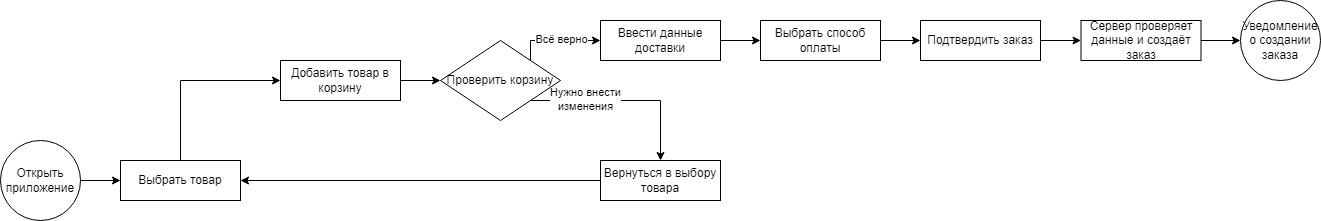
**1. Бизнес-процесс создания заказа.**

Использована упрощенная версия нотации BPMN (Business Process Model and Notation).

**Описание процесса:**

1. Начало процесса: Пользователь открывает приложение.
2. Выбор товаров: Пользователь просматривает меню и выбирает желаемые товары.
3. Добавление в корзину: Пользователь добавляет товары в корзину.
4. Просмотр корзины: Пользователь переходит к корзине для проверки выбранных товаров.
5. Создание заказа: Пользователь нажимает на кнопку "Оформить заказ".
6. Ввод данных: Пользователь вводит свои персональные данные (имя, адрес, телефон).
7. Оплата: Пользователь выбирает способ оплаты и завершает процесс.
8. Подтверждение заказа: Получение подтверждения о создании заказа.
9. Конец процесса: Заказ успешно создан, пользователь может вернуть к меню или выйти из приложения.

[](https://github.com/drgchft/Analitika1/blob/main/BPMN%20%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0.drawio%20(1).png)

[**BPMN диаграмма**](https://github.com/drgchft/Analitika1/blob/main/BPMN%20%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0.drawio%20(1).png)

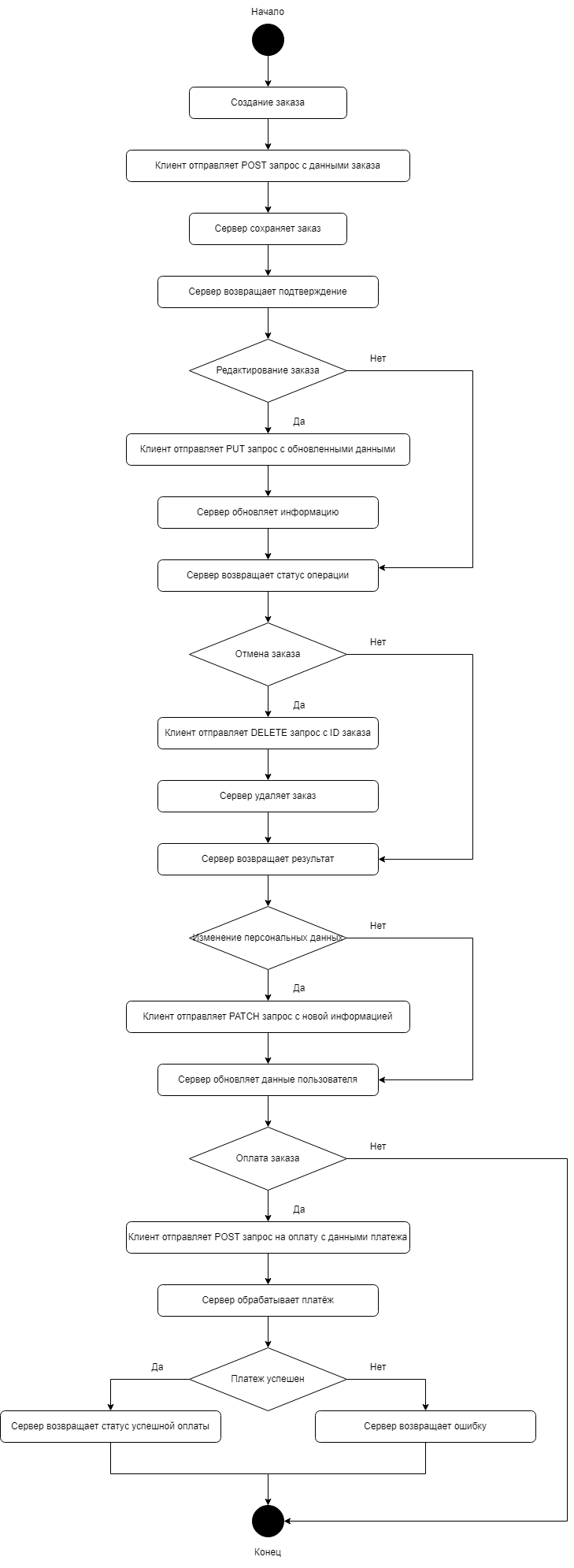
**2. Процесс синхронизации данных между клиентом и сервером**

