**Задание 1. Дано: Необходимо описать: Мобильное клиент-серверное приложение «Список покупок в магазинах» (Примечание: приложение без интеграции с магазинами). 1. Бизнес-логику мобильного приложения (использовать любую удобную нотацию моделирования бизнес-процессов).**  
  
Основные бизнес-процессы:

1. Регистрация и авторизация пользователя  
 Ввод email и пароля, получение access token  
 После авторизации доступ ко всем функциям  
2. Создание и управление списками покупок  
 Создание, редактирование, удаление списков  
 Добавление товаров с названием, количеством, категорией  
3. Отметка товаров как купленных  
 Изменение флага `purchased`, визуальное отображение  
4. Совместное использование списков  
 Приглашение по email, доступ к совместному редактированию, Push-уведомления при изменении списков другими пользователями, ведения истории редактирования каждого пользователя.  
5. Синхронизация с сервером  
 Все данные хранятся на сервере, доступны с разных устройств  
6. Выход из системы  
 Токен удаляется, данные сохраняются на сервере

**Задание 2 Основные функции данного приложения (список должен быть составлен в порядке убывания важности функций для пользователя)**

|  |  |
| --- | --- |
| Функция | Описание и назначение |
| Создание и редактирование списков | Создание нового списка покупок, его переименование или удаление. |
| Добавление, редактирование товаров | Наполнение списка позициями, изменение количества, названия и категории. |
| Отметка товара как купленного | Фиксация завершённой покупки — ключевое действие для работы со списком. |
| Синхронизация с сервером | Обеспечивает сохранность и доступность данных на разных устройствах. |
| Регистрация и авторизация | Идентификация пользователя и обеспечение доступа к персонализированным данным. |
| Массовая отметка товаров | Позволяет одним действием отметить все товары как купленные («Купить всё»). |
| История покупок | Хранение информации о купленных ранее товарах с указанием времени покупки. |
| Push-уведомления | Информируют пользователя об изменениях в совместных списках или о напоминаниях. |
| Совместное использование списка | Возможность коллективного редактирования и синхронизации между пользователями. |
| Просмотр и управление несколькими списками | Управление несколькими активными списками по сценариям или магазинам. |
| Категоризация товаров | Группировка товаров по типам (например, овощи, напитки), фильтрация и сортировка. |
| Оффлайн-режим | Работа со списками без подключения к интернету с последующей синхронизацией. |
| Аналитика покупок | Сбор и отображение статистики: часто покупаемые товары, категории, суммы. |
| Интеллектуальные автоподсказки | Автоматическое предложение товаров на основе предыдущих покупок и истории. |
| Импорт/экспорт списков | Выгрузка списка в файл или текст, отправка через мессенджеры и почту. |
| Темная/светлая тема интерфейса | Настройки внешнего вида приложения под предпочтения пользователя. |
| Голосовой ввод | Добавление товаров голосом — удобно в дороге или при занятых руках. |
| Интеграция с голосовыми помощниками | Управление списками через Siri, Google Assistant и другие платформы. |
| Избранные товары и шаблоны | Создание шаблонов и сохранение часто используемых товаров для быстрого доступа. |
| Уведомления о сроках | Оповещения о регулярных или предстоящих покупках на заданные даты или периоды. |

# Задание 3. Синхронизация данных между клиентом и сервером

## Основные сценарии синхронизации

|  |  |  |
| --- | --- | --- |
| Сценарий | Действие клиента | Ответ сервера |
| Создание списка | POST /lists | Новый list\_id и created\_at |
| Удаление списка | DELETE /lists/{list\_id} | Статус выполнения |
| Получение всех списков | GET /lists | Списки + товары |
| Добавление товара | POST /lists/{id}/items | item\_id, created\_at |
| Редактирование товара | PUT /lists/{id}/items/{item\_id} | Обновлённые поля |
| Отметка товара как купленного | PATCH /lists/{id}/items/{item\_id}/toggle | purchased: true, purchased\_at |
| Массовая отметка товаров | PATCH /lists/{id}/items/bulk-toggle | Обновлённый список |
| Удаление товара | DELETE /lists/{id}/items/{item\_id} | Статус выполнения |
| Push-уведомления | WebSocket / Firebase | Оповещение участников |
| Оффлайн-синхронизация | POST /sync | Слияние / разрешение конфликтов |

## Примеры REST API

### Создание нового списка — POST /api/v1/lists

Request:  
{  
 "name": "Закупка на выходные"  
}  
  
Response:  
{  
 "list\_id": "abc123",  
 "name": "Закупка на выходные",  
 "created\_at": "2025-06-25T10:15:00Z"  
}

