ДЗ 4М. Система заказов по покупку билетов

# *Цель:*

Разработать два отдельных микросервиса на основе RESTful API для системы заказов на покупку билетов, первый из которых реализует авторизацию пользователя, а второй – формирование заказов на покупку билетов авторизованными пользователями.

## **I. Микросервис авторизации пользователей**

## I.1. Требования к API микросервиса авторизации

### I.1.1. Регистрация нового пользователя

- Реализовать конечную точку RESTful API для регистрации нового пользователя.

- Регистрация должна фиксировать информацию о пользователе с минимальным набором полей: имя пользователя (ник), адрес электронной почты (логин), пароль.

- Необходимо предусмотреть проверку входных данных на непустоту и корректность (например, имейл должен содержать символы “@”, “.” и т. д.; пароль должен состоять из не менее восьми символов, включая буквы обоих регистров, цифры и специальные символы; это можно реализовать с помощью регулярных выражений).

- важно предусмотреть различные коды ответов на HTTP-запросы клиентов и ответные сообщения для успешных и неудачных попыток регистрации.

- Необходимо продемонстрировать подтверждающее сообщение после успешной регистрации.

### I.1.2. Вход пользователя в систему (авторизация)

- Реализовать конечную точку RESTful API для входа зарегистрированного пользователя в систему.

- Процесс входа в систему предполагает предоставление сервису электронной почты и пароля ранее зарегистрированного пользователя.

- Необходимо реализовать управление сессией с помощью JWT (токена) для поддержания статуса аутентификации пользователя.

- Требуется возвращать соответствующие коды состояния HTTP и ответные сообщения для успешных и неудачных попыток входа в систему.

- Необходимо продемонстрировать соответствующие сообщения об ошибках в случае неудачных попыток входа в систему с предоставлением некорректных данных.

### I.1.3. Предоставление информации о пользователе

- Реализовать конечную точку RESTful API для выдачи информации о пользователе по токену.

- Требуется возвращать соответствующие коды состояния HTTP.

## I.2. Требования к содержимому реляционной БД микросервиса авторизации

### I.2.1. Таблица `user`

- `id` (integer, primary key, auto-increment): уникальный идентификатор пользователя.

- `nickname` (varchar): отображаемое имя (ник), выбранное пользователем.

- `email` (varchar, unique): адрес электронной почты пользователя.

- `password` (varchar): хешированная версия пароля пользователя.

- `created` (timestamp): дата и время регистрации пользователя.

### I.2.2. Таблица `session`

- `id` (integer, primary key, auto-increment): уникальный идентификатор для каждой сессии.

- `user\_id` (integer, foreign key): связанный идентификатор пользователя.

- `token` (varchar): токен, используемый для аутентификации.

- `expires` (timestamp): дата и время истечения срока действия сеанса.

## I.3. SQL-запросы для создания таблиц БД микросервиса авторизации пользователей (с использованием синтаксиса H2)

### I.3.1. Создание таблицы `user`:

*CREATE TABLE user (*

*id INT AUTO\_INCREMENT PRIMARY KEY,*

*nickname VARCHAR(50) NOT NULL,*

*email VARCHAR(100) UNIQUE NOT NULL,*

*password VARCHAR(255) NOT NULL,*

*created TIMESTAMP DEFAULT CURRENT\_TIMESTAMP()*

*);*

### I.3.2. Создание таблицы `session`:

*CREATE TABLE session (*

*id INT AUTO\_INCREMENT PRIMARY KEY,*

*user\_id INT NOT NULL,*

*token VARCHAR(255) NOT NULL,*

*expires TIMESTAMP NOT NULL,*

*FOREIGN KEY (user\_id) REFERENCES user(id)*

*);*

# **II. Микросервис заказов на покупку билетов**

# II.1. Требования к API микросервиса заказов на покупку билетов

### II.1.1. Создание заказа на покупку билета

- Реализовать конечную точку RESTful API, которая позволяет авторизованным пользователям создавать новые заказы на покупку билетов.

