*PROYECTO***:**

**GymSync: Gestión de rutinas para gimnasio**



*Realizado por:* Zecheng Zhou

|  |  |
| --- | --- |
| http://alba9.es/imagenes/logo_galileo_1.jpg | **Desarrollo de Aplicaciones Web (DAW)** |
| **Valladolid, a 30 de mayo de 2024** |
| Nota: |  |

[1. Introducción 2](#_Toc466166712)

[1.1 Objetivo de la Aplicación 3](#_Toc1899133213)

[2. Análisis 4](#_Toc1221513846)

[2.1 Requisitos de Hardware 5](#_Toc1131948736)

[2.2 Requisitios de Software 5](#_Toc838856322)

[2.3 Arquitectura propuesta 5](#_Toc2046626767)

[2.4 Usuarios de la Aplicación 5](#_Toc56887681)

[2.5 Requisitos Funcionales y No Funcionales 6](#_Toc2009807020)

[2.5.1 Requisitos Funcionales 6](#_Toc68096721)

[2.5.2 Requisitos No Funcionales 7](#_Toc775444452)

[2.6 Diagrama de Casos de uso 7](#_Toc556733930)

[2.6.1 Diagrama de Usuario 8](#_Toc1542339049)

[2.6.2 Diagrama de Entrenador 8](#_Toc495081063)

[2.6.3 Diagrama de Administrador 9](#_Toc1484432220)

[3. Diseño del sistema 10](#_Toc488451012)

[3.1 Base de Datos 11](#_Toc1790588996)

[3.1.1 Modelo Relacional 11](#_Toc1619102134)

[3.1.2 Modelo Entidad-Relación 13](#_Toc1040450400)

[3.3 Diagrama de flujo 14](#_Toc481562328)

[3.3.1 Login 15](#_Toc763320091)

[3.3.2 Dashboard Cliente 15](#_Toc1901864258)

[3.3.3 Dashborad Entrenador 17](#_Toc1025118826)

[3.3.4 Dashboard Admin 19](#_Toc1502879256)

[1. Gestión de Entrenadores 20](#_Toc1882915879)

[2. Gestión de Usuarios 20](#_Toc1235642553)

[4.Desarrollo e implementación 21](#_Toc1179190128)

[4.1 Tecnologías utilizadas 22](#_Toc1984016690)

[4.1.1 Frontend 22](#_Toc269383154)

[4.1.2 Backend 22](#_Toc893338471)

[4.2 Organizacion del proyecto 24](#_Toc1671418607)

[4.2.1 Frontend (Angular) 25](#_Toc492938667)

[4.2.2 Backend (Spring Boot) 26](#_Toc1176910663)

[4.3 Difucultades Técnicas durante el desarrollo 30](#_Toc1098500392)

[4.4 Funcionalidades principales 31](#_Toc1443567239)

[5. Seguridad 33](#_Toc832225583)

[6. Pruebas y Validaciones 35](#_Toc353005004)

[6.1 Pruebas Unitarias 36](#_Toc2106560744)

[6.2 Pruebas de Integración 36](#_Toc982386522)

[7. Despliegue 37](#_Toc1198188760)

[8. Plan de Negocio 45](#_Toc1821678998)

[9. Conclusiones 48](#_Toc292986288)

[10.Bibliografía 50](#_Toc1612071074)

# 1. Introducción

## 1.1 Objetivo de la Aplicación

GymSync es una plataforma diseñada para revolucionar la gestión de entrenamientos en gimnasios, ofreciendo una solución integral que conecta a entrenadores y usuarios en un entorno digital eficiente. Su objetivo principal es:

1. Automatizar y personalizar la creación de rutinas de entrenamiento.
2. Facilitar el seguimiento del progreso de los clientes (peso, repeticiones, rendimiento).
3. Optimizar la comunicación entre entrenadores y usuarios mediante herramientas integradas.
4. Mejorar la retención de clientes con planes adaptados y motivación constante.

Con una interfaz intuitiva y funcionalidades avanzadas, GymSync busca eliminar el uso de planillas físicas o sistemas desconectados, centralizando toda la gestión en una única plataforma accesible desde cualquier dispositivo.

**1.2 Descripción**

GymSync es una aplicación web/móvil que opera bajo un sistema de perfiles diferenciados:

* Administradores (dueños de gimnasios): Gestionan membresías, entrenadores y métricas globales.
* Entrenadores: Crean planes personalizados, asignan rutinas y monitorizan el avance de sus clientes.
* Usuarios: Acceden a sus rutinas, registran entrenamientos y reciben feedback en tiempo real.

Funcionalidades Clave:

* Calendario de Entrenamientos: Visualización semanal/mensual de rutinas asignadas.
* Progreso Automatizado: Gráficos de evolución (fuerza, medidas corporales, etc.).
* Sistema de Notificaciones: Recordatorios de entrenamientos y logros alcanzados.
* Chat Integrado: Comunicación directa entre cliente-entrenador.
* Biblioteca de Ejercicios: Vídeos y descripciones de técnicas correctas.

**1.3 Motivación Personal**

El desarrollo de GymSync surge de la necesidad de modernizar la gestión en gimnasios, donde aún predominan métodos tradicionales (libretas, Excel, WhatsApp). Como apasionado del fitness y la tecnología, busco:

* Resolver problemas reales: Falta de personalización y seguimiento ineficiente.
* Impulsar la eficiencia: Reducir tiempo perdido en coordinación manual.
* Escalabilidad: Crear una herramienta adaptable a gimnasios pequeños y cadenas.

La visión a largo plazo es posicionar a GymSync como el estándar en gestión deportiva, integrando inteligencia artificial para ajustar rutinas automáticamente según el progreso del usuario.

# 2. Análisis

## 2.1 Requisitos de Hardware

* Procesador de 2GHz o superior.
* 4 GB de RAM

## 2.2 Requisitios de Software

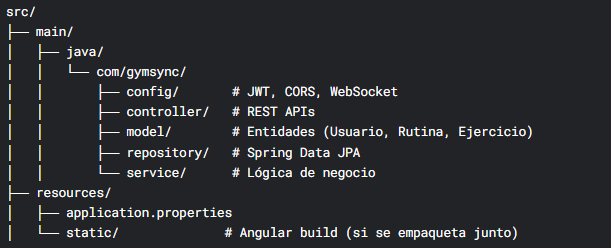
* Sistema Operativo: Win32 x64 / macOS 10.13 o superior / Linux
* Navegador: Chrome, Firefox o Edge (última versión)
* Node.js 22.13.1
* NPM 10.9.2
* Visual Studio (extensiones: Spring boot, Maven for java)
* Java 23.0.2
* Apache Maven 3.9.9
* Angular 19.2.9
* Tomcat 11.0
* Base de datos: MySQL

## 2.3 Arquitectura propuesta

Frontend (Angular 19.2.9 , puerto 4200):

* Componentes clave:
* Módulo de autenticación (JWT)
* Dashboard interactivo con gráficos de progreso
* Librerías útiles:
* SweetAlert2
* Angular Material para UI consistente (botones, cards, forms)
* RxJS para manejo de observables (API calls)

Backend (Spring Boot + Maven, puerto 8080):



## 2.4 Usuarios de la Aplicación

GymSync maneja tres roles principales con permisos específicos:

1. Clientes (Usuarios Normales):

* Visualizar sus rutinas asignadas por el entrenador.
* Registrar el progreso diario (peso, repeticiones, tiempo de entrenamiento).
* Acceder a estadisticas de evolución física y rendimiento.

1. Entrenadores:

* Crear y personalizar rutinas para cada cliente.
* Monitorear el progreso de sus clientes asignados.
* Enviar mensajes directos y ajustar planes de entrenamiento.
* Ver su lista de clientes activos.

1. Administradores (Dueños/Gestión del Gimnasio):

* Gestionar altas/bajas de clientes y entrenadores.
* Configurar membresías y planes de pago.
* Asignar clientes a entrenadores específicos.

## 2.5 Requisitos Funcionales y No Funcionales

### 2.5.1 Requisitos Funcionales

1. Gestión de Usuarios (Admin)

* Añadir/editar/eliminar clientes y entrenadores.
* Asignar roles (cliente, entrenador).

1. Creación de Rutinas (Entrenadores)

* Diseñar planes con ejercicios, series y repeticiones personalizadas.
* Asignar rutinas a clientes específicos.

1. Seguimiento de Progreso (Clientes y Entrenadores)

* Registrar pesos, medidas y marcas personales.
* Visualizar evolución.

1. Autenticación Segura

* Login y contraseña (JWT).
* Responsividad

### 2.5.2 Requisitos No Funcionales

1. Seguridad

Contraseñas almacenadas con bcrypt

1. Rendimiento

Tiempo de carga < 2 segundos en 90% de las peticiones.

1. Escalabilidad

Arquitectura modular para añadir futuros módulos (ej: nutrición).

1. Compatibilidad

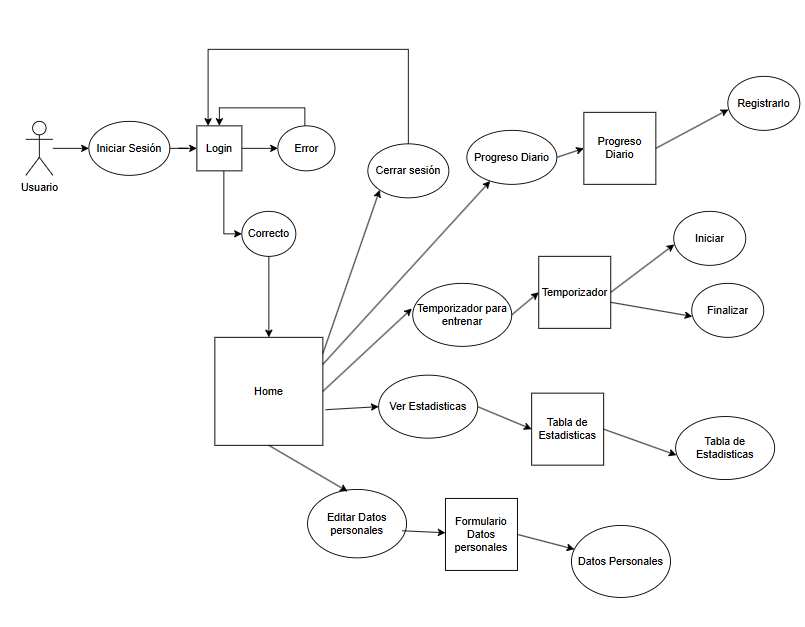
Soporte para Chrome, Firefox, Safari y Edge (últimas 2 versiones).

