UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño y desarrollo de una aplicación para la gestión de un equipo de natación

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Gallifa Tronch, Enrique

Tutor/a: Vidal Oriola, Germán Francisco

CURSO ACADÉMICO: 2022/2023

Resumen

Con este trabajo final de grado se pretende diseñar y desarrollar un sistema que ayude a los entrenadores de equipos de natación a mejorar los resultados obtenidos en las competiciones por puntos. Este sistema pertenece al ámbito de las aplicaciones web, permitiendo su acceso desde cualquier dispositivo con un navegador web integrado. Cada vez hay más actividad en torno a las páginas web y con ello aparecen muchos *frameworks* que son de gran utilidad para desarrollarlas con facilidad.

El objetivo principal es el de implementar una primera versión usable (*Minimum Viable Product*, *MVP*) capaz de obtener la lista de los deportistas mejor capacitados para maximizar los puntos obtenidos en la competición. Esta versión permite, además de obtener esta lista, obtener una serie de información importante para los entrenadores de cara a mejorar los entrenamientos grupales de sus nadadores (ritmo ideal de entrenamiento, marcas de competición). Con estas herramientas, un entrenador puede ser capaz de hacer mejorar a su equipo para obtener un mayor rendimiento individual de cada nadador.

Para alcanzar dicho objetivo, se ha utilizado un *framework* de *JavaScript*, *NextJS*, junto a *TypeScript*, conectado a una base de datos no relacional, *MongoDB*. Este *framework* ha facilitado el uso de la arquitectura por capas, debido a su sistema de carpetas, junto a una metodología ágil derivada de *SCRUM*, realizando varios *sprints* con un objetivo bien definido a desarrollar en cada uno de ellos. Además, para poder realizar una interfaz amigable y fácil de utilizar, se ha utilizado *Tailwind CSS* para dar estilo a todas las páginas de este sistema web.

Palabras clave: entrenador, nadador, *JavaScript*, *NextJS*, *SCRUM*

Resum

Amb aquest treball final de grau es pretén dissenyar i desenvolupar un sistema que ajude als entrenadors d'equips de natació a millorar els resultats obtinguts per a les competicions per punts. Aquest sistema pertany de l'àmbit de les aplicacions web, permetent el seu accés des de qualsevol dispositiu amb un navegador web integrat. Cada vegada hi ha més activitat al voltant de les pàgines web i amb això apareixen molts *frameworks* que són de gran utilitat per desenvolupar-les amb facilitat.

L'objectiu principal és el d'implementar una primera versió usable (*Minimum Viable Product*, *MVP*) capaç d'obtindre la llista dels esportistes més ben capacitats per a maximitzar els punts obtinguts en la competició. Aquesta versió permet, a més d'obtindre aqueta llista, obtindre una sèrie d'informació important per als entrenadors de cara a millorar els entrenaments grupals dels seus nadadors (ritme ideal d'entrenament, marques de competició). Amb aquestes eines, un entrenador pot ser capaç de fer millorar al seu equip per a obtindre un major rendiment individual de cada nadador.

Per a aconseguir aquest objectiu, s'ha utilitzat un *framework* de *JavaScript*, *NextJS*, juntament amb *TypeScript*, connectat a una base de dades no relacional, *MongoDB*. Aquest *framework* ha facilitat l'ús de l'arquitectura per capes, a causa del seu sistema de carpetes, al costat d'una metodologia àgil derivada de *SCRUM*, realitzant diversos *sprints* amb un objectiu ben definit a desenvolupar en cadascun d'ells. A més, per a poder realitzar una interfície amigable i fàcil d'utilitzar, s'ha emprat *Tailwind CSS* per a donar estil a totes les pàgines d'aquest sistema web.

Paraules clau: entrenador, nadador, *JavaScript*, *NextJS*, *SCRUM*

Abstract

With this final degree project, we intend to design and develop a system to help swimming coaches to improve the results in championships. This system belongs to the area of web-applications, allowing its access from any device with a web browser. There is more and more activity around web pages and with it, a lot of frameworks we appeared to develop them easily.

The main objective is to implement a first usable version (*Minimum Viable Product*, *MVP*) capable of obtaining the best trained swimmers to maximize the number of points obtained in the competition. This version allows, besides getting this list, to get a series of important information for coaches in order to improve group training (ideal training pace, championship marks). With this tool, a coach may be able to improve their team in order to get a better individual performance from every single swimmer.

In order to obtain such objectives, it has been used a *JavaScript framework*, *NextJS*, together with *TypeScript*, connected to a non-relational database, *MongoDB*. This framework has eased the use of a layered architecture, due to its folder system, besides the agile methodology derived from *SCRUM*, carrying out several sprints with a well-defined objective to develop in each sprint. Moreover, in order to prepare a friendly and easy-going interface, *Tailwind CSS* has been used to format every page of this web system.

Keywords: coach, swimmer, *JavaScript*, *NextJS*, *SCRUM*

Índice general

Índice de ilustraciones

Índice de figuras

Índice de tablas

1. Introducción

1.1 Contexto y motivación

Desde hace ya varios años la natación se ha convertido en un deporte muy practicado y reconocido a nivel mundial, pero el acceso al mundo de la competición genera mucha inquietud por lo sacrificado que es y lo difícil que resulta conseguir alcanzar los objetivos que se propone la persona que quiere practicar la disciplina.

La natación es vista como un deporte individual, pues mayoritariamente nada una persona sola contra hasta otros 7 deportistas (en piscina olímpica de 8 calles), pero lo cierto es que es un deporte de equipo. Un atleta olímpico no podría llegar solo a los Juegos Olímpicos si no tuviera un equipo detrás que le ayudase a entrenar y competir. Este equipo va desde un equipo técnico de uno o varios entrenadores hasta otros compañeros que practican el deporte a su lado.

Además de estas competiciones en las que destaca el deportista como individuo, también existen competiciones por equipos, en las que los deportistas compiten por su equipo contra otros deportistas que representan a sus respectivos equipos y en función de la posición consiguen más o menos puntos.

Personalmente, este tipo de competición es la que más gusta como deportista, ya que se vive con mucha emoción toda la competición y motiva mucho a los componentes de los equipos. Esta competición se vive con mucha intensidad y dependiendo de la posición en la tabla, el equipo asciende de categoría, se mantiene o desciende, lo cual ayuda a incrementar la emoción de esta competición.

1.2 Objetivos

Como en la mayoría de las competiciones deportivas, las posibilidades de conseguir más puntos en las distintas pruebas varían en función de las personas que participen.

Este trabajo pretende abordar las dificultades de gestionar la parte deportiva de un equipo de natación focalizado en las competiciones. El principal usuario de la aplicación será el entrenador del equipo, el cual tendrá acceso a información relevante de los deportistas en su equipo (como nombre, fecha de nacimiento y tablas de tiempos), además de información sobre entrenamientos y las próximas competiciones para maximizar la puntuación de su equipo en un campeonato importante.

La principal dificultad reside en la obtención de los datos (acceso a la base de datos en la que se ubican los tiempos oficiales) y la realización del cálculo de la mejor formación añadiendo parámetros.

Para el correcto desarrollo y funcionamiento de la aplicación se deben cumplir una serie de requisitos a nivel técnico:

- Implementar un *backend* con el que la aplicación ejecute la mayoría de su lógica, como conectarse a una base de datos o realizar los cálculos necesarios para la suma de puntos.
- Implementar un *frontend* amigable con el que los equipos técnicos se sientan cómodos para utilizar la aplicación.
- Definir y realizar un MVP (*Minimum Viable Product*), con el que se puedan realizar gestiones internas de equipo y realizar el cálculo teórico de puntos con una formación inicial dada.

1.3 Impacto esperado

Previsiblemente el uso de la aplicación ayudará a los equipos a conseguir una mayor cantidad de puntos para los campeonatos colectivos.

Su uso por parte de todos los equipos propiciaría llevar al máximo exponente la competición.

1.4 Metodología

La aplicación resultante deberá atravesar una serie de procesos a lo largo del ciclo de desarrollo que permitirán que esté disponible el producto mínimo viable en un espacio corto de tiempo.

Se ha decidido utilizar una metodología basada en SCRUM, recogiendo todos los requisitos del sistema al principio y realizando un estudio del alcance, para posteriormente trabajar por iteraciones de 3 semanas, desarrollando los requisitos en orden de prioridad.

Hay conceptos de SCRUM que no se van a aplicar, las reuniones de retrospectiva, ya que el equipo de desarrollo es de una sola persona, es más complicado realizar una reunión para cambiar la forma de trabajar.

1.4.1 Ciclo de vida de desarrollo

Comprender estos procesos es importante para elegir la metodología a utilizar, sirviendo como guía para el completo y correcto desarrollo del producto.

- **Planificación:** localizar los requisitos necesarios del sistema para limitar el producto final.
- **Análisis:** una vez recopilados los requisitos, priorizarlos y clasificarlos para decidir el orden de implementación.

Estas dos fases se realizarán una única vez al principio del desarrollo, dado el contexto de desarrollo (equipo de una única persona) para simplificar la toma de decisiones y definir el alcance del proyecto desde el principio (esta parte choca

directamente con uno de los principios de SCRUM de "estar abierto a cambios", pero es una forma sencilla de evitar cambios muy grandes en los requisitos y la planificación iniciales).

Las siguientes fases se repetirán con cada iteración del desarrollo (*sprints* de 3 semanas de duración cada uno):

- **Diseño:** documentar las características a desarrollar durante el próximo *sprint* (realización de *mockups* y diseño de pruebas manuales/automáticas para dar por finalizado el desarrollo de la característica).
- **Codificación:** basándose en el diseño, implementar los requisitos.
- **Testing:** ejecutar las pruebas manuales diseñadas e implementar las pruebas automáticas para confirmar la ausencia de errores y *bugs* introducidos en la codificación.
- **Revisión:** verificar que el producto en desarrollo es el especificado.

Como se puede observar, durante el desarrollo de la aplicación se mezclan ideas de metodología tradicional, realizando una única elicitación de requisitos (se parte con un alcance establecido), con conceptos de la metodología ágil SCRUM, realizando el desarrollo por *sprints* centrados en desarrollar unas características concretas.

1.5 Estructura de la memoria

La presente memoria tiene una estructura basada en capítulos, en los cuales se va a documentar todo el proceso de desarrollo de este trabajo.

1. Introducción
2. Estado del arte
3. Análisis y especificación de requisitos
4. Arquitectura y diseño del sistema software
5. Implementación
6. Pruebas
7. Conclusión y trabajo futuro

2. Estado del arte

2.1 Contexto tecnológico

En el entorno de la natación regional a nivel de la Comunidad Valenciana hay pocas aplicaciones que puedan ser útiles para el día a día de un equipo que no busca principalmente el éxito deportivo, sino proponer una actividad deportiva divertida que es saludable y fomentar el ambiente competitivo de una manera sana.

La principal aplicación utilizada está incluida en la página de la Federación Valenciana de Natación (Figura 1). En ella se realizan la gran mayoría de gestiones referentes a la federación (inscripción de deportistas en competiciones, ver listas de inscripciones, dar de baja en competiciones). Esta aplicación, al ser la oficial de la Federación Valenciana de Natación, está conectada directamente con la base de datos de Leverade, que contiene toda la información relacionada con los deportistas y sus tiempos de competición.

La aplicación está diseñada para realizar las gestiones necesarias para la federación (inscripción y generación de listas de competiciones, cálculo de marcas de deportistas), por lo que no tiene ningún sistema de cómputo de puntuación antes de realizarse la competición, sino que se realiza mediante una aplicación externa durante el transcurso de las propias competiciones.

Figura 1 - Federación Valenciana de Natación

Otra aplicación bastante conocida es “My Nedalia Swimmer” (Figura 2). La principal finalidad de esta aplicación es la de consultar la información que publica el entrenador sobre cada nadador en su equipo (marcas, tests, campeonatos, mejores marcas, tiempos mínimos de acceso a campeonato), es decir, una herramienta de consulta que no está pensada para ser utilizada durante una competición ni para realizar cálculos de tiempo de entrenamiento.

Esta herramienta provee al deportista y al entrenador de varias gráficas que muestran la evolución del deportista en los tiempos de las distintas pruebas en competición, mostrando una función creciente o decreciente en el tiempo, dependiendo de la mejoría del deportista.

Esta aplicación está conectada con "My Nedalia", que es la aplicación pensada para entrenadores, donde se realizan los entrenamientos, registran los tiempos de competición y se ve la información de tus deportistas.

Figura 2 - Icono aplicación Nedalia

Por otro lado, "*TrainingPeaks*" (Figura 3) es una aplicación pensada tanto para deportistas como para entrenadores. Su principal objetivo son los deportes de triatlón, ciclismo, *running*, natación, aunque también tiene secciones de remo, deportes de invierno y ejercicios de peso.

La forma de trabajar de esta aplicación es mediante los entrenamientos que realizan los entrenadores y estos ponen a la venta su servicio, de forma que los deportistas tienen a alguien pendiente de su progreso durante las semanas que dura el plan de entrenamiento escogido.

La principal ventaja que tiene esta aplicación es su conectividad con varios dispositivos *Smart-watch*, de forma que se pueden obtener datos de entrenamiento en tiempo real para, posteriormente, enviar los datos a *TrainingPeaks* para tener un seguimiento del progreso obtenido.

La principal desventaja de esta aplicación reside en que no utiliza el concepto de club, por lo que el deportista entrena de forma individual, sin tener un equipo con el que competir a la vez que entrena. Cabe destacar, que al no tener el equipo, no tiene capacidad para gestionar competiciones ni sistema de inscripción.

