ДЗ 4. Система обработки заказов ресторана

Цель:

Разработать два отдельных микросервиса на основе RESTful API для системы обработки заказов в ресторане, первый из которых реализует авторизацию пользователей с различными ролями, а второй – управляет заказами и отслеживает запас блюд.

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 информацию о посетителе с минимальным набором полей: имя пользователя (ник), адрес электронной почты (логин), пароль.
- Должна быть учтена возможность предоставления различных ролей пользователю.
- Необходимо предусмотреть проверку входных данных на корректность (например, имейл должен содержать символ “@” и т.д.).
- 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 и его доступах к системе.
- 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 для каждого пользователя.
- `username` (varchar, unique): отображаемое имя (ник), выбранное пользователем.
- `email` (varchar, unique): адрес электронной почты пользователя.
- `password_hash` (varchar): хешированная версия пароля пользователя.
- `role` (varchar): роль пользователя.
- `created_at` (timestamp): дата и время регистрации пользователя.
- `updated_at` (timestamp): дата и время последнего обновления информации о пользователе.

I.2.2. Таблица `session`

- `id` (integer, primary key, auto-increment): уникальный идентификатор для каждой сессии.
- `user_id` (integer, foreign key): связанный идентификатор пользователя.
- `session_token` (varchar): токен, используемый для аутентификации.
- `expires_at` (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,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(10) NOT NULL CHECK (role IN ('customer', 'chef', 'manager')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP() ON UPDATE
CURRENT_TIMESTAMP()
);
```

I.3.2. Создание таблицы `session`:

```
CREATE TABLE session (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    session_token VARCHAR(255) NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

II. Микросервис обработки заказов

II.1. Требования к API микросервиса обработки заказов

II.1.1. Создание заказов

- Реализовать конечную точку RESTful API, которая позволяет пользователям создавать новые заказы.
- Каждый заказ должен включать как минимум следующую информацию: идентификатор пользователя, список блюд с их количеством, специальные запросы 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

- Внедрить внутренний обработчик заказов, который извлекает из таблицы `dish` заказы в статусе “в ожидании”, с некоторой задержкой обрабатывает заказ и меняет его статус на “выполнен”.
- Обновление статуса заказа соответствующим образом (например, “в ожидании”, “в работе”, “выполнен” или “отменен”).

II.1.3. Предоставление информации о заказе

- Реализовать конечную точку RESTful API, которая возвращает по идентификатору заказ и его статус.
- Возвращайте соответствующие коды состояния HTTP.

II.1.4. Управление блюдами

- Внедрить логику управления блюдами, чтобы отслеживать наличие блюд в ресторане.
- Реализовать RESTful API для CRUD таблицы `dish`.
- Доступ к управлению блюдами имеют только пользователи с ролью менеджера.
- Каждое блюдо должно иметь количество штук в наличии (если это 0, значит блюдо недоступно для заказа).
- Возвращайте соответствующие коды состояния HTTP.

II.1.5. Предоставление меню

- Реализовать конечную точку RESTful API, которая возвращает информацию о блюдах в виде меню с учётом доступности.
- Возвращайте соответствующие коды состояния HTTP.

II.2. Требования к содержимому реляционной БД микросервиса обработки заказов

II.2.1. Таблица `dish`

- `id` (integer, primary key, auto-increment): уникальный идентификатор для каждого элемента
- `name` (varchar): название пункта меню
- `description` (text): Описание пункта меню
- `price` (decimal): цена товара.
- `quantity` (integer): количество, доступное для заказа (если 0, то недоступно)

II.2.2. Таблица `order`

- `id` (integer, primary key, auto-increment): уникальный идентификатор для каждого заказа
- `user_id` (integer, foreign key): идентификатор пользователя, разместившего заказ
- `status` (varchar): Текущий статус заказа (в ожидании, в процессе, завершен, отменен)
- `special_requests` (text): дополнительные запросы или инструкции от пользователя.
- `created_at` (timestamp): дата и время размещения заказа.
- `updated_at` (timestamp): Дата и время последнего обновления информации о заказе.

II.2.3. Таблица `order_dish`

- `id` (integer, primary key, auto-increment): уникальный идентификатор для каждого элемента заказа.
- `order_id` (integer, foreign key): ID ассоциированного ордера
- `dish_id` (integer, foreign key): идентификатор связанного пункта меню.
- `quantity` (integer): количество заказанного товара
- `price` (decimal): цена товара на момент заказа

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

II.3.1. Создание таблицы `dish`:

```
CREATE TABLE dish (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    quantity INT NOT NULL,
    is_available BOOLEAN NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP() ON UPDATE
CURRENT_TIMESTAMP()
);
```

II.3.2. Создание таблицы `order`:

```
CREATE TABLE order (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    status VARCHAR(50) NOT NULL,
    special_requests TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP() ON UPDATE
CURRENT_TIMESTAMP(),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

II.3.3. Создание таблицы `order_dish`:

```
CREATE TABLE order_dish (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    dish_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES order(id),
    FOREIGN KEY (dish_id) REFERENCES dish(id)
);
```

Критерии оценки ДЗ

1. Корректная реализация сервиса авторизации пользователей (2 балла):
   1.1. Регистрация пользователя (1 балл)
   1.2. Вход пользователя в систему (0.5 балла)
   1.3. Выдача информации о пользователе (0.5 балла)

2. Корректная реализация сервиса обработки заказов (5 баллов в случае реализации сервиса авторизации пользователей)
   2.1. Управление заказами (2 балла)
   2.2. Управление блюдами (2 балла)
   2.3. Предоставление информации о меню (1 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 поддерживаемый код.
   4.2. Качественная документация, включая краткое описание архитектуры системы и спецификацию API.

Дата выдачи ДЗ: 2 мая 2023 г.

Дедлайн сдачи ДЗ: 29 мая 2023 г. 05:59.