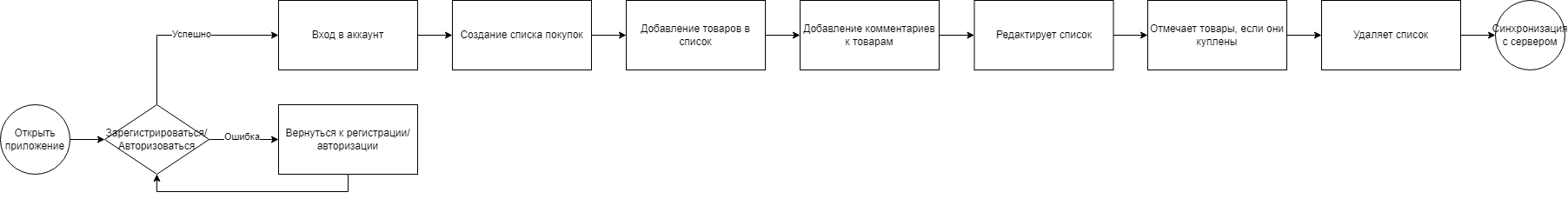
**Дано:** Мобильное клиент-серверное приложение «Список покупок в магазинах» (Примечание: приложение без интеграции с магазинами).

**1. Бизнес-логика мобильного приложения.**

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

**Описание логики:**

1. Регистрация/Авторизация: Пользователь создает аккаунт или входит в систему.
2. Создание списка покупок: Пользователь формирует новый список.
3. Добавление товаров в список: Пользователь выбирает товары, добавляет их в список, добавляет комментарии.
4. Редактирование списка: Пользователь изменяет количество, удаляет товары.
5. Отмечание покупок: При покупке товар помечается как купленный.
6. Удаление списка: Есть возможность очистить или удалить весь список**.**
7. Синхронизация с сервером: Выполняется автоматическая или ручная отправка изменений.



[**BPMN диаграмма**](https://github.com/drgchft/practikaAnalitika/blob/main/BPMN(%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA).drawio%20(1).png)

**2. Основные функции приложения (в порядке важности).**

1. Управление списком покупок:

* Создание нового списка покупок.
* Добавление, редактирование и удаление товаров.
* Группировка товаров по категориям.
* Возможность отмечать купленные товары.
* Поддержка нескольких списков покупок для разных случаев.

1. Синхронизация и совместный доступ:

* Хранение данных в облаке и синхронизация между устройствами.
* Совместный доступ к списку покупок для нескольких пользователей.
* История изменений списка.
* Уведомления об изменениях в списке в реальном времени.

1. Работа в офлайн-режиме:

* Возможность создания и редактирования списка без подключения к интернету.
* Автоматическая синхронизация данных при подключении.

1. Удобный поиск и сортировка товаров:

* Быстрый поиск по названию товара в списке.
* Сортировка товаров.
* Предложения популярных товаров на основе истории покупок.

1. Умные функции и автоматизация:

* Автоматическое добавление товаров на основе истории покупок.
* Напоминания о необходимости купить определенные товары.
* Рекомендации по списку покупок.
* Автозаполнение при вводе товаров.

1. Интерактивный интерфейс и пользовательский опыт:

* Быстрое добавление товаров с помощью голосового ввода.
* Цветовые метки и приоритеты для важных товаров.

1. Уведомления и напоминания:

* Push-уведомления о важных изменениях в списке.
* Напоминания о покупках.

1. Интеграция с внешними сервисами (опционально):

