UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA
INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

CLUBCONNECT

Aplicación móvil para descubrir clubes deportivos cercanos

Por: José Sebastián Rojas Rocha

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título de Ingeniero Civil Informático

Profesor Guía: Geoffrey Hecht

Enero 2025

Concepción, Chile

Resumen

La presente memoria de título tiene como objetivo desarrollar una aplicación móvil denominada "ClubConnect", destinada a promover y dar a conocer a los distintos tipos de clubes deportivos que se encuentran en las cercanías de los usuarios. La aplicación contará con diversas funcionalidades, entre ellas, la gestión de eventos para cada club, un control de asistencia para los eventos creados y la incorporación de sistema de estadísticas de asistencia. Esta última funcionalidad mencionada estará disponible para los usuarios de mayor nivel jerárquico dentro de los clubes, tales como entrenadores y/o administradores, permitiéndoles tener acceso a un registro histórico de asistencia dentro de un rango de fechas señalado, con el fin de evaluar la participación general de los equipos en los eventos organizados.

Summary

The objective of the present report is to develop a mobile application called "ClubConnect", aimed at promoting and publicizing the different types of sports clubs in the vicinity of the users. The application will have several functionalities, including event management for each club, an attendance control for the events created and the incorporation of an attendance statistics system. This last mentioned functionality will be available to the highest level users within the clubs, such as coaches and/or administrators, allowing them to have access to a historical record of attendance within a given date range, in order to evaluate the general participation of the teams in the organized events.

Índice

Lista de Tablas

Lista de Figuras

1. Introducción

La realidad de la salud en Chile es preocupante, la Encuesta Nacional de Salud arrojó alarmantes cifras donde entre ellas se encuentran que un 39,8 % de la población chilena mayor de 15 años tiene sobrepeso, 31,2 % obesidad, y un 3,2 % sufre obesidad mórbida. El sedentarismo es una causa importante de estas cifras, ya que un 86,7 % de los chilenos no realiza actividad física. De esta cifra, un 90 % de las mujeres es sedentaria y un 83 % de los hombres mayores de 15 años [6].

No solo se presentan problemáticas en ámbitos físicos, sino, también psicológicos. Según la séptima ronda del "Termómetro de Salud Mental en Chile ACHS-UC", elaborado por la Asociación Chilena de Seguridad (ACHS) y el Centro de Encuestas y Estudios Longitudinales de la Universidad Católica, un 17,5 % de las personas encuestadas exhibieron síntomas asociados a una probable presencia o sospecha de problemas de salud mental, un 13,7 % muestra síntomas moderados o severos de depresión y un 22,3 % tiene síntomas moderados o severos de ansiedad. La actividad física juega un papel fundamental en combatir estas problemáticas dado que trae beneficios tanto mentales como físicos, lo que contribuye a mejorar la calidad de vida de las personas. En particular, la actividad física puede mejorar las funciones cognitivas y propiciar un mejor bienestar en personas que padecen de alguna enfermedad mental, como es el caso de un trastorno de ansiedad, depresión o estrés [14]. En cuanto a beneficios físicos, esta ayuda a prevenir enfermedades como la hipertensión, la obesidad, entre otras [12].

Una excelente alternativa para realizar actividad física es unirse a algún club deportivo (se considera club deportivo como algún conjunto de personas y/o organización municipal que ejecute la realización de algún deporte, ya sean clubes de ANFA (Asociación Nacional de Futbol Amateur), talleres municipales deportivos, entre otros), para poder practicar algún deporte. Sin embargo, en ocasiones encontrar variedad de clubes cercanos que ofrezcan un deporte de interés puede resultar dificultoso. Buscar en redes sociales puede ser una opción para esto, pero suele suceder que muchas veces estos no se encuentran actualizados o no cuentan con información relevante. En algunos casos, cabe la posibilidad de que los clubes existentes no cuenten con las expectativas esperadas, por lo tanto, una buena alternativa es crear un club con personas cercanas y encontrar más gente que se pueda sumar a este.

Además, en conjunto con el *Club Deportivo Mewlen Angol* se obtuvo que existen diversas

problemáticas y decisiones relacionadas con la cantidad de asistencia de los usuarios a los entrenamientos. Por ejemplo, el nivel de asistencia es un factor importante a la hora de planificar las actividades, ya que un bajo nivel de asistencia podría dificultar la organización o ejecución de ciertas dinámicas. Por otro lado, la asistencia frecuente es valorada en el contexto de participación de torneos, ya que muchos entrenadores consideran la participación y constancia de los deportistas como un criterio para la selección de deportistas en competencia.

Mencionado lo anterior, la presente memoria de título tiene como objetivo crear una aplicación móvil que ayude a fomentar el deporte mediante la visualización y descubrimiento de clubes deportivos cercanos, y que distintas personas puedan encontrar un espacio donde practicar algún deporte de interés o aventurarse a practicar algún otro deporte que se lleve a cabo en sus cercanías, ya sea clubes de fútbol, senderismo, ciclismo, zumba, entre otros. Además, de proveer un sistema de registro y estadísticas de asistencia sobre los eventos de cada club, con el fin de poder planificar de mejor manera las actividades a realizar y monitorear la participación de cada uno de los miembros del club.

Para abordar las problemáticas mencionadas y cumplir con los objetivos propuestos, este informe se estructura en diferentes capítulos que detallan el proceso del desarrollo de la aplicación. Se presentan las alternativas actuales destacando la diferenciación con ClubConnect, el análisis de la aplicación donde se exponen los requerimientos funcionales y no funcionales, junto con la descripción detallada del software. El diseño, arquitectura y desarrollo, el cual define la estructura de la solución, las tecnologías utilizadas, el enfoque metodológico aplicado y implementación mediante capturas de pantalla. Además, se incluyen resultados obtenidos en las pruebas de integración y validación de usuarios.

Finalmente, se resumen los principales logros y se plantean recomendaciones para mejorar, ampliar las funcionalidades y escalabilidad de la aplicación en un ámbito deportivo.

1.1. Objetivos

En esta sección se presentan los objetivos generales y específicos de la aplicación móvil diseñada. Los objetivos generales buscan dar a conocer el hecho por el cual se propuso crear la aplicación y los objetivos específicos detallan las acciones concretas a realizar.

1.1.1. Objetivos Generales

El objetivo principal de esta memoria de título es crear una aplicación móvil que facilite la búsqueda de clubes deportivos que se encuentren cercanos, promoviendo así la participación en actividades deportivas y la interacción social, con el objetivo de fomentar un estilo de vida saludable y activo para combatir tanto el sedentarismo, obesidad, problemas de salud mental, entre otros.

El enfoque de la aplicación no se limita únicamente a la búsqueda de clubes, sino, que también incluye un sistema de registro de asistencias y estadísticas de los eventos organizados por los propios clubes, para que los administradores y/o entrenadores tengan una noción de la cantidad de participantes y así organizar de manera eficiente las actividades a realizar.

1.1.2. Objetivos Específicos

Para cumplir con los objetivos específicos se deberá definir la arquitectura del sistema, como también realizar la implementación de la base de datos correspondiente para soportar el almacenamiento y consultas que requiera el sistema. A continuación, se mencionan los objetivos específicos.

- **Implementar mapa de clubes deportivos cercanos:** Se implementará un mapa, el cual desplegará los clubes cercanos, con el fin de que los usuarios descubran distintos clubes y soliciten unirse, para estar al tanto de los eventos que estos realicen.
- **Implementar sistema de eventos y estadísticas de asistencia:** Se permitirá que los administradores y entrenadores puedan crear eventos para los equipos de cada club y que los miembros de estos pueda registrar su asistencia, con el fin de que los usuarios tengan un perfil de estadísticas de asistencia en cada equipo, así como estadísticas generales de asistencia de los equipos. De esta forma, los administradores y entrenadores tendrán facilidad de seguimiento y análisis de la participación de los eventos programados.
- **Integrar sistema de notificaciones:** Se incluirá un sistema de notificaciones, para alertar a los usuarios de la creación de eventos y a los administradores de las nuevas solicitudes de unión.
- **Crear pruebas de integración automatizadas.**

Además, para validar que la aplicación cumple con lo esperado se deberá realizar la validación con usuarios reales, para calificar la aplicación.

2. Alternativas Actuales

Esta sección tiene como propósito presentar las principales alternativas disponibles que ofrecen servicios relacionados con la gestión de clubes y el descubrimiento de clubes, diferenciando estas con *ClubConnect*.

La primera plataforma se llama *SportEasy*[13], disponible en IOS, Android y una versión web, tiene por objetivo brindar soporte en la gestión a clubes deportivos, ofreciendo funcionalidades como gestión de equipos y jugadores, asistencia, calendario de eventos, resultados y estadísticas. Lo más destacado de la plataforma es que simplifica la organización de sus actividades y mejora la comunicación de los jugadores. Pero esta carece de descubrimiento de clubes deportivos, debido a que el ingreso a los clubes es de acuerdo a un código privado.

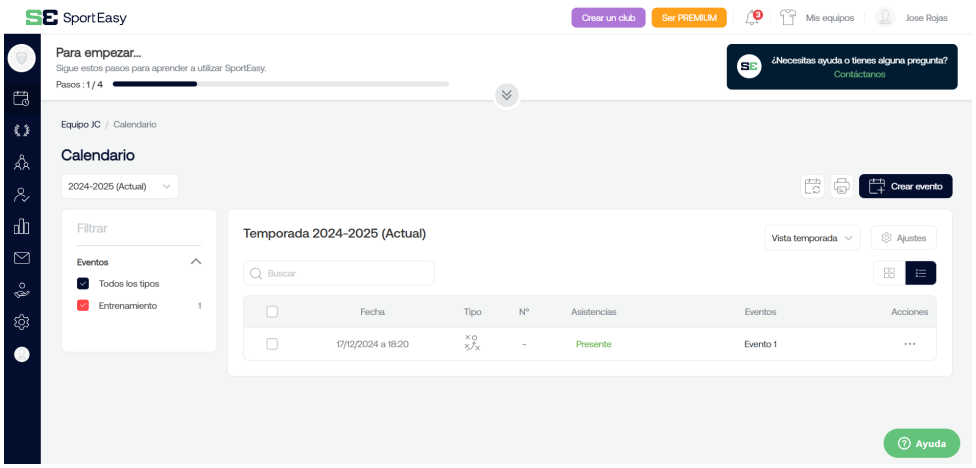

Figura 2.1: Web SportEasy

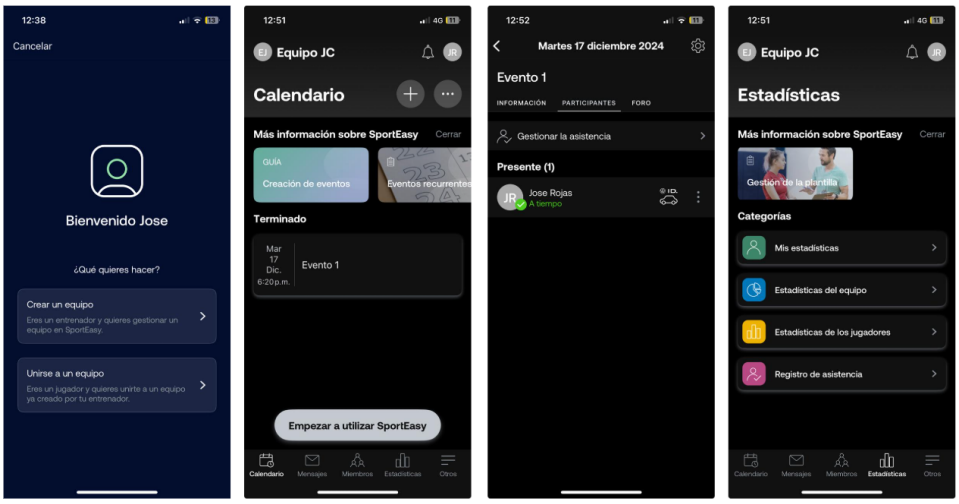

Figura 2.2: App IOS SportEasy

Otra opción es *Timpik*[4], disponible en IOS, Android y en web, ofrece organización de encuentros, búsqueda de clubes y reserva de canchas. La funcionalidad destacada de esta aplicación es la organización de encuentros. No obstante, esta aplicación cuenta con un diseño desactualizado en su versión web y la aplicación móvil no recibe actualizaciones de manera frecuente, siendo sus últimas actualizaciones en un rango de 5 años, siendo su actualización mas reciente en noviembre de 2024, y la anterior en 2019. Además, carece de información de los clubes deportivos y gestión de asistencia de estos, centrándose únicamente en la organización de encuentros.

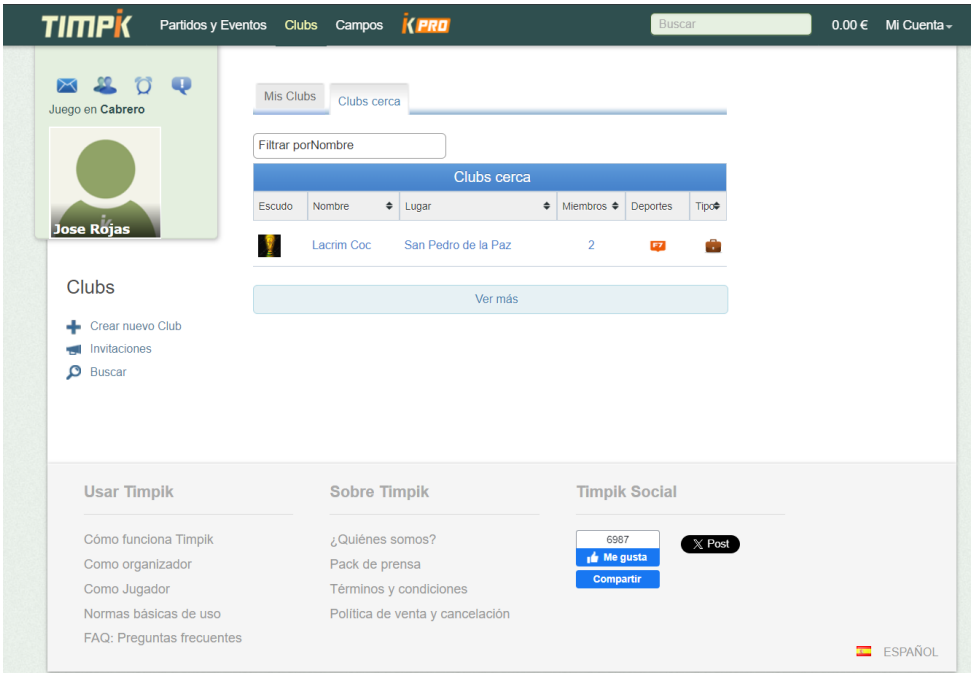

Figura 2.3: Web Timpik

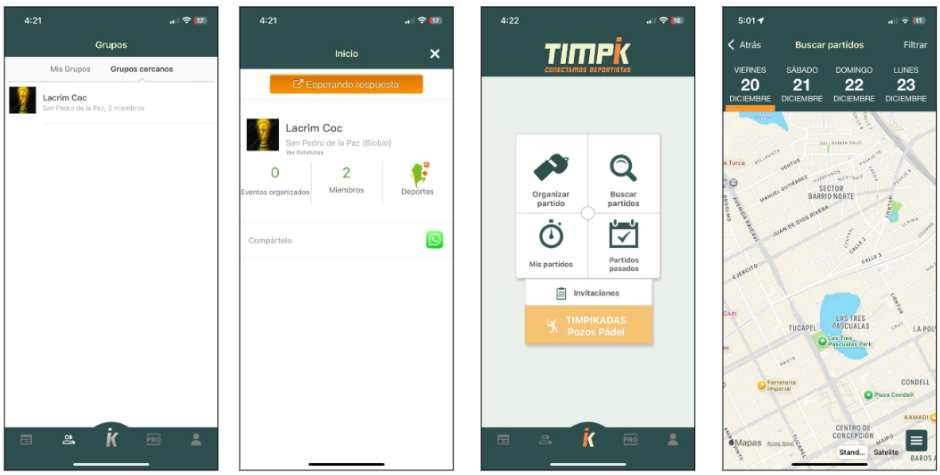

Figura 2.4: App IOS Timpik

Otra alternativa válida para la búsqueda de clubes deportivos es utilizar *Google Maps*, si bien, esto podría cubrir cierta funcionalidad, no contaría con funcionalidades de gestión de eventos y estadísticas de asistencia como lo provee *ClubConnect*.

