**Беляков Антон Исдо-42**

Дано:

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

Задание 2.

Заказы (Orders)

1. POST /orders - создание заказа.

**Входные данные:**

{

"items": [

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

{ "item\_id": 5, "quantity": 3 }

],

"total\_price": 1200,

"user\_id": 789,

"delivery\_address": "ул. Горького, 25",

"payment\_method": "debit\_card"

}

**Ответ:**

{

"order\_id": 321,

"status": "created"

}

2. PUT /orders/{id} - обновление заказа.

**Входные данные:**

{

"items": [

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

{ "item\_id": 6, "quantity": 1 }

],

"total\_price": 1400,

"delivery\_address": "ул. Чехова, 18",

"payment\_method": "online\_payment"

}

**Ответ:**

{

"order\_id": 321,

"status": "updated"

}

3. DELETE /orders/{id} - отмена заказа.

**Входные данные:**

**Нет.**

**Ответ:**

{

"order\_id": 321,

"status": "cancelled"

}

Пользователи (Users)

PUT /users/{id} - обновление персональных данных.

**Входные данные:**

{

"name": "Сергей Сергеев",

"email": "sergey@example.com",

"phone": "+79007654321",

"default\_address": "ул. Толстого, 12"

}

**Ответ:**

{

"user\_id": 789,

"status": "updated"

}

Оплаты (Payments)

POST /payments - оплата заказа.

**Входные данные:**

{

"order\_id": 321,

"payment\_method": "debit\_card",

"amount": 1400

}

**Ответ (успешная оплата):**

{

"payment\_id": 654,

"status": "success"

}

**Ответ (неуспешная оплата):**

{

"payment\_id": null,

"status": "failed",

"message": "Оплата не прошла. Пожалуйста, попробуйте снова."

}

Дополнительные методы для синхронизации:

GET /orders/{id} - получить детали заказа.

**Входные данные: Нет.**

**Ответ:**

{

"order\_id": 321,

"status": "created",

"total\_price": 1200,

"delivery\_address": "ул. Горького, 25",

"items": [

{ "item\_id": 2, "name": "Пирожок с мясом", "price": 300, "quantity": 1 },

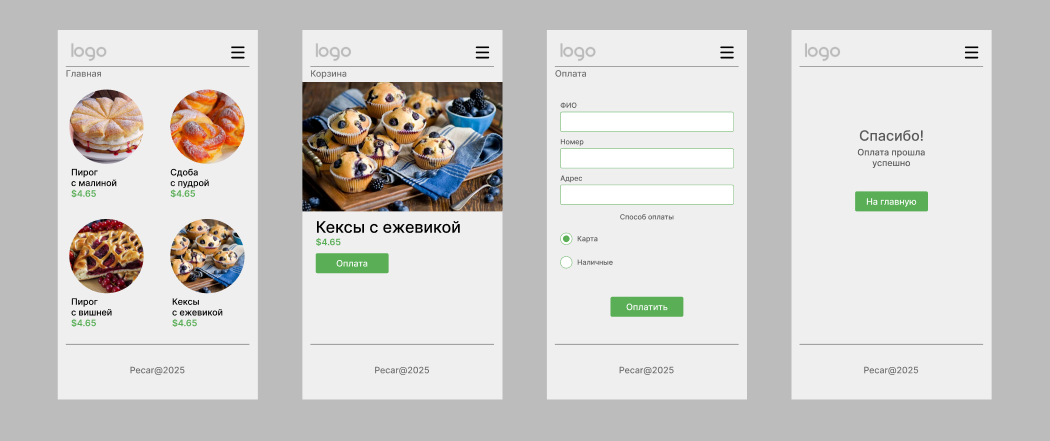
{ "item\_id": 5, "name": "Чизкейк", "price": 150, "quantity": 3 }

]

}

Задание 3.

<https://www.figma.com/design/XtWn0GMTCMQ677alnApXOX/Untitled?node-id=0-1&m=dev&t=CfTolLhljDZG4MWA-1>



Задание 4.

Описание функции: *Обновление заказа*

Описание функционала

Условия обновления:

• Обновление доступно только для заказов со статусами: created, pending.

• Заказы со статусами paid, shipped или cancelled не могут быть изменены.

Параметры, доступные для изменения:

• Товары в заказе:

• Добавление новых товаров.

• Удаление существующих товаров.

• Изменение количества уже добавленных товаров.

• Адрес доставки.

• Способ оплаты.

Обработка изменений:

• При изменении товаров пересчитывается общая стоимость заказа.

• Все изменения должны быть синхронизированы с сервером.

• Обновленный заказ должен быть подтвержден пользователем перед сохранением.

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"

}

Бизнес-логика:

1. Проверка статуса заказа:

• Если статус заказа не позволяет обновление, возвращается ошибка.

2. Проверка наличия товаров:

• Проверяется наличие на складе для добавляемых товаров.

• Если товара недостаточно, возвращается ошибка.

3. Пересчет общей стоимости:

• На основании стоимости каждого товара и их количества формируется итоговая сумма.

• Если пользователь применял скидку или промокод, эти условия также учитываются.

4. Сохранение изменений:

• Обновленные данные сохраняются в базе данных.

• Пользователь получает обновленное состояние заказа.

Хранение информации о покупках пользователя:

Структура хранения данных:

**Orders (Заказы):**

• id: уникальный идентификатор заказа.

• user\_id: идентификатор пользователя.

• status: статус заказа (created, paid, cancelled).

• total\_price: итоговая стоимость.

• delivery\_address: адрес доставки.

• payment\_method: способ оплаты.

**Order\_Items (Товары в заказе):**

• id: уникальный идентификатор позиции заказа.

• order\_id: идентификатор заказа.

• item\_id: идентификатор товара.

• quantity: количество.

**Items (Товары):**

• id: уникальный идентификатор товара.

• name: название товара.

• price: цена товара.

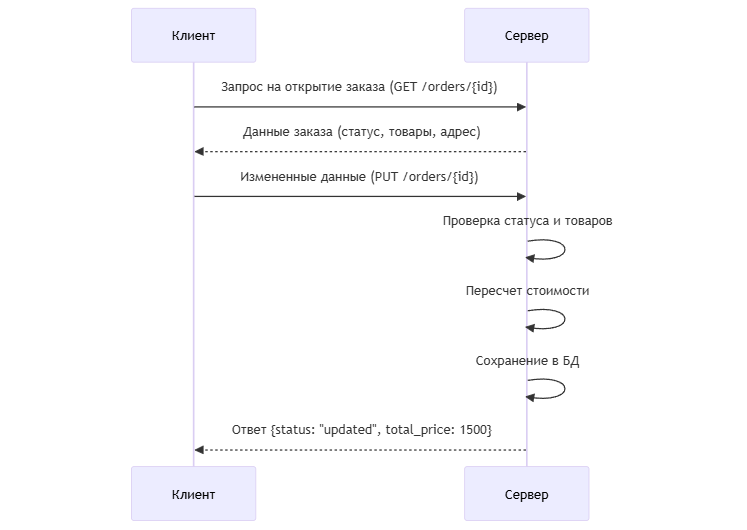
Действия при редактировании:

• Удаление старых позиций заказа.

• Добавление новых позиций заказа.

• Обновление общей стоимости в таблице Orders.

UML последовательности:



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;