**Дано:**

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

**Необходимо описать:**

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

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

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.

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

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).

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

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

**BPMN Нотация**

* **Процесс 1: Управление Списками Покупок**
  + **Старт:** Пользователь открывает приложение.
  + **Действие 1.1: Создать Новый Список / Выбрать Существующий Список.**
  + **Действие 1.2 (Если создание):** Присвоить списку название (например, "Продукты на неделю", "Строймаркет", "День Рождения").
  + **Действие 1.3 (Если выбор):** Открыть выбранный список.
  + **Конец:** Пользователь видит содержимое списка или пустой шаблон нового списка.
* **Процесс 2: Управление Элементами Списка (Товарами)**
  + **Старт:** Пользователь находится внутри конкретного списка покупок.
  + **Действие 2.1: Добавить Товар.**
    - Пользователь вводит название товара вручную или выбирает из истории/избранного.
    - (Опционально) Пользователь указывает количество (1, 2 кг, 500г, пачка).
    - (Опционально) Пользователь указывает категорию (Овощи, Молочка, Бакалея и т.д.) для группировки.
    - (Опционально) Пользователь добавляет комментарий ("Только красные", "Без глютена").
  + **Действие 2.2: Отметить Товар как Купленный/Не Купленный.**
    - Пользователь тапает на чекбокс или название товара.
    - Система визуально меняет статус товара (зачеркивание, изменение цвета, перемещение в секцию "куплено").
  + **Действие 2.3: Редактировать Товар.**
    - Пользователь выбирает опцию "редактировать" для товара.
    - Пользователь изменяет название, количество, категорию, комментарий.
  + **Действие 2.4: Удалить Товар.**
    - Пользователь выбирает опцию "удалить" для товара или делает свайп.
    - Система запрашивает подтверждение.
    - Товар удаляется из списка.
  + **Конец:** Изменения в списке сохранены (автоматически или по команде).
* **Процесс 3: Синхронизация и Общий Доступ (Если Есть)**
  + **Старт:** Пользователь хочет синхронизировать списки между устройствами или поделиться списком.
  + **Действие 3.1 (Синхронизация):**
    - Пользователь входит в аккаунт (email, соцсеть) на всех устройствах.
    - Система автоматически отправляет изменения на сервер при наличии интернета.
    - Система автоматически получает изменения с сервера на других устройствах при запуске/обновлении.
  + **Действие 3.2 (Общий Доступ):**
    - Пользователь выбирает опцию "Поделиться списком".
    - Пользователь выбирает способ приглашения (ссылка, email, телефон).
    - Пользователь выбирает уровень доступа приглашаемого (только просмотр, редактирование).
    - Система генерирует ссылку/приглашение.
    - Принявший приглашение пользователь получает доступ к списку в своем приложении.
    - Изменения, вносимые любым участником с правами на редактирование, синхронизируются со всеми.
  + **Конец:** Списки актуальны на всех устройствах пользователя и/или доступны выбранным участникам.

