



**Universidad**  
Zaragoza

1542

## Trabajo Fin de Grado

**Una red social para entusiastas del fitness**

*A social network for fitness enthusiasts*

Autor:  
Javier Sin Pelayo

Director:  
Rubén Béjar Hernández

Titulación del autor:  
Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura, Universidad de Zaragoza  
2024 - 2025

# Una red social para entusiastas del fitness

## Resumen

FitConnect es una aplicación web diseñada para crear una comunidad de entusiastas del fitness, permitiendo a los usuarios interactuar, compartir y gestionar sus actividades deportivas en un entorno seguro y saludable.

Entre sus funcionalidades principales se encuentra un robusto sistema de autenticación y seguridad: los usuarios pueden registrarse a través de un formulario que valida el correo electrónico y la contraseña, recibir un correo de verificación, iniciar y cerrar sesión de forma segura (con caducidad de 1 hora) y recuperar la contraseña mediante un enlace de restablecimiento. En cuanto a la gestión del perfil, FitConnect permite actualizar la foto de perfil, el nombre de usuario, la biografía y otros datos personales como peso y altura, además de cambiar la contraseña actual o eliminar la cuenta tras confirmar la acción.

La aplicación también soporta la creación, edición y eliminación de rutinas de entrenamiento personalizadas, donde se pueden definir ejercicios, repeticiones, series, último peso utilizado y duración de cada actividad. Asimismo, se fomenta la interacción social permitiendo a los usuarios crear publicaciones, comentar y dar “me gusta” a las publicaciones de otros.

Además, FitConnect incorpora un sistema de mensajería en tiempo real, permitiendo tanto chats individuales como grupales, con funcionalidades para marcar mensajes como leídos o no leídos. Los grupos de chat se pueden crear, unir mediante invitación y abandonar de manera definitiva. Por otro lado, la gestión de eventos ofrece la posibilidad de crear eventos con detalles como nombre, fecha, ubicación, aforo máximo, descripción e imagen representativa, mientras que se integra un buscador avanzado para localizar usuarios, chats y eventos basados en palabras clave.

Finalmente, el sistema incluye funcionalidades de administración que permiten a los administradores visualizar y eliminar cuentas de usuario, publicaciones, comentarios o eventos que no se ajusten a las políticas de la plataforma. FitConnect se centra en fomentar un ambiente de motivación y colaboración, ofreciendo una experiencia integral tanto para deportistas de gimnasio como para aquellos que disfrutan de actividades al aire libre, diferenciándose a través de la integración de eventos y grupos que potencian la interacción social y el compromiso con el estilo de vida saludable.

La plataforma está desarrollada con Next.js 14, React.js y TypeScript, mientras que Supabase se encarga de la autenticación y la gestión de la base de datos PostgreSQL.

# Tabla de contenidos

<b>Resumen.....</b>	<b>1</b>
<b>Tabla de contenidos.....</b>	<b>2</b>
<b>1. Introducción.....</b>	<b>3</b>
1.1 ¿Cómo se usa FitConnect?.....	4
<b>2. Análisis del problema.....</b>	<b>7</b>
2.1. Requisitos.....	7
Tipos de usuario.....	7
Requisitos Funcionales.....	7
Requisitos No Funcionales.....	9
2.2. Interfaces de usuario.....	9
<b>3. Diseño de la aplicación.....</b>	<b>10</b>
3.1. Arquitectura.....	10
3.1.1. Modelo de datos.....	11
3.1.2. Componentes y conectores.....	17
3.2. Implementación.....	18
3.3. Pruebas.....	20
<b>4. Gestión del proyecto.....</b>	<b>21</b>
<b>5. Conclusiones y trabajo futuro.....</b>	<b>23</b>
<b>Referencias.....</b>	<b>24</b>
<b>Anexos.....</b>	<b>26</b>
Anexo I: Casos de uso.....	26
Anexo II: Interfaces de usuario.....	39

# 1. Introducción

En la actualidad, el deporte se ha consolidado como un elemento clave para el bienestar de las personas. FitConnect es un proyecto innovador que surge con el propósito de crear una plataforma digital que impulse un estilo de vida activo y saludable. La idea fundamental detrás de FitConnect es utilizar el ejercicio físico no solo como medio para mejorar la condición física, sino también como una herramienta para aliviar el estrés, fomentar la autoestima y fortalecer las relaciones interpersonales. La visión es construir una comunidad en la que el deporte se viva de forma colaborativa, motivando a los usuarios a compartir sus rutinas y experiencias, y permitiendo a los administradores mantener un ambiente libre de toxicidad.

El propósito principal de FitConnect es ofrecer a los usuarios una herramienta integral en la que puedan gestionar sus actividades deportivas y sociales. La plataforma permite realizar un seguimiento detallado de entrenamientos, crear y compartir publicaciones, interactuar mediante "me gusta" y comentarios, y participar en chats individuales y grupales. Además, se incorpora la posibilidad de crear y unirse a eventos deportivos, facilitando la socialización entre personas con intereses afines. FitConnect se orienta tanto a usuarios comunes como a administradores, quienes tienen la capacidad de moderar contenidos y gestionar la comunidad para garantizar que se mantengan los valores de respeto, colaboración y salud.

Existen en el mercado diversas aplicaciones que se centran en aspectos específicos del entrenamiento y la actividad física, como *Hevy* [1] y *Strong* [2], que se enfocan en el seguimiento de rutinas de gimnasio y ejercicios de fuerza, y *Strava* [3], que se especializa en deportes de resistencia y actividades al aire libre. Sin embargo, ninguna de estas soluciones integra de forma completa ambas aproximaciones, ni fomenta la creación de una comunidad saludable mediante la organización de eventos y grupos de entrenamiento. FitConnect se diferencia al combinar el seguimiento personalizado del rendimiento deportivo con funcionalidades sociales avanzadas, permitiendo a los usuarios interactuar, competir de manera saludable y apoyarse mutuamente en la consecución de sus objetivos.

El desarrollo de FitConnect se ha llevado a cabo utilizando tecnologías modernas y probadas. El frontend se implementa con *Next.js 14* [4] y *React.js* [5] en combinación con *TypeScript*, lo que permite una construcción de interfaces de usuario dinámicas, escalables y con un *diseño responsive* [6]. Para el diseño visual se ha empleado *TailwindCSS* [7], facilitando la creación de layouts modernos y adaptables a distintos dispositivos. En el backend, *Supabase* [8] se encarga de la autenticación y del manejo de la base de datos PostgreSQL, simplificando el proceso de integración y permitiendo un desarrollo ágil y seguro. La elección de estas tecnologías se basó en criterios de rendimiento, facilidad de mantenimiento y escalabilidad, comparándolas favorablemente con alternativas como *Firebase* [9] o *Express.js* [10], que si

bien son robustas, en este caso no se ajustaban tan adecuadamente a las necesidades específicas del proyecto.

FitConnect se desarrolla en el contexto de un creciente interés por la salud y el bienestar, en el que la actividad física se considera esencial para contrarrestar el estrés y mejorar la calidad de vida. El proyecto está orientado a un público que valora tanto el entrenamiento individual como la motivación que se obtiene al practicar deporte en grupo. La plataforma también está diseñada para empresas y organizaciones que desean promover un estilo de vida activo entre sus empleados o miembros, a través de funcionalidades que permiten organizar eventos y competiciones saludables. Además, la aplicación incluye un robusto sistema de moderación para garantizar que la comunidad se mantenga libre de comportamientos tóxicos, permitiendo a los administradores gestionar contenidos y usuarios según las políticas establecidas.

El presente informe sigue la recomendación de 20 páginas de extensión y se organiza en varias secciones. La primera parte se centra en el análisis del problema, donde se detallan los requisitos funcionales y no funcionales, los casos de uso y las interfaces de usuario. La segunda sección aborda el diseño de la solución, incluyendo la arquitectura del sistema y los modelos de datos. Seguidamente se describe el proceso de implementación, las técnicas de procesamiento de datos, las estrategias de testing y la gestión del proyecto. Por último, se presentan el análisis de negocio, las conclusiones y las posibles líneas de trabajo futuro.

## 1.1 ¿Cómo se usa FitConnect?

Para entender mejor el funcionamiento de FitConnect, a continuación se ilustra una jornada de uso típica por parte de un usuario activo:

- **Inicio del día:** El usuario accede a la aplicación, revisa las notificaciones de nuevos mensajes o comentarios en sus publicaciones y responde desde el chat. [Figura 1](#).

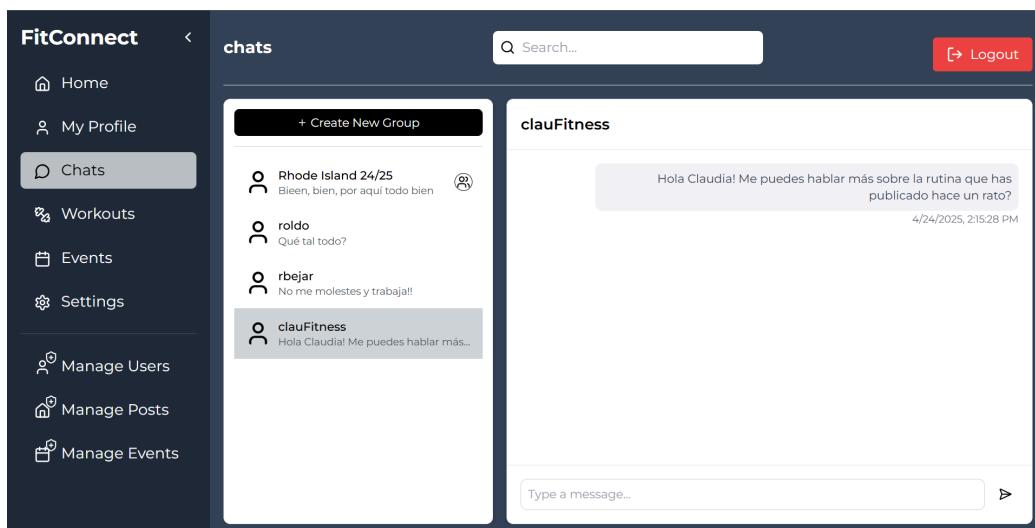


Figura 1: Pantalla final de Chats

- **Creación de rutina:** Decide registrar una nueva rutina de entrenamiento para el gimnasio. Accede a la sección de entrenamientos, añade ejercicios, define repeticiones, series y peso utilizado. Todo desde un entorno visual e intuitivo. [Figura 2](#).

The screenshot shows the 'Workouts' section of the FitConnect app. On the left is a sidebar with navigation links: Home, My Profile, Chats, Workouts (which is selected and highlighted in grey), Events, Settings, Manage Users, Manage Posts, and Manage Events. The main content area has a title 'Push Day' and a subtitle 'Workout routine designed to develop Chest, Shoulder and Triceps hypertrophy.' Below this is a table showing three exercises:

EXERCISE	MUSCULAR GROUP	SETS	REPS	LAST WEIGHT USED (KG)	DURATION (SEC)
Inclined Dumbbell Press	Chest	6	7	22	-
Cable Fly Crossovers	Chest	4	10	60	-
Test Ex	Test Gr	2	4	69	0

At the bottom of the table are two buttons: 'Show weight in lb' (grey) and 'Delete Workout' (red). There are also 'Edit Workout' and 'Delete Workout' buttons at the bottom right of the main content area.

Figura 2: Pantalla final de Workouts

- **Publicación motivacional:** Tras finalizar el entrenamiento, comparte una imagen del progreso con una breve reflexión. Recibe comentarios y reacciones de otros miembros de la comunidad. [Figura 3](#).

The screenshot shows the 'home' screen of the FitConnect app. The sidebar on the left is identical to Figura 2. The main content area shows a post by a user named 'javisin' from 1 day ago. The post features a large image of a shirtless man performing a chest press exercise on an incline bench. Below the image is the caption 'Starting the week with a good chest workout!! 💪💪'. At the bottom right of the post area is a blue circular button with a white plus sign (+).

Figura 3: Pantalla final de Feed

- **Consulta de eventos:** Explora el buscador para descubrir eventos deportivos próximos en su zona. Se apunta a una carrera popular organizada por otro usuario del grupo de running. [Figura 4](#).

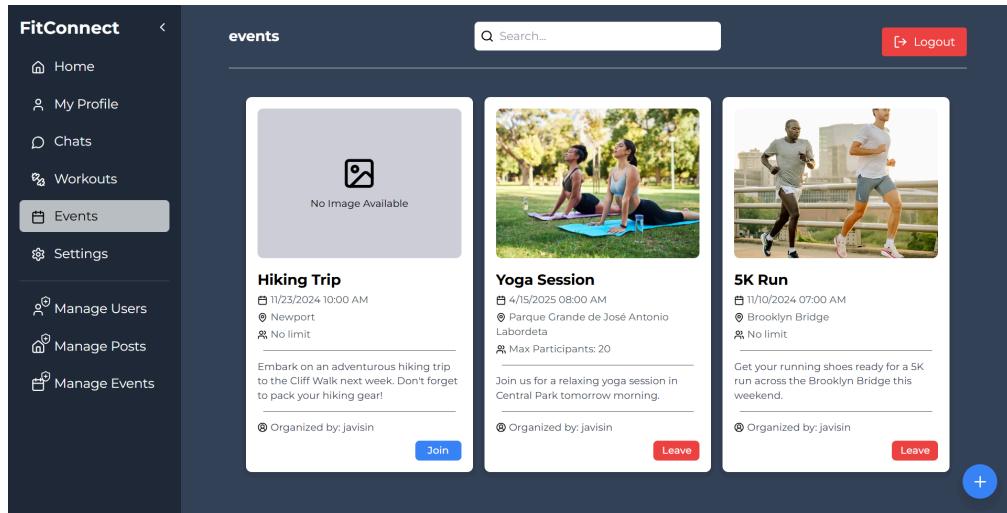


Figura 4: Pantalla final de Events

- **Moderación y comunidad:** Si se trata de un usuario administrador, revisa publicaciones recientes y detecta una que no cumple con las normas. La elimina desde el panel de administración, manteniendo el espacio libre de toxicidad. [Figura 5](#).

