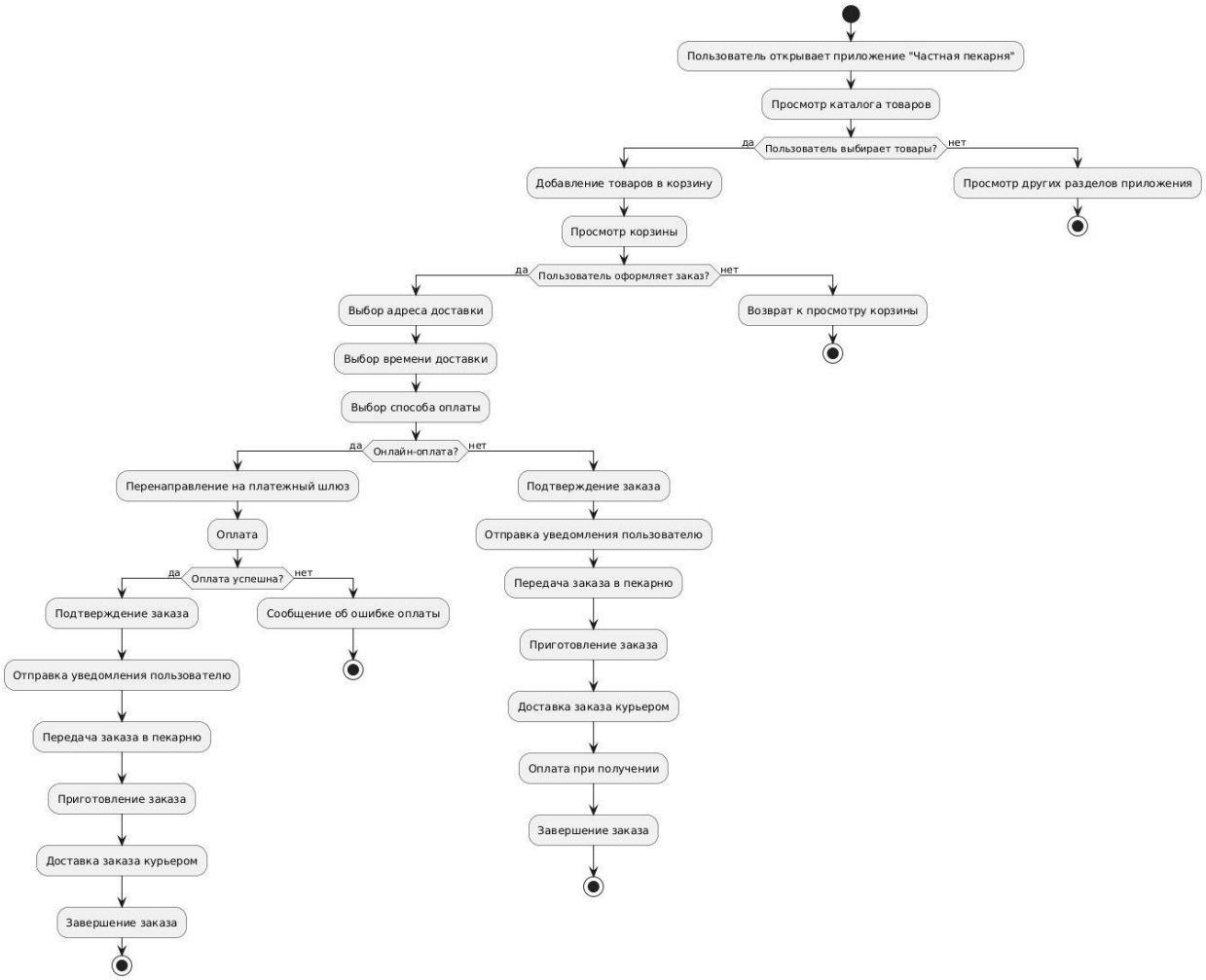
Производственная практика.

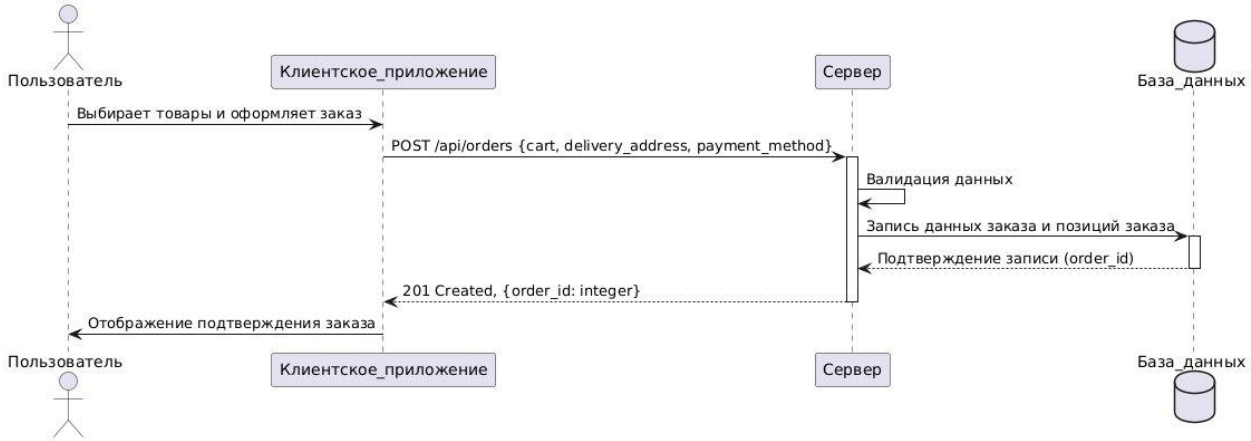
Мобильное клиент-серверное приложение "Частная пекарня" (приложение для продажи выпечки). Аналитика.

