, se parte sin restricciones para el desarrollo y por lo tanto, nos facilita mucho más el desarrollo
- Permite al desarrollador conocer mejor la aplicación y por lo tanto, le da más control sobre la misma. Este es un aspecto muy importante para la resolución de errores y la fase de pruebas.
- Mantiene una base de datos limpia, sin datos innecesarios.

Precisamente, las ventajas que presenta la opción elegida es la que nos hace descartar las anteriores, no porque sean peores opciones sino que dado el dominio de nuestro problema, los factores mencionados como ventajas son los que veo necesario para que el desarrollo se lleve a cabo de forma correcta.

Para llevar a cabo la estrategia seleccionada anteriormente se va a seguir la metodología de desarrollo iterativo e incremental. La elección de esta metodología viene dada por las características del proyecto.

Como se ha descrito anteriormente, la aplicación que se pretende desarrollar tiene unos bloques funcionales muy delimitados como son la gestión de los usuarios, la gestión de las rutas y la gestión del medio ambiente. La intención con el uso de esta metodología (3) es la de ir produciendo software usable desde un primer momento de manera que se puede ir teniendo una visión de la aplicación que ayude a tomar decisiones en las futuras decisiones. Por otro lado, dado que la dedicación para la realización del TFM es variable, esta metodología ayuda a minimizar los riesgos en este sentido ya que por cada entregable permite valorar el estado en el que se encuentra el proyecto.

Finalmente, destacar que el estudiante no conoce la complejidad que va a suponer el desarrollo de la aplicación y por lo tanto, la utilidad de esta metodología que trabaja por iteraciones permite evaluar dicha

complejidad y realizar divisiones en las iteraciones planteadas en un primer momento para controlar los entregables.
Para entender un poco mejor este ciclo de vida, se puede seguir la siguiente figura.

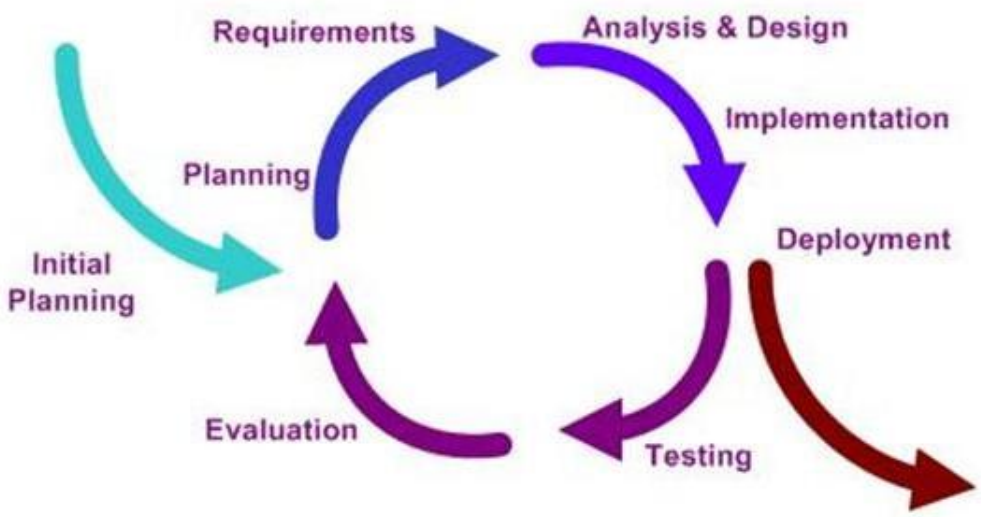


Ilustración 1 Desarrollo iterativo e incremental

En cada una de las iteraciones, se desarrolla una iteración del ciclo de vida completo para la parte acotada que se pretende desarrollar en esa iteración. A partir de la primera, en el resto de iteraciones se realizará las mismas tareas para las nuevas funcionalidades (parte incremental del ciclo de vida) pero añadiendo mejoría a lo ya desarrollado (parte iterativa del ciclo de vida).

Una vez destacadas las características por las que hemos seleccionado este ciclo de vida, se va a valorar la elección del mismo comparando con otros ciclos de vidas cotidianos como son Scrum y el ciclo de vida tradicional o en cascada.

Con respecto al ciclo de vida tradicional claramente se puede ver la facilidad con la que una mejora o cambio en la aplicación puede suponer un coste muy grande en el desarrollo de la aplicación además de que todos los requisitos tienen que estar muy claros desde el inicio y admite poca gestión de riesgos. Por lo tanto, dado el escenario en el que se desarrolla este proyecto, el ciclo de vida en cascada no es un buen candidato.

Otro de los ciclos de vida más usados es Scrum que podría encajar perfectamente en nuestro desarrollo. De hecho, el ciclo de vida iterativo incremental es padre de esta metodología. La elección simplemente está hecha a gusto del estudiante puesto que, en mi opinión Scrum es una metodología que aporta mucho a equipos de trabajo más que a un desarrollador individual dada las fases que plantea como la fase de planificación, restrospectiva, etc.

1.5. Planificación del Trabajo

En cuanto a los recursos necesarios para llevar a cabo el trabajo hay que tener en cuenta la misma descripción del trabajo. Como se ha dicho anteriormente, este trabajo consiste en el desarrollo de una aplicación Android.

Es por ello que este trabajo necesita de los siguientes recursos:

- Ordenador/Portátil con capacidad necesaria para utilizar la herramienta Android Studio y un emulador Android
- Tener instalado en dicho dispositivo las herramientas mencionadas.
- Tener Java y Android SDK instalados.

Actualmente se dispone de un ordenador portátil MSI GE72 2QF con Android Studio y emulador instalados. Además, ya están instalados y configurados tanto Java como el SDK de Android por lo que se dispone de todos los recursos necesarios para el desarrollo del trabajo.

A continuación se pasa a detallar las tareas que se van a desarrollar a lo largo del trabajo fin de máster. Para hacer el seguimiento del trabajo, se va a hacer uso de la aplicación online Trello.

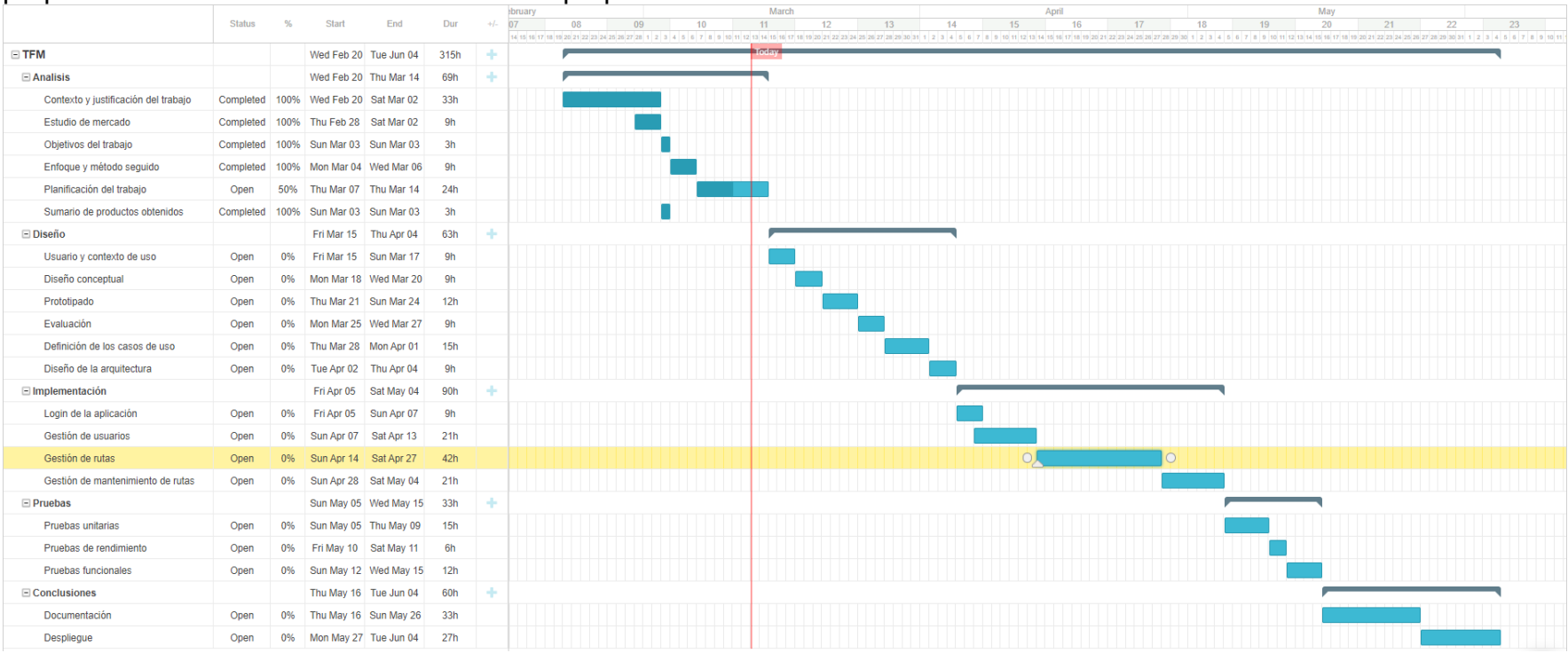
Según las fases del ciclo de vida, las tareas por cada uno de ellos son las siguientes:

- Análisis
    - Contexto y justificación del trabajo
    - Estudio de mercado
    - Objetivos del trabajo
    - Enfoque y método seguido
    - Planificación del trabajo
    - Sumario de productos obtenidos
- Diseño
    - Usuario y contexto de uso.
    - Diseño conceptual.
    - Prototipado
    - Evaluación
    - Definición de los casos de uso
    - Diseño de la arquitectura
- Implementación
    - Login de la aplicación
    - Gestión de usuarios
    - Gestión de rutas
    - Gestión de mantenimiento de rutas
- Pruebas
    - Pruebas unitarias
    - Pruebas de rendimiento
    - Pruebas funcionales
- Conclusiones
    - Documentación
    - Despliegue

Finalmente, se expone una planificación temporal siguiendo las entregas propuestas en la plataforma de la UOC y estableciendo las fechas de cada uno de los hitos propuesto en la asignatura al final de cada una de las fases del ciclo de vida (salvo implementación y pruebas que van en un único entregable). Hay que tener en cuenta que se va a trabajar los días festivos con la misma dedicación que los días laborales.

En primer lugar, se presenta una vista general de todas las actividades a realizar en el desarrollo del trabajo fin de máster. En él se puede ver que

Las fechas planteadas concuerdan con las fechas finales establecidas en las entregas de la asignatura. Además, se propone una estimación inicial de las tareas propuestas inicialmente.



	Status	%	Start	End	Dur
TFM			Wed Feb 20	Tue Jun 04	315h
Analisis			Wed Feb 20	Thu Mar 14	69h
Contexto y justificación del trabajo	Completed	100%	Wed Feb 20	Sat Mar 02	33h
Estudio de mercado	Completed	100%	Thu Feb 28	Sat Mar 02	9h
Objetivos del trabajo	Completed	100%	Sun Mar 03	Sun Mar 03	3h
Enfoque y método seguido	Completed	100%	Mon Mar 04	Wed Mar 06	9h
Planificación del trabajo	Open	50%	Thu Mar 07	Thu Mar 14	24h
Sumario de productos obtenidos	Completed	100%	Sun Mar 03	Sun Mar 03	3h
Diseño			Fri Mar 15	Thu Apr 04	63h
Usuario y contexto de uso	Open	0%	Fri Mar 15	Sun Mar 17	9h
Diseño conceptual	Open	0%	Mon Mar 18	Wed Mar 20	9h
Prototipado	Open	0%	Thu Mar 21	Sun Mar 24	12h
Evaluación	Open	0%	Mon Mar 25	Wed Mar 27	9h
Definición de los casos de uso	Open	0%	Thu Mar 28	Mon Apr 01	15h
Diseño de la arquitectura	Open	0%	Tue Apr 02	Thu Apr 04	9h
Implementación			Fri Apr 05	Sat May 04	90h
Login de la aplicación	Open	0%	Fri Apr 05	Sun Apr 07	9h
Gestión de usuarios	Open	0%	Sun Apr 07	Sat Apr 13	21h
Gestión de rutas	Open	0%	Sun Apr 14	Sat Apr 27	42h
Gestión de mantenimiento de rutas	Open	0%	Sun Apr 28	Sat May 04	21h
Pruebas			Sun May 05	Wed May 15	33h
Pruebas unitarias	Open	0%	Sun May 05	Thu May 09	15h
Pruebas de rendimiento	Open	0%	Fri May 10	Sat May 11	6h
Pruebas funcionales	Open	0%	Sun May 12	Wed May 15	12h
Conclusiones			Thu May 16	Tue Jun 04	60h
Documentación	Open	0%	Thu May 16	Sun May 26	33h
Despliegue	Open	0%	Mon May 27	Tue Jun 04	27h

Ilustración 2: Planificación general

Entrando en detalla en cada una de las fases del ciclo de vida, se va a ir mostrar cada una de las fases.

En primer lugar se presenta la fase de análisis en la que se ha desarrollado el material de la primera entrega del trabajo fin de máster en la que se ha introducido un apartado nuevo como es el estudio de mercado sobre aplicaciones sobre senderismo. Este apartado se ha añadido como un apartado extra a la plantilla propuesta a petición del tutor del trabajo.

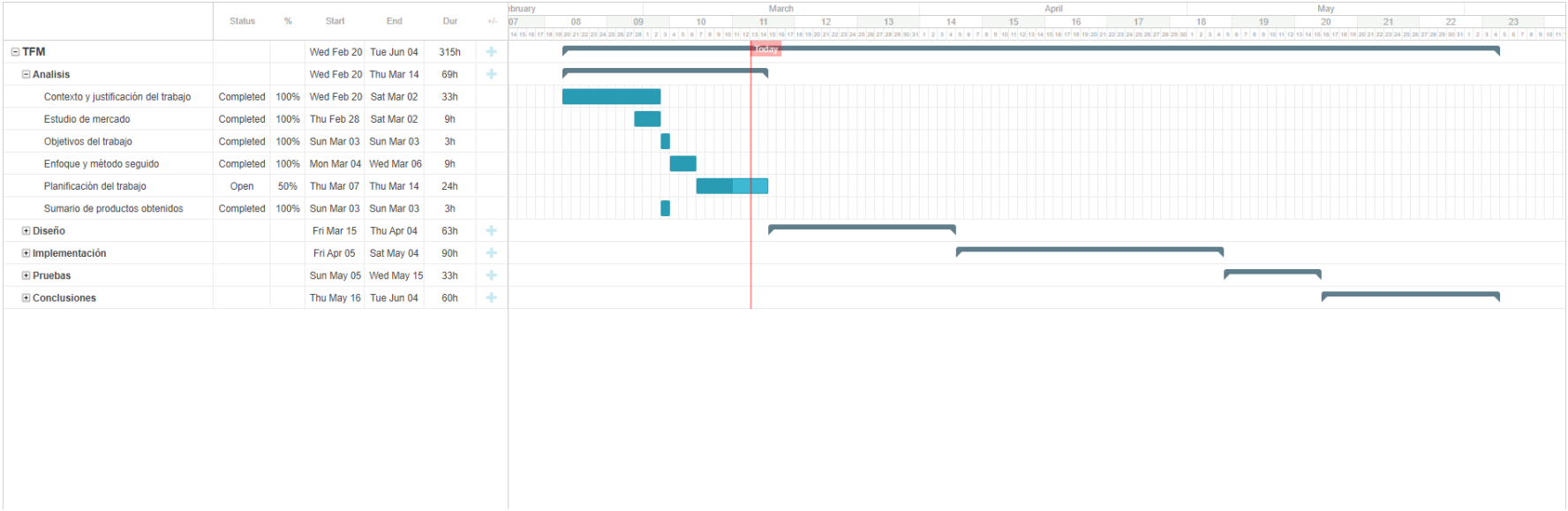


Ilustración 3: Fase Análisis

En segundo lugar, se plasman las tareas de la segunda PEC del trabajo final del master que recoge las tareas sobre el diseño. En esta fase, se ha repartido las tareas en un tiempo equitativo puesto que se considera que las tareas a desarrollada tienen una complejidad similar.