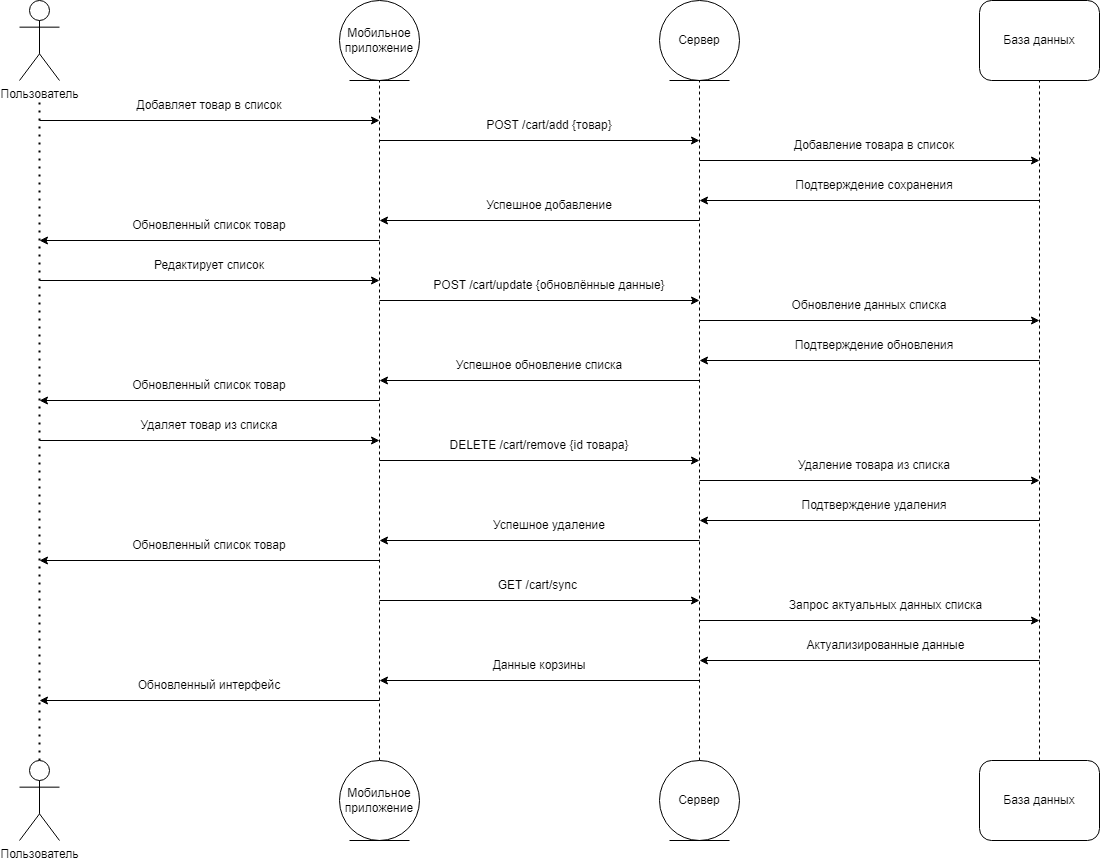
* Экспорт списка покупок.
* Интеграция с голосовыми ассистентами (Google Assistant, Siri).
* Возможность делиться списком покупок через мессенджеры.

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

**Процесс:**

1. Добавление товара в список: Пользователь добавляет товар в список через интерфейс приложения. Мобильное приложение отправляет запрос на сервер с данными товара, который необходимо добавить в список. Сервер обрабатывает запрос, добавляет список в корзину пользователя в базе данных и возвращает обновленную информацию о списке. Сервер возвращает клиенту статус добавления товара в список и актуализированные данные. Мобильное приложение отображает пользователю обновленный список.
2. Редактирование списка: Пользователь редактирует список. Мобильное приложение отправляет запрос на сервер с обновленной информацией. Сервер проверяет запрос, обновляет данные списка в базе данных и возвращает актуализированную информацию о списке. Сервер возвращает успешный ответ с актуализированными данными списка. Мобильное приложение обновляет интерфейс с новым списком товаров.
3. Удаление товара: Пользователь решает удалить товар из списка. Мобильное приложение отправляет запрос на сервер с данными для удаления товара. Сервер проверяет запрос, удаляет товар из корзины, обновляя соответствующие данные в базе данных. Сервер подтверждает успешное удаление товара. Мобильное приложение обновляет интерфейс.
4. Обновление данных (синхронизация при перезапуске приложения или при изменении данных на сервере): При перезапуске приложения или периодической синхронизации, мобильное приложение запрашивает актуальные данные о пользователе, корзине с сервера. Сервер возвращает актуализированные данные о корзине. Мобильное приложение обновляет интерфейс в зависимости от полученной информации.