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

1. **Создание и Удаление Списков Покупок:** Самая базовая функция. Без возможности создать новый список (например, "Еженедельные продукты", "Для пикника", "День рождение") приложение бесполезно. Удаление ненужных списков также необходимо.
2. **Добавление Товаров в Список:** Возможность вручную вписать название нужного товара (например, "Молоко", "Хлеб", "Яблоки") - фундаментальное действие. Без этого списки пусты.
3. **Отметка Товаров как Купленных:** Ключевая функция во время использования списка в магазине. Позволяет отслеживать прогресс (чекбокс, зачеркивание). Без нее теряется основная утилитарная ценность приложения в точке продаж.
4. **Просмотр Списка Покупок (Чтение):** Четкое, удобное для чтения отображение списка товаров (название, количество, статус) - необходимо как при планировании, так и в магазине.
5. **Редактирование Товаров (Изменение названия, количества, комментария) и Удаление Товаров:** Возможность исправить опечатку, изменить необходимое количество или удалить ненужный товар до или во время похода в магазин критична для актуальности списка.
6. **Организация Товаров по Категориям (Группировка):** Автоматическая или ручная группировка товаров по отделам магазина (Овощи/Фрукты, Молочные продукты, Бакалея, Заморозка и т.д.) **значительно** ускоряет и упрощает процесс покупок, делая список логичным. Это важное улучшение базового функционала.
7. **Сохранение Списков Локально:** Гарантия того, что созданные списки не исчезнут после закрытия приложения. Базовое требование к надежности.
8. **История Часто Добавляемых Товаров:** Предложение или быстрый доступ к товарам, которые пользователь часто добавлял в прошлые списки, ускоряет создание нового списка.
9. **Возможность Указать Количество для Товара:** "2 пачки", "1 кг", "500г" - делает список более точным и полезным, особенно для продуктов, где количество важно.
10. **Синхронизация Списков между Устройствами Пользователя (Через Аккаунт):** Позволяет создавать/редактировать список на одном устройстве (например, на планшете дома) и иметь актуальную версию на другом (например, на телефоне в магазине). Повышает удобство, но требует интернета и регистрации.
11. **Совместный Доступ к Списку (Шаринг):** Возможность поделиться списком с другим пользователем приложения (членом семьи, другом) для совместного редактирования или просто просмотра (например, "Купи, что есть в списке"). Полезно для семей или совместных мероприятий.
12. **Сохранение/Копирование Списков (Шаблоны):** Возможность сохранить часто используемый список (например, "Стандартная неделя") как шаблон для быстрого создания его копии в будущем, избегая ручного ввода заново.
13. **Добавление Комментариев к Товару:** Дополнительные пометки ("Только марка X", "Без сахара", "Для торта") повышают точность списка.
14. **Поиск по Товарам в Списке:** Полезен в длинных списках для быстрого нахождения конкретного товара.
15. **Сортировка Товаров (по названию, категории, статусу):** Дополнительный способ организации, улучшающий просмотр.

**Обоснование приоритетов:**

* **1-5:** Абсолютно необходимый минимум для выполнения основной задачи приложения. Без этих функций приложение не может выполнять свою главную роль.
* **6 (Категории):** Резко повышает удобство использования в магазине, делая процесс покупок гораздо более эффективным и менее стрессовым.
* **7 (Сохранение):** Базовое требование надежности.
* **8-9 (История, Количество):** Существенно ускоряют и улучшают процесс создания списка, делая его более точным.
* **10-11 (Синхр, Шаринг):** Добавляют удобство и возможности для совместного использования, но не являются строго обязательными для одиночного пользователя.
* **12-15:** Дополнительные функции, повышающие удобство, гибкость и эффективность работы с приложением, но не влияющие на выполнение его основной функции.

**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.**

#### 3.1 Диаграмма последовательности (UML Sequence Diagram) - Основной сценарий синхронизации



<https://www.plantuml.com/plantuml/png/lLLTJzH057tlhxZ15um1DvBmi0-63p4nOa319sDCtHr9mzAkqox641B0Y0yIYGbFQY36Fr2GZo9i-Gjt_f5dxh1fhOiFnfY6cSxCkOTppxtJ7OiJNoUTfPQN14bB4UzmnYb_DsiOp_6an1USctK-uoC-v9mFE3KRU3ypMzor9rXbTU9TOBfoRiTJmi4cO5rEmPN91gQkUOjbZksCJt2qYJdpuiKWRFlQNwBdVcDnGKUTi3aPjI9Dr-xMx5D2nEfbHuKDHIiUqRZMqQjBPAqsCZ8wQZSRIQGd8jrKkdeq7RnGqq6e-f_EmegW4RJzCEa7MFKytrBJ0EznBWQS_6h80FanFOwLzje55U2Vu5vcpJ0R0fbi1IfCAg3FuFh1gJCMdDl47tb7yDDHms_TcQW4x9ftV0GtKHUpHOY-aAKpzLG27tPKHrN2lWhgm8w2cLDwMUchGLKn3WyrdisKXcvVfaDrspqvxoE70sInZU0DWZNEeZMmVh6sIPhvWFKoMFOJrNF3SpC8U_c24K9oQPaKvZd6GMavU6xUcFKwjLNO3CA5wmLVpwqUor1PefJk40GkjsC-b2P5TRocgbAKDOcLVNEiM6y-qEIZgV7vgMVtvgSU5B9SBMJy6qakt1iU7Yw9ShK0n-pCt3pTZ5-73Lg9sahxIH25SPsU0FzqjH3X0ZpFpGM5CtjFhkgvT6g_r3YhCi6ZteiavHP_J8hawf_BqcsTIfuLta9ZsQjRjHhDtASRvBurvZtutB-GpqlsEzCLVRNB-q9HLB7SLZHehLJD_zTLtn2IbZjgtveEaRbL1Qwkv5gjnxzkBnSb17AxhLC8xSavnf_JLzW7Db4ddtYeop3YYeEyUiF7S5NnK_CJ>

