UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación para entrenamientos de personas con diversidad funcional "Sanroma Entrena Salud" - Backend

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Oliver Cortés, Francisco Javier

Tutor/a: Albert Albiol, Manuela

Cotutor/a externo: WOJCIECHOWSKI, ADAM

CURSO ACADÉMICO: 2022/2023

Agradecimientos

Agradezco de corazón a todas las personas que me han brindado su apoyo incondicional a lo largo de mi Trabajo de Fin de Grado. Han sido pilares fundamentales en este emocionante y desafiante camino hacia la culminación de mis estudios.

En primer lugar, quiero agradecer a mi tutora, Manuela Albert Albiol, por su guía experta y dedicación durante todo el proceso de desarrollo y redacción de este trabajo. Gracias por brindarme su tiempo, paciencia y por creer en mi capacidad para llevar a cabo este TFG.

También deseo expresar mi agradecimiento a mi compañero de TFG, [nombre del compañero]. Trabajar juntos en este proyecto ha sido una experiencia enriquecedora. Gracias por ser un excelente compañero de equipo y por contribuir a que nuestro TFG progresara y creciese.

En cuanto vida personal, debo agradecer a mi familia, por siempre apoyarme y motivarme a seguir adelante cuando más lo necesitaba y cuando todo se hacía cuesta arriba. Finalmente, agradecer a Álvaro y a Antonio, por enseñarme a relajarme y tomar también tiempo para mí, y a mis amistades de este Erasmus, por sus palabras de aliento y por escucharme sin importar el momento que fuese.

Resumen

El Trabajo Final de Grado (TFG) consiste en el desarrollo de una aplicación de soporte al trabajo del profesional Luis Sanroma. Luis Sanroma está especializado en entrenamientos para personas con enfermedades degenerativas. La aplicación permitirá a este profesional crear rutinas personalizadas para sus clientes de una manera más sencilla, y a los clientes acceder a sus entrenos de una forma intuitiva y organizada. La aplicación será una aplicación web responsive, con dos perfiles de usuario: el administrador/entrenador que subirá ejercicios, tendrá chats con sus clientes, seguimiento de clientes, etc., y sus clientes, que consultarán los ejercicios, verán logros, tendrán chats con el entrenador, etc... Este TFG es un trabajo en colaboración con el TFG del alumno Adrian Hernandez Monterroso. El presente TFG se centrará en la parte del Back-end de la plataforma web "Sanroma Entrena Salud" mientras que el TFG de Adrian Hernandez Monterroso se centrará en la parte de Front-end. La tecnología que se utilizará para dicho Back-end será javascript usando Nodejs y Express, y para la base de datos usaremos MongoDB.

Palabras clave: Entrenamientos personalizados; plataforma web; enfermedades degenerativas; back-end.

Abstract

The Final Degree Project (TFG) consists of the development of an application to support the work of the professional Luis Sanroma. Luis Sanroma specializes in training for people with degenerative diseases. The application will allow this professional to create personalized routines for his clients in a simpler way, and the clients to access their workouts in an intuitive and organized way. The application will be a responsive web application, with two user profiles: the administrator/trainer who will upload exercises, have chats with his clients, follow up with clients, etc., and his clients, who will consult the exercises, see achievements, have chats with the trainer, etc... This TFG is a collaborative work with the TFG of the student Adrian Hernandez Monterroso. The present TFG will focus on the Back-end part of the web platform "Sanroma Entrena Salud" while the TFG of Adrian Hernandez Monterroso will focus on the Front-end part. The technology to be used for the Back-end will be javascript using Nodejs and Express, and for the database we will use MongoDB.

Keywords : Personalized trainings; web platform; degenerative diseases; back-end

Resum

El Treball Final de Grau (TFG) consisteix en el desenvolupament d'una aplicació de suport al treball del professional Luis Sanroma. Luis Sanroma està especialitzat en entrenaments per a persones amb malalties degeneratives. L'aplicació permetrà a aquest professional crear rutines personalitzades per als seus clients d'una manera més senzilla, i als clients accedir als seus entrenaments d'una forma intuïtiva i organitzada. L'aplicació serà una aplicació web responsive, amb dos perfils d'usuari: l'administrador/entrenador que pujarà exercicis, tindrà xats amb els seus clients, seguiment de clients, etc., i els seus clients, que consultaran els exercicis, veuran assoliments, tindran xats amb l'entrenador, etc... Aquest TFG és un treball en col·laboració amb el TFG de l'alumne Adrián Hernández Monterroso. El present TFG se centrarà en la part del Back-end de la plataforma web "Sanroma Entrena Salut" mentre que el TFG d'Adrián Hernández Monterroso se centrarà en la part de Front-end. La tecnologia que s'utilitzarà per a aquest Back-end serà javascript usant Nodejs i Express, i per a la base de dades usarem MongoDB.

Paraules clau: Entrenaments personalitzats; plataforma web; malalties degeneratives; back-end.

Tabla de contenidos

Agradecimientos 3
1. Introducción 11
1.1 Motivación 11
1.2 Objetivos 11
1.3 Impacto esperado 11
1.4 Metodología 12
1.5 Estructura 13
1.6 Colaboraciones 14
2. Situación actual de la tecnología 15
2.1 Aplicaciones para la gestión de entrenamientos 15
2.1.1 Nike Training Club 15
2.1.2 MyFitnessPal 16
2.1.3 JEFIT 17
2.2 Análisis de la situación actual de la tecnología 17
2.3 Propuesta 18
3. Análisis del problema 19
3.1 Requisitos 19
3.2 Casos de uso 19
3.3 Modelo de dominio 23
3.4 Análisis de soluciones posibles 24
3.5 Plan de trabajo 25
3.6 Presupuesto 26
4. Diseño de la solución 27
4.1 Arquitectura del sistema 27
4.2 Diseño detallado 28
4.3 Tecnología utilizada 30
5. Desarrollo de la solución 33
5.1 Planificación y primeros pasos 33
5.2 Creación de la aplicación node.js y la BBDD 33
5.3 Programación 34
6. Pruebas 41

6.1 Pruebas del backend 41
6.2 Pruebas de funcionalidad de la aplicación 51
7. Conclusiones 60
7.1 Análisis personal del desarrollo del TFG 60
7.2 Relación del trabajo desarrollado con los estudios cursados 60
8. Trabajos futuros 62
9. Referencias 63
10. Glosario 65
Anexo 1: ODS 68

Tabla de Ilustraciones

Ilustración 1 Herramienta para Scrum - Meister Task 13
Ilustración 2: Aplicación de entrenamiento Nike Training Club 15
Ilustración 3 Aplicación de entrenamiento MyFitnessPal 16
Ilustración 4 Aplicación de entrenamiento JEFIT 17
Ilustración 5 Caso de uso del entrenador 20
Ilustración 6 Caso de uso del cliente 23
Ilustración 7 Modelo de dominio 24
Ilustración 8 Arquitectura 27
Ilustración 9 Diagrama de la BBDD 29
Ilustración 10 MongoDB Atlas 33

Tabla de Carpetas

Carpetas 1 39

Tabla de Ilustraciones

Tabla 1 Presupuesto 26

Tabla de Código

Código 1 34
Código 2 35
Código 3 35
Código 4 35
Código 5 35
Código 6 35
Código 7 36
Código 8 36
Código 9 36
Código 10 37
Código 11 37
Código 12 37
Código 13 38
Código 14 38
Código 15 40

Tabla de Cronogramas

Cronograma 1 Plan de trabajo previsto 25
Cronograma 2 Plan de trabajo final 26

Tabla de Terminal

Terminal 1 34
Terminal 2 34
Terminal 3 34

Tabla de Pruebas

Pruebas 1 41
Pruebas 2 42
Pruebas 3 43
Pruebas 4 43
Pruebas 5 44
Pruebas 6 44
Pruebas 7 45
Pruebas 8 46
Pruebas 9 47
Pruebas 10 47
Pruebas 11 48
Pruebas 12 48
Pruebas 13 49
Pruebas 14 49
Pruebas 15 50
Pruebas 16 51
Pruebas 17 52
Pruebas 18 52
Pruebas 19 53
Pruebas 20 53
Pruebas 21 54
Pruebas 22 54
Pruebas 23 55
Pruebas 24 55
Pruebas 25 56
Pruebas 26 56
Pruebas 27 56
Pruebas 28 57
Pruebas 29 57
Pruebas 30 58
Pruebas 31 58
Pruebas 32 58
Pruebas 33 59
Pruebas 34 59

1. Introducción

En el mundo actual, el envejecimiento de la población y el aumento de enfermedades degenerativas representan desafíos significativos para la sociedad. Las personas de la tercera edad o personas que padecen enfermedades degenerativas a menudo enfrentan limitaciones físicas y necesitan una atención especializada para mantener una buena calidad de vida.

En este contexto, las aplicaciones móviles han surgido como una herramienta valiosa para brindar apoyo y atención personalizada a este grupo de personas. La propuesta de este Trabajo Final de Grado (TFG) tiene como objetivo proporcionar una plataforma accesible y personalizada para la gestión de entrenamientos físicos a personas de la tercera edad y con enfermedades degenerativas.

Esta plataforma permitirá a un profesional de este campo gestionar de forma más eficiente los entrenamientos que prepara para sus clientes. Esta aplicación pretende cubrir las necesidades tanto de este profesional como las de sus clientes.

1.1 Motivación

El profesional Luis Sanroma, fisiólogo, se dedica a entrenar a personas con problemas o limitaciones físicas. Para ello, prepara entrenamientos de acuerdo con sus condiciones y necesidades, haciendo de cada paciente un caso único. Puestos en contacto con este profesional, mi compañero Adrián Hernández Monterroso y yo quisimos poner nuestro granito de arena y ayudar a Luis a gestionar estos entrenamientos de forma más sencilla y eficiente. Esto le podría ayudar a entrenar a más personas al mismo tiempo, manteniendo la calidad del entrenamiento.

Con esta motivación, han surgido dos TFGs, el de Adrián Hernández Monterroso, que se centra en la parte de la aplicación relacionada con la interfaz de usuario (frontend), y el presente TFG, que se centra en la parte lógica (backend).

1.2 Objetivos

El objetivo de este TFG es desarrollar la parte de backend de una aplicación web para manejar el entrenamiento físico de múltiples personas de una manera sencilla y que ahorre tiempo al administrador y entrenador de dichas personas.

Los subobjetivos enmarcados dentro de este objetivo principal son:

- Construir una API REST escalable.
- Garantizar la seguridad del backend
- Diseñar una base de datos con MongoDB escalable

1.3 Impacto esperado

Desde el punto de vista del entrenador, esta aplicación ayudará a:

- Planificar y programar mejor los entrenamientos de un mayor número de clientes (por lo que podría conseguir aumentar los ingresos al poder tener más)
- Permitir un mayor seguimiento de los pacientes
- Organizar los entrenamientos
- Mejorar la comunicación entre el cliente y el entrenador

A su vez, desde el punto de vista del cliente, esta aplicación servirá como una forma organizada de monitorizar su progreso entrenando, mejorar la comunicación con el entrenador en el caso que no entrene de manera presencial con el mismo y tener un entrenamiento personalizado en cualquier momento.

1.4 Metodología

Para el desarrollo del proyecto hemos aplicado la metodología Scrum (*What is Scrum?, s.f.)* [8] Scrum es un enfoque ágil de gestión de proyectos que se utiliza comúnmente en el desarrollo de software. Se basa en equipos autoorganizados y multidisciplinarios que trabajan en ciclos cortos de tiempo llamados "sprints". Cada sprint tiene una duración fija.

El marco de trabajo Scrum se compone de roles, eventos y artefactos clave:

Roles:

- Product Owner: es responsable de definir los objetivos del proyecto y priorizar los requisitos del producto.
- Scrum Master: es el facilitador del equipo Scrum, asegura que se sigan los principios y prácticas de Scrum.
- Equipo de Desarrollo: es el grupo de personas encargadas de desarrollar el producto.

Eventos:

- Sprint Planning: reunión donde se planifica el trabajo para el próximo sprint.
- Daily Scrum: reunión diaria de 15 minutos donde el equipo sincroniza su trabajo y planifica el día.
- Sprint Review: reunión al final de cada sprint para revisar el trabajo completado y recibir retroalimentación.
- Sprint Retrospective: reunión para analizar el sprint anterior y encontrar mejoras para el próximo.

Artefactos:

- Product Backlog: lista priorizada de requisitos y funcionalidades del producto.
- Sprint Backlog: conjunto de elementos seleccionados del Product Backlog que se trabajarán durante el sprint.
- Incremento: versión del producto que se desarrolla al final de cada sprint.

Scrum promueve la transparencia, la colaboración y la adaptación continua. A través de la retroalimentación frecuente y la inspección y adaptación constantes, los equipos

Scrum pueden responder rápidamente a los cambios y entregar valor de manera incremental en cada sprint.

Llevando los componentes de Scrum a nuestro caso, nuestro Product Owner sería nuestro cliente Luis. Atendiendo a quienes formarían el equipo de desarrollo, lo formaríamos yo y mi compañero Adrián y el Scrum Master, debido a que ambos debíamos pertenecer al equipo de desarrollo, también debía ser uno de nosotros, y finalmente nos decantamos que yo lo sería.

La Ilustración 1 muestra una imagen de un sprint en la que podemos ver el backlog, el Sprint Backlog (pestaña Abiertas) y el incremento (pestaña Listo).

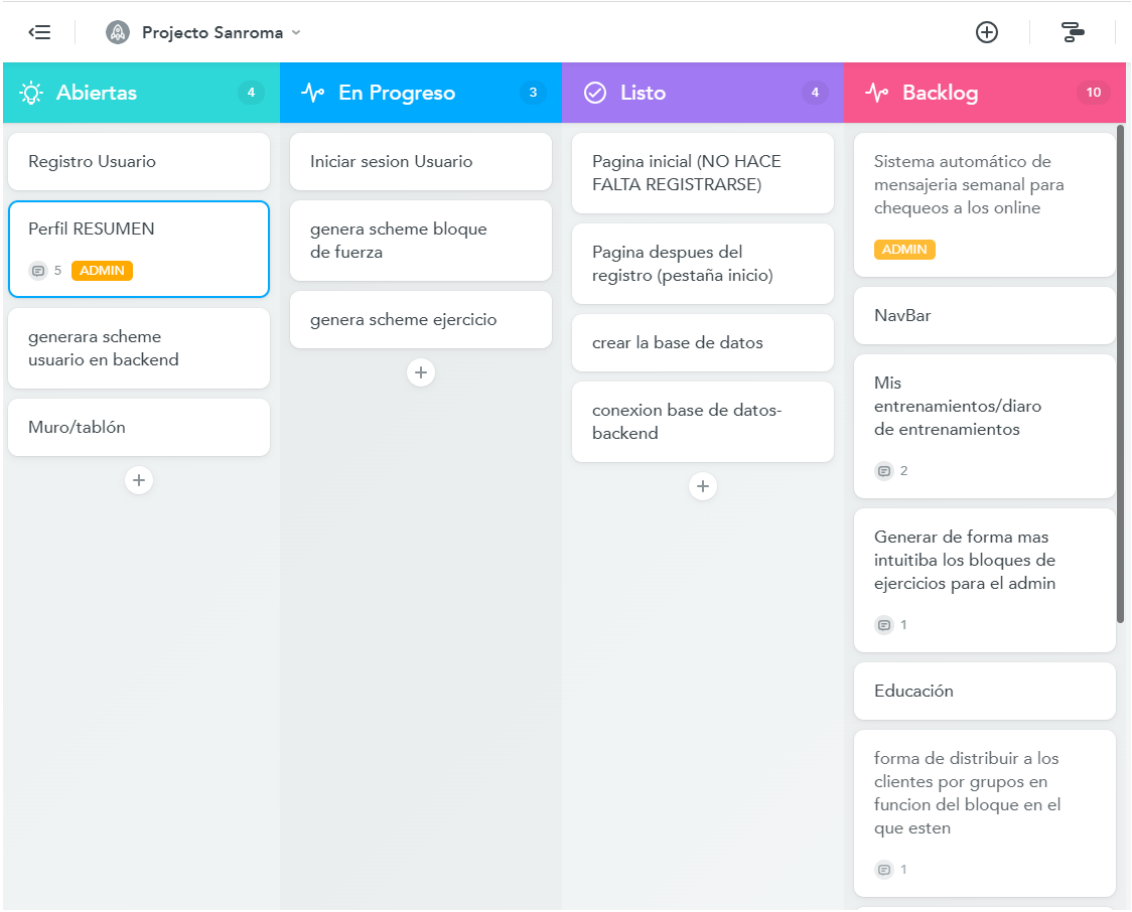


Ilustración 1 Herramienta para Scrum - Meister Task

Nuestros sprints fueron de 3 semanas. Al final de cada sprint, revisábamos que hubiésemos cumplido con el trabajo del sprint y planificábamos en que estaríamos trabajando en el siguiente Sprint. Los Daily Scrums se realizaban cada 4 días, poniendo en común las cosas que habíamos hecho y hablando sobre necesidades que tuviésemos sobre el otro campo para nuestro trabajo. Nuestro Sprint Retrospective se realizaba el día siguiente al Sprint Review, que se realizaba una vez se acababan esas 3 semanas.

1.5 Estructura

A continuación, se detalla la estructura de la memoria.

- Capítulo 1. Introducción, se tratarán temas como la motivación detrás del proyecto, la metodología que se usado o los objetivos que se desean conseguir con el mismo.
- Capítulo 2. Situación actual de la tecnología, se comparará esta aplicación con otras aplicaciones del mercado las cuales cumplirían una función similar a la que estamos ofreciendo en la aplicación a desarrollar.
- Capítulo 3. Análisis del problema, estudiaremos el problema y veremos las soluciones que podemos usar para solucionarlo.
- Capítulo 4. Diseño de la solución, se explicará cómo se va a organizar la solución para obtener el objetivo perseguido
- Capítulo 5. Desarrollo de la solución, se explicará cómo se ha desarrollado la solución detallando el proceso de desarrollo.
- Capítulo 6. mostraremos las pruebas que se han realizado tanto al backend como a la aplicación en su totalidad.
- Capítulo 7. Conclusión, se mencionará lo aprendido con este TFG, como conecta con los ODS y con lo aprendido durante los años de estudio.
- Capítulo 8. Futuros trabajos, se mencionará en que se gustaría trabajar para mejorar la aplicación en un futuro
- Capítulo 9. Referencias, se nombrará de donde se ha obtenido la información necesaria para hacer este TFG

1.6 Colaboraciones

En colaboración con este TFG se encuentra el trabajo del estudiante Adrián Hernández Monterroso, enfocado en la parte front-end de la aplicación.

2. Situación actual de la tecnología

En este capítulo se analizan algunas aplicaciones que existen en el mercado para entrenar y mantener entrenamientos personalizados. Se presentan las conclusiones de este análisis y se introduce brevemente la solución propuesta en este TFG.

2.1 Aplicaciones para la gestión de entrenamientos

A continuación, se describen 3 aplicaciones bastantes conocidas en el mundo del entrenamiento físico.

2.1.1 Nike Training Club

Ilustración 2: Aplicación de entrenamiento Nike Training Club

Nike Training Club (NTC) *(ilustración 2)* es la Plataforma de fitness de Nike (*Nike Training Club App. Home Workouts & More.*, s. f.).[1] [2] Nike presenta una gama de entrenamientos, programas y recursos para que los clientes puedan alcanzar sus metas de entrenamiento y físicas. La aplicación está diseñada tanto para principiantes como para personas más experimentadas. Es una aplicación gratuita para IOS y Android, aunque tiene una versión de pago con una tarifa mensual.

En la aplicación hay una amplia biblioteca de entrenamientos en video que cubren varias disciplinas, como entrenamiento de fuerza, entrenamiento de alta intensidad o yoga. Entrenadores profesionales ejemplifican y dan instrucciones sobre cómo hacer dichos ejercicios de forma correcta

Además de los entrenamientos en video, la aplicación también ofrece planes de entrenamiento personalizados que se adaptan a las necesidades y objetivos específicos de los usuarios. Se pueden establecer metas, como perder peso, ganar fuerza o mejorar la

resistencia. La aplicación proporciona un programa de entrenamiento hecho con el objetivo de alcanzar esas metas.

Nike Training Club también ofrece funciones adicionales, como seguimiento de actividad física, recordatorios de entrenamiento, desafíos y recompensas, y la posibilidad de conectarte con otros usuarios para compartir tus logros y mantener la motivación.

