Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Análisis, Diseño e Implementación de una Aplicación Web Java para la Gestión de Carreras Deportivas

Estudiante: Diego Sánchez Cal
Dirección: José Losada Pérez

A Coruña, Junio de 2020

A mi familia y amigos.

Agradecimientos

En primer lugar, me gustaría agradecer a mis padres el apoyo durante estos años. Además, me gustaría agradecer al director de este trabajo, José Losada Pérez, por su tiempo dedicado y consejos para la creación y desarrollo de este proyecto.

Resumen

El objetivo de este proyecto consiste en el análisis, diseño e implementación de una aplicación web con arquitectura MVC (Modelo-Vista-Controlador) para la gestión de todo tipo de carreras populares de distintas distancias, desde las más pequeñas de tan solo una milla, hasta pruebas de fondo como por ejemplo maratones de 42 kilómetros.

La aplicación web soportará diferentes perfiles de usuario, permitiendo además el acceso a algunas funcionalidades a los usuarios sin registrar. El administrador será el encargado de gestionar las carreras y los usuarios.
Cada usuario registrado en la aplicación podrá gestionar la lista de carreras a las que asistirá o ya ha asistido así como realizar comentarios y emitir puntuaciones.
La aplicación tendrá un buscador que permitirá realizar búsquedas para ayudar a los usuarios a encontrar las carreras que coincidan con dicha búsqueda. Para cada carrera la aplicación mostrará información con sus detalles (nombre, fecha, distancia, hora de salida, etc.) y un mini-mapa con su ubicación utilizando el servicio externo de Google Maps.

La metodología empleada para el desarrollo de este proyecto será el Proceso Unificado de Desarrollo Software, que se basa en la descomposición del proyecto en pequeños subproyectos dentro de un esquema iterativo e incremental, en la que en cada iteración se realizan las fases de análisis, diseño, implementación y pruebas, de tal forma que cada iteración incorpora nuevas funcionalidades sobre la versión anterior hasta alcanzar el objetivo establecido.
Para el diseño e implementación de este proyecto, se harán uso de distintas tecnologías y frameworks enfocados a una arquitectura MVC.
Entre las más relevantes destacan la plataforma de programación JavaEE, el mapeador objeto-relacional Hibernate, el framework Spring, el motor de plantillas Thymeleaf, la librería jQuery y el framework Bootstrap.

Abstract

The objective of this project consists of the analysis, design and implementation of a web application with MVC (Model-View-Controller) architecture for the management of all kinds of popular races of different distances, from the smallest of just one mile, up to deep tests such as marathons of 42 kilometers.

The web application will support different user profiles, also allowing unregistered users to access some features. The administrator will be in charge of managing the races and the users.
Each user registered in the application will be able to manage the list of races they will attend or have already attended, as well as make comments and issue scores.
The application will have a search engine that will allow searches to be performed to help users find the races that match this search. For each race, the application will display information with its details (name, date, distance, start time, etc.) and a mini-map with its location using the external Google Maps service.

The methodology used for the development of this project will be the Unified Software Development Process, which is based on the decomposition of the project into small subprojects within an iterative and incremental scheme, in which the analysis phases are carried out in each iteration, design, implementation and testing in such a way that each iteration incorporates new functionalities over the previous version until it reaches the established objective.
For the design and implementation of this project, different technologies and frameworks focused on an MVC architecture will be used.
Among the most relevant are the JavaEE programming platform, the object-relational mapper Hibernate, the Spring framework, the Thymeleaf template engine, the jQuery library and the Bootstrap framework.

Palabras clave:

- Carreras Deportivas
- Aplicación Web
- Java
- HTML
- CSS
- MySQL
- Spring
- Thymeleaf
- Bootstrap

Keywords:

- Running Races
- Web Application
- Java
- HTML
- CSS
- MySQL
- Spring
- Thymeleaf
- Bootstrap

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

1.1 Motivación

En la actualidad ha crecido la popularidad de todo tipo de carreras populares de distintas distancias, desde las más pequeñas de tan solo una milla, hasta pruebas de fondo como por ejemplo maratones de 42 kilómetros. Existen multitud de alternativas web que permiten a los usuarios la consulta de información de las carreras, inscribirse a dichas carreras y posteriormente consultar sus resultados una vez participado.

Un ejemplo de estas webs más conocidas son:

- **Carreiras Galegas** (https://www.carreirasgalegas.com)
  Portal web del departamento de carreras populares oficiales de la *Federación Galega de Atletismo*. Ofrece el calendario e información de las carreras populares que se disputan en Galicia.

- **Champion Chip Norte** (https://www.championchipnorte.com)
  Empresa experta en eventos deportivos que colabora con todo tipo de organizadores (clubes, ayuntamientos, federaciones, etc.). Ofrecen servicio de gestión online para distintos deportes y cronometraje electrónico.

Este proyecto pretende crear una aplicación web en la que además de poder consultar información de las carreras e inscribirse, los usuarios puedan realizar comentarios y emitir puntuaciones sobre dichas carreras, obteniendo un **valor añadido** en la aplicación ofreciendo ”feedback” a otros usuarios interesados en esas carreras y permitiendo a la aplicación mostrar una lista con las carreras mejor valoradas, de utilidad para los usuarios/atletas que estén interesados en competir en futuras ediciones de esas carreras.

1.2 Alcance y objetivos

Este proyecto pretende crear un sistema en el que sus usuarios puedan obtener información sobre carreras deportivas en las que pueden participar, inscribirse en ellas y emitir comentarios y valoraciones que sirvan para otros usuarios. A continuación se expondrán las funcionalidades más relevantes:

- **Gestión de usuarios:** Operaciones relacionadas con el alta de un usuario en el sistema, modificación y eliminación. Existirán tres tipos de usuario: administrador, usuario sin registrar y usuario registrado.
- **Gestión de competiciones:** Operaciones relacionadas con las carreras, entre ellas se encuentran las operaciones típicas de un administrador como añadir, modificar y eliminar.
- **Gestión de comentarios:** La aplicación permitirá a los usuarios emitir comentarios sobre una carrera.
- **Gestión de valoraciones:** La aplicación permitirá a a los usuarios emitir una valoración de una carrera con un valor del uno al cinco. El sistema con todas las valoraciones de los usuarios calculará la media obteniendo una valoración de la carrera en el sistema. Con estos datos, el sistema mostrará una lista de las carreras mejor valoradas.
- **Gestión de inscripciones:** La aplicación permitirá a los usuarios registrarse en las carreras disponibles. Además los usuarios podrán ver el historial de carreras a las que se han inscrito.
- **Gestión de búsquedas:** La aplicación permitirá búsquedas sobre las carreras registradas en el sistema aplicando diferentes filtros.

1.3 Estructura de la memoria

La presente memoria se ha estructurado en 10 capítulos, 3 apéndices y el apartado de bibliografía. A continuación se explica brevemente su contenido:

- **Capítulo 1:** Capítulo en el que se introduce el trabajo, sus motivaciones y su alcance.
- **Capítulo 2:** Capítulo en el que se exponen y explican las principales herramientas y tecnologías utilizadas durante el desarrollo.
- **Capítulo 3:** Capítulo en el que se analiza la viabilidad del proyecto.
- **Capítulo 4:** Capítulo en el que se explica la arquitectura global del sistema y la metodología utilizada para el desarrollo.
- **Capítulo 5:** Capítulo en el que se realiza la planificación del proyecto estimando el coste de las tareas y simulando un proyecto real.
- **Capítulo 6:** Capítulo en el que se exponen todos los requisitos que debe de cumplir el sistema para ser aceptado.
- **Capítulo 7:** Capítulo en el que se expone el diseño que se ha realizado para la aplicación.
- **Capítulo 8:** Capítulo en el que se explica la implementación realizada utilizando diversas herramientas.
- **Capítulo 9:** Capítulo en el que se explican los tipos de pruebas realizadas.
- **Capítulo 10:** Capítulo en el que se desarrollan las conclusiones adquiridas al finalizar el proyecto y posibles futuras líneas de trabajo.
- **Anexo A:** API REST
- **Anexo B:** Lista de acrónimos
- **Anexo C:** Glosario
- **Bibliografía**

Capítulo 2

Tecnologías y herramientas utilizadas

En este capítulo de la memoria se explicarán, de forma resumida, las tecnologías y herramientas que se han utilizado para la realización de este proyecto.

2.1 Lenguajes de programación

2.1.1 Java y Java EE

Java [1] es un lenguaje de programación de alto nivel, basado en clases y orientado a objetos. Es un lenguaje pensado para entornos de producción. Incluye gestión automática de almacenamiento, generalmente usando un recolector de basura, para evitar los problemas de seguridad de la desasignación explícita (como ocurre en C o C++).
Los programadores utilizan el Java Development Kit (JDK) que es una herramienta de desarrollo para la creación de programas en Java. Este Kit viene con un entorno Java Runtime Environment (JRE), que contiene una Java Virtual Machine (JVM) y todas las librerías necesarias para desarrollar una aplicación Java.
En este proyecto se hizo uso de la plataforma de programación **Java Enterprise Edition** (Java EE), que consiste en un conjunto de especificaciones de Java Standard Edition (Java SE) con funcionalidades extras usadas en el ámbito empresarial como servicios web.
Las aplicaciones Java EE se ejecutan en servidores de aplicaciones, que manejan las transacciones, seguridad, escalabilidad, concurrrencia de los componentes que está implementando. Las APIs de Java EE incluyen varias tecnologías que amplían las funciones básicas de Java SE, algunas de ellas son:

- Java Naming and Directory Interface (JNDI)
- Java Database Connectivity (JDBC)

- Java Persistence API (JPA)
- Java API for RESTful Web Services (JAX-RS)

Figura 2.1: Logo de Java EE

2.1.2 HTML

HTML (HyperText Markup Language) [2], es el lenguaje principal y más importante de la World Wide Web (WWW). Es el estándar de la World Wide Consorcium (W3C), organización dedicada a la estandarización de todas las tecnologías ligadas a la web. Este lenguaje de marcado sirve para la visualización de páginas web, actualmente todos los navegadores lo implementan.

Figura 2.2: Logo de HTML

2.1.3 CSS

CSS (Cascading Style Sheets) [2], es un lenguaje de diseño gráfico que permite controlar el aspecto o presentación de documentos HTML o XML. Con CSS se define el aspecto de cada elemento, dándole estilo y diseño a las páginas web.
La especificación CSS es mantenida por el World Wide Web Consortium (W3C).

Figura 2.3: Logo de CSS

2.1.4 JavaScript

JavaScript (JS) [3], es un lenguaje de programación ligero e interpretado, orientado a objetos con funciones de primera clase. El propósito general del lenguaje es permitir incluir contenido ejecutable en las páginas web.
Es una de los lenguajes base de la Open Web y posee una especificación estandarizada por parte de la W3C.

Figura 2.4: Logo de JavaScript

2.1.5 SQL

SQL (Structured Query Languaje) [4], es un lenguaje de dominio específico diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.

Figura 2.5: Logo de SQL

2.1.6 LaTeX

LaTeX [5] es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Es usado con mayor frecuencia en la generación de artículos y libros científicos por sus características y posibilidades.

Figura 2.6: Logo de LaTeX

2.2 Frameworks y librerías

A continuación se detallan los frameworks y librerías utilizadas en el proyecto.

2.2.1 Hibernate

Hibernate [6] es un mapeador objeto-relacional (ORM) implementado para la plataforma Java. El término mapeador objeto-relacional se refiere a la técnica de mapeo de atributos entre el modelo de datos de una aplicación y la base de datos relacional, ayudándose de anotaciones en los beans de las entidades o ficheros declarativos XML que permiten establecer estas relaciones. La característica principal de Hibernate es el mapeo de clases Java a las tablas de la base de datos, y el mapeo de los tipos de datos Java a los tipos de datos SQL.

Figura 2.7: Logo de Hibernate

2.2.2 Spring

Spring [7] es un framework para el desarrollo de aplicaciones y contenedor de inversión de control (IoC), de código abierto para la plataforma Java.
Permite el desarrollo de aplicaciones Java centrándose en la propia aplicación, delegando en Spring el manejo de la infraestructura.
Spring está compuesto por distintas características organizadas en unos 20 módulos. Todos sus módulos están agrupados en 6 grandes grupos: Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation y Test.
Con el paso del tiempo Spring se volvió demasiado complejo, necesitando pasar por un largo proceso de requisitos para empezar un nuevo proyecto, para solventar esto surgió Spring Boot.

Spring Boot facilita la creación de aplicaciones basadas en Spring. Encargándose del proceso de la configuración, de modo que el usuario se pueda centrar en la lógica de negocio.

Algunas de sus ventajas son:

- Reduce el tiempo de desarrollo, aumentando la productividad.
- Evita especificar anotaciones y configuraciones XML.

- Facilidad a la hora de integrarse con ecosistemas de Spring, como por ejemplo Spring Security.
- Proporciona servidores HTTP, como Tomcat o Jetty.
- Proporciona CLI (Command Line Interface) para desarrollar o probar las aplicaciones desde un prompt.
- Incluye pluggins para trabajar con bases de datos embebidas o en memoria.

Figura 2.8: Logo de Spring

2.2.3 Thymeleaf

Thymeleaf [8] es un motor de plantillas Java capaz de procesar HTML, XML, JavaScript o CSS para mostar datos y/o texto producido por las aplicaciones. El objetivo principal de Thymeleaf es permitir la creación de plantillas de una manera elegante y un código bien formateado.

Figura 2.9: Logo de Thymeleaf

2.2.4 jQuery

jQuery [9] es una librería multiplataforma que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Figura 2.10: Logo de jQuery

2.2.5 Bootstrap

Bootstrap [10] es un framework de front-end para un desarrollo web más rápido y fácil. Incluye plantillas basadas en HTML y CSS para componentes comunes de la interfaz de usuario como tipografía, formularios, botones, barras de navegación, etc.

Figura 2.11: Logo de Bootstrap

2.3 Sistemas de Gestión de Bases de Datos

2.3.1 MySQL

MySQL [11] es un sistema de gestión de bases de datos relacional. Ofrece un servidor de base de datos SQL (Structured Query Languaje) muy rápido, multiproceso, multiusuario y robusto. Es considerada la base de datos de código abierto más popular del mundo, y una de las más importantes junto con las de Oracle o Microsoft SQL Server. MySQL es usado sobre todo en entornos de desarrollo web.