#### 3.2 Диаграмма состояний (UML State Diagram) для элемента списка

#### <https://www.plantuml.com/plantuml/png/bLNFJnD16BtFNp69joOGi0a73WQ5a7Y063aQ3xMTrWtBhbaMY1gJ0lum0Qan78o94ELiKajNbbQs_yBCV-HxCx1kbsYK9jlPcU-zxxtlcw_ChSUrADvOyyNwgXSygKMrDVceLbzjHU56q9WF_J2IDnQd-2b401Gh-Ln8UUTQhFtZIHWrL7IvEpLLhSxCOFTs58LRvSt5C8YNQsiAfrxDlznOyPufELqLBuI8lTXNKX_gL7VrgMdZ-HEVhZJREZFRPWVFZhx0yRaqx_HG9pXDz0KERCn8ftWEy89bfWTcrtqBNpLZ6OSoybgFOzdm8bMFlJ0Gua7bePoOk2MNrPQSbVeBeizq7tcP9BcMH32EyVTKqF225Z67m7m7ecT3BwJkISXyenFJndlVx8khO09NdWPrrI3kX88HzHQvkgP3_Icol08ii-dFiFqQZZfZ1FDXqFIzUaoA8oGck0sZFrpo0S7c9RIayByZN3wBN0eRNjDpoOyHDZJxo951U1VU4hvBfdLsiHgA75BoV0J0248Bhg7ZpnOEH2wRJFVLUkXlEYMVnWLBotg69of3Pf47ZvVla4q2SaydhjjzQEgnwYX57qWS5TqlA5_5VpUFAhBrGslTnJiiRmbmtugN8O-s8eMRbDmbHmOefrb5coegzkq3PTfYfInSmi9HoWXyu3G7Q9miuyg5EuPJLhjpTVjftHN0NIY-k3IcyxzqUi0oEASSgNyaN59Hoz6zrpsEAwSNtoddmV9apksG76x8EFX1YEBP8GsRTXP1NwdepwmAEboUzjbA1G4JO_Cg4P1HA5i9vin5VBIO1Fq2K_kA0YHalRjmIKFFCBfhjcVpXfHlSchNFUOIc0H9pKwv-rcXzBmOJlzN04TcxzfKaiTvc9xa5FUfTrmV2rMSniGL2v2RW9paLy_kkkdJVVj3SCggItg18qe-vmh22QvE5czVbbUYhF2tmpciy6_Z5m00>

#### 3.3 RESTful API методы для синхронизации

#### 

#### {

#### "type": "CREATE\_ITEM",

#### "entity": "ShoppingItem",

#### "entityId": "temp-123",

#### "data": {

#### "name": "Молоко",

#### "listId": "list-456",

#### "purchased": false

#### },

#### "timestamp": 1716981234567

#### }

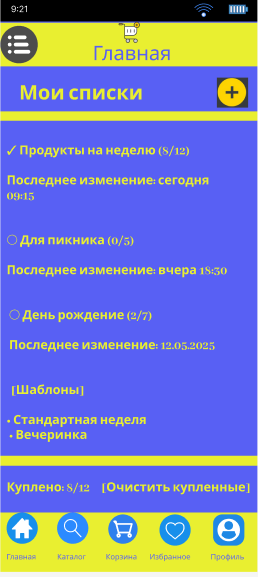
#### <https://www.plantuml.com/plantuml/png/bLHTJrCn6BxlhtY3YHUBp0dAnI8VQ90GII32XGboKSuADcodiwSB8OG4Hb0JDCOh4qD04t_0D4oGiV4Nsd_aysuiRZ9rRZbd_NW-dhvjDv4uQLsfa1V9kewBqie2hSXeVSsQKfnxQFB6qer76Vvs81BregJYID6M89gqrcnS8JEPuU6H4HwCdB4FZCqf-_lKh5vLipfMlMVd4KT7kYXZzm_wjPJDJxV4dq3RGZZjyee6_77OyOrGzWr_ZgT6OTUV-NhOvs5VH_lKLy9B_9vbwSdKmk9gAK-JypFaBtorIJi91mCYhrOTEKDMhprtbDDMHKwRM8WY4iWrDlheQr2fGhiyq2pSaZy4_Ii6Q-6DhwDT9Ny9NmoWyvsnp5X6t0wTpxuAp3T-N-bmd6dfr8RSNEO-83sN4DvXTIqsEhijx7jVuNdo9sZKGJi8-qt7lkFHuE1OTmZAqfqqpSyjBDBjP3EEdiLRfgYiv98aMLfAfzFBswAR0f7-ayDWA4t-4BCxuJMUlL3kGdCJOI43NaNqBymnUXU5JOTX-k6NCc4E8AuIwWE9MhieehV6_o_pBXm_GVKqxF4-X3AGvUkMeFQ_o9uRTWz6Ntm3e0k0e01mfRKOuWl29uxTUsvRloSbIoDf6ifaQEunpaJYf2ibjoWddSI1YurJj6ASCmKIHAcKF_AN7BYTe-FESJn2a6mg9QZZfD4W-ICmMhNXzuyErbykDrIcfo2IrubBnXCTHshqljC5XP25uXZh7eDJmqbxrLuFiG2herjjRp5xj5AVSuQshdQga8op-ankzEdDFWt2MzxVbaKeirwdV4MeE4TSDZ615lx5Vm80>