Задание 1.

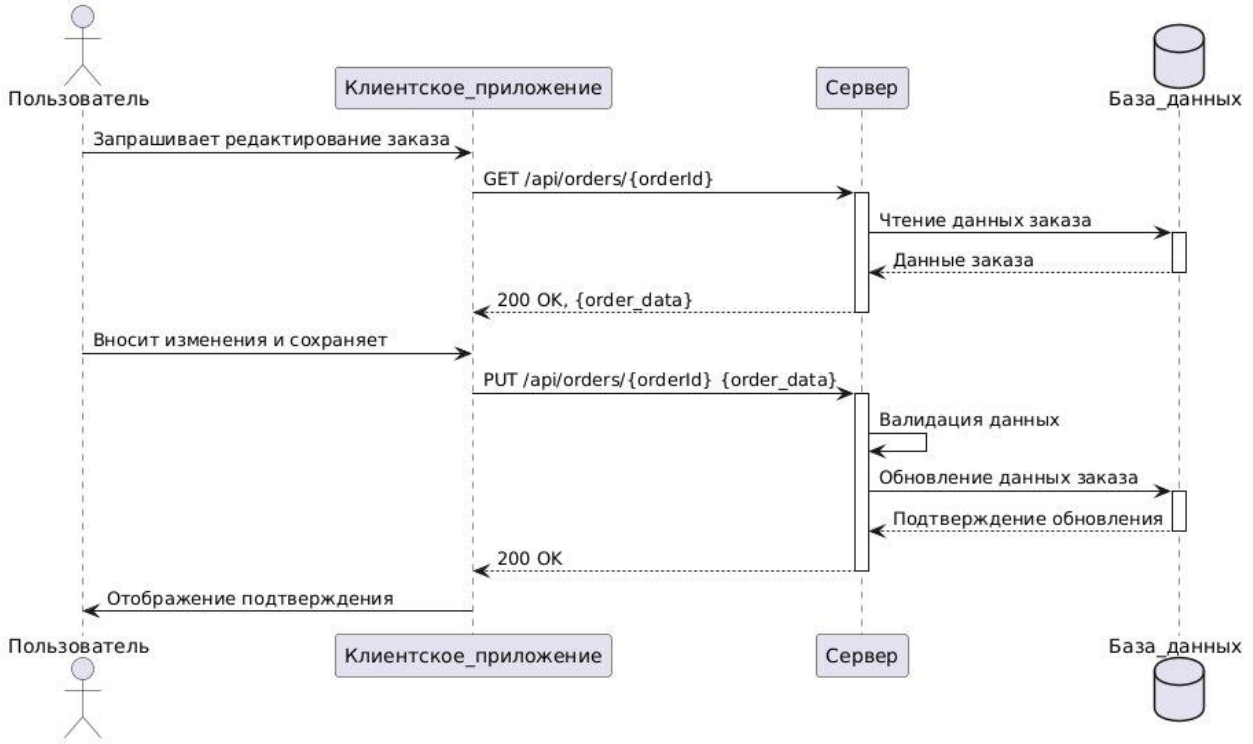


Задание 2.

