UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Aplicación web para la gestión de clubes de fútbol

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Lopez Berges, Juan

Tutor/a: Canós Cerdá, José Hilario

CURSO ACADÉMICO: 2021/2022

Resumen

Toach App es una aplicación web enfocada a la gestión deportiva y administrativa para clubes de fútbol. Permite a la directiva del club coordinar los equipos y sus entrenadores. A su vez, el entrenador será la figura principal de esta aplicación; permitiéndole organizar e individualizar los entrenamientos según las necesidades del equipo. Sin embargo, el receptor final de esta aplicación es el usuario, que en este caso será el jugador del equipo, que tiene acceso a un calendario personalizado, videos explicativos y más funcionalidades. Se intenta conseguir una mejora en el uso de la tecnología facilitando la accesibilidad a este tipo de aplicaciones, por tanto, se puede adaptar a cualquier dispositivo móvil.

Palabras clave: Deporte; Fútbol; Aplicación web; Metodología ágil; TDD; Proceso software.

Abstract

Toach App is a web application focused on sports and administrative management for football clubs. It allows the club's management to coordinate the teams and their coaches. In turn, the coach will be the main figure of this application; allowing him to organise and individualise training sessions according to the needs of the team. However, the final receiver of this application is the user, who in this case will be the team's player, who has access to a personalised calendar, explanatory videos and more functionalities. The aim is to achieve an improvement in the use of technology by facilitating accessibility to this type of applications, therefore, it can be adapted to any mobile device.

Keywords : Sport; Football; Agile methodology; TDD; Web app; Software process.

Tabla de contenidos

Tabla de imagenes

1. Introducción

A lo largo de la historia, la tecnología ha jugado un papel primordial en el desarrollo de la vida material y en la cultura de la humanidad [1]. La tecnología es el conjunto de conocimientos y técnicas que, aplicados de forma lógica y ordenada, modifican el entorno material o virtual del ser humano para satisfacer sus necesidades. Es un proceso combinado de pensamiento y acción con la finalidad de crear soluciones útiles [2].

La incorporación de las Tecnologías de la Información y la Comunicación (TIC) en los diferentes contextos de la vida del ser humano se ha incrementado considerablemente en las últimas décadas, y el ámbito educativo y deportivo no es la excepción [3].

1.1 Objetivo

El objetivo de este trabajo consiste en realizar una aplicación web adaptable a dispositivos móviles, dirigida a los clubes de fútbol, en la que se permite gestionar un club de una manera minimalista y eficaz. Está enfocada a la directiva y a los entrenadores, quienes pueden realizar el seguimiento de su equipo de fútbol, evitando tener que transportar folios, bolígrafos y todo tipo de material para apuntar. Los jugadores también pueden acceder a la aplicación donde se les mostrará una serie de características para poder ver su seguimiento deportivo.

1.2 Metodología

Para el presente trabajo se ha utilizado la metodología del desarrollo ágil de software el cual envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería basados en el desarrollo iterativo e incremental. Los requisitos y soluciones en este tipo de metodología evolucionan con el tiempo según la necesidad del proyecto [4].

Dentro de estas metodologías existen varias propuestas, de las cuales se ha utilizado el método *Scrum.* Este término hace referencia a un proceso en el que se aplican de manera regular un conjunto de prácticas para trabajar en equipo y obtener el mejor resultado posible para el proyecto [5]. En este proceso se necesita:

- Indicar el tipo de proyecto para obtener resultados pronto, ya que suelen tener requisitos cambiantes o poco definidos.

- Realizar entregas parciales y regulares del producto final, priorizando el beneficio que aportan al receptor del proyecto.

- Conseguir innovación, flexibilidad, competitividad y productividad [5, 6].

- El desarrollo de la aplicación web va unida al proceso de software TDD (Desarrollo Dirigido por Tests), el cual permite tener un código bien testeado desde el inicio.

1.3 Estructura del documento

En este apartado se realiza una visión general de los apartados que se van a ir desarrollando a lo largo del documento.

En el segundo capítulo se realizará un análisis de la competencia previo al comienzo del desarrollo de la aplicación. Este análisis, permite observar las características y los puntos débiles que tienen las distintas aplicaciones existentes.

Tras establecer las características, se realizará un análisis del problema junto con la creación de los distintos diagramas y su especificación.

El capítulo número 4 se dedicará a mostrar el diseño de la solución, mostrando la arquitectura del sistema, el diseño detallado y la tecnología empleada para el desarrollo de la aplicación. El siguiente capítulo tratará sobre la implementación, dividido en los distintos niveles que componen la aplicación. Para finalizar con la parte de desarrollo, se encuentra el capítulo de pruebas, donde hablaremos de todos los tipos de pruebas realizadas en el desarrollo.

En el capítulo número 7 se hablará sobre las conclusiones de este trabajo, relacionándolo con las asignaturas cursadas en el grado de Ingeniería Informática. En el capítulo número 8 se comentará el posible futuro de la aplicación. Los últimos capítulos tratan de la bibliografía del documento y un glosario con las palabras difíciles de comprender. Por último, para complementar el contenido se encuentran los anexos.

2. Estado del arte

Antes del auge tecnológico de la última década, los entrenadores de clubes deportivos estaban obligados a llevar todo el seguimiento y planificación de los entrenamientos en papel y pizarra. No se podían transportar los ordenadores a los campos de fútbol.

A medida que las *tablets* y los *smartphones* han ido madurando y aumentando su capacidad, se ha apostado por las aplicaciones en las que se puede gestionar su uso desde dispositivos móviles. Por tanto, gracias a estos dispositivos, los entrenadores pueden realizar el seguimiento desde el campo de fútbol, lo que permite tomar apuntes y jugadas rápidas, acceder a los datos necesarios, etc. Para el desarrollo de este proyecto, se ha realizado un estudio de las aplicaciones competidoras, documentando las más importantes del mercado. Se puede observar que el número de aplicaciones en el mercado no es muy grande. Mediante un análisis sobre cada una de ellas, se obtienen las características principales que debe obtener la aplicación de este proyecto.

A continuación, para cada herramienta se establece una breve descripción, así como una enumeración de las ventajas e inconvenientes de la misma. Finalmente, se realiza una propuesta de mejora a las aplicaciones analizadas, la cual se desarrolla en el resto del proyecto.

2.1.1 Gesdep

- Empresa: Gestión Deportiva Consultores, S.L. [7]

Gesdep es la aplicación pionera deportiva pionera en España en la gestión de clubes deportivos y la única con la que puede gestionar toda la actividad de su club, tanto a nivel deportivo como administrativo. Esta aplicación cuenta con una gran cantidad de clientes importantes, como por ejemplo *Valencia CF,* y se utiliza en escuelas prestigiosas como *Barça Academy*. Esta aplicación web ofrece grandes funcionalidades, dando al entrenador un mayor control sobre el club [7, 8].

Las funcionalidades se encuentran organizadas en menús y submenús. En el apartado de “Departamento técnico” que se muestra en la *Ilustración 1*, se encuentra la gestión de jugadores. Esta permite aglutinar y manejar diversa información sobre los mismos. Permite, además, la gestión de competiciones, encontrándose todas las funcionalidades centradas en las convocatorias, partidos disputados, etc. Igualmente, permite planificar, diseñar y gestionar entrenamientos. Existe un apartado donde se pueden gestionar los partes médicos, casos COVID y más funcionalidades relacionadas con la gestión médica. En cuanto a la administración del club, es posible gestionar los cobros, los documentos, los suministros, etc. [8]

Gesdep permite gestionar el personal del club, pudiendo observar en tan solo un vistazo todos los entrenadores y delegados que se encuentran activos. Otras

características destacadas son: la generación de estadísticas, la vista del calendario, y la mensajería.

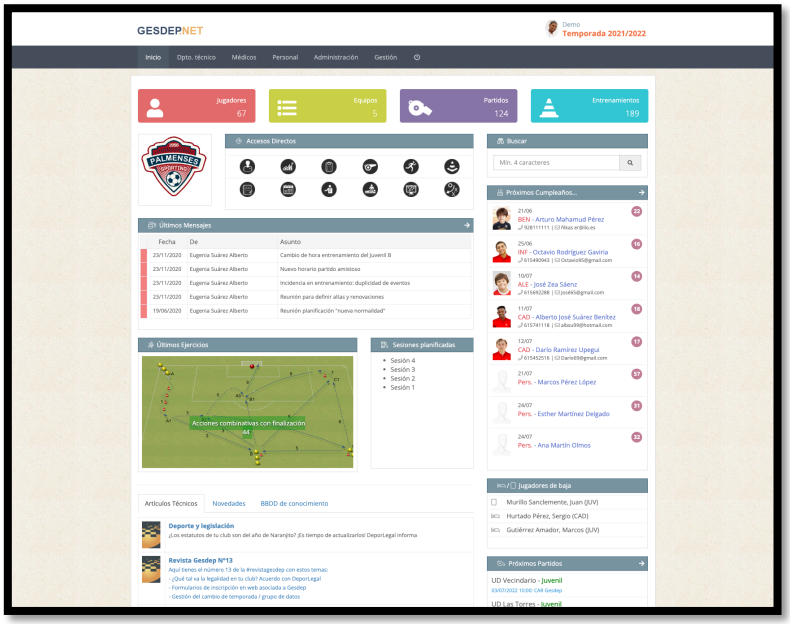

Figura 1 Página principal Gesdep.

Ventajas:

- Presenta gran cantidad de funcionalidades que permiten cubrir las necesidades del usuario.
- Tiene una alta personalización permitiendo personalizar el panel principal con distintos widgets.
- La interfaz de usuario de la aplicación se adapta a todo tipo de dispositivo.

Desventajas:

- La interfaz de la aplicación se encuentra desfasada ya que utiliza componentes antiguos.

2.1.2 Easy2Coach

- Empresa: Gestión Deportiva Consultores, S.L [9]

Easy2coach es una aplicación para la gestión de un club de fútbol. La empresa Gestión Deportiva Consultores desarrolla *software* y aplicaciones de entrenamiento de futbol para todas las edades y niveles de habilidad. Tiene como clientes: el club Borussia M'gladbach, Hamburgo entre muchos otros.

En cuanto a las funcionalidades, Easy2Coach presenta bastantes, y se encuentran relacionadas con los entrenadores. Permite la gestión de jugadores del mismo equipo. También ofrece una gestión básica de competiciones y entrenamientos, pero, solo se pueden crear y observar. Se puede disponer además de una gestión más básica del club, generación de estadísticas, gestión de asistencia de los entrenamientos y un buzón de mensajería.

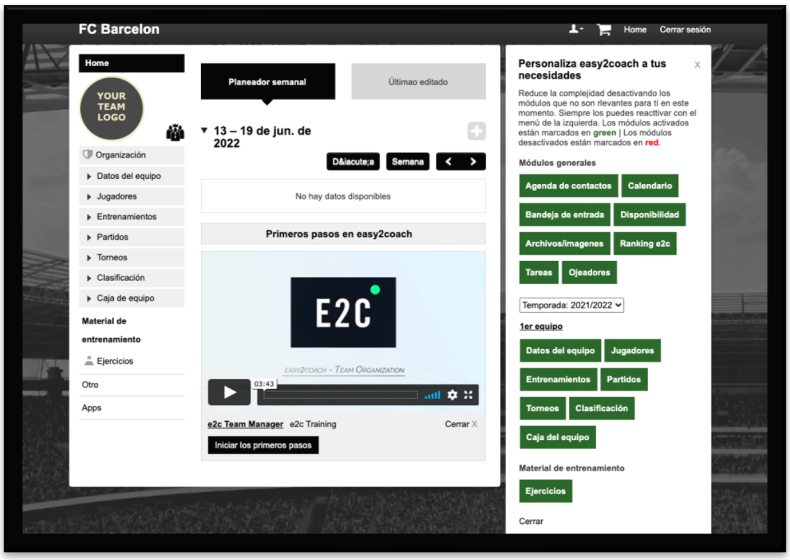

Figura 2 Página principal Easy2Coach.

Ventajas:

- Gran cantidad de funcionalidades que permiten cubrir las necesidades del usuario.

Desventajas:

- La interfaz de la aplicación está desfasada.
- La sobrecarga de menús y submenús provoca un aprendizaje más lento y una mala experiencia de uso.

- Se encuentran palabras mal traducidas al idioma español
- No es *responsive,* es decir, esta aplicación no realiza un diseño web adaptable a todo tipo de dispositivo.

2.1.3 Sportlyzer

- Empresa: Sportlyzer LLC [10]

Se trata de una aplicación con más de 7 años en el sector que permite la gestión de un club de fútbol. Es la ganadora del premio “Mejor tecnología para entrenadores” en la ceremonia de premios *Sports Technology Awards 2016* [11].

Sportlyzer ofrece una visión general de los jugadores, permitiendo realizar diferentes acciones con ellos, como por ejemplo la preparación de entrenamientos enfocados en un jugador. Tiene una buena integración del calendario, ya que se pueden añadir nuevas actividades. Lo más destacable de la aplicación es el apartado de *Smart coaching,* donde el entrenador puede personalizar las intensidades del entrenamiento observando la carga semanal de trabajo que llevan los jugadores. La aplicación tiene un apartado de pruebas, en el cual el entrenador puede apuntar los resultados de evaluación de los jugadores, como, por ejemplo, pruebas de aptitud física. Otro tipo de características: la generación de estadísticas, el buzón de mensajería, los ajustes del club mediante la visión de director [10].

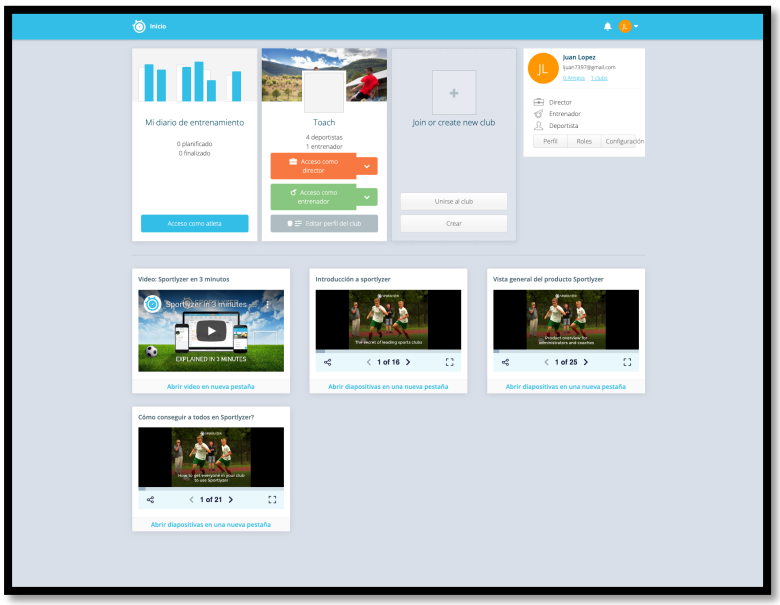

Figura 3 Página principal Sportlyzer.

Ventajas:

- Permite la personalización de la página principal con widgets basados en las funcionalidades.

- Aplicación fácil de usar, con menús intuitivos
- La aplicación se encuentra en las distintas tiendas de aplicaciones, adaptada a todos dispositivos.

Desventajas:

- Interfaces con errores, textos fuera de botones y malas traducciones
- Valoración negativa en appStore debido a la traducción en español.

2.1.4 SportEasy

Empresa: SportEasy [12]

SportEasy es una aplicación multidispositivo francesa con más de 1.8M de usuarios, un equipo de 30 empleados y 1000 clubes a sus espaldas como clientes. Esta aplicación está empezando a comercializarse de manera más directa en España, siendo este el segundo país con más clientes seguido de Francia.

SportEasy presenta una serie de características como por ejemplo el calendario que ofrece, siendo este muy completo e intuitivo donde se pueden crear eventos detallados y de manera sencilla. Existe la opción de crear nuevas competiciones dentro de la aplicación. Se puede gestionar la plantilla del equipo permitiendo observar en detalle a cada jugador que la forma. El entrenador puede gestionar convocatorias y comprobar la disponibilidad de jugadores, pudiendo comunicarse con ellos. Además, permite visualizar y trabajar con estadísticas [13].

Cabe destacar además la gestión de asistencia, la generación de estadísticas, el buzón de mensajería, el cobro de los jugadores y los ajustes de club, como detalles importantes.

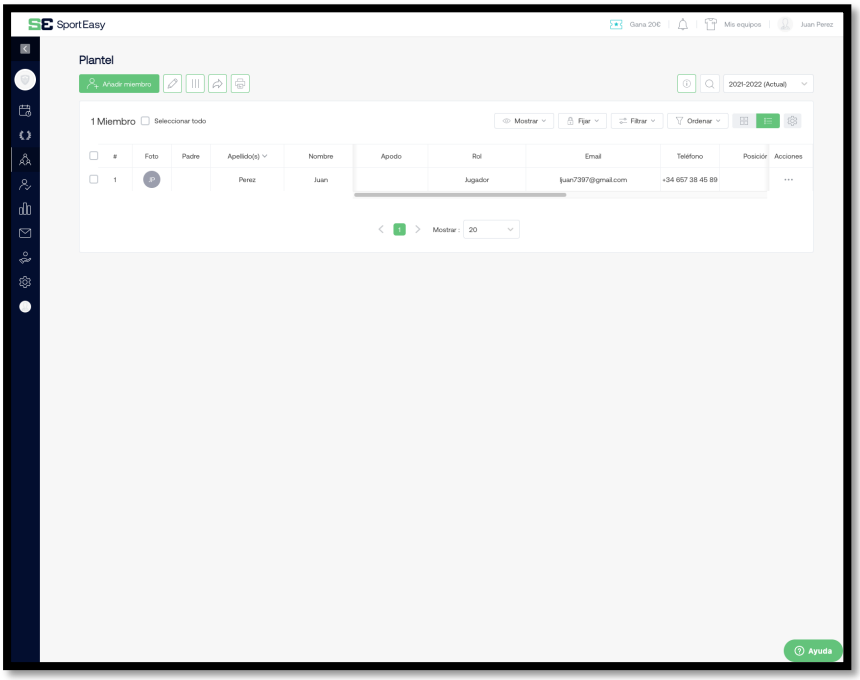

Figura 4 Página principal SportEasy.

Ventajas:

- La estética de la aplicación es muy buena empleando componentes de *frameworks* actuales.
- La aplicación se encuentra en las distintas tiendas de aplicaciones, adaptada a todos dispositivos.

