FUNDACIÓN KINAL

Centro Educativo Técnico Laboral Kinal

Materia: 5to Perito Informática

Profesor: Eduardo Hor

Sección: IN5CM

Manual de Procedimiento

Nombre: Josué Daniel Jolón Motta

Carné: 2022205

Fecha de Entrega:

06/02/2026

Documentación de Procedimiento

Sistema Repuestos Automotrices – API REST con Spring Boot

1. Descripción general del proyecto

En este proyecto desarrollé una API REST utilizando Spring Boot desde IntelliJ IDEA.
El objetivo principal fue crear un sistema que permitiera administrar empleados dentro de un sistema de repuestos automotrices mediante operaciones CRUD.

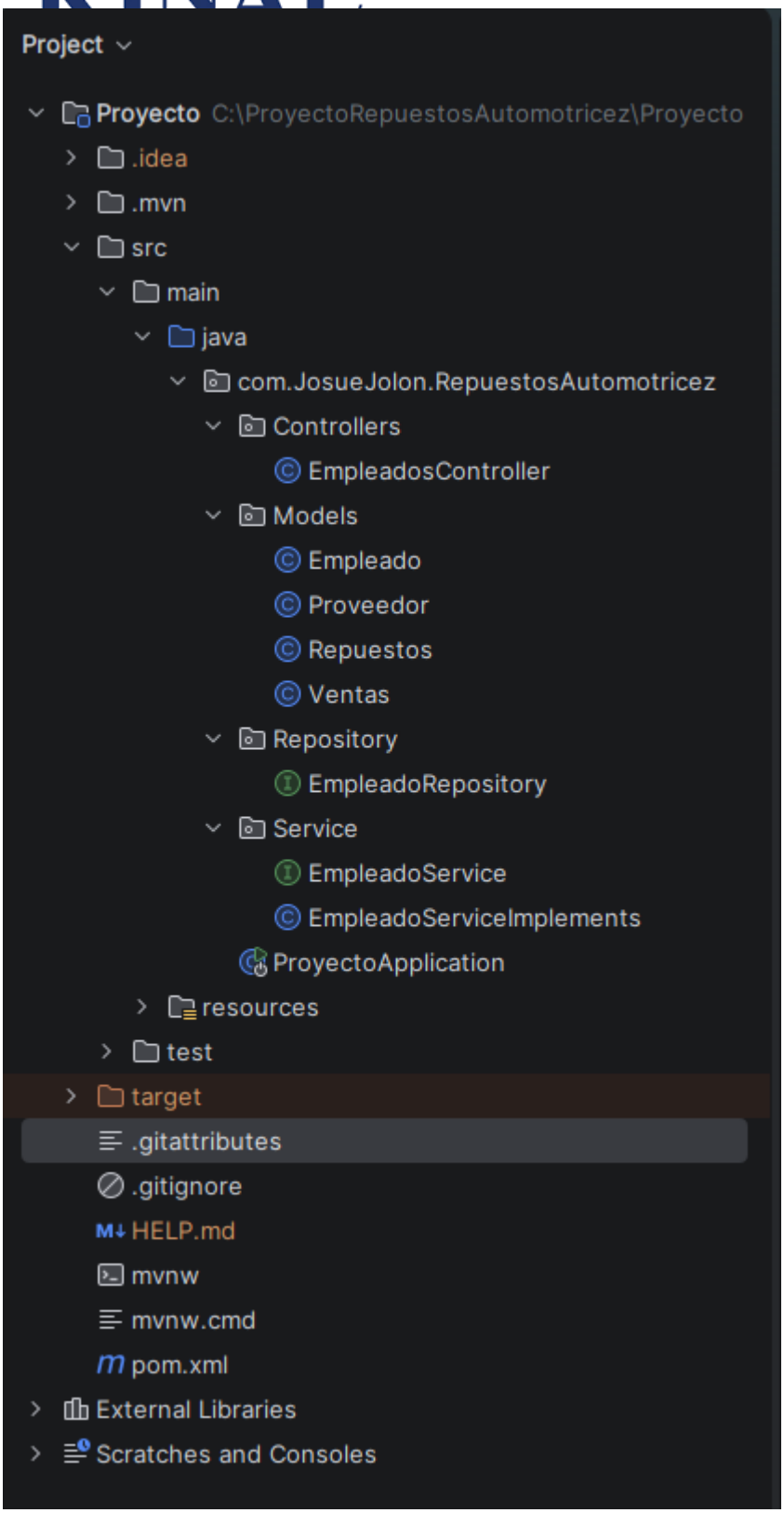
El proyecto permite:

