UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Hércules: una aplicación de gestión de la actividad deportiva en universidades

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Ferrer Sánchez-Barcaiztegui, Javier

Tutor/a: Pelechano Ferragud, Vicente

CURSO ACADÉMICO: 2022/2023

Resum

El present projecte té com a objectiu el desenvolupament d'una aplicació web de gestió d'activitat esportiva en universitats que permeta als gestors administrar instal·lacions, activitats i competicions esportives, així com als usuaris realitzar reserves i consultar informació sobre les activitats i competicions. El projecte s'ha desenvolupat usant una metodologia àgil, concretament Scrum, i s'ha dividit en tres sprints. Les tecnologies utilitzades per al desenvolupament han sigut React, Typescript, Express i MySQL. També s'ha utilitzat Visual Studio Code com a editor de text i Git com a sistema de control de versions.

L'aplicació resultant compleix tots els requisits plantejats i s'han aplicat proves d'integració i analitzadors estàtics per garantir la qualitat del codi.

Paraules clau: Web, Typescript, React, Aplicació, Esport, Àgil

Resumen

El presente proyecto tiene como objetivo el desarrollo de una aplicación web de gestión de actividad deportiva en universidades que permita a los gestores administrar instalaciones, actividades y competiciones deportivas, así como a los usuarios realizar reservas y consultar información sobre las actividades y competiciones. El proyecto se ha desarrollado usando una metodología ágil, concretamente Scrum, y se ha dividido en tres sprints. Las tecnologías utilizadas para el desarrollo han sido React, Typescript, Express y MySQL. También se ha utilizado Visual Studio Code como editor de texto y Git como sistema de control de versiones.

La aplicación resultante cumple todos los requisitos planteados y se han aplicado pruebas de integración y analizadores estáticos para garantizar la calidad del código.

Palabras clave: Web, Typescript, React, Aplicación, Deporte, Ágil

Abstract

The objective of this project is to develop a sports management web app for universities. The app will enable managers to oversee sports facilities, activities, and competitions. Additionally, users will be able to book facilities and access information regarding activities and competitions. The project has been developed using Scrum, an agile methodology, and has been divided into three sprints. The technologies employed include React, TypeScript, Express, and MySQL. Moreover, Visual Studio Code and Git have been used for code editing and version control, respectively.

The resulting application meets all the requirements and includes the use of integration tests and static analyzers to ensure the quality of the code.

Key words: Web, Typescript, React, Application, Sport, Agile

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En la era de la digitalización, el acceso global a Internet ha transformado radicalmente la forma en que interactuamos con el mundo que nos rodea. Con la digitalización, lo que antes suponía un gran esfuerzo, ahora se puede hacer con un solo clic, incluyendo los trámites y gestiones, que antes requerían desplazarse físicamente a un lugar concreto. En consecuencia, la digitalización ha mejorado la eficiencia y la velocidad de los procesos, lo cual ha tenido un impacto positivo en la calidad de vida de las personas.

El evento de la pandemia de la COVID-19 aceleró la digitalización de muchos procesos, ya que las restricciones de movilidad obligaron a las personas a buscar alternativas digitales para realizar sus gestiones. En este escenario, tener soluciones en línea se convirtió en una necesidad para garantizar la seguridad de las personas y la eficiencia de los procesos.

En el presente entorno, las universidades no se quedan atrás. La transformación digital de los procesos universitarios se ha vuelto imprescindible con el fin de mejorar la eficiencia y la rapidez en las operaciones, lo que puede repercutir positivamente en la experiencia de los estudiantes. Por tanto, las universidades están adoptando estrategias de digitalización para agilizar diversas áreas, incluso la gestión de actividades deportivas.

Sin embargo, implementar la digitalización de las actividades deportivas en las universidades presenta desafíos significativos. Las instituciones académicas tienen una amplia gama de instalaciones deportivas y ofrecen muchas opciones para actividades físicas, lo que dificulta su administración. Adicionalmente, al tratarse de un servicio adicional a la actividad académica, la gestión deportiva no suele ser una prioridad para las universidades. Como resultado, los servicios de gestión deportiva suelen ser ineficientes, no satisfacen las necesidades de los usuarios y no fomentan la participación de los estudiantes debido a su interfaz poco amigable. Sumado a eso, algunos de los procesos se siguen realizando de forma manual y presencial, lo que puede ser tedioso y causar retrasos en la gestión.

El objetivo de crear una aplicación de gestiones deportivas para las universidades moderna, rápida y completamente digital da origen a la aplicación Hércules.

1.1 Motivación

Desde que empecé a estudiar en la Universidad Politécnica de Valencia, he sido socio de deportes. Esto me ha permitido participar en una gran variedad de actividades deportivas organizadas por la universidad, como clases de tenis, crossfit y musculación y he participado en numerosas competiciones de tenis, dónde se compite en una liga contra otros jugadores del mismo nivel. También he reservado numerosas veces pistas de tenis y pádel para jugar con mis amigos, especialmente durante la pandemia del COVID-19, donde sustituí los planes de quedar para cenar o ir al cine por partidas de tenis y pádel, ya que eran actividades al aire libre y más seguras. En consecuencia, he tenido que usar la web del servicio de deportes de la universidad durante mucho tiempo y he podido observar sus fallos y mejoras.

El diseño de la página web usado actualmente por la Universidad Politécnica de Valencia presenta un aspecto desactualizado, reflejando un enfoque visual poco atractivo y poco funcional en comparación con los estándares actuales de diseño web. Comunicándome con mis compañeros de la universidad he podido comprobar que muchos de ellos no son conscientes de la existencia de competiciones y actividades organizadas por la universidad y, por lo tanto, no participan en ellas. Se le puede atribuir este hecho al diseño poco atractivo, ya que es carente de estética contemporánea y no incita a los usuarios a explorar la página web.

El diseño desactualizado me ha afectado personalmente, sobre todo cuando se navega la aplicación usando un dispositivo móvil, ya que la aplicación web no es *responsive*. Cuando se accede con un dispositivo móvil te redirige a una página distinta, por lo que cambia completamente la apariencia y la distribución de los elementos, lo que causa mucha confusión para los usuarios. El diseño es completamente distinto y es igual de anticuado que el de la versión de escritorio, como si estuviera desarrollado para un dispositivo móvil de hace 10 años. Cuando participé en la liga de tenis siempre que nos tocaba al contrincante y a mí comprobar el estado de una competición o el nombre de la pista que habíamos reservado usando el móvil (ya que nos encontrábamos en la pista sin acceso a un PC), hemos sentido y comentado el sentimiento de frustración que nos causa el diseño de la página web.

Durante la pandemia de la COVID-19, las competiciones de tenis era una de las pocas actividades que podía realizar, ya que se trataba de una actividad al aire libre y segura. Sin embargo, el acta de los partidos se rellenaba por ambos jugadores de forma manual, con papel y boli, en el pabellón. Además de ser un proceso lento y tedioso, ya que una persona del área de competiciones tiene que transcribir los resultados de la hoja de papel a un programa donde se gestionan las competiciones (sobre todo cuando el programa y la hoja tienen *inputs* idénticos), en su momento me pareció un proceso poco seguro, ya que se compartía el mismo boli y papel entre todos los jugadores, y además se pasaba a estar en un espacio cerrado: el pabellón. Otro problema del sistema actual es que hasta que el personal del área de competiciones no transcriba los resultados al programa (que puede llegar a tardar varios días si hay un fin de semana de por medio), la clasificación se encuentra desactualizada. Aunque la pandemia ya no está tan presente

en la sociedad, muchos de los problemas siguen presentes, y fueron momentos como estos los que me motivaron a desarrollar Hércules.

En cuanto al ámbito tecnológico, llevaba un tiempo queriendo desarrollar una aplicación con un backend usando *Express*, ya que es una librería muy popular y quería aprender a usarla. También quería aprender a usar la última versión de *Next.js 13*, ya que introducía la posibilidad de definir estructuras de interfaces de usuario, y así poder desarrollar aplicaciones web más complejas. Por último, quería aprender más sobre *TypeScript*, ya que es un lenguaje de programación que está ganando mucha popularidad y no lo estaba usando hasta su completo potencial.

Finalmente, siempre he querido probar a aplicar una guía de diseño estándar de la industria del software, especialmente la guía de diseño *Material Design* de *Google*, ya que es una de las más populares y lanzó una nueva versión en 2021.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación web que permita a los usuarios gestionar la actividad deportiva en campus deportivos de universidades. Para ello, se han definido los siguientes objetivos secundarios:

- Unificar la gestión de la actividad deportiva en una única aplicación web.
- Permitir a los socios reservar instalaciones deportivas, apuntarse a actividades y competiciones deportivas.
- La aplicación debe permitir a los administradores gestionar las instalaciones deportivas, actividades y competiciones deportivas.
- Permitir a los participantes de las competiciones deportivas ver su estado en la competición.
- La aplicación debe ser fácil de usar y adaptarse a las necesidades de los usuarios.
- La aplicación debe conectar los horarios de las instalaciones con las actividades y competiciones para tener un único origen de la verdad.
- Usar tecnologías modernas para mejorar la usabilidad, eficiencia y velocidad.
- Diseñar una interfaz de usuario intuitiva, fácil de usar y atractiva. Además de tener un rendimiento óptimo.
- La aplicación debe promover la participación de los estudiantes en las competiciones deportivas y actividades organizadas por la universidad.

Con estos objetivos se ha analizado el problema, identificado los requisitos, se ha diseñado la arquitectura y se ha establecido un proceso de desarrollo para desarrollar la aplicación con éxito.

1.3 Estructura de la memoria

Los próximos capítulos tendrán la siguiente estructura:

- **Estado del arte**: Se compararán varias aplicaciones del mismo ámbito, comentando sus ventajas y desventajas, y se analizará que funcionalidades son las más importantes.
- **Análisis del problema**: Se identificará los problemas que se quieren resolver y se planteará soluciones para resolverlos, además de hacer un modelado conceptual de la aplicación.
- **Proceso de desarrollo del software**: Se explicará el proceso de desarrollo que se ha seguido para desarrollar la aplicación.
- **Especificación de requisitos**: Se especificarán los requisitos funcionales y no funcionales de la aplicación.
- **Diseño de la aplicación**: Se detallará la arquitectura de la aplicación y el diseño de la base de datos.
- **Diseño de la interfaz de usuario**: Se explicará el diseño de la interfaz de usuario y se mostrarán los bocetos de las pantallas más importantes.
- **Desarrollo de la aplicación**: Se explicarán las tecnologías usadas para el desarrollo de la aplicación, la estructura de la aplicación y los patrones de diseño usados.
- **Pruebas**: Se explicará el proceso de pruebas que se ha seguido para asegurar la calidad de la aplicación.
- **Aplicación final**: Se mostrará la aplicación final y se explicarán sus funcionalidades.
- **Conclusiones**: Se comentarán los resultados obtenidos y se propondrán posibles mejoras.

CAPÍTULO 2
Estado del Arte

Existen muchas aplicaciones para gestionar instalaciones, manejar ligas, torneos y actividades, pero ninguna de ellas realiza todas las funciones en un ámbito universitario. En este capítulo se analizarán las aplicaciones más populares para determinar que funcionalidades son las más importantes, cuáles se pueden descartar y cuáles harán que Hércules destaque sobre las demás.

2.1 Aplicaciones similares

2.1.1. Universidad Politécnica de Valencia

Actualmente la Universidad Politécnica de Valencia (UPV) dispone de dos aplicaciones para manejar los servicios deportivos. La primera es una aplicación web desarrollada usando HTML, JavaScript y CSS. Esta aplicación es la que utilizan los socios para reservar pistas e inscribirse a actividades y competiciones. En la figura 2.1 se puede ver como es la página de competiciones de la página web actual de la UPV.

Aunque la aplicación web solo tiene una funcionalidad básica, tiene un diseño poco intuitivo que dificulta la navegación y la búsqueda de información. Además, cuando se accede a la aplicación desde un dispositivo móvil, la aplicación cambia de aspecto completamente, ya que se carga un "tema móvil" implementado usando la versión 1.1.0 del *framework jQuery mobile*. Este *framework* de desarrollo de aplicaciones web para dispositivos móviles es obsoleto y no se actualiza desde 2014.

En cuanto a la aplicación utilizada por los administradores para gestionar las instalaciones, actividades y competiciones, es una aplicación de escritorio desarrollada en Java llamada *Llucena*. Esta aplicación tampoco tiene un diseño moderno. Además, no es responsive y no se puede utilizar desde dispositivos móviles. En la figura 2.2 se puede ver la pantalla de gestión de instalaciones de la aplicación Llucena.

Otro de los problemas del sistema actual es que a pesar de disponer de toda la información en línea, todavía existen muchos procedimientos que se realizan a mano rellenando hojas de papel. Por ejemplo, para escribir el acta de un partido es necesario que los jugadores rellenen una hoja de papel con los resultados y la

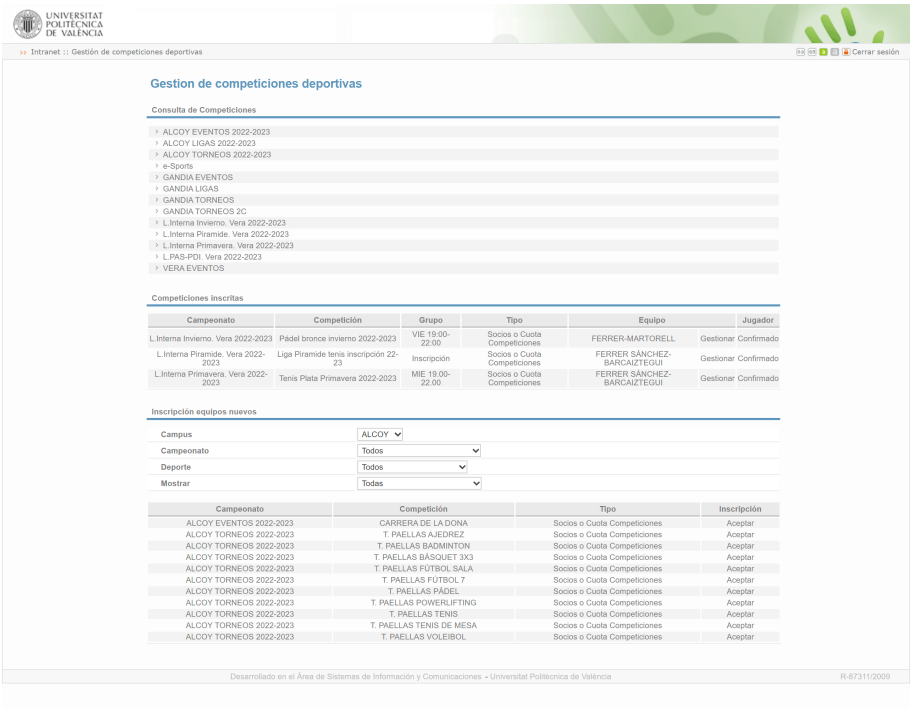

Figura 2.1: Captura de pantalla de la aplicación web de competiciones disponible para los socios

entreguen al administrador. Este procedimiento es lento, propenso a errores, y malo para el medio ambiente. Además, la información no se puede consultar en línea hasta que el administrador no la introduce en el sistema.

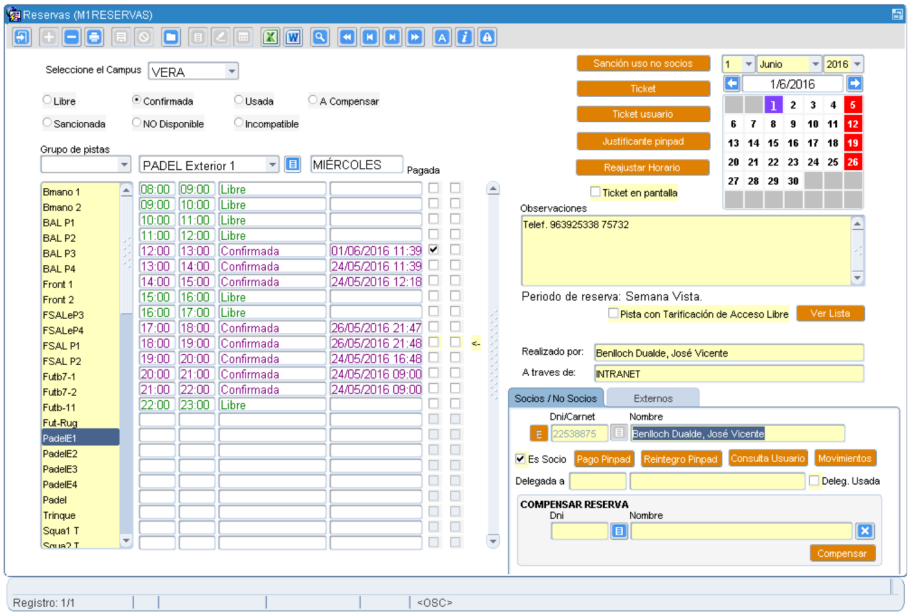

Figura 2.2: Captura de pantalla de la gestión de instalaciones de la aplicación Llucena

2.1.2. TeamSnap Tournaments

TeamSnap Tournaments es una aplicación móvil que permite a los usuarios gestionar ligas, torneos y competiciones. Esta aplicación permite a los usuarios

inscribirse a competiciones, ver los resultados de los partidos y consultar la clasificación. Además, los administradores pueden crear competiciones, gestionar los equipos y los partidos. Esta aplicación también dispone de una aplicación móvil para iOS y Android que permite a los usuarios consultar la información de las competiciones en sus dispositivos móviles. En la figura 2.3 se puede ver la aplicación móvil de TeamSnap Tournaments, donde se muestra una lista de los próximos partidos de una competición.

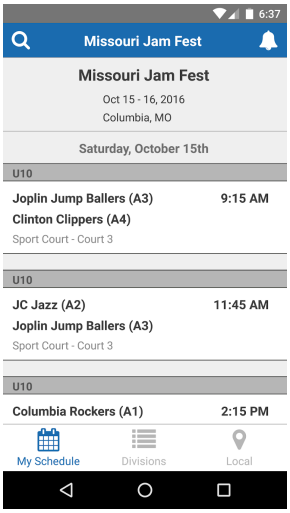

Figura 2.3: Captura de pantalla de la aplicación móvil de TeamSnap Tournaments.

Sin embargo, esta aplicación no permite gestionar las instalaciones deportivas, ni reservar pistas. Además, no permite gestionar las actividades deportivas organizadas por el club deportivo de la universidad.

2.1.3. Swift

Swift es un software de gestión de instalaciones que incluye varias características como la reserva de instalaciones en línea, la gestión de los horarios de las instalaciones y la gestión de los socios. Es una aplicación web con diseño moderno y además es responsive, por lo que funciona correctamente en dispositivos móviles. También incluye la posibilidad de añadir clases y actividades. La figura 2.4 muestra la pantalla principal de la aplicación, mostrando características como los horarios de las instalaciones y poder ver para que está siendo utilizado.

En cambio, esta aplicación está orientada a gimnasios y centros deportivos donde no se realizan competiciones, por lo que se encuentra limitada en este aspecto.

2.1.4. Spond

Spond es una aplicación y solución de gestión de clubes que proporciona una plataforma gratuita para simplificar la administración de clubes, equipos y grupos, independientemente del deporte. También incluye servicios de mensajería instantánea entre jugadores y entrenadores, y espacio para compartir archivos

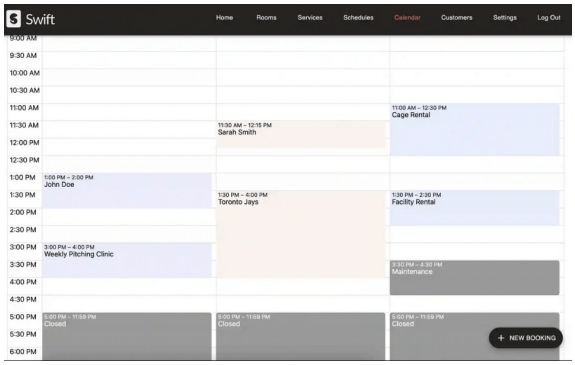

Figura 2.4: Captura de pantalla de la aplicación Swift.

con el equipo. La aplicación está más orientada a clubes deportivos que a universidades, pero incluye algunas características interesantes como la gestión de equipos y lista de partidos de las competiciones. En la figura 2.5 se puede una captura de la aplicación donde se muestra la lista de eventos de un equipo.

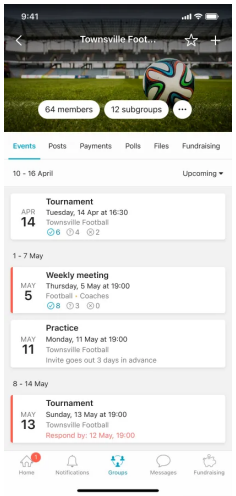

Figura 2.5: Captura de pantalla de la aplicación Spond.

2.1.5. Upper Hand

Upper Hand es una aplicación de programación deportiva para entrenamiento deportivo e instalaciones deportivas. Fue creada para facilitar el entrenamiento deportivo. Contiene herramientas como la programación de clientes, el procesamiento de pagos, la comunicación y más. También permite configurar el software para adaptarse a ofertas comerciales, ya sea que realice lecciones privadas y grupales y organización de campamentos o clínicas.

La aplicación tiene varios puntos positivos como el manejo de instalaciones y la posibilidad de crear actividades. Sin embargo, no permite gestionar competiciones deportivas.

2.2 Propuesta

Para elaborar la propuesta de este proyecto se va a resumir las principales características que presentan cada una de las aplicaciones analizadas anteriormente. Para clasificar las características y cómo de importantes son para el proyecto se ha utilizado el método **MoSCoW**. Este método consiste en clasificar las características en cuatro grupos:

- **Must have**: características que son imprescindibles para el proyecto.
- **Should have**: características que son importantes para el proyecto, pero no imprescindibles.
- **Could have**: características que son deseables, pero no necesarias.
- **Won't have**: características que no se incluirán en el proyecto.

En la tabla 2.1 se puede ver el resultado del análisis.

Existen algunas aplicaciones con características como la *Administración de hijos*, *Mensajería instantánea* y *Información de entrenadores* que no se van a incluir en el proyecto porque no son necesarias en el contexto de este proyecto. Por lo tanto, se han marcado como *Won't have*. En cuanto a la administración de hijos, esta característica es necesaria en el contexto de una aplicación para clubes deportivos, pero no en el contexto de una aplicación para universidades. En cuanto a la mensajería instantánea, esta característica puede ser interesante en el contexto de clubes deportivos, pero los usuarios y equipos de la universidad ya tienen otras vías de comunicación como correo electrónico y WhatsApp. Sobre la información de entrenadores, podría ser interesante para el manejo de clubes con equipos, pero en las competiciones internas de la universidad los equipos no tienen entrenador, ya que son los propios jugadores los que se organizan para jugar los partidos. Por lo tanto, no es necesario incluir esta característica en el proyecto.

Se propone desarrollar una aplicación con las características marcadas con *Must have* y *Should have*. Estas características son *Procedimientos completamente en línea*, *Gestión de instalaciones*, *Reservas de instalaciones*, *Gestión de actividades*, *Equipos y competiciones* y *Diseño moderno y atractivo*.

Tabla 2.1: Comparativa de las aplicaciones analizadas.

-	UPV	TeamSnap	Swift	Spond	Upper Hand	Hércules
Procedimientos completamente en línea	No	Sí	Sí	Sí	Sí	MUST
Guardianes / Administración de hijos	No	No	No	Sí	No	WON'T
Gestión de instalaciones	Sí	No	Sí	No	Sí	MUST
Mensajería instantánea	No	Sí	No	Sí	No	WON'T
Reservas de instalaciones	Sí	No	Sí	No	Sí	MUST
Gestión de actividades	Sí	No	Sí	No	Sí	MUST
Equipos y competiciones	Sí	Sí	No	Sí	No	MUST
Información de entrenadores	No	Sí	No	No	No	WON'T
Diseño moderno y atractivo	No	Sí	Sí	Sí	Sí	SHOULD
Almacenamiento de archivos	No	No	No	Sí	No	COULD

CAPÍTULO 3
Análisis del problema

3.1 Identificación y análisis de las soluciones

El software de gestión de actividades relacionadas con el ámbito deportivo se suele centrar en la gestión de las competiciones deportivas, las instalaciones deportivas o las actividades deportivas. Sin embargo, pocos de ellos se centran en gestionar todas las áreas. Uno de los principales problemas que puede producirse es la falta de unificación de los datos, ya que cada área se gestiona de forma independiente. Las instalaciones están disponibles para reservar por los usuarios, pero también deberán estar disponibles para las competiciones y las actividades. Si se realiza una actividad o un partido de una competición en una instalación, esta no estará disponible para reservar por los usuarios, de modo que es importante que exista una coordinación entre las tres áreas para que no se produzcan problemas de disponibilidad.

Al establecer como objetivo el desarrollo de una aplicación que unifique la gestión de las tres áreas, es necesario analizar algunos problemas que pueden surgir.

3.1.1. Tipos de usuario

En un sistema de gestión deportiva es necesario diferenciar los diferentes tipos de usuario que pueden llegar a usar el sistema y los diferentes usos que pueden llegar a dar. En el caso de las competiciones los usuarios serán:

- **Jugadores:** Los jugadores son los usuarios que participan en las competiciones.
- **Árbitros:** Los árbitros son los usuarios que se encargan de arbitrar los partidos.
- **Gestores:** Los gestores son los usuarios que se encargan de gestionar las competiciones.

En el caso de las instalaciones deportivas los usuarios serán:

- **Socios:** Los socios son los usuarios que utilizan las instalaciones deportivas.

- **Gestores:** Los gestores son los usuarios que se encargan de gestionar las instalaciones deportivas.

En el caso de las actividades deportivas los usuarios serán:

- **Socios participantes:** Los socios participantes son los usuarios que participan en las actividades deportivas.
- **Monitores:** Los monitores son los usuarios que se encargan de impartir las actividades deportivas.
- **Gestores:** Los gestores son los usuarios que se encargan de gestionar las actividades deportivas.

Por lo tanto, podemos distinguir dos tipos de usuarios que se encuentran presentes en las tres áreas: los socios y los gestores. También existen dos tipos de usuarios específicos y especializados en dos áreas: los monitores en el área de actividades y los árbitros en el área de competiciones.

3.1.2. Usos de instalaciones

En un sistema de gestión deportiva es necesario diferenciar los diferentes usos de instalaciones deportivas que pueden llegar a existir, ya que es el elemento más básico del sistema. En el caso de las competiciones deportivas, las instalaciones deportivas se utilizan para disputar los partidos. En el caso de las actividades deportivas, las instalaciones deportivas se utilizan para que los monitores puedan impartir las actividades deportivas. Por último, los socios pueden reservar las instalaciones deportivas para utilizarlas de forma individual o en grupo.

Analizando los usos de las instalaciones, es imprescindible que la aplicación permita a los administradores crear instalaciones deportivas y crear horarios para cada instalación donde se especifique la disponibilidad. Además, debe existir una coordinación entre las áreas de competiciones, actividades y reservas para que la instalación no se pueda dar la posibilidad de que se utilice para dos usos diferentes al mismo tiempo.

3.2 Modelado Conceptual

Para el modelado conceptual de la aplicación Hércules se va a utilizar la notación de diagramas de clases de UML. En la figura 3.1 se muestra el modelo conceptual de la aplicación Hércules.

A la hora de modelar la aplicación el primer desafío que se presenta es como relacionar las tres partes principales de la aplicación: las instalaciones, las actividades y las competiciones. Por ejemplo, una competición de squash solo se debe jugar en las pistas que permiten jugar a squash, y las actividades de atletismo solo se pueden realizar en las pistas de atletismo. Para solucionar esto se introduce la clase «*Sport*» (deporte). Las clases «*Facility*» (instalación), «*Competition*»

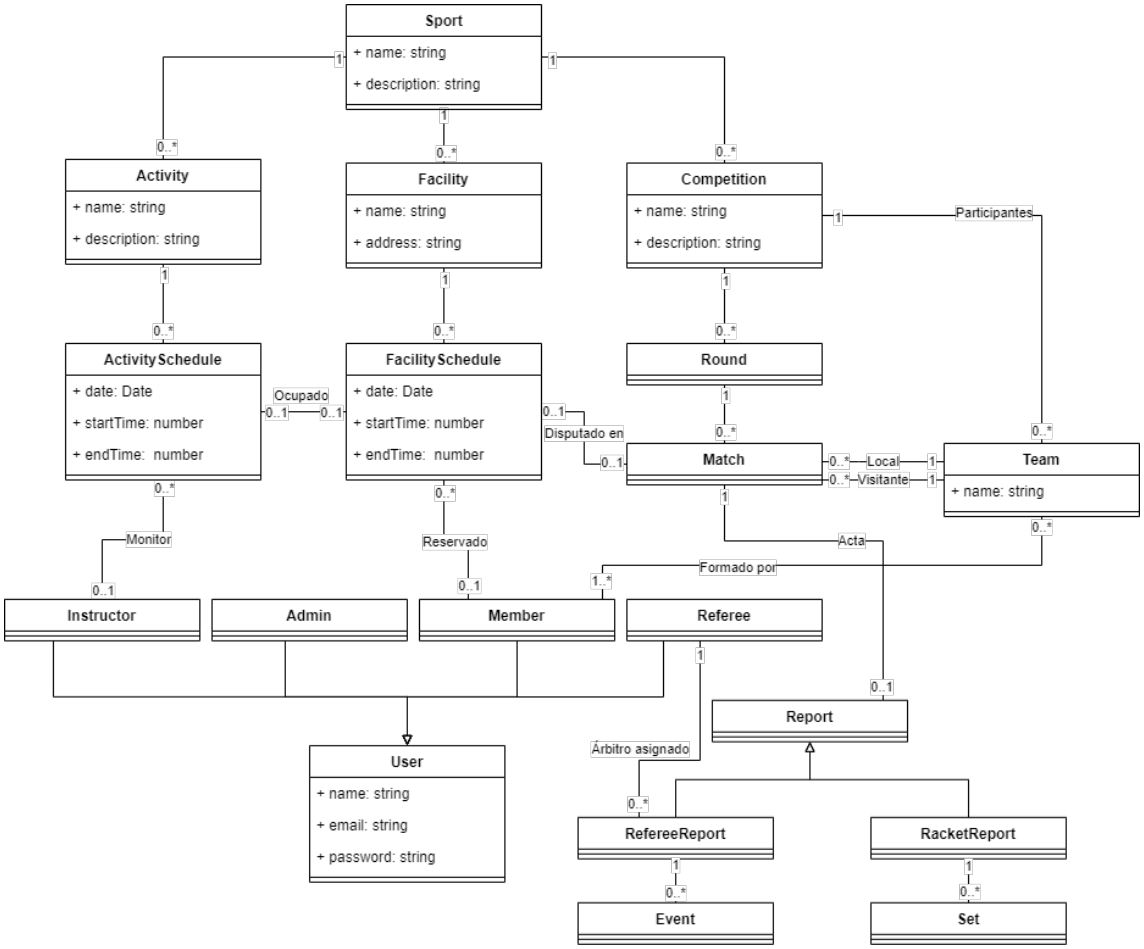

Figura 3.1: Modelo conceptual

(competición) y «*Activity*» (actividad) tienen una relación de uno a muchos con la clase «*Sport*», ya que una instalación, una competición o una actividad puede estar relacionada con un único deporte, pero un deporte puede estar relacionado con varias instalaciones, competiciones o actividades.

Las instalaciones tienen un horario, por lo que se introduce la clase «*FacilitySchedule*» (horario de instalación). Esta clase tiene las propiedades «*date*» (fecha), «*startTime*» (hora de inicio) y «*endTime*» (hora de fin). La clase «*FacilitySchedule*» tiene una relación de uno a muchos con la clase «*Facility*», ya que una instalación puede tener varios horarios, pero un horario solo puede estar relacionado con una instalación.

Igualmente, las actividades tienen un horario, por lo que se añade la clase «*ActivitySchedule*» (horario de actividad). Esta clase tiene las propiedades «*date*» (fecha), «*startTime*» (hora de inicio) y «*endTime*» (hora de fin). La clase «*ActivitySchedule*» tiene una relación de uno a muchos con la clase «*Activity*», ya que una actividad puede tener varios horarios, pero un horario solo puede estar relacionado con una actividad. Además, tiene una referencia a la clase «*FacilitySchedule*», ya que una actividad se imparte en una instalación y por lo tanto debe ocupar un horario de la instalación.

En cuanto a las competiciones, se introduce la clase «*Team*» (equipo). Cada equipo tiene un nombre y una lista de jugadores. La clase «*Team*» tiene una rela-

ción de uno a muchos con la clase «*Competition*», ya que una competición puede tener varios equipos, pero un equipo solo puede estar relacionado con una competición. Una vez la competición ha empezado aparece la clase «*Round*» (jornada), que está formada por «*Match*» (partido), que tiene un «*Team*» asignado como *team1* (local) y otro como *team2* (visitante).

Cada partido puede tener una acta de partido asociada, que es la que se rellena al finalizar el partido. La clase «*Match*» tiene una relación de uno a uno con la clase «*Report*» (acta de partido). Si el acta es de raqueta, la clase «*Report*» tiene una relación de uno a muchos con la clase «*Set*» (set), que almacena el resultado de cada set del partido. Si es un evento de deporte de equipo, la clase «*Report*» tiene una relación de uno a muchos con la clase «*Event*» (evento), que almacena los eventos del partido.

CAPÍTULO 4
Proceso de desarrollo del software

En este apartado se describirá el proceso de desarrollo software que se ha seguido para el desarrollo de la aplicación. Para el desarrollo de Hércules se ha escogido el proceso de desarrollo de software ágil Scrum como metodología de desarrollo. En primer lugar, se explicará el funcionamiento de Scrum. En segundo lugar, se detallará como se ha adaptado Scrum para el desarrollo de la aplicación Hércules. Por último, se analizará el resultado de los varios *sprints* que se han realizado durante el desarrollo de la aplicación. En la figura 4.1 se muestra el proceso de Scrum.

4.1 Scrum

Scrum es un marco de trabajo para el desarrollo ágil de software basado en la flexibilidad, colaboración y la entrega iterativa de productos. Scrum se basa en ciclos de trabajos de duración de 1 a 4 semanas llamados *sprints*. Al final de cada *sprint* se entrega un incremento del producto. Aunque se diseñó originalmente para equipos de desarrollo de alrededor de 7 personas, también se puede adaptar para equipos de desarrollo de una sola persona. En Scrum existen varias *ceremonias* que se realizan durante el desarrollo del proyecto. Estas son:

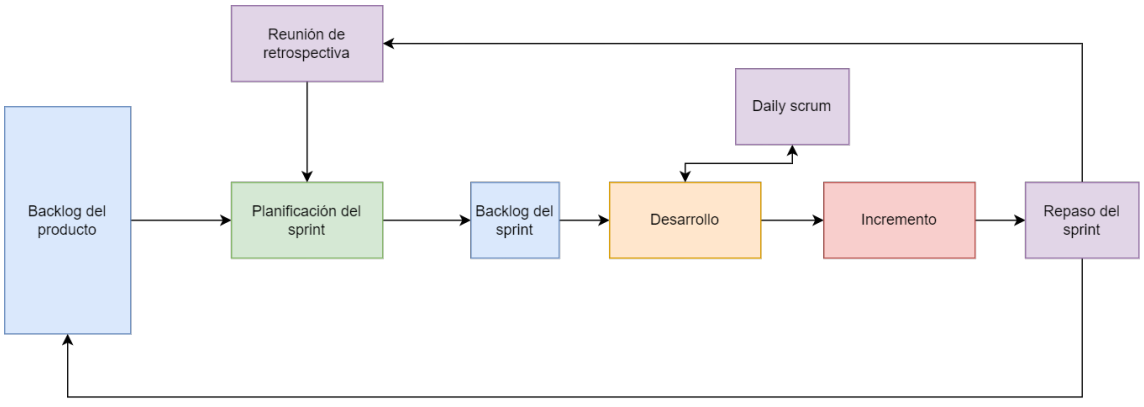

Figura 4.1: Proceso de Scrum

- **Planificación del sprint**: al inicio de cada *sprint* se realiza una reunión de planificación en la que se decide que tareas se van a realizar durante el *sprint*. En esta reunión se decide que tareas se van a realizar.

- **Reunión diaria**: (también conocida como *Daily scrum*) cada día se realiza una reunión de 15 minutos en la que se explica que se ha hecho hasta la fecha, que se va a hacer en los próximos días y si hay algún problema que impida realizar el trabajo.

- **Repaso del sprint**: al final de cada *sprint* se realiza una reunión en la que se muestra el incremento del producto y basado en los resultados del *sprint* se modifican las estimaciones de las tareas del backlog.

- **Reunión de retrospectiva**: al final de cada *sprint* se realiza una reunión en la que se analiza el proceso de desarrollo del *sprint* y se proponen mejoras para el siguiente *sprint*.

A continuación se describirán las ventajas y desventajas de Scrum y se explicará como se ha adaptado para el desarrollo de la aplicación Hércules.

4.1.1. Ventajas

1. **Flexibilidad**: Scrum permite adaptarse a los cambios de requisitos durante el desarrollo del proyecto de manera rápida y eficiente, ya que los requisitos y prioridades se pueden ajustan cada *sprint*.

2. **Visibilidad y transparencia**: Scrum permite que el desarrollador pueda ver el progreso del proyecto en todo momento, lo que ayuda a identificar posibles problemas y retrasos.

3. **Mejora incremental**: Scrum promueve la mejora continua del proceso de desarrollo. Al final de cada *sprint* se realiza una retrospectiva para analizar los puntos fuertes y débiles del proceso de desarrollo y se proponen mejoras para el siguiente *sprint*. Esto permite que el desarrollador pueda analizar y mejorar su proceso para mejorar su productividad y eficiencia al desarrollar.