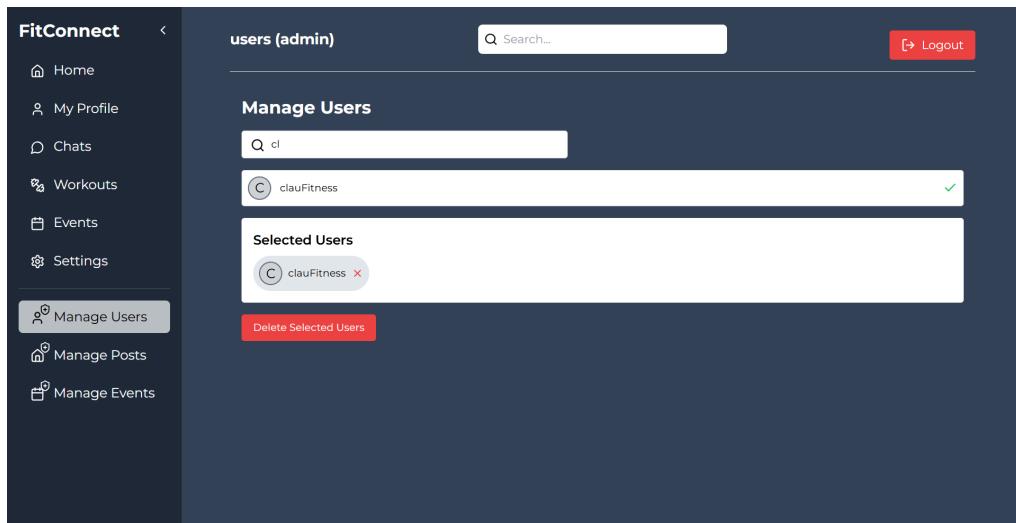


Figura 5: Pantalla final de Administración de usuarios

Este recorrido refleja cómo FitConnect no es solo una herramienta para registrar entrenamientos, sino una plataforma viva que conecta a personas con objetivos comunes. A continuación se incluyen capturas reales de cada una de estas funcionalidades para facilitar su comprensión visual.

## 2. Análisis del problema

A continuación se procede a analizar los requisitos tanto funcionales como no funcionales de la aplicación y las interfaces de usuario de la misma.

### 2.1. Requisitos

#### Tipos de usuario

La aplicación está dirigida principalmente para los *usuarios estándar* que la usarán de manera diaria. Sin embargo, se ha implementado de manera adicional el usuario *administrador* para realizar labores de mantenimiento y moderar la aplicación web.

#### Requisitos Funcionales

La aplicación debe permitir que los usuarios:

##### 1. Autenticación y Seguridad

*RF1.1. Se registren mediante un formulario que requiera un correo electrónico válido y una contraseña segura. Después del registro, el usuario debe recibir un correo electrónico de verificación para activar su cuenta.*

*RF1.2. Inicien y cierren sesión de manera segura. La sesión debe expirar automáticamente después de un período de 1 hora.*

*RF1.3. Recuperen su contraseña mediante el envío de un enlace de restablecimiento a su correo electrónico.*

##### 2. Gestión de Perfil

*RF2.1. Gestionen su perfil personal, ofreciendo la capacidad de subir o actualizar una foto de perfil, cambiar su nombre de usuario, actualizar la biografía y ajustar información personal como peso y altura.*

*RF2.2. Cambien su contraseña actual proporcionando la contraseña antigua y una nueva contraseña.*

*RF2.3. Eliminen su cuenta permanentemente, tras confirmar esta acción mediante un aviso previo.*

##### 3. Gestión de Rutinas

*RF3.1. Creen rutinas de entrenamiento personalizadas, especificando los siguientes detalles: ejercicios, repeticiones, series, último peso utilizado y duración del ejercicio (si éste lo requiere).*

*RF3.2. Editen rutinas de entrenamiento existentes, con la opción de agregar, modificar o eliminar ejercicios.*

*RF3.3. Eliminen rutinas de entrenamiento existentes de manera definitiva.*

#### **4. Interacciones Sociales**

*RF4.1. Creen publicaciones con una foto y una descripción.*

*RF4.2. Editen sus propias publicaciones, pudiendo cambiar la descripción y la foto.*

*RF4.3. Eliminen sus propias publicaciones de manera definitiva.*

*RF4.4. Interactúen con las publicaciones de otros usuarios: comentar y dar "me gusta".*

#### **5. Mensajería y Comunicación**

*RF5.1. Envíen y reciban mensajes privados en tiempo real, con la opción de ver el historial de mensajes y marcar las conversaciones como leídas o no leídas.*

#### **6. Grupos de Chat**

*RF6.1. Creen grupos de chat, especificando el nombre y la lista de integrantes.*

*RF6.2. Se unan a grupos de chat existentes mediante un sistema de invitación.*

*RF6.3. Salgan de grupos de chat de manera definitiva.*

#### **7. Gestión de Eventos**

*RF7.1. Creen eventos, especificando los siguientes detalles: nombre, fecha, lugar, aforo máximo, descripción, y una imagen representativa.*

#### **8. Búsqueda**

*RF8.1. Busquen otros usuarios, chats o eventos mediante un buscador integrado que ofrezca resultados relevantes basados en palabras clave.*

#### **9. Administración**

El sistema debe permitir que los administradores:

*RF9.1. Visualicen y eliminén cualquier cuenta de usuario, publicaciones, comentarios o eventos inapropiados.*

## Requisitos No Funcionales

### 1. Seguridad

**RNF1.1.** *El sistema debe proteger las contraseñas y cualquier dato sensible mediante técnicas de encriptación robustas. Para ser exactos, se utiliza el algoritmo de encriptación ‘bcrypt’.*

**RNF1.2.** *El sistema deberá realizar las comprobaciones de autenticación en el lado del servidor para evitar acciones maliciosas por parte del cliente.*

### 2. Compatibilidad y Accesibilidad

**RNF2.1.** *La aplicación se halla totalmente internacionalizada. De este modo, todo se encuentra escrito en inglés. Además, se permite en todo momento el cambio de las unidades de peso y medida, optando el sistema imperial y el métrico.*

**RNF2.2.** *El sistema debe ofrecer un diseño adaptable (responsive design) que asegure la correcta visualización y usabilidad en diferentes navegadores y dispositivos. Exactamente funciona correctamente en las últimas versiones de Chrome (134.0.6998.165) y Firefox (136.0.2), y en las siguientes resoluciones:*

- Ordenador: 1920 x 1080 (Full HD), 1280 x 800
- Tablet: 768 x 1024 (modo retrato), 1024 x 768 (modo paisaje), 800 x 1280 (común en tablets Android)
- Móviles: 360 x 640 (común en móviles Android), 375 x 667 (iPhone 6/7/8), 375 x 812 (iPhone X, XS, 11 Pro), 414 x 896 (iPhone 11/11 Pro Max, iPhone XS Max)

Una vez definidos y detallados los requisitos funcionales y no funcionales de la aplicación, se han desarrollado casos de uso que especifican los flujos de interacción entre el usuario y el sistema. Estos casos de uso se pueden encontrar en el [Anexo I](#).

### 2.2. Interfaces de usuario

Se han elaborado bocetos de las interfaces de usuario que corresponden a cada uno de los casos de uso definidos. Estos bocetos permiten visualizar la experiencia y los flujos de interacción previstos en la aplicación.

Aunque el desarrollo de la aplicación se realizó de manera independiente y personalizada, se ha seguido un enfoque basado en las mejores prácticas en cuanto a la implementación, tales como el *diseño responsive* para dispositivos móviles, y se han considerado aspectos críticos de accesibilidad, como el uso adecuado de contrastes y tipografías legibles. En este sentido, se han seguido directrices de implementación reconocidas en la industria, lo que garantiza que la

experiencia de usuario sea óptima en distintos dispositivos y cumpla con los estándares de usabilidad y accesibilidad. Estas pautas han sido tenidas en cuenta tanto en el diseño visual como en la estructura del código, permitiendo un producto final que, pese a su carácter único, se alinea con las expectativas y recomendaciones actuales en desarrollo web.

La información completa sobre el prototipado de las interfaces de usuario se ha incluido en el [Anexo II](#).

## 3. Diseño de la aplicación

### 3.1. Arquitectura

El sistema se estructura como una aplicación web con **3-tiers** que sigue el **modelo cliente-servidor**, donde la lógica del backend se apoya en un proveedor externo de servicios Backend-as-a-Service (BaaS), en este caso *Supabase*. Un BaaS (Backend-as-a-Service) es un modelo de servicio en la nube que proporciona funcionalidades del lado del servidor listas para usar, como autenticación, bases de datos, almacenamiento y notificaciones, sin necesidad de implementar y mantener una infraestructura propia. Esto permite centrarse en la lógica de negocio y la experiencia de usuario, delegando en el BaaS tareas complejas y repetitivas del backend.

A continuación se definen los tres tiers, además de introducir los frameworks y tecnologías utilizadas junto a sus implicaciones:

- **Tier de Presentación (cliente):**

Este tier engloba toda la parte visible e interactiva de la aplicación. Aunque se ha desarrollado con *Next.js*, lo que finalmente se renderiza en el navegador es HTML, CSS y JavaScript generado a partir del código fuente de *React* y *TypeScript*.

*Next.js* es un framework de desarrollo web basado en *React* que permite construir aplicaciones optimizadas, escalables y modernas. Ofrece funcionalidades avanzadas como renderizado híbrido (estático y del lado del servidor), enrutamiento integrado, API routes y soporte completo para *TypeScript*. En el contexto de FitConnect, el uso de *Next.js* permite una organización clara del proyecto, mejora el rendimiento mediante la renderización eficiente de las páginas, y proporciona una experiencia de usuario fluida y rápida gracias a su sistema de rutas dinámicas y precarga automática.

En este tier se incluyen las interfaces de usuario, la lógica de interacción y la navegación, así como la integración con Contexts y otras librerías que gestionan el estado y la comunicación en tiempo real. Es decir, el Frontend se encarga de presentar los datos y de recoger la interacción del usuario para enviarla al backend.

- **Tier de Servidor Web (*Next.js*):**

Aunque tanto el cliente como el servidor comparten el mismo mono-repositorio, *Next.js* introduce una capa intermedia ejecutada en el servidor. Esta capa se encarga de:

- Renderizar páginas desde el servidor utilizando **Server Components**.
- Definir y gestionar las **rutas API** en la carpeta '/api' mediante archivos como *route.ts*.
- Aplicar la lógica de negocio previa al acceso a datos.
- Interactuar con el servicio externo *Supabase* mediante llamadas HTTP o SDKs.

En desarrollo, este servidor se ejecuta en *localhost:3000*, pero en producción requiere ser desplegado en una plataforma de hosting.

- **Tier de Datos (Backend-as-a-service: *Supabase*):**

Esta capa está gestionada completamente por *Supabase*, un proveedor de servicios BaaS que centraliza funcionalidades backend críticas sin requerir una infraestructura propia. Entre sus responsabilidades se encuentran:

- Gestión de la **base de datos PostgreSQL**.
- Provisión de un sistema de **autenticación y control de sesiones**.
- **Almacenamiento de archivos** (como las fotos de perfil o las imágenes de las publicaciones).
- Comunicación en **tiempo real** mediante WebSockets [11] (como el sistema de chat en tiempo real).
- Control de acceso basado en **políticas** (Row-Level Security o RLS).

### 3.1.1. Modelo de datos

Esta sección describe el modelo de datos implementado en la base de datos del backend de FitConnect, gestionado mediante *Supabase*. Se detallan las principales entidades, sus atributos y las relaciones entre ellas, las cuales han sido diseñadas para soportar eficientemente las funcionalidades del sistema.

Leyenda:

- Clave Principal
- Clave Ajena
- Atributo único
- Puede ser nulo

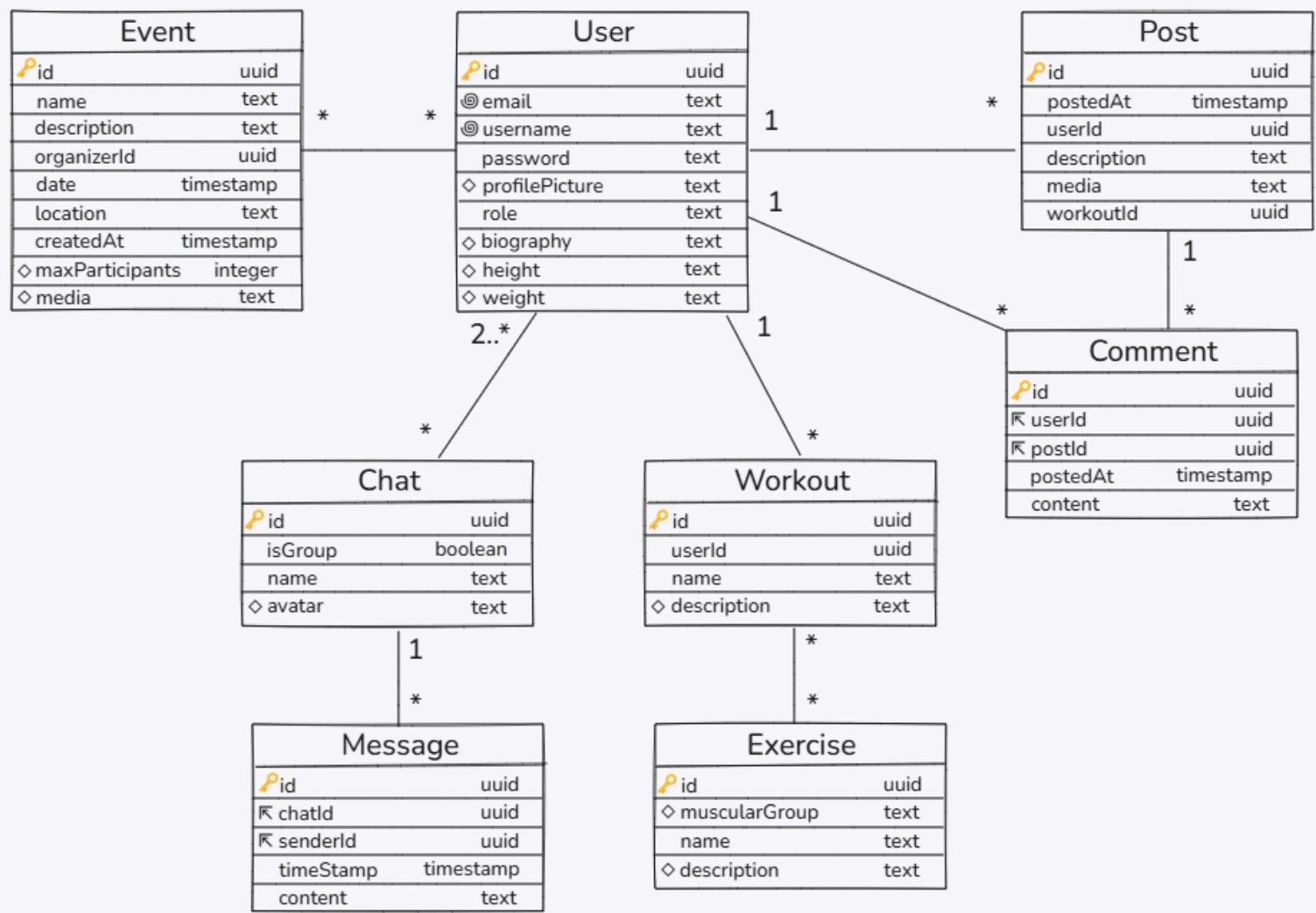


Figura 27: Modelo de datos

- **User (Usuario):** El usuario es la entidad central que representa a cada persona que interactúa con la aplicación. Sus atributos permiten gestionar tanto la autenticación (email, password), como la información personal y de perfil.
- **Post (Publicación):** Cada publicación es creada por un usuario. Incluye una descripción y una imagen. Además, permite la interacción social mediante comentarios y “me gusta”.
- **Comment (Comentario):** Los comentarios permiten a los usuarios interactuar con los posts. Cada comentario se asocia a un único post y es realizado por un usuario.
- **Workout (Rutina de entrenamiento):** Un workout representa una rutina de entrenamiento creada por un usuario. Permite almacenar información que luego se relacionará con ejercicios específicos.
- **Exercise (Ejercicio):** Es la entidad que define cada ejercicio disponible en la aplicación. Se utiliza para vincular ejercicios a las rutinas (workouts).
- **Chat:** Representa una conversación, ya sea individual o grupal. Su atributo **isGroup** permite distinguir el tipo de chat.
- **Message (Mensaje):** Cada mensaje se envía en el contexto de un chat y permite la comunicación en tiempo real entre usuarios.
- **Event (Evento):** Permite la creación y gestión de eventos a los que pueden unirse los usuarios.

