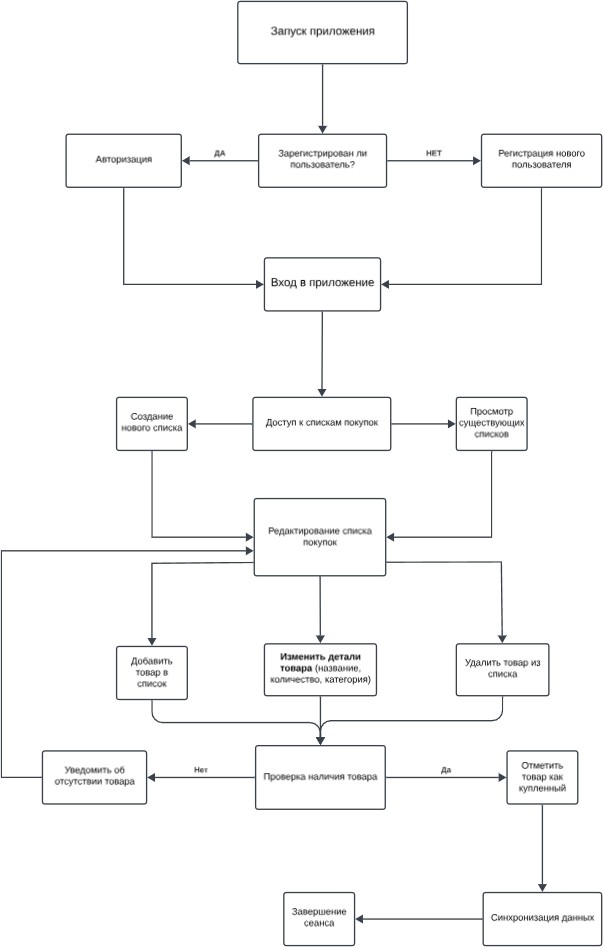
Задание 1.

Бизнес-логику мобильного приложения (использовать любую удобную нотацию моделирования бизнес-процессов).



Задание 2.

Основные функции данного приложения (список должен быть составлен в порядке убывания важности функций для пользователя).

## 1.Создание и управление списками покупок

* 1. Создание нового списка покупок.
  2. Редактирование существующего списка (добавление, удаление, изменение товаров).
  3. Просмотр списка покупок.

## 2.Отметка купленных товаров

* 1. Возможность помечать товары как купленные.
  2. Автоматическое обновление статуса списка.

## 3. Синхронизация данных с сервером

* 1. Сохранение данных для доступа с разных устройств.
  2. Автосинхронизация в режиме онлайн.

## 4. Работа в оффлайн-режиме

4.1 Возможность создавать, редактировать и просматривать списки без доступа к интернету.

## 5. Уведомления о состоянии списка

* 1. Напоминания о необходимости завершить покупки.
  2. Уведомления об изменениях списка, если он синхронизирован с другими пользователями.

## 6. Шаблоны списков

* 1. Создание шаблонов для часто используемых покупок.
  2. Быстрое добавление товаров из шаблонов в список.

## 7. Поделиться списком

* 1. Возможность отправить список другим пользователям или через мессенджеры.
  2. Совместное редактирование списка в режиме реального времени.

## 8. Категоризация товаров

* 1. Упорядочивание товаров по категориям (продукты, бытовая химия и т.д.).
  2. Фильтрация списка по категориям.

## 9. История покупок

* 1. Просмотр ранее купленных товаров.
  2. Возможность восстанавливать списки из истории.

## 10 Поиск товаров в списк

10.1 Быстрый поиск по названию или категории.

## 11. Визуализация данных

* 1. Статистика по покупкам (например, на что тратится больше всего).
  2. Графики и диаграммы для анализа расходов.

Задание 3.

Процесс синхронизации данных между клиентом и сервером (добавление и удаление списка, наполнение и редактирование списка, покупка/«откупка продукта» и т.д.). Представить все в диаграммах UML, API методах и других представлениях, также составить ER-диаграмму сущностей.

