UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MASTER

"Challenge Memory: Red social basada en resultados deportivos"

DIRECTOR: Benjamín López Pérez

AUTOR: Isaac Fernández Muñiz

Vº Bº del Director del Proyecto

Agradecimientos

He de dar la gracias a dos personas que han sido claves para finalizar mis estudios del Máster en Ingeniería Web, que por desgracia había dejado olvidados simplemente por no disponer del tiempo y la motivación necesaria:

- A Benjamín, mi director de TFM. Que no ha dudado en apoyarme con la dirección de esta idea propia y ha estado siempre para guiarme en cualquier momento de duda, pero también para exigirme lo que de un trabajo así se espera.

- Y especialmente a mi mujer, Patri. Que es la persona que siempre ha estado ahí para animarme cuando quería bajar los brazos, para insistir en que no podía rendirme justo a un paso de la meta, y sobre todo, por sacrificarse, quererme y apoyarme en estos meses en los que mi tiempo libre lo he dedicado a este trabajo y no a disfrutarlo con ella. Te quiero

Resumen

Este proyecto dará lugar al desarrollo de un prototipo de aplicación web, que permitirá a sus usuarios el almacenamiento histórico de todos sus resultados deportivos en disciplinas individuales, en un principio natación, atletismo, ciclismo y triatlón. Con esta información se creará una red social que permitirá a los usuarios interactuar entre ellos, creándose automáticamente clasificaciones virtuales de competiciones registradas y permitirá realizar filtros por diferentes conceptos (amigos, sexo...). Se encontrarán funcionalidades como el seguimiento de resultados de otros usuarios, generación de gráficas de evolución en diferentes ediciones de la misma competición.

Se optará por el uso de una tecnología relativamente joven como es Spring Boot. De gran potencial, que permitirá al desarrollador obtener nuevos conocimientos y avanzar con mayor rapidez, aunque implicará un riesgo al no contar con experiencia con la misma.

El proyecto será ejecutado entre los meses de diciembre de 2017 y mayo de 2018 para ser presentado como Proyecto Fin de Máster del Máster de Ingeniería Web de la Universidad de Oviedo.

Palabras Clave

Deporte, Competición, Historial, Red Social, Spring Boot.

Abstract

This project will lead to the development of a web application prototype, which will allow its users to store all their sporting results in individual disciplines, initially swimming, athletics, cycling and triathlon. With this information a social network will be created that will allow users to interact with each other, automatically creating virtual classifications of registered competitions and will allow filtering by different concepts (friends, sex...). You will find features such as tracking the results of other users, generation of evolution graphs in different editions of the same competition.

It will opt for the use of a relatively young technology such as Spring Boot. Of great potential, which will allow the developer to obtain new knowledge and advance more quickly, although it will involve a risk when not having experience with it.

The project will be carried out between the months of December 2017 and May 2018 to be presented as the Final Master Project of the Master of Web Engineering at the University of Oviedo.

Keywords

Sports, competition, historic, social network, Spring Boot.

Índice General

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

El objetivo principal del proyecto es la finalización del Máster en Ingeniería Web de la Universidad de Oviedo. Como objetivo secundario, se obtendría un prototipo para dar una alternativa online a personas que practican actividades deportivas competitivas en diferentes deportes.

El proyecto busca dar la posibilidad de almacenar un histórico de todos los resultados deportivos en diferentes disciplinas.

Con esta información se creará una red social que permitirá a los usuarios interactuar entre ellos, creándose automáticamente clasificaciones virtuales de competiciones registradas y permitirá realizar filtros por diferentes conceptos (amigos, sexo...). Se encontrarán funcionalidades como el seguimiento de resultados de otros usuarios y la generación de gráficas de evolución en diferentes ediciones de la misma competición.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

El proyecto actual surge para cubrir una necesidad de un perfil de usuario en auge como son las personas que practican actividades deportivas competitivas en diferentes deportes. En principio centrándose en el atletismo, el ciclismo, la natación y el triatlón.

Los usuarios podrán almacenar un histórico de todos sus resultados deportivos en diferentes disciplinas. Con esta información se creará una red social que permitirá a los usuarios interactuar entre ellos, creándose automáticamente clasificaciones virtuales de competiciones registradas y permitirá realizar filtros por diferentes conceptos (amigos, sexo...). Se encontrarán funcionalidades como el seguimiento de resultados de otros usuarios y la generación de gráficas de evolución en diferentes ediciones de la misma competición.

2.2 Objetivos del Proyecto

Los siguientes objetivos han de ser cubiertos a la finalización del proyecto:

- Concluir los estudios del Máster en Ingeniería web

- Creación de un prototipo de aplicación web que permita a personas aficionadas o profesionales del deporte:

    - El registro y autenticación de usuarios y su acceso mediante credenciales.
    - Gestionar la información de los usuarios, configurando su privacidad.
    - Incluir y modificar la información de resultados deportivos.
    - Seguir a otros usuarios.
    - Interactuar con otros usuarios con comentarios y "me gusta".
    - Buscar usuarios y resultados.

También se podrán acometer funcionalidades avanzadas como la generación de gráficas de evolución en diferentes ediciones de la misma competición u otra explotación de datos que pudiese ser interesante.

2.3 Estudio de la Situación Actual

En la actualidad, existen herramientas con fines similares, pero que no centran su objetivo final en el mismo concepto que el presente proyecto, como son los sistemas de registro de actividad deportiva y la plataforma Strava.

En este proyecto, centrándonos en exclusiva en la información de resultados deportivos, se pretende conseguir que el usuario tenga un registro histórico de estos, con información detallada de cada competición y que esta información a su vez permita generar una competencia entre los usuarios al estilo Strava. Una funcionalidad de la que no conocemos una alternativa en la actualidad.

En el aspecto tecnológico, se han valorado varias alternativas:

- *PHP*: lenguaje muy extendido y de gran facilidad de implantación en multitud de alojamientos web.
- *Ruby On Rails*: tecnología de gran potencia pero muy desconocida por el autor del proyecto.
- *Spring Boot (J2EE)*: plataforma relativamente joven y de gran potencial, que integra multitud de tecnologías contrastadas que a su vez se sustentan en todo un referente del desarrollo web, como es la tecnología J2EE.

Finalmente se decide el uso de **Sprint Boot**. Pese a no tener experiencia concreta con ella, si se cuenta con un bagaje importante en diferentes plataformas J2EE y por tanto se espera una alta productividad en un periodo de tiempo breve.

2.3.1 Evaluación de Alternativas

2.3.1.1 Sistemas de registro de actividad deportiva

Existen plataformas para el almacenamiento de toda la actividad deportiva de un usuario, casi siempre asociada a algún sistema de registro GPS. Ejemplos de este tipo de sistemas existen multitud, de los cuales podemos destacar: *Garmin Connect*, *Polar Flow*, *MovesCount* o *Endomondo*. Estas alternativas permiten el registro detallado de cada entrenamiento y competición, así como un análisis en profundidad del mismo. Aunque la información de cada competición queda registrada, no ofrecen la posibilidad de almacenar datos concretos de las competiciones disputadas ni la explotación de esa información entre usuarios.

2.3.1.2 Strava

Por otro lado tenemos la red social deportiva por excelencia, *Strava*. Una aplicación que permite explotar toda la información registrada normalmente en plataformas ajenas (como las citadas en el párrafo anterior) y, a través del procesamiento de los recorridos GPS, establecer tablas de clasificación en "segmentos" de esos recorridos definidos por los propios usuarios. En *Strava* tenemos el concepto de red social basada en la competencia entre usuarios, pero no permite el almacenamiento y seguimiento concreto de resultados en competiciones.

Capítulo 3. Aspectos Teóricos

3.1 Spring Boot

Spring es un framework muy utilizado en los últimos años y como parte del crecimiento de su ecosistema, varios proyectos fueron desarrollados, como Spring Data, Spring Security, etc. Pero, la integración de estos proyectos requiere de configuración, la cual es considerada una tarea repetitiva en los diversos proyectos y el cual ha traído problemas a algunos.

SPRING BOOT

Spring Boot es el proyecto más reciente de spring que nos ayuda a iniciar nuestro proyecto utilizando los diferentes proyectos de spring de una manera ágil y evitar la configuración, la cual hemos lidiado por mucho tiempo. Toda esta magia se debe a la configuración por defecto que trae dentro y que puede ser configurada vía propiedades. Spring Boot nos permite poner enfoque en agregar valor y mejorar la experiencia del desarrollador.

CARACTERÍSTICAS

Spring Boot provee lo siguiente:

Convención sobre configuración

En lugar de estar escribiendo la configuración necesaria y validar si es correcta, Spring Boot provee las configuraciones necesarias bajo diferentes escenarios. De esta manera evitamos la tarea repetitiva de estar agregando las configuraciones.

Gestión de dependencias

A veces, tenemos que lidiar con las versiones de las dependencias que usamos y las dependencias transitivas que existen entre ellas. Lo cual nos lleva a hacer un análisis de las dependencias de nuestro proyecto y limpiar la duplicidad de dependencias con diferentes versiones y hacer funcionar nuestros proyectos con las dependencias indicadas. Spring Boot provee un análisis de las dependencias que los proyectos alrededor de spring utilizan de manera que nosotros solo tenemos que indicar que dependencia necesitamos sin necesidad de indicar la versión. Al mismo tiempo, ofrece una manera práctica de actualizar la versión en caso de que se quiera hacer un upgrade o downgrade de la misma.

Auto-configuration

Como se mencionó en un inicio, spring y sus diferentes proyectos necesitan ser configurados para que funcionen de manera adecuada y podamos continuar agregando valor a nuestra aplicación. Pero la realidad es que muchas veces invertimos mucho tiempo en esas configuraciones. Spring Boot es lo suficientemente inteligente para poder activar las configuraciones necesarias si cumple con ciertas condiciones como si las clases están en el classpath o los beans han sido creados o los properties correspondientes han sido habilitados.

Starters dependencies

¿Puedes recordar todas las dependencias que necesitas si quieres utilizar spring-data-jpa o spring-web? ¿Necesitas echarle un vistazo a tu anterior proyecto? El trabajo se hace más fácil si tan solo pudiera decirle a mi proyecto necesito usar jpa y web. Por esta razón, spring-boot provee este tipo de dependencias que traen consigo las dependencias necesarias para empezar nuestro proyecto y ahorrarnos el trabajo de ir a buscar nuestra plantilla con dependencias a otro lado.

Embedded Server

Spring Boot da soporte para Tomcat, Jetty y Undertow embebidos.

PROYECTO SPRING UTILIZADOS

Proyectos Spring más reseñables utilizados en este desarrollo e integrados a través de Spring Boot: Spring MVC, Spring Security, MongoRepository, i18n

3.2 Plantillas Thymeleaf

Thymeleaf es un framework de templating que ofrece flexibilidad y adecuación a los nuevos estándares como HTML5. Entre sus muchas características, cabe destacar las siguientes:

- Diseñado para XML, XHTML y HTML5, pero extensible a otros formatos: Puede ser usado en cualquier ámbito, no sólo Web. No tiene dependencias con el API de servlets.
- Tiene soporte para la internacionalización del contenido: Ofrece una alto rendimiento gracias a la implementación de un sistema de caché
- Integración con Spring: Fácil de utilizar y de integrar al sólo proporcionar atributos que enriquecen el marcado sin incorporar tags nuevos.
- Soporte para la evaluación de expresiones: OGNL (módulo estándar) o SpEL (Spring).

ENCONTRAR PLANTILLAS

El primero de los pasos es conseguir un TemplateResolver. Este interfaz es la base para que Thymeleaf sea capaz de encontrar la plantilla que queremos usar, incluso si esta se encuentra almacenada en la caché.

Tendremos la posibilidad de indicar:

- La ruta exacta al directorio que contiene nuestras plantillas (suffix).
- El modo de trabajo que deseamos: XML, VALIDXML, XHTML, VALIDXHTML, HTML5 o LEGACYHTML5.
- Si la caché está activa o no
- El tiempo de cacheo de las plantillas en milisegundos.

EXPRESIONES Y EL LENGUAJE OGNL

Thymeleaf soporta OGNL (Object-Graph Navigation Language) para la definición de expresiones y como forma de acceder a las variables que le proporcionamos a la plantilla como entrada.

Este es un lenguaje que, en el contexto de Thymeleaf, simplifica y flexibiliza la creación de plantillas.

BUCLES

Con Thymeleaf podemos iterar de forma muy sencilla sobre una colección que recibimos como parámetro.

CONCLUSIONES

Thymeleaf es un motor de procesamiento muy flexible, adaptado a las necesidades actuales, poco intrusivo en el marcado y muy muy rápido.

3.3 Semantic UI

Semantic UI es un framework para crear el diseño de interfaces de manera responsive utilizando HTML/CSS legible. Empezó su andadura en 2013 y actualmente va por la versión 2.2. Viene integrado con otros frameworks o librerías como son Angular, React, Ember o Meteor.

COMPONENTES:

- UI Element: un bloque de construcción básico. Puede aparecer solo o en grupo.
- UI Collection: es un grupo de diferentes tipos de elementos que son inter-dependientes.
- UI View: representa una pieza común de contenido del sitio web.
- UI Module: es un componente con funcionalidad interactiva basada en JavaScript, como son los accodions, modales etc.
- UI Behaviour: es un componente que no puede existir de forma independiente, sino que se utiliza para inyectar funcionalidad en otros componentes.

¿QUÉ DIFERENCIA A SEMANTIC UI?

Lenguaje natural

Una de las grandes diferencias con el resto de frameworks es la utilización de la sintaxis para crear los componentes de manera legible. Las clases usan la sintaxis de los lenguajes naturales como las relaciones sustantivo/modificador, orden de las palabras, y la pluralidad para vincular conceptos intuitivamente.

Las clases en Semantic UI son mucho más intuitivas y adecuadas a nuestro lenguaje, a nuestra manera de hablar.

Reutilización de las clases en diferentes componentes

Se utiliza la misma clase para dar apariencia a diferentes componentes, describiendo sus propiedades.

Javascript intuitivo

Utiliza frases simples, llamados behaviours, para lanzar los eventos de los componentes en javascript.

Componentes prediseñados

Dispone de más de 50 componentes prediseñados que en otros frameworks no disponemos y que nos hacen la vida más fácil a la hora de diseñar dichos componentes.

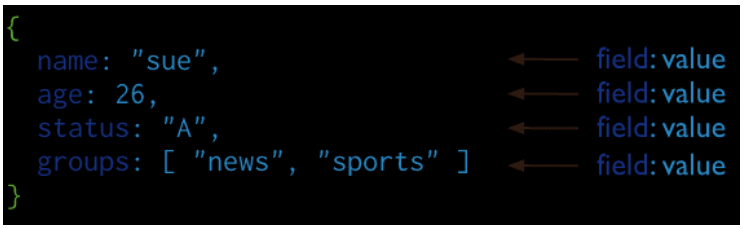
CONCLUSIONES

Como hemos visto, uno de los puntos fuertes de Semantic UI es la sintaxis descriptiva que nos ofrece. Es soportado por los navegadores: Firefox, Chrome y Safari Mac, Internet Explorer 11+, Android 4.4+, Chrome for Android 44+, iOS Safari 7+ y Microsoft Edge 12 & 13.

3.4 Mongo DB

Es una base de datos NoSQL de código abierto orientado a documentos con un alto rendimiento y disponibilidad. Guarda estructuras de datos en documentos similares a JSON con esquema dinámico llamadas BSON.

ESTRUCTURA DE UN DOCUMENTO EN MONGODB.



CARACTERÍSTICAS

- Se almacenan documentos (registros) en colecciones (tablas).
- Se puede utilizar el lenguaje Javascript en el servidor.

ALGUNAS DIFERENCIAS ENTRE SQL Y NOSQL

SQL	*NoSQL*
Estructura definida	No requiere estructura definida
Debe cumplir requisitos de integridad en tipo de dato y compatibilidad	Se puede ejecutar en máquinas con pocos recursos
Atomicidad	Escalabilidad, soportan estructuras distribuidas
Mayor soporte y mejores suites de productos	Permite realizar cambios a los esquemas sin parar la base de datos

¿CUÁNDO UTILIZAR SQL O NOSQL?

- Cuando los datos deben ser consistentes sin dar posibilidad al error. SQL.
- Cuando nuestro presupuesto no se puede permitir grandes máquinas y debe destinarse a máquinas de menor rendimiento. NoSQL.
- Cuando las estructuras de datos que manejamos son variables. NoSQL.
- Análisis de grandes cantidades de datos en modo lectura. NoSQL.

¿QUIÉNES USAN MONGODB?

- Google
- Facebook
- CISCO
- Ebay
- Linkedin

3.5 Librería Lombok

Cuando se trabaja con Java siempre se agradece que Eclipse, NetBeans e IntelliJ ofrezcan un conjunto de refactorings potente que evite repetir las tareas más tediosas. Quizás la tarea más aburrida de realizar es añadir getter/setter y constructores a una clase a través de las herramientas de refactoring. ¿No sería más cómodo y más limpio no tener que escribirlos? Esto es de lo que se encarga la librería Lombok.

Con un ejemplo lo veremos más claro:

Supongamos que se dispone de la clase Persona

```
 public class Persona {

        private String nombre;
        private String apellidos;
        private int edad;
}
```

Esta clase no es operativa y hay que usar las diferentes herramientas de generación de código para convertirla en esto:

```
public class Persona {

        private String nombre;
        private String apellidos;
        private int edad;
        public String getNombre() {
            return nombre;
        }
        public void setNombre(String nombre) {
            this.nombre = nombre;
        }
        public String getApellidos() {
            return apellidos;
        }
        public void setApellidos(String apellidos) {
            this.apellidos = apellidos;
        }
        public int getEdad() {
            return edad;
        }
        public void setEdad(int edad) {
            this.edad = edad;
        }
        public Persona(String nombre, String apellidos, int edad) {
            super();
            this.nombre = nombre;
            this.apellidos = apellidos;
            this.edad = edad;
        }
    }
```

No lleva mucho tiempo hacerlo, pero sí que aumenta de forma importante el código con el que trabajar.

Lombok es una librería que usa un conjunto de anotaciones reducido y permite solventar esta situación de una forma elegante. Lo primero que hay que hacer es incluir el JAR de lombok en el proyecto con el que se está trabajando. Realizada esta operación hay que configurar el entorno de desarrollo para que pre procese las anotaciones, en nuestro caso Eclipse.

Finalizado este paso podemos definir la clase Persona apoyándonos en las anotaciones de Lombok.

```
import lombok.Data;

public @Data class Persona {

    private String nombre;
    private String apellidos;
    private int edad;


}
```

Ya no será necesario añadir para nada los métodos set/get o los constructores, lombok los añadirá de forma dinámica y la clase Persona funcionará sin problemas.

Capítulo 4. Planificación y Presupuesto inicial

4.1 Adaptación de procesos PMBOK al TFM

Hemos decidido realizar una versión del PMBOK simplificada y adaptada a una empresa de pequeño tamaño con proyectos no muy complejos de gestionar.

La visión general del flujo del proyecto se dividiría en cuatro etapas. Inicio, Planificación, Ejecución y control y por último, el cierre del proyecto. La planificación, mediante las tareas de control podrá ser actualizada y permitir la mejora en las distintas facetas a controlar: evolución de trabajo, riesgos, calidad, etc.

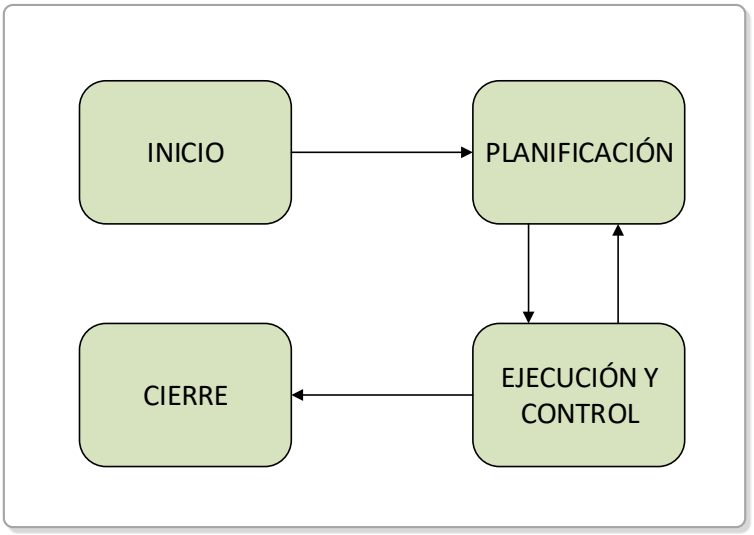


Figura 4.1. Diagrama de procesos PMBOK simplificado

4.2 Planificación

En la tabla siguiente se puede observar la planificación inicial realizada para el proyecto. En dicha tabla podemos observar en negrita y cursiva las Tareas resumen de otras tareas, así como el tiempo estimado y las fechas de inicio y fin de cada una de ellas. También podemos observar los hitos marcados en cursiva y subrayado. Dentro de la tarea de ejecución del proyecto, cada módulo queda reflejado como tarea resumen, sin incluir sus subtareas para no incluir demasiada información que no aporta excesiva información a la tabla. Simplemente citar que cada módulo constará de las tareas siguientes: análisis, diseño, implementación, pruebas y finalmente, posterior a la entrega, seguimiento del proyecto.

TAREA	TIEMPO	INICIO	FIN
Aplicación web Challenge Memory	**51,2 días**	**02/12/17**	**27/05/18**
Planificación			
Análisis del alcance del proyecto y toma de decisiones técnicas	10 horas	02/12/17	02/12/17
Identificación de tareas	6 horas	09/12/17	09/12/17
Elaboración del cronograma y presupuesto	10 horas	10/12/17	10/12/17
Análisis de riesgos	8 horas	16/12/17	16/12/17
Entrega del presupuesto			
Instalación de servidores de aplicación y BD	4 horas	16/12/17	17/12/17
Instalación de repositorio de software	4 horas	17/12/17	17/12/17
Ejecución del proyecto			
Análisis de la arquitectura del proyecto	5 horas	17/12/17	23/12/17
Construcción de la arquitectura del proyecto	10 horas	23/12/17	24/12/17
Diseño de la interfaz de la aplicación	10 horas	24/12/17	30/12/17
Módulo de inicio de la aplicación	2,2 días	30/12/17	06/01/18
Entrega del módulo			
Módulo de registro de usuarios	1,5 días	06/01/18	07/01/18
Entrega del módulo			
Módulo de acceso de usuarios	1,5 días	07/01/18	14/01/18
Entrega del módulo			
Módulo de resumen de resultados	2,3 días	14/01/18	21/01/18
Entrega del módulo			
Módulo de edición de perfil de usuario	1,7 días	21/01/18	28/01/18
Entrega del módulo			
Módulo de vista global de datos de usuario	1,1 días	28/01/18	03/02/18
Entrega del módulo			
Módulo de inserción de resultados	2,4 días	03/02/18	10/02/18
Entrega del módulo			
Módulo de clasificación virtual de competición	2,6 días	10/02/18	18/02/18
Entrega del módulo			
Módulo de vista global de un resultado	1,3 días	18/02/18	24/02/18
Entrega del módulo			
Módulo de aviso de información falsa	1,1 días	24/02/18	25/02/18
Entrega del módulo			
Módulo de búsqueda: competiciones y atletas	3,2 días	25/02/18	10/03/18

Entrega del módulo			
Módulo de evolución de atleta en una competición	3,8 días	11/03/18	24/03/18
Entrega del módulo			
Cierre del proyecto			
Entrega y puesta en marcha			
Mantenimiento y resolución de incidencias	60 horas	24/03/18	14/04/18
Presentación del Proyecto	**2 horas**	**10/07/18**	**09/07/18**

Como podemos observar, el proyecto dará inicio el 2 de diciembre de 2017 y concluirá, con su presentación, el 9 de julio (fecha estimada), con un tiempo invertido entre los diferentes perfiles que dedicarán tiempo al mismo de 51,2 días.

Durante la evolución de los trabajos se ha establecido un calendario de hitos, coincidiendo todos ellos con entregas parciales o finales:

- Entrega del presupuesto el *16 de diciembre de 2017*.
- Entrega del Módulo de inicio de la aplicación el *6 de enero de 2018*.
- Entrega del Módulo de registro de usuarios el *7 de enero de 2018*.
- Entrega del Módulo de acceso de usuarios el *14 de enero de 2018*.
- Entrega del Módulo de resumen de resultados el *21 de enero de 2018*.
- Entrega del Módulo de edición de perfil de usuario el *28 de enero de 2018*.
- Entrega del Módulo de vista global de datos de usuario el *3 de febrero de 2018*.
- Entrega del Módulo de inserción de resultados el *10 de febrero de 2018*.
- Entrega del Módulo de clasificación virtual de competición el *18 de febrero de 2018*.
- Entrega del Módulo de vista global de un resultado el 24 de febrero de 2018.
- Entrega del Módulo de aviso de información falsa el *25 de febrero de 2018*.
- Entrega del Módulo de búsqueda: competiciones y atletas el *10 de marzo de 2018*.
- Entrega del Módulo de evolución de atleta en competición el *24 de marzo de 2018*.
- Entrega y puesta en marcha el *24 de marzo de 2018*.
- Fin de pruebas del sistema y mantenimiento el *14 de abril de 2018*.
- Presentación del Proyecto el *9 de julio de 2018*.

4.3 Análisis de Riesgos

4.3.1 Plan de Gestión de Riesgos

4.3.1.1 Metodología

4.3.1.1.1 IDENTIFICACIÓN

Se deberán analizar los factores que podrían ser amenazados por algún tipo de riesgo, como puede ser: hardware, software y trabajadores.

4.3.1.1.2 ANÁLISIS Y PLANIFICACIÓN

Es importante clasificar los riesgos, en función de su probabilidad, importancia y efectos para definir las acciones pertinentes. La clasificación permiten pensar con mayor amplitud sobre los riesgos que pueden afectar al proyecto dado que disponemos de una lista de posibles riesgos catalogados.

Es importante realizar un análisis cualitativo y cuantitativo de los riesgos más importantes. Estos riesgos han de ser descritos de forma precisa. Debemos describir la situación o suceso del proyecto que el equipo prevé que pueda afectar negativamente o causar una reducción de beneficios. Debemos identificar, bajo esas circunstancias que consecuencias tendría para nuestro proyecto.

Otra tarea importante en nuestra metodología será el análisis y prioridad de los riesgos Con este análisis de riesgos conseguiremos establecer las prioridades de los riesgos y determinar cuál de ellos justifica la reserva de recursos. Por otro lado la asignación de prioridades a los riesgos permitirá tratar en primer lugar los riesgos más importantes del proyecto.

Debemos realizar una estimación de la probabilidad del riesgo. De esta manera se calcula la probabilidad de que la situación descripta llegue a producirse de verdad. Se emplearán las categorizaciones expresadas en lenguaje natural o estableciendo en un cuadro de referencia.

Realización de la estimación del impacto. Calcular la gravedad de los efectos adversos, la magnitud de las pérdidas o el costo si el riesgo llega a producirse dentro del proyecto.

4.3.1.1.3 MONITORIZACIÓN Y CONTROL

Para conseguir la gestión y supervisión de los riesgos mencionados anteriormente, se elaborará un Plan de Acción y un Plan de Contingencias para cada uno de los riesgos identificados como críticos.

Con el Plan de Acción se buscarán acciones preventivas, de manera que se minimice la probabilidad de que un riesgo ocurra y mitigar el impacto que el mismo podría ocasionar en el proyecto, encarando los problemas en forma proactiva.

El Plan de Contingencia, sim embargo, irá encaminado a dar respuestas rápidas para mitigar los efectos en caso de que los riesgos se concreten. Este plan, además definirá ciertos indicadores

que permitirán poner en marcha las acciones previstas, es decir, en caso que se verifiquen ciertos disparadores se adoptarán las medidas indicadas.

4.3.1.1.4 CONCEPTOS GENERALES

Con el fin de evitar la ambigüedad, se deben definir tres conceptos utilizados en la Gestión de Riesgos:

- Riesgo: es cualquier evento que pueda causar un efecto en el proyecto. El efecto bien podría ser una amenaza o podría ser una oportunidad. La primera de ellas debe ser evitada y la segunda debe ser explotada.
- Factor de riesgo: Los factores de riesgo son circunstancias asociadas con el riesgo de que aumenta la posibilidad de conseguir los efectos descritos por el riesgo. Por lo general, los riesgos son imposibles o difíciles de medir o controlar directamente y las decisiones se toman controlando los factores de riesgo con el fin de evitar o minimizar (o potenciar) los efectos de riesgo.
- Indicadores de Riesgo: Los indicadores de riesgo son elementos asociados al riesgo y / o sus factores, que se puede medir y sirve como información acerca de la posibilidad de que se produzca el efecto del riesgo.

4.3.1.2 Herramientas y tecnología

- Opinión de los miembros del equipo, con más peso en función de la responsabilidad el mismo.
- Tormenta de ideas.
- Consulta de opiniones expertas.
- Lista de Riesgos clasificados y priorizados.
- Análisis de los Riesgos críticos.
- Cálculos probabilísticos.

4.3.1.3 Roles y responsabilidades

Jefe de Proyecto: Responsable de identificación, priorización y seguimiento de riesgos y proponer acciones para afrontar los riesgos identificados

Equipo de Trabajo: Responsable asesoramiento de riesgos, identificación de los riesgos.

4.3.1.4 Presupuesto

El presupuesto de contingencia asignado para riesgos es del 5% del coste total del proyecto, ya que no es posible asumir una cantidad mayor para finalizar el proyecto con beneficios.

4.3.1.5 Calendario

Se ha definido un calendario en el que se realizará un estudio de los Riesgos del Proyecto en su primera fase, durante 8 horas.

Posteriormente, el Jefe de Proyecto mantendrá un seguimiento al final de la entrega de cada módulo. Posteriormente, valorará el resultado de los indicadores de riesgos y su posible activación.

4.3.1.6 Definiciones de probabilidad

Probabilidad muy baja	Menos del 10% El valor usado en la matriz de probabilidad e impacto es: 5%
Probabilidad baja	Entre el 10 y el 30% El valor usado en la matriz de probabilidad e impacto es: 20%
Probabilidad media	Entre el 30 y el 60% El valor usado en la matriz de probabilidad e impacto es: 45%
Probabilidad alta	Entre el 60 y el 80% El valor usado en la matriz de probabilidad e impacto es: 70%
Probabilidad muy alta	Más del 80% El valor usado en la matriz de probabilidad e impacto es: 90%

4.3.1.7 Definiciones de impacto por objetivos

Condiciones definidas para las escalas de impacto de un riesgo sobre los objetivos principales del proyecto (Se muestran ejemplos para impactos negativos únicamente)					
Objetivos de proyecto	Escalas relativas o numéricas				
	Muy bajo / 5%	Bajo / 10%	Moderado / 20%	Alto / 40%	Muy alto / 80%
Coste	Incremento del coste insignificante	Incremento del coste <10%	Incremento del coste entre el 10-20%	Incremento del coste entre el 20-40%	Incremento del coste >40%
Tiempo	Incremento de tiempo insignificante	Incremento de tiempo <5%	Incremento de tiempo entre el 5-10%	Incremento de tiempo entre el 10-20%	Incremento de tiempo >20%
Alcance	Reducciones del alcance inapreciables	Afectadas áreas poco importantes del alcance	Afectadas áreas importantes del alcance	Reducciones del alcance inaceptables para el cliente	El resultado final del proyecto no es realmente útil
Calidad	La degradación de la calidad es inapreciable	Sólo las aplicaciones muy exigentes se ven afectadas	La reducción de la calidad requiere la aceptación del cliente	Reducción de la calidad inaceptable para el cliente	El resultado final del proyecto no es realmente útil
Esta tabla presenta ejemplos de definiciones impacto de riesgo para cuatro obejtivos de proyecto diferentes. Debe ser ajustado en el proceso de elaboración del Plan de Gestión de Riesgos a cada proyecto concreto y a lo sumbrales de riesgo de la organización. Las definiciones del impacto deben ser desarrolladas para los riesgos positivos (oportunidades) de una manera similar.					

4.3.1.8 Matriz de probabilidad e impacto

Muy Alta	***0.90***	0.05	0.09	0.18	0.36	0.72
Alta	***0.70***	0.04	0.07	0.14	0.28	0.56
Media	***0.45***	0.02	0.08	0.09	0.18	0.36
Baja	***0.20***	0.01	0.05	0.04	0.08	0.16
Muy baja	***0.05***	0.00	0.01	0.01	0.02	0.04
		0.05	***0.10***	***0.20***	***0.40***	***0.80***
		Muy bajo	**Bajo**	**Medio**	**Alto**	**Crítico**

4.3.1.9 Niveles de tolerancia

Se establece un nivel de tolerancia de 0.5

4.3.1.10 Plan de contingencia

Después de identificados los riesgos del proyecto, establecidos los diferente indicadores a valorar y controlar. En caso de observar que un riesgo pasa a estar activo, se procederá a iniciar el Plan de Contingencia.

Este Plan irá encaminado a paliar el problema detectado, de manera que se aprobarán actuaciones formativas, contrataciones temporales, adquisición de material necesario, o cualquier cuestión que permita controlar o mitigar el efecto del riesgo detectado.

4.3.2 Registro de riesgos

ID	Nombre del Riesgo	Responsable	Impacto	0,50 Prioridad	Respuesta
1	Afrontar tecnología de desarrollo nueva para el equipo desarrollador (Spring Boot)	Jefe de Proyecto	0,81		Mitigar
2	Retraso en la entrega del servidor de desarrollo	Jefe de Proyecto	0,17		Retener
3	Problemas en la configuración del servidor de desarrollo	Analista	0,28		Retener
4	Poca implicación del cliente puede provocar no atender consultas sobre el desarrollo.	Jefe de Proyecto	0,50		Retener
5	Servidor de desarrollo queda fuera de servicio	Analista	0,27		Retener
6	Nueva tecnología es menos productiva de lo esperado	Jefe de Proyecto	0,17		Retener

4.3.3 Análisis de riesgos críticos

<table>
<tr><td>ID:
1</td><td colspan="6">Nombre:
Afrontar tecnología de desarrollo nueva para el equipo desarrollador (Spring Boot)</td></tr>
<tr><td colspan="7">Descripción:
Los plazos de entrega estimados han sido ajustados para un desarrollo ágil como propone la documentación de la tecnología elegida, Spring Boot. No obstante el desarrollador no conoce la tecnología directamente, aunque viene del mundo del J2EE y se cree que su adaptación será breve. Dada esta circunstancia, se estima que la demora en la fase inicial será compensada con la mejora de rendimiento que la nueva tecnología aporta en pocas jornadas de trabajo. Es altamente probable que la adaptación no sea tan rápida como se prevé, con lo que el plazo de entrega se verá altamente comprometido, y esto generará un desajuste económico en el proyecto, que no puede asumir retrasos.</td></tr>
<tr><td colspan="7">Categoría(s) de riesgo:
Tecnología, planificación, recursos</td></tr>
<tr><td>Status:
Activo</td><td colspan="6">Causas del Riesgo (Identificación de los factores del riesgo):
• Que el desarrollador no sean capaz a adaptarse con rapidez a la nueva tecnología
• Que la tecnología sea más compleja de lo esperado
• Que la tecnología no tenga suficiente documentación online disponible.</td></tr>
<tr><td rowspan="2">Probabilidad</td><td colspan="4">Impacto</td><td rowspan="2">Impacto Total</td><td rowspan="2">Respuestas</td></tr>
<tr><td>Presupuesto</td><td>Planificación</td><td>Alcance</td><td>Calidad</td></tr>
<tr><td>Alta</td><td>Bajo</td><td>Crítico</td><td>Bajo</td><td>Alto</td><td>0,81</td><td>Diseñar una estrategia proactiva de evitación. Controlar los tiempos del desarrollo y la evolución de los desarrolladores con la nueva tecnología.</td></tr>
</table>

<table>
<tr><td colspan="2">Riesgos derivados de éste:
Que el proyecto no puede ser presentado a tiempo</td></tr>
<tr><td>Plan de Contingencia:
1. Controlar la desviación entre el cronograma planificado y la evolución real.
a. Se controla que la desviación no excede del plazo para presentar el proyecto.
2. Conseguir recursos online gratuitos para que el equipo los tenga disponibles.
a. Si no es posible, adquirir libros avanzados sobre la tecnología por un importe no superior a 100€.
3. Buscar formación presencial avanzada e intensiva (10 horas) para impartir en dos jornadas.
a. En esta circunstancia, no superar la inversión de 400€.</td><td>Presupuesto para contingencias:
500€, calculados como suma de los posibles planes de contingencia a afrontar. Se hace la suma por ser una cantidad asumible.
Planificación temporal de las contingencias:
El Jefe de Proyecto hará un seguimiento y valorará el resultado de los indicadores de riesgos y su posible activación.</td></tr>
</table>

Monitorización (*Descripción de la relación de los factores de riesgo y los indicadores y descripción de los planes de monitorización***):**

Se monitorizan tres indicadores que se corresponden con los factores de riesgo.

Por un lado se controlará con exhaustividad el avance de las tareas, de tal modo que monitorizaremos las horas de desviación existentes. Dado el límite establecido para la presentación del proyecto, daremos valores de riesgo desde 0 a 20 horas, donde 20 es el valor máximo.

Otro indicador que controlaremos será el número de veces que el desarrollador necesita detener su actividad por la necesidad de consultar como realizar una tarea (la tecnología es más compleja de lo esperado). Estimamos que un valor límite sería el de 15 consultas por jornada de trabajo. Distribuiremos los valores desde 0 a 15.

Por último, se monitorizarán las interrupciones de los desarrolladores para realizar consultas que superen los 10 minutos de tiempo sin resultado satisfactorio (poca documentación online sobre la tecnología). Dada la incidencia directa en la planificación se estima que este tipo de sucesos no debería de producirse más de 2 veces por jornada.

Los indicadores medidos se valoran:

	Valor del riesgo				
INDICADORES	***Muy alto***	***Alto***	***Medio***	***Bajo***	***Muy Bajo***
Indicador 1: Desviación en horas de la programación prevista	20,00	15,00	10,00	5,00	0,00
Indicador 2: Número de consultas/jornada para realizar una tarea	15,00	12,00	8,00	4,00	0,00
Indicador 3: Interrupciones de más de 10 minutos por consulta	2,00		1,00		0,00

El riesgo se valorará como el más alto valor de cualquier indicador.

Indicadores (*Descripción de los indicadores, del modo de evaluación y de la consolidación de indicadores si existe.***):**

Indicador 1: Desviación en horas de la programación prevista • Horas que la progresión de las tareas se desvía de lo planificado.	**Evaluación:** Se evalúa en el seguimiento del proyecto
Indicador 2: Número de consultas/jornada para realizar una tarea • Número de veces que un desarrollador interrumpe su actividad para consultar como realizar una tarea.	**Evaluación:** Se evalúa en el seguimiento del proyecto
Indicador 3: Interrupciones de más de 10 minutos por consulta • Interrupciones de los desarrolladores para consultar, que después de 10 minutos no obtienen resultado satisfactorio.	**Evaluación:** Se evalúa en el seguimiento del proyecto

4.4 Presupuesto inicial

A continuación se expone una tabla con el desglose del presupuesto inicial del proyecto. Dicho presupuesto refleja los gastos internos que supondrá la elaboración del proyecto y que permitirá una visión más concreta del coste del mismo y si es asumible su elaboración.

CONCEPTO	CANTIDAD	COSTE
Análisis del alcance del proyecto y toma de decisiones técnicas	10	800,00 €
Identificación de tareas	6	480,00 €
Elaboración del cronograma y presupuesto	10	1.000,00 €
Análisis de riesgos	8	800,00 €
Entrega del presupuesto	1	100,00 €
Instalación de servidores de aplicación y BD	4	200,00 €
Instalación de repositorio de software	4	200,00 €
Analisis de la arquitectura del proyecto	5	400,00 €
Construcción de la arquitectura del proyecto	10	500,00 €
Diseño de la interfaz de la aplicación	10	300,00 €
Módulo de inicio de la aplicación	22	1.440,00 €
Módulo de registro de usuarios	15	970,00 €
Módulo de acceso de usuarios	15	970,00 €
Módulo de resumen de resultados	23	1.490,00 €
Módulo de edicion de perfil de usuario	17	1.130,00 €
Módulo de vista global de datos de usuario	11	770,00 €
Módulo de inserción de resultados	24	1.540,00 €
Módulo de clasificación virtual de competición	26	1.640,00 €
Módulo de vista global de un resultado	13	870,00 €
Módulo de aviso de información falsa	11	770,00 €
Módulo de búsqueda: competiciones y atletas	32	2.000,00 €
Módulo de evolución de atleta en una competición	38	2.360,00 €
Entrega y puesta en marcha	2	200,00 €
Mantenimiento y resolución de incidencias	60	3.000,00 €
Presentación del Proyecto	2	200,00 €
Servidor Web y de BD	1	1.000,00 €
	SUBTOTAL	25.130,00 €
	IVA (21%)	5.277,30 €
	TOTAL	**30.407,30 €**

El presupuesto está desglosado en bloques que permitan observar rápidamente el coste de las tareas esenciales a ejecutar, siendo conscientes de que en la planificación ya se definió con más detalle la subdivisión que cada una de ellas implica para la organización.

Capítulo 5. Análisis

5.1 Requisitos del Sistema

5.1.1 Obtención de los Requisitos del Sistema

A continuación se enumeran los requisitos funcionales.

Código	Nombre Requisito	Descripción del Requisito
R01	Ver página principal	Se mostrará la página de inicio de la aplicación o la página de inicio del usuario si este esta autenticado
R02	Registrar	Se creará un usuario inactivo después de la validación de datos del usuario y se enviará un mail de confirmación
R03	Confirmar registro	A través del enlace suministrado mediante un email, el proceso de confirmación activará el usuario
R04	Autenticar	Tras la inserción del email y clave, el usuario accederá a su zona privada
R05	Editar perfil y privacidad	Tras la validación de datos, el usuario modificará sus datos y privacidad
R06	Crear competición	Tras la validación de datos, los datos de una competición serán guardados
R07	Crear edición de competición	Tras la validación de datos, los datos de una edición de competición serán guardados
R08	Crear resultado	Tras la validación de datos, los datos un resultado serán almacenados
R09	Editar resultado	Tras la validación de datos, los datos de un resultado serán modificados
R10	Borrar resultado	Tras la confirmación de la acción. Se borrará el resultado
R11	Ver resultado	Se mostrarán todos los datos de un resultado
R12	Ver usuario	Se mostrarán todos los resultados de un usuario
R13	Añadir usuario amigo	El usuario seleccionado será añadido a la lista de amigos del usuario ejecutor de la acción
R14	Eliminar usuario amigo	El usuario seleccionado será eliminado de la lista de amigos del usuario ejecutor de la acción
R15	Avisar resultado falso	Tras la validación de datos, se almacenará información de notificación de un resultado falso
R16	Ver competición	Mostrará la clasificación de la última edición registrada de la competición y un desplegable para ver ediciones anteriores
R17	Ver clasificación de edición	Mostrará una clasificación de todos los resultados introducidos para la edición

		ordenados por tiempo realizado en la competición
R18	Ver evolución competición	Generará una gráfica donde se verá la evolución de los tiempos del usuario en las diferente ediciones disputadas
R19	Dar "Me gusta"	Marca el resultado como “me gusta” para el usuario ejecutante
R20	Quitar "Me gusta"	Quita del resultado el “me gusta” registrado en algún momento previo
R21	Comentar resultado	Se almacena un comentario asociado a un resultado
R22	Buscar competición	Mediante un texto de búsqueda, se muestran las competición con nombre coincidente
R23	Buscar usuario	Mediante un texto de búsqueda, se muestran los usuario con nombre coincidente
R24	Listar resultados con aviso	Un administrador verá un listado de avisos de resultado falso
R25	Desactivar usuario	Un administrador, tras la confirmación de la acción, desactivará al usuario seleccionado

A continuación se enumeran los requisitos tecnológicos.

Código	Nombre Requisito	Descripción del Requisito
R26	Servidor de aplicaciones con versión Java 8	La versión mínima de la que dispondrá el servidor de aplicaciones será Java 8
R27	Base de datos Mongo DB	El sistema utilizará una base de datos No SQL Mongo DB.
R28	Internacionalización	La aplicación tendrá la posibilidad de internacionalización, mostrando los contenidos en el idioma configurado en el navegador que accede, si se encontrara disponible.

A continuación se enumeran los requisitos de seguridad.

Código	Nombre Requisito	Descripción del Requisito
R29	Comunicación a través del protocolo seguro SSL	Será necesario que las comunicaciones del usuario con el servidor se establezcan a través del protocolo seguro SSL

5.1.2 Identificación de Actores del Sistema

A continuación se enumeran los actores del sistema

Actor	Descripción
Usuario	Sujeto que utiliza la aplicación para el almacenamiento de información personal e interrelacionarse con otros usuarios
Administrador	Sujeto que tiene una especial consideración en el sistema y gestiona ciertos aspectos del mismo, con funcionalidad restringida. Es considerado un usuario más de la plataforma.

5.1.3 Especificación de Casos de Uso

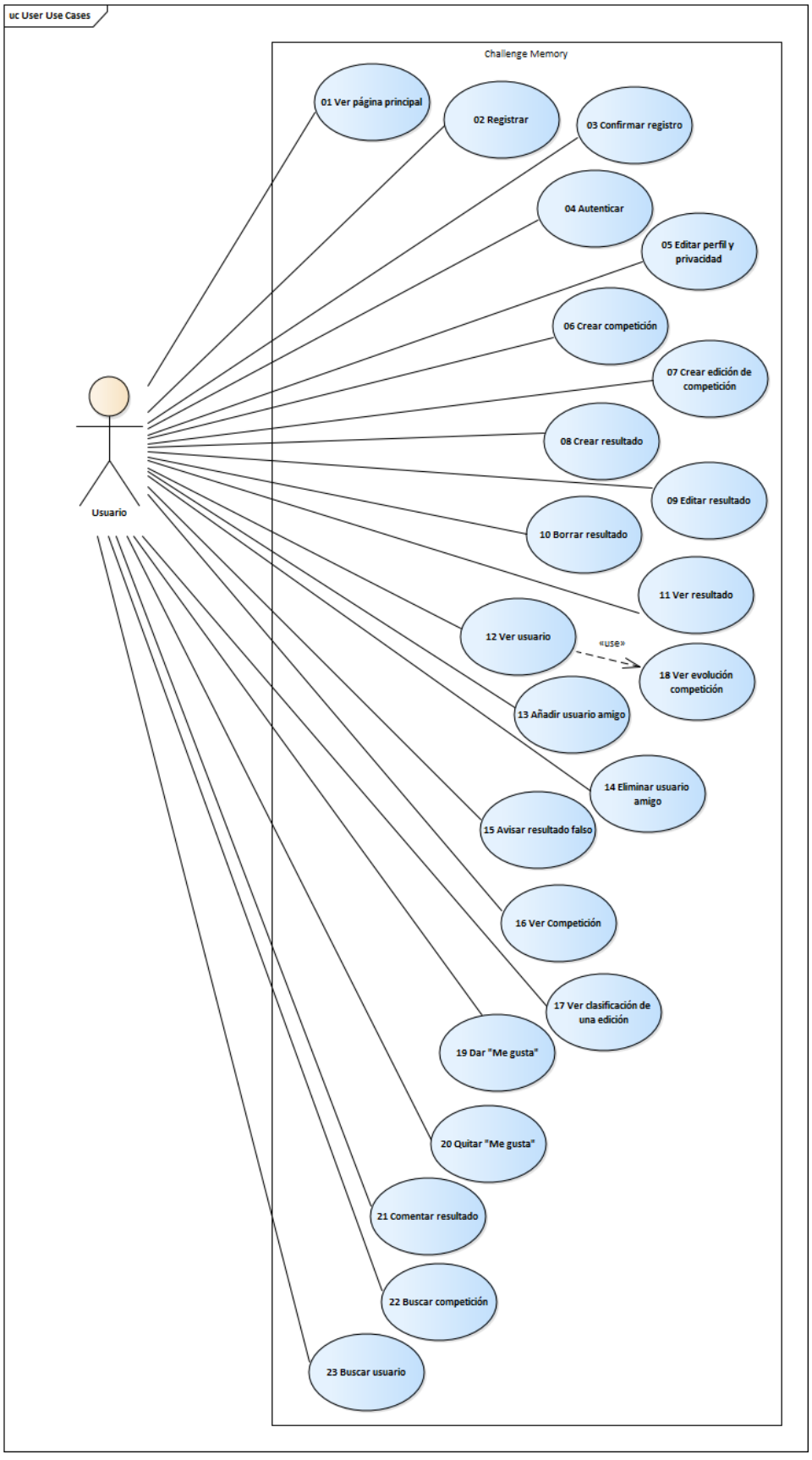


Figura 5.1. Diagrama de casos de uso del actor Usuario

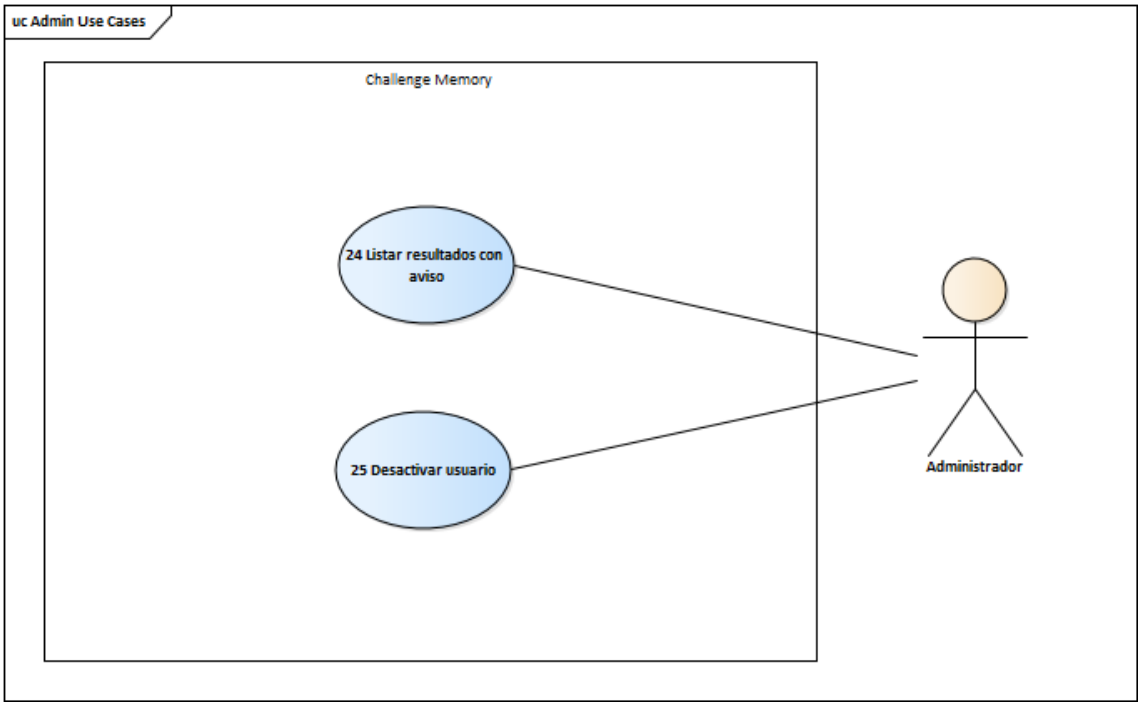


Figura 5.2. Diagrama de casos de uso del actor Administrador

Nombre del Caso de Uso
Ver página principal
Descripción
Se mostrará la página inicial pública de la aplicación en el caso de que el usuario no esté registrado o autenticado. Si el usuario está autenticado, se muestra su página de inicio personal, donde se muestra un listado ordenado por fecha de publicación de todos sus resultados y de sus amigos

Nombre del Caso de Uso
Registrar
Descripción
Tras introducir sus datos personales, el sistema almacenará los datos del usuario en estado inactivo, y procederá a enviar un email para la confirmación del mismo.

Nombre del Caso de Uso
Confirmar registro
Descripción
A través de un enlace suministrado en el email generado tras el Registro, el usuario accederá a una URL única que confirmará el registro, pasando a ser un usuario activo de la plataforma.

Nombre del Caso de Uso
Autenticar
Descripción
A través de una página que solicitará el email del usuario y la clave, se procederá a almacenar el usuario en sesión y darle acceso a su página inicial personal, donde se muestra un listado ordenado por fecha de publicación de todos sus resultados y de sus amigos

Nombre del Caso de Uso
Editar perfil y privacidad
Descripción
Se mostrará una página con los datos personales completos del usuario, un control de activado/desactivado para indicar la privacidad de la cuenta (privado o público) y un listado de amigos, si el usuario tiene privacidad de tipo público.

Nombre del Caso de Uso
Crear competición
Descripción
Cuando un usuario procede a registrar un resultado en la plataforma, este ha de estar asociado a una edición de una competición, que a su vez está asociado a una competición. Si en el proceso de búsqueda no arroja resultados, el usuario podrá dar de alta una nueva competición, que podrá ser de cuatro tipos: • Atletismo • Ciclismo • Natación • Triatlón

Nombre del Caso de Uso
Crear edición de competición
Descripción
Cuando un usuario procede a registrar un resultado en la plataforma, este ha de estar asociado a una edición de una competición. Si en el proceso de búsqueda no arroja resultados, el usuario podrá dar de alta una nueva edición de una competición existente. A través de un cuadro de búsqueda, seleccionará la competición asociada y dará de alta los datos de la edición correspondiente.

Nombre del Caso de Uso
Crear resultado
Descripción
Un usuario podrá dar de alta sus resultados deportivos. A través de un formulario, seleccionará mediante un cuadro de texto la edición de la competición asociada. Si esta no existiese, podrá dar de alta una nueva. Igualmente sucederá si no existiese la competición deseada. Tras asociar la edición de la competición, dará de alta el resto de información del resultado, que será almacenado al final del proceso.

Nombre del Caso de Uso
Editar resultado
Descripción
Cuando un usuario acceda a ver un resultado que haya sido dado de alta por él mismo, tendrá la opción de modificarlo a través de un formulario que mostrará toda la información del resultado anteriormente registrado.

Nombre del Caso de Uso
Borrar resultado
Descripción
Cuando un usuario acceda a ver un resultado que haya sido dado de alta por él mismo, tendrá la opción de borrarlo. El sistema pedirá confirmación de la acción antes de completar el proceso.

Nombre del Caso de Uso
Ver resultado
Descripción
Cualquier usuario de la plataforma podrá ver resultados públicos de otros usuarios o suyos, bien sea a través del buscador de competiciones, del perfil de otros usuarios, de su página inicial o de las clasificaciones virtuales de la plataforma. En la página de un resultado se mostrará toda la información relativa al mismo, como: edición de competición, tiempo, puesto, puesto de la categoría, evolución en la competición, fotos…

Nombre del Caso de Uso
Ver usuario
Descripción
La plataforma permitirá siempre el acceso a perfiles públicos. Siempre que se muestre el nombre de un usuario en algún lugar de la aplicación, dicho texto estará enlazado a su perfil público, donde se verán todos los resultados registrados por el usuario, ordenados por fecha de publicación y se mostrará la opción de añadir a dicho usuario al listado de amigos del usuario que lo esté visualizando.

Nombre del Caso de Uso
Añadir usuario amigo
Descripción
Al ver cualquier usuario con perfil público, se habilitará la opción de añadir a este usuario como amigo del usuario que lo esté visualizando en ese momento. Tras completar la acción, el usuario pasará a formar parte de la lista de amigos de usuario ejecutante de la acción, y éste a partir de ese momento comenzará a ver sus resultados en su página de inicio personal.

Nombre del Caso de Uso
Eliminar usuario amigo
Descripción
En la página de edición del perfil de usuario, se mostrará un listado de amigos. En dicho listado se dispondrá de la opción para eliminar al amigo, y dejar de visualizar sus resultados. El sistema pedirá confirmación de la acción antes de completarla.

Nombre del Caso de Uso
Avisar resultado falso
Descripción
Cualquier usuario que visualice un resultado tendrá la opción de enviar aviso de resultado falso sobre el mismo. A través de un formulario, el usuario indicará la anomalía detectada en el mismo.

Nombre del Caso de Uso
Ver Competición
Descripción
A través del buscador de competiciones, se accederá a dichas competiciones. Ver una competición implica mostrar la clasificación virtual de su última edición. Además, se mostrará un desplegable con el resto de ediciones registradas, mediante el cual se podrá ver la clasificación de la edición seleccionada.

Nombre del Caso de Uso
Ver clasificación de una edición
Descripción
En cualquier lugar de la plataforma que se muestre el nombre de una edición de una competición, se habilitará un enlace para acceder a la misma. Dentro se podrá ver una clasificación virtual de todos los resultados registrados para esa edición ordenados por tiempo final. En la tabla que muestra la clasificación, se podrá acceder a cada resultado mediante un enlace sobre el tiempo final del mismo.

Nombre del Caso de Uso
Ver evolución competición
Descripción
El caso de uso “Ver resultado” hace uso de este caso de uso. Cuando se muestre un resultado, se consultará la información de todos los resultados del usuario que dio de alta el mismo en las diferentes ediciones de una competición. Esta información será tratada y mostrada a través de un gráfico que tendrá como eje X los años de la ediciones y como eje Y el tiempo correspondiente

Nombre del Caso de Uso
Dar "Me gusta"
Descripción
En cualquier lugar que se muestre la información de un resultado, cualquier usuario podrá dar “me gusta” al mismo. Esto será visible a través de un corazón. En el caso de que el usuario haya dado “me gusta” en el resultado, el corazón se mostrará rojo, en caso contrario, el corazón será blanco.

Nombre del Caso de Uso
Quitar "Me gusta"
Descripción
En cualquier lugar que se muestre la información de un resultado, cualquier usuario podrá quitar un “me gusta” al mismo. Tras la realización de la acción, el corazón que se mostraba de color rojo, pasa a mostrase se color blanco.

Nombre del Caso de Uso
Comentar resultado
Descripción
En cualquier lugar que se muestre la información de un resultado, cualquier usuario podrá dar escribir un comentario asociado al mismo. A través de un icono al efecto, se accederá a un formulario sencillo, donde se introducirá el texto del comentario. Tras completar la acción, el comentario aparecerá al ver un resultado, mostrando quien los escribió y la hora del mismo.

Nombre del Caso de Uso
Buscar competición
Descripción
Se dispondrá de un buscador sencillo, con un campo de texto que hará búsquedas avanzadas sobre los campos textuales de las diferentes competiciones registradas en la plataforma. Tras introducir el texto y realizar la búsqueda, se presentarán las competiciones coincidentes y se podrá acceder a ver las mismas.

Nombre del Caso de Uso
Buscar usuario
Descripción
Se dispondrá de un buscador sencillo, con un campo de texto que hará búsquedas avanzadas sobre los campos textuales de los diferentes usuarios registrados en la plataforma. Tras introducir el texto y realizar la búsqueda, se presentarán los usuarios coincidentes y se podrá acceder a ver su información.

Nombre del Caso de Uso
Listar resultados con aviso
Descripción
Solo los usuarios de tipo administrador visualizarán una página con un listado de los avisos de resultado falso que se hayan registrado en la aplicación y así poder acceder al mismo y borrarlo si fuera necesario.

Nombre del Caso de Uso
Desactivar usuario
Descripción
Solo los usuarios de tipo administrador podrán desactivar usuarios a través de una opción que se mostrará al ver a dicho usuario.