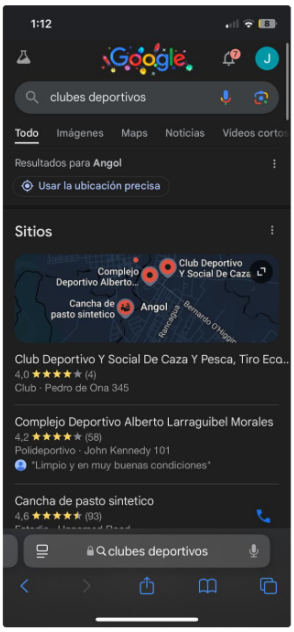

Figura 2.5: Búsqueda Clubes Deportivos Google Maps

Por lo tanto, ClubConnect en su primera versión se diferencia de estas alternativas presentando estas funcionalidades de manera unificada, destacando su búsqueda de clubes de manera clara mediante geolocalización, y además se espera que a futuro cuente con el nivel de funcionalidades más específicas para las necesidades de los usuarios y clubes, en conjunto de funcionalidades sobre eventos públicos, para aumentar la participación de

público y deportistas en los diferentes torneos y/o actividades.

3. Análisis

Esta sección tiene como objetivo analizar "ClubConnect", tanto sus requerimientos funcionales como no funcionales. Esto permitirá comprender las características de la aplicación desde una perspectiva menos técnica.

3.1. Descripción del Software

ClubConnect es una aplicación móvil dirigida a clubes deportivos y público en general, con el objetivo de facilitar la búsqueda de clubes donde realizar distintos tipos de deportes y ofrecer a estos un sistema de registro de eventos y asistencia. Por lo tanto, ClubConnect permitirá a todos los usuarios la visualización de clubes deportivos e información relevante de estos en el mapa geográfico con la posibilidad de solicitar unirse a los clubes y estar al tanto de los eventos que se realicen. Por otra parte, a los administradores y entrenadores de cada club se les permitirá crear eventos, donde los miembros de cada equipo dentro del club podrán marcar su asistencia, con la finalidad de poder llevar un registro y obtener estadísticas para un mejor análisis.

Esta aplicación deberá estar disponible tanto para IOS como para Android dada la variedad de sistemas operativos que existen, cabe destacar que en la presente memoria de título no se realiza el proceso de despliegue.

3.2. Requerimientos

Para definir los requerimientos de la aplicación, es fundamental identificar a los principales actores que interactúan con ella. A continuación, se describen brevemente:

- **Usuario general:** Es el actor inicial, a quien se le permite crear clubes, descubrir clubes y formar parte de ellos. Es necesario que se registre.
- **Deportista:** Es el usuario general, quien solicitó unirse a un club y fue asignado como "Deportista" a un equipo.
- **Entrenador:** Es el usuario general que fue asignado previamente como "Entrenador" a un equipo, tiene la capacidad de gestionar los eventos, visualizar información y

estadísticas de asistencia de los miembros dentro del equipo.

- **Administrador:** Es el usuario general que creó el club, tiene control total sobre todo el club, sus equipos y miembros. Es el ente máximo dentro de un club.

Esta categorización se define considerando que, en todos los tipos de clubes se cuenta con una persona encargada de la gestión general del club, quien tiene control sobre todos los equipos que lo conforman. Además, se identifican roles como deportistas y/ o entrenadores. Los entrenadores suelen estar presentes en clubes deportivos con múltiples equipos, teniendo privilegios sobre estos. Sin embargo, en clubes conformado de un solo equipo, como un club de zumba, es posible que no cuente con entrenadores. En tales casos, el administrador es quien toma decisiones y control sobre este equipo. Por lo tanto, el rol entrenador es opcional, por lo que depende de las necesidades del club.

Cabe recalcar que el rol de usuario general está asociado cuando el usuario navega por la aplicación sin tener seleccionado un club. Una vez que se encuentra seleccionado un club, toma el rol que se le designó previamente dentro de este. Por lo cual, el usuario general puede poseer distintos roles en cada uno de los clubes que pertenezca.

Para ser administrador de un club, el usuario debió haber creado el club previamente. Por otro lado, para asumir el rol de deportista o entrenador de un equipo, es necesario que el administrador del club lo acepte y asigne previamente.

3.2.1. Requerimientos Funcionales

Los requerimientos funcionales fueron definidos en base a la experiencia del memorista en clubes deportivos, complementada con reuniones y aporte de los miembros del Club Deportivo Mewlen Angol. Particularmente, se enfocaron en los requerimientos relacionados sobre la gestión de asistencia y la creación de eventos, dado que estas áreas presentaban mayor problemática a la hora de organizar las actividades a realizar en cada entrenamiento.

Una vez obtenidos estos requerimientos, se agruparon considerando aquellos requerimientos que están asociados con funcionalidades dentro de un club y/o equipo y los que no. Por lo tanto, los requerimientos funcionales cuando el usuario se comporta como un usuario general sin un rol específico son:

- Registrarse en la plataforma.
- Acceder con credenciales.

- Explorar clubes deportivos cercanos, filtrar por los deportes de interés y obtener información de ellos.
- Crear clubes deportivos.
- Editar información del perfil propio.
- Solicitar la integración a clubes deportivos.

Por otro lado, cuando el usuario se encuentra dentro de un club y/o equipo dispondrá de alguno de los roles señalados anteriormente. A continuación, se presenta una tabla que detalla las funcionalidades asociadas a cada rol.

La plataforma permitirá...	Deportista	Entrenador	Administrador
Gestionar club			✓
Gestionar eventos		✓	✓
Registrar asistencia a eventos	✓	✓	
Visualizar eventos activos con miembros asistentes	✓	✓	✓
Visualizar estadísticas de asistencia de los miembros de un equipo		✓	✓
Visualizar estadísticas propias de asistencia	✓	✓	✓
Visualizar estadísticas de asistencias por equipo		✓	✓
Gestionar solicitudes de unión			✓
Visualizar miembros por equipo		✓	✓
Gestionar miembros de clubes y equipos			✓
Recibir notificaciones acerca de la creación de eventos	✓	✓	✓
Recibir notificaciones acerca de nuevas solicitudes de unión			✓
Modificar lista de asistentes		✓	✓

Cuadro 3.1: Requerimientos funcionales por rol correspondiente

Para representar los requerimientos funcionales y cómo los usuarios (actores) interactúan con el sistema se utilizará un diagrama de casos de uso, dado que provee una descripción del sistema de fácil comprensión. En la Figura 3.1 se puede apreciar el diagrama de casos de uso.

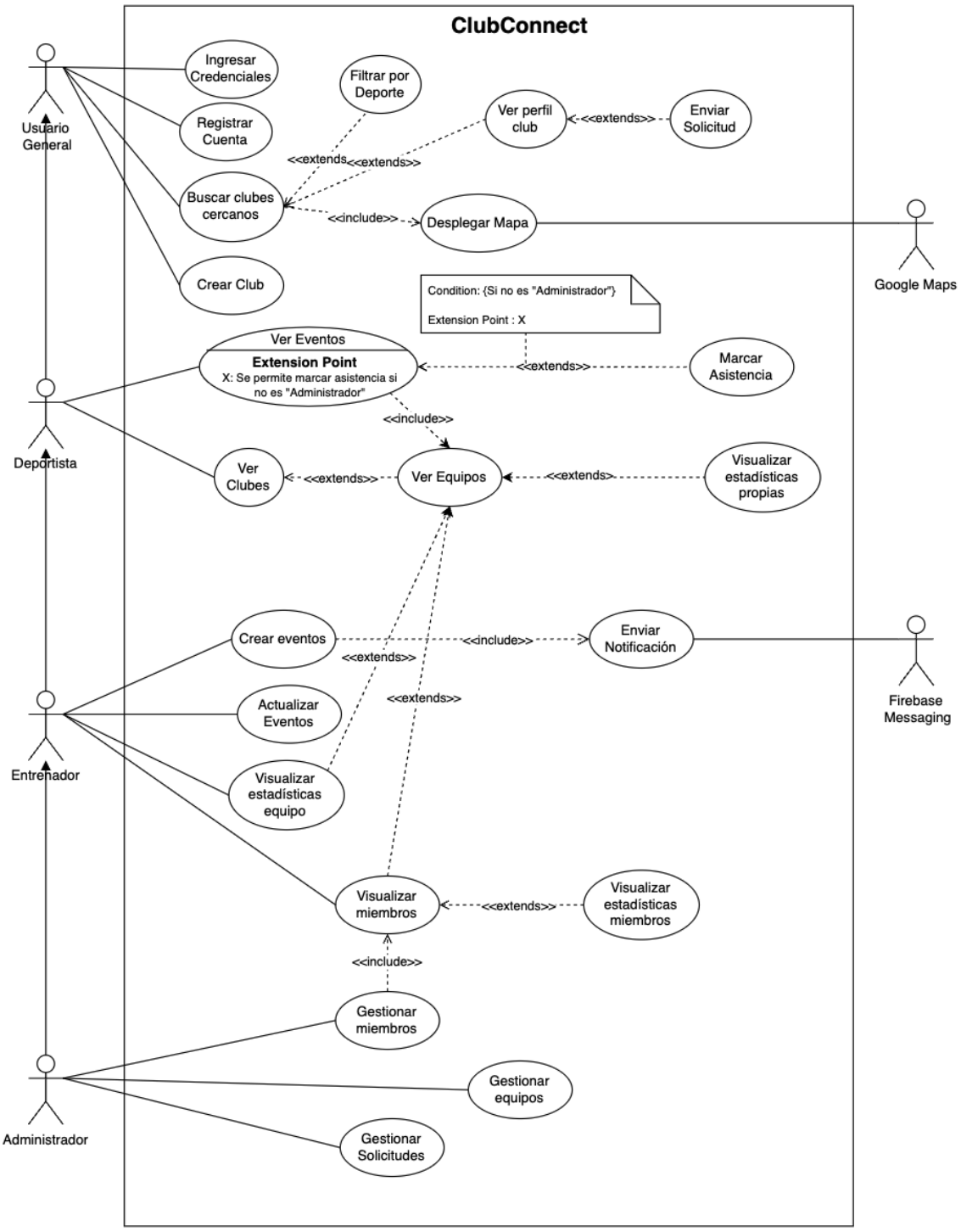

Figura 3.1: Diagrama de Casos de Uso

3.2.2. Requerimientos No Funcionales

- **Rendimiento:** Teniendo como cantidad inicial 750 usuarios, es decir, 25 clubes con un promedio de 30 usuarios cada uno, el sistema debe ser capaz de manejar la cantidad de usuarios, garantizando tiempos de respuesta inferiores a 4 segundos.
- **Seguridad:** El sistema debe implementar medidas de seguridad para proteger la integridad y la confidencialidad de los datos, especialmente la información de los usuarios. Por ejemplo, verificando la autenticación de los usuarios.
- **Usabilidad:** El sistema debe tener una interfaz de usuario intuitiva y sencilla, con el fin de brindar una fluida y grata experiencia a los usuarios.
- **Escalabilidad:** Se espera que el sistema reciba actualizaciones periódicamente durante el tiempo y aumento en su carga de trabajo. Por lo tanto, el sistema debe ser capaz de adaptarse a los cambios de manera sencilla.

4. Diseño y Arquitectura

La arquitectura en la que se construyó la aplicación sigue el estilo arquitectónico cliente - servidor, uno de los más utilizados en la industria de software. Esta arquitectura permite una clara separación de responsabilidades entre el cliente y el servidor, lo que facilita la administración de los datos, mejora la seguridad de la información, dado que se encuentra en el entorno del servidor, brinda escalabilidad y facilidad de mantenimiento para la aplicación [5]. A continuación se muestra el modelo C4 del proyecto, abarcando sus tres niveles principales: diagrama de contexto del sistema, diagrama de contenedores y diagrama de componentes.

El diagrama de contexto del sistema (4.1), ilustra las personas que interactúan con *ClubConnect* y cómo este se relaciona con los sistemas externos. Entre estos sistemas se encuentran *Firebase Messaging*, utilizado para implementar la funcionalidad de envío de notificaciones y *Google Maps* para el despliegue de mapa y descubrimiento de clubes mediante geolocalización.

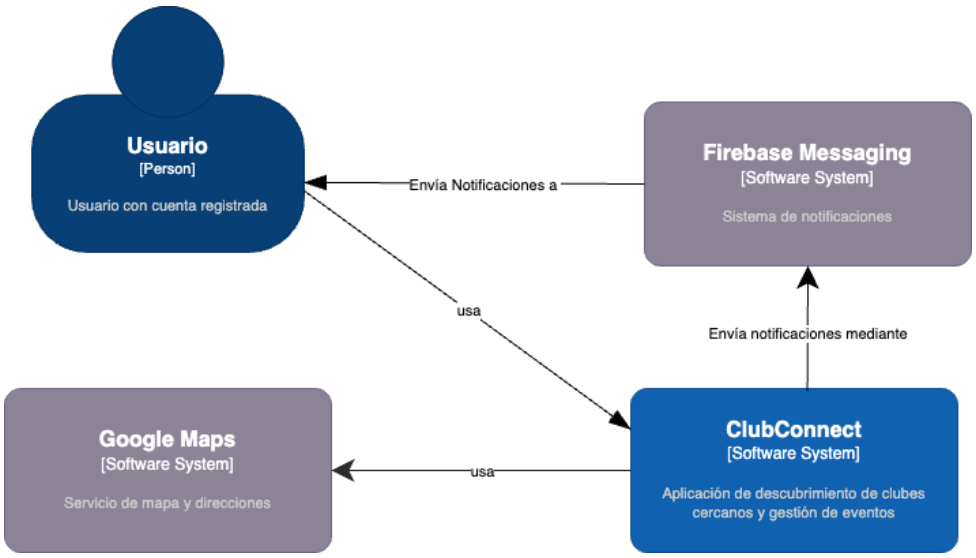

Figura 4.1: Diagrama de Contexto

El diagrama de contenedor (4.2), representa que la aplicación se compone de 3 contenedores, una aplicación móvil, una API del lado del servidor y una base de datos. Para la aplicación móvil, se optó por utilizar Flutter como framework debido al conocimiento previo, su flexibilidad y el soporte multiplataforma que ofrece. Esta aplicación se comunica con la API Aplicación mediante solicitudes HTTP.

Para la API del lado del servidor, se decidió por usar node.js debido a su alta escalabilidad y eficacia en el manejo de solicitudes simultáneas. Además, se integró el framework Express, que simplifica el manejo de solicitudes.

Finalmente, la API obtiene información de la base de datos mediante consultas SQL. La base de datos se encuentra alojada en Supabase, que ofrece servicios de alojamiento de base de datos y utiliza PostgreSQL como motor de base de datos.

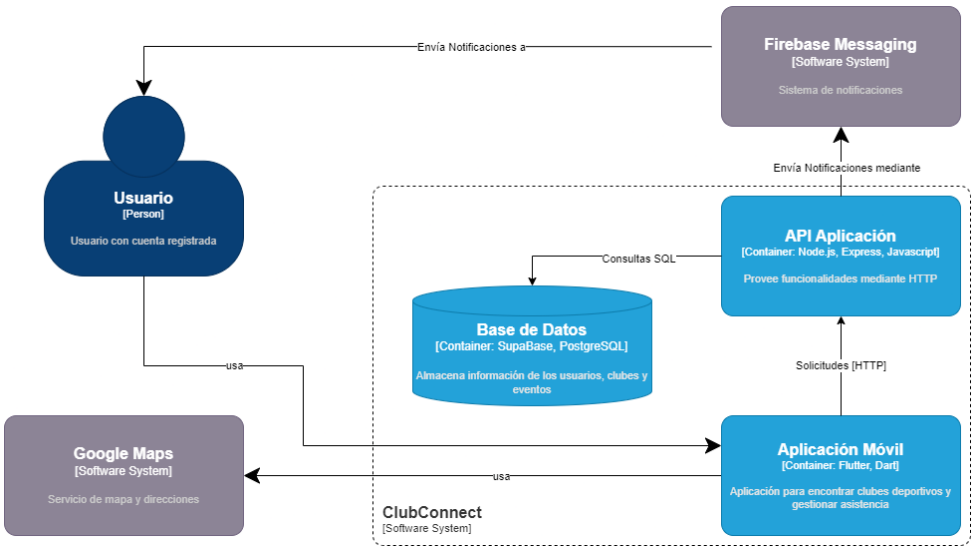

Figura 4.2: Diagrama del Contenedor

El diagrama de componentes (4.3) representa la estructura modular del backend, diseñada para facilitar el mantenimiento y escalabilidad. La API Aplicación está compuesta de 3 componentes principales, los cuales se describen a continuación :

- **Routes (Rutas) :** Este componente actúa como la capa de entrada para las solicitudes HTTP. Aquí se define las rutas, las cuales mapean las solicitudes hacia el controlador correspondiente, garantizando una estructura organizada.
- **Controllers (Controladores) :** Los controladores son responsables de procesar la lógica de negocio de las solicitudes recibidas desde las rutas, realizando operaciones y gestionando la interacción con la base de datos.
- **Test (Pruebas) :** Este componente abarca las pruebas de integración automatizadas de las rutas, con el objetivo de garantizar que las rutas y los controladores se comuniquen correctamente. Se optó por usar Jest como framework de pruebas junto con Supertest, para realizar la simulación de las solicitudes HTTP.