Figura 3 - Icono aplicación TrainingPeaks

Por último, *"Commit Swimming"* (Figura 4) es una aplicación enfocada a los entrenamientos y a la gestión económica y de asistencia a los entrenamientos de los deportistas. Así mismo, esta aplicación provee la posibilidad de tener una

página web con dominio propio para el club, y que la información de esta esté conectada a la base de datos de entrenamientos.

"*Commit Swimming*" tiene integrado un chat entre los deportistas y los entrenadores, y a su vez, tiene los entrenamientos y los resultados de estos en el mismo sitio. De esta forma los deportistas no tienen que utilizar diferentes aplicaciones para comunicarse y ver al mismo tiempo la información de los entrenamientos.

Dentro de la misma aplicación hay un calendario en el que aparecen las próximas competiciones, además de poder añadir eventos del equipo, como sesiones de entrenamiento especiales o eventos del club. Sin embargo, estos eventos del calendario no están enlazados con la inscripción a este directamente y se tendrían que hacer desde la aplicación oficial. Además, contiene herramientas y una gran variedad de gráficas para seguir el progreso del rendimiento y la asistencia de los deportistas del club.

Figura 4 - Icono aplicación CommitSwimming

2.2 Análisis de las aplicaciones

Una vez analizadas las aplicaciones relacionadas con el mundo de la gestión de equipo en la natación, conseguimos ver varias características que comparten, y varias que las hacen destacar sobre las otras. A continuación, se muestra una tabla con esas características importantes, que servirá como fuente para obtener características a desarrollar en la primera versión del aplicativo:

Característica localizada	Federación Valenciana de Natación	Nedalia	TrainingPeaks	CommitSwimming
Calendario de competiciones	✅	❌	❌	✅
Ver lista de deportistas	✅	✅	✅	✅
Ver marcas de deportistas	✅	✅	✅	✅
Inscripción en competiciones	✅	❌	❌	❌
Dar de baja deportista en una competición	✅	❌	❌	❌
Obtener cálculo teórico de puntos	❌	❌	❌	❌
Pruebas por nadador en competición	❓	❌	❌	❌
Ver clasificación de la competición	✅	❌	❌	❌
Exportar tiempos de competición de deportista	❓	❌	❌	❓
Exportar tiempos para entrenar de deportista	❌	❓	❓	❌
Ver lista de “mis inscritos” en una competición	✅	❌	❌	❌
Ver lista de inscritos en una competición	✅	❌	❌	❌

Tabla 1 - Comparativa de características entre aplicaciones de interés

La inclusión de las características de la Tabla 1 *se ven indicadas mediante los siguientes símbolos:*

1. 🤔: incluida con ciertos matices
2. ❌: característica no incluida en la primera versión del aplicativo
3. ✅: característica incluida en la primera versión del aplicativo

2.3 Propuesta de mejora

Las aplicaciones mencionadas anteriormente se centran mayoritariamente en el desarrollo deportivo del deportista como individuo, necesitando en la mayoría de los casos la figura de un entrenador que le ayude para registrar los datos y consultarlos posteriormente. Estas aplicaciones ofrecen muchos datos sobre la mejora de rendimiento y datos de entrenamiento, centrados mayoritariamente en la práctica del deporte en sí mismo, centrado en el deportista, pero no ofrece ninguna funcionalidad que ayude a los entrenadores para preparar competiciones específicas y la planificación para un objetivo concreto.

La idea de mejora consiste en implementar un sistema que ayude a los entrenadores a planificar las competiciones y, a la vez, muestre unas marcas de entrenamiento en función del ritmo que se quiera entrenar (ritmo aeróbico ligero, ritmo aeróbico medio o ritmo aeróbico intenso) para que el nadador tenga una mejor idea sobre el ritmo de entrenamiento que se desea seguir. Además, el principal objetivo consiste en la mejora de la clasificación en la competición, por lo que se potencia las características de la competición, desarrollando un sistema para las inscripciones y el cómputo de puntos en las competiciones.

3. Análisis del problema

En el presente capítulo se discute la realidad del problema para poder plantear una solución viable para este y definir un camino para el desarrollo.

3.1 Modelo de Dominio

En el apartado 3.1 vamos a presentar el modelo de dominio del problema, para tener una imagen más clara de cuáles son los conceptos importantes y cómo se relacionan entre ellos en el sistema.

Para conocer los términos propios del sistema y cómo se relacionan, se presenta un diagrama de clases utilizando el lenguaje de modelado UML [1].

Los nombres de las clases están en inglés porque durante el proceso de desarrollo se codificará en inglés para una mejor comprensión, aunque se escribirá el *front-end* que el cliente verá en su navegador en castellano.

- Clase *USER*:
    - Esta clase hace referencia a una persona que interactúa con la aplicación, puede ser un deportista o un entrenador.
    - Tiene los siguientes atributos:
        - *name*: nombre del usuario;
        - *birthYear*: año de nacimiento del usuario;
        - *genre*: género del usuario.
- Clase *TRAINER*:
    - Es una especialización de la clase *USER*, representa a un entrenador de un equipo.
    - Además de los atributos de la clase *USER*, tiene los siguientes atributos:
        - *userName*: representa el nombre de usuario para acceder a la funcionalidad de la aplicación;
        - *password*: contraseña del usuario para acceder a la aplicación;
        - *club*: equipo al que pertenece el usuario.
- Clase *SWIMMER*:
    - Es una especialización de la clase *USER*, representa a un nadador.
    - Además de los atributos de la clase *USER*, tiene los siguientes atributos:
        - *club*: equipo al que pertenece el nadador.
- Clase *CLUB*:

    - Esta clase representa a un equipo; es la conexión entre entrenadores y deportistas.
    - Los atributos de esta clase son los siguientes:
        - *name*: nombre del equipo;
        - *foundationYear*: año de fundación del equipo;
        - *location*: ciudad o pueblo donde el equipo tiene su sede.
- Clase *MARK*:
    - Esta clase representa el tiempo realizado por un nadador en una prueba determinada.
    - Los atributos de esta clase son los siguientes:
        - *Swimmer*: nadador al que pertenece el tiempo;
        - *Style*: estilo de la prueba (mariposa, espalda, braza, crol o estilos);
        - *Distance*: distancia de la prueba (varía entre los 50 y 1500 metros dependiendo del estilo);
        - *Min*: valor de minutos que se ha tardado en nadar la prueba;
        - *Sec*: valor de segundos que se ha tardado en nadar la prueba;
        - *Mil*: valor de milisegundos que se ha tardado en nadar la prueba.
- Clase *CHAMPIONSHIP*:
    - Representa una competición, con sus sesiones y las distintas pruebas que se nadan durante el transcurso de esta.
    - Los atributos de esta clase son los siguientes:
        - *name*: nombre de la competición;
        - *location*: localización de la competición;
        - *startDate*: fecha en la que empieza la competición;
        - *endDate*: fecha en la que termina la competición.
- Clase *EVENT*:
    - Esta clase representa una prueba en la competición. Sirve para diferenciar cada prueba individualmente en cada competición.
    - Los atributos que contiene son:
        - *championship*: nombre de la competición;
        - *distance*: distancia que dura la prueba;
        - *stroke*: estilo al que se nada la prueba (mariposa, espalda, braza, crol o estilos).

- Clase *REGISTERED*:
    - Esta clase representa la inscripción de un nadador para nadar una prueba de una competición.
    - Los atributos de esta clase son:
        - *event*: hace referencia a la prueba y competición en la que se ha inscrito al nadador;
        - *swimmer*: nadador inscrito en la prueba.

El diagrama de clases referenciado en la Ilustración 1 muestra la estructura que tiene la aplicación para relacionar los distintos objetos dentro de ella. En él se puede apreciar que hay una relación de herencia entre la clase *USER* (la superclase) y las clases *TRAINER* y *SWIMMER* (clases hijas), las cuales comparten los atributos "name", "birthYear" y "genre". A su vez, estas dos clases están relacionadas con la clase *CLUB*, que contiene como mínimo un *TRAINER* y un *SWIMMER*, además de sus atributos "name", "foundationYear" y "location". La clase *TRAINER* tiene, además, "userName", "password" y "birthYear" como atributos.

Del mismo modo, un *CLUB* puede participar en varias *CHAMPIONSHIP*. Por último, un *SWIMMER* tiene varios *MARK* (tiempo realizado en una prueba de natación), y cada *MARK* relaciona a un nadador con un *EVENT*. Un *SWIMMER* puede tener varios *MARK* que hacen referencia al mismo *EVENT*, pero se tiene en cuenta el último *MARK* registrado, puesto que es la marca que tiene más valor (es un valor más cercano a la realidad).

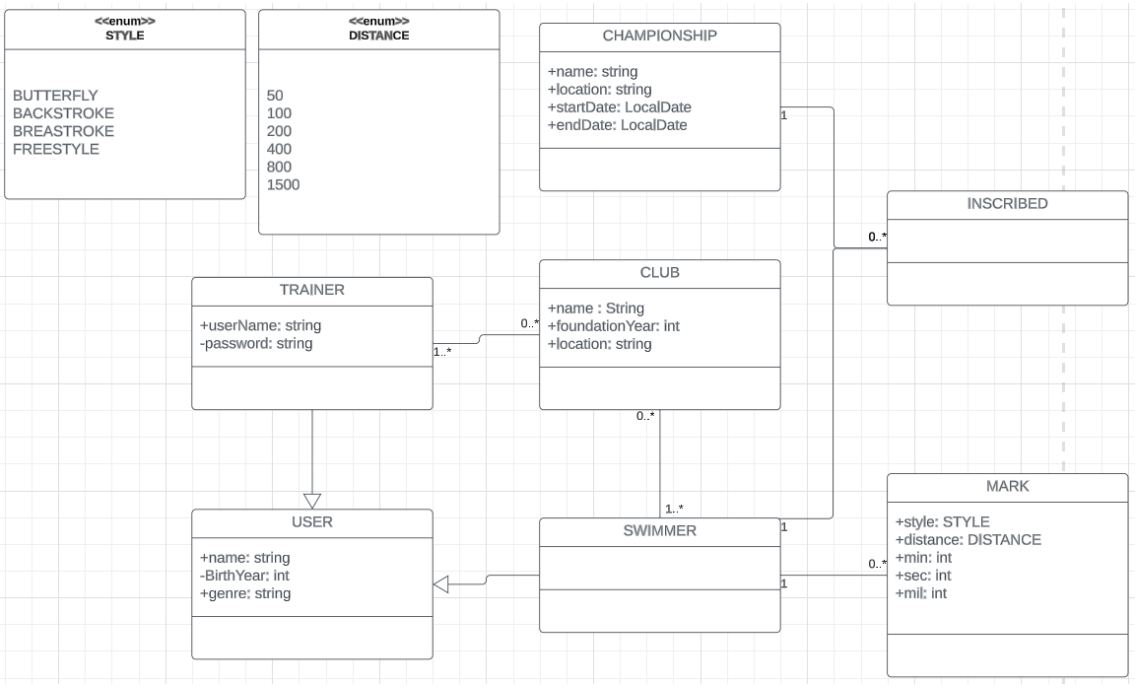


Ilustración 1 - Modelo de Dominio

3.2 Especificación de requisitos

Para llevar a cabo la especificación de requisitos se seguirá parcialmente la norma establecida por el estándar IEEE Std 830-1998 [2], ampliamente conocido por la facilidad que ofrece a la hora de gestionar requisitos.

3.2.1 Introducción

En esta sección se proporcionará una introducción a la Especificación de Requisitos. Consta de varias subsecciones para dar contexto sobre la Especificación de Requisitos realizada.

3.2.1.2 Propósito

El objetivo principal de esta sección es la de definir los requisitos que deben reflejarse en el *MVP* del sistema, agilizando y guiando el desarrollo de este.

3.2.1.2 Ámbito

El ámbito es el de las webApp, aplicaciones web que pueden funcionar a su misma vez como páginas web.

Las principales funciones del sistema residen en el cómputo de puntos en las diferentes pruebas realizadas durante un campeonato de natación. Además, ayudará a los entrenadores para definir los tiempos de entrenamiento de los deportistas en función del ritmo aeróbico que se quiera emplear. El sistema tendrá un registro propio de los deportistas, equipos y entrenadores incluidos en las competiciones, separando por divisiones a estos equipos.

El sistema no estará conectado en una primera versión con los tiempos y nombres reales de los nadadores. Tampoco asegurará que los resultados obtenidos por el equipo dada la configuración ideal teórica den como resultado final el mejor resultado real, ya que hay distintos factores externos que el sistema no mide.

El principal beneficio del aplicativo es el de la mejora potencial de resultados obtenidos en las competiciones deportivas. Se espera que los equipos que utilicen la aplicación mejoren la cantidad de puntos obtenidos de manera teórica, consiguiendo mejorar sus resultados reales al final de la competición.

3.2.1.3 Definiciones, Acrónimos y Abreviaturas

Los términos relacionados con la ontología del sistema se enumeran y explican en la Tabla 2.

Término	Descripción
Nadador	Persona que nada
Club	Equipo, está formado por uno o más entrenadores y varios nadadores
Entrenador	Persona encargada de gestionar el equipo. Persona a la que principalmente va dirigido el sistema
Prueba	Conjunto de distancia (en metros) y estilo (mariposa, espalda, braza, crol o estilos) único que delimita lo que se va a nadar.
Tiempo	El tiempo que tarda un nadador en realizar una prueba específica

Tabla 2 - Glosario de términos ontológicos del sistema

3.2.1.4 Resumen

El apartado 3.2 consta de 3 subsecciones. En la primera se ha realizado una introducción al mismo, proporcionando una visión general de la especificación de requisitos del sistema.

