Практика Java, задание №1. Разработка Java RESTful сервиса

Цели задания

- Познакомится с понятием сервис-ориентированная архитектура, web-сервис, микросервисная архитектура, микросервис
- Познакомится с архитектурным стилем взаимодействия компонентов распределенного приложения REST (от англ. Representational State Transfer — «передача состояния представления») и RESTful сервисами, отвечающими требованиям REST
- Познакомится со спецификацией “The OpenAPI Specification”, инструментарием Swagger Editor, описанием API в формате yaml
- Познакомится с понятиями java web-приложение (war), java сервлет и контейнер сервлетов
- Познакомится с фреймворком Spring (Spring Boot), широко применяющимся для реализации web-сервисов, архитектурой приложения Spring, генератором шаблона приложения “Spring Initializr” (https://start.spring.io)
- Попрактиковаться в реализации @RestCotroller, @Service, @Reprository компонентов Spring, отвечающих за разные аспекты работы RESTful сервиса
- Попрактиковаться в работе с СУБД, на примере PostgreSQL, попробовать использование MyBatis Framework для работы
- Попрактиковаться в работе с инструментарием разработчика: интегрированной средой разработки (IDE), системой управления версиями Git (GitHub)

Задание

Необходимо разработать тестовый RESTful сервис “ORDER”, отвечающий за хранение информации о заказах в Медицинской Информационной Системе (МИС). Схема базы состоит из двух таблиц:

1. ORDER, отвечающая за хранение заказов
2. ORDER_ITEM, отвечающая за хранение детализации позиций заказов

Скрипт по созданию тестовых таблиц представлен в приложении №1. Скрипт по наполнению таблиц в тестовых данных представлен в приложении №3.

Сервис должен реализовывать методы CRUD (от создание (англ. create), чтение (read), модификация (update), удаление (delete)) в соответствии с представленной в приложении №2 спецификацией API в формате OpenAPI. Рекомендуемый инструмент для чтения спецификации https://editor.swagger.io

- Используемый стек технологий: Java 11 + Spring Boot + MyBatis + PostgreSQL
- Оформление результата: Maven Project + исходный код в репозиторий на GitHub

Приложение №1. SQL скрипт для создания схемы данных

```
CREATE SEQUENCE ORDER_SEQ  START WITH 1000 INCREMENT BY 1;
CREATE SEQUENCE ORDER_ITEM_SEQ  START WITH 1000 INCREMENT BY 1;

-- ORDER
CREATE TABLE "order" (
    ID INTEGER,
    ORDER_STATUS_ID SMALLINT,
    CUSTOMER_NAME VARCHAR(64),
    CUSTOMER_PHONE VARCHAR(24),
    CUSTOMER_COMMENT VARCHAR(128));

ALTER TABLE "order" ADD CONSTRAINT ORDER_ID_PK PRIMARY KEY (ID);

CREATE INDEX ORDER_PK_IDX ON "order"(ID);

-- ORDER ITEM
CREATE TABLE ORDER_ITEM (
    ID INTEGER,
    ORDER_ID INTEGER,
    ITEM_NAME VARCHAR(64));

ALTER TABLE ORDER_ITEM ADD CONSTRAINT ORDER_ITEM_ID_PK PRIMARY KEY (ID);
ALTER TABLE ORDER_ITEM ADD CONSTRAINT OI_O_FK FOREIGN KEY (ORDER_ID)
REFERENCES "order"(ID);

CREATE INDEX ORDER_ITEM_PK_IDX ON ORDER_ITEM(ID);
CREATE INDEX ORDER_ITEM_FK_IDX ON ORDER_ITEM(ORDER_ID);
```