Por ejemplo, se tiene la ruta **GET** /eventos, definida en el archivo *eventos.js* , la cual utiliza el método *getEventos* del controlador definido en *eventosControllers.js*. Este controlador es el responsable de implementar la lógica necesaria para obtener los eventos, considerando los parámetros proporcionados en la solicitud.

Adicionalmente, la ruta **GET** /eventos cuenta con una prueba de integración definida en un archivo *eventos_ asistencia.spec.js*. Esta prueba garantiza la correcta comunicación entre componentes y verifica que la respuesta obtenida cumpla con lo esperado.

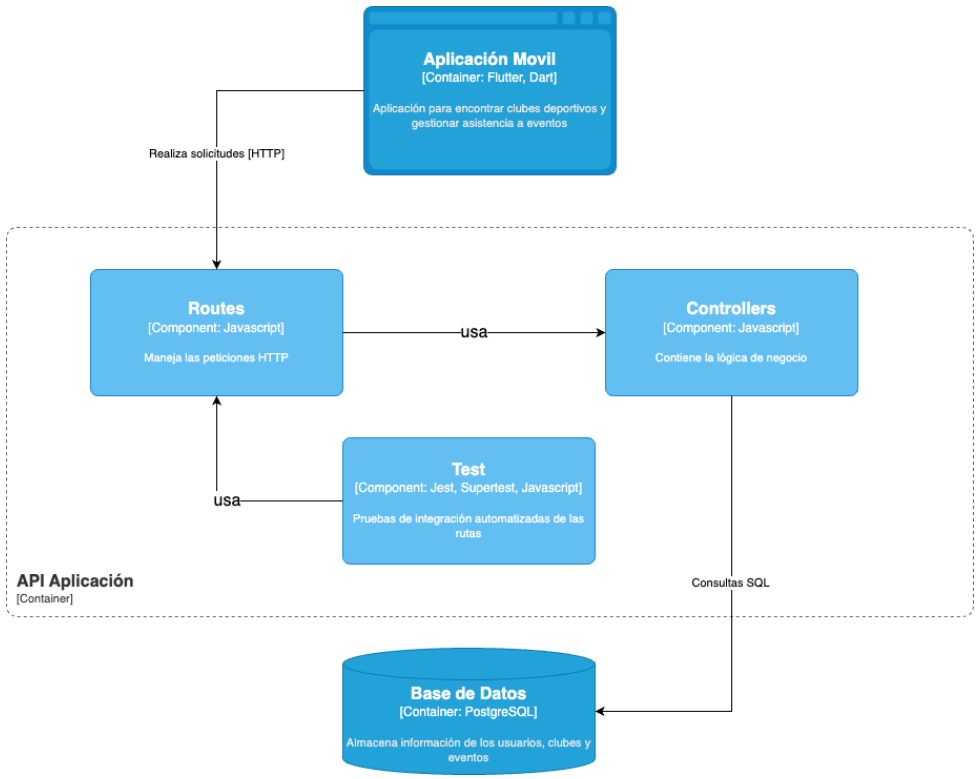

Figura 4.3: Diagrama de Componentes Backend

El diagrama de componentes (4.4) representa la estructura del frontend, una estructura basada en capas. La cual está compuesta de los siguientes 4 componentes:

- **Datasource (Fuente de Datos) :** Este componente se encarga de comunicarse con la API (Componente API Aplicación).
- **Repositories (Repositorios) :** Este componente actúa como una capa intermedia entre la fuente de datos y la capa de presentación. Siendo responsable de proveer los métodos para ser utilizados y obtener los datos necesarios en la interfaz.
- **Presentación :** Este componente representa la capa de la interfaz de usuario, que incluye, vistas, widgets y la lógica necesaria para interactuar con los usuarios. Es el responsable de renderizar los datos obtenidos desde los repositorios.
- **Modelos :** Los modelos aseguran la consistencia de la estructura de los datos entre los componentes.

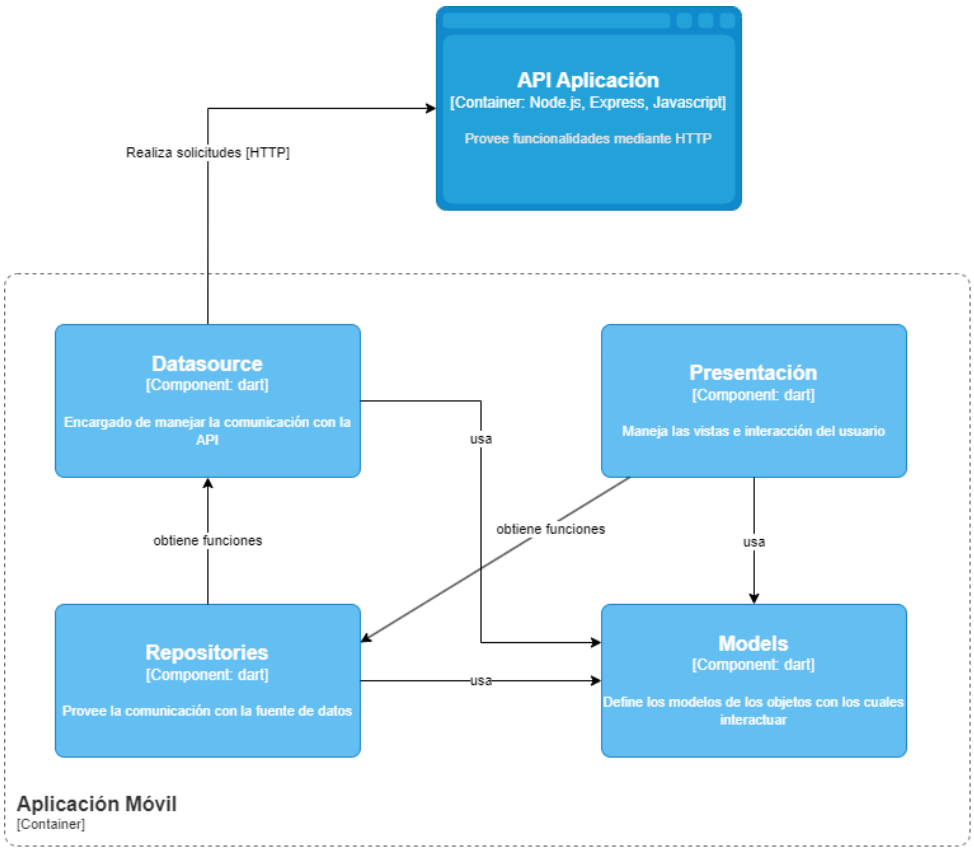

Figura 4.4: Diagrama de Componentes Frontend

5. Desarrollo

5.1. Metodología

Para el desarrollo de la presente memoria de título, se optó por aplicar un enfoque de trabajo basado en el desarrollo iterativo con el propósito de estructurar el desarrollo del proyecto de manera ordenada. Durante cada iteración se definieron las funcionalidades a desarrollar o mejorar, priorizándolas según su nivel de importancia, aplicando las pruebas de integración correspondientes.

Al término de cada iteración, se realizaba una reunión con el profesor patrocinante Geoffrey Hecht, con el objetivo de presentar avances y recibir retroalimentación.

Iteración	Tareas a implementar y/o modificar
Iteración 1 06/05/2024 - 27/05/2024	▪ Crear clubes. ▪ Mapa de exploración de clubes. ▪ Registrar cuenta. ▪ Autenticación (Inicio Sesión).
Iteración 2 27/05/2024 - 17/06/2024	▪ Crear equipos dentro del club. ▪ Solicitudes de afiliación a clubes. ▪ Visualizar solicitudes, cancelar o aceptar y asignar usuario a un equipo . ▪ Mapa de exploración de clubes.
Iteración 3 17/05/2024 - 08/07/2024	▪ Operaciones CRUD sobre los eventos. ▪ Confirmar y/o cancelar asistencia. ▪ Filtrar clubes por deporte. ▪ Visualizar miembros y gestionar su rol en equipos. ▪ Sistema de cluster en el mapa de exploración.
Iteración 4 08/07/2024 - 29/07/2024	▪ Notificación de creación de eventos. ▪ Modificar interfaz de visualizar eventos. ▪ Estadísticas de miembros. ▪ Agregar redes sociales a los clubes.
Iteración 5 29/07/2024 - 19/08/2024	▪ Eventos Recurrentes. ▪ Estadísticas de miembros. ▪ Finalizar eventos automáticamente.
Iteración 6 19/08/2024 - 16/09/2024	▪ Estadísticas por equipo ▪ Notificación solicitud de afiliación. ▪ Agregar Icono aplicación.

Cuadro 5.1: Iteraciones del proceso de desarrollo

5.2. Tecnologías Utilizadas

En este apartado se describen brevemente las principales tecnologías y herramientas utilizadas durante el proceso de desarrollo del proyecto, junto con la relevancia de cada una de ellas:

- **Flutter :**
  - ¿Qué es?: Es un framework de código abierto para crear aplicaciones

multiplataformas mediante el lenguaje Dart [1].

- Relevancia: Es altamente relevante debido a que fue el framework utilizado para la creación del frontend del proyecto, fundamental para crear una aplicación de manera rápida y sencilla adaptable a diferentes plataformas con interfaces personalizadas y llamativas.

- **Supabase :**
  - ¿Qué es?: Es una plataforma de desarrollo de código abierto que ofrece una serie de herramientas y servicios, entre ellos un sistema de gestión de base de datos relacional sólido y escalable para manejar conjunto de datos de cualquier tamaño, basado en PostgreSQL [3].
  - Relevancia: Fue relevante para alojar la base de datos en la nube, permitiendo realizar operaciones mediante su integración con node.js.
- **GitHub :**
  - ¿Qué es?: Es una plataforma de gestión y organización de proyectos basada en la nube que incorpora las funciones de control de versiones de Git permitiendo a los desarrolladores trabajar en proyectos simultáneamente.
  - Relevancia: Es relevante dado que en proceso del desarrollo del proyecto, se fueron almacenando los avances del código fuente [2], facilitando el seguimiento de los cambios.
- **Figma :**
  - ¿Qué es?: Es un editor de gráficos y una herramienta para generar prototipos.
  - Relevancia: Se utilizó al comienzo del proyecto para generar el prototipo de lo que sería la interfaz de usuario.
- **Node.js & Express:**
  - ¿Qué es?: Node js es un entorno de ejecución basado en JavaScript para la capa del servidor, y Express que es un framework que ofrece una estructura clara y herramienta para manejar solicitudes HTTP.
  - Relevancia: Es muy relevante dado que mediante estas tecnologías está construido el backend del proyecto, permitiendo la gestión de datos,

autenticación y comunicación entre el servidor y el cliente.

- **Jest :**
  - ¿Qué es?: Es un framework de automatización de prueba unitarias y de integración basada en JavaScript.
  - Relevancia: Con el fin de realizar pruebas de integración automatizadas se utilizó este framework para prevenir errores de manera temprana.
- **Render:**
  - ¿Qué es?: Es una plataforma de la nube que ofrece servicios de alojamiento y despliegue de aplicaciones web, aplicaciones estáticas y funciones serverless.
  - Relevancia: Fue relevante para alojar la API y realizar solicitudes al servidor en dispositivos físicos al momento de llevar a cabo la validación con usuarios.

5.3. Base de Datos

Para la creación de la base de datos se diseñó el diagrama MER[1] de la figura 5.1 para modelar como las entidades se relacionan entre sí.

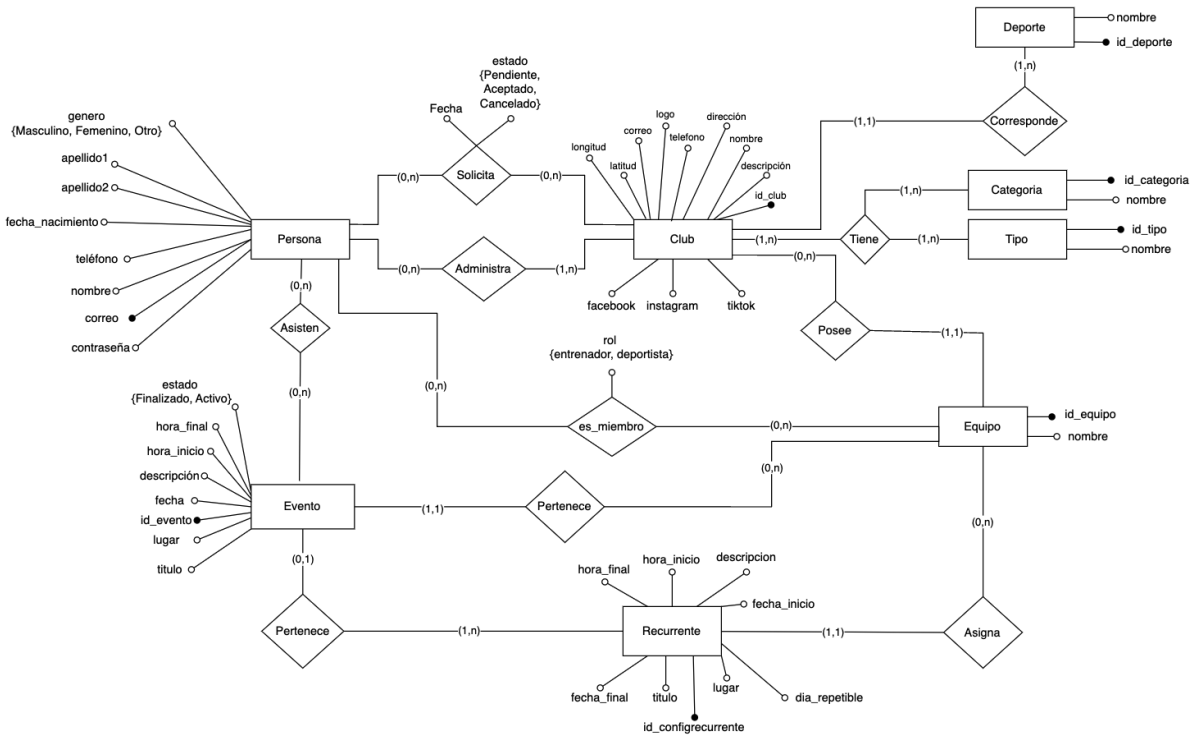

Figura 5.1: Modelo Entidad Relación

[1] Modelo Entidad Relación

El diagrama de la figura 5.2 fue creado con la herramienta db.diagrama [7] desde la importación directa del archivo .sql de la base de datos desplegada en Supabase.

