Universidad Zaragoza
1542

Proyecto Fin de Carrera

SiRAM: Un sistema automático de recomendación para las actividades de montaña

Autor/es

Cristofer Sanz Blasco

Director/es y/o ponente

Co-director: Ángel Luis Garrido Marín
Co-director: Javier Rincón Borobia
Ponente: Sergio Ilarri Ártigas

Ingeniería en Informática
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura

2015

Me gustaría dedicar este PFC a:

Mis padres y mi hermana, por su cariño y su apoyo incondicional en todo momento de mi vida.

Elena, mi pareja y alma gemela, por su amor, su infinita paciencia y por estar siempre a mi lado.

Resumen

El mundo de las actividades de montaña siempre ha sido considerado como un terreno lleno de peligros más allá del esfuerzo físico que exige. Mucha gente no se atreve a practicarlo y lo dejan a deportistas y expertos acostumbrados a tratar con ello.

El actual Proyecto Fin de Carrera se desarrolla en este ámbito, el de las actividades al aire libre. Lo hace en colaboración con SSUMMON, una comunidad de actividades en el medio natural. Consta de dos partes: una centrada en la investigación de un sistema experto y otra en el desarrollo de un proyecto software.

La primera parte trata de atajar el problema de la falta de seguridad y el desconocimiento de la gente que carece de experiencia en estas actividades. Su objetivo es investigar la posibilidad de utilizar un sistema de recomendación en el entorno de las actividades al aire libre. Este tipo de sistemas han sido explotados en otros contextos como el *comercio online* y el ocio literario, ofreciendo resultados satisfactorios. En esta parte se presenta **SiRAM** (Sistema de Recomendación para Actividades de Montaña), un sistema de recomendación diseñado en base a las necesidades de SSUMMON y a los datos de que dispone. Como paso previo a ello, se introducen esta clase de sistemas, sus funciones, aplicaciones y técnicas. Finalmente, se realizan experimentos sobre la herramienta diseñada para probar su efectividad.

En la segunda parte se construye **myssummon**, una aplicación de escritorio multiplataforma para gestionar las actividades de montaña. Su funcionalidad es doble: servir como sistema alternativo al utilizado hasta ahora por SSUMMON (una aplicación web) y apoyar en la realización de los experimentos sobre SiRAM. El desarrollo involucra las fases de análisis del contexto, de los requisitos, diseño del sistema e implementación de una versión de demostración, siguiendo un ciclo de vida iterativo.

Agradecimientos

Por el apoyo recibido, me gustaría agradecer la ayuda prestada a:

- **Ángel**, por haberme acompañado a lo largo de todo el proyecto y, sobre todo, por haberme guiado en los momentos más difíciles.
- **Javier**, por haberme enseñado parte de mundo del montañismo, el cual era completamente desconocido para mí. También por resolver todas las dudas que me han surgido sin ningún inconveniente.
- **Sergio**, por sus correcciones y ayuda altruista tanto en este PFC como a lo largo de toda la carrera.

Índice general

Capítulo 1

Introducción

En este capítulo se ofrece una visión muy general sobre el actual proyecto. En primer lugar, se describe en que contexto se ha desempeñado, se explican cuales son los objetivos perseguidos y que motivaciones lo han hecho posible. También se enumeran las tecnologías empleadas y se muestra cómo ha sido la planificación. En último lugar, se mencionará la estructura que tiene el resto del documento.

1.1. Contexto

El proyecto por completo está asentado en el contexto de las actividades al aire libre. Se ha desarrollado en colaboración con SSUMMON, una joven red social de montañismo.

SSUMMON nació en 2012 con la intención de cubrir un nicho que hasta el momento de su creación estaba desocupado. A través de ella, los deportistas y aficionados al montañismo pueden almacenar y compartir las actividades llevadas a cabo en la montaña. También se pueden crear nuevos grupos de personas, útil tanto para asociaciones de deportistas como para grupos de amigos. Además, es posible programar salidas, establecer amistades entre usuarios y comunicarse con cualquiera de ellos.

La red social cubre una gran variedad de actividades, todas ellas relacionadas con la montaña. De entre todas se pueden destacar la escalada, el ascenso a picos, el senderismo o la carrea por montaña. Por el momento, el acceso a la red social se realiza a través de una aplicación web que se encuentra en estado de desarrollo activo. La aplicación web se aloja en http://www.ssummon.com.

1.2. Objetivos

Este proyecto pretende lograr dos objetivos:

1. Estudiar la viabilidad para poner en marcha un sistema de recomendación

en el contexto de las actividades al aire libre. A este sistema se le ha apodado *SiRAM* (Sistema de Recomendación de Actividades de Montaña). Para ello, será necesario:

   a) Comprender el estado actual de estos sistemas expertos.

   b) Hallar en el contexto de SSUMMON información relevante para utilizarla en los experimentos de viabilidad.

   c) Diseñar el sistema de recomendación que se ajuste al entorno.

   d) Ejecutar experimentos sobre el sistema que evidencien la posibilidad y eficiencia de utilizar el sistema diseñado en este contexto.

2. Diseñar un sistema software de gestión de actividades al aire libre. Este debe cumplir dos objetivos:

   a) Desempeñar un subconjunto de las funciones que realiza el actual sistema de SSUMMON (su aplicación web).

   b) Transformar las actividades creadas en *datasets* utilizados en los experimentos sobre SiRAM.

   Se le ha dado el nombre de *myssummon*, queriendo evidenciar que se trata de una versión más personal de la aplicación web. Como paso previo, se analizará el contexto del sistema actual y los procesos que lo componen. Además, se implementará una versión de demostración del sistema diseñado.

1.3. Motivaciones

El mundo de las actividades de montaña siempre ha sido considerado como un terreno lleno de peligros más allá del esfuerzo físico que exige. Mucha gente no se atreve a practicarlo y lo dejan a deportistas y expertos acostumbrados a tratar con ello. La motivación de *SiRAM*, y la principal motivación de este proyecto, es conseguir acercar a la gente no habituada a practicar deportes en el medio natural a éste. Una solución para este problema sería a través de una herramienta que les recomendase la actividad idónea para sus gustos y sus capacidades.

Por otra parte, existen multitud de razones que motivan el desarrollo de *myssummon*. El propósito por parte de SSUMMON es construir dos versiones alternativas a la aplicación web. Una de ellas sería la aplicación de escritorio, y la otra, una aplicación para dispositivos móviles. Ambas deben estar preparadas para ser utilizadas sin conexión a Internet, de modo que la mayoría de las operaciones se puedan realizar sin ella. Dado el contexto de la red social, actividades en montaña, el poder prescindir de conexión a Internet para poder utilizar la aplicación es un punto a tener muy en cuenta. Quizás sea esta la característica diferencial más importante entre ambas aplicaciones. Por otro lado, atendiendo a su arquitectura monolítica, una aplicación de escritorio puede ser considerada

más robusta que una aplicación web, pues participan menos componentes en su ejecución.

Por último, se debe tener en cuenta que la red social está pensada para ser explotada económicamente. De este modo, la aplicación de escritorio forma parte de una estrategia comercial, siendo ofertada a aquellos usuarios reacios de compartir sus experiencias públicamente, como un mero organizador de actividades de montaña, obsequiando con funcionalidades adicionales a aquellos usuarios que las publiquen.

1.4. Tecnologías utilizadas

En esta sección se enumeran cuáles han sido las tecnologías empleadas en todo el proyecto de modo que el lector conozca de antemano que herramientas y estándares se han utilizado. Las tecnologías son presentadas en grupos atendiendo a la parte del proyecto en la que hn sido utilizadas.

1.4.1. Entorno de experimentación de SiRAM

Aquí se describen las herramientas que forman parte del entorno de experimentación de SiRAM y que han permitido ejecutar los experimentos diseñados:

- **Entorno Python:** Esta herramienta se ha ocupado de ejecutar los experimentos, desde el tratamiento previo de los datos hasta la aplicación de los algoritmos. Su elección es analizada en Apéndice A.1. Este entorno está compuesto por:
  - **Python 3.4:** Lenguaje de programación interpretado de propósito general.
  - **Numpy y Scipy:** Librerías de cálculo numérico consideradas la base de la computación científica con el entorno Python.
  - **Pandas:** Librería que introduce *Dataframes* en este entorno. Hace más cómodo y ágil el tratamiento de los conjuntos de datos.
  - **scikit-learn:** Completa librería de algoritmos sobre *machine learning*. También incorpora funciones de tratamiento previo de los datos (por ejemplo: normalización, *feature scaling*, tratamiento de valores nulos, etc.)
  - **IPython notebook:** Entorno interactivo que permite mezclar texto, código y gráficos. Agiliza la ejecución de experimentos.
- **Microsoft Office Excel 2013:** Popular aplicación de hojas de cálculo. Ha sido empleada para generar los resultados y su gráfico.

1.4.2. Desarrollo de myssummon

Las tecnologías que han participado en el desarrollo de la aplicación son:

- **Intellij IDEA 14:** Completo IDE (*Integrated Development Environment*) para Java. Ha hecho más sencillo el proceso de codificación incorporando funciones avanzadas, como por ejemplo, refactorización o detección de errores de compilación entre otras.
- **Scene Builder 2.0:** Entorno de diseño de interfaces visuales para JavaFX.
- **Git:** Herramienta para el control de versiones. También se ha utilizado para compartir el código en un repositorio online.
- **Maven:** Herramienta de gestión de proyectos. Permite automatizar la compilación, ejecución, testeo y empaquetado. También es útil como gestor de dependencias.
- **NSIS Installer:** Entorno para generar herramientas de instalación de software basado en *scripts*.

A continuación, aquellas tecnologías que forman parte del producto en sí:

- **Java 8:** Lenguaje de programación compilado multiplataforma.
- **JavaFX:** Conjunto de librerías para la construcción de interfaces visuales. JavaFX forma parte de la librería estándar de Java a partir de la versión 8. Su uso es analizado en Apéndice B.3.2.
- **Launch4j:** *Plugin* para Maven que envuelve artefactos JAR de Java en ficheros ejecutables de Windows.
- **SQLite:** Sistema gestor de base de datos relacional *serverless*. Su empleo es analizado en Apéndice B.3.1.

1.4.3. Documentación

Estas son las herramientas que han sido utilizadas para producir toda la documentación del proyecto, entre la que se encuentra esta memoria:

- **LaTeX:** Sistema de composición de textos orientado a escritos científicos. Para la composición de la memoria se ha utiliza el entorno **TeXstudio**.
- **Microsoft Office Visio 2010:** Potente herramienta para crear diagramas. Se ha utilizado para generar los gráficos de los apartados técnicos.
- **DbSchema:** Se ha utilizado para generar diagramas relacionales de los esquemas de base de datos.

1.4.4. Tecnologías comunes

A este conjunto pertenecen los **sistemas operativos**:

- **Debian 8:** En él se han llevado a cabo los experimentos de SiRAM y se ha desarrollado la aplicación myssummon.
- **Windows 7:** En éste se ha generado la documentación, incluyendo la creación de los diagramas.

1.5. Planificación

Este apartado comienza por describir la planificación que se trazó al comienzo del proyecto. Posteriormente se compara con la ejecución real del mismo. Las explicaciones se acompañan de cronogramas para una mejor comprensión.

En Figura 1.1 se muestra el cronograma de la planificación inicial. En él, se ven las fases agrupadas en función de la parte a la que pertenecen. La cabecera del diagrama muestra el tiempo medido en meses. La línea roja discontinua indica el *deadline* del PFC, la fecha de depósito.

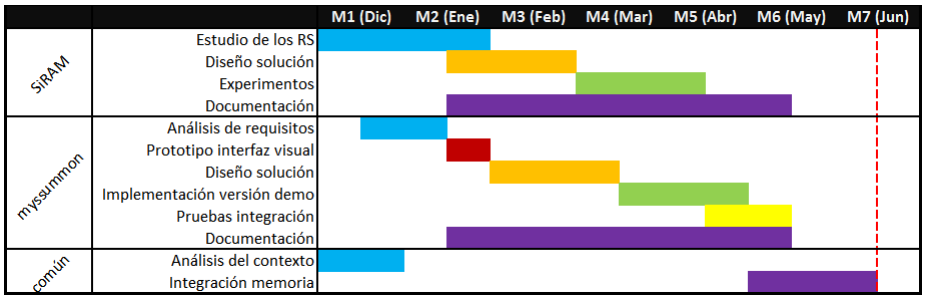


Figura 1.1: Cronograma planificado.

Como se ve en el diagrama, existen tareas que se pueden solapar entre ellas, como por ejemplo, la documentación de una parte y el resto de sus tareas, y otras que no, como por ejemplo, el diseño del myssummon y su implementación.

Atendiendo a la parte de myssummon, hay que aclarar que aunque el cronograma muestre un ciclo de vida en cascada, esto sólo es algo orientativo y muestra en que fase se debería haber comenzado a trabajar. En la realidad, el proyecto ha seguido un ciclo de vida iterativo volviendo a fases anteriores con información más precisa que la que se contaba al principio.

Sin embargo, ha habido sucesos que han impedido seguir el plan trazado al pie de la letra. En Figura 1.2 se muestra cuál ha sido el transcurso real del proyecto.

Como se ve, las fases de análisis de ambas partes comenzaron más tarde. Esto fue debido a exámenes pendientes que el alumno tuvo que terminar. También apareció una nueva fase en ambas partes, el análisis de alternativas, en la cual

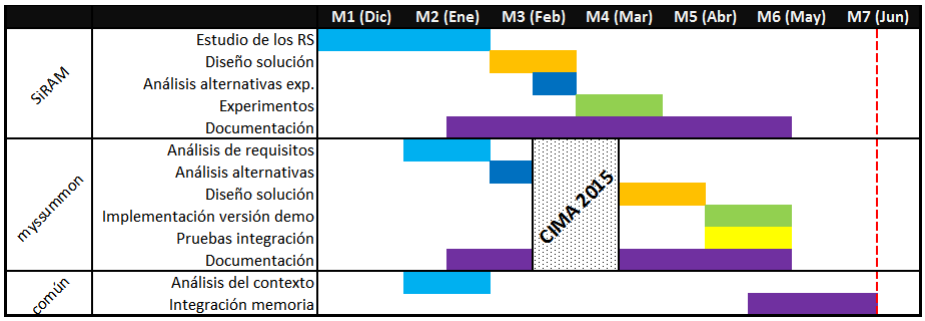


Figura 1.2: Cronograma real.

no se reparó en un comienzo pero que fue muy importante. Otras sin embargo desaparecieron, como es la fase de prototipo visual en la parte de myssummon. Se decidió que un prototipo no sería necesario dado que, como se puede leer más adelante (Apéndice B.2.1), la interfaz visual sería lo más similar posible a la de su aplicación web.

El hecho que más alteró el desarrollo del proyecto fue la conferencia CIMA 2015 en la que se publicó un artículo (Apartado 4.1). No se reparó en su existencia hasta dos meses antes de la entrega de los artículos, de modo que durante ese tiempo, el desarrollo de SiRAM se aceleró, y la construcción de myssummon se pausó.

1.6. Estructura del documento

Este documento tiene una estructura bastante sencilla. Está dividido en cuatro capítulos, de los cuales dos forman la parte central de la memoria:

- *CAPÍTULO 1. Introducción*: En el capítulo actual, se presenta el proyecto, sus objetivos y motivaciones. Además, se muestran las tecnologías empleadas y la planificación que se ha seguido durante el proyecto.
- *CAPÍTULO 2. SiRAM*: Este capítulo se dedica completamente a los sistemas de recomendación. Se presentan estos sistemas para poner en contexto al lector, se describe la herramienta creada así como los experimentos y resultados obtenidos.
- *CAPÍTULO 3. myssummon*: En él se trata todo lo relativo a la creación de esta herramienta software. Se muestran todas las fases de su desarrollo, desde el enunciado de sus objetivos hasta la implementación de una versión de demostración. Junto con el capítulo anterior, forman la parte principal de la memoria.
- *CAPÍTULO 4. Notas finales*: En este capítulo se recoge la valoración personal, las dificultades que ha habido que afrontar y los agradecimientos a los co-directores y el ponente del PFC.

Además de la parte principal, a este documento lo acompañan tres anexos que recogen todos los detalles necesarios para entender el sistema en profundidad, ya que por extensión o nivel de detalle no pueden estar situados en la parte principal.

- *ANEXO A. SiRAM* y *ANEXO B. myssummon*: Estos anexos amplían la información de los capítulos 2 y 3 sobre las herramientas SiRAM y myssummon.
- *ANEXO C. Manuales*: Este anexo alberga guías de usuario para instalar y poner en marcha la implementación de myssummon.

Capítulo 2

SiRAM: Sistema de recomendación de actividades de montaña

En el siguiente apartado, como paso previo al diseño de SiRAM, se introducen los sistema de recomendación, sus funciones, aplicaciones y técnicas más importantes.

2.1. Introducción a los Sistemas de Recomendación

Los sistemas de recomendación (RS en adelante) son herramientas software que utilizan un conjunto de técnicas para proporcionar sugerencias y ayudar en procesos de toma de decisiones. Su orientación ha sido mayoritariamente comercial aunque son utilizados en multitud de campos para aplicaciones muy diversas. Su concepción fue consecuencia de una simple observación: la gente tiende a confiar en opiniones ajenas a la hora de tomar decisiones. Éstos son especialmente útiles cuando se ha de tomar una decisión *en un ámbito desconocido* o cuando el *conjunto de opciones es muy amplio*.

Normalmente, los RS procuran decisiones personalizadas para cada usuario del mismo. En el otro extremo, están los sistemas que ofrecen recomendaciones no personalizadas o genéricas. Son sistemas mucho más simples, que no utilizan información particular de los usuarios ni algoritmos complejos. Es por ello que no suelen ser incluidos en el mismo grupo que los primeros.

En la práctica, las razones para hacer uso de un RS son incalculables. Enumerando las más interesantes:

- Ayudan a aumentar el número de artículos o servicios vendidos debido a que las recomendaciones del RS encajarán razonablemente bien entre las

necesidades que el usuario busca suplir.

- Dan a conocer productos menos habituales. Esta es una de las mayores ventajas que un RS puede brindar y que más puede interesar a alguien que busca explotarlo económicamente. El hecho es que vender un producto muy exitoso o conocido no es una tarea complicada (es la principal labor de un sistema que ofrece sugerencias no personalizadas). Sin embargo, poder ofrecer a un usuario un producto más extraño o especial, y que la recomendación tenga buena acogida es todo un logro.

- Hacer que los usuarios sean más fieles, lo cual es conseguido tratando al usuario como una persona de valor, reconociéndolo y personalizando sus preferencias. Cuanto mejor diseñado esté un RS (incluyendo su aspecto visual e interacciones), mejor se podrá aprovechar esta característica.

Como se ve, un sistema de recomendación tiene mucho que ofrecer. Las ventajas son aún más ciertas cuanto mejor sea su diseño.

2.1.1. Funciones de los RS

Aunque las funciones que ofrece un RS están basadas en sugerir algo, todas difieren en ciertos aspectos. A continuación se listan interesantes funciones que un RS aporta:

- Encontrar los mejores artículos. Es una de las características más sencillas y habituales, de utilidad para aquellos usuarios que simplemente buscan unos cuantos productos o artículos que encajen perfectamente con su perfil.

- Encontrar todos los artículos buenos. Esta función, menos habitual que la anterior, se da en aquellos casos donde es necesario conocer todos los artículos que encajan en un patrón determinado.

- Encontrar un conjunto o una secuencia de artículos buenos. Son funciones avanzadas donde se busca que la combinación de varios artículos (p.e. un paquete vacacional) o incluso el orden de consumo (p.e. una lista de reproducción de música) encajen en el perfil del usuario.

Además, existen otras funciones secundarias, a veces simples efectos colaterales, que ayudan en pos del uso de los RS. Algunas son:

- Procurar que el usuario cree un perfil lo suficientemente bueno, utilizando la premisa de que cuanto mejor sea el perfil, mejor serán las recomendaciones ofertadas. Del mismo modo, esto puede generar un comportamiento positivo entre usuarios que contribuyen ofreciendo opiniones útiles para otros usuarios.

- Demostrar el buen funcionamiento, dado que algunos usuarios no confían en estos sistemas lo suficiente.

2.1.2. Objetos de un RS

Tratar los sistemas de recomendación como el mismo tipo de sistema en el que se pueden variar unos pocos parámetros supone perder flexibilidad y precisión. Esto es debido a que cada sistema puede ser tan particular como se pretenda pudiendo definir exhaustivamente modelos, algoritmos, su interacción persona-ordenador y cada aspecto del mismo. A pesar de lo amplio del dominio de los RS, la mayoría de ellos, si no todos, presentan varios objetos en común [9]:

Artículos Son los objetos a recomendar. Se pueden caracterizar de varias formas, siendo interesante por ejemplo usar tanto la utilidad como la complejidad. La utilidad indica en qué grado el artículo será interesante para el usuario. La complejidad indica cómo de difícil será "modelar" el artículo si queremos explotar la información que lo define en la recomendación.

Usuarios Es el sujeto al que van destinadas las recomendaciones, que interactúa activamente con el sistema de recomendación. Un usuario no sólo se define por los atributos que se le piden de forma explícita. También se puede tener en consideración aspectos implícitos como pueden ser las acciones que realiza dentro del sistema informático relacionado con el software de recomendación.

Transacciones Son denominadas así a las interacciones explícitas de los usuarios con los artículos, por ejemplo, las valoraciones.

2.1.3. Técnicas de recomendación

En términos generales, la predicción de utilidad de cualquier RS puede expresarse como una función real $R(u, i)$ donde u es el usuario para el cual se calcula la utilidad de i, el ítem o artículo. Cada técnica implementará la función anterior como sea debido y, cada RS en particular, utilizará modelos de usuarios y artículos tan complejos como sea requerido.

Son muchos los autores que tratan de definir claramente los diferentes tipos de sistemas recomendadores. La mayoría de autores los agrupan en cuatro clases bien definidas [9]:

Filtrado colaborativo (*collaborative filtering*) Esta técnica es la más simple siendo, de hecho, el enfoque original en los RS. El sistema recomienda artículos que fueron útiles a otros usuarios similares. Su principal ventaja, es que esta similitud entre usuarios es hallada en base a las transacciones realizadas por los usuarios, por lo que no precisa información sobre el artículo a recomendar.

Basadas en contenido (*content-based*) El sistema recomienda artículos similares en contenido (las características asociadas a ellos) a otros que fueron útiles en el pasado. Este tipo de sistema analiza el contenido valorado por el usuario y, en función de ello, construye su perfil. Este perfil

refleja las características u objetos que interesan al usuario en cuestión. La predicción consiste sencillamente en comparar el perfil construido con las característica del objeto de modo que cuanto más parecidos sean entre sí más útil será la recomendación.

Basado en conocimiento (*knowledge-based*) El sistema requiere conocimiento específico sobre un dominio para calcular cómo de útil será un artículo para un usuario. Ese conocimiento se puede expresar mediante ontologías, por ejemplo. Dado que se basan en un conocimiento externo y no en opiniones de otros usuarios, como en el filtrado colaborativo, son métodos buenos para resolver casos donde escaseen las valoraciones ajenas (por ejemplo, en la venta de vehículos).

Híbridos Este tipo de sistemas utiliza una combinación de técnicas donde las ventajas de unas suplen las desventajas de otras. Por ejemplo, el filtrado colaborativo sufre de un problema denominado arranque en frío (*Cold-Start*) que consiste en que no puede recomendar artículos en el comienzo del mismo debido a la falta de transacciones. Este problema no afecta a los sistemas basados en conocimiento, por lo que es ideal para corregir dicha deficiencia.

Todos estos enfoques tienen en cuenta información acerca de los usuarios y/o de los artículos que recomiendan. Sin embargo, existen ciertos dominios donde la información contextual juega un papel importante. Información como *la fecha*, *la ubicación*, o *el tipo de dispositivo usado* es decisiva en estos casos. Los encargados de explotar estos datos se denominan sistemas dependientes del contexto.[1]

2.2. Diseño de SiRAM

En esta sección se presentan las fuentes datos que podría utilizar SiRAM para sus recomendaciones. También se muestra la arquitectura de la herramienta y su evolución hasta ser lo que ahora es.

2.2.1. Fuentes de información

Desde que se comenzó a diseñar la aplicación web, SSUMMON ha seguido la filosofía de solicitar al usuario la mínima información posible. Esto queda patente en el formulario de registro, el cual carece de campos para introducir la edad, teléfono u otros datos secundarios considerados obligatorios o importantes en otras aplicaciones web.

Esta filosofía puede beneficiar a la usabilidad del sistema, pero por otro lado, ha dificultado la labor de diseño del sistema de recomendación, además de la obtención de datos por parte de la aplicación web.

Como alternativa a los campos del formulario de registro, se muestran a continuación las fuentes de información para el sistema de recomendación:

[1] Habitualmente funcionan complementando otras aproximaciones que utilizan información más relevante que la contextual.

1. Valoraciones de actividades: Proporciona al modelo la afinidad por una determinada actividad. Es el dato base del sistema de recomendación.

2. Acciones que demuestran interés: Se puede considerar equivalente a una valoración bastante positiva de una actividad en particular. Por ejemplo: actividades compartidas en redes sociales, generación de informes PDF, etc.

3. Búsquedas avanzadas: Proporcionan al modelo información sobre los tipos de actividades que le interesan al usuario, localizaciones preferidas, si habitúa a viajar con niños, etc. Ayuda a reafirmar la afinidad hacia un tipo de actividad concreta.

4. Creación de actividades y salidas: Proporciona al modelo información muy similar a la obtenida mediante los parámetros de búsqueda.

2.2.2. Arquitectura

Desde el principio se tuvo la idea de combinar varias de las técnicas vistas en la introducción, Apartado 2.1.3, en un *sistema híbrido*. La concepción de SiRAM ha sido fruto de la evolución y la versión actual de SiRAM está precedida por otras dos:

- En un primer instante, se pensó en un sistema que utilizase el **Filtrado Colaborativo**, el **Filtrado Basado en Contenido** y el **Filtrado Basado en Conocimiento**. La intención era que las tres técnicas funcionasen como módulos desacoplados en paralelo de modo que cada uno produjese sus propias recomendaciones y el sistema las entregase en un conjunto haciendo invisible la arquitectura modular.

- Similar a la primera, la segunda versión mantenía el **Filtrado Colaborativo** y el **Filtrado Basado en Contenido** como dos módulos independientes trabajando en paralelo. Las recomendaciones que estos módulos producían alimentaban el módulo de **Filtrado Basado en Conocimiento**, que esta vez trabajaba refinando[2] las filtraciones de los primeros métodos en vez producir las suyas propias.

La versión actual de SiRAM, la cual se puede observar en Figura 2.1, es muy parecida a las anteriores. En ella siguen presentes las técnicas de **Filtrado Colaborativo** y de **Filtrado Basado en Contenido** como dos módulos independientes. Además, ahora el módulo de refinado también utiliza variables contextuales.

Tras esta breve presentación de la arquitectura, se describen los módulos que conforman el sistema en mayor detalle.

[2] Aumentando o disminuyendo la relevancia que tenían las recomendaciones.

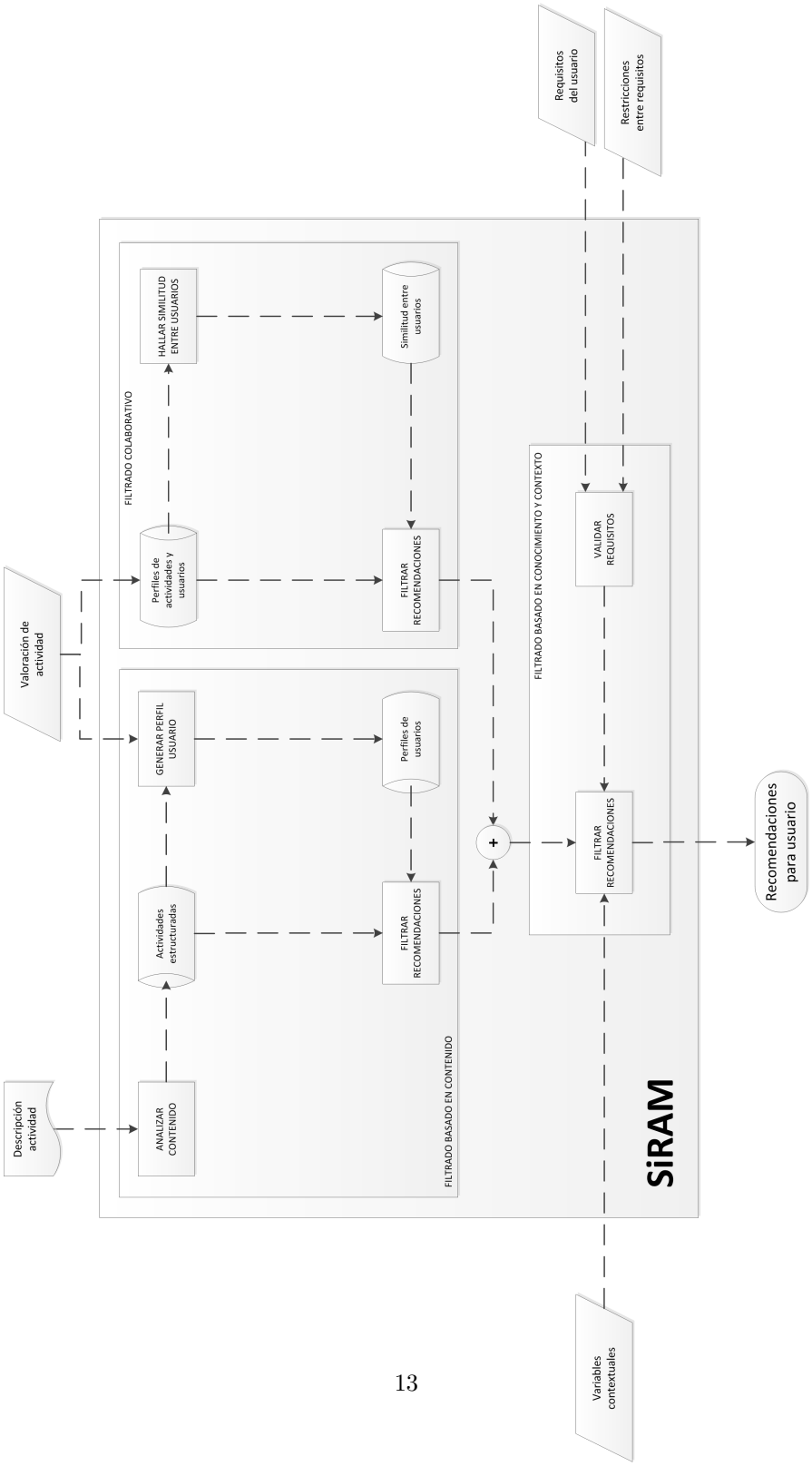


Figura 2.1: Arquitectura de la versión actual de SiRAM.

Filtrado basado en contenido
Este módulo comienza analizando una actividad de entrada. En este proceso, el sistema extrae y convierte los atributos de la actividad *no estructurada* o *semiestructurada*. Éstos se guardarán en el **almacén de actividades estructuradas**.

Por otra parte, un conjunto de actividades estructuradas junto con las valoraciones aportadas por un usuario, generarán un perfil de usuario. Este se guardará en el **almacén de perfiles de usuarios**.

Finalmente, se produce el filtrado de recomendaciones mediante un proceso de comparación de las actividades estructuradas y el perfil del usuario.

Filtrado colaborativo
El módulo de filtrado colaborativo sólo recibe como entrada las valoraciones de los usuarios a las actividades. Normalmente, estos datos están representados de forma matricial donde las columnas son los usuarios y las filas, las actividades. Así, esta matriz contiene los perfiles tanto de usuarios como de actividades.

Una vez se cuenta con los perfiles, se halla la similitud entre los usuarios y se almacena.

Las recomendaciones con este método se filtran en función de los perfiles de las actividades y la similitud entre los usuarios.

Filtrado basado en conocimiento y contexto
Como se ha mencionado, el último módulo afina las recomendaciones producidas por los otros dos módulos. Esto es así porque las variables contextuales suelen ser menos relevantes para las recomendaciones que la definición de una actividad o los gustos del usuario.

La aproximación de filtrado basado en contenido utiliza información del usuario en forma de requisitos. Teniendo en cuenta las fuentes de información que SiRAM puede utilizar, definidas en Apartado 2.2.1, estos requisitos provendrían de las búsqueda realizadas por el usuario o de las actividades y salidas creadas por él. Ésta aproximación tiene en consideración una lista de restricciones entre requisitos del usuario, por ejemplo: *una actividad físicamente muy exigente no puede ser apta para niños menores de 2 años*). Si dos requisitos son incompatibles entre sí, no se utilizarían en el proceso de filtrado, evitando perjudicar las recomendaciones.

La aproximación de filtrado basado en el contexto utilizaría fuentes de información externas, por ejemplo: *la época del año, la localización actual del usuario, etc.*

2.3. Experimentos

Se ha decidido hacer pruebas intensivas con las siguientes metodologías de trabajo: las basadas en contenido y los filtros colaborativos.

El motivo de centrarnos en primer lugar en estas dos aproximaciones es que son métodos ampliamente estudiados y su uso es muy extendido en otros

Atributo	Descripción
Noches	Número de noches que ha durado la actividad.
Duración	Tiempo (en días) que ha costado realizarla.
Tipo de recorrido	Recorrido de ida y vuelta, circular o travesía.
Apta para niños	Si esta para niños y a partir de que edad.
Tipo de actividad	El deporte en que consiste.
Dificultad física	Forma física requerida para poder realizarla.
Dificultad técnica	Conocimiento técnico requerido para poder realizarla.
Distancia	Longitud (en metros) del recorrido.
Tipo de terreno	Tipo de suelo del recorrido

Tabla 2.1: Atributos del modelo de una actividad.

sistemas de recomendación, por lo que sirven perfectamente como **línea base de trabajo** sobre la cual ir introduciendo mejoras futuras (Apartado 4.2) a medida que avance la investigación.

2.3.1. Conjunto de datos

Para la realización de los experimentos se han usado datos reales proporcionados por la plataforma SSUMMON. Se han escogido 30 actividades y tomado como referencia las evaluaciones de 15 usuarios. Las actividades han sido escogidas de modo que haya un buen espectro en lo referente a dificultad. Del mismo modo, se han elegido usuarios con distintos niveles de experiencia en actividades de montaña y con diferentes situaciones contextuales (jóvenes, mayores, solteros, casados, con familia, etc.)

