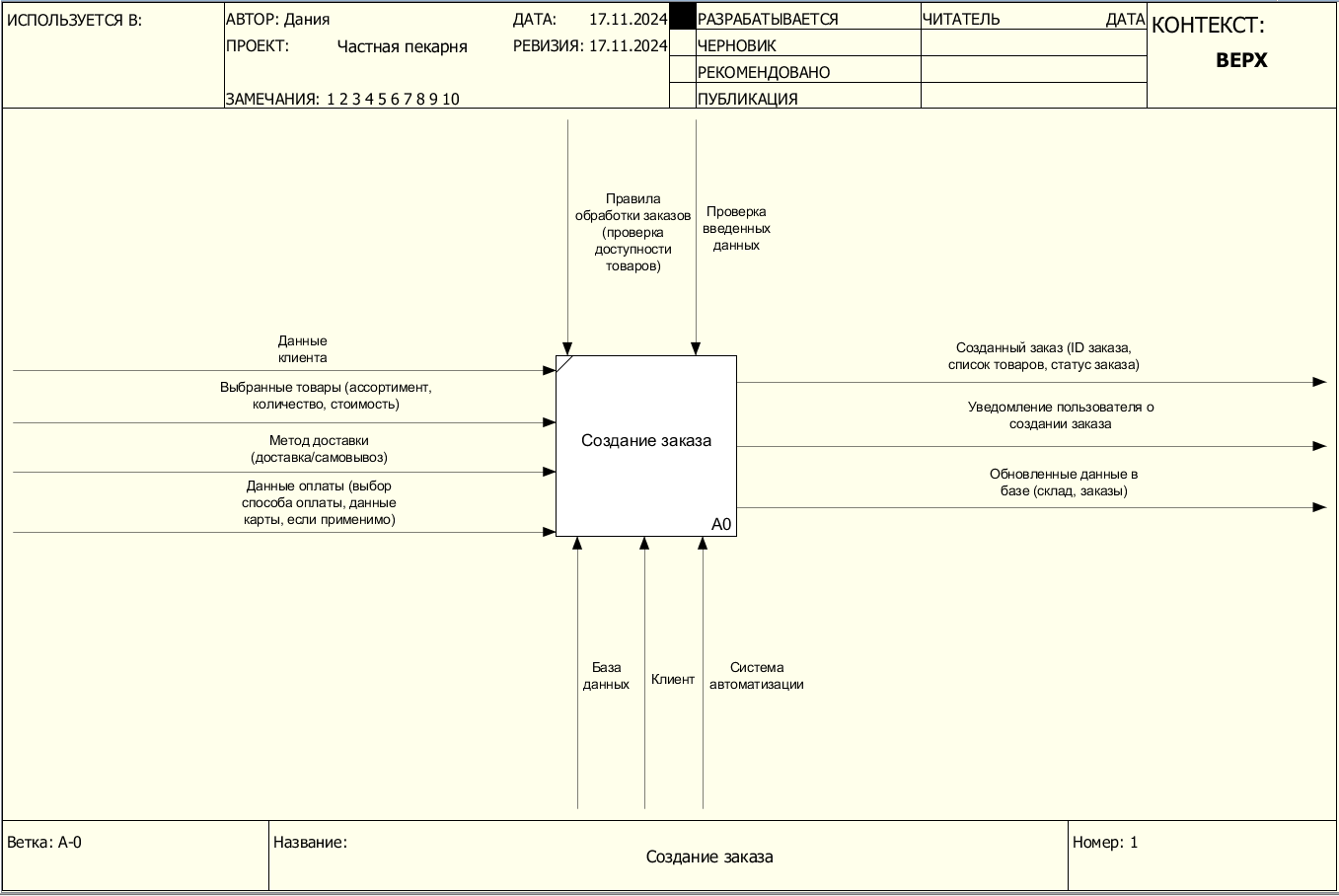
# Дано:

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

# Необходимо описать:

1. Бизнес-процесс создания заказа (использовать любую удобную нотацию моделирования бизнес-процессов).

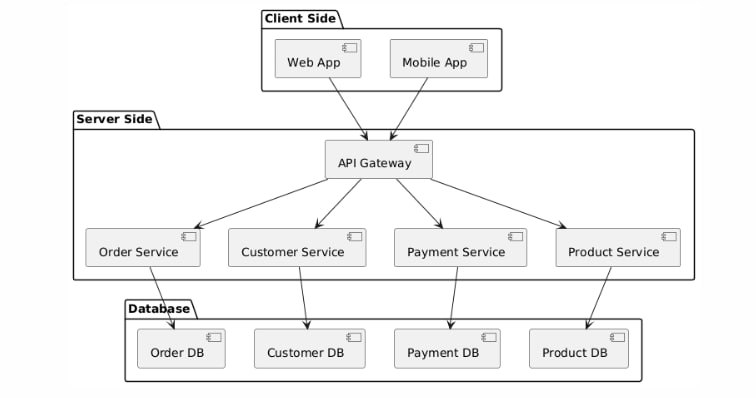
Для выполнения данного задания была выбрана нотация IDEF0. Ниже представлена диаграмма бизнес-процесса создания заказа.



1. Опишите процесс синхронизации данных между клиентом и сервером

(создание, редактирование и отмена заказа, изменение персональных данных, оплата заказа и т.д.). Представить все в диаграммах UML, API методах и других представлениях, также составить ER-диаграмму сущностей.