Identificación de los Subsistemas en la Fase de Análisis

5.1.4 Descripción de los Subsistemas

A continuación se muestra un diagrama con los diferentes subsistemas identificados para el proyecto:

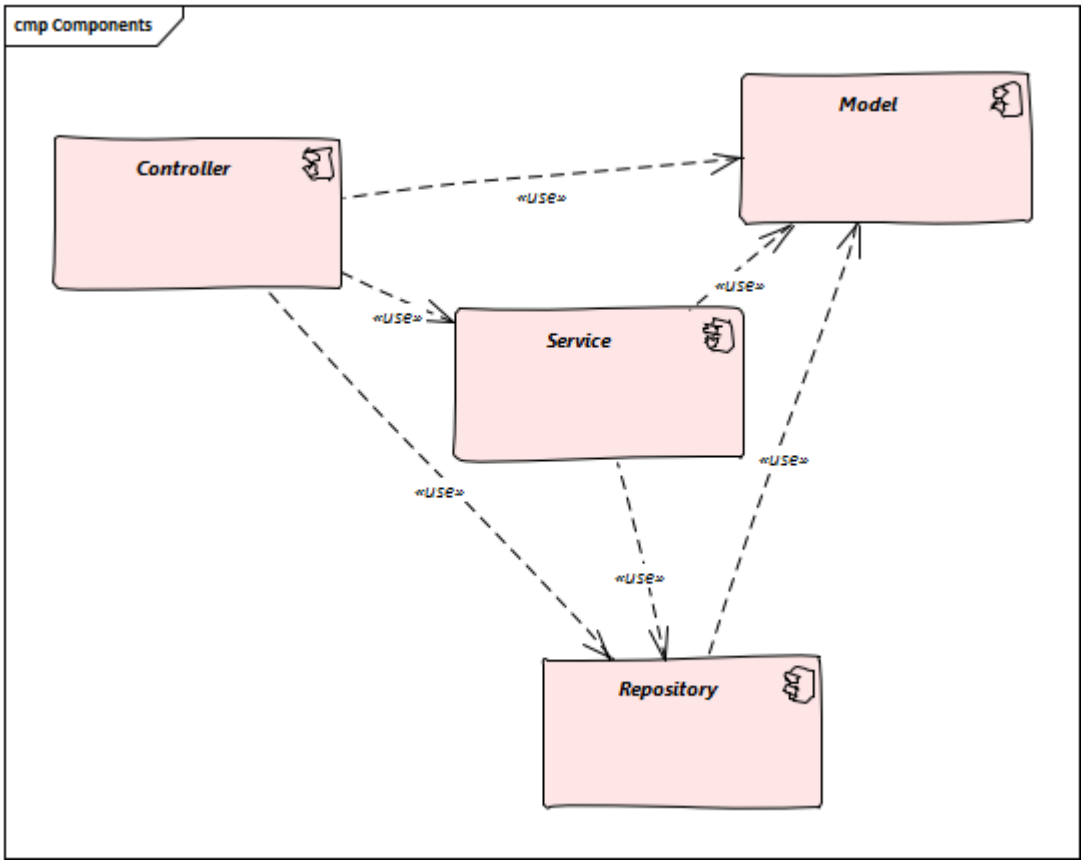


Figura 5.3. Diagrama de subsistemas

5.1.5 Descripción de los Interfaces entre Subsistemas

La comunicación entre los subsistemas reseñados anteriormente se realiza en todo momento de forma local, mediante invocaciones directas.

5.2 Diagrama de Clases Preliminar del Análisis

5.2.1 Diagrama de Clases

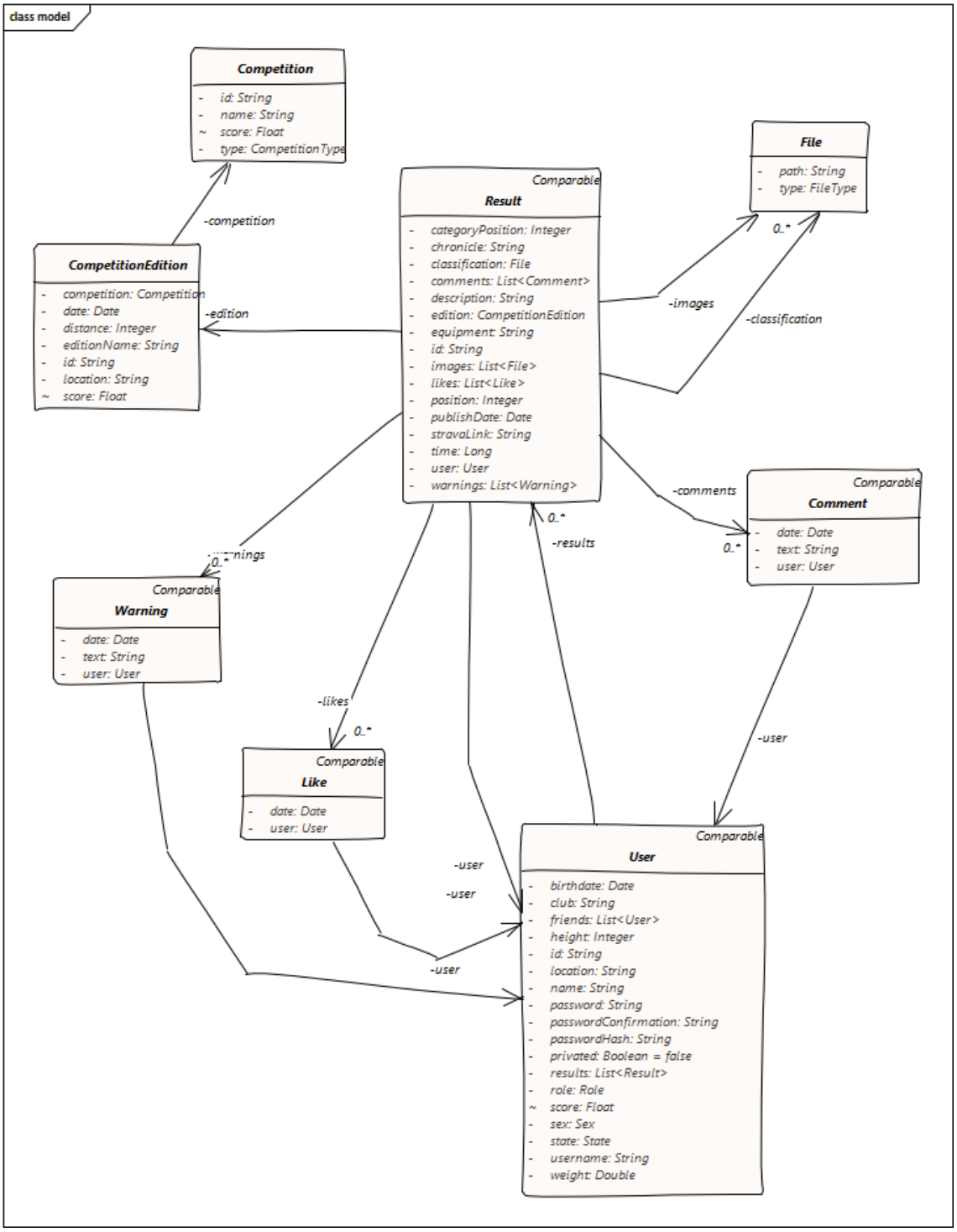


Figura 5.4. Diagrama de clases del subsistema modelo

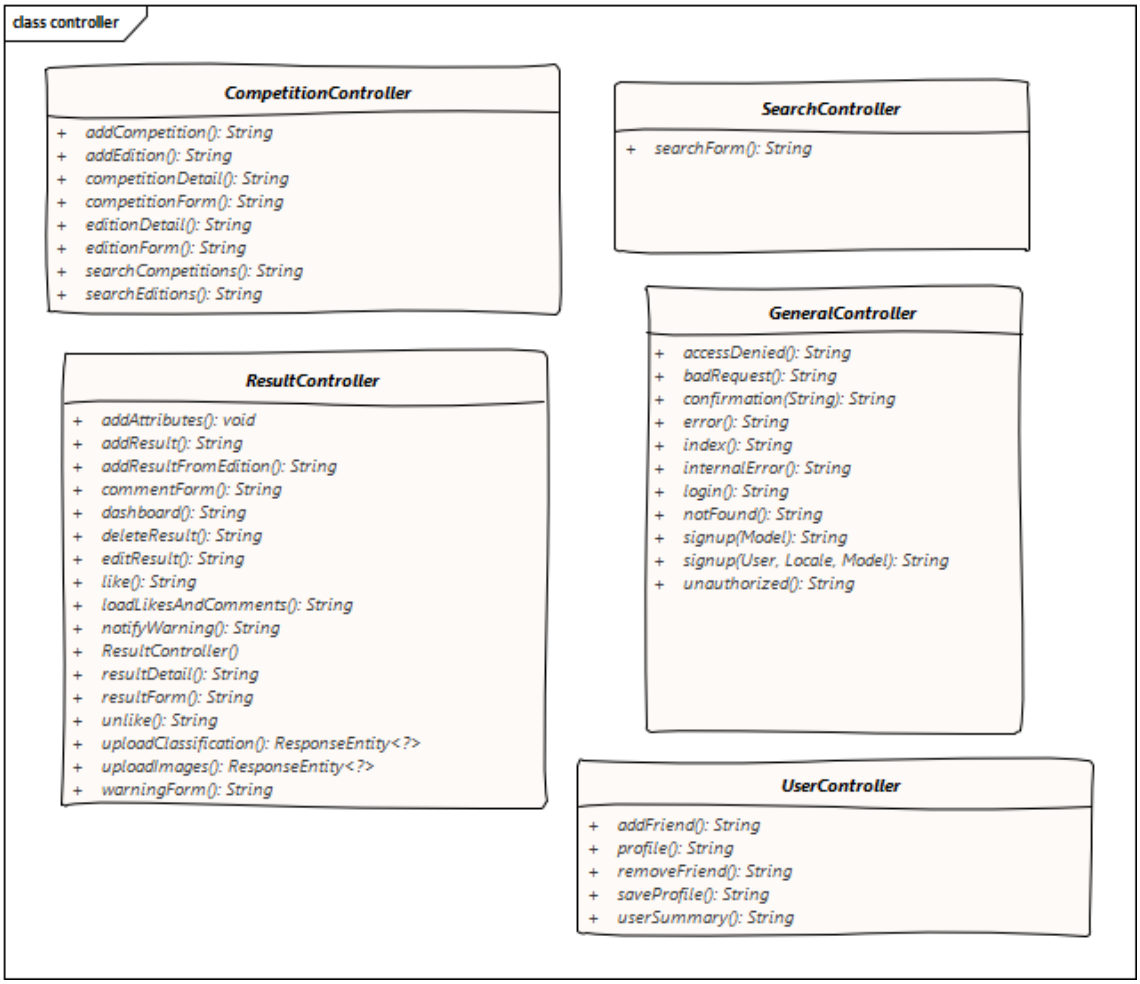


Figura 5.5. Diagrama de clases del subsistema controlador

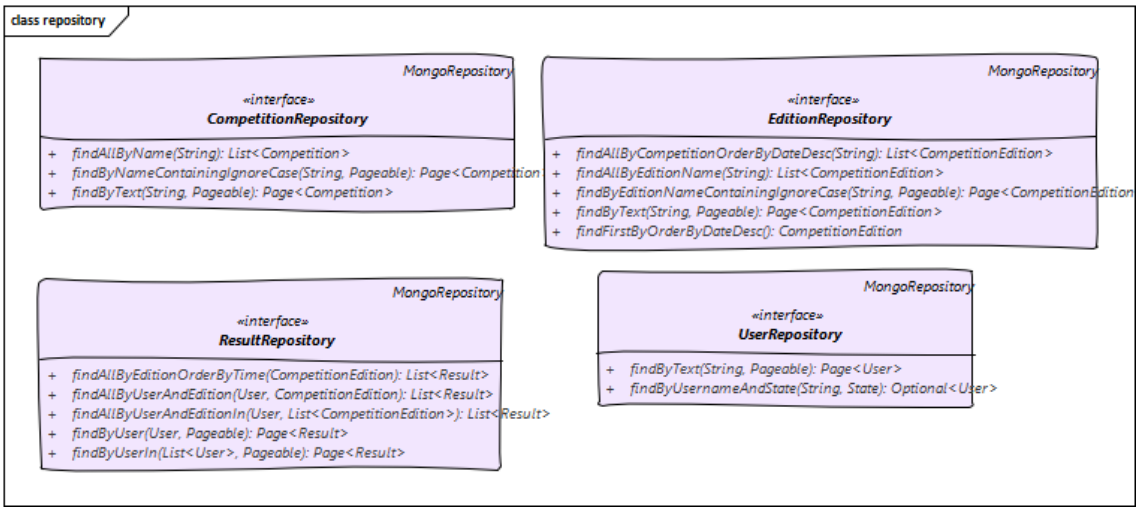


Figura 5.6. Diagrama de clases del subsistema repositorio (acceso BD)

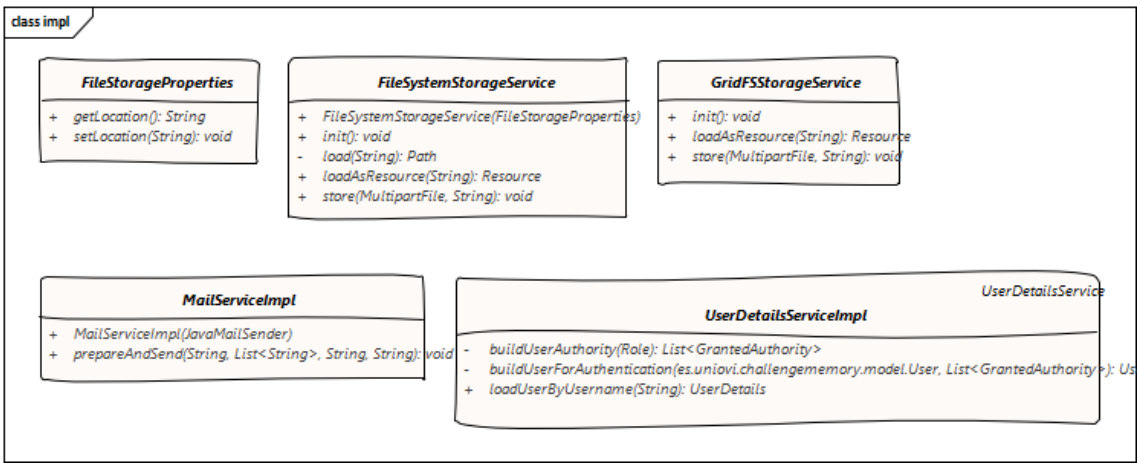


Figura 5.7. Diagrama de clases del subsistema servicios

5.2.2 Descripción de las Clases

A continuación se detallan las clases de cada subsistema:

5.2.2.1 Modelo

Nombre de la Clase
User
Descripción
Clase del modelo asociada a un usuario de la aplicación
Responsabilidades
Almacenar la información de un usuario
Atributos Propuestos
Id: identificador de la clase **username:** nombre del usuario, se corresponde con el email de alta **name:** nombre completo del usuario **birthdate:** fecha de nacimiento **sex:** sexo del usuario **location:** localización del usuario **weight:** peso del usuario **height:** altura del usuario **club:** club del usuario **passwordHash:** clave de acceso cifrada **role:** rol del usuario (usuario o administrador) **state:** estado del usuario (activo o inactivo) **privated:** indentifica si la cuenta es privada o pública (verdadero o falso) **results:** resultados asociados al usuario **friends:** usuarios amigos

Nombre de la Clase
Result
Descripción
Clase del modelo asociada un resultado en una competición
Responsabilidades
Almacenar la información de un resultado
Atributos Propuestos
id: identificador de la entidad **edition:** edición de la competición asociada **time:** tiempo total realizado **position:** posición absoluta **categoryPosition:** posición en la categoría **description:** descripción de la competición **chronicle:** crónica de la competición **equipment:** equipamiento utilizado **stravaLink:** enlace de la actividad en la aplicación Strava **classification:** fichero con la clasificación **publishDate:** fecha de publicación **user:** usuario que dio de alta el resultado **likes:** "me gusta" que recibió el resultado **comments:** comentarios que recibió el resultado **warnings:** avisos de resultado falso que recibió el resultado **images:** images asociadas

Nombre de la Clase
CompetitionEdition
Descripción
Clase del modelo asociada una edición de una competición
Responsabilidades
Almacenar la información de una edición de una competición
Atributos Propuestos
id: identificador de la entidad **competition:** competición asociada **editionName:** nombre completo de la edición (autogenerado con el nombre de la competición y el año de la edición) **distance:** distancia de la edición **date:** fecha de la edición **location:** localización donde trascurrió la edición de la competición **id:** identificador de la entidad

Nombre de la Clase
Competition
Descripción
Clase del modelo asociada a una competición
Responsabilidades
Almacenar la información de una competición
Atributos Propuestos
id: identificador de la entidad **name:** nombre de la competición **type:** tipo de la competición (atletismo, ciclismo, natación o triatlón)

Nombre de la Clase
Like
Descripción
Clase del modelo asociada un “me gusta” de un resultado
Responsabilidades
Almacenar la información de un “me gusta”
Atributos Propuestos
id: identificador de la entidad **date:** fecha en la que se registro **usuario:** usuario que indicó “me gusta”

Nombre de la Clase
Comment
Descripción
Clase del modelo asociada un comentario de un resultado
Responsabilidades
Almacenar la información de un resultado
Atributos Propuestos
id: identificador de la entidad **text:** texto del comentario **date:** fecha en la que se registro **usuario:** usuario que realizó el comentario

Nombre de la Clase
Warning
Descripción
Clase del modelo asociada un aviso de resultados falso de un resultado
Responsabilidades
Almacenar la información de aviso de resultado falso
Atributos Propuestos
id: identificador de la entidad **text:** texto del aviso **date:** fecha en la que se registro **usuario:** usuario que realizó el aviso

Nombre de la Clase
File
Descripción
Clase del modelo asociada un fichero
Responsabilidades
Almacenar la información de un fichero
Atributos Propuestos
id: identificador de la entidad **path:** nombre o ruta del fichero **type:** tipo del fichero (image o documento)

5.2.2.2 Controlador

Nombre de la Clase
GeneralController
Descripción
Clase que se encargará de gestionar las peticiones desde la interfaz de usuario de tipo general
Responsabilidades
Actuar de controlador entre la interfaz de usuario y el modelo de datos
Métodos Propuestos
index: comprueba si existe usuario autenticado, en ese caso redirige a la página de inicio personal, en otro caso muestra la página de inicio de la aplicación **login**: comprueba las credenciales de un usuario, establece la sesión y lo redirige a su página de inicio personal **signup**: da de alta un nuevo usuario en estado inactivo en el sistema con los datos suministrados **confirmation**: cambio el estado de un usuario inactivo a un usuario activo **métodos de control de errores:** respuesta a errores de interfaz, badRequest, unauthorized, notFound, accessDenied…

Nombre de la Clase
UserController
Descripción
Clase que se encargará de gestionar las peticiones desde la interfaz relacionadas con gestión de usuarios
Responsabilidades
Actuar de controlador entre la interfaz de usuario y el modelo de datos
Métodos Propuestos
profile: carga los datos del perfil de usuario **userSummary**: carga los resultados de un usuario **addFiend**: añade un usuario amigo al usuario que realiza la petición **removeFriend**: elimina un usuario amigo del usuario que realiza la petición **saveProfile**: guarda las modificaciones realizadas en el perfil de usuario **inactivateUser**: desactiva un usuario tras pedir confirmación (solo administrador)

Nombre de la Clase
CompetitionController
Descripción
Clase que se encargará de gestionar las peticiones desde la interfaz relacionadas con gestión de competiciones y ediciones de competiciones
Responsabilidades
Actuar de controlador entre la interfaz de usuario y el modelo de datos
Métodos Propuestos
addCompetition: carga la información necesaria y da acceso a la pantalla de alta **addEdition**: carga la información necesaria y da acceso a la pantalla de alta **competitionDetail**: carga los datos para mostrar la información de una competición **editionDetail**: carga los datos para mostrar la información de una edición de una competición **competitionForm**: almacena los datos de una nueva competición **editionForm**: almacena los datos de una nueva edición de una competición **searchCompetitions**: busca competiciones para ser asociadas a una edición **searchEditions**: busca ediciones para ser asociadas a un resultado

Nombre de la Clase
ResultController
Descripción
Clase que se encargará de gestionar las peticiones desde la interfaz relacionadas con gestión de resultados
Responsabilidades
Actuar de controlador entre la interfaz de usuario y el modelo de datos
Métodos Propuestos
addResult: accede a la pantalla que permite registrar un nuevo resultado **addResultFromEdition**: accede a la pantalla para dar de alta un nuevo resultado proveniente del alta de una nueva edición **commentForm**: almacena un comentario en un resultado **dashboard**: carga la información de la página de inicio personal de un usuario **deleteResult**: realiza el borrado de un resultado **editResult**: accede a la funcionalidad para modificar un resultado ya registrado **like**: almacena un "me gusta" en un resultado **unlike**: elimina un "me gusta" de un resultado **loadLikesAndComments**: carga la información de "me gusta" y comentarios de un resultado **notifyWarning**: da acceso a la pantalla que permite registrar un aviso de resultado falso **resultDetail**: carga toda la información de un resultado y da acceso a la pantalla pertinente **resultForm**: almacena los datos de un resultado **uploadClassification**: guarda el fichero de clasificación de un resultado **uploadImages**: guarda los ficheros de imagen de un resultado **warningForm**: almacena la información de un aviso de resultado falso **warnings**: muestra la lista de avisos de resultado falso (solo administrador) **warningDetail**: muestra los comentarios recibidos como aviso de un resultado (solo administrador)

Nombre de la Clase
SearchController
Descripción
Clase que se encargará de gestionar las peticiones desde la interfaz relacionadas con búsquedas
Responsabilidades
Actuar de controlador entre la interfaz de usuario y el modelo de datos
Métodos Propuestos
search: recoge el texto de búsqueda y el tipo de búsqueda realizado (competiones o usuario) y devuelve los resultados coincidente.

5.2.2.3 Repositorios (acceso a BD)

Nombre de la Clase
UserRepository
Descripción
Clase que se comunica con la información de BD de un usuario
Responsabilidades
Dispone de base de las acciones CRUD e implementa funcionalidades de consulta a BD
Métodos Propuestos
findByText: busca un usuario por el nombre o nombre de usuario a partir de un texto dado **findByUsernameAndState**: busca un usuario por nombre de usuario y estado

Nombre de la Clase
CompetitionRepository
Descripción
Clase que se comunica con la información de BD de una competición
Responsabilidades
Dispone de base de las acciones CRUD e implementa funcionalidades de consulta a BD
Métodos Propuestos
findAllByName: busca competiciones por el nombre **findByNameContainingIgnoreCase**: busca competiciones por nombre ignorando mayúsculas **findByText**: busca una competición por el nombre a partir de un texto dado

Nombre de la Clase
EditionRepository
Descripción
Clase que se comunica con la información de BD de una edición de una competición
Responsabilidades
Dispone de base de las acciones CRUD e implementa funcionalidades de consulta a BD
Métodos Propuestos
findAllByCompetitionOrderByDateDesc: busca todas las ediciones de una competición dada **findAllByEditionName**: busca ediciones por nombre de la edición **findByEditionNameContainingIgnoreCase**: busca una edición por el nombre ignorando mayusculas **FindByText**: busca una edición de competición por el nombre a partir de un texto dado **findFirstByOrderByDateDesc**: devuelve la primera edición encontrada ordenada por fecha

Nombre de la Clase
ResultRepository
Descripción
Clase que se comunica con la información de BD de un resultado
Responsabilidades
Dispone de base de las acciones CRUD e implementa funcionalidades de consulta a BD
Métodos Propuestos
findAllByEditionOrderByTime: busca resultados de una edición ordenados por tiempo **findAllByUserAndEdition**: busca resultados de un usuario y edición **findAllByUserAndEditionIn**: busca resultados de un usuario en varias ediciones **findByUser**: busca resultados de un usuario **findByUserIn**: busca resultado de varios usuarios

5.2.2.4 Servicios

Nombre de la Clase
UserDetailsServiceImpl
Descripción
Servicio que gestiona la sesión de usuario
Responsabilidades
Carga los datos necesarios del usuario en sesión
Métodos Propuestos
loadUserByUsername: busca la información del usuario y la carga en sesión compeltando

Nombre de la Clase
MailServiceImpl
Descripción
Servicio que permite el envío de emails
Métodos Propuestos
prepareAndSend: envía emails a los destinatarios suministrados con el texto indicado.

Nombre de la Clase
GridFSStorageService
Descripción
Servicio que implementa StorageService y permite almacenar ficheros en la BD Mongo
Métodos Propuestos
loadAsResource: recupera un fichero **store**: almacena un fichero

Nombre de la Clase
FileSystemStorageService
Descripción
Servicio que implementa StorageService y permite almacenar ficheros en el sistema
Métodos Propuestos
loadAsResource: recupera un fichero **store**: almacena un fichero

5.3 Análisis de Casos de Uso y Escenarios

5.3.1 Ver página principal

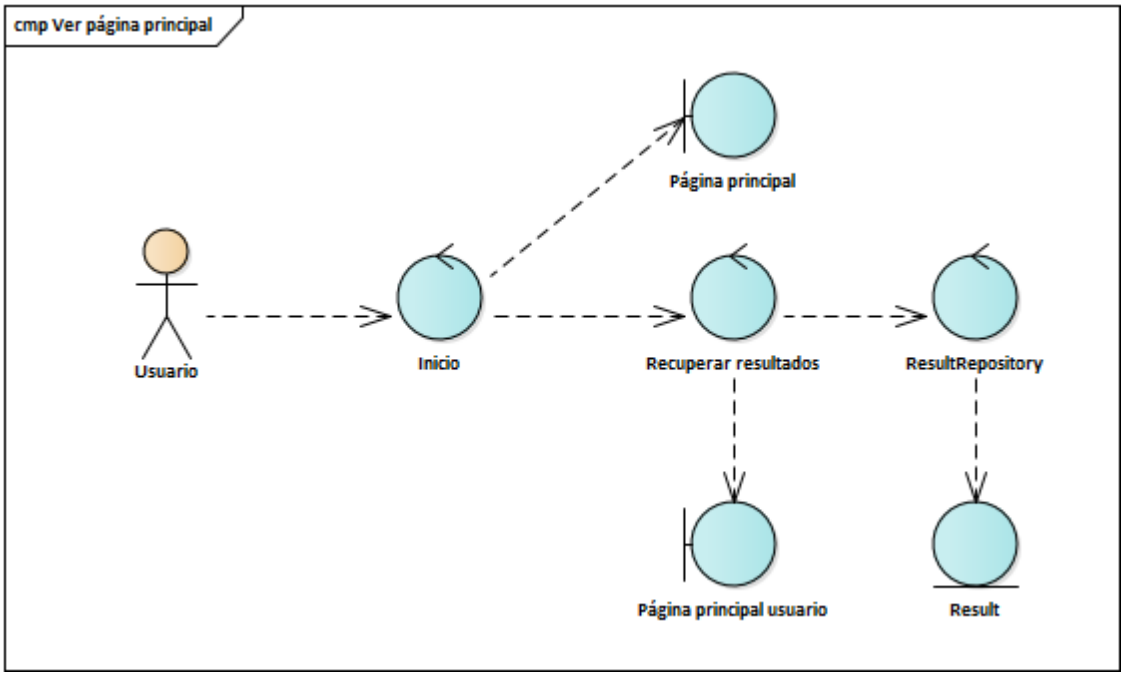


Figura 5.7. Diagrama de robustez Ver página principal

Ver página principal	
Precondiciones	No
Poscondiciones	No
Actores	Iniciado y terminado por el usuario
Descripción	El usuario: 1. Accederá a la pantalla de inicio 2. El sistema comprobará si está ya autenticado 3. En caso de no estar autenticado se le redirige a la página principal de la aplicación, si lo está, se le redirige a la página principal del usuario 4. Se recuperan los resultados visibles para el usuario (propios y de amigos) 5. Se muestra la página
Excepciones	• **La base de datos no está disponible**: No se pueden obtener los resultados de la base de datos 1. Notificar un error asociado al problema encontrado

5.3.2 Registrar

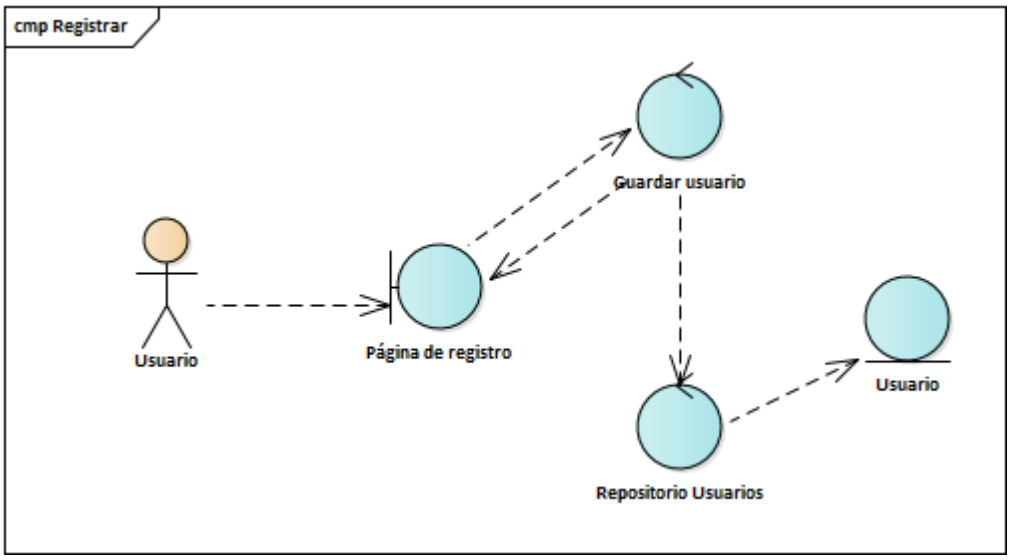


Figura 5.8. Diagrama de robustez de Registrar

Registrar	
Precondiciones	No
Poscondiciones	Se añade un registro a la BD con el usuario inactivo y se envía un email de confirmación al usuario registrado
Actores	Iniciado por un usuario cualquier
Descripción	1. El sistema muestra la pantalla de registro. 2. El usuario introduce sus datos personales 3. El sistema valida la información introducida 4. Vuelve a la página de registro y se muestra una notificación de que se le ha enviado un email para la confirmación del registro
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Alta errónea porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal • **Escenario Alternativo 2**: Usuario ya existe en el sistema 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información del usuario 1. Notificar un error asociado al problema encontrado

5.3.3 Confirmar registrar

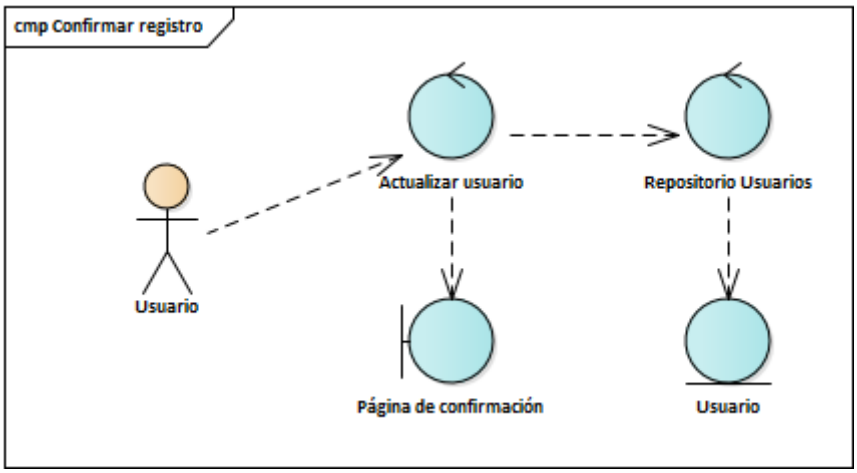


Figura 5.9. Diagrama de robustez de Confirmar registro

Confirmar registro	
Precondiciones	El usuario se ha registrado previamente en el sistema
Poscondiciones	El usuario en BD se actualiza pasando a tener estado activo
Actores	Iniciado por un usuario que haya realizado el proceso de registro
Descripción	1. El usuario, a través de una URL única suministrado a través de un email accede a la página de confirmación 2. El sistema utiliza la información para activar el usuario 3. Se muestra una página informando de la finalización del proceso correctamente
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Usuario no existe en el sistema 1. Notificar el hecho al usuario 2. Ir a la página de inicio
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información del usuario 1. Notificar un error asociado al problema encontrado

5.3.4 Autenticar

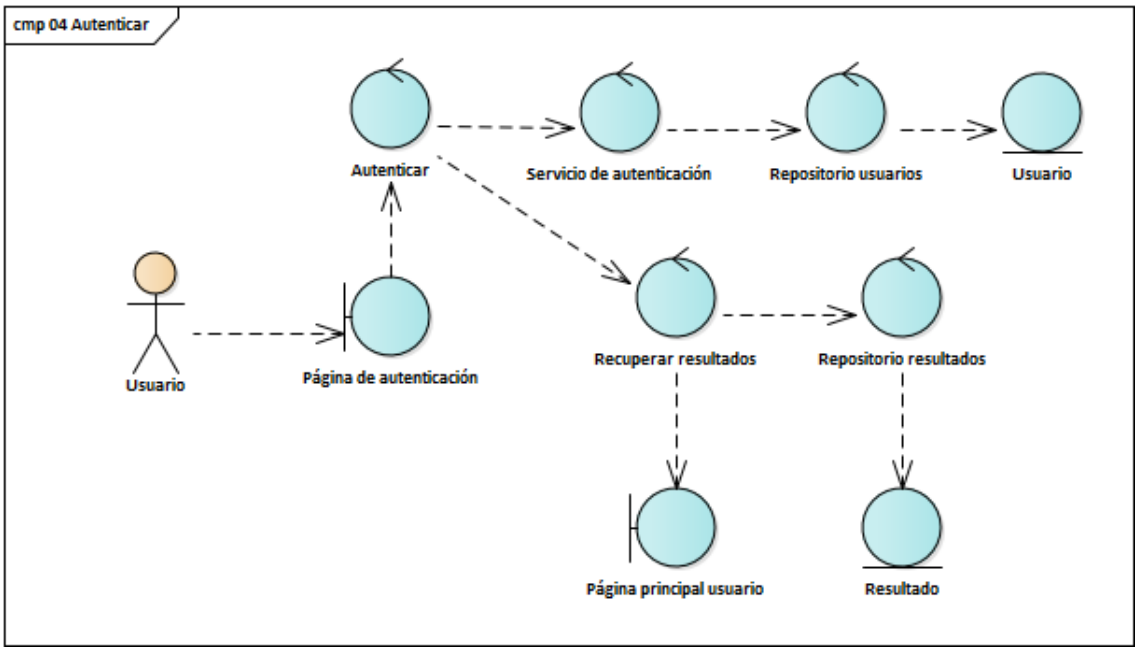


Figura 5.10. Diagrama de robustez de Autenticar

Autenticar	
Precondiciones	Existe un usuario activo en el sistema con los datos oportunos
Poscondiciones	El usuario inició sesión y accede a su página principal
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de autenticación 2. El usuario introduce su usuario y contraseña 3. El sistema valida la información introducida 4. Si los datos suministrados son correctos se le da acceso a su página principal
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Autenticación errónea porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal • **Escenario Alternativo 2**: Usuario no existe en el sistema 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.5 Editar perfil y privacidad

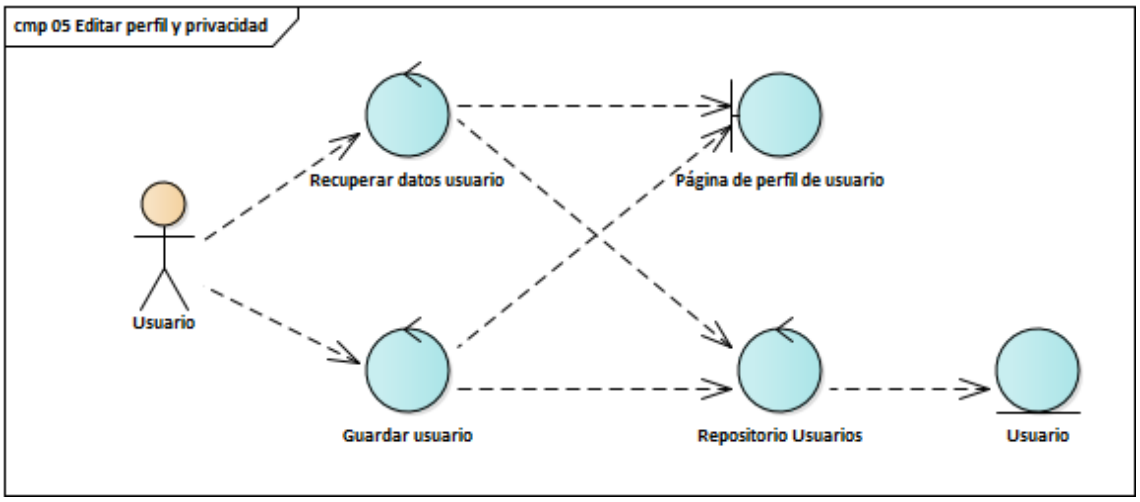


Figura 5.11. Diagrama de robustez de editar perfil y privacidad

Editar perfil y privacidad	
Precondiciones	Existe un usuario autenticado en el sistema
Poscondiciones	Los datos del usuario en BD son modificados con la información suministrada
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de edición de perfil 2. El usuario modifica sus datos personales y privacidad 3. El sistema valida la información introducida por el usuario 4. La información se actualiza en BD 5. El sistema devuelve al usuario a la página de perfil mostrando una notificación del proceso terminó correctamente
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede guardar porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.6 Crear competición

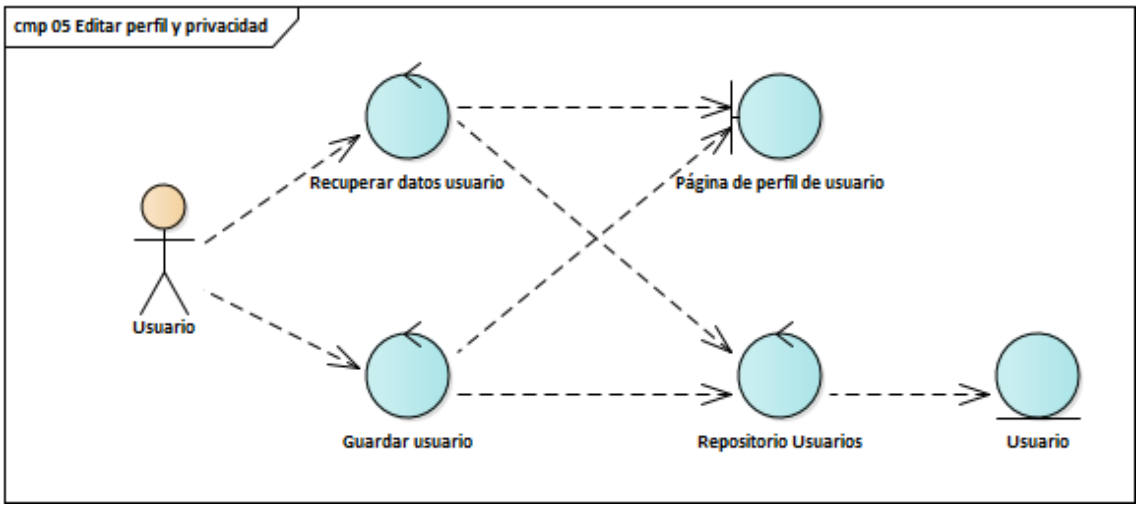


Figura 5.12. Diagrama de robustez de crear competición

Crear competición	
Precondiciones	Existe un usuario autenticado en el sistema y se encuentra en la página de creación de una edición de resultado
Poscondiciones	Se inserta una nueva competición en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede al formulario de competición 2. El usuario introduce los datos necesarios 3. El sistema valida la información introducida por el usuario 4. La información se almacena en BD 5. El sistema devuelve al usuario a la página de formulario de edición de competición
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede guardar porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.7 Crear edición de competición

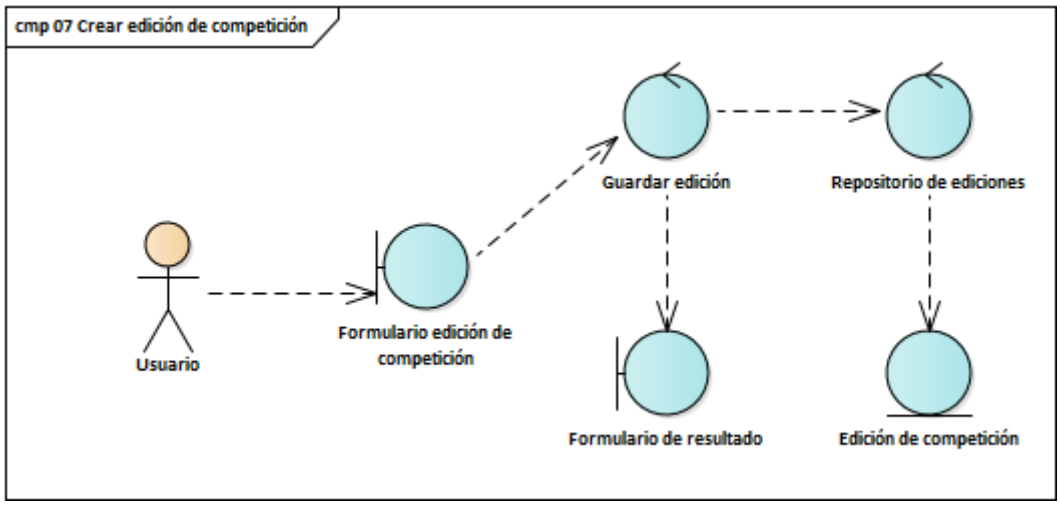


Figura 5.13. Diagrama de robustez de crear edición de competición

Crear edición de competición	
Precondiciones	Existe un usuario autenticado en el sistema y se encuentra en la página de creación de un resultado
Poscondiciones	Se inserta una nueva edición de competición en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede al formulario de edición de competición 2. El usuario introduce los datos necesarios 3. El sistema valida la información introducida por el usuario 4. La información se almacena en BD 5. El sistema devuelve al usuario a la página de formulario de resultado
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede guardar porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.8 Crear resultado

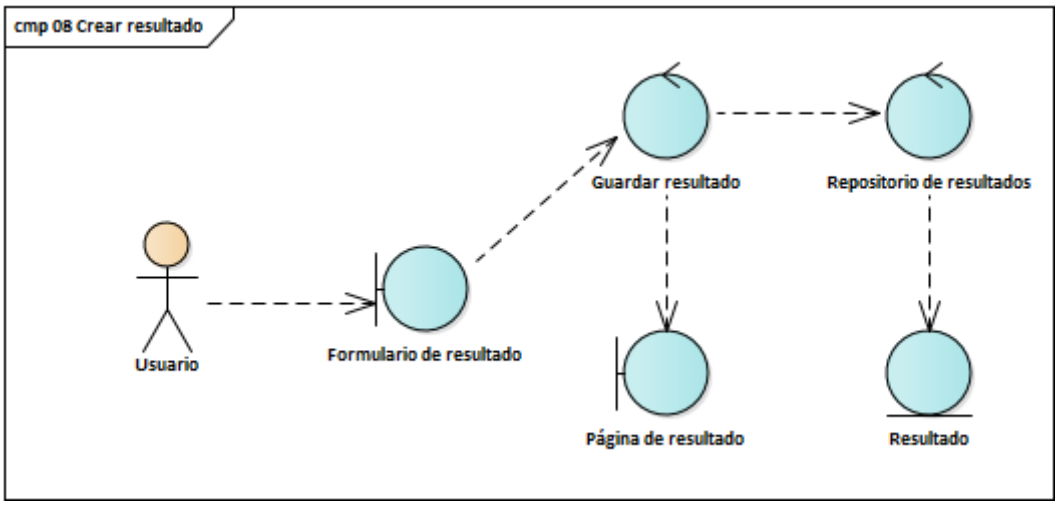


Figura 5.14. Diagrama de robustez de crear resultado

Crear resultado	
Precondiciones	Existe un usuario autenticado en el sistema y se encuentra en la página de creación de resultado
Poscondiciones	Se inserta un nuevo resultado en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede al formulario de resultado 2. El usuario introduce los datos necesarios 3. El sistema valida la información introducida por el usuario 4. La información se almacena en BD 5. El sistema devuelve al usuario a la página de detalle de resultado
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede guardar porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal • **Escenario Alternativo 1**: No se puede guardar ya existe un resultado de este usuario en esta edición 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.9 Editar resultado

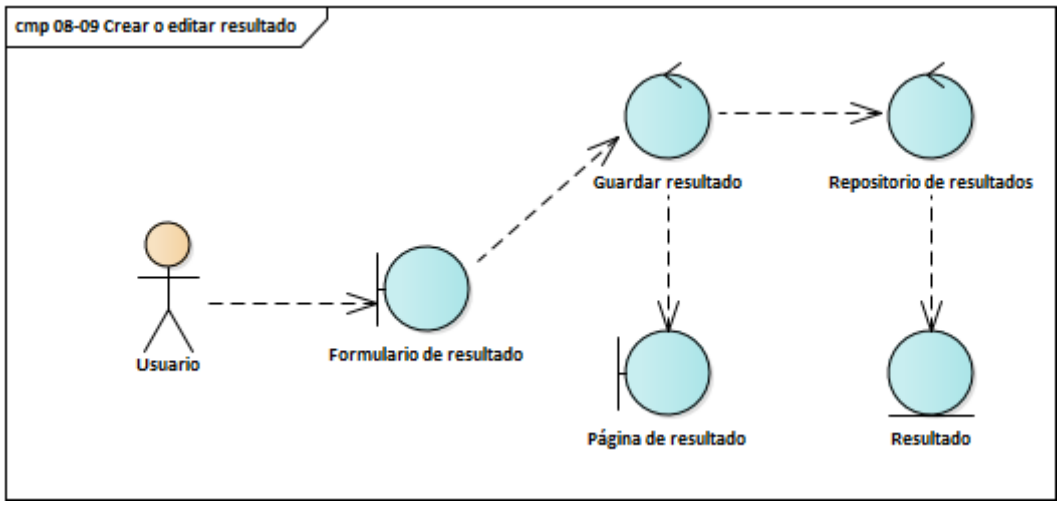


Figura 5.15. Diagrama de robustez de editar resultado

Editar resultado	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado a editar
Poscondiciones	Se actualiza la información del resultado en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede al formulario de resultado 2. El usuario introduce los datos necesarios 3. El sistema valida la información introducida por el usuario 4. La información se almacena en BD 5. El sistema devuelve al usuario a la página de detalle de resultado
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede actualizar porque faltan campos obligatorios en el formulario 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.10 Borrar resultado

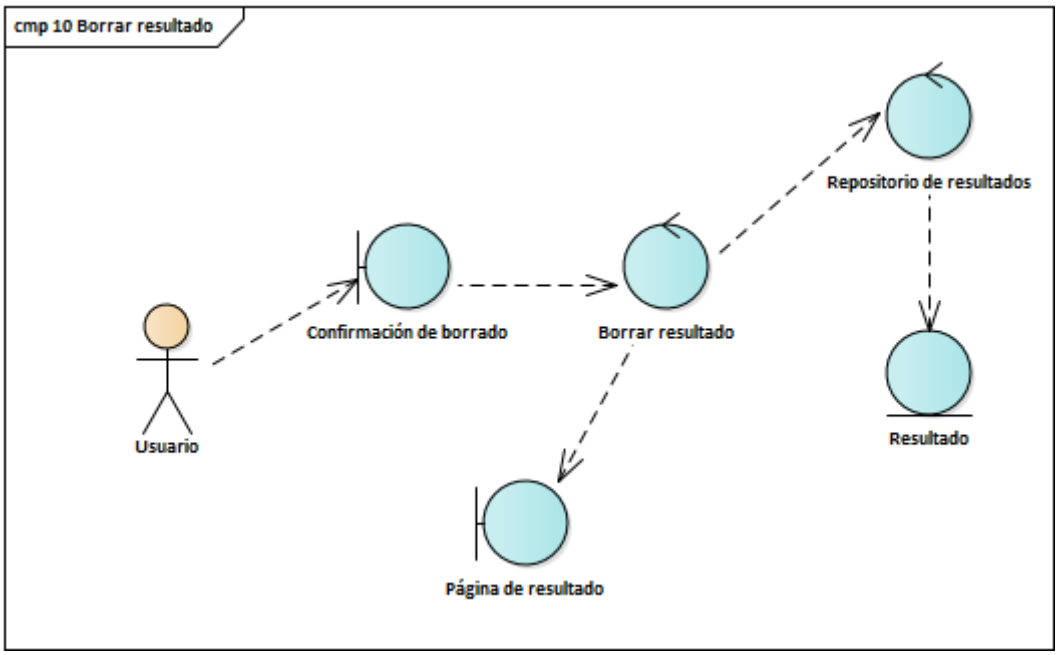


Figura 5.16. Diagrama de robustez de borrar resultado

Borrar resultado	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado a borrar
Poscondiciones	Se elimina el resultado en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario invoca la acción de borrar resultado 2. El sistema muestra un página para confirmar la acción 3. El usuario confirma o cancela la acción 4. Si se confirma la acción el resultado se borra en BD. 5. El sistema devuelve al usuario a la página de inicio del usuario mostrando una notificación de borrado completado
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede borrar el resultado porque no existe en base de datos 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede almacenar la información 1. Notificar un error asociado al problema encontrado

5.3.11 Ver resultado

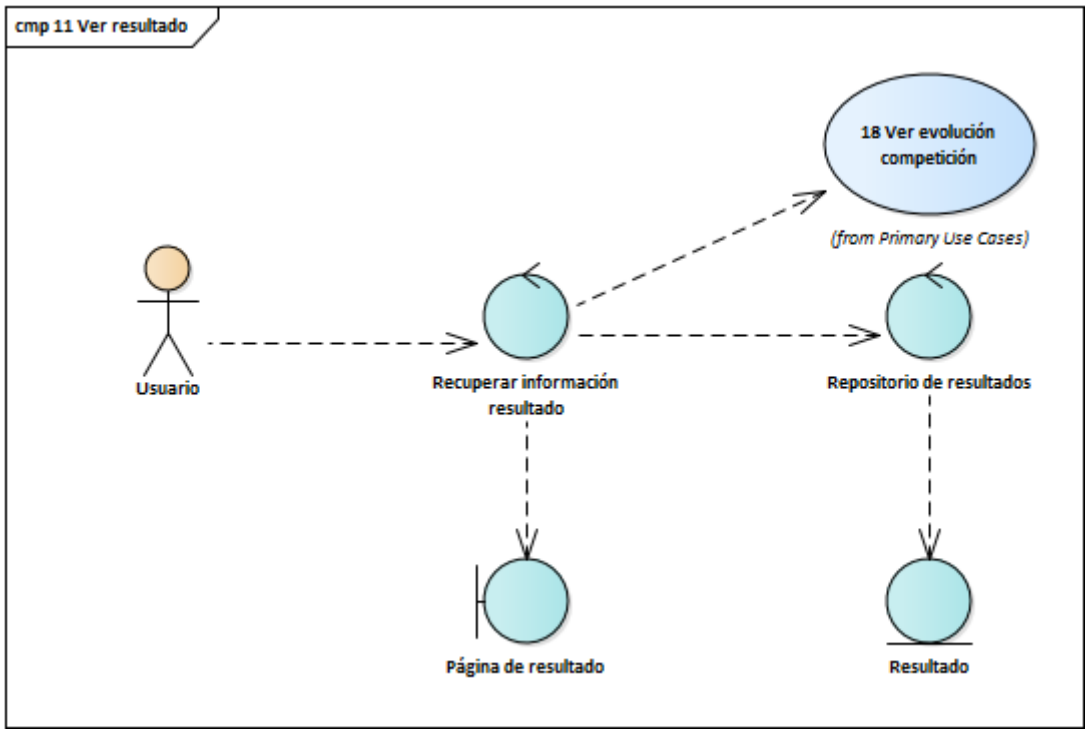


Figura 5.17. Diagrama de robustez de ver resultado

Ver resultado	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado a mostrar
Poscondiciones	No
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de resultado 2. Se recuperan los datos del resultado y se invoca a "Ver evolución competición" 3. Se presentan los datos en la página de resultado
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede mostrar el resultado porque no existe en base de datos 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.12 Ver usuario

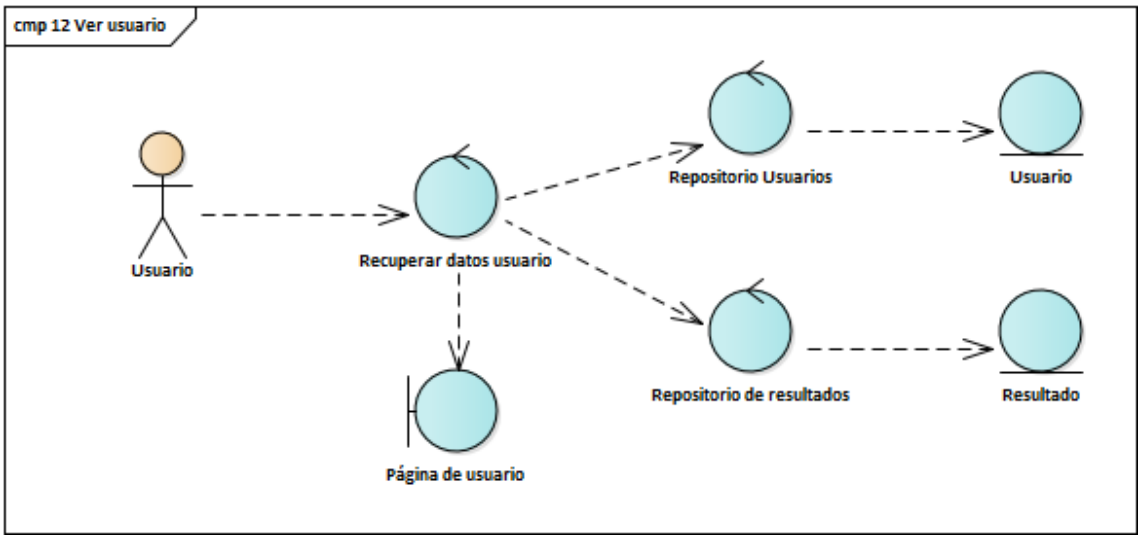


Figura 5.18. Diagrama de robustez de ver usuario

Ver usuario	
Precondiciones	Existe un usuario autenticado en el sistema y el usuario a mostrar
Poscondiciones	No
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de usuario 2. Se recuperan los datos del usuario consultado 3. Se presentan los datos en la página de usuario
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se puede mostrar el usuario consultado porque no existe en base de datos 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.13 Añadir usuario amigo

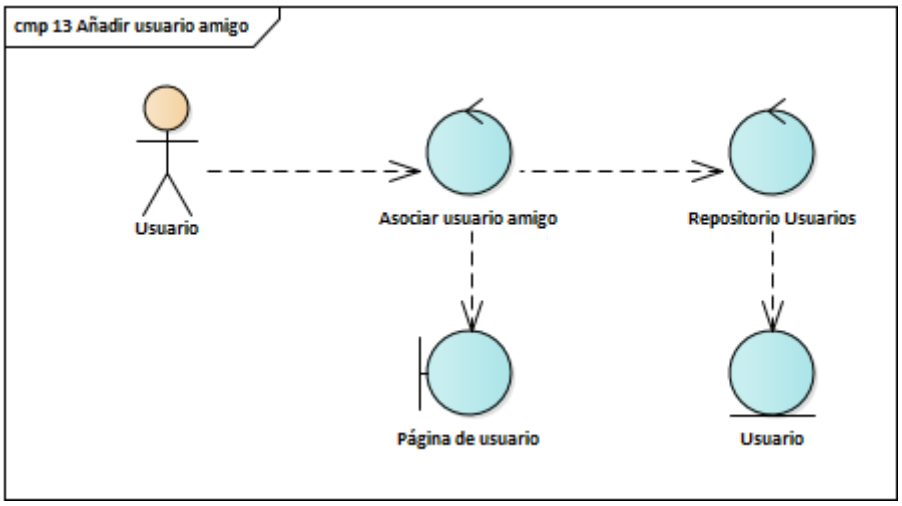


Figura 5.19. Diagrama de robustez de añadir usuario amigo

Añadir usuario amigo	
Precondiciones	Existe un usuario autenticado en el sistema y el usuario a añadir
Poscondiciones	Se almacena en BD una relación entre el usuario y el usuario amigo
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario invoca la acción de añadir amigo desde la página de ver usuario 2. Se añade la relación de amistad en BD 3. El sistema muestra la página de usuario notificando que la operación se completó correctamente
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se asociar el usuario porque no existe en base de datos 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal • **Escenario Alternativo 2**: No se asociar el usuario porque el usuario consultado cambió su privacidad (cuenta privada) 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.14 Eliminar usuario amigo

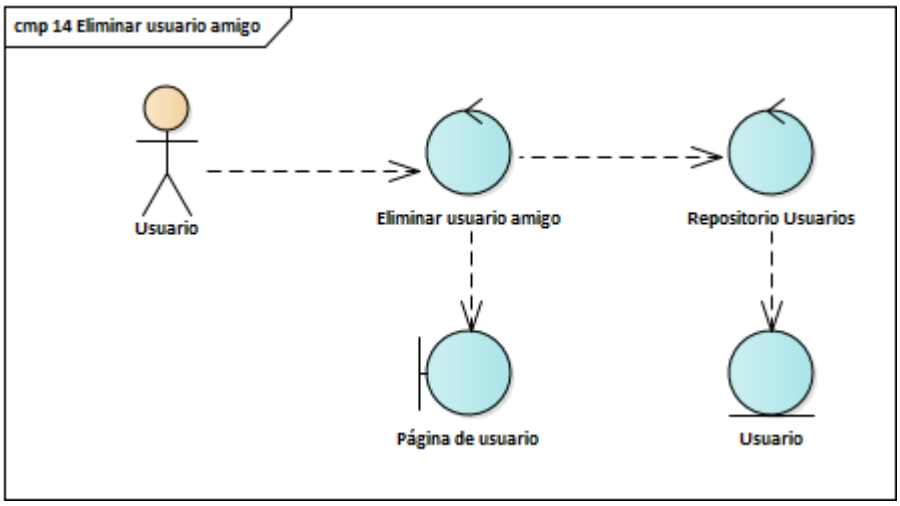


Figura 5.20. Diagrama de robustez de eliminar usuario amigo

Eliminar usuario amigo	
Precondiciones	Existe un usuario autenticado en el sistema y el usuario amigo asociado
Poscondiciones	Se elimina en BD una relación entre el usuario y el usuario amigo
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario invoca la acción de eliminar amigo desde la página de edición de perfil 2. Se muestra una notificación de confirmación 3. Si se confirma la acción se elimina la relación de amistad en BD 4. El sistema muestra la página de edición de perfil eliminando al usuario de la lista de amigos
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.15 Avisar resultado falso

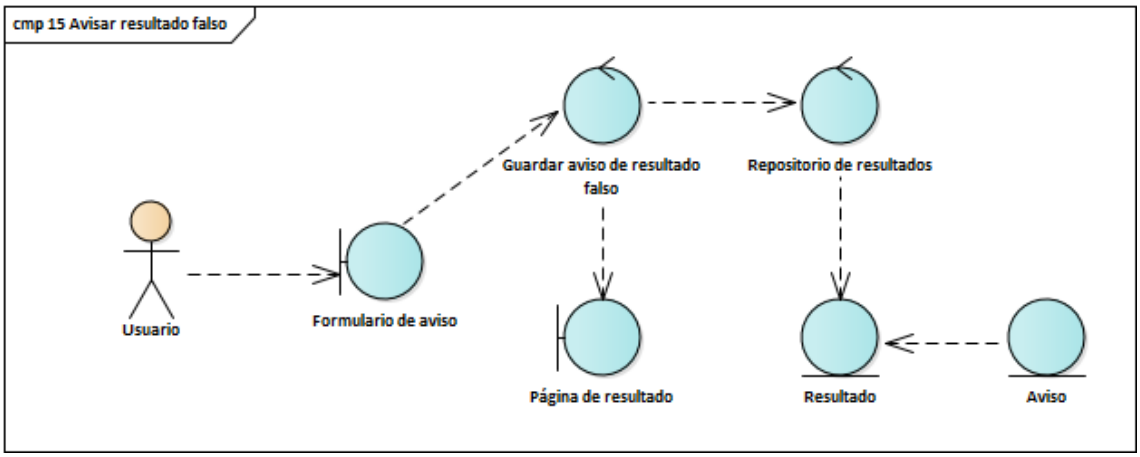


Figura 5.21. Diagrama de robustez de avisar resultado falso

Avisar resultado falso	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado a notificar
Poscondiciones	Se almacena en BD un aviso asociado al resultado
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede al formulario de avisos desde la página de resultado 2. El usuario introduce la información a enviar 3. El sistema almacena la información de aviso en BD 4. El sistema muestra la página de resultado notificando que la operación se completó correctamente
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: No se guardar el aviso porque no existe el resultado en base de datos 1. Notificar el hecho al usuario 2. Volver al paso 1 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.16 Ver competición

