

GUÍA TRABAJO FIN DE GRADO



IT FITLAB

DESARROLLO DE APLICACIONES WEB

Nombre: Imanol Trespaderne Barón

Curso Académico: 2024-2025

Centro: I.E.S. Alixar





ÍNDICE

Introducción	3
Identificación de las necesidades del proyecto.	4
Breve análisis/comparativa con las alternativas del mercado	6
Justificación del proyecto	7
Stack tecnológico	9
Modelo de Datos.	10
Prototipo de la Aplicación Web.	14
Definición API REST publicación servicios	16
Manual de Despliegue	22
- DESPLIEGUE EN ENTORNO DE DESARROLLO.	22
- DESPLIEGUE EN ENTORNO DE PRODUCCIÓN	29
Conclusiones del proyecto.	52



Introducción

Desde hace tiempo he estado reflexionando sobre cómo unir mis dos pasiones: el deporte y la programación. Con el proyecto "IT FitLab", mi objetivo es materializar esa fusión de intereses en una aplicación fullstack que ofrezca soluciones prácticas tanto para el ámbito fitness como para el nutricional. La idea central es desarrollar una herramienta que permita programar entrenamientos completos, desde la selección de ejercicios hasta la definición de series y repeticiones, y al mismo tiempo, gestionar dietas de forma personalizada. Esto incluye funciones como el cálculo del metabolismo basal, el seguimiento del consumo de kilocalorías y la selección de alimentos orientados a formular dietas adaptadas a cada necesidad.

La motivación para este proyecto proviene de mi propio camino profesional y personal. Hace dos años decidí reorientar mi futuro laboral hacia la informática, pero sin perder mi conexión con el deporte, un ámbito que siempre me ha impulsado a superarme. "IT FitLab" se presenta como la oportunidad perfecta para integrar estas dos áreas, permitiéndome cerrar el círculo entre el mundo tecnológico y el deportivo, y ofreciendo a los usuarios una forma real de mejorar tanto su salud como su rendimiento físico.

La aplicación está pensada para ser fácil de usar y accesible para cualquier persona interesada en llevar un estilo de vida más saludable, independientemente de su nivel de experiencia. Además, el proyecto contempla un stack tecnológico moderno, que incluye herramientas como Figma para el diseño, MySQL como base de datos relacional, Spring Boot en el backend y Angular complementado con Bootstrap/Tailwind en el frontend. Con este conjunto de tecnologías se garantiza un sistema escalable, seguro y eficiente, ideal para afrontar los retos de un entorno en constante evolución.

Identificación de las necesidades del proyecto.

En la actualidad, cada vez es más habitual que las personas quieran llevar un estilo de vida más saludable, pero carecen de herramientas que les permitan organizar adecuadamente tanto su alimentación como su entrenamiento físico. A pesar del aumento de aplicaciones móviles en el ámbito del fitness y la nutrición, la mayoría de soluciones disponibles en el mercado se encuentran divididas. Algunas se enfocan exclusivamente en la planificación deportiva, mientras que otras centran sus funcionalidades en el control dietético, sin ofrecer una integración real entre ambos aspectos.

Este enfoque parcial limita al usuario, ya que impide obtener una visión completa de su progreso, dificultando la toma de decisiones respecto a su salud. Por ejemplo, una persona que inicia un programa de pérdida de peso debe poder relacionar su plan de entrenamiento con el número de calorías consumidas diariamente, ajustando su alimentación de forma dinámica según su nivel de actividad física y objetivos concretos. La falta de plataformas que ofrezcan este nivel de integración representa una barrera significativa.

Otro problema común en las aplicaciones existentes es la escasa personalización: muchas soluciones solo ofrecen recomendaciones genéricas o requieren una suscripción premium para acceder a funcionalidades avanzadas. A menudo no consideran variables fisiológicas clave como el metabolismo basal, el nivel de actividad o el historial del usuario, lo que reduce considerablemente la efectividad de las recomendaciones ofrecidas.

Por tanto, surge la necesidad de desarrollar una solución intuitiva, flexible y completamente personalizada que permita a cada usuario gestionar su bienestar de forma autónoma, combinando entrenamiento y nutrición en una única interfaz. Esta herramienta debe ser accesible desde cualquier dispositivo y contar con una experiencia de usuario clara y fluida, adaptada tanto a perfiles principiantes como avanzados.



A partir del análisis anterior, se definen los siguientes requisitos funcionales del sistema:

RF01. El sistema permitirá a los usuarios registrarse, iniciar sesión y acceder a su panel de perfil.

RF02. El sistema contará con control de acceso basado en roles (administrador y usuario normal).

RF03. Los usuarios podrán introducir sus datos fisiológicos (peso, altura, edad, sexo, nivel de actividad) para el cálculo automático de su metabolismo total (TDEE).

RF04. La aplicación permitirá crear, visualizar, editar y eliminar dietas personalizadas organizadas por día de la semana y tipo de comida.

RF05. Se podrán añadir alimentos a las dietas, con su información nutricional detallada (calorías, proteínas, carbohidratos, grasas).

RF06. El sistema mostrará un resumen diario de calorías consumidas y macronutrientes en cada dieta.

RF07. La plataforma permitirá crear entrenamientos personalizados, especificando series, repeticiones, peso y dificultad del entrenamiento.

RF08. Los usuarios podrán consultar programas de entrenamiento genéricos predefinidos por los administradores.

RF09. Los usuarios podrán valorar los programas de entrenamiento genéricos y dejar comentarios sobre ellos.

RF10. Se incluirá un gráfico interactivo para visualizar el progreso nutricional.

RF11. Los administradores podrán crear, editar y eliminar alimentos y ejercicios disponibles en la plataforma.

RF12. Los administradores podrán crear a otros usuarios administradores.

RF13. Los administradores podrán editar información de otros usuarios. Así mismo, podrán eliminar y desactivar/reactivar cuentas de otros usuarios.

RF14. La plataforma será accesible desde navegador en cualquier dispositivo.

RF15. La solución podrá desplegarse fácilmente mediante Docker, facilitando la instalación en cualquier entorno.

Breve análisis/comparativa con las alternativas del mercado

Actualmente existen numerosas aplicaciones móviles y plataformas web que ofrecen servicios relacionados con el deporte y la nutrición, tales como **MyFitnessPal**, **Freeletics**, **Fitbit** o **Strong**. Sin embargo, al analizarlas en profundidad se pueden identificar limitaciones que IT FitLab pretende superar:

- **MyFitnessPal**: Su enfoque está principalmente en el seguimiento calórico y registro de alimentos. Si bien tiene una amplia base de datos nutricional, sus herramientas de entrenamiento son limitadas y poco personalizables.
- **Freeletics**: Muy centrada en el entrenamiento funcional, con rutinas predefinidas. No ofrece funcionalidades completas para la gestión de dietas ni personalización basada en datos fisiológicos del usuario.
- **Fitbit**: Es una plataforma potente cuando se combina con sus dispositivos, pero muchos de sus servicios avanzados requieren suscripción premium y no permite modificar o crear rutinas personalizadas desde cero.
- **Strong**: Ideal para usuarios experimentados en musculación, pero sin funcionalidades relacionadas con la nutrición ni seguimiento calórico.

IT FitLab surge como una alternativa que cubre tanto la parte de nutrición como la de entrenamiento, de manera gratuita y adaptable. A diferencia de las anteriores, permite una personalización profunda, el uso de rutinas personalizadas y genéricas, dietas cíclicas, y un seguimiento constante con visualizaciones gráficas modernas. Esto lo convierte en una solución diferenciada, especialmente útil para quienes buscan una gestión unificada de su bienestar físico.

Justificación del proyecto

Descripción del proyecto

El proyecto "IT FitLab" se basa en el desarrollo de una aplicación que facilite la planificación, seguimiento y gestión tanto de entrenamientos físicos como de planes de alimentación personalizados, orientada a mejorar la calidad de vida y el bienestar de sus usuarios. La aplicación se ha creado para ser una herramienta práctica y accesible, permitiendo a cada persona organizar su rutina de ejercicios y ajustar su dieta según necesidades y objetivos individuales. Gracias a una interfaz intuitiva y amigable, se posibilita un seguimiento continuo del progreso, ofreciendo un soporte que se adapte al estilo de vida de cada usuario.

La aplicación abarca dos áreas fundamentales:

- **Gestión de Entrenamientos:** Permite la creación y organización de entrenamientos personalizados basados en ejercicios específicos. Cada usuario puede acceder a rutinas predefinidas o diseñar sus propias rutinas según sus objetivos personales, como pérdida de peso, aumento de masa muscular o mantenimiento.
- **Gestión de Dietas:** Facilita la creación y seguimiento de dietas personalizadas con base en alimentos específicos organizados en días y tipos de comida (desayuno, almuerzo, cena, etc.). La aplicación proporciona herramientas para el control nutricional detallado, incluyendo seguimiento de calorías y distribución de macronutrientes (proteínas, carbohidratos y grasas).

Alcance del proyecto

El alcance del proyecto incluye:

1. **Autenticación y autorización:** Implementación segura de autenticación con gestión de roles diferenciados (admin, normal) para controlar el acceso a funcionalidades específicas.
2. **Gestión de Usuarios:** Registro de usuarios con perfil completo (peso, altura, edad, sexo y nivel de actividad física) para cálculo del metabolismo basal (TDEE) en tiempo real.
3. **Gestión de Entrenamientos:** Creación, actualización y eliminación de entrenamientos. Registro de ejercicios personalizados con descripciones detalladas, permitiendo asignarlos a distintas rutinas según las necesidades del usuario.
4. **Gestión de Dietas:** Funcionalidad para crear dietas semanales repetibles en ciclos, asignar alimentos específicos a días concretos y tipos de comida, además de cálculo y visualización del desglose nutricional diario.



5. **Visualización y análisis de datos:** Inclusión de gráficos interactivos que muestran la distribución de macronutrientes, calorías consumidas y progreso del usuario en tiempo real.
6. **Interfaz de usuario moderna y adaptable:** Diseño intuitivo y minimalista, orientado a dispositivos móviles y adaptado a modo oscuro para mayor comodidad visual del usuario.

Stack tecnológico

El stack tecnológico elegido para este proyecto garantiza escalabilidad, seguridad y rendimiento. Se han seleccionado tecnologías modernas y ampliamente adoptadas en la industria, permitiendo facilidad en el desarrollo, mantenimiento y futuras mejoras.

Backend

- **Java con Spring Boot:** Se emplea Spring Boot para construir APIs REST, gestionando las entidades principales. La persistencia de datos se maneja a través de Spring Data JPA.
- **Base de datos MySQL:** Utilizada con Spring Boot para la gestión de relaciones y consultas eficientes.
- **Docker:** Para contenerizar la aplicación y facilitar despliegues consistentes en distintos entornos.
- **JWT (JSON Web Token):** Para autenticación segura y stateless entre frontend y backend.