1. Mantenibilidad

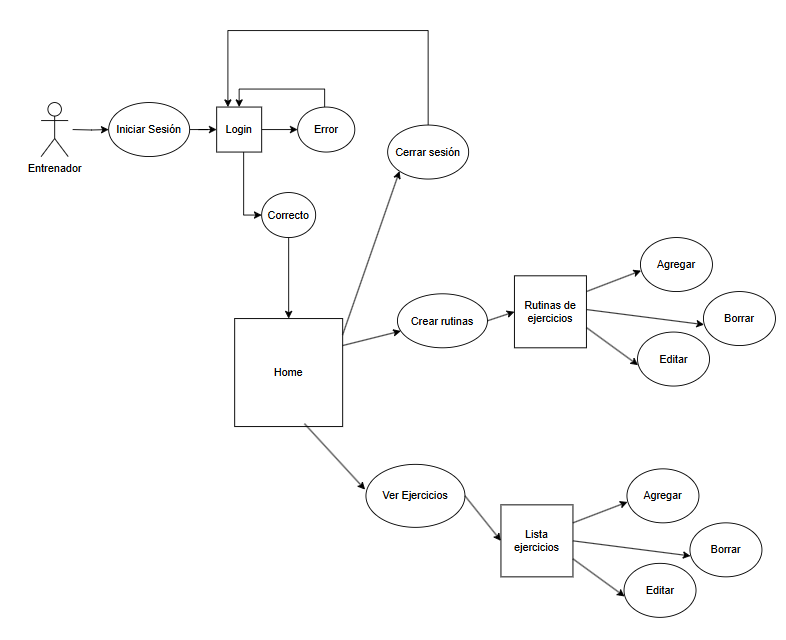
Código documentado con Swagger (APIs) y comentarios en Angular.

## 2.6 Diagrama de Casos de uso

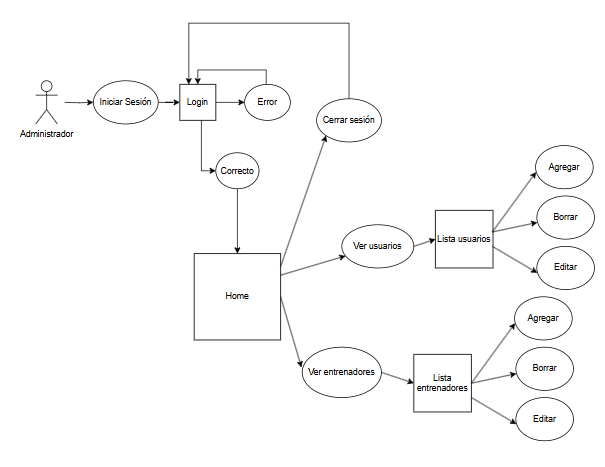
### 2.6.1 Diagrama de Usuario



### 2.6.2 Diagrama de Entrenador

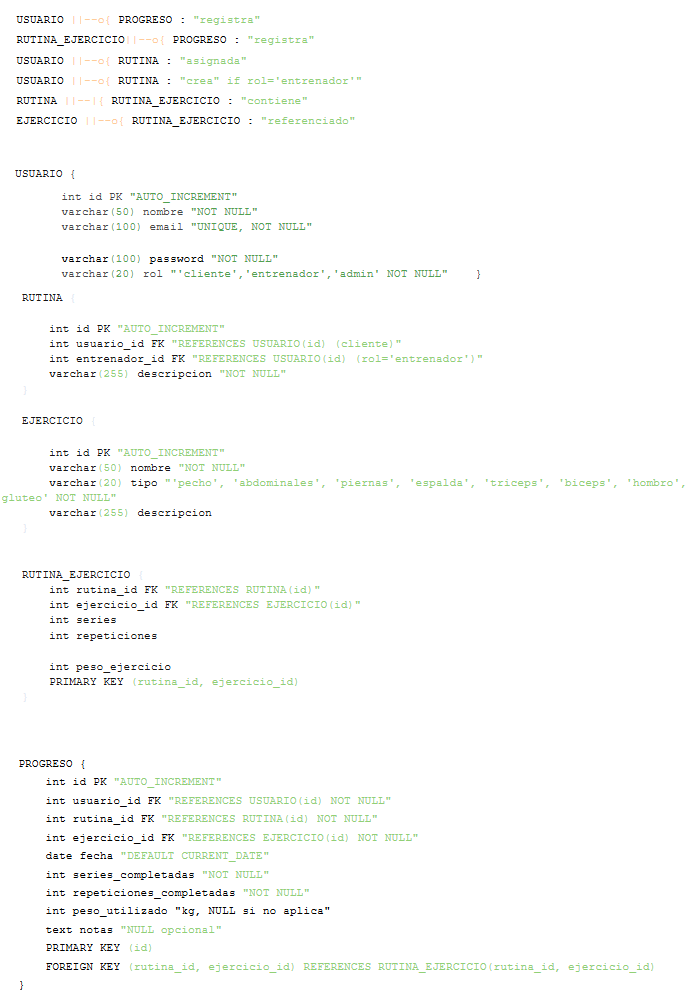


### 2.6.3 Diagrama de Administrador



# 3. Diseño del sistema

## 3.1 Base de Datos



### 3.1.1 Modelo Relacional

USUARIO(id, nombre, email, rol)

* rol ∈ {'cliente', 'entrenador', 'admin'}

RUTINA(id, usuario\_id, entrenador\_id, descripcion)

* usuario\_id → USUARIO.id (cliente, relación 1:1)
* entrenador\_id → USUARIO.id (rol='entrenador', relación N:1)
* NOTA: Se eliminó la restricción UNIQUE en usuario\_id según la nueva estructura

EJERCICIO

(id, nombre, tipo, descripcion)

* tipo ∈ {'pecho', 'abdominales', 'piernas', 'espalda', 'triceps', 'biceps', 'hombro', 'gluteo'}

RUTINA\_EJERCICIO (rutina\_id, ejercicio\_id, series, repeticiones, peso\_ejercicio)

* PK(rutina\_id, ejercicio\_id)

PROGRESO(id, usuario\_id, rutina\_id, ejercicio\_id, fecha, series\_completadas, repeticiones\_completadas, peso\_utilizado, notas)

* FK(usuario\_id, rutina\_id, ejercicio\_id)

Relaciones implítas

* RUTINA.usuario\_id → USUARIO.id (relación 1:N, cliente)
* RUTINA.entrenador\_id → USUARIO.id (relación N:1, entrenador)
* RUTINA\_EJERCICIO.rutina\_id → RUTINA.id
* RUTINA\_EJERCICIO.ejercicio\_id → EJERCICIO.id
* PROGRESO.usuario\_id → USUARIO.id
* PROGRESO.rutina\_id → RUTINA.id
* PROGRESO.ejercicio\_id → EJERCICIO.id
* PROGRESO(rutina\_id, ejercicio\_id) → RUTINA\_EJERCICIO(rutina\_id, ejercicio\_id)

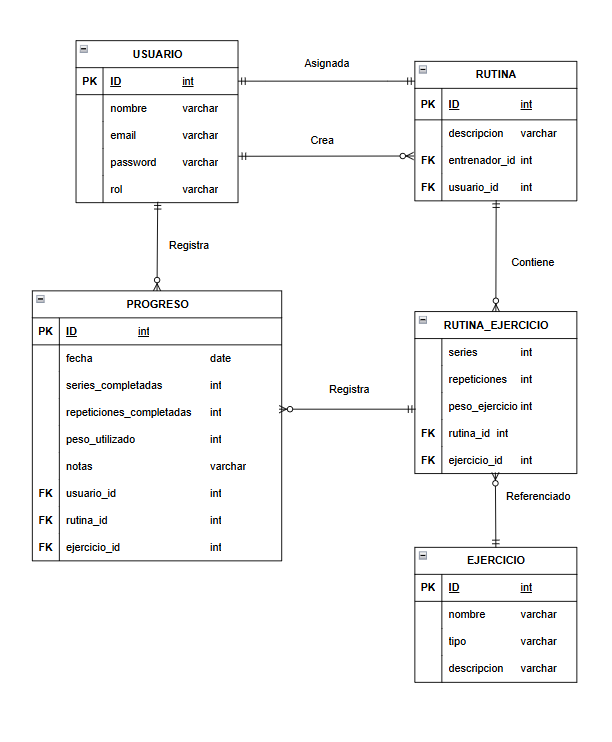
Ejemplo semántico

1. USUARIO(1, 'Ana', '[ana@gmail.com](mailto:ana@gmail.com)',

'$2a$12$AB9qGFUgcShf6u16K/bUbuiTkEwNxmGw.gP1jVe9wcPF2zF.lIwmi', 'cliente')

1. RUTINA(1, 1, 2, 'Rutina de fuerza inicial') → Ana tiene una rutina creada por Pedro
2. EJERCICIO(1, 'Sentadillas', 'piernas', 'Ejercicio para fortalecer piernas')
3. RUTINA\_EJERCICIO(1, 1, 4, 12, 20) → La rutina 1 incluye sentadillas, 4 series de 12 con 20kg recomendados
4. PROGRESO(1, 1, 1, 1, '2025-05-30', 4, 12, 20.5, 'Buena ejecución, última serie con peso extra')