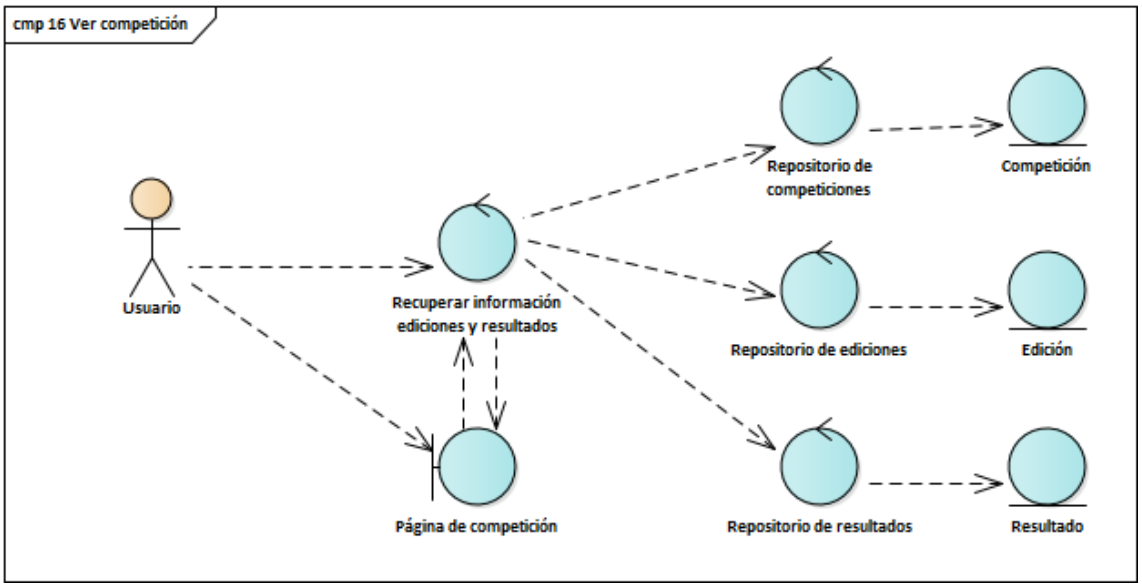


Figura 5.22. Diagrama de robustez de ver competición

Ver competición	
Precondiciones	Existe un usuario autenticado en el sistema y la competición a consultar
Poscondiciones	No
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de una competición a través del buscador 2. El sistema recupera las todas la ediciones de la competición y los resultados de la edición seleccionada (última celebrada por defecto) ordenados por tiempo 3. Se muestra la página con la clasificación virtual de la edición 4. El usuario puede seleccionar de una lista otra edición a visualizar, volviendo al paso 2
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.17 Ver clasificación de edición de competición

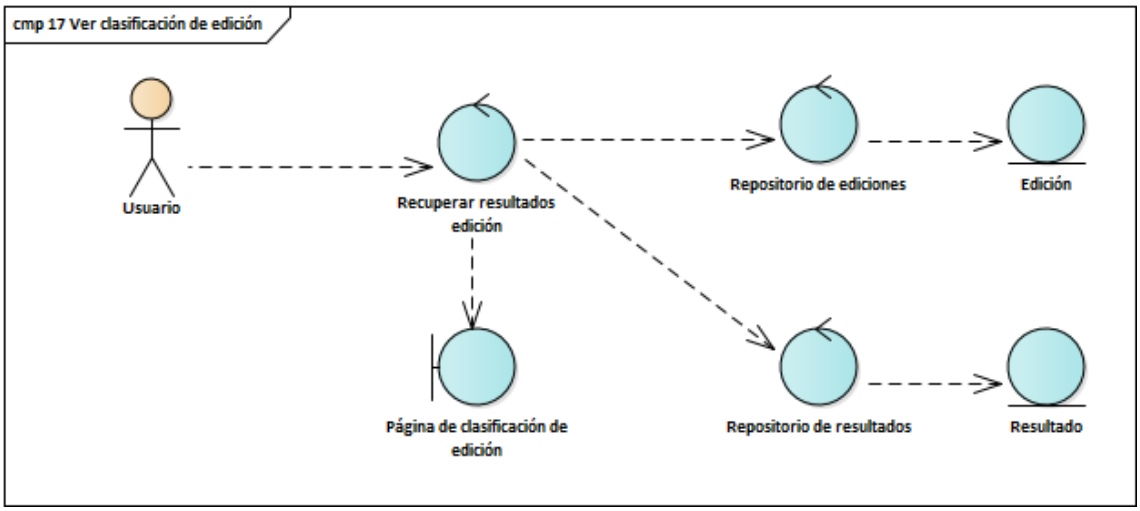


Figura 5.23. Diagrama de robustez de ver clasificación de edición de competición

Ver clasificación de edición de competición	
Precondiciones	Existe un usuario autenticado en el sistema y la edición a consultar
Poscondiciones	No
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de una edición de competición 2. El sistema recupera los resultados de la edición ordenados por tiempo 3. Se muestra la página con la clasificación virtual de la edición
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.18 Ver evolución en una competición

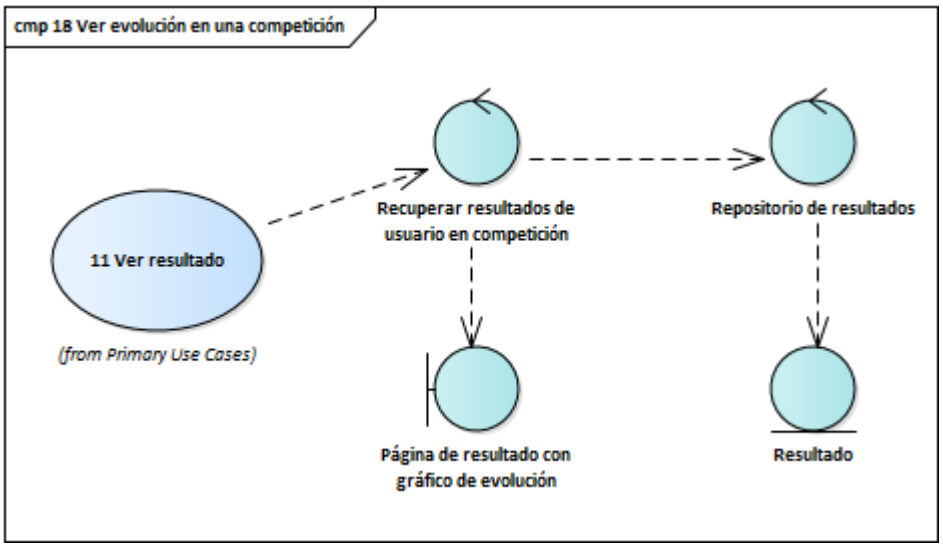


Figura 5.24. Diagrama de robustez de ver evolución en una competición

Ver evolución en una competición	
Precondiciones	Existe un usuario autenticado en el sistema y la competición a consultar
Poscondiciones	No
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de resultado 2. El sistema consulta los resultados del usuario asociado al resultado en otras ediciones y los ordena por fecha 3. Se genera un gráfico con los datos obtenidos, con año de la edición en el eje X y tiempo total en el eje Y
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.19 Dar “me gusta”

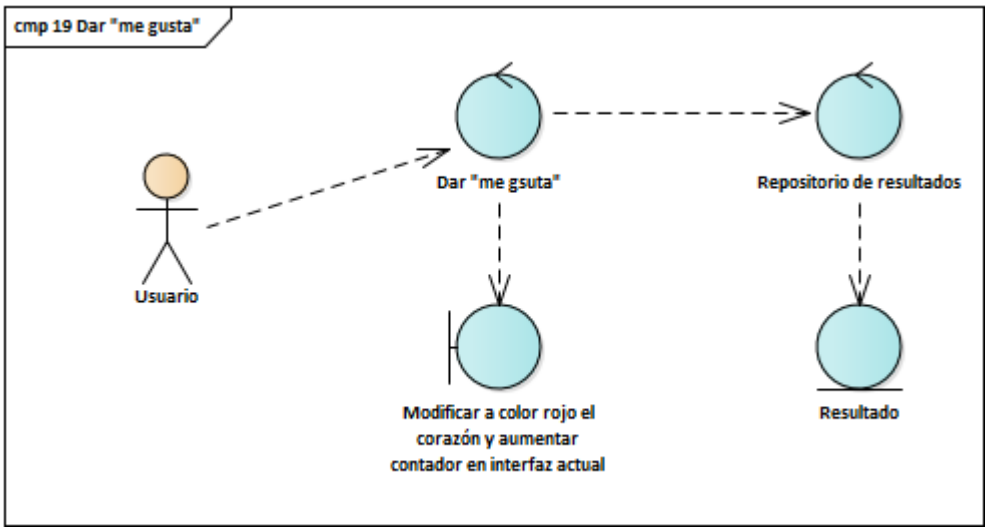


Figura 5.25. Diagrama de robustez de dar “me gusta”

Dar “me gusta”	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado objetivo
Poscondiciones	Se asocia al resultado un “me gusta” en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario invoca la acción dar “me gusta” 2. El sistema asocia un “me gusta” al resultado y lo almacena 3. Se modifica la interfaz para mostrar en color rojo el corazón asociado al resultado y se incrementa el contador de “me gustas” del mismo
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.20 Quitar “me gusta”

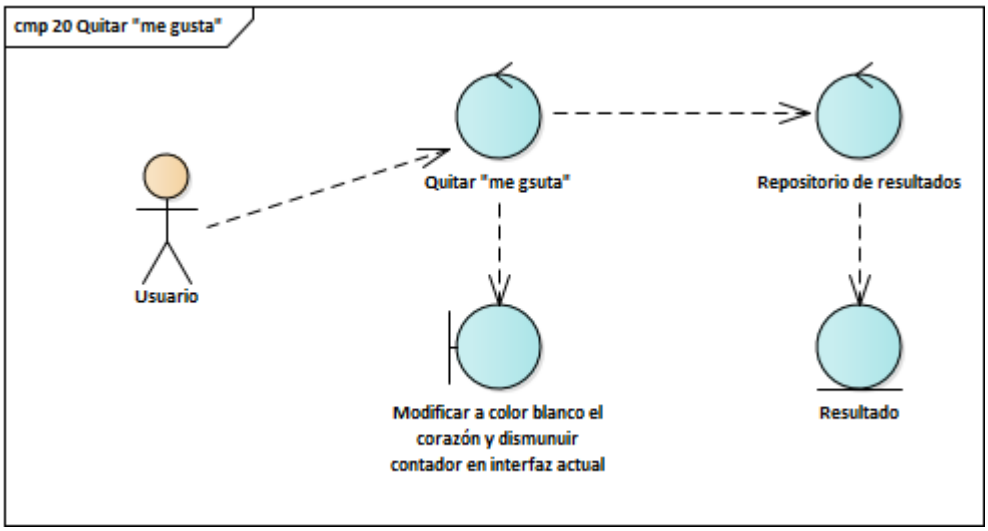


Figura 5.26. Diagrama de robustez de quitar “me gusta”

Quitar “me gusta”	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado objetivo tiene un “me gusta” del usuario
Poscondiciones	Se elimina la asociación del “me gusta” en BD
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario invoca la acción quitar “me gusta” 2. El sistema elimina la asociación de “me gusta” al resultado 3. Se modifica la interfaz para mostrar en color blanco el corazón asociado al resultado y se disminuye el contador de “me gustas” del mismo
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.21 Comentar resultado

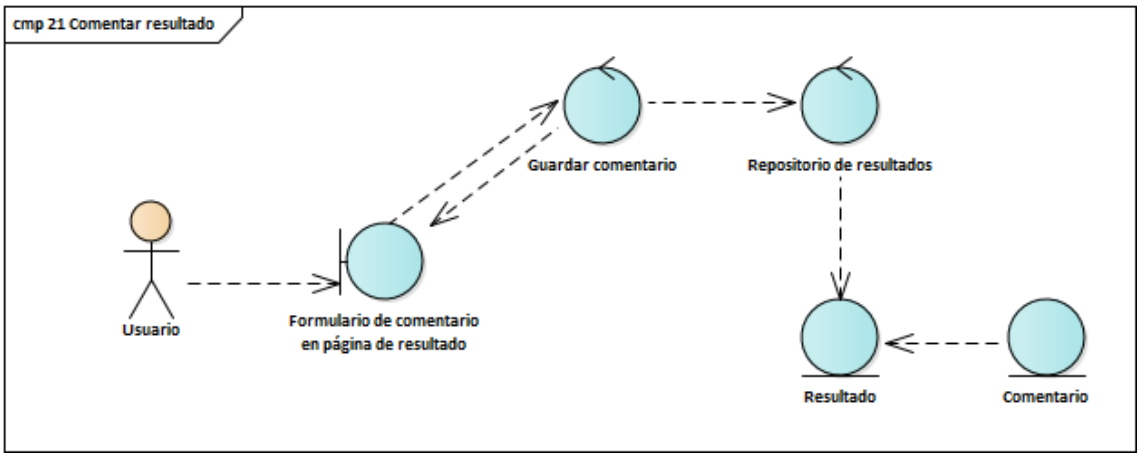


Figura 5.27. Diagrama de robustez de comentar resultado

Comentar resultado	
Precondiciones	Existe un usuario autenticado en el sistema y el resultado a comentar
Poscondiciones	Se almacena en BD un comentario asociado al resultado
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede al formulario de comentario en la página de resultado 2. El usuario introduce el texto a enviar 3. El sistema almacena la información de comentario en BD 4. El sistema muestra la página de resultado actualizado con el comentario
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.22 Buscar competición

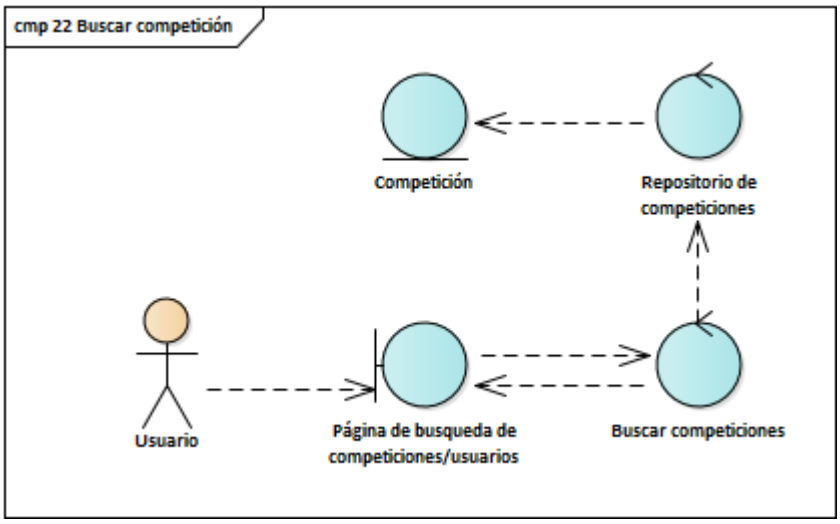


Figura 5.28. Diagrama de robustez de buscar competición

Buscar competición	
Precondiciones	Existe un usuario autenticado en el sistema
Poscondiciones	Se recuperan las competiciones coincidentes con el texto de búsqueda
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de búsqueda de competiciones/usuarios 2. El usuario introduce el texto a buscar 3. El sistema consulta las competiciones con nombre coincidente con el texto introducido 4. El sistema muestra la página de búsqueda con los resultado obtenidos
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.23 Buscar usuarios

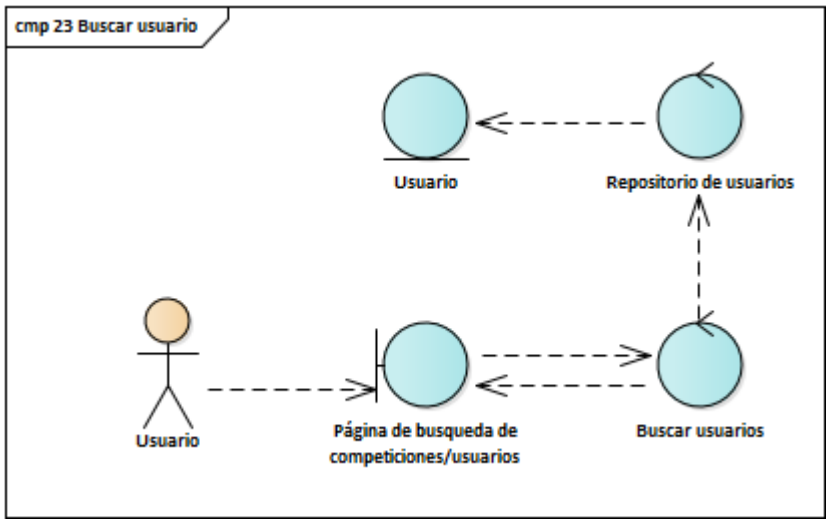


Figura 5.29. Diagrama de robustez de buscar usuarios

Buscar usuarios	
Precondiciones	Existe un usuario autenticado en el sistema
Poscondiciones	Se recuperan los usuarios coincidentes con el texto de búsqueda
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El usuario accede a la página de búsqueda de competiciones/usuarios 2. El usuario introduce el texto a buscar 3. El sistema consulta los usuarios con nombre o nombre de usuario coincidente con el texto introducido 4. El sistema muestra la página de búsqueda con los resultado obtenidos
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.3.24 Listar resultados con aviso

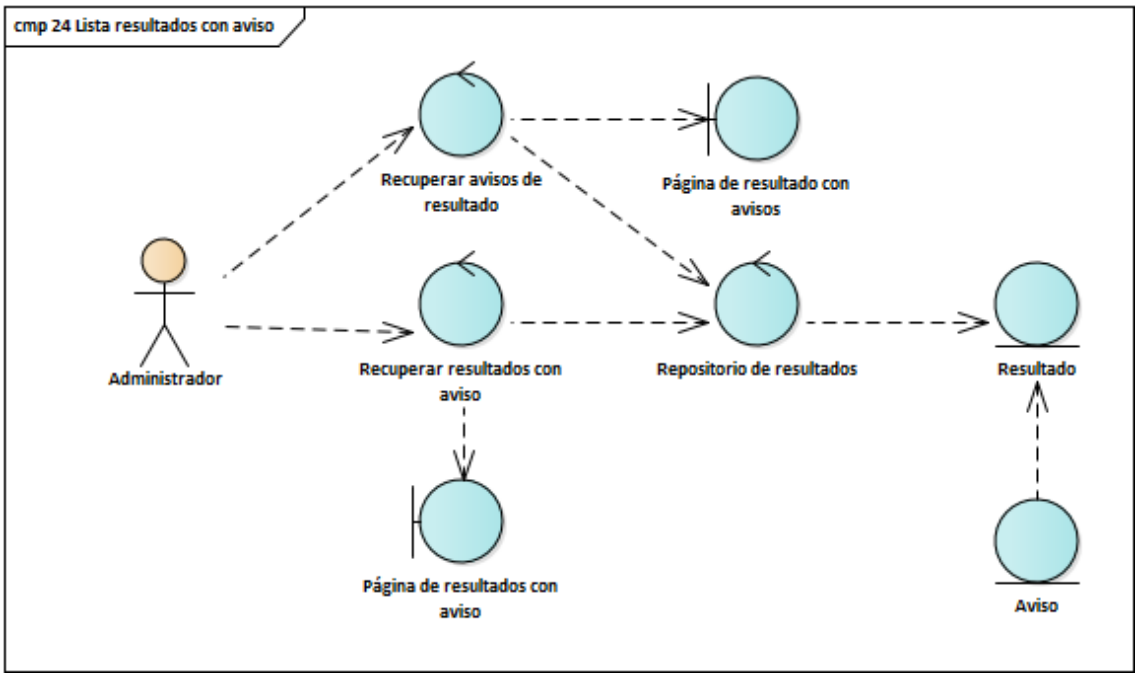


Figura 5.30. Diagrama de robustez de listar resultados con aviso

Listar resultados con aviso	
Precondiciones	Existe un administrador autenticado en el sistema
Poscondiciones	No
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El administrador accede a la página de resultados con avisos 2. El sistema consulta los resultado con aviso 3. El sistema muestra el listado de resultado con aviso 4. El usuario accede a un resultado con aviso 5. El sistema consulta los aviso asociados al resultado 6. El sistema muestra la página de resultado con aviso 7. El usuario puede volver a la página de resultados con aviso
Excepciones	• **La base de datos no está disponible**: No se puede consultar la información 1. Notificar un error asociado al problema encontrado

5.3.25 Desactivar usuario

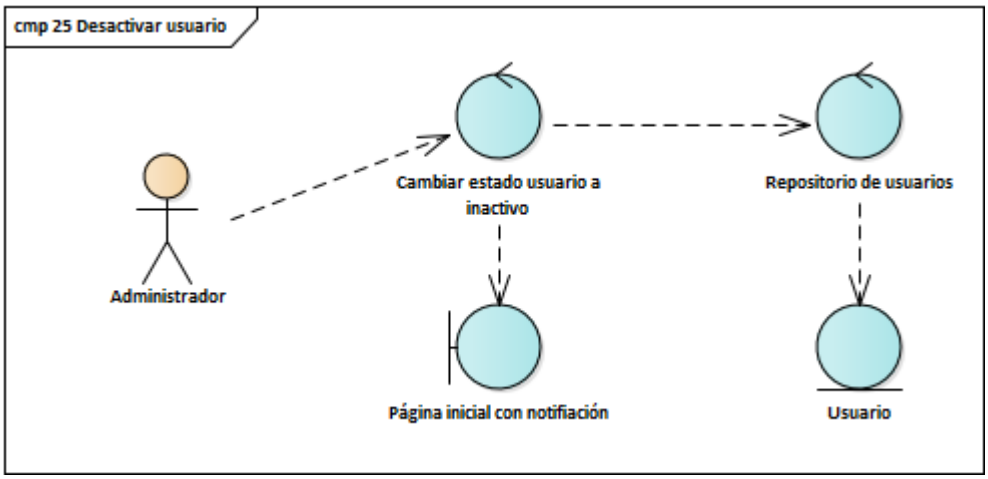


Figura 5.31. Diagrama de robustez de desactivar usuario

Desactivar usuario	
Precondiciones	Existe un administrador autenticado en el sistema y existe el usuario objetivo
Poscondiciones	El estado del usuario objetivo pasa a ser inactivo
Actores	Iniciado por un usuario registrado en el sistema
Descripción	1. El administrador invoca la acción de desactivar usuario 2. El sistema solicita confirmación de la acción 3. Si la acción se confirma, el estado del usuario se modifica a inactivo 4. El sistema muestra la página inicial del administrador y una notificación de que la acción se completó con éxito
Excepciones	• **La base de datos no está disponible**: No se puede guardar la información 1. Notificar un error asociado al problema encontrado

5.4 Análisis de Interfaces de Usuario

5.4.1 Descripción de la Interfaz

A continuación, se muestra de forma gráfica la distribución de pantalla elegida para garantizar la usabilidad y sencillez de la aplicación. Dicha interfaz se adaptará al tipo de dispositivo que acceda a la aplicación (responsive), de tal modo que una pantalla grande, tipo PC, obtendrá una distribución diferente que un dispositivo móvil con pantalla pequeña.

5.4.1.1 Distribución para más de 700 pixeles de ancho

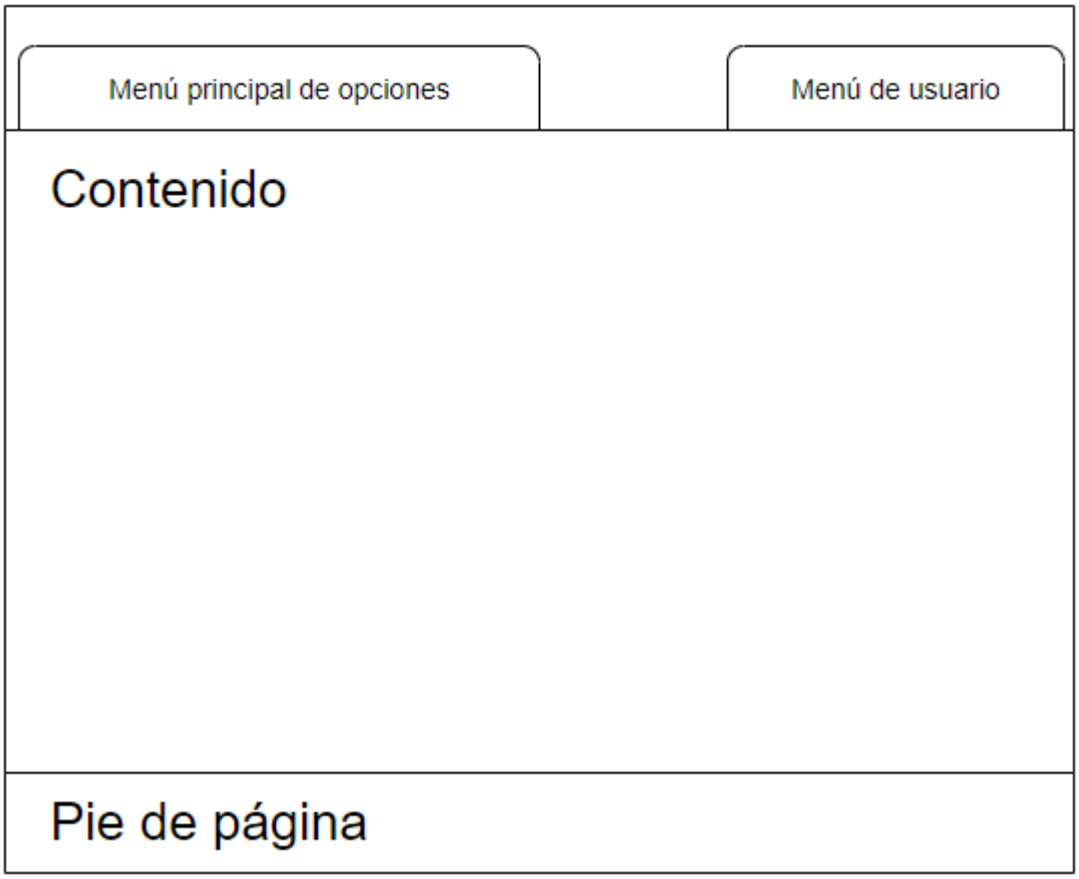


Figura 5.32. Distribución de la interfaz de usuario para pantallas grandes

5.4.1.2 Distribución para menos de 700 pixeles de ancho

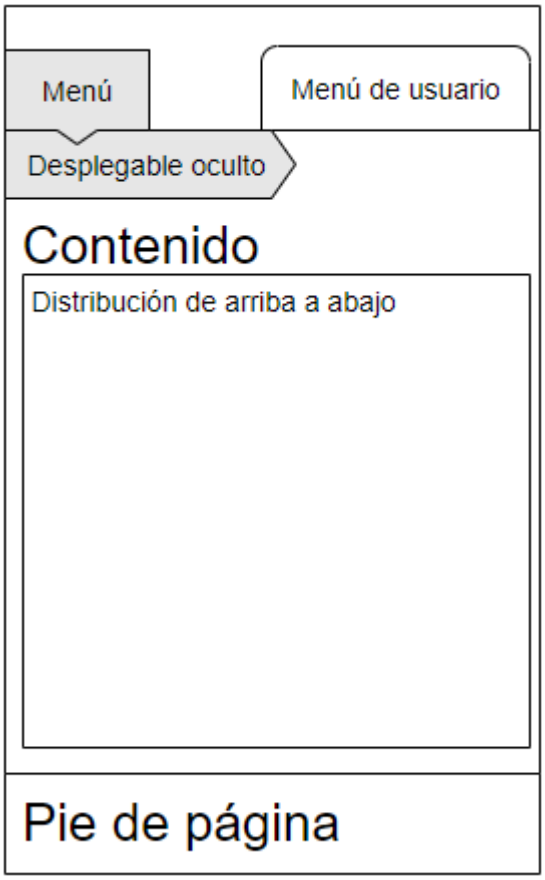


Figura 5.33. Distribución de la interfaz de usuario para pantallas pequeñas

5.4.2 Diagrama de Navegabilidad

A continuación se muestra el diagrama de navegabilidad de la aplicación, de manera resumida y mostrando los flujos más habituales, dado que la aplicación ofrece acceso prácticamente a todas las pantallas desde cualquier otro punto de la aplicación.

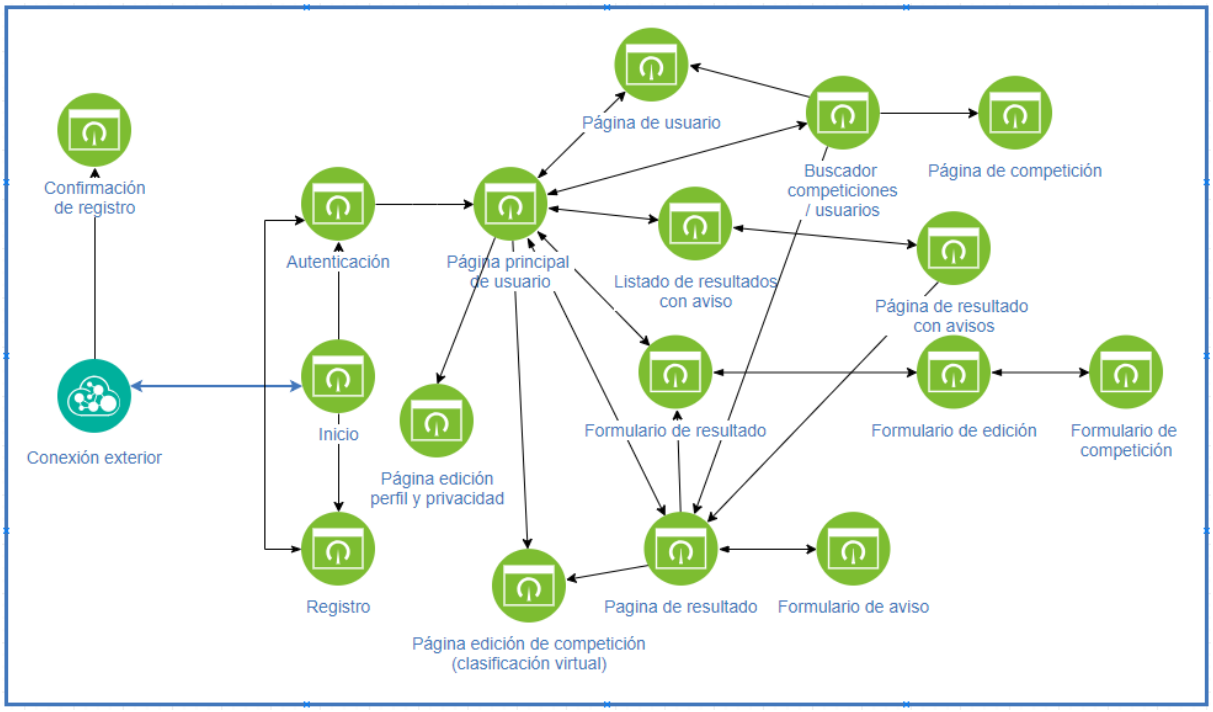


Figura 5.34. Diagrama de navegabilidad

5.5 Especificación del Plan de Pruebas

En esta sección crearemos y diseñaremos el plan de pruebas de la aplicación y sus funciones, así como todos los mecanismos que utilizaremos para detectar errores y corregirlos ya en la fase de implementación.

Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes de la misma.

Se realizarán las siguientes pruebas:

- **Pruebas Unitarias**: Se realizan pruebas unitarias de todos los repositorios de la aplicación, realizando inserciones y borrados de todos los elementos en BD y comprobando que el resultado de la inserción es correcto

- **Pruebas de Integración**: Durante el desarrollo, estructurado secuencialmente, se ejecutarán pruebas de integración a medida que la aplicación se vaya completando.

- **Pruebas del sistema**: Para finalizar el plan de pruebas, se ejecutarán pruebas de todos los escenarios descritos en apartados anteriores. No se detalla tabla de pruebas, ya que se estima que la descripción de los mismos es suficiente para llevarlas a cabo y comprobar si el resultado es satisfactorio sin ser redundantes con la información.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagrama de Paquetes

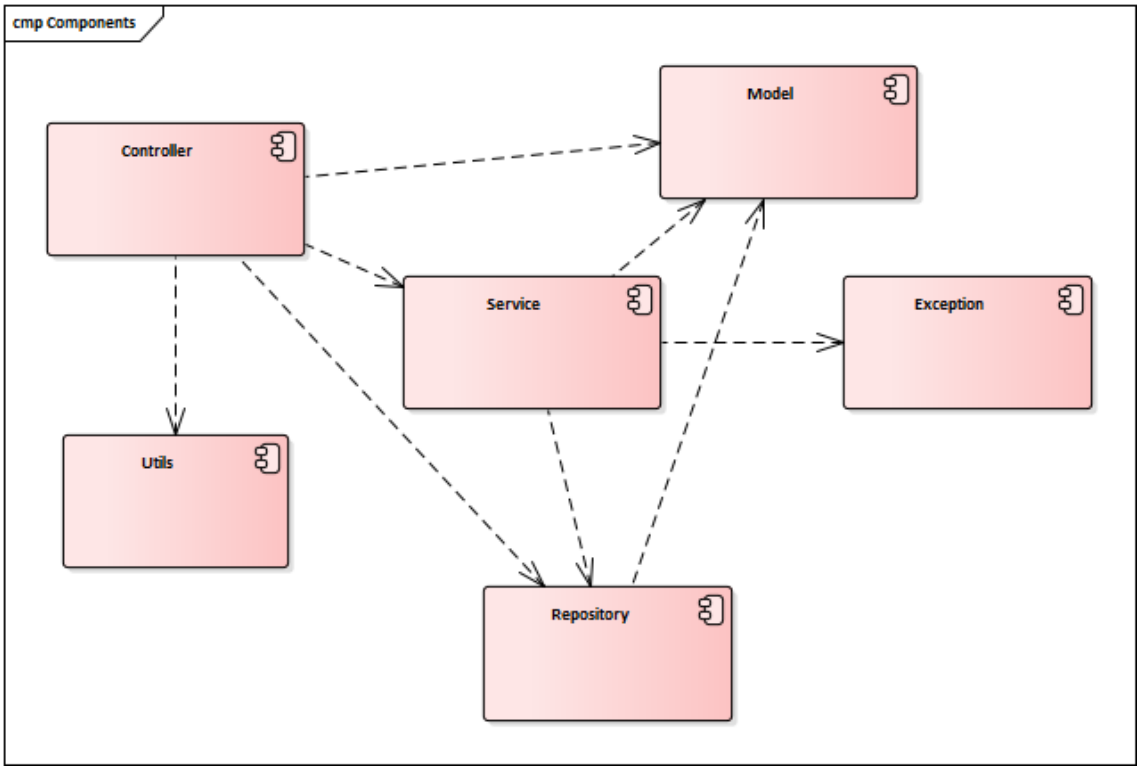


Figura 6.1. Diagrama de paquetes

6.1.1.1 Modelo (model)

En este paquete se agrupan todas las clases que representan el modelo de datos de la aplicación y sus relaciones. Adicionalmente, son clases que contienen mapeos con la base de datos y que hacen uso de Hibernate (ORM) para conseguir la integración con un modelo entidad/relación de forma transparente para el programador.

6.1.1.2 Repositorios (repository)

Paquete que engloba a las clases que utilizan el modelo para realizar las tareas propias de la persistencia de datos, y que gracias al concepto de repositorio suministrado por el Framework Spring, permiten al desarrollador disponer de la funcionalidad CRUD (Create-Read-Update-Delete) y ciertas consultas básicas sin la necesidad de generar código repetitivo.

6.1.1.3 Servicios (services)

Paquete contenedor de clases que implementan servicios del sistema, como los relacionados con el manejo de ficheros o la gestión de autenticación.

6.1.1.4 Controlador (controller)

En este paquete está la funcionalidad de la aplicación. Estas clases implementan el controlador de la aplicación, respondiendo a las peticiones realizadas desde la vista

6.1.1.5 Excepciones (exceptions)

Paquete que engloba a las clases con la implementación de excepciones personalizadas de la aplicación.

6.1.1.6 Utilidades (utils)

Paquete que contiene clases de utilidad.

6.1.2 Diagrama de Despliegue

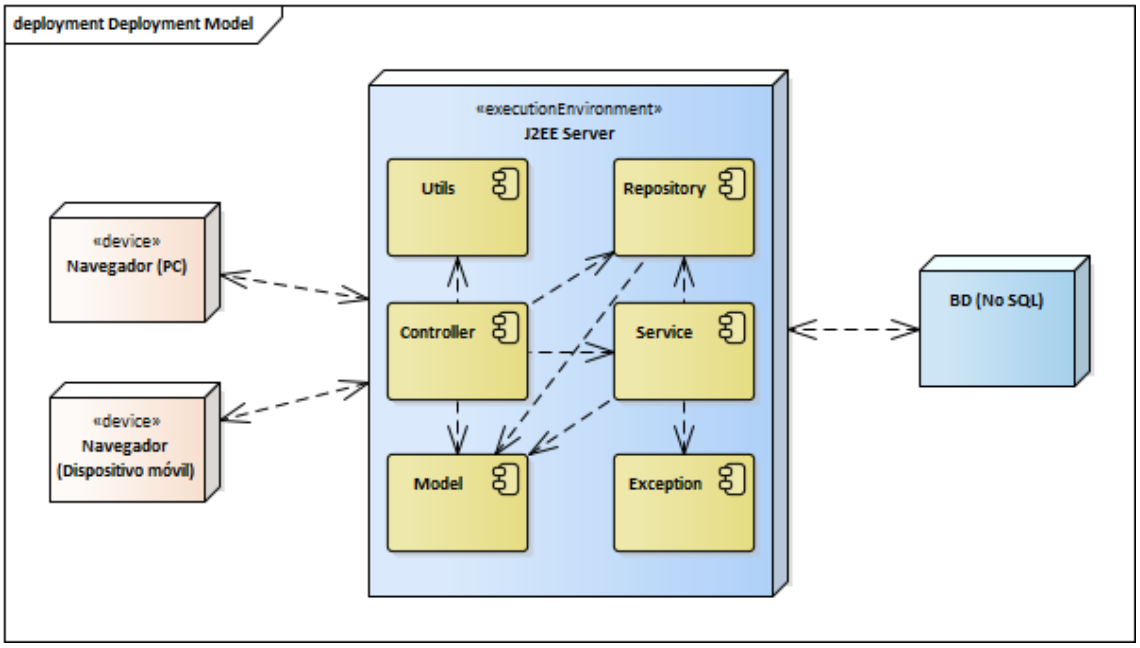


Figura 6.2. Diagrama de despliegue

6.1.2.1 Navegador (PC o dispositivo móvil)

El acceso a la aplicación se realizará a través de un navegador web. Diferenciamos entre distintos dispositivos, ya que la interfaz que visualizará el cliente será diferente en función del tipo de pantalla del que disponga el mismo. Por ello, englobamos los dispositivos en dos grandes grupos: PC u ordenador de escritorio y dispositivo móvil (Smartphone, Tablet…). Desde estos dispositivos se realizarán las peticiones a la aplicación web y se visualizará el resultado devuelto.

6.1.2.2 Servidor J2EE

La aplicación será empaquetada en un archivo de despliegue War o Jar que puede ser desplegado en un servidor de aplicaciones J2EE (Tomcat 8 en nuestro caso) y dónde se ejecutará toda la lógica de los diferentes subsistemas que intervienen en el sistema para ofrecer el resultado oportuno de cada petición del cliente.

6.1.2.3 Base de datos No Sql

Dada la tipología de la aplicación, una red social, se ha decido el uso de una base de dato No Sql (Mongo DB), con características que permiten un mejor escalabilidad, mucha flexibilidad y un buen rendimiento en búsquedas y recuperación de información en este tipo de sistemas.

6.2 Diseño de Clases

6.2.1 Diagrama de Clases

A continuación se muestra un diagrama de clases global de la aplicación (agrupado por paquetes)