**Процесс:**

1. Создание заказа: Клиент отправляет POST запрос на сервер с данными заказа. Сервер сохраняет заказ и возвращает подтверждение.
2. Редактирование заказа: Клиент отправляет PUT запрос с обновленными данными заказа. Сервер обновляет информацию и возвращает статус операции.
3. Отмена заказа: Клиент отправляет DELETE запрос с идентификатором заказа. Сервер удаляет заказ и возвращает результат.
4. Изменение персональных данных: Клиент отправляет PATCH запрос с обновленной информацией. Сервер обновляет данные пользователя.
5. Оплата заказа: Клиент отправляет POST запрос на оплату, включая данные платежа. Сервер обрабатывает платеж и возвращает статус (успешно или ошибка).

**UML диаграмма:**

[**UML диаграмма синхронизация**](https://github.com/drgchft/Analitika1/blob/main/UML%20%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F.drawio.png)

[](https://github.com/drgchft/Analitika1/blob/main/UML%20%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F.drawio.png)**API методы:**

POST /orders — создание заказа.

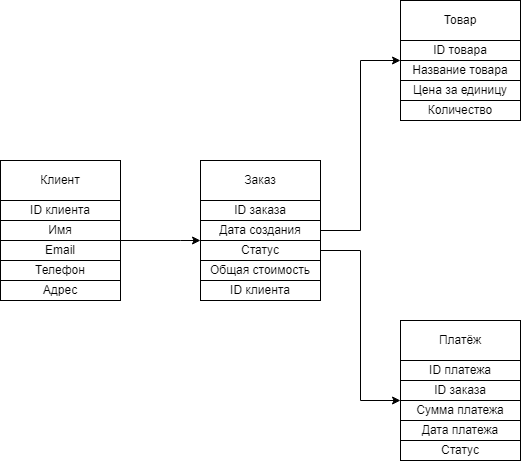
PUT /orders/{id} — редактирование заказа.

DELETE /orders/{id} — отмена заказа.

PATCH /users/{id} — изменение персональных данных.

POST /payments — оплата заказа.

**ER-диаграмма сущностей:**

[](https://github.com/drgchft/Analitika1/blob/main/ER%20%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0.drawio.png)

[ER диаграмма](https://github.com/drgchft/Analitika1/blob/main/ER%20%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0.drawio.png)

**3. Прототип экрана (Главный экран)**

**Описание пользовательского интерфейса:**

Главная страница приложения «Пекарня» представлена структурой, которая легко ориентирует пользователя и позволяет быстро выбрать нужные товары или услуги.

1. Город в верхней части экрана: В верхней части приложения находится строка с названием города, в котором работает пекарня. Это может быть полезно для пользователей, чтобы убедиться, что они находятся в правильном регионе.
2. Выбор способа доставки: Под названием города расположена область, в которой пользователь может выбрать способ получения заказа. Здесь предусмотрены две опции:

* Доставка — для тех, кто хочет получить заказ на дом.
* В зале — если пользователь планирует забрать свой заказ самостоятельно в пекарне.