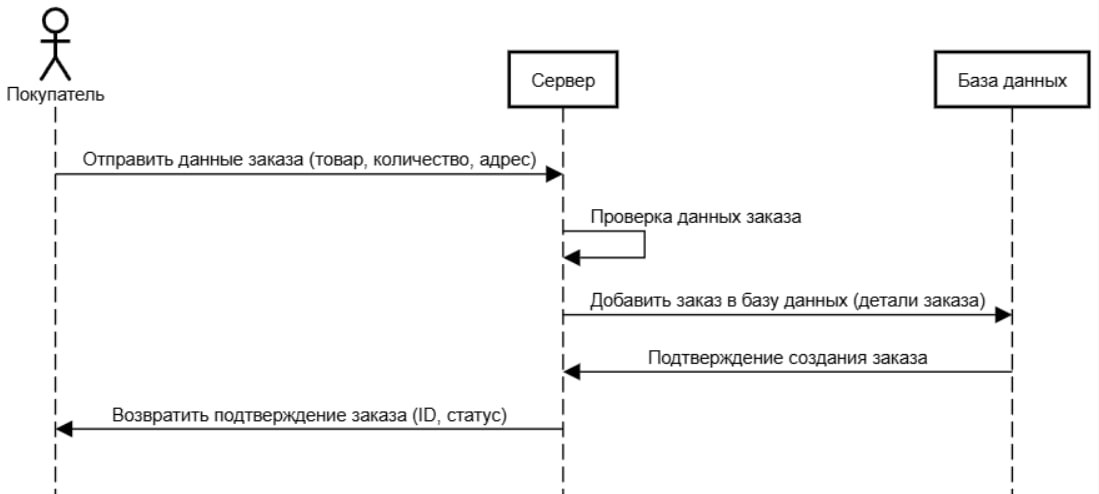
**Диаграмма классов:**



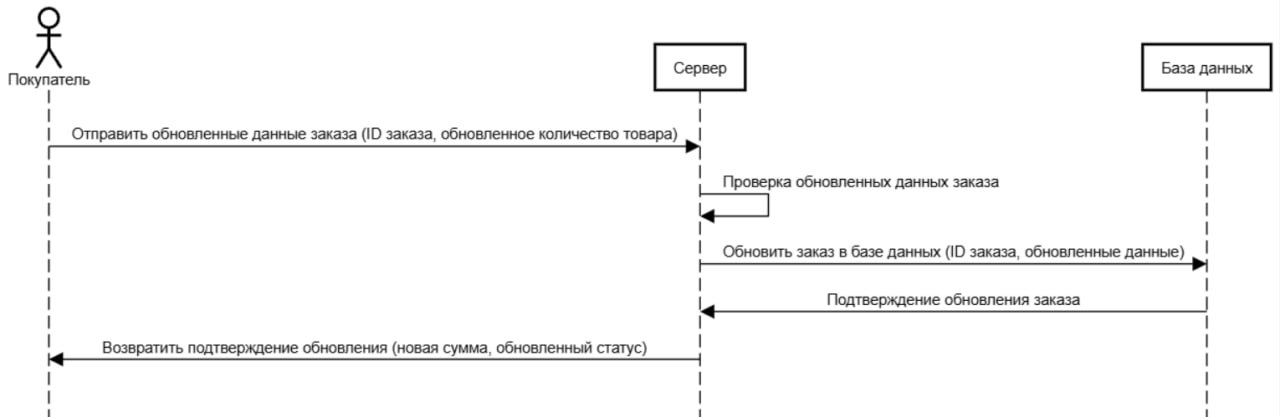
**Диаграммы последовательностей:**

Каждая из диаграмм представляет собой последовательность шагов для определенной операции.

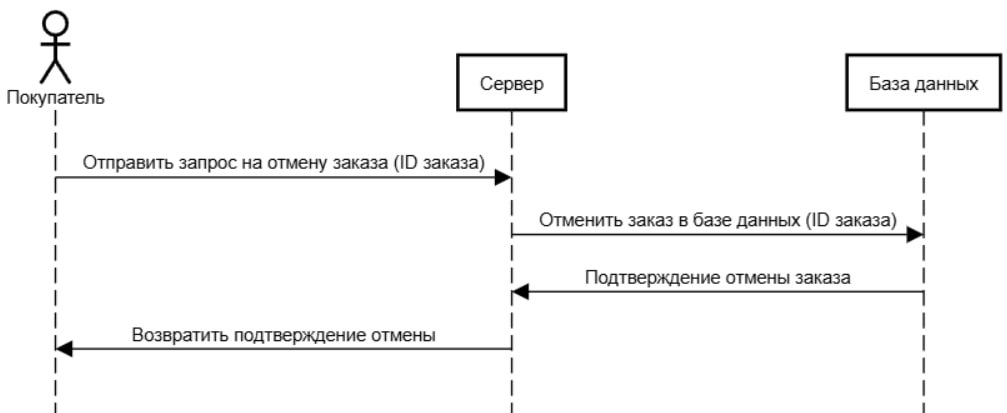
Создание заказа:



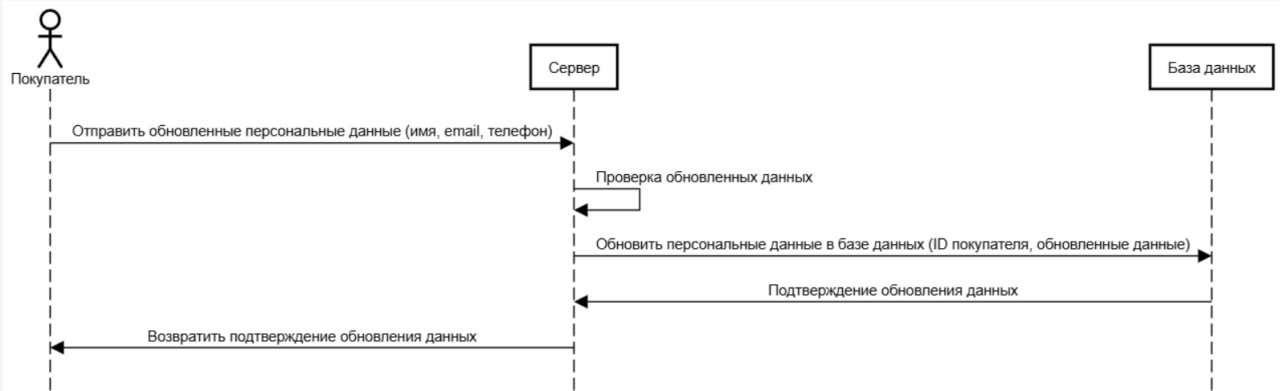
Редактирование заказа:



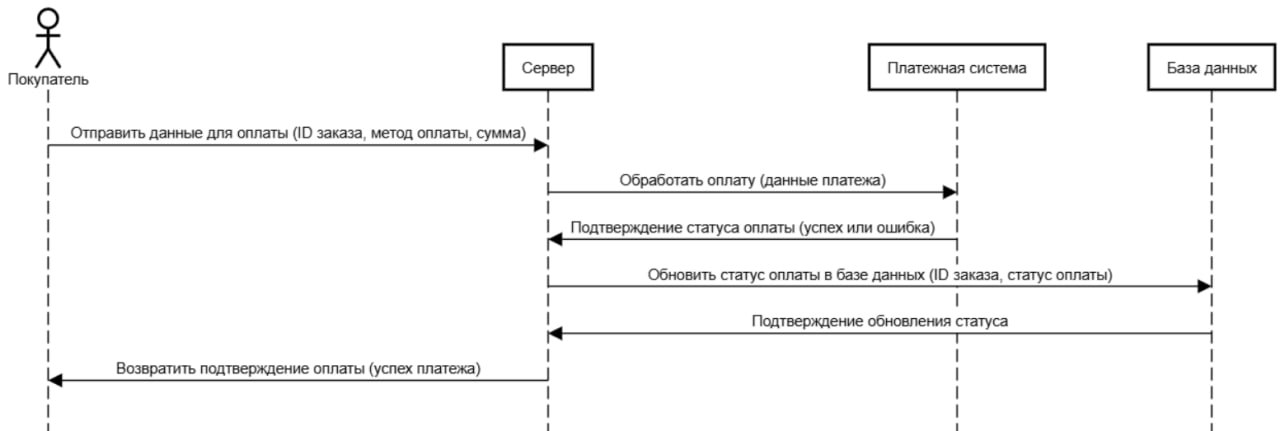
Отмена заказа:



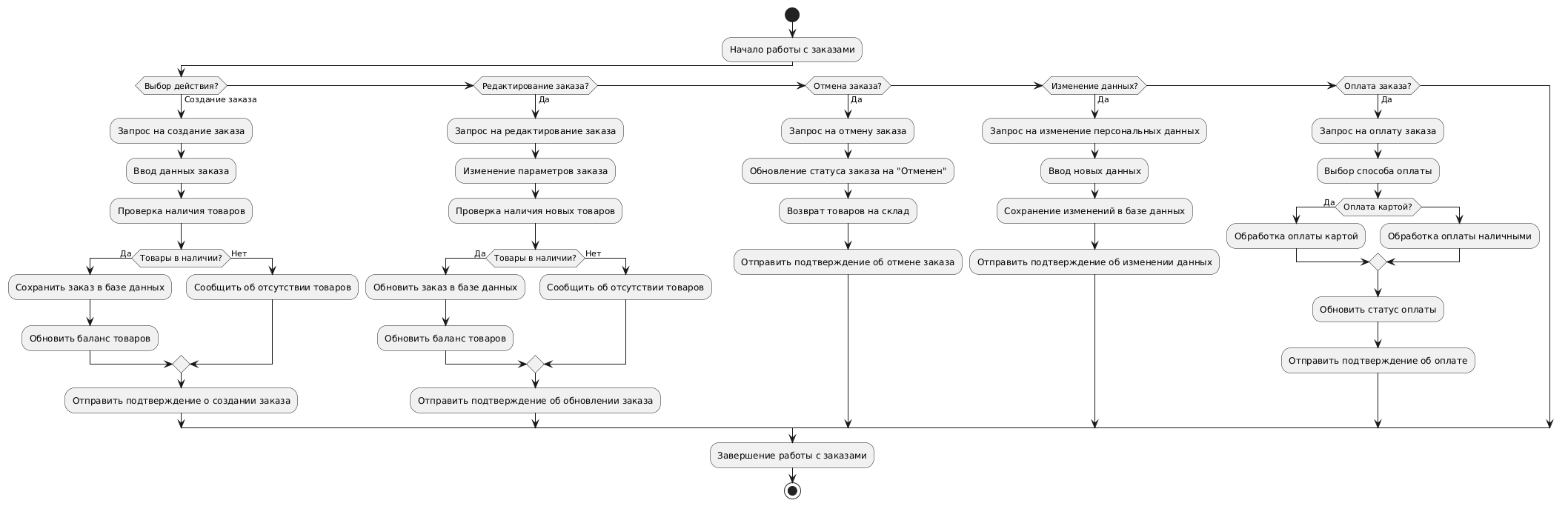
Изменение данных:



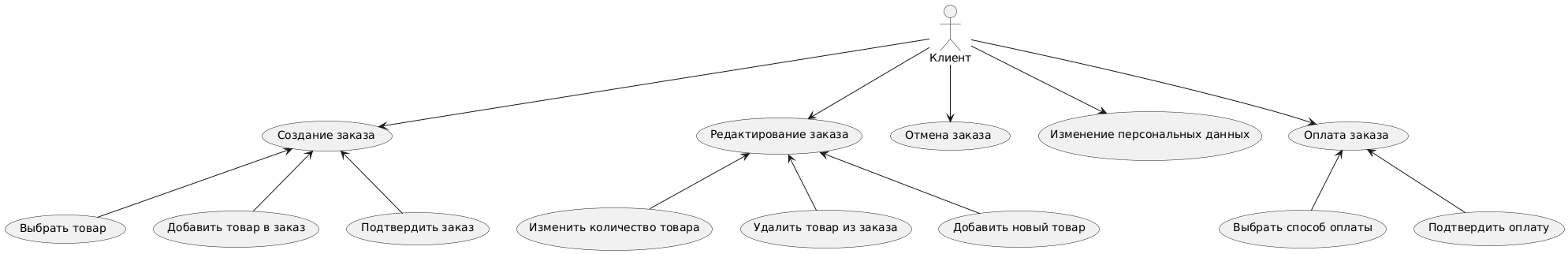
Оплата:



**Диаграмма активности:**



**Диаграмма сценариев:**

****

**API методы:**

1. Создание заказа

Метод: POST /orders/create

Описание: Создание нового заказа для клиента.