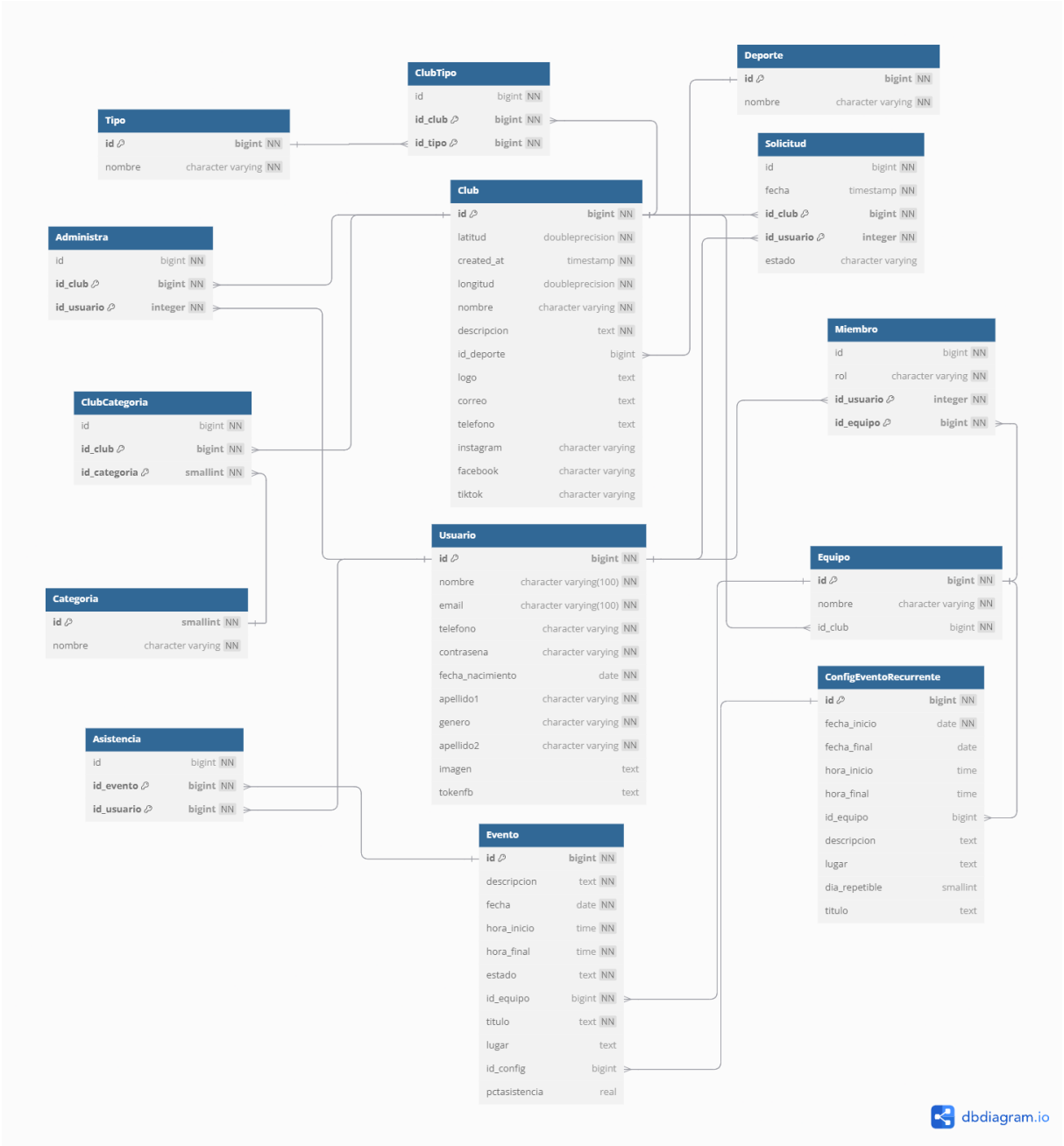

Figura 5.2: MR generado por dbdiagram.io

A continuación se describirán las entidades y relaciones del modelo :

- **Club** : Entidad encargada de almacenar la información de los clubes.
  - **Relaciones :**
    - **id_deporte**: Hace referencia al deporte, enlaza al club con un deporte específico.

- **Deporte** : Entidad encargada de almacenar los distintos deportes disponibles.
- **Tipo**: Entidad que contiene los distintos tipos de clubes, ya sean estos "Formativos", "Competitivos" y/o "Recreativos".
- **Club_Tipo**: Entidad encargada de almacenar los tipos a los cual está asociado un Club.
    - **Relaciones :**
        - **id_club**: Hace referencia a un club.
        - **id_tipo**: Hace referencia a un tipo, al cual está asociado un club.
- **Categoria**: Entidad encargada de almacenar las categorías deportivas que pueden haber en un club. Con categoría se hace referencia a Sub-8, Sub-18, todo competidor, etc.
- **Club_Categoria**: Entidad encargada de almacenar a los clubes con sus categorías correspondientes.
    - **Relaciones :**
        - **id_club**: Hace referencia a un club.
        - **id_categoria**: Hace referencia a una categoría.
- **Usuario** : Entidad encargada de almacenar información de los usuarios registrados.
- **Administra**: Entidad encargada de almacenar a todos los administradores de cada club. Se ingresa una nueva fila automáticamente cada vez que se crea un club enlazando al usuario con el club creado.
    - **Relaciones :**
        - **id_club**: Hace referencia a un club.
        - **id_usuario**: Usuario que administra el club.
- **Solicitud**: Entidad encargada de almacenar las solicitudes de los usuarios. Cabe destacar que incluye un atributo "estado", que indica si la solicitud está "Pendiente", "Aceptada" o "Cancelada".
    - **Relaciones :**

- **id_club**: Hace referencia a un club.
- **id_usuario**: Usuario que envió una solicitud de afiliación al club.

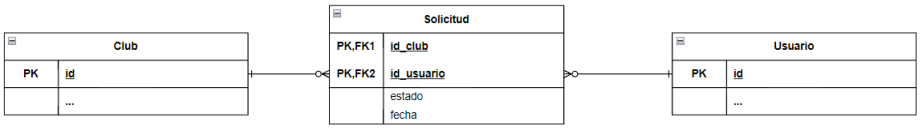

Figura 5.3: Relación Solicitud

- **Equipo**: Entidad encargada de almacenar los equipos que poseen los clubes.
  - **Relaciones :**
    - **id_club**: Hace referencia a un club, al que pertenece el equipo.
- **Miembro**: Entidad encargada de almacenar a los usuarios que pertenecen a un equipo específico. Cabe destacar que incluye un atributo "rol", que indica si el usuario es "Deportista" o "Entrenador" de un equipo.
  - **Relaciones :**
    - **id_usuario**: Hace referencia a un usuario.
    - **id_equipo**: Hace referencia al equipo al cual el usuario es miembro.
- **Evento**: Entidad encargada de almacenar los eventos. Cabe destacar, que los eventos solo pueden ser creados por administradores o entrenadores de cada club.
  - **Relaciones :**
    - **id_equipo**: Hace referencia al equipo para el cual se creó el evento.
    - **id_config**: Este atributo puede o no hacer referencia a una configuración de un evento recurrente. Con la finalidad de poder realizar acciones sobre estos eventos.
- **ConfigEventoRecurrente**: Entidad encargada de almacenar información sobre los eventos recurrentes que han sido creados por el administrador de un club o entrenadores de un equipo. Posee los mismos atributos que un evento normal, incluyendo atributos como la fecha de inicio, fecha fin y día en que se repetirá este evento.

- **Relaciones :**
  - **id_equipo**: Enlaza a la configuración de eventos recurrentes con el equipo que se referencia.

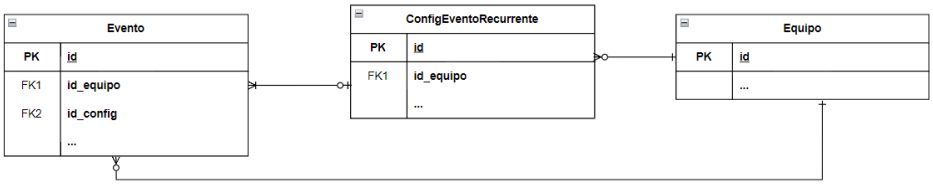

Figura 5.4: Relación entre eventos, equipos y configEventoRecurrente

- **Asistencia**: Entidad encargada de almacenar a los miembros de los clubes que marcan asistencia a los eventos.
  - **Relaciones :**
    - **id_evento**: Hace referencia al evento que realiza un equipo.
    - **id_usuario**: Hace referencia al usuario que confirmó que asistirá al evento.

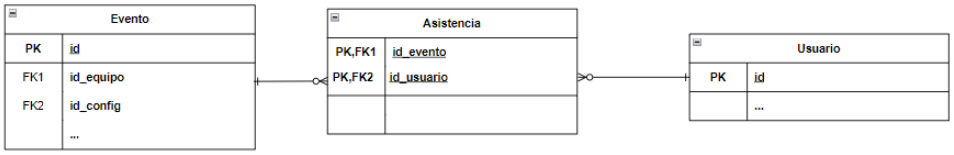

Figura 5.5: Relación Asistencia

Además, la base de datos cuenta con un trigger, que se encarga de calcular y actualizar el porcentaje de asistencia de un evento (atributo : pctasistencia) cada vez que se marca como finalizado. Los eventos se pueden finalizar de manera automática mediante "tareas programadas" que se ejecuta diariamente en el servidor. Esta tarea verifica si la fecha de los eventos es menor a la actual, en caso de cumplirse, el evento cambia su estado a finalizado.

5.4. Interfaz de Usuario

La interfaz de usuario de la aplicación mayoritariamente es la misma para administradores, entrenadores y/o deportistas. A excepción que en ocasiones existen rutas de acceso disponibles solo para distintos roles, las cuales se describirán a medida que se presentan las vistas que posee la aplicación.

5.4.1. Ingreso

Casos de usos relacionados: Ingresar credenciales

Esta vista está diseñada para que los usuarios ingresen sus credenciales y tengan acceso al sistema. En caso de autenticar al usuario este será dirigido al sistema principal. Si el usuario no posee una cuenta tendrá la opción de registrarse mediante la opción *"Registrarse"*.

¿Cómo funciona?. El usuario envía sus credenciales, y mediante el backend se verifica que estas sean correctas mediante JWT (JSON Web Token) y el hash que este realiza, entregando como respuesta el token de acceso e información del usuario.

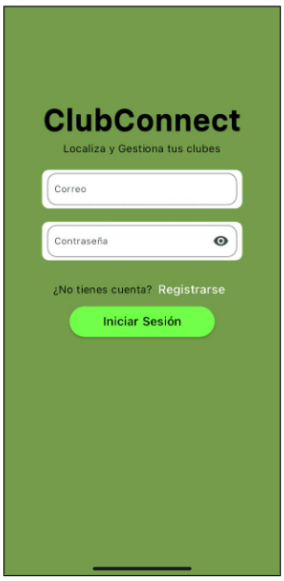

Figura 5.6: Vista del Ingreso

5.4.2. Registro Usuarios

Casos de usos relacionados: Registrar cuenta.

Esta vista tiene como objetivo permitir el registro de usuarios, para que puedan crear una cuenta y ser almacenados en la base de datos. De esta manera, los usuarios podrán acceder a la aplicación en futuros ingresos utilizando sus credenciales creadas previamente.

Figura 5.7: Vista para registro de usuarios.

5.4.3. Explorar Clubes

Casos de usos relacionados: Buscar clubes cercanos, filtrar por deporte.

Este aspecto fue diseñado para que los usuarios puedan encontrar clubes cercanos a su ubicación al momento de ingresar. Los usuarios contarán con la opción de filtrar estos clubes por deportes que estimen de interés propio, teniendo la posibilidad de encontrar información relevante de los clubes, como información de contacto, tipo de club, categorías en las cuales se trabaja en el club, ubicación del club y redes sociales. Además, los usuarios pueden enviar una solicitud de unión al club, la cual será gestionada por el administrador correspondiente.

¿Cómo funciona?. Para obtener los clubes desplegados en el mapa, se deben obtener previamente la ubicación del usuario, coordenadas del mapa y deportes de interés. En caso de querer buscar en otras coordenadas, se debe desplazar el mapa y presionar el botón *"Buscar aquí"*, lo cual hará una recarga de los clubes en el mapa. Además, cuenta con la funcionalidad de sistema de clúster para no sobrecargar la vista cuando existan clubes que se encuentren muy cercanos. Para visualizar información de los clubes se debe presionar sobre el club, luego presionar *"Ver"* y se redirigirá al perfil público del club.

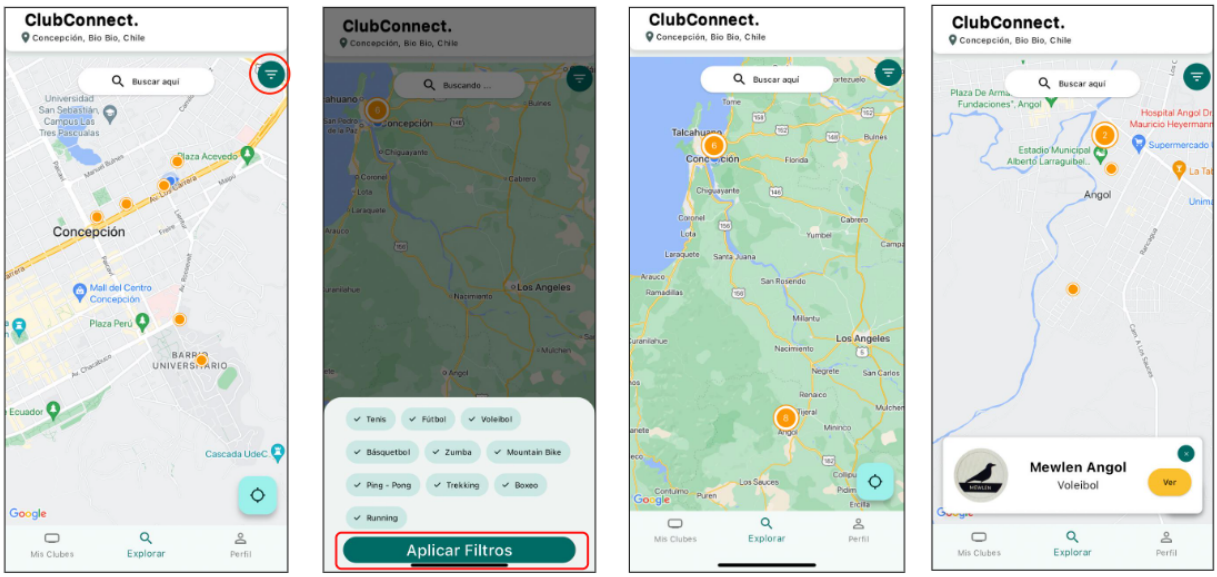

Figura 5.8: Vista para explorar clubes.

5.4.4. Perfil público Club

Casos de usos relacionados: Ver perfil club, enviar solicitud.

Esta vista tiene como objetivo desplegar la información del club, además de permitir el envío de solicitudes de unión. Como se aprecia en la figura 5.9, hay un botón posterior a la información desplegada, con el cual se puede enviar o cancelar una solicitud previamente enviada, la solicitud enviada contiene información del perfil del usuario, como información de contacto, nombre y fecha de nacimiento, como se puede apreciar en la figura 5.14. En caso de que el usuario sea administrador, este botón cambiará su etiqueta a *"Eres Administrador"*, o a *"Ya eres Miembro"* en caso de ser entrenador o deportista de algún equipo del club. Además, en el apartado de redes, estas cuentan con enlaces directos a los perfiles de las redes sociales que se hayan registrado.

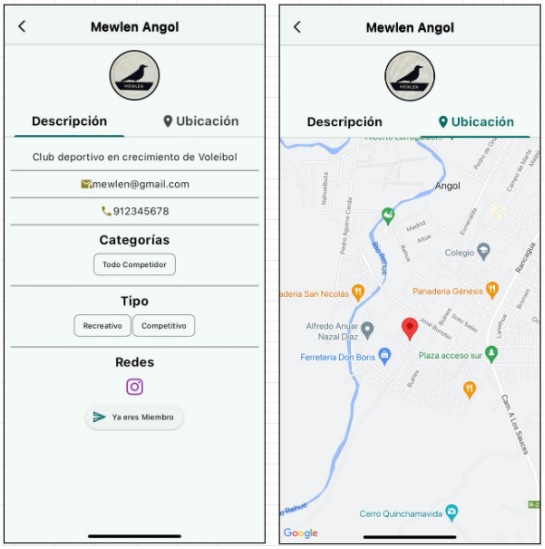

Figura 5.9: Vista del perfil público del club.

5.4.5. Clubes Pertenecientes

Casos de usos relacionados: Ver clubes.

En esta pantalla se mostrarán todos los clubes en los cuales el usuario ha sido aceptado con anterioridad. Dentro de esta vista se encuentran funcionalidades tales como, la opción de crear un nuevo club (5.4.6), visualizar la información principal del club (5.4.4), abandonar el club, y acceder a los equipos en los que el usuario es miembro dentro del club (5.4.7).

Figura 5.10: Vista lista de clubes enlazados

5.4.6. Registro Clubes

Casos de usos relacionados: Crear club.

Esta vista tiene como objetivo permitir el registro de clubes.

¿Cómo funciona?. El usuario debe completar todos los campos establecidos, luego presionar *"Crear Club"*, una vez que se crea el club se asocia automáticamente al usuario como administrador del club.

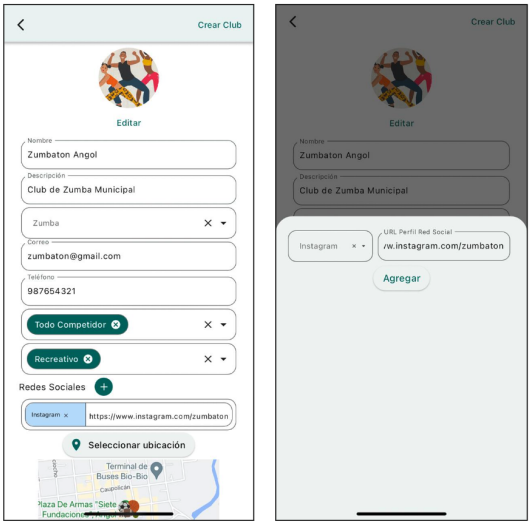

Figura 5.11: Vista Registro de Club.

