

# stratton app

[asesoramiento energético]

Aplicación web para la gestión de empleados y clientes de una asesoría energética.

---

Álvaro Rodríguez Molina    
2ºDAW.

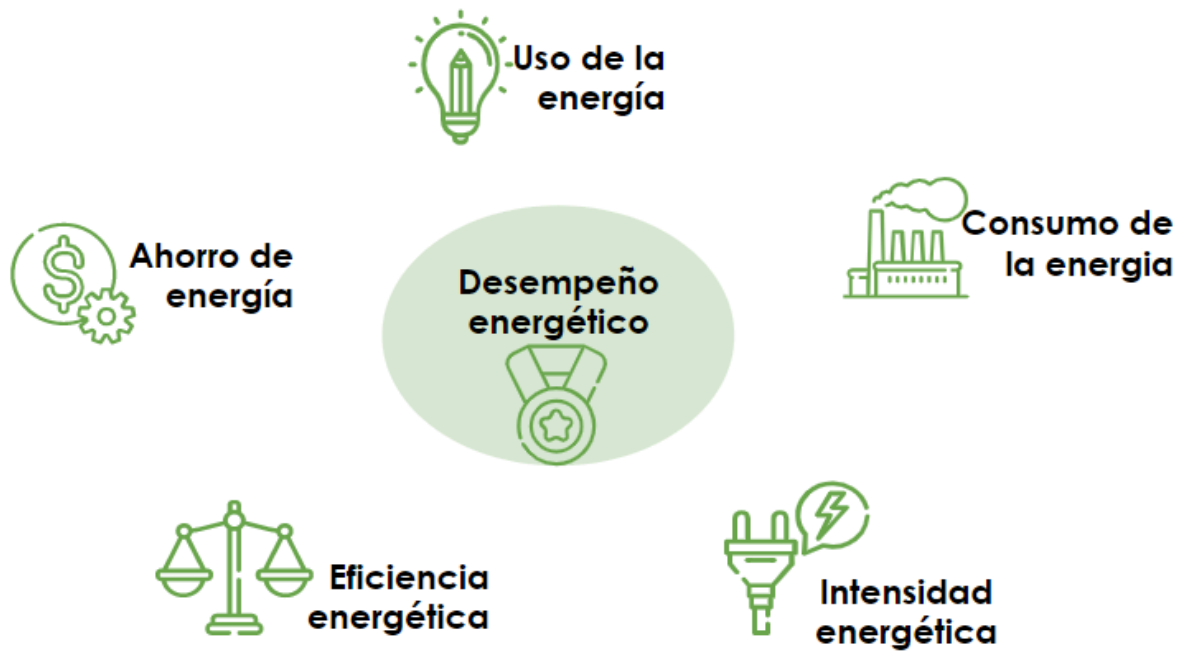


# ÍNDICE

1. Objetivos
2. Escenario
  - a. Descripción
3. Fundamentos teóricos y conceptos
  - a. Software
    - i. Spring Boot
    - ii. Angular
    - iii. MySQL
    - iv. Docker
    - v. Git
  - b. Herramientas de Desarrollo
    - i. Spring Boot Tools con Eclipse
    - ii. Visual Studio Code
    - iii. GitHub Desktop
    - iv. Docker Desktop
4. Descripción
  - 4.1 Demostración del escenario (asesoramiento energético)
    - 4.1.2 Pasos a seguir
  - 4.2 Instalaciones y configuraciones
    - 4.2.1 MySQL
    - 4.2.2 Docker
    - 4.2.3 Configuración de Spring Security con JWT
5. Manual de despliegue Docker
6. Conclusión
7. Referencias
  - 7.1 Recursos
  - 7.2 Presentación
  - 7.3 Bibliografía