2.1.2 MyFitnessPal

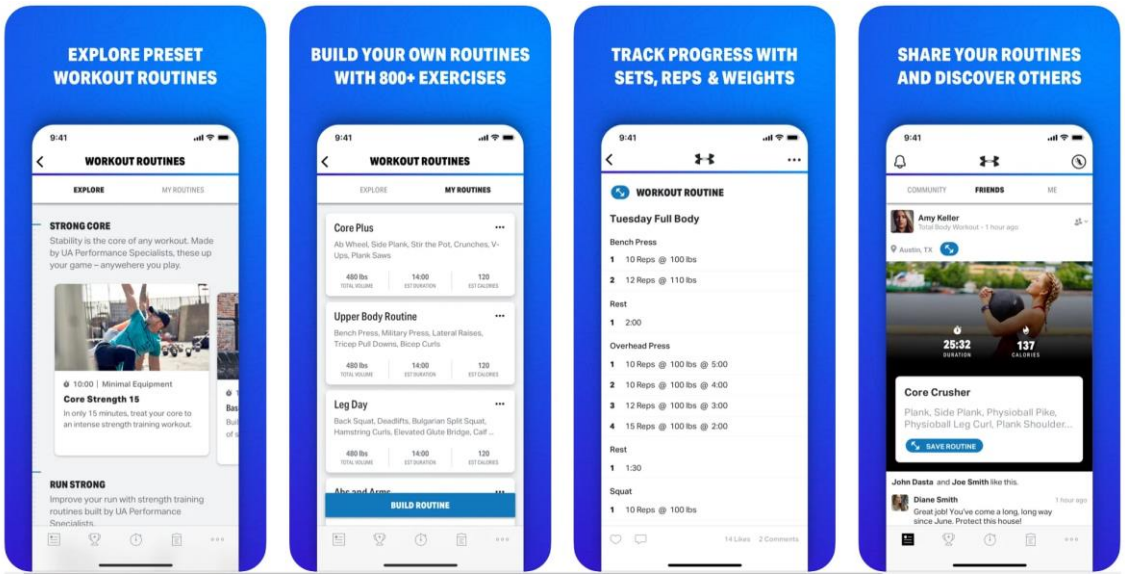


Ilustración 3 Aplicación de entrenamiento MyFitnessPal

MyFitnessPal *(ilustración 3)* es una aplicación popular de seguimiento de la alimentación y la actividad física ofrecida por la marca Under Armour (*MyFitnessPal | MyFitnessPal*, s. f.). [3] [4] Permite a los usuarios registrar su ingesta de alimentos y realizar un seguimiento de sus calorías, macronutrientes y micronutrientes. También ofrece herramientas para registrar y analizar el ejercicio, establecer metas de peso y seguimiento del progreso. La aplicación es gratuita para iOS y Android con la posibilidad de ampliar las funcionalidades con una suscripción premium.

Centrándonos en la parte que sería parecida a nuestra aplicación, es decir, la parte de seguimiento y registro de entrenamientos, MyFitnessPal permite registrar el ejercicio físico realizado, incluyendo actividades como correr, caminar, levantar pesas, yoga, entre otros. Puedes estimar las calorías quemadas y realizar un seguimiento de tu actividad física.

Puedes establecer metas de peso y la aplicación te proporcionará un plan diario de calorías para ayudarte a alcanzar ese objetivo. También puedes realizar un seguimiento del progreso a lo largo del tiempo, ya sea a través de gráficos o registros de peso.

La aplicación cuenta con una comunidad en línea donde los usuarios pueden conectarse para comentar recetas, dar consejos o compartir mensajes a otros usuarios sobre sus objetivos.

2.1.3 JEFIT

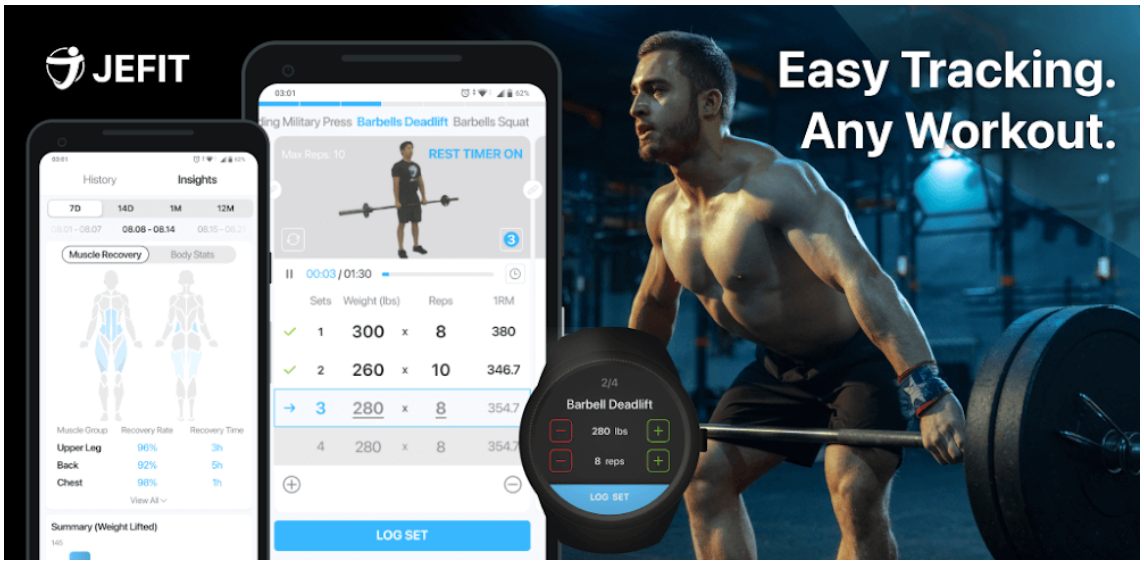


Ilustración 4 Aplicación de entrenamiento JEFIT

JEFIT *(ilustración 4)* es una aplicación móvil diseñada para ayudar a las personas a realizar un seguimiento de sus entrenamientos y lograr sus objetivos de acondicionamiento físico (Jefit, Inc., 2023). [5] [6] La aplicación está disponible tanto para dispositivos iOS como Android y ofrece una variedad de características y herramientas útiles. Al igual que las otras 2 aplicaciones antes mencionadas, JEFIT tiene una versión totalmente gratuita y otra de pago ampliando y desbloqueando varias funcionalidades que tiene.

JEFIT permite a los usuarios crear y personalizar sus propios programas de entrenamiento, realizar un seguimiento de tus entrenamientos al registrar los ejercicios, repeticiones, series, pesos y tiempo de descanso, cuenta con una extensa biblioteca de ejercicios con demostraciones visuales y descripciones detalladas, permite registrar y hacer un seguimiento de tus mediciones corporales, como peso, porcentaje de grasa corporal y medidas, cuenta con una comunidad activa de usuarios donde puedes conectarte con otros entusiastas del fitness, compartir tus logros, obtener motivación y descubrir nuevos programas de entrenamiento y ofrece una variedad de programas predefinidos para diferentes objetivos, niveles de experiencia y estilos de entrenamiento.

2.2 Análisis de la situación actual de la tecnología

Las aplicaciones que hemos analizado carecen de un responsable detrás de la misma, es decir, todas ellas prepararan un plan de entreno automáticamente a partir de una serie de parámetros como peso o altura utilizando algoritmos genéricos.

Sin embargo, si algo hemos aprendido con Luis Sanroma, es que no es únicamente el peso o la altura lo que hará que el entreno sea efectivo y adecuado para una persona. La flexibilidad, conocimiento en los movimientos o fisiología juegan un papel importante en saber si un entreno es propio para una persona o no.

Por mucho que se quiera agrupar a las personas por similitudes, cada una es diferente. Por ello, los entrenamientos que ofrecen estas aplicaciones podrían llegar a ser bastante básicos, repetitivos, simples y no tan personalizados como se venden en sus anuncios o en la propia aplicación.

2.3 Propuesta

Como hemos visto, todas las aplicaciones que hemos mencionado son aplicaciones en las que con datos muy básicos se generan entrenamientos automáticos para el usuario. No se tiene en cuenta algunos datos importantes del usuario, como la flexibilidad, ni tampoco existe comunicación con el entrenador. Además, estas aplicaciones se basan en el cliente, y no sirven para que un entrenador planifique entrenamientos.

La aplicación propuesta en este TFG tiene detrás un entrenador personal. Es una herramienta tanto para clientes como para entrenadores. Es una aplicación de entreno con un entrenador personal pendiente del usuario y su progreso. La diferencia de esta aplicación con las aplicaciones analizadas es que, en todo momento, un profesional del entrenamiento será la persona que personalice los entrenamientos y proponga los ejercicios y movimientos aptos para el usuario.

3. Análisis del problema

En este capítulo nos centraremos en analizar el problema que se quiere resolver con la aplicación web. Se van a presentar los requisitos de la aplicación, los casos de usos, el modelo de dominio, las características de la solución propuesta, el plan de trabajo y el presupuesto.

3.1 Requisitos

Para la elicitación de requisitos se organizaron varias reuniones con el profesional Luis Sanroma, y se definieron los siguientes requisitos:

- El entrenador deberá poder hacer entrenamientos para sus clientes.
- El usuario podrá ver los entrenamientos creados por su entrenador.
- El entrenador deberá tener un calendario donde administrar los entrenamientos de sus clientes.
- Los clientes podrán ver un histórico de sus entrenamientos.
- Existirá un tablón para gente no registrada para que se puedan ver posts del entrenador sobre diversos temas.
- El entrenador podrá organizar pagos.
- Los clientes podrán gestionar por sus pagos.
- El usuario deberá acceder a la aplicación de forma fácil y sin que sus datos se vean comprometidos.

Además, la aplicación web deberá garantizar la seguridad de los datos de clientes.

3.2 Casos de uso

Un caso de uso de una aplicación de software describe una interacción específica entre un usuario y el sistema. Proporciona un escenario detallado que describe cómo un usuario interactúa con la aplicación para lograr un objetivo particular. Lo usaremos para comprender y especificar los requisitos funcionales de la aplicación. A continuación, mostraremos los casos de uso sobre las actividades que deberán ser capaces de hacer cada uno de los actores.

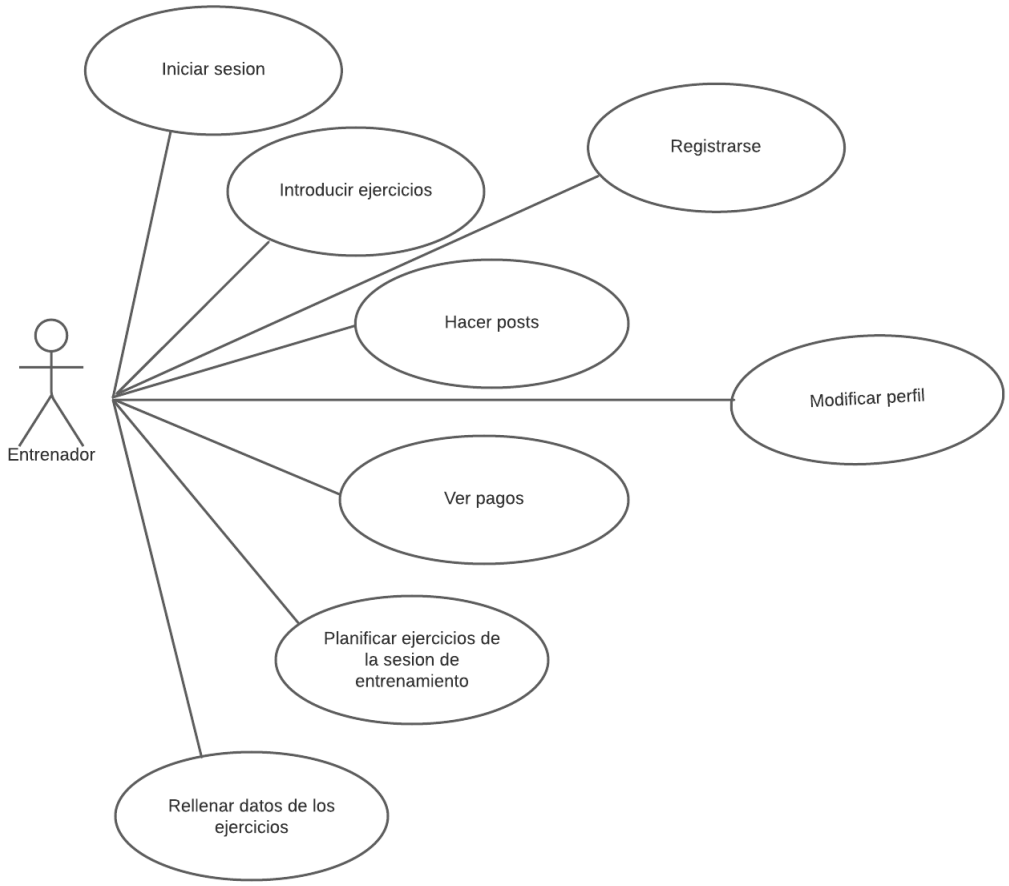


Ilustración 5 Caso de uso del entrenador

En la *ilustración 5* se puede ver el diagrama de casos de uso del entrenador. A continuación, haremos la especificación de los casos de uso de este diagrama.

CASO DE USO: Registrarse
DESCRIPCION: El entrenador introducirá sus datos para poder crear una cuenta con la que poder acceder al sistema
ACTORES: Entrenador, cliente
PRECONDICION:
RELACIONES:
ESCENARIO PRINCIPAL:

1. El actor introducirá sus datos personales
2. El sistema verificara que el formato con el que se introduce es correcto
3. El actor tendrá una cuenta con la que acceder al sistema

CASO DE USO: Iniciar sesión
DESCRIPCION: El entrenador introducirá sus credenciales para poder iniciar sesión en el sistema
ACTORES: Entrenador, cliente
PRECONDICION: Registrarse
RELACIONES:
ESCENARIO PRINCIPAL:

1. El actor introducirá su email y contraseña
2. El sistema verificara que son correctos
3. El actor podrá acceder al sistema

CASO DE USO: Introduce ejercicios
DESCRIPCION: El entrenador deberá tener un apartado donde podrá guardar ejercicios en la base de datos para usarlos posteriormente en la planificación de los entrenamientos.
ACTORES: Entrenador
PRECONDICION: Iniciar sesión
RELACIONES:
ESCENARIO PRINCIPAL:

1. El entrenador deberá ir a la sección de ejercicios
2. El entrenador añadirá un ejercicio en un grupo de ejercicios
3. El sistema guardara dicho ejercicio en la base de datos,

CASO DE USO: Hacer posts
DESCRIPCION: El entrenador tendrá la opción de hacer post sobre diversos temas para que los usuarios los vean.
ACTORES: Entrenador
PRECONDICION: Iniciar sesión
RELACIONES:
ESCENARIO PRINCIPAL:

1. El entrenador deberá ir a la sección de creación de post
2. El entrenador introducirá los datos del post (titulo, descripción, etc)
3. El sistema guardara dicho post en la base de datos
4. El post será visible por el resto de los usuarios

CASO DE USO: Ver pagos
DESCRIPCION: El entrenador deberá tener un apartado donde podrá guardar los clientes que han pagado, escribir los que quedan por pagar y se podrá visualizar la cantidad del pago
ACTORES: Entrenador
PRECONDICION: Iniciar sesión
RELACIONES:
ESCENARIO PRINCIPAL:

1. El entrenador deberá ir a la sección de pagos
2. El entrenador podrá ver un calendario de pagos y podrá editarlo para insertar pagos efectuados y los que faltan
3. El sistema guardara dichas ediciones en el calendario

CASO DE USO: Planificar ejercicios de una sesión de entrenamiento
DESCRIPCION: El entrenador deberá poder editar mediante un apartado los ejercicios que tendrá que hacer un cliente en una sesión de entrenamiento
ACTORES: Entrenador
PRECONDICION: Iniciar sesión
RELACIONES:
ESCENARIO PRINCIPAL:

1. El entrenador deberá ir a la sección de crear entreno
2. El entrenador seleccionara un cliente
3. El entrenador insertara una serie de ejercicios
4. El sistema guardara dichos ejercicios

CASO DE USO: Rellenar datos de los ejercicios
DESCRIPCION: El entrenador podrá editar la intensidad de los ejercicios del entrenamiento de un cliente (repeticiones, series, carga…)
ACTORES: Entrenador
PRECONDICION: Planificar ejercicios de una sesión de entrenamiento
RELACIONES:
ESCENARIO PRINCIPAL:

1. El entrenador deberá elegir un ejercicio de los introducidos a una sesión de entrenamiento.
2. El entrenador introducirá los datos de intensidad de estos.
3. El sistema guardara dichos datos en la base de datos,

CASO DE USO: Modificar perfil
DESCRIPCION: El actor deberá tener un apartado donde podrá editar y modificar los datos que introdujo a la hora de crear la cuenta.
ACTORES: Entrenador, cliente
PRECONDICION: Iniciar sesión
RELACIONES:
ESCENARIO PRINCIPAL:

1. El actor deberá ir a la sección de su cuenta
2. El actor modificara sus datos
3. El sistema guardara dichos cambios

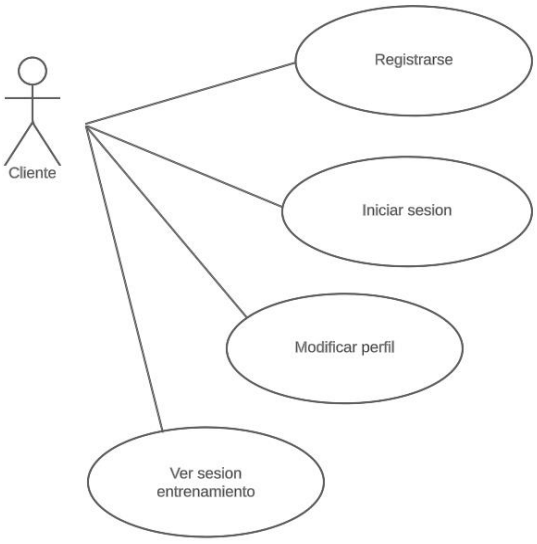


Ilustración 6 Caso de uso del cliente

En la *ilustración 6* se puede ver el diagrama de casos de uso del cliente. El cliente comparte casos de uso con el entrenador, por lo que no repetiremos sus especificaciones de caso de uso.

CASO DE USO: Ver sesión entrenamiento
DESCRIPCION: El cliente deberá ser capaz de ver la sesión entrenamiento que le ha agenciado el entrenador
ACTORES: Cliente
PRECONDICION: Iniciar sesión, Rellenar datos de los ejercicios
RELACIONES:
ESCENARIO PRINCIPAL:

1. El cliente deberá ir a la sección de training
2. El sistema cargara los datos de los ejercicios que debe hacer el cliente
3. El cliente podrá visualizar los ejercicios

3.3 Modelo de dominio

El modelo de dominio es una estructura que captura la comprensión del mundo real en el cual la aplicación opera. Proporciona una visión clara de cómo se organizan y se relacionan los conceptos en el dominio, lo que nos ayudara a comprender mejore los requisitos y las interacciones necesarias para la implementación en la aplicación.

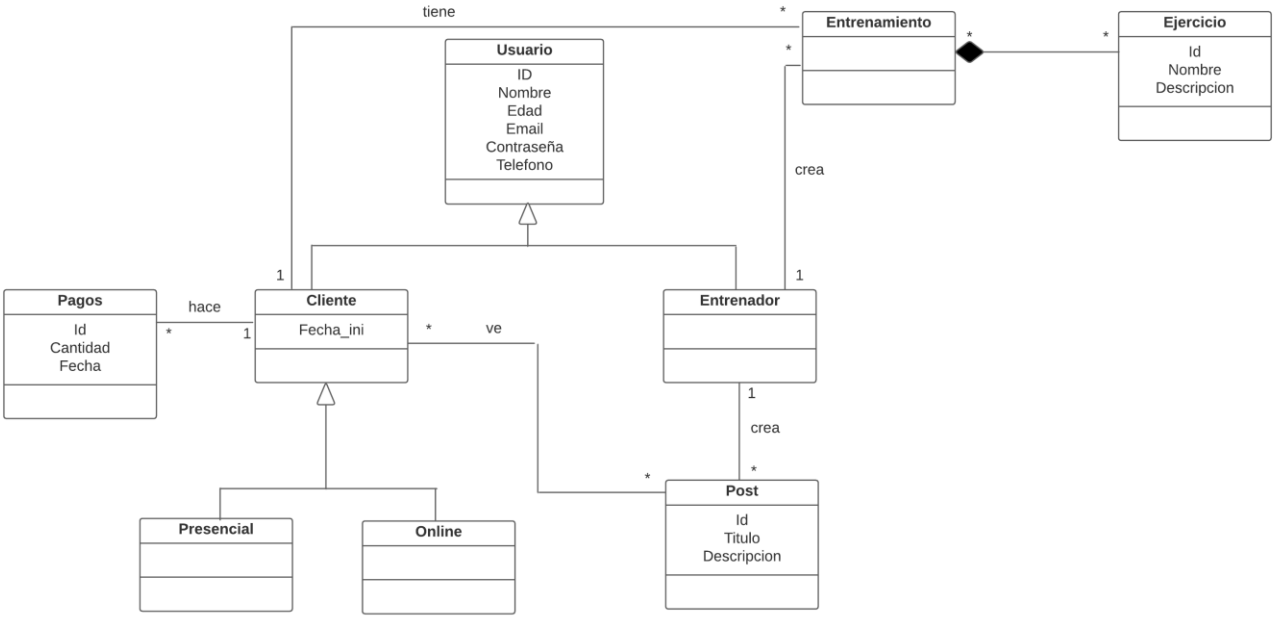


Ilustración 7 Modelo de dominio

En la *ilustración 7* se muestra el modelo de dominio de nuestra aplicación. Como se puede ver existen dos tipos de usuarios: los clientes y los entrenadores. A su vez, los clientes podrán ser tanto presenciales como online. Los clientes hacen pagos al entrenador para seguir entrenando. Mientras, el entrenador hace posts, que son publicaciones con información, videos o fotos en los que se encuentra información relacionada con el entrenamiento, los cuales pueden ver los clientes. Finalmente, se puede ver que el entrenador crea entrenamientos conformados por ejercicios para los clientes.