5.4.7. Perfil privado de un club

Casos de usos relacionados: Ver equipos, gestionar miembros, visualizar miembros, gestionar solicitudes, gestionar equipos.

Esta vista cuenta con funcionalidades dependiendo del rol del usuario dentro del club. La vista principal corresponde a *"Todos los equipos"* , la cual despliega los equipos en los cuales el usuario es miembro, en caso de ser un administrador se despliegan todos los equipos para tener acceso a estos y/o eliminarlos. A continuación se describen las demás funcionalidades sobre un club a las que un usuario administrador tiene acceso:

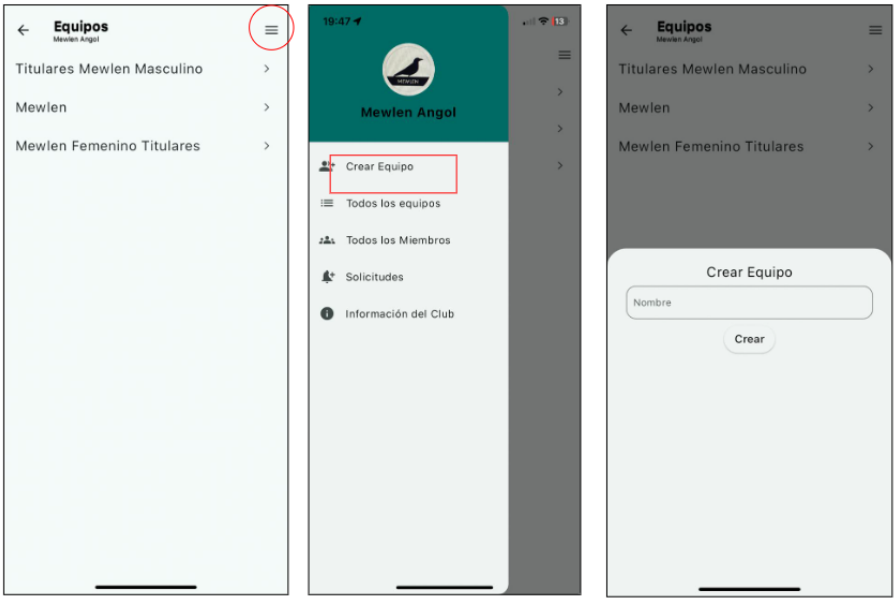

Figura 5.12: Vista de Equipos por club

- **Crear Equipo:** Crear Equipo: Esta funcionalidad despliega un modal en la parte inferior de la pantalla (Imagen 3 - 5.12), permitiendo crear un equipo dentro del club.
- **Todos los miembros:** Este apartado tiene como objetivo gestionar a los miembros del club. Las funcionalidades que contiene permiten gestionar a que equipos el usuario es miembro, teniendo la posibilidad de agregar al usuario a un nuevo equipo con su rol correspondiente, eliminar de un equipo y/o expulsar del club.

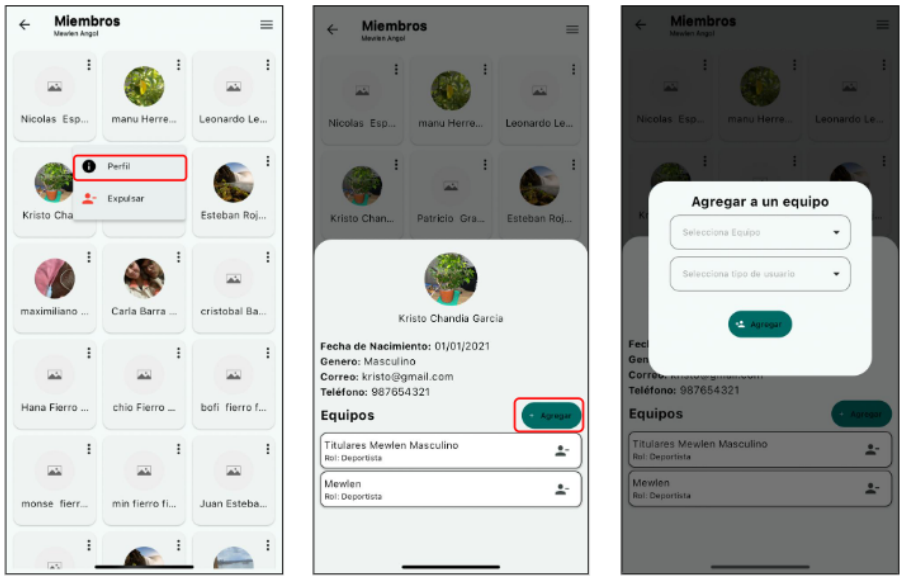

Figura 5.13: Vista de Miembros del Club

- **Todos los equipos:** Esta es la vista inicial (Imagen 1 - 5.12), que tiene funcionalidades sobre los equipos , como se describió previamente.
- **Solicitudes:** Esta vista tiene como objetivo desplegar todas las solicitudes de unión que se hayan realizado al club, estas solicitudes despliegan información personal del usuario, como contactos, nombre y fecha de nacimiento. Se podrá gestionar estas solicitudes, cancelarlas o aceptarlas. Si la solicitud es aceptada, se deberá asignar al usuario a los equipos que el administrador estime correspondiente.

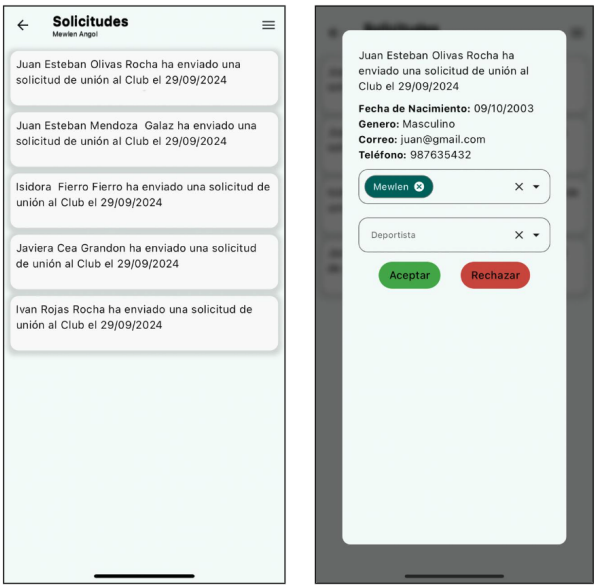

Figura 5.14: Vista de Solicitudes de unión al club

- **Información del Club:** Esta vista tiene como objetivo visualizar la información actual que se encuentra asociada al club con la funcionalidad de permitir editar.

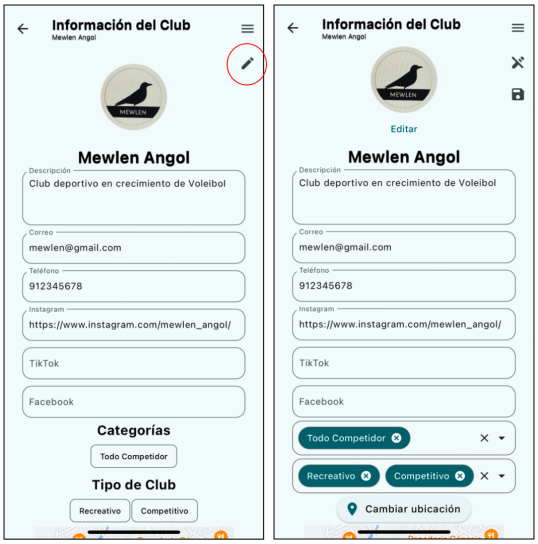

Figura 5.15: Vista perfil del club

5.4.8. Perfil privado de un equipo

Casos de usos relacionados: Crear eventos, actualizar eventos, visualizar estadísticas equipo, marcar asistencia, visualizar estadísticas propias, visualizar miembros, visualizar estadísticas miembros.

Esta vista tiene como objetivo implementar las funcionalidades relacionadas sobre un equipo específico. Cuenta como vista principal a *"Eventos Activos"* , la cual muestra los eventos que se encuentran activos del mes actual y los próximos 5 meses. Los deportistas y/o entrenadores podrán ver la información del evento, marcar asistencia y ver los usuarios que han marcado asistencia. Además, en caso de ser deportista este contará con un botón de estadísticas en la parte superior para que el deportista pueda visualizar su perfil de estadísticas (Vista Deportista, imagen 2, 5.16).

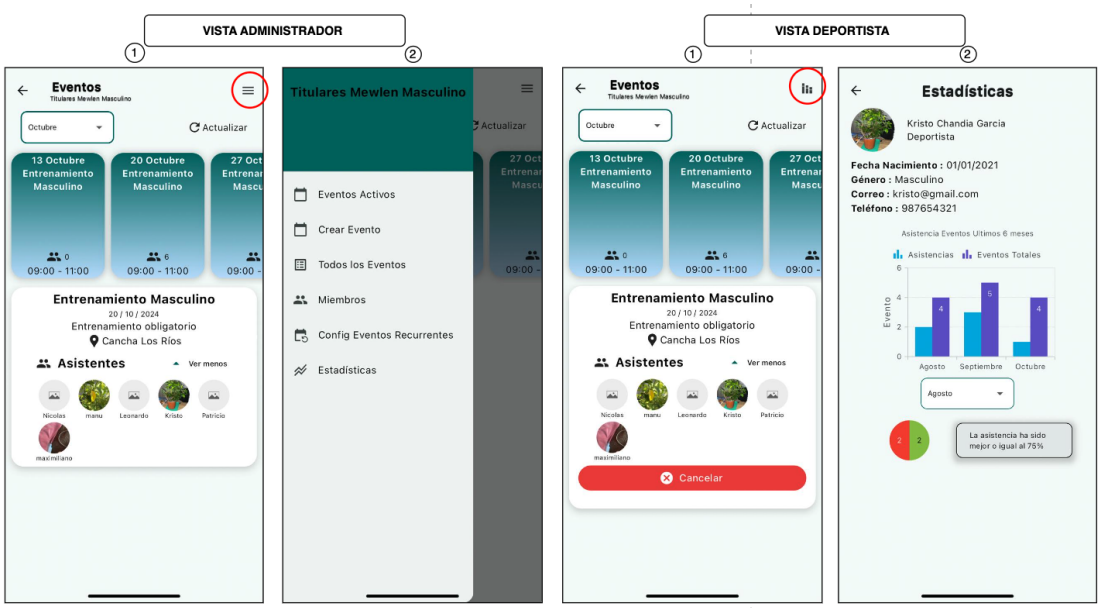

Figura 5.16: Vista Principal del equipo específico (Eventos Activos).

A continuación se describen las demás funcionalidades a las que un usuario administrador del club tiene acceso mediante el botón “3 Filas”:

- **Eventos Activos**: Esta es la vista inicial descrita anteriormente.
- **Crear Evento**: Esta vista tiene como objetivo implementar la funcionalidad de crear eventos. Estos eventos pueden ser creados individualmente o de manera múltiple, esto es en base a la cantidad de días que se seleccione.

Figura 5.17: Vista Crear evento individual o múltiples.

- **Todos los Eventos:** Esta vista tiene como objetivo que los administradores y/o entrenadores puedan visualizar todos los eventos en un mes y año señalado junto con su información y lista de asistentes. Además de implementar funcionalidades tales como, poder editar información de los eventos, lista de asistentes, estado (Activo o Terminado) y eliminar eventos.

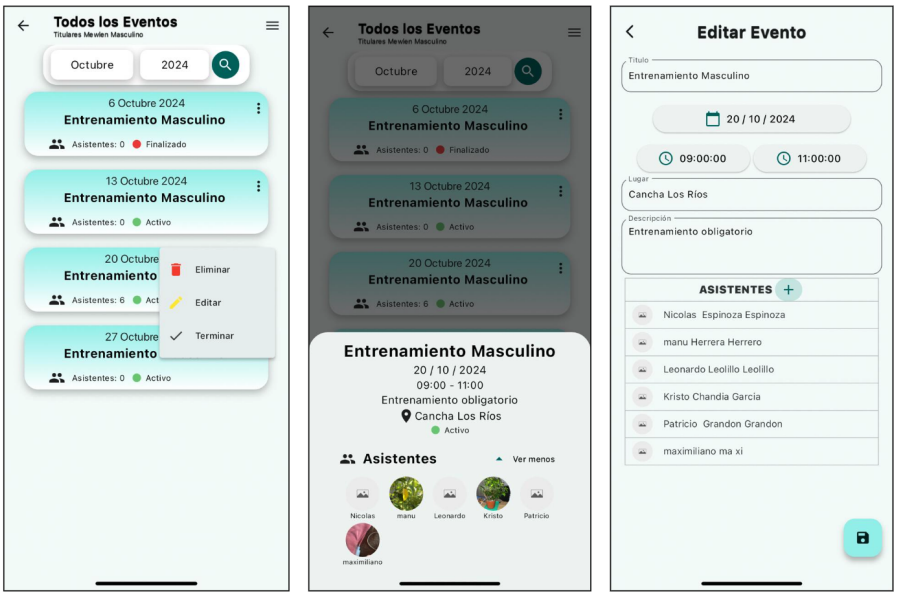

Figura 5.18: Vista todos los eventos.

- **Miembros**: El objetivo de esta vista es visualizar a todos los miembros del equipo junto con sus estadísticas de los últimos 6 meses. Estas estadísticas contemplan la cantidad de asistencias en comparación con la cantidad total de eventos por mes. Además, se muestra un indicador estadístico que muestra el porcentaje de participación en comparación con la del resto de los miembros. Esta información facilita el seguimiento de la participación de los miembros.

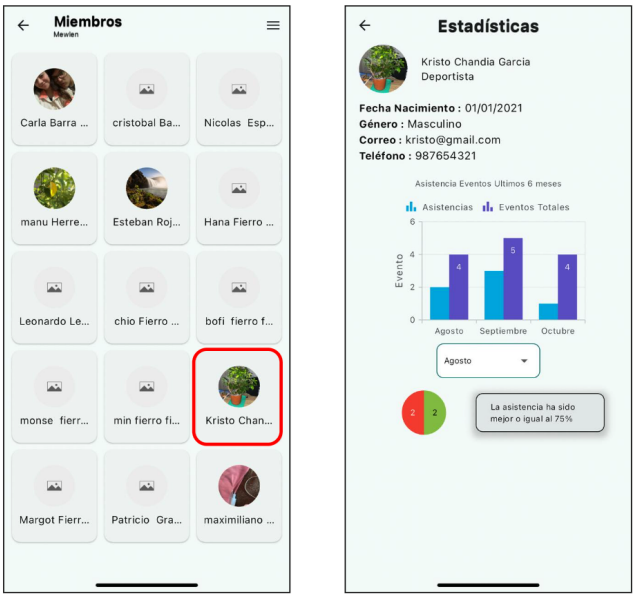

Figura 5.19: Vista miembros por equipo

- **Config Eventos Recurrentes**: Con el fin de facilitar la creación de este tipo de eventos, se consideró abordar esta funcionalidad donde el administrador y/o entrenador podrán crear configuraciones de eventos. Estas configuraciones incluyen la información principal del evento (señalada en apartados anteriores) junto con el día en que se repetirá este evento.

  ¿Cómo funciona?. Cuando el administrador o entrenador crea una configuración, se crearán automáticamente los eventos en el día señalado, desde la fecha inicial hasta la fecha final registrada. Esta configuración se puede editar o eliminar.

  En caso de editar la configuración (excepto día de repetición), los cambios sólo se aplicarán en los eventos que aún no hayan finalizado y que estén enlazadas con esta configuración a través del campo *"id_config"*. Si se modifica el día de repetición, se eliminarán los eventos que se encuentran en estado activo enlazados con la configuración y se vuelven a crear.

  En caso de eliminar la configuración, se eliminan automáticamente los eventos que se encuentran activos y enlazados a través del campo *"id_config"*, los eventos asociados que se encuentran en estado finalizado siguen intactos, para no afectar el sistema de estadísticas.

