Step 1: Pagination

We'll implement pagination for the employee list using Page and Pageable.

EmployeeService.java:

package com.example.employeemanagementsystem.service;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.repository.EmployeeRepository;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Sort;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

private final EmployeeRepository employeeRepository;

@Autowired

public EmployeeService(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

public Page<Employee> getEmployees(int page, int size, String sortBy) {

return employeeRepository.findAll(PageRequest.of(page, size, Sort.by(sortBy)));

}

}

Step 2: Sorting

We'll add sorting functionality to our queries and combine pagination and sorting in our search endpoint.

EmployeeController.java:

package com.example.employeemanagementsystem.controller;

import com.example.employeemanagementsystem.entity.Employee;

import com.example.employeemanagementsystem.service.EmployeeService;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Sort;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

private final EmployeeService employeeService;

@Autowired

public EmployeeController(EmployeeService employeeService) {

this.employeeService = employeeService;

}

@GetMapping

public ResponseEntity<Page<Employee>> getEmployees(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size,

@RequestParam(defaultValue = "name") String sortBy) {

return new ResponseEntity<>(employeeService.getEmployees(page, size, sortBy), HttpStatus.OK);

}

// ...

}

Now, when you call the /api/employees endpoint, you can add query parameters page, size, and sortBy to paginate, limit the number of results, and sort the results, respectively.