3.4 Análisis de soluciones posibles

Con estos requisitos propuestos, valoramos principalmente 3 posibles soluciones para la creación de la aplicación:

1. Aplicación de escritorio. Esta posibilidad era la que ayudaría más a Luis Sanroma a su gestión. Sería la más cómoda debido a que trabajaría con una aplicación instalada en su ordenador. Sin embargo, no parecía una opción tan cómoda para los clientes, ya que deberían ver los entrenos desde el ordenador, y como posiblemente entrenarían en un gimnasio, no es demasiado cómodo. Por lo que pensamos que no era la mejor idea.[10]
2. Aplicación de smartphone. También sonaba atractiva, pero con esta opción pasaba exactamente lo contrario que con la aplicación de escritorio. Los clientes tendrían en su smartphone toda la información de los entrenamientos, lo que ayudaría a llevar dicha información a cualquier lugar al que quisieran entrenar. Pero, para Luis Sanroma hubiese sido muchísimo más tedioso y molesto el tener que administrar todo a través de un teléfono. Por esto, descartamos también esta posibilidad. [12]

3. Aplicación web con un diseño “responsive”. Esta aplicación serviría tanto para acceder a ella desde teléfonos, como para acceder a ella desde ordenadores personales. Esta opción permite tanto al entrenador como al cliente acceder de manera más cómoda. [11]

Finalmente decidimos hacer una aplicación web de entrenamientos personales. Una aplicación web es un programa o software que se ejecuta en un navegador web y se accede a través de internet, además de no necesitar instalación. Además de esto, construiremos un diseño responsive, es decir, que se adapte a los distintos dispositivos desde los que se acceda, proporcionando una experiencia óptima a los usuarios.

En el presente TFG nos centraremos en el desarrollo únicamente del backend. Este backend estará compuesto de una API REST conectada a una base de datos. Para garantizar la seguridad de la aplicación, se construirá un middleware que proteja la API REST y la base de datos.

3.5 Plan de trabajo

En este apartado, se presenta la estimación de tiempo que se realizó de las tareas que conlleva el desarrollo de la aplicación, y la distribución de tiempo que costó realmente realizarlas.

Cronograma 1 Plan de trabajo previsto

Para realizar esta estimación de tiempo, se ha organizado el trabajo en 4 tareas que hemos considerado las más relevantes:

- Redacción de requisitos
- Despliegue de BBDD y API Rest
- Programación
- Pruebas

En el *cronograma 1*, podemos ver el tiempo que se previó para las diferentes tareas del desarrollo del backend. Como podemos comprobar, se planteó que la tarea que más tiempo ocupara fuera la de programación, ya que en ella es donde se desarrollará todo el código del backend.

El plan de trabajo será el siguiente: se empezará con las reuniones con el cliente para recabar los requisitos. Una vez obtenida la primera lista de requisitos se ordenarán los objetos del backlog y del primer sprint. Seguidamente, se montaría la base de datos y se conectaría con la API REST. Se iniciaría la programación y se terminaría haciendo las pruebas correspondientes para ver que todo funciona correctamente.

Cronograma 2 Plan de trabajo final

En el *cronograma 2*, podemos ver el tiempo real que se ha invertido para hacer el backend. La redacción de requisitos, el despliegue y las pruebas acabaron siendo realizadas en menos tiempo del que se preveía. Sin embargo, la programación, debido a problemas que surgieron durante el desarrollo, hicieron que requiriera más tiempo del planeado.

Durante todo el desarrollo, se tuvieron numerosas reuniones con el cliente. Se le ha ido mostrando el estado de la app web**,** esperando su feedback, con cosas que deberían cambiarse, eliminarse, mejorarse, e incluso optando a nuevas ideas para el desarrollo.

Para la gestión de tareas, usamos la aplicación web Meister task, la cual nos ayudó a organizar los requisitos, objetos del backlog y Sprints.

3.6 Presupuesto

Para estudiar el presupuesto, se han analizado y presupuestado todos los elementos que se necesitan para desarrollar el backend. En la tabla 1 se muestra el presupuesto.

Ordenador	600 €
Teclado	40 €
Ratón	30 €
Micrófono	30 €
Cámara	90 €
Servidor	25€/mes
Internet	30€/mes
Electricidad	100€/mes

Tabla 1 Presupuesto

Como se puede ver en la *tabla 1* el mayor gasto ha sido el del dispositivo personal utilizado para la programación. Después, se han incluido las herramientas que se han necesitado para llevar a cabo el backend, ya fuese el teclado y ratón para la programación o el micrófono y la cámara para las reuniones. Además, se han incluido las suscripciones al servidor y también los gastos en la electricidad que se ha necesitado para que el ordenador funcione y el internet para la búsqueda de información, tener contacto con el servidor donde guardar la información (del cual se paga una suscripción para tener un mayor almacenamiento en el) o poder guardar el código en la nube.

4. Diseño de la solución

En este capítulo, explicaremos la arquitectura con la que hemos trabajado para realizar la aplicación, además de explicar las diferentes capas y componentes explicando las tecnologías usadas para ello.

4.1 Arquitectura del sistema

En este punto se detalla la arquitectura que hemos elegido para la aplicación y los motivos que han llevado a su elección. La arquitectura aplicada es una arquitectura en capas, la cual consiste en dividir el sistema en capas lógicas, donde cada capa tiene una responsabilidad específica. [13] La razón por la que elegimos esta arquitectura es porque permite separar de una manera sencilla las tareas que se llevarían a cabo en los dos TFGs involucrados en el desarrollo de la aplicación.

Además, puesto que en los TFGs se va a trabajar con tecnologías con las que los estudiantes no somos familiares, queríamos que las capas se pudieran probar de manera independiente, sin perjudicar el funcionamiento de otra. Además, si en un futuro la aplicación escalara y tuviésemos que añadir más funcionalidades con nuevas tecnologías, esta arquitectura nos ayudaría a facilitar todo ese proceso.

A continuación, podremos ver las capas de la que está formada nuestra aplicación y explicaremos cada una de ellas:

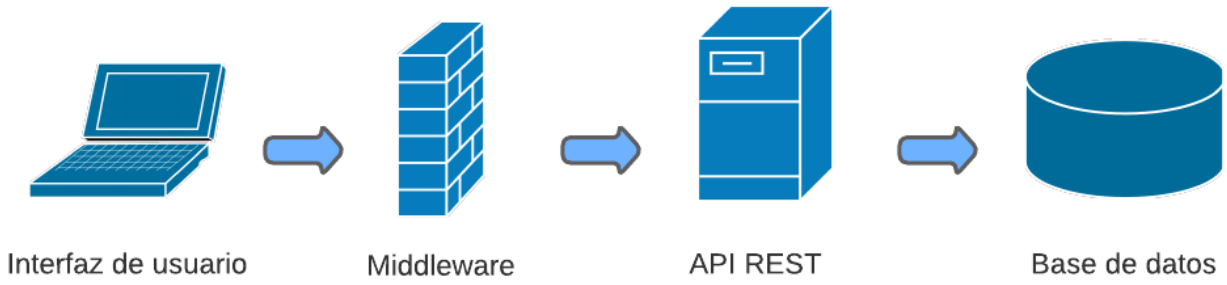


Ilustración 8 Arquitectura

Interfaz gráfica de usuario (GUI): Esta capa contiene los componentes que permiten a los usuarios interactuar con el sistema a través de elementos visuales, como ventanas, botones, menús y otros elementos gráficos. Esta capa pertenece a la parte del frontend, desarrollada en el TFG de Adrián Hernández Monterroso.

Middleware: La capa middleware de la aplicación es una capa previa a que los datos sean gestionados por la API. En esta capa se comprueban los derechos de los que dispondría cada petición (comprobar si la petición está hecha por un usuario registrado entre otras cosas). Para esta capa se ha usado la tecnología del JWT, explicada en un apartado posterior. Esta capa pertenece al trabajo que se explica en este TFG.

API (Application Programming Interface): En esta capa utilizamos una API REST (Representational State Transfer), en la cual se realizarán los métodos con los que se recibirán objetos de la base de datos, se eliminaran, crearan, o modificaran. En esta capa se encuentra la lógica y el procesamiento de datos, Esta capa pertenece al trabajo que se explica en este TFG**.** [15]

Base de datos: La capa de la base de datos se encarga de gestionar el almacenamiento y la recuperación de los datos utilizados por la aplicación. Esta capa se comunica con las demás usando la API REST. Esta capa pertenece al trabajo que se explica en este TFG.

4.2 Diseño detallado

En este apartado explicaremos con más detalle las tres capas que conciernen a este TFG.

Middleware. Las peticiones a la API REST pasan por esta capa, a excepción de la petición de registrarse o de iniciar sesión. El funcionamiento será el siguiente:

1. Un usuario se registrará en la aplicación y posteriormente iniciará sesión en ella.
2. Una vez iniciada la sesión, al usuario tendrá un token gracias al JWT
3. Cada vez que el frontend mande una petición al backend, tendrá que enviar el token que se genera cuando inicia sesión
4. El middleware verificara el token, si es correcto, la petición se hará sin mayor problema, si no, dará un error

Con esto gestionamos y aseguramos que la base de datos no sea modificada ni se pueda acceder a ella si no se tienen permisos.

Para proteger la key con la que generamos los tokens, la creamos dentro de un archivo ENV, que sirven para almacenar variables de entorno, las cuales contienen información sensible o de configuración

API REST. Sirve para sistemas de comunicación en red, especialmente en el contexto de servicios web, como podría ser nuestra aplicación. Con la API REST los datos se modelan como objetos o entidades y se accede a ellos a través de URLs (Uniform Resource Locators) o URIs (Uniform Resource Identifiers). Los métodos HTTP, como GET, POST, PUT y DELETE, se utilizan para realizar operaciones en estos recursos. El formato en el que enviamos y recibimos estos datos es JSON (JavaScript Object Notation). Además, la API REST puede hacer que el servidor devuelva códigos de estado HTTP para indicar el resultado de una solicitud, como 200 OK para una solicitud exitosa o 404 Not Found si el recurso no existe.

Se encarga de las peticiones a la base de datos. La comunicación se realiza mediante URLs. En el frontend, se usa una herramienta llamada Axios, mediante la cual, introduciendo una cabecera (en la que se incluirá el token), una función (GET, PUT, DELETE, POST), un cuerpo (que contendrá la información que queramos transmitir a la base de datos) y una url, se podrán hacer las peticiones.

La API REST es la que se conecta a la base de datos MongoDB. Las credenciales de esta se introducen también en un archivo ENV. La API REST contiene todos los modelos que deben crearse en el backend y, mediante ellas, podemos modificar dichas colecciones en la base de datos. Esto lo podemos hacer con mayor facilidad gracias a la herramienta de mongoose, que ayuda a crear dichas colecciones en la base de datos.

Después contiene los endpoints donde podemos diferenciar si lo que queremos hacer es crear, eliminar, modificar u obtener los objetos de la base de datos.

Base de datos. Para la base de datos hemos usado Mongo DB, un sistema de base de datos NoSQL. A diferencia de los sistemas de base de datos SQL tradicionales que utilizan tablas y filas para organizar los datos, MongoDB utiliza un modelo de almacenamiento de documentos. La *ilustración 9* muestra el diagrama de la base de datos.

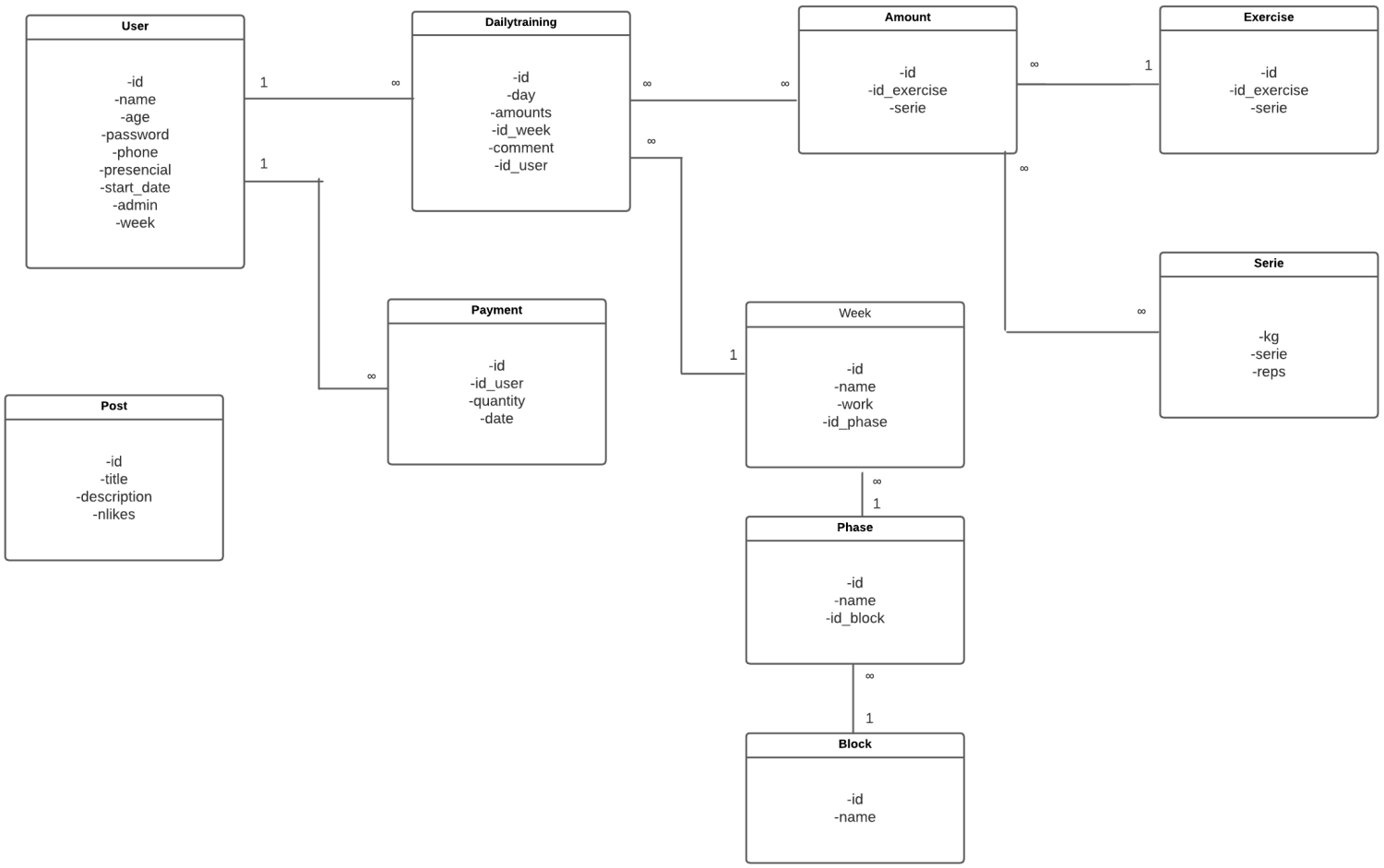


Ilustración 9 Diagrama de la BBDD

Las colecciones de la base de datos son las siguientes:

- User: La colección User contiene la información relevante de los usuarios de la aplicación, ya sean clientes online o presenciales o administradores
- Dailytraining: Esta colección contiene la información de los entrenamientos diarios de cada cliente.
- Amount: Esta colección es necesaria para guardar la intensidad del entrenamiento (repeticiones, carga, series) debido a que esta información entre series puede variar. Por ejemplo, haciendo el ejercicio de sentadilla, es posible que en la primera serie hagamos 8 repeticiones, en la siguiente 10 y en la ultima 5, y lo mismo con la carga del ejercicio
- Exercise: Contiene la información de los ejercicios que se pueden agenciar a los clientes
- Serie: Contiene la carga, número de serie y repeticiones de cada serie
- Week, Phase y Block: Estas colecciones son colecciones que ayudan a los entrenadores a ver por qué estado de un programa de entrenamiento están (si acaban de empezarlo, están por la mitad o ya lo están acabando)
- Payment: Esta colección contiene la información de pago de los clientes
- Post: Esta colección existe para guardar los posts que pueden crear los administradores y ver los clientes

Hay que destacar en este esquema de la base de datos, que MongoDB tiene una característica especial, y es que en las colecciones se aceptan arrays (en las bases de datos SQL no se pueden definir). Esta característica no es posible de representar en el diagrama de la base de datos, por lo que a continuación explicaremos donde se encuentran dichos arrays:

- La colección Amount contiene un array de series, la cual hemos representado como otro objeto, sin embargo, si miramos la base de datos, no existirá la colección Serie, sino que el array serie guardará los atributos de Serie (kg, serie y reps).
- La colección Routine contiene un array de amounts, esta vez sí hemos tenido que crear la colección Amount debido a que se deben agenciar las series a un ejercicio, y esta colección serviría como ese punto de unión.

Además, al hacer la colección serie en un array, nos ahorraríamos la cantidad de JSON que se generarían en ella por cada entrenamiento de cada usuario.

4.3 Tecnología utilizada

La tecnología utilizada según el componente es la siguiente:

Middleware. Para el Middleware se utiliza la siguiente tecnología:

- JWT (JSON Web Token) es un estándar para la emisión de tokens seguros en forma de objetos JSON, utilizados para autenticación y autorización en aplicaciones web y servicios API. [18]Tienen tres partes separadas por puntos:

    - Header (encabezado): Contiene información sobre el tipo de token y el algoritmo de firma utilizado.
    - Payload (carga útil): Contiene los datos que se desean transmitir.
    - Signature (firma): Es la firma digital creada a partir del header, payload y una clave secreta. Permite verificar la autenticidad del token y garantizar que no haya sido modificado durante la transmisión. Nuestra clave para generar la Signature la guardaremos en un archivo ENV.
- Dotenv es una librería de software utilizada en el desarrollo de aplicaciones web y de software para cargar variables de entorno desde archivos de configuración. Guardaremos las variables de entorno en un archivo. env dentro de la aplicación. [16]

API REST. Para la API REST se utiliza la siguiente tecnología:

- NodeJS es un entorno de ejecución de código JavaScript en el lado del servidor. Proporciona un rendimiento escalable y eficiente, además que con npm hay acceso a una amplia gama de módulos y bibliotecas a través de npm. [17]
- Bcrypt es una función de hashing criptográfica diseñada específicamente para almacenar contraseñas de forma segura. El propósito principal de Bcrypt es mitigar el riesgo asociado con el almacenamiento de contraseñas en sistemas informáticos. Este hashea las contraseñas antes de guardarlas en la base de datos convirtiéndolas en un valor ininteligible. [19]
- CORS es un mecanismo de seguridad utilizado en los navegadores web para permitir que un servidor pueda enviar recursos a un dominio diferente al que originó la solicitud. Con esto el servidor especifica qué orígenes están permitidos y qué tipos de solicitudes se pueden realizar.[20]
- Express es un framework de desarrollo web para NodeJS para crear aplicaciones web y APIs de manera eficiente. Es muy conocido debido a su simplicidad, flexibilidad y rendimiento.[21]
- Mongoose es una biblioteca de modelado de objetos para Node.js que nos permite trabajar con MongoDB y que proporciona una solución basada en esquemas para modelar los datos de una aplicación.[22]
- Nodemon proporciona una forma conveniente de reiniciar automáticamente la aplicación Node.js cada vez que se detectan cambios en los archivos del proyecto. Esto evita tener que reiniciar manualmente la aplicación después de cada modificación de código, lo que agiliza el proceso de desarrollo. [23]
- Postman es una herramienta de colaboración y desarrollo de API que permite a los desarrolladores probar, documentar y compartir las API de manera más eficiente. En el caso de esta aplicación, utilizamos Postman para hacer las comprobaciones de las peticiones a la API REST. [24]

Base de datos. Para la BBDD se utiliza la siguiente tecnología:

- MongoDB, como hemos dicho antes es un sistema de base de datos NoSQL de código abierto (almacenamiento de documentos). Los datos se almacenan en

forma de documentos BSON (Binary JSON), que son similares a los documentos JSON, pero en una representación binaria. Algunas de las razones por las que MongoDB parece atractivo es porque es muy escalable, tienen una alta tolerancia a fallos debido a sus mecanismos de replicación y tiene numerosas bibliotecas para distintos lenguajes de programación. [14]

5. Desarrollo de la solución

En este capítulo vamos a explicar cómo ha sido desarrollar la aplicación web.

5.1 Planificación y primeros pasos

En primer lugar, se tuvieron numerosas reuniones con nuestro cliente. En estas reuniones se aclararían los requisitos que la aplicación debería cumplir, aparte de plazos y de detalles que le gustarían al cliente que tuviesen en su aplicación. Al estar trabajando con Scrum, se metieron todos los requisitos en el backlog y se les dieron las mayores prioridades. Las siguientes reuniones con el cliente a lo largo del resto del proceso de la aplicación se sucedieron cada dos semanas.

Una vez aclaradas las prioridades y antes de empezar a programar, se debía crear la base de datos y la API REST. Para la base de datos se tenía buscar un servidor donde alojarla. Finalmente, y después de barajar varias opciones, fue elegida la opción alojarla en los servidores de la propia empresa de MongoDB, debido a que, durante nuestra producción, buscamos un servidor que ofreciera un plan gratuito, ya fuera limitado de tiempo o capacidad, y eso es exactamente lo que ofrecía la empresa con su servicio MongoDB ATLAS. Además, si un futuro fuera a escalar mucho la aplicación únicamente habría que contratar uno de sus servicios de pago para tener más espacio.