Параметры запроса (Request body):

{

"customerId": "int",

"items": [

{

"productId": "int",

"quantity": "int"

}

],

"paymentMethod": "string" // Оплата картой или наличными

}

Ответ (Response body):

{

"orderId": "int",

"status": "string"

}

Описание: Создает новый заказ с указанными товарами и методом оплаты. Возвращает идентификатор заказа и его статус.

2. Редактирование заказа

Метод: PUT /orders/{orderId}/edit

Описание: Редактирование существующего заказа.

Параметры запроса (Request body):

{

"items": [

{

"productId": "int",

"quantity": "int"

}

],

"removeItems": [

"int" // Список productId товаров для удаления

],

"status": "string" // Новый статус заказа (например, "В процессе", "Отменен")

}

Ответ (Response body):

{

"orderId": "int",

"status": "string"

}

Описание: Редактирует заказ, обновляет количество товаров, удаляет товары, изменяет статус заказа. Возвращает обновленный статус заказа.

3. Отмена заказа

Метод: DELETE /orders/{orderId}/cancel

Описание: Отмена существующего заказа.

Параметры запроса (Request body): Нет.

Ответ (Response body):

{

"orderId": "int",

"status": "string"

}

Описание: Отменяет заказ, изменяя его статус на "Отменен". Возвращает подтверждение отмены.

4. Изменение персональных данных клиента

Метод: PUT /customers/{customerId}/update

Описание: Обновление персональных данных клиента.

Параметры запроса (Request body):

{

"firstName": "string",

"lastName": "string",

"email": "string",

"phone": "string"

}

Ответ (Response body):

{

"customerId": "int",

"status": "string"

}

Описание: Изменяет персональные данные клиента (например, имя, email, телефон). Возвращает подтверждение успешного обновления.

5. Запрос на оплату заказа

Метод: POST /payments/{orderId}/process

Описание: Инициация процесса оплаты для заказа.

Параметры запроса (Request body):

{

"paymentMethod": "string", // "card" или "cash"

"amount": "float"

}

Ответ (Response body):

{

"paymentId": "int",

"status": "string"

}

Описание: Инициирует оплату для указанного заказа. Возвращает идентификатор платежа и его статус ("Успешно", "Неудачно").

6. Проверка статуса платежа

Метод: GET /payments/{paymentId}/status

Описание: Проверка статуса платежа.

Параметры запроса (Query parameters):

{

"paymentId": "int"

}

Ответ (Response body):

{

"paymentId": "int",

"status": "string", // "Завершен", "В процессе", "Неудачен"

"amount": "float"

}

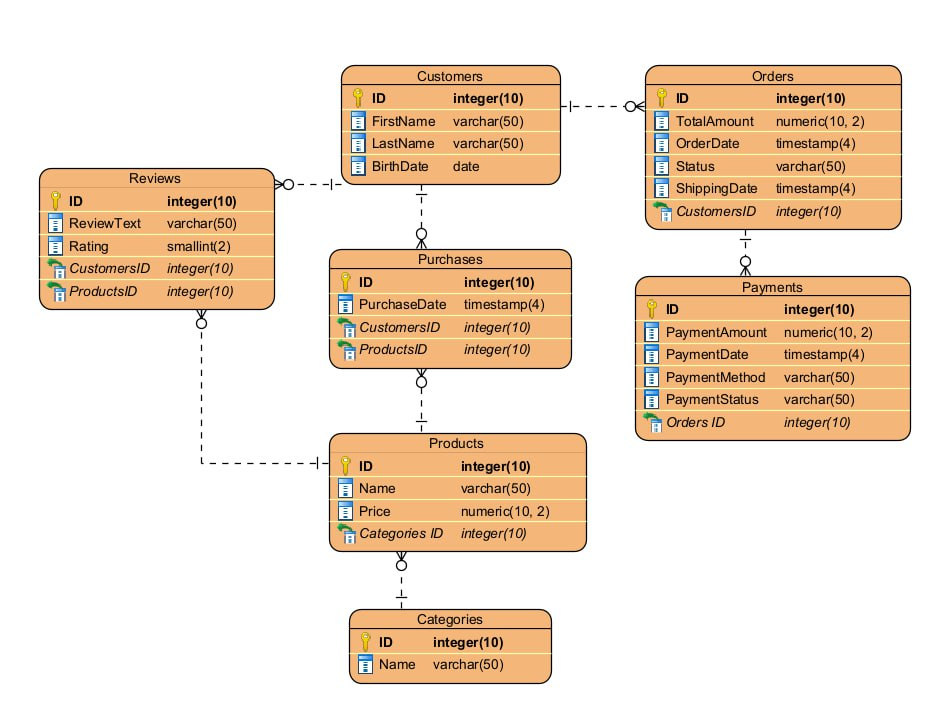
Описание: Возвращает статус платежа и его сумму.

Примечания:

API Gateway принимает запросы от клиента и передает их соответствующему сервису (например, сервису заказов, платежному сервису, и т. д.).

Все взаимодействия с базой данных выполняются через соответствующие сервисы (например, OrderService, PaymentService, CustomerService), которые обрабатывают запросы, получают и сохраняют данные в базе.