En la segunda subsección, se realiza una descripción general del sistema y de los factores externos a este que le afectan, con el fin de conocer las principales funciones que este debe realizar, los datos asociados y los factores, restricciones y supuestos que afectan al desarrollo, sin detallar excesivamente.

Para terminar este apartado, en la tercera subsección se describen detalladamente los requisitos que debe satisfacer el sistema.

3.2.2 Descripción general

En esta sección describimos los factores que afectan al producto y a sus requisitos. Describimos el contexto de los requisitos.

3.2.2.1 Perspectiva del producto

El sistema software a desarrollar se diseñará para trabajar en un entorno WEB, de esta forma será fácilmente accesible por sus usuarios.

El aplicativo es un sistema independiente a cualquier otro sistema al que pueda parecerse. Sin embargo, el objetivo a largo plazo es que forme parte del sistema que utiliza la Federación de Natación a nivel autonómico e integrarlo con la aplicación de Leverade para poder ofrecer funcionalidad relacionada con los entrenamientos a los entrenadores.

3.2.2.2 Funciones del producto

Para conocer la funcionalidad del sistema se presenta un Diagrama de Casos de Uso, con el que podemos tener una imagen del actor y su interacción con el sistema.

En la Ilustración 2 se puede apreciar que el único actor es *TRAINER* y cuenta con una gran variedad de funciones. Es el tipo de usuario hacia el que va destinado este sistema.

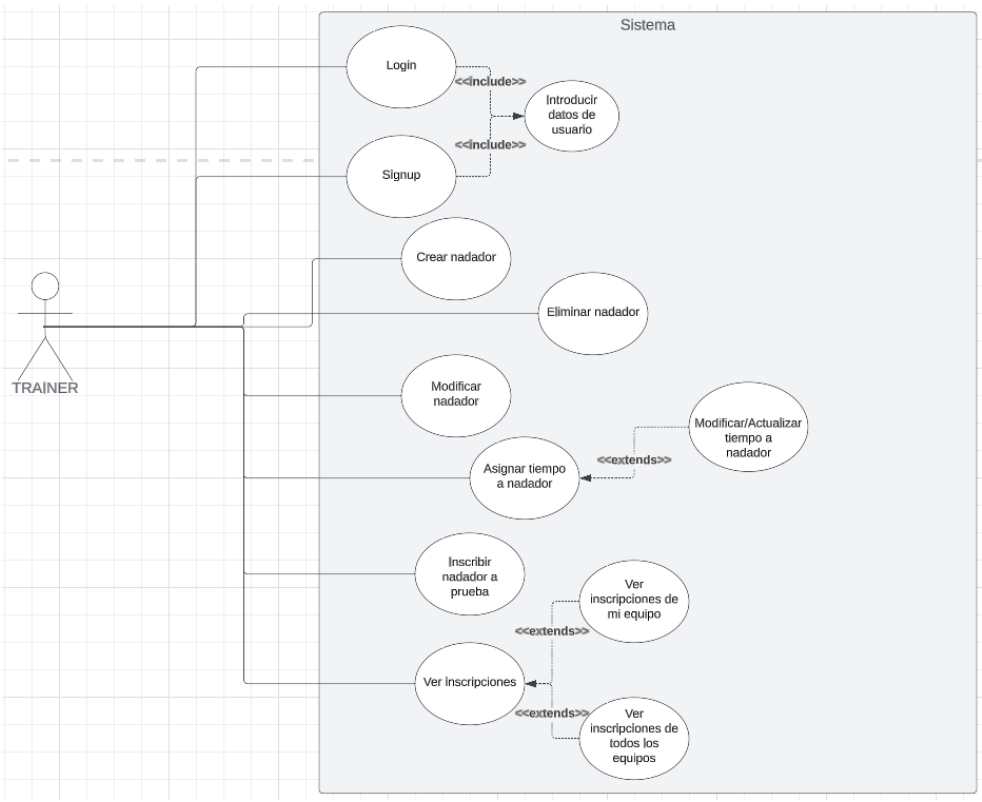


Ilustración 2 - Diagrama de casos de uso

3.2.2.3 Características de los usuarios

En la Tabla 3 mostrada a continuación, se puede ver las características esperadas del tipo de usuario al que va destinado la aplicación.

Tipo de usuario	TRAINER
Formación	Entrenador del equipo
Actividades	- Gestiona a los deportistas de su equipo - Tiene un sistema de registro de los resultados de sus deportistas - Inscribe a los deportistas en pruebas que posteriormente nadarán
Conocimiento tecnológico	Nivel básico

Tabla 3 - Tipo de usuario

3.2.2.4 Restricciones

El sistema presenta una serie de restricciones que limitan su funcionalidad y sus tiempos de desarrollo.

Limitaciones por uso:

- Se necesita conexión a Internet.
- Se accederá al sistema a través de un navegador de Internet.

Limitaciones por seguridad (debido a que se manejan nombres y edades de personas menores de edad):

- Las contraseñas de los entrenadores estarán cifradas.
- Se utilizarán tokens de autentificación para verificar que los usuarios son quienes dicen ser.

3.2.3 Requisitos específicos

En esta subsección se describirá con detalle todos los requisitos tanto funcionales como no funcionales que el sistema deberá satisfacer, definiendo así las funciones que se deben dar en el sistema.

REQUISITOS FUNCIONALES

Identificador del requisito	RF01
Nombre del requisito	Identificación de usuario
Característica	Los usuarios deberán identificarse para acceder a cualquier funcionalidad del aplicativo.
Descripción	El usuario accede a la vista de "Iniciar sesión" para introducir sus credenciales y poder acceder a las funciones del sistema.
Prioridad	Alta

Tabla 4 - Descripción Requisito funcional RF01

Para el RF01 se ha diseñado el *mockup* mostrado en la Ilustración 3.

Ilustración 3 - Mockup RF01

Identificador del requisito	RF02
Nombre del requisito	Registro de usuario
Característica	Un nuevo usuario deberá registrarse para obtener sus credenciales y acceder a las funciones del sistema
Descripción	Un nuevo usuario accede a la vista de "Registrarse" e introduce sus datos (nombre, nombre de usuario, contraseña, fecha de nacimiento, género y club) para verificar sus credenciales.
Prioridad	Alta

Tabla 5 - Descripción Requisito funcional RF02

Para el RF02 se ha diseñado el *mockup* mostrado en la Ilustración 4.

Ilustración 4 - Mockup RF02

Identificador del requisito	RF03
Nombre del requisito	Registrar nuevo nadador
Característica	Un entrenador puede añadir un nuevo deportista que se enlazará a su equipo.
Descripción	Un entrenador accede a la vista de "Mis nadadores" y selecciona la opción de "Añadir nuevo nadador". A continuación, el entrenador introduce el nombre, género y año de nacimiento de este, y recibe una notificación de que el nadador ha sido guardado correctamente.
Prioridad	Alta

Tabla 6 - Descripción Requisito funcional RF03

Para el RF03 se ha diseñado el *mockup* mostrado en la Ilustración 5.

Ilustración 5 - Mockup RF03

Identificador del requisito	RF04
Nombre del requisito	Eliminar nadador existente
Característica	Un entrenador puede eliminar un deportista que está enlazado a su equipo.
Descripción	Un entrenador accede a la vista de "Mis nadadores" y selecciona un nadador de los que están en su equipo. A continuación, en la vista individual de nadador, el entrenador selecciona la opción de "Eliminar". Cuando el entrenador confirma la eliminación, recibe una notificación de que los datos del nadador se han eliminado correctamente. Si el entrenador decide cancelar esta operación, no se elimina el nadador seleccionado.
Prioridad	Media

Tabla 7 - Descripción Requisito funcional RF04

Para el RF04 se ha diseñado el *mockup* mostrado en la Ilustración 6.

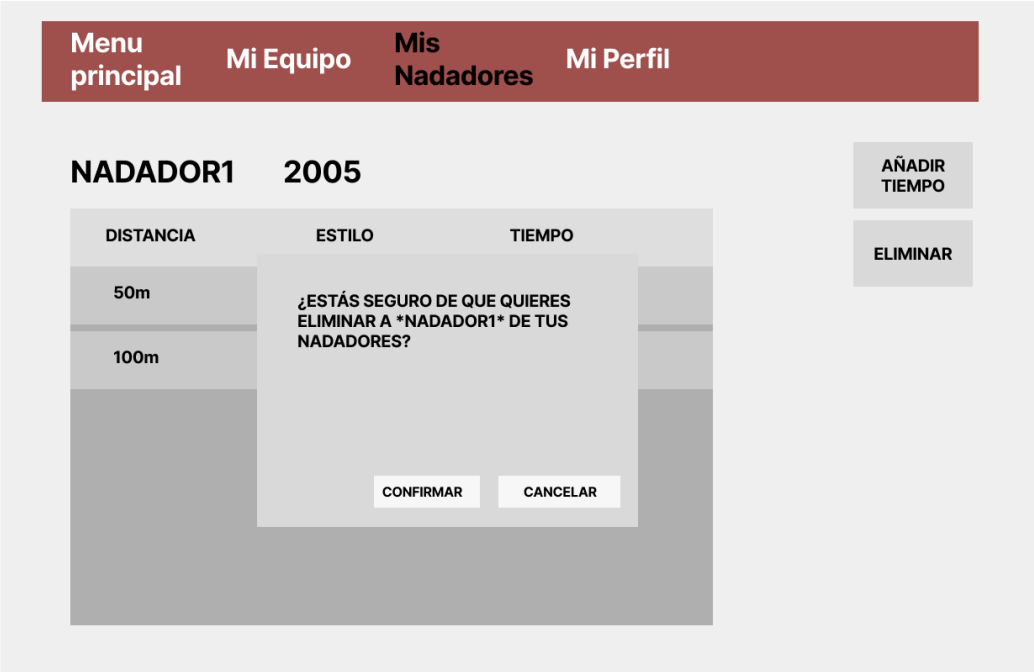


Ilustración 6 - Mockup RF04

Identificador del requisito	RF05
Nombre del requisito	Añadir tiempo a nadador
Característica	Un entrenador puede añadir un tiempo a un deportista de su equipo.
Descripción	Un entrenador accede a la vista de "Mis nadadores" y selecciona un nadador de los que están en su equipo. A continuación, en la vista individual de nadador, el entrenador selecciona la opción de "Añadir tiempo" y rellena todos los campos (selecciona la prueba e introduce el tiempo). Cuando el entrenador confirma los datos, recibe una notificación diciendo que los tiempos del nadador se han actualizado correctamente. Si el entrenador decide cancelar esta operación, no se añade el tiempo al nadador en la prueba seleccionada.
Prioridad	Alta

Tabla 8 - Descripción Requisito funcional RF05

Para el RF05 se ha diseñado el *mockup* mostrado en la Ilustración 7.

Ilustración 7 - Mockup RF05

Identificador del requisito	RF06
Nombre del requisito	Modificar/Actualizar tiempo de nadador
Característica	Un entrenador puede modificar un tiempo a un deportista de su equipo.
Descripción	Un entrenador accede a la vista de "Mis nadadores" y selecciona un nadador de los que están en su equipo. A continuación, en la vista individual de nadador, muestra los tiempos del nadador seleccionado, el entrenador selecciona la opción de "Añadir tiempo" y rellena todos los campos (selecciona una prueba que ya tiene tiempo e introduce el tiempo). Cuando el entrenador confirma los datos, recibe una notificación diciendo que los tiempos del nadador se han actualizado correctamente y se ha sobre escrito el tiempo en esa prueba. Si el entrenador decide cancelar esta operación, no se modifica el tiempo al nadador en la prueba seleccionada.
Prioridad	Alta

Tabla 9 - Descripción Requisito funcional RF06

Para el RF06 se ha diseñado el *mockup* mostrado en la Ilustración 7.

Identificador del requisito	RF07
Nombre del requisito	Inscribir nadador a prueba
Característica	Un entrenador puede inscribir a un deportista de su equipo a una prueba de la competición.
Descripción	Un entrenador accede a la vista principal, selecciona una competición del calendario elige la opción de "Inscripción". A continuación, llega a la vista de "Pruebas", donde escoge la prueba en la que quiere realizar la inscripción del nadador. Por último, selecciona al nadador de entre sus nadadores y selecciona "Inscribir". Después de esta acción, el nombre del nadador inscrito aparece en la lista de inscritos.
Prioridad	Alta

Tabla 10 - Descripción Requisito funcional RF07

Para el RF07 se ha diseñado el *mockup* mostrado en la Ilustración 8.

Ilustración 8 - Mockup RF07

Identificador del requisito	RF08
Nombre del requisito	Ver inscripciones de mi equipo
Característica	Un entrenador puede acceder a una vista de pruebas para ver los inscritos de su equipo.
Descripción	Un entrenador accede a la vista principal, selecciona una competición del calendario elige la opción de "Mis inscritos". A continuación, llega a la vista de "Mis Inscritos", donde puede ver todas las pruebas y los nadadores que tiene inscritos en ese momento. Desde esta vista, si el entrenador selecciona una prueba, accede a la inscripción de esa prueba
Prioridad	Alta

Tabla 11 - Descripción Requisito funcional RF08

Para el RF08 se ha diseñado el *mockup* mostrado en la Ilustración 9.

Ilustración 9 - Mockup RF08

Identificador del requisito	RF09
Nombre del requisito	Ver inscripciones de todos los equipos
Característica	Un entrenador puede acceder a una vista de pruebas para ver los inscritos de todos los equipos, por sesión.
Descripción	Un entrenador accede a la vista principal, selecciona una competición del calendario elige la opción de "Inscripciones". A continuación, llega a la vista de "Inscripciones", donde puede seleccionar la sesión que desee para ver las pruebas en esa sesión y todos los nadadores inscritos en esta.
Prioridad	Alta

Tabla 12 - Descripción Requisito funcional RF09

Para el RF09 se ha diseñado el *mockup* mostrado en la Ilustración 9.