Frontend

- **Angular:** Framework principal seleccionado por su facilidad de escalado y gestión eficiente del estado de la aplicación.
- **Tailwind CSS:** Para proporcionar un estilo coherente, minimalista y adaptable, facilitando una experiencia visual moderna y agradable.
- **Heroicons:** Íconos simples y estilizados que complementan la interfaz minimalista del proyecto.
- **Chart.js:** Para visualización gráfica de datos nutricionales ofreciendo información clara y en tiempo real.

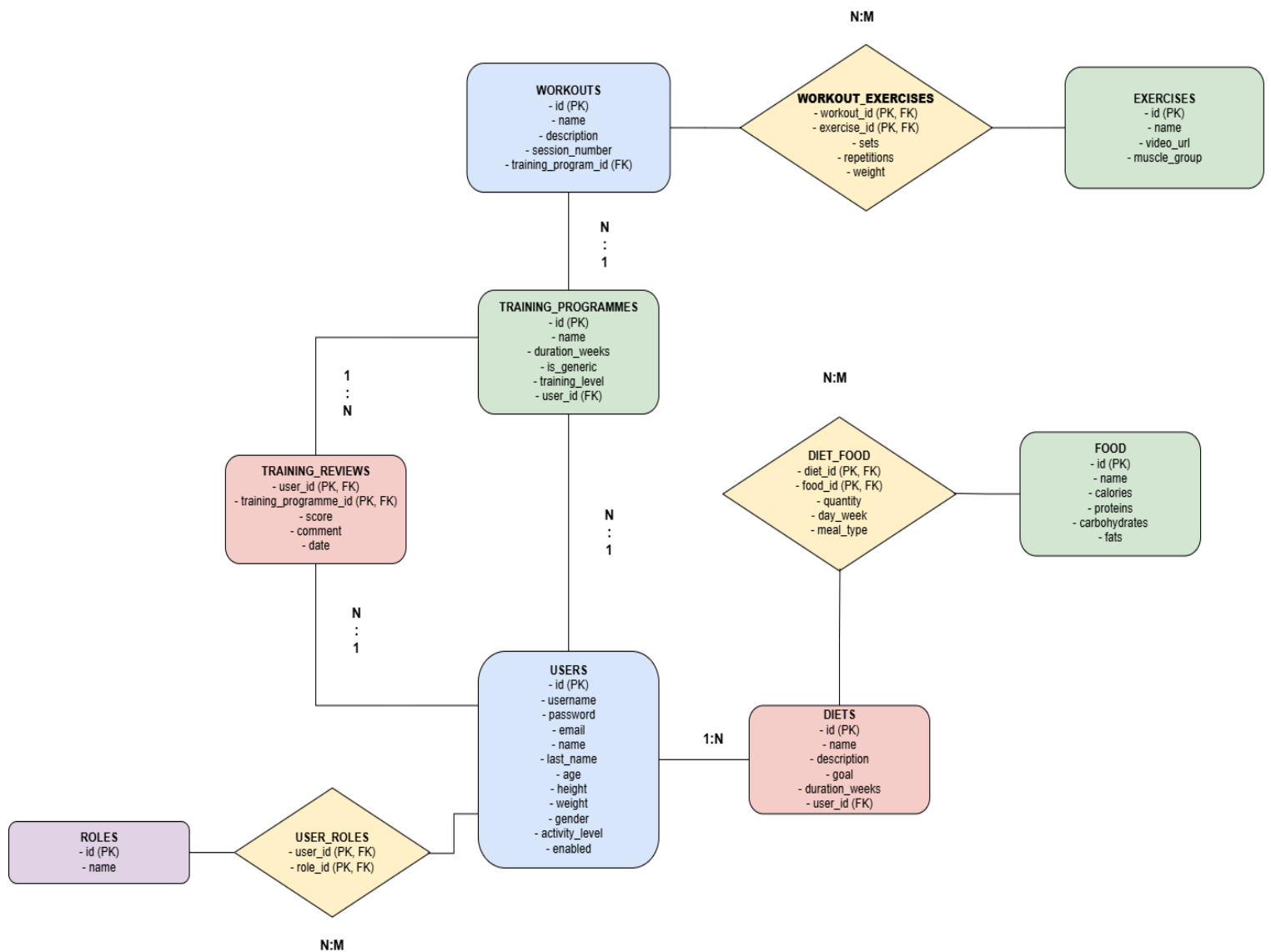
Herramientas adicionales

- **Postman:** Utilizado para pruebas y documentación de las APIs REST desarrolladas, facilitando el testeo y la validación constante del backend.
- **Git/GitHub:** Para el control de versiones del proyecto, asegurando un desarrollo organizado.

Modelo de Datos.

La base de datos IT_FitLab está diseñada para gestionar una plataforma de dietas y entrenamiento personalizada, donde los usuarios pueden seguir dietas, programas de entrenamiento, registrar sus progresos y valorar los programas. La estructura está normalizada y se compone de 11 tablas.

Modelo Entidad Relación.



Estructura y Relaciones por Tabla

- users ↔ roles (N:M mediante user_roles)

Un usuario puede tener varios roles (admin, user, y un rol puede aplicarse a varios usuarios. Se resuelve mediante la tabla intermedia 'user_roles' con clave compuesta (user_id, role_id).

- users ↔ diets (1:N)

Un usuario puede tener varias dietas. Cada dieta pertenece a un único usuario (user_id como FK en 'diets').

- diets ↔ food (N:M mediante diet_food)

Una dieta puede contener muchos alimentos y un alimento puede estar en muchas dietas. Se usa la tabla intermedia 'diet_food', que además contiene día de la semana, tipo de comida y cantidad.

- users ↔ training_programmes (1:N)

Un usuario puede seguir varios programas de entrenamiento personalizados. También existen programas genéricos (is_generic = TRUE).

- training_programmes ↔ workouts (1:N)

Un programa de entrenamiento se compone de múltiples workouts (sesiones de entrenamiento).

- workouts ↔ exercises (N:M mediante workout_exercises)

Un entrenamiento puede incluir muchos ejercicios y un ejercicio puede formar parte de varios entrenamientos. La tabla intermedia 'workout_exercises' especifica número de series, repeticiones y peso.

- users ↔ training_reviews (1:N) y training_programmes ↔ training_reviews (1:N)

Cada usuario puede valorar programas de entrenamiento. Cada valoración está identificada de forma única por user_id y training_programme_id.



MODELO RELACIONAL

ROLES(id, name)

- PK: id

USERS(id, username, password, email, name, last_name, age, height, weight, gender, activity_level, enabled)

- PK: id

USER_ROLES(user_id, role_id)

- PK: user_id, role_id
- FK: user_id REFERENCIA a USERS(id)
- FK: role_id REFERENCIA a ROLES(id)

FOOD(id, name, calories, proteins, carbohydrates, fats)

- PK: id

DIETS(id, name, description, goal, duration_weeks, user_id)

- PK: id
- FK: user_id REFERENCIA a USERS(id)

DIET_FOOD(diet_id, food_id, quantity, day_week, meal_type)

- PK: diet_id, food_id, day_week, meal_type
- FK: diet_id REFERENCIA a DIETS(id)
- FK: food_id REFERENCIA a FOOD(id)

TRAINING_PROGRAMMES(id, name, duration_weeks, user_id, is_generic, training_level)

- PK: id
- FK: user_id REFERENCIA a USERS(id)

WORKOUTS(id, name, description, session_number, training_program_id)

- PK: id
- FK: training_program_id REFERENCIA a TRAINING_PROGRAMMES(id)

EXERCISES(id, name, video_url, muscle_group)

- PK: id




WORKOUT_EXERCISES(workout_id, exercise_id, sets, repetitions, weight)

- PK: workout_id, exercise_id
- FK: workout_id REFERENCIA a WORKOUTS(id)
- FK: exercise_id REFERENCIA a EXERCISES(id)

TRAINING_REVIEWS(user_id, training_programme_id, score, comment, date)

- PK: user_id, training_programme_id
- FK: user_id REFERENCIA a USERS(id)
- FK: training_programme_id REFERENCIA a TRAINING_PROGRAMMES(id)

Diccionario de datos.

 Diccionario_Por_Tabla_IT_FitLab.xlsx

Prototipo de la Aplicación Web.

El prototipo de la aplicación ha sido diseñado mediante Figma, una herramienta de diseño de interfaces en la nube basada en crear prototipos interactivos y maquetas. El objetivo principal ha sido el de definir una experiencia de usuario fluida, moderna y coherente en todo momento. Se ha optado por una interfaz minimalista, con predominio de tonos neutros y elementos gráficos simples para asegurar una navegación cómoda, incluso en sesiones prolongadas. Asimismo, se prioriza el diseño responsive para su correcta visualización en dispositivos móviles, tablets y ordenadores.

El prototipo incluye:

- **Pantalla de inicio y registro/login**, con campos validados y diseño accesible.
- **Navegación superior** sencilla, con enlaces a las secciones principales: Home, Entrenamientos y Dietas, además de iconos de acceso a modo oscuro y perfil de usuario.
- **Landing page visual**, con imagen de fondo en blanco y negro, eslogan motivacional destacado y logotipo de la aplicación en la parte superior izquierda.
- **Vista de entrenamientos**, con acceso a rutinas genéricas o personalizadas, sistema de valoraciones y editor para la creación de programas adaptados a distintos objetivos.
- **Vista de dietas**, donde el usuario puede gestionar los alimentos por día y tipo de comida, visualizar resúmenes nutricionales diarios y consultar gráficos de calorías y macronutrientes.
- **Perfil de usuario**, que permite editar datos personales clave como peso, altura, edad, sexo y nivel de actividad física, los cuales son utilizados para calcular el metabolismo basal (TDEE) en tiempo real.
- **Vista de admin**, que permite que los usuarios administradores puedan gestionar a otros usuarios, ejercicios y alimentos.

https://www.figma.com/design/RvMbfXEPkCU8ZnPY2FmOuS/IT_Fitlab?node-id=15-1439&t=UouKeddy8Tjp2byz-0



Este prototipo ha servido como base para la implementación final en Angular, facilitando tanto la maquetación como la integración de funcionalidades en el frontend.

Definición API REST publicación servicios

La aplicación IT FitLab expone una API RESTful que permite gestionar todos los recursos necesarios para el funcionamiento de la plataforma: usuarios, entrenamientos, dietas, alimentos, ejercicios, relaciones entre sesiones y ejercicios, reseñas, autenticación y archivos multimedia.