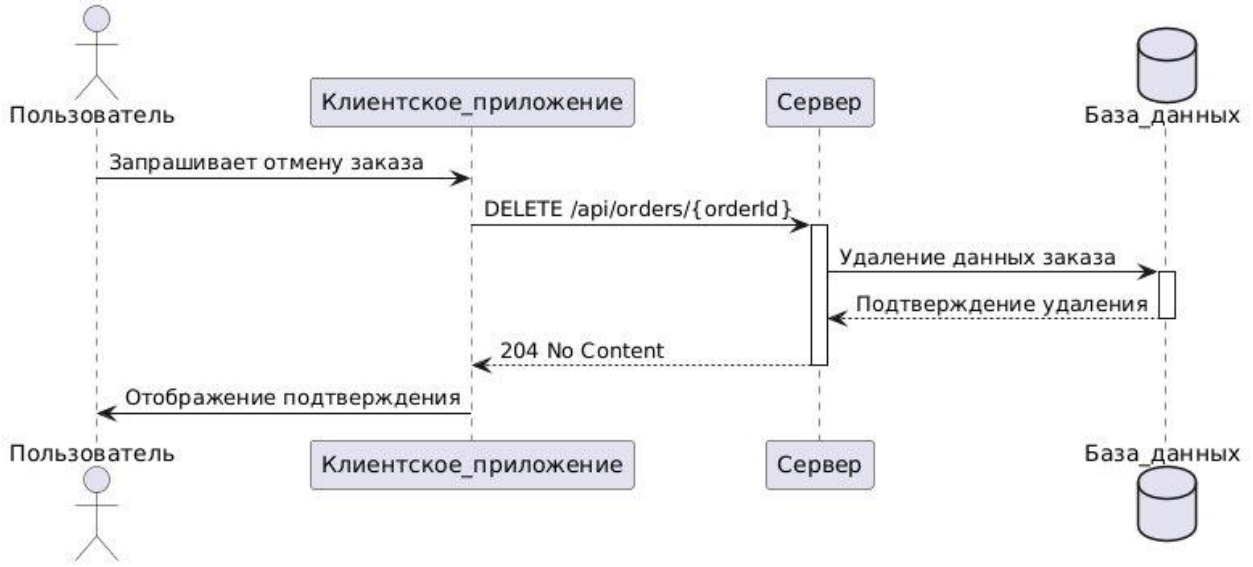
<u>Создание заказа:</u>



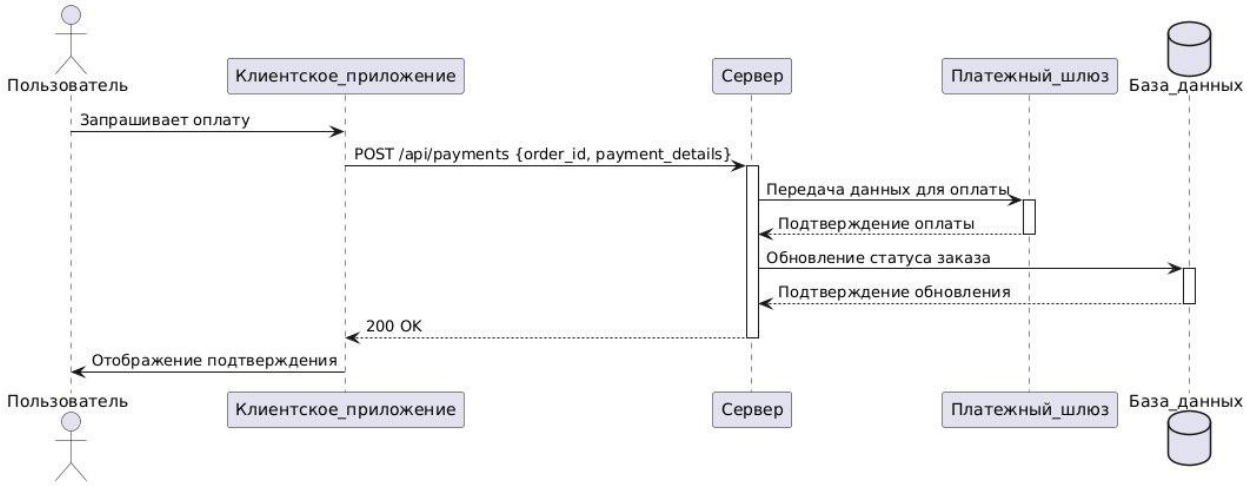
Редактирование заказа:



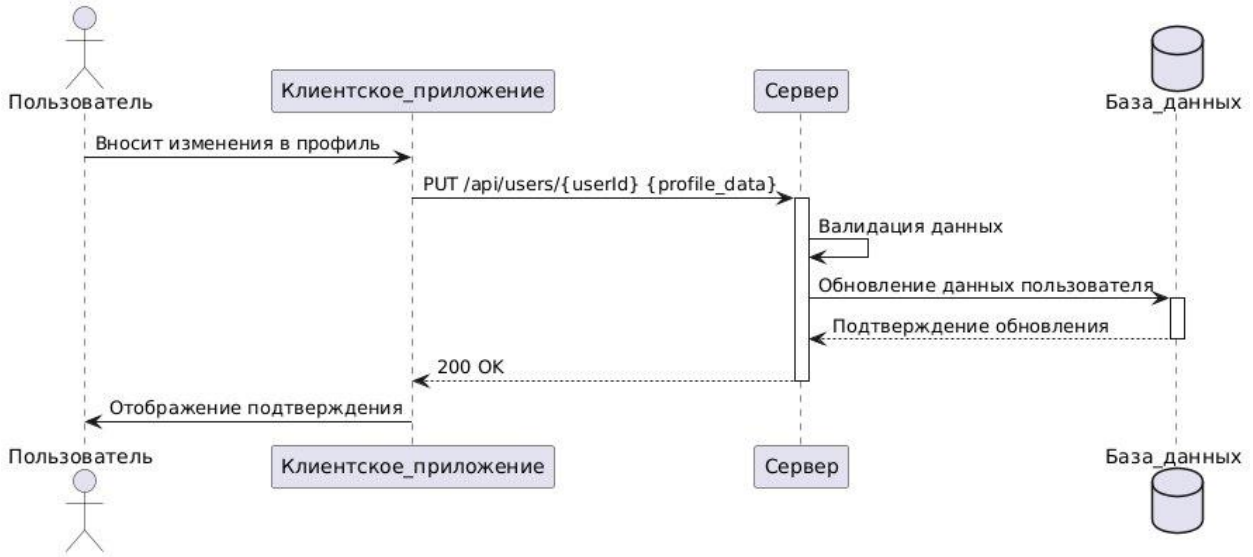
Отмена заказа:



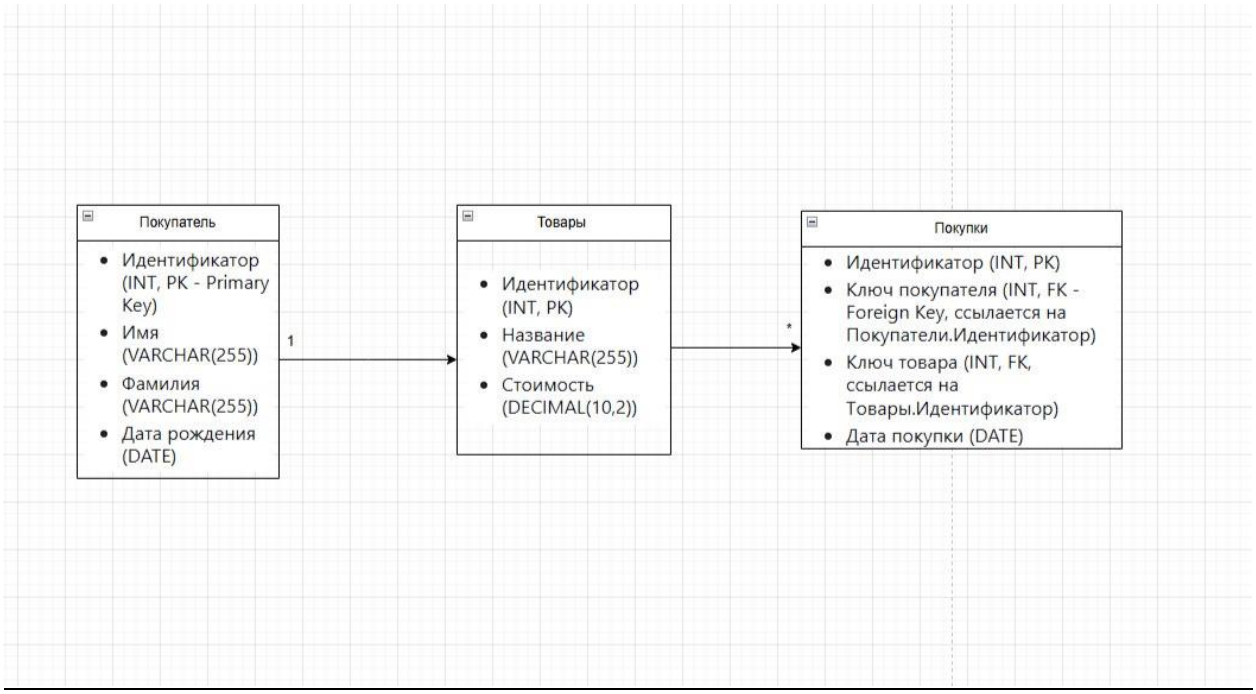
Оплата заказа:



Изменение персональных данных:



UML диаграмма классов:

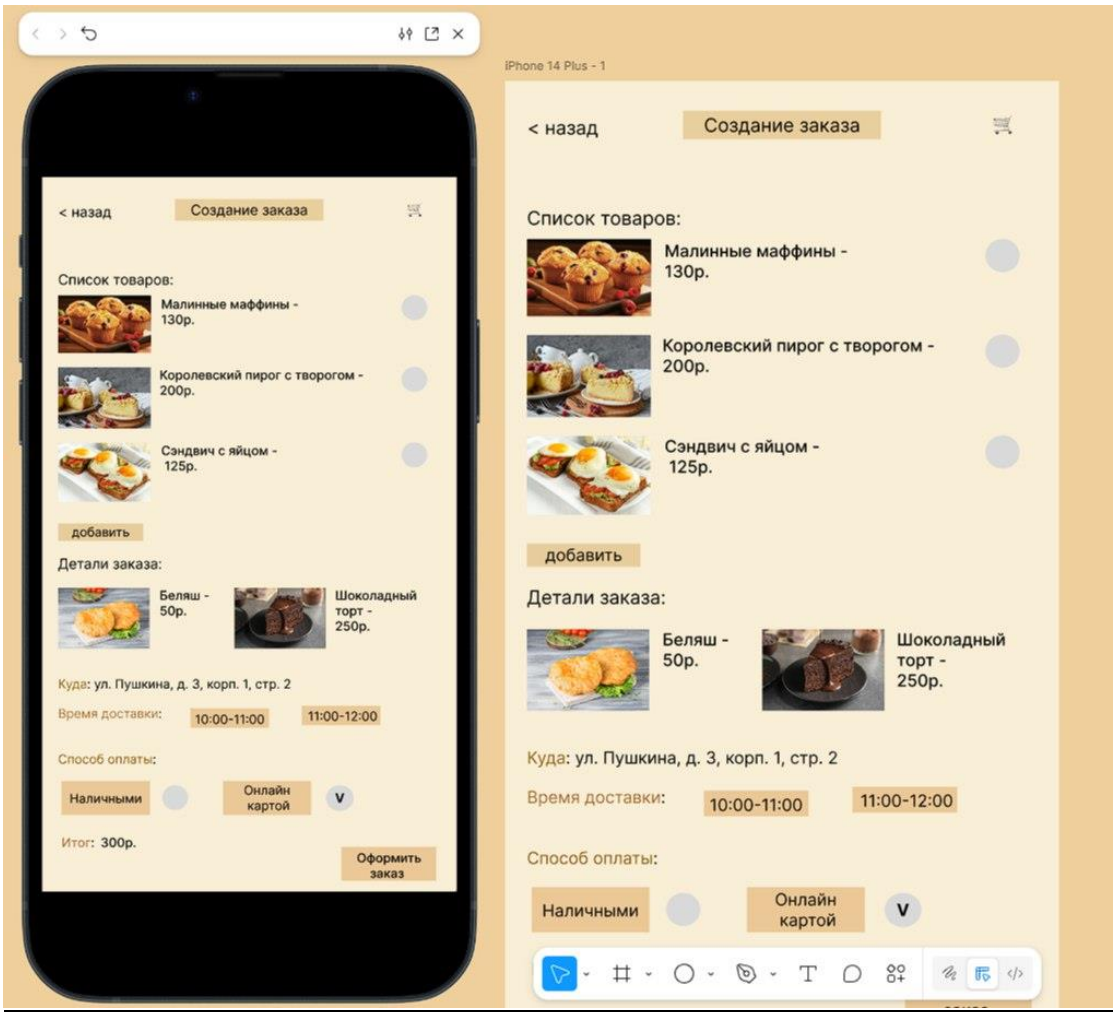


Задание 3.

Прототип одного из экранов данного мобильного приложения.

Его так же можно посмотреть по этой ссылке (делалось на сайте Figma) –

https://www.figma.com/proto/OnhFzb9MpF0SAlBhJ4Mgww/Untitled?node-id=1-2379&p=f&t=DQ6JmvufZswOmKYO-1&scaling=scale-down&content-scaling=fixed&page-id=0%3A1



Описание пользовательского интерфейса экрана “Создание заказа”:

Этот экран представляет собой ключевой элемент мобильного приложения “Частная пекарня”, позволяющий пользователю формировать и отправлять заказы на выбранную выпечку. Он состоит из нескольких основных разделов, каждый из которых выполняет определенную функцию.

1. **Верхняя панель (Header):**
   Кнопка “Назад”: Расположена в левом верхнем углу, позволяет вернуться к предыдущему экрану (например, к каталогу товаров).

Заголовок “Создание заказа”: Центральное отображение названия текущего экрана, обеспечивающее понимание пользователем своего местоположения в приложении.

Иконка корзины: Расположена в правом верхнем углу (возможно), отображает количество товаров в корзине и служит ссылкой для перехода к экрану корзины.

2. **Список товаров:**
   Заголовок “Список товаров”: Четко обозначает раздел, содержащий перечень доступной выпечки.

   Карточки товаров:

   Фотография товара: Визуальное представление продукта.
   Название товара: Текстовое описание продукта.
   Цена товара: Указание стоимости товара.

   Кнопка “Добавить”: Элемент управления, позволяющий добавить товар в корзину. Альтернативой может быть переключатель или поле для ввода количества.

3. **Детали заказа:**
   Заголовок “Детали заказа”: Обозначает раздел, содержащий информацию о выбранных товарах.

4. **Информация о доставке:**
   Заголовок “Куда”: Обозначает поле ввода адреса доставки.
   Заголовок “Время доставки”: Обозначает раздел выбора времени доставки.
   Кнопки выбора времени доставки:
   Два варианта (10:00-11:00 и 11:00-12:00) для выбора пользователем предпочтительного интервала доставки. Выбранный интервал выделяется визуально.

5. **Информация об оплате:**
   Заголовок “Способ оплаты”: Обозначает раздел выбора способа оплаты.
   Кнопки выбора способа оплаты:
   “Наличными”: Выбор оплаты наличными при получении заказа. **“Онлайн картой”:** Выбор онлайн оплаты банковской картой. Выбранный способ оплаты выделяется визуально (галочка V).

6. **Итоговая информация и подтверждение заказа:**
   “Итог: …”: Текстовое поле, отображающее общую стоимость заказа с учетом выбранных товаров и доставки.

Кнопка “Оформить заказ”: Основная кнопка действия, запускающая процесс оформления заказа и отправки его в систему.

Общее впечатление: Интерфейс выглядит простым и интуитивно понятным. Используются четкие заголовки и подписи, облегчающие навигацию. Визуальное представление товаров с помощью фотографий способствует более осознанному выбору. Интерактивные элементы (кнопки выбора времени доставки и способа оплаты) позволяют пользователю легко настраивать параметры заказа.

Задание 4.

Описание функции редактирования заказа в мобильном приложении “Частная пекарня”.

1. Введение:
Эта спецификация описывает функцию редактирования заказа, которая позволит пользователям изменять состав заказа, адрес и время доставки после его создания, но до момента, когда заказ переходит в стадию “В обработке” (или другую аналогичную стадию, означающую, что заказ уже готовится/передан на доставку).

2. Цель:
Предоставить пользователям гибкость в управлении своими заказами, позволяя им корректировать детали заказа до того, как он будет окончательно обработан пекарней.

3. Пользователь:
Клиент (зарегистрированный пользователь мобильного приложения)

4. Предусловия:
- Пользователь авторизован в приложении.
- Заказ существует в системе и находится в статусе “Создан” или “Ожидает подтверждения” (или другом статусе, разрешающем редактирование).

5. Основные шаги (Use Case):

1. **Выбор заказа:** Пользователь переходит в раздел “История заказов” в приложении и выбирает конкретный заказ, который хочет отредактировать.

2. **Отображение деталей заказа:** Система отображает детали выбранного заказа, включая список товаров, адрес доставки, время доставки, способ оплаты и общую стоимость.