Dado que SSUMMON proporciona información *semiestructurada* de los recorridos, se ha sacado partido de esta ventaja para modelar las actividades. Para el modelo, se ha escogido una serie de atributos considerados como los más relevantes según el asesoramiento de montañeros expertos pertenecientes a la FAM (Federación Aragonesa de Montaña). Se muestran estos atributos en Tabla 2.1 junto con una descripción de cada uno de ellos.

Para poder explotar estos datos, ha sido necesario aplicar dos transformaciones previas:

1. La primera consiste en codificar valores no numéricos sustituyendo atributos de tipo textual por valores numéricos. Se puede consultar la codificación en Apéndice A.2.1 (Tabla A.4).

2. La segunda transformación consiste en normalizar los valores concretos de cada atributo para cada actividad, logrando que todos ellos queden expresados en la misma escala (entre -1 y 1).

Tras estos dos procesos, los datos están preparados para operar con ellos. El conjunto de datos se puede consultar en Apéndice A.2.1 en ambas versiones: natural (Tabla A.1) y codificada (Tabla A.2).

2.3.2. Metodología

1) Método basado en contenido

El método analizado en primer lugar es el basado en contenido. Su premisa es que a los usuarios les interesarán actividades similares a otras que les hayan interesado en el pasado.

El proceso de recomendación de este método es como sigue: el sistema deberá "construir" el perfil de cada usuario a partir de la *descripción estructurada de las actividades* y de las *valoraciones numéricas* que le haya aportado el usuario a cada actividad. Una vez el sistema conozca el tipo de actividades que le interesa a cada usuario, predecir la puntuación es tan sencillo como comparar el *perfil del usuario* con el *perfil de la actividad* utilizando alguna *métrica de correlación* y expresar el resultado en la misma escala que la de las valoraciones aportadas. A continuación, se describe este proceso en mayor detalle:

a) Aprendizaje del perfil de usuario

Aprender el perfil de un usuario es crear un modelo ajustado a los gustos de un usuario que pueda decidir en qué medida una actividad en concreto le puede interesar. En este caso, el perfil de un usuario es un vector similar al de cualquier actividad, donde el valor de los atributos está cercano al que encontraría en una actividad que le pareciese de interés. Para calcular estos perfiles se utiliza el algoritmo de Rocchio [11, 1]:

$$x_u = \sum_{i \in I_u} r_{ui} x_i \tag{2.1}$$

En 2.1, x_u es el perfil del usuario u, x_i el perfil de la actividad i y r_{ui} la valoración asignada por el usuario u a la actividad i. Así pues, el perfil de un usuario es la media de los perfiles de las actividades con las que interactuó, ponderada con las valoraciones que él ha proporcionado. De este modo, si un usuario vota muy positivamente una actividad, ésta pesará más en su perfil que si la vota con una puntuación baja o mediocre.

b) Predicción de una actividad

Un usuario queda representado por el mismo vector que una actividad, salvando la diferencia de sus valores. Esto hace que se puedan comparar directamente mediante alguna métrica de correlación como el *coeficiente de Pearson* o la *similitud del coseno* [1, 13]:

$$\cos(x_u, x_i) = \frac{x_u^T x_i}{||x_u|| ||x_i||} \tag{2.2}$$

En 2.2, x_u es el perfil del usuario u y x_i el perfil de la actividad i. El resultado de esta métrica es un valor continuo perteneciente al

rango $[-1, 1]$ que, trasladándolo al rango de los valores posibles de las valoraciones del conjunto de datos, $[1, 5]$,será la predicción de la valoración que asignaría el usuario u a la actividad i.

2) Método de filtrado colaborativo

El segundo método que analizamos es el filtrado colaborativo. A diferencia del primero, no necesita información sobre las actividades. Éste método utiliza únicamente los datos de las transacciones, es decir, las valoraciones de los usuarios sobre las actividades. Explota la idea de que las actividades que fueron interesantes para ciertos usuarios lo serán para usuario de perfil similar.

El proceso de recomendación de este método es el siguiente [13]: el sistema calcula un coeficiente por cada par de usuarios que indica la similitud entre ambos. Para hallar la valoración que un usuario aportaría a una actividad dada, basta con ponderar las valoraciones que otros usuarios le han dado en el pasado utilizando los pesos de similitud para cada usuario. Detallando el proceso paso a paso:

a) Cálculo de pesos

Como se ha mencionado, los pesos muestran la similitud entre cada par de usuarios. Para hallarlos se emplea la *similitud coseno*, métrica de correlación que ya se empleó en el método previo, que a continuación vemos expandida [1]:

$$\cos(x_u, x_v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2} \sqrt{\sum_{i \in I_v} r_{vi}^2}} \tag{2.3}$$

En 2.3, I_{uv} es el conjunto de actividades valoradas por ambos usuarios u y v, r_{ui} la valoración aportada por el usuario u a la actividad i y r_{vi} la valoración del usuario v a la actividad i.

b) Predicción de la actividad

En este método, predecir la valoración de un usuario hacia una actividad consiste en calcular el valor medio de las valoraciones del resto de los usuarios considerando los pesos entre ellos:

$$\hat{r}_{ui} = \frac{\sum_{v \in I_v} w_{uv} r_{vi}}{\sum_{v \in I_v | w_{uv}|}} \tag{2.4}$$

En 2.4, I_v es el conjunto de usuarios que ha votado la actividad i, w_{uv} la correlación entre los usuarios u y v y r_{vi} la valoración del usuario v hacia la actividad i. En este caso, la predicción se hallará en el rango del conjunto de valoraciones de entrada, $[1, 5]$.

2.3.3. Resultados

Una vez los métodos son capaces de realizar predicciones, estamos en disposición de averiguar cómo de eficaces resultan y cuál de los dos ofrece mejores predicciones. El experimento consiste en dividir los datos de que disponemos en dos conjuntos: *conjunto de entrenamiento* y *conjunto de prueba*. El primero de ellos será utilizado para calcular pesos entre usuarios y aprender los perfiles de los usuarios. El segundo será el conjunto con el que mediremos el rendimiento de cada método, realizando predicciones y comparándolas con valoraciones reales aportadas por usuarios. La proporción utilizada para dividir los datos es 80/20, constituyendo el conjunto de entrenamiento el 80 % de los datos y el de prueba el 20 % restante.

Al mismo tiempo que se preparan los conjuntos de datos, se debe escoger un estimador del error producido por cada método al efectuar los pronósticos. Se ha escogido como estimador el RMSE [14], debido a su popularidad y fiabilidad, así como al hecho de ofrecer un resultado normalizado:

$$RMSE(f) = \sqrt{\frac{1}{R_{test}} \sum_{r_{ui} \in R_{test}} (f(u,i) - r_{ui})^2} \tag{2.5}$$

En 2.5, $f(u,i)$ es el valor predicho por el usuario u para la actividad i y r_{ui} el valor real con el que el usuario u votó la actividad i. Cuanto menor sea el valor de RMSE para un método, más eficaz se considerará.

Finalmente, conscientes del papel que podía jugar el azar a favor de uno de los métodos donde las predicciones fueran más favorables por la disposición de los datos al dividirlos, se ha hecho uso de la técnica denominada validación cruzada (*cross-validation*). Con ella, en lugar de estimar el error sobre un único conjunto de datos, lo hemos estimado sobre *cuatro permutaciones* del mismo, de modo que los datos utilizados durante el entrenamiento pueden ser datos de prueba en otra permutación y viceversa. La estimación del error final será la media del error estimado para cada permutación.

En Figura 2.2, se muestra el RMSE para cada método y permutación ("Perm") del conjunto de datos. En todas las iteraciones, el método de filtrado colaborativo proporciona un RMSE menor frente al método de filtrado basado en contenido. Es interesante observar las últimas columnas: los valores medios del RMSE para cada método. Éstos indican en qué medida las predicciones realizadas por cada técnica se desviarán de las valoraciones reales que los usuarios harían.

Más allá de la comparación entre ambos métodos, los resultados se consideran positivos, pues muestran un error medio aproximado a 0.6 puntos con filtrado colaborativo, y 1.3 puntos con filtrado basado en contenido. Esta desviación se considera aceptable para que el sistema pueda diferenciar una actividad que no gusta a un usuario de otra que encaja completamente con sus gustos.

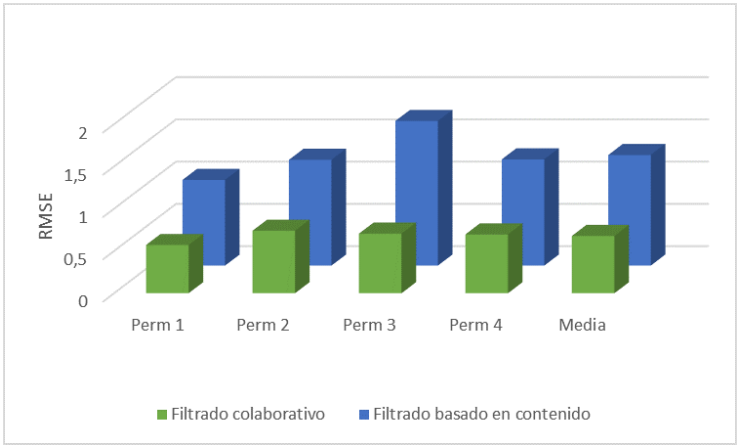


Figura 2.2: RMSE de filtrado colaborativo y de filtrado basado en contenido.

Capítulo 3

myssummon: Sistema gestor de actividades de montaña

Este capítulo está dedicado por completo al desarrollo de la aplicación de escritorio. Cada sección coincide con una fase del ciclo de vida de myssummon.

3.1. Introducción

Se pretende desarrollar una aplicación de escritorio que complemente en funciones a la aplicación web *www.ssummon.com*. Los objetivos que ha de cumplir son:

- Servir como organizador de actividades en la naturaleza de los usuarios.
- Poder sincronizar los datos relativos a estas actividades la aplicación web.
- Funcionar sin necesidad de conexión constante a Internet.
- Generar ficheros que contengan todas las actividades del usuario para ser utilizados como *datasets* en los experimentos de SiRAM.

Por tanto, la aplicación debe proporcionar una interfaz para recoger los datos relativos a actividades y un sistema de almacenamiento propio, a ser posible similar al ya existente en ssummon.com.

Muchas de las funciones de la aplicación web estarán presentes en esta, tales como el tratamiento (inserción, modificación y eliminación) de actividades. Aunque no lo estarán otras con **componente social** como el contacto entre usuarios, valoración de actividades públicas, gestión de grupos de usuarios, etc.

El único objetivo de negocio que se plantea es maximizar el porcentaje de usuarios de la aplicación que comparten en la aplicación web las actividades introducidas en el organizador.

3.1.1. Resumen de objetivos

El desarrollo de myssummon se plantea un objetivo técnico:

> Construir una aplicación de escritorio que sirva como organizador de actividades montañesas de los usuarios. Deberá cumplir las siguientes características:
>
> - Seguridad y robustez, evitando comprometer los datos del usuario, los servicios de SSUMMON y realizando un correcto tratamiento de errores.
> - Ofrecer funciones similares a las de la aplicación web centrándose en la gestión de actividades deportivas y prescindiendo de la faceta social.
> - Ofrecer una función extra que permita generar *datasets* a partir de las actividades del usuario para ser utilizados en los experimentos de SiRAM.
> - Operar sin necesidad de conexión a internet salvo en el momento en que se ejecuta por primera vez, para poder comprobar los credenciales del usuario que la utiliza.
> - Funcionar en diferentes plataformas.
> - Permitir la sincronización de datos con la aplicación web mediante la publicación de actividades. Para ello deberá existir conexión a internet.

También se plantea un objetivo de negocio:

> Aumentar el número de actividades publicadas en la red social mientras se maximiza el número de usuarios que comparten actividades al público. Esto ayudará a atraer a otros usuarios a la red social.

En los siguientes apartados, se aclaran rápidamente ciertas nociones del contexto de SSUMMON. En Apéndice B.1 se realiza un estudio en profundidad de su entorno, procesos y necesidades.

3.1.2. Sobre los actores

Debido a que SSUMMON utiliza una estrategia económica basada en el modelo freemium existen diferentes tipos de usuarios, donde cada uno extiende las funcionalidades del anterior:

- Usuario normal: Es el rol asignado a los usuarios recién registrados. Puede llevar a cabo la mayor parte de las funciones: ver y administrar actividades, salidas, grupos, realizar búsquedas, etc.

- Usuario pro: Es el rol idóneo para usuarios habituales. Además de ofrecer las funciones del usuario normal, ofrece algunas adicionales (Figura 3.1): almacenamiento propio para las fotografías incrustadas, tramificación de actividades, creación de sumarios en PDF y eliminación de publicidad.
- Usuario guía: SSUMMON también fue ideado con la intención de que guías, empresas o personas individuales dedicadas a dirigir grupos en actividades de montaña, pudieran sacar provecho de eso. Es por ello que existe un tercer tipo de usuario, el usuario guía, el cual además de poder llevar a cabo todas las operaciones del usuario pro, cuenta con dos nuevas funcionalidades orientadas a gestionar su actividad: podrá visualizar estadísticas detalladas sobre el tráfico en sus actividades y aparecerá en la sección de guías de la aplicación web.

Figura 3.1: Funciones adicionales del usuario PRO.

Figura 3.2: Funciones adicionales del usuario GUÍA.

Además de los usuarios que emplean el sistema, existe otro tipo de actor: el **administrador general**. Este actor engloba todas las labores de administración y mantenimiento de la aplicación web. En sistemas más complejos, estas labores se suelen dividir en diferentes tipos de actores. Sin embargo, dadas las condiciones de desarrollo de SSUMMON y el equipo encargado de ello, todas las tareas son desempeñadas por este actor.

3.1.3. Sobre las actividades

El concepto de actividad es muy importante en SSUMMON. Todo gira alrededor de ellas y se representan con objetos complejos, por lo que conviene comprenderlas bien.

Una actividad está compuesta por tramos (al menos por un tramo). Cada tramo representa un tipo de actividad de entre las siguientes: Pico, Corredor,

Cresta, Vía Ferrata, Escalada, Senderismo, BTT, Esquí de travesía, Esquí de fondo y Carrera por montaña.

Las últimas actividades (Senderismo, BTT, Esquí de travesía, Esquí de fondo y Carrera por montaña) se agrupan internamente en un nuevo tipo de actividad: Ruta. Esto se debe a que todas ellas se definen con los mismos parámetros, y aunque los deportes no compartan una base común, a efectos técnicos forman parte del mismo grupo.

Por otro lado, existen varios tipos de escalada: Clásica o Alpina, Deportiva, Artificial, Mixta, en Hielo, Psicobloc y Builder. Esto queda reflejado en el modelo de datos de la aplicación (Apartado 3.3.1) puesto que los atributos que definen una escalada Alpina son diferentes de los que definen una escalada Builder.

3.2. Análisis

En esta sección se tratan todos aquellos procesos que han sido necesarios para comprender el sistema en desarrollo. Se comienza describiéndolo en profundidad y modelando la interacción que tienen los usuarios con éste. También se ilustran con detenimiento los procesos más complejos y se concluye con una vista de la arquitectura del sistema completo.

3.2.1. Descripción del sistema

A través de sucesivas reuniones con el equipo de SSUMMON, se ha conseguido obtener una descripción completa del sistema a desarrollar. Está redactada en un estilo estructurado para facilitar la extracción de requisitos de ella. Éstos se pueden consultar en Apéndice B.2.1.

Gestión de usuarios La aplicación sólo podrá ser utilizada por usuarios de la aplicación web. Durante la primera ejecución, en la que se requiere conexión a Internet, será necesario que el usuario introduzca sus credenciales de la aplicación web para autenticarlo o, adicionalmente se podrá crear un nuevo usuario de la web, debiendo proporcionar el usuario los datos necesarios para registrar un nuevo usuario.

Desde la aplicación de escritorio sólo se podrán crear nuevos usuarios, pero no se permitirá modificarlos ni eliminarlos.

El usuario podrá visualizar información estadística sobre su perfil del mismo modo que está disponible en la web.

La operación de creación de un nuevo usuario se realizará a través de *web services*.

Gestión de actividades La aplicación podrá insertar, modificar y eliminar actividades tanto de la base de datos local como de la ya existente en ssummon.

Del mismo modo que en la creación de un nuevo usuario, cualquier operación de gestión de actividades remota se llevará a cabo mediante el uso de *web*

services. El proceso de las operaciones remotas se describe en más detalle en el apartado de sincronización.

Sólo los *usuarios pro* podrán administrar actividades compuestas por varios tramos. Al crear una nueva actividad multitramo se comprobará que el usuario sea de tipo pro. De no ser así, un mensaje alertará de que la operación no se puede llevar a cabo. Si un usuario deja de ser pro, sus actividades multitramo se conservarán en su perfil pero, al ser usuario normal, no podrá modificarlas ni crear nuevas.

Sincronización La sincronización es el mecanismo que permite mantener los datos de las actividades entre ambas aplicaciones en estado coherente. Puesto que es requisito que la aplicación de escritorio esté preparada para trabajar sin conexión a Internet, la forma de sincronizar el contenido entre las actividades es explícita y a voluntad del usuario. Existirá un mecanismo de sincronización que el usuario activará cuando desee el mismo estado en la actividad para ambas plataformas. En ese momento la conexión a Internet será requisito indispensable.

Además del mecanismo para sincronizar las actividades individualmente, habrá otro para sincronizar varias actividades a la vez, la sincronización por lote. Se deberán indicar las actividades a sincronizar y posteriormente se activará el mecanismo de sincronización por lote.

Existe la posibilidad de mantener actividades de forma privada sin que aparezcan también en la aplicación web. Para estas actividades el mecanismo de sincronización estará deshabilitado. En cada actividad existirá un mecanismo que fije las actividades como públicas o privadas con los siguientes efectos:

- Transición de Privado a Público: Crea una actividad nueva en la web de forma inmediata.
- Transición de Público a Privado: Elimina la actividad correspondiente en la de forma inmediata.

Al sincronizar se deben comparar las fechas de última modificación de la actividad en ambas plataformas. Si la actividad en la aplicación web ha sido modificada más recientemente que en la aplicación de escritorio, se preguntará al usuario cuál de las dos versiones mantener, mostrando las fechas de última modificación de cada plataforma.

De igual modo que en la gestión de actividades, la sincronización de actividades multitramo sólo estará disponible para los usuarios pro. Los usuarios que dejen de ser pro no podrán sincronizar sus actividades multitramo, pues los usuarios normales no cuentan con tales privilegios. Las actividades monotramo podrán ser sincronizadas por cualquier tipo de usuario.

Para poder sincronizar las mismas actividades bajo dos plataformas es necesario contar con una forma de identificar la misma actividad en ambas aplicaciones. Para esto se almacenará la clave de identificación de la actividad en la web, IDW, en la aplicación local y se asociará a la clave de identificación de la actividad en la aplicación de escritorio, IDE.

Búsqueda y filtrado Se podrá realizar búsquedas de actividades entre todas las existentes en el perfil del usuario buscando en los campos de título y descripción de la actividad.

Se podrán filtrar las actividades existentes en el perfil del usuario atendiendo a su tipo de actividad, de igual forma que en la aplicación web.

Mapa A diferencia del resto de funciones, el mapa requiere una conexión permanente a Internet para su plena funcionalidad. En caso de existir esta, se mostrará el track de la actividad sobre la cartografía online de OpenStreetMap. Como alternativa, en caso de no existir conexión a Internet, mostrará únicamente el track de la actividad.

Editor de textos Al desarrollar la aplicación web, se consideró importante que la descripción de las actividades pudiera contener texto formateado, imágenes y demás elementos multimedia. Por ello, el campo descripción en las actividades es un elemento HTML. Para evitar que fuese el usuario el que tuviese que codificarlo, se pensó en utilizar un editor de textos HTML. La elección fue CKEditor por tratarse de una herramienta muy potente en su ámbito.

El contrapunto aquí es que al encargarse de tareas como la inserción y extracción de imágenes incrustadas en el texto, dificulta la tarea de sincronización de las imágenes y vídeos entre ambas plataformas. Por este motivo, se estudiará utilizar la misma herramienta en la aplicación de escritorio o el uso de otra que solvente los problemas de acoplamiento funcional.

Informes PDF Deberá de estar disponible la generación de informes de actividad en pdf ya presente en la aplicación web. Igualmente, sólo estará disponible para los usuarios pro por considerarse una característica avanzada.

Algunas de las funcionalidades presentes en la web no lo estarán en la aplicación de escritorio. Aquí se enumeran algunas de las más importantes para aclarar que no se han olvidado en el análisis del sistema:

- La información meteorológica que se muestra junto a cada actividad no estará disponible en la aplicación de escritorio. Esta requeriría de conexión permanente a Internet, lo que contradice la filosofía del sistema.
- En las descripciones de las actividades se podrá incluir enlaces a vídeos que existen en internet. Sin embargo, no se podrán incluir vídeos ubicados en el equipo local, por razones obvias de tamaño.
- No estará disponible el comparador de actividades presente en la aplicación web.

***Datasets* CSV** Deberá de estar disponible la generación de *datasets* como ficheros CSV conteniendo todas las actividades del usuario. Aunque esta fun-

cionalidad va ligada a la experimentación con SiRAM, estará disponible para cualquier usuario[1].

Esta funcionalidad deberá cumplir los siguiente requisitos para asegurar la compatibilidad con los ficheros de *datasets* que SiRAM emplea:

- Los ficheros utilizarán el caracter "," como separador de campos.
- Cada fila contendrá un *identificador numérico autoincremental* seguido por los valores de los campos de la actividades que se observan en Apartado 2.3.1 (Tabla 2.1) codificados (Tabla A.4).
- La primera fila del fichero será la cabecera de los valores con los nombres de los campos correspondientes.

Interfaz visual La principal directriz en el diseño de la interfaz visual es que se asemeje lo más posible al de la aplicación web. Así pues contará con muchos elementos en común con ciertas diferencias:

- Las actividades que pertenezcan al perfil del usuario, ya sean públicas o privadas. Se mostrarán todos los campos que se muestran en la aplicación web, incluyendo los atributos, descripción y mapa.
- El perfil de usuario incluyendo sus estadísticas.
- Un cuadro de búsqueda de actividades y otro de filtros por tipo de actividad.

También habrá nuevos elementos:

- Un mecanismo para activar la sincronización y otro para la sincronización por lotes.
- Cuadros de alerta para avisar al usuario cuando se vaya a producir alguna operación destructiva.
- Enlaces a elementos importantes como son: ayuda, blog de ssummon o la web principal de ssummon.

Por último, la aplicación carece de otros elementos como:

- Los relacionados con la faceta social: grupos, salidas, contactos, etc.
- La función de búsqueda avanzada.
- La función de comparación de actividades.
- La Información meteorológica asociada a cada actividad.

[1] Posiblemente en un futuro, la función de importación/exportación de actividades forme parte de las operaciones de los usuarios como mecanismo de *backup*.

Persistencia Simplificando la tarea de diseño del sistema, se utilizará un esquema de base de datos lo más similar posible al de la aplicación web. Esto implica continuar utilizando un SGBD relacional.

La aplicación web utiliza el servicio Geonames, una base de datos con ingente cantidad de información geográfica, para verificar la existencia de las localidades introducidas por el usuario en la gestión de actividades. Con el fin de evitar la necesidad de conexión a Internet, se descargará la base de datos de Geonames, integrándola en la de la aplicación y evitando el uso de un servicio adicional.

Sistema de ficheros En la aplicación, la base de datos permitirá almacenar entre otras cosas, atributos propios de cada actividad. Pero además de estos atributos, cada actividad lleva asociada cierta información en forma de ficheros, como son los tracks, imágenes y croquis. En Figura 3.3 se puede observar la topología del sistema de ficheros que utiliza la aplicación web, la cual se quiere seguir utilizando en myssummon.

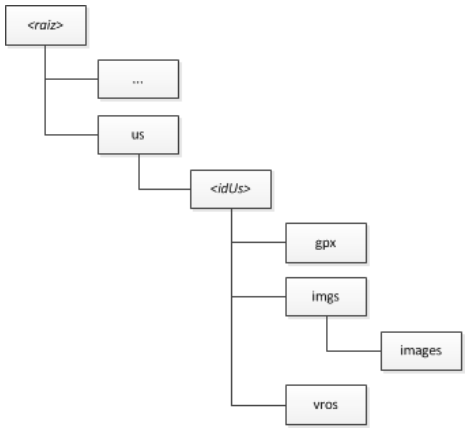


Figura 3.3: Topología del sistema de ficheros.

Los elementos encerrados entre "« »" son variables. El significado de cada directorio es:

«**raíz**» Es el directorio base donde está instalada la aplicación.

us Directorio donde se almacenan los ficheros relativos a todos los usuarios.

«**idUs**» Es el número identificador único correspondiente a un usuario determinado.

gpx Contiene las rutas de todas las actividades pertenecientes a un usuario dado.

imgs Contiene imágenes tanto del perfil del usuario como la imagen portada de las actividades.

images Contiene todas las imágenes pertenecientes a las actividades a excepción de la de portada (imágenes insertadas como parte de su descripción).

vros Contiene los croquis de las actividades.

Además de la estructura de ficheros, en el caso de las imágenes también quedan definidas las reglas de denominación, siendo éstas:

- Para el perfil de usuario: imgus«**idUs**».«**ext**».
- Para la portada de actividad: imgpor«**idAc**».«**ext**».

Donde «**idUs**», igual que antes, es el identificador único del usuario al que pertenece la imagen; «**idAc**» es el identificador único de la actividad; «**ext**» es la extensión del fichero. Estas reglas son utilizadas por el programa para renombrar las imágenes que el usuario proporciona. El resto de ficheros mantienen el nombre original.

Finalmente, las extensiones utilizadas y permitidas son:

- Tracks: Extensión **.gpx**.
- Imágenes: Extensión **.jpg** o **.png**.

Conexión a Internet La aplicación estará preparada para trabajar sin conexión continua a Internet. No obstante, efectúa operaciones remotas, para lo cual requiere conexión a Internet de forma discreta. Estas actividades son:

- Autenticar o registrar un nuevo usuario al comienzo de la aplicación.
- Hacer públicas o privadas las actividades del perfil del usuario.
- Sincronizar una o varias actividades.
- Visualizar el *track* de una actividad sobre la cartografía de *tiles* de *OpenStreetMap*.

Además, cualquier operación que implique la modificación de la base de datos remota, utilizada por la aplicación web, se hará a través de *web services* (gestión de usuarios y actividades, pág. 24) habilitados para tal fin.

3.2.2. Casos de uso

En base a la descripción del sistema del apartado anterior, se puede deducir la interacción de los actores con el sistema. La interacción se modela a través de casos de uso. Su diagrama, se puede observar a continuación. Un desglose más detallado de éstos se puede consultar en Apéndice B.2.2.

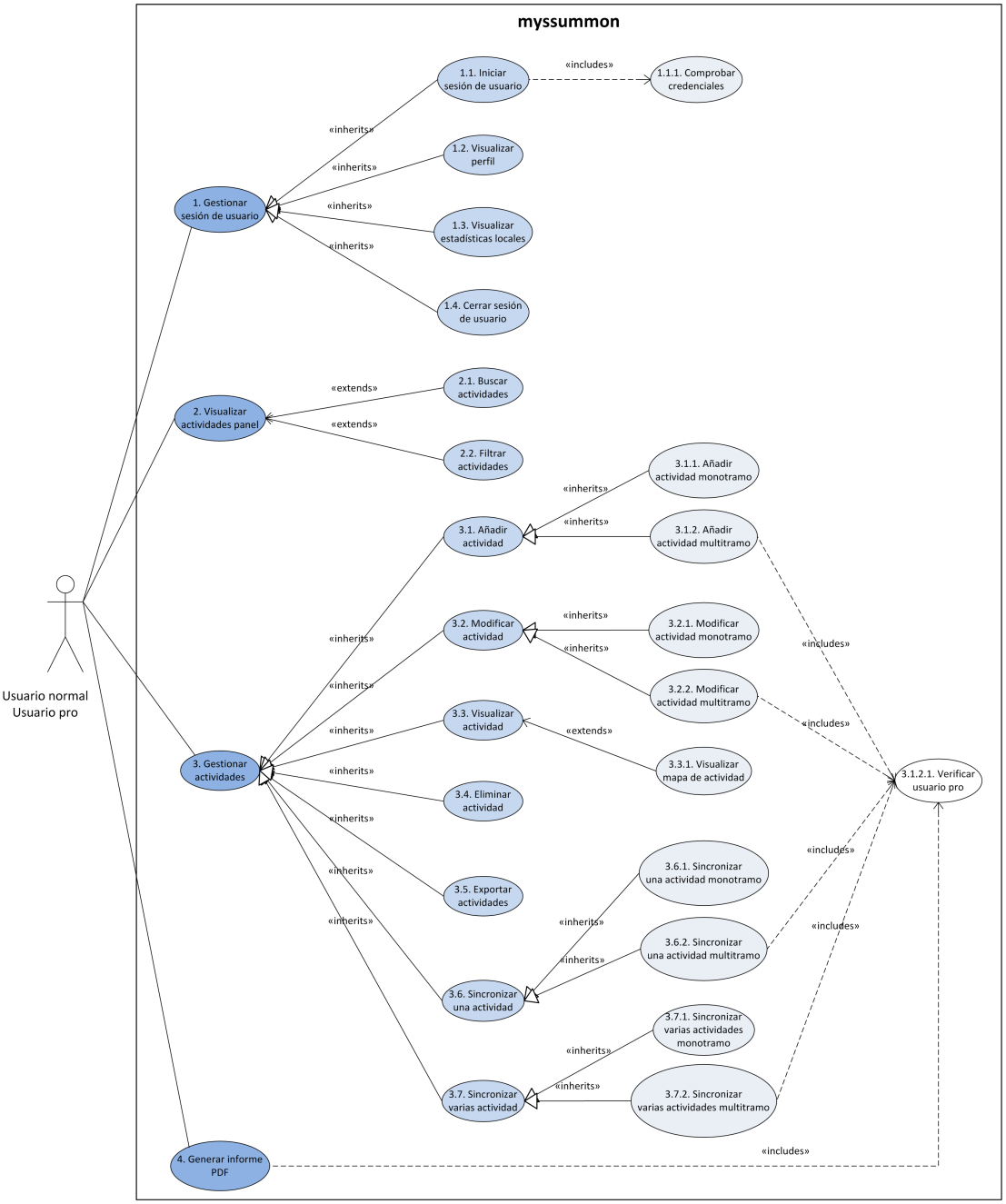


Figura 3.4: Diagrama de casos de uso.

3.2.3. Procesos especiales

A continuación se explican aquellos procesos que pueden ser complejos o no triviales, apoyándonos en diagramas de actividad. Aquellos procesos que resulten más difíciles de explicar se describirán también mediante diagramas de secuencia, ya que con ellos el nivel de detalle puede ser mayor.

Autenticación por primera vez
El proceso de autenticación de un usuario por primera vez es especial. Cada vez que un usuario entra en el sistema por primera vez, a parte del proceso normal de autenticación, se deben obtener datos que pertenecen al usuario. Esto se puede ver en Figura 3.5, donde a las acciones normales de autenticación (*Solicitar credenciales* y *Comprobar credenciales*) le siguen:

- Descarga de metadatos: Todos los datos del perfil del usuario.
- Descarga de actividades: Las actividades creadas por el usuario. Se marcarán automáticamente como públicas ya que a partir de este momento existen en ambos entornos.

Registro
Este proceso es bastante similar al de autenticación. Como se ve en Figura 3.6, existen dos diferencias:

- En lugar de comprobar los credenciales, se verifican que los datos de registro del usuario sean correctos y sigan el formato adecuado.
- En vez de descargar los datos del usuario, se crea uno nuevo con la información que se ha proporcionado mediante el uso de *web services*.

Publicación
El proceso de publicación se puede ver en Figura 3.7, y en más profundidad en Figura 3.8.

Al publicar una actividad lo primero que se hace es crear una actividad an la web (esta no puede existir si la actividad no es pública).[2] La creación de la actividad también implica crear los objetos dependientes de ella (como los tramos, imágenes y tracks).

El segundo paso consiste en asociar las actividades de ambas plataformas. Tal y como se describió en "la descripción del sistema" (pág. 24), manteniendo un campo de identificación remoto, IDW, con el valor del identificador de la actividad en la base de datos remota.

[2]La definición de actividad pública es que exista en ambas plataformas aunque no mantenga un estado coherente.

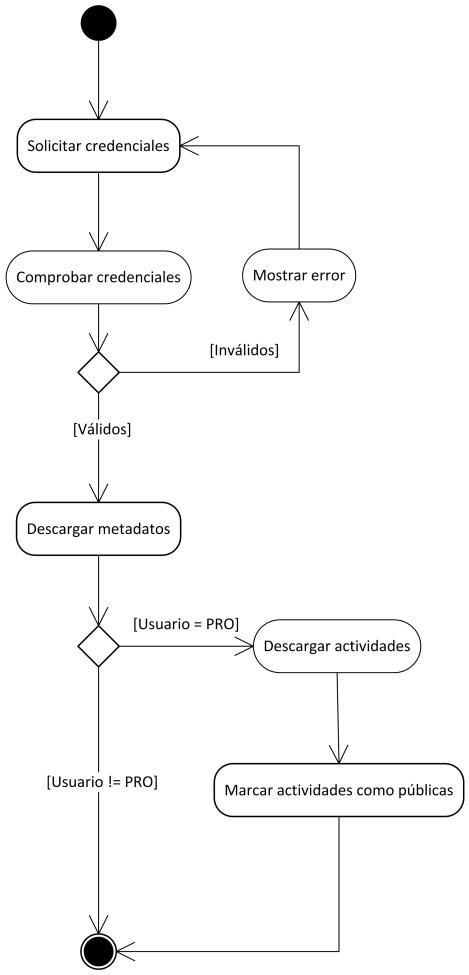


Figura 3.5: Diagrama de actividad del proceso de primera autenticación.

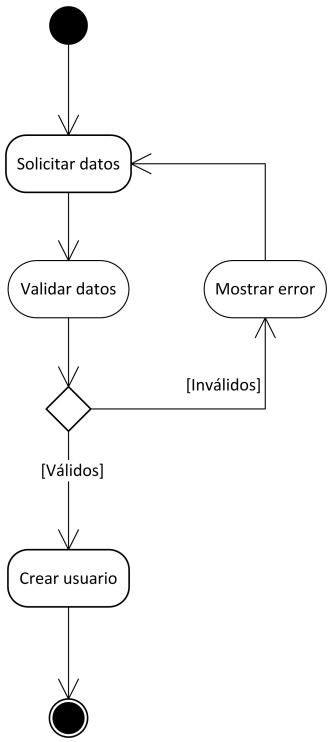


Figura 3.6: Diagrama de actividad del proceso de registro de usuario.