4. **Enfoque en el valor del producto**: Scrum se centra en entregar el valor del negocio. Esto permite que el desarrollador pueda centrarse en las funcionalidades que aportan más valor al producto final.

5. **Mayor motivación y compromiso personal**: Al aplicar Scrum, el desarrollador tiene propósitos claros, metas alcanzables y un compromiso con la entrega de resultados en cada *sprint*. Esto permite que el desarrollador se sienta más motivado y comprometido con el proyecto, además de aumentar la motivación y satisfacción personal. Esto es muy importante en proyectos donde el equipo de desarrollo es de una sola persona, ya que es muy fácil centrarse en pequeños detalles y perder la motivación al ver que se alarga el desarrollo del proyecto.

4.1.2. Desventajas

1. **Exceso de ceremonias**: Scrum requiere de muchas reuniones y ceremonias, lo que puede consumir mucho tiempo y parecer excesivas en proyectos pequeños donde el equipo de desarrollo es de una sola persona.

2. **Sobrecarga de responsabilidad**: Al trabajar solo en un proyecto Scrum, el desarrollador tiene que asumir todas las responsabilidades incluyendo la planificación, desarrollo, pruebas y documentación. Esto puede provocar que el desarrollador se sienta sobrecargado de trabajo y pierda la motivación.

3. **Falta de enfoque en la mejora de equipo**: Scrum promueve el aprendizaje y la mejora continua del equipo. En un equipo de una sola persona, la mejora de equipo se limita a la mejora personal. Además, la autoevaluación puede ser más difícil de identificar y aplicar debido a la falta de perspectivas de otros miembros del equipo.

4. **Limitación de la diversidad de habilidades**: Al desarrollar en un equipo de una sola persona, el desarrollador puede tener limitaciones en cuanto a su experiencia y conocimiento de algunas áreas, lo que limita la calidad y eficiencia del desarrollo.

5. **Riesgo de agotamiento**: Scrum requiere de un alto nivel de compromiso y responsabilidades. Al asumir todas las tareas existe un riesgo de agotamiento físico y mental debido a la carga de trabajo.

4.1.3. Adaptación de Scrum

Teniendo en cuenta las ventajas y desventajas de Scrum para proyectos de un desarrollador, se ha decidido adaptar Scrum para reducir las desventajas y aprovechar las ventajas. Para ello, se ha decidido reducir el número de ceremonias y reuniones, ya que en un proyecto de una sola persona no es necesario realizar tantas reuniones y ceremonias. También, se ha convertido la *daily* (reunión diaria) una visión del estado de las tareas en JIRA[1]. Además, se ha decidido reducir el tiempo de los *sprints* a una semana, ya que al trabajar solo en el proyecto se puede avanzar más rápido y es más fácil estimar el tiempo que se va a tardar en desarrollar una tarea.

4.1.4. Fases del proceso de desarrollo

En esta sección se describen las diferentes fases por las que pasa una tarea durante el proceso de desarrollo. La figura 4.2 muestra el esquema de las diferentes fases.

- **Fase *TO DO***: En esta fase se encuentra la tarea cuando se añade por primera vez al tablero de tareas. La tarea no debe estar mucho tiempo en esta fase,

[1]JIRA es un producto de software propietario desarrollado por la empresa australiana Atlassian para la gestión de proyectos, seguimiento de errores e incidencias.

Figura 4.2: Fases de la tarea

ya que puede provocar que se olvide estimarla y por lo tanto no se tenga en cuenta en la planificación del *sprint*.

- **Fase *ESTIMATING***: En esta fase se encuentra la tarea cuando se está estimando. En esta fase se debe estimar el tiempo que se va a tardar en desarrollar la tarea. Esta fase tiene una *Definition of done* que consiste en que la tarea debe estar estimada y debe tener unos bocetos de interfaz de usuario proporcionados. Si la tarea no cumple con la *Definition of done* no se puede pasar a la siguiente fase.
- **Fase *ESTIMATED***: En esta fase se encuentran las tareas que ya han sido estimadas y es la fase en la que se encuentran todas las tareas del *backlog*[2]. Es importante que todas las tareas que se encuentren en esta fase y en el *backlog* se encuentren estimadas y que al final de cada *sprint* se vuelvan a estimar para tener en cuenta los posibles cambios de requisitos.
- **Fase *IN PROGRESS***: En esta fase se encuentra la tarea cuando se está desarrollando. Esta fase tiene una *Definition of done* que consiste en que la tarea debe haberse desarrollado completamente, debe haberse probado y debe considerarse si se debe desarrollar pruebas unitarias. En caso de que se considere que se deben desarrollar pruebas unitarias, la tarea debe tener las pruebas unitarias desarrolladas y deben pasar correctamente. Si la tarea no cumple con la *Definition of done* no se puede pasar a la siguiente fase.
- **Fase *DONE***: En esta fase se encuentra la tarea cuando se ha desarrollado completamente.

4.2 Aplicación de Scrum al proyecto

Antes de comenzar los sprints, se desarrolló una planificación inicial del proyecto. En esta planificación se definió el alcance, se creó el *backlog* y se estimaron todas las tareas. También se priorizaron las tareas por orden de prioridad. Cada sprint tuvo una duración de 8 días. Al comienzo de cada sprint se seleccionaron las tareas para abordar en el sprint. Al final de cada sprint se realizó una retrospectiva para analizar el sprint y mejorar el proceso de desarrollo.

4.2.1. Preparación

Antes del inicio del sprint se realizaron tareas como la investigación inicial sobre el tema y estudio de las tecnologías que se iban a utilizar para desarrollar el

[2]El **backlog** en Scrum es una lista de tareas pendientes que se organizan por orden de prioridad para gestionar el trabajo del equipo.

proyecto. Además, se inicializó el repositorio *Git* y se creó el modelo de dominio y el modelo de base de datos.

4.2.2. Primer sprint

Durante el primer sprint, se puso en marcha la implementación del proyecto. Se desarrollaron las características relacionadas con el inicio de sesión y registro, la gestión de deportes e instalaciones y la reserva de instalaciones. En la figura 4.3 se puede ver el *burndown chart*[3] del primer sprint. Analizando el burndown chart se puede ver como durante los primeros días no se realizó completó mucho trabajo, ya que era un proyecto nuevo con tecnologías que no se había usado anteriormente y se necesitaba tiempo para aprender a utilizarlas. A partir del día 4 se comenzó a avanzar más rápido y se completaron todas las tareas planificadas para el sprint. En la retrospectiva de este sprint se llegó a la conclusión de que se debía haber dedicado más tiempo a la investigación de las tecnologías y a la planificación del proyecto, pero que este problema no se volvería a repetir en los siguientes sprints. Después del sprint se reestimaron todas las tareas del *backlog* para tener en cuenta los posibles cambios de requisitos y la experiencia adquirida durante el sprint.

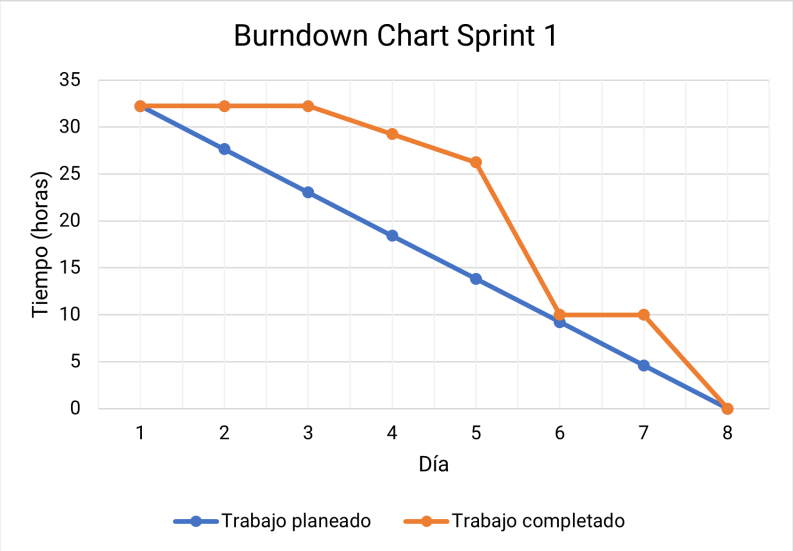

Figura 4.3: Burndown chart del primer sprint

4.2.3. Segundo sprint

Durante el segundo sprint, se continuó con la implementación del proyecto. Se desarrollaron las características relacionadas con las actividades y la asignación de instalaciones y monitores a las actividades. Analizando la figura 4.4 se puede ver como la línea de trabajo completado estaba mucho más cerca de la de trabajo

[3]El **burndown chart** (gráfico de avance o consumo) es una herramienta utilizada en la metodología Scrum para visualizar y realizar un seguimiento del progreso del trabajo durante un sprint. Representa la cantidad de trabajo pendiente en el eje vertical y el tiempo transcurrido en el eje horizontal.

completado, ya que se había aprendido a utilizar las tecnologías y se había avanzado en la implementación del proyecto. A partir del día 4 se comenzó a avanzar aún más rápido. En la retrospectiva de este sprint se ha llegado a la conclusión de que hay que mejorar la cantidad de trabajo terminado durante los primeros días del sprint, pero que se había mejorado mucho con respecto al primer sprint.

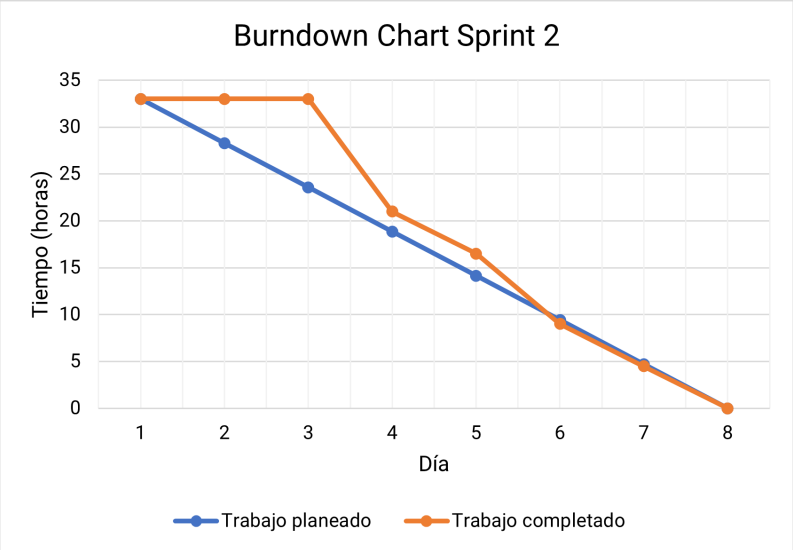

Figura 4.4: Burndown chart del segundo sprint

4.2.4. Tercer sprint

Durante el tercer y último sprint, se terminó con la implementación del proyecto. Se desarrollaron las funcionalidades de las competiciones, equipos y actas. Analizando la figura 4.5 se puede ver como la línea de trabajo completado estaba mucho más cerca del de trabajo planeado desde el inicio. Durante unos días estuvo por encima, pero al final se logró terminar correctamente. En la retrospectiva de este sprint se ha llegado a la conclusión de que hay que el proyecto y la planificación ha sido un éxito, y en el caso de hacer más sprints en el futuro se intentará que el trabajo completado esté por debajo del trabajo planeado desde el inicio para tener un margen de error.

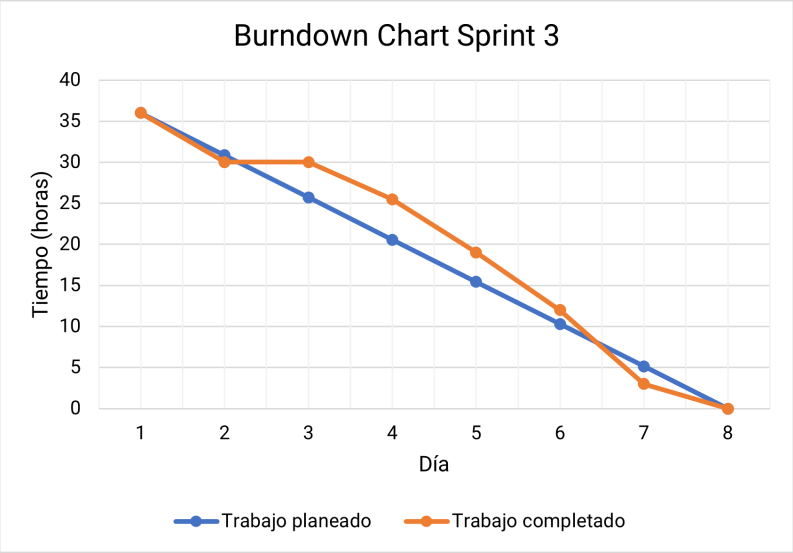

Figura 4.5: Burndown chart del tercer sprint

CAPÍTULO 5
Especificación de requisitos

En esta sección se describen los requisitos funcionales y no funcionales de la aplicación. Los requisitos funcionales describen las funcionalidades que debe tener la aplicación. Los requisitos no funcionales describen las características que debe tener la aplicación.

5.1 Introducción

Para la especificación de requisitos se utilizan las técnicas de entrevistas, observación y casos de uso para identificar los *stakeholders*, los requisitos funcionales y los requisitos no funcionales de la aplicación.

5.2 Stakeholders

En esta sección se van a identificar los *stakeholders* del proyecto. Los *stakeholders* son las personas o grupos que tienen un interés en el proyecto. Para cada *stakeholder* se identificará si es usuario directo de la aplicación, y los intereses que tiene. Los *stakeholders* del proyecto son los siguientes:

- **Socios que hacen uso de las reservas de pistas**
  - Usuario directo
  - Mayor rapidez y facilidad a la hora de buscar pistas disponibles y poder reservarlas de forma sencilla y rápida
- **Socios que participan en competiciones**
  - Usuario directo
  - Inscribirme en las competiciones y poder ver las fechas de los partidos, los resultados y la calificación fácilmente
- **Socios que participan en actividades deportivas**
  - Usuario directo

  - Informarme de las actividades deportivas que se realizan en la universidad y poder inscribirme rápidamente

- **Empleados de administración de pistas**
  - Usuario directo
  - Gestionar las instalaciones disponibles y elegir cuáles están disponibles para la reserva
- **Empleados de administración de competiciones**
  - Usuario directo
  - Crear equipos, inscribir a los socios a los equipos y permitir que los árbitros y socios puedan ver y anotar los resultados de los partidos
- **Empleados de administración de actividades deportivas**
  - Usuario directo
  - Gestionar las actividades, los horarios, las instalaciones asignadas y el monitor encargado
- **Monitores de las actividades deportivas**
  - Usuario indirecto
  - Gestión de actividades más rápida para que se me informe de los cambios con tiempo. Mayor información de las actividades a los socios para aumentar la asistencia

Una vez identificados los *stakeholders*, se va a realizar un diagrama de contexto para identificar los límites del sistema y los usuarios que interactúan con el sistema. En la figura 5.1 se puede ver el diagrama de contexto del sistema.

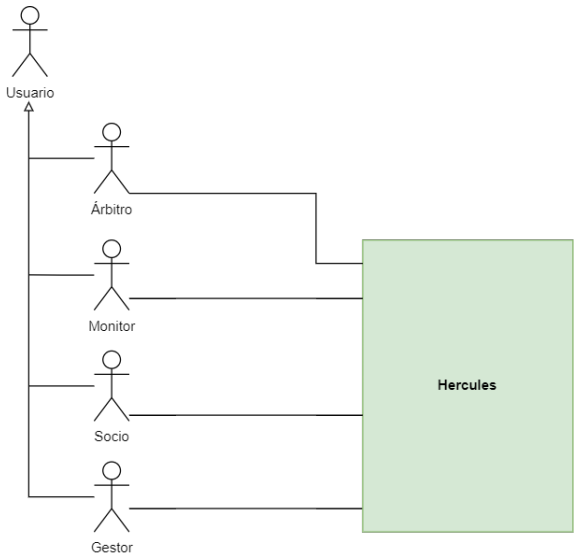

Figura 5.1: Diagrama de contexto

Podemos dividir los *stakeholders* en dos grupos:

- **Socios**: Los socios son los usuarios que hacen uso de las instalaciones deportivas de la universidad. Los socios pueden ser estudiantes, profesores o personal de administración y servicios de la universidad.
- **Empleados**: Los empleados son los usuarios que trabajan en la universidad. Los empleados pueden ser monitores de las actividades deportivas, personal de administración de las competiciones, personal de administración de las pistas o personal de administración de las actividades deportivas.

A continuación, se va a especificar las funcionalidades que debe tener la aplicación. Primero se van a identificar las características que debe tener la aplicación. Después se van a identificar requisitos funcionales para cada característica. Por último, se van a identificar los casos de uso para cada requisito funcional.

5.3 Características

- **C1: Registro, login y logout**: Esta característica engloba las funcionalidades relacionadas con el autenticado, registro, login y logout de los usuarios.
- **C2: Gestión de instalaciones**: Esta característica engloba las funcionalidades relacionadas con la gestión e inscripciones de las instalaciones deportivas de la universidad.
- **C3: Gestión de actividades**: Esta característica engloba las funcionalidades relacionadas con la gestión e inscripciones de las actividades deportivas de la universidad.
- **C4: Gestión de competiciones**: Esta característica engloba las funcionalidades relacionadas con la gestión de las competiciones deportivas de la universidad.
- **C5: Equipos y actas**: Esta característica engloba las funcionalidades relacionadas con la gestión de los equipos y las actas de las competiciones deportivas de la universidad.

5.4 Requisitos funcionales

Tabla 5.1: RF1: Registro

Identificador	RF1
Tipo	Funcional
Característica	C1: Registro, login y logout
Nombre	Registro
Descripción	Los usuarios podrán registrarse en la aplicación mediante un correo electrónico.
Prioridad	Alta

Tabla 5.2: RF2: Login

Identificador	RF2
Tipo	Funcional
Característica	C1: Registro, login y logout
Nombre	Login
Descripción	Los usuarios podrán iniciar sesión en la aplicación mediante un correo electrónico y una contraseña.
Prioridad	Alta

Tabla 5.3: RF3: Logout

Identificador	RF3
Tipo	Funcional
Característica	C1: Registro, login y logout
Nombre	Logout
Descripción	Los usuarios podrán cerrar sesión en la aplicación.
Prioridad	Alta

Tabla 5.4: RF4: CRUD Instalaciones

Identificador	RF4
Tipo	Funcional
Característica	C2: Gestión de instalaciones
Nombre	CRUD Instalaciones
Descripción	Los gestores podrán crear, leer, actualizar y eliminar instalaciones.
Prioridad	Alta

Tabla 5.5: RF5: CRUD Horarios de instalaciones

Identificador	RF5
Tipo	Funcional
Característica	C2: Gestión de instalaciones
Nombre	CRUD Horarios de instalaciones
Descripción	Los gestores podrán crear, leer, actualizar y eliminar horarios de las instalaciones.
Prioridad	Alta

Tabla 5.6: RF6: Reserva de instalaciones

Identificador	RF6
Tipo	Funcional
Característica	C2: Gestión de instalaciones
Nombre	Reserva de instalaciones
Descripción	Los socios podrán reservar instalaciones.
Prioridad	Alta

Tabla 5.7: RF7: CRUD Actividades

Identificador	RF7
Tipo	Funcional
Característica	C3: Gestión de actividades
Nombre	CRUD Actividades
Descripción	Los gestores podrán crear, leer, actualizar y eliminar actividades.
Prioridad	Alta

Tabla 5.8: RF8: CRUD Horarios de actividades

Identificador	RF8
Tipo	Funcional
Característica	C3: Gestión de actividades
Nombre	CRUD Horarios de actividades
Descripción	Los gestores podrán crear, leer, actualizar y eliminar horarios de las actividades.
Prioridad	Alta

Tabla 5.9: RF9: Asignación de monitores a actividades

Identificador	RF9
Tipo	Funcional
Característica	C3: Gestión de actividades
Nombre	Asignación de monitores a actividades
Descripción	Los gestores podrán asignar monitores a los horarios de las actividades.
Prioridad	Media

Tabla 5.10: RF10: Asignación de instalaciones a actividades

Identificador	RF10
Tipo	Funcional
Característica	C3: Gestión de actividades
Nombre	Asignación de instalaciones a actividades
Descripción	Los gestores podrán asignar instalaciones a actividades.
Prioridad	Alta

Tabla 5.11: RF11: Consultar actividades asignadas

Identificador	RF11
Tipo	Funcional
Característica	C3: Gestión de actividades
Nombre	Consultar actividades asignadas
Descripción	Los monitores podrán ver que actividades tienen asignadas
Prioridad	Media

Tabla 5.12: RF12: Inscripción de socios a actividades

Identificador	RF12
Tipo	Funcional
Característica	C3: Gestión de actividades
Nombre	Inscripción de socios a actividades
Descripción	Los socios podrán inscribirse a actividades.
Prioridad	Alta

Tabla 5.13: RF13: CRUD Competiciones

Identificador	RF13
Tipo	Funcional
Característica	C4: Gestión de competiciones
Nombre	CRUD Competiciones
Descripción	Los gestores podrán crear, leer, actualizar y eliminar competiciones.
Prioridad	Alta

Tabla 5.14: RF14: Inscripción en competiciones

Identificador	RF14
Tipo	Funcional
Característica	C4: Gestión de competiciones
Nombre	Inscripción en competiciones
Descripción	Los socios podrán inscribirse en competiciones.
Prioridad	Alta

Tabla 5.15: RF15: CRUD Equipos

Identificador	RF15
Tipo	Funcional
Característica	C5: Equipos y actas
Nombre	CRUD Equipos
Descripción	Los gestores podrán crear, leer, actualizar y eliminar equipos.
Prioridad	Alta

Tabla 5.16: RF16: CRUD Actas

Identificador	RF16
Tipo	Funcional
Característica	CC5: Equipos y actas
Nombre	CRUD Actas
Descripción	Los gestores podrán crear, leer, actualizar y eliminar actas.
Prioridad	Alta

Tabla 5.17: RF17: Generación de cruces

Identificador	RF17
Tipo	Funcional
Característica	C5: Equipos y actas
Nombre	Generación de cruces
Descripción	Los gestores podrán generar cruces de competiciones.
Prioridad	Alta

Tabla 5.18: RF18: Actas de raqueta

Identificador	RF18
Tipo	Funcional
Característica	C5: Equipos y actas
Nombre	Actas de raqueta
Descripción	Los gestores podrán crear, leer, actualizar y eliminar actas de raqueta.
Prioridad	Media

Tabla 5.19: RF19: Actas de árbitro

Identificador	RF19
Tipo	Funcional
Característica	C5: Equipos y actas
Nombre	Actas de árbitro
Descripción	Los gestores podrán crear, leer, actualizar y eliminar actas de árbitro.
Prioridad	Media

5.5 Requisitos no funcionales

En esta sección se va a especificar los requisitos no funcionales que debe cumplir la aplicación.

Tabla 5.20: RNF1: Compatibilidad

Identificador	RNF1
Tipo	No funcional
Nombre	Compatibilidad
Descripción	La aplicación deberá ser compatible con los diferentes navegadores web de escritorio y móvil.
Prioridad	Alta

Tabla 5.21: RNF2: Seguridad

Identificador	RNF2
Tipo	No funcional
Nombre	Seguridad
Descripción	La aplicación deberá ser segura y no permitir el acceso a los datos de los usuarios sin autenticarse.
Prioridad	Media

Tabla 5.22: RNF3: Usabilidad

Identificador	RNF3
Tipo	No funcional
Nombre	Usabilidad
Descripción	La aplicación deberá ser fácil de usar y de entender para los usuarios.
Prioridad	Alta

Tabla 5.23: RNF4: Interfaz atractiva

Identificador	RNF4
Tipo	No funcional
Nombre	Interfaz atractiva
Descripción	La aplicación deberá tener una interfaz atractiva y fácil de usar.
Prioridad	Alta

Tabla 5.24: RNF5: Rendimiento

Identificador	RNF5
Tipo	No funcional
Nombre	Rendimiento
Descripción	La aplicación deberá ser rápida de navegar e interactuar, especialmente para los gestores.
Prioridad	Media

5.6 Descripciones de casos de uso

5.6.1. Registro, login y logout

En la figura 5.2 se muestra el diagrama de casos de uso de la característica C1: Registro, login y logout.

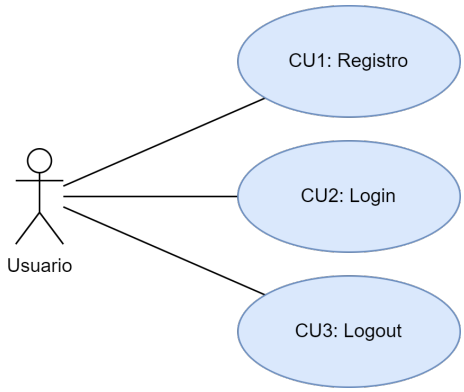

Figura 5.2: Diagrama de casos de uso: Registro, login y logout

Tabla 5.25: CU1: Registro

CU1	**Registro**
Descripción	El usuario se registra en la aplicación mediante un correo electrónico y una contraseña.
Actores	Usuario.
Precondiciones	El usuario no ha iniciado sesión en la aplicación.
Escenario principal	1. El usuario indica que desea registrarse. 2. El sistema muestra un formulario para el registro. 3. El usuario introduce su correo electrónico, nombre, contraseña, DNI, y rol. 4. El usuario confirma el registro. 5. El sistema registra al usuario en la aplicación. 6. El sistema confirma el registro al usuario.
Alternativas	4a. El usuario ha introducido un correo electrónico que ya está registrado en la aplicación. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3. — 4b. El usuario ha introducido un DNI que ya está registrado en la aplicación. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3.
Postcondiciones	El usuario se ha registrado en la aplicación.

Tabla 5.26: CU2: Inicio de sesión

CU2	**Login**
Descripción	El usuario inicia sesión en la aplicación mediante un correo electrónico y una contraseña.
Actores	Usuario.
Precondiciones	El usuario no ha iniciado sesión en la aplicación.
Escenario principal	1. El usuario indica que desea iniciar sesión. 2. El sistema muestra un formulario para el inicio de sesión. 3. El usuario introduce su correo electrónico y contraseña. 4. El usuario confirma el inicio de sesión. 5. El sistema inicia la sesión del usuario en la aplicación.
Alternativas	4a. El usuario ha introducido un correo electrónico que no está registrado en la aplicación. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3. — 4b. El usuario ha introducido una contraseña incorrecta. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3.
Postcondiciones	El usuario ha iniciado sesión en la aplicación.

Tabla 5.27: CU3: Cierre de sesión

CU3	**Logout**
Descripción	El usuario cierra sesión en la aplicación.
Actores	Usuario.
Precondiciones	El usuario ha iniciado sesión en la aplicación.
Escenario principal	1. El usuario indica que desea cerrar sesión. 2. El sistema cierra la sesión del usuario en la aplicación.
Alternativas	—
Postcondiciones	El usuario ha cerrado sesión en la aplicación.

5.6.2. Gestión de instalaciones

En la figura 5.3 se muestra el diagrama de casos de uso de la característica C2: Gestión de instalaciones.

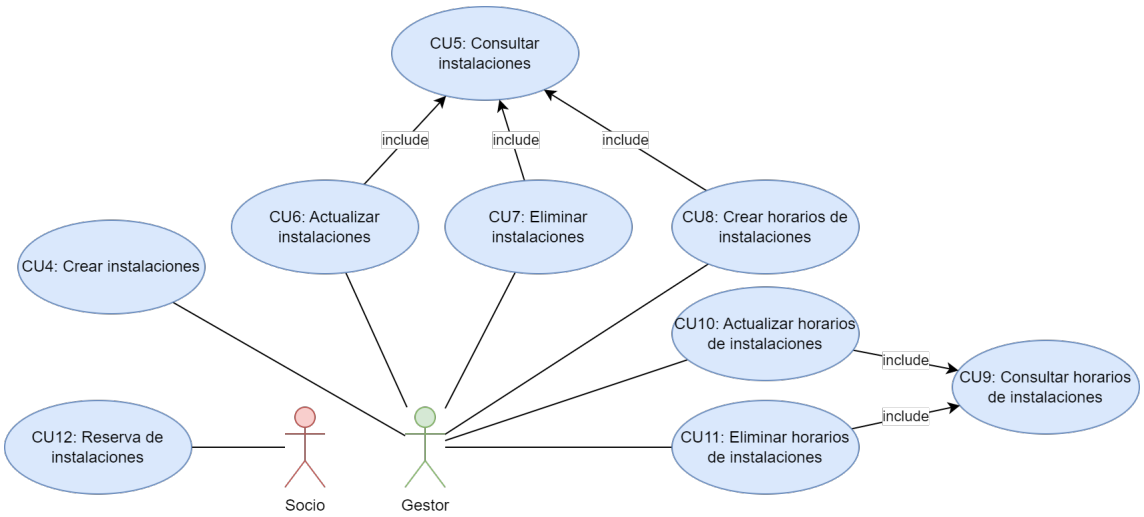

Figura 5.3: Diagrama de casos de uso: Gestión de instalaciones

Tabla 5.28: CU4: Crear instalaciones

CU4	**Crear instalaciones**
Descripción	El gestor crea una instalación.
Actores	Gestor.
Precondiciones	El gestor ha iniciado sesión en la aplicación.
Escenario principal	1. El gestor indica que desea crear una instalación. 2. El sistema muestra un formulario para la creación de la instalación. 3. El gestor introduce el nombre, la dirección y el deporte de la instalación. 4. El gestor confirma la creación de la instalación. 5. El sistema crea la instalación.
Alternativas	—
Postcondiciones	Se ha creado una nueva instalación.

Tabla 5.29: CU5: Consultar instalaciones

CU5	**Consultar instalaciones**
Descripción	El gestor consulta la información de las instalaciones.
Actores	Gestor.
Precondiciones	El gestor ha iniciado sesión en la aplicación.
Escenario principal	1. El gestor indica que desea consultar las instalaciones. 2. El sistema muestra la información de las instalaciones. 3. El gestor selecciona una instalación. 4. El sistema muestra la información de la instalación seleccionada.
Alternativas	2b. No existe ninguna instalación. 1. El sistema muestra un mensaje indicando que no hay instalaciones. 2. Se vuelve al paso 1.
Postcondiciones	—

Tabla 5.30: CU6: Actualizar instalaciones

CU6	**Actualizar instalaciones**
Descripción	El gestor actualiza la información de una instalación.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una instalación.
Escenario principal	1. El gestor indica que desea actualizar una instalación. 2. El sistema muestra un formulario con la información de la instalación. 3. El gestor modifica los datos de la instalación. 4. El gestor confirma la actualización de la instalación. 5. El sistema actualiza la instalación con los nuevos datos.
Alternativas	—
Postcondiciones	La información de la instalación se ha actualizado.
Incluye	CU5: Consultar instalaciones.

Tabla 5.31: CU7: Eliminar instalaciones

CU7	**Eliminar instalaciones**
Descripción	El gestor elimina una instalación.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una instalación.
Escenario principal	1. El gestor indica que desea eliminar una instalación. 2. El sistema muestra un mensaje de confirmación. 3. El gestor confirma la eliminación de la instalación. 4. El sistema elimina la instalación.
Alternativas	—
Postcondiciones	Se ha eliminado una instalación.
Incluye	CU5: Consultar instalaciones.

Tabla 5.32: CU8: Crear horarios de instalaciones

CU8	**Crear horarios de instalaciones**
Descripción	El gestor crea un horario para una instalación.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una instalación.
Escenario principal	1. El gestor indica que desea crear un horario para una instalación. 2. El sistema muestra un formulario con los datos del horario. 3. El gestor introduce la hora de inicio, la hora de fin y si la instalación está disponible para reservar o no. 4. El gestor confirma la creación del horario. 5. El sistema crea el horario.
Alternativas	4a. El gestor ha introducido una hora de inicio o fin que coincide con otro horario existente. 1. El sistema muestra un mensaje de error indicando que la hora de inicio o fin coincide con otro horario existente. 2. Se vuelve al paso 3.
Postcondiciones	Se ha creado un horario para una instalación.
Incluye	CU5: Consultar instalaciones.

Tabla 5.33: CU9: Consultar horarios de instalaciones

CU9	**Consultar horarios de instalaciones**
Descripción	El gestor consulta los horarios de una instalación.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una instalación.
Escenario principal	1. El gestor indica que desea consultar los horarios de una instalación. 2. El sistema muestra los horarios de la instalación.
Alternativas	—
Postcondiciones	—
Incluye	CU5: Consultar instalaciones.

Tabla 5.34: CU10: Actualizar horarios de instalaciones

CU10	**Actualizar horarios de instalaciones**
Descripción	El gestor actualiza el horario de una instalación.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una instalación con un horario.
Escenario principal	1. El gestor indica que desea actualizar un horario de una instalación. 2. El sistema muestra un formulario con los datos del horario. 3. El gestor modifica los datos del horario. 4. El gestor confirma la actualización del horario. 5. El sistema actualiza el horario.
Alternativas	4a. El gestor ha introducido una hora de inicio o fin que coincide con otro horario existente. 1. El sistema muestra un mensaje de error indicando que la hora de inicio o fin coincide con otro horario existente. 2. Se vuelve al paso 3.
Postcondiciones	El horario de la instalación ha sido actualizado
Incluye	CU9: Consultar horarios de instalaciones.

Tabla 5.35: CU11: Eliminar horarios de instalaciones

CU11	**Eliminar horarios de instalaciones**
Descripción	El gestor elimina el horario de una instalación.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una instalación con un horario.
Escenario principal	1. El gestor indica que desea eliminar un horario de una instalación. 2. El sistema muestra un mensaje de confirmación. 3. El gestor confirma la eliminación del horario. 4. El sistema elimina el horario.
Alternativas	—
Postcondiciones	El horario de la instalación ha sido eliminado
Incluye	CU9: Consultar horarios de instalaciones.

Tabla 5.36: CU12: Reserva de instalaciones

CU12	**Reserva de instalaciones**
Descripción	El socio reserva una instalación.
Actores	Socio.
Precondiciones	1. El socio ha iniciado sesión en la aplicación. 2. Existe al menos una instalación con un horario que permite la reserva de usuarios y que esté libre.
Escenario principal	1. El socio indica que desea reservar una instalación. 2. El sistema presenta todos los deportes que tienen instalaciones disponibles para reservar. 3. El socio selecciona un deporte. 4. El sistema muestra las instalaciones disponibles para el deporte seleccionado. 5. El socio selecciona una instalación. 6. El sistema presenta los horarios disponibles para la instalación seleccionada, separados por días. 7. El socio selecciona un horario. 8. El socio confirma la reserva. 9. El sistema reserva la instalación.
Alternativas	7a. El socio reserva una instalación que ya está reservada. 1. El sistema muestra un mensaje de error indicando que la instalación ya está reservada. 2. Se vuelve al paso 6.
Postcondiciones	Se ha reservado la instalación.

5.6.3. Gestión de actividades

En esta sección se describen los casos de uso relacionados con la gestión de actividades. En la figura 5.4 se puede observar el diagrama de casos de uso correspondiente.

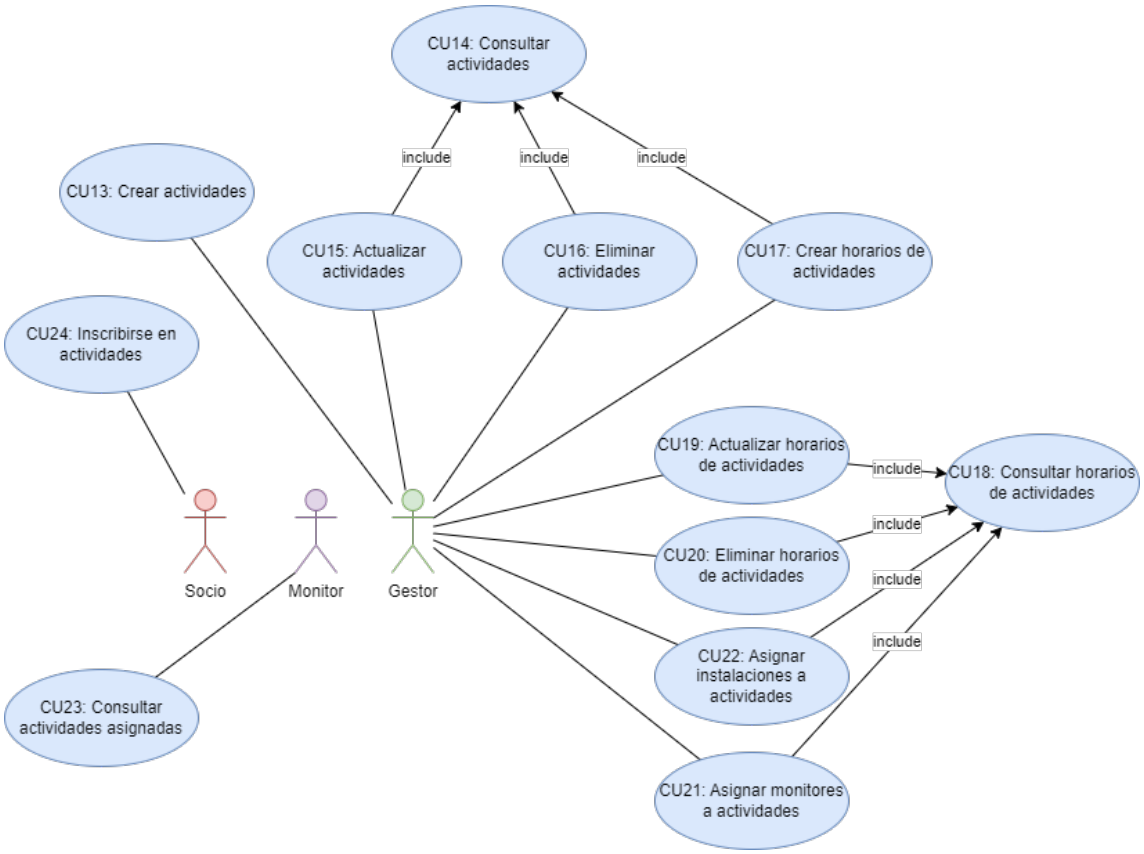

Figura 5.4: Diagrama de casos de uso: Gestión de actividades

Tabla 5.37: CU13: Crear actividades

CU13	**Crear actividades**
Descripción	El gestor crea una actividad.
Actores	Gestor.
Precondiciones	El gestor ha iniciado sesión en la aplicación.
Escenario principal	1. El gestor indica que desea crear una actividad. 2. El sistema muestra un formulario con los datos de la actividad. 3. El gestor introduce el nombre, la descripción y el deporte asociado. 4. El gestor confirma la creación de la actividad. 5. El sistema crea la actividad.
Alternativas	—
Postcondiciones	Se ha creado la actividad.

