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

**Ключевые сущности**

**Пользователь (User)**

**Представляет человека, работающего с приложением.**

**Список покупок (Shopping List)**

**Содержит набор товаров, которые нужно приобрести.**

**Товар (Item/Product)**

**Единица, которая добавляется в список. У товара есть информация: имя, количество, статус "куплено/не куплено", возможность редактирования.**

**🔗**

**Взаимодействия между сущностями**

**Пользователь создает (или выбирает существующий) Список покупок.**

**Внутри Списка пользователь добавляет, редактирует, отмечает Товары (products).**

**Для каждого Списка пользователь может:**

**Добавлять новые товары;**

**Отмечать отдельные товары "купленным";**

**Завершить (закрыть) список покупок — при этом вся информация попадает в Историю.**

**Связь:**

**User 1 - M ShoppingList (один пользователь — много списков)**

**ShoppingList 1 - M Item (один список — много товаров)**

**📘**

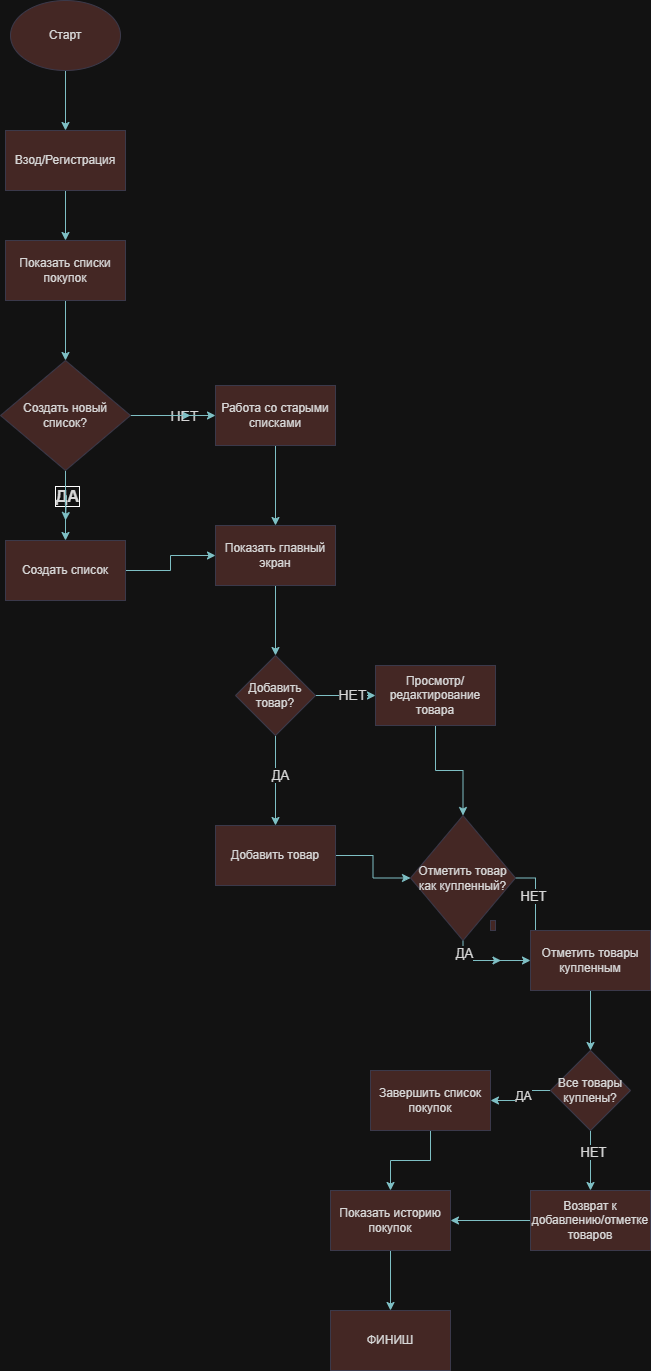
**Описание связи по схеме**

**После входа/регистрации пользователь видит свои списки (история списков хранится за ним!)**

**При выборе/создании списка, он работает уже с конкретным списком и товарами в нём:**

**— добавляет, отмечает купленные, завершает и сохраняет в историю.**

**Вся работа — это взаимодействие трёх сущностей: User → ShoppingList → Item**

****

**2. Основные функции приложения (в порядке убывания важности для пользователя):**

1. **Создание и ведение списков покупок** — пользователь может создавать новые списки покупок для разных магазинов или случаев.
2. **Добавление товаров в список** — возможность быстро добавить продукт в нужный список покупок.
3. **Отметка товаров как купленных** — удобно отмечать покупки как завершённые (чек-лист).
4. **Редактирование и удаление товаров/списков** — можно изменить количество, название или полностью удалить товар/список.
5. **Синхронизация данных между устройством и сервером** — автоматическое сохранение прогресса и доступ с разных устройств.