Figura 3.7: Diagrama de actividad del proceso de publicación de actividad.

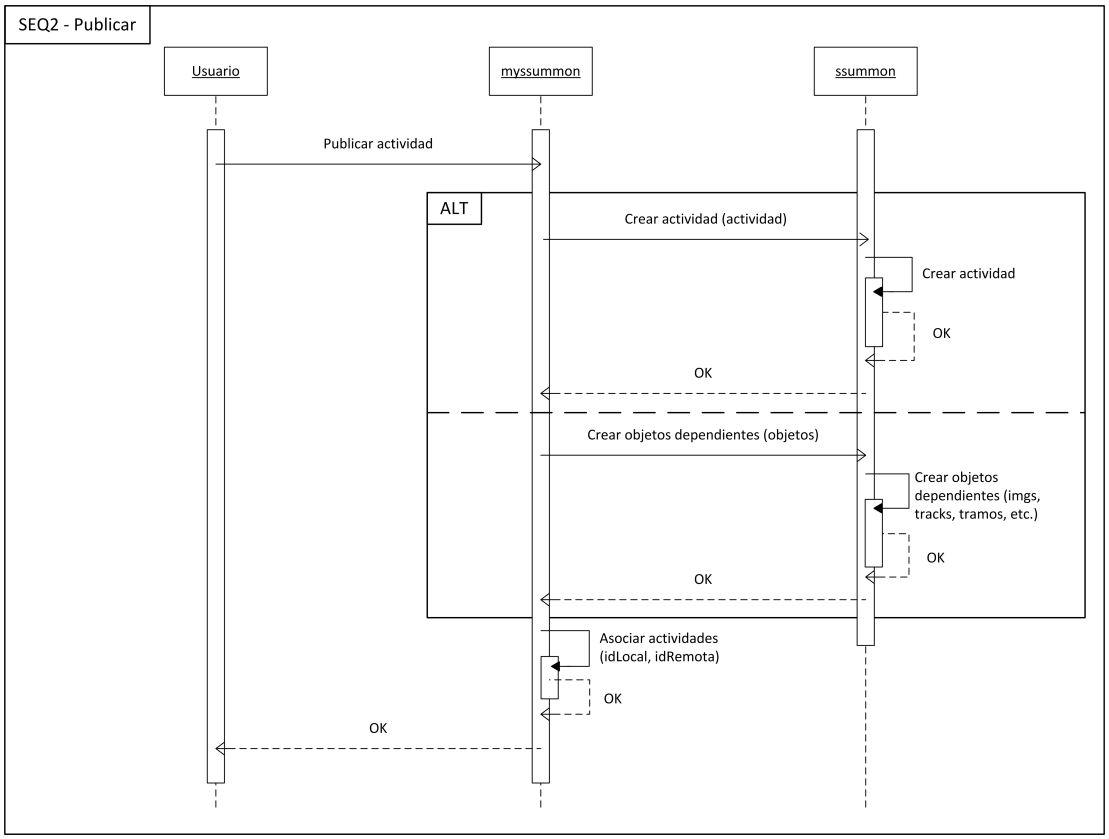


Figura 3.8: Diagrama de secuencia del proceso de publicación de actividad.

Privatización
La privatización es el proceso opuesto a la publicación de una actividad y, como consecuencia, hará que la actividad, y sus objetos dependientes desaparezcan de la aplicación web.

Este proceso (Figura 3.9 y Figura 3.10) comienza hallando el identificador de la actividad en la base de datos remota, IDW (pág. 24). Después, se solicitará la eliminación de la actividad y sus objetos mediante *web services*. Finalmente, se deshará la asociación creada al publicar una actividad.

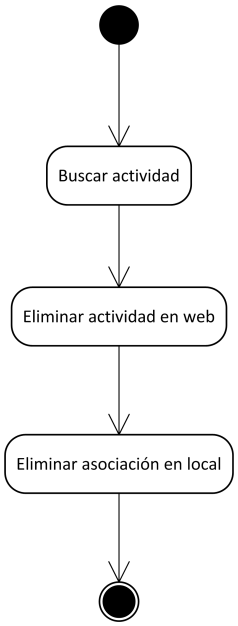


Figura 3.9: Diagrama de actividad del proceso de privatización de actividad.

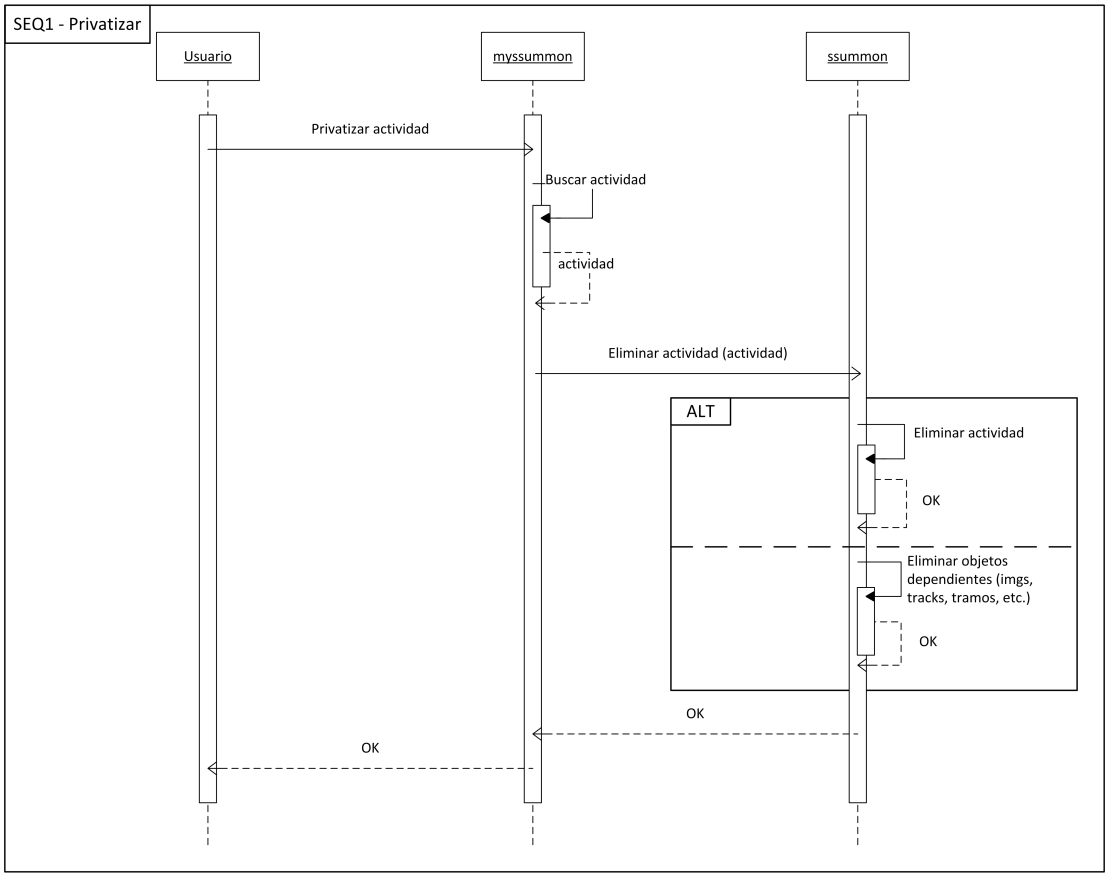


Figura 3.10: Diagrama de secuencia del proceso de privatización de actividad.

Sincronización

La sincronización es el proceso al que más atención se le dedica ya que guarda ciertos detalles que no conviene descuidar.

Este proceso consiste en igualar los estados de una actividad en las dos plataformas, es decir, actualizar la información de la actividad en una de las plataformas con la información en la otra. Como es evidente, la condición inicial para poder ejecutar este proceso sobre una actividad es que ésta sea pública, esto es, que exista en ambas plataformas.

Como se aprecia en Figuras 3.11 y 3.12 (esta última proporciona mayor detalle), una vez se cuenta con los estados de la aplicación en ambas plataformas, se comparan las fechas de actualización para conocer en que plataforma la actividad ha sido modificada más recientemente. En función de esto, se realizan las siguientes acciones:

- Si la versión de la actividad de myssummon es la más reciente, se actualiza la versión de ssummon con la información de la versión de myssummon. Este proceso, mostrado en Figura 3.13, consiste simplemente en actualizar tanto los atributos de la actividad en ssummon como los de sus objetos dependientes a partir de la misma información desde myssummon.[3]

- Si la versión de la actividad de ssummon es la más reciente, se advertirá al usuario sobre ello y se le pedirá que confirme cual de las dos versiones desea conservar. En caso de escoger la versión de escritorio se ejecutará el proceso que se acaba de describir. Si por el contrario escoge la versión de ssummon, tal y como se aprecia en Figura 3.14, se descargarán la actividad y sus objetos dependientes desde ssummon, y se actualizará con ellos los objetos propios de myssummon.

[3] De nuevo, esta operación se llevará a cabo mediante los *web services* disponibles en SSUMMON.

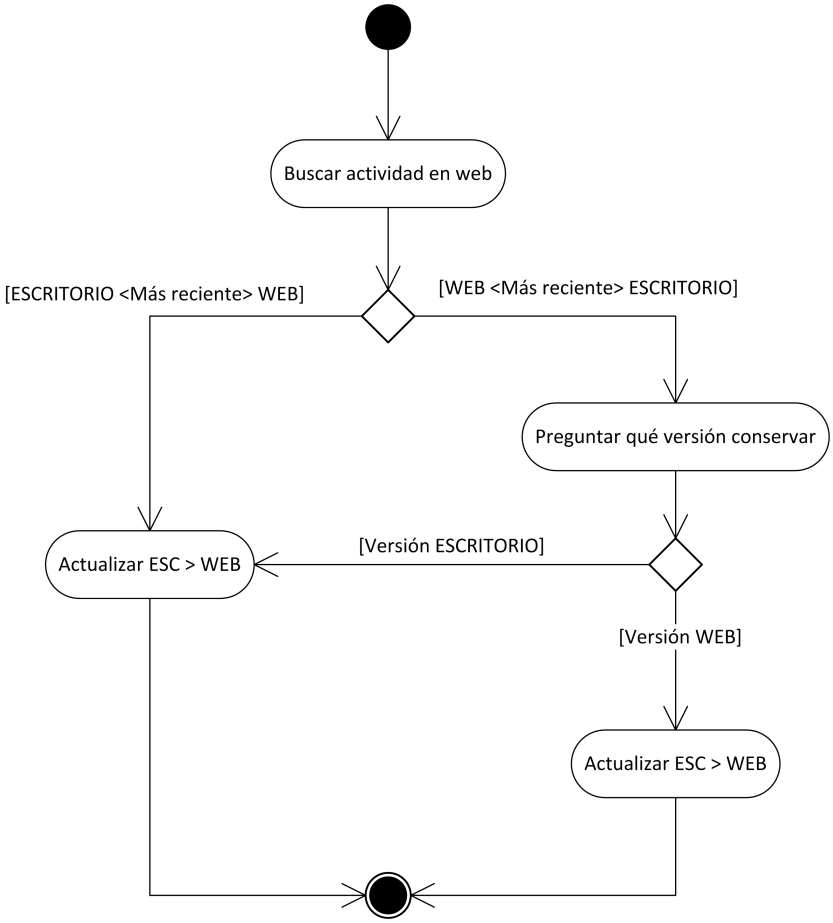


Figura 3.11: Diagrama de actividad del proceso de sincronización de actividad.

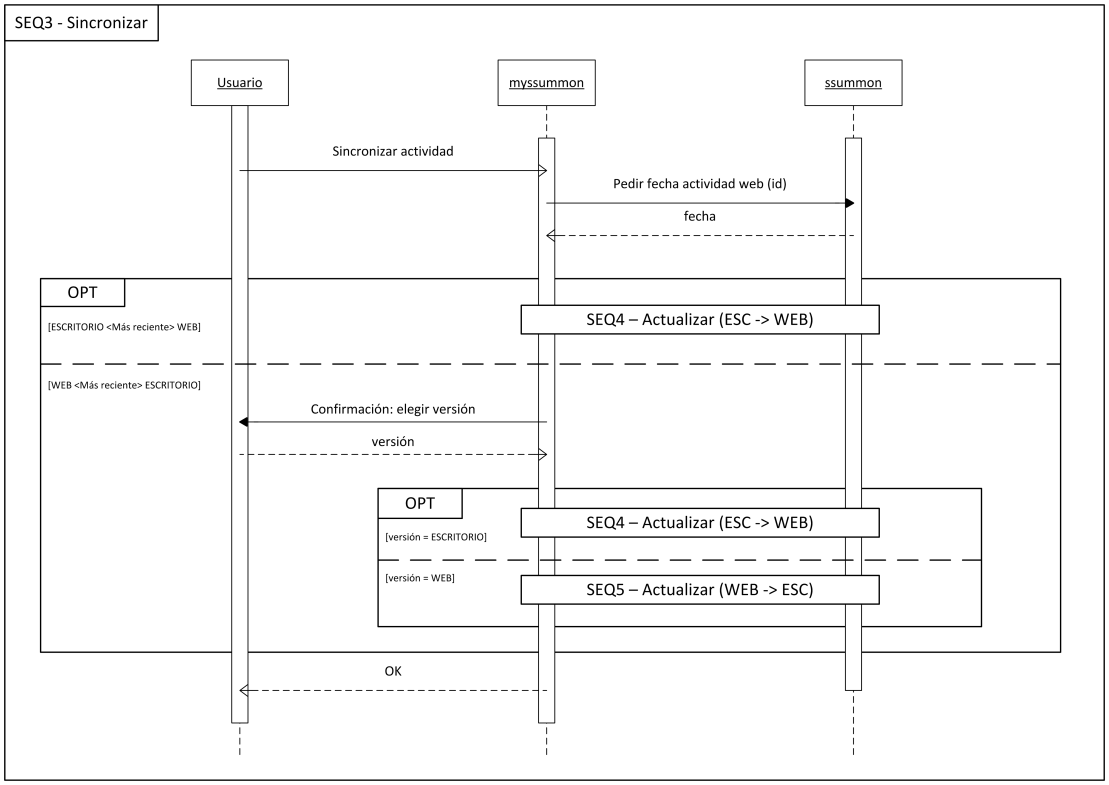


Figura 3.12: Diagrama de secuencia del proceso de sincronización de actividad.

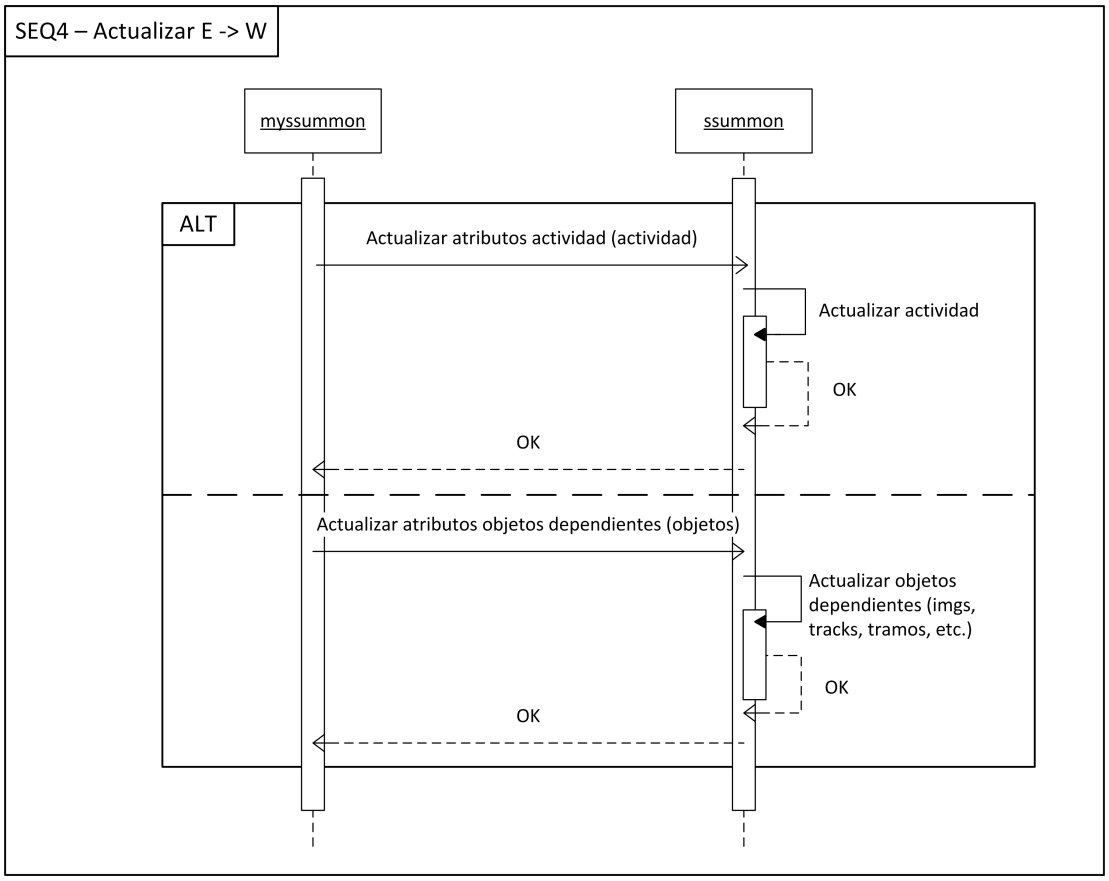


Figura 3.13: Actualización de actividad de escritorio a web.

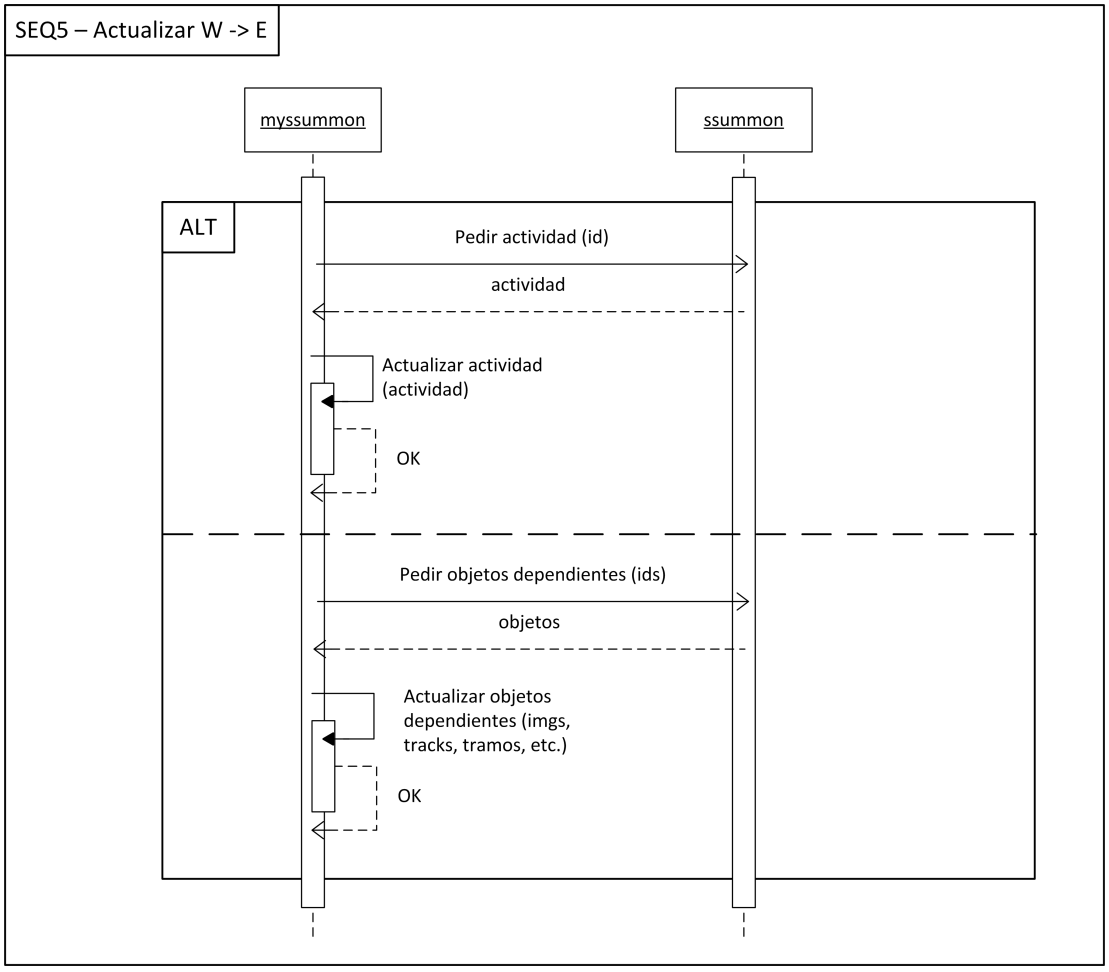


Figura 3.14: Actualización de actividad de web a escritorio.

3.2.4. Arquitectura

En Apéndice B.1, se presenta el contexto de SSUMMON. Para completar la visión del sistema legado[4], se muestra en Figura 3.15 un diagrama de despliegue que representa la arquitectura de SSUMMON antes del desarrollo de este proyecto. En él se observa como el navegador web del cliente se conecta al interfaz público de SSUMMON, su página web. Ésta depende de la base de datos integrada en el mismo sistema y de los *web services* de Geonames para proveer sus funciones.

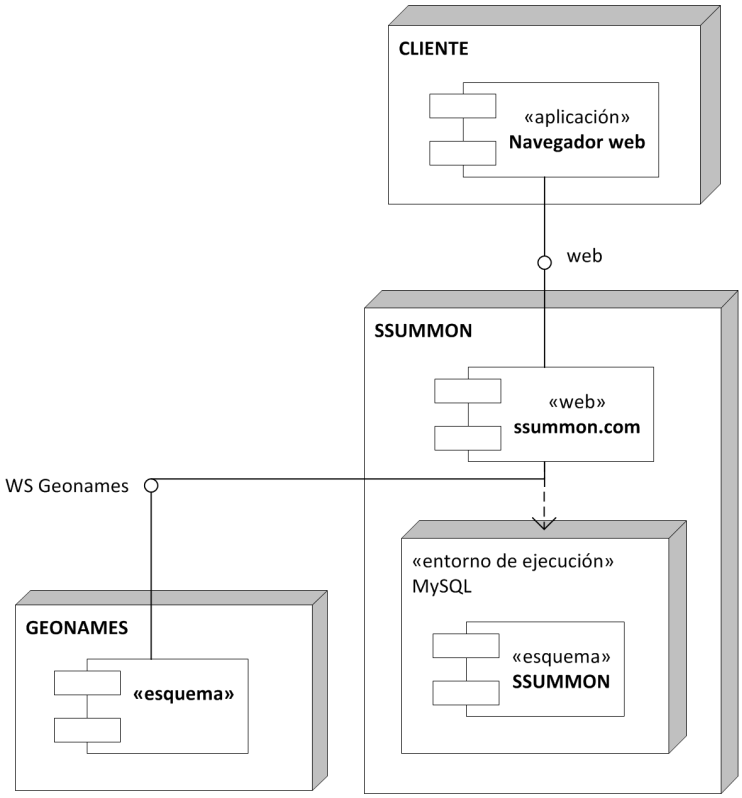


Figura 3.15: Arquitectura del sistema legado.

Sumando a la arquitectura del sistema legado la información recabada sobre el nuevo sistema (descripción, requisitos, interacción de usuarios, etc.) se produce un nuevo diagrama de despliegue que representa a la arquitectura del sistema completo.

En este diagrama (Figura 3.16) se aprecia como el nodo cliente ahora integra el entorno myssummon. Éste aparece como una aplicación vista como una

[4]Se utiliza el término sistema legado no como un sistema que ha quedado anticuado, si no como la parte del futuro sistema que ya existía anteriorente.

caja negra (no se revela ningún detalle de su diseño) dentro de un entorno de ejecución Java y un par de esquemas de base de datos.[5] Además, la aplicación utiliza los *web services* de SSUMMON para las operaciones de gestión remota.

Por otro lado, la aplicación web ha dejado de depender directamente de la base de datos para utilizar los *web services* corriendo en el mismo entorno, resolviendo así el segundo problema que se detectó anteriormente en el estudio del entorno (Apéndice B.1.2).

Finalmente, en el entorno de SSUMMON, se ve como se integra el esquema de Geonames en la base de datos global.

[5] Nótese que los esquemas aparecen como artefactos fuera de un entorno de ejecución. Esto se debe a que el SGBD del sistema actual es *serverless* y las bases de datos son simples ficheros.

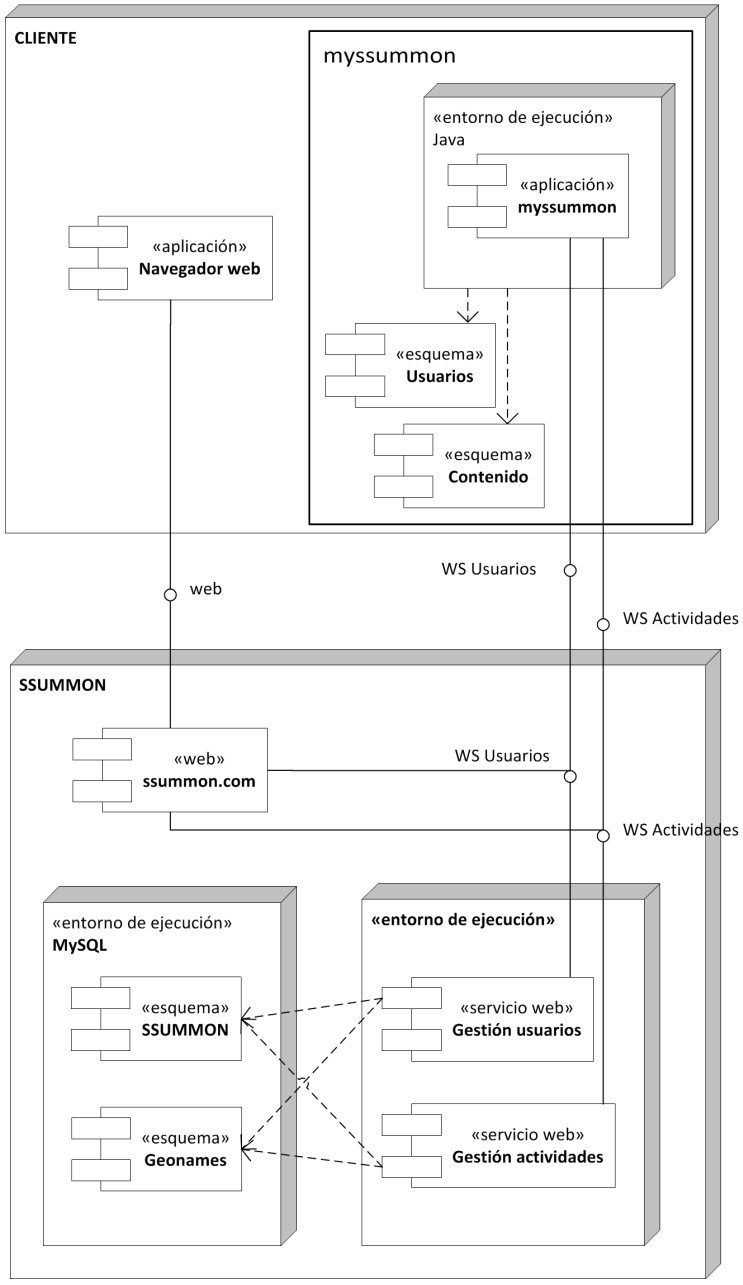


Figura 3.16: Arquitectura del sistema completo.

3.3. Diseño

Tras las primeras reuniones con el equipo de SSUMMON, las funcionalidades de la aplicación quedaron asentadas. En las sucesivas reuniones, se aclararon detalles sobre el diseño de este sistema.

3.3.1. Modelo de datos

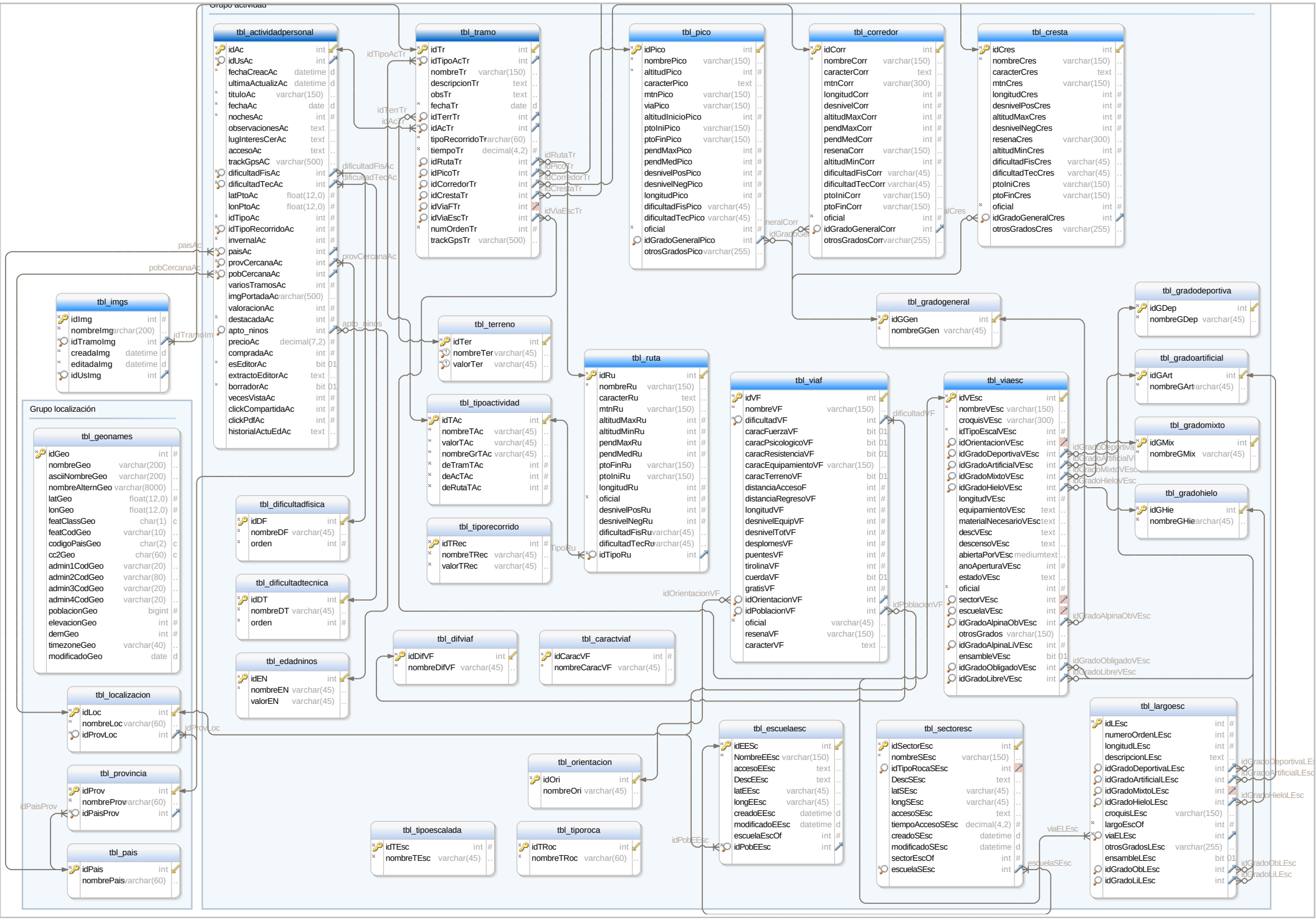
El equipo de SSUMMON manifestó la intención de utilizar el mismo sistema gestor de base de datos, decisión que no prosperaría debido a las ventajas que ofrecían otros (Apéndice B.3.1). También se tenía en mente emplear un esquema de base de datos lo más parecido posible al usado por la aplicación web. Las razones para esto último fueron varias:

- La definición de las actividades suele ser compleja y llena de atributos, a veces muy similares (por ejemplo: existen más de 5 tipos de grado para definir la dificultad en una escalada). El modelo de datos utilizado en SSUMMON está diseñado por expertos en actividades de montaña, lo cual garantiza la correcta definición de estas actividades.
- Muchas de las funciones que se realizan en myssummon son bastante similares a las realizadas en la aplicación web.
- Cuando se pongan en marcha los *web services* en SSUMMON, los objetos manipulados tendrán los mismos campos e iguales nombres.

A pesar de utilizar el mismo modelo de datos, se le han aplicaron ciertas modificaciones:

- Eliminadas las tablas que nunca se van a utilizar.
- Desechados algunos comandos SQL al crear los esquemas para asegurar la compatibilidad con el nuevo SGBD: comentarios y valores autoincrementales. Por ejemplo:
- Separado el único esquema de SSUMMON en dos esquemas: esquema usuarios y esquema contenido. Cada esquema se almacena en un fichero. La estrategia es implementar una vez el esquema de usuarios conteniendo información sobre todos los usuario, y una vez por usuario el esquema de contenido conteniendo todas las actividades pertenecientes a ese usuario.

Habitualmente se muestra el diseño del modelo de datos a través de diagramas entidad-relación. Sin embargo, el utilizar un modelo que ya ha sido implementado no es necesario inferir el diagrama por lo que se muestran directamente los esquemas mediante diagramas relacionales.