Desventajas:

- Poca personalización de la página principal.

2.2 Propuesta y resumen de la competencia

Después de realizar el análisis de la competencia, podemos observar carencias muy grandes en cuanto a calidad de interfaces. Estas aplicaciones no utilizan nuevas tecnologías para la creación del diseño de la web.

Las aplicaciones de la competencia destacan por la cantidad de funcionalidades que tienen, siendo la mayoría innecesarias para un entrenador de fútbol. Sería recomendable reducir estas funcionalidades o por lo menos llevarlas a un panel más escondido, de esta manera conseguimos simplificar la vista principal del entrenador, el cual en medio de un partido no quiere perderse por los largos menús de la aplicación.

Después de realizar pequeñas reuniones con entrenadores de clubes deportivos, se llegó a la conclusión que todos prefieren una estética minimalista antes que una aplicación sobrecargada de widgets.

El equipo de desarrollo pretende realizar una aplicación con una mejora considerable de la usabilidad y la estética de la interfaz siempre pensando en el usuario final, los entrenadores y la directiva. Para mejorar la usabilidad se pretende reducir el número de *clicks* que el entrenador debe de presionar para acceder a una funcionalidad principal, como por ejemplo, el listado de jugadores de su equipo. Además, se va a apostar por la calidad antes que por la cantidad en nuestros menús. En cuanto a la mejora de la calidad de las interfaces, se pretende usar una librería para facilitar la creación de estas. En un futuro se pretende contratar a un profesional UX UI, centrándose en diseñar un producto más profesional.

Con estas mejoras planteadas, el producto que se ofrecerá a los clubes de fútbol será mucho más visual y profesional que las otras aplicaciones del mercado.

3. Análisis del problema

El objetivo principal consiste en implementar una aplicación web en la que la directiva y los entrenadores de un club de fútbol puedan administrar sus equipos, permitiendo la gestión de jugadores, partidos, entrenamientos… Con el único objetivo de facilitar la labor de estos. Sin embargo, los límites temporales en los que se enmarca un TFG, sumado a los pocos recursos del equipo de desarrollo, ha llevado a realizar una primera versión de la aplicación, dejando para un futuro la ampliación que soporte la gestión de los partidos, más funcionalidades para la directiva y una gran mejora de las interfaces.

La directiva de un club de fútbol desea tener en un mismo sitio los datos de los entrenadores encargados de dirigir los distintos equipos. A su vez, los entrenadores necesitan acceder a la aplicación desde cualquier lugar, sobre todo en el campo de fútbol. En estos, se puede acceder, modificar y generar apuntes de una forma rápida y sencilla. Una funcionalidad indispensable de la aplicación es la gestión de los entrenamientos, llevando un registro ordenado de los mismos. Los entrenamientos pueden personalizarse, además, añadir ejercicios ya creados. De esta manera, el entrenador puede llevar control y planificación de estos.

Es importante que un entrenador tenga a mano información detallada y concreta de los jugadores, como, por ejemplo: si es una persona con problemas de atención, problemas musculares, motivos personales por los que se les permite llegar tarde, etc.

3.1 Modelo conceptual

La ilustración 5 muestra el diagrama de clases UML que resume la estructura del dominio de estudio.

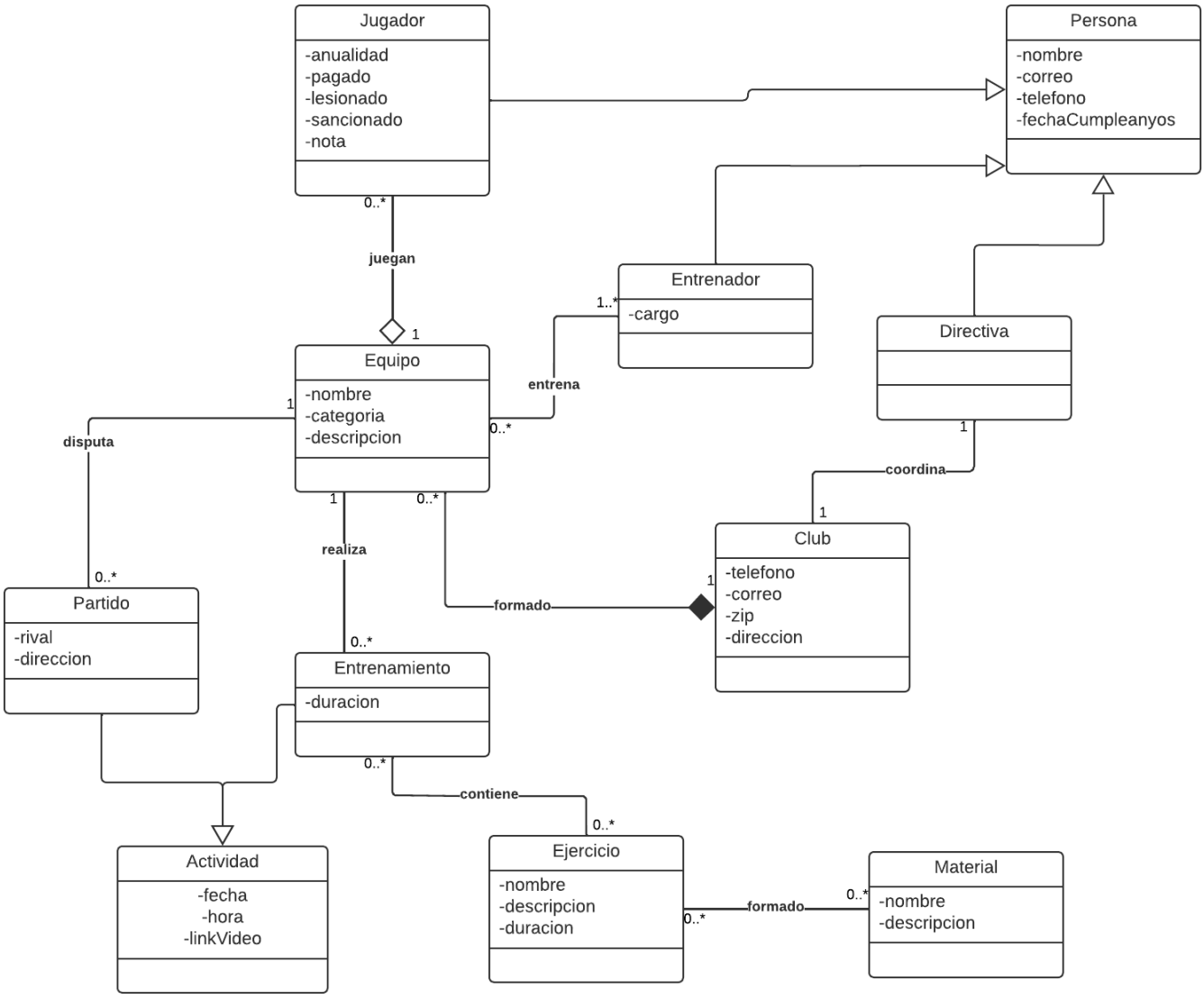

Figura 5 Diagrama de clases.

Como observamos en el diagrama, es necesario separar en distintas clases las personas que pueden acceder a la aplicación, comenzando por la directiva, quien se encarga en crear el club. Una vez creado el club la directiva ya puede crear los equipos y los entrenadores.

Un equipo debe de tener mínimo un entrenador, quién se encargará de crear los jugadores de su plantilla y darles acceso a la aplicación. Estos jugadores pertenecen al equipo creado por el entrenador.

Cada equipo realiza entrenamientos o disputa partidos. Estos tienen una fecha y hora para poder añadirlos al calendario, además de la posibilidad de añadir un video con detalles del partido o del entrenamiento. Los entrenamientos están compuestos por ejercicios, estos son creados por cualquier entrenador del mismo club y pueden ser usados por los demás entrenadores. Los ejercicios pueden incluir materiales, estos también son compartidos en todo el club, por tanto, cualquier entrenador puede hacer uso de estos.

3.2 Modelo de casos de uso

A continuación, se presenta la funcionalidad de la aplicación en la forma de un modelo de casos de uso UML. Para simplificar la vista, se ha separado cada característica de la aplicación en un diagrama de casos de uso. Las descripciones de los casos de uso mostrados en las ilustraciones 7-13 se encuentran compiladas en el Apéndice A.

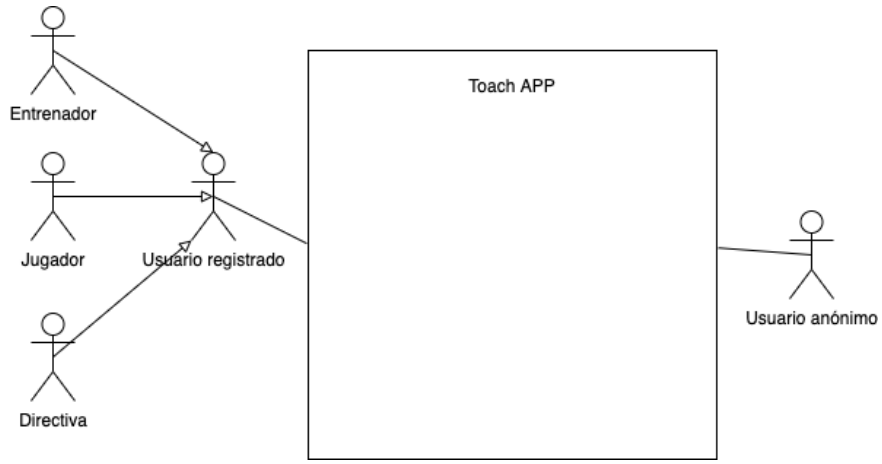

Figura 6 Diagrama de contexto.

La ilustración 6 representa el diagrama de contexto definiendo los límites del sistema y mostrando los actores que interactúan.

En cuanto al funcionamiento de la aplicación, la persona de la directiva debe registrarse en la aplicación y dar de alta al club aportando los datos necesarios. Una vez registrado, puede dar de alta a los entrenadores y crear los equipos de fútbol con sus distintas categorías.

Los entrenadores son los encargados de gestionar sus equipos, dando de alta a los jugadores que pertenecen a este.

Los jugadores pueden acceder a la aplicación para visualizar sus datos personales y los entrenamientos creados por el entrenador.

Actor	Directiva	A1
Descripción	Persona encargada de gestionar un club de fútbol.	
Características	La directiva se encarga de crear las cuentas de los entrenadores y de registrar los equipos del club.	
Referencias	Anexo 2: CU-5, CU-6, CU-7 Anexo 3: CU-8, CU-9, CU-10, CU-11	

Actor	Entrenador	A2
Descripción	Persona encargada de gestionar el equipo de fútbol.	
Características	El entrenador se encarga de gestionar los jugadores, entrenamientos, ejercicios de un equipo de fútbol.	
Referencias	Anexo 4: CU-12, CU-13, CU-14, CU-15, CU-16 Anexo 5: CU-17, CU-18, CU-19 Anexo 6: CU-20, CU-21, CU-22 Anexo 7: CU-23, CU-24, CU-25, CU-26, CU-27, CU-28, CU-29	

Actor	Jugador	A3
Descripción	Persona perteneciente a un equipo de fútbol.	
Características	El jugador pertenece a un equipo de fútbol, puede ver sus datos personales, el calendario de entrenamientos junto con los ejercicios.	
Referencias	Anexo 4: CU-14, CU-15 Anexo 6: CU-21 Anexo 7: CU-26, CU-27	

Actor	Usuario anónimo	A4
Descripción	Persona que no tiene una cuenta iniciada en la aplicación.	
Características	Interesado en iniciar sesión o crear una nueva cuenta en la aplicación para comenzar un nuevo club	
Referencias	Anexo 1: CU-1, CU-2, CU-3	

Actor	Usuario registrado	A5
Descripción	Persona tiene una cuenta iniciada en la aplicación.	
Características		
Referencias	Anexo 1: CU-4	

Acceso a la aplicación

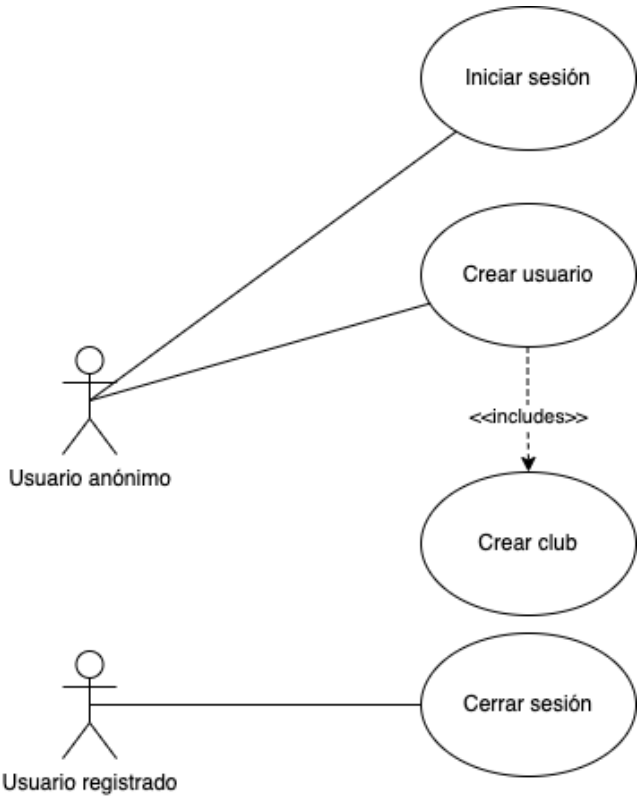

Figura 7 Diagrama casos de uso (acceso a la aplicación).

Un usuario anónimo que desea entrar en la aplicación inicia sesión CU-3 (Anexo 1) o crea una nueva cuenta de directiva junto con un club (CU-1 y CU-2). No puede haber una directiva que no tenga un club asociado. Un usuario ya registrado en el sistema puede cerrar la sesión (CU-4).

Gestión de equipos

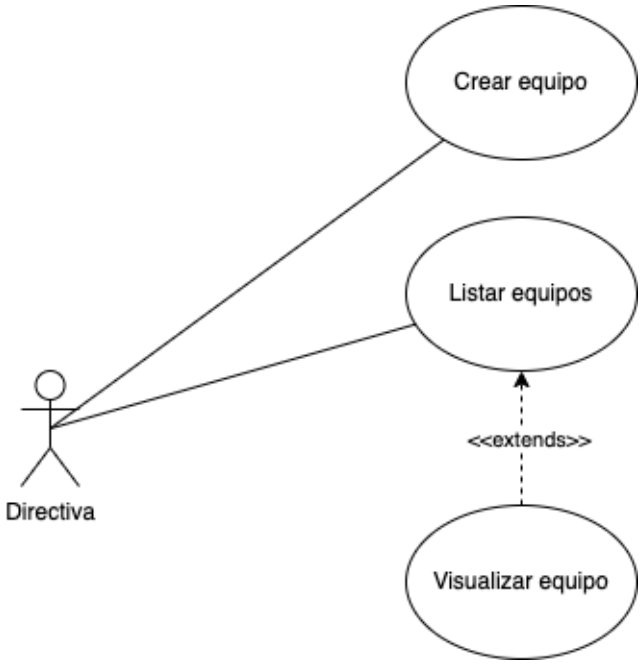

Figura 8 Diagrama casos de uso (Gestión de equipos).

La directiva se encarga de crear los equipos que pertenecen a su club de fútbol, como se muestra en la ilustración 8 en el CU-5 (Anexo 2). Puede visualizarlos en forma de listado (CU-6) y observar en detalle estos (CU-7).

Gestión de entrenadores

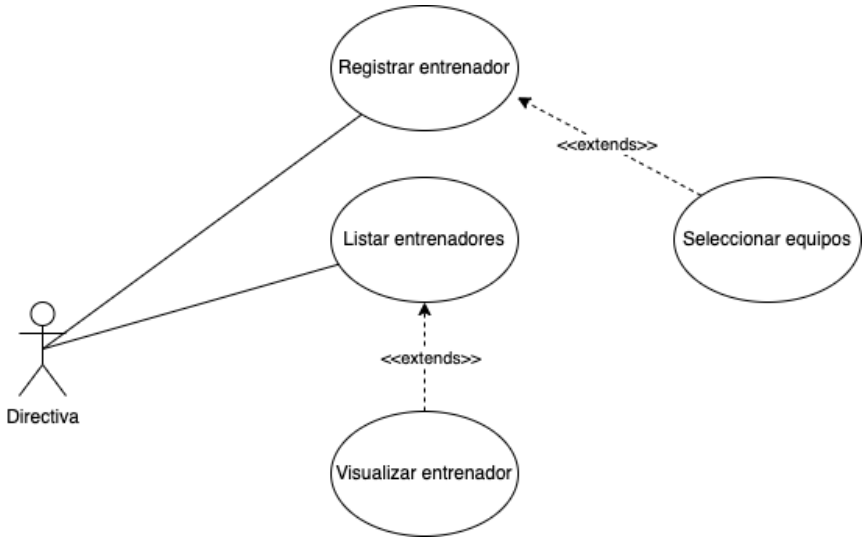

Figura 9 Diagrama casos de uso (Gestión de entrenadores).

La directiva se encarga de registrar los entrenadores, como se muestra en la ilustración 9 en el CU-8 (Anexo 3), en el registro selecciona el o los equipos que entrena (CU-9). La directiva observa el listado de entrenadores creados (CU-10) y los detalles de estos (CU-11).

Gestión de jugadores

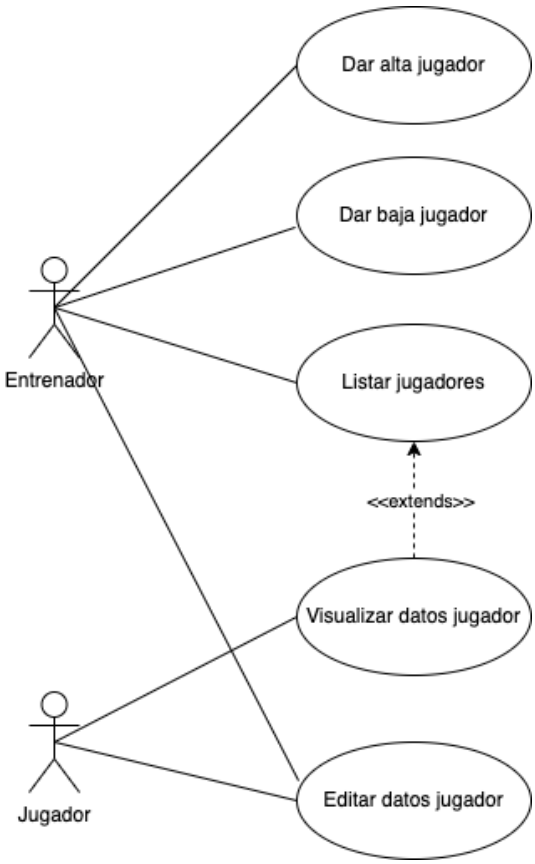

Figura 10 Diagrama casos de uso (gestión de jugadores).

