Осягиной Е.В. ИСдо-34

Дано:

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

Задание 2. API методы

**Заказы (Orders)**

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

**Входные данные**:

{

"items": [

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

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

],

"total\_price": 1000,

"user\_id": 123,

"delivery\_address": "ул. Ленина, 10",

"payment\_method": "credit\_card"

}

Ответ:

{

"order\_id": 456,

"status": "created"

}

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

Входные данные:

{

"items": [

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

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

],

"total\_price": 1500,

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

"payment\_method": "cash"

}

Ответ:

{

"order\_id": 456,

"status": "updated"

}

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

Входные данные:  
Нет.

Ответ:

{

"order\_id": 456,

"status": "cancelled"

}

**Пользователи (Users)**

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

**Входные данные**:

{

"name": "Иван Иванов",

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

"phone": "+79001234567",

"default\_address": "ул. Ленина, 5"

}

**Ответ**:

{

"user\_id": 123,

"status": "updated"

}

**Оплаты (Payments)**

**POST /payments** - оплата заказа.

**Входные данные**:

{

"order\_id": 456,

"payment\_method": "credit\_card",

"amount": 1500

}

**Ответ (успешная оплата)**:

{

"payment\_id": 789,

"status": "success"

}

**Ответ (неуспешная оплата)**:

{

"payment\_id": null,

"status": "failed",

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

}

**Дополнительные методы для синхронизации:**

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

**Входные данные**: Нет.

**Ответ**:

{

"order\_id": 456,

"status": "created",

"total\_price": 1000,

"delivery\_address": "ул. Ленина, 10",

"items": [

{ "item\_id": 1, "name": "Круассан с шоколадом", "price": 200, "quantity": 2},

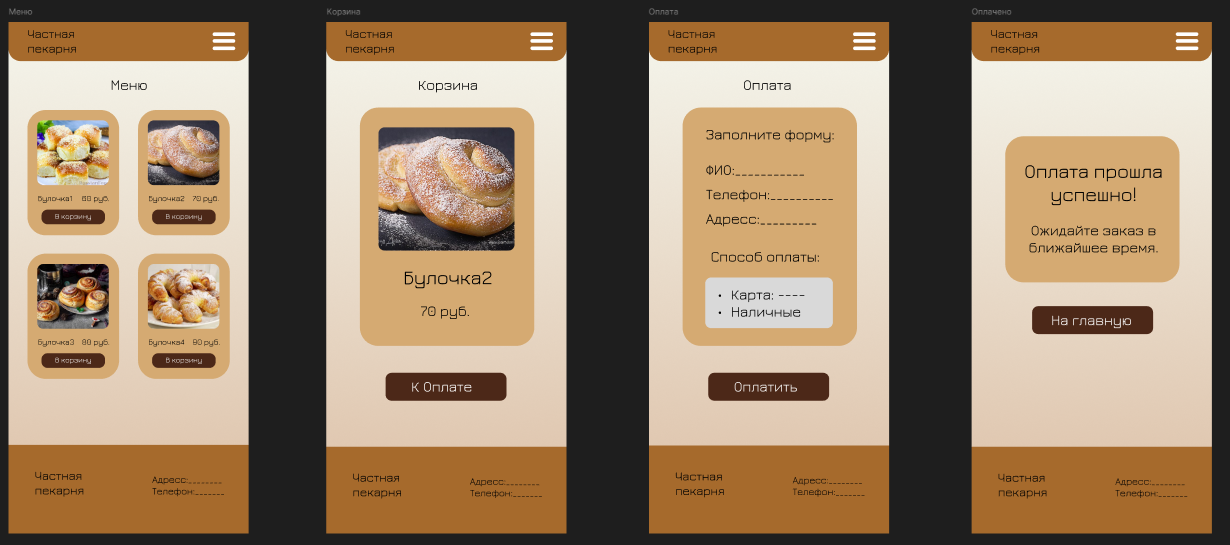
{"item\_id": 3, "name": "Булочка с корицей", "price": 100, "quantity": 1}

]

}

Задание 3.

<https://www.figma.com/design/svEFrVuS8LeMkf46MD5RBj/%D0%97%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-3.-MediaSoftPractice?t=UxBMT4Q7kGNmzmO0-1>



Задание 4.

**Описание функции: Редактирование заказа**

**Цель:**

Обеспечить возможность изменения параметров существующего заказа, таких как список товаров, их количество, адрес доставки и способ оплаты. Изменения возможны только для заказов, которые еще не были оплачены или доставлены.

**Описание функционала**

**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. 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:**

**Проверка статуса заказа:**

* Если статус не позволяет редактирование, возвращается ошибка.

**Проверка доступности товаров:**

* Проверяется наличие на складе для добавляемых товаров.
* Если товара недостаточно, возвращается ошибка.

**Пересчет общей стоимости:**

* На основе стоимости каждого товара и их количества формируется сумма.
* Если пользователь применял скидку или промокод, эти условия тоже учитываются.

**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.

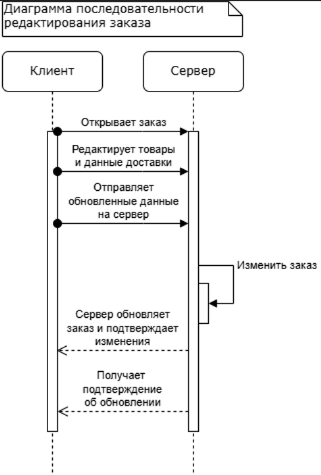
**Items (Товары)**:

* 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;