Grupo actividad
tbl_actividadpersonal
idAc int
idUsAc int
fechaCreacAc datetime d
ultimaActualizAc datetime d
tituloAc varchar(150)
fechaAc date d
nochesAc int #
observacionesAc text
lugInteresCerAc text
accesoAc text
trackGpsAC varchar(500)
dificultadFisAc int
dificultadTecAc int
latPtoAc float(12,0) #
lonPtoAc float(12,0) #
idTipoAc int #
idTipoRecorridoAc int #
invernalAc int #
paisAc int
provCercanaAc int
pobCercanaAc int
variosTramosAc int #
imgPortadaAc varchar(500)
valoracionAc int #
destacadaAc int #
apto_ninos int
precioAc decimal(7,2) #
compradaAc int #
esEditorAc bit 01
extractoEditorAc text
borradorAc bit 01
vecesVistaAc int #
clickCompartidaAc int #
clickPdfAc int #
historialActuEdAc text
tbl_tramo
idTr int
idTipoAcTr int
nombreTr varchar(150)
descripcionTr text
obsTr text
fechaTr date d
idTerrTr int
idAcTr int
tipoRecorridoTr varchar(60)
tiempoTr decimal(4,2) #
idRutaTr int
idPicoTr int
idCorredorTr int
idCrestaTr int
idViaFTr int
idViaEscTr int
numOrdenTr int #
trackGpsTr varchar(500)
tbl_pico
idPico int
nombrePico varchar(150)
altitudPico int #
caracterPico text
mtnPico varchar(150)
viaPico varchar(150)
altitudInicioPico int #
ptoIniPico varchar(150)
ptoFinPico varchar(150)
pendMaxPico int #
pendMedPico int #
desnivelPosPico int #
desnivelNegPico int #
longitudPico int #
dificultadFisPico varchar(45)
dificultadTecPico varchar(45)
oficial int #
idGradoGeneralPico int
otrosGradosPico varchar(255)
tbl_corredor
idCorr int
nombreCorr varchar(150)
caracterCorr text
mtnCorr varchar(300)
longitudCorr int #
desnivelCorr int #
altitudMaxCorr int #
pendMaxCorr int #
pendMedCorr int #
resenaCorr varchar(150)
altitudMinCorr int #
dificultadFisCorr varchar(45)
dificultadTecCorr varchar(45)
ptoIniCorr varchar(150)
ptoFinCorr varchar(150)
oficial int #
idGradoGeneralCorr int
otrosGradosCorr varchar(255)
tbl_cresta
idCres int
nombreCres varchar(150)
caracterCres text
mtnCres varchar(150)
longitudCres int #
desnivelPosCres int #
altitudMaxCres int #
desnivelNegCres int #
resenaCres varchar(300)
altitudMinCres int #
dificultadFisCres varchar(45)
dificultadTecCres varchar(45)
ptoIniCres varchar(150)
ptoFinCres varchar(150)
oficial int #
idGradoGeneralCres int
otrosGradosCres varchar(255)
tbl_imgs
idImg int #
nombreImg varchar(200)
idTramoImg int
creadaImg datetime d
editadaImg datetime d
idUsImg int
tbl_terreno
idTer int
nombreTer varchar(45)
valorTer varchar(45)
tbl_tipoactividad
idTAc int
nombreTAc varchar(45)
valorTAc varchar(45)
nombreGrTAc varchar(45)
deTramTAc int #
deAcTAc int #
deRutaTAc int #
tbl_ruta
idRu int
nombreRu varchar(150)
caracterRu text
mtnRu varchar(150)
altitudMaxRu int #
altitudMinRu int #
pendMaxRu int #
pendMedRu int #
ptoFinRu varchar(150)
ptoIniRu varchar(150)
longitudRu int #
oficial int #
desnivelPosRu int #
desnivelNegRu int #
dificultadFisRu varchar(45)
dificultadTecRu varchar(45)
idTipoRu int
tbl_gradogeneral
idGGen int
nombreGGen varchar(45)
tbl_gradodeportiva
idGDep int
nombreGDep varchar(45)
tbl_gradoartificial
idGArt int
nombreGArt varchar(45)
tbl_grademixto
idGMix int
nombreGMix varchar(45)
tbl_gradohielo
idGHie int
nombreGHie varchar(45)
tbl_viaf
idVF int
nombreVF varchar(150)
dificultadVF int
caracFuerzaVF bit 01
caracPsicologicoVF bit 01
caracResistenciaVF bit 01
caracEquipamientoVF varchar(150)
caracTerrenoVF bit 01
distanciaAccesoF int #
distanciaRegresoVF int #
longitudVF int #
desnivelEquipVF int #
desnivelTotVF int #
desplomesVF int #
puentesVF int #
tirolinaVF int #
cuerdaVF bit 01
gratisVF int #
idOrientacionVF int
idPoblacionVF int
oficial varchar(45)
resenaVF varchar(150)
caracterVF text
tbl_viaesc
idVEsc int
nombreVEsc varchar(150)
croquisVEsc varchar(300)
idTipoEscalVEsc int #
idOrientacionVEsc int
idGradoDeportivaVEsc int
idGradoArtificialVEsc int
idGradoMixtoVEsc int
idGradoHieloVEsc int
longitudVEsc int #
equipamientoVEsc text
materialNecesarioVEsc text
descVEsc text
descensoVEsc text
abiertaPorVEsc mediumtext
anoAperturaVEsc int #
estadoVEsc text
oficial int #
sectorVEsc int
escuelaVEsc int
idGradoAlpinaObVEsc int
otrosGrados varchar(150)
idGradoAlpinaLiVEsc int #
ensambleVEsc bit 01
idGradoObligadoVEsc int
idGradoLibreVEsc int
Grupo localización
tbl_geonames
idGeo int #
nombreGeo varchar(200)
asciiNombreGeo varchar(200)
nombreAlternGeo varchar(8000)
latGeo float(12,0) #
lonGeo float(12,0) #
featClassGeo char(1) c
featCodGeo varchar(10)
codigoPaisGeo char(2) c
cc2Geo char(60) c
admin1CodGeo varchar(20)
admin2CodGeo varchar(80)
admin3CodGeo varchar(20)
admin4CodGeo varchar(20)
poblacionGeo bigint #
elevacionGeo int #
demGeo int #
timezoneGeo varchar(40)
modificadoGeo date
tbl_dificultadfisica
idDF int
nombreDF varchar(45)
orden int #
tbl_dificultadtecnica
idDT int
nombreDT varchar(45)
orden int #
tbl_edadninos
idEN int
nombreEN varchar(45)
valorEN varchar(45)
tbl_tiporecorrido
idTRec int #
nombreTRec varchar(45)
valorTRec varchar(45)
tbl_difviaf
idDifVF int
nombreDifVF varchar(45)
tbl_caractviaf
idCaracVF int #
nombreCaracVF varchar(45)
tbl_localizacion
idLoc int
nombreLoc varchar(60)
idProvLoc int
tbl_provincia
idProv int
nombreProv varchar(60)
idPaisProv int
tbl_pais
idPais int
nombrePais varchar(60)
tbl_orientacion
idOri int
nombreOri varchar(45)
tbl_escuelaesc
idEESc int
NombreEEsc varchar(150)
accesoEEsc text
DescEEsc text
latEEsc varchar(45)
longEEsc varchar(45)
creadoEEsc datetime d
modificadoEEsc datetime d
escuelaEscOf int #
idPobEEsc int
tbl_sectoresc
idSectorEsc int
nombreSEsc varchar(150)
idTipoRocaSEsc int
DescSEsc text
latSEsc varchar(45)
longSEsc varchar(45)
accesoSEsc text
tiempoAccesoSEsc decimal(4,2) #
creadoSEsc datetime d
modificadoSEsc datetime d
sectorEscOf int #
escuelaSEsc int
tbl_largoesc
idLEsc int #
numeroOrdenLEsc int #
longitudLEsc int #
descripcionLEsc text
idGradoDeportivaLEsc int
idGradoArtificialLEsc int
idGradoMixtoLEsc int
idGradoHieloLEsc int
croquisLEsc varchar(150)
largoEscOf int #
viaELEsc int
otrosGradosLEsc varchar(255)
ensambleLEsc bit 01
idGradoObLEsc int
idGradoLiLEsc int
tbl_tipoescalada
idTEsc int #
nombreTEsc varchar(45)
tbl_tiporoca
idTRoc int
nombreTRoc varchar(60)
idTipoAcTr
idTerrTr
idAcTr
idRutaTr
idPicoTr
idCorredorTr
idCrestaTr
idViaEscTr
dificultadFisAc
dificultadTecAc
paisAc
pobCercanaAc
provCercanaAc
apto_ninos
idTramoImg
idTipoRu
dificultadVF
idOrientacionVF
idPoblacionVF
idGradoAlpinaObVEsc
idGradoObligadoVEsc
idGradoLibreVEsc
idProvLoc
idPaisProv
idPobEEsc
viaELEsc
escuelaSEsc
idGradoDeportivaLEsc
idGradoArtificialLEsc
idGradoHieloLEsc
idGradoObLEsc
idGradoLiLEsc

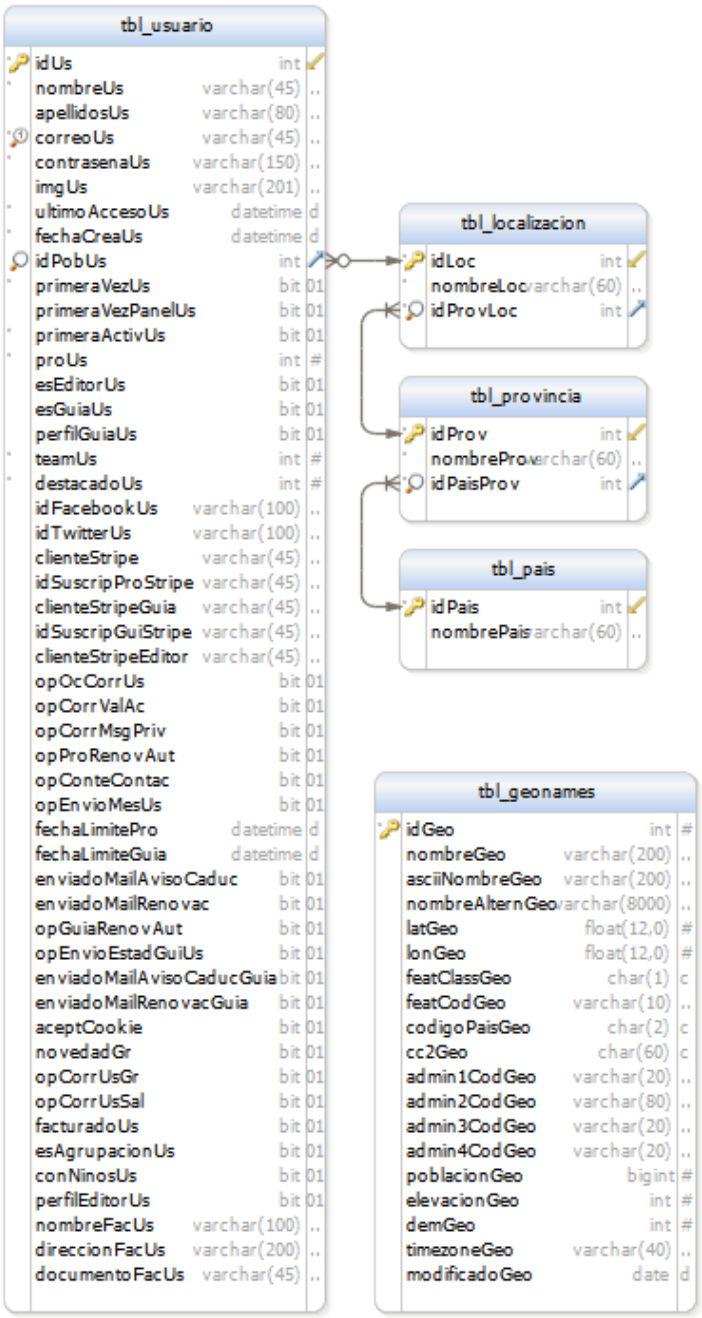


Figura 3.17: Esquema usuarios.

3.3.2. Arquitectura

Durante la fase de análisis se presentó la arquitectura del sistema actual (Apartado 3.2.4). En esta sección se muestra una vista de la arquitectura del sistema en desarrollo.

Se observa en el diagrama de despliegue, Figura 3.18, la interacción de los diferentes módulos de la aplicación entre sí. El módulo de la interfaz visual, GUI, está compuesto de dos conjuntos de artefactos: conjunto Controller y conjunto FXML. Los últimos contienen la estructura de la interfaz en sí, mientras que los primeros contienen la lógica de la aplicación y conectan la interfaz visual con el resto de la aplicación.

Por otro lado, se percibe la conexión del módulo GUI a otros dos módulos: DAO y Files. El módulo DAO, *Data Access Object*, está compuesto de objetos que definen las operaciones en los esquemas de base de datos (esquema Usuarios y esquema Contenido). Éste mismo módulo depende de otro, VO (*Value Object*), el cual lo componen objetos que representan las entidades de la base de datos. El segundo módulo al que está conectado la interfaz visual, Files, define el tratamiento de los ficheros que utiliza el sistema (incluyendo los esquemas de base de datos)[6]. Como nota aclaratoria, el rectángulo de puntos englobando los últimos cuatro artefactos indica que éstos pertenecen al *entorno personal* de los usuarios (nótese la multiplicidad de estos).

[6]Tal y como se define en la arquitectura de la fase de análisis, Apartado 3.2.4, la base de datos está contenida en dos ficheros ordinarios.

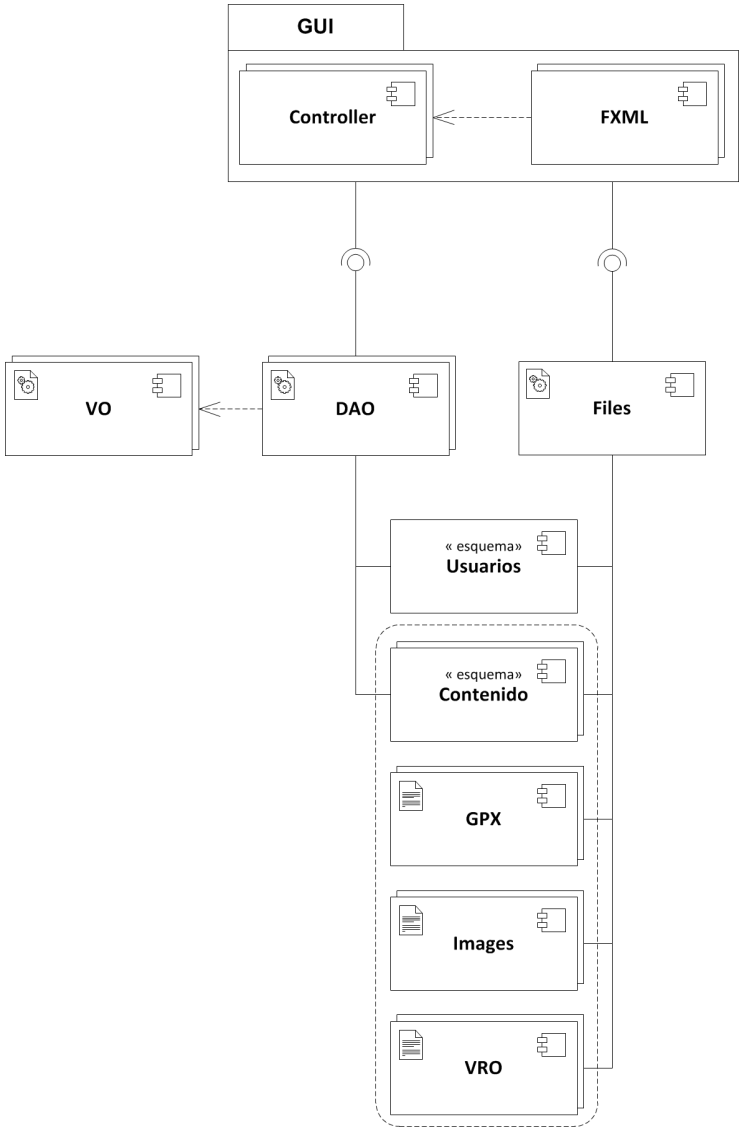


Figura 3.18: Arquitectura del sistema en desarrollo.

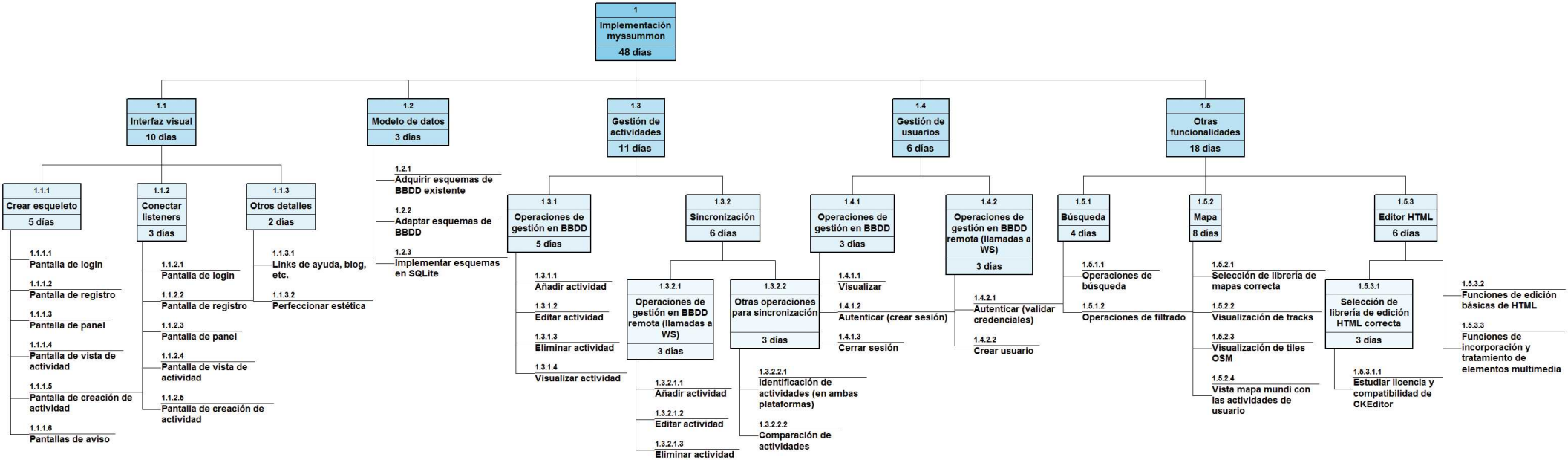
3.4. Implementación de versión demostración

3.4.1. Descomposición del trabajo

Como paso previo a la implementación de la versión de demostración, se vio la necesidad de hacer una descomposición de las tareas a ejecutar para implementar el sistema completo. Esto permitiría distinguir ambas versiones, conocer qué funciones se pondrían en marcha en la versión de demostración y, lo más importante, ayudaría a tener una estimación del coste en días que suma implementar el sistema completo.

En el diagrama EDT (Estructura de Descomposición de Trabajo) de la página siguiente se muestran todas las tareas a realizar para poner en funcionamiento el sistema completo. Un subconjunto de todas ellas son las que forman parte de la versión de demostración:

- **1.1 Interfaz visual:** Se realizan las tareas "1.1.1 Crear esqueleto" y "1.1.2 Conectar listeners". Ambas son esenciales.
- **1.2 Modelo de datos:** Se realiza todas las tareas. También son tareas cruciales.
- **1.3 Gestión de actividades:** Se realiza la tarea "1.3.1 Operaciones de gestión en BBDD". Es la tarea básica en la gestión de actividades. No obstante, no ha sido posible implementar la tarea de "1.3.2 Sincronización" ya que aun están disponibles los *web services* de SSUMMON.
- **1.4 Gestión de usuarios:** Se realiza la tarea "1.4.1 Operaciones de gestión en BBDD". Igual que en la gestión de actividades, la tarea "1.4.2 Operaciones de gestión en BBDD remota" dependen de la disponibilidad de los *web services* de SSUMMON.
- **1.5 Otras funcionalidades:** No se realiza ninguna de estas tareas. Sin embargo, éstas son tareas complementarias y no forman parte de la base del sistema. El resto de la aplicación no depende de ellas y podrán ser incorporadas sin dificultad.


1
Implementación myssummon
48 días
1.1
Interfaz visual
10 días
1.2
Modelo de datos
3 días
1.3
Gestión de actividades
11 días
1.4
Gestión de usuarios
6 días
1.5
Otras funcionalidades
18 días
1.1.1
Crear esqueleto
5 días
1.1.2
Conectar listeners
3 días
1.1.3
Otros detalles
2 días
1.1.1.1
Pantalla de login
1.1.1.2
Pantalla de registro
1.1.1.3
Pantalla de panel
1.1.1.4
Pantalla de vista de actividad
1.1.1.5
Pantalla de creación de actividad
1.1.1.6
Pantallas de aviso
1.1.2.1
Pantalla de login
1.1.2.2
Pantalla de registro
1.1.2.3
Pantalla de panel
1.1.2.4
Pantalla de vista de actividad
1.1.2.5
Pantalla de creación de actividad
1.1.3.1
Links de ayuda, blog, etc.
1.1.3.2
Perfeccionar estética
1.2.1
Adquirir esquemas de BBDD existente
1.2.2
Adaptar esquemas de BBDD
1.2.3
Implementar esquemas en SQLite
1.3.1
Operaciones de gestión en BBDD
5 días
1.3.2
Sincronización
6 días
1.3.1.1
Añadir actividad
1.3.1.2
Editar actividad
1.3.1.3
Eliminar actividad
1.3.1.4
Visualizar actividad
1.3.2.1
Operaciones de gestión en BBDD remota (llamadas a WS)
3 días
1.3.2.2
Otras operaciones para sincronización
3 días
1.3.2.1.1
Añadir actividad
1.3.2.1.2
Editar actividad
1.3.2.1.3
Eliminar actividad
1.3.2.2.1
Identificación de actividades (en ambas plataformas)
1.3.2.2.2
Comparación de actividades
1.4.1
Operaciones de gestión en BBDD
3 días
1.4.2
Operaciones de gestión en BBDD remota (llamadas a WS)
3 días
1.4.1.1
Visualizar
1.4.1.2
Autenticar (crear sesión)
1.4.1.3
Cerrar sesión
1.4.2.1
Autenticar (validar credenciales)
1.4.2.2
Crear usuario
1.5.1
Búsqueda
4 días
1.5.2
Mapa
8 días
1.5.3
Editor HTML
6 días
1.5.1.1
Operaciones de búsqueda
1.5.1.2
Operaciones de filtrado
1.5.2.1
Selección de librería de mapas correcta
1.5.2.2
Visualización de tracks
1.5.2.3
Visualización de tiles OSM
1.5.2.4
Vista mapa mundi con las actividades de usuario
1.5.3.1
Selección de librería de edición HTML correcta
3 días
1.5.3.2
Funciones de edición básicas de HTML
1.5.3.3
Funciones de incorporación y tratamiento de elementos multimedia
1.5.3.1.1
Estudiar licencia y compatibilidad de CKEditor

3.4.2. Paquetes

En Figura 3.19 se muestra un diagrama de paquetes de ésta implementación. En él aparecen todos los paquetes y las clases más importantes. Este diagrama se asemeja bastante a la arquitectura mostrada en la fase de diseño (Figura 3.18) por los pocos imprevistos que ha habido en la codificación.

Como se ve en la figura, hay cuatro paquetes principales:

- `gui`: En este paquete está contenida la interfaz visual. Está compuesta a su vez por dos paquetes:
  - `utils`: Contiene las clases necesarias para manipular la interfaz visual. La clase `Presentador.java` se encarga del alternado entre pantallas, mostrado de diálogos e inicialización de las ventanas. Ésta depende de `Cargador.java`, la cual carga los ficheros que definen la estructura de las pantalla o de parte de ellas. Por último, la clase `Elemento.java` es un *JavaBean*[7] que contiene la estructura y lógica de negocio de cada pantalla.
  - `controllers`: Este paquete está compuesto por los *controladores*, clases que encargan de la lógica de negocio que hay detrás de una pantalla y manipulan el contenido de ésta. Los controladores de las pantallas están albergados en el paquete `main`. El resto de paquetes contienen los controladores de parte de estas pantallas.

  En el paquete también hay clases excepciones encargadas de los errores al tratar los datos de los formularios.
- `dal`: El nombre de este paquete es acrónimo de *Data Access Layer*. Lo forman dos paquetes:
  - `dao`: Define tanto la configuración del acceso a los esquemas de base de datos como las operaciones que se realizan en ella.
  - `vo`: Contiene *JavaBeans* que definen las entidades de la base de datos.

  Ambos paquetes contienen tantas clases como entidades tiene la base de datos.

  También forman parte de este paquete dos clases, las cuales tratan la configuración del acceso a cada uno de los esquemas de la base de datos.
- `files`: Se encarga de todo lo relativo a los ficheros y su tratamiento. La clase `EspacioPersonal.java` genera un espacio personal para cada usuario de la aplicación. Esta clase depende de `OperacionesFicheros.java` que define todas las operaciones que la aplicación efectúa sobre ficheros y directorios. También depende de la clase `Rutas.java` la cual gestiona la ruta a cada fichero y directorio que la aplicación tenga que utilizar.

[7] Un *JavaBean* es una clase utilizada para encapsular objetos más simples.

- `utils`: El paquete alberga las clases cuya funcionalidad no sea lo suficientemente fuerte como forman su propio paquete. Actualmente, una clase forma parte de él, `Props.java`, la cual es la encargada de tratar las operaciones de lectura sobre el fichero de configuración[8] (`conf.properties`).

Además de los paquetes mencionados arriba, existe una clase principal, `Main.java`. Ésta, es ejecutada al iniciar la aplicación y se encarga de inicializar los componentes e invocar la interfaz visual.

[8]Este tipo de ficheros están compuesto por pares *clave-valor* que definen opciones importantes de la configuración de la aplicación. Añaden un grado de modularidad a las aplicaciones Java.

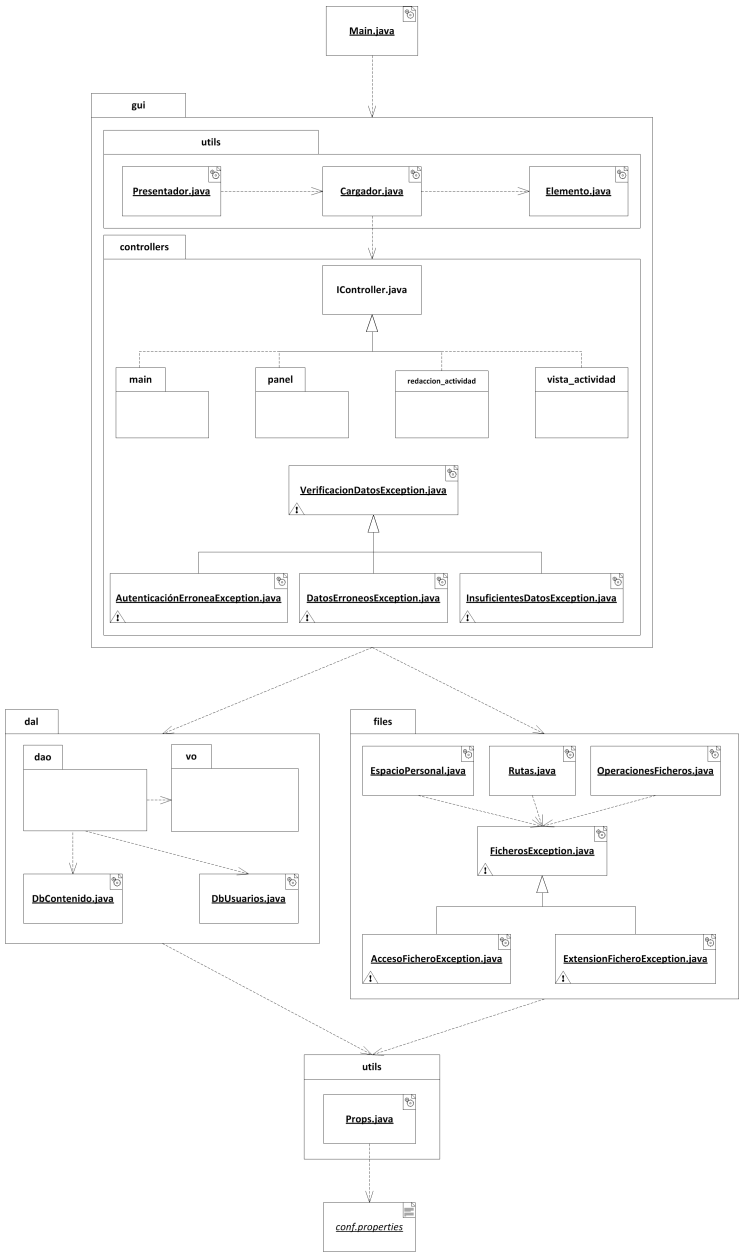


Figura 3.19: Paquetes de la versión de demostración.

Capítulo 4

Conclusiones y trabajo futuro

Este capítulo está dedicado a expresar las conclusiones sobre ambas partes así como las líneas de futuro que pueden seguirse.

4.1. Conclusiones

En este apartado se hace un breve repaso a los resultados obtenidos en la construcción de los dos sistemas.

Conclusiones de SiRAM
Los resultados de los experimentos dejan entrever un mejor funcionamiento de la aproximación del filtro colaborativo, lo cual entraba dentro de lo que se podía esperar dado que normalmente este método se comporta mejor cuando el experimento se basa en predecir evaluaciones, el conjunto de datos no es excesivamente grande y los usuarios tienen un comportamiento similar y estable. Las aproximaciones basadas en contenido se comportan normalmente mejor cuando el tipo de experimento es de predicción de los elementos más adecuados dado un usuario concreto o las evaluaciones no están tan bien distribuidas como en nuestro conjunto de datos de prueba, por lo que no hay que descartar su uso para el nuevo diseño híbrido. Aun así, el buen resultado en lo referente a RMSE del filtro colaborativo anima a darle un peso destacado en el desarrollo que se está llevando a cabo.

En tono más personal, los sistemas de recomendación han cambiado bastante mi perspectiva sobre ellos y sobre el aprendizaje automático. Todo el conocimiento que tenía acerca de éstos fue adquirido como parte de una asignatura de la carrera, lo cual ha supuesto un inconveniente por el tiempo que se ha tenido que dedicar a su estudio. No obstante, trabajando con ellos me he dado cuenta de su verdadero potencial y la gran variedad de técnicas que utilizan.

Conferencia CIMA 2015
CIMA 2015 es la primera edición del Congreso Internacional de Montañismo. Se celebró los días 26, 27 y 28 de Marzo en Zaragoza.

Aprovechando el trabajo que se estaba haciendo en éste ámbito, se redactó un artículo para dar a conocer SiRAM bajo una aproximación más introductoria y menos técnica al público que más podía interesar: los montañistas. Este artículo, producto del ponente, directores y autor de este PFC, se exhibió a través de una comunicación oral, la cual tuvo buena acogida.

La participación en el congreso alteró la planificación que se trazó al comienzo del PFC, pues no fuimos conscientes de la existencia de la conferencia hasta pocas semanas previas a la fecha de entrega de las comunicaciones.

El artículo está pendiente de publicación.

Conclusiones de myssummon
A lo largo del capítulo de myssummon, se ha mostrado el desarrollo de un sistema software de media escala. Se ha estudiado el contexto en el que debe funcionar, los requisitos que debe cumplir y se ha creado un diseño completo acorde a ello. También se ha realizado una implementación de demostración que, aunque no implementa todas las funciones del diseño, responde como se esperaba. Por todo esto, la sensación es positiva.

4.2. Trabajo futuro

Como se ha visto en la sección anterior, sendas partes han ofrecido resultados prometedores. Ésto anima a continuar el desarrollo de los sistemas, sabiendo que las líneas futuras son abundantes.

Líneas futuras de SiRAM
Como trabajos futuros se piensa introducir el uso de la semántica y la minería de textos para explotar la información existente en las descripciones de las rutas tal y como se plantea en otros trabajos como [10] para así comprobar su influencia y balancear los pesos de cada uno de los métodos en el sistema híbrido.

Asimismo, se quiere continuar experimentando con las otras aproximaciones que ya forman parte del diseño de SiRAM: la metodología basada en conocimiento y el análisis del contexto.

Por otro lado, hay estudios que desmuestran una mayor efectividad de otras técnicas más complejas como *Content-boosted Collaborative Filtering* frente al filtrado colaborativo y el basado en contenido [7]. Sería interesante comprobar si esas mejoras también se dan en el ámbito de las actividades al aire libre.

La aspiración final es convertir a SIRAM en una herramienta útil para la sociedad que permita promover las actividades de montaña de forma fiable y segura entre la comunidad de usuarios de Internet que acceden a la Web buscando este tipo de información.

Líneas futuras de myssummon

Existen varios caminos por los que este sistema software puede continuar. Quizás el más evidente es implementar una versión completamente funcional. Esto no debería suponer un gran esfuerzo ya que la versión de demostración establece una base sólida sobre la que incluir el resto de las funcionalidades sin necesidad de reestructurar el código para poder hacerlo.

Otra labor pendiente es mejorar el tratamiento de los errores y la seguridad del software. Esto implica llevar a cabo dos tareas: *a)* aumentar el conjunto de las excepciones existentes y mejorar su tratamiento, y *b)* incluir una batería de test unitarios que ayuden a construir nuevas funcionalidades sin correr el riesgo de destrozar las existentes.

Apéndice A

SiRAM: Sistema de recomendación de actividades de montaña

A.1. Análisis de alternativas de entornos de experimentación

Esta sección recoge el análisis de las opciones valoradas a la hora de escoger un entorno para la ejecución de los experimentos. Las alternativas que se contemplan son: *LensKit*, *Apache Mahout*, *R*, *Python*. También se valoró el uso de *MyMedia toolkit*, pero no se ha incluido en este análisis por ser un *framework* basado en tecnología .net, con la cual no se ha tenido experiencia previa en absoluto.

De las cuatro alternativas, las dos primeras son soluciones específicas para aprendizaje automático. Las dos últimas son lenguajes de programación con un conjunto muy rico de librerías matemáticas y de aprendizaje automático.

LensKit
LensKit es una plataforma de experimentación de sistemas de recomendación implementado en Java [2]. Destaca por por su modularidad, su documentación y su implementación limpia. Esto es importante pues es una plataforma orientada a la investigación más que a la puesta en marcha de sistemas en producción. Estas son sus principales ventajas e inconvenientes:

+ Es una plataforma centrada en la **investigación** de sistemas de recomendación.

– Está implementado en Java. Esto supone una pérdida de agilidad al experimentar por ser un lenguaje compilado y no tratarse de un entorno interactivo.

- No soporta las aproximaciones basadas en contenido ni los sistemas híbridos.
- No soporta la técnica de validación cruzada (*cross-validation*), una características importante para evitar azar altere el resultado de los experimentos.

Apache Mahout
Apache Mahout es un *framework* para Java que apoya el desarrollo de sistemas de recomendación [12]. A diferencia de LensKit, es una plataforma orientada a la puesta en producción de estos sistemas. Una muestra de esto es la compatibilidad con Hadoop, una plataforma de la fundación Apache, dedicada a la computación distribuida.