# 1. Objetivos

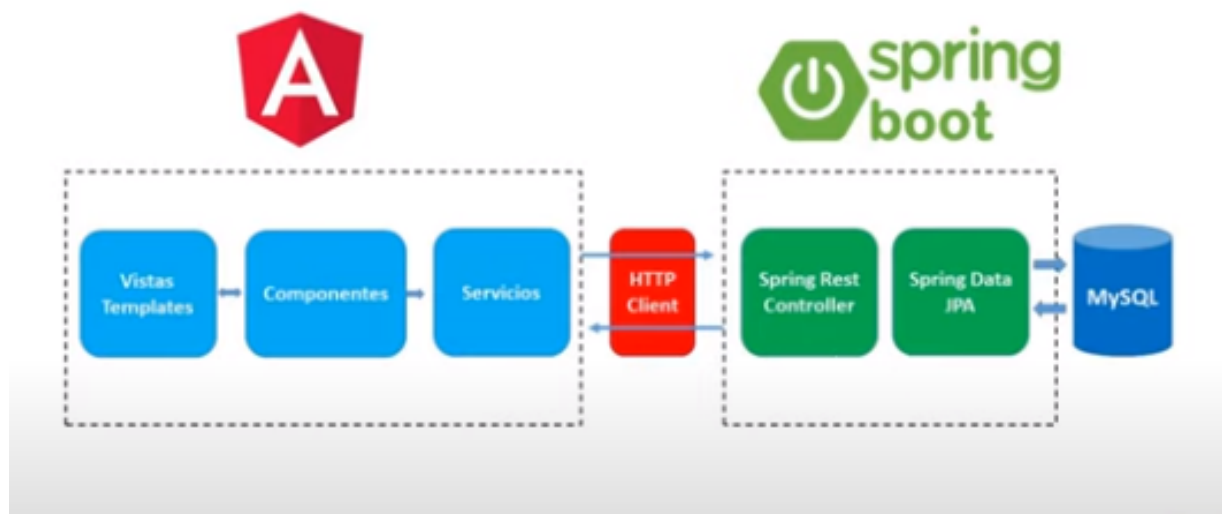
El objetivo principal de este proyecto es desarrollar una aplicación de asesoramiento energético utilizando tecnologías modernas de desarrollo web y herramientas de contenedorización y orquestación. La aplicación proporcionará a los usuarios recomendaciones sobre cómo mejorar la eficiencia energética en sus hogares o empresas.



## 2. Escenario

### 2.1 Descripción

El proyecto se centra en la creación de una plataforma de asesoramiento energético que permite a los usuarios obtener sugerencias personalizadas para optimizar el uso de la energía. La plataforma se compondrá de un backend desarrollado con Spring Boot y un frontend creado con Angular. La infraestructura de despliegue utilizará Docker para garantizar un desarrollo continuo y una implementación eficiente



## 3. Fundamentos teóricos y conceptos

### Spring Boot

Spring Boot es un framework para el desarrollo de aplicaciones Java que facilita la creación de aplicaciones independientes y productivas. Ofrece configuraciones predeterminadas para los componentes más comunes y simplifica la gestión de dependencias.

### Angular

Angular es un framework para el desarrollo de aplicaciones web de una sola página (SPA) desarrollado por Google. Facilita la creación de aplicaciones dinámicas y escalables con una arquitectura basada en componentes.

Componentes principales de Angular:

1. **Componentes:** Son las unidades básicas de una aplicación Angular. Cada componente consiste en:
  - **Template (HTML):** Define la estructura y el contenido de la vista.
  - **Styles (CSS):** Define el estilo específico del componente.
  - **Class (TypeScript):** Contiene la lógica del componente, incluyendo propiedades y métodos.
  -
2. **Módulos:** Agrupan componentes, directivas, servicios y otros módulos. El módulo raíz es `AppModule`, que se define en `app.module.ts`. En Angular 18 ya no es así y los módulos principales se diferencian en uno para la configuración `app.config.ts` otro general para los componentes y otro para las rutas.
3. **Servicios:** Son clases que contienen lógica compartida entre diferentes componentes. Se utilizan para realizar tareas como llamadas HTTP a la API backend.
4. **Rutas:** Permiten la navegación entre diferentes vistas o componentes de la aplicación. Se definen en el módulo de enrutamiento (`app.routes.ts`).
5. **Directivas:** Son instrucciones en el DOM que indican a Angular cómo modificar o manipular elementos.
6. **Pipes:** Transforman los datos mostrados en una plantilla. Un ejemplo común es el pipe `| date` para formatear fechas.

### MySQL

MySQL es un sistema de gestión de bases de datos relacional que se utiliza para almacenar los datos de la aplicación. Proporciona una estructura robusta y eficiente para manejar grandes volúmenes de datos.

## Docker

Docker es una plataforma de contenerización que permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores. Facilita la portabilidad y la escalabilidad de las aplicaciones.