Usuarios

Método	Endpoint	Descripción
GET	/api/v1/user/{id}	Obtener un usuario por ID
PUT	/api/v1/user/{id}	Actualizar los datos de un usuario
DELETE	/api/v1/user/{id}	Eliminar un usuario
POST	/api/v1/user/register	Registrar un nuevo usuario
POST	/api/v1/user/admin/register	Registrar un nuevo administrador
PATCH	/api/v1/user/{id}/deactivate	Desactivar un usuario
PATCH	/api/v1/user/reactivate/{email}	Reactivar un usuario por email
GET	/api/v1/user	Obtener todos los usuarios
GET	/api/v1/user/username/{username}	Buscar usuario por nombre de usuario
GET	/api/v1/user/email/{email}	Buscar usuario por email
GET	/api/v1/user/check-username/{username}	Verificar disponibilidad de username
GET	/api/v1/user/check-email/{email}	Verificar disponibilidad de email
GET	/api/v1/user/tdee	Calcular el TDEE del usuario autenticado



Workouts

Método	Endpoint	Descripción
GET	/api/v1/workouts/{id}	Obtener workout por ID
PUT	/api/v1/workouts/{id}	Actualizar workout existente
DELETE	/api/v1/workouts/{id}	Eliminar workout por ID
GET	/api/v1/workouts	Obtener todos los workouts
POST	/api/v1/workouts	Crear un nuevo workout
GET	/api/v1/workouts/{id}/workouts	Obtener workouts por ID de programa

Programas de Entrenamiento

Método	Endpoint	Descripción
GET	/api/v1/trainingProgrammes/{id}	Obtener programa de entrenamiento por ID
PUT	/api/v1/trainingProgrammes/{id}	Actualizar programa de entrenamiento
DELETE	/api/v1/trainingProgrammes/{id}	Eliminar programa de entrenamiento
GET	/api/v1/trainingProgrammes	Obtener todos los programas de entrenamiento
POST	/api/v1/trainingProgrammes	Crear nuevo programa de entrenamiento
GET	/api/v1/trainingProgrammes/user/{userId}	Obtener programas por ID de usuario
GET	/api/v1/trainingProgrammes/generic	Obtener programas genéricos



Dietas

Método	Endpoint	Descripción
GET	/api/v1/diets/{id}	Obtener dieta por ID
PUT	/api/v1/diets/{id}	Actualizar dieta existente
DELETE	/api/v1/diets/{id}	Eliminar dieta
GET	/api/v1/diets	Obtener todas las dietas
POST	/api/v1/diets	Crear una nueva dieta
GET	/api/v1/diets/{id}/day/{dayOfWeek}	Obtener alimentos por día
GET	/api/v1/diets/user	Obtener dietas del usuario autenticado

Alimentos

Método	Endpoint	Descripción
GET	/api/v1/food/{id}	Obtener alimento por ID
PUT	/api/v1/food/{id}	Actualizar alimento existente
DELETE	/api/v1/food/{id}	Eliminar alimento
GET	/api/v1/food	Obtener todos los alimentos
POST	/api/v1/food	Crear nuevo alimento



Método	Endpoint	Descripción
PUT	/api/v1/dietfood/{dietId}/{dayWeek}/ {mealType}	Reemplazar alimentos en una dieta
POST	/api/v1/dietfood	Añadir alimento a dieta
GET	/api/v1/dietfood/{dietId}/{dayWeek}	Obtener alimentos por día de dieta
DELETE	/api/v1/dietfood/{dietId}/{foodId}	Eliminar alimento de una dieta

Método	Endpoint	Descripción
GET	/api/v1/exercises/{id}	Obtener ejercicio por ID
PUT	/api/v1/exercises/{id}	Actualizar ejercicio existente
DELETE	/api/v1/exercises/{id}	Eliminar ejercicio
GET	/api/v1/exercises	Obtener todos los ejercicios
POST	/api/v1/exercises	Crear nuevo ejercicio

Sesiones de Entrenamiento

Método	Endpoint	Descripción
GET	/api/v1/workoutexercises/{workoutId}/ {exerciseld}	Obtener relación Workout-Exercise
PUT	/api/v1/workoutexercises/{workoutId}/ {exerciseld}	Actualizar relación Workout-Exercise
DELETE	/api/v1/workoutexercises/{workoutId}/ {exerciseld}	Eliminar relación Workout-Exercise
GET	/api/v1/workoutexercises	Obtener todas las relaciones Workout-Exercise
POST	/api/v1/workoutexercises	Crear relación Workout-Exercise
GET	/api/v1/workoutexercises/workout/ {workoutId}	Obtener ejercicios de un workout

Reseñas de Entrenamientos

Método	Endpoint	Descripción
PUT	/api/v1/reviews/{reviewId}	Actualizar una reseña
DELETE	/api/v1/reviews/{reviewId}	Eliminar una reseña
GET	/api/v1/reviews	Obtener todas las reseñas
POST	/api/v1/reviews	Crear una nueva reseña
GET	/api/v1/reviews/programme/ {programmeld}	Obtener reseñas de un programa



Archivos Multimedia

Método	Endpoint	Descripción
GET	/api/v1/videos/{filename}	Obtener video por nombre de archivo

Autenticación

Método	Endpoint	Descripción
POST	/api/v1/login	Iniciar sesión y obtener token JWT

Durante el desarrollo y las pruebas en entorno local, se habilita el acceso a la documentación interactiva de la API mediante Swagger UI, accesible a través del navegador (<http://localhost:8080/swagger-ui/index.html#/>). Esta interfaz permite visualizar todos los endpoints disponibles, junto con sus métodos HTTP, parámetros y ejemplos de respuesta, facilitando la validación y testeo de los servicios expuestos por la aplicación.

Sin embargo, por motivos de seguridad, esta funcionalidad está deshabilitada en entornos de producción. El motivo principal es evitar que terceros puedan visualizar la estructura completa de la API y acceder a rutas sensibles o no documentadas públicamente. De esta forma, reducimos el riesgo de exposición no autorizada de recursos.

Manual de Despliegue

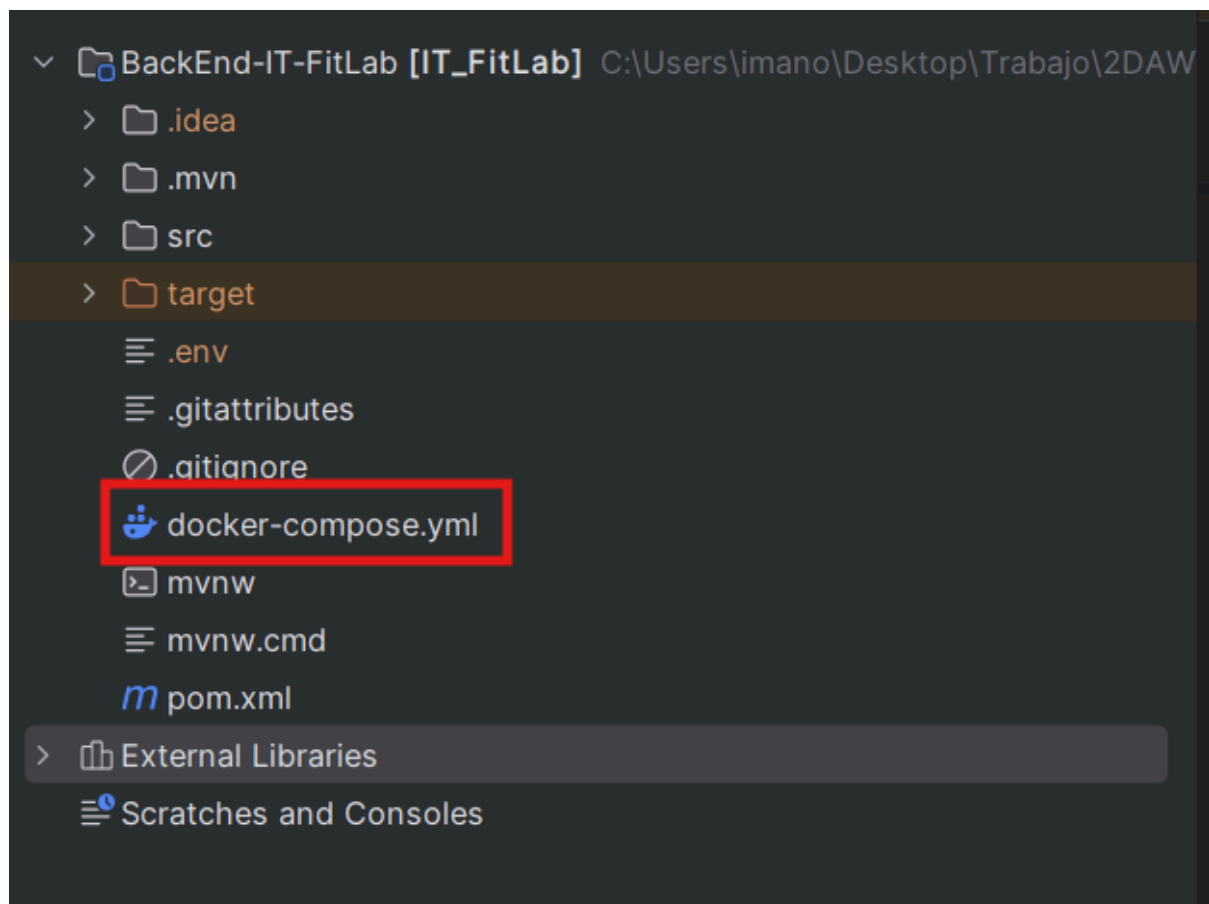
DESPLIEGUE EN ENTORNO DE DESARROLLO.

Para el despliegue en el entorno local haremos uso de docker. En este caso, al disponer de un entorno gráfico usaremos la herramienta de docker Desktop para ver los contenedores que tenemos desplegados. Además, docker desktop ya incluye el docker engine y docker compose, por lo que nos ahorraremos pasos en el despliegue para instalarlo. Dependiendo del sistema operativo deberemos seguir unos pasos u otros. [Descargar Docker.](#)

Una vez tengamos instalado Docker Desktop, iremos al IDE donde configuramos el backend, en mi caso IntelliJ. Los pasos a seguir son los siguientes:

- Creación archivo docker-compose.yml

Este archivo lo crearemos en la raíz del proyecto, a la misma altura que el pom.xml



