Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DEL SOFTWARE

Sistema de organización de carreras de orientación con códigos QR

Autor:
Abel Herrero Gómez

Tutora:
Yania Crespo González-Carvajal

Agradecimientos

A mis padres y mi hermana, porque gracias a ellos he llegado hasta aquí y he descubierto la orientación.

A mis profesores de la Universidad de Valladolid, por ayudarme en mi formación durante estos cuatro años.

A María, por todo.

Resumen

En este documento se expone el proceso de diseño e implementación de un **sistema de organización de carreras de orientación** compuesto por tres partes principales: una **API REST**, una **página web** y una **aplicación móvil**.

En la página web se llevarán a cabo las tareas de organización tales como la creación de nuevas carreras y gestión de las ya existentes, entre otras. La aplicación móvil permitirá a los corredores participar en dichas carreras, registrando los puntos de control con su smartphone mediante **códigos QR** dispuestos previamente por el organizador en el entorno real.

Su propósito principal es el de servir como herramienta útil para impulsar la **promoción de la orientación** de forma más sencilla y como material docente de cara a su enseñanza en las aulas.

Abstract

This document describes the design and implementation process of an **orienteering races organization system** made up of three main parts: an **API REST**, a **web page** and a **mobile application**.

Organization tasks such as creating new races and managing existing ones will be carried out on the website, among others. The mobile application will allow runners to participate in these races, registering the checkpoints with their smartphone using **QR codes** previously arranged by the organizer in the real terrain.

Its main purpose is to serve as a useful tool to promote the **promotion of orienteering** in a simpler way and as a educational material for its teaching.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

En la actualidad, casi todo el mundo tiene acceso a Internet gracias a un smartphone u ordenador. La tecnología ha sufrido un gran avance en las últimas décadas, haciendo posible lo que antes no podía ni imaginarse, haciendo más fácil y entretenida la vida de las personas gracias a las apps, programas y juegos que se han desarrollado, y que se van a desarrollar como es el caso de este proyecto.

El deporte de la orientación es poco conocido, pero cuya promoción está en alza. Es un deporte introducido en España en la década de los años 60 en el contexto militar, pero que pasó al deportivo en 1972 gracias a un profesor del INEF de Madrid: Martín Harald Kronlund, quien incluyó la orientación en los contenidos impartidos a sus alumnos [1].

Ahora mismo, en el año 2020, hay un total de 4622 federados en España [2] y su visibilidad en los medios es cada vez mayor [3]. Este proyecto tratará de facilitar e impulsar la promoción de este deporte desde la base.

1.2. Motivación

Conocí la orientación en el colegio hace 13 años, y desde entonces me apasioné por ella. Durante mis años en el colegio e instituto fueron varios los profesores que la incluyeron en sus clases, pero o bien no la conocían de manera apropiada o no disponían de los recursos necesarios. Años después se me ocurrió la idea en la que se utilizaría un smartphone para realizar las carreras y que el contacto con el deporte fuera un poco más real. Si bien es cierto que ya existen aplicaciones con la misma temática (iOrienteering, MOBO, DIB), no se pretende mejorar ni sustituir dichas aplicaciones, sino que se trata de hacer realidad una idea personal.

1.3. Objetivos

El objetivo principal de este proyecto es la creación de herramientas para la creación, gestión y uso de las carreras creadas por los usuarios.

1.3.1. Objetivos del sistema

- Permitir crear carreras de diferentes tipos.
- Permitir la participación de usuarios en las carreras.
- Permitir la búsqueda de carreras.
- Permitir la visualización de resultados de carreras.

1.3.2. Objetivos de formación

- Aprendizaje de los framework Spring Boot e Hibernate para la creación de una API REST con persistencia.
- Aprendizaje de los framework Angular (9) y Bootstrap para la creación de la página web.
- Aprendizaje del patrón MVVM para la estructuración de la aplicación Android.

Capítulo 2

Dominio

2.1. Orientación

La orientación es una actividad deportiva que requiere exigencias físicas e intelectuales de forma constante, ya sea practicada como juego, entretenimiento o competición. Se define como una carrera individual sobre terreno variado con un recorrido determinado por una serie de controles que el deportista debe registrar realizando rutas elegidas por él mismo, sirviéndose únicamente de un mapa que describe los aspectos del terreno y una brújula [1].

Los controles (punto de paso obligatorio) dispuestos en el recorrido están marcados sobre el mapa con círculos rojos. La salida por un triángulo y la llegada por dos círculos concéntricos. Sobre el terreno se materializan en forma balizas naranjas y blancas de 30 centímetros de lado con una pinza marcadora. Los controles se colocan en lugares característicos del terreno representados sobre el mapa [1].

En el año 2020 la Federación Española de Orientación (FEDO) cuenta con 4622 deportistas federados [2], pero se estima en 20 000 la cifra de practicantes total [1].

Este deporte está dirigido para todos los públicos, tanto niños, adolescentes, adultos y personas mayores, con recorridos acordes a las posibilidades y limitaciones de cada grupo poblacional. A pesar de esto y aunque el número de participantes se va elevando cada año sigue siendo más bajo respecto al resto de deportes federados que se desarrollan en el país [4].

2.1.1. Historia

En el año 1890, el capitán del ejército sueco y scout Enest Killander se dio cuenta que utilizando los mapas y la brújula como entrenamiento de sus soldados para desplazarse por el bosque resultaba un método de instrucción excepcional y lo más importante vio como

Figura 2.1: Mapa de orientación y brújula.
Fuente: Casa Summer Mathematics Adventures.

la resolución de estos problemas les dotaba de una personalidad y un carácter diferentes: seguridad en si mismos, autocontrol en las situaciones difíciles y capacidad de decisión. Así tuvo lugar la primera competición oficial por el club nórdico de Tjalve en 1897 [1].

2.1.2. Circuitos Permanentes de Orientación

Una de las opciones más interesantes para la promoción del deporte son los llamados Circuitos Permanentes de Orientación. Estos circuitos se pueden ubicar en diferentes zonas (bosques, cascos urbanos) y de diferentes formas, pero la habitual es en forma de estacas ancladas al suelo con una pinza de marcado tradicional (ver Figura 2.2) y, de forma opcional, alguna de las tecnologías existentes como la del proyecto actual. Normalmente este tipo de proyectos, debidos a su coste de infraestructura, necesitan el apoyo de un ayuntamiento o diputación en conjunto con un club de orientación.

Recientemente el orientador murciano Juan Carlos Escaravajal Rodríguez realizó un trabajo de investigación sobre los circuitos permanentes en España [5]. En él se recoge un listado de circuitos por comunidades autónomas, prototipos de controles y una serie de gráficos con su evolución y características.

Como puede apreciarse en la Figura 2.3 existe una tendencia a la alta en la creación de circuitos permanentes. Las comunidades principales son Cataluña y Aragón, que de forma conjunta reúnen más de la mitad de los circuitos existentes. En cuanto al sistema de control (cómo registran el paso por los controles), los sistemas mayoritariamente utilizados son los artefactos visuales y la pinza tradicional de orientación. El tercer lugar lo ocupa la aplicación iOrienteering [6], semejante a la de este proyecto.

Figura 2.2: Baliza de circuito permanente
Fuente: e-proarte.com

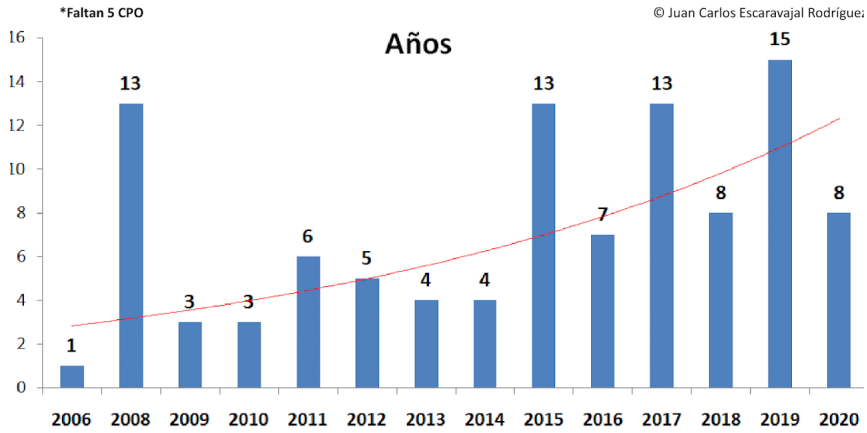

Figura 2.3: Evolución de los circuitos permanentes en España [5].

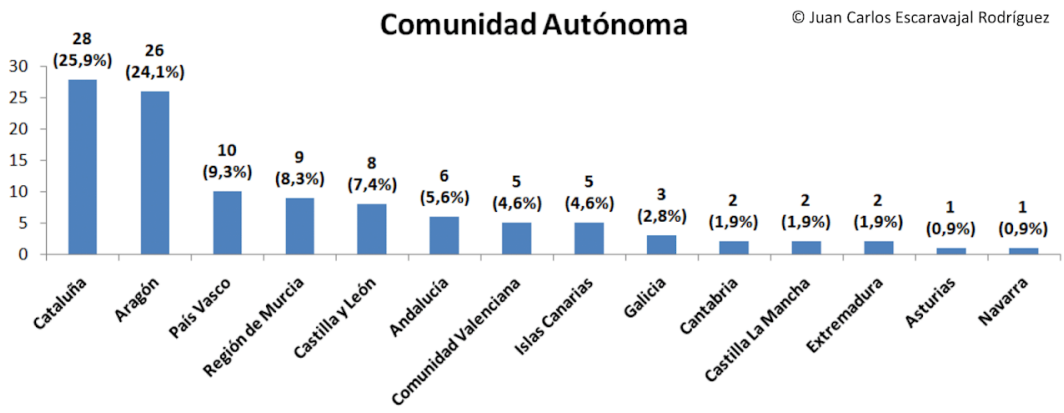

Figura 2.4: Resumen de circuitos por comunidad [5].

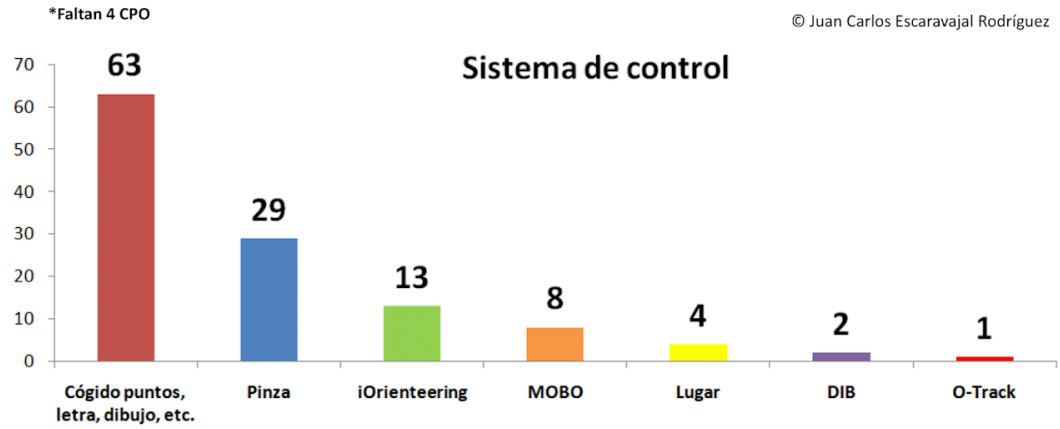

Figura 2.5: Resumen de sistemas utilizados [5].

2.2. Explicación del proyecto

De forma resumida, los usuarios del sistema podrán crear carreras para que otros usuarios puedan participar en ellas. Los usuarios 'organizadores' utilizan la página web para crear y gestionar sus carreras, mientras que los usuarios 'participantes' utilizan la aplicación móvil para ver el mapa y registrar los controles.

Se establecen **dos tipos de carrera**: **evento** y **circuito**. Los eventos son aquellas carreras que se planean organizar en una fecha concreta, como por ejemplo una carrera popular. Por el otro lado, los circuitos son aquellas carreras que se pretende que su disponibilidad sea permanente, es decir, que los usuarios puedan participar en ellos generalmente durante todo el año, aunque pueda haber excepciones debido a horarios, temporadas, etcétera.

Las carreras pueden organizarse según **dos modalidades: recorridos con trazados o en *score***. Las carreras con recorridos trazados pueden tener hasta 10 recorridos con 30 controles cada uno (incluyendo salida y meta). Dichos recorridos tienen un trazado, que indica el orden por el que el participante debe pasar y registrar los controles de la carrera. Por el contrario, las carreras score no tienen recorridos, sino que el participante debe elegir el orden en que pasará y registrará los controles.

Los controles que el organizador debe disponer en el terreno queda a su libre elección, pero deben incluir los **códigos QR** generados para dicha carrera.

Para participar en una carrera, los participantes escanean el código QR de la salida del recorrido o del score, dependiendo del tipo de la carrera. Si el organizador ha dispuesto un mapa en el sistema, los corredores podrán visualizar el mapa con el trazado o controles. Es entonces cuando los participantes escanean los códigos QR de los controles y por último la meta para que el tiempo deje de correr.

De forma posterior, un usuario puede ver sus **resultados de un recorrido** de una carrera, junto a los de los otros participantes.

Capítulo 3

Planificación

La planificación del proyecto vendrá determinada por una metodología ágil SCRUM [7] [8] adaptado a un equipo de una sola persona que cumplirá varios roles.

Los requisitos de usuario se recogen en forma de historias de usuario, las cuales se reparten durante el transcurso del proyecto en intervalos de tiempo preestablecidos denominados *sprints*.

El comienzo del proyecto se ha fijado para el 4 de mayo de 2020. Se establecen 8 iteraciones de 2 semanas de duración cada una, resultando en un tiempo total de 16 semanas. El trabajo de fin de grado consta de 12 créditos ECTS, y cada crédito está estimado en una dedicación del alumno de entre 25 y 30 horas, por lo que se debería emplear un tiempo entre 300 y 360 horas para desarrollar el proyecto. Para cumplir con las horas mínimas, la duración del sprint debe ser de al menos 37.5 horas. En el caso en que no se lleguen a cumplir las horas mínimas, se optará por la implementación de más funcionalidades o mejoras adicionales de las aplicaciones. Por el contrario, en el caso en que se espere un exceso, se dejarán sin implementar las funcionalidades y detalles menos importantes.

La fecha prevista de finalización del proyecto será entonces el 4 de septiembre de 2020, teniendo en cuenta periodos vacacionales durante su desarrollo.

Sprint	Fechas de inicio / fin
1	04/05/2020 - 15/05/2020
2	18/05/2020 - 29/05/2020
3	01/06/2020 - 12/06/2020
4	15/06/2020 - 26/06/2020
5	29/06/2020 - 10/07/2020
6	13/07/2020 - 24/07/2020
7	10/07/2020 - 21/08/2020
8	24/08/2020 - 04/09/2020

Tabla 3.1: Fechas de inicio y finalización previstas para cada sprint

3.1. Historias de usuario

Scrum determina que los requisitos del proyecto estén determinados en forma de historias de usuario, las cuales reflejan las necesidades del cliente desde su perspectiva [9]. A continuación se recogen todas las historias de usuario extraídas:

Historia de usuario
Como usuario quiero registrarme para utilizar la aplicación.
Como usuario quiero iniciar sesión para utilizar la aplicación.
Como usuario quiero cerrar sesión para dejar de usar la aplicación.
Como usuario quiero borrar mi cuenta para dejar de usar la aplicación.
Como usuario quiero poder recuperar mi cuenta si se me olvida la contraseña.
Como usuario quiero crear una carrera.
Como usuario que crea una carrera quiero poder trazar los recorridos en la propia web.
Como usuario que crea una carrera quiero poder importar los recorridos y controles desde un documento externo de las herramientas PurplePen y OCAD.
Como usuario quiero editar una carrera que haya creado.
Como usuario quiero explorar las carreras existentes.
Como usuario quiero ver los detalles de una carrera.
Como usuario quiero ver los resultados de una carrera.
Como usuario quiero participar en una carrera.
Como administrador quiero gestionar los usuarios.
Como administrador quiero gestionar las carreras.
Como administrador quiero gestionar los resultados.
Como administrador quiero añadir contenido en la página principal.

Tabla 3.2: Historias de usuario

Debido a la envergadura del proyecto no se implementará la parte de administración en el ámbito del trabajo de fin de grado, quedando pendiente para el desarrollo posterior del proyecto. No se implementará tampoco la importación de recorridos y controles desde documentos de herramientas externas, también pendiente como funcionalidad futura.

3.2. Product backlog

En el product backlog se recogen las expectativas del proyecto, los requisitos de usuario que se pretenden implementar y a los que se les asigna una prioridad (normalmente respecto al retorno de la inversión o ROI). Representa el qué va a ser construido en su totalidad y puede evolucionar durante el desarrollo. Adicionalmente se les asigna una estimación del esfuerzo que supondrá su implementación en una escala de 0 a 3, tomando el valor 0 los requisitos que menos esfuerzo requieran y 3 los que más.

Las historias de usuario a implementar se añaden a un proyecto en GitHub para gestionar su estado y añadir más si es necesario.

#	Descripción	Prioridad	Esfuerzo
US-001	Como usuario quiero registrarme para utilizar la aplicación.	1	2
US-002	Como usuario quiero iniciar sesión para utilizar la aplicación.	2	2
US-003	Como usuario quiero crear una carrera.	3	3
US-004	Como usuario que crea una carrera quiero poder trazar los recorridos en la propia web.	4	3
US-005	Como usuario quiero ver los detalles de una carrera.	5	1
US-006	Como usuario quiero participar en una carrera.	6	3
US-007	Como usuario quiero ver los resultados de una carrera.	7	2
US-008	Como usuario quiero explorar las carreras existentes.	8	1
US-009	Como usuario quiero editar una carrera que haya creado.	9	2
US-010	Como usuario quiero cerrar sesión para dejar de usar la aplicación.	10	1
US-011	Como usuario quiero borrar mi cuenta para dejar de usar la aplicación.	11	1
US-012	Como usuario quiero poder recuperar mi cuenta si se me olvida la contraseña.	12	1

Tabla 3.3: Product backlog

3.3. Gestión de riesgos

Los proyectos software pueden verse afectados por una serie de riesgos [10] [11], categorizados en:

- **Riesgos del proyecto**: afectan a la planificación temporal y al coste del proyecto. Se identifican problemas potenciales de presupuesto, calendario, personal, recursos, etc.
- **Riesgos técnicos**: amenazan la calidad y la planificación temporal del software que hay que producir. Se identifican posibles problemas de diseño, implementación, interfaz, verificación y mantenimiento.
- **Riesgos del negocio**: amenazan la viabilidad del software. Pueden producir la cancelación del proyecto.

Como el equipo del proyecto está compuesto por tan solo 1 desarrollador, los riesgos identificados pueden tener un impacto bastante notable, los cuales son:

Riesgo 1	**Retrasos en la planificación**
Tipo	Riesgo de proyecto
Probabilidad	Alta
Impacto	Medio
Descripción	Se requieren más horas de trabajo debido a una mala planificación del proyecto.
Plan de mitigación	Adelantar el inicio del proyecto de forma que la fecha prevista de finalización tenga un colchón de tiempo amplio hasta la fecha límite.
Plan de contingencia	Añadir un sprint más si es necesario.

Tabla 3.4: Riesgo de retrasos en la planificación

Riesgo 2	**Enfermedad de miembro del equipo**
Tipo	Riesgo de proyecto
Probabilidad	Media
Impacto	Alto
Descripción	Un miembro del equipo contrae una enfermedad que provoca su indisposición para trabajar.
Plan de mitigación	Adelantar el inicio del proyecto de forma que la fecha prevista de finalización tenga un colchón de tiempo amplio hasta la fecha límite.
Plan de contingencia	Evaluar el tiempo perdido, aumentar las horas de trabajo diarias y posponer las tareas que no han podido completarse al siguiente sprint.