Tabla 5.38: CU14: Consultar actividades

CU14	**Consultar actividades**
Descripción	El gestor consulta la información de una actividad.
Actores	Gestor.
Precondiciones	El gestor ha iniciado sesión en la aplicación.
Escenario principal	1. El gestor indica que desea consultar una actividad. 2. El sistema muestra un listado con todas las actividades. 3. El gestor selecciona una actividad. 4. El sistema muestra la información de la actividad seleccionada.
Alternativas	2b. No existe ninguna actividad. 1. El sistema muestra un mensaje indicando que no existen actividades. 2. Se vuelve al paso 1.
Postcondiciones	—

Tabla 5.39: CU15: Actualizar actividades

CU15	**Actualizar actividades**
Descripción	El gestor actualiza la información de una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad.
Escenario principal	1. El gestor indica que desea actualizar una actividad. 2. El sistema muestra un formulario con los datos de la actividad. 3. El gestor modifica los datos de la actividad. 4. El gestor confirma la actualización de la actividad. 5. El sistema actualiza la actividad.
Alternativas	—
Postcondiciones	La información de la actividad se ha actualizado.
Incluye	CU14: Consultar actividades.

Tabla 5.40: CU16: Eliminar actividades

CU16	**Eliminar actividades**
Descripción	El gestor actualiza la información de una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad.
Escenario principal	1. El gestor indica que desea eliminar una actividad. 2. El sistema muestra un mensaje de confirmación. 3. El gestor confirma la eliminación de la actividad. 4. El sistema elimina la actividad.
Alternativas	—
Postcondiciones	Se ha eliminado la actividad.
Incluye	CU14: Consultar actividades.

Tabla 5.41: CU17: Crear horarios de actividades

CU17	**Crear horarios de actividades**
Descripción	El gestor crea un horario para una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad.
Escenario principal	1. El gestor indica que desea crear un horario para una actividad. 2. El sistema muestra un formulario con los datos del horario. 3. El gestor introduce la hora de inicio, la hora de fin y el número máximo de participantes. 4. El gestor confirma la creación del horario. 5. El sistema crea el horario.
Alternativas	4a. El gestor ha introducido una hora de inicio anterior a la hora de fin. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3. — 4b. El gestor ha introducido un número máximo de participantes menor que 0. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3. — 4c. El gestor ha introducido una hora de inicio o fin que coincide con otro horario de la actividad. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3.
Postcondiciones	Se ha creado un horario para la actividad.
Incluye	CU14: Consultar actividades.

Tabla 5.42: CU18: Consultar horarios de actividades

CU18	**Consultar horarios de actividades**
Descripción	El gestor consulta los horarios de una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad.
Escenario principal	1. El gestor indica que desea consultar los horarios de una actividad. 2. El sistema muestra los horarios de la actividad.
Alternativas	—
Postcondiciones	—
Incluye	CU14: Consultar actividades.

Tabla 5.43: CU19: Actualizar horarios de actividades

CU19	**Actualizar horarios de actividades**
Descripción	El gestor actualiza la información de un horario de una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad con un horario.
Escenario principal	1. El gestor indica que desea actualizar un horario de una actividad. 2. El sistema muestra un formulario con los datos del horario de la actividad. 3. El gestor modifica los datos del horario. 4. El gestor confirma la actualización del horario. 5. El sistema actualiza el horario.
Alternativas	4a. El gestor ha introducido una hora de inicio anterior a la hora de fin. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3. — 4b. El gestor ha introducido un número máximo de participantes menor que 0. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3. — 4c. El gestor ha introducido una hora de inicio o fin que coincide con otro horario de la actividad. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3.
Postcondiciones	Se han actualizado el horario de la actividad.
Incluye	CU18: Consultar horarios de actividades

Tabla 5.44: CU20: Eliminar horarios de actividades

CU20	**Eliminar horarios de actividades**
Descripción	El gestor elimina el horario de una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad con un horario.
Escenario principal	1. El gestor indica que desea eliminar un horario de una actividad. 2. El sistema muestra un mensaje de confirmación de la eliminación. 3. El gestor confirma la eliminación del horario. 4. El sistema elimina el horario.
Alternativas	—
Postcondiciones	Se ha eliminado el horario de la actividad.
Incluye	CU18: Consultar horarios de actividades

Tabla 5.45: CU21: Asignar monitores a actividades

CU21	**Asignar monitores a actividades**
Descripción	El gestor asigna un monitor a un horario de una actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad con un horario. 3. Existe al menos un monitor.
Escenario principal	1. El gestor indica que desea asignar un monitor a un horario de una actividad. 2. El sistema muestra un formulario con los monitores disponibles. 3. El gestor selecciona un monitor. 4. El gestor confirma la asignación del monitor. 5. El sistema asigna el monitor al horario.
Alternativas	4a. El gestor ha asignado un monitor a una actividad que ocurre a la vez que otra actividad del mismo monitor. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 2.
Postcondiciones	Se ha asignado un monitor a una actividad.
Incluye	CU18: Consultar horarios de actividades

Tabla 5.46: CU22: Asignar instalaciones a actividades

CU22	**Asignar instalaciones a actividades**
Descripción	El gestor asigna un horario de instalación a un horario de actividad.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una actividad con un horario. 3. Existe al menos una instalación del mismo deporte que la actividad con un horario disponible.
Escenario principal	1. El gestor indica que desea asignar una instalación a un horario de una actividad. 2. El sistema muestra un formulario con las instalaciones disponibles. 3. El gestor selecciona una instalación. 4. El gestor confirma la asignación de la instalación. 5. El sistema asigna la instalación al horario.
Alternativas	4a. El gestor ha asignado una instalación a una actividad que ocurre a la vez que otra actividad de la misma instalación. 1. El sistema muestra un mensaje de error. 2. Se vuelve al paso 3.3.
Postcondiciones	Se ha asignado una instalación a una actividad.
Incluye	CU18: Consultar horarios de actividades

Tabla 5.47: CU23: Consultar actividades asignadas

CU23	**Consultar actividades asignadas**
Descripción	El monitor consulta las actividades asignadas que tiene asignadas.
Actores	Monitor.
Precondiciones	1. El monitor ha iniciado sesión en la aplicación. 2. El monitor tiene al menos una actividad asignada.
Escenario principal	1. El monitor solicita consultar las actividades asignadas. 2. El sistema muestra las actividades asignadas al monitor.
Alternativas	—
Postcondiciones	—

Tabla 5.48: CU24: Inscribirse en actividades

CU24	**Inscribirse en actividades**
Descripción	El socio se inscribe en una actividad.
Actores	Socio.
Precondiciones	El socio ha iniciado sesión en la aplicación, existe al menos una actividad y el socio no está inscrito en la actividad.
Escenario principal	1. El socio indica que desea inscribirse en una actividad. 2. El sistema muestra información de todas las actividades con horarios disponibles. 3. El socio selecciona una actividad. 4. El sistema presenta los horarios disponibles de la actividad seleccionada. 5. El socio selecciona un horario. 6. El socio confirma la inscripción. 7. El sistema inscribe al socio en la actividad.
Alternativas	6a. El socio se ha inscrito en una actividad que ya está completa. 1. El sistema muestra un mensaje de error indicando que la actividad está completa. 2. Se vuelve al paso 4.
Postcondiciones	El socio se ha inscrito en la actividad.

5.6.4. Gestión de competiciones

En la figura 5.5 se muestra el diagrama de casos de uso correspondiente a la gestión de competiciones.

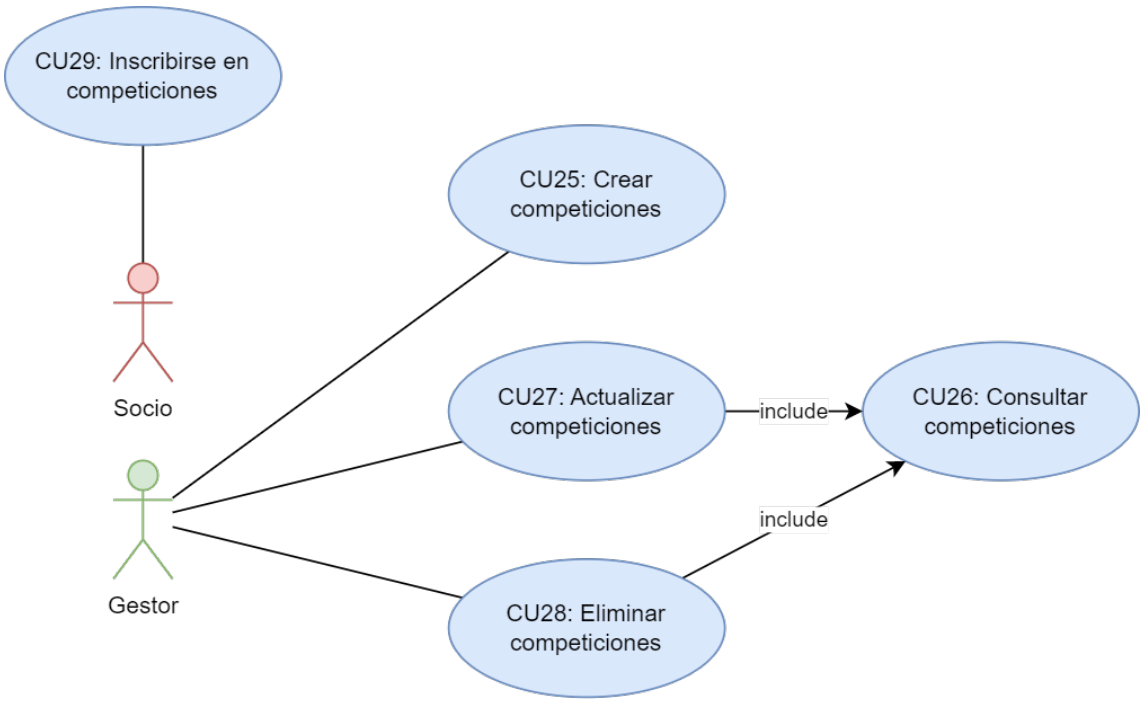

Figura 5.5: Diagrama de casos de uso: Gestión de competiciones

Tabla 5.49: CU25: Crear competiciones

CU25	**Crear competiciones**
Descripción	El gestor crea una competición.
Actores	Gestor.
Precondiciones	El gestor ha iniciado sesión en la aplicación.
Escenario principal	1. El gestor indica que desea crear una competición. 2. El sistema muestra un formulario para crear la competición. 3. El gestor introduce el nombre, la fecha de inicio de la inscripción, la fecha de comienzo de la competición, la fecha de fin de la competición y el número de participantes de la competición. 4. El gestor confirma la creación de la competición. 5. El sistema crea la competición.
Alternativas	4a. El gestor ha introducido una fecha de inicio de inscripción anterior a la fecha actual. 1. El sistema muestra un mensaje de error indicando que la fecha de inscripción debe ser posterior a la fecha actual. 2. Se vuelve al paso 3. — 4b. El gestor ha introducido una fecha de comienzo anterior a la fecha de la fecha de inicio de la inscripción. 1. El sistema muestra un mensaje de error indicando que la fecha de comienzo debe ser posterior a la fecha de inicio. 2. Se vuelve al paso 3. — 4c. El gestor ha introducido una fecha de fin anterior a la fecha de inicio. 1. El sistema muestra un mensaje de error indicando que la fecha de fin debe ser posterior a la fecha de inicio. 2. Se vuelve al paso 3.
Postcondiciones	Se ha creado una competición.

Tabla 5.50: CU26: Consultar competiciones

CU26	**Consultar competiciones**
Descripción	El usuario accede a la información de las competiciones.
Actores	Usuario.
Precondiciones	1. El usuario ha iniciado sesión en la aplicación. 2. Existe al menos una competición.
Escenario principal	1. El usuario indica que desea consultar las competiciones. 2. El sistema muestra información de todas las competiciones. 3. El usuario selecciona una competición. 4. El sistema muestra información de la competición seleccionada.
Alternativas	—
Postcondiciones	—

Tabla 5.51: CU27: Actualizar competiciones

CU27	**Actualizar competiciones**
Descripción	El gestor actualiza una competición.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una competición.
Escenario principal	1. El gestor indica que desea actualizar una competición. 2. El sistema muestra un formulario con los datos de la competición. 3. El gestor actualiza los datos de la competición. 4. El gestor confirma la actualización de la competición. 5. El sistema actualiza la competición.
Alternativas	4a. El gestor ha introducido una fecha de inicio de inscripción anterior a la fecha actual. 1. El sistema muestra un mensaje de error indicando que la fecha de inscripción debe ser posterior a la fecha actual. 2. Se vuelve al paso 3. — 4b. El gestor ha introducido una fecha de comienzo anterior a la fecha de la fecha de inicio de la inscripción. 1. El sistema muestra un mensaje de error indicando que la fecha de comienzo debe ser posterior a la fecha de inicio. 2. Se vuelve al paso 3. — 4c. El gestor ha introducido una fecha de fin anterior a la fecha de inicio. 1. El sistema muestra un mensaje de error indicando que la fecha de fin debe ser posterior a la fecha de inicio. 2. Se vuelve al paso 3.
Postcondiciones	Se ha actualizado la competición.
Incluye	CU26: Consultar competiciones

Tabla 5.52: CU28: Eliminar competiciones

CU28	**Eliminar competiciones**
Descripción	El gestor elimina una competición.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una competición.
Escenario principal	1. El gestor accede a la información de las competiciones. 2. El gestor elige una competición. 3. El gestor elimina la competición.
Alternativas	—
Postcondiciones	Se ha eliminado la competición.
Incluye	CU26: Consultar competiciones

Tabla 5.53: CU29: Inscribirse en competiciones

CU29	**Inscribirse en competiciones**
Descripción	El socio se inscribe en una competición.
Actores	Socio.
Precondiciones	1. El socio ha iniciado sesión en la aplicación. 2. Existe al menos una competición. 3. La competición no está completa. 4. La competición se encuentra en proceso de inscripción. 5. El socio no se ha inscrito a la competición.
Escenario principal	1. El socio indica que desea inscribirse en una competición. 2. El sistema presenta las competiciones que se encuentran en proceso de inscripción. 3. El socio elige una competición. 4. El sistema muestra información en detalle de la competición. 5. El socio confirma la inscripción. 6. El sistema inscribe al socio en la competición.
Alternativas	5a. El socio se inscribe en una competición que ya está completa. 1. El sistema muestra un mensaje de error indicando que la competición está completa. 2. Se vuelve al paso 2.
Postcondiciones	El socio se ha inscrito en la competición.

5.6.5. Equipos y actas

En la figura 5.6 se muestra el diagrama de casos de uso correspondiente a los equipos y las actas.

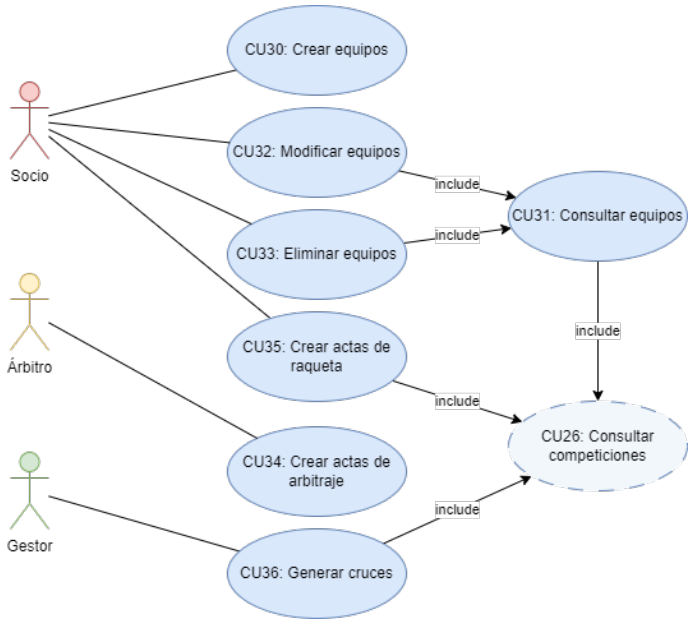

Figura 5.6: Diagrama de casos de uso: Equipos y actas

Tabla 5.54: CU30: Crear equipos

CU30	**Crear equipos**
Descripción	El socio crea un equipo para una competición.
Actores	Socio.
Precondiciones	1. El socio ha iniciado sesión en la aplicación. 2. Existe al menos una competición. 3. La competición no está completa. 4. La competición se encuentra en proceso de inscripción. 5. El socio se ha inscrito a la competición.
Escenario principal	1. El socio accede a una competición en la que está inscrito. 2. El socio indica que desea crear un equipo. 3. El sistema muestra un formulario para crear un equipo. 4. El socio añade los datos de los integrantes del equipo. 5. El socio confirma la creación del equipo. 6. El sistema crea el equipo.
Alternativas	—
Postcondiciones	Se ha creado un equipo.

Tabla 5.55: CU31: Consultar equipos

CU31	**Consultar equipos**
Descripción	El socio consulta los equipos de la competición.
Actores	Socio.
Precondiciones	1. El socio ha iniciado sesión en la aplicación. 2. Existe al menos una competición. 3. La competición no está completa. 4. La competición se encuentra en proceso de inscripción. 5. El socio se ha inscrito a la competición.
Escenario principal	1. El socio accede a la información de las competiciones. 2. El socio elige una competición en la que está inscrito. 3. El socio indica que desea consultar los equipos de la competición. 4. El sistema muestra los equipos de la competición. 5. El socio elige un equipo. 6. El sistema muestra la información del equipo.
Alternativas	—
Postcondiciones	—
Incluye	CU26: Consultar competiciones

Tabla 5.56: CU32: Modificar equipos

CU32	**Modificar equipos**
Descripción	El socio modifica su equipo de la competición.
Actores	Socio.
Precondiciones	El socio ha iniciado sesión en la aplicación, existe al menos una competición, el socio está inscrito en la competición, la competición no ha empezado y el socio tiene un equipo creado para la competición.
Escenario principal	1. El socio accede a la información de las competiciones. 2. El socio elige una competición en la que está inscrito. 3. El socio modifica su equipo de la competición.
Alternativas	—
Postcondiciones	El socio ha modificado su equipo de la competición.
Incluye	CU31: Consultar equipos

Tabla 5.57: CU33: Eliminar equipos

CU33	**Eliminar equipos**
Descripción	El socio elimina su equipo de la competición.
Actores	Socio.
Precondiciones	1. El socio ha iniciado sesión en la aplicación. 2. Existe al menos una competición. 3. La competición no está completa. 4. La competición se encuentra en proceso de inscripción. 5. El socio se ha inscrito a la competición. 6. El socio tiene un equipo creado para la competición.
Escenario principal	1. El socio accede a la información de la competición en la que está inscrito. 3. El socio indica que desea eliminar su equipo. 4. El sistema elimina el equipo del socio.
Alternativas	—
Postcondiciones	El socio ha eliminado su equipo de la competición.
Incluye	CU31: Consultar equipos

Tabla 5.58: CU34: Crear actas de arbitraje

CU34	**Crear actas de arbitraje**
Descripción	El árbitro crea un acta para un partido.
Actores	Árbitro.
Precondiciones	1. El árbitro ha iniciado sesión en la aplicación. 2. El árbitro tiene asignado un partido. 3. El acta aún no ha sido realizada.
Escenario principal	1. El árbitro accede a la información de los partidos que tiene asignados. 2. El sistema muestra los partidos que tiene asignados. 3. El árbitro elige un partido. 4. El sistema muestra la información del partido. 5. El árbitro indica que desea crear un acta para el partido. 6. El sistema crea un acta para el partido. 7. El árbitro introduce los datos del acta. 8. El árbitro indica que desea finalizar el acta. 9. El sistema finaliza el acta.
Alternativas	—
Postcondiciones	Se ha creado un acta de arbitraje para el partido.

Tabla 5.59: CU35: Crear actas de raqueta

CU35	**Crear actas de raqueta**
Descripción	El socio participante del partido crea un acta para un partido.
Actores	Socio.
Precondiciones	1. El socio ha iniciado sesión en la aplicación. 2. El socio participa en una competición. 3. El socio acaba de jugar un partido de la competición.
Escenario principal	1. El socio accede a la información de la competición en la que participa. 2. El sistema muestra la información de la competición. 3. El socio accede a la información de los partidos de la competición. 4. El sistema muestra los partidos de la competición. 5. El socio elige el partido que acaba de jugar. 6. El sistema muestra la información del partido. 7. El socio indica que desea crear un acta para el partido. 8. El sistema crea un acta para el partido. 9. El socio introduce los datos del acta. 10. El socio confirma que desea finalizar el acta. 11. El sistema finaliza el acta.
Alternativas	—
Postcondiciones	Se ha creado un acta para el partido.
Incluye	CU26: Consultar competiciones

Tabla 5.60: CU36: Generar cruces

CU36	**Generar cruces**
Descripción	El gestor genera los cruces de una competición.
Actores	Gestor.
Precondiciones	1. El gestor ha iniciado sesión en la aplicación. 2. Existe al menos una competición que aun no ha empezado y aún no se han generado los cruces.
Escenario principal	1. El gestor accede a la competición. 2. El sistema muestra la información de la competición. 3. El gestor indica que desea generar los cruces de la competición. 4. El sistema genera los cruces de la competición.
Alternativas	—
Postcondiciones	Se han generado los cruces de la competición.
Incluye	CU26: Consultar competiciones

CAPÍTULO 6
Diseño de la aplicación

En esta sección se describe el diseño de la aplicación. Hércules es una *webapp* o aplicación web. A diferencia de las aplicaciones tradicionales que se instalan en un dispositivo, las web apps se alojan en un servidor y se accede a ellas a través de internet. Las web apps son accesibles desde cualquier dispositivo con un navegador web, como por ejemplo un ordenador, una tablet o un teléfono móvil. Además, las web apps son multiplataforma, es decir, se pueden utilizar desde cualquier sistema operativo como Windows, macOS, Linux, Android o iOS. También, las web apps no necesitan ser instaladas en el dispositivo, por lo que no ocupan espacio en el dispositivo y no necesitan ser actualizadas.

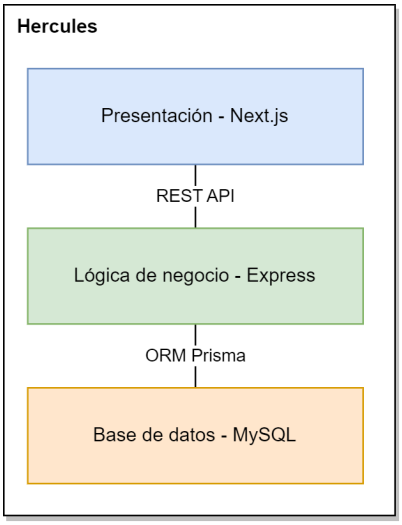

Figura 6.1: Arquitectura de Hércules

Como se puede observar en la figura 6.1, Hércules tiene una arquitectura de tres capas. La capa de presentación o frontend es la capa que se encarga de mostrar la interfaz de usuario de la aplicación. La capa de negocio o backend es la capa que se encarga de gestionar los datos de la aplicación. Finalmente, la capa de datos o base de datos es la capa que se encarga de almacenar los datos de la aplicación. La comunicación entre la capa de presentación y la de lógica de negocio se realiza mediante una API REST. La comunicación entre la capa de lógica de negocio y la de datos se realiza mediante instrucciones SQL usando un ORM *(Object-Relational Mapping)*.

6.1 Modelo de base de datos

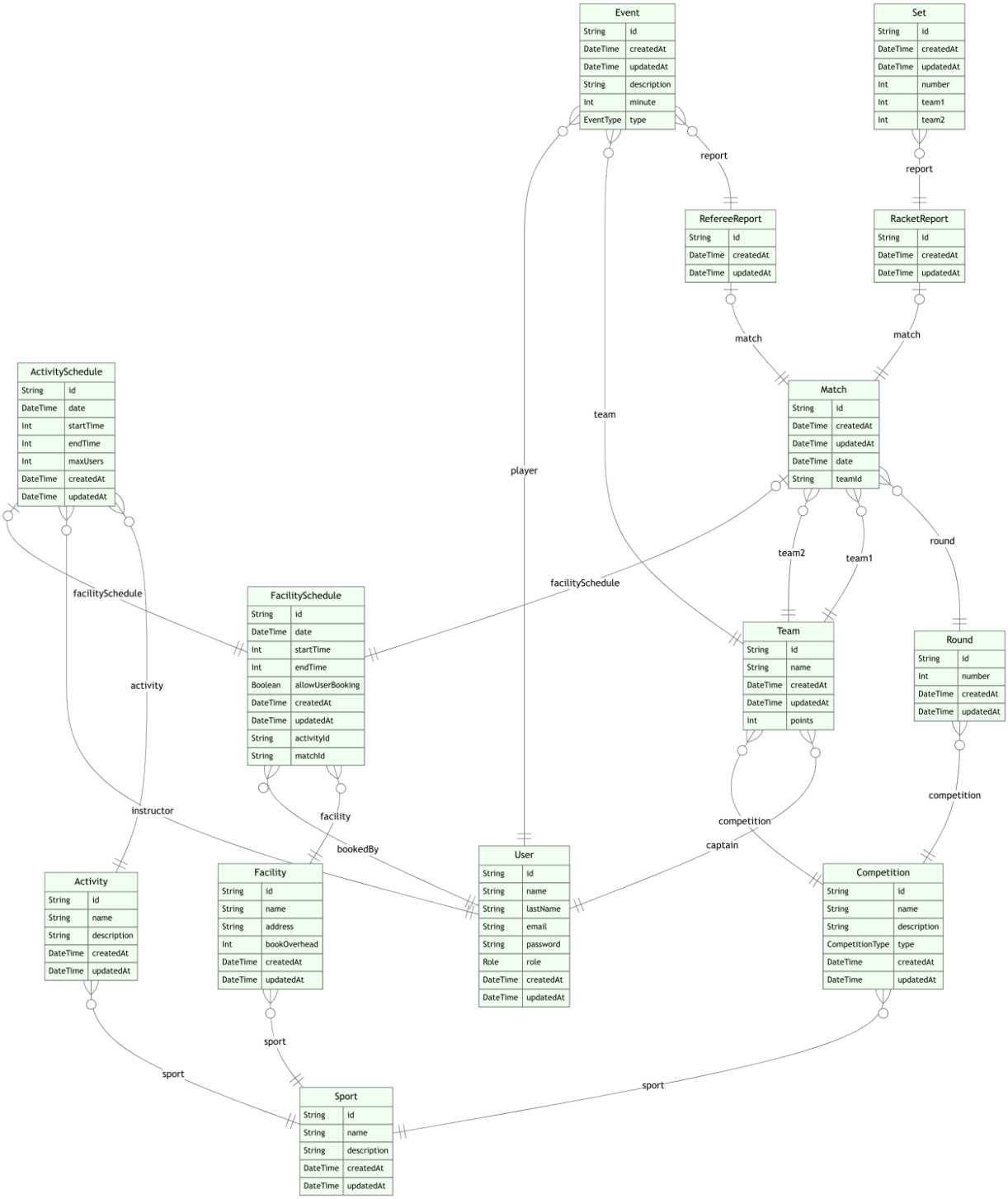

Figura 6.2: Modelo de base de datos

La base de datos es de tipo SQL y está definida usando Prisma, un ORM *(Object-Relational Mapping)* para Node.js y TypeScript. Prisma permite definir la base de datos usando un lenguaje de modelado de datos y genera automática-

mente el esquema de la base de datos. Además, Prisma permite realizar peticiones a la base de datos usando un lenguaje de consulta de datos.

Los modelos se establecen en el archivo `schema.prisma` del backend y se definen usando el lenguaje de modelado de datos de Prisma. Para migrar los modelos a la base de datos se usa el comando `prisma migrate dev`. Este comando crea las tablas de la base de datos y las relaciones entre las tablas. La base de datos utilizada es MySQL.

El sistema de base de datos SQL está formado por las tablas que se muestran en la figura 6.2. La figura es un modelo de tipo entidad relación que muestra las tablas de la base de datos y las relaciones entre las tablas. En el modelo, los extremos de las relaciones con dos líneas perpendiculares indican que el extremo es `1..1`. Los extremos de las relaciones con una línea perpendicular y un círculo indican que el extremo es `0..1`. Los extremos de las relaciones con tan solo un círculo indican que el extremo es `0..*`. Finalmente, los extremos de las relaciones con tan solo una línea perpendicular indican que el extremo es `1..*`.

Es importante recalcar que todas las tablas de la base de datos tienen una columna con un identificador único. Este identificador es un identificador *uuidv4*[1] genera automáticamente cuando se crea una nueva fila en la tabla (excepto el de la tabla *«User»*, que es su DNI). Además, todas las tablas tienen una columna `createdAt` y una columna `updatedAt`. La columna `createdAt` almacena la fecha de creación de la fila y la columna `updatedAt` almacena la fecha de la última actualización de la fila. Las tablas de la base de datos son las siguientes:

Role: es una enumeración que almacena los roles de los usuarios. Los roles de los usuarios pueden ser *MEMBER* (socio), *ADMIN* (gestor), *REFEREE* (árbitro) o *INSTRUCTOR* (monitor).

«User»: la tabla *«User»* almacena la información de los usuarios de la aplicación.

- `id`: DNI del usuario. Es el identificador único de la tabla.
- `name`: nombre del usuario.
- `lastName`: apellido del usuario.
- `email`: correo electrónico del usuario.
- `password`: contraseña del usuario.
- `role`: rol del usuario. Puede ser cualquiera de los valores de la enumeración *Role*.

«Sport»: la tabla *«Sport»* almacena la información de los deportes.

- `id`: identificador único del deporte. Es el identificador único de la tabla.
- `name`: nombre del deporte.

[1] El identificador **uuidv4** es un identificador único universal formado por un número de 128 bits que se representa como una cadena de 32 dígitos hexadecimales separados por guiones que se genera de forma aleatoria con poca probabiliad de que se generen dos identificadores iguales. [21]

- `description`: descripción del deporte.

«Facility»: la tabla *«Facility»* almacena la información de las instalaciones.

- `id`: identificador único de la instalación. Es el identificador único de la tabla.
- `name`: nombre de la instalación.
- `address`: dirección de la instalación.
- `bookOverhead`: el máximo de días de antelación para reservar una pista.
- `sportId`: identificador único del deporte. Es una clave foránea que referencia a la tabla *«Sport»*.

«FacilitySchedule»: la tabla *«FacilitySchedule»* almacena la información de los horarios de las instalaciones.

- `id`: identificador único del horario de la instalación. Es el identificador único de la tabla.
- `date`: fecha del horario.
- `startTime`: hora de inicio de la franja del horario.
- `endTime`: hora de inicio de la franja del horario.
- `facilityId`: identificador único de la instalación. Es una clave foránea que referencia a la tabla *«Facility»*.
- `bookedById`: DNI del usuario que ha reservado la franja del horario. Es una clave foránea que referencia a la tabla *«User»*.
- `activityId`: identificador único de la actividad. Es una clave foránea que referencia a la tabla *«Activity»*.
- `matchId`: identificador único del partido. Es una clave foránea que referencia a la tabla *«Match»*.

«Activity»: la tabla *«Activity»* almacena la información de las actividades.

- `id`: identificador único de la actividad. Es el identificador único de la tabla.
- `name`: nombre de la actividad.
- `description`: descripción de la actividad.
- `sportId`: identificador único del deporte. Es una clave foránea que referencia a la tabla *«Sport»*.

«ActivitySchedule»: la tabla *«ActivitySchedule»* almacena la información de los horarios de las actividades.

- `id`: identificador único del horario de la actividad. Es el identificador único de la tabla.
- `date`: fecha del horario de actividad.
- `startTime`: hora de inicio de la franja del horario.
- `endTime`: hora de inicio de la franja del horario.
- `maxUsers`: número máximo de usuarios que pueden apuntarse a la actividad.
- `users`: número de usuarios que están apuntados a la actividad. Es una relación muchos a muchos con la tabla *«User»*.
- `activityId`: identificador único de la actividad. Es una clave foránea que referencia a la tabla *«Activity»*.
- `instructorId`: DNI del usuario que ha reservado la franja del horario. Es una clave foránea que referencia a la tabla *«User»*.
- `facilityScheduleId`: identificador único del horario de la instalación donde se va a realizar la actividad. Es una clave foránea que referencia a la tabla *«FacilitySchedule»*.

CompetitionType: es una enumeración que almacena los tipos de competiciones. Los tipos pueden ser *RACKET* (raqueta) o *REFEREE* (árbitro).

«Competition»: la tabla *«Competition»* almacena la información de las competiciones.

- `id`: identificador único de la competición. Es el identificador único de la tabla.
- `name`: nombre de la competición.
- `description`: descripción de la competición.
- `type`: tipo de competición. Puede ser cualquiera de los valores de la enumeración *CompetitionType*.
- `sportId`: identificador único del deporte. Es una clave foránea que referencia a la tabla *«Sport»*.

«Team»: la tabla *«Team»* almacena la información de los equipos.

- `id`: identificador único del equipo. Es el identificador único de la tabla.
- `name`: nombre del equipo.
- `points`: puntos del equipo.
- `players`: jugadores del equipo. Es una relación muchos a muchos con la tabla *«User»*.

- `captain`: DNI del capitán del equipo. Es una clave foránea que referencia a la tabla *«User»*.
- `competitionId`: identificador único de la competición. Es una clave foránea que referencia a la tabla *«Competition»*.

«Round»: la tabla *«Round»* almacena la información de las rondas de las competiciones.

- `id`: identificador único de la ronda. Es el identificador único de la tabla.
- `number`: número de la ronda.
- `competitionId`: identificador único de la competición. Es una clave foránea que referencia a la tabla *«Competition»*.

«Match»: la tabla *«Match»* almacena la información de los partidos de las competiciones.

- `id`: identificador único del partido. Es el identificador único de la tabla.
- `date`: fecha del partido.
- `team1Id`: identificador único del equipo 1. Es una clave foránea que referencia a la tabla *«Team»*.
- `team2Id`: identificador único del equipo 2. Es una clave foránea que referencia a la tabla *«Team»*.
- `roundId`: identificador único de la ronda. Es una clave foránea que referencia a la tabla *«Round»*.
- `facilityScheduleId`: identificador único del equipo ganador. Es una clave foránea que referencia a la tabla *«Team»*.

«RacketReport»: la tabla *«RacketReport»* almacena la información de las actas de las competiciones de tipo *RACKET*.

- `id`: identificador único del acta. Es el identificador único de la tabla.
- `matchId`: identificador único del partido. Es una clave foránea que referencia a la tabla *«Match»*.

«Set»: la tabla *«Set»* almacena la información de los sets de los partidos de las competiciones de tipo *RACKET*.

- `id`: identificador único del set. Es el identificador único de la tabla.
- `number`: número del set.
- `team1`: puntuación del equipo 1.
- `team2`: puntuación del equipo 2.

- `reportId`: identificador único del acta. Es una clave foránea que referencia a la tabla *«RacketReport»*.

«RefereeReport»: la tabla *«RefereeReport»* almacena la información de las actas de las competiciones de tipo *REFEREE*.

- `id`: identificador único del acta. Es el identificador único de la tabla.
- `matchId`: identificador único del partido. Es una clave foránea que referencia a la tabla *«Match»*.

EventType: es una enumeración que almacena los tipos de eventos. Los tipos pueden ser *GOAL* (gol), *SUBSTITUTION* (sustitución), *INJURY* (lesión) o *OTHER* (otro).

«Event»: la tabla *«Event»* almacena la información de los eventos de las actas de las competiciones de tipo *REFEREE*.

- `id`: identificador único del evento. Es el identificador único de la tabla.
- `type`: tipo del evento. Puede ser cualquiera de los valores de la enumeración *EventType*.
- `description`: descripción del evento.
- `minute`: minuto del evento.
- `teamId`: identificador único del equipo. Es una clave foránea que referencia a la tabla *«Team»*.
- `playerId`: identificador único del jugador. Es una clave foránea que referencia a la tabla *«User»*.
- `reportId`: identificador único del acta. Es una clave foránea que referencia a la tabla *«RefereeReport»*.

CAPÍTULO 7
Diseño de la interfaz de usuario

En esta sección se detalla el proceso de diseño de la interfaz de usuario. En el contexto de la ingeniería del software, el diseño de las interfaces de usuario es cada vez más importante debido a la creciente demanda de sistemas y aplicaciones intuitivas y fáciles de usar. Una vez obtenidos los requisitos del sistema y los objetivos, se desarrollan unos *wireframes* o bocetos de interfaz de usuario en el que se establecen los elementos de la interfaz y la disposición de los elementos. Después, se adaptarán las guías de diseño de *Material Design 3* de Google para crear una interfaz de usuario intuitiva y fácil de usar.

7.1 Bocetos de interfaz de usuario

Los bocetos se han desarrollado usando una herramienta de diseño de interfaces de usuario. En el caso de Hércules, se ha usado Drawio, una herramienta multiplataforma de código abierto desarrollada usando HTML5 y JavaScript para crear diagramas UML, diagramas de flujo y bocetos.