## Git

Git es un sistema de control de versiones distribuido que permite a los desarrolladores rastrear los cambios en el código fuente durante el desarrollo del software.

Comandos básicos de Git:

Inicializar un repositorio:

```
git init
```

Clonar un repositorio:

```
git clone <url-del-repositorio>
```

Añadir cambios al área de preparación:

```
git add <archivo-o-directorio>
```

Confirmar cambios:

```
git commit -m "Mensaje de confirmación"
```

Ver estado del repositorio:

```
git status
```

Ver historial de confirmaciones:

```
git log
```

La aplicación se desplegará en Docker

## 3.2 Herramientas de Desarrollo

### Spring Boot Tools con Eclipse

Eclipse es un entorno de desarrollo integrado (IDE) para programar en Java. Spring Boot Tools es un conjunto de herramientas que facilita el desarrollo de aplicaciones Spring Boot dentro de Eclipse.

### Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft. Es ligero pero potente y soporta múltiples lenguajes de programación, incluyendo TypeScript y HTML, que son esenciales para el desarrollo con Angular.

### GitHub Desktop

GitHub Desktop es una aplicación de escritorio que proporciona una interfaz gráfica para trabajar con repositorios Git, facilitando el control de versiones y la colaboración en proyectos.

### Docker Desktop

Docker Desktop es una aplicación para MacOS y Windows que permite a los desarrolladores construir, compartir y ejecutar aplicaciones en contenedores Docker desde sus escritorios.




## 4. Descripción



### 4.1 Demostración del escenario (asesoramiento energético)

A partir de la experiencia, el mercado eléctrico es algo confuso y se hace popular por la cantidad de spam que recibimos a diario, con esta aplicación nos mantendremos siempre al tanto de todas las ofertas, donde en nuestro blog damos pautas de que contratar y que no a parte de consejos de asesoramiento profesional y gratuito sin compromiso.

Para los empleados de la empresa tenemos una interfaz intuitiva de gestión de clientes junto con filtros de búsqueda avanzada que facilitarán mucho la experiencia del usuario.









 ARSYNC

Contratos

 álvaro rodríguez molina 

BÚSQUEDA AVANZADA

Subir Contrato

| user   | Fecha      | Nombre/Razón Social        | Compañía  | CUPS                 | Estado  |
|--------|------------|----------------------------|---|----------------------|---|
| alvrch | 2024-04-11 | LUISA BEGOÑA MUNICIO SANZ  |  | ES0021000005213375MP |  |
| alvrch | 2024-04-11 | LUISA BEGOÑA MUNICIO SANZ  |  | ES0022000005277010VS |  |
| alvrch | 2024-04-11 | TERESA DE JESUS LOPEZ ROSA |  | ES0031101746174008PN |  |
| alvrch | 2024-04-11 | TERESA DE JESUS LOPEZ ROSA |  | ES0205170662056007LB |  |





## Solicitud

## Datos del cliente

Nombre:

Apellidos:

DNI:

Correo:

Dirección:

Teléfono:

IBAN:

Asesoría:

## Datos del suministro

LUZ: ☐

Compañía:

CUPS:

Potencia:

Tarifa:

SVA: ☐GAS: ☐

Compañía:

CUPS:

Producto:

Tarifa:

SVA: ☐Factura Electrónica: ☐

Observaciones:

Adjuntar Archivos 

Guardar como borrador

Confirmar

### 4.1.2 Pasos a seguir

1. Instalación y configuración del entorno de desarrollo:
  - Instalar JDK 17 para ejecutar aplicaciones Spring Boot.
  - Instalar Node.js y npm para ejecutar aplicaciones Angular.
  - Instalar Docker Desktop para gestionar contenedores.
2. Configuración del backend con Spring Boot:
  - Crear un nuevo proyecto Spring Boot.
  - Añadir las dependencias necesarias en el archivo `pom.xml`.
  - Configurar la conexión a la base de datos MySQL.
3. Desarrollo del frontend con Angular:
  - Crear un nuevo proyecto Angular.
  - Diseñar y desarrollar componentes y servicios.
  - Implementar la comunicación con el backend utilizando HttpClient.
4. Despliegue con Docker:
  - Crear un archivo `Dockerfile` para el backend y otro para el frontend.
  - Crear un archivo `docker-compose.yml` para orquestar los contenedores.