Tabla 3.5: Riesgo de enfermedad de miembro del equipo

Riesgo 3	**Falta de formación y experiencia del equipo**
Tipo	Riesgo técnico
Probabilidad	Alta
Impacto	Medio
Descripción	El equipo no tiene los conocimientos previos necesarios para el correcto desarrollo del proyecto.
Plan de mitigación	Dedicar parte de las horas a la formación del equipo.
Plan de contingencia	Recurrir a personas con más experiencia y foros con preguntas/respuestas ya solucionadas.

Tabla 3.6: Riesgo de falta de formación y experiencia del equipo

Riesgo 4	**Indisponibilidad del entorno de trabajo**
Tipo	Riesgo técnico
Probabilidad	Baja
Impacto	Alto
Descripción	El entorno de trabajo habitual del equipo de desarrollo sufre algún daño o similar que causa la detención del proyecto.
Plan de mitigación	Utilizar un entorno fiable, realizar comprobaciones periódicas y mantener actualizado el repositorio del proyecto de forma frecuente.
Plan de contingencia	Reparar o sustituir las partes dañadas y añadir un sprint más si es necesario.

Tabla 3.7: Riesgo de indisponibilidad del entorno de trabajo

Riesgo 5	**Cambios en los requisitos**
Tipo	Riesgo técnico
Probabilidad	Baja
Impacto	Medio
Descripción	Se producen modificaciones a considerar en los requisitos durante el transcurso del proyecto.
Plan de mitigación	Evaluar varias veces las historias de usuario buscando situaciones problemáticas.
Plan de contingencia	Evaluar los cambios y su factibilidad. Añadir un sprint adicional si es necesario.

Tabla 3.8: Riesgo de cambios en los requisitos

Riesgo 6	**Cambios en las tecnologías utilizadas**
Tipo	Riesgo técnico
Probabilidad	Media
Impacto	Alto
Descripción	Durante el transcurso del proyecto se detecta una situación que impide la utilización de una tecnología concreta.
Plan de mitigación	Dedicar tiempo a investigar las tecnologías utilizadas en proyectos similares y detectar incompatibilidades entre ellas.
Plan de contingencia	Buscar una tecnología sustituta que permita conseguir los requisitos. Añadir un sprint adicional si es necesario.

Tabla 3.9: Riesgo de cambios en las tecnologías utilizadas

3.4. Análisis de costes y presupuesto

Como durante el proyecto el miembro del equipo realizará labores tanto de front-end como de back-end, se ha utilizado como salario base de referencia el un desarrollador Full Stack.

Según el sitio web GlassDoor [12] su valor ronda una media de 30 723€al mes. Basándose en un horario de jornada completa, supone un salario de 12,489€/h. Para un trabajo de 330 horas se estima un coste total de personal de **4121,38€**.

El desarrollo del proyecto se realizará en un ordenador valorado en 1000€. Según la tabla de coeficientes de amortización lineal [13], los equipos informáticos tienen un coeficiente lineal máximo del 25 % en un periodo máximo de 8 años. La amortización resultante durante 4 meses sería de 1000€* 0.25 * 4/12 años de trabajo = **83,3€**.

Para minimizar el impacto de los posibles riesgos que puedan producirse, se añade un coste del 20 % del presupuesto base, resultando en un **presupuesto total de 5045,62€** (ver Tabla 3.10).

Concepto	**Coste**
300 horas laborales	4121,38€
Amortización	83,3€
Total base	4204,68€
Total	**5045,62€**

Tabla 3.10: Presupuesto estimado

Capítulo 4

Herramientas y Tecnologías

A continuación se describen las herramientas utilizadas durante el transcurso del proyecto.

4.1. Herramientas de planificación y gestión

GitHub

GitHub [14] es una forja para alojar proyectos utilizando el sistema de control de versiones Git, el cual realiza un seguimiento de los cambios en el código fuente durante el desarrollo de software. Es utilizado principalmente por programadores, aunque puede usarse para controlar cambios en cualquier tipo de archivos.

Se hace uso de la cuenta educativa para el desarrollo en privado del proyecto.

- Gratuito.
- Tiene un cliente GUI para una gestión de Git más sencilla.
- Permite la creación de proyectos.
- Facilita el despliegue en el entorno de Heroku.

WakaTime

WakaTime [15] es una herramienta de *"time tracking"* que permite controlar el tiempo empleado en el proyecto. Se utilizará para calcular el tiempo empleado en los sprints. En su versión gratuita solo guarda los datos de hasta una semana de antigüedad pero envía al correo cada semana un resumen suficiente para el control necesario.

- Permite discernir entre proyectos y lenguajes utilizados.
- Puede ser utilizada en gran cantidad de IDEs.
- Envía un reporte semanal al correo.

4.2. Herramientas de diseño y desarrollo

Astah

Astah [16] es una herramienta de modelado UML creada por la compañía japonesa Change Vision. Permite la creación de diagramas de clases, casos de uso y secuencia, entre otros.

Se elige esta herramienta debido al uso previo en múltiples asignaturas de la carrera. En el proyecto se ha utilizado la versión gratuita para estudiantes *Astah UML*.

NetBeans

NetBeans [17] es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java pero que puede extenderse gracias al uso de módulos. Es un producto libre y gratuito sin restricciones de uso.

Se elige esta herramienta debido a la experiencia con él durante la carrera, con las siguientes ventajas:

- Utiliza un editor de texto que resalta la sintaxis.
- Realiza una compilación en tiempo real que permite detectar fallos de programación.
- Puede extenderse gracias al uso de módulos.

Balsamiq Wireframes

Balsamiq Wireframes [18] es una herramienta de esquematizado que reproduce la experiencia de bocetado en un cuaderno o pizarra, pero usando un ordenador.

- Agiliza la creación de bocetos gracias a la paleta de componentes prediseñados.
- Permite centrarse en la estructura y el contenido.
- Permite exportar los diseños en diferentes formatos.

Overleaf

Overleaf [19] es un editor colaborativo de LaTeX basado en la nube que se utiliza para escribir, editar y publicar documentos técnicos. Se ha utilizado para redactar la documentación que estás leyendo ahora mismo.

- Permite la edición simultánea de varios usuarios.
- Permite el uso de plantillas.
- Exportación del documento en PDF.
- Sincronización con Dropbox, Git y GitHub.
- Puede ser desplegado en un entorno local.

4.3. Backend - API

Maven

Maven [20] es una herramienta open source que facilita la gestión y construcción de proyectos Java basada en XML. En concreto se utilizará para declarar las dependencias del proyecto de Spring Boot en el archivo *"pom.xml"*.

- Permite la reutilización de software
- Es configurable y extensible
- Integración con múltiples IDEs (incluyendo NetBeans).

Spring

Spring [21] es un framework open source que facilita el desarrollo y despliegue de aplicaciones Java. Se centra en el orquestado de aplicaciones empresariales de forma que los desarrolladores puedan centrarse en el desarrollo de la lógica de negocio a nivel de aplicación, abstrayendo de los detalles específicos de los entornos de despliegue.

- Modular: se puede elegir qué componentes se desean reutilizar en un proyecto.
- Acceso e integración sencilla de los datos gracias a múltiples módulos que abstraen de los detalles específicos de distintas tecnologías como JDBC y ORM.
- Pruebas unitarias y de integración

Hibernate

Hibernate [22] es una herramienta libre de mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Para este proyecto se utilizarán anotaciones las entidades que permitan establecer las relaciones entre ellas. Permite abstraer al desarrollador de las implementaciones JPA.

PostgreSQL

PostgreSQL [23], también llamado Postgres, es un sistema de gestión de bases de datos relacional multiplataforma orientado a objetos y de código abierto. Se elige debido a su compatibilidad directa con Heroku y a su experiencia anterior durante la carrera universitaria.

pgAdmin

pgAdmin [24] es un entorno gráfico de código abierto que permite conectarse a bases de datos PostgreSQL y que facilita su gestión y administración.

4.4. Cliente web

Angular

Angular [25] es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

Bootstrap

Bootstrap [26] es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.

Node.js

Node.js [27] es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

npm

npm [28] es el sistema de gestión de paquetes por defecto para Node.js. Se utiliza para gestionar las dependencias del proyecto Angular y para añadir nuevas funcionalidades de otras librerías.

4.5. Cliente Android

Gradle

Gradle [30] es una herramienta de compilación enfocada en la automatización y soporte de un desarrollo en múltiples lenguajes, ofreciendo un modelo flexible durante todo el proceso de desarrollo, desde la compilación hasta la publicación.

Retrofit

Retrofit [36] es un cliente HTTP de tipado seguro utilizado para consumir servicios web REST. Tiene como dependencia OkHttp [35] (de los mismos desarrolladores).

Room

Room [32] es una biblioteca de persistencias que brinda una capa de abstracción para SQLite que permite acceder a la base de datos sin problemas aprovechando toda la potencia de SQLite.

4.6. Despliegue

Heroku

Heroku [33] es una plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación. Permite abstraer de los detalles de la infraestructura que ejecutará el proyecto software del desarrollador.

- Gratuito en su versión básica.
- Extensible: pueden usarse add-ons para aumentar las funcionalidades de la infraestructura. Por lo general, los add-ons son también gratuitos en su versión básica.
- Escalable: se puede gestionar de forma sencilla la cantidad de recursos utilizados por la infraestructura.
- Despliegue automático desde GitHub.

Capítulo 5

Seguimiento del proyecto

El seguimiento del proyecto se realizará sprint a sprint. Al comienzo de cada sprint se definirá el backlog del sprint que contiene las tareas necesarias de cara a conseguir los requisitos del producto. A cada tarea se le asignará una estimación de tiempo necesario para completarla, y se indicará el estado en el que se encuentra cuando el sprint finalice (pendiente, en progreso o completado). Además se indicará si las tareas están asociadas con alguna historia de usuario en concreto.

5.1. Sprint 1

En el primer sprint se planteará el alcance del proyecto estableciendo los objetivos a conseguir y se investigarán las tecnologías a utilizar. Además el miembro del equipo se formará en el framework Spring Boot e investigará acerca del patrón MVVM, cada vez más popular en Android. Se crearán los proyectos en los entornos de desarrollo preferidos del equipo a modo de prueba.

Se recabarán las historias de usuario estableciendo cuáles se van a implementar y cuales no. Se realizará un bocetado inicial de las interfaces de usuario de los clientes, y se crearán a partir de ellos las pantallas de conexión en los clientes web y Android.

Se estima la duración del sprint en **40 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Investigación del alcance del proyecto.	4	Completado
	Recabar historias de usuario.	1	Completado
	Bocetar la interfaz gráfica web.	3	Completado
	Bocetar la interfaz gráfica móvil.	3	Completado
	Planificar del desarrollo del proyecto, análisis de costes y riesgos.	4	En progreso
	Arquitectura y modelo de dominio.	2	Completado
	Configurar el entorno de desarrollo web.	2	Completado
	Configurar el entorno de desarrollo Android.	2	Completado
	Formarse en el patrón MVVM de Android.	6	En progreso
	Formarse en el framework Spring Boot y configurar su entorno de desarrollo.	6	Completado
US-001 US-002	Crear pantallas de inicio, login y registro sin funcionalidad.	6	Completado

Tabla 5.1: Backlog del sprint 1

5.1.1. Revision de sprint 1

Para el sprint 1 se han empleado un **tiempo total de 44 horas**, 4 horas más de lo previsto.

Las tecnologías escogidas para la API (Spring Boot + Hibernate) tienen numerosos ejemplos de implementación y recursos disponibles en la red [34], por lo que se espera que el desarrollo se vea facilitado y la resolución de fallos sea más sencilla. El patrón MVVM es interesante para aprenderlo y utilizarlo en la aplicación móvil, sobre el cual se encontraron muchos ejemplos pero el más útil y similar al proyecto es el de un ejemplo del desarrollador Mitch Tabian [37].

El despliegue del proyecto tiene pensado realizarse en Heroku, debido a las ventajas disponibles de forma gratuita. Debido a esta decisión, se añade la restricción de usar PostgreSQL como motor de la base de datos. Como ya se ha utilizado previamente durante la carrera, no se espera mucha complicación en este tema.

Para el siguiente sprint quedan pendientes los análisis de costes y riesgos, cuya estimación de horas no fue suficiente.

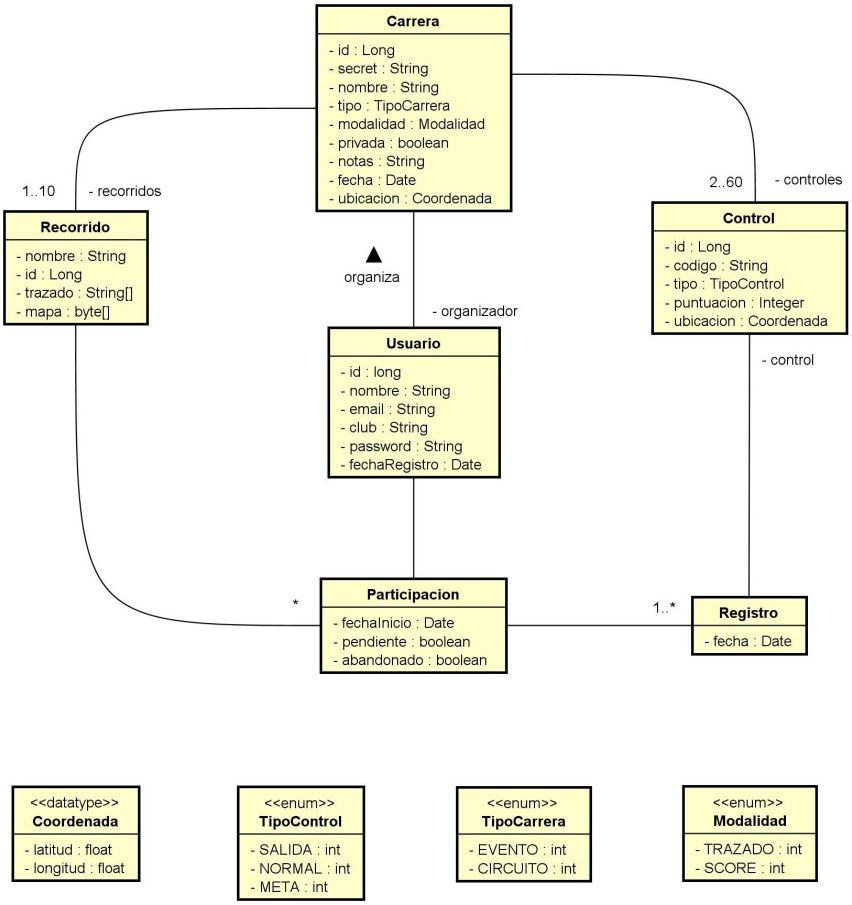

Figura 5.1: Diagrama de clases del dominio

5.2. Sprint 2

Para el segundo sprint se finalizarán los análisis de costes y riesgos que quedaron pendientes en el sprint anterior.

Se realizará el diseño lógico de la base de datos y se implementará el servicio de autenticación en la API y en los clientes. Se creará un controlador de prueba para verificar la correcta autenticación de los usuarios.

Se estima la duración del sprint en **38 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Análisis de costes y riesgos.	2	Completado
	Diseñar el modelo lógico de la base de datos.	3	Completado
US-001 US-002	Configurar base de datos y crear tabla de usuarios.	3	Completado
	Crear servicio y repositorio de usuarios en la API.	3	Completado
	Crear controladores en la API de autenticación, de usuarios y de prueba.	4	Completado
US-003	Crear tablas de carreras y recorridos en la BD.	1	Completado
	Crear servicio y repositorio de carreras en la API.	5	Completado
	Crear controlador de carreras en la API.	5	Completado
	Crear servicio y repositorio de recorridos en la API.	5	Completado
	Crear controlador de recorridos en la API.	5	Completado
	Crear formulario de creación de carreras en la web.	1	Completado

Tabla 5.2: Backlog del sprint 2

5.2.1. Revision de sprint 2

Para el sprint 2 se han empleado un **tiempo total de 41 horas**, 3 horas más de lo previsto.

Surgieron algunos problemas de autenticación externa de la base de datos desplegada en una máquina virtual y su acceso desde el proyecto Spring Boot.

Se ha elegido el estándar JWT para la autenticación de los usuarios en ambos clientes debido a ciertas ventajas que incorpora, como por ejemplo que el servidor no necesita almacenar las sesiones de los usuarios, sino que simplemente se puede validar el token. En el cliente web mediante JWT se hace uso de los HTTP Interceptors en Angular para enviar en cada petición el token almacenado en el almacenamiento local del navegador, donde no puede ser accedido mediante ataques típicos como XSS y CSRF. En el caso del cliente Android, se hace uso de OkHttp [35] junto con Retrofit [36] para manejar las peticiones a la API.

Los controles que forman parte del trazado ordenado de un recorrido se han persistido en una tabla que referencia al recorrido y contiene el código del control y su orden. Se ha intentado referenciar el control por medio de su ID en vez de su código, pero se han producido problemas a la hora de configurar la entidad *Recorrido* con Hibernate, por lo que de momento se deja como una lista ordenada de cadenas de texto.

No se ha llegado a conectar la interfaz gráfica de la web con el endpoint de la API para la creación de carreras debido a la posible modificación de la interfaz. Se usa la extensión de navegador *RESTClient* para depurar las peticiones que llegan al controlador de carreras en el servidor.

5.3. Sprint 3

En el tercer sprint se dedicarán algunas horas a la documentación del diseño actual.

Se abordará la historia *US-004*, consistente en la creación de un trazador web. En dicho trazador se podrá cargar una imagen a elección del creador y realizar un trazado sobre ella. También se podrá aplicar a las carreras score. Se estima que la carga de trabajo de esta funcionalidad será exigente si se quiere lograr un buen resultado.

Para mantener los datos entre los componentes, y además para que el usuario evite perder los datos abandonando sin querer, se creará un sistema de borrador en el que se persistirá en el almacenamiento local del navegador los datos de la carrera que se está creando.

Se estima la duración del sprint en **42 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Documentación y revisión de código.	3	Completado
US-003 US-004	Crear sistema de borrador de carrera.	3	Completado
US-004	Crear interfaz gráfica base.	4	Completado
	Implementar carga de imagen externa.	2	Completado
	Dibujar los distintos tipos de control.	5	Completado
	Crear sistema de desplazamiento y zoom.	5	En progreso
	Unir tramos entre controles y dibujar números de orden.	5	En progreso
	Sincronizar datos entre el componente trazador y de creación de recorridos.	5	En progreso
	Permitir borrado y reubicación de controles.	5	Completado
	Permitir edición de nombre y borrado de recorrido.	2	Completado
	Diferenciar entre creación de recorridos y de controles.	2	En progreso

Tabla 5.3: Backlog del sprint 3

5.3.1. Revision de sprint 3

En general las estimaciones de tiempo han sido muy incorrectas, requiriendo mucho más tiempo del esperado. En total se han empleado **49 horas** y no se han completado todas las tareas.

Se ha conseguido el desplazamiento sobre el mapa, pero las implementaciones abordadas para el zoom no son buenas.

Se ha conseguido unir los controles del trazado con líneas, pero ha de mejorarse para que las líneas no aparezcan dentro del control y que se muestre el orden de los controle. Además, el triángulo de salida debe tener un vértice orientado hacia el primer control (ver Figura 5.2).

Por último, queda pendiente que el componente soporte la creación no solo de recorridos, sino de controles para las carreras score. La disposición de la vista varía dependiendo de este factor y además de si el usuario ha elegido utilizar el trazador o no.