### Добавление товара — POST /api/v1/lists/abc123/items

Request:  
{  
 "name": "Яйца",  
 "quantity": 10,  
 "category": "Продукты"  
}  
  
Response:  
{  
 "item\_id": "itm001",  
 "name": "Яйца",  
 "quantity": 10,  
 "category": "Продукты",  
 "purchased": false,  
 "created\_at": "2025-06-25T10:16:00Z"  
}

### Обновление товара — PUT /api/v1/lists/abc123/items/itm001

Request:  
{  
 "name": "Яйца деревенские",  
 "quantity": 12,  
 "category": "Продукты"  
}  
  
Response:  
{  
 "item\_id": "itm001",  
 "name": "Яйца деревенские",  
 "quantity": 12,  
 "category": "Продукты",  
 "updated\_at": "2025-06-25T10:45:00Z"  
}

### Отметка как купленного — PATCH /api/v1/lists/abc123/items/itm001/toggle

Response:  
{  
 "item\_id": "itm001",  
 "purchased": true,  
 "purchased\_at": "2025-06-25T11:05:00Z"  
}

### Массовая покупка — PATCH /api/v1/lists/abc123/items/bulk-toggle

Request:  
{  
 "item\_ids": ["itm002", "itm003", "itm004"]  
}  
  
Response:  
{  
 "updated": [  
 {"item\_id": "itm002", "purchased": true, "purchased\_at": "2025-06-25T11:07:00Z"},  
 {"item\_id": "itm003", "purchased": true, "purchased\_at": "2025-06-25T11:07:01Z"},  
 {"item\_id": "itm004", "purchased": true, "purchased\_at": "2025-06-25T11:07:02Z"}  
 ]  
}

### Оффлайн-синхронизация — POST /api/v1/sync

Request:  
{  
 "pending\_updates": [  
 {  
 "action": "update\_item",  
 "item\_id": "itm005",  
 "fields": {  
 "name": "Кефир",  
 "quantity": 2,  
 "purchased": true,  
 "purchased\_at": "2025-06-25T09:50:00Z"  
 }  
 }  
 ]  
}  
  
Response:  
{  
 "synced": true,  
 "conflicts": []  
}

### Получение уведомлений — GET /api/v1/notifications

Response:  
[  
 {  
 "notification\_id": "n001",  
 "type": "list\_shared",  
 "message": "Ирина поделилась с вами списком «Семейный»",  
 "created\_at": "2025-06-25T08:00:00Z",  
 "is\_read": false  
 },  
 {  
 "notification\_id": "n002",  
 "type": "item\_purchased",  
 "message": "Андрей отметил «Молоко» как купленное",  
 "created\_at": "2025-06-25T08:30:00Z",  
 "is\_read": false  
 }  
]

**Задание 4 Подготовить прототип одного из экранов мобильного приложения и описать пользовательский интерфейс для данного экрана (например, покупка товара)**

# Описание интерфейса экрана «Список покупок»

Экран представляет собой список товаров, сгруппированных по статусу: «Нужно купить» и «Куплено». Каждый товар отображается в виде строки с названием, количеством и чекбоксом для отметки как купленного. Верхняя панель содержит название списка и иконку меню. Внизу — кнопка добавления товара.  
Основные элементы экрана:

Заголовок: название списка (например, «На неделю»)  
 Кнопка меню (иконка «⋮») — доступ к действиям со списком: переименовать, удалить, поделиться  
 Блок товаров:  
 Название товара  
 Количество (если задано)  
 Чекбокс — отметить как купленный  
 Отображение purchased\_at под названием, если товар куплен  
 Кнопка «+» — добавление нового товара  
 Кнопка «Купить всё» (массовая отметка)  
 Статус оффлайн/онлайн — отображается при потере соединения

https://www.figma.com/design/31vBiHuiIYoIWVGHfl1Khj/Untitled?node-id=0-1&t=6MK39P87205nDu6r-1

**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).**

# Описание поведения

1. Пользователь отмечает товар как купленный через UI (нажатие чекбокса, свайп или голос).  
2. UI немедленно обновляет визуальный статус товара (✓, серый цвет, зачёркнутый текст).  
3. Клиент отправляет PATCH-запрос на сервер.  
4. Сервер устанавливает `purchased = true` и сохраняет `purchased\_at` (время покупки).  
5. В ответ клиент получает обновлённый товар, обновляет UI и, если список совместный, показывает уведомление.  
6. При отсутствии подключения изменения сохраняются локально и позже отправляются батчем через `/sync`.