6. **Просмотр истории или завершённых списков** — возвращаться к предыдущим покупкам можно быстро.
7. **Сортировка товаров внутри списка** — по алфавиту, времени добавления или другим параметрам.
8. **Пуш-уведомления с напоминаниями** — возможность получать напоминания о запланированных покупках.
9. **Возможность поделиться списком** — переслать список другому человеку (через мессенджер или ссылку).

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

Описание процесса:

Пользователь выполняет действие в приложении (например: добавляет или удаляет список, добавляет, меняет или удаляет товар, отмечает товар как купленный).

Мобильное приложение отправляет соответствующий запрос к серверу через API.

Сервер фиксирует изменения в базе данных.

Сервер возвращает обновлённые данные или подтверждение обратно на мобильное приложение.

Мобильное приложение обновляет отображение списков и товаров для пользователя.

Примеры API методов:

GET /api/lists — получить списки пользователя

POST /api/lists — создать новый список

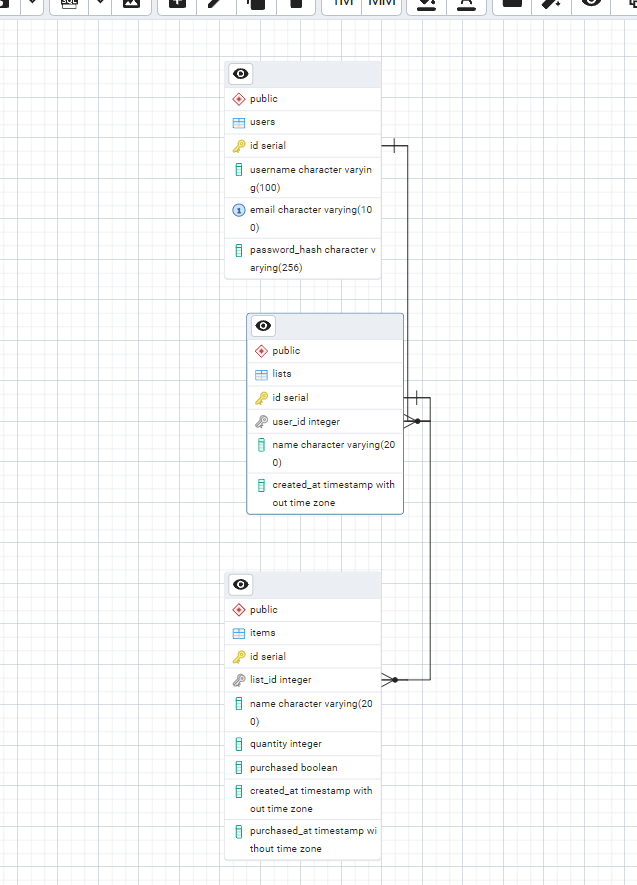
DELETE /api/lists/{listId} — удалить список

POST /api/lists/{listId}/items — добавить товар

PATCH /api/lists/{listId}/items/{itemId} — редактировать товар

DELETE /api/lists/{listId}/items/{itemId} — удалить товар

POST /api/lists/{listId}/items/{itemId}/check — отметить товар как купленный



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

* **≡** — “гамбургер-меню”, три полоски слева сверху, открывает боковое меню (настройки, профиль, другие списки).
* **Магазин Пятёрочка** — название текущего списка покупок.
* **[ ]** — пустой чекбокс, товар не куплен.
* **[✔]** — отмеченный чекбокс, товар уже купили.
* **Хлеб / Молоко / Яйцо** — название товара.
* **1 шт. / 2 л / 10 шт.** — количество и единицы измерения товара.
* **⠇** (три вертикальные точки или …) — кнопка для дополнительных действий с товаром (редактировать, удалить и др.).
* **+** — кнопка “Добавить товар”, обычно справа внизу (FAB — плавающая кнопка).
* **Добавить товар** — текстовая кнопка для добавления нового товара в список.