Figura 2.12: Logo de MySQL

2.4 APIs

Una **API** (Application Programming Interface) es un conjunto de subrutinas, funciones y métodos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción. Se trata de un conjunto de llamadas a ciertas librerías que ofrecen acceso a servicios desde los procesos.

2.4.1 API Google Maps

La **API** de **Google Maps** [12] es un servicio de mapas de Google que brinda diferentes funcionalidades tales como: marcadores, asignación de rutas, trazar sectores dentro de un mapa, entre otras.

Figura 2.13: Logo de Google Maps

2.5 Herramientas de desarrollo

A continuación se detallan las herramientas de desarrollo utilizadas.

2.5.1 Eclipse IDE

Eclipse [13] es un IDE (Integrated Development Environment). Esta diseñado para facilitar el trabajo de los programadores al desarrollar proyectos software.
Es un editor de código fuente, contiene un compilador y un intérprete, ofrece autocompletado inteligente del código, herramientas de construcción automáticas, etc.
Está escrito en los lenguajes de programación Java y C, y está disponible para los sistemas operativos Windows, Linux y Mac OS.

Figura 2.14: Logo de Eclipse

2.5.2 Sublime Text

Sublime Text [14] es un editor multiplataforma de código fuente y de texto que contiene multitud de características como por ejemplo el resaltado de sintaxis. En este proyecto, Sublime Text fue utilizado para el desarrollo de las plantillas en HTML.
Está disponible para los sistemas operativos Windows, Linux y Mac OS.

Figura 2.15: Logo de Sublime Text

2.5.3 Maven

Maven [15] es una herramienta software diseñada para gestionar y construir proyectos Java.
Se utiliza para crear y administrar cualquier proyecto basado en el lenguaje de programación Java, facilitando el trabajo inicial de los programadores. También consigue estandarizar la creación de los proyectos Java, permitiendo publicar de manera sencilla la información del proyecto y compartir los JARs (Java ARvhive) del mismo.

Figura 2.16: Logo de Maven

2.5.4 Git

Git [16] es un sistema de control de versiones de código abierto gratuito, diseñado para manejar todo tipo de proyectos, desde los más pequeños a los más grandes. Algunas de sus características son que permite ramificación local, área de preparación y múltiples flujos de trabajo.

Figura 2.17: Logo de Git

2.5.5 Overleaf

Overleaf [17] es un editor de LaTeX online, posee una gran variedad de plantillas disponibles para editar, además cuenta con control de versiones.

Figura 2.18: Logo de Overleaf

2.5.6 MagicDraw

MagicDraw [18] es una herramienta UML CASE compatible con el estándar UML 2.3, desarrollo de código para diversos lenguajes de programación (Java, C++, entre otros) así como para modelar datos.

Figura 2.19: Logo de MagicDraw

2.5.7 Gantt Project

Gantt Project [19] es un programa para la administración de proyectos usando el diagrama de Gantt. Está disponible para los sistemas operativos Windows, Linux y Mac OS.

Figura 2.20: Logo de Gantt Project

2.6 Servidores de Aplicaciones

2.6.1 Apache Tomcat

Apache Tomcat [20] o simplemente Tomcat es un contenedor web con soporte de servlets y JavaServer Pages (JSP). Tomcat funciona como servidor web por sí mismo y es usado en entornos con alto nivel de tráfico y alta disponibilidad. Tomcat viene embebido en Spring Boot y no es necesario disponer de él por separado.

Figura 2.21: Logo de Apache Tomcat

2.7 Otros

2.7.1 HTTP

HTTP (HyperText Transfer Protocol) [21], es un protocolo de la capa de aplicación del modelo OSI (Open System Interconnection) que sirve para transmitir documentos que tienen hipermedia, como HTML.

2.7.2 REST

REST (Representational State Transfer) [22], es un estilo de arquitectura Web utilizada para construir APIs de servicios Web modernos. Cuando una API Web cumple con las reglas de arquitectura definidas por REST se llama API REST.

2.7.3 XML

XML (eXtensible Markup Language) [23], es un lenguaje de marcado desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

Capítulo 3

Análisis de viabilidad

En este capítulo se analizará la viabilidad del proyecto, que es una de las primeras tareas que se debe de llevar a cabo antes de comenzar cualquier proyecto, para determinar si un proyecto puede llegar a finalizar con éxito. El análisis de viabilidad se ha dividido en tres apartados: viabilidad económica, viabilidad técnica y viabilidad de mercado.

3.1 Viabilidad económica

Para comenzar, se analizó el proyecto a nivel económico para saber si realmente es factible llevarlo a cabo o no. Para ello se ha realizado el análisis en base a los tres principales costes que tiene un proyecto de software: costes en los recursos humanos, costes en los recursos software y costes en los recursos hardware.

3.1.1 Recursos humanos

En cuanto a los costes de los recursos humanos, al ser en este caso, una persona la encargada del diseño, implementación y pruebas, el coste es bajo. En una empresa, en un entorno real, habría que tener en cuenta a todas las personas, con sus respectivos costes por hora de cada uno de ellos, que se necesitan para el desarrollo del proyecto.

3.1.2 Recursos software

En cuanto a los costes de los recursos software, el software que fue necesario para cumplir con los requisitos del proyecto no supuso un problema, ya que se utilizó en la medida de lo posible software con licencias de código abierto o software libre.
Al ser una aplicación web el objetivo de este proyecto, no existe ningún inconveniente para encontrar software adecuado para su realización, ya que existen multitud de opciones e información que se encuentran fácilmente en la web. Con estos dos factores se consigue que no haya un coste económico añadido en el proyecto.

3.1.3 Recursos hardware

En cuanto a los costes de los recursos hardware, para alcanzar los objetivos del proyecto, solamente es necesario un ordenador portátil de gama media para desarrollar la aplicación y una conexión a Internet para buscar y redactar documentación sobre las tecnologías usadas en el proyecto. Por tanto se puede decir que el proyecto es viable económicamente, ya que la inversión económica necesaria no es elevada.

3.2 Viabilidad técnica

Las tecnologías y herramientas utilizadas en el proyecto son altamente reconocidas y empleadas en numerosos proyectos software, por lo que suponen un riesgo bajo. Esto ayudará, en gran medida, a cumplir los objetivos propuestos al inicio del proyecto. Por estos motivos se puede decir que el proyecto es viable técnicamente.

3.3 Viabilidad de mercado

En la actualidad, existen varias opciones relacionadas con las carreras deportivas, por ejemplo CarreirasGalegas o ChampionChipNorte.
Carreiras Galegas (https://www.carreirasgalegas.com): Portal web del departamento de carreras populares oficiales de la *Federación Galega de Atletismo*. Esta web ofrece a los usuarios la posibilidad de ver el calendario de carreras disponibles, su información y la posibilidad de poder inscribirse.
Champion Chip Norte (https://www.championchipnorte.com): Empresa experta en eventos deportivos que ofrece a los usuarios la posibilidad de ver las carreras que hay disponibles, ver su información y la posibilidad de poder inscribirse.

Como se ha comentado anteriormente, en estos dos portales web los usuarios pueden ver el detalle e inscribirse en las carreras disponibles.
Pero no ofrecen la posibilidad de comentar y valorar una carrera a los usuarios que han participado en ellas, creando ”feedback” entre los usuarios, lo que supone un valor añadido.
Por este motivo, se puede afirmar que **el proyecto** tiene un hueco en el mercado actual y **es viable** su desarrollo.

Capítulo 4

Introducción al desarrollo realizado

En este capítulo se explicará la arquitectura global del sistema y la metodología empleada.

4.1 Aquitectura global del sistema

La arquitectura global del sistema ha sido desarrollada empleando principalmente el patrón de diseño **Modelo-Vista-Controlador** o **Model-View-Controller (MVC)**[24]. Esta arquitectura se emplea ampliamente en aplicaciones web.

Divide el sistema en tres partes diferentes que se encargan de la lógica de control de la interfaz y del acceso a datos.
Por un lado tenemos la **vista**, que se encarga de presentarle al usuario la información del sistema y responder a sus peticiones. Por otra parte tenemos al **controlador**, cuya principal función es la de procesar las peticiones realizadas por los usuarios al interactuar con el sistema. El controlador se encarga de las comunicaciones entre la vista y el modelo. Por último tenemos al **modelo** que es la parte del sistema que contiene la información que muestra la vista y la lógica de negocio.

Dentro de cada una de las partes de la arquitectura MVC se ha dividido la aplicación en más capas lógicas[25], en cada una de ellas se utilizan diferentes tecnologías y frameworks de programación.
La **vista** está formada por la capa de presentación y la capa de cliente.
El **controlador** está formado por la capa controladora.
El **modelo** está formado por la capa de acceso a datos y la capa de lógica de negocio.

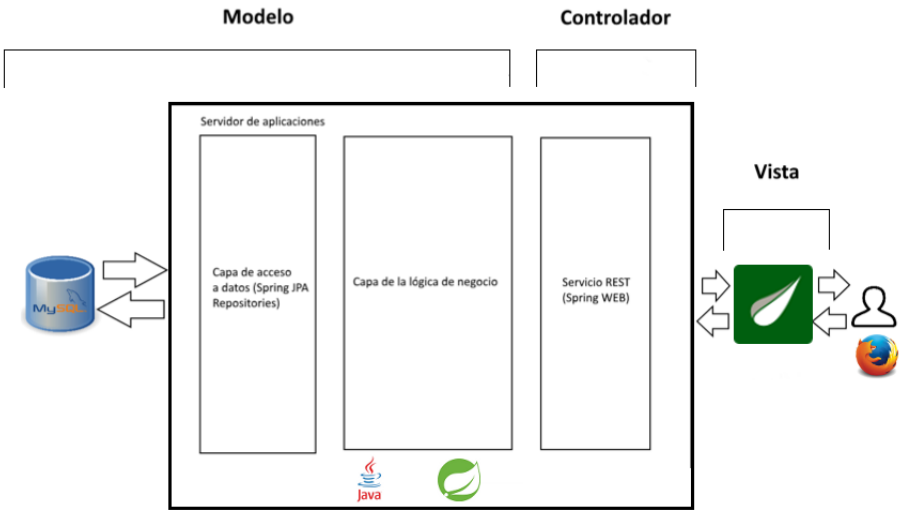


Figura 4.1: Esquema MVC

A continuación se explicará brevemente cada una de las partes.

- **Capa de cliente:** Está formada por la lógica de la aplicación que el usuario accede directamente a través de la interfaz de usuario. La lógica de esta capa incluye clientes basados en navegadores web.
- **Capa de presentación:** Está formada por la lógica de la aplicación, la cual prepara los datos para enviarlos a la capa cliente y procesa solicitudes desde la capa cliente para su envío a la lógica de negocios del servidor. En esta capa ha sido desarrollada usando HTML, CSS, Thymeleaf, JavaScript, jQuery y Bootstrap.
- **Capa del controlador:** Es una capa intermedia entre la capa de presentación y la capa de lógica de negocio, se encarga de recibir las peticiones HTTP de los usuarios finales. Se trata de un servicio REST implementado en Java con la ayuda de Spring.
- **Capa de lógica de negocio:** Es la capa en la que se implementa la lógica interna de los casos de uso de la aplicación. Esta capa ha sido desarrollada en Java apoyándose en Spring, se hace uso de los repositorios propios de la capa de acceso a datos.
- **Capa de acceso a datos:** Es la capa lógica de código que se encarga de acceder a la base de datos subyacente y recuperar los datos necesarios para la capa de lógica de negocio. Esta capa está programada en Java y se ha utilizado la tecnología Spring Data JPA (Java Persistence API), la cual permite mapear objetos Java a entidades de Base de Datos y realizar operaciones sobre ellos a través de la implementación de los repositorios.

4.2 Metodología utilizada

Para la creación y desarrollo de este proyecto software se ha basado en una versión simplificada de la metodología llamada Proceso Unificado de Desarrollo de Software (Rational Unified Process/RUP)[26].
Siguiendo un desarrollo basado en iteraciones, de manera que cada iteración incorpora más funcionalidades, que la iteración anterior, a la aplicación web hasta obtener el objetivo deseado.

4.2.1 Proceso Unificado de Desarrollo de Software

Esta metodología propone el desarrollo a través de subproyectos dentro de un esquema iterativo e incremental. Algunas de sus características son:

- **Iterativo e incremental:** Se compone por cuatro fases denominadas ”Inicio”, ”Elaboración”, ”Construcción” y ”Transición”. Cada una de ellas se dividen en más iteraciones, las cuales ofrecen como resultado un incremento del producto desarrollado que añade o mejora funcionalidades del sistema. Igual que en el desarrollo en cascada, tienen las fases de ”Análisis de Requisitos”, ”Diseño”, ”Implementación” y ”Pruebas”.

- **Dirigido por los casos de uso:** los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones.

- **Centrado en la arquitectura:** asume que no existe un modelo único que cubre todos los aspectos del sistema. Existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

- **Enfocado en los riesgos:** requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, deben de ser seleccionados en una orden en que los riesgos principales son considerados en primer lugar.

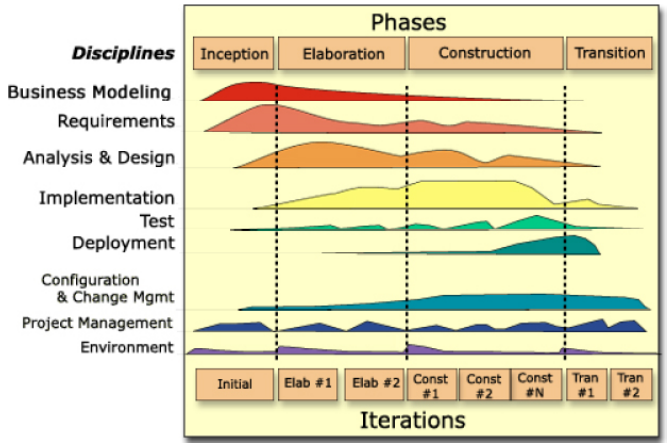


Figura 4.2: Fases del Proceso Unificado de Desarrollo del Software

Como podemos observar en la figura 4.2, el Proceso Unificado de Desarrollo de Software se divide en dos dimensiones o ejes:

- La **Dimension horizontal** representa la estructura dinámica o dimensión temporal del proceso. Muestra cómo se desarrolla el proceso, expresado en términos de ciclos, fases e iteraciones.
- La **Dimension vertical** representa la estructura estática del proceso. Describe cómo los elementos de proceso (actividades, roles) se agrupan lógicamente en flujos de trabajo.

