**UNIVERSIDAD POLITÉCNICA DE MADRID**



PROYECTO DE FIN DE GRADO

GRADO EN TECNOLOGÍAS PARA LA SOCIEDAD DE LA INFORMACIÓN

CURSO 2024-2025

# Diseño y programación de una aplicación web de gestión personal

AUTOR: BRIAN EDUARDO PARDO GAONA

TUTOR: BORJA BORDEL SÁNCHEZ

# RESUMEN

La organización de nuestras actividades es importante para poder conseguir un alto rendimiento, tanto en el ámbito profesional como en el personal. En una sociedad en constante cambio y con múltiples exigencias resulta crucial contar con herramientas que faciliten el seguimiento de objetivos y la evaluación del progreso de estos a lo largo del tiempo.

Este proyecto tiene sentido al cubrir esta necesidad de optimizar la gestión personal de metas, permitiendo al usuario disponer de un sistema que ayude a planificar, priorizar y tener un seguimiento de las tareas. A través de esta herramienta se pretende también fomentar buenos hábitos que contribuyan al desarrollo personal y al cumplimiento de metas a corto, medio y largo plazo.

La propuesta está dirigida al colectivo universitario, ofreciendo una solución práctica que les permita mejorar su planificación académica y personal.

# ABSTRACT

The organisation of our activities is important to achieve high performance, both professionally and personally. In a society in constant change and with multiple demands, it is crucial to have tools that facilitate the monitoring of objectives and the evaluation of their progress over time.

This project makes sense by covering this need to optimise the personal management of goals, allowing the user to have a system that helps to plan, prioritise and monitor tasks. This tool also aims to promote good habits that contribute to personal development and the achievement of short, medium and long-term goals.

The proposal is aimed at university students, offering a practical solution that allows them to improve their academic and personal planning.

# ÍNDICE

# ÍNDICE DE FIGURAS

## 1.1. Contexto

Durante la época universitaria se adquieren tanto conocimientos técnicos como habilidades personales que son clave para el desarrollo del estudiante tanto a nivel profesional como personal, respectivamente. La gestión del tiempo, la planificación y el cumplimiento de metas personales son un conjunto de aspectos que se deben lograr para mantener un equilibrio entre su vida personal y académica.

Sin embargo, muchos estudiantes encuentran dificultades al momento de organizar tareas y establecer objetivos, así como mantener un seguimiento de los mismos. Por lo tanto, esto afecta de manera significativa a su rendimiento académico.

Teniendo en cuenta este problema, una aplicación para gestionar, planificar y hacer seguimiento de metas personales resulta ser una herramienta útil y necesaria para apoyar el desarrollo personal del conjunto universitario.

Finalmente, esta problemática no sucede únicamente para el alumnado, pues cualquier persona tiene la necesidad de organizarse. Es por ello que esta herramienta tiene el objetivo de organizar cualquier tipo de tarea y para cualquier tipo de usuario, simplemente el enfoque es hacia el alumnado debido a la naturaleza del proyecto que está dentro del marco universitario.

## 1.2. Objetivos

El objetivo de este proyecto es el desarrollo de una herramienta digital que facilite la organización personal y el seguimiento de objetivos, con el fin de contribuir a la mejora de gestión de tiempo y al aumento de la productividad dentro del entorno universitario.

Para alcanzar este propósito, se mencionan a continuación los objetivos específicos dentro del proyecto:

- Facilitar la planificación de tareas y metas a corto, medio y largo plazo.
- Permitir a los usuarios tener un seguimiento visual de su progreso a lo largo del tiempo.
- Fomentar hábitos organizativos que mejoren el rendimiento del alumnado.
- Reducir la sensación de desorganización al disponer de una aplicación con una estructura clara y simple.
- Potenciar la autonomía del usuario en la toma decisiones y en la priorización de sus tareas diarias.

## 1.3. Estructura del documento

Este documento presenta toda la vida la del proyecto, desde su organización hasta su desarrollo.

En primer lugar, se presenta información introductoria del proyecto que corresponde con el capítulo actual del documento en el que se centra en la presentación del propósito de la herramienta.

En segundo lugar, se presentarán las herramientas y tecnologías empleadas en el desarrollo de la aplicación web. Asimismo, se detallarán las medidas de seguridad implementadas, haciendo especial énfasis en cómo estas guardan la información del usuario y aseguran la integridad y confidencialidad del sistema.

En tercer lugar, se presentará la arquitectura del proyecto, detallando cuales son los puntos fuertes de usarla para este tipo de herramientas.

En cuarto lugar, se describirán todas las funcionalidades que existen dentro de la aplicación.

En quinto lugar, se mostrará la distribución de la base de datos, los motivos por los que se usa ese modelo y cuáles son las ventajas de las tecnologías utilizadas.

En sexto lugar, se realizará una extensa descripción del backend de la aplicación, mostrando cuál es la estructura dentro del proyecto y cuál es la función de cada uno de los elementos contenidos en él.

En séptimo lugar, se realizará la descripción de los elementos relevantes del frontend, indicando cuál es su finalidad y mostrando algunas de las soluciones a problemas complejos dentro de lado del cliente.

En octavo lugar, se presentación la solución de la página web mediante ilustraciones reales del estado final de la misma.

En noveno lugar, se describirán los aspectos éticos, medioambientales y económicos con los que este proyecto está comprometido.

En décimo lugar, se comentarán los aspectos que resultaron más problemáticos durante la realización del proyecto, tanto en su definición como en su desarrollo.

Finalmente, existirá un glosario con un espacio dedicado a explicar términos que pueden tener un significado técnico más profundo.

En el ámbito del desarrollo web existe una gran variedad de tecnologías que permiten soluciones muy distintas en desarrollo para la creación de proyectos complejos. Para este proyecto se han utilizado herramientas y frameworks que he considerado que son los más adecuados para construir una experiencia del usuario ágil. La elección de estas se debe también a su capacidad de facilitar el mantenimiento y la escalabilidad del producto a lo largo del tiempo.

## 2.1. Herramientas de desarrollo

### 2.1.1. Visual Studio Code

VS Code es un editor de código fuente ligero pero potente. Se ha utilizado este IDE debido a su integración Git, su sistema de depuración y su capacidad de personalización con diferentes extensiones.


*Ilustración 1. Visual Studio Code*

### 2.1.2. Github

GitHub es una plataforma de control de versiones y colaboración basada en Git. Permite alojar y revisar código, y es para lo que se ha utilizado en este proyecto.


*Ilustración 2. Github*

## 2.2. Base de datos

### 2.2.1. PostreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, reconocido por su estabilidad y potencia. Soporta transacciones complejas, relaciones entre tablas, y es altamente escalable. Ha sido utilizado para almacenar toda la información relacionada con usuarios, metas y objetivos, siendo éstas las entidades del proyecto.


*Ilustración 3. Postgre SQL*

## 2.3. Lenguajes

### 2.3.1. Frontend

- JavaScript → Lenguaje de programación principal para los dos lados de la aplicación, tanto del lado del cliente como del servidor.
- JavaScript Syntax eXtension (JSX) → Extensión de JavaScript usada en React que permite escribir código HTML dentro de JavaScript.
- CSS → Lenguaje utilizado para definir los estilos visuales de la interfaz, como colores, tamaños, fuentes, etc.
- HTML → Estructura básica del contenido mostrado en la interfaz web. Define los elementos del DOM que React manipula.

### 2.3.2. Backend

- JavaScript → También empleado en el servidor mediante Node.js, permitiendo una base común de lenguaje para front y back.
- SQL (a través de Sequelize) → Lenguaje para gestionar la base de datos relacional. Sequelize permite trabajar con SQL de forma más intuitiva mediante objetos JavaScript, sin necesidad de utilizar lenguaje SQL directamente.

## 2.4. Librerías

### 2.4.1. React

React es una librería usada para construir las interfaces de usuario basadas en componentes. Su uso facilita la creación de aplicaciones dinámicas, donde el DOM se actualiza eficientemente.

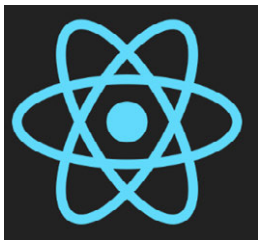


*Ilustración 4. React*

### 2.4.2. Node.js

Node.js es un entorno de ejecución de JavaScript en el servidor. Ha servido para construir la aplicación de forma eficiente y escalable, usando el mismo lenguaje que en el frontend.


*Ilustración 5. Node.js*

### 2.4.3. Express

Express es un framework flexible para aplicaciones web basado en Node.js, diseñado para construir servidores HTTP de forma sencilla. En este proyecto ha servido para gestionar rutas, peticiones, respuestas, middleware y otra cantidad de servicios que ofrece para el desarrollo del backend de la aplicación. También facilita la conexión con Sequelize y la base de datos PostgreSQL.


*Ilustración 6. Express*

### 2.4.4. Sequelize

Sequelize es un ORM (Object Relational Mapper) que permite interactuar con la base de datos SQL usando código JavaScript. Hace más fácil definir modelos, relaciones y ejecutar consultas complejas sin utilizar directamente SQL.


*Ilustración 7. Sequelize*

### 2.4.5. Fetch API

Permite realizar peticiones HTTP de forma sencilla. En este proyecto se utiliza para la comunicación entre el frontend y el backend.


*Ilustración 8. Fetch API*

### 2.4.6. React Router DOM

Esta librería permite gestionar la navegación dentro de un aplicación React, como es esta, manteniendo la URL sincronizada en todos los componentes.


*Ilustración 9. React Router DOM*

### 2.4.7. Bootstrap

Bootstrap es un framework que proporciona estilos predefinidos. Ha sigo utilizado para asegurar una interfaz atractiva y fiable.


*Ilustración 10. Bootstrap*

### 2.4.8. Chart.js

Es una librería JavaScript que permite crear gráficos interactivos y visuales con facilidad. Se ha utilizado para representar datos de forma visual, como estadísticas de progreso.


*Ilustración 11. Chart.js*

### 2.4.9. React Feather

Es una colección de iconos minimalistas construidos como componentes de React. Se ha utilizado para mejorar la apariencia visual de botones e interfaces.

*Ilustración 12. React Feather*

### 2.4.10. React Beautiful DnD

Es una librería que permite crear interfaces con elementos arrastrables (Drag and Drop) de forma intuitiva. En este proyecto se ha empleado para ordenar objetivos visualmente mediante arrastrado.

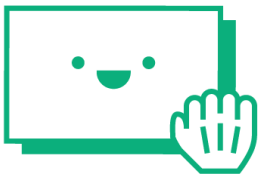


*Ilustración 13. React Beautiful DnD*

## 2.5. Seguridad

### 2.5.1. JSON Web Token

JSON Web Token es una tecnología utilizada para la autenticación segura entre cliente y servidor. Nos permite la generación de un token codificado que contiene información del usuario y que es verificado en cada petición protegida. Se ha utilizado para mantener sesiones seguras dentro de la aplicación.


*Ilustración 14. JSON Web Token*

## 3.1. Modelo Cliente-Servidor: Fundamentos y aplicación en el proyecto

El presente proyecto ha sido desarrollado bajo el paradigma clásico de arquitectura cliente-servidor. Esta arquitectura es de las más comunes para el desarrollo web, ya que establece una separación clara de responsabilidades y componentes en el sistema, lo que favorece la estructura del sistema. El cliente (frontend) se encarga de la interfaz e interacción con el usuario y el servidor (backend) de la lógica de negocio y la persistencia de los datos.

## 3.2. Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo en el que múltiples clientes puede solicitar y recibir servicios desde un único servidor centralizado. Generalmente se sigue el protocolo HTTP y se basa la comunicación asíncrona de peticiones y respuestas.

El cliente realiza peticiones y el servidor las procesa. El servidor puede acceder a la base de datos si fuera necesario.

Este enfoque favorece la escalabilidad y la separación de responsabilidades.

## 3.3. Aplicación práctica en el proyecto

En este proyecto, se ha utilizado esta arquitectura de la siguiente manera:

### 3.3.1. Cliente (frontend)

El cliente ha sido construido con React, una librería orientada a la creación de interfaces de usuario. En esta capa se pueden realizar los siguientes puntos:

- Autenticarse mediante formularios.
- Crear, modificar, visualizar y eliminar objetos.
- Ver estadísticas mediante gráficos.
- Interactuar con componentes dinámicos como drag and drop.

Todas estas acciones son en el fondo peticiones HTTP al servidor, enviadas mediante Fetch API. Esta capa no contiene lógica de negocio ni se conecta con la base de datos. La función principal sería recoger las entradas del usuario y mostrar lo que el servidor envíe en forma de respuesta.

### 3.3.2. Servidor (Backend)

El backend ha sido desarrollado utilizando Node.js y Express, una combinación común para la gestión de servidores web en JavaScript. Esta capa realiza las siguientes funciones de forma resumida:

- Validar y autentica las credenciales del usuario.
- Recibir y procesar las solicitudes enviadas desde el frontend.
- Manejar la base de datos en función de las peticiones recibidas.
- Aplicar la lógica de negocio (Cambios de estado, creación, eliminación, ...)

El servidor se comporta como un intermediario recibiendo los datos en formato JSON, procesándolos, accediendo a la base de datos, y devolviendo una respuesta también en JSON al cliente.

### 3.3.3. Base de datos (Persistencia)

La base de datos utilizada ha sido PostgreSQL, una solución muy recurrente para proyectos similares. La estructura de la base de datos está formada por diferentes tablas interconectadas entre sí:

- Usuario
- Meta
- Objetivo

Cada entidad está normalizada y vinculada, por ejemplo, un usuario tiene muchas alertas y cada meta tiene muchos objetivos. Esta organización permite una gestión de datos coherente y evita la duplicidad.

## 3.4. Ventajas

Esta arquitectura ofrece una variedad de ventajas para este proyecto que hacen que su elección cobre sentido.

- **Escalabilidad**: Permite aumentar la carga de datos sin comprometer al rendimiento, ya que cada parte del proyecto puede escalar de forma independiente.
- **Mantenimiento y evolución:** Separar cliente y servidor facilita la posibilidad de trabajar en las funcionalidades de cada lado sin necesidad de depender del otro.
- **Seguridad:** Se emplea una autenticación con JWT cuyo token se almacena en las cookies en el lado del cliente, luego cada vez que se intenta comprobar la validez de este se realiza desde el backend y esto reduce considerablemente el riesgo de vulnerabilidades.
- **Desarrollo paralelo:** Si bien para este proyecto no es una ventaja determinante, es importante tener en cuenta que lo normal es trabajar en equipo y separar las responsabilidades es importante y necesario para este tipo de desarrollos.

## 3.5. Conclusión

El modelo cliente-servidor se adapta perfectamente a los objetivos del proyecto, permitiendo una experiencia de usuario dinámica en el cliente y al mismo tiempo garantiza la seguridad y escalabilidad en el servidor.

En este capítulo se describirán todas las funcionalidades que el usuario puede realizar en la aplicación.

## 4.1. Funcionalidades del usuario

**FUN-1**. El usuario podrá iniciar sesión en la aplicación web ingresando su nombre de usuario y contraseña.

**FUN-2.** El usuario podrá registrarse en la aplicación proporcionando un nombre, apellido, nombre de usuario y contraseña válidos.

**FUN-3.** El usuario podrá ver el perfil de su cuenta, incluyendo su nombre, apellidos y nombre de usuario.

**FUN-4.** El usuario podrá modificar los datos de su cuenta, como nombre, apellidos, nombre de usuario y contraseña.

**FUN-5.** El usuario podrá cambiar su contraseña, siempre y cuando ingrese su contraseña actual correctamente.

**FUN-6.** El usuario podrá eliminar su cuenta, validando primero su contraseña para confirmar la acción. Esta acción eliminará también todas sus metas y objetivos asociados.

## 4.2. Funcionalidades del usuario en relación con las metas

**FUN-7.** El usuario podrá crear una nueva meta, indicando su título, descripción y, opcionalmente, una lista de objetivos asociados.

**FUN-8.** Al crear una meta sin objetivos definidos, la aplicación generará automáticamente un objetivo con el mismo título y descripción.

**FUN-9.** El usuario podrá ver el listado completo de sus metas, incluyendo los objetivos asociados a cada una.

**FUN-10.** El usuario podrá consultar el detalle de una meta específica, con toda su información y los objetivos relacionados.

**FUN-11.** El usuario podrá modificar una meta existente, incluyendo cambios en su título, descripción y objetivos relacionados (agregar, editar o eliminar).

**FUN-12.** El usuario podrá eliminar una meta, y con ella se eliminarán automáticamente todos los objetivos asociados.

**FUN-13.** El usuario podrá cambiar el estado de una meta alternando entre "pendiente" y "completada". El estado de la meta cambia automáticamente de pendiente a completada en cuanto el usuario haya completado todos los objetivos de la meta en cuestión.

**FUN-14.** El usuario podrá consultar sus metas junto con un resumen del estado de sus objetivos mediante gráficas, diferenciando entre los completados y los pendientes.

**FUN-15.** El usuario podrá ordenar sus metas por título y fecha de creación, manteniendo organizados los objetivos dentro de cada meta con un orden personalizable.

## 4.3. Funcionalidades del usuario en relación con los objetivos

**FUN-16.** El usuario podrá crear un objetivo para una meta específica, proporcionando un título, descripción y estado inicial.

**FUN-17.** El usuario podrá modificar un objetivo existente, permitiéndole cambiar su título, descripción y estado (pendiente, en progreso o completado).

**FUN-18.** El usuario podrá eliminar un objetivo, si ya no lo necesita o si ha sido completado.

**FUN-19.** Si un objetivo es marcado como completado, se actualizará la fecha de finalización.

**FUN-20.** El usuario podrá consultar los objetivos asociados a una meta específica.

**FUN-21.** El usuario podrá consultar en el apartado de agenda aquellos objetivos que se encuentren en estado en progreso.

**FUN-22.** El usuario podrá comprobar la cantidad de objetivos completados por mes en los últimos 6 meses.

**FUN-23.** El usuario podrá consultar un resumen general del progreso de sus objetivos, mostrando un conteo de los objetivos completados y pendientes de todos los objetivos asociados a su cuenta.

**FUN-24.** El usuario podrá reordenar los objetivos “en progreso” según un orden personalizado, de manera que los más importantes o prioritarios se muevan al principio de la lista agenda.

## 5.1. Introducción

La base de datos es fundamental dentro del desarrollo de cualquier aplicación web. Es la encargada de almacenar de manera persistente toda la información necesaria para el correcto funcionamiento del sistema. La persistencia de datos permite que la información se mantenga disponible incluso si los servicios están apagados o el servidor esté reiniciándose.

Un diseño adecuado de la base de datos facilita la escalabilidad, la eficiencia en las consultas y la seguridad de la información, todos ellos pilares esenciales de cualquier proyecto.

## 5.2. Justificación del uso de PostgreSQL

Para este proyecto se ha optado por PostgreSQL como sistema de gestión de bases de datos (SGBD) por las siguientes razones:

- **Fiabilidad y robustez:** PostgreSQL es conocido por su alta estabilidad y consistencia en el manejo de datos, incluso en sistemas complejos y de gran volumen.
- **Soporte de tipos de datos avanzados:** Permite el manejo de tipos de datos personalizados, arrays y JSON, ofreciendo una gran flexibilidad para distintos escenarios de desarrollo.
- **Cumplimiento de estándares:** Es uno de SGBD más fieles a los estándares SQL y esto facilita su portabilidad y el mantenimiento del proyecto.
- **Open Source:** Al ser gratuito y de código abierto, se puede adaptar a las necesidades del proyecto sin costes adicionales.
- **Seguridad:** Incluye mecanismos de autenticación y autorización avanzados, lo que mejora la protección de los datos almacenados.
- **Facilidad de integración:** Se integra perfectamente con herramientas como Sequelize (ORM utilizado en el backend del proyecto) y Node.js.

Gracias a estas características, PostgreSQL ha sido la elección ideal para garantizar una gestión de datos segura, eficiente y escalable del proyecto.

## 5.3. Descripción de la base de datos del proyecto

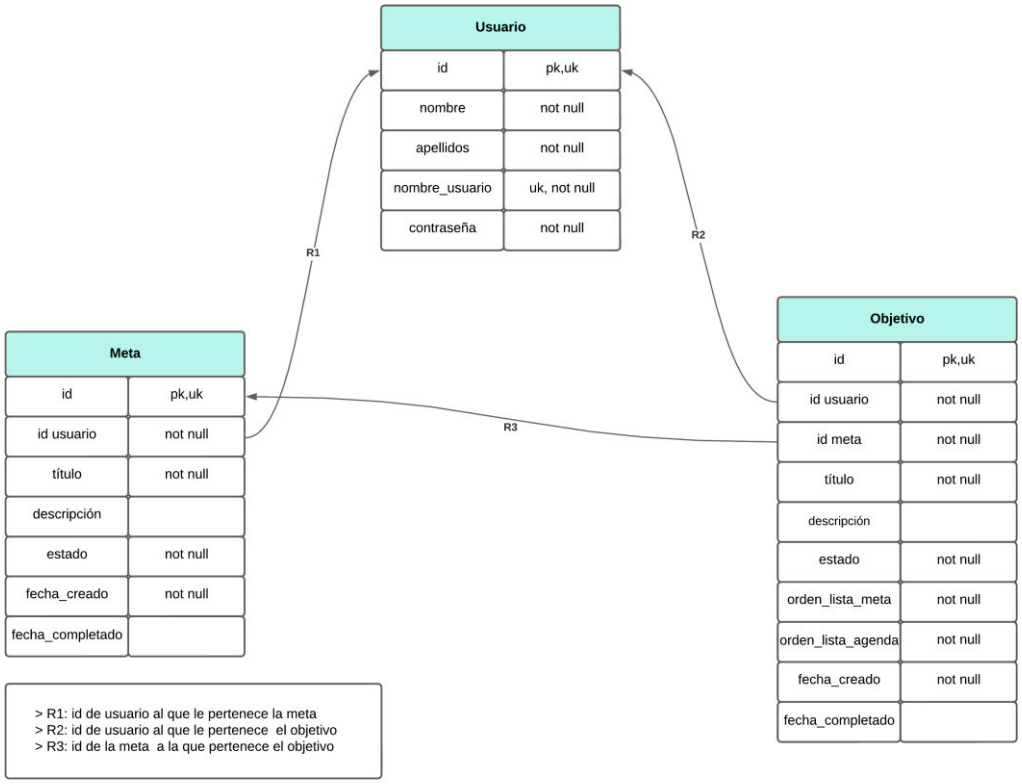



*Ilustración 15. Base de datos*

La base de datos diseñada para este proyecto sigue un modelo relacional clásico, como se muestra en el diagrama anterior. Se estructura en tres entidades principales: Usuario, Meta y Objetivo.

- **Usuario**
  Representa a los usuarios registrados en el sistema. Cada usuario tiene atributos como nombre, apellidos, nombre de usuario (único) y contraseña. Se asegura la unicidad mediante claves primarias y restricciones.