El entrenador del equipo es el encargado en dar de alta a los jugadores, como se muestra en la ilustración 10 en el CU-12 (Anexo 4). Los jugadores están asignados a un equipo, siendo estos los que juegan el partido o realizan el entrenamiento creado por el entrenador. El entrenador puede visualizar en forma de listado todos los jugadores de su equipo (CU-16), pero no los jugadores de otros equipos; además, puede ver los detalles de cada jugador (CU-15) y editar en cualquier momento la ficha de este (CU-14). Cuando un jugador abandona el equipo, este tiene que ser dado de baja por el entrenador (CU-13). El jugador puede visualizar sus datos personales (CU-15) y editarlos (CU-14).

Gestión de ejercicios

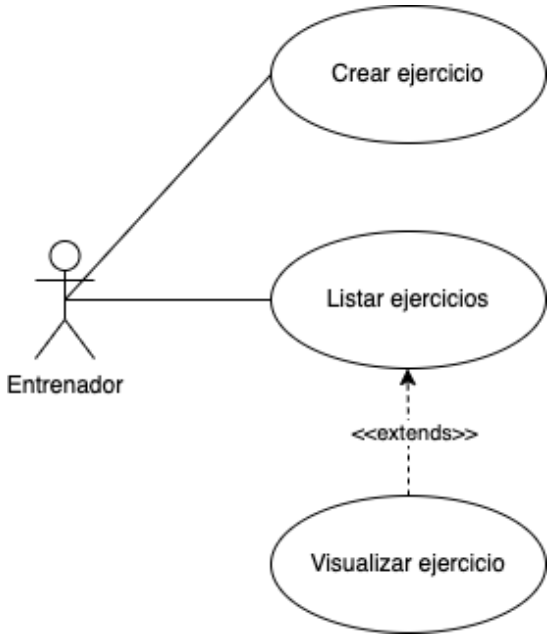

Figura 11 Diagrama casos de uso (gestión de ejercicios).

Un entrenamiento contiene ejercicios, los entrenadores pueden ir añadiendo nuevos, como se muestra en la ilustración 11 en el CU-17 (Anexo 5). Al crear el entrenamiento, los entrenadores visualizan los ejercicios pertenecientes a su club en formato lista (CU-18). Un entrenador puede utilizar un ejercicio creado por otro, además de visualizarlos en detalle (CU-19).

Gestión de videos

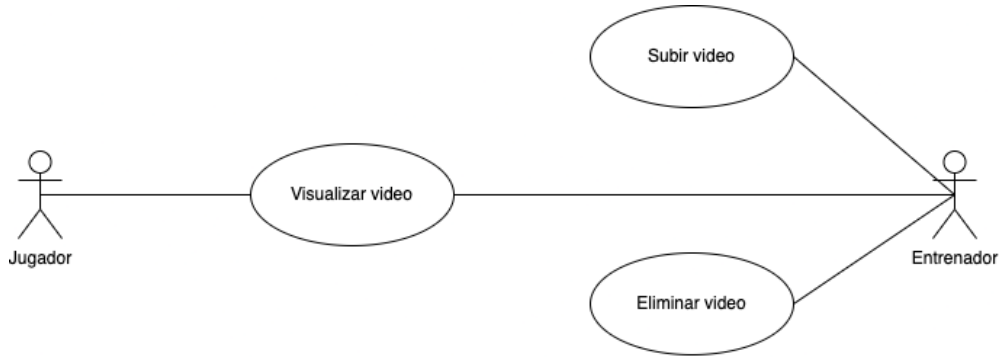

Figura 12 Diagrama casos de uso (gestión de videos).

Los entrenadores pueden subir los entrenamientos y ejercicios del equipo en formato video CU-20 (Anexo 6). Tanto los jugadores como entrenadores de un mismo equipo pueden visualizar el video (CU-21). El entrenador es la única persona que tiene la posibilidad de eliminar el video (CU-22).

Gestión de entrenamiento

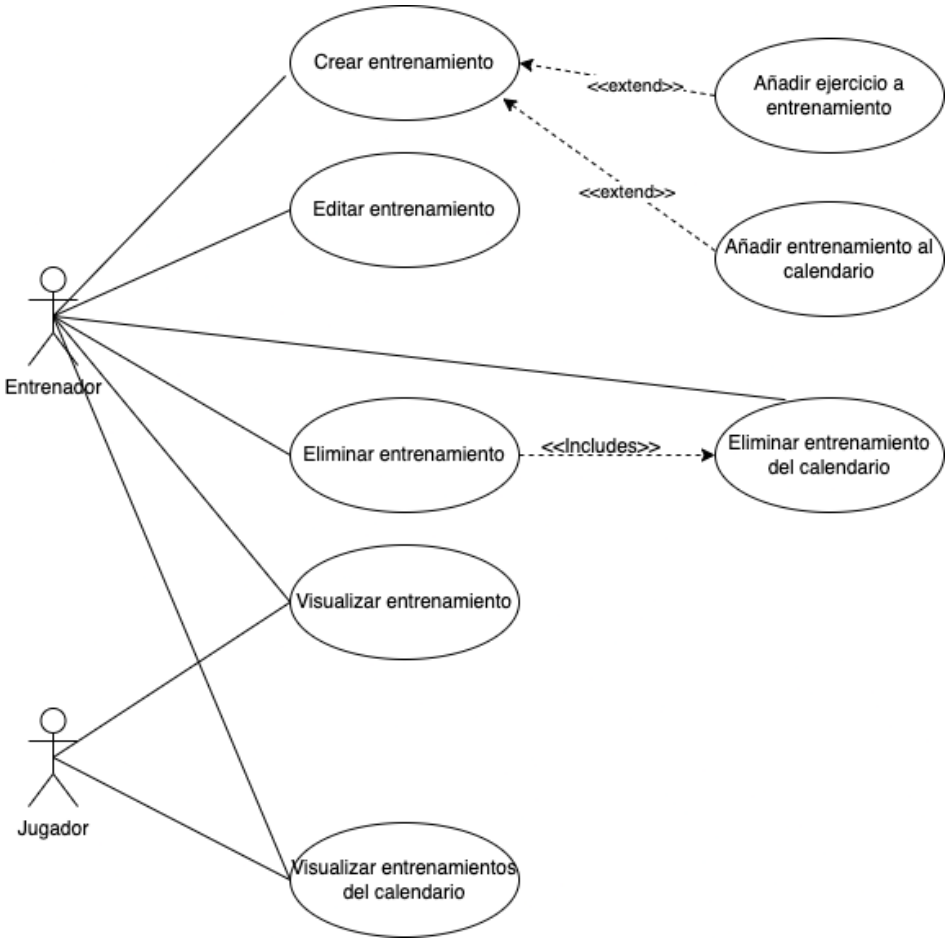

Figura 13 Diagrama casos de uso (gestión de entrenamiento).

Los entrenadores crean los distintos entrenamientos CU-23 (Anexo 7) añadiéndolos al calendario (CU-28). Los entrenadores y jugadores visualizan este calendario con los entrenamientos (CU-27). El entrenador también tiene la posibilidad de quitar los entrenamientos del calendario (CU-29) o directamente eliminando el entrenamiento (CU-25). Tanto el entrenador como el jugador pueden visualizar los detalles del entrenamiento (CU-26).

3.3 Prototipos

Con el fin de validar los requisitos junto con directivas y entrenadores de distintos clubes amateurs, se han desarrollado una serie de prototipos de interfaz de usuario, enlazados a los casos de uso que se muestran en el Apartado 3.2 del documento. Algunos bocetos se han omitido debido al parecido de otros.

3.3.1 Gestión de equipos

Listado de equipos

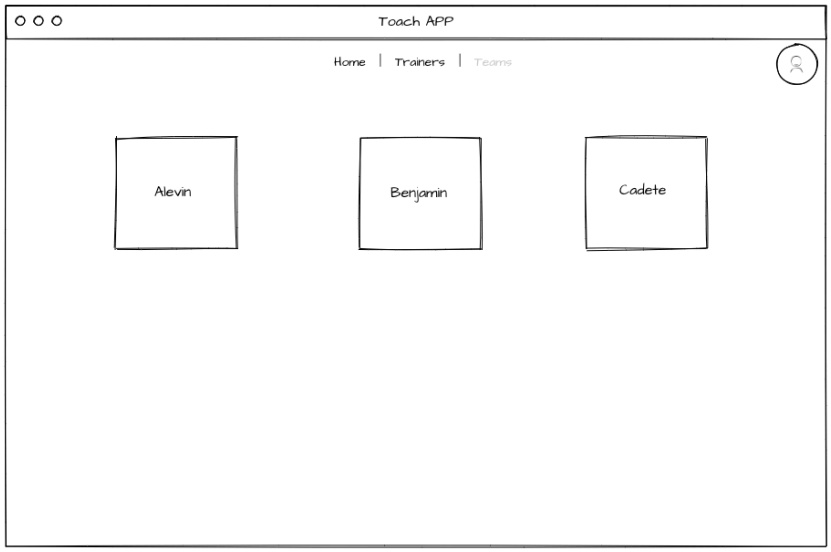

Figura 14 Boceto listado de equipos.

La ilustración 14 muestra el diseño de la interfaz para el caso de uso número 6 (CU-6) del anexo 2. La directiva visualiza el listado de los equipos que se encuentran en el club, puede seleccionar cualquiera de estos para ver los detalles del equipo.

3.3.2 Gestión de jugadores

Listado de jugadores

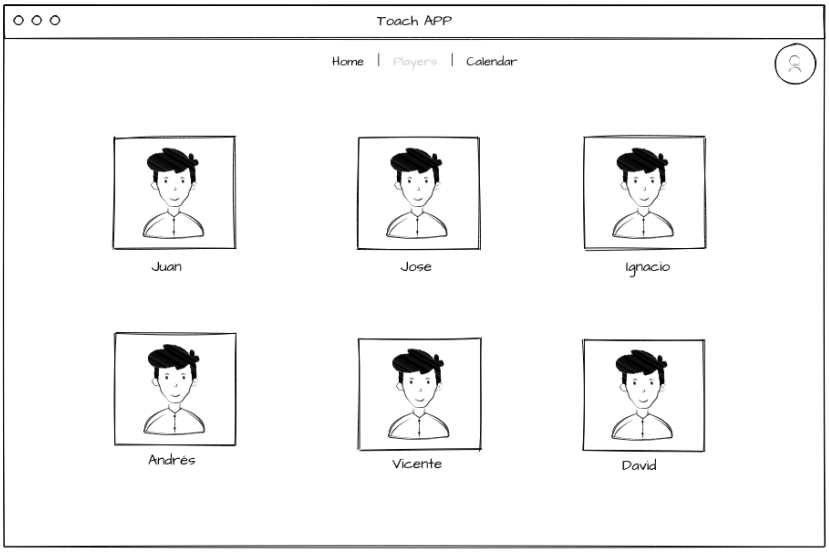

Figura 15 Boceto listado de jugadores.

La ilustración 15 muestra el diseño de la interfaz para el caso de uso número 16 (CU-16) del anexo 4, en el que el entrenador visualiza el listado de los jugadores que forman el equipo, puede seleccionar cualquiera de estos para ver los detalles del jugador.

Visualizar datos del jugador

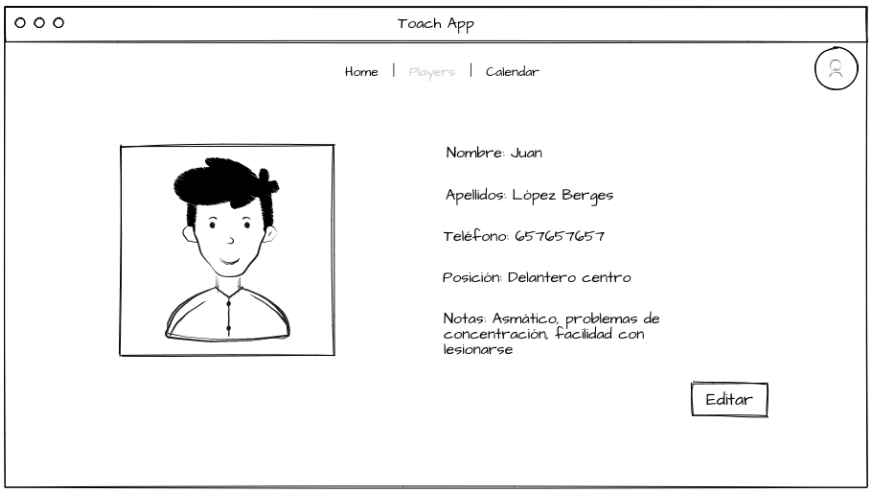

Figura 16 Boceto visualizar datos de jugador.

La ilustración 16 hace referencia al caso de uso número 15 (CU-15) del anexo 4, donde el entrenador muestra los detalles del jugador. Presionando el botón “editar” accede a modificar los datos.

Editar / eliminar jugador

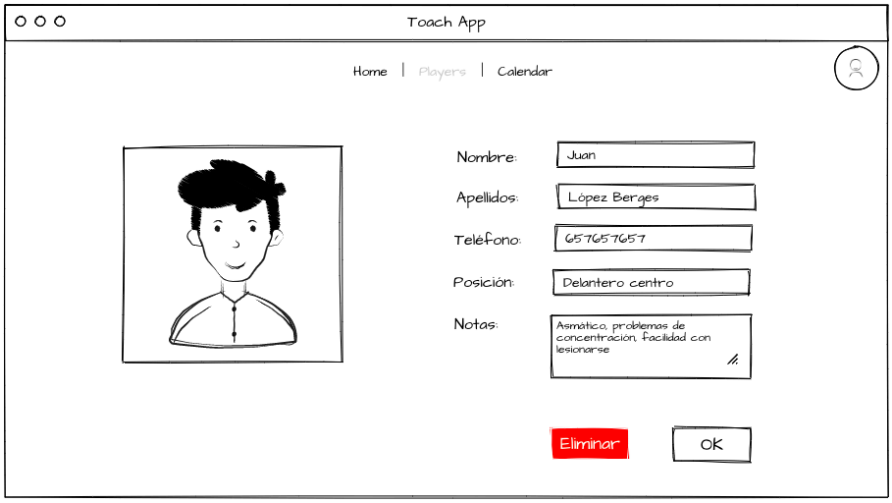

Figura 17 Boceto editar / eliminar jugador.

La ilustración 17 hace referencia a los casos de uso 13 y 14 (CU-13) y (CU-14) del anexo 4. Después de haber presionado el botón “editar” de la ilustración 16 se muestra el boceto de la ilustración 17, el cual editamos los datos del jugador o lo eliminamos presionando el botón “eliminar”.

Dar alta jugador

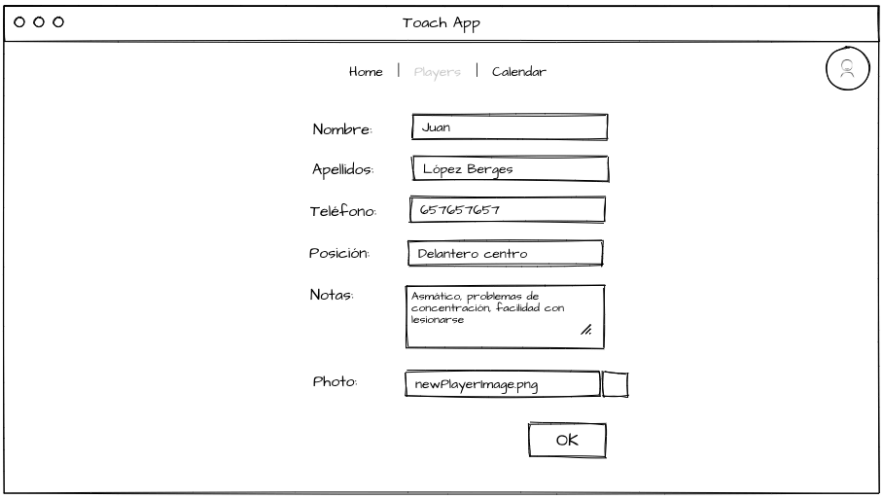

Figura 18 Boceto dar alta jugador.

La ilustración 18 hace referencia al caso de uso 12 (CU-12) del anexo 4. El entrenador añade a un nuevo jugador rellenando los datos correspondientes.

3.3.3 Gestión de ejercicios

Nuevo ejercicio

Figura 19 Boceto creación de ejercicio.

La ilustración 19 hace referencia al caso de uso 17 (CU-17) del anexo 5. El entrenador añade nuevos ejercicios para incorporarlos a los entrenamientos. Además, este ejercicio puede ser usado por todos los entrenadores que pertenecen al mismo club.

3.3.4 Gestión de entrenamientos

Vista del calendario

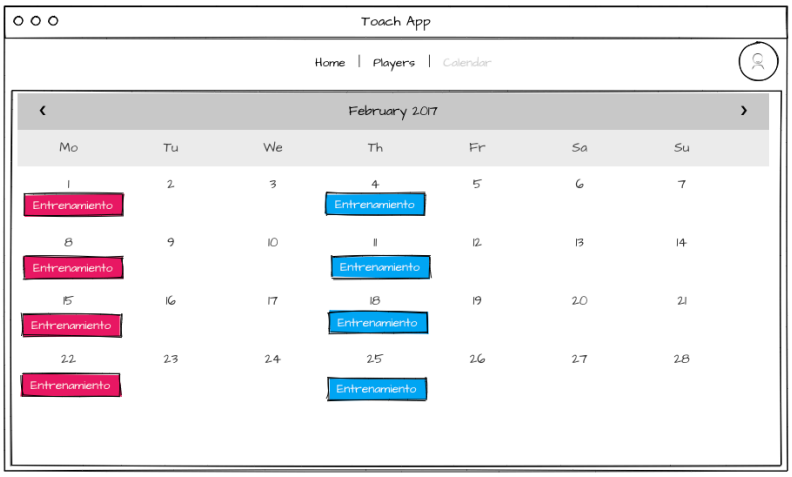

Figura 20 Boceto calendario de entrenamientos.

