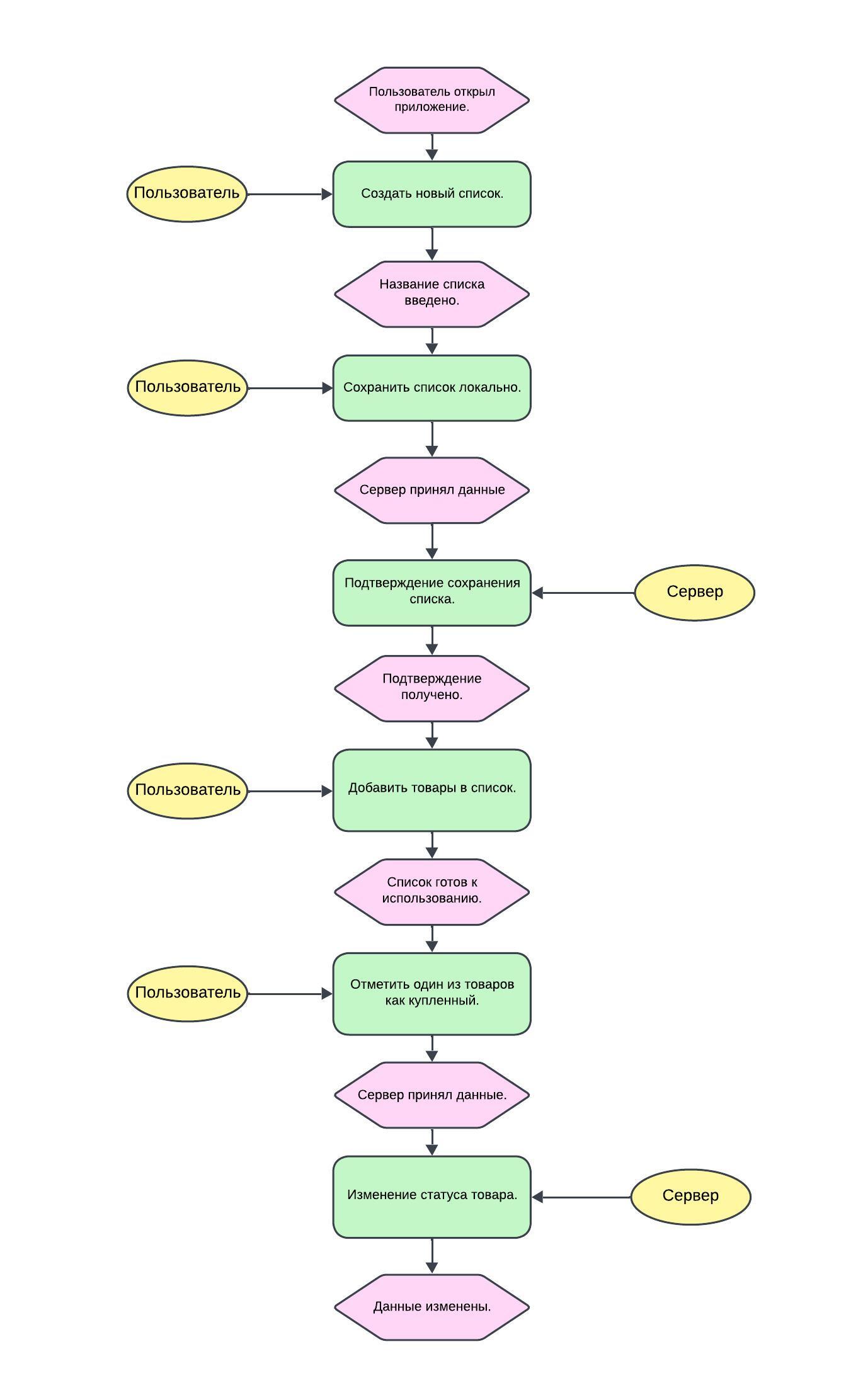
Осягиной Е.В. ИСдо-34

Задание 1.