- **Meta**
  Cada usuario puede tener varias metas, que representan grandes metas que el usuario quiere alcanzar. Cada meta está ligada a un

usuario a través de una relación de clave foránea (id_usuario). Además, se almacena información relevante como título, descripción y estado de la meta (pendiente o completada), y fechas de creación y finalización.

- **Objetivo**
  Los objetivos son tareas o pasos concretos que componen una meta. Cada objetivo pertenece a un usuario y a una meta concreta, lo que se gestiona mediante claves foráneas (id_usuario, id_meta). También incluyen atributos como título, descripción, estado (pendiente, en progreso y completada), orden dentro de la meta y dentro de la agenda, así como fechas de creación y de finalización.

Las relaciones entre las tablas están definidas de la siguiente manera:

- **R1**: Un usuario puede tener muchas metas.
- **R2**: Un usuario puede tener muchos objetivos.
- **R3:** Una meta puede tener muchos objetivos asociados.

El backend de la aplicación se ha diseñado siguiendo una arquitectura modular, donde cada componente tiene una responsabilidad clara y diferenciada. Esta separación facilita el mantenimiento, mejora la escalabilidad y favorece el desarrollo de nuevas funcionalidades de manera organizada y eficiente. Para ello, se han utilizado principalmente Modelos, Servicios, Rutas y un Middleware, cada un cumpliendo un papel específico en la estructura del sistema.

- **Modelos**

Los modelos representan la estructura de los datos en la base de datos. Cada modelo define los atributos de su respectiva entidad (Usuario, Meta y Objetivo), sus tipos de datos, restricciones y relaciones. Se ha utilizado Sequelize, un ORM de Node.js, para definir estos modelos de forma más flexible y mantener la base de datos sincronizada con el código de la aplicación.

- **Servicios**

Los servicios contienen la lógica de negocio de la aplicación. Son funciones que agrupan todas las operaciones que se pueden realizar sobre una entidad concreta, como crear, leer, actualizar o eliminar la información (CRUD).

Esto permite que el código esté mejor organizado, separando claramente la lógica de los datos del resto de la aplicación, y facilita la reutilización de funciones en diferentes rutas o controladores.

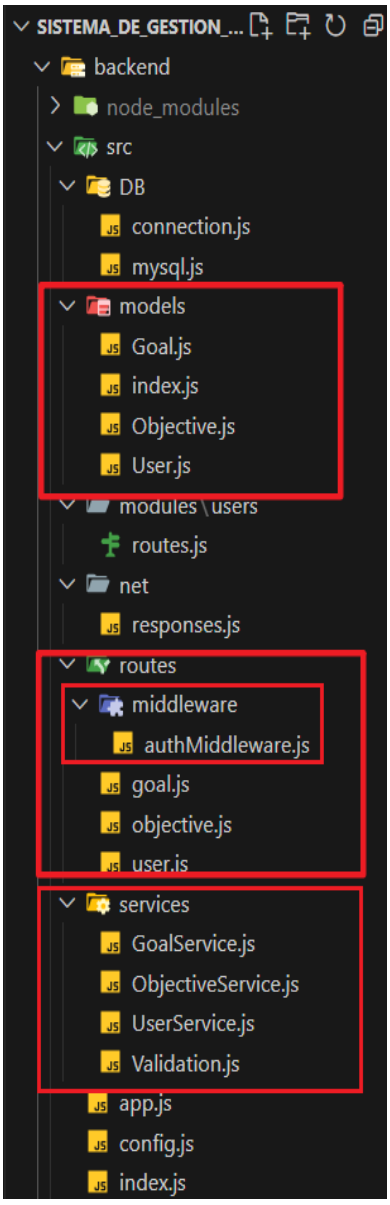



*Ilustración 16. Estructura Backend*

- **Rutas**

Las rutas son los puntos de entrada al backend. Cada ruta expone el conjunto de endpoints (URLs) que permiten al frontend comunicarse con el servidor. A través de estas rutas, el cliente puede solicitar operaciones específicas, como registrar un nuevo usuario, crear una meta, actualizar un objetivo o eliminar información, entre otros.

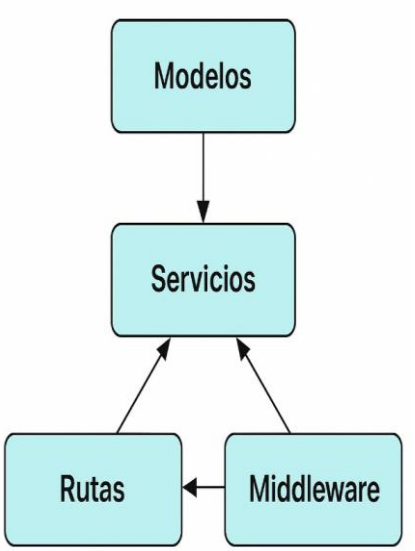



*Ilustración 17. Relación elementos Backend*

- **Middleware**

El middleware actúa como una capa intermedia entre la solicitud del cliente y la respuesta del servidor. En este proyecto esta capa se encuentra en el servidor, pero no deja pasar la petición hasta que se haya comprobado que proviene de un usuario autorizado. Es decir, se utiliza principalmente para gestionar la seguridad mediante la validación de JSON Web Tokens (JWT).

Gracias a este middleware, se garantiza que solo los usuarios autenticados puedan acceder o modificar información sensible del sistema.

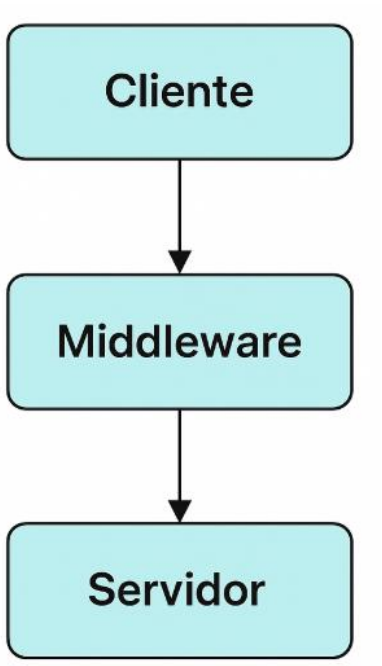



*Ilustración 18. Middleware en relación Cliente-Servidor*

## 6.1. Modelos

En este apartado se explicarán los modelos utilizados en el desarrollo del proyecto. Cada modelo representa una tabla en la base de datos PostgreSQL y establece relaciones claras entre ellas.

### 6.1.1. Modelo User

El modelo User representa a los usuarios de la aplicación.

- ***Identificación***

Cada usuario es identificado mediante un UUID, garantizando la unicidad de usuarios en la aplicación.

- ***Campos principales***

Username, name, lastName y password, todos ellos son obligatorios.

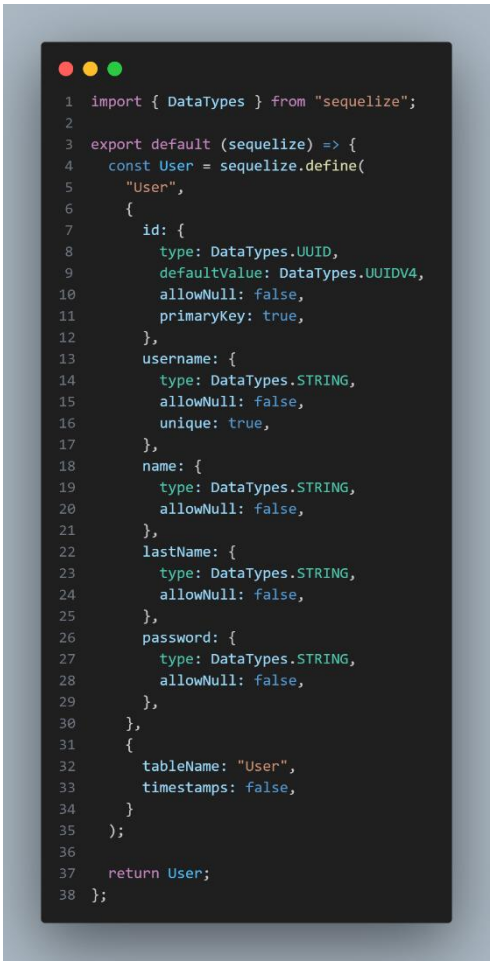
- ***Restricciones***

El campo user username debe ser único para evitar duplicados en el registro.

- ***Notas***

No se utilizan marcas de tiempo automáticas (timestamps: false).

Se utiliza UUIDV4 como valor predeterminado para el identificador id. Este tipo de dato se genera de forma única un valor alfanumérico.



```
import { DataTypes } from "sequelize";

export default (sequelize) => {
  const User = sequelize.define(
    "User",
    {
      id: {
        type: DataTypes.UUID,
        defaultValue: DataTypes.UUIDV4,
        allowNull: false,
        primaryKey: true,
      },
      username: {
        type: DataTypes.STRING,
        allowNull: false,
        unique: true,
      },
      name: {
        type: DataTypes.STRING,
        allowNull: false,
      },
      lastName: {
        type: DataTypes.STRING,
        allowNull: false,
      },
      password: {
        type: DataTypes.STRING,
        allowNull: false,
      },
    },
    {
      tableName: "User",
      timestamps: false,
    }
  );

  return User;
};
```

*Ilustración 19. Modelo User*

El modelo es fundamental para la autenticación y gestión de permisos en toda la aplicación.

### 6.1.2. Modelo Goal

El modelo Goal contiene las metas creadas por los usuarios.

- ***Relación***

Cada meta pertenece a un único usuario (userId) mediante una clave foránea con borrado en cascada (onDelete: CASCADE).

- ***Campos principales***

Title y descripción describen la meta de forma textual.

State gestiona el estado de la meta, utilizando valores restringidos (pending, inprogress, completed).

CreatedAt registra la fecha de creación y completedAt permite saber cuando fue comletada la meta.

- ***Notas***

El control de estados facilita el seguimiento del progreso de cada meta dentro de la aplicación.

```
import { DataTypes } from "sequelize";

export default (sequelize) => {
  const Goal = sequelize.define(
    "Goal",
    {
      id: {
        type: DataTypes.INTEGER,
        autoIncrement: true,
        allowNull: false,
        primaryKey: true,
      },
      userId: {
        type: DataTypes.UUID,
        allowNull: false,
        field: "userId",
        references: {
          model: "User",
          key: "id",
        },
        onDelete: "CASCADE",
      },
      title: {
        type: DataTypes.STRING,
        allowNull: false,
      },
      description: {
        type: DataTypes.STRING,
        allowNull: true,
      },
      state: {
        type: DataTypes.ENUM("pending", "inprogress", "completed"),
        allowNull: false,
        defaultValue: "pending",
      },
      createdAt: {
        type: DataTypes.DATE,
        allowNull: false,
        defaultValue: DataTypes.NOW,
      },
      completedAt: {
        type: DataTypes.DATE,
        allowNull: true,
      },
    },
    {
      tableName: "Goal",
      timestamps: false,
    }
  );

  return Goal;
};
```

*Ilustración 20. Modelo Goal*

Este modelo organiza y agrupa los objetivos individuales, modelados por Objective, bajo una misma finalidad.

Dentro del proyecto se utilizarán solo los estados pending y completed para Meta, ya que depende de los objetivos. Estos últimos si utilizarán los tres estados.

### 6.1.3. Modelo Objective

El modelo Objective contiene los objetivos del usuario que pertenecen a una meta previamente definida.

- ***Relaciones***

Cada objetivo pertenece a un usuario (userId) y a una meta específica (goalId), ambos como claves foráneas.

Se configuran eliminaciones en cascada también para mantener la integridad de la base de datos.

- ***Campos principales***

Title y descripción describen al objetivo igual que sucedía con las metas.

State gestiona el estado actual (pending, inprogress, completed).

GoalListOrder establece el orden de los objetivos dentro de la lista de la meta a la que pertenecen.

AgendaListOrder establece el orden en el que el usuario ha decidido priorizar en la vista de la agenda. Es opcional ya que hasta que no se encuentra en la agenda y por tanto no tiene orden en un principio (allowNull: true).

CreatedAt y completedAt registran la fecha de creación y completado, respectivamente.

```
import { DataTypes } from "sequelize";

export default (sequelize) => {
  const Objective = sequelize.define(
    "Objective",
    {
      id: {
        type: DataTypes.INTEGER,
        autoIncrement: true,
        allowNull: false,
        primaryKey: true,
      },
      userId: {
        type: DataTypes.UUID,
        allowNull: false,
        field: "userId",
        references: {
          model: "User",
          key: "id",
        },
        onDelete: "CASCADE",
      },
      goalId: {
        type: DataTypes.INTEGER,
        allowNull: false,
        field: "goalId",
        references: {
          model: "Goal",
          key: "id",
        },
        onDelete: "CASCADE",
      },
      title: {
        type: DataTypes.STRING,
        allowNull: false,
      },
      description: {
        type: DataTypes.STRING,
        allowNull: true,
      },
      state: {
        type: DataTypes.ENUM("pending", "inprogress", "completed"),
        allowNull: false,
        defaultValue: "pending",
      },
      goalListOrder: {
        type: DataTypes.INTEGER,
        allowNull: false,
      },
      agendaListOrder: {
        type: DataTypes.INTEGER,
        allowNull: true,
      },
      createdAt: {
        type: DataTypes.DATE,
        allowNull: false,
        defaultValue: DataTypes.NOW,
      },
      completedAt: {
        type: DataTypes.DATE,
        allowNull: true,
      },
    },
    {
      tableName: "Objective",
      timestamps: false,
    }
  );

  return Objective;
};

```

*Ilustración 21. Modelo Objective*

- ***Notas***

Es uno de los modelos más dinámicos ya que admite reordenamientos y mútiples visualizaciones.

El modelo Objective es clave para subdividir una meta en tareas manejables que pueden ser seguidas y completadas progresivamente.

## 6.2. Servicios

### 6.2.1. Clase UserService – Gestión de usuarios

Esta clase encapsula toda la lógica relacionada con la gestión de usuarios del sistema. Sus funcionalidades cubren tanto el ciclo de vida de los usuarios como las operaciones necesarias para la autenticación, modificación y eliminación segura de cuentas.

**Funcionalidades principales:**

- Creación de usuario (**createUser**):
  Permite registrar nuevos usuarios, validando sus datos (nombre, apellidos, nombre de usuario y contraseña), encriptando la contraseña y asegurando la unicidad del nombre de usuario.

- Inicio de sesión (**login**):
  Autentica a un usuario comparando las credenciales proporcionadas (nombre de usuario y contraseña) con los datos almacenados en la base de datos.

- Obtención de todos los usuarios (**getAllUsers**):
  Devuelve una lista de todos los usuarios registrados en el sistema.

- Obtención de un usuario específico (**getUser**):
  Permite consultar la información de un usuario a través de su ID.

- Modificación de un usuario (**modifyUser**):
  Permite modificar los datos personales del usuario. Si se modifica la contraseña, se vuelve a validar y encriptar.

- Eliminación de un usuario (**deleteUser**):
  Permite eliminar un usuario del sistema de forma individual, sin necesidad de contraseña.

- Perfil del usuario (**getUserProfile**):
  Devuelve los datos básicos del perfil del usuario (nombre, apellidos y nombre de usuario) con la finalidad de ser visualizado por el front.

- Cambio de contraseña (**changePassword**):
  Permite al usuario cambiar su contraseña tras verificar la contraseña actual.

- Eliminación completa y segura de cuenta (**deleteUserWithPassword**):
  Verifica la contraseña antes de eliminar el usuario y todos los datos asociados (metas y objetivos), garantizando una eliminación completa.

### 6.2.2. Clase GoalService – Gestión de metas

Esta clase se encarga de toda la lógica funcional relaciona con las metas del usuario. Las metas representan grandes objetivos (Metas) que contienen sub-objetivos o tareas (Objetivos).

**Funcionalidades principales:**

- Creación de meta (**createGoal**):
  Crear una nueva meta, con o sin objetivos asociados. Si no se proporcionan objetivos, se genera un automáticamente con el mismo título y descripción.

- Eliminación de meta (**deleteGoal**):
  Elimina una meta específica a partir de su ID.

- Modificación de meta (**modifyGoal**):
  Permite actualizar los campos de una meta existente.

- Cambio de estado de la meta (**toggleGoalState**):
  Cambia el estado de una meta entre pending, inprogress y completed.

- Obtención de metas por usuario (**getUserGoals**):
  Devuelve todas las metas de un usuario, incluyendo sus objetivos asociados si existen.

- Obtención de metas en progreso (**getGoalsWithCompletionData**):
  Devuelve toda la información de cada meta con sus objetivos completados y pendientes, ordenados por título.

- Obtención de metas ordenadas (**getSortedGoals**):
  Devuelve todas las metas del usuario y sus objetivos ordenador por un criterio específico, que puede ser por título o descripción.

- Obtención de metas con objetivos (**getGoalsWithObjectives**):
  Devuelve todas las metas del usuario junto con los detalles de sus objetivos ordenados por goalListOrder.

- Modificación de metas con sus objetivos (**modifyGoalWithObjectives**):
  Permite actualizar tanto la meta como sus objetivos. Se pueden añadir, editar o eliminar objetivos en función de los cambios enviados.

- Consulta de una meta específica (**getGoalById**):
  Devuelve todos los datos de una meta concreta, incluyendo la lista ordenada de objetivos asociados.

### 6.2.3. Clase ObjectiveService – Gestión de objetivos

Esta clase contiene todas las funcionalidades relacionadas con los objetivos, que son pequeños objetivos individuales dentro de una meta. Permite la gestión completa de su ciclo de vida, estados y orden dentro de listas.

**Funcionalidades principales:**

- Creación de objetivo (**createObjective**):
  Permite crear un nuevo objetivo asociado a una meta y a un usuario.

- Actualización de objetivo (**updateObjective**):
  Permite modificar los atributos de un objetivo específico.

- Eliminación de objetivo (**deleteObjective**):
  Elimina un objetivo específico a través de su ID.

- Cambio de estado del objetivo (**toggleObjectiveState**):
  Actualiza el estado del objetivo a pending, inprogress o completed, aplicando algunas lógicas adicionales como:
  - ✓ Guarda la fecha de finalización si es completado.
  - ✓ Elimina y asigna un orden de la agenda si cambia de estado a inprogress.
  - ✓ Si todos los objetivos de una meta están completados, marca también la meta como completada.

- Reordenación de la agenda (**reorderAgendaListForUser**):
  Reasigna el orden (agendaListOrder) de los objetivos en progreso según su posición actual enviada desde el frontend.

- Obtención de objetivos por meta (**getObjectivesByGoalId**):
  Devuelve todos los objetivos asociados a una meta específica.

- Filtrado por estado (**getObjectivesByState**):
  Devuelve todos los objetivos del usuario con un estado específico (pending, inprogress, completed).

- Objetivos completados por mes (**getCompletedObjectivesByMonth**):
  Devuelve una estadística mensual de los objetivos completados por mes durante los últimos siete meses, útil para las visualizaciones en el lado del cliente.

- Progreso general del usuario (**getGeneralObjectiveCompletion**):
  Devuelve el número total de objetivos completados y no completados del usuario, permitiendo calcular su avance.

- Actualización del orden de la agenda (**updateAgendaOrder**):
  Permite al usuario reordenar sus objetivos en progreso de forma manual, siempre y cuando estos objetivos estén en progreso en la lista agenda.

## 6.3. Rutas

### 6.3.1. Rutas de Goal (/goals)

Estas rutas permiten la gestión de metas (Goals), incluyendo su creación, edición eliminación y consulta:

#### POST /create

Crea una nueva meta.

#### GET /:id

Obtiene una meta específica por su ID junto con todos sus objetivos asociados.

#### DELETE /:id

Elimina una meta específica.

#### PATCH /:id

Modifica una meta existente con los datos enviados en el body de la petición.

#### PATCH /:id/toggle

Cambia el estado de una meta (ej. Pending a inprogress o completed).

#### GET /user

Recupera todas las metas asociadas al usuario autenticado.

#### GET /user/goals-completion

Obtiene estadísticas de cumplimiento de metas y objetivos del usuario.

#### GET /user/sorted-goals

Recupera las metas del usuario, ordenadas por un campo específico (sortBy) y en orden ascendente o descendente (order).

#### POST /objective/create-multiple

Permite la creación masiva de objetivos relacionados con metas.

#### GET /user/goal-with-objectives

Devuelve todas las metas del usuario junto a sus objetivos.

#### PATCH /:id/update

Actualiza tnato la meta como sus objetivos asociados.

### DELETE /:id/delete

Elimina una meta junto con todos sus objetivos relacionados.

### 6.3.2. Rutas de Objective (/objectives)

Estas rutas permiten la gestión de los objetivos:

### POST /create

Crea un nuevo objetivo asociado a una meta.

### GET /completed-per-month

Obtiene estadísticas de cuántos objetivos fueron completados por mes.

### DELETE /:id

Elimina un objetivo específico por su ID.

### PATCH /:id

Modifica una meta existente con los datos enviados en el body de la petición.

### PUT /:id

Modifica un objetivo existente (datos completos).

### PATCH /:id/toggle

Permite cambiar el estado a un objetivo por su id a cualquiera de los estados (pending, inprogress, completed).

### GET /user

Devuelve todos los objetivos del usuario autenticado.

### GET /user/state/:state

Devuelve los objetivos del usuario de un estado en concreto.

### GET /general-completion

Devuelve la cantidad de objetivos completados (completed) y no completados (inprogress y pending) del usuario.

### PATCH /user/update-agenda_order

Modifica el orden de los objetivos según la petición del usuario.

### 6.3.3. Rutas de User (/users)

En este apartado se describirán las rutas que gestionan las operaciones de los usuarios.

#### GET /

Devuelve la lista de todos los usuarios.

#### PATCH /

Permite modificar el perfil del usuario autenticado.

#### POST /login

Permite iniciar sesión. Tras iniciar sesión, devuelve el token JWT que después se guardará en las cookies para mantener la sesión. Las sesiones son de una hora.

#### POST /logout

Cierra la sesión eliminando la cookie de autenticación.

#### GET /profile

Devuelve los datos del perfil del usuario autenticado.

#### POST /register

Permite registrar un usuario y devuelve el token JWT automáticamente.

#### PATCH /change-password

Permite cambiar la contraseña del usuario autenticado.

#### DELETE /

Eliminar la cuenta del usuario autenticado (requiere confirmar la contraseña).

#### GET /validate-token

Verifica si el token de autenticación es válido y devuelve los decodificados del usuario.

## 6.4. Middleware

La función authenticateToken es un middleware de Expresss que hace lo siguiente:

- Valida que el usuario tenga una cookie llamada “access-token” en su solicitud.
- Verifica si el token JWT es válido utilizando la clave secreta (SECRET_JWT_KEY) almacenada en las variables de entorno.
- Si el token es válido, agrega la información del usuario al objeto req.user, permitiendo que los siguientes controladores accedan a los datos del usuario autenticado.
- Si no hay token o el token es inválido/expirado, respondee inmediatamente con un error, impiendo el acceso a las rutas protegidas.