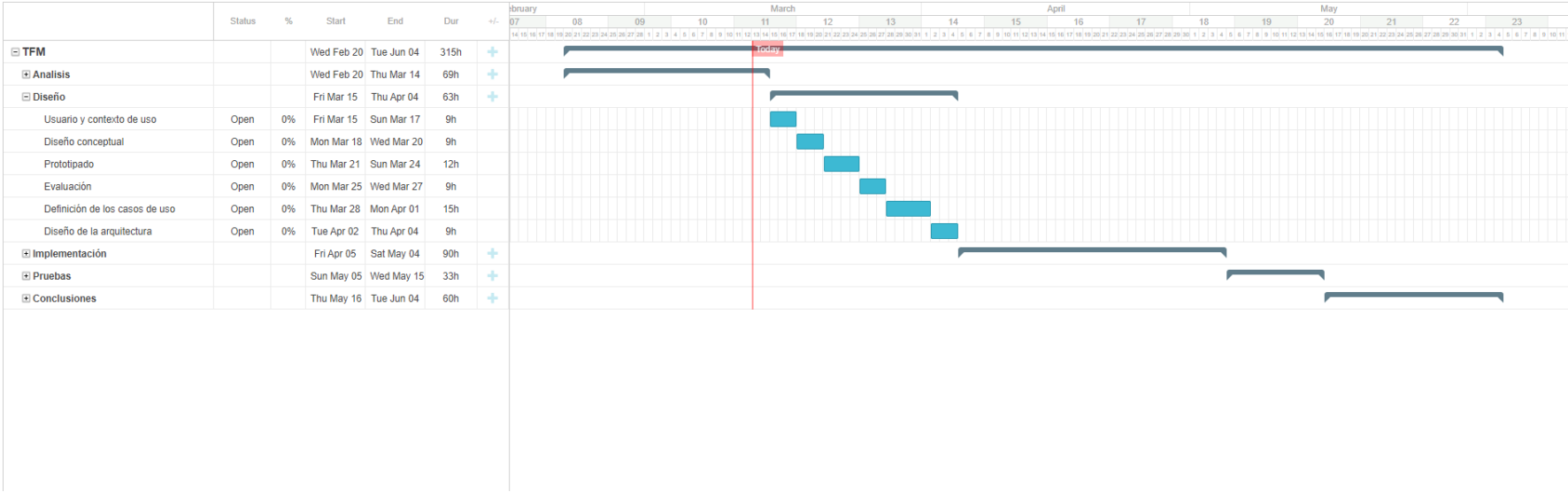


Ilustración 4: Fase Diseño

En tercer lugar, se presenta dos fases del ciclo de vida que corresponde con lo solicitado en la tercera PEC del trabajo y que compondrán el grueso de nuestro trabajo: el desarrollo de la aplicación, del producto final junto con la memoria que nos ocupa.

Como se puede en la planificación, la mayor parte del tiempo está destinada al desarrollo de la aplicación y en menor medida (10 días) a la realización de las pruebas de ésta.

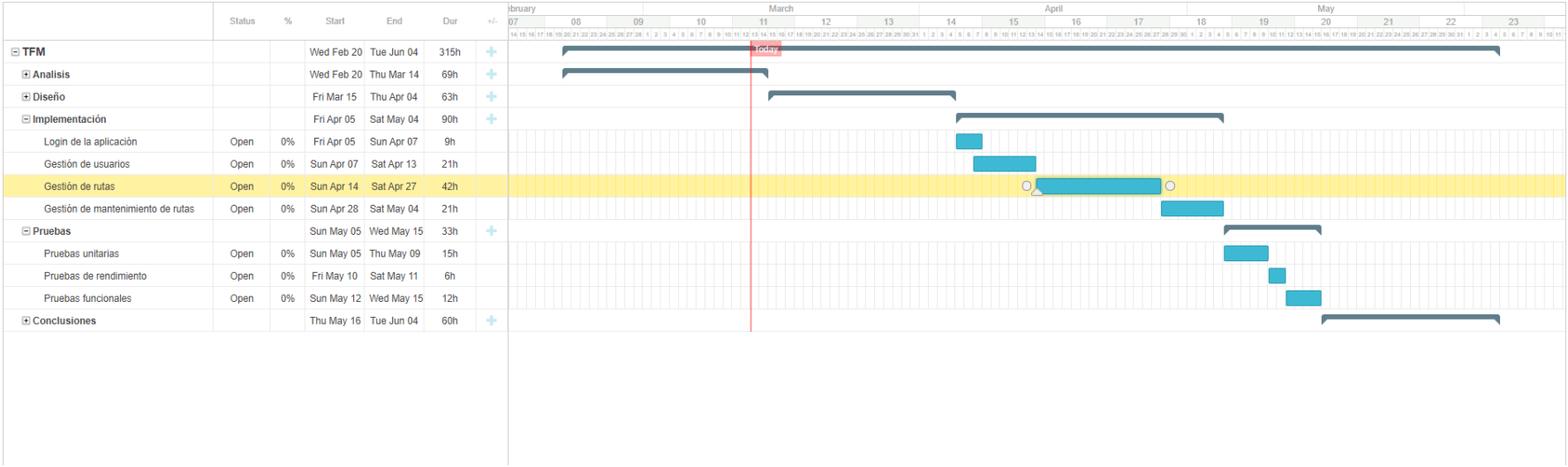


Ilustración 5: Fase Implementación & Pruebas

Para acabar, se establece un periodo final para realizar las últimas consideraciones sobre el trabajo desarrollado y preparar la documentación para la presentación final.

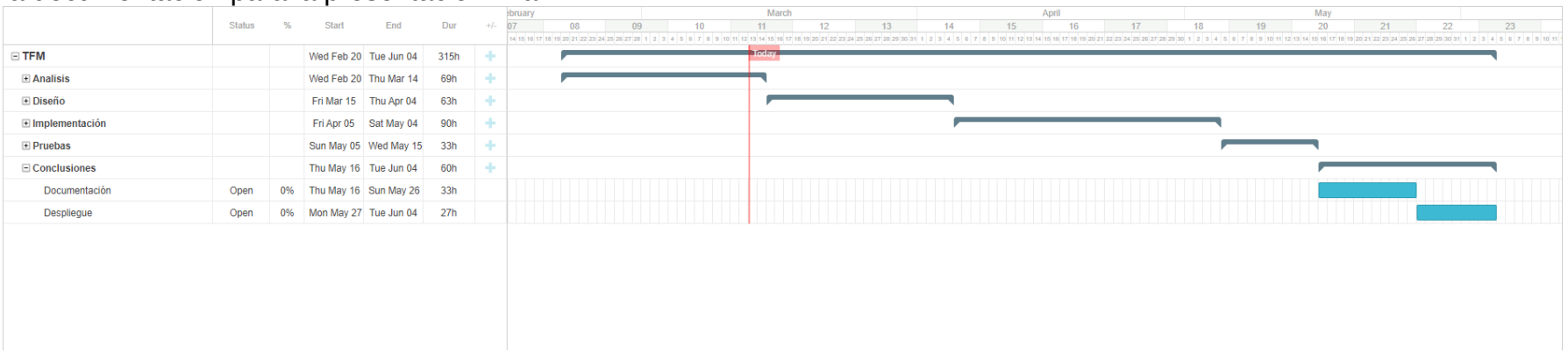


Ilustración 6: Fase Conclusiones & Entrega Fina

Esta estimación, es una estimación inicial que puede variar conforme se vaya desarrollando el proyecto pero siempre es bueno establecerse una serie de objetivos para tener un camino y un objetivo claro durante el desarrollo.

1.6. Justificación de decisiones técnicas

El lenguaje de programación seleccionado para desarrollar el proyecto del Trabajo Fin de Máster es Android principalmente porque el desarrollador tiene más conocimientos de programación en este lenguaje con respecto a IOS o algún lenguaje híbrido. Además, el lenguaje Android es el más extendido hoy día ya que el número de dispositivos Android es mayor al de IOS u otros lenguajes.

Para el desarrollo del backend se plantea utilizar ExpressJS con base de datos Mongo DB. Se han valorado otras tecnologías como el desarrollo de una API en Ruby on Rails o Spring con Java ya que el desarrollador tiene conocimientos en estos lenguajes pero se ha optado por ExpressJS porque se desea adquirir los conocimientos en este lenguaje y dado que no hay que realizar un desarrollo demasiado extenso en la parte de backend es una buena oportunidad para poder iniciarse en dicho lenguaje.

Destacando otra ventaja de ExpressJS con respecto a los anteriores propuestos, ExpressJS necesita de menos configuración y ficheros para desarrollar la aplicación y dado que el dominio de nuestro problema no conlleva un desarrollo muy amplio en la parte de backend, esta opción es mucho más limpia.

Con respecto a la base de datos Mongo DB ha sido también seleccionado por la oportunidad que supone para el desarrollador ampliar los conocimientos en nuevas tecnologías y porque se trata de una base de datos ligera que encaja perfectamente con los requisitos de la aplicación.

1.7. Breve sumario de productos obtenidos

Los productos que deberemos obtener al finalizar el trabajo fin de máster son los siguientes:

- Memoria del ciclo de vida de desarrollo de la aplicación propuesta.
- Aplicación móvil para dispositivos Android de la aplicación propuesta.
- Presentación para el tribunal y video demostrativo.

1.8. Breve descripción de los otros capítulos de la memoria

En el siguiente punto se pasa a explicar brevemente los siguientes capítulos que se desarrollarán a lo largo de la memoria:

- Diseño: En este apartado se tratarán los aspectos de Diseño Centrado en el usuario (DCU) así como aspectos más técnicos del diseño como

son los casos de uso a desarrollar y la arquitectura que se va a seguir para el desarrollo de la aplicación.

- Implementación: En este apartado se mostrarán los aspectos más relevantes llevados a cabo en el desarrollo de la aplicación: lenguaje, framework y métodos o funciones relevantes en nuestro desarrollo.
- Pruebas: En este punto, se presentará el plan de pruebas realizado para la aplicación concerniente a las pruebas unitarias, funcionales y de rendimiento.
- Conclusiones: Finalmente, en este punto se recogerán una serie de conclusiones finales y posibles trabajos futuros a partir del resultado final de la aplicación.

2. Diseño

2.1. Usuarios y contexto de uso

Para el análisis de los posibles usuarios que van a utilizar la aplicación se ha acudido a una serie de personas que habían mostrado interés ante la idea de la aplicación que se va a desarrolla en este trabajo fin de máster. Para la extracción de información se ha hecho uso de la simple observación así como de dinámica de grupo con los posibles usuarios.

En primer lugar, se ha pedido a estos usuarios que se descarguen alguna aplicación para buscar una ruta para hacer senderismo y que comenten sus primeras impresiones. La mayor parte ha quedado satisfecha aunque como comentario han dicho que los filtros no son los adecuados.

En segundo lugar, se les ha preguntado que debería de tener una aplicación de este tipo para que fuera perfecta. La respuesta más consensuada ha sido la de un buen buscador y la posibilidad de interactuar con otros usuarios dentro de la aplicación. Además, como un detalle a tener en cuenta, si fuera posible hacer uso de la aplicación de manera offline aunque no cubra todas las funcionalidades.

La tercera pregunta ha sido sobre el apartado del medio ambiente que se pretende meter y salvo una persona al resto de los participantes les ha parecido una gran idea.

Por último, se les ha planteado un supuesto. En caso de que existiese una aplicación que contemple todo lo comentado, donde y cuando le darían uso a dichas aplicación. La mayoría obviamente contesto durante la realización o previamente a realizar la ruta. Además, guiando la entrevista grupal, se pudo sacar otros momentos en los que hacer uso de la misma como puede ser para hacer planes, para organizaciones de medio ambiente o simplemente por conocimiento o información de rutas de senderismo.

Así pues, tras esta dinámica de grupo y las observaciones pertinentes, quedan definidas las necesidades de los usuarios, las limitaciones (offline) y el contexto de uso de la aplicación que se va a desarrollar.

2.2. Diseño conceptual

Para la realización del estudio de posibles usuarios se ha analizado los hábitos y gustos de algunas personas las cuales participó en la dinámica de grupo. Con esta información se plantearán una serie de escenarios que se presentan a continuación

Nombre	Aguas Santas
Edad	27
Nivel de estudios	Graduada
Profesión	Maestra
Descripción de la persona	
Aguas Santas está soltera y viviendo aún con sus padres en Sevilla. Actualmente, está estudiando para prepararse las oposiciones de maestra de educación infantil lo que normalmente supone muchísimas horas de estudios diarias. Aguas Santas suele escuchar música o salir a pasear para despejarse. Además, los fines de semana realiza alguna escapada a la sierra con su pareja.	
Descripción de un escenario	
Hoy Aguas Santas ha salido a dar una vuelta con una amiga para despejarse un poco del estudio. Durante la conversación, Aguas Santas le dijo a su amiga que quería hacer alguna actividad en el fin de semana porque llevaba un largo periodo encerrada en casa para estudiar. Su amiga le dijo que ella iba a ir de senderismo, que había encontrado una aplicación para formar grupos y salir de senderismo con otras personas. Aguas Santas le gustó la idea por lo que se registró en la aplicación y se unió al plan.	

Nombre	Leticia
Edad	25
Nivel de estudios	Máster
Profesión	Ingeniera de la Salud
Descripción de la persona	
Leticia es una chica que vive en Sevilla en un piso compartido dado que está trabajando en la Universidad en un grupo de investigación. Sus aspiraciones profesionales pasan por obtener un doctorado que le permita dar clases en la universidad. A Leticia le gusta mucho viajar y es por ello que le encanta su trabajo. Además, es una chica que le gusta conocer y relacionarse con gente. Ella suele salir de fiesta con sus amigos cada fin de semana pero además le gusta hacer otro tipo de actividades como los escape room, el senderismo o ir a la playa.	
Descripción de un escenario	
Leticia había salido de trabajar con su compañera y estaban hablando sobre las rutas de senderismo que habían realizado este mes. Ellas son muy competitivas así que entre ellas se decían que habían hecho más kilómetros que la otra. Rápidamente, Leticia saco el móvil y miró en el perfil de su aplicación de senderismo. Su compañera hizo lo mismo y efectivamente, Leticia había conseguido hacer más kilómetros este mes que su compañera aunque en el cómputo global, ella llevaba más kilómetros realizados. Finalmente quedaron en volver a comprobarlo el mes siguiente.	