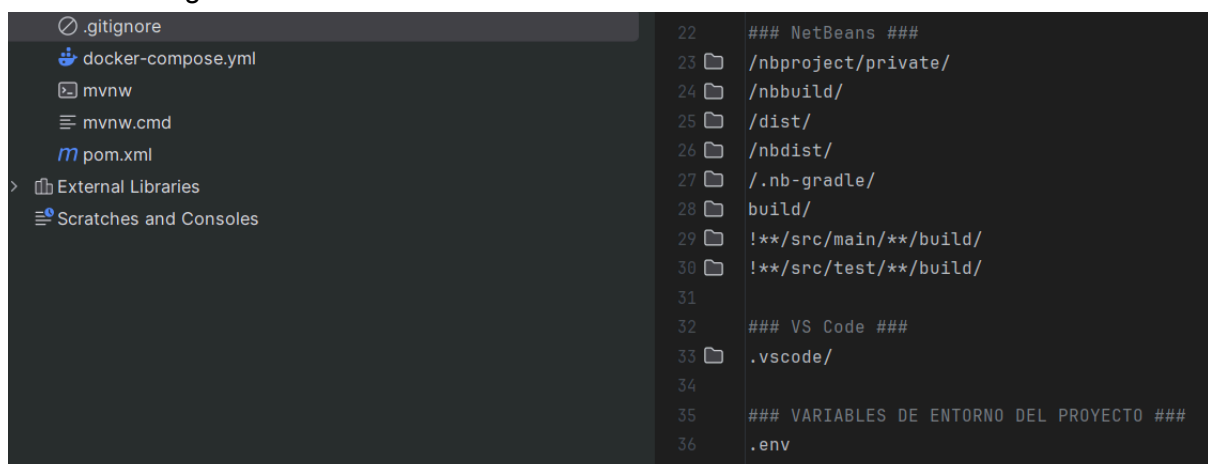
Este archivo, contiene un servicio que lanza una base de datos MariaDB lista para ser usada. Las variables de entorno las crearemos mediante un .env y guardaremos los datos en un volumen para no perderlos al reiniciar la aplicación.

```
services:
  db:
    image: mariadb:latest
    environment:
      - MARIADB_ROOT_PASSWORD=${DB_ROOT_PASSWORD}
      - MARIADB_DATABASE=${DB_DATABASE}
      - MARIADB_USER=${DB_USER}
      - MARIADB_PASSWORD=${DB_PASSWORD}
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql

volumes:
  db_data:
```

- Crear archivo .env y añadirlo a .gitignore

Como hemos dicho previamente, las variables de entorno las almacenaremos en un .env. Importante: Antes de crear el .env en el .gitignore debemos incluir este archivo que vamos a crear. De esta manera, cuando subamos los cambios a github no se comparta el .env por motivos de seguridad.



```
22  ### NetBeans ###
23  /nbproject/private/
24  /nbbuild/
25  /dist/
26  /nbdist/
27  /.nb-gradle/
28  build/
29  !**/src/main/**/build/
30  !**/src/test/**/build/
31
32  ### VS Code ###
33  .vscode/
34
35  ### VARIABLES DE ENTORNO DEL PROYECTO ###
36  .env
```

El .env quedaría de la siguiente manera:

```
DB_DRIVER=org.mariadb.jdbc.Driver
DB_URL=jdbc:mariadb://localhost:3306/IT_FitLab
DB_ROOT_PASSWORD=root_password
DB_DATABASE=IT_FitLab
DB_USER=itrebar
DB_PASSWORD=itrebar
JWT_TOKEN=B397!|S//_3mLa1Ñi%3DHI`6Vgae(+lp
JWT_KEYSTORE_PATH=C:\Users\imano\Desktop\Trabajo\2DAW\certs\jwt-keystore.jks
JWT_KEYSTORE_PASSWORD=itrebar
JWT_KEYSTORE_ALIAS=jwt-keypair
CORS_ALLOWED_ORIGINS=http://localhost:4200
UPLOAD_PATH=C:\Users\imano\Desktop\Trabajo\2DAW\it_fitlab\videos
```

Aparte de las variables de la base de datos de las que hace uso el docker-compose.yml almacenaremos otras variables que por seguridad no queremos mostrar en el código de nuestra aplicación. Estas variables de entorno las cargaremos en el application properties.

```
spring.datasource.url=${DB_URL}
spring.datasource.username=${DB_USER}
spring.datasource.password=${DB_PASSWORD}
spring.datasource.driver-class-name=${DB_DRIVER}
```

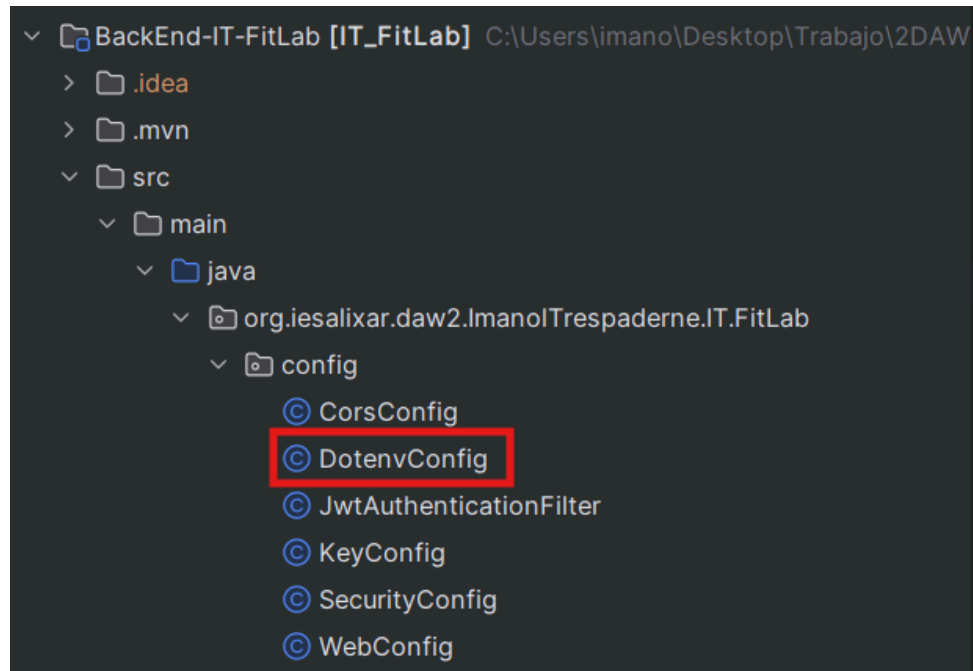
```
jwt.secret=${JWT_TOKEN}
jwt.keystore.path= ${JWT_KEYSTORE_PATH}
jwt.keystore.password= ${JWT_KEYSTORE_PASSWORD}
jwt.keystore.alias= ${JWT_KEYSTORE_ALIAS:jwt-keypair}

cors.allowed-origins= ${CORS_ALLOWED_ORIGINS}
upload.path=${UPLOAD_PATH}
```




- Clase de configuración Dotenv.

Por último, debemos de añadir una clase de configuración para que cada vez que arranque la aplicación lo primero que se haga sea cargar las variables de entorno:



```
@Configuration 1 usage 1 ImanolTB
public class DotenvConfig {

    // Se crea un logger para la clase utilizando SLF4J para registrar eventos importantes y posibles errores
    private static final Logger logger = LoggerFactory.getLogger(DotenvConfig.class); 4 usages

    // Bloque estático para inicializar y cargar las variables de entorno del archivo .env
    static {
        try {
            // Registrar en el log un mensaje informativo indicando que el proceso de carga del archivo .env ha comenzado
            logger.info("Loading environment variables from .env file...");

            // Configuración y carga del archivo .env usando la librería Dotenv
            Dotenv dotenv = Dotenv.configure().load();

            // Iterar sobre todas las entradas (clave-valor) del archivo .env y establecerlas como propiedades del sistema
            dotenv.entries().forEach( DotenvEntry entry -> {
                // Establecer cada variable de entorno como una propiedad del sistema
                System.setProperty(entry.getKey(), entry.getValue());

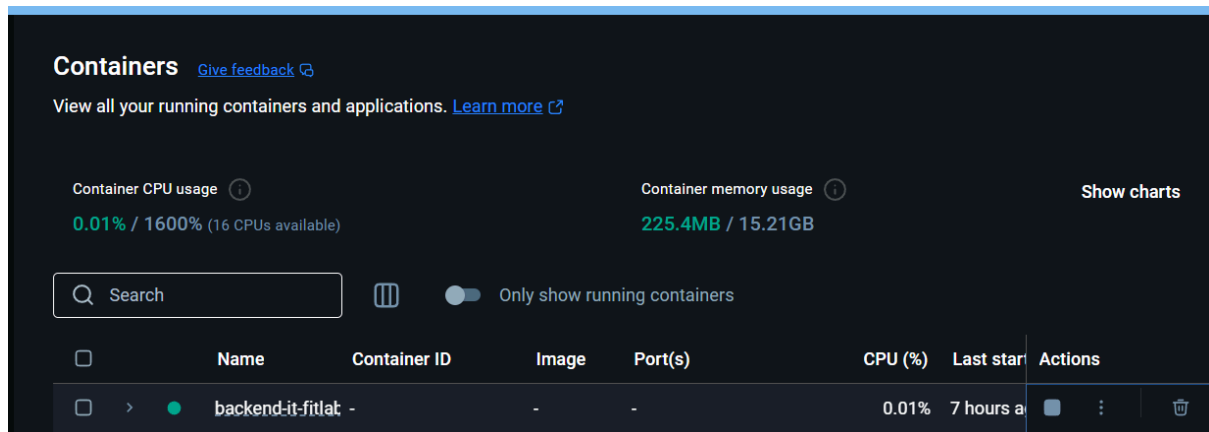
                // Registrar en el log el valor de cada variable cargada para fines de depuración (nivel DEBUG)
                logger.debug("Set system property: {} = {}", entry.getKey(), entry.getValue());
            });

            // Registrar en el log un mensaje informativo indicando que el archivo .env se cargó correctamente
            logger.info(".env file loaded successfully.");
        } catch (Exception e) {
            // Si ocurre alguna excepción durante el proceso de carga del archivo .env, se captura y se registra un mensaje de error
            logger.error("Failed to load .env file", e);
        }
    }
}
```

Para hacer uso de la clase dotenv tendremos que instalar dependencias específicas del repositorio central de maven.

```
<dependency>
    <groupId>io.github.cdimascio</groupId>
    <artifactId>dotenv-java</artifactId>
    <version>3.0.2</version>
</dependency>
```

Una vez tengamos todo configurado, tan solo tenemos que levantar el contenedor mediante el comando docker compose up -d, para ello debemos estar en el directorio que contiene el docker-compose.yml. Una vez levantado, nos vamos a Docker Desktop y nos aseguramos de que esté corriendo.