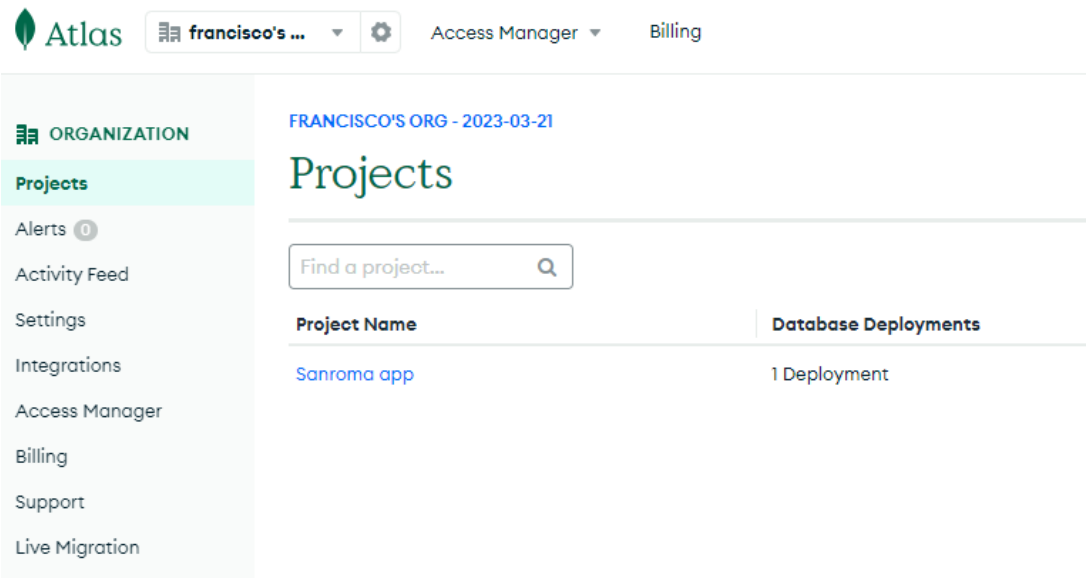


Ilustración 10 MongoDB Atlas

5.2 Creación de la aplicación node.js y la BBDD

Para crear la aplicación se creó una nueva aplicación javascript y se usó el gestor de paquetes npm para instalar todas las dependencias que usaríamos para nuestra aplicación. [15] Para crear la aplicación con node únicamente se debía escribir el código mostrado en *Terminal 1*.

```
PS C:\Users\Usuario\Desktop\sanroma\backend_sanroma> node app.js
```

Terminal 1

Y posterior a ello únicamente se usaría npm para instalar las dependencias, como podemos ver *Terminal 2*.

```
PS C:\Users\Usuario\Desktop\sanroma\backend_sanroma> npm install mongoose nodemon
```

Terminal 2

Para lanzar el backend lo único que se debe hacer es lanzar un script que llama a nodemon, el cual nos ayudará a modificar el backend y no tener que reiniciarlo cada vez que se cambia. El script tendrá el nombre de start y solo se tendría que llamar al gestor de paquetes npm y usar el script para inicializar el backend.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "nodemon app.js"
},
```

1

Código 1

```
PS C:\Users\Usuario\Desktop\sanroma\backend_sanroma> npm start
```

Terminal 3

Una vez ya se tenía creada la aplicación y la base de datos, lo único que faltaba era conectar ambas cosas. Para ello, se usaron las herramientas de dotenv y mongoose. Con mongoose resulta muy fácil la conexión de la base de datos, ya que únicamente usando la función connect() e introduciendo la URI de la base de datos, se dispone de la conexión entre ambas. Sin embargo, colocar la URI directamente en la función no es muy seguro, ya que queda muy expuesta la misma para el resto del mundo, por lo que se usó dotenv para guardar la URI en una variable de entorno que la protegiese.

5.3 Programación

Una vez montada toda la estructura de lo que necesita nuestro backend, se empezó a programar los modelos de las colecciones de nuestra base de datos. El código 2 muestra un ejemplo de modelo de una de las colecciones que queríamos crear en la base de datos. Podemos ver sus atributos, que serían user, date y amount y el nombre que tendría la colección en la base de datos, el cual seria “payment”

```
const mongoose = require("mongoose");

const Payment = new mongoose.Schema({
    user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "user",
        required: true
    },
    date: {
        type: String,
        required: true
    },
    amount: {
        type: Number,
        required: true
    }

})

module.exports = mongoose.model("payment",Payment);
```

Código 2

Gracias a Nodemon, se pueden hacer modificaciones en la base de datos sin tener que reiniciar el backend, ya que este lo auto reinicia cada vez que se guardan cambios.

Seguidamente, se programaron los endpoints. Empezamos con los de usuario, para crear, modificar, eliminar y obtener a los usuarios de la base de datos. Para ello, lo único que debíamos poner para hacer cada petición era indicar el método que se iba a usar. *(Código 3, 4, 5, 6).*

```
router.post("/create",middleware.authenticateToken, async (req, res
```

Código 3

```
router.get("/getbyid",middleware.authenticateToken,async(req, res)
```

Código 4

```
router.delete("/deletebyid",middleware.authenticateToken,async(req,
```

Código 5

```
router.put("/changebyid",middleware.authenticateToken,async(req, re
```

Código 6

En este punto haciendo los endpoints del primer modelo, se obtuvieron numerosos errores. El más destacable fue debido a que se necesitaba que los métodos fueran asíncronos, para que Node pudiese hacer el manejo de múltiples solicitudes concurrentes y no quedara bloqueado. Al estar utilizando la última versión de mongoose, siendo esta la 7.0.2, los métodos find() que ayudaban a buscar en la base de datos no aceptaban callbacks, sino que se necesitaban poner las funciones como asíncronas. Con esto se solucionaba el problema *(Código 7)*.

```
router.get("/allusers", async(req, res) =>{
    user.find({})
    .populate({
        path: 'week',
        populate:{
            path: 'phase',
            populate: {path: 'block'}
        }})
        .then(result => res.status(200).json({ result }))
        .catch(error => res.status(500).json({msg: error}))

});
```

Código 7

Para hacer endpoints que buscaran por título, nombre o id, hacemos que el parámetro que se quiere buscar se introduzca en la url como una query. En el apartado de las pruebas mostraremos como deberá ser una url que haga una búsqueda según alguna condición. En *Código 8* podemos ver lo que debíamos escribir en el código para hacer la búsqueda.

```
post.findOne({ _id: req.query.id })
```

Código 8

El nombre que se colocó después de “req.query” es el nombre del parámetro que se coloca en la url. Lo que esta antes del “req.query” es el nombre del parámetro en la base de datos.

Una vez creados los endpoints del usuario, se usó Postman para comprobar que funcionaban correctamente, y si, todos daban los resultados esperados. Estos necesitaban que se guardaran los modelos en una constante, la cual se llamaría para hacer las distintas acciones en la base de datos como encontrar un objeto o modificarlo.

```
const week = require ('../../model/trainingGroup/week.js')


router.post("/create", async (req, res) => {
    week.create(req.body)
        .then(result => res.status(200).json({ result }))
        .catch((error) => res.status(500).json({msg:  error }))
});
```

Código 9

A la hora de conectarlo con el frontend, se produjo un error debido al CORS, ya que no se había tenido en cuenta. Esto se produjo debido a que se estaba intentando realizar una petición desde otro origen. Para solucionarlo, únicamente se tuvo que instalar la extensión de CORS y autorizar el origen desde el que se estaba intentando acceder a la API REST. Con esto ya se solucionó el problema y el frontend podría recibir los objetos del backend. En el *Código 10* podemos ver lo que se escribió.

```
app.use(cors({
  origin: 'http://localhost:5173'
}))
```

Código 10

Cabe destacar que MongoDB ofrece el método populate() que permite recuperar automáticamente los documentos referenciados completos y rellenar los campos de referencia con esos documentos. Esto se logra mediante la resolución de la referencia y la recuperación de los documentos relacionados en una consulta adicional a la base de datos. Esto se incluyó en la API REST para ayudar al frontend a obtener los datos de objetos referenciados *(Código 10).*

```
const Phase = new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    block: {
        type:  mongoose.Schema.Types.ObjectId, ref: "block"
    },
    objetive: {
        type: String
        }
```

Código 11

Una vez se crearon todos los endpoint, se vio que la aplicación no era segura del todo. Para solucionar esto se decidió hashear las contraseñas de los usuarios y crear una API.

Primero, se usó la herramienta de bcrypt para las contraseñas, la cual convertía en un string irreconocible. Este string se guardaba en la base de datos. También se podía usar la herramienta para compararla con la contraseña correcta cuando el usuario se fuese a logear. *(Código 12)*

```
const passwordhashed = await bcrypt.hash(req.body.password,10)
const user = req.body
user.password = passwordhashed
user.create(user)
```

Código 12

Seguidamente usamos JWT para crear un middleware que protegiese la base de datos. Se creaba un token cada vez que un usuario iniciara sesión en la aplicación, con el que podría acceder a las peticiones de la base de datos. Para proteger los métodos solo se tenía que usar el algoritmo del middleware para comprobar antes de cada función que el token que está usando el usuario que se le ha dado al iniciar sesión es correcto para hacer la petición.

```
const jwt = require('jsonwebtoken')

function authenticateToken(req, res, next){
    const  authHeader = req.headers['authorization']
    const token =authHeader && authHeader.split(' ')[1]
    if (token == null) return res.sendStatus(401)

    jwt.verify(token,process.env.ACCESS_TOKEN_SECRET, (err, aux) => {
        if(err) return res.sendStatus(403)
        req.user = aux
        next()
    })

}

module.exports = {
    "authenticateToken": authenticateToken
}
```

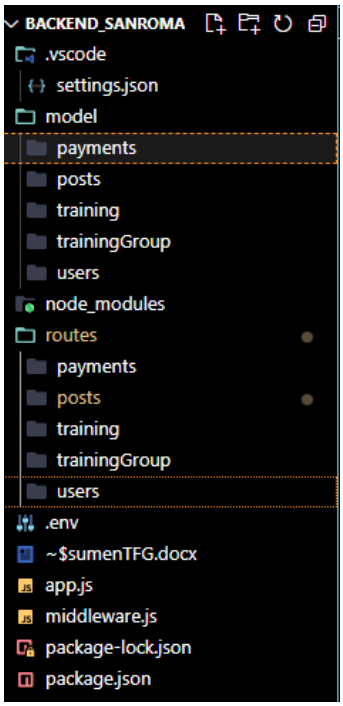
Código 13

```
router.get("/allposts",middleware.authenticateToken, async(req, res) =>{
    post.find({})
```

Código 14

Con esto se acabó la parte de programación, y se pasó a la parte de pruebas, para lo que se usó Postman. El objetivo es que los endpoints y middleware funcionaran correctamente, y así fue.

Para organizar el código, lo que se hizo fue crear una carpeta donde se guardan los modelos y dentro de la misma, se organizan los objetos según su función. Lo mismo se hizo con los endpoints. En *Carpetas 1* podemos ver de manera visual la estructura de carpetas.



Carpetas 1

El punto de unión es un archivo llamado “app.js” con la aplicación principal que se llama para iniciar el backend. Este archivo contiene todos los endpoints, todos los requires, la conexión con la base de datos y los modelos. Otro archivo llamado “middleware.js” contiene el middleware que hace las autentificaciones de token *(Código 15)*.

```
const mongoose = require("mongoose");
const express = require("express");
const cors = require("cors");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcrypt")

const User = require("./model/users/user.js")
const Amount = require("./model/training/amountTrain.js")
const Exercise = require("./model/training/exercise.js")
const Training = require("./model/training/dailyTrain.js")
const Post = require("./model/posts/post.js")
const Block = require("./model/trainingGroup/block.js")
const Phase = require("./model/trainingGroup/phase.js")
const Week = require("./model/trainingGroup/week.js")

const app = express();

const userRoutes= require('./routes/users/userRoutes.js')
const amountTrain = require("./routes/training/amountTrainRoutes.js")
const exerciseRoutes = require('./routes/training/exerciseRoutes.js')
const postRoutes = require('./routes/posts/postRoutes.js')
const trainingRoutes = require("./routes/training/dailyTrainRoutes.js")
const blockRoutes = require('./routes/trainingGroup/blockRoutes.js')
const phaseRoutes = require("./routes/trainingGroup/phaseRoutes.js")
const weekRoutes = require("./routes/trainingGroup/weekRoutes.js")

require('dotenv').config()

app.use(express.json());

app.use(cors({
  origin: 'http://localhost:5173'
}))

mongoose.connect(process.env.DB_CONNECTION_STRING)
    .then((result) => app.listen(3000,() => {
      console.log("Listening to 3000");
    })).catch((err) => console.log(Error));


