EmployeeController.java

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

// Create a new employee

@PostMapping

public ResponseEntity<Employee> createEmployee(@RequestBody Employee employee) {

Employee savedEmployee = employeeRepository.save(employee);

return new ResponseEntity<>(savedEmployee, HttpStatus.CREATED);

}

// Get all employees

@GetMapping

public ResponseEntity<List<Employee>> getAllEmployees() {

List<Employee> employees = employeeRepository.findAll();

return new ResponseEntity<>(employees, HttpStatus.OK);

}

// Get an employee by ID

@GetMapping("/{id}")

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

Optional<Employee> employee = employeeRepository.findById(id);

return employee.map(value -> new ResponseEntity<>(value, HttpStatus.OK))

.orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

// Update an employee

@PutMapping("/{id}")

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

Optional<Employee> employee = employeeRepository.findById(id);

if (employee.isPresent()) {

Employee existingEmployee = employee.get();

existingEmployee.setName(employeeDetails.getName());

existingEmployee.setEmail(employeeDetails.getEmail());

existingEmployee.setDepartment(employeeDetails.getDepartment());

Employee updatedEmployee = employeeRepository.save(existingEmployee);

return new ResponseEntity<>(updatedEmployee, HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

// Delete an employee

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {

if (employeeRepository.existsById(id)) {

employeeRepository.deleteById(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

}

DepartmentController.java

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Department;

import com.example.employeemanagementsystem.repository.DepartmentRepository;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

// Create a new department

@PostMapping

public ResponseEntity<Department> createDepartment(@RequestBody Department department) {

Department savedDepartment = departmentRepository.save(department);

return new ResponseEntity<>(savedDepartment, HttpStatus.CREATED);

}

// Get all departments

@GetMapping

public ResponseEntity<List<Department>> getAllDepartments() {

List<Department> departments = departmentRepository.findAll();

return new ResponseEntity<>(departments, HttpStatus.OK);

}

// Get a department by ID

@GetMapping("/{id}")

public ResponseEntity<Department> getDepartmentById(@PathVariable Long id) {

Optional<Department> department = departmentRepository.findById(id);

return department.map(value -> new ResponseEntity<>(value, HttpStatus.OK))

.orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

// Update a department

@PutMapping("/{id}")

public ResponseEntity<Department> updateDepartment(@PathVariable Long id, @RequestBody Department departmentDetails) {

Optional<Department> department = departmentRepository.findById(id);

if (department.isPresent()) {

Department existingDepartment = department.get();

existingDepartment.setName(departmentDetails.getName());

Department updatedDepartment = departmentRepository.save(existingDepartment);

return new ResponseEntity<>(updatedDepartment, HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

// Delete a department

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteDepartment(@PathVariable Long id) {

if (departmentRepository.existsById(id)) {

departmentRepository.deleteById(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

}