Por otro lado, este middleware requiere que Express esté configurado para leer cookies, usando cookie-parser.

```
1  import jwt from "jsonwebtoken";
2
3  export const authenticateToken = (req, res, next) => {
4    const token = req.cookies["access-token"];
5    if (!token) {
6      return res.status(401).json({ error: "No autorizado, token requerido" });
7    }
8
9    jwt.verify(token, process.env.SECRET_JWT_KEY, (err, user) => {
10     if (err) {
11       return res.status(403).json({ error: "Token inválido o expirado" });
12     }
13
14     req.user = user;
15     next();
16   });
17 };
```

*Ilustración 22. Middleware*

El frontend de la aplicación está construido en React, una biblioteca de JavaScript moderna. Esta sección del documento describe los distintos componentes que forman la interfaz de usuario, explicando, entre otras cosas, su propósito, su lógica interna más relevante y las decisiones técnicas que se han tomado para mejorar la experiencia de usuario.

Cada componente tiene una función específica dentro del flujo de la aplicación, y muchos de ellos se apoyan en estados locales y navegación en función de la entrada del usuario.

## 7.1. Componentes

### 7.1.1. Login

El componente Login es la puerta de entrada a la aplicación. Su función principal es autenticar a los usuarios existentes a través de un formulario de nombre de usuario y contraseña. Si la autenticación es exitosa, redirige automáticamente al usuario a la página de inicio y, si falla, informa con una animación visual del error.

- **Autenticación**

Para la autenticación de usuarios se llama a la función login(username, password) de forma asíncrona.

Si la respuesta es correcta, es decir, tiene el status == 200, se llevará al usuario a la página de inicio de la aplicación (/inicio).

Si falla, se muestra un mensaje de error con una animación tipo “shake”.

- **Animaciones para la mejora de experiencia de usuario**

El formulario del login contiene los inputs de username y password, y se mostrarán conforme se vayan rellenando. Además, el botón de iniciar sesión aparece sólo cuando los campos tienen contenido o no hay un error de validación.

La animación “shake” aparecerá en rojo cuando el usuario no introduzca bien sus credenciales y dejará los campos del formulario de entrada estáticos.

- **Tema oscuro por defecto**

Al montar el componente, se añade el tema oscuro de la clase dark, usando useEffect.

Al desmontarse, se limpia esta clase ya que el resto de la aplicación no tiene es tema. Esto permite a otras vistas definir su propio estilo sin interferencias.

- **Soporte para la tecla Enter**

El formulario esta diseñado para esperar activamente entradas del teclado. Si se pulsa esta tecla, se hace un intento de acceso a la aplicación, lo que mejora la usabilidad.

- **Separación lógica**

La lógica del login se encuentra fuera del componente, ya que se trata de gestionar los accesos. Favorece la seguridad y facilita la lectura del código.

### 7.1.2. Register

El componente Register permite a los nuevos usuarios crear un cuenta en la aplicación mediante un formulario que solicita los datos personales (nombre y apellidos) y las credenciales (username y password) que usará cuando quiera acceder a la aplicación.

- **Validación local de la contraseña**

Antes de enviar los datos al servidor, se hace una verificación de que ambas coincidan. Si no coincide se detiene el procesos y se lanza una alerta inmediata al usuario.

- **Conexión con la API**

Se realiza una llamada Fetch a http://localhost:4000/api/user/register, enviando los datos del formulario como JSON.

Si la respuesta es exitosa, se informa al usuario y se le redirige directamente al inicio.

En caso de error (username existente, por ejemplo), se le informa al usuario sobre qué es lo que tiene que cambiar.

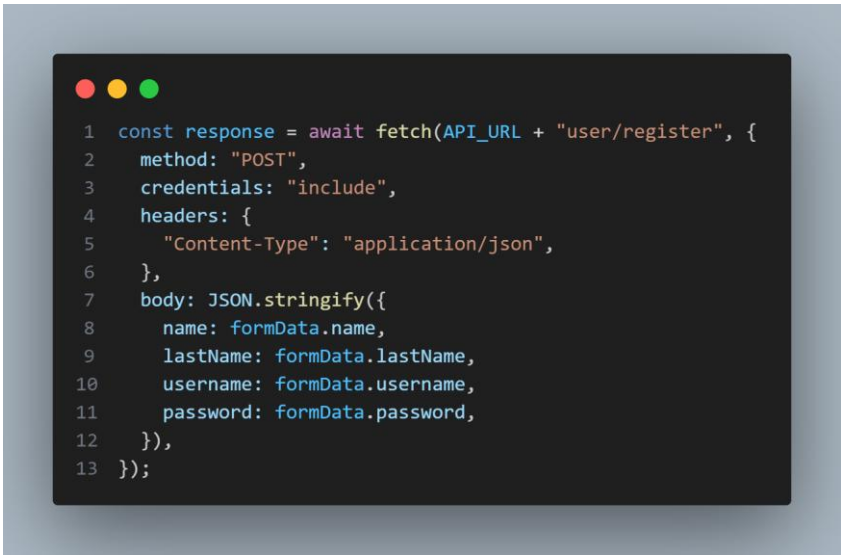



*Ilustración 23. Petición Fetch para el registro*

Si la respuesta es exitosa, se informa al usuario y se le redirige directamente al inicio.

En caso de error (username existente, por ejemplo), se le informa al usuario sobre qué es lo que tiene que cambiar.

- **Navegación**

Para navegar hacia la página principal se utiliza el hook useNavigate() de React Routeer para redirigir al usuario. Será el que se use en toda la aplicación.

### 7.1.3. Navbar

El componente Navbar es la barra de navegación principal de la aplicación. Permite al usuario moverse entre las distintas secciones del sistema (Inicio, Agenda, Mis Metas y Mi Cuenta), además de la opción de cerrar sesión. Su objetivo es proporcionar al usuario la posibilidad de moverse fácilmente por toda la aplicación.

- **Componente NavItem**

Este componente permite personalizar los ítems que llevará la barra de navegación. Es reutilizable y de la que se puede configurar:

- Un icono
- Un texto
- Una ruta
- La ruta activa actual
- Una función onClick para navegar


*Ilustración 24. Componente NavItem*

- **Cierre sesión**

El botón de salir realiza un llamada POST al backend para cerrar la sesión del usuario.

La página te redirige inmediatamente al login, usando navigate con la opción replace para evitar que el usuario pueda volver a atrás con el botón del navegador.

### 7.1.4. Grafica

El componente Grafica ofrece una visualización estadística de los objetivos del usuario, proporcionando tres vistas diferentes.

- Gráfica de barras
    a. Esta gráfica muestra los objetivos completados por el usuario al mes durante los últimos 7 meses.
- Gráfica circular general
    a. Esta gráfica muestra los objetivos completados y no completados de manera general, incluyendo todas las metas.
- Gráfica circular por meta
    a. Esta gráfica muestra los objetivos completados y no completados por meta. El usuario puede elegir hasta un máximo de cuatro metas de todas las que tenga.

Este componente tiene la intención de proporcionar una vista rápida del progreso personal del usuario.

- **Gestión de pestañas**

El estado activeTab tiene como objetivos saber qué gráfica debe mostrarse en función de lo que el usuario seleccione.

Encontramos tres tipos de visualización:

- “bar”: muestra el gráfico de barras mensual.
- “doughnut”: gráfico circular de objetivos generales.
- “select”: gráficos personalizados para cada meta.

Estos nombres corresponden con las variables que almacenan las gráficas dentro del componente.

- **Obtención de datos desde la API**

Se hacen las siguientes llamadas importantes a los siguientes endpoints:

| Endpoint | Propósito |
|---|---|
| /objective/completed-per-month | Objetivos completados por mes. |
| /objective/general-completion | Estadísticas generales de todos los objetivos completados y no completados. |
| /goal/user/goals-completion | Metas, y sus objetivos completados y no completados. |

Estas peticiones se hacen dentro del useEffect y los resultados se almacenan en variables que posteriormente se usarán para mostrar en las gráficas correspondientes.

- **Renderización de gráficos con Chart.js**

A partir de los datos del apartado anterior, se muestran las gráficas tanto de barra como circulares.

Siempre que se quiere renderizar una gráfica, se destruye la anterior para despejar el componente y pase a ser ocupado por la siguiente gráfica correspondiente.

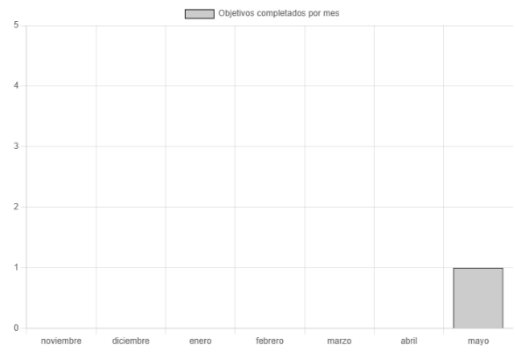



*Ilustración 25. Gráfica de tipo barras*

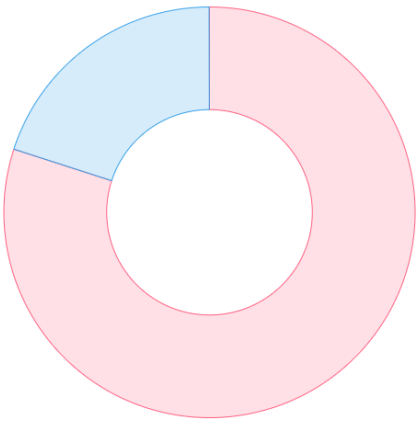


*Ilustración 26. Gráfica de tipo circular*

- **Selección de gráficas**

En el apartado de *Objetivos por meta* se permite al usuario seleccionar hasta un máximo de cuatro metras de entre todas las que tiene. El objetivo es mantener una página ordenada sin gran cantidad de elementos que entorpezcan la calidad visual.

Se mostrarán todas las metas del usuario con su título y un checkbox que permita seleccionar las cuatro que el usuario desee. Estas gráficas serán circulares y contendrán los objetivos completados y no completados para esa meta.

### 7.1.5. Inicio

El componente Inicio representa la página principal de la aplicación cuando el usuario termina de iniciar sesión o registrarse.

Esta sección es la que contiene el componente explicado en el apartado anterior *Gráfica*. En resumen, los elementos más importantes que este componente contiene:

- NavBar: la barra de navegación superior.
- Gráfica: componente principal con las distintas visualizaciones.
- useAuth: hook para validación y autenticación.

El componente cumple con la función de ser la página de inicio del usuario autenticado, enfocándose en mostrar un análisis visual gracias al componente Gráfica.

### 7.1.6. Agenda

Este componente contiene una lista de objetivos en progreso. Esta lista puede ser ordenada en prioridad según el usuario desee. Sirve como una agenda personal en la que los objetivos se pueden cambiar de orden con *drag and drop*.

Este componente también contiene el hook useAuth() que solo permite acceder a los usuarios que estén autenticados protegiendo así la ruta.

La conexión con la API se hace gracias al siguiente endpoint de objetivos:

*GET /objective/user/state/inprogress*

### 7.1.7. Lista Metas

El componente contiene las metas del usuario junto con sus objetivos.

Este componente tiene una serie de características que cumplen varias de las funcionalidades que el usuario puede hacer respecto a metas y objetivos:

- Permite buscar por título todas las metas del usuario
- Permite ordenar mediante filtros de ordenación tanto de nombre como de fecha de creación ascendente y descendentemente.
- Permite ver los detalles de cada meta haciendo click sobre el nombre de la meta.
- Permite crear metas mediante un botón.
- Permite editar metas mediante el lápiz que se encuentra en cada meta.
- Permite extender cada meta para consultar sus objetivos.
- Permite cambiar el estado a cada objetivo (pendiente, en progreso y completado).
- Permite consultar visualmente si una meta se ha completado (con un check) una vez todos sus objetivos se hayan completado.

Este componente se actualiza ante cada cambio, ya que hace una petición por cada acción que realiza el usuario. Esto permite tener fluidez en el flujo de la aplicación a coste de realizar una gran cantidad de peticiones cada vez que se realiza una acción.

### 7.1.8. Crear Meta

Este componente permite crear una meta a partir del apartado *Mis Metas*. Este componente contempla las siguientes acciones:

- Se puede crear una meta con título y descripción (opcional).
- Se puede crear objetivos para la meta.
- Se puede ordenar los objetivos en el orden en el que el usuario desee.
- Si el usuario no introduce un objetivo, se creará uno con el mismo nombre y descripción de la meta.

Entre las características técnicas más destacadas se encuentran:

- Gestiones de estados y cambios pendientes. Es decir, evita la pérdida accidental de datos.

- Se realizan validaciones antes de guardar. Por ejemplo, no se puede guardar una meta vacía. Básicamente antes de enviar la petición, se realiza una limpieza de los objetivos vacíos por el frontend.

### 7.1.9. Editar Meta

Este componente hereda del componente *Crear Meta* tanto en diseño como en estructura.

Tiene lógica adicional ya que debe hacer una petición previa para mostrar la información que se encuentra en ese momento en la base de datos acerca de la meta que el usuario haya seleccionado.

Entre las características más destacadas encontramos:

- Mediante una petición compara el contenido de la meta con el actual para saber qué debe modificar y saber cuál será el estado final.
- Se mantiene el estado de aquellos objetivos independientemente de sí se le ha modificado el orden, el nombre o la descripción.
- Existe la opción de eliminar la meta.

### 7.1.10. Detalle Meta

Este componente muestra los detalles de una meta de forma individual, incluyendo su título, descripción, estado, fecha de creación, y la lista de objetivos ordenados con sus respectivos estados y fechas de completado (si procede).

La dificultad lógica de este componente se ve resumida en una petición de los datos de una meta y en mostrarlo de una forma amigable al usuario.

### 7.1.11. Mi Cuenta

Este componente permite al usuario ver y editar su información de perfil (nombre, apellidos, nombre de usuario), cambiar su contraseña y eliminar su cuenta. Utiliza autenticación para restringir el acceso y hace uso de un modal para las acciones sensibles.

En cuanto a la lógica más compleja encontramos:

- Autenticación de usuario para acceder a la vista de datos sensibles del usuario.
- El componente puede estar en modo edición o no. Si el usuario desea editar su cuenta, podrá hacerlo mientras estos campos se encuentren disponibles para editar. Este modo edición se activa cuando el usuario selecciona *Editar cuenta*.
- El usuario puede cambiar de contraseña. Se le pedirá la contraseña actual.
- El usuario puede eliminar su cuenta. Se le pedirá la contraseña actual.

En este componente se reutiliza el modal para el cambio de contraseña y para la eliminación de la cuenta.

Ante cualquier error de validación se le informará al usuario mediante una alerta (contraseña actual incorrecta, por ejemplo).

### 7.1.12. Mis Metas

Este componente representa la vista principal donde se muestran todas las metas del usuario. Es un contenedor estructura que garantiza la autenticación, renderiza la barra de navegación y carga la lista de metas mediante un componente hijo.

En cuanto a la lógica principal encontramos:

- La protección de la ruta con useAuth().
- Una estructura que solo renderiza componentes, no contiene lógica de objetos.
- Carga los estilos de Prime React que se usarán en el componente de *Lista Metas*.

El componente simplemente organiza los demás con la finalidad de tener organizado y delimitado el espacio que usarán en la pantalla.

### 7.1.13. Modal Cuenta

Este componente tiene dualidad en su funcionamiento ya es reutilizado en función de lo que haya seleccionado el usuario en el componente *Mi Cuenta*.

Tendrá dos vistas principalmente:

- Cambiar la contraseña del usuario (tipo === “editar”)
- Confirmar la eliminación de la cuenta (tipo === “eliminar”)

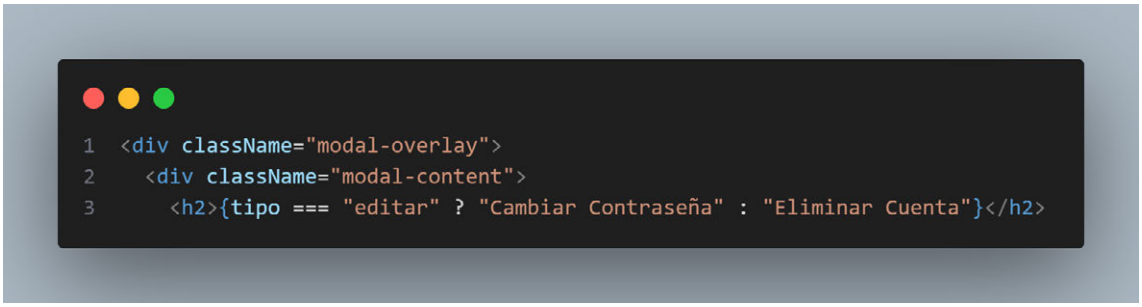



*Ilustración 25. Modal Cuenta*

Los inputs según el tipo varían:

- Si es de tipo editar
  - Contraseña actual
  - Nueva contraseña
  - Confirmación de la nueva contraseña
- Si es de tipo eliminar
  - Muestra el campo para la contraseña actual
  - Muestra una advertencia de que la acción es irreversible

En este capítulo se mostrarán los resultados de la página web. Se ilustrará mediante imágenes todas las funcionalidades explicadas durante el documento.

## 8.1. Pantalla Inicio de sesión

En esta pantalla se encuentra el formulario de inicio de sesión donde el usuario usará sus credenciales para entrar a la aplicación.

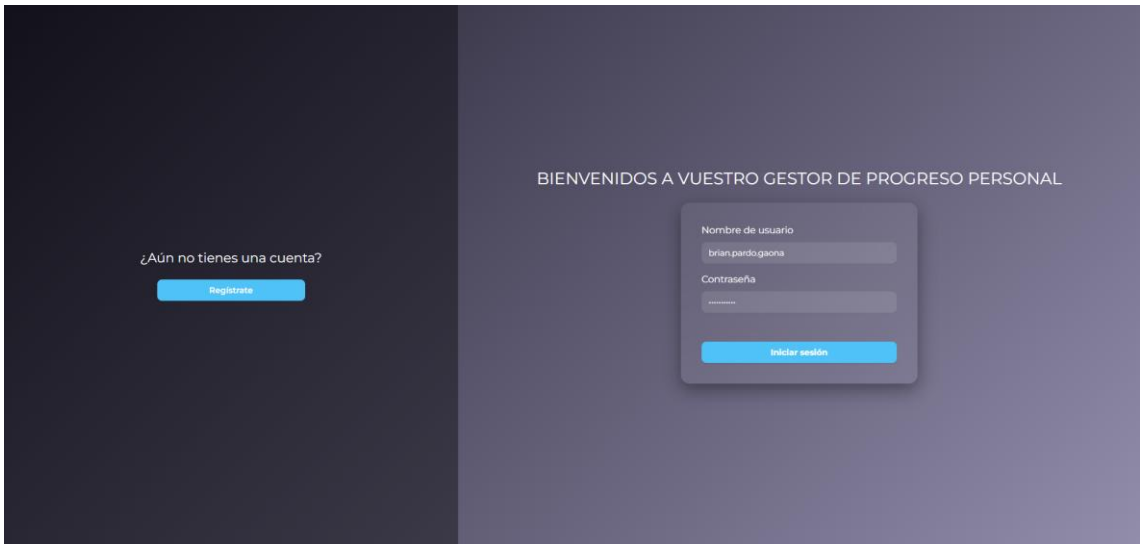



*Ilustración 26. Pantalla Inicio de sesión*

## 8.2. Pantalla Registro

En esta pantalla cualquier persona se puede dar de alta y pasar a ser usuario de la aplicación.

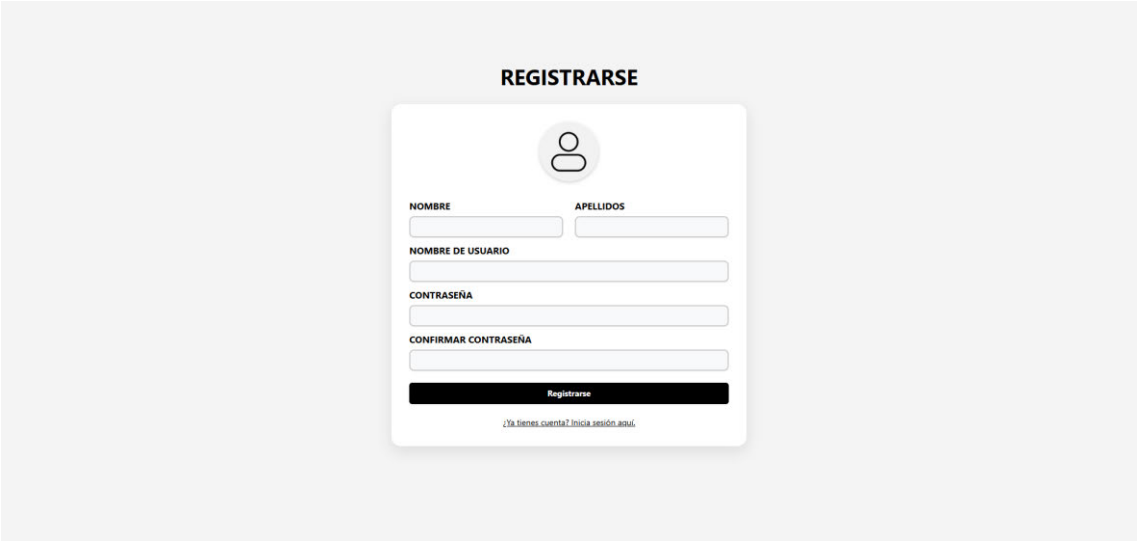



*Ilustración 27. Pantalla de registro*

## 8.3. Pantalla Inicio – Objetivos por mes

En esta pantalla se encuentra la primera vista que ve el usuario al entrar a la aplicación. Muestra un resumen de los objetivos completados a lo largo del tiempo.

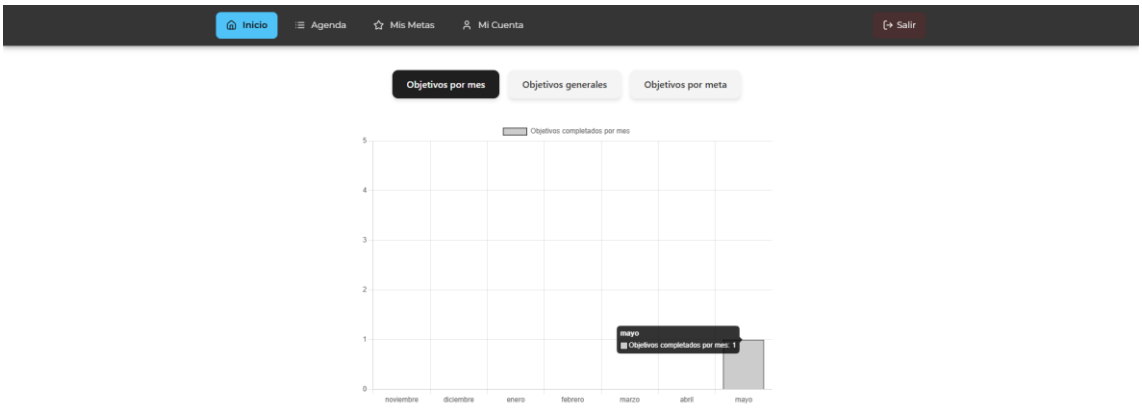



*Ilustración 28. Pantalla de inicio - Objetivos por mes*

## 8.4. Pantalla Inicio – Objetivos generales

En esta vista el usuario puede ver los objetivos completados y no completados entre todas sus metas.