Lo siguiente será arrancar el backend mediante el comando:

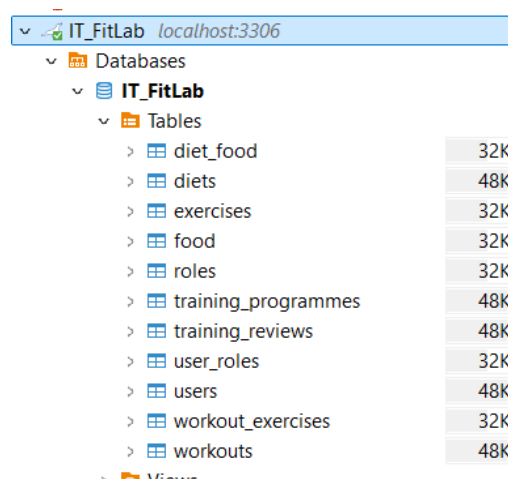
```
./mvnw spring-boot:run
```

```
o.i.d.I.IT.FitLab.config.DotenvConfig : Loading environment variables from .env file...
o.i.d.I.IT.FitLab.config.DotenvConfig : .env file loaded successfully.
```

```
2025-06-02T16:26:31.311+02:00 INFO 65596 --- [IT FitLab] [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 11.6.2
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
```

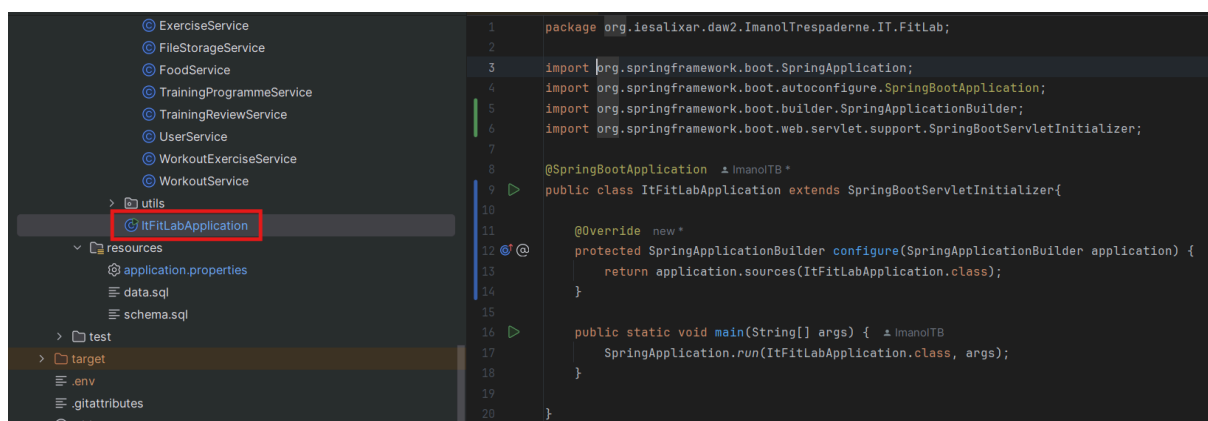
Podemos observar como se ha cargado el .env de manera exitosa y el despliegue de la bbdd.

Podemos usar un cliente de base de datos (dbeaver) para visualizar que la base de datos se haya creado con éxito.



IT_FitLab localhost:3306	
Database	
IT_FitLab	
Tables	
diet_food	32K
diets	48K
exercises	32K
food	32K
roles	32K
training_programmes	48K
training_reviews	48K
user_roles	32K
users	48K
workout_exercises	32K
workouts	48K

Cabe destacar que no hemos tenido que desplegar un contenedor con el servidor, ya que en este caso hemos hecho uso del servidor embebido que trae el propio IDE. En caso de querer desplegar en un servidor a parte, tendríamos que añadir en la clase de arranque del proyecto que el servidor va a ser uno distinto al embebido por el IDE, como haremos en el entorno de producción, de la siguiente manera:



```
1 package org.iesalixar.daw2.ImanolTrespaderne.IT.FitLab;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.boot.builder.SpringApplicationBuilder;
6 import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
7
8 @SpringBootApplication
9 public class ItFitLabApplication extends SpringBootServletInitializer{
10
11     @Override
12     protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
13         return application.sources(ItFitLabApplication.class);
14     }
15
16     public static void main(String[] args) {
17         SpringApplication.run(ItFitLabApplication.class, args);
18     }
19
20 }
```



DESPLIEGUE EN ENTORNO DE PRODUCCIÓN

1. Preparando el VPS

- [Actualización](#) del sistema operativo a Ubuntu Server 24.04 LTS.

Para actualizar el sistema operativa lanzaremos los siguientes comandos:

Para ver la versión de ubuntu actual:

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.1 LTS
Release:        24.04
Codename:       noble
```

Para actualizar los paquetes disponibles desde los repositorios configurados en el sistema.(Este comando ni instala ni actualiza paquetes, solo descarga información sobre las versiones más recientes de los paquetes disponibles)

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ sudo apt update
[sudo] password for itrebar:
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1028 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:10 http://archive.ubuntu.com/ubuntu noble-security/main amd64 Components [8988 B]
Get:11 http://archive.ubuntu.com/ubuntu noble-security/restricted amd64 Components [208 B]
Get:12 http://archive.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Fetched 1857 kB in 2s (922 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
73 packages can be upgraded. Run 'apt list --upgradable' to see them.
```



Mediante el siguiente comando actualizaremos todos los paquetes instalados en el sistema a sus versiones más recientes.

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  base-files bsdxtrutils bsdxtrutils containerd.io dmsetup
  libcryptsetup12 libdevmapper1.02.1 libdw1t64 libelf1t64
  libpackagekit-glib2-18 libpam-systemd libpcre2-8-0 lib
  netplan-generator netplan.io packagekit packagekit-too
  systemd-timesyncd ubuntu-minimal ubuntu-release-upgrad
73 upgraded, 0 newly installed, 0 to remove and 0 not up
Need to get 146 MB of archives.
After this operation, 8332 kB disk space will be freed.
```

A continuación lanzaremos el comando `sudo apt dist-upgrade` para actualizar paquetes e instalar o eliminar paquetes en caso de que sea necesario para resolver dependencias.

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ sudo apt dist-upgrade
[sudo] password for itrebar:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Una vez están los paquetes actualizados, lanzaremos el siguiente comando:
`sudo do-release-upgrade`

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ sudo do-release-upgrade
Checking for a new Ubuntu release
There is no development version of an LTS available.
To upgrade to the latest non-LTS development release
set Prompt=normal in /etc/update-manager/release-upgrades.
itrebar@2DAW-itrebar-UbuntuLXC:~$
```

Volvemos a comprobar la versión de ubuntu.

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.2 LTS
Release:        24.04
Codename:       noble
itrebar@2DAW-itrebar-UbuntuLXC:~$
```

Ahora tenemos la última versión LTS de Ubuntu.

- [Instalación de Docker](#) Engine y Docker Compose en Ubuntu.

Ya teníamos docker instalado previamente en esta máquina.

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ docker --version
Docker version 28.0.1, build 068a01e
```

- Crear un usuario con tu nombre y añadirlo a grupos sudo y docker.

Ya teníamos un usuario en un grupo sudo y docker.

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ getent group docker
docker:x:989:itrebar
```

```
itrebar@2DAW-itrebar-UbuntuLXC:~$ id
uid=1000(itrebar) gid=1000(itrebar) groups=1000(itrebar),27(sudo),100(users),989(docker)
```

En caso de que no lo tuviésemos creado, tendríamos que añadir el usuario a los grupos docker y sudo.

sudo usermod -aG docker itrebar(miUsuario)

sudo usermod -aG sudo itrebar(miUsuario)

- Asignar una IP fija al VPS dentro del rango 192.168.0.41-70/24, según este listado. IP, máscara, gateway, DNS.

Resumen	Agregar	Eliminar	Editar							
Consola	ID	Nombre ↑	Puente	Cortafu...	Etiquet...	Dirección MAC	Dirección IP	Puerta de enlace	MTU	Desconect...
Recursos	net0	eth0	vmbr5	Sí		BC:24:11:E1:...	192.168.0.74/24	192.168.0.1		No
Red										

Editar

Revertir

Nombre del Host	2DAW-itrebar-UbuntuLXC
Dominio DNS	iesalixar.org
Servidor DNS	192.168.0.1



```
itrebar@2DAW-itrebar-UbuntuLXC:/$ cat /etc/systemd/network/eth0.network
[Match]
Name = eth0

[Network]
Description = Interface eth0 autoconfigured by PVE
Address = 192.168.0.74/24
Gateway = 192.168.0.1
DHCP = no
IPv6AcceptRA = false
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.74 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::be24:11ff:fe01:614e prefixlen 64 scopeid 0x20<link>
    ether bc:24:11:e1:61:4e txqueuelen 1000 (Ethernet)
    RX packets 164475 bytes 199655851 (199.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 59230 bytes 31440083 (31.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- Establecer el timezone correcto:
sudo timedatectl set-timezone Europe/Madrid.

```
itrebar@2DAW-itrebar-UbuntuLXC:/$ sudo timedatectl set-timezone Europe/Madrid
itrebar@2DAW-itrebar-UbuntuLXC:/$ timedatectl
    Local time: Mon 2025-03-03 13:44:13 CET
    Universal time: Mon 2025-03-03 12:44:13 UTC
    RTC time: n/a
    Time zone: Europe/Madrid (CET, +0100)
System clock synchronized: yes
    NTP service: inactive
    RTC in local TZ: no
itrebar@2DAW-itrebar-UbuntuLXC:/$
```




2. Dockerfile del contenedor Spring Boot

Realizaremos un despliegue multistage, es decir, utilizaremos varias etapas (stages) dentro de un mismo Dockerfile con el objetivo de generar una imagen final más liviana, optimizada y segura.

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ cat backend/Dockerfile
FROM maven as builder
RUN mkdir -p /build
COPY ./BackEnd-IT-FitLab/BackEnd-IT-FitLab/ ./build/it_fitlab
COPY ./BackEnd-IT-FitLab/BackEnd-IT-FitLab/src/main/resources/.env ./build/it_fitlab
WORKDIR /build/it_fitlab/
RUN mvn clean package -DskipTests

FROM eclipse-temurin:21

WORKDIR /app
COPY --from=builder /build/it_fitlab/target/*.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar", "app.jar"]
```

- Usamos una imagen oficial de Maven como base para compilar el proyecto. Le asignamos un alias build.
- Creamos un directorio /build dentro del contenedor para almacenar los archivos necesarios para la construcción.
- Copiamos el código fuente del backend desde el host al directorio /build/it_fitlab del contenedor.
- Copiamos el .env que debemos tener en la carpeta resources del proyecto al directorio /build/it_fitlab. Importante: Cuando hacemos git clone del proyecto, no traemos el .env, tendremos que configurarlo manualmente.

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ cat backend/BackEnd-IT-FitLab/BackEnd-IT-FitLab/src/main/resources/.env
DB_DRIVER=org.mariadb.jdbc.Driver
DB_URL=jdbc:mariadb://db:3306/IT_FitLab
DB_ROOT_PASSWORD=root_password
DB_DATABASE=IT_FitLab
DB_USER=itrebar
DB_PASSWORD=itrebar
JWT_TOKEN=B397!!S//_3mLa1ñi%3DHI`6Vgae(+lp
JWT_KEYSTORE_PATH=/app/keystore/jwt-keystore.jks
JWT_KEYSTORE_PASSWORD=itrebar
JWT_KEYSTORE_ALIAS=jwt-keypair
CORS_ALLOWED-ORIGINS=https://itfitlab.alixarblue.team
```

- Establecemos el directorio de trabajo a /build/it_fitlab
- Ejecuta Maven para compilar el proyecto y generar el .jar, omitiendo los tests (-DskipTests).
- Usamos una imagen más ligera con JDK de producción (Temurin) para ejecutar el backend
- Establecemos /app como directorio de trabajo en la imagen final.
- Copiamos el .jar generado por Maven desde la imagen builder al contenedor final, renombrándolo como app.jar



- Exponemos el puerto 8080, que es donde correremos la aplicación Spring Boot.