El Proceso Unificado divide un ciclo en cuatro fases consecutivas: comienzo, elaboración, construcción y transición.

- **Fase de comienzo:** Durante esta fase se estable el alcance del proyecto, esto implica identificar las entidades externas con las que el sistema interactuará (actores) e identificar todos los casos de uso. El caso de negocio incluye la evaluación de riesgos y estimación de los recursos necesarios. En esta fase también es donde todas las partes se ponen de acuerdo para decidir si el proyecto se pone en marcha o no.
- **Fase de elaboración:** En esta fase se analiza el dominio del problema, se implementa iterativamente el núcleo de la aplicación, se establece una base arquitectónica sólida, se desarrolla el plan de proyecto y se eliminan los elementos de mayor riesgo del proyecto.
- **Fase de construcción:** En esta fase se realiza la mayor parte de la implementación para obtener una primera versión operativa del sistema.
- **Fase de transición:** El propósito de esta fase es hacer la transición del producto software a los usuarios. La aplicación debe de estar lista para ser probada, instalada y utilizada

por el cliente sin problema. Al final de esta fase, se comienza a pensar en futuras mejoras para la aplicación.
Una vez que la aplicación se entrega al usuario final, si se origina algún problema que necesite solucionarse, será necesario desarrollar nuevas versiones de la aplicación para corregir los problemas y alcance las funciones que se propusieron inicialmente.

4.2.2 Iteraciones

En cada iteración se hace el análisis, diseño, implementación y pruebas, de manera que cada iteración incorpora más funcionalidades que la anterior, hasta que en la última iteración se obtiene un software que implementa todas las funcionalidades de los casos de uso y cumple con todos los requisitos iniciales. Empleando la metodología descrita anteriormente se ha dividido el proyecto en 11 iteraciones.

- **Iteración 1: Puesta en marcha del proyecto**
  En esta iteración se ha llevado a cabo el análisis de viabilidad y la definición del dominio del proyecto. Una vez valorado que es factible, se ha llevado a cabo un estudio sobre las tecnologías disponibles para determinar cuáles se iban a utilizar. Durante esta iteración también se han definido los requisitos, especificado los casos de uso y que tipos de usuarios podrán utilizar cada uno de ellos, y las entidades persistentes necesarias. En esta iteración se ha preparado el entorno de desarrollo con todas las herramientas necesarias. Se crearon los ficheros de configuración, por ejemplo el ”pom.xml” de Maven para gestionar todo el proyecto y los ficheros de configuración de Hibernate y Spring.

- **Iteración 2: Gestión de usuarios**
  En esta iteración se llevó a cabo la gestión de usuarios y su acceso a la aplicación (dependiendo si es ”usuario sin registrar”, ”usuario registrado” o ”administrador”). Se han creador las entidades persistentes y todas las operaciones necesarias sobre ellas. Se han realizado las fases de análisis, diseño implementación y pruebas sobre una primera versión que permite el registro y acceso de los usuarios.

- **Iteración 3: Gestión de carreras**
  En esta iteración se llevó a cabo la gestión de las carreras. Se obtuvo una nueva versión en la que los usuarios sin registrar pueden ver la lista de carreras, los usuarios registrados el detalle de cada carrera y el administrador puede realizar operaciones sobre las carreras: añadir, modificar, eliminar.

- **Iteración 4: Perfil de usuario**
  En esta iteración se ha desarrollado el perfil de los usuarios registrados, donde pueden ver y modificar sus datos personales.

- **Iteración 5: Gestión de comentarios**
  En esta iteración se ha implementado la funcionalidad que permite comentar una carrera. Como en las iteraciones anteriores se han realizado las cuatro fases del proceso de desarrollo.

- **Iteración 6: Gestión de valoraciones**
  En esta iteración se ha implementado la funcionalidad que permite puntuar una carrera y visualizar la puntuación media de dicha carrera. Como en las iteraciones anteriores se han realizado las cuatro fases del proceso de desarrollo para llevarla a cabo.

- **Iteración 7: Carreras mejor valoradas y buscador**
  En esta iteración se ha implementado la funcionalidad que permite mostrar la lista de las carreras mejor valoradas. Además se han implementado un filtro para refinar todavía más la búsqueda.

- **Iteración 8: Integración API Google Maps**
  Se implementó la integración con la API de Google Maps, mostrando en el detalle de cada carrera un mini-mapa con su ubicación.

- **Iteración 9: Gestión de inscripciones**
  Se implementó la funcionalidad de que los usuarios puedan inscribirse en una carrera. En caso de que la carrera no sea gratuita, los usuarios deberán de introducir datos adicionales en una pasarela de pago, en este caso por simplificar el proyecto, una pasarela de pago ficticia.

- **Iteración 10: Internacionalización (i18n)**
  Se modificaron las plantillas y se crearon los ficheros ".properties" con las traducciones para poder soportar la internacionalización en dos idiomas: español e inglés. Los usuarios pueden elegir idioma en la pantalla principal.

- **Iteración 11: Elaboración de la memoria**
  En esta última iteración se ha recopilado en una memoria final todo el trabajo del proyecto realizado. Se ha utilizado LaTeX debido a su calidad para crear documentos académicos.

Capítulo 5

Planificación y análisis de costes

Para la planificación del proyecto software se deben de ter en cuenta dos factores, que son el tiempo y el coste. Estos dos parámetros son muy importantes para cualquier proyecto y deben de estar presentes desde el principio hasta el final. Minimizando ambos se consigue un resultado óptimo y predecible.

5.1 Recursos

Para la realización de este proyecto han sido necesarios dos tipos de recursos: humanos y técnicos.

5.1.1 Recursos humanos

Para la realización de este proyecto se han simulado varios perfiles de recursos humanos. Sin embargo, detrás de todos estos perfiles se encuentran el autor de este proyecto en colaboración con el director del proyecto. Los perfiles son:

- **Jefe de proyecto:** es el encargado de coordenar y gestionar todos los recursos necesarios para desarrollar el proyecto para que este se desarrolle correctamente, desde el inicio hasta la versión final de lanzamiento al público. También se encarga de definir los requisitos del proyecto, calcular el coste y realizar la planificación y el seguimiento.

- **Analista:** es el encargado de capturar los requisitos establecidos por el jefe de proyecto, además de realizar el análisis técnico y funcional. También se encarga de vigilar que se cumplan los requisitos del cliente y resolver cualquier duda que tengan los programadores.

- **Programador:** es el encargado de implementar y desarrollar la aplicación. También se encarga de realizar las pruebas de la aplicación.

5.1.2 Recursos técnicos

Los recursos técnicos necesarios fueron prácticamente todos productos de licencia abierta o software libre. El ordenador portátil utilizado para desarrollar todas las tareas, desde la aplicación hasta la elaboración de la memoria, ha sido:

- **Marca y modelo:** Apple MacBook Pro
- **Sistema operativo:** OS X Versión 10.8.5
- **Procesador:** 2.9 GHz Intel Core i7
- **Memoria RAM:** 8 GB 1600 MHz DDR3

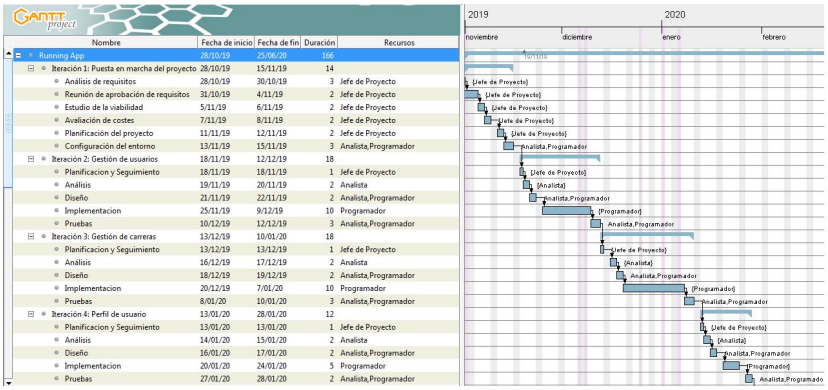
5.2 Planificacion y costes

El proyecto ha sido realizado siguiendo las pautas del Proceso Unificado de Desarrollo Software. Las iteraciones definidas en el capítulo anterior son las que se llevaron a cabo, planificando los recursos, el coste y el tiempo para cada una de ellas. El coste de cada recurso que se utilizará para el coste total del proyecto es el siguiente:

Recurso	Coste
Jefe de proyecto	30 €/h
Analista	20 €/h
Programador	15 €/h
Ordendador portátil	1.500 €

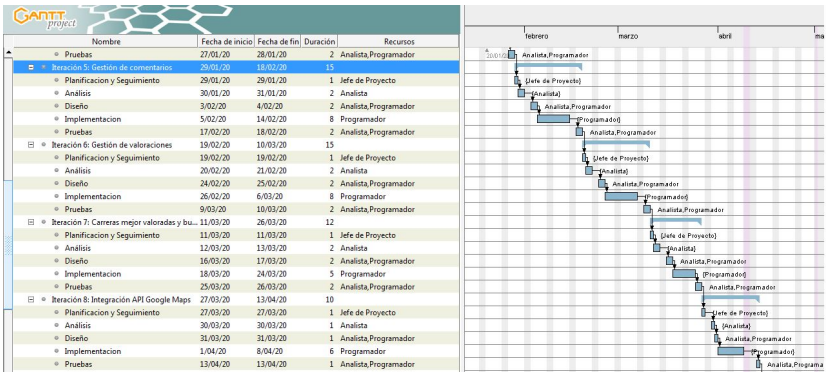
Tabla 5.1: Costes de los recursos del proyecto

Una vez descritos el coste de los recursos y las iteraciones en el capítulo anterior, se detalla el tiempo que llevará realizar cada tarea. Para ello se utiliza un Diagrama de Gantt como herramienta para ilustrar de forma más gráfica la duración total del proyecto con todas las tareas a realizar.
A continuación, en las capturas siguientes se muestran los detalles de cada tarea: nombre, fecha de inicio, fecha de fin, duración y los recursos involucrados en la realización de cada tarea.



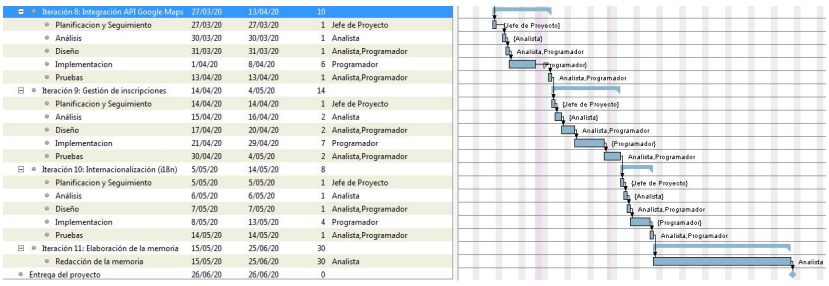
Nombre	Fecha de inicio	Fecha de fin	Duración	Recursos
Running App	28/10/19	25/06/20	166	
Iteración 1: Puesta en marcha del proyecto	28/10/19	15/11/19	14	
Análisis de requisitos	28/10/19	30/10/19	3	Jefe de Proyecto
Reunión de aprobación de requisitos	31/10/19	4/11/19	2	Jefe de Proyecto
Estudio de la viabilidad	5/11/19	6/11/19	2	Jefe de Proyecto
Avaliación de costes	7/11/19	8/11/19	2	Jefe de Proyecto
Planificación del proyecto	11/11/19	12/11/19	2	Jefe de Proyecto
Configuración del entorno	13/11/19	15/11/19	3	Analista,Programador
Iteración 2: Gestión de usuarios	18/11/19	12/12/19	18	
Planificacion y Seguimiento	18/11/19	18/11/19	1	Jefe de Proyecto
Análisis	19/11/19	20/11/19	2	Analista
Diseño	21/11/19	22/11/19	2	Analista,Programador
Implementacion	25/11/19	9/12/19	10	Programador
Pruebas	10/12/19	12/12/19	3	Analista,Programador
Iteración 3: Gestión de carreras	13/12/19	10/01/20	18	
Planificacion y Seguimiento	13/12/19	13/12/19	1	Jefe de Proyecto
Análisis	16/12/19	17/12/19	2	Analista
Diseño	18/12/19	19/12/19	2	Analista,Programador
Implementacion	20/12/19	7/01/20	10	Programador
Pruebas	8/01/20	10/01/20	3	Analista,Programador
Iteración 4: Perfil de usuario	13/01/20	28/01/20	12	
Planificacion y Seguimiento	13/01/20	13/01/20	1	Jefe de Proyecto
Análisis	14/01/20	15/01/20	2	Analista
Diseño	16/01/20	17/01/20	2	Analista,Programador
Implementacion	20/01/20	24/01/20	5	Programador
Pruebas	27/01/20	28/01/20	2	Analista,Programador

Figura 5.1: Diagrama de Gantt - Iteraciones detalladas (Parte 1)