## Paquetes y clases

Esta sección presenta la organización del código fuente de FitConnect a nivel de paquetes y clases, detallando la estructura modular empleada en el desarrollo de la aplicación. Se describen los principales módulos del sistema, sus responsabilidades y cómo se relacionan entre sí.

Diagrama de paquetes y clases - Estructura de código (FitConnect)

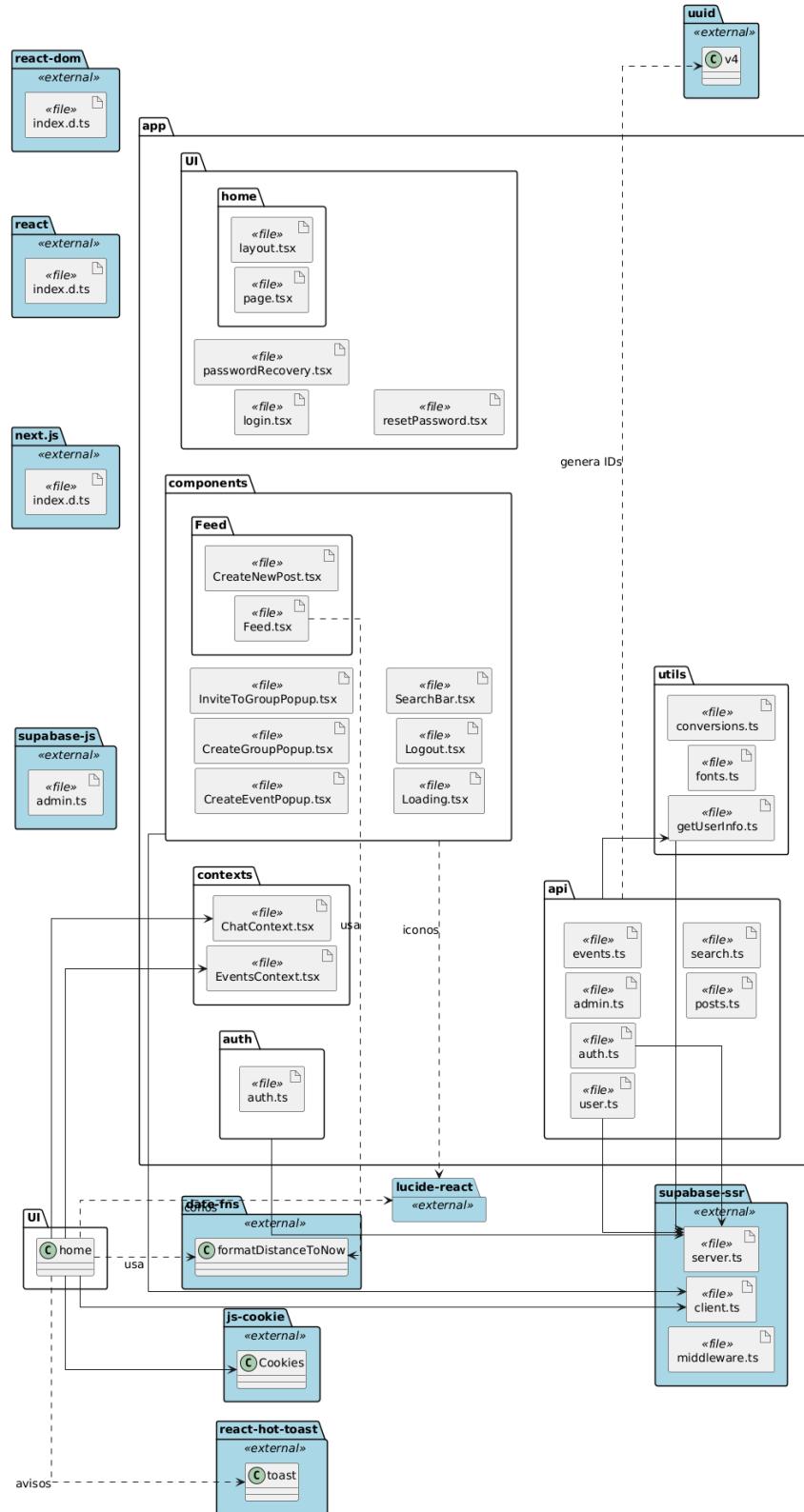


Figura 28: Diagrama de Paquetes

El diagrama representa la arquitectura de paquetes y clases de la aplicación FitConnect. Se divide en dos grandes bloques:

- **Bibliotecas externas:** Agrupa las clases proporcionadas por *Supabase*, que se encargan de funcionalidades esenciales como la creación de clientes para la autenticación y gestión de sesiones, así como el acceso a la base de datos.
- **Frontend principal ("app"):** Contiene la estructura interna de la aplicación, organizada en varios paquetes que agrupan las funcionalidades de la interfaz de usuario, lógica de negocio, comunicación con el backend y utilidades.

A continuación se procede a la explicación en detalle de los distintos paquetes y clases:

- **supabase-ssr & supabase-js:** Este paquete es externo y representa la plataforma *Supabase* que provee servicios esenciales (autenticación, gestión de sesiones, acceso a base de datos).
  - **client.ts:** Se utiliza para crear una instancia de cliente que se ejecuta en el navegador, facilitando la autenticación y acceso a datos en el frontend.
  - **server.ts:** Se encarga de crear un cliente en el servidor, lo que resulta útil para operaciones que requieran mayor seguridad o interacción directa con la base de datos.
  - **admin.ts:** Se encarga de crear un cliente con permisos de administrador. Éste es usado exclusivamente a la hora de borrar un usuario de la base de datos.
  - **middleware.ts:** Se usa para actualizar el estado de una sesión de usuario. Actúa como middleware.
- **app:** Dentro del paquete app se encuentran varios subpaquetes que organizan las funcionalidades de la aplicación.
  - a) **UI (Interfaz de Usuario):** Este subpaquete agrupa todos los elementos relacionados con la presentación de la aplicación y las páginas o vistas que el usuario ve. En él, se encuentra “**home**”, donde se encuentran las distintas páginas post-login (chats, events, profile, settings, workouts, admin). A su vez, también residen aquí las páginas pre-login: login, passwordRecovery, resetPassword.
  - b) **components:** Este paquete agrupa los elementos de interfaz interactivos o modulares de la aplicación.
  - c) **contexts:** Este paquete contiene las clases que gestionan los contextos de estado en la aplicación, como la comunicación en tiempo real o el manejo de eventos dentro de la aplicación.
  - d) **utils:** Agrupa funciones o clases utilitarias que son utilizadas en distintos puntos de la aplicación. La clase “conversions” contiene funciones para la conversión de unidades de peso y de medida del sistema imperial al métrico, y viceversa.

“fonts” se encarga de gestionar el estilo de la fuente de la aplicación, y “getUserInfo” obtiene la información del usuario a partir de llamadas a *Supabase*, muy útil ya que en llamadas de API o en el propio frontend se requiere a menudo.

- e) **api:** Este paquete centraliza la lógica de comunicación con la API o servicios del backend. Cada clase representa un endpoint o grupo de funcionalidades específicas de la API.
- f) **auth:** Este paquete recoge funciones de autenticación tales como la recuperación de la contraseña en caso de olvido.

Las relaciones entre paquetes y clases se reflejan en el propio diagrama con las flechas, indicando sus respectivas dependencias.

En el caso de FitConnect, el enrutado se realiza mediante la creación de la carpeta ‘api’. Por consiguiente, se deben crear subdirectorios acorde a la ruta API que se desea crear, terminando al final siempre en un fichero llamado ‘route.ts’ (en mi caso con extensión ‘.ts’ ya que estoy utilizando TypeScript). Por ejemplo, para la ruta “/posts/getPosts” se debería de crear una estructura de directorios tal que:

```
.  
└── api/  
    └── posts/  
        └── getPosts/  
            └── route.ts
```

## Patrones de Diseño y Decisiones Arquitecturales

- **Patrón MVC (Modelo-Vista-Controlador):**

Se puede observar cómo las decisiones tomadas se alinean con el enfoque MVC.

En cuanto a la **vista**, se encuentran los paquetes UI y components. Éstos agrupan el código encargado del renderizado de la aplicación. Hablando del **controlador** se puede ver cómo la lógica que coordina las interacciones entre la vista y otros componentes se encuentra en paquetes como contexts y utils. Por último, el acceso y la gestión de datos están más relacionados con los paquetes que interactúan con *Supabase* y las API. De este modo, el paquete api agrupa funcionalidades para gestionar datos y operaciones, lo que puede considerarse parte del **modelo** en términos de encapsulación de datos y lógica de negocio.

- **Separación de Responsabilidades:**

El diagrama muestra de forma clara la separación de responsabilidades mediante la división en dos grandes áreas: **supabase** y **frontend (app)**.

### 3.1.2. Componentes y conectores

Como se ha mencionado anteriormente en la introducción a la arquitectura de la aplicación, FitConnect adopta una arquitectura de 3 tiers (presentación, servidor web y BaaS). En esta sección se describen brevemente los puertos, las interfaces y los protocolos que conectan la capa de presentación en el navegador, el servidor *Next.js* que orquesta la lógica de negocio, y el backend gestionado por *Supabase*, garantizando la integridad, la confidencialidad y la modularidad de todo el sistema.

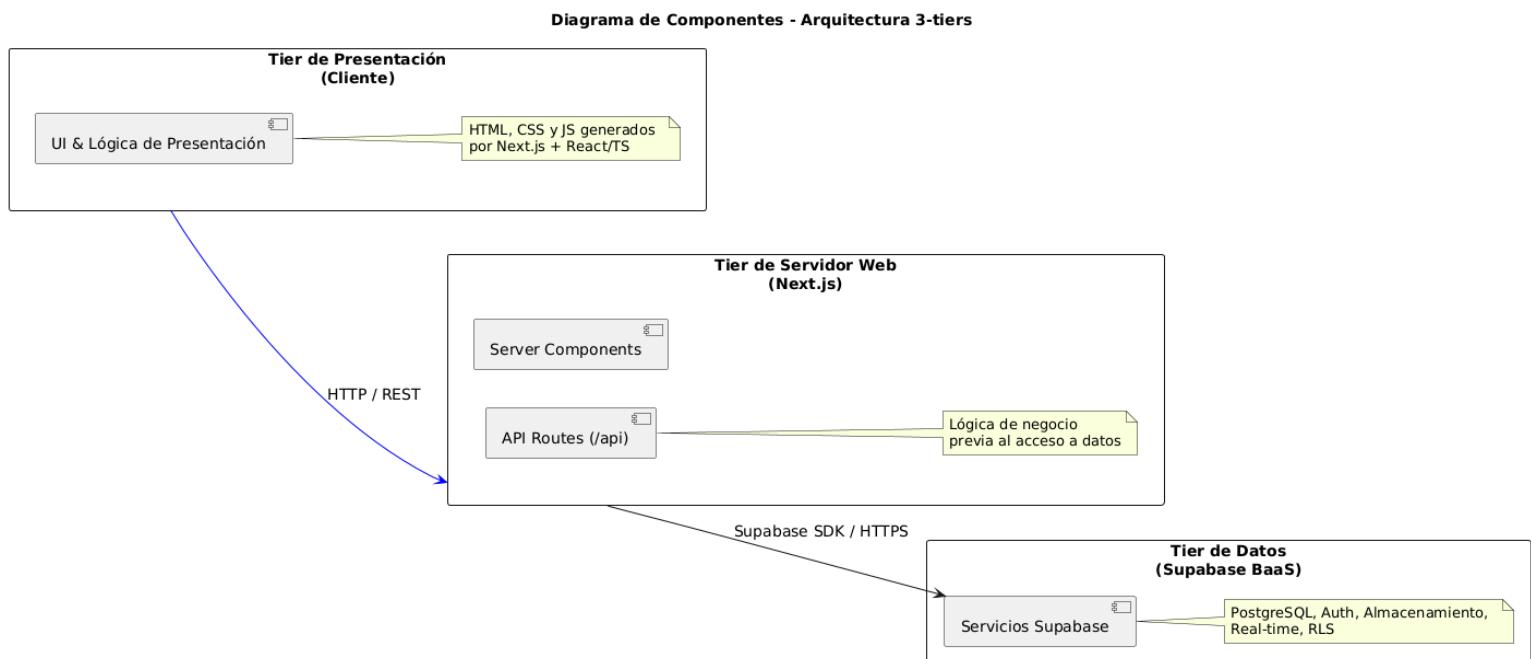


Figura 29: [Diagrama de Componentes](#)

#### Puertos:

- **Frontend (Cliente):** En desarrollo, el servidor *Next.js* atiende el cliente en el puerto 3000, entregando HTML, CSS y JavaScript del navegador.
- **Servidor Web (*Next.js* API & SSR):** Las mismas rutas de API y Server Components se exponen también en el puerto 3000, de modo que tanto la presentación como la lógica intermedia comparten el mismo punto de entrada en local.
- **Backend (*Supabase BaaS*):** *Supabase* opera de forma externa sobre el puerto seguro 443, al que el servidor *Next.js* se conecta para autenticación, consulta de datos y almacenamiento de archivos.

## Interfaces:

- **Cliente → Servidor Web:** El navegador interactúa con *Next.js* mediante solicitudes HTTP REST (fetch o axios) hacia las **API Routes** (/api/...) y renderiza páginas híbridas (Server Components + Client Components).
- **Servidor Web → Supabase:** El servidor utiliza el **Supabase SDK** o llamadas HTTPS directas a la API REST de *Supabase* para operaciones de base de datos, autenticación y tiempo real, sin exponer internamente la infraestructura.
- **Cliente (interno) de Supabase:** En algunos casos (Client Components), el frontend también puede invocar directamente el SDK de *Supabase* para suscribirse a canales en tiempo real o actualizar el estado del usuario.

## Protocolos:

- En **desarrollo local** se emplea **HTTP** para todas las comunicaciones entre el cliente y el servidor (puerto 3000), simplificando la configuración y la depuración.
- Para acceder a **Supabase**, tanto el servidor como el cliente (si está permitido por RLS) usan **HTTPS**, garantizando encriptación en tránsito y autenticación segura mediante tokens.
- El estilo de interacción en la capa de API Routes sigue el patrón **REST** (GET, POST, PUT, DELETE) mientras que la comunicación en tiempo real con *Supabase* utiliza **WebSockets** bajo TLS cuando está habilitado.