En las figuras 7.1, 7.2, 7.3, 7.4 y 7.5 se pueden ver los bocetos de las pantallas de inicio de sesión, creación de horario para la instalación desde el punto de vista del gestor, creación de horario para la actividad desde el punto de vista del gestor, selección de deporte desde el punto de vista del socio y reserva de instalación desde el punto de vista del socio respectivamente.

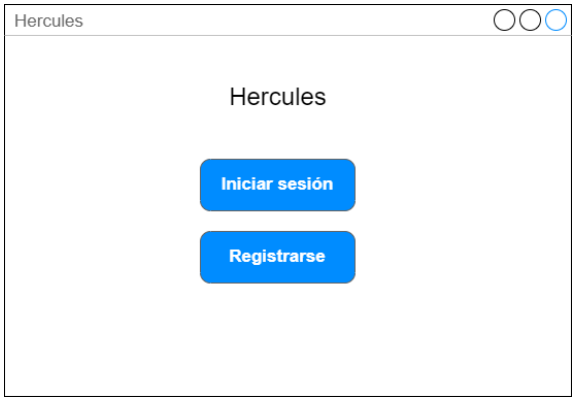

Figura 7.1: Boceto de la pantalla de inicio de sesión

Figura 7.2: Boceto de la pantalla de creación de horario para la instalación desde el punto de vista del gestor

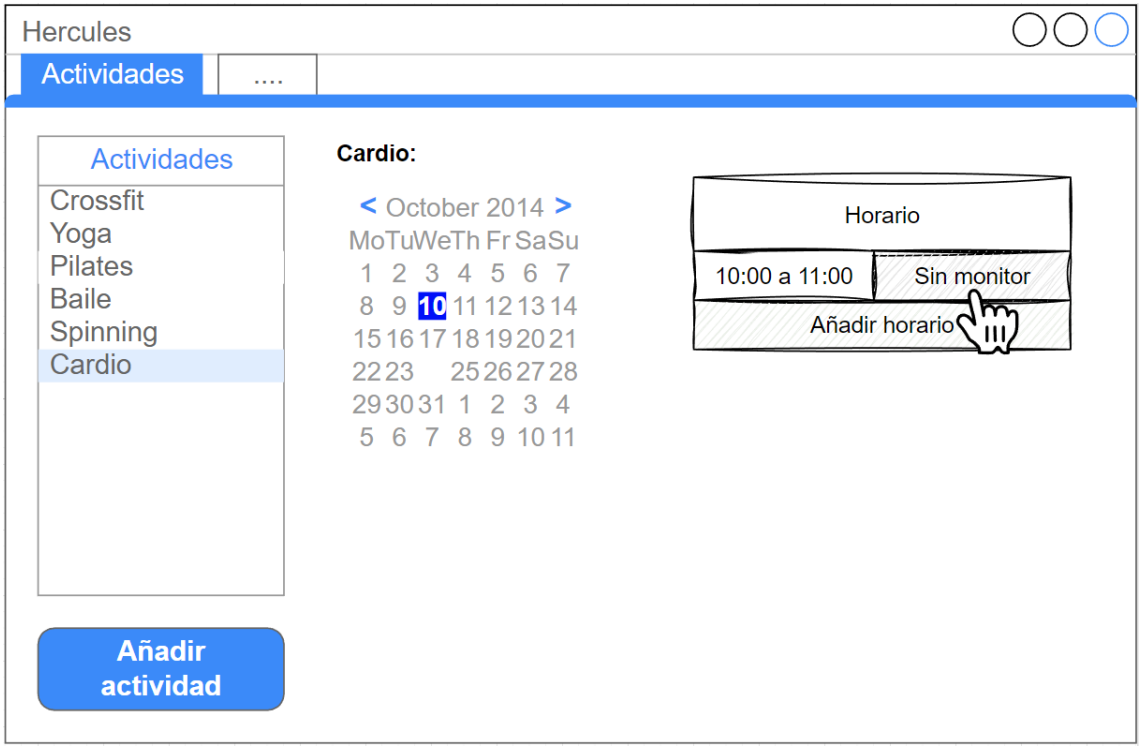

Figura 7.3: Boceto de la pantalla de creación de horario para la actividad desde el punto de vista del gestor

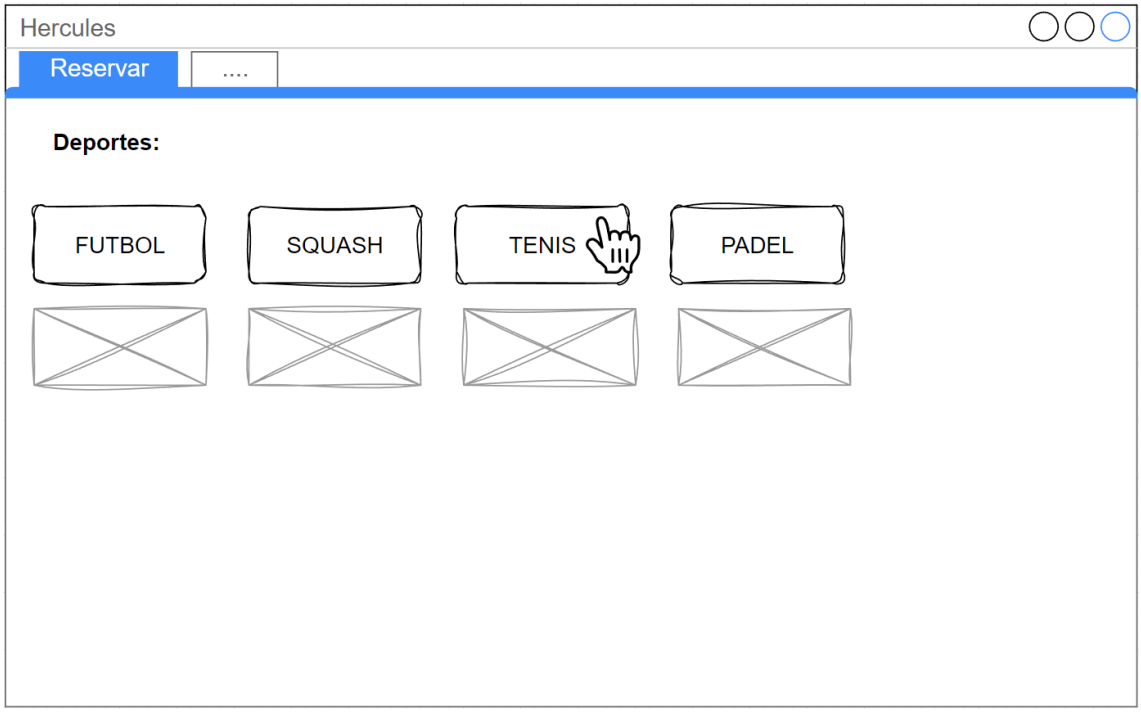

Figura 7.4: Boceto de la pantalla de selección de deporte desde el punto de vista del socio

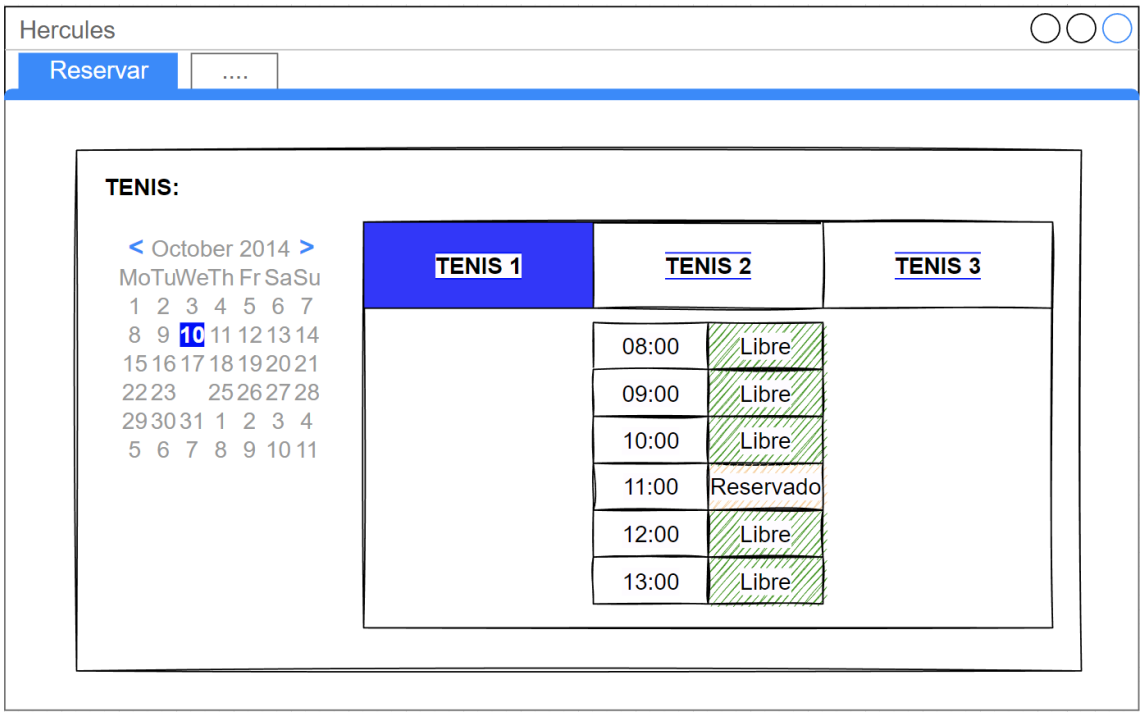

Figura 7.5: Boceto de la pantalla de reserva de instalación desde el punto de vista del socio

7.2 Guías de diseño de Material Design 3

Una vez obtenidos los bocetos de las pantallas de la aplicación, se adaptarán las guías de diseño de *Material Design 3* de Google para crear una interfaz de

usuario intuitiva y fácil de usar. *Material Design* es un sistema de diseño visual desarrollado por Google en 2014. El sistema de diseño que se aplicará será la última versión de *Material Design*, *Material Design 3*, que se lanzó en 2021.

7.2.1. Color tokens

Uno de los elementos más importantes de *Material Design 3* son los *color tokens*. Estos *tokens* son valores predefinidos que representan colores específicos y se utilizan de manera consistente en todo el sistema de diseño. El propósito principal de los color tokens es permitir una rápida personalización y adaptación del aspecto visual de una aplicación o sitio web, manteniendo al mismo tiempo una apariencia coherente y armoniosa. Al utilizar *tokens* en lugar de valores de color estáticos, se logra una mayor flexibilidad y facilidad para realizar cambios en el diseño. También, cada *token* tiene su color equivalente para el modo oscuro, lo que permite que la aplicación se adapte automáticamente al modo oscuro sin tener que cambiar los colores manualmente.

Para utilizar los tokens en la aplicación web se guardan en un archivo *tokens.css* y se importan en el archivo *globals.css* de la aplicación web. En la figura 7.6 se muestran algunos de los *color tokens* utilizados en la aplicación web:

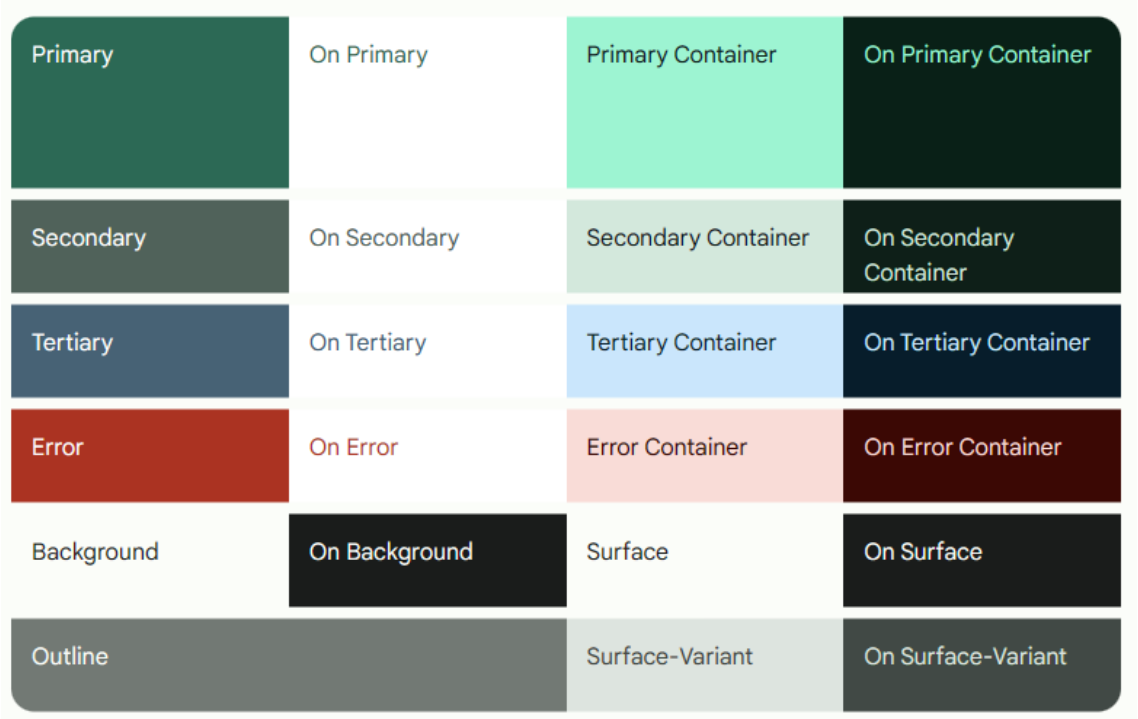

Figura 7.6: *Color tokens* utilizados en el modo claro de la aplicación web

7.2.2. Librería de componentes

En el momento del desarrollo del proyecto no existía una librería de componentes de *Material Design 3* para los navegadores web. Por lo tanto, se ha desarrollado un abanico de componentes reutilizables que siguen las guías de diseño de

Material Design 3 para la aplicación web. Para facilitar el desarrollo de los componentes se ha usado la librería *Radix UI*[1] y se ha personalizado el aspecto visual de los componentes para que se adapten a las guías de diseño de *Material Design 3*. En la figura 7.7 se muestran algunos de los componentes desarrollados:

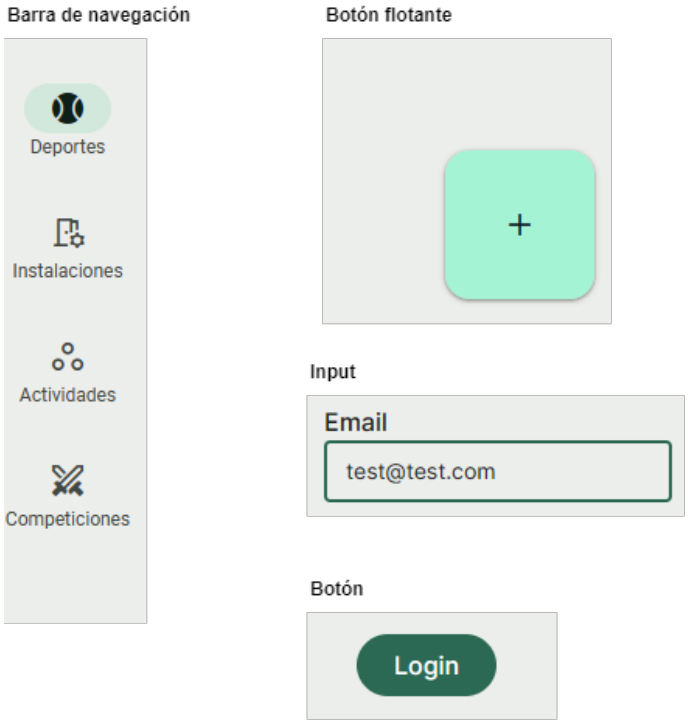

Figura 7.7: Componentes desarrollados para la aplicación web

7.3 Diseño de interfaz resultante

Una vez se han definido los bocetos de las pantallas de la aplicación y se han adaptado las guías de diseño de *Material Design 3* para crear una interfaz de usuario intuitiva y fácil de usar, se ha desarrollado la interfaz de usuario resultante. En la figura 7.8 y 7.9 se muestra el diseño de la interfaz de usuario resultante de la aplicación web:

[1] ***Radix UI Primitives*** es una biblioteca de componentes de interfaz de usuario de bajo nivel con un enfoque en la accesibilidad, la personalización y la experiencia del desarrollador. Los componentes se presentan sin estilos, por lo que son totalmente personalizables y se pueden usar para construir sistemas de diseño.

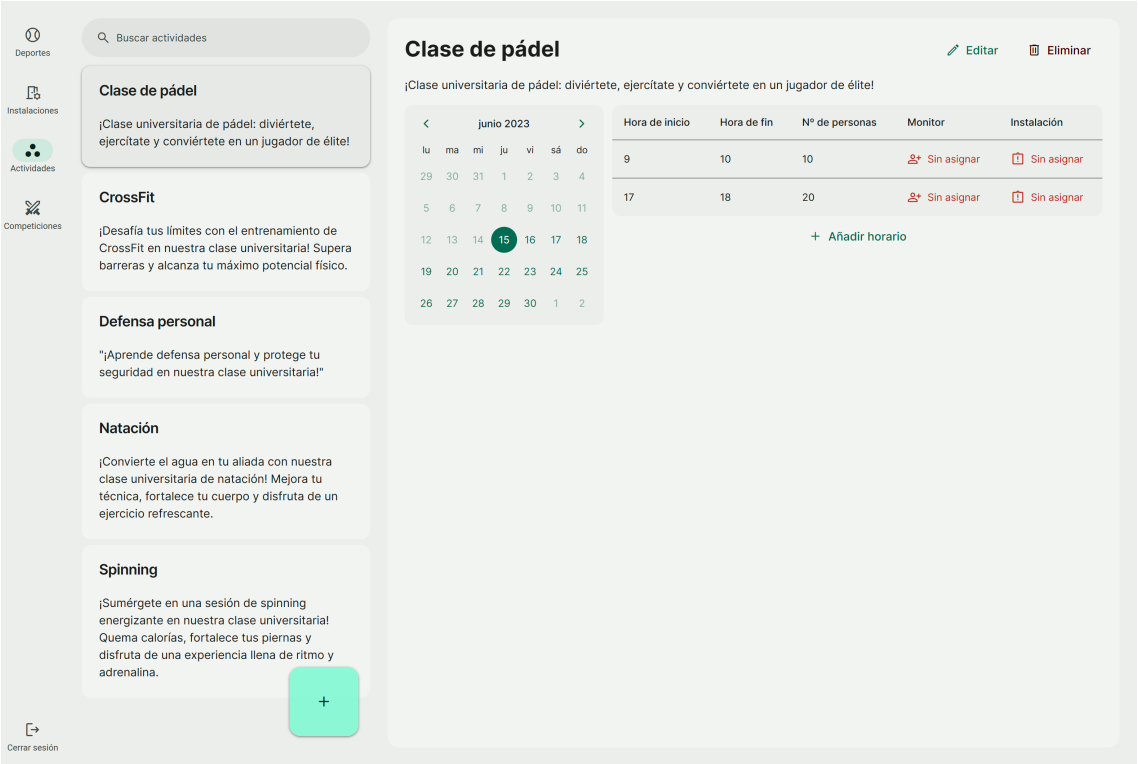

Figura 7.8: Diseño de la interfaz de usuario resultante de la aplicación web en modo claro

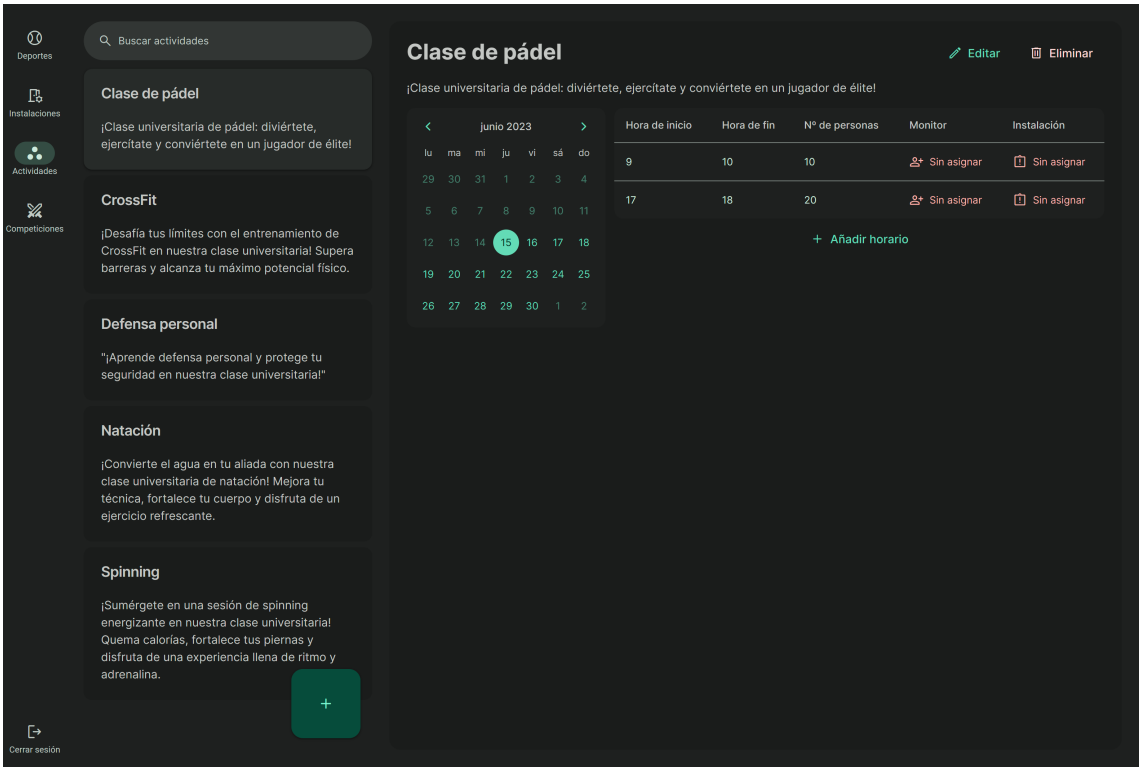

Figura 7.9: Diseño de la interfaz de usuario resultante de la aplicación web en modo oscuro

CAPÍTULO 8
Desarrollo de la aplicación

8.1 Herramientas utilizadas

En esta sección se describen las herramientas utilizadas para el desarrollo de la aplicación.

Visual Studio Code: es un editor de texto desarrollado por Microsoft. Es un editor de texto muy ligero y rápido que dispone de una gran cantidad de extensiones para facilitar el desarrollo de aplicaciones web. Además, dispone de un depurador integrado que permite depurar aplicaciones web desarrolladas con Node.js y Express, lo cual es muy útil para el desarrollo de la aplicación.

Node.js: es un entorno de ejecución de JavaScript que permite ejecutar código JavaScript fuera del navegador. Node.js permite desarrollar aplicaciones web con JavaScript tanto en el lado del cliente como en el lado del servidor. En el caso de la aplicación, Node.js se utiliza para desarrollar el servidor de la aplicación.

Express: es un *framework* de Node.js que permite desarrollar aplicaciones web de forma rápida y sencilla. Express se encarga de la configuración de las herramientas necesarias para Node.js como el servidor HTTP o el enrutador. En nuestro caso se ha usado Express para desarrollar un *API*[1] *REST*[2] que permite acceder a los datos de la aplicación.

Git: es un sistema de control de versiones distribuido. Permite mantener un historial de cambios de los archivos de la aplicación. Además, permite trabajar en equipo de forma eficiente. También permite crear ramas para desarrollar nuevas funcionalidades sin afectar al código de la aplicación.

React: es una librería de JavaScript desarrollada por Meta. Permite desarrollar interfaces de usuario de forma rápida y sencilla. Además, permite crear componentes reutilizables que facilitan el desarrollo de aplicaciones web.

[1] **API** *(Application Programming Interface)*: permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados.

[2] **REST** *(REpresentational State Transfer)*: es un estilo de arquitectura para realizar comunicaciones entre cliente y servidor. *REST* es una interfaz para conectar varios sistemas basados en el protocolo *HTTP* y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como *XML* (Extensible Markup Language) y *JSON* (JavaScript Object Notation).

Next.js: es un *framework* de React que permite desarrollar aplicaciones web de forma rápida y sencilla. Next.js se encarga de la configuración de las herramientas necesarias para React como Webpack o Babel. Además, Next.js viene con funcionalidades útiles para el desarrollo de aplicaciones como:

- **Enrutamiento**: permite crear rutas para cada página de la aplicación.
- **Servidor de desarrollo**: permite ejecutar la aplicación en un servidor de desarrollo local.
- **Generación de páginas estáticas**: permite generar páginas estáticas para mejorar el rendimiento de la aplicación.
- **Preprocesado de CSS**: permite utilizar Sass o TailwindCSS para escribir CSS.
- **Preprocesado de imágenes**: permite optimizar las imágenes para mejorar el rendimiento de la aplicación.
- **Prefetching**: permite precargar los recursos de las páginas a las que se puede acceder desde la página actual.
- **Soporte para TypeScript**: permite desarrollar la aplicación utilizando TypeScript.

TypeScript: es un lenguaje de programación que añade tipado estático a JavaScript. TypeScript permite detectar errores en tiempo de compilación y ofrece una mejor experiencia de desarrollo gracias a la ayuda del autocompletado. Además, TypeScript permite utilizar las últimas características de JavaScript como las clases, los módulos o las *arrow functions* (funciones flecha).

SQL: *Structured Query Language* es una forma de comunicarse con una base de datos relacional que permite realizar consultas para insertar, modificar o eliminar datos de la base de datos. Usando la sintaxis de SQL se pueden construir instrucciones que extraigan información de acuerdo a un criterio.

MySQL: desarrollado por Oracle Corporation, es el sistema de gestión de bases de datos SQL de código abierto más popular. Permite almacenar, acceder y procesar datos de forma estructurada y flexible utilizando lenguaje SQL y tablas relacionales. Además, MySQL ofrece una opción de licencia comercial aparte de su uso gratuito bajo licencia GPL.

Prisma: es un *ORM* (Mapeo Objeto-Relacional) para Node.js y TypeScript. Prisma permite conectar la aplicación con una base de datos y realizar consultas de forma sencilla. Además, Prisma permite utilizar diferentes bases de datos como MySQL, PostgreSQL o SQLite.

TailwindCSS: Tailwind CSS es un framework de CSS que prioriza la utilidad. A diferencia de otros frameworks de CSS como Bootstrap, Tailwind no ofrece una serie de componentes ya defininidos. En su lugar, Tailwind opera a un nivel más bajo y proporciona un conjunto de clases de utilidad para la estructura y el estilo. Esto permite crear rápidamente diseños personalizados utilizando las clases.

8.2 Estructura del frontend

El frontend está desarrollado usando Next.js y React. Como se puede observar en la figura 8.1, el frontend está organizado en cinco secciones principales: **app**, **components**, **hooks**, **services** y **lib**.

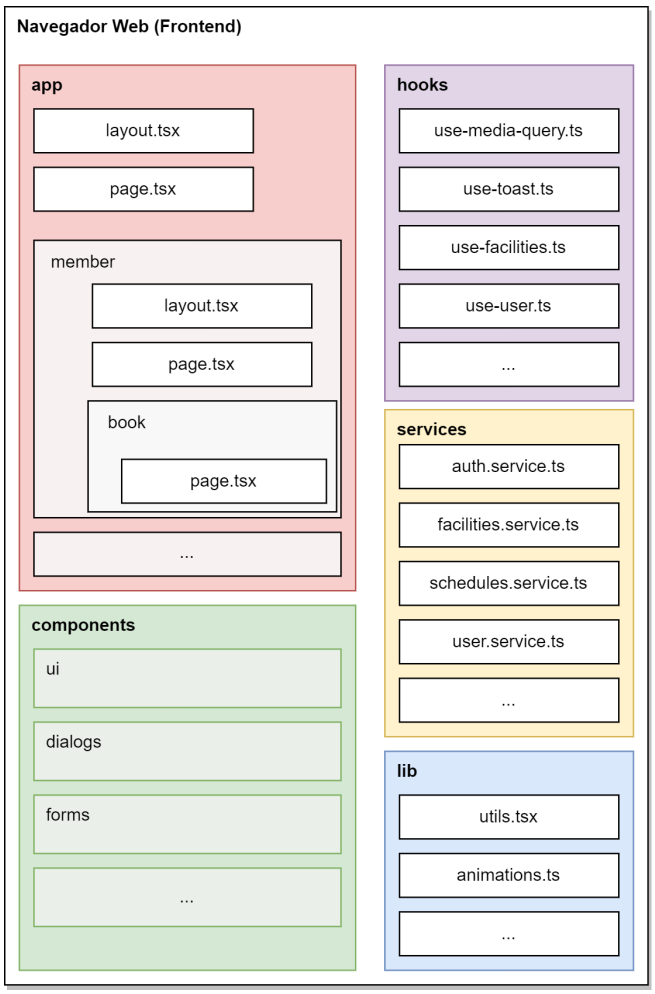

Figura 8.1: Arquitectura del frontend

app: el apartado app actúa como enrutador: las rutas de la aplicación se definen como carpetas y sub carpetas de este apartado. Dentro de cada carpeta existe un archivo `page.tsx` para indicar que es una página de la aplicación y establecer el contenido. Además, cada carpeta puede contener opcionalmente un archivo `layout.tsx` en el que se establece la interfaz de usuario común a todas las páginas de la carpeta y sub carpetas en la que se encuentra. En la figura 8.2 se explica de forma visual el funcionamiento de los archivos `page.tsx` y `layout.tsx`.

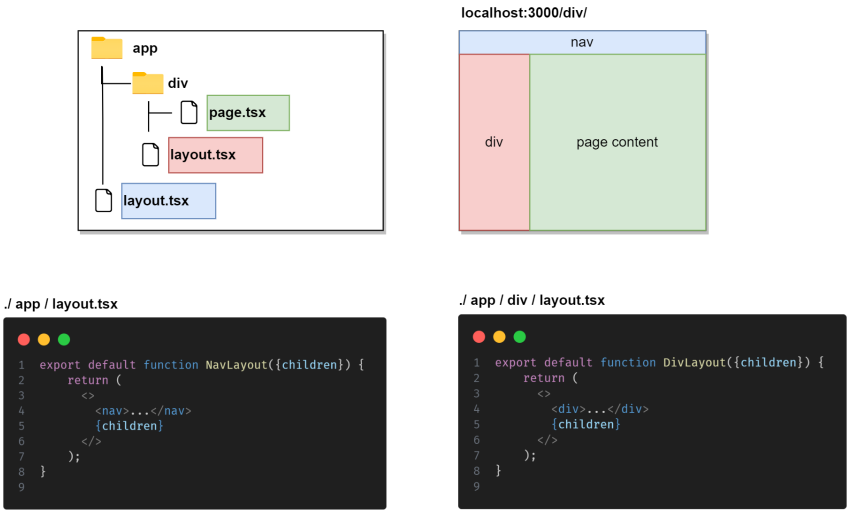

Figura 8.2: Funcionamiento de los archivos `page.tsx` y `layout.tsx` del apartado app

components: el directorio de componentes sirve para almacenar y organizar los componentes reutilizables de la aplicación. Entre los componentes reutilizables se encuentran los componentes de la interfaz de usuario como los botones, los formularios o los iconos. Además, también se encuentran los componentes utilizados en casos más específicos como la visualización de la información relacionada con la lista de instalaciones o de deportes.

hooks: el directorio de hooks sirve para almacenar y organizar los hooks personalizados de la aplicación. Los hooks son funciones que permiten añadir funcionalidad a los componentes de React. Entre los hooks personalizados se encuentran los hooks para gestionar el estado de la aplicación, los hooks para realizar peticiones a la API o los hooks para gestionar los formularios.

services: el directorio de servicios sirve para almacenar y organizar los servicios de la aplicación. Los servicios son funciones que permiten realizar peticiones a la API. Entre los servicios se encuentran los servicios para gestionar las instalaciones, los servicios para gestionar los deportes o los servicios para gestionar los usuarios.

lib: el directorio de lib sirve para almacenar y organizar las librerías adicionales usadas en la aplicación. Entre las librerías adicionales se encuentran las librerías de ayuda a la mezcla de estilos CSS, y otras que establecen parámetros para las animaciones de la aplicación.

8.3 Estructura del backend

El backend está desarrollado usando Node.js, TypeScript y Express. La figura 8.3 muestra la estructura del backend. El backend está organizado en siete secciones principales: **app.ts**, **routes**, **controllers**, **middlewares**, **services**, **lib** y **utils**.

app.ts: el archivo `app.ts` es el archivo principal de la aplicación. En este archivo se establece la configuración de la aplicación y las rutas de la aplicación.

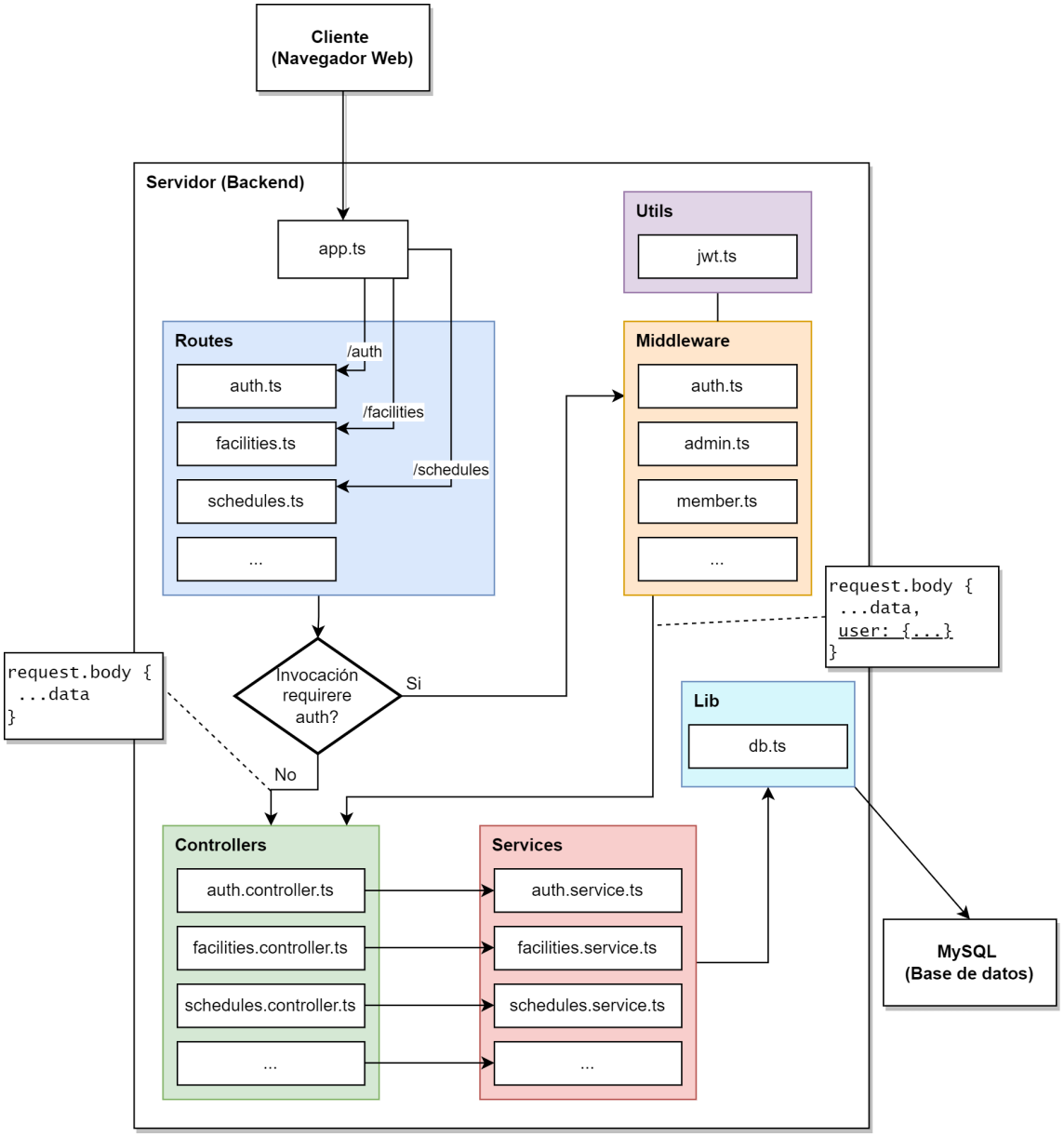

Figura 8.3: Arquitectura del backend

routes: el apartado routes actúa como enrutador: las rutas de la aplicación se definen como archivos de este apartado. Dentro de cada archivo se definen las rutas de la aplicación y se establece la función que se ejecuta cuando se accede a la ruta. En la figura 8.4 se incluye un fragmento de código de un archivo de enrutador de Express.

controllers: el apartado controllers sirve para almacenar y organizar los controladores de la aplicación. Los controladores son funciones que se ejecutan cuando se accede a una ruta de la aplicación. Manejan la información de la petición y llaman al método correspondiente del servicio.

middlewares: el apartado middlewares sirve para almacenar y organizar los middlewares de la aplicación. Los middlewares son funciones que se ejecutan antes de que se ejecute el controlador de una ruta. Entre los middlewares se encuentran los middlewares para comprobar si el usuario está autenticado o los middlewares para comprobar si el usuario tiene permisos para acceder a la ruta.

```
1  const router: Router = express.Router();
2
3  router.post("/", auth, createSport);
4
5  router.get("/", auth, getAll);
6
7  router.get("/full", auth, getAllFull);
8
9  export { router as sportsRouter };
10
```

Figura 8.4: Ejemplo de un archivo de enrutador de Express

services: el apartado services sirve para almacenar y organizar los servicios de la aplicación. Los servicios son funciones que permiten realizar peticiones a la base de datos. Entre los servicios se encuentran los servicios para gestionar las instalaciones, los servicios para gestionar los deportes o los servicios para gestionar los usuarios.

lib: el directorio de *lib* sirve para almacenar y organizar las librerías adicionales usadas en la aplicación. Entre las librerías adicionales se encuentra la librería de acceso a la base de datos.

utils: el directorio de *utils* sirve para almacenar y organizar las utilidades de la aplicación. Entre las utilidades se encuentran las utilidades para gestionar la autenticación de los usuarios.

