**Кулагиной Валерии ИСдо-42**

# **Дано:**

Мобильное клиент-серверное приложение "Частная пекарня (приложение для продажи выпечки)

**Задание 2.**

**Создание заказа POST**

{

"products": [

{ "product\_code": "A123", "count": 2 },

{ "product\_code": "B456", "count": 1 }

],

"final\_cost": 100,

"customer\_id": "USR001",

"shipping\_address": "Main Str, Apt 12",

"payment\_type": "card"

}

Ответ:

{

"purchase\_id": "ORD789",

"state": "pending"

}

**Модификация существующего заказа (PUT /{purchase\_id})**

Запрос:

{

"products": [

{ "product\_code": "A123", "count": 3 },

{ "product\_code": "C789", "count": 2 }

],

"final\_cost": 150,

"shipping\_address": "Oak Av, House 5",

"payment\_type": "online"

}

Ответ :

{

"purchase\_id": "ORD789",

"state": "modified"

}

**Отмена заказа (DELETE /{purchase\_id})**

Запрос: Нет тела запроса.

Ответ:

{

"purchase\_id": "ORD789",

"state": "cancelled"

}

**Профиль Клиента (Endpoint: /clients)**

Обновление данных клиента (PUT /{client\_id})

Запрос:

{

"full\_name": "Jane Doe",

"contact\_email": "jane.doe@email.com",

"mobile\_number": "+15551234567",

"shipping\_location": "Pine Ln, Building A"

}

Ответ:

{

"client\_id": "USR001",

"status": "profile\_updated"

}

**Обработка Платежей (Endpoint: /transactions)**

Инициация платежа (POST)

Запрос:

{

"purchase\_id": "ORD789",

"payment\_method": "card",

"amount\_due": 150

}

*Ответ (Успех):*

{

"transaction\_id": "TXN101",

"status": "approved"

}

*Ответ (Неудача):*

{

"transaction\_id": null,

"status": "declined",

"reason": "Insufficient Funds"

}

**Дополнительные Операции (Синхронизация Данных)**

Получение информации о заказе (GET /{purchase\_id})

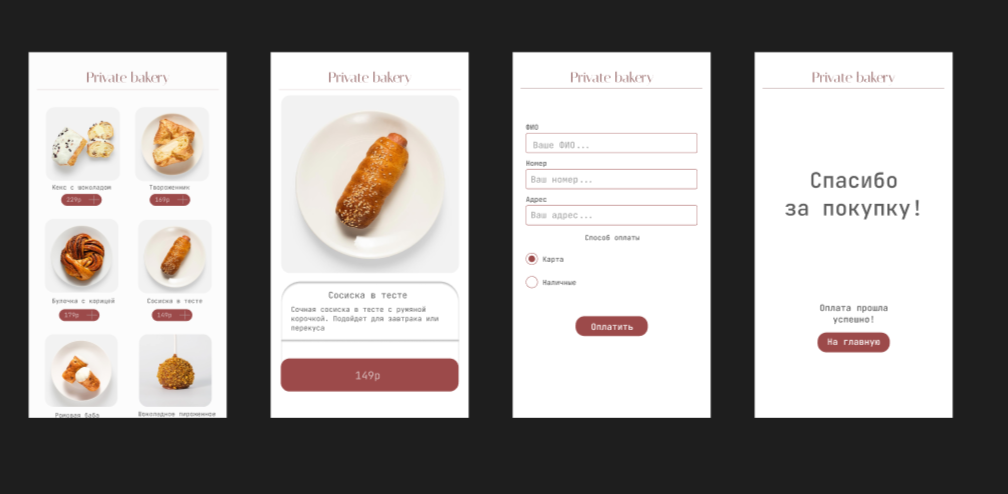
Запрос: Нет тела запроса

Ответ:

{ "purchase\_id": "ORD789", "state": "approved", "final\_cost": 100, "shipping\_address": "Main Str, Apt 12", "items": [ { "product\_code": "A123", "description": "Widget X", "unit\_price": 50, "count": 2}, { "product\_code": "B456", "description": "Gadget Y", "unit\_price": 50, "count": 1} ] }

**Задание 3.**

[**Макет Фигма**](https://www.figma.com/design/0EUceRjXgkRYV9E7wLbY6r/Untitled?node-id=0-1&p=f&t=ba2vd644BYovCw79-0)



**Задание 4.**

**Условия редактирования заказов**

1. **Доступные для редактирования заказы**:  
    - Заказы в статусах "создан" и "ожидает обработки".   
    - Статус заказа не позволяет вносить изменения после перехода в статус "оплачен", "доставлен" или "аннулирован".
2. **Запрещенные к изменению заказы**:  
    - Заказы, статус которых:   
    - "Оплачен"   
    - "Доставлен"   
    - "Аннулирован"   
    - Все изменения в таких заказах игнорируются.
3. **Доступные для изменения параметры**:  
    - **Ассортимент товаров**:   
    - Добавление новых позиций.   
    - Удаление товаров из заказа.   
    - Корректировка количества существующих единиц.   
    - **Адрес доставки**:   
    - Полное изменение адреса.   
    - **Способ оплаты**:   
    - Замена метода оплаты (например, с наличного на безналичный расчет).
4. **Обработка изменений**:  
    - При модификации товаров автоматически производится перерасчет общей стоимости заказа.   
    - Новейшие данные сохраняются в базе данных.
5. **Подтверждение изменений**:  
    - Все внесенные изменения требуют подтверждения пользователем перед окончательным сохранением.   
    - Это позволяет избежать случайных или ошибочных правок.
6. **Синхронизация данных**:  
    - Любые изменения должны быть синхронизированы с серверной базой данных для обеспечения целостности информации.   
    - В случае потери данных из-за технических сбоев система восстанавливает их из базы