Con esta configuración, FitConnect mantiene una separación clara de responsabilidades: el cliente renderiza la UI, *Next.js* orquesta la lógica y el enrutado, y *Supabase* gestiona de forma segura la persistencia y autenticación, todo ello sobre endpoints y protocolos estándar que facilitan la escalabilidad y el mantenimiento.

## 3.2. Implementación

Un aspecto que cabe destacar de la implementación es la gestión de lectura de archivos de manera asíncrona mediante el uso de “*FileReader*”. Ésto se realiza ya que en diversas ocasiones la aplicación utiliza la funcionalidad de “Arrastrar y soltar” (*Drag & Drop*) para subir imágenes. Dicha gestión se realiza en: creación de nuevas publicaciones, modificación de publicaciones, creación de nuevos eventos, cambio de avatar de perfil. Para asegurar la unicidad, los archivos se guardan con el nombre que sigue el patrón: `\${Date.now()}-\${file.name}`.

También se debe tener en cuenta el chat en tiempo real de la aplicación. Para conseguir esta funcionalidad se ha utilizado el sistema de tiempo real de *Supabase*, el cual utiliza *WebSockets*. De este modo, cuando el usuario entra en un chat, automáticamente se suscribe recibiendo de este modo los mensajes en tiempo real.

A su vez, también el usuario es suscrito a todos los chats disponibles para poder recibir mensajes de cualquier persona en todo momento. Si alguien de un chat distinto al seleccionado como activo le envía un mensaje al usuario con la sesión iniciada, entonces aparece dicho mensaje con una vista preliminar, y aparece un circulo rojo en el chat con éste mensaje sin leer. El usuario podrá en todo momento marcar un chat como leído/no leído mediante click derecho, y seleccionando dicha opción.

Otros detalles de la implementación a destacar son:

- **Arquitectura Basada en Componentes:**

En el frontend, el uso de *React* y *Next.js* se apoya en componentes reutilizables, permitiendo una interfaz modular y escalable.

- **Uso de Server Components en *Next.js* App Router:**

La decisión de usar *Next.js* con el App Router permite que, por defecto, los componentes se rendericen en el servidor (Server Components), mejorando el rendimiento y reduciendo el tamaño del bundle enviado al cliente, mientras que se marcan explícitamente como Client Components aquellos que requieran interactividad.

- **Seguridad y Gestión de usuarios:**

La autenticación y gestión de sesiones son realizadas a través de *Supabase*. Se implementa el hashing de contraseñas (mediante *bcrypt*) para asegurar que las credenciales se almacenen de forma segura. Además, se utiliza un sistema de control de acceso basado en roles que garantiza que solo usuarios autorizados puedan acceder a funcionalidades críticas.

Para gestionar el código fuente de la aplicación, se ha utilizado [GitHub \[12\]](#) como sistema de control de versiones, manteniendo un único repositorio centralizado. Esta estrategia ha permitido llevar un registro claro y ordenado de todos los cambios. En cuanto al flujo de trabajo, me he basado en commits frecuentes y descriptivos, detallando los cambios añadidos. Además, me he asegurado de solamente realizar un commit cuando una funcionalidad está implementada y no genera fallos. No he tenido la necesidad de implementar ramas ya que siempre he dividido en partes pequeñas el trabajo, facilitando así las tareas a implementar en cada sesión de trabajo.

### 3.3. Pruebas

- Tests unitarios y de integración (**Jest**)

He utilizado **Jest** [13] para verificar los componentes de **React**. Para ello he simulado interacciones de usuario, he verificado que el resultado tras el renderizado de dichos componentes sea correcto, y me he asegurado que los componentes individuales funcionan de la manera que se espera.

- Test manual de resoluciones

Para la verificación del **RNF 2.2** se ha comprobado manualmente el funcionamiento de la aplicación en las distintas resoluciones mencionadas para ordenador, tablet y dispositivo móvil. Ésto se ha realizado gracias a las herramientas de desarrollador del navegador web, las cuales permiten al usuario seleccionar el tamaño de pantalla a utilizar.

- Sonar (**Lighthouse**)

**Lighthouse** [14] es una herramienta automatizada de código abierto para auditar la calidad de las páginas web. Se puede ejecutar en cualquier página web, pública o autenticada. Incluye auditorías de rendimiento, accesibilidad, aplicaciones web progresivas (PWA), SEO y más. Se puede acceder desde las Chrome DevTools, desde la línea de comandos o como un módulo de Node.

Las medidas de rendimiento obtenidas han sido las siguientes:

- Ordenador:

	MÉTRICA				
PANTALLA	Rendimiento	Accesibilidad	Mejores prácticas	SEO	Media
Home	83	87	100	100	92.5
Profile	85	89	100	100	93.5
Chats	78	88	100	100	91.5
Workouts	87	100	100	100	96.75
Events	86	89	100	100	93.75
Settings	85	94	100	100	94.75
Media	84	91.2	100	100	

Tabla 1: Métricas de **Lighthouse** para Ordenador

- **Móvil:**

PANTALLA	MÉTRICA				
	Rendimiento	Accesibilidad	Mejores prácticas	SEO	Media
<b>Home</b>	53	87	100	100	85
<b>Profile</b>	58	89	100	100	86.75
<b>Chats</b>	55	88	100	100	85.75
<b>Workouts</b>	55	100	100	100	88.75
<b>Events</b>	58	89	100	100	86.75
<b>Settings</b>	55	94	100	100	87.25
<b>Media</b>	55.6	91.2	100	100	

Tabla 2: Métricas de *Lighthouse* para Móvil

Como se puede observar, en el ordenador se obtienen unos mejores resultados, ya que en un principio la aplicación iba a ser diseñada únicamente para dicho dispositivo. Sin embargo, tras la implementación adicional basada en dispositivos móviles, aunque se obtienen peores resultados en las métricas, se mantienen en un 85% o superior a lo largo de las distintas pantallas del sistema.

## 4. Gestión del proyecto

Se ha seguido un desarrollo con *metodología ágil* [15]. No me he basado en una metodología específica como puede ser *Scrum* [16], pero sí he seguido ideas similares emulando los “sprints” propios de *Scrum*. Se ha trabajado en iteraciones, tocando en cada una de éstas el desarrollo frontend, rutas de la API, y modificaciones necesarias en la base de datos. De este modo, la implementación de requisitos funcionales ha sido gradual y se ha ido evolucionando desde una aplicación sencilla con pocas funcionalidades, a una cada vez más completa con más y más características finales implementadas de manera efectiva.

Destacar la utilización de *Git* [17] y *GitHub* como sistema de control de versiones. Se ha utilizado un mono-repositorio donde se encuentra tanto el frontend como el backend. Esta decisión fue tomada ya que al utilizar *Next.js* 14 era la manera más cómoda de trabajo debido a su enrutado dinámico llamado “App Router” y la manera que tiene de gestionar las rutas de la API al backend. Gracias a este mono-repositorio ha sido muy cómodo programar configurando un linter y formateador, habiendo elegido *ESLint* [18] y *Prettier* [19] respectivamente.

En cuanto a las horas de esfuerzo, como el trabajo se comenzó a finales de Septiembre de 2024, se ha trabajado progresivamente y se ha cuidado mucho la implementación realizada. A continuación se muestra una estimación del reparto de horas de esfuerzo reflejadas en FitConnect:

Apartado del proyecto	Horas de esfuerzo
Análisis de Requisitos	30
Casos de Uso	30
Bocetos de Interfaces	15
Arquitectura (diagramas y modelado)	25
Implementación (desarrollo del código fuente)	150
Pruebas	25
Redacción de la Memoria	50
<b>TOTAL</b>	<b>325</b>

Tabla 3: Horas de esfuerzo invertidas en el proyecto

## 5. Conclusiones y trabajo futuro

El desarrollo de FitConnect ha demostrado ser un reto técnico y personal enriquecedor, ya que ha permitido combinar mi pasión por la tecnología y el deporte en una solución innovadora para fomentar un estilo de vida saludable. El sistema implementado integra de forma efectiva funcionalidades complejas, tales como la autenticación segura, la gestión de perfiles, la creación y administración de rutinas de entrenamiento, publicaciones, mensajería en tiempo real, y la organización de eventos deportivos. La utilización de *Next.js* junto con *React.js* ha permitido desarrollar una aplicación web escalable, responsive y orientada a la experiencia de usuario, mientras que *Supabase* ha simplificado el manejo de la base de datos y la autenticación, aumentando la robustez del proyecto.

Entre los principales logros se destaca la implementación de un sistema integral en el que se combinan tanto funcionalidades individuales como elementos que favorecen la interacción social y el trabajo en grupo, facilitando a los usuarios el seguimiento de su progreso y la creación de una comunidad activa. Además, se ha logrado cumplir con altos estándares de accesibilidad y *diseño responsive*, asegurando que la aplicación sea usable en dispositivos de escritorio, tablets y móviles.

El proyecto, sin embargo, presenta oportunidades para continuar su desarrollo. En futuras iteraciones se podría potenciar la gamificación de la experiencia, añadiendo indicadores de consumo calórico (basados en los entrenamientos, peso y altura del usuario) y la implementación de un tracker de hábitos que fomente la constancia a través de recordatorios y desafíos entre usuarios. También se valora la incorporación de nuevas funcionalidades orientadas a la personalización y análisis detallado del rendimiento deportivo.

Desde una perspectiva personal, FitConnect ha sido una experiencia de aprendizaje integral, en la que se han enfrentado y resuelto diversas incidencias técnicas, optimizando tanto la arquitectura del software como la interacción entre sus componentes.

Uno de los principales desafíos ha sido aprender a utilizar *Supabase* desde cero. Aunque ofrece muchas facilidades para el desarrollo, hay detalles que pueden ser tediosos si no se tiene experiencia previa con esta tecnología. Por ejemplo, algunos errores que aparecen en la terminal pueden no ser del todo explicativos, lo que dificulta el diagnóstico del problema. Un caso concreto ha sido el de las *RLS (Row Level Security)*: al no modificar estas políticas adecuadamente, se generan errores al llamar a la API porque no se tienen los permisos necesarios para leer, escribir o modificar registros en una tabla. Entender cómo funcionan y cómo configurarlas correctamente ha sido un punto clave para avanzar en el desarrollo.

Este trabajo sienta las bases para futuras mejoras y contribuye a la discusión sobre cómo la tecnología puede impulsar hábitos de vida más saludables y colaborativos.

# Referencias

Las citas bibliográficas que se hayan hecho en el texto. Numeradas consecutivamente según se indica en las recomendaciones de la EINA.

**Nota:** Todas las referencias fueron consultadas por última vez el 14-4-2025.

- [1] **Hevy**  
Hevy. (2019). *Gym Workout Tracker*. Recuperado de <https://www.hevyapp.com/>
- [2] **Strong**  
Strong. (2011). *Workout Tracker*. Recuperado de <https://www.strong.app>
- [3] **Strava**  
Strava. (2023). *Strava Documentation*. Recuperado de <https://www.strava.com/>
- [4] **Next.js 14**  
Vercel. (2023). *Next.js 14 Documentation*. Recuperado de <https://nextjs.org/docs>
- [5] **React.js 18**  
React. (2022). *React Documentation*. Recuperado de <https://react.dev/learn>
- [6] **Diseño Responsive**  
Marcotte, E. (2010). *Responsive Web Design*. A List Apart. Recuperado de <https://alistapart.com/article/responsive-web-design>
- [7] **TailwindCSS**  
Tailwind CSS. (2024). *Tailwind CSS Documentation*. Recuperado de <https://tailwindcss.com/docs>
- [8] **Supabase**  
Supabase. (2020). *Supabase Documentation*. Recuperado de <https://supabase.com/docs>
- [9] **Firebase**  
Firebase. (2011). *Firebase Documentation*. Recuperado de <https://firebase.google.com/docs>
- [10] **Express.js**  
Express.js. (2010). *Express Documentation*. Recuperado de <https://expressjs.com/en/starter/installing.html>

- **[11] WebSockets**  
Mozilla Developer Network. (2008). *The WebSocket API (WebSockets)*. Recuperado de [https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)
- **[12] GitHub**  
GitHub. (2008). *GitHub Documentation*. Recuperado de <https://docs.github.com/>
- **[13] Jest**  
Jest. (2023). *Jest Documentation*. Recuperado de <https://jestjs.io/docs/getting-started>
- **[14] Lighthouse**  
Lighthouse. (2023). Open-source, automated tool for improving the quality of web pages. Recuperado de <https://developers.google.com/web/tools/lighthouse>
- **[15] Metodología ágil**  
Agile Manifesto. (2001). *Manifesto for Agile Software Development*. Recuperado de <https://agilemanifesto.org/>
- **[16] Scrum**  
Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. Recuperado de <https://scrumguides.org/>
- **[17] Git**  
Chacon, S., & Straub, B. (2014). *Pro Git (2<sup>a</sup> ed.)*. Recuperado de <https://git-scm.com/book/en/v2>
- **[18] ESLint**  
ESLint. (2013). *Find and fix problems in your JavaScript code*. Recuperado de <https://eslint.org/>
- **[19] Prettier**  
Prettier. (2016). *Opinionated Code Formatter*. Recuperado de <https://prettier.io/docs/>

# Anexos

## Anexo I: Casos de uso

Inicio de sesión

Referencia: [Figura 6.](#)

Actor: Usuario y Administrador

Objetivo: Acceder a la cuenta en el sistema.

Precondiciones:

- El usuario debe estar registrado y la cuenta verificada.

### **Flujo principal:**

1. El usuario accede a la página de inicio de sesión.
2. Ingresa su nombre de usuario y contraseña.
3. Al presionar “Log In”, el sistema valida las credenciales.
4. Si son correctas, el usuario es redirigido a la pantalla principal de la aplicación.

### **Flujos alternativos:**

#### **- Error en credenciales:**

- 2a. Si el usuario ingresa datos incorrectos o deja campos vacíos y presiona “Log In”, el sistema muestra un mensaje de error.
- 2b. El usuario corrige la información y el flujo continúa por el paso 3 del flujo principal.

#### **- Usuario no registrado:**

- 2a. Si el usuario no dispone de cuenta, puede presionar “Sign Up” para ser redirigido a la página de registro, finalizando este caso de uso y dando inicio al caso [“Creación de Cuenta”](#).

#### **- Olvido de contraseña:**

- 2a. Si el usuario no recuerda su contraseña, puede presionar “Forgot password?”, lo que lo redirige a la pantalla de recuperación de contraseña, finalizando este caso de uso y dando inicio al caso [“Recuperación de Contraseña”](#).

## Creación de cuenta

Referencia: [Figura 7.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario crear una nueva cuenta en el sistema.

Precondiciones:

- El usuario se encuentra en la pantalla de inicio de sesión.