# REST API методы

## PATCH /api/v1/lists/{list\_id}/items/{item\_id}/toggle

Response:  
{  
 "item\_id": "itm123",  
 "purchased": true,  
 "purchased\_at": "2025-06-25T13:55:00Z"  
}

## PATCH /api/v1/lists/{list\_id}/items/bulk-toggle

Request:  
{  
 "item\_ids": ["itm123", "itm124", "itm125"]  
}  
  
Response:  
{  
 "updated": [  
 {"item\_id": "itm123", "purchased": true, "purchased\_at": "2025-06-25T13:56:00Z"},  
 {"item\_id": "itm124", "purchased": true, "purchased\_at": "2025-06-25T13:56:01Z"},  
 {"item\_id": "itm125", "purchased": true, "purchased\_at": "2025-06-25T13:56:02Z"}  
 ]  
}

## POST /api/v1/sync

Request:  
{  
 "pending\_updates": [  
 {  
 "action": "update\_item",  
 "item\_id": "itm001",  
 "fields": {  
 "purchased": true,  
 "purchased\_at": "2025-06-25T14:10:00Z"  
 }  
 }  
 ]  
}  
  
Response:  
{  
 "synced": true,  
 "conflicts": []  
}

# Структура хранения в БД

id: UUID - уникальный идентификатор  
 list\_id: UUID - привязка к списку  
 name: TEXT - название товара  
 quantity: INTEGER - количество  
 purchased: BOOLEAN - статус покупки  
 purchased\_at: TIMESTAMP - время покупки  
 is\_favorite: BOOLEAN - избранное  
 updated\_at: TIMESTAMP - дата последнего изменения

# Поддерживаемые сценарии

|  |  |
| --- | --- |
| Сценарий | Поведение |
| Индивидуальная покупка | PATCH /toggle — изменяет purchased и устанавливает purchased\_at. |
| Массовая покупка | PATCH /bulk-toggle — все выбранные товары получают статус purchased. |
| Оффлайн-режим | Изменения сохраняются в локальной очереди и отправляются позже через /sync. |
| История покупок | Поле purchased\_at используется для отображения и статистики. |
| Push-уведомления | Уведомления участникам совместного списка через Firebase/WebSocket. |
| UI | После покупки товар помечается визуально и отображается дата/время покупки. |
| Аналитика | Система учитывает товары с purchased=true для построения рекомендаций. |
| Голосовое управление | Пользователь может голосом активировать покупку товара. |

**Задание 6 Основные, на ваш взгляд, сложности разработки такого приложения. Вопросы, возникшие при выполнении тестового задания, которые вы бы задали заказчику.**

# Основные сложности разработки приложения «Список покупок»

1. Синхронизация данных между клиентом и сервером

Обновление статусов товаров и списков в реальном времени.

Разрешение конфликтов при одновременном редактировании одним или несколькими пользователями.

Обработка push-уведомлений при совместной работе.

1. Работа с оффлайн-режимом

Корректное кэширование локальных изменений.

Синхронизация данных через батч-запросы после восстановления соединения.

Разрешение конфликтов между оффлайн- и онлайн-версиями данных.

1. Аутентификация и безопасность

Поддержка входа по email/паролю и OAuth.

Безопасное хранение токенов и защита API.

Контроль доступа к общим спискам и их разграничение (editor/viewer).

1. Масштабируемость базы данных

Эффективная структура хранения товаров и истории покупок.

Индексация по `user\_id`, `purchased\_at`, `updated\_at`.

Поддержка избранных товаров и шаблонов.

1. UI/UX и инклюзивность

Простота интерфейса при богатом функционале.

Интуитивность при использовании push-уведомлений, истории, голосового ввода.

Быстрая доступность функций: чекбокс, свайп, фильтры, избранное.

1. Поддержка расширенного функционала

Массовые действия над товарами.

Автоматическая аналитика (популярные товары, частота, сезонность).

Голосовое управление через Google Assistant / Siri.

# Вопросы к заказчику

- Будет ли реализована совместная работа над списками?

- Какие роли пользователей предусмотрены (администратор, редактор, наблюдатель)?

- Нужно ли реализовать избранные товары и шаблоны списков?

- Нужна ли история покупок? На какой срок хранить и можно ли очищать?

- Должны ли push-уведомления быть обязательными или пользователь может отключать их?

- Требуется ли веб-версия или только мобильная платформа?

- Нужна ли поддержка тёмной темы и доступности (voice over)?

- Какие требования по скорости синхронизации и оффлайн-работе?

- Требуется ли интеграция с внешними API (например, API цен магазинов)?

- Какие бизнес-метрики важны: частота использования, количество покупок, завершённость?

**Задание 7!**

### Общая стоимость книг для каждого автора (отсортировано по убыванию)

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;

### Авторы, у которых стоимость книг превышает 150

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;

### Авторы 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;

### Авторы без книг

SELECT a.AuthorName

FROM Authors a

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

WHERE b.Id IS NULL;