Nombre	Fecha de inicio	Fecha de fin	Duración	Recursos
Pruebas	27/01/20	28/01/20	2	Analista,Programador
Iteración 5: Gestión de comentarios	29/01/20	18/02/20	15	
Planificacion y Seguimiento	29/01/20	29/01/20	1	Jefe de Proyecto
Análisis	30/01/20	31/01/20	2	Analista
Diseño	3/02/20	4/02/20	2	Analista,Programador
Implementacion	5/02/20	14/02/20	8	Programador
Pruebas	17/02/20	18/02/20	2	Analista,Programador
Iteración 6: Gestión de valoraciones	19/02/20	10/03/20	15	
Planificacion y Seguimiento	19/02/20	19/02/20	1	Jefe de Proyecto
Análisis	20/02/20	21/02/20	2	Analista
Diseño	24/02/20	25/02/20	2	Analista,Programador
Implementacion	26/02/20	6/03/20	8	Programador
Pruebas	9/03/20	10/03/20	2	Analista,Programador
Iteración 7: Carreras mejor valoradas y bu...	11/03/20	26/03/20	12	
Planificacion y Seguimiento	11/03/20	11/03/20	1	Jefe de Proyecto
Análisis	12/03/20	13/03/20	2	Analista
Diseño	16/03/20	17/03/20	2	Analista,Programador
Implementacion	18/03/20	24/03/20	5	Programador
Pruebas	25/03/20	26/03/20	2	Analista,Programador
Iteración 8: Integración API Google Maps	27/03/20	13/04/20	10	
Planificacion y Seguimiento	27/03/20	27/03/20	1	Jefe de Proyecto
Análisis	30/03/20	30/03/20	1	Analista
Diseño	31/03/20	31/03/20	1	Analista,Programador
Implementacion	1/04/20	8/04/20	6	Programador
Pruebas	13/04/20	13/04/20	1	Analista,Programador

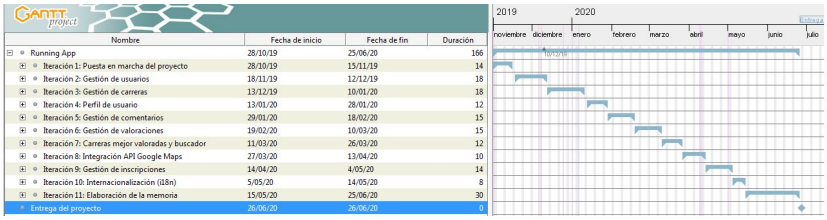
Figura 5.2: Diagrama de Gantt - Iteraciones detalladas (Parte 2)



Iteración 8: Integración API Google Maps	27/03/20	13/04/20	10	
Planificacion y Seguimiento	27/03/20	27/03/20	1	Jefe de Proyecto
Análisis	30/03/20	30/03/20	1	Analista
Diseño	31/03/20	31/03/20	1	Analista,Programador
Implementacion	1/04/20	8/04/20	6	Programador
Pruebas	13/04/20	13/04/20	1	Analista,Programador
Iteración 9: Gestión de inscripciones	14/04/20	4/05/20	14	
Planificacion y Seguimiento	14/04/20	14/04/20	1	Jefe de Proyecto
Análisis	15/04/20	16/04/20	2	Analista
Diseño	17/04/20	20/04/20	2	Analista,Programador
Implementacion	21/04/20	29/04/20	7	Programador
Pruebas	30/04/20	4/05/20	2	Analista,Programador
Iteración 10: Internacionalización (i18n)	5/05/20	14/05/20	8	
Planificacion y Seguimiento	5/05/20	5/05/20	1	Jefe de Proyecto
Análisis	6/05/20	6/05/20	1	Analista
Diseño	7/05/20	7/05/20	1	Analista,Programador
Implementacion	8/05/20	13/05/20	4	Programador
Pruebas	14/05/20	14/05/20	1	Analista,Programador
Iteración 11: Elaboración de la memoria	15/05/20	25/06/20	30	
Redacción de la memoria	15/05/20	25/06/20	30	Analista
Entrega del proyecto	26/06/20	26/06/20	0	

Figura 5.3: Diagrama de Gantt - Iteraciones detalladas (Parte 3)

Una vez visto el detalle de las tareas, se puede ver que el proyecto tiene como fecha de inicio el día **28 de Octubre de 2019**, y como fecha de fin el día **26 de Junio de 2020**.
Esto da un total de **166 días trabajados**, donde se dedicaban aproximadamente 6 horas de trabajo por día, haciendo un total de **1218 horas** entre todos los recursos.
A continuación se muestra el resumen por cada iteración con su fecha de inicio, fecha de fin y su duración.



Nombre	Fecha de inicio	Fecha de fin	Duración
Running App	28/10/19	25/06/20	166
Iteración 1: Puesta en marcha del proyecto	28/10/19	15/11/19	14
Iteración 2: Gestión de usuarios	18/11/19	12/12/19	18
Iteración 3: Gestión de carreras	13/12/19	10/01/20	18
Iteración 4: Perfil de usuario	13/01/20	28/01/20	12
Iteración 5: Gestión de comentarios	29/01/20	18/02/20	15
Iteración 6: Gestión de valoraciones	19/02/20	10/03/20	15
Iteración 7: Carreras mejor valoradas y buscador	11/03/20	26/03/20	12
Iteración 8: Integración API Google Maps	27/03/20	13/04/20	10
Iteración 9: Gestión de inscripciones	14/04/20	4/05/20	14
Iteración 10: Internacionalización (i18n)	5/05/20	14/05/20	8
Iteración 11: Elaboración de la memoria	15/05/20	25/06/20	30
Entrega del proyecto	26/06/20	26/06/20	0

Figura 5.4: Diagrama de Gantt - Todas las iteraciones

A continuación se muestra el cálculo de los costes de todo el proyecto, para ello hay que tener en cuenta los recursos técnicos (hardware y software) y los recursos humanos.

- **Recursos técnicos:** Los costes en recursos técnicos fueron los siguientes.
  - **Hardware:** sólo se contabiliza el coste de la compra del ordenador portátil, por valor de 1.500 €.
  - **Software:** gracias al uso de herramientas y tecnologías de software libre, el coste es de 0 €.
- **Recursos humanos:** La estimación de los costes en recursos humanos se calcula teniendo en cuenta el tiempo empleado por cada recurso en el proyecto. En total el proyecto ha supuesto un total de 1218 horas de trabajo, las cuales se reparten de la siguiente forma.
  - **Jefe de proyecto:** realizó 120 horas, con un coste estimado de 30 €/h resulta un total de 3.600 €.
  - **Analista:** realizó 498 horas, con un coste estimado de 20 €/h resulta un total de 9.960 €.

- **Programador:** realizó 600 horas, con un coste estimado de 15 €/h resulta un total de 9.000 €.

En la siguiente tabla se muestra el coste total estimado del proyecto:

Concepto	**Coste**
Recursos técnicos - Hardware	1.500 €
Recursos técnicos - Software	0 €
Recursos humanos - Jefe de Proyecto	3.600 €
Recursos humanos - Analista	9.960 €
Recursos humanos - Programador	9.000 €
Subtotal	**24.060 €**
IVA (21 %)	5.052,60 €
Total	**29.112,60 €**

Tabla 5.2: Coste total estimado del proyecto

Capítulo 6

Requisitos del sistema

En este capítulo se expondrán los requisitos que debe de cumplir el sistema final. Se definirán los actores que interactuan con la aplicación y se detallarán los casos de uso.

6.1 Especificación de requisitos

El principal objetivo de la especificación de requisitos software es el de exponer de forma clara las necesidades de los usuarios finales del sistema.

6.1.1 Requisitos funcionales

Los requisitos funcionales son los que definen las funciones que realizará el sistema.

6.1.2 Requisitos no funcionales

Los requisitos no funcionales no se refieren directamente a las funciones específicas del sistema, si no a otras propiedades. En este proyecto destacan los siguientes requisitos no funcionales:

- **Idioma:** La aplicación soporta internacionalización (i18n) en dos idiomas, español e inglés.
- **Consistencia:** El modelo de la aplicación se mantiene actualizado en todo momento, sin perder ningún dato.
- **Fiabilidad:** Gestiona de manera controlada los fallos que se puedan producir.
- **Seguridad:** Autenticación de usuarios, almacenando los contraseñales encriptados en la base de datos. Control de acceso a las diferentes URL de la aplicación en función del tipo de usuario (usuario sin registrar, usuario registrado o administrador) mediante roles.

- **Simplicidad:** El sistema es simple e intuitivo para el usuario final.
- **Usabilidad:** El sistema es amigable para el usuario final.

6.2 Actores

En este proyecto existen tres tipos de usuarios, usuarios sin registrar, usuarios registrado y el administrador.

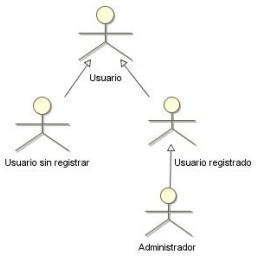


Figura 6.1: Diagrama de Actores

- **Usuario sin registrar:**
  El usuario sin registrar es aquel que accede a la aplicación web, pero no está autenticado. Puede usar funciones básicas de la aplicación (ver lista de carreras, buscar por palabras clave y aplicar filtros). No puede ver el detalle de las carreras, comentar, valorar ni inscribirse en ellas.
  Este usuario puede darse de alta en la aplicación web, convirtiéndose así en un usuario registrado.
- **Usuario registrado:**
  El usuario registrado es aquel que está autenticado en la aplicación web. Puede usar las mismas funcionalidades que el usuario sin registrar, y además puede ver el detalle de cada carrera, inscribirse en ella, comentarla o valorarla.
  Además tiene un perfil propio en donde puede consultar y modificar su información.
- **Usuario administrador:**
  El administrador puede acceder a todas las funcionalidades comentadas anteriormente, además tiene acceso a funciones propias para gestionar las carreras como por ejemplo crear, modificar y eliminar carreras.

6.3 Casos de uso

Los casos de uso se crean para refinar un conjunto de requisitos de acuerdo a una tarea concreta. Un caso de uso representa una lista de tareas que pueden realizar los actores y está directamente relacionado con los requisitos del proceso empresarial.

6.3.1 Casos de uso de usuarios

CU-101: Registrarse	
Descripción	Registro de un nuevo usuario en el sistema.
Actores	Usuario sin registrar.
Precondiciones	El usuario no debe de existir en el sistema.
Flujo	1. El usuario solicita registrarse en el sistema. 2. El sistema muestra los datos que debe de rellenar el usuario. 3. El usuario cubre todos los datos correctamente. 4. El sistema valida que todos los datos son correctos y los almacena en la base de datos.
Flujo Alternativo	1. Se muestra un error si el usuario introduce algún dato de manera incorrecta.
Postcondiciones	Se crea el nuevo usuario en el sistema.

Tabla 6.1: CU-101: Registrarse

CU-102: Iniciar Sesión	
Descripción	El usuario inicia sesión en la aplicación.
Actores	Usuario sin registrar.
Precondiciones	El usuario no está autenticado en la aplicación.
Flujo	1. El usuario solicita iniciar sesión en el sistema. 2. El sistema muestra un formulario con los campos que debe de completar el usuario. 3. El usuario cubre los campos correctamente. 4. El sistema valida que todos los campos son correctos y redirige al usuario a la página principal de la aplicación.
Flujo Alternativo	1. Se muestra un error si el usuario introduce algún dato de manera incorrecta.
Postcondiciones	El usuario inicia sesión en la aplicación.

Tabla 6.2: CU-102: Iniciar Sesión

CU-103: Cerrar Sesión	
Descripción	El usuario cierra su sesión en la aplicación.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario está autenticado en la aplicación.
Flujo	1. El usuario solicita cerrar sesión en el sistema. 2. El sistema cierra la sesión del usuario.
Flujo Alternativo	-
Postcondiciones	Se cierra la sesión del usuario en la aplicación.

Tabla 6.3: CU-103: Cerrar Sesión

CU-104: Ver Datos Usuario	
Descripción	La aplicación permite a un usuario ver sus datos.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado.
Flujo	1. El usuario solicita visualizar los datos de su perfil. 2. El sistema redirige al usuario a la página principal de su perfil. 3. El usuario selecciona ”Ver Mis Datos de Usuario”. 4. El sistema abre una nueva ventana y muestra los datos del usuario.
Flujo Alternativo	-
Postcondiciones	Se muestran los datos del usuario.

Tabla 6.4: CU-104: Ver Datos Usuario

CU-105: Modificar Datos Usuario	
Descripción	La aplicación permite a un usuario modificar sus datos.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado.
Flujo	1. El usuario solicita acceder a sus datos en el sistema. 2. El sistema muestra la página principal de su perfil con las acciones disponibles. 3. El usuario selecciona ”Ver Mis Datos de Usuario”. 4. El sistema abre una nueva ventana y muestra los datos del usuario. 5. El usuario solicita la acción ”Editar Mis Datos”. 6. El sistema muestra un formulario con los datos que se pueden modificar. 7. El usuario introduce correctamente los datos. 8. El sistema valida los datos y guarda los nuevos datos.
Flujo Alternativo	1. Se muestra un error si el usuario introduce de manera incorrecta algún dato.
Postcondiciones	Se modifican los datos del usuario.

Tabla 6.5: CU-105: Modificar Datos Usuario

CU-106: Eliminar Cuenta Usuario	
Descripción	La aplicación permite a un usuario eliminar su cuenta.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado.
Flujo	1. El usuario solicita acceder a sus datos en el sistema. 2. El sistema muestra el perfil usuario con las acciones disponibles. 3. El usuario selecciona ”Ver Mis Datos de Usuario”. 4. El sistema muestra los datos del usuario. 5. El usuario solicita eliminar su cuenta. 6. El sistema cierra la sesión del usuario y elimina su cuenta.
Flujo Alternativo	-
Postcondiciones	Se elimina la cuenta del usario.

Tabla 6.6: CU-106: Eliminar Cuenta Usuario

6.3.2 Casos de uso de carreras

CU-201: Mostrar Carreras Mejor Valoradas	
Descripción	La aplicación muestra las carreras mejor valoradas.
Actores	Usuario sin registrar, Usuario registrado, Administrador
Precondiciones	-
Flujo	1. El usuario solicita ver las carreras mejor valoradas. 2. El sistema muestra las carreras mejor valoradas ordenadas de forma descendente según su nota media.
Flujo Alternativo	-
Postcondiciones	La aplicación muestra las carreras mejor valoradas.

Tabla 6.7: CU-201: Mostrar Carreras Mejor Valoradas

CU-202: Mostrar Lista de Todas las Carreras	
Descripción	La aplicación muestra la lista de todas las carreras disponibles.
Actores	Usuario sin registrar, Usuario registrado, Administrador
Precondiciones	-
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras.
Flujo Alternativo	-
Postcondiciones	La aplicación muestra la lista de todas las carreras.