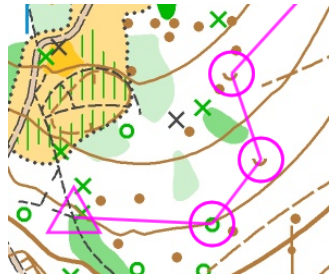

Figura 5.2: Apariencia incorrecta de los tramos

5.4. Sprint 4

Además de las tareas gue quedaron pendientes en el sprint anterior, se finalizará el proceso de creación de la carrera. Una vez conseguido, se abordará la historia de usuario *US-005* que mostrará los detalles de una carrera, incluyendo la generación de los códigos QR de los controles.

Se estima la duración del sprint en **42 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Documentación y revisión de código.	3	Completado
US-004	Unir tramos entre controles y dibujar números de orden.	3	Completado
	Sincronizar datos entre el componente trazador y de creación de recorridos.	2	Completado
	Reestructurar el sistema de canvas.	5	Completado
	Diferenciar entre creación de recorridos y de controles.	2	Completado
	Generar mapas a partir del trazador.	2	Completado
US-003	Permitir adición/eliminación de mapa de un recorrido.	2	Completado
	Permitir asignación de puntuación de control en carreras score.	3	Completado
	Permitir subida/borrado de mapas de recorridos.	2	Completado.
	Crear tabla de controles y de registros.	1	Completado
	Crear servicio y repositorio de controles.	3	Completado
	Crear servicio y repositorio de registros.	3	Completado
US-005	Crear vista de carrera.	3	Completado
	Generar controles QR de la carrera.	6	Completado
	Exportación de QR en PDF	3	Completado

Tabla 5.4: Backlog del sprint 4

5.4.1. Revisión de sprint 4

En total se han empleado **44 horas**.

Se ha corregido la apariencia de los tramos y se han añadido el orden de los controles (ver Figura 5.3). El número se muestra a una distancia predeterminada en la bisectriz del ángulo cóncavo formado por las líneas de los tramos anterior y siguiente. De esta forma se mejora la visibilidad del mapa.

Se ha reestructurado el sistema de los canvas para el desplazamiento y zoom. La estructura actual es: un canvas para el mapa base, otro canvas para el trazado y otro canvas para dibujar el marcador del ratón. Antes el tamaño de los canvas era el mismo que el del elemento padre, pero ahora tienen el tamaño *real"* y tienen asociada una posición absoluta con respecto al elemento padre. El desplazamiento se controla variando los valores de los offset superior e izquierdo. El zoom se logra alterando el tamaño real del canvas, haciéndolo más grande para

Figura 5.3: Apariencia correcta de los tramos

aumentar el zoom y viceversa. Al cambiar el zoom se actualizan las métricas de los dibujos para alterar su aspecto, siempre teniendo en cuenta la norma ISOM 2017 [38] que recoge las dimensiones de los elementos del trazado en un mapa de orientación.

Una vez conseguida la creación de la carrera, se crea un nuevo componente para su visualización (tras crear la carrera, y si ha tenido éxito, se redirige al usuario a esta vista). En ella se muestran los datos de la carrera, así como la lista de recorridos. Próximamente, tras seleccionar uno de ellos se mostrarán sus resultados.

También se crea un nuevo componente para generar los controles QR de la carrera. Se crea en la API un endpoint nuevo que responde con un objeto JSON con los códigos de control y el contenido que se codificará en el QR asociado a ellos. En el componente web se renderizan haciendo uso de la librería *EasyQRCodeJS* [39]. También se da la opción de generar un PDF con los controles haciendo uso de la librería PDFMake [40].

5.5. Sprint 5

Tras conseguir la creación de una carrera, el siguiente paso es permitir a un usuario participar en ella (*US-006*). Para ello se creará una actividad en la aplicación Android, en la que primero se escaneará un código QR de salida de un recorrido o de la carrera score. Tras confirmar el usuario que quiere iniciar dicho recorrido, y si puede hacerlo, el servidor registrará la participación del usuario en el recorrido y la aplicación le mostrará el mapa (si existe) y le indicará el siguiente control que debe validar, en una pantalla de escaneo de código QR.

Una vez lograda la participación del usuario, al finalizar su recorrido le será mostrado los resultados del recorrido (*US-007*). También se implementará la vista de resultados en la web.

La estimación de tiempo para el sprint 5 es de **40 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Documentación y revisión de código.	3	Completado
US-006	Crear actividad de escaneo y comprobar funcionamiento de lector de QR.	10	Completado
	Crear diálogo de éxito/error global.	4	Completado
	Permitir inicio de recorrido.	2	Completado
	Mostrar mapa del recorrido.	4	Completado
	Permitir registro de control.	4	Completado
	Realizar comprobación de recorrido pendiente.	2	Completado
	Dar opción de abandonar recorrido.	2	Completado
US-007	Crear componente web de resultados de recorrido.	4	Pendiente
	Crear actividad de resultados de recorrido.	4	En progreso

Tabla 5.5: Backlog del sprint 5

5.5.1. Revisión de sprint 5

En total se han empleado **42 horas**.

Tras varias pruebas e investigaciones, se ha implementado el lector de QR usando la API ***Mobile Vision*** de Android [41]. Se ha comprobado la correcta detección del QR y se han creado el ViewModel de la actividad y el repositorio destinado al control de estos registros.

La visualización del mapa del recorrido ha resultado ser más complicada de lo que se esperaba, debido a limitaciones del sistema operativo. El problema principal es que las imágenes de los mapas suelen tener una resolución muy alta, lo cual en Android genera problemas de memoria que impiden la implementación de lo deseado de forma directa. Tras investigar el problema, se encuentra la solución haciendo uso de la librería ***Subsampling Scale Image View*** [42], la cual permite la carga de imágenes muy grandes y de algunos eventos deseados.

Se ha implementado también que cuando el usuario accede a la actividad de escaneo, se realiza una comprobación en la API de si el usuario tiene algún recorrido pendiente. En dicho caso, la aplicación notifica al usuario de este suceso y le da a elegir entre reanudar el recorrido o abandonarlo. Si decide reanudarlo proseguirá la carrera en el punto donde la había dejado.

Debido al exceso de horas acumulado en las tareas anteriores, no se ha empezado la implementación de la visualización de los resultados en la aplicación móvil, aunque sí se ha hecho en el cliente web, aún sin completar.

5.6. Sprint 6

En este sprint se finalizará la visualización de resultados en la web y en la aplicación móvil.

Se comenzará con la implementación de la historia de usuario *US-008* consistente en la exploración de las carreras existentes en el sistema, además de visualizar un mapa en el que se muestren los circuitos permanentes existentes. El usuario también podrá ver la lista de carreras que ha creado y en las que ha participado.

Por último, en el proyecto Spring Boot se creará un test que borrará los datos de la base de datos e introducirá datos aleatorios de usuario, carreras y resultados (para más información ver sección 7.1).

Se estima que la duración del sprint 6 será de **39 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Documentación y revisión de código.	3	Completado
US-007	Crear vista de resultados de recorrido (app)	3	Completado
	Crear vista de resultados score (app).	3	Completado
	Crear vista de resultados de recorrido (web).	3	Completado
	Crear vista de resultados score (web).	3	Completado
US-008	Crear interfaz gráfica web del explorador.	3	Completado
	Crear endpoint en la API para la búsqueda de carreras.	6	Completado
	Generar lista de resultados de la búsqueda de la web.	3	Completado
	Crear mapa de circuitos permanentes.	5	Completado
	Generar datos de prueba aleatorios de usuarios, carreras y resultados.	6	Completado

Tabla 5.6: Backlog del sprint 6

5.6.1. Revisión de sprint 6

En total se han empleado **43 horas** en este sprint.

Se ha implementado una lista de resultados con una paginación de scroll infinito, es decir,

que se cargarán más resultados a medida que el usuario haga scroll. Se ha utilizado la librería **ngx-infinite-scroll** [43] para este fin.

Se ha hecho uso de la librería **Leaflet** [44] para la creación de un mapa con los circuitos permanentes en el sistema. El usuario puede visualizar dónde están ubicados geográficamente y puede acceder a ver más detalles. También se ha añadido de paso en el formulario de creación de carrera para poder seleccionar la ubicación y en los detalles de la carrera.

5.7. Sprint 7

Gracias a los datos de prueba, se ha detectado que la utilización de filas en la base de datos es demasiado alta (más de 50000) para ser utilizada en Heroku, que establece un límite de 10000 filas de uso gratuito. Se decide optimizar las tablas con más filas y pocas columnas, pasando a ser campos JSON en las tablas a las que pertenece su entidad:

- Los registros de controles pasan a ser participaciones, que asocian a un usuario con un recorrido de forma que los corredores solo puedan participar una única vez.
- Los elementos de los trazados pasan a ser un campo JSON en la tabla de recorridos, en forma de JsonArray de cadenas de texto.
- Como los elementos de la tabla de controles ya no son referenciados desde los registros, se elimina y se incluyen dentro de la tabla carreras en una columna JSON en forma de JsonObject.

De cara a implementar la historia de usuario *US-009* se reutilizará el mismo componente que el de creación. Sin embargo, debido a problemas con la integridad de los datos en cuanto a recorridos y controles se refiere, el organizador solamente podrá modificar datos como el nombre, ubicación, tipo, notas y fecha.

Buscando información sobre JWT para lograr el cierre de sesión de un usuario, tratando de implementar algún ejemplo y atendiendo a las restricciones de autorización de la lógica de negocio, se han detectado inconvenientes en el uso de esta tecnología por la que sus ventajas ya no valen la pena. Por ello, se implementa un sistema de autenticación más sencillo mediante el uso de Spring Security [45].

Se estima que la duración del sprint 7 será de **43 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Documentación y revisión de código.	3	Completado
	Reorganizar y optimizar el diseño lógico de la BD.	4	Completado
	Actualizar los servicios y repositorios de la API por el cambio anterior.	6	Completado
	Actualizar las partes afectadas de la web por el cambio anterior.	6	Completado
	Actualizar las partes afectadas de la app por el cambio anterior.	6	Completado
	Cambiar sistema de autenticación.	5	Completado
US-009	Permitir edición de carrera.	4	Completado
US-010	Permitir cerrado de sesión.	4	Completado
US-011	Permitir borrado de cuenta.	4	Completado

Tabla 5.7: Backlog del sprint 7

5.7.1. Revisión de sprint 7

En total se han empleado **46 horas** en este sprint.

La reorganización debido al cambio en la base de datos hizo que gran parte de la lógica en la API se simplificase y optimizase (ver subsección 6.2.6).

El cambio del sistema de autenticación fue menos complicado y tomó menos tiempo de lo previsto, debido a la gran cantidad de recursos online. El cerrado de sesión se ha podido implementar de forma mucho más sencilla que con JWT.

5.8. Sprint 8

En la recta final del proyecto de fin de grado se dedicarán varias horas a la corrección de bugs necesaria, documentación y revisión del código en general.

Se investigará y se configurará el entorno de despliegue en Heroku, incluyendo la base de datos.

Se estima que la duración del último sprint será de **40 horas**.

Historia de usuario	Descripción	Estimación (horas)	Estado
	Planear sprint.	1	Completado
	Documentación y revisión de código	15	En progreso
	Corregir bugs web	6	Completado
	Corregir bugs app	6	En progreso
	Implementación HTTPS	4	Completado
	Despliegue en Heroku.	8	Completado

Tabla 5.8: Backlog del sprint 8

5.8.1. Revisión de sprint 8

En total se han empleado **44 horas** en este sprint.

El despliegue en Heroku fue muy sencillo para la parte del backend. Se modificó el archivo application.properties para que utilizase variables de entorno. Sin embargo, el front-end se complicó bastante debido a problemas con el servidor Express y la redirección de peticiones a la API que provocó problemas con CORS.

Heroku utiliza por defecto HTTPS para las conexiones a su página, por lo que no se necesita configuración adicional en producción. Sin embargo, en el entorno de desarrollo sí que hace falta generar un certificado autofirmado para usar HTTPS. En el archivo **package.json** del proyecto Angular se crea el script **devstart** con el siguiente contenido:

```
ng serve --proxy-config proxy.conf.json --host 0.0.0.0 --ssl true
    --ssl-cert ./ssl/server.crt --ssl-key ./ssl/server.key
```

La clave y el certificado se registran en Git, por lo que están expuestos. De todas formas, solamente tienen propósitos de desarrollo en un entorno privado por lo que no tiene impacto. Por otro lado, para que el cliente de Android no de problemas, se implementa un cliente HTTPS inseguro que carga el certificado como Autoridad Certificadora. Por motivos obvios, este cliente solamente será utilizado en el entorno de desarrollo y será borrado en el de producción.

5.9. Resultados del seguimiento

Se han producido constantes bloqueos debido a la falta de experiencia del equipo de desarrollo, la mayoría relacionadas con la persistencia de los datos. Sin embargo, no ha hecho falta añadir ningún sprint adicional. En general las estimaciones de tiempo han sido incorrectas, debido a la falta de experiencia y al desconocimiento de algunas de las tecnologías utilizadas.

Sprint	Tiempo estimado	Tiempo empleado
1	40h	44h
2	38h	41h
3	42h	49h
4	42h	44h
5	40h	42h
6	39h	43h
7	43h	46h
8	40h	44h
TOTAL	**324h**	**353h**

Tabla 5.9: Resumen de tiempos estimados y empleados en los sprints

5.9.1. Costes reales

En la sección 3.4 se realizó el análisis de coste inicial y se calculó el presupuesto del proyecto (5045,62€). Ahora, a partir de las horas reales empleadas se pueden calcular los costes reales del proyecto:

Concepto	Coste
353 horas laborales	4408,62€
Amortización	83,3€
Total	4491,92€

Tabla 5.10: Costes reales

El coste real se mantiene por debajo del presupuesto estimado gracias al colchón del 20 %, sin el cual se hubiera producido un sobrecoste debido al exceso de horas empleadas.

Capítulo 6

Diseño

Las decisiones de diseño aplicadas en el proyecto son:

- **Arquitectura cliente-servidor con 2 clientes**: uno web y otro móvil. El cliente web contendrá las herramientas para llevar a cabo la gestión de carreras, y el cliente móvil las de participación en recorridos. Desde ambos clientes se podrán gestionar los datos de los usuarios, la búsqueda de carreras y la visualización de resultados.
- **API REST como back-end** utilizando el patrón MVC con los framework **Spring Boot** e **Hibernate**.
- Para persistir los datos utilizados en el sistema se utilizará el motor de base de datos **PostgreSQL**.
- **Front-end** web con framework **Angular** junto con framework **Bootstrap** para estilos, con despliegue en entorno **Node.js**.
- Patrón **MVVM** en la aplicación Android.

6.1. Bocetos de la interfaz gráfica

A continuación se muestran los bocetos de las vistas principales de los clientes que manejarán los usuarios (web y móvil).