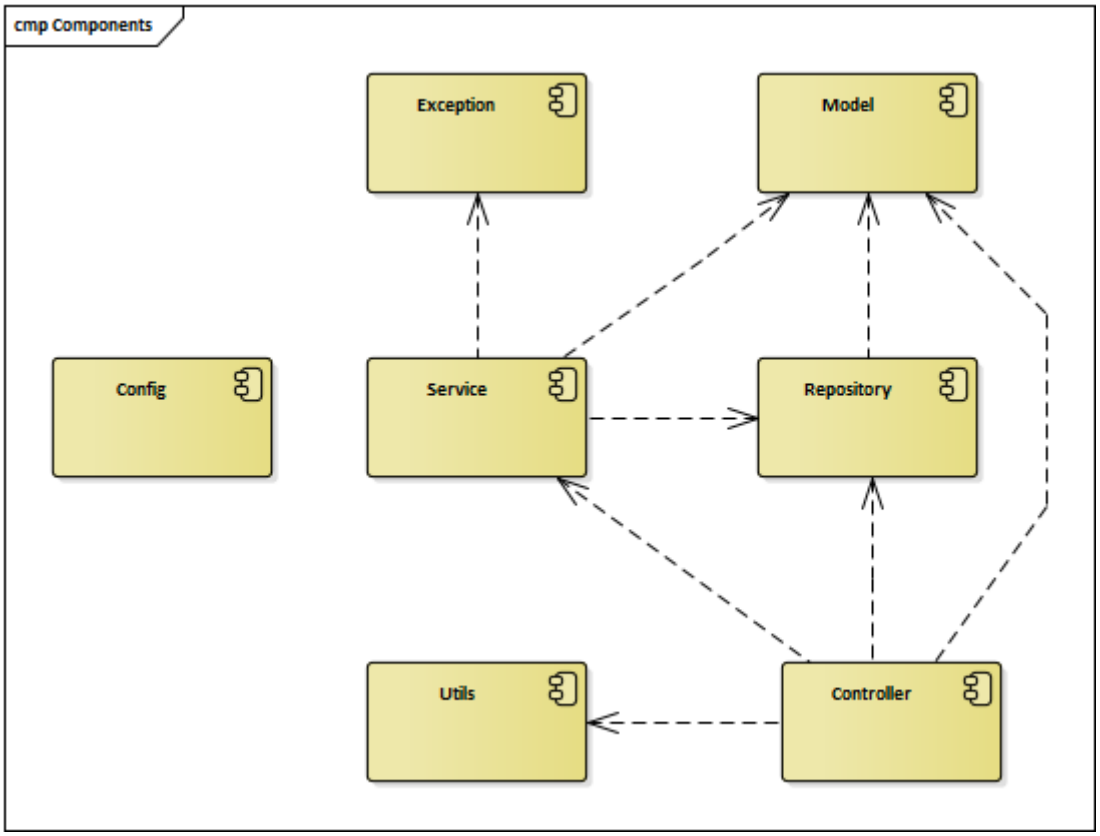


Figura 6.3. Diagrama de clases global

6.2.1.1 Diagrama de clases del modelo

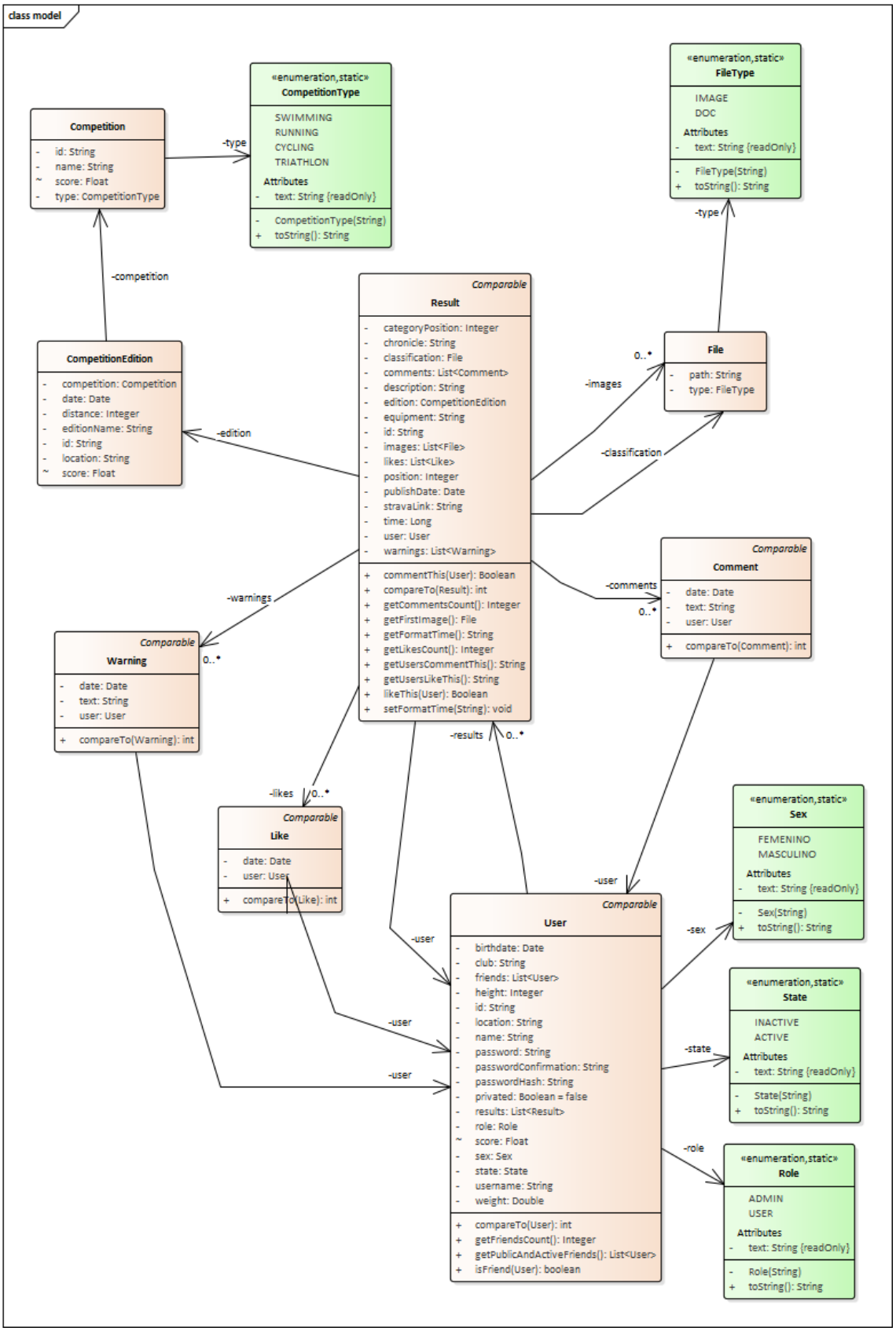


Figura 6.4. Diagrama del modelo

6.2.1.2 Diagrama de clases de repositorios

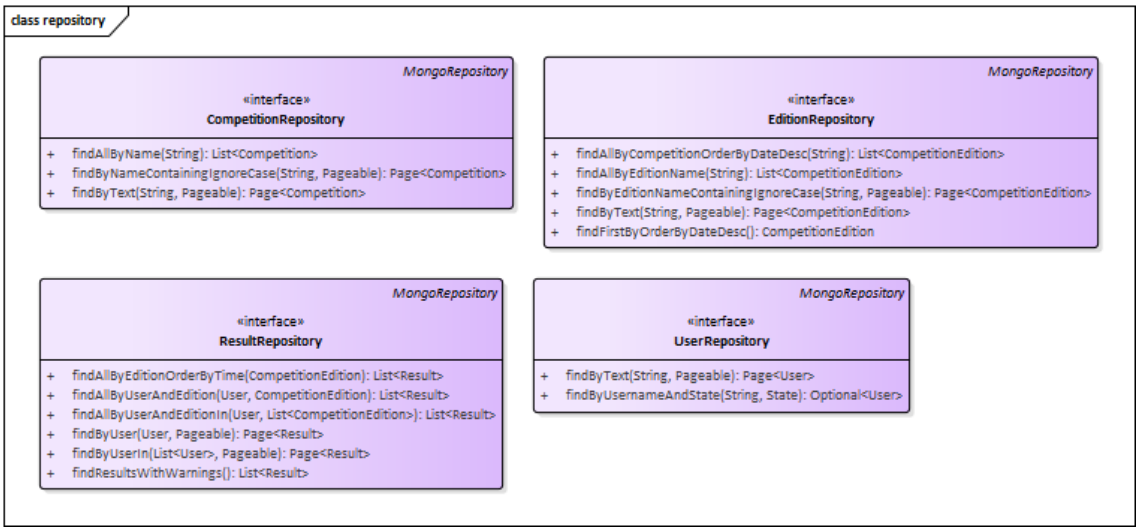


Figura 6.5 Diagrama de clases de repositorios

6.2.1.3 Diagrama de clases de los servicios

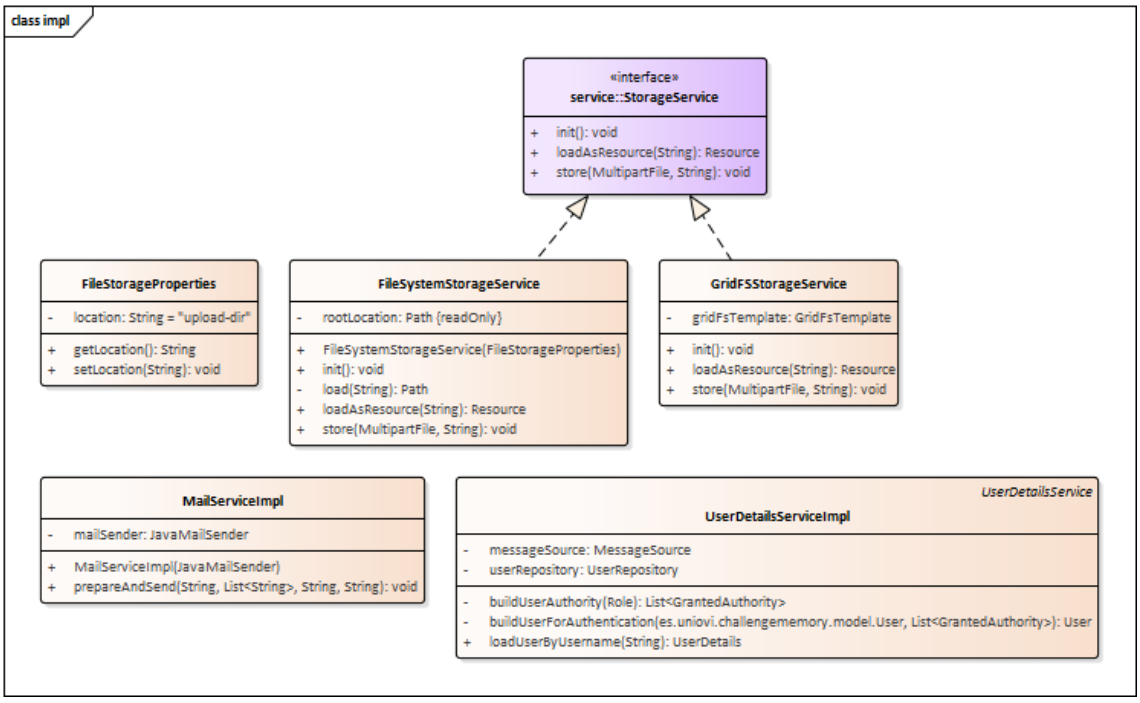


Figura 6.6. Diagrama de clases de los servicios

6.2.1.4 Diagrama de clases del controlador

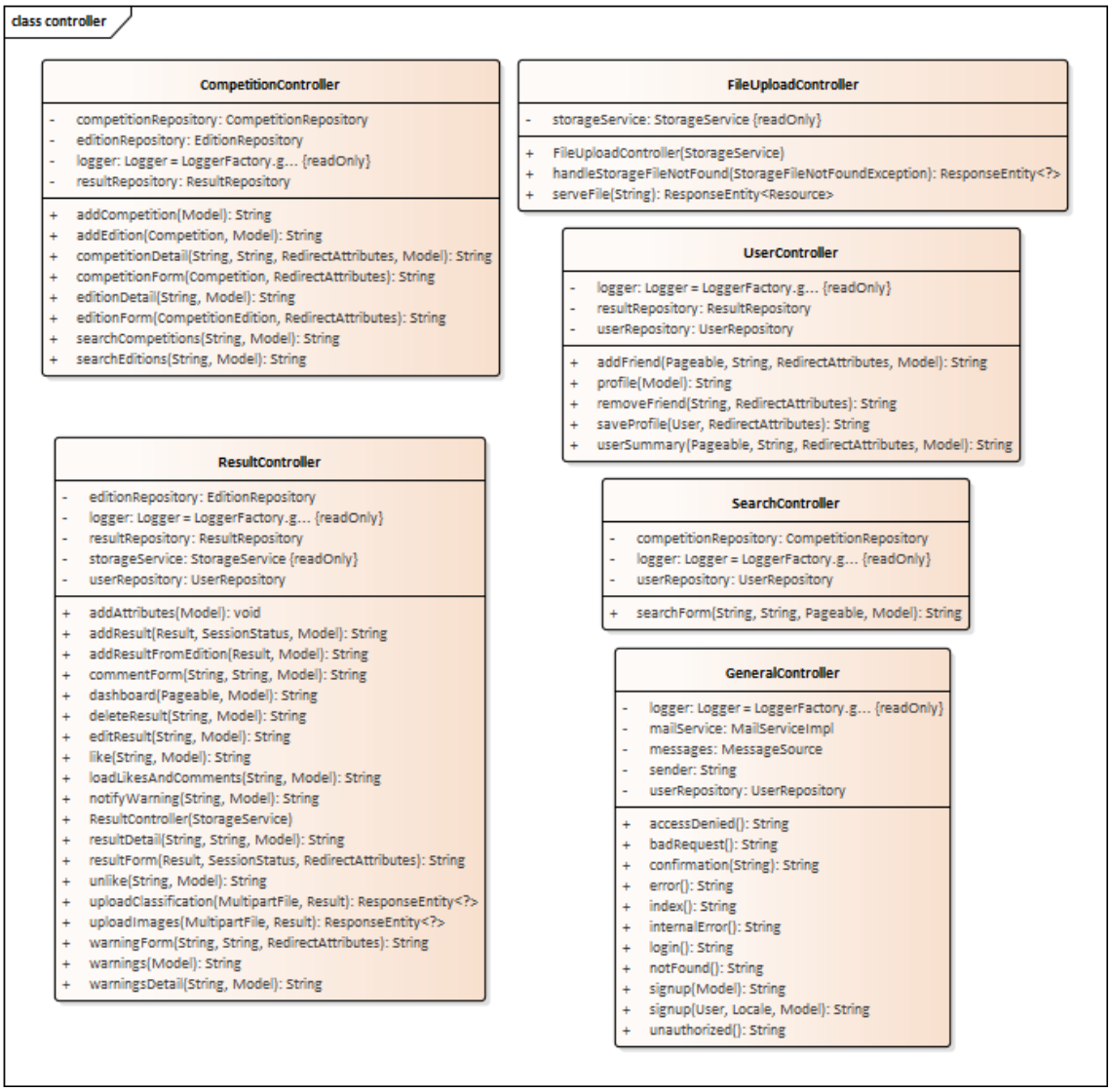


Figura 6.7. Diagrama de clases del controlador

6.2.1.5 Diagrama de clases de excepciones

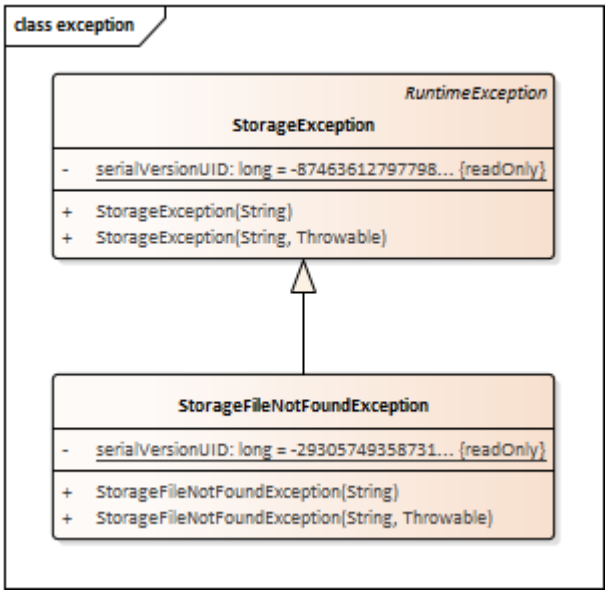


Figura 6.8. Diagrama de clases de excepciones

6.2.1.6 Diagrama de clases de utilidades

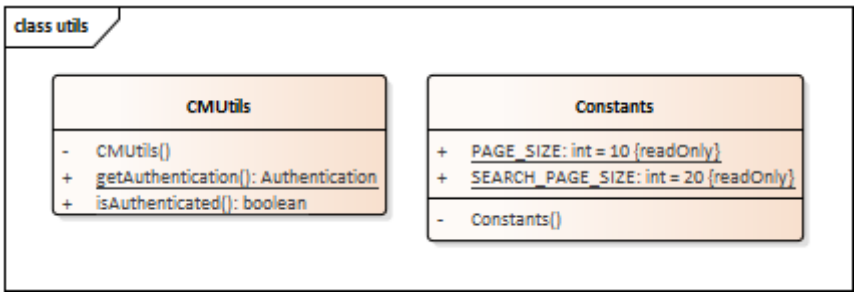


Figura 6.9. Diagrama de clases utilidades

6.2.1.7 Diagrama de clases de configuración

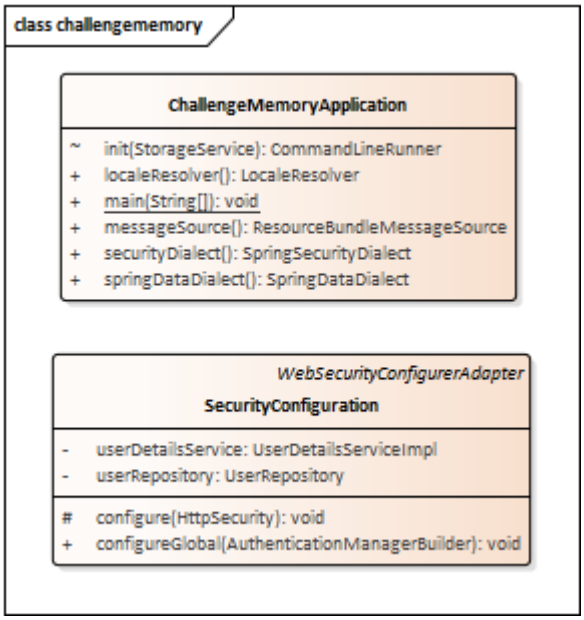


Figura 6.9. Diagrama de clases de configuración

6.3 Diagramas de Interacción (Secuencia)

6.3.1 Ver página principal

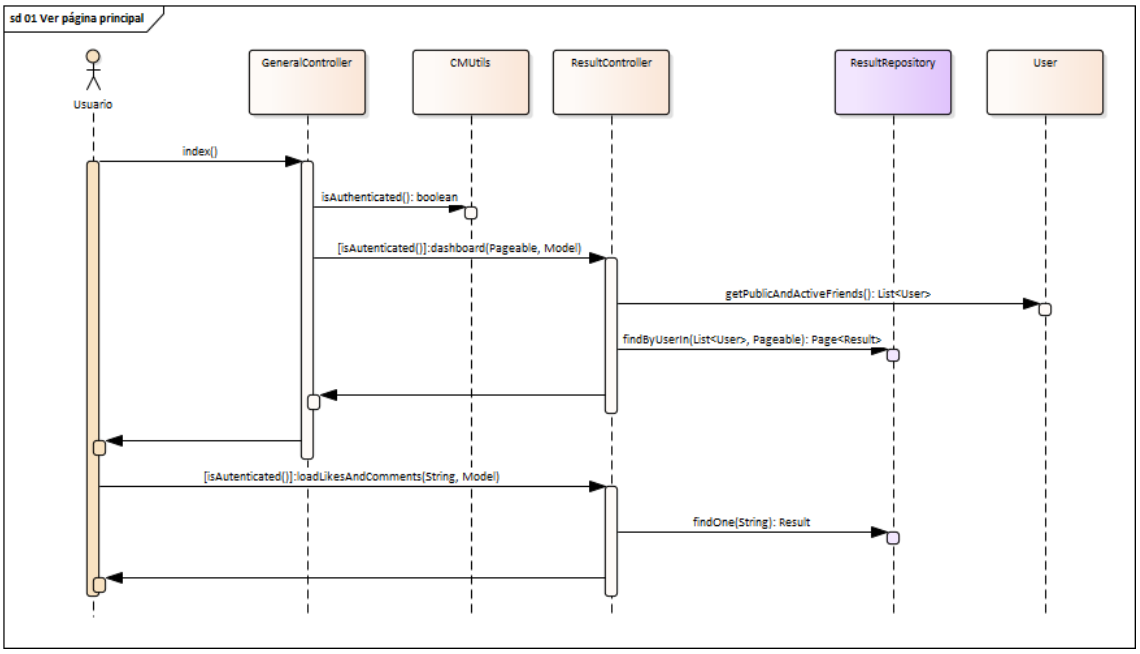


Figura 6.10. Diagrama de secuencia del caso de uso Ver página principal

6.3.2 Registrar

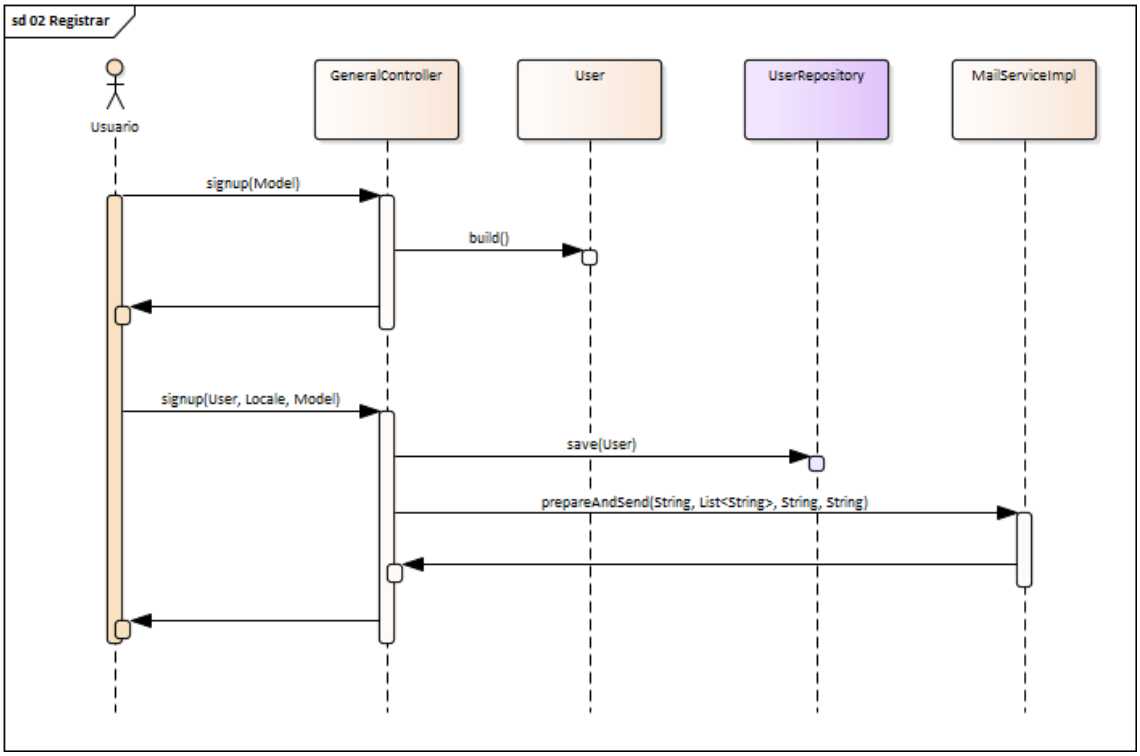


Figura 6.11. Diagrama de secuencia del caso de uso Registrar

6.3.3 Confirmar registro

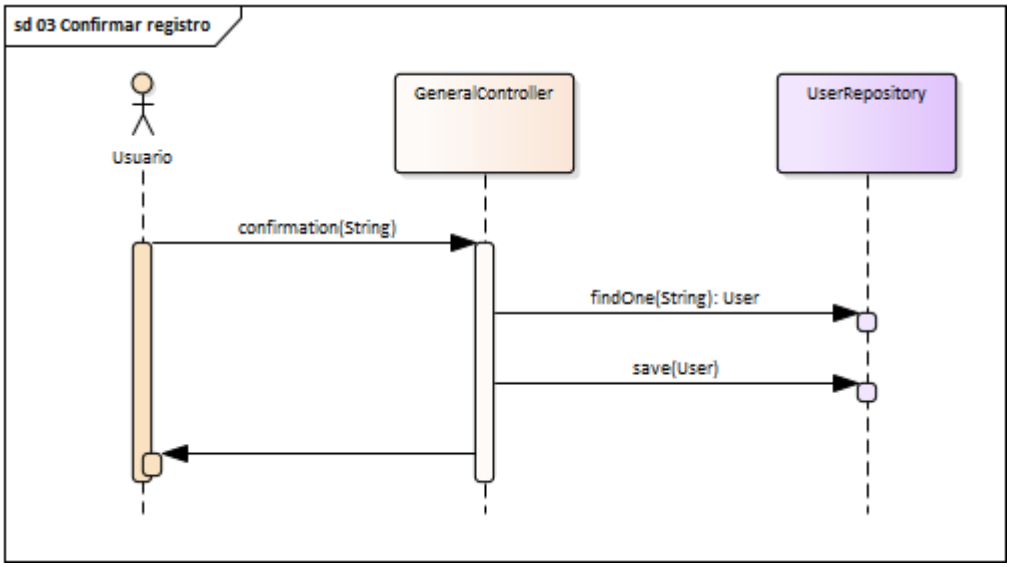


Figura 6.12. Diagrama de secuencia del caso de uso Confirmar registro

6.3.4 Autenticar

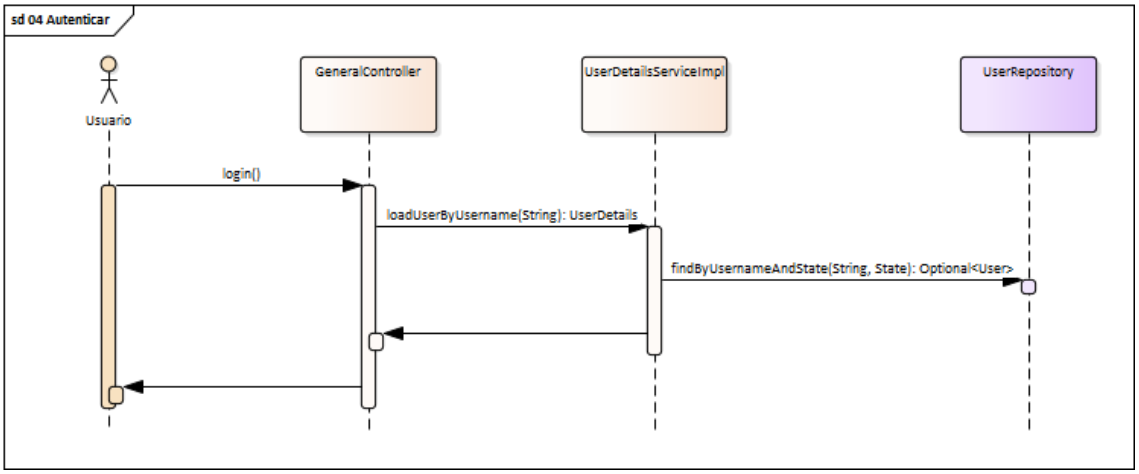


Figura 6.13. Diagrama de secuencia del caso de uso Autenticar

6.3.5 Editar perfil y privacidad

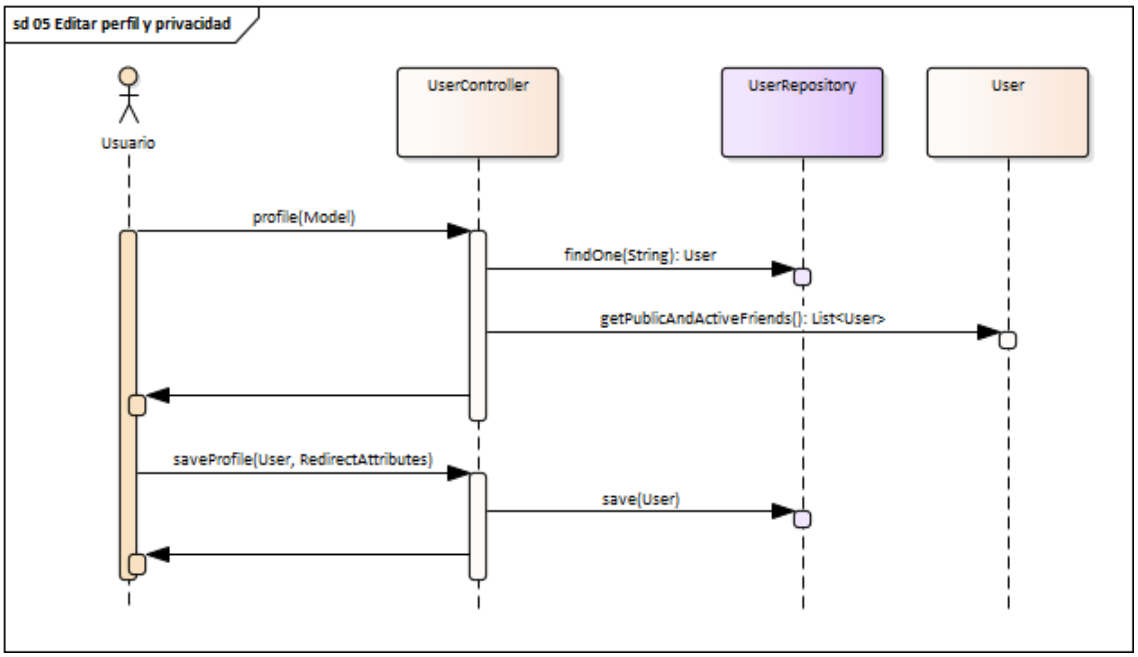


Figura 6.14. Diagrama de secuencia del caso de uso Editar perfil

6.3.6 Crear competición

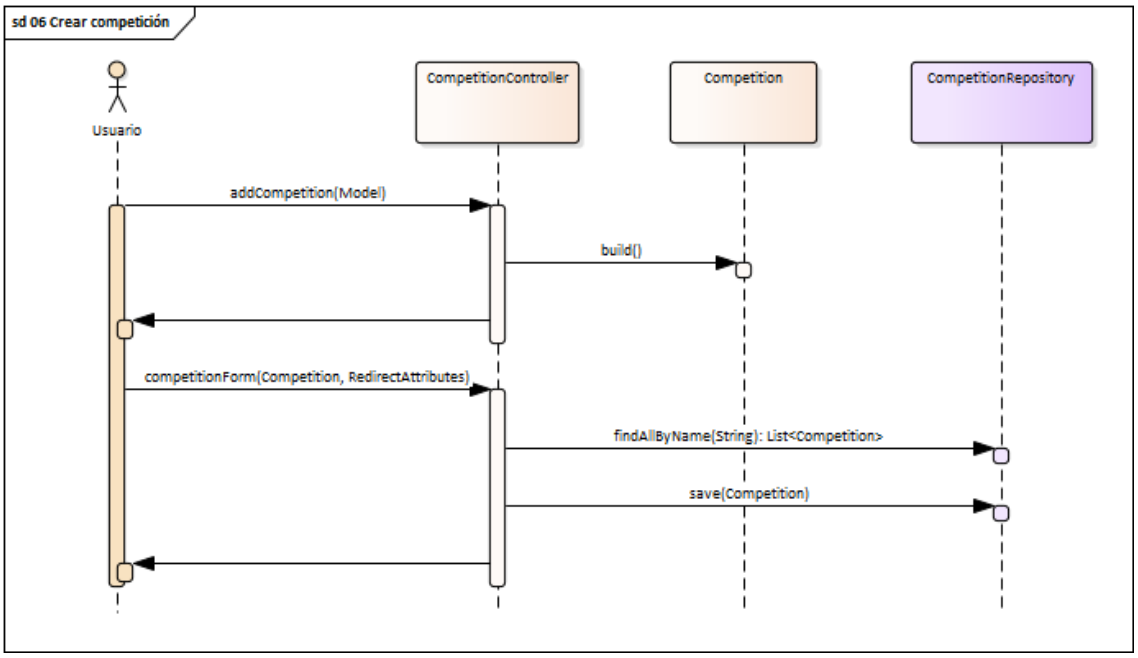


Figura 6.15. Diagrama de secuencia del caso de uso Crear competición

6.3.7 Crear edición de competición

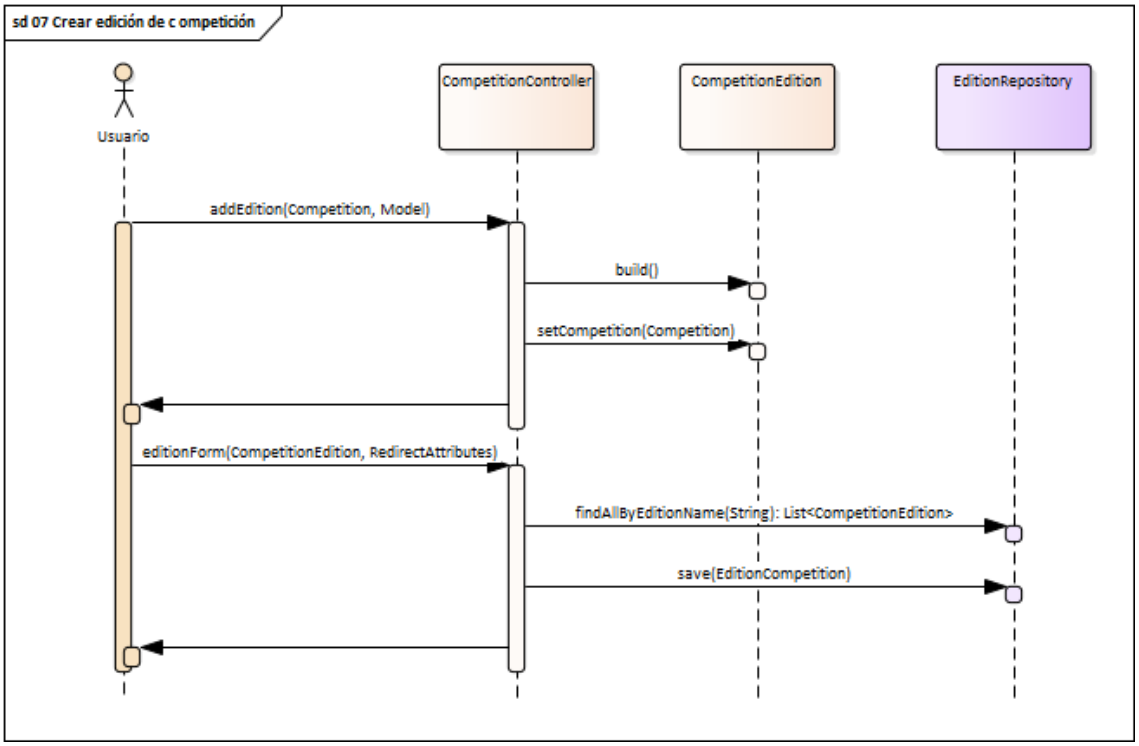


Figura 6.16. Diagrama de secuencia del caso de uso Crear edición

6.3.8 Crear resultado

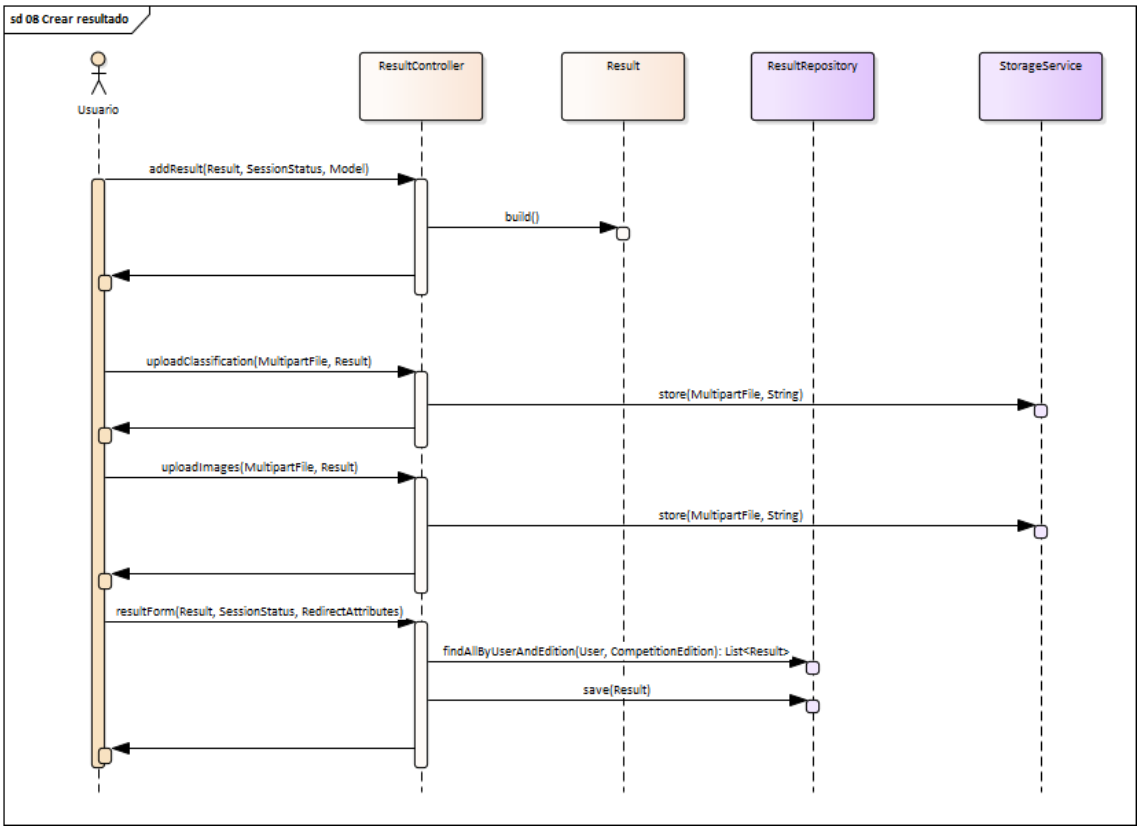


Figura 6.17. Diagrama de secuencia del caso de uso Crear resultado

6.3.9 Editar resultado

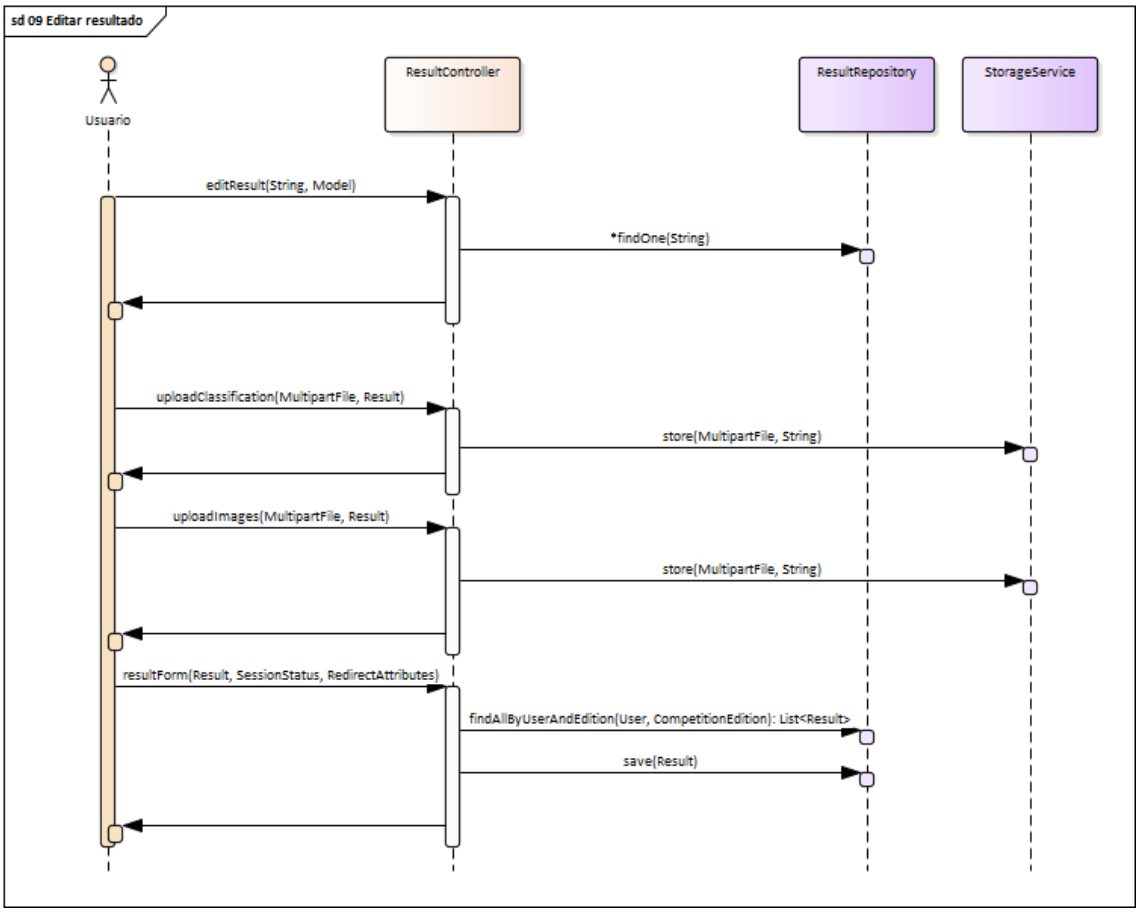


Figura 6.18. Diagrama de secuencia del caso de uso Editar resultado

6.3.10 Borrar resultado

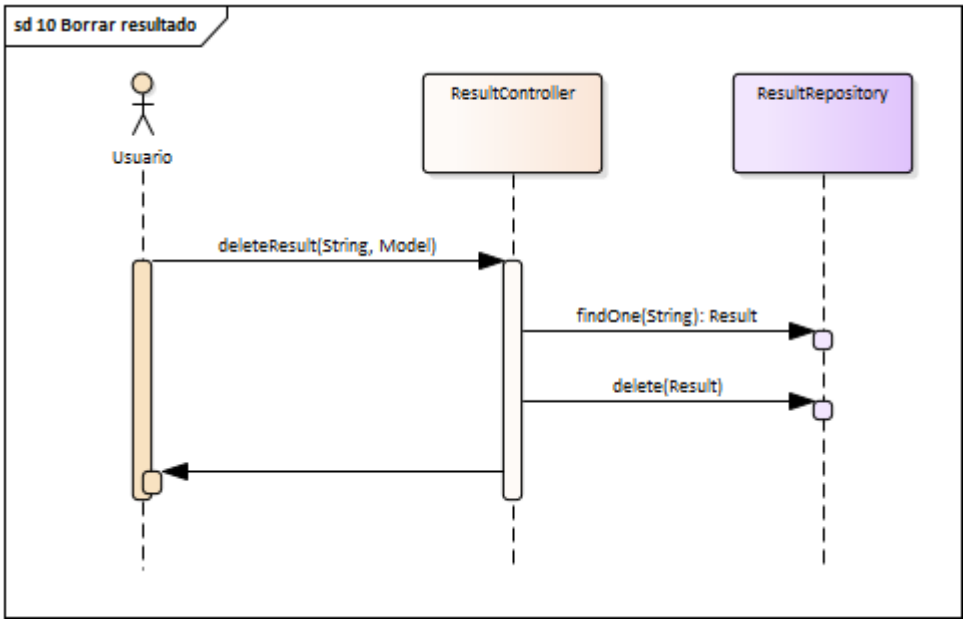


Figura 6.19. Diagrama de secuencia del caso de uso Borrar resultado

6.3.11 Ver resultado

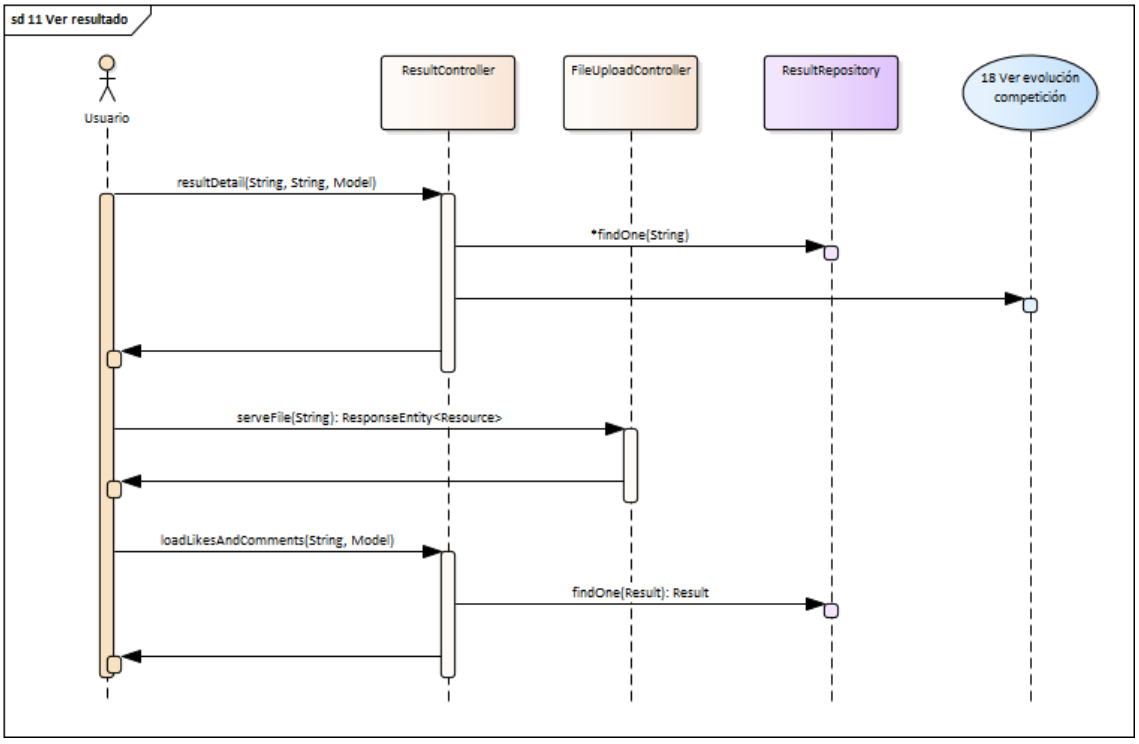


Figura 6.20. Diagrama de secuencia del caso de uso Ver resultado

6.3.12 Ver usuario

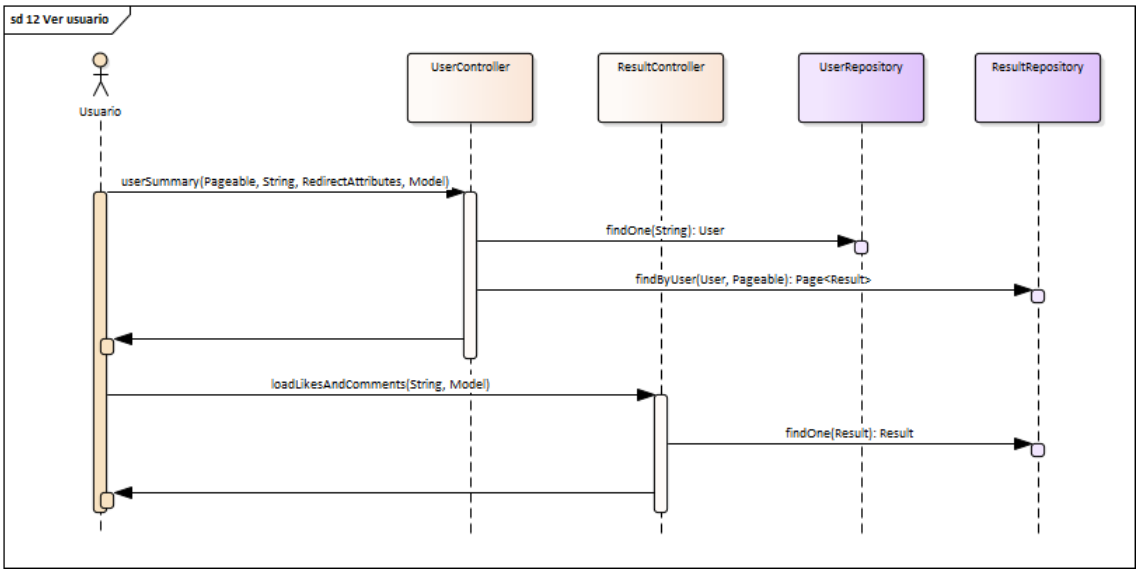


Figura 6.21. Diagrama de secuencia del caso de uso Ver usuario

6.3.13 Añadir usuario amigo

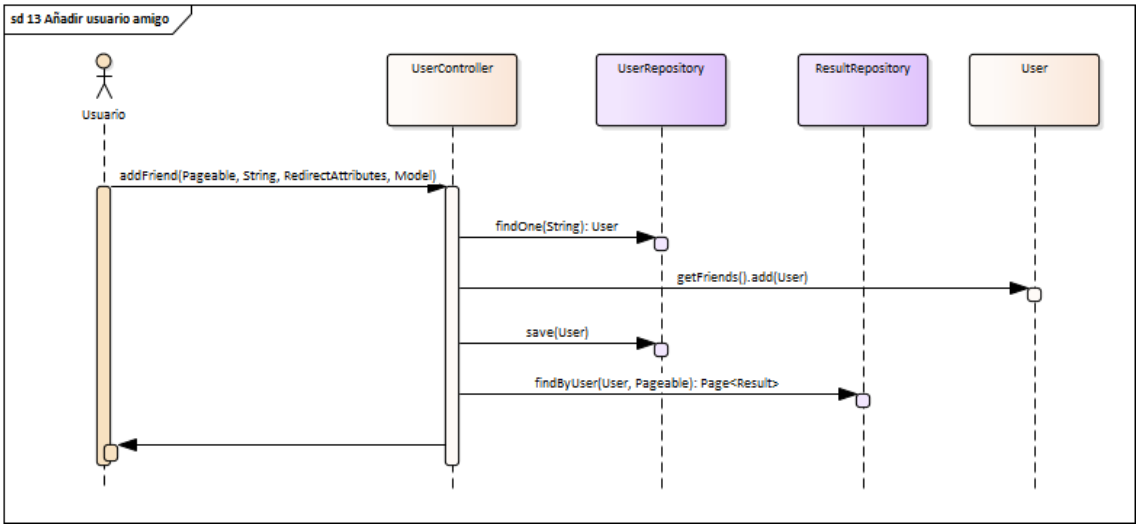


Figura 6.22. Diagrama de secuencia del caso de uso Añadir amigo

6.3.14 Eliminar usuario amigo

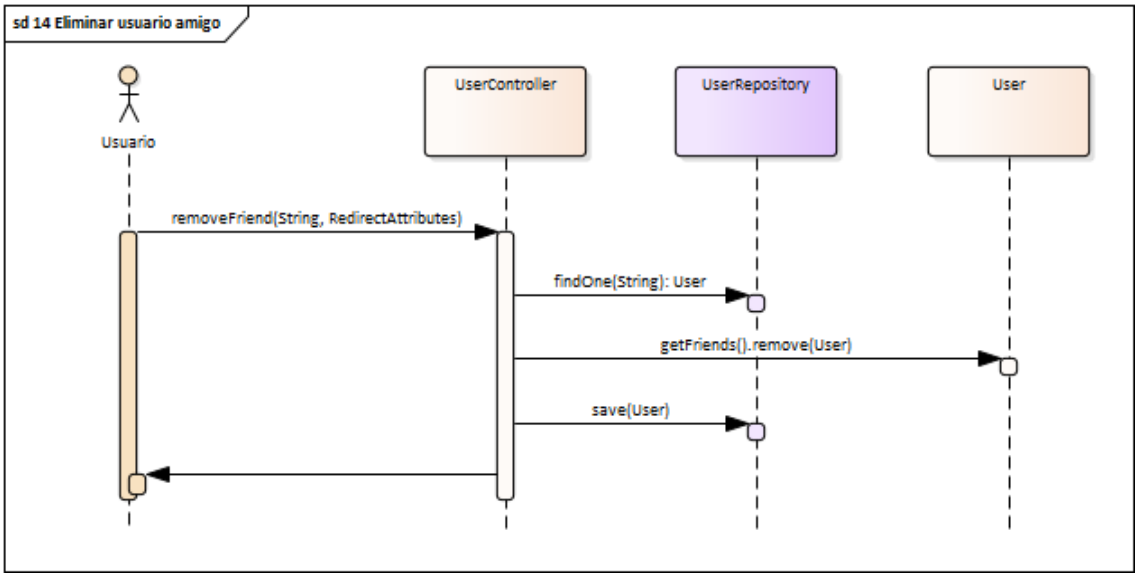


Figura 6.22. Diagrama de secuencia del caso de uso Eliminar amigo

6.3.15 Avisar resultado falso

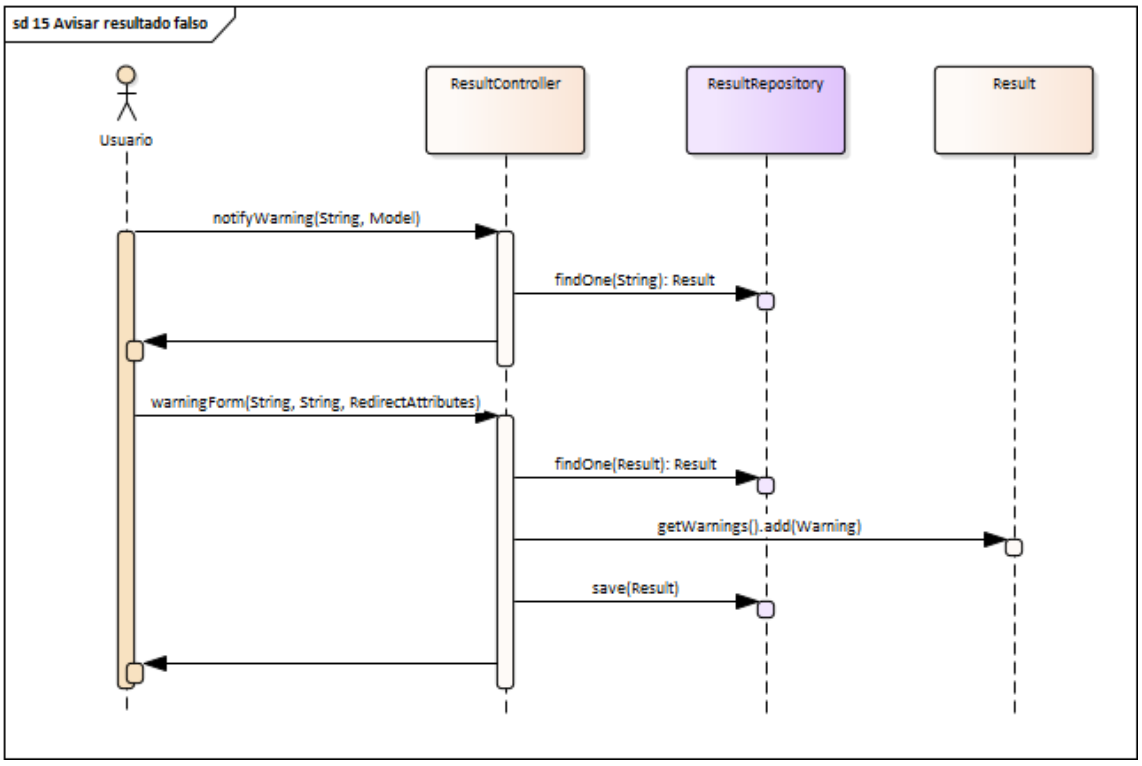


Figura 6.23. Diagrama de secuencia del caso de uso Avisar resultado falso

6.3.16 Ver Competición

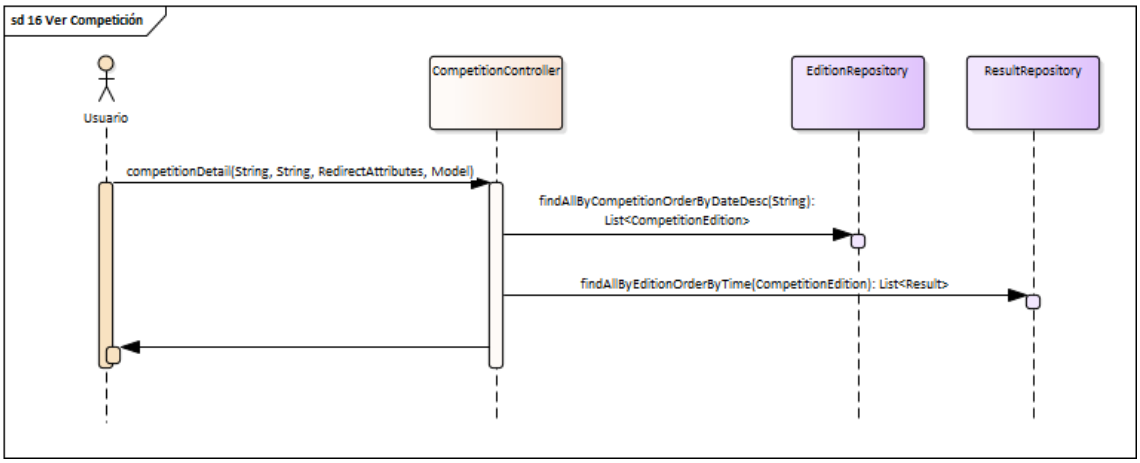


Figura 6.24. Diagrama de secuencia del caso de uso Ver competición

6.3.17 Ver clasificación de una edición

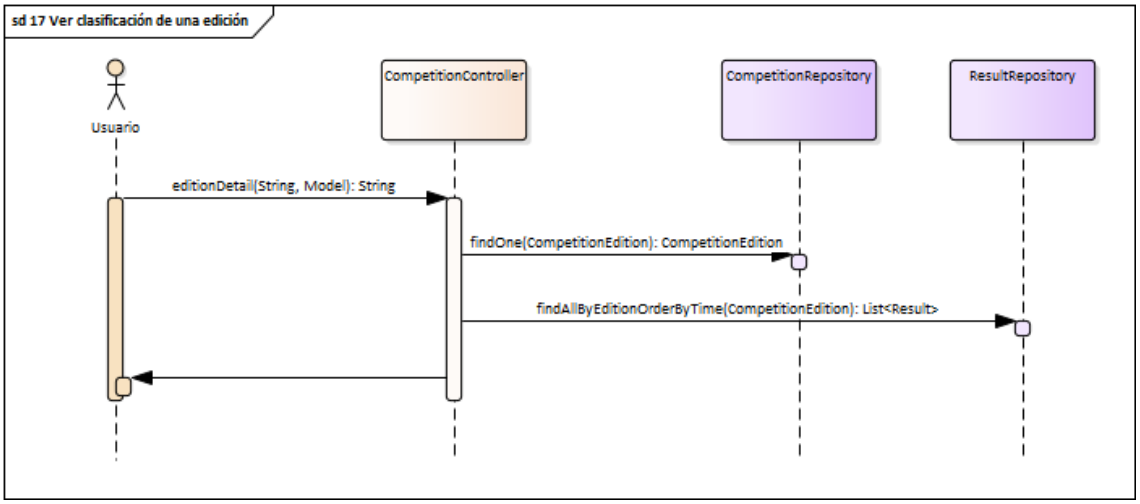


Figura 6.25. Diagrama de secuencia del caso de uso Ver clasificación edición

6.3.18 Ver evolución competición

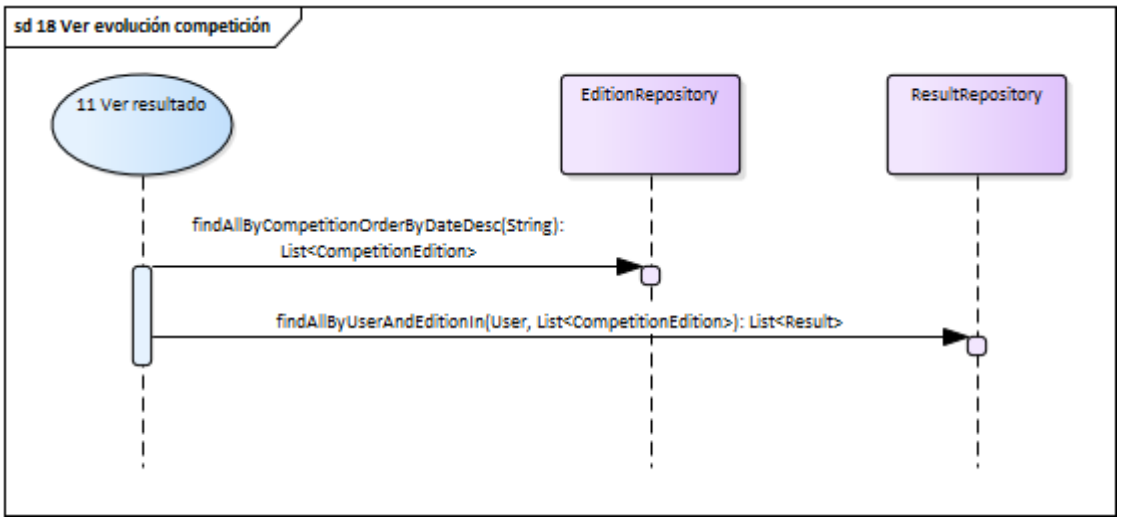


Figura 6.26. Diagrama de secuencia del caso de uso Ver evolución

6.3.19 Dar "Me gusta"

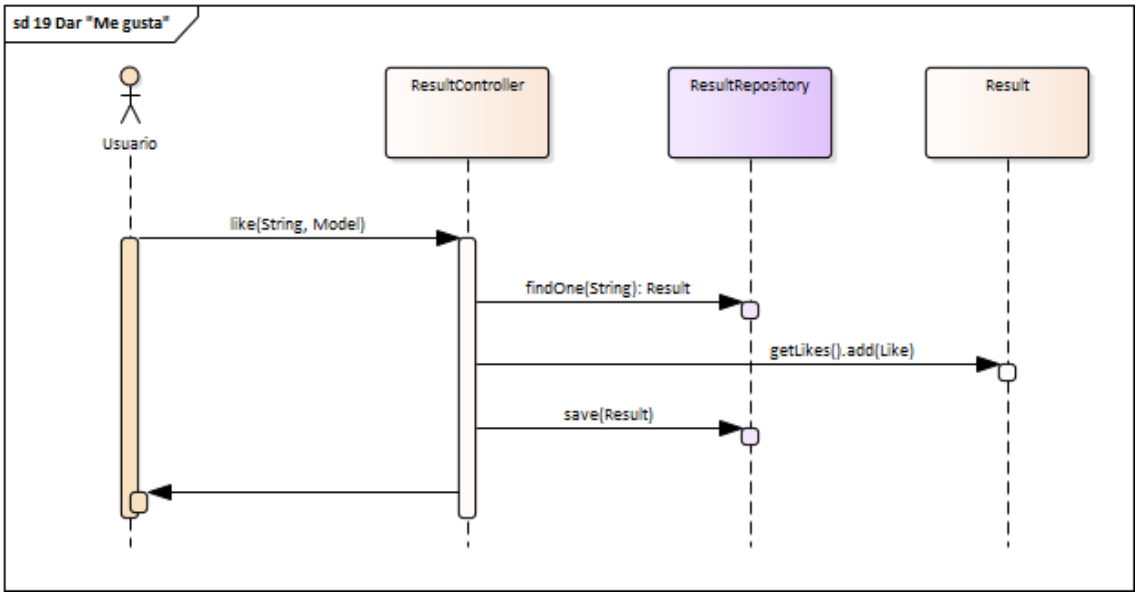


Figura 6.27. Diagrama de secuencia del caso de uso Dar "me gusta"

6.3.20 Quitar "Me gusta"

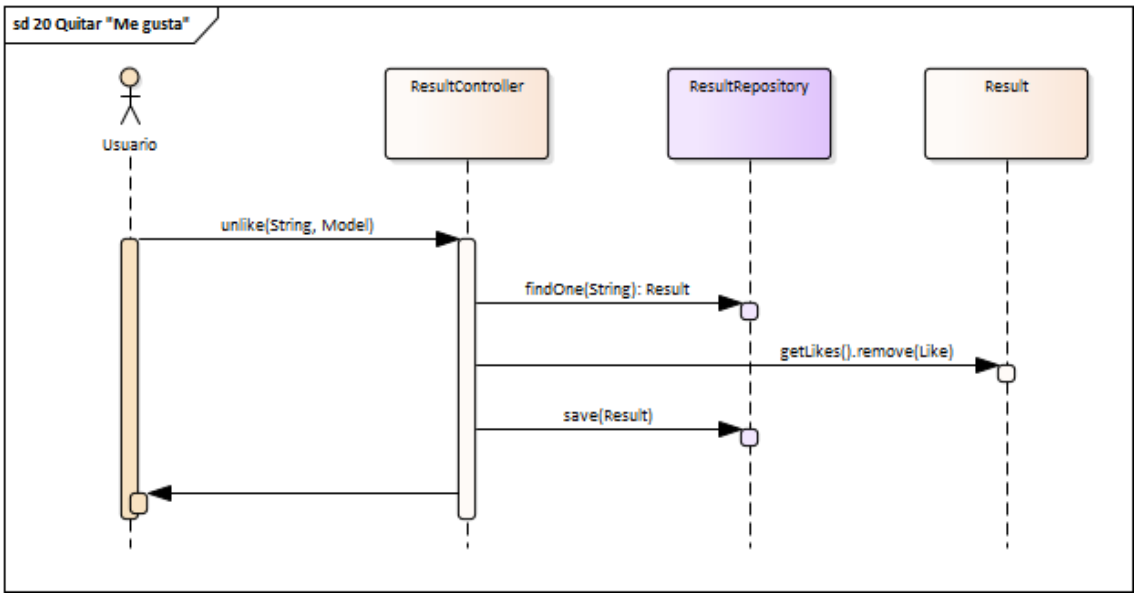


Figura 6.28. Diagrama de secuencia del caso de uso Quitar "me gusta"

6.3.21 Comentar resultado

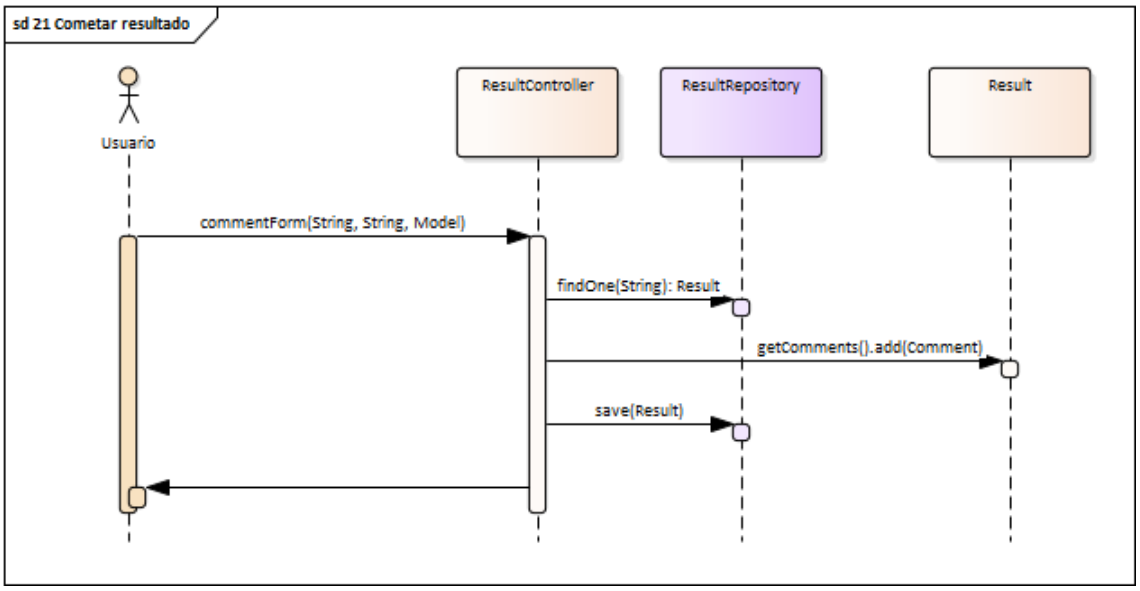


Figura 6.29. Diagrama de secuencia del caso de uso Comentar resultado

6.3.22 Buscar competición

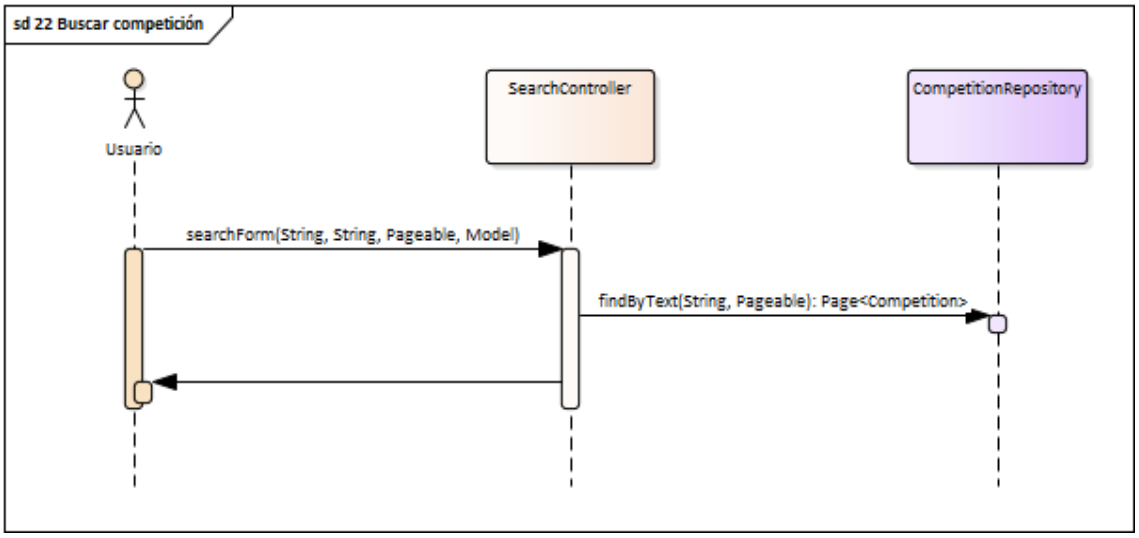


Figura 6.30. Diagrama de secuencia del caso de uso Buscar competición

6.3.23 Buscar usuario

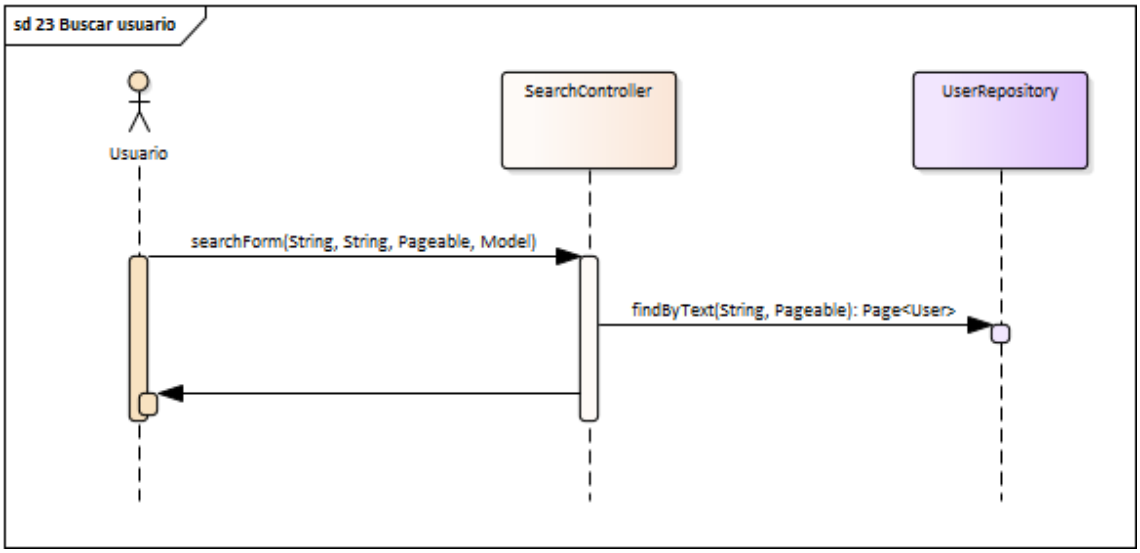


Figura 6.31. Diagrama de secuencia del caso de uso usuario

6.3.24 Listar resultados con aviso

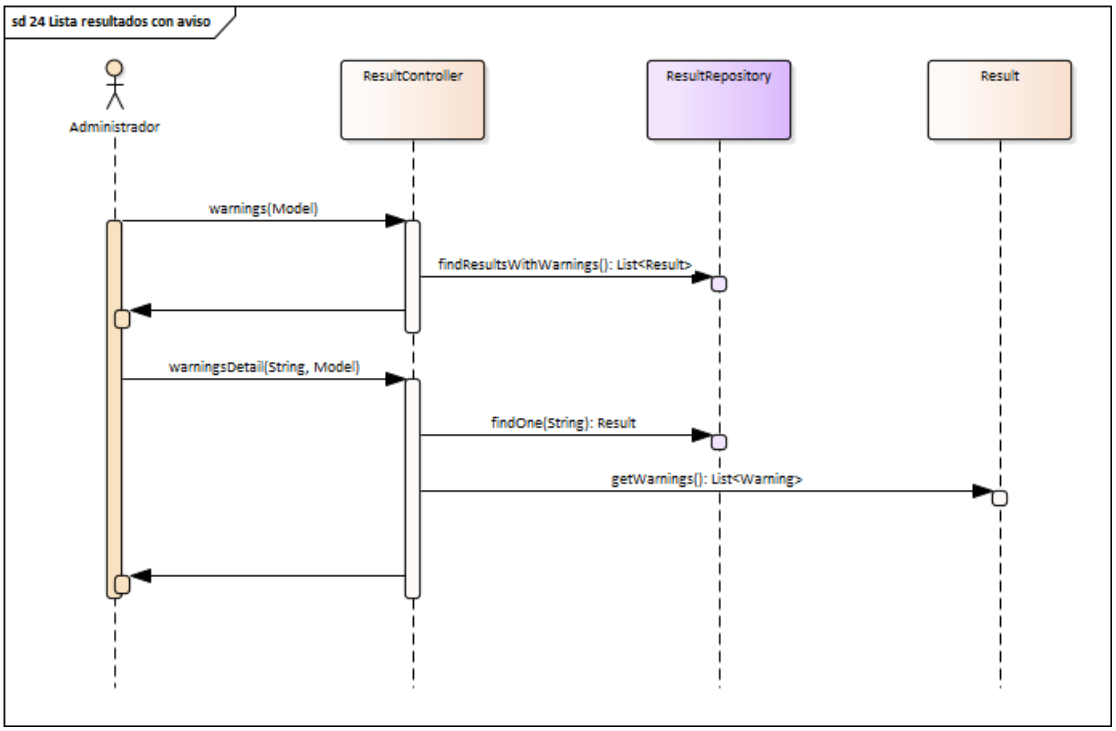


Figura 6.32. Diagrama de secuencia del caso de uso Listar avisos

6.3.25 Desactivar usuario

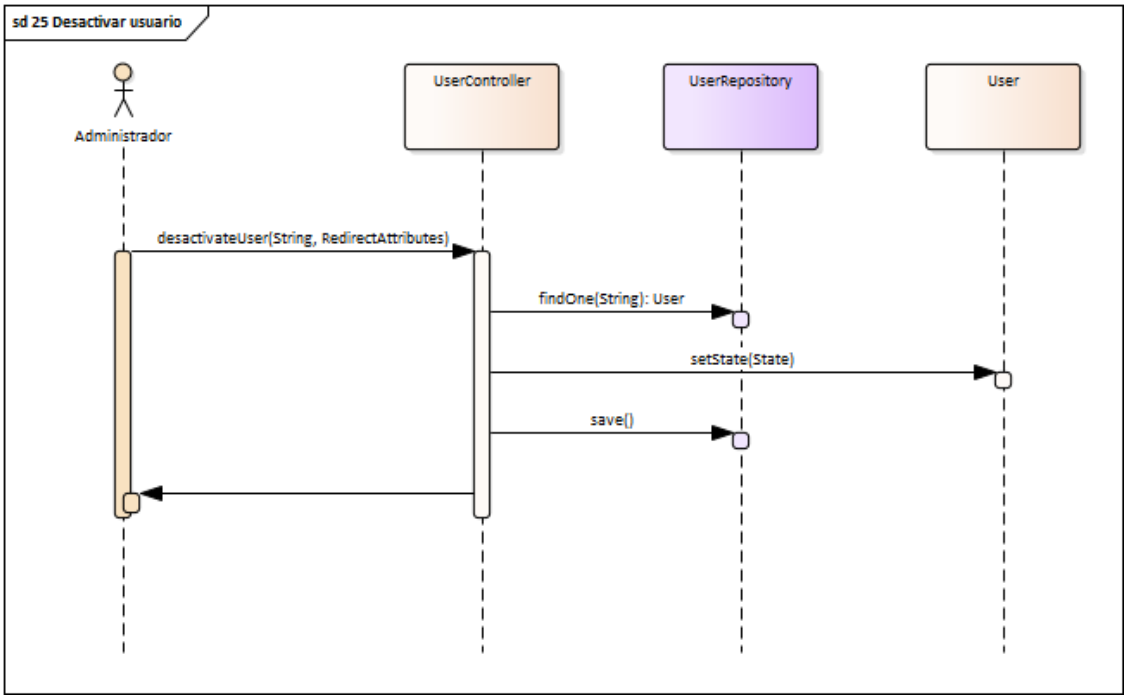


Figura 6.33. Diagrama de secuencia del caso de uso Desactivar usuario

6.4 Diseño de la Base de Datos

6.4.1 Descripción del SGBD (MongoDB)

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección (concepto similar a una tabla de una base de datos relacional), pueden tener esquemas diferentes.

Los datos se almacenan de forma semi estructurada. Las colecciones en MongoDB no necesitan definir un esquema, aunque es importante que diseñemos uno a seguir. MongoDB es especialmente útil en entornos que requieran escalabilidad. Con sus opciones de replicación y sharding, que son muy sencillas de configurar, podemos conseguir un sistema que escale horizontalmente sin demasiados problemas.

Adicionalmente, nuestro sistema utiliza MongoDB para el almacenamiento de ficheros, a través de su implementación específica GridFS y que permite tratar a los ficheros como una entidad más.

6.4.2 Integración del SGBD

Gracias al uso de Spring Boot en nuestro sistema, la integración con este SGBD es algo sencillo y transparente para el desarrollador. Simplemente configurando el acceso al servidor MongoDB adecuadamente y utilizando la librería Mongo Repository, que implementa toda la funcionalidad necesaria para la realización de las operaciones más comunes en este tipo de sistemas, tendremos nuestro sistema operando con el SGBD. El único requisito adicional que tendremos que tener en cuenta, es indicar en nuestras clases del modelo, mediante la anotación java *@Document*, que dichas clases son las que componen nuestro modelo de datos para que sean mapeadas a documentos de MongoDB.

6.4.3 Esquema de documentos

Como hemos podido observar en las secciones anteriores, la configuración elegida en este proyecto, permite al desarrollador olvidarse en gran medida de las operaciones relacionadas con el SGBD, delegando su creación y gestión en librerías especialmente orientadas a ese fin. Como consecuencia directa de esta decisión, obtenemos una definición de documentos sumamente sencilla y fácil de explotar.