La ilustración 20 hace referencia al caso de uso 27 (CU-27) del anexo 7. El entrenador visualiza el calendario con los entrenamientos creados, pinchando en ellos observa los detalles de estos.

Visualizar entrenamiento

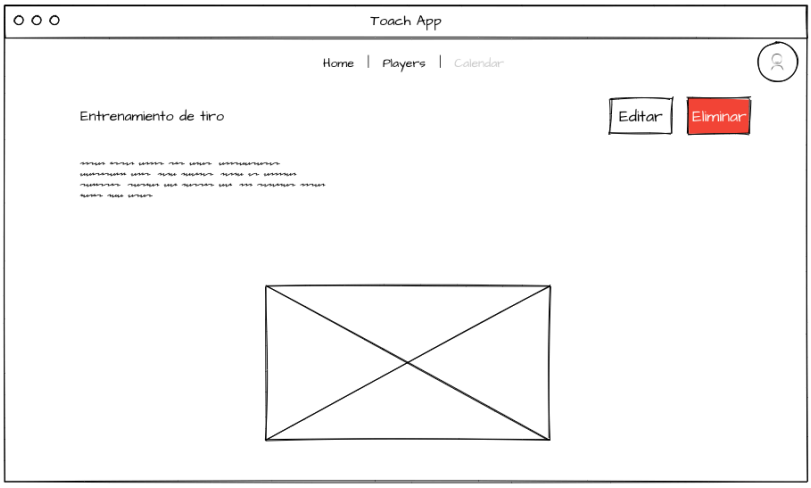

Figura 21 Boceto detalles de entrenamiento.

La ilustración 21 hace referencia al caso de uso 26 (CU-26) del anexo 7. El entrenador del equipo visualiza los detalles del entrenamiento que ha seleccionado en el calendario, pudiendo editarlo o eliminarlo. En esta vista se muestra la descripción del entrenamiento, los ejercicios que lo componen y el video adjuntado a este.

3.4 Identificación y análisis de posibles soluciones

En esta sección se pretende describir las posibles soluciones a la hora de crear la aplicación, comentando el abanico de posibilidades que existen para elegir la metodología de desarrollo de software más adecuada a nuestro proyecto y sobre el tipo de desarrollo a seguir.

3.4.1 Metodologías de desarrollo de software

RUP

RUP (Rational Unified Process) es una metodología tradicional de desarrollo de software basada en la generación de documentación exhaustiva y seguimiento estricto del plan de proyecto. Es útil cuando el proyecto no es cambiante y donde los requisitos, coste y tiempo son conocidos de una manera precisa. RUP hace uso de diagramas UML, dirigido principalmente por casos de uso. Además, RUP es iterativo e incremental, con incrementos de larga duración.

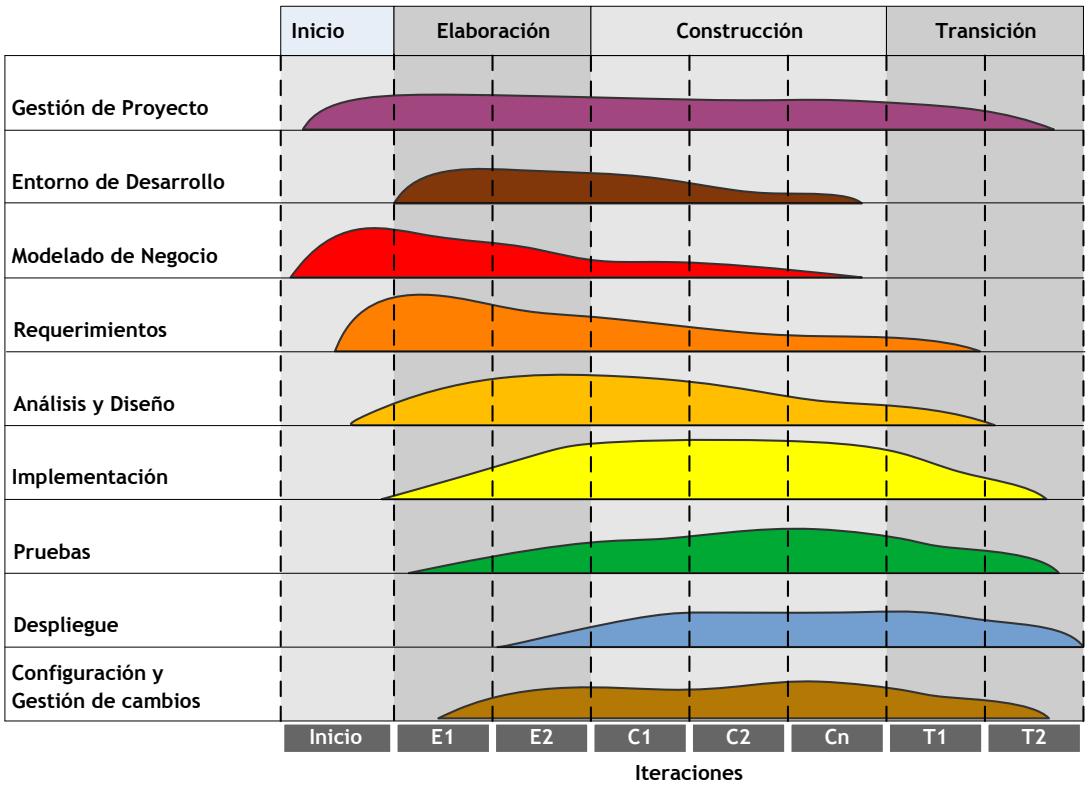

Figura 22 Diagrama de trabajo RUP [14].

Las fases del proceso son: [15, 16]

Inicio: Da comienzo el proyecto y se acuerda el alcance de este con los clientes, se identifican los posibles riesgos que pueden aparecer y se elabora un plan de negocio para determinar los recursos.

Elaboración: Se construyen las bases para la construcción de la arquitectura del sistema, mediante diagramas de casos de uso.

Construcción: Esta fase trata de implementar las distintas funcionalidades del proyecto, realizar las pruebas y observar las evaluaciones de los clientes.

Transición: El objetivo de esta fase es garantizar que los requisitos definidos anteriormente han sido cumplidos. Cuando los *stakeholders* están satisfechos debido a que se han alcanzado los objetivos definidos en la primera fase. Finaliza la fase de transición con el lanzamiento de la aplicación [17].

XP

La metodología XP (eXtreme Programming), es una metodología ágil enfocada en equipos de trabajo pequeños, con requerimientos imprecisos y cambiantes [18]. XP se puede separar en distintas fases [19]:

Exploración: El cliente expone lo que necesita mediante “historias de usuario”, requisitos escritos en una o dos frases haciendo uso de un lenguaje común por el cliente. Los desarrolladores hacen una primera estimación de tiempo de cada “historia de usuario”.

Planificación: El cliente y los desarrolladores agrupan las historias de usuario por entregas, después el cliente las ordenará según sus prioridades.

Desarrollo: Las historias de usuario son llevadas a cabo por los desarrolladores, generando un entregable funcional de cada una. El cliente participa activamente durante esta fase, aportando a los desarrolladores los datos necesarios sobre las historias.

Puesta en producción: En esta fase el cliente decide si la aplicación se pone en producción o faltan entregables.

XP sigue un conjunto de reglas y prácticas, a continuación, se mostrarán las más destacadas:

Programación en pares: XP propone que se programe en pares, dos personas trabajando en el mismo ordenador. Esto minimiza los errores, logrando mejores diseños y obteniendo un código de mejor calidad que cuando se desarrolla individualmente.

Programación dirigida por pruebas (*Test Driven Development)*: XP propone un modelo en el que primero se escriben las pruebas y después el desarrollo. Se profundizará sobre este tema en el siguiente punto del documento (3.4.2).

Pruebas de aceptación: El cliente debe especificar distintos escenarios para comprobar que una historia ha sido desarrollada correctamente. Una historia no ha finalizado hasta que pase correctamente todas las pruebas de aceptación.

Scrum

Scrum es una metodología ágil donde se fomenta el trabajo en equipo y entregas continuas con el objetivo de ser altamente productivos. Esta metodología incluye distintos roles:

Product Owner: Representa la voz del cliente, deben de entender las necesidades de estos y encargarse de comunicar el objetivo del proyecto, negociar con el equipo de desarrollo las tareas que realizarán y además dar una opinión acerca de los entregables en cada iteración.

Scrum Master: Es el encargado del cumplimiento de las reglas de la metodología Scrum, ayudando y fomentando el trabajo en equipo. El *Scrum Master* programa y dirige las reuniones del equipo, asegurando que se cumplen las reglas de estas. Por último, elimina cualquier bloqueo o impedimento que pueda tener el equipo.

Equipo de desarrollo: Es el encargado de desarrollar las tareas que se encuentran programadas en el sprint, realizar las estimaciones de tiempo de estas, negociar con el *product owner*, y completar las entregas del producto [20].

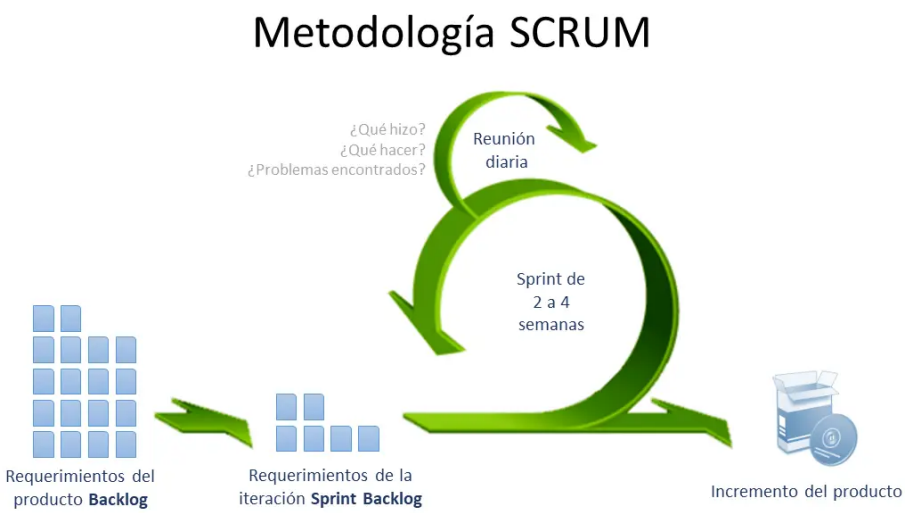

Figura 23 Metodología Scrum [21].

La Ilustración 23 muestra los fundamentos de Scrum. Este tipo de metodología trata de realizar entregas del producto final, fijadas en un acuerdo entre el equipo de desarrollo y el *product owner*, para conseguir un resultado completo con cada iteración. Normalmente las iteraciones son de 2 semanas, aunque puede haber equipos que prefieren 3 o 4 semanas.

El backlog es un almacén en el cual se encuentran las tareas a realizar, ordenadas por prioridad. El equipo de desarrollo es el encargado de añadir y realizar las estimaciones de las tareas, siendo el *product owner* el responsable de priorizar estas. Una vez las tareas se encuentran priorizadas de mayor a menor importancia, según la capacidad del equipo de desarrollo, estos proceden a crear el *Sprint*. Es muy importante tener bien definida la tarea, junto con las pruebas de aceptación y una estimación en horas o puntos.

Antes de comenzar un *sprint*, se realiza la reunión llamada *sprint planning*, para seleccionar y comunicar qué tareas es capaz de realizar el equipo, las cuales son seleccionadas del inicio del *backlog*. Una vez finalizado el *sprint* el *product owner* realiza una revisión del trabajo entregado por el equipo de desarrollo y una reunión de retrospectiva en la cual todos los integrantes dejan sus impresiones con el objetivo de mejorar como equipo.

3.4.2 Proceso de desarrollo de software

TLD

El proceso de desarrollo TLD (*Test Last Development)* es el más usado en proyectos de desarrollo tradicionales. En TLD se escribe primero el código y después se crean las pruebas. En caso de que estas fallen, se corrige el código hasta que pasen correctamente [22].

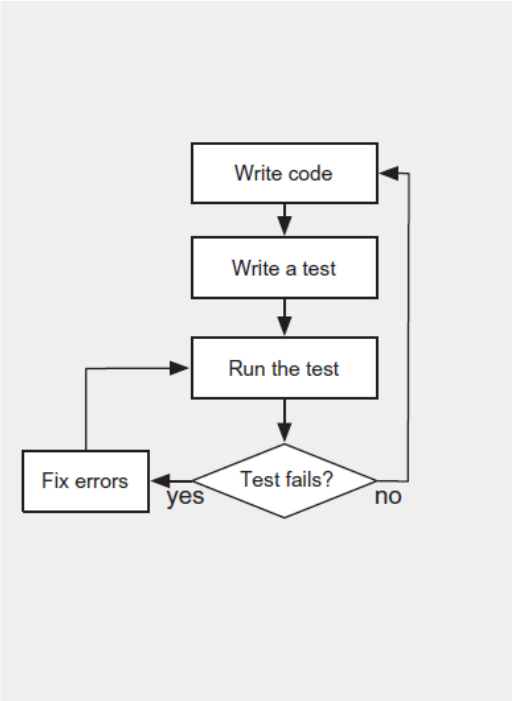

Figura 24 Proceso TLD [22].

TDD

El proceso TDD (*Test Driven Development*) es una técnica de ingeniería de software en el que las pruebas se crean antes del código [22]. Esta técnica mejora la calidad del código y de las pruebas, además de mejoras en cuanto a productividad debido al rápido entendimiento de los requisitos por parte de los desarrolladores. TDD se emplea en diversas metodologías ágiles, como XP.

Para aplicar TDD se debe seguir un ciclo conocido como *Red-Green-Refactor* [23] que trata de seleccionar una característica a implementar y escribir una prueba de acuerdo con esta. En un primer momento, la prueba debe de fallar (*Red*) debido a que el código todavía no se ha escrito. Después de crear la prueba se comienza a escribir el código para hacer que pase correctamente (*Green*). Una vez pasada la prueba, se refactoriza el código (*Refactor*), para crear un código más limpio y de mayor calidad.

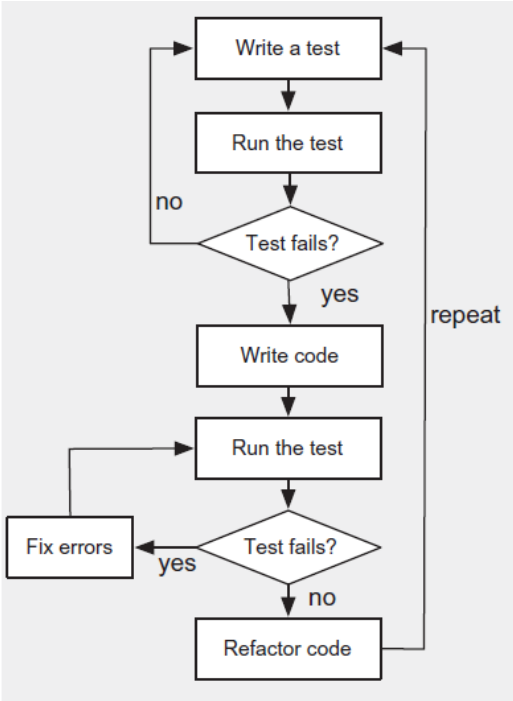

Figura 25 Proceso TDD [22].

3.5 Solución propuesta

Tras revisar el abanico de posibilidades metodológicas, y dadas las características del proyecto y del equipo de desarrollo, se ha decidido adaptar la metodología SCRUM a las necesidades del proyecto. Siendo el equipo solamente una persona (el autor de este trabajo), no existe ningún rol, pero se mantiene la creación y el mantenimiento del backlog, siguiendo una organización de *sprints*, permitiendo generar gráficas y ver el estado de trabajo. Si el equipo aumentara en un futuro, se comenzaría a hacer uso de los roles definidos en la metodología.

Se ha elegido el proceso de desarrollo TDD para poder conseguir una mejora como desarrollador en cuanto a calidad externa e interna y productividad. De acuerdo con K. Beck, TDD ayuda a la comprensión del código, la eficiencia a encontrar el problema cuando se introducen nuevos fallos, y la reducción de defectos introducidos en el código durante el mantenimiento [24].

4. Diseño de la solución

En esta sección se explica el diseño del sistema, comenzando por su arquitectura y después entrando en detalle sobre el diseño.

4.1 Arquitectura del sistema

Para la realización de este proyecto se apuesta por la división de 3 niveles: presentación, lógica de negocio y persistencia (ver ilustración 26).

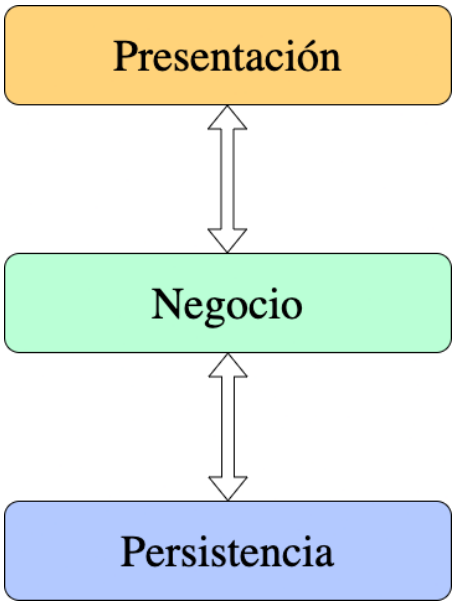

Figura 26 Arquitectura de la aplicación.

La arquitectura de tres niveles organiza la aplicación en distintos niveles lógicos y físicos. El nivel de presentación, donde se encuentra la interfaz del usuario; el nivel de negocio, donde se encuentra toda la lógica para procesar los datos y, por último, el nivel de persistencia, donde se almacenan los datos [25]. Cada nivel puede encontrarse en una infraestructura distinta y desacoplado de los demás, por lo que estos niveles pueden escalar y actualizarse según sea necesario.

Suele haber una confusión entre nivel y capa. La división en capas es simplemente la división funcional, sin embargo, la división en niveles es una división funcional y de infraestructura. Esto quiere decir que el software se ejecuta en distintas máquinas y no en la misma.

El nivel de presentación es la encargada de mostrar al usuario la información, interactuar con él y ajustar estos datos a las especificaciones que son demandadas por la capa de negocio. En el caso de nuestra aplicación este nivel se ejecuta en el

navegador web. El nivel de negocio es el encargado de facilitar al nivel de presentación los datos necesarios, a la vez que recolecta los datos aportados por este para procesarlos. Este nivel está también conectado con el nivel de persistencia, enviándole peticiones para almacenar o recuperar datos de este [25].