- Sufre de la misma desventaja que LensKit: su implementación en Java le hace perder agilidad si se utiliza en entornos de experimentación.
- Centrado en la puesta en marcha de sistemas de recomendación en entornos de alto rendimiento, más que en la experimentación de éstos.

Entorno R
R es un lenguaje de programación para el análisis estadístico [8]. Se ha considerado en este análisis de alternativas por

+ Es un lenguaje dedicado al análisis estadístico.
+ Conjunto enorme de librerías.
+ Existen diferentes entornos interactivos para él (por ejemplo: *RStudio*, *Tinn-R* o *StatET*)
- Debido a su sintaxis poco habitual, su curva de aprendizaje es bastante lenta.
- Ofrece mal rendimiento en bucles, creando la necesidad de vectorizar el código desarrollado.

Entorno Python
Python es un lenguaje de programación interpretado, de propósito general y uno de los más populares en computación científica. El entorno de Python estaría formado por las librerías: *numpy* y *scipy*, librerías de cálculo numérico; *pandas*, librería que permite el uso de *Dataframes* para un tratamiento más cómodo de los conjuntos de datos; y *scikit-learn*, paquete con una amplia batería de funciones de aprendizaje automático.

También se podría añadir al conjunto de librerías, IPython Notebook, un entorno interactivo que combina código, texto y gráficos. Muy útil para reproducir la ejecución de experimentos.

+ Conjunto muy amplio de funciones matemáticas y de algoritmos de aprendizaje automático (incluyendo el tratamiento previo y posterior de los datos).

+ Además de ser un lenguaje utilizado en el entorno académico, es un referente en la industria. Sería una solución válida si se quisiera implementar el sistema en un entorno real.

+ Ejecución ágil de los experimentos mediante el entorno interactivo IPython Notebook.

+ Experiencia previa en el uso del lenguaje.

Finalmente, es el entorno de Python el que más convence como alternativa para experimentar con SiRAM. Sus grandes ventajas y nulos inconvenientes hacen que Python sea la opción idónea.

A.2. Experimentos

En esta sección se muestran los conjuntos de datos utilizados durante los experimentos en SiRAM (incluyendo la codificación numérica de los campos) y el código desarrollado propios de estos experimentos.

A.2.1. *Datasets*

NUM	NOCHES	RECORRIDO	NIÑOS	TIPO ACTIVIDAD	DIFICULTAD FÍSICA	DIFICULTAD TÉCNICA	TIEMPO (min)	LONGITUD (m)	TERRENO
1	0	Ida y vuelta	Bebes	Pico	Baja	Poca	45	2000	
2	0	Circular	16	Pico	Baja	Poca	120	4600	
3	0	Circular	No	Senderismo	Baja	Poca	291	11600	Mixto
4	0	Circular	16	Senderismo	Baja	Poca	447	20540	
5	0	Travesía	Bebes	Senderismo	Baja	Poca		2110	
6	0	Circular	12	Senderismo	Baja	Poca		10500	
7	0	Circular	12	Senderismo	Baja	Poca	235		
8	0	Circular	No	Senderismo	Normal	Normal	200	11290	Varios
9	0	Circular	No	Senderismo	Normal	Normal	336	12690	Varios
10	0	Circular	No	Senderismo	Normal	Normal	296	10640	Varios
11	0	Circular	No	Pico	Normal	Normal			
12	0	Circular	16	Pico	Normal	Normal			
13	0	Circular	16	Pico	Normal	Normal	360	13000	Nieve
14	0	Circular	No	Pico	Normal	Normal	450		Nieve
15	1	Ida y vuelta	No	Senderismo	Normal	Normal	240	6000	Mixto
16	1	Ida y vuelta	No	Corredor	Exigente	Normal	180	4000	Nieve
17	1	Ida y vuelta	No	Pico	Baja	Poca	30		Mixto
18	1	Ida y vuelta	No	Travesía	Baja	Poca	180	9000	Mixto
19	1	Ida y vuelta	No	Aproximación			240	6000	Nieve
20	1	Ida y vuelta	No	Pico	Exigente	Normal	180		Nieve
21	1	Ida y vuelta	No	Travesía			360	8000	Nieve
22	0	Ida y vuelta	No	Senderismo	Baja	Poca	50		Sendero
23	0	Ida y vuelta	No	Aproximación	Baja	Poca	120		Varios
24	0	Ida y vuelta	No	Pico	Normal	Normal	40		Pedrera
25	0	Circular	No	Corredor	Exigente	Alta	720	600	Mixto
26	0	Circular	No	Cresta	Exigente	Alta	960	20000	Varios
27	0	Circular	No	Cresta	Normal	Normal	300		
28	0	Circular	No	Aproximación	Baja	Poca			Otros
29	0	Circular	No	Pico	Exigente	Alta	745	19690	

Tabla A.1: *Dataset* de actividades.

NUM	NOCHES	RECORRIDO	NIÑOS	TIPO ACTIVIDAD	DIFICULTAD FÍSICA	DIFICULTAD TÉCNICA	TIEMPO (min)	LONGITUD (m)	TERRENO
1	0	1	0	5	0	0	45	2000	3
2	0	2	16	5	0	0	120	4600	3
3	0	2	18	0	0	0	291	11600	2
4	0	2	16	0	0	0	447	20540	3
5	0	0	0	0	0	0	297	2110	3
6	0	2	12	0	0	0	297	10500	3
7	0	2	12	0	0	0	235	9570	3
8	0	2	18	0	1	1	200	11290	3
9	0	2	18	0	1	1	336	12690	3
10	0	2	18	0	1	1	296	10640	3
11	0	2	18	5	1	1	297	9570	3
12	0	2	16	5	1	1	297	9570	3
13	0	2	16	5	1	1	360	13000	4
14	0	2	18	5	1	1	450	9570	4
15	1	1	18	0	1	1	240	6000	2
16	1	1	18	3	2	1	180	4000	4
17	1	1	18	5	0	0	30	9570	2
18	1	1	18	2	0	0	180	9000	2
19	1	1	18	1	1	1	240	6000	4
20	1	1	18	5	2	1	180	9570	4
21	1	1	18	2	1	1	360	8000	4
22	0	1	18	0	0	0	50	9570	0
23	0	1	18	1	0	0	120	9570	3
24	0	1	18	5	1	1	40	9570	1
25	0	2	18	3	2	2	720	600	2
26	0	2	18	4	2	2	960	20000	3
27	0	2	18	4	1	1	300	9570	3
28	0	2	18	1	0	0	297	9570	5
29	0	2	18	5	2	2	745	19690	3

Tabla A.2: *Dataset* de actividades codificado.

ID	USUARIOS														
ACT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		5	4	5			3		3						
2	4	5	3	4	5	3	3	3							
3		3	3	4	4						5				
4	5	5	5	5	5				3		4				
5	5		4	5	4		3	4				4			
6	5	5	5	5	4	3									
7	5	5		5	5				5	5					3
8			1	2		4	5	5	3	5	4				5
9			3		3	5	4		4	4			5		
1						5	4	4	3	4		5			5
11						5	5	5	4	4				5	
12					3	3	5	5	4	4				4	
13		3				4	5	4	4	5	4		5		
14			3	3		4	5	5	4	3		5	4		5
15			1		2	3					3	5	4	4	4
16	1			2		4		3		4	4	4	4	5	5
17	2	1				4	4	3	5		5	4	5	5	4
18		2			1		4		3	4	5	5	3	5	5
19	2										4	5	5	5	4
20		1	1						3		4	4			5
21	2										4	4	3	3	3
22					1		4					4	5	4	5
23		1		1			3				4	3	5	4	4
24								4			5	4	4		5
25										4	4	4	3	4	4
26					2	3	2	5			4	5	5		5
27		2									3	3	4	5	3
28			2				3	5			5	5	4	5	
29			1						5		5	5	4	4	5

Tabla A.3: *Dataset* de valoraciones.

CAMPO	VALOR NATURAL	VALOR CODIF.
Tipo Actividad	Senderismo	0
	Carrera montaña	1
	BTT	2
	Esquí de montaña	3
	Corredor	4
	Cresta	5
	Pico	6
Dif. técnica	Poca	0
	Normal	1
	Alta	2
Dif. física	Baja	0
	Normal	1
	Exigente	2
Terreno	Sendero	0
	Pista	1
	Roca	2
	Mixto	3
	Varios	4
	Nieve	5
	Hielo	6
Recorrido	Travesía	0
	Ida y vuelta	1
	Circular	2
Niños	Bebes	0
	No	18

Tabla A.4: Codificación de valores de actividades.

En Tabla A.4 se muestran los valores con los que se han codificado los *datasets*. Estos valores no son arbitrarios, si no que representan la dificultad de cada atributo de manera ascendente.

A.2.2. Algoritmos

```
import numpy as np
from scipy.spatial.distance import cosine
import pandas

class Collaborative:

  def __init__(self, ratings_train,
     ratings_test):
    self.ratings_train = pandas.DataFrame(
       ratings_train)
    self.ratings_test = pandas.DataFrame(
       ratings_test)
    self.weights = self.compute_users_weights()

  ###########
  # TRAINING
  ###########

  def cosine_similarity(self, u1_idx, u2_idx):
    # cosine_similarity = 1 - cosine_distance
    return (1 - cosine(self.ratings_train[
       u1_idx], self.ratings_train[u2_idx]))

  def compute_user_weights(self, u_idx):
    weights = []

    for i in self.ratings_train.columns:
      weights.append(self.cosine_similarity(
         u_idx, i))

    return weights

  def compute_users_weights(self):
    weights = []

    for i in range(len(self.ratings_train.
       columns)):
      weights.append(self.compute_user_weights(
         i))

    return pandas.DataFrame(weights)

  ############
```

```
  # PREDICTING
  ############

  def predict(self, u_idx, i_idx):
    weighted_ratings = np.dot(self.ratings_test
       .iloc[i_idx], self.weights.iloc[u_idx])
    weights_sum = np.sum(self.weights.iloc[
       u_idx][self.ratings_test.iloc[i_idx] !=
       0])     # Exclude non-contributing weights
        from the sum

    return (weighted_ratings / weights_sum)
```

```
import numpy as np
from scipy.spatial.distance import cosine
import pandas

class Content:

  def __init__(self, acts_train, ratings_train,
      acts_test, ratings_test):
    self.acts_train = pandas.DataFrame(
       acts_train)
    self.ratings_train = pandas.DataFrame(
       ratings_train)
    self.acts_test = pandas.DataFrame(acts_test
       )
    self.ratings_test = pandas.DataFrame(
       ratings_test)
    self.users = self.compute_users_vectors()

  ###########
  # TRAINING
  ###########

  def compute_user_vector(self, u_idx):
    return np.average(self.acts_train, axis=0,
       weights=self.ratings_train[u_idx])

  def compute_users_vectors(self):
    users = []

    for i in range(len(self.ratings_train.
       columns)):
      users.append(self.compute_user_vector(i))
```

```
    return pandas.DataFrame(users)

  ############
  # PREDICTING
  ############

  def cosine_similarity(self, user_idx, act_idx
     ):
    # cosine_similarity = 1 - cosine_distance
    return (1 - cosine(self.users.iloc[user_idx
       ], self.acts_test.iloc[act_idx]))

  def cos_to_rate(self, cos):
    return (cos + 2) / 3 * 5

  def predict(self, u_idx, i_idx):
    return self.cos_to_rate(self.
       cosine_similarity(u_idx, i_idx))
```

```
import numpy as np
import pandas
import scipy

from Collaborative import Collaborative
from Content import Content

#CARGANDO DATOS

### LECTURA DE DATASETS
acts = pandas.read_csv("acts_enc.csv", usecols=
    range(1,10))
rats = pandas.read_csv("trans2.csv", usecols=
   range(1,16), header=0, names=range(0,15))
acts_norm = (acts - acts.mean()) / (acts.max()
   - acts.min())

### SPLITTING DATASETS (TRAINING, TESTING)
ss = sklearn.cross_validation.ShuffleSplit(28,
   test_size=0.16)

i = 0

for train_idx, test_idx in ss:
  acts_train = acts_norm.iloc[train_idx]
```

```
acts_test = acts_norm.iloc[test_idx]
rats_train = rats.iloc[train_idx]
rats_test = rats.iloc[test_idx]

col = Collaborative(rats_train, rats_test)
con = Content(acts_train, rats_train,
   acts_test, rats_test)

rats_cf = pandas.DataFrame(np.zeros((5,15)))
rats_cb = pandas.DataFrame(np.zeros((5,15)))

for c in range(len(rats_test.columns)):
  for r in range(len(rats_test.index)):
    rats_cb[c].iloc[r] = con.predict(c,r)
    rats_cf[c].iloc[r] = col.predict(c,r)

    rats_cb.to_excel("resultados/
       resultados_cb" + str(i) + ".xlsx")"
    rats_cf.to_excel("resultados/
       resultados_cf" + str(i) + ".xlsx")"
    rats_test.to_excel("resultados/
       resultados_test" + str(i) + ".xlsx")

    i += 1
```

Apéndice B

myssummon: Sistema gestor de actividades de montaña

B.1. Contexto

En este apartado se exponen los resultados de analizar los procesos del antiguo sistema que también sean utilizados en el nuevo, así como las carencias y necesidades detectadas de cara a diseñar el nuevo.

B.1.1. Procesos y operaciones comunes

Antes de analizar en detalle cada uno de los procesos involucrados en ambos sistemas, se enumeran todos ellos:

- Autenticación.
- Creación de actividades.
- Modificación de actividades.
- Eliminación de actividades.
- Visualización de actividades.
- Búsqueda simple.
- Búsqueda avanzada.

Existen además otros procesos pertenecientes al sistema actual (aplicación web) que no se analizan en este apartado porque no son implementados en el nuevo sistema (Apartado 3.1.1) como la gestión de grupos, de salidas y todo lo relativo con el aspecto social. Del mismo modo, hay procesos adicionales en el nuevo sistema no expuestos en el apartado por pertenecer exclusivamente a este.

A continuación se describen y analizan todos y cada uno de los procesos de la lista anterior.

Autenticación Mediante este proceso el sistema conoce que usuario lo está utilizando. Para poder utilizar cualquier funcionalidad o realizar cualquier operación es necesario que primero el usuario se autentique en él utilizando sus credenciales, compuestos por el correo y la contraseña que el usuario proporcionó al registrarse. Dado que el usuario no podrá registrarte en el sistema desde la aplicación de escritorio, este proceso no se detalla.

Como se aprecia en Figura B.1, el usuario puede recuperar su contraseña desde la misma pantalla de autenticación.

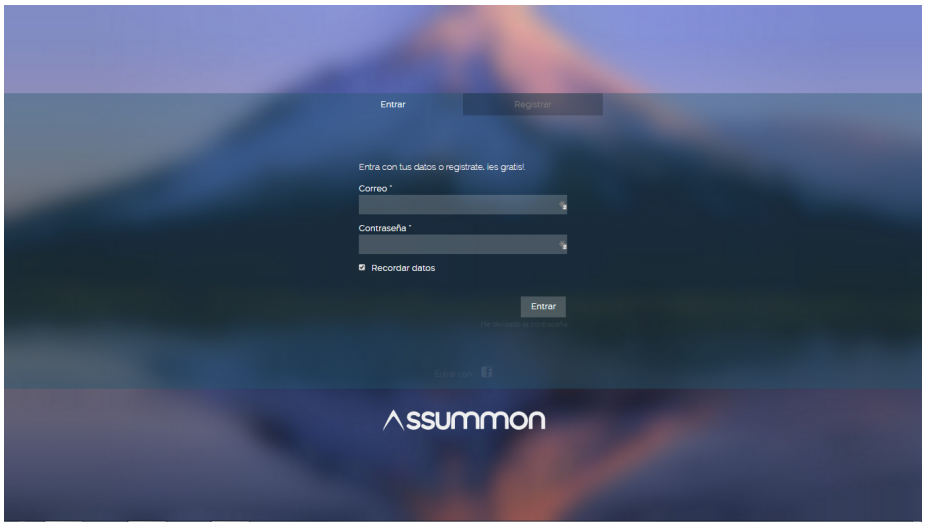


Figura B.1: Autenticación del usuario.

Si introducimos un usuario que no está en el sistema, éste mostrará un error diciendo que el usuario es erróneo (Figura B.2).

Si conseguimos autenticarnos correctamente en el sistema, nos redireccionará a la pantalla principal desde la que podemos ver un resumen de las últimas novedades (actividades, usuarios, grupos, etc).

Creación de actividad En SSUMMON los dos elementos centrales son los usuarios y las actividades. En la aplicación de escritorio las actividades cobran aún más importancia ya que no está presente la parte social.

Con la creación de una nueva actividad se pretende lograr que el usuario relate una experiencia única relacionada con el montañismo en la que ha llevado a cabo alguno de los tipos de actividad que admite SSUMMON. El usuario puede extenderse tanto como desee. Además, puede proporcionar las imágenes que haya tomado in-situ, vídeos externos, tracks de la ruta, etc, con el fin de enriquecer y hacer más comprensible el relato.

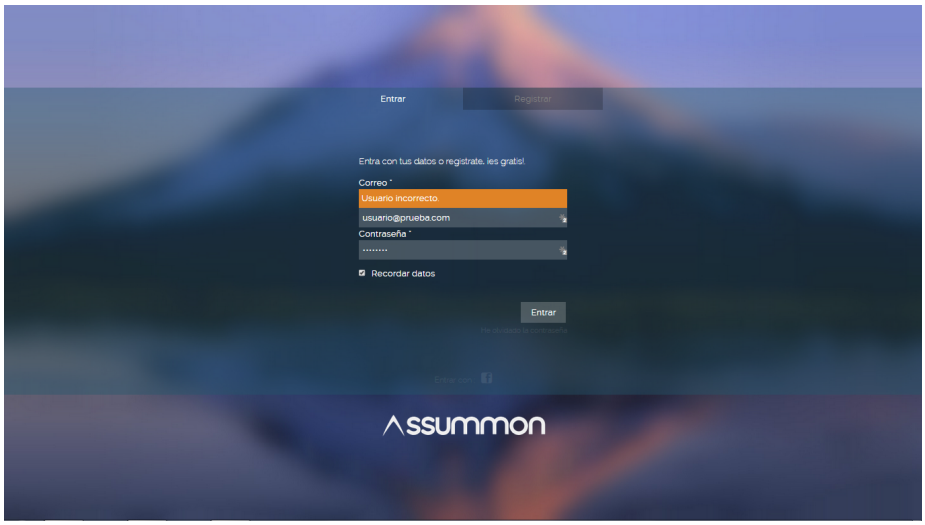


Figura B.2: Error de autenticación.

Como se aprecia en Figura B.3, desde la pantalla principal se puede crear una nueva actividad (desde el botón rojo). Alternativamente, se puede crear una nueva actividad desde cualquier página de la aplicación web a través del menú superior (Figura B.4).

A la hora de crear una nueva actividad, igual que en muchos otros procesos en SSUMMON, el sistema va solicitando en pasos los datos que el usuario ha de introducir. Esto es así para poder facilitar la tarea al usuario teniendo en cuenta que cada actividad puede contener una gran cantidad de detalles.

En el primer paso (Figura B.5) se solicitan atributos generales como el tipo de actividad, tipo de recorrido, la fecha y localidad en la que se llevó a cabo, además de las noches pernoctadas y si la actividad puede ser realizada con niños.

Se debe prestar especial atención a la forma de introducir el parámetro de localización. El sistema valida el texto que el usuario escribe y comprueba que realmente se trate de una localización existente. El usuario deberá después escoger de una lista de localidades aproximadas (Figura B.6) la que desee. A mayor precisión al insertarla, más arriba aparecerá en la lista la localidad. Si se tiene problemas utilizando este mecanismo, se puede escoger la localidad jerárquicamente (Figura B.7).

Es en el segundo paso (Figura B.8) en el que el usuario debe narrar la actividad. La aplicación utiliza un editor de texto con el que poder dar formato, vincular vídeos, etc. También se tienen que determinar el título de la actividad y otros parámetros que varían en función del tipo de actividad que se está creando. Aparte de estos parámetros obligatorios, es posible aportar otros de carácter optativo y variable respecto al tipo de actividad. más adelante se ve una captura de este paso.

Los *usuarios pro* cuentan con una gran ventaja en este proceso. Este tipo

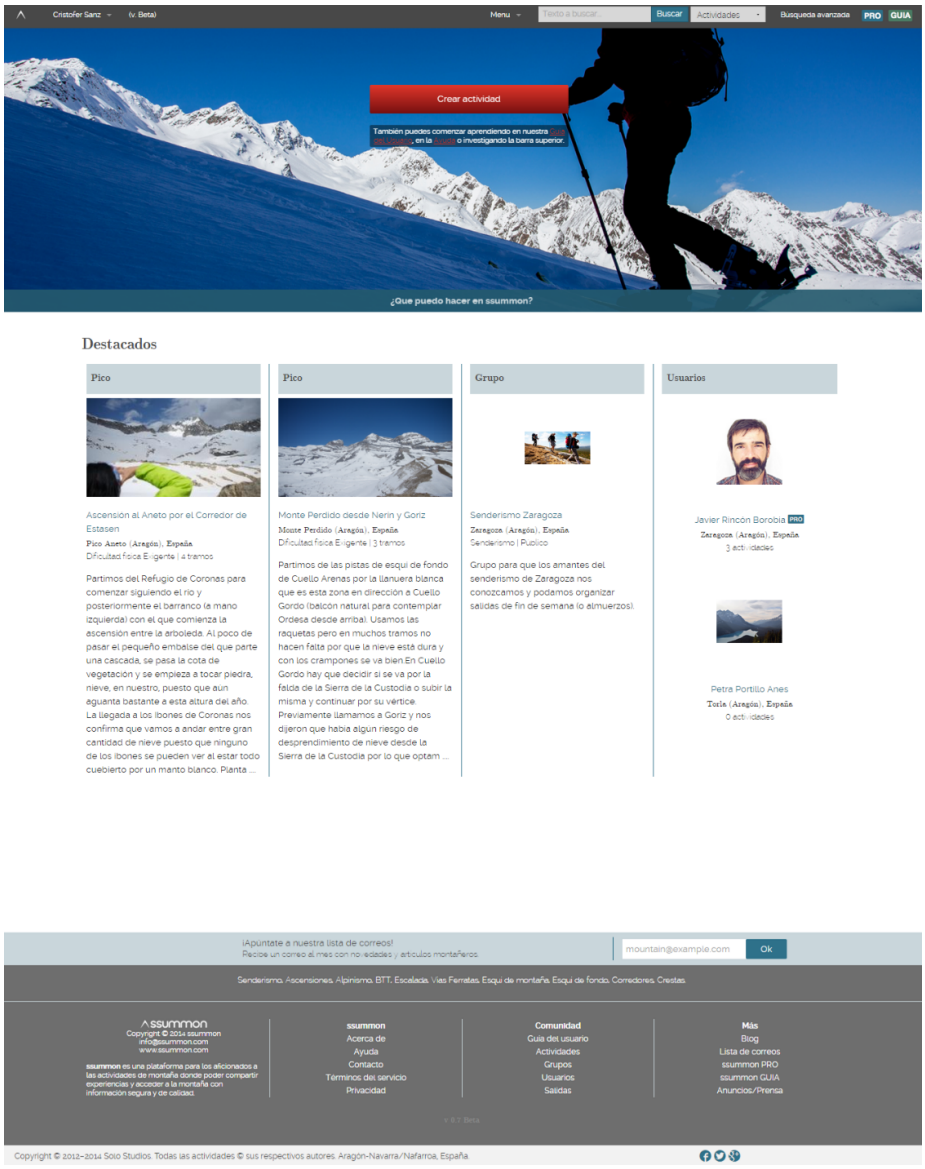


Figura B.3: Pantalla principal (panel).

Figura B.4: Menú superior

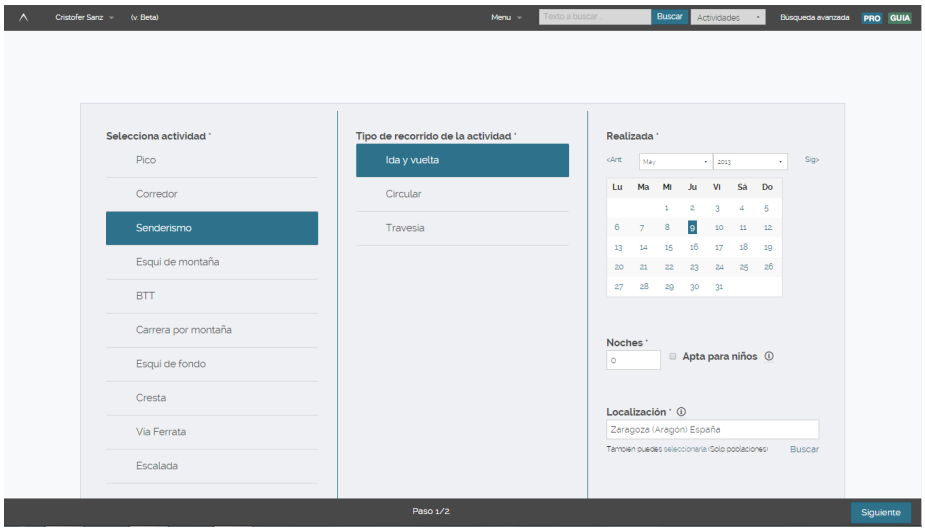


Figura B.5: Creación de actividad.

Figura B.6: Especificación aproximada de localidad.

Figura B.7: Especificación jerárquica de localidad.

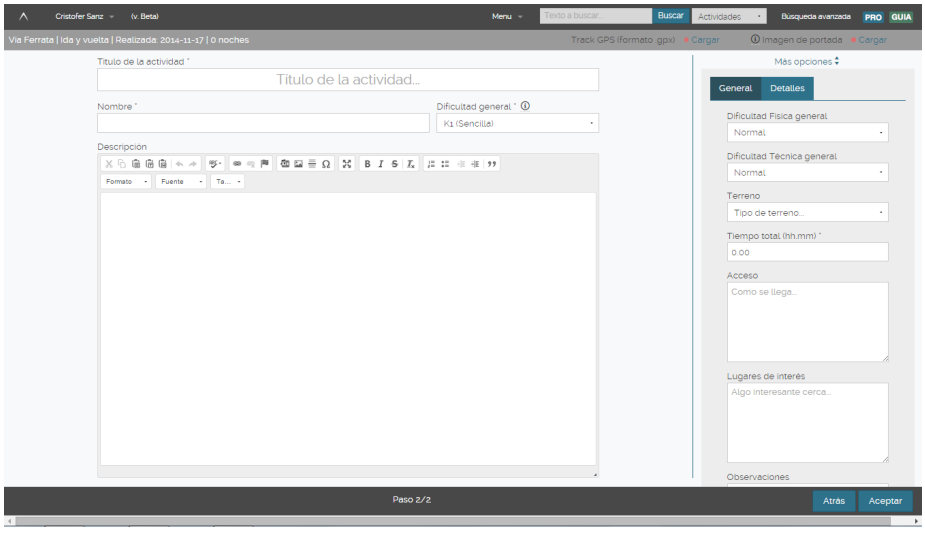


Figura B.8: Redacción de actividades.

de usuarios pueden detallar una actividad en varios tramos (Figura B.9), de modo que cada tramo puede ser de diferente tipo y contener atributos propios y diferentes del resto de tramos. Esto dota al proceso de mayor precisión y flexibilidad.

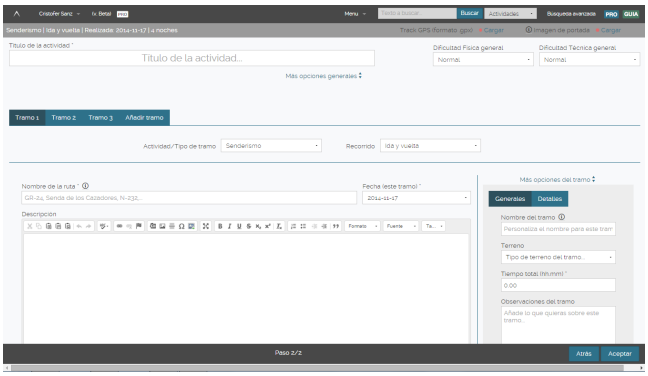


Figura B.9: Redacción de actividades de varios tramos.

Tras crear una actividad, esta es agregada al sistema para que el resto de los usuarios puedan verla y se asocia al usuario desde el que fue creada. Posteriormente se puede acceder a ella y al resto de contenido creado por el usuario, como grupos y salidas, desde el menú de usuario (Figura B.10).

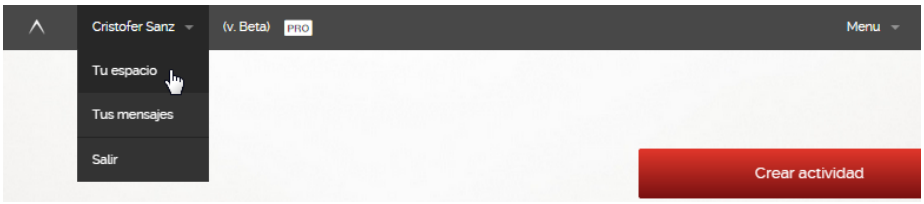


Figura B.10: Menú de usuario.

Visualización de actividad Cada vez que se crea una actividad, la aplicación web redirecciona a la vista de la actividad. Como se ha comentado, si queremos acceder a otras actividades creadas en el pasado lo haremos desde el menú de usuario (Figura B.10). Además del resto de contenido creado por el usuario, aparecerán todas las actividades generadas por él. Bastará con hacer clic en cualquiera de ellas para ir a la vista de la actividad (Figura C.13).

Si por otro lado queremos visualizar una actividad creada por otro usuario, podremos hacerlo desde el perfil de este o realizando una búsqueda, como se explicará más adelante.

Modificación de actividad Con la gran cantidad de parámetros que se puede asociar a cualquier actividad es posible que se erre al introducir alguno o,

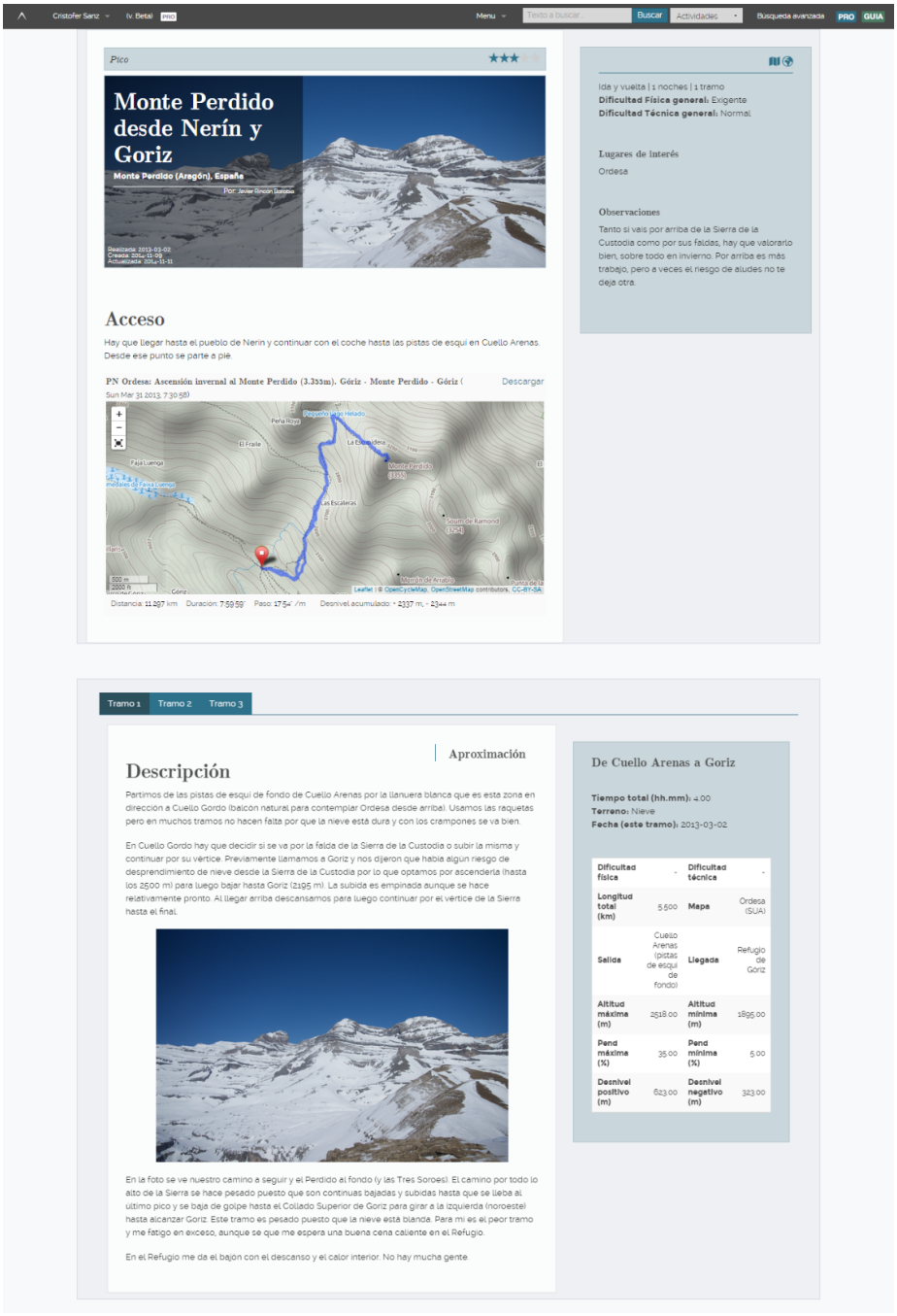


Figura B.11: Vista de actividad de varios tramos.

simplemente, se desee actualizar algún otro. Para lograrlo, se debe acceder a la vista de cualquier actividad propia del usuario y hacer clic en editar, bajo la cabecera de la actividad (Figura B.12.

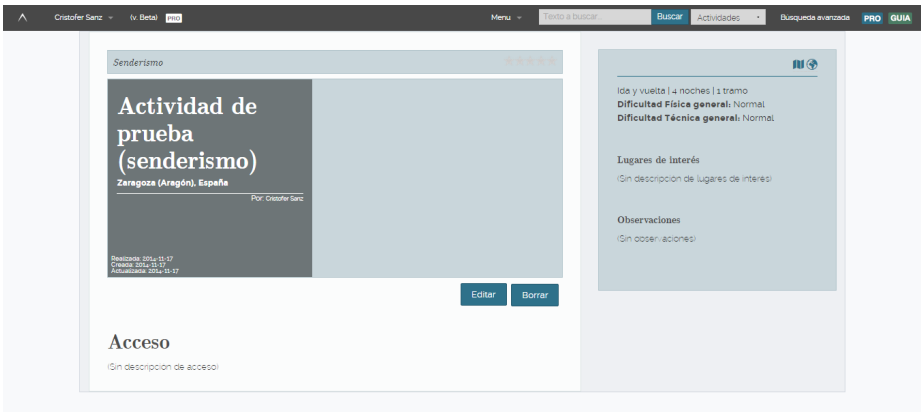


Figura B.12: Modificación de actividad.