### 3.1.2 Modelo Entidad-Relación



## 3.3 Diagrama de flujo

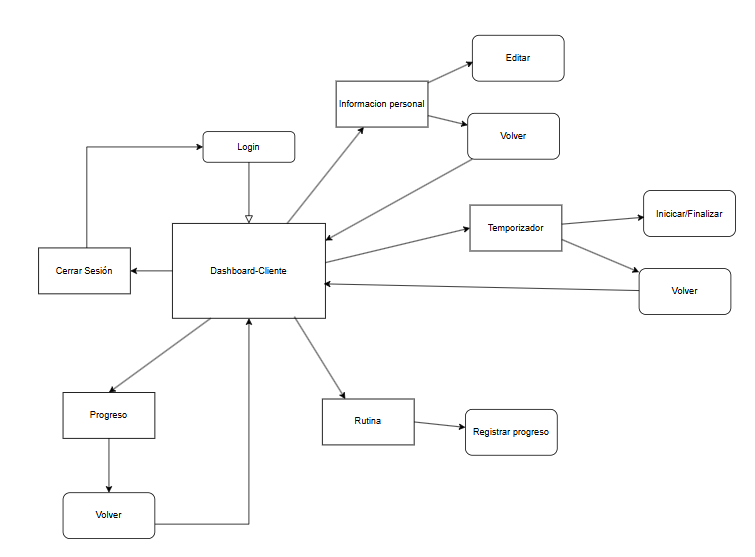
### 3.3.1 Login

Este diagrama representa el proceso de inicio de sesión en mi app. El flujo comienza con inicio del proceso y contiúa con la etapa de login, donde se ingresan las credenciales:

1. Si son incorrectas, el proceso finaliza
2. Si son correctas, procede:

* Si es cliente, accede a su dashboard, donde puede ver sus rutinas y progresos.
* Si es administrador, accede a su dashboard, donde puede eliminar y registrar usuarios.
* Si es entrenador, accede a su dashboard, donde puede crear rutinas y ver los usuarios asignados

### 3.3.2 Dashboard Cliente



Este diagrama representa la navegación principal del cliente dentro de la aplicación, comenzando desde su panel de control (Dashboard) y desplegando las siguientes funcionalidades clave:

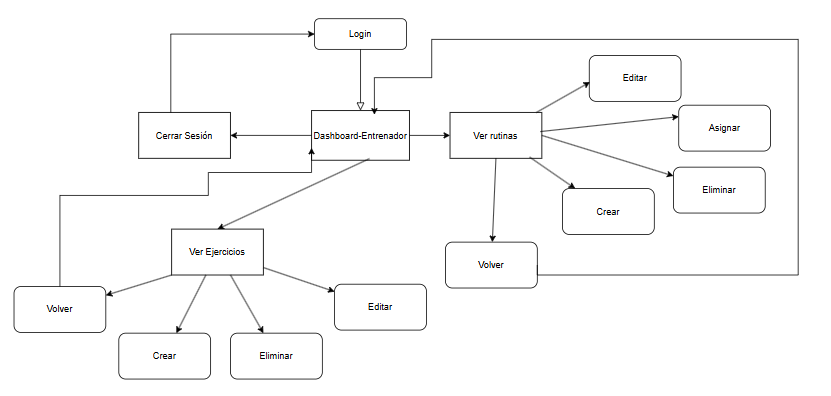
1. **Gestión de Perfil**
   1. Acceder a información personal.
   2. Editar datos (nombre, email).
   3. Volver al dashboard principal.
2. **Temporizador de Entrenamiento**
   1. Iniciar sesión de entrenamiento.
   2. Volver al menú principal.
3. **Rutina Asignada**
   1. Visualizar ejercicios del día/semana.
   2. Registrar progreso (repeticiones, notas).
   3. Botón Ver Progreso que muestra gráficos históricos
   4. Opción de Volver.

Ejemplo de interacción:

**Dashboard → [Rutina Asignada] → [Registrar Progreso]**

**→ Ingresa datos → [Guardar] → *Vuelve al Dashboard***

### 3.3.3 Dashborad Entrenador

Este diagrama describe la navegación principal del entrenador dentro de la aplicación, destacando las siguientes funcionalidades clave:

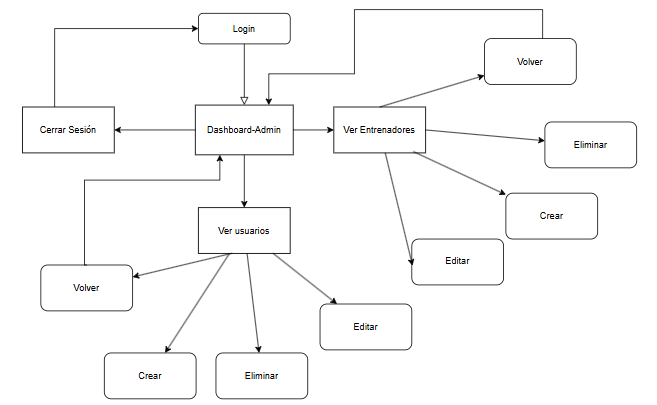
1. **Gestión de Rutinas**
   1. **Ver listado completo** de rutinas asignadas a clientes.
   2. **Crear nuevas rutinas** (selección de ejercicios, series, repeticiones).
   3. **Editar/eliminar** rutinas existentes.
   4. Opción de *Volver* al dashboard principal.
2. **Gestión de Ejercicios**
   1. **Explorar biblioteca** de ejercicios (pecho, espalda, hombro, pierna, biceps, triceps, abdominales, gluteos).
   2. **Agregar nuevos ejercicios** (nombre, descripción, multimedia).
   3. **Editar/eliminar** ejercicios existentes.
   4. *Volver* al menú principal.
3. **Seguimiento de Clientes**
   1. **Ver lista de clientes asignados**.
   2. **Acceder a progresos individuales** (gráficos de peso, medidas, rendimiento).
   3. **Modificar rutinas** basado en el desempeño.
   4. *Volver* al dashboard principal.

Ejemplo de interacción:

**Dashboard → [Clientes Asignados] → [Ver Progreso de Juan]**

**→ [Ajustar Rutina] → [Guardar Cambios] → *Vuelve al Dashboard***

### 3.3.4 Dashboard Admin



Este diagrama describe las funcionalidades clave del panel de administración, diseñado para la gestión integral del gimnasio:

#### **1. Gestión de Entrenadores**

* **Visualización completa**
  + Listado de todos los entrenadores registrados
* **Acciones administrativas**
  + **Creación de nuevos entrenadores**  
    (registro completo con email y credenciales)
  + **Edición de perfiles**  
    (modificar datos personales: nombre e email)
  + **Eliminación controlada**  
    (con reasignación opcional de clientes)
  + *Botón de Volver* al panel principal

#### **2. Gestión de Usuarios**

* **Herramientas de administración**
  + **Registro manual de nuevos usuarios**
  + **Edición completa de perfiles**  
    (incluyendo reset de contraseñas)
  + *Retorno* al panel

Ejemplo de interacción:

**Dashboard Admin → [Entrenadores] → [Acción] → [Confirmación] → Volver → [Usuarios] → [Acción] → [Confirmación] → Volver**

# 4.Desarrollo e implementación

## 4.1 Tecnologías utilizadas

### 4.1.1 Frontend

**Tecnologías:**

* **Angular 19** (Framework principal)
* **HTML5 & CSS3** (Estructura y estilos)
* **JavaScript/TypeScript** (Lógica de la aplicación)
* **npm** (Gestión de dependencias)

**Descripción de la Interfaz:**

* **Estructura Modular:**
  + Componentes principales: Dashboard, Gestión de Rutinas, Progreso.
  + Servicios Angular para llamadas API (RutinaService, UsuarioService).
* **Interacciones:**
  + Router para navegación entre vistas.
* **Funcionalidades Clave:**
  + Visualización de rutinas asignadas.
  + Registro de métricas físicas (serie, repeticiones).

### 4.1.2 Backend

**Tecnologías:**

* **Java 23** (Lenguaje principal)
* **Spring Boot** (Framework backend)
* **Maven** (Gestión de dependencias)
* **MySQL** (Base de datos)

**APIs disponibles:**

**Usuarios**

|  |  |  |
| --- | --- | --- |
| **Método** | **Ruta** | **Descripción** |
| GET | /api/usuarios | Obtiene todos los usuarios |
| GET | /api/usuarios/{id} | Obtiene un usuario por ID |
| GET | /api/usuarios?nombre=valor | Filtra por nombre |
| GET | /api/usuarios?rol=entrenador | Obtiene solo entrenadores (filtrados por rol) |
| GET | /api/usuarios?email=valor | Verificar si existe un email |
| POST | /api/usuarios | Crea un nuevo usuario |
| PUT | /api/usuarios/{id} | Actualiza un usuario |
| DELETE | /api/usuarios/{id} | Elimina un usuario |

**Rutinas**

|  |  |  |
| --- | --- | --- |
| **Método** | **Ruta** | **Descripción** |
| GET | /api/rutinas | Obtiene todas las rutinas |
| GET | /api/rutinas/{id} | Obtiene una rutina por ID |
| GET | /api/rutinas/cliente/{cliente\_id} | Rutinas asignadas a un cliente |
| GET | /api/rutinas/entrenador/{entrenador\_id} | Rutinas creadas por un entrenador |
| POST | /api/rutinas | Crea una nueva rutina |
| PUT | /api/rutinas/{id} | Actualiza una rutina existente |
| DELETE | /api/rutinas/{id} | Elimina una rutina |

**Ejercicios**

|  |  |  |
| --- | --- | --- |
| **Método** | **Ruta** | **Descripción** |
| GET | /api/ejercicios | Obtiene todos los ejercicios |
| GET | /api/ejercicios/{id} | Detalles de un ejercicio |
| POST | /api/ejercicios | Crea un nuevo ejercicio |
| PUT | /api/ejercicios/{id} | Actualiza un ejercicio |
| DELETE | /api/ejercicios/{id} | Elimina un ejercicio |

**Gestión de Ejercicios en Rutinas**

|  |  |  |
| --- | --- | --- |
| **Método** | **Ruta** | **Descripción** |
| GET | /api/rutinas/{rutina\_id}/ejercicios | Todos ejercicios de una rutina |
| POST | /api/rutinas/{rutina\_id}/ejercicios | Añadir ejercicio a rutina |
| GET | /api/rutinas/{rutina\_id}/ejercicios/{ejercicio\_id} | Detalle de ejercicio en rutina |
| PUT | /api/rutinas/{rutina\_id}/ejercicios/{ejercicio\_id} | Modificar ejercicio en rutina |
| DELETE | /api/rutinas/{rutina\_id}/ejercicios/{ejercicio\_id} | Quitar ejercicio de rutina |