**ER-диаграмма сущностей:**



1. Customers (Покупатели)

• Атрибуты:

* ID (PK): Уникальный идентификатор покупателя.
* FirstName: Имя покупателя.
* LastName: Фамилия покупателя.
* BirthDate: Дата рождения покупателя.

• Связи:

* Один покупатель может иметь несколько заказов (Orders).
* Один покупатель может оставить несколько отзывов (Reviews).
* Один покупатель может совершить несколько покупок (Purchases).

2. Products (Товары)

• Атрибуты:

* ID (PK): Уникальный идентификатор товара.
* Name: Название товара.
* Price: Стоимость товара.

• Связи:

* Один товар, может быть, в нескольких покупках (Purchases).
* Один товар может быть связан с несколькими отзывами (Reviews).
* Один товар может принадлежать одной или нескольким категориям (Categories).

3. Purchases (Покупки)

• Атрибуты:

* ID (PK): Уникальный идентификатор покупки.
* CustomerID (FK): Идентификатор покупателя.
* ProductID (FK): Идентификатор товара.
* PurchaseDate: Дата покупки.

• Связи:

* Множество покупок могут быть сделаны одним покупателем (Customers).
* Множество товаров могут быть куплены через покупки (Products).

4. Reviews (Отзывы)

• Атрибуты:

* ID (PK): Уникальный идентификатор отзыва.
* CustomerID (FK): Идентификатор покупателя, оставившего отзыв.
* ProductID (FK): Идентификатор товара.
* Rating: Оценка товара (от 1 до 5).
* Comment: Текст отзыва.

• Связи:

* Один отзыв принадлежит одному покупателю (Customers).
* Один отзыв связан с одним товаром (Products).

5. Categories (Категории)

• Атрибуты:

* ID (PK): Уникальный идентификатор категории.
* Name: Название категории.

• Связи:

* Одна категория может включать несколько товаров (Products).

6. Orders (Заказы)

• Атрибуты:

* ID (PK): Уникальный идентификатор заказа.
* CustomerID (FK): Идентификатор покупателя.
* OrderDate: Дата создания заказа.
* TotalAmount: Общая сумма заказа.
* Status: Статус заказа (например, «обрабатывается», «оплачен», «отменен»).

• Связи:

* Один заказ может быть сделан одним покупателем (Customers).
* Один заказ может иметь несколько платежей (Payments).

7. Payments (Платежи)

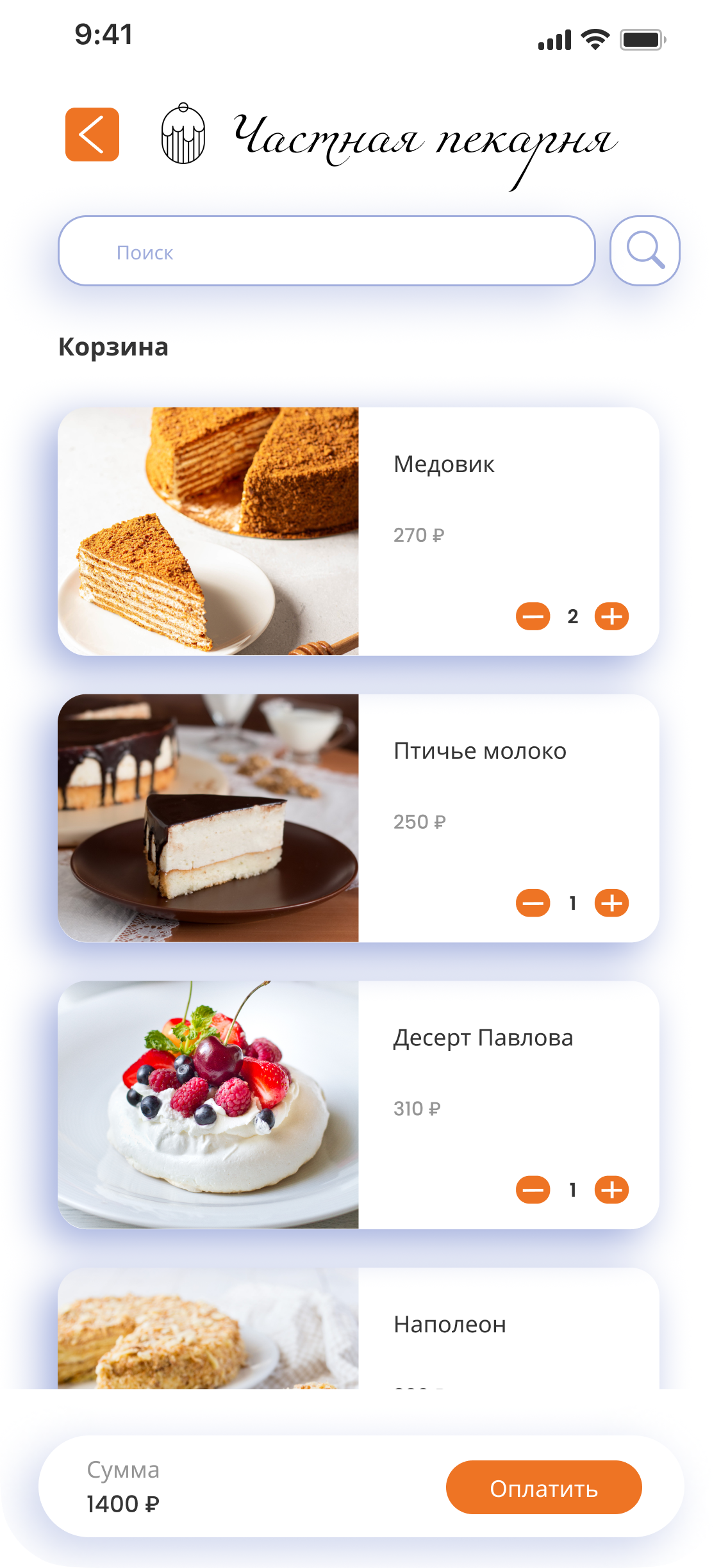
• Атрибуты:

* ID (PK): Уникальный идентификатор платежа.
* OrderID (FK): Идентификатор заказа.
* Amount: Сумма платежа.
* PaymentDate: Дата платежа.
* PaymentMethod: Способ оплаты (например, карта, наличные, онлайн).