app.use("/user",userRoutes)
app.use("/exercise",exerciseRoutes)
app.use("/amount",amountTrain)
```

Código 15

En el archivo “app.js” se escribe el inicio de la url para hacer las acciones CRUD a un objeto, estando el resto de la url en los distintos métodos.

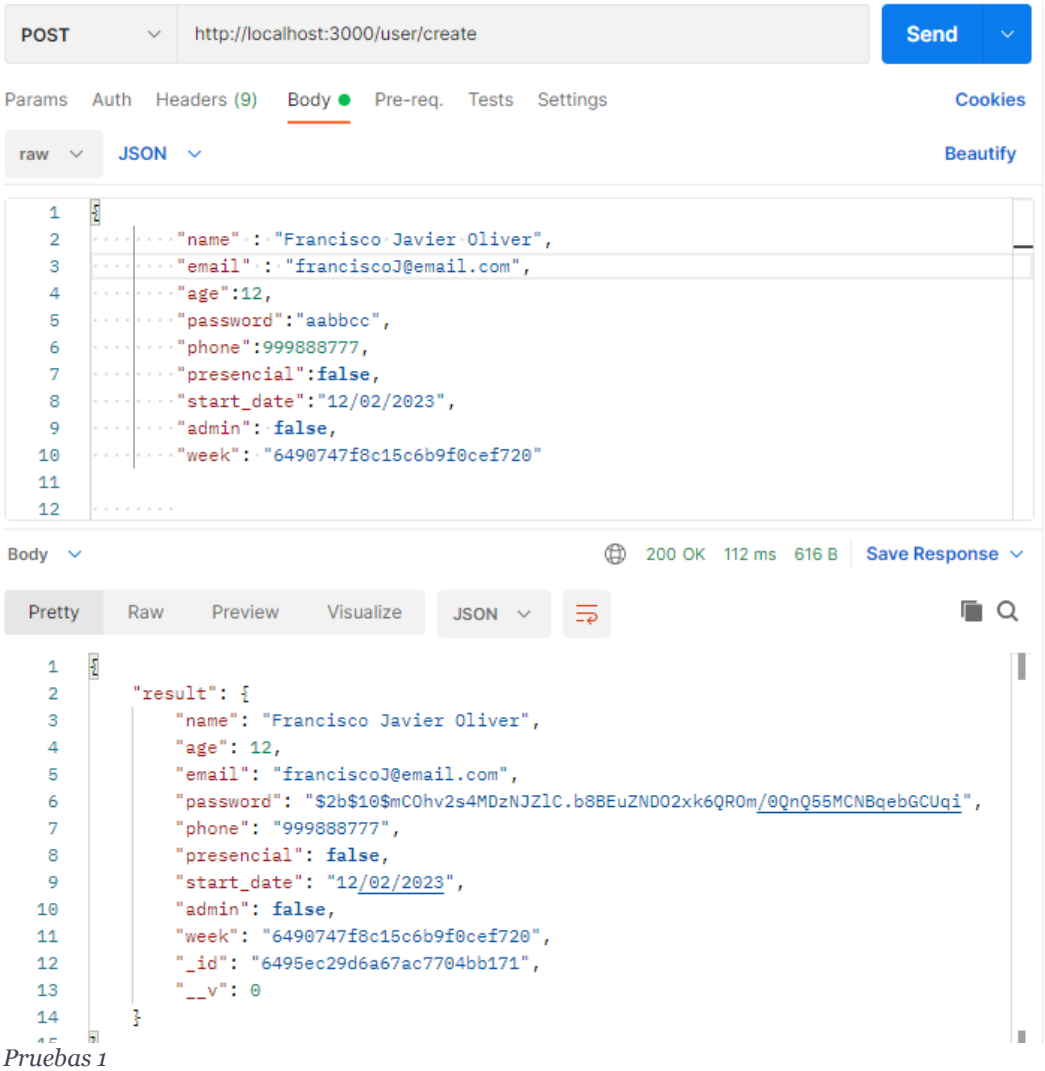
6. Pruebas

En este apartado se presentan las pruebas realizadas. Primero haremos las pruebas del backend usando Postman [24], para comprobar que nos devuelve los resultados esperados y posteriormente, integraremos el backend y el frontend y haremos pruebas de funcionalidad de la aplicación web.

6.1 Pruebas del backend

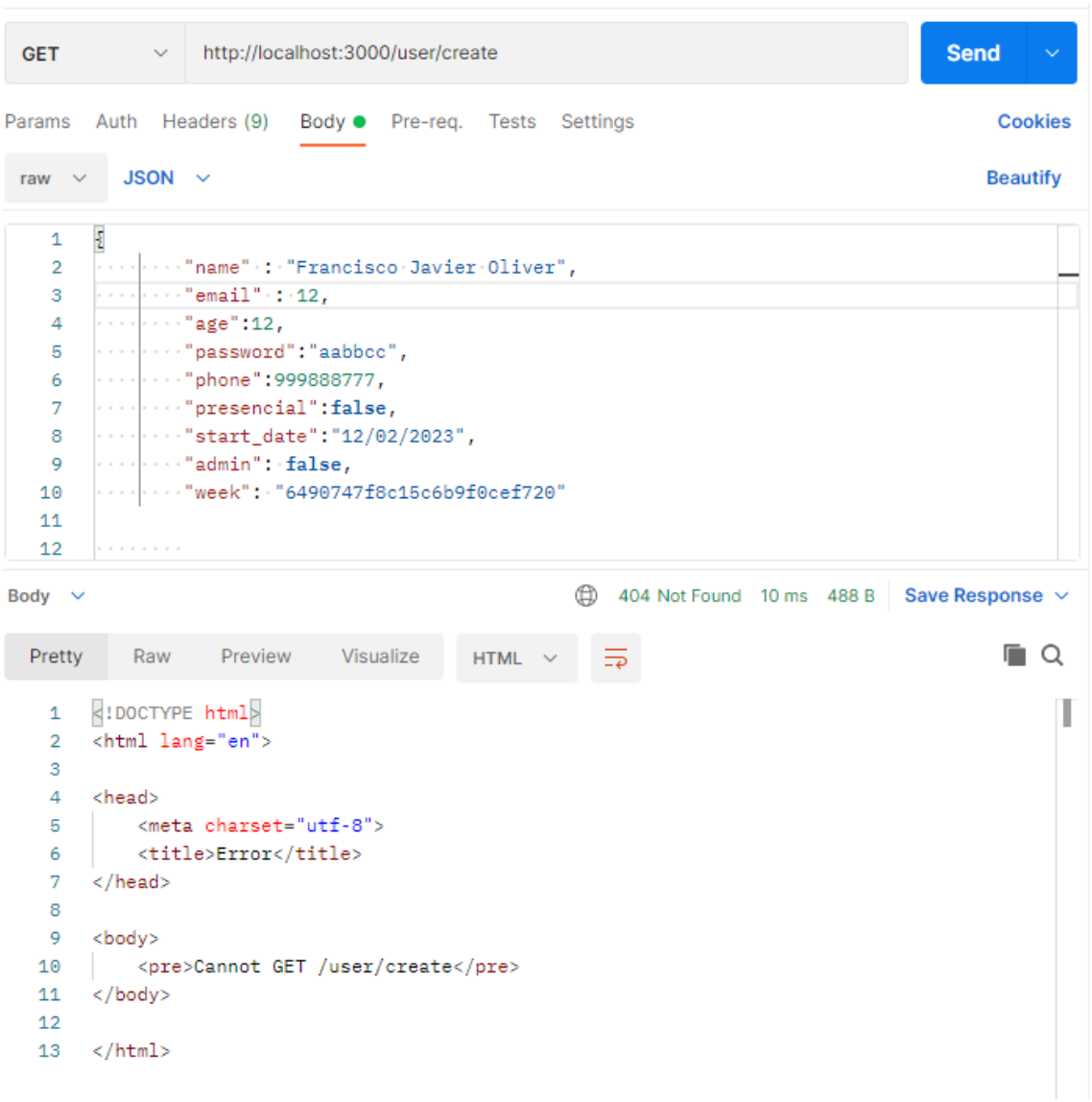
Se realizaron pruebas para comprobar que las operaciones de creación, modificación, obtención y eliminación de todas las colecciones de la BBDD eran correctas. Como estas colecciones poseen métodos muy similares, para no repetirnos, se presentarán únicamente las más destacables. A continuación, se presentan las pruebas realizadas.

Crear un usuario. Empecemos por crear un usuario, ya que esto será lo primero que se hará en el backend al acceder a la aplicación.



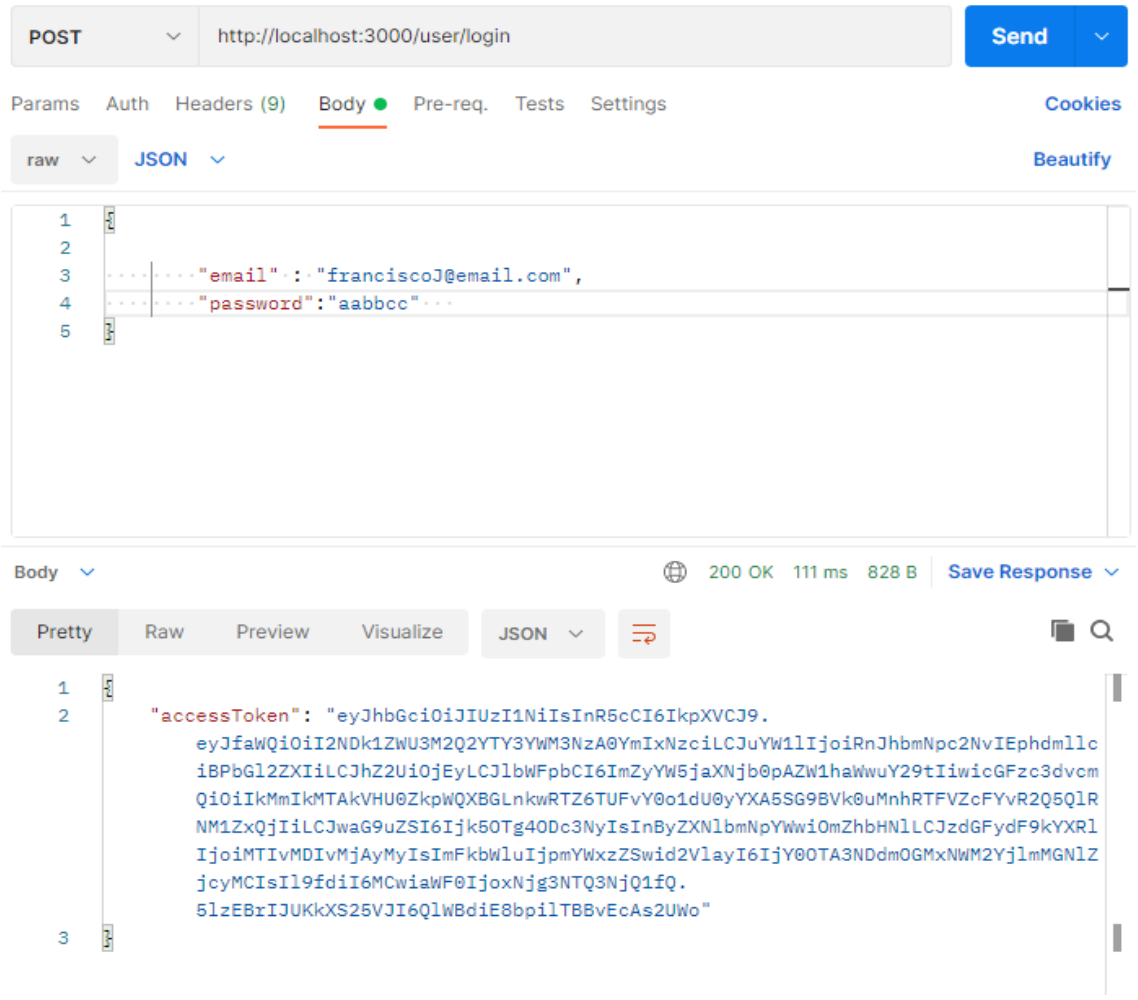
Pruebas 1

Escribiendo los datos del usuario con el formato correcto, se recibirá un código 200 que significa que ha ido bien y el objeto creado de regreso. La contraseña podemos ver que no coincide debido a que estaría hasheada. Si intentáramos usar la url con otro tipo de método nos daría un código 404 *(Pruebas 1 y 2)*.

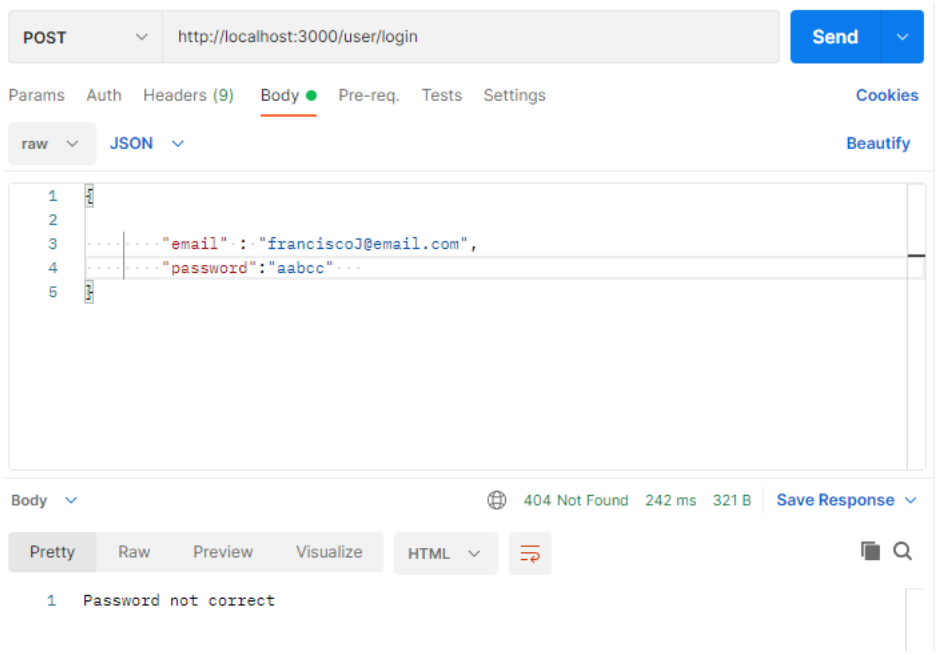


Pruebas 2

Para iniciar sesión también usaríamos un método post, usando el body para llevar únicamente el email y la contraseña. Esta petición, al contrario que la anterior, devolverá el token que será usado por los demás métodos para verificar el acceso a la base de datos. En el caso de que la contraseña este mal, nos devolverá un código 404, al no haber encontrado un usuario con ese email y que tenga asignada esa contraseña *(Pruebas 3 y 4)*.

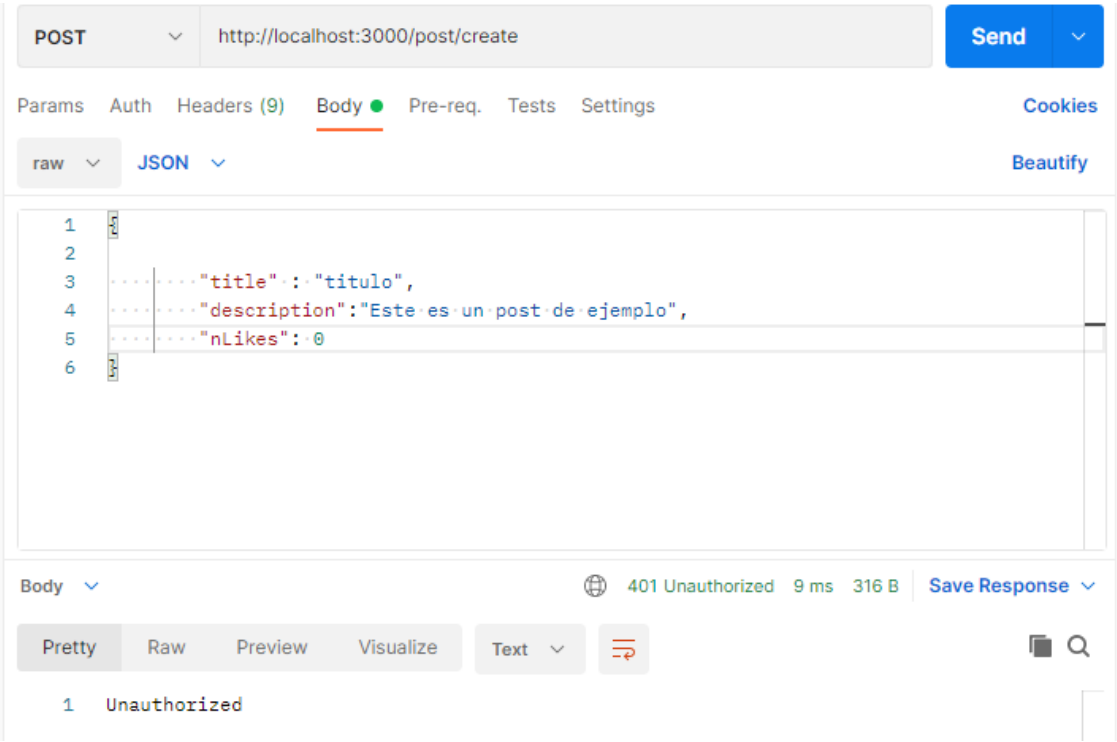


Pruebas 3



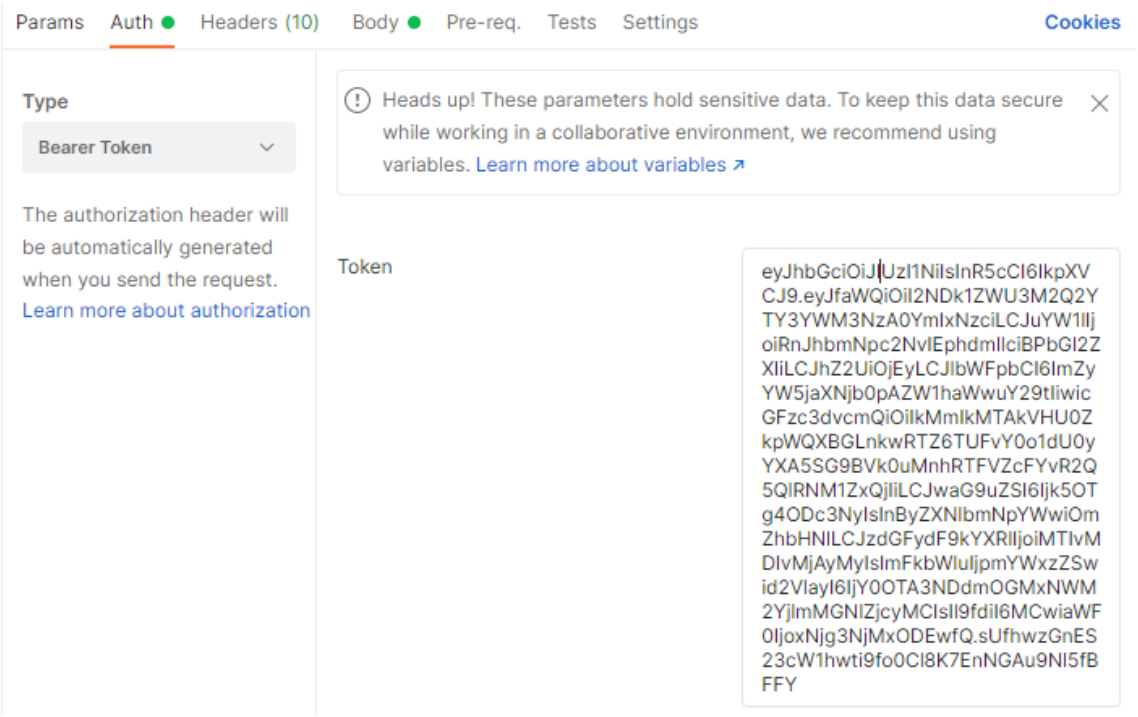
Pruebas 4

A continuación, usaremos de ejemplo los posts para practicar el uso del token para autorizar las peticiones. Empecemos intentar hacer un post sin el token.



Pruebas 5

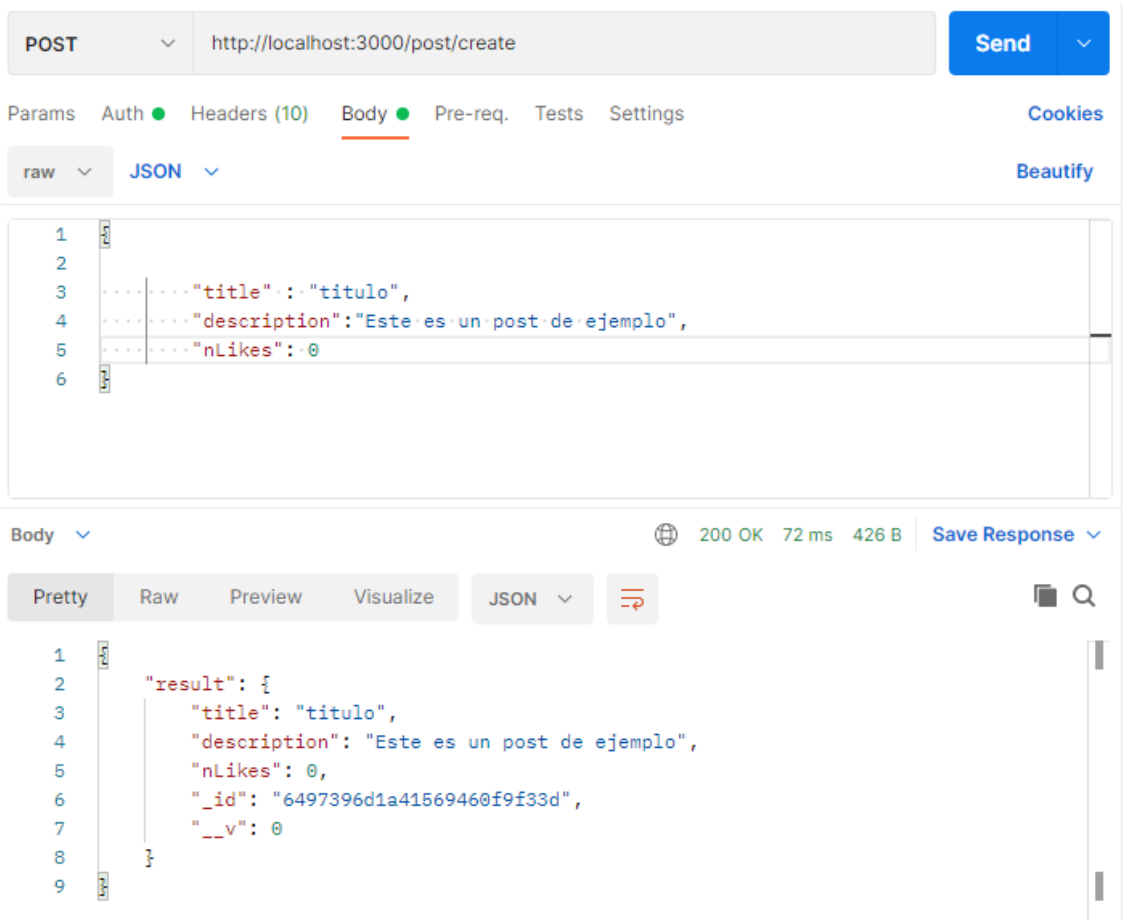
Como podemos ver en *Pruebas 5*, nos devolverá un código 401 de que la petición no está autorizada. Esto es debido a que no hemos introducido el token en la cabecera. En el caso de Postman debemos introducirlo en el apartado de Auth (*Pruebas 6).*



Pruebas 6

Ahí seleccionamos que será un Bearer token, el cual significa que será un token que estará en la cabecera.

Una vez hecho esto volvemos a hacer la petición:

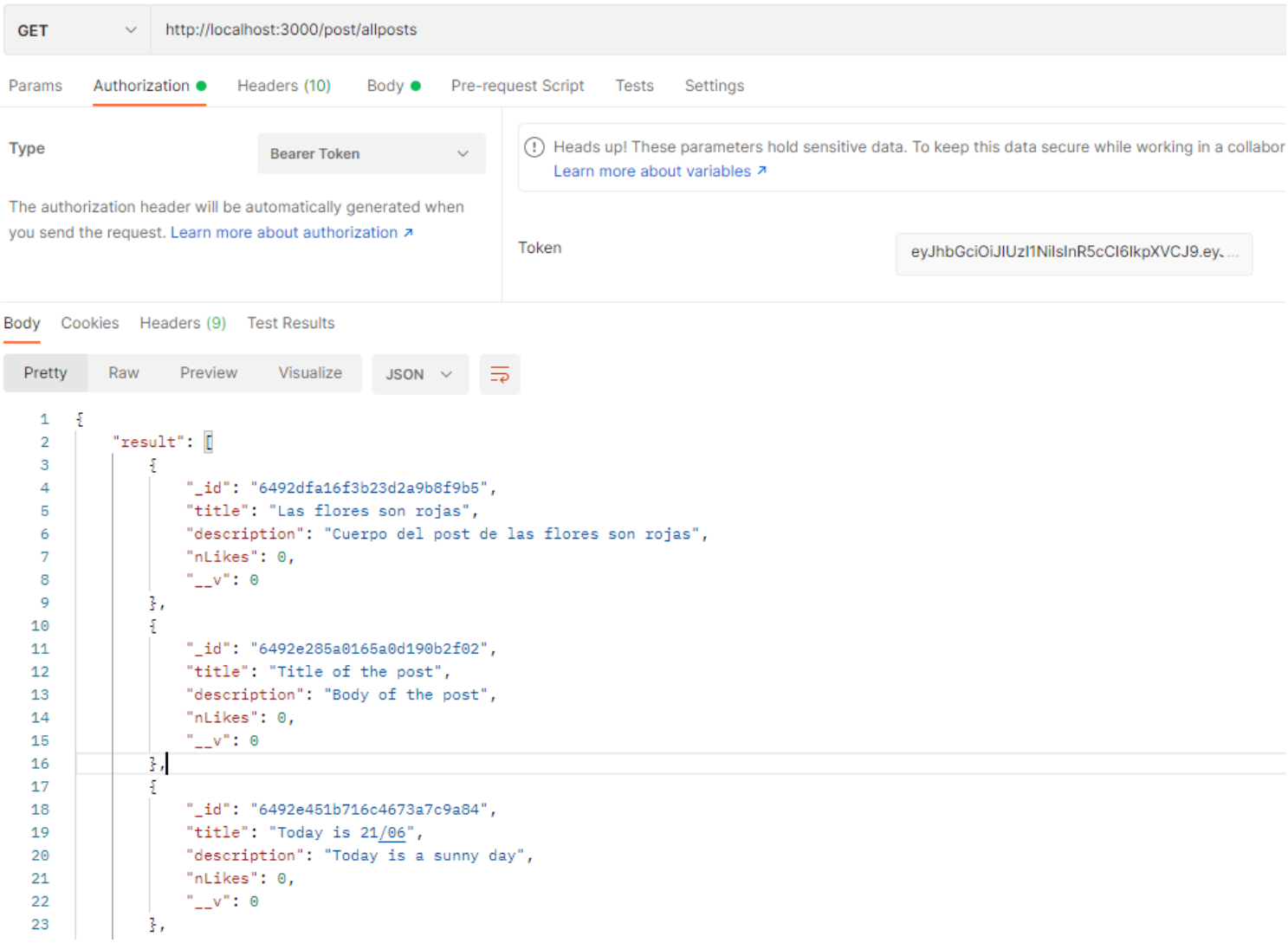


Pruebas 7

Esta vez se ha generado sin problemas, y este método devolverá el post creado *(Pruebas 7).*

Una vez enseñado que el método POST funciona, usaremos los posts para también hacer las demostraciones de los otros métodos.

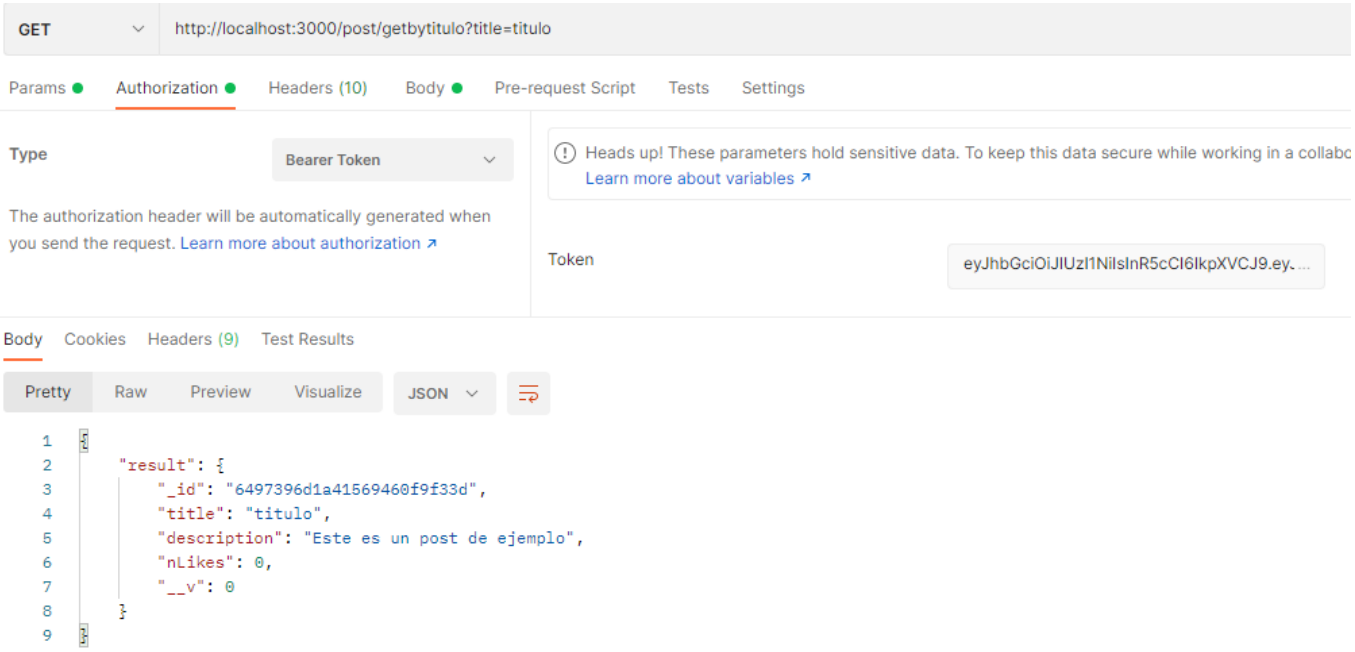
Ahora iremos con el método GET. La mayoría de los modelos tienen un método para obtener todos los objetos de la base de datos de ese modelo y otro para buscar a alguno en particular ya fuese por el id o por algún otro parámetro (título, nombre, etc.) a parte de los métodos particulares que tendrá cada modelo para buscar los objetos que siguen una condición en particular.



Pruebas 8

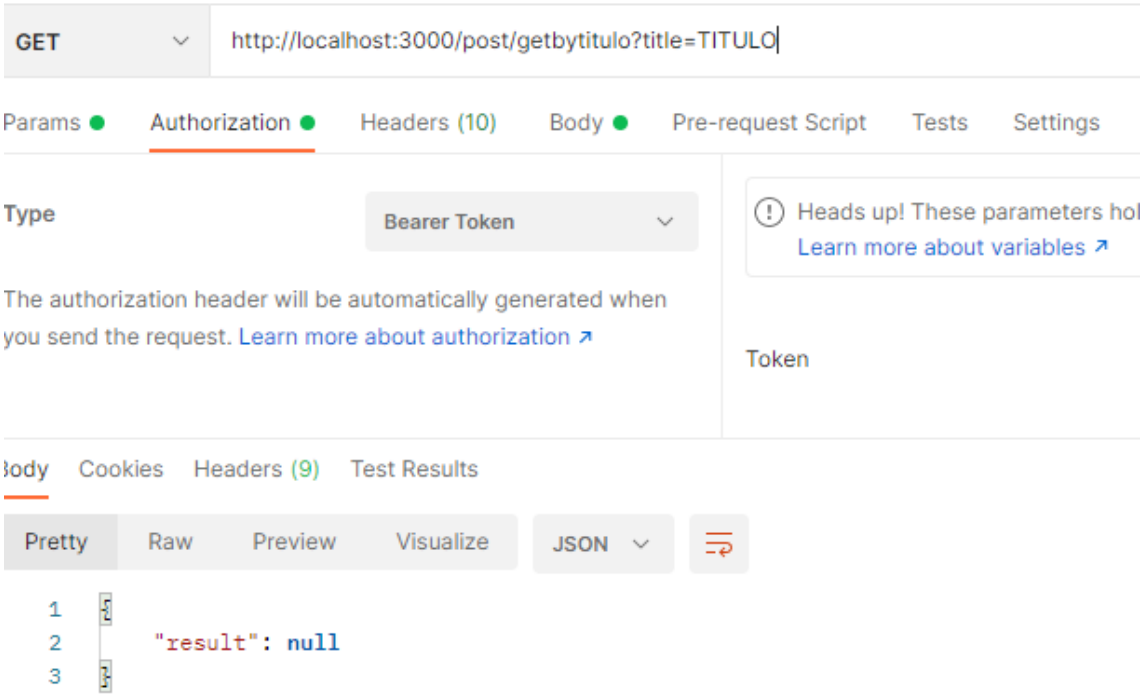
A continuación, podemos ver en *Pruebas 8* un método GET para obtener todos los posts en el que, introduciendo el token, y sin necesidad de introducir un cuerpo nos devuelve un JSON con todos los posts de la base de datos.

Ahora mostraremos la obtención de un objeto introduciendo el parámetro que queramos buscar el la url en forma de una query, en el caso del post, buscaremos por título, aunque también es posible hacerlo por id. Obviamente, también deberemos introducir el token para ser autorizados a hacer la búsqueda en la base de datos *(Pruebas 9)*.



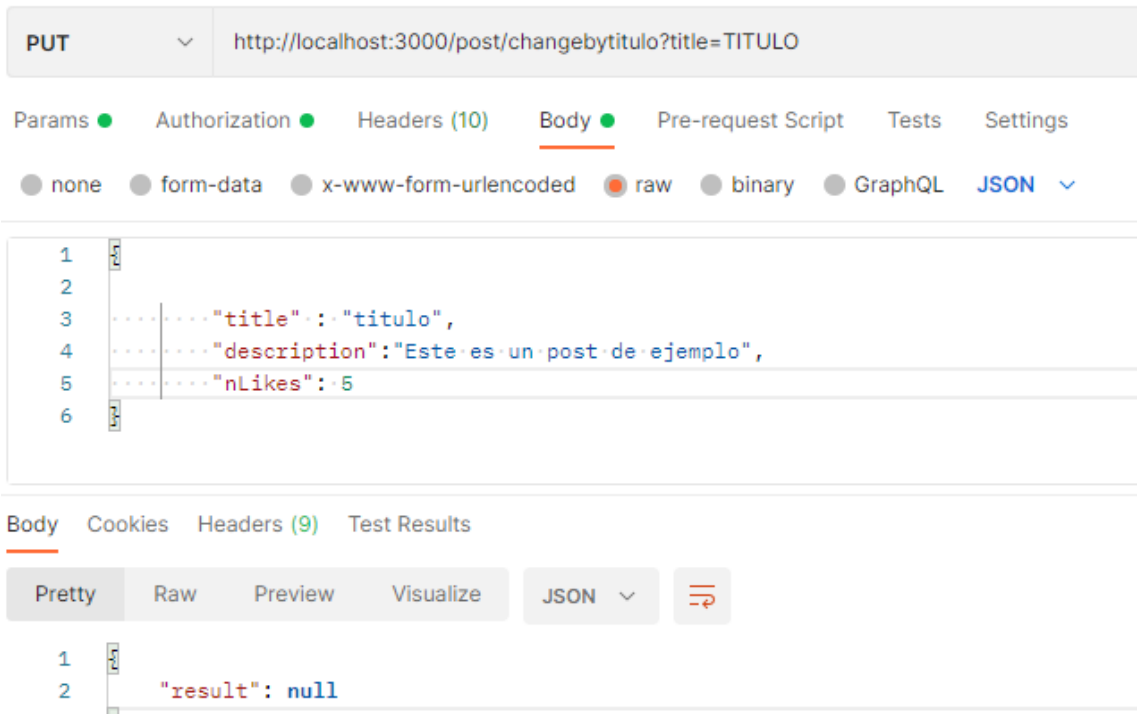
Pruebas 9

En caso de escribir un título que no coincida con ningún objeto, el resultado nos dará null debido a que no ha encontrado ningún post con ese título en la base de datos *(Pruebas 10).*



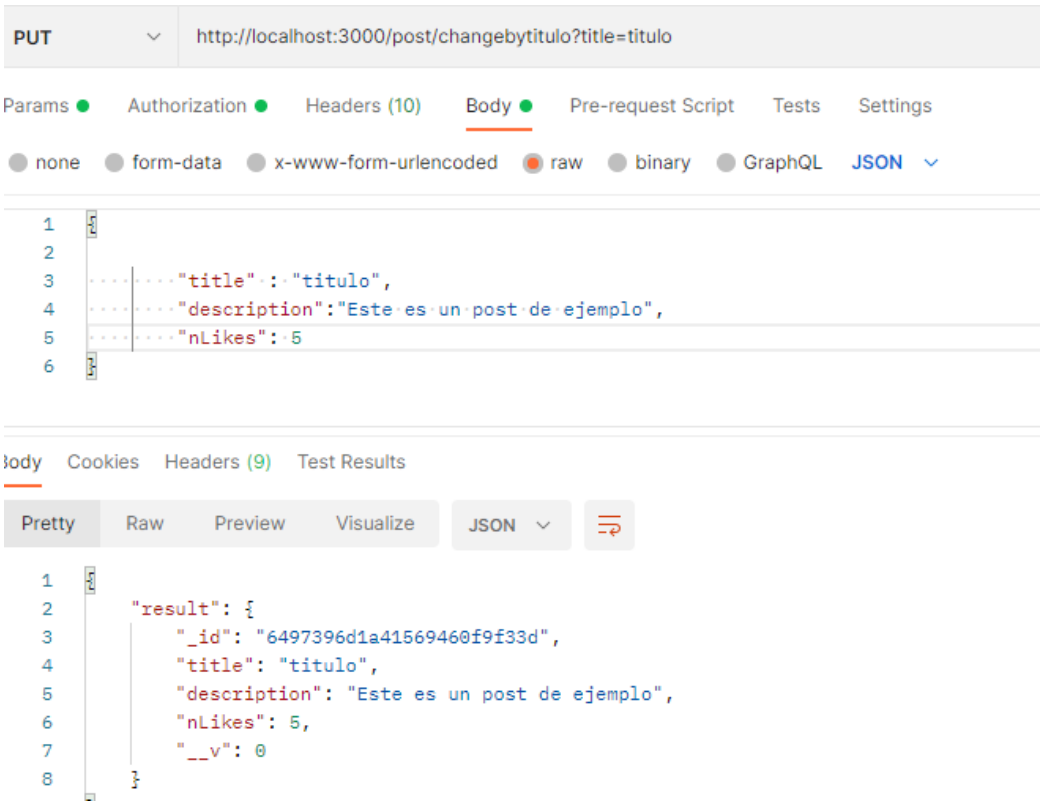
Pruebas 10

A continuación, intentaremos modificar un post con el método PUT. Nuevamente, lo buscaremos por título y, nuevamente, si el titulo no coincide con ninguno de la base de datos *(Pruebas 11).*



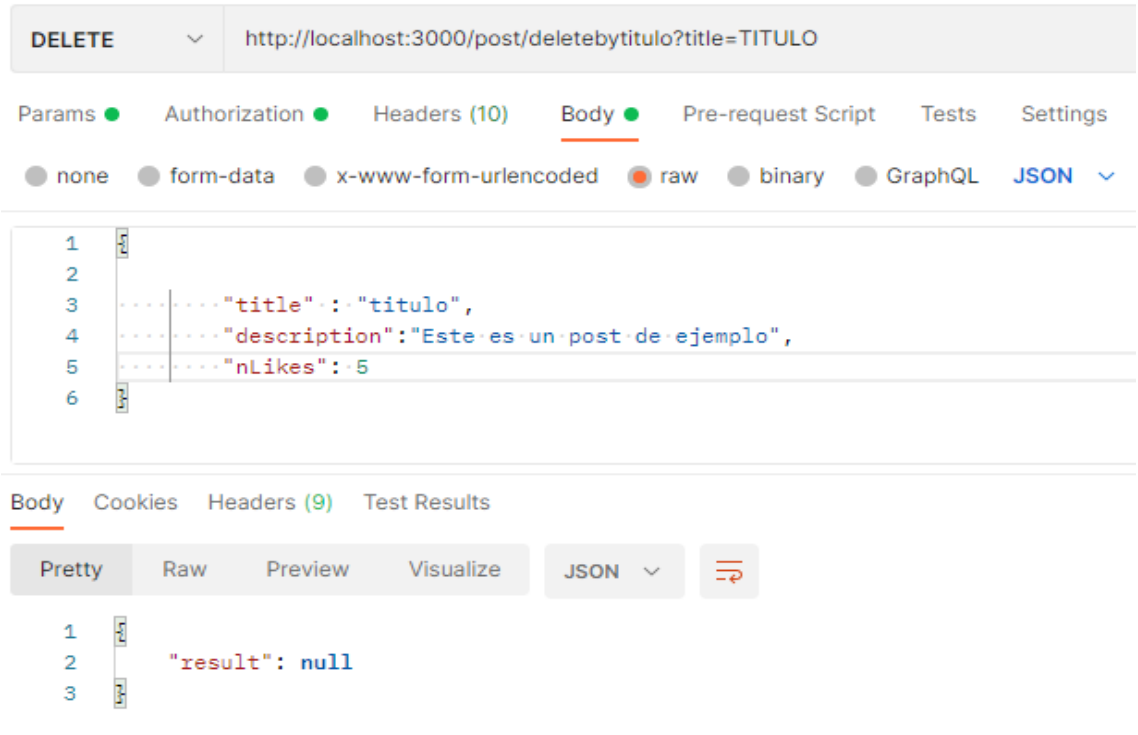
Pruebas 11

En el cuerpo de la petición de modificación debemos introducir como deberá ser el cuerpo del nuevo objeto. El objeto será el mismo, como podemos comprobar debido a que tendrá el mismo id, pero los parámetros serán los que hayamos escrito en el cuerpo *(Pruebas 12).*



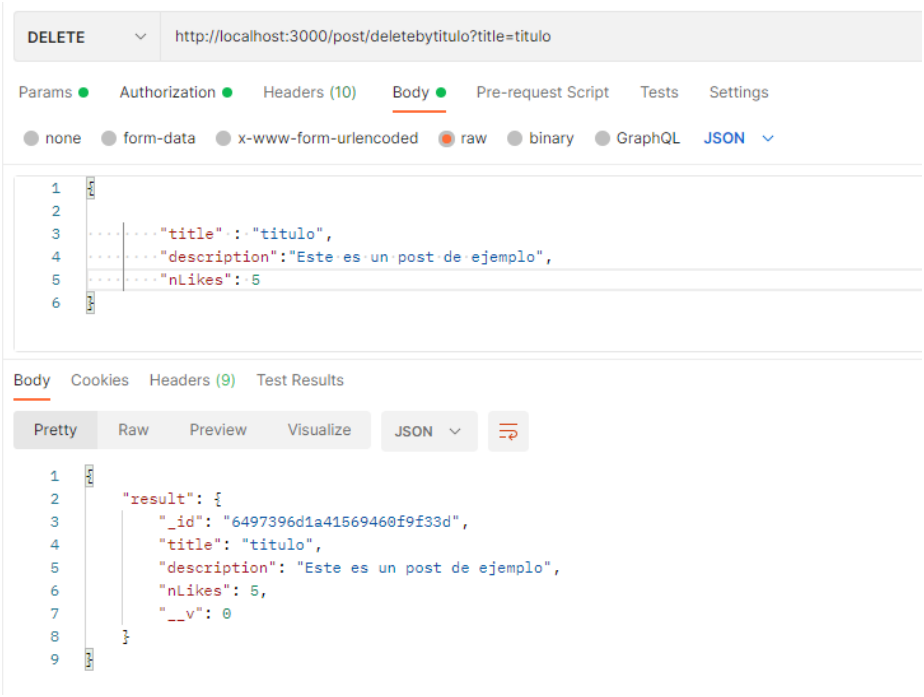
Pruebas 12

Por último, vamos a eliminar el objeto. Lo haremos usando el método DELETE. Pasará lo mismo que con el método PUT y GET si el título es incorrecto *(Pruebas 13).*



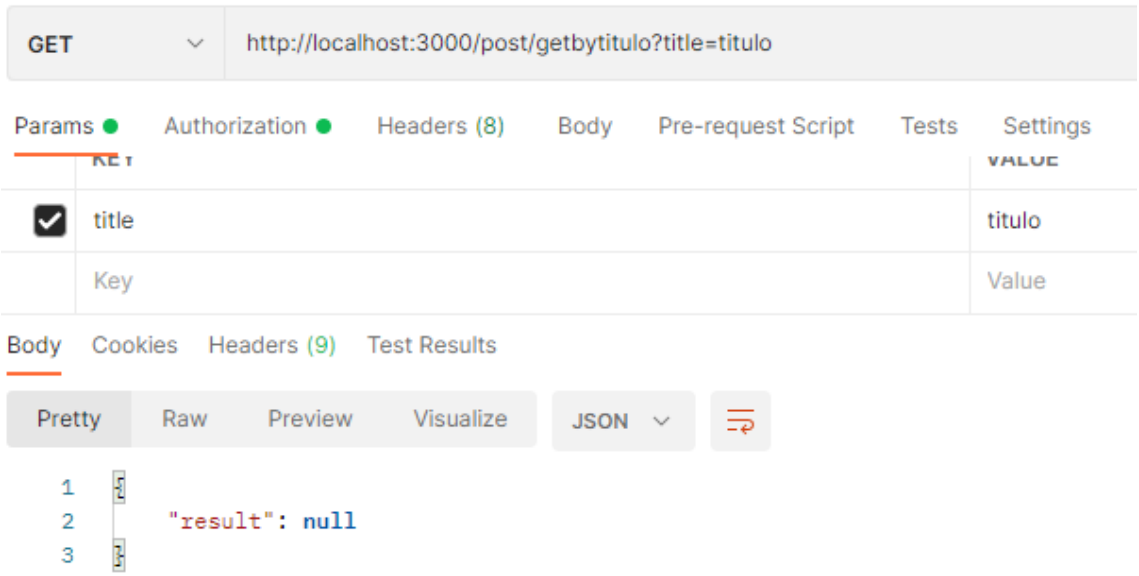
Pruebas 13

Una vez escribamos el titulo correctamente, eliminará el objeto y lo devolverá. *(Pruebas 14).*