**5. Подробное описание функции покупки товара (отметить товар купленным)**

**1.1 Описание функции**

Пользователь может в любой момент отметить товар в списке покупок как купленный, нажав на чекбокс рядом с названием товара. После этого у товара меняется статус: он может отображаться зачёркнутым или бледным, а галочка становится активной. Изменение автоматически передаётся на сервер, чтобы обновить данные во всех устройствах пользователя.

Логика работы

Пользователь нажимает чекбокс товара.

Приложение отправляет запрос на сервер, чтобы обновить статус товара на «куплен».

Сервер обновляет запись товара в базе данных.

Сервер возвращает приложению подтверждение и обновлённые данные по этому товару.

Приложение обновляет интерфейс: отмечает чекбокс, выводит «куплен» (зачеркивает или делает серым).

Пример использования (User Flow)

👤 Пользователь: «Поставить галочку на молоке».

📱 Приложение: отправить запрос на сервер.

🌍 Сервер: сохранить новое состояние.

📱 Приложение: показать, что «Молоко» теперь куплено.

Хранение в базе данных

В таблице товаров (items) есть поле purchased (булево: да/нет) и временая метка purchased\_at (дата покупки).

Пример API-запроса

Метод:

PATCH /api/lists/{listId}/items/{itemId}

Тело запроса:

{

"purchased": true

}

Пример ответа:

{

"id": 12,

"list\_id": 3,

"name": "Молоко",

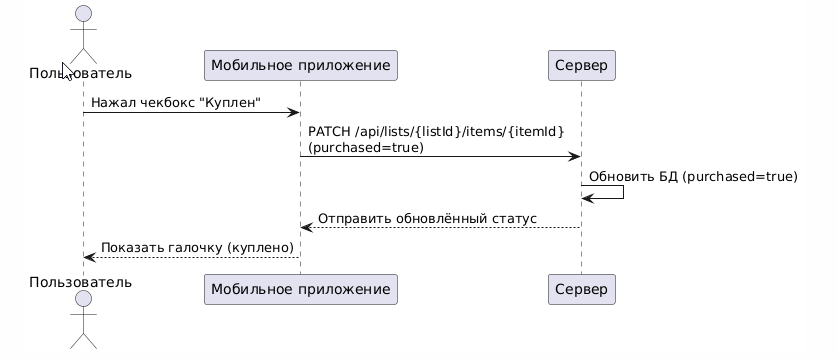
"quantity": "2 л",

"purchased": true,

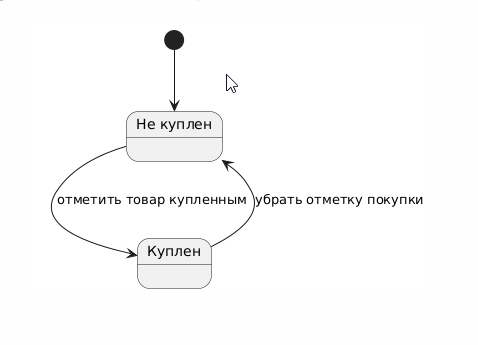
"purchased\_at": "2024-06-22T11:23:10Z"

}

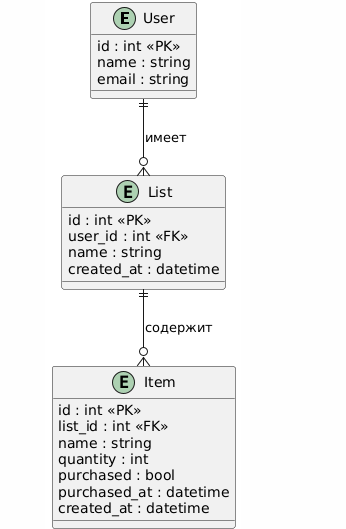
**1.2**

**Диаграмма последовательности**

**1.3 Диаграмма состояний**



**1.4 ER-диаграмма сущностей**



**6. Основные сложности разработки и вопросы заказчику**

🚧

Основные сложности разработки такого приложения:

Сложности синхронизации данных

Необходимо надёжно обрабатывать ситуацию, когда пользователь вносит изменения в список с нескольких устройств, а также предусмотреть работу с плохим интернетом или офлайн.

