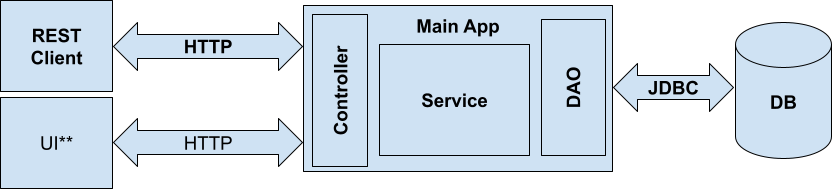
**Тестовое задание [Java Mastery]:**

У нас есть база сотрудников (**employeedb**). Нам необходимо добавлять новых сотрудников, удалять уволенных, обновлять данные о сотрудниках, а также должна быть возможность просмотреть весь список сотрудников сразу и отдельно получить каждого сотрудника. Для этого требуется разработать веб-приложение (Main App), представляющее из себя **REST** сервис, которое должно позволять выполнять CRUD операции над сотрудниками.

Архитектура приложения должна быть следующей:



**Main App** - самостоятельное приложение предоставляющее **REST API**(это обязательно должен быть REST сервис) и позволяющее выполнять CRUD операции над сотрудниками(employee). Приложение состоит из трех основных элементов(слоев): **Controller** - это REST контроллер, позволяющий отобразить HTTP запросы на методы обрабатывающие их (для этого используйте Spring Web); **Service** - это сервис выполняющий бизнес-логику, в данном задании бизнес-логики практически нет, и сервис просто делает вызов DAO (для DI используйте Spring); **DAO** - выполняет запрос в базу данных(БД) (используйте Spring JDBC). В рамках тестового задания Controller, Service и DAO - это практически три java-класса в вашем приложении. Для приложения подготовлен скелетон, который вы можете найти здесь:

<https://github.com/java-mastery-brest/simplewebapp> ..

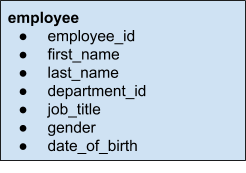
Полезно ознакомиться:

<https://spring.io/guides/gs/rest-service/>

<https://restfulapi.net/>

В реализации Main App вам следует использовать Java 8, Maven, Spring JDBC, Spring Boot (Spring Boot может быть заменен на Spring с развертыванием приложения на Tomcat)..

**DB** - база данных. Используя PostgreSQL, создайте БД **‘employeedb’**. В базе данных создайте таблицу **‘employee’** и добавьте в нее несколько сотрудников, для этого напишите SQL скрипт. Таблицу **‘employee’** требуется создать в соответствии со схемой:



Для проверки работы вашего приложения вам понадобится, либо **Rest Client**, либо разработать свой **UI\*** - написать несколько своих страниц, которые будут вызывать **REST API**.

**Rest Client** - можно использовать Postman или любой другой REST клиент, из которого вы будете делать HTTP запросы к своему приложению.

**Unit Tests\*** - не обязательная часть задания, но будет большим плюсом и очень полезно, если вы напишете парочку модульных тестов используя JUnit/Mockito (в первую очередь такой тест стоит написать для сервиса).

При разработке можете использовать удобную для вас IDE (IntelliJ Idea, Eclipse и т.п.).

**UI\*\*** - не обязательная часть задания, за которую следует браться, если вы в полном объеме выполнили основную часть задания, написали тесты и разобрались, как это все работает. Для этого достаточно написать несколько страниц, используя HTML и JS(JQuery). Для выполнения HTTP запросы к своему приложению со страничек можно воспользоваться Ajax-запросами:

<http://jquery.page2page.ru/index.php5/Ajax-%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81>

Свое приложение следует выложить на <https://github.com/> .

**ПОМНИТЕ, основная цель задания - разобраться, как работает ваше приложение!!!**

1. JPA (Spring Data JPA)

2. error handling (@ControllerAdvice + custom exception)

3. Logging - попробовать разные логгеры(log4j, logback), разобраться, зачем log4j и зачем нужен slf4j

4. Используя API актуатора попробовать поменять уровень логирования налету, посмотреть, что такое spring boot actuator

5. Swagger

6. Добавить валидацию для Employee @Valid

7. JMS (ActiveMQ)

8. Liquibase/Flyway

9. Docker