**Формат входных данных:**

API-метод: PUT /orders/{id}

**Входные данные:**

{

"items": [

{ "item\_id": 1, "quantity": 2 },

{ "item\_id": 3, "quantity": 1 }

],

"delivery\_address": "ул. Пушкина, 15",

"payment\_method": "cash"

}

**Формат ответа**

**Успешное выполнение:**

{

"order\_id": 456,

"status": "updated",

"total\_price": 1500,

"message": "Order successfully updated"

}

**Ошибка выполнения (например, заказ не подлежит редактированию):**

{

"order\_id": 456,

"status": "error",

"message": "Order cannot be updated as it is already paid or shipped"

}

Логика работы системы:

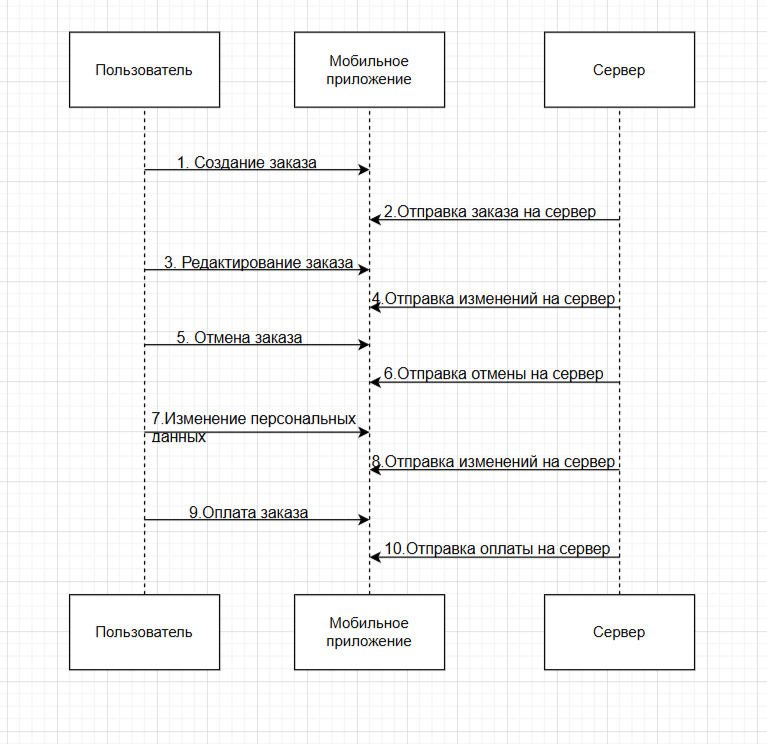
1. Проверка статуса заказа. Если статус не позволяет вносить изменения, то выводится сообщение об ошибке.
2. Проверка наличия товаров на складе для добавляемых позиций. Если товара недостаточно, то выводится сообщение об ошибке.
3. Расчёт общей стоимости. На основе стоимости каждого товара и их количества формируется итоговая сумма. Если пользователь применил скидку или промокод, то эти условия также учитываются.
4. Сохранение изменений. Обновлённые данные сохраняются в базе данных. Пользователь получает информацию об обновленном состоянии заказа.
5. Хранение информации о покупках пользователя.

Структура хранения данных:

1. Заказы:  
    — id — уникальный идентификатор заказа.  
    — user\_id — идентификатор пользователя.  
    — status — статус заказа (создан, оплачен, отменён).  
    — total\_price — итоговая стоимость.  
    — delivery\_address — адрес доставки.  
    — payment\_method — способ оплаты.
2. Товары в заказе:  
    — id — уникальный идентификатор позиции заказа.  
    — order\_id — идентификатор заказа.  
    — item\_id — идентификатор товара.  
    — quantity — количество.
3. Товары:  
    — id — уникальный идентификатор товара.  
    — name — название товара.  
    — price — цена товара.

Действия при редактировании:

1. Удаление старых позиций заказа.
2. Добавление новых позиций заказа.
3. Обновление общей стоимости в таблице Orders.



**Задание 5. SQL-запросы**

**1. Вывести покупателей с количеством осуществленных покупок:**

SELECT

p.id AS customer\_id,

p.name AS customer\_name,

p.surname AS customer\_surname,

COUNT(b.id) AS purchase\_count

FROM

Покупатель p

LEFT JOIN

Покупки b ON p.id = b.key\_customer

GROUP BY

p.id, p.name, p.surname

ORDER BY

purchase\_count DESC;

**2. Общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания:**

SELECT

p.id AS customer\_id,

p.name AS customer\_name,

p.surname AS customer\_surname,

COALESCE(SUM(t.cost), 0) AS total\_spent

FROM

Покупатель p

LEFT JOIN

Покупки b ON p.id = b.key\_customer

LEFT JOIN

Товары t ON b.key\_product = t.id

GROUP BY

p.id, p.name, p.surname

ORDER BY

total\_spent DESC;

**3. Получить покупателей, купивших только один товар:**

SELECT

p.id AS customer\_id,

p.name AS customer\_name,

p.surname AS customer\_surname

FROM

Покупатель p

JOIN

Покупки b ON p.id = b.key\_customer

GROUP BY

p.id, p.name, p.surname

HAVING

COUNT(b.id) = 1;