Домашнее задание: реализовать приложение учета сотрудников на Spring Boot с использованием JPA:

1. **Описание системы:**
    - **Название системы:** Учет сотрудников.
    - **Цель системы:** Разработка веб-приложения для учета сотрудников с использованием технологий Spring Boot и JPA.
    - **Технологии:** Spring Boot, Spring Data JPA, любая база данных по выбору.
2. **Функциональные требования:**

2.1 **Сущность "Employee":**

- **Поля:**
    - **id** (Long) — уникальный идентификатор сотрудника.
    - **firstName** (String) — имя сотрудника.
    - **lastName** (String) — фамилия сотрудника.
    - **position** (String) — должность сотрудника.
    - **department** (String) — отдел, в котором работает сотрудник.
    - **salary** (BigDecimal) — заработная плата сотрудника.

3. **API для управления данными сотрудников:**

3.1 **Получение одного сотрудника по идентификатору:**

- **URL:** /api/employees/{id}
- **HTTP-метод:** GET
- **Описание:** Получение информации о сотруднике по его уникальному идентификатору.
- **Параметры пути:**
    - **id** (Long): Уникальный идентификатор сотрудника.
- **Ответ:**
    - Код HTTP-ответа 200 (OK) и объект Employee в формате JSON.

- Код HTTP-ответа 404 (Not Found), если сотрудник с указанным ID не найден.

3.2 **Получение списка всех сотрудников:**

- **URL:** /api/employees
- **HTTP-метод:** GET
- **Описание:** Получение списка всех сотрудников в формате JSON.
- **Ответ:**
  - Код HTTP-ответа 200 (OK) и массив объектов Employee в формате JSON.

3.3 **Создание нового сотрудника:**

- **URL:** /api/employees
- **HTTP-метод:** POST
- **Описание:** Создание нового сотрудника.
- **Запрос:**
  - Объект Employee в формате JSON с заполненными полями firstName, lastName, position, department и salary.
- **Ответ:**
  - Код HTTP-ответа 201 (Created) и объект Employee в формате JSON с информацией о созданном сотруднике.

3.4 **Обновление данных существующего сотрудника:**

- **URL:** /api/employees
- **HTTP-метод:** PUT
- **Описание:** Обновление данных существующего сотрудника.
- **Запрос:**
  - Объект Employee в формате JSON с заполненными полями id, firstName, lastName, position, department и salary.
- **Ответ:**
  - Код HTTP-ответа 200 (OK) и объект Employee в формате JSON с информацией об обновленном сотруднике.

3.5 **Удаление сотрудника по идентификатору:**

- **URL:** /api/employees/{id}
- **HTTP-метод:** DELETE
- **Описание:** Удаление сотрудника по его уникальному идентификатору.
- **Параметры пути:**
  - **id** (Long): Уникальный идентификатор сотрудника.
- **Ответ:**
  - Код HTTP-ответа 204 (No Content) в случае успешного удаления сотрудника.
  - Код HTTP-ответа 404 (Not Found), если сотрудник с указанным ID не найден.

4. **База данных:**
   - Использование Spring Data JPA для взаимодействия с базой данных.
5. **Тестирование:**
   - Проведение тестирования вручную с использованием Postman для каждого из описанных выше эндпоинтов.
6. **Дополнительные требования:**
   - Реализация обработки ошибок с возвратом соответствующих HTTP-кодов и сообщений об ошибках в формате JSON.
   - Реализация логирования событий в приложении.
7. **Среда выполнения:**
   - Java 8 и выше.
   - Apache Maven для сборки и управления зависимостями.
   - Postman для тестирования API.
8. **План работ:**

- Настройка проекта Spring Boot в Spring Initializr.
- Создание сущности Employee.

- Разработка контроллера с методами для выполнения операций CRUD.
- Интеграция с базой данных.
- Тестирование вручную.

Проект с занятия:

https://drive.google.com/file/d/1rODTKz0sV2UjsYR5wJlvUCM1vWtg-x8z/view?usp=sharing

Ссылка на итоговый проект (экзаменационное задание):

https://drive.google.com/file/d/1aOUID49sxvAVbC1vFDAuuNAst0Ltx6rl/view?usp=sharing