### Flujo principal:

1. El usuario, desde la pantalla de inicio de sesión, presiona el botón “Sign Up”.
2. El sistema redirige al usuario a la pantalla de registro.
3. El usuario completa los campos requeridos:
  - a. Nombre de usuario
  - b. Correo electrónico
  - c. Contraseña
  - d. Confirmación de contraseña.
4. El usuario presiona el botón “Sign Up” para enviar el formulario.
5. El sistema valida la información ingresada; si todos los datos cumplen con los requisitos, se crea la cuenta y se envía un correo electrónico automático de verificación de la cuenta. Una vez el usuario entra en dicho enlace, éste es autenticado, pudiendo entonces iniciar sesión en el sistema.

### Flujos alternativos:

#### - Datos incorrectos o incompletos:

- 4a. Al presionar “Sign Up”, se detecta que:

- El nombre de usuario o el correo electrónico están incompletos o no tienen el formato correcto.
- Las contraseñas no cumplen con los requisitos mínimos de seguridad, no coinciden entre sí o están sin llenar.

- 4b. El sistema muestra un mensaje de error en pantalla.

- 4c. Una vez que el usuario corrige los datos erróneos, el flujo se retoma en el paso 4 y continúa el proceso de creación de cuenta de manera normal.

## Recuperación de contraseña

Referencias: [Figura 8](#) y [Figura 9.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario recuperar su contraseña mediante el envío de un correo electrónico con un enlace de restablecimiento.

Precondiciones:

- El usuario debe estar en la pantalla de inicio de sesión y tener una cuenta registrada.

### Flujo principal:

1. El usuario presiona el enlace “Forgot password?” en la pantalla de inicio de sesión.
2. El sistema redirige al usuario a la pantalla de recuperación de contraseña.
3. El usuario ingresa su correo electrónico y presiona el botón “CONFIRM”.
4. El sistema envía un correo electrónico con un enlace de restablecimiento y muestra un mensaje informativo al usuario.
5. El usuario sigue el enlace, cambia su contraseña y, posteriormente, inicia sesión satisfactoriamente.

#### **Flujos alternativos:**

- **Correo ingresado de forma errónea**
  - a. Si el usuario ingresa un correo incorrecto o mal escrito, no se envía el correo de confirmación.
  - b. El sistema notifica el error y el usuario corrige el correo
  - c. El usuario presiona “RE-SEND” y, al ingresar el correo correctamente, el sistema envía el correo y el flujo continúa en el paso 5.

#### Creación de una publicación

Referencia: [Figura 10](#) y [Figura 11](#).

Actor: Usuario y Administrador

Objetivo: Permitir al usuario crear una nueva publicación en el feed principal.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla del feed principal.

#### **Flujo principal:**

1. El usuario presiona el botón “+” ubicado en la esquina inferior derecha del feed.
2. El sistema redirige al usuario a la pantalla de creación de publicación.
3. El usuario tiene la opción de subir una imagen y añadir una descripción.
4. El usuario presiona el botón “Post”.
5. El sistema valida la información; si es correcta, crea la publicación y la muestra en el feed principal.

#### **Flujos alternativos:**

- **Contenido insuficiente**
  - a. Si el usuario presiona “Post” sin haber subido una imagen, el sistema muestra un mensaje de error resaltando la omisión.
  - b. El usuario añade el contenido faltante y el flujo continúa en el paso 4.

#### Visualización del perfil

Referencia: [Figura 12](#).

Actor: Usuario y Administrador

Objetivo: Permitir al usuario consultar la información de su perfil personal.

Precondiciones:

- El usuario ha iniciado sesión.

**Flujo principal:**

1. El usuario selecciona la opción “Profile” en el menú lateral izquierdo.
2. El sistema redirige al usuario a la pantalla de perfil, donde se muestra su información (publicaciones, nombre de usuario, seguidores y seguidos).
3. El caso de uso concluye cuando el usuario navega a otra sección.

Edición de una publicación

*Referencia: [Figura 13](#).*

Actor: Usuario y Administrador

Objetivo: Permitir al usuario editar una publicación propia.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de visualización de su perfil.

**Flujo principal:**

1. El usuario selecciona el ícono de edición (bolígrafo) en la esquina superior derecha de una publicación.
2. El sistema muestra una ventana emergente con las opciones de edición (modificar imagen y/o descripción).
3. El usuario realiza las modificaciones deseadas y presiona “Save Changes”.
4. El sistema valida y guarda los cambios, actualizando la publicación en la pantalla.

**Flujos alternativos:**

- **Cancelación de la edición**

- a. Si el usuario presiona “Cancel” en la ventana emergente, el sistema descarta los cambios y cierra la ventana, finalizando el caso de uso sin modificar la publicación.

Consultar chats

*Referencia: [Figura 14](#).*

Actor: Usuario y Administrador

Objetivo: Permitir al usuario ver la lista de chats y consultar el contenido de uno seleccionado.

Precondiciones:

- El usuario ha iniciado sesión.

**Flujo principal:**

1. El usuario selecciona la opción “Chats” en el menú lateral izquierdo.
2. El sistema muestra la lista de chats (individuales y grupales).

3. El usuario selecciona un chat de la lista.
4. El sistema muestra en la parte derecha el contenido del chat seleccionado.

#### **Flujos alternativos:**

- **Sin chats existentes**
  - a. Si el usuario no tiene chats iniciados, el sistema muestra un mensaje informativo indicando la ausencia de conversaciones.

### Buscar chat

Referencia: (Sin figura específica; parte de la funcionalidad de chats)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario buscar un chat específico mediante un término de búsqueda.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de chats.

#### **Flujo principal:**

1. El usuario presiona el buscador (rectángulo con el icono “Buscar” con lupa) en la parte superior de la pantalla de chats.
2. El sistema permite al usuario escribir un término y muestra en pantalla los resultados en tiempo real
3. El usuario selecciona el chat deseado de los resultados.
4. El sistema abre el chat seleccionado.

#### **Flujos alternativos:**

- **Sin resultados**
  - a. Si no se encuentran chats que coincidan con el término, el sistema muestra un mensaje informativo indicando la ausencia de coincidencias.

### Enviar mensaje

Referencia: (Sin figura específica; parte de la funcionalidad de chats)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario enviar un mensaje en un chat activo.

Precondiciones:

- El usuario ha iniciado sesión y ha seleccionado un chat.

#### **Flujo principal:**

1. El usuario escribe un mensaje en el campo “Type a message...”.
2. El usuario presiona la tecla “ENTER” o hace clic en el botón de envío (flecha).
3. El sistema envía el mensaje y lo muestra en el historial del chat.

#### **Flujos alternativos:**

- **Error al enviar el mensaje**
  - a. Si el mensaje no se envía correctamente, el sistema muestra un mensaje de error y permite reintentar el envío.

## Crear nuevo grupo de chat

Referencia: [Figura 15.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario crear un grupo de chat nuevo.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de chats.

### Flujo principal:

1. El usuario presiona el botón “Create New Group”.
2. El sistema muestra una ventana emergente con las opciones para crear un grupo.
3. El usuario ingresa el nombre del grupo y selecciona los integrantes deseados.
4. El usuario presiona el botón “Create Group”.
5. El sistema valida la información, crea el grupo y lo añade a la lista de chats.

### Flujos alternativos:

- **Datos incompletos**
  - a. Si el usuario presiona “Create Group” sin haber ingresado un nombre o sin seleccionar integrantes, el sistema muestra un mensaje de error indicando el problema.
  - b. El usuario corrige los datos y el flujo continúa en el paso 4.

## Salir de un grupo de chat

Referencia: [Figura 16.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario abandonar un grupo de chat de manera definitiva.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en un grupo de chat activo.

### Flujo principal:

1. El usuario presiona el botón “Leave Group” en la pantalla del chat grupal.
2. El sistema elimina el grupo de la interfaz del usuario y cierra el chat (o lo redirige a otro chat activo).

## Listado de rutinas de entrenamiento

Referencia: [Figura 17.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario visualizar sus rutinas de entrenamiento

Precondiciones:

- El usuario ha iniciado sesión.

### Flujo principal:

1. El usuario selecciona la opción “Workouts” en el menú lateral izquierdo.
2. El sistema redirige al usuario a una pantalla que muestra un listado de sus rutinas de entrenamiento.
3. El usuario puede navegar por el listado.
4. El caso de uso finaliza cuando el usuario cambia de pantalla o selecciona una rutina para ver sus detalles.

**Flujos alternativos:**

- **Sin rutinas creadas**
  - a. Si el usuario no tiene rutinas, el sistema muestra un mensaje informativo indicando la ausencia de éstas, y un botón para que cree su primera rutina.

Ver rutina completa

Referencia: [Figura 19.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario ver en detalle una rutina de entrenamiento.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de listado de rutinas.

**Flujo principal:**

1. El usuario selecciona el botón “View Full Workout” en una rutina listada.
2. El sistema redirige al usuario a una pantalla que muestra el detalle completo de la rutina (ejercicios, repeticiones, series, etc.).
3. El caso de uso concluye cuando el usuario navega a otra pantalla.

Editar rutina de entrenamiento

Referencia: [Figura 20.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario modificar una rutina de entrenamiento existente.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de “Ver Rutina Completa”.

**Flujo principal:**

1. El usuario presiona el botón “Edit Workout”.
2. El sistema cambia la pantalla a modo edición, permitiendo modificar o agregar ejercicios y otros campos editables.
3. El usuario realiza las modificaciones deseadas y presiona “Save Changes”.
4. El sistema guarda los cambios y vuelve al modo de solo lectura.

**Flujos alternativos:**

- **Cancelación de la edición**
  - a. Si el usuario presiona “Cancel”, el sistema descarta los cambios y vuelve al modo de solo lectura sin modificar la rutina.

## Eliminar rutina de entrenamiento

Referencia: (Extiende “[Ver Rutina Completa](#)”)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario eliminar de forma definitiva una rutina de entrenamiento.

Precondiciones:

- El usuario ha iniciado sesión y visualiza una rutina completa.

### Flujo principal:

1. El usuario selecciona el botón “Delete Workout”.
2. El sistema elimina la rutina y redirige al usuario al listado de rutinas.

## Listado eventos

Referencia: [Figura 21.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario visualizar los eventos disponibles.

Precondiciones:

- El usuario ha iniciado sesión.

### Flujo principal:

1. El usuario presiona el buscador (ícono de lupa) en la parte superior de la pantalla de eventos.
2. El sistema redirige al usuario a una pantalla que muestra un listado de eventos disponibles.
3. El usuario navega por el listado de eventos.
4. El caso de uso finaliza cuando el usuario cambia de pantalla o interactúa con un evento.

### Flujos alternativos:

- **Sin eventos disponibles**
  - a. Si no existen eventos, el sistema muestra un mensaje informativo indicando la ausencia de eventos.

## Búsqueda de evento

Referencia: (Sin figura específica; parte de la funcionalidad de eventos)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario buscar eventos mediante palabras clave.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de eventos.

### Flujo principal:

1. El usuario presiona el buscador (rectángulo con el ícono “Buscar” con lupa) en la parte superior de la pantalla de eventos.

2. El sistema permite al usuario escribir un término y muestra en pantalla los resultados en tiempo real.
3. El usuario selecciona el evento deseado de los resultados y el sistema posiciona dicho evento en primera posición del listado.

## Creación nuevo evento

Referencia: [Figura 22.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario crear un nuevo evento.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de eventos.

### **Flujo principal:**

1. El usuario presiona el botón “+” ubicado en la parte inferior derecha de la pantalla de eventos.
2. El sistema muestra una ventana emergente para la creación del evento.
3. El usuario completa los campos requeridos:
  - a. Nombre del evento
  - b. Descripción
  - c. Localización
  - d. Fecha y hora
  - e. Cantidad de aforo
  - f. Imagen representativa
4. El usuario presiona el botón “Create Event”.
5. El sistema valida la información, crea el evento, cierra la ventana emergente y añade el nuevo evento al listado.

### **Flujos alternativos:**

- **Datos incompletos**
  - a. Si el usuario presiona “Create Event” sin completar alguno de los campos requeridos, el sistema muestra un mensaje de error en cada campo faltante (resaltándolos en rojo).
  - b. El usuario corrige los datos y el flujo continúa en el paso 4.

## Unirse a un evento

Referencia: (Sin figura específica; parte de la funcionalidad de eventos)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario unirse a un evento.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de eventos.
- El usuario no forma parte de algún evento del listado.

### **Flujo principal:**

1. El usuario presiona el botón “Join” (presente en cada evento del listado).
2. El sistema muestra un mensaje emergente indicando al usuario que se ha unido satisfactoriamente.
3. El sistema añade al usuario a dicho evento, y cambia el botón “Join” por “Leave” para realizar la acción contraria, acabando así el caso de uso.

## Salirse de un evento

Referencia: (Sin figura específica; parte de la funcionalidad de eventos)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario salirse de un evento del que forma parte.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de eventos.
- El usuario forma parte de algún evento del listado.

### **Flujo principal:**

1. El usuario presiona el botón “Leave” (presente en cada evento del listado).
2. El sistema muestra un mensaje emergente indicando al usuario que se ha salido satisfactoriamente.
3. El sistema añade al usuario a dicho evento, y cambia el botón “Leave” por “Join” para realizar la acción contraria, acabando así el caso de uso.

## Cambiar foto perfil

Referencia: [Figura 23.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario cambiar su foto de perfil por otra distinta, o simplemente eliminar su foto de perfil.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de ajustes.

### **Flujo principal:**

1. El usuario presiona el botón “Change Avatar”.
2. El sistema abre el explorador de archivos para que el usuario seleccione la imagen correspondiente.
3. El sistema cambia la foto de perfil de manera provisional.
4. El usuario presiona “Save Changes”, haciendo de este modo efectivo el cambio en el sistema.

## Realizar cambios en ajustes

Referencia: [Figura 23.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario realizar cambios en su nombre de usuario, biografía, peso, altura, y/o contraseña.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de ajustes.

**Flujo principal:**

1. El usuario realiza los cambios deseados en cualquiera de los campos mencionados anteriormente.
2. El usuario presiona “Save Changes”, haciendo de este modo efectivo el cambio en el sistema.

**Flujos alternativos:**

- **Contraseña incorrecta**

- a. Si el usuario cambia la contraseña y presiona el botón “Save Changes” siendo que la contraseña actual es incorrecta, aparece un mensaje de error resaltando en rojo que no coincide la contraseña actual con la escrita.
- b. El usuario corrige la contraseña y el flujo continúa en el paso 2.

## Borrar cuenta

Referencia: [Figura 23.](#)

Actor: Usuario y Administrador

Objetivo: Permitir al usuario eliminar su cuenta de manera definitiva en el sistema.

Precondiciones:

- El usuario ha iniciado sesión y se encuentra en la pantalla de ajustes.

**Flujo principal:**

1. El usuario presiona el botón “Delete Account”.
2. El sistema muestra un mensaje emergente indicando al usuario si está seguro de querer borrar su cuenta de manera definitiva.
3. El usuario presiona “OK”, realizando efectivo el borrado de su cuenta.

