Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

TrainStats: Aplicación multiplataforma de seguimiento de entrenamientos deportivos

---

Alumno: Pablo Gil Martín

Tutor: José Vicente Álvarez Bravo

TrainStats: Aplicación multiplataforma de seguimiento de entrenamientos deportivos

Pablo Gil Martín

Índice

Índice de figuras

Índice de tablas

Parte I: Fundamentación y contextualización

1. Introducción y conceptos básicos

La transformación digital ha llegado para quedarse en todos los aspectos de la vida moderna, y el ámbito del deporte y el bienestar no es una excepción. La cuantificación del rendimiento, el seguimiento de la progresión y la gestión de rutinas de entrenamiento se han convertido en prácticas habituales para deportistas de todos los niveles. Este paradigma ha generado una alta demanda de herramientas tecnológicas que faciliten estas tareas. El mercado actual ofrece una serie de soluciones para este fin, pero a menudo presentan barreras de entrada significativas: algunas requieren la compra de **hardware específico** (como relojes inteligentes o pulseras de actividad), otras imponen **modelos de suscripción** costosos, y una gran mayoría se desarrollan como **aplicaciones nativas**, lo que obliga al usuario a pasar por las tiendas de aplicaciones (App Store, Google Play) y limita su uso a un ecosistema de sistema operativo concreto.

Este Trabajo Fin de Grado nace como respuesta a estas limitaciones, planteando el **diseño y desarrollo de una aplicación multiplataforma de seguimiento de entrenamientos deportivos**. El objetivo fundamental es crear una solución tecnológica que sea **accesible, universal e intuitiva**, buscando democratizar el seguimiento deportivo al alcance de cualquier usuario, y para ello ofreciendo una herramienta potente y gratuita que funcione de manera consistente en cualquier dispositivo con un navegador web moderno, eliminando así la necesidad de instalar absolutamente nada – aunque se podrá hacer si así lo quiere el usuario – y reduciendo, por tanto, al máximo posible, las barreras de acceso a la plataforma.

La aplicación permitirá a los usuarios registrar de forma meticulosa sus sesiones de entrenamiento, detallando cada ejercicio, el peso levantado, las repeticiones y las series completadas. Este registro no será un mero almacén de datos, ya que el sistema proporcionará al usuario herramientas de visualización, como gráficos y paneles de visualización de datos históricos, que le permitirán analizar su evolución, identificar patrones de estancamiento o progreso y, en definitiva, tomar decisiones más informadas sobre su entrenamiento. La interfaz de usuario se diseñará con un enfoque en la simplicidad y la eficiencia, minimizando el tiempo que el usuario pasa interactuando con la aplicación para que pueda centrarse en su actividad física.

El alcance del proyecto se centrará en el desarrollo de las funcionalidades esenciales de seguimiento deportivo, dejando fuera, intencionadamente, características más complejas como la gestión de dietas, la integración con redes sociales o la prescripción de entrenamientos usando agentes de inteligencia artificial generativa, funcionalidades que, si bien sobre el papel suenan útiles en el contexto del proyecto, no han sido desarrolladas en esta primera versión. Esta delimitación permite concentrar los esfuerzos en construir un núcleo funcional robusto y una experiencia de usuario pulida, satisfaciendo así el objetivo principal del proyecto.

El Trabajo Fin de Grado no es, pues, solamente el desarrollo del software en sí, sino también todo lo que el Proceso de Desarrollo completo conlleva, comenzando por la planificación del mismo, así como la investigación en conceptos relativos a las tecnologías web modernas – como la arquitectura **Serverless,** el uso de Firebase y el paradigma de las **Aplicaciones Web Progresivas (PWA)** –, para posteriormente llevar a cabo la implementación de la plataforma anteriormente descrita y documentar todo el proceso realizado.

2. Metodología utilizada

Aunque las metodologías ágiles como Scrum o Kanban son el estándar actual en la industria para equipos de desarrollo, en este proyecto, al ser abordado por un único desarrollador, se ha optado por el **modelo en cascada**. Esta decisión estratégica se basa en la búsqueda de la **simplicidad y la efectividad**. Al no existir la necesidad de coordinar un equipo, las ceremonias y artefactos propios de Scrum (como los sprints, las reuniones diarias o las retrospectivas) representarían una complejidad innecesaria. El modelo en cascada, por el contrario, ofrece un marco de trabajo secuencial, predecible y rigurosamente documentado, donde cada fase se completa antes de iniciar la siguiente. Esto asegura una base sólida y bien definida desde el principio, convirtiéndolo en el enfoque más directo y eficiente para un proyecto de esta escala.

El tratamiento de **Casos de Uso** como vehículo para la especificación de requisitos permite una comprensión profunda y sin ambigüedades de las interacciones entre el usuario y el sistema, lo que resulta fundamental para garantizar que el producto final se alinee con los objetivos planteados.

Este enfoque disciplinado proporciona un marco de trabajo riguroso, ideal para un entorno académico donde la trazabilidad de los requisitos y la calidad de la documentación son tan importantes como el software resultante. El ciclo de vida se articulará en cuatro fases bien diferenciadas:

2.1. Fase I – Planificación

En esta primera fase del Proyecto es donde se establecen las bases para su gestión y control. Antes de profundizar en los requisitos técnicos, se define el marco de trabajo, los recursos y las metas. El objetivo es asegurar la viabilidad y crear una hoja de ruta clara. Algunas de las tareas que se realizan en esta fase son:

- **Definición de objetivos generales:** Se establecen las metas de alto nivel que el proyecto debe alcanzar para ser considerado un éxito.
- **Estimación temporal:** A partir de los objetivos previamente descritos, se extraen una serie de hitos clave, se planifica la duración de alcance de los mismos
    - La **Estructura de Desglose del Trabajo (EDT)** será el documento resultante de esta tarea en el que se descompone el alcance total del proyecto en paquetes de trabajo más pequeños y manejables. Esta estructura es fundamental para la asignación de tareas y el seguimiento del progreso basado en hitos.
    - **Estimación de Costes:** Se asigna un presupuesto para cubrir todos los costes previstos (horas de desarrollo, herramientas, etc.).

El entregable principal de esta fase en un proceso de Desarrollo real sería el Plan de Proyecto, un documento formal que sirve como guía maestra para todas las fases posteriores.

2.1. Fase II - Análisis de requisitos

Esta fase inicial se dedica a la definición completa y exhaustiva de todos los requisitos del sistema para establecer una visión clara y un alcance cerrado del proyecto. El objetivo es documentar qué debe hacer la aplicación sin entrar en detalles de implementación. Las tareas clave incluyen:

- **Identificación de Actores:** Se define el perfil del "Usuario Deportista" como el único y principal actor del sistema.
- **Especificación de Casos de Uso:** Se comienza diseñando el **Diagrama de Casos de Uso** que permita ver, desde una visión global, los distintos casos de uso que podrán realizar los distintos actores previamente definidos. Posteriormente se redactan y detallan todos los casos de uso, incluyendo flujos principales, alternativos y de excepción.

El entregable de esta fase en un proceso de desarrollo real sería la Especificación de Requisitos de Software (ERS), un documento formal que servirá como única fuente de requisitos inmutables para las fases venideras en el proceso de desarrollo.

2.2. Fase III – Diseño y arquitectura del sistema

Con los requisitos cerrados, esta fase se centra en modelar y diseñar la arquitectura del software a desarrollar, para ello, se realizan diversas tareas:

- **Modelado de Datos:** Se genera un diagrama conceptual que represente el esquema que deberán seguir los datos presentes en el sistema.
- **Diseño de la arquitectura:** Se define la estructura general de la aplicación así como la organización de los componentes que la conforman. Es en esta tarea donde se diseñan los diagramas de arquitectura – tanto a nivel lógico como a nivel físico.
- **Diseño de los flujos de información:** Mediante la confección de diagramas de secuencia se diseñan los distintos flujos en los que radicará la interacción del usuario con el software.

2.3. Fase IV – Implementación

En esta fase se produce la totalidad del código fuente del software a desarrollar. El principal objetivo de la fase, por tanto, es traducir el diseño arquitectónico que satisface los requisitos en código funcional, hasta obtener un producto completo y operativo. La única tarea de la fase, pues, será la mera codificación de los distintos componentes y vistas de la aplicación siguiendo el stack tecnológico planificado.

2.4. Fase V – Pruebas

Una vez finalizada la implementación, se procede a una fase dedicada exclusivamente a la verificación del sistema completo para asegurar que cumple con todos los requisitos y está libre de errores.

- **Pruebas de Integración:** Se ensamblan todos los componentes y se verifica que interactúan correctamente entre sí y se obtienen los resultados esperados.
- **Pruebas de Aceptación del Usuario (UAT):** Se valida formalmente que la aplicación final satisface las necesidades del usuario principal de la misma – el deportista que quiere llevar un seguimiento de sus entrenamientos.

2.5. Fase VI – Despliegue en entorno cloud y mantenimiento

Esta fase final se enfoca en hacer el sistema desarrollado accessible por cualquier usuario, asegurando su operatividad a largo plazo.

- **Despliegue en entorno cloud:** La aplicación se hace pública, desplegándose en un entorno cloud y por tanto siendo accessible por cualquier usuario con conexión a internet.
- **Elaboración de Documentación Final:** Se redactan los manuales de usuario y la documentación técnica de despliegue.
- **Mantenimiento:** En un proceso de desarrollo de un software real, comenzaría el ciclo de vida de mantenimiento del producto para la corrección de errores y la planificación de futuras mejoras.

3. Fundamento teórico del proyecto

La selección tecnológica es un pilar fundamental de este proyecto, orientada a construir una aplicación moderna, eficiente y escalable. La arquitectura y las herramientas han sido elegidas para maximizar la calidad de la experiencia de usuario y optimizar el ciclo de desarrollo. En este capítulo del Trabajo Fin de Grado se pretende describir, desde un nivel teórico, una serie de conceptos adquiridos tras la investigación acerca de tecnologías web modernas, con el fin de facilitar la comprensión de la arquitectura final utilizada.

3.1. Arquitectura Serverless y Backend as a Service (BaaS)

El proyecto adopta una **arquitectura Serverless**, un paradigma emergente de computación en la nube cuyo objetivo es abstraer al desarrollador de la gestión de la infraestructura del servidor. Como señala Mike Roberts, uno de los fundadores de Symphonia, en el artículo [1] de la bibliografía, el término Serverless no implica la ausencia de servidores, sino que las tareas que tradicionalmente se realizaban en el servidor son transparentes para el desarrollador. Esta abstracción permite que los equipos se centren exclusivamente en la lógica de negocio que aporta valor directo al proyecto final.

Dentro del ecosistema **Serverless**, este proyecto se apoya específicamente en un proveedor de **Backend as a Service (BaaS),** que es una implementación particular de este paradigma donde una plataforma de terceros ofrece un conjunto de servicios backend listos para ser consumidos a través de APIs. Para este caso, la plataforma seleccionada es **Firebase** de Google.[1]

Esta decisión estratégica ofrece ventajas cruciales:

- **Enfoque en el desarrollo de valor:** Al delegar por completo la gestión de la infraestructura (configuración de servidores, sistemas operativos de los mismos, parches de seguridad, balanceo de carga, mantenimiento de bases de datos), el esfuerzo de desarrollo se puede concentrar en la construcción de una interfaz de

[1] Documentación: https://firebase.google.com/docs?hl=es-419

usuario rica y una lógica de aplicación robusta, que son las partes del sistema con las que el usuario interactúa directamente.

- **Escalabilidad elástica y automática**: La infraestructura de un proveedor como Google escala de forma automática y transparente para soportar la demanda. El sistema puede gestionar desde un único usuario hasta millones de manera concurrente sin necesidad de intervención manual, provisionar más recursos o rediseñar la arquitectura. Se paga únicamente por el consumo real, optimizando los costes operativos, y para este Trabajo Fin de Grado la capa gratuita que ofrece Firebase ha resultado ser más que suficiente.
- **Reducción en el tiempo de desarrollo:** Un BaaS como Firebase provee servicios preconstruidos y listos para integrar, como la autenticación de usuarios, bases de datos en tiempo real o almacenamiento de archivos. Esto reduce drásticamente el tiempo de desarrollo, ya que evita la necesidad de "reinventar la rueda" para funcionalidades comunes a la mayoría de las aplicaciones modernas.

Como se ha comentado, Firebase será la plataforma que proveerá todos los servicios de backend, destacando los siguientes:

- **Firestore:** Es una base de datos NoSQL documental y flexible, diseñada para ofrecer escalabilidad global y sincronización de datos en tiempo real. Su modelo de datos (colecciones de documentos JSON) se adapta perfectamente a los cambios en los requisitos de la aplicación. Su capacidad de "escuchar" cambios en la base de datos y actualizar la UI instantáneamente es ideal para este proyecto, donde los datos deben estar siempre sincronizados entre dispositivos.
- **Firebase Authentication:** Proporciona un servicio completo de gestión de identidad (IdP), manejando de forma segura el ciclo de vida de la autenticación de usuarios (registro, inicio de sesión, recuperación de contraseña). Abstrae la complejidad de protocolos como OAuth 2.0 y permite una fácil integración con proveedores de identidad como Google o Email/Contraseña.
- **Security Rules:** Constituyen la capa de seguridad fundamental del backend. Es un sistema declarativo que permite definir reglas de acceso a los datos en Firestore de forma extremadamente granular. Con ellas, se puede garantizar que cada usuario solo pueda leer y escribir su propia información, implementando un modelo de

seguridad robusto directamente en la capa de datos sin escribir una sola línea de código de servidor.

Para emular una arquitectura clásica cliente-servidor y de paso, conseguir comunicar la aplicación con Firebase, se ha optado por **Netlify**[2] como proveedor de alojamiento para el frontend, de modo que la aplicación en sí se alojará en Netlify mientras que los datos y las reglas y restricciones de los mismos estarán en Firebase.

3.2. Aplicaciones Web Progresivas (PWAs)

Para cumplir el requisito de ser una aplicación multiplataforma de alto rendimiento, el proyecto se desarrollará como una **Aplicación Web Progresiva (PWA).** Las PWAs representan una evolución fundamental de las aplicaciones web, un paradigma impulsado por Google que busca fusionar las mejores cualidades de la web (accesibilidad, enlace directo) con la experiencia de usuario de las aplicaciones nativas (rendimiento, funcionalidad sin conexión, notificaciones push).

Según los desarrolladores de Google en el artículo [2] de la bibliografía, una aplicación web progresiva es una aplicación web que utiliza capacidades de navegadores modernos para ofrecer una experiencia de usuario equiparable a la de una aplicación nativa. Su implementación se basa en dos tecnologías clave:

- **Web App Manifest:** Es un archivo de configuración en formato JSON que proporciona metadatos sobre la aplicación. Este manifiesto le indica al sistema operativo del dispositivo cómo debe comportarse la web una vez "instalada": el nombre de la app, los iconos para la pantalla de inicio, el color del tema, la orientación de la pantalla, etc.
- **Service Workers:** Son el pilar técnico de una PWA. Un Service Worker es un script que el navegador ejecuta en segundo plano, separado de la página web principal. Actúa como un middleware entre la aplicación y la red, interceptando todas las peticiones de red. Esto permite implementar estrategias de caché avanzadas, habilitar la funcionalidad sin conexión (vital en un entorno como un gimnasio con cobertura deficiente) y gestionar notificaciones push.

[2] https://docs.netlify.com/

Las características clave que justifican la elección de este paradigma para el desarrollo que se pretende realizar son:

- **Instalable de forma opcional:** Si así lo desea, el usuario puede añadir la aplicación a la pantalla de inicio de su dispositivo, logrando un acceso directo y una presencia constante similar a una app nativa, pero sin la fricción de tener que buscarla y descargarla desde una tienda de aplicaciones.
- **Rendimiento nativo:** Gracias al control granular que ofrecen los Service Workers sobre el almacenamiento en caché, los recursos de la aplicación (código, estilos, imágenes) se guardan en el dispositivo tras la primera visita. Esto resulta en tiempos de carga casi instantáneos en visitas posteriores.
- **Segura por defecto:** En sistemas operativos móviles modernos, es un requisito indispensable que las PWAs se sirvan a través de HTTPS. Esto garantiza que toda la comunicación entre el usuario y el servidor esté cifrada, protegiendo la integridad y confidencialidad de los datos.