Tabla 6.8: CU-202: Mostrar lista de Todas las Carreras

CU-203: Buscar Carrera	
Descripción	La aplicación permite buscar una carrera.
Actores	Usuario sin registrar, Usuario registrado, Administrador
Precondiciones	-
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario introduce alguna palabra clave en el cuadro de búsqueda de la parte superior. 4. El sistema filtra la lista de carreras mostrando sólo las que tienen coincidencia en algún campo con la palabra clave.
Flujo Alternativo	-
Postcondiciones	La aplicación muestra la lista de carreras que coinciden con la búsqueda.

Tabla 6.9: CU-203: Buscar Carrera

CU-204: Filtrar Carrera	
Descripción	La aplicación permite buscar una carrera.
Actores	Usuario sin registrar, Usuario registrado, Administrador
Precondiciones	-
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona filtrar por el ”Tipo de carrera”. 4. El sistema filtra la lista de carreras mostrando sólo las que tienen coincidencia con el tipo seleccionado.
Flujo Alternativo	-
Postcondiciones	La aplicación muestra la lista de carreras que coinciden con la búsqueda.

Tabla 6.10: CU-204: Filtrar Carrera

CU-205: Ver Detalle de una Carrera	
Descripción	La aplicación permite ver el detalle de una carrera.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras 3. El usuario selecciona ”Ver Detalles”. 4. El sistema muestra toda la información relativa a esa carrera.
Flujo Alternativo	-
Postcondiciones	La aplicación muestra los detalles de la carrera seleccionada.

Tabla 6.11: CU-205: Ver Detalle de una Carrera

CU-206: Añadir Carrera	
Descripción	La aplicación permite añadir una carrera.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de carreras. 3. El administrador selecciona ”Añadir Carrera”. 4. El sistema muestra los datos que debe de completar el administrador. 5. El administrador completa los datos correctamente. 6. El sistema valida los datos y guarda la nueva carrera.
Flujo Alternativo	1. El sistema muestra un error si el administrador introduce algún dato de manera incorrecta.
Postcondiciones	Se crea la nueva carrera en el sistema.

Tabla 6.12: CU-206: Añadir Carrera

CU-207: Modificar Carrera	
Descripción	La aplicación permite modificar los datos de una carrera.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El administrador selecciona ”Editar Carrera”. 4. El sistema muestra los datos que debe de completar el administrador. 5. El administrador completa los datos correctamente. 6. El sistema valida los datos y guarda la carrera con los nuevos datos.
Flujo Alternativo	1. El sistema muestra un error si el administrador introduce algún dato de manera incorrecta.
Postcondiciones	Se modifica la carrera en el sistema.

Tabla 6.13: CU-207: Modificar Carrera

CU-208: Eliminar Carrera	
Descripción	La aplicación permite eliminar una carrera.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El administrador selecciona ”Eliminar Carrera”. 4. El sistema elimina la carrera.
Flujo Alternativo	-
Postcondiciones	Se elimina la carrera del sistema.

Tabla 6.14: CU-208: Eliminar Carrera

CU-209: Añadir Prueba	
Descripción	La aplicación permite añadir una prueba de una carrera.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El administrador selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera deseada. 5. El administrador selecciona ”Añadir Prueba”. 6. El sistema muestra en una nueva ventana los datos a completar por el administrador. 7. El administrador completa los datos correctamente. 8. El sistema valida los datos y guarda la nueva prueba.
Flujo Alternativo	1. El sistema muestra un error si el administrador introduce algún dato de manera incorrecta.
Postcondiciones	Se añade la nueva prueba al sistema.

Tabla 6.15: CU-209: Añadir Prueba

CU-210: Modificar Prueba	
Descripción	La aplicación permite modificar una prueba de una carrera.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El administrador selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera deseada con la lista de pruebas. 5. El administrador selecciona ”Editar” al lado de la prueba deseada. 6. El sistema muestra los datos a completar por el administrador. 7. El administrador completa los datos correctamente. 8. El sistema valida los datos y modifica la prueba con los nuevos datos.
Flujo Alternativo	1. El sistema muestra un error si el administrador introduce algún dato de manera incorrecta.
Postcondiciones	Se modifica la prueba en el sistema.

Tabla 6.16: CU-210: Modificar Prueba

CU-211: Eliminar Prueba	
Descripción	La aplicación permite eliminar una prueba de una carrera.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El administrador selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera deseada. 5. El administrador en la lista de pruebas selecciona ”Eliminar” al lado de la prueba deseada a eliminar. 6. El sistema elimina la prueba.
Flujo Alternativo	-
Postcondiciones	Se elimina la prueba del sistema.

Tabla 6.17: CU-211: Eliminar Prueba

CU-212: Ver Mapa Carrera	
Descripción	La aplicación permite ver el mapa de una carrera.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera. 5. El usuario selecciona ”Ver Mapa Completo”. 6. El sistema muestra el mapa de la carrera.
Flujo Alternativo	-
Postcondiciones	Se muestra el mapa de una carrera.

Tabla 6.18: CU-212: Ver Mapa Carrera

6.3.3 Casos de uso de comentarios

CU-301: Ver Comentarios de una Carrera	
Descripción	La aplicación permite ver los comentarios de una carrera.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera. 5. El usuario selecciona ”Ver Comentarios”. 6. El sistema muestra los comentarios de esa carrera.
Flujo Alternativo	-
Postcondiciones	Se muestran los comentarios de una carrera.

Tabla 6.19: CU-301: Ver Comentarios de una Carrera

CU-302 Añadir Comentario	
Descripción	La aplicación permite añadir un comentario a una carrera.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de carreras que hay disponibles. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera. 5. El usuario selecciona ”Ver Comentarios” 6. El sistema muestra los comentarios de una carrera. 7. El usuario introduce en el cuadro de texto su comentario. 8. El sistema guarda y muestra el nuevo comentario.
Flujo Alternativo	-
Postcondiciones	Se añade el nuevo comentario al sistema.

Tabla 6.20: CU-302: Añadir Comentario

CU-303: Ver Comentarios Usuario	
Descripción	La aplicación permite ver los comentarios de un usuario.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado
Flujo	1. El usuario solicita ver su información en el sistema. 2. El sistema muestra el perfil del usuario con las acciones disponibles. 3. El usuario selecciona ”Ver Mis Comentarios”. 4. El sistema muestra los comentarios del usuario.
Flujo Alternativo	-
Postcondiciones	Se muestran los comentarios de un usuario.

Tabla 6.21: CU-303: Ver Comentarios Usuario

6.3.4 Casos de uso de valoraciones

CU-401: Ver Valoraciones de una Carrera	
Descripción	La aplicación permite ver las valoraciones de una carrera.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera. 5. El usuario selecciona ”Ver Valoraciones”. 6. El sistema muestra las valoraciones de esa carrera.
Flujo Alternativo	-
Postcondiciones	Se muestran las valoraciones de una carrera.

Tabla 6.22: CU-401: Ver Valoraciones de una Carrera

CU-402: Añadir Valoración	
Descripción	La aplicación permite añadir una valoración a una carrera.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra los detalles de la carrera. 5. El usuario selecciona ”Ver Valoraciones”. 6. El sistema muestra las valoraciones de esa carrera. 7. El usuario introduce la valoración deseada 8. El sistema guarda y muestra la nueva valoración.
Flujo Alternativo	-
Postcondiciones	Se añade la nueva valoración al sistema.

Tabla 6.23: CU-402: Añadir Valoración

CU-403: Ver Valoraciones Usuario	
Descripción	La aplicación permite ver las valoraciones de un usuario.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado
Flujo	1. El usuario solicita ver sus datos en el sistema. 2. El sistema muestra el perfil del usuario con las acciones disponibles. 3. El usuario selecciona ”Ver Mis Reviews”. 4. El sistema muestra las valoraciones del usuario.
Flujo Alternativo	-
Postcondiciones	Se muestran las valoraciones de un usuario.

Tabla 6.24: CU-403: Ver Valoraciones Usuario

6.3.5 Casos de uso de inscripciones

CU-501: Inscribirse	
Descripción	La aplicación permite inscribirse en una prueba.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra la carrera con la lista de pruebas disponibles. 5. El usuario selecciona ”Inscribirse” al lado de la prueba deseada. 6. El sistema solicita confirmación por parte del usuario. 7. El usuario confirma la inscripción. 8. El sistema guarda la nueva inscripción y muestra un mensaje de éxito.
Flujo Alternativo	7. Si la carrera es de pago, el usuario debe de añadir datos adicionales.
Postcondiciones	Se añade la nueva inscripción al sistema.

Tabla 6.25: CU-501: Inscribirse

CU-502: Ver Inscripciones de una Prueba	
Descripción	La aplicación permite ver las inscripciones de una prueba.
Actores	Usuario registrado, Administrador
Precondiciones	El usuario registrado o el administrador debe de estar autenticado.
Flujo	1. El usuario solicita ver las carreras que hay en el sistema. 2. El sistema muestra la lista de todas las carreras. 3. El usuario selecciona ”Ver Detalles” de la carrera correspondiente. 4. El sistema muestra la carrera con la lista de las pruebas. 5. El usuario selecciona ”Ver Inscritos” al lado de la prueba deseada. 6. El sistema muestra las inscripciones de la prueba deseada.
Flujo Alternativo	-
Postcondiciones	Se muestran las inscripciones de una prueba.

Tabla 6.26: CU-502: Ver Inscripciones de una Prueba

CU-503: Ver Inscripciones Usuario	
Descripción	La aplicación permite ver las inscripciones de un usuario.
Actores	Usuario registrado
Precondiciones	El usuario registrado debe de estar autenticado
Flujo	1. El usuario solicita ver su información en el sistema. 2. El sistema muestra el perfil del usuario con las acciones disponibles. 3. El usuario selecciona ”Ver Mis Inscripciones”. 4. El sistema muestra las inscripciones del usuario.
Flujo Alternativo	-
Postcondiciones	Se muestran las inscripciones de un usuario.

Tabla 6.27: CU-503: Ver Inscripciones Usuario

6.3.6 Casos de uso de administrador

CU-601: Ver Todos los Comentarios	
Descripción	La aplicación permite al administrador ver todos los comentarios.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver la lista de todos los comentarios. 4. El sistema muestra todos los comentarios de las carreras.
Flujo Alternativo	-
Postcondiciones	Se muestran todos los comentarios.

Tabla 6.28: CU-601: Ver Todos los Comentarios

CU-602: Ver Todas las Valoraciones	
Descripción	La aplicación permite al administrador ver todos las valoraciones.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver la lista de todas las valoraciones. 4. El sistema muestra todas las valoraciones de las carreras.
Flujo Alternativo	-
Postcondiciones	Se muestran todas las valoraciones.

Tabla 6.29: CU-602: Ver Todas las Valoraciones

CU-603: Ver Todas las Inscripciones	
Descripción	La aplicación permite al administrador ver todos las inscripciones.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver la lista de todas las inscripciones. 4. El sistema muestra todas las inscripciones.
Flujo Alternativo	-
Postcondiciones	Se muestran todas las inscripciones.

Tabla 6.30: CU-603: Ver Todas las Inscripciones

CU-604: Ver Todos los Usuarios	
Descripción	La aplicación permite al administrador ver todos los usuarios.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver la lista de todos los usuarios. 4. El sistema muestra todos los usuarios.
Flujo Alternativo	-
Postcondiciones	Se muestran todos las usuarios.

Tabla 6.31: CU-604: Ver Todos los Usuarios

CU-605: Ver Todas las Pruebas	
Descripción	La aplicación permite al administrador ver todas las pruebas.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver la lista de todas las pruebas. 4. El sistema muestra todas las pruebas.
Flujo Alternativo	-
Postcondiciones	Se muestran todas las pruebas.

Tabla 6.32: CU-605: Ver Todas las Pruebas

CU-606: Ver Todas las Carreras	
Descripción	La aplicación permite al administrador ver todas las carreras.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema.. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver la lista de todas las carreras. 4. El sistema muestra todas las carreras.
Flujo Alternativo	-
Postcondiciones	Se muestran todas las carreras.

Tabla 6.33: CU-606: Ver Todas las carreras

CU-607: Ver Datos Administrador	
Descripción	La aplicación permite al administrador ver sus datos.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador solicita ver sus datos de administrador. 4. El sistema muestra los datos del administrador.
Flujo Alternativo	-
Postcondiciones	Se muestran los datos del administrador.

Tabla 6.34: CU-607: Ver Datos Administrador

CU-608: Modificar Datos Administrador	
Descripción	La aplicación permite al administrador modificar sus datos.
Actores	Administrador
Precondiciones	El administrador debe de estar autenticado.
Flujo	1. El administrador solicita acceder a su ”panel de control” en el sistema. 2. El sistema muestra el perfil administrador con las acciones disponibles. 3. El administrador selecciona ”Ver Datos Usuario Administrador”. 4. El sistema muestra los datos del administrador. 5. El administrador selecciona ”Editar Datos”. 6. El sistema muestra un formulario con los datos que se pueden modificar. 7. El administrador introduce sus nuevos datos correctamente. 8. El sistema cambia los datos del administrador.
Flujo Alternativo	El sistema muestra un mensaje de error si se introduce mal algún dato.
Postcondiciones	Se guardan los nuevos datos del administrador.

Tabla 6.35: CU-608: Modificar Datos Administrador

Capítulo 7

Diseño de la aplicación

En este capítulo se detallan los aspectos y decisiones importantes con respecto al diseño de la aplicación y su arquitectura. La construcción de este proyecto software se basa en el patrón de arquitectura **Model-View-Controller** o **Modelo-Vista-Controlador**, el patrón principal del que depende el resto de decisiones tomadas.

7.1 Patrones de diseño

7.1.1 Model-View-Controller

Model-View-Controller (MVC) es un patrón de arquitectura software que divide una aplicación en tres partes interconectadas: **modelo**, **vista** y **controlador**.[27] Esto se hace para separar las representaciones internas de la información de la forma en que se presenta la información a los usuarios finales de la aplicación. Además permite la reutilización eficiente del código fuente y el desarrollo de la aplicación en paralelo. Esta arquitectura es muy popular para diseñar aplicaciones web.