*Ilustración 29. Pantalla de Inicio - Objetivos generales*

## 8.5. Pantalla Inicio – Objetivos por meta

En esta pantalla el usuario puede consultar de forma individual los objetivos que ha completado y los que no para cada una de sus metas.

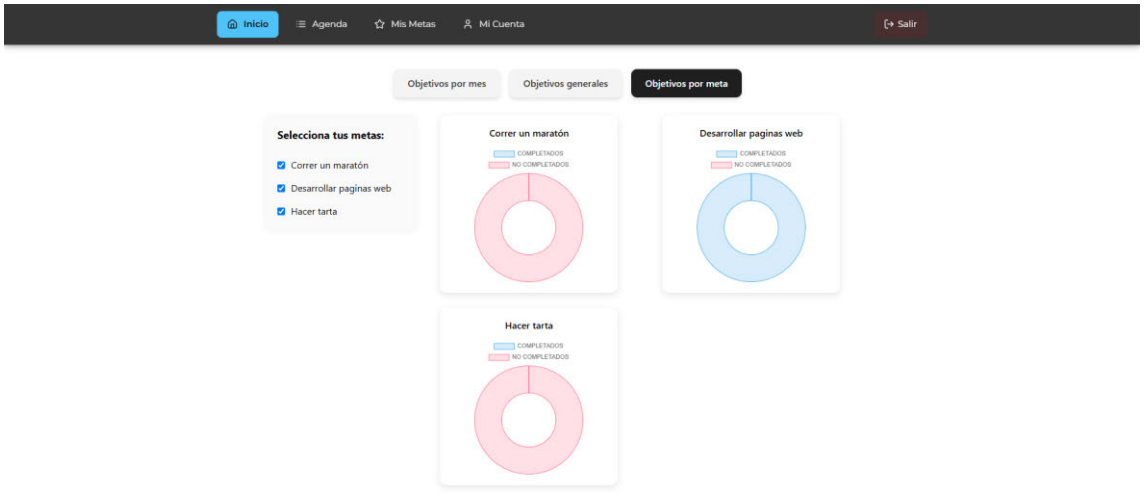



*Ilustración 30. Pantalla de inicio - Objetivos por meta*

## 8.6. Pantalla Agenda

En esta pantalla el usuario se encuentra todos sus objetivos en progreso.


*Ilustración 31. Pantalla de Agenda*

## 8.7. Pantalla Mis Metas

En esta pantalla el usuario se encuentra todas sus metas junto a sus respectivos objetivos.

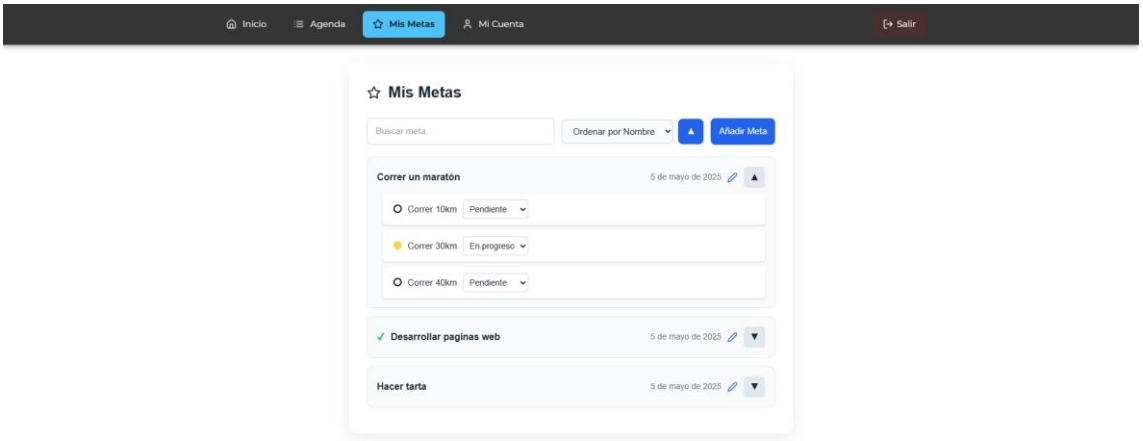



*Ilustración 32. Pantalla de Mis Metas*

## 8.8. Pantalla Crear Meta

En esta pantalla el usuario puede crear su meta con sus objetivos.


*Ilustración 33. Pantalla de Crear Meta*

## 8.9. Pantalla Editar Meta

En esta pantalla el usuario puede modificar su meta y sus objetivos.

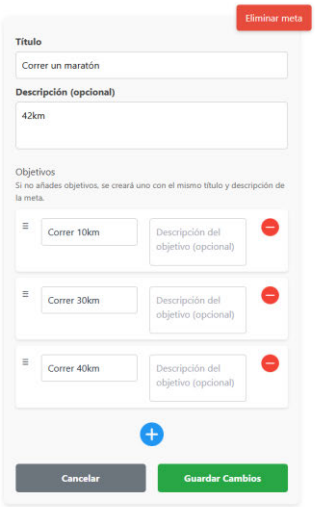



*Ilustración 34. Pantalla de Editar Meta*

## 8.10. Pantalla Detalle Meta

En esta pantalla el usuario puede consultar todos los detalles de su meta y objetivos.

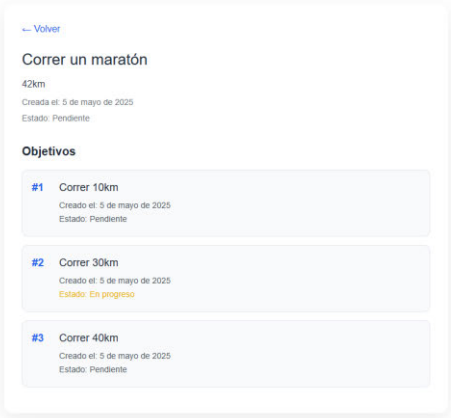



*Ilustración 35. Pantalla de Detalle Meta*

## 8.11. Pantalla Mi Cuenta

En esta pantalla el usuario puede consultar y modificar su información personal.

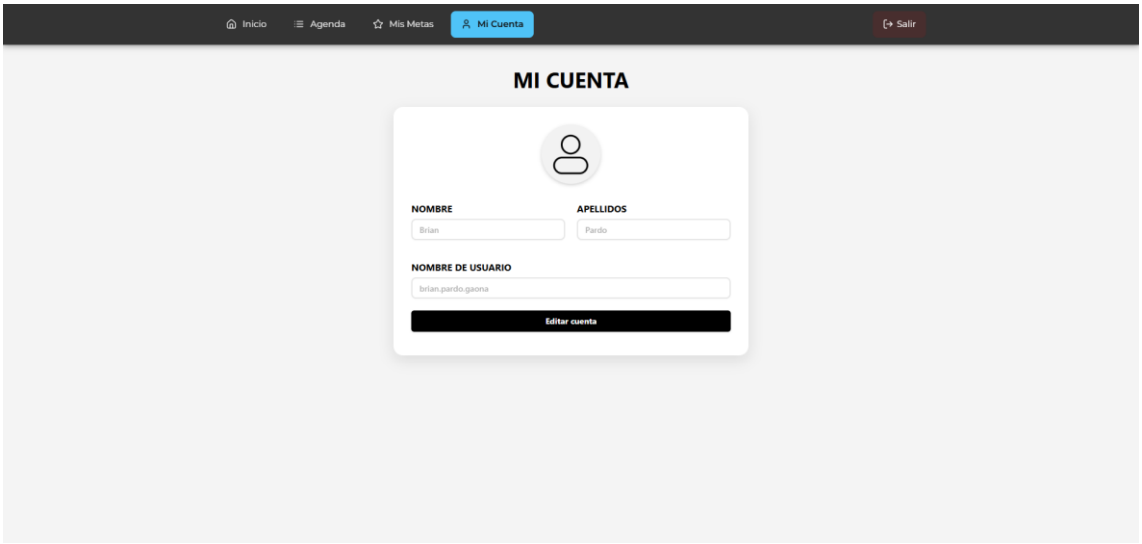



*Ilustración 36. Pantalla de Mi Cuenta*

## 8.12. Barra de navegación

Esta barra de navegación está presente en todas las pantallas que se encuentran dentro de la aplicación. Incluye todos los ítems para permitir al usuario moverse por la aplicación, además de cerrar la sesión actual.


*Ilustración 37. Barra de navegación*

En este apartado se realizarán las pruebas de la página web comprobando el correcto funcionamiento cada una de las funcionalidades que se han propuesto en este mismo documento en el capítulo 4.

Se realizarán las pruebas clasificándolas según la índole de las funcionalidades:

- Pruebas de las funcionalidades del usuario
- Pruebas de las funcionalidades del usuario en relación con las metas
- Pruebas de las funcionalidades del usuario en relación con los objetivos

## 9.1. Pruebas de las funcionalidades del usuario

En este apartado se realizarán las pruebas de las funcionalidades que puede realizar el usuario en la página web en relación con su cuenta en la plataforma.

### 9.1.1. Prueba – Registro

Para realizar esta prueba empezaremos desde la página de principal de la página. Accedemos a la pantalla de registro.

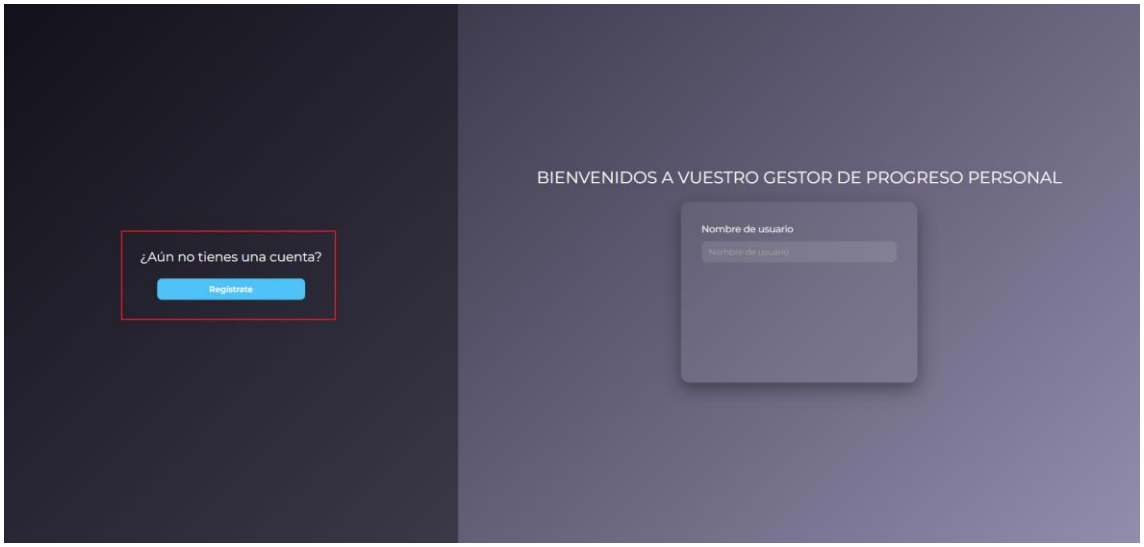



*Ilustración 38. Prueba Registro - Paso 1*

A continuación, rellenamos los datos del usuario a crear.

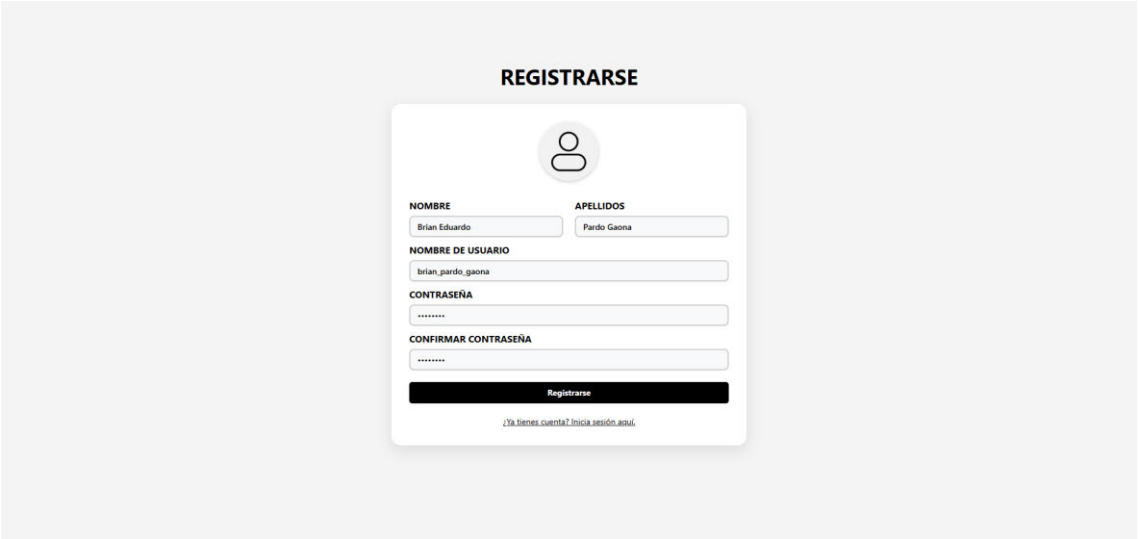



*Ilustración 39. Prueba Registro - Paso 2*

Tras registrarnos, nos redirige directamente al inicio de la aplicación.


*Ilustración 40. Prueba Registro - Paso 3*

Podemos comprobar que los datos corresponden tanto en el apartado Mi Cuenta como en la base de datos.

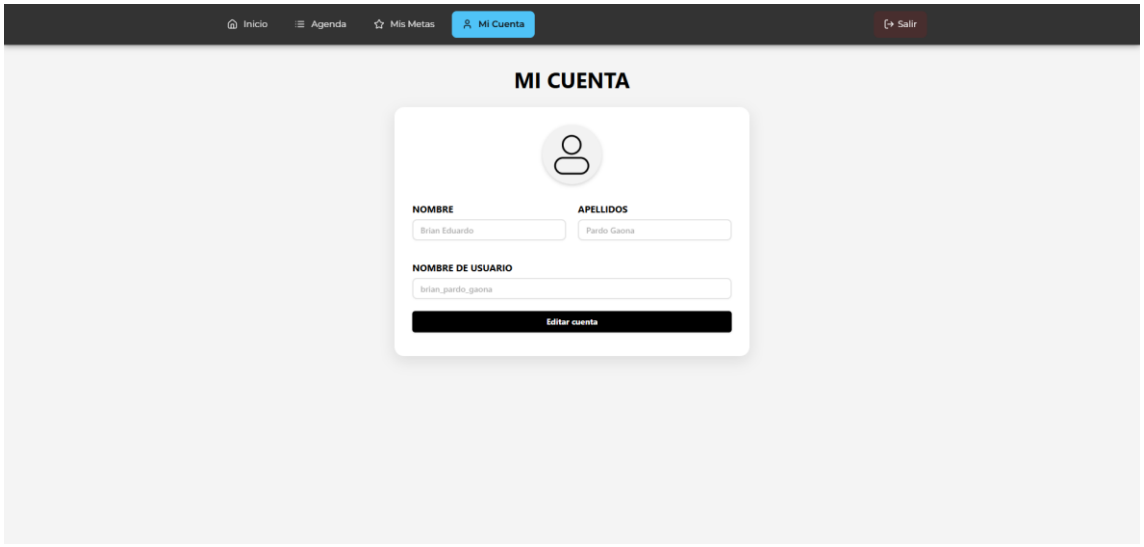



*Ilustración 41. Prueba Registro - Paso 4*

| | id [PK] uuid | username character varying (255) | name character varying (255) | lastName character varying (255) | password character varying (255) |
|---|---|---|---|---|---|
| 1 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | brian_pardo_gaona | Brian Eduardo | Pardo Gaona | $2b$10$lMGVK1YpSGMKj6hGL988L.NBDL9J4aU.9o9LRNbCck01QmZI7.3... |

*Ilustración 42. Prueba Registro - Paso 5*

Resultado de la prueba: **Superada**

### 9.1.2. Prueba – Inicio de sesión

Cerramos la sesión de la prueba anterior y comprobamos que se puede iniciar con las credenciales introducidas en el registro.

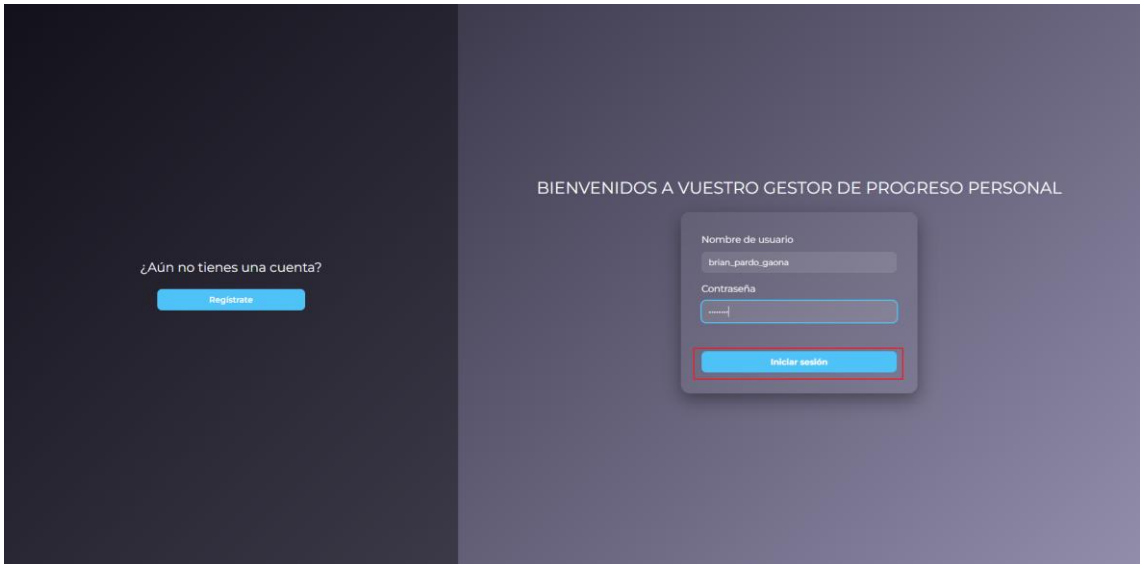



*Ilustración 43. Prueba Inicio Sesión - Paso 1*

Comprobamos que el usuario inicia sesión en la página y que su token se almacena en las cookies.

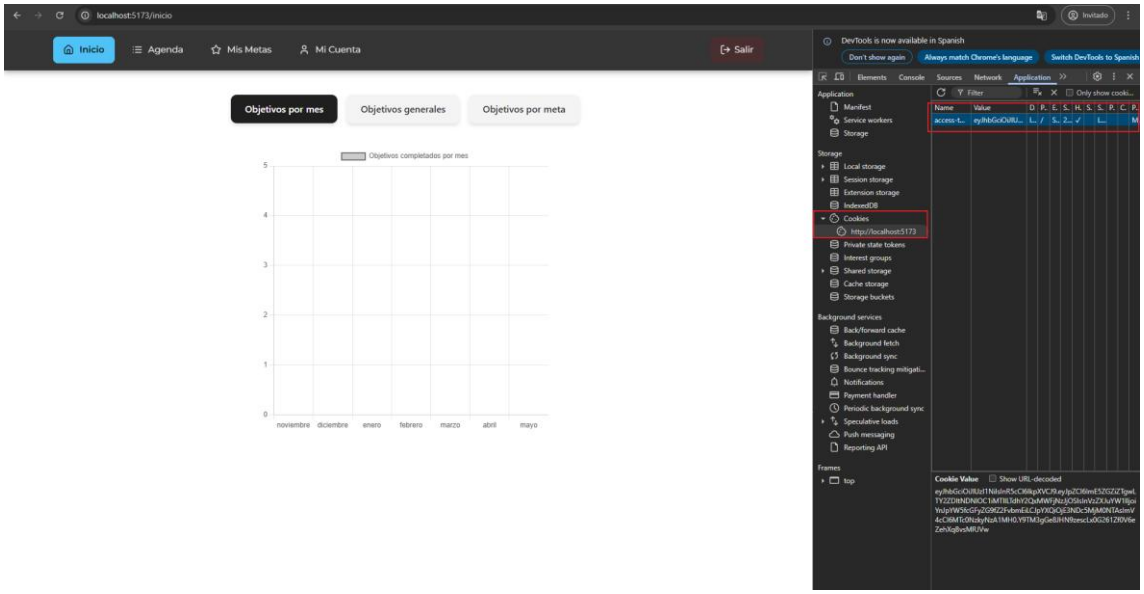



*Ilustración 44. Prueba Inicio Sesión - Paso 2*

Resultado de la prueba: **Superada**

### 9.1.3. Prueba – Consulta de perfil

Comprobamos que se puede consultar el perfil y que coincide con el usuario introducido en la ventana de registro, que se encuentra en el apartado 'Mi Cuenta'.

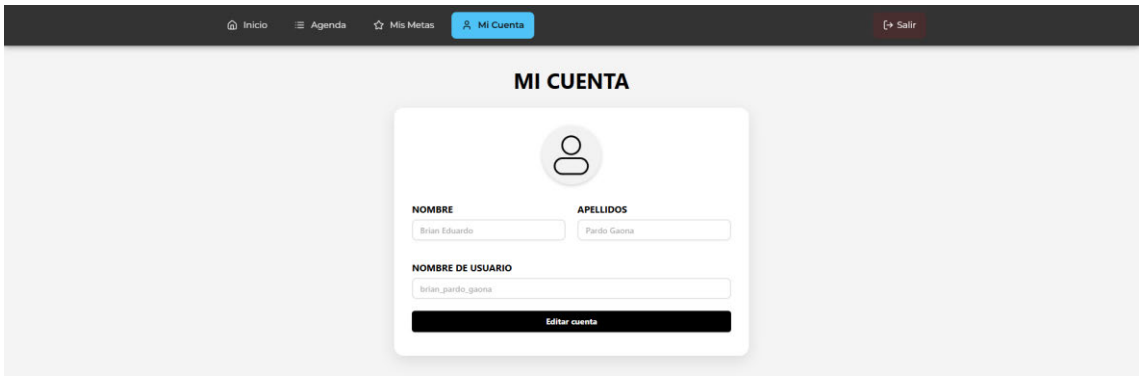



*Ilustración 45. Prueba Consulta de perfil - Paso 1*

Resultado de la prueba: **Superada**

### 9.1.4. Prueba – Modificación de perfil

A continuación, se va a probar la funcionalidad de modificación de perfil. Para ello vamos a cambiar el nombre actual 'Brian Eduardo' por 'Rafael' y los apellidos 'Pardo Gaona' por 'Nadal'.