## 4.2 Instalaciones y configuraciones

### 4.2.1 MySQL

Instalar y configurar MySQL para la aplicación. Crear una base de datos y un usuario con los permisos adecuados.

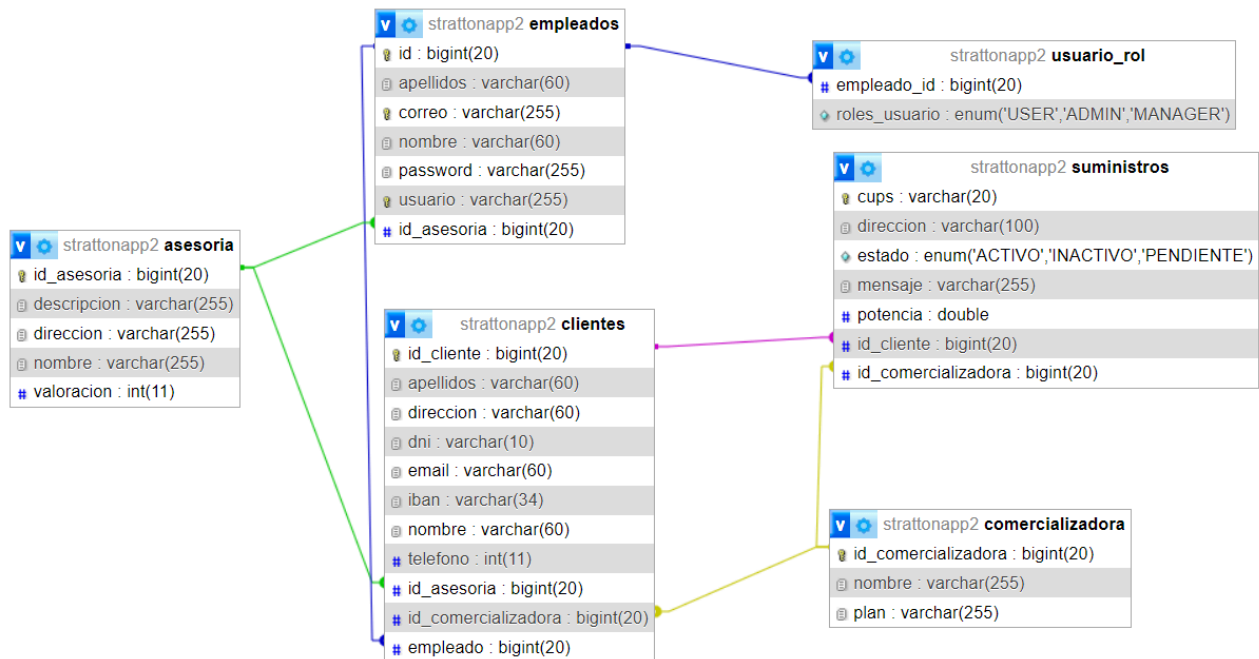
```
CREATE DATABASE strattonapp;
```

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

```
GRANT ALL PRIVILEGES ON strattonapp.* TO 'usuario'@'localhost';
```

```
FLUSH PRIVILEGES;
```

## Modelo de la Base de Datos:



### 4.2.2 Docker

Instalar Docker y configurar los contenedores para el backend y el frontend.

### 4.2.3 Configuración de Spring Security con JWT

Configurar la seguridad en Spring Boot utilizando JWT para la autenticación y autorización de usuarios.

## 5. Manual de despliegue, docker:

Es necesario tener Docker instalado en nuestro entorno y conexión a internet.

### Construcción y despliegue del Backend.

1. Creamos un archivo DockerFile a raíz del directorio del proyecto Backend y añadimos lo siguiente:

```
# Etapa de construcción
FROM maven:3-openjdk-17 as builder
COPY src /usr/src/app/src
COPY pom.xml /usr/src/app
RUN mvn -f /usr/src/app/pom.xml clean package -DskipTests

# Etapa de producción
FROM openjdk:17-alpine

# Variables de entorno
ENV BBDD_HOST="db"
ENV BBDD_NAME="strattonapp"
ENV APP_PORT=8080
ENV LOG_LEVEL="INFO"
ENV DLL_AUTO="update"

# Volumen para datos temporales y archivos multimedia
VOLUME /tmp
VOLUME /mediafiles
RUN mkdir -p /mediafiles

# Puerto expuesto
EXPOSE 8081

# Copiar el archivo JAR construido de la etapa de construcción
COPY --from=builder /usr/src/app/target/StrattonApp-Backend-0.0.1-SNAPSHOT.jar /app/app.jar

# Establecer directorio de trabajo

# Comando de inicio
ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```

2. A continuación creamos el archivo docker-compose.yml pero esta vez en la raíz principal del proyecto y añadimos lo siguiente:

```
Docker-compose.yml
1  version: '3.1'
2  services:
3    db:
4      image: mysql:latest
5      container_name: strattonapp-bbdd
6      restart: unless-stopped
7      ports:
8        - "3306:3306"
9      environment:
10       MYSQL_ROOT_PASSWORD: root
11       MYSQL_DATABASE: strattonapp
12      volumes:
13        - mysql_data:/data/db
14
15    strattonapp_back:
16      build:
17        context: ./Backend
18        dockerfile: ./Dockerfile
19      container_name: strattonapp_back
20      restart: unless-stopped
21      ports:
22        - "8080:8080"
23      environment:
24        BBDD_HOST: db
25        BBDD_NAME: strattonapp
26        DATABASE_USERNAME: root
27        DATABASE_PASSWORD: root
28        APP_PORT: 8080
29        LOG_LEVEL: DEBUG
30        DLL_AUTO: create
31      volumes:
32        - ./src/main/resources:/app/resources
33      depends_on:
34        - db
35
36    strattonapp_front:
37      build:
38        context: ./Frontend
39        dockerfile: ./Dockerfile
40      container_name: strattonapp_front
41      restart: unless-stopped
42      ports:
43        - "80:80"
44      depends_on:
45        - strattonapp_back
46
47  volumes:
48    mysql_data:
```

## Construcción y despliegue del Frontend.

3. Crearemos el archivo DockerFile para el Front en el directorio del mismo y agregamos lo siguiente:

```
Frontend > DockerFile > ...
1  # Imagen
2  FROM node:20.9.0 as build
3  # Directorio
4  WORKDIR /app
5  # Copia los archivos menos el dockerfile
6  COPY . .
7  # Instala las dependencias
8  RUN npm install
9  # Construye la aplicación Angular
10 RUN npm run build --prod
11 # Usa una imagen de Nginx para servir la aplicación Angular
12 FROM nginx:stable-alpine
13 # Copia los archivos
14 COPY --from=build /app/dist/strattonapp-front/browser /usr/share/nginx/html
15
16 EXPOSE 4200
17 # Comando para iniciar Nginx
18 CMD ["nginx", "-g", "daemon off;"]
```

4. Ahora creamos un archivo llamado nginx.conf en el mismo directorio que el DockerFile anterior, agregamos lo siguiente:

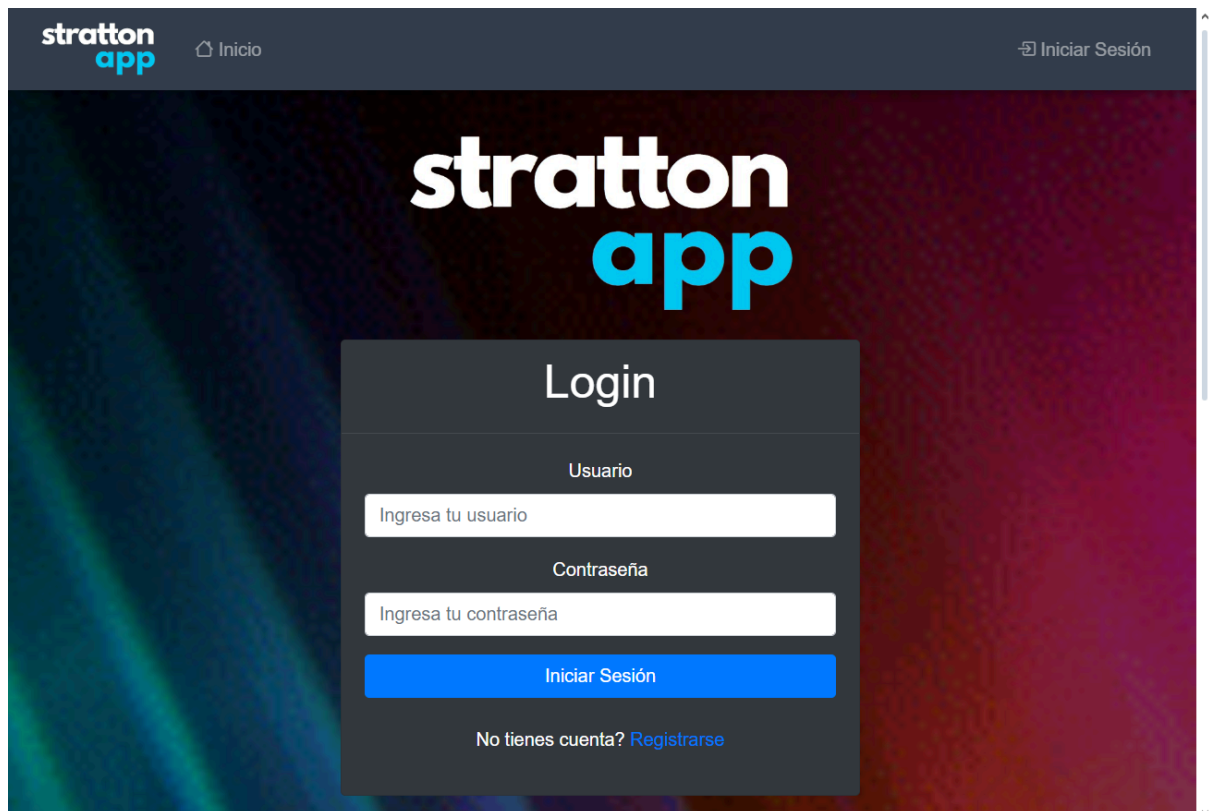
```
Frontend > nginx.conf
2  worker_processes 1;
3  events {
4      worker_connections 1024;
5  }
6
7  http {
8      include mime.types;
9      default_type application/octet-stream;
10
11     # Bloque del servidor para la aplicación Angular
12     server {
13         listen 80;
14         server_name localhost;
15
16         location / {
17             root /usr/share/nginx/html;
18             index index.html index.htm;
19             try_files $uri $uri/ /index.html;
20         }
21     }
22 }
```

## Ejecutamos la aplicación

5. Nos situamos en el directorio raíz del proyecto `cd ..` y ejecutamos el comando para iniciar los contenedores de docker `docker-compose -up -d`.