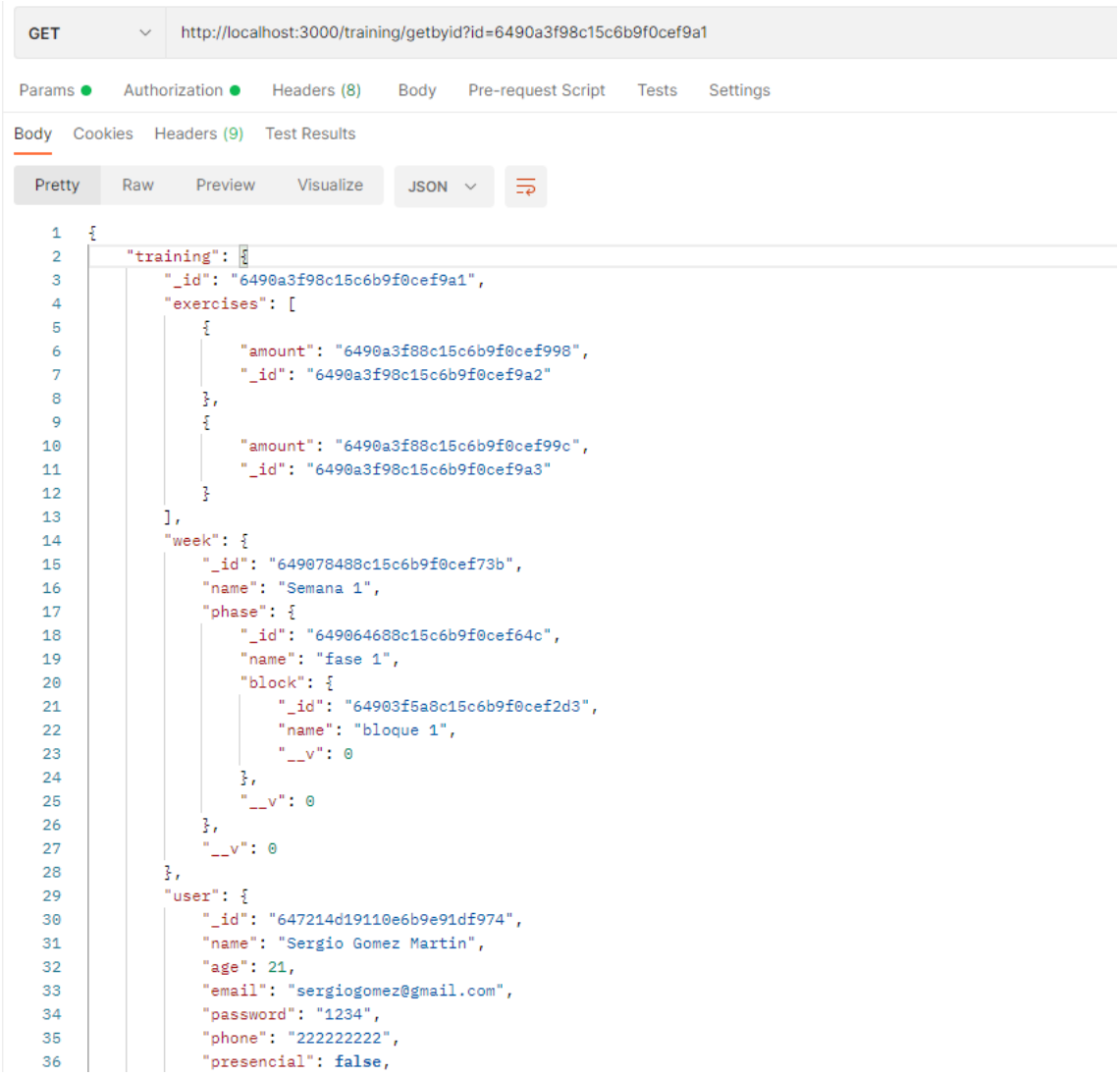
Pruebas 14

En el caso de que hagamos un método GET para buscarlo en la base de datos, podremos comprobar que, efectivamente ha sido eliminado *(Pruebas 15)*.



Pruebas 15

Ahora mostraremos como se deberían mostrar las referencias a otros objetos, ya que deberían aparecer todos los parámetros del objeto referenciado. En *Pruebas 16* vemos que los 2 objetos que están referenciados (el user y la week) aparecen con todos los parámetros.



Pruebas 16

6.2 Pruebas de funcionalidad de la aplicación

A continuación, vamos a hacer las pruebas de funcionalidad de la aplicación, viendo que todas las acciones que se deben hacer en ella funcionen correctamente.

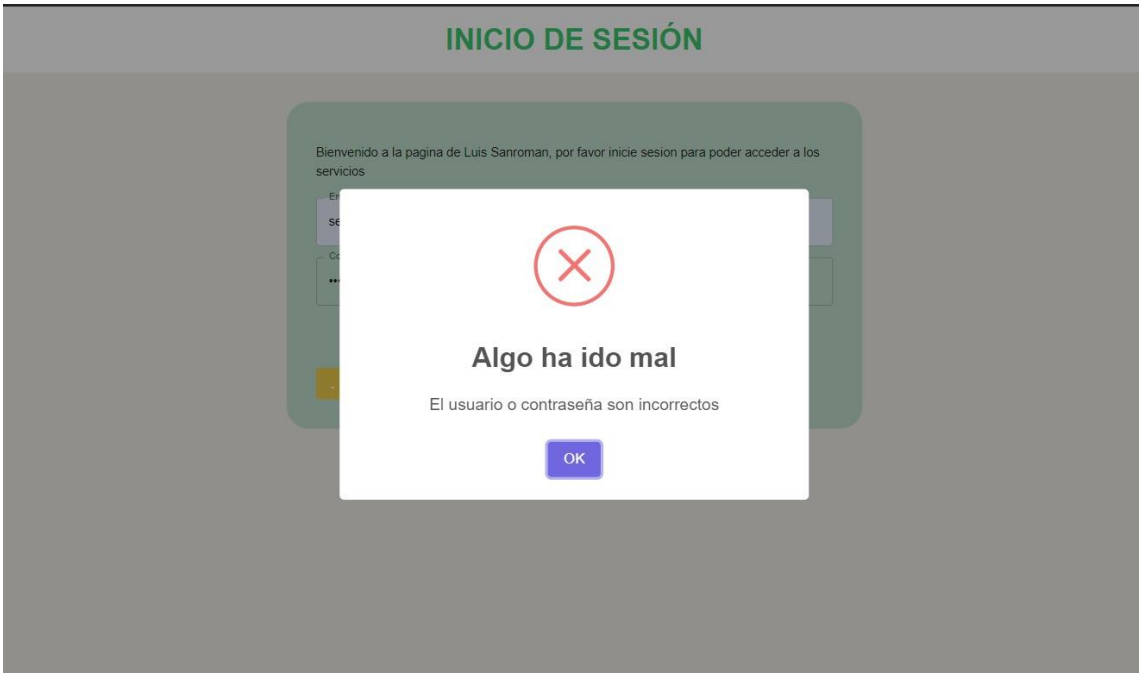
Empecemos con la parte de creación de usuario, registrando un usuario. Si dejamos los campos vacíos nos devolverá un error *(Pruebas 17)*, al igual que si se deja el número de teléfono o el email con un formato incorrecto *(Pruebas 18)*. En el caso de que todo este correcto, nos introducirá a la página inicial de la aplicación, al igual que iniciando sesión.

Pruebas 17

Pruebas 18

Pasaremos ahora a iniciar sesión con un usuario que ya se ha registrado en la aplicación. Al igual que en iniciar sesión, nos dará error si los campos están vacíos, al igual que si la contraseña o el usuario son erróneos.

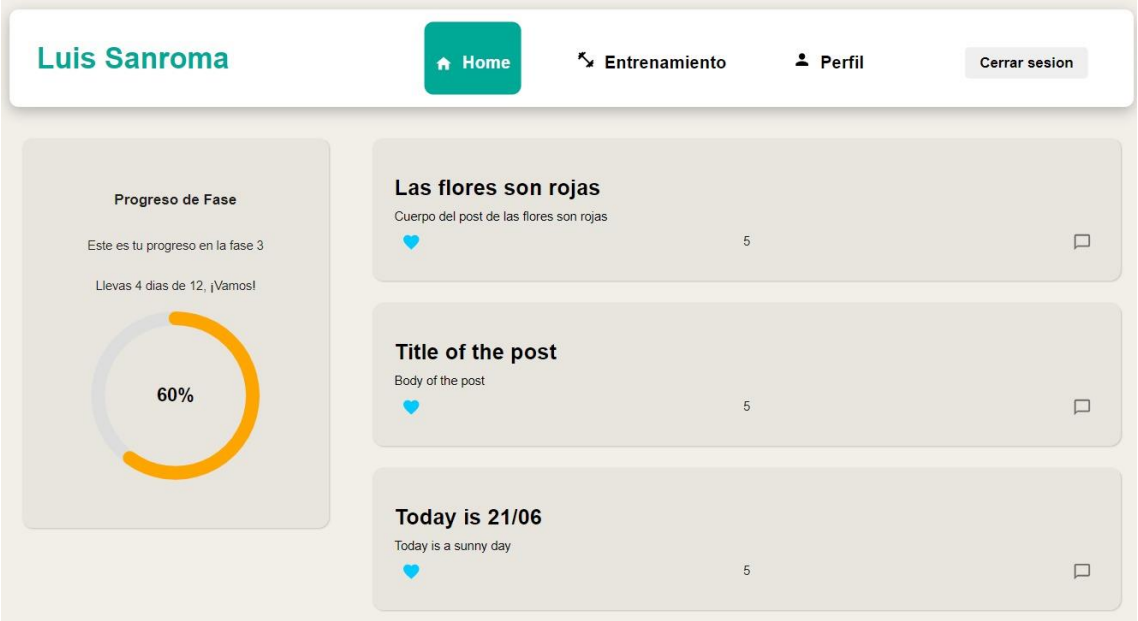
Pruebas 19



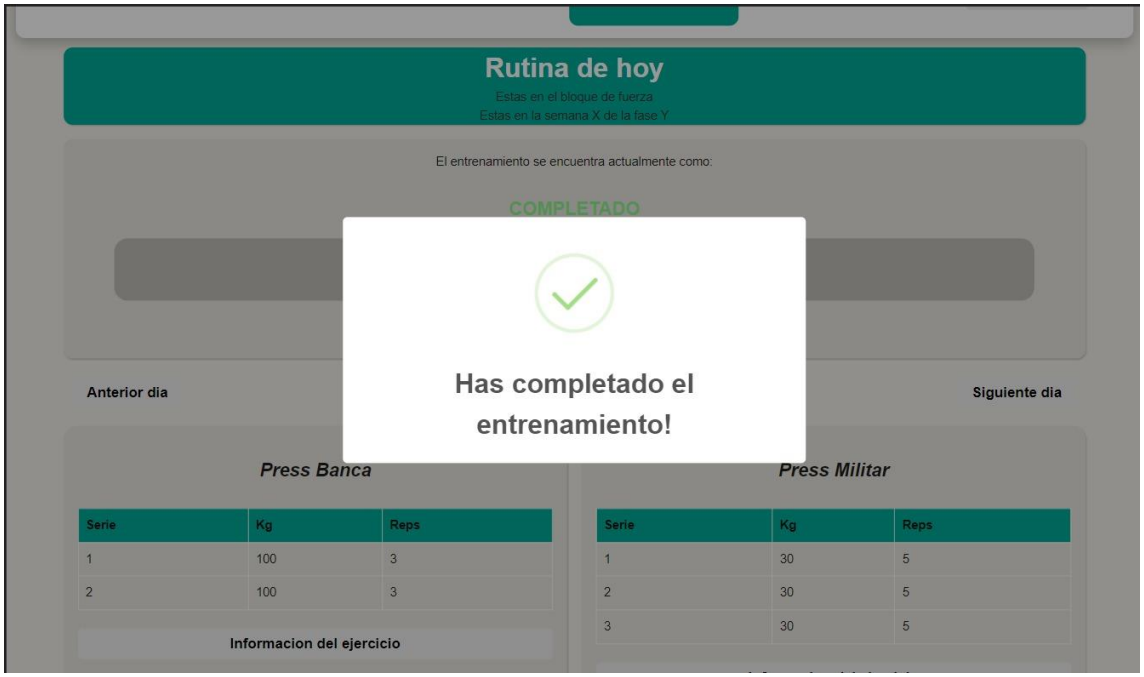
Pruebas 20

Una vez iniciada la sesión, tendríamos la página inicial, *(Pruebas 21)* donde el usuario podrá ver el progreso por el que va en su entrenamiento y los posts que los entrenadores crean. Desde aquí podrán ir la página de entrenamiento, donde podrán ver los ejercicios que los entrenadores les han puesto y completar el ejercicio en el momento que hayan acabado con todo el entrenamiento propuesto *(Pruebas 22)*. Una vez

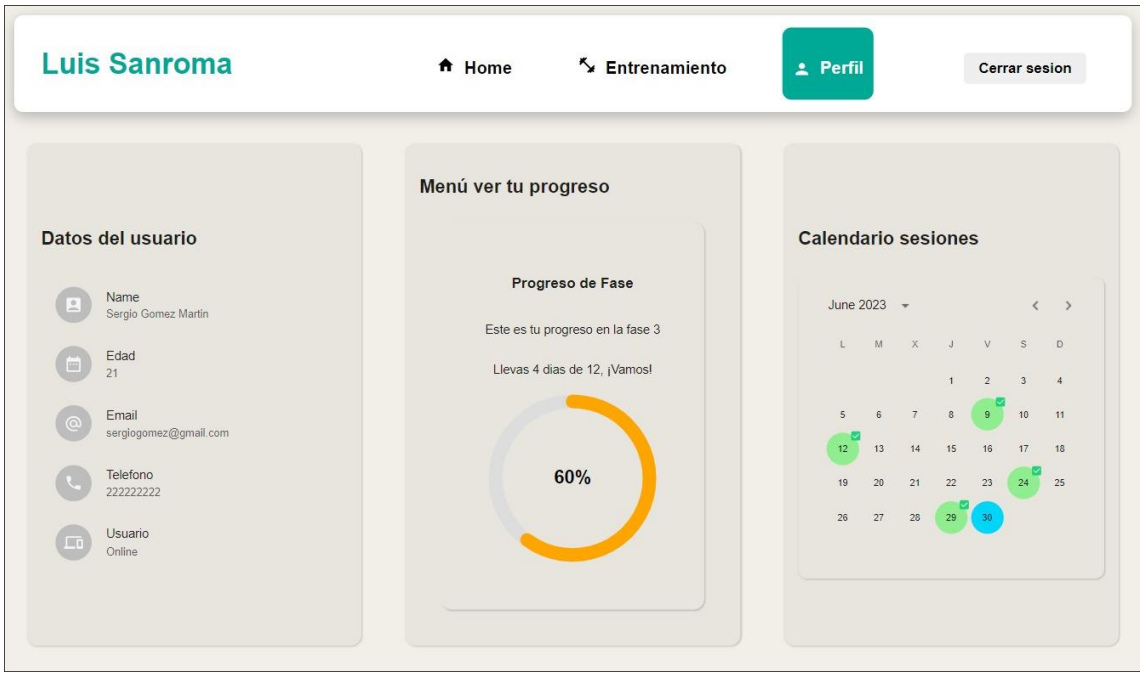
seleccionen perfil, podrán ver un resumen de su usuario, como sus datos personales, la fase por la que van o los entrenamientos que han hecho *(Pruebas 23)*.



Pruebas 21

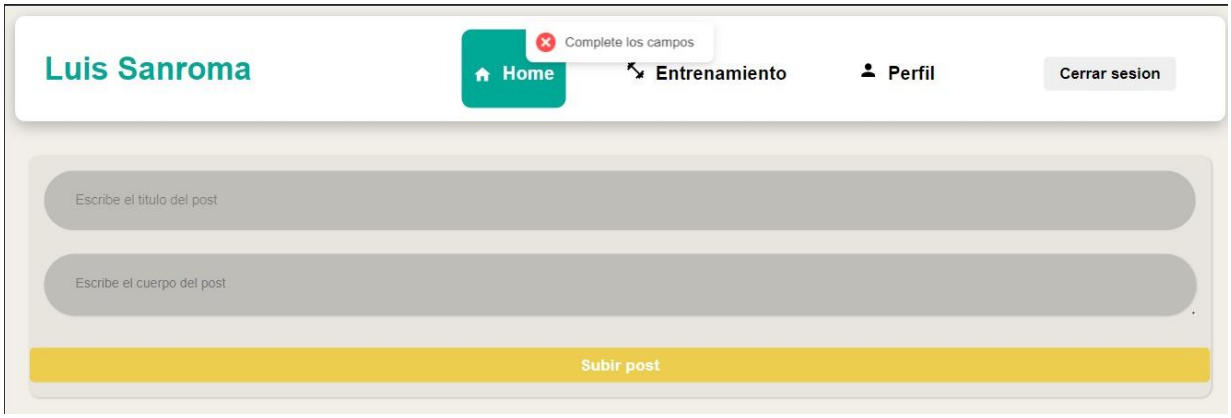


Pruebas 22

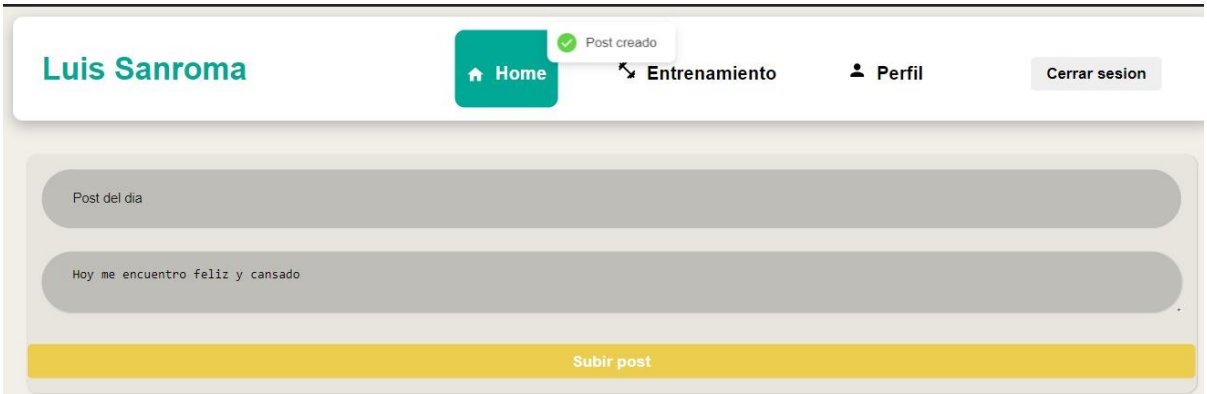


Pruebas 23

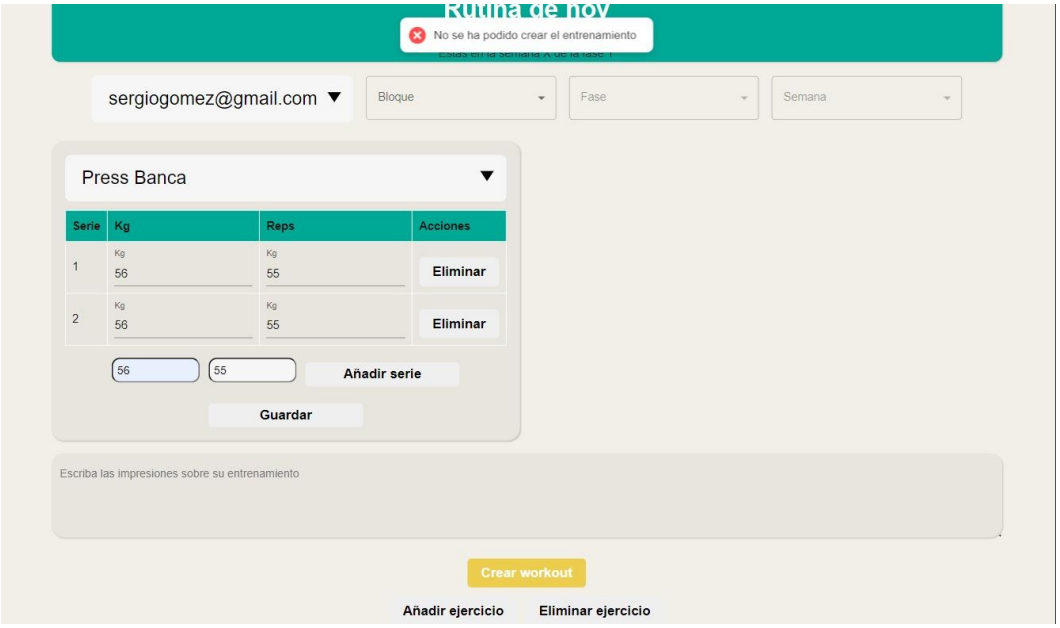
Respecto a lo que un usuario entrenador vera en primera instancia será el menú para poder crear posts, la cual dará error en el caso de que el entrenador quiera crear un post habiendo rellenado de forma incorrecta el titulo o la descripción del post *(Pruebas 24 y 25).* Seguidamente se tendría la función de crear un entrenamiento para un usuario, que dará error en el caso de que no se haya seleccionado un usuario o una fase, semana o bloque *(Pruebas 26,27 y 28).*



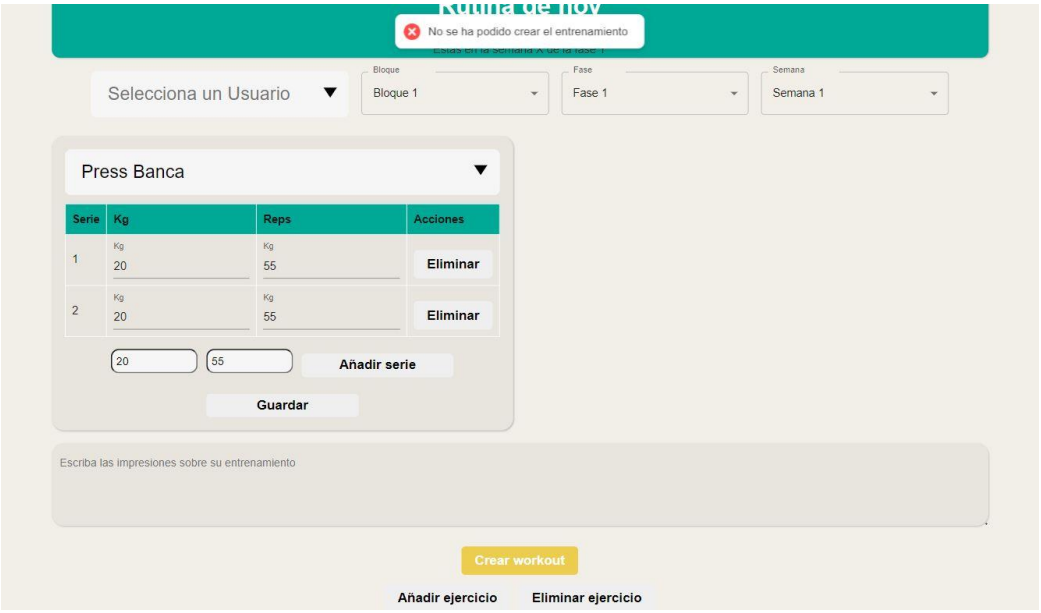
Pruebas 24



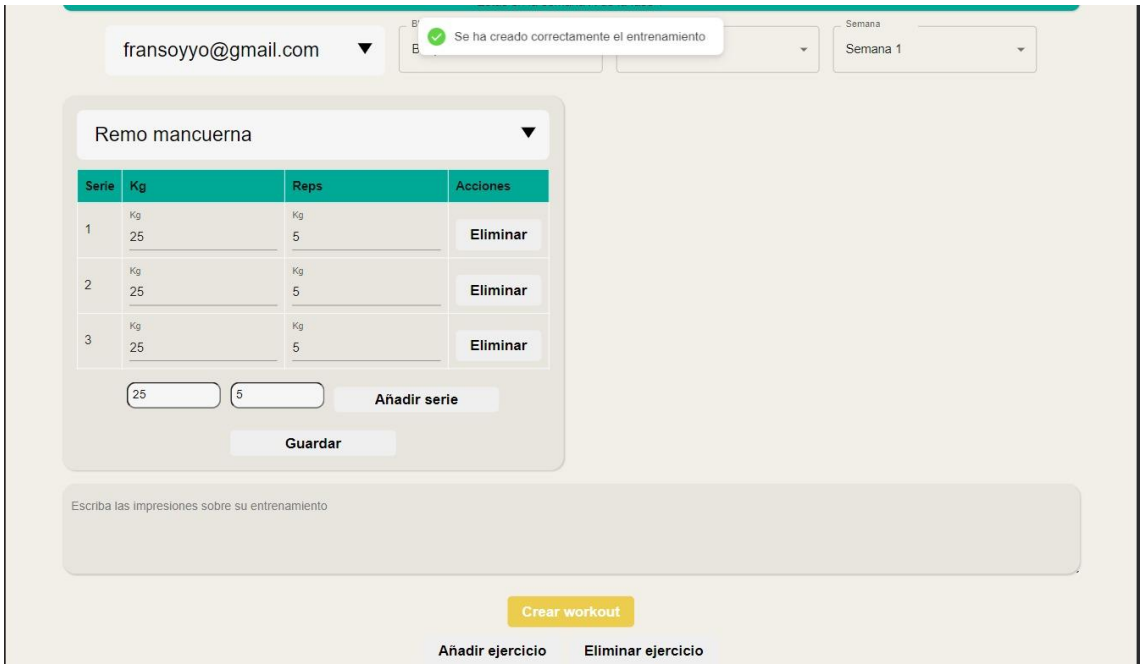
Pruebas 25



Pruebas 26

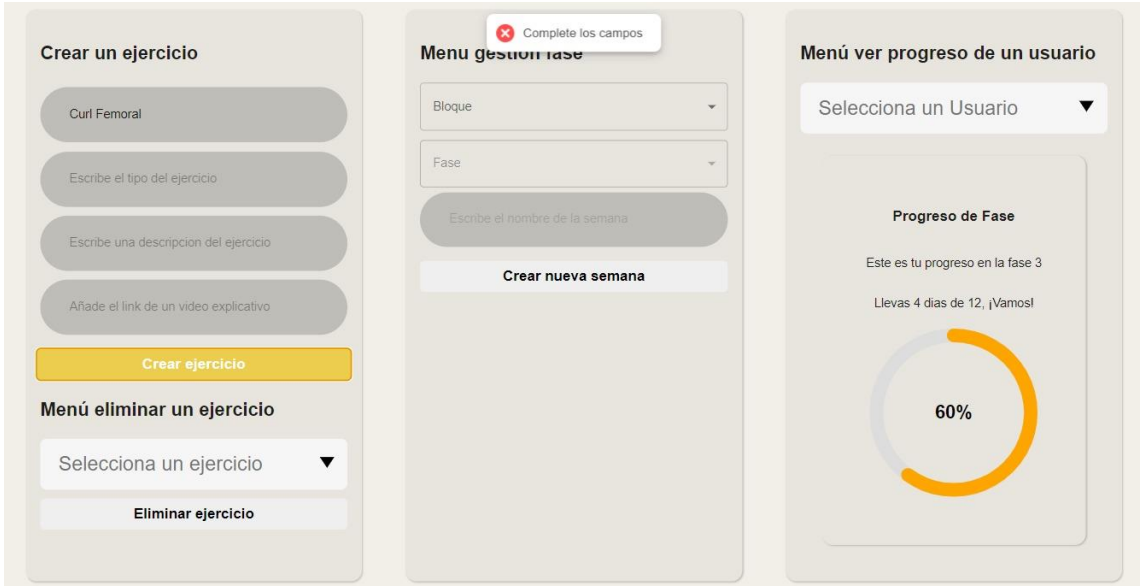


Pruebas 27

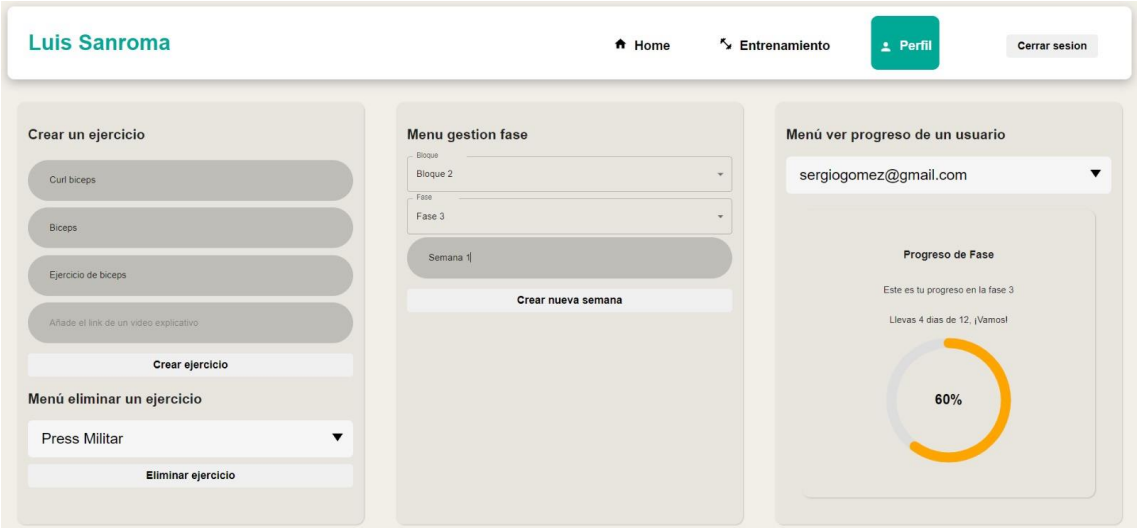


Pruebas 28

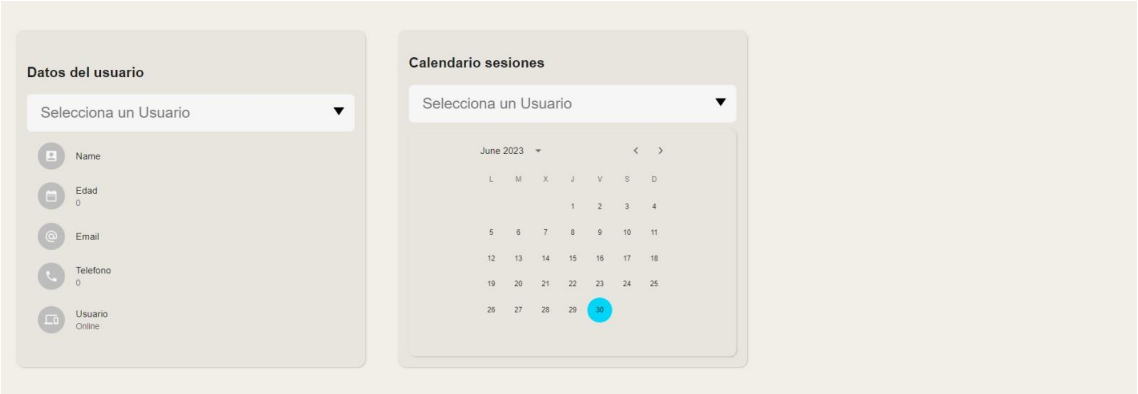
Si el entrenador está en la sección de perfil podrá crear un ejercicio o eliminarlo (el cual dará error si los parámetros están vacíos o el ejercicio para eliminar no está creado), podrá crear nuevas semanas (que dará error si los campos están vacíos), se podrá ver el progreso de un usuario (que no estará actualizado si el usuario no está seleccionado), se podrán ver los datos de un usuario (que no estarán actualizado si el usuario no está seleccionado) o se podrá ver las sesiones de un usuario (nuevamente desactualizadas si no se selecciona al usuario) *(Pruebas 29, 30, 31, 32)*. Una vez actualizado el calendario de sesiones, se podrán ver los ejercicios hechos en un día, junto a la información del ejercicio *(Pruebas 33 y 34)*.



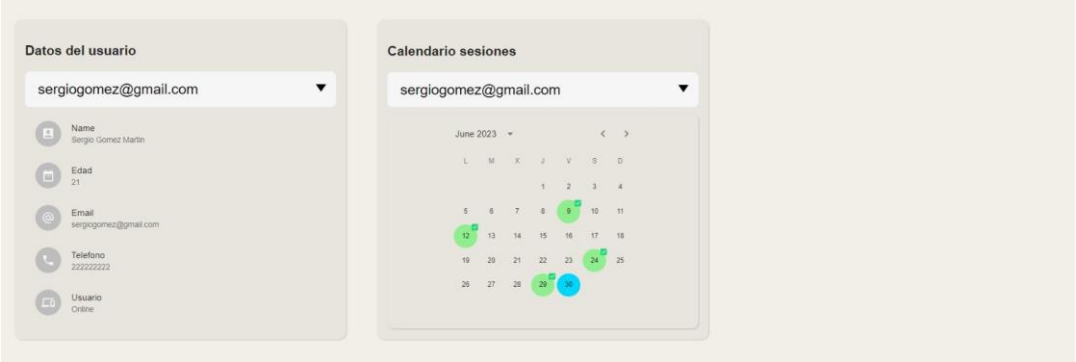
Pruebas 29



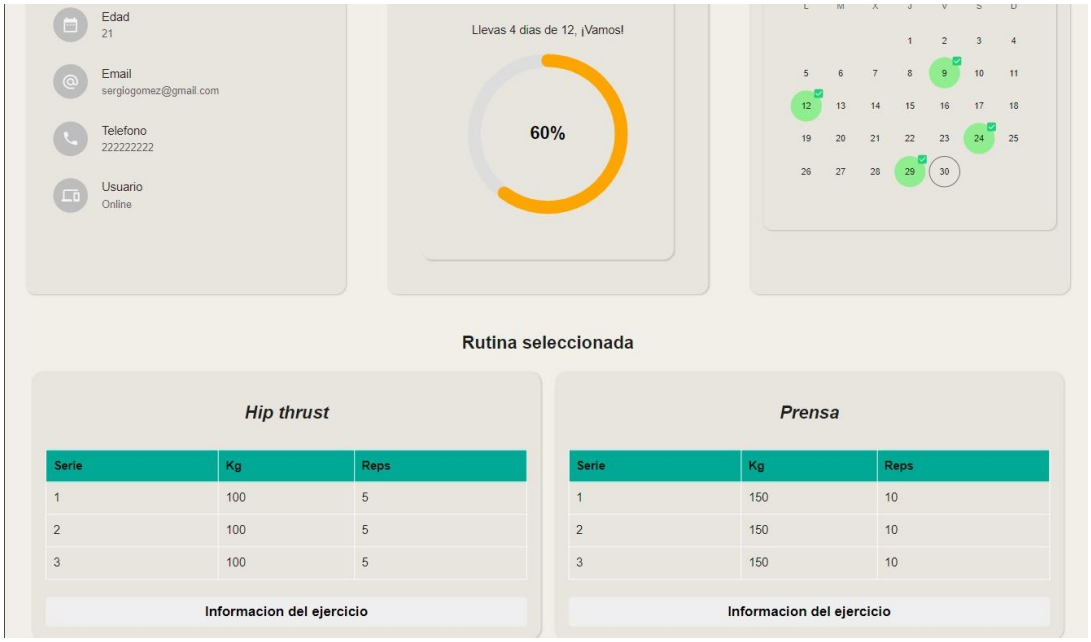
Pruebas 30



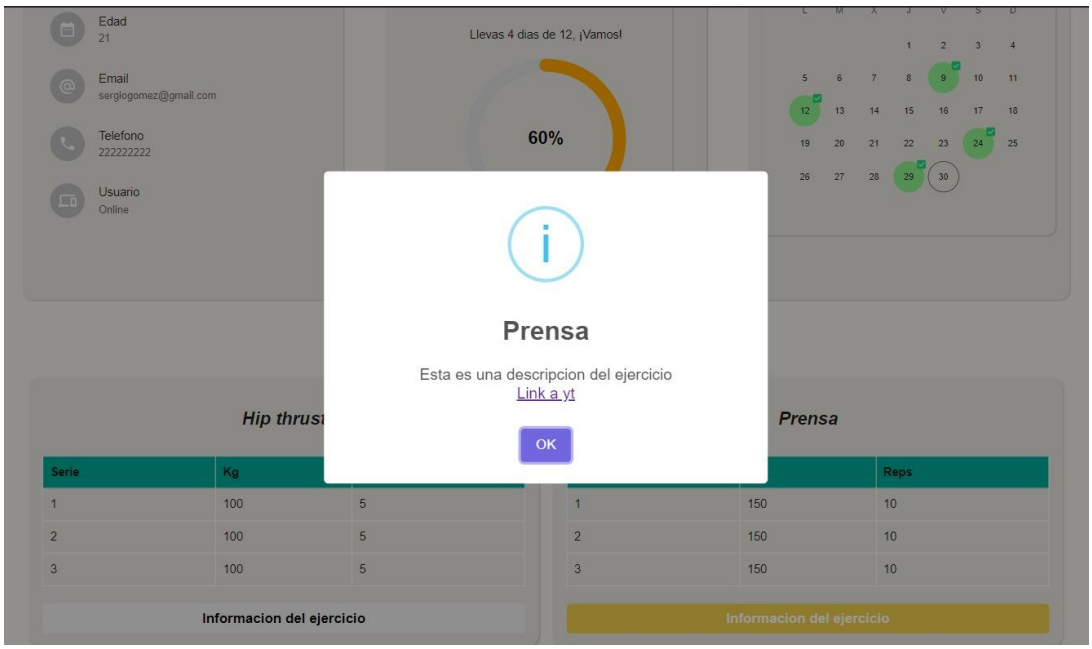
Pruebas 31



Pruebas 32



Pruebas 33



Pruebas 34

7. Conclusiones

En este capítulo se tratará de resumir qué se ha aprendido con este TFG, uniéndolo con lo aprendido durante los años de estudio en el grado de Ingeniería Informática y dedicando un tiempo a ver qué objetivos de los Objetivos de Desarrollo Sostenible se cumplen con la aplicación.

7.1 Análisis personal del desarrollo del TFG

El desarrollo de un backend utilizando Node.js y MongoDB, junto con la implementación de middleware para mejorar la seguridad, ha sido fundamental para el éxito de nuestra aplicación de entrenamiento personal. A lo largo de este trabajo, he explorado las ventajas de utilizar Node.js como entorno de ejecución y MongoDB como base de datos NoSQL.

Node.js me ha permitido construir un backend altamente eficiente y escalable, gracias a su arquitectura basada en eventos y a su capacidad para manejar múltiples solicitudes simultáneamente. Además, su amplio ecosistema de módulos nos ha brindado acceso a una gran cantidad de librerías y herramientas que facilitan el desarrollo de la aplicación como mongoose, dotenv o nodemon

Por otro lado, MongoDB ha sido una elección acertada para nuestra base de datos, ya que su modelo de documentos nos ha permitido almacenar y recuperar datos de manera flexible y eficiente.