6.1.1. Bocetos del cliente web

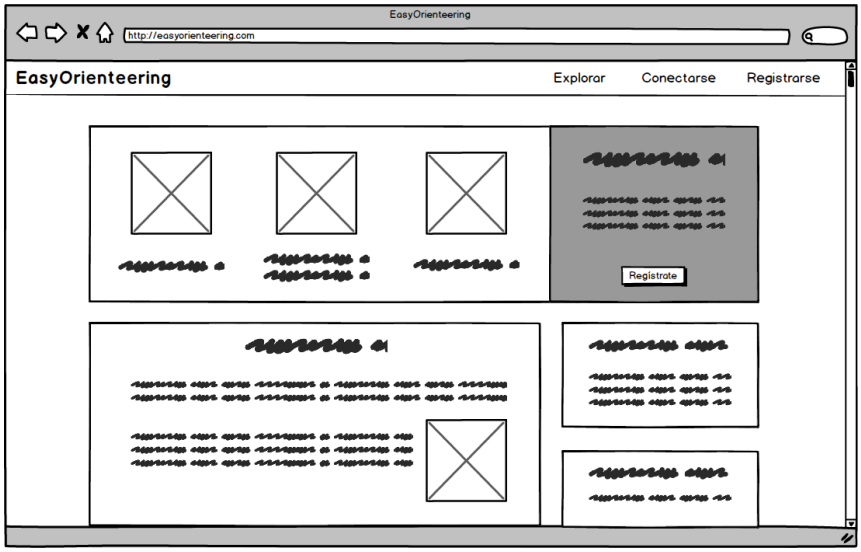

Figura 6.1: Pantalla de inicio

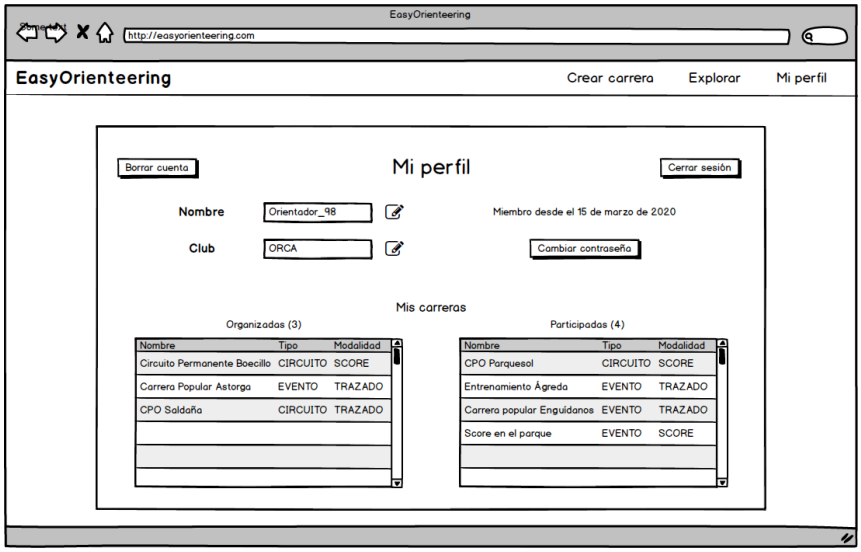

Figura 6.2: Pantalla de perfil

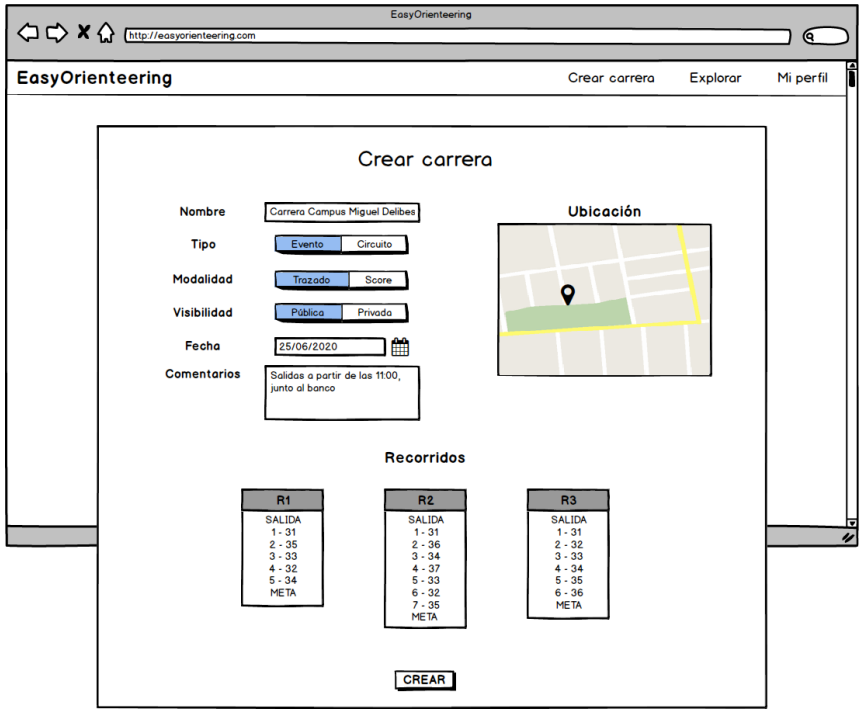

Figura 6.3: Pantalla de creación de carrera

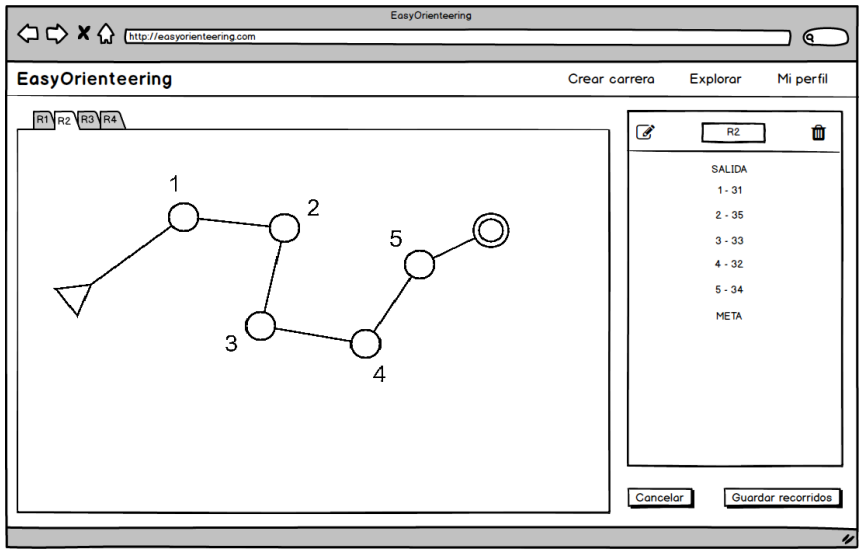

Figura 6.4: Pantalla de trazador web

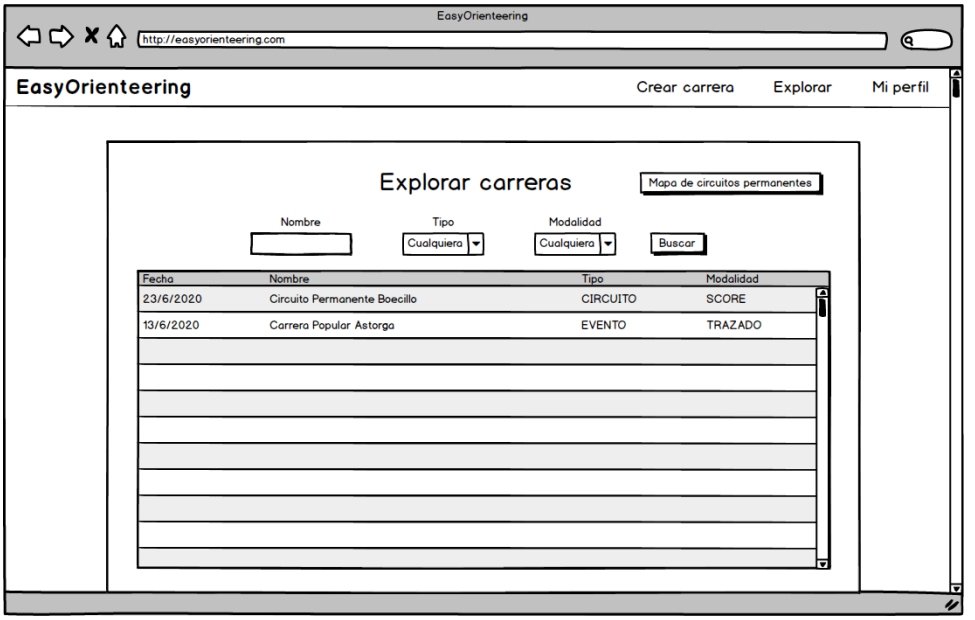

Figura 6.5: Pantalla de exploración de carreras

6.1.2. Bocetos cliente móvil

Figura 6.6: Primer inicio de la aplicación

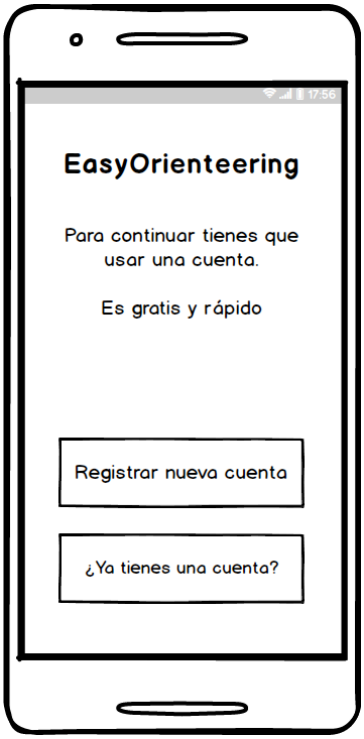

Figura 6.7: Pantalla principal de la aplicación (no conectado)

Figura 6.8: Pantalla de login

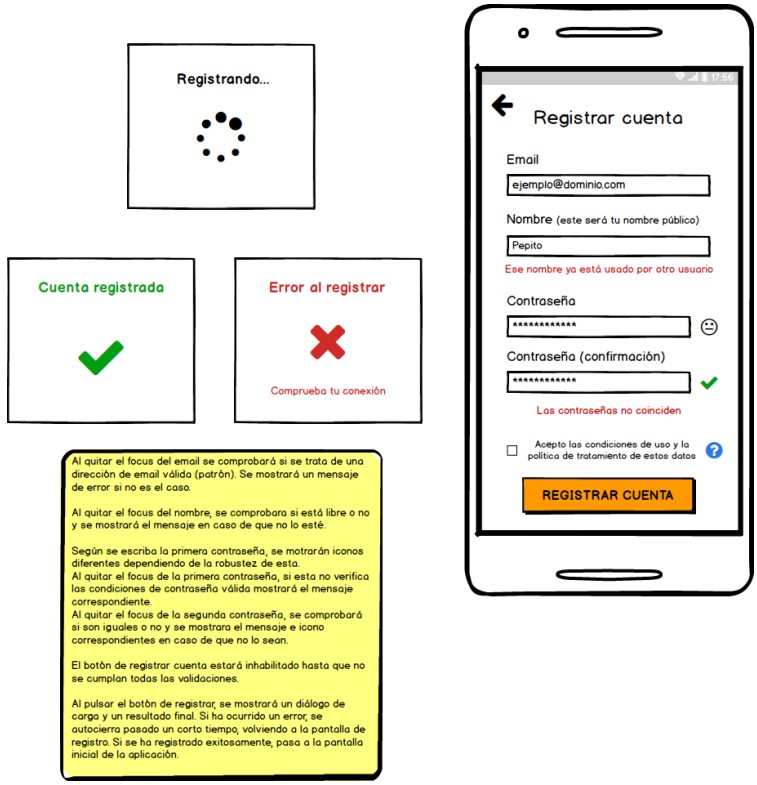

Figura 6.9: Pantalla de registro

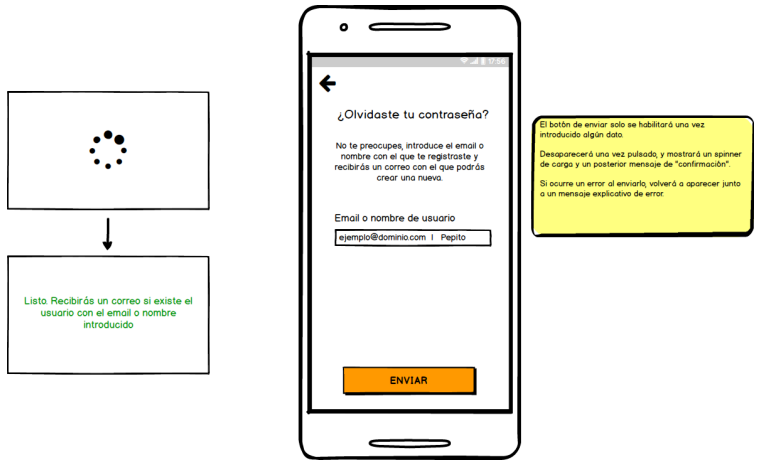

Figura 6.10: Pantalla de restablecimiento de contraseña

Figura 6.11: Pantalla principal de la aplicación (conectado)

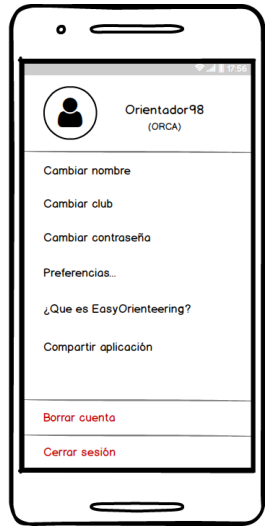

Figura 6.12: Pantalla de perfil

Figura 6.13: Pantalla de visión de carreras participadas y organizadas

Figura 6.14: Pantalla de escaneo de control

Figura 6.15: Pantalla de mapa

6.2. Diseño de la base de datos

A continuación se expone en la figura 6.16 el modelo relacional generado para la base de datos:

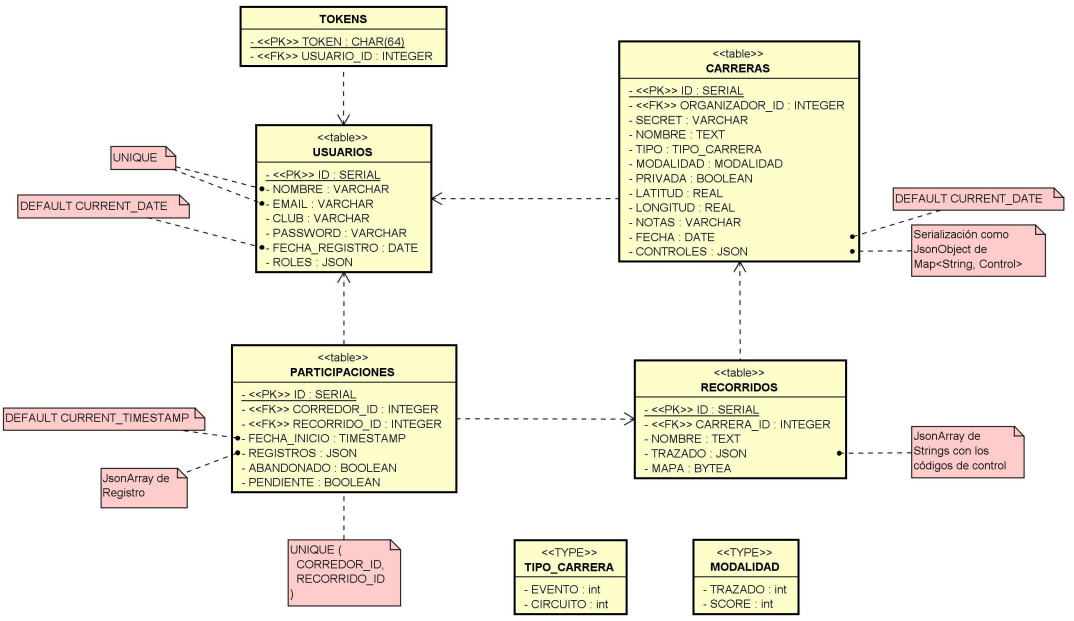

Figura 6.16: Diagrama relacional de la base de datos.

6.2.1. Tabla Usuarios

Los campos de nombre e email serán únicos de forma individual en la tabla, de forma que no haya 2 usuarios distintos con el mismo nombre ni con el mismo email.

Para el proyecto actual no se espera añadir ninguna funcionalidad en el campo del club del usuario excepto la de su actualización. En otras palabras, solo será un detalle de los datos del usuario y no se utilizará para nada más.

Como es necesario almacenar la contraseña de los usuarios se hará de forma de hash, en concreto BCrypt debido a su facilidad de implementación en Spring Boot y sus ventajas de seguridad.

6.2.2. Tabla Carreras

Se utiliza la clave foránea *ORGANIZADOR_ID* para referenciar el organizador de la carrera.

La columna **secret** almacena una cadena de caracteres generada de forma aleatoria en el servidor en el momento de la carrera. Dicha cadena es la que se utiliza para generar el secret de los controles QR, factor adicional de seguridad para evitar trampas.

El tipo de carrera y su modalidad se definen como *enums* de forma previa a la creación de las tablas, de la siguiente forma:

```
CREATE TYPE TIPO_CARRERA AS ENUM('EVENTO', 'CIRCUITO');
CREATE TYPE MODALIDAD AS ENUM('TRAZADO', 'SCORE');
```

Si la carrera tiene asignada una ubicación geográfica, sus coordenadas se reflejan en las columnas de *LATITUD* y *LONGITUD*.

Por último, los datos de los controles de la carrera se persisten en un campo JSON. Su estructura es la de un *JsonObject* para poder realizar una conversión inmediata a un mapa asociativo de cadenas de caracteres a datos del propio control.

6.2.3. Tabla Recorridos

Se utiliza la clave foránea *CARRERA_ID* para referenciar la carrera a la que pertenece el recorrido.

El trazado del recorrido se persiste como un JsonArray con los códigos de los controles que lo componen, en una columna de tipo JSON.

Como el organizador puede asignar un mapa a los recorridos, estos se guardan como un array de bytes en la base de datos.

6.2.4. Tabla Participaciones

Se utilizan las claves foráneas *CORREDOR_ID* y *RECORRIDO_ID* para asociar una participación a un usuario y a un recorrido, respectivamente. Se añade la restricción de que la tupla de ambas claves foráneas sea única, para garantizar que un usuario solo puede participar en un recorrido una única vez.

Se recoge la fecha de inicio como una marca de tiempo en el momento de la creación en la base de datos.

Se añaden dos valores booleanos para controlar el estado de la participación.

6.2.5. Tabla Tokens

Se utiliza la clave foránea *USUARIO_ID* para asociar el token de autenticación a un usuario. El token es usado como clave primaria, por lo que se asegura su unicidad.

6.2.6. Comentarios

En un principio el diseño inicial de la base de datos contaba con más tablas acorde al modelo relacional, en concreto las tablas para los controles, los registros de las participaciones y el trazado de los recorridos. El problema era que para los datos de prueba generados se creaban más de 45 000 registros, 4000 componentes de los trazados y 3000 controles; cada uno con una fila en la tabla correspondiente. Debido a que el servidor en el que se desplegará el proyecto tiene una limitación de uso máximo de 10 000 filas (en su versión gratuita), se optó por redefinir la estructura de forma que estas colecciones pasasen a recogerse en una columna de tipo JSON, soportada de forma nativa por PostgreSQL.

Esta reestructuración fue un acierto debido a que la lógica del negocio se simplificó y se hizo más manejable. Además, si antes los datos de prueba tardaban en generarse más de 2 minutos, ahora han pasado a generarse en 15 segundos y con un total menor de 3000 filas.

6.3. Arquitectura de diseño

6.3.1. Diseño y arquitectura de la API

Se han establecido cuatro ramas principales para los endpoint de la API dependiendo de los recursos que controla, cada una con su controlador correspondiente.

- **Autenticación**: se encarga de gestionar las conexiones, desconexiones, registros y cambios de contraseña de los usuarios del sistema.
- **Usuarios**: se encarga de acceder a los datos de un usuario y permite cambiar sus datos (nombre y club).
- **Carreras**: se encarga de los aspectos principales de las carreras (creación, edición, borrado), búsqueda de carreras a partir de parámetros de búsqueda o de organización/participación y de generar los archivos relacionados con la gestión de una carrera (contenido de los códigos QR y mapas en un archivo comprimido).
- **Recorridos**: se encarga de la interacción de los usuarios participantes con los recorridos: inicio de recorrido, abandono, registro de control y descarga de mapa. También genera los datos necesarios para interpretar los resultados de un recorrido.