Comprobamos que, de primeras, no nos deja cambiar ningún campo puesto que está en modo consulta.

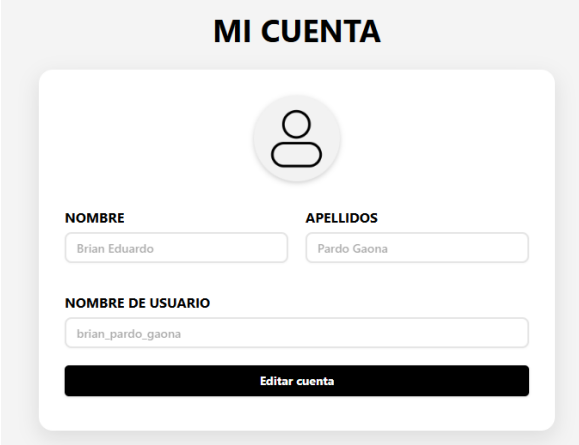



*Ilustración 46. Prueba Modificación perfil - Paso 1*

Para activarlo, debemos seleccionar el botón 'Editar cuenta' y veremos cómo los campos se activan y, además, habilitan dos opciones más.


*Ilustración 47. Prueba Modificación perfil - Paso 2*

Realizamos los cambios y seleccionamos ‘Guardar cambios’.


*Ilustración 48. Prueba Modificación perfil - Paso 3*

Comprobamos que nos aparece una alerta que nos indica que el cambio se ha realizado satisfactoriamente.

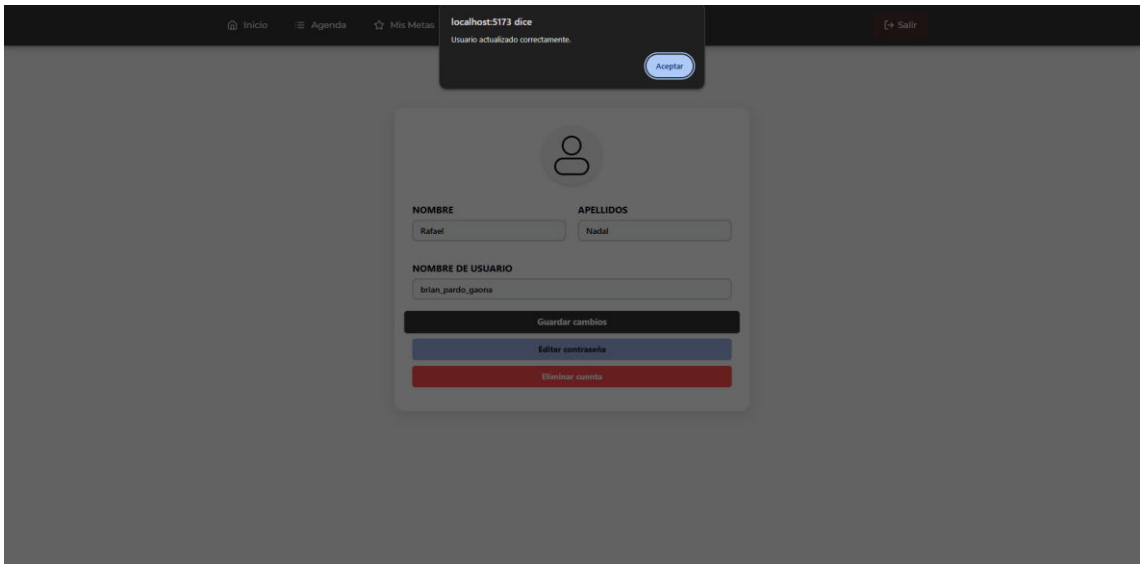



*Ilustración 49. Prueba Modificación perfil - Paso 4*

Comprobamos que los datos están actualizados en el mismo apartado de ‘Mi Cuenta’ y comprobamos en la base de datos que en la tabla User aparece el usuario modificado.

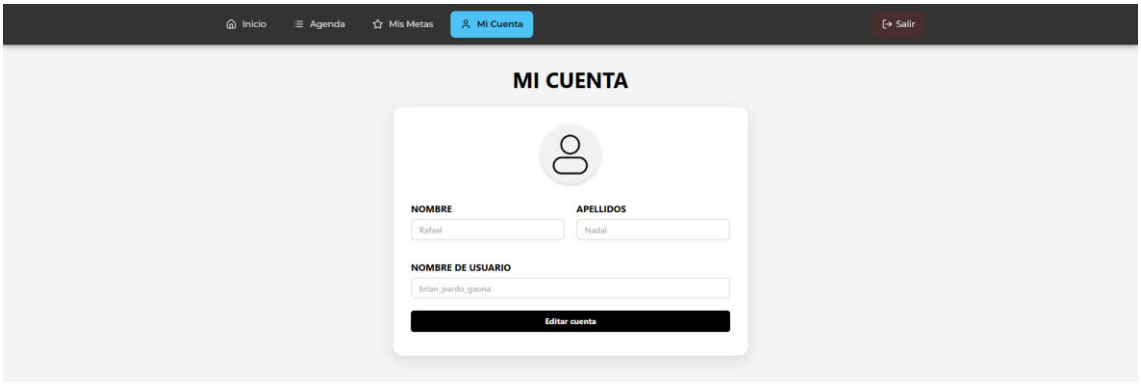



*Ilustración 50. Prueba Modificación perfil - Paso 5*

| | id [PK] uuid | username character varying (255) | name character varying (255) | lastName character varying (255) | password character varying (255) |
|---|---|---|---|---|---|
| 1 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | brian_pardo_gaona | Rafael | Nadal | $2b$10$iMGVK1YpSGMKj6hGL988L.NBDL9J4aU.9o9LRNbCck01QmZI7.3... |

*Ilustración 51. Prueba Modificación perfil - Paso 6*

Resultado de la prueba: **Superada**

### 9.1.5. Prueba – Cambio de contraseña

Para esta prueba accedemos a 'Mi Cuenta' y de las opciones que se nos habilitan tras seleccionar 'Editar cuenta', seleccionamos modificar contraseña.


*Ilustración 52. Prueba Cambio contraseña - Paso 1*

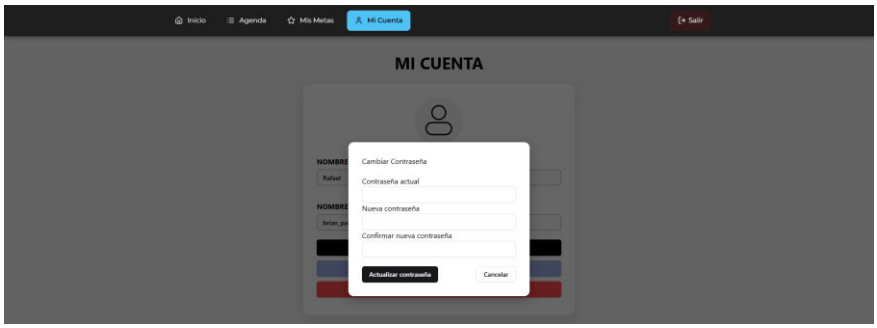



*Ilustración 53. Prueba Cambio contraseña - Paso 2*

Probaremos primero poniendo una contraseña que no coincide con la actual.

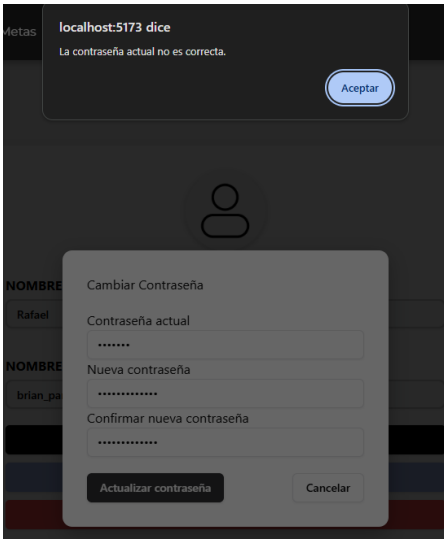



*Ilustración 54. Prueba Cambio contraseña - Paso 3*

También probaremos poniendo dos contraseñas que no coinciden.

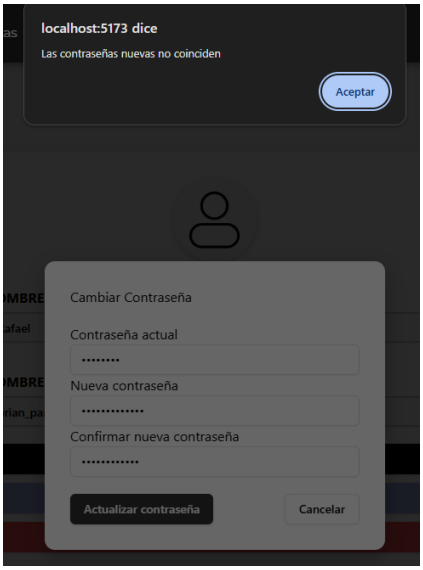



*Ilustración 55. Prueba Cambio contraseña - Paso 4*

A continuación, cambiamos la contraseña correctamente. Para este caso, es complicado mostrar la contraseña, por lo tanto, comprobaremos su cambio en la base de datos, que, aunque esté cifrada, debe ser diferente.

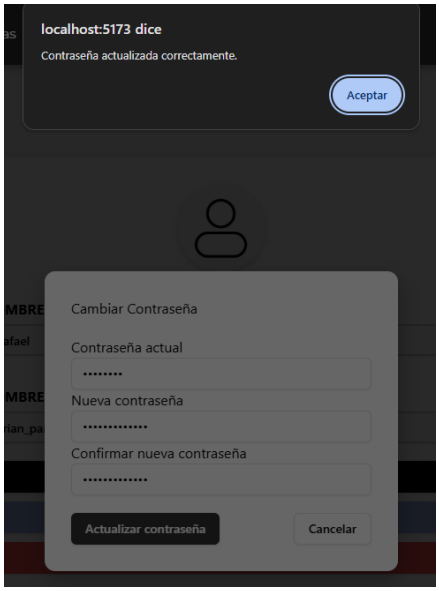



*Ilustración 56. Prueba Cambio contraseña - Paso 5*

- Contraseña antes del cambio

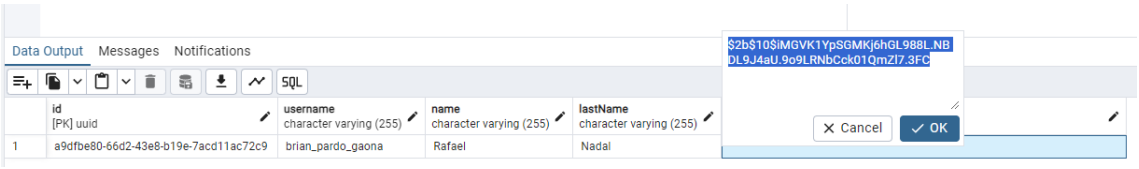



*Ilustración 57. Prueba Cambio contraseña - Paso 6*

- Contraseña después del cambio

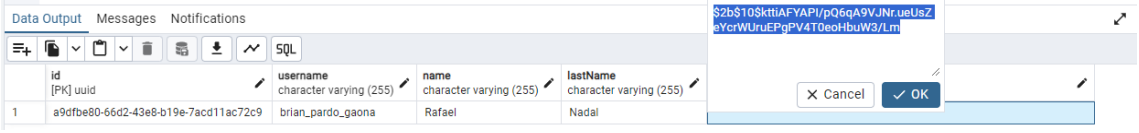



*Ilustración 58. Prueba Cambio contraseña - Paso 7*

Resultado de la prueba: **Superada**

### 9.1.6. Prueba – Eliminación de cuenta

Para la eliminación de la cuenta, hay que acceder al apartado de 'Mi Cuenta', habilitar la edición de cuenta y seleccionar 'Eliminar cuenta'. Aparecerá un modal así:

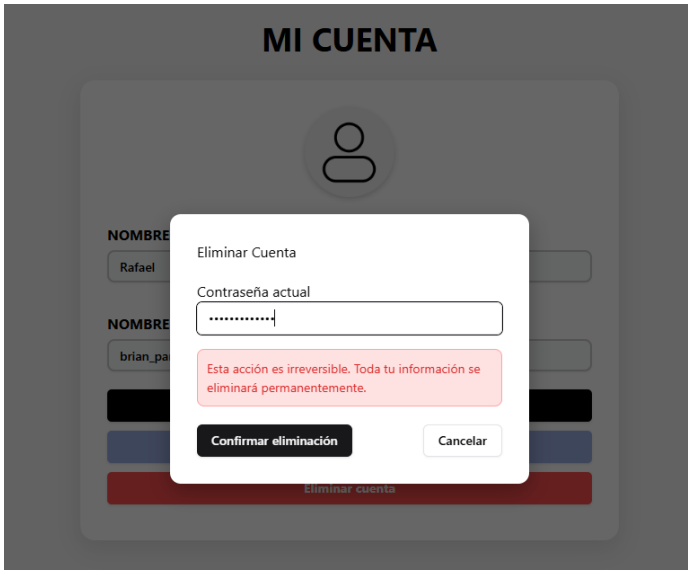



*Ilustración 59. Prueba Eliminación cuenta - Paso 1*

Una vez eliminada, te redirige al inicio de sesión confirmando la eliminación con una alerta.

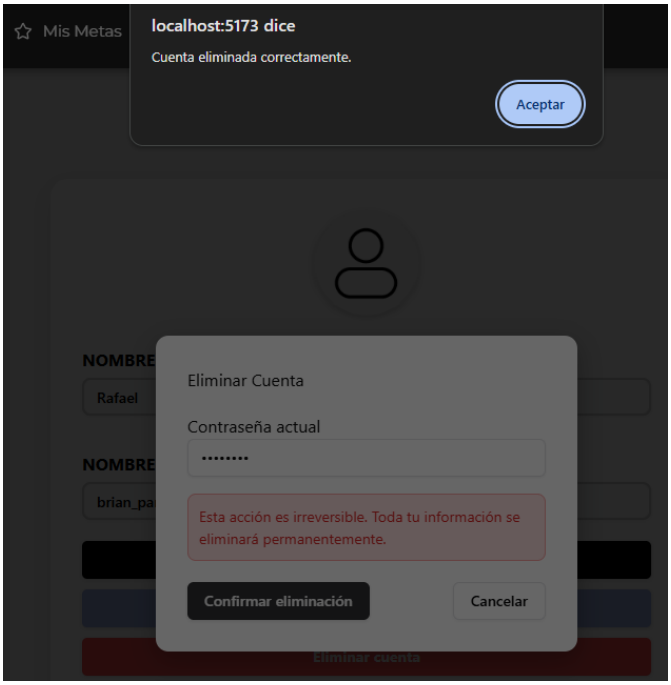



*Ilustración 60. Prueba Eliminación cuenta - Paso 2*

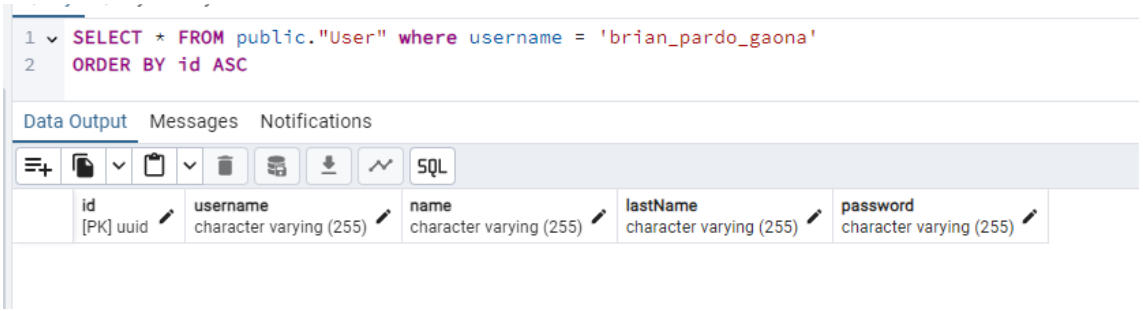



*Ilustración 61. Prueba Eliminación cuenta - Paso 3*

Resultado de la prueba: **Superada**

## 9.2. Pruebas de las funcionalidades del usuario en relación con las metas

### 9.2.1. Prueba – Crear Meta con objetivos

Para esta prueba accederemos al apartado de 'Mis Metas' y seleccionaremos 'Añadir meta'.

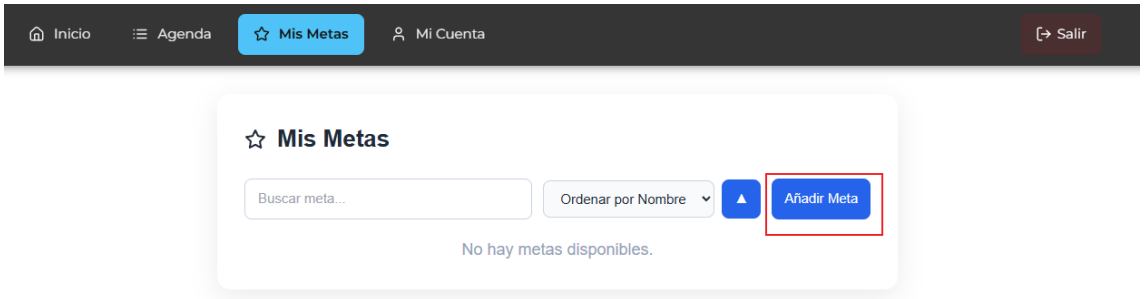



*Ilustración 62. Prueba Crear meta con objetivos - Paso 1*

Después rellenaremos la meta y sus objetivos, y seleccionamos 'Guardar'.

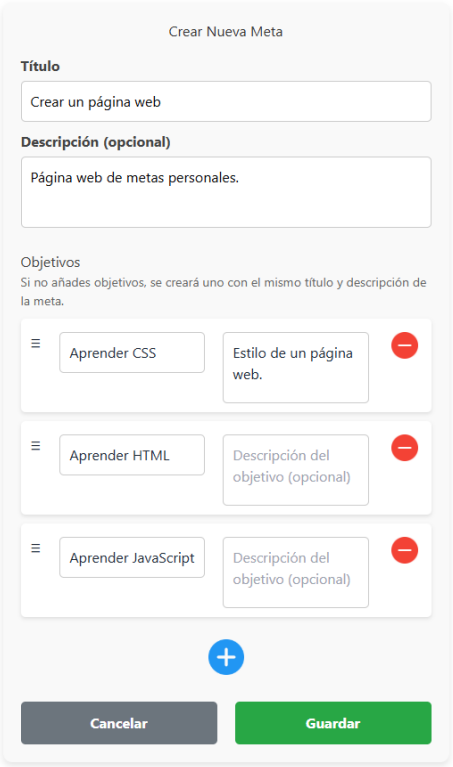



*Ilustración 63. Prueba Crear meta con objetivos - Paso 2*

Vemos que se crea y se muestra en la lista. Comprobamos también la base de datos.

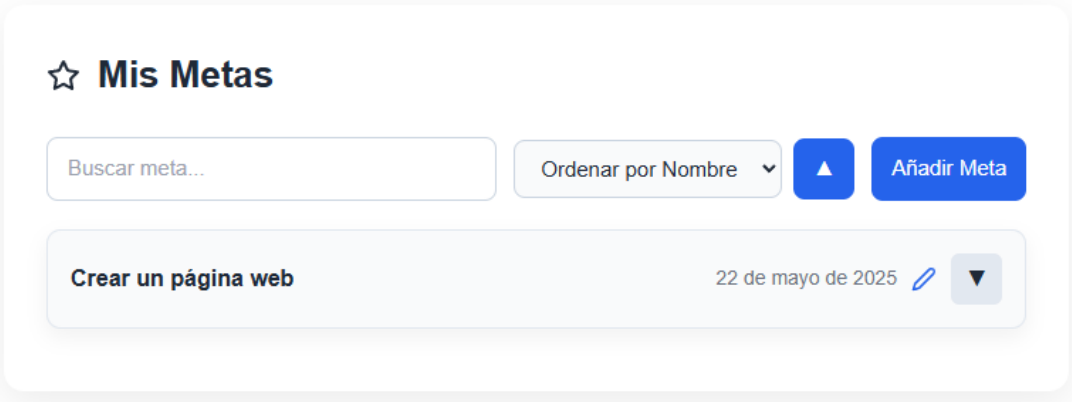



*Ilustración 64. Prueba Crear meta con objetivos - Paso 3*

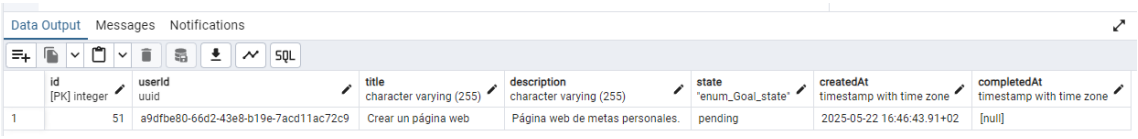



*Ilustración 65. Prueba Crear meta con objetivos - Paso 4*

Resultado de la prueba: **Superada**

### 9.2.2. Prueba – Crear Meta sin objetivos

A continuación, creamos una meta sin objetivos. Esto implica que se deberá crear un objetivo con el mismo nombre y descripción que la meta.

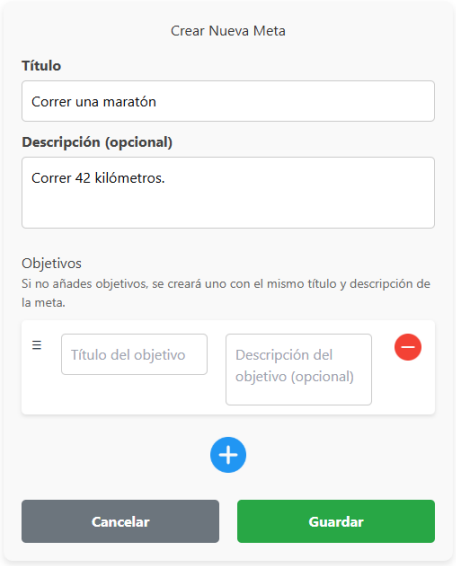



*Ilustración 66. Prueba Crear meta sin objetivos - Paso 1*

Comprobamos que existe un objetivo dentro de la meta con su mismo nombre.