**Flujos alternativos:**

- **Cancelación del borrado**

- a. Si el usuario, tras el paso 2 del flujo principal, presiona “Cancel”, entonces se cierra el mensaje emergente y se termina el flujo sin ningún cambio en el sistema.

## Eliminar usuario del sistema

Referencia: [Figura 24.](#)

Actor: Administrador

Objetivo: Permitir al administrador eliminar cualquier cuenta de manera definitiva en el sistema.

Precondiciones:

- El usuario ha iniciado sesión y tiene permisos de administrador.

**Flujo principal:**

1. El usuario selecciona “Manage Users” en el menú lateral izquierdo.
2. El usuario presiona el buscador (rectángulo con el icono “Buscar” con lupa) con el texto “Search users...”.
3. El sistema permite al usuario escribir un término y muestra en pantalla los resultados en tiempo real.
4. El usuario selecciona el usuario, o una lista de éstos, de los resultados.
5. El usuario presiona el botón “Delete Selected Users”, haciendo efectivo el borrado de los usuarios del sistema.

Eliminar publicación o comentario del sistema

Referencia: [Figura 25.](#)

Actor: Administrador

Objetivo: Permitir al administrador eliminar cualquier publicación/es, o comentario/s de manera definitiva en el sistema.

Precondiciones:

- El usuario ha iniciado sesión y tiene permisos de administrador.

**Flujo principal:**

1. El usuario selecciona “Manage Posts” en el menú lateral izquierdo.
2. El usuario presiona el buscador (rectángulo con el icono “Buscar” con lupa) con el texto “Search posts...”.
3. El sistema permite al usuario escribir un término y muestra en pantalla los resultados en tiempo real.
4. El usuario selecciona la publicación, o una lista de éstas; o bien el comentario, o una lista de éstos, de los resultados.
5. El usuario presiona el botón “Delete Selected Posts” y/o “Delete Selected Comments”, haciendo efectivo el borrado de las publicaciones/comentarios del sistema.

Eliminar evento del sistema

Referencia: [Figura 26.](#)

Actor: Administrador

Objetivo: Permitir al administrador eliminar cualquier evento de manera definitiva en el sistema.

Precondiciones:

- El usuario ha iniciado sesión y tiene permisos de administrador.

**Flujo principal:**

1. El usuario selecciona “Manage Events” en el menú lateral izquierdo.

2. El usuario presiona el buscador (rectángulo con el icono “Buscar” con lupa) con el texto “Search events...”.
3. El sistema permite al usuario escribir un término y muestra en pantalla los resultados en tiempo real.
4. El usuario selecciona el evento, o una lista de éstos, de los resultados.
5. El usuario presiona el botón “Delete Selected Events”, haciendo efectivo el borrado de los eventos del sistema.

## Anexo II: Interfaces de usuario

A continuación se incluyen los prototipos iniciales de las pantallas de FitConnect. Éstos bocetos iniciales se han creado con la herramienta *Excalidraw* [x], a través de su página web.

En las GUI reales de la aplicación web se ha necesitado realizar algún ajuste ya que conforme se ha realizado la implementación del sistema, se han modificado levemente algunos requisitos tanto funcionales como no funcionales.

Un ejemplo de ésto, sería el [RNF 2.2](#), el cual fomenta la compatibilidad entre distintos dispositivos. De este modo, en los bocetos iniciales no se han tenido en cuenta las resoluciones de pantalla para versiones móviles o tablet de la aplicación.

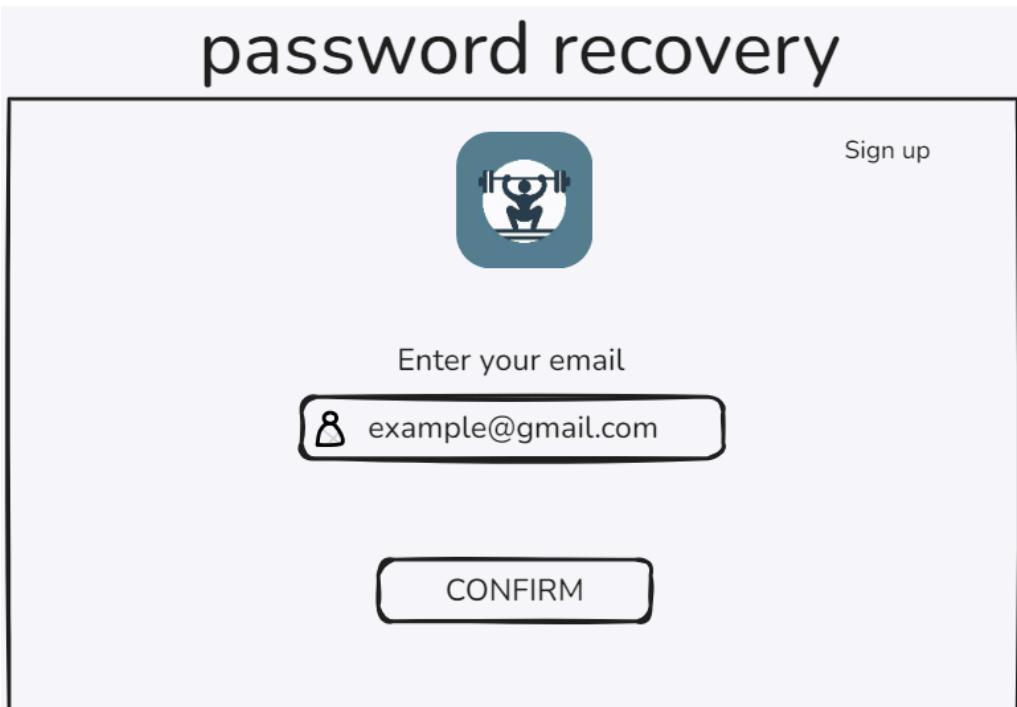
A su vez, como se puede apreciar en la [Figura 18](#) (pantalla de ajustes), existen múltiples configuraciones de preferencia de notificaciones. Ésto se ha eliminado en la versión final de la aplicación ya que, bajo mi punto de vista y siguiendo la filosofía de FitConnect, he preferido que no existiese la posibilidad de envío de notificaciones para reducir las posibilidades de adicción al móvil/tablet/ordenador. Es una decisión personal, ya que pienso que hoy en día estamos sobrecargados con cantidades excesivas de notificaciones y al suprimirlas, se consigue una gran reducción del tiempo de uso de pantalla.



[Figura 6: Boceto pantalla Log In](#)



[Figura 7: Boceto pantalla creación de cuenta](#)



[Figura 8: Boceto pantalla recuperación contraseña](#)

# password recovery (2)

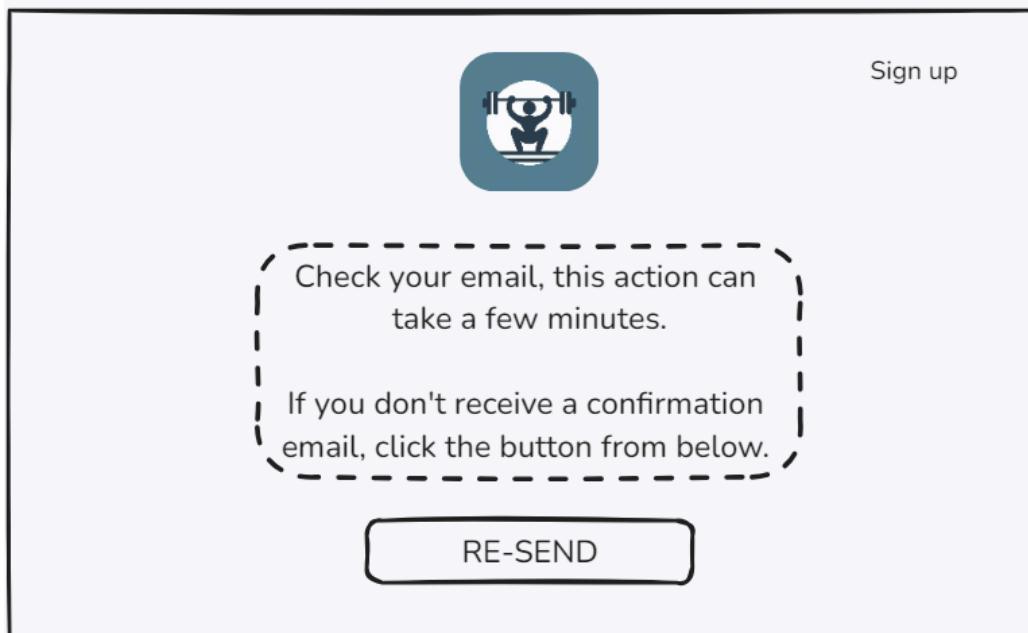


Figura 9: Boceto pantalla recuperación de contraseña (2)