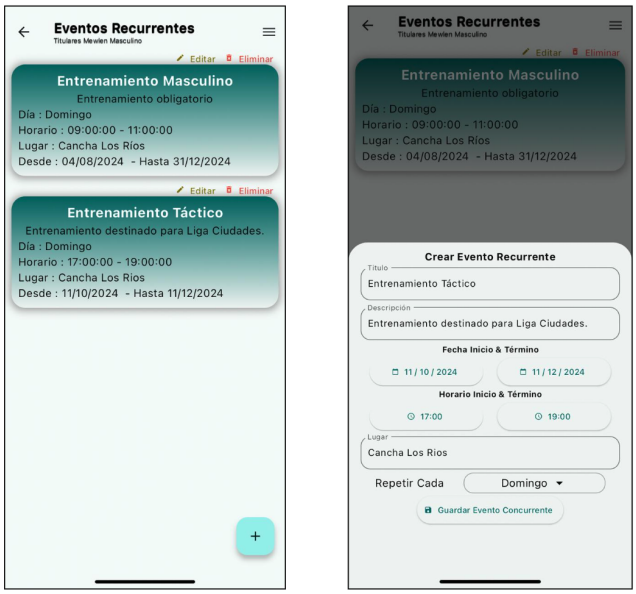

Figura 5.20: Vista configuración de eventos Recurrentes

- **Estadísticas:** El objetivo de esta pantalla es brindar estadísticas sobre la tendencia de asistencia a los eventos, lista de asistencia entre fechas determinadas e información de asistencia por cada evento.

  ¿Cómo funciona?. Para visualizar el gráfico que indica la tendencia de asistencia a los eventos, el usuario debe seleccionar una fecha inicial y una fecha límite. Una vez establecidas, se desplegará el gráfico con la información obtenida. Además, se permite filtrar los eventos por configuración, lo que facilita un análisis específico por configuración registrada.

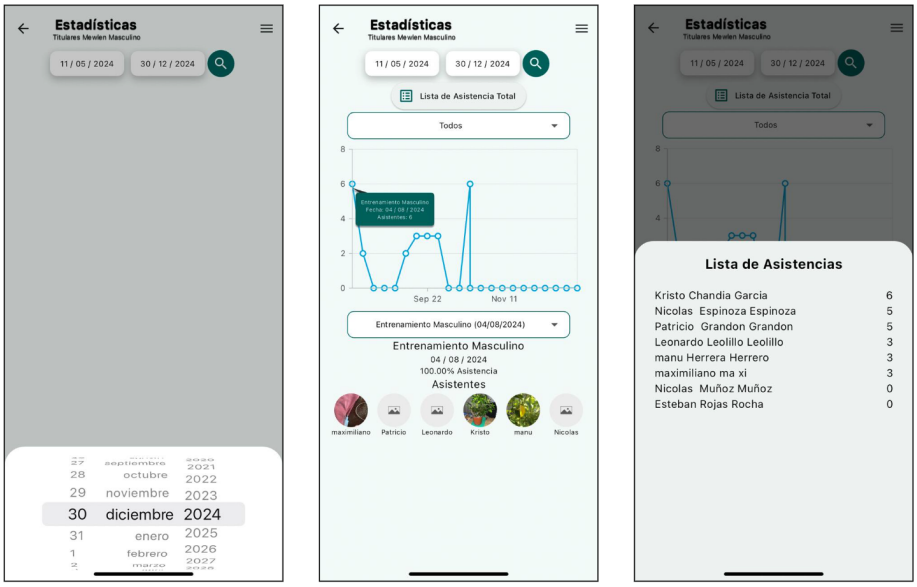

Figura 5.21: Vista estadísticas por equipo.

En el transcurso del desarrollo, se realizó un cambio de interfaz de las vistas “Estados Activos” (5.16) y “Todos los Eventos” (5.18), con el objetivo de mejorar la usabilidad. Este cambio se centró en crear una interfaz más intuitiva, atractiva y amigable para los usuarios, facilitando la interacción y experiencia del usuario.

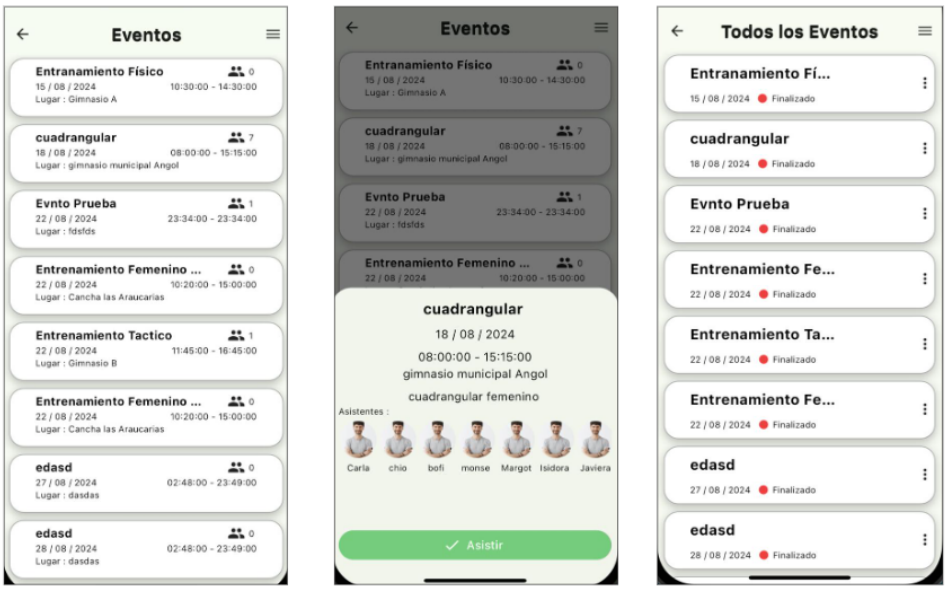

Figura 5.22: Interfaz Anterior de “Eventos Activos” y “Todos los Eventos”

5.4.9. Perfil

Esta pantalla presenta el perfil del usuario, donde se puede cerrar la sesión y visualizar la información de este presionando *“Información del Usuario”*. Además, también cuenta con la funcionalidad de poder editar la información del usuario.

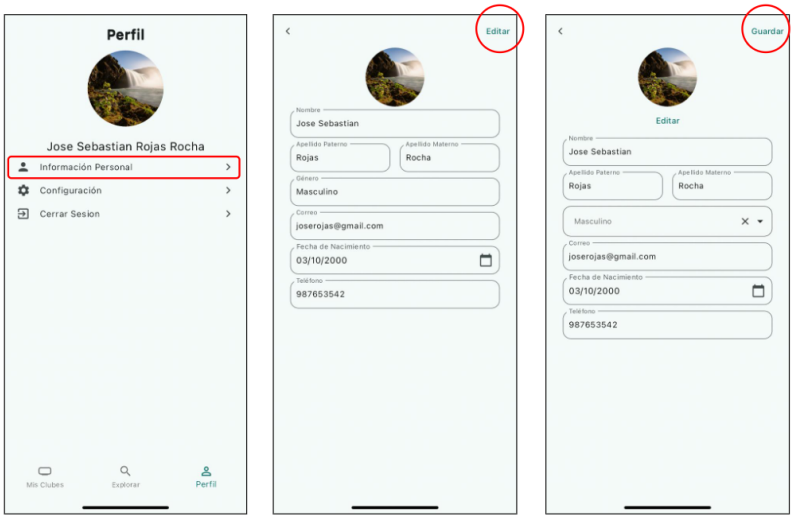

Figura 5.23: Vista perfil usuario

5.5. Sistema de Notificaciones

Para la integración del sistema de notificaciones se utilizó "Firebase Cloud Messaging" [9], dada su sencilla integración con sistemas Android e IOS. Actualmente, la versión de la aplicación solo incluye integración con Android, ya que la integración con IOS requiere de cuenta "Apple Developer". El envió de notificación es fundamental para mantener a los usuarios informados sobre la creación de nuevos eventos y a los administradores sobre las nuevas solicitudes de unión a los clubes.

¿Cómo funciona?. Cada usuario tiene un atributo *tokenfb*, el cual es el encargado de almacenar el token del dispositivo generado mediante la librería de "Firebase Cloud Messaging (FCM)" al iniciar sesión en un dispositivo. Este token permite gestionar el envío de notificaciones a los usuarios correspondientes.

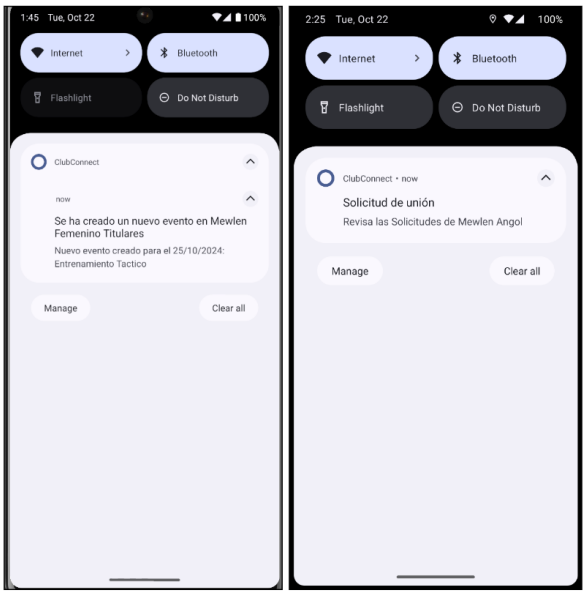

Figura 5.24: Notificaciones ClubConnect

5.6. Limitaciones

En esta sección se describen las principales limitaciones que presenta la aplicación, las cuales podrían afectar su funcionalidad, usabilidad o alcance.

- Falta de flexibilidad para clubes que incluyan múltiples deportes. Si bien, este caso se daría en clubes de mayor alcance y experiencia, esta carencia podría afectar a clubes deportivos que buscan centralizar la administración de todas sus disciplinas deportivas.

- Al aceptar un jugador, este debe ser agregado a un equipo en ese momento. Esta restricción podría generar inconvenientes, ya que el administrador podría no tener claridad inmediata sobre a qué equipo asignar a un nuevo jugador. Complicando la gestión de miembros, en clubes con múltiples equipos.
- Falta de funcionalidad para crear eventos generales para todos los miembros del club. Esta carencia puede ser suplida creando un equipo que incluya a todos los miembros del club. Sin embargo, esta solución alternativa puede no ser la ideal, debido a que resulta poco práctica y crear un equipo solo para este caso puede ser engorroso y afectar la experiencia del usuario.

6. Validación y Resultados

En este apartado se darán a conocer las pruebas de integración y validación de las funcionalidades hasta las opiniones de los usuarios objetivos.

En el ítem **5.1** se pueden visualizar las pruebas de integración de los endpoints de la API creada con Node JS [11], con el fin de prevenir y reducir errores de comunicación entre los distintos componentes.

En el ítem **5.2** se realiza la validación con usuarios mediante SUS [8] (System Usability Scale) y otras preguntas añadidas a la encuesta, con el fin de obtener una retroalimentación y verificar que se han cumplido los objetivos propuestos.

6.1. Pruebas de Integración

Con el objetivo de validar el correcto funcionamiento entre componentes, se han realizado las pruebas de integración con "Jest"[10] en conjunto con la biblioteca "SuperTest", la cual es una biblioteca de node.js que facilita la simulación de solicitudes HTTP. Por lo que, se utilizó de esta para realizar las solicitudes HTTP a la API desarrollada y validar que las respuestas del servidor sean las esperadas.

A continuación se detallarán los test de integración realizados en los distintos archivos *.spec.js* (Ejemplo en los anexos, 9.1):

- **Auth:** Este archivo tiene como objetivo probar las solicitudes que se ejecutan para que un usuario pueda ingresar a la aplicación. Estas solicitudes son *POST /login*,

que verifica la autenticidad del usuario, y *PATCH /token*, que actualizar el token del dispositivo para el envió de notificaciones mediante firebase messaging.

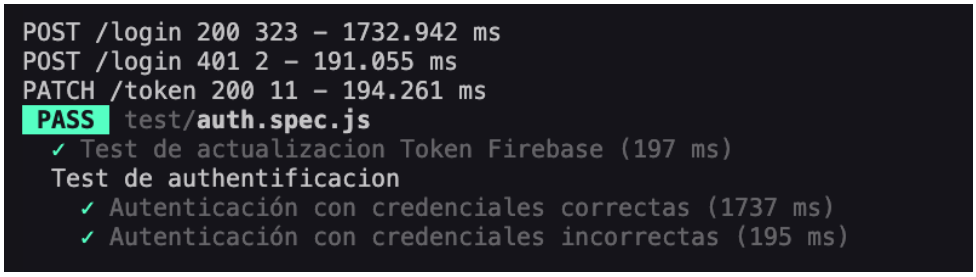

Figura 6.1: Resultados *auth.spec.js*

Las resultados de pruebas de integración definidas en el archivo auth.spec.js validan que la respuesta de las solicitudes contiene el código de estado esperado, ya sea 200 o 401, según corresponda.

- **Predefinidos:** Este archivo tiene como objetivo probar las consultas que traen datos preestablecidos para la aplicación, tales como, los tipo de clubes que pueden existir, todas las categorías disponibles al momento de la creación de clubes y los deportes predefinidos.

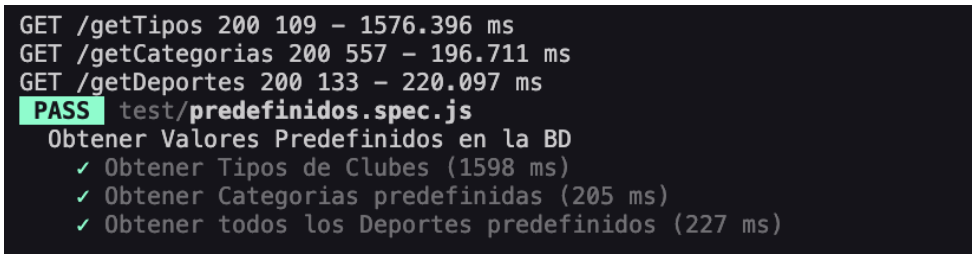

Figura 6.2: Resultados *predefinidos.spec.js*

Se valida que las pruebas realizadas en el archivo *predefinidos.spec.js* cumplen con los resultados esperados, lo que garantiza que la respuesta sea un arreglo de elementos con un tamaño superior a 0. Esto asegura que las operaciones sobre la base de datos se ejecutan correctamente.

- **Club:** En este archivo, se busca cubrir las principales solicitudes relacionadas sobre un club. Estas solicitudes corresponden a:

  - *GET /club/getclubs:* Obtiene los clubes que se encuentran dentro de los límites de coordenadas del mapa de búsqueda y que pertenezcan a los deportes filtrados por el usuario. Se entregan coordenadas donde existen clubes, por lo que se

espera que la respuesta obtenida tenga un código de estado 200 y sea un arreglo de elementos de tamaño superior a 0.

- *POST /club:* Crea un club utilizando los valores proporcionados en el cuerpo de la solicitud. Se utilizan tanto datos válidos como datos no válidos. Se espera que la respuesta contenga el código esperado, 200 o 400 según sea el caso.
- *GET /club/getclub:* Obtiene información de un club dado un id correspondiente. Se utiliza tanto un id no válido como uno válido, que corresponde al club creado previamente. Se espera que se entregue el código de estado correspondiente, ya sea 200 o 400, según sea el caso. Además, en el caso de ser un id válido, se espera que los datos entregados correspondan a un objeto.
- *GET /club/getmiembros:* Obtiene los usuarios que son miembros de un club. Se entrega como parámetro de búsqueda un club válido con miembros, por lo que se espera que en la respuesta incluya un código de estado 200, junto con un arreglo de elementos con tamaño superior a 0.
- *PUT /club/editClub:* Realiza la edición de atributos de un club. Se espera que, al editar un club válido, la respuesta entregue un código de estado 200 y el mensaje *"Club actualizado con éxito"*.