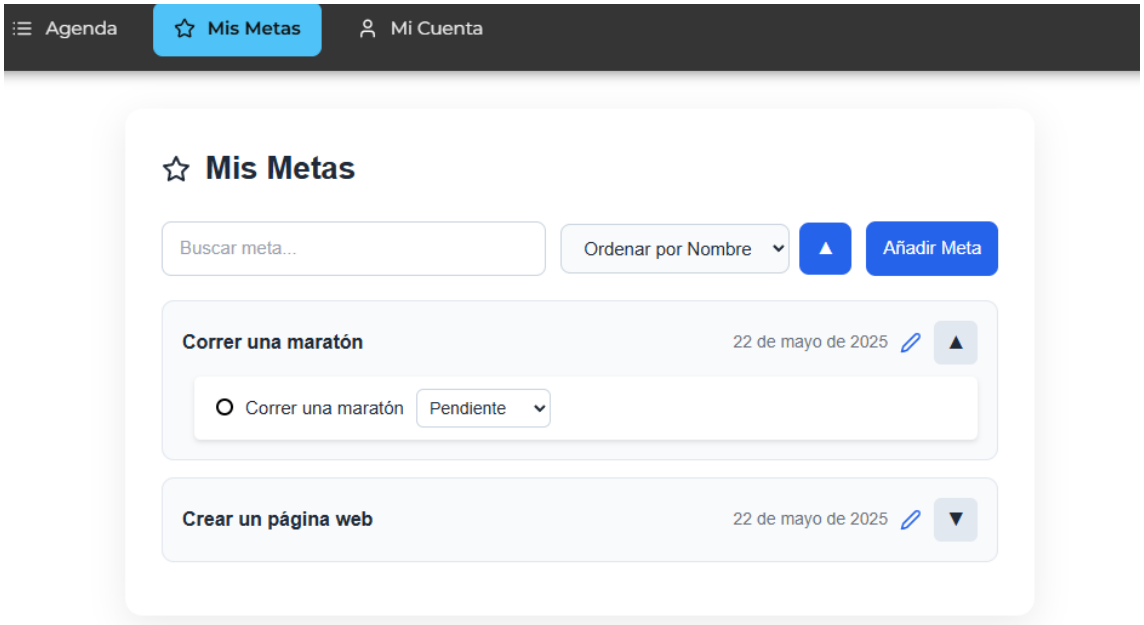



*Ilustración 67. Prueba Crear meta sin objetivos - Paso 2*

Resultado de la prueba: **Superada**

### 9.2.3. Prueba – Consulta de metas

Gracias a esta funcionalidad podemos comprobar la meta y sus objetivos de un vistazo rápido, simplemente desde Mis Metas.

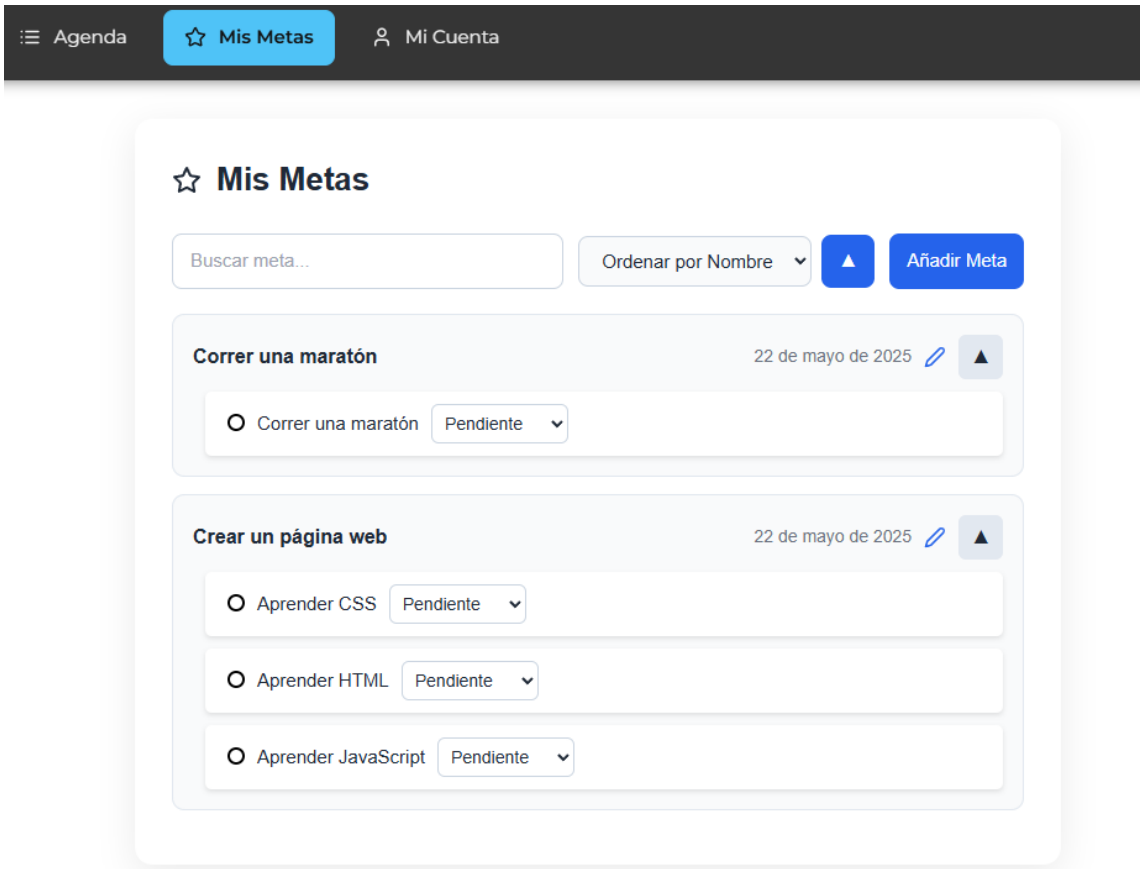



*Ilustración 68. Prueba Consulta metas - Paso 1*

Resultado de la prueba: **Superada**

### 9.2.4. Prueba – Consulta del detalle de una meta

Para esta prueba, consultaremos toda la información de las dos metas creadas anteriormente con la finalidad de comprobar que su información es correcta.

Para ello, deberemos seleccionar el nombre de la meta.

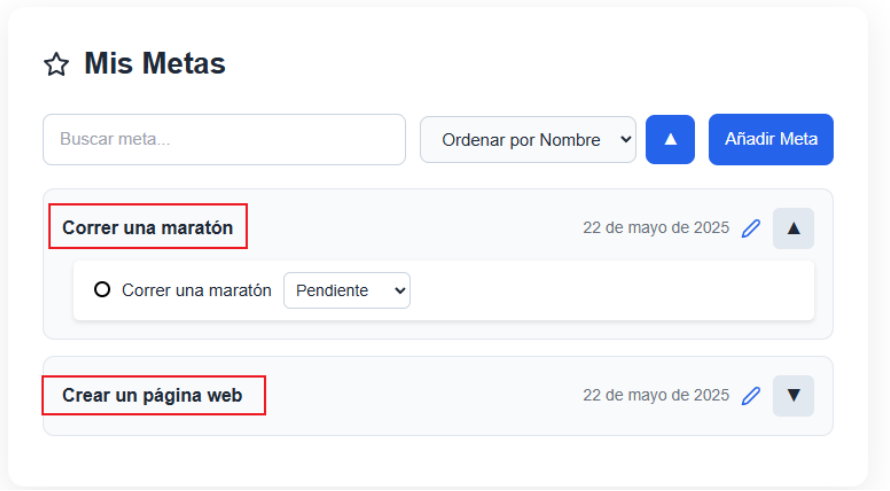



*Ilustración 69. Prueba Consulta detalle meta - Paso 1*

Y comprobamos el contenido de las dos metas es correcto.

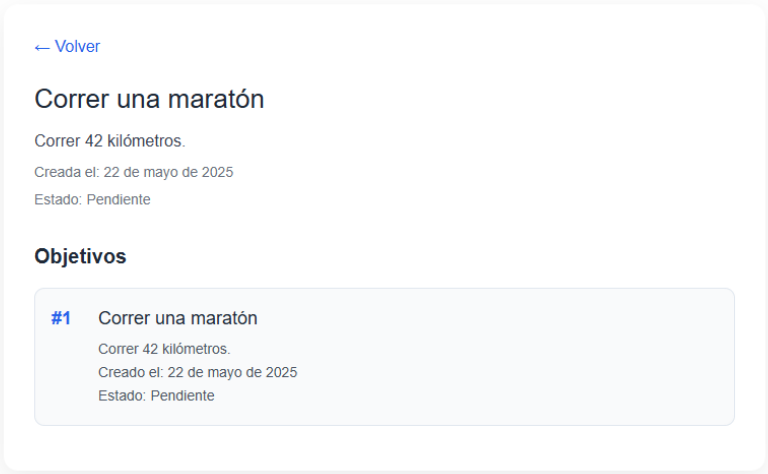



*Ilustración 70. Prueba Consulta detalle meta - Paso 2*

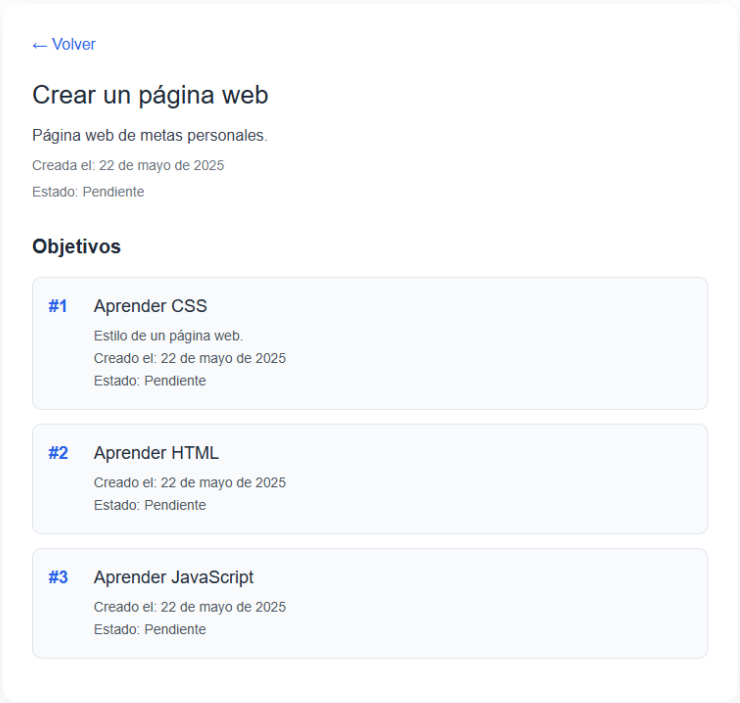



*Ilustración 71. Prueba Consulta detalle meta - Paso 3*

Resultado de la prueba: **Superada**

### 9.2.5. Prueba – Modificación de una meta

Para modificar una meta, simplemente debemos seleccionar el botón con forma de lápiz a la derecha de cada meta. Este botón nos llevará a una ventana similar a la de creación de metas, pero con el contenido de la meta a modificar. Vamos a cambiar el título a 'Página web'.

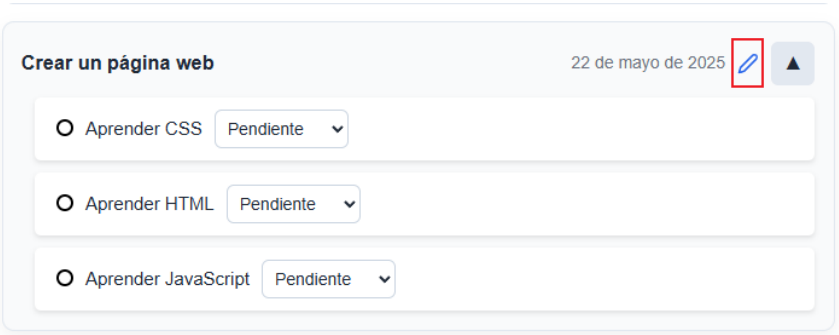



*Ilustración 72. Prueba Modificación meta - Paso 1*

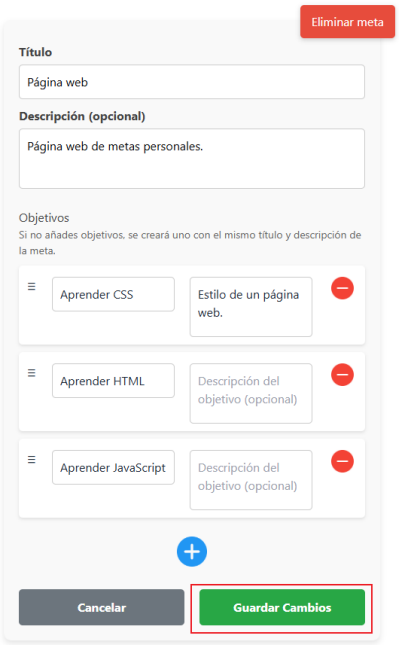



*Ilustración 73. Prueba Modificación meta - Paso 2*

Comprobamos que el nombre se ha cambiado en la lista y en la base de datos.

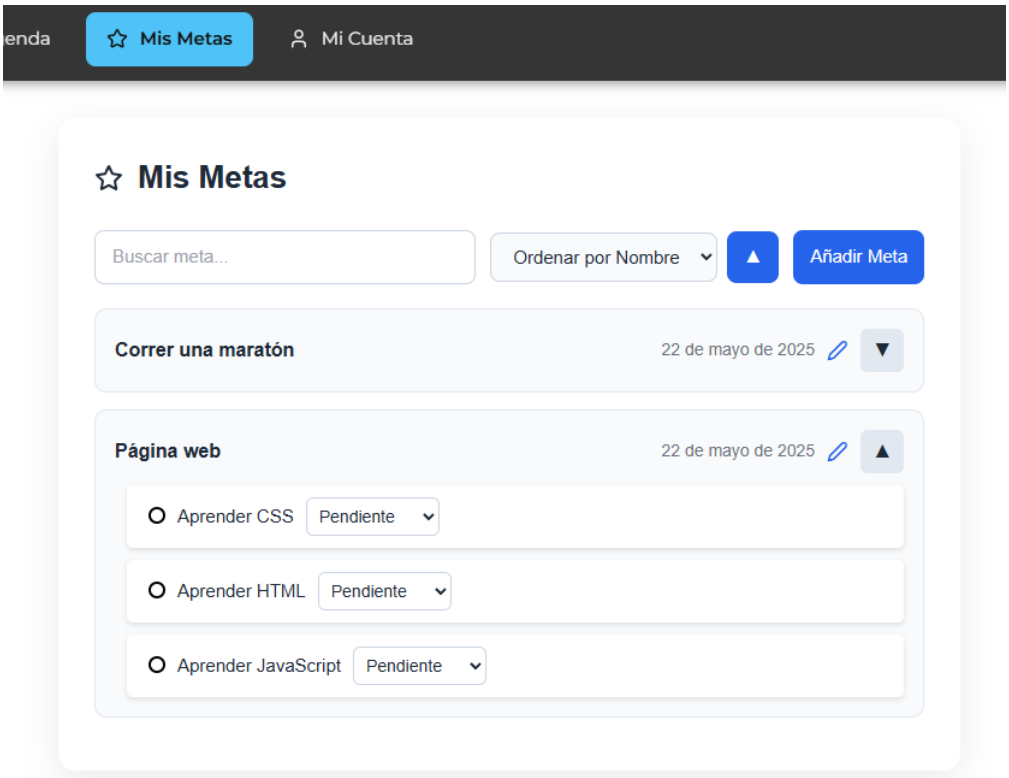
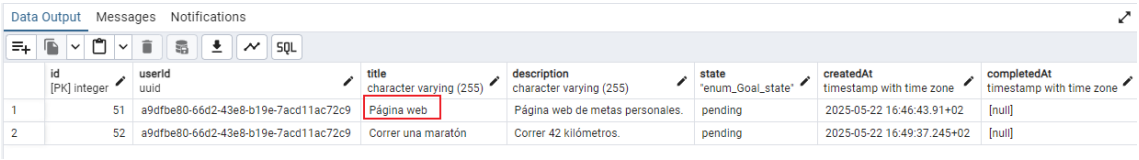



*Ilustración 74. Prueba Modificación meta - Paso 3*




| | id [PK] integer | userId uuid | title character varying (255) | description character varying (255) | state "enum_Goal_state" | createdAt timestamp with time zone | completedAt timestamp with time zone |
|---|---|---|---|---|---|---|---|
| 1 | 51 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Página web | Página web de metas personales. | pending | 2025-05-22 16:46:43.91+02 | [null] |
| 2 | 52 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Correr una maratón | Correr 42 kilómetros. | pending | 2025-05-22 16:49:37.245+02 | [null] |

*Ilustración 75. Prueba Modificación meta - Paso 4*

Resultado de la prueba: **Superada**

### 9.2.6. Prueba – Eliminación de una meta

Para esta prueba simplemente, tenemos que seleccionar el botón 'Eliminar Meta' que se encuentra arriba a la derecha de la edición de una meta.

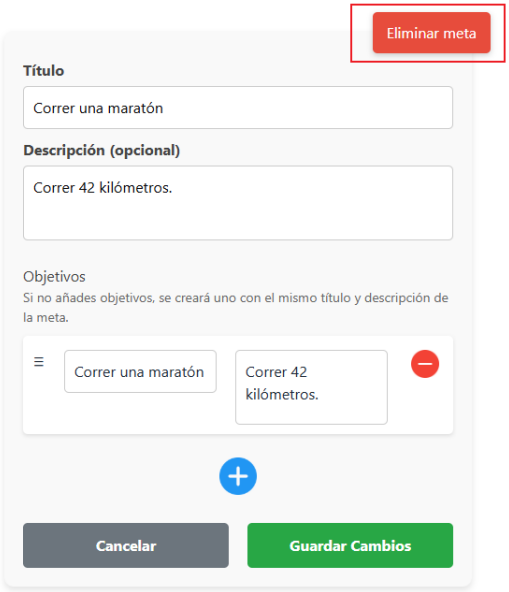



*Ilustración 76. Prueba Eliminación meta - Paso 1*

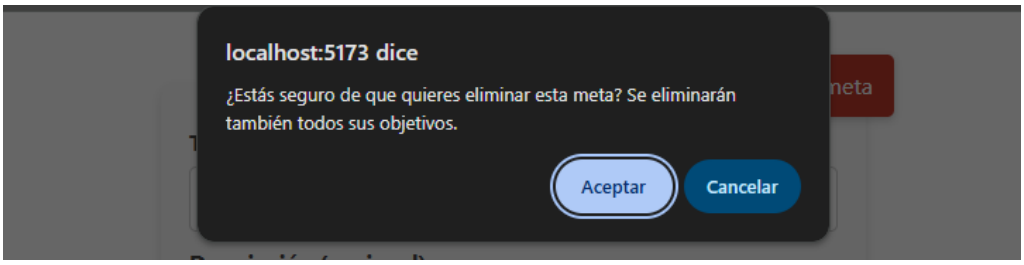



*Ilustración 77. Prueba Eliminación meta - Paso 2*

Vemos que la meta queda totalmente eliminada.

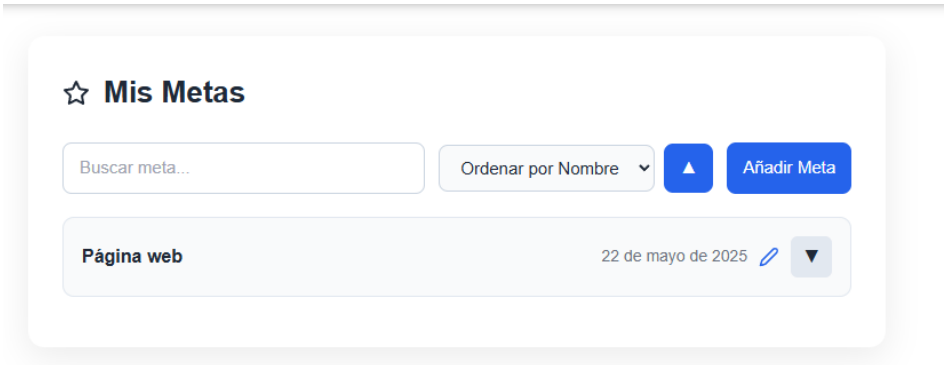



*Ilustración 78. Prueba Eliminación meta - Paso 3*

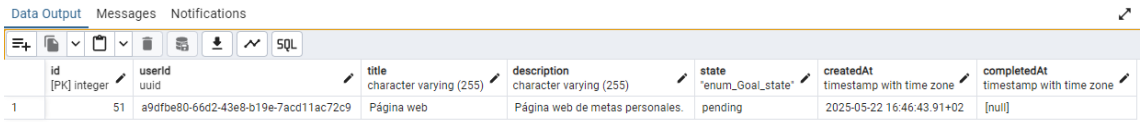



*Ilustración 79. Prueba Eliminación meta - Paso 4*

Resultado de la prueba: **Superada**

### 9.2.7. Prueba – Cambio de estado de una meta

Para que una meta cambie de estado a completada, todos sus objetivos deben estar completados.

Comprobamos que si están solo algunos de ellos la meta no está completada.

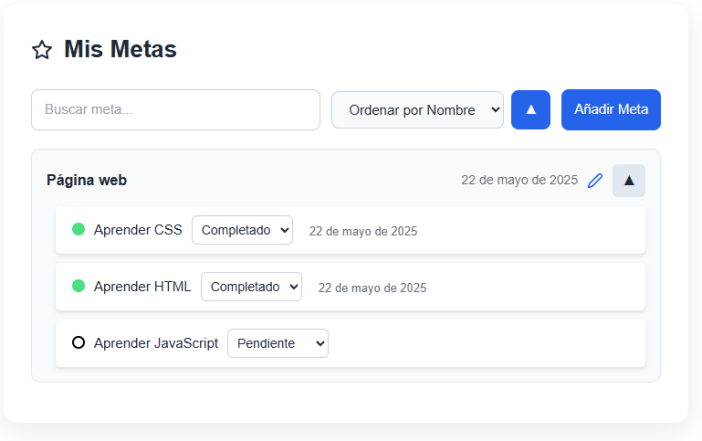
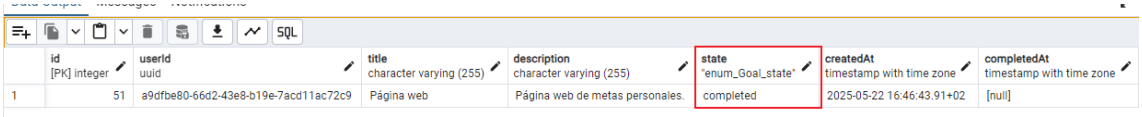



*Ilustración 80. Prueba Cambio estado meta - Paso 1*

Comprobamos que, si están todos los objetivos completados, la meta también.