Para el autenticado de la aplicación se usa JWT *(JSON Web Token)*. JWT es un estándar abierto basado en JSON para crear tokens de acceso que permiten la transmisión segura de información entre las partes como un objeto JSON. En el caso de Hércules, el token contiene la información del usuario autenticado y se envía en la cabecera de las peticiones a la API. El token se guarda en `localStorage` del navegador y se elimina cuando el usuario cierra sesión. Además, el token tiene una fecha de expiración de 24 horas, por lo que el usuario tiene que volver a iniciar sesión cada 24 horas.

8.4 Patrones de diseño

En la aplicación se han aplicado varios patrones de diseño. En esta sección se explicarán los patrones de diseño más importantes que se han aplicado en la aplicación.

8.4.1. Observador

El patrón de diseño observador es un patrón de diseño de comportamiento que permite a un objeto publicar cambios en su estado. Otros objetos se suscriben a ser notificados cuando cambia el estado. En el caso de Hércules, se ha usado el patrón de diseño observador para los *hooks*[3] de React. En la figura 8.5 se pue-

[3]Los **hooks** te permiten usar diferentes funciones de React desde los componentes.

de ver un diagrama de clases del patrón de diseño observador en Hércules usado en el *hook* `useActivities`. En este caso, los componentes `ActivitiesList` y `ActivityDetail` se suscriben a los cambios en el estado de las actividades. Cuando se produce un cambio en la lista de actividades, los componentes `ActivitiesList` y `ActivityDetail` se actualizan automáticamente.

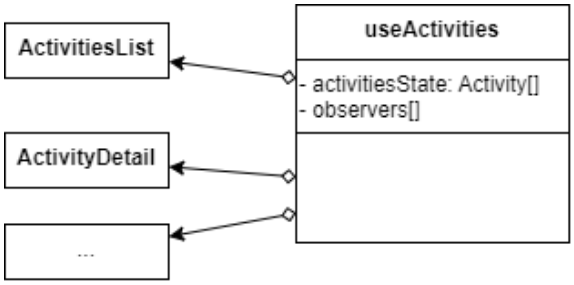

Figura 8.5: Diagrama de clases del patrón de diseño observador en Hércules

En la figura 8.6 se puede ver como el componente `ActivitiesList` accede a la lista de actividades mediante `useActivities`. En la figura 8.7 se puede ver como el componente `ActivitiesList` actualiza la lista de actividades mediante `setActivities`. Finalmente, en la figura 8.8 se puede ver como el componente `ActivitiesList` se suscribe y se desuscribe a los cambios en la lista de actividades mediante las propiedades del hook `useEffect` que viene incluido con React[4]. En este caso, el componente `ActivitiesList` se suscribe a los cambios en la lista de actividades cuando se monta el componente y se desuscribe cuando se desmonta el componente.

```
1  export default function ActivitiesList(props: ActivitiesListProps) {
2    const { activities } = useActivities();
3    ...
4
5    return (
6      ...
7    );
8  }
```

Figura 8.6: Código de `ActivitiesList` que accede a la lista de actividades usando el hook `useActivities`

[4]El hook **useEffect** permite realizar efectos secundarios en componentes funcionales. Los efectos secundarios son operaciones asíncronas como llamadas a API, actualizaciones de estado, etc. El hook acepta una función y una lista de dependencias. La función se ejecutará cada vez que haya un cambio en la lista de dependencias. Si la lista de dependencias está vacía, la función solo se ejecutará una vez. Finalmente, el hook permite devolver una función que se ejecutará cuando el componente se desmonte.

```
let activities: Activity[] = [];
let observers: React.Dispatch<React.SetStateAction<Activity[]>>[] = [];

export const setActivities = (newActivities: Activity[]) => {
  activities = newActivities.slice();
  observers.forEach((update) => {
    update(activities);
  });
};
```

Figura 8.7: Código de `useActivities` donde se muestra donde se mantiene el estado, la lista de observadores y como se notifica a los observadores

```
export default function ActivitiesList(props: ActivitiesListProps) {
  const { activities } = useActivities();
  ...

  return (
    ...
  );
}
```

Figura 8.8: Código de `useActivities` donde se muestra donde se mantiene cuando se añade y elimina un observador

8.4.2. Singleton

El patrón de diseño singleton es un patrón de diseño creacional que garantiza que solo exista una instancia de una clase y proporciona un punto de acceso global a ella. En el caso de Hércules, se ha usado el patrón de diseño singleton para la conexión de Prisma con la base de datos. En el *backend* solo debe existir una conexión con la base de datos. En la figura 8.9 se puede ver un diagrama de clases del patrón de diseño singleton en Hércules. En este caso, la clase `Database` es la clase singleton ya que tiene un método estático `getInstance` que devuelve la instancia de la clase `Database`. Además, `Database` tiene un constructor privado. Esto es necesario para que no se pueda crear una instancia de la clase `Database` fuera de la clase. Finalmente, se puede ver como la clase `Database` tiene una propiedad estática `instance` que es la instancia de la clase `Database`. Ya que el objeto del que se quiere tener una sola instancia es de tipo `PrismaClient` y es un paquete de terceros, la clase `Database` envuelve el objeto `PrismaClient` por lo que se ha añadido un método `getPrisma` que devuelve el objeto `PrismaClient`. En la figura 8.10 se puede ver la implementación del patrón de diseño singleton en Hércules.

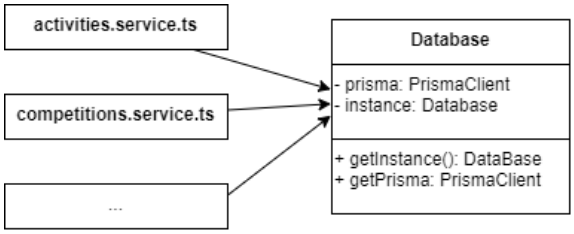

Figura 8.9: Diagrama de clases del patrón de diseño singleton en Hércules

```
1  import { PrismaClient } from "@prisma/client";
2
3  class Database {
4    private static instance: Database;
5    private prisma: PrismaClient;
6
7    private constructor() {
8      this.prisma = new PrismaClient();
9    }
10
11   public static getInstance(): Database {
12     if (!Database.instance) {
13       Database.instance = new Database();
14     }
15
16     return Database.instance;
17   }
18
19   public getPrisma(): PrismaClient {
20     return this.prisma;
21   }
22 }
23
24 export default Database;
```

Figura 8.10: Implementación del patrón singleton

CAPÍTULO 9
Pruebas

9.1 Pruebas de integración

Las pruebas de integración se realizan para comprobar que los diferentes componentes de la aplicación funcionan correctamente cuando se integran. En el caso de Hércules, se han realizado pruebas de integración para comprobar que el backend funciona correctamente con la base de datos. Para ello, se ha usado *vitest*, un *framework* de pruebas que usa *Vite*[1] para ejecutar las pruebas de forma muy rápida. Vitest también funciona con TypeScript y tiene compatibilidad con las funciones de *Jest*[2], otro *framework* de pruebas muy popular, por lo que si se ha usado Jest con anterioridad no es necesario aprender una nueva sintaxis.

Para poder hacer las pruebas de integración sin afectar a los datos de producción, se ha usado una base de datos de prueba. Para ello, se ha creado un contenedor de *Docker*[3] con una base de datos MySQL. En la figura 9.1 se puede ver el fichero de configuración del contenedor de Docker para la base de datos. En este se establece una imagen base que contiene el sistema operativo y la base de datos. Despúes se introduce la información de la base de datos como el nombre de la base de datos, el usuario y la contraseña. Por último, se establece el puerto por el que se puede acceder a la base de datos desde el sistema operativo del anfitrión, el puerto 3310 en este caso.

Para que cada prueba se ejecute en un estado limpio de base de datos, se crea un fichero de configuración de vitest que elimina todos los datos de la base de datos antes de ejecutar cada prueba. En la figura 9.2 se puede ver el fichero de configuración de vitest. En este se establece el script de limpieza de la base de datos que se ejecuta antes de cada prueba. En la figura 9.3 se puede ver el script de limpieza de la base de datos. En este se eliminan todas las tablas de la base de datos.

[1] **Vite** es una herramienta de compilación que tiene como objetivo proporcionar una experiencia de desarrollo más rápida y ágil para proyectos web modernos [17].

[2] **Jest** es el *framework* más popular del ecosistema JavaScript. Es usado por grandes empresas como Twitter, Instagram, Pinterest y Airbnb.

[3] **Docker** es una plataforma de código abierto que permite a los desarrolladores crear, desplegar, ejecutar y gestionar contenedores, que son componentes estandarizados y ejecutables que combinan el código fuente de aplicación con las dependencias y las bibliotecas del sistema operativo (SO) necesarias para ejecutar dicho código en cualquier entorno [20].

```yaml
version: "3.3"
services:
  db:
    image: mysql:latest
    restart: always
    environment:
      MYSQL_DATABASE: "hercules"
      # So you don't have to use root, but you can if you like
      MYSQL_USER: "hercules"
      # You can use whatever password you like
      MYSQL_PASSWORD: "contraseña"
      # Password for root access
      MYSQL_ROOT_PASSWORD: "contraseña"
    ports:
      # <Port exposed> : < MySQL Port running inside container>
      - "3310:3306"
    expose:
      # Opens port 3306 on the container
      - "3306"
      # Where our data will be persisted
    volumes:
      - my-db:/var/lib/mysql
# Names our volume
volumes:
  my-db:

```

Figura 9.1: Fichero de configuración del contenedor de Docker con una base de datos MySQL

```ts
import { defineConfig } from "vitest/config";

export default defineConfig({
  test: {
    include: ["src/tests/**/*.test.ts"],
    threads: false,
    setupFiles: ["src/tests/helpers/setup.ts"],
  },
});

```

Figura 9.2: Fichero de configuración de vitest

```
import prisma from "../../lib/db";

export default async () => {
  await prisma.$transaction([
    prisma.activity.deleteMany(),
    prisma.activitySchedule.deleteMany(),
    prisma.competition.deleteMany(),
    prisma.event.deleteMany(),
    prisma.facility.deleteMany(),
    prisma.facilitySchedule.deleteMany(),
    prisma.match.deleteMany(),
    prisma.racketReport.deleteMany(),
    prisma.refereeReport.deleteMany(),
    prisma.round.deleteMany(),
    prisma.set.deleteMany(),
    prisma.sport.deleteMany(),
    prisma.team.deleteMany(),
    prisma.user.deleteMany(),
  ]);
};

```

Figura 9.3: Script de limpieza de base de datos

Una vez configurado el entorno de pruebas, se pueden crear las pruebas de integración. Las pruebas se escriben en ficheros con extensión `.test.ts` y se guardan en la carpeta *tests* del proyecto. En los ficheros de pruebas se utiliza *describe* para agrupar las pruebas y *it* para definir cada prueba. Las pruebas de integración de Hercules están organizadas por rutas y operaciones de las rutas. Por ejemplo, todas las pruebas de la operación `[POST] /auth/register` se encuentran agrupadas y se enuenctran dentro de una agrupación más grande relacionada con todas las operaciones de la ruta `/auth`. También se utiliza *beforeEach* para definir las instrucciones que se deben ejecutar antes de cada prueba. Dentro de cada prueba se utiliza *expects* para comprobar que el resultado de la prueba es el esperado.

En la figura 9.4 se puede ver un ejemplo de una prueba de integración. Primero se define el nombre de la prueba que explica lo que debe hacer. Después se hace un *request* a la ruta `[POST] /sport` con los datos de la prueba. En este caso, se envía un nombre de deporte y una descripción. Después se comprueba que el código de estado de la respuesta es 201 y que el cuerpo de la respuesta contiene el nombre, identificador, y la descripción del deporte. Por último, se comprueba que el deporte se ha guardado en la base de datos.

```ts
it("should respond with a `201` status code and the sport must be created", async () => {
  const { status, body } = await request(app)
    .post("/sports")
    .set("Authorization", await createUserToken())
    .send({
      name: exampleSportData.futbol.name,
      description: exampleSportData.futbol.description,
    });
  expect(status).toBe(201);
  expect(body.message).toBe("Deporte creado correctamente");
  expect(body).toHaveProperty("data");
  expect(body.data.id).toMatch(
    /^[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}$/i
  );
  expect(body.data.name).toBe(exampleSportData.futbol.name);
  expect(body.data.description).toBe(exampleSportData.futbol.description);
  const sport = await prisma.sport.findUnique({
    where: {
      id: body.data.id,
    },
  });
  expect(sport).not.toBeNull();
  expect(sport?.name).toBe(exampleSportData.futbol.name);
  expect(sport?.description).toBe(exampleSportData.futbol.description);
});
```

Figura 9.4: Prueba de integración que comprueba el funcionamiento correcto de `[POST] /sport`

En la figura 9.5 se puede ver un ejemplo de una prueba de integración que comprueba que la ruta `[POST] /sport` devuelve un error 401 cuando el usuario no está autenticado. En este caso, se hace un *request* a la ruta `[POST] /sport` con los datos de la prueba. En este caso, se envía un nombre de deporte y una descripción. Después se comprueba que el código de estado de la respuesta es 401 y que el cuerpo de la respuesta contiene un mensaje de error. Por último, se comprueba que el deporte no se ha guardado en la base de datos.

Otro ejemplo de prueba de integración es la prueba que comprueba que la ruta `[GET] /sport` devuelve un error 403 cuando ya existe un deporte con ese nombre, ya que el nombre de deporte está definido como único en la base de datos. En la figura 9.6 se puede ver el código de la prueba. En este caso, se hace un *request* a la ruta `[POST] /sport` con los datos de la prueba. Se envía un nombre de deporte y una descripción. Después se comprueba que el código de estado de la respuesta es 403 y que el cuerpo de la respuesta contiene un mensaje de error.

```
it("should respond with a `401` status code if user is not authenticated", async () => {
  const { status, body } = await request(app).post("/sports").send({
    name: exampleSportData.futbol.name,
    description: exampleSportData.futbol.description,
  });
  expect(status).toBe(401);
  expect(body.message).toBe("Usuario no autenticado");
  expect(body).not.toHaveProperty("data");
  const sport = await prisma.sport.findUnique({
    where: {
      name: exampleSportData.futbol.name,
    },
  });
  expect(sport).toBeNull();
});
```

Figura 9.5: Prueba de integración que comprueba que `[POST] /sport` devuelve un error 401 cuando el usuario no está autenticado

```
it("should respond with a `403` status code if a sport with that name already exists", async () => {
  await prisma.sport.create({
    data: exampleSportData.futbol,
  });
  const { status, body } = await request(app)
    .post("/sports")
    .set("Authorization", await createUserToken())
    .send({
      name: exampleSportData.futbol.name,
      description: exampleSportData.futbol.description,
    });
  expect(status).toBe(403);
  expect(body.message).toBe("Ya existe un deporte con ese nombre");
  expect(body).not.toHaveProperty("data");
});
```

Figura 9.6: Prueba de integración que comprueba que `[POST] /sport` devuelve un error 403 cuando ya existe un deporte con ese nombre

Finalmente, se ejecutan los tests con el comando `npm run test`. Vitest ofrece una aplicación de consola y una interfaz web más visual para ejecutar las pruebas. En la figura 9.7 se puede ver la aplicación de consola de vitest ejecutando las pruebas de integración de Hercules. En la figura 9.8 se puede ver la misma información en la interfaz web.

```
stdout | src/tests/auth/auth.test.ts > /auth > [POST] /auth/login > should respond with a `400` status code and error message if password is incorrect
POST /auth/login

stdout | src/tests/auth/auth.test.ts > /auth > [POST] /auth/login > should respond with a `400` status code and error message if password is incorrect
<empty line>
stderr | src/tests/auth/auth.test.ts > /auth > [POST] /auth/login > should respond with a `400` status code and error message if password is incorrect
Correo electrónico o contraseña incorrecto

 ✓ src/tests/facilities/facilities.test.ts (17) 1959ms
 ✓ src/tests/competitions/competitions.test.ts (17) 1792ms
 ✓ src/tests/activities-schedules/activities-schedules.test.ts (17) 1743ms
 ✓ src/tests/instructors/instructors.test.ts (17) 1818ms
 ✓ src/tests/reports/reports.test.ts (17) 1857ms
 ✓ src/tests/sports/sports.test.ts (17) 1766ms
 ✓ src/tests/schedules/schedules.test.ts (17) 1806ms
 ✓ src/tests/matches/matches.test.ts (17) 1831ms
 ✓ src/tests/activities/activities.test.ts (17) 1800ms
 ✓ src/tests/auth/auth.test.ts (8) 1232ms
```

Figura 9.7: Aplicación de consola de vitest ejecutando las pruebas de integración de Hercules

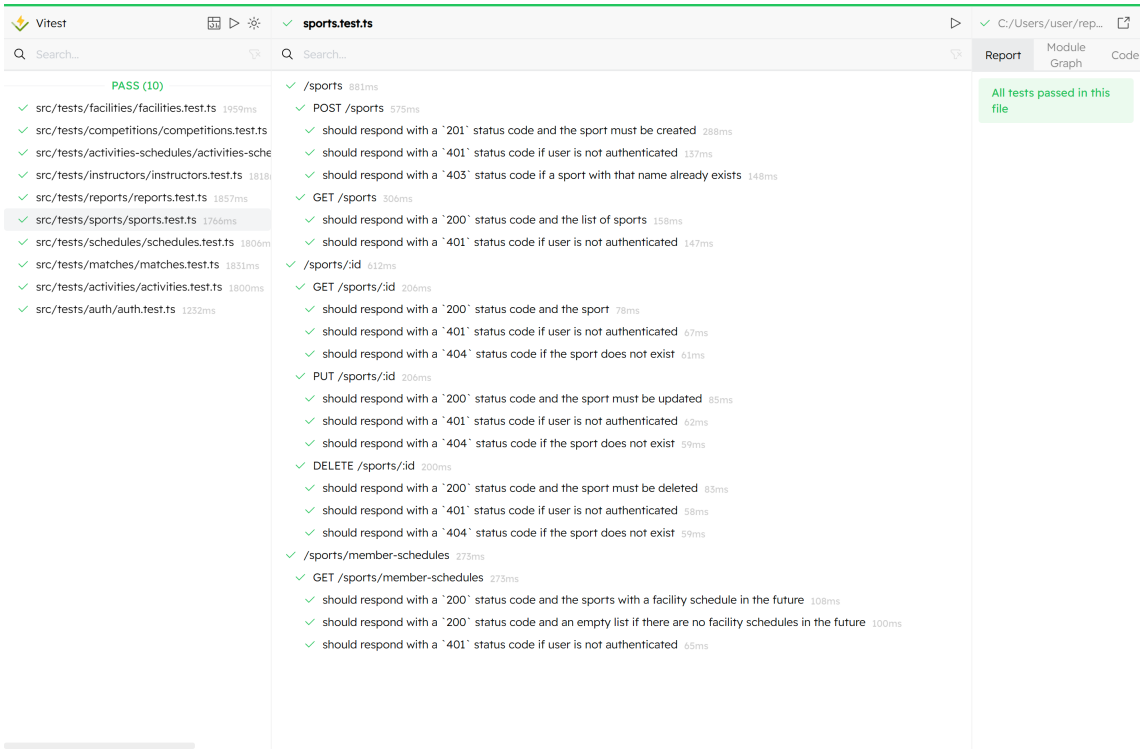

Figura 9.8: Interfaz web de vitest ejecutando las pruebas de integración de Hercules

9.2 Analizadores estáticos

La calidad del software es un aspecto muy importante en el desarrollo de software. Esta se puede medir de muchas formas, pero una de las formas más comunes es mediante el análisis estático del código. El análisis estático del código es un proceso que analiza el código fuente de un programa sin ejecutarlo y se puede usar para encontrar errores, vulnerabilidades, y otros problemas en el código fuente. En el caso de Hercules, se ha usado el analizador estático *ESLint*. ESLint es una herramienta de análisis estático de código para JavaScript y TypeScript. Su objetivo principal es ayudar a los desarrolladores a detectar y corregir errores, mantener la consistencia del código y seguir las mejores prácticas de programación. ESLint analiza el código fuente de TypeScript en busca de problemas y emite mensajes de advertencia o error cuando se encuentran violaciones a las reglas definidas.

Para ejecutar ESLint se usa el comando `npm run lint`. En la figura 9.9 se puede ver la salida de ESLint en la terminal. Se puede ver como detecta variables sin utilizar y variables de tipo any. Aunque no hay problema al usarlo en la terminal, se ha integrado ESLint en el editor de código Visual Studio Code para que muestre los errores y advertencias en tiempo real. Para ello, se ha instalado la extensión *ESLint* de Microsoft.

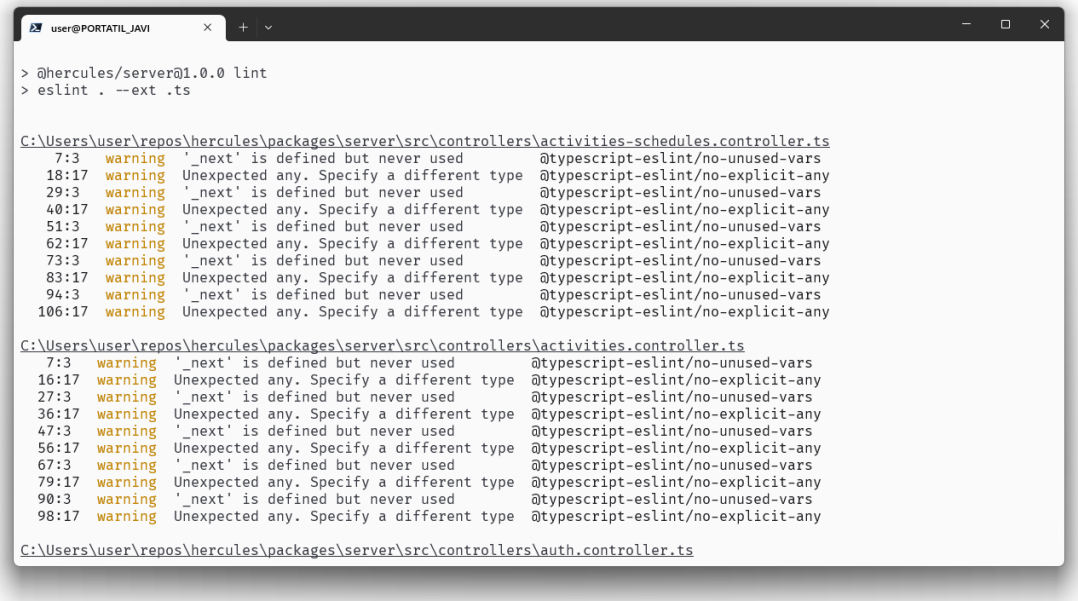

Figura 9.9: Salida de ESLint en la terminal

CAPÍTULO 10
Aplicación Final

10.1 Login

El primer paso para acceder a la aplicación es iniciar sesión. Para ello, el usuario debe introducir su correo electrónico y su contraseña. En la figura 10.1 se muestra la pantalla de inicio de sesión de la aplicación final. En la figura 10.2 se muestra la pantalla de inicio de sesión con errores de validación, en caso de que el usuario introduzca un correo electrónico o una contraseña con un formato incorrecto.

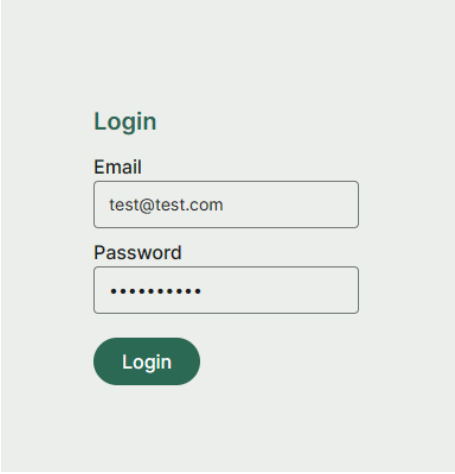

Figura 10.1: Login

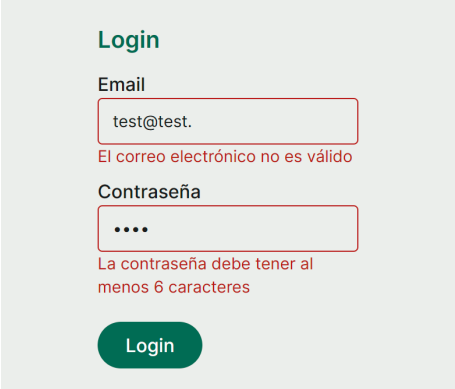

Figura 10.2: Login con errores de validación

10.2 Vista de gestión

Primero se va a ver las funcinalidades del gestor, que es el usuario con más permisos. Para acceder a la vista de gestión se debe iniciar sesión en la aplicación con un usuario con el rol de gestor.

La vista de gestor tiene una barra lateral de navegación con cinco elementos: Deportes, Instalaciones, Actividades, Competiciones y Cerrar sesión. A continuación se va a explicar cada una de las funcionalidades de la vista de gestión.

10.2.1. Gestionar deportes

El primer elemento de la barra de navegación es Deportes. Al hacer click en este elemento se muestra una lista con todos los deportes.

Al hacer click en un deporte de la lista se puede ver en la derecha una vista con más detalles sobre el deporte. En la figura 10.3 se puede ver la vista de deportes, con el deporte "Boxeo" seleccionado. En esta vista se puede ver el nombre y la descripción de cada deporte.

Debajo de la lista hay un botón para crear un nuevo deporte. Al hacer click en el botón se muestra un formulario para crear un nuevo deporte. En la figura 10.4 se puede ver el formulario para crear un nuevo deporte.

En la parte superior derecha de la vista del deporte seleccionado hay un botón para editar ese deporte y otro para eliminarlo. Al hacer click en el botón de editar muestra un formulario para editar el deporte. En la figura 10.5 se puede ver el formulario para editar un deporte y al hacer click en el botón de eliminar se muestra un mensaje para confirmar la eliminación del deporte. En la figura 10.6 se puede ver el mensaje de confirmación. Si el usuario hace click en el botón de confirmar, se elimina el deporte. Si el usuario hace click en el botón de cancelar, se cierra el mensaje de confirmación.

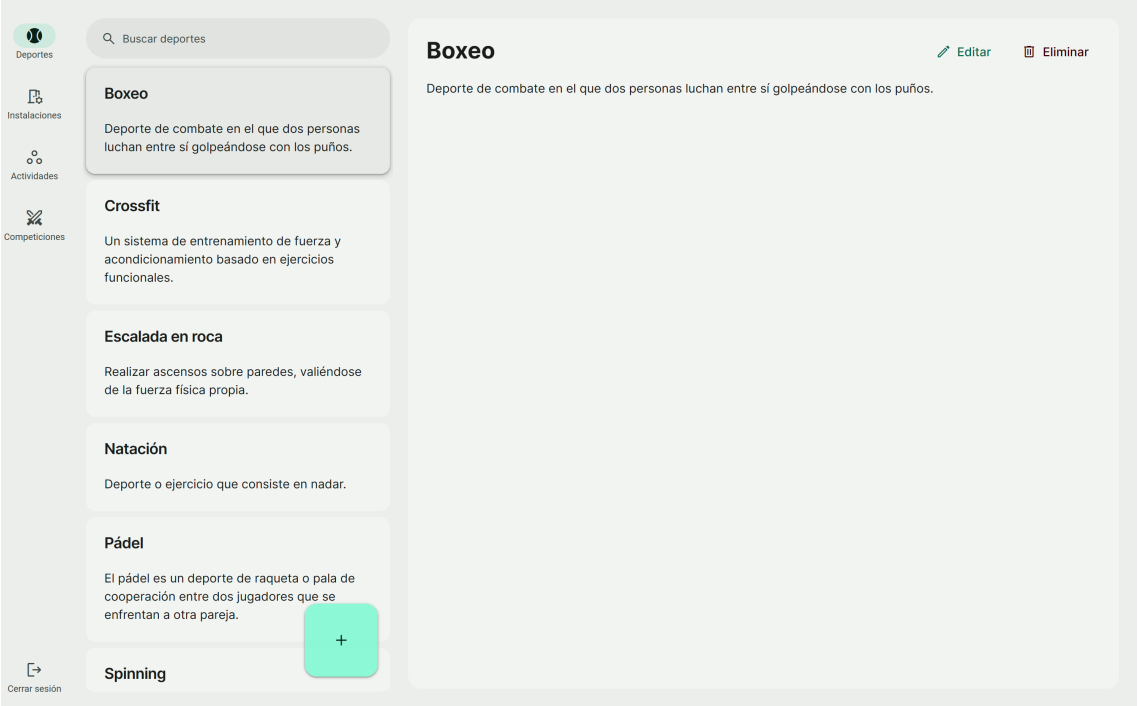

Figura 10.3: Vista de deportes

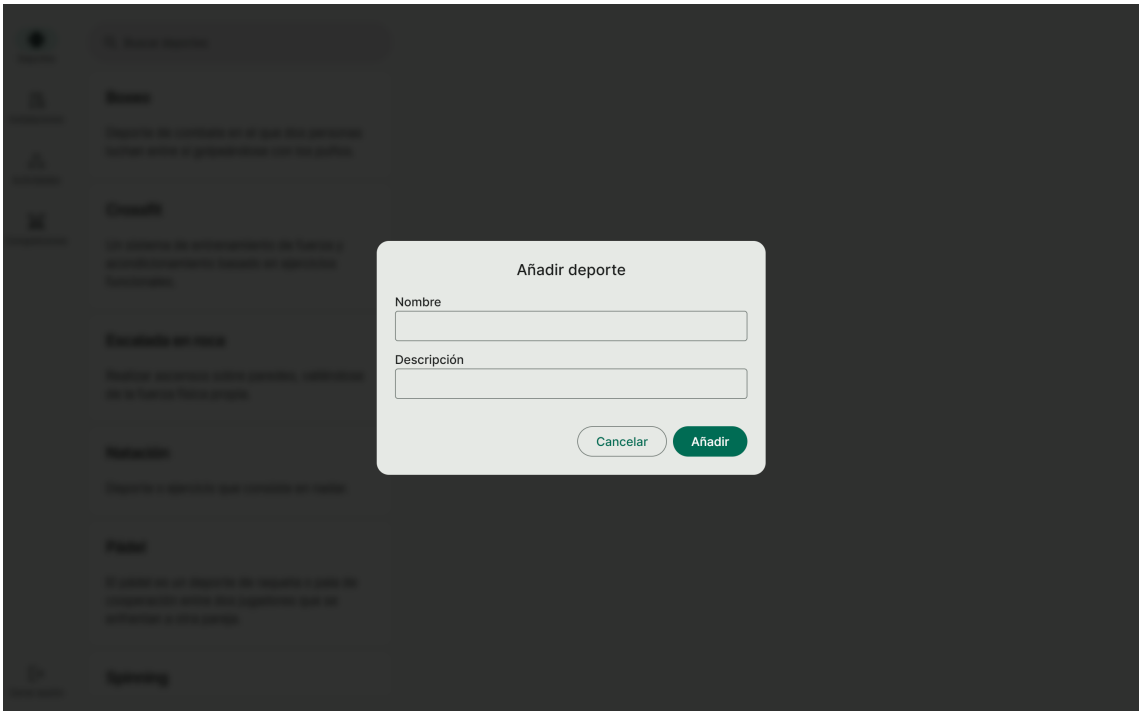

Figura 10.4: Creación de deportes

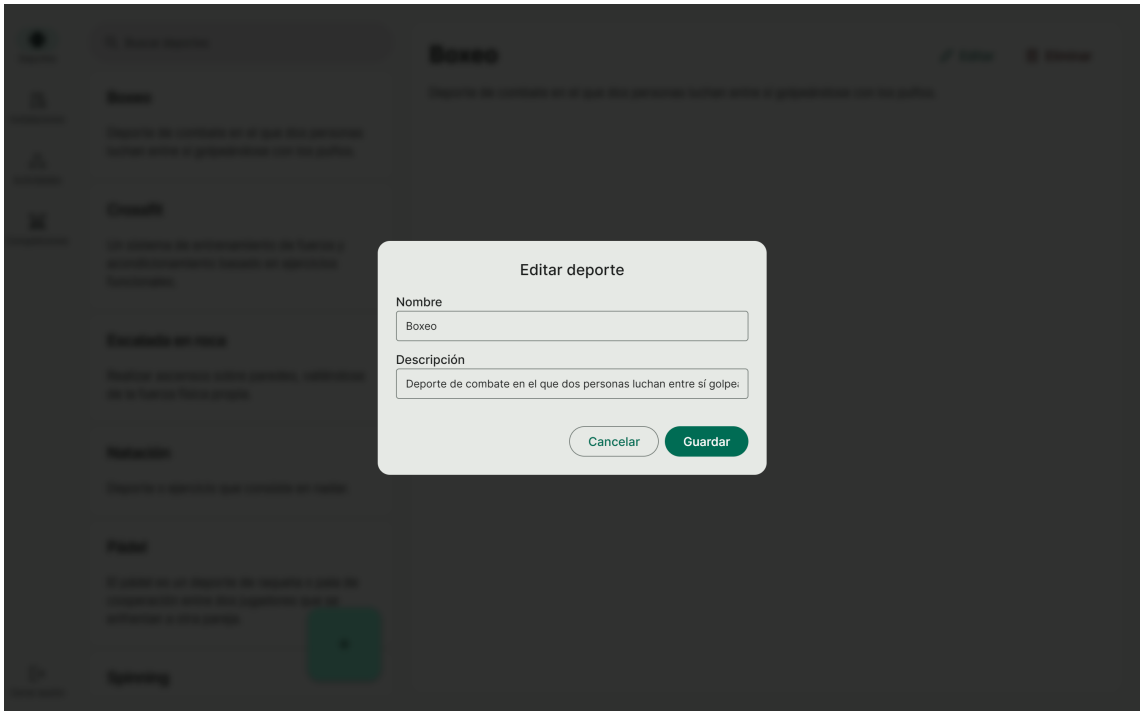

Figura 10.5: Modificación de deportes

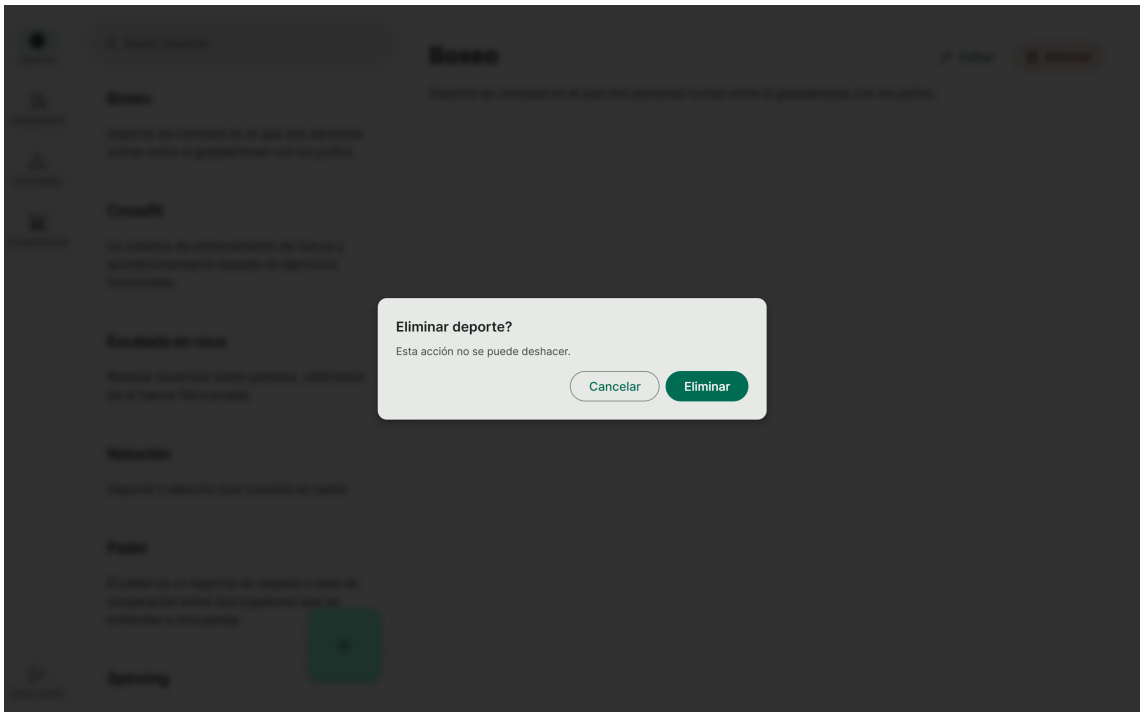

Figura 10.6: Eliminación de deportes

10.2.2. Gestionar instalaciones

Se puede acceder a la pantalla de gestión de instalaciones haciendo click a el botón de instalaciones de la barra de navegación. Esta pantalla es similar a las otras pantallas del gestor, con una vista de lista y otra de detalle para la instalación seleccionada.

En la figura 10.7 se muestra la vista principal de las instalaciones sin ninguna instalación seleccionada. Al final de la lista hay un botón para añadir una instalación. Al hacer click se muestra el formulario de creación de instalación de la figura 10.8, donde se pide el nombre, el deporte, la dirección y el número mínimo de días que debe pasar para que se permita a los socios reservarla. El formulario relleno con datos de ejemplo se puede ver en la figura 10.9.