El nivel de persistencia es donde residen los datos y generalmente está formada por un gestor encargado de realizar el almacenamiento de datos. Este nivel recibe las peticiones de negocio, quien le dice qué datos necesita o qué datos tiene que almacenar. El nivel de persistencia no se puede comunicar directamente con el nivel de presentación, siempre debe de comunicarse con negocio previamente [25].

4.2 Diseño detallado

4.2.1 Nivel de persistencia

Toach App utiliza mongoDB, un sistema de base de datos no relacional o Not Only SQL (NoSQL) de código abierto orientada a documentos. Es decir, almacenan documentos no estructurados, como texto o semiestructurados, como XML o JSON, almacenando conjuntos de clave valor [26].

Las bases de datos relacionales solamente pueden escalar verticalmente, mientras que las bases de datos NoSQL consiguen escalar tanto horizontalmente como verticalmente [27]. El escalamiento vertical consiste en agregar recursos a un nodo, como por ejemplo aumentando su capacidad o su potencia, mientras que el escalamiento horizontal consiste en el aumento de la cantidad de nodos para distribuir la carga de trabajo [28].

Para la creación de datos, es necesario realizar el esquema que define la estructura y el tipo de contenido de los datos a almacenar, formado por modelos. La Ilustración 27 muestra los modelos que tiene el esquema de nuestra aplicación, basados en el diagrama de clases de la ilustración 5.

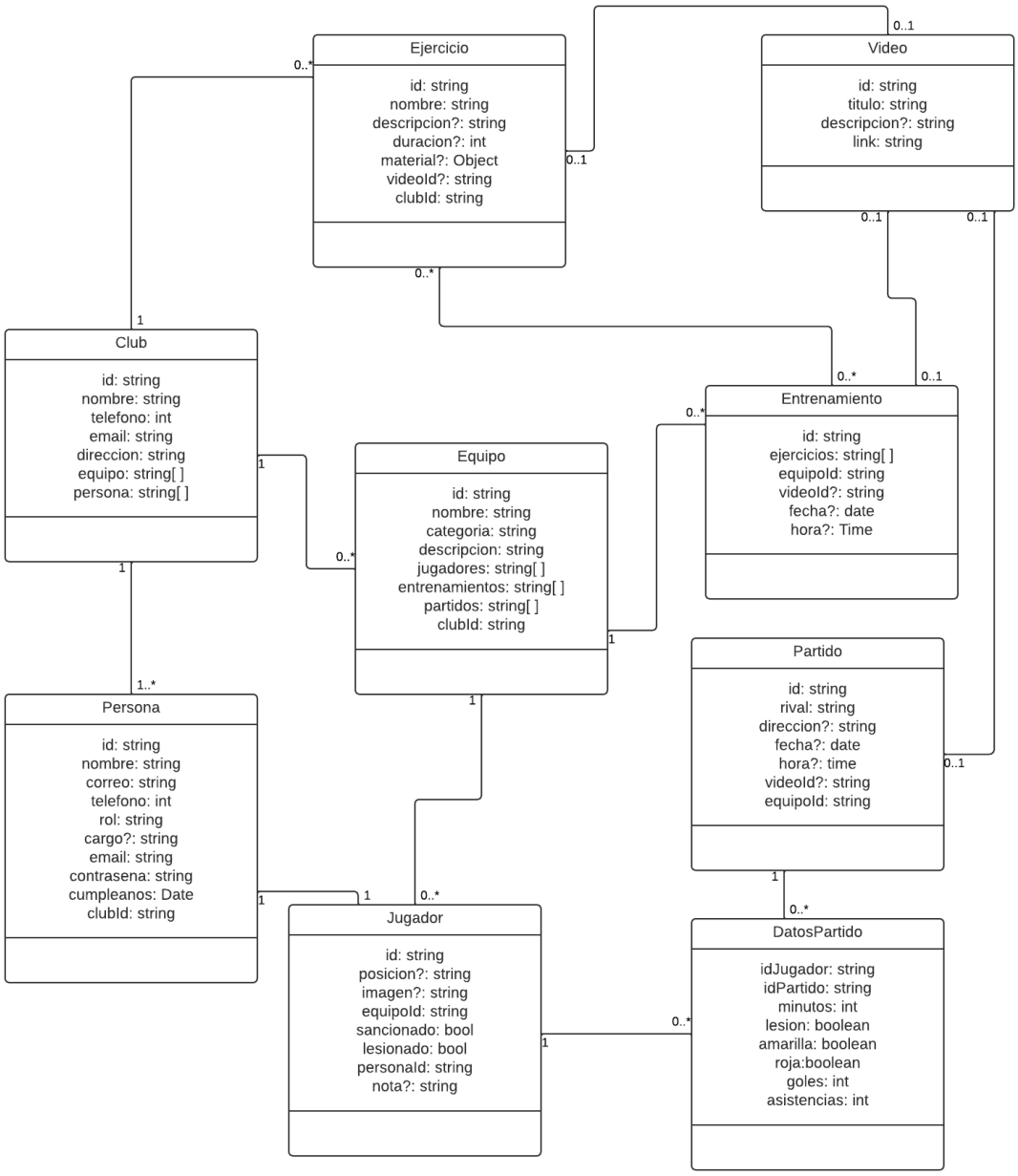

Figura 27 Diagrama de base de datos.

En el estudio realizado por Robert T. Mason muestra un ejemplo del proceso de modelar un sistema NoSQL orientado a documentos [27]. En la ilustración 27 se encuentra el diagrama de base de datos.

Se han realizado algunas modificaciones en los modelos. Los modelos con una relación de “muchos”, se crea un parámetro de tipo *Array* en el que se encuentran los identificadores de los documentos relacionados, por ejemplo, en el caso de Entrenamiento puede contener o no ejercicios. Cuando la relación es de “uno” el identificador del documento se añade en formato *String* al documento que desea relacionar.

4.2.2 Transporte de los datos entre presentación y negocio

La siguiente ilustración muestra el flujo de datos entre el nivel de presentación y el nivel de negocio.

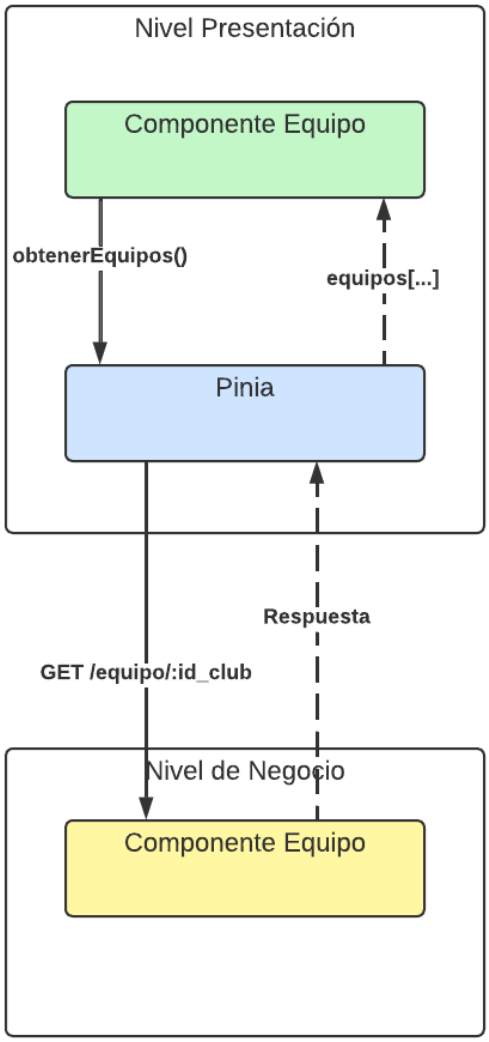

Figura 28 Comunicación entre presentación y negocio.

La ilustración 28 representa la petición realizada por la directiva para la obtención de los equipos de un club. Cuando el usuario accede en el navegador a la ventana de equipos, internamente el componente equipo del nivel de presentación realiza una llamada a la función de Pinia, tecnología que se explica en el apartado 4.3. Esta lanza la petición a la API, la cual escucha y responde a las peticiones de los clientes [29], que se encuentra en el nivel de negocio, encargada de realizar las acciones necesarias para recopilar los datos y enviar la respuesta de vuelta a Pinia. Cuando ya ha obtenido la respuesta, este envía los datos al componente equipo, quien muestra el listado de equipos por el navegador.

4.2.3 Transporte de los datos entre Negocio y Persistencia

La siguiente ilustración muestra el flujo de datos entre el nivel de negocio y el nivel de persistencia.

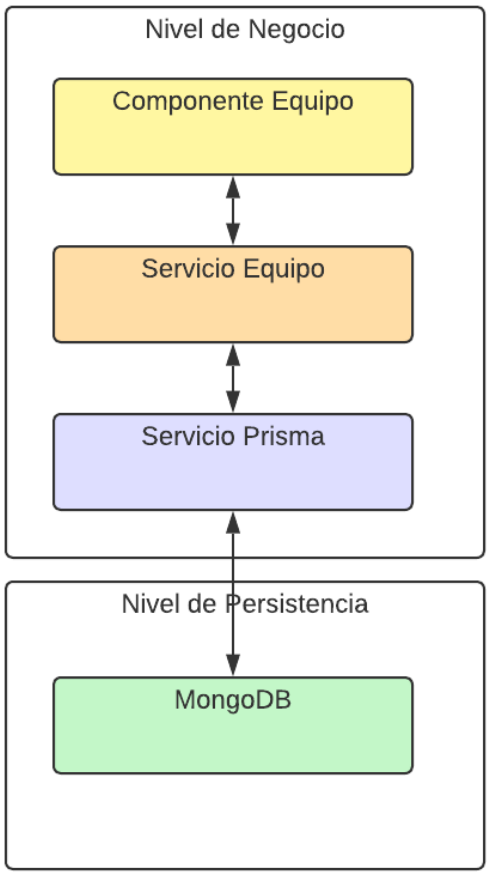

Figura 29 Comunicación entre negocio y persistencia.

Se ha explicado anteriormente cómo los datos son pedidos por el nivel de presentación y devueltos por el nivel de negocio. En la ilustración 29 se muestra cómo el nivel de negocio realiza las acciones necesarias para obtenerlos. Una vez la petición ha llegado al componente equipo, este llama a su servicio, encargado de comunicar la acción a realizar a prisma, tecnología que se explica en el apartado 4.3.1. Prisma se comunica con mongoDB y realiza la consulta; una vez obtenido los datos, se los devuelve al servicio equipo para retornarlos al componente. El componente es el encargado de generar la respuesta para el nivel de presentación.

4.3 Tecnología utilizada

4.3.1 Nivel de negocio

Como tecnología a usar para el desarrollo del nivel de negocio se ha elegido NestJS, framework de NodeJS desarrollado en el lenguaje de programación Typescript, que amplía JavaScript añadiendo tipos estáticos y objetos basados en clases.

NestJS utiliza como base el famoso framework de NodeJS express, proporcionando toda su potencia y facilidad a la hora de crear APIs. La principal característica del framework es facilitar una arquitectura muy parecida a la que podemos observar en Angular, framework de JavaScript para la creación de aplicaciones web, aumentando la escalabilidad, testeabilidad y productividad de la lógica de negocio [30]. NestJS hace uso del patrón inyección de dependencias; este patrón puede crear y proporcionar instancias de las clases que se necesitan en la aplicación con el objetivo de lograr un bajo acoplamiento entre los objetos [31].

Se construye la aplicación módulo a módulo, pudiendo ser estos controladores o proveedores:

- Controladores: Son la pieza fundamental de Nest, encargados de implementar las APIs donde reciben peticiones y devuelven las respuestas al cliente [32, 33].

- Proveedores: Conjunto diverso de clases que pueden ser inyectados dentro de otras clases, esto significa que los objetos pueden crear diversas relaciones entre sí [33].

Para la comunicación del nivel de negocio con el nivel de persistencia se ha apostado por el kit de herramientas de base de datos de código abierto Prisma. Nos facilita el acceso a la base de datos un generador automático de consultas adaptado a nuestros esquemas [34].

Prisma está dividido en 3 partes:

- Prisma Client: Generador de consultas.

- Prisma migrate: Sistema de migración y modelado de los esquemas de datos.

- Prisma Studio: Proporciona una interfaz gráfica en la que se pueden visualizar los datos de nuestra base de datos.

4.3.2 Nivel de presentación

En cuanto al nivel de presentación se ha elegido el lenguaje Typescript junto con la versión 3 del framework VueJS.

VueJS es un framework de javascript que nos permite crear interfaces de manera rápida, escalable y sencilla. Permite desarrollar software de distintas formas, una de ellas es la reactiva, renderizando vistas cuando ocurre un evento, como su rival React. El núcleo de VueJS es muy ligero, convirtiéndolo en un framework muy versátil [35].

En este proyecto se ha trabajado mediante API de composición, lanzada en la versión 3 de Vue, que se centra en funciones de composición, las cuales favorecen la reusabilidad de la lógica entre componentes. Una pieza muy importante es el uso de *setup*, donde se escribe el código de inicialización del componente a modo de constructor y a su vez se declara el estado reactivo exponiéndolo a la plantilla [36, 14].

Hoy en día es necesario hacer uso de alguna librería que nos permita gestionar el estado, almacén de datos compartido por los componentes de la aplicación. La versión 3 de VueJS nos facilita Pinia, librería que gestiona el estado haciendo uso de un *store* [37]. Un *store* es un almacén centralizado para mantener los datos que están disponibles, se ha generado uno por cada módulo favoreciendo a la división del código.

Para crear unas interfaces más profesionales de forma ágil, se ha utilizado Tailwind, framework basado en clases que se pueden aplicar con facilidad en el código HTML, optimizando el peso del código CSS. Tailwind es una herramienta muy potente a la hora de crear interfaces, esta se apoya en PostCSS para alcanzar un flujo de desarrollo avanzado y optimizado. Gracias a PostCSS se limpian todas las clases que no se están haciendo uso, manteniendo solamente las que se ha utilizado en el proyecto [38].

4.3.3 Nivel de persistencia

El sistema de base de datos elegido es MongoDB, donde se ha desplegado la base de datos en Mongo Atlas, servicio que nos permite configurar y utilizar nuestra base de datos en un servidor proporcionado por Amazon Web Service (AWS). Este servidor ofrecido por mongo es totalmente gratuito debido a que es compartido con otras bases de datos. MongoDB Atlas nos permite crear colecciones sobre nuestra base de datos, observar métricas del servidor en tiempo real y el manejo de este mediante consola [39].

4.3.4 Administración del proceso de desarrollo

Para la administración de tareas, se ha utilizado la herramienta Jira, creada por la empresa Atlassian, que permite realizar una gestión ágil del proyecto. Jira crea

automáticamente el *backlog*, que es la lista de trabajo donde se encuentran las tareas encontrándose arriba las más importantes por el propietario del producto [40].

Dentro de Jira, se puede crear distintos *sprints*, explicado anteriormente en el punto 3.4.1, indicando las fechas y las tareas que existen. Además, permite visualizar los informes del sprint, siendo muy útiles para el equipo de desarrollo y sobre todo para el Scrum máster.

La plataforma no solo permite la creación y el seguimiento de los sprints, también es posible realizar un seguimiento de los bugs registrados.

4.3.5 Creación de modelos

La herramienta elegida para la creación de modelos es la aplicación de código abierto Diagrams.net, que permite crear todo tipo de diagramas de forma muy intuitiva y fácil de usar.

4.3.6 Entorno de trabajo

El entorno de trabajo está formado por varias herramientas. Se ha elegido el famoso editor creado por Microsoft, Visual Studio Code, como editor de texto. Este nos ofrece una gran cantidad de *plugins* y una personalización en la que permite al desarrollador crear su entorno de trabajo favorito.

4.3.7 Gestión de versiones

El sistema que realiza un control de versiones sobre el presente proyecto es Git, creado por el gran conocido Ingeniero de Software Linus Torvalds. Esta herramienta se encuentra integrada en el entorno de trabajo Visual Studio Code y permite realizar imágenes del código elegido en un momento dado. Gracias a las ramas, Git permite un desarrollo simultaneo por varios desarrolladores, trabajando cada desarrollador en una rama distinta [41].

Para alojar el proyecto haciendo uso de Git, se utiliza la plataforma GitHub, cuyo actual propietario es Microsoft. GitHub proporciona una interfaz al usuario para gestionar los repositorios del usuario. Además, los repositorios son por defecto de acceso público permitiendo la clonación por cualquier usuario. Mediante una cuenta de pago es posible crear repositorios privados [42].

4.3.8 Testing

Nestjs nos proporciona una integración total de Jest, framework de testing del lenguaje JavaScript creado y mantenido por Facebook [43]. Según la encuesta anual realizada a los desarrolladores *State of JavaScript 2021* [44], Jest es el la herramienta de *testing* más usada.

Para completar con el kit de herramientas de *testing,* se hace uso de Supertest, módulo de NodeJs que nos permite probar servicios HTTP.

5. Desarrollo de la solución propuesta

En este punto, habiendo presentado anteriormente el diseño de la aplicación, se va a realizar una breve explicación del desarrollo de los distintos niveles que componen esta. En cada nivel se va a destacar lo más importante de cada uno, evitando extendernos en detalles técnicos.

5.1 Nivel de negocio

La siguiente ilustración muestra el conjunto de APIs que compone el nivel de negocio. La tecnología que ofrece la herramienta Swagger nos permite visualizar y probar estas APIs desde el navegador. Además, aporta la ventaja de documentar y especificar las entradas de una operación definida en cada API [45].