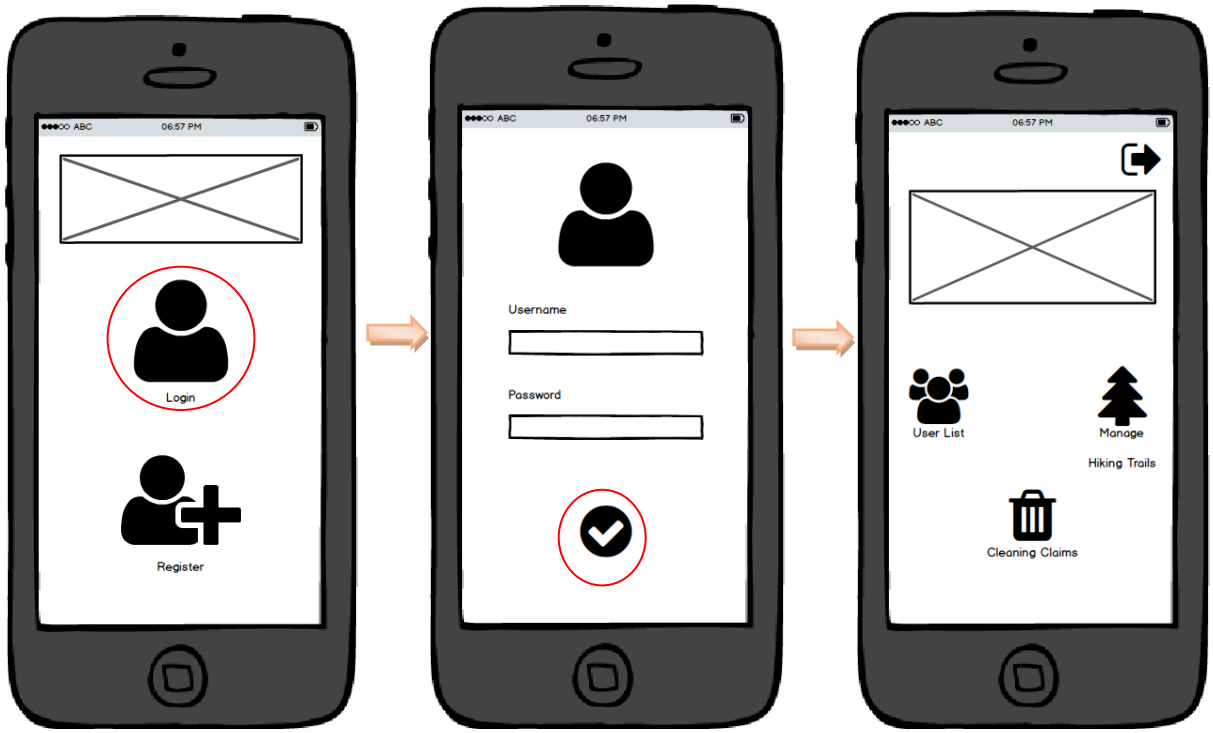
Nombre	Jesús
Edad	29
Nivel de estudios	Graduado
Profesión	Ingeniero de la Salud
Descripción de la persona	
Jesús es un hombre que vive compartiendo piso con su mejor amigo en Sevilla. Actualmente, está trabajando de informático mientras termina su carrera de Ingeniería de la Salud. Jesús es un chico hiperactivo al que le encanta realizar actividades de todo tipo pero sobre todo deporte y salir con sus amigos. Es un amante de la naturaleza y de los animales. Le encanta la tecnología, ir al gimnasio y practicar senderismo.	
Descripción del escenario	
Jesús acababa de salir de la protectora de animales como cada jueves y ahora tocaba una charla sobre medio ambiente. De camino a la asociación, se unió a varios compañeros que venían conversando sobre lo sucias que estaban las rutas a las que habían ido el fin de semana pasado. Jesús, sacó su móvil y le instó a solicitar una limpieza de esa ruta mediante su aplicación móvil. Además, contacto con un grupo de personas que también habían reportado esos problemas para formar un grupo de limpieza. Esto también lo iban a proponer en la asociación.	

A continuación, se va a plantear un flujo de interacción de la aplicación donde se expondrá las diferentes transiciones que tendrá la aplicación por las diferentes pantallas. Esto permitirá visualizar las diferentes transiciones por las que se debe pasar para realizar una acción. Esto permitirá tener una primera impresión sobre la aplicación y su navegabilidad.

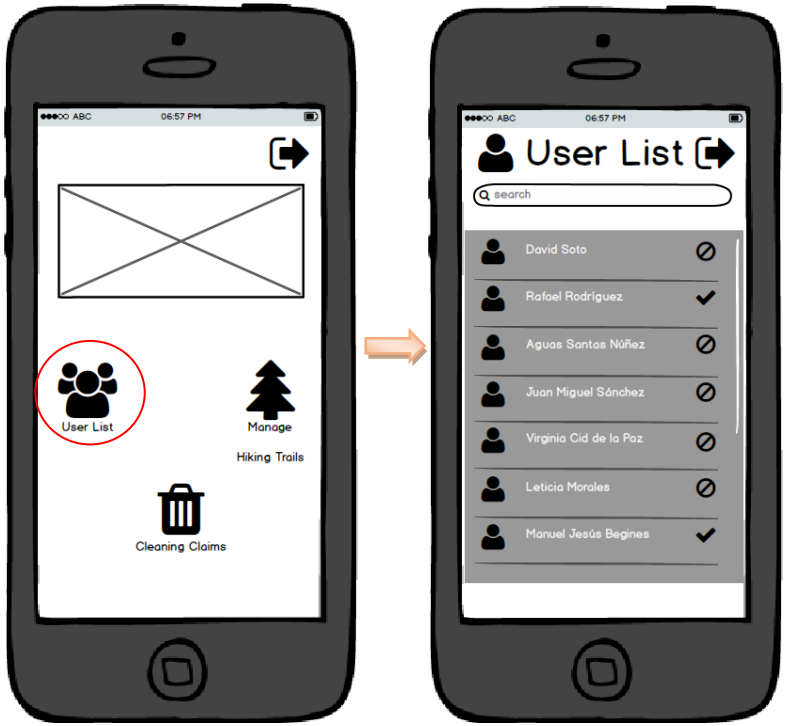
Para la realización de dicho flujo, se optará por una estructura jerárquica que diferencie los roles de administrador y usuario.

2.2.1. Administrador

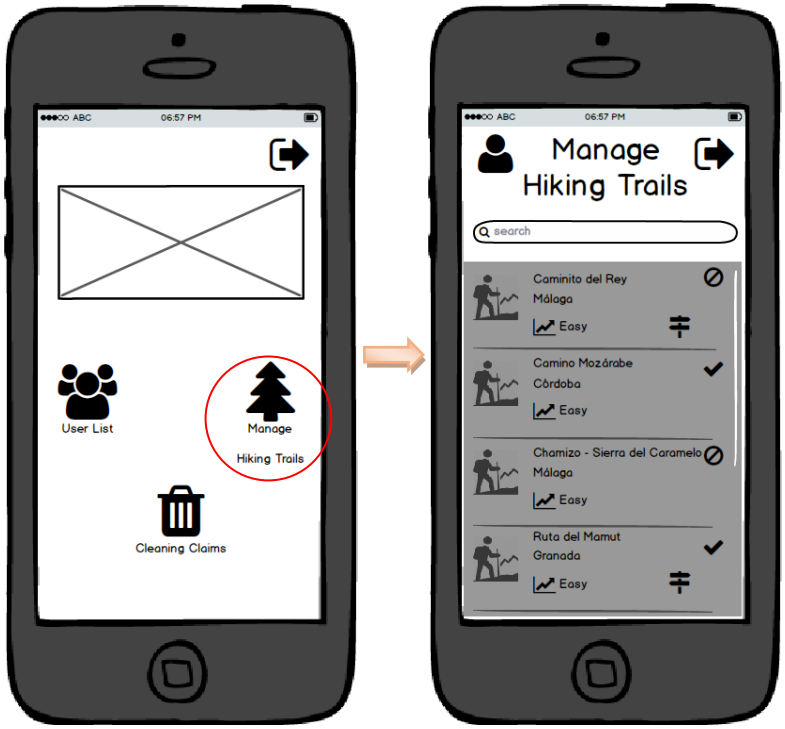
Login Administrador



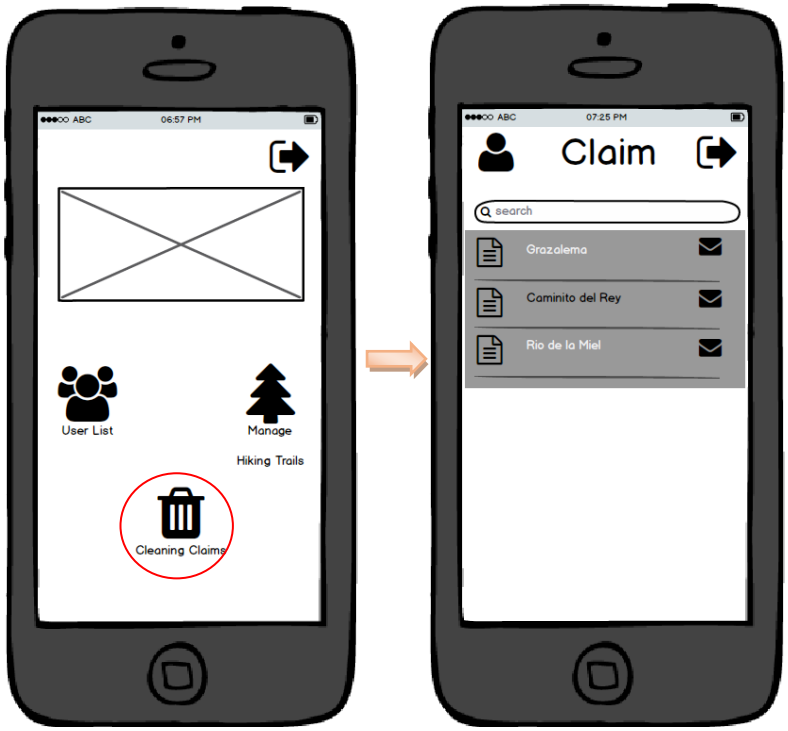
User List


ABC
06:57 PM
User List
Manage
Hiking Trails
Cleaning Claims
ABC
06:57 PM
User List
search
David Soto
Rafael Rodríguez
Aguas Santas Núñez
Juan Miguel Sánchez
Virginia Cid de la Paz
Leticia Morales
Manuel Jesús Begines