3.3. El paradigma de la arquitectura por componentes

La interfaz de usuario se construirá con React, una biblioteca de JavaScript desarrollada por Meta. Su popularidad se debe a dos conceptos fundamentales: su paradigma declarativo y su arquitectura basada en componentes. La filosofía de React consiste en descomponer una interfaz de usuario compleja en piezas más pequeñas, aisladas y reutilizables llamadas **componentes**. Cada componente gestiona su propio estado y lógica. De forma análoga a bloques de construcción, estos componentes se pueden ensamblar para construir vistas complejas, fomentando un desarrollo modular, escalable y mantenible.

Dos de los conceptos más esenciales en React son los **componentes funcionales** y el lenguaje **JSX.**

3.3.1. Componentes funcionales

En React moderno, un componente se escribe típicamente como una **función** de JavaScript. Este enfoque se alinea con el paradigma de la programación funcional y se puede expresar con una simple analogía matemática, donde como en toda función, una serie de inputs son transformados en un output. En este caso, los inputs serían los **Props** que recibe el componente, siendo el propio componente web en lenguaje JSX su salida.

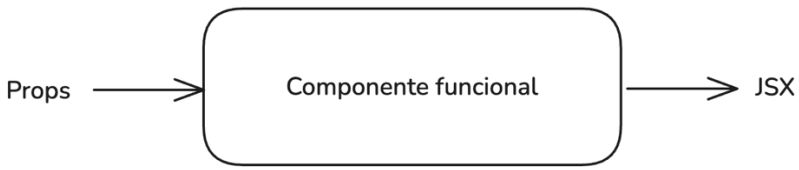

Figura 1 - Esquema representativo de un componente functional

3.3.2. El lenguaje JSX

JSX (JavaScript XML) es una extensión de la sintaxis de JavaScript que permite escribir una estructura similar a HTML directamente en el código. No es HTML ni es entendido de forma nativa por los navegadores, siendo preciso por tanto el uso de una herramienta similar a un **compilador,** que "traduce" este lenguaje en llamadas de función de JavaScript, siendo esta versión traducida la que posteriormente es renderizada por el servidor web.

4. Planificación y estimación

3.4. *Definición de objetivos*

A continuación se definen, desde un alto nivel, los **objetivos principales** del software a desarrollar:

- **Digitalizar** el registro de cada sesión de entrenamiento del usuario, siendo estas persistentes y por tanto permitiendo al usuario consultar posteriormente datos históricos.
- Estructurar los datos por **sesiones de entrenamiento** y **rutinas,** siendo la principal diferencia entre ambas el propósito de las mismas:
    - Una sesión de entrenamiento viene a ser la entidad que modela un entrenamiento realizado en una fecha determinada
    - Una rutina, en cambio, es una colección de ejercicios parametrizados (con parámetros como series y repeticiones) que podrá ser compartida o usada como plantilla para realizar sesiones de entrenamiento
- **Centralizar** todo el historial de actividad en un único lugar, reemplazando por completo el seguimiento manual en papel.
- **Visualizar** el progreso personal con vistas claras e intuitivas.
- Ofrecer una **experiencia de uso idéntica** en móvil, tableta y escritorio.
- Permitir la **instalación opcional** en la pantalla de inicio del dispositivo.
- Garantizar la total **privacidad y seguridad** de los datos del usuario.
- Proporcionar una **interfaz de usuario** rápida, limpia y fácil de usar.

3.5. *Estimación y planificación temporal*

La planificación del proyecto se ha realizado utilizando un enfoque de **Estructura de Desglose del Trabajo (EDT)**, dividiendo el esfuerzo total en las fases definidas por la metodología y, dentro de estas, en paquetes de trabajo y tareas más pequeñas y manejables. A continuación, se presenta una tabla que resume las fases, los hitos principales y una estimación temporal. La duración total estimada del proyecto es de **12 semanas**.

Fase	Hitos	Estimación
I - Planificación y estimación	• Definición del alcance del proyecto en base a una serie de objetivos. • Estimación temporal mediante la creación de la Estructura de Desglose del Trabajo (EDT). • Estimación de costes del proyecto.	1 semana
II - Análisis de requisitos	• Definición de actores que interactuarán con el sistema • Análisis exhaustivo de las necesidades del usuario y traducción de las mismas en requisitos funcionales y no funcionales. • Redacción y validación de la Especificación de Requisitos de Software (ERS).	2 semanas
III - Diseño y arquitectura	• Diseño de la arquitectura de la información mediante el modelado lógico de datos para definir el esquema que organizará la información en Firebase. • Definición de la arquitectura lógica simplificada • Definición de la arquitectura lógica detallada • Definición de la arquitectura física del sistema • Diseño de la experiencia de usuario (UX) y la interfaz gráfica (UI)	1 semana
IV - Implementación	• Configuración inicial del proyecto (React, PWA, Firebase). • Desarrollo del sistema de autenticación y gestión de rutas. • Implementación de todas las vistas CRUD para entrenamientos y ejercicios. • Codificación del Dashboard principal y el resto de vistas.	7 semanas
V - Pruebas	• Preparación y ejecución del plan de pruebas • Realización de las Pruebas de Aceptación de Usuario (UAT) para la validación final del proyecto.	2 semanas
VI - Despliegue y Mantenimiento	• Puesta en producción de la aplicación en el entorno cloud (Firebase y Netlify) • Configuración y monitorización del entorno final • Redacción y entrega de la documentación final (manual de usuario, documentación técnica	1 semana
TOTAL		14 semanas

Tabla 1 - Estructura de Desglose del Trabajo

3.6. Estimación de costes y presupuesto

El propósito de esta sección es analizar la viabilidad económica del proyecto y proporcionar una estimación detallada de los costes asociados a su desarrollo, desde los recursos humanos hasta la infraestructura tecnológica. Aunque este Trabajo Fin de Grado se desarrolla en un marco académico donde muchos costes son asumidos o inexistentes (como el salario del desarrollador), es un ejercicio fundamental realizar una valoración a precios de mercado para comprender el valor real del producto desarrollado.

Concepto	Descripción	Coste Estimado	Concepto
Recursos Humanos	240 horas de un desarrollador junior a 20 €/hora.	4.800,00€	Recursos Humanos
Amortización Hardware	Amortización de un portátil de 1.200 € durante 3 meses.	75,00€	Amortización Hardware
Software y Licencias	Uso de herramientas open source y gratuitas.	0,00€	Software y Licencias
Servicios en la Nube	Uso del plan gratuito "Spark" de Firebase.	0,00€	Servicios en la Nube
Costes Indirectos	Electricidad e Internet.	60,00€	Costes Indirectos
TOTAL		4.935,00€	TOTAL

Tabla 2 - Presupuesto estimado del proyecto

El **presupuesto total** estimado para el desarrollo del proyecto, valorado a precios de mercado, es de **4.935,00 €.** Es importante destacar que el coste real de desembolso para mí ha sido prácticamente nulo, ya que el coste principal (recursos humanos) es el propio trabajo del autor, y tanto el software como la infraestructura en la nube se obtienen de forma gratuita. Este análisis confirma la alta viabilidad económica del proyecto y, sobre todo, sirve para cuantificar el valor de mercado del producto final, demostrando que el conocimiento y el tiempo invertidos se traducen en un activo tecnológico tangible y valorable.

Parte II: Ejecución

5. Análisis

La fase de análisis es fundamental en una metodología clásica, ya que su propósito es definir de manera precisa e inequívoca **qué** debe hacer el sistema, antes de tomar decisiones sobre **cómo** lo hará. En esta etapa se identifican los requisitos funcionales y no funcionales, estableciendo un contrato claro que guiará las fases posteriores de diseño y desarrollo.

5.1. Identificación de Actores

El sistema presenta dos actores principales que interactúan con él:

- **Usuario No Identificado**: Representa a cualquier persona que visita la aplicación web pero no ha iniciado sesión. Su interacción está limitada a las funcionalidades públicas, como conocer la aplicación, registrarse para crear una nueva cuenta o iniciar sesión si ya tiene una.
- **Usuario Deportista**: Este actor representa a un usuario que se ha autenticado exitosamente en el sistema. Hereda las capacidades del usuario no identificado (aunque no las use una vez logueado) y además tiene acceso a todas las funcionalidades privadas de la aplicación, como la gestión de sus entrenamientos y la visualización de su progreso.

5.2. Requisitos Funcionales (Casos de Uso)

Los requisitos funcionales se capturan mediante Casos de Uso, que describen las interacciones entre el actor y el sistema para alcanzar un objetivo concreto. A continuación se adjunta, siguiendo estándares, un Diagrama de Casos de Uso desde el que se puede ver una visión global de todos los CU, y posteriormente, se especifican dichos CU:

5.2.1. Diagrama de Casos de Uso

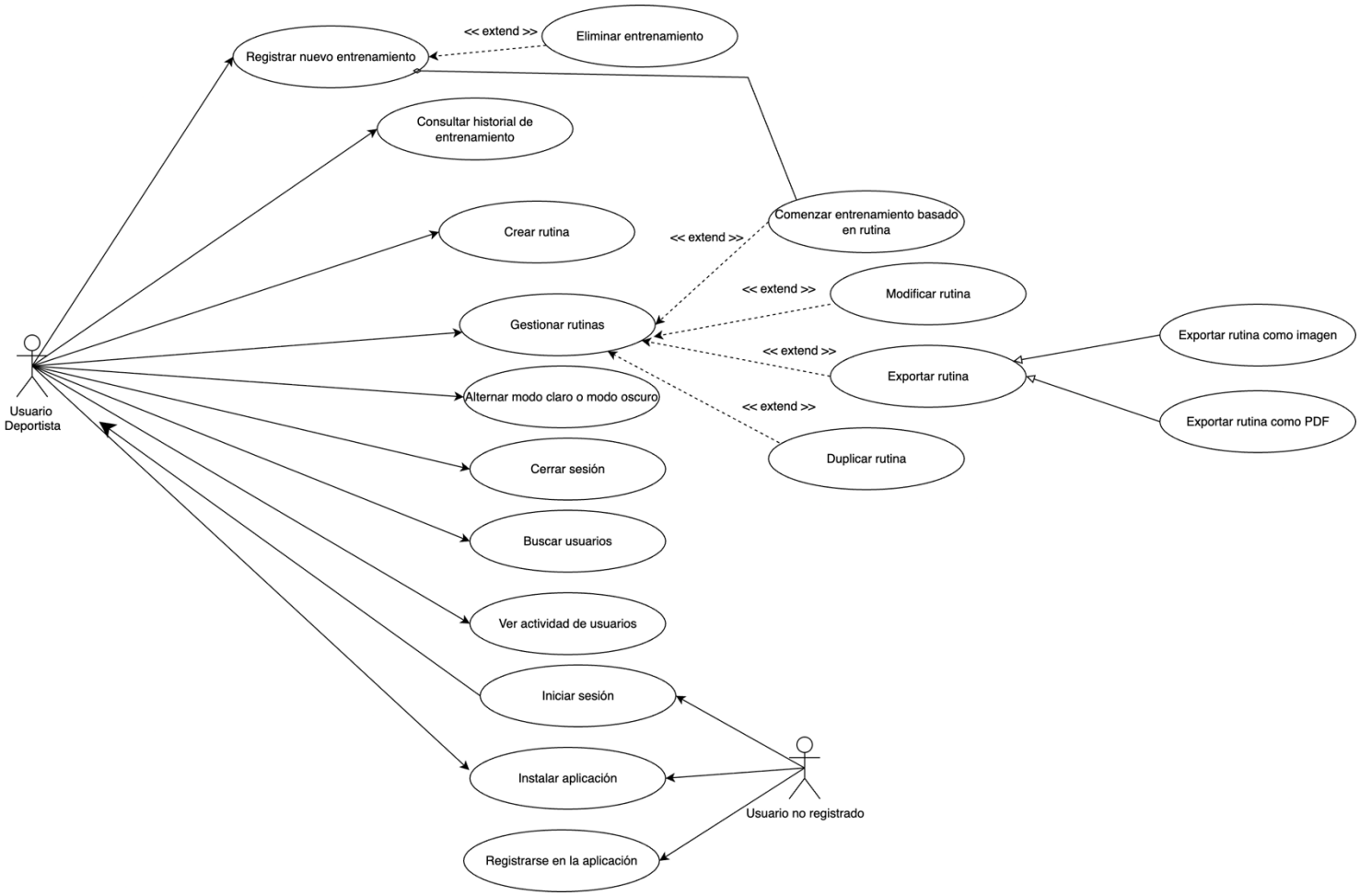

Figura 2 - Diagrama de Casos de Uso

5.2.2. Especificación de Casos de Uso

CU-01	**Registrarse en la aplicación**
Actor Principal	Usuario No Identificado
Descripción	Permite a un nuevo usuario crear una cuenta personal en la aplicación para poder acceder a sus funcionalidades
Precondiciones	El usuario no debe tener una sesión activa en el sistema
Postcondiciones	Se crea una nueva cuenta de usuario en el sistema. El usuario es automáticamente autenticado y redirigido al panel principal de la aplicación
Flujo Principal	1. El usuario selecciona la opción "Registrarse" 2. El sistema muestra un formulario solicitando un nombre de usuario, una dirección de correo electrónico y una contraseña 3. El usuario completa los campos y envía el formulario 4. El sistema valida que los datos son correctos y que el correo electrónico no está ya en uso 5. El sistema crea la nueva cuenta de usuario 6. El sistema inicia sesión automáticamente para el nuevo usuario y le muestra el panel principal
Flujos Alternativos	4a. **Datos inválidos**: Si algún campo no cumple con el formato requerido (ej. email inválido, contraseña insegura), el sistema muestra un mensaje de error y no crea la cuenta. 4b. **Email ya registrado**: Si el correo electrónico ya existe en la base de datos, el sistema informa al usuario y le sugiere iniciar sesión o usar otro correo.

Tabla 3 - CU-01: Registrarse en la aplicación

CU-02	**Iniciar Sesión**
Actor Principal	Usuario No Identificado
Descripción	Permite a un usuario con una cuenta existente acceder al sistema
Precondiciones	El usuario debe tener una cuenta registrada y no tener una sesión activa
Postcondiciones	El sistema valida las credenciales del usuario, establece una sesión activa y le concede acceso a las funcionalidades privadas de la aplicación
Flujo Principal	1. El usuario selecciona la opción "Iniciar Sesión" 2. El sistema presenta un formulario solicitando correo electrónico y contraseña 3. El usuario introduce sus credenciales y envía el formulario 4. El sistema verifica que las credenciales son correctas 5. El sistema crea una sesión para el usuario y lo redirige al panel principal
Flujos Alternativos	4a. **Credenciales incorrectas**: Si el correo o la contraseña no coinciden con los registros, el sistema muestra un mensaje de error y no permite el acceso

Tabla 4 - CU-02: Iniciar Sesión

CU-03	**Instalar aplicación**
Actor Principal	Usuario No Identificado / Usuario Deportista
Descripción	Permite al usuario instalar la aplicación web en la pantalla de inicio de su dispositivo para un acceso rápido y una experiencia similar a una app nativa
Precondiciones	El usuario está accediendo a la aplicación desde un navegador compatible con PWA
Postcondiciones	La aplicación aparece como un icono en la pantalla de inicio del dispositivo
Flujo Principal	1. El navegador detecta que el sitio es una PWA y muestra un indicador o un aviso de instalación 2. El usuario hace clic en el botón "Instalar" 3. El sistema operativo instala la aplicación, añadiendo un icono a la pantalla de inicio 4. El usuario puede ahora lanzar la aplicación directamente desde su icono

Tabla 5 - CU-03: Instalar aplicación

CU-04	**Cerrar sesión**
Actor Principal	Usuario Deportista
Descripción	Permite a un usuario autenticado finalizar su sesión de forma segura
Precondiciones	El usuario debe tener una sesión activa en la aplicación
Postcondiciones	La sesión del usuario se invalida y el sistema lo redirige a la página de inicio pública
Flujo Principal	1. El usuario selecciona la opción "Cerrar Sesión". 2. El sistema finaliza la sesión activa del usuario. 3. El sistema redirige al usuario a la página de inicio para usuarios no identificados

Tabla 6 - CU-04: Cerrar Sesión

CU-05	**Registrar nuevo entrenamiento**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario registrar los detalles de una sesión de entrenamiento realizada, ya sea siguiendo una rutina o de forma libre
Precondiciones	El usuario debe haber iniciado sesión
Postcondiciones	Se guarda un nuevo registro de entrenamiento en el historial del usuario.
Flujo Principal	1. El usuario selecciona "Registrar Entrenamiento" o elige una rutina para iniciarla 2. El sistema presenta la interfaz de registro 3. Para cada ejercicio, el usuario introduce los datos reales (peso, repeticiones, etc.) 4. El usuario marca el entrenamiento como finalizado 5. El sistema guarda la sesión en el historial 6. El sistema muestra un resumen del entrenamiento
Flujos Alternativos	4a. **Salir sin guardar**: Si el usuario abandona la sesión antes de finalizar, el sistema pregunta si desea guardar el progreso parcial o descartarlo.