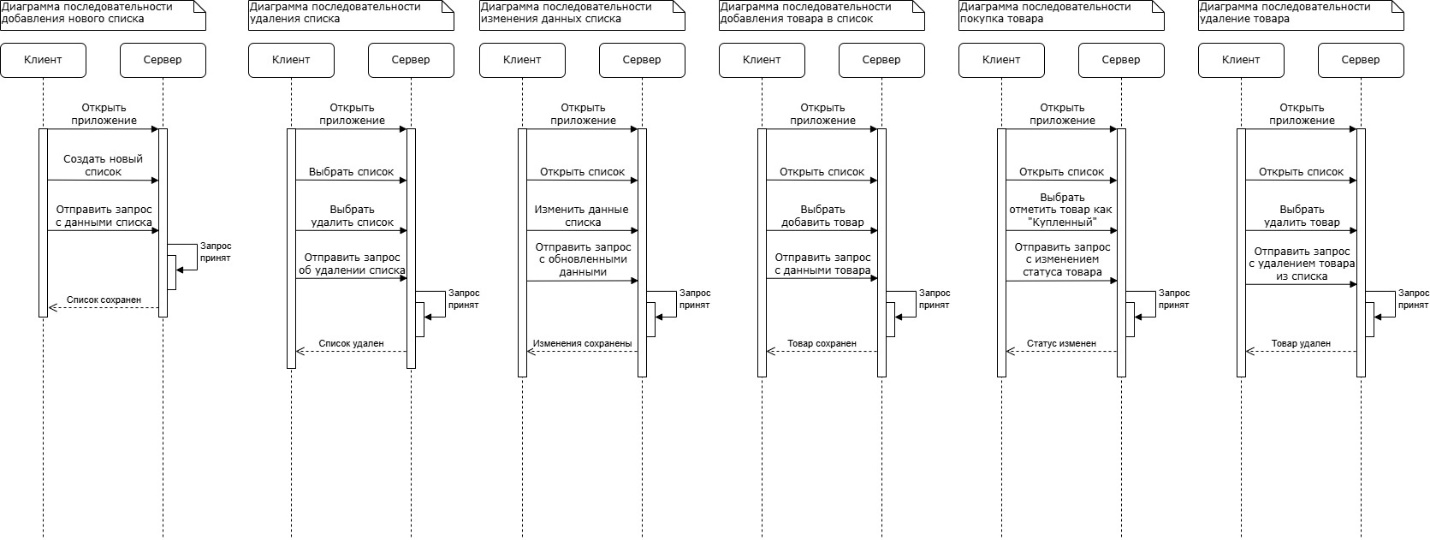


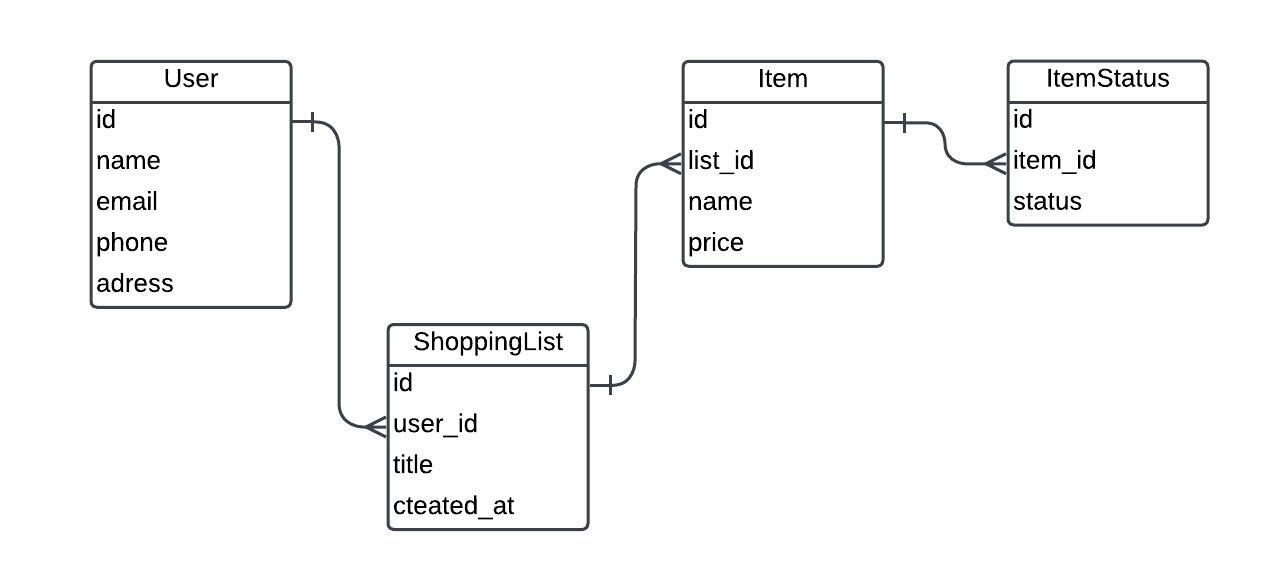
Задание 2.

Основные функции:

1. Создание списков покупок
   * Возможность создать новый список покупок, указав название и описание.
   * Это базовая функция