Диаграмма последовательности

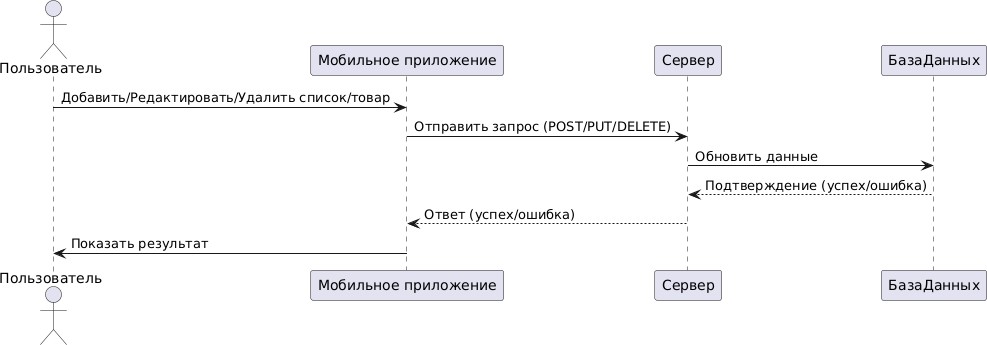
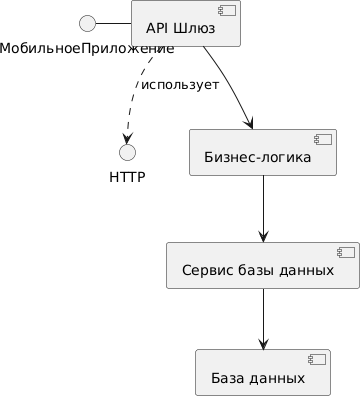
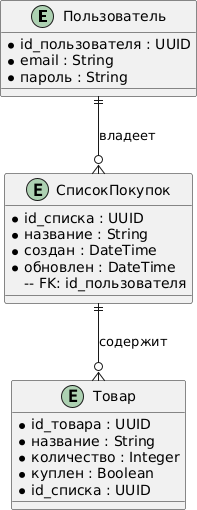


Диаграмма компонентов



ER-диаграмма сущностей



API методы

**Метод:** POST

1. Добавление:
   1. Запрос { "name": "Grocery Shopping" }
   2. Ответ { "list\_id": "12345", "name": "Grocery Shopping", "created\_at": "2025-02- 13T10:00:00Z" }

**Метод:** DELETE

1. Удаление:
   1. Удаление пример: { "message": "List deleted successfully" }

**Метод:** POST

1. Добавление товара в список:
   1. Запрос: { "name": "Milk", "quantity": 2 }
   2. Ответ { "item\_id": "67890", "name": "Milk", "quantity": 2, "purchased": false }

**Метод:** PUT

1. Редактирование товара
   1. Запрос : { "name": "Almond Milk", "quantity": 1, "purchased": true }

Задание 4.

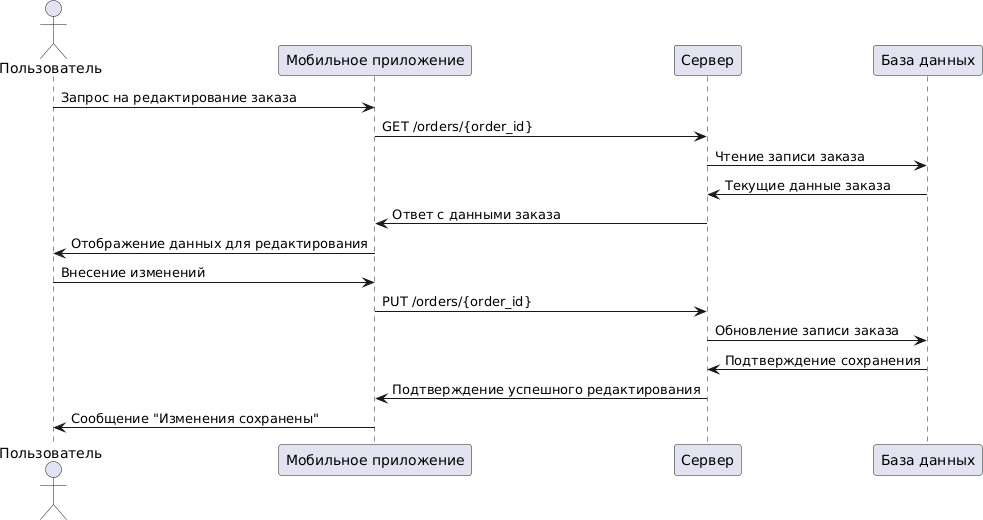
Подготовить прототип одного из экранов мобильного приложения и описать пользовательский интерфейс для данного экрана (например, покупка товара).