- Каждый заказ (на покупку только одного билета) должен включать следующую информацию: идентификатор пользователя (покупателя), дату и время заказа, идентификатор станции отправления, идентификатор станции назначения и статус заказа.

- Необходимо проверять правильность предоставленных данных.

- Важно возвращать соответствующие коды состояния HTTP и ответные сообщения для успешных и неудачных попыток создания заказа.

### II.1.2. Обработка заказов на покупку билетов

- Создать внутренний эмулятор обработчика заказов, который извлекает из таблицы `order` заказы в статусе 1 (“check”) и с небольшой задержкой в случайном порядке меняет их статусы на 2 (“success”) или 3 (“rejection”).

### II.1.3. Предоставление информации о заказе на покупку билета

- Реализовать конечную точку RESTful API, которая возвращает по идентификатору информацию о заказе на покупку билета.

- Требуется возвращать соответствующие коды состояния HTTP.

## II.2. Требования к содержимому реляционной БД микросервиса обработки заказов на покупку билетов

### II.2.1. Таблица `station`

- `id` (integer, primary key, auto-increment): уникальный идентификатор станции.

- `station` (varchar): название станции.

### II.2.2. Таблица `order`

- `id` (integer, primary key, auto-increment): уникальный идентификатор для каждого заказа на покупку только одного билета

- `user\_id` (integer, foreign key): идентификатор пользователя, разместившего заказ

- `from\_station\_id` (integer, foreign key): идентификатор станции отправления

- `to\_station\_id` (integer, foreign key): идентификатор станции назначения

- `status` (integer): текущий статус заказа (1 – check, 2 – success, 3 – rejection)

- `created` (timestamp): дата и время создания заказа на покупку билета.

## II.3. SQL-запросы для создания таблиц реляционных БД микросервиса обработки заказов на покупку билетов (с использованием синтаксиса H2)

### II.3.1. Создание таблицы `station`:

*CREATE TABLE station (*

*id INT AUTO\_INCREMENT PRIMARY KEY,*

*station VARCHAR(50) NOT NULL*

*);*

### II.3.2. Создание таблицы `order`:

*CREATE TABLE order (*

*id INT AUTO\_INCREMENT PRIMARY KEY,*

*user\_id INT NOT NULL,*

*from\_station\_id* *INT NOT NULL,*

*to\_station\_id INT NOT NULL,*

*status INT NOT NULL,*

*created TIMESTAMP DEFAULT CURRENT\_TIMESTAMP(),*

*FOREIGN KEY (user\_id) REFERENCES user(id),*

*FOREIGN KEY (from\_station\_id) REFERENCES station(id),*

*FOREIGN KEY (to\_station\_id) REFERENCES station(id)*

*);*

# *Критерии оценки ДЗ*

1. Корректная реализация сервиса авторизации пользователей (4 балла):

1.1. Регистрация пользователя (2 балл)

1.2. Вход пользователя в систему (1,5 балла)

1.3. Предоставление информации о пользователе (0.5 балла)

2. Корректная реализация сервиса обработки заказов на покупку билетов (3 балла в случае реализации сервиса авторизации пользователей)

2.1. Обработка заказов на покупку билетов (2.5 балла)

2.2. Предоставление информации о заказе (0.5 балл)

3. Реализация коллекции Postman (или Swagger), которая должна демонстрировать функциональность реализованн(ого/ых) микросервис(а/ов), охватывая все API (1 балл в случае реализации сервис(а/ов)).

4. Качество кода и документация (2 балла):

4.1. Хорошо организованный, модульный и “чистый” код.

4.2. Качественная документация, включая краткое описание архитектуры системы и спецификацию API.

# *Дата выдачи ДЗ: 19 мая 2024 г.*

# *Дедлайн сдачи ДЗ: 10 июня 2024 г. 05:59.*