Identificador del requisito	RF10
Nombre del requisito	Acceso a información del equipo
Característica	Un entrenador puede acceder a una vista donde aparecen datos relevantes sobre su equipo (número de nadadores, clasificación y puntuación).
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de "Mi Equipo" y la selecciona. A continuación, aparece una vista con el nombre del equipo al que pertenece, el número de nadadores que pertenecen al equipo (tanto del equipo masculino como del femenino) y una tabla con la clasificación de la liga.
Prioridad	Media

Tabla 13 - Descripción Requisito funcional RF10

Para el RF10 se ha diseñado el *mockup* mostrado en la Ilustración 10.

Ilustración 10 - Mockup RF10

Identificador del requisito	RF11
Nombre del requisito	Acceso a los nadadores del equipo
Característica	Un entrenador puede acceder a una vista donde aparecen todos los nadadores de su equipo.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de "Mis Nadadores" y la selecciona. A continuación, aparece una vista con una lista de todos los nadadores que pertenecen al equipo del entrenador, junto a las opciones de "Registrar nuevo nadador" (Tabla 6 - Descripción Requisito funcional RF03) y opciones de filtrado.
Prioridad	Alta

Tabla 14 - Descripción Requisito funcional RF11

Para el RF11 se ha diseñado el *mockup* mostrado en la Ilustración 11.

Ilustración 11 - Mockup RF11

Identificador del requisito	RF12
Nombre del requisito	Filtrar en lista de mis nadadores
Característica	Un entrenador puede acceder a una vista donde aparecen todos los nadadores de su equipo y filtrar.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de "Mis Nadadores" y la selecciona. A continuación, aparece una lista con todos los nadadores que pertenecen al equipo del entrenador (-Descripción Requisito funcional RF1). Además, aparece una opción de filtrado. La opción de filtrado permite filtrar la búsqueda con varios parámetros (género, masculino o femenino, y rango de años de nacimiento).
Prioridad	Media

Tabla 15 - Descripción Requisito funcional RF12

Para el RF12 se ha diseñado el *mockup* mostrado en la Ilustración 12.

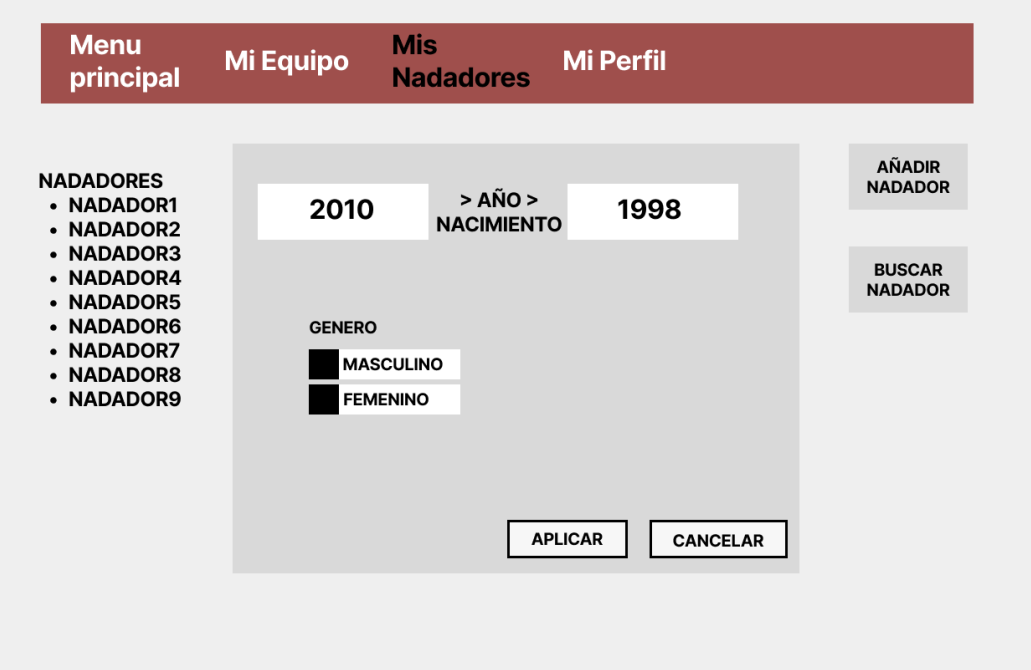


Ilustración 12 - Mockup RF12

Identificador del requisito	RF13
Nombre del requisito	Vista de Mi Perfil
Característica	Un entrenador puede acceder a una vista donde aparecen la información de su perfil (nombre, fecha de nacimiento y equipo al que pertenece).
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de "Mi Perfil" y la selecciona. A continuación, aparece una vista con los datos del entrenador (nombre, año de nacimiento y nombre del equipo al que pertenece).
Prioridad	Baja

Tabla 16 - Descripción Requisito funcional RF13

Para el RF13 se ha diseñado el *mockup* mostrado en la Ilustración 13.

Ilustración 13 - Mockup RF13

Identificador del requisito	RF14
Nombre del requisito	Modificar contraseña de entrenador
Característica	El entrenador puede modificar su contraseña de acceso a la aplicación.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de “Mi Perfil” y la selecciona. A continuación, en la vista de “Mi Perfil”, aparece una opción para cambiar la contraseña. Al seleccionarla el usuario es redirigido a una vista con un formulario donde se le pide escribir la contraseña anterior y la nueva contraseña.
Prioridad	Baja

Tabla 17 - Descripción Requisito funcional RF14

Para el RF14 se ha diseñado el *mockup* mostrado en la Ilustración 14.

Ilustración 14 - Mockup RF14

Identificador del requisito	RF15
Nombre del requisito	Cerrar sesión
Característica	Un entrenador puede cerrar su sesión.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de “Mi Perfil” y la selecciona. A continuación, en la vista de “Mi Perfil”, aparece una opción para cerrar sesión. Al seleccionarla el usuario es redirigido a la vista de bienvenida.
Prioridad	Media

Tabla 18 - Descripción Requisito funcional RF15

Identificador del requisito	RF16
Nombre del requisito	Exportar datos de tiempos de competición y entrenamiento
Característica	Un entrenador puede exportar en formato PDF los tiempos de entrenamiento y competición de un deportista.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de "Mis Nadadores" y la selecciona. A continuación, en la vista de "Mis Nadadores", selecciona el deportista que desee y accede a la vista del nadador. Aquí tiene una opción para descargar los tiempos que aparecen en las tablas.
Prioridad	Alta

Tabla 19 - Descripción Requisito funcional RF16

Para el RF16 se ha diseñado el *mockup* mostrado en la Ilustración 15.

Ilustración 15 - Mockup RF16

Identificador del requisito	RF17
Nombre del requisito	Calendario de competiciones
Característica	Un calendario con las competiciones de la temporada.
Descripción	En la ventana principal, aparecerá un calendario con las futuras competiciones, además de la sede donde se realizará, y las fechas de inicio y fin de esta.
Prioridad	Alta

Tabla 20 - Descripción Requisito funcional RF17

Para el RF17 se ha diseñado el *mockup* mostrado en la Ilustración 16.

Ilustración 16 - Mockup RF17

Identificador del requisito	RF18
Nombre del requisito	Vista individual del nadador
Característica	Un entrenador puede acceder a una vista personalizada de un nadador en concreto.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de “Mis Nadadores” y la selecciona. A continuación, en la vista de “Mis Nadadores”, una lista de todos los nadadores del equipo. El entrenador selecciona a un deportista y accede a esta vista individual. En esta aparecen los tiempos oficiales de competición y de entrenamiento, además de la posibilidad de añadir o actualizar estos tiempos, eliminar el deportista o exportar los datos en formato PDF.
Prioridad	Alta

Tabla 21 - Descripción Requisito funcional RF18

Para el RF18 se ha diseñado el *mockup* mostrado en la Ilustración 15.

Identificador del requisito	RF19
Nombre del requisito	Actualizar tiempos de entrenamiento
Característica	Cuando se actualiza un tiempo oficial, se actualiza automáticamente el tiempo de entrenamiento.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de “Mis Nadadores” y la selecciona. A continuación, en la vista de “Mis Nadadores”, aparece una lista con todos los deportistas del equipo. El entrenador selecciona uno de los deportistas. En la vista individual, cuando el entrenador actualiza o añade un tiempo de 100m o 200m, los tiempos de entrenamiento se actualizan de acuerdo con los nuevos tiempos.
Prioridad	Alta

Tabla 22 - Descripción Requisito funcional RF19

Identificador del requisito	RF20
Nombre del requisito	Ver puntos iniciales
Característica	Dadas las inscripciones, se calculan los puntos totales y de cada sesión, separado por género.
Descripción	Desde el calendario, el entrenador puede acceder a la opción de “Ver puntos iniciales”. En esta vista, aparecen las listas de inscripción junto a varias tablas de puntuación, indicando los puntos sumados por cada club en cada sesión, separado por género.
Prioridad	Alta

Tabla 23 - Descriptión Requisito funcional RF20

Para el RF20 se ha diseñado el *mockup* mostrado en la Ilustración 17.

Ilustración 17 - Mockup RF20

Identificador del requisito	RF21
Nombre del requisito	Calcular mejores nadadores por prueba
Característica	Un entrenador puede acceder a una vista para conocer los nadadores con las mejores marcas en cada prueba.
Descripción	Desde el calendario, cuando el entrenador selecciona un campeonato, puede acceder a una opción para ver sus inscritos y conocer la posibilidad de mejorar las listas escogidas (y a su vez realizar cambios en las listas de inscritos para obtener más puntos en el campeonato).
Prioridad	Alta

Tabla 24 - Descripción Requisito funcional RF21

Para el RF21 se ha diseñado el *mockup* mostrado en la Ilustración 18.

Ilustración 18 - Mockup RF21

Identificador del requisito	RF22
Nombre del requisito	Ver programas de pruebas
Característica	Un entrenador puede acceder a una vista para conocer el orden de las pruebas de un campeonato.
Descripción	Desde el calendario, aparece una opción a través de la cual aparece una vista en la que salen todos los deportistas del equipo del entrenador inscritos por pruebas, y para cada prueba, aparece una lista con todos los deportistas con marca acreditada. Desde esta vista el entrenador puede cambiar automáticamente las inscripciones.
Prioridad	Alta

Tabla 25 - Descripción Requisito funcional RF22

Para el RF22 se ha diseñado el *mockup* mostrado en la Ilustración 19.

Ilustración 19 - Mockup RF22

Identificador del requisito	RF23
Nombre del requisito	Dar de baja a nadador en prueba
Característica	Un entrenador puede dar de baja a los deportistas de una prueba, si no pueden asistir o si quiere inscribir a otro deportista en su lugar.
Descripción	Desde el programa de pruebas, el entrenador selecciona una prueba con nadadores inscritos. Cuando está en la vista de inscripción, tiene a su vez, una opción para dar de baja a los nadadores inscritos de su club.
Prioridad	Alta

Tabla 26 - Descripción Requisito funcional RF23

Para el RF23 se ha diseñado el *mockup* mostrado en la Ilustración 8.

Identificador del requisito	RF24
Nombre del requisito	Cambiar nombre de usuario
Característica	El entrenador puede modificar su nombre de usuario de la aplicación.
Descripción	Un entrenador accede a la vista principal. En el menú superior tiene la opción de "Mi Perfil" y la selecciona. A continuación, en la vista de "Mi Perfil", aparece una opción para cambiar el nombre de usuario. Al seleccionarla el usuario es redirigido a una vista con un formulario donde se le pide escribir el nuevo nombre de usuario, posteriormente es redirigido a la ventana inicial.
Prioridad	Baja

Tabla 27 - Descripción Requisito funcional RF24

REQUISITOS NO FUNCIONALES

Identificador del requisito	RNF01
Nombre del requisito	Encriptado de la contraseña de los entrenadores
Característica	Encriptación de la contraseña en la base de datos
Descripción	Debido a la sensibilidad de los datos manipulados (nombre y edad de personas que pueden ser menores de edad), la contraseña de acceso en la base de datos aparecerá encriptada.
Prioridad	Alta

Tabla 28 - Descripción Requisito no funcional RNF01

3.3 Límites del Sistema

El aplicativo tiene un único actor externo, que es *TRAINER*, que realiza todas las interacciones con el sistema.

Debido a la simplicidad del diagrama de contexto, se ha decidido omitirlo.

4. Diseño de la solución

En el presente capítulo se describe la arquitectura y las tecnologías utilizadas para desarrollar el sistema. Esta arquitectura sirve de base para el desarrollo, así como para sentar las bases de este.

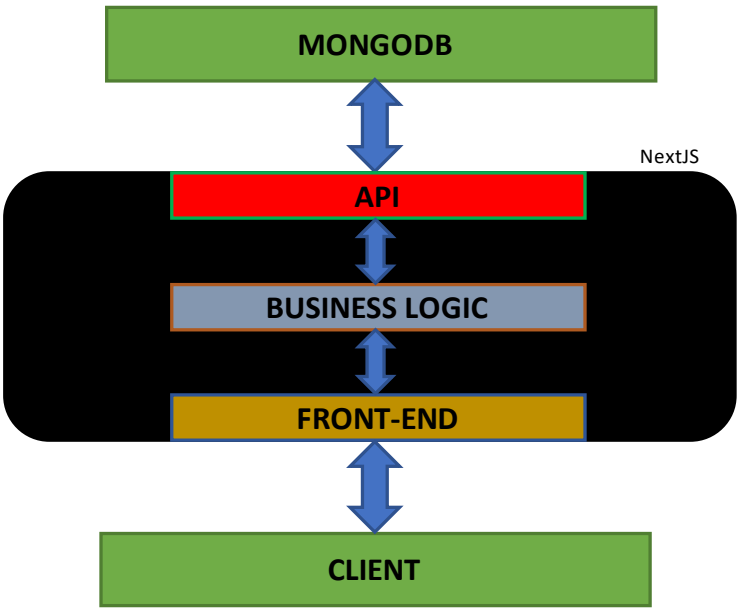


Ilustración 20 - Arquitectura de la aplicación

4.1 Diseño detallado

Como se puede observar en la Ilustración 20, la aplicación se divide en una arquitectura de 4 capas separas e interconectadas, de las cuales 3 capas están dentro del mismo proyecto (el proyecto de *NextJS*).