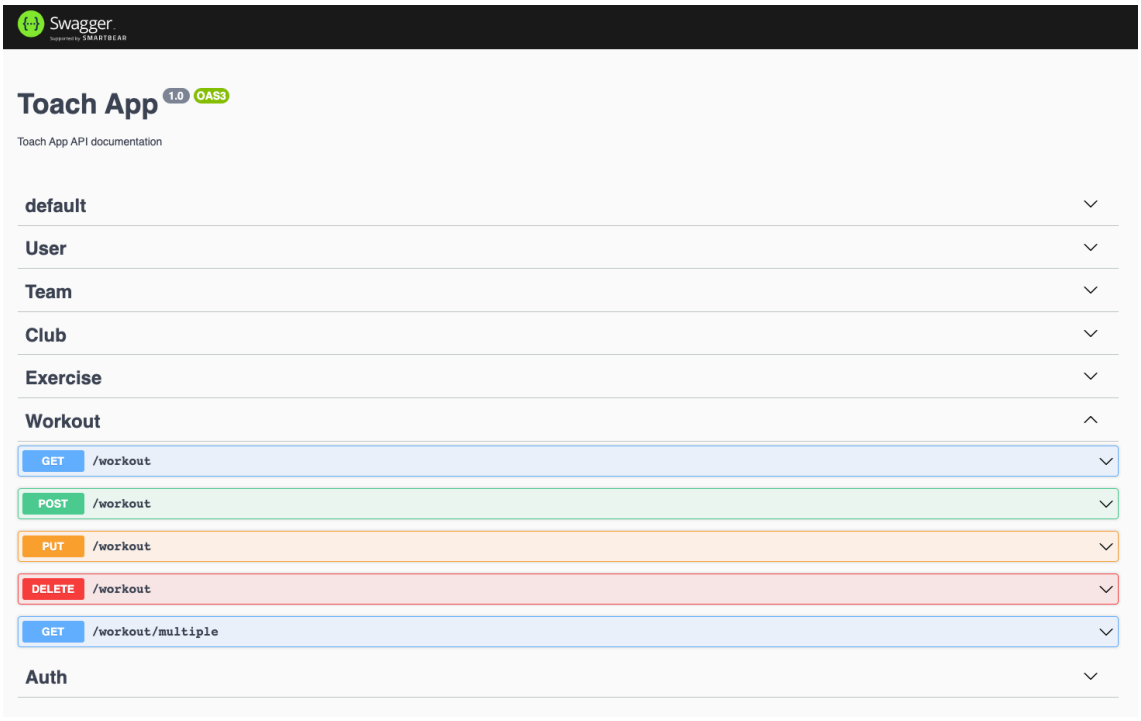

Figura 30 Conjunto de APIs.

En la ilustración 30 se muestra el conjunto de APIs que está formado el nivel de negocio. A modo de ejemplo, se muestran los métodos para hacer uso de la API

workout. Estos son métodos de petición HTTP (Protocolo de Transferencia de Hipertexto en español) que indican la acción a realizar:

GET /workout: Obtener todos los entrenamientos de un equipo.

POST /workout: Crear un nuevo entrenamiento.

PUT /workout: Actualizar un entrenamiento.

DELETE /workout: Eliminar un entrenamiento.

GET /workout/multiple: Obtener los entrenamientos de los multiples equipos que puede pertenecer el entrenador.

Para finalizar con la explicación del desarrollo del nivel de negocio las ilustraciones 33 y 34 del anexo 8 muestran el código del controlador y del servicio respectivamente, ambos usados para la creación de la API.

5.2 Nivel de presentación

El desarrollo de las vistas del nivel de presentación está compuesto por componentes, piezas de código con un comportamiento específico que unidos crean aplicaciones web [46], si estos se abstraen correctamente pueden ser reutilizables, reduciendo el desarrollo repetitivo [47].

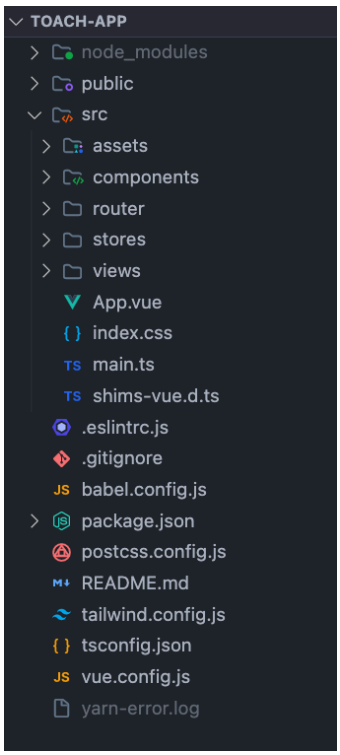

Figura 31 Estructura carpetas nivel de presentación.

En la ilustración 31 muestra la estructura por carpetas y archivos usados para el desarrollo del nivel de presentación. La carpeta “src” almacena todos los archivos con el código para la creación de la aplicación; el resto de las carpetas y ficheros son configuraciones del proyecto, muy importantes para el desarrollo de este, pero no se van a explicar. Dentro de esta encontramos las distintas carpetas y archivos:

Assets: Almacena las imágenes de la aplicación.

Components: Contiene los componentes explicados anteriormente.

Router: Contiene todas las rutas que muestran las vistas.

Stores: Contiene todos los *stores* creados mediante la librería Pinia.

Views: Contiene todas las vistas de la aplicación, como hemos mencionado anteriormente, están compuestas por componentes.

App.vue: Fichero global de la aplicación. En dicho fichero se incrustan todas las rutas para cargar las distintas vistas.

Index.css: Fichero encargado de cargar la librería Tailwind.

Main.ts: Fichero encargado de cargar la aplicación. En él se adjuntan las librerías principales, como Pinia y el fichero principal (App.vue).

Shims-vue.d.ts: Fichero generado automáticamente por Vue para el correcto funcionamiento de la librería.

5.3 Nivel de persistencia

Para finalizar con la explicación del desarrollo de los niveles de la aplicación, la siguiente ilustración muestra una instantánea del servidor donde se encuentra la base de datos.

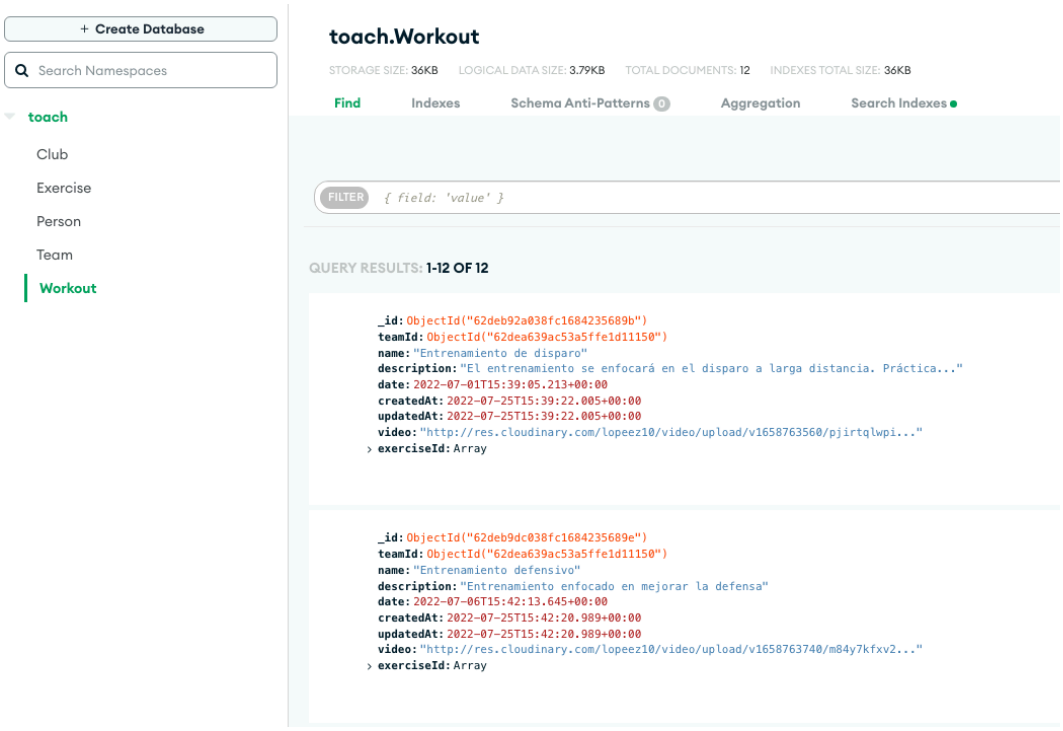

Figura 32 Base de datos en mongoDB Atlas.

La ilustración 32 está dividida en dos secciones, en el lado izquierdo de la imagen se encuentran las colecciones de la aplicación, creadas desde el fichero de la ilustración 35 del anexo 8. En el lado derecho se encuentran los documentos con los datos almacenados, estos documentos son definidos en el fichero de la ilustración 36, gracias a prisma. La ilustración 32 muestra los documentos del modelo *workout*.

Además del almacenamiento de datos, mongo ofrece una ventana muy interesante que nos indica cuando estamos realizando un esquema con antipatrones, un patrón que conduce a una mala solución [48]. En el caso de *workout* no se detecta ningún antipatrón.

6. Pruebas

A lo largo del proyecto se han realizado distintas pruebas, tanto de validación como de verificación.

Verificación: ¿Estoy construyendo el producto correctamente?

Validación: ¿Estoy construyendo el producto correcto?

Como forma de validación de requisitos, se utilizan los prototipos del Apartado 3.4 de este documento. Se realizan reuniones con distintos entrenadores en la que se muestran los bocetos realizados, estos validan los requisitos dando su opinión acerca de ellos. De esta manera se añadirán los cambios aportados por los *stakeholders* para que los desarrolladores programen un software validado por estos.

La verificación del software trata de asegurar que se implementa correctamente una función, de acuerdo con la especificación y satisfaciendo los requisitos funcionales y no funcionales, esta se puede realizar a través de pruebas [49]. Para la verificación de los componentes se han usado las pruebas unitarias. Estas consisten en probar una parte del código y verificar que funciona correctamente, la ilustración 35 muestra un ejemplo de prueba unitaria sobre el controlador y el servicio de autenticación o *auth*.

En cada tarea definida en un sprint se encuentran las pruebas de aceptación, definidas por el equipo de desarrollo. Estas pruebas tratan de verificar que se cumplen las expectativas del cliente [50]. Una vez finalizado el sprint, en la reunión *Sprint review,* el *Product Owner* valida que el producto realiza la funcionalidad esperada y cumple con las expectativas. A su vez, se verifica que la funcionalidad desarrollada funciona correctamente, sin fallos.

Figura 33 Tarea de Jira "Dar alta jugador".

En la ilustración 33 se muestra la tarea “dar alta jugador”, en ella podemos encontrar las pruebas de aceptación. La prueba de aceptación esta compuesta por una descripción de una posible situación, la condición de esta situación, los pasos para llevarla a cabo y por último los resultados esperados [51].

7. Conclusiones

Al comienzo del proyecto se planteaba la creación de una web app basada principalmente en la directiva, es decir, gestionando el club desde un rol más administrativo. Después de realizar un análisis sobre la competencia, la cantidad de aplicaciones enfocadas a la administración de un club era mayor y, además, estas aplicaciones estaban mejor desarrolladas, dando lugar a un aumento de dificultad a la hora de intentar hacernos un hueco en el mercado. Al contrario que las aplicaciones enfocadas a los entrenadores, donde hacerse un hueco en el mercado es más fácil gracias a que la mayoría de estas están obsoletas, como hemos observado en el análisis de la competencia. Por este motivo, se decidió comenzar por este tipo de aplicaciones, dejando para futuras versiones aumentar las características de la administración del club dentro de la web app.

Se ha desarrollado una aplicación web mediante un proceso de software, aplicando lo aprendido durante el Grado en Ingeniería Informática, haciendo un seguimiento mediante la metodología Scrum, la cual ha tenido que ser adaptada al equipo de desarrollo debido a que solamente estaba formado por una persona. A medida que avanzaba el proyecto, las estimaciones de tiempo de las tareas iban siendo cada vez mejores, debido al conocimiento sobre la capacidad de trabajo del equipo. También se ha actualizado continuamente el *backlog*, priorizando siempre las tareas negociadas con el tutor.

Una práctica mencionada pero no aprendida en la carrera ha sido *Test Driven Development* (TDD) la cual me ha permitido mejorar como desarrollador a la hora de pensar y crear pruebas. Esta práctica ha costado mucho esfuerzo ponerla en uso, debido a que siempre hemos creado las pruebas una vez desarrollado el código.

Una pequeña dificultad a la hora de crear el proyecto ha sido la capa de presentación, debido al uso del *framework* Vue 3, con el cual el equipo de desarrollo no había trabajado anteriormente. La curva de aprendizaje ha sido bastante más rápida en comparación a sus competidores, sin embargo, este *framework* utiliza un tipo de programación reactiva la cual al inicio es un poco compleja de comprender.

7.1 Relación del trabajo desarrollado con los estudios cursados

Durante la realización de este trabajo, se han aplicado los conocimientos aprendidos durante la carrera de Ingeniería Informática. La metodología empleada en este proyecto ha sido aprendida en las asignaturas de la rama de Ingeniería de Software, Proceso del Software (PSW) y Proyecto de Ingeniería de Software (PIN). Estas dos asignaturas han sido de gran ayuda a la hora de crear un proyecto desde cero, enseñándonos el ciclo de vida de un proyecto de emprendimiento.

En cuanto a la creación de modelos, se han aplicado los conocimientos aprendidos en la asignatura de Ingeniería de Software (ISW). Entrando en más detalle sobre el modelado, se encuentran las asignaturas Desarrollo Dirigido por Modelos (DSM) y

Análisis y Especificación de Requisitos (AER), ayudándonos a extraer las necesidades del cliente y obteniendo así los requisitos del sistema. Comienza por la elicitación, realizando entrevistas con los stakeholders, en nuestro caso entrenadores de un club de fútbol. Gracias a estas entrevistas se han podido identificar los requisitos, creando un modelo de dominio donde se representan los términos y relaciones de interés del problema. Además, se creó el diagrama de contexto para la identificación de los actores y de los límites del sistema. Después da comienzo la especificación de requisitos, mediante casos de uso. La especificación de requisitos nos facilita la comunicación con los desarrolladores siguiendo la sintaxis y semántica del UML. Obtenida una primera versión de requisitos, estos deben de ser validados por los *stakeholders* para asegurar que el software es adecuado. Se realizó una revisión de requisitos mediante reuniones con los entrenadores, y además se les realizó una demostración mediante un prototipo para poder comprobar su completitud.

La asignatura Integración e Interoperabilidad (IEI) nos ha aportado el conocimiento para la creación de las APIs en la capa de negocio.

Se ha elegido el proceso Test Driven Development (TDD) debido a la importancia de realizar *testing*. Se ha aplicado lo aprendido en la asignatura Análisis, Validación y Depuración de Software (AVD), intentando cubrir las distintas posibilidades que puede realizar una función.

La asignatura Mantenimiento del Software (MES), nos enseñó a realizar un buen control de versiones mediante la herramienta Git. Una vez el equipo aumente y la aplicación se encuentre en producción, se comenzará a hacer uso de Bugzilla, herramienta de seguimiento de errores [49] donde usuarios y desarrolladores subirán los bugs encontrados, facilitando la documentación de estos.

8. Trabajos futuros

Como continuación de la entrega de este trabajo, el proyecto puede continuar avanzando con características pendientes de implementar, desde las más básicas hasta las más ambiciosas. Es necesario añadir, de forma prioritaria, las características más destacadas de la competencia, que serán añadidas al *backlog*. Una característica diferenciadora, interesante, pero a la vez muy compleja sería el estudio de datos, con el objetivo de crear una Inteligencia Artificial para evitar lesiones, deduciendo en qué momento un jugador es probable que se lesione. Para llevar a cabo esto, es necesaria la ampliación del equipo, siendo importante la contratación de un científico de datos. Esta nueva característica nos permitirá negociar con clubes de mayor tamaño, quienes están interesados en este tipo de características.

Por otro lado, para el cumplimiento de las propuestas realizadas en el análisis de la competencia (Punto 3 de este documento) es necesaria la ampliación del equipo con un nuevo diseñador de interfaces de usuario (UX UI) encargado de crear interfaces mucho más profesionales, consiguiendo mejorar de manera notoria la experiencia del usuario.

9. Bibliografía

[1] «La tecnología,» [En línea]. Available: https://www.edu.xunta.gal/espazoAbalar/sites/espazoAbalar/files/datos/1464945204/contido/1_la_tecnologa.html. [Último acceso: 2 Mayo 2022].

[2] «Que son los avances tecnologicos,» [En línea]. Available: https://www.euroinnova.edu.es/blog/que-son-los-avances-tecnologicos. [Último acceso: 2 Mayo 2022].

[3] M. Á. Hernandez Briseño, Diferentes miradas sobre el empleo de las tecnologías de la información y la comunicación en educación, Red Durango de Investigadores Educativos, 2017.

[4] «¿Qué es el Desarrollo Ágil?,» [En línea]. Available: https://www.institutodemarketingagil.com/single-post/que-es-el-desarrollo-agil. [Último acceso: 3 Mayo 2022].

[5] R. S. Pressman, Ingeniería de Software un enfoque práctico, McGraw-Hill.

[6] K. Beck, Extreme Programming Explained, Addison-Wesley Professional.

[7] «Gesdep,» [En línea]. Available: http://www.gesdep.net/v3. [Último acceso: 5 Mayo 2022].

[8] «Demo Gesdep,» [En línea]. Available: https://demoges.net/apps/gestiondeportiva/web/Default.aspx. [Último acceso: 4 Mayo 2022].

[9] «easy2coach,» [En línea]. Available: https://www.easy2coach.net/es/. [Último acceso: 5 Mayo 2022].

[10] «Sportlyzer,» [En línea]. Available: https://www.sportlyzer.com/es/. [Último acceso: 4 Mayo 2022].

[11] T. Lloyd, «2016 Sports Technology Awards reveals best in sector,» [En línea]. Available: https://www.sportspromedia.com/announcements/2016_sports_technology_awards_reveals_best_in_sector/. [Último acceso: 4 Mayo 2022].

[12] «SportEasy,» [En línea]. Available: https://www.sporteasy.net/es/home/. [Último acceso: 6 Mayo 2022].

[13] «Llega a España SportEasy, una app de gestión deportiva para clubes y seguidores,» [En línea]. Available: https://as.com/meristation/2022/04/20/betech/1650479191_316249.html. [Último acceso: 6 Mayo 2022].

[14] «Fases y Flujos de trabajo en PUR,» [En línea]. Available: https://es.wikipedia.org/wiki/Archivo:Fases_y_Flujos_de_trabajo_en_PUR.svg.

[15] «Metodologias RUP,» [En línea]. Available: https://metodologiadesoftware.blogspot.com/2012/11/fases-del-modelo-rup_27.html. [Último acceso: 20 Mayo 2022].

[16] «Fases del modelo RUP,» [En línea]. Available: https://1library.co/article/fases-del-modelo-rup-metodolog%C3%ADa-rup-marco-teórico.y4g38ery. [Último acceso: 20 Mayo 2022].