3. Dockerfile del contenedor BD

- MariaDB y phpMyAdmin

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ cat bd/Dockerfile
FROM bitnami/mariadb
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ cat phpMyAdmin/Dockerfile
FROM bitnami/phpmyadmin
```

4. Dockerfile del contenedor NodeJS ([bitnami/node](#))

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ cat node/Dockerfile
FROM bitnami/node
```



5. Docker compose

```
services:
  node:
    tty: true # Enables debugging capabilities when attached to this container.
    command: sh -c 'npm install && npm start'
    container_name: angular
    build:
      context: ./node
      dockerfile: Dockerfile
    ports:
      - 4200:3000
    volumes:
      - ./node/FrontEnd-IT-FitLab:/app
    restart: always
  db:
    container_name: db
    build:
      context: ./bd
      dockerfile: Dockerfile
    volumes:
      - mariadb_data:/bitnami/mariadb
    environment:
      - MARIADB_ROOT_PASSWORD=${DB_ROOT_PASSWORD}
      - MARIADB_USER=${DB_USER}
      - MARIADB_PASSWORD=${DB_PASSWORD}
      - MARIADB_DATABASE=${DB_DATABASE}
    networks:
      - web_network
    restart: always

  phpmyadmin:
    container_name: phpmyadmin
    build:
      context: ./phpMyAdmin
      dockerfile: Dockerfile
    volumes:
      - 'phpmyadmin_data:/bitnami/mariadb'
    environment:
      - DATABASE_HOST=db
      - DATABASE_PORT_NUMBER=3306
      - PHPMYADMIN_ALLOW_REMOTE_CONNECTIONS = yes
    ports:
      - 80:8080
      - 443:8443
    depends_on:
      - db
    networks:
      - web_network
```



```
aplicacionjava:
  container_name: temurin
  build:
    context: ./backend
    dockerfile: Dockerfile
  volumes:
    - ./backend/certs:/app/keystore
    - ./backend/videos:/app/videos
  ports:
    - 8080:8080
    - 8443:8443
  depends_on:
    - db
  networks:
    - web_network
  restart: always
volumes:
  mariadb_data:
    driver: local
  phpmyadmin_data:
    driver: local
networks:
  web_network:
    driver: bridge
```

En la raíz del proyecto tendremos que implementar un archivo .env para almacenar las variables de entorno de la base de datos.

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ cat .env
DB_ROOT_PASSWORD=root_password
DB_DATABASE=IT_FitLab
DB_USER=itrebar
DB_PASSWORD=itrebar
```

Al hacer uso de una clave privada para firmar tokens JWT, necesitaremos el archivo .jks. Para obtenerlo, lo descargaremos desde una máquina virtual que comparta red con la máquina de despliegue. Una vez descargado, haremos una copia mediante scp:

```
imanol@srvweb:~/Descargas$ scp /home/imanol/Descargas/jwt-keystore.jks itrebar@192.168.0.74:/home/itrebar/proyectoFinal/backend/certs
itrebar@192.168.0.74's password:
jwt-keystore.jks                                100% 2818      2.6MB/s   00:00
```

Comprobamos que se haya copiado en la máquina en la que realizaremos el despliegue:

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal/backend/certs$ ls
jwt-keystore.jks
```

En el fichero .env del backend indicamos la ubicación del certificado dentro del contenedor:

```
JWT_KEYSTORE_PATH=/app/keystore/jwt-keystore.jks
```

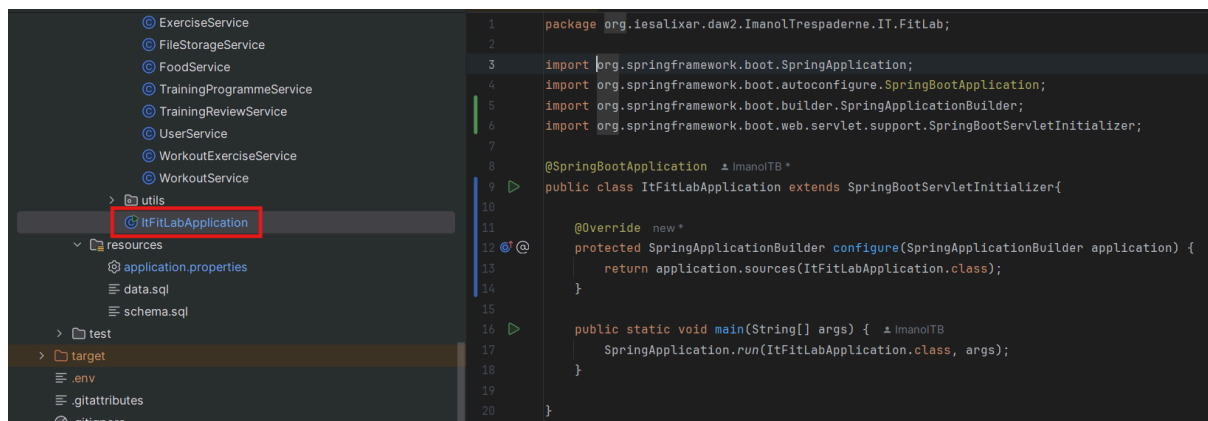
Así mismo, guardaremos videos en el servidor. Esta es la razón por la cual hemos montado un volumen en el contenedor del backend. En el .env del backend tenemos que apuntar a este volumen creado.

```
UPLOAD_PATH= /app/videos
```

De esta manera, cuando creemos un ejercicio con su respectivo video, el video se guardará en el servidor. Para proporcionar espacio suficiente y no consumir recursos de la propia máquina servidora, le montaremos un volumen de un dispositivo de almacenamiento que se conecta a una red local (NAS)

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal/backend/videos$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vgssd-vm--16070--disk--0 25G   5.9G   18G   26% /
none            492K   4.0K  488K    1% /dev
tmpfs           95G    0     95G    0% /dev/shm
tmpfs           38G   452K   38G    1% /run
tmpfs           5.0M    0    5.0M    0% /run/lock
/dev/loop0      15G    28K   14G    1% /home/itrebar/proyectoFinal/
backend/videos
```

Debemos tener en cuenta que el backend no va a utilizar un servidor embebido del IDE y debemos indicarlo en la clase de arranque de la aplicación de la siguiente manera:





Una vez configurados el docker-compose.yml y cada uno de los Dockerfiles de los contenedores correspondientes debemos tener una estructura similar a la siguiente:

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ tree -L 2
.
|-- app
|-- backend
|   |-- BackEnd-IT-FitLab
|   |-- Dockerfile
|   |-- certs
|   |-- it_fitlab
|   `-- videos
|-- bd
|   |-- Dockerfile
|   `-- mariadb_data
|-- docker-compose.yml
|-- node
|   |-- Dockerfile
|   `-- FrontEnd-IT-FitLab
`-- phpMyAdmin
    |-- Dockerfile
    `-- phpmyadmin_data
```

Para desplegar, primero construiremos las imágenes de Docker que hemos definido en el docker-compose.yml, a partir de los Dockerfile asociados a cada servicio. Para ello usaremos el comando:

```
docker compose build
```

Para desplegar la aplicación utilizando los contenedores definidos en el docker-compose.yml lanzaremos el comando:

```
docker compose up -d
```

Usaremos el flag -d para que se lance en segundo plano. De esta manera, crearemos los contenedores y los arrancaremos de manera automática.

Para comprobar si están en pie los contenedores, lanzamos:

docker ps

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
62f48422022a   proyectofinal-phpmyadmin            "/opt/bitnami/script..." 32 minutes ago
683226da03ce   proyectofinal-aplicacionjava         "java -jar app.jar"        6 days ago
8bc28d3757d3   proyectofinal-node                   "sh -c 'npm install ...'"  6 days ago
fb366df552a6   proyectofinal-db                     "/opt/bitnami/script..."  8 days ago
```

Podemos ver los logs de cada uno de los contenedores para asegurarnos que el despliegue ha sido exitoso, y en caso de no serlo, podremos ver donde se origina el error. Para ello:

docker logs + nombre del contenedor

Comprobamos que el contenedor de phpmyadmin está bien desplegado y entramos en el gestor de la base de datos con los datos aportados en la base de datos mariadb. De esta manera, podemos comprobar que las tablas y los datos configurados en el backend(schema.sql y data.sql) se han creado. Para ello, accedemos a una máquina virtual creada en la misma red que en la máquina del servidor de aplicaciones (lxc: 192.168.0.0/24)

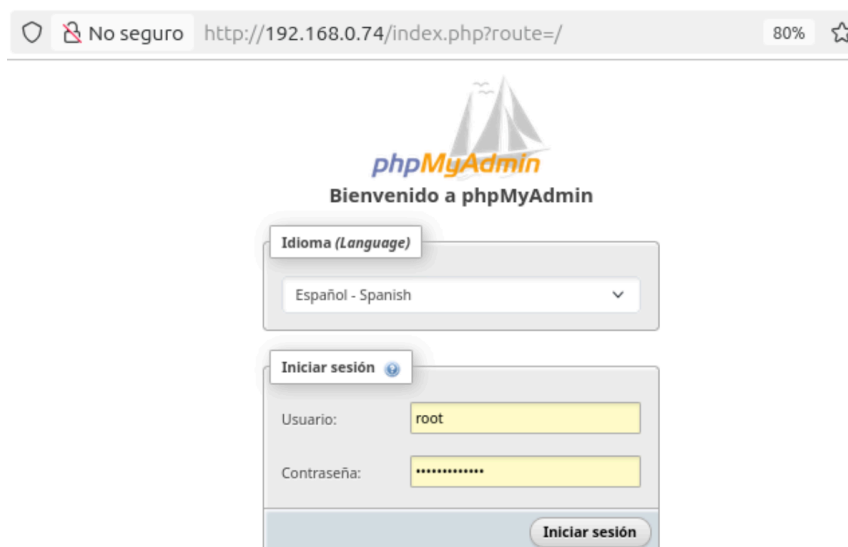




Tabla	Acción	Filas	Tipo	Cod
<input type="checkbox"/> diets	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	11	InnoDB	utf
<input type="checkbox"/> diet_food	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	43	InnoDB	utf
<input type="checkbox"/> exercises	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	90	InnoDB	utf
<input type="checkbox"/> food	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	50	InnoDB	utf
<input type="checkbox"/> roles	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf
<input type="checkbox"/> training_programmes	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	11	InnoDB	utf
<input type="checkbox"/> training_reviews	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	8	InnoDB	utf
<input type="checkbox"/> users	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	13	InnoDB	utf
<input type="checkbox"/> user_roles	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	12	InnoDB	utf
<input type="checkbox"/> workouts	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	8	InnoDB	utf
<input type="checkbox"/> workout_exercises	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	16	InnoDB	utf
11 tablas	Número de filas	264	InnoDB	utf

6. Servidor web proxy inverso

- Generar en Nginx un certificado Let's Encrypt para el dominio mediante challenge DNS.

Para generar un certificado con lets encrypt primero debemos de haber creado un dominio, en nuestro caso gratuito. Para ello, nos registramos en Dynu y creamos un subdominio, itrebar.freedomdns.org (posteriormente, itfitlab.alixarblue.team).

Instalamos el agente Certbot junto con el plugin para Dynu. Hay que crear un fichero /root/dynu-credentials.ini (chmod 600) con el token asociado a tu cuenta (dns_dynu_auth_token =<token>) para que el agente pueda hacer peticiones al API de Dynu. Puedes consultar el API-key dentro dynu en Control panel > API Credentials.

Con el usuario root (su -), instalamos certbot y el plugin certbot-dns-dynu:

```
apt install python3-full
mv /usr/lib/python3.12/EXTERNALLY-MANAGED .
pip install certbot
pip install certbot-dns-dynu
```



```
daw2@Nginx:/etc/dynu/itrebar$ sudo chmod 600 dynu-credential.ini
daw2@Nginx:/etc/dynu/itrebar$ ls -l
total 1
-rw----- 1 root root 54 Feb 18 14:37 dynu-credential.ini
```

```
GNU nano 6.2 dynu-credential.ini
dns_dynu_auth_token= 6TU5Yb67TVU4V4X5cTtf6YT6fVbdW34b
```

Como no están abiertos los puertos TCP 80 ni 443, hay que hacer el challenge DNS indicando el FQDN del virtualhost para que el servicio Let's Encrypt valide tu dominio y emita el certificado correspondiente

```
root@Nginx:/etc/nginx/sites-available# certbot --authenticator dns-dynu --dns-dynu-credentials /etc/dynu/itrebar/dynu-credential.ini -d itrebar.freedomdns.org -d www.itrebar.freedomdns.org certonly
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for itrebar.freedomdns.org and www.itrebar.freedomdns.org

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/itrebar.freedomdns.org-0001/fullchain.pem
Key is saved at: /etc/letsencrypt/live/itrebar.freedomdns.org-0001/privkey.pem
This certificate expires on 2025-05-26.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

.....
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
 * Donating to EFF: https://eff.org/donate-le
.....
root@Nginx:/etc/nginx/sites-available#
```

Localizamos el certificado y visualizamos los detalles del certificado.

```
root@Nginx:/etc/letsencrypt/live/itrebar.freedomdns.org-0001# ls
README  cert.pem  chain.pem  fullchain.pem  privkey.pem
root@Nginx:/etc/letsencrypt/live/itrebar.freedomdns.org-0001#
```

openssl x509 --noout --text -in ruta archivo fullchain.pem

```
root@Nginx:/etc/nginx/sites-available# openssl x509 --noout --text -in /etc/letsencrypt/live/itrebar.freddns.org-0001/fullchain.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      04:3f:16:1b:47:8a:09:6f:26:d1:61:09:66:15:74:ad:60:c7
    Signature Algorithm: ecdsa-with-SHA384
    Issuer: C = US, O = Let's Encrypt, CN = E6
    Validity
      Not Before: Feb 25 12:34:27 2025 GMT
      Not After : May 26 12:34:26 2025 GMT
    Subject: CN = itrebar.freddns.org
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:f6:95:b4:a3:02:66:81:e4:6e:0e:a5:66:bb:1f:
        59:41:d7:31:6f:6f:ae:d3:15:88:8c:50:37:dd:00:
        77:60:ea:94:b5:41:e3:37:2f:39:1f:8d:22:a6:0e:
        77:4a:9b:5b:ff:d5:b8:22:9f:d2:cb:9d:94:79:93:
        84:f2:b7:b6:be
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Key Usage: critical
        Digital Signature
      X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
      X509v3 Basic Constraints: critical
        CA:FALSE
      X509v3 Subject Key Identifier:
        81:FE:F2:0F:DF:E8:7F:28:45:24:BA:39:22:21:32:B8:9E:5C:A1:4A
      X509v3 Authority Key Identifier:
        93:27:46:99:03:A9:51:68:8E:06:D6:C4:42:48:DB:23:BF:58:94:02
      Authority Information Access:
        OCSP - URI:http://e6.o.letsencrypt.org
        CA Issuers - URI:http://e6.i.letsencrypt.org/
      X509v3 Subject Alternative Name:
        DNS:itrebar.freddns.org, DNS:www.itrebar.freddns.org
      X509v3 Certificate Policies:
```

- Reenvío de peticiones desde Nginx a NodeJS.

Añade la IP correcta 192.168.0.X/24 de NodeJS asociada al nombre en el fichero /etc/hosts de Nginx

```
daw2@Nginx:/ $ grep -Ei 'itre|itfit' /etc/hosts
192.168.0.74      itrebar.freddns.org
192.168.0.74      www.itrebar.freddns.org
192.168.0.74      itfitlab.alixarblue.team
192.168.0.74      www.itfitlab.alixarblue.team
daw2@Nginx:/ $
```



- Mod de seguridad

En nuestro caso haremos uso de un WAF (Web Application Firewall) que debemos de instalar y configurar manualmente. De esta manera crearemos una capa de seguridad adicional analizando las peticiones HTTP y sus cabeceras y aplicando un conjunto de reglas (OWASP) para identificar patrones maliciosos.

* En caso de que estén saltando reglas que consideremos no maliciosas deberemos deshabilitarlas en:

/etc/nginx/modsec/custom_put_allow.conf;

```
daw2@Nginx:/$ cat /etc/nginx/modsec/custom_put_allow.conf

# Las peticiones estan siendo detectadas como maliciosas por nginx
# por lo que es posible que se tenga que deshabilitar esta regla temporalmente en algunas ocasiones
# si se produce el error inesperado y luego volver a comentarla para no dejar vulnerable el sistema

#SecRuleRemoveById 911100

SecRule REQUEST_METHOD "!^(?:GET|HEAD|POST|OPTIONS|DELETE|PUT|PROPFIND|PATCH)$" \
    "id:1000001,phase:1,deny,status:403,msg:'Method not allowed',severity:2"

SecRule REQUEST_METHOD "@streq PUT" \
    "id:1000002,phase:1,pass,nolog,ctl:ruleRemoveById=911100"

SecRule REQUEST_METHOD "@streq PUT" \
    "id:1000003,phase:1,pass,nolog,ctl:ruleRemoveById=920420"

SecRule REQUEST_METHOD "@streq PROPFIND" \
    "id:1000004,phase:1,pass,nolog,ctl:ruleRemoveById=911100"

SecRule REQUEST_HEADERS:Host "@rx ^[\d.:]+$" \
    "id:1000005,phase:1,pass,nolog,ctl:ruleRemoveById=920350"

SecRule REQUEST_METHOD "@streq DELETE" "chain,phase:1,id:1001006,pass,nolog"
    SecRule REQUEST_URI "@rx ^/[^/]+/api/"
    "ctl:ruleRemoveById=911100"
```

Así mismo añadiremos las siguientes líneas en el virtual host.

```
modsecurity on;
modsecurity_rules_file /etc/nginx/modsec/custom_put_allow.conf;
modsecurity_rules_file /etc/nginx/modsec/main.conf;
```



También deberemos definir los dominios para los que el bloque server será responsable y definir las rutas donde se guardarán los logs de acceso.

```
server {  
  
    # SSL configuration  
    #  
    # listen 443 ssl default_server;  
    # listen [::]:443 ssl default_server;  
    #  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    #  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782  
    #  
    # Self signed certs generated by the ssl-cert package  
    # Don't use them in a production server!  
    #  
    # include snippets/snakeoil.conf;  
  
    #root /var/www/html;  
  
    modsecurity on;  
    modsecurity_rules_file /etc/nginx/modsec/custom_put_allow.conf;  
    modsecurity_rules_file /etc/nginx/modsec/main.conf;  
  
    # Add index.php to the list if you are using PHP  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name itfitlab.alixarblue.team www.itfitlab.alixarblue.team;  
    access_log /var/log/nginx/itfitlab.alixarblue.team;
```



- Modifica la directiva proxy_pass http://subdominio:4200

Para que funcione el reenvío de la petición, será preciso además añadir, dentro de location, las cabeceras de la petición, los timeouts y los buffers:

```
location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    #try_files $uri $uri/ =404;

    proxy_pass http://itfitlab.alixarblue.team:4200;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_cache_bypass $http_upgrade;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

    # Permitir encabezados necesarios
    proxy_set_header Authorization $http_authorization;
    proxy_set_header Cookie $http_cookie;

    # Aumentar los tiempos de espera y el tamaño del buffer
    proxy_connect_timeout 300s;
    proxy_send_timeout 300s;
    proxy_read_timeout 300s;
    send_timeout 300s;

    proxy_buffer_size 128k;
    proxy_buffers 4 256k;
    proxy_busy_buffers_size 256k;

    # Permitir CORS (Cross-Origin Resource Sharing)
    add_header Access-Control-Allow-Origin *;
    add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";
    add_header Access-Control-Allow-Headers "Authorization, Origin, X-Requested-With, Content-Type, Accept";

    # Manejar m todos OPTIONS (para CORS)
    if ($request_method = OPTIONS) {
        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Methods "GET, POST, OPTIONS";
        add_header Access-Control-Allow-Headers "Authorization, Origin, X-Requested-With, Content-Type, Accept";
        add_header Content-Length 0;
        add_header Content-Type text/plain;
        return 204;
    }
}
```

```
location /api/ {

    proxy_pass      http://itfitlab.alixarblue.team:8080;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_cache_bypass $http_upgrade;

    proxy_set_header    Host                $host;
    proxy_set_header    X-Real-IP           $remote_addr;
    proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host    $host;
    proxy_set_header    X-Forwarded-Server $host;
    proxy_set_header    X-Forwarded-Port    $server_port;
    proxy_set_header    X-Forwarded-Proto  $scheme;

    # Cabeceras necesarias
    proxy_set_header Authorization $http_authorization;
    proxy_set_header Cookie $http_cookie;

    # Aumentar los tiempos de espera y el tamaño del buffer
    proxy_connect_timeout 300s;
    proxy_send_timeout 300s;
    proxy_read_timeout 300s;
    send_timeout 300s;

    proxy_buffer_size 128k;
    proxy_buffers 4 256k;
    proxy_busy_buffers_size 256k;

    proxy_request_buffering off;
    client_max_body_size 50M;

    # Permitir CORS (Cross-Origin Resource Sharing)
    add_header Access-Control-Allow-Origin *;
    add_header Access-Control-Allow-Methods "GET, POST, PUT, OPTIONS, DELETE, PATCH";
    add_header Access-Control-Allow-Headers "Authorization, Origin, X-Requested-With, Content-Type, Accept";

    # Manejar todos los OPTIONS (para CORS)
    if ($request_method = OPTIONS) {
        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Methods "GET, POST, PUT, OPTIONS, DELETE, PATCH";
        add_header Access-Control-Allow-Headers "Authorization, Origin, X-Requested-With, Content-Type, Accept";
        add_header Content-Length 0;
        add_header Content-Type text/plain;
        return 204;
    }
}
```

Cerramos el bloque `server {}` configurando la conexión https en nginx haciendo uso del certificado SSL de Lets Encrypt, gestionado por certbot.