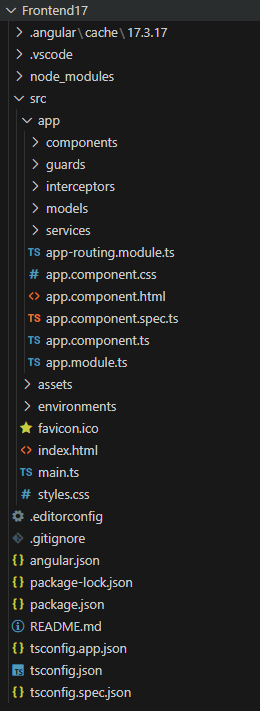
**Progreso**

|  |  |  |
| --- | --- | --- |
| **Método** | **Ruta** | **Descripción** |
| GET | /api/progreso | Todos los registros de progreso |
| GET | /api/progreso/{id} | Progreso específico por ID |
| GET | /api/progreso/usuario/{usuario\_id} | Todo progreso del usuario |
| GET | /api/progreso/usuario/{usuario\_id}/rutina/{rutina\_id} | Progreso en una rutina |
| GET | /api/progreso/usuario/{usuario\_id}/ejercicio/{ejercicio\_id} | Progreso en un ejercicio |
| GET | /api/progreso/usuario/{usuario\_id}/rutina/{rutina\_id}/ejercicio/{ejercicio\_id} | Progreso en ejercicio específico dentro de rutina |
| POST | /api/progreso | Registrar nuevo progreso |
| PUT | /api/progreso/{id} | Actualizar registro progreso |
| DELETE | /api/progreso/{id} | Eliminar registro progreso |

## 4**.2 Organizacion del proyecto**

### 4.2.1 Frontend (Angular)

Vista General de la Organización del Proyecto:

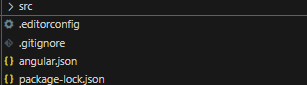


Desglose de partados:

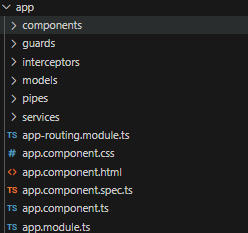
* Inde.html: Es el archivo inicial que se carga al iniciar la app, el cual muestra: el titulo, el icono y carga el componenete de app.



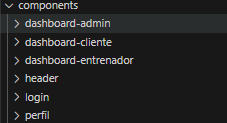
* Angular.json/Package.json: Son los archivos que se encargan de la configuración de Angular(angular.json) y la gestión de dependencias(package.json).



* El directorio app es el núcleo de la aplicación y contiene los siguientes subdirectorios y componentes principales:



* Components: Contiene todos los componentes de la aplicación, organizados por funcionalidad y roles de usuario.



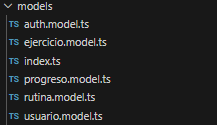
* guards: Aloja los guards que protegen las rutas según los permisos del usuario.



* interceptors: Maneja interceptores HTTP para peticiones y respuestas.



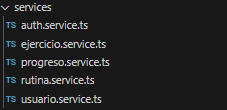
* models: Define las interfaces y clases de datos utilizadas en la aplicación.



* pipes: Contiene pipes personalizados para transformar datos en las vistas.

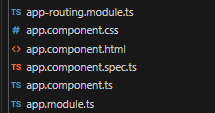


* services: Incluye servicios para lógica de negocio, llamadas a APIs y gestión de estados.



#### **Componentes principales:**

* app-routing.module.ts: Gestiona la navegación, rutas y redirecciones según la URL.
* app.component (HTML/CSS/TS): Componente raíz que muestra el layout común (header, navegación) y renderiza otros componentes según la ruta.
* app.module.ts: Carga todos los módulos, componentes y dependencias globales de la app.

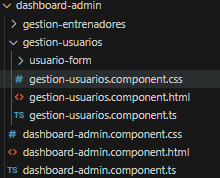


* environment.ts: Define constantes globales como la URL base de la API, claves de terceros, etc.



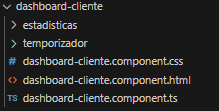
**Dashboard Admin**

* gestion-entrenadores: CRUD de entrenadores.
* gestion-usuarios:
  + usuario-form: Formulario para crear/editar usuarios.



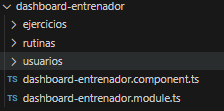
#### **Dashboard Cliente**

* estadisticas: Visualización de progreso del usuario.
* temporizador: Herramienta para sesiones de entrenamiento.



#### **Dashboard Entrenador**

* ejercicios:
  + ejercicios-form: Formulario de ejercicios.
  + ejercicios-list: Listado y filtrado de ejercicios.
* rutinas:
  + rutina-detalle: Vista detallada de una rutina.
  + rutina-form: Creación/edición de rutinas.

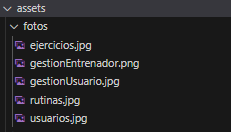


#### **Componentes globales**

* header: Barra de navegación superior.
* login: Página de autenticación.
* perfil: Vista/edición del perfil de usuario.

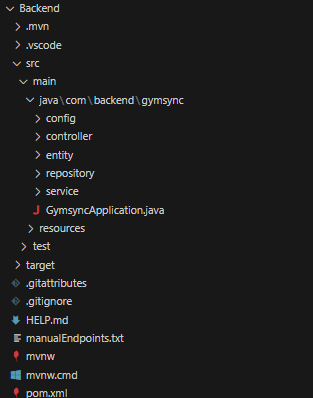


* Asssets: Almacena recursos estáticos como imágenes, fuentes o archivos multimedia.



### 4.2.2 Backend (Spring Boot)

Vista general de la organización del proyecto:



Desglose de apartados:

* **El manual de endpoints:** detalla todas las rutas de la API, incluyendo métodos HTTP, parámetros, ejemplos y requisitos de seguridad.



* **Pom.xml**: gestiona las dependencias, plugins y configuración de construcción.



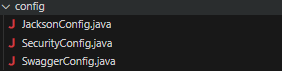
* **Target**: almacena el JAR generado y los archivos compilados.



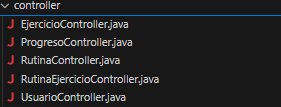
* **application.properties**: define configuraciones globales como conexiones a BD, puertos y variables de entorno. Al mismo nivel esta la bd



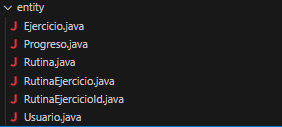
* **Config**: contiene clases especializadas como SecurityConfig (seguridad), SwaggerConfig (documentación) y JacksonConfig (serialización JSON)



* **controller**: manejan las peticiones HTTP



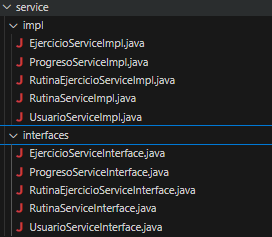
* **entity**: representan el modelo de datos



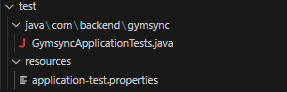
* **repository**: gestionan el acceso a la base de datos



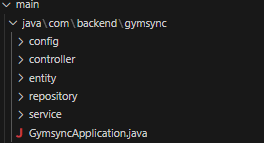
* **service**: la lógica de negocio se organiza en servicios (service/ e impl/)



* **test**: las pruebas automatizadas (test/) garantizan la calidad con unitarias, de integración y específicas para repositorios y controladores.



* **GymSyncApplicaction.java**: El proyecto sigue la estructura estándar de Spring Boot, este archivo inicia el contexto de Spring y carga toda la configuración automáticamente.



## 4.3 Difucultades Técnicas durante el desarrollo

Durante la creación de GymSync —una aplicación web para gestión de entrenamientos y progreso en gimnasios— enfrenté varios desafíos técnicos clave. A continuación, detallo los más relevantes y cómo los resolví:

Integración Angular-Spring Boot (API REST)

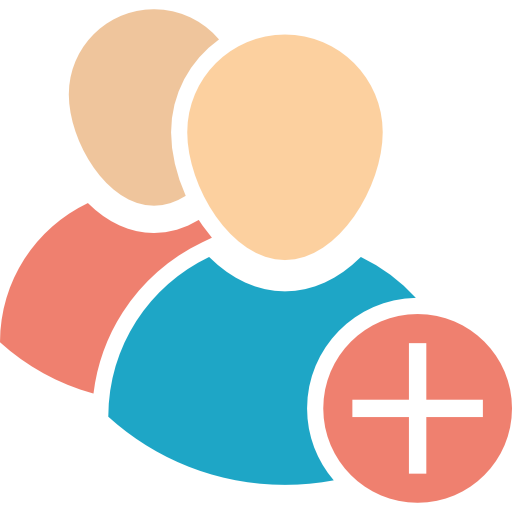
* Problema:
  + La comunicación entre el frontend (Angular) y el backend (Spring Boot) presentó errores críticos al inicio:
  + Errores de CORS al hacer peticiones HTTP (GET/POST).
  + Inconsistencias en los DTOs: Los modelos de Angular no coincidían con los del backend.
  + Manejo de errores: Las respuestas de la API (401, 500) no se gestionaban correctamente en el frontend.
* Solución:
  + Configuré CORS global en Spring Boot (@CrossOrigin en controladores + WebMvcConfig).
  + Estandaricé los DTOs (clases con los mismos campos en ambos lados).
  + Implementé interceptores en Angular para manejar errores (ej.: redirigir a login si el token JWT expira).
* Aprendizaje:
  + La importancia de definir contratos claros (Swagger/OpenAPI) y usar herramientas como Postman para probar endpoints antes de integrarlos.

## 4.4 Funcionalidades principales

La aplicación GymSync está diseñada para gestionar rutinas de entrenamiento, ejercicios y el progreso de los usuarios. A continuación, se detallan las principales funcionalidades del sistema:

Autenticación y seguridad

* Inicio de sesión seguro: Los usuarios pueden acceder al sistema mediante credenciales verificadas (email y contraseña).
* Gestión de roles: Se asignan distintos permisos según el tipo de usuario (Cliente, Entrenador o Administrador).
* Cambio de contraseña: Los usuarios pueden actualizar su contraseña de manera segura.