- **Model:** El **modelo** es la representación de la información con la que el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones. Envía a la vista aquella parte de la información que en cada momento se solicita para ser mostrada en la interfaz de usuario.
  Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.

- **View:** La **vista** se encarga de la representación de los datos contenidos en el modelo, presenta la interfaz de usuario, mostrando la información al usuario.

- **Controller:** El **controlador** es el encargado de interpretar las instrucciones y acciones que realizan los usuarios finales de la aplicación, haciendo uso de las funcionalidades

implementadas en el modelo. Se puede decir que el controlador hace de intermediario entre la vista y el modelo.

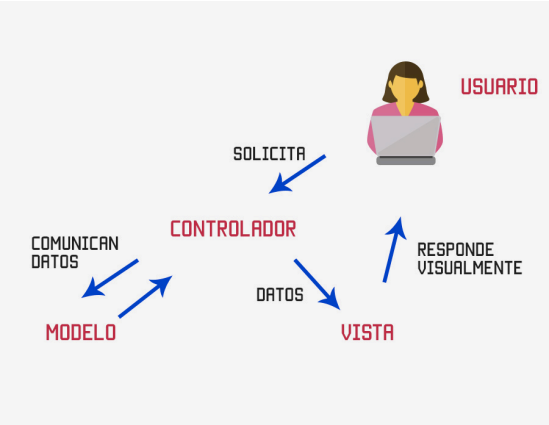


Figura 7.1: Patrón de diseño MVC

7.1.2 Facade

El patrón **Facade** (o Fachada) [28] es un tipo de patrón de diseño estructural.
Viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división de subsistemas, minimizando las comunicaciones y dependencias entre ellos.
Se aplica cuando se necesita proporcionar una interfaz simple para un subsistema complejo o cuando se quiere estructurar varios subsistemas en capas lógicas.

7.1.3 Repository

El patrón **Repository** [29] define una capa intermedia que separa la lógica de negocio de las interacciones con la fuente de datos que es la base de datos.
Se encarga de recuperar los datos el origen, mapearlos a las entidades y persistir cambios que se hagan en las entidades.
Este patrón se utiliza en la capa de acceso a datos, donde se definen los repositorios de la aplicación.

7.1.4 Data Transfer Object

Un **Data Transfer Object** (DTO) [30], es un objeto que se utiliza para intercambiar datos entre procesos.
La motivación de su uso tiene relación con que la comunicación entre procesos generalmente se realiza a interfaces remotas (como por ejemplo, los servicios web), donde cada llamada es

una operación costosa.
Una manera de reducir el número de llamadas es usar un objeto (DTO) que junte los datos a enviar, de esta forma se entregarían realizando una única llamada.
Los DTOs son objetos simples que no deben de contener ningún tipo de lógica de negocio, tan solo se necesitarán para transportar información entre los procesos.

7.1.5 Inversion of Control

Inversion of Control (IoC) [31] o inversión de control es un patrón de diseño cuya finalidad es la de eliminar del código las dependencias explícitas de los componentes, minimizando así el acoplamiento.
En entornos que no utilizan la inversión de control, los propios componentes son los encargados de localizar a otros componentes de los que dependen. Utilizando inversión de control, en lugar de ser el componente el que llama a una infraestructura en tiempo de ejecución para vincularse a otros componentes, la infraestructura es la que llama al componente y le proporciona los recursos necesarios para se ejecute.
La implementación del patrón ”Inversion of Control” fue realizada utilizando las anotaciones ”@Autowired” mediante el framework de Spring.

7.2 Arquitectura del sistema

7.2.1 Modelo

La capa modelo es la parte de la aplicación que encapsula toda la lógica de negocio y los accesos a la base de datos. En esta capa de la arquitectura es donde se almacenan, modifican y eliminan todos los datos que maneja la aplicación.

Entidades Persistentes

En la siguiente figura se puede ver el conjunto de todas las entidades persistentes de la aplicación representadas utilizando el lenguaje de modelado UML.

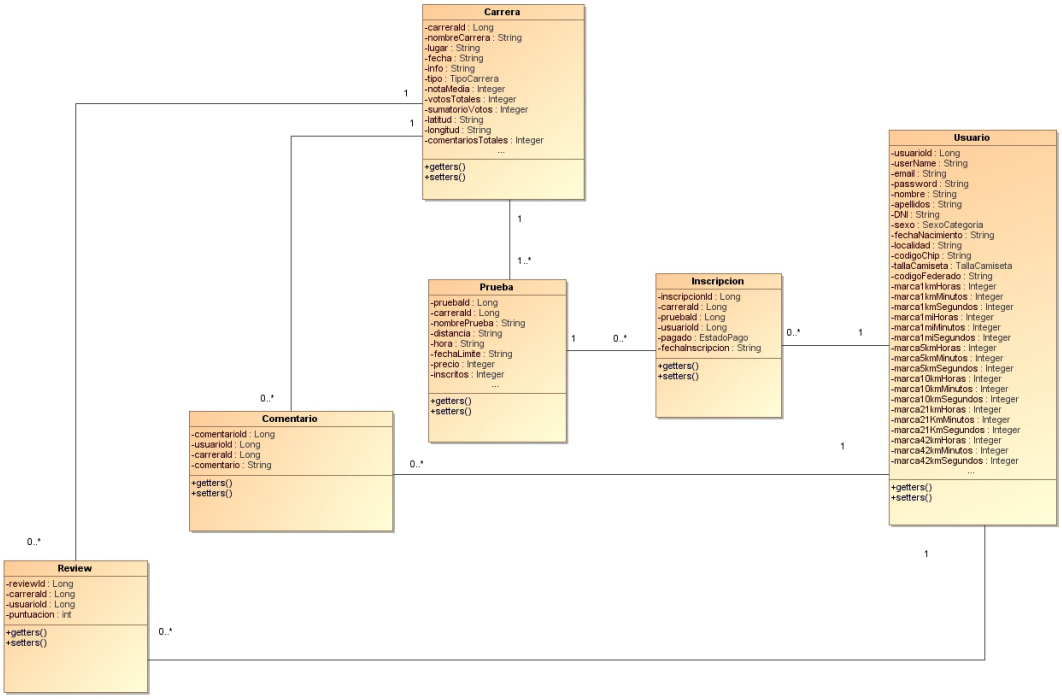


Figura 7.2: Diagrama de Clases

A continuación se detallan cada una de las entidades y se comentan algunos de sus atributos.

Usuario: Representa los datos de un usuario existente en el sistema.
El atributo sexo es del tipo enumerado SexoCategoria y puede tener los valores: ”Hombre” o ”Mujer”.
El atributo tallaCamiseta es del tipo enumerado TallaCamiseta y puede tener los valores: ”S”, ”M”, ”L”, ”XL” o ”XXL”.

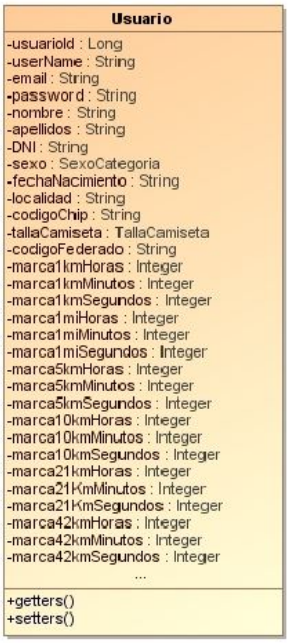


Figura 7.3: Entidad Usuario

Carrera: Representa los datos de una carrera existente en el sistema.
El atributo tipo es del tipo enumerado TipoCarrera y puede tener los valores: ”Running”, ”Cross” o ”Trail”.

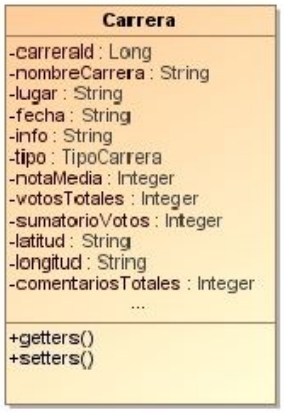


Figura 7.4: Entidad Carrera

Prueba: Representa los datos de una prueba existente en el sistema.

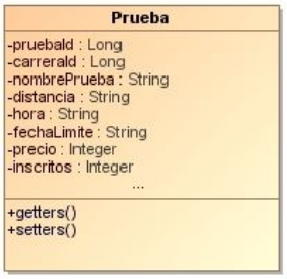


Figura 7.5: Entidad Prueba

Inscripcion: Representa los datos de una inscripción en el sistema.
El atributo pagado es del tipo enumerado EstadoPago y puede tener los valores: ”Pendiente” o ”Pagado”.

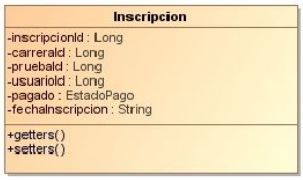


Figura 7.6: Entidad Inscripcion

Comentario: Representa los comentarios de una carrera.

Figura 7.7: Entidad Comentario

Review: Representa las valoraciones de una carrera.

Figura 7.8: Entidad Review

Capa de acceso a datos

Una vez vistas las entidades persistentes que forman la aplicación, a continuación, se explicará el diseño que se ha realizado para la capa de acceso a datos. En esta capa se ha hecho uso del patrón Repository. Como se observa en la figura 7.9, se definen todos los repositorios utilizados por la aplicación.

Los repositorios propios, extienden de una clase genérica llamada CrudRepository, que a su vez extiende de Repository. Estas dos clases genéricas son de tipo <T,ID>, en la que T es la entidad persistente, e ID es el tipo de dato que identifica dicha clase persistente. Gracias a extender de dichas clases los repositorios propios permiten realizar las operaciones CRUD (CREATE, READ, UPDATE, DELETE) sobre las entidades persistentes.

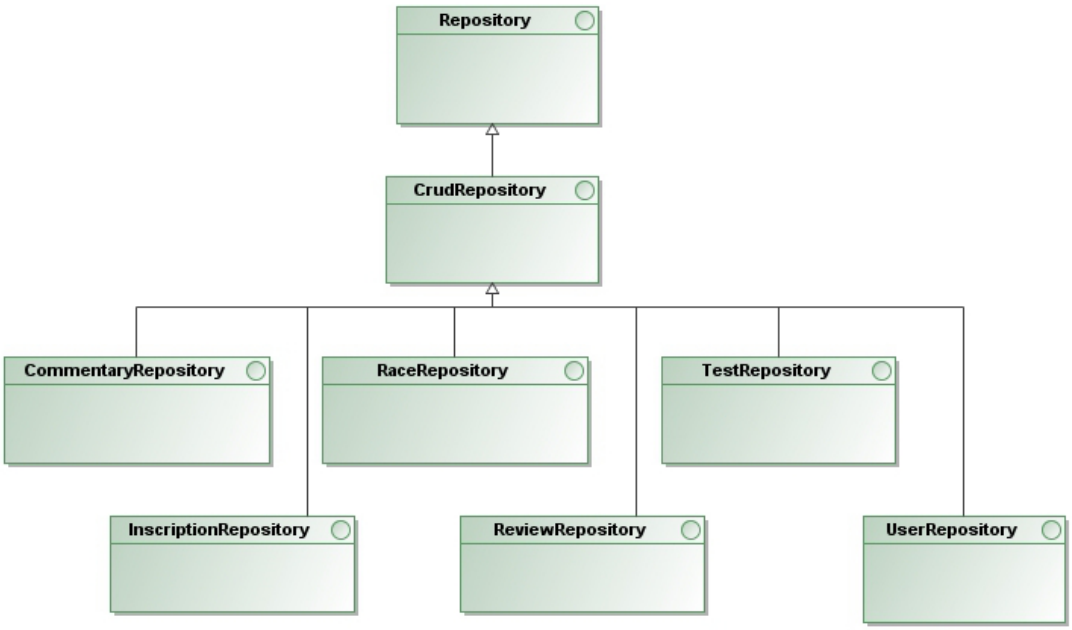


Figura 7.9: Diagrama de Repositorios

En la siguientes figuras se muestra el detalle de las interfaces de los repositorios.

UserRepository: Este repositorio contiene los métodos de acceso a datos de la entidad de usuarios. Además de las operaciones CRUD, se definen una serie de búsquedas personalizadas.

Figura 7.10: UserRepository

RaceRepository: Este repositorio contiene los métodos de acceso a datos de la entidad de carreras. Además de las operaciones CRUD, se definen una serie de búsquedas personalizadas, en las que destacan la operación de buscar por el tipo de carrera y la operación de buscar las 5 carreras mejor valoradas.

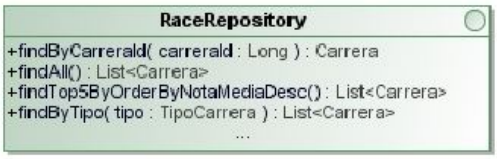


Figura 7.11: RaceRepository

TestRepository: Este repositorio contiene los métodos de acceso a datos de la entidad de pruebas. Además de las operaciones CRUD, se definen una serie de búsquedas personalizadas.

Figura 7.12: TestRepository

InscriptionRepository: Este repositorio contiene los métodos de acceso a datos de la entidad de inscripciones. Además de las operaciones CRUD, se definen una serie de búsquedas personalizadas.

Figura 7.13: InscriptionRepository

CommentaryRepository: Este repositorio contiene los métodos de acceso a datos de la entidad de comentarios. Además de las operaciones CRUD, se definen una serie de búsquedas personalizadas.

Figura 7.14: CommentaryRepository

ReviewRepository: Este repositorio contiene los métodos de acceso a datos de la entidad de valoraciones. Además de las operaciones CRUD, se definen una serie de búsquedas personalizadas.

Figura 7.15: ReviewRepository

Capa de servicio

La capa de servicio es la encargada de implementar la lógica de negocio, y exponerla en forma de funciones que serán llamadas por otras capas superiores. Esto se realiza utilizando los repositorios de la capa de acceso a datos del punto anterior. A continuación se detallan los servicios implementados:

UserService: Este servicio contiene las operaciones principales para gestionar los usuarios. Operaciones de búsqueda, creación, modificación y borrado.

