**Exercise 5: Employee Management System - Defining Query Methods**

Business Scenario:

Enhance your repository to support custom queries.

**1.Introduction**

The Employee Management System is a Spring Boot application designed to manage employees and departments within an organization. The system allows for the creation, retrieval, updating, and deletion of employee and department records.

**2. Key Components**

**1. Entities:**

* **Employee**: Represents an individual working within a department.
  + Attributes:
    - id: Unique identifier for the employee.
    - name: Name of the employee.
    - email: Email address of the employee.
    - department: The department to which the employee belongs (many-to-one relationship).
* **Department**: Represents a department within the organization.
  + Attributes:
    - id: Unique identifier for the department.
    - name: Name of the department.
    - employees: A set of employees that belong to the department (one-to-many relationship).

**2. Repositories:**

* **DepartmentRepository**: Interface extending JpaRepository for CRUD operations and custom queries related to the Department entity.
* **EmployeeRepository**: Interface extending JpaRepository for CRUD operations and custom queries related to the Employee entity, including methods to find employees by name and email.

**3. Controllers:**

* **DepartmentController**: Manages HTTP requests related to Department entities.
  + Endpoints:
    - GET /departments: Retrieve all departments.
    - POST /departments: Create a new department.
    - GET /departments/{id}: Retrieve a department by ID.
    - PUT /departments/{id}: Update a department by ID.
    - DELETE /departments/{id}: Delete a department by ID.
* **EmployeeController**: Manages HTTP requests related to Employee entities.
  + Endpoints:
    - GET /employees: Retrieve all employees.
    - POST /employees: Create a new employee.
    - GET /employees/{id}: Retrieve an employee by ID.
    - PUT /employees/{id}: Update an employee by ID.
    - DELETE /employees/{id}: Delete an employee by ID.

**5. Configuration**

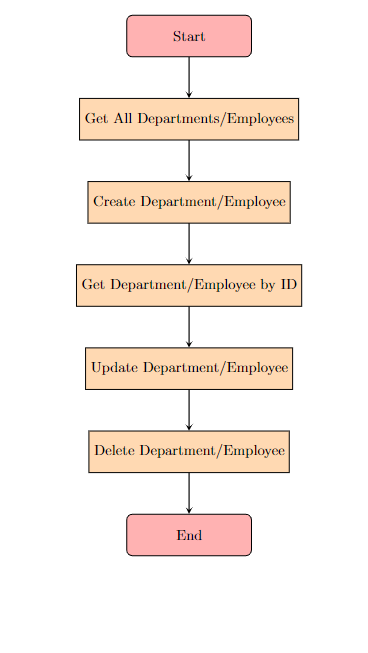
**Application Properties:**

* Configures an in-memory H2 database.

**Dependencies:**

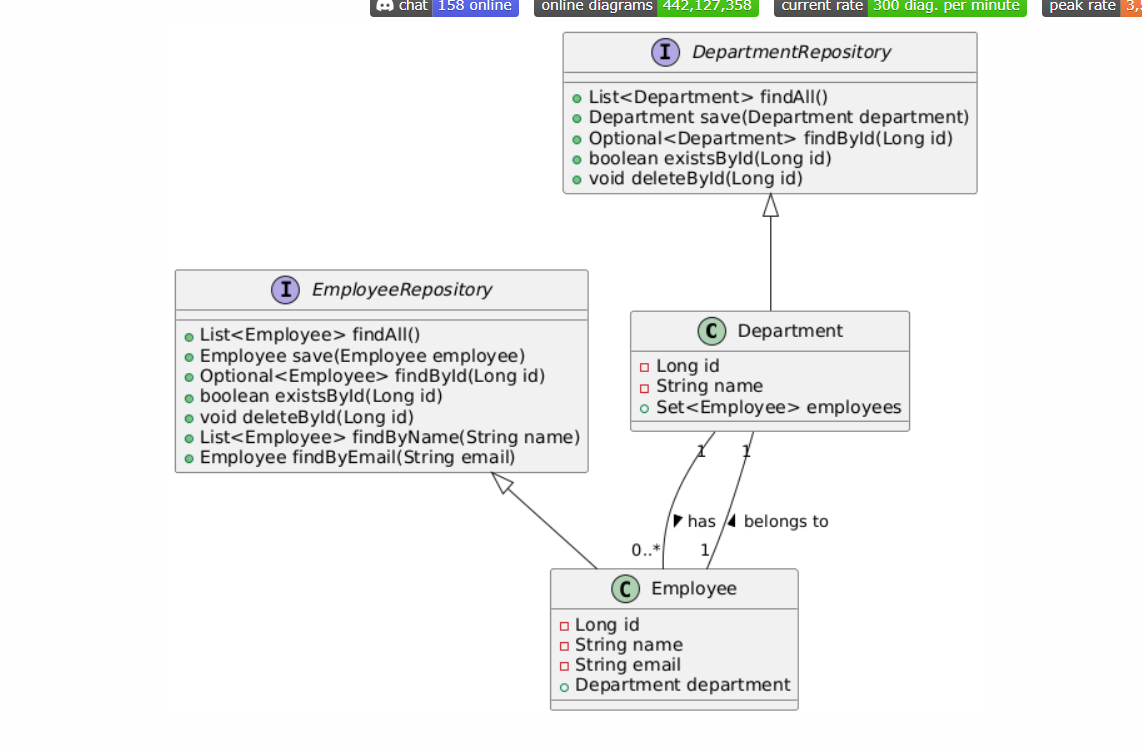
* Spring Boot Starter Data JPA
* Spring Boot Starter Web
* H2 Database
* Lombok

**FLOWCHART :**



**Explanation:**

1. **Start**: The initial point of the flowchart.
2. **Get All Departments/Employees**: Represents the action to retrieve all records from the database.
3. **Create Department/Employee**: Represents the action to create a new record in the database.
4. **Get Department/Employee by ID**: Represents the action to retrieve a specific record based on its ID.
5. **Update Department/Employee**: Represents the action to update an existing record in the database.
6. **Delete Department/Employee**: Represents the action to delete a record from the database.
7. **End**: The endpoint of the flowchart.

**CLASS DIAGRAM :**

**Explanation:**

1. **Department Class**:
   * id: Identifier for the department.
   * name: Name of the department.
   * employees: A set of employees that belong to the department (one-to-many relationship).
2. **Employee Class**:
   * id: Identifier for the employee.
   * name: Name of the employee.
   * email: Email address of the employee.
   * department: The department to which the employee belongs (many-to-one relationship).
3. **DepartmentRepository Interface**:
   * Provides methods for CRUD operations and querying Department entities.
4. **EmployeeRepository Interface**:
   * Provides methods for CRUD operations, querying by name, and finding by email for Employee entities.
5. **Relationships**:
   * A Department has many Employees.An employee belongs to one department.