```
[+] Running 11/11
✓ db Pulled
✓ 07bc88e18c4a Pull complete
✓ 1a9c1668bf49 Pull complete
✓ 1021dda8eefc Pull complete
✓ fb61b56acac1 Pull complete
✓ 0bca83908a5b Pull complete
✓ 165e8b3d37ca Pull complete
✓ 3e1b086f1295 Pull complete
✓ dba651668484 Pull complete
✓ ed90f5355e12 Pull complete
✓ 0412f59ab2b5 Pull complete
[+] Building 0.1s (1/1) FINISHED
=> [strattonapp_back internal] load build definition from Dockerfile
=> => transferring dockerfile: 20
19.6s
7.2s
7.3s
7.3s
7.6s
7.6s
7.6s
11.7s
11.7s
17.2s
17.2s
docker:default
0.1s
0.0s
```

6. abrimos la aplicación.



## 6. Conclusión

Por fin llega la solución a todos los problemas tanto de las compañías eléctricas como para aquellos asesores energéticos que buscan la simplicidad y el trabajo bien hecho, son muchos años y muchas personas han ensuciado el nombre de los asesores, el objetivo es unificar tanto a los clientes como a los empleados en un entorno de transparencia y sencillez, garantizando la seguridad y la robustez de los datos y el mejor servicio posible.

## 7. Referencias

### 7.1 Recursos

#### 7.1.1 Aplicación

Código fuente del proyecto disponible en [GitHub](#).

#### 7.1.2 Librería compartida (Shared Library)

Repositorio de la librería compartida en GitHub.

#### 7.1.3 Presentación

Diapositivas de la presentación del proyecto disponibles en Google Slides.

### 7.2 Bibliografía

1. Documentación oficial de Spring Boot: <https://spring.io/projects/spring-boot>
2. Documentación oficial de Angular: <https://angular.io/docs>
3. Documentación oficial de Docker: <https://docs.docker.com/>
4. Documentación oficial de MySQL: <https://dev.mysql.com/doc/>
5. Tutoriales y guías de Git: <https://git-scm.com/doc>