Lo anterior conducirá a la pantalla de edición, muy similar en aspecto a de creación de nueva actividad, para eliminar o actualizar el contenido de cualquiera de sus parámetros.

Eliminación de actividad El proceso a realizar para eliminar una actividad es prácticamente igual al anterior. Ésta vez, en lugar de pulsar el botón editar, se ha de pulsar el botón borrar. El sistema mostrará un mensaje de confirmación y, si aceptamos, mostrará otro mensaje de eliminación exitosa.

Búsqueda simple No cabe duda de que poder crear, actualizar o eliminar elementos en un sistema gestor son operaciones cruciales. Sin embargo, estas operaciones no lo hacen destacar por encima de otro, por bien implementadas que estén. La capacidad de búsqueda y de extracción de información relevante sí es un componente que puede marcar la diferencia.

La aplicación web permite realizar búsquedas entre actividades, usuarios, grupos, salidas y conversaciones. De nuevo, si eliminamos la parte social, lo interesante son las búsquedas de actividades.

Efectuar una búsqueda simple es tan sencillo como introducir el nombre de la actividad en la casilla de búsqueda que hay en la barra superior (Figura B.13) y hacer clic en buscar.

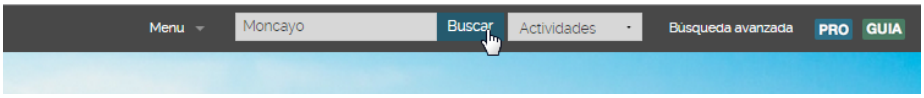


Figura B.13: Búsqueda simple.

Tras pulsar el botón, aparecerá una lista de resultados con las actividades que guarden relación con el texto buscado.

Búsqueda avanzada Como consecuencia del número de atributos asociados a cada actividad, la búsqueda avanzada ayuda a encontrar exactamente la actividad deseada, aumentando la precisión. Para poder ejecutar una búsqueda avanzada debemos pulsar el botón de búsqueda avanzada de la barra superior, próximo a la zona de búsqueda simple (Figura B.13). Accederemos a la pantalla de búsqueda avanzada (Figura B.14) donde, tras elegir el tipo de actividad a buscar, el sistema presentará un formulario con parámetros que varían dependiendo del tipo de actividad. Después de cumplimentarlo, el sistema buscará las actividades que coincidan exactamente con los valores proporcionados.

Figura B.14: Búsqueda avanzada.

B.1.2. Problemas y necesidades

Durante el proceso de exploración del sistema actual se han detectado ciertos problemas que dificultan el desarrollo del nuevo. Algunos se pueden simplemente evitar, otros es necesario corregirlos antes de implementarlo. Comenzamos exponiendo primero aquellos problemas con carácter más general para acabar enunciando aquellos más técnicos.

El primer inconveniente al realizar el proyecto es el estado en el que se encuentra el sistema actual, en **desarrollo activo**. Si bien no es una contra mayor, pues el sistema está bien diseñado e implementado casi por completo, sí afecta a la toma de decisiones. Los efectos de estar en desarrollo activo son continuas mutaciones en el sistema de base de datos, alteración de algunas funcionalidades o pequeños detalles estéticos. Éste es uno de los inconvenientes con los que hay que lidiar y para los que no hay solución.

La segunda dificultad surge analizando la **arquitectura del sistema**. La aplicación web accede a la capa de datos de modo directo, sin capas intermedias, con los inconvenientes que ello acarrea. Primeramente, la seguridad de todo el

sistema, tanto de la aplicación como de los datos, puede ser vulnerada con mayor facilidad (aumenta el número de posibles vectores de ataque contra la capa de datos). En segundo lugar, operar directamente sobre una capa de datos puede ocasionar la pérdida de integridad en los datos, especialmente si el tamaño de esta crece en exceso, aun siendo cauto en el desarrollo de las operaciones sobre ella. Por último, se ha de tener en cuenta que ambos sistemas operaran sobre el mismo conjunto de datos. Por ello, surge una necesidad imperiosa de crear una capa que defina el acceso a datos y sea utilizada por ambos sistemas por igual. De no ser así, los mismos procesos en diferentes sistemas pueden tratar de diferente manera los mismos datos.

B.2. Análisis

B.2.1. Listado de requisitos

Esta sección contiene un listado exhaustivo de los requisitos extraídos de la descripción del sistema (Apartado 3.2.1). Estos están clasificados en:

- Requisitos funcionales: Describen una tarea que ha de cumplir el sistema.
- Requisitos no funcionales: Describen restricciones no relacionadas con las funciones que debe cumplir el sistema (por ejemplo: usar un tipo de base de datos, definir la interfaz visual, etc.).
- Requisitos de desarrollo: Aquellos impuestos por el equipo de desarrollo, derivados de los requisitos del cliente (funcionales y no funcionales).

Requisitos funcionales

ID	RF1	ÁREA	Gestión de usuarios
TIPO	Funcional		
DESCRIPCIÓN	En la primera ejecución, no hay ningún usuario autenticado.		

ID	RF2	ÁREA	Gestión de usuarios
TIPO	Funcional		
DESCRIPCIÓN	La aplicación requiere que haya un único usuario autenticado para realizar cualquier operación.		

ID	RF3	ÁREA	Gestión de usuarios
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite autenticar un usuario la base de datos de la aplicación web.		

ID	RF4	ÁREA	Gestión de usuarios
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite crear un usuario nuevo en la base de datos de la aplicación web.		

ID	RF5	ÁREA	Gestión de usuarios
TIPO	Funcional		
DESCRIPCIÓN	La aplicación no podrá modificar ni eliminar los usuarios existentes.		

ID	RF6	ÁREA	Gestión de usuarios
TIPO	Funcional		
DESCRIPCIÓN	La aplicación mostrará información estadística sobre el perfil del usuario.		

ID	RF7	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite insertar una nueva actividad en la base de datos local.		

ID	RF8	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite modificar una actividad existente en la base de datos local.		

ID	RF9	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite eliminar una actividad existente en la base de datos local.		

ID	RF10	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite exportar todas las actividades en un fichero CSV.		

ID	RF11	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	Las actividades monotramo pueden ser gestionadas por cualquier tipo de usuario: normal y pro.		

ID	RF12	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	Las actividades multitramo sólo pueden ser gestionadas por usuarios pro.		

ID	RF13	ÁREA	Gestión de actividades
TIPO	Funcional		
DESCRIPCIÓN	Si un usuario pro se convierte en usuario normal sus actividades multitramo se conservarán, podrá visualizarlas y eliminarlas, pero no modificarlas ni crear nuevas.		

ID	RF14	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Las actividades creadas por la aplicación serán privadas por defecto.		

ID	RF15	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite cambiar el estado de una actividad de privada a pública y de pública a privada.		

ID	RF16	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Al cambiar una actividad de privada a pública, la aplicación crea una nueva actividad en la web.		

ID	RF17	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Al cambiar una actividad de pública a privada, la aplicación elimina la actividad correspondiente de la web.		

ID	RF18	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite sincronizar las actividades manualmente (mediante un botón).		

ID	RF19	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite sincronizar varias actividades a la vez seleccionándolas previamente.		

ID	RF20	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Los usuarios normales sólo podrán sincronizar sus actividades monotramo.		

ID	RF21	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Los usuarios pro podrán sincronizar todas sus actividades.		

ID	RF22	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Si un usuario pro se convierte en usuario normal sólo podrá sincronizar sus actividades monotramo.		

ID	RF23	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Existe un mecanismo de identificación de una misma actividad en ambas plataformas.		

ID	RF24	ÁREA	Sincronización
TIPO	Funcional		
DESCRIPCIÓN	Al sincronizar una actividad, si ésta ha sido modificada más recientemente en la aplicación web, se le pedirá al usuario elegir que versión consevar mostrando timestamps de ambas versiones en una ventana diálogo.		

ID	RF25	ÁREA	Búsqueda y filtrado
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite buscar actividades comparando el texto con los campos del título y descripción de la actividad.		

ID	RF26	ÁREA	Búsqueda y filtrado
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite filtrar las actividades atendiendo a su tipo de actividad.		

ID	RF27	ÁREA	Mapa
TIPO	Funcional		
DESCRIPCIÓN	Si la actividad contiene un fichero track, la aplicación lo mostrará en un mapa.		

ID	RF28	ÁREA	Mapa
TIPO	Funcional		
DESCRIPCIÓN	Si hay conexión a Internet, el mapa muestra el track de la actividad sobre la capa de OpenStreetMap.		

ID	RF29	ÁREA	Mapa
TIPO	Funcional		
DESCRIPCIÓN	Si no hay conexión a Internet, el mapa muestra sólo el track de la actividad.		

ID	RF30	ÁREA	Editor de textos
TIPO	Funcional		
DESCRIPCIÓN	La aplicación web permitirá insertar texto formateado, imágenes y vídeos en la descripción de las actividades.		

ID	RF31	ÁREA	Informes PDF
TIPO	Funcional		
DESCRIPCIÓN	La aplicación permite generar informes pdf de actividades a los usuarios pro.		

ID	RF32	ÁREA	Intefaz visual
TIPO	Funcional		
DESCRIPCIÓN	Si el usuario intenta ejecutar una operación no permitida, la aplicación muestra mediante un mensaje de error en un diálogo.		

ID	RF33	ÁREA	Interfaz visual
TIPO	Funcional		
DESCRIPCIÓN	Si el usuario intenta ejecutar una operación destructiva, la aplicación solicita confirmación en un diálogo.		

Requisitos no funcionales

ID	RNF1	ÁREA	Conexión a Internet
TIPO	No Funcional		
DESCRIPCIÓN	La creación de un nuevo usuario se realizará a través de *web services*.		

ID	RNF2	ÁREA	Interfaz visual
TIPO	No Funcional		
DESCRIPCIÓN	La interfaz visual de la aplicación es lo más similar posible a la de la aplicación web, incluyendo pantallas y elementos.		

ID	RNF3	ÁREA	Persistencia (base de datos)
TIPO	No Funcional		
DESCRIPCIÓN	La aplicación requiere un sistema gestor de base de datos relacional.		

ID	RNF4	ÁREA	Persistencia (base de datos)
TIPO	No Funcional		
DESCRIPCIÓN	El esquema de base de datos ha de ser lo más parecido posible al utilizado por la aplicación web.		

ID	RNF5	ÁREA	Persistencia (ficheros)
TIPO	No Funcional		
DESCRIPCIÓN	La aplicación utilizará un sistema de ficheros con las misma topología que el descrito en Apartado 3.2.1 (Figura 3.3)		

ID	RNF6	ÁREA	Persistencia (ficheros)
TIPO	No Funcional		
DESCRIPCIÓN	Las imágenes que cargue el usuario en la aplicación deberán tener formato JPEG o PNG.		

ID	RNF7	ÁREA	Persistencia (ficheros)
TIPO	No Funcional		
DESCRIPCIÓN	Los tracks que cargue el usuario en la aplicación deberán tener formato GPX.		

ID	RNF8	ÁREA	Conexión a Internet
TIPO	No Funcional		
DESCRIPCIÓN	La aplicación no requiere conexión continua a Internet.		

ID	RNF9	ÁREA	Conexión a Internet
TIPO	No Funcional		
DESCRIPCIÓN	La aplicación requiere conexión a Internet para: autenticar/registrar usuario, hacer pública/privada una actividad, sincronizar actividades o visualizar la capa de cartografía de OpenStreetMap.		

ID	RNF10	ÁREA	Herramientas
TIPO	No Funcional		
DESCRIPCIÓN	La aplicación se implementará utilizando el lenguaje de programación Java.		

Requisitos de desarrollo

ID	RD1	ÁREA	Persistencia (base de datos)
TIPO	Desarrollo		
DESCRIPCIÓN	Se utilizará SQLite como SGBD (Apéndice B.3.1)		

ID	RD2	ÁREA	Persistencia (base de datos)
TIPO	Desarrollo		
DESCRIPCIÓN	El esquema de base de datos integrará la base de datos del servicio de información geográfica Geonames.		

ID	RD3	ÁREA	Interfaz visual
TIPO	Desarrollo		
DESCRIPCIÓN	Se utilizará JavaFX para crear el interfaz visual de la aplicación (Apéndice B.3.2).		

ID	RD4	ÁREA	Conexión a Internet
TIPO	Desarrollo		
DESCRIPCIÓN	La aplicación utilizará web services cuando se comunique con la aplicación web.		

B.2.2. Casos de uso

ID	1.1	NOMBRE	Iniciar sesión de usuario
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario inicia sesión para poder realizar cualquier operación. La aplicación verifica la existencia y corrección de sus credenciales y éste pasa a ser el usuario activo.		
CONDICIÓN INICIAL	Aplicación iniciada		
FLUJO DE EVENTOS NORMAL	**Usuario** Introduce sus credenciales. **myssummon** Verifica los credenciales. **myssummon** (Si credenciales válidos) Carga los datos del usuario.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario** Introduce sus credenciales. **myssummon** Verifica los credenciales. **myssummon** (Si credenciales inválidos) Muestra mensaje de error.		

ID	1.2	NOMBRE	Visualizar perfil
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario visualiza la información pública de su perfil en una ficha.		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en “Tu espacio”, bajo el menú de usuario. **myssummon** Muestra la ficha de perfil del usuario.		

ID	1.3	NOMBRE	Visualizar estadísticas locales
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario visualiza estadísticas sobre las actividades de su perfil local.		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en "Tu espacio", bajo el menú de usuario. **Usuario** Hace clic en "Estadísticas", en la cabecera del panel. **myssummon** Muestra las estadísticas del usuario.		

ID	1.4	NOMBRE	Cerrar sesión de usuario
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario cierra sesión y la aplicación descarga los datos de usuario (cierra conexión de la base de datos de sus contenidos).		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en "Salir", bajo el menú de usuario. **myssummon** Cierra conexión con la base de datos de contenidos del usuario y deja de haber usuarios activos. **myssummon** Muestra la pantalla de inicio de sesión.		

ID	2	NOMBRE	Visualizar actividades panel
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario visualiza el panel con todas sus actividades.		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en "Panel", en el menú superior. **myssummon** Muestra el panel con las actividades.		

ID	2.1	NOMBRE	Buscar actividades
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario visualiza las actividades que contengan los términos de búsqueda en sus campos de título y/o descripción.		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario** Introduce los términos de búsqueda en el buscador (menú superior). **myssummon** Muestra las actividades que encajen con los términos de búsqueda.		

ID	2.2	NOMBRE	Filtrar actividades
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario visualiza las actividades cuyo tipo de actividad coincida con el filtro.		
CONDICIÓN INICIAL	Usuario autenticado, en el panel de actividades.		
FLUJO DE EVENTOS NORMAL	**Usuario** Selecciona (haciendo clic) el filtro de actividades. **myssummon** Muestra las actividades cuyo tipo de actividad encaje con el filtro.		

ID	3.1.1	NOMBRE	Añadir actividad monotramo
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario inserta una nueva actividad de un solo tramo.		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en "Nueva actividad", en el menú superior. **myssummon** Muestra la pantalla de creación de actividad. **Usuario** Rellena los atributos generales, comunes a cualquier actividad, seleccionando como tipo de recorrido, "Un tramo". **myssummon** (Si los datos son válidos) Muestra la pantalla de redacción de actividad. **Usuario** Rellena los atributos específicos, así como la descripción y atributos del tramo. **myssummon** (Si los datos son válidos) Inserta la actividad en la base de datos y muestra la pantalla de vista de la actividad recién creada.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario** Hace clic en "Nueva actividad", en el menú superior. **myssummon** Muestra la pantalla de creación de actividad. **Usuario** Rellena los atributos generales olvidando alguno. **myssummon** Muestra mensaje de error informando de que faltan datos.		

ID	3.1.2	NOMBRE	Añadir actividad multitramo
ACTOR(ES)	Usuario pro		
DESCRIPCIÓN	El usuario inserta una nueva actividad con varios tramos.		
CONDICIÓN INICIAL	Usuario autenticado, en cualquier pantalla.		
FLUJO DE EVENTOS NORMAL	**Usuario pro** Hace clic en "Nueva actividad", en el menú superior. **myssummon** Muestra la pantalla de creación de actividad. **Usuario pro** Rellena los atributos generales, comunes a cualquier actividad, seleccionando como tipo de recorrido, "Varios tramos". **myssummon** (Si los datos son válidos) Muestra la pantalla de redacción de actividad. **Usuario pro** Rellena los atributos específicos de la actividad. **Usuario pro** Crea tantos tramos como desee, rellenando sus atributos y descripciones. **myssummon** (Si los datos son válidos) Inserta la actividad en la base de datos y muestra la pantalla de vista de la actividad recién creada.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario normal** Hace clic en "Nueva actividad", en el menú superior. **myssummon** Muestra la pantalla de creación de actividad. **Usuario normal** Rellena los atributos generales, comunes a cualquier actividad, seleccionando como tipo de recorrido, "Varios tramos". **myssummon** Muestra mensaje de error informando de que es una opción sólo disponible para usuarios pro.		

ID	3.2.1	NOMBRE	Modificar actividad monotramo
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario modifica una actividad de un solo tramo que ya existe en su perfil.		
CONDICIÓN INICIAL	Usuario autenticado, en vista de una actividad monotramo.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en "Editar", bajo la portada de la actividad. **myssummon** Muestra la pantalla de creación de actividad con los datos de la actividad rellenos en sus campos. **Usuario** Modifica los atributos que desee y hace clic en "Siguiente". **myssummon** (Si los datos son válidos) Muestra la pantalla de redacción de actividad con los atributos específicos rellenos en sus campos. **Usuario** Modifica los atributos que desee y hace clic en "Finalizar". **myssummon** (Si los datos son válidos) Modifica la actividad en la base de datos incorporando los cambios y muestra la pantalla de vista de la actividad.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario** Hace clic en "Editar", bajo la portada de la actividad. **myssummon** Muestra la pantalla de creación de actividad con los datos de la actividad rellenos en sus campos. **Usuario** Modifica los atributos, dejando alguno vacío, y hace clic en "Siguiente". **myssummon** Muestra un mensaje de error informando de que los datos no son válidos.		

ID	3.3	NOMBRE	Visualizar actividad
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	La aplicación muestra la vista de una actividad de uno o varios tramos incluyendo los atributos de la actividad, los de los tramos, tracks en mapa, etc.		
CONDICIÓN INICIAL	Usuario autenticado, en panel.		
FLUJO DE EVENTOS NORMAL	**Usuario** Escoge una de las actividades que aparecen en el panel (puede buscar entre ellas o filtrarlas) y hace clic en su título. **myssummon** Muestra la vista de la actividad.		

ID	3.5	NOMBRE	Exportar actividades
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario exporta todas las actividades de su perfil en un fichero CSV.		
CONDICIÓN INICIAL	Usuario autenticado, en panel.		
FLUJO DE EVENTOS NORMAL	**Usuario** Hace clic en “Exportar actividades”. **myssummon** Muestra una ventana para elegir nombre del fichero y ruta donde exportarlo. **Usuario** Escoge la ruta y rellena el nombre del fichero. **myssummon** Exporta el fichero CSV con todas las actividades del usuario.		

ID	3.6.2	NOMBRE	Sincronizar una actividad multitramo
ACTOR(ES)	Usuario pro		
DESCRIPCIÓN	El usuario sincroniza una actividad de varios tramos haciendo que mantenga el mismo estado en ambos entornos.		
CONDICIÓN INICIAL	Usuario autenticado, en vista de una actividad multitramo.		
FLUJO DE EVENTOS NORMAL	**Usuario pro** Hace clic en "Sincronizar", bajo la portada de la actividad. **myssummon** (Si la actividad en myssummon es más reciente que en ssummon) Actualiza la actividad en ssummon. **myssummon** Muestra un mensaje de éxito.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario pro** Hace clic en "Sincronizar", bajo la portada de la actividad. **myssummon** (Si la actividad en ssummon es más reciente que en myssummon) Pregunta al usuario que versión desea conservar. **Usuario pro** Elige la versión que desee. **myssummon** Actualiza con los datos de la versión escogida la actividad en el otro entorno.		

ID	3.7.2	NOMBRE	Sincronizar varias actividades multitramo
ACTOR(ES)	Usuario normal/pro		
DESCRIPCIÓN	El usuario sincroniza varias actividades de varios tramos en lote haciendo que todas ellas mantengan el mismo estado en ambos entornos.		
CONDICIÓN INICIAL	Usuario autenticado, en panel.		
FLUJO DE EVENTOS NORMAL	**Usuario** Escoge las actividades que desee sincronizar y hace clic en "Marcar para sincronizar" en cada una de ellas. **Usuario** Hace clic en "Sincronizar en lote". (Para cada una de ellas) **myssummon** Actualiza la actividad en ssummon o solicita elegir versión a conservar. **Usuario** (Si solicita confirmación) Elige la versión que desee. **myssummon** Actualiza con los datos de la versión escogida (si ha solicitado confirmación) o con los de myssummon. **myssummon** Muestra un mensaje de éxito.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario normal** Escoge las actividades que desee sincronizar y hace clic en "Marcar para sincronizar" en cada una de ellas. **myssummon** Muestra mensaje de error informando de que sólo está disponible para usuarios pro.		

ID	4	NOMBRE	Generar informe PDF
ACTOR(ES)	Usuario pro		
DESCRIPCIÓN	La aplicación genera una guía de campo en PDF con datos sobre la actividad (incluyendo atributos y mapa).		
CONDICIÓN INICIAL	Usuario autenticado, en vista de una actividad.		
FLUJO DE EVENTOS NORMAL	**Usuario pro** Hace clic en "Generar informe PDF". **myssummon** Muestra un diálogo para elegir el tipo de informe y sus características. **Usuario pro** Elige el informe que desee. **myssummon** Genera el informe y lo abre con el visor predeterminado de documentos PDF instalado en el equipo.		
FLUJO DE EVENTOS ALTERNATIVO	**Usuario normal** Hace clic en "Generar informe PDF". **myssummon** Muestra mensaje de error informando de que sólo está disponible para usuarios pro.		

B.3. Análisis de alternativas

En esta sección se discute las elecciones de alternativas que se han realizado previas al diseño de la aplicación, describiendo las opciones que existen y valorando pros y contras en cada una de ellas.

B.3.1. Sistema gestor de base de datos

El equipo de SSUMMON era partidario de continuar utilizando MySQL ya que la reutilización del modelo de datos sería más sencilla. No obstante, inconvenientes como la necesidad de contar con un servidor MySQL en el nodo donde se fuera a ejecutar la aplicación animaron la búsqueda de alternativas.

El punto de partida en la elección del SGBD apropiado ha sido descartar todos los sistemas no relacionales (sistemas NoSQL, clave/valor, etc). De entre todos ellos, interesaban especialmente aquellos sistemas que no basaran su funcionamiento en una arquitectura cliente/servidor, como en el caso de MySQL, ya que la distribución de la aplicación sería más sencilla. Estas restricciones limitaron las posibles alternativas a dos: SQLite y H2.

MySQL
Fue la primera opción por ser el sistema utilizado en SSUMMON. Es el más potente y veterano de los tres. Aunque es el único que ofrece características avanzadas como el *clustering* o el *sharding*, estas no llegan a entrar en juego en el sistema actual. Su arquitectura basada en cliente/servidor ha impulsado la búsqueda de alternativas.

Una lista de sus ventajas y desventajas:

+ Implementación inmediata del esquema existente en SSUMMON.
+ Máximo aprovechamiento de los mecanismos creados. Ciertos mecanismos como las claves o los comentarios de atributos y tablas no son soportados por otros SGBD por no haberlos implementado o por formar parte del dialecto MySQL.
+ Larga comunidad de usuarios. Es uno de los puntos clave que hay que observar a la hora de decidir entre alternativas software.
– Arquitectura cliente servidor. Como se ha comentado, puede complicar la distribución, la instalación por parte del usuario e incluso cargar el sistema con procesos que consumen muchos recursos.
– Doble licencia: GPL (*General Public License*) y licencia comercial. Dependiendo del uso dado a MySQL será necesario utilizar uno y otro tipo de licencia. Puesto que el sistema software en desarrollo no se ha licenciado por el momento, puede suponer un gasto económico.

SQLite
SQLite fue la primera alternativa en la que se pensó. Es una buena opción para tratar con bases de datos locales por ser pequeña, rápida y por su arquitectura *serverless*. No implementa ciertas características del lenguaje SQL, tales como algunos tipos de *JOIN*, o el comando *ALTER TABLE* en su totalidad [15]. Aun así, son inconvenientes con los que se puede lidiar ya que no afectan al desarrollo del sistema.

Un resumen de sus pros y sus contras:

+ Arquitectura *serverless*.
+ También existe una larga comunidad de usuarios detrás.
+ El autor cuenta con experiencia previa utilizando este SGBD.
+ Su uso no requiere licencia, es de dominio público [16].
– No permite uso de roles ni usuarios. Dado que es un sistema que no utiliza servidor, la única gestión de permisos es la que establece el sistema de ficheros. Esto puede mermar la seguridad del sistema.

H2
Se denomina la base de datos SQL de Java. Es un SGBD que puede funcionar integrado en Java y ejecutar sentencias SQL sin necesidad de utilizar *sockets*.

+ Dos modos de funcionamiento: embebido en Java o cliente/servidor.
+ Tiene dos licencias posibles: MPL 2.0 (*Mozilla Public License*) y EPL 1.0 (*Eclipse Public License*) [6]. Por lo tanto, se puede utilizar gratuitamente incluso en software comercial.
– Relativamente nueva, lo cual puede suponer una mayor probabilidad de aparición de fallos críticos.
– El autor carece de cualquier experiencia con este SGBD.

Para terminar, se ha querido analizar la popularidad de los tres SGBD. En Figura B.15, se aprecia como ha evolucionado la popularidad de los sistemas durante los dos últimos años. El análisis de popularidad proviene de DB-Engines [4]. Para calcular los índices de popularidad se han tenido en cuenta varios factores: menciones e interés del SGBD en la web (a través de Google Trends), frecuencia de las discusiones técnicas (en foros como StackOverflow), ofertas de trabajo publicadas, etc.) [3].

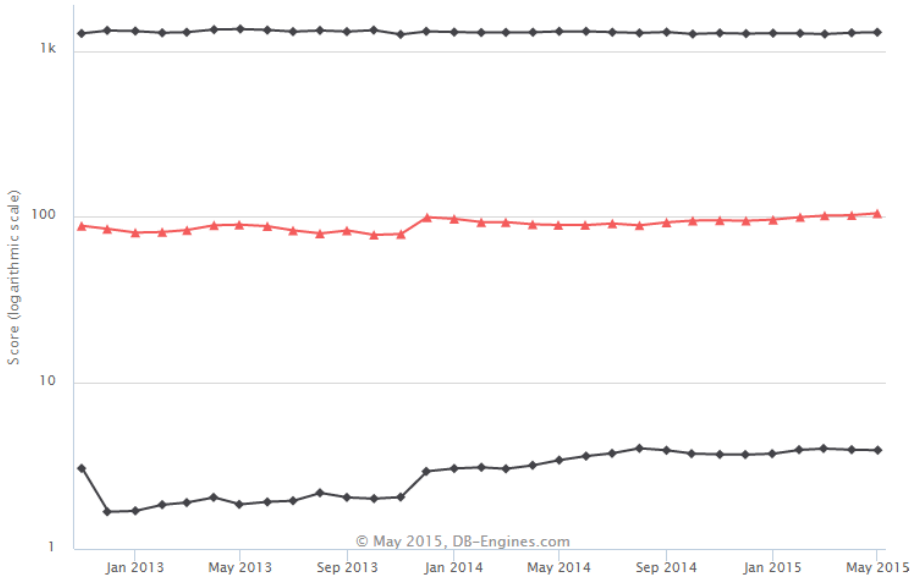


Figura B.15: Comparación de popularidad de MySQL, SQLite y H2. MySQL, línea negra superior; SQLite, línea roja media; H2, línea roja inferior.

La elección tomada ha sido SQLite por sus apreciables ventajas y su diferencia de popularidad con respecto a H2.

B.3.2. Interfaz visual

Uno de los requisitos que se impuso desde el principio (pag. 83) fue que el estilo visual de la aplicación fuera lo más similar posible al de la aplicación web. Este ha sido el objetivo al explorar las opciones disponibles para implementar la interfaz visual.

En un principio se tuvieron en cuenta las siguientes opciones: AWT, Swing, SWT, SwingX y JavaFX. Sin embargo, debido al bajo grado de personalización, se descartó AWT y SWT, librerías gráficas estables pero antiguas, que carecen de componentes avanzados y de un aspecto visual moderno. También se descarto SwingX por tratarse de un proyecto bajo desarrollo poco activo. Así pues, las opciones que permanecieron fueron Swing y JavaFX.

Swing

Swing es una veterana biblioteca gráfica para Java que contiene componentes para construir interfaces visuales. Su catálogo de componentes es bastante extenso (Figura B.16) además de existir bastantes elementos de terceros. Su principal forma de personalización es a través de temas (*look & feels*). He aquí un resumen de sus ventajas e inconvenientes:

+ Su catálogo de componentes es más extenso que el de JavaFX.
+ Es una biblioteca gráfica bastante madura (apareció en 1996), por lo que la probabilidad de fallos críticos es baja.
– El diseño de *layouts* es engorroso. El uso de componentes contenedores no es sencillo y no se proporciona una herramienta para diseñarlos (aunque existen opciones de terceros).

JavaFX

JavaFX es otra biblioteca gráfica, más reciente que Swing pero con menor catálogo de componentes. Tiene soporte de características avanzadas como gráficos 3D, API para dibujos canvas y gráficos acelerados por hardware. Aunque estas características no se empleen en este proyecto por el momento, es posible que se lleguen a usar según evolucione la aplicación.

Éstos son sus principales puntos a favor y en contra:

+ Aspecto visual mucho más atractivo y limpio con respecto a Swing.
+ Su grado de personalización es muy alto (Figura B.17). Soporta hojas de estilo en cascada.
+ Hace uso del patrón MVC. Separa la vista, implementada mediante ficheros FXML (ficheros XML que contienen la estructura del layout), de la lógica, albergada en clases (denominadas "controladores").
+ Existe una herramienta para el diseño de layouts: JavaFX Scene Builder 2.

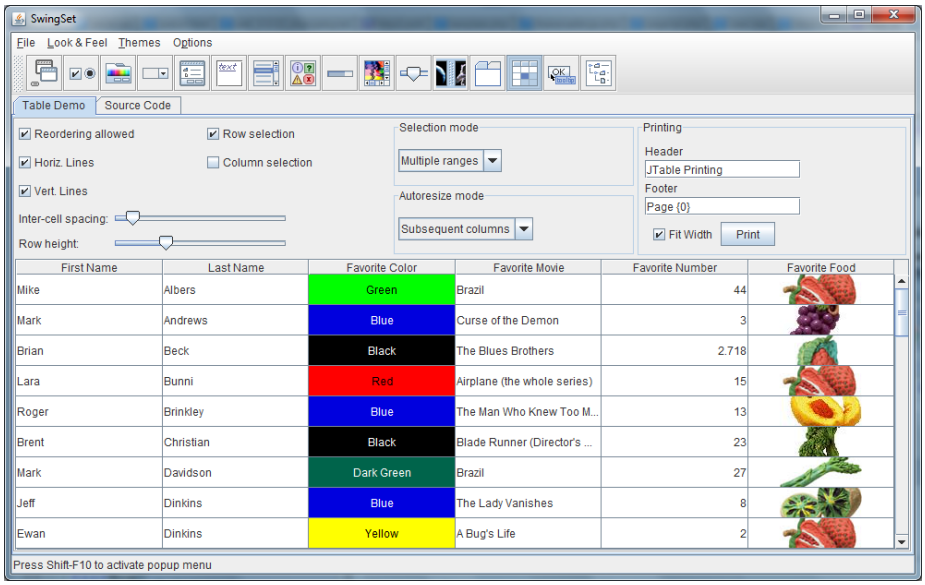


Figura B.16: Ejemplo de ventana con componentes Swing.

+ Su *API* es más consistente entre los diferentes elementos (tratar de buscar una ref).

– Implica usar Java 8 (la librería está integrada en Java a partir de esta versión) o integrarlo como dependencia a parte.

La solución utilizada ha sido JavaFX. Lo que más influyó en esta decisión fue su capacidad absoluta de personalización y su aspecto limpio y cuidado. No obstante, no hay que dejar de lado el resto de ventajas mencionadas que lo convierten en una solución sobresaliente.

B.4. Diseño

B.4.1. Modelo de datos

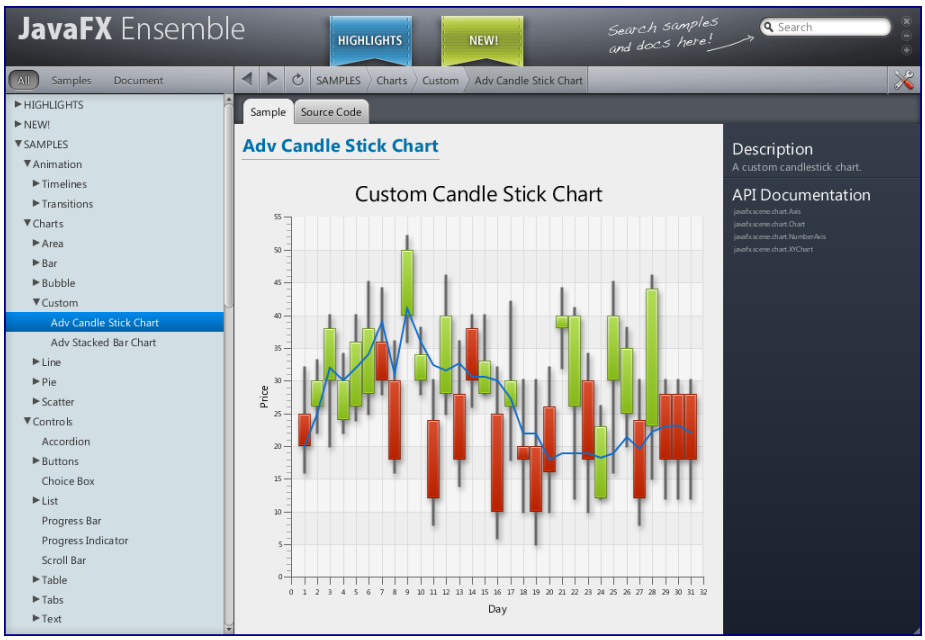


Figura B.17: Ejemplo de ventana JavaFX y su alto grado de personalización.