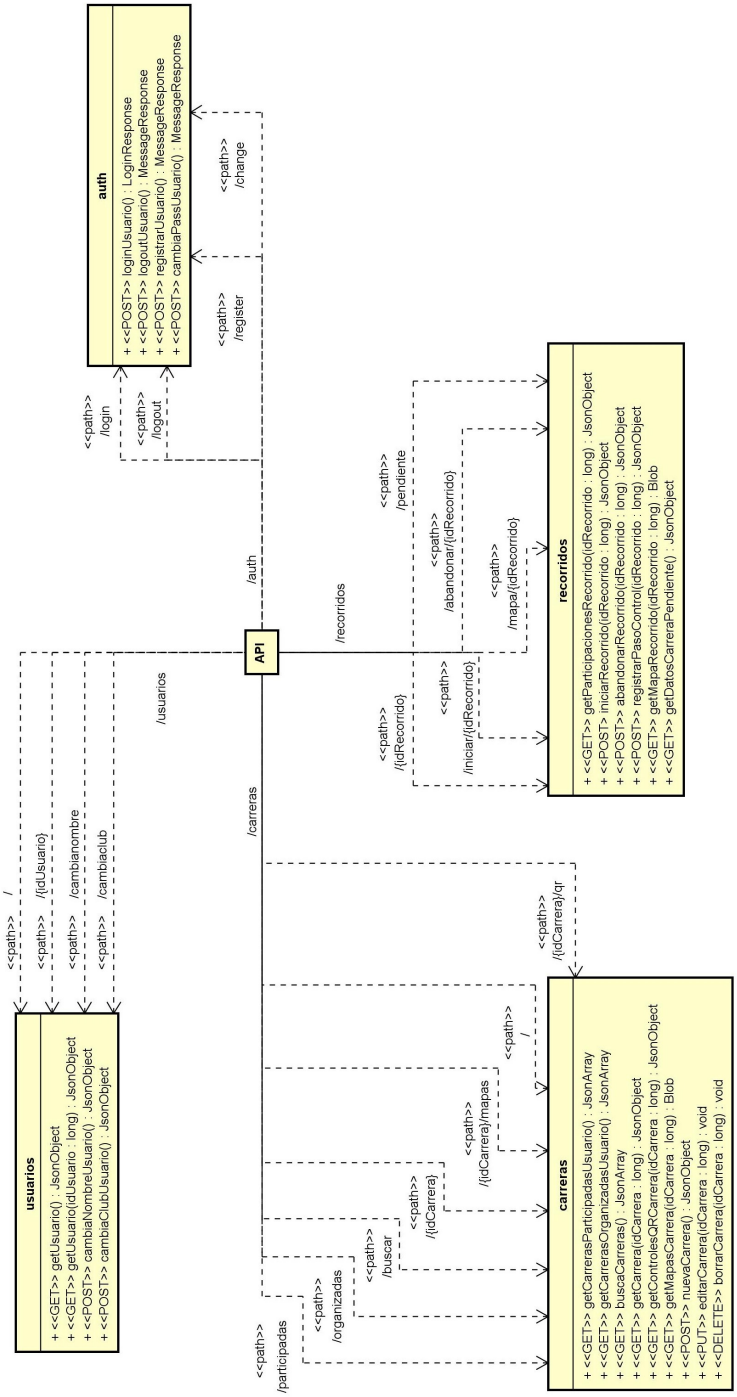

Figura 6.17: Esquema de la API

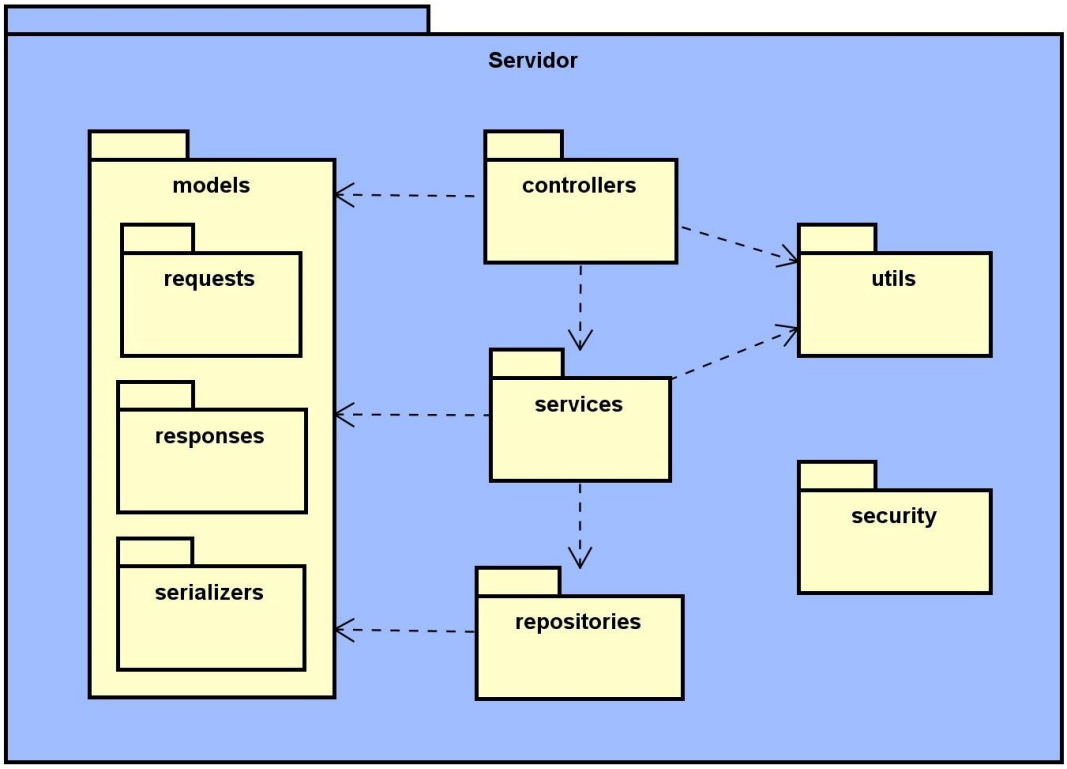

Figura 6.18: Dependencias en la API

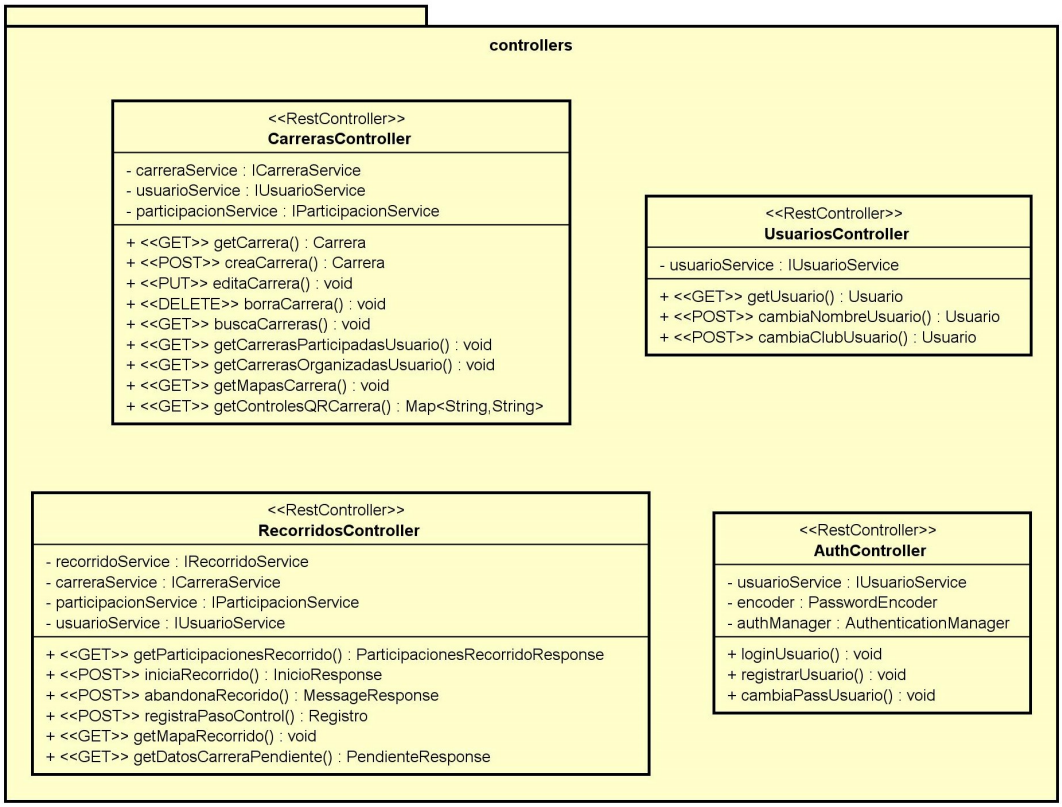

Figura 6.19: Controladores de la API

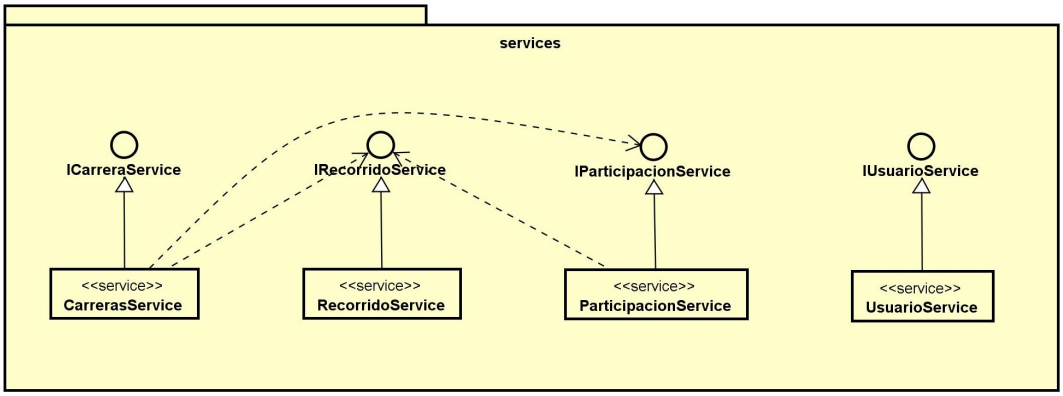

Figura 6.20: Dependencia de servicios de la API

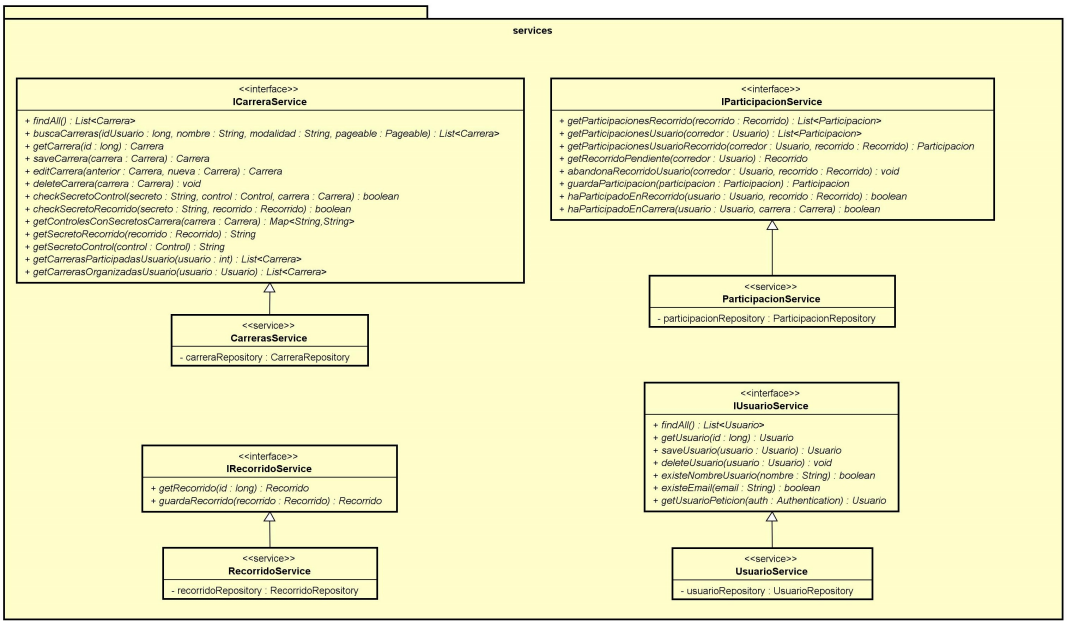

Figura 6.21: Detalle de servicios de la API

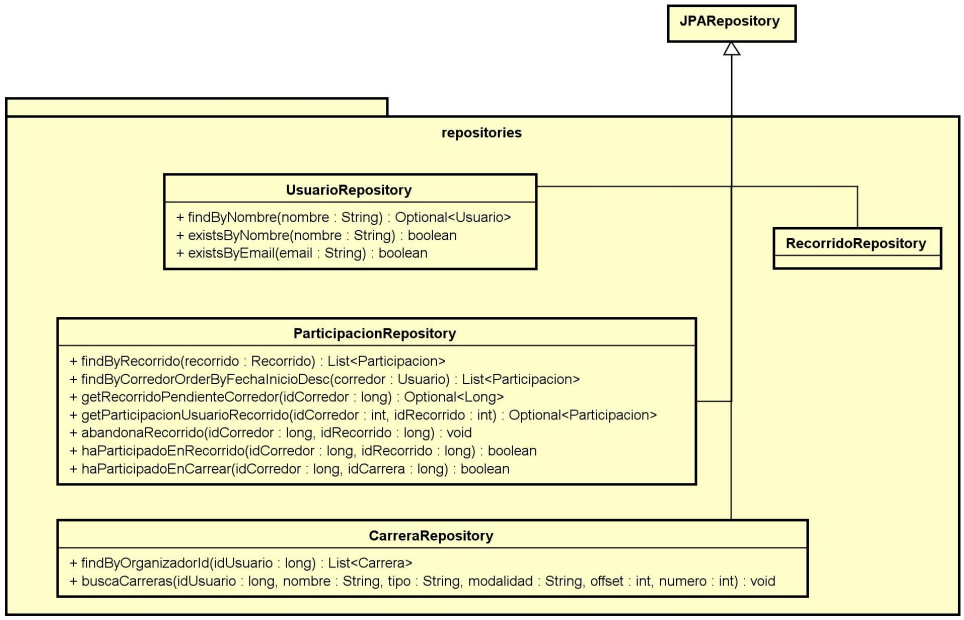

Figura 6.22: Repositorios de la API

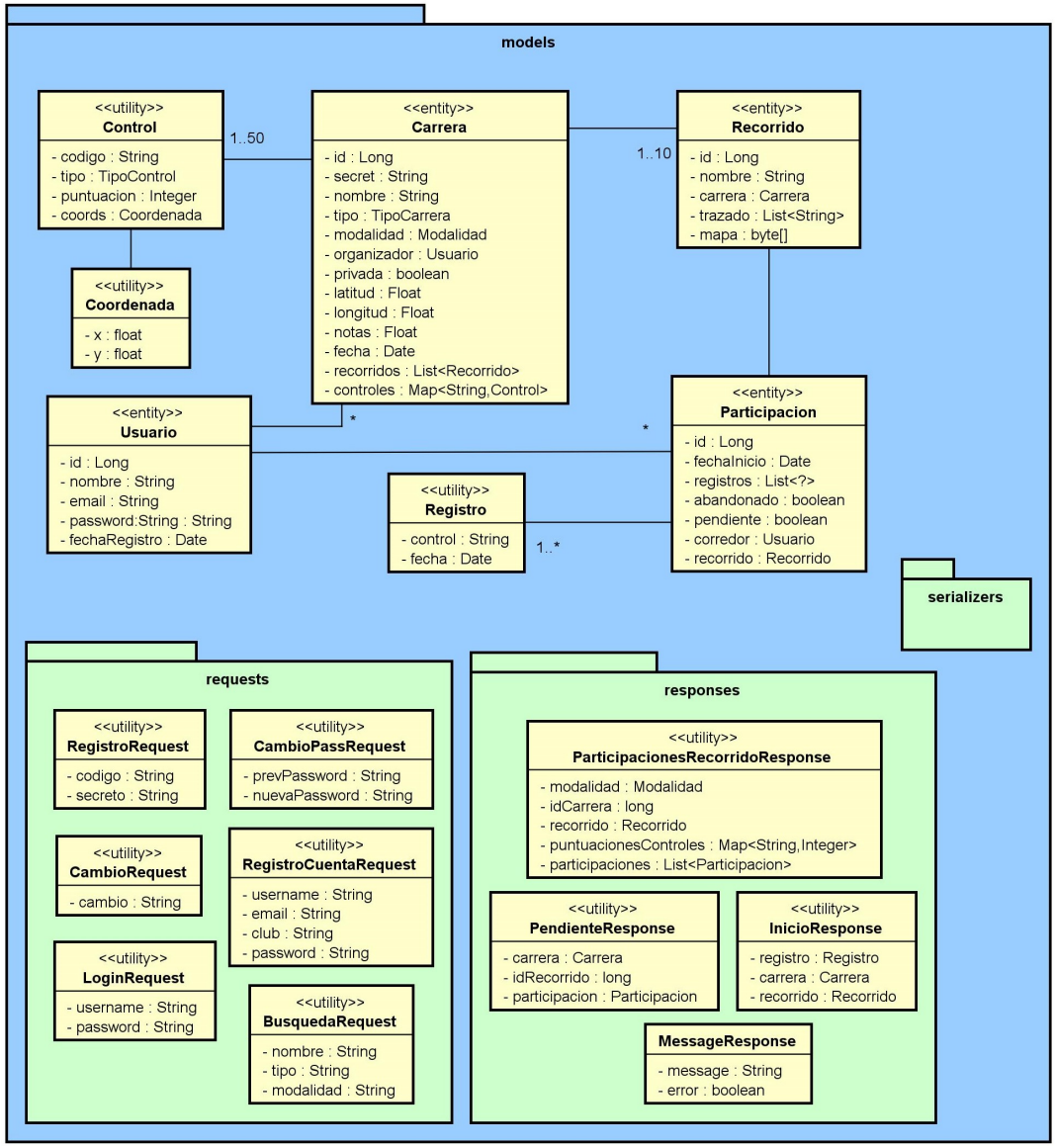

Figura 6.23: Modelos de la API

6.3.2. Arquitectura del cliente Android

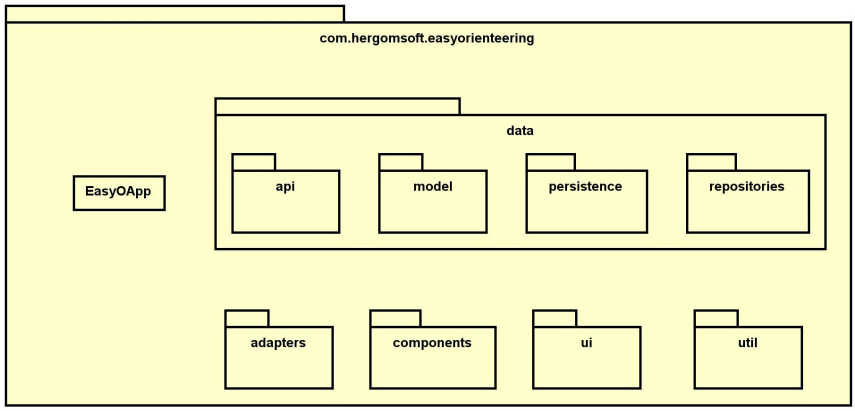

Figura 6.24: Estructura general de la app

Se omiten los detalles en aquellas entidades que son semejantes a las de la API:

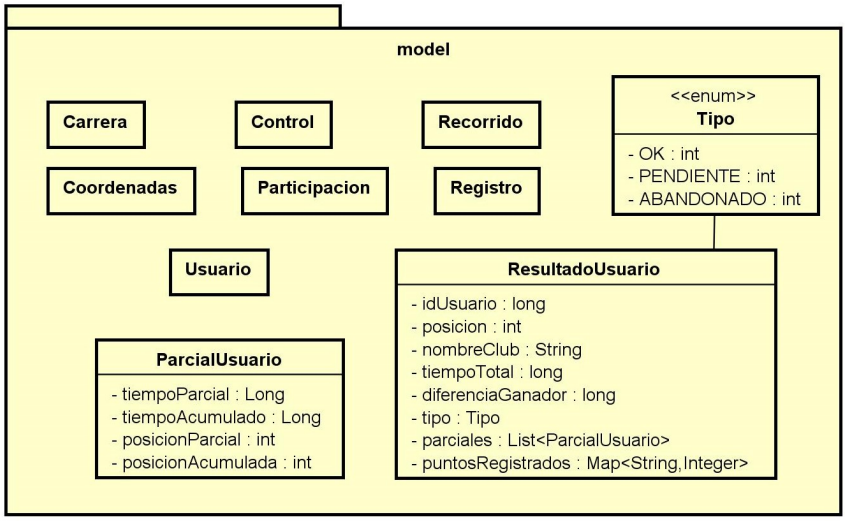

Figura 6.25: Diagrama de paquete *model*

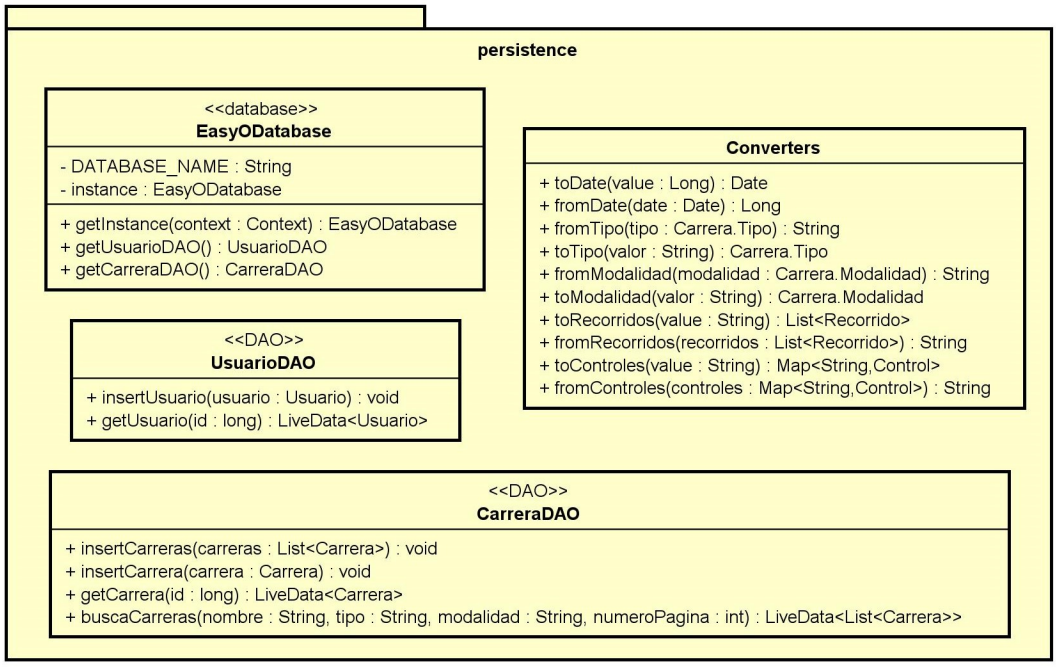

Figura 6.26: Diagrama de paquete *persistence*

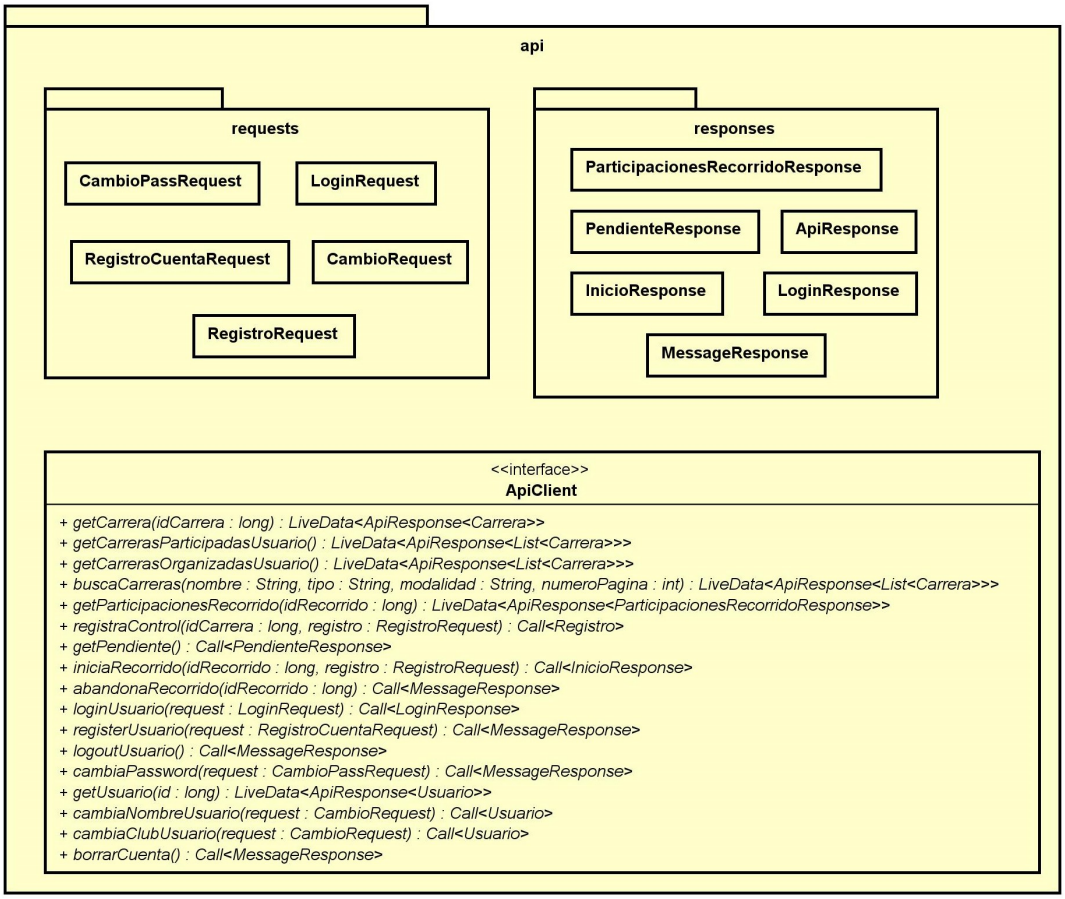

Figura 6.27: Diagrama de paquete *api*

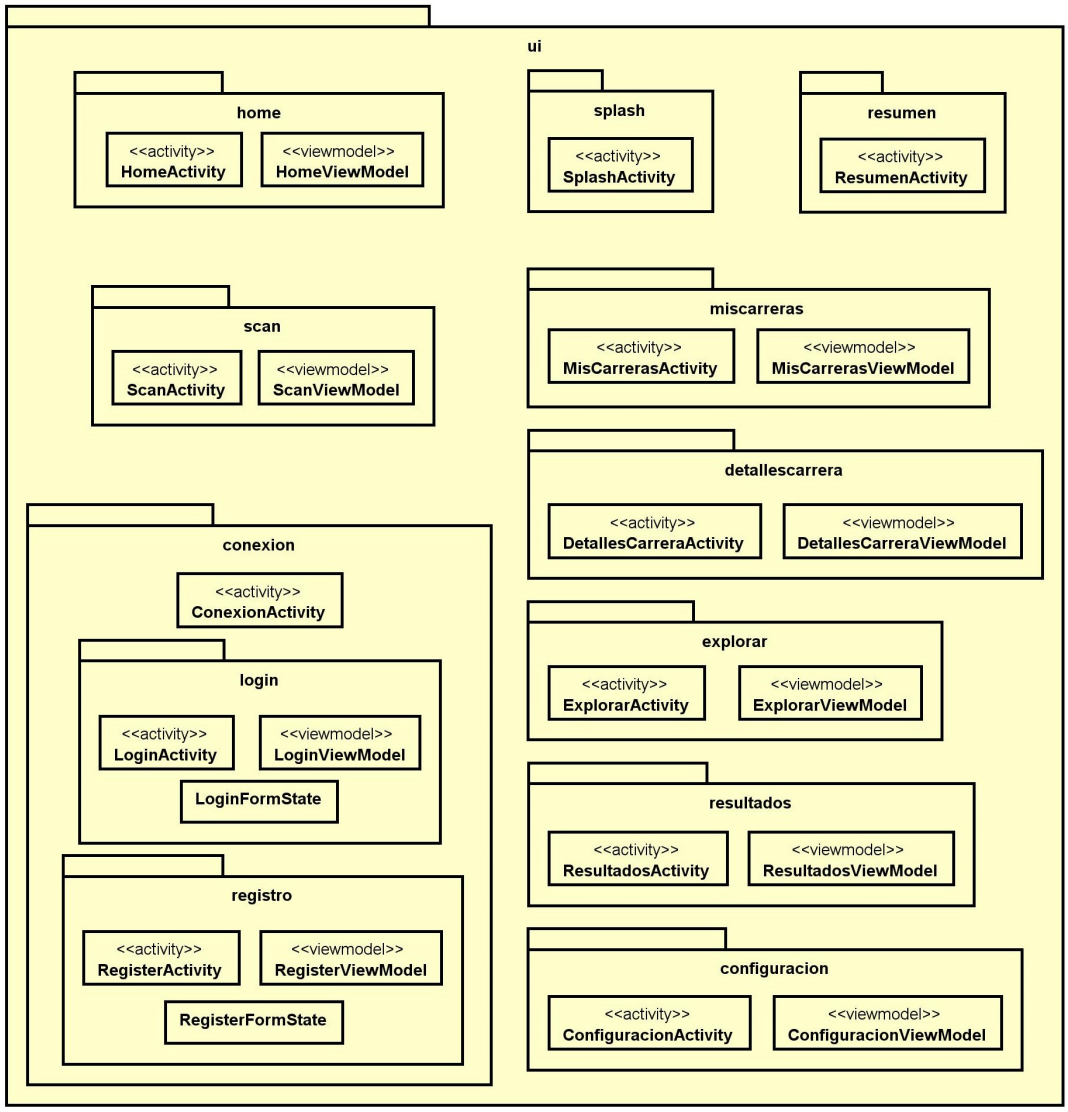

Figura 6.28: Diagrama de paquete *ui*

6.3.3. Arquitectura del cliente web

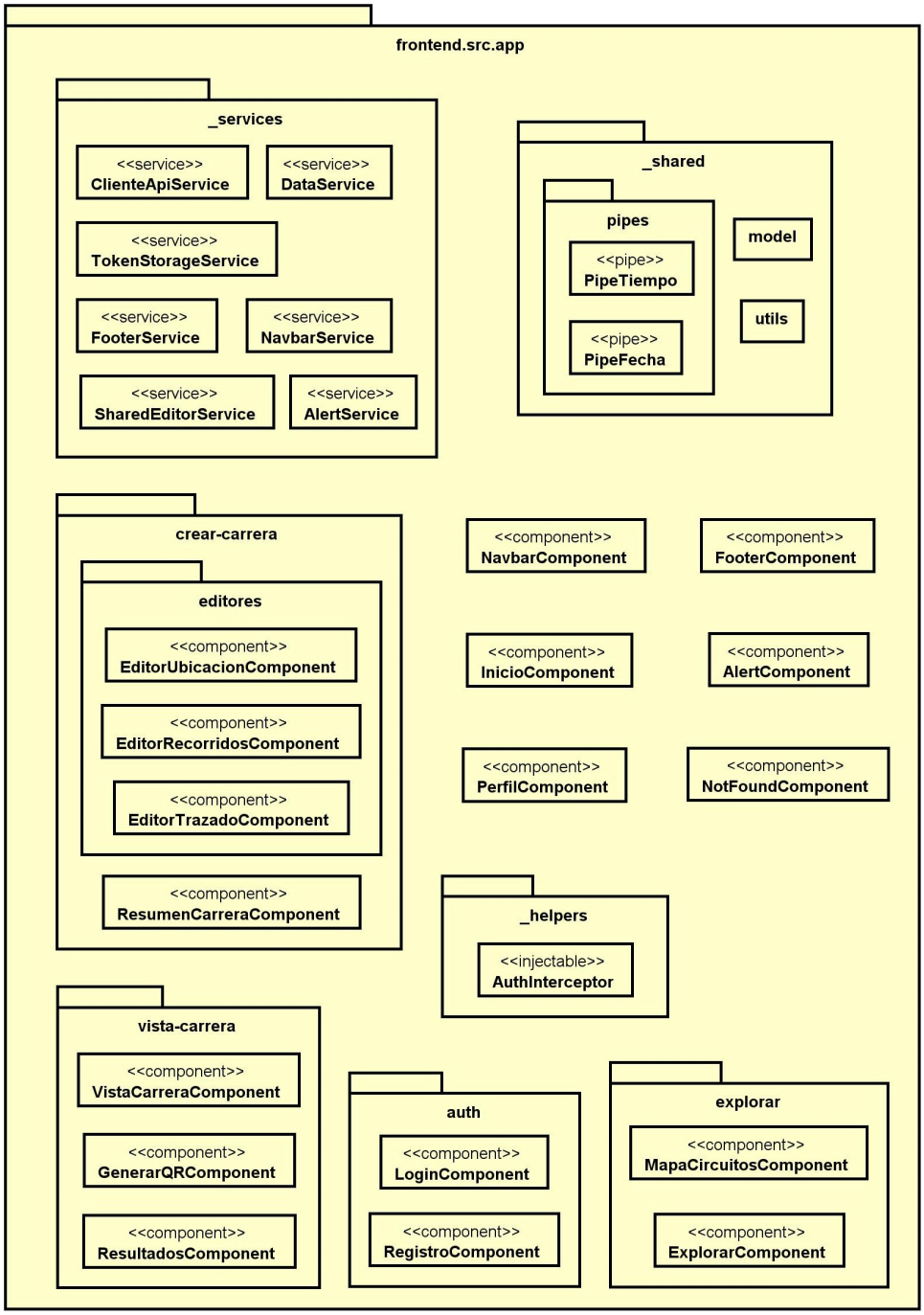

Figura 6.29: Diagrama del cliente web

Capítulo 7

Despliegue y Pruebas

Lo ideal sería que las aplicaciones se ejecutaran en la misma máquina para reducir la latencia de red, al igual que la base de datos, pero debido a las restricciones y configuraciones de Heroku el esquema de despliegue es el siguiente:

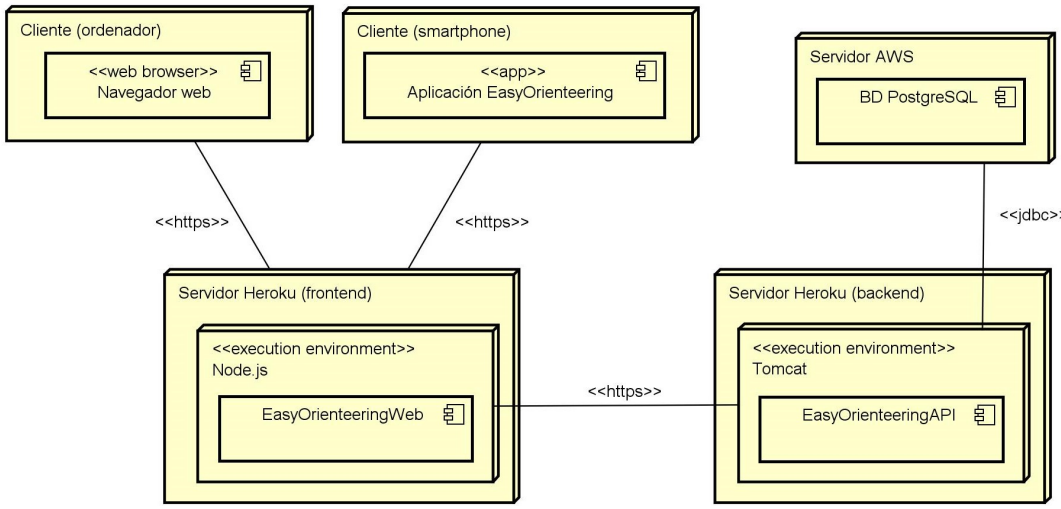

Figura 7.1: Esquema de despliegue en Heroku

7.1. Datos de prueba

Se han generado de forma automática y aleatoria los siguientes elementos:

- **300 usuarios**
  - Nombre de usuario: a partir de un diccionario de nombres extraído de datos del INE [46]. Añadidos con terminaciones.
  - Email: nombre normal del usuario + *@test.com*
  - Club: a partir de un diccionario de clubes extraído de datos del servicio SICO de la FEDO (https://sico.fedo.org/). Probabilidad del 70 % de tener un club asignado.
  - Contraseña: para facilitar la depuración y comprobaciones finales es el nombre del usuario.
- **100 carreras** (50 eventos, 50 circuitos)
  - Nombre: aleatorio según prefijos comunes y un nombre de una localidad española [47].
  - Organizador: aleatorio
  - Notas: número aleatorio de párrafos de Lorem ipsum [48]
  - Latitud y longitud correspondientes a la localidad [47]. Optativas en eventos (80 % de que haya) y obligatorias en circuitos.
  - Privacidad. Probabilidad del 70 % de que sean públicas (eventos) y 100 % en circuitos.
  - Controles: aleatorios. Si es una carrera score se asigna la puntuación asociada al valor de las decenas del código de control.
  - Recorridos: aleatorios en una carrera trazada.
- **Resultados**: para cada carrera generada se elige un número aleatorio de corredores de entre los usuarios generados.
  - En el caso de una carrera con recorridos trazados, se itera cada control de un recorrido, generando un tiempo aleatorio "óptimo". Para cada corredor, se genera una desviación positiva de este tiempo óptimo que resulta en su parcial del control.
  - En el caso de una carrera score, se calcula en un rango un porcentaje aleatorio de corredores que registran todos los controles. Los restantes se calcula en un rango un porcentaje aleatorio de controles registrados, también aleatorios.

7.2. Variables de entorno

Se deben configurar las siguientes variables de entorno:

```
SPRING_DATASOURCE_URL=jdbc:postgresql://IP:PUERTO/BD
SPRING_DATASOURCE_USERNAME=postgres
SPRING_DATASOURCE_PASSWORD=<redacted>
EASYO_CARRERAS_SECRETGENERAL=<redacted>
```

7.3. Configuración de PostgreSQL

1. Instalar PostgreSQL:

```
sudo apt-get install postgresql
```

2. Cambio de contraseña del usuario postgres:

```
sudo -u postgres psql postgres
postgres=# \password postgres
```

3. Creación de un nuevo usuario de gestión de la BD:

```
sudo -u postgres createuser --interactive --password gestordb
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) y
Shall the new role be allowed to create more new roles? (y/n) n
Password:
```

4. Creación de la BD:

```
sudo -u postgres createdb easyodb -O gestordb
```

5. Configuración de pg_hba.conf:

```
sudo vi /etc/postgresql/<version>/main/pg_hba.conf
```

6. Configuración de postgresql.conf (si se requiere conexión remota):

```
sudo vi /etc/postgresql/<version>/main/postgresql.conf
# Descomentar la variable listen_addresses y configurar
  según se requiera conexión local (localhost) o remota (*).
```

7. Configuración del timezone a GMT

```
psql -U gestordb -d easyodb
#easyodb => SET TIMEZONE='GMT';
```

7.4. Spring Boot

Contenido de application.properties:

```
spring.data.rest.basePath=/api

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults=false
spring.jpa.hibernate.ddl-auto=update

spring.jackson.serialization.fail-on-empty-beans=false
spring.jackson.time-zone=UTC
```

7.5. Angular

7.5.1. Redirección de peticiones a back-end

Las peticiones realizadas a la API desde el navegador web irán a través de un proxy local, para que todas las interacciones del usuario sean a través del puerto web.

Para el entorno de desarrollo local, crear el archivo **proxy.conf.json** en la raíz del proyecto Angular con el siguiente contenido:

```
{
    "/api/*": {
    "target": "http://localhost:8080",
    "secure": true,
    "logLevel": "debug",
    "changeOrigin": true
  }
}
```

Para el entorno de producción, se utiliza una redirección en el servidor Express mediante el uso del paquete **http-proxy-middleware** [49] y de la configuración de la variable de entorno *EASYO_BACKEND*:

```
...

// Proxy API
var proxyOptions = {
    target: 'https://' + process.env.EASYO_BACKEND,
    changeOrigin: true
}
app.use('/api', createProxyMiddleware(proxyOptions));

...
```

Capítulo 8

Conclusiones

Se han conseguido realizar los objetivos principales del proyecto (ver 1.3.1). Sin embargo, se han dejado sin implementar algunos detalles que hacen que el nivel de satisfacción en cuanto al resultado del proyecto se vea influido negativamente.

El desarrollo del proyecto se ha guiado principalmente con el uso de tecnologías y herramientas libres y gratuitas, no solo por el ahorro en costes del proyecto sino también por aumentar la visibilidad y popularidad de estas para apoyar su desarrollo. El repositorio con los proyectos se ha hecho público, para que cualquier otro desarrollador pueda corregirlos, mejorarlos e incluir nuevas funcionalidades. La dirección del respositorio es **https://github.com/Zarkrosh/EasyOrienteering**.

Aparte de haber conseguido la satisfacción de hacer realidad una idea pendiente de hace años, durante el transcurso del proyecto se han aprendido bastantes cosas que han mejorado la capacidad de desarrollo del miembro del equipo y que, con seguridad, reutilizará en sus proyectos futuros.

8.1. Objetivos no conseguidos

- Restablecimiento de la contraseña del usuario (US-012).

El desarrollo de este proyecto ha sido en múltiples ocasiones duro debido al desconocimiento inicial de los frameworks, en especial los de persistencia (Hibernate y Room).

8.2. Ampliaciones futuras

- Asignación de tiempo máximo de participación en una carrera, tras el cual quedará marcada como "*fuera de tiempo*".
- Ocultación de participaciones para usuarios que no quieran aparecer en los resultados de un recorrido.
- Mejoras en el trazador web: inserción de control entre varios, etc.
- Límite de tiempo en carreras score con penalización de puntuación por exceso.
- Vista móvil de la web con Bootstrap.
- Generación de recorridos y mapa a partir de archivos de trazado de herramientas existentes: PurplePen y OCAD.
- Gráficos en la vista de resultados (evolución de puestos, agrupamientos, etc).
- Uso de la especificación internacional de descripciones de control [51].
- Georeferenciación de mapas: el usuario asocia 3 puntos separados del mapa con 3 puntos en una ortofoto para triangular.
- Modo ayuda: si está activado el modo y el GPS del móvil, la ubicación aparece en el mapa georeferenciado. Puede ser permitido o no por el organizador, y activado por el usuario durante una carrera.
- Personalización de apariencia y tamaño de controles QR.
- Comentarios y puntuación de carreras.
- Reportar error en carrera
- Visualización de rutas GPS de corredores en mapa georeferenciado (subida de GPX o posible sincronización con algún servicio).
- Cronometraje y registro de controles mediante GPS.

Apéndice A

Manual de usuario

A.1. Registro de cuenta y conexión

Ya sea un usuario que va a tomar el papel de organizador o de participante, puede registrarse en el sistema tanto desde la aplicación web como la aplicación móvil. En el caso de la página web, en la esquina superior derecha podrá elegir la opción 'Registrarse' y rellenar el formulario con los datos necesarios. En el caso de la aplicación móvil, tras visualizar una pantalla inicial en la que se explica el sistema, puede elegir la opción 'Registrarse' y rellenar el formulario como en el caso anterior.