1. Адрес: Ниже области выбора доставки указывается актуальный адрес пекарни. Это может быть либо общий адрес магазина, либо адрес с уточнением для доставки.
2. Акции: После указания адреса идет область с текущими акциями пекарни. Здесь можно увидеть привлекательные предложения, скидки или бонусы.
3. Новое и популярное: Далее идет блок, в котором отображаются товары, которые стали новыми или приобрели популярность среди пользователей. Это могут быть специальные предложения или новинки пекарни.
4. Горизонтальная лента с категориями меню: Под разделом «Новое и популярное» располагается горизонтальная лента с категориями меню. Например, «Комбо», «Пирожное», «Булочки» и т. д.
5. Пример комбо: Ниже категорий можно увидеть блок с предложением «Комбо», где представлены наборы товаров по сниженной цене.
6. Навигационная панель внизу: В самом низу экрана расположена панель навигации, которая включает:

* Меню — для перехода в основное меню с категориями товаров.
* Профиль — для доступа к личным данным, истории заказов, настройкам.
* Контакты — для быстрого доступа к информации о пекарне, телефонным номерам и адресам.
* Корзина — для перехода в корзину и оформления заказа.

[****](https://github.com/drgchft/Analitika1/blob/main/%D0%9F%D1%80%D0%BE%D0%BE%D1%82%D0%B8%D0%BF%20%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0.png)

[**Прототип (Главный экран)**](https://github.com/drgchft/Analitika1/blob/main/%D0%9F%D1%80%D0%BE%D0%BE%D1%82%D0%B8%D0%BF%20%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0.png)

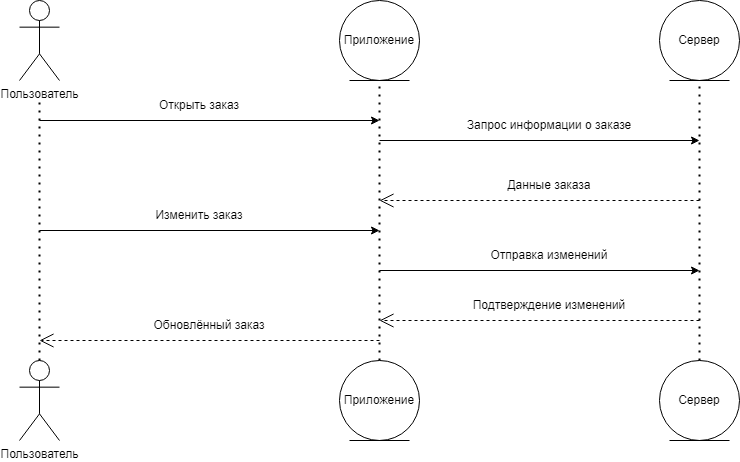
**4. Описание функции редактирования заказа**

**Описание:** Функция редактирования заказа предназначена для обеспечения пользователю возможности изменения деталей уже оформленного, но еще не подтвержденного или не выполненного заказа. Редактирование заказа недоступно, когда курьер уже в пути и статус заказа “готов” при заказе в зале. Изменения касаются: списка товаров в заказе (добавление/удаление позиций, изменение количества); адреса и времени доставки; комментариев к заказу. К полям “адрес” и “время доставки” действует ограничение в редактировании, адрес можно изменить только через службу поддержки, а время можно устанавливать не раньше чем через час. После редактирования заказ обновляется на сервере, и пользователь видит актуальную информацию.

**Процесс:**

1. Запрос данных о заказе: Клиент отправляет запрос к серверу для получения информации о заказе. Сервер возвращает информацию о составе заказа, его статусе и деталях.
2. Изменение данных пользователем: Пользователь может: удалять/добавлять товары из заказа; изменять количество уже добавленных товаров; обновлять адрес и время доставки; оставить комментарий.
3. Валидация изменений на клиентской стороне: Проверка на корректность данных.
4. Отправка изменений на сервер: Клиент формирует запрос с обновленными данными и отправляет его на сервер.
5. Обновление информации на сервере: Сервер обрабатывает запрос, обновляет данные о заказе в базе данных, пересчитывает итоговую стоимость и возвращает результат клиенту.
6. Отображение обновленных данных: На клиенте отображаются подтвержденные сервером изменения.