• Связи:

* Один платеж связан с одним заказом (Orders).

1. Подготовить прототип одного из экранов данного мобильного приложения и описать пользовательский интерфейс для данного экрана (например, создание заказа).



Прототип экрана для создания заказа в приложении "Частная пекарня" представлен выше. Он включает список товаров с изображениями, названиями, ценами и полями для ввода количества. Внизу размещена кнопка "Оплатить". Дизайн выполнен в приятных тонах, создавая комфортный и дружественный интерфейс для пользователя.

1. Подготовить подробное описание функции редактирования заказа, которую можно было бы использовать в качестве постановки задачи для разработки (помимо текстового описания, использовать UML диаграммы, указать используемые API методы, передаваемые и получаемые параметры, описать процесс хранения информации о покупках пользователя).

#### ****Текстовое описание функции:****

Стандартный процесс редактирования заказа:

Пользователь заходит в свой профиль или раздел с активными заказами.

Выбирает заказ, который хочет отредактировать.

На экране отображаются текущие товары в заказе, их количество и общая сумма.

Пользователь может изменить количество каждого товара, добавить новые товары или удалить текущие.

После внесения изменений система пересчитывает общую сумму и обновляет статус заказа.

После подтверждения изменений заказ сохраняется в базе данных, и пользователь получает обновленный заказ на экране.

#### ****UML-диаграммы****

**Диаграмма классов:**

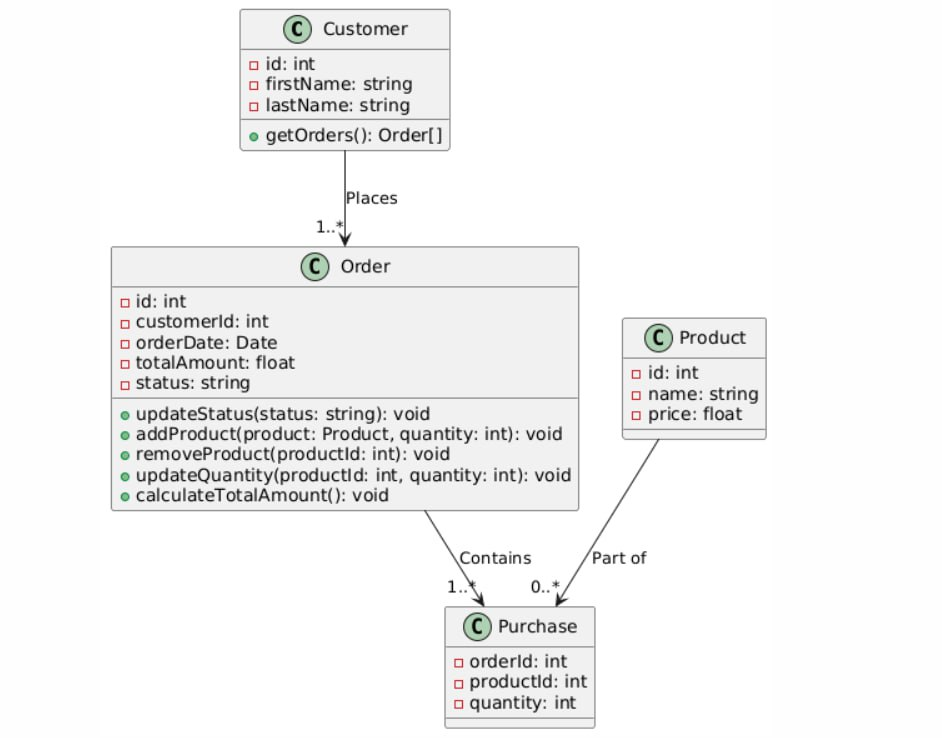
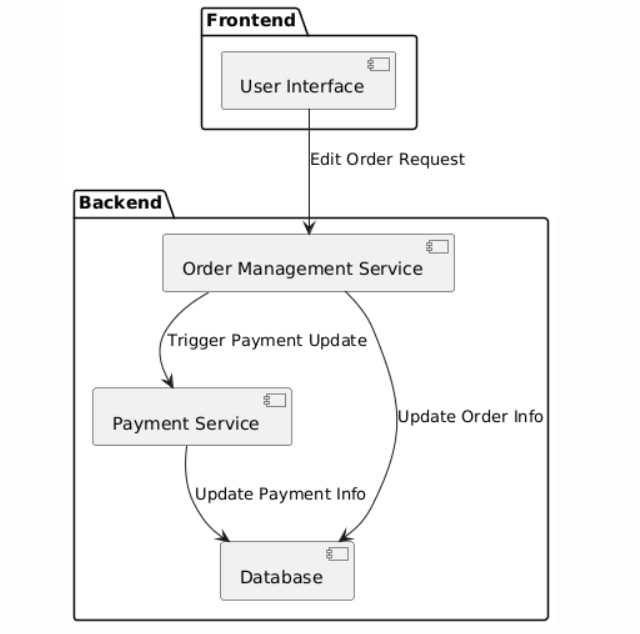
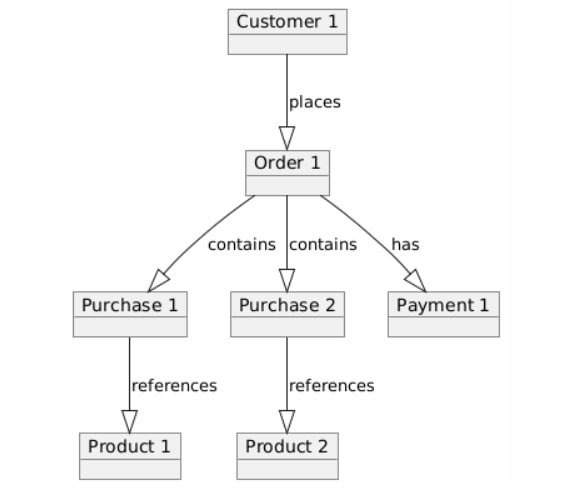


Диаграмма классов описывает систему управления заказами с четырьмя основными сущностями: покупатель (Customer), заказ (Order), товар (Product) и покупка (Purchase).