Gestión de usuarios

* Registro y edición de usuarios: Permite crear, actualizar y eliminar cuentas de clientes y entrenadores.
* Búsqueda y filtrado: Los usuarios pueden ser filtrados por nombre, rol o email.

Gestión de ejercicios

* Catálogo de ejercicios: Los ejercicios se organizan por tipo (pecho, abdominales, piernas, etc.) y pueden incluir descripciones, imágenes y videos demostrativos.
* CRUD de ejercicios: Los administradores pueden crear, editar y eliminar ejercicios.

Gestión de rutinas

* Creación de rutinas: Los entrenadores pueden diseñar rutinas personalizadas para los clientes, asignando ejercicios específicos con series, repeticiones y pesos recomendados.
* Asignación de ejercicios: Las rutinas pueden incluir múltiples ejercicios con configuraciones detalladas.
* Visualización por cliente/entrenador: Las rutinas pueden filtrarse por cliente o entrenador.

Seguimiento del progreso

* Registro de progreso: Los clientes pueden registrar su desempeño en cada ejercicio (series, repeticiones, peso utilizado).
* Historial de progreso: Los usuarios pueden visualizar su evolución en ejercicios específicos o rutinas completas.
* Filtrado avanzado: El progreso puede consultarse por usuario, rutina, ejercicio o combinaciones de estos.

Panel de administración

* Dashboard intuitivo: Ofrece una visión general de clientes y entrenadores.
* Acceso controlado: Solo usuarios con permisos de administrador pueden gestionar entrenadores y usuarios.

Interfaz y accesibilidad

* Navegación intuitiva: Diseño optimizado para una experiencia de usuario fluida.
* APIs RESTful: Endpoints bien estructurados para integración con frontend y aplicaciones móviles.

# 5. Seguridad

* Autenticación y Autorización: Se ha implementado un sistema de autenticación basado en JWT (JSON Web Token), el cual es un estándar de seguridad que permite la verificación de identidad de usuarios mediante tokens. Este token se envía al cliente y se debe incluir en cada solicitud HTTP para acceder a recursos protegidos.
  + Formato del JWT: Está compuesto por tres partes codificadas en Base64:
    - Header: Contiene el tipo de token y el algoritmo de firma usado (Ej. HMAC SHA256).
    - Payload: Contiene los datos que describen al usuario (Ej. su rol, ID, etc.).
    - Signature: Se usa para verificar la autenticidad del token.

Cada endpoint de la API cuenta con un mecanismo de autorización basado en roles. Esto significa que antes de acceder a un recurso específico, el sistema verifica el JWT del usuario y determina si tiene los permisos adecuados.

* Protección contra ataques: El uso de JWT también ayuda a proteger la API contra ciertos tipos de ataques, como:
  + Ataques de reproducción (Replay Attacks): JWT incluye una fecha de expiración (exp), evitando que tokens antiguos puedan reutilizarse después de su vencimiento.
  + Cross-Site Request Forgery (CSRF): JWT es enviado en los encabezados de las peticiones HTTP (Authorization: Bearer <token>), lo que reduce el riesgo de ataques CSRF en comparación con sesiones almacenadas en cookies.
* Encriptación de Contraseñas: Para proteger las contraseñas almacenadas en la base de datos se usa la biblioteca Bcrypt, que proporciona un método seguro de almacenamiento.
* Cómo funciona Bcrypt:
  + Aplica múltiples iteraciones de hashing para aumentar la complejidad de descifrado.
  + Usa un "sal" (salt), un valor aleatorio agregado a la contraseña antes de aplicar el hash, lo que previene ataques de diccionario.
  + Genera un hash único incluso para contraseñas idénticas debido al uso de salt.

# 6. Pruebas y Validaciones

## 6.1 Pruebas Unitarias

* Herramientas utilizadas: Postman / Swagger (para probar los endpoints de la API).
* Pruebas realizadas:
  + Creación de entidades:
    - Usuarios (clientes, entrenadores, administradores).
    - Ejercicios (con tipos, descripciones y multimedia).
    - Rutinas (asignación de ejercicios, series, repeticiones).
    - Progresos (registro de entrenamientos).
  + Actualización de entidades:
    - Modificación de datos de usuarios.
    - Edición de ejercicios (nombre, tipo, descripción).
    - Ajustes en rutinas (cambio de ejercicios, series, repeticiones).
    - Actualización de progresos (peso, repeticiones, observaciones).
  + Borrado de entidades:
    - Eliminación de usuarios.
    - Eliminación de ejercicios.
    - Eliminación de rutinas y sus relaciones.
    - Borrado de registros de progreso.
  + Visualización de datos:
    - Consulta de usuarios por rol (cliente, entrenador).
    - Listado de ejercicios por tipo (pecho, piernas, etc.).
    - Rutinas asignadas a un cliente o creadas por un entrenador.
    - Historial de progreso por usuario, rutina o ejercicio.

## 6.2 Pruebas de Integración

1. Pruebas de autenticación:

* ✔ Inicio de sesión correcto: Validación de credenciales válidas (email + contraseña).
* ✖ Inicio de sesión erróneo: Contraseña incorrecta, usuario no existente.
* 🔐 Pruebas por rol:
  + Acceso como cliente (solo puede ver sus rutinas y progresos).
  + Acceso como entrenador (puede gestionar rutinas de sus clientes).
  + Acceso como administrador (control total sobre usuarios).

1. Pruebas como cliente:

* 📅 Visualización de sus rutinas asignadas.
* 🏋️ Registro de progreso en ejercicios (series, repeticiones, peso).
* 🚫 Intentos de acceso no autorizado:
  + Modificar rutinas de otros usuarios.
  + Eliminar ejercicios o rutinas.

1. Pruebas como entrenador:

* ➕ Creación y asignación de rutinas a clientes.
* ✏ Edición de rutinas (agregar/eliminar ejercicios).
* 📊 Visualización del progreso de sus clientes.
* 🚫 Restricciones:
  + No puede eliminar usuarios ni ejercicios del sistema.

1. Pruebas como administrador:

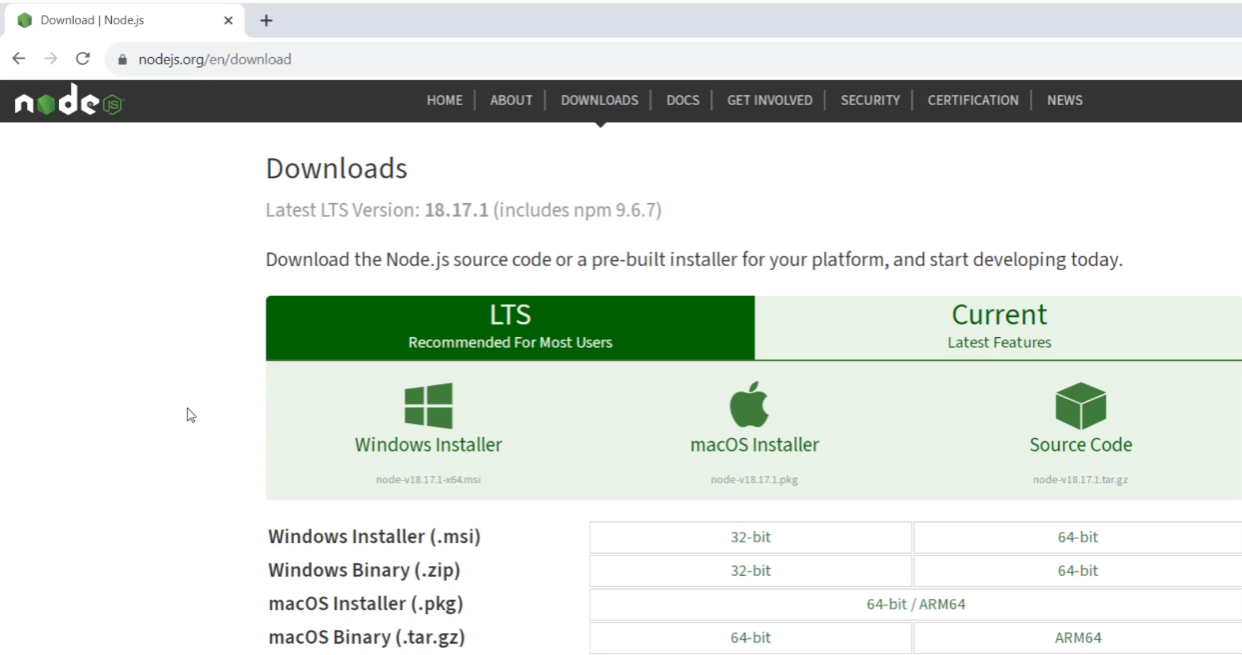
* 👥 Gestión de usuarios (creación, edición, eliminación).