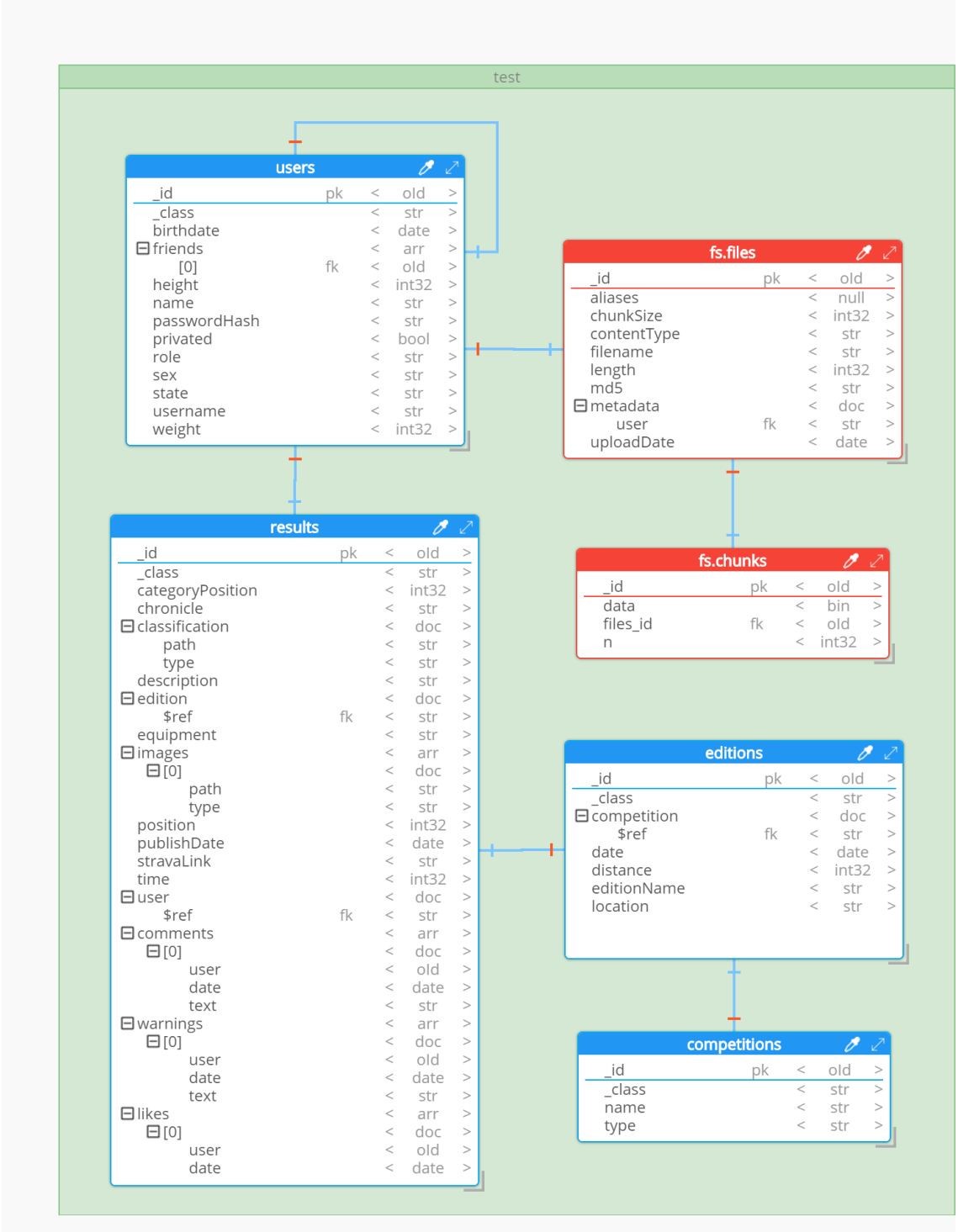


Figura 6.34. Esquema de documentos MongoDB

6.5 Diseño de la Interfaz

6.5.1 Pantalla de inicio

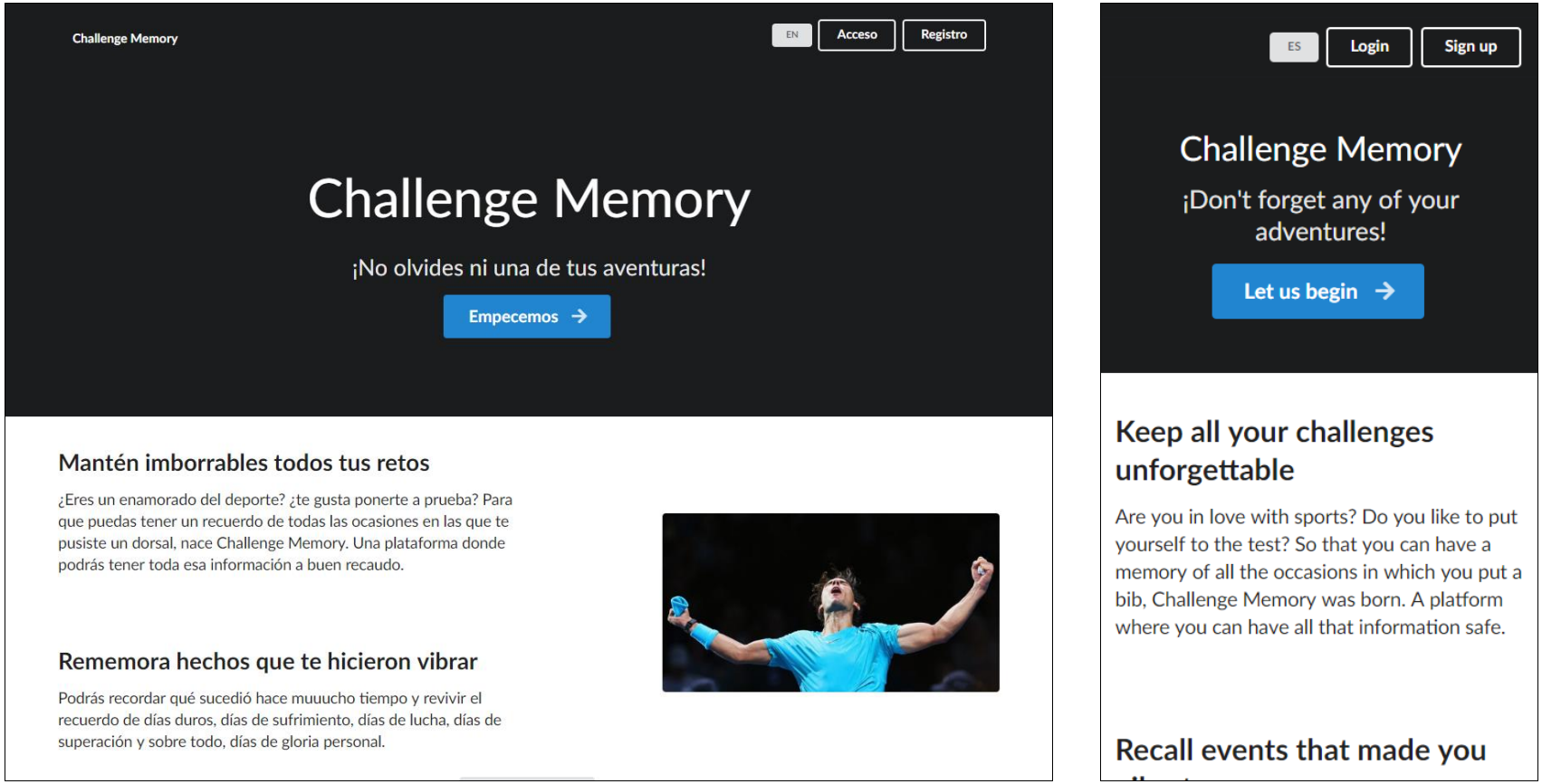
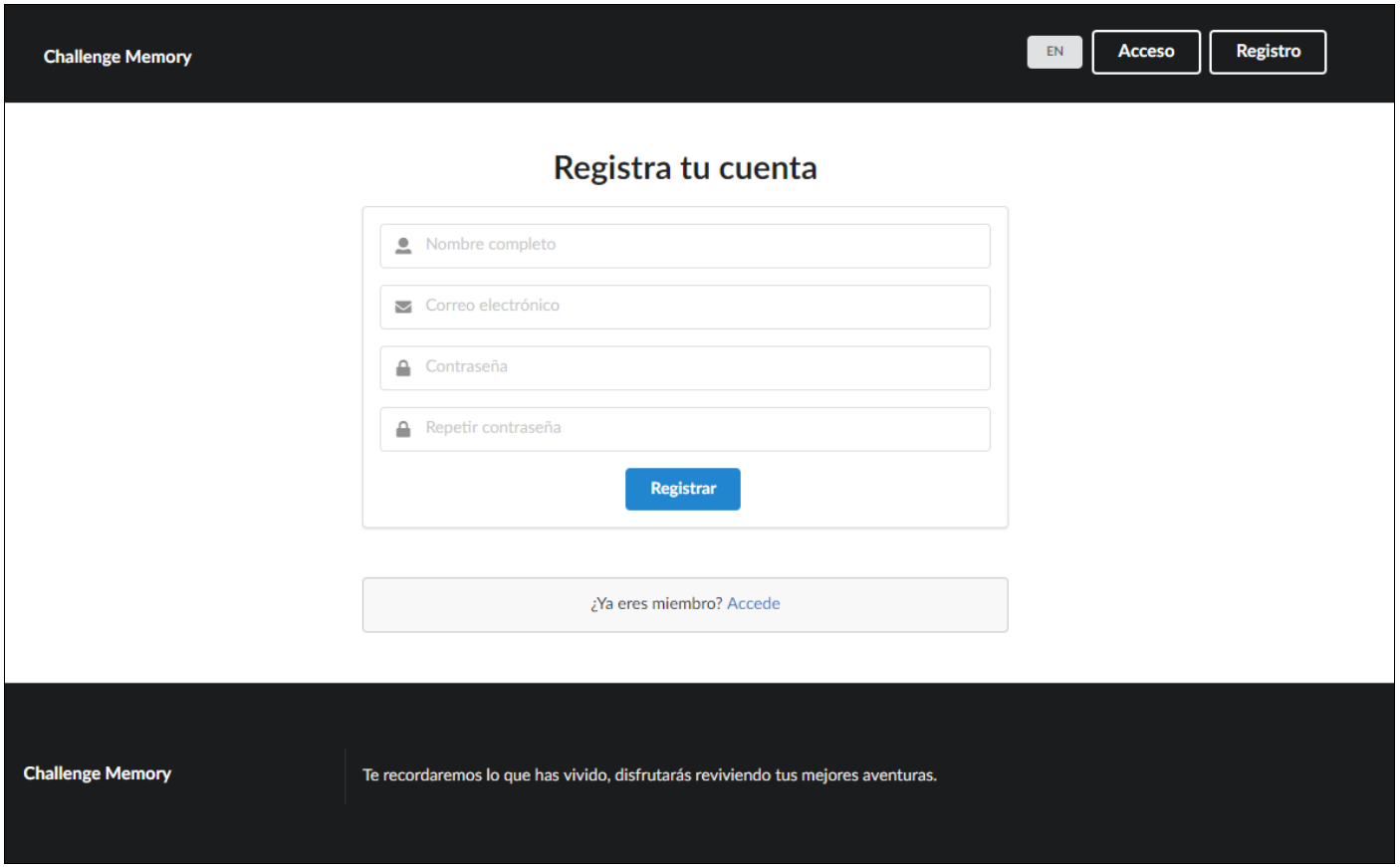


Figura 6.35. Formato PC y móvil

6.5.2 Pantalla de registro



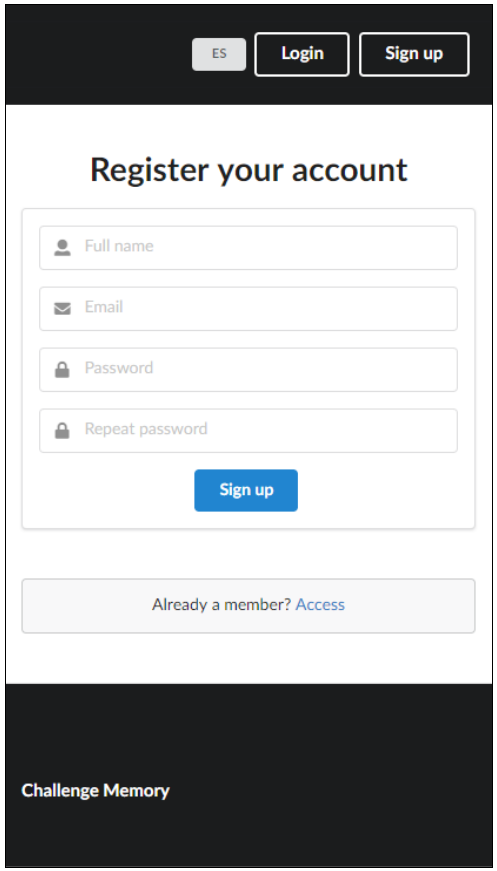


Figura 6.36. Formato PC y móvil

6.5.3 Pantalla de autenticación

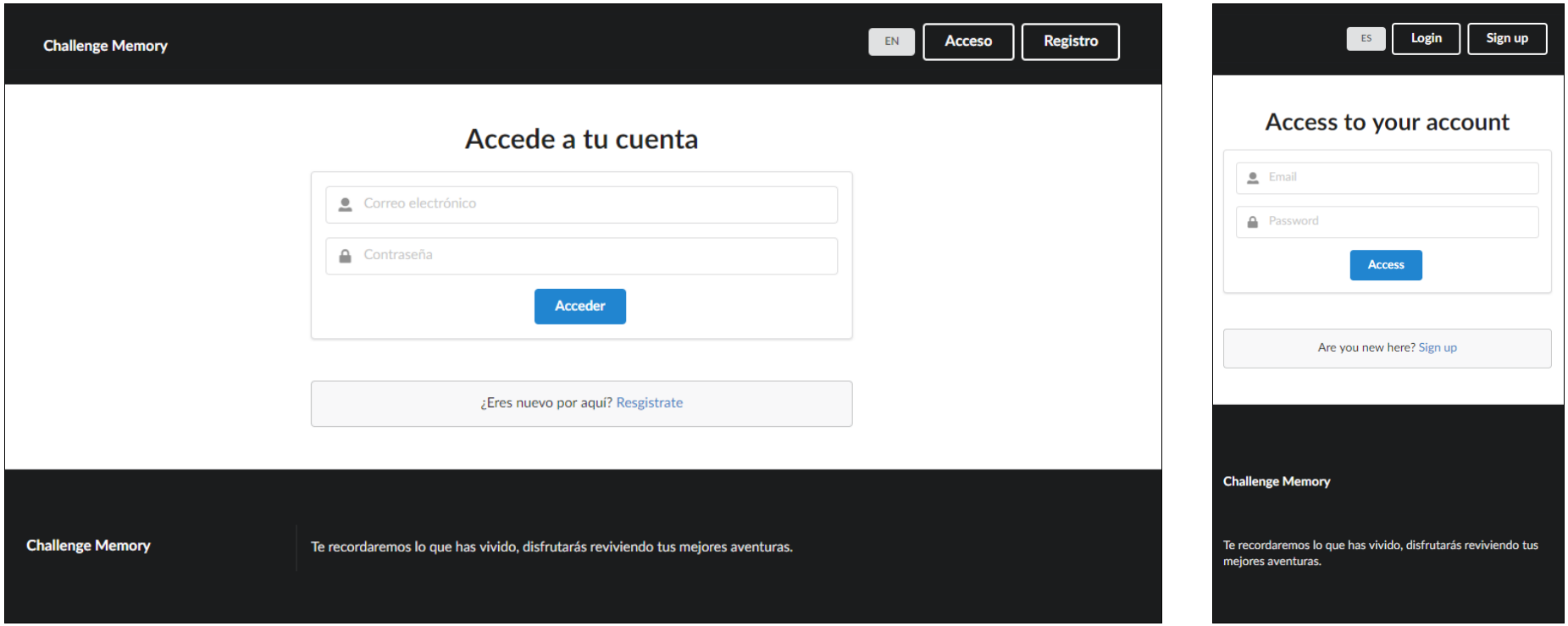


Figura 6.37. Formato PC y móvil

6.5.4 Pantalla de inicio de usuario

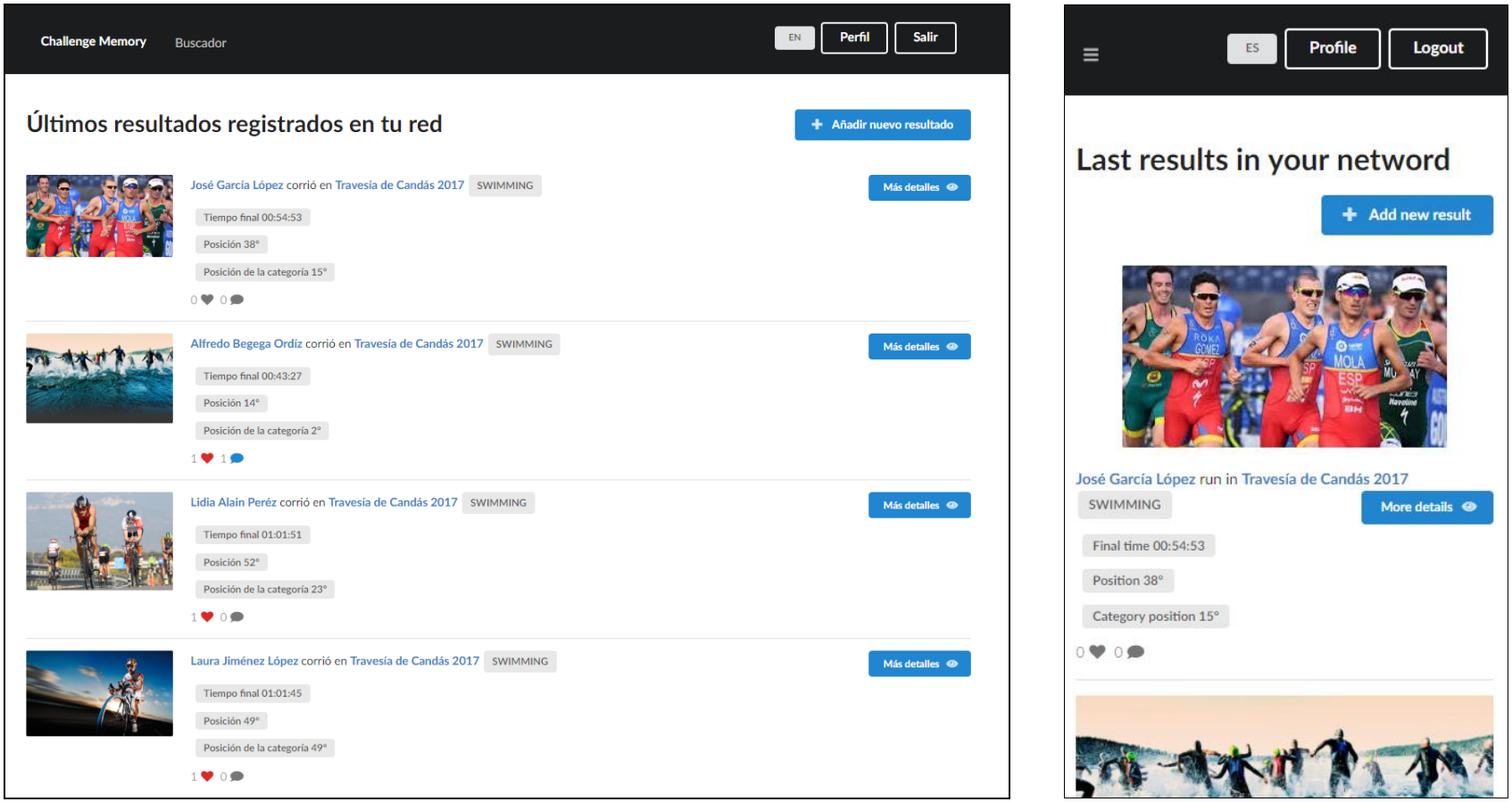


Figura 6.38. Formato PC y móvil

6.5.5 Pantalla de inicio de usuario (Administrador)

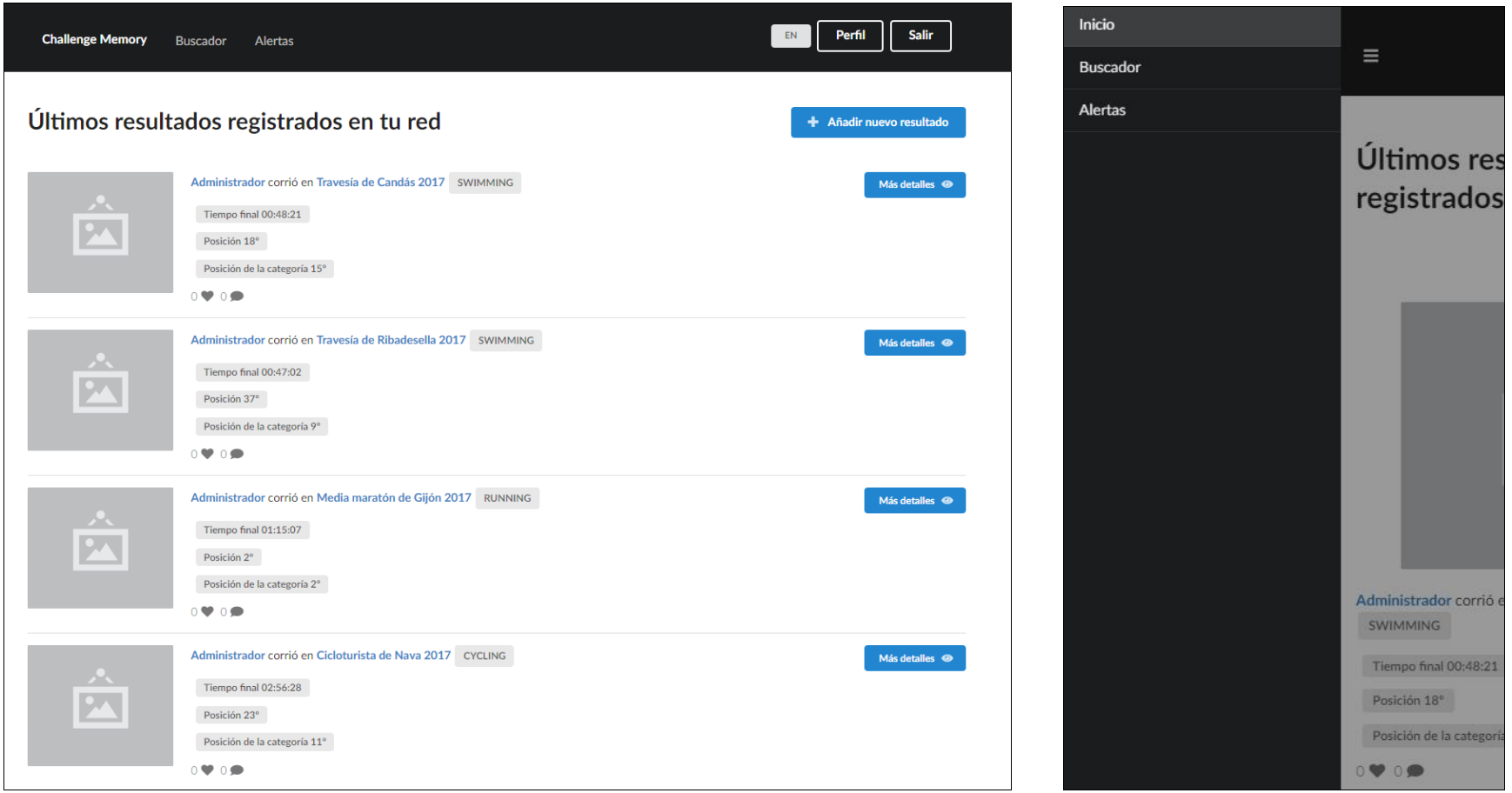


Figura 6.39. Formato PC y móvil

6.5.6 Pantalla de resultado

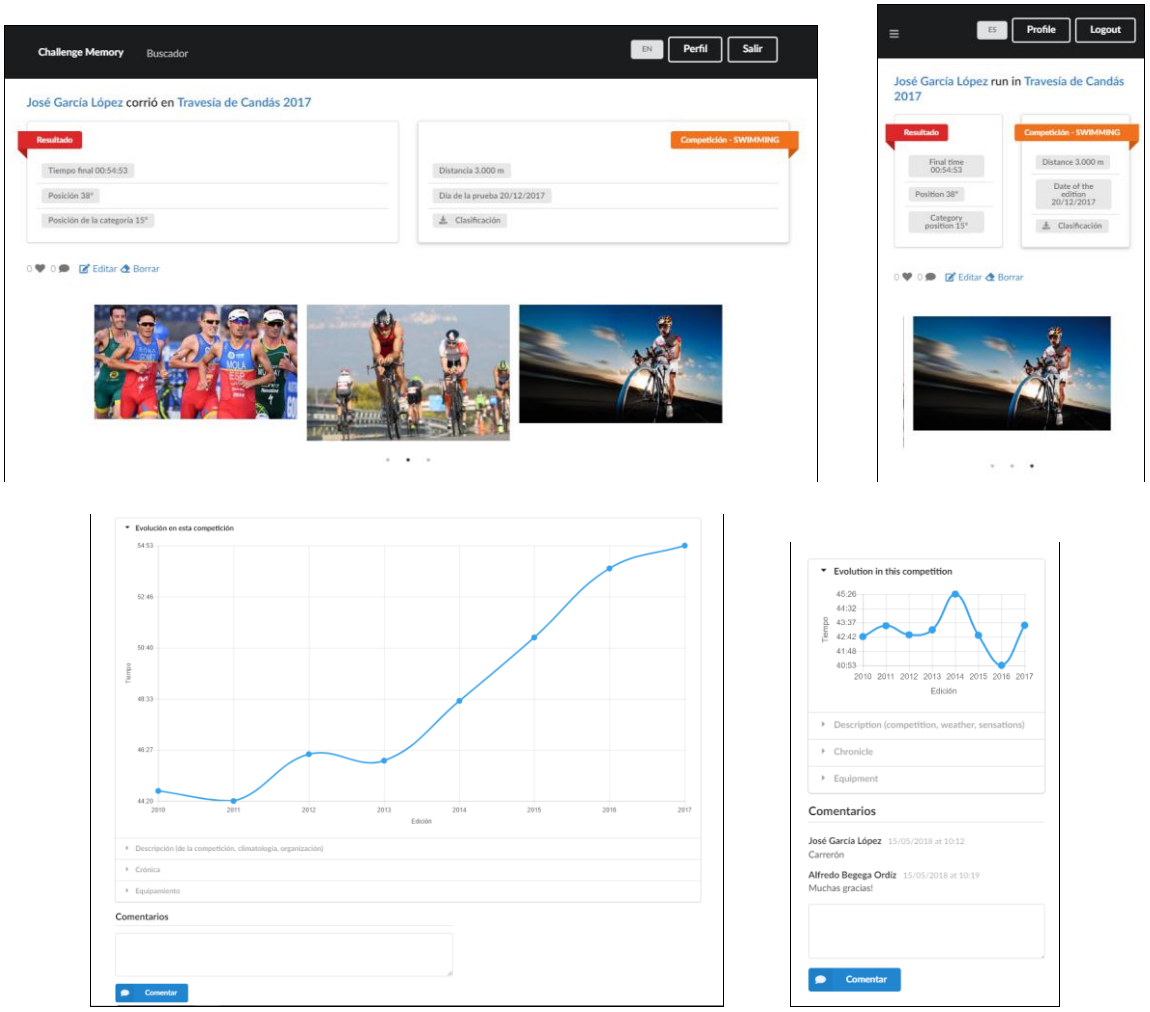


Figura 6.40. Formato PC y móvil

6.5.7 Pantalla de perfil

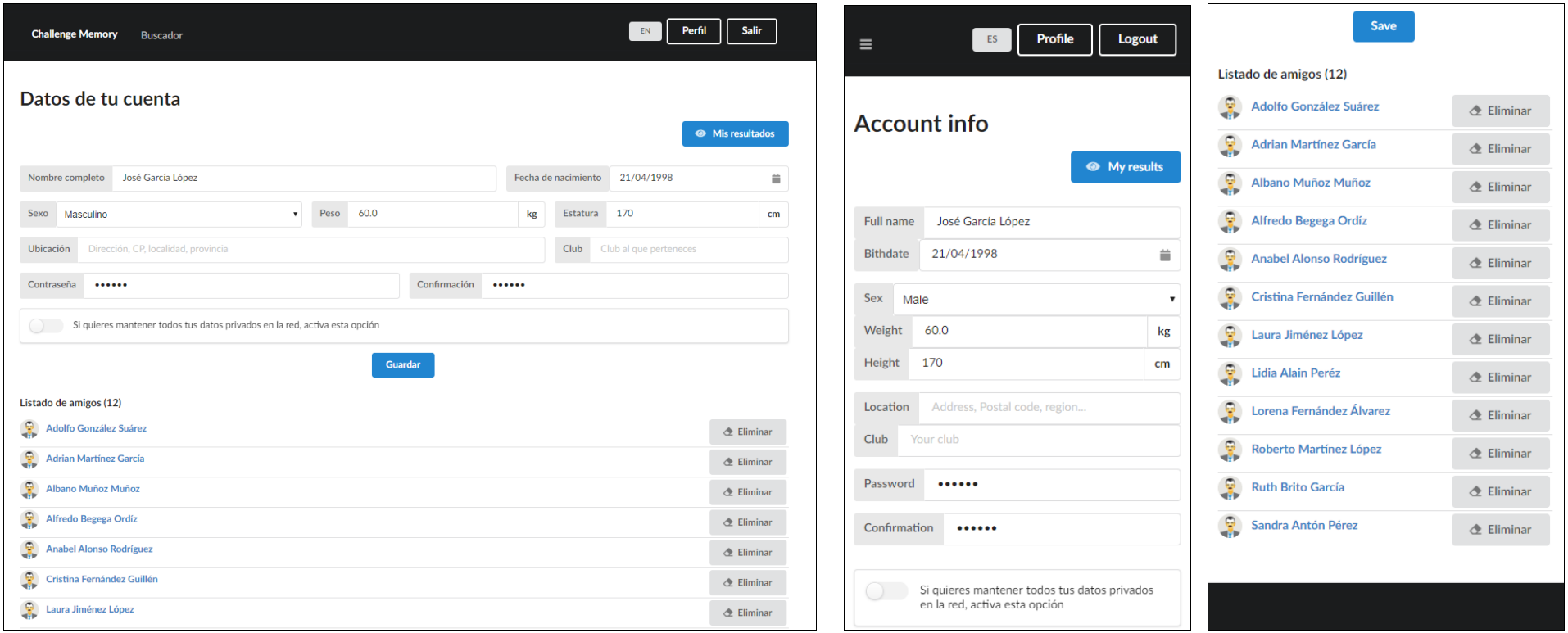


Figura 6.41. Formato PC y móvil

6.5.8 Pantalla buscador de competiciones

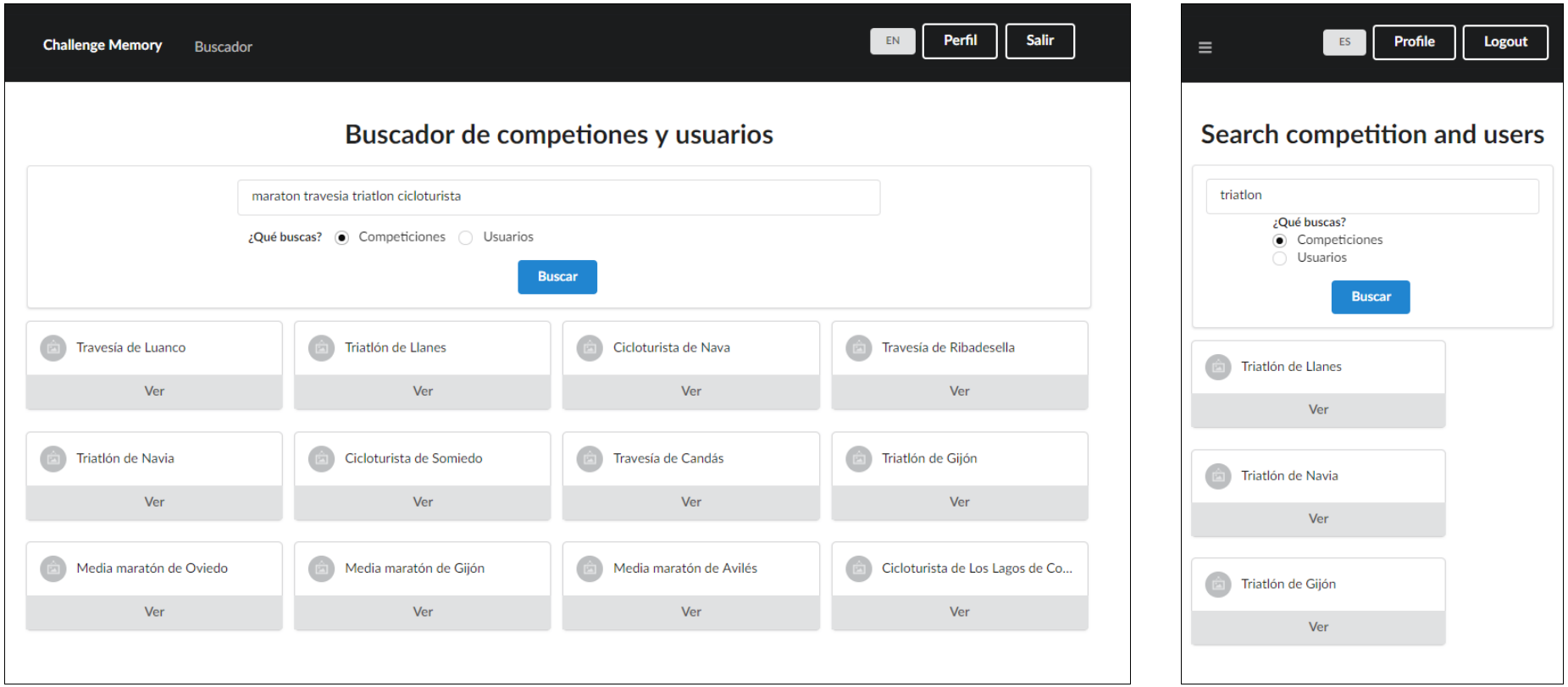


Figura 6.42. Formato PC y móvil

6.5.9 Pantalla buscador de usuarios

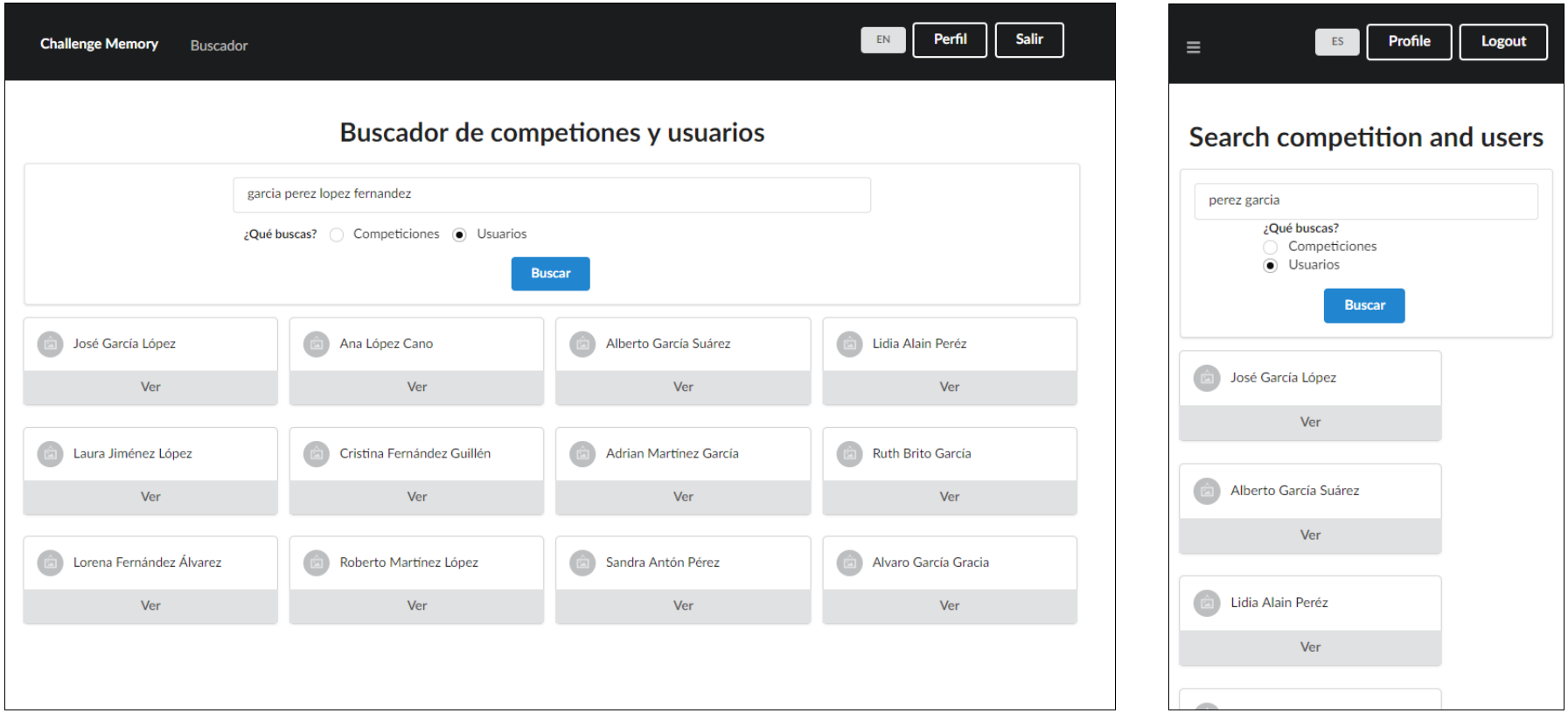


Figura 6.43. Formato PC y móvil

6.5.10 Pantalla de detalle de competición

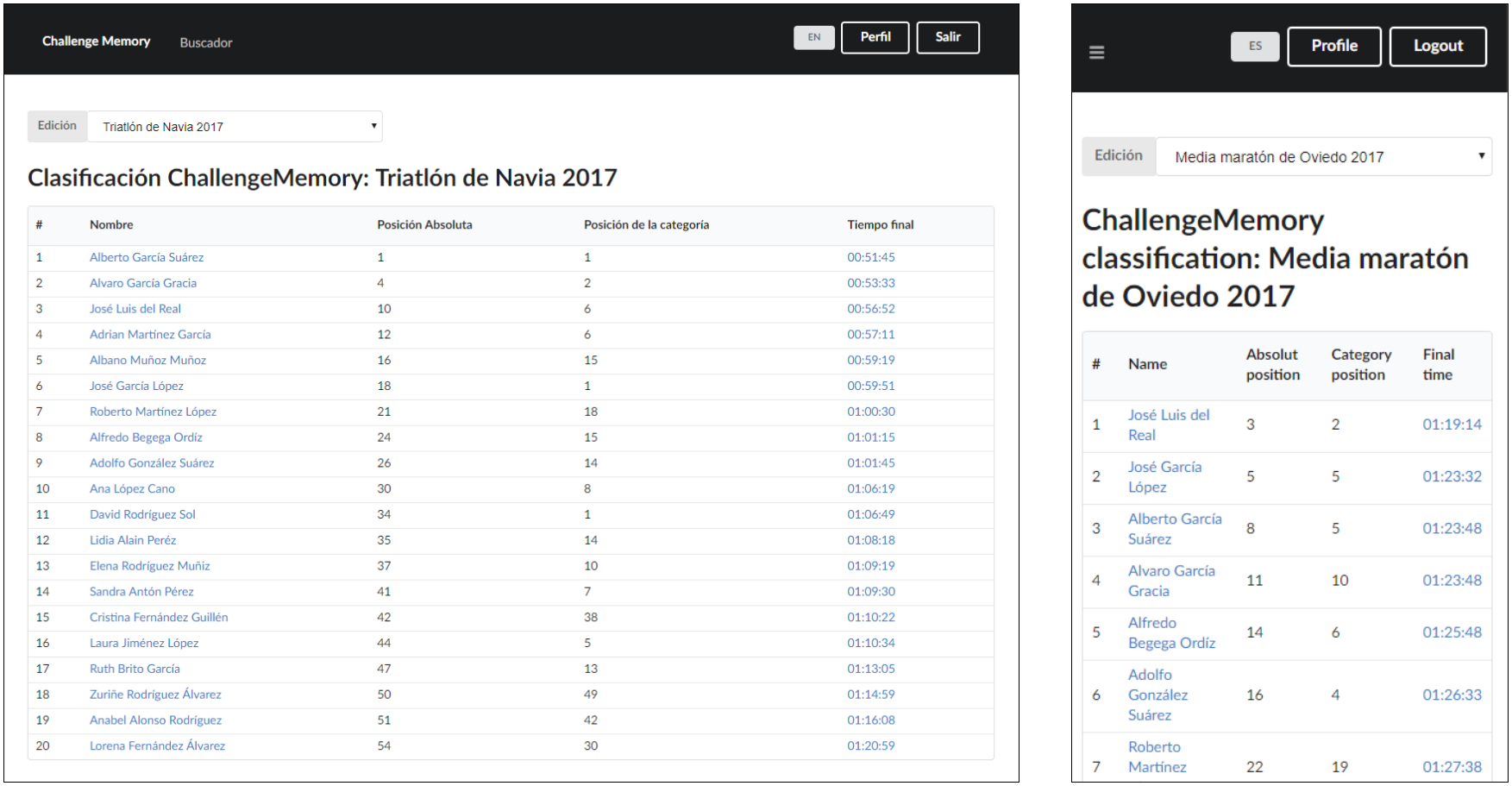


Figura 6.44. Formato PC y móvil

6.5.11 Pantalla de clasificación de edición

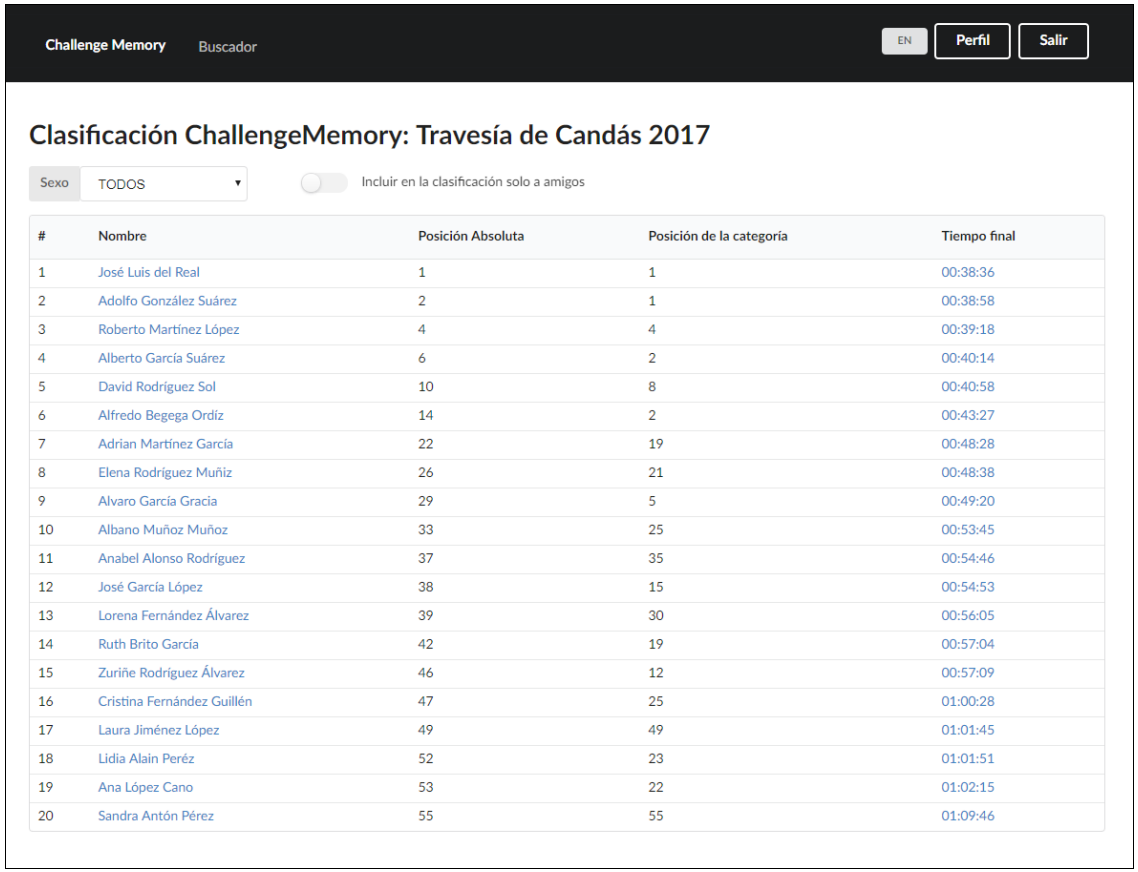


Figura 6.45. Formato PC y móvil

6.5.12 Pantalla de nuevo resultado

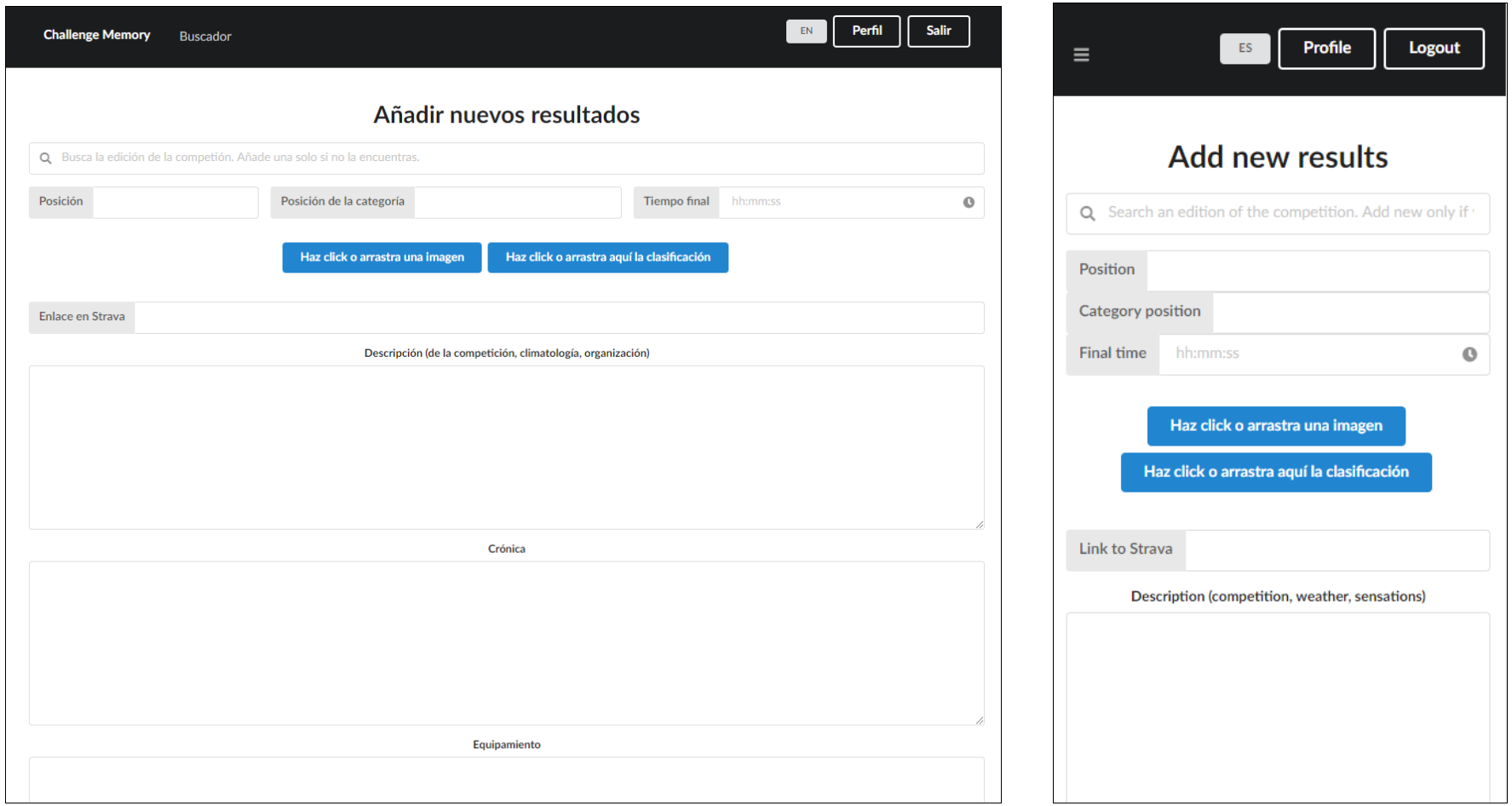
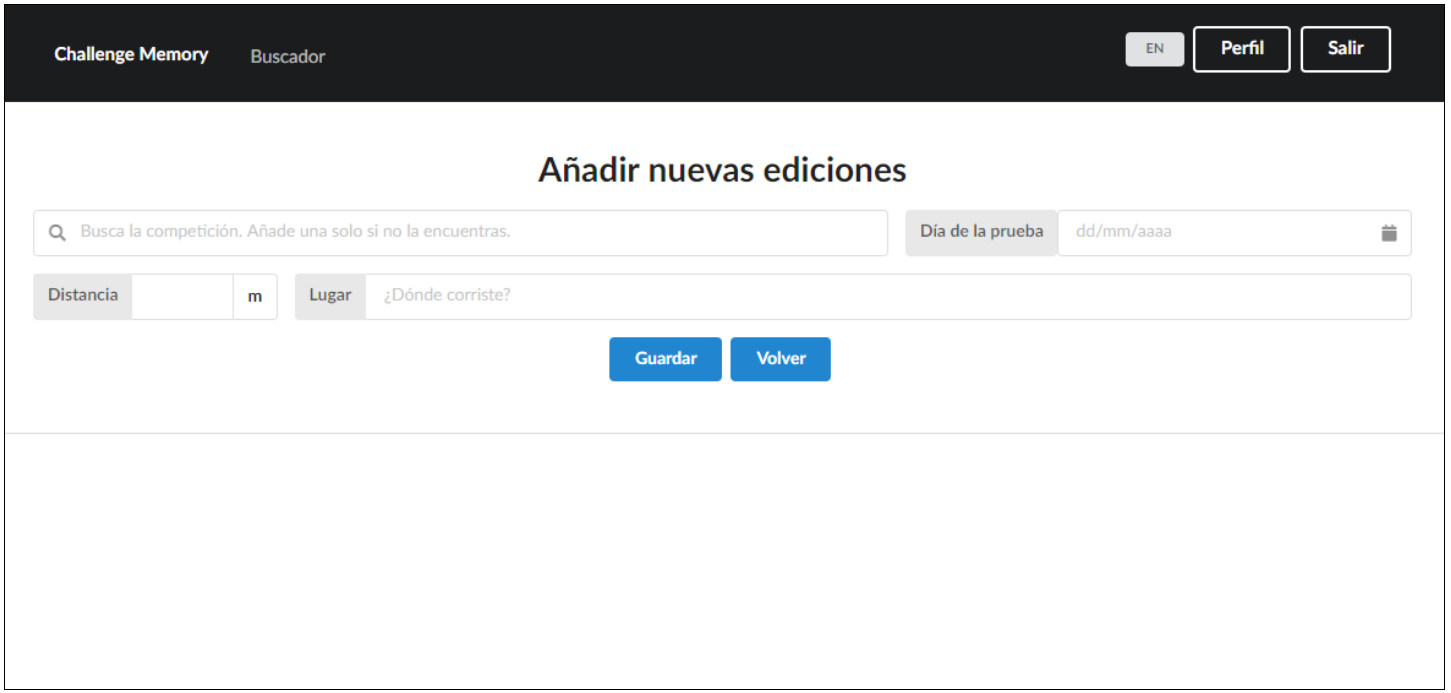