[17] «Cidecame,» [En línea]. Available: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro10/334_fase_de_transicin.html. [Último acceso: 20 Mayo 2022].

[18] K. Molina Montero, H. Vite Cevallos y J. Dávila Cuesta, «Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software,» [En línea]. Available: https://www.researchgate.net/profile/Harry-Vite-Cevallos/publication/327537074_Metodologias_agiles_frente_a_las_tradicionales_en_el_proceso_de_desarrollo_de_software/links/5b942061a6fdccfd542a2b13/Metodologias-agiles-frente-a-las-tradicionales-en-el-proce. [Último acceso: 20 Mayo 2022].

[19] J. Joskowicz, «Reglas y Prácticas en eXtreme Programming,» [En línea]. Available: https://blog.facialix.com/wp-content/uploads/2022/03/XP-Jose-Joskowicz.pdf. [Último acceso: 20 Mayo 2022].

[20] J. Saez Hurtado, «Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla,» [En línea]. Available: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/. [Último acceso: 21 Mayo 2022].

[21] D. Calvo, «Metodología SCRUM (Metodología ágil),» [En línea]. Available: https://www.diegocalvo.es/metodologia-scrum-metodologia-agil/.

[22] O. Dieste, E. R. Fonseca, G. Raura y P. Rodríguez, «Efectividad del Test-Driven Development: Un Experimento Replicado,» [En línea]. Available: http://revistas.unla.edu.ar/software/article/view/706. [Último acceso: 23 Mayo 2022].

[23] T. Khine, «Red, Green, Refactor!,» [En línea]. Available: https://medium.com/@tunkhine126/red-green-refactor-42b5b643b506. [Último acceso: 24 Mayo 2022].

[24] K. Beck, Test-Driven Development By Example, Addison-Wesley.

[25] «Arquitectura de tres niveles,» [En línea]. Available: https://www.ibm.com/es-es/cloud/learn/three-tier-architecture. [Último acceso: 26 Mayo 2022].

[26] M. Dave, «SQL and NoSQL Databases,» [En línea]. Available: https://www.researchgate.net/profile/Meenu-Dave/publication/303856633_SQL_and_NoSQL_Databases/links/5758557f08ae9a9c954a7573/SQL-and-NoSQL-Databases.pdf. [Último acceso: 27 Mayo 2022].

[27] R. T. Mason, «NoSQL Databases and Data Modeling Techniques for a Document-oriented NoSQL Database,» [En línea]. Available: http://proceedings.informingscience.org/InSITE2015/InSITE15p259-268Mason1569.pdf. [Último acceso: 27 Mayo 2022].

[28] G. Covarrubias, «Escalamiento Vertical Vs Horizontal – Aquitectura de Sistemas,» [En línea]. Available: https://c4-technologies.com/escalamiento-vertical-vs-horizontal/. [Último acceso: 27 Mayo 2022].

[29] M. Masse, Rest API Design Rulebook, O'Reilly.

[30] «NestJS,» [En línea]. Available: https://desarrolloweb.com/home/nestjs. [Último acceso: 28 Mayo 2022].

[31] M. Dustano Contreras, G. Gutierrez y J. Diaz Vega, «Las buenas prácticas de los Framework en las Capas Arquitectónicas,» [En línea]. Available: https://repository.usta.edu.co/bitstream/handle/11634/13595/Dustano%20Mario%202016.pdf?sequence=1&isAllowed=y. [Último acceso: 28 Mayo 2022].

[32] «Controllers,» [En línea]. Available: https://docs.nestjs.com/controllers. [Último acceso: 29 Mayo 2022].

[33] «Providers,» [En línea]. Available: https://docs.nestjs.com/providers. [Último acceso: 29 Mayo 2022].

[34] D. Adhemar, «Prisma, un toolkit para bases de datos (¿ORM?) para TypeScript y Node.js,» [En línea]. Available: https://dev.to/denispixi/prisma-un-toolkit-para-bases-de-datos-orm-para-typescript-y-node-js-3g9. [Último acceso: 29 Mayo 2022].

[35] L. M. Avila, S. Aviles Matute y D. F. Avila Pesantez, «Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso,» [En línea]. Available: https://www.researchgate.net/profile/Diego-Avila-Pesantez/publication/346973093_Desarrollo_de_sistema_Web_basado_en_los_frameworks_de_Laravel_y_VueJs_para_la_gestion_por_procesos_Un_estudio_de_caso/links/5fde923992851c13fea37623/Desarrollo-de-sistema-Web-. [Último acceso: 29 Mayo 2022].

[36] «Composition API FAQ,» [En línea]. Available: https://vuejs.org/guide/extras/composition-api-faq.html. [Último acceso: 1 Junio 2022].

[37] J. C. López Muñoz, «De Vuex a Pinia, la gestión de estados definitiva para Vue,» [En línea]. Available: https://www.enmilocalfunciona.io/vuex-vs-pinia-7/. [Último acceso: 1 Junio 2022].

[38] «Tailwind CSS,» [En línea]. Available: https://desarrolloweb.com/home/tailwind-css. [Último acceso: 1 Junio 2022].

[39] «Atlas,» [En línea]. Available: https://www.mongodb.com/docs/atlas/. [Último acceso: 1 Junio 2022].

[40] «El backlog del producto: la lista de tareas pendientes definitiva,» [En línea]. Available: https://www.atlassian.com/es/agile/scrum/backlogs. [Último acceso: 1 Junio 2022].

[41] M. Jacobs, «¿Qué es Git?,» [En línea]. Available: https://docs.microsoft.com/es-es/devops/develop/git/what-is-git. [Último acceso: 3 Junio 2022].

[42] F. J. Lopez Pellicer, R. Bejar, M. A. Latre, J. Nogueras Iso y J. Zarazaga Soria, «GitHub como herramienta docente,» [En línea]. Available: https://upcommons.upc.edu/bitstream/handle/2117/76761/JENUI2015_76-83.pdf?sequence=1&isAllowed=y. [Último acceso: 4 Junio 2022].

[43] B. Moroz, «UNIT TEST AUTOMATION OF A REACT-REDUX APPLICATION WITH JEST AND ENZYME,» [En línea]. Available: https://www.theseus.fi/bitstream/handle/10024/184586/Moroz_Bogdan.pdf?sequence=2&isAllowed=y. [Último acceso: 4 Junio 2022].

[44] «Testing,» [En línea]. Available: https://2021.stateofjs.com/en-US/libraries/testing/. [Último acceso: 4 Junio 2022].

[45] V. Surwase, «REST API Modeling Languages - A Developer's Perspective,» [En línea]. Available: https://www.ijste.org/articles/IJSTEV2I10199.pdf. [Último acceso: 6 Junio 2022].

[46] C. Fernando, «Crea tu primer componente con Vue.js,» [En línea]. Available: https://dev.to/duxtech/crea-tu-primer-componente-con-vue-js-para-dummies-2n4c. [Último acceso: 8 Junio 2022].

[47] J. Song, M. Zhang y H. Xie, «Design and Implementation of a Vue.js-Based College Teaching System,» [En línea]. Available: https://www.researchgate.net/figure/Vuejs-component-system-schematic_fig2_334468164. [Último acceso: 8 Junio 2022].

[48] «Antipatrón de diseño,» [En línea]. Available: https://es.wikipedia.org/wiki/Antipatr%C3%B3n_de_dise%C3%B1o. [Último acceso: 2 Septiembre 2022].

[49] A. Ibañez Neri, «Validar/verificar: ¿Cuál es la diferencia?,» [En línea]. Available: https://academy.ibro-cvm.com/es/blog/validar-verificar-diferencia-b27.html. [Último acceso: 2 Septiembre 2022].

[50] «Pruebas de aceptación: el qué y el por qué,» [En línea]. Available: https://www.digite.com/es/agile/pruebas-de-aceptacion/.

[51] P. Letelier, «Desarrollo ágil basado en pruebas de aceptación,» [En línea]. Available: https://www.aragon.es/documents/20127/674325/3_AGIL1.pdf/2a6ac510-17ee-193c-420a-e840e086ed0c. [Último acceso: 2 Septiembre 2022].

[52] «Bugzilla,» [En línea]. Available: https://es.wikipedia.org/wiki/Bugzilla. [Último acceso: 10 Junio 2022].

[53] «Widget,» [En línea]. Available: https://es.wikipedia.org/wiki/Widget. [Último acceso: 3 Septiembre 2022].

[54] «¿Qué es Diseño UX/UI y qué hace un Diseñador UX/UI?,» [En línea]. Available: https://www.ironhack.com/es/diseno-ux-ui/que-es-y-que-hace-un-disenador-ux-ui. [Último acceso: 3 Septiembre 2022].

[55] «Lenguaje unificado de modelado,» [En línea]. Available: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado. [Último acceso: 3 Septiembre 2022].

[56] «Sprint Planning: Definición,» [En línea]. Available: https://blog.comparasoftware.com/sprint-planning-definicion-ejemplos/. [Último acceso: 3 Septiembre 2022].

[57] «Interfaz de programación de aplicaciones,» [En línea]. Available: https://es.wikipedia.org/wiki/Interfaz_de_programación_de_aplicaciones. [Último acceso: 3 Septiembre 2022].

[58] «Generalidades del protocolo HTTP,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Overview. [Último acceso: 3 Septiembre 2022].

[59] «Objetivos y metas de desarrollo sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/. [Último acceso: 2 Septiembre 2022].

[60] «17 objetivos para transformar nuestro mundo,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/. [Último acceso: 2 Septiembre 2022].

[61] «La obesidad y el sobrepeso provoca 2,8 millones de muertes al año,» [En línea]. Available: https://www.diariomedico.com/medicina/endocrinologia/la-obesidad-y-el-sobrepeso-provoca-28-millones-de-muertes-al-ano.html. [Último acceso: 15 Junio 2022].

[62] A. M. Lépez Sobaler, A. Aparicio, M. D. Salas González, V. Loria Kohen y L. M. Bermejo López, «Obesidad en la población infantil en España y factores asociados,» [En línea]. Available: https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0212-16112021000500007#:~:text=Los%20datos%20de%20la%20última,población%20infantil%20tiene%20exceso%20ponderal. [Último acceso: 15 Junio 2022].

[63] S. Melo, «¿Cuánto le está costando a tu negocio los residuos del papel?,» [En línea]. Available: https://datascope.io/es/blog/residuos-de-papel/. [Último acceso: 18 Junio 2022].

10. Glosario

***Widget*:** Es una pequeña aplicación que facilitan el acceso a funciones que son usadas frecuentemente [53].

AppStore: Tienda de aplicaciones creada por Apple. Permite a los usuarios de dispositivos Apple descargar aplicaciones.

Convocatoria: Selección de jugadores por parte del entrenador para disputar un partido.

***Framework*:** Es una plantilla que simplifica la elaboración de funcionalidades gracias a prácticas ya realizadas.

Diseñador UX (*User Experience*): Crear experiencias que cubren las necesidades de los clientes volviendo fácil, intuitivo, eficiente y bonito la el uso de la aplicación [54].

Diseñador UI (*User Interface*): Crear Diseño del producto buscando impacto en la primera impresión visual del usuario [54].

UML (*Unified Modeling Language*): Es lenguaje de modelado de software más conocido [55].

***Stakeholder*:** Son los actores interesados que pueden participar directa o indirectamente en el proyecto .

Refactorizar el código: Técnica de ingeniería que trata de reestructurar el código ya creado sin cambiar el comportamiento de la funcionalidad.

***Sprint*:** Periodo de tiempo determinado para realizar el trabajo acordado para alcanzar los objetivos propuestos [56].

API: Conjunto de métodos creados para ser consumidos por otro software [57].

***Plugins*:** Complementos que añaden mejoras a los programas. Estos no funcionan por sí solos.

HTTP (Protocolo de Transferencia de Hipertexto en español): Es el protocolo que permite realizar peticiones de datos y recursos [58].

Anexo 1: Especificación (Acceso a la aplicación)

Caso de uso	Crear usuario	CU-1
Actores	Usuario anónimo	
Resumen	El usuario rellena los datos necesarios para crear un acceso a la aplicación.	
Precondición	No	
Secuencia Principal	1. El usuario anónimo rellena los datos necesarios para crear la cuenta. 2. El sistema valida los datos. 3. Si los datos son correctos se crea la cuenta de usuario.	
Postcondición	Se crea una cuenta en la aplicación.	
Excepciones	3.1 Si los datos son incorrectos, el sistema pedirá que se rellenen los datos correctamente.	

Caso de uso	Crear club	CU-2
Actores	Usuario anónimo	
Resumen	El usuario anónimo rellena los datos para crear un nuevo club.	
Precondición	Haber rellenado los datos del CU-1	
Secuencia Principal	1. El usuario anónimo rellena los datos necesarios para crear un club. 2. El sistema valida los datos. 3. Si los datos son correctos se creará un nuevo club asociado a la cuenta creada en el CU-1	
Postcondición	Se crea un club en la aplicación.	
Excepciones	3.1 Si los datos son incorrectos, el sistema pedirá que se rellenen los datos correctamente.	

Caso de uso	Iniciar sesión	CU-3
Actores	Usuario anónimo	
Resumen	Un usuario anónimo accede a la aplicación con sus credenciales.	
Precondición	El usuario debe de tener una cuenta creada.	
Secuencia Principal	1. El usuario anónimo rellena los datos necesarios para acceder a la aplicación.	

	2. El sistema valida los datos. 3. Si los datos son correctos, el usuario accede a la aplicación con la cuenta correspondiente.
Postcondición	Acceder a la aplicación con su cuenta.
Excepciones	3.1 Si los datos introducidos son incorrectos, el sistema pedirá al usuario que introduzca otros datos.

Caso de uso	Cerrar sesión	CU-4
Actores	Usuario registrado	
Resumen	El usuario registrado cerrará la sesión de su cuenta.	
Precondición	Tener una sesión abierta.	
Secuencia Principal	1. El usuario presionará el botón de cerrar sesión. 2. El sistema finalizará la sesión, eliminará todos los datos guardados en el navegador del usuario y mostrará la pantalla de iniciar sesión	
Postcondición	Sesión cerrada y datos almacenados en el navegador eliminados	
Excepciones	No	

Anexo 2: Especificación (Gestión de equipos)

Caso de uso	Crear equipo	CU-5
Actores	Directiva	
Resumen	La directiva rellena los datos necesarios para crear un nuevo equipo en el club.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva.	
Secuencia Principal	1. La directiva entra en la página de equipos. 2. La directiva presiona el botón de crear equipo. 3. La directiva rellena los datos necesarios. 4. El sistema valida los datos. 5. El sistema crea el equipo si los datos son válidos.	
Postcondición	Se crea un nuevo equipo en el club.	
Excepciones	5.1 Si los datos no son válidos no se crea el equipo y se pide al usuario que rellene correctamente los datos.	

Caso de uso	Listar equipos	CU-6
Actores	Directiva	
Resumen	La directiva puede visualizar los equipos creados dentro del club.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva.	
Secuencia Principal	1. La directiva entra en la página de equipos para visualizar el listado de estos si existe. 2. El sistema muestra el listado de equipos.	
Postcondición	Se visualiza el listado de equipos	
Excepciones	No	

Caso de uso	Visualizar equipo	CU-7
Actores	Directiva	
Resumen	La directiva visualiza los detalles de un equipo creado dentro del club.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva. La directiva debe de haber creado el equipo previamente.	

Secuencia Principal	1. La directiva entra en la página de equipos. 2. El sistema muestra el listado de equipos. 3. La directiva selecciona un equipo. 4. El sistema muestra los detalles del equipo seleccionado.
Postcondición	Se muestra los detalles del equipo seleccionado
Excepciones	No

Anexo 3: Especificación (Gestión de entrenadores)

Caso de uso	Registrar entrenador	CU-8
Actores	Directiva	
Resumen	La directiva rellena los datos necesarios para crear un nuevo entrenador en el club.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva.	
Secuencia Principal	1. La directiva entra en la página de entrenadores. 2. El sistema muestra el listado de entrenadores. 3. La directiva presiona en el botón de crear un entrenador. 4. El sistema muestra el formulario para crear el entrenador. 5. La directiva rellena los campos necesarios. 6. El sistema valida los campos. 7. Si los campos son correctos, el sistema crea el entrenador.	
Postcondición	El sistema crea el entrenador	
Excepciones	7.1 Si los datos son incorrectos el sistema pedirá que se escriban datos correctos y no creará el entrenador.	

Caso de uso	Seleccionar equipos	CU-9
Actores	Directiva	
Resumen	La directiva selecciona el o los equipos de fútbol que entrena el entrenador.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva. La directiva debe de haber creado mínimo un equipo.	
Secuencia Principal	1. La directiva presiona en el campo de selección de equipos. 2. El sistema muestra los equipos del club. 3. La directiva selecciona los equipos que el entrenador entrena. 4. El sistema asocia al entrenador estos equipos.	
Postcondición	Se asocian los equipos al entrenador	
Excepciones	No	

Caso de uso	Listar entrenadores	CU-10
Actores	Directiva	
Resumen	La directiva visualiza los entrenadores que se han creado en el club.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva.	
Secuencia Principal	1. La directiva accede a la página de entrenadores. 2. El sistema muestra el listado de entrenadores del club.	
Postcondición	Se muestra el listado de entrenadores del club.	
Excepciones	No	

Caso de uso	Visualizar entrenador	CU-11
Actores	Directiva	
Resumen	La directiva visualiza los detalles de un entrenador creado previamente.	
Precondición	Se debe de haber iniciado sesión con una cuenta de directiva. La directiva debe de haber creado el entrenador.	
Secuencia Principal	1. La directiva accede a la página de entrenadores. 2. El sistema muestra el listado de entrenadores del club. 3. La directiva selecciona el entrenador. 4. El sistema muestra los detalles del entrenador seleccionado	
Postcondición	Se muestran los detalles del entrenador seleccionado	
Excepciones	No	

Anexo 4: Especificación (Gestión de jugadores)

Caso de uso	Dar alta jugador	CU-12
Actores	Entrenador	
Resumen	El entrenador añade un nuevo jugador a su equipo añadiendo los datos del jugador. Una vez dado de alta, el jugador puede tener acceso a la plataforma desde la vista del jugador	
Precondición	Haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al apartado de jugadores. 2. El sistema muestra los jugadores. 3. El entrenador presiona el botón de añadir jugador. 4. El sistema muestra el formulario. 5. El entrenador rellena los datos necesarios. 6. Si los datos son correctos, el sistema creará al jugador.	
Postcondición	Se crea al jugador	
Excepciones	6.1 Si los datos son incorrectos, se pedirá que se rellenen correctamente y no se creará el jugador.	