```
listen [::]:443 ssl; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/alixarblue.team-0002/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/alixarblue.team-0002/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
```



Por último, fuera del bloque server principal configuraremos otro bloque server {} para las peticiones que llegan al puerto 80 (http), sean redirigidas al 443 (https).

```
server {  
    if ($host = www.itfitlab.alixarblue.team) {  
        return 301 https://$host$request_uri;  
    } # managed by Certbot  
  
    if ($host = itfitlab.alixarblue.team) {  
        return 301 https://$host$request_uri;  
    } # managed by Certbot  
  
    listen 80;  
    listen [::]:80;  
  
    server_name itfitlab.alixarblue.team www.itfitlab.alixarblue.team;  
    return 404; # managed by Certbot  
}
```

- Chequea la configuración con nginx -t y recárgala con sudo systemctl reload nginx

```
daw2@Nginx:/etc/nginx/sites-available$ sudo nginx -t
```

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful  
daw2@Nginx:/etc/nginx/sites-available$ sudo systemctl reload nginx.service
```

7. Despliegue de app Angular

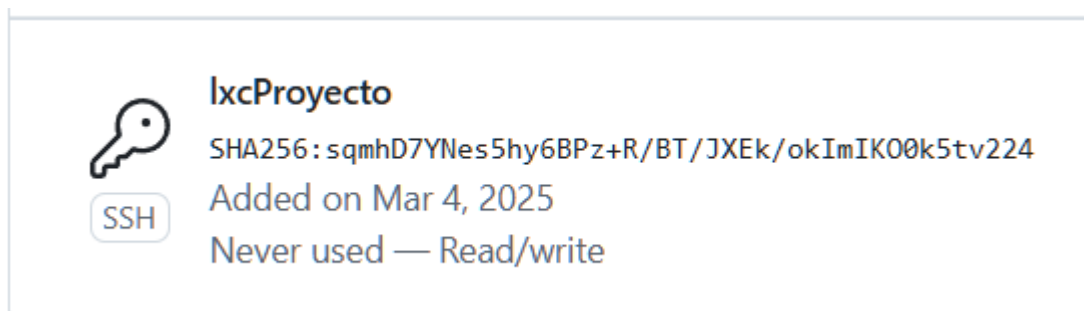
- Configuración de git y repositorio remoto por SSH

Creamos el par de claves ssh mediante el comando:

```
itrebar@2DAW-itrebar-UbuntuLXC:~/.ssh$ ssh-keygen -t rsa -b 4096 -C "imanoltrespadernebaron@gmail.com"
```

Posteriormente pegamos en github la parte pública de la clave, id_rsa.pub.

```
itrebar@2DAW-itrebar-UbuntuLXC:~/.ssh$ ls
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```



- Git clone del repositorio de GitHub.

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal/node$ git clone git@github.com:ImanolTB/FrontEnd-IT-FitLab.git
Cloning into 'FrontEnd-IT-FitLab'...
Enter passphrase for key '/home/itrebar/.ssh/id_rsa':
remote: Enumerating objects: 818, done.
remote: Counting objects: 100% (818/818), done.
remote: Compressing objects: 100% (490/490), done.
remote: Total 818 (delta 536), reused 567 (delta 310), pack-reused 0 (from 0)
Receiving objects: 100% (818/818), 6.71 MiB | 7.75 MiB/s, done.
Resolving deltas: 100% (536/536), done.
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal/node$
```


- Editar angular.json para que la aplicación escuche en la IP del contenedor y en el puerto 3000.

```
"serve": {  
  "builder": "@angular-devkit/build-angular:dev-server",  
  "options": {  
    "browserTarget": "project:build",  
    "host": "0.0.0.0",  
    "port": 3000,  
    "disableHostCheck": true  
  },  
},
```

```
"serve": {  
  "builder": "@angular-devkit/build-angular:dev-server",  
  "options": {  
    "buildTarget": "club-nautico-angular:build",  
    "host": "0.0.0.0",  
    "port": 3000,  
    "disableHostCheck": true  
  },  
},
```

Añadimos también lo siguiente para que no se pare npm start:

```
"cli": {  
  "analytics": false  
}  
}
```

```
}  
"cli": {  
  "analytics": false  
}  
}  
}
```



- Comprobamos en la salida del comando docker logs que el servidor express está escuchando en el puerto 3000.

Network: http://IP_contenedor:3000

```
itrebar@2DAW-itrebar-UbuntuLXC:~/proyectoFinal$ docker logs angular
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory.
more comprehensive and powerful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 1077 packages, and audited 1078 packages in 35s

182 packages are looking for funding
  run `npm fund` for details

1 low severity vulnerability

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 11.3.0 -> 11.4.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.1
npm notice To update run: npm install -g npm@11.4.1
npm notice

> front-end-it-fit-lab@0.0.0 start
> ng serve

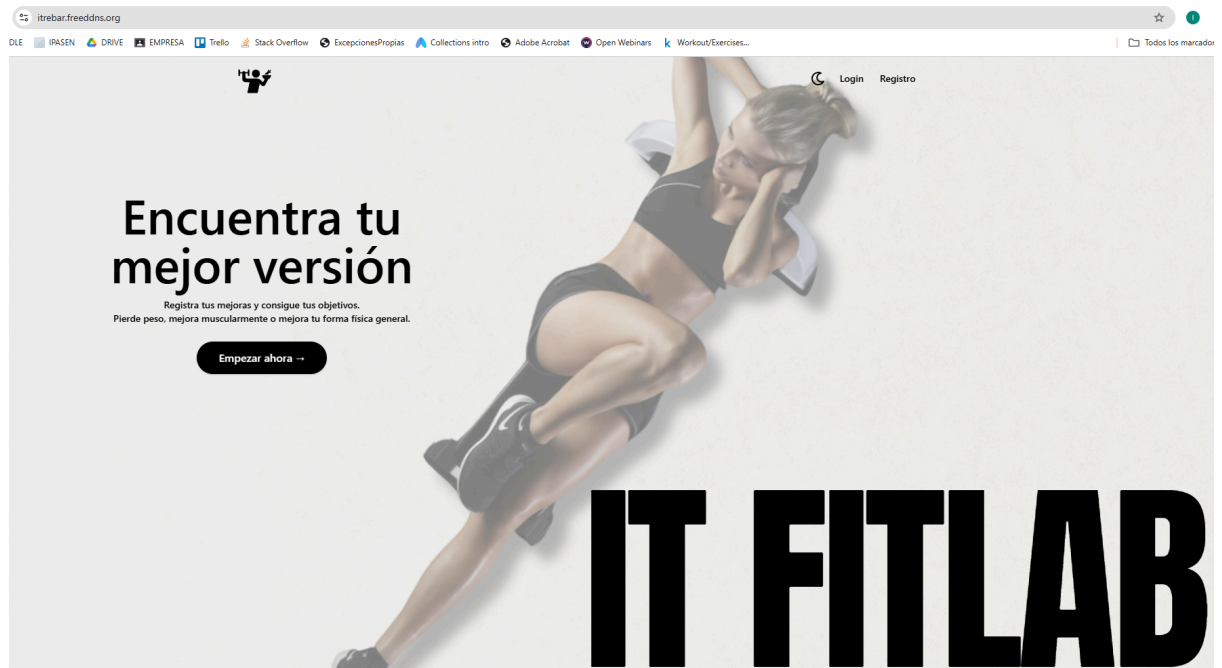
Warning: Running a server with --disable-host-check is a security risk. See https://
The "disableHostCheck" option will not be used because it is not supported by the "@
Component HMR has been enabled.
If you encounter application reload issues, you can manually reload the page to bypa
Please consider reporting any issues you encounter here: https://github.com/angular/

Browser bundles
Initial chunk files | Names | Raw size
main.js | main | 719.92 kB |
styles.css | styles | 137.34 kB |
polyfills.js | polyfills | 90.20 kB |
| Initial total | 947.45 kB

Server bundles
Initial chunk files | Names | Raw size
main.server.mjs | main.server | 721.11 kB |
polyfills.server.mjs | polyfills.server | 570.97 kB |
server.mjs | server | 1.86 kB |

Application bundle generation complete. [13.738 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:3000/
→ Network: http://172.19.0.2:3000/
```



Conclusiones del proyecto.

El desarrollo de IT FitLab ha sido mucho más que la construcción de una aplicación funcional; ha representado el cierre de dos años intensos de aprendizaje, esfuerzo y evolución personal. Durante este tiempo, he pasado de tener dificultades para asimilar los conceptos más básicos a sentirme realmente capaz de hacer cosas que ni siquiera habría imaginado al comienzo del grado, cómo abordar el diseño y desarrollo completo de un proyecto real, desde la planificación hasta su despliegue.

El proyecto ha permitido consolidar habilidades clave como el diseño de bases de datos relacionales, la implementación de servicios REST seguros, el uso de frameworks modernos como Angular y Spring Boot, así como la utilización de herramientas profesionales como Docker, Postman y Figma. Pero más allá del aspecto técnico, este proyecto me ha enseñado a organizar mi tiempo, a afrontar problemas complejos con perseverancia, y sobre todo, a confiar en mi capacidad para encontrar soluciones.

Además de cumplir con el objetivo principal de unir tecnología y deporte en una sola aplicación, el desarrollo de este proyecto deja abiertas nuevas posibles funcionalidades. En el futuro, sería posible incorporar funcionalidades como la contratación de entrenadores personales o nutricionistas directamente desde la plataforma, así como la integración con dispositivos de monitoreo físico. Estas mejoras no solo enriquecerían la experiencia del usuario, sino que también abrirían la puerta a una posible monetización de la aplicación, dándole un enfoque más profesional y con potencial real en el mercado.

Por otro lado, también sería muy útil permitir la clasificación de entrenamientos según el tipo de deporte, lo que facilitaría su adopción en clubes deportivos y centros especializados. Esto permitiría dar seguimiento a la evolución de atletas, nadadores, futbolistas y otros perfiles, ampliando así el alcance de la aplicación y la posibilidad de monetización.

IT FitLab representa el punto de encuentro entre dos pasiones personales: el desarrollo de aplicaciones y el mundo del deporte. Y más allá de su utilidad como proyecto académico, simboliza el cierre de una etapa formativa y el inicio de una nueva etapa profesional en la que seguiré con muchas ganas de seguir aprendiendo y construyendo nuevos proyectos.