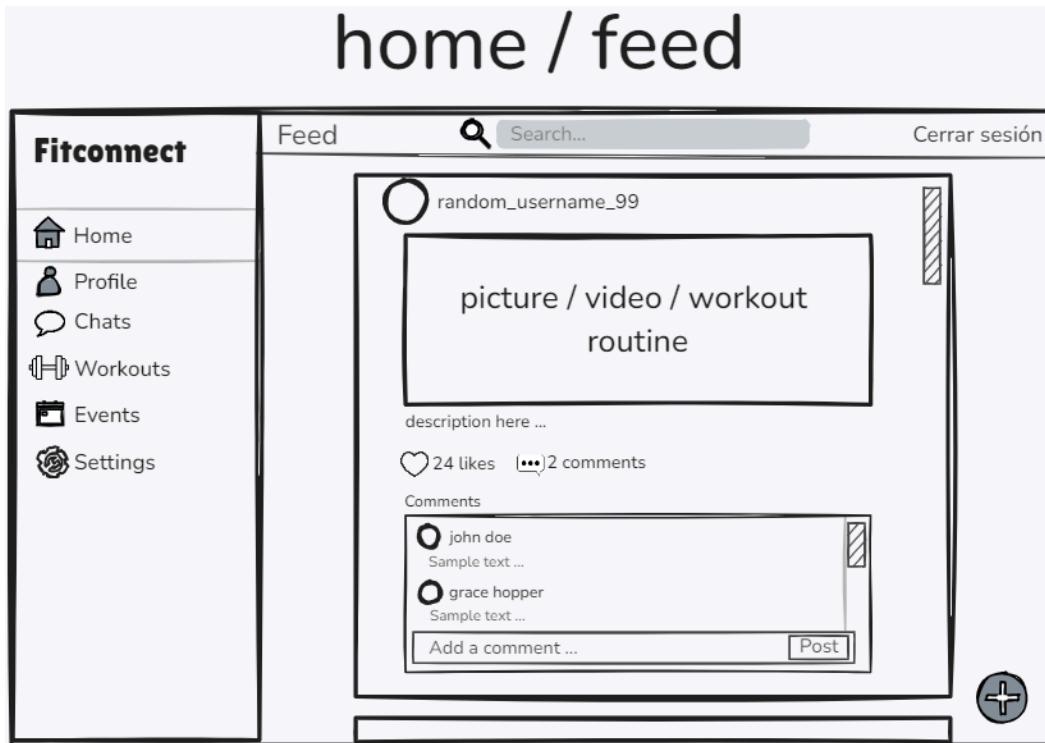


Figura 10: Boceto pantalla home

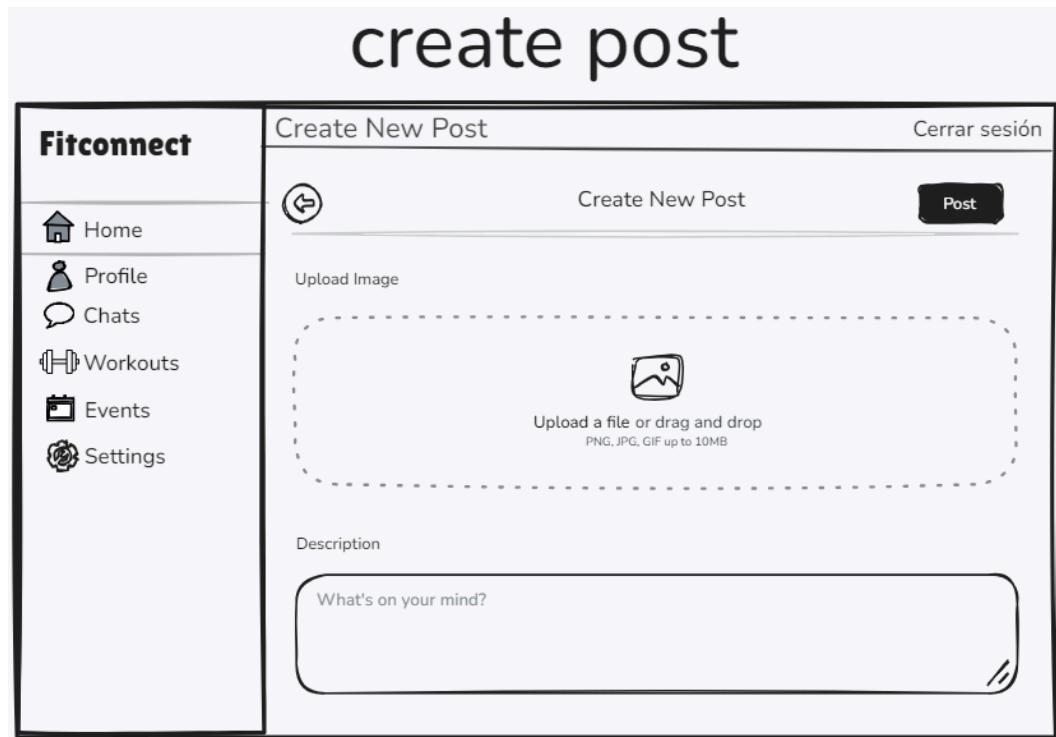


Figura 11: Boceto pantalla creación de una publicación

# profile overview

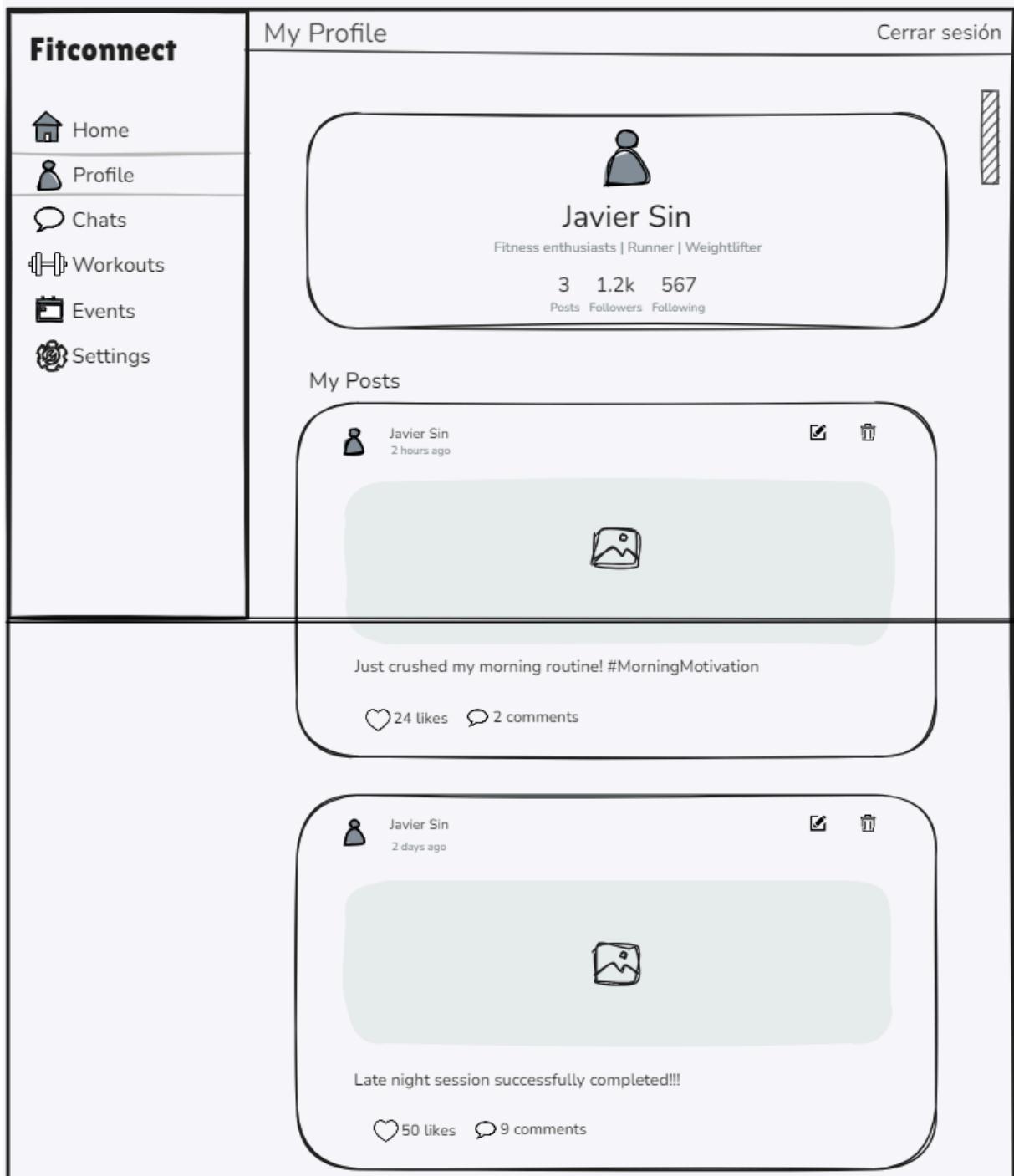


Figura 12: Boceto pantalla visualización del perfil

# post edition

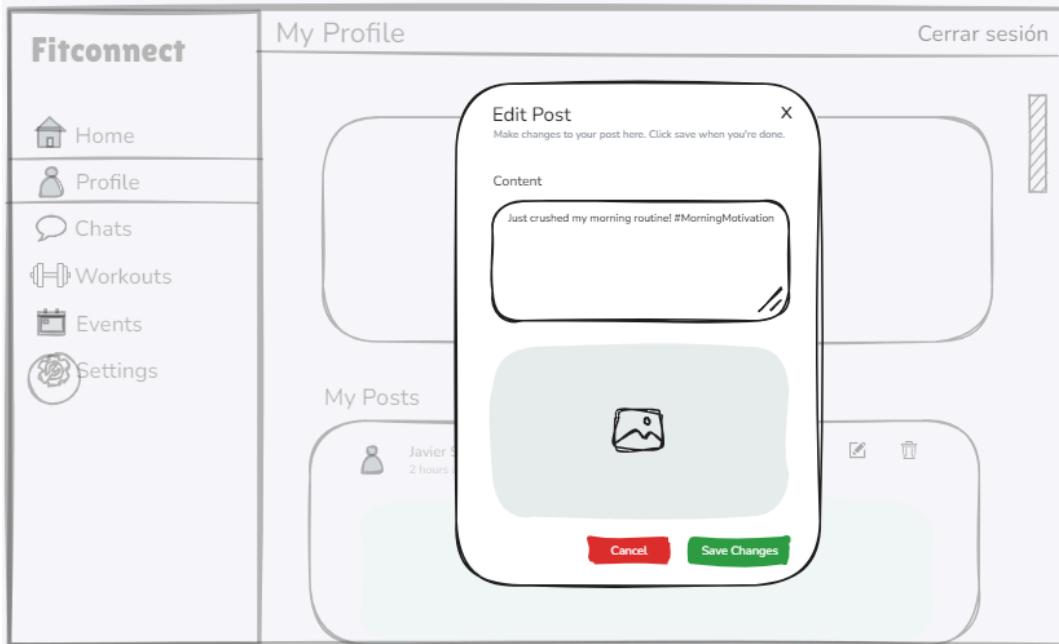
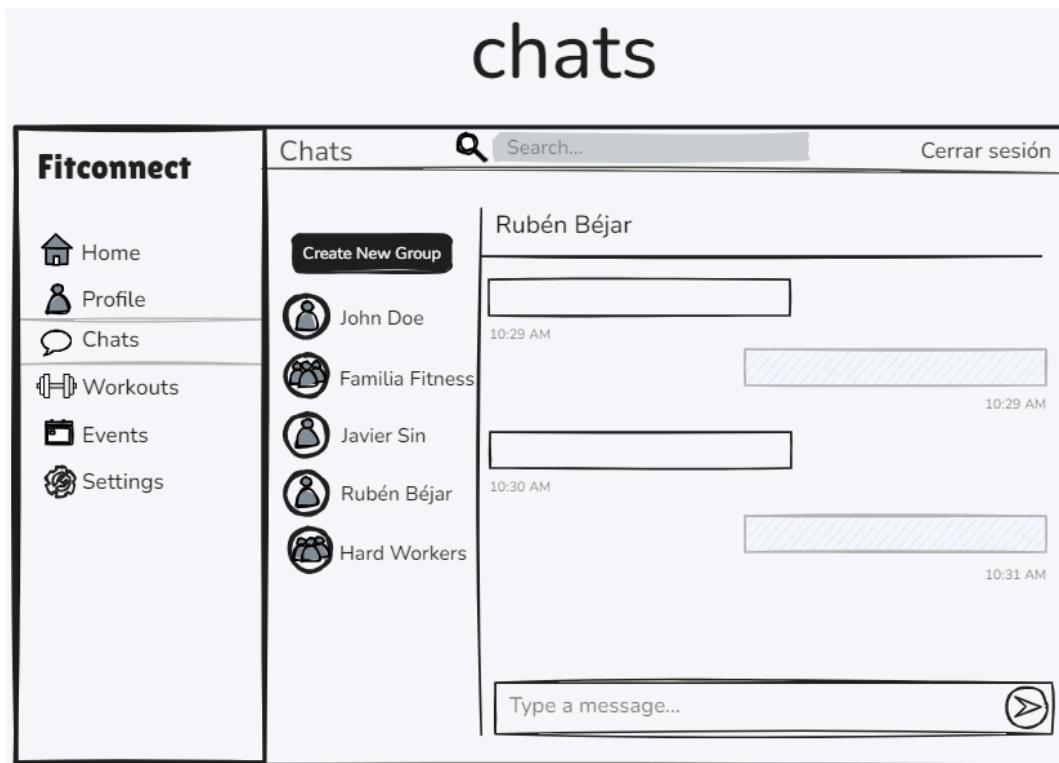
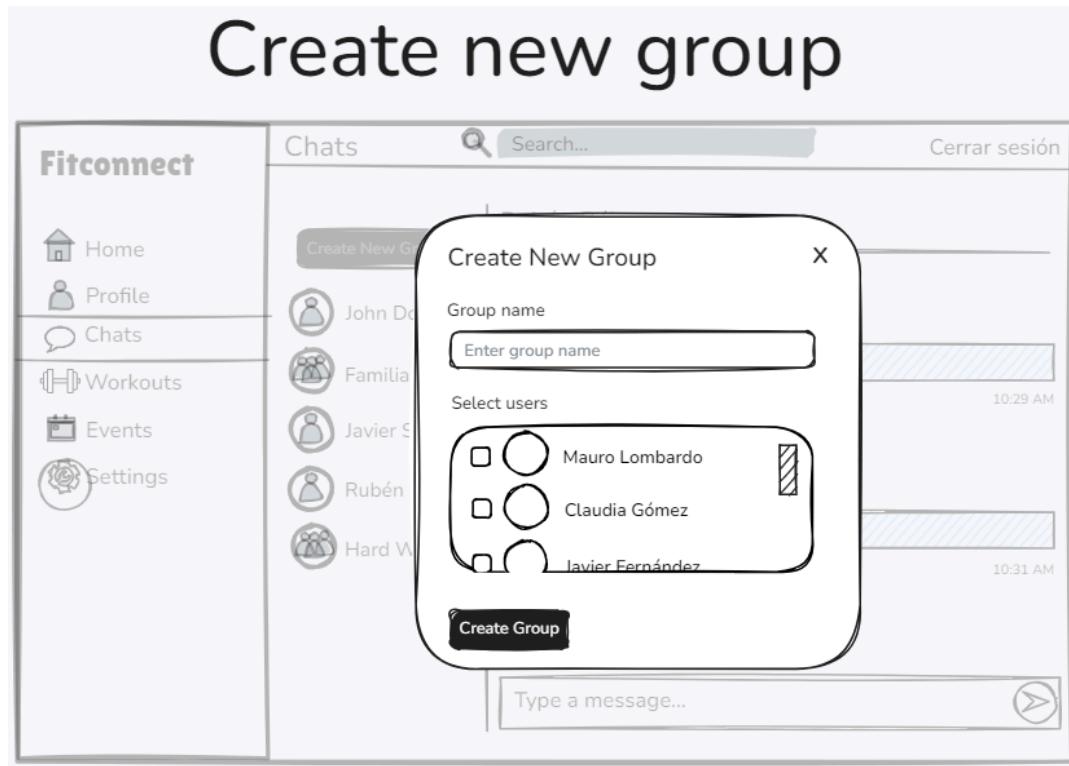


Figura 13: Boceto pantalla edición de una publicación



[Figura 14: Boceto pantalla chats](#)



[Figura 15: Boceto pantalla creación de nuevo grupo](#)

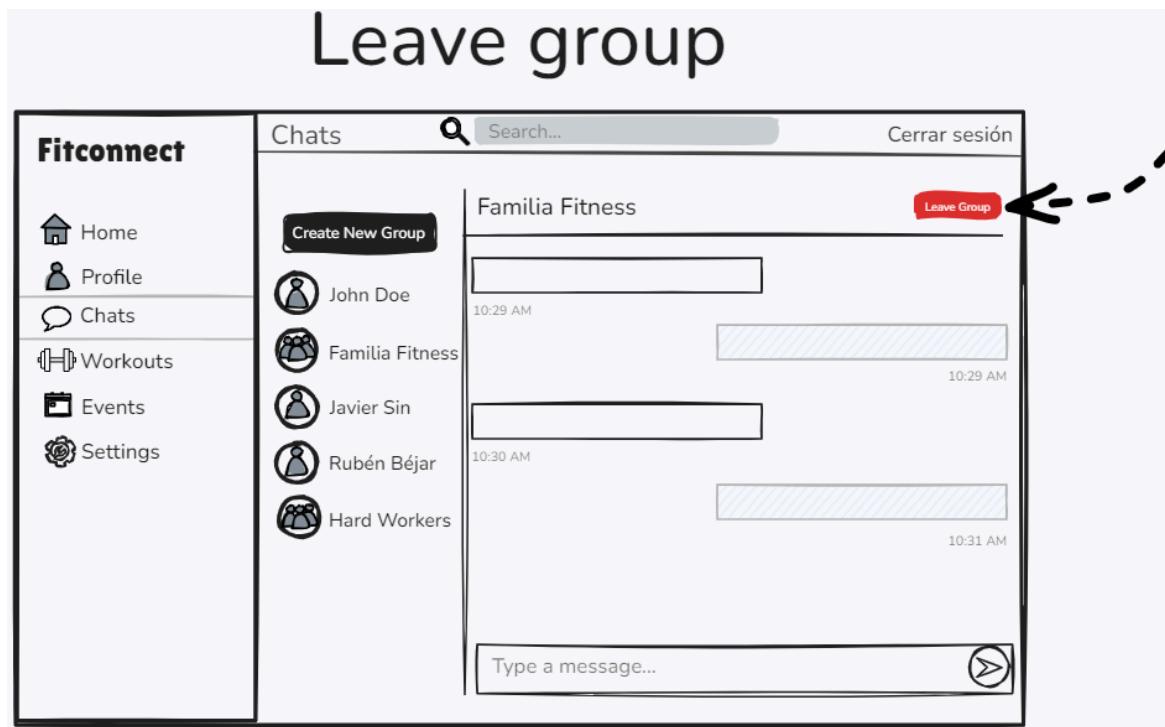


Figura 16: Boceto pantalla salir de un grupo de chat

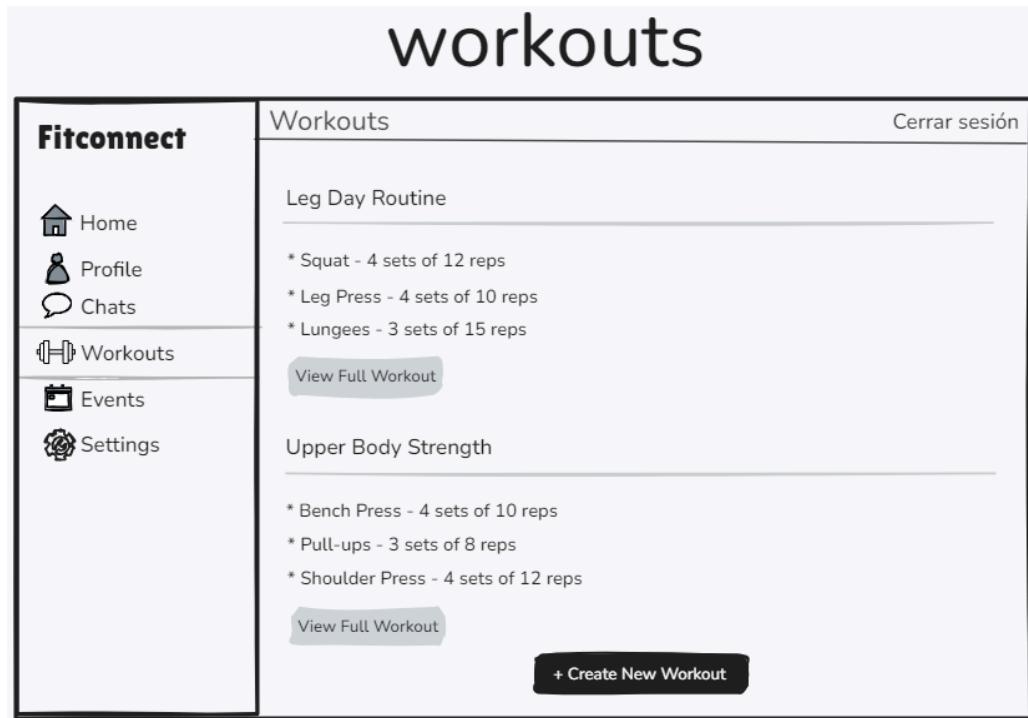


Figura 17: Boceto pantalla visualización rutinas

# create new workout

The screenshot shows a mobile application interface for 'Fitconnect'. On the left is a vertical sidebar with icons for Home, Profile, Chats, Workouts (selected), Events, and Settings. The main area has a header 'Workouts' and a 'Cerrar sesión' button. Below is a list titled 'Leg Day Routine' with a back arrow icon. A table for adding exercises has columns: exercise, sets, repetitions, and weight / time. The first row is fixed with '-' values. A blue button 'Add a new exercise' is at the bottom right of the table. At the bottom are green 'Save Changes' and red 'Cancel' buttons. A hand-drawn style arrow points from the text 'The texts will be INPUT areas so that the user can modify the fields, except from the first row (fixed)' to the 'sets' column of the table.