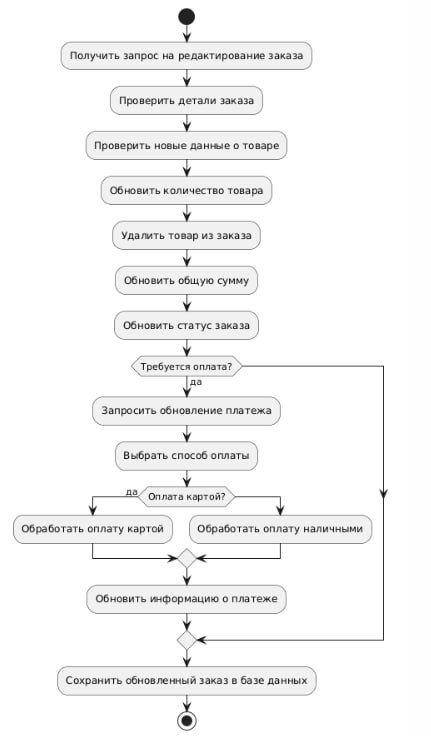
**Диаграмма компонентов:**



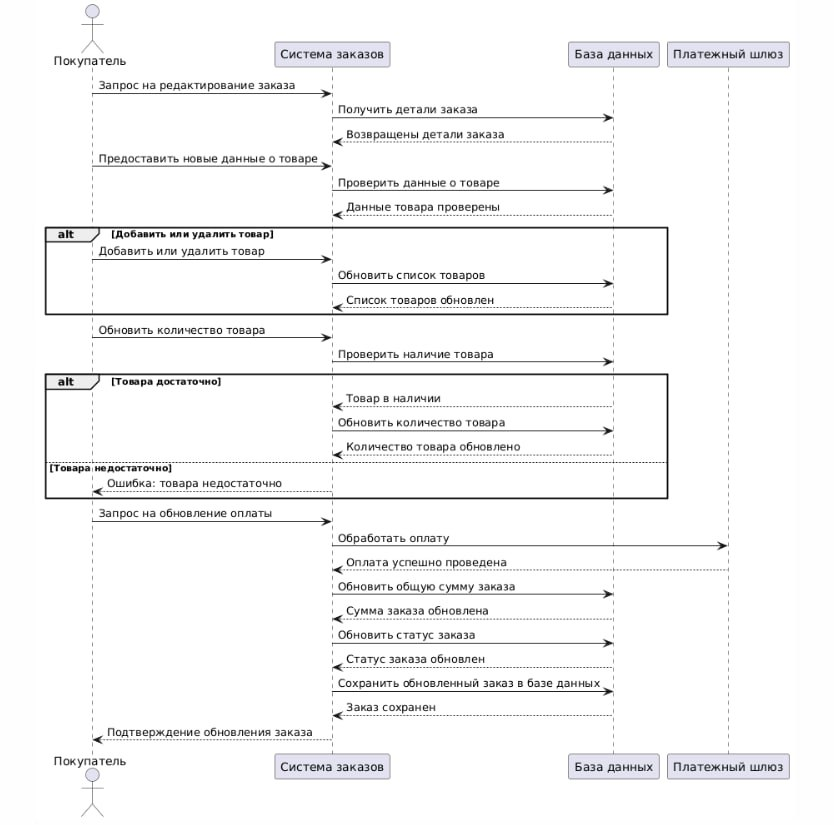
**Диаграмма объектов:**



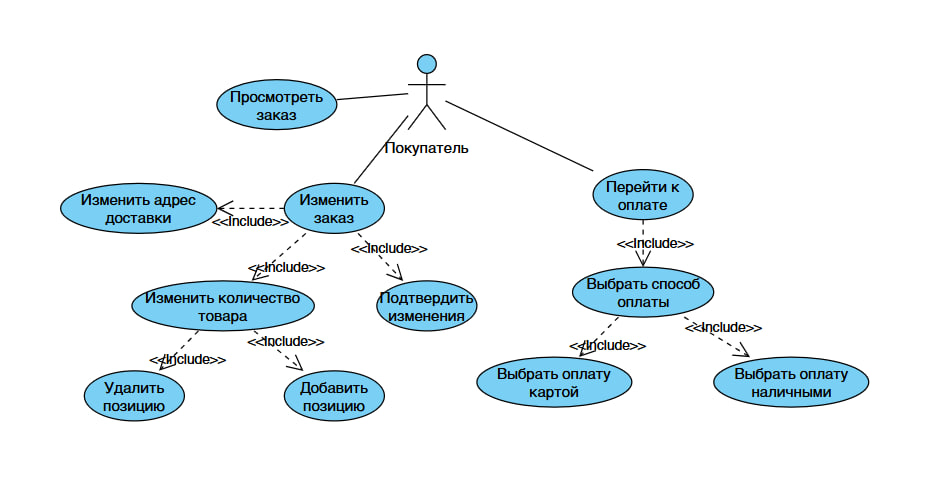
**Диаграмма активности:**



**Диаграмма последовательностей:**



**Диаграмма сценариев:**



**Используемые API методы**

Для реализации функции редактирования заказа могут быть использованы следующие API методы:

Получение деталей заказа:

Метод: GET /orders/{orderId}

Параметры: orderId — уникальный идентификатор заказа.