Además, hemos implementado middleware de seguridad para proteger nuestros recursos y garantizar la confidencialidad, integridad y autenticidad de los datos.

Si cabe destacar que el inicio del proyecto a resultado complicado debido a la novedad de algunas herramientas como MongoDB o la inexperiencia creando middlewares o encriptando. Además, falto organización al inicio de la aplicación con mi compañero de frontend y con el cliente.

Mediante este trabajo, he aprendido mucho. He aprendido a como trabajar en el backend de un proyecto, indagando más en los términos y componentes de este, en varias capas. Además, también he aprendido a trabajar en equipo y teniendo un cliente, pudiendo resolver discrepancias con otros compañeros y entendiendo lo que un cliente quiere y lo que un cliente necesita.

7.2 Relación del trabajo desarrollado con los estudios cursados

En esta aplicación hemos usado ante todo los conocimientos obtenidos en las prácticas de empresa, en las cuales, entre otras cosas se me requirió hacer un backend, y se me pidió hacer con NodeJS, por lo que aplique los conocimientos con la tecnología para también hacer mi proyecto con él.

Obviamente, aplicamos la formación de programación que recibimos de enseñanza al inicio. Además, use los conocimientos en hacer diagramas UML como caso de uso o de dominio para organizar el proyecto y poder poner términos en común con mi compañero trabajando en el frontend. Aplicamos los conocimientos que adquirimos de gestión de empresa para hacer el presupuesto y el plan de trabajo. Usamos las buenas prácticas que pudimos usar en este backend. Usamos uno de los patrones arquitectónicos aprendidos en la carrera. Aplicamos los conocimientos que obtuvimos sobre la base de datos para aprender a hacer las peticiones y a trabajar con ellas. Aprendimos como usar Scrum y lo usamos para gestionar de una manera productiva el proyecto

8. Trabajos futuros

Se seguirá trabajando en este proyecto en un futuro, debido a los vínculos personales con el cliente LUIS SANROMA y a la motivación personal de los dos estudiantes involucrados en los dos TFGs relacionados.

Ha habido algunas funciones que no nos dio tiempo a incluir y que se pretenden llevar a cabo, como es el caso de la gestión de pagos. En un futuro, nos gustaría que la aplicación contara con un apartado donde se le facilitara a la gente pagar las sesiones con el entrenador. Se seguirán incluyendo extensiones y arreglando fallos y bugs, además de modificar la aplicación en función del feedback que reciba de las personas que entrene el cliente.

Además, también se pretende mejorar el middleware, ampliando las funcionalidades de este, haciendo que compruebe que los datos pasados son correctos, etc.

9. Referencias

[1] Hendricks, S. (2023). Nike Training Club App Review. *Reviewed*. https://reviewed.usatoday.com/health/content/nike-training-club-review-workout-app

[2] *Nike Training Club App. Home Workouts & More.* (s. f.). Nike.com. https://www.nike.com/ntc-app

[3] Drum, F. (2023, 20 junio). MyFitnessPal Review 2023 - Everything You Need to Know | Fitness Drum. *Fitness Drum*. https://fitnessdrum.com/myfitnesspal-review/

[4] *MyFitnessPal | MyFitnessPal*. (s. f.). https://www.myfitnesspal.com/es

[5] Cissn, G. R. M. C. (2023). JEFIT Workout App Critical Review and Alternative for Smart Lifters in 2022. *Dr. Muscle*. https://dr-muscle.com/jefit-review-alternative/

[6] Jefit, Inc. (2023, 20 junio). *Home | Jefit - #1 Gym / Home workout app*. Jefit - #1 Gym / Home workout app. https://www.jefit.com/

[7] colaboradores de Wikipedia. (s. f.-b). *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/

[8] *What is Scrum?* (s. f.). Scrum.org. https://www.scrum.org/learning-series/what-is-scrum

[9] *Stack Overflow - Where Developers Learn, Share, & Build Careers*. (s. f.-b). Stack Overflow. https://stackoverflow.com/

[10] Euroinnova Formación. (2022). Aplicaciones de escritorio. *Euroinnova Business School*. https://www.euroinnova.edu.es/blog/aplicaciones-de-escritorio

[11] *¿Qué es una aplicación web? - Explicación de las aplicaciones web - AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/web-application/

[12] BBC News Mundo. (2011, 9 abril). Qué son las «apps» y para qué sirven. *BBC News Mundo*. https://www.bbc.com/mundo/noticias/2011/04/110408_1336_tecnologia_apps_negocios_celulares_telefonos_inteligentes_dc#:~:text=Una%20app%20es%20un%20programa,con%20el%20paso%20del%20tiempo.

[13] *Arquitectura en Capas*. (s. f.). https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

[14] MongoDB. (s. f.). *MongoDB: The Developer Data Platform*. https://www.mongodb.com/

[15] Juanda. (s. f.). *Creación de una API con node.js · Desarrollo de aplicaciones web*. https://juanda.gitbooks.io/webapps/content/api/creacion_de_una_api_con_nodejs.html

[16] Eduardo. (2023). Dotenv: variables de entorno Node js. *Eduardo Arias*. https://eduardo-arias.com/dotenv-variables-de-entorno-node-js/

[17] *Acerca | Node.js*. (s. f.). Node.js. https://nodejs.org/es/about

[18] Sergiodxa. (2017). Introducción a JSON Web Tokens (JWT). *Platzi*. https://platzi.com/blog/introduccion-json-web-tokens/