**UML диаграмма:**

****

[**UML диаграмма**](https://github.com/drgchft/practikaAnalitika/blob/main/UML%D0%B4%D0%B8%D0%B0%D0%B3%D1%80(%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA).drawio.png)

**API методы:**

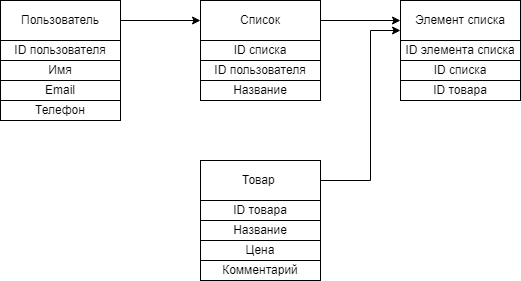
1. POST /cart/add — добавляет товар в список пользователя.

2. PUT /cart/update — редактирует список.

3. DELETE /cart/remove — удаляет товар из списка пользователя.

4. GET /cart/sync — синхронизация списка.

**ER-диаграмма сущностей:**

****

[**ER диаграмма**](https://github.com/drgchft/practikaAnalitika/blob/main/ER%D0%B4%D0%B8%D0%B0%D0%B3%D1%80(%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA).drawio.png)

**4. Прототип экрана (Список товаров).**

**Описание пользовательского интерфейса:**

Страница «Список товаров» приложения «Список покупок в магазинах» представлена структурой, которая легко ориентирует пользователя и позволяет быстро добавить нужные товары, отредактировать список и пометить товары как купленные .

1. Верхняя панель управления:

* Кнопка «Назад» (стрелка влево) – расположена в верхнем левом углу. При нажатии возвращает пользователя на предыдущую страницу.
* Название списка – отображается по центру верхней панели. При двойном нажатии становится редактируемым.
* Кнопка «Три точки» (меню настроек) – находится в верхнем правом углу. При нажатии открывает настройки списка, где можно изменить название, удалить или очистить список и выполнить другие действия.

1. Панель управления списком:

* Кнопка «Добавить товар» – расположена внизу, слева. Позволяет пользователю добавить новый товар в список.
* Кнопка «Поделиться списком» – расположена справа от кнопки добавления. Позволяет отправить ссылку на список другому пользователю.

1. Дополнительные настройки отображения товаров:

* Кнопка «Настроить порядок» – расположена слева. Позволяет выбрать порядок отображения товаров в списке.
* Кнопка «Все товары» – расположена справа. Открывает фильтр отображаемых товаров.

1. Карточки товаров:

* Фотография товара – находится слева.
* Название товара – расположено справа от фотографии, в верхней части карточки.
* Цена товара – указана под названием.
* Кнопка «Развернуть комментарий» – расположена под ценой, позволяет пользователю просмотреть комментарий, добавленный к товару.
* Кнопка «Три точки» (меню товара) – находится в правом верхнем углу карточки. Открывает меню с возможностью редактирования, удаления, отметки о покупке и т. д.
* Кнопка «В магазин» – расположена в правом нижнем углу карточки. Перенаправляет пользователя в магазин, где можно приобрести товар.



[**Прототип экрана (Список товаров)**](https://github.com/drgchft/practikaAnalitika/blob/main/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA%20%D1%82%D0%BE%D0%B2%D0%B0%D1%80%D0%BE%D0%B2.png)