Figura 7.16: UserService

RaceService: Este servicio contiene las operaciones para gestionar las carreras. Operaciones de búsqueda, creación, modificación y borrado.

Figura 7.17: RaceService

TestService: Este servicio contiene las operaciones para gestionar las pruebas. Operaciones de búsqueda, creación, modificación y borrado.

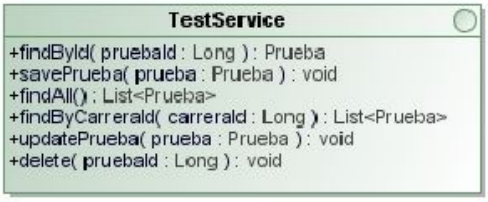


Figura 7.18: TestService

InscriptionService: Este servicio contiene las operaciones para gestionar las inscripciones. Operaciones de búsqueda y creación.

Figura 7.19: IncsriptionService

CommentaryService: Este servicio contiene las operaciones para gestionar los comentarios. Operaciones de búsqueda y creación.

Figura 7.20: CommentaryService

ReviewService: Este servicio contiene las operaciones para gestionar las valoraciones. Operaciones de búsqueda y creación.

Figura 7.21: ReviewService

7.2.2 Controlador

El controlador es la capa intermedia de la arquitectura, que actúa como ”intermediario” entre la vista y el modelo.
Su finalidad es la de procesar las peticiones que llegan desde la vista, gestionarlas y comunicarse con la parte del modelo que corresponde a esa petición, una vez el modelo realiza las operaciones requeridas, el controlador le devuelve a la vista los datos para mostrarlos al usuario.
Para esta aplicación, se ha desarrollado un controlador creando un servicio REST con diferentes URLs para cada funcionalidad que se muestran en el Apéndice A. Se ha utilizado el Framework de Spring, que se encarga de recibir las peticiones HTTP que provienen de la vista, decide que método Java de la capa de servicios es necesario utilizar y responde con la salida de dicho método.
A continuación se detallan los controladores que existen en la aplicación, se decidió implementar varios controladores REST diferentes para agrupar casos de uso relacionados con una misma temática.

- **MapController:** Este controlador gestiona la comunicación con el API de Google Maps.
- **MyErrorController:** Este controlador gestiona las páginas de error.
- **RaceController:** Este controlador se encarga de recibir y responder a peticiones relacionadas con el ámbito de las carreras. Algunas de esas peticiones son:
  - Añadir, modificar, eliminar carreras
  - Añadir, modificar, eliminar pruebas.
  - Mostrar, añadir comentarios de una carrera.
  - Mostrar, añadir valoraciones de una carrera.
  - Mostrar, añadir inscripciones de una prueba.
  - Mostrar detalles de una carrera.
  - etc.
- **UserController:** Este controlador se encarga de recibir y responder a peticiones relacionadas con el ámbito de los usuarios (añadir usuario, modificar usuario, etc).
  - Añadir usuario.
  - Añadir modificar usuario.
  - Borrar usuario.
  - Mostrar detalles de un usuario.

  - Mostrar comentarios de un usuario.
  - Mostrar valoraciones de un usuario.
  - Mostrar inscripciones de un usuario.
  - etc.

- **WelcomeController:** Este es el controlador ”principal” de la aplicación y se encarga de otras peticiones como por ejemplo ”/home”, ”/login”, etc.

Los controladores anteriores harán uso de diferentes DTOs para el intercambio de información, esto consigue independizar y desacoplar los subsistemas. Los DTOs implementados son los siguientes:

- **CommentaryDto**
- **InscripcionDto**
- **RaceDto**
- **ReviewDto**
- **TestDto**
- **UserDto**

7.2.3 Vista

La vista es la última capa de la arquitectura del sistema, es la capa que interactúa con el usuario, mostrándole la información y enviando las peticiones del usuario al controlador.

Interfaz de Usuario

Debido a que la aplicación se trata de una aplicación web, la capa de la vista se va a ejecutar en un navegador web, por lo tanto la vista estará compuesta por páginas HTML, con CSS para los estilos y JavaScript y jQuery para ofrecer más dinamismo y mejorar la interacción de los usuarios con las páginas. También se hizo uso de las plantillas del framework Bootstrap para desarrollar de una manera rápida y fácil los elementos más comunes de las páginas web.

Motor de plantillas

Se decidió usar el framework Thymeleaf ya que es un motor de plantillas extensible que permite definir y personalizar la manera en la que dichas plantillas serán procesadas hasta un gran nivel de detalle. Además ofrece una muy buena integración con Spring Framework.

Internacionalización (i18n)

La internacionalización (también conocida de manera abreviada como ”i18n”) es el proceso de diseñar software de manera que se pueda adaptar a diferentes idiomas y regiones sin necesidad de realizar cambios en el código fuente de la aplicación.
Para ofrecer la internacionalización en dos idiomas (español e inglés) a los usuarios de la aplicación será necesario crear dos ficheros ”.properties”.

7.3 Integracion con Google Maps API

En este proyecto se decidió implementar la integración con la API de Google Maps. Esta API permite que un usuario pueda utilizar los mapas online de Google.
La API de Google permite múltipes operaciones, pero en esta aplicación se ha centrado en la operación de obtener un marcador en el mapa dadas unas coordenadas GPS mediante las coordenadas de latitud y longitud. Para cuando el usuario necesite obtener información acerca de una carrera, obtenga además un ”mini-mapa” con su localización. [32]

Capítulo 8

Implementación

En este capítulo se van a exponer los detalles del proceso para poder implementar la aplicación. También se explicarán los pasos necesarios para poder compilar y ejecutar el proyecto.

8.1 Software requerido

A continuación se enumeran las herramientas y las versiones que se utilizaron para implementar todas las funcionalidades de la aplicación.
La mayoría de de las versiones utilizadas se encuentran en el fichero ”pom.xml” del proyecto.

- **Eclipse IDE for Java Developers:** Version Neon Release (4.6.0)
- **JDK:** 1.8.0
- **Apache Maven:** 1.8
- **Hibernate:** 5.0.11.Final
- **MySQL:** 5.1.40
- **Spring Boot:** 1.4.2.release
- **Sublime Text:** Version 3.2.2
- **Thymeleaf:** 2.1.5.release
- **Bootstrap:** 3.3.7
- **jQuery:** 1.11.1
- **Git:** 2.1.11
- **Tomcat:** 8.5.6

8.2 Estructura del proyecto

Para gestionar todo el proyecto se utilizó Maven, que se encarga de orquestar todos los elementos para el buen funcionamiento de la aplicación, desde la lógica de negocio hasta las plantillas de la vista. También se encarga de gestionar todas las dependencias y versiones del software utilizado.
Esta configuración se define en el fichero ”pom.xml”, ubicado en la raíz del proyecto.

8.2.1 Estructura proyecto Java

A continuación se muestra la jerarquía de los directorios y paquetes existentes en el desarrollo del proyecto Java, junto con una explicación de los directorios más importantes.

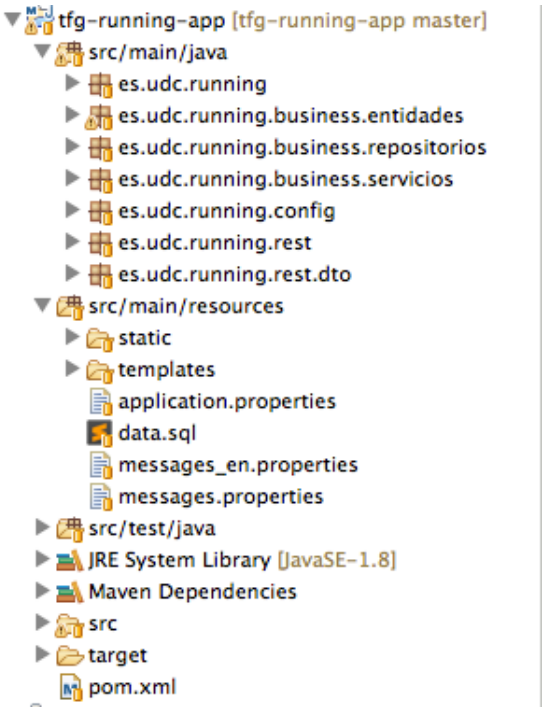


Figura 8.1: Estructura del proyecto Java

En el directorio **src/main/java** se encuentra el código fuente de la parte del modelo y de la parte controladora de la aplicación.

- **es.udc.running:** Dentro de este paquete solamente se encuentra la clase SpringBootWebApplication.java que es la que permite lanzar la aplicación Spring Boot de manera sencilla.
- **es.udc.running.business.entidades:** Paquete en el que están las clases que representan las entidades persistentes de la base de datos.

- **es.udc.running.business.repositorios:** Paquete en el que están los repositorios de la aplicación.
- **es.udc.running.business.servicios:** Paquete en el que están los servicios del modelo, donde se implementa la lógica de negocio de la aplicación.
- **es.udc.running.config**: Paquete en el que están las clases que permiten configurar Spring, aquí se encuentran las clases Internationalization.java y SpringSecurityConfig.java.
- **es.udc.running.rest**: Paquete en el que se encuentran las clases que actúan como controladoradoras dentro de la aplicación.
- **es.udc.running.rest.dto**: Paquete en donde se encuentran los DTOs para transmitir información entre la interfaz de usuario y la lógica de negocio.

En el directorio **src/main/resources** se encuentran los ficheros relacionados con la interfaz de usuario o vista de la aplicación web además de otros ficheros que se van a explicar a continuación:

- **src/main/resources/static:** directorio en donde se encuentran los archivos CSS y las imágenes que necesita la aplicación.
- **src/main/resources/templates:** directorio en donde se encuentran todas las plantillas HTML de la aplicación.
- **application.properties:** fichero que define propiedades para configurar el entorno Spring.
- **data.sql:** fichero que funciona como un script para insertar datos en la base de datos al arrancar la aplicación.
- **messages_en.properties:** fichero que contiene los mensajes que utilizan las plantillas HTML en inglés.
- **messages.properties:** fichero que contiene los mensajes que utilizan las plantillas HTML en español.

8.3 Instrucciones de compilación y ejecución

Una vez instalado todo el software necesario, se podrá compilar y ejecutar el proyecto. Los siguientes pasos indican como se consigue ejecutar y probar la aplicación:

1. Clonar el proyecto que está subido en GitLab FIC o copiar el código fuente del CD. **git clone git@git.fic.udc.es:diego.sanchez.cal/tfg-dsc.git**
2. Entrar en el directorio raíz.
3. Ejecutar el proyecto utilizando Maven con el comando: **mvn spring-boot:run**
4. Abrir un navegador web en la dirección: **http://localhost:8080/**

Capítulo 9

Pruebas

En este capítulo se van a explicar las pruebas que se realizaron durante el desarrollo de la aplicación para comprobar su correcto funcionamiento. La fase de pruebas es muy importante en un proyecto software para asegurar que la implementación está libre de fallos y que además cumple con los requisitos acordados previamente.

9.1 Pruebas unitarias

Las pruebas unitarias se realizan para comprobar que un componente del sistema funciona correctamente.
Se tienen que poder ejecutar de manera automática, repetir tantas veces como se desee, ejecutarse independientemente y que no afecte a la ejecución de otra prueba unitaria.
Se realizan al final de cada iteración, comprobando cada caso de uso por separado y verificando que los resultados son los esperados.
Estas pruebas han sido realizadas de forma manual siguiendo una batería de pruebas, para asegurar el correcto comportamiento de las funciones en todos los casos de uso, comprobando los resultados en la base de datos.

9.2 Pruebas de integración

Las pruebas de integración consisten en la comprobación de que dos o más unidades software individuales funcionan juntas correctamente, probándolas en grupo.
La aplicación se integra con las funcionalidades de la API de Google Maps, para mostrar un mini-mapa en el detalle de cada carrera.

9.3 Pruebas de sistema

El objetivo de las pruebas de sistema es comprobar el correcto funcionamiento de la aplicación en todo su conjunto, es decir el modelo, la vista y el controlador.
Estas pruebas se han realizado utilizando dos navegadores web diferentes, Firefox y Google Chrome, comprobando que todas las páginas funcionan correctamente.

9.4 Pruebas de aceptación

Las pruebas de aceptación se realizan una vez finalizado todo el desarrollo, para determinar si la aplicación cumple con las necesidades y requisitos del cliente.
En este caso, las pruebas se realizaron por el autor del proyecto junto con el director del proyecto, validando el correcto funcionamiento de la aplicación y comprobando que se cumplen los requisitos iniciales.

Capítulo 10

Conclusiones y futuras líneas de trabajo

10.1 Conclusiones

El sistema cumple con todos los requisitos previamente acordados, ofreciendo a los usuarios una aplicación web con una interfaz intuitiva y sencilla. Se ha desarrollado una aplicación web que permite buscar carreras, obtener información sobre ellas, inscribirse, emitir comentarios y valoraciones.

El objetivo ha sido desarrollar la aplicación web lo más cercana al mundo real posible e intentando que el trabajo final se aproximase a una aplicación real.

La interfaz web ha sido desarrollada utilizando HTML junto con BootStrap y el motor de plantillas Thymeleaf. Proporcionándole estilo con CSS y dinamismo con JavaScript y JQuery, para mejorar la interacción de los usuarios con la aplicación. Además la aplicación ofrece internacionalización (i18n) de la aplicación, traduciendo los textos en dos idiomas: español e inglés.
A su vez, se ha desarrollado un API REST, empleando las tecnologías Java y Spring Boot, que permitieron minimizar el tiempo de puesta en marcha del proyecto y el tiempo de desarrollo. Esta API es utilizada por la vista para recuperar datos de la base de datos e realizar operaciones.

En cuanto a la API externa de Google Maps, se ha implementado la función de mostrar la ubicación de una carrera, mostrando un mapa.

Mediante el desarrollo de la aplicación, se han adquirido nuevos conceptos, experiencia y conocimiento de tecnologías empleadas en el mundo laboral.

10.2 Futuras líneas de trabajo

Como cualquier proyecto software, siempre está abierto a implementar nuevas funcionalidades y mejoras que incrementen el valor del producto final. A continuación se exponen las posibles líneas de trabajo si se desean implementar nuevas funciones o mejoras para aumentar el valor del producto.