*Ilustración 81. Prueba Cambio estado meta - Paso 2*



| | id [PK] integer | userId uuid | title character varying (255) | description character varying (255) | state "enum_Goal_state" | createdAt timestamp with time zone | completedAt timestamp with time zone |
|---|---|---|---|---|---|---|---|
| 1 | 51 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Página web | Página web de metas personales. | completed | 2025-05-22 16:46:43.91+02 | [null] |

*Ilustración 82. Prueba Cambio estado meta - Paso 3*

Resultado de la prueba: **Superada**

### 9.2.8. Prueba – Consulta de estado de la meta mediante gráficas

Para comprobar el estado de la meta correctamente, vamos a dejar, al menos un objetivo sin completar, de la siguiente manera:

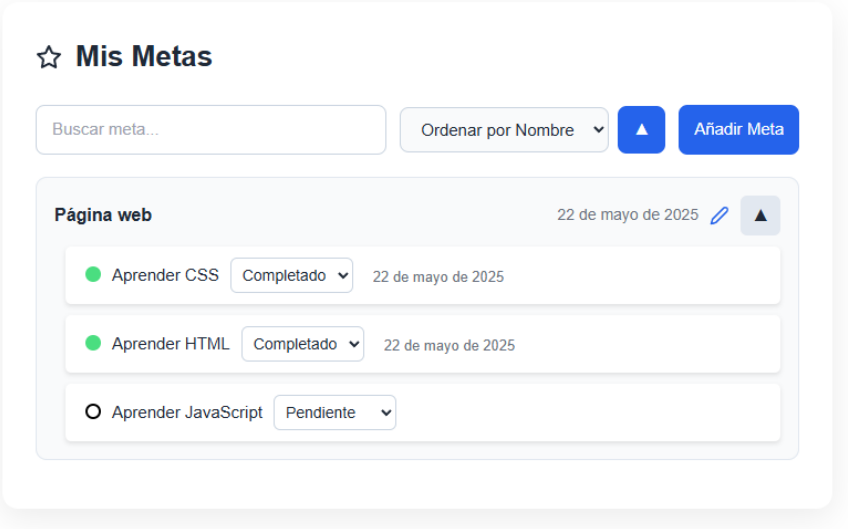



*Ilustración 83. Prueba Consulta estado meta mediante gráficas - Paso 1*

A continuación, comprobamos accediendo al apartado de Inicio > Objetivos por meta y seleccionando la meta en cuestión.

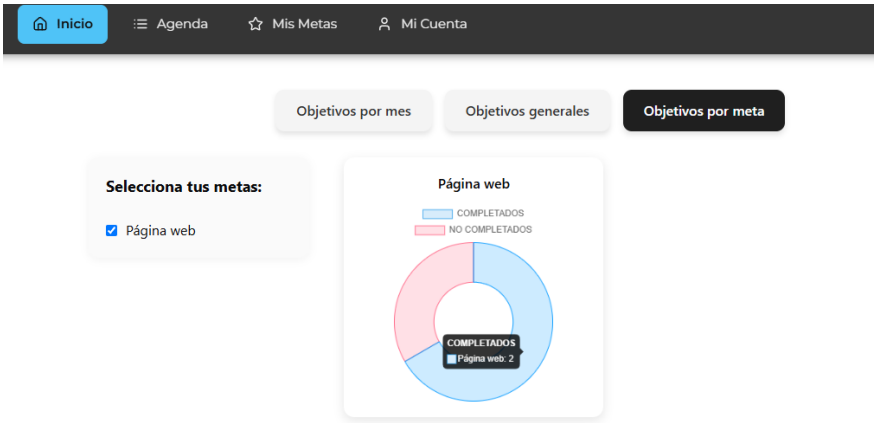



*Ilustración 84. Prueba Consulta estado meta mediante gráficas - Paso 2*

Resultado de la prueba: **Superada**

### 9.2.9. Prueba – Ordenamiento de metas por distintos filtros

Para probar este ejemplo, vamos a crear dos metas más. Existen dos filtros, por nombre y por fecha de creación. Se creará primero una meta que empiece por una letra posterior a P y otra anterior a P. La meta con letra anterior a P se creará después para que en el filtro aparezca la última por orden creación, pero la primera si se filtra por nombre.

Las metas serán por orden alfabético:

- Crear una receta.
- Página web.
- Realizar un maratón.

Y por orden de creación:

- Página web.
- Realizar una maratón.
- Crear una receta.

En ambos casos se puede ordenar ascendente y descendentemente.

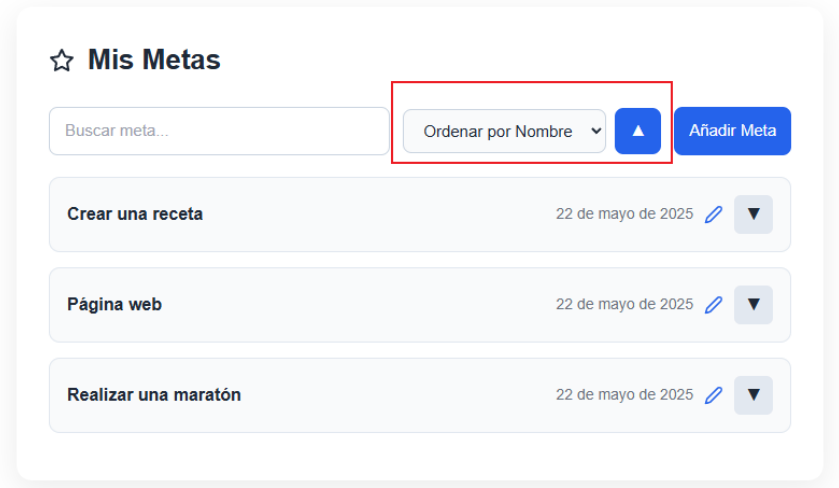



*Ilustración 85. Prueba Ordenación metas por filtros - Paso 1*

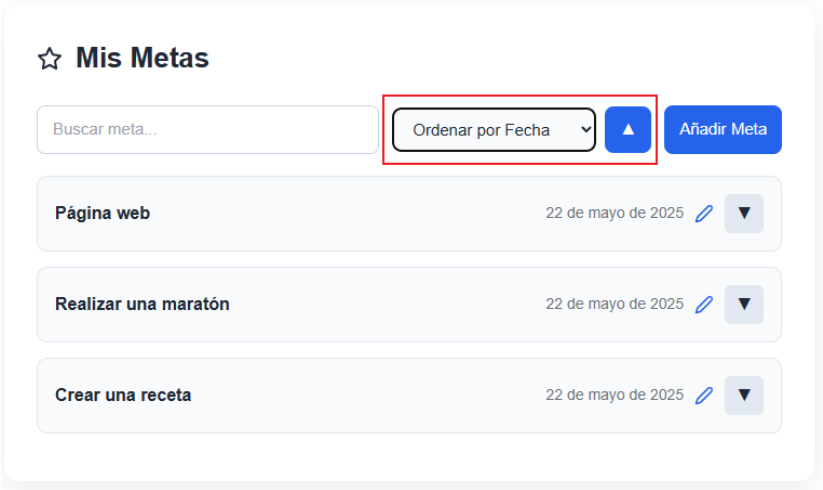
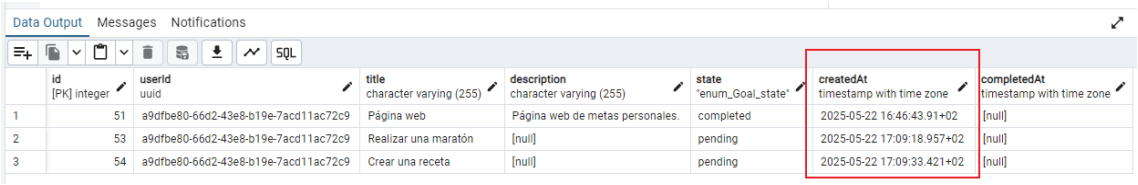



*Ilustración 86. Prueba Ordenación metas por filtros - Paso 2*



Data Output | Messages | Notifications

| | id [PK] integer | userId uuid | title character varying (255) | description character varying (255) | state "enum_Goal_state" | createdAt timestamp with time zone | completedAt timestamp with time zone |
|---|---|---|---|---|---|---|---|
| 1 | 51 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Página web | Página web de metas personales. | completed | 2025-05-22 16:46:43.91+02 | [null] |
| 2 | 53 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Realizar una maratón | [null] | pending | 2025-05-22 17:09:18.957+02 | [null] |
| 3 | 54 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Crear una receta | [null] | pending | 2025-05-22 17:09:33.421+02 | [null] |

*Ilustración 87. Prueba Ordenación metas por filtros - Paso 3*

Resultado de la prueba: **Superada**

## 9.3. Pruebas de las funcionalidades del usuario en relación con los objetivos

### 9.3.1. Prueba – Crear un objetivo para una meta existente

Para esta prueba se va a crear un nuevo objetivo para la meta de crear páginas web. Añadiremos al final el objetivo de 'Aprender React'.

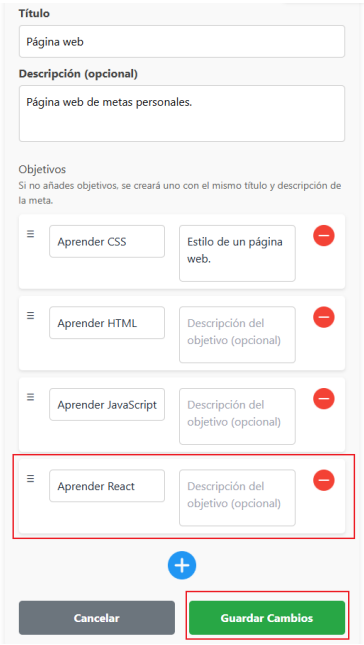
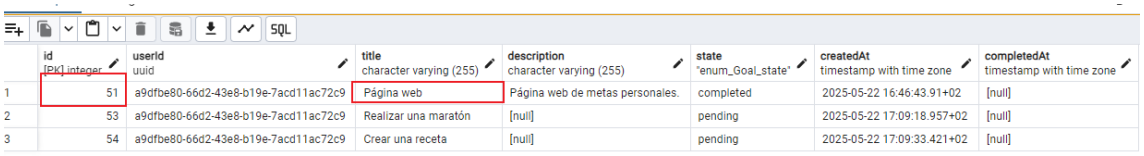


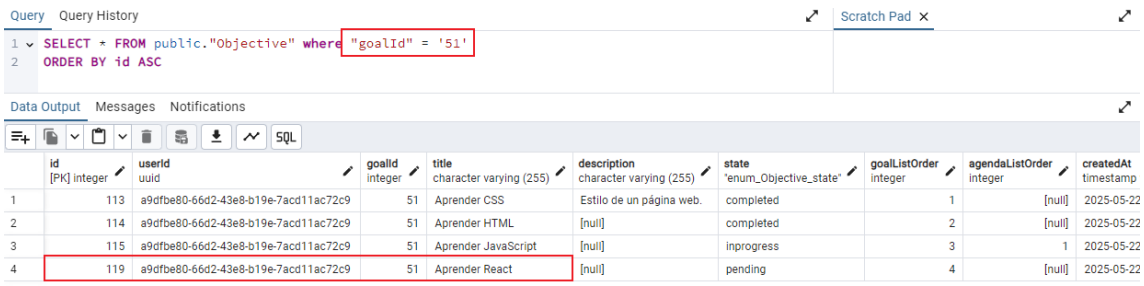

*Ilustración 88. Prueba Creación objetivo - Paso 1*

Comprobamos el id de la meta y que aparece dentro de los objetivos como clave foránea, con el nuevo objetivo.



| | id [PK] integer | userId uuid | title character varying (255) | description character varying (255) | state "enum_Goal_state" | createdAt timestamp with time zone | completedAt timestamp with time zone |
|---|---|---|---|---|---|---|---|
| 1 | 51 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Página web | Página web de metas personales. | completed | 2025-05-22 16:46:43.91+02 | [null] |
| 2 | 53 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Realizar una maratón | [null] | pending | 2025-05-22 17:09:18.957+02 | [null] |
| 3 | 54 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | Crear una receta | [null] | pending | 2025-05-22 17:09:33.421+02 | [null] |

*Ilustración 89. Prueba Creación objetivo - Paso 2*




| | id [PK] integer | userId uuid | goalId integer | title character varying (255) | description character varying (255) | state "enum_Objective_state" | goalListOrder integer | agendaListOrder integer | createdAt timestamp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 113 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender CSS | Estilo de un página web. | completed | 1 | [null] | 2025-05-22 |
| 2 | 114 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender HTML | [null] | completed | 2 | [null] | 2025-05-22 |
| 3 | 115 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender JavaScript | [null] | inprogress | 3 | 1 | 2025-05-22 |
| 4 | 119 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender React | [null] | pending | 4 | [null] | 2025-05-22 |

*Ilustración 90. Prueba Creación objetivo - Paso 3*

Resultado de la prueba: **Superada**

### 9.3.2. Prueba – Cambiar el estado de un objetivo

Para cambiar de estado simplemente hay que usar el desplegable que se encuentra a la derecha de cada objetivo, este cambia de estado automáticamente y lo actualiza en la base de datos. Por ejemplo, 'Aprender HTML' pasa a estar en progreso.

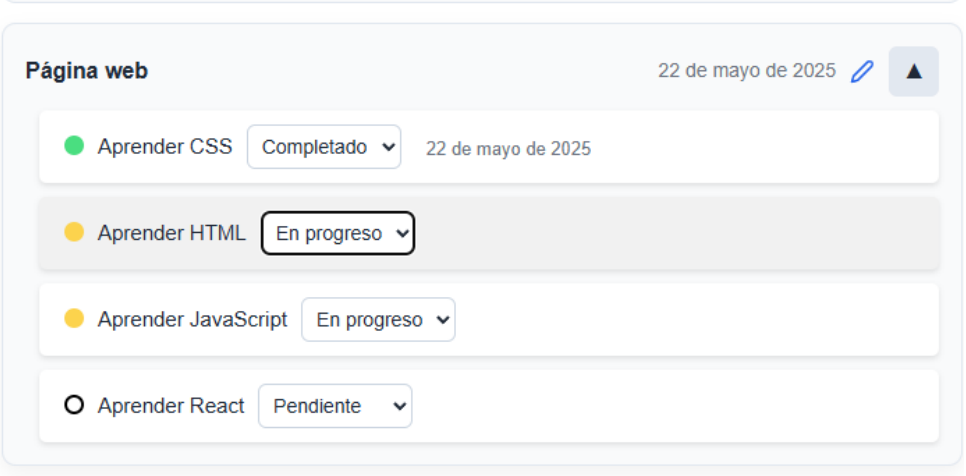
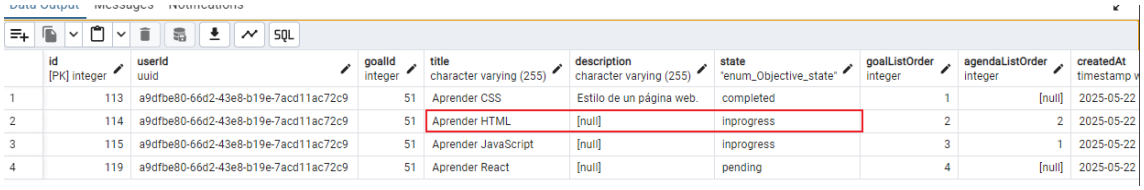



*Ilustración 91. Prueba Cambiar estado objetivo - Paso 1*



| | id [PK] integer | userId uuid | goalId integer | title character varying (255) | description character varying (255) | state "enum_Objective_state" | goalListOrder integer | agendaListOrder integer | createdAt timestamp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 113 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender CSS | Estilo de un página web. | completed | 1 | [null] | 2025-05-22 |
| 2 | 114 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender HTML | [null] | inprogress | 2 | 2 | 2025-05-22 |
| 3 | 115 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender JavaScript | [null] | inprogress | 3 | 1 | 2025-05-22 |
| 4 | 119 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender React | [null] | pending | 4 | [null] | 2025-05-22 |

*Ilustración 92. Prueba Cambiar estado objetivo - Paso 2*

Resultado de la prueba: **Superada**

### 9.3.3. Prueba – Eliminación de un objetivo

Para eliminar un objetivo hay que seleccionar el botón rojo a la derecha de cada objetivo en la ventana de edición de la meta y guardar los cambios.

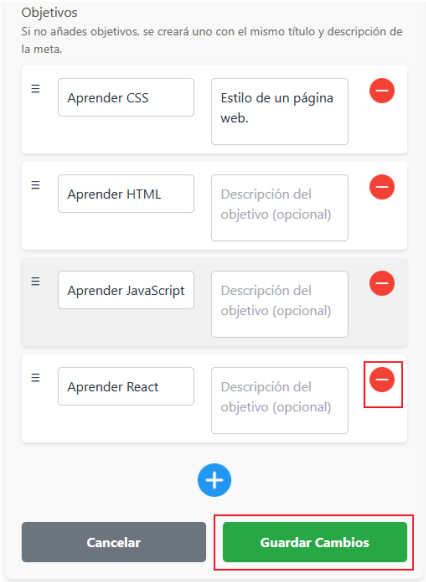



*Ilustración 93. Prueba Eliminación objetivo - Paso 1*

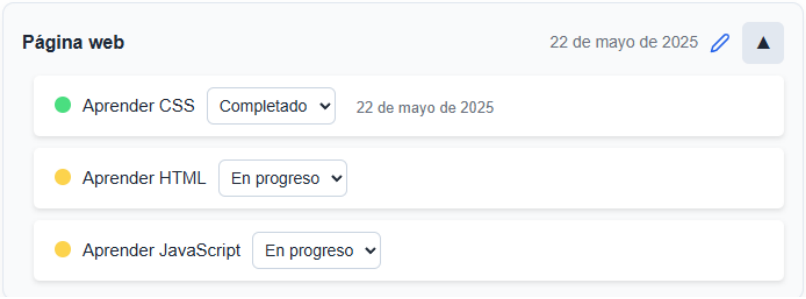
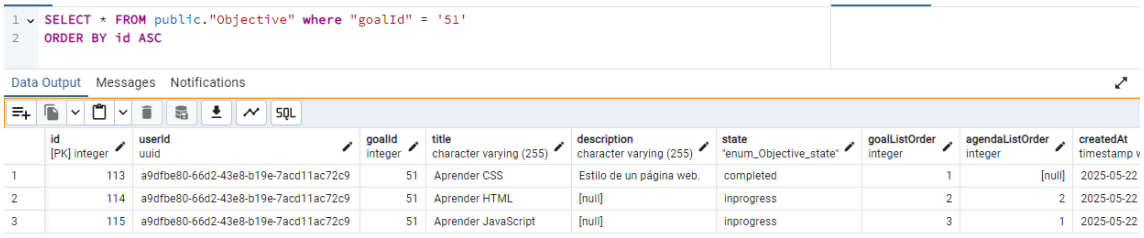



*Ilustración 94. Prueba Eliminación objetivo - Paso 2*



```
SELECT * FROM public."Objective" where "goalId" = '51'
ORDER BY id ASC
```

Data Output | Messages | Notifications

| | id [PK] integer | userId uuid | goalId integer | title character varying (255) | description character varying (255) | state "enum_Objective_state" | goalListOrder integer | agendaListOrder integer | createdAt timestamp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 113 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender CSS | Estilo de un página web. | completed | 1 | [null] | 2025-05-22 |
| 2 | 114 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender HTML | [null] | inprogress | 2 | 2 | 2025-05-22 |
| 3 | 115 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender JavaScript | [null] | inprogress | 3 | 1 | 2025-05-22 |

*Ilustración 95. Prueba Eliminación objetivo - Paso 3*

Resultado de la prueba: **Superada**

### 9.3.4. Prueba – Consulta de la fecha de finalización de un objetivo

La fecha de cuándo se finalizó un objetivo aparece al lado de cada objetivo, siempre que esté objetivo esté en estado completado.

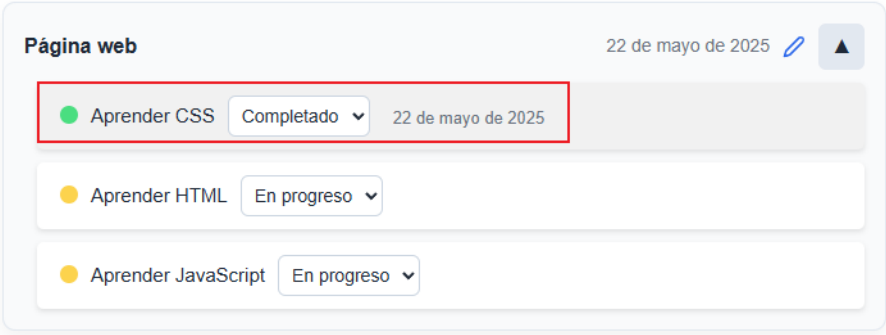



*Ilustración 96. Prueba Consulta fecha finalización objetivo - Paso 1*

Resultado de la prueba: **Superada**

### 9.3.5. Prueba – Consulta de los objetivos de una meta

Los objetivos de la meta se pueden consultar desplegando la meta gracias al botón habilitado para su fin.

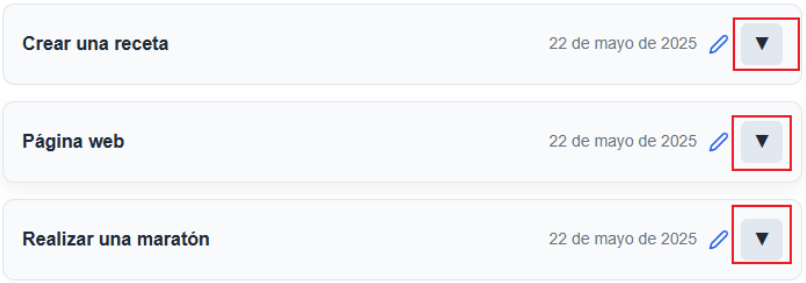



*Ilustración 97. Prueba Consulta objetivos de meta - Paso 1*

Resultado de la prueba: **Superada**

### 9.3.6. Prueba – Consulta de la agenda de objetivos en progreso

La lista de objetivos en progreso se llama 'Agenda', por lo que para que un objetivo aparezca en esta ventana, los objetivos deben estar en progreso.

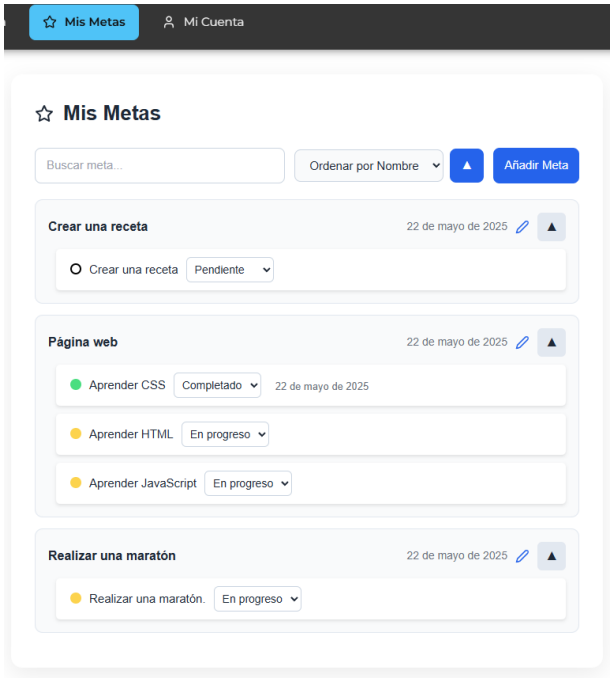



*Ilustración 98. Prueba Consulta agenda de objetivos - Paso 1*

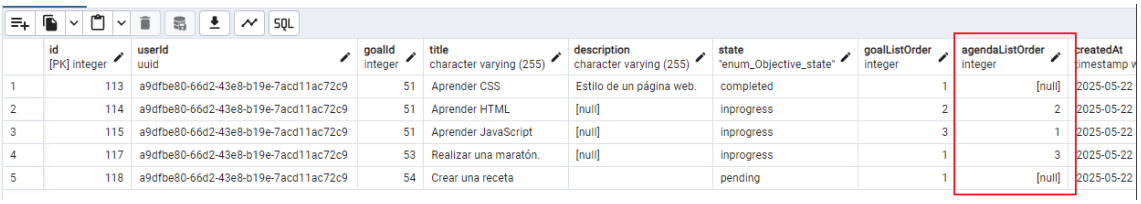
En este caso en concreto, los objetivos que deberían aparecer en el apartado 'Agenda' deben ser:

- Aprender HTML
- Aprender JavaScript
- Realizar una maratón

Comprobamos la agenda.