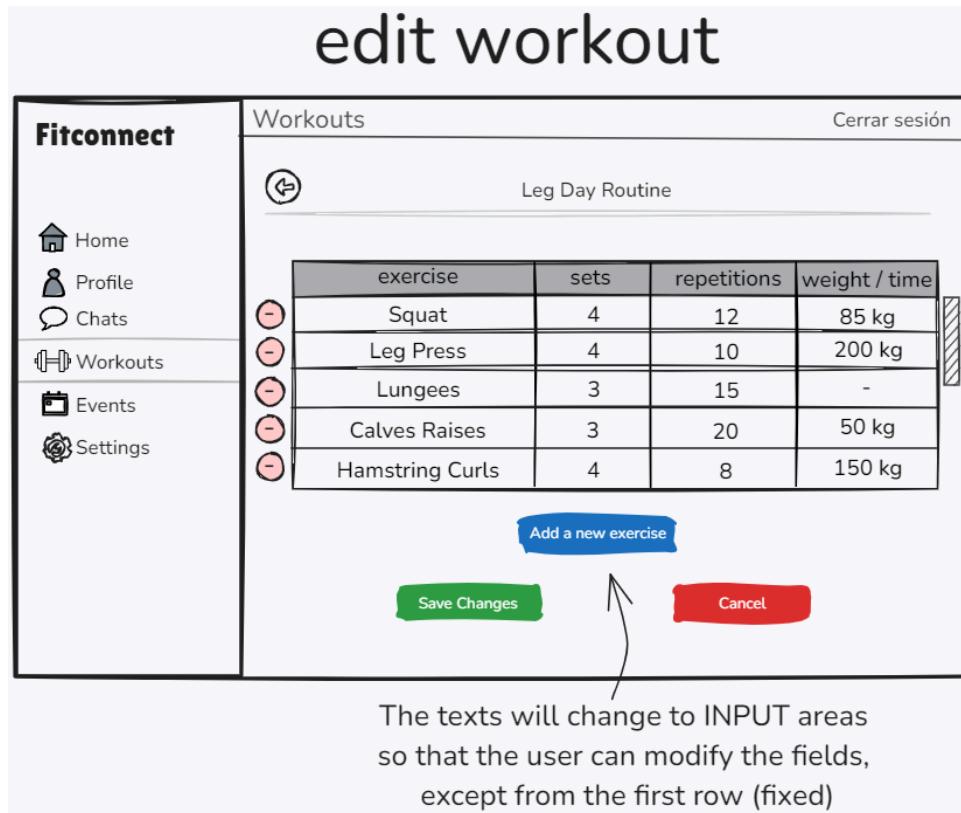
The texts will be INPUT areas  
so that the user can modify the fields,  
except from the first row (fixed)

Figura 18: Boceto pantalla creación de nueva rutina

# view full workout

The screenshot shows a mobile application interface for 'Fitconnect'. On the left is a vertical sidebar with icons for Home, Profile, Chats, Workouts (selected), Events, and Settings. The main area has a header 'Workouts' and a 'Cerrar sesión' button. Below is a list titled 'Leg Day Routine' with a back arrow icon. A table lists exercises with their details: Squat (4 sets, 12 repetitions, 85 kg), Leg Press (4 sets, 10 repetitions, 200 kg), Lungees (3 sets, 15 repetitions, -), Calves Raises (3 sets, 20 repetitions, 50 kg), Hamstring Curls (4 sets, 8 repetitions, 150 kg), and Treadmill (1 set, - repetitions, 15 min). At the bottom are grey 'Edit Workout' and red 'Delete Workout' buttons.

[Figura 19: Boceto pantalla rutina completa](#)



[Figura 20: Boceto pantalla editar una rutina](#)

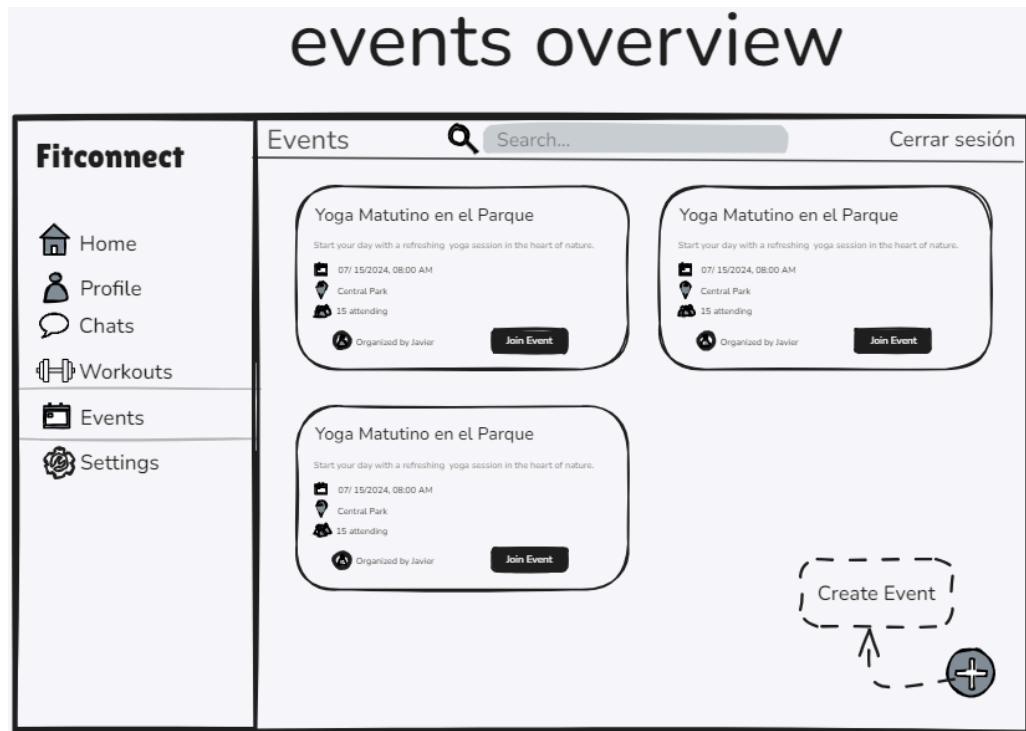


Figura 21: Boceto pantalla listado de eventos

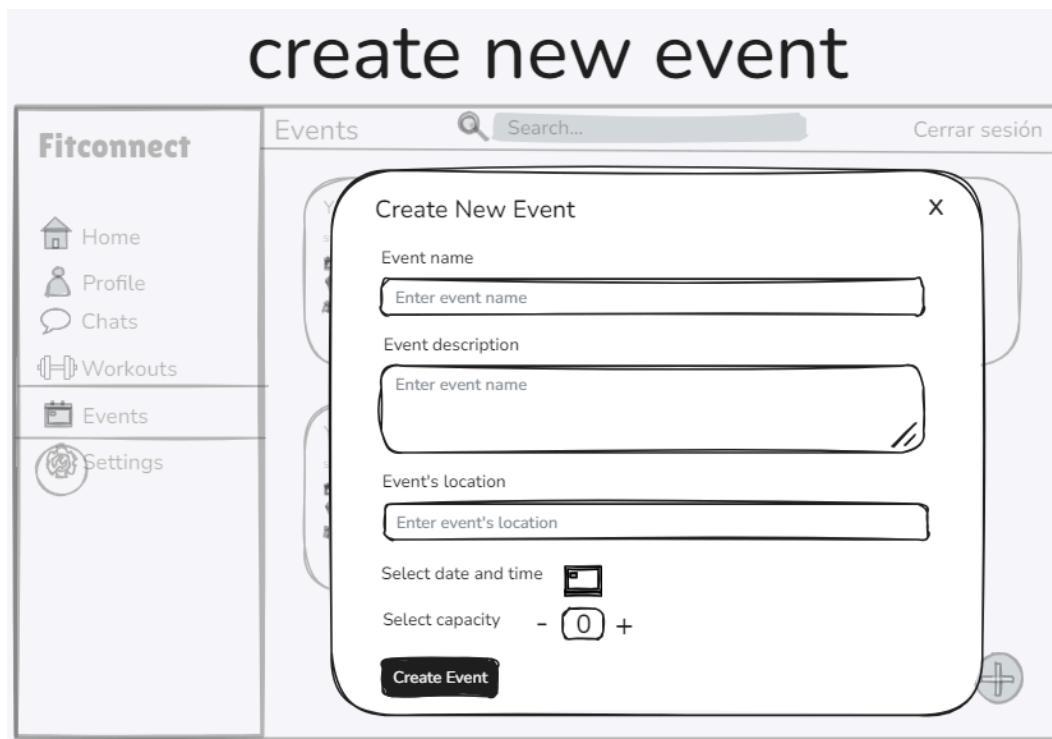


Figura 22: Boceto pantalla creación de nuevo evento

# settings

**Fitconnect**

- Home
- Profile
- Chats
- Workouts
- Events
- Settings

**Settings**

**Profile settings**

Change Avatar

Username:

Weight: 70 kg

Email:

Height: 178 cm

Bio:

**Save Changes**

**Security**

Current Password:

New Password:

Confirm New Password:

**Change Password**

**Notifications**

Email Notifications

Push Notifications

Marketing Emails

**Delete Account**

Once you delete your account, there is no going back. Please be certain.

**Delete Account**

Figura 23: Boceto pantalla ajustes

# manage users

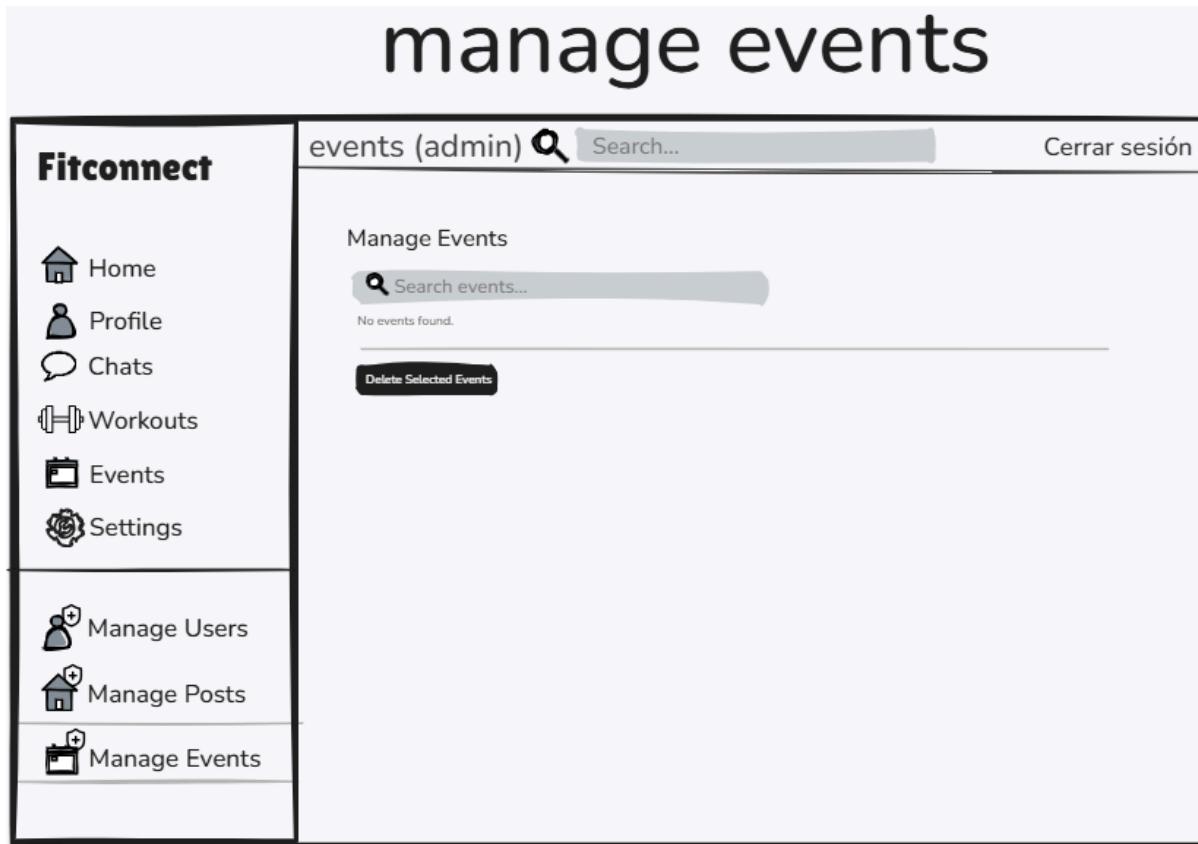
The screenshot shows the 'Manage Users' section of the Fitconnect application. The left sidebar contains links for Home, Profile, Chats, Workouts, Events, and Settings, along with three additional management links: Manage Users, Manage Posts, and Manage Events. The main content area is titled 'users (admin)' and features a search bar with placeholder text 'Search...'. Below the search bar, it says 'Manage Users' and includes a search bar with placeholder 'Search users...'. A message 'No users found.' is displayed. At the bottom of the content area are two buttons: 'Delete Selected Users' and 'Delete Selected Comments'.

[Figura 24: Boceto pantalla gestión de usuarios](#)

# manage posts

The screenshot shows the 'Manage Posts' section of the Fitconnect application. The left sidebar contains links for Home, Profile, Chats, Workouts, Events, and Settings, along with three additional management links: Manage Users, Manage Posts, and Manage Events. The main content area is titled 'posts (admin)' and features a search bar with placeholder text 'Search...'. Below the search bar, it says 'Manage Posts' and includes a search bar with placeholder 'Search posts...'. A message 'No posts found.' is displayed. At the bottom of the content area are two buttons: 'Delete Selected Posts' and 'Delete Selected Comments'.

[Figura 25: Boceto pantalla gestión de publicaciones y comentarios](#)



[Figura 26: Boceto pantalla gestión eventos](#)