- Registrar empleados
- Mostrar todos los empleados
- Buscar empleados por ID
- Actualizarlos
- Eliminarlos

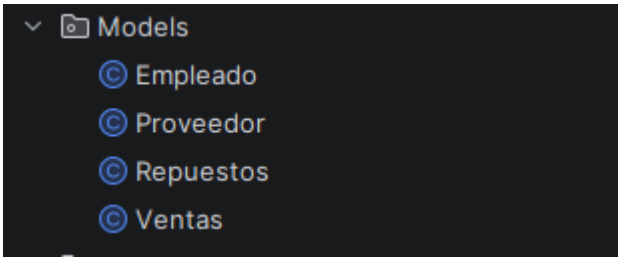
Para lograr esto utilicé una estructura basada en capas, separando responsabilidades para mantener el código ordenado y fácil de entender.

2. Estructura del Proyecto

El proyecto se organizó en paquetes principales para dividir el trabajo:


Project
Proyecto C:\ProyectoRepuestosAutomotricez\Proyecto
.idea
.mvn
src
main
java
com.JosueJolon.RepuestosAutomotricez
Controllers
EmpleadosController
Models
Empleado
Proveedor
Repuestos
Ventas
Repository
EmpleadoRepository
Service
EmpleadoService
EmpleadoServiceImplements
ProyectoApplication
resources
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles

Models:



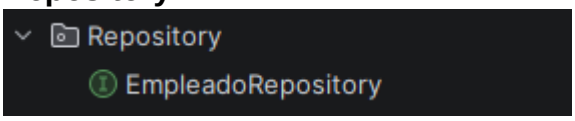
Aquí definí las clases que representan las tablas de la base de datos.

En esta parte:

- Creé la entidad **Empleado, Proveedor, Repuestos, Ventas**
- Definí sus atributos (id, nombre, apellido, etc.)
- Usé anotaciones para indicar que es una entidad de base de datos
- Definí los getters y setters para acceder a la información

El objetivo fue representar cómo se guarda la información dentro de la base de datos.

Repository:



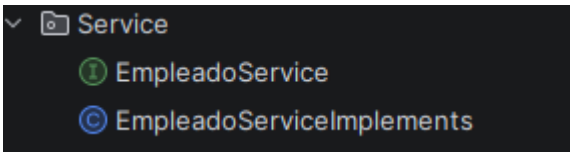
En esta capa creé las interfaces que se encargan de la conexión con la base de datos.

Aquí:

- Extendí JpaRepository
- Permití que Spring Boot genere automáticamente operaciones CRUD
- Evité escribir consultas SQL manualmente

Esta parte funciona como el puente entre el programa y la base de datos.

Service:



En esta capa desarrollé la lógica del sistema.

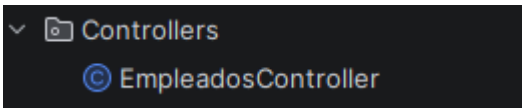
Aquí hice:

- Métodos para obtener empleados
- Métodos para buscar por ID
- Métodos para guardar nuevos registros
- Métodos para actualizar datos
- Métodos para eliminar empleados

En el empleado service definimos los métodos que vamos a utilizar, en este caso seria el CRUD

Con el buscarPorId y en el Implements va la lógica para mandar a llamar en los controllers

Controllers:



En esta parte creé los endpoints REST para que el usuario pueda interactuar con la API.

Utilicé:

- @RestController
- @RequestMapping
- @GetMapping
- @PostMapping
- @PutMapping
- @DeleteMapping

Cada método responde a una petición HTTP diferente.