#### 3.4 ER-диаграмма сущностей

#### <https://www.plantuml.com/plantuml/png/dLJTQXD15BxVfpZ4cpOaK2dqOYcbDjbAj4rBarnu5SRic2pTtLbdPvNG5cogWbJnNh2I1n22DZRzINo5cJVop6oGZJO5xTBixFdEdFs-xvoTzLWGBfB0j-9zBum89m4y9QtzDcT9w9QOppZSsrpILoQ3SZYm03OOTocV9YqjBI-lhEZeNzlDdtLaMS8JFeMSirUKvt8a1_AxUeM_DtWDR91zTQBUoR6Sg6FLaoDGnt9abaDveqxruqyTa5To0BYSo2jrajvpLiTpARGytlAfPT4GtzI5dFo2uBLwBo_m_aqEJ2bypW698OcdClBWkJOq6fKohAxkFbvRmrYnQ00Q4C-teIQu5xPD929n_19rDpiaxcGGdyIY6NV3bWrrBw1eQn1PHnag_PIyPli7USF6zs8nbqtUa6rcWCqPciRN39aMfqHGjqb4bWr04hctnbtgqpG-Z_rNUQsxWFzZrPlfX67l2HhCPs_4pMSVaY1B_db2p5kpUf1rc_7kR2SIti8cK2oxmPXFIOZX5vJ77Wjjg8IuXV9_DYCh-WnL3lM4ghUezgEPHzJ_1aCJESRrHGh9aJ51Tx_9el_lOZT27vngOtkXjESyg3k5ncvPtyhEbbDt5arIIcwQjbMfrGkLkhEjiHJ1yXcNM4Gv4MX84xKI6nxLTgeOHgaYYQT5TfrgkL9zM83Qash9AHUWj5FTtAgKwejppKEJj1GuF2mMsK4wkpRaxeDW492mcxDCA8NDSCp2sTs_RPlDi48cA72ltH70dfaQMdzUzbLFVyXuL1phC-53PZ0HiFDvRIZ8p_gBLwVo4XQmLSFRUZV2jJaMv7WntNGcVo1oZf6XwjtHxZIxVrV5DEMJXjHhUOtm5PP430VA5Bx4END1gxJMSOL7y2y0>

#### 

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

**Экран 1: Главный экран (Мои списки)**

**Элементы:**

**1. AppBar:**

**-Меню навигации**: Доступ к профилю, настройкам синхронизации

**-Логотип приложения в виде тележки и название экрана приложения**

**2. Раздел “мои списки”:**

-Кнопка создания списка: Создание нового списка с возможностью выбора шаблона

**-Карточка списка:** Индикатор выполнения (✓/○)

-Название списка

-Прогресс (куплено/всего), очистить купленное

**-**Дата последнего изменения

**3. TabBar:**

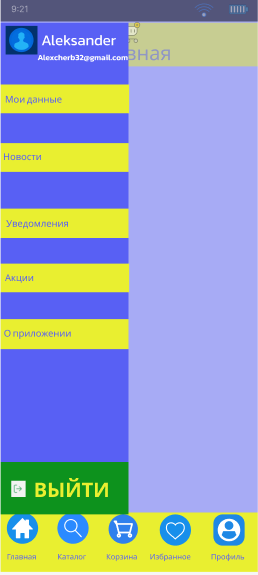
- Иконка перехода на главную страницу(текущую).