Tabla 7 - CU-05: Registrar Nuevo Entrenamiento

CU-06	**Crear rutina**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario diseñar una nueva rutina de entrenamiento, añadiendo ejercicios y especificando series, repeticiones, etc
Precondiciones	El usuario debe haber iniciado sesión
Postcondiciones	Se guarda una nueva rutina en el perfil del usuario
Flujo Principal	1. El usuario accede a la sección "Mis Rutinas" 2. El usuario selecciona la opción "Crear Nueva Rutina" 3. El sistema presenta una interfaz para nombrar la rutina y añadir ejercicios 4. El usuario busca y selecciona ejercicios 5. Para cada ejercicio, el usuario define detalles como series, repeticiones, peso y tiempo de descanso 6. El usuario guarda la rutina 7. El sistema confirma que la rutina ha sido guardada
Flujos Alternativos	6a. **Rutina sin ejercicios**: Si el usuario intenta guardar una rutina vacía, el sistema muestra un mensaje de error solicitando que añada al menos un ejercicio.

Tabla 8 - CU-06: Crear rutina

CU-07	**Gestionar rutinas**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario visualizar, modificar y eliminar sus rutinas de entrenamiento personalizadas
Precondiciones	El usuario ha iniciado sesión en la aplicación y tiene al menos una rutina creada para poder gestionarla
Postcondiciones	La lista de rutinas del usuario se actualiza para reflejar las acciones realizadas (modificación, duplicación o eliminación) o bien el sistema devuelve el archivo con la rutina (en caso de exportación)
Flujo Principal	1. El usuario accede a la sección "Rutinas” 2. El sistema muestra la lista de rutinas guardadas por el usuario
Sub-flujo: Comenzar entrenamiento basado en rutina	3a. El usuario selecciona una rutina existente y elige la opción "Comenzar" 4a. El sistema ejecuta los pasos del **CU-05: Registrar nuevo entrenamiento** tomando como lista de ejercicios los ejercicios que componen a la rutina
Sub-flujo: Modificar rutina (CU-07.1 en esta especificación)	3b. El usuario selecciona una rutina existente y elige la opción "Modificar” 4b. El sistema ejecuta los pasos del **CU-07.2: Modificar rutina**
Sub-flujo: Eliminar rutina	3c. El usuario selecciona una rutina existente y elige la opción "Eliminar" 4c. El sistema solicita confirmación para el borrado 5c. El usuario confirma la eliminación 6c. El sistema elimina la rutina de la base de datos y actualiza la lista 7c. El sistema muestra un mensaje de confirmación
Sub-flujo: Duplicar rutina	3d. El usuario selecciona una rutina existente y elige la opción "Duplicar" 4d. El sistema crea una copia de la rutina seleccionada, guardándola en el perfil del usuario.
Sub-flujo: Exportar rutina (CU-07.2 en esta especificación)	3b. El usuario selecciona una rutina existente y elige la opción "Exportar" 4b. El sistema ejecuta los pasos del **CU-07.2: Exportar Rutina**

Flujos Alternativos	2a. **Sin rutinas creadas**: Si el usuario no tiene ninguna rutina, el sistema muestra un mensaje invitándole a crear su primera rutina 5c. **Cancelar eliminación**: Si el usuario cancela la acción de borrado, el sistema vuelve a la lista de rutinas sin realizar cambios.

Tabla 9 - CU-07: Gestionar rutinas

CU-07.1	**Modificar rutina**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario editar una rutina de entrenamiento previamente creada
Precondiciones	Se ha realizado el CU-07, llegando a la pantalla de gestión de rutinas, seleccionando una rutina para su modificación. Las precondiciones de este Caso de Uso aplican también aquí por extensión: el usuario ha iniciado sesión y tiene al menos una rutina creada, por tanto, CU-07. Gestionar rutinas
Postcondiciones	La rutina seleccionada se actualiza con los nuevos cambios
Flujo Principal	1. El usuario navega a "Rutinas” 2. El usuario selecciona la rutina que desea editar 3. El usuario elige la opción "Modificar" 4. El sistema muestra la rutina en modo editable 5. El usuario añade, elimina o modifica los ejercicios y sus detalles 6. El usuario guarda los cambios 7. El sistema confirma que la rutina ha sido actualizada.
Flujos Alternativos	6a. **Descartar cambios**: El usuario puede salir sin guardar. El sistema pedirá confirmación antes de descartar los cambios.

Tabla 10 - CU-07.1: Modificar rutina

CU-07.2	**Exportar rutinas**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario exportar una rutina para descargarla, en formato de imagen o documento PDF
Precondiciones	Se ha realizado el CU-07, llegando a la pantalla de gestión de rutinas, seleccionando una rutina para su exportación a imagen o PDF. Las precondiciones de este Caso de Uso aplican también aquí por extensión: el usuario ha iniciado sesión y tiene al menos una rutina creada, por tanto, CU-07. Gestionar rutinas
Postcondiciones	El sistema devuelve al usuario el archivo en el formato seleccionado con los datos de la rutina.
Flujo Principal	1. El usuario navega a "Mis Rutinas” 2. El usuario selecciona la rutina que desea exportar 3. El sistema le muestra las dos opciones de exportación que soporta (imagen y PDF) 4. El usuario selecciona una de ellas 5. El sistema genera un documento según el formato especificado y se descarga

Tabla 11 - CU-08: Exportar rutinas

CU-08	**Consultar historial de entrenamiento**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario revisar un registro detallado y cronológico de todas las sesiones de entrenamiento que ha completado.
Precondiciones	El usuario ha iniciado sesión
Postcondiciones	Ninguna (es una operación de solo lectura)
Flujo Principal	1. El usuario accede a la sección "Historial de Entrenamiento" 2. El sistema recupera y muestra una lista de todos los entrenamientos registrados por el usuario, generalmente ordenados del más reciente al más antiguo 3. Cada elemento de la lista muestra información clave como la fecha y el nombre de la rutina o un resumen 4. El usuario puede seleccionar un entrenamiento específico de la lista 5. El sistema muestra todos los detalles de la sesión seleccionada, incluyendo cada ejercicio, las series, repeticiones y pesos registrados
Flujos Alternativos	2a. **Historial vacío**: Si el usuario no ha registrado ningún entrenamiento, el sistema muestra un mensaje indicándolo

Tabla 12 - CU-08: Consultar Historial de Entrenamientos

CU-09	**Alternar modo claro/oscuro**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario cambiar el tema visual de la interfaz de la aplicación entre un modo claro y un modo oscuro
Precondiciones	El usuario ha iniciado sesión
Postcondiciones	La interfaz de la aplicación cambia inmediatamente al tema seleccionado. La preferencia se guarda para futuras sesiones del usuario
Flujo Principal	1. El usuario localiza y activa el botón de la interfaz para cambiar de tema 2. El sistema cambia el esquema de colores de la interfaz al modo alternativo (de claro a oscuro, o viceversa) 3. El sistema guarda esta preferencia en su perfil

Tabla 13 - CU-09: Alternar modo claro/oscuro

CU-10	**Buscar usuarios**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario buscar otros usuarios de la aplicación y seguirlos, con la finalidad de poder visualizar sus rutinas y entrenamientos
Precondiciones	El usuario ha iniciado sesión
Postcondiciones	El sistema marca al usuario buscado como usuario seguido en el perfil del usuario actual
Flujo Principal	1. El usuario accede a la sección "Amigos" 2. El sistema muestra una pantalla de búsqueda de usuarios 3. El usuario introduce el nombre de usuario del usuario que quiere añadir 4. Si existe en la aplicación, el sistema se lo mostrará, junto con un botón que lo permita seguir 5. Al pulsar el botón de seguir, el sistema almacenará al usuario buscado en la lista de seguidos del usuario actual
Flujos Alternativos	4a. **Usuario no encontrado**: Si el nombre de usuario introducido no se corresponde con ninguno existente en la aplicación, el sistema

	mostrará un mensaje indicando que no se han encontrado usuarios

Tabla 14 - CU-10: Buscar usuarios

CU-11	**Ver actividad de usuarios**
Actor Principal	Usuario Deportista
Descripción	Permite al usuario ver las rutinas y entrenamientos realizados por otros usuarios a los que esté siguiendo (ver **CU-12: Buscar usuarios**)
Precondiciones	El usuario ha iniciado sesión, el usuario sigue a al menos un usuario
Postcondiciones	Ninguna
Flujo Principal	1. El usuario accede a la sección "Feed" 2. El sistema muestra, en orden cronológico, las rutinas creadas por los usuarios seguidos, así como los entrenamientos registrados por estos
Flujos Alternativos	2a. F**eed vacío**: Si el usuario no sigue a ningún otro usuario, o si los usuarios a los que sigue aún no han registrado ninguna rutina o entrenamiento, el sistema mostrará un mensaje indicando que aún no se ha publicado nada

Tabla 15 - CU-11: Ver actividad de usuarios

5.3. Requisitos No Funcionales

Son los atributos de calidad que debe poseer el sistema son los siguientes:

- **Rendimiento**: La aplicación debe cargar en menos de 3 segundos en una conexión 3G. Las interacciones de la UI (añadir series, cambiar de vista) deben ser instantáneas y no suponer un problema a la usabilidad de la plataforma.
- **Usabilidad**: La interfaz debe ser intuitiva y autoexplicativa. Procesos como registrar un entrenamiento o gestionar rutinas deben ser rápidos y no tener una curva de aprendizaje elevada, de modo que el usuario pueda interactuar con la plataforma sin necesidad de tener que aprender cómo. Asimismo, el diseño será responsive, adaptándose a pantallas de móvil, tableta y escritorio, dada la naturaleza multiplataforma del proyecto.
- **Fiabilidad**: La aplicación debe estar disponible el 99.9% del tiempo, tener resiliencia y tolerancia a fallos. La funcionalidad offline (PWA) asegura que el usuario pueda consultar sus datos y registrar nuevos entrenamientos incluso sin conexión. (¿?)
- **Seguridad**: Toda la comunicación entre el cliente y el servidor debe estar cifrada mediante HTTPS. Mecanismos como la doble validación (validación en la app y el uso de reglas de seguridad de Firestore) garantizarán que un usuario solo pueda acceder a sus propios datos.
- **Compatibilidad**: La aplicación deberá funcionar correctamente en las dos últimas versiones de los principales navegadores web: Google Chrome, Mozilla Firefox, Safari y Microsoft Edge.

6. Diseño y arquitectura

En esta fase se traduce el "qué" del análisis al "cómo" se construirá el sistema. Se define la estructura, los componentes y las tecnologías que darán vida a los requisitos.

3.7. Modelo Lógico de Datos

3.7.1. Diagrama de Documentos

A continuación, se adjunta un **diagrama de documentos** que pretende representar el **modelo lógico de datos** que seguirá la información sobre la que se sustenta la aplicación. Al ser Firestore una base de datos no relacional (NoSQL), **no es posible representarlo con un diagrama Entidad/Relación** como suele ser habitual, ya que, en este paradigma, no existen las relaciones entre entidades.

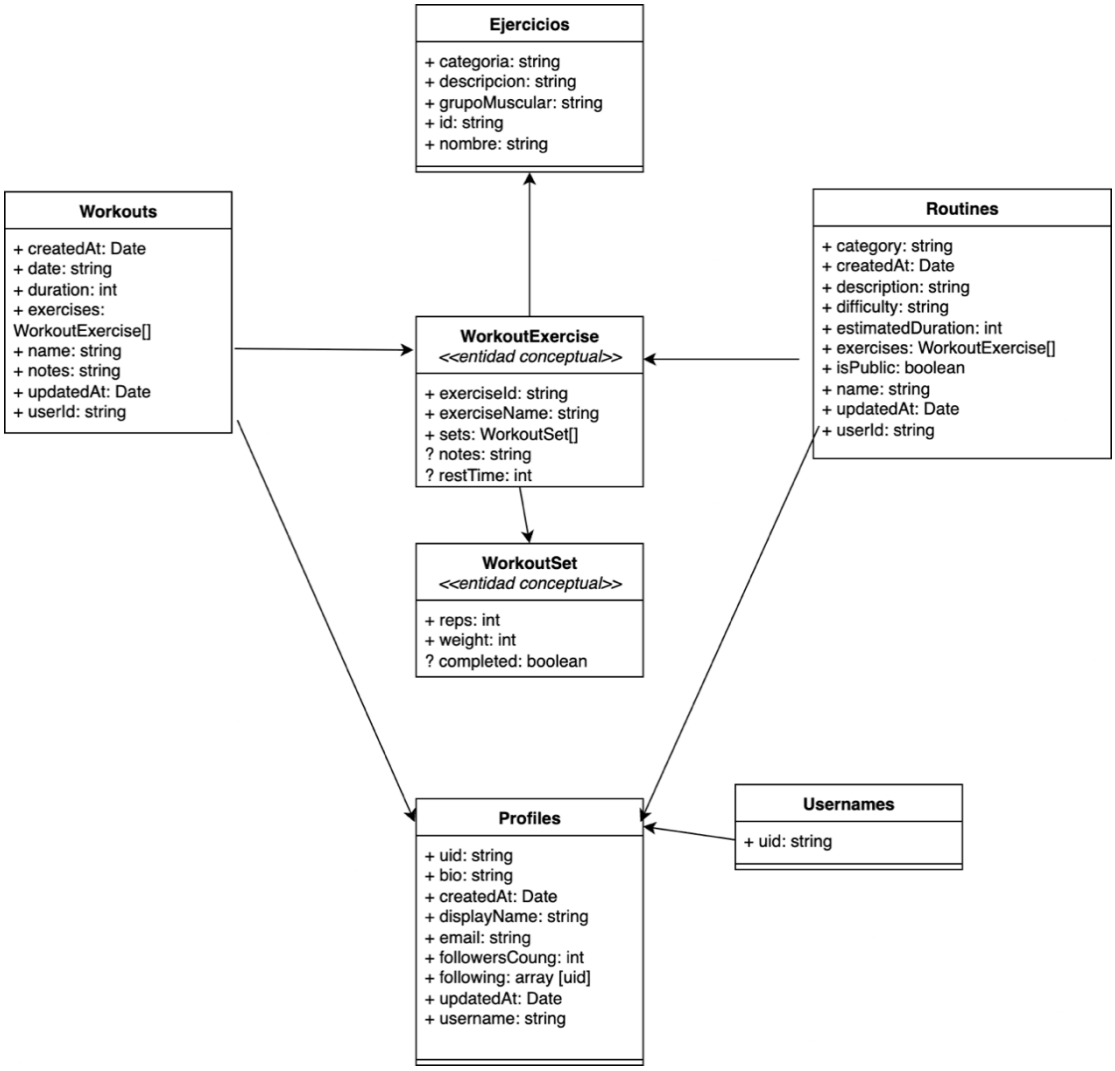

Figura 3 - Modelo Lógico de Datos

3.7.2. Especificación del esquema

Se aportan más detalles sobre los esquemas del diagrama previamente adjuntado:

- **Ejercicios** es una colección de Firestore donde cada documento representa un tipo de ejercicio que el usuario podrá añadir a sus entrenamientos y rutinas.
    - Un ejercicio es identificado unívocamente por un **id** de tipo string.
- **Workouts y Routines**, son dos colecciones de Firestore donde cada documento representa una sesión de entrenamiento o una rutina, respectivamente.
    - Tanto las sesiones de entrenamiento como las rutinas están identificadas unívocamente por un id de tipo string.
    - Tanto las sesiones de entrenamiento como las rutinas tienen un **userId,** que no es más que una referencia al id del usuario creador de dichas sesiones o rutinas.
    - Si una rutina es **pública** (isPublic) aparecerá en el buscador de rutinas de otros usuarios, pero no podrá ser modificada o eliminada por ellos.
    - Ambas colecciones hacen uso de las dos entidades conceptuales descritas en el diagrama, **WorkoutExercise y WorkoutSet,** que **no son colecciones en Firestore,** sino entidades lógicas virtuales para modelar cada ejercicio y repetición, respectivamente. Dentro de las colecciones, están representados como un objeto JSON (una especie de "sub-documento" dentro del documento)
- **Profiles** es la colección de Firestore donde se almacenan los perfiles de los usuarios
- **Usernames** es la colección de Firestore que almacena nombres de usuario y los relaciona con los perfiles mediante una referencia al id de un perfil.
    - Se optó por este enfoque ya que los nombres de usuario deben ser únicos y un usuario, al registrarse, introduce un nombre de usuario único como último paso.