La base de datos está alojada en MongoDB Atlas. Es una base de datos no relacional, por lo que no tiene un esquema especificado, y se guarda la información en documentos con formato "*.json*" (*JavaScript Object Notation*).

La capa *API* es la capa de acceso de datos, en esta se hacen todas las peticiones a la base de datos para realizar las operaciones CRUD (*Create, Read, Update, Delete*) sobre la base de datos. Esta es la única conexión que hay entre la base de datos y la aplicación, marcando así una separación clara entre la lógica de negocio y la base de datos.

La capa *business logic* es la capa donde se implementan todas las operaciones de cálculo en el lado del servidor (funciones tales como *getAllClubs, getSwimmersOfTeam, getTrainer…*).

Por último, la capa *front-end*, donde se llama a las operaciones de la capa *business logic* para obtener los datos necesarios y mostrárselos al CLIENTE en el navegador.

El equipo ha escogido esta arquitectura sobre otros tipos de arquitectura (Cliente-Servidor, *Broker*…) debido a que es una arquitectura que separa claramente las distintas capas y las funcionalidades, pudiendo cambiar de tecnología en una capa sin tener que modificar todo el sistema para que el comportamiento esperado siga siendo el mismo.

Una de las ventajas más importantes de esta arquitectura es la facilidad de implementar *tests* automáticos, ya que permite *testear* cada capa de forma independiente.

4.2 Tecnología utilizada

Para desarrollar una aplicación web se pueden utilizar distintas herramientas que ayuden a codificar. Se ha decidido trabajar con las siguientes herramientas.

4.2.1 Tecnologías de entorno

Con estas herramientas se ha preparado un entorno amigable y fácil de utilizar para desarrollar la aplicación y seguir la metodología explicada.

4.2.1.1 Visual Studio Code

Como *IDE* se ha decidido utilizar *Visual Studio Code* [3], desarrollado por *Microsoft*, debido a la facilidad de uso y a la cantidad de *plug-ins* distintos que tiene para facilitar los procesos de codificación y pruebas durante el desarrollo.

4.2.1.2 Git

Git [4] se emplea para el control de versiones, alojando el proyecto en un repositorio en *GitHub*[1]. Además, siguiendo el modelo de buenas prácticas de *Git*, se presenta un modelo en el que en la rama *main* el sistema siempre está funcionando correctamente. A partir de esta rama, hay una segunda rama principal llamada *tests*, donde se ejecutan todas las pruebas diseñadas e implementadas para verificar que la característica o requisito implementado funciona correctamente de acuerdo con la especificación, y que su integración con el resto de la aplicación es correcta (no surgen problemas o *bugs* inesperados que no se han tenido en cuenta durante el desarrollo). Por último, las ramas de desarrollo, donde en cada una se

[1] https://github.com/

trabaja una *feature* específica y al terminar, se junta con la rama *tests* para verificar su correcto desarrollo.

4.2.2 Tecnologías de diseño

Con estas tecnologías, se han realizado los diseños y los diferentes modelos y diagramas utilizados para el desarrollo presentes en esta memoria.

4.2.2.1 Figma

Figma [5] es una herramienta de edición de gráficos vectoriales (no pixelado) desarrollada por *Figma Inc.* Es muy útil para desarrollar los diseños de las interfaces, ya que tiene una gran variedad de elementos por defecto y, además, tiene acceso a una tienda interna de *plug-ins* y *componentes* que se pueden descargar en tu librería y se pueden utilizar sin restricción.

Del mismo modo, se pueden insertar imágenes de otras fuentes y exportar los componentes diseñados en varios formatos (PNG, JPG, SVG o PDF).

A su vez, cuando se han diseñado varias interfaces, se puede realizar una presentación escogiendo los diseños deseados, como si se tratara de la misma aplicación en desarrollo.

4.2.2.2 Lucid Chart

Lucid Chart [6] es otra herramienta gráfica para realizar interfaces. Una peculiaridad que tiene es que contiene *plug-ins* con componentes para realizar modelos en *UML* (modelos de dominio, diagramas de contexto o diagramas de casos de uso).

Esta herramienta, al igual que 4.2.2.1 Figma4.2.2.1 Figma, permite importar imágenes y exportar los diseños generados en formato PDF, PNG, JPG o SVG.

4.2.3 Tecnologías de desarrollo

Estas tecnologías son las utilizadas por el equipo de desarrollo para desarrollar e implementar todos los requisitos (tanto funcionales como no funcionales) mencionados.

4.2.3.1 NextJS

NextJS [7] es un *framework* de *JavaScript* que pertenece a la compañía Vercel. Este *framework* aporta *building blocks* para crear aplicaciones web de manera sencilla.

NextJS incorpora toda la funcionalidad de *React* y la mejora, haciendo muchas de sus funcionalidades mucho más intuitivas y fáciles de utilizar.

Con este *framework* el sistema de *routing* (las rutas de las distintas páginas) se genera automáticamente dentro de su carpeta *pages* (cada archivo creado en la carpeta *pages* tiene su propia ruta en el navegador) y contiene, además, un sistema para tener la *API* dentro del mismo proyecto.

Del mismo modo, poder manejar el sistema de *front-end* y el *back-end* dentro del mismo proyecto ayuda a poder separar las capas de *"Business Logic"* y "*Presentación"* mostradas en la Ilustración 20.

4.2.3.2 TypeScript

TypeScript [8] es una tecnología implementada por *Miscrosoft* que es, a su vez, un superset de *JavaScript*. *TypeScript* es una solución a muchos problemas de *JavaScript*, debido a que a todo el sistema que utiliza *JavaScript*, le añade la característica de los tipos y de crear interfaces. Esto ayuda a que el desarrollo en *TypeScript* sea mucho más seguro y mantenible que el desarrollo en *JavaScript*.

4.2.3.3 Tailwind

TailwindCSS [9] es un *framework* de *CSS* (*Cascading Style Sheets*) de utilidades básicas, es decir, que aplica clases de *CSS* al código de *HTML* sin necesidad de escribir código en un archivo ".css".

5. Desarrollo de la solución

En este capítulo se explicará el proceso seguido durante el desarrollo del aplicativo hasta conseguir una primera versión viable para su uso.

El primer paso ha sido capturar los requisitos, documentarlos y ordenarlos según su prioridad, de tal forma que, al finalizar esta etapa, se tiene una lista de los requisitos (esta lista está explicada en la sección 3.2.3 Requisitos específicos).

Además, como se ha explicado en la sección 1.4, se ha seguido una metodología ágil para el desarrollo con *sprints* de 3 semanas, en las que se empieza realizando el diseño de los casos de uso y definiendo una serie de pruebas de aceptación para verificar que estaba desarrollado correctamente.

El código se ha dividido por carpetas, de manera que quede un sistema de carpetas bien estructurado y fácil de comprender. Para diferenciar la capa *business logic* de la capa de *front-end* se ha separado en carpetas su implementación. Se han desarrollado todas las funciones de la capa *business logic* en la carpeta "**lib**", referenciando todas las funciones que se pueden realizar de cada entidad (un archivo con las operaciones sobre la clase *TRAINER*, otro archivo sobre las operaciones en la clase *SWIMMER*, *MARK*, *INSCRIPTION*, *CLUB* y *CHAMPIONSHIP*, tal y como se muestra en la Ilustración 21. Estas operaciones se exportan para poder ser utilizadas posteriormente en las páginas que se van a desarrollar en la capa de *front-end*. En la Ilustración 21 también se puede apreciar un archivo nombrado *mongodb.ts*, este se utiliza para obtener un objeto que realiza la conexión con la base de datos *MongoDB*.

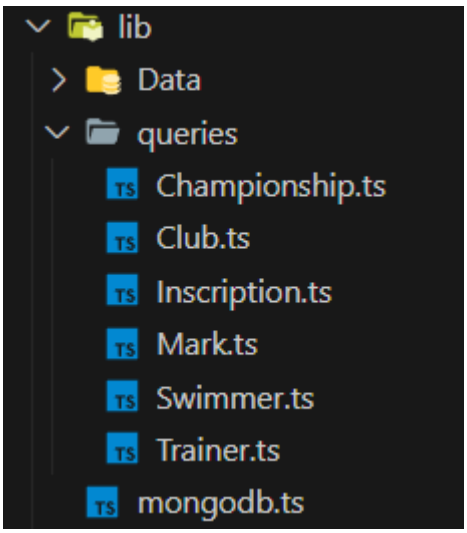


Ilustración 21 - Contenido carpeta lib

Aislar estas dos capas en distintas carpetas permite que todas las conexiones a la *API* y a la base de datos se realice desde los archivos en esta carpeta. Estas llamadas a la *API* se realizan mediante la librería *axios*, que es un cliente HTTP basado en promesas (asíncrono) muy utilizado y con una amplia documentación.

Esto significa que cuando se llama a una de las funciones de la *business logic* desde el *front-end*, este tiene que esperar a que se termine de ejecutar la operación para poder continuar con su ejecución normal.

Para comprobar la independencia de capas y que no se realiza ninguna operación directamente desde el *front-end* se ha evitado importar *axios* desde cualquier archivo en el *front-end*, de forma que todas las operaciones en la base de datos se realizan desde la capa *business logic*, que sí importa la librería *axios*.

5.1 Primer *Sprint*

En el primer *sprint* se decidió implementar funciones sencillas para realizar la toma de contacto con la tecnología y familiarizarse con el entorno de desarrollo escogido.

Durante el *sprint* se han desarrollado los siguientes requisitos:

- Identificación de usuario (Tabla 4)
- Registrarse en la aplicación (Tabla 5)
- Acceso a tu equipo (Tabla 14)
- Cerrar sesión (Tabla 19)
- Cambiar contraseña (Tabla 17)
- Cambiar nombre de usuario (Tabla 27)

Para explicar el desarrollo de cada requisito se ha decidido mostrar las pruebas manuales que se han definido para dar por finalizado el desarrollo de cierto requisito. Este esquema se ha seguido para explicar todos los *sprints* aquí mencionados.

5.1.1 Identificación de usuario

Pruebas de aceptación:

Entrada	Salida esperada
Nombre de usuario inexistente, Contraseña rellenada (userName= “trainer” password = “123”)	Mensaje de error: “¡Credenciales incorrectas!”
Nombre de usuario existente, Contraseña incorrecta (userName = “trainer1” password = “trainer”	Mensaje de error: “Contraseña incorrecta”
Nombre de usuario o Contraseña vacío (userName = “trainer1” password = “”) o (userName = “” password = “3r123”)	No permite pulsar botón de “Iniciar sesión”
Nombre de usuario y Contraseña correctos (userName = “trainer1” password = “trainer1”)	Accede a la ventana principal de la aplicación
El usuario selecciona la opción de “Registrarse”	Accede a la ventana de registro de la aplicación

Tabla 29 - Pruebas de aceptación RF01

Desarrollo:

Para desarrollar este requisito se ha utilizado una librería de autentificación llamada *nextAuth* junto con *bcrypt* para encriptar las contraseñas, con el objetivo de tener un sistema de seguridad a la hora de iniciar sesión (también utilizado en 5.1.2 Registro de usuario). Se aplica esta medida de seguridad debido a que hay información sensible sobre nombres y edades de personas menores de edad.

NextAuth tiene la posibilidad de realizar la verificación de credenciales mediante *tokens* de verificación, y además permite la integración con aplicaciones de terceros (*providers* desde su documentación oficial[2]), tales como *GitHub, Gmail* o *Facebook*, a la vez que permite realizar conexiones con varias bases de datos (*adapters*), como *Fauna, Firebase* o *MongoDB*. En este caso hemos utilizado el *adapter* con *MongoDB* (la base de datos utilizada) y, además, hemos creado nuestro propio *provider*, para poder crear un formulario personalizado a nuestro gusto.

[2] https://next-auth.js.org/getting-started/introduction

5.1.2 Registro de usuario

Pruebas de aceptación:

Entrada	**Salida esperada**
Algún campo sin rellenar	Mensaje de error: "Rellena todos los campos, por favor"
Las contraseñas no coinciden (password = "trainer123" rPassword = "trainer")	Mensaje de error: "Las contraseñas no coinciden"
Nombre de usuario existente (userName = "trainer1")	Mensaje de error: "Nombre de usuario ya registrado"
Rellenas todos los campos correctamente (name= "trainer3" userName = "trainer3" año de nacimiento = 1976 contraseña = "trainer3" repite contraseña = "trainer3" género = **selecciona una opción** club = **selecciona una opción**)	Se completa el registro, se inicia sesión y se accede a la ventana principal de la aplicación

Tabla 30 - Pruebas de aceptación RF02

Desarrollo:

Como se ha comentado en la sección 5.1.1 Identificación de usuario, se ha utilizado la librería *nextAuth* para realizar el control de sesión, además del registro de nuevos usuarios. Además, para guardar la contraseña en la base de datos, se ha utilizado la librería *bcrypt* para encriptarla, de forma que las personas que tienen acceso a la base de datos no tienen acceso a la contraseña de los usuarios de la aplicación.