*Ilustración 99. Prueba Consulta agenda de objetivos - Paso 2*

Esta agenda se ordena, en un principio, en el mismo orden en el que se estableció un objetivo como en progreso. Se le asigna un número que en pruebas posteriores comprobaremos que pueden variar.



| | id [PK] integer | userId uuid | goalId integer | title character varying (255) | description character varying (255) | state "enum_Objective_state" | goalListOrder integer | agendaListOrder integer | createdAt timestamp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 113 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender CSS | Estilo de un página web. | completed | 1 | [null] | 2025-05-22 |
| 2 | 114 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender HTML | [null] | inprogress | 2 | 2 | 2025-05-22 |
| 3 | 115 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender JavaScript | [null] | inprogress | 3 | 1 | 2025-05-22 |
| 4 | 117 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 53 | Realizar una maratón. | [null] | inprogress | 1 | 3 | 2025-05-22 |
| 5 | 118 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 54 | Crear una receta | | pending | 1 | [null] | 2025-05-22 |

*Ilustración 100. Prueba Consulta agenda de objetivos - Paso 3*

Resultado de la prueba: **Superada**

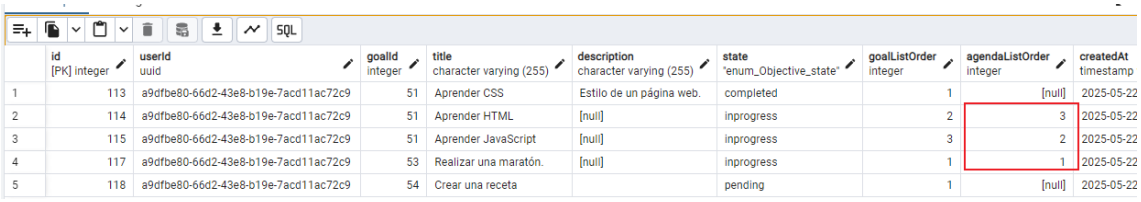
### 9.3.7. Prueba – Reordenación de los objetivos de forma personalizada en la lista de agenda

Para ordenar los objetivos en progreso en función de las preferencias del usuario es posible. Simplemente hay que arrastrar los objetivos hasta establecer el orden deseado. Este orden se establecerá en la tabla de objetivos, en el campo agendaListOrder, indicado anteriormente.

Vamos a establecer un orden distinto.


*Ilustración 101. Prueba Reordenación de objetivos - Paso 1*

Comprobamos en la tabla el cambio.



| | id [PK] integer | userId uuid | goalId integer | title character varying (255) | description character varying (255) | state "enum_Objective_state" | goalListOrder integer | agendaListOrder integer | createdAt timestamp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 113 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender CSS | Estilo de un página web. | completed | 1 | [null] | 2025-05-22 |
| 2 | 114 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender HTML | [null] | inprogress | 2 | 3 | 2025-05-22 |
| 3 | 115 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 51 | Aprender JavaScript | [null] | inprogress | 3 | 2 | 2025-05-22 |
| 4 | 117 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 53 | Realizar una maratón. | [null] | inprogress | 1 | 1 | 2025-05-22 |
| 5 | 118 | a9dfbe80-66d2-43e8-b19e-7acd11ac72c9 | 54 | Crear una receta | | pending | 1 | [null] | 2025-05-22 |

*Ilustración 102. Prueba Reordenación de objetivos - Paso 2*

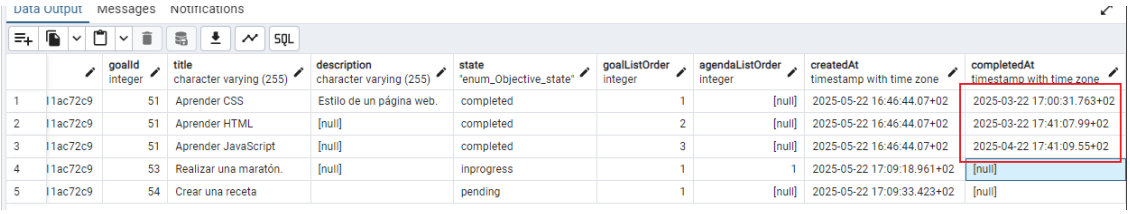
Resultado de la prueba: **Superada**

### 9.3.8. Prueba – Consulta mediante gráfica de los objetivos completados en los últimos 6 meses

Para este caso vamos a realizar una modificación por base de datos para probar que todo funciona correctamente.

Vamos a establecer dos objetivos como completados el mes de marzo y uno en abril. Por lo tanto, debería aparecer en la gráfica esta estadística.


*Ilustración 103. Prueba Consulta gráfica 6 meses atrás - Paso 1*



Data Output | Messages | Notifications

| | | goalId integer | title character varying (255) | description character varying (255) | state "enum_Objective_state" | goalListOrder integer | agendaListOrder integer | createdAt timestamp with time zone | completedAt timestamp with time zone |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 11ac72c9 | 51 | Aprender CSS | Estilo de un página web. | completed | 1 | [null] | 2025-05-22 16:46:44.07+02 | 2025-03-22 17:00:31.763+02 |
| 2 | 11ac72c9 | 51 | Aprender HTML | [null] | completed | 2 | [null] | 2025-05-22 16:46:44.07+02 | 2025-03-22 17:41:07.99+02 |
| 3 | 11ac72c9 | 51 | Aprender JavaScript | [null] | completed | 3 | [null] | 2025-05-22 16:46:44.07+02 | 2025-04-22 17:41:09.55+02 |
| 4 | 11ac72c9 | 53 | Realizar una maratón. | [null] | inprogress | 1 | 1 | 2025-05-22 17:09:18.961+02 | [null] |
| 5 | 11ac72c9 | 54 | Crear una receta | | pending | 1 | [null] | 2025-05-22 17:09:33.423+02 | [null] |

*Ilustración 104. Prueba Consulta gráfica 6 meses atrás - Paso 2*

Comprobamos la gráfica de objetivos por mes.

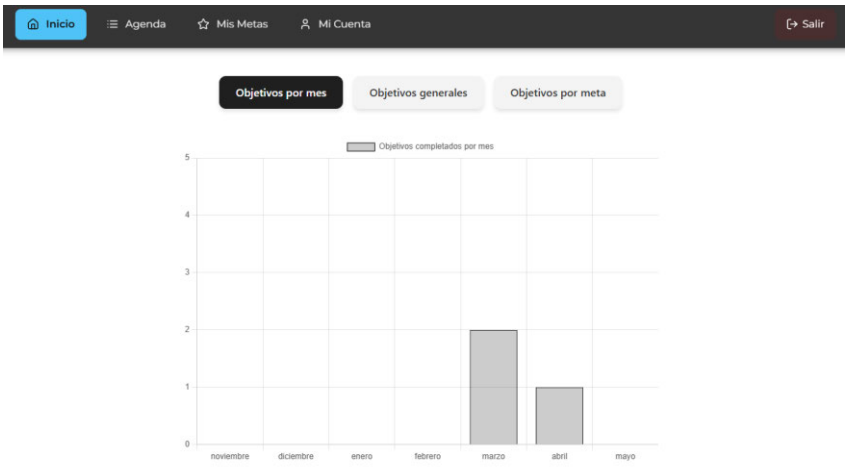



*Ilustración 105. Prueba Consulta gráfica 6 meses atrás - Paso 3*

Resultado de la prueba: **Superada**

### 9.3.9. Prueba – Consulta mediante gráfica de los objetivos completados y no completados de todas las metas

Para esta prueba debemos tener en cuenta que los objetivos no completados son aquellos que se encuentran en estado en progreso o en pendiente.

Consultamos como se encuentran los objetivos y consultamos que la gráfica sea coherente.

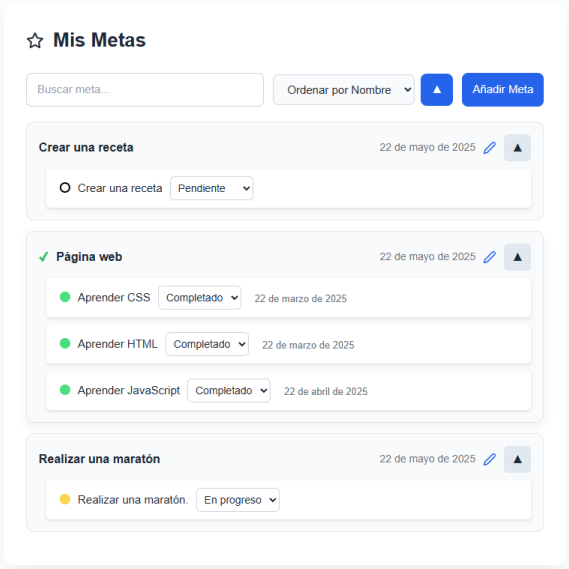



*Ilustración 106. Prueba Consulta gráfica completados/no completados - Paso 1*

Vemos que hay tres objetivos completados y dos no completados, lo que significa un 56% y un 46%, respectivamente.

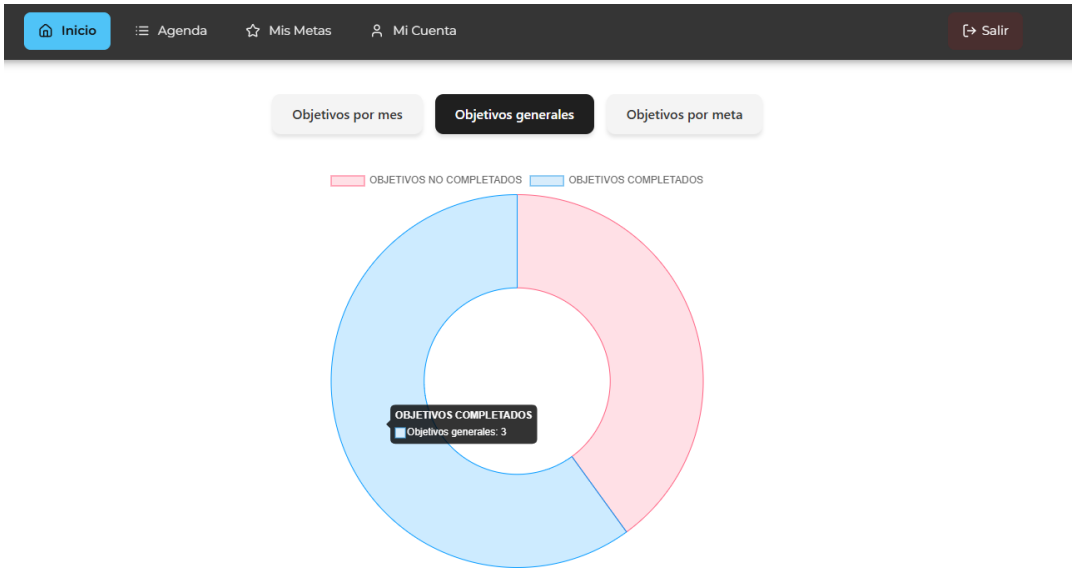



*Ilustración 107. Prueba Consulta gráfica completados/no completados - Paso 2*

Resultado de la prueba: **Superada**

## 9.4. Resumen de las pruebas realizadas

Se ha realizado las siguientes pruebas y sus resultados se resumen en la siguiente tabla.

| Pruebas | | Superada | No Superada |
|---|---|---|---|
| **Usuario** | Prueba – Registro | X | |
| | Prueba – Inicio de sesión | X | |
| | Prueba – Consulta de perfil | X | |
| | Prueba – Modificación de perfil | X | |
| | Prueba – Cambio de contraseña | X | |
| | Prueba – Eliminación de cuenta | X | |
| **Metas** | Prueba – Crear Meta con objetivos | X | |
| | Prueba – Crear Meta sin objetivos | X | |
| | Prueba – Consulta de metas | X | |
| | Prueba – Consulta del detalle de una meta | X | |
| | Prueba – Modificación de una meta | X | |
| | Prueba – Eliminación de una meta | X | |
| | Prueba – Cambio de estado de una meta | X | |
| | Prueba – Consulta de estado de la meta mediante gráficas | X | |
| | Prueba – Ordenamiento de metas por distintos filtros | X | |
| **Objetivo** | Prueba – Crear un objetivo para una meta existente | X | |
| | Prueba – Cambiar el estado de un objetivo | X | |
| | Prueba – Eliminación de un objetivo | X | |
| | Prueba – Consulta de la fecha de finalización de un objetivo | X | |
| | Prueba – Consulta de los objetivos de una meta | X | |
| | Prueba – Consulta de la agenda de objetivos en progreso | X | |
| | Prueba – Reordenación de los objetivos de forma personalizada en la lista de agenda | X | |
| | Prueba – Consulta mediante gráfica de los objetivos cumplidos en los últimos 6 meses | X | |
| | Prueba – Consulta mediante gráfica de los objetivos cumplidos y no cumplidos de todas | X | |

Tras realizar todas las pruebas de todas las funcionalidades podemos comprobar que se han superado todas ellas. El comportamiento de la página web abarca todas las funcionalidades propuestas en este proyecto.

## 10.1. Aspectos éticos

Este proyecto de gestión de metas tiene como prioridad la privacidad del usuario y la ética digital como pilares fundamentales. Toda la información personal se gestiona bajo protocolos de seguridad, usando cifrado y autenticación evitando accesos no autorizados.

El sistema no comparte ni utiliza los datos del usuario con fines comerciales ni de terceros. De la misma manera, se promueve la creación de un espacio seguro, donde los usuarios puedan desarrollar sus metas personales sin interferencias externas ni exposición de su información privada.

Además, se respetan plenamente los derechos de autor de los recursos utilizados (librerías, iconos, imágenes y tipografías).

## 10.2. Aspectos medioambientales

Aunque el objetivo principal de esta plataforma no tiene vinculación con el medioambiente, el proyecto contribuye de forma indirecta a una digitalización sostenible. Al centralizar la gestión de metas personales en un entorno digital, se reduce el uso de papel y otros recursos físicos que normalmente se han utilizado para el seguimiento de tareas.

El proyecto promueve un estilo de vida más organizado, lo que puede conllevar al usuario a tomar decisiones más conscientes y seguras en su vida personal y profesional.

## 10.3. Aspectos económicos

El sistema de gestión de metas ayuda a los usuarios a mejorar la organización de su tiempo, lo que puede traducirse en una mayor productividad y mejor toma de decisiones financieras personales. Dado que se trata de una solución digital de bajo coste y fácil mantenimiento, se presenta como una buena alternativa accesible para estudiantes y profesionales independientes que buscan herramientas de planificación sin depender de soluciones de suscripción o pago más complejas.

## 11.1. Problemas técnicos en el desarrollo

### 11.1.1. Gestión de estados en los componentes React

Uno de los principales retos fue mantener sincronizado el estado entre distintos componentes, especialmente al editar información del usuario o actualizar metas y objetivos. En algunos casos, los datos no se refrescaban correctamente tras realizar ciertas operaciones. Para solucionarlo, tuve que reorganizar el flujo de datos y que aprender a utilizar hooks (useEffect y useState, principalmente) de la forma correcta para no provocar inconsistencia en los datos.

### 11.1.2. Autenticación y persistencia de sesión

Asegurar que el usuario permanezca autenticado entre sesiones o durante la navegación fue complejo. Inicialmente, al recargar la página, la información del usuario se perdía. Para solucionarlo, se implementó un sistema basado en cookies con la opción credentials: 'include', permitiendo mantener la sesión del usuario sin necesidad de iniciar sesión repetidamente.

### 11.1.3. Comunicación con la API y manejo de errores

Manejar correctamente las repuestas del backend fue difícil, especialmente cuando existían errores de validación o cuando sucedían errores inesperados. Incorporé una gestión errores algo más robusta para que fuera personalizable. Esto facilitaba al apartado del frontend entender qué es lo que se estaba enviando o recibiendo mal.

### 11.1.4. Organización visual de la información

Mostrar los objetos de la página requirió de un motón de pruebas, ya que era necesario cuadrar colores y tamaños con tal de conseguir un estilo agradable al usuario.

### 11.1.5. Modales dinámicos con múltiples funciones

El componente 'ModalCuenta.jsx' debía tener adaptaciones tanto para cuando se quiere hacer un cambio de contraseña como para cuando se quiere eliminar la cuenta. Manejar esta lógica condicional en un solo componente hace que la calidad del código aumente, pero la dificultad aumenta consigo.

## 11.2. Desafíos personales

Tras haber tenido algo de experiencia desarrollando el backend de una página web en Java y Spring Boot, uno de los mayores desafíos fue adaptarse a un nuevo stack distinto, en este caso con Node.js, Express y React. Aunque tenía conocimientos generales del desarrollo web, el cambio de lenguaje y entorno supuso un reajuste importante en la forma de pensar y de estructurar el código.

También fue un reto aprender a manejar React de una forma más compleja, sobre todo en relación con los hooks, el control de estados de la aplicación y la separación correcta de componentes.

Otro desafío importante fue entender como funcionan las peticiones entre el frontend y backend en este nuevo entorno. En Spring Boot, muchas operaciones tenían cierta encapsulación, mientras que aquí el desarrollo es más manual y se debía construir una estructura organizada para controlar las respuestas.

Finalmente, trabajar en un proyecto con una lógica de usuario más compleja (edición de cuenta, control de metas y objetivos, interacción con el backend a tiempo real) supuso un salto respecto a proyectos anteriores. Este proyecto lo traduzco como una experiencia que me ayudó a reforzar mi autonomía, la toma de decisiones técnicas y la capacidad de investigación constante.

**Token**

Cadena única de caracteres que se utiliza para autenticar y autorizar a un usuario en una aplicación. Suele generarse al iniciar sesión y puede almacenarse en cookies o en el almacenamiento local del navegador para validar futuras peticiones sin necesidad de volver a iniciar sesión.

**useState**

Hook de React que permite declarar y gestionar variables de estado de dentro de componentes funcionales.

**useEffect**

Hook de React que permite ejecutar efectos secundarios, como peticiones de una API, después del renderizado del componente.

**API REST**

Conjunto de convenciones que permite la comunicación entre cliente y servidor mediante peticiones HTTP como GET, POST, PATCH o DELETE.

**Routing (React Router)**

Sistema que permite manejar diferentes vistas o páginas dentro de una aplicación React, basado en la URL actual.

**Hook personalizado (useAuth)**

Función reutilizable en React que encapsula lógica relacionada con la autenticación del usuario, como verificar si está logueado.

**CSS Modularizado**

Enfoque para organizar estilos utilizando archivos CSS específicos por componente, lo cual mejora la mantenibilidad y evita conflicto de clases.

**Modal**

Ventana emergente utilizada para interactuar con el usuario sin abandonar la vista actual, empleada en este proyecto para cambiar la contraseña o eliminar la cuenta.

**CRUD**

Acrónimo de Create, Read, Update, Delete. Representa las operaciones básicas sobre datos que realiza la mayoría de las aplicaciones web, como el manejo de metas y objetivos.

**Frontend**

Es la parte visual de la aplicación con la que interactúa el usuario (hecho en React).

**Backend**

Es la parte que gestiona la lógica y los datos (hecho con Express y Node.js).

**Estado global/local**

Se refiere a cómo se almacena y comparte la información entre componentes. El estado local solo afecta al componente donde se declara, mientras que el global se puede compartir.

**Gestión de formularios**

Proceso de capturar y manejar la información que el usuario introduce en inputs.

**Control de errores**

Técnica para detectar, mostrar y manejar errores que ocurren tanto en el cliente como en el servidor, con el objetivo de ofrecer una experiencia más robusta y segura.

**Enpoint**

URL específica dentro de una API que representa un recurso o una acción concreta.

## Documentación de React

https://es.react.dev/

https://es.legacy.reactjs.org/

https://es.legacy.reactjs.org/community/courses.html

https://cursoreact.dev/

https://codigofacilito.com/cursos/curso-gratis-de-react

https://developer.mozilla.org/es/docs/Learn_web_development/Core/Frameworks_libraries/React_getting_started

## Documentación Node.js

https://aprendenode.dev/

https://apuntes.de/nodejs/

https://www.freecodecamp.org/espanol/news/aprende-node-js-y-express-curso-desde-cero/

https://carlosazaustre.es/cursos/nodejs-gratis

## Documentación Express

https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction

https://www.freecodecamp.org/espanol/news/aprende-node-js-y-express-curso-desde-cero/

## Documentación Sequelize

https://sequelize.org/

https://www.npmjs.com/package/sequelize

https://www.luisllamas.es/como-usar-sequalize-con-nodejs/

## Documentación de Bases de Datos

Uso de la teoría de asignaturas como Bases de datos, Bases de datos Avanzadas y Gestión y administración de bases de datos.

## Documentación de Desarrollo Web

Uso de los conceptos adquiridos en la asignatura de Tecnologías de Desarrollo para la Web.

*Repositorio*

El repositorio del proyecto se encuentra en el siguiente link:

https://github.com/brianpardogaona/Sistema_de_Gestion_Personal

Para utilizar el contenido del repositorio se recomienda el uso de Visual Studio Code.

## Guía de instalación

1. Es necesario crear una base de datos con usuario y contraseña para conectarlo con el proyecto.

2. Instalar todas las dependencias tanto en el frontend como en el backend con:
   ***npm install***

3. Clonar el repositorio del proyecto.

4. Se debe configurar las variables de entorno de la base de datos en la siguiente ruta:
   ***Sistema_de_Gestion_Personal\backend\.env***

5. Para ejecutar el backend es necesario encontrarse sobre la carpeta:
   ***Sistema_de_Gestion_Personal\backend***
   Y ejecutar el siguiente comando:
   ***npm run dev***

6. Para ejecutar el frontend es necesario encontrarse sobre la carpeta:
   ***Sistema_de_Gestion_Personal\frontend***
   Y ejecutar el siguiente comando:
   ***npm run dev***

7. Finalmente, acceder al link que te proporciona la terminal tras ejecutar el frontend.

## *Agradecimientos*

*A mi familia, que siempre han sido mi principal apoyo emocional durante mi vida.*

*A mi madre, que siempre ha estado dispuesta a ayudarme de forma incondicional.*

*A mis amigos más cercanos, que siempre han colaborado en mis proyectos de vida sin importar que parecieran imposibles.*

*Por último, a todos los compañeros y amigos que he conocido durante mi trayecto en mi etapa universitaria.*