Caso de uso	Dar baja jugador	CU-13
Actores	Entrenador	
Resumen	El entrenador da de baja a un usuario de su mismo equipo. Una vez dado de baja, este ya no podrá acceder a la plataforma.	
Precondición	Haber iniciado sesión con una cuenta de entrenador. El entrenador debe de ser el entrenador del jugador.	
Secuencia Principal	1. El entrenador accede al apartado de jugadores y selecciona al jugador. 2. El sistema muestra los detalles del jugador junto con un botón de eliminar y editar. 3. El entrenador presiona el botón de eliminar 4. El sistema elimina al jugador seleccionado.	
Postcondición	Se elimina al jugador.	
Excepciones	No	

Caso de uso	Editar datos de jugador	CU-14
Actores	Jugador, entrenador	
Resumen	El entrenador puede editar los datos personales de los jugadores de su equipo. El jugador puede editar sus datos personales.	
Precondición	El jugador debe de iniciar sesión o el entrenador debe de haber iniciado sesión y pertenecer al equipo del jugador.	
Secuencia Principal	1. El entrenador accede a la vista de jugadores, selecciona al jugador y presiona el botón editar. 2. El sistema muestra el formulario para editar al jugador. 3. El entrenador modifica los datos y presiona el botón de aceptar. 4. Si los datos son correctos, el sistema los modificará. 1. El jugador presiona el botón editar. 2. El sistema muestra el formulario para editar los datos. 3. El jugador modifica los datos y presiona el botón aceptar. 4. Si los datos son correctos el sistema los modificará.	
Postcondición	Se modificán los datos de un usuario.	
Excepciones	8.1 y 4.1 Si los datos son incorrectos, el sistema no los modificará y mostrará un mensaje de error.	

Caso de uso	Visualizar datos de jugador	CU-15
Actores	Jugador, entrenador	
Resumen	El entrenador puede visualizar los datos personales de los jugadores de su equipo. El jugador puede visualizar sus datos personales.	
Precondición	El jugador debe de iniciar sesión o el entrenador debe de haber iniciado sesión y pertenecer al equipo del jugador.	
Secuencia Principal	1. El entrenador accede al apartado de jugadores. 2. El sistema muestra los jugadores. 3. El entrenador selecciona al jugador. 4. El sistema muestra los detalles del jugador junto con un botón de eliminar y editar. 1. El jugador accede a su pantalla principal. 2. El sistema le muestra sus datos.	
Postcondición	Se muestran los datos del jugador.	
Excepciones	No	

Caso de uso	Listar jugadores	CU-16
Actores	Entrenador	
Resumen	El entrenador puede visualizar los jugadores de su equipo.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al apartado de jugadores. 2. El sistema muestra los jugadores.	
Postcondición	Se muestra el listado de jugadores.	
Excepciones	No	

Anexo 5: Especificación (Gestión de ejercicios)

Caso de uso	Crear ejercicio	CU-17
Actores	Entrenador	
Resumen	El entrenador añade un nuevo ejercicio al sistema pudiendo añadir este en los entrenamientos. Además, este ejercicio aparecerá a los demás entrenadores del club	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede a la vista entrenamientos presiona el botón de crear entrenamiento y después presiona el botón de crear un ejercicio. 2. El sistema muestra el formulario de crear un ejercicio. 3. El entrenador rellena los datos del formulario y presiona el botón de crear. 4. Si los datos son correctos el sistema creará el ejercicio y mostrará la ventana del formulario de entrenamiento.	
Postcondición	Se creará un nuevo ejercicio.	
Excepciones	4.1 Si los datos son incorrectos, se mostrará un mensaje de error y no se creará el ejercicio.	

Caso de uso	Visualizar ejercicios	CU-18
Actores	Entrenador	
Resumen	Todos los entrenadores de un mismo club pueden visualizar los ejercicios que pertenecen a este.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al formulario de crear un entrenamiento y presiona el botón selección de ejercicios. 2. El sistema mostrará el listado de ejercicios del club	
Postcondición	Se mostrará el listado de ejercicios.	
Excepciones	No	

Caso de uso	Seleccionar ejercicio	CU-19
Actores	Entrenador	
Resumen	Todos los entrenadores de un mismo club pueden seleccionar los ejercicios.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al formulario de crear un entrenamiento y presiona el botón selección de ejercicios. 2. El sistema mostrará el listado de ejercicios del club. 3. El entrenador seleccionará los ejercicios.	
Postcondición	El entrenador selecciona los ejercicios.	
Excepciones		

Anexo 6: Especificación (Gestión de videos)

Caso de uso	Subir video	CU-20
Actores	Entrenador	
Resumen	El entrenador puede añadir videos dentro de la plataforma para que los jugadores de su equipo puedan visualizarlos.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al formulario de crear entrenamiento y selecciona el video a subir. 2. El sistema sube el video al servidor.	
Postcondición	Se sube el video al servidor y se añade al entrenamiento.	
Excepciones		

Caso de uso	Visualizar el video	CU-21
Actores	Entrenador, Jugador	
Resumen	Tanto el entrenador como el jugador pueden visualizar el video que pertenecen al equipo	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador o jugador. El entrenamiento seleccionado debe de tener un video.	
Secuencia Principal	1. El entrenador o jugador acceden al calendario de entrenamientos y seleccionan el entrenamiento a visualizar. 2. El sistema muestra los detalles del entrenamiento junto con el video.	
Postcondición	Se muestra el video del entrenamiento	
Excepciones		

Caso de uso	Eliminar video	CU-22
Actores	Entrenador	
Resumen	El entrenador puede eliminar el video dentro de la plataforma y perteneciente a su equipo.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador. El entrenamiento seleccionado debe de tener un video.	
Secuencia Principal	1. El entrenador accede al calendario de entrenamientos y seleccionan el entrenamiento a visualizar. 2. El sistema muestra los detalles del entrenamiento junto con el video. 3. El entrenador presiona el botón de editar entrenamiento. 4. El sistema muestra el formulario de editar. 5. El entrenador elimina el video presionando el botón.	
Postcondición	El video se elimina.	
Excepciones		

Anexo 7: Especificación (Gestión de entrenamiento)

Caso de uso	Crear entrenamiento	CU-23
Actores	Entrenador	
Resumen	El entrenador añade un nuevo entrenamiento a su equipo. Una vez creado, puede añadir ejercicios a este y los jugadores pueden visualizar el entrenamiento.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede a la vista de entrenamientos y presiona el botón de crear entrenamiento. 2. El sistema carga el formulario. 3. El entrenador rellena el formulario y presiona el botón aceptar. 4. El sistema valida los datos del formulario. 5. Si los datos son correctos, el sistema crea el entrenamiento.	
Postcondición	Se crea un entrenamiento	
Excepciones	5.1 Si los datos son incorrectos, se muestra un mensaje con los datos erróneos y no se crea el entrenamiento.	

Caso de uso	Editar entrenamiento	CU-24
Actores	Entrenador	
Resumen	El entrenador puede editar un entrenamiento perteneciente a su equipo, pudiendo añadir ejercicios y editar los datos del entrenamiento.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede a la vista de entrenamientos y selecciona un entrenamiento del calendario. 2. El sistema muestra los detalles del entrenamiento junto con el botón de editar o eliminar. 3. El entrenador presiona el botón de editar. 4. El sistema muestra el formulario para editar el entrenamiento. 5. El entrenador modifica los datos del formulario. 6. El sistema valida los datos. 7. Si los datos son correctos, el sistema edita el entrenamiento.	
Postcondición	Se edita un entrenamiento.	

Excepciones	7.1 Si los datos son incorrectos, el sistema muestra un mensaje de error y no edita el entrenamiento.

Caso de uso	Eliminar entrenamiento	CU-25
Actores	Entrenador	
Resumen	El entrenador puede eliminar un entrenamiento perteneciente a su equipo.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede a la vista de entrenamientos y selecciona un entrenamiento del calendario. 2. El sistema muestra los detalles del entrenamiento junto con el botón de editar o eliminar. 3. El entrenador presiona el botón de eliminar. 4. El sistema elimina el entrenamiento.	
Postcondición	Se elimina un entrenamiento.	
Excepciones	No	

Caso de uso	Visualizar entrenamiento	CU-26
Actores	Entrenador, jugador	
Resumen	Tanto el jugador como el entrenador pueden visualizar los detalles del entrenamiento, siempre que pertenezcan al equipo.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede a la vista de entrenamientos y selecciona un entrenamiento del calendario. 2. El sistema muestra los detalles del entrenamiento junto con el botón de editar o eliminar.	
Postcondición	Se muestran los detalles de un entrenamiento.	
Excepciones	No	

Caso de uso	Visualizar entrenamientos del calendario	CU-27
Actores	Entrenador, jugador	
Resumen	Tanto el jugador como el entrenador pueden visualizar los entrenamientos desde la visión de un calendario, siempre que pertenezcan al equipo.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador o jugador.	

Secuencia Principal	1. El entrenador o jugador accede a la vista de entrenamientos. 2. El sistema carga los entrenamientos en el calendario.
Postcondición	Se muestran los entrenamientos en el calendario.
Excepciones	

Caso de uso	Añadir entrenamiento al calendario	CU-28
Actores	Entrenador	
Resumen	El entrenador añade un entrenamiento ya creado al calendario. Este programa la sesión de este entrenamiento, permitiendo que los jugadores lo visualicen.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al entrenamiento y presiona el botón de editar. 2. El sistema muestra el formulario del entrenamiento. 3. El entrenador selecciona una fecha y presiona aceptar. 4. El sistema valida si la fecha seleccionada tiene un formato correcto. 5. Si el formato es correcto, el sistema añade el entrenamiento al calendario.	
Postcondición	Se añade un entrenamiento al calendario.	
Excepciones	5.1 Si el formato es incorrecto, el sistema muestra un mensaje de error y no añade el entrenamiento al calendario.	

Caso de uso	Eliminar entrenamiento del calendario	CU-29
Actores	Entrenador	
Resumen	El entrenador puede eliminar un entrenamiento programado del calendario.	
Precondición	Se debe de haber iniciado sesión con una cuenta de entrenador.	
Secuencia Principal	1. El entrenador accede al entrenamiento y presiona el botón de editar. 2. El sistema muestra el formulario del entrenamiento. 3. El entrenador elimina la fecha y presiona aceptar. 4. El sistema valida si la fecha seleccionada tiene un formato correcto. 5. El sistema elimina el entrenamiento del calendario.	
Postcondición	Se elimina el entrenamiento del calendario	
Excepciones	No	

Anexo 8: Código nivel negocio

```
@ApiTags('Workout')
@Controller('workout')
export class WorkoutController {
  constructor(private workoutService: WorkoutService) {}

  @Get()
  @UseGuards(AuthGuard('jwt'))
  @Roles(Role.Trainer, Role.Player)
  async getWorkouts(@Req() { query }: any) {
    console.log(query);
    return await this.workoutService.getAllWorkouts(query.teamId);
  }
  @Get('/multiple')
  @UseGuards(AuthGuard('jwt'))
  @Roles(Role.Trainer, Role.Player)
  async getMultipleWorkouts(@Req() { query }: any) {
    console.log(query);
    const teams: any = [];
    for (const teamId of query.teamsId) {
      const prueba = await this.workoutService.getAllWorkouts(teamId);
      teams.push(prueba);
    }

    return teams.flat();
  }

  @Post('')
  @UseGuards(AuthGuard('jwt'), RolesGuard)
  @Roles(Role.Trainer)
  async newWorkout(@Body() workout: Workout): Promise<void> {
    return await this.workoutService.newWorkout(workout);
  }

  @Put()
  @UseGuards(AuthGuard('jwt'), RolesGuard)
  @Roles(Role.Trainer)
  async updateWorkout(@Req() { body }: any) {
    const { id, data } = body;
    console.log('component ', id, data);
    return await this.workoutService.updateWorkout(id, data);
  }

  @Delete()
  @UseGuards(AuthGuard('jwt'), RolesGuard)
  @Roles(Role.Admin, Role.Trainer)
  async deleteWorkout(@Req() { query }: any) {
    return await this.workoutService.deleteWorkout(query.id);
  }
}
```

Figura 34 Código del controlador workout.

```
@Injectable()
export class WorkoutService {
  constructor(private prismaService: PrismaService) {}

  async getAllWorkouts(teamId: string) {
    return await this.prismaService.workout.findMany({
      where: {
        teamId,
      },
    });
  }

  async newWorkout(workout: Workout) {
    await this.prismaService.workout.create({ data: workout });
  }

  async updateWorkout(id, data: Workout) {
    console.log('service ', id, data);
    await this.prismaService.workout.update({
      where: { id },
      data,
    });
  }

  async deleteWorkout(id: string) {
    await this.prismaService.workout.delete({
      where: {
        id,
      },
    });
  }
}
```

Figura 35 Código del servicio workout.

```
model Workout {
  id          String    @id @default(auto()) @map("_id") @db.ObjectId
  teamId      String?   @db.ObjectId
  name        String?
  description String?
  team        Team?     @relation(fields: [teamId], references: [id])
  date        DateTime?
  createdAt   DateTime? @default(now())
  updatedAt   DateTime? @updatedAt
  video       String?
  exerciseId  String[]  @db.ObjectId
  exercise    Exercise? @relation(fields: [exerciseId], references: [id])
}

model Exercise {
  id          String    @id @default(auto()) @map("_id") @db.ObjectId
  name        String?
  description String?
  duration    Int?
  material    String[]
  video       String?
  club        Club?     @relation(fields: [clubId], references: [id])
  clubId      String?   @db.ObjectId
  Workout     Workout[]
}

model Team {
  id             String    @id @default(auto()) @map("_id") @db.ObjectId
  name           String
  category       String
  description    String
  Person_players Person[]
  Club           Club?     @relation(fields: [clubId], references: [id])
  clubId         String    @db.ObjectId
  Training       Workout[]
}

model Club {
  id       String     @id @default(auto()) @map("_id") @db.ObjectId
  name     String
  phone    String
  email    String
  address  String
  Team     Team[]
  Person   Person[]
  Exercise Exercise[]
}
```

Figura 36 Esquema de base de datos.

```
describe('LoginController', () => {
  let controller: LoginController;
  let service: LoginService;

  beforeEach(async () => {
    const module: TestingModule = await Test.createTestingModule({
      controllers: [LoginController],
      providers: [LoginService, PrismaService, UserService, JwtService],
    }).compile();

    controller = module.get<LoginController>(LoginController);
    service = module.get<LoginService>(LoginService);
  });

  it('should be defined', () => {
    expect(controller).toBeDefined();
  });
  describe('Posts', () => {
    it('Should return mail and password is valid', async () => {
      const user: User = {
        email: 'prueba@prueba.com',
        password: '1234Prueba',
      };

      expect(service.isValidMail(user.email)).toBe(true);
      expect(service.isValidPassword(user.password)).toBe(true);
    });
    it('Should return mail is not valid and password is valid', async () => {
      const user: User = {
        email: 'prueba.hola',
        password: '1234Prueba',
      };
      expect(service.isValidMail(user.email)).toBe(false);
      expect(service.isValidPassword(user.password)).toBe(true);
    });
    it('Should return mail is valid and password is not valid', async () => {
      const user: User = {
        email: 'prueba@prueba.com',
        password: '123',
      };
      expect(service.isValidMail(user.email)).toBe(true);
      expect(service.isValidPassword(user.password)).toBe(false);
    });
  });
});
```

Figura 37 Código de testing.

Anexo 9: ODS

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

Los Objetivos de Desarrollo Sostenible son 17 objetivos para conseguir un futuro más sostenible [59]. Estos objetivos fueron aprobados por la ONU en el año 2015 [60]. El presente trabajo está relacionado con 3 objetivos, el objetivo 3 (Salud y bienestar), el objetivo 5 (Igualdad de género) y el objetivo número 13 (Acción por el clima).

Objetivo 3: Salud y bienestar

Garantizar una vida sana y promover el bienestar para todos en todas las edades.

Toach App es una aplicación de fútbol que promueve la actividad física, ayudando a mejorar a nivel de salud y de bienestar. Existen una gran cantidad de enfermedades provocadas por la ausencia de la actividad física, como por ejemplo la obesidad y el sobrepeso, causante de 2,8 millones de muertes al año [61]. En España, el 25% de la población adulta tiene sobrepeso y según el estudio ALADINO un 23,3% de escolares españoles también tienen sobrepeso y el 17,3% tienen obesidad. Estas cifras son alarmantes, mostrando la falta de deporte que presenta la población española [62]. Por esto, es muy importante la creación de aplicaciones que ayudan y motivan a los usuarios a realizar cualquier actividad deportiva. El fútbol es uno de los deportes más famosos y practicados del mundo, debido a que solamente es necesario una pelota y unas marcas para crear una portería, gracias a esto países tercermundistas pueden jugar al fútbol, mejorando su salud y disfrutando del juego en equipo.

Objetivo 5: Igualdad de género

Lograr la igualdad entre los géneros y empoderar a todas las mujeres y niñas.

El fútbol es un deporte practicado mayoritariamente por el género masculino, vivimos en una sociedad donde de pequeños no estaba bien visto que una mujer jugase al fútbol. Poco a poco esta visión ha ido desapareciendo, encontrando cada vez más jugadores en los campos de fútbol. El fútbol femenino español dio ejemplo consiguiendo la espectacular cifra de 91.500 espectadores, logrando así el record mundial de asistencias a un partido femenino. Toach App, está a favor de la igualdad de género, promoviendo la inclusión de la mujer en este deporte desde la aplicación, utilizando un lenguaje inclusivo y añadiendo contenido para dar la publicidad que se merece al fútbol femenino.

Objetivo 13: Acción por el clima

Adoptar medidas urgentes para combatir el cambio climático y sus efectos.

En el año 2020, el consumo de papel y cartón alcanzo la escalofriante cifra de 6,5 millones de toneladas. Según Resource Conservation Alliance, el 40% de los árboles talados se usan para producir papel [63]. Los entrenadores de fútbol utilizan papel para realizar los apuntes rápidos en el terreno de juego, La creación de una App en la que un entrenador puede realizar apuntes rápidos desde su propio dispositivo móvil o incluso desde su propia tablet, favorece al planeta reduciendo el consumo innecesario de papel y a su vez reduciendo la tala masiva de árboles.