Tal y como se muestra en la Ilustración 222, se utiliza la librería *bcrypt* para generar un *salt* y posteriormente pasa la contraseña simple introducida por el usuario por un algoritmo de *hash* utilizando este *salt*. El proceso de aplicar *salt* a una contraseña consiste en introducir *bytes* aleatorios en esta, de manera que genera un *hash* impredecible en longitud. Esta medida de seguridad se toma porque utilizando solamente la función "*hash()*" el resultado para la misma contraseña es siempre el mismo. Dos personas que comparten contraseña tienen el campo "*password*" diferente en la base de datos.

```
export default async function handler(req: NextApiRequest, res: NextApiResponse) {
  const body = req.body
  const user = await Trainer.findOne({ userName: body.userName })
  if (user) {
    res.status(200).json({ success: false,  message: 'Nombre de usuario ya registrado' })
    return;
  }
  const salt = await bcrypt.genSalt(10);

  const hashPass = await bcrypt.hash(body.password, salt)
  await Trainer.create({
    name: body.name,
    birthYear: body.birthYear,
    userName: body.userName,
    genre: body.genre,
    password: hashPass,
    club: body.club + "",
  });
  res.status(200).json({ success: true, message: 'success' })
}
```

Ilustración 22 - Fragmento de código, registro de un nuevo usuario

5.1.3 Acceso a mi equipo

Pruebas de aceptación:

Entrada	Salida esperada
Entra al apartado de “Mi Equipo”	Sale la ventana de “Mi Equipo” con los datos del club del usuario *loggeado*
Aparece el número de nadadores de tu equipo.	Al pulsar sobre el texto, redirige al apartado de “Mis Nadadores”.
Aparece una tabla con la clasificación de la liga.	Aparece una tabla con la clasificación con los puntos obtenidos hasta la fecha.

Tabla 31- Pruebas de aceptación RF10

5.1.4 Acceso a mi perfil

Pruebas de aceptación:

Entrada	Salida esperada
Selecciona el nombre de su equipo	Es redirigido a la vista de “Mi Equipo”
En el menú lateral, selecciona la opción “Cerrar sesión”	Se cierra la sesión del entrenador y le redirige automáticamente a la ventana de bienvenida
En el menú lateral, selecciona la opción “Cambiar nombre”	Aparece la vista para cambiar el nombre de usuario
En el menú lateral, selecciona la opción “Cambiar contraseña”	Aparece la vista para cambiar la contraseña de acceso

Tabla 32- Pruebas de aceptación RF13

5.1.5 Cerrar sesión

Pruebas de aceptación:

Entrada	Salida esperada
Selecciona la opción de “Cerrar sesión”	Es redirigido a la página de bienvenida y el *token* de autentificación es borrado.

Tabla 33- Pruebas de aceptación RF15

Desarrollo:

Para realizar la salida de la sesión se ha utilizado la librería *nextAuth*, ya que a la vez que aporta las herramientas para iniciar sesión y registrar nuevos usuarios, aporta las funciones necesarias para cerrar sesión y eliminar el *token* de autentificación, de forma que no es posible volver a acceder a la información interna de la aplicación si no se inicia sesión nuevamente.

5.1.6 Cambiar contraseña

Pruebas de aceptación:

Entrada	Salida esperada
Selecciona la opción de “Cambiar contraseña”.	Se abre el formulario para rellenar los datos.
Rellena los datos incorrectamente (contraseña antigua es incorrecta): OldPass: “1234” NewPass: “abcdef” RepNewPass: “abcdef”	Mensaje de error: “Contraseña incorrecta”.
Rellena los datos incorrectamente (contraseña nueva no se repite correctamente): OldPass: “1234entrenador” NewPass: “abcdef” RepNewPass: “meHeEquivocado”	Mensaje de error: “Las contraseñas no coinciden”.
Rellena los datos correctamente	Mensaje de confirmación: “Contraseña cambiada con éxito”.

Tabla 34- Pruebas de aceptación RF14

Desarrollo:

Para realizar el cambio de contraseña y la verificación entre la contraseña guardada en la base de datos y la contraseña introducida en el formulario, se ha utilizado la librería *bcrypt*, ya que las contraseñas en la base de datos están encriptadas, y es necesario encriptar las nuevas para compararlas antes de realizar el cambio en la base de datos.

Para realizar este cambio se han seguido los mismos pasos que para registrar a un nuevo usuario, para mantener segura la contraseña, primeramente, se ha generado

un *salt* con la contraseña, y posteriormente se ha calculado el *hash* de esta, para después registrarla en la base de datos.

5.1.7 Cambiar nombre de usuario

Pruebas de aceptación:

Entrada	**Salida esperada**
Selecciona la opción de "Cambiar nombre de usuario".	Se abre el formulario para rellenar los datos.
Rellena los datos incorrectamente (nombre de usuario ya está en uso): NewUserName: "trainer1"	Mensaje de error: "Nombre de usuario ya en uso".
Rellena los datos correctamente	Mensaje de confirmación: "Nombre de usuario cambiada con éxito". Cierra la sesión del usuario

Tabla 35- Pruebas de aceptación RF24

Desarrollo:

Para realizar el cambio de nombre de usuario surgió un problema para verificar el nombre del usuario con el *token* de autentificación, lo que impedía mantener la sesión iniciada. Se ha decidido cerrar la sesión del usuario una vez se cambie el nombre de usuario, para que el usuario inicie sesión con el nuevo nombre, y el *token* esté asociado a ese nuevo nombre.

5.2 Segundo *Sprint*

El segundo *sprint* se ha enfocado en las distintas herramientas para mostrar información del equipo y de los deportistas. Se han preparado las operaciones *CRUD* (*Create*, *Read*, *Update*, *Delete*) para las inscripciones, las marcas y los nadadores en la base de datos, además de preparar una serie de competiciones para el siguiente *sprint*.

Durante el *sprint* se han desarrollado los siguientes requisitos:

- Registrar nuevo nadador (Tabla 6)
- Ver mis nadadores (Tabla 14)
- Filtrar mis nadadores (Tabla 15)
- Calendario de competiciones (Tabla 20)

5.2.1 Registrar nuevo nadador

Pruebas de aceptación:

Entrada	Salida esperada
Se rellenan todos los campos con datos válidos (un nuevo deportista, sin repetir nombre y edad dentro del club)	Se añade el nuevo deportista y se actualiza la lista automáticamente
Se rellena los campos con un nombre y edad repetidos dentro del mismo equipo	Mensaje de error: "Este nadador ya está registrado"
Se queda al menos un campo sin rellenar.	La opción de crear nadador está bloqueada.
Se cancela la creación de nadador	Se cierra el *popup* de "Añadir nadador"
Se confirma la creación de nadador con datos válidos	Se cierra el *popup* de "Añadir nadador" y se actualiza la lista de nadadores, apareciendo el nuevo nadador creado

Tabla 36- Pruebas de aceptación RF03

5.2.2 Ver mis nadadores

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador entra a la vista de "Mis Nadadores"	Aparece una lista con todos los nadadores del equipo.
El entrenador selecciona a uno de sus nadadores	Se abre una vista personalizada del nadador seleccionado.
El entrenador no tiene nadadores asociados	Aparece una lista vacía.

Tabla 37- Pruebas de aceptación RF11

5.2.3 Filtrar mis nadadores

Pruebas de aceptación:

Entrada	Salida esperada
Se deja el filtro de “Año de nacimiento” en los valores por defecto	Aparecen todos los nadadores del equipo
Se marca un único género y el filtro de “Año de nacimiento” en valor por defecto	Aparecen todos los nadadores del género marcado
Se cambia el año superior del filtro “Año de nacimiento”	Aparecen todos los nadadores de edad inferior al año superior (no año igual)
Se cambia el año inferior del filtro “Año de nacimiento”	Aparecen todos los nadadores de edad superior al año inferior (no año igual)
El año inferior es superior al año superior	Mensaje de error: “Año superior debe ser superior al año inferior”
Se cambian los valores de filtro y se pulsa cancelar	Los filtros no se aplican en la lista

Tabla 38- Pruebas de aceptación RF12

Desarrollo:

Para desarrollar este requisito, se ha decidido separar en dos listas a todos los nadadores del entrenador, de forma que puede ver en una lista el equipo masculino y en otra lista el equipo femenino. Para cada lista existe un filtro, que es independiente del otro. Hasta que no se selecciona la opción de “APLICAR”, los filtros definidos no se aplican a la lista en cuestión.

5.2.4 Calendario de competiciones

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador selecciona el día de hoy	Se resalta en el calendario
Selecciona un día distinto a hoy, no hay competición	Se resalta en el calendario de otro color, muestra un mensaje indicando que ese día no hay competición
Selecciona un día distinto a hoy, hay competición	Se resalta en el calendario de otro color, muestra un mensaje indicando que ese día hay competición

Tabla 39- Pruebas de aceptación RF17

Desarrollo:

Para poder introducir un calendario en la aplicación se ha utilizado la librería *react-calendar* y se ha desarrollado en un componente externo, aplicando el concepto de *ReactJS* para su reutilización en diferentes páginas. El principal problema surgido con esta librería es que no está integrada con el uso de *Tailwind CSS*, por lo que no se ha podido utilizar este *framework* para aplicar estilo al calendario y se ha tenido que utilizar *CSS* directamente.

5.3 Tercer *Sprint*

Para el desarrollo del tercer *sprint* se decidió incorporar todas las funcionalidades preparadas en el anterior *sprint*, relacionadas con la vista individual del nadador, para poder acceder a sus tiempos y poder manipular información sobre los deportistas y sus marcas.

Durante el *sprint* se han desarrollado los siguientes requisitos:

- Información de nadadores de mi equipo (Tabla 21)
- Eliminar nadador (Tabla 7)
- Añadir tiempo a nadador (Tabla 8)
- Modificar/Actualizar tiempo de nadador (Tabla 9)
- Actualizar tiempo de entrenamiento (Tabla 22)
- Exportar tiempos de competición y entrenamiento (Tabla 19)

5.3.1 Información de nadadores de mi equipo

Pruebas de aceptación:

Entrada	**Salida esperada**
El entrenador selecciona un nadador que no tiene tiempos registrados	Aparece la lista vacía
El entrenador selecciona un nadador que tiene tiempos registrados	Aparece la lista con los tiempos ordenador por prueba del nadador seleccionado y tiempos de entrenamiento asociados.
El entrenador elimina al nadador	Se elimina de la base de datos al nadador y todos los tiempos asociados a este
Se añade un tiempo de una prueba nueva al nadador	Aparece la lista de tiempos actualizada
Se añade un tiempo de una prueba ya registrada al nadador	Aparece la lista de tiempos con el nuevo registro
Se añade un tiempo con algún campo vacío	Mensaje de error: “Rellena el formulario correctamente”

Tabla 40- Pruebas de aceptación RF18

5.3.2 Eliminar nadador

Pruebas de aceptación:

Entrada	Salida esperada
Se elimina el nadador	Se elimina de la base de datos al nadador y todos los tiempos asociados a este y el usuario es redirigido a la vista de “Mis Nadadores”.
Se cancela “Eliminar nadador”	El nadador no se elimina.

Tabla 41- Pruebas de aceptación RF04

5.3.3 Añadir tiempo a nadador

Pruebas de aceptación:

Entrada	Salida esperada
Se introduce un tiempo nuevo (estilo: CR, Distancia :100 Tiempo: Min=1, Sec=2, Mil=36)	Se añade un nuevo registro en la tabla con los valores (100 CR 1’ 2” 36) y se coloca de forma ordenada.
No se rellenan todos los campos (estilo: CR Distancia: “” Tiempo: Min=1, Sec=2, Mil=36)	Mensaje de error: “Rellena el formulario correctamente”.
Se cancela “Añadir tiempo”	La lista no se actualiza, y se reinicia el formulario (todos los valores vuelven a los valores por defecto).

Tabla 42- Pruebas de aceptación RF05

5.3.4 Modificar/Actualizar tiempo de nadador

Pruebas de aceptación:

Entrada	Salida esperada
Se introduce un tiempo repetido (estilo: CR, Distancia :100 Tiempo: Min=1, Sec=2, Mil=36)	Se actualiza en la tabla el antiguo registro con el nuevo valor de tiempo (1’ 2” 36) y se coloca de forma ordenada.
No se rellenan todos los campos (estilo: CR Distancia: “” Tiempo: Min=1, Sec=2, Mil=36)	Mensaje de error: “Rellena el formulario correctamente”.
Se cancela “Añadir tiempo”	La lista no se actualiza, y se reinicia el formulario (todos los valores vuelven a los valores por defecto).

Tabla 43- Pruebas de aceptación RF06

Desarrollo:

Para realizar este requisito, se tenía que comprobar previamente que existiera una marca del nadador en la misma distancia, siendo así la forma de actualizar este tiempo o, en caso contrario, se crearía un nuevo registro, dando lugar al requisito explicado en el punto anterior (

5.3.3 Añadir tiempo a nadador)

5.3.5 Actualizar tiempo de entrenamiento

Pruebas de aceptación:

Entrada	**Salida esperada**
El nadador no tiene tiempos de competición registrados.	La tabla "Tiempos de entrenamiento" no contiene ningún registro.
El nadador solamente tiene tiempos registrados en distancias de 100m	La tabla "Tiempos de entrenamiento" no contiene ningún registro.
El nadador tiene tiempos registrados en distancias de 100m y 200m (100m crol: 1' 10", 200m crol: 2' 50")	La tabla "Tiempos de entrenamiento" contiene un registro con valor (100m crol, 1' 40")
El entrenador cambia el valor del 200m crol a 2' 40"	La tabla "Tiempos de entrenamiento" se actualiza y el registro de 100m crol tiene el valor 1' 30".