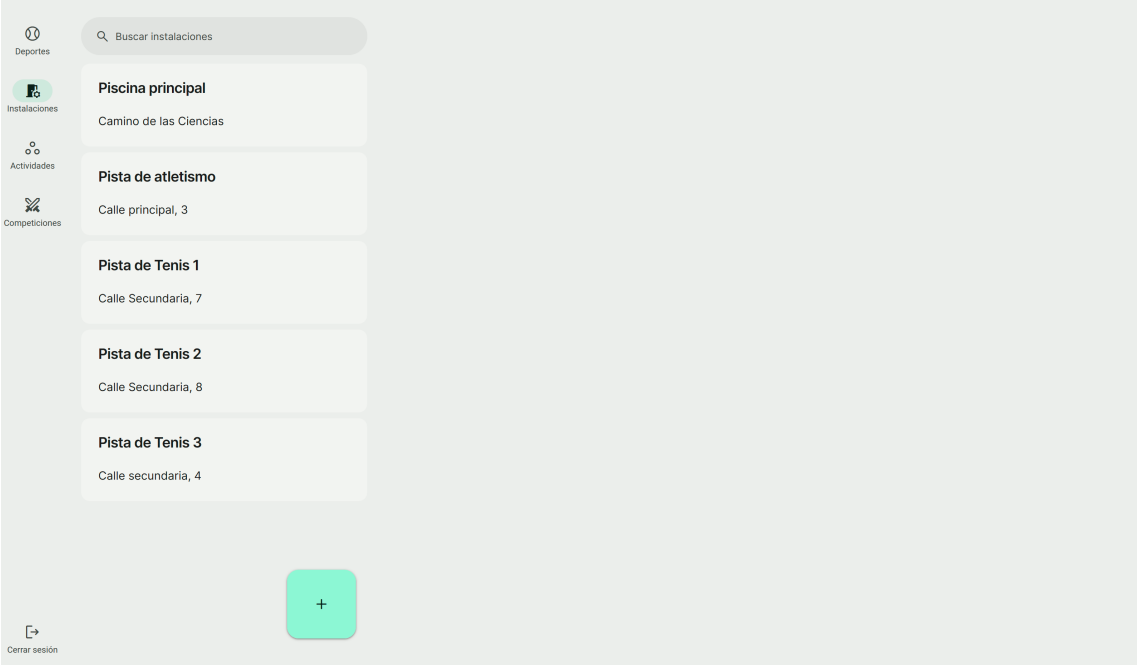

Figura 10.7: Lista de instalaciones sin seleccionar

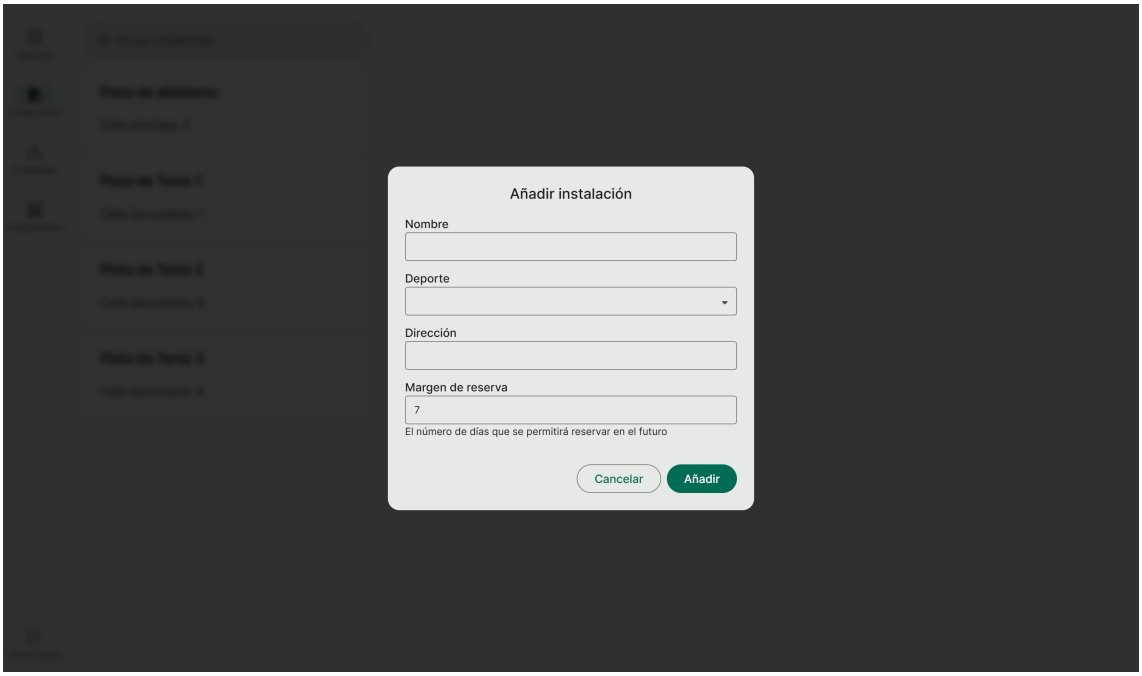

Figura 10.8: Creación de instalaciones

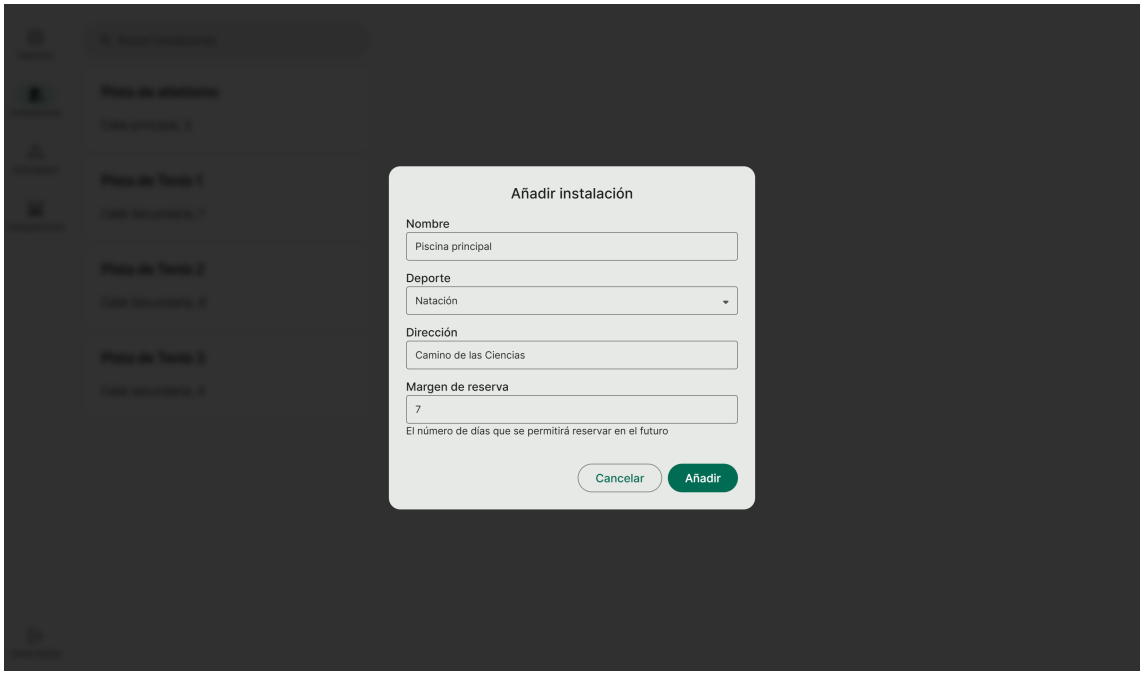

Figura 10.9: Creación de instalaciones con datos de ejemplo

Al seleccionar una instalación de la lista se muestra la vista de detalle de la instalación seleccionada. En la figura 10.10 se puede ver la vista de detalle de la instalación "Pista de Tenis 1". Esta vista permite seleccionar una fecha y editar los horarios de esa fecha. Una vez seleccionada la fecha se escoge la opción "Añadir horario" para añadir un horario para la instalación. En la figura 10.11 se puede ver el formulario para añadir un horario. En el formulario se puede escoger la hora de inicio, la hora de fin y si está disponible para reservar. En la figura 10.12 se puede ver como queda la instalación con varios horarios. Esta vista incluye para cada horario las siguiente información: Hora de inicio, hora de fin, si está disponible para reservar y el estado. El estado puede ser "disponible", "reservado", "actividad" o "partido", dependiendo de si está disponible, reservado por un socio, reservado para una actividad o reservado para un partido. También al seleccionar un horario se puede editar los datos de ese horario. En la figura 10.13 se puede ver el formulario para editar un horario. En el formulario se puede cambiar la hora de inicio, la hora de fin y si está disponible para reservar.

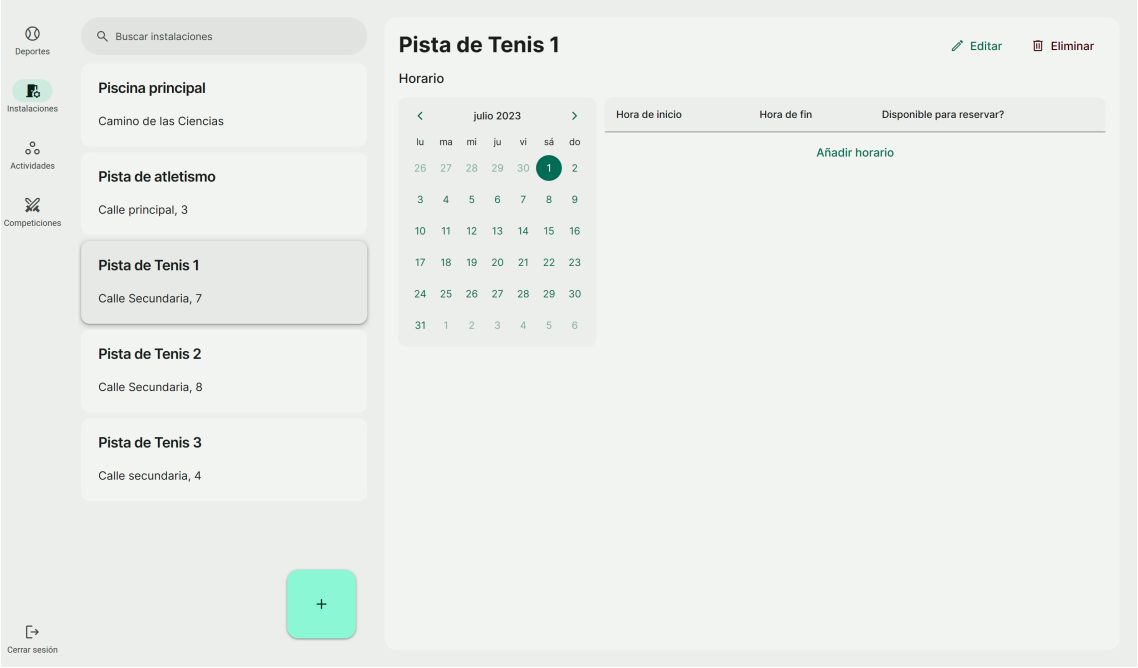

Figura 10.10: Vista de detalle de instalación

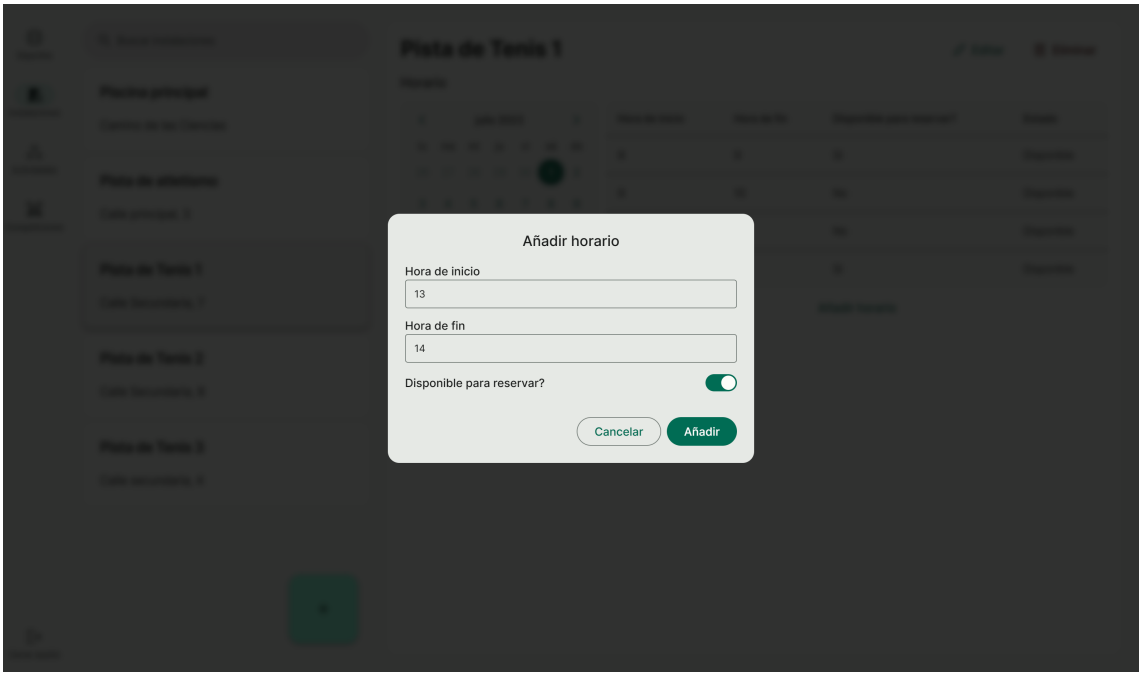

Figura 10.11: Creación de horarios de instalaciones

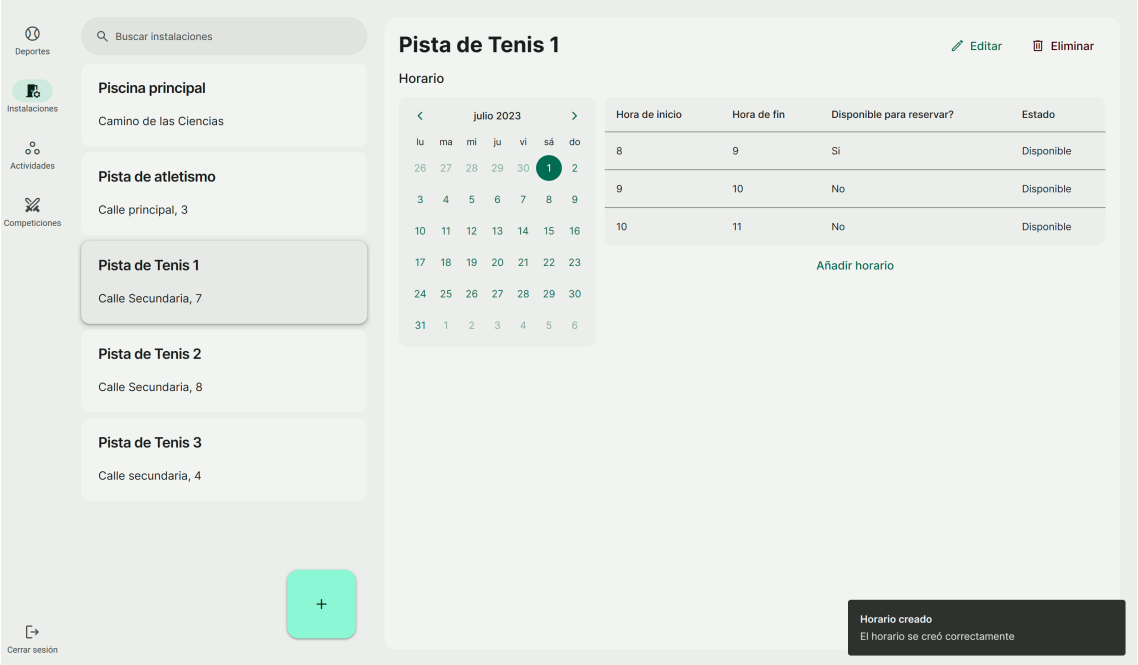

Figura 10.12: Vista de detalle de instalación con horarios

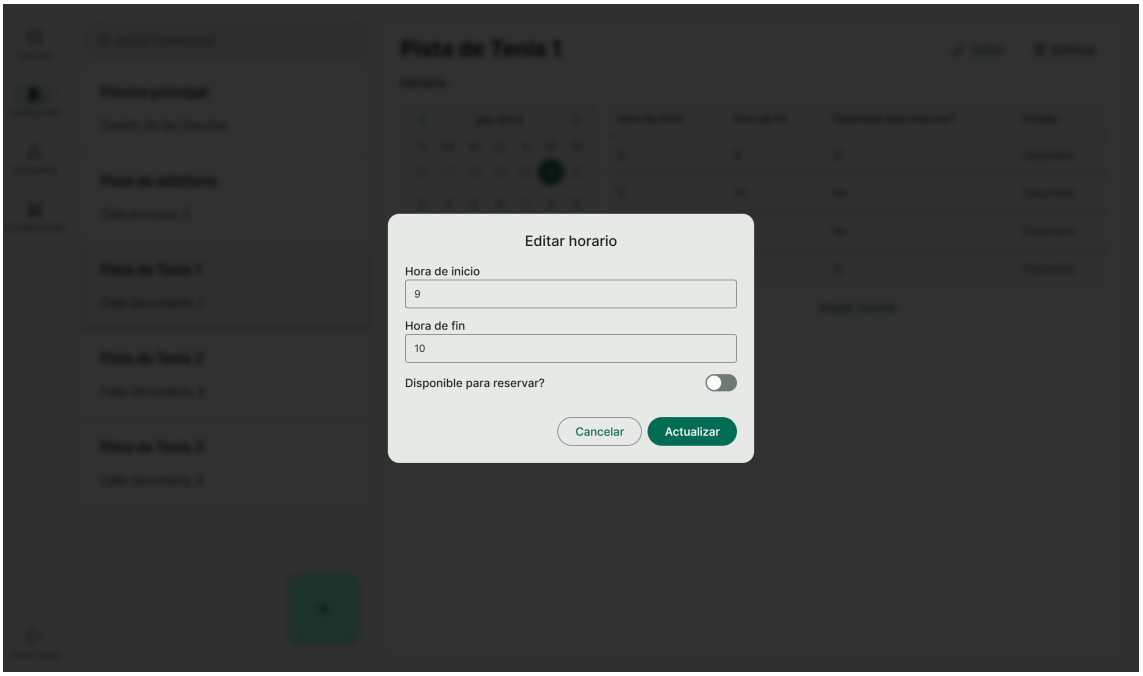

Figura 10.13: Edición de horarios de instalaciones

10.2.3. Gestionar actividades

Al hacer click en el apartado de actividades de la barra de navegación se puede acceder a la pantalla de gestión de actividades del gestor, que es similar a las otras pantallas del gestor, con una vista de lista y otra de detalle para la actividad seleccionada.

En la figura 10.14 se muestra la vista principal de actividades con la actividad "Clase de pádel" seleccionada.

Usando el botón disponible al final de la lista se puede añadir una actividad. En la figura 10.15 se muestra la pantalla de creación de actividades, que acepta el nombre, descripción y deporte asociado.

En la vista de detalle se puede ver arriba a la derecha los botones de editar y eliminar. Al hacer click en editar aparece el diálogo de la figura 10.16 dónde se puede editar el nombre, descripción y el deporte de la actividad seleccionada. Cuando se selecciona el botón de eliminar aparece el mensaje de confirmación de la figura 10.17.

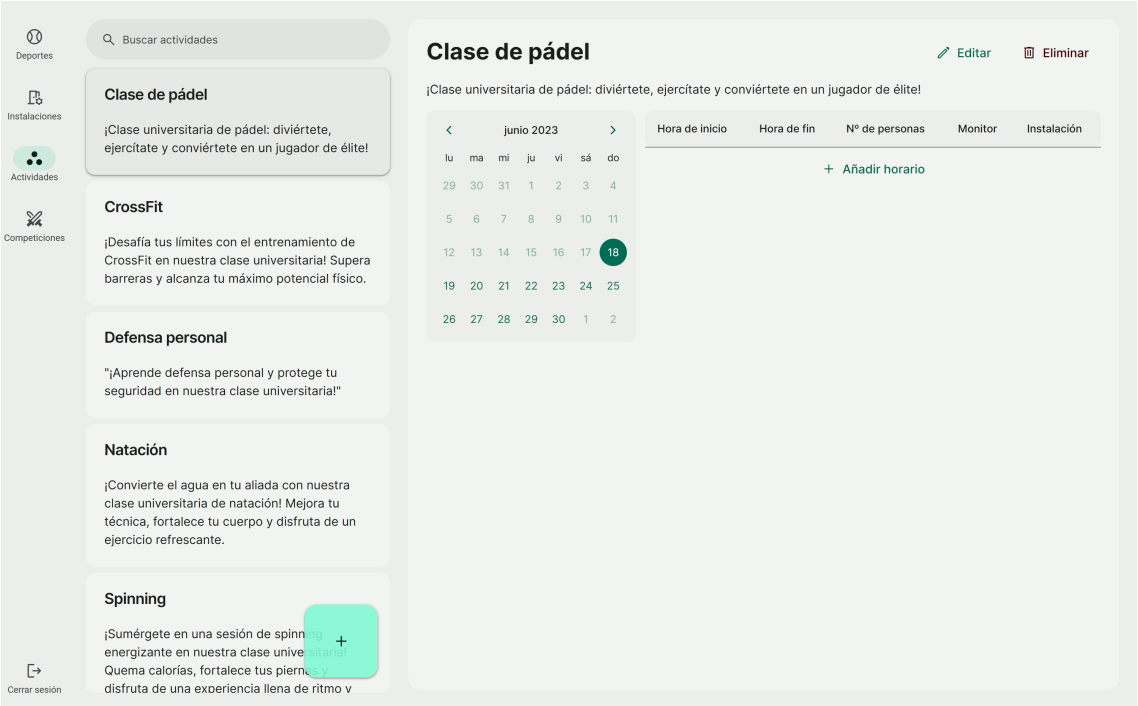

Figura 10.14: Vista de actividades

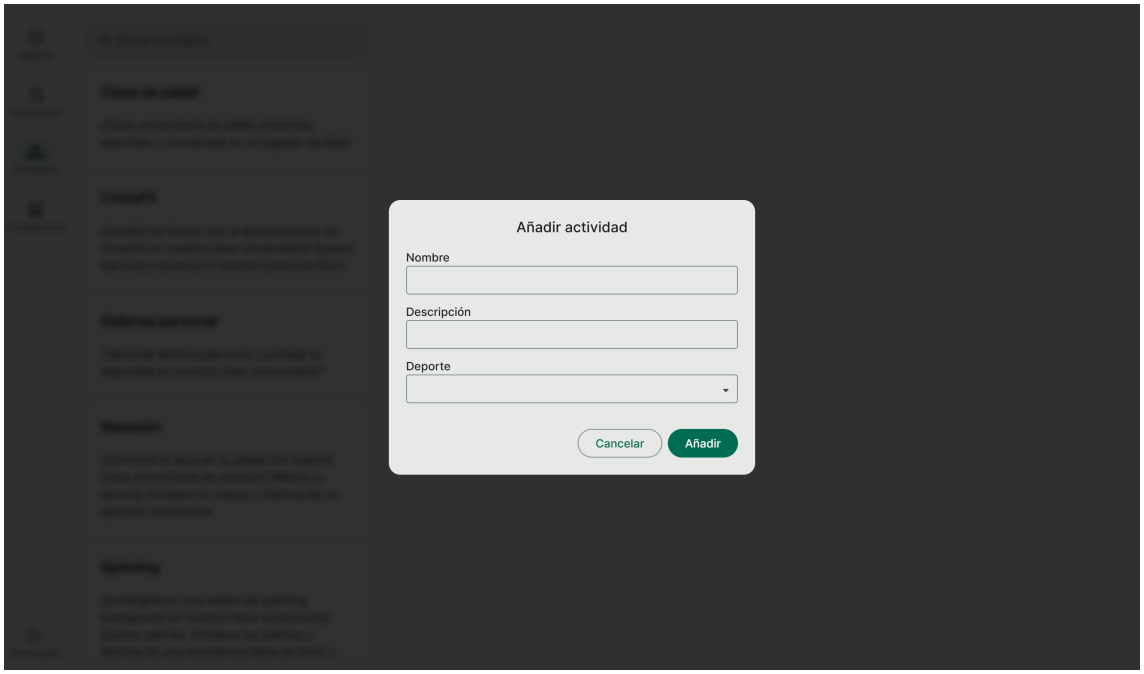

Figura 10.15: Creación de actividades

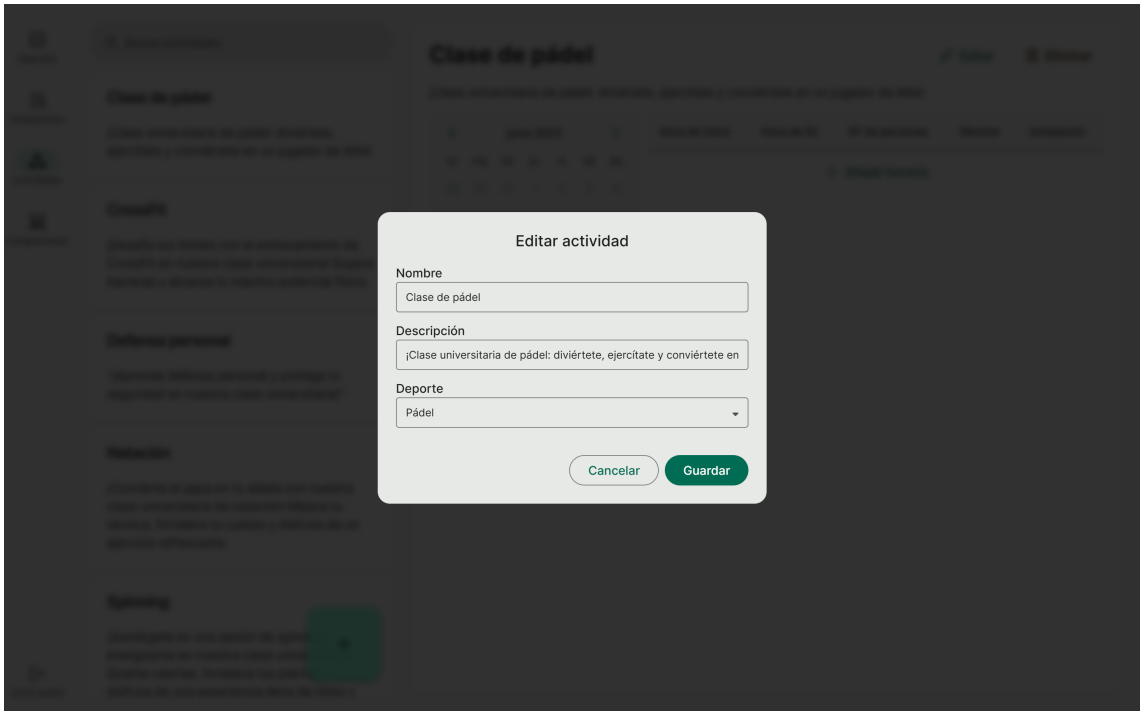

Figura 10.16: Modificación de actividades

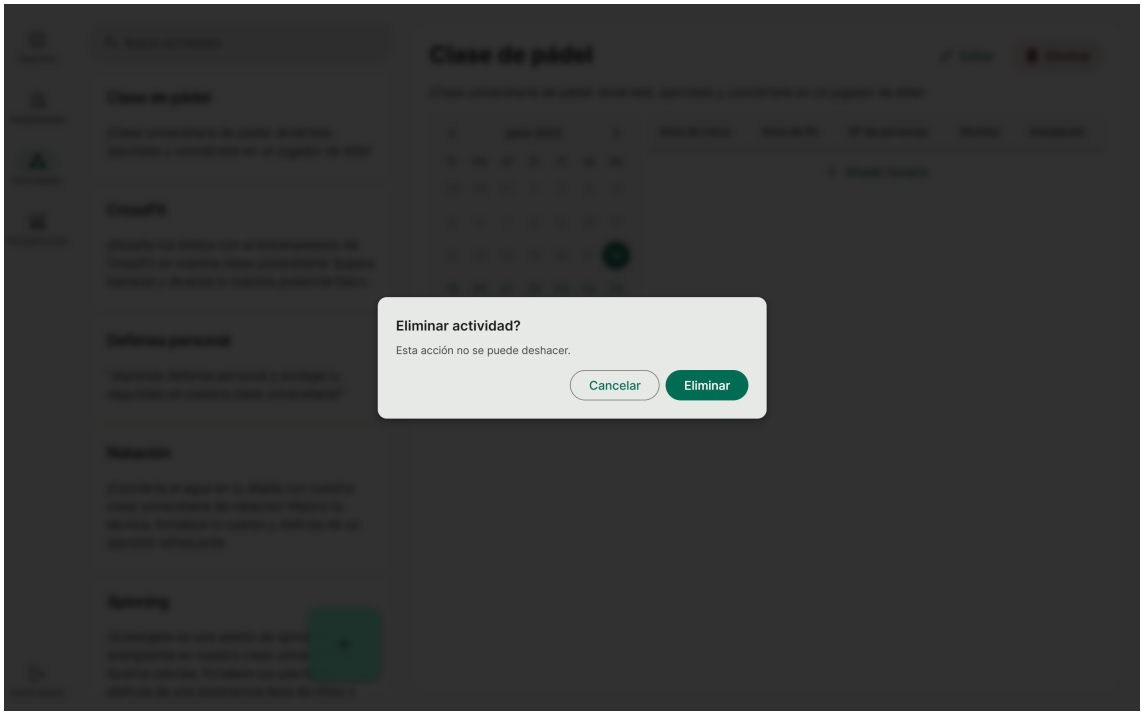

Figura 10.17: Eliminación de actividades

Una vez seleccionada una actividad para ver en detalle se ofrece un calendario para elegir la fecha y una tabla donde se puede añadir los horarios. En la figura 10.18 se muestra la vista de actividad en detalle. En la figura 10.19 se muestra la pantalla de creación de horarios de actividades, que acepta la fecha de inicio, fin y número máximo de personas inscritas. Una vez creada la actividad se puede asignar una instalación y un monitor. En la figura 10.20 se muestra la pantalla de asignación de instalación a un horario de instalación. En la figura 10.21 se muestra la pantalla de asignación de monitor a un horario de actividad. En la figura 10.22 se muestra la vista de actividad en detalle con horarios añadidos, donde tan solo el segundo tiene monitor e instalación asignado.

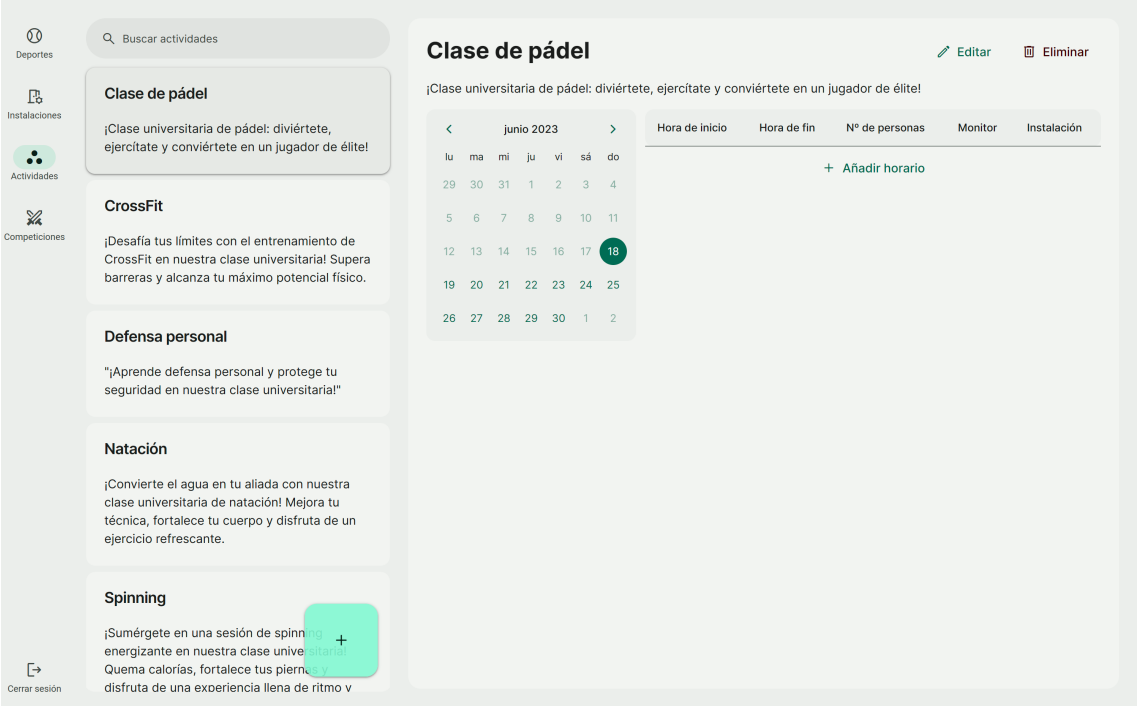

Figura 10.18: Vista de actividad en detalle

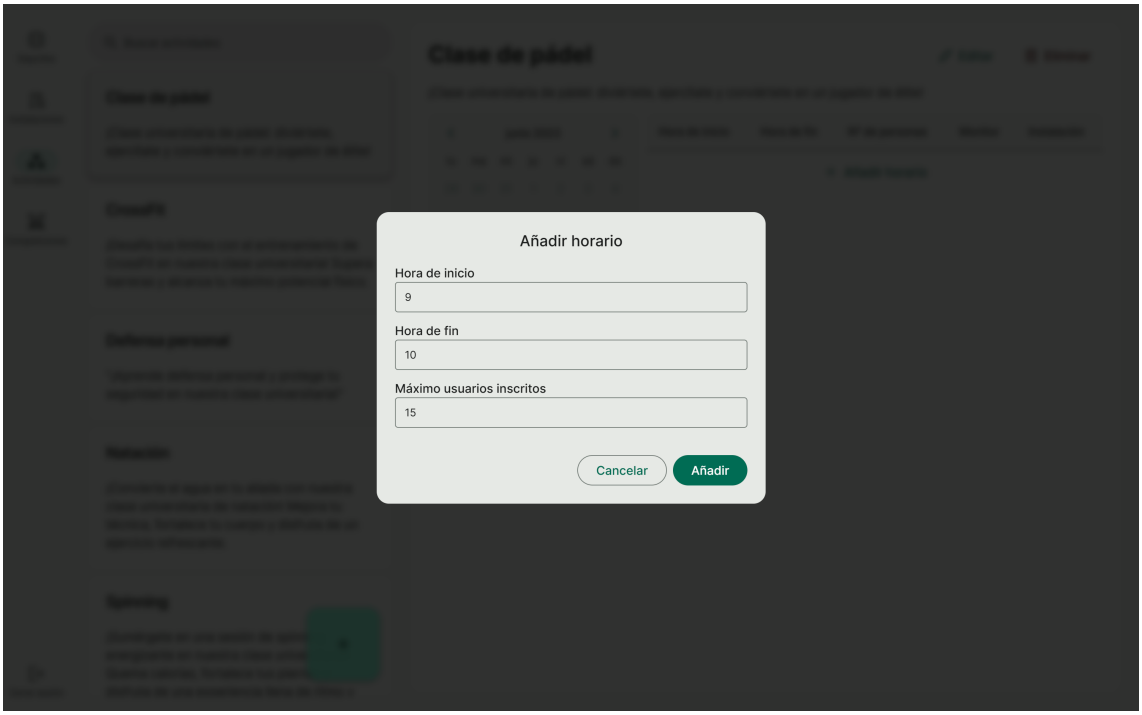

Figura 10.19: Añadir horario de actividad

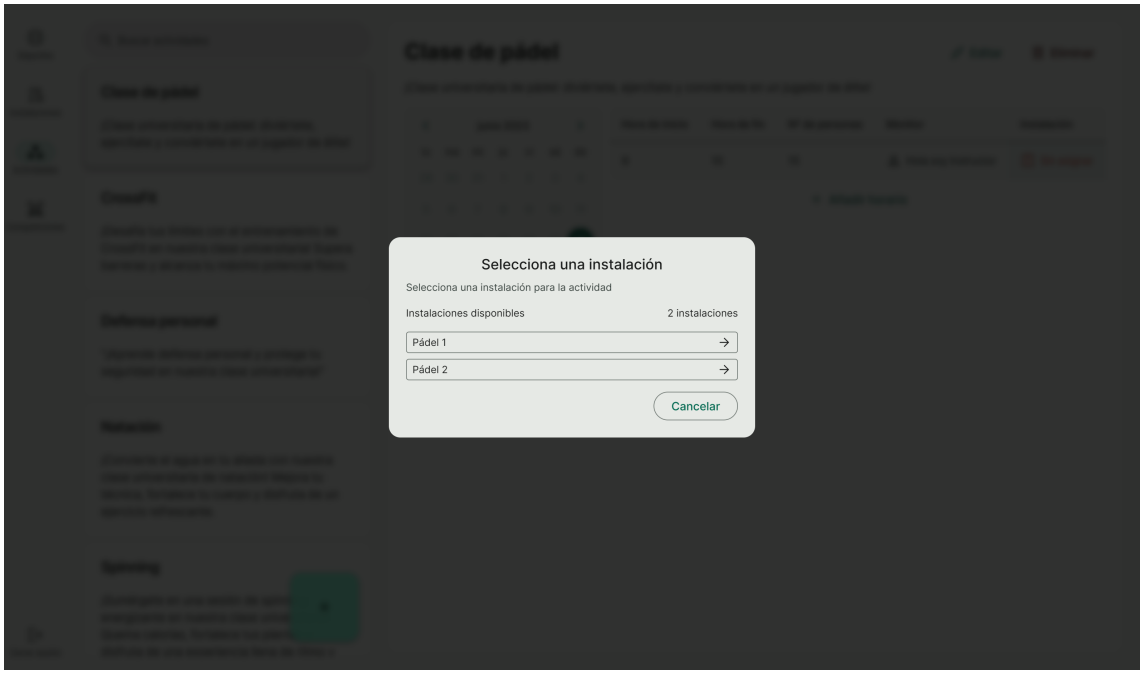

Figura 10.20: Asignar instalación a horario de actividad

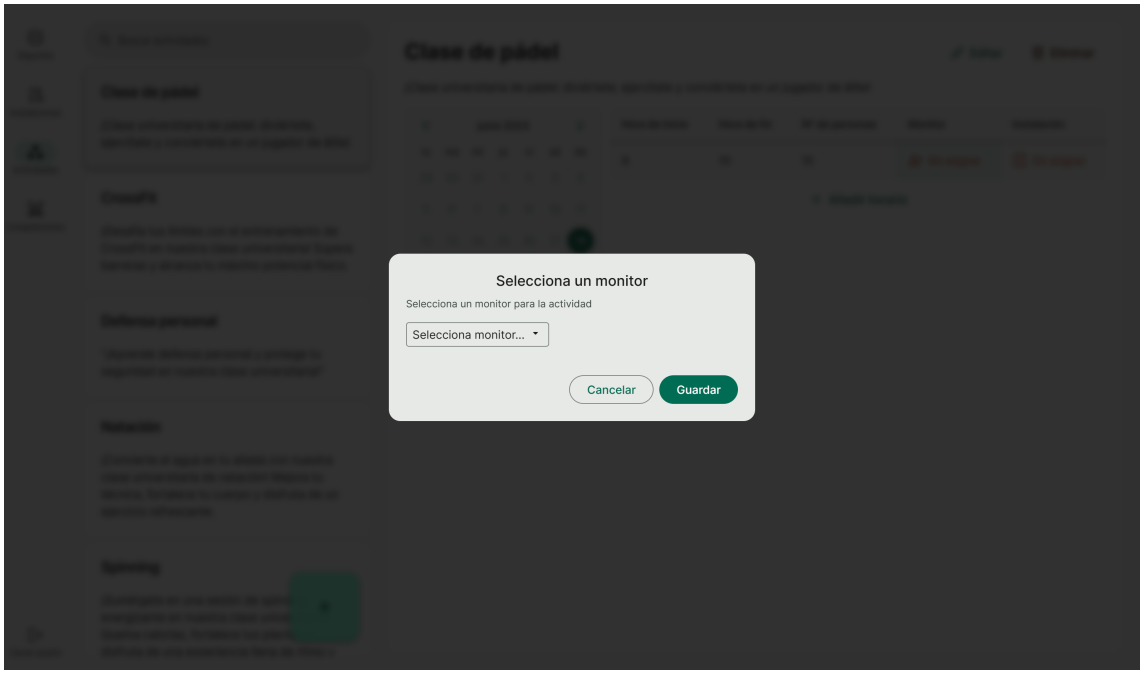

Figura 10.21: Asignar monitor a horario de actividad

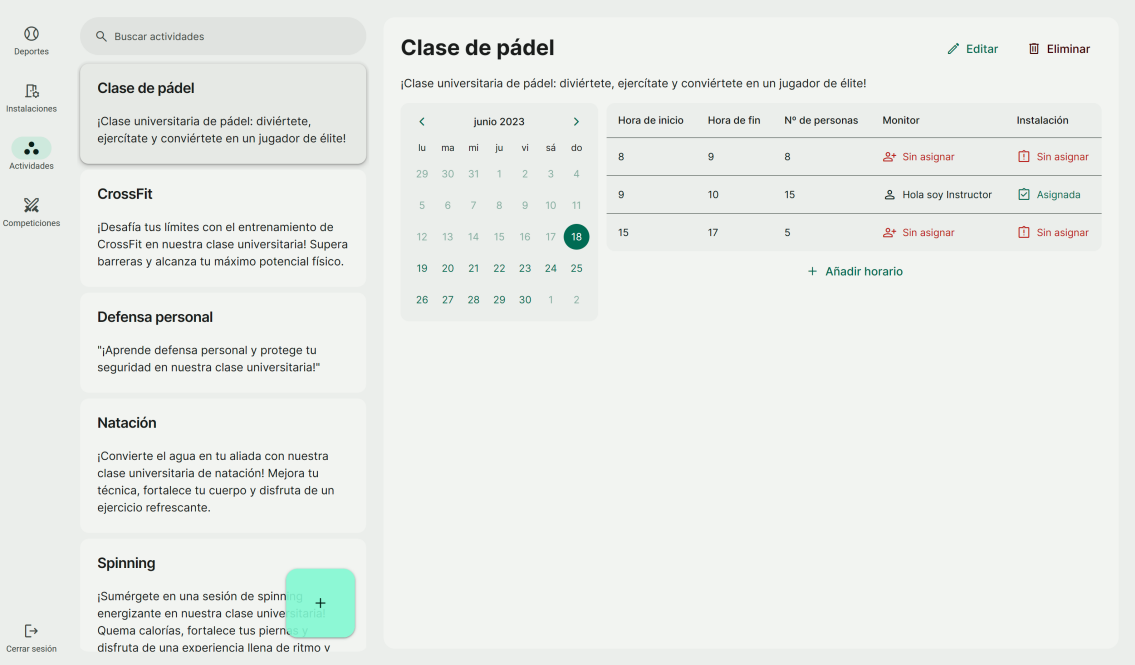

Figura 10.22: Actividad con varios horarios

10.2.4. Gestionar competiciones

Al hacer click en el apartado de competiciones de la barra de navegación se puede acceder a la pantalla de gestión de competiciones, que es similar a las otras pantallas del gestor, con una vista de lista y otra de detalle para la competición seleccionada.