3. **Инициация редактирования:** Пользователь нажимает кнопку “Редактировать заказ”.

4. **Загрузка данных для редактирования:** Система получает данные о заказе через API.

5. **Отображение формы редактирования:** Система отображает форму, позволяющую пользователю изменить следующие параметры:

Список товаров:

- Добавление новых товаров (переход в каталог).

- Изменение количества товаров (увеличение/уменьшение).

- Удаление товаров из заказа.

Адрес доставки:

- Выбор из сохраненных адресов (если есть)

- Редактирование текущего адреса

- Добавление нового адреса

Время доставки:

Выбор даты и времени доставки.

1.**Ввод изменений:** Пользователь вносит изменения в выбранные параметры.

2. **Подтверждение изменений:** Пользователь нажимает кнопку “Сохранить изменения”.

3. **Валидация данных:** Система выполняет валидацию введенных данных (например, проверка адреса, доступности времени доставки).

4. **Обновление данных заказа:** Система отправляет запрос на сервер для обновления данных заказа.

5. **Обратная связь:**

- Успех: После успешного обновления данных система отображает сообщение об успешном сохранении изменений и обновленную информацию о заказе.

- Ошибка: Если при сохранении произошла ошибка (например, недоступность товара, ошибка валидации, проблемы с сетью), система отображает сообщение об ошибке и предоставляет информацию о причине ошибки.

6. UML-диаграммы:Диаграмма вариантов использования (Use Case Diagram):

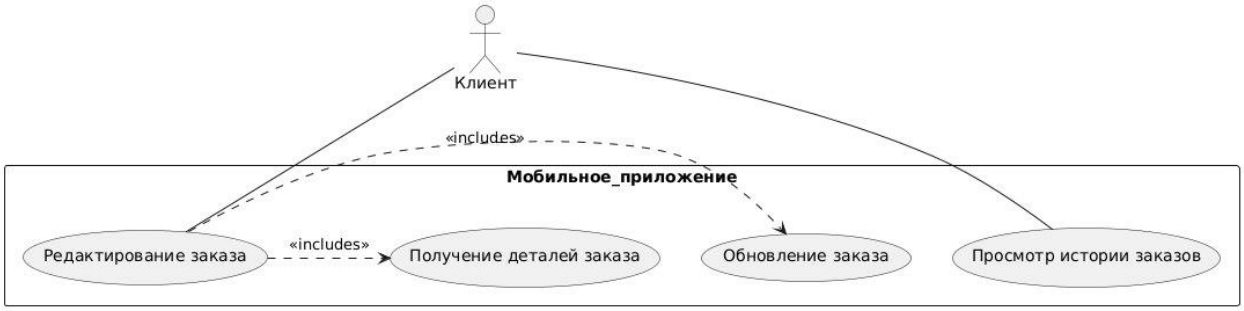
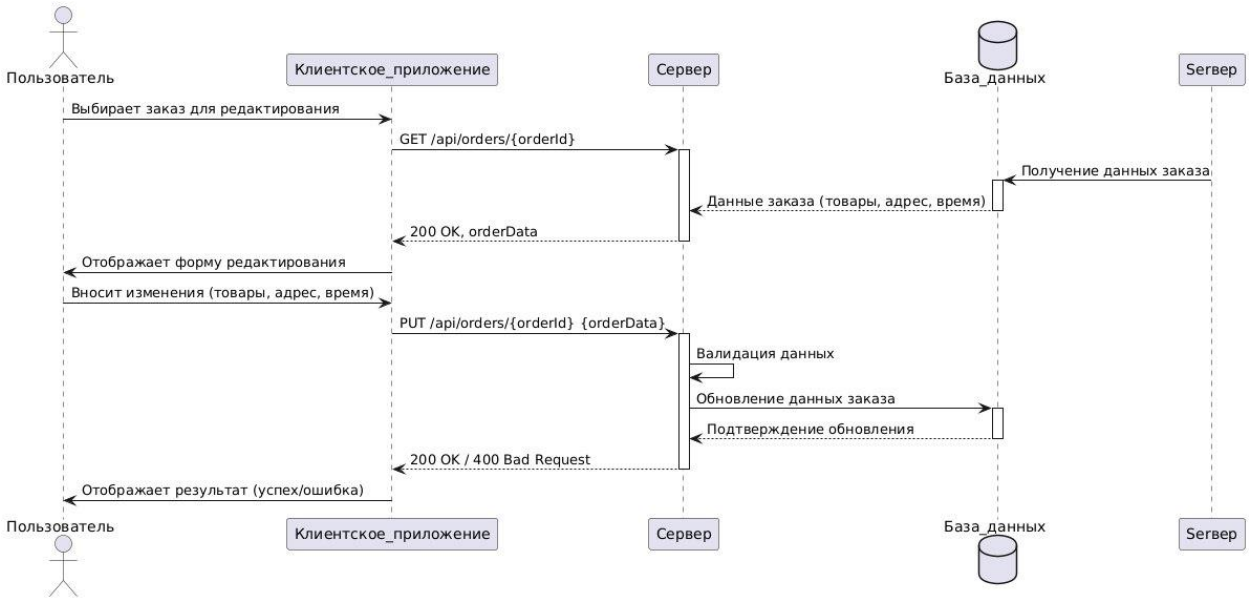


Диаграмма последовательности (Sequence Diagram):



7. API Методы:

GET /api/orders/{orderId}: Получение данных заказа для редактирования.