[https://www.figma.com/design/zvDbNXyl26qpUqiPD5kJka/Untitled?node-id=0-](https://www.figma.com/design/zvDbNXyl26qpUqiPD5kJka/Untitled?node-id=0-1&p=f&t=UgzmMqra90qIciaX-0) [1&p=f&t=UgzmMqra90qIciaX-0](https://www.figma.com/design/zvDbNXyl26qpUqiPD5kJka/Untitled?node-id=0-1&p=f&t=UgzmMqra90qIciaX-0)

Задание 5.

Подготовить подробное описание функции покупки товара, 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).

UML Диаграмма



Api методы

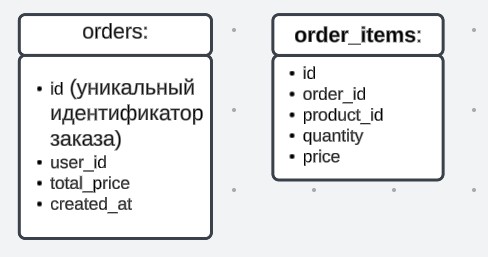
1. Получение информации о заказе.

## Метод GET:

* 1. GET /orders/{order\_id}
  2. Пример: { "order\_id": "12345", "items": [ { "product\_id": "67890",

"name": "Продукт 1", "quantity": 2, "price": 500 } ], "total\_price": 1000 }

1. Обновление информации о заказе.
   1. **Метод**: PUT /orders/{order\_id}:
   2. Пример: { "items": [ { "product\_id": "67890", "quantity": 3 } ] }
   3. ответ: { "message": "Order updated successfully" }



Задание 6:

Основные, на ваш взгляд, сложности разработки такого приложения. Вопросы, возникшие при выполнении тестового задания, которые вы бы задали заказчику

# Основные сложности разработки:

1. **Синхронизация данных** — нужно синхронизировать изменения на разных устройствах.
2. **Управление состоянием** — обработка актуальности списка покупок и ошибок при сети.
3. **Безопасность** — защита данных пользователей и использование безопасных протоколов.
4. **Уведомления** — реализация системы уведомлений для напоминаний о покупках.
5. **Масштабируемость** — сервер должен справляться с большим количеством пользователей.
6. **Платформенная совместимость** — поддержка разных ОС и экранов.
7. **Хранение данных** — кэширование и локальное хранение данных на устройствах.

# Вопросы заказчику:

1. Как обрабатывать дублирующиеся товары в списках?
2. Какие категории товаров должны быть?
3. Должны ли пользователи редактировать товары?
4. Нужно ли синхронизировать данные между устройствами?
5. Требуется ли аутентификация пользователей?
6. Какие требования к интерфейсу?
7. Есть ли потребность в интеграции с другими сервисами?

Задание 7:

Есть таблицы Books и Authors, где AuthorId табл. Books равно Id табл. Authors. Необходимо написать SQL-запрос, чтобы найти: 1. Общую стоимость книг для каждого автора и отсортировать результат в порядке убывания; 2. Стоимость книг автора превышает 1500; 3. Вывести авторов с количеством книг; 4. Получить автора без книг.

1. Общая стоимость книг для каждого автора, отсортированная по убыванию:

SELECT a.Name AS AuthorName, SUM(b.Price) AS TotalCost FROM Authors a JOIN Books b ON a.Id = b.AuthorId GROUP BY a.Name ORDER BY TotalCost DESC;

1. Стоимость книг автора превышает 1500:

SELECT a.Name AS AuthorName, SUM(b.Price) AS TotalCost FROM Authors a JOIN Books b ON a.Id = b.AuthorId GROUP BY a.Name HAVING SUM(b.Price) > 1500;

1. Вывести авторов с количеством книг:

SELECT a.Name AS AuthorName, COUNT(b.Id) AS BookCount FROM Authors a LEFT JOIN Books b ON a.Id = b.AuthorId GROUP BY a.Name;

1. Получить автора без книг:

SELECT a.Name AS AuthorName FROM Authors a LEFT JOIN Books b ON a.Id = b.AuthorId WHERE b.Id IS NULL;