**Documentación Técnica del Backend**

**Tecnologías Utilizadas**

* **Lenguaje: Java 17**
* **Framework: Spring Boot 3.4.4**
* **ORM: Spring Data JPA**
* **Base de Datos: MySQL 8**
* **Autenticación y Autorización: JWT (JSON Web Token) + Spring Security**
* **Control de Acceso: Roles gestionados con @PreAuthorize**
* **Gestor de dependencias: Maven**

**Arquitectura General**

**El backend está estructurado en capas siguiendo la arquitectura en capas habitual de Spring Boot:**

* **Controladores: Definen los endpoints REST**
* **Servicios: Contienen la lógica de negocio**
* **Repositorios: Interactúan con la base de datos**
* **DTOs: Separan las entidades del modelo de los datos expuestos**
* **Modelos: Representan las entidades de la base de datos**

**Entidades Principales**

* **Usuario: contiene datos personales y credenciales.**
* **Empresa: información de empresas registradas.**
* **Vacante: contiene información de ofertas laborales.**
* **Solicitud: solicitudes realizadas por usuarios a vacantes.**
* **Perfil: define los roles (ADMIN, USUARIO, EMPRESA).**
* **UsuarioPerfil: asocia usuarios con uno o varios perfiles.**
* **Categoria: categorías de empleo.**

**Endpoints Destacados (ejemplos)**

**Autenticación**

* **POST /api/auth/login: login y generación de token JWT**
* **POST /api/auth/register: registro de nuevo usuario**

**Vacantes**

* **GET /api/vacantes: (ADMIN) listar todas las vacantes**
* **GET /api/vacantes/creadas: (USUARIO) ver vacantes activas**
* **POST /api/vacantes: (EMPRESA) crear nueva vacante**

**Solicitudes**

* **POST /api/solicitudes: (USUARIO) postular a vacante**
* **GET /api/solicitudes/usuario/{username}: (USUARIO) ver sus solicitudes**
* **PUT /api/solicitudes/{id}/adjudicar: (EMPRESA) adjudicar una vacante**

**Seguridad con Roles**

**Se usa @PreAuthorize para limitar el acceso según el rol JWT:**

**@PreAuthorize("hasRole('ADMIN')")**

**@PreAuthorize("hasRole('USUARIO')")**

**@PreAuthorize("hasRole('EMPRESA')")**

**Autenticación JWT**

* **Se genera un JWT en el login**
* **El filtro JwtFilter intercepta las peticiones y valida el token**
* **El SecurityContextHolder se configura con los roles del usuario**

**🛠 Guía de Despliegue del Backend**

**1. Requisitos Previos**

**Antes de iniciar el despliegue, asegúrate de contar con:**

* **✅ Java 17 instalado en tu equipo.**
* **✅ MySQL 8 en funcionamiento con acceso a crear bases de datos y tablas.**
* **✅ Editor o IDE (recomendado: IntelliJ IDEA o Spring Tool Suite).**
* **✅ Maven instalado y correctamente configurado.**
* **✅ Script SQL con estructura de tablas e inserciones básicas (reto9\_db.sql).**

**2. Configurar la Base de Datos**

1. **Crea la base de datos:**

**CREATE DATABASE reto9\_db CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;**

1. **Ejecuta el script SQL proporcionado con las tablas, relaciones, perfiles e inserciones iniciales.**

**3. Configurar el archivo application.properties**

**Edita el archivo src/main/resources/application.properties con tus credenciales locales de MySQL:**

**spring.datasource.url=jdbc:mysql://localhost:3306/reto9\_db?useSSL=false&serverTimezone=UTC**

**spring.datasource.username=tu\_usuario\_mysql**

**spring.datasource.password=tu\_contraseña\_mysql**

**4. Compilar y Ejecutar el Proyecto**

**Opción 1: Ejecutar desde consola (desarrollo)**

**mvn spring-boot:run**

**Opción 2: Generar .jar y ejecutar (producción)**

**mvn clean package**

**java -jar target/Reto9-0.0.1-SNAPSHOT.jar**

**5. Verificación del Funcionamiento**

* **El servidor se ejecuta por defecto en http://localhost:8080.**
* **Puedes probar autenticación en:**
  + **POST /api/auth/register**
  + **POST /api/auth/login**
* **Accede a otros endpoints protegidos usando el token JWT en el header:**

**Authorization: Bearer <tu\_token>**

**6. Probar con Postman (recomendado)**

1. **Haz login (/api/auth/login) y copia el token recibido.**
2. **En cada petición autenticada, añade en "Headers":**

**Key: Authorization**

**Value: Bearer <token>**

1. **Prueba los endpoints de /api/usuarios, /api/vacantes, /api/solicitudes, etc.**

**Notas Finales**

* **Toda la seguridad se realiza en el backend, usando roles.**
* **El backend puede ser consumido tanto por la app Android como por la interfaz Swing.**
* **Se puede probar con Postman o Swagger (si se habilita).**

**Anexos**

* **Script de base de datos (estructura + inserciones)**
* **Proyecto backend en ZIP o GitHub**
* **.jar ejecutable para pruebas**