En la figura 10.23 se muestra la vista principal de competiciones con la competición "Fútbol Supremo" seleccionada. Al final de la lista se encuentra un botón que permite crear nuevas competiciones. En el extremo superior derecho de la vista de detalle se encuentran los botones de editar y eliminar. Con ellos se puede editar la competición seleccionada o eliminarla.

Las competiciones pueden estar en diversos estados dependiendo de su situación. En la figura 10.23 se muestra una competición en estado "Planificada". Esto quiere decir que la competición está creada pero aún no ha comenzado ni la competición ni el periodo de inscripciones. En la figura 10.24 se muestra una competición en estado "Inscripción". En este estado los socios se pueden inscribir y los gestores pueden ver los equipos que se han apuntado. Junto a los botones de editar y eliminar, en este estado aparece el botón de "Generar cruces". Este botón permite generar los cruces de la competición y automaticamente pasa la competición al estado "En curso".

En la figura 10.25 se muestra una competición en estado "En curso", que ocurre cuando ya ha empezado. En este caso se puede ver como hay 4 equipos inscritos y 3 rondas de partidos. Además, en el extremo superior derecho de la vista de detalle se encuentra un botón para ver la clasificación de la competición en directo. Si se hace click en este botón aparece el mensaje mostrado en la figura 10.26. En la figura 10.27 se muestra una competición en estado "Finalizada", que ocurre cuando ya se han jugado todos los partidos. Aunque la competición esté

finalizada, está habilitado el botón “Ver resultados” que permite ver la clasificación final de la competición. En la figura 10.28 se muestra la clasificación final de la competición “Fútbol Supremo”.

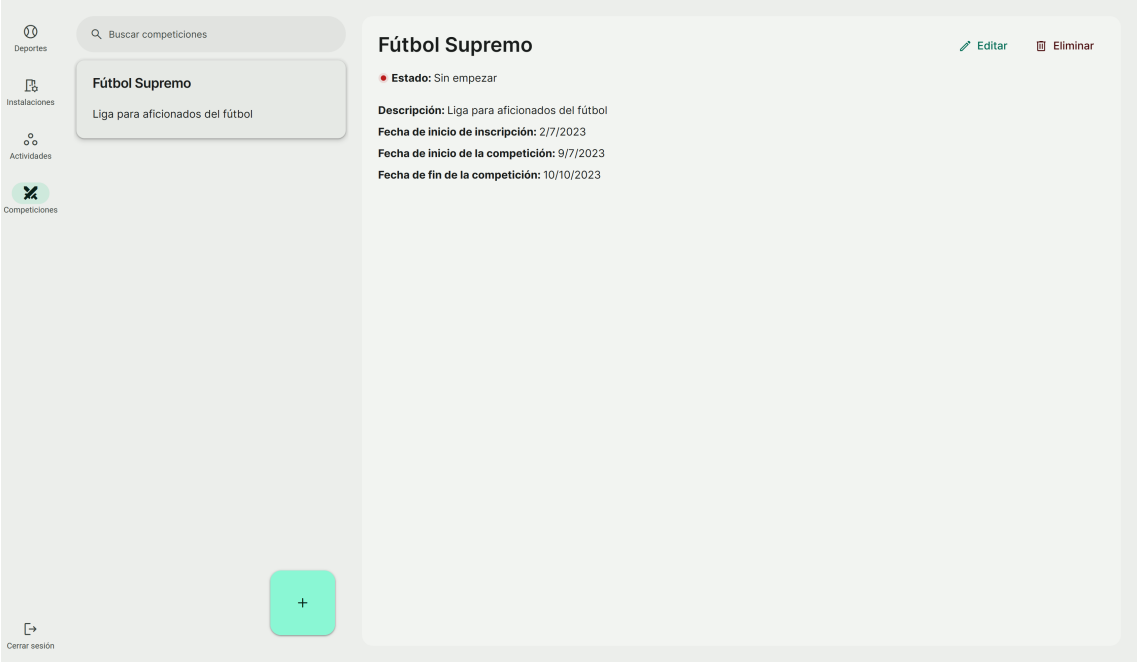

Figura 10.23: Competición planificada

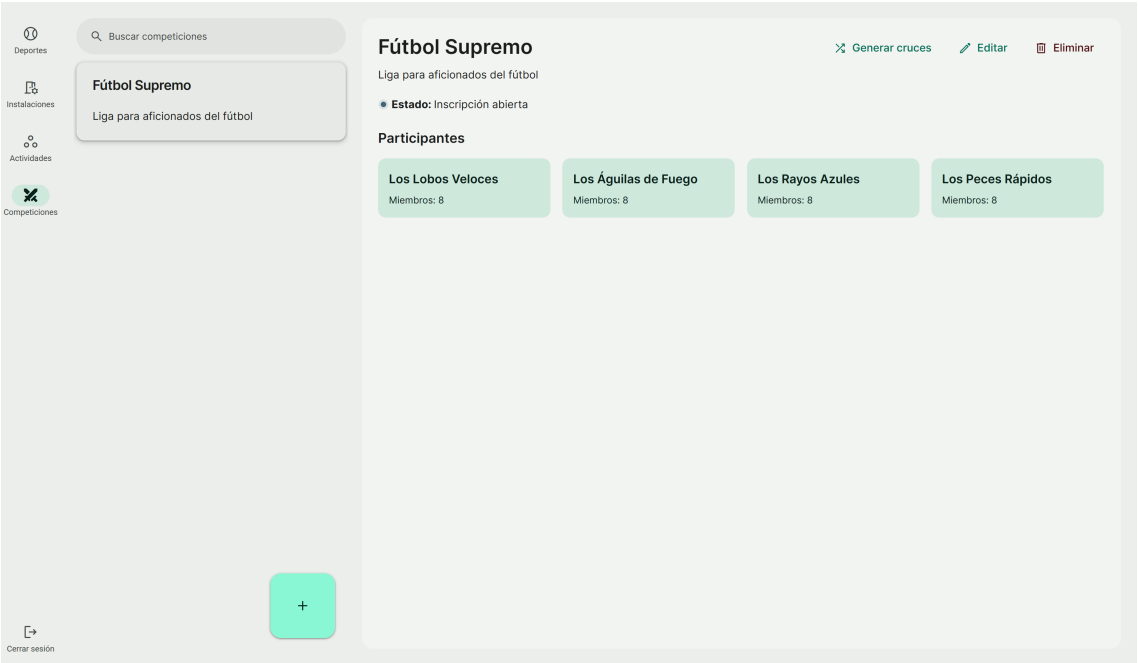

Figura 10.24: Competición en periodo de inscripción

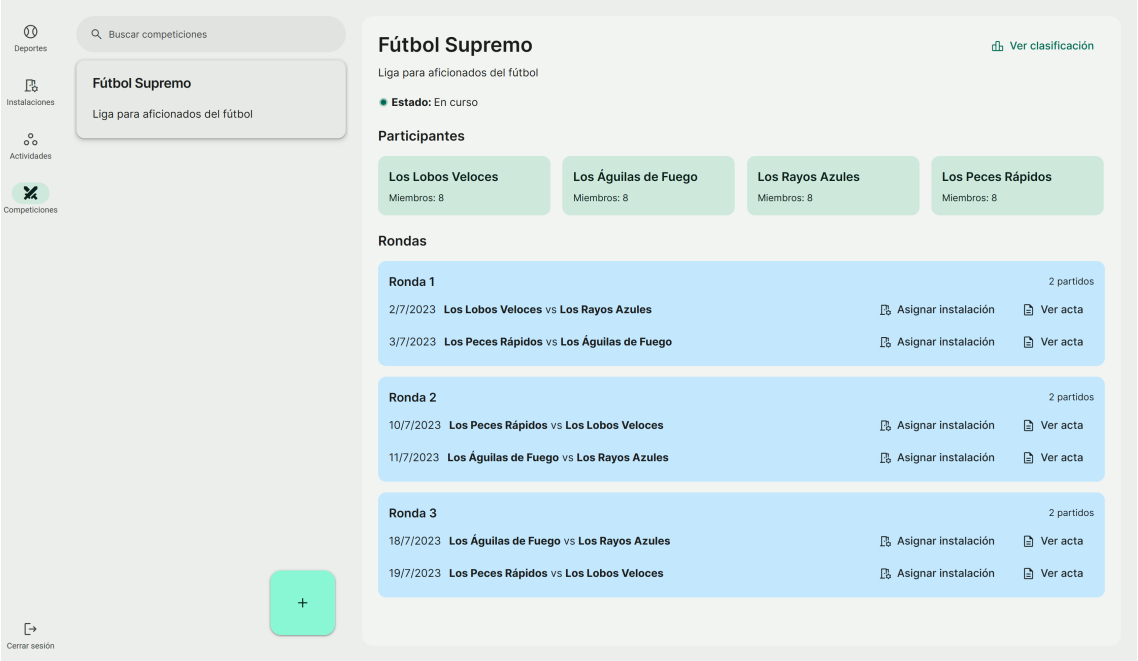

Figura 10.25: Competición en curso

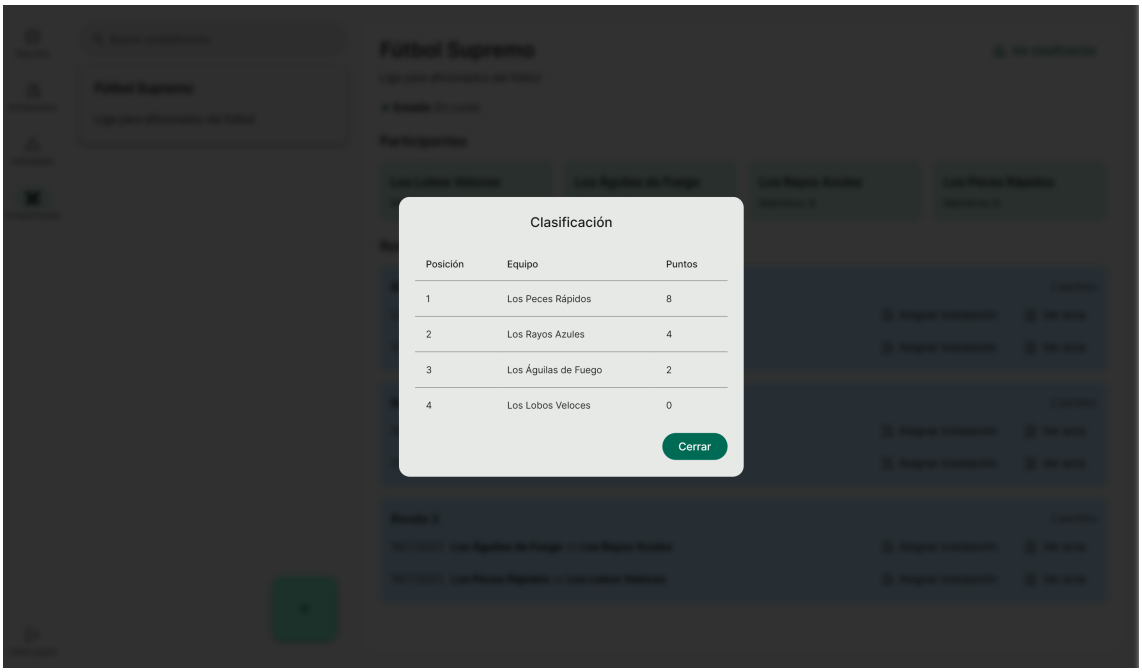

Figura 10.26: Clasificación en directo de una competición en curso

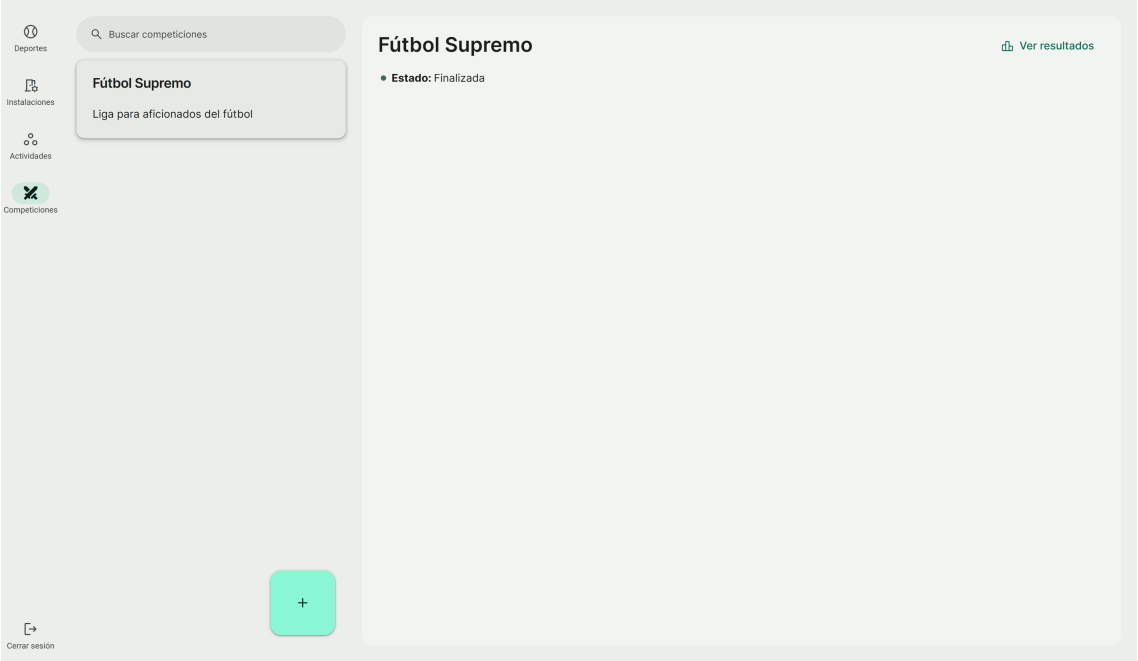

Figura 10.27: Competición finalizada

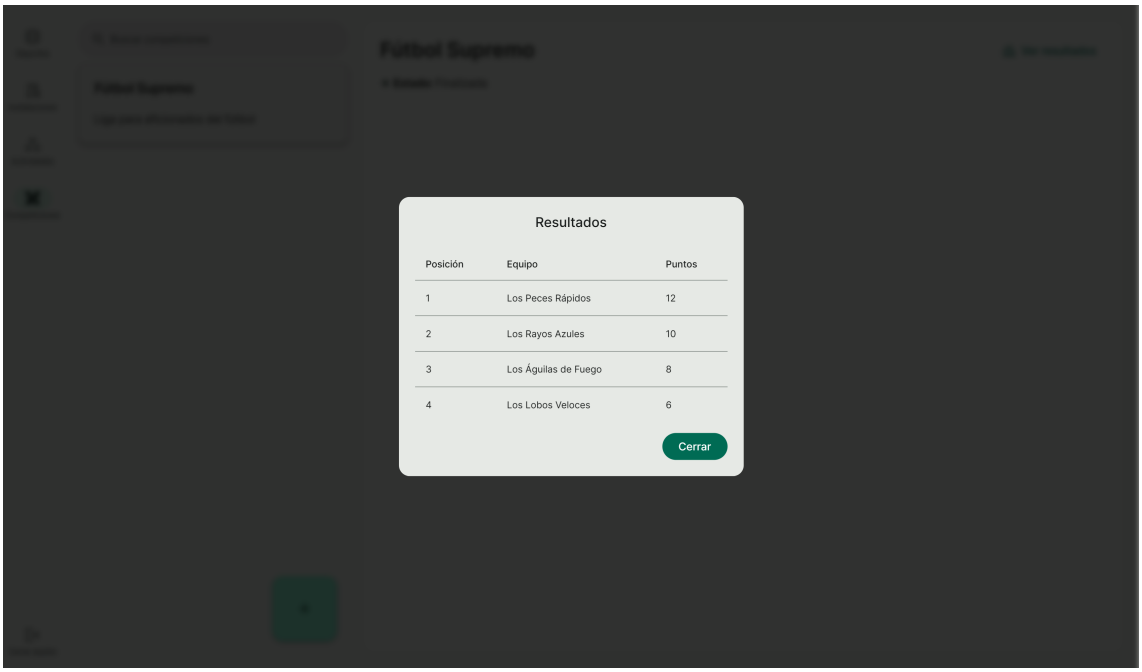

Figura 10.28: Clasificación final de una competición finalizada

10.3 Vista de socio

La vista de socio es la parte de la aplicación que se ofrece a los usuarios registrados como socios. Para acceder a la vista de gestión se debe iniciar sesión en la aplicación con un usuario con el rol de socio.

La vista de gestor tiene una barra lateral de navegación con cuatro elementos: Reservas, Actividades, Competiciones y Cerrar sesión. A continuación se va a explicar cada una de las funcionalidades de la vista de socio.

10.3.1. Reservar instalaciones

Para reservar una instalación se hace click en el apartado de Reservas de la barra de navegacion. Al entrar se presenta con una vista donde se muestran los deportes que permiten reservas. Esto se puede ver en la figura 10.29 dónde se muestra que hay dos deportes disponibles para reservar las instalaciones: Atletismo y Tenis. Al hacer click en un deporte se puede escoger la fecha y se muestra el horario de la instalación con la disponibilidad. En la figura 10.30 se puede ver el horario disponible para la pista de tenis. Encima de los horarios hay una selección de instalación para poder cambiar fácilmente entre las instalaciones de un mismo deporte para buscar los horarios. Al hacer click en un horario libre aparece un mensaje de confirmación para confirmar la reserva. En la figura 10.31 se puede ver el mensaje de confirmación de reserva de la pista de tenis. Al confirmar la reserva se muestra un mensaje de éxito y se actualiza el horario de la instalación para mostrar la reserva. En la figura 10.32 se puede ver el mensaje de éxito de la reserva de la pista de tenis.

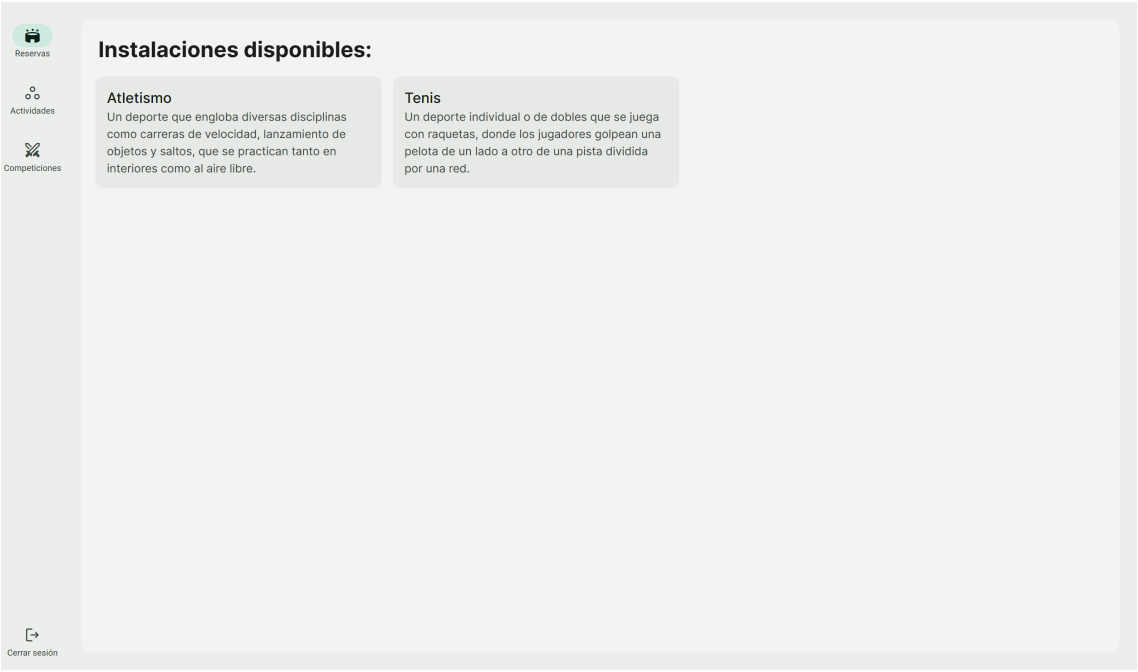

Figura 10.29: Lista de deportes que permiten reservas

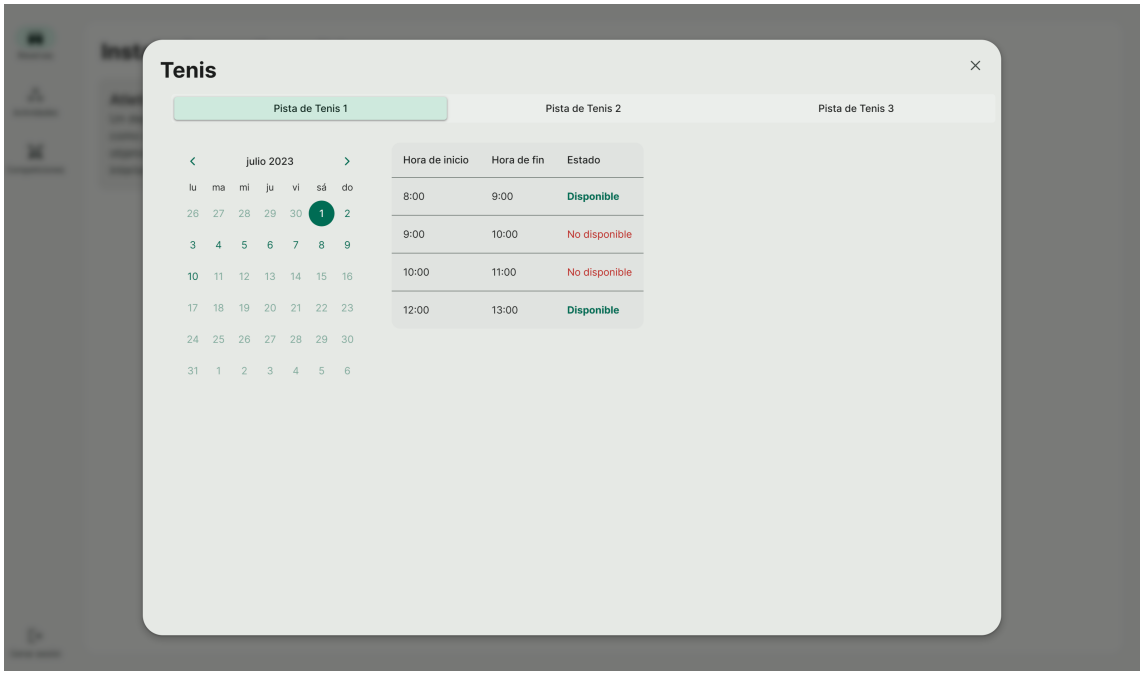

Figura 10.30: Horarios disponibles de una instalación

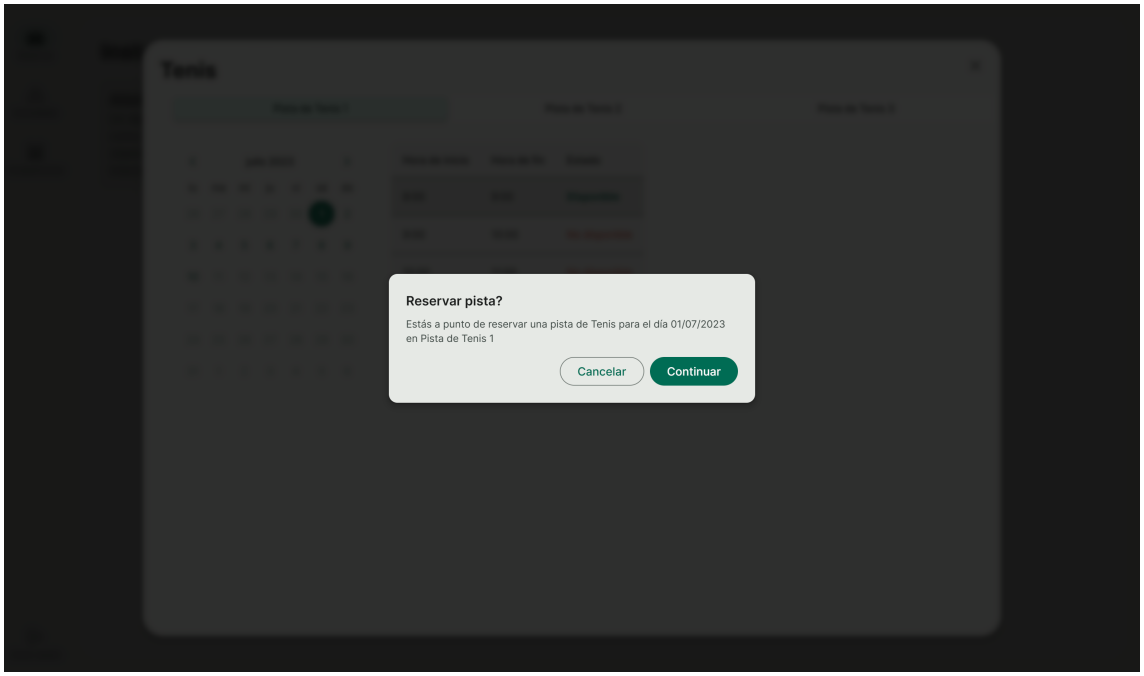

Figura 10.31: Confirmación de reserva de una instalación

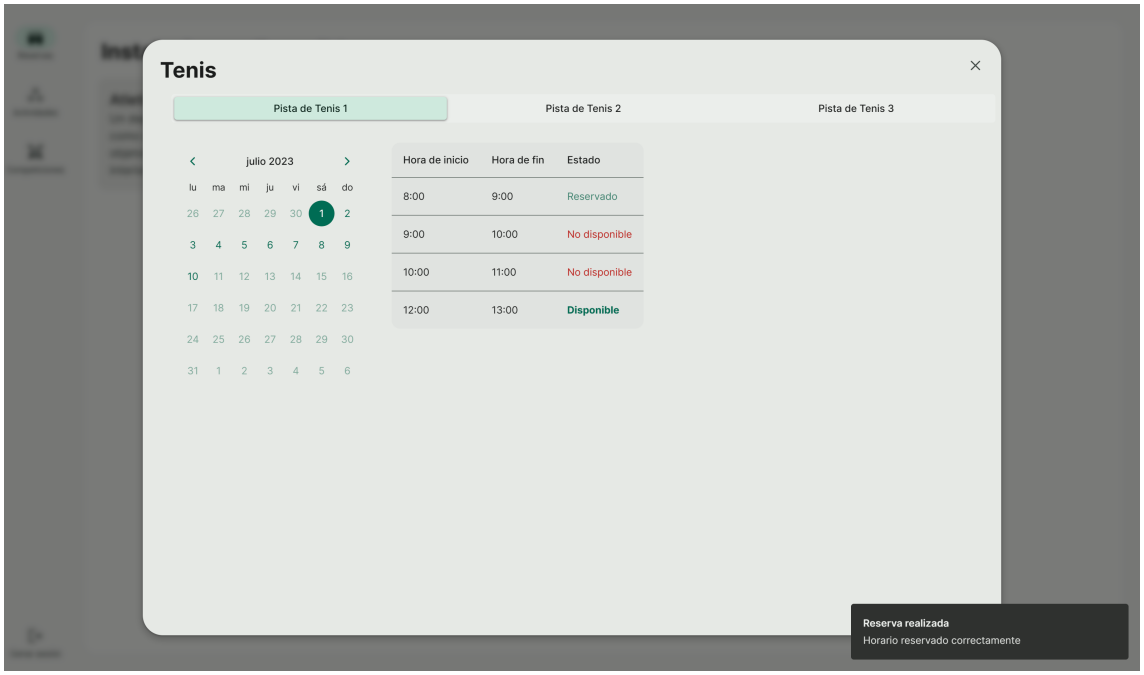

Figura 10.32: Mensaje de éxito de reserva de una instalación

10.3.2. Inscribirse a actividades

Para inscribirse a una actividad se hace click en el apartado de Actividades de la barra de navegacion. Al entrar se presenta con una vista donde se muestran las actividades disponibles. Esto se puede ver en la figura 10.33 dónde se muestra que hay dos actividades disponibles. Al hacer click en una actividad se puede escoger la fecha y aparecen los horarios de la actividad. En la figura 10.34 se puede ver el horario disponible para la actividad "Iniciación al Atletismo". Al hacer click en un horario libre aparece un mensaje de confirmación para confirmar la inscripción. En la figura 10.35 se puede ver el mensaje de confirmación de inscripción a la actividad. Al confirmar la inscripción se muestra un mensaje de éxito y se actualiza el horario de la actividad para mostrar la inscripción. En la figura 10.36 se puede ver el mensaje de éxito de la inscripción a la actividad "Iniciación al Atletismo".