Tabla 44- Pruebas de aceptación RF19

Desarrollo:

Cuando un entrenador ha introducido valores para las distancias de 100m y 200m de un determinado estilo (crol, mariposa, espalda o braza), en la tabla de "Tiempos de entrenamiento" aparece un registro que determina el tiempo ideal por cada 100m que se nadan a cierto estilo.

Cada vez que se introduce un nuevo "tiempo de competición" o se actualiza uno de estos, la tabla de "tiempos de entrenamiento" se actualiza automáticamente, de forma que siempre están en correlación con los tiempos realizados durante una competición.

Este tiempo de entrenamiento se ha calculado mediante la fórmula de la **Velocidad Crítica de Nado** [10] (*CSS* en inglés, *Critical Swimming Speed*). Esta es la velocidad máxima que puede mantenerse durante un tiempo prolongado sin agotamiento. La fórmula es la siguiente:

$$V_{crit} = \frac{(D_2 - D_1)}{(T_2 - T_1)}$$

En esta ecuación, la distancia D_2 es mayor a la distancia D_1, igual que con el tiempo (pasado a unidades de segundo) T_2 es mayor al tiempo T_1 .Generalmente, esta prueba se realiza con $D_2 = 400$ y $D_1 = 200$, pero en este caso se realizará con los valores $D_2 = 200$ y $D_1 = 100$, de forma que se puede calcular la velocidad crítica para todos los estilos. Utilizando esta fórmula se obtiene la velocidad en m/s (porque utilizamos unidades del sistema internacional) y posteriormente, se divide 100 entre este valor y así se obtiene el tiempo total en una distancia de 100m.

5.3.6 Exportar tiempos de competición y entrenamiento

Descripción:

Desde la vista individual del nadador, el entrenador tiene la opción de descargar los tiempos de competición y de entrenamiento (en el caso que tenga tiempos registrados) en formato PDF para poder acceder a ellos desde fuera de la aplicación.

Este documento PDF contiene la información esencial para saber tanto los tiempos de entrenamiento como los de competición. Se guardan con el nombre "marcas_[NombreNadador].pdf". Este documento contiene el nombre del deportista al que pertenecen los tiempos, una tabla con los tiempos de competición, y en la página siguiente, los tiempos de entrenamiento, si tiene estos tiempos registrados.

Pruebas de aceptación:

Entrada	**Salida esperada**
El nadador no tiene tiempos de competición registrados.	No se puede exportar el documento PDF.
El nadador solamente tiene tiempos de competición registrados.	En el documento generado solamente aparece la tabla de tiempos de competición (únicamente hay 1 página en el documento).
El nadador tiene tanto tiempos de competición como tiempos de entrenamiento registrados.	El documento generado contiene 2 páginas. La primera página contiene los tiempos de competición, y la segunda página contiene los tiempos de entrenamiento.

Tabla 45- Pruebas de aceptación RF16

Desarrollo:

Para desarrollar esta funcionalidad se han utilizado las librerías *jsPDF* y *autoTable*, ambas para manipular documentos PDF.

La primera librería, *jsPDF*, sirve para crear, insertar y manipular texto plano en un documento PDF. Las principales funciones son "*.text()*" y "*.save()*". Con la primera

se introduce el texto plano deseado como primer parámetro y a continuación, margen lateral y superior. La función *save()* recibe un único parámetro, que es el nombre del archivo que se va a generar.

Por otro lado, *autoTable* permite la inserción de tablas en los documentos PDF.

5.4 Cuarto *Sprint*

Este último *sprint* se ha centrado en toda la funcionalidad de las competiciones. Durante el *sprint* se ha diseñado todo el sistema de inscripciones y se ha implementado para terminar con un primer producto usable.

Durante el *sprint* se han desarrollado los siguientes requisitos:

- Ver orden de pruebas (Tabla 25)
- Inscribir nadador a prueba (Tabla 11)
- Ver inscripciones de "Mi Equipo" (Tabla 12)
- Ver todas las inscripciones (Tabla 13)
- Ver puntos iniciales (Tabla 23)
- Calcular mejores nadadores por pruebas (Tabla 24)
- Dar de baja (Tabla 26)

5.4.1 Ver orden de pruebas

Pruebas de aceptación:

Entrada	**Salida esperada**
El entrenador accede a la vista de orden de pruebas	Se muestra una lista con las pruebas en orden por sesión.
El entrenador selecciona una de las pruebas de la lista	El entrenador es redirigido a la vista de inscripción.

Tabla 46- Pruebas de aceptación RF22

5.4.2 Inscribir nadador a prueba

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador accede a la vista de inscripciones	Si hay inscritos: Muestra la lista con los inscritos del club del entrenador. Si no hay inscritos: Muestra una lista vacía.
No hay ningún inscrito y el entrenador selecciona 2 deportistas para inscribir	Esos 2 deportistas se inscriben en la prueba
El entrenador selecciona 2 veces al mismo deportista	Mensaje error: "No puedes inscribir al mismo nadador 2 veces"
El entrenador selecciona a más de 2 deportistas	Mensaje error: "No puedes inscribir a más de 2 nadadores en cada prueba"
Hay 1 inscrito y el entrenador selecciona a 1 deportista para inscribir	Se actualiza la lista de inscritos con los dos deportistas
Hay 1 inscrito y el entrenador selecciona al deportista inscrito para inscribir	Mensaje error: "No puedes inscribir al mismo nadador 2 veces"
Hay 1 inscrito y el entrenador selecciona 2 deportistas para inscribir	Mensaje error: "No puedes inscribir a más de 2 nadadores en cada prueba"
No hay ningún deportista inscrito y el entrenador selecciona 2 para inscribir	Se actualiza la lista de inscritos con los dos deportistas

Tabla 47- Pruebas de aceptación RF07

Desarrollo:

La principal limitación para desarrollar este requisito es que como máximo puede haber 2 inscripciones por género y prueba (2 deportistas masculinos en una prueba y 2 deportistas femeninos en la misma), con un máximo de 4 deportistas en total para cada prueba por club. Esta limitación ha derivado que se tenga que separar la mayoría de las funciones con un parámetro extra que hace referencia al equipo masculino o femenino, como se puede observar en la Ilustración 233, se pueden apreciar dos funciones, una para dar de baja el equipo masculino y otra para dar de baja al equipo femenino, pero que llaman a una función auxiliar que es la encargada realmente de dar de baja a los deportistas.

```
56  //Unsubscription of the male team
57  async function unSubscribeMale(e: any) {
58    unSubscribe(e, thisClubInscribedMale);
59  }
60
61  //Unsubscription of the female team
62  async function unSubscribeFem(e: any) {
63    unSubscribe(e, thisClubInscribedFemale);
64  }
65
66  //Wrapped function to unsubscribe a swimmer from the event
67  async function unSubscribe(e: any, myInscribed: Mark[]) {
68    e.preventDefault();
69    while (myInscribed.length > 0) {
70      await unsubscribeSwimmer(
71        Number(event[0]),
72        event[1] + "",
73        event[2] + "",
74        myInscribed[0].swimmer?._id + "",
75      );
76      myInscribed.shift();
77    }
78    setInscribed(!inscribed);
79  }
```

Ilustración 23 - Fragmento de código, función separada de equipo masculino y equipo femenino

5.4.3 Dar de baja a nadador

Pruebas de aceptación:

Entrada	Salida esperada
Hay al menos 1 inscrito en la prueba	La opción “Dar de baja” está habilitada
No hay inscritos en la prueba	La opción “Dar de baja” está deshabilitada
Selecciona la opción “Dar de baja”	Los deportistas en la prueba son todos dados de baja (ya no están inscritos)

Tabla 48- Pruebas de aceptación RF23

5.4.4 Ver inscripciones de “Mi Equipo”

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador selecciona una sesión	Si hay inscritos: Muestra la lista con los inscritos del club del entrenador. Si no hay inscritos: Muestra una lista vacía.

Tabla 49- Pruebas de aceptación RF08

5.4.5 Ver todas las inscripciones

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador selecciona una sesión	Para cada prueba: • Si hay inscritos: Muestra la lista con los inscritos con el nombre del club al lado del nadador. • Si no hay inscritos: Muestra una lista vacía.
En una prueba hay inscritos con marca acreditada	Los deportistas inscritos en la prueba aparecen ordenados en una lista por orden de tiempo
En una prueba hay inscritos sin marca acreditada	Los deportistas inscritos aparecen al final con marca 59’ 59” 59

Tabla 50- Pruebas de aceptación RF09

Desarrollo:

Para este requisito, el entrenador tiene acceso a las inscripciones de todos los clubs, de manera que los deportistas quedan ordenados por sus respectivas marcas en la prueba correspondiente (el primero es el más rápido, y por último los que más tiempo tienen o no han nadado aun la prueba, NT significa No Tiempo). Se ha tenido que realizar la recogida de todas las inscripciones de una prueba de un campeonato, y ordenar esas inscripciones por tiempo. El algoritmo utilizado para realizar la ordenación es el *insertion sort*.

5.4.6 Ver puntos iniciales

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador accede a esta vista de una competición sin inscritos	Aparece una tabla para el equipo masculino y otra para el equipo femenino con 0 puntos en el equipo del entrenador.
El entrenador tiene inscrito solamente a componentes del equipo masculino	La tabla muestra puntuación en el equipo masculino, pero no en el equipo femenino
El entrenador tiene inscrito solamente a componentes del equipo femenino	La tabla muestra puntuación en el equipo femenino, pero no en el equipo masculino
El entrenador tiene inscrito a componentes tanto del equipo masculino como del equipo femenino	La tabla muestra puntuación en ambos equipos

Tabla 51- Pruebas de aceptación RF20

Desarrollo:

A la hora de realizar la suma de los puntos, se ha utilizado el sistema que utiliza la Federación Valenciana de Natación, el cual otorga 19 puntos al primer clasificado, 16 puntos al segundo clasificado, y a partir del 3r puesto se utiliza la fórmula:

$$Puntos = 17 - posición$$

Para reducir comparaciones costosas (el 3r clasificado obtiene 17-3=14 puntos, el 4º clasificado 17-4=13 puntos y así sucesivamente).

5.4.7 Calcular mejores nadadores por prueba

Pruebas de aceptación:

Entrada	Salida esperada
El entrenador selecciona una prueba	Si no hay deportistas del equipo con marca, muestra una lista vacía Si hay deportistas del equipo con marca, muestra una lista con todos ellos, además de mostrar los 2 más rápidos
En entrenador acepta cambiar los inscritos	Si no había nadadores inscritos, inscribe a los deportistas que aparece en la lista Si había nadadores inscritos, se dan de baja los que estaban anteriormente inscritos, y se inscriben los dos deportistas más rápidos

Tabla 52- Pruebas de aceptación RF21

Desarrollo:

Para este requisito, se muestra las inscripciones del equipo, para compararlas con las marcas de otros integrantes del equipo y obtener a los dos nadadores más rápidos para esa prueba. Además de la complejidad de encontrar a los dos nadadores más veloces en la prueba seleccionada, se debe tener en cuenta que, idealmente, los nadadores deberían nadar una única prueba por cada sesión, para no incrementar su fatiga y evitar que puedan dar unos resultados peores a los esperados. A su vez, cuando se acepta realizar el cambio de inscripción, se comprueba que uno de los dos deportistas más veloces no esté ya inscrito, para así evitar realizar operaciones extras que son innecesarias.

6. Pruebas

Probar el *software* es algo fundamental para poder descubrir *bugs* y asegurar una cierta fiabilidad y seguridad cuando se vaya a utilizar fuera del entorno de desarrollo.

Se han preparado distintos tipos de prueba para probar la aplicación, empezando por las pruebas de aceptación, para verificar que el requisito deseado se ha terminado de implementar, posteriormente pruebas unitarias, para verificar que cada operación por separado funcionaba correctamente.

6.1 Pruebas manuales

Las pruebas manuales son las realizadas al finalizar el desarrollo de un requisito para asegurar que este está finalizado de acuerdo con las especificaciones dadas por el cliente. Este tipo de pruebas son del tipo de caja negra, ya que para probar el comportamiento de la aplicación no se necesita tener acceso al código fuente, se prueba la aplicación mediante la interacción del usuario con la interfaz del sistema.

Estas pruebas son muy limitadas si no se realizan de forma exhaustiva, ya que, al hacerlas manualmente, el *tester* tiene que esperar a los tiempos de respuesta de la aplicación para cada función y pueden ser costosas en el tiempo. A pesar de esto, son las más rápidas y fáciles de ejecutar debido a que se pueden realizar desde la misma aplicación en desarrollo sin necesidad de una nueva tecnología.

El principal problema de este tipo de pruebas es que realiza la manipulación de los datos directamente sobre la base de datos y puede dar lugar a generar datos incongruentes, como en este caso, tener varios deportistas en un club con nombres muy parecidos (Pablo, Palbo, Pabol) o nombres inexistentes (asd, fsda, qwer, rewq) para rellenar rápidamente los formularios.

En este caso, las pruebas manuales han sido sencillas, por lo que en la mayoría de los casos no se prueban casos límite entre comportamientos aceptables y no aceptables.

6.2 Pruebas unitarias

Para realizar las pruebas unitarias se ha utilizado la herramienta *Jest*, que es una tecnología de *testing* para JavaScript.

Se ha creado un archivo "*entidad.test.ts*" para cada entidad del sistema, como se muestra en la Ilustración 24. En ella podemos observar que hay 6 archivos distintos de pruebas.