# 7. Despliegue

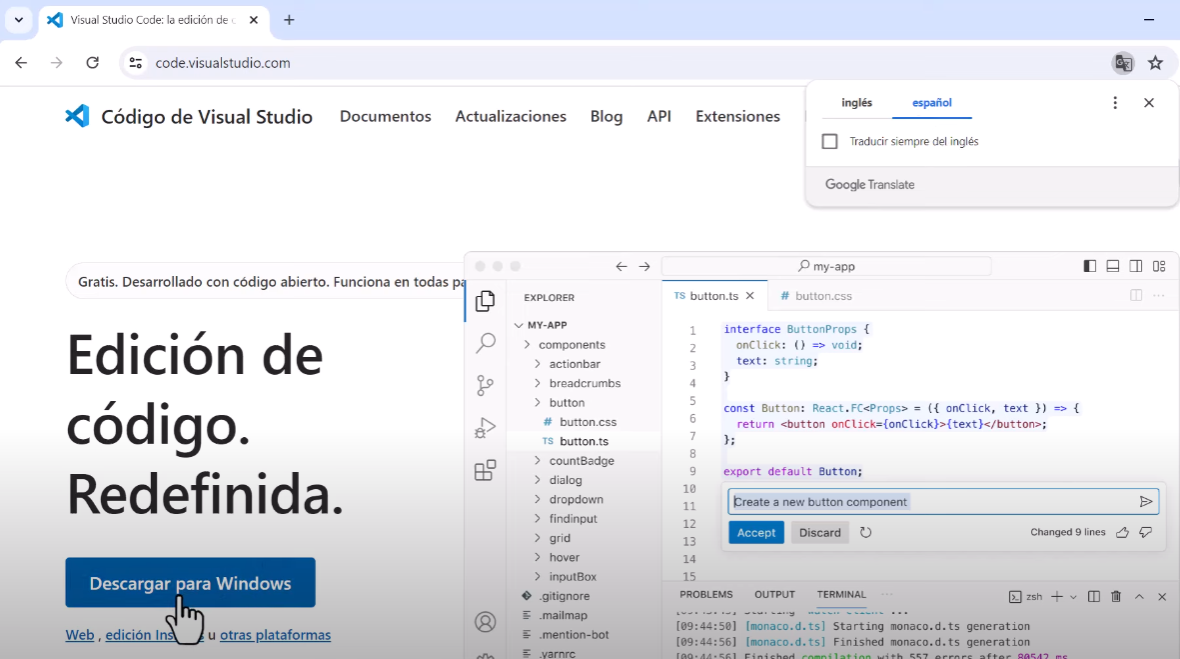
Instalación del entorno

* Node.js 22.13.1

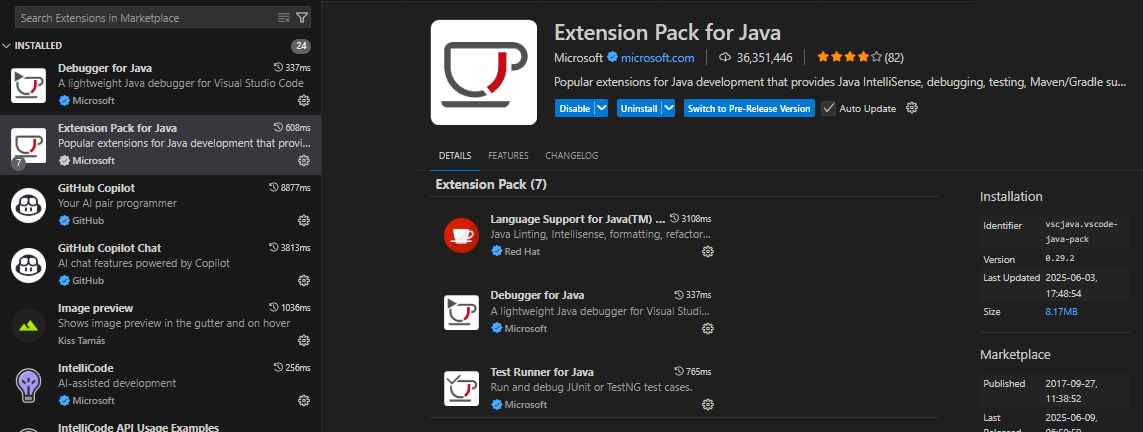
Incluye ya NPM



* Visual Studio Code(extensiones: Spring boot, Maven for java)



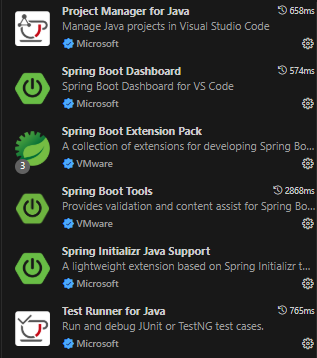
Extensiones para Visual Studio Code:





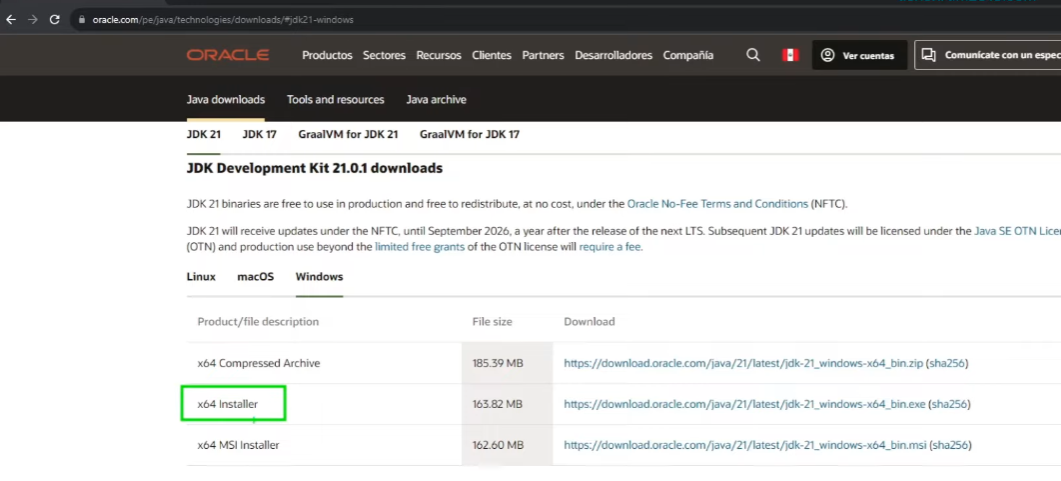


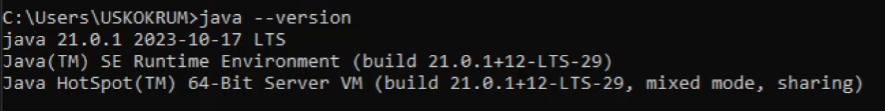
Otras extensiones utiles:

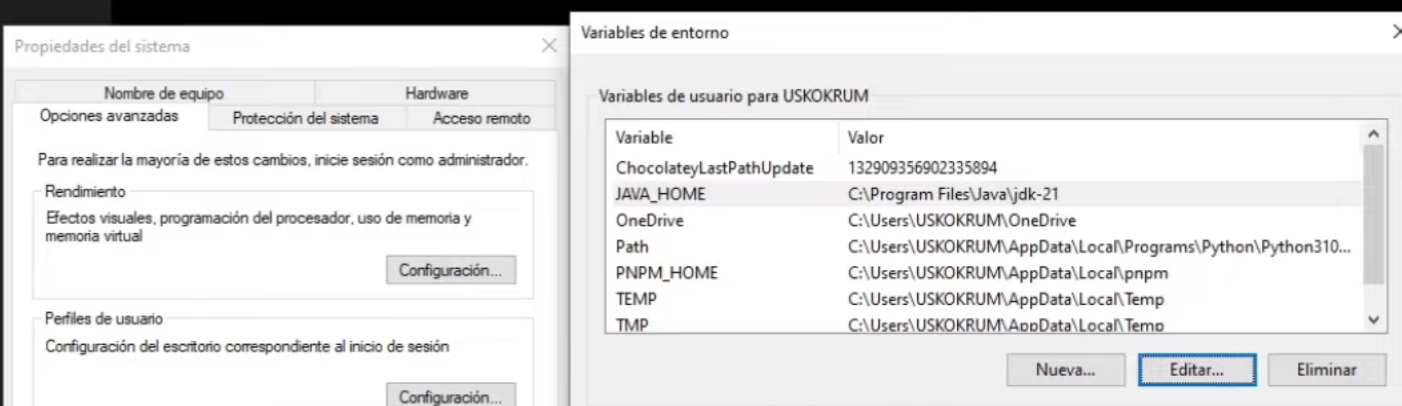


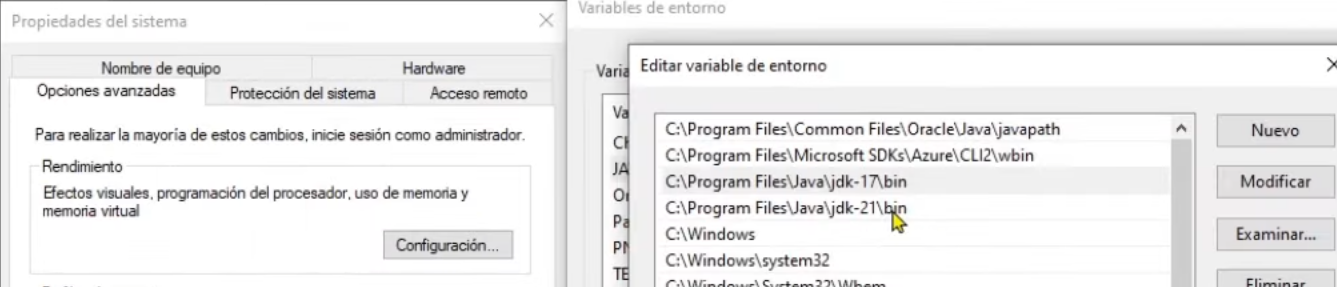
* Java 21

Buscamos una version LTS de java

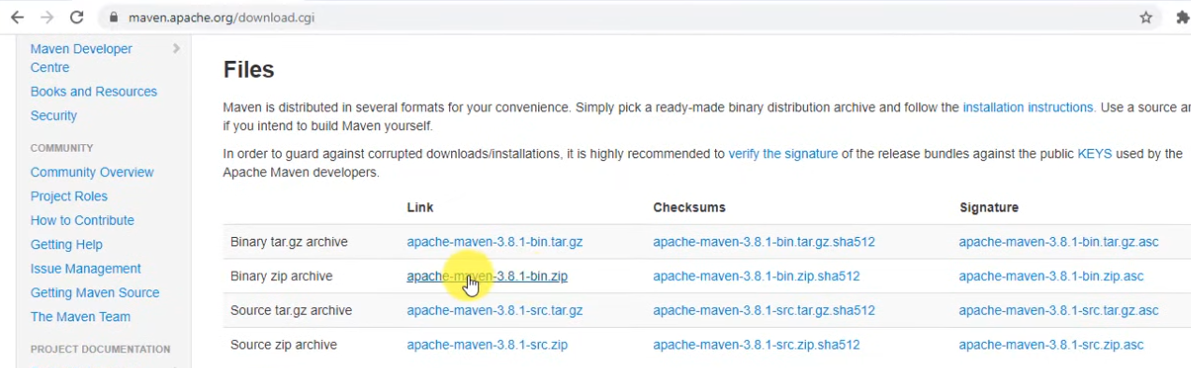


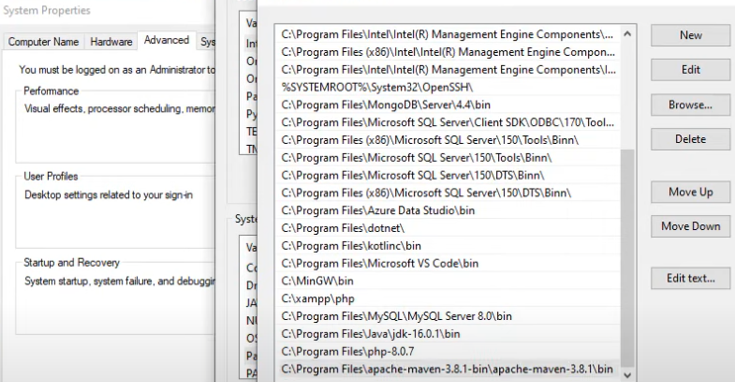






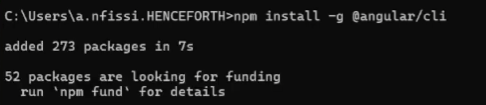
* Apache Maven 3.9.9

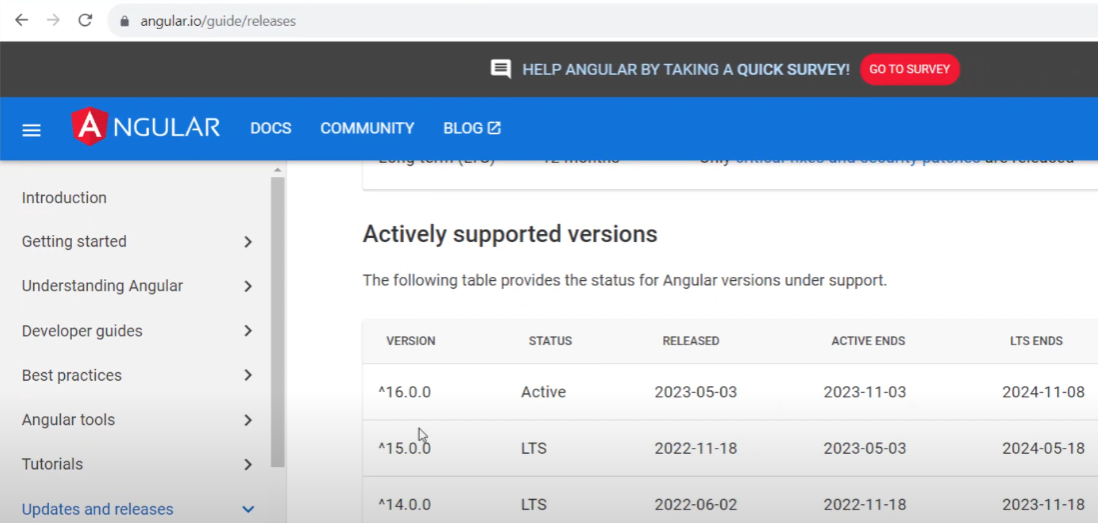


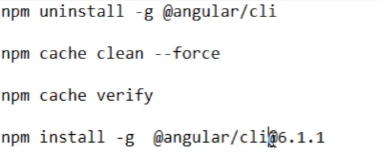
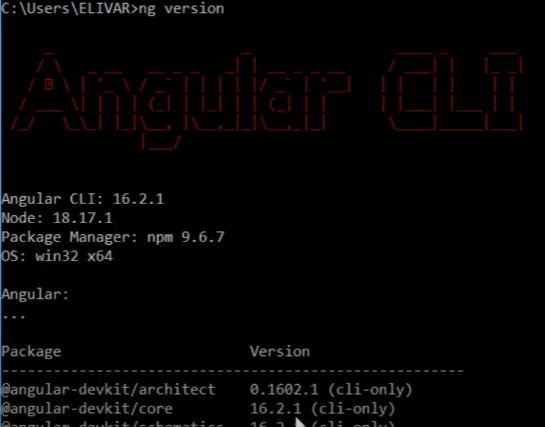


* Angular 19.2.9

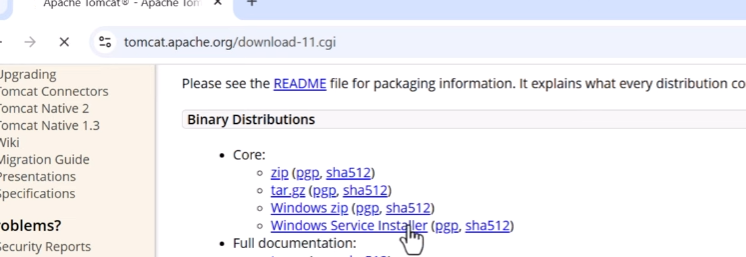
Hay que instalar primero node.js

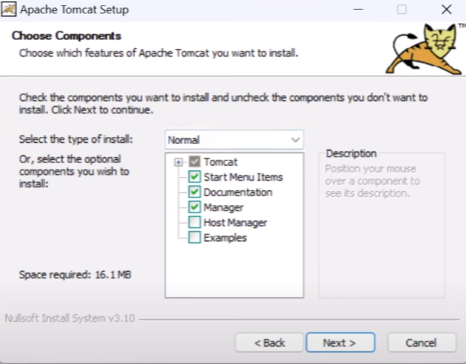


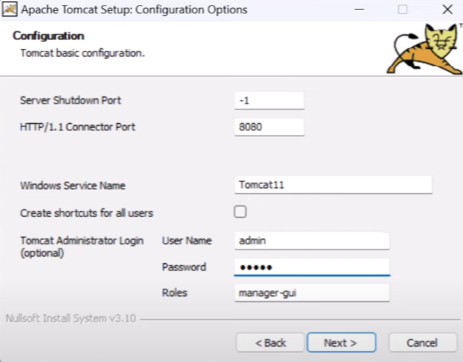


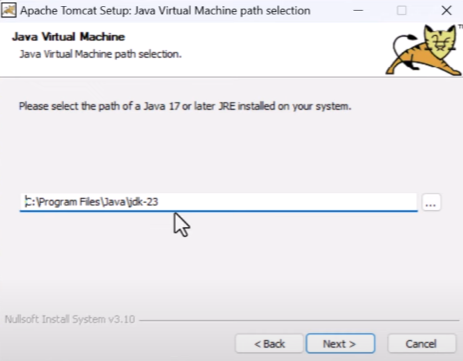


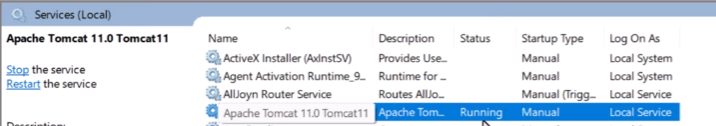
* Tomcat 11.0



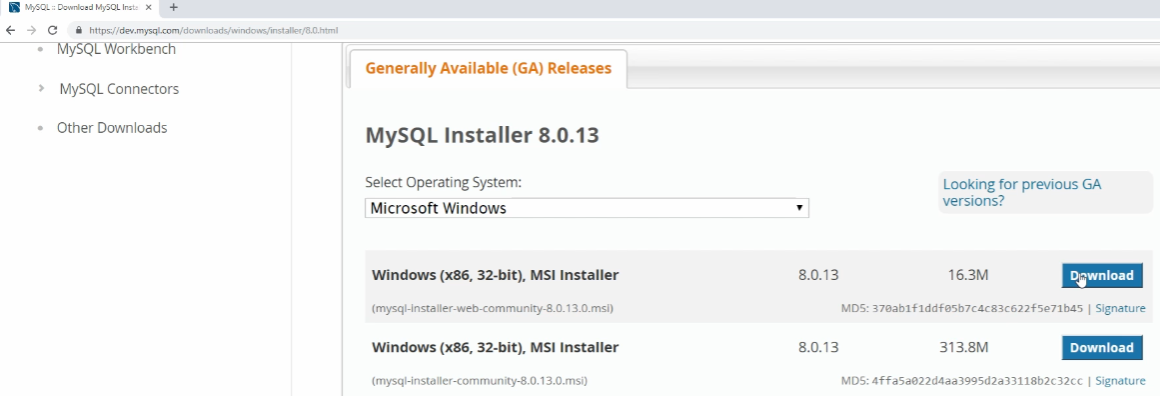


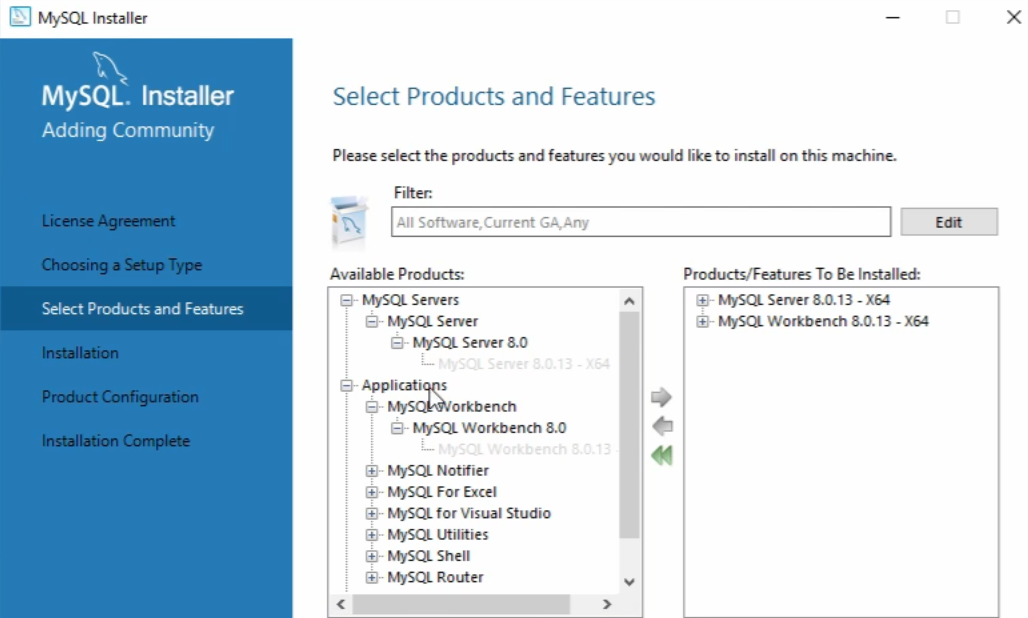
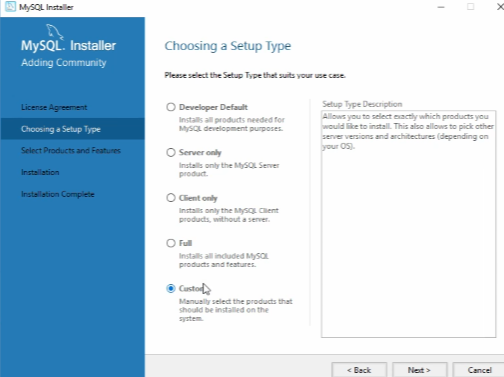