Table tbl_actividadpersonal		
*	idAc	int AUTOINCREMENT
*	idUsAc	int
*	fechaCreacAc	datetime
	ultimaActualizAc	datetime
*	tituloAc	varchar(150)
*	fechaAc	date
*	nochesAc	int DEFO 0
	observacionesAc	text
	lugInteresCerAc	text
	accesoAc	text
	trackGpsAC	varchar(500)
*	dificultadFisAc	int DEFO 2
*	dificultadTecAc	int DEFO 2
	latPtoAc	float(12, 0)
	lonPtoAc	float(12, 0)
*	idTipoAc	int
*	idTipoRecorridoAc	int DEFO 1
*	invernalAc	int DEFO 0
*	paisAc	int
*	provCercanaAc	int
*	pobCercanaAc	int
	variosTramosAc	int DEFO 0
	imgPortadaAc	varchar(500)
	valoracionAc	int DEFO 0
*	destacadaAc	int UNSIGNED DEFO 0
	apto_ninos	int DEFO 0
	precioAc	decimal(7, 2)
	compradaAc	int DEFO 0
*	esEditorAc	bit DEFO 0
	extractoEditorAc	text
*	borradorAc	bit DEFO 0
	vecesVistaAc	int DEFO 0
	clickCompartidaAc	int DEFO 0
	clickPdfAc	int DEFO 0
	historialActuEdAc	text
Indexes		
Pk	pk_tbl_actividadpersonal	ON idAc
	FKDifFAc_idx	ON dificultadFisAc
	FKDifTAc_idx	ON dificultadTecAc
	FKpaisAc_idx	ON paisAc
	FKPobAc_idx	ON pobCercanaAc
	FKprovAc_idx	ON provCercanaAc
	FKTipoAc_idx	ON apto_ninos
	FKTRecAc_idx	ON idTipoRecorridoAc
	FKUs_idx	ON idUsAc
Foreign Keys		
	FKDifFAc	(dificultadFisAc) ref tbl_dificultadfisica (idDF)
	FKDifTAc	(dificultadTecAc) ref tbl_dificultadtecnica (idDT)
	FKTipoAc	(apto_ninos) ref tbl_edadninos (idEN)
	FKpobAc	(pobCercanaAc) ref tbl_localizacion (idLoc)
	FKpaisAc	(paisAc) ref tbl_pais (idPais)
	FKprovAc	(provCercanaAc) ref tbl_provincia (idProv)
	FKAcUs	(idUsAc) ref tbl_usuario (idUs)

Figura B.18: Tabla actividad personal. Almacena las actividades de cada usuario y sus parámetros comunes a todos los tramos y tipos de actividad. Sus atributos más importantes: idAc, clave primaria; idUsAc, clave ajena al usuario al que pertenece (en el esquema usuarios); fechaCreacAc, timestamp en el que se creó la actividad; tituloAc, el título de la actividad; fechaAc, la fecha en la que se realizó; nochesAc, el número de noches que ha durado la actividad.

Table tbl_tramo		
*	idTr	int AUTOINCREMENT
*	idTipoAcTr	int
	nombreTr	varchar(150)
	descripcionTr	text
	obsTr	text
*	fechaTr	date
	idTerrTr	int
*	idAcTr	int
*	tipoRecorridoTr	varchar(60) DEFO 'IdaVuelta'
*	tiempoTr	decimal(4, 2) DEFO 0.00
	idRutaTr	int
	idPicoTr	int
	idCorredorTr	int
	idCrestaTr	int
	idViaFTr	int
	idViaEscTr	int
*	numOrdenTr	int DEFO 0
	trackGpsTr	varchar(500)
Indexes		
Pk	pk_tbl_tramo	ON idTr
	FKCorredorTr_idx	ON idCorredorTr
	FKCrestaTr_idx	ON idCrestaTr
	FKPicoTr_idx	ON idPicoTr
	FKRutaTr_idx	ON idRutaTr
	FKTrAc_idx	ON idAcTr
	FKTrTAc_idx	ON idTipoAcTr
	FKTrTerr_idx	ON idTerrTr
	FKViaEscTr_idx	ON idViaEscTr
	FKViaFTr_idx	ON idViaFTr
Foreign Keys		
	FKTrAc	(idAcTr) ref tbl_actividadpersonal (idAc)
	FKCorredorTr	(idCorredorTr) ref tbl_corredor (idCorr)
	FKCrestaTr	(idCrestaTr) ref tbl_cresta (idCres)
	FKPicoTr	(idPicoTr) ref tbl_pico (idPico)
	FKRutaTr	(idRutaTr) ref tbl_ruta (idRu)
	FKTrTerr	(idTerrTr) ref tbl_terreno (idTer)
	FKTrTAc	(idTipoAcTr) ref tbl_tipoactividad (idTAc)
	FKViaEscTr	(idViaEscTr) ref tbl_viaesc (idVEsc)
	FKViaFTr	(idViaFTr) ref tbl_viaf (idVF)

Figura B.19: Tabla tramo. Almacena los tramos de las actividades (uno para actividades monotramo; varios para actividades multitramo). Sus atributos más importantes: idTr, clave primaria; idTipoAcTr, clave ajena del tipo de actividad; nombreTr, el nombre del tramo; descripcionTr, la descripción en texto HTML; fechaTr, fecha en la que se realizó (la misma que la de la actividad a la que pertenece si ésta es monotramo); idAcTr, clave ajena a la actividad a la que pertenece; tipoRecorridoTr; tiempoTr, el tiempo que tarda en recorrerse.

Table tbl_terreno		
*	idTer	int AUTOINCREMENT
*	nombreTer	varchar(45)
*	valorTer	varchar(45)
Indexes		
Pk	pk_tbl_terreno	ON idTer
U	nombreTer_UNIQUE	ON nombreTer
U	valorTer_UNIQUE	ON valorTer

Figura B.20: Tabla terreno. Almacena los tipos de terreno (por ejemplo: nieve, hielo, pedrera, etc). No suele variar. Sus atributos más importante: idTer, clave primaria; nombreTer, nombre que representa el tipo de terreno.

Table tbl_tipoactividad		
*	idTAc	int AUTOINCREMENT
*	nombreTAc	varchar(45)
*	valorTAc	varchar(45)
*	nombreGrTAc	varchar(45)
*	deTramTAc	int
*	deAcTAc	int
*	deRutaTAc	int
Indexes		
Pk	pk_tbl_tipoactividad	ON idTAc

Figura B.21: Tabla tipo actividad. Almacena los tipos de actividad en actividades y tramos. No suele variar. Sus atributos más importantes: idTAc, clave primaria; nombreTAc, nombre que representa al tipo de actividad.

Table tbl_tiporecorrido		
*	idTRec	int AUTOINCREMENT
*	nombreTRec	varchar(45)
*	valorTRec	varchar(45)
Indexes		
Pk	pk_tbl_tiporecorrido	ON idTRec

Figura B.22: Tabla tipo recorrido. Almacena los tipos de recorrido de los tramos ("Ida y vuelta", "Circular" y "Travesía"). No suele variar. Sus atributos más importantes: idTRec, clave primaria; nombreTRec, nombre que representa al tipo de actividad.

Table tbl_dificultadfisica		
*	idDF	int AUTOINCREMENT
*	nombreDF	varchar(45)
*	orden	int
Indexes		
Pk	pk_tbl_dificultadfisica	ON idDF

Figura B.23: Tabla dificultad física. Almacena los niveles de dificultad física para actividades y tramos. No suele variar. Sus atributos más importantes: idDF, clave primaria; nombreDF, nombre que representa el nivel de dificultad física.

Table tbl_dificultadtecnica		
*	idDT	int AUTOINCREMENT
*	nombreDT	varchar(45)
*	orden	int
Indexes		
Pk	pk_tbl_dificultadtecnica	ON idDT

Figura B.24: Tabla dificultad técnica. Almacena los niveles de dificultad técnica para actividades y tramos. No suele variar. Sus atributos más importantes: idDT, clave primaria; nombreDT, nombre que representa el nivel de dificultad técnica.

Table tbl_edadninos		
*	idEN	int AUTOINCREMENT
*	nombreEN	varchar(45)
	valorEN	varchar(45)
Indexes		
Pk	pk_tbl_edadninos	ON idEN

Figura B.25: Tabla edad niños. Almacena las edades de aptitud para las actividades. No suele variar. Sus atributos más importantes: idEN, clave primaria; nombreEN, nombre que representa la edad del niño o si no es apto.

Table tbl_pico		
*	idPico	int AUTOINCREMENT
*	nombrePico	varchar(150)
*	altitudPico	int
	caracterPico	text
	mtnPico	varchar(150)
	viaPico	varchar(150)
	altitudInicioPico	int
	ptoIniPico	varchar(150)
	ptoFinPico	varchar(150)
	pendMaxPico	int
	pendMedPico	int
	desnivelPosPico	int
	desnivelNegPico	int
	longitudPico	int
	dificultadFisPico	varchar(45)
	dificultadTecPico	varchar(45)
*	oficial	int DEFO 0
	idGradoGeneralPico	int
	otrosGradosPico	varchar(255)
Indexes		
Pk	pk_tbl_pico	ON idPico
	FKGGenPico	ON idGradoGeneralPico
Foreign Keys		
	FKGGenPico	(idGradoGeneralPico) ref tbl_gradogeneral (idGGen)

Figura B.26: Tabla pico. Almacena los atributos propios de un pico para aquellos tramos cuyo tipo de actividad sea pico. Sus atributos más importantes: idPico, clave primaria; nombrePico, nombre del pico; altitudPico, altitud oficial del pico.

Table tbl_corredor		
*	idCorr	int AUTOINCREMENT
*	nombreCorr	varchar(150)
	caracterCorr	text
	mtnCorr	varchar(300)
	longitudCorr	int
	desnivelCorr	int
	altitudMaxCorr	int
	pendMaxCorr	int
	pendMedCorr	int
	resenaCorr	varchar(150)
	altitudMinCorr	int
	dificultadFisCorr	varchar(45)
	dificultadTecCorr	varchar(45)
	ptoIniCorr	varchar(150)
	ptoFinCorr	varchar(150)
*	oficial	int DEFO 0
	idGradoGeneralCorr	int
	otrosGradosCorr	varchar(255)
Indexes		
Pk	pk_tbl_corredor	ON idCorr
	FKGGenCorr	ON idGradoGeneralCorr
Foreign Keys		
	FKGGenCorr	(idGradoGeneralCorr) ref tbl_gradogeneral (idGGen)

Figura B.27: Tabla corredor. Almacena los atributos propios de un corredor para aquellos tramos cuyo tipo de actividad sea corredor. Sus atributos más importantes: idCorr, clave primaria; nombreCorr, nombre del corredor.

Table tbl_cresta		
*	idCres	int AUTOINCREMENT
*	nombreCres	varchar(150)
	caracterCres	text
	mtnCres	varchar(150)
	longitudCres	int
	desnivelPosCres	int
	altitudMaxCres	int
	desnivelNegCres	int
	resenaCres	varchar(300)
	altitudMinCres	int
	dificultadFisCres	varchar(45)
	dificultadTecCres	varchar(45)
	ptoIniCres	varchar(150)
	ptoFinCres	varchar(150)
*	oficial	int DEFO 0
	idGradoGeneralCres	int
	otrosGradosCres	varchar(255)
Indexes		
Pk	pk_tbl_cresta	ON idCres
	FKGGenCres	ON idGradoGeneralCres
Foreign Keys		
	FKGGenCres	(idGradoGeneralCres) ref tbl_gradogeneral (idGGen)

Figura B.28: Tabla cresta. Almacena los atributos propios de una cresta para aquellos tramos cuyo tipo de actividad sea cresta. Sus atributos más importantes: idCres, clave primaria; nombreCres, nombre de la cresta.

Table tbl_ruta		
*	idRu	int AUTOINCREMENT
*	nombreRu	varchar(150)
	caracterRu	text
	mtnRu	varchar(150)
	altitudMaxRu	int
	altitudMinRu	int
	pendMaxRu	int
	pendMedRu	int
	ptoFinRu	varchar(150)
	ptoIniRu	varchar(150)
	longitudRu	int
*	oficial	int DEFO 0
	desnivelPosRu	int
	desnivelNegRu	int
	dificultadFisRu	varchar(45)
	dificultadTecRu	varchar(45)
*	idTipoRu	int
Indexes		
Pk	pk_tbl_ruta	ON idRu
	FKTipoRu_idx	ON idTipoRu
Foreign Keys		
	FKTipoRu	(idTipoRu) ref tbl_tipoactividad (idTAc)

Figura B.29: Tabla ruta. Almacena los atributos propios de una ruta para aquellos tramos cuyo tipo de actividad sea ruta. Sus atributos más importantes: idRu, clave primaria; nombreRuta, nombre de la ruta; idTipoRu, clave ajena de tipo de actividad (ruta es una agrupación de tipos de actividad).

Table tbl_viaf		
*	idVF	int AUTOINCREMENT
*	nombreVF	varchar(150)
*	dificultadVF	int
	caracFuerzaVF	bit
	caracPsicologicoVF	bit
	caracResistenciaVF	bit
	caracEquipamientoVF	varchar(150)
	caracTerrenoVF	bit
	distanciaAccesoF	int
	distanciaRegresoVF	int
	longitudVF	int
	desnivelEquipVF	int
	desnivelTotVF	int
	desplomesVF	int
	puentesVF	int
	tirolinaVF	int
	cuerdaVF	bit DEFO 0
	gratisVF	int DEFO 1
	idOrientacionVF	int
	idPoblacionVF	int
*	oficial	varchar(45) DEFO '0'
	resenaVF	varchar(150)
	caracterVF	text
Indexes		
Pk	pk_tbl_viaf	ON idVF
	FKdifVf_idx	ON dificultadVF
	FKOrienVF_idx	ON idOrientacionVF
	FKPobVF_idx	ON idPoblacionVF
Foreign Keys		
	FKdifVf	(dificultadVF) ref tbl_difviaf (idDifVF)
	FKPobVF	(idPoblacionVF) ref tbl_localizacion (idLoc)
	FKOrienVF	(idOrientacionVF) ref tbl_orientacion (idOri)

Figura B.30: Tabla vía ferrata. Almacena los atributos propios de una vía ferrata para aquellos tramos cuyo tipo de actividad sea vía ferrata. Sus atributos más importantes: idVF, clave primaria; nombreVF, nombre de la vía ferrata; dificultadVF, clave ajena de dificultad de vía ferrata; caracFuerzaVF, caracPsicologicoVF, caracResistenciaVF, EquipamientoVF: son claves ajenas de caracter de vía ferrata y forman la escala Hüsler. Nótese que los atributos de las vías ferratas son bastante menos técnicos que los del resto de tipos de actividad.

Table tbl_viaesc		
*	idVEsc	int AUTOINCREMENT
*	nombreVEsc	varchar(150)
	croquisVEsc	varchar(300)
*	idTipoEscalVEsc	int
	idOrientacionVEsc	int
	idGradoDeportivaVEsc	int
	idGradoArtificialVEsc	int
	idGradoMixtoVEsc	int
	idGradoHieloVEsc	int
	longitudVEsc	int
	equipamientoVEsc	text
	materialNecesarioVEsc	text
	descVEsc	text
	descensoVEsc	text
	abiertaPorVEsc	mediumtext
	anoAperturaVEsc	int
	estadoVEsc	text
*	oficial	int DEFO 0
	sectorVEsc	int
	escuelaVEsc	int
	idGradoAlpinaObVEsc	int
	otrosGrados	varchar(150)
	idGradoAlpinaLiVEsc	int
	ensambleVEsc	bit
	idGradoObligadoVEsc	int
	idGradoLibreVEsc	int
Indexes		
Pk	pk_tbl_viaesc	ON idVEsc
	FKEscuelaVEsc_idx	ON escuelaVEsc
	FKGAlpLiVEsc_idx	ON idGradoAlpinaLiVEsc
	FKGArtVEsc_idx	ON idGradoArtificialVEsc
	FKGDepVEsc_idx	ON idGradoDeportivaVEsc
	FKGHieVEsc_idx	ON idGradoHieloVEsc
	FKGLiVEsc	ON idGradoLibreVEsc
	FKGMixVEsc_idx	ON idGradoMixtoVEsc
	FKGObVEsc	ON idGradoObligadoVEsc
	FKOriVEsc_idx	ON idOrientacionVEsc
	FKSectorVEsc_idx	ON sectorVEsc
	FKTEscVEsc_idx	ON idGradoAlpinaObVEsc
Foreign Keys		
	FKEscuelaVEsc	(escuelaVEsc) ref tbl_escuelaesc (idEESc)
	FKGArtVEsc	(idGradoArtificialVEsc) ref tbl_gradoartificial (idGArt)
	FKGDepVEsc	(idGradoDeportivaVEsc) ref tbl_gradodeportiva (idGDep)
	FKGLiVEsc	(idGradoLibreVEsc) ref tbl_gradodeportiva (idGDep)
	FKGObVEsc	(idGradoObligadoVEsc) ref tbl_gradodeportiva (idGDep)
	FKTEscVEsc	(idGradoAlpinaObVEsc) ref tbl_gradogeneral (idGGen)
	FKGHieVEsc	(idGradoHieloVEsc) ref tbl_gradohielo (idGHie)
	FKGMixVEsc	(idGradoMixtoVEsc) ref tbl_grademixto (idGMix)
	FKOriVEsc	(idOrientacionVEsc) ref tbl_orientacion (idOri)
	FKSectorVEsc	(sectorVEsc) ref tbl_sectoresc (idSectorEsc)

Figura B.31: Tabla vía escalada. Almacena los atributos propios de una escalada para aquellos tramos cuyo tipo de actividad sea escalada. Sus atributos más importantes: idVEsc, clave primaria; nombreVEsc, nombre de la vía escalada; idTipoEscalVEsc, clave ajena de tipo de escalada; sectorVEsc, clave ajena de sector de escalada; escuelaVEsc, clave ajena de escuela de escalada. Además de esos atributos, están los referentes a los grados de dificultad, posiblemente más difíciles de comprender. Todos ellos son claves ajenas: idGradoObligadoVEsc, idGradoLibreVEsc: los grados obligados y libres para todos los tipos de escalada; idGradoDeportivaVEsc, idGradoArtificialVEsc, idGradoMixtoVEsc, idGradoHieloVEsc, idGradoAlpinaObVEsc, idGradoAlpinaLiVEsc: grados específicos del tipo de escalada.

Table tbl_difviaf		
*	idDifVF	int AUTOINCREMENT
*	nombreDifVF	varchar(45)
Indexes		
Pk	pk_tbl_difviaf	ON idDifVF

Figura B.32: Tabla dificultad vía ferrata. Almacena los niveles de dificultad general de los tramos cuyo tipo de actividad sea vía ferrata (K1, K2, K3, K4, K5, K6). No suele variar. Sus atributos: idDifVF, clave primaria; nombreDifVF, nombre que representa el nivel de dificultad general de la vía ferrata.

Table tbl_caractviaf		
*	idCaracVF	int AUTOINCREMENT
*	nombreCaracVF	varchar(45)
Indexes		
Pk	pk_tbl_caractviaf	ON idCaracVF

Figura B.33: Tabla caracter vía ferrata. Almacena los niveles de la nueva escala Hüsler (K1, K2, K3, K4, K5, K6). No suele variar. Sus atributos: idCaracVF, clave primaria; nombreCaracVF, nombre que representa el nivel de dificultad.

Table tbl_gradogeneral		
*	idGGen	int AUTOINCREMENT
*	nombreGGen	varchar(45)
Indexes		
Pk	pk_tbl_gradogeneral	ON idGGen

Figura B.34: Tabla grado general. Almacena los grados de la escalada clásica o alpina. Se utiliza para calificar todos los tipos de escalada. Las escaladas que no son de tipo alpina, pueden utilizar su propio grado. No suele variar. Sus atributos: idGGen, clave primaria; nombreGGen, nombre del grado general.

Table tbl_gradodeportiva		
*	idGDep	int AUTOINCREMENT
*	nombreGDep	varchar(45)
Indexes		
Pk	pk_tbl_gradodeportiva	ON idGDep

Figura B.35: Tabla grado deportiva. Almacena los grados de la escalada deportiva. No suele variar. Sus atributos: idGDep, clave primaria; nombreGDep, nombre del grado para la escalada deportiva.

Table tbl_gradoartificial		
*	idGArt	int AUTOINCREMENT
*	nombreGArt	varchar(45)
Indexes		
Pk	pk_tbl_gradoartificial	ON idGArt

Figura B.36: Tabla grado artificial. Almacena los grados de la escalada artificial. No suele variar. Sus atributos: idGArt, clave primaria; nombreGArt, nombre del grado para la escalada artificial.

Table tbl_gradomixto		
*	idGMix	int AUTOINCREMENT
*	nombreGMix	varchar(45)
Indexes		
Pk	pk_tbl_gradomixto	ON idGMix

Figura B.37: Tabla grado mixto. Almacena los grados de las escaladas mixtas. No suele variar. Sus atributos: idGMix, clave primaria; nombreGMix, nombre del grado para las escaladas mixtas.

Table tbl_gradohielo		
*	idGHie	int AUTOINCREMENT
*	nombreGHie	varchar(45)
Indexes		
Pk	pk_tbl_gradohielo	ON idGHie

Figura B.38: Tabla grado hielo. Almacena los grados de la escalada en hielo. No suele variar. Sus atributos: idGHie, clave primaria; nombreGHie, nombre del grado para la escalada en hielo.

Table tbl_tipoescalada		
*	idTEsc	int AUTOINCREMENT
*	nombreTEsc	varchar(45)
Indexes		
Pk	pk_tbl_tipoescalada	ON idTEsc

Figura B.39: Tabla tipo escalada. Almacena los tipos de escalada de los tramos. No suele variar. Sus atributos: idTEsc, clave primaria; nombreTEsc, nombre que representa el tipo de escalada.

Table tbl_orientacion		
*	idOri	int AUTOINCREMENT
	nombreOri	varchar(45)
Indexes		
Pk	pk_tbl_orientacion	ON idOri

Figura B.40: Tabla orientación. Almacena los puntos cardinales. No suele variar. Sus atributos: idOri, clave primaria; nombreOri, nombre que representa el punto cardinal.

Table tbl_tiporoca		
*	idTRoc	int AUTOINCREMENT
*	nombreTRoc	varchar(60)
Indexes		
Pk	pk_tbl_tiporoca	ON idTRoc

Figura B.41: Tabla tipo roca. Almacena los tipos de roca de los sectores de escalada. No suele variar. Sus atributos: idTRoc, clave primaria; nombreTRoc, nombre que representa el tipo de roca.

Table tbl_escuelaesc		
*	idEESc	int AUTOINCREMENT
*	NombreEEsc	varchar(150)
	accesoEEsc	text
	DescEEsc	text
	latEEsc	varchar(45)
	longEEsc	varchar(45)
	creadoEEsc	datetime
	modificadoEEsc	datetime
	escuelaEscOf	int DEFO 1
*	idPobEEsc	int
Indexes		
Pk	pk_tbl_escuelaesc	ON idEESc
	FKPobEEsc_idx	ON idPobEEsc
Foreign Keys		
	FOPoblacionEEsc	(idPobEEsc) ref tbl_localizacion (idLoc)

Figura B.42: Tabla escuela escalada. Almacena los atributos de una escuela de escalada. Sus atributos más importantes: idEEsc, clave primaria; NombreEEsc, nombre de la escuela de escalada; idPobEEsc, clave ajena de población a la que pertenece.

Table tbl_sectoresc		
*	idSectorEsc	int AUTOINCREMENT
*	nombreSEsc	varchar(150)
	idTipoRocaSEsc	int
	DescSEsc	text
	latSEsc	varchar(45)
	longSEsc	varchar(45)
	accesoSEsc	text
	tiempoAccesoSEsc	decimal(4, 2)
	creadoSEsc	datetime
	modificadoSEsc	datetime
	sectorEscOf	int DEFO 1
*	escuelaSEsc	int
Indexes		
Pk	pk_tbl_sectoresc	ON idSectorEsc
	FKEscSEsc_idx	ON escuelaSEsc
	FKTipoRocaSEs_idx	ON idTipoRocaSEsc
Foreign Keys		
	FKEscuelaSEs	(escuelaSEsc) ref tbl_escuelaesc (idEESc)
	FKTipoRocaSEs	(idTipoRocaSEsc) ref tbl_tiporoca (idTRoc)

Figura B.43: Tabla sector escalada. Almacena los atributos de un sector de escalada. Sus atributos más importantes: idSectorEsc, clave primaria; nombreSEsc, nombre del sector de escalada; idTipoRocaSEsc, clave ajena de tipo de roca que compone el sector; escuelaSEsc, clave ajena de la escuela a la que pertenece.

Table tbl_largoesc		
*	idLEsc	int AUTOINCREMENT
	numeroOrdenLEsc	int
	longitudLEsc	int
	descripcionLEsc	text
	idGradoDeportivaLEsc	int
	idGradoArtificialLEsc	int
	idGradoMixtoLEsc	int
	idGradoHieloLEsc	int
	croquisLEsc	varchar(150)
*	largoEscOf	int DEFO 0
*	viaELEsc	int
	otrosGradosLEsc	varchar(255)
	ensambleLEsc	bit
	idGradoObLEsc	int
	idGradoLiLEsc	int
Indexes		
Pk	pk_tbl_largoesc	ON idLEsc
	FKGArtLEsc_idx	ON idGradoArtificialLEsc
	FKGDepLEsc_idx	ON idGradoDeportivaLEsc
	FKGHieLEsc_idx	ON idGradoHieloLEsc
	FKGLiLEsc	ON idGradoLiLEsc
	FKGMixLEsc_idx	ON idGradoMixtoLEsc
	FKGObLEsc	ON idGradoObLEsc
	FKViaEsLEsc_idx	ON viaELEsc
Foreign Keys		
	FKGArtLEsc	(idGradoArtificialLEsc) ref tbl_gradoartificial (idGArt)
	FKGDepLEsc	(idGradoDeportivaLEsc) ref tbl_gradodeportiva (idGDep)
	FKGLiLEsc	(idGradoLiLEsc) ref tbl_gradodeportiva (idGDep)
	FKGObLEsc	(idGradoObLEsc) ref tbl_gradodeportiva (idGDep)
	FKGHieLEsc	(idGradoHieloLEsc) ref tbl_gradohielo (idGHie)
	FKGMixLEsc	(idGradoMixtoLEsc) ref tbl_gradomixto (idGMix)
	FKViaEsLEsc	(viaELEsc) ref tbl_viaesc (idVEsc)

Figura B.44: Tabla largo escalada. Almacena los atributos de un largo de escalada. Sus atributos más importantes: idLEsc, clave primaria; numeroOrdenLEsc, posición de este largo en la vía de escalada a la que pertenece; viaELEsc, clave ajena de vía de escalada a la que pertenece este largo.

Table tbl_imgs		
*	idImg	int AUTOINCREMENT
*	nombreImg	varchar(200)
*	idTramoImg	int
*	creadaImg	datetime
*	editadaImg	datetime
*	idUsImg	int
Indexes		
Pk	pk_tbl_imgs	ON idImg
	FKTramoImg_idx	ON idTramoImg
	FKUsImg_idx	ON idUsImg
Foreign Keys		
	FKTramoImg	(idTramoImg) ref tbl_tramo (idTr)
	FKUsImg	(idUsImg) ref tbl_usuario (idUs)

Figura B.45: Tabla imágenes. Almacena las imágenes (en realidad almacena sus nombres, no el fichero binario) adjuntadas en las descripciones de los tramos de las actividades. Sus atributos más importantes: idImg, clave primaria; nombreImg, nombre del fichero de imagen; idTramoImg, clave ajena del tramo en cuya descripción se ha insertado la imagen; creadaImg, editadaImg: timestamps de creación y última modificación de la imagen; idUsImg, clave ajena del usuario al que pertenece la imagen.

Table tbl_localizacion		
*	idLoc	int AUTOINCREMENT
*	nombreLoc	varchar(60)
*	idProvLoc	int
Indexes		
Pk	pk_tbl_localizacion	ON idLoc
	FKProPob_idx	ON idProvLoc
Foreign Keys		
	FKProPob	(idProvLoc) ref tbl_provincia (idProv)

Figura B.46: Tabla localización. Almacena las localizaciones consultadas en los *web services* de Geonames. Esta tabla desaparecerá por migración. Sus atributos: idLoc, clave primaria; nombreLoc, nombre de la localización; idProvLoc, clave ajena de la provincia a la que pertenece.

Table tbl_provincia		
*	idProv	int AUTOINCREMENT
*	nombreProv	varchar(60)
*	idPaisProv	int
Indexes		
Pk	pk_tbl_provincia	ON idProv
	FKpaisProv_idx	ON idPaisProv
Foreign Keys		
	FKpaisProv	(idPaisProv) ref tbl_pais (idPais)

Figura B.47: Tabla provincia. Almacena las provincias consultadas en los *web services* de Geonames. Esta tabla desaparecerá por migración. Sus atributos: idProv, clave primaria; nombreProv, nombre de la provincia; idPaisProv, clave ajena del país al que pertenece.

Table tbl_pais		
*	idPais	int AUTOINCREMENT
	nombrePais	varchar(60)
Indexes		
Pk	pk_tbl_pais	ON idPais

Figura B.48: Tabla país. Almacena los países consultados en los *web services* de Geonames. Esta tabla desaparecerá por migración. Sus atributos: idPais, clave primaria; nombrePais, nombre del país.

Table tbl_geonames		
*	idGeo	int UNSIGNED
	nombreGeo	varchar(200)
	asciiNombreGeo	varchar(200)
	nombreAlternGeo	varchar(8000)
	latGeo	float(12, 0)
	lonGeo	float(12, 0)
	featClassGeo	char(1)
	featCodGeo	varchar(10)
	codigoPaisGeo	char(2)
	cc2Geo	char(60)
	admin1CodGeo	varchar(20)
	admin2CodGeo	varchar(80)
	admin3CodGeo	varchar(20)
	admin4CodGeo	varchar(20)
	poblacionGeo	bigint
	elevacionGeo	int
	demGeo	int
	timezoneGeo	varchar(40)
	modificadoGeo	date
Indexes		
Pk	pk_tbl_geonames	ON idGeo

Figura B.49: Tabla geonames. Almacena un enorme conjunto de localizaciones de todo el mundo. Pertenece a la base de datos geográfica Geonames, la cual se ha integrado en ambos esquemas. Esta tabla sustituirá a las de localización, provincia y país. Sus atributos más importantes: idGeo, clave primaria; nombreGeo, nombre del punto geográfico; latGeo, lonGeo: coordenadas del punto geográfico; codigoPaisGeo, código del país al que pertenece. Para más información sobre esta tabla [5].

Table tbl_usuario		
*	idUs	int AUTOINCREMENT
*	nombreUs	varchar(45)
	apellidosUs	varchar(80)
*	correoUs	varchar(45)
*	contrasenaUs	varchar(150)
	imgUs	varchar(201)
*	ultimoAccesoUs	datetime
*	fechaCreaUs	datetime
	idPobUs	int
*	primeraVezUs	bit DEFO 1
	primeraVezPanelUs	bit DEFO 1
*	primeraActivUs	bit DEFO 1
*	proUs	int DEFO 0
	esEditorUs	bit DEFO 0
	esGuiaUs	bit DEFO 0
	perfilGuiaUs	bit DEFO 0
*	teamUs	int DEFO 0
*	destacadoUs	int UNSIGNED DEFO 0
	idFacebookUs	varchar(100)
	idTwitterUs	varchar(100)
	clienteStripe	varchar(45)
	idSuscripProStripe	varchar(45)
	clienteStripeGuia	varchar(45)
	idSuscripGuiStripe	varchar(45)
	clienteStripeEditor	varchar(45)
	opOcCorrUs	bit DEFO 0
	opCorrValAc	bit DEFO 1
	opCorrMsgPriv	bit DEFO 1
	opProRenovAut	bit DEFO 1
	opConteContac	bit DEFO 1
	opEnvioMesUs	bit DEFO 1

	fechaLimitePro	datetime
	fechaLimiteGuia	datetime
	enviadoMailAvisoCaduc	bit DEFO 0
	enviadoMailRenovac	bit DEFO 0
	opGuiaRenovAut	bit DEFO 1
	opEnvioEstadGuiUs	bit DEFO 1
	enviadoMailAvisoCaducGuia	bit DEFO 0
	enviadoMailRenovacGuia	bit DEFO 0
	aceptCookie	bit DEFO 0
	novedadGr	bit DEFO 0
	opCorrUsGr	bit DEFO 1
	opCorrUsSal	bit DEFO 1
	facturadoUs	bit DEFO 0
	esAgrupacionUs	bit DEFO 0
	conNinosUs	bit DEFO 0
	perfilEditorUs	bit DEFO 0
	nombreFacUs	varchar(100)
	direccionFacUs	varchar(200)
	documentoFacUs	varchar(45)
Indexes		
Pk	pk_tbl_usuario	ON idUs
U	correoUs_UNIQUE	ON correoUs
	FKPob_idx	ON idPobUs
Foreign Keys		
	FKPoblacionUs	(idPobUs) ref tbl_localizacion (idLoc)

Figura B.50: Tabla usuarios. Almacena los usuarios creados y todos sus atributos. Sus atributos más importantes: idUs, clave primaria; nombreUs, nombre del usuario; apellidosUs, apellidos del usuario; correoUs, correo electrónico con el que se registró; contrasenaUs: hash de la contraseña con la que el usuario se registró; imgUs, imagen del perfil del usuario; fechaCreacUs, timestamp del registro del usuario; ultimoAccesoUs, timestamp del último acceso del usuario.

Apéndice C

Manuales

C.1. Manual de instalación de myssummon

Uno de los objetivos de myssummon es que pueda funcionar bajo diferentes plataformas. Si bien en la implementación esto se ha logrado gracias al entorno multiplataforma Java, el instalador ha de ser propio y adecuado a cada plataforma.