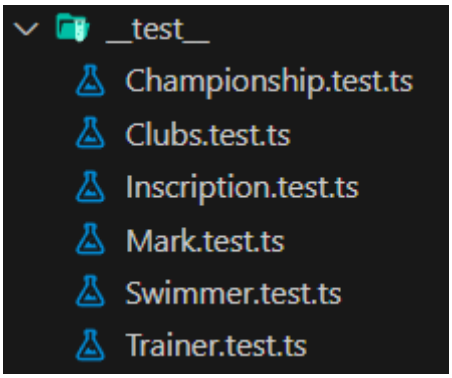


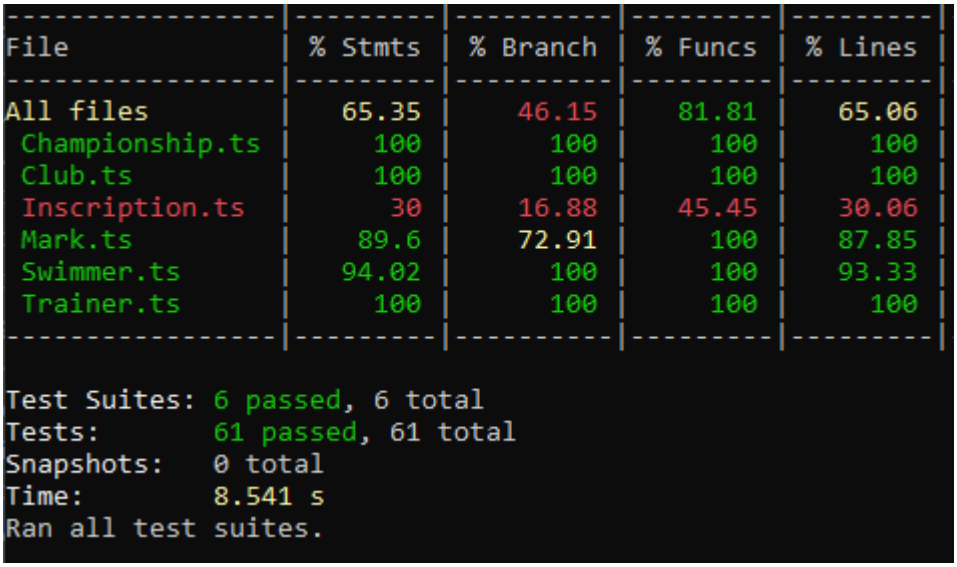
Ilustración 24 - Carpeta de pruebas

Estas son pruebas automatizadas de tipo caja blanca (tiene acceso al código fuente) que permiten simular la ejecución de la aplicación para comprobar el comportamiento esperado de la aplicación.

Para cada archivo se han planificado y codificado las pruebas necesarias para probar la mayor cantidad posible de casos de prueba, alcanzando como mínimo el 70% de las ramas de ejecución detectadas por *Jest*. Aunque esto no asegura la ausencia de *bugs* o errores en el código, ayuda a asegurar que los casos más típicos tienen un rendimiento esperado. Para buscar diferentes casos de prueba se ha utilizado la herramienta de inteligencia artificial *CodiumAI*. Esta herramienta se ha instalado en el editor de texto y se ha ejecutado para cada función que se ha probado con *Jest*.

Como se puede observar en la Ilustración 25, en la mayoría de los archivos se superan los valores recomendados, (100% en "*Championship.ts*", "*Club.ts*" y "*Trainer.ts*"), pero en "*Inscription.ts*" este valor es menor debido a la complejidad de las operaciones.

Para evitar que este archivo se quedara sin probar, se han realizado pruebas manuales extra sobre las funciones desarrolladas.



File	% Stmts	% Branch	% Funcs	% Lines
All files	65.35	46.15	81.81	65.06
Championship.ts	100	100	100	100
Club.ts	100	100	100	100
Inscription.ts	30	16.88	45.45	30.06
Mark.ts	89.6	72.91	100	87.85
Swimmer.ts	94.02	100	100	93.33
Trainer.ts	100	100	100	100

Test Suites: 6 passed, 6 total
Tests: 61 passed, 61 total
Snapshots: 0 total
Time: 8.541 s
Ran all test suites.

Ilustración 25 - Cobertura de pruebas automáticas

7. Conclusiones

Como conclusiones, comparando el resultado obtenido con los objetivos planteados al inicio del desarrollo de este proyecto, se considera que estos se han alcanzado satisfactoriamente. El principal objetivo era poder presentar un *MVP* que permitiera calcular el mejor equipo para aspirar a la mayor cantidad de puntos posible en una competición. A falta de realizar una prueba con datos reales, los resultados con datos de prueba son satisfactorios, permitiendo obtener la mejor lista teórica, mostrando a su vez la cantidad de puntos que se podrían obtener en cada sesión de la competición. Además de esta funcionalidad, se ha conseguido mostrar al entrenador una gran cantidad de información con respecto a su equipo y sus deportistas, aportándole información de valor.

Para desarrollar esta aplicación se ha profundizado sobre el lenguaje de programación *JavaScript* y su *superset TypeScript* utilizando el *framework* de *NextJS*. Utilizar *TypeScript* ha ayudado a realizar código más útil evitando realizar operaciones de un tipo sobre objetos que no las soportan. Utilizar este *framework* basado en *ReactJS* ha facilitado muchas tareas que vienen integradas en el mismo, y que se habrían tenido que implementar a mano si no se hubiera escogido como opción para el desarrollo.

Así mismo, se ha optado por utilizar una base de datos no relacional como es *MongoDB*, que permite mayor libertad a la hora de guardar datos, pero junto a *TypeScript* se ha mantenido una consistencia en la estructura de los tipos en todas las capas.

El proyecto ha podido llegar a buen puerto gracias al análisis de la documentación del *framework* y la cantidad de artículos, medios digitales y físicos disponibles, además de una gran abundancia de ejemplos para ayudar a la comprensión de estos conceptos.

7.1 Relación del trabajo con los estudios cursados

Para la realización de este trabajo han resultado imprescindibles los conocimientos obtenidos durante el Grado de Ingeniería Informática, principalmente los presentados en las asignaturas de la rama de *Ingeniería de Software*. Las principales asignaturas han sido las siguientes:

Para escoger la metodología óptima de entre todas las metodologías *software* disponibles, la asignatura *Proceso de Software* y *Proyecto de Ingeniería de Software* ha presentado una gran variedad de ellas mostrando sus ventajas y desventajas, explicando principalmente metodologías ágiles basadas en *Scrum* y *Kanban*.

Siguiendo con las fases del desarrollo, la asignatura *Análisis y Especificación de Requisitos* ha proporcionado muchas herramientas de elicitación y especificación como el estándar IEEE 830.

Para realizar un código reutilizable, limpio y entendible, se han utilizado conocimientos de la asignatura *Diseño de Software* y de *Mantenimiento y evolución de software*.

Gracias a la asignatura *Ingeniería de Software* se ha planificado una arquitectura por capas que ha sido muy sencilla de aplicar en el proyecto.

Por último, para poder realizar unas pruebas de garantía para asegurar el correcto funcionamiento del sistema, han sido fundamentales los conocimientos obtenidos en la asignatura *Análisis, Validación y Depuración de Software*.

7.2 Trabajo futuro

Durante el transcurso de este trabajo, se ha buscado principalmente la solución a los problemas mencionados. A pesar de esto, han surgido nuevos problemas durante el desarrollo que no han podido ser abordados para esta solución.

El principal problema es la falta de datos que se manejan al principio. Antes de que los primeros usuarios reales entren en contacto con la aplicación, la base de datos estaría vacía, por lo que inicialmente los entrenadores tendrían que completarla con los datos propios obtenidos por sus propios medios desde su federación regional (en este caso la Federación Valenciana de Natación). De esta base surgen dos posibles soluciones. La primera consiste en conectar la aplicación con la *API* de Leverade que tiene acceso a los datos oficiales de todos los deportistas y equipos, a su vez esta solución permitiría tener datos en tiempo real cuando se actualice la aplicación de Leverade. La otra opción, sería generar un segundo usuario de tipo entrenador, que fuera el auxiliar encargado de manipular los datos desde la aplicación, mientras que el usuario entrenador estaría encargado de consultar los datos.

En caso de escoger esta segunda opción con más tipos de usuario, habría que implementar un sistema de concurrencia para evitar las condiciones de carrera en la base de datos *MongoDB* y mantener los datos consistentes en todo momento.

Por último, inicialmente se pensó la aplicación para que hubiera un único equipo al cargo del entrenador, siendo todos los componentes de este equipo de la misma categoría. Además, podría ampliarse esta limitación y que el entrenador seleccionara la vista para cambiar entre categorías.

8. Bibliografía

[1] M. Fowler, UML Distilled: A brief Guide to the Standard Object Modeling, Addison-Wesley Professional, 2003.

[2] IEEE, Recommended Practice for Software Requirements SpeciÞcations, 1998.

[3] Microsoft, 29 04 2015. [En línea]. Available: https://code.visualstudio.com/.

[4] L. T. Junio Hamano, «Git,» 7 4 2005. [En línea]. Available: https://git-scm.com/.

[5] Inc, Figma, 2020. [En línea]. Available: https://www.figma.com.

[6] Inc, Lucid Software, 2008. [En línea]. Available: https://www.lucidchart.com/pages/.

[7] B. R. Raymond Camden, Jamstack Book, The: Beyond static sites with JavaScript, APIs, and Markup, Manning Publications, 2022.

[8] J. Goldberg, Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript, O'Reilly Media, 2022.

[9] N. Rappin, Modern CSS with Tailwind: Flexible Styling without the Fuss, Pragmatic Bookshelf, 2021.

[10] A. O. Gaía, «G-SE,» 30 10 2013. [En línea]. Available: https://g-se.com/uso-de-la-velocidad-critica-para-el-entrenamiento-de-la-resistencia-aerobica-en-nadadores-jovenes-bp-Y57cfb26d6195d.

9. Anexo

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**	**X**			
ODS 4. **Educación de calidad.**		**X**		
ODS 5. **Igualdad de género.**		**X**		
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**		**X**		
ODS 13. **Acción por el clima.**			**X**	
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Tabla 53 - Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS)

Los ODS abordan problemas sociales, económicos y ambientales que existen hoy en día. En la búsqueda de un mundo completamente digitalizado. Las aplicaciones informáticas juegan un papel importante para suplir los recursos utilizados anteriormente, tales como el papel, y aumentar el rendimiento para obtener los resultados con celeridad. Este TFG es un claro ejemplo de que el mundo informático y los ODS pueden convivir, promoviendo, además, un estilo de vida saludable.

La relación entre el ODS 3, “Salud y bienestar”, y el TFG realizado es, sobre todo, promover la práctica de este deporte en unos rangos de esfuerzo saludables para permitir a los deportistas estar en un buen estado de salud. Además, realizar actividad física facilita la buena salud mental, consiguiendo de esta forma un mayor bienestar general. También fomentar una competitividad sana ayuda a mejorar tanto la salud física como la mental, rodeado de un ambiente sano.

El ODS número 4 quiere promover una educación de calidad. Con la aplicación desarrollada durante este TFG, se pretende que haya un ambiente sano de competición, para disfrutar el ambiente competitivo y mejorar el rendimiento como nadadores. Este entorno sano se consigue a través de una educación adecuada que otorgue unos valores correctos a los deportistas. Además de educar sobre el ambiente de competición, el mundo del deporte también aporta una buena educación con valores y conocimiento sobre la natación.

Con respecto a la igualdad de género, el ODS número 5, se refleja igualmente. Históricamente, ha habido barreras sociales y culturales que las mujeres han tenido que enfrentar para realizar actividades en piscina. Con la aplicación desarrollada, los entrenadores observan que para obtener buenos resultados en las competiciones se necesita un buen equipo femenino tanto como un buen equipo masculino, de forma que, si tu equipo masculino consigue muchos puntos, pero tu equipo femenino no está a la altura por diferencias en los entrenamientos, el resultado global del equipo va a ser inferior. Así, se promueve la participación de la mujer en el mundo de la natación.

Además, el principal objetivo de este TFG es el de encontrar la mejor lista para optimizar la cantidad de puntos obtenidos en la competición. Para realizar esta tarea de manera manual, los entrenadores debían hacer uso de mucho papel para probar diferentes combinaciones y realizar el cálculo de puntos a mano, además de necesitar disponer de mucho tiempo para escribir las combinaciones y realizar correctamente los cálculos. Este problema es el relacionado directamente con el ODS 12, producción y consumo responsable, así, con una menor cantidad de tiempo y recursos se puede obtener un resultado mejor.

Por último, la utilización de diferentes algoritmos de ordenación se relaciona con el ODS 13, “Acción por el clima”. Aunque no se vea una relación directa, el uso de algoritmos óptimos para la ordenación de los deportistas por su tiempo en una determinada prueba evita que se realicen operaciones extra, que, a pesar de no ser muy costosas, su repetición a lo largo del tiempo puede generar un consumo de recursos excesivo, acarreando un consumo eléctrico superior al necesario y por ende aumentando la huella de carbono. De esta forma utilizar algoritmos de ordenación con un buen rendimiento puede reducir el consumo eléctrico.

En resumen, la realización de este TFG abarca ODS que afectan a priori sectores muy dispersos. Desde cierto punto de vista, se puede calificar este trabajo como una herramienta muy útil para avanzar en la consecución de estos objetivos, al centrarse en la mejoría de la salud fomentando la actividad física mejorando la salud y bienestar de estas personas. Aunque no sea la principal motivación, este

proyecto también ayuda a conseguir un mejor rendimiento a los recursos explotados, realizando un consumo responsable y eficaz de estos.