# Федорин Максим Исдо-42

# Дано:

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

Задание 2. API методы

**1. POST /orders** - создание заказа.

**Входные данные**:

{

"items": [

{ "item\_id": 2, "quantity": 3 },

{ "item\_id": 5, "quantity": 1 }

],

"total\_price": 1200,

"user\_id": 456,

"delivery\_address": "ул. Чехова, 12",

"payment\_method": "debit\_card"

}

Ответ:

{

"order\_id": 789,

"status": "created"

}

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

Входные данные:

{

"items": [

{ "item\_id": 2, "quantity": 4 },

{ "item\_id": 6, "quantity": 3 }

],

"total\_price": 1800,

"delivery\_address": "ул. Гоголя, 20",

"payment\_method": "online\_payment"

}

Ответ:

{

"order\_id": 789,

"status": "updated"

}

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

Ответ:

{

"order\_id": 789,

"status": "cancelled"

}

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

**Входные данные**:

{

"name": "Петр Петров",

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

"phone": "+79007654321",

"default\_address": "ул. Толстого, 8"

}

**Ответ**:

{

"user\_id": 456,

"status": "updated"

}

**POST /payments** - оплата заказа.

**Входные данные**:

{

"order\_id": 789,

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

"amount": 1800

}

**Ответ (успешная оплата)**:

{

"payment\_id": 123,

"status": "success"

}

**Ответ (неуспешная оплата)**:

{

"payment\_id": null,

"status": "failed",

"message": "Payment failed. Please try again."

}

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

**Ответ**:

{

"order\_id": 789,

"status": "created",

"total\_price": 1200,

"delivery\_address": "ул. Чехова, 12",

"items": [

{ "item\_id": 2, "name": "Сэндвич с курицей", "price": 300, "quantity": 3},

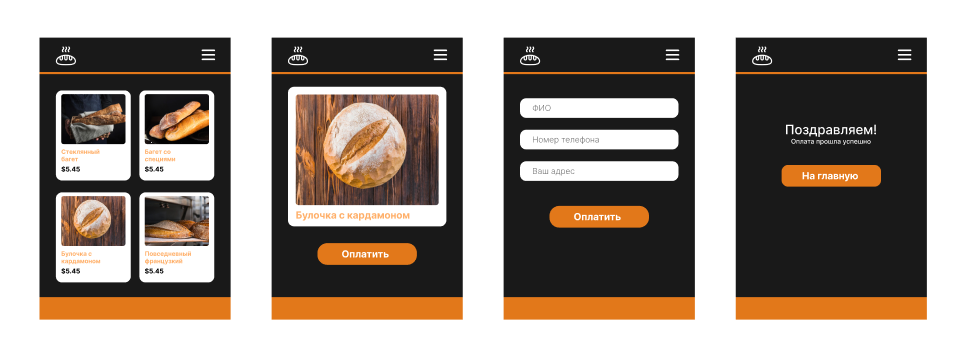
{ "item\_id": 5, "name": "Торт Наполеон", "price": 150, "quantity": 1}

]

}

Задание 3.

https://www.figma.com/design/N3LD7k60cpmco3heB4MYkD/Untitled?node-id=0-1&t=8CRVg7wUKa73QsQG-1



Задание 4.

**Описание функции: Редактирование заказа**

**Описание функционала:**

**Функция редактирования заказа позволяет пользователю вносить изменения в существующий заказ при условии, что заказ находится в определенных статусах. Редактирование недоступно для заказов, которые уже оплачены, отправлены или отменены.**

**Условия редактирования:**

* **Редактирование доступно только для заказов со статусами: created, pending.**
* **Заказы со статусами paid, shipped или cancelled не подлежат изменению.**

**Параметры, доступные для изменения:**

**Товары в заказе:**

* **Добавление новых товаров.**
* **Удаление товаров.**
* **Изменение количества существующих 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. Лермонтова, 10",

"payment\_method": "credit\_card"

}

**Формат ответа**

**{**

**"order\_id": 123,**

**"status": "updated",**

**"total\_price": 2000,**

**"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 не позволяет редактирование, возвращается ошибка.

1. **Проверка доступности товаров:**

Проверяется наличие на складе для добавляемых товаров.

Если товара недостаточно, возвращается ошибка.

1. **Пересчет общей стоимости:**

На основе стоимости каждого товара и их количества формируется сумма.

Если пользователь применял скидку или промокод, эти условия тоже учитываются.

1. **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.

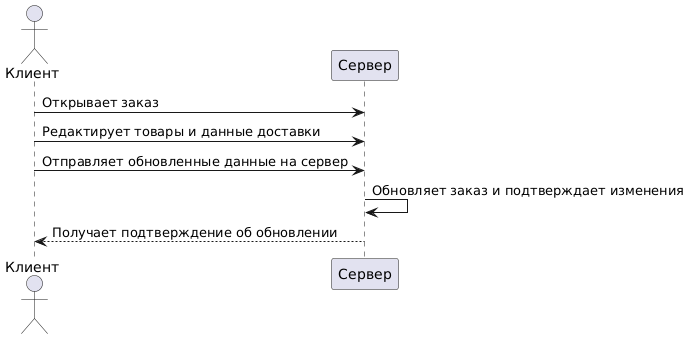
**Items (Товары):**

* 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;