### **Задание 2**

### **API Методы**

#### **Заказы (Orders)**

1. **POST /orders** - создание заказа.
   * Входные данные: { "items": [...], "total\_price": ..., "user\_id": ... }
   * Ответ: { "order\_id": ..., "status": "created" }
2. **PUT /orders/{id}** - обновление заказа.
   * Входные данные: { "items": [...], "total\_price": ... }
   * Ответ: { "order\_id": ..., "status": "updated" }
3. **DELETE /orders/{id}** - отмена заказа.
   * Входные данные: нет.
   * Ответ: { "order\_id": ..., "status": "cancelled" }

#### **Пользователи (Users)**

1. **PUT /users/{id}** - обновление персональных данных.
   * Входные данные: { "name": ..., "email": ..., "phone": ... }
   * Ответ: { "user\_id": ..., "status": "updated" }

#### **Оплаты (Payments)**

1. **POST /payments** - оплата заказа.
   * Входные данные: { "order\_id": ..., "payment\_method": ..., "amount": ... }
   * Ответ: { "payment\_id": ..., "status": "success" }

### **Задание 4 Функциональные требования**

### **Изменение состава заказа:**

### **Возможность добавления новых товаров в заказ.**

### **Возможность удаления товаров из заказа.**

### **Изменение количества каждого товара в заказе.**

### **Обновление общей стоимости заказа:**

### **Перерасчёт стоимости заказа на основании изменений.**

### **Сохранение изменений на сервере:**

### **Обновление данных заказа в базе данных.**

### **Возврат обновлённой информации о заказе клиенту.**

### **Проверка доступности товаров:**

### **Проверка наличия добавляемых товаров перед сохранением изменений.**

### **Логирование действий:**

### **Сохранение изменений для аудита и отката в случае ошибок.**

#### **Нефункциональные требования**

### **Производительность: Обновление данных должно происходить за время не более 1 секунды при нормальной нагрузке.**

### **Надёжность: Обеспечить целостность данных при одновременном редактировании заказа несколькими пользователями.**

### **Безопасность: Все запросы должны быть авторизованы и проходить валидацию на стороне сервера.**

#### **Интерфейс API**

**POST, PUT, DELETE**

#### **Бизнес-логика**

### **Получить идентификатор заказа и данные о товарах из запроса.**

### **Проверить существование заказа в базе данных.**

### **Убедиться в наличии товаров на складе.**

### **Обновить состав заказа и пересчитать стоимость.**

### **Сохранить изменения в базе данных.**

### **Вернуть обновлённые данные клиенту.**

#### **Пример использования**

### **Пользователь выбирает заказ для редактирования в приложении.**

### **Вносит изменения в состав заказа (добавляет, удаляет, изменяет количество товаров).**

### **Сохраняет изменения, отправляя запрос на сервер.**

### **Сервер возвращает обновлённые данные заказа, которые отображаются в интерфейсе пользователя.**