3.8. Arquitectura Lógica Simplificada

Se adjunta el diagrama que representa, desde un alto nivel, la **arquitectura lógica** de la aplicación:

Figura 4 - Arquitectura lógica simplificada

3.9. Arquitectura Lógica Detallada

A continuación se adjunta el diagrama que modela la arquitectura lógica de la aplicación desarrollada desde un nivel de abstracción menor:

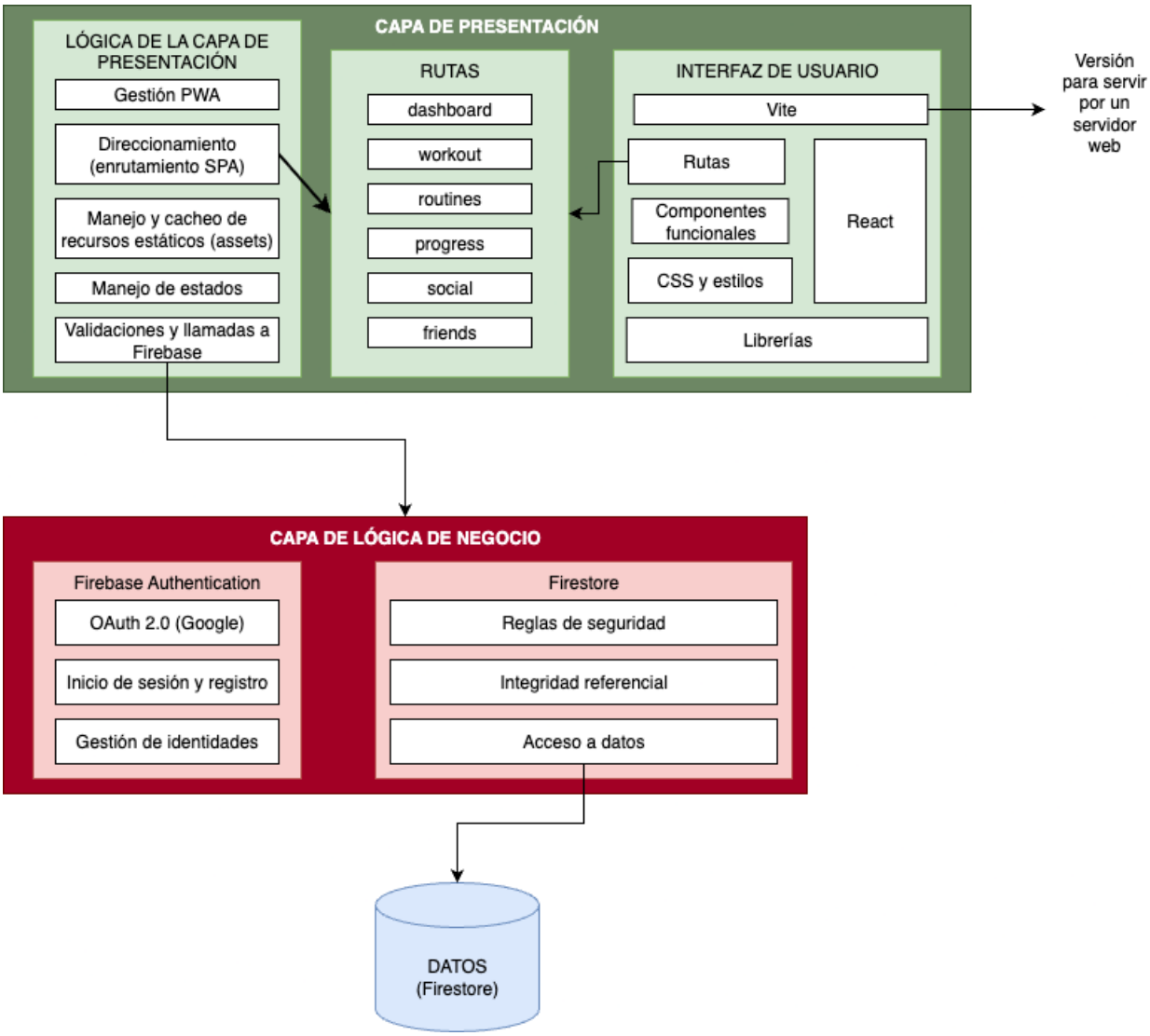

Figura 5 - Arquitectura lógica detallada

Como podemos ver en los diagramas anteriormente adjuntados, a nivel lógico la **aplicación se compone de tres capas esenciales,** siguiendo el modelo arquitectónico tradicional de 3 niveles: **Capa de Presentación** (vistas e interfaz de usuario), **Capa de Lógica de Negocio** (autenticación, manejo de identidades, reglas de acceso a datos e integridad…) y **Capa de Datos** que es donde se alojan los propios datos en sí.

3.10. Arquitectura física

Se adjunta el **diagrama de arquitectura física,** que pretende modelar dónde se ubica físicamente cada componente esencial para el funcionamiento de la aplicación:

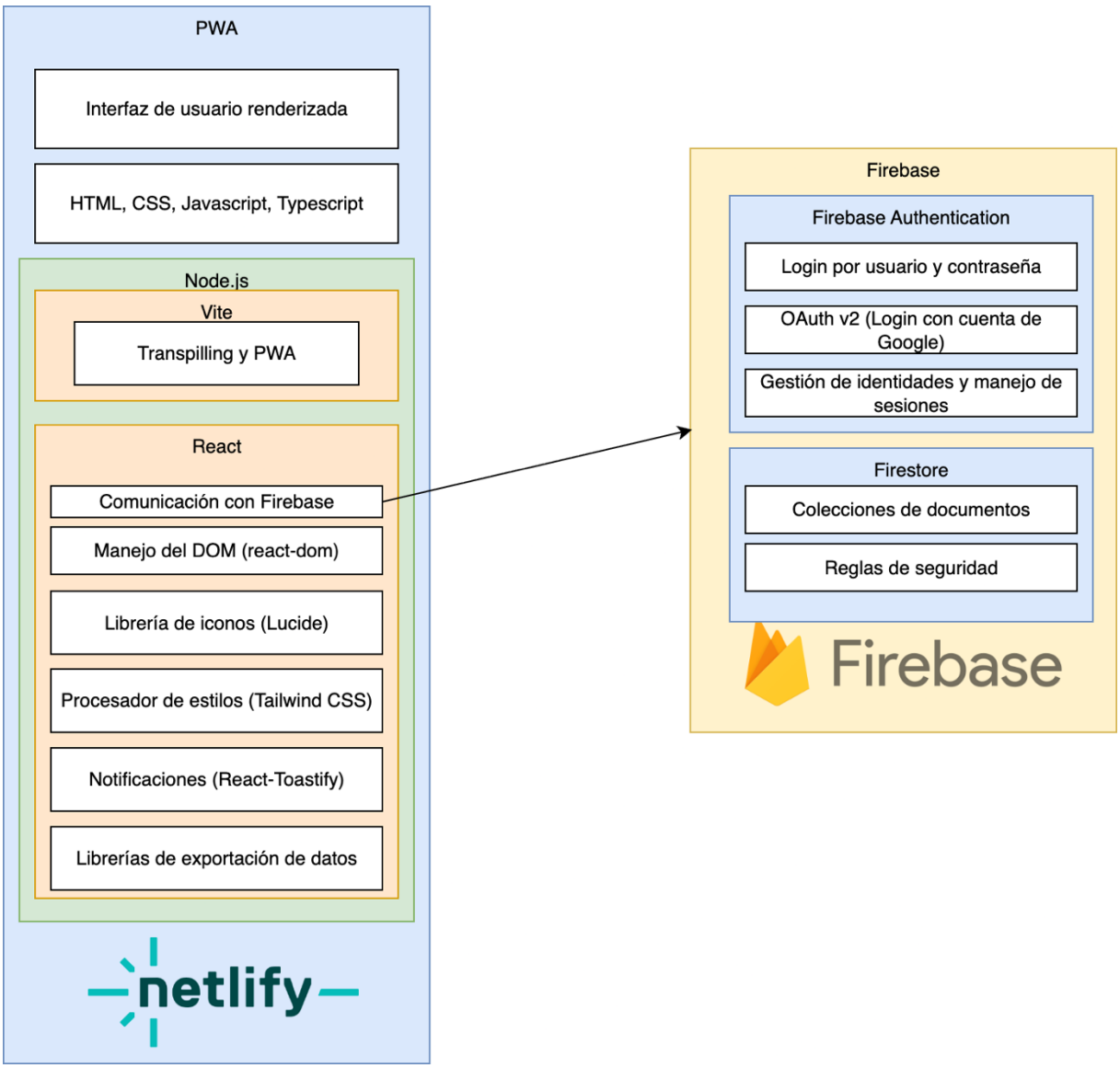

Figura 6 - Arquitectura física

En este diagrama se pretende representar la relación entre el uso de los dos servicios cloud que se usarán para el despliegue de la plataforma, recayendo en Netlify la capa arquitectónica lógica de Presentación, haciendo las veces de servidor web que sirve la aplicación compilada por Vite. En cambio, en Firebase los servicios de BaaS (Backend as a Service) que proporcionarán la capa arquitectónica lógica de Lógica de Negocio y Datos. Estos dos servicios se comunicarán mediante una serie de claves API que ofrece Firebase, guardadas de forma segura como variables de entorno en Netlify.

7. Implementación y desarrollo

Esta sección describe el conjunto de herramientas, estándares y procesos que se han seguido para construir la aplicación **TrainStats**.

- **Gestor de Paquetes:** Se utiliza un gestor de paquetes de Node.js como npm para manejar las dependencias del proyecto.
- **Entorno de Desarrollo y Compilación:** El proyecto está construido sobre Vite, un entorno de desarrollo moderno que ofrece técnicas modernas como el hot reload (actualización automática a la vez que se desarrolla) y transpiling de React a PWA (lo que necesitamos para construir una PWA)
- **Framework Principal y Lenguaje:** La aplicación utiliza React (versión 18.3+) como librería principal para la construcción de la interfaz de usuario. El lenguaje de programación es TypeScript, lo que añade tipado estático al código para mayor robustez y mantenibilidad.
- **Estilos:** Para el diseño de la interfaz se utiliza el framework CSS Tailwind CSS, que permite crear interfaces personalizadas de forma rápida mediante clases de utilidad. Se apoya en PostCSS y Autoprefixer para procesar y optimizar el CSS.
- **Base de Datos y Autenticación:** Se integra con los servicios de Firebase (usando el SDK modular v10+) para gestionar la base de datos, la autenticación de usuarios y otros servicios de backend.
- **Librerías Adicionales Clave:**
    - ***Lucide React:*** Para la implementación de iconos SVG consistentes y ligeros.[3]
    - ***React Toastify:*** Para mostrar notificaciones y alertas no intrusivas a los usuarios.[4]
    - ***html2canvas y jsPDF:*** Utilizadas para generar exportaciones de componentes o vistas a formato de imagen y PDF.[5]

[3] Los iconos completos están en https://lucide.dev/icons/

[4] Más información: https://www.npmjs.com/package/react-toastify

[5] Documentaciones en https://html2canvas.hertzen.com/ y https://github.com/parallax/jsPDF respectivamente

- Calidad de Código: Se emplea **ESLint** para el análisis estático del código, garantizando que se sigan las convenciones y se detecten errores de forma temprana.
- Visual Studio Code como IDE

8. Pruebas

La estrategia de pruebas de este proyecto tiene como objetivo garantizar la calidad, fiabilidad y corrección del software desde una perspectiva funcional y centrada en el usuario. El enfoque adoptado es el de **pruebas de caja negra**, lo que significa que todas las validaciones se realizan interactuando con la interfaz de la aplicación, sin necesidad de conocer su estructura de código interna. Se verifica el comportamiento del sistema basándose en las entradas proporcionadas y las salidas esperadas, tal como lo haría un usuario final.

Esta estrategia se articula en dos niveles de prueba manuales y secuenciales:

3.11. Pruebas de Integración (Manuales)

El objetivo de las pruebas de integración es verificar que los distintos módulos o componentes del sistema funcionan correctamente cuando se combinan. Se busca asegurar que la comunicación y el flujo de datos entre las diferentes partes de la aplicación son correctos.

- **Proceso**: Estas pruebas se realizarán de forma manual directamente sobre la aplicación desplegada en un entorno de desarrollo. Se seguirán una serie de casos de prueba diseñados para cubrir las interacciones clave entre módulos.
- **Ejemplo**: Se probará el flujo completo de "Registrar Nuevo Entrenamiento". El proceso consistirá en:
    - Iniciar sesión con un usuario de prueba.
    - Navegar a la página de registro de entrenamiento.
    - Rellenar el formulario con datos válidos de ejercicios, series y repeticiones.
    - Guardar el entrenamiento.
    - Verificar en la sección "Historial" que el nuevo entrenamiento aparece correctamente.
    - Validar, accediendo directamente a la consola de la base de datos Firestore, que el documento del entrenamiento se ha creado con la estructura y los datos correctos, y que está correctamente asociado al ID del usuario.

3.12. Pruebas de Aceptación del Usuario (UAT)

Esta es la fase final de validación y es la más crucial, ya que determina si el software cumple con los requisitos y expectativas del usuario final. Estas pruebas validan el sistema en su

totalidad, confirmando que cada uno de los Casos de Uso definidos en la fase de Análisis se ha implementado correctamente.

Parte III - Manuales

9. Manual de Despliegue

9.1. Arranque en el entorno de desarrollo (en local)

Para poder arrancar el proyecto en un ordenador y poder levantarlo en local, el primer paso después de instalar Node y clonar el repositorio donde se aloja el código[6] es ejecutar **npm install.** Este comando instalará todas las dependencias necesarias para poder arrancar el proyecto. El proceso tarda un par de minutos en la mayoría de casos y cuando finaliza, podemos arrancar el proyecto en local.