-Иконка перехода на страницу каталога.

**-**Иконка перехода на страницу корзины.

**-**Иконка перехода на страницу с избранным.

-Иконка перехода на страницу профиля.

**Экран 2: Экран профиля**

**Элементы:**

**1. Секция "Личные данные":**

* Имя (редактируемое поле)
* Email (с подтверждением)

**2. Секция** **"Новости":**

* Новые товары
* Новые списки
* Обновления приложения
* Отзывы

3. **Секция "Уведомления":**

* + Переключатели для разных типов уведомлений:
  + Напоминания о покупках (за час до обычного времени похода в магазин)
  + Изменения в общих списках (когда кто-то редактирует общий список)
  + Рекомендации и акции (маркетинговые рассылки)

4. **Секция "Акции и предложения":**

* Персональные скидки в магазинах (на основе истории покупок)
* Рекомендуемые товары (интеграция с партнерскими программами)
* Кнопка "Показать доступные предложения"

5. **Секция "О приложении":**

* Версия приложения
* Политика конфиденциальности
* Условия использования
* Обратная связь
* Кнопка "Рассказать о приложении"

**6.** **Действия:**

* Кнопка "Выйти из аккаунта"

**Ссылка на макет экранов приложения:** [**https://www.figma.com/design/9c1cQCHhEWcymm5JxzmBUZ/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%22%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BE%D0%BA-%D0%B2-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%D1%85%22%22?node-id=0-1&p=f&t=WXJQMZ62Sn8HF45m-0**](https://www.figma.com/design/9c1cQCHhEWcymm5JxzmBUZ/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%22%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BE%D0%BA-%D0%B2-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%D1%85%22%22?node-id=0-1&p=f&t=WXJQMZ62Sn8HF45m-0)

**Ссылка на экран 1:** [**https://www.figma.com/design/9c1cQCHhEWcymm5JxzmBUZ/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%22%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BE%D0%BA-%D0%B2-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%D1%85%22%22?node-id=2-25&m=dev&t=aOGoLGOazdwHZaWA-1**](https://www.figma.com/design/9c1cQCHhEWcymm5JxzmBUZ/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%22%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BE%D0%BA-%D0%B2-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%D1%85%22%22?node-id=2-25&m=dev&t=aOGoLGOazdwHZaWA-1)

**Ссылка на экран 2:** [**https://www.figma.com/design/9c1cQCHhEWcymm5JxzmBUZ/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%22%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BE%D0%BA-%D0%B2-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%D1%85%22%22?node-id=2-26&m=dev&t=aOGoLGOazdwHZaWA-1**](https://www.figma.com/design/9c1cQCHhEWcymm5JxzmBUZ/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5-%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%22%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BE%D0%BA-%D0%B2-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%D1%85%22%22?node-id=2-26&m=dev&t=aOGoLGOazdwHZaWA-1)

**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).**

### **Подробное описание функции покупки товара**

#### ****5.1 Текстовое описание процесса****

**Цель:** Позволить пользователю отметить товар как купленный, сохранить детали покупки и синхронизировать данные между устройствами.

**Основные шаги:**

1. **Выбор товара:**
   * Пользователь отмечает товар(ы) в активном списке покупок.
   * Поддерживается множественный выбор (пакетная обработка).
2. **Подтверждение покупки:**
   * Открывается форма с полями:
     + **Магазин** (выбор из сохраненных или ручной ввод).
     + **Цена за единицу** (руб).
     + **Количество** (автоподстановка из списка, с возможностью редактирования).
     + **Дата/время** (по умолчанию — текущие).
   * Опционально: добавление фото чека.
3. **Сохранение данных:**
   * Товар перемещается в раздел "История покупок".
   * Активный список покупок обновляется (удаляются купленные позиции).
4. **Синхронизация:**
   * Данные отправляются на сервер и сохраняются в облачной БД.
   * Все устройства пользователя получают обновленный список.