Manage Hiking Trails



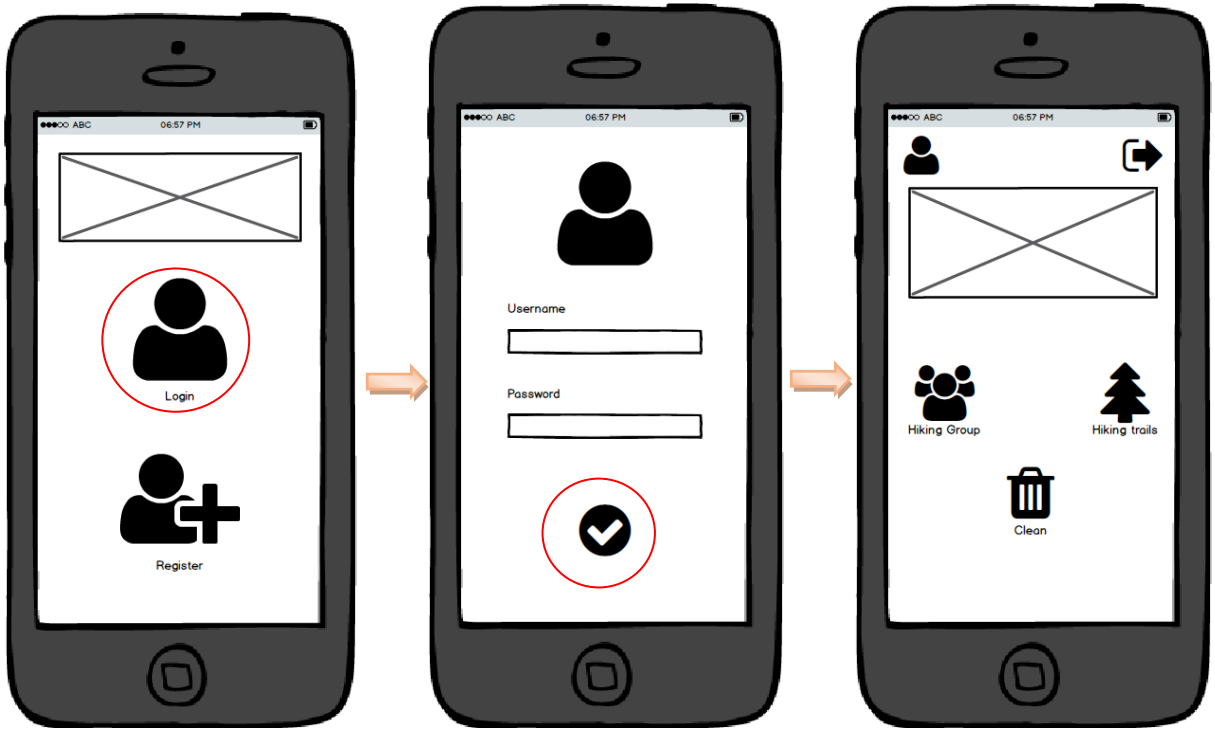
Cleaning Claims



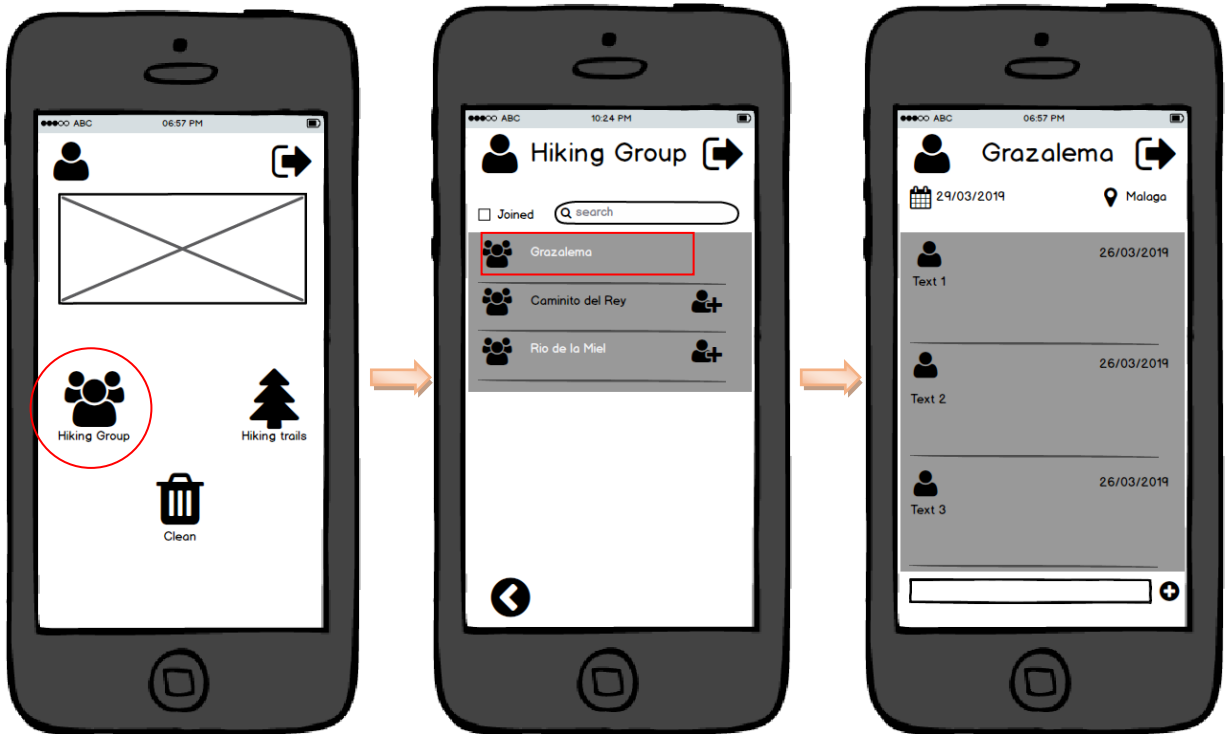
2.2.2. Usuario

Registro

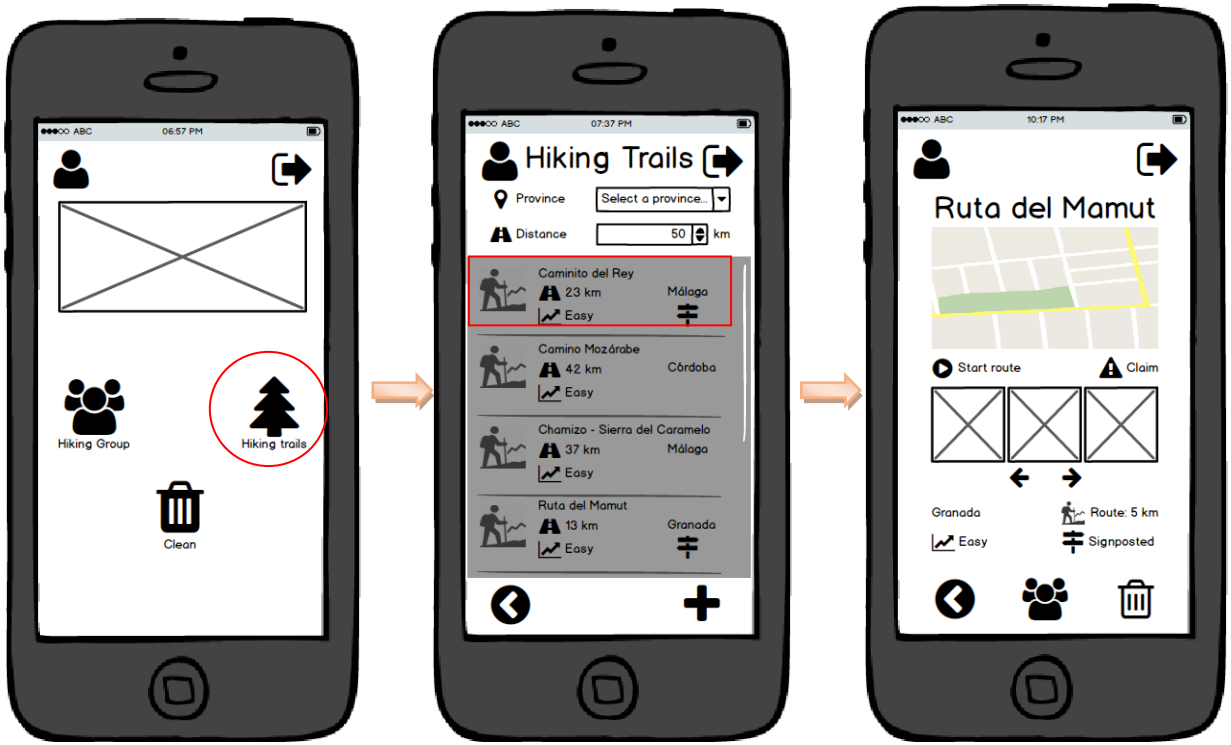
Login de Usuario



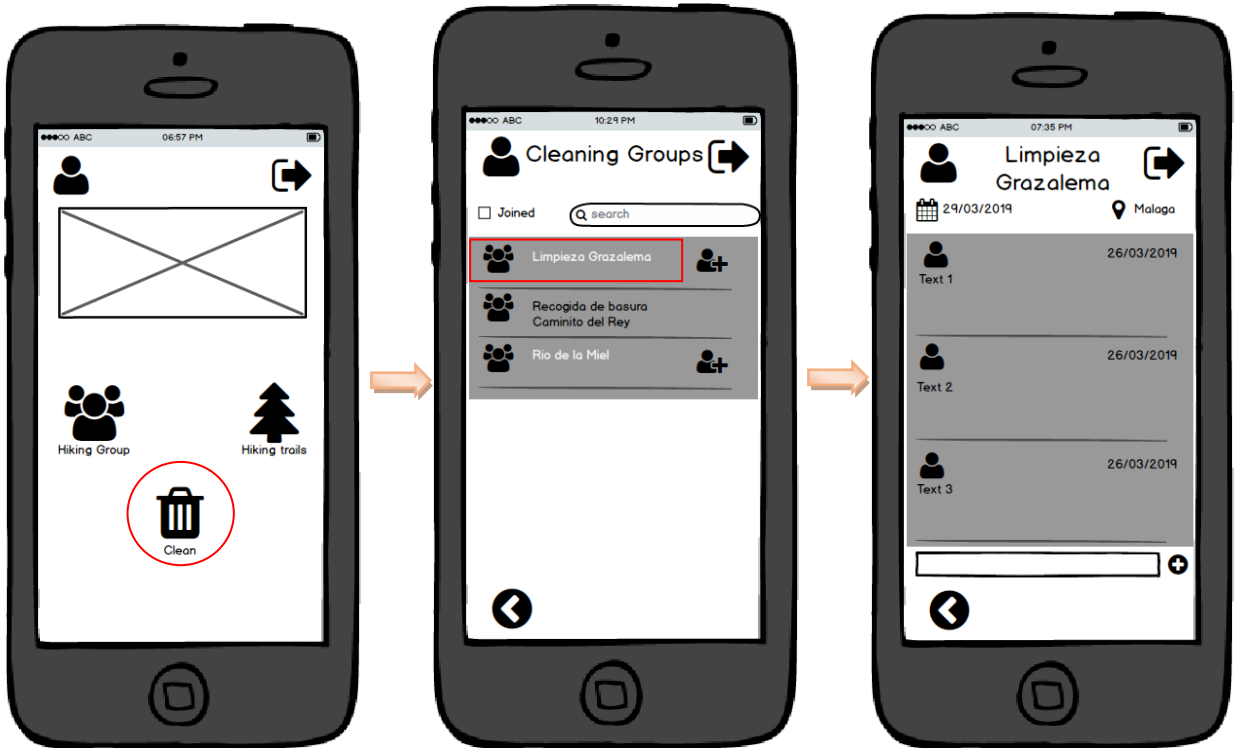
Hiking Group



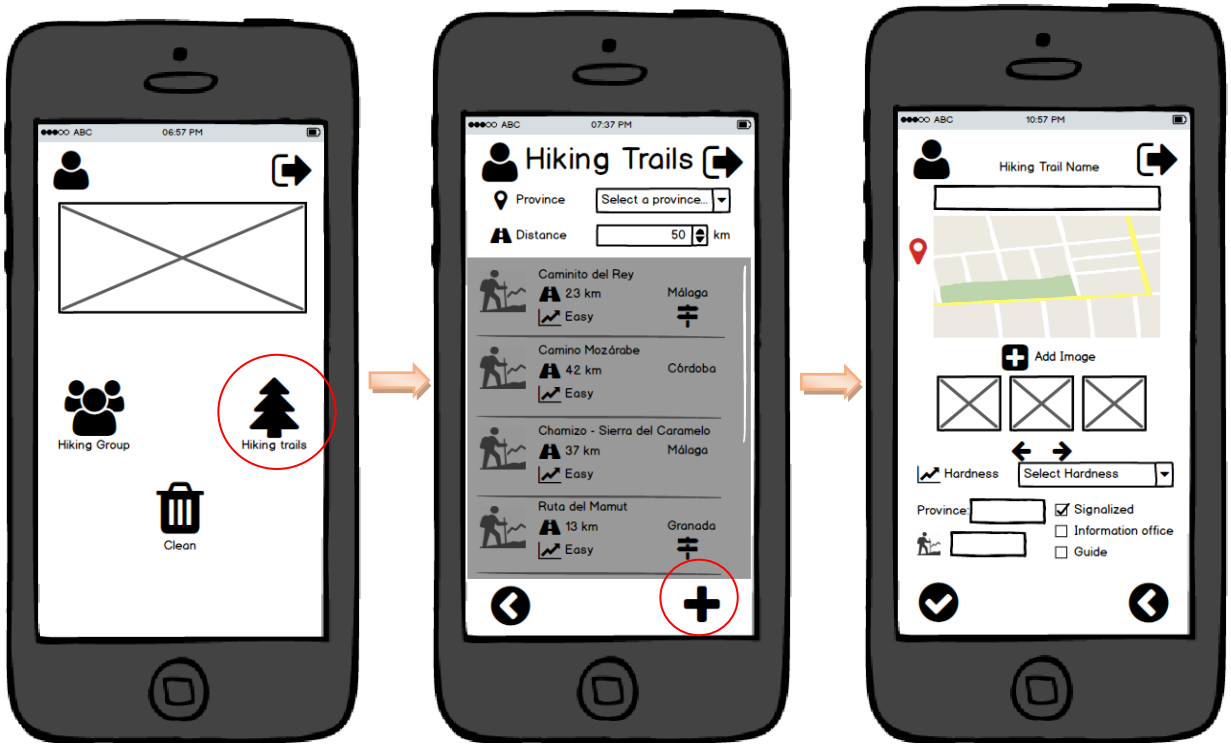
Hiking Trails



Cleaning Claim



Create Hiking Trail



Create Hiking Group

Create Cleaning Group

2.3. Prototipado

Para la realización del prototipo de alta definición se ha hecho uso de la herramienta Justinmind. Esta herramienta nos permite añadir algunos comportamientos que hace más real el prototipo creado para nuestra aplicación. Con esto, se pretende conseguir un prototipo que permita hacer pruebas con usuarios reales para poder detectar problemas de diseño o usabilidad antes de comenzar con el desarrollo.

Para poder acceder al prototipo se ha generado un enlace público para ello. El enlace es el siguiente: prototipo.

Con este prototipo se podrá dar al usuario final una primera toma de contacto con la que será la aplicación final. Con esto, se pretende conseguir un prototipo que permita hacer pruebas con usuarios reales para poder detectar problemas de diseño o usabilidad antes de comenzar con el desarrollo.

2.4. Evaluación

Para la evaluación del prototipo se ha definido una encuesta que nos permitirá conocer las sensaciones de los usuarios una vez probado el prototipo. En esta encuesta, se realizan preguntas acerca del diseño y la funcionalidad de la aplicación con el objetivo de mejorar la misma.

La encuesta se ha realizado mediante un formulario de Google Drive

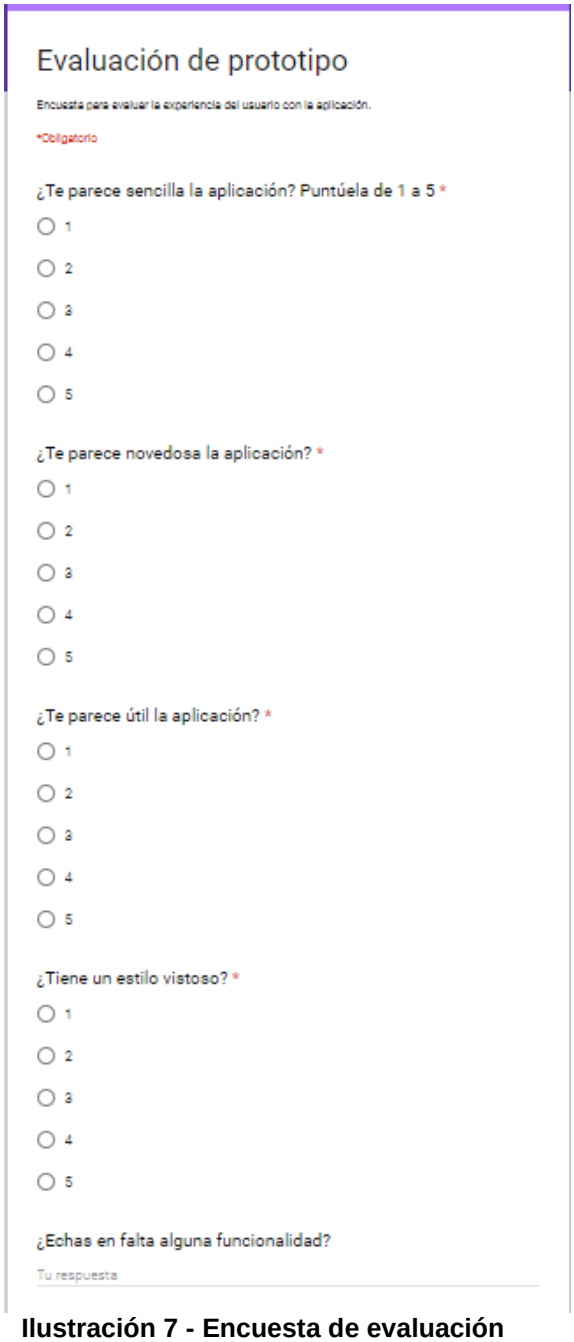


Ilustración 7 - Encuesta de evaluación

2.5. Definición de los casos de uso

El diagrama de casos de uso para el usuario es el siguiente

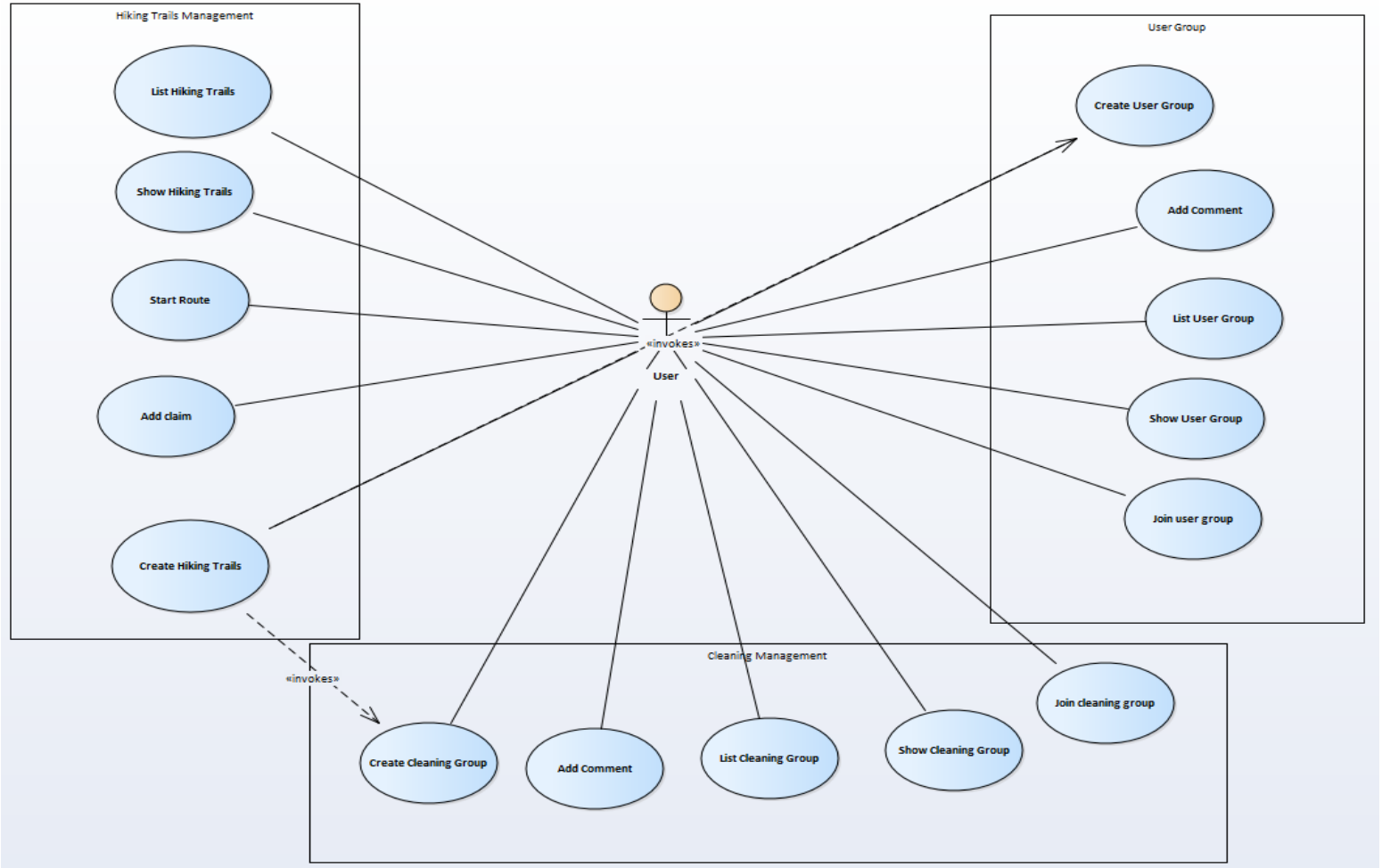


Ilustración 8 - Diagrama de casos de uso de usuario

Para el caso del administrador, el diagrama de casos de uso es:

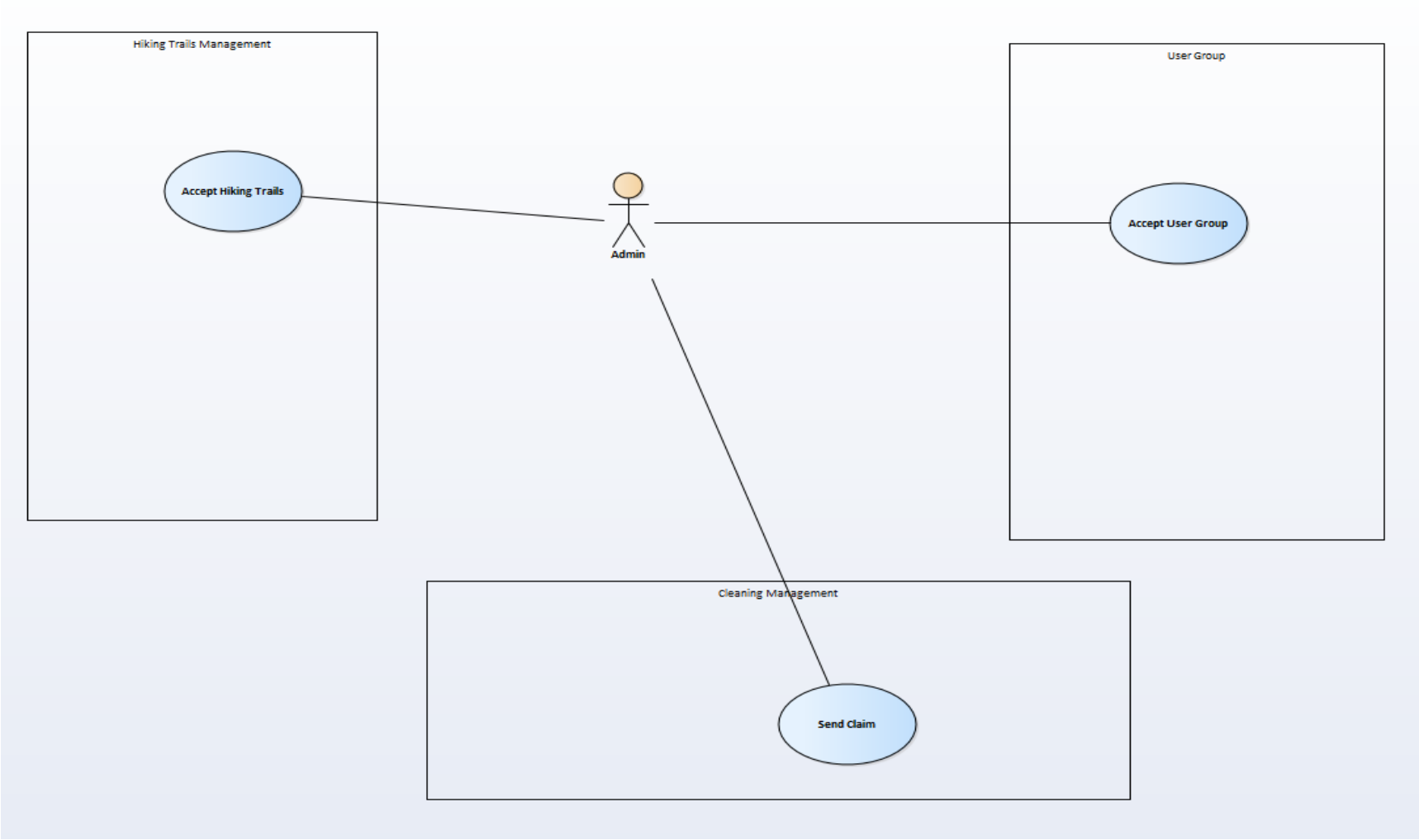


Ilustración 9 - Diagrama de casos de uso del administrador

Finalmente, para cualquier usuario externo a la aplicación, el diagrama de casos de uso es:

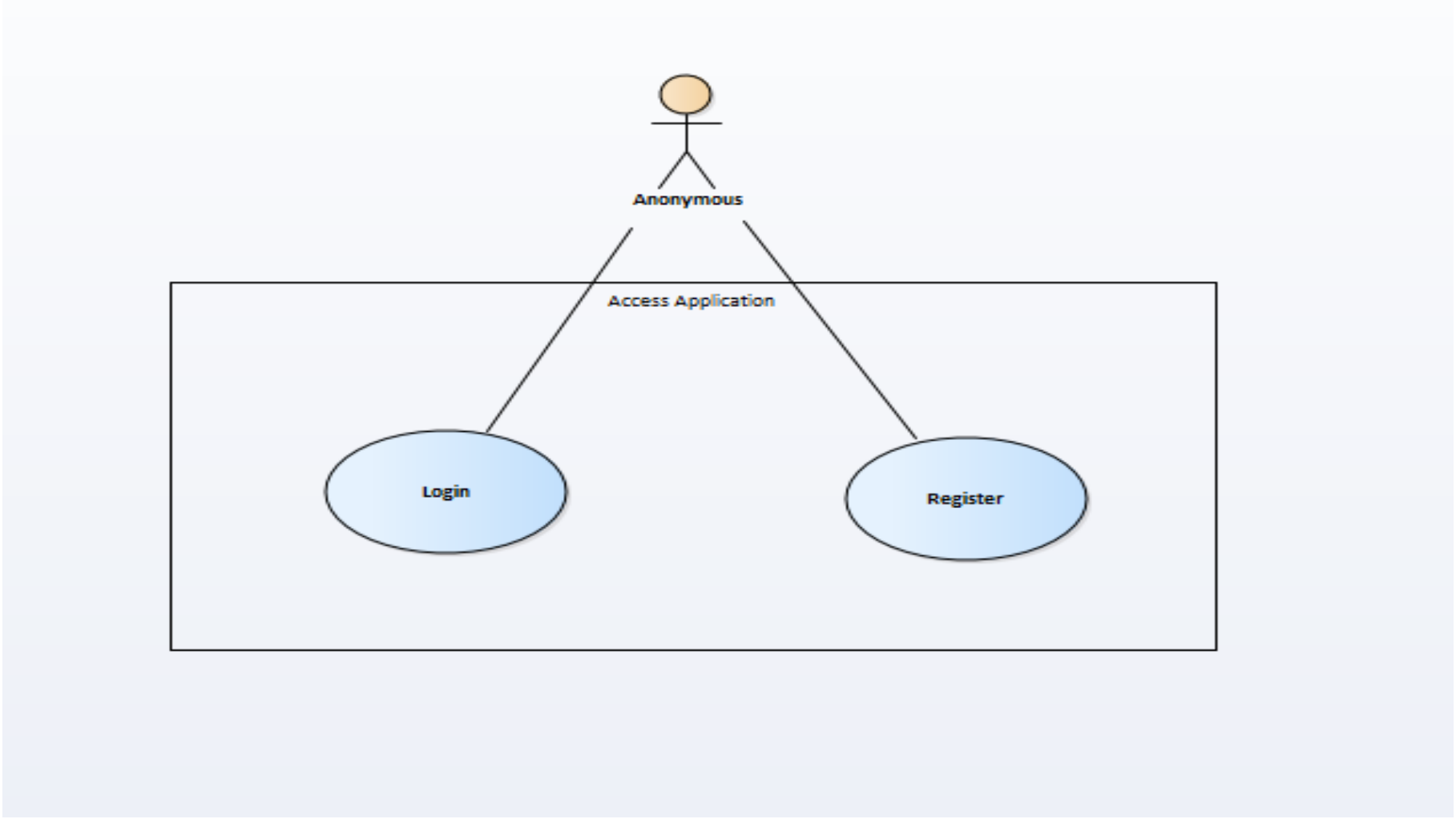


Ilustración 10 - Diagrama de casos de uso de usuarios externos

A continuación se detalla los distintos casos de usos planteados anteriormente mediante tablas:

CU1	Login
Precondición	El usuario no debe estar logueado en la aplicación
Pasos	
1	El usuario selecciona la opción de login
2	El sistema presenta el formulario para introducir el usuario y la contraseña
3	El usuario introduce los datos en el formulario
4	Si los datos son correctos, el sistema presenta la página principal del usuario según el tipo de rol que tenga.
5	Si los datos son incorrectos, el sistema avisa de los errores al usuario y vuelve al paso 3
Postcondición	El usuario está logado en la aplicación.

CU2	Register
Precondición	El usuario no debe estar logueado en la aplicación ni tener cuenta en el sistema
Pasos	
1	El usuario selecciona la opción de registro
2	El sistema presenta el formulario para introducir el usuario, la contraseña y el email
3	El usuario introduce los datos en el formulario
4	Si los datos son correctos, el sistema presenta la página de perfil del usuario.
5	Si los datos son incorrectos, el sistema avisa de los errores al usuario y vuelve al paso 3
Postcondición	El usuario está logado en la aplicación y registrado en el sistema

CU3	Accept Hiking Trails
Precondición	El usuario debe estar logado como administrador
Pasos	
1	El usuario selecciona la opción de Manage Hiking Trails
2	El sistema presenta el listado de rutas existentes en la aplicación
3	El usuario acepta o bloquea las rutas solicitadas
4	Si el usuario acepta una ruta, el sistema habilita dicha ruta para que los usuarios puedan visualizarla
5	Si el usuario bloquea una ruta, el sistema borra dicha ruta.
Postcondición	Las rutas quedan habilitadas o borradas

CU4	Accept User Group
Precondición	El usuario debe estar logado como administrador
Pasos	
1	El usuario selecciona la opción de User List
2	El sistema presenta el listado de grupos existentes en la aplicación
3	El usuario acepta o bloquea los grupos solicitados
4	Si el usuario acepta un grupo, el sistema habilita dicho grupo para que los usuarios puedan hablar en el mismo
5	Si el usuario bloquea un grupo, el sistema inhabilita dicho grupo.
Postcondición	Los grupos quedan habilitados o inhabilitados

CU5	Send Claim
Precondición	El usuario debe estar logado como administrador
Pasos	
1	El usuario selecciona la opción de Cleaning Claims
2	El sistema presenta el listado de reclamaciones existentes en la aplicación
3	El usuario envía una reclamación de una ruta
4	El sistema toma la plantilla del correo para enviarla a la entidad correspondiente de la provincia
Postcondición	Se envía un correo a la provincia de dicha ruta.

CU6	List Hiking Trails
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Trails
2	El sistema presenta el listado de rutas disponibles en la aplicación.
Postcondición	Se visualizan todas las listas disponibles en la aplicación

CU7	Show Hiking Trails
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Trails
2	El sistema presenta el listado de rutas disponibles en la aplicación.
3	El usuario selecciona la ruta que desea visualizar
4	El sistema presenta los datos de la ruta seleccionada
Postcondición	Se visualizan los datos de la ruta seleccionada

CU8	Create Hiking Trails
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Trails
2	El sistema presenta el listado de rutas disponibles en la aplicación.
3	El usuario selección la opción de añadir
4	El sistema presenta el formulario para crear la ruta de senderismo
5	El usuario completa todos los datos necesarios para la creación de la ruta
6	El sistema chequea que la información es correcta
7	Si la información es correcta, la ruta pasa a disposición del administrador de la aplicación para su aceptación o rechazo
8	Si la información es incorrecta, el sistema indica que campos son los erróneos y volvemos al paso 5
Postcondición	La ruta es añadida al listado de rutas pendientes de aceptar o rechazar.

CU9	List User Group
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Group
2	El sistema presenta el listado de grupos de usuario disponibles en la aplicación.
Postcondición	Se visualizan todas los grupos disponibles en la aplicación

CU10	Show User Group
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Group
2	El sistema presenta el listado de grupos de usuario disponibles en la aplicación.
3	El usuario selecciona el grupo que desea visualizar
4	El sistema presenta los datos del grupo seleccionado
Postcondición	Se visualiza los datos del grupo de usuario seleccionado

CU11	Create User Group
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Trails
2	El sistema presenta el listado de rutas disponibles en la aplicación
3	El usuario selección una ruta
4	El sistema presenta la información de la ruta seleccionada
5	El usuario selecciona el icono de los usuarios
6	El sistema carga el formulario de creación de User Group
7	El usuario rellena toda la información del formulario
8	Si la información es correcta, el grupo pasa a disposición del administrador
9	Si la información es incorrecta, el sistema muestra que campos del formularios son erróneos y se vuelve al paso 7

CU12	Add Comment User Group
Precondición	El usuario debe estar logado como usuario y debe estar unido al grupo
Pasos	
1	El usuario selecciona la opción Hiking Group
2	El sistema presenta el listado de grupos de usuario disponibles en la aplicación.
3	El usuario selecciona el grupo que desea visualizar
4	El sistema presenta los datos del grupo seleccionado con el formulario de comentario
5	El usuario introduce el mensaje que quiere publicar en el grupo
6	El sistema añade el comentario al grupo.
Postcondición	Se añade el comentario a los comentarios del grupo.

CU13	List Cleaning Group
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Clean
2	El sistema presenta el listado de grupos de limpieza disponibles en la aplicación.
Postcondición	Se visualizan todas los grupos de limpieza disponibles en la aplicación

CU14	Show Cleaning Group
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Clean
2	El sistema presenta el listado de grupos de limpieza disponibles en la aplicación.
3	El usuario selecciona el grupo que desea visualizar
4	El sistema presenta los datos del grupo seleccionado
Postcondición	Se visualiza los datos del grupo de limpieza seleccionado

CU15	Create Cleaning Group
Precondición	El usuario debe estar logado como usuario
Pasos	
1	El usuario selecciona la opción Hiking Trails
2	El sistema presenta el listado de rutas disponibles en la aplicación
3	El usuario selección una ruta
4	El sistema presenta la información de la ruta seleccionada
5	El usuario selecciona el icono de la papelera
6	El sistema carga el formulario de creación de Cleaning Group
7	El usuario rellena toda la información del formulario
8	Si la información es correcta, el grupo pasa a disposición del administrador
9	Si la información es incorrecta, el sistema muestra que campos del formularios son erróneos y se vuelve al paso 7

CU16	Add Comment Cleaning Group
Precondición	El usuario debe estar logado como usuario y debe estar unido al grupo
Pasos	
1	El usuario selecciona la opción Clean
2	El sistema presenta el listado de grupos de limpieza disponibles en la aplicación.
3	El usuario selecciona el grupo que desea visualizar
4	El sistema presenta los datos del grupo seleccionado con el formulario de comentario
5	El usuario introduce el mensaje que quiere publicar en el grupo
6	El sistema añade el comentario al grupo.
Postcondición	Se añade el comentario a los comentarios del grupo.

CU17	Join user group
Precondición	El usuario debe estar logado como usuario y no debe estar unido al grupo
Pasos	
1	El usuario selecciona la opción Hiking Group
2	El sistema presenta el listado de grupos de usuarios disponibles en la aplicación.
3	El usuario selecciona el icono de añadir al grupo que desea
4	El sistema lo añade al grupo
Postcondición	El usuario se une al grupo

CU18	Join cleaning group
Precondición	El usuario debe estar logado como usuario y no debe estar unido al grupo
Pasos	
1	El usuario selecciona la opción Clean
2	El sistema presenta el listado de grupos de limpieza disponibles en la aplicación.
3	El usuario selecciona el icono de añadir al grupo que desea
4	El sistema lo añade al grupo
Postcondición	El usuario se une al grupo

CU19	Start route
Precondición	El usuario debe estar logado como usuario.
Pasos	
1	El usuario selecciona la opción de visualizar una ruta
2	El sistema presenta los datos y funciones que se puede realizar en la vista
3	El usuario selecciona el icono de Start Route
4	El sistema comienza un contador de los kilómetros recorridos
5	El usuario detiene el contador
6	El sistema añade los kilómetros recorridos
Postcondición	El usuario añade más kilómetros recorridos en su perfil.