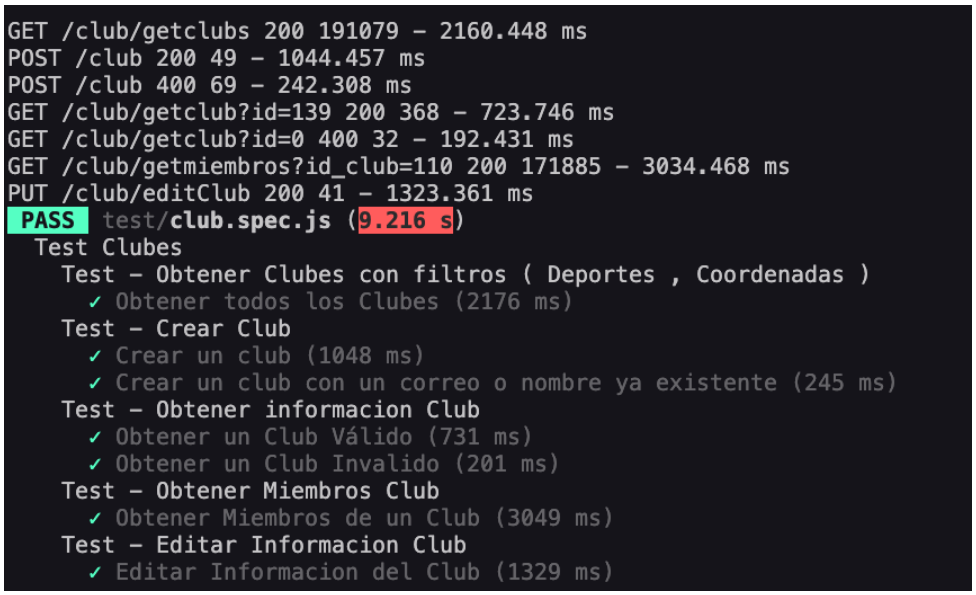

Figura 6.3: Resultados *club.spec.js*

- **Usuarios:** En este archivo se tiene como objetivo realizar las pruebas de integración sobre las principales solicitudes que tienen relación con los usuarios.

- *POST /usuarios/create:* Registra un usuario utilizando los datos proporcionados en el cuerpo de la solicitud. Para la prueba, se utilizaron tanto datos válidos como no válidos. Según la validez de los datos, se espera que la respuesta contenga un código de estado 201 o 400.

- *GET /usuarios/getUser:* Obtiene información de un usuario. Se entrega como valores de prueba un id no válido y el id del usuario creado previamente. Se espera que la respuesta contenga un código de estado 200 o 400 y el mensaje *"Usuario no encontrado"* en caso de no ser un id válido.

- *GET /usuarios/rol:* Obtiene el rol que cumple un usuario en un club y/o equipo. Para realizar la prueba de esta solicitud se prueban 3 casos, un usuario administrador, deportista y un id no válido, por lo que se espera que la respuesta entregue el rol del usuario, *"Administrador"*, *"Deportista"* o *"No existe información"* en caso de que el id no sea válido.

- *GET /usuarios/getclubesUser:* Obtiene la lista de clubes a los que un usuario pertenece. Se entrega como casos de prueba un usuario con clubes y un usuario no existente o sin clubes, por lo que, se espera como respuesta para un usuario con clubes un arreglo de tamaño superior a 0, en caso contrario un arreglo vacío.

- *GET /usuarios/stadistic:* Obtiene estadísticas de un usuario en un equipo. Se entrega como casos de prueba un usuario con estadísticas asociadas y un usuario no existente o sin estadísticas, por lo que se espera, como respuesta un arreglo de tamaño superior a 0 con sus estadísticas, o un arreglo vacío según sea el caso.

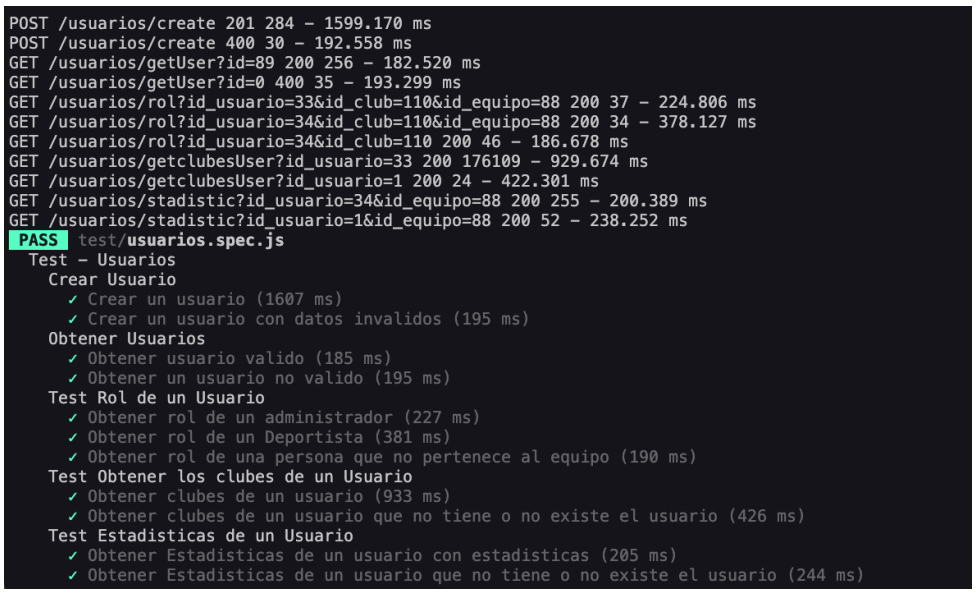

Figura 6.4: Resultados *usuarios.spec.js*

- **Equipo:** En este archivo se tiene como objetivo realizar las pruebas de integración sobre las principales solicitudes que tienen relación sobre los equipos de cada club para lograr las diferentes funcionalidades.

  - *GET /equipo/getEquipos:* Obtiene todos los equipos de un club. Se utiliza como caso de prueba un club con equipos y otro no válido. La respuesta debe contener un arreglo de elementos con tamaño superior a 0 o vacío, según sea el caso.

  - *GET /equipo/getEquiposByUser:* Obtiene los equipos del club al que pertenece un usuario. Se entrega como parámetro un usuario perteneciente a equipos, por lo cual, se espera como respuesta un arreglo con elementos.

  - *POST /equipo/createEquipo:* Crea un equipo dentro de un club. Se espera que la respuesta contenga un código de estado 201, lo cual indica que ha sido creado correctamente.

  - *DELETE /equipo/deleteEquipo:* Elimina un equipo, para realizar las pruebas se utiliza un código válido y no válido. Se espera como respuesta un código de estado 200 o 400 según sea el caso.

  - *GET /equipo/stadistic:* Obtiene estadísticas de un equipo dentro de una fecha de inicio y una fecha final. Se utiliza como caso de prueba un equipo con datos y uno no válido o sin datos, por lo que se espera como respuesta un objeto con elementos o un objeto vacío, según corresponda.

- *GET /equipo/miembros:* Obtiene los miembros del equipo. Se espera que la solicitud se ejecute de manera correcta entregando como respuesta un código de estado 200, o un arreglo vacío en caso de no haber miembros o no existir el equipo.

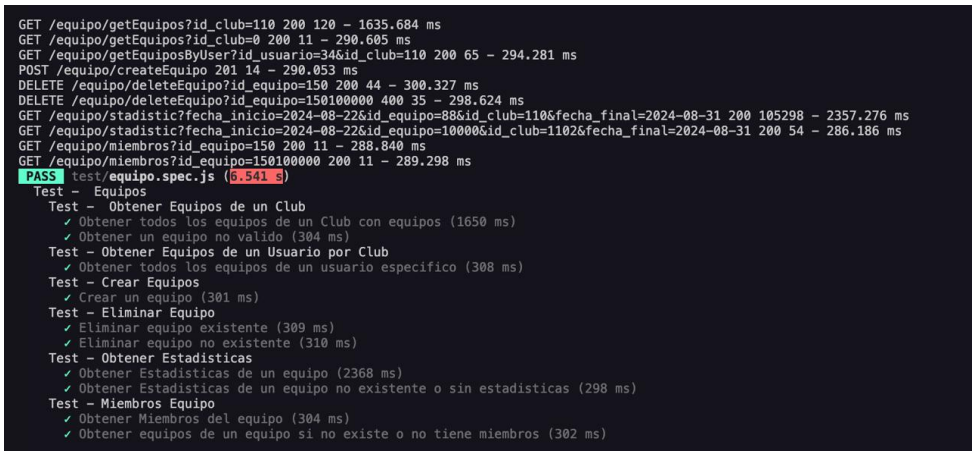

Figura 6.5: Resultados *equipo.spec.js*

- **eventos_asistencia:** Este archivo tiene como objetivo realizar las pruebas de integración sobre las solicitudes que tienen relación con los eventos y la asistencia a estos. Por lo tanto,para realizar las pruebas, se emulará el flujo de las solicitudes en un orden esperado.

  - *POST /eventos:* Crea un evento dentro de un equipo. Se espera como respuesta un código de estado 201.

  - *POST /asistencia:* Confirma y/o cancela asistencia al evento creado dentro del equipo. Se espera que la respuesta contenga un código de estado 200, en conjunto con el mensaje *"Asistencia confirmada con éxito"* o *"Asistencia cancelada con éxito"*.

  - *PATCH /eventos/estado:* Realiza el cambio de estado del evento. Marcando cómo finalizado o activo. Se espera que la respuesta contenga un código de estado 200, en conjunto con el mensaje *"Evento actualizado"*.

  - *PUT /eventos:* Actualiza los datos que se tienen sobre un evento. Se espera que la respuesta contenga un código de estado 200, en conjunto con el mensaje *"Actualizado con éxito"*.

  - *GET /eventos:* Obtiene los eventos creados dentro de un mes y año especifico.

Se utilizan 2 casos de prueba, caso en el cual el evento creado previamente exista, y un caso donde no existan eventos, por lo que, para el primer caso, se comprueba que la respuesta entregue un arreglo de elementos que contenga el evento creado anteriormente, en caso contrario se espera que la respuesta contenga un arreglo vacío.

- *DELETE /eventos:* Elimina el evento previamente creado, por lo que se espera que la respuesta contenga un código de estado 200, junto con el mensaje *"Eliminado con éxito"*.

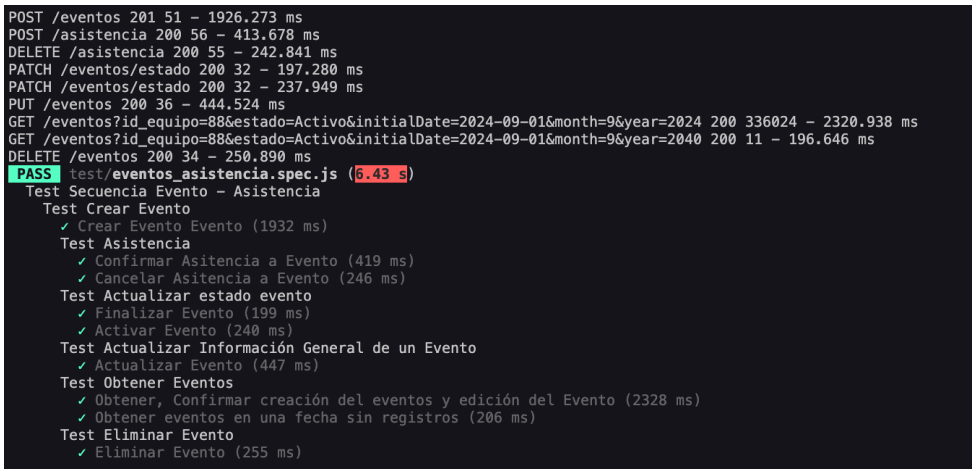

Figura 6.6: Resultados *eventos_ asistencia.spec.js*

- **solicitud_miembro:** Este archivo tiene como objetivo realizar las pruebas de integración sobre las solicitudes que tienen relación con las solicitudes de afiliación a los clubes.

  - *POST /solicitud/send:* Crea una solicitud en estado "Pendiente" entre un club y un usuario.

  - *GET /solicitud/getEstado:* Obtiene el estado de una solicitud de afiliación. Se obtiene la solicitud creada anteriormente comprobando que esta exista y que contenga un estado *"Pendiente"*.

  - *GET /solicitud/getPendientes:* Obtiene las solicitudes de un club que se encuentran en estado "Pendiente". Se espera que la respuesta sea un arreglo de elementos.

  - *POST /miembro/assignMiembro:* Acepta la solicitud y agrega al usuario a un equipo con un rol específico. Se espera que la respuesta entregue un código de

estado 200 en conjunto con el mensaje *"Solicitud aceptada con éxito"*.

- *DELETE /miembro/deleteMiembroEquipo:* Elimina al miembro de un equipo. Se espera que la respuesta entregue un código de estado 200 con el mensaje *"Miembro eliminado con éxito"*.

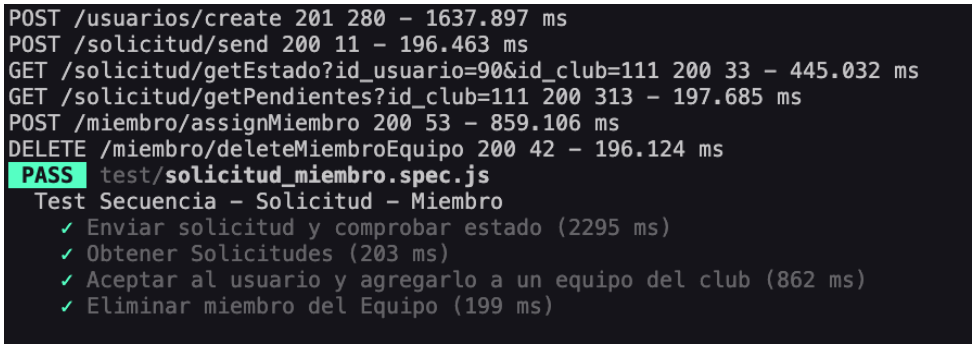

Figura 6.7: Resultados *solicitud_ miembro.spec.js*

- **config_evento:** Este archivo tiene como objetivo realizar las pruebas de integración sobre las solicitudes que tienen relación con los eventos recurrentes. Para realizar estas pruebas se emulará una secuencia de solicitudes en un orden válido.

  - *POST /configEventos:* Crea una configuración de evento recurrente un día de la semana entre una fecha de inicio y fecha final. En caso de no haber algún día entre esas fechas se recibe como respuesta un mensaje *"No sé encontraron fechas en las que se repetirá el evento"*.

  - *GET /configEventos:* Obtiene información de las configuraciones de los eventos de un equipo. Se espera que la respuesta corresponda a un arreglo de elementos con tamaño superior a 0, y que contenga la configuración creada previamente.

  - *DELETE /configEventos:* Elimina la configuración del evento recurrente, lo cual elimina todos los eventos en estado activo que se encontraban asociada a esta configuración. Para comprobar que esta solicitud funcione correctamente, se ejecuta esta solicitud 2 veces tratando de eliminar la misma configuración, por lo tanto, se espera como respuesta para la primera instancia un código de estado 200 con el mensaje *"Se ha eliminado con éxito"*, en cambio en la segunda instancia se espera un código de estado 400 con el mensaje *"No se logró eliminar esta Configuración"*.

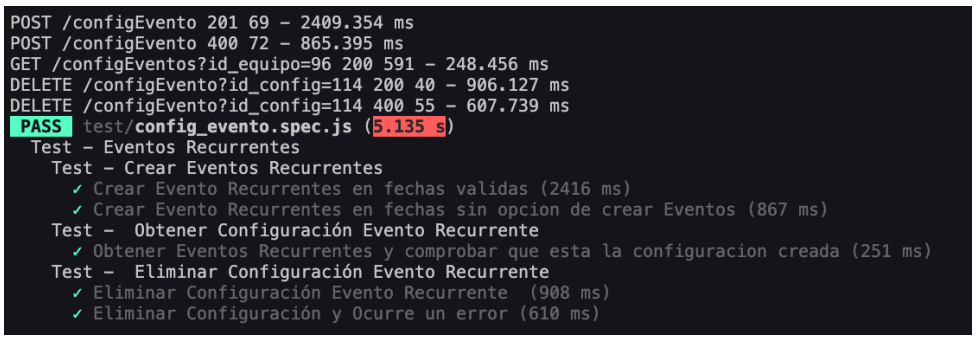

Figura 6.8: Resultados *config_ evento.spec.js*

En la figura 6.9, se puede apreciar el total de pruebas realizadas, todas ellas de manera exitosa, lo que reduce la cantidad de posibles errores en la aplicación. Ademas, cabe destacar que la cantidad de solicitudes abordadas fueron 35, lo cual corresponde a un 79,5 % del total de solicitudes disponibles. Esto garantiza que un gran porcentaje del backend se encuentra debidamente evaluado, ofreciendo confiabilidad en su funcionamiento.