#### 5.2 UML Диаграмма последовательности

#### 

<https://www.plantuml.com/plantuml/png/ZLAxJZ915ExdAKwe8FzFl9GK1f7AGapm1KROH1B5pRBO61CkO8C9qSR6Q7Y35LaTBeklSEQD_6OsYyI4X6Rdp9blvhkSpJTywVdDgqiXA_wrH_p6Ivxh1_x2EkP0TpWqU-42LglKN5dtASKlE7rdPP4Hwf3uMxTC0vjFN8bGXocI3JfqtJ-NHnm2EpPVYoWxtetZYQhqvRbiE40ySW010V44HSIH7kY-XHOBGcoIIDa3GvOZVj83emua0KWw14ICRANr82C00dQZrLnyCZCQyD0uUQJ7Ixouq5ruDUUwYtN6QWj1pnm02n2i3ErhS8Q50xGKfNc1wiEQLXpz9trl8mnMMkE8SdHQAf_HZjlqAXScgTlZSkaa-viIXtSYXk8ImWBr2Ahxq6qGIdUWQaqzlg9OeEmQWpKpWJGx7ynoaaoJeCxOKBhDQaMJUDpVtQCZpv6-KzqYu5VC9xBTUH88Z0C0Jjq3RAhRb3PKw25sL4FwH-2VeQ5mpxYT4gJ4Wk0e8_9ElOfV-GS0>

**5.3 Диаграмма классов (Class Diagram**)

#### 

#### <https://www.plantuml.com/plantuml/png/bL71IiD04BtlLmmvAYtsceFsq4lFshE4TD504cLtSnAXLT2Bu0y8Wb-mr8Q6sDHVUFj7Pa1D6jf3oI4xR-QzsVTsg4sWJ9R48em3hGdlsE3BlM1L_nUmxW4vrtGdY8wZgK-JoNZ4PnHOkrUVpesAqYjnBzeAfNl4DqhO7RmbMDNwD0gCl8WIoQYRrunr_J4kmoW9OeQrkL7oiatFj5GDq6nzWyKdB5OeKEtOMd6F7Q52WVp_tjopMBePSZV_GyzQmX-_GSmE52qvI1JkcOLhmmjiV1gdXflk2JcgZjVRJ8NNWPPRxlQ-XpVmgDSx9U-atp_oEgcBxR8P7l1mozF-GQyJgHZAT9ebyGy0>

#### 5.4 API методы

**Endpoint:** POST /sync  
**Content-Type:** application/json

**Request:**

{

"deviceId": "a1b2c3d4",

"operations": [

{

"type": "UPDATE\_ITEM",

"entityId": 789,

"changes": {

"purchased": true

},

"timestamp": 1716980000,

"version": 24

}

]

}

**Response (Success):**

{

"status": "success",

"conflicts": [],

"serverTimestamp": 1716980050

}

**Response (Conflict):**

{

"status": "conflict",

"conflicts": [

{

"entityId": 789,

"serverVersion": 25,

"serverData": {

"purchased": false

}

}

]

}

### **5.5 Процесс хранения информации о покупках пользователя**

#### ****Локальное хранилище (SQLite на клиентском устройстве)****

**Таблица items:**

* **Назначение:** Основное хранилище элементов списка покупок.
* **Структура:**
  + id - уникальный идентификатор элемента (совпадает с серверным id)
  + list\_id - принадлежность к конкретному списку покупок
  + name - название товара
  + purchased - статус покупки (0 = не куплен, 1 = куплен)
  + version - текущая версия данных для контроля изменений
  + last\_updated - время последнего обновления в формате UNIX timestamp

**Таблица sync\_queue:**

* **Назначение:** Очередь несинхронизированных изменений.
* **Структура:**
  + operation\_type - тип операции (например, UPDATE\_ITEM)
  + entity\_type - тип сущности (всегда 'item')
  + entity\_id - ID изменяемого элемента
  + data - JSON с изменениями (например, {"purchased": true})
  + created\_at - время создания операции в UNIX timestamp

#### ****Серверное хранилище (PostgreSQL)****

**Таблица shopping\_items:**

* **Назначение:** Основное хранилище элементов списка покупок.
* **Структура:**
  + id - уникальный идентификатор (совпадает с клиентским id)
  + list\_id - связь со списком покупок (shopping\_lists)
  + name - название товара
  + purchased - статус покупки (true/false)
  + version - счетчик версий для контроля изменений
  + last\_updated - временная метка изменения

**Система версионирования:**

CREATE FUNCTION update\_version() RETURNS TRIGGER AS $$

BEGIN

NEW.version = OLD.version + 1; -- Автоинкремент версии

NEW.last\_updated = NOW(); -- Обновление временной метки

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER item\_version\_trigger

BEFORE UPDATE ON shopping\_items

FOR EACH ROW EXECUTE FUNCTION update\_version();