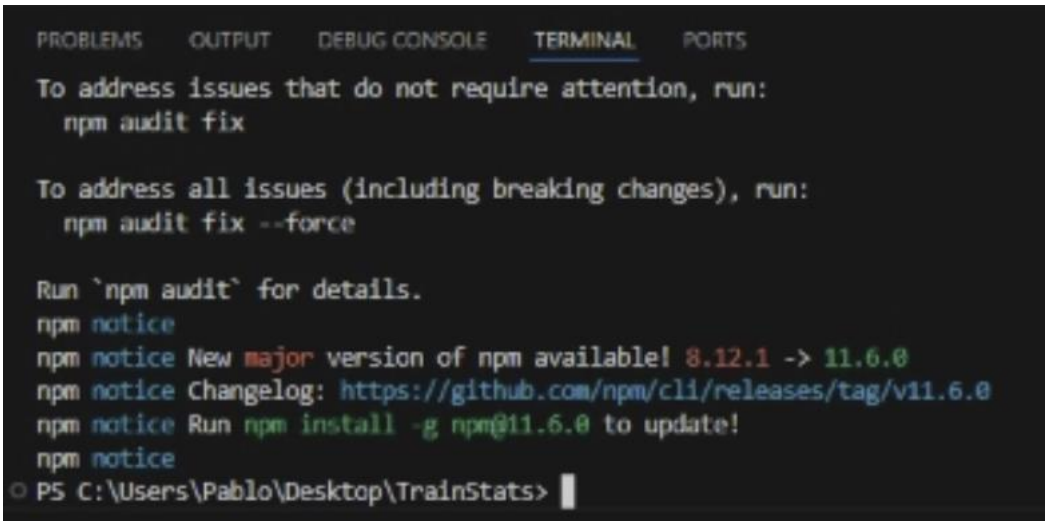

Figura 7 - Resultado de la instalación de dependencias con npm install

Posteriormente, debemos añadir las **variables de entorno,** que por seguridad y al ser el repositorio de GitHub público, no han sido añadidas al mismo. Para ello, bastará con crear un fichero .env y pegarle las siguientes variables de comunicación con Firebase:

```
VITE_FIREBASE_API_KEY=AIzaSyDysX7ScymoHXAjhNb9oxnNJbGpK9bYGqg
VITE_FIREBASE_AUTH_DOMAIN=sample-7cd8f.firebaseapp.com
VITE_FIREBASE_PROJECT_ID=sample-7cd8f
VITE_FIREBASE_STORAGE_BUCKET=sample-7cd8f.firebasestorage.app
VITE_FIREBASE_MESSAGING_SENDER_ID=579777483807
VITE_FIREBASE_APP_ID=1:579777483807:web:1f1c94932c5085f54a712a
```

[6] Repositorio público en https://github.com/pablo-gil-m/TrainStats

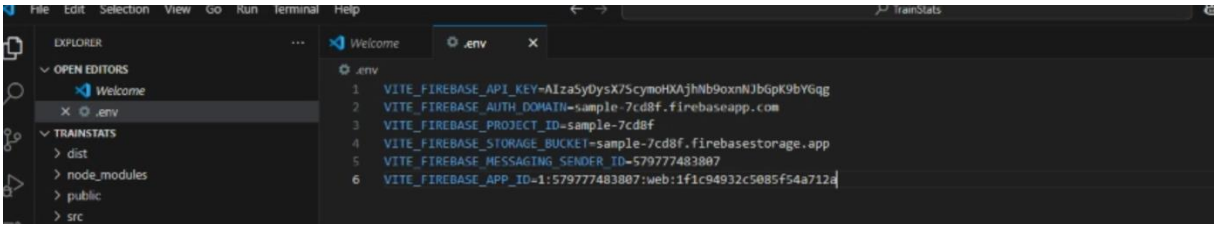

Figura 8 - Inyección de variables de entorno mediante un .env

Después, para levantar el proyecto en sí podemos usar **npm run dev.** Este comando ejecuta el proyecto en el entorno de desarrollo y tiene características como el **hot reload,** útil para recargar el proyecto en tiempo real con los cambios que vayamos haciendo cuando desarrollamos. Si queremos probar una versión similar a la que se desplegará, ejecutaremos **npm run build** para que Vite compile el código y **npm run preview** para que sirva el código compilado de la misma forma que haría un servidor web tradicional.

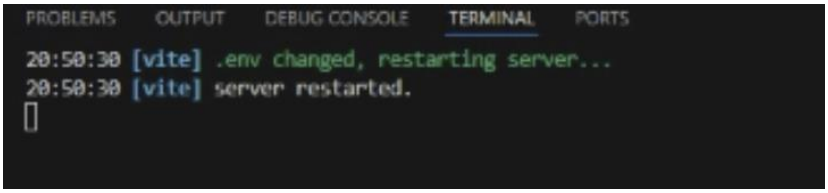

Figura 9 - Aviso de Vite de que se ha producido un "hot reload", al editar un archivo

9.2. Despliegue en entorno cloud

Para desplegar la aplicación en un entorno cloud, partiré del ejemplo que he realizado yo en Netlify. Si bien el proceso es similar para otros proveedores (cargar el código compilado, inyectar las variables de entorno y servir), Netlify ofrece algunas características adicionales que me han hecho considerarlo como el más interesante, por ejemplo, ofrece de manera gratuita un subdominio (en mi caso trainstats.netlify.app), certificado SSL para la conexión HTTPS y la posibilidad de importar proyectos directamente desde GitHub.

Tras registrarme en Netlify y conectarlo con mi cuenta de GitHub, al darle clic a **Import existing project** puedo seleccionar qué repositorio de GitHub importar en la plataforma, y posteriormente, asignar al repositorio diversos parámetros como el comando a ejecutar para llevar a cabo la compilación (en nuestro caso, **npm run build),** así como la inyección de las variables de entorno previamente descritas:

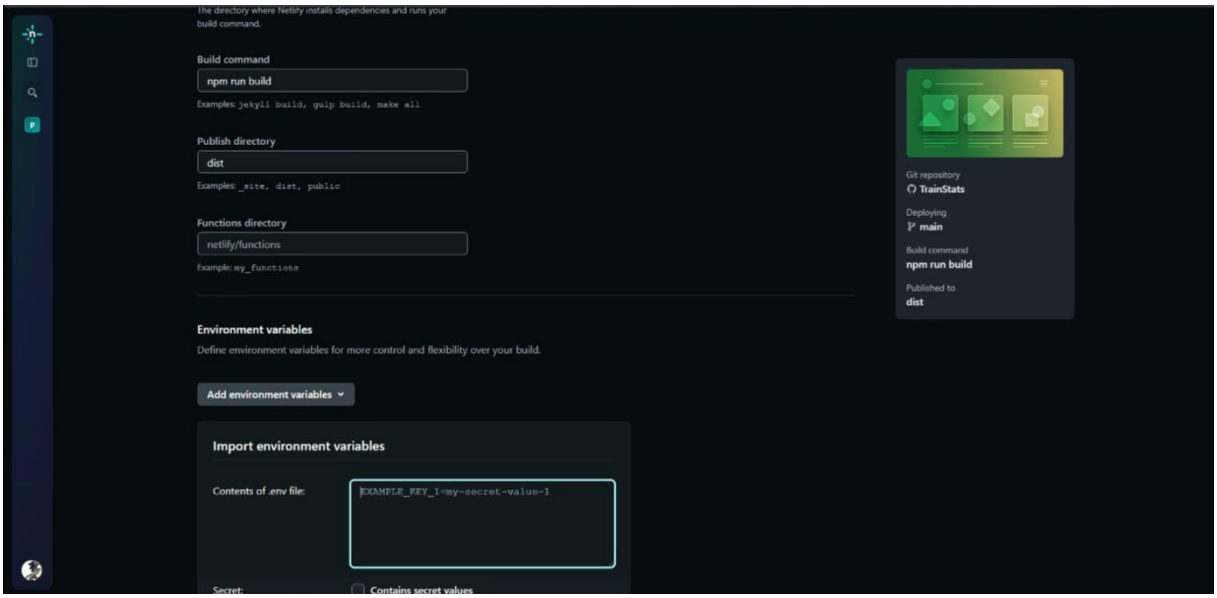

Figura 10 - Configuración del proyecto en Netlify

Finalmente, tras configurarlo, en el dashboard principal de Netlify vemos cómo el despliegue se está realizando, recibiendo una notificación cuando éste ha finalizado:

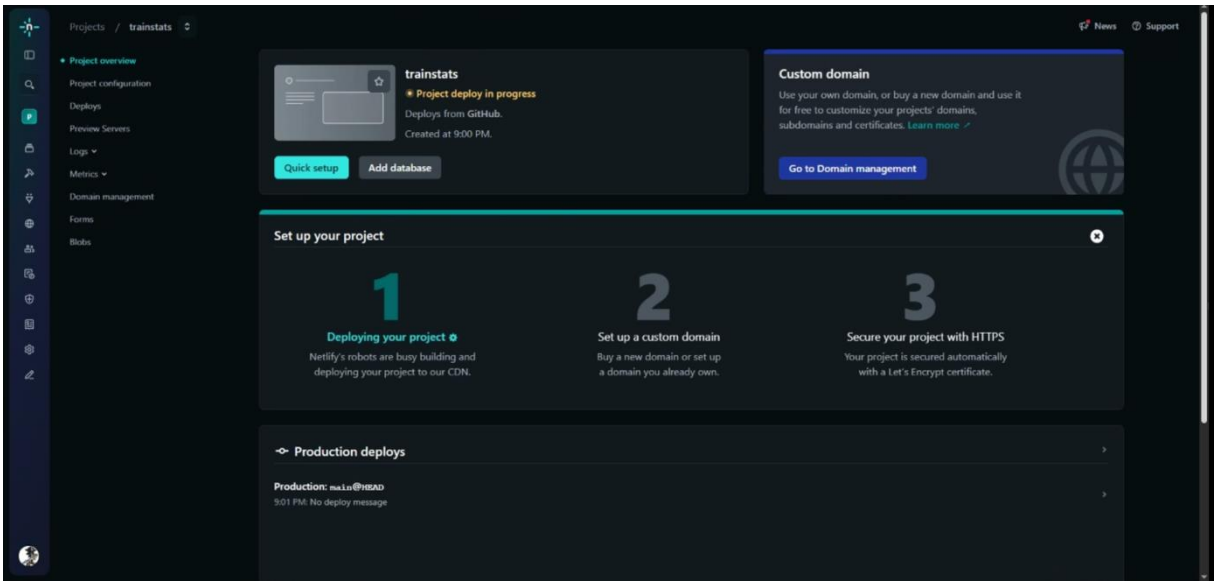

Figura 11 - Despliegue en curso

Netlify ofrece además la posibilidad de añadir dominios personalizados, lo cual aumenta su viabilidad comercial para proyectos reales.

El último paso, será añadir este dominio de Netlify como dominio autorizado dentro de Firebase Authentication, ya que de lo contrario, el usuario nunca podrá iniciar sesión. Esto es

parte de las características de seguridad que ofrece Firebase al conectarlo con servicios externos:

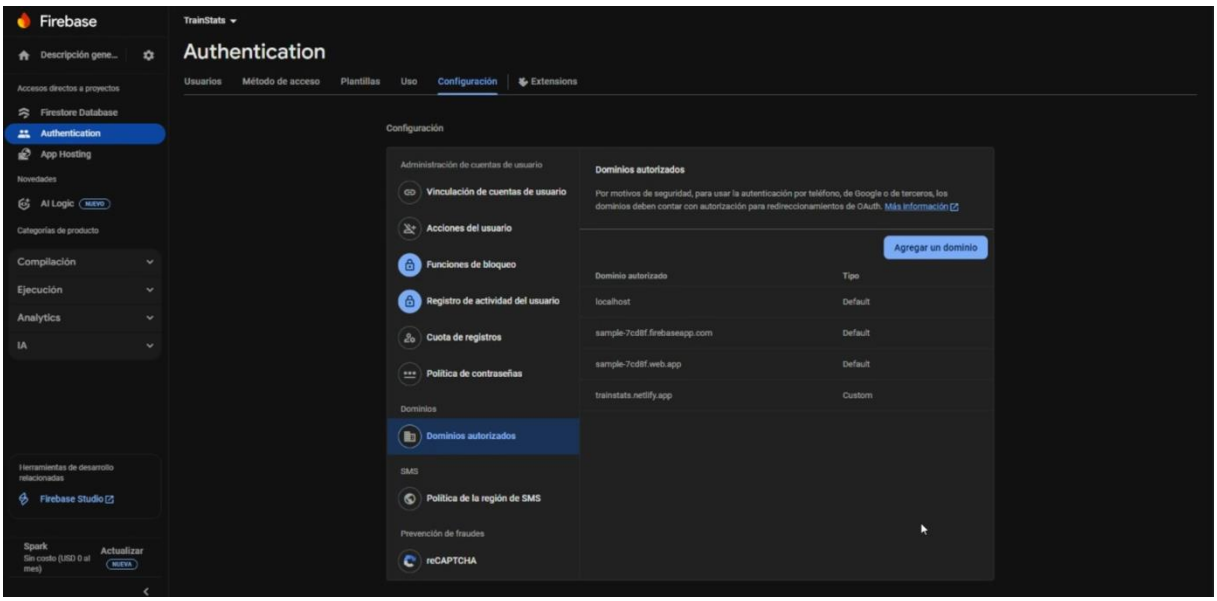

Figura 12 - Dominios autorizados en Firebase Authentication

Otra de las características de seguridad de Firebase es el **log de actividad de usuarios,** que nos permite ver en todo momento qué usuarios existen en el sistema y cuándo han iniciado sesión:

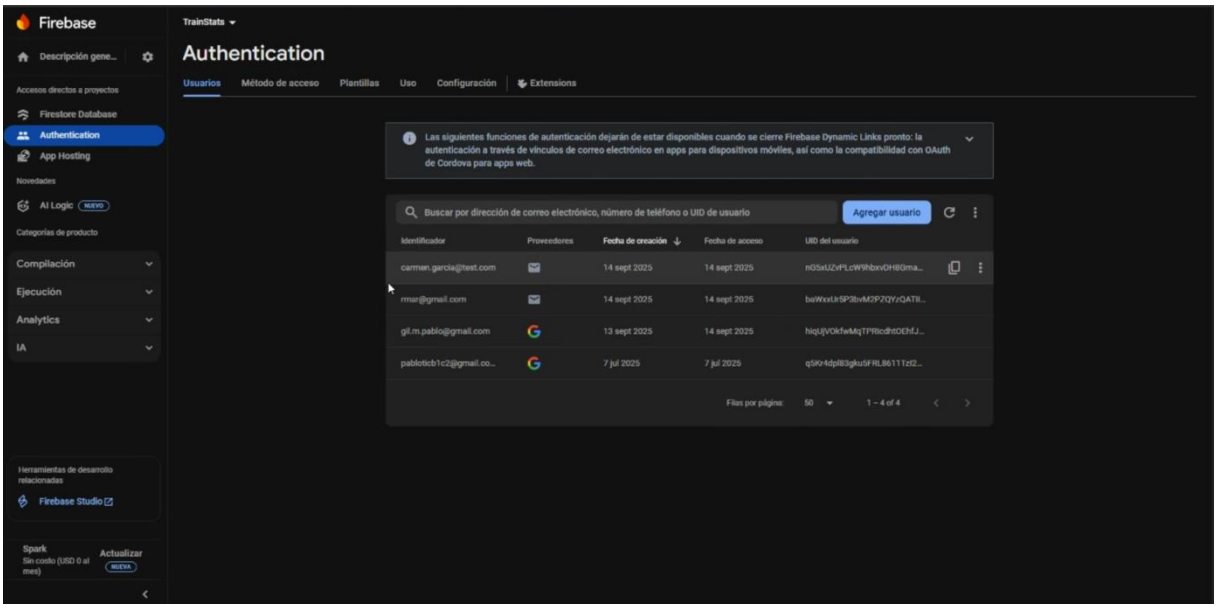

Figura 13 - Log de actividad de usuarios de Firebase Authentication

Por último, con todos estos pasos realizados, si entramos en el dominio podremos no solo usar la app en su versión web sino también instalarla dada su naturaleza de aplicación web híbrida (PWA), viéndose así, por ejemplo, en un equipo de escritorio con Windows:

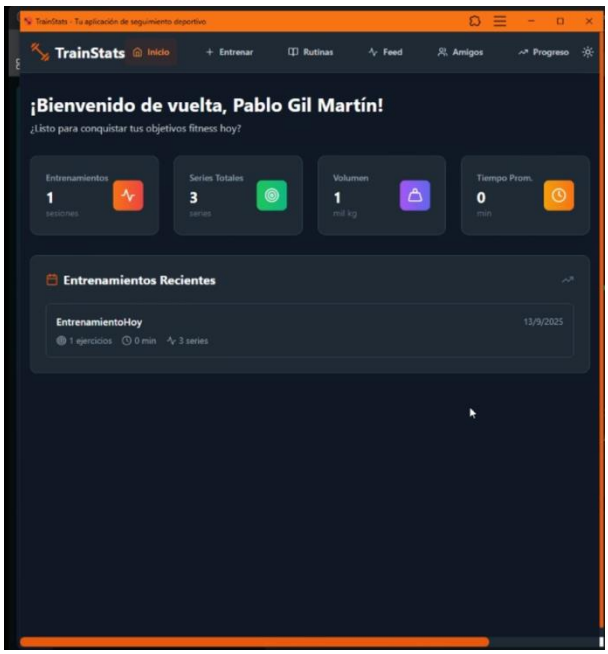

Figura 14 - Aplicación instalada en Windows

10. Manual de Usuario

10.1. Usuario no identificado

Se accede a la aplicación mediante el siguiente enlace:

https://trainstats.netlify.app

Figura 15 - Landing page o página de inicio de sesión

Desde la página de bienvenida se podrá acceder a la aplicación identificándose con cuenta de google, o de manera convencional especificando el correo electrónico y la contraseña de una cuenta que previamente se haya registrado.