**5. Описание функции покупки товара:**

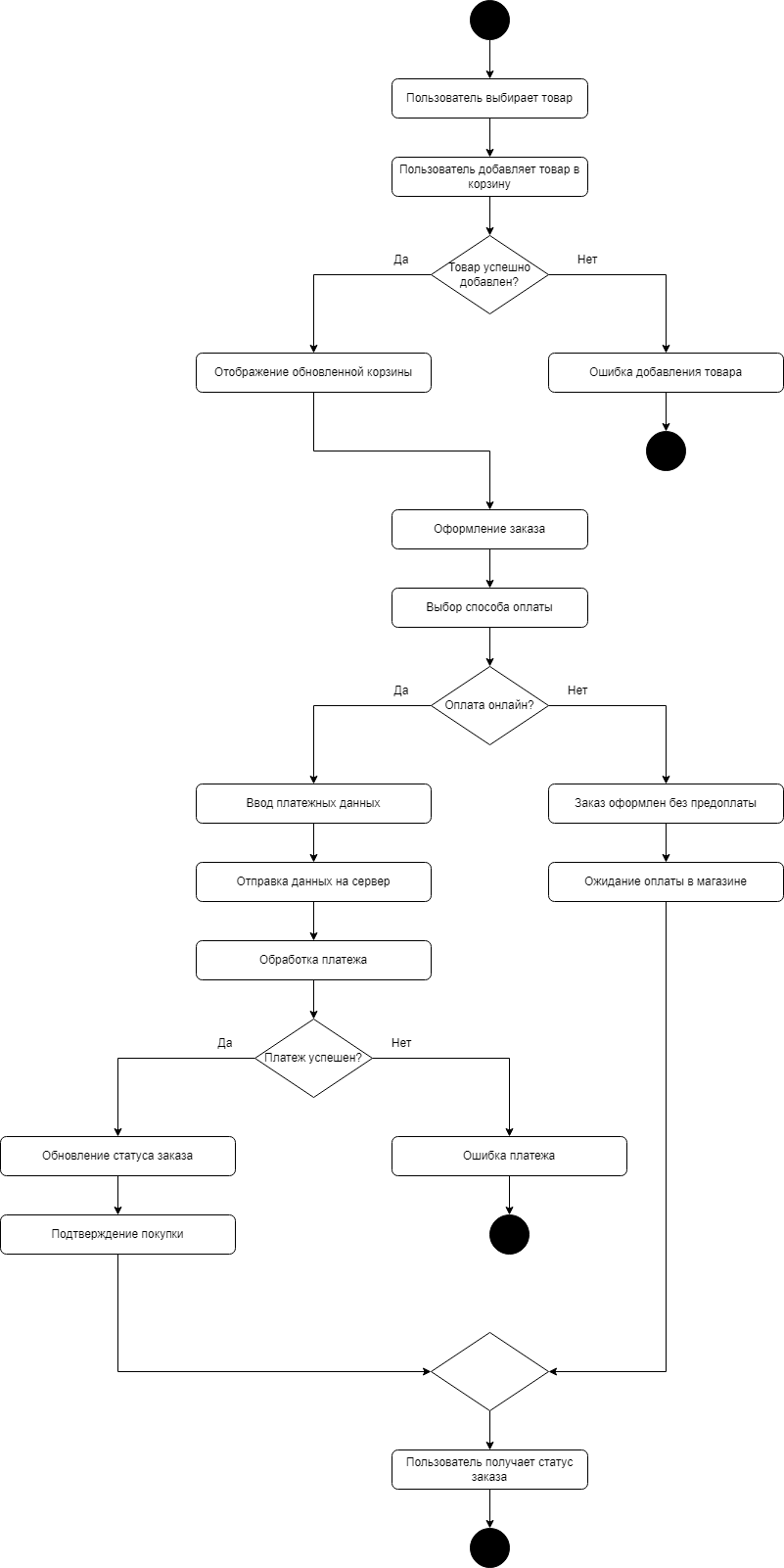
**Описание:**

Функция покупки товара возможна тогда, когда данное мобильное приложение будет интегрировать с магазинами. Функция покупки товара в мобильном приложении включает несколько ключевых этапов: добавление товара в корзину, оформление заказа, выбор способа оплаты и обработка платежной информации. Пользователь выбирает товары, проверяет их в корзине и инициирует оформление заказа. После выбора метода оплаты, приложение обрабатывает платежные данные, подтверждает успешную транзакцию и завершает процесс покупки. После оплаты пользователю отображается подтверждение успешной покупки с информацией о заказе.

**Процесс:**

1. Добавление товара в корзину: Пользователь выбирает товар. Приложение отправляет запрос на сервер для добавления товара. Если операция успешна, отображается обновленная корзина.
2. Оформление заказа: Пользователь выбирает способ оплаты. Если оплата онлайн, вводятся платежные данные. Запрос на сервер, обработка платежа.
3. Оплата заказа: Сервер передает данные в платежную систему. При успешной оплате статус заказа обновляется. Если оплата не прошла, пользователь получает сообщение об ошибке.
4. Получение статуса заказа: Пользователь проверяет статус через API. Сервер возвращает текущий статус .

**UML диаграмма:**

****

[**UML диаграмма**](https://github.com/drgchft/practikaAnalitika/blob/main/UML(%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA).drawio.png)

**API методы:**

POST /api/cart/add — добавляет товар в корзину.

GET /api/cart/{user\_id} — получает содержимое корзины.