Figura 6.46. Formato PC y móvil

6.5.13 Pantalla nueva edición de competición



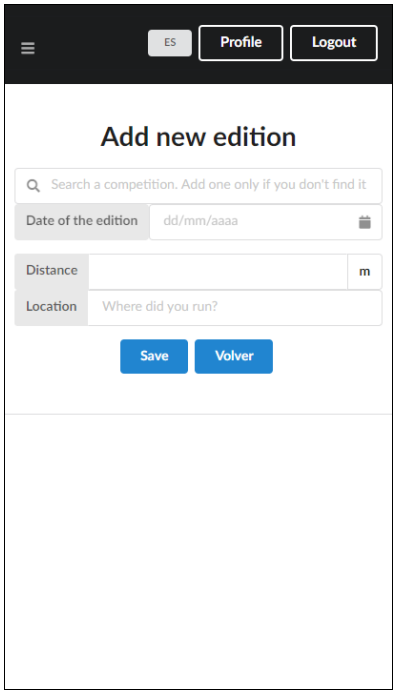


Figura 6.47. Formato PC y móvil

6.5.14 Pantalla de nueva competición

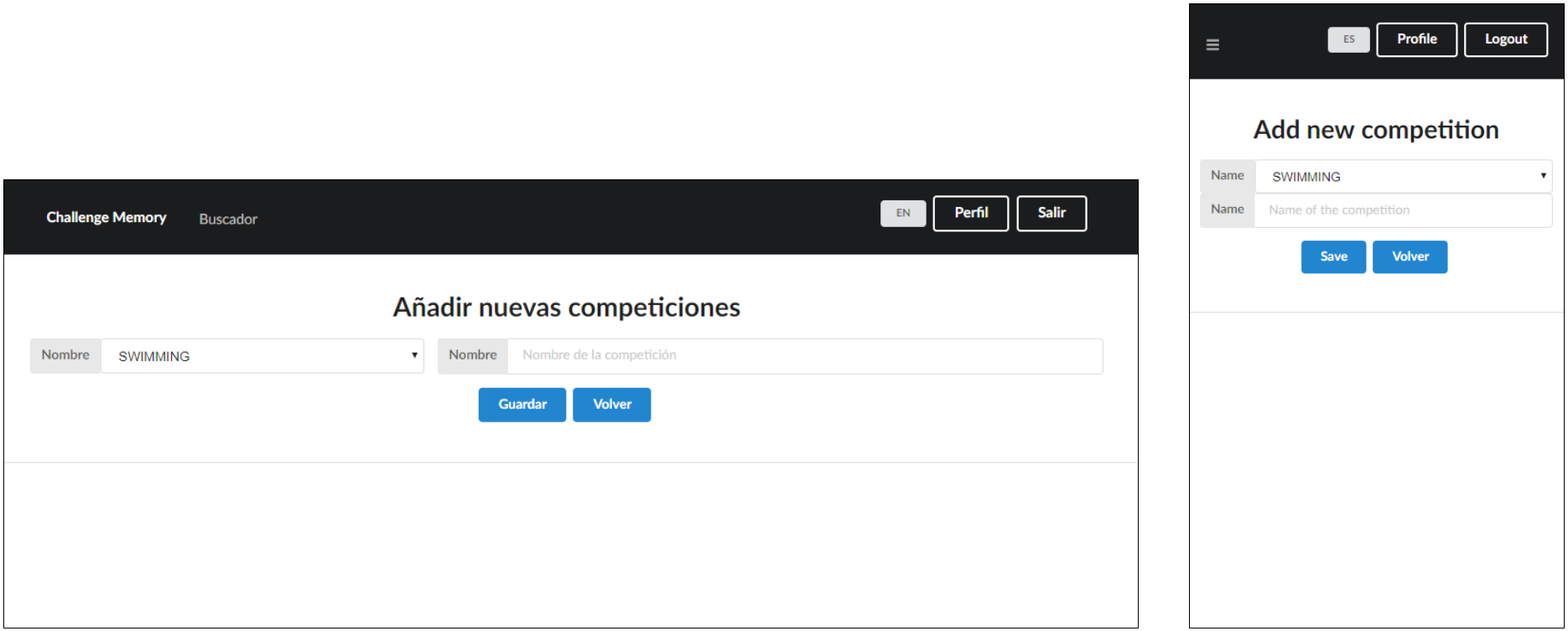


Figura 6.48. Formato PC y móvil

6.5.15 Pantalla de edición de resultado

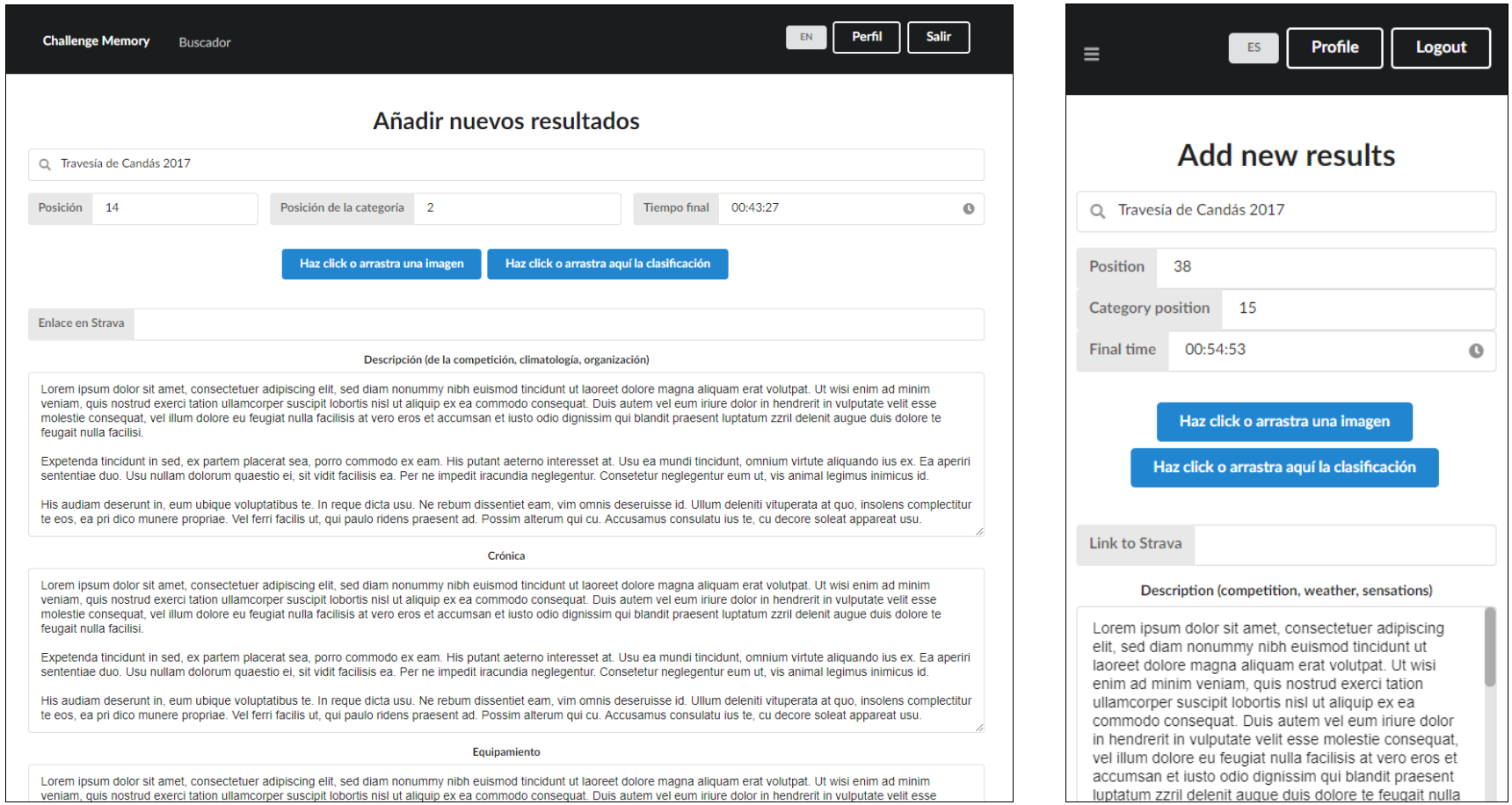


Figura 6.49. Formato PC y móvil

6.5.16 Pantalla de aviso de resultado falso

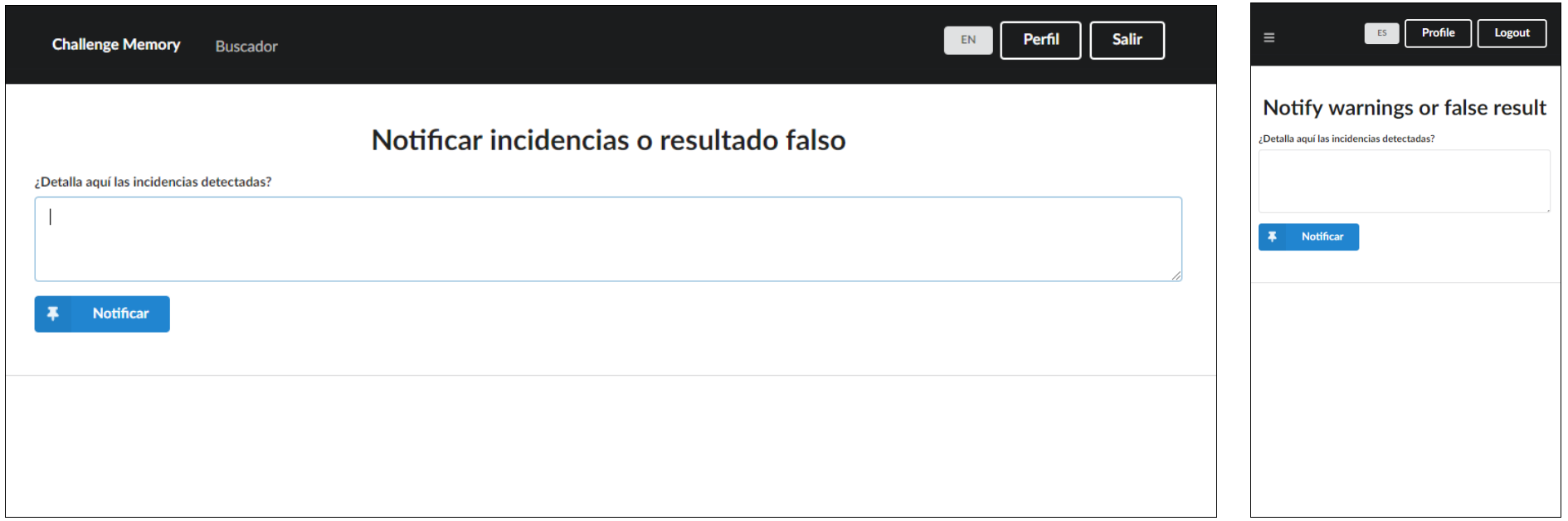


Figura 6.50. Formato PC y móvil

6.5.17 Pantalla de gestión de avisos (Administrador)

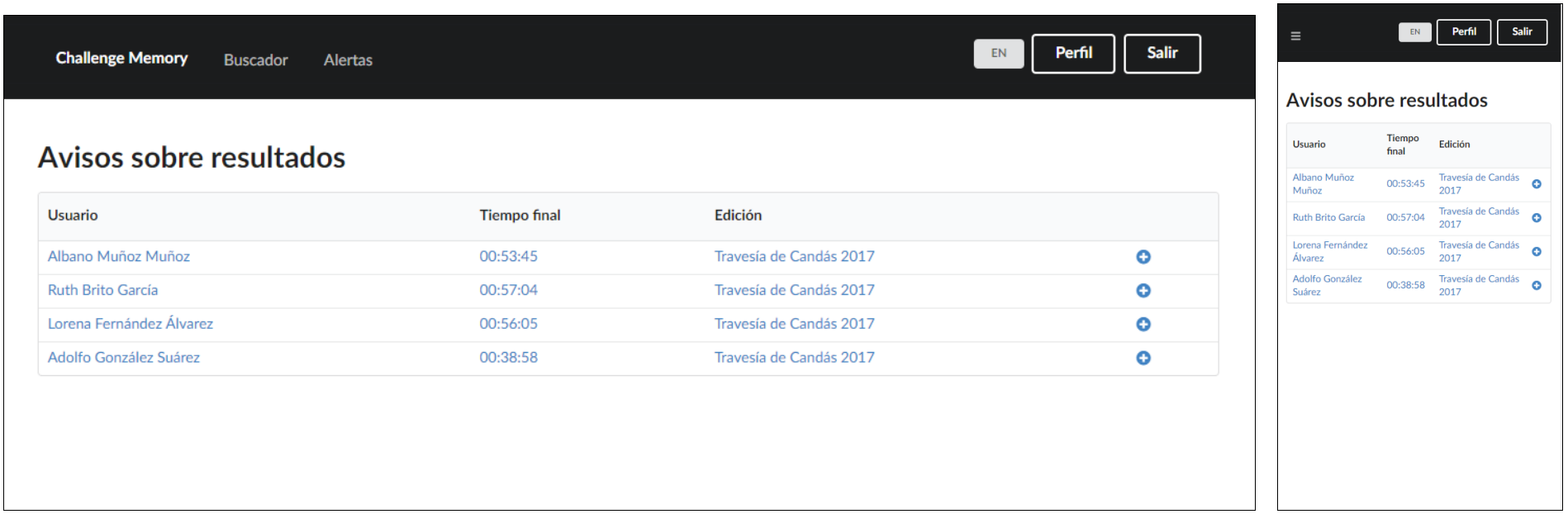


Figura 6.51. Formato PC y móvil

6.5.18 Pantalla de detalles de aviso (Administrador)

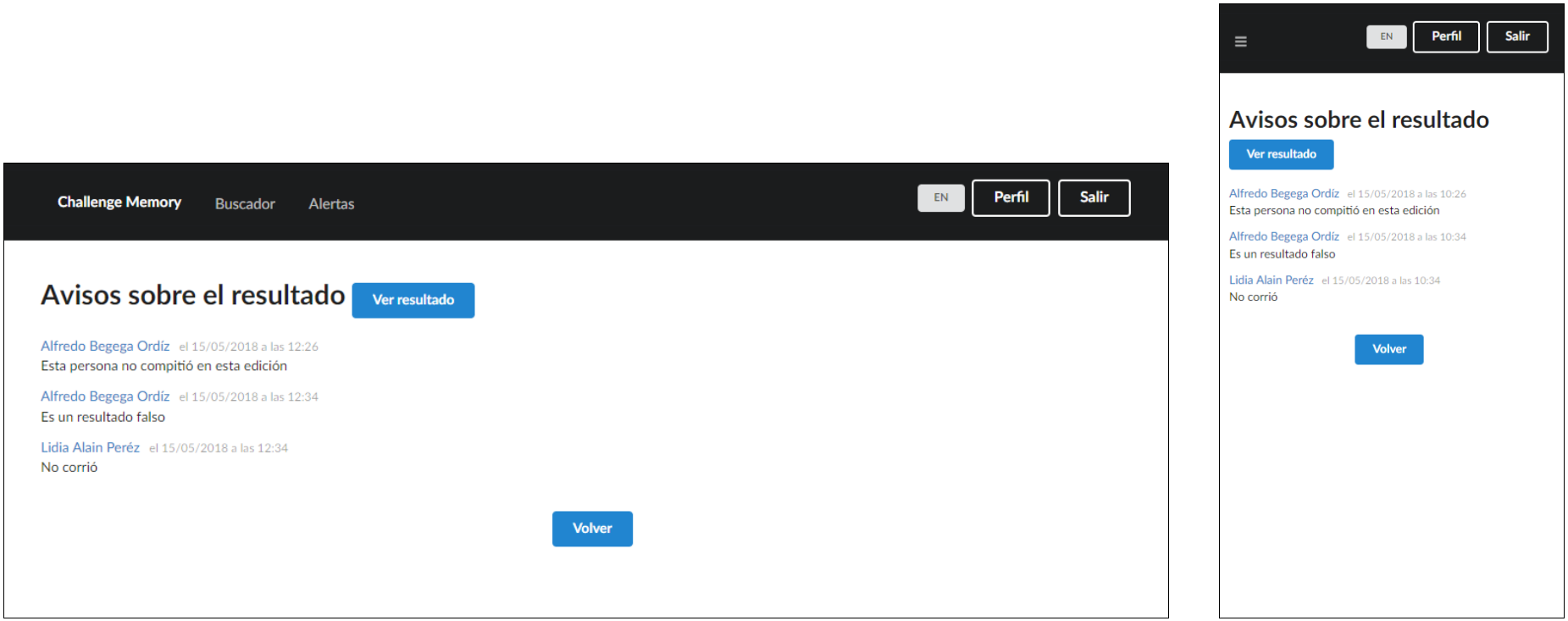


Figura 6.52. Formato PC y móvil

6.5.19 Elementos generales de la interfaz (notificaciones, validaciones…)

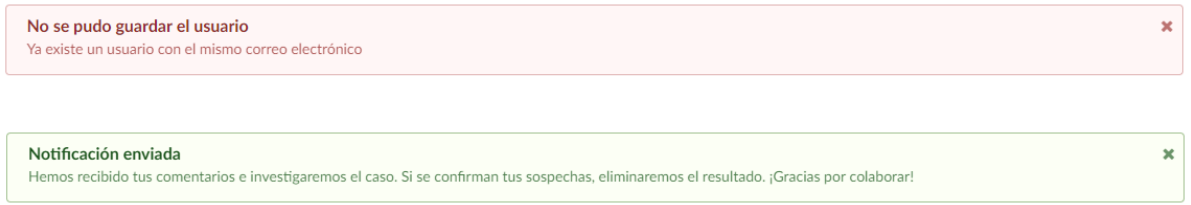


Figura 6.53. Notificaciones de error y éxito

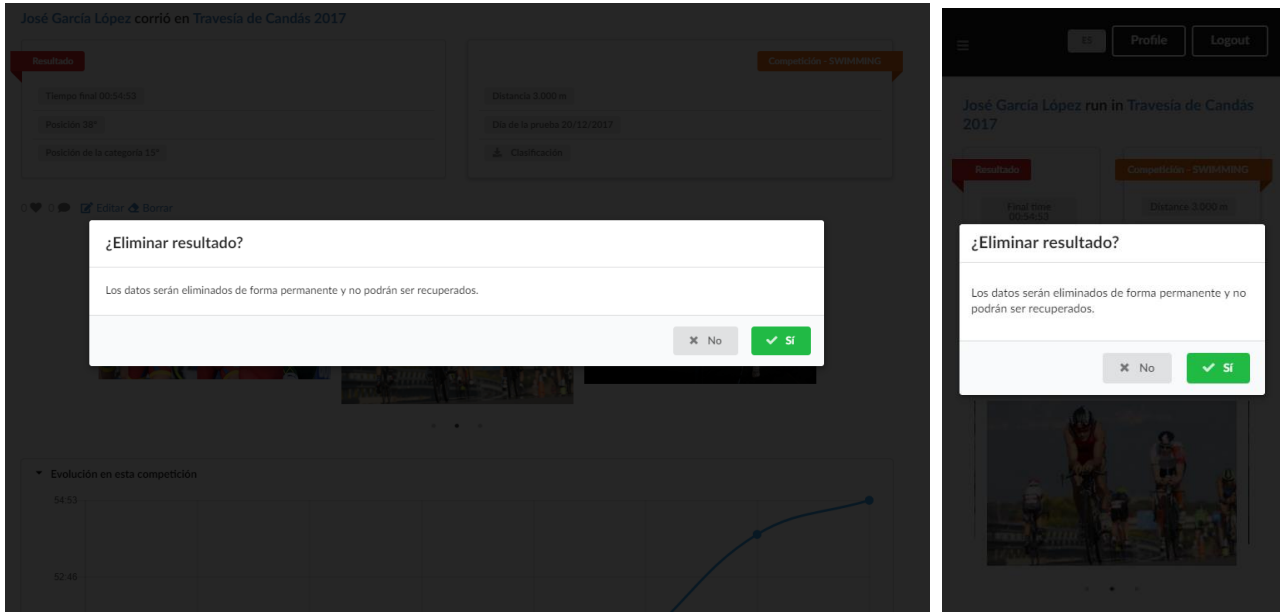


Figura 6.54. Ventana de confirmación de acción

Figura 6.55. Validaciones en formulario

6.6 Especificación Técnica del Plan de Pruebas

A continuación se detallan las diferentes pruebas a las que será sometido el sistema, durante y posteriormente al desarrollo. Las pruebas se realizarán en una única máquina, que actuará de cliente y servidor a través de red local accediendo a través de localhost.

El software empleado para la realización de las pruebas será el siguiente:

- Sistema operativo: Windows 10
- Entorno de desarrollo: Spring Tool Suite 3.9.2
- Explorador de BD: MongoDB Compass Community 1.13.1
- Navegadores: Google Chrome 66.0.3359.181, Microsoft Edge 42.17134.1.0 y Mozilla Firefox 60.0.1

6.6.1 Pruebas Unitarias

Las pruebas unitarias se realizarán con la ayuda de la herramienta *JUnit* (Java). Dichas pruebas se acometerán para comprobar el correcto funcionamiento de los repositorios de acceso a datos, realizando operaciones sobre la BD implantada en el sistema. Además, después de su correcta ejecución, establecerán una carga inicial de datos para continuar adecuadamente el proceso de desarrollo.

Las pruebas serán ejecutadas en el instante que exista el modelo de datos de la parte a probar así como su repositorio de acceso a datos. Cronológicamente se irán completando en el siguiente orden: Usuarios, Competiciones, Ediciones, Resultados y Ficheros.

6.6.1.1 Pruebas Unitarias del Repositorio de Usuarios

Como inicialización de las pruebas se realizará un borrado completo de los datos previos de usuarios.

Tras la inicialización, se utilizará un array de 20 strings con nombres reales inventados y que servirán para realizar un proceso iterativo donde se utilizará el string como nombre del usuario y se asignarán valores aleatorios a los diferentes campos de cada usuario. Tras esta tarea, se guardará la información a través de la interfaz de repositorio en la BD.

Posteriormente, se recuperarán todos los usuarios. A cada usuario se le asociarán como amigos un máximo de 20 usuarios elegidos aleatoriamente y se procederá a la actualización de los datos.

Finalmente, se contarán los registros almacenados en la BD y se confirmará que el número coincide con la cantidad planificada.

6.6.1.2 Pruebas Unitarias del Repositorio de Competiciones

Como inicialización de las pruebas se realizará un borrado completo de los datos previos de competiciones.

Tras la inicialización, se utilizará un array de 12 strings con nombre de competiciones inventados y que servirán para realizar un proceso iterativo donde se utilizará el string como nombre de la competición y se asignarán valores aleatorios a los diferentes campos de cada competición. Tras esta tarea, se guardará la información a través de la interfaz de repositorio en la BD.

Finalmente, se contarán los registros almacenados en la BD y se confirmará que el número coincide con la cantidad planificada.

6.6.1.3 Pruebas Unitarias del Repositorio de Ediciones

Como inicialización de las pruebas se realizará un borrado completo de los datos previos de ediciones de competiciones.

Tras la inicialización, se recuperarán todas las competiciones existentes en el sistema. A través de un proceso iterativo, se generarán 8 ediciones de cada competición, entre los años 2010 y 2017, y se asignarán valores aleatorios a los diferentes campos de cada edición. Tras esta tarea, se guardará la información a través de la interfaz de repositorio en la BD.

Finalmente, se contarán los registros almacenados en la BD y se confirmará que el número coincide con la cantidad planificada.

6.6.1.4 Pruebas Unitarias del Repositorio de Resultados

Como inicialización de las pruebas se realizará un borrado completo de los datos previos de resultados.

Tras la inicialización, se recuperarán todas las ediciones de competiciones y todos los usuarios presentes en el sistema. Mediante la iteración de ambas colecciones, se generarán resultados con datos aleatorios que serán almacenados en la BD a través del interfaz de repositorio de resultados.

Posteriormente, se recuperarán todos los resultados de cada edición ordenados por tiempo, para asignar la posición correctamente y actualizar la información del resultado.

Finalmente, se contarán los registros almacenados en la BD y se confirmará que el número coincide con la cantidad planificada.

6.6.2 Pruebas del Sistema

Las pruebas del sistema se realizarán una vez finalizado el proceso de desarrollo completo. Previamente, y de manera informal, se han realizado pruebas de integración durante el desarrollo, pero éstas últimas no se harán de forma guiada y será el propio desarrollador el que las ejecute para comprobar la correcta evolución del desarrollo.

Finalizado el desarrollo será momento de acometer las pruebas de forma estructurada y ordena para verificar el correcto funcionamiento de la aplicación y dar lugar a las incidencias oportunas si fuera necesario. Este trabajo está incluido dentro de la tarea de mantenimiento de la aplicación, programada inicialmente.

6.6.2.1 Prueba: Ver página principal

Ver página principal	
Procedimiento	1. Acceder a la pantalla de inicio sin estar autenticado en el sistema. 2. Se muestra la página de inicio pública. 3. Autenticarse en el sistema 4. Volver a cargar la página de inicio. 5. Se muestra la página de inicio del usuario.

6.6.2.2 Prueba: Registrar

Registrar	
Procedimiento	1. Acceder a la pantalla de registro. 2. Introduce datos personales ficticios. 3. Se muestra la página de registro con una notificación de que se le ha enviado un email para la confirmación del registro
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen todos los datos obligatorios 2. Se muestra notificación de que existen campos obligatorios sin completar • **Escenario Alternativo 2**: 1. Se introduce un email existente en el sistema 2. Se muestra notificación de que el usuario ya existe

6.6.2.3 Prueba: Confirmar registro

Confirmar registro	
Procedimiento	1. Se accede a través de la URL única suministrada a través de email en la prueba anterior 2. El sistema utiliza la información para activar el usuario 3. Se muestra la página notificando de la finalización del proceso correctamente
Escenarios secundarios	• **Escenario Alternativo 1**: 1. Se borra manualmente el usuario en el sistema (BD) 2. Al acceder mediante el enlace suministrado por email, se notifica que el usuario no existe.

6.6.2.4 Prueba: Autenticar

Autenticar	
Procedimiento	1. Se accede a la página de autenticación 2. Se introduce usuario y contraseña 3. Se muestra la página de inicial del usuario
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen los campos del formulario 2. Se muestra notificación de que existen campos obligatorios sin completar • **Escenario Alternativo 2**: 1. Se introducen datos de un usuario inexistente 2. Se muestra notificación de que el usuario no existe

6.6.2.5 Prueba: Editar perfil y privacidad

Editar perfil y privacidad	
Procedimiento	1. Se accede a la página de edición de perfil 2. Se modifican los datos personales y privacidad 3. Se muestran los nuevos datos y una notificación de que el proceso concluyó correctamente
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen todos los datos obligatorios 2. Se muestra notificación de que existen campos obligatorios sin completar

6.6.2.6 Prueba: Crear competición

Crear competición	
Procedimiento	1. Se accede al formulario de nueva competición a través de la funcionalidad de nuevo resultado 2. Se introducen los datos necesarios inventados 3. Se muestra la página de formulario de edición de competición con la nueva competición seleccionada.
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen todos los datos obligatorios 2. Se muestra notificación de que existen campos obligatorios sin completar

6.6.2.7 Prueba: Crear edición de competición

Crear edición de competición	
Procedimiento	1. Se accede al formulario de edición de competición 2. Se introducen los datos necesarios inventados 3. Se muestra la página de formulario de resultado con la nueva edición seleccionada
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen todos los datos obligatorios 2. Se muestra notificación de que existen campos obligatorios sin completar

6.6.2.8 Prueba: Crear resultado

Crear resultado	
Procedimiento	1. Se accede al formulario de resultado 2. Se introducen los datos necesarios inventados 3. Se muestra la página de detalle de resultado
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen todos los datos obligatorios 2. Se muestra notificación de que existen campos obligatorios sin completar

	• **Escenario Alternativo 2:** 1. Se introducen datos para una edición donde el usuario ya tiene registrado un resultado 2. Se muestra notificación de que el usuario ya tiene un resultado para esta edición.

6.6.2.9 Prueba: Editar resultado

Editar resultado	
Procedimiento	1. Se accede al formulario de resultado 2. Se introducen los datos necesarios inventados 3. Se muestra la página de detalle de resultado con los nuevos datos y notificando que el proceso concluyó correctamente
Escenarios secundarios	• **Escenario Alternativo 1**: 1. No se introducen todos los datos obligatorios 2. Se muestra notificación de que existen campos obligatorios sin completar

6.6.2.10 Prueba: Borrar resultado

Borrar resultado	
Procedimiento	1. Se invoca la acción de borrar resultado 2. Se indica "Sí" en la página para confirmar la acción 3. Se muestra la página de inicio del usuario mostrando una notificación de borrado completado
Escenarios secundarios	• **Escenario Alternativo 1**: 1. Se elimina manualmente el resultado en BD 2. Se muestra la página de inicio del usuario notificando de que el resultado ya no existe

6.6.2.11 Prueba: Ver resultado

Ver resultado	
Procedimiento	1. Se accede a la página de resultado 2. Se muestran los datos en la página de resultado

6.6.2.12 Prueba: Ver usuario

Ver usuario	
Procedimiento	1. Se accede a la página de usuario 2. Se muestran la página de usuario con todos los resultados de éste.

6.6.2.13 Prueba: Añadir usuario amigo

Añadir usuario amigo	
Procedimiento	1. Se invoca la acción de añadir amigo desde la página de ver usuario 2. Se muestra la página de usuario notificando que la operación se completó correctamente
Escenarios secundarios	• **Escenario Alternativo 1**: 1. Antes de invocar la acción de añadir amigo, se modifica la privacidad del usuario objetivo. 2. Se muestra notificación de que el usuario cambió su privacidad y no se puede consultar la información

6.6.2.14 Prueba: Eliminar usuario amigo

Eliminar usuario amigo	
Procedimiento	1. Se invoca la acción de eliminar amigo desde la página de edición de perfil 2. Se indica “Sí” en la página para confirmar la acción 3. Se muestra la página de edición de perfil eliminando al usuario de la lista de amigos y notificando de que la acción se completó correctamente

6.6.2.15 Prueba: Avisar resultado falso

Avisar resultado falso	
Procedimiento	1. Se accede al formulario de avisos desde la página de resultado 2. Se introduce información inventada a enviar 3. Se muestra la página de resultado notificando que la operación se completó correctamente

6.6.2.16 Prueba: Ver competición

Ver competición	
Procedimiento	1. Se accede a la página de una competición a través del buscador 2. Se muestra la página con la clasificación virtual de la edición 3. Se selecciona de la lista otra edición a visualizar 4. Se muestra la página con la clasificación virtual de la edición seleccionada

6.6.2.17 Prueba: Ver clasificación de edición de competición

Ver clasificación de edición de competición	
Procedimiento	1. Se accede a la página de una edición de competición 2. Se muestra la página con la clasificación virtual de la edición

6.6.2.18 Prueba: Ver evolución en una competición

Ver evolución en una competición	
Procedimiento	1. Se accede a la página de resultado 2. Se genera un gráfico con los datos obtenidos, con año de la edición en el eje X y tiempo total en el eje Y

6.6.2.19 Prueba: Dar "me gusta"

Dar "me gusta"	
Procedimiento	1. Se invoca la acción dar "me gusta" 2. Se modifica la interfaz para mostrar en color rojo el corazón asociado al resultado y se incrementa el contador de "me gustas" del mismo

6.6.2.20 Prueba: Quitar "me gusta"

Quitar "me gusta"	
Procedimiento	1. Se invoca la acción quitar "me gusta" 2. Se modifica la interfaz para mostrar en color blanco el corazón asociado al resultado y se disminuye el contador de "me gustas" del mismo

6.6.2.21 Prueba: Comentar resultado

Comentar resultado	
Procedimiento	1. Se accede al formulario de comentario en la página de resultado 2. El usuario introduce el texto a enviar inventado 3. Se muestra la página de resultado actualizado con el comentario

6.6.2.22 Buscar competición

Buscar competición	
Procedimiento	1. Se accede a la página de búsqueda de competiciones/usuarios 2. El usuario introduce el texto a buscar coincidente con algún registro en el sistema 3. Se muestra la página de búsqueda con los resultados obtenidos coincidentes

6.6.2.23 Buscar usuarios

Buscar usuarios	
Procedimiento	1. Se accede a la página de búsqueda de competiciones/usuarios 2. El usuario introduce el texto a buscar coincidente con algún registro en el sistema 3. Se muestra la página de búsquedas con los resultados obtenidos coincidentes

6.6.2.24 Listar resultados con aviso

Listar resultados con aviso	
Procedimiento	1. Se accede como administrador a la página de resultados con avisos 2. Se muestra el listado de resultados con aviso 3. Se accede a un resultado con aviso 4. Se muestra la página de resultados con aviso con la información de los mismos

6.6.2.25 Desactivar usuario

Desactivar usuario	
Procedimiento	1. Se accede como administrador y se invoca la acción de desactivar usuario de la página de usuario objetivo 2. Se confirma la acción 3. Se muestra la página inicial del administrador y una notificación de que la acción se completó correctamente

6.6.3 Pruebas de Usabilidad y Accesibilidad

Dado el fin del proyecto, que pretende el desarrollo de un prototipo inicial que permita un punto de partida para un desarrollo final mucho más ambicioso, este apartado quedará postergado hasta la confirmación por parte del cliente de la continuidad del mismo con mayores recursos económicos, que actúan de limitante para la realización de estas tareas.

En cualquier caso, se realizará una revisión preliminar de las normas de accesibilidad más básicas (*Normas WCAG 2.0 nivel A*), que se estima han de ser satisfechas por el sistema. También se realizará validación de código HTML5 y CSS, que garantiza una buena base para que el código pueda ser correctamente interpretado por la mayoría de navegadores actuales.

6.6.4 Pruebas de Rendimiento

Para la medición del rendimiento de la aplicación se utilizarán los datos previamente insertados a través de las pruebas unitarias del sistema, que ofrecen una cantidad de información suficiente para la realización de las mismas.

En concreto, se realizará la medición de las operaciones que operan con listados, donde se implementa paginación para poder cumplir con un rendimiento mínimo y que el sistema no colapse en un hipotético escenario con una gran cantidad de datos. También se medirá el tiempo de carga de la visualización de un resultado, ya que dicho proceso implica consulta de información de numerosas entidades.

Los tiempos de carga a medir serán los de las siguientes páginas:

- Página inicial de un usuario
- Página de detalle de un usuario
- Buscador de competiciones
- Buscador de usuarios
- Página de detalle de un resultado

Capítulo 7. Implementación del Sistema

7.1 Lenguajes de Programación

Para la implementación del proyecto se han utilizado los siguientes lenguajes de programación.

- ***Java 1.8***: es un lenguaje de programación de propósito general, concurrente y orientado a objetos. Es un lenguaje compilado y multiplataforma (no requiere compilación en el sistema objetivo). Java es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con millones de usuarios.

- ***HTML5:*** es la última versión de HTML. El término representa dos conceptos diferentes:

    - Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos.
    - Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance.

7.2 Herramientas y Programas Usados para el Desarrollo

Descripción de todas las herramientas de desarrollo (Eclipse, Visual Studio, etc.), sistemas adicionales existentes, complementos y otros productos software que necesitemos para la implementación de nuestro sistema. Debemos dejar claro que versión usamos, para qué y cómo interactuará con nuestro sistema.

7.2.1 Spring Tool Suite 3.9.2

El desarrollo completo se ha realizado utilizando la herramienta de desarrollo Spring Tool Suite. Una Suite basada en Eclipse 4.7.2 que integra plugins específicos para el desarrollo con J2EE y proyectos Spring.

7.2.2 MongoDB Compass Community 1.13.1

MongoDB Compass Community es una herramienta gratuita para desarrollar con MongoDB e incluye las siguientes características:

- Ver, agregar y eliminar bases de datos y colecciones
- Ver e interactuar con documentos con funcionalidad CRUD completa
- Crear y ejecutar consultas ad-hoc
- Ver y optimizar el rendimiento de la consulta con planes de explicación visual
- Administrar índices: ver estadísticas, crear y eliminar

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

A continuación se muestran los resultados obtenidos tras la ejecución de las pruebas unitarias elaboradas:

Pruebas Unitarias del Repositorio de Usuarios	
Prueba	**Resultado Esperado**
Borrado de usuarios	La colección de usuarios se vacía
	Resultado Obtenido
	La colección de usuarios se vacía
Prueba	**Resultado Esperado**
Inserción y actualización de 20 usuarios	La colección de usuarios contiene 20 usuarios con la información generada y un listado de amigos
	Resultado Obtenido
	La colección de usuarios contiene 20 usuarios con la información generada y un listado de amigos

Pruebas Unitarias del Repositorio de Competiciones	
Prueba	**Resultado Esperado**
Borrado de competiciones	La colección de competiciones se vacía
	Resultado Obtenido
	La colección de competiciones se vacía
Prueba	**Resultado Esperado**
Inserción de 12 competiciones	La colección de competiciones contiene 12 competiciones con la información generada
	Resultado Obtenido
	La colección de competiciones contiene 12 competiciones con la información generada

Pruebas Unitarias del Repositorio de Ediciones	
Prueba	**Resultado Esperado**
Borrado de ediciones	La colección de ediciones se vacía
	Resultado Obtenido
	La colección de ediciones se vacía
Prueba	**Resultado Esperado**
Inserción de 8 ediciones por competición. *96 ediciones*	La colección de ediciones contiene 96 ediciones con la información generada
	Resultado Obtenido
	La colección de ediciones contiene 96 ediciones con la información generada

Pruebas Unitarias del Repositorio de Resultados	
Prueba	**Resultado Esperado**
Borrado de resultados	La colección de resultados se vacía
	Resultado Obtenido
	La colección de resultados se vacía
Prueba	**Resultado Esperado**
Inserción y actualización de resultados para cada usuario y edición. *1920 resultados*	La colección de resultados contiene 1920 resultados con la información generada y asignados a usuarios
	Resultado Obtenido
	La colección de resultados contiene 1920 resultados con la información generada y asignados a usuarios

8.2 Pruebas del Sistema

A continuación se muestran las pruebas funcionales ya diseñadas y los resultados obtenidos, comparándolo con el especificado anteriormente:

Resultados de la pruebas de sistema	
Prueba	**Resultado Esperado**
Ver página principal	Se muestra la página de inicio del usuario
	Resultado Obtenido
	Después de seguir el procedimiento especificado, efectivamente se muestra la página de inicio del usuario
Prueba	**Resultado Esperado**
Registrar	Se muestra la página de registro con una notificación de que se le ha enviado un email para la confirmación del registro
	Resultado Obtenido
	Después de seguir el procedimiento especificado, efectivamente se muestra la página de registro con una notificación de que se le ha enviado un email para la confirmación del registro
Prueba	**Resultado Esperado**
Registrar – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Registrar – Escenario alternativo 2	Se muestra notificación de que el usuario ya existe
	Resultado Obtenido
	Efectivamente se muestra notificación de que el usuario ya existe
Prueba	**Resultado Esperado**
Confirmar registro	Se muestra la página notificando de la finalización del proceso correctamente
	Resultado Obtenido
	Efectivamente se muestra la página notificando de la finalización del proceso correctamente
Prueba	**Resultado Esperado**
Confirmar registro – Escenario alternativo 1	Al acceder mediante el enlace suministrado por email, se notifica que el usuario no existe.
	Resultado Obtenido
	Efectivamente al acceder mediante el enlace suministrado por email, se notifica que el usuario no existe.
Prueba	**Resultado Esperado**
Autenticar	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra la página de inicial del usuario

Prueba	**Resultado Esperado**
Autenticar – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Autenticar – Escenario alternativo 2	Se muestra notificación de que el usuario no ya existe
	Resultado Obtenido
	Efectivamente se muestra notificación de que el usuario no existe
Prueba	**Resultado Esperado**
Editar perfil y privacidad	Se muestran los nuevos datos y una notificación de que el proceso concluyó correctamente
	Resultado Obtenido
	Efectivamente se muestran los nuevos datos y una notificación de que el proceso concluyó correctamente
Prueba	**Resultado Esperado**
Editar perfil y privacidad – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Crear competición	Se muestra la página de formulario de edición de competición con la nueva competición seleccionada
	Resultado Obtenido
	Efectivamente se muestra la página de formulario de edición de competición con la nueva competición seleccionada.
Prueba	**Resultado Esperado**
Crear competición – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Crear edición de competición	Se muestra la página de formulario de resultado con la nueva edición seleccionada
	Resultado Obtenido
	Efectivamente se muestra la página de formulario de resultado con la nueva edición seleccionada
Prueba	**Resultado Esperado**
Crear edición de competición – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Crear resultado	Se muestra la página de detalle de resultado
	Resultado Obtenido
	Efectivamente se muestra la página de detalle de resultado

Prueba	**Resultado Esperado**
Crear resultado – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Crear resultado – Escenario alternativo 2	Se muestra notificación de que el usuario ya tiene un resultado para esta edición
	Resultado Obtenido
	Efectivamente se muestra notificación de que el usuario ya tiene un resultado para esta edición
Prueba	**Resultado Esperado**
Editar resultado	Se muestra la página de detalle de resultado con los nuevos datos y notificando que el proceso concluyó correctamente
	Resultado Obtenido
	Efectivamente se muestra la página de detalle de resultado con los nuevos datos y notificando que el proceso concluyó correctamente
Prueba	**Resultado Esperado**
Editar resultado – Escenario alternativo 1	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra notificación de que existen campos obligatorios sin completar
Prueba	**Resultado Esperado**
Borrar resultado	Se muestra la página de inicio del usuario mostrando una notificación de borrado completado. El resultado fue eliminado
	Resultado Obtenido
	Efectivamente se muestra la página de inicio del usuario mostrando una notificación de borrado completado. El resultado fue eliminado
Prueba	**Resultado Esperado**
Borrar resultado – Escenario alternativo 1	Se muestra la página de inicio del usuario notificando de que el resultado ya no existe
	Resultado Obtenido
	Efectivamente se muestra la página de inicio del usuario notificando de que el resultado ya no existe
Prueba	**Resultado Esperado**
Ver resultado	Se muestran los datos en la página de resultado
	Resultado Obtenido
	Efectivamente se muestran los datos en la página de resultado
Prueba	**Resultado Esperado**
Ver usuario	Se muestran la página de usuario con todos los resultados de éste
	Resultado Obtenido
	Efectivamente se muestran la página de usuario con todos los resultados de éste

Prueba	**Resultado Esperado**
Añadir usuario amigo	Se muestra la página de usuario notificando que la operación se completó correctamente. El usuario tiene un amigo más.
	Resultado Obtenido
	Efectivamente se muestra la página de usuario notificando que la operación se completó correctamente. El usuario tiene un amigo más.
Prueba	**Resultado Esperado**
Añadir usuario amigo – Escenario alternativo 1	Se muestra notificación de que el usuario cambió su privacidad y no se puede consultar la información
	Resultado Obtenido
	Efectivamente se muestra notificación de que el usuario cambió su privacidad y no se puede consultar la información
Prueba	**Resultado Esperado**
Eliminar usuario amigo	Se muestra la página de edición de perfil eliminando al usuario de la lista de amigos y notificando de que la acción se completó correctamente
	Resultado Obtenido
	Efectivamente se muestra la página de edición de perfil eliminando al usuario de la lista de amigos y notificando de que la acción se completó correctamente
Prueba	**Resultado Esperado**
Avisar resultado falso	Se muestra notificación de que existen campos obligatorios sin completar
	Resultado Obtenido
	Efectivamente se muestra la página de resultado notificando que la operación se completó correctamente
Prueba	**Resultado Esperado**
Ver competición	Se muestra la página con la clasificación virtual de la edición seleccionada
	Resultado Obtenido
	Efectivamente se muestra la página con la clasificación virtual de la edición seleccionada
Prueba	**Resultado Esperado**
Ver clasificación de edición de competición	Se muestra la página con la clasificación virtual de la edición
	Resultado Obtenido
	Efectivamente se muestra la página con la clasificación virtual de la edición
Prueba	**Resultado Esperado**
Ver evolución en una competición	Se genera un gráfico con los datos obtenidos, con año de la edición en el eje X y tiempo total en el eje Y
	Resultado Obtenido
	Efectivamente se genera un gráfico con los datos obtenidos, con año de la edición en el eje X y tiempo total en el eje Y
Prueba	**Resultado Esperado**
Dar “me gusta”	Se modifica la interfaz para mostrar en color rojo el corazón asociado al resultado y se incrementa el contador de “me gustas” del mismo
	Resultado Obtenido
	Efectivamente se modifica la interfaz para mostrar en color rojo el corazón asociado al resultado y se incrementa el

	contador de “me gustas” del mismo
Prueba	**Resultado Esperado**
Quitar “me gusta”	Se modifica la interfaz para mostrar en color blanco el corazón asociado al resultado y se disminuye el contador de “me gustas” del mismo
	Resultado Obtenido
	Efectivamente se modifica la interfaz para mostrar en color blanco el corazón asociado al resultado y se disminuye el contador de “me gustas” del mismo
Prueba	**Resultado Esperado**
Comentar resultado	Se muestra la página de resultado actualizado con el comentario
	Resultado Obtenido
	Efectivamente se muestra la página de resultado actualizado con el comentario
Prueba	**Resultado Esperado**
Buscar competición	Se muestra la página de búsqueda con los resultados obtenidos coincidentes
	Resultado Obtenido
	Efectivamente se muestra la página de búsqueda con los resultado obtenidos coincidentes
Prueba	**Resultado Esperado**
Buscar usuarios	Se muestra la página de búsqueda con los resultados obtenidos coincidentes
	Resultado Obtenido
	Efectivamente se muestra la página de búsqueda con los resultados obtenidos coincidentes
Prueba	**Resultado Esperado**
Listar resultados con aviso	Se muestra la página de resultados con aviso con la información de los mismos
	Resultado Obtenido
	Efectivamente se muestra la página de resultados con aviso con la información de los mismos
Prueba	**Resultado Esperado**
Desactivar usuario	Se muestra la página inicial del administrador y una notificación de que la acción se completó correctamente. Es usuario no tiene acceso a la aplicación.
	Resultado Obtenido
	Efectivamente se muestra la página inicial del administrador y una notificación de que la acción se completó correctamente. Es usuario no tiene acceso a la aplicación.

8.3 Pruebas de Usabilidad y Accesibilidad

8.3.1 Pruebas de Accesibilidad

Como páginas más representativas para esta evaluación se elegirán las siguientes:

- Página de inicio
- Página de registro (página con formulario)
- Página inicio de usuario (página con listado)
- Página de detalle de edición (página con tablas)

8.3.1.1 Evaluación de Conformidad WCAG 1.0 *(A)*

Puntos de verificación Prioridad 1:

Prioridad 1	Sí	No	N/A
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.	**X**		
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.	**X**		
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).	**X**		
6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.	**X**		
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.	**X**		
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.	**X**		
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.	**X**		
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.	**X**		
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.			**X**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.			**X**

8.3.1.2 Evaluación de Conformidad HTML5 y CSS3

Se superan sin ningún error las validaciones de conformidad HTML5 y CSS versión 3 del código de las páginas seleccionadas con la ayuda de las herramientas que suministra el W3C a través de su página web oficial:

https://jigsaw.w3.org/css-validator/

https://jigsaw.w3.org/css-validator/

8.3.2 Conclusiones sobre Accesibilidad

Tras la evaluación básica de accesibilidad de los puntos que aplican en el caso del proyecto actual, se han superado todos excepto el punto 6.3. Este punto se dejará sin verificar, ya que el proyecto hace uso de scripts AJAX que no pueden ser reemplazados de forma sencilla. Se estima que los usuarios objetivo de la aplicación accederán siempre mediante navegadores con funcionalidades javascript activadas.

8.4 Pruebas de Rendimiento

Los resultados obtenidos después de la medición de tiempos de carga de las páginas objetivo de estas pruebas y para las que se ha utilizado la herramienta de desarrollador de Google Chrome, son los siguientes:

Pruebas de Rendimiento (tiempo de carga)	
Página de inicio	**Resultado Esperado**
	Menos de 3 segundos
	Resultado Obtenido
	0,787 segundos
Página de inicio de usuario	**Resultado Esperado**
	Menos de 3 segundos
	Resultado Obtenido
	2,01 segundos
Página de detalle de un usuario	**Resultado Esperado**
	Menos de 3 segundos
	Resultado Obtenido
	1,21 segundos
Buscador de competiciones	**Resultado Esperado**
	Menos de 3 segundos
	Resultado Obtenido
	0,688 segundos
Buscador de usuarios	**Resultado Esperado**
	Menos de 3 segundos
	Resultado Obtenido
	0,783 segundos
Página de detalle de un resultado	**Resultado Esperado**
	Menos de 3 segundos
	Resultado Obtenido
	2,53 segundos

Para la obtención de dichos valores se ha utilizado un protocolo basado en la repetición de la carga de la página objetivo 5 veces, de cuyos resultados se registra el valor más alto.

A la luz de los resultados arrojados por las pruebas y teniendo en cuenta que el volumen de datos existentes, podemos concluir que el rendimiento de la aplicación es óptimo y su escalabilidad es adecuada.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

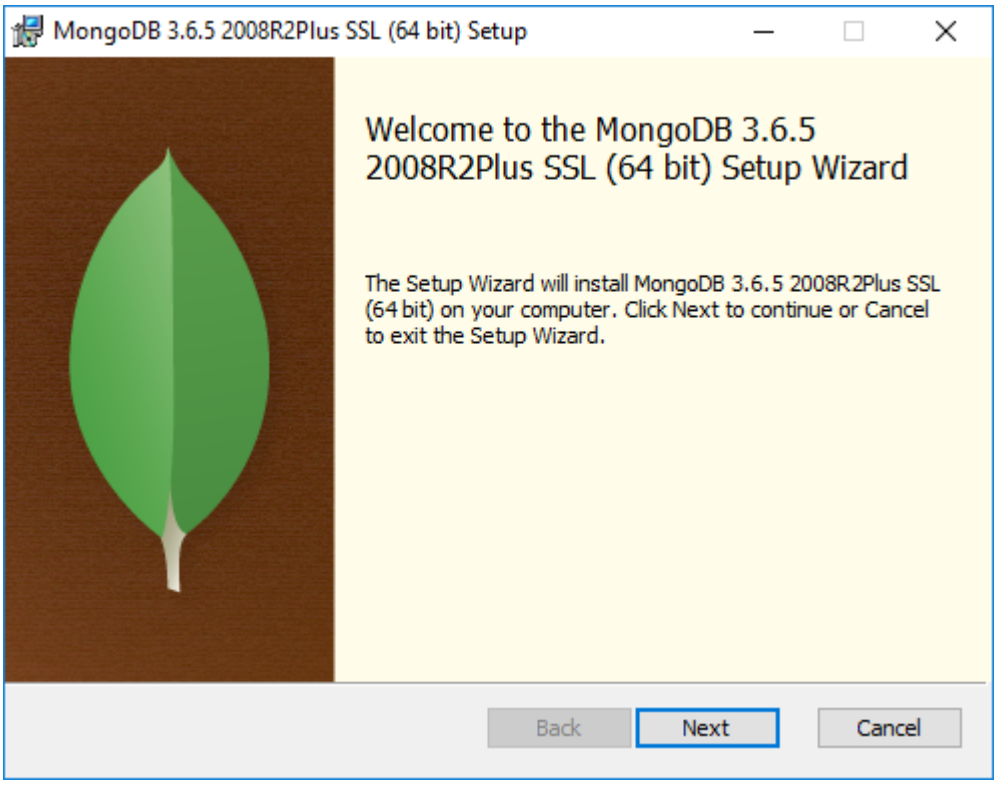
A continuación se enumeran los pasos necesarios para hacer funcionar la aplicación.

9.1.1 Instalación de MongoDB

Para instalar MongoDB lo primero que tenemos que hacer es descargar el Community Server desde la página oficial, se debe elegir la versión para sistema operativo Windows:

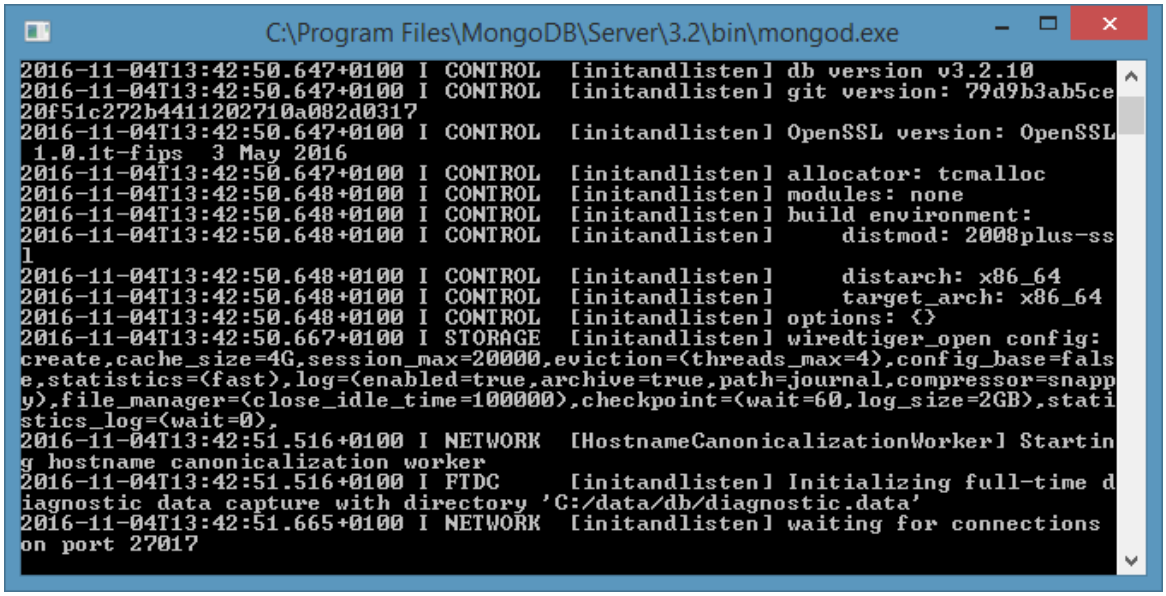
https://www.mongodb.com/download-center#community

Una vez descargado, ejecutamos el instalador y vamos siguiendo los pasos del asistente.



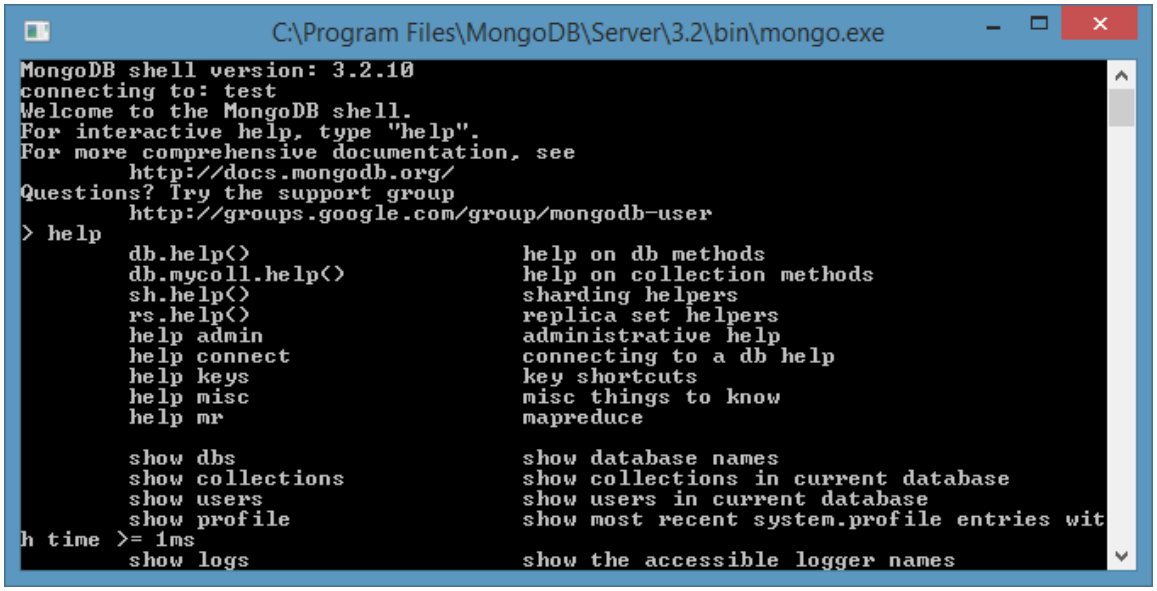
El siguiente paso es crear el directorio C:\data\db así que vamos a la unidad C: y creamos la carpeta data y dentro de data db. Aquí se van a guardar configuraciones de las bases de datos, etc.

Ahora tenemos que arrancar MongoDB, para eso vamos a C:\Program Files\MongoDB\Server\3.6\bin y ejecutamos mongod.exe



Finalmente veremos en una ventana de línea de comandos de Windows el resultado del arranque del servidor. Debemos mantener esta ventana abierta, ya que en otra caso el servidor se pararía.

En este momento ya podremos trabajar con Mongo y utilizar su consola a través de mongo.exe



Con esto ya tendríamos completada la instalación de la base de datos de la aplicación.

No es necesaria la instalación de más software para continuar con la ejecución del sistema.

9.2 Manual de Ejecución

9.2.1 Configuración de la aplicación

Después de la instalación de la base de datos del sistema, deberemos configurar la aplicación para su correcto arranque.

Entre los ficheros de la distribución de la aplicación, encontraremos un archivo comprimido denominado "*challenge-memory-dist.zip"* que contiene los ficheros de ejecución de la misma. Descomprimiremos dicho archivo en la ruta deseada, y dentro de ella encontraremos un fichero con el nombre *"challenge-memory-1.0.jar"*, debemos abrir dicho fichero con una herramienta de descompresión de archivos y editar el fichero en la ruta interna *"BOOT-INF\classes\application.yml"*.

En concreto debemos configurar la url de la base de datos:

```
data:
    mongodb:
       uri: mongodb://<ip-servidor-bd>:27017/challenge-memory
```

Y los datos de acceso a un servidor de correo electrónico disponible:

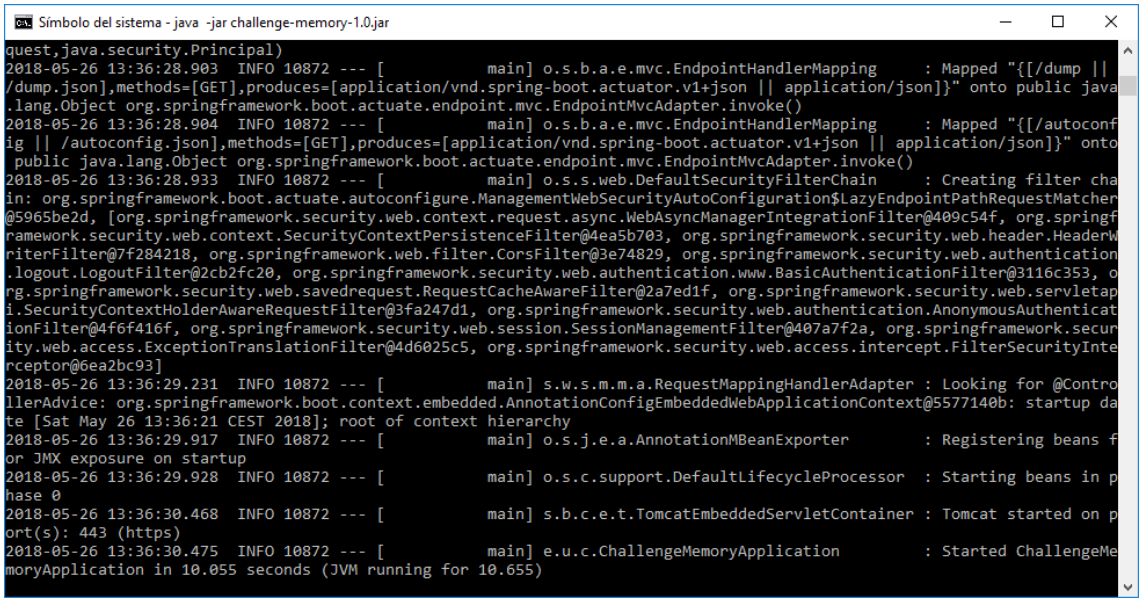
```
mail:
    host: <ip-host-servidor-correo>
    port: <puerto>
    username: <usurio>
    password: <contraseña>
```

Ahora ya estamos en disposición de lanzar la aplicación.

9.2.2 Arranque del sistema

Para lanzar el servidor que servirá la aplicación debemos ejecutar el siguiente comando:

```
java -jar challenge-memory-1.0.jar
```



Ahora ya podremos acceder a la aplicación utilizando cualquier navegador escribiendo la ruta:

https://<dirección-ip-servidor>/

9.3 Manual de Usuario

9.3.1 Inicio de la aplicación

Cuando iniciamos la aplicación web Challenge Memory, lo primero que podremos observar será la página de inicio pública.

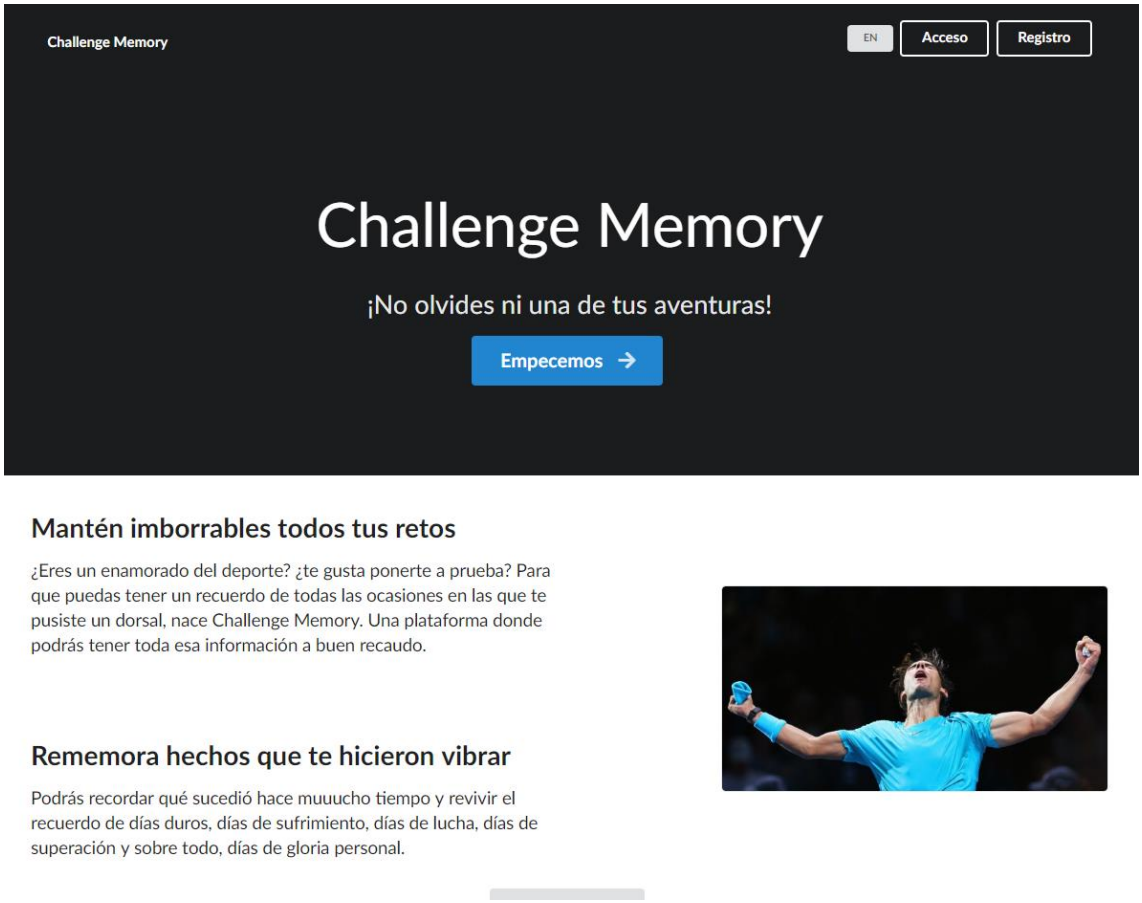


Figura 9.1. Página de inicio publica de Challenge Memory

En ella se nos da la bienvenida y se explica el fin de la aplicación. También dispondremos de las opciones para usuarios no registrados, como son acceder con un usuario o proceder al registro de uno nuevo. Adicionalmente, podremos cambiar el idioma de la interfaz de usuario, que en este momento dispone de dos idiomas, castellano e inglés.

9.3.2 Registro de usuarios

El primer paso que un usuario debe realizar para poder comenzar a utilizar la aplicación será el registro en el sistema. Desde la página de inicio de la aplicación, accederemos a la opción del menú superior "*Registro*".

Una vez cargada la página de registro encontraremos el siguiente formulario:

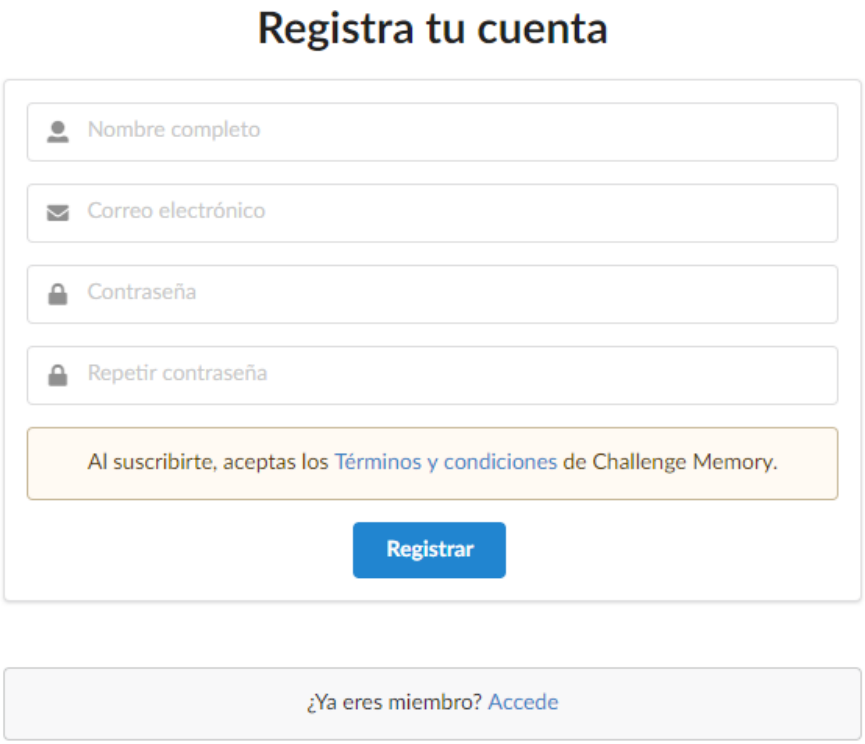


Figura 9.2. Formulario de registro

En él podremos indicar los datos mínimos para crear un usuario en la aplicación. Introduciremos los datos solicitados tras leer atentamente los términos y condiciones de Challenge Memory, que serán aceptados al proceder al registro.

Si los datos introducidos son correctos, veremos la página que nos indica que el registro completado correctamente y recibiremos un email para proceder a la confirmación de nuestra cuenta.

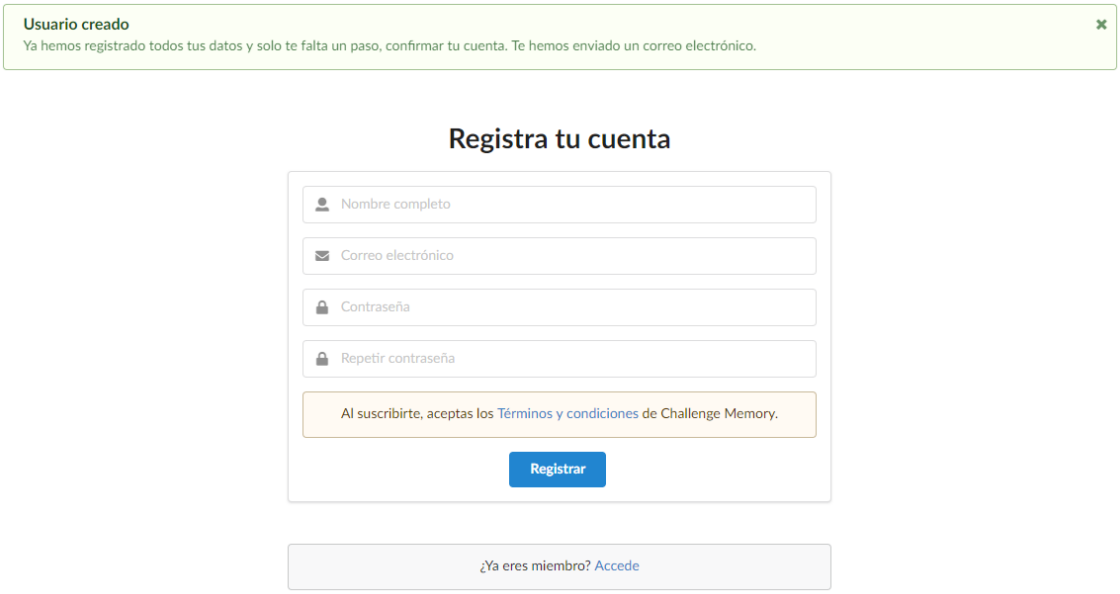


Figura 9.3. Aviso de registro correcto

9.3.3 Confirmación de registro

Tras el registro, recibiremos un email en el correo electrónico indicado durante el proceso. En el cuerpo del mismo se incluye un enlace que permitirá finalizar el registro del usuario.

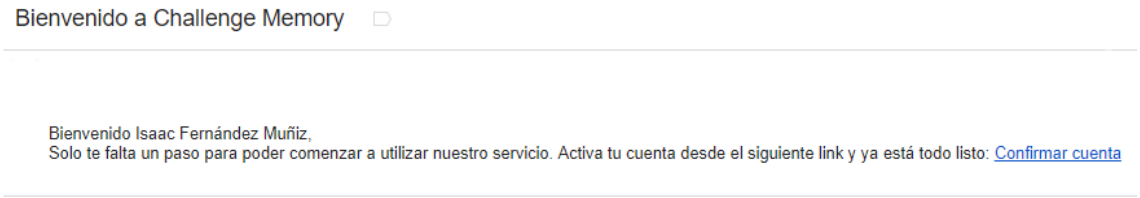


Figura 9.4. Email de confirmación de registro

Tras utilizar el enlace "*Confirmar cuenta*", se nos abrirá la aplicación en nuestro navegador por defecto informándonos que la confirmación se ha completado con éxito y podemos comenzar a utilizar la aplicación.