Офлайн-режим

Нужно реализовать локальное хранение и последующую синхронизацию изменений, что усложняет архитектуру приложения.

Безопасность данных

Важно обеспечить безопасную аутентификацию пользователя и хранение его списков.

Удобство пользовательского интерфейса (UX/UI)

Необходимо сделать приложение максимально быстрым и интуитивным для быстрого добавления и отметки покупок.

Масштабируемость и производительность серверной части

Сервер должен работать стабильно, даже при большом количестве пользователей и данных.

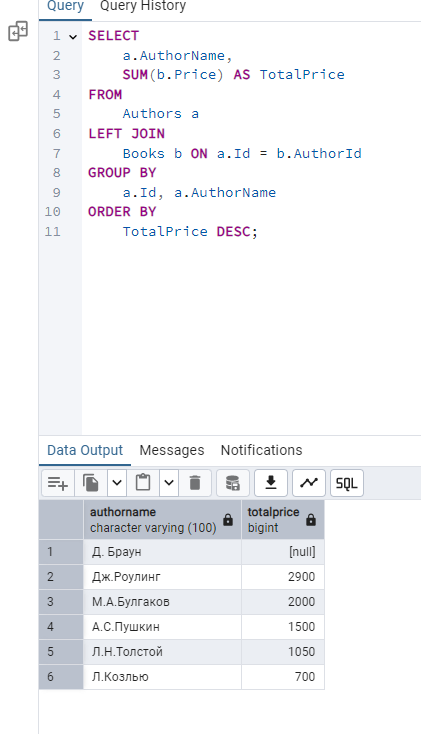
Работа с напоминаниями и уведомлениями

Важно реализовать пуш-уведомления так, чтобы они не раздражали пользователя.

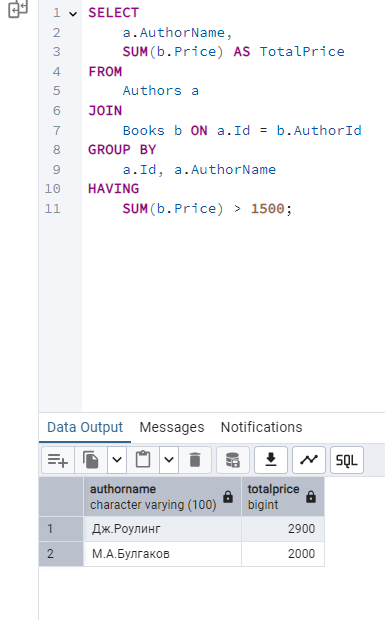
Межплатформенность

При необходимости поддержки Android и iOS возможны сложности в синхронизации и унификации поведения приложения.

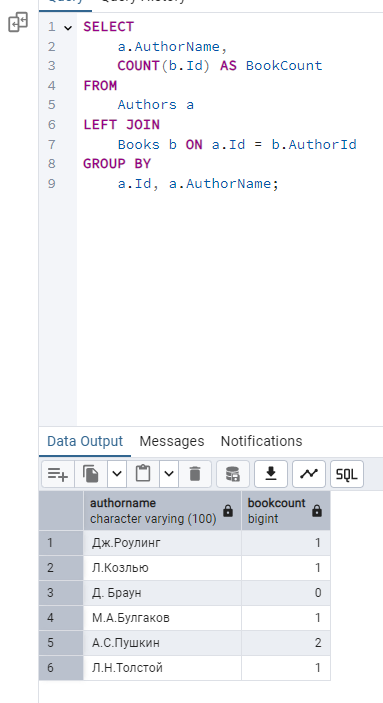
**7. Есть таблицы Books и Authors, где AuthorId табл. Books равно Id табл. Authors. Необходимо написать SQL-запрос, чтобы найти: 1. Общую стоимость книг для каждого автора и отсортировать результат в порядке убывания;**



2. Стоимость книг автора превышает 1500;



3. Вывести авторов с количеством книг;



4. Получить автора без книг