Figura 6.9: Resultados del total de Pruebas de Integración

6.2. Validación con Usuarios

Con el objetivo de evaluar la experiencia del usuario y obtener retroalimentación, se realizó la validación con usuarios objetivos, es decir, miembros y administradores de un club. La validación consistió en probar la aplicación para que posteriormente se respondiera una encuesta de manera inmediata y precisa. La encuesta fue basada en “System Usabilty Scale(SUS)” [8], una herramienta creada por John Brooke en 1986, que se ha convertido en una de las más utilizadas para evaluar la usabilidad de un sistema, producto o servicio. La encuesta consiste de 10 preguntas, cada pregunta se puntúa en una escala de Likert del 1 al 5, siendo 1 “Totalmente en desacuerdo” y 5 “Totalmente de acuerdo”. Además, a la encuentra se agregaron otras preguntas (6.1) para obtener retroalimentación de la aplicación. Los resultados proporcionan una evaluación cuantitativa de la usabilidad, los cuales definen la usabilidad de acuerdo a la tabla 6.2.

Preguntas	**Escala Evaluación**				
1. Creo que me gustaría utilizar este sistema con frecuencia	1	2	3	4	5
2. Encontré el sistema innecesariamente complejo	1	2	3	4	5
3. Pensé que el sistema era fácil de usar	1	2	3	4	5
4. Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema	1	2	3	4	5
5. Encontré que las diversas funciones de este sistema estaban bien integradas	1	2	3	4	5
6. Pensé que había demasiada inconsistencia en este sistema	1	2	3	4	5
7. Me imagino que la mayoría de la gente aprendería a utilizar este sistema muy rápidamente	1	2	3	4	5
8. Encontré el sistema muy complicado de usar	1	2	3	4	5
9. Me sentí muy seguro usando el sistema	1	2	3	4	5
10. Necesitaba aprender muchas cosas antes de empezar con este sistema	1	2	3	4	5
Otras					
11. Del 1 al 10. ¿Cómo evaluaría la interfaz/diseño de la plataforma?					
12. ¿Qué funcionalidad de la plataforma ha sido la que más le ha gustado?					
13. ¿Qué otras funcionalidades extras agregarías a la plataforma?					
14. Comentario/opinión en general					

Cuadro 6.1: Encuesta de usabilidad basada en SUS

Puntuación SUS	Aceptación
[70 - 100]	Aceptable
[50 - 70 [	Marginal
[0 - 50[	No Aceptable

Cuadro 6.2: Relación entre la puntuación SUS y la aceptación del sistema

La encuesta fue realizada a un grupo de 7 personas las cuales forman parte del *Club Deportivo Mewlen Angol*, club encuestado para obtener los requerimientos, estas 7 personas

(6 deportistas y 1 administrador) utilizaron la aplicación con privilegios de deportistas y administrador, a las cuales se les encomendó realizar distintas acciones para probar las funcionalidades presentes. A los deportistas se les encomendó explorar clubes, aplicar filtros, enviar solicitud de afiliación, visualizar eventos, confirmar asistencia y visualizar sus estadísticas. Por otro lado, el administrador realizó las acciones: crear equipo, aceptar solicitudes de afiliación, crear eventos, crear configuración de eventos recurrentes, visualizar estadísticas de los miembros y visualizar estadísticas de los equipos.

La puntuación obtenida por la encuesta realizada fue de 92.5, lo que indica que la usabilidad de la aplicación móvil es aceptable según los estándares de la escala. También, dada las repuestas obtenidas en las preguntas de retroalimentación se destacan las funcionalidades principales como la gestión de eventos, registro de asistencia, generación de estadísticas y la facilidad de explorar clubes de acuerdo a deportes de interés. Ademas, se destacó la idea y el potencial que podría tener la aplicación. Entre algunos comentarios obtenidos se destacan los siguientes :

- “ La aplicación tiene un diseño muy fácil de usar, así que incluso las personas que no son muy expertas en tecnología pueden manejarla sin problemas. Además, ofrece muchas funciones útiles.”
- “ Es una aplicación práctica y eficaz para organizar eventos deportivos.”

Además, como funcionalidades extras que resultarían útiles señaladas por los usuarios se encuentran las siguientes:

- “ Las estadísticas en competiciones a nivel club.”
- “ Que haya la opción de enviarse mensajes entre personas que tengan la aplicación, para recomendarse clubes.”
- “ Quizá más roles de usuario en los cuales cada usuario a medida que más organice eventos, suba de rol o etc... para mayor credibilidad del individuo organizador.”

7. Conclusión

El propósito principal del presente proyecto era crear una aplicación, *ClubConnect*, con el objetivo de fomentar la actividad física e interacción social mediante la visibilización de espacios deportivos. A lo largo de este informe se ha descrito el proceso de la creación de la aplicación, su base de datos, tecnologías, arquitecturas e interfaz con la cual el usuario interactúa. En esta sección se discutirá el proceso de desarrollo y resultados obtenidos del proyecto.

Inicialmente, la propuesta se centraba en desarrollar una aplicación móvil que desplegara los clubes deportivos cercanos e incluyera un sistema de eventos y asistencia para estos. Sin embargo, en el transcurso del desarrollo se fueron agregando nuevas funcionalidades en conjunto con el profesor encargado, como el sistema de estadísticas por usuario y equipo en general, ademas de mejoras en la interfaz de usuario, con el objetivo de aumentar el valor agregado de la aplicación.

Un punto importante a destacar de esta versión de *ClubConnect* es la libertad que ofrece a los usuarios para crear clubes deportivos. Sin embargo, se plantea a futuro lograr presentar como propuesta que la creación de clubes esté administrado por unidades municipales de deporte. Estas entidades actuarían como super administradores, quienes posean permisos sobre la creación de clubes y seguimientos de estos.

En conclusión, en esta primera versión de *ClubConnect* se ha logrado cumplir con los objetivos propuestos al inicio del documento y desafíos en el transcurso del desarrollo. Se ha validado la aplicación con usuarios objetivo, obteniendo como resultado que la usabilidad de la aplicación es aceptable, expresando opiniones positivas y destacando las funcionalidades que presenta.

8. Trabajo a Futuro

Para garantizar la continua evolución de la aplicación y aumentar su crecimiento en términos de funcionalidades, se sugiere implementar las limitaciones mencionadas en apartados anteriores y las siguientes mejoras y/o funcionalidades:

- **Sistema de torneos y estadísticas:** Desarrollar un sistema de gestión de torneos donde se permita crear, organizar y dar seguimiento a distintas competencias. Este sistema debería permitir equipos participantes, programación de encuentro entre equipos y registro de resultados. Cabe destacar que un sistema así dependería netamente del deporte que se practique, por ende, esta mejora requeriría de un gran trabajo, pero que aumentaría en gran cantidad el valor agregado de la aplicación.
- **Mejora sistema de estadísticas:** Desarrollar mejoras en el sistema de estadísticas para entregar mayor información sobre la asistencia de cada miembro y eventos. Para identificar de mejor manera patrones de comportamientos, mejoras en la planificación y compromiso de los miembros. Además, en caso de ser útil se podría implementar exportación de archivos de asistencia u otra métrica que se requiera.
- **Feed de eventos públicos:** Desarrollar esta funcionalidad donde los clubes publiquen eventos públicos, con el objetivo de visibilizar eventos y que los usuarios puedan encontrar actividades recreativas a su alrededor.
- **Sistema web:** En caso de que la aplicación incremente sus funcionalidades, se recomendaría a futuro crear una versión web dirigida especialmente a los administradores. Permitiendo una interfaz más amplía con un mayor despliegue de información, facilitando la visualización y el análisis de estadísticas. Como también, el seguimiento detallado de torneos y eventos, así como la gestión de miembros, equipos y asistencias.

Referencias

[1] Flutter. https://flutter.dev/, 2024.

[2] Repositorio proyecto. https://github.com/JoseRoj/MT, 2024.

[3] Supabase. https://supabase.com/, 2024.

[4] Timpik. https://www.timpik.com/, 2024. Accedido el 19 de diciembre de 2024.

[5] Arsys. Todo sobre la arquitectura cliente-servidor. https://www.arsys.es/blog/todo-sobre-la-arquitectura-cliente-servidor#tree-8, 2024. Último acceso: 28 de octubre de 2024.

[6] Clínica UANDES. Obesidad y sedentarismo, 2024. https://www.clinicauandes.cl/noticia/obesidad-y-sedentarismo Último acceso: 28 de octubre de 2024.

[7] Dbdiagram. Dbdiagram. https://dbdiagram.io/home, 2024.

[8] UI from Mars. Cómo medir usabilidad: ¿qué es? https://www.uifrommars.com/como-medir-usabilidad-que-es-sus/, 2024. Último acceso: 28 de octubre de 2024.

[9] Google Firebase. Firebase cloud messaging. https://firebase.google.com/docs/cloud-messaging?hl=es-419, 2024.

[10] Jest. Jest, 2024. https://jestjs.io/.

[11] Node.js. Node.js. https://nodejs.org/en, 2024.

[12] All Nutrition. ¿por qué el deporte mejora la calidad de vida?, 2024. https://allnutrition.cl/blogs/nutriblog/por-que-el-deporte-mejora-la-calidad-de-vida Último acceso: 28 de octubre de 2024.

[13] SportEasy. Sporteasy: Gestor de equipos deportivos. https://www.sporteasy.net/es/, 2024. Último acceso: 19 de diciembre de 2024.

[14] Gustavo Ramón Suárez William Ramírez, Stefano Vinaccia. El impacto de la actividad física y el deporte sobre la salud, la cognición, la socialización y el rendimiento académico: una revisión teórica, 2004. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-885X2004000200008 Último acceso: 28 de octubre de 2024.

9. Anexos

9.1. Ejemplo Test de Integración

```
const request = require("supertest");
const { app, server } = require("../app");
const api = request(app);
const connectionPostgres = require("../database/db");

const club = {
  nombre: "Club Test",
  descripcion: "Agregando un Club de Testing",
  latitud: 0,
  longitud: 0,
  id_deporte: 1,
  logo: "",
  correo: "corretest@gmail.com",
  telefono: "987654321",
  categorias: [1, 2],
  tipos: [1, 2],
  id_usuario: 34,
};

const clubEdit = {
  nombre: "Nueva Prueba",
  descripcion: "Nueva Descricion",
  latitud: 90,
  longitud: 180,
  id_deporte: 2,
  logo: "ASFDSDOSIJIO232",
  correo: "nuevocorre@gmail.com",
  telefono: "976543765",
  categorias: [1, 2, 3],
  tipos: [1, 2, 3],
  facebook: "https://Nuevoclub.com",
  instagram: "https://Nuevoclub.com",
  tiktok: "https://Nuevoclub.com",
};

const deportes = [1, 2, 3, 4];
```

```
const coordenada = -73.020855858922 - 73.05304236710072;

const idValido = 110;

describe("Test Clubes", () => {
  let idClub;

  describe("Test - Obtener Clubes con filtros ( Deportes , Coordenadas )", () => {
    it("Obtener todos los Clubes", async () => {
      const response = await api.get("/club/getclubs").send({
        deportes: deportes,
        northeastLat: -36.80786306470817,
        northeastLng: -73.020855858922,
        southwestLat: -36.84922148848782,
        southwestLng: -73.05304236710072,
      });
      expect(response.statusCode).toBe(200);
      expect(response.body.data).toBeInstanceOf(Array);
      expect(response.body.data.length).toBeGreaterThan(0);
    });
  });
  describe("Test - Crear Club", () => {
    it("Crear un club", async () => {
      const response = await api.post("/club").send(club);
      idClub = response.body.data;
      expect(response.statusCode).toBe(200);
    });
    it("Crear un club con un correo o nombre ya existente", async () => {
      const response = await api.post("/club").send({
        ...club,
      });
      expect(response.statusCode).toBe(400);
    });
  });
  describe("Test - Obtener informacion Club", () => {
    it("Obtener un Club Valido", async () => {
      const response = await api.get("/club/getclub").query({ id: idClub
          });
```

```
74        expect(response.statusCode).toBe(200);
75        expect(response.body.data).toBeInstanceOf(Object);
76      });
77      it("Obtener un Club Invalido", async () => {
78        const response = await api.get("/club/getclub").query({ id: 0 });
79        expect(response.statusCode).toBe(400);
80      });
81    });
82
83    describe("Test - Obtener Miembros Club", () => {
84      it("Obtener Miembros de un Club", async () => {
85        const response = await api.get("/club/getmiembros").query({
              id_club: idValido });
86        expect(response.statusCode).toBe(200);
87        expect(response.body.data).toBeInstanceOf(Array);
88        expect(response.body.data.length).toBeGreaterThan(0);
89      });
90    });
91
92    describe("Test - Editar Informacion Club", () => {
93      it("Editar Informacion del Club", async () => {
94        const response = await api.put("/club/editClub").send({ ...
              clubEdit, id: idClub });
95        expect(response.statusCode).toBe(200);
96        expect(response.body.message).toBe("Club actualizado con exito");
97      });
98    });
99    // TODO : GETmiembros
100 });
101
102 afterAll(() => {
103   connectionPostgres.end();
104   server.close();
105 });
```

Listing 1: Test de ejemplo club.spec.js

9.2. Resultados Encuesta System Usability Scale

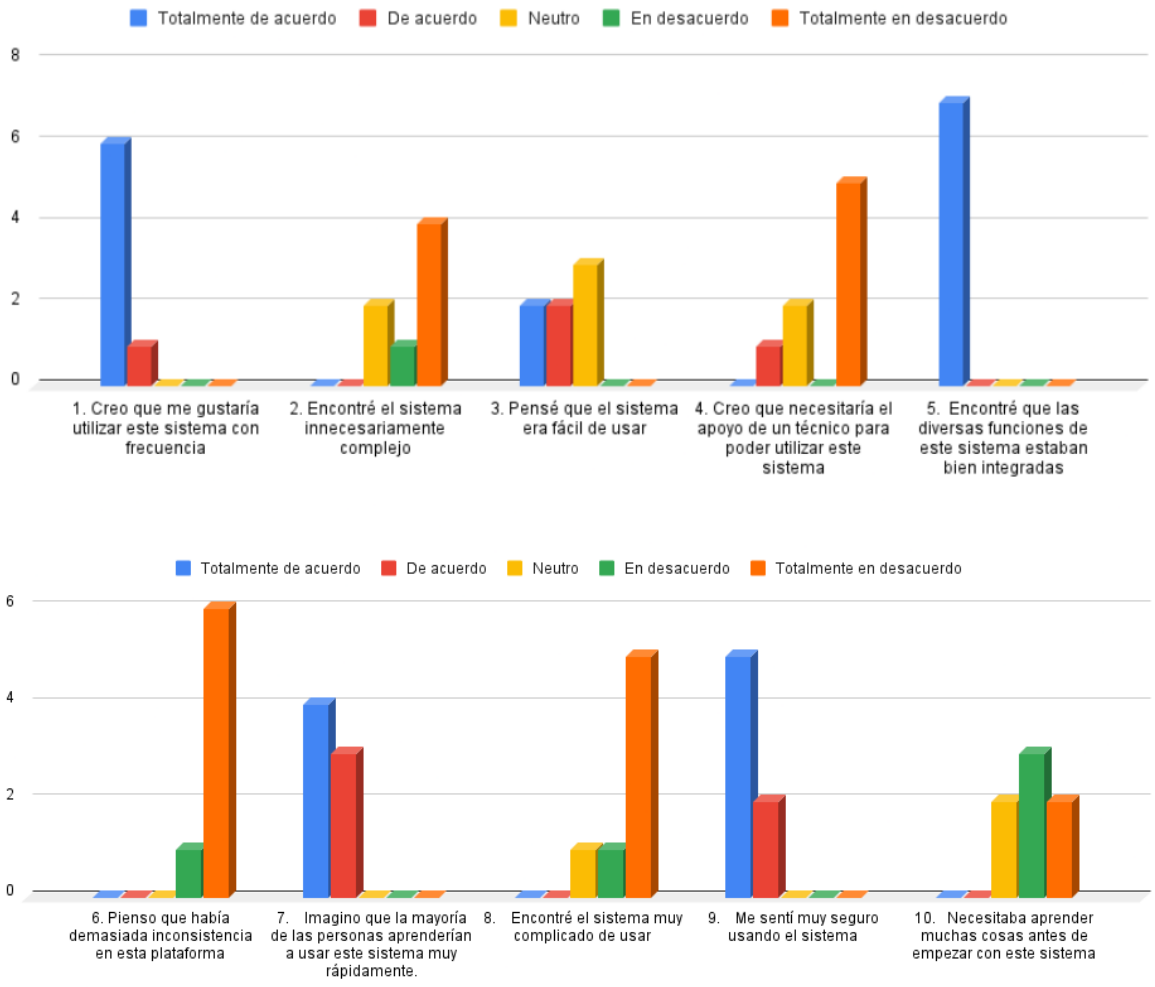

Figura 9.1: Resultados Encuesta SUS