Тестовое задание: Разработка бэкенда для сети кофеен на вынос языком программирования PYTHON.

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. Не забудьте реализовать очистку не верифицированных пользователей из базы данных через например два дня
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. **Чат поддержки:** реализуйте чат поддержки на технологии WEBSOCKET. Это не упускать так как вам придется работать с вебсокетами на данном проекте.
6. **Фильтры:** Добавить для GET маршрутов общие фильтры, которые будут иметь в себе: сортировку, фильтры по любому полю на любое значение, фильтр любого поля по поиску. Добавьте обязательно пагинацию.
7. **Статичная информация:** создать маршрут, который будет отдавать статичные данные, например расположение, график работы и тд.
8. **Защита данных:** создать базовую защиту CSRF, XSS и SQL инъекций.
9. **Swagger:** натянуть приложение на сваггер.
10. **Docker:** для удобства контейнерзируете приложение с помощью docker.

Технологии и библиотеки:

1. **FastAPI** — как основной фреймворк для разработки. Если вам удобно можете выбрать django.
2. **FastAPI UploadFile** — встроенный инструмент 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.
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 /verification** - маршрут для подтверждении личности 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

6. **Cart:**
   POST /cart - маршрут для добавления товаров в корзину
   DELETE /cart/:id - маршрут для удаления одного товара
   DELETE /cart/ - маршрут для удаления всех товаров

Описание работы программы:

Приложение — это REST API для управления пользователями, товарами, корзиной и заказами. Пользователи проходят регистрацию, получают роль, определяющую доступный функционал. "Пользователь" может просматривать каталог, фильтровать товары, добавлять их в корзину и оформлять заказы. "Администратор" управляет пользователями, ролями, категориями, товарами и получает уведомления о заказах на почту.

Приложение работает автономно, без фронтенда. Реализовано на FastAPI, использует PostgreSQL, JWT для аутентификации, SMTP для отправки уведомлений и Docker для контейнеризации. API документировано через встроенный Swagger.

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. FastAPI является минимальным фреймворком для этого тз, если вам удобно использовать более масштабируемые технологии, можете их использовать, например DJANGO. Если вам необходима какая то библиотека которая тут не описана, можете ее использовать.