Funciones principales:

- Listar empleados
- Buscar por ID
- Crear empleados

- Actualizar empleados
- Eliminar empleados

3. Funcionamiento General del Sistema

El flujo del sistema funciona así:

1. El usuario envía una petición desde Postman o navegador
2. El Controller recibe la solicitud
3. El Controller llama al Service
4. El Service ejecuta la lógica
5. El Repository accede a la base de datos
6. Se devuelve una respuesta al usuario

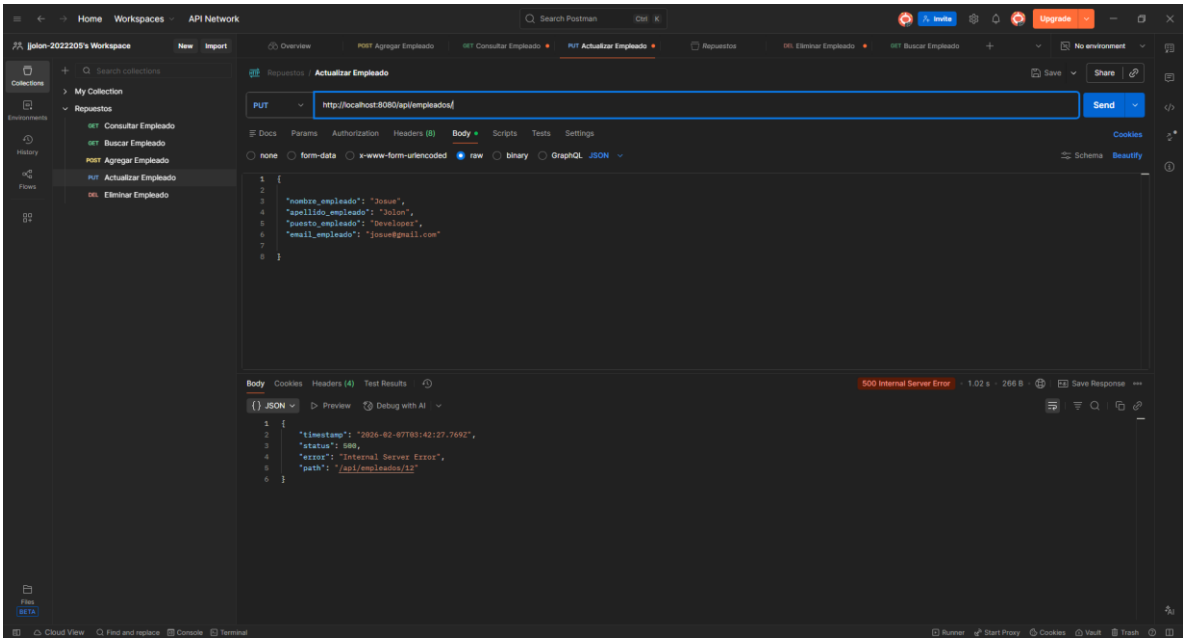
Esto permite tener un sistema organizado y fácil de mantener.

4. Base de Datos

La configuración con la base de datos nos la proporciono el profesor, solo era de cambiar el usuario y contraseña para trabajar en nuestra casa

La información de empleados se guarda automáticamente gracias a JPA.

5. Pruebas del Sistema



Para probar el funcionamiento utilicé:

- Postman
-
- Realicé pruebas de:

- GET → listar registros
- GET por ID
- POST → crear
- PUT → actualizar
- DELETE → eliminar

Esto permitió verificar que cada endpoint funcionara correctamente.

6. Aprendizajes durante el desarrollo

Durante la realización del proyecto aprendí:

- Cómo estructurar proyectos Spring Boot
- Cómo crear APIs REST
- Cómo usar anotaciones de Spring
- Cómo conectar con MySQL
- Cómo separar lógica por capas

7. Conclusión del procedimiento

El desarrollo del sistema permitió implementar un CRUD completo utilizando Spring Boot.
Se aplicaron buenas prácticas como la separación por capas, uso de repositorios JPA y controladores REST.
El sistema quedó funcional y listo para ser probado mediante herramientas externas.

Link del repositorio:

https://github.com/josuejolon-2022205/ProyectoRepuestosAutomotricez.git