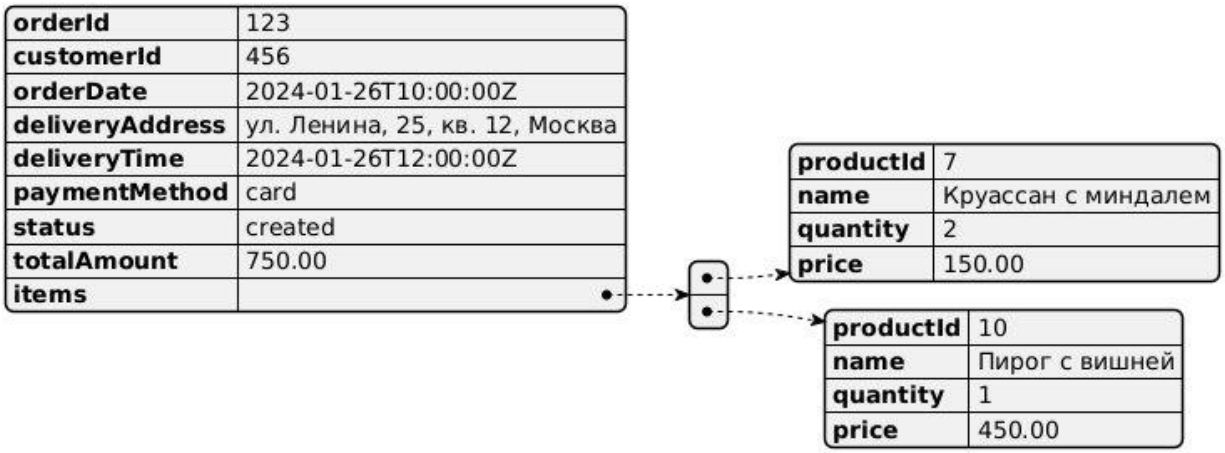
Метод: GET

URL: /api/orders/{orderId} (где {orderId} — идентификатор заказа)

Запрос (Request): Ничего (no request body)

Ответ (Response):

Успех (200 OK):



Ошибка (404 Not Found): Если заказ не найден.

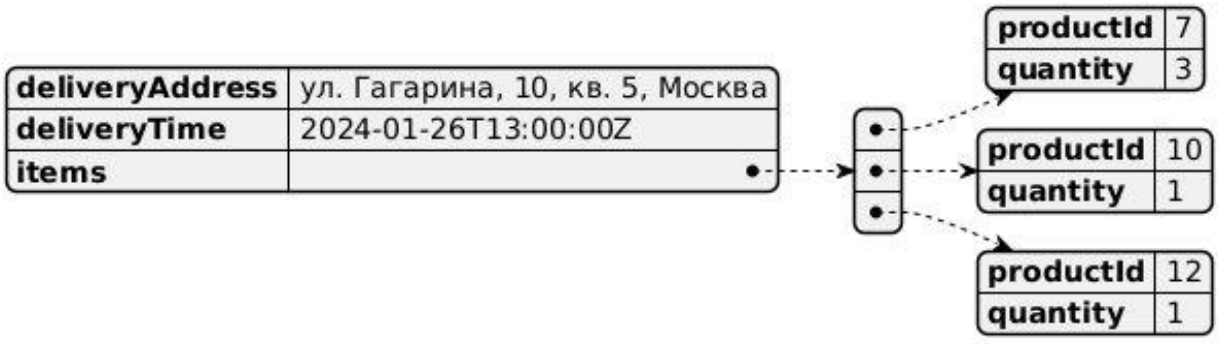
Ошибка (403 Forbidden): Если заказ нельзя редактировать (неверный статус).

PUT /api/orders/{orderId}: Обновление данных заказа.

Метод: PUT

URL: /api/orders/{orderId} (где {orderId} — идентификатор заказа)

Запрос (Request):



Ответ (Response):

Успех (200 OK): Заказ успешно обновлен.

Ошибка (400 Bad Request): Ошибка валидации (например, неверный адрес, недоступное время доставки, недостаточный товар). В ответе должно быть подробное описание ошибок.

8. Параметры, передаваемые и получаемые:

GET /api/orders/{orderId}: Параметры не передаются. Получаем данные заказа (см. пример ответа).

PUT /api/orders/{orderId}:

- **Передаваемые параметры (request body):** deliveryAddress, deliveryTime, items (массив объектов, описывающих товары в заказе).

- **Получаемые параметры (response body):** (Успех) ничего (200 OK). (Ошибка) JSON с описанием ошибок.

9. Хранение информации о покупках пользователя:

Таблицы в базе данных (ER-диаграмма, см. предыдущие разделы):

Customers (Покупатели): Хранит информацию о пользователях.

- customerId (PK)
- … (другие атрибуты: имя, телефон, адрес и т.д.)

Orders (Заказы): Хранит информацию о заказах.

- orderId (PK)
- customerId (FK, ссылается на Customers.customerId)
- orderDate
- deliveryAddress
- deliveryTime
- paymentMethod
- status
- totalAmount

OrderItems (ПозицииЗаказа): Хранит информацию о товарах в заказе.