Para registrar una cuenta utilizando un correo electrónico se debe seleccionar el enlace llamado “Regístrate” del panel de login.

Al hacer click en ese enlace, el panel de login se transforma en el de registro, en el que se podrá crear una nueva cuenta especificando nombre, correo electrónico y contraseña.

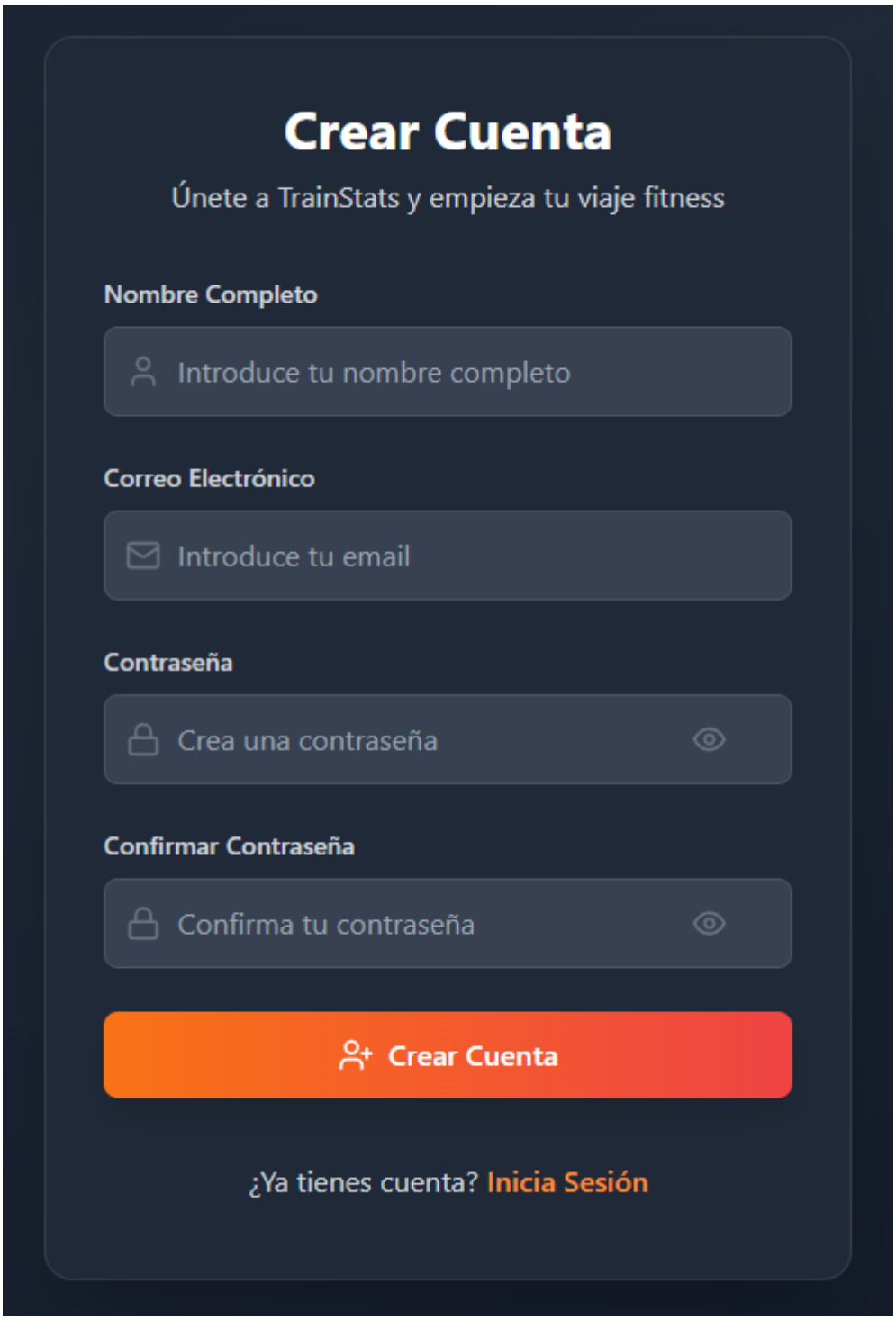

Figura 16 - Componente de registro de usuarios

Finalmente, al registrar una nueva cuenta, aparecerá el siguiente modal en el que se deberá introducir el nombre de usuario por el cuál podrán identificarse los usuarios.

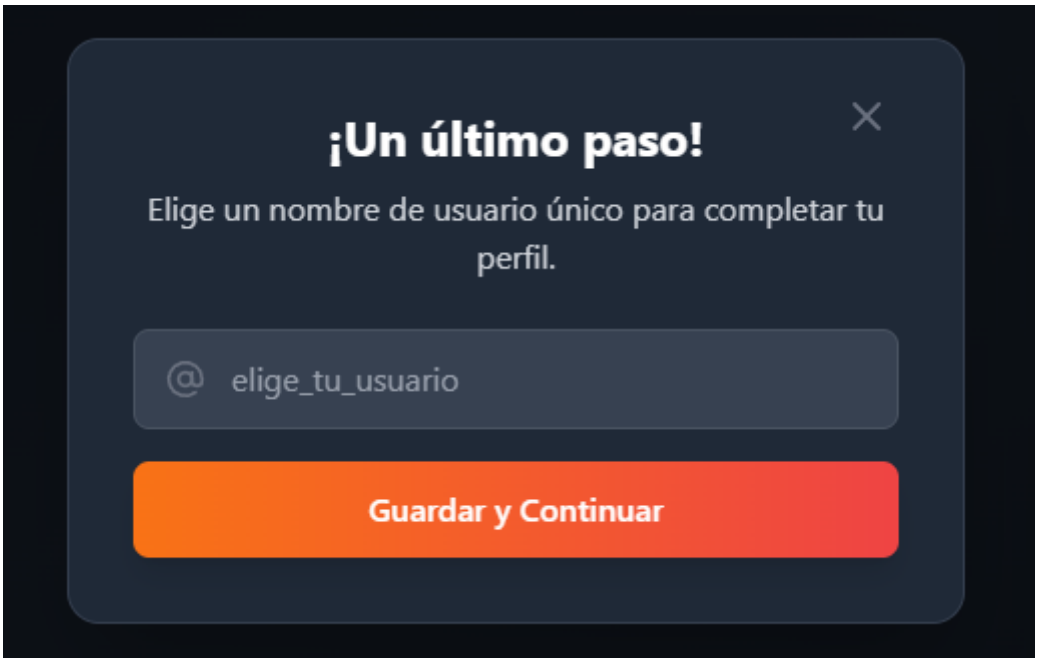

Figura 17 - Modal de asignación de nombre de usuario único

Al elegir un nombre de usuario se llevará al nuevo usuario a la aplicación.

10.2. Usuario identificado

Al entrar en la aplicación le aparecerá la siguiente pantalla de inicio al usuario.

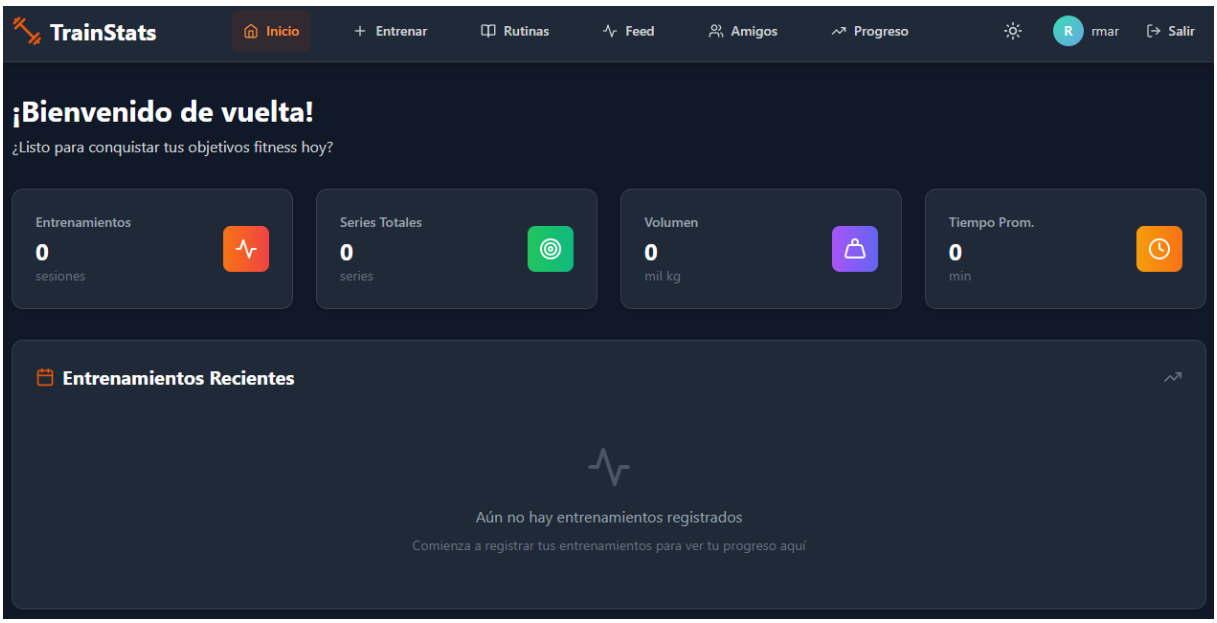

Figura 18 - Dashboard "Inicio"

En esta pantalla de inicio se muestra un resumen de los entrenamientos que ha realizado recientemente.

Desde la barra de navegación, el usuario podrá acceder a los distintos paneles de las funcionalidades de la aplicación, así como a personalizar la interfaz entre modo claro y modo oscuro, y salir haciendo logout de la aplicación.

En el panel de Entrenar el usuario encontrará dos opciones: registrar un nuevo entrenamiento y consultar los entrenamientos realizados anteriormente

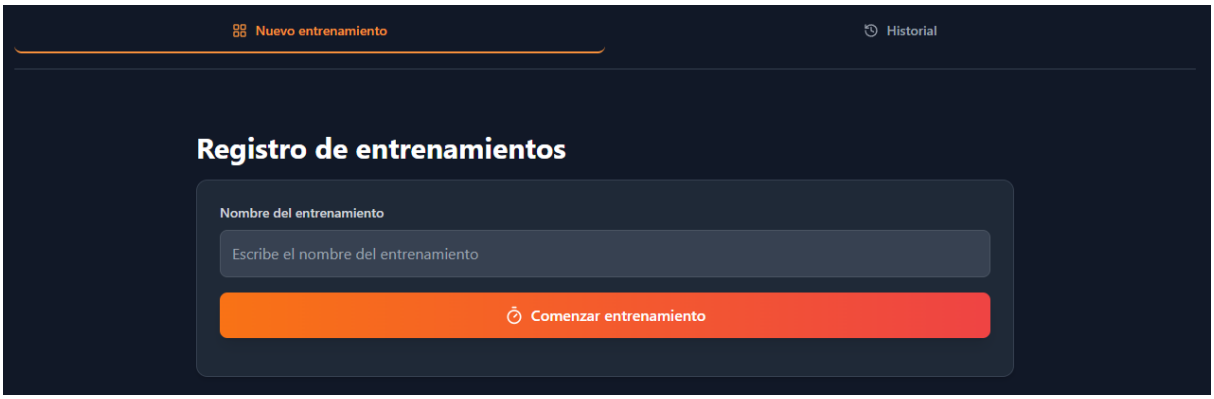

Figura 19 - Pantalla de registro de entrenamientos

Para registrar un nuevo entrenamiento, el usuario debe darle un nombre (o no, y se generará automáticamente basado en la fecha) y hacer click en Comenzar entrenamiento, que le llevará a la siguiente pantalla.

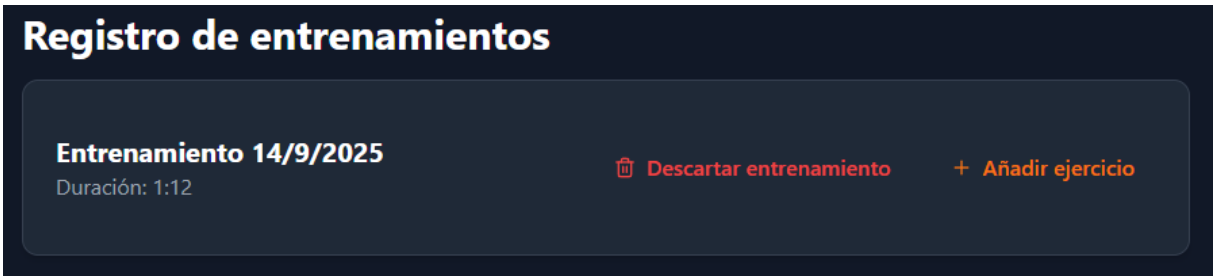

Figura 20 - Vista de un entrenamiento en curso

Aquí se empezará a cronometrar el entrenamiento y se podrán añadir los ejercicios realizados en este, así como cancelarlo eliminándolo.

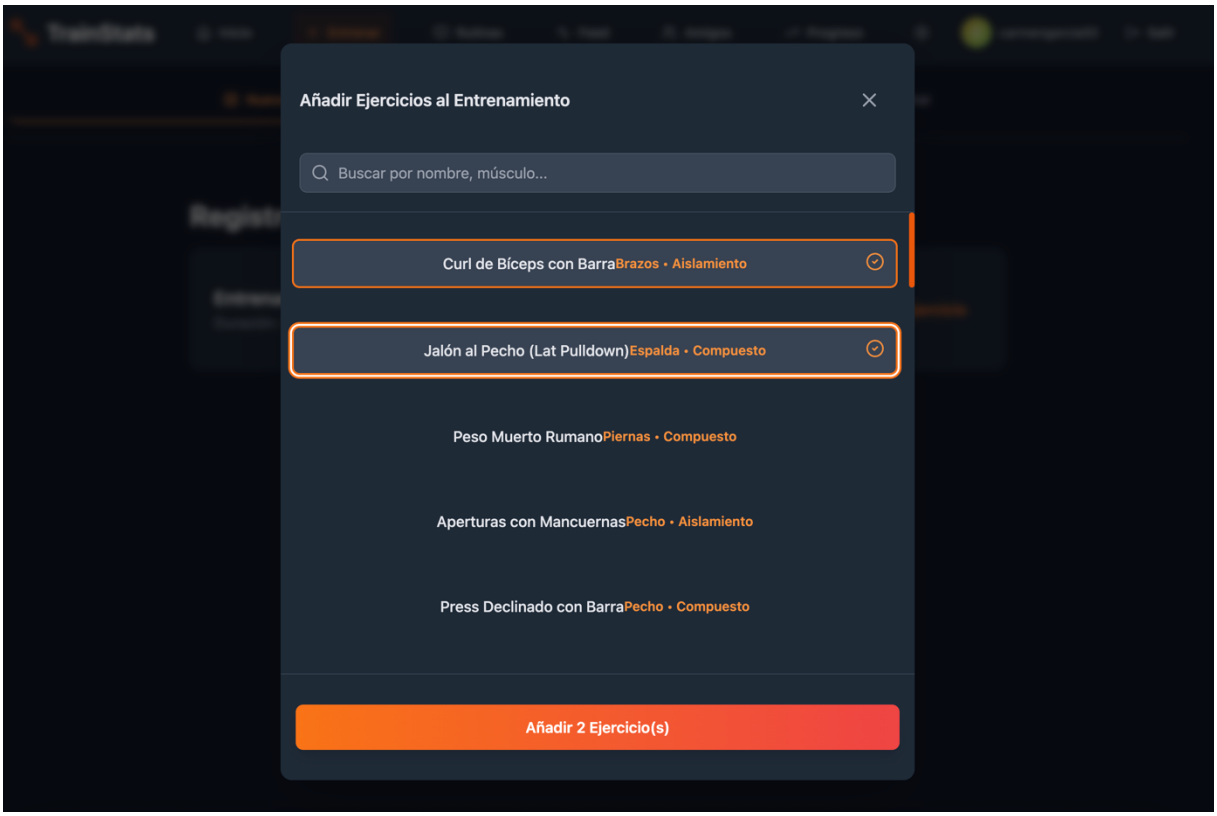

Figura 21 - Modal de asignación de ejercicios

Utilizando este modal se selecciona el ejercicio o los ejercicios a realizar:

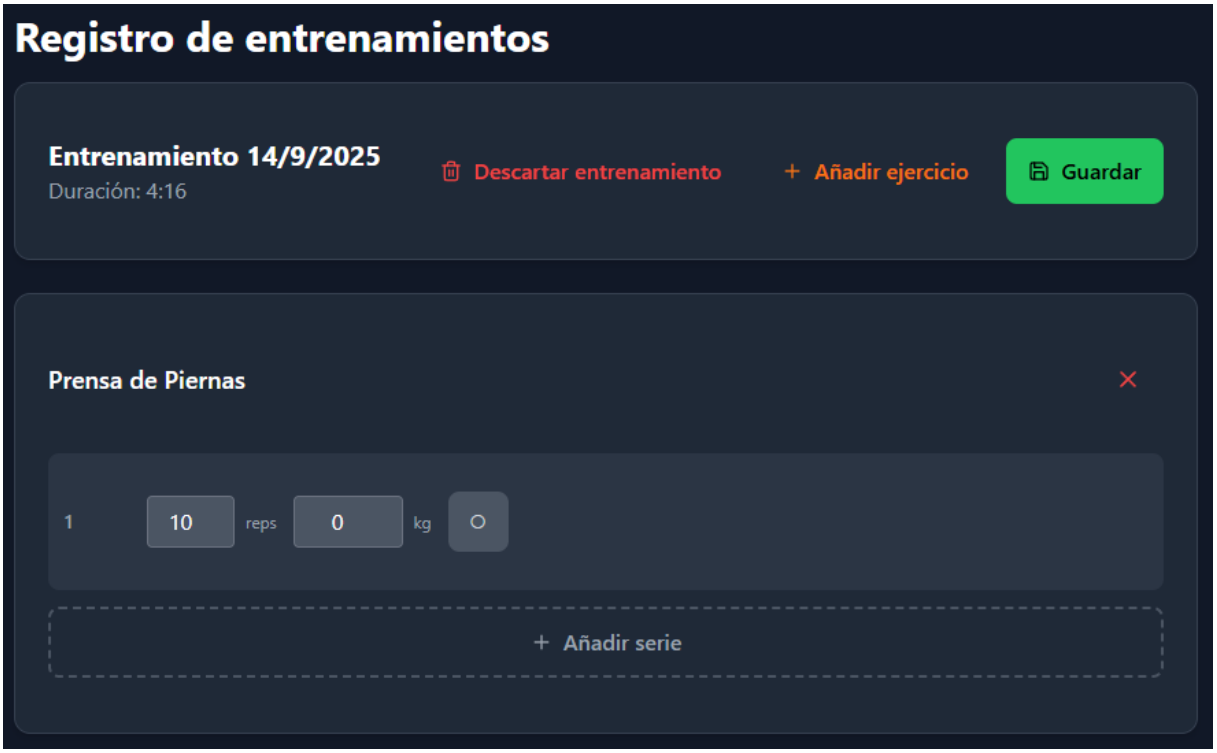

Figura 22 - Vista de asignación de series a un ejercicio asignado

Una vez añadido el ejercicio el usuarió ajustará las repeticiones, la carga levantada en kg, así como añadir el número de series que le correspondan al ejercicio dentro del entrenamiento. Cada serie se marcará como realizada activando el toggle de la derecha.

Figura 23 - Vista de asignación de series con un ejemplo de caso real

Por ejemplo, así se representaría un entrenamiento en el que se habían planificado cuatro series pero finalmente solo se realizaron dos.

Al terminar de registrar todos los ejercicios del entrenamiento, el usuario debe guardarlo presionando el botón Guardar.

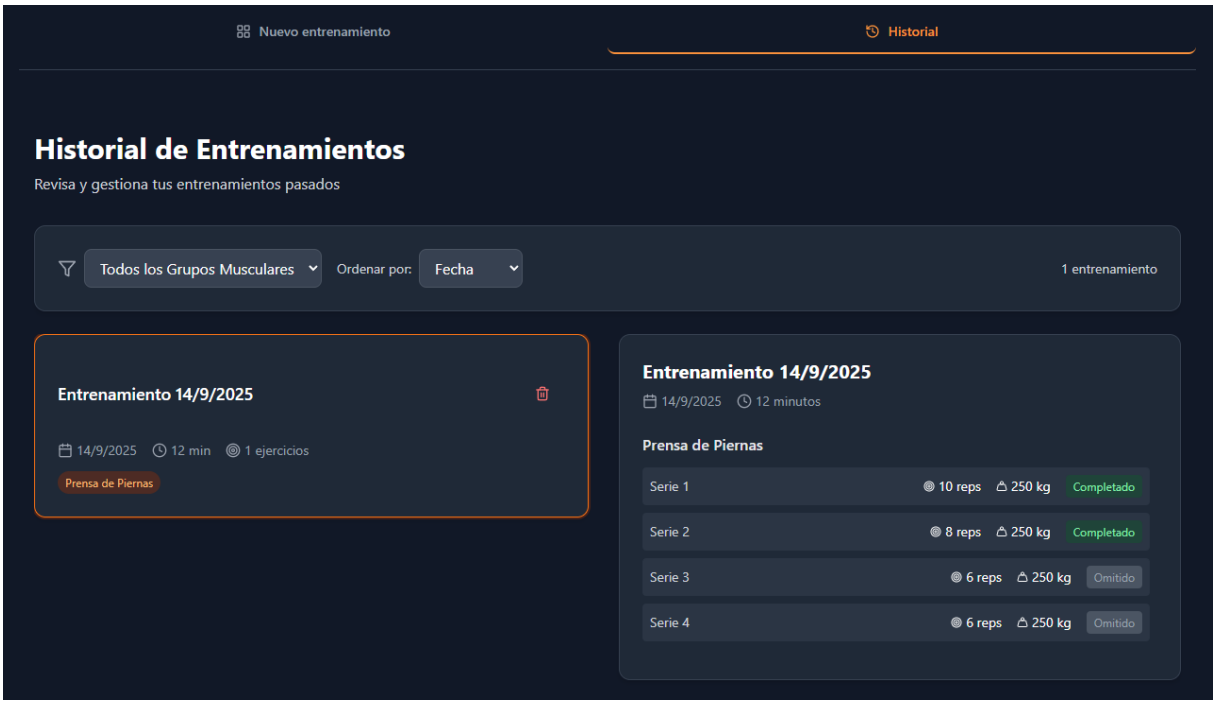

Figura 24 - Historial de entrenamientos

De esta manera, el entrenamiento queda registrado y lo podemos consultar en el historial de entrenamientos.

En el panel de rutinas podremos consultar nuestras rutinas, así como buscar las que hayan publicado otros usuarios. Para crear una nueva rutina se presiona el botón correspondiente.

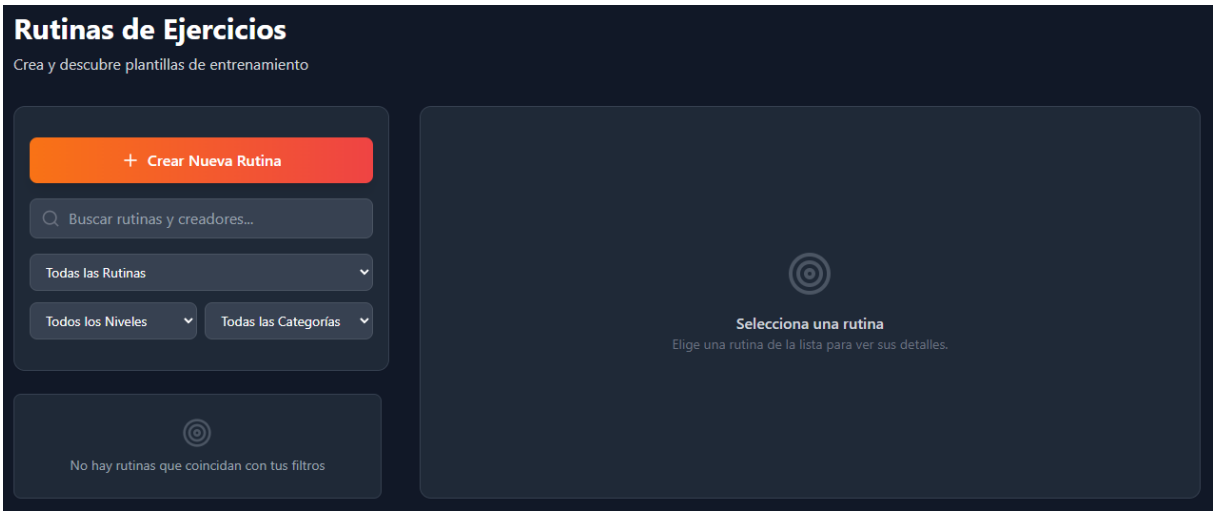

Figura 25 - Dashboard de gestión de rutinas

El usuario podrá rellenar los siguientes campos de la rutina, siendo solamente obligatorio el nombre.

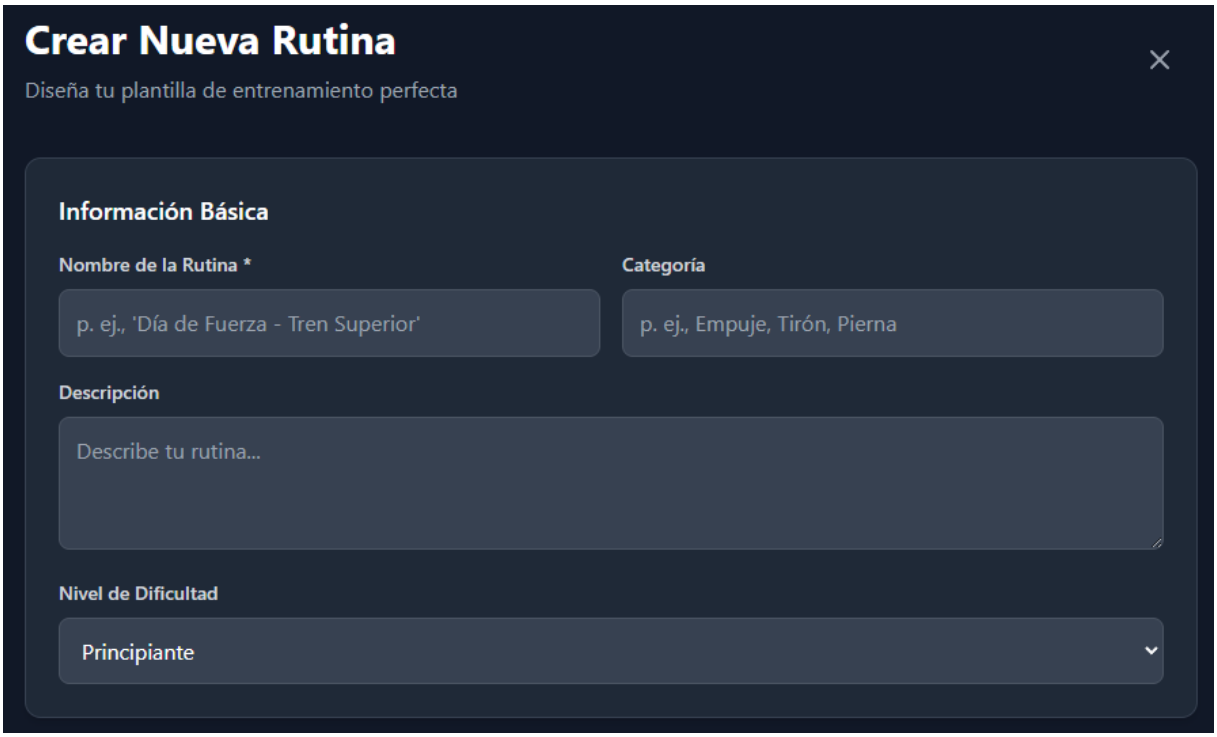

Figura 26 - Creación de una nueva rutina

Para añadir los ejercicios a la rutina se hará de una manera muy similar a cuando se registra un entrenamiento, con la diferencia de que se debe especificar el tiempo de descanso entre series y además se podrá añadir un comentario referente a este ejercicio.

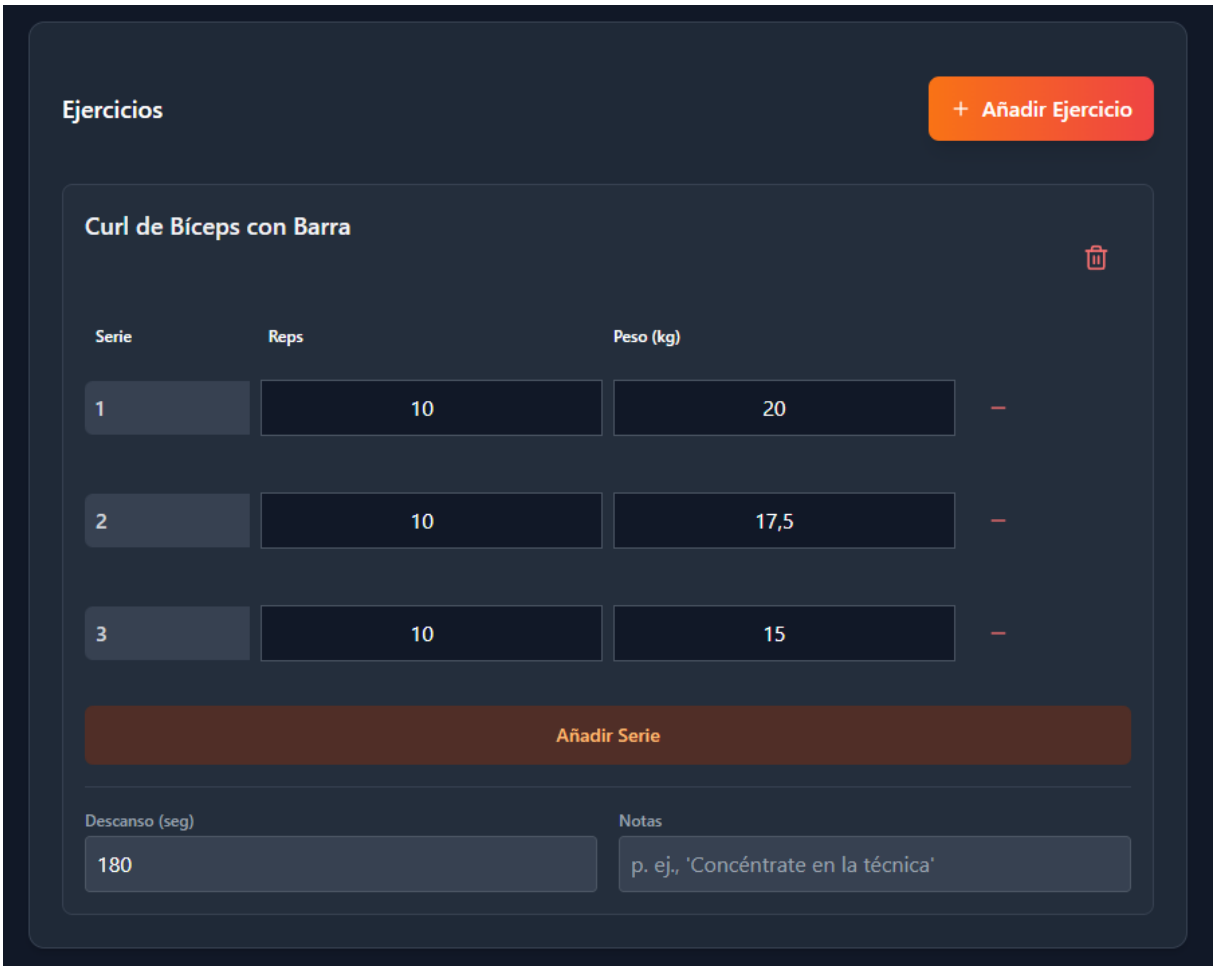

Figura 27 - Asignación de ejercicios y series a rutinas

Una vez guardada la rutina aparecerá en el menú de rutinas, donde el usuario podrá ver sus detalles y tendrá distintas opciones para interactuar con ella:

- Comenzar un entrenamiento utilizándola como plantilla
- Exportarla a un archivo, ya sea una imagen o documento PDF para poder compartirla fácilmente con personas que no tengan la aplicación
- Modificar cualquiera de sus parámetros
- Duplicarla
- Eliminarla

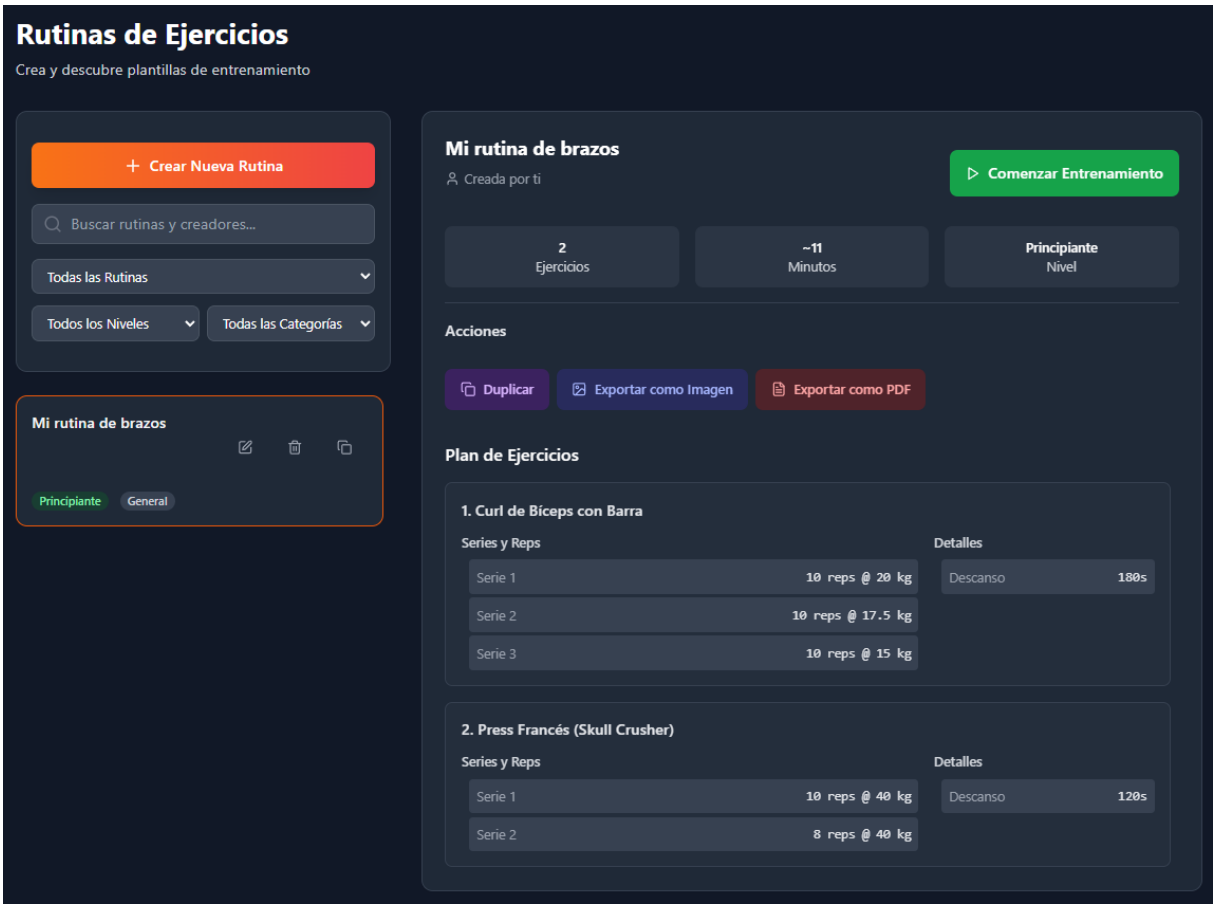

Figura 28 - Vista de gestión de rutinas: consulta de una rutina previamente existente

En la sección de Amigos el usuario podrá buscar otros usuarios de la aplicación para seguirlos. El buscador implementa lazy search por lo que con poner el comienzo del nombre de usuario ya aparecerá.

Figura 29 - Buscador de usuarios

Al seguir a un usuario, aparecerá en la lista de seguidos como se ve a continuación.

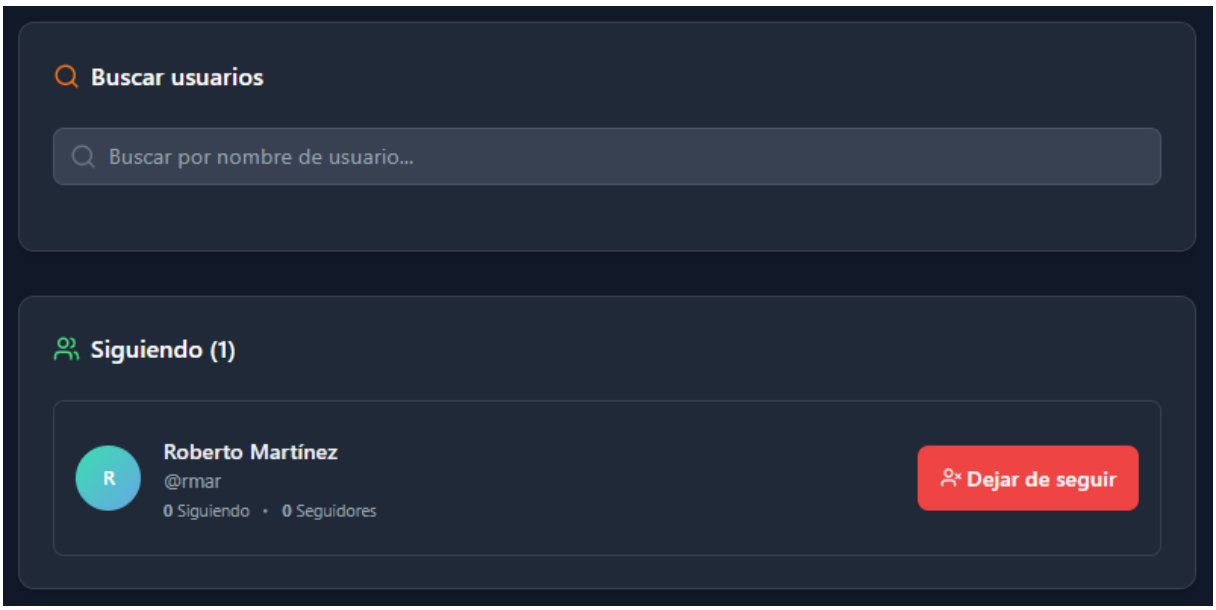

Figura 30 - Usuario recién seguido

En la pantalla Feed se podrán ver los entrenamientos realizados por los usuarios a los que se sigue, así como los ejercicios que los componen.

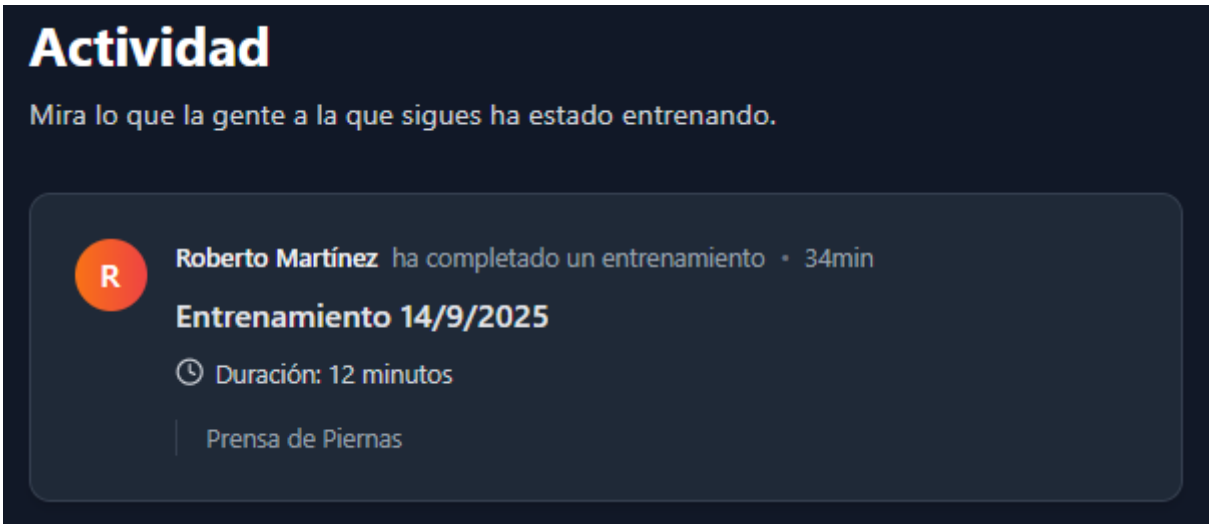

Figura 31 - Feed de actividad de usuarios a los que sigues

En la pantalla Progreso se accede al seguimiento del progreso, en el que se puede consultar la evolución en cargas durante distintos periodos de tiempo, además de un resumen semanal.

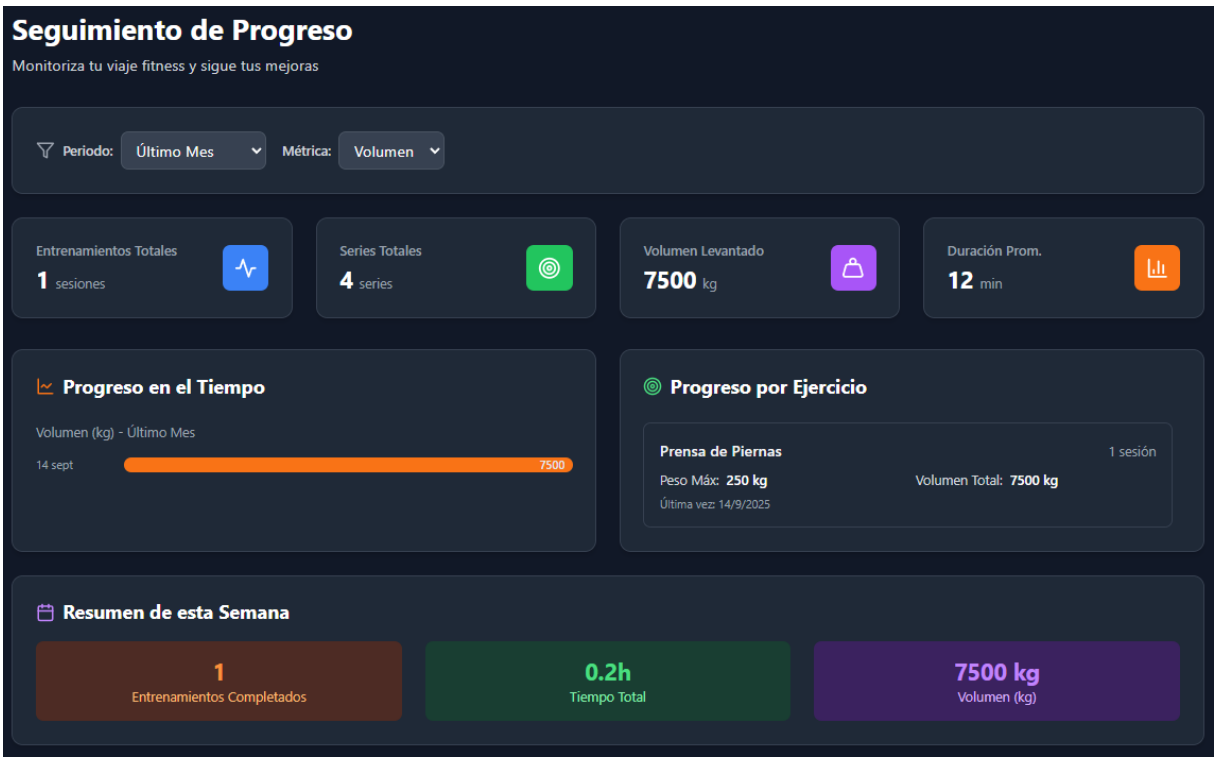

Figura 32 - Dashboard de seguimiento de progreso

Parte IV

11. Conclusiones y líneas de trabajo futuras

El recorrido de este Trabajo Fin de Grado ha culminado con la entrega de un producto de software funcional que responde a una necesidad real en el ámbito del seguimiento deportivo. La culminación de este proceso permite extraer una serie de conclusiones tanto a nivel de producto como de aprendizaje técnico y metodológico.

El producto desarrollado representa un núcleo funcional sólido y una base tecnológica estable, pero su potencial de crecimiento es enorme. A continuación, se detallan varias líneas de trabajo futuras que podrían expandir y enriquecer la aplicación en futuras versiones.

- **Funcionalidades de Entrenamiento Avanzadas**
  - ***Cronómetro de Descanso****:* Integrar un temporizador configurable entre series y ejercicios directamente en la interfaz de registro en lugar de ser un valor definido para una rutina.
  - ***Registro de Récords Personales (PRs)****:* Detectar y almacenar automáticamente las mejores marcas del usuario en diferentes ejercicios (ej. mayor peso levantado, mayor número de repeticiones).
  - ***Soporte para Cardio y Otros Tipos de Ejercicio****:* Ampliar el modelo de datos para permitir el registro de actividades cardiovasculares (distancia, tiempo, velocidad) o ejercicios isométricos (duración).
- **Funcionalidades Sociales y de Comunidad**
  - ***Compartir Entrenamientos***: Dar la opción de generar un enlace público o una imagen de un entrenamiento completado para compartirlo en redes sociales.
  - **Clasificaciones (Leaderboards)**: Crear clasificaciones semanales o mensuales (privadas, entre amigos) basadas en métricas como el volumen total levantado o el número de entrenamientos.
  - **Gamificación mediante logros y medallas**: Crear un sistema de recompensas virtuales por alcanzar hitos (ej. "Primer mes entrenando", "Levantados 10.000 kg en total").
  - **Establecimiento de metas**: Permitir que el usuario se fije objetivos personales (ej. "Entrenar 3 veces por semana") y visualizar su cumplimiento.

- **Mejoras Técnicas y de Plataforma**
    - **Notificaciones Push**: Aprovechar las capacidades de la PWA para enviar recordatorios de entrenamiento o mensajes de motivación.
    - **Informes de Progreso Detallados**: Generar informes y gráficos más sofisticados sobre el volumen de entrenamiento, la frecuencia, la intensidad y la densidad por grupo muscular.
    - **Integración con APIs de Salud y dispositivos wearables**: Conectar la aplicación con plataformas como Google Fit o Apple Health para sincronizar datos de actividad y ofrecer una visión holística de la salud del usuario.
    - **Internacionalización (i18n)**: Adaptar la aplicación para soportar múltiples idiomas, ampliando así su alcance global.
    - **Migración a React Native**: Explorar la posibilidad de reutilizar parte de la lógica de negocio para desarrollar una aplicación móvil nativa con React Native, lo que permitiría una integración más profunda con el hardware del dispositivo (sensores, notificaciones avanzadas).

En definitiva, este proyecto no solo concluye con la entrega de una aplicación funcional, sino que también abre la puerta a un ecosistema de funcionalidades que podrían convertirla en una herramienta de referencia en el sector del fitness digital.

Bibliografía

Developers, G. (s.f.). *Progressive web apps.* Obtenido de web.dev: https://web.dev/explore/progressive-web-apps?hl=es-419

Google. (s.f.). *Documentación.* Obtenido de Firebase: https://firebase.google.com/docs?hl=es-419

Netlify. (s.f.). *Netlify Docs.* Obtenido de https://docs.netlify.com/

React Developers. (s.f.). *Learn React.* Obtenido de https://react.dev/learn

Roberts, M. (22 de mayo de 2018). *Serverless architectures.* Obtenido de Martin Fowler: https://martinfowler.com/articles/serverless.html