### **Жизненный цикл данных**

#### ****Пользователь изменяет статус покупки (пример):****

1. Клиент обновляет локальную запись:

UPDATE items

SET purchased = 1,

version = version + 1,

last\_updated = 1716980000

WHERE id = 789;

Добавляет операцию в очередь синхронизации:

1. INSERT INTO sync\_queue (operation\_type, entity\_type, entity\_id, data, created\_at)

1. VALUES (
2. 'UPDATE\_ITEM',
3. 'item',
4. 789,
5. '{"purchased": true}',
6. 1716980000
7. );

#### ****Синхронизация с сервером****

Клиент отправляет запрос:

POST /sync

{

"deviceId": "a1b2c3d4",

"operations": [

{

"type": "UPDATE\_ITEM",

"entityId": 789,

"changes": {"purchased": true},

"timestamp": 1716980000,

"version": 24 // Версия ДО изменения

}

]

}

#### ****Обработка на сервере****

1. Проверка актуальности версии:

SELECT version FROM shopping\_items WHERE id = 789;

* + Если серверная версия == 24:
    - Применяет изменения
    - Триггер автоматически обновляет version=25 и last\_updated
  + Если серверная версия > 24:
    - Возвращает конфликт

#### ****Реакция клиента на ответ****

**При успехе:**

{

"status": "success",

"serverTimestamp": 1716980050

}

Действия клиента:

1. Обновляет локальную запись:
2. UPDATE items
3. SET version = 25,
4. last\_updated = 1716980050
5. WHERE id = 789;
6. Удаляет операцию из sync\_queue

**При конфликте:**

{

"status": "conflict",

"conflicts": [

{

"entityId": 789,

"serverVersion": 25,

"serverData": {"purchased": false}

}

]

}

Действия клиента:

1. Обновляет локальные данные:

UPDATE items

SET purchased = 0,

version = 25,

last\_updated = 1716980050

WHERE id = 789;

Удаляет конфликтную операцию из sync\_queue

Уведомляет пользователя о расхождении данных

**Локальное хранилище (SQLite):**

CREATE TABLE items (

id INTEGER PRIMARY KEY,

list\_id INTEGER,

name TEXT NOT NULL,

purchased BOOLEAN DEFAULT 0,

version INTEGER DEFAULT 0,

last\_updated INTEGER -- UNIX timestamp

);

CREATE TABLE sync\_queue (

id INTEGER PRIMARY KEY AUTOINCREMENT,

operation\_type TEXT NOT NULL, -- 'UPDATE\_ITEM'

entity\_type TEXT NOT NULL, -- 'item'

entity\_id INTEGER NOT NULL,

data TEXT NOT NULL, -- JSON

created\_at INTEGER

);

**Серверное хранилище (PostgreSQL):**

CREATE TABLE shopping\_items (

id SERIAL PRIMARY KEY,

list\_id INTEGER REFERENCES shopping\_lists(id),

name VARCHAR(255) NOT NULL,

purchased BOOLEAN DEFAULT false,

version INTEGER NOT NULL DEFAULT 0,

last\_updated TIMESTAMP NOT NULL

);

-- Система версий для оптимистичной блокировки

CREATE FUNCTION update\_version() RETURNS TRIGGER AS $$

BEGIN

NEW.version = OLD.version + 1;

NEW.last\_updated = NOW();

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER item\_version\_trigger

BEFORE UPDATE ON shopping\_items

FOR EACH ROW EXECUTE FUNCTION update\_version();

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

#### 6.1 Технические сложности:

1. **Конфликт-менеджмент синхронизации:**
   * Разрешение коллизий при одновременном редактировании
   * Стратегии слияния изменений (последнее изменение vs ручное разрешение)
   * Потерянные операции при длительном офлайн-режиме
2. **Производительность больших списков:**
   * Оптимизация рендеринга 500+ товаров
   * Ленивая загрузка элементов
   * Индексация локальной БД
3. **Кросс-платформенные особенности:**
   * Различия в поведении SQLite на iOS/Android
   * Фоновая синхронизация: WorkManager (Android) vs Background Tasks (iOS)
   * Потребление батареи при постоянной синхронизации
4. **Безопасность данных:**
   * Шифрование локального хранилища
   * Защита PII (персональных данных) в синхронизации
   * Предотвращение инъекций через API
5. **Тестирование edge-кейсов:**
   * Синхронизация при переключении сетей (WiFi → 4G)
   * Одновременное редактирование на 3+ устройствах
   * Сценарии с переполнением очереди операций