- Gestión de los resultados de las carreras.
- Edición del perfil de usuario con una imagen o avatar.
- Integración de diferentes APIs de redes sociales como Facebook o Twitter.
- Integración con una API de pagos como por ejemplo PayPal.
- Internacionalización con otros idiomas.
- Implementar un sistema recomendador de carreras basado en las valoraciones realizadas por los usuarios.

Apéndices

Apéndice A

API REST

A continuación se especifican los controladores junto con sus ”mappings” de las peticiones HTTP más relevantes para el buen funcionamiento de la aplicación web.

Controlador MapController:

/carreras/{carreraId}/mapa	
Descripción	Obtiene el mapa completo de una carrera
Método HTTP	GET
Parámetros	carreraId

Tabla A.1: Obtener el mapa completo de una carrera

Controlador RaceController:

/formularioCarreras	
Descripción	Obtiene el formulario para añadir una carrera.
Método HTTP	GET
Parámetros	-

Tabla A.2: Obtener el formulario para añadir una carrera

/formularioCarreras	
Descripción	Añade una carrera.
Método HTTP	POST
Parámetros	-

Tabla A.3: Añadir una carrera

/carreras/{carreraId}	
Descripción	Obtiene los detalles de una carrera.
Método HTTP	GET
Parámetros	carreraId

Tabla A.4: Obtener los detalles de una carrera

/carreras/{carreraId}/comentarios	
Descripción	Obtiene los comentarios de una carrera.
Método HTTP	GET
Parámetros	carreraId

Tabla A.5: Obtener los comentarios de una carrera

/carreras/{carreraId}/comentarios	
Descripción	Añade un comentario a una carrera.
Método HTTP	POST
Parámetros	carreraId

Tabla A.6: Añadir un comentario a una carrera

/carreras/{carreraId}/reviews	
Descripción	Obtiene las valoraciones de una carrera.
Método HTTP	GET
Parámetros	carreraId

Tabla A.7: Obtener las valoraciones de una carrera

/carreras/{carreraId}/reviews	
Descripción	Añade una valoración a una carrera.
Método HTTP	POST
Parámetros	carreraId

Tabla A.8: Añadir una valoración a una carrera

/carreras/{carreraId}/delete	
Descripción	Elimina una carrera.
Método HTTP	GET
Parámetros	carreraId

Tabla A.9: Eliminar una carrera

/carreras/{carreraId}/update	
Descripción	Obtiene el formulario para modificar una carrera
Método HTTP	GET
Parámetros	carreraId

Tabla A.10: Obtener el formulario para modificar una carrera

/carreras/carreraId/update	
Descripción	Modifica una carrera.
Método HTTP	POST
Parámetros	carreraId

Tabla A.11: Modificar una carrera

/carreras/carreraId/formularioPruebas	
Descripción	Obtiene el formulario para añadir una prueba.
Método HTTP	GET
Parámetros	carreraId

Tabla A.12: Obtener el formulario para añadir una prueba

/carreras/carreraId/formularioPruebas	
Descripción	Añade una prueba.
Método HTTP	POST
Parámetros	carreraId

Tabla A.13: Añadir una prueba

/carreras/carreraId/pruebaId/delete	
Descripción	Elimina una prueba.
Método HTTP	GET
Parámetros	carreraId, pruebaId

Tabla A.14: Eliminar una prueba

/carreras/carreraId/pruebaId/update	
Descripción	Obtiene el formulario para modificar una prueba.
Método HTTP	GET
Parámetros	carreraId, pruebaId

Tabla A.15: Obtener el formulario para modificar una prueba

/carreras/carreraId/pruebaId/update	
Descripción	Modifica una prueba.
Método HTTP	POST
Parámetros	carreraId, pruebaId

Tabla A.16: Modificar una prueba

/carreras/carreraId/pruebaId/inscripciones	
Descripción	Obtiene las inscripciones de una prueba.
Método HTTP	GET
Parámetros	carreraId, pruebaId

Tabla A.17: Obtener las inscripciones de una prueba

/carreras/carreraId/pruebaId/inscribirse	
Descripción	Obtiene la página de insripción de una prueba.
Método HTTP	GET
Parámetros	carreraId, pruebaId

Tabla A.18: Obtener la página de inscripción de una prueba

/carreras/carreraId/pruebaId/inscribirse	
Descripción	Añade una nueva inscripción.
Método HTTP	POST
Parámetros	carreraId, pruebaId

Tabla A.19: Añadir una nueva inscripción

/carreras/carreraId/pruebaId/inscribirse/success	
Descripción	Obtine la página de inscripción aceptada.
Método HTTP	GET
Parámetros	carreraId, pruebaId

Tabla A.20: Obtener la página de inscripción aceptada

/carreras/carreraId/pruebaId/inscribirse/payment	
Descripción	Obtiene el formulario de pago.
Método HTTP	GET
Parámetros	carreraId, pruebaId

Tabla A.21: Obtener el formulario de pago

/carreras/carreraId/pruebaId/inscribirse/payment	
Descripción	Añade una inscripción cuando una prueba es de pago.
Método HTTP	POST
Parámetros	carreraId, pruebaId

Tabla A.22: Añadir inscripción cuando una prueba es de pago

Controlador UserController:

/formulario	
Descripción	Obtiene el formulario para registrarse.
Método HTTP	GET
Parámetros	-

Tabla A.23: Obtener el formulario para registrarse

/formulario	
Descripción	Añade un nuevo usuario.
Método HTTP	POST
Parámetros	-

Tabla A.24: Añadir un nuevo usuario

/usuarios	
Descripción	Obtiene la lista de usuarios.
Método HTTP	GET
Parámetros	-

Tabla A.25: Obtener la lista de usuarios

/user/details	
Descripción	Obtiene los detalles del usuario.
Método HTTP	GET
Parámetros	-

Tabla A.26: Obtener los detalles del usuario

/user/delete	
Descripción	Elimina un usuario.
Método HTTP	GET
Parámetros	-

Tabla A.27: Eliminar un usuario

/user/update	
Descripción	Obtiene el formulario para modificar un usuario.
Método HTTP	GET
Parámetros	-

Tabla A.28: Obtener el formulario para modificar un usuario

/user/update	
Descripción	Modifica un usuario.
Método HTTP	POST
Parámetros	-

Tabla A.29: Modificar un usuario

/user/comentarios	
Descripción	Obtiene los comentarios de un usuario.
Método HTTP	GET
Parámetros	-

Tabla A.30: Obtener los comentarios de un usuario

/user/reviews	
Descripción	Obtiene las valoraciones de un usuario.
Método HTTP	GET
Parámetros	-

Tabla A.31: Obtener las valoraciones de un usuario

/admin/details	
Descripción	Obtiene los detalles del administrador.
Método HTTP	GET
Parámetros	-

Tabla A.32: Obtener los detalles del administrador

/admin/update	
Descripción	Obtiene el formulario para modificar los datos del administrador.
Método HTTP	GET
Parámetros	-

Tabla A.33: Obtener el formulario para modificar los datos del administrador

/admin/update	
Descripción	Modifica los datos del administrador.
Método HTTP	POST
Parámetros	-

Tabla A.34: Modificar los datos del administrador

/user/inscripciones	
Descripción	Obtiene las inscripciones de un usuario.
Método HTTP	GET
Parámetros	-

Tabla A.35: Obtener las inscripciones de un usuario

Controlador WelcomeController:

/home	
Descripción	Obtiene la página principal.
Método HTTP	GET
Parámetros	-

Tabla A.36: Obtener la página principal

/login	
Descripción	Obtiene la página de login.
Método HTTP	GET
Parámetros	-

Tabla A.37: Obtener la página de login

/admin	
Descripción	Obtiene la página del perfil administrador.
Método HTTP	GET
Parámetros	-

Tabla A.38: Obtener la página del perfil administrador

/user	
Descripción	Obtiene la página del perfil usuario.
Método HTTP	GET
Parámetros	-

Tabla A.39: Obtener la página del perfil usuario

/about	
Descripción	Obtiene la página ”Acerca De”.
Método HTTP	GET
Parámetros	-

Tabla A.40: Obtener la página ”Acerca De”

/help	
Descripción	Obtiene la página de ayuda.
Método HTTP	GET
Parámetros	-

Tabla A.41: Obtener la página de ayuda

/carreras	
Descripción	Obtiene la lista de todas las carreras.
Método HTTP	GET
Parámetros	-

Tabla A.42: Obtener la lista de todas las carreras

/pruebas	
Descripción	Obtiene la lista de todas las pruebas.
Método HTTP	GET
Parámetros	-

Tabla A.43: Obtener la lista de todas las pruebas

/inscripciones	
Descripción	Obtiene la lista de todas las inscripciones.
Método HTTP	GET
Parámetros	-

Tabla A.44: Obtener la lista de todas las inscripciones

/comentarios	
Descripción	Obtiene la lista de todos los comentarios.
Método HTTP	GET
Parámetros	-

Tabla A.45: Obtener la lista de todos los comentarios

/reviews	
Descripción	Obtiene la lista de todas las valoraciones.
Método HTTP	GET
Parámetros	-

Tabla A.46: Obtener la lista de todas las valoraciones

Lista de acrónimos

AJAX *Asynchronous JavaScript And XML.*

API *Application Programming Interface.*

CLI *Command Line Interface.*

CRUD *Create, Read, Update, Delete.*

CSS *Cascading Style Sheets.*

DOM *Document Object Model.*

DTO *Data Transfer Object.*

HTML *HyperText Markup Language.*

HTTP *HyperText Transfer Protocol.*

IDE *Integrated Development Environment.*

IoC *Inversion of Control.*

JDK *Java Development Kit.*

JPA *Java Persistence API.*

MVC *Model-VIew-Controller.*

OSI *Open System Interconnection.*

POM *Project Object Module.*

REST *Representational State Transfer.*

RUP *Rational Unified Process.*

SQL *Structured Query Language.*

UML *Unified Modeling Language.*

URL *Uniform Resource Locator.*

XML *eXtensible Markup Languaje.*

W3C *World Wide Web Consortium.*

WWW *World Wide Web.*

Glosario

Bean Un Bean es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los componentes uno a uno.

Framework Un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual, otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Bibliografía

[1] B. J. G. S. G. B. James Gosling and A. Buckley, "The java language specification java se 8 edition," 2015. [En línea]. Disponible en: https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[2] "Documentación html y css de la w3c," 2020. [En línea]. Disponible en: https://www.w3.org/standards/webdesign/htmlcss

[3] "Documentación javascript de la w3c," 2020. [En línea]. Disponible en: https://www.w3.org/standards/webdesign/script

[4] "Sql -wikipedia," 2020. [En línea]. Disponible en: https://en.wikipedia.org/wiki/SQL

[5] "Página oficial de latex," 2020. [En línea]. Disponible en: https://www.latex-project.org/

[6] "Página oficial de hibernate," 2020. [En línea]. Disponible en: https://hibernate.org/orm/

[7] "Página oficial de spring," 2020. [En línea]. Disponible en: https://spring.io/

[8] "Documentación de thymeleaf," 2020. [En línea]. Disponible en: https://www.thymeleaf.org/documentation.html

[9] "About jquery," 2020. [En línea]. Disponible en: https://learn.jquery.com/about-jquery/

[10] "Documentación de bootstrap," 2020. [En línea]. Disponible en: https://getbootstrap.com/docs/3.3/

[11] "Documentación de mysql," 2020. [En línea]. Disponible en: https://dev.mysql.com/doc/

[12] "Pagina oficial de google maps," 2020. [En línea]. Disponible en: https://cloud.google.com/maps-platform?hl=es

[13] "Eclipse documentation," 2020. [En línea]. Disponible en: https://www.eclipse.org/documentation/

[14] "Página oficial de sublime text," 2020. [En línea]. Disponible en: https://www.sublimetext.com/

[15] "What is maven?" 2020. [En línea]. Disponible en: https://maven.apache.org/what-is-maven.html

[16] "Página oficial de git," 2020. [En línea]. Disponible en: https://git-scm.com/

[17] "Página oficial de overleaf," 2020. [En línea]. Disponible en: https://es.overleaf.com/

[18] "Magicdraw introduction," 2020. [En línea]. Disponible en: https://www.nomagic.com/products/magicdraw#intro

[19] "Página oficial de ganttproject," 2020. [En línea]. Disponible en: https://www.ganttproject.biz/

[20] "Página oficial de apache tomcat," 2020. [En línea]. Disponible en: http://tomcat.apache.org/

[21] T. B.-L. et al., "Hypertext transfer protocol – http/1.1," 1999. [En línea]. Disponible en: https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf

[22] "Rest - mdn web docs mozilla," 2020. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Glossary/REST

[23] "Documentacion xml de la w3c," 2020. [En línea]. Disponible en: https://www.w3.org/XML/

[24] R.F.GroveandE.Ozkan, "The mvc-web design pattern." [En línea]. Disponible en: https://www.scitepress.org/Papers/2011/32969/32969.pdf

[25] "Descripción de capas lógicas," 2010. [En línea]. Disponible en: https://docs.oracle.com/cd/E19528-01/820-0888/aaubb/index.html

[26] "Rational unified process: Best practices for software development teams," 2001. [En línea]. Disponible en: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[27] "Mvc architecture - mdn web docs mozilla," 2019. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[28] "The facade design pattern - problem, solution, and applicability - w3sdesign.com," 2014. [En línea]. Disponible en: http://w3sdesign.com/?gr=s05&ugr=proble

[29] "The repository pattern - docs.microsoft.com," 2010. [En línea]. Disponible en: https://docs.microsoft.com/gl-es/previous-versions/msp-n-p/ff649690(v=pandp.10)

[30] "Data transfer object - docs.microsoft.com," 2014. [En línea]. Disponible en: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649585(v=pandp.10)?redirectedfrom=MSDN

[31] M. Fowler, "Inversion of control containers and the dependency injection pattern," 2004. [En línea]. Disponible en: https://martinfowler.com/articles/injection.html

[32] "Tutorial maps javascript api," 2020. [En línea]. Disponible en: https://developers.google.com/maps/documentation/javascript/tutorial?hl=es