**2. Develop Spring Boot Backend Application**

## **2.1: Create a Spring Boot Application**

You can refer to the below address to create a Spring Boot application.

https://start.spring.io/



**2.2:** Configure application.properties to connect to your MySQL database. Add the following content to the application.properties file:

spring.application.name=BackendApp

spring.datasource.url=jdbc:postgresql://localhost:5432/emploi

spring.datasource.username=postgres

spring.datasource.password=1234

spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

**2.3:** You need to create the database ***emploi*** in PostgreSQL server from the **PgAdmin**

In the folder **demo** you should create the **4** folders***: model, repository, controller and exception***

## **2.4. Create JPA Entity - Employee.java**

Create the *Employee* class inside the *model* package with the following contents –

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

 @Id

 @GeneratedValue(strategy = GenerationType.IDENTITY)

 private long id;

 @Column(name = "first\_name")

 private String firstName;

 @Column(name = "last\_name")

 private String lastName;

 @Column(name = "email\_id")

 private String emailId;

 public Employee() {

 }

 public Employee(String firstName, String lastName, String emailId) {

 super();

 this.firstName = firstName;

 this.lastName = lastName;

 this.emailId = emailId;

 }

 public long getId() {

 return id;

 }

 public void setId(long id) {

 this.id = id;

 }

 public String getFirstName() {

 return firstName;

 }

 public void setFirstName(String firstName) {

 this.firstName = firstName;

 }

 public String getLastName() {

 return lastName;

 }

 public void setLastName(String lastName) {

 this.lastName = lastName;

 }

 public String getEmailId() {

 return emailId;

 }

 public void setEmailId(String emailId) {

 this.emailId = emailId;

 }}

## **2.5. Create a Spring Data Repository - EmployeeRepository.java**

Create the following interface inside the *repository* package –

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import net.javaguides.springboot.model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long>{

}

## **2.6. Create Custom Exception**

create an ResourceNotFoundException class and add the following code to it:

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT\_FOUND)

public class ResourceNotFoundException extends RuntimeException{

 private static final long serialVersionUID = 1L;

 public ResourceNotFoundException(String message) {

 super(message);

 }

}

## **2.7. Create Spring Rest Controller - EmployeeController.java**

Create an EmployeeController class and add the following code to it:

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.exception.ResourceNotFoundException;

import com.example.demo.model.Employee;

import com.example.demo.repository.EmployeeRepository;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/api/v1/")

public class EmployeeController {

 @Autowired

 private EmployeeRepository employeeRepository;

 // get all employees

 @GetMapping("/employees")

 public List<Employee> getAllEmployees(){

 return employeeRepository.findAll();

 }

 // create employee rest api

 @PostMapping("/employees")

 public Employee createEmployee(@RequestBody Employee employee) {

 return employeeRepository.save(employee);

 }

 // get employee by id rest api

 @GetMapping("/employees/{id}")

 public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

 Employee employee = employeeRepository.findById(id)

 .orElseThrow(() -> new ResourceNotFoundException("Employee not exist with id :" + id));

 return ResponseEntity.ok(employee);

 }

 // update employee rest api

 @PutMapping("/employees/{id}")

 public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails){

 Employee employee = employeeRepository.findById(id)

 .orElseThrow(() -> new ResourceNotFoundException("Employee not exist with id :" + id));

 employee.setFirstName(employeeDetails.getFirstName());

 employee.setLastName(employeeDetails.getLastName());

 employee.setEmailId(employeeDetails.getEmailId());

 Employee updatedEmployee = employeeRepository.save(employee);

 return ResponseEntity.ok(updatedEmployee);

 }

 // delete employee rest api

 @DeleteMapping("/employees/{id}")

 public ResponseEntity<Map<String, Boolean>> deleteEmployee(@PathVariable Long id){

 Employee employee = employeeRepository.findById(id)

 .orElseThrow(() -> new ResourceNotFoundException("Employee not exist with id :" + id));

 employeeRepository.delete(employee);

 Map<String, Boolean> response = new HashMap<>();

 response.put("deleted", Boolean.TRUE);

return ResponseEntity.ok(response); }}

## **2.8. Running Application**

Or you can start the spring boot application via the command line using **mvn spring-boot:run** command

## **2.9. Testing REST APIs using Postman**



Use below Rest endpoints to test CRUD Rest APIs and in Angular application.

### **Get All Employees:**

HTTP Method: GET

http://localhost:8080/api/v1/employees

### **Get Employee By Id:**

HTTP Method GET

http://localhost:8080/api/v1/employees/{employeeId}

### **Create Employee:**

HTTP Method - POST

http://localhost:8080/api/v1/employees

### **Update Employee**

HTTP Method - POST

http://localhost:8080/api/v1/employees/{employeeId}

### **Delete Employee By Id:**

HTTP Method - DELETE

http://localhost:8080/api/v1/employees/{employeeId}

This completes the development of Spring boot CRUD Rest APIs.