* Base de datos: MySQL





# 8. Plan de Negocio – Modelo SaaS

Plan de Negocio: GymSync - Gestión de Entrenamientos para Gimnasios

Ubicación: Valladolid, España

Sector: Tecnología + Fitness

**1. Resumen Ejecutivo**

* Producto: Software SaaS para gimnasios que gestiona entrenamientos personalizados, seguimiento de clientes y comunicación entrenador-usuario.
* Oportunidad: 85% de gimnasios en Valladolid usan métodos manuales (encuestas propias).
* Ventaja Competitiva:
  + Diseñado para necesidades locales (ej: integración con eventos deportivos de Valladolid).
  + Precios 100% más bajos que soluciones nacionales (ej: WodGym).
* Objetivo 1er Año: Hacerse conocer en las redes sociales.

**2. Análisis de Mercado (Valladolid)**

Público Objetivo:

* Gimnasios pequeños/medianos: 45 establecimientos (Fuente: Registro Mercantil de Valladolid).
* Entrenadores independientes: +120 registrados en el Colegio de Entrenadores de CyL.

Competencia Local:

|  |  |  |
| --- | --- | --- |
| **Competidor** | **Fortalezas** | **Debilidades (oportunidades)** |
| WodGym | Funciones avanzadas | Precio alto (80€/mes) |
| FitMaster | Diseño intuitivo | Sin soporte en español |
| **GymSync** | *Precio competitivo + Enfoque local* | *Nuevo en el mercado* |

**3. Modelo de Negocio**

Precios (Monetización):

|  |  |  |
| --- | --- | --- |
| **Plan** | **Precio** | **Características** |
| **Básico** | 29€/mes | 1 admin + 2 entrenadores + 50 clientes |
| **Premium** | 59€/mes | Admins ilimitados + App móvil + Estadísticas avanzadas |

Canales de Venta:

* Directo: Visitas a gimnasios (zonas: Huerta del Rey, Pilarica, Centro).
* Digital:
  + Facebook/Instagram Ads (dirigido a dueños >15 años en Valladolid).
  + Colaboraciones con influencers fitness locales (ej:@fit\_valladolid).

**4. Plan Operativo (Valladolid)**

Recursos Necesarios:

* Desarrollo: Tú (Angular/Spring Boot) + 1 freelancer para app móvil (4000€ iniciales).
* Local: Trabajo remoto (ahorra costes). Oficina eventual en Vivero de Empresas de Valladolid (gratis primer año).

Proveedores

* Clave: Hosting: Gigas (empresa castellano-leonesa).
* Mentoría: Programa Valladolid StartUp (asesoría gratuita).

5. Estrategia Comercial

* Piloto gratuito: Ofrecer 3 meses gratis a 3 gimnasios referentes (ej: Gimnasio Huerta del Rey).
* Eventos presenciales:
  + Stand en la Feria de Deporte de Valladolid (octubre 2024).
  + Charlas en Centros Cívicos (Rondilla, Delicias).
* Alianzas:
  + Universidad de Valladolid: Oferta especial para estudiantes INEF.
  + Federación de Fútbol de CyL: Gestión de entrenamientos para equipos base.

**6. Finanzas Proyectadas**

|  |  |  |
| --- | --- | --- |
| **Concepto** | **Coste/Año** | **Ingresos/Año (Año 1)** |
| Desarrollo App | 5.000€ | **Suscripciones:** 8.400€ (15 clientes x 59€ x 12 meses) |
| Marketing | 2.000€ | **Implementación:** 2.250€ (15 x 150€) |
| Gastos Varios | 1.500€ | **Total Ingresos:** **10.650€** |
| **Total Gastos** | **8.500€** | **Beneficio Neto:** **2.150€** |

**7. Ayudas y Subvenciones (Valladolid)**

* IPE: Ayudas a emprendedores (Junta de CyL) → Hasta 10.000€.
* ENISA: Préstamos para startups tecnológicas (sin aval).
* Vivero de Empresas: Espacio gratis + mentorización (solicitar en valladolidadelante.es).

**8. Roadmap Año 1**

|  |  |
| --- | --- |
| **Trimestre** | **Acciones Clave** |
| **T1** | MVP funcional + Captar 3 gimnasios piloto |
| **T2** | Lanzamiento oficial + Participación en Feria de Deporte |
| **T3** | Alianzas con universidades + App móvil |
| **T4** | Expansión a Palencia/Burgos (15 clientes total) |

**9. Riesgos y Mitigación**

|  |  |
| --- | --- |
| **Riesgo** | **Solución** |
| **Baja adopción gimnasios** | Ofertas piloto extendidas + demostraciones in situ |
| **Competencia nacional** | Focalización en servicio personalizado local |
| **Retrasos desarrollo** | MVP mínimo viable con Spring Boot + Angular |

**10. Documentos Necesarios (Tramitar en Valladolid)**

1. Alta de autónomo: Gestoría (coste ~250€).
2. Protección de datos: Adaptación LOPD (asesoría especializada).
3. Marca registrada: Oficina Española de Patentes (€250).

# 9. Conclusiones

El desarrollo de GymSync ha sido una experiencia profundamente enriquecedora, tanto a nivel técnico como personal. A lo largo de este proyecto, he consolidado conocimientos clave y descubierto nuevas herramientas que han transformado mi enfoque del desarrollo de software.

**Aprendizajes técnicos destacados**

* Arquitectura moderna:

La integración de Spring Boot (Java) para el backend y Angular (TypeScript) para el frontend me permitió dominar el flujo completo de una aplicación RESTful, desde la creación de APIs robustas hasta la construcción de interfaces dinámicas.

* Frameworks revolucionarios:

Herramientas como Spring Data JPA (para persistencia con MySQL), Spring Security (autenticación JWT) y Angular Material (UI/UX) optimizaron procesos complejos, mejorando la escalabilidad y mantenibilidad del código.

* Nuevos paradigmas:

Profundicé en conceptos avanzados como:

* + Relaciones Many-to-Many (rutinas-ejercicios).
  + Gestión de roles (cliente, entrenador, administrador).
  + Serialización JSON con @JsonManagedReference y @JsonBackReference para evitar ciclos infinitos.

**Superación de desafíos**

* Integración de entidades complejas:

La estructura de RutinaEjercicio con clave compuesta (RutinaEjercicioId) requirió un diseño cuidadoso de servicios y controladores.

* Seguimiento del progreso:

La entidad Progreso con triple relación (usuario, rutina, ejercicio) demandó consultas JPQL personalizadas para filtrados avanzados.

**Crecimiento personal**

* De la teoría a la práctica:

Convertir requisitos abstractos (como "registrar series/repeticiones") en flujos de trabajo reales (ProgresoController) reforzó mi capacidad para traducir necesidades de usuarios en funcionalidades técnicas.

* Satisfacción creativa:

Ver cómo GymSync resuelve problemas reales (gestión de entrenamientos, evolución física de clientes) ha sido inmensamente gratificante. Cada endpoint representa horas de depuración y aprendizaje.

**Reflexión final**

GymSync no es solo una app: es un testimonio de cómo la perseverancia y la curiosidad tecnológica pueden materializar ideas que impactan positivamente en hábitos de salud. Este proyecto marca un hito en mi viaje como desarrollador, y siento orgullo al compartirlo con la comunidad.

«Código que mueve pesos, también mueve vidas.»

— Zecheng Zhou

# 10.Bibliografía

1. Frontend

* Angular CLI
  + Author/a: Google
  + Año: 2016
* Node.js
  + Author/a: Ryan Dahl
  + Año: 2009

1. Backend

* [MySQL Workbench](https://dev.mysql.com/downloads/workbench/)
  + Author/a: Oracle
  + Año: 2005
* Java JDK 17+
  + Author/a: Oracle
  + Año: 1995 (primera versión)
* Apache Maven
  + Author/a: Apache Software Foundation
  + Año: 2004

1. Herramientas Adicionales

* [Visual Studio Code](https://code.visualstudio.com/download)
  + Author/a: Microsoft
  + Año: 2015
* [Postman](https://www.postman.com/downloads/)
  + Author/a: Postman Inc.
  + Año: 2012
* [Git](https://git-scm.com/downloads)
  + Author/a: Linus Torvalds
  + Año: 2005

1. Videos de Youtube

* [Curso Spring](https://www.youtube.com/watch?v=kFIvslQQZ9k&list=PLU8oAlHdN5Blq85GIxtKjIXdfHPksV_Hm)
* [JWT Security](https://youtu.be/-Z4a0bKr2Pg?si=O-00yzJg22Y9cOkp)
* [Login con JWT Autherntication](https://youtu.be/nwqQYCM4YT8?si=A_YdS-GIcLkM770W)
* [Curso Angular](https://www.youtube.com/watch?v=soInCF7nbDw)
* [Spring Security](https://www.youtube.com/watch?v=KBvBY5qyfEM)

Cosas que faltan:

* Normalizar base de datos (3FN)
* Diagrama de Gantt
* Mockup o wireframe
* Manual de usuario