- orderItemId (PK)
- orderId (FK, ссылается на Orders.orderId)
- productId (FK, ссылается на Products.productId)
- quantity
- price

Процесс хранения информации:

Когда пользователь оформляет заказ, создается новая запись в таблице Orders. В эту запись записывается информация о заказе (адрес, время доставки, способ оплаты, общая сумма, customerId).

Для каждого товара в заказе создается запись в таблице OrderItems. В эту запись записывается orderId, productId, quantity, price.

При редактировании заказа:

- Обновляется существующая запись в таблице Orders. Обновляются поля deliveryAddress, deliveryTime и другие соответствующие поля.
- Удаляются существующие записи из таблицы OrderItems для данного заказа.
- Создаются новые записи в таблице OrderItems для новых позиций заказа (или обновляются существующие, если изменилось количество товаров).

Пользовательская история заказов:

- Для отображения истории заказов пользователя, система использует запросы к таблицам Orders и OrderItems.
- Запрос к Orders с фильтром по customerId позволяет получить список заказов пользователя.
- Для каждого заказа из списка система выполняет запрос к OrderItems, чтобы получить информацию о товарах в заказе.

10. Дополнительные требования:

Безопасность: Доступ к API методам должен быть защищен (например, аутентификация по токену).

Валидация: Сервер должен выполнять строгую валидацию входных данных, чтобы предотвратить ошибки и обеспечить целостность данных.

Обработка ошибок: При возникновении ошибок система должна отображать понятные сообщения об ошибках для пользователя.

Производительность: API методы должны быть оптимизированы для быстрой работы.

Уведомления: После успешного редактирования заказа пользователю может быть отправлено уведомление (например, push-уведомление).

Логирование: Необходимо логирование всех операций редактирования заказа (для отслеживания изменений и решения проблем).

Ограничения: Редактирование заказа не должно быть доступно, если заказ находится в статусе, который уже подразумевает начало обработки заказа (например, “В обработке”, “Готовится”).

Задание 5.

Вот SQL-запросы для решения постановленных задач:

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

SELECT
Покупатели

LEFT JOIN
Покупки **ON** Покупатели. Идентификатор = Покупки. КлючПокупателя

GROUP BY
Покупатели. Идентификатор, Покупатели. Имя, Покупатели. Фамилия

ORDER BY
КоличествоПокупок DESC;

2. **Общая стоимость товара для каждого покупателя и отсортировать результат в порядке убывания:**

SELECT

Покупатели. Имя,

Покупатели. Фамилия,

SUM (Товары. Стоимость) AS ОбщаяСтоимость

FROM

Покупатели

JOIN

Покупки ON Покупатели. Идентификатор = Покупки. КлючПокупателя

JOIN

Товары ON Покупки. КлючТовара = Товары. Идентификатор

GROUP BY

Покупатели. Идентификатор, Покупатели. Имя, Покупатели. Фамилия

GROUP BY

ОбщаяСтоимость DESC;

3. <u>Получить покупателей, получивших только один товар:</u>

SELECT

Покупатели. Имя,

Покупатели. Фамилия

FROM

Покупатели

WHERE

Покупатели. Идентификатор IN (

SELECT

Покупки. КлючПокупателя

FROM

Покупки

GROUP BY

Покупки. КлючПокупателя

HAVING

COUNT (DISTINCT Покупки. КлючТовара) = 1);

Пояснения.

Запрос 1:

Использует LEFT JOIN для учёта покупателей, не совершивших ни одной покупки (у них количество покупок будет 0).

COUNT (Покупки.Идентификатор) подсчитывает количество покупок для каждого покупателя.

GROUP BY группирует результаты по идентификатору покупателя, имени и фамилии.

ORDER BY сортирует результат по убыванию количества покупок.

Запрос 2:

Использует JOIN для объединения таблиц “Покупатели”, “Покупки” и “Товары”.

SUM (Товары.Стоимость) подсчитывает общую стоимость товаров для каждого покупателя.

Важно: Этот запрос предполагает, что каждая запись в таблице **Покупки** соответствует покупке *одного* товара. Если в таблице **Покупки** может быть несколько товаров, то необходимо добавить таблицу, связывающую “Покупки” и “Товары” (как таблица OrderItems в предыдущем ответе) и изменить запрос.

GROUP BY группирует результаты по идентификатору покупателя, имени и фамилии.

ORDER BY сортирует результат по убыванию общей стоимости.

Запрос 3:

Использует подзапрос для выбора идентификаторов покупателей, купивших только один *разный* товар (даже если они покупали его несколько раз, но это был *один и тот же*товар).

COUNT(DISTINCT Покупки.КлючТовара) подсчитывает количество *разных* товаров, купленных каждым покупателем. Если требуется найти покупателей, которые сделали *только одну*покупку (независимо от того,

сколько товаров они купили в этой покупке), то следует упростить запрос, убрав

DISTINCT и условие COUNT(Покупки.Идентификатор) = 1.

Внешний запрос выбирает информацию о покупателях, чьи идентификаторы были возвращены подзапросом.

Важно: Эти запросы написаны на основе *предположения* о структуре вашей базы данных, основанного на визуальном представлении. Необходимо адаптировать запросы к *фактической* структуре ваших таблиц и полей. Протестируйте запросы на вашей базе данных, чтобы убедиться в их корректной работе.