Debido al gran número de sistemas operativos de tipo *nix* (Debian, Red Hat, Mac os, Solaris, etc.), no existe hasta el momento un instalador para estos entornos. Aunque todos pertenezcan a la misma familia, siempre hay diferencias en cuanto a rutas.

Windows Para sistemas Windows la instalación es muy sencilla. Se ha construido un instalado ejecutable que contiene todos los ficheros necesarios para utilizar la aplicación. Estos son los pasos necesarios para instalar myssummon en Windows:

1. Ejecutar el instalador haciendo *doble clic*.
2. Aparecerá la ventana de instalación (Figura C.1). Hacer clic en "Siguiente".
3. Escoger la ruta de instalación (valor por defecto recomendado). Hacer clic en "Siguiente".
4. Escoger la carpeta de Menú Inicio bajo la que instalarla (valor por defecto recomendado). Hacer clic en "Instalar".
5. Se ejecuta la instalación (Figura C.2). Si no se desea ejecutar el programa al cerrar el instalador, desmarcar la casilla "Ejecutando myssummon". Hacer clic en "Terminar".

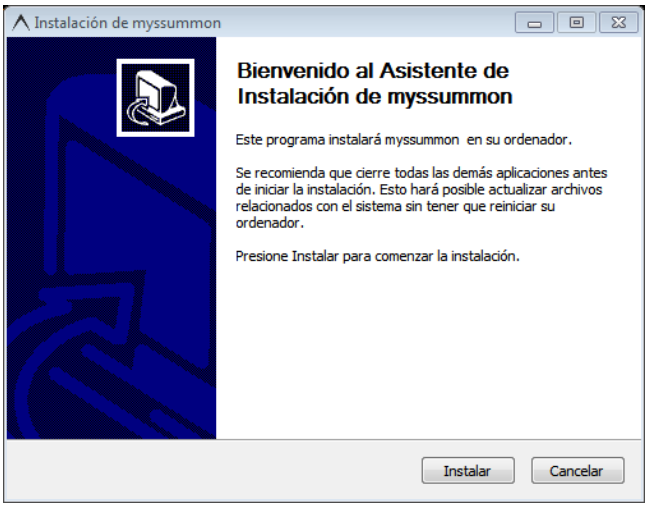


Figura C.1: Ventana inicial de instalación.

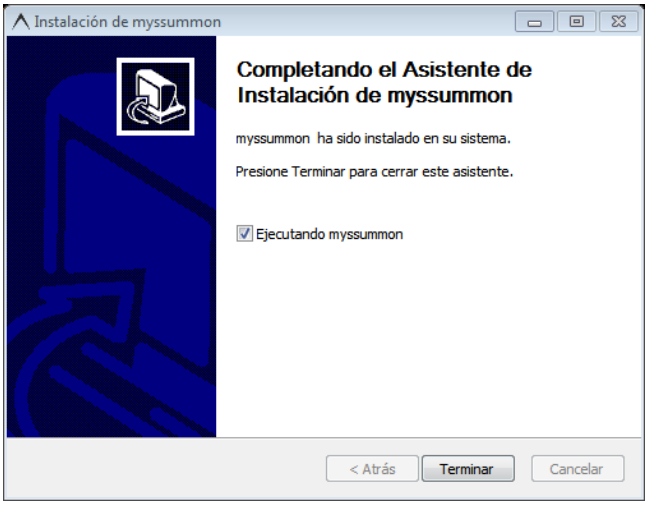


Figura C.2: Ventana final de instalación.

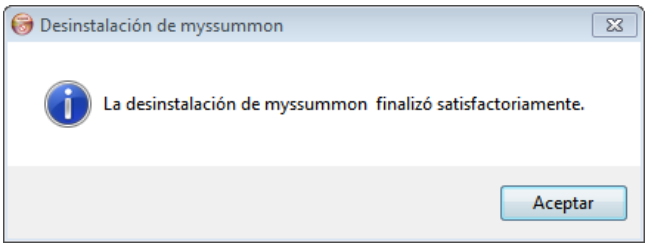


Figura C.3: Diálogo de confirmación de la desinstalación.

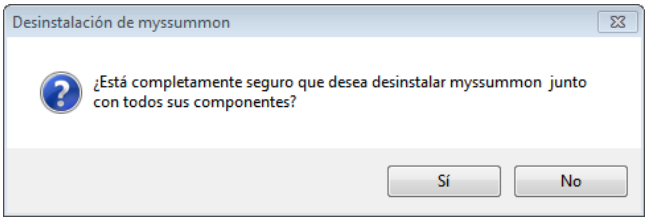


Figura C.4: Desinstalación finalizada.

El instalador crea un acceso directo para desinstalar la aplicación. Se puede encontrar en el directorio de instalación de myssummon o bajo la carpeta de myssummon en el menú inicio. La desinstación es como sigue:

1. Ejecutar el desinstalador haciendo *doble clic*.
2. Aparecerá un diálogo de confirmación de la desinstalación (Figura C.3). Hacer clic en "Si".
3. Se ejecuta la desinstalación y se informará de ello en un diálogo (Figura C.4). Hacer clic en "Aceptar".

Sistemas *nix* Aunque no exista un instalador como lo hay para la plataforma Windows, la puesta en marcha de la aplicación en entornos *nix* no es una tarea complicada. Basta con seguir estos pasos:

1. Copiar los ficheros de instalación (Tabla C.1) en el directorio desde el que se va a ejecutar la aplicación.
2. Abrir con un editor de texto el fichero de configuración *conf.properties*.
3. Modificar la primera línea del fichero apuntando a la ruta donde se encuentran los ficheros. Por ejemplo:

```
BASE_PATH = /home/cristofer/Escritorio/Entorno_myssummon/
```

4. Abrir un terminal en la ruta donde se han copiado los ficheros la siguiente instrucción:

```
java -jar myssummon.jar
```

NOMBRE	DESCRIPCIÓN
conf.properties	Rutas y propiedades de la aplicación.
myssummon.jar	Artefacto JAR de la aplicación.
usuarios.db	Base de datos de usuarios.
proto/	Directorio que contiene los prototipos de base de datos para crear un entorno personal.

Tabla C.1: Ficheros de instalación en sistemas *nix*.

C.2. Manual de usuario de myssummon

En esta sección se explica como llevar a cabo las operaciones implementadas en esta versión de myssummon. Hay que recordar que ésta es una versión de demostración y que por tanto, no está disponible todo el conjunto de operaciones que fueron diseñadas. Las operaciones disponibles pueden consultarse en Tabla C.2.

NOMBRE	DESCRIPCIÓN
Autenticar usuario	Iniciar sesión con un usuario existente
Registrar usuario	Crear nuevo usuario e iniciar sesión
Crear actividad	Crear nueva actividad de un tramo
Ver panel	Visualizar todas las actividades del usuario
Ver actividades	Visualizar los detalles de una actividad
Ver ficha personal	Visualizar los datos del usuario
Ver estadísticas	Visualizar estadísticas locales sobre las actividades de un usuario
Eliminar actividad	Eliminar una actividad existente
Exportar actividades	Exporta todas las actividades en un fichero CSV
Cerrar sesión	Cerrar sesión del usuario actual y desconectar su base de datos

Tabla C.2: Operaciones disponibles en versión de demostración.

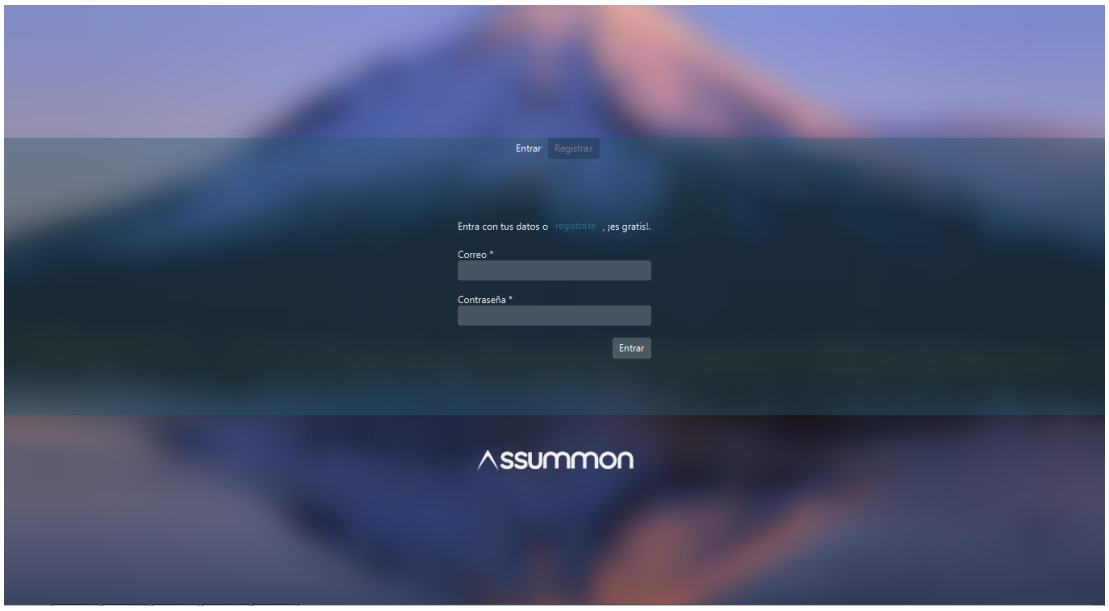


Figura C.5: Pantalla de autenticación de usuario.

C.2.1. Gestión de usuario

Este apartado explica en detalle el proceso de realización de todas las operaciones relacionadas con el usuario y su gestión.

Autenticar usuario Es la primera operación que se debe realizar en cuanto se ejecuta la aplicación. Para autenticar un usuario en la aplicación este debe existir, y para ello hemos tenido que crearlo antes (pag. 123).

Al autenticar un usuario se comprobará su existencia y, de ser así, se cargará el contenido de su base de datos y se mostrará el panel con todas sus actividades.

Los pasos a seguir son:

PASO 1: Introducir el correo y la contraseña con los que se registró en sus cuadros (Figura C.5).

PASO 2: Hacer clic en “Entrar”.

Registrar usuario Antes de poder autenticar un usuario (pag. 123), éste debe registrarse. Se creará un usuario nuevo con los datos que se especifiquen y se introducirá en la base de datos de usuarios. Igual que en la operación de autenticación, se mostrará el panel de actividades, pero vacío en este caso.

Los pasos a seguir son:

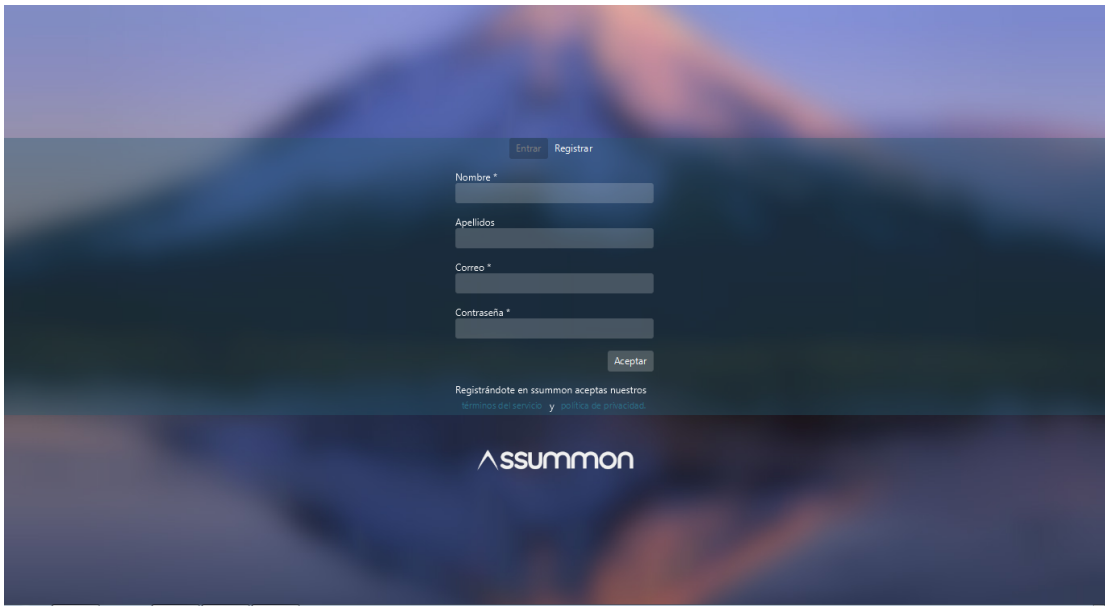


Figura C.6: Pantalla de registro de usuario.

PASO 1: Hacer clic en la pestaña "Registrar". Aparecerá el formulario de registro (Figura C.6).

PASO 2: Rellenar los campos y hacer clic en "Aceptar". Desde la misma pantalla se pueden leer los **términos del servicio** y la **política de privacidad.**

Ver ficha personal La ficha personal contiene atributos del usuario como su correo, antigüedad, localización y número de actividades creadas por él. Para consultar esta información se deben seguir los siguientes pasos:

PASO 1: Estando autenticados o registrados, desde cualquier pantalla de la aplicación, hacer clic en "Tu espacio", bajo el menú de usuario.

PASO 2: En la pantalla que aparece, hacer clic en "Ficha personal" (Figura C.7).

Ver estadísticas Ésta operación es similar a la anterior. Las estadísticas del usuario contienen información interesante sobre las actividades que ha creado, como el número de actividades de cada tipo, la longitud máxima, etc.

Los pasos a seguir son:

PASO 1: Estando autenticados o registrados, desde cualquier pantalla de la aplicación, hacer clic en "Tu espacio", bajo el menú de usuario.

PASO 2: En la pantalla que aparece, hacer clic en "Estadísticas" (Figura C.8).

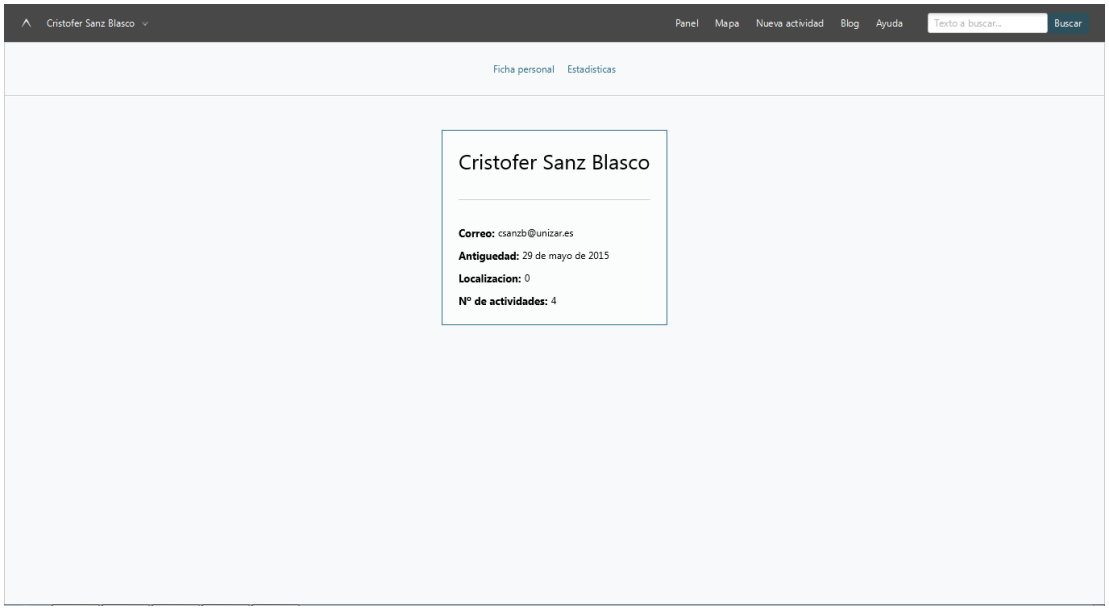


Figura C.7: Pantalla de ficha de perfil de usuario.

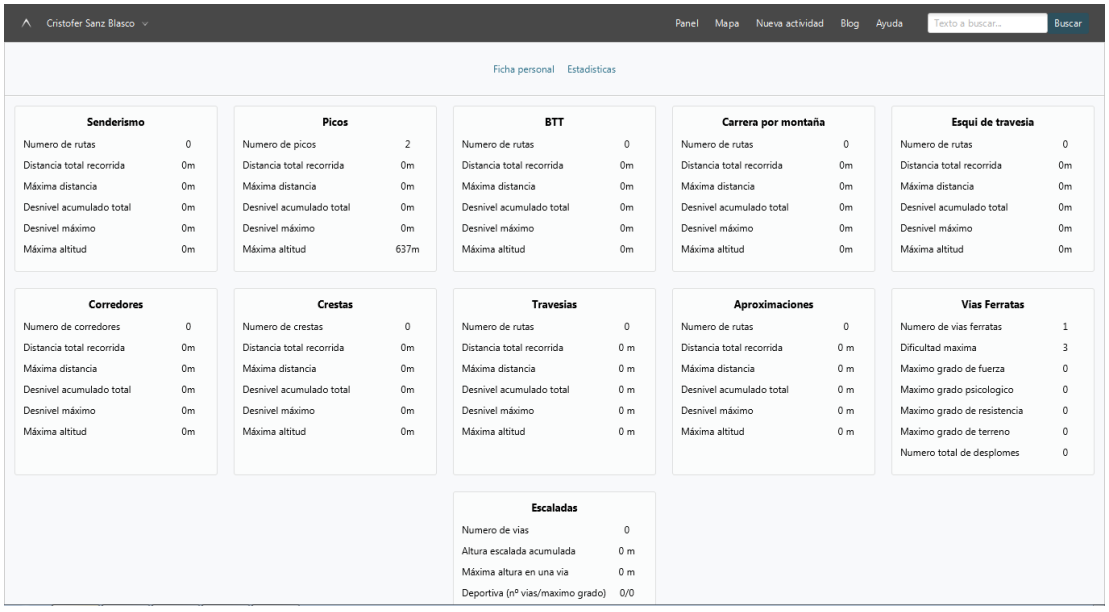


Figura C.8: Pantalla de estadísticas del usuario.

Cerrar sesión Al cerrar sesión la aplicación se desconecta de la base de datos de contenidos del usuario. Deja de haber un usuario activo y se muestra de nuevo la pantalla de autenticación (Figura C.5.

Para cerrar sesión estando autenticados o registrados, desde cualquier pantalla de la aplicación, se hace clic en "Salir", bajo el menú de usuario.

C.2.2. Gestión de actividades

Crear actividad Posiblemente la creación de una actividad es la operación más importante de la aplicación. La limitación en esta versión con respecto a la gestión de actividades es que sólo existen actividades de un tramo.

Para crear una actividad, los pasos a seguir son:

PASO 1: Estando autenticados o registrados, desde cualquier pantalla de la aplicación, hacer clic en "Nueva actividad", en el menú superior.

PASO 2: Aparecerá la pantalla de creación de actividad dividida en tres columnas (Figura C.9). Según se vayan rellenando los datos se irá mostrando más contenido. Una vez rellenados todos los campos, hacer clic en "Siguiente".

PASO 3: Esta vez aparecerá la pantalla de redacción de actividad (Figura C.10), donde se deben especificar los datos concretos del tipo de actividad que se haya elegido. También se debe proporcionar la información relativa al tramo, como son el nombre y la descripción.

PASO 4: Si se desea adjuntar una imagen de portada, hacer clic en "Cargar imagen de portada", en la esquina superior derecha de la pantalla. Localizar la imagen en la ventana de selección de ficheros que aparece (Figura C.11).

PASO 5: Hacer clic en "Finalizar". Aparecerá la vista de la actividad recién creada.

Ver panel En el panel se pueden ver todas las actividades que el usuario ha creado. Se accede a él cada vez que se autentica o registra un usuario. Para ir a la pantalla del panel (Figura C.12) basta con hacer clic en el botón "Panel" en el menú superior.

Ver actividades Aunque el panel muestra el conjunto de todas las actividades pertenecientes al usuario, lo hace presentando muy poca información sobre ellas. Si se quieren explorar los atributos y detalles de una actividad, se ha de acceder a su vista. Para ello, desde el panel, hacer clic en el título de la actividad que se quiera explorar. Se mostrará la vista de la actividad (**??**).

Eliminar actividad Si se desea eliminar una actividad, los pasos a seguir son:

PASO 1: Desde la vista de la actividad, hacer clic en "Borrar", bajo la portada de la actividad.

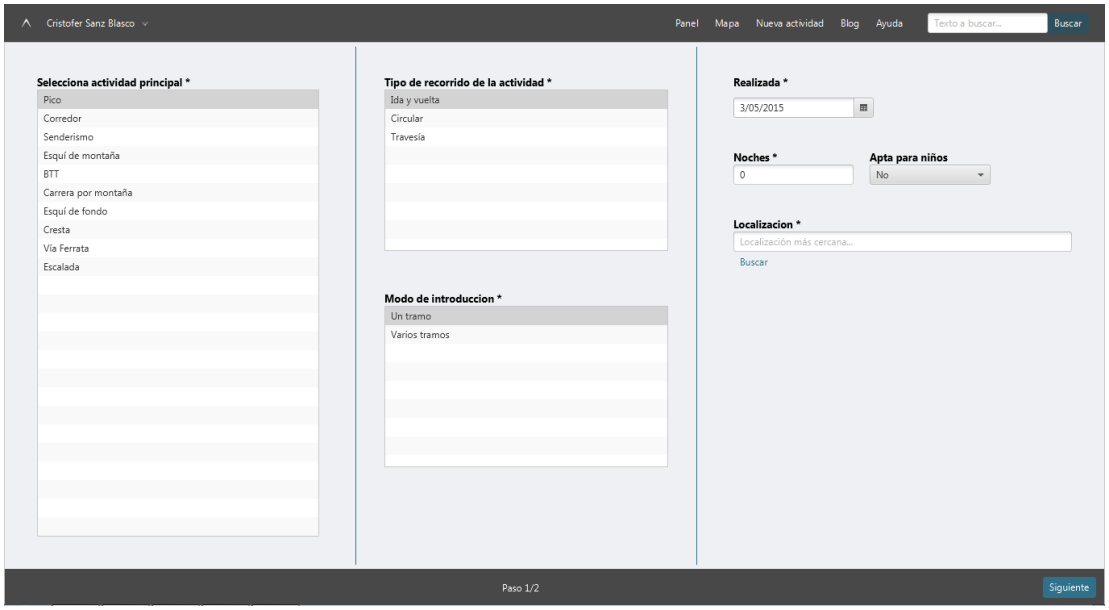


Figura C.9: Pantalla de nueva actividad.

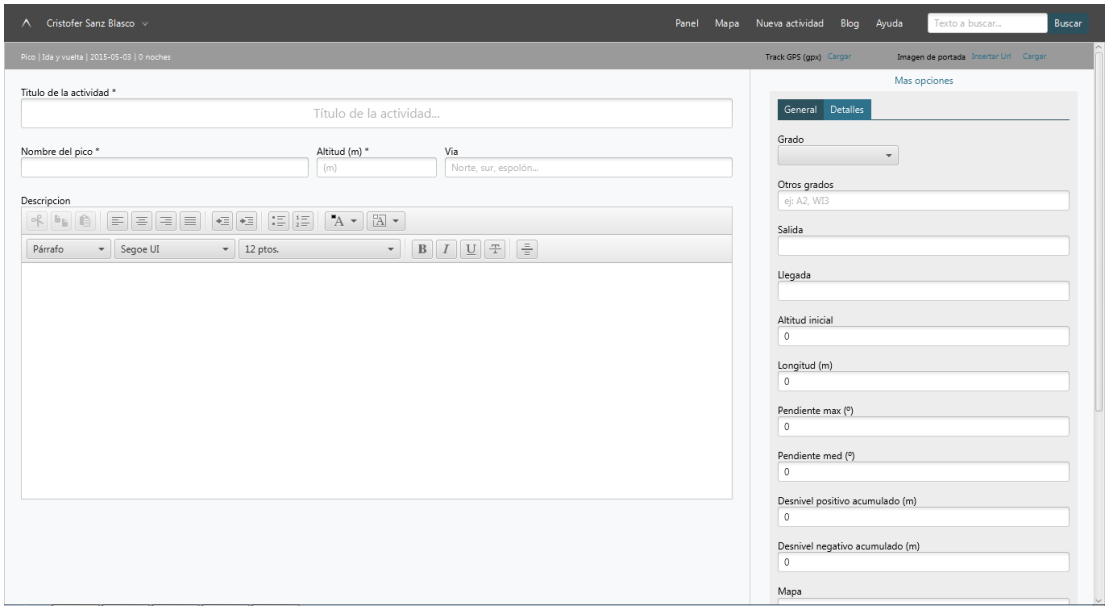


Figura C.10: Pantalla de redacción de actividad.

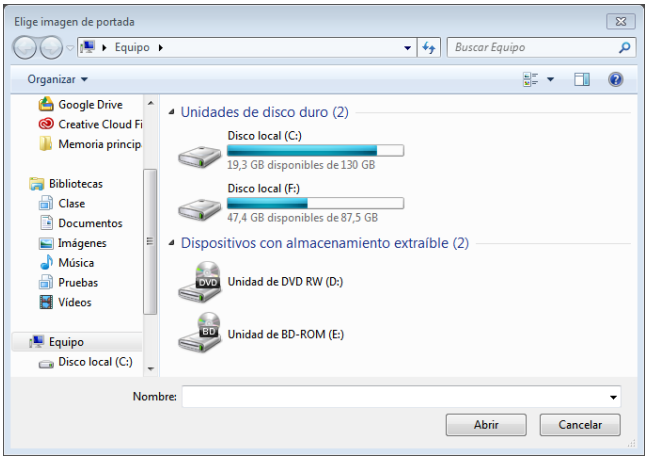


Figura C.11: Diálogo de selección de imagen de portada.

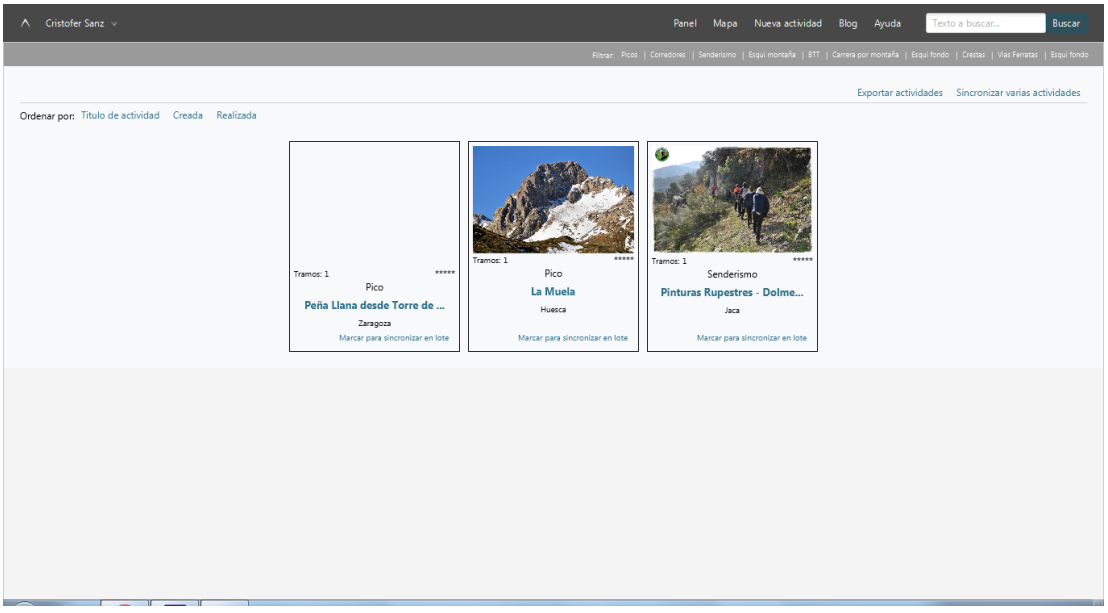


Figura C.12: Pantalla de panel.

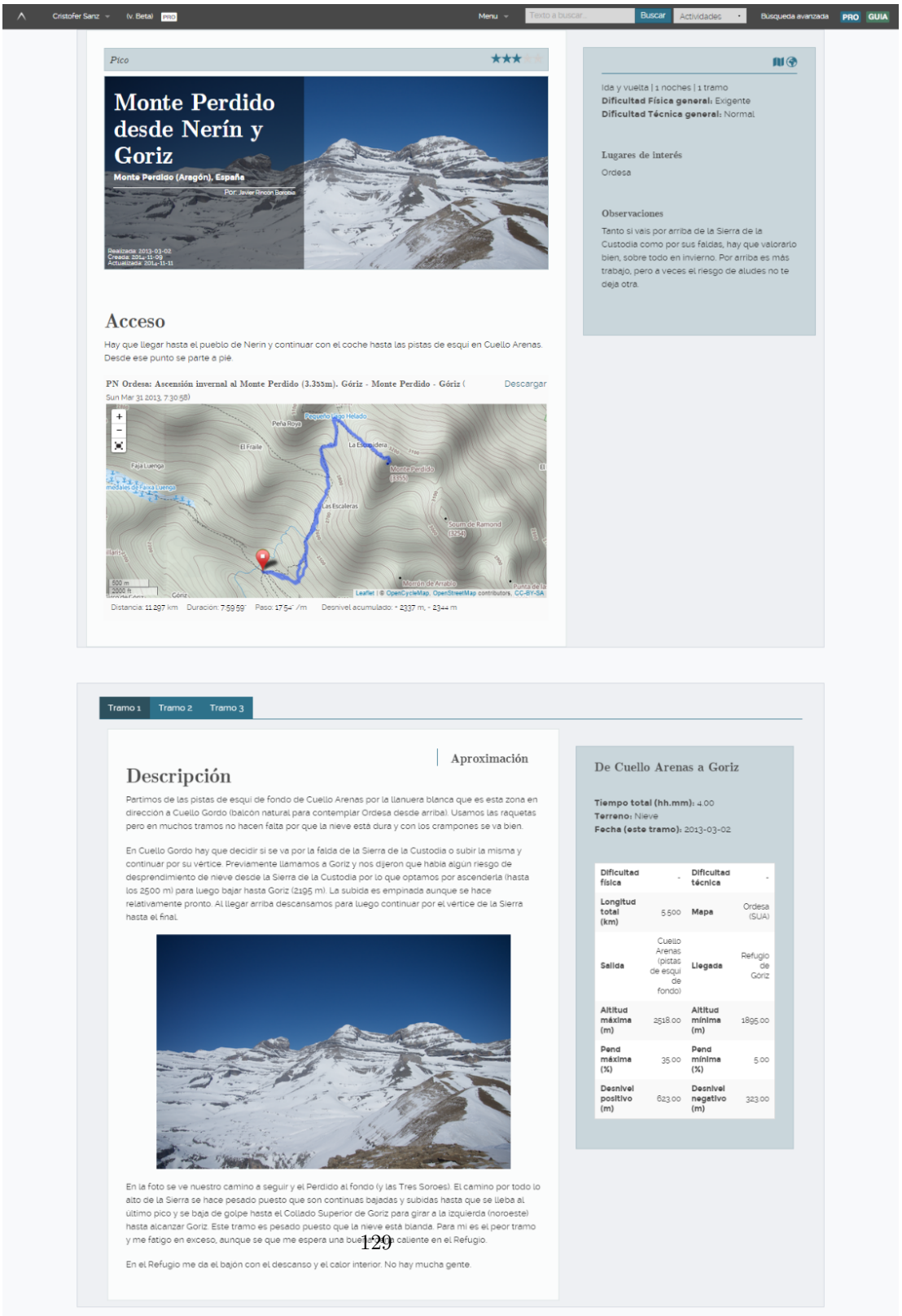


Figura C.13: Pantalla de vista de actividad.

PASO 2: Se mostrará un diálogo de confirmación, informando de que es una operación destructiva. Confirmar la eliminación.

Exportar actividades Esta función genera un fichero CSV que contiene todas las actividades del usuario. Los pasos a seguir son:

PASO 1: Desde el panel, hacer clic en "Exportar actividades", en la esquina superior derecha, bajo el menú superior.

PASO 2: Se mostrará una ventana que permite elegir la ruta donde guardar el fichero. Una vez escogida la ruta y escrito el nombre del fichero, hacer clic en "Guardar".

Bibliografía

[1] C. Desrosiers y G. Karypis. «A Comprehensive Survey of Neighborhood-based Recommendation Methods». En: *Recommender Systems Handbook*. Ed. por Francesco Ricci y col. Springer US, 2011, págs. 107-144.

[2] Michael D. E. y col. «Rethinking The Recommender Research Ecosystem: Reproducibility, Openness, and LensKit». En: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. ACM, 2011, págs. 133-140.

[3] DB-Engines. *Cálculo de puntuación para ranking DB-Engines*. 2015. URL: `http://db-engines.com/en/ranking_definition`.

[4] DB-Engines. *Ranking de SGBD por su popularidad*. 2015. URL: `http://db-engines.com/en/ranking`.

[5] Geonames. *Readme for GeoNames Gazetteer extract files*. 2015. URL: `http://download.geonames.org/export/dump/readme.txt`.

[6] H2. *Licencias de H2*. 2015. URL: `http://www.h2database.com/html/license.html`.

[7] Prem M., Raymond J. M. y Ramadass N. «Content-Boosted Collaborative Filtering for Improved Recommendations». En: *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. 2002, págs. 187-192.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2014. URL: `http://www.R-project.org`.

[9] F. Ricci, L. Rokach y B. Shapira. *Recommender systems handbook*. Springer, 2011.

[10] J. Rincon Borobia y col. «SIWAM: Using Social Data to Semantically Access the Difficulties in Mountain Activities». En: *Proceedings of the International Conference on Web Information Systems and Technologies*. SCITEPRESS, 2014, págs. 41-48.

[11] J.J. Rocchio. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Ed. por Gerard Salton. Prentice Hall, 1971.

[12] Carlos E. S. y David C. W. «Case Study Evaluation of Mahout as a Recommender Platform». En: (2012).

[13] B. Sarwar y col. «Item-based Collaborative Filtering Recommendation Algorithms». En: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. ACM, 2001, págs. 285-295.

[14] G. Shani y A. Gunawardana. «Evaluating recommendation systems.» En: *Recommender systems handbook*. Springer, 2011, págs. 257-297.

[15] SQLite. *Características de SQL no implementadas en SQLite*. 2015. URL: `http://www.sqlite.org/omitted.html`.

[16] SQLite. *Licencia de SQLite*. 2015. URL: `http://www.sqlite.org/copyright.html`.

Índice de figuras

Índice de tablas