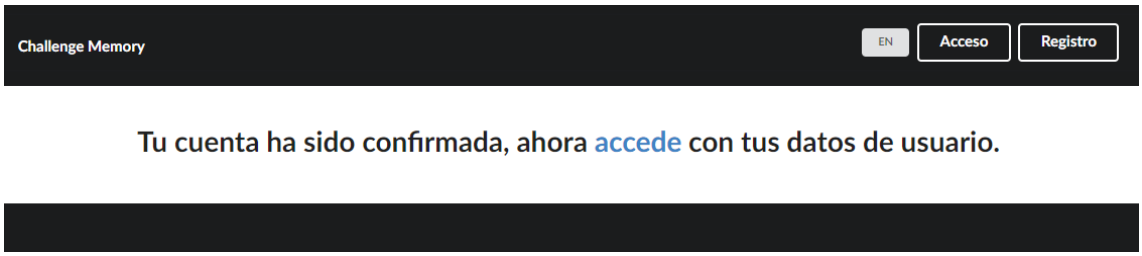


Figura 9.5. Pantalla de confirmación de registro

9.3.4 Acceso y página de inicio de usuario

Una vez completado el proceso de registro y confirmación, el usuario podrá acceder a su página de inicio a través de la opción del menú superior "*Acceso*". Dicha opción dará lugar al formulario de autenticación.

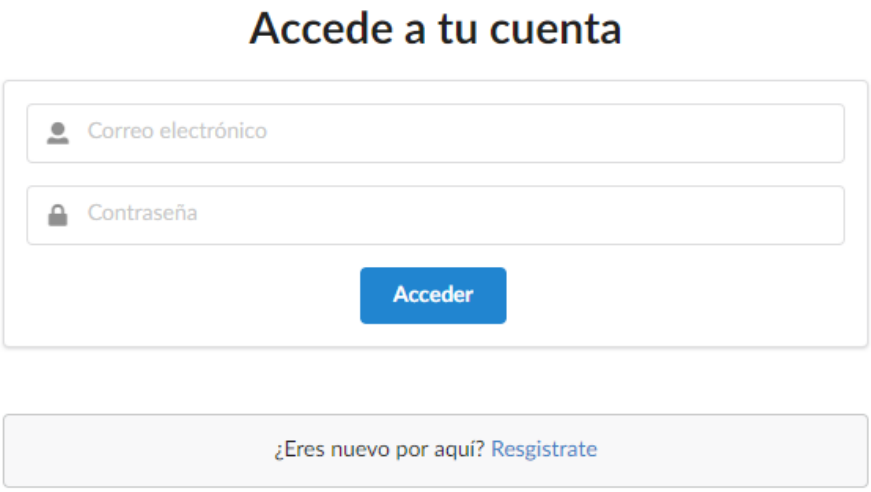


Figura 9.6. Formulario de autenticación

Tras introducir los datos necesarios el usuario será conducido a su página de inicio, que en el primer acceso estará vacía.

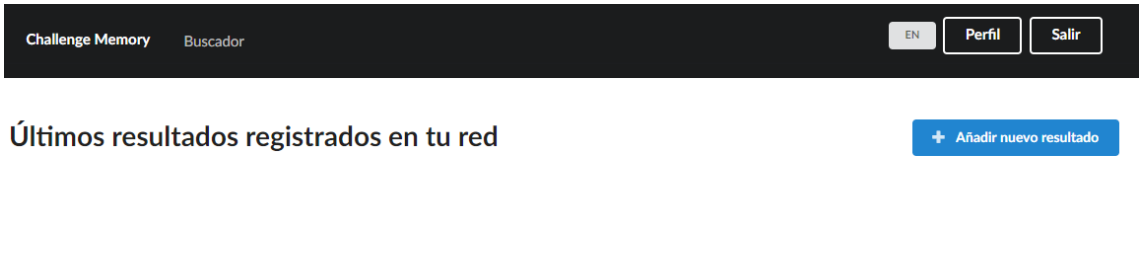


Figura 9.7. Página de inicio de un nuevo usuario

Desde ella, se podrán acceder a las siguientes opciones:

- *Buscador*: en el menú superior izquierdo se podrá acceder al buscador de la aplicación.
- *Perfil*: en el menú superior derecho se podrá ver la información del perfil del usuario.
- *Salir*: en el menú superior derecha también tendremos la opción de salir de la sesión de usuario. Opción que nos conducirá a la página de inicio pública de la aplicación nuevamente.
- *Añadir nuevo resultado*: ya en el cuerpo de la página podremos ver un botón en la parte derecha para añadir un nuevo resultado en nuestra cuenta.

9.3.5 Datos de perfil y privacidad de usuario

Si utilizamos la opción "*Perfil*", podremos editar los datos del perfil y la privacidad de un usuario. Dentro de esta página también veremos el listado de amigos del mismo y un botón que nos dará acceso a los resultados registrados por él.

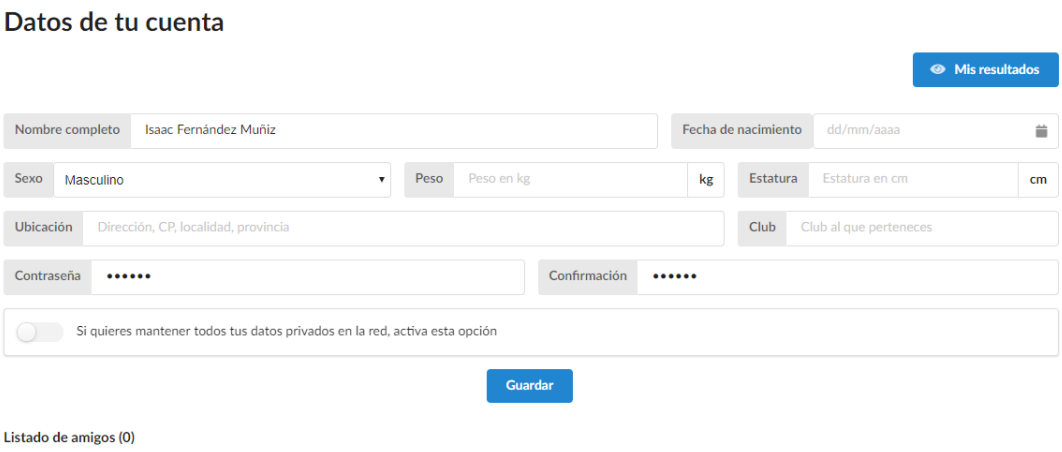


Figura 9.8. Página de inicio de un nuevo usuario

Es importante tener en cuenta el tipo de privacidad de una cuenta de usuario. Tenemos dos alternativas:

- *Cuenta pública* (por defecto): Si la opción de perfil privado está desactivada, todos los resultados que registremos en nuestra red podrán ser observados por cualquier otro usuario registrado y aparecerán en las clasificaciones virtuales de la aplicación. Además, un usuario público podrá ser añadido como amigo por cualquier otro usuario.

- *Cuenta privada*: si la opción de perfil privado está activada, este usuario no aparecerá en ninguna de las opciones de la aplicación, ni clasificaciones virtuales, ni buscador, ni podrá añadir ni ser añadido como amigo por otros usuarios. En este caso, Challenge Memory servirá al usuario como historial de resultados propios solamente.

9.3.6 Búsqueda de competiciones y usuarios

Haciendo uso de la opción "*Buscador*" del menú superior izquierdo tendremos acceso a la página de búsqueda. Dicha página nos dará la opción de realizar búsquedas de dos tipos:

- *Competiciones*: que realizará la búsqueda sobre las competiciones dadas de alta por todos los usuarios de Challenge Memory.

- *Usuarios*: que realizará la búsqueda sobre los usuarios públicos existentes en el sistema.

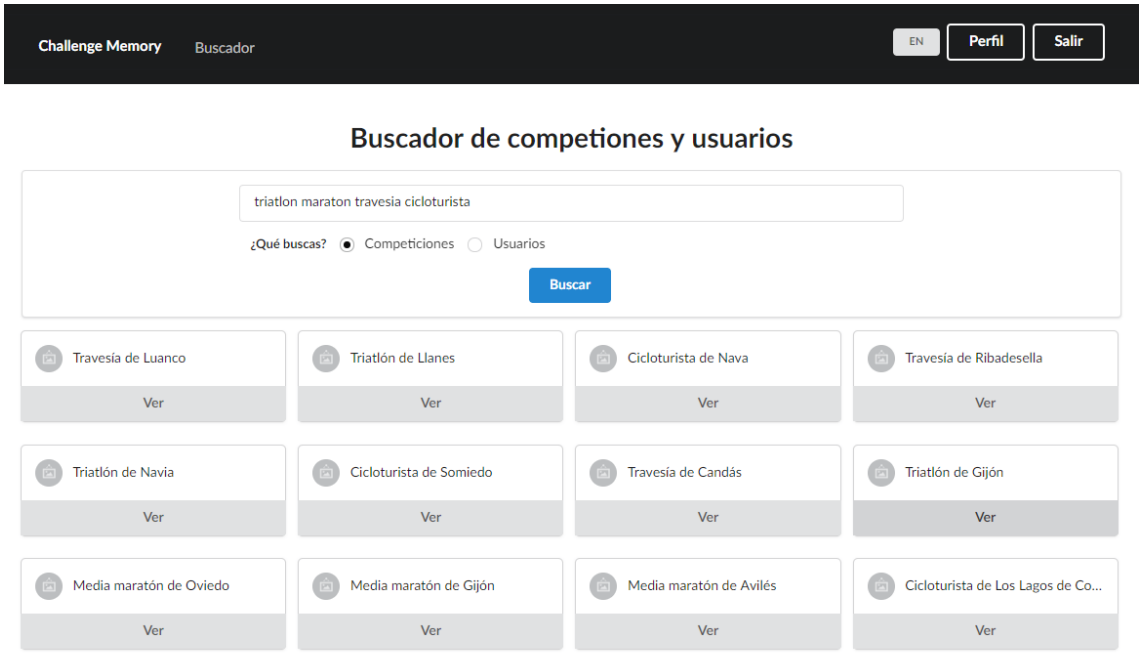


Figura 9.9. Búsqueda de competiciones

El funcionamiento es muy sencillo y el usuario solo deberá escribir los términos de búsqueda y seleccionar el tipo de búsqueda a realizar. Los términos de búsqueda serán tenidos en cuenta por separado y sin caracteres especiales (tildes, eñes…). Adicionalmente, se permitirá el uso de operadores especiales (+ y -) para permitir el refinado de las búsquedas, de tal modo que si usamos el operador +, la búsqueda deberá incluir la palabra justo a su derecha y si utilizamos el operador -, la búsqueda solo arrojará resultados que no contengan la palabra a su derecha.

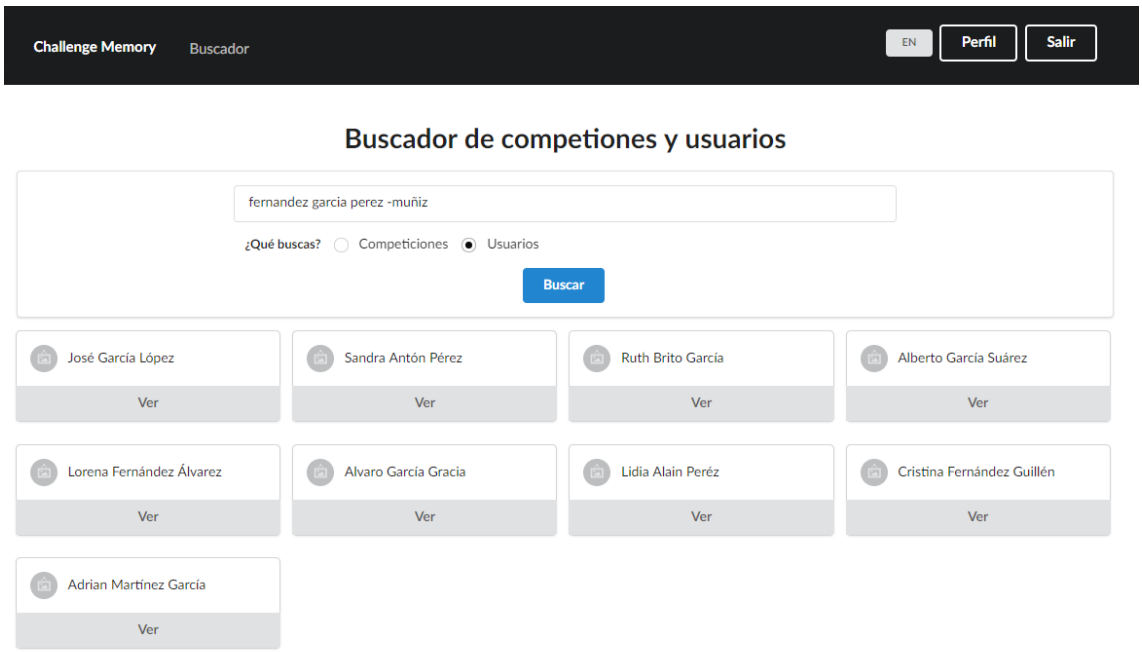


Figura 9.10. Búsqueda de usuarios

9.3.7 Ver resultados de un usuario

Si accedemos a un usuario utilizando cualquiera de las vías que la aplicación permite: búsqueda de usuarios y enlaces en los nombre de usuario en diferentes secciones, podremos visualizar todos los resultados introducidos por dicho usuario ordenados por fecha de publicación.

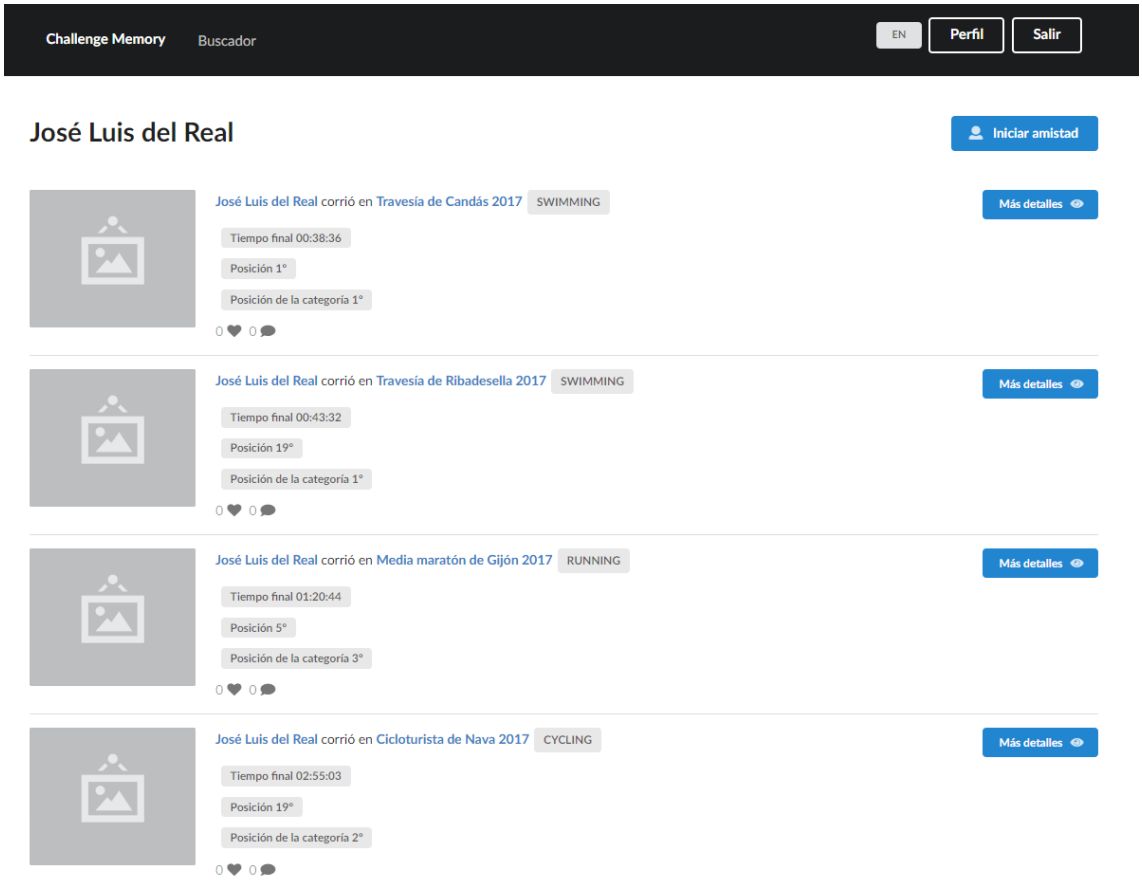


Figura 9.11. Página de resultados de un usuario

Si no existiese relación de amistad entre el usuario que acceder y el consultado, se mostrará la opción de "*Iniciar amistad*" en la parte superior izquierda del contenido de la página.

9.3.8 Clasificación virtual de una edición

Si utilizamos los enlaces con el nombre de la edición que se muestran en las diferentes páginas que con información sobre resultados (listados de resultados y detalle de resultado), tendremos acceso a la tabla de clasificación virtual generada por Challenge Memory. Dicha tabla, muestra ordenados por tiempo lo resultados dados de alta en el sistema para la edición consultada. De esta forma, se puede establecer una competición interna, al margen de la competición real, de los usuarios miembros de la plataforma.

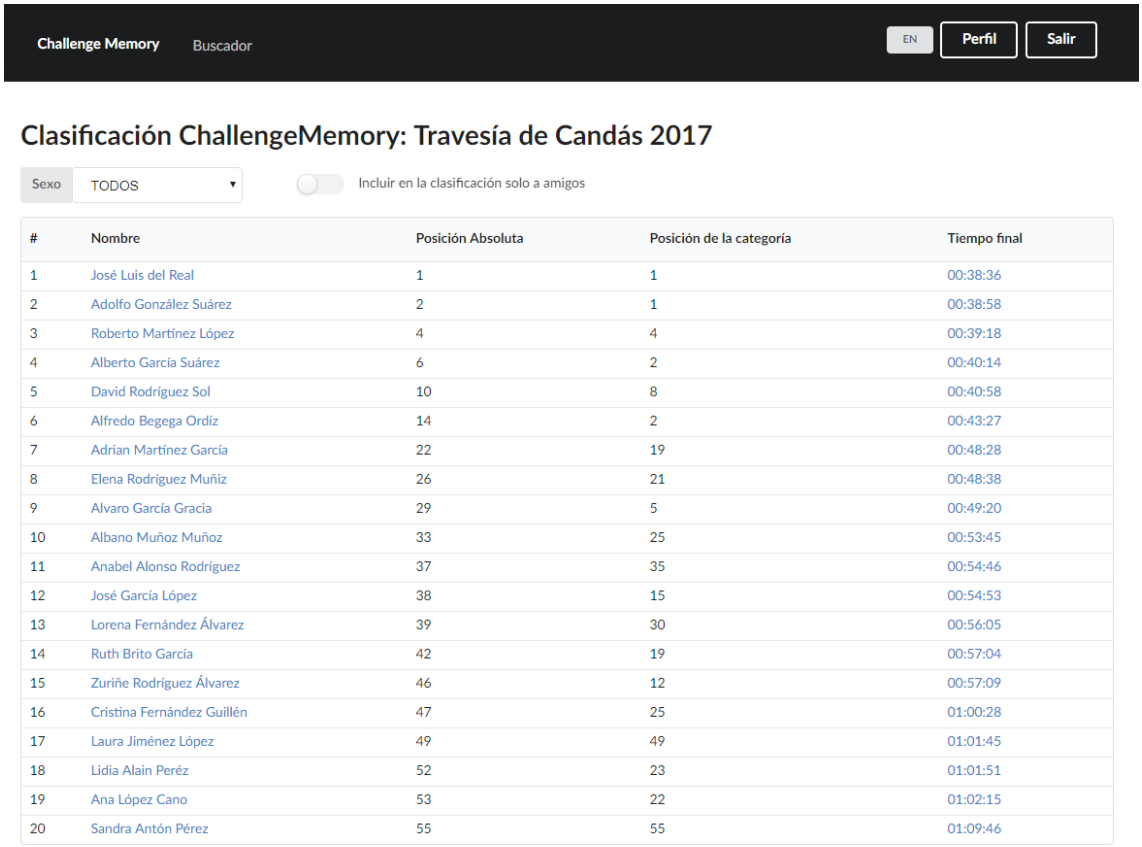


Figura 9.12. Clasificación virtual de una edición

A través de los filtros superiores, podremos generar tablas de clasificación parciales, filtrando los resultados por sexo o visualizando solo los resultados de usuarios amigos.

Utilizando los enlaces en el nombre del usuario o en el tiempo del resultado, podremos ver la página de usuario o el detalle del resultado respectivamente.

9.3.9 Ver una competición

Cuando accedemos a los resultados ofrecidos por la búsqueda de competiciones, podremos ver la clasificación virtual de cada una de las ediciones existentes de esa competición.

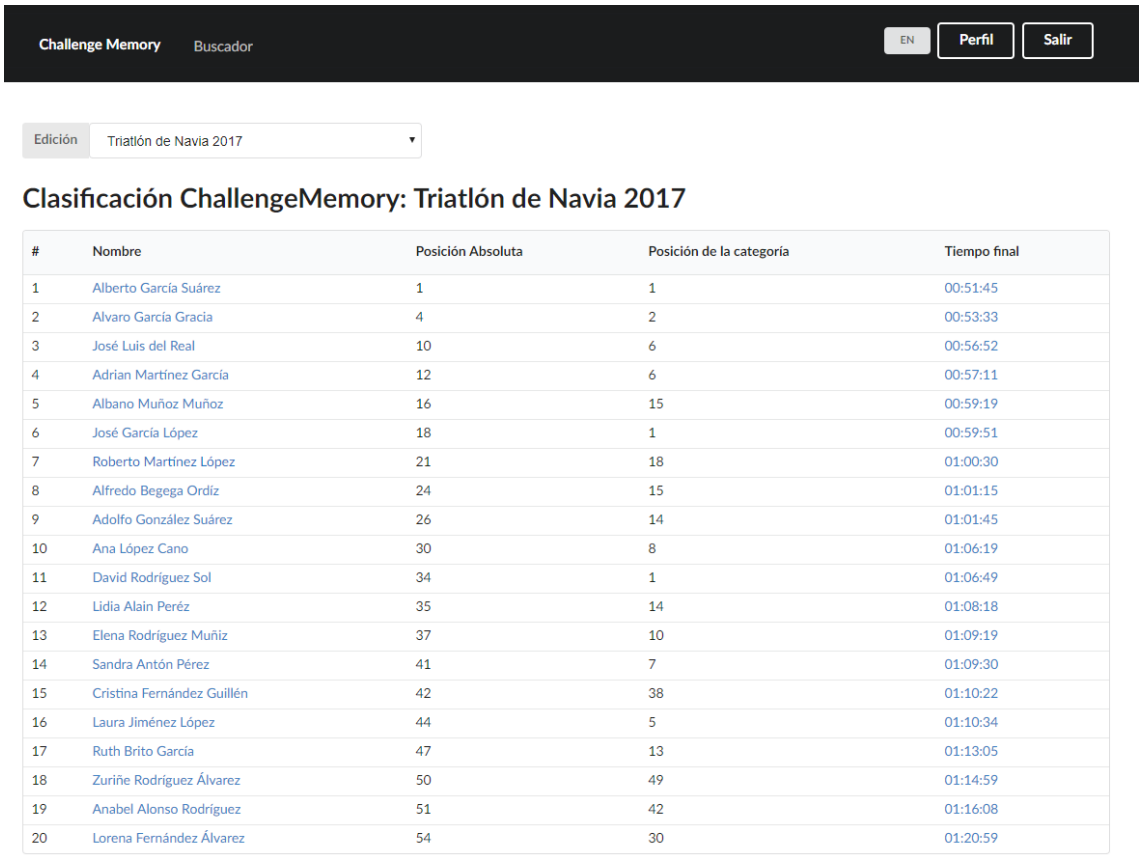


Figura 9.13. Clasificación virtual de una edición

Para cambiar la edición consultada, simplemente tendremos que utilizar el desplegable superior que incluye todas las ediciones.

Utilizando los enlaces en el nombre del usuario o en el tiempo del resultado, podremos ver la página de usuario o el detalle del resultado respectivamente.

9.3.10 Registro de un nuevo resultado

Para el registro de un nuevo resultado en la plataforma, debemos usar la opción "*Añadir nuevo resultado*" que podremos encontrar en la página de inicio de usuario, dentro del contenido en la parte superior derecha. Esto nos dará acceso a un formulario donde deberemos de introducir los datos del mismo.

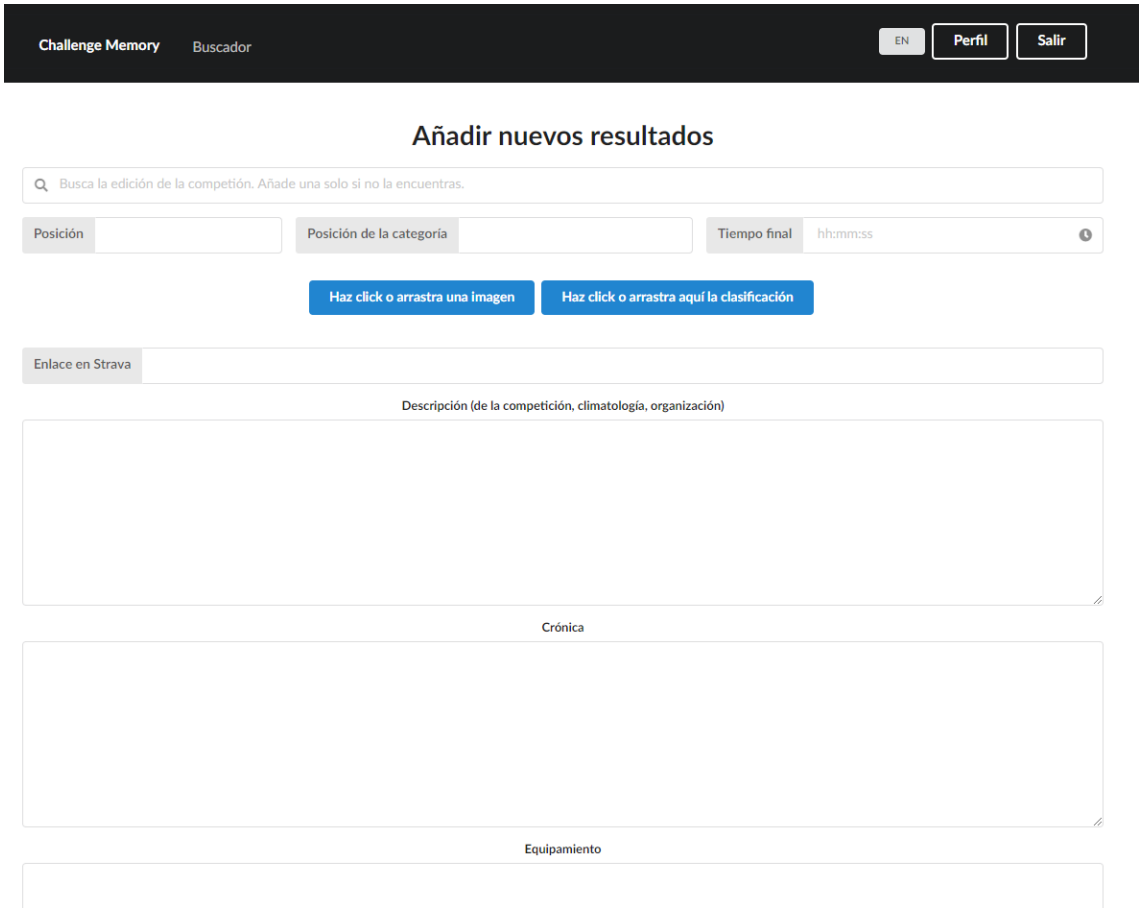


Figura 9.14. Formulario de alta de resultado

Un paso importante en este proceso es seleccionar adecuadamente la edición de competición para la cual estamos registrando el resultado. Por este motivo es de suma importancia que el usuario realice un esfuerzo para buscar adecuadamente la edición en el panel desplegable al efecto.

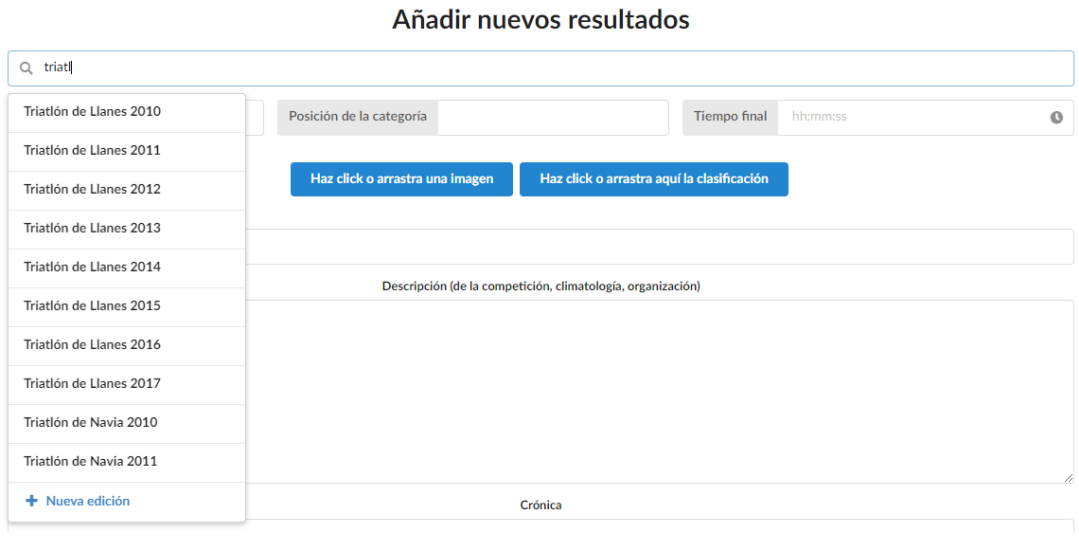


Figura 9.15. Panel desplegable de búsqueda de ediciones

En el caso de no encontrar la edición oportuna, se utilizaría la opción "*Nueva edición*" que se muestra al final del mismo listado de ediciones, dándonos acceso al formulario de alta de ediciones donde podremos registrar la edición necesaria.

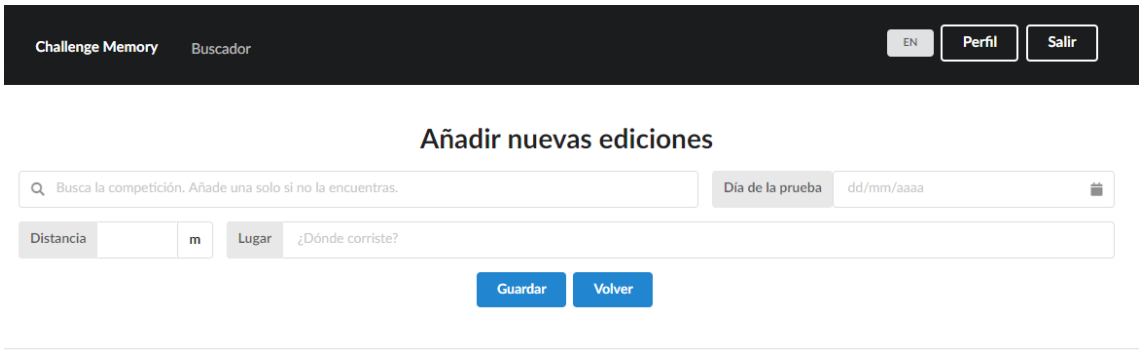


Figura 9.16. Formulario de nueva edición

De la misma forma que es necesario buscar adecuadamente la edición de un resultado, lo mismo sucede con la competición asociada a una edición, y por tanto deberemos de buscar adecuadamente la misma.

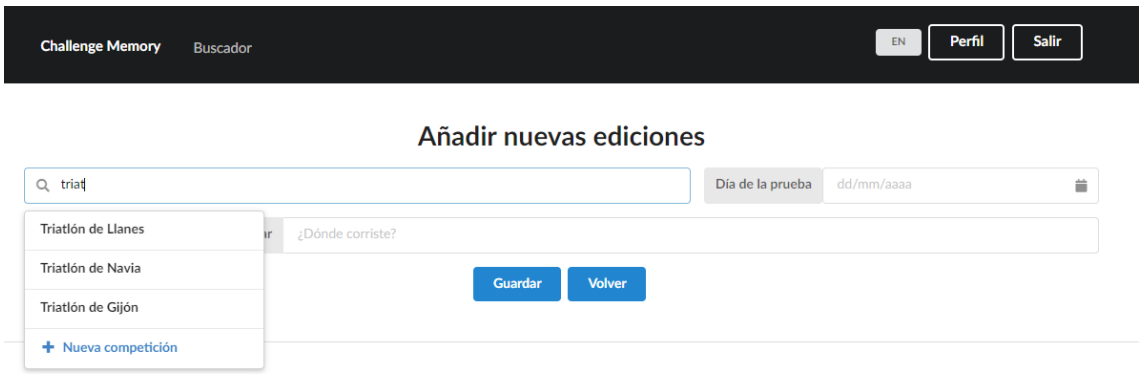


Figura 9.17. Panel desplegable de búsqueda de competiciones

Si no fuese posible encontrar la competición, podemos utilizar la opción de "*Nueva competición*", que nos dará acceso al formulario de alta de competiciones, donde registraremos la información de la competición.

Figura 9.18. Formulario de nueva competición

Cuando guardemos la información en los diferentes formularios el sistema nos conducirá automáticamente a la pantalla anterior con la competición o edición, creados y seleccionados en el formulario oportuno.

Finalmente podremos finalizar el proceso de alta del resultado utilizando el botón guardar, lo que nos conducirá nuevamente a la pantalla de inicio del usuario con el nuevo resultado en primer lugar.

9.3.11 Detalle de un resultado

Si accedemos a la información detallada de un resultado podemos ver la información del mismo estructurada en diferentes secciones.

En una primera sección podremos ver los datos básicos del resultado, así como las acciones que podemos realizar con él, como son:

- Dar o quitar "me gusta" y ver qué usuarios indicaron "me gusta" y cuántos "me gusta" tiene el resultado.
- Número de comentarios del resultado y quienes lo comentaron.
- Si el resultado no es del usuario que accede a él, la opción de notificar aviso de resultado falso
- Si el resultado es del usuario que accede a él, la opción de editar sus datos.
- Si el resultado es del usuario que accede a él, la opción de borrar el resultado.

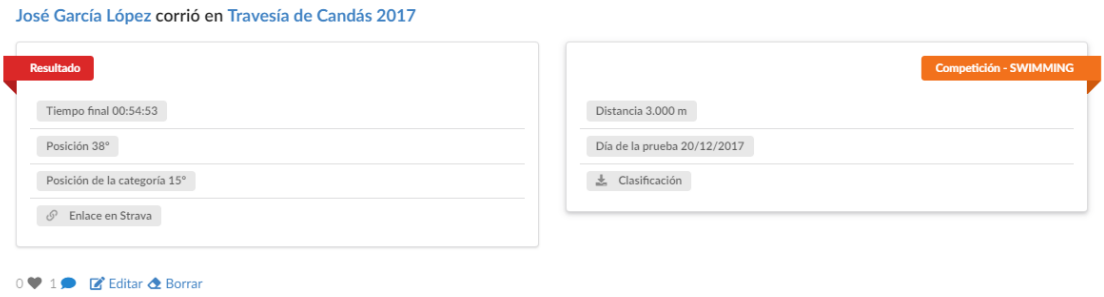


Figura 9.19. Datos básicos del resultado

En una sección posterior, podremos ver la evolución de tiempos totales del usuario en las diferentes ediciones de esta competición.

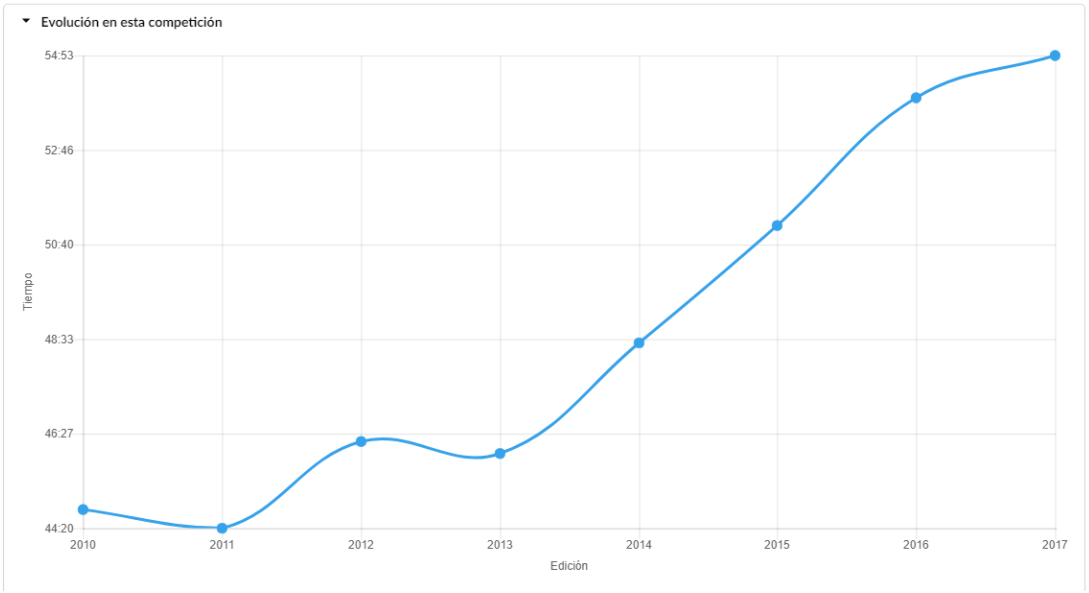


Figura 9.20. Evolución de resultados

En la siguiente sección encontraremos campos de texto libre para especificar con más detalle otra información de interés del resultado: descripción general, crónica y equipamiento.

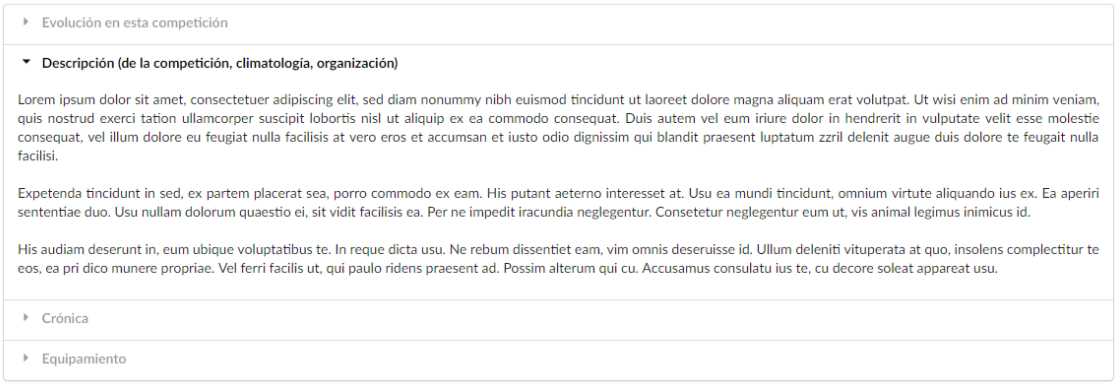


Figura 9.21. Descripción detallada, crónica y equipamiento

Y por último, podremos ver los comentarios realizados sobre este resultado o insertar uno nuevo.

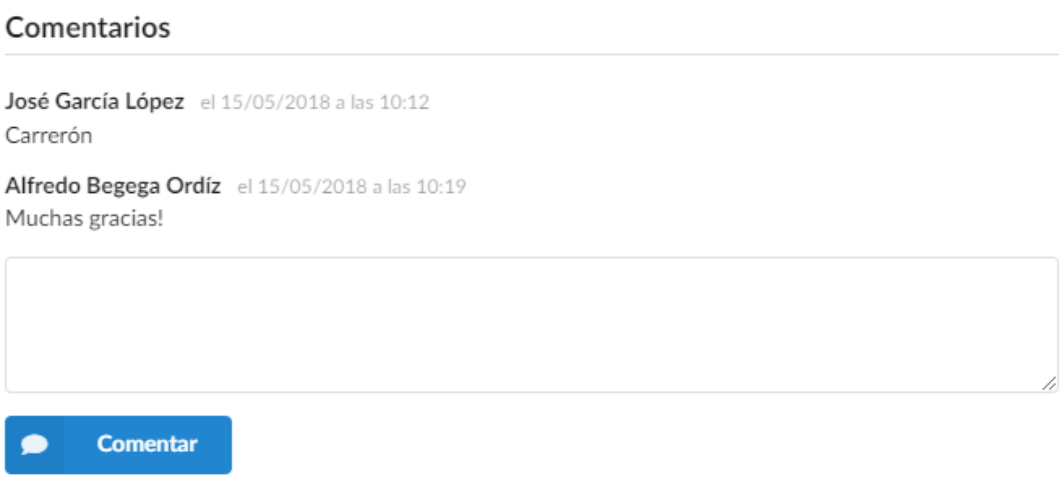


Figura 9.22. Comentarios de un resultado

9.3.12 Dar o quitar “me gusta” a un resultado

Ya sea a través de los listados de resultados que se muestran en la aplicación, o en el detalle de los mismos, dar o quitar “me gusta” es sumamente sencillo. Simplemente haremos click en el corazón asociado al resultado. En el caso de que este sea de color blanco, el corazón pasará a tener color rojo y habremos asignado un “me gusta” al resultado. En el caso de que el corazón esté de color rojo, pasará a color blanco y habremos quitado un “me gusta”.

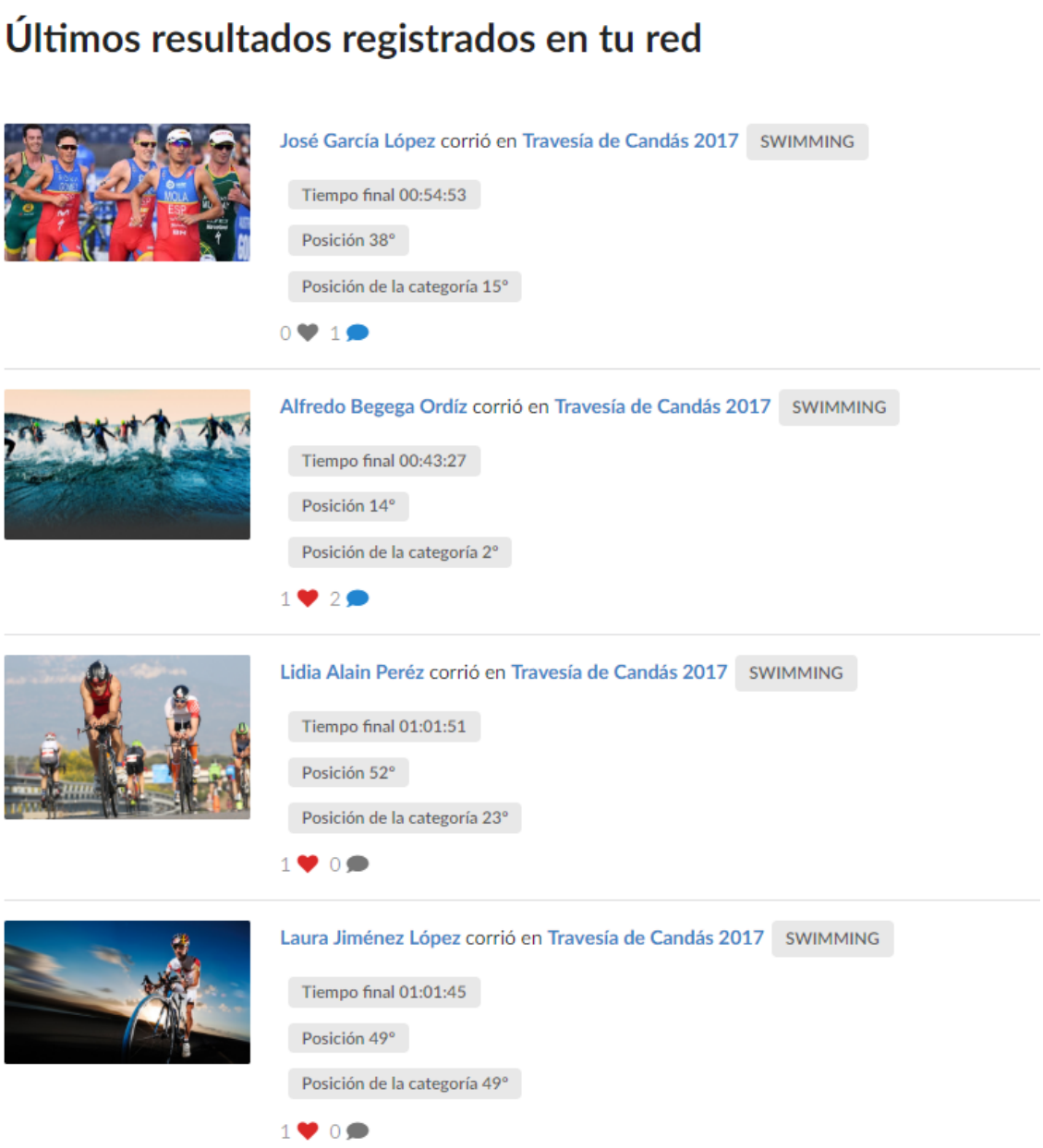


Figura 9.22. Listado de resultado con “me gusta”

También podremos observar el número de me gustas registrados para un resultado y si pasamos el curso sobre el corazón, ver que usuarios los indicaron.

9.3.13 Comentar un resultado

Utilizando los listados de resultados que se muestran en la aplicación, o en el detalle de los mismos se podrá utilizar el icono con forma de bocadillo para comentar un resultado. Esto nos conducirá al detalle del resultado, donde en su parte inferior, dispondremos del formulario habilitado para tal efecto.

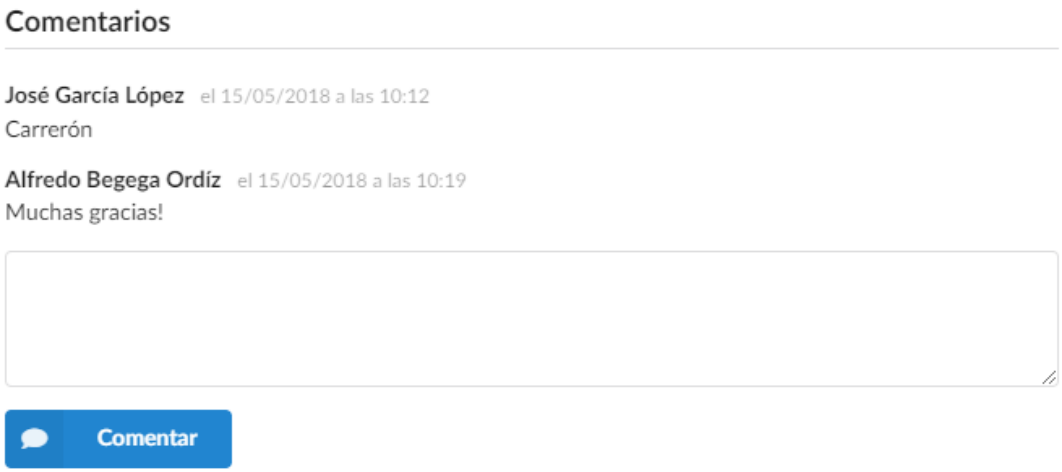


Figura 9.23. Formulario de comentarios

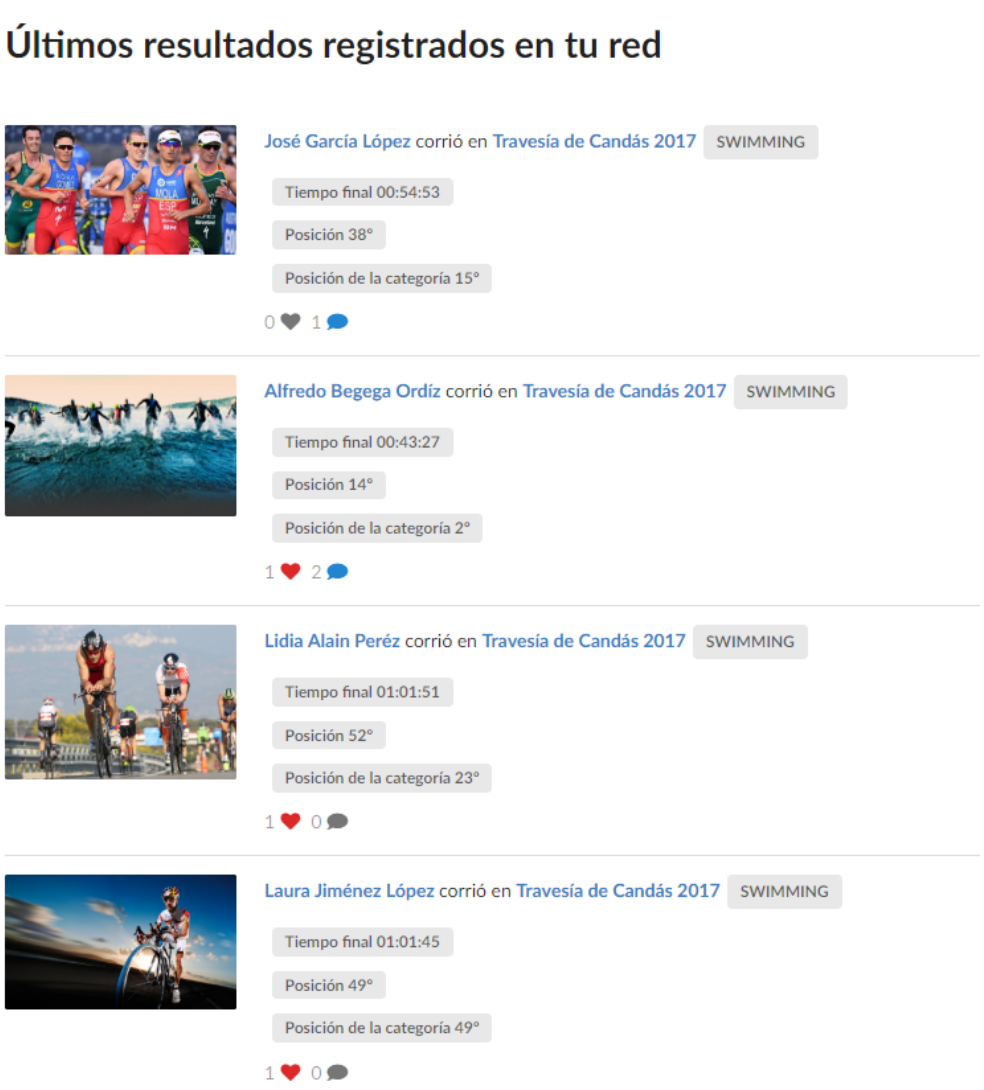


Figura 9.24. Listado de resultado con comentarios

También podremos observar el número de comentarios registrados para un resultado y si pasamos el curso sobre el bocadillo, ver que usuarios los hicieron.

9.3.14 Añadir un amigo a mi red

Cuando accedemos a los resultados de un usuario, si este aun no es amigo nuestro, podremos ver la opción de "*Iniciar amistad*" que nos permitirá añadir a este usuario a nuestra lista de amigos, y consecuentemente podremos ver en nuestra página de inicio de resultados, los resultados registre.

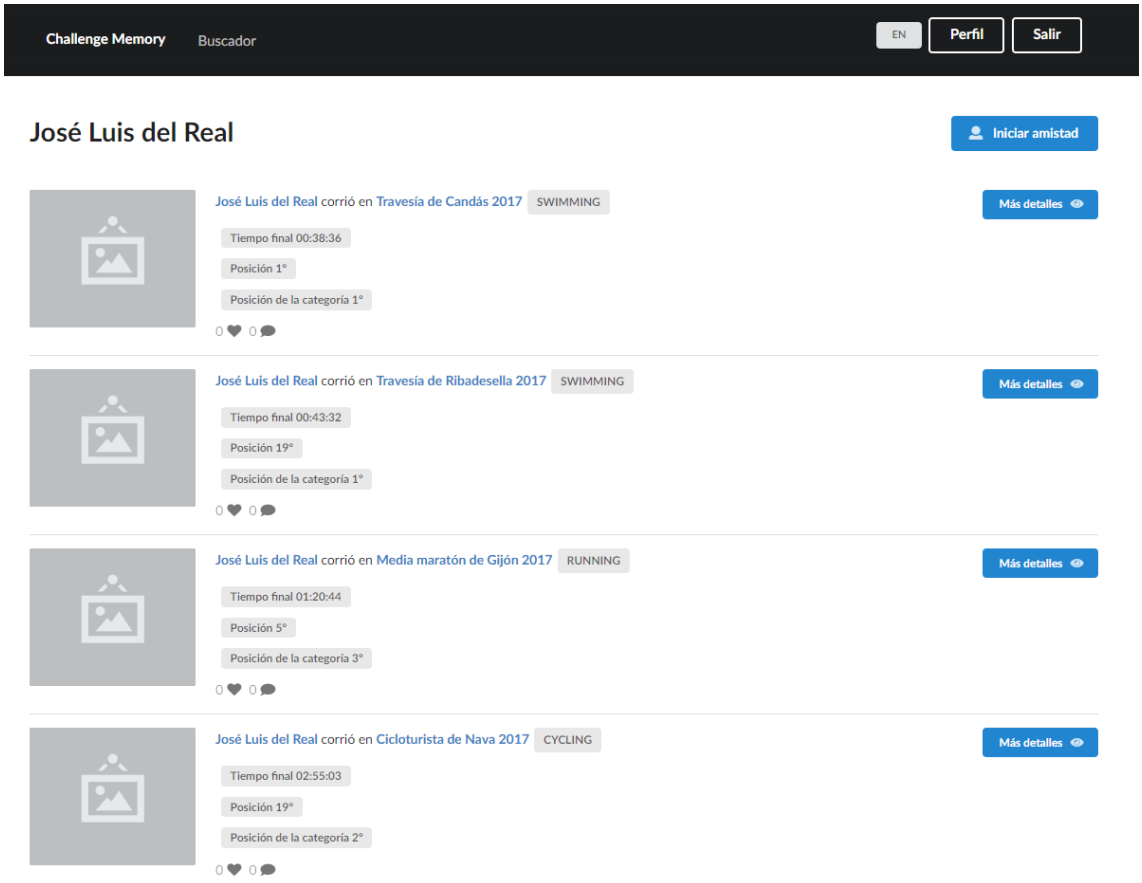


Figura 9.25. Página de resultados de un usuario con opción de iniciar amistad

9.3.15 Eliminar un amigo de mi red

Si queremos eliminar a un usuario de nuestra lista de amistades, simplemente tendremos que acceder a nuestro perfil. En la parte inferior de dicha pantalla, dispondremos de un listado con nuestros amigos, y la opción de eliminar para cada uno de ellos

Figura 9.26. Listado de amigos y opción eliminar

9.3.16 Ver resultados con avisos (administrador)

Si el usuario que accede a la aplicación es un administrador, en el menú superior izquierdo de la aplicación dispondrá de la opción "*Alertas*"

Figura 9.27. Menú superior del administrador

Dicha opción le dará acceso a la pantalla de revisión de alertas enviadas por los diferentes usuarios de la plataforma.

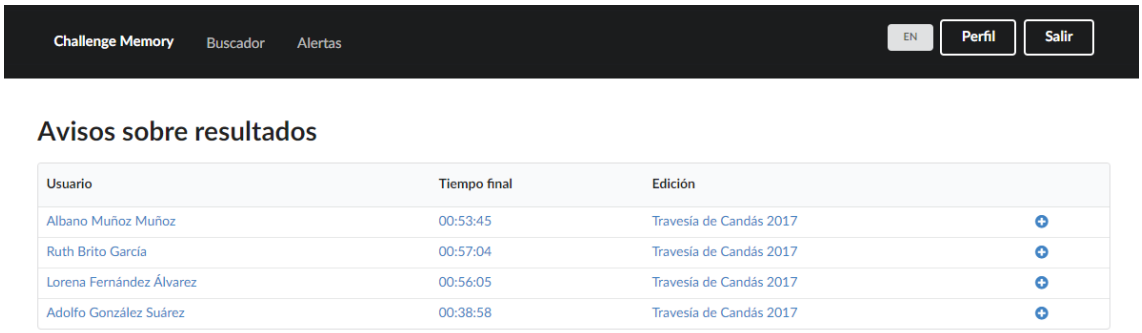


Figura 9.28. Pantalla de alertas del administrador

En este listado, haciendo uso de los enlaces podemos acceder al usuario, al resultado, a la edición del resultado y al detalle de los avisos. Reseñar que si accedemos al resultado, dispondremos de las opciones "*Borrar*" y "*Editar*" resultado como si fuésemos propietarios del mismo, y que será la vía para eliminar un resultado falso o sospechoso.

Ya en el detalle de un aviso, podemos ver qué usuarios enviaron aviso sobre el resultado y sus comentarios.

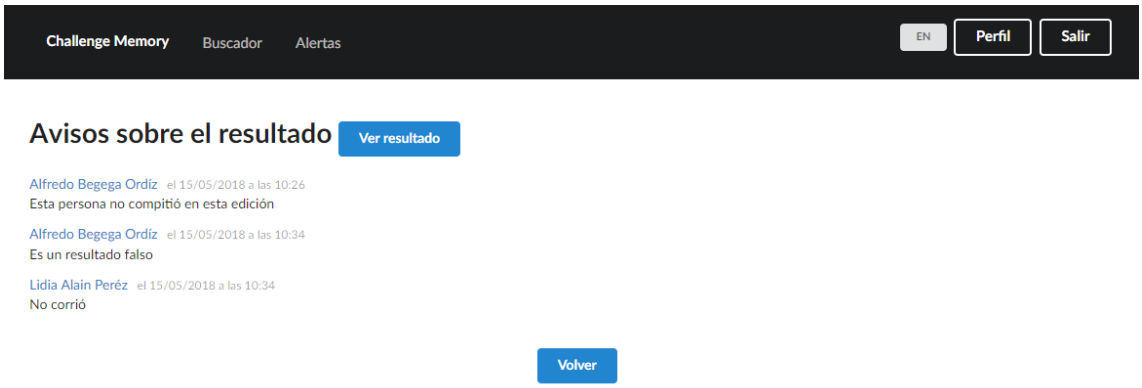


Figura 9.29. Detalle de un resultado con avisos

9.3.17 Desactivar un usuario (administrador)

Si el usuario que accede a la aplicación es un administrador, cuando acceda a los resultados de un usuario dispondrá la opción de "*Desactivar usuario*". Dicho opción, desactiva la cuenta del usuario objetivo de la acción, dejándole sin acceso a la plataforma.

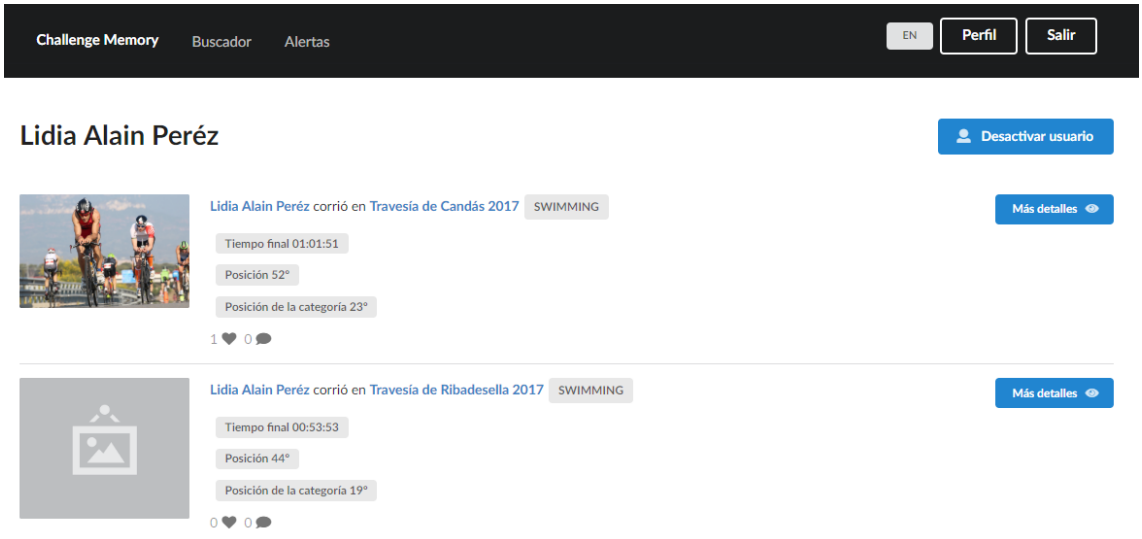


Figura 9.30. Opción de Desactivar Usuario en el detalle de usuario

9.4 Manual de Desarrollador

A continuación se darán unas pautas generales que permitirán al desarrollador poder establecer el entorno de desarrollo necesario para modificar o extender la funcionalidad de la aplicación.