Приложение №2. Спецификация API в формате yaml

```
openapi: 3.0.0
info:
  title: Test Order Microservice
  version: 1.0.0
  description: Order microservice REST API v1.0
servers:
  - url: http://192.168.0.7:8080
paths:
  /order:
    post:
      summary: Request for adding a new Order
      description: Creates a new order with parameters are contained in the
request body
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Order'
            example:
              orderStatusId: 1
              customerName: "John Doe"
              customerPhone: "+7-905-345-4523"
              customerComment: "Pls. call me back"
              orderItems : [
                {
                  "itemName": "Item #1"
                },
                {
                  "itemName": "Item #2"
                },
                {
                  "itemName": "Item #3"
                }
              ]
      responses:
        '200':
          description: A new order has been successfully created (200 or 201)
        '201':
          description: A new order has been successfully created
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Order'
              example:
                id: 100001
                orderStatusId: 1
                customerName: "John Doe"
                customerPhone: "+7-905-345-4523"
                customerComment: "Pls. call me back"
```

```
              orderItems : [
                {
                  "id": 1000,
                  "orderId": 100001,
                  "itemName": "Item #1"
                },
                {
                  "id": 1001,
                  "orderId": 100001,
                  "itemName": "Item #2"
                },
                {
                  "id": 1002,
                  "orderId": 100001,
                  "itemName": "Item #3"
                }
              ]
      '500':
        description: Server error
/order/{id}:
  get:
    summary: Get the Order by id
    description: Returns object by "orderId" or returns null
    parameters:
      - in: path
        name: id
        description: Numeric ID of the Order object
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: A successful response
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Order'
            example:
              id: 100001
              orderStatusId: 1
              customerName: "John Doe"
              customerPhone: "+7-905-345-4523"
              customerComment: "Pls. call me back"
              orderItems : [
                {
                  "id": 1000,
                  "orderId": 100001,
                  "itemName": "Item #1"
                },
                {
                  "id": 1001,
                  "orderId": 100001,
                  "itemName": "Item #2"
```

```
                },
                {
                  "id": 1002,
                  "orderId": 100001,
                  "itemName": "Item #3"
                }
              ]
        '404':
          description: A resource with requested ID not found
        '500':
          description: Server error
    put:
      summary: Request for editing the Order by id
      description: Updates order by id with parameters are contained in
request body
      parameters:
        - in: path
          name: id
          description: Numeric ID of the Order which has to be updated
          required: true
          schema:
            type: integer
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Order'
            example:
              id: 100001
              orderStatusId: 1
              customerName: "John Doe"
              customerPhone: "+7-905-345-4523"
              customerComment: "call me back"
              orderItems : [
                {
                  "id": 1000,
                  "orderId": 100001,
                  "itemName": "Item #1"
                },
                {
                  "id": 1001,
                  "orderId": 100001,
                  "itemName": "Item #2"
                },
                {
                  "id": 1002,
                  "orderId": 100001,
                  "itemName": "Item #3"
                }
              ]
      responses:
        '200':
```

```
          description: Order has been updated successfully
        '500':
          description: Server error
    delete:
      summary: Request for removing the Order by id
      description: Removes order by id
      parameters:
        - in: path
          name: id
          description: Numeric ID of the Order object to be deleted
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: The Order has been deleted successfully
        '500':
          description: Server error
components:
  schemas:
    Order:
      type: object
      properties:
        id:
          type: integer
        orderStatusId:
          type: integer
        customerName:
          type: string
        customerPhone:
          type: string
        customerComment:
          type: string
        orderItems:
          type: array
          items:
            $ref: '#/components/schemas/OrderItem'
    OrderItem:
      type: object
      properties:
        id:
          type: integer
        orderId:
          type: integer
        itemName:
          type: string
```

Приложение №3. Пример SQL скрипта для генерации тестовых данных

```
DO $$
DECLARE
  v_order_id integer;
  v_order_item_id integer;
BEGIN
  SELECT NEXTVAL('order_seq') INTO v_order_id;
  INSERT INTO "order"(ID, ORDER_STATUS_ID, CUSTOMER_NAME, CUSTOMER_PHONE,
CUSTOMER_COMMENT)
    VALUES (v_order_id, 1, 'Ivanov I.I', '+7(952)-634-55-23', 'Pls. call
before delivery');

  SELECT NEXTVAL('order_item_seq') INTO v_order_item_id;
  INSERT INTO ORDER_ITEM (ID, ORDER_ID, ITEM_NAME)
    VALUES (v_order_item_id, v_order_id, 'Order Item #1');

  SELECT NEXTVAL('order_item_seq') INTO v_order_item_id;
  INSERT INTO ORDER_ITEM (ID, ORDER_ID, ITEM_NAME)
    VALUES (v_order_item_id, v_order_id, 'Order Item #2');

  SELECT NEXTVAL('order_item_seq') INTO v_order_item_id;
  INSERT INTO ORDER_ITEM (ID, ORDER_ID, ITEM_NAME)
    VALUES (v_order_item_id, v_order_id, 'Order Item #3');

END $$;
```