POST /api/order/create — создает заказ на основе корзины.

POST /api/payment/confirm — подтверждает оплату заказа.

GET /api/order/{order\_id}/status — получает текущий статус заказа.

**Хранение информации о покупках:**

Информация хранится в базе данных:

1. Таблица пользователей (users): хранит информацию о пользователях, их данных для авторизации и профиле.
2. Таблица товаров (products): хранит информацию о товарах, которые могут быть добавлены в корзину или куплены.
3. Таблица корзин (carts): хранит информацию о корзине пользователя, которая состоит из нескольких товаров.
4. Таблица заказов (orders): хранит информацию о заказах, созданных пользователями, включая состояние оплаты и дату оформления.
5. Таблица покупок (order\_items): хранит информацию о каждом товаре, который был куплен в рамках заказа.
6. Таблица платежей (payments): хранит информацию о платежах по заказам.

**6. Основные сложности разработки.**

**Сложности:**

1. Синхронизация данных между клиентом и сервером: Обеспечение надежной двусторонней синхронизации, чтобы изменения, внесенные на одном устройстве, моментально отражались на других. Решение проблем конфликта версий списка покупок при одновременном редактировании несколькими пользователями.
2. Оптимизация работы с базой данных: Быстрое выполнение запросов к БД для работы с большими списками товаров. Кэширование данных для минимизации нагрузки на сервер.
3. Реализация офлайн-режима: Хранение локальной копии списка покупок и её последующая синхронизация с сервером. Разработка механизма обработки конфликтов при изменениях в офлайн-режиме.
4. Обеспечение безопасности данных: Шифрование передаваемых данных. Защита API от несанкционированного доступа, включая аутентификацию и авторизацию пользователей.
5. Удобство пользовательского интерфейса: Проектирование интуитивно понятного UX/UI. Возможность быстрого добавления, редактирования и отметки купленных товаров.
6. Интеграция с внешними сервисами (опционально): Возможность распознавания текста (сканирование чеков). Подключение уведомлений (например, напоминания о покупках).

**Вопросы заказчику:**

1. Нужно ли реализовать авторизацию пользователей или список должен быть доступен без регистрации?
2. Требуется ли многопользовательский доступ к одному списку?
3. Нужно ли учитывать категории товаров или список будет простым?
4. Должен ли приложение работать офлайн с последующей синхронизацией?
5. Какие способы уведомлений требуются (push-уведомления, email и т.д.)?
6. Планируется ли будущая интеграция с магазинами или API для сравнения цен?
7. Есть ли необходимость в статистике покупок (анализ трат, популярные товары и т. д.)?

**7. SQL-запросы.**

1. Общая стоимость книг для каждого автора и отсортировать результат в порядке убывания:

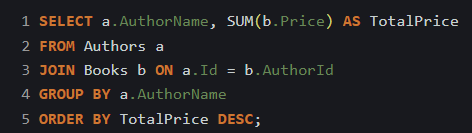
SELECT a.AuthorName, SUM(b.Price) AS TotalPrice

FROM Authors a

JOIN Books b ON a.Id = b.AuthorId

GROUP BY a.AuthorName

ORDER BY TotalPrice DESC;



1. Стоимость книг автора превышает 1500:

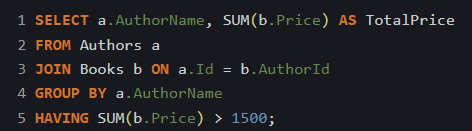
SELECT a.AuthorName, SUM(b.Price) AS TotalPrice

FROM Authors a

JOIN Books b ON a.Id = b.AuthorId

GROUP BY a.AuthorName

HAVING SUM(b.Price) > 1500;



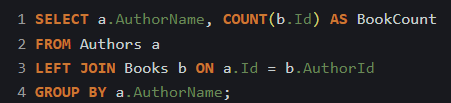
1. Вывести авторов с количеством книг:

SELECT a.AuthorName, COUNT(b.Id) AS BookCount

FROM Authors a

LEFT JOIN Books b ON a.Id = b.AuthorId

GROUP BY a.AuthorName;



1. Получить авторов без книг:

SELECT a.AuthorName

FROM Authors a

LEFT JOIN Books b ON a.Id = b.AuthorId

WHERE b.Id IS NULL;