Ответ: возвращает детали заказа, включая товары, их количество и общую сумму.

Обновление товара в заказе:

Метод: PATCH /orders/{orderId}/items

Параметры: orderId — идентификатор заказа, productId — идентификатор товара, quantity — новое количество товара.

Ответ: подтверждает успешное обновление количества товара в заказе или сообщение об ошибке (например, если товара недостаточно).

Удаление товара из заказа:

Метод: DELETE /orders/{orderId}/items/{productId}

Параметры: orderId — идентификатор заказа, productId — идентификатор удаляемого товара.

Ответ: подтверждает удаление товара из заказа.

Обновление статуса заказа:

Метод: PATCH /orders/{orderId}/status

Параметры: orderId — идентификатор заказа, status — новый статус заказа.

Ответ: подтверждает успешное обновление статуса заказа.

Обновление информации о платеже

Метод: POST /payments/update

Параметры: orderId — идентификатор заказа, paymentMethod — выбранный способ оплаты, amount — сумма.

Ответ: подтверждает успешное обновление информации о платеже.

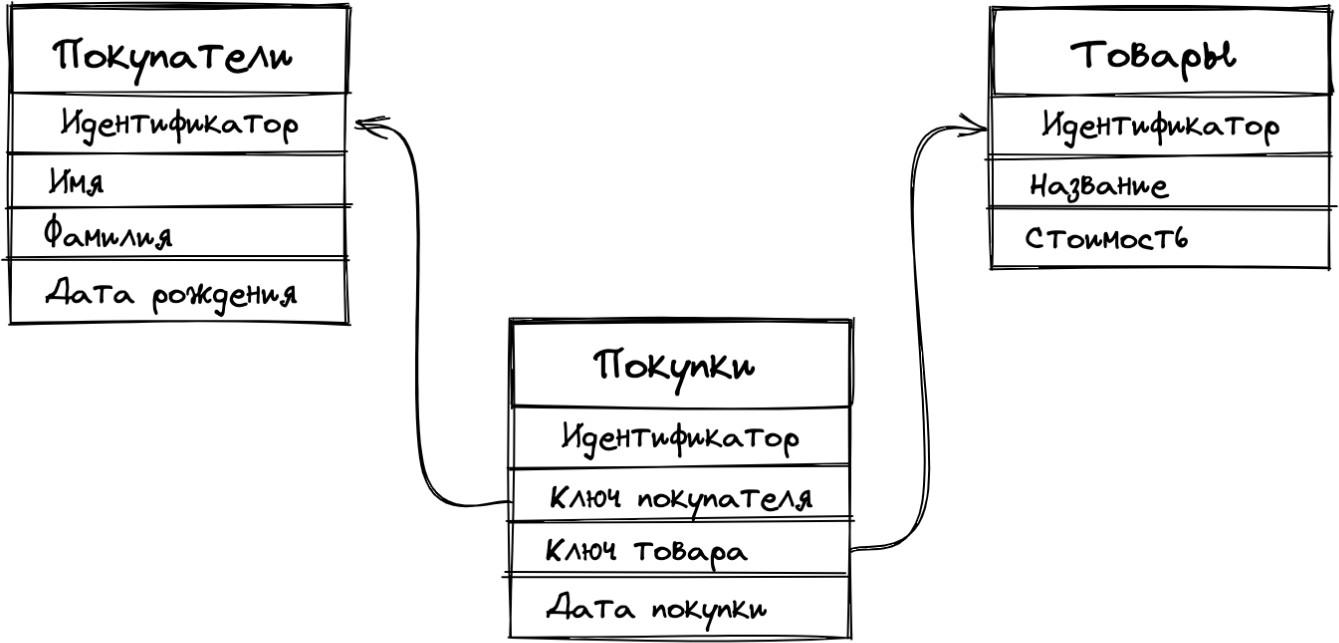
Сохранение обновленного заказа:

Метод: PUT /orders/{orderId}

Параметры: orderId — идентификатор заказа, данные о заказе (товары, сумма, статус и т. д.).

Ответ: подтверждает успешное сохранение обновленного заказа в базе данных.

1. Перед вами реляционная модель данных. Необходимо написать SQL-запросы.



1. Вывести покупателей с количеством осуществленных покупок

Для этого запроса мы можем использовать JOIN между таблицами "Покупатели" и "Покупки" и группировать результаты по каждому покупателю, считая количество покупок.

SELECT Customers.FirstName, Customers.LastName, COUNT(Purchases.ID) AS Purchases\_count

FROM Customers

JOIN Покупки ON Customers.ID = Purchases.CustomersID

GROUP BY Customers.FirstName, Customers.LastName;

1. Общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания

Здесь нам нужно объединить таблицы "Покупатели", "Покупки" и "Товары" для расчета общей стоимости товаров, купленных каждым покупателем. Результат затем отсортируем по общей стоимости в порядке убывания.

SELECT Customers.FirstName, Customers.LastName, SUM(Products.Price) AS General\_cost

FROM Customers

JOIN Purchases ON Customers.ID = Purchases.CustomersID

JOIN Products ON Purchases.ProductsID = Products.ID

GROUP BY Customers.FirstName, Customers.LastName

ORDER BY General\_cost DESC;

1. Получить покупателей, купивших только один товар

Для этого запроса мы используем GROUP BY, но добавляем условие HAVING, чтобы выбрать только тех покупателей, у которых количество покупок равно 1.

SELECT Customers.FirstName, Customers.LastName

FROM Customers

JOIN Purchases ON Customers.ID = Purchases. CustomersID

GROUP BY Customers.FirstName, Customers.LastName

HAVING COUNT(Purchases.ID) = 1;