Consideraciones a tener en cuenta:

- El nombre de usuario es único y público. Ese será el nombre con el que aparecerá en los resultados y detalles de una carrera.
- Solo puede asociarse una única dirección de email a una cuenta de usuario.
- El campo de club del usuario es opcional.
- La contraseña del usuario debe tener al menos 8 caracteres.

Cuando un usuario haya registrado una cuenta, podrá conectarse al sistema tanto desde la web como desde el dispositivo móvil. Si las credenciales de acceso son incorrectas se le informa del error.

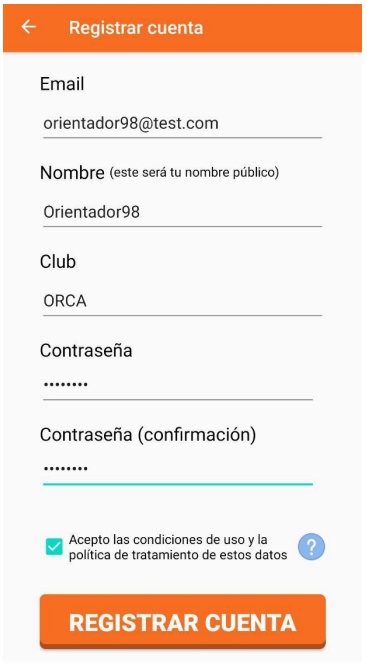

Figura A.1: Formulario de registro (móvil)

Figura A.2: Formulario de conexión (móvil)

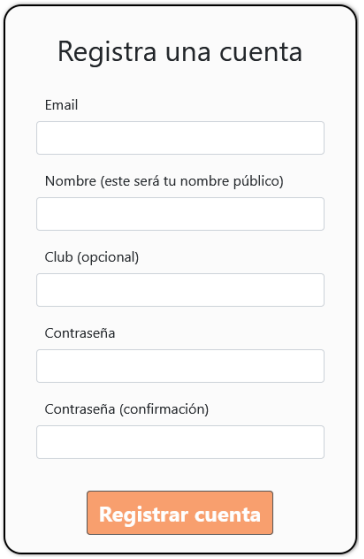

Figura A.3: Formulario de registro (web)

Figura A.4: Formulario de conexión (web)

A.2. Perfil de usuario

Una vez conectado, en la pantalla de su perfil puede modificar su nombre de usuario, su club y su contraseña si así lo desea, siempre y cuando los cambios sean válidos. En el caso de la página web, en su perfil también podrá visualizar las carreras en las que ha participado y que ha organizado. Esta misma vista la tiene el usuario del cliente móvil en la pantalla *'Mis carreras'*.

Figura A.5: Pantalla de perfil (web)

Figura A.6: Pantalla de inicio (móvil)

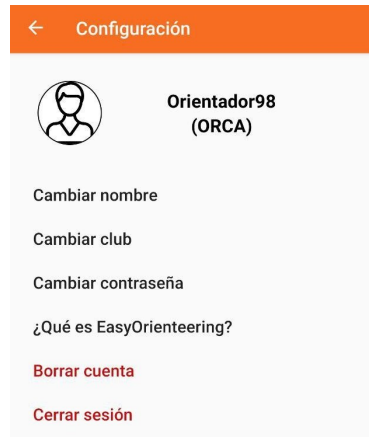

Figura A.7: Pantalla de perfil (móvil)

Figura A.8: Carreras participadas (móvil)

Figura A.9: Carreras organizadas (móvil)

A.3. Cambio de datos de usuario

Para modificar su nombre de usuario o club, el usuario debe elegir la opción deseada y proporcionar un valor válido para el cambio.

- **Nombre de usuario**: longitud de entre 3 y 30 caracteres y que no esté siendo utilizado por otro usuario.

- **Club**: longitud de entre 0 y 30 caracteres.

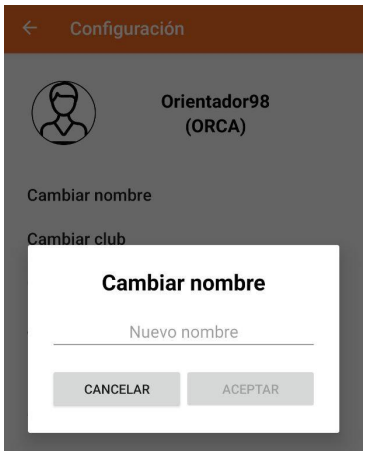

Figura A.10: Cambio de datos de usuario (móvil)

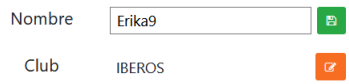

Figura A.11: Cambio de datos de usuario (web)

A.4. Cambio de contraseña

Para cambiar su contraseña, el usuario debe introducir la contraseña actual y proporcionar una nueva que cumpla los requisitos de seguridad: al menos 8 caracteres de longitud.

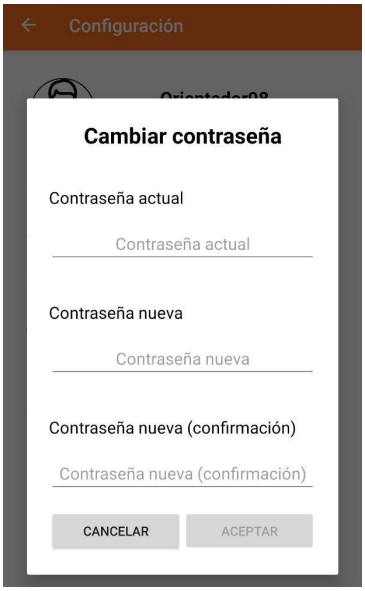

Figura A.12: Cambio de contraseña (móvil)

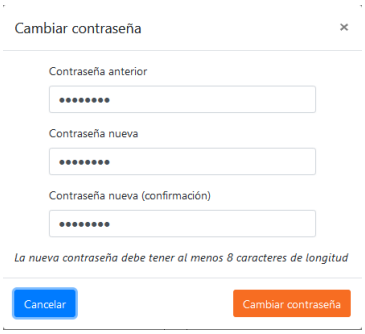

Figura A.13: Cambio de contraseña (web)

A.5. Vista de carrera y resultados

Desde la web, tanto si el usuario está autenticado como si no, se pueden visualizar los detalles y resultados de todas las carreras exceptuando las privadas, las cuales solo pueden accedidas por su organizador y por sus participantes (en los recorridos que hayan participado).

En el caso de las carreras trazadas, los resultados se muestran como una tabla de posiciones con los tiempos parciales de cada control, que indican las posiciones relativas en cuanto al tiempo parcial y acumulado hasta el momento.

En el caso de las carreras score, se muestra una tabla de posiciones con las puntuaciones obtenidas por los usuarios atendiendo a la puntuación de los controles de la carrera, mostrando cuáles ha picado el usuario y cuáles no.

En el caso de que un corredor haya abandonado o no haya acabado aún su carrera, también aparecerán en los resultados, con la marca correspondiente.

Resultados R1

#	Nombre / Club	Tiempo	S-1 (36)	1-2 (50)	2-3 (46)	3-4 (43)	4-5 (37)
1	Unai_13 UNIVERSIDAD ALICANTE	1:00.39	02.35 (2) 02.35 (2)	03.20 (3) 05.55 (2)	09.10 (5) 15.05 (4)	**04.18 (1)** 19.23 (3)	06.42 (5) 26.05 (5)
2	Ainara_483	1:00.53 +00.14	02.37 (3) 02.37 (3)	**02.51 (1)** **05.28 (1)**	07.47 (2) 13.15 (2)	05.10 (4) 18.25 (2)	04.57 (4) 23.22 (2)
3	Valentin400 O-RIENTATE	1:00.56 +00.17	03.15 (5) 03.15 (5)	03.01 (2) 06.16 (3)	**06.48 (1)** **13.04 (1)**	04.54 (3) **17.58 (1)**	04.38 (2) **22.36 (1)**
4	Judith39	1:00.59 +00.20	03.10 (4) 03.10 (4)	03.42 (4) 06.52 (5)	08.16 (4) 15.08 (5)	05.27 (5) 20.35 (5)	04.39 (3) 25.14 (4)
5	Salvador_97 SOTOBOSQUE	1:01.10 +00.31	**02.25 (1)** **02.25 (1)**	04.05 (5) 06.30 (4)	08.12 (3) 14.42 (3)	04.47 (2) 19.29 (4)	**04.34 (1)** 24.03 (3)

Figura A.14: Resultados de carrera trazada (móvil)

Resultados SCORE

#	Nombre / Club	Tiempo	Puntuación	31	32	33	34	35	36	37	38	39	META
1	Daniel_7463 APU-O	26.31	27	X	X	X	X	X	X	X	X	X	X
2	Rosalia_6	26.51	27	X	X	X	X	X	X	X	X	X	X
3	Susana464	31.03	27	X	X	X	X	X	X	X	X	X	X
4	Valeria77 NORTE-SUR	34.45	27	X	X	X	X	X	X	X	X	X	X
5	Elsa_3023 COHU	42.53	27	X	X	X	X	X	X	X	X	X	X
6	Francesc_68 GALITIUS	16.37	21	X	X		X	X	X	X		X	X

Figura A.15: Resultados de carrera score (móvil)

Volver a carrera

Actualizar

Resultados *R3*

#	Nombre Club	Tiempo	S-1 (41)	1-2 (32)	2-3 (34)	3-4 (35)	4-5 (39)	5-6 (31)	6-7 (33)	7-8 (37)	8-9 (38)	9-10
1	Luis2711	**53:08**	1:47 (8)	5:12 (11)	7:11 (8)	**4:27 (3)**	**1:25 (3)**	2:09 (14)	7:59 (5)	**5:45 (1)**	3:30 (7)	**6:09**
	COAraba		1:47 (8)	6:59 (11)	14:10 (9)	18:37 (7)	20:02 (6)	22:11 (7)	30:10 (6)	35:55 (4)	39:25 (4)	**45:34**
2	Julian978	**53:09**	**1:27 (1)**	**4:27 (2)**	7:10 (7)	4:46 (6)	1:43 (10)	2:07 (13)	**6:49 (2)**	**5:51 (2)**	**2:58 (2)**	7:30
		+0:01	**1:27 (1)**	**5:54 (2)**	**13:04 (2)**	**17:50 (2)**	**19:33 (2)**	21:40 (5)	**28:29 (1)**	**34:20 (1)**	**37:18 (1)**	**44:48**
3	Hugo4742	**53:48**	1:37 (4)	5:13 (12)	6:42 (4)	**4:26 (2)**	1:44 (11)	1:38 (4)	7:40 (4)	**5:45 (1)**	3:24 (6)	7:45
	IES SABON	**+0:40**	1:37 (4)	6:50 (10)	13:32 (5)	17:58 (4)	19:42 (5)	21:20 (4)	**29:00 (3)**	**34:45 (2)**	**38:09 (2)**	**45:54**
4	Rosalia_6	**54:51**	**1:31 (2)**	**4:31 (3)**	7:36 (10)	4:31 (4)	1:31 (6)	**1:33 (3)**	**7:22 (3)**	7:06 (7)	3:42 (9)	7:38
		+1:43	**1:31 (2)**	**6:02 (3)**	13:38 (7)	18:09 (5)	19:40 (4)	**21:13 (3)**	**28:35 (2)**	**35:41 (3)**	**39:23 (3)**	47:01
5	Eusebio_3	**56:25**	1:49 (9)	**3:57 (1)**	6:54 (5)	5:12 (8)	1:44 (11)	**1:32 (2)**	8:35 (9)	8:10 (13)	3:04 (4)	6:50
		+3:17	1:49 (9)	**5:46 (1)**	**12:40 (1)**	**17:52 (3)**	**19:36 (3)**	**21:08 (2)**	29:43 (4)	37:53 (8)	40:57 (6)	47:47
6	Ariadna_6509	**56:35**	1:54 (10)	**4:31 (3)**	**6:41 (3)**	4:33 (5)	1:29 (5)	1:57 (10)	8:52 (10)	6:04 (4)	3:33 (8)	7:44
	COMCU	**+3:27**	1:54 (10)	6:25 (4)	**13:06 (3)**	**17:39 (1)**	**19:08 (1)**	**21:05 (1)**	29:57 (5)	36:01 (5)	39:34 (5)	47:18
7	Elsa_3023	**57:05**	2:05 (13)	4:35 (5)	6:57 (6)	5:43 (11)	1:37 (8)	1:53 (9)	8:25 (8)	**5:56 (3)**	3:55 (10)	**6:10**
	COHU	**+3:57**	2:05 (13)	6:40 (8)	13:37 (6)	19:20 (8)	20:57 (8)	22:50 (9)	31:15 (9)	37:11 (7)	41:06 (8)	47:16
8	Ainara_483	**57:27**	1:37 (4)	4:53 (8)	9:27 (14)	5:11 (7)	**1:24 (2)**	**1:31 (1)**	**6:42 (1)**	6:05 (5)	4:15 (15)	6:50
		+4:19	1:37 (4)	6:30 (6)	15:57 (14)	21:08 (13)	22:32 (13)	24:03 (12)	30:45 (7)	36:50 (6)	41:05 (7)	47:55

Figura A.16: Resultados de carrera trazada (web)

Figura A.17: Resultados de carrera score (web)

A.6. Creación de carrera

En la página web, tras la autenticación del usuario se mostrará la opción 'Crear carrera' en la barra superior de navegación. Si el usuario pulsa en ella, se mostrará el formulario con los datos a rellenar de la carrera.

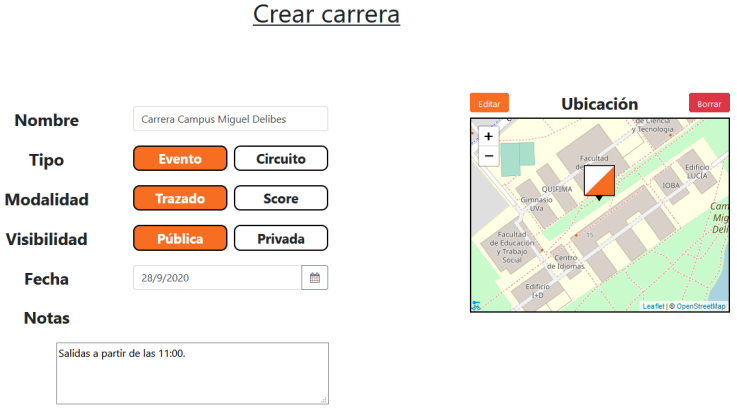

Figura A.18: Formulario principal de creación de carrera

Es importante tener en cuenta que una carrera puede ser o bien un evento, o bien un circuito. La diferencia radica en la disponibilidad: los eventos se realizan en una fecha concreta, en la que el organizador coloca los controles y luego son retirados; mientras que los circuitos

tratan de lograr una disponibilidad de los controles permanente. Además, las carreras que sean del tipo circuito aparecerán en un mapa especial en la sección '*Explorar*'.

Es aconsejable asignar un **nombre descriptivo para la carrera**, acorde a su tipo y sin dar lugar a la ambigüedad. Por ejemplo, si se trata de una carrera para un circuito permanente, se puede usar un nombre parecido a '*Circuito Permanente Campus Miguel Delibes*' o '*CPO Campus Miguel Delibes*'. Si en cambio, se trata de un evento para una prueba popular se puede utilizar un nombre como '*1ª Carrera popular Campus Miguel Delibes*'.

Las carreras tienen **2 modalidades**:

- Las **carreras trazadas** pueden tener varios recorridos, cada uno con un trazado distinto. Tiene la ventaja de que se pueden crear recorridos de distinta dificultad para que los participantes corran a niveles distintos, y también la ventaja de aprovechar de diferentes formas el mismo terreno y el mismo mapa.
- Por el contrario, las **carreras score** no tienen recorridos con trazado. En el mapa solo se muestran los controles que existen en la carrera, y cada control tiene una puntuación asignada por el organizador. El orden de paso por los controles es responsabilidad del participante, añadiendo un factor adicional de estrategia.

Se puede modificar la visibilidad de una carrera, de forma que si una carrera se marca como privada no aparecerá en las búsquedas de aquellos que no han participado en ella, ni tampoco será accesible. Es importante tener en cuenta que **un circuito debe ser obligatoriamente público**.

Para un evento se puede asignar una fecha de realización, siempre posterior o igual al día en que se crea la carrera. No delimita la participación en ese día, sino que sirve como dato adicional útil en las búsquedas.

Las **notas** de una carrera son importantes, puesto que los participantes probablemente necesiten alguna **información adicional sobre la carrera**. Se puede introducir información acerca del acceso a la zona de carrera, ubicación de los mapas (si hay), horario de salidas, etc. Es importante tener en cuenta que esta información es pública, por lo que no se recomienda la exposición de datos personales como números de teléfono, direcciones de correo o similares.

Es recomendable asignar una ubicación para los eventos, y obligatorio para los circuitos. Tras pulsar en 'Añadir ubicación', se mostrará un mapa en el que se podrá clicar en el para añadir un marcador (con el nivel de zoom suficiente) y luego guardarla.

Creación de recorridos o controles

Dependiendo de la modalidad, el siguiente paso es crear los recorridos o controles. Pulsando el botón correspondiente, se redirigirá a una página en la que se preguntará acerca del modo de organización: usando el trazador web o describiendo manualmente los controles o recorridos.

Figura A.19: Creación de recorridos

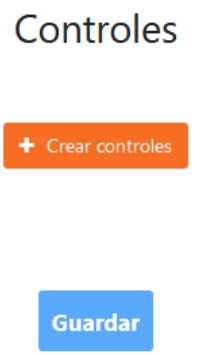

Figura A.20: Creación de controles

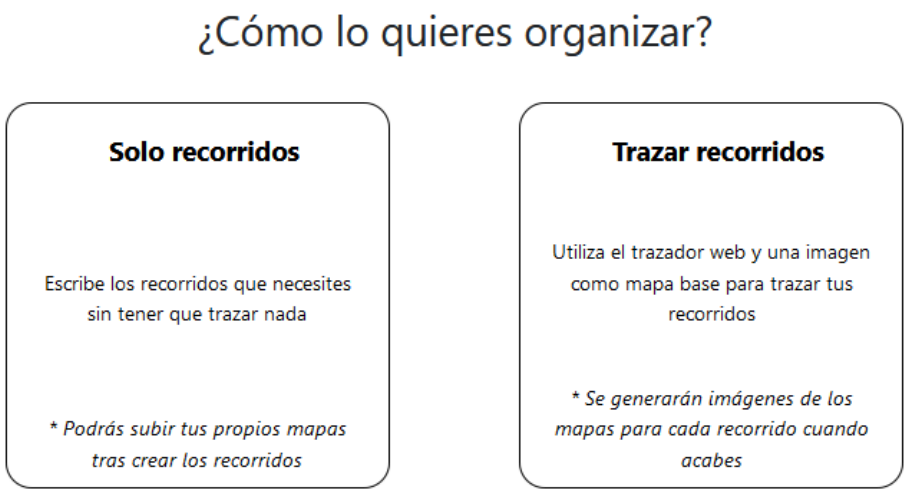

Figura A.21: Elección de tipo de creación

Si la modalidad de carrera es trazada, se podrán crear hasta 10 recorridos con hasta 32 controles por recorrido, con un máximo de 50 controles diferentes en total. Para editar el nombre de un recorrido se puede seleccionar el icono azul que aparece a la izquierda en la barra lateral, o hacer doble click sobre el nombre del recorrido. Para guardar el nombre del recorrido, si es válido, se puede pulsar 'Enter' o seleccionar el icono de la izquierda, que habrá cambiado para indicar el guardado. Para borrar el recorrido actual, pulsar el icono rojo de la derecha, tras lo cual aparecerá un diálogo de confirmación.

