Тестовое задание: Разработка бэкенда для сети кофеен на вынос с использованием Express.js.

Задание:

1. **Управление пользователями:** вам нужно построить систему для управления пользователей, создать им необходимые модели и таблицы внутри базы данных, создать регистрацию, идентификацию и аутентификацию. Создать json-web-token для распознавания пользователя.
2. **Управления заказами:** вам нужно создать модель и таблицу, внутри базы данных, определить контроллер или любой другой обработчик, работающий с crud операциями, который будет создавать, удалять, изменять и отдавать данные о поступившем заказе.
3. **Управление меню системы:** вам нужно создать таблицы и модели для управления меню системы, определить связь между промежуточными и основными таблицами, составить crud операции, для взаимодействием с меню.
4. **Управление корзиной:** вам нужно создать таблицы и модели для корзины, установить необходимые связи, и реализовать функционал для корзины товаров.
5. **Статичная информация:** создать маршрут, который будет отдавать статичные данные, например расположение, график работы и тд.
6. **Защита данных:** создать базовую защиту CSRF, XSS и SQL инъекций.
7. **Swagger:** натянуть приложение на сваггер.
8. **Docker:** для удобства контейнерзируете приложение с помощью docker.

Технологии и библиотеки:

1. **Express.js** - для создания веб сервера и работы с HTTP запросами.
2. **Multer.js** - для загрузки файлов на сервер
3. **SQL типа база данных** - можно использовать любую реляционную бд (mysql, postgresql, sqlite) и тд.
4. **ORM** - вы по желанию можете использовать какую нибудь orm для облегчения работы с бд, например sequalize или prisma
5. **JWT** - можете использовать этот подход для авторизации
6. **Docker** - для контейнеризации
7. **Swagger** - для документации

Примерное описание маршрутов

1. **Users-management:**
    - **POST /registration** - маршрут для регистрации пользователя
    - **POST /authentication** - маршрут для авторизации пользователя
    - **POST /me** - маршрут для получения информации о текущем пользователе
    - **GET /users** - маршрут получения всех пользователей

- **GET /user/:id** - маршрут получения одного пользователя
- **PUT /user/:id** - маршрут для изменения пользователя
- **PATCH /user/:id** - маршрут для редактирования пользователя
- **DELETE /user/:id** - маршрут для удаления пользователя

2. **Categories-management:**
   - **POST /category** - маршрут для создания новой категории
   - **GET /categories** - маршрут получения всех категорий
   - **GET /category/:id** - маршрут получения одной категории
   - **PUT /category/:id** - маршрут для изменения категории
   - **PATCH /category/:id** - маршрут для редактирования категории
   - **DELETE /category/:id** - маршрут для удаления категории

3. **Product-management:**
   - **POST /product** - маршрут для создания нового товара
   - **GET /products -** маршрут получения всех товаров
   - **GET /product/:id -** маршрут получения одного товара
   - **PUT /product/:id** - маршрут для изменения товара
   - **PATCH /product/:id** - маршрут для редактирования товара
   - **DELETE /product/:id** - маршрут для удаления товара

4. **Order-management:**
   - **POST /order -** маршрут для создания нового заказа
   - **GET /orders -** маршрут получения всех заказов
   - **GET /order/:id** - маршрут получения одного заказа
   - **PUT /order/:id** - маршрут для изменения заказа
   - **PATCH /order/:id** - маршрут для редактирования заказа
   - **DELETE /order/:id** - маршрут для удаления заказа

5. **Работа с JWT**
   - **POST /access** - маршрут получения токена
   - **POST /refresh** - маршрут обновления токена

Описание работы программы

Приложение представляет из себя систему, которая работает посредством REST API, с фронтендом. Основной цикл работы программы таков: пользователь, проходит регистрацию внутри приложения, ему показывается интерфейс (интерфейс делать не обязательно) в соответствии его роли. Если роль юзер, то пользователю выводится каталог, с фильтрами, он может открыть карточку товара, добавить один или *несколько* разных товаров в корзину, и сделать заказ на основе этой самой корзины. Если пользователь залогинился под ролью администратора, то в таком случае, администратор может: добавить, удалить, получить одного или всех юзеров, редактировать юзеров, назначать и менять роли. Создавать, редактировать, получать, удалять категории. На основе категорий можно проделывать те же самые операции CRUD с товарами. И внутри админ может получить письмо с заказом на обработку.
Конец программы.

Примечания: данное приложение навесить на сваггер, для тестирования на стороне фронтенда, создать гитхаб репозиторий и прикрепить ссылку на него.Создать блок схему базы данных и загрузить ее на гитхаб репозиторий через readme.md. Express является минимальным фреймворком для этого тз, если вам удобно использовать более масштабируемые технологии, можете их использовать, например nest.js. Если вам необходима какая то библиотека которая тут не описана, можете ее использовать, например, если вы захотите провести регистрацию через почту для пересылки можете использовать nodemailer.