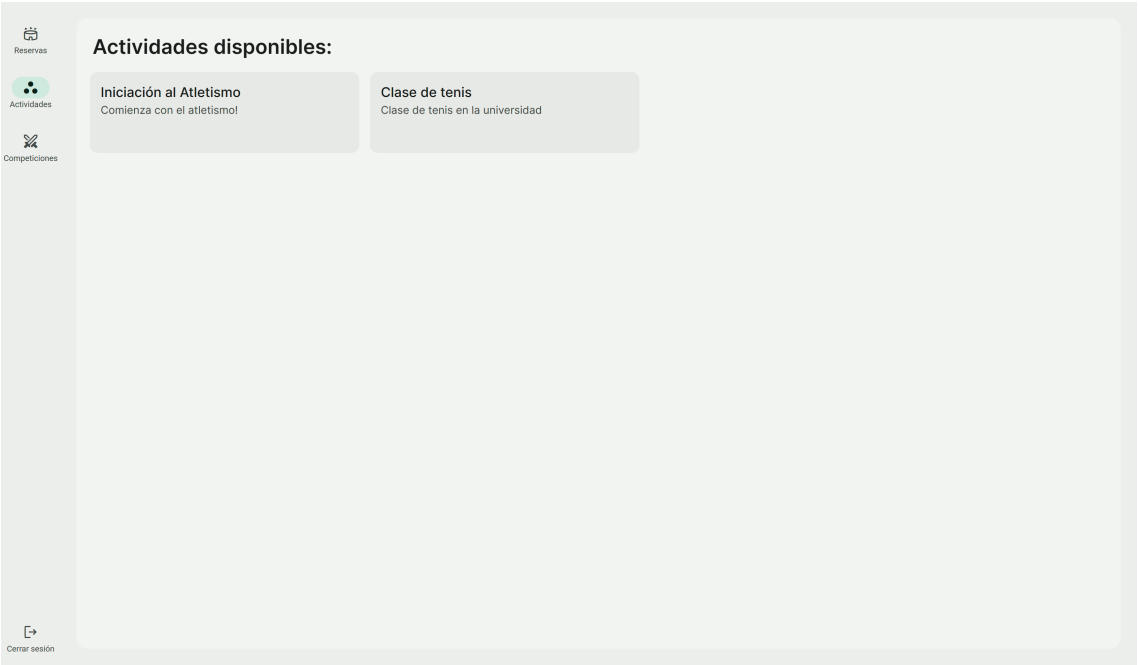

Figura 10.33: Lista de actividades disponibles

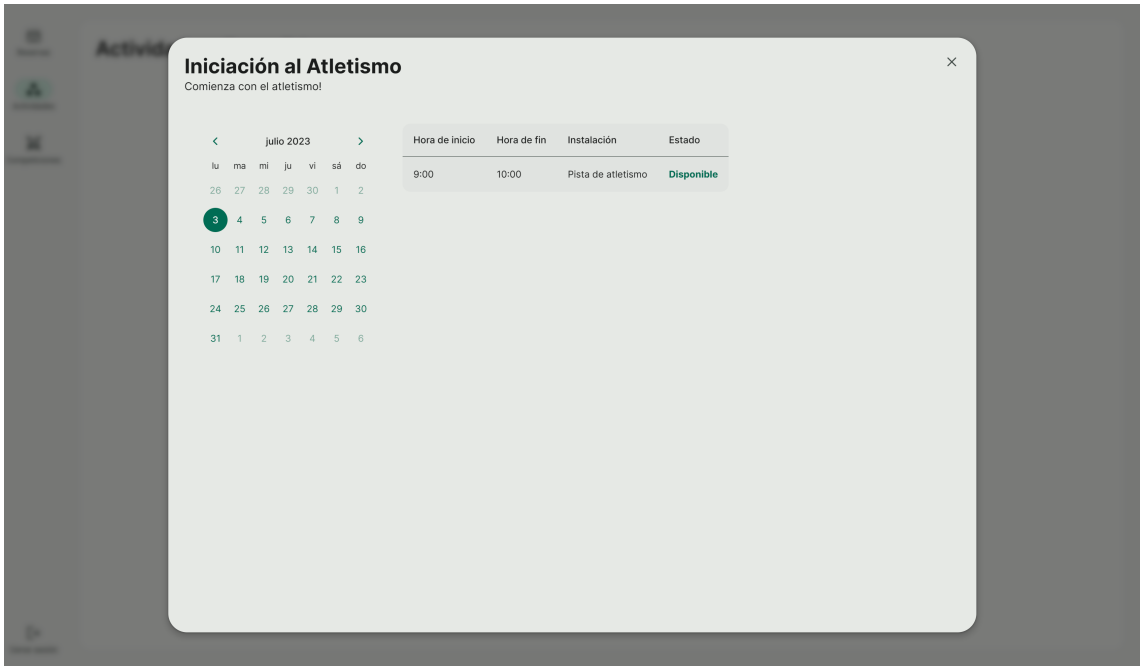

Figura 10.34: Horarios disponibles de una actividad

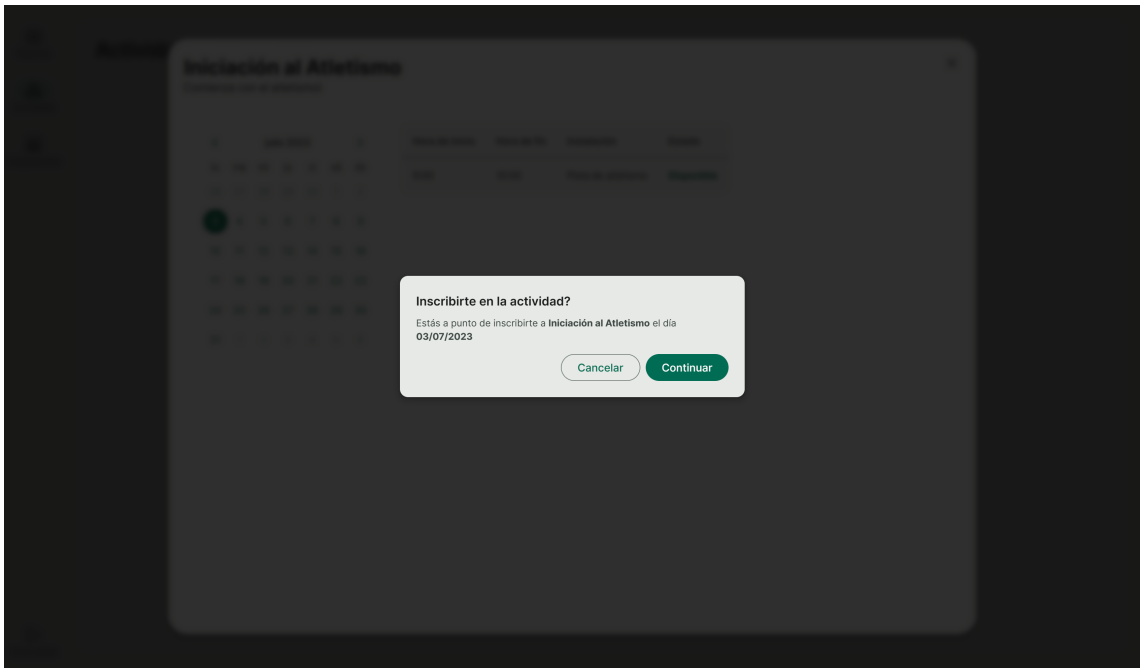

Figura 10.35: Confirmación de inscripción a una actividad

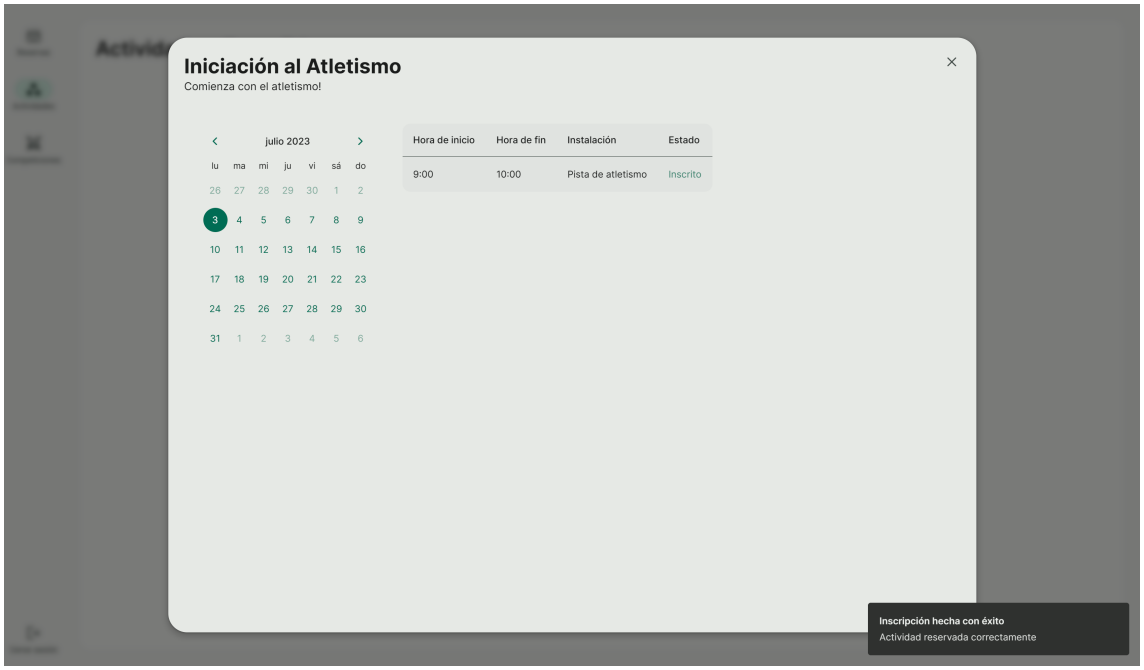

Figura 10.36: Mensaje de éxito de inscripción a una actividad

10.3.3. Participar en competiciones

Para participar en una competición se accede al apartado de Competiciones de la barra de navegacion. Al entrar se presenta con una vista donde se muestran las competiciones disponibles. Al entrar al detalle de la competición se muestran los equipos inscritos. En la figura 10.37 se pueden ver los equipos inscritos en la competición. Al final de la lista está la opción de inscribirse en la competición. Al hacer click en este botón se abre el modal mostrado en la figura 10.38 dónde se escoge el nombre del equipo y se permite añadir a los compañeros usando el DNI.

Una vez la competición ha empezado se permite ver los partidos. En la figura 10.39 se muestran los partidos de la competición en la que se participa. Si se juega una competición de un deporte de raqueta donde las actas las rellenan los socios, se ofrece la opción de rellenar el acta. En la figura 10.40 aparece el modal para rellenar el acta del partido. Finalmente, una vez ha terminado la competición tan solo se muestran los resultados, como se puede observar en la figura 10.41.

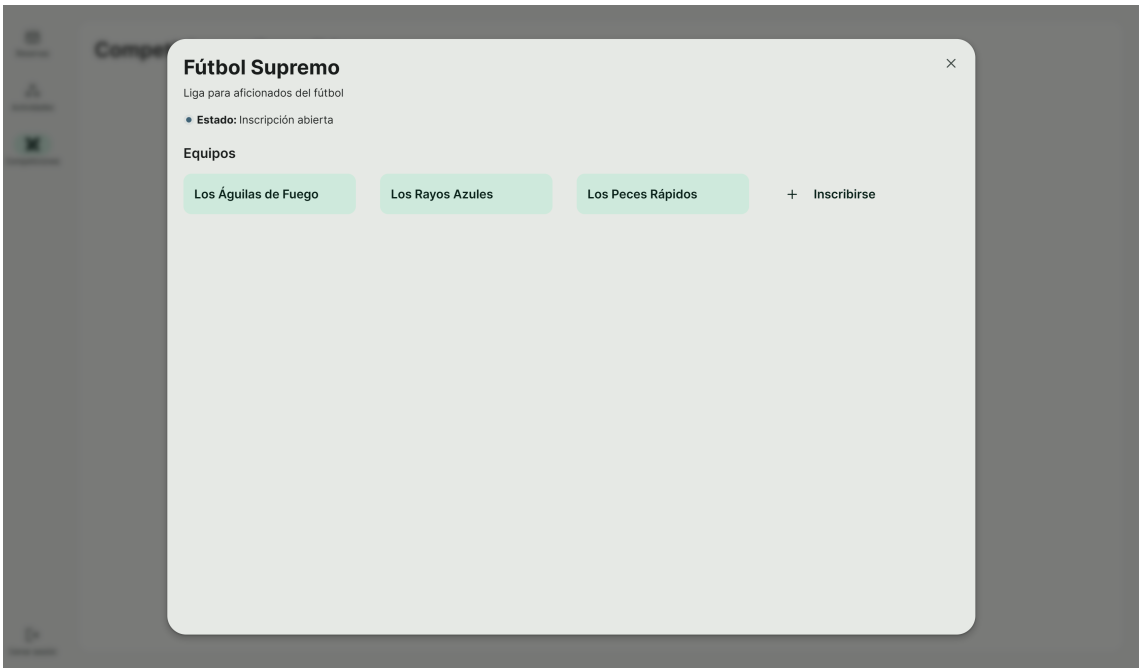

Figura 10.37: Equipos inscritos en una competición

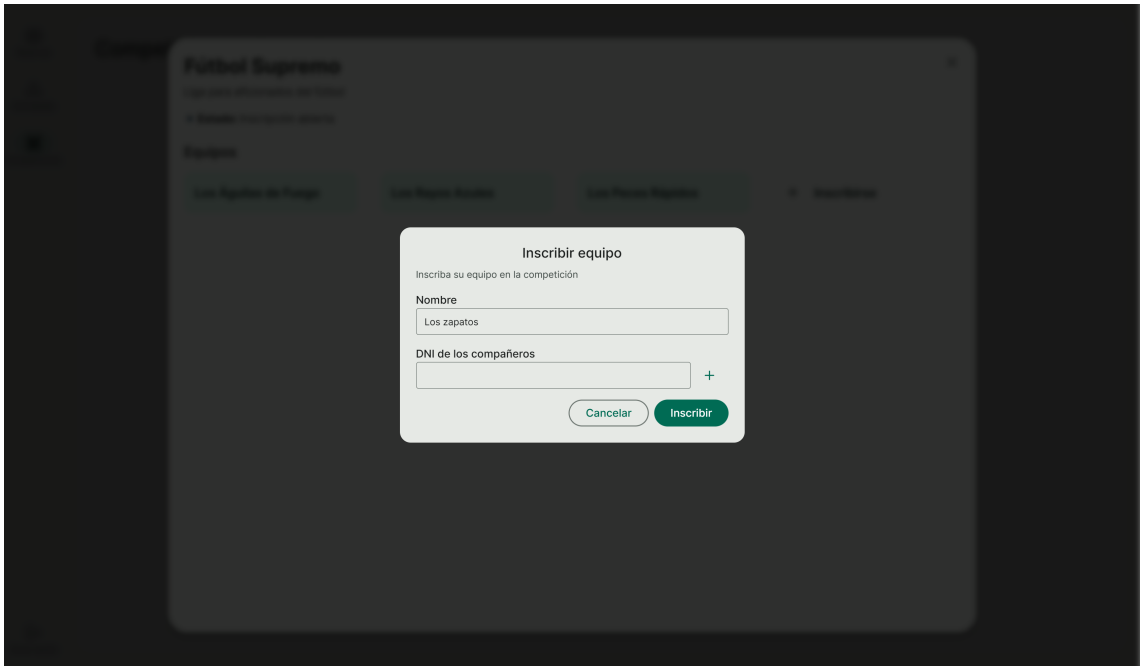

Figura 10.38: Modal para inscribirse en una competición

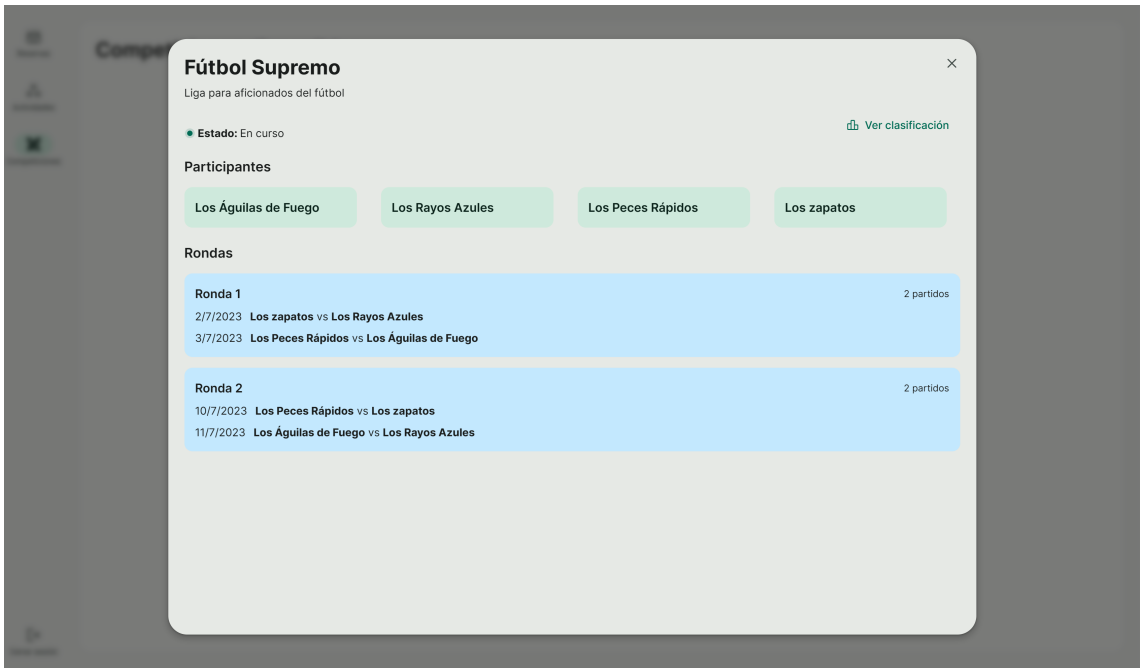

Figura 10.39: Partidos de una competición

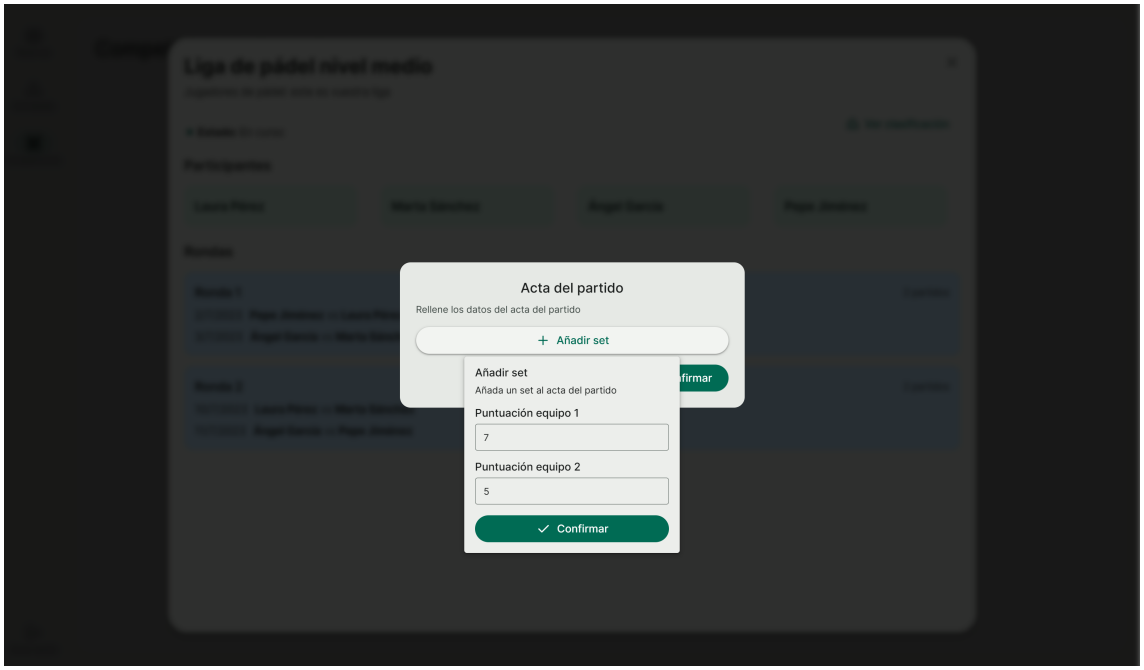

Figura 10.40: Modal para rellenar el acta de un partido

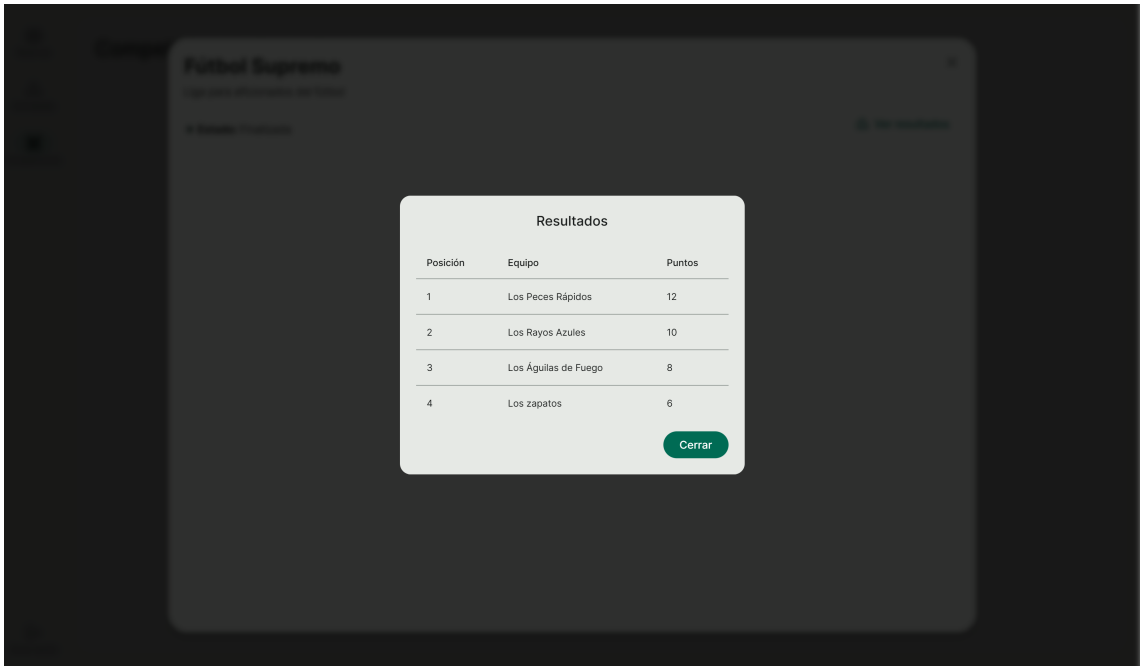

Figura 10.41: Resultados de una competición

10.4 Vista de monitor

La vista de monitor es la parte de la aplicación que se ofrece a los monitores de las actividades. Para acceder a esta vista se debe iniciar sesión con un usuario que sea un monitor.

La vista de monitor tiene una barra lateral de navegación con tan solo un elemento: actividades. En este apartado se muestra una tabla con las actividades

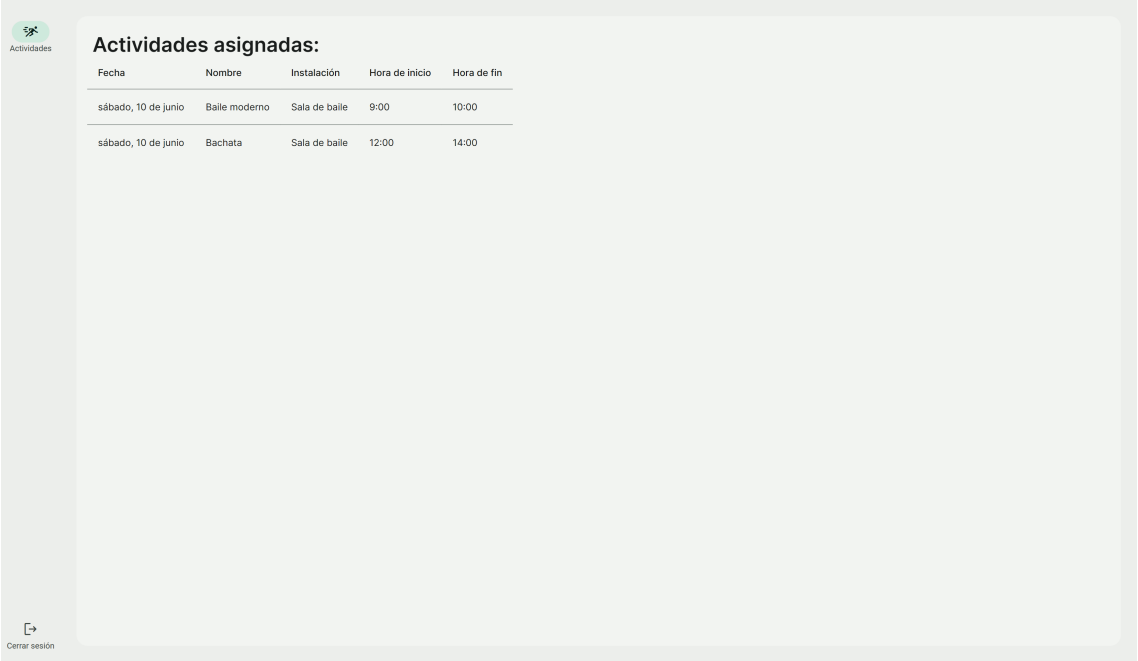

Figura 10.42: Actividades asignadas a un monitor

que el usuario tiene asignadas. En la figura 10.42 se puede ver como el usuario tiene dos actividades asignadas. En la tabla se muestra la fecha, el nombre, la instalación, la hora de inicio y la hora de fin de cada actividad.

10.5 Vista de árbitro

La vista de árbitro es la parte de la aplicación que se ofrece a los árbitros de los partidos de las competiciones. Para acceder a esta vista se debe iniciar sesión con un usuario que sea un árbitro.

La vista de ábitro tiene una barra lateral de navegación con solo un elemento: partidos. En este apartado se muestra una tabla con los partidos que tiene asignados. En la figura 10.43 se puede ver como el usuario tiene un partido asignado en el campo de fútbol siete. Al hacer click en un partido asignado se muestra el modal de la figura 10.44 donde se permite rellenar el acta del partido. En el acta se le permite al árbitro añadir eventos que han ocurrido durante el partido. Cada evento tiene un tipo, un jugador asociado, un equipo asociado, un minuto del partido y una descripción. En la figura 10.45 se muestra el formulario para añadir un evento.

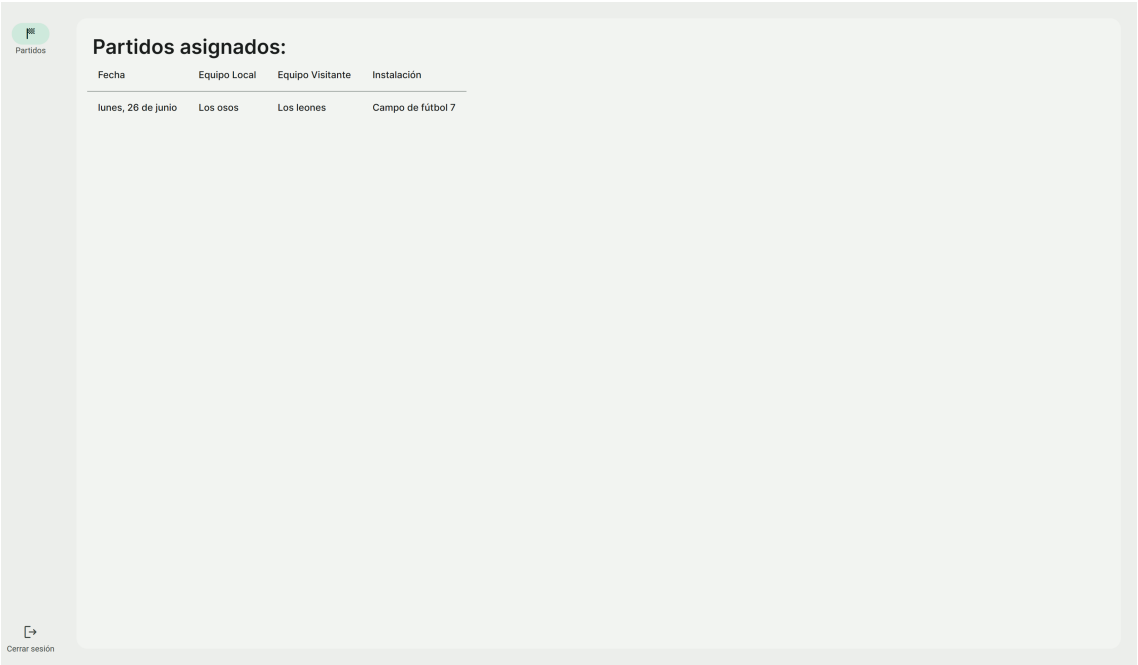

Figura 10.43: Partidos asignados a un árbitro

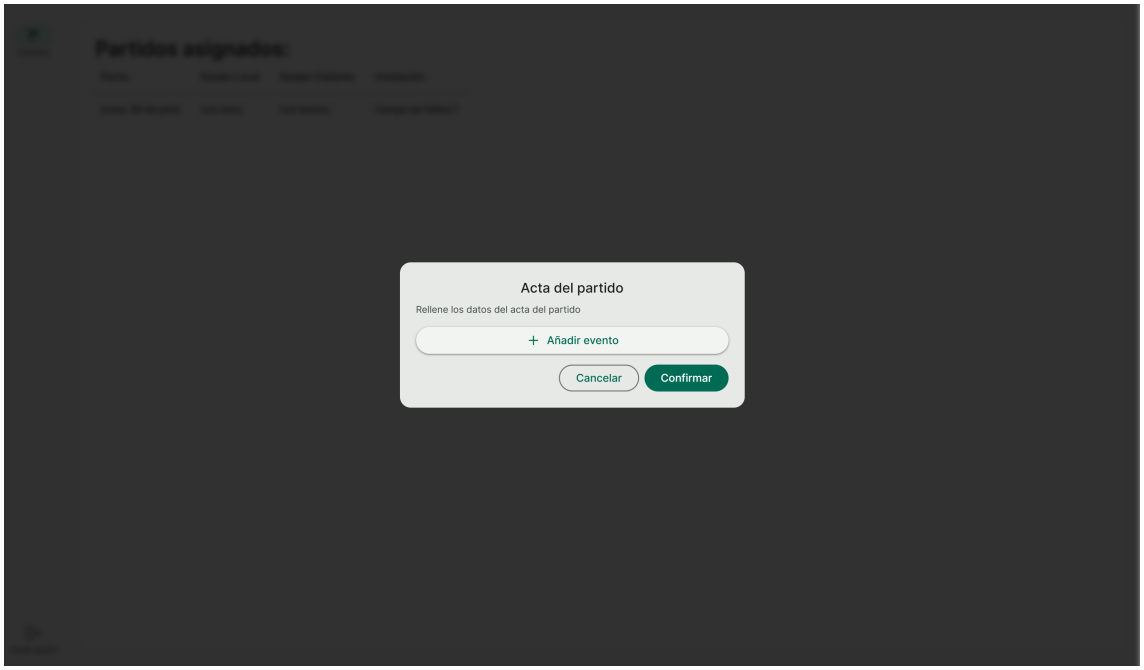

Figura 10.44: Modal para rellenar el acta de un partido

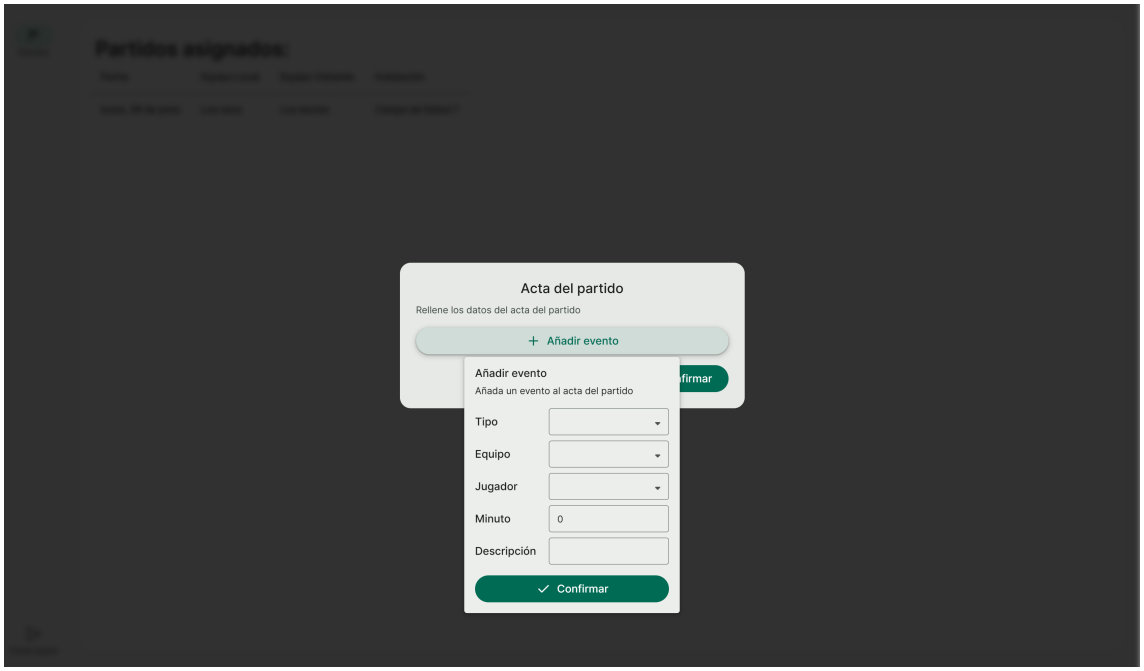

Figura 10.45: Formulario para añadir un evento al acta

CAPÍTULO 11
Conclusiones

Esta memoria recoge la documentación del Trabajo de Fin de Grado que consiste en el desarrollo de una aplicación web para la gestión de actividades deportivas en un centro deportivo.

La aplicación resultante del proyecto es plenamente funcional y satisface los requisitos planteados en la fase de análisis. Recoge la gestión de la actividad deportiva universitaria en una aplicación web, permitiendo a los usuarios gestionar los deportes, las actividades, los horarios, las instalaciones y las competiciones. Gracias a las guías de diseño de *Material Design 3* y la librería de componentes, la aplicación tiene un aspecto moderno, cohesivo y atractivo para el usuario. También, la aplicación coordina los horarios de las instalaciones con los horarios de las actividades y los partidos de las competiciones, evitando así que se solapen. Por lo tanto, todos los objetivos definidos en el apartado 1.2 de este documento se han cumplido. La aplicación mejora en todos los aspectos las aplicaciones disponibles actualmente, convirtiéndose en una herramienta indispensable para la gestión de actividades deportivas en un campus universitario.

En cuanto a las tecnologías utilizadas, el proyecto me ha ayudado a afianzar los conocimientos adquiridos durante la carrera, especialmente en la ingeniería del software. También, he aprendido a utilizar nuevas tecnologías como *Next.js 13.4* y *Zod* que me han permitido desarrollar una aplicación web moderna y con un rendimiento alto. Además, he aprendido lo que conlleva desarrollar una librería de componentes y cómo se puede utilizar para mantener un diseño consistente en toda la aplicación. Diseñando la interfaz de usuario he podido aplicar los conocimientos adquiridos en la asignatura de *Interfaces Persona-Computador* y durante la planificación del proyecto he podido aplicar los conocimientos adquiridos en la asignatura de *Proceso de Software* y *Proyecto de Ingeniería de Software*.

11.1 Trabajo futuro

En la versión final de la aplicación se han implementado todas las funcionalidades definidas en la fase de análisis. Sin embargo, se pueden añadir nuevas funcionalidades para mejorar la aplicación. A continuación, se detallan algunas de las funcionalidades que se podrían añadir en un futuro:

- **Diferentes tipos de competiciones**: en la versión final de la aplicación se ha implementado el tipo de competición *Liga*. Sin embargo, se podrían añadir otros tipos de competiciones como *Eliminatoria* o *Liga y eliminatoria*.
- **Uso de APIs con seguridad de tipos**: en la versión final de la aplicación se ha utilizado *REST* para la comunicación entre el cliente y el servidor. Sin embargo, se podrían utilizar APIs con seguridad de tipos como *GraphQL*[1] o *tRPC*[2] para mejorar la comunicación entre el cliente y el servidor, reducir el número de errores y mejorar la mantenibilidad.
- **Búsqueda fácil de instalaciones libres**: en la versión final de la aplicación se ha implementado la búsqueda de instalaciones libres en un horario concreto. Sin embargo, se podrían añadir filtros para que la búsqueda sea más fácil. Por ejemplo, se podrían añadir filtros para buscar instalaciones libres en un horario concreto y seleccionando los días que el usuario tiene disponible.
- **Elección de tema**: en la versión final de la aplicación se ha implementado un tema claro y otro oscuro. Sin embargo, se podrían añadir más temas para que el usuario pueda elegir el que más le guste. Además, el tema sigue la configuración del sistema operativo, por lo que se podría añadir la opción de que el usuario pueda elegir el tema independientemente de la configuración del sistema operativo.

[1] **GraphQL** es un lenguaje de consulta y un tiempo de ejecución del servidor para las interfaces de programación de aplicaciones (API); su función es brindar a los clientes exactamente los datos que solicitan y nada más.

[2] **tRPC** es una libería parecida a *GraphQL*, pero a diferencia de *GraphQL*, *tRPC* no es un esquema, sino un protocolo (o 'método') para exponer las funciones del backend al frontend.

Bibliografía

[1] SQL | Microsoft Learn. Consultado en https://learn.microsoft.com/en-us/cpp/data/odbc/sql?view=msvc-170.

[2] Documentación de Next.js. Consultado en https://nextjs.org/docs.

[3] Documentación de Tailwind CSS. Consultado en https://tailwindcss.com/docs.

[4] Steve Suehring *MySQL Bible*. Wiley Publishing, Inc. 2002.

[5] MySQL 8.0 Reference Manual. Consultado en https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[6] About draw.io. Consultado en https://github.com/jgraph/drawio.

[7] ¿Qué es Jira Software? Consultado en https://www.atlassian.com/es/software/jira/guides/getting-started/introduction#what-is-jira-software.

[8] Sport Facility Management Software | Swift. Consultado en https://www.runswiftapp.com/.

[9] Documentación de Zod. Consultado en https://zod.dev/.

[10] ¿Qué es GraphQL? Consultado en https://www.redhat.com/es/topics/api/what-is-graphql.

[11] tRPC, ¿la alternativa a GraphQL? Consultado en https://www.itdo.com/blog/trpc-la-alternativa-a-graphql/.

[12] ¿Qué es una API REST? Consultado en https://www.ibm.com/es-es/topics/rest-apis.

[13] ¿Qué es una API y cómo funciona? Consultado en https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces.

[14] Documentación de ESLint. Consultado en https://eslint.org/docs/latest/.

[15] Guías de estilo de Material Design 3. Consultado en https://m3.material.io/.

[16] Información de Upper Hand. Consultado en https://upperhand.com/.

[17] Características de Vite. Consultado en https://vitejs.dev/.

[18] Nguyen, Huong Single-page application and front-end testing methods: built with React and React Router, tested with Jest and Cypress 2022.

[19] Merkel, Dirk Docker: lightweight linux containers for consistent development and deployment *Linux Journal*, vol. 239, 2014.

[20] Características de Docker. Consultado en https://www.ibm.com/es-es/topics/docker

[21] Internet Engineering Task Force, .A Universally Unique IDentifier (UUID) URN Namespace,RFC 4122, June 2005. https://datatracker.ietf.org/doc/html/rfc4122.

APÉNDICE A
Objetivos de Desarrollo Sostenible

Tabla A.1: 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 7: **Trabajo decente y crecimiento económico.**				X
ODS 9: **Industria, innovación e infraestructuras.**				X
ODS 10: **Reducción de 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

El objetivo del proyecto es promover el deporte en las universidades a través de la creación de una aplicación moderna que permita a la comunidad universitaria reservar instalaciones y participar en actividades y competiciones deportivas. Por lo tanto, el proyecto tiene una relación alta con el ODS de Salud y Bienestar, ya que el deporte contribuye al bienestar físico y mental de las personas.

El deporte ofrece aprendizaje de valores fundamentales como el trabajo en equipo, la disciplina, la tolerancia, la solidaridad, el respeto de las normas y de los demás, la igualdad, el juego limpio y la cooperación. Por esta razón, el proyecto también tiene una relación alta con el ODS de Educación de Calidad, ya que el deporte es una herramienta educativa que complementa a la de los cursos universitarios y ayuda a la comunidad a desarrollar habilidades y valores que les serán útiles en su vida personal y profesional. Estos valores también están relacionados con el ODS de Paz, Justicia e Instituciones Sólidas, ya que el deporte ayuda a desarrollar competencias sociales como denominador común y pasión comparti-

da, que puede promover la colaboración y comunicación entre comunidades con diferencias culturales, religiosas y políticas.

El proyecto también tiene una relación con el ODS de Igualdad de Género, ya que el deporte es una herramienta que ayuda a promover la igualdad de género y la inclusión social. El deporte también puede fomentar la igualdad de género en la educación, la salud y el empleo, y puede ayudar a reducir la violencia de género.