Si la modalidad de carrera es score, se podrán crear los controles deseados (hasta un máximo de 50). Aparecerán en una lista en la barra lateral junto con una puntuación por

defecto (valor de las decenas) que puede ser modificada a gusto del organizador.

- **Usando trazador**: para usar el trazador web es obligatorio disponer de una **imagen** que actúe de **mapa base**. Puede ser cualquier imagen, pero es recomendable utilizar un **mapa de orientación**. Para empezar, pulsar en **'Seleccionar mapa base'**, tras lo cual se abrirá un selector de archivos en el que se deberá elegir la imagen deseada. Una vez cargada, se puede comenzar a trazar los controles o recorridos sobre el mapa. Con el **click izquierdo se colocan los controles** y con el **derecho se borran** (haciendo click sobre el control borra el control, haciendo click en general borra el último control añadido). También se puede **resituar un control** pulsando click izquierdo y arrastrando.

  Para añadir un recorrido, basta con pulsar sobre la pestaña con el símbolo '+'. Para cambiar de recorrido se pulsará sobre la pestaña del recorrido deseado. Mientras se está trazando, aparecerán los controles usados en otros recorridos para sugerir su reutilización.

- **Sin usar trazador**: la vista se simplifica dejando a la izquierda la lista de recorridos y a la derecha la barra lateral con los trazados de los recorridos o la puntuación de los controles, según la modalidad de carrera.

En cualquier caso, una vez creados los recorridos o controles se puede pulsar el botón 'Guardar recorridos' o 'Guardar controles' (respectivamente) para volver a la pantalla de resumen de la creación de la carrera. Ahora aparecerán los datos de los controles o recorridos creados, junto con sus mapas si se ha optado por la opción de usar el trazador. En caso contrario, **se pueden subir de forma manual los mapas de los recorridos** haciendo click sobre el botón 'Añadir mapa' del recorrido deseado y seleccionando una imagen con el mapa.

Una vez rellenados los datos necesarios, el usuario puede proceder a la creación de la carrera pulsando el botón 'Guardar' al final del formulario. Si la creación es exitosa, se produce una redirección a la vista de la carrera. Una vez allí, el organizador puede optar por descargar en un archivo comprimido ZIP las imágenes de los mapas (si alguna fue asignada) y generar los códigos QR de los controles de la carrera pulsando el botón **'Generar controles QR'**. Se mostrará una previsualización del PDF que puede descargarse el usuario pulsando el botón **'Descargar PDF'**.

Así entonces, el organizador puede pasar a imprimir el PDF descargado o manipular las imágenes con los códigos QR para situarlos de forma física en el terreno, utilizando el método que prefiera. Se recomienda la protección del elemento en el que está impreso el código, en especial en el caso de los circuitos permanentes, para evitar la degradación ante condiciones adversas como el clima.

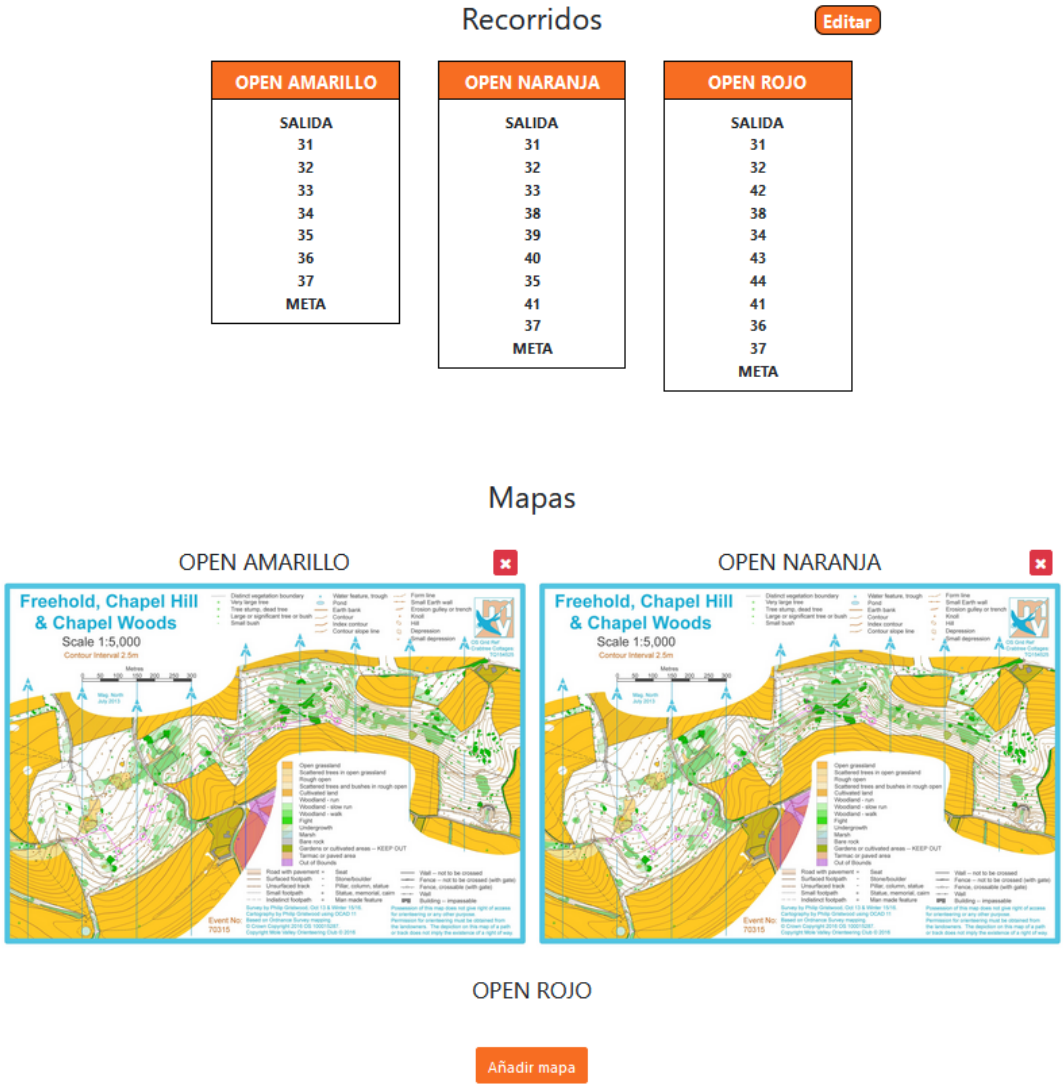

Figura A.22: Resumen con recorridos y mapas

Figura A.23: Generación de códigos QR de los controles

A.7. Participación en carrera

Una vez que un usuario se ha conectado a su cuenta desde la aplicación móvil, puede escoger la opción **'Unirme a carrera'**, tras lo cual es posible que se produzca una petición de los permisos necesarios de la aplicación, como puede ser la toma de imágenes de la cámara. Si el usuario tiene un recorrido pendiente de finalizar, se mostrará un mensaje con las opciones de abandonar o de reanudarlo nada más entrar a la pantalla.

Una vez configurado, se comenzará a escanear en busca de un código QR de salida de un recorrido (triángulo magenta en el centro). Tras su escaneo, se confirma al usuario si desea iniciarlo y, en caso afirmativo y con éxito, se iniciará la participación del usuario en el recorrido.

Si el recorrido es trazado, se indicará al usuario cuál es el siguiente control a registrar y si registra el que no es, se le notificará. Si el recorrido es score, se indica al usuario que puede registrar el control que desee y en caso de que registre uno que ya ha validado, se le notificará. Si el organizador ha dispuesto un mapa para el recorrido en el sistema, el usuario podrá verlo en la aplicación pulsando el botón inferior derecho, que alterna la vista del mapa y de escaneo.

Para finalizar el recorrido, el usuario escanea el control de meta de la carrera, tras lo cual es redirigido a los resultados del recorrido.

Figura A.24: Selección de opción 'Unirme a carrera'

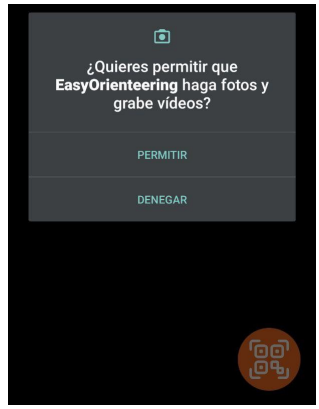

Figura A.25: Solicitud de permisos necesarios

Figura A.26: Escaneo de triángulo de salida

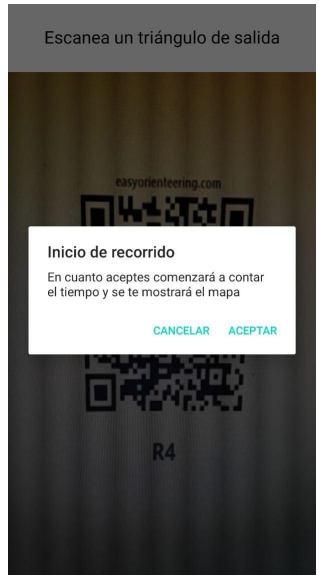

Figura A.27: Confirmación de inicio de recorrido

Figura A.28: Escaneo de control

Figura A.29: Escaneo de control (incorrecto)

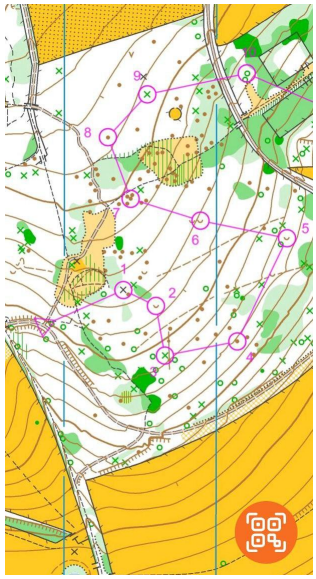

Figura A.30: Vista de mapa

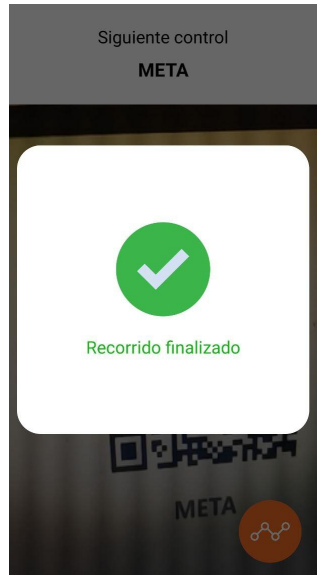

Figura A.31: Escaneo de meta

Apéndice B

Manual de desarrollo y mantenimiento

B.1. Angular

Para instalar las dependencias del proyecto, ejecutar en la raíz del proyecto (/frontend):

```
npm install
```

Nota: una de las dependencias (**html2canvas**) tiene un issue abierto a día de hoy que afecta a la compilación del proyecto [53]. Para evitarlo se debe usar una versión anterior que no presenta dicho problema: **1.0.0-RC1**.

B.2. Spring Boot

B.2.1. Configuración de propiedades

Se deben definir los parámetros de conexión a la base de datos y configuraciones adicionales necesarias.

Se pueden usar dos métodos:

- En las variables de entorno del sistema, definir:

```
SPRING_DATASOURCE_URL=jdbc:postgresql://IP:PUERTO/BD
SPRING_DATASOURCE_USERNAME=postgres
```

```
SPRING_DATASOURCE_PASSWORD=<redacted>
EASYO_CARRERAS_SECRETGENERAL=<redacted>
```

- En el archivo **application.properties** se definen las siguientes propiedades:

```
spring.datasource.url=jdbc:postgresql://IP:PUERTO/BD
spring.datasource.username=postgres
spring.datasource.password=<redacted>
easyo.carreras.secretgeneral=<redacted>
```

Se recomienda el primer método debido a que los datos sensibles no figuran en ningún archivo controlado por Git.

B.3. Despliegue en local

En la raíz del proyecto **frontend** ejecutar el siguiente comando para correr el servidor de desarrollo. Tener en cuenta que hace uso de un certificado y clave para el SSL y que la redirección al backend, por defecto, se hace al localhost en el puerto 8080.

```
npm run-script devstart
```

En la raíz del proyecto **backend** se puede o bien ejecutar el proyecto desde NetBeans o bien mediante maven (mvn easyorienteering-api:run).

B.4. Despliegue en producción

En la raíz del proyecto ejecutar el script ***despliegue_heroku.bat*** ubicado en la raíz del repositorio, el cual se conecta a Heroku y despliega en las aplicaciones correspondientes las ramas master de los proyectos **frontend** y **backend**. El contenido del script es el siguiente:

```
@echo off
title Despliegue en Heroku

echo [*] Estableciendo conexión con Heroku
call heroku login

echo [*] Configurando backend
call heroku git:remote -a easyorienteering-backend
call git remote remove heroku-easyo-backend
```

```
call git remote rename heroku heroku-easyo-backend

echo [*] Configurando frontend
call heroku git:remote -a easyorienteering
call git remote remove heroku-easyo-frontend
call git remote rename heroku heroku-easyo-frontend

echo [*] Desplegando backend
call git subtree push --prefix=backend heroku-easyo-backend master

echo [*] Desplegando frontend
call git subtree push --prefix=frontend heroku-easyo-frontend master
```

B.5. Base de datos

Se utiliza la herramienta pgAdmin [24] para la administración sencilla de las bases de datos del entorno local y del de producción.

En Heroku se pueden programar copias de seguridad de las bases de datos de una aplicación. En el caso de este proyecto:

Programar creación de copia de seguridad a las 4 de la madrugada (UTC):

```
heroku pg:backups:schedule --at "04:00 UTC" -a easyorienteering-backend
```

Detener creación de copias de seguridad:

```
heroku pg:backups:unschedule -a easyorienteering-backend
```

Ver copias de seguridad:

```
heroku pg:backups -a easyorienteering-backend
```

Ver información detallada de una copia/restaurado:

```
heroku pg:backups:info ID -a easyorienteering-backend
```

Restaurar copia de seguridad:

```
heroku pg:backups:restore ID -a easyorienteering-backend
o
heroku pg:backups:restore "https://some.site/mydb.dump"
    -a easyorienteering-backend
```

Descargar copias de seguridad:

```
heroku pg:backups:download -a easyorienteering-backend
```

Borrar copia de seguridad:

```
heroku pg:backups:delete ID -a easyorienteering-backend
```

Bibliografía

[1] Federación Española de Orientación, *Historia de la FEDO*
https://www.fedo.org/web/orientacion/historia-de-la-fedo

[2] Federación Española de Orientación, *Servicio de Inscripciones*
https://sico.fedo.org

[3] Federación Española de Orientación, *Vídeos Deporte de Orientación a Pie - RTVE.es*
https://www.fedo.org/web/videos-o-pie/rtve

[4] Andrea Herrero Gómez, *Estados de ánimo y clima motivacional de los corredores de orientación de alto nivel*

[5] Juan Carlos Escaravajal Rodríguez, *Circuitos Permanentes de Orientación en España*
https://sites.google.com/view/circuitosorientacion

[6] iOrienteering, *iOrienteering*
https://www.iorienteering.com

[7] Joel Francia, *¿Qué es Scrum?*
https://www.scrum.org/resources/blog/que-es-scrum

[8] Proyectos Ágiles, *Qué es Scrum*
https://proyectosagiles.org/que-es-scrum

[9] Max Rehkopf, *Historias de usuario con ejemplos y plantilla*
https://www.atlassian.com/es/agile/project-management/user-stories

[10] Wikiversity, *Gestión de riesgos de proyectos software*
https://es.wikiversity.org/wiki/Gestión_de_riesgos_de_proyectos_software

[11] Gestion de Proyectos Software, *Tipos de riesgos*
https://sites.google.com/site/gestiondeproyectossoftware/unidad-3-planificacion-de-proyecto/3-5-1-tipos-de-riesgos

[12] GlassDoor, *Sueldo: Full Stack Developer*
https://www.glassdoor.es/Sueldos/full-stack-developer-sueldo-SRCH_KO0,20.htm

[13] Agencia Tributaria, *Tabla de coeficientes de amortización lineal*
https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml

[14] GitHub Inc., *GitHub*
https://github.com

[15] WakaTime, *WakaTime*
https://wakatime.com

[16] Astah*, *Astah**
https://astah.net

[17] Apache Software Foundation, *NetBeans*
https://netbeans.apache.org

[18] Balsamiq Studios, *Balsamiq Wireframes*
https://balsamiq.com/wireframes

[19] Overleaf, *Overleaf*
https://overleaf.com

[20] Apache Software Foundation, *Maven*
https://maven.apache.org

[21] Pivotal Software Inc., *Spring Framework*
https://spring.io/projects/spring-framework

[22] Red Hat, *Hibernate*
http://hibernate.org

[23] PostgreSQL Global Development Group, *PostgreSQL*
https://www.postgresql.org

[24] pgAdmin, *pgAdmin*
https://www.pgadmin.org

[25] Angular, *Angular*
https://angular.io

[26] Twitter Inc., *Bootstrap*
https://getbootstrap.com

[27] Node.js Foundation, *Node.js*
https://nodejs.org

[28] npm Inc, *npm*
https://www.npmjs.com

[29] Square Open Source, *Retrofit*
https://square.github.io/retrofit

[30] Gradle, *Gradle*
https://github.com/gradle/gradle

[31] Square Open Source, *OkHttp*
https://square.github.io/okhttp

[32] Android, *Biblioteca de persistencias Room*
https://developer.android.com/topic/libraries/architecture/room

[33] Salesforce, *Heroku*
https://www.heroku.com

[34] Baeldung, *Learn Spring Boot*
https://www.baeldung.com/spring-boot

[35] Square Inc., *OkHttp*
https://square.github.io/okhttp

[36] Square Inc., *Retrofit*
https://github.com/square/retrofit

[37] Mitch Tabian, *Local Database Cache with REST API, Retrofit2, MVVM Architecture*
https://github.com/mitchtabian/Local-db-Cache-Retrofit-REST-API-MVVM

[38] Federación Española de Orientación, *Especificación internacional para Mapas de Orientación*
https://www.fedo.org/web/ficheros/cartografia/normativa/ISOM-2017-castellano-MAY17.pdf

[39] ushelp, *EasyQRCodeJS*
https://github.com/ushelp/EasyQRCodeJS

[40] bpampuch, *pdfmake*
https://github.com/bpampuch/pdfmake

[41] Google LLC, *Mobile Vision*
https://developers.google.com/vision

[42] davemorrissey, *Subsampling Scale Image View*
https://github.com/davemorrissey/subsampling-scale-image-view

[43] orizens, *Angular Infinite Scroll*
https://github.com/orizens/ngx-infinite-scroll

[44] Leaflet, *Leaflet*
https://leafletjs.com

[45] Pivotal Software Inc., *Spring Security*
https://spring.io/projects/spring-security

[46] Instituto Nacional de Estadística, *Apellidos y nombres más frecuentes*
https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica$_C cid$ = 1254736177009$menu = resultadosidp$ = 1254734710990

[47] Business Intelligence fácil, *Longitud y latitud de los municipios de España*
https://www.businessintelligence.info/varios/longitud-latitud-pueblos-espana.html

[48] mdeanda, *Lorem Ipsum generator for Java*
https://github.com/mdeanda/lorem

[49] chimurai, *http-proxy-middleware*
https://github.com/chimurai/http-proxy-middleware

[50] ZetCode, *Spring Boot PostgreSQL tutorial*
http://zetcode.com/springboot/postgresql

[51] International Orienteering Federation, *Control Descriptions*
hthttps://orienteering.sport/iof/rules/control-descriptions

[52] Jason Watmore, *Angular 8 - Alert (Toaster) Notifications*
https://jasonwatmore.com/post/2019/07/05/angular-8-alert-toaster-notifications

[53] bampakoa, *Angular build fails when using html2canvas with await/async*
https://github.com/niklasvh/html2canvas/issues/1896