#### 6.2 Вопросы к заказчику:

1. **Бизнес-логика:**
   * Нужна ли история изменений товаров (кто и когда пометил)?
   * Требуется ли ограничение на размер списка (макс. товаров)?
   * Должны ли купленные товары автоматически архивироваться?
2. **Синхронизация:**
   * Приоритет: скорость синхронизации vs экономия трафика?
   * Допустимая задержка синхронизации в офлайн-режиме?
   * Как обрабатывать конфликты: автоматически или с участием пользователя?
3. **Безопасность:**
   * Требуется ли двухфакторная аутентификация?
   * Нужно ли соответствие GDPR для европейских пользователей?
   * Политика хранения данных: сроки автоматического удаления старых списков?
4. **Интеграции:**
   * Планируется ли интеграция с умными холодильниками?
   * Нужен ли экспорт в Excel/Google Keep?
   * Возможность сканирования штрих-кодов?
5. **Монетизация:**
   * Планируются ли платные функции (e.g., неограниченная история)?
   * Допустима ли реклама в бесплатной версии?
   * Партнерские программы с магазинами?
6. **UX-нюансы:**
   * Анимации при отметке: обязательны или опциональны?
   * Поддержка для слабовидящих: высокий приоритет?
   * Кастомизация: возможность менять иконки/цвета?
7. **Юридические аспекты:**
   * Нужен ли возрастной рейтинг (при алкогольных товарах)?
   * Региональные ограничения для функций шаринга?
   * Политика модерации пользовательского контента?

**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**

CREATE TABLE Authors (

Id INTEGER PRIMARY KEY,

AuthorName VARCHAR(50) NOT NULL

);

INSERT INTO Authors (Id, AuthorName) VALUES

(1, 'A.C.Пушкин'),

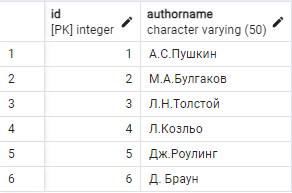
(2, 'М.А.Булгаков'),

(3, 'Л.Н.Толстой'),

(4, 'Л.Козльо'),

(5, 'Дж.Роулинг'),

(6, 'Д. Браун');



CREATE TABLE Books (

Id SERIAL PRIMARY KEY,

AuthorId INTEGER REFERENCES Authors(Id),

BookName VARCHAR(255) NOT NULL,

Price INTEGER NOT NULL

);

INSERT INTO Books (AuthorId, BookName, Price) VALUES

(1, 'Евгений Онегин', 500),

(1, 'Капитанская дочка', 1000),

(2, 'Мастер и Маргарита', 2000),

(3, 'Война и мир', 1050),

(3, 'Анна Каренина', 550),

(4, 'Алхимик', 700),

(5, 'Гарри Поттер', 2900);



**7.1 Общая стоимость книг для каждого автора и отсортировать результат в порядке убывания:**

SELECT

a.AuthorName,

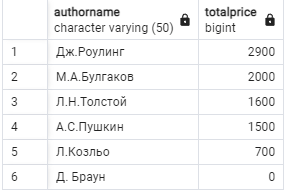
COALESCE(SUM(b.Price), 0) AS TotalPrice

FROM Authors a

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

GROUP BY a.Id, a.AuthorName

ORDER BY TotalPrice DESC;



**7.2 Стоимость книг автора превышает 1500**

SELECT

a.AuthorName,

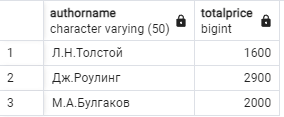
SUM(b.Price) AS TotalPrice

FROM Authors a

JOIN Books b ON a.Id = b.AuthorId

GROUP BY a.Id, a.AuthorName

HAVING SUM(b.Price) > 1500;



**7.3 Вывести авторов с количеством книг**

SELECT

a.AuthorName,

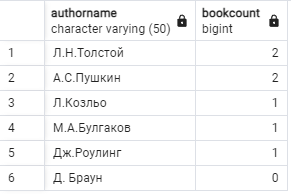
COUNT(b.Id) AS BookCount

FROM Authors a

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

GROUP BY a.Id, a.AuthorName

ORDER BY BookCount DESC;



**7.4 Получить автора без книг**

SELECT

a.AuthorName

FROM Authors a

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

WHERE b.Id IS NULL;

