**Задание: Разработка API для управления складом**

**Описание задачи:**

Необходимо разработать небольшое REST API с использованием **FastAPI** для управления процессами на складе. API должно позволять управлять товарами, складскими запасами и заказами.

**Основные требования:**

1. **Создание структуры базы данных**:
    - Использовать **SQLAlchemy** (2 версии) для взаимодействия с базой данных.
    - Спроектировать таблицы для следующих сущностей:
        - **Product** (Товар): id, название, описание, цена, количество на складе.
        - **Order** (Заказ): id, дата создания, статус (напр. "в процессе", "отправлен", "доставлен").
        - **OrderItem** (Элемент заказа): id, id заказа, id товара, количество товара в заказе.
2. **Реализация REST API**:
    - **Эндпоинты для товаров**:
        - Создание товара (POST /products).
        - Получение списка товаров (GET /products).
        - Получение информации о товаре по id (GET /products/{id}).
        - Обновление информации о товаре (PUT /products/{id}).
        - Удаление товара (DELETE /products/{id}).
    - **Эндпоинты для заказов**:
        - Создание заказа (POST /orders).
        - Получение списка заказов (GET /orders).
        - Получение информации о заказе по id (GET /orders/{id}).
        - Обновление статуса заказа (PATCH /orders/{id}/status).
3. **Бизнес-логика**:
    - При создании заказа проверять наличие достаточного количества товара на складе.
    - Обновлять количество товара на складе при создании заказа (уменьшение доступного количества).
    - В случае недостаточного количества товара – возвращать ошибку с соответствующим сообщением.
4. **Документация**:
    - Использовать встроенную документацию FastAPI (Swagger/OpenAPI).

**Дополнительные требования:**

1. **Тестирование**:
    - Написать несколько тестов с использованием **pytest** для проверки основных функций API.

2. **Docker**:
    - Создать Dockerfile и docker-compose файл для запуска проекта вместе с базой данных (например, PostgreSQL).

**Оценка:**

1. Корректность структуры базы данных и её проектирование.
2. Чистота и логичность кода.
3. Уровень работы с FastAPI и SQLAlchemy.
4. Наличие и качество тестов (если реализовано).
5. Docker-конфигурация (если реализовано).
6. Структура проекта (использование паттернов проектирования).

**Желаем удачи!**🍀