**UML диаграмма:**

[****](https://github.com/drgchft/Analitika1/blob/main/UML%20%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%B8%D1%80%D0%B2%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5.drawio.png)

[**UML диаграмма редактирование**](https://github.com/drgchft/Analitika1/blob/main/UML%20%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%B8%D1%80%D0%B2%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5.drawio.png)

**API методы:**

1. Получение информации о заказе.

GET/orders/{id}

1. Редактирование заказа.

PUT/orders/{id}

1. Валидация доступности товаров.

POST/inventory/check

**Хранение информации о покупках пользователя.**

Информация хранится в базе данных:

1. Таблица пользователей (users): хранит информацию о клиентах. При регистрации или добавлении нового пользователя его данные сохраняются в таблице users.
2. Таблица заказов (orders): хранит информацию о заказах. Когда пользователь делает покупку, создается запись в таблице orders, связанная с его идентификатором (user\_id). Эта запись фиксирует общий заказ, но еще не содержит подробностей о товарах.
3. Таблица товаров в заказе (order\_items): хранит список товаров, связанных с заказом. Каждый заказ может включать несколько товаров. Информация о них сохраняется в таблице order\_items, которая связывает заказы и товары.
4. Таблица товаров (products): хранит данные о товарах, доступных для заказа.

**5. SQL-запросы**

1. Вывести покупателей с количеством осуществленных покупок:

SELECT

Покупатели.Идентификатор AS Покупатель\_ID,

Покупатели.Имя,

Покупатели.Фамилия,

COUNT(Покупки.Идентификатор) AS Количество\_покупок

FROM

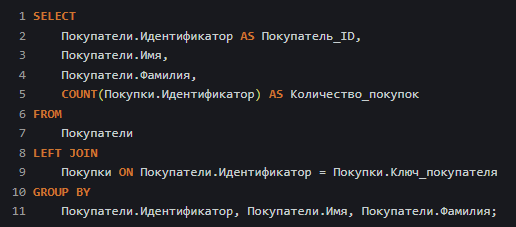
Покупатели

LEFT JOIN

Покупки ON Покупатели.Идентификатор = Покупки.Ключ\_покупателя

GROUP BY

Покупатели.Идентификатор, Покупатели.Имя, Покупатели.Фамилия;



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

SELECT

Покупатели.Идентификатор AS Покупатель\_ID,

Покупатели.Имя,

Покупатели.Фамилия,

SUM(Товары.Стоимость) AS Общая\_стоимость

FROM

Покупатели

JOIN

Покупки ON Покупатели.Идентификатор = Покупки.Ключ\_покупателя

JOIN

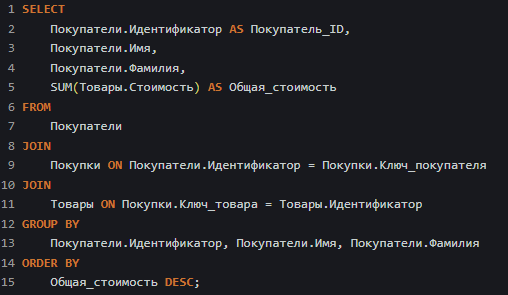
Товары ON Покупки.Ключ\_товара = Товары.Идентификатор

GROUP BY

Покупатели.Идентификатор, Покупатели.Имя, Покупатели.Фамилия

ORDER BY

Общая\_стоимость DESC;



1. Получить покупателей, купивших только один товар:

SELECT

Покупатели.Идентификатор AS Покупатель\_ID,

Покупатели.Имя,

Покупатели.Фамилия

FROM

Покупатели

JOIN

Покупки ON Покупатели.Идентификатор = Покупки.Ключ\_покупателя

GROUP BY

Покупатели.Идентификатор, Покупатели.Имя, Покупатели.Фамилия

HAVING

COUNT(Покупки.Ключ\_товара) = 1;