La estructura del entorno de desarrollo utilizada, sigue las convenciones más habituales en trabajos J2EE con tecnología Spring. Por tanto, el desarrollador con conocimientos es este ámbito no tendrá que dedicar tiempo alguno a entenderlo.

Los pasos necesarios para trabajar con el desarrollo son los siguientes:

1. Instalar la versión más reciente de Spring Tool Suit (basado en eclipse)
2. Importar el código del proyecto como un proyecto Maven
3. Instalar la herramienta Compass MongoDB Community para consulta de BD MongoDB

Con estos tres pasos, ya tenemos todo lo necesario para trabajar en el desarrollo.

A partir de este punto, todas las funcionalidades han sido desarrolladas de una manera sencilla y fácilmente comprensibles por cualquier desarrollador. Simplemente deberá remitirse al Capítulo 12: Referencia Bibliográficas de esta documentación.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Después de todo el trabajo realizado, puedo sacar las siguientes conclusiones a nivel personal y profesional:

Personalmente ha sido un proceso complejo dada la falta de tiempo por mi condición de trabajador y persona con otras responsabilidades diarias. Pese a ello, el balance que hago es muy positivo, ya que ha incrementado mi capacidad de sacrificio y esfuerzo, además de mejorar la eficacia con la que realizo mis tareas, sean estas de índole profesional o personal.

Desde el punto de vista profesional, he intentado utilizar al máximo mi bagaje previo de más de 14 años laborales en el sector TIC. No tanto en el apartado de reutilización de conocimientos anteriores, de hecho he optado por la vía opuesta, en la que he iniciado un desarrollo desde cero y estableciendo sus cimientos, si no utilizando este bagaje para seleccionar adecuadamente el camino a seguir, optimizando al máximo y prescindiendo de pérdidas de tiempo innecesarias.

Si por un lado, el proyecto y su desarrollo se han podido ver beneficiados por mi experiencia previa, no serán de menos utilidad para mi trabajo futuro los conocimientos adquiridos durante este trabajo. Ya sea en parcelas menos habituales, como la planificación de proyectos o la realización de presupuestos, como en el trabajo de mí día a día, que ya en el momento que escribo este texto se ha visto afectado de manera positiva por el uso de la plataforma de desarrollo construida aquí. Y es este uno de los puntos que me gustaría destacar. De una manera progresiva, sin ser un objetivo inicialmente, todo el proceso ha ido derivando en la constitución de una plataforma de desarrollo tremendamente potente y a su vez fácil de utilizar, eficiente y que satisface la mayor parte de necesidades que cualquier desarrollo web pueda necesitar hoy en día.

Finalmente, también he conseguido la creación de un prototipo de red social que creo que puede tener un gran número de usuarios potenciales con las necesidades que cubre el servicio que ofrece. Esta afirmación no surge de un estudio de mercado, que queda fuera del alcance de este proyecto, si no desde la experiencia deportiva propia durante más de 15 años. En cualquier caso, considero que para el éxito final de este proyecto aún resta mucho trabajo por delante y que en este prototipo se han establecido las bases y se puede observar el potencial del trabajo.

Por tanto, considero que el proceso ha sido tremendamente enriquecedor a nivel profesional y personal, y que el sacrificio ha merecido la pena.

10.2 Ampliaciones

Este proyecto solamente establece las bases, tanto a nivel conceptual como de desarrollo de una red social que podría tener un largo recorrido. A partir de la estructura propuesta, se pueden llevar a cabo ampliaciones innumerables en diferentes ámbitos.

A continuación se reseñan algunas posibilidades, todas ellas no llevadas a cabo por falta de recursos materiales o disponibilidad de tiempo de ejecución:

- Mejorar las búsquedas e inserción de nuevos resultados mediante motores de inteligencia artificial y explotación de Big Data.
- Generación de Currículum Vitae deportivo.
- Desplegar la aplicación en un entorno distribuido de servidores en cluster.
- Aumentar la información almacenada en cada resultado, edición y competición tras realizar una encuesta a potenciales usuarios finales.
- Incluir geolocalización de resultados y GIS con información asociada.

Capítulo 11. Presupuesto

11.1 Desarrollo de Presupuesto

11.1.1 Presupuesto de Costes

En el Capítulo 4 de esta documentación, Sección 4.4 se elaboró un Presupuesto inicial de costes. Dicho presupuesto es suficiente y se adapta correctamente a las necesidades del proyecto para una valoración adecuada de costes internos y que asciende a **30.407,30 €.**

A continuación, se elaborará un presupuesto específico para hacer entrega al cliente, que incluya el beneficio adecuado para el cierre del proyecto y permita al cliente entender los costes que en él se reflejan.

11.1.2 Presupuesto del Cliente

11.1.2.1 Forma de trabajo

Antes de pasar al desarrollo de tiempos y el desglose de costes del proyecto, queremos que nuestro cliente conozca de manera global la forma de trabajo empleada a lo largo de este desarrollo. Para ello, le presentamos de manera gráfica la forma de trabajo utilizada.

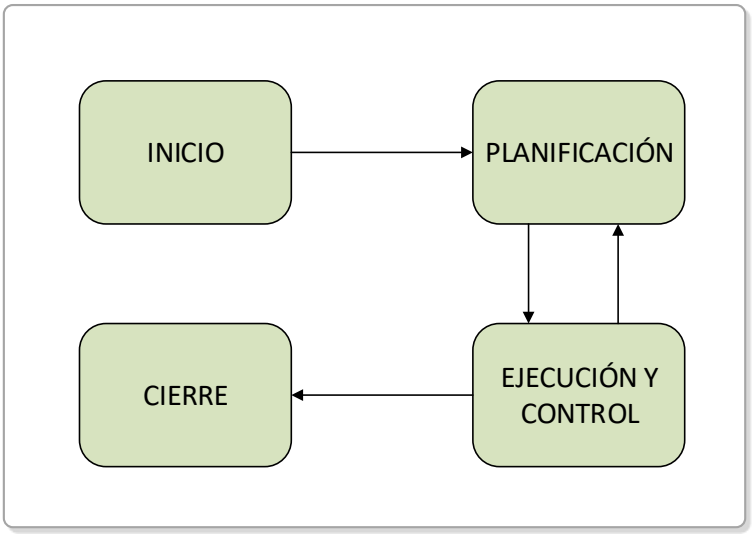


Figura 11.1. Etapas del proceso de trabajo

Como se puede ver, el trabajo comenzará en una fase de inicio, donde se sentarán las bases del proyecto, llegando a compromisos con el cliente y se establecerán las necesidades para completar el desarrollo. Más tarde, se procederá a una planificación de todo el trabajo y que dará como resultado un presupuesto que se presentará a continuación. Tras la planificación, será el momento de inicial la ejecución y control del proyecto, donde se irán controlando las

variables de riesgos detectadas y que podría afectar a la planificación inicialmente elaborada. Finalmente, será el momento de proceder al cierre del proyecto. Para ello, este debe ponerse en funcionamiento y pasar la pruebas de sistema y recibir el visto bueno por parte del cliente, momento en el cual se procederá a al cierre definitivo.

11.1.2.2 Tabla las entregas definidas

HITOS DEL PROYECTO	FIN
Aplicación web Challenge Memory	**27/05/18**
Planificación	
Análisis del alcance del proyecto y toma de decisiones técnicas	02/12/17
Identificación de tareas	09/12/17
Elaboración del cronograma y presupuesto	10/12/17
Análisis de riesgos	16/12/17
Entrega del presupuesto	
Instalación de servidores de aplicación y BD	17/12/17
Instalación de repositorio de software	17/12/17
Ejecución del proyecto	
Análisis de la arquitectura del proyecto	23/12/17
Construcción de la arquitectura del proyecto	24/12/17
Diseño de la interfaz de la aplicación	30/12/17
Módulo de inicio de la aplicación	06/01/18
Entrega del módulo	
Módulo de registro de usuarios	07/01/18
Entrega del módulo	
Módulo de acceso de usuarios	14/01/18
Entrega del módulo	
Módulo de resumen de resultados	21/01/18
Entrega del módulo	
Módulo de edición de perfil de usuario	28/01/18
Entrega del módulo	
Módulo de vista global de datos de usuario	03/02/18
Entrega del módulo	
Módulo de inserción de resultados	10/02/18
Entrega del módulo	
Módulo de clasificación virtual de competición	18/02/18
Entrega del módulo	
Módulo de vista global de un resultado	24/02/18
Entrega del módulo	
Módulo de aviso de información falsa	25/02/18
Entrega del módulo	
Módulo de búsqueda: competiciones y atletas	10/03/18
Entrega del módulo	
Módulo de evolución de atleta en una competición	24/03/18
Entrega del módulo	

Cierre del proyecto	
Entrega y puesta en marcha	
Mantenimiento y resolución de incidencias	14/04/18

11.1.2.3 Presupuesto detallado

CONCEPTO	Uds./Horas	COSTE
Inicio y planificación del proyecto	35	3.498,00 €
Adquisición de servidor Web y BD	1	1.100,00 €
Diseño de estructura e instalación de servidores	23	1.430,00 €
Diseño de la interfaz de la aplicación	10	330,00 €
Módulo de inicio de la aplicación	22	1.584,00 €
Módulo de registro de usuarios	15	1.067,00 €
Módulo de acceso de usuarios	15	1.067,00 €
Módulo de resumen de resultados	23	1.639,00 €
Módulo de edición de perfil de usuario	17	1.243,00 €
Módulo de vista global de datos de usuario	11	847,00 €
Módulo de inserción de resultados	24	1.694,00 €
Módulo de clasificación virtual de competición	26	1.804,00 €
Módulo de vista global de un resultado	13	957,00 €
Módulo de aviso de información falsa	11	847,00 €
Módulo de búsqueda: competiciones y atletas	32	2.200,00 €
Módulo de evolución de atleta en una competición	38	2.596,00 €
Entrega y puesta en marcha	2	220,00 €
Mantenimiento y resolución de incidencias	62	3.520,00 €
Inicio y planificación del proyecto	35	3.498,00 €
Adquisición de servidor Web y BD	1	1.100,00 €
	SUBTOTAL	27.643,00 €
	IVA (21%)	5.805,03 €
	TOTAL	**33.448,03 €**

Capítulo 12. Referencias Bibliográficas

12.1 Referencias en Internet

[Spring Boot Reference Guide]: Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze, Michael Simons, Vedran Pavić, Jay Bryant, Madhura Bhave.

https://docs.spring.io/spring-boot/docs/current/reference/html/index.html

[Using Thymeleaf]:

https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html

[Thymeleaf + Spring]:

https://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html

[Getting up and running with Semantic UI]:

https://semantic-ui.com/introduction/getting-started.html

[Welcome to the MongoDB Docs]:

https://docs.mongodb.com/

[Lombok features]:

https://projectlombok.org/features/all

Capítulo 13. Apéndices

13.1 Contenido Entregado

13.1.1 Contenidos

13.1.1.1.1 Estructura general directorios

Directorio	Contenido
./challenge-memory	Contiene toda la estructura de directorios del proyecto para desarrollo.
./instalacion	Ficheros utilizados para la instalación del proyecto.
./documentacion	Contiene toda la documentación asociada al proyecto.
./presentacion	Contiene la presentación del TFM
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto (lógicamente sólo las que sean distribuibles).
./herramientas/desarrollo	Ficheros de instalación de las herramientas utilizadas en el desarrollo

13.1.1.1.2 Estructura de Directorios de desarrollo

Directorio	Contenido
./ Directorio raíz	Contiene los ficheros de proyecto del IDE utilizado.
./dist	Directorio donde se sitúan los ficheros para la distribución del proyecto.
./doc	Contiene toda la documentación relativa al proyecto
./src	Ficheros fuente
./src/main/java	Todos los ficheros *Java*, lógicamente agrupados en los paquetes correspondientes.
./src/main/resources	Este directorio contiene los ficheros *de recursos de la aplicación, como: las plantillas HTML, CSS, javascript, properties y configuración de la aplicación.*
./src/test/java	Contiene todas las pruebas unitarias utilizadas en el proceso de prueba automatizado

13.1.2 Ficheros de Configuración

Toda la configuración se localiza en el fichero application.yml de Spring Boot que se encuentra en el directorio src/main/resources. La información relacionada sobre la configuración y parametrización del mismo se puede consultar en enlace a continuación:

https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

13.2 Código Fuente

13.2.1 es.uniovi.challengememory

13.2.1.1 ChallengeMemoryApplication.java

```
@EnableAutoConfiguration
@SpringBootApplication
@EnableMongoRepositories
@EnableConfigurationProperties(FileStorageProperties.class)
public class ChallengeMemoryApplication extends WebMvcConfigurerAdapter {

	public static void main(String[] args) {
		SpringApplication.run(ChallengeMemoryApplication.class, args);
	}

	@Bean
	public CommandLineRunner init(StorageService storageService) {
		return (args) -> {
			storageService.init();
		};
	}

	@Bean
	public LocaleResolver localeResolver() {
		SessionLocaleResolver slr = new SessionLocaleResolver();
		slr.setDefaultLocale(new Locale("es"));
		return slr;
	}

	@Bean
	public LocaleChangeInterceptor localeChangeInterceptor() {
		LocaleChangeInterceptor lci = new LocaleChangeInterceptor();
		lci.setParamName("lang");
		return lci;
	}

	@Override
	public void addInterceptors(InterceptorRegistry registry) {
		registry.addInterceptor(localeChangeInterceptor());
	}

	@Bean
	public Function<String, String> currentUrlWithoutParam() {
	    return param ->
ServletUriComponentsBuilder.fromCurrentRequest().replaceQueryParam(param).toUriString();
	}

	@Bean
	public ResourceBundleMessageSource messageSource() {
	 ResourceBundleMessageSource source = new ResourceBundleMessageSource();
	 source.setBasenames("i18n/messages");
	 source.setDefaultEncoding("UTF-8");
	 source.setUseCodeAsDefaultMessage(true);
	 return source;
	}

	@Bean
    public SpringDataDialect springDataDialect() {
	    return new SpringDataDialect();
    }

	@Bean
	public SpringSecurityDialect securityDialect() {
	    return new SpringSecurityDialect();
	}

}
```

13.2.1.2 SecurityConfiguration.java

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled=true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

        @Autowired
        @Qualifier("userDetailsService")
        private UserDetailsServiceImpl userDetailsService;

        @Autowired
        private UserRepository userRepository;

        @Override
    protected void configure(HttpSecurity http) throws Exception {

                http.headers().frameOptions().disable();
                http.csrf().disable();
                http.authorizeRequests()
                 .antMatchers(
                                "/",
                                "/error",
                                "/404",
                                "/500",
                                "/h2/**",
                                "/signup",
                                "/terms",
                                "/confirmation/**",
                         "/assets/**",
                         "/assets/semantic-ui-calendar/**",
                         "/assets/semantic/**").permitAll()
                 .anyRequest()
                 .authenticated()
                 .and()
                 .formLogin()
                        .loginPage("/login")
                        .defaultSuccessUrl("/")
                        .failureUrl("/login?error")
                        .permitAll()
              .and()
             .logout()
                .invalidateHttpSession(false)
                .deleteCookies("JSESSIONID")
                .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                .logoutSuccessUrl("/")
                 .permitAll()
             .and()
             .exceptionHandling().accessDeniedPage("/403");

    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

        Optional<es.uniovi.challengememory.model.User> user =
userRepository.findByUsernameAndState("admin", State.ACTIVE);
        if (!user.isPresent()) {
                        userRepository.save(es.uniovi.challengememory.model.User.builder()
                                                .name("Administrador")
                                                .username("admin")
                                                .passwordHash(new
BCryptPasswordEncoder().encode("admin"))
                                                .privated(true)
                                                .role(Role.ADMIN)
                                                .state(State.ACTIVE).build());
                }

        auth.userDetailsService(userDetailsService).passwordEncoder(new
BCryptPasswordEncoder());

    }

}
```

13.2.2 es.uniovi.challengememory.controller

13.2.2.1 CompetitionController.java

```
@Controller
public class CompetitionController {

	private final Logger logger =
LoggerFactory.getLogger(CompetitionController.class);

	@Autowired
	private EditionRepository editionRepository;

	@Autowired
	private CompetitionRepository competitionRepository;

	@Autowired
	private ResultRepository resultRepository;

	@Autowired
	private UserRepository userRepository;

	@Autowired
	private MessageSource messages;


	@GetMapping("/editionDetail")
    public String editionDetail(@RequestParam String id, @RequestParam(required=false)
String sex,
		@RequestParam(required=false) String onlyFriends, Model model) {

		Authentication authentication = CMUtils.getAuthentication();
		User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

		CompetitionEdition edition = editionRepository.findOne(id);
		List<Result> results =
resultRepository.findAllByEditionOrderByTime(edition);

		final Sex sexFilter = (Strings.isEmpty(sex)) ? null : Sex.valueOf(sex);
		final boolean friends = (Strings.isEmpty(onlyFriends) &&
!"on".equals(onlyFriends)) ? false : true;

		results = results.stream()
				.filter(r -> !r.getUser().getPrivated()
						&&
State.ACTIVE.equals(r.getUser().getState())
						&& (sexFilter == null || (sexFilter
!= null && sexFilter.equals(r.getUser().getSex())))
						&& (!friends || (friends &&
user.isFriend(r.getUser())))
					)
				.collect(Collectors.toList());

		String[] sexs =
Arrays.stream(Sex.values()).map(Enum::name).toArray(String[]::new);

		model.addAttribute("selectedSex", sex);
		model.addAttribute("onlyFriends", friends);
		model.addAttribute("sexs", sexs);
		model.addAttribute("edition", edition);
		model.addAttribute("results", results);

	 if (logger.isDebugEnabled())
			logger.debug("Se accede a la clasificacion de la edición con
identificador " + edition.getId());

	 return "editionDetail";
    }

	@GetMapping("/addEdition")
    public String addEdition(@ModelAttribute Competition competition, Model model) {
```

```
                CompetitionEdition edition = CompetitionEdition.builder().build();
                if (competition != null && competition.getId() != null)
                        edition.setCompetition(competition);

                model.addAttribute("edition", edition);
                return "editionForm";
        }

        @PostMapping("/editionForm")
    public String editionForm(@ModelAttribute CompetitionEdition edition, Locale locale,
final RedirectAttributes redirectAttributes) {

                try {

                        SimpleDateFormat sdf = new SimpleDateFormat("yyyy");
                        edition.setEditionName(edition.getCompetition().getName() + " " +
sdf.format(edition.getDate()));

                        List<CompetitionEdition> persistedEditions =
editionRepository.findAllByEditionName(edition.getEditionName());

                        if (persistedEditions.size() > 0) {

                                redirectAttributes.addFlashAttribute("result",
Result.builder().build());
                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("editionSave.error", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"warning");
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("editionSave.error.msg", new String[] {edition.getEditionName()},
locale));

                        } else {

                                if (edition.getId() != null) {

                                        CompetitionEdition persistedEdition =
editionRepository.findOne(edition.getId());
                                        BeanUtils.copyProperties(edition, persistedEdition,
"id", "competition");

                                        edition = persistedEdition;
                                }

                                edition = editionRepository.save(edition);

                                Result result = Result.builder().edition(edition).build();

                                redirectAttributes.addFlashAttribute("result", result);
                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("editionSaved.msg", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"success");
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("editionSaved.msg", null, locale));

                         if (logger.isDebugEnabled())
                                logger.debug("Se guardaron los datos de la edición Id: " +
edition.getId());

                        }

                } catch (Exception ex) {

                        logger.error("Se produjo un error mientras se guardaba la
edición", ex);

                        redirectAttributes.addFlashAttribute("result",
Result.builder().build());
                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("editionSave.error", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("editionSave.error.msg", null, locale));
                }

         return "redirect:/addResult/edition";
    }
```

```
        @GetMapping("/searchEditions")
    public String searchEditions(@RequestParam String q, Model model) {

                Page<CompetitionEdition> editions = editionRepository
                                .findByEditionNameContainingIgnoreCase(q, new
PageRequest(0, Constants.PAGE_SIZE));

                model.addAttribute("editions", editions);

                return "fragments/fragments :: editions";
        }

        @GetMapping("/addCompetition")
    public String addCompetition(Model model) {

                String[] competitionTypes =
Arrays.stream(CompetitionType.values()).map(Enum::name).toArray(String[]::new);

                model.addAttribute("competitionTypes", competitionTypes);
                model.addAttribute("competition", Competition.builder().build());

                return "competitionForm";
        }

        @PostMapping("/competitionForm")
    public String competitionForm(@ModelAttribute Competition competition, Locale
locale, final RedirectAttributes redirectAttributes) {

                try {

                        List<Competition> persistedCompetitions =
competitionRepository.findAllByName(competition.getName());

                        if (persistedCompetitions.size() > 0) {

                                redirectAttributes.addFlashAttribute("result",
Result.builder().build());
                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("competitionSave.error", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"warning");
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("competitionExist.error.msg", new String[] {competition.getName()},
locale));

                        } else {

                                if (competition.getId() != null) {

                                        Competition persistedCompetition =
competitionRepository.findOne(competition.getId());
                                        BeanUtils.copyProperties(competition,
persistedCompetition, "id");
                                        competition = persistedCompetition;
                                }

                                competition = competitionRepository.save(competition);

                                redirectAttributes.addFlashAttribute("competition",
competition);
                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("competitionSaved", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"success");
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("competitionSaved.msg", null, locale));

                         if (logger.isDebugEnabled())
                                logger.debug("Se guardaron los datos de la competición Id:
" + competition.getId());

                        }

                } catch (Exception ex) {

                        logger.error("Se produjo un error mientras se guardaba la
edición", ex);
```

```
                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("competitionSave.error", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("competitionSave.error.msg", null, locale));
                }

                return "redirect:/addEdition";
        }

                @GetMapping("/searchCompetitions")
        public String searchCompetitions(@RequestParam String q, Model model) {

                Page<Competition> competitions = competitionRepository
                                .findByNameContainingIgnoreCase(q, new PageRequest(0,
Constants.PAGE_SIZE));

                model.addAttribute("competitions", competitions);

                return "fragments/fragments :: competitions";
        }

                @GetMapping("/competitionDetail")
        public String competitionDetail(@RequestParam String id,
@RequestParam(required=false) String editionId, Locale locale,
                final RedirectAttributes redirectAttributes, Model model) {

                List<CompetitionEdition> editions =
editionRepository.findAllByCompetitionOrderByDateDesc(id);

                if (editions != null && editions.size() > 0) {

                        CompetitionEdition edition = editions.get(0);
                        if (editionId != null) {
                                for (CompetitionEdition e : editions) {
                                        if (editionId.equals(e.getId()))
                                                edition = e;
                                }
                        }

                        List<Result> results =
resultRepository.findAllByEditionOrderByTime(edition);

                        results = results.stream()
                                        .filter(r -> !r.getUser().getPrivated() &&
State.ACTIVE.equals(r.getUser().getState()))
                                        .collect(Collectors.toList());

                        model.addAttribute("editions", editions);
                        model.addAttribute("edition", edition);
                        model.addAttribute("results", results);

                 if (logger.isDebugEnabled())
                        logger.debug("Se accede al detalle de la competicion  " +
edition.getId());

                 return "competitionDetail";

                } else {

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("competitionEditions.error", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("competitionEditions.error.msg", null, locale));

                        return "redirect:/search";
                }
        }

}
```

13.2.2.2 GeneralController.java

```
@Controller
public class GeneralController {

	private final Logger logger = LoggerFactory.getLogger(GeneralController.class);

	@Autowired
	private UserRepository userRepository;

	@Autowired
	private MailServiceImpl mailService;

	@Autowired
	private MessageSource messages;

	@Value("${mail.sender}")
	private String sender;

	@GetMapping("/")
    public String index() {

		if (CMUtils.isAuthenticated())
			return "redirect:/dashboard";
		else
			return "index";
    }

	@GetMapping("/login")
    public String login() {
        return "login";
    }

	@GetMapping("/signup")
    public String signup(Model model) {

        model.addAttribute("user", User.builder().build());
        return "signup";
    }

	@GetMapping("/terms")
    public String terms(Model model) {
        return "terms";
    }

	@PostMapping("/signup")
    public String signup(@ModelAttribute User user, Locale locale, Model model) {

		if (user.getTerms()) {

			if (!user.getPassword().isEmpty() &&
user.getPassword().equals(user.getPasswordConfirmation())) {

				user.setPasswordHash(new
BCryptPasswordEncoder().encode(user.getPassword()));
				user.setRole(Role.USER);
				user.setState(State.INACTIVE);
				user.setPrivated(false);

				try {
					if ("admin".equals(user.getUsername()))
						throw new DuplicateKeyException("admin es un
nombre de usuario no permitido");

					user = userRepository.save(user);

					ServletUriComponentsBuilder builder =
ServletUriComponentsBuilder.fromCurrentContextPath();
					builder.scheme("https");
					URI uri = builder.build().toUri();

					String[] params = new String[] {
						user.getName(),
						uri.toASCIIString() + "/confirmation/" +
user.getId(),
					};
```

```
                                        String[] destinatariosList =
user.getUsername().split(",");
                                        mailService.prepareAndSend(sender,
Arrays.asList(destinatariosList),

        messages.getMessage("confirmation.email.title", null, locale),

        messages.getMessage("confirmation.email.text", params, locale)
                                        );

                                        model.addAttribute("message",
messages.getMessage("userCreated", null, locale));
                                        model.addAttribute("messageType", "success");
                                        model.addAttribute("messageDescription",
messages.getMessage("userCreated.msg", null, locale));

                                } catch (Throwable ex) {

                                        if (ex instanceof DuplicateKeyException)
                                                model.addAttribute("messageDescription",
messages.getMessage("userExist.error", null, locale));
                                        if (ex instanceof MailException)
                                                model.addAttribute("messageDescription",
messages.getMessage("mailServer.error", null, locale));

                                        model.addAttribute("message",
messages.getMessage("userSave.error", null, locale));
                                        model.addAttribute("messageType", "error");
                                        logger.error(ex.getMessage());

                                }

                        }
                } else {
                        model.addAttribute("message",
messages.getMessage("userSave.error", null, locale));
                        model.addAttribute("messageDescription",
messages.getMessage("terms.error", null, locale));
                        model.addAttribute("messageType", "warning");
                }

                model.addAttribute("user", User.builder().build());
         return "signup";
    }

        @GetMapping("/confirmation/{id}")
    public String confirmation(@PathVariable String id) {

                User user = userRepository.findOne(id);
                user.setState(State.ACTIVE);
                userRepository.save(user);

         return "confirmation";
    }

        @GetMapping("/error")
        public String error() {
            return "errors/error";
        }

        @GetMapping("/500")
        public String internalError () {
            return "errors/500";
        }

        @GetMapping("/403")
        public String accessDenied() {
            return "errors/403";
        }

        @GetMapping("/404")
        public String notFound() {
            return "errors/404";
        }

        @GetMapping("/400")
        public String badRequest() {
            return "errors/400";
        }
```

```
        @GetMapping("/401")
        public String unauthorized () {
            return "errors/401";
        }

}
```

13.2.2.3 ResultController.java

```
@Controller
@SessionAttributes({"result"})
public class ResultController {

        private final Logger logger = LoggerFactory.getLogger(ResultController.class);

        private final StorageService storageService;

        @Autowired
        private UserRepository userRepository;

        @Autowired
        private EditionRepository editionRepository;

        @Autowired
        private ResultRepository resultRepository;

        @Autowired
        private MessageSource messages;

        @Autowired
    public ResultController(StorageService storageService) {
        this.storageService = storageService;
    }

        @GetMapping("/dashboard")
    public String dashboard(@PageableDefault(size = Constants.PAGE_SIZE, page = 0)
                        @SortDefault(sort = "publishDate", direction =
Sort.Direction.DESC)
                    Pageable pageable, Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                List<User> users = user.getPublicAndActiveFriends();
                if (users == null) users = new ArrayList<>();
                users.add(user);
                Page<Result> paginatedResults = resultRepository.findByUserIn(users,
pageable);

                model.addAttribute("paginatedResults", paginatedResults);
                model.addAttribute("user", user);

                return "dashboard";
        }

        @GetMapping("/resultDetail")
    public String resultDetail(@RequestParam String id, @RequestParam(required=false)
String comment, Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                Result result = resultRepository.findOne(id);

                List<CompetitionEdition> editions = editionRepository

        .findAllByCompetitionOrderByDateDesc(result.getEdition().getCompetition().getId()
);
                List<Result> resultsHistory =
resultRepository.findAllByUserAndEditionIn(result.getUser(), editions);
```

```
                resultsHistory.sort((r1, r2) ->
r1.getEdition().getDate().compareTo(r2.getEdition().getDate()));

                if (comment != null)
                        model.addAttribute("comment", comment);

                model.addAttribute("result", result);
                model.addAttribute("resultsHistory", resultsHistory);
                model.addAttribute("user", user);

        if (logger.isDebugEnabled())
                logger.debug("Se accede al detalle del resultado con identificador " +
result.getId());

        return "resultDetail";
    }

        @GetMapping("/editResult")
    public String editResult(@RequestParam String id, Model model) {

                List<CompetitionEdition> editions = editionRepository.findAll();
                Result result = resultRepository.findOne(id);

                model.addAttribute("result", result);
                model.addAttribute("editions", editions);

                return "resultForm";
        }

        @GetMapping("/addResult/edition")
    public String addResultFromEdition(@ModelAttribute Result result, Model model) {

                if (result == null)
                        result = Result.builder().build();

                model.addAttribute("result", result);
                return "resultForm";
        }

        @GetMapping("/addResult")
    public String addResult(@ModelAttribute Result result, SessionStatus status, Model
model) {

                status.setComplete();
                model.addAttribute("result", Result.builder().build());

                return "resultForm";
        }

        @GetMapping("/deleteResult")
    public String deleteResult(@RequestParam String id, Locale locale, final
RedirectAttributes redirectAttributes) {


                Result result = resultRepository.findOne(id);
                resultRepository.delete(result);

                redirectAttributes.addFlashAttribute("message",
messages.getMessage("resultDeleted", null, locale));
                redirectAttributes.addFlashAttribute("messageType", "success");
                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("resultDeleted.msg", null, locale));

                return "redirect:/dashboard";
        }

        @PostMapping("/uploadImage")
    @ResponseBody
    public ResponseEntity<?> uploadImages(@RequestParam("file") MultipartFile file,
@ModelAttribute Result result) {

                if (result.getImages() == null) result.setImages(new ArrayList<>());

                String filename = "image_" + new Date().getTime() +
file.getOriginalFilename().substring(file.getOriginalFilename().indexOf("."));
                result.getImages().add(
                        File.builder()
                                .type(FileType.IMAGE)
                                .path(filename)
```

```
                        .build()
                );

        storageService.store(file, filename);

         return ResponseEntity.ok().build();
    }

        @PostMapping("/uploadClassification")
    @ResponseBody
    public ResponseEntity<?> uploadClassification(@RequestParam("file") MultipartFile
file, @ModelAttribute Result result) {

                String filename = "classification_" + new Date().getTime() +
file.getOriginalFilename().substring(file.getOriginalFilename().indexOf("."));
                result.setClassification(
                        File.builder()
                                .type(FileType.DOC)
                                .path(filename)
                        .build()
                );

        storageService.store(file, filename);

         return ResponseEntity.ok().build();
    }

        @PostMapping("/resultForm")
    public String resultForm(@ModelAttribute Result result, SessionStatus status, Locale
locale, final RedirectAttributes redirectAttributes) {

                String returnUrl = "redirect:/dashboard";

                try {

                        Authentication authentication = CMUtils.getAuthentication();
                        User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                        List<Result> userEditionResults =
resultRepository.findAllByUserAndEdition(user, result.getEdition());

                        if (result.getId() == null && userEditionResults.size() > 0) {

                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("resultSave.error", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"warning");
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("resultExist.error.msg", new String[]
{result.getEdition().getEditionName()}, locale));

                        } else {

                                if (result.getId() != null) {

                                        Result persistedResult =
resultRepository.findOne(result.getId());
                                        BeanUtils.copyProperties(result, persistedResult,
                                                        "id", "user", "publishDate",
"edition");
                                        result = persistedResult;

                                        returnUrl = "redirect:/resultDetail?id=" +
result.getId();

                                } else {

                                        result.setPublishDate(new Date());
                                        result.setUser(user);
                                }

                                resultRepository.save(result);

                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("resultSaved", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"success");
```

```
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("resultSaved.msg", null, locale));

                        if (logger.isDebugEnabled())
                                logger.debug("Se guardaron los datos de resultado Id: " +
result.getId());
                        }

                } catch (Exception ex) {

                        logger.error("Se produjo un error mientras se guardaba el
resultado", ex);

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("resultSave.error", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("resultSave.error.msg", null, locale));
                }

                status.setComplete();

         return returnUrl;
    }

        @GetMapping("/loadLikesAndComments")
    public String loadLikesAndComments(@RequestParam String id, Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                Result result = resultRepository.findOne(id);

                model.addAttribute("loadedResult", result);
                model.addAttribute("user", user);

                return "fragments/fragments :: resultLikesAndComments";
        }

        @GetMapping("/like")
    public String like(@RequestParam String id, Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                Result result = resultRepository.findOne(id);

                if (result.getLikes() == null) result.setLikes(new ArrayList<>());

                result.getLikes().add(Like.builder().date(new
Date()).user(user).build());
                result = resultRepository.save(result);

                model.addAttribute("loadedResult", result);
                model.addAttribute("user", user);

                return "fragments/fragments :: resultLikesAndComments";
        }

        @GetMapping("/unlike")
    public String unlike(@RequestParam String id, Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                Result result = resultRepository.findOne(id);

                Like likeToRemove = null;
                if (result.getLikes() != null)
                 for (Like like : result.getLikes())
                        if (like.getUser().equals(user))
                                likeToRemove = like;

                result.getLikes().remove(likeToRemove);

                result = resultRepository.save(result);
```

```
                model.addAttribute("loadedResult", result);
                model.addAttribute("user", user);

                return "fragments/fragments :: resultLikesAndComments";
        }

        @PostMapping("/commentForm")
        public String commentForm(@RequestParam String id, @RequestParam String text,
Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                Result result = resultRepository.findOne(id);

                if (result.getComments() == null) result.setComments(new ArrayList<>());

                result.getComments().add(Comment.builder().text(text).date(new
Date()).user(user).build());
                resultRepository.save(result);

                return "redirect:/resultDetail?id=" + result.getId();
        }

        @GetMapping("/warnings")
        @PreAuthorize("hasAuthority('ADMIN')")
    public String warnings(Model model) {

                List<Result> warningResults = resultRepository.findResultsWithWarnings();

                model.addAttribute("warningResults", warningResults);

                return "warnings";
        }

        @GetMapping("/warningsDetail")
        @PreAuthorize("hasAuthority('ADMIN')")
    public String warningsDetail(@RequestParam String id, Model model) {

                Result result = resultRepository.findOne(id);

                model.addAttribute("result", result);

                return "warningsDetail";
        }

        @GetMapping("/notifyWarning")
    public String notifyWarning(@RequestParam String id, Model model) {

                Result result = resultRepository.findOne(id);

                model.addAttribute("result", result);

                return "warningForm";
        }

        @PostMapping("/warningForm")
        public String warningForm(@RequestParam String id, @RequestParam String text,
Locale locale, final RedirectAttributes redirectAttributes) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                Result result = resultRepository.findOne(id);

                if (result.getWarnings() == null) result.setWarnings(new ArrayList<>());

                result.getWarnings().add(Warning.builder().text(text).date(new
Date()).user(user).build());
                resultRepository.save(result);

                redirectAttributes.addFlashAttribute("message",
messages.getMessage("warningSent", null, locale));
                redirectAttributes.addFlashAttribute("messageType", "success");
                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("warningSent.msg", null, locale));
```

```
                return "redirect:/resultDetail?id=" + result.getId();
        }

        @ModelAttribute
        public void addAttributes(Model model) {
                if (!model.containsAttribute("result"))
                        model.addAttribute("result", Result.builder().build());
        }

}
```

13.2.2.4 SearchController.java

```
@Controller
public class SearchController {

        private final Logger logger = LoggerFactory.getLogger(SearchController.class);

        @Autowired
        private UserRepository userRepository;

        @Autowired
        private CompetitionRepository competitionRepository;

        @GetMapping("/search")
    public String searchForm(@RequestParam(required=false) String q,
@RequestParam(required=false) String t,
                @PageableDefault(size = Constants.SEARCH_PAGE_SIZE, page = 0)
                        @SortDefault(sort = "score", direction = Sort.Direction.DESC)
                    Pageable pageable, Model model) {

                Object searchResults = null;

                if ("users".equals(t))
                        searchResults = userRepository.findByText(q, pageable);

                if ("competitions".equals(t))
                        searchResults = competitionRepository.findByText(q, pageable);

                model.addAttribute("q", q != null ? q : "");
                model.addAttribute("t", t != null ? t : "competitions");
                model.addAttribute("searchResults", searchResults);

                 if (logger.isDebugEnabled())
                 logger.debug("Se realiza la busqueda de " + t + " con el texto: " + q);

         return "search";
    }

}
```

13.2.2.5 UserController.java

```
@Controller
public class UserController {

        private final Logger logger = LoggerFactory.getLogger(UserController.class);

        @Autowired
        private UserRepository userRepository;

        @Autowired
        private ResultRepository resultRepository;

        @Autowired
        private MessageSource messages;
```

```
       @GetMapping("/profile")
    public String profile(Model model) {

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                List<User> friends = user.getPublicAndActiveFriends();
                Collections.sort(friends);

                user.setPassword("------");
                user.setPasswordConfirmation("------");
         model.addAttribute("user", user);
         model.addAttribute("friends", friends);

         if (logger.isDebugEnabled())
                logger.debug("Se accede al perfil del usuario con identificador " +
user.getId());

         return "profile";
    }

       @GetMapping("/desactivateUser")
    public String desactivateUser(@RequestParam String id, Locale locale, final
RedirectAttributes redirectAttributes) {


                User user = userRepository.findOne(id);
                user.setState(State.INACTIVE);
                userRepository.save(user);

                redirectAttributes.addFlashAttribute("message",
messages.getMessage("userDeactivate", null, locale));
                redirectAttributes.addFlashAttribute("messageType", "success");
                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("userDeactivate.msg", null, locale));

                return "redirect:/dashboard";
        }

       @PostMapping("/profile")
    public String saveProfile(@ModelAttribute User user, Locale locale, final
RedirectAttributes redirectAttributes) {

                try {

                        User persistedUser = userRepository.findOne(user.getId());
                        BeanUtils.copyProperties(user, persistedUser,
                                        "id", "username", "role", "passwordHash", "state",
"results", "friends");

                        if (!user.getPassword().isEmpty() && !"------
".equals(user.getPassword())
                                        &&
user.getPassword().equals(user.getPasswordConfirmation()))
                                persistedUser.setPasswordHash(new
BCryptPasswordEncoder().encode(user.getPassword()));

                        userRepository.save(persistedUser);

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("profileUpdated", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "success");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("profileUpdated", null, locale));

                 if (logger.isDebugEnabled())
                        logger.debug("Se actualizan los datos del perfil del usuario Id: "
+ user.getId());

                } catch (Exception ex) {

                        logger.error("Se produjo un error mientras se guardaba el perfil
de usuario", ex);

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("profileUpdate.error", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
```

```
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("profileUpdate.error.msg", null, locale));
                }

        return "redirect:/dashboard";
    }

        @GetMapping("/userSummary")
    public String userSummary(@PageableDefault(size = Constants.PAGE_SIZE, page = 0)
                        @SortDefault(sort = "publishDate", direction =
Sort.Direction.DESC)
                    Pageable pageable, @RequestParam String id, Locale locale, final
RedirectAttributes redirectAttributes, Model model) {

                User summaryUser = userRepository.findOne(id);

                Authentication authentication = CMUtils.getAuthentication();
                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                if (summaryUser.getId().equals(user.getId()) ||
!summaryUser.getPrivated()) {

                        Page<Result> paginatedResults =
resultRepository.findByUser(summaryUser, pageable);

                        model.addAttribute("user", user);
                        model.addAttribute("summaryUser", summaryUser);
                        model.addAttribute("paginatedResults", paginatedResults);

                 if (logger.isDebugEnabled())
                        logger.debug("Se accede al resumen del usuario con identificador "
+ summaryUser.getId());

                } else {

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("privateUser", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("privateUser", null, locale));

                        return "redirect:/dashboard";
                }

        return "userSummary";
    }

        @GetMapping("/addFriend")
    public String addFriend(@PageableDefault(size = Constants.PAGE_SIZE, page = 0)
                        @SortDefault(sort = "publishDate", direction =
Sort.Direction.DESC)
                    Pageable pageable, @RequestParam("id") String friendId, Locale
locale, final RedirectAttributes redirectAttributes, Model model) {

                try {

                        User friend = userRepository.findOne(friendId);

                        if (!friend.getPrivated()) {

                                Authentication authentication =
CMUtils.getAuthentication();
                                User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                                if (user.getFriends() == null) user.setFriends(new
ArrayList<>());

                                if (!user.getFriends().contains(friend))
                                        user.getFriends().add(friend);
                                userRepository.save(user);

                                Page<Result> paginatedResults =
resultRepository.findByUser(friend, pageable);

                                model.addAttribute("user", user);
                                model.addAttribute("summaryUser", friend);
                                model.addAttribute("paginatedResults", paginatedResults);
```

```
                                model.addAttribute("message",
messages.getMessage("newFriend", null, locale));
                                model.addAttribute("messageType", "success");
                                model.addAttribute("messageDescription",
messages.getMessage("newFriend.msg", null, locale));

                        } else {

                                redirectAttributes.addFlashAttribute("message",
messages.getMessage("privateUser", null, locale));
                                redirectAttributes.addFlashAttribute("messageType",
"error");
                                redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("newFriend.privated", null, locale));

                                return "redirect:/dashboard";
                        }

                } catch (Throwable ex) {

                        logger.error("Se produjo un error mientras se guardaba la
información", ex);

                        model.addAttribute("message",
messages.getMessage("operationError", null, locale));
                        model.addAttribute("messageType", "error");
                        model.addAttribute("messageDescription",
messages.getMessage("operationError.msg", null, locale));
                }

                return "userSummary";
        }

        @GetMapping("/removeFriend")
    public String removeFriend(@RequestParam("id") String friendId, Locale locale, final
RedirectAttributes redirectAttributes) {

                try {

                        User friend = userRepository.findOne(friendId);

                        Authentication authentication = CMUtils.getAuthentication();
                        User user =
userRepository.findByUsernameAndState(authentication.getName(), State.ACTIVE).get();

                        if (user.getFriends() == null) user.setFriends(new ArrayList<>());

                        user.getFriends().remove(friend);
                        userRepository.save(user);

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("friendDeleted", new String[] {friend.getName()}, locale));
                        redirectAttributes.addFlashAttribute("messageType", "success");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("friendDeleted.msg", null, locale));

                } catch (Throwable ex) {

                        logger.error("Se produjo un error mientras se guardaba la
información", ex);

                        redirectAttributes.addFlashAttribute("message",
messages.getMessage("operationError", null, locale));
                        redirectAttributes.addFlashAttribute("messageType", "error");
                        redirectAttributes.addFlashAttribute("messageDescription",
messages.getMessage("operationError.msg", null, locale));
                }

                return "redirect:/profile";
        }

}
```

13.2.2.6 FileUploadController.java

```
@Controller
public class FileUploadController {

    private final StorageService storageService;

    @Autowired
    public FileUploadController(StorageService storageService) {
        this.storageService = storageService;
    }

    @GetMapping("/files/{filename:.+}")
    @ResponseBody
    public ResponseEntity<Resource> serveFile(@PathVariable String filename) throws
IOException {

        Resource file = storageService.loadAsResource(filename);
        return ResponseEntity.ok()
                    .header(HttpHeaders.CONTENT_DISPOSITION,
                "attachment; filename=\"" + file.getFilename() + "\"")
                    .body(file);
    }

    @ExceptionHandler(StorageFileNotFoundException.class)
    public ResponseEntity<?> handleStorageFileNotFound(StorageFileNotFoundException exc)
{
        return ResponseEntity.notFound().build();
    }

    @ExceptionHandler(IOException.class)
    public ResponseEntity<?> handleIOException(IOException exc) {
        return ResponseEntity.notFound().build();
    }

}
```

13.2.3 es.uniovi.challengememory.exception

13.2.3.1 StorageException.java

```
public class StorageException extends RuntimeException {

        private static final long serialVersionUID = -8746361279779875464L;

        public StorageException(String message) {
         super(message);
    }

    public StorageException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

13.2.3.2 StorageFileNotFoundException.java

```
package es.uniovi.challengememory.exception;

public class StorageFileNotFoundException extends StorageException {

        private static final long serialVersionUID = -2930574935873104375L;
```

```
        public StorageFileNotFoundException(String message) {
            super(message);
    }

    public StorageFileNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

13.2.4 es.uniovi.challengememory.model

13.2.4.1 Comment.java

```
@Getter
@Setter
@ToString
@Builder
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "comments")
public class Comment implements Comparable<Comment> {

        @NotNull private Date date;
        @NotNull private String text;
        @NotNull @DBRef(lazy=true) private User user;


        public int compareTo(Comment o) {
                return this.date.compareTo(o.getDate());
        }
}
```

13.2.4.2 Competition.java

```
@Data
@Builder
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "competitions")
public class Competition {

        @Id private String id;
        @TextIndexed(weight=2) @NotNull private String name;
        @NotNull private CompetitionType type;

        @TextScore Float score;

        public static enum CompetitionType {

                SWIMMING("SWIMMING"),
                RUNNING("RUNNING"),
                CYCLING("CYCLING"),
                TRIATHLON("TRIATHLON");

            private final String text;

            private CompetitionType(final String text) {
                this.text = text;
            }

            @Override
            public String toString() {
                return text;
            }
        }

}
```

13.2.4.3 CompetitionEdition.java

```
@Data
@Builder
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "editions")
public class CompetitionEdition {

	@Id private String id;
	@NotNull @DBRef(lazy=true) private Competition competition;
	@TextIndexed(weight=2) @NotNull private String editionName;
	@NotNull @DateTimeFormat(pattern = "dd/MM/yyyy")  private Date date;
	@NotNull private Integer distance;
	@TextIndexed private String location;

	@TextScore Float score;

}
```

13.2.4.4 File.java

```
@Getter
@Setter
@ToString
@Builder
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "files")
public class File {

	@NotNull private FileType type;
	@NotNull private String path;


	public static enum FileType {

		IMAGE("IMAGE"),
		DOC("DOC");

	    private final String text;

	    private FileType(final String text) {
	        this.text = text;
	    }

	    @Override
	    public String toString() {
	        return text;
	    }
	}
}
```

13.2.4.5 Like.java

```
@Getter
@Setter
@ToString
@Builder
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "likes")
public class Like implements Comparable<Like> {

	@NotNull private Date date;
```

```
	@NotNull @DBRef(lazy=true) private User user;


	public int compareTo(Like o) {
		return this.date.compareTo(o.getDate());
	}
}
```

13.2.4.6 Result.java

```
@Getter
@Setter
@ToString
@Builder
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "results")
public class Result implements Comparable<Result> {

	@Id private String id;
	@NotNull @DBRef(lazy=true) private CompetitionEdition edition;
	@NotNull private Long time;
	@NotNull private Integer position;
	private Integer categoryPosition;
	private String description;
	private String chronicle;
	private String equipment;
	private String stravaLink;
	private File classification;
	@NotNull private Date publishDate;
	@NotNull @DBRef(lazy=true) private User user;

	private List<Like> likes;
	private List<Comment> comments;
	private List<Warning> warnings;
	private List<File> images;

	@Transient
	public File getFirstImage() {
		return (this.images != null && this.images.size() > 0) ?
this.images.get(0) : null;
	}

	@Transient
	public Integer getLikesCount() {
		return this.likes != null ? this.likes.size() : 0;
	}

	@Transient
	public Boolean likeThis(User user) {

		if (user != null && this.likes != null)
			for (Like like : this.likes)
				if (like.getUser().equals(user))
					return true;

		return false;
	}

	@Transient
	public String getUsersLikeThis() {

		String users = "";
		if (this.likes != null)
			users = this.likes.stream().map(l ->
l.getUser().getName()).collect(Collectors.joining(", "));

		return users;
	}

	@Transient
	public Integer getCommentsCount() {
		return this.comments != null ? this.comments.size() : 0;
```

```
	}

	@Transient
	public Boolean commentThis(User user) {

		if (user != null && this.comments != null)
			for (Comment comment : this.comments)
				if (comment.getUser().equals(user))
					return true;

		return false;
	}

	@Transient
	public String getUsersCommentThis() {

		String users = "";
		if (this.comments != null)
			users = this.comments.stream().map(l ->
l.getUser().getName()).collect(Collectors.joining(", "));

		return users;
	}

	@Transient
	public String getFormatTime() {

		if (this.time != null) {

			Period period = new Duration(this.time).toPeriod();
			PeriodFormatter formatedTime = new PeriodFormatterBuilder()
					.printZeroAlways()
					.minimumPrintedDigits(2)
					.appendHours()
					.appendSeparator(":")
					.appendMinutes()
					.appendSeparator(":")
					.appendSeconds()
				.toFormatter();

			return formatedTime.print(period);

		} else {

			return null;
		}
	}

	@Transient
	public void setFormatTime(String formatTime) {

		PeriodFormatter formatedTime = new PeriodFormatterBuilder()
				.printZeroAlways()
				.appendHours()
				.appendSeparator(":")
				.appendMinutes()
				.appendSeparator(":")
				.appendSeconds()
			.toFormatter();

		this.time =
formatedTime.parsePeriod(formatTime).toStandardDuration().getMillis();
	}

	public int compareTo(Result o) {
		return this.publishDate.compareTo(o.getPublishDate());
	}
}
```

13.2.4.7 User.java

```
@Getter
```

```
@Setter
@ToString
@EqualsAndHashCode(of="id")
@Builder
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "users")
public class User implements Comparable<User> {

	@Id private String id;
	@TextIndexed @Indexed(unique=true) @NotNull private String username;
	@TextIndexed(weight=2) @NotNull private String name;
	@DateTimeFormat(pattern = "dd/MM/yyyy") private Date birthdate;
	private Sex sex;
	@TextIndexed private String location;
	private Double weight;
	private Integer height;
	private String club;
	@NotNull private String passwordHash;
	@NotNull private Role role;
	@NotNull private State state;
	@Builder.Default @NotNull private Boolean privated = false;
	@Transient private String password;
	@Transient private String passwordConfirmation;
	@Builder.Default @Transient private Boolean terms = false;

	@TextScore Float score;

	@DBRef(lazy = true) private List<Result> results;
	@DBRef(lazy = true) private List<User> friends;


	@Transient
	public boolean isFriend(User user) {
		if (this.friends != null && this.friends.contains(user))
			return true;
		else
			return false;
	}

	@Transient
	public Integer getFriendsCount() {
		return this.friends != null ? this.friends.size() : 0;
	}

	@Transient
	public List<User> getPublicAndActiveFriends() {

		List<User> publicFriends = new ArrayList<>();
		if (friends != null)
			publicFriends = friends.stream()
					.filter(f -> !f.getPrivated() &&
State.ACTIVE.equals(f.getState()))
					.collect(Collectors.toList());

		return publicFriends;
	}

	public static enum Role {
		ADMIN("ADMIN"),
		USER("USER");

	    private final String text;

	    private Role(final String text) {
	        this.text = text;
	    }

	    @Override
	    public String toString() {
	        return text;
	    }
	}

	public static enum State {
		INACTIVE("INACTIVE"),
		ACTIVE("ACTIVE");

	    private final String text;
```

```
            private State(final String text) {
                this.text = text;
            }

            @Override
            public String toString() {
                return text;
            }
        }

        public static enum Sex {
                FEMENINO("FEMENINO"),
                MASCULINO("MASCULINO");

            private final String text;

            private Sex(final String text) {
                this.text = text;
            }

            @Override
            public String toString() {
                return text;
            }
        }

        @Override
        public int compareTo(User u) {
                return this.getName().compareTo(u.getName());
        }

}
```

13.2.4.8 Warning.java

```
@Getter
@Setter
@ToString
@Builder
@AllArgsConstructor(access = AccessLevel.PACKAGE)
@NoArgsConstructor(access = AccessLevel.PACKAGE)
@Document(collection = "falseResults")
public class Warning implements Comparable<Warning> {

        @NotNull private Date date;
        @NotNull private String text;
        @NotNull @DBRef(lazy=true) private User user;


        public int compareTo(Warning o) {
                return this.date.compareTo(o.getDate());
        }
}
```

13.2.5 es.uniovi.challengememory.repository

13.2.5.1 CompetitionRepository.java

```
public interface CompetitionRepository extends MongoRepository<Competition, String> {
```

```
        public Page<Competition> findByNameContainingIgnoreCase(String q, Pageable
pageable);
        public List<Competition> findAllByName(String name);

        @Query("{$text: {$search: ?0}}")
        public Page<Competition> findByText(String text, Pageable pageable);

}
```

13.2.5.2 EditionRepository.java

```
public interface EditionRepository extends MongoRepository<CompetitionEdition, String> {

        public Page<CompetitionEdition> findByEditionNameContainingIgnoreCase(String q,
Pageable pageable);
        public CompetitionEdition findFirstByOrderByDateDesc();
        public List<CompetitionEdition> findAllByCompetitionOrderByDateDesc(String id);
        public List<CompetitionEdition> findAllByEditionName(String editionName);

        @Query("{$text: {$search: ?0}}")
        public Page<CompetitionEdition> findByText(String text, Pageable pageable);
}
```

13.2.5.3 ResultRepository.java

```
public interface ResultRepository extends MongoRepository<Result, String> {

        public Page<Result> findByUser(User users, Pageable pageable);
        public Page<Result> findByUserIn(List<User> users, Pageable pageable);
        public List<Result> findAllByEditionOrderByTime(CompetitionEdition edition);
        public List<Result> findAllByUserAndEdition(User user, CompetitionEdition
edition);
        public List<Result> findAllByUserAndEditionIn(User user, List<CompetitionEdition>
editions);

        @Query("{warnings: {$exists: true, $not: {$size: 0}}}")
        public List<Result> findResultsWithWarnings();
}
```

13.2.5.4 UserRepository.java

```
public interface UserRepository extends MongoRepository<User, String> {

        public Optional<User> findByUsernameAndState(String username, State state);

        @Query("{$text: {$search: ?0}, privated: false, state: 'ACTIVE'}")
        public Page<User> findByText(String text, Pageable pageable);

}
```

13.2.6 es.uniovi.challengememory.service

13.2.6.1 StorageService.java

```
public interface StorageService {

    void init();

    void store(MultipartFile file, String name);

    Resource loadAsResource(String filename);

}
```

13.2.7 es.uniovi.challengememory.service.impl

13.2.7.1 UserDetailsServiceImpl.java

```
@Service("userDetailsService")
public class UserDetailsServiceImpl implements UserDetailsService {

	@Autowired
	private MessageSource messageSource;

	@Autowired
	private UserRepository userRepository;

	@Transactional(readOnly=true)
	@Override
	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

		final String notFoundMessage = messageSource.getMessage("user.notFound",
null, LocaleContextHolder.getLocale());

		Optional<es.uniovi.challengememory.model.User> user =
userRepository.findByUsernameAndState(username, State.ACTIVE);
		if (user != null && user.isPresent()) {

			List<GrantedAuthority> authorities =
buildUserAuthority(user.get().getRole());
			return buildUserForAuthentication(user.get(), authorities);

		} else {
			throw new UsernameNotFoundException(notFoundMessage);
		}

	}

	private User buildUserForAuthentication(es.uniovi.challengememory.model.User
user, List<GrantedAuthority> authorities) {

		return new User(user.getUsername(), user.getPasswordHash(), authorities);
	}

	private List<GrantedAuthority> buildUserAuthority(Role role) {

		Set<GrantedAuthority> setAuths = new HashSet<GrantedAuthority>();
		setAuths.add(new SimpleGrantedAuthority(role.name()));

		return new ArrayList<GrantedAuthority>(setAuths);
	}

}
```

13.2.7.2 FileSystemStorageService.java

```
@Service
public class FileSystemStorageService implements StorageService {

    private final Path rootLocation;

    @Autowired
    public FileSystemStorageService(FileStorageProperties properties) {
        this.rootLocation = Paths.get(properties.getLocation());
    }

    @Override
    public void store(MultipartFile file, String name) {
            String filename = StringUtils.cleanPath(name);
            try {
                    if (file.isEmpty()) {
                            throw new StorageException("Failed to store empty file " +
filename);
                    }
                    if (filename.contains("..")) {
                            // This is a security check
                            throw new StorageException(
                                            "Cannot store file with relative path
outside current directory " + filename);
                    }
                    try (InputStream inputStream = file.getInputStream()) {
                            Files.copy(inputStream,
this.rootLocation.resolve(filename), StandardCopyOption.REPLACE_EXISTING);
                    }
            } catch (IOException e) {
                    throw new StorageException("Failed to store file " + filename, e);
            }
    }

    private Path load(String filename) {
        return rootLocation.resolve(filename);
    }

    @Override
    public Resource loadAsResource(String filename) {
            try {
                    Path file = load(filename);
                    Resource resource = new UrlResource(file.toUri());
                    if (resource.exists() || resource.isReadable()) {
                            return resource;
                    } else {
                            throw new StorageFileNotFoundException("Could not read
file: " + filename);

                    }
            } catch (MalformedURLException e) {
                    throw new StorageFileNotFoundException("Could not read file: " +
filename, e);
            }
    }

    @Override
    public void init() {
            try {
                    Files.createDirectories(rootLocation);
            } catch (IOException e) {
                    throw new StorageException("Could not initialize storage", e);
            }
    }
}
```

13.2.7.3 GridFSStorageService.java

```
@Service
@Primary
```

```
public class GridFSStorageService implements StorageService {

	@Autowired
    private GridFsTemplate gridFsTemplate;

	@Override
    public void store(MultipartFile file, String name) {

	String filename = StringUtils.cleanPath(name);
	try {
		DBObject metaData = new BasicDBObject();
		metaData.put("user", "alex");
		 gridFsTemplate.store(file.getInputStream(), filename,
file.getContentType(), metaData).getId().toString();

	} catch (IOException e) {
			throw new StorageException("Failed to store file " + filename, e);
		}
    }

    @Override
    public Resource loadAsResource(String filename) {

		GridFSDBFile gridFSDBFile = gridFsTemplate.findOne(new
Query(Criteria.where("filename").is(filename)));
		Resource resource = new
InputStreamResource(gridFSDBFile.getInputStream());
		if (resource.exists() || resource.isReadable()) {
			return resource;
		} else {
			throw new StorageFileNotFoundException("Could not read file: " +
filename);

		}

    }

    @Override
    public void init() {}
}
```

13.2.7.4 MailServiceImpl.java

```
@Service
public class MailServiceImpl {

	private JavaMailSender mailSender;

	public MailServiceImpl(JavaMailSender mailSender) {
		this.mailSender = mailSender;
	}

	public void prepareAndSend(String sender, List<String> recipients, String
subject, String message) throws MailException {

		MimeMessagePreparator messagePreparator = mimeMessage -> {

			MimeMessageHelper messageHelper = new
MimeMessageHelper(mimeMessage, true);
			messageHelper.setFrom(sender);
			for (String recipient : recipients) {
				messageHelper.addBcc(new InternetAddress(recipient));
			}
			messageHelper.setSubject(subject);
			messageHelper.setText(message, true);
		};

		mailSender.send(messagePreparator);

	}

}
```

13.2.8 es.uniovi.challengememory.utils

13.2.8.1 CMUtils.java

```
public class CMUtils {

	public static boolean isAuthenticated() {

		Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

		if (authentication != null && authentication.isAuthenticated()
				&& !(authentication instanceof
AnonymousAuthenticationToken))
			return true;
		else
			return false;
	}

	public static Authentication getAuthentication() {

		Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

		if (authentication != null && authentication.isAuthenticated()
				&& !(authentication instanceof
AnonymousAuthenticationToken))
			return authentication;
		else
			return null;
	}

	private CMUtils() { }
}
```

13.2.8.2 Constants.java

```
public class Constants {

	public static final int PAGE_SIZE = 10;
	public static final int SEARCH_PAGE_SIZE = 20;

	private Constants() { }
}
```