[19] *npm: bcrypt*. (s. f.). npm. https://www.npmjs.com/package/bcrypt

[20] *Cross-Origin Resource Sharing (CORS) - HTTP | MDN*. (2023, 10 mayo). https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[21] *Express - Infraestructura de aplicaciones web Node.js*. (s. f.). https://expressjs.com/es/

[22] *Mongoose ODM v7.3.1*. (s. f.). https://mongoosejs.com/

[23] *nodemon*. (s. f.). https://nodemon.io/

[24] *Postman*. (s. f.). https://www.postman.com/

10. Glosario

URL: Uniform Resource Locator. Es una secuencia de caracteres que se utiliza paraidentificar y localizar recursos en la web. Una URL típicamente incluye el protocolo utilizado para acceder al recurso (como HTTP o HTTPS), el nombre de dominio, la ruta del recurso y, opcionalmente, parámetros adicionales.

URI: Uniform Resource Identifier. Es una secuencia de caracteres que identifica un recurso en particular, ya sea en la web o en otros sistemas. Una URI puede ser una URL, pero también puede ser un identificador único para otros tipos de recursos, como correos electrónicos o documentos.

JSON: JavaScript Object Notation. Es un formato de intercambio de datos ligero y fácil de leer que se utiliza comúnmente en la comunicación entre servidores y clientes web.

Backend: También conocido como el lado del servidor, se refiere a la parte de una aplicación web o sistema que se encarga del procesamiento y almacenamiento de datos, la lógica de negocio y la comunicación con la base de datos.

Frontend: También conocido como el lado del cliente, se refiere a la parte de una aplicación web o sistema que interactúa directamente con los usuarios.

API: Application Programming Interface. Es un conjunto de reglas y protocolos que permite la comunicación y la interacción entre diferentes componentes de software.

HTTP: HyperText Transfer Protocol. Es el protocolo utilizado para la transferencia de datos en la web. HTTP define la forma en que los mensajes se formatean y

transmiten entre clientes (como navegadores web) y servidores. Permite solicitar y enviar recursos, como páginas web, a través de direcciones URL.

Endpoint: Es un punto final o URL específico en una API que se utiliza para acceder a un recurso o realizar una operación específica. Los endpoints son las URL a las que se envían las solicitudes para interactuar con una API y obtener una respuesta correspondiente.

CRUD: Es un acrónimo que representa las operaciones básicas realizadas en la gestión de datos en sistemas informáticos. CRUD significa Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar). Estas operaciones describen las acciones comunes para crear, leer, actualizar y eliminar datos en una base de datos u otro sistema de almacenamiento.

Gestor de paquetes: Es una herramienta de software que se utiliza para gestionar la instalación, actualización y desinstalación de paquetes de software.

Query: En el contexto de bases de datos y lenguajes de consulta como SQL, una query (consulta) es una solicitud o instrucción enviada a una base de datos para recuperar o manipular datos.

Base de datos: Las bases de datos se utilizan para almacenar y administrar grandes volúmenes de información de manera eficiente, permitiendo el acceso, la manipulación y el análisis de los datos de forma segura y consistente.

Anexo 1: ODS

Los Objetivos de Desarrollo Sostenible (ODS) son un conjunto de metas globales adoptadas por la Asamblea General de las Naciones Unidas en septiembre de 2015 como parte de la Agenda 2030 para el Desarrollo Sostenible.

Entre los 17 objetivos que se presentan, se encuentra el objetivo relacionado con la "Salud y bienestar", y debido a que esta aplicación web es destinada para el entrenamiento de personas con enfermedades degenerativas y restricciones de movimiento, que son las personas que entrena el cliente, estaríamos ayudando a que las personas menos favorecidas en este campo recibiesen algo más de ayuda.

Con esta aplicación web también intentamos influenciar en el objetico de la "Reducción de las desigualdades" debido a que ayudaremos a que las personas que viven en lugares más remotos como pueblos no muy cercanos a ciudades o pueblos grandes puedan tener una atención algo más parecida a los que viven en grandes o medianos centros urbanos

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Proceede**
ODS 1. **Fin de la pobreza.**			**x**	
ODS 2. **Hambre cero.**				**x**
ODS 3. **Salud y bienestar.**	**x**			
ODS 4. **Educación de calidad.**		**x**		
ODS 5. **Igualdad de género.**			**x**	
ODS 6. **Agua limpia y saneamiento.**				**x**
ODS 7. **Energía asequible y no contaminante.**				**x**
ODS 8. **Trabajo decente y crecimiento económico.**		**x**		
ODS 9. **Industria, innovación e infraestructuras.**			**x**	
ODS 10. **Reducción de las desigualdades**.		**x**		
ODS 11. **Ciudades y comunidades sostenibles**.				**x**
ODS 12. **Producción y consumo responsables.**				**x**
ODS 13. **Acción por el clima.**				**x**
ODS 14. **Vida submarina.**				**x**
ODS 15. **Vida de ecosistemas terrestres.**				**x**

ODS 16. **Paz, justicia e instituciones sólidas.**			**x**	
ODS 17. **Alianzas para lograr objetivos.**		**x**		