CU20	Add claim
Precondición	El usuario debe estar logado como usuario.
Pasos	
1	El usuario selecciona la opción de visualizar una ruta
2	El sistema presenta los datos y funciones que se puede realizar en la vista
3	El usuario selecciona el icono de Claim
4	El sistema aumenta el contador de reclamaciones y en caso de llegar a 100 reclamaciones, se genera un mensaje en la administración.
Postcondición	Se aumenta el contador de reclamaciones de una ruta

2.6. Diseño de la arquitectura

El modelo de entidades para el dominio del problema planteado es el siguiente:

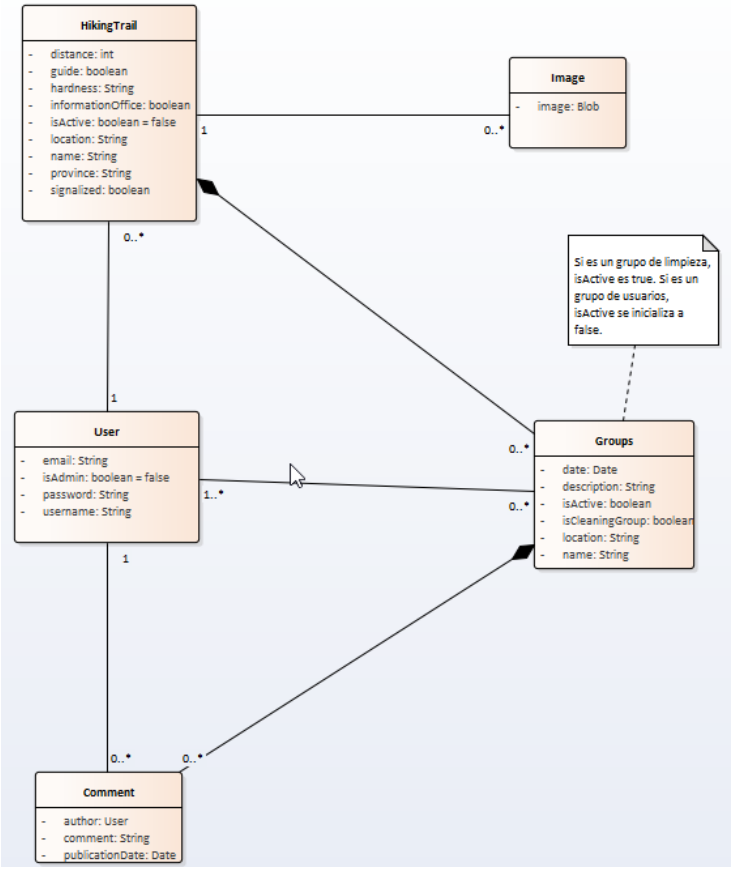


Ilustración 11 - Modelo de entidades

El modelo de base de datos define el mapeo que tiene el modelo de entidades en base de datos.

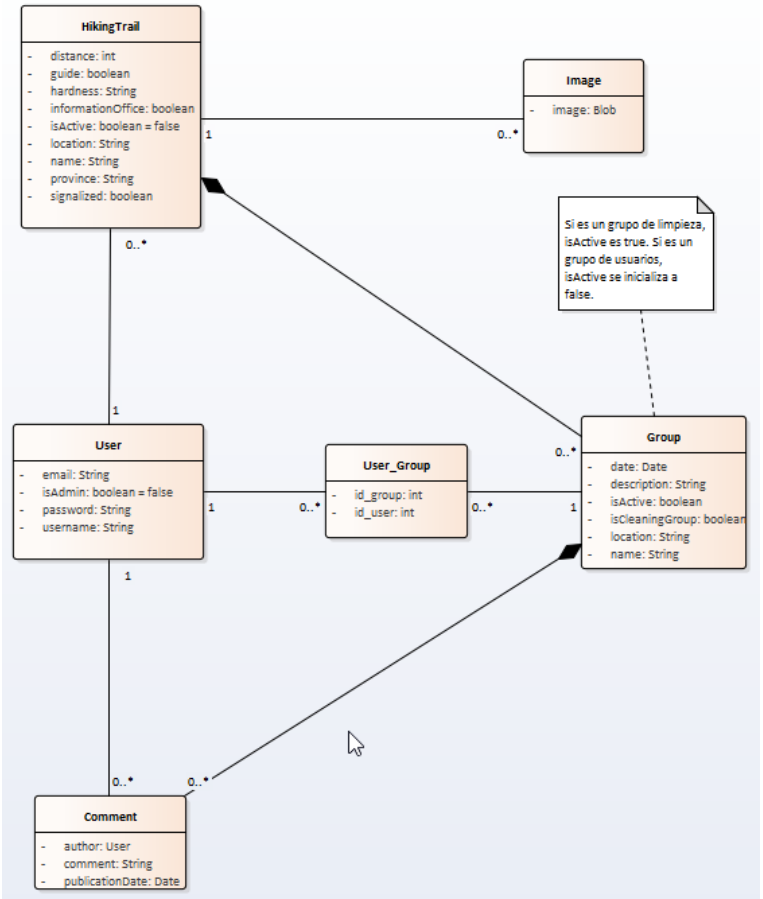


Ilustración 12 - Modelo de base de datos

Para el diseño de la arquitectura de la aplicación se va a hacer uso de la arquitectura cliente servidor. Básicamente, la arquitectura cliente-servidor es un modelo de diseño donde existen dos roles: los proveedores de recursos o servicios y los clientes. Estos clientes solicitan los recursos que necesitan a estos proveedores los cuales les da las respuestas que solicitan.

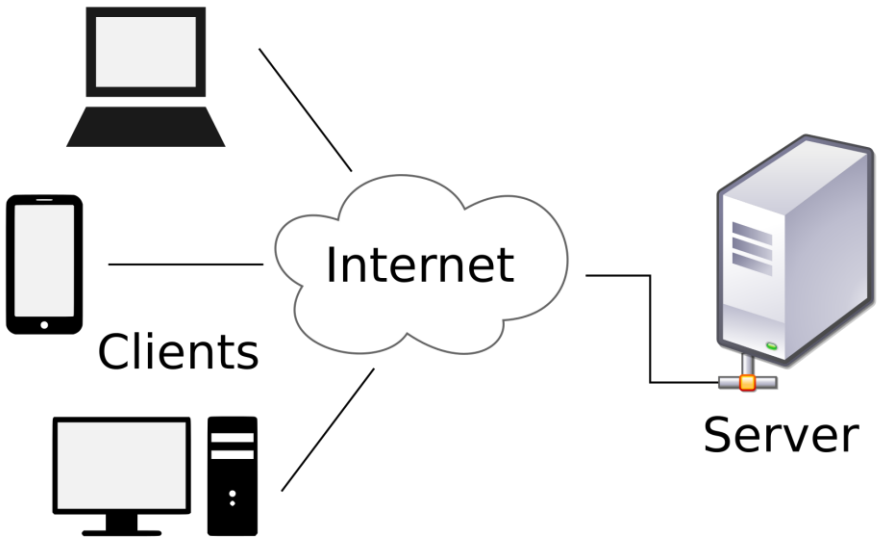


Ilustración 13 - Arquitectura cliente-servidor

Para la parte cliente, se usará Android ya que nuestra aplicación es para dispositivos móviles. Para la parte servidor, se va a desarrollar en ExpressJS con base de datos Mongo DB tal y como se ha definido en los apartados anteriores.

Se ha optado por esta opción porque la idea principal es hacer que esta aplicación sea únicamente para dispositivos móviles y por lo tanto necesitamos de un servicio que nos provea de los datos necesarios que no se pueden centralizar en una aplicación móvil.

3. Desarrollo

3.1. Backend

Para realizar la parte de Backend de la aplicación, se ha desarrollado una API en el lenguaje ExpressJS junto con la base de datos MongoDB, usando el módulo Mongoose. Para construir la estructura mínima se ha seguido el manual de ejemplo [4].

3.1.1. Api Routes

En el archivo api-routes.js se indica las diferentes rutas que son accesibles vía web mediante la API. En este archivo se indican cada una de las acciones que se pueden llevar a cabo en la aplicación móvil. A continuación, se muestra un ejemplo de cómo realizar un enrutamiento:

```
// Import controllers
var userController = require('./user/userController');
// User routes
router.route('/users')
    .get(userController.index)
    .post(userController.new);
router.route('/users/:user_id')
    .get(userController.view)
    .patch(userController.update)
    .put(userController.update)
    .delete(userController.delete);
router.route('/authentication')
    .post(userController.authentication);
router.route('/activeUsers')
    .get(userController.activeUsers);
```

Ilustración 14 - API Routes

3.1.2. Modelo en el servidor

Para poder hacer una correspondencia entre los objetos de la base de datos con los objetos de la aplicación móvil, es necesario implementar los mismos objetos que se quieren almacenar en la base de datos. Un ejemplo de ello, es por ejemplo los usuarios de la aplicación. En ExpressJS se implementa de la siguiente manera:

```
// UserModel.js
var mongoose = require('mongoose');
Schema = mongoose.Schema;
// Setup schema
var userSchema = mongoose.Schema({
    name: {type: String...},
    email: {type: String...},
    password: {type: String...},
    isAdmin: {type: Boolean...},
    isActive: {type: Boolean...},
    hikingTrails:
    [
        {
            type: Schema.Types.ObjectId,
            ref: 'HikingTrail'
        }
    ],
    comments:
        [...],
    groups:
    [...]

});
// Export User model
var User = module.exports = mongoose.model( name: 'User', userSchema);
module.exports.get = function (callback, limit) {
    User.find(callback).limit(limit);
};
```

Ilustración 15 - Modelo ExpressJS

Como se puede ver en la imagen, se implementa cada uno de sus atributos y se corresponde con los de la base de datos y los modelos de la aplicación móvil.

3.1.3. Controlador en el servidor

Para indicar que operaciones hay que realizar cuando se consulta alguna de las acciones definidas en API Routes, hay que realizar un controlador por cada uno de los modelos que se han implementado. En este controlador se indicaran las acciones de guardado, actualización, visualización, etc. Un ejemplo de ello es el siguiente:

```
exports.authentication = function(req, res) {
    var name = req.body.name ? req.body.name : "";
    var password = req.body.password ? req.body.password : "";
    User.findOne({'name': name, 'password': password}).exec( op: function(err, user){
        if (err)
            return handleError(err);

        if(user != null) {
            res.json({
                message: 'User Logged',
                data: user
            });
        }
        else
        {
            res.json({
                message: 'User not found'
            });
        }
    });
};
```

Ilustración 16 - Controlador del servidor

Como se puede ver en la imagen, en este caso se consulta en la base de datos si existe un usuario con una contraseña igual a las suministradas poder entrar en la aplicación.

3.2. Aplicación Android

3.2.1. Modelos

Estos modelos se usan exactamente igual a los del servidor, como correspondencia entre los objetos de base de datos y los objetos que manejamos en la aplicación. Estos modelos tienen sus atributos y sus métodos GET y SET para su manejo. Cabe destacar que cada uno de estos modelos tiene un convertidor entre objetos JSON y el modelo en cuestión. Un ejemplo de ello es el siguiente:

```java
// Método para parsear un objeto recibido en una petición.
public void parseFromJSON(JSONObject jsonObject) {
    try {
        this.id = jsonObject.getString( name: "_id");
        this.name = jsonObject.getString( name: "name");
        this.description = jsonObject.getString( name: "description");
        this.location = jsonObject.getString( name: "location");
        SimpleDateFormat formatter=new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");
        this.date = formatter.parse(jsonObject.getString( name: "date"));
        this.isActive = Boolean.parseBoolean(jsonObject.getString( name: "isActive"));
        this.isCleaningGroup = Boolean.parseBoolean(jsonObject.getString( name: "isCleaningGroup"));
        if(jsonObject.has( name: "hikingTrail")) {
            HikingTrail hikingTrail = new HikingTrail();
            hikingTrail.parseFromJSON(jsonObject.getJSONObject("hikingTrail"));
            this.hikingTrail = hikingTrail;
        }

        if(jsonObject.has( name: "users")){
            JSONArray usersJSON = jsonObject.getJSONArray( name: "users");
            int userSize = usersJSON.length();
            for(int i =0; i < userSize; i++) {
                User user = new User();
                user.parseFromJSON(usersJSON.getJSONObject(i));
                this.users.add(user);
            }
        }

        if(jsonObject.has( name: "comments")) {
            JSONArray commentsJSON = jsonObject.getJSONArray( name: "comments");
            int commentSize = commentsJSON.length();
            for (int i = 0; i < commentSize; i++) {
                Comment comment = new Comment();
                comment.parseFromJSON(commentsJSON.getJSONObject(i));
                this.comments.add(comment);
            }
        }

    } catch (JSONException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    }

}
```

Ilustración 17 - Convertidor de JSON a Objeto

3.2.2. Peticiones al servidor

Para la realización de las peticiones al servidor, se va a utilizar la librería de AsyncHttpClient. Se ha implementado una clase auxiliar para la gestión de estas peticiones así como para configurar la URL de la API del servidor. La clase en cuestión implementa los métodos GET, POST y PUT que son las utilizadas para esta aplicación. Actualmente, la URL apunta a una IP pública por lo que con la APK proporcionada podremos probar la aplicación. En caso de que se quiera desplegar en otro servidor, es necesario cambiar este fichero para configurar la URL donde esté alojada la aplicación servidor y volver a generar la APK.

Esta clase presenta las siguientes funciones:

```
public class AsyncHttpUtils {
    private static final String BASE_URL = "http://192.168.1.134:8080/api/";

    private static AsyncHttpClient client = new AsyncHttpClient();

    // Método para realizar una petición get al servidor
    public static void get(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {
        client.setMaxRetriesAndTimeout( retries: 1,  timeout: 600);
        client.get(getAbsoluteUrl(url), params, responseHandler);
    }

    // Método para realizar una petición post al servidor
    public static void post(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {
        client.setMaxRetriesAndTimeout( retries: 1,  timeout: 600);
        client.post(getAbsoluteUrl(url), params, responseHandler);
    }

    // Método para realizar una petición put al servidor
    public static void put(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {
        client.setMaxRetriesAndTimeout( retries: 1,  timeout: 600);
        client.put(getAbsoluteUrl(url), params, responseHandler);
    }

    // Método para obtener la url completa a la API del servidor
    private static String getAbsoluteUrl(String relativeUrl) { return BASE_URL + relativeUrl; }

}
```

Ilustración 18 – AsyncHttpUtils

3.2.3. Actividades y Array Adapter

En ese apartado describiremos muy poco puesto que ya es bien conocido el funcionamiento de este componente en una aplicación móvil. En estas actividades se define el comportamiento de nuestra aplicación y en caso de manejar las listas de nuestros modelos comentados anteriormente, utilizamos los Array Adapters donde gestionamos su visibilidad y comportamiento.

3.2.4. Vistas

Para el aspecto de nuestra aplicación, resaltaremos algunos de los casos más concretos de las vistas que hemos implementado. En primer lugar, las listas de nuestros objetos se han desarrollado con una vista independiente que se insertan en un Recycler View. Dentro de esta vista, cada objeto va gestionado dentro de un Card View donde se encuentran los componentes que mostrarán los atributos que se crean conveniente para cada caso. Un ejemplo de ello es la siguiente vista:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginBottom="5dp">

    <android.support.v7.widget.CardView xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:id="@+id/card_view"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        card_view:cardCornerRadius="4dp"
        card_view:layout_constraintEnd_toEndOf="parent"
        card_view:layout_constraintStart_toStartOf="parent">

        <TextView
            android:id="@+id/textUsername"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <ImageButton
            android:id="@+id/acceptButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="20dp"
            android:layout_gravity="right|center"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_confirm" />

        <ImageButton
            android:id="@+id/blockButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="20dp"
            android:layout_gravity="right|center"
            android:background="@android:color/transparent"
            android:src="@drawable/ic_block" />

    </android.support.v7.widget.CardView>

</LinearLayout>
```

Ilustración 19 - Vista de la aplicación móvil

3.2.5. Google Maps

Un último aspecto a tener en cuenta en el desarrollo de la aplicación es el uso de Google Maps para indicar donde está situada la ruta de senderismo.
Para ello, se ha tenido que generar un proyecto en Firebase así como una serie de configuraciones como es la incorporación de las librerías a gradle o la configuración de la API KEY

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Ilustración 20 - Configuración API KEY

Además, hay habilitar las api en la consola de google.

El componente de la vista para el mapa queda de la siguiente forma:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NewHikingTrailActivity">
    <TextView...>
    <EditText...>
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        map:layout_constraintEnd_toEndOf="parent"
        map:layout_constraintStart_toStartOf="parent"
        map:layout_constraintTop_toBottomOf="@+id/nameHikingTrailEditText"
        tools:context=".ShowHikingTrailActivity" />
    <ImageButton...>
    <TextView...>
    <ImageView...>
    <TextView...>
    <Spinner...>
    <TextView...>
    <EditText...>
    <TextView...>
    <EditText...>
    <CheckBox...>
    <CheckBox...>
    <CheckBox...>
    <ImageButton...>
</android.support.constraint.ConstraintLayout>
```

Ilustración 21 - Vista del mapa

Con respecto a la implementación, la clase que use dicho mapa deberá de implementar las siguientes interfaces:

```
public class NewHikingTrailActivity extends AppCompatActivity implements OnMapReadyCallback,
        GoogleMap.OnCameraMoveListener{
```

Ilustración 22 - Implementar métodos para Mapas

Además, se han definido los siguientes métodos:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {
        getDeviceLocation();
        mMap.setMyLocationEnabled(true);
        mMap.getUiSettings().setZoomControlsEnabled(true);
        mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
        mMap.setOnCameraMoveListener(this);
    }

    mMap.setOnMapClickListener((point) -> {
            mMap.clear();
            locationMarker = new MarkerOptions().position(point);
            mMap.addMarker(locationMarker);
    });
}

private void initMap(){

    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
    mapFragment.getMapAsync( onMapReadyCallback: NewHikingTrailActivity.this);
}

private void moveCamera(LatLng latLng, float zoom){
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom));
}

@Override
public void onCameraMove() {
    if(DEFAULT_ZOOM != mMap.getCameraPosition().zoom) {
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putFloat("DEFAULT_ZOOM", mMap.getCameraPosition().zoom);
        editor.commit();

        DEFAULT_ZOOM = sharedPref.getFloat( key: "DEFAULT_ZOOM",  defValue: 12f);
    }
}
```

Ilustración 23 - Método para configurar el Mapa

En estos métodos se define el comportamiento de nuestros mapas y se solicitan los permisos necesarios para ello.

4. Pruebas

4.1. Pruebas unitarias

Para el desarrollo de las pruebas unitarias, se ha implementado una serie de pruebas que verifiquen el correcto funcionamiento de los convertidores de los modelos.
Un ejemplo de ello es el siguiente:

```
public class UserTest {

    private JSONObject userJSON;

    @Before
    public void setUp() throws JSONException {
        userJSON = new JSONObject();
        userJSON.put( name: "_id", value: "1234");
        userJSON.put( name: "name", value: "usuarioTest");
        userJSON.put( name: "password", value: "passwordTest");
        userJSON.put( name: "email", value: "emailTest@test.org");
        userJSON.put( name: "isActive", value: "false");
        userJSON.put( name: "isAdmin", value: "false");
    }

    @Test
    public void parseUserTest(){
        User user = new User();
        user.parseFromJSON(userJSON);

        assertTrue(user.getId().equals("1234"));
        assertTrue(user.getName().equals("usuarioTest"));
        assertTrue(user.getPassword().equals("passwordTest"));
        assertTrue(user.getEmail().equals("emailTest@test.org"));
        assertTrue(user.getIsActive().equals(false));
        assertTrue(user.getAdmin().equals(false));

        assertThat(user.getIsActive(), instanceOf(Boolean.class));
    }
}
```

Ilustración 24 - Test unitario

Como se puede ver en el test, creamos un objeto JSON simulando el comportamiento de respuesta de la aplicación backend y convertimos dicho objeto en uno de los modelos definidos verificando los valores.

Para la realización de las pruebas unitarias, se ha utilizado la librería JUnit

4.2. Pruebas funcionales

Para la realización de las pruebas funcionales se ha hecho uso de la librería Espresso. Se ha realizado pruebas sobre las funcionalidades mínimas de la aplicación.
Un ejemplo de este tipo de aplicación es la siguiente:

```
@Test
public void loginActivityTest2() {
    ViewInteraction appCompatButton = onView(allOf(withId(R.id.buttonLogin), withText("Login"),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 1), isDisplayed()));
    appCompatButton.perform(click());

    ViewInteraction appCompatEditText = onView(allOf(withId(R.id.editUsername),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 2),isDisplayed()));
    appCompatEditText.perform(click());

    ViewInteraction appCompatEditText2 = onView(allOf(withId(R.id.editUsername),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 2),isDisplayed()));
    appCompatEditText2.perform(replaceText( stringToBeSet: "marta"), closeSoftKeyboard());

    ViewInteraction appCompatEditText3 = onView(allOf(withId(R.id.editUsername), withText("marta"),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 2),isDisplayed()));
    appCompatEditText3.perform(pressImeActionButton());

    ViewInteraction appCompatEditText4 = onView(allOf(withId(R.id.editPassword),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 4),isDisplayed()));
    appCompatEditText4.perform(replaceText( stringToBeSet: "marta"), closeSoftKeyboard());

    ViewInteraction appCompatEditText5 = onView(allOf(withId(R.id.editPassword), withText("marta"),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 4),isDisplayed()));
    appCompatEditText5.perform(pressImeActionButton());

    ViewInteraction appCompatImageButton = onView(allOf(withId(R.id.confirm), withContentDescription( text: "Login"),
            childAtPosition(childAtPosition(withId(android.R.id.content), position: 0), position: 5),isDisplayed()));
    appCompatImageButton.perform(click());
}

private static Matcher<View> childAtPosition(
        final Matcher<View> parentMatcher, final int position) {

    return new TypeSafeMatcher<View>() {
        @Override
        public void describeTo(Description description) {
            description.appendText("Child at position " + position + " in parent ");
            parentMatcher.describeTo(description);
        }

        @Override
        public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup && parentMatcher.matches(parent)
                    && view.equals(((ViewGroup) parent).getChildAt(position));
        }
    };
}
```

Ilustración 25 - Prueba funcional Login

Como vemos en la imagen, vamos seleccionando cada uno de los componentes y realizamos la acción correspondiente en cada caso hasta completar la funcionalidad que queremos probar.

Un problema que nos hemos encontrado ha sido la realización de dichas pruebas con los Array Adapter. No detectan los objetos a seleccionar y por lo tanto el test no puede finalizar. Finalmente, se ha encontrado una solución externa [5] donde se implementa una clase capaz de acceder a dicho objeto desde el recycler view.

5. Conclusiones

Tras la realización del trabajo fin de máster me he podido enfrentar a muchos retos y tecnologías nuevas además de consolidar las que ya conocía y ponerlas en práctica en la realización de una aplicación.

Con este trabajo me he dado cuenta lo importante que es planificar para organizar y proponer las metas a lo largo del proyecto; y estudiar el mercado para conocer el aporte de valor que podemos ofrecer con nuestra aplicación. Hoy en día es complicado encontrar una aplicación que no exista pero si es posible encontrar una especialización de una aplicación en concreto y por ello es importante hacer este estudio previo al desarrollo. Por otro lado, es importante conocer las tecnologías actuales y que más convenga al entorno donde se vaya a implantar y a los clientes que vaya dirigido. Por todo ello son necesarios estos estudios y evaluaciones previos.

En cuanto a realizar un buen diseño previo de la aplicación facilita mucho el desarrollo de la aplicación así como la considerable aceleración de los tiempos de desarrollo. He aprendido que la inversión del tiempo en un buen diseño ahorra mucho tiempo en el desarrollo a posteriori y por lo tanto, los tiempos se reducen y los objetivos se quedan mucho más claros.

Puesto que ya he contado con una experiencia previa en el desarrollo mediante un ciclo de vida iterativo, me ha sido mucho más fácil trabajar de esta manera y viendo cómo se planteaba la aplicación, me ha servido para ir evolucionándola conforme ha pasado las iteraciones alcanzando un producto con el que estaba satisfecho aunque con mucho recorrido de mejora.

Gracias al conocimiento de esta metodología, he podido realizar una buena planificación que he podido seguir sin demasiada dificultad. Los objetivos planteados se han alcanzado en los tiempos que me había propuesto.

Finalmente comentar las líneas de trabajo futura que han quedado planteadas en este trabajo pero que finalmente no se ha podido llevar a cabo por falta de tiempo como son la subida de imágenes a la aplicación, la contabilización de los kilómetros de los usuarios en las rutas de senderismo, dibujar la ruta con Google Maps para que los usuarios conozcan el recorrido de dicha ruta así confirmar la asistencia y llegada en los grupos de senderismo. Como se puede ver, la aplicación contiene un producto mínimo viable aunque tiene aún mucho recorrido por desarrollar.

En definitiva, creo que ha sido una buena experiencia para aprender tecnologías nuevas y enfrentarme a una serie de retos nuevos como es el desarrollo de aplicaciones móviles. Espero poder seguir desarrollando

esta aplicación completamente dado que ha sido un proyecto que me ha motivado y con el que he aprendido mucho.

6. Glosario

- Desarrollo Iterativo: Este modelo de desarrollo, que no es más que un conjunto de tareas agrupadas en pequeñas etapas repetitivas.
- Controlador: Componente encargado de gestionar la transferencia de información entre el usuario y el sistema.
- Modelo: Definición de un objeto en la programación orientada a objetos donde se presenta su esqueleto y las funcionalidades que puede realizar.
- API: Es una interfaz de programación de aplicaciones donde se provee acceso a funciones de un determinado software.
- Actividad Android: Son cada una de las pantallas de una aplicación Android donde se gestiona tanto su visualización como su comportamiento.
- Prototipo: Es un artefacto que permite la simulación de las pantallas y su navegabilidad de una aplicación.

7. Bibliografía

1. http://www.impulsodigital.elmundo.es/aprende/5-apps-imprescindibles-para-hacer-senderismo, 02/03/2019
2. https://www.columbus-outdoor.com/blog/mejores-aplicaciones-para-senderismo-trekking/, 02/03/2019
3. https://proyectosagiles.org/desarrollo-iterativo-incremental/, 25/03/2019
4. https://medium.com/@dinyangetoh/how-to-build-simple-restful-api-with-nodejs-expressjs-and-mongodb-99348012925d 28/03/2019
5. https://github.com/dannyroa/espresso-samples/blob/master/RecyclerView/app/src/androidTest/java/com/dannyroa/espresso_samples/recyclerview/RecyclerViewMatcher.java 15/05/2019
6. https://github.com/mohammadatif/Animatoo para la realización de las animaciones. Ultimo acceso 01/06/2019

8. Anexos

8.1. Manual de usuario

En el siguiente apartado vamos a exponer el manual de usuario para el uso de la aplicación. Esta se dividirá según los interesados en el uso de la aplicación que han sido identificados como: Administrador, Usuario y Administrador de Sistemas.

Administrador de sistemas

Despliegue

En este apartado definiremos como se implanta el sistema así como hacer las configuraciones necesarias para el buen funcionamiento de la aplicación. Los pasos a seguir son los siguientes:

Aplicación del servidor:

1. Tener las aplicaciones instaladas
    a. Node
    b. Npm
    c. MongoDB
2. Descomprimir la aplicación en el lugar donde se desee guardar.
3. Dentro de la aplicación, ejecutar el comando "npm install" para instalar todos los módulos necesarios para nuestra aplicación.
4. Para ejecutar el servidor, en el directorio de nuestra aplicación ejecutamos el comando "node index". Esto desplegará nuestra aplicación en el puerto 8080 por lo que deberemos tener libre dicho puerto.

Aplicación Móvil:

1. Antes que nada hay que desplegar el servidor.
2. Cambiar la URL del archivo AsyncHttpUtils por la URL del servidor desplegado
3. Generar APK
4. Subir al Store

Crear usuario administrador

Otro punto importante es la creación de un usuario administrador. Dado que la aplicación es accesible por cualquier persona, para poder crear un usuario administrador, hay que realizar los siguientes pasos.

1. Registramos un usuario con la aplicación móvil mediante el formulario ordinario de registro.
2. Accedemos al servidor donde tenemos la base de datos y abrimos dicha base de datos.

3. Si el servidor es Windows, simplemente con el cliente accedemos al documento de usuario y modificamos el atributo isAdmin del usuario registrado anteriormente tal y como se muestra en la siguiente imagen.

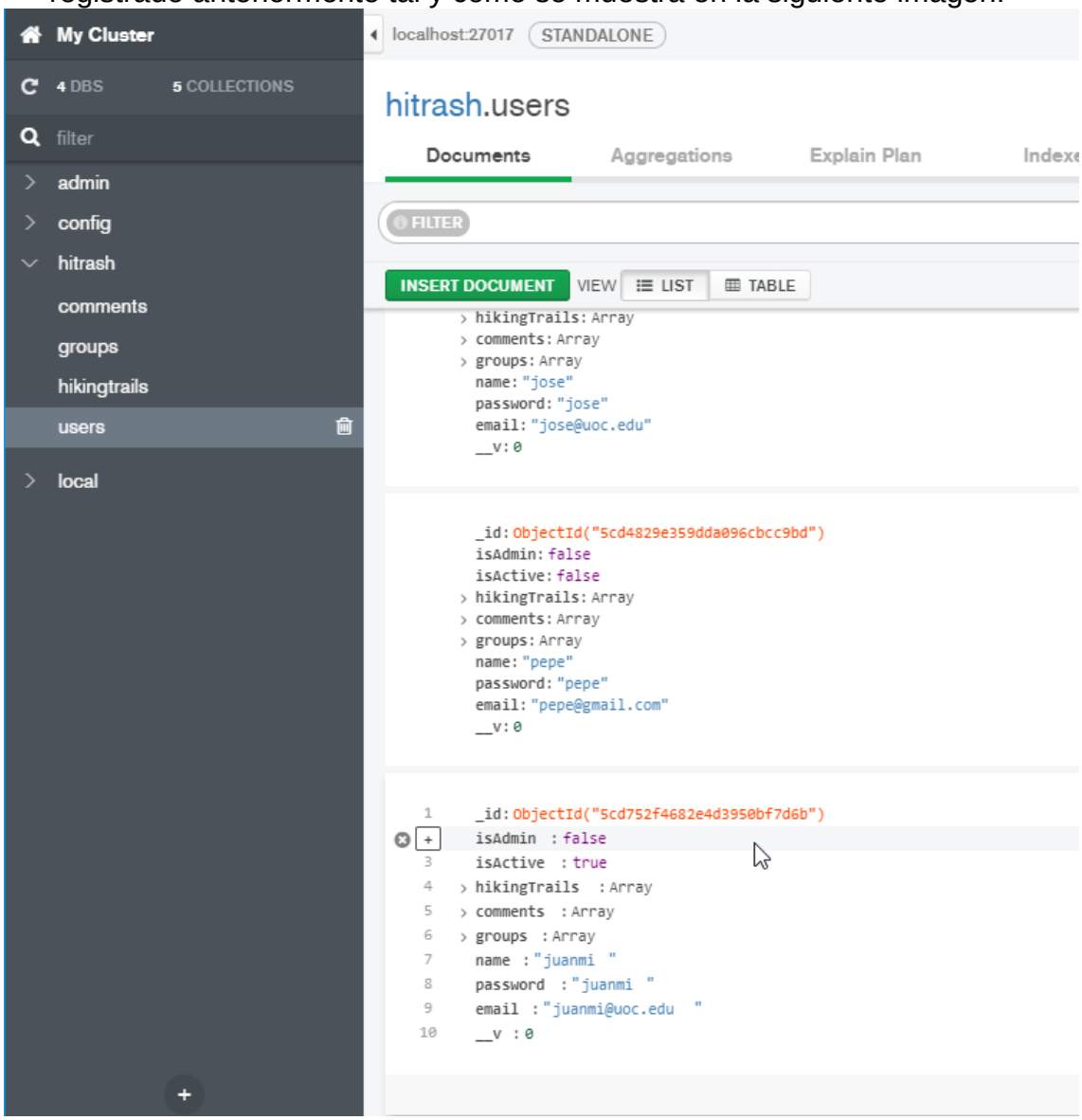


Ilustración 26 - Edición del campo Admin en Windows

En caso de que sea Ubuntu ejecutamos los siguientes comandos para la actualización de los datos para seleccionar el documento y actualizar el dato isAdmin del usuario con el nombre “admin” o en nuestro caso, el usuario que hayamos creado con la aplicación móvil.

```
> use hitrash
switched to db hitrash
> db.users.updateOne( {"name": "admin"},{ $set: {"isAdmin": true}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Administrador

A continuación se presenta cada una de las funciones a las que el administrador tiene acceso.
En primer lugar, una vez se accede a la aplicación, se presenta el siguiente menú

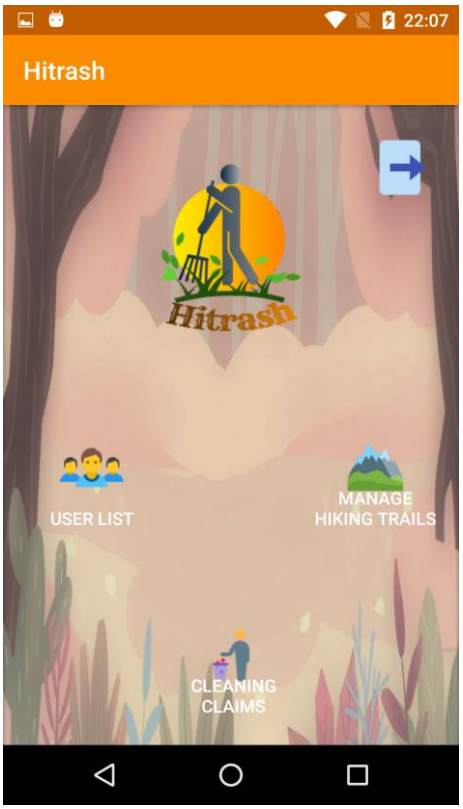


Ilustración 27 - Menú principal Admin

Este rol puede admitir los usuarios registrados para que puedan logarse en la aplicación, aceptar las rutas añadidas por los usuarios y enviar las reclamaciones a las organizaciones para su limpieza.

A continuación mostramos cada una de estas funciones:

- User list: Podemos pulsar sobre el botón de aceptar o bloquear al usuario.

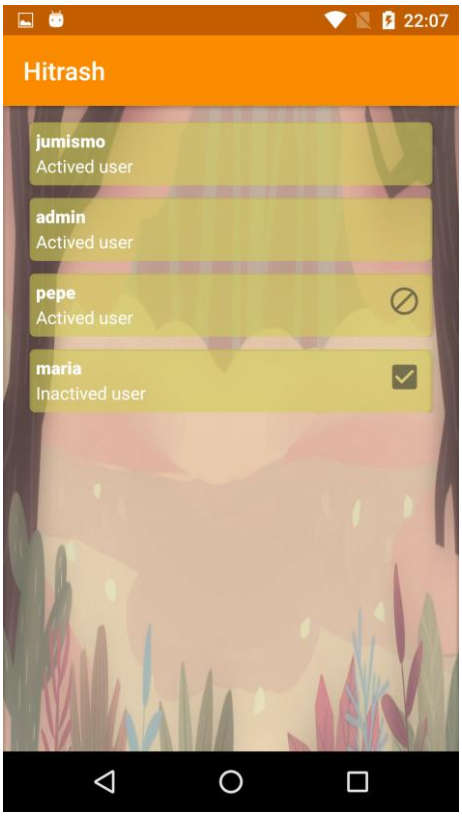


Ilustración 28 - User List Admin

- Manage Hiking Trails: En esta vista podemos aceptar o bloquear una ruta como en la pantalla anterior. Con esto conseguiremos que la ruta sea visible para todos los usuarios.

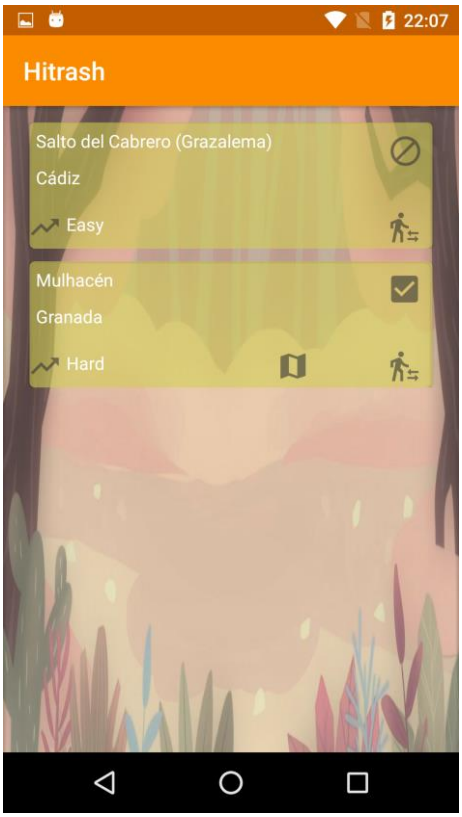


Ilustración 29 - Manage Hiking Trails Admin

- Cleaning Trails: En esta vista podremos enviar un correo electrónico a la organización pertinente para poner en marcha un limpiado de la ruta por parte de dicha organización.

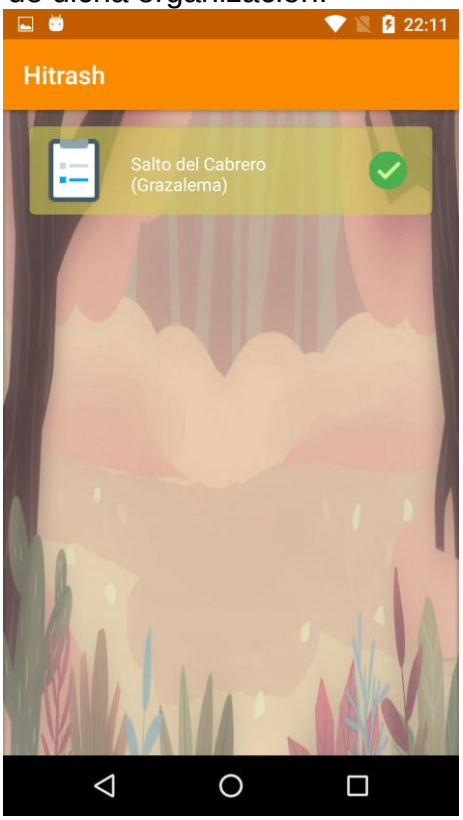


Ilustración 30 - Cleaning Trails Admin

Todas estas son las funcionalidades que se pueden hacer con el usuario administrador.

Usuario

Por último, vamos a explicar las distintas funcionalidades que se puede realizar con un usuario, que será nuestro interesado más común en la aplicación. Como en el rol anterior, presentamos el menú principal del que dispone un usuario:

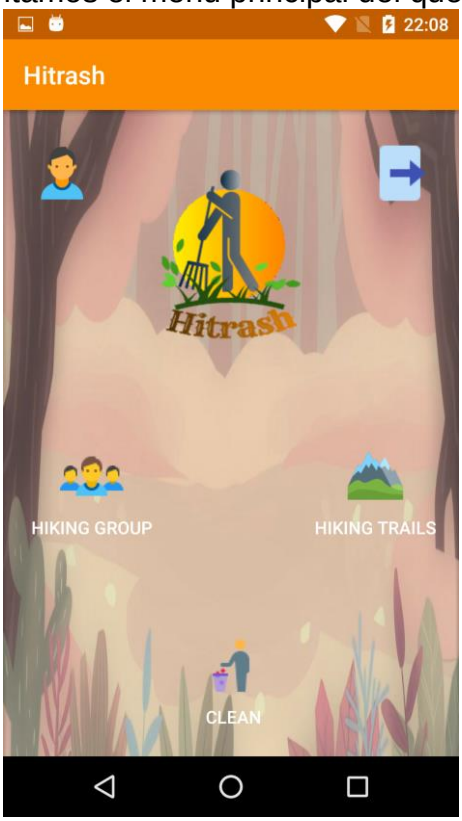


Ilustración 31 - Menú User

Con este rol, podemos unirnos a grupos de limpieza o para hacer senderismo donde podremos conversar con la gente que esté unida a dicho grupo añadiendo comentarios.

Por otro lado, podemos visualizar las rutas de senderismo añadidas por los usuarios o crear las nuestras propias así como visualizar su información y crear grupos tanto de senderismo como de limpieza.

A continuación expondremos cada una de estas funcionalidades:

- Hiking and Clean Group: En esta pantalla podemos visualizar los grupos existentes y podremos unirnos a aquellos que queramos pulsando sobre el botón de unirse.

Ilustración 32 - Hiking and Clean Group

- Mostrar y añadir comentarios: Los usuarios podrán añadir comentarios a los grupos en los que estén insertando el comentario en el campo de texto del final de la pantalla como se muestra a continuación

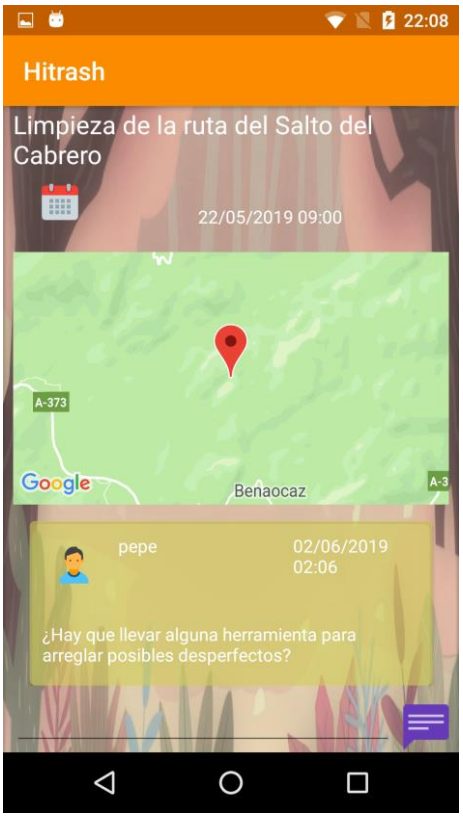


Ilustración 33 - Mostrar y añadir comentarios

- Hiking Trail List: En esta pantalla podremos visualizar todas las rutas de la aplicación.

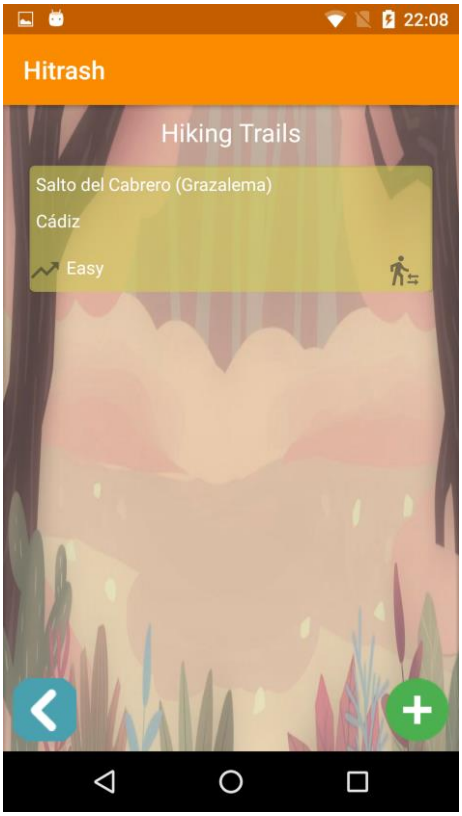


Ilustración 34 - Hiking Trail List

Esta vista ofrece un resumen de las rutas de senderismos para los casos en los que se quiera obtener alguna información rápida.

- Show Hiking Trail: Pulsando sobre la ruta deseada podemos ver el detalle de ésta además de poder crear grupos tanto de limpieza como de senderismo. Por otro lado, podemos añadir una reclamación para que se proponga la limpieza a la organización pertinente pulsando sobre el icono de reclamación.

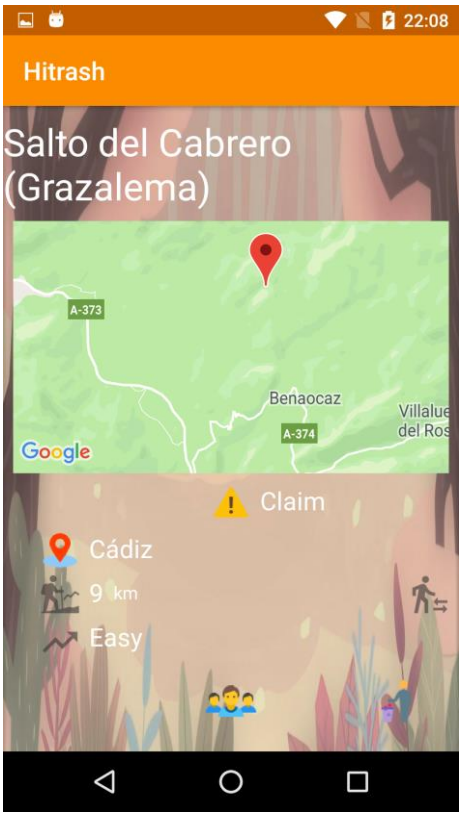


Ilustración 35 - Mostrar Hiking Trail

- Create Group: El usuario podrá crear un grupo para una ruta ya sea para realizar la ruta o para hacer una limpieza de la misma. Para ello, se presenta el siguiente formulario

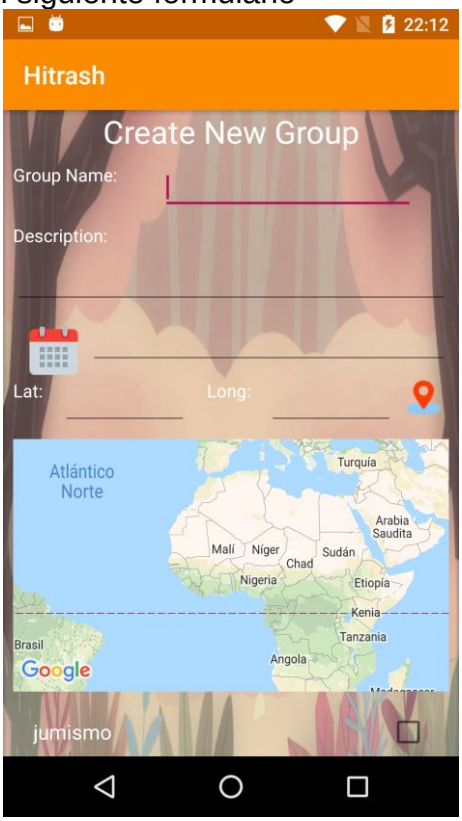


Ilustración 36 - Create Group

- Create Hiking Trail: Como se puede ver en la vista de lista de rutas, también podemos crear una ruta de senderismo pulsando sobre el

botón más de la parte inferior derecha. Para ello se presenta el siguiente formulario

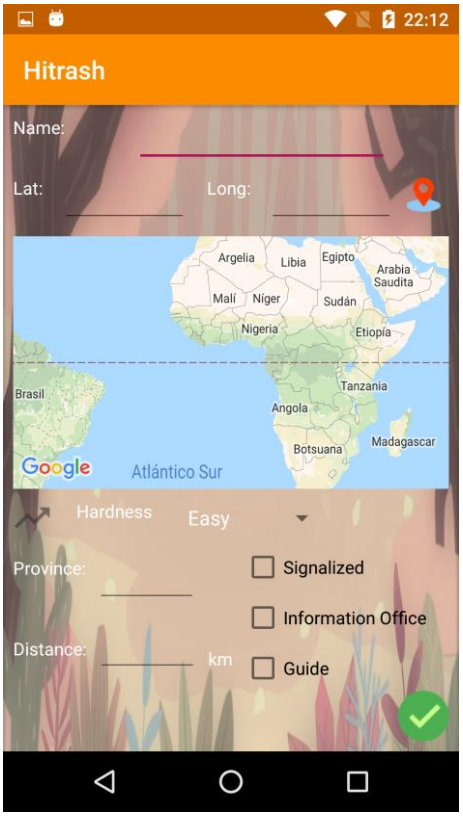


Ilustración 37 - Create Hiking Trail

Y con esto ya hemos completado todas las funcionalidades que pueden hacerse con la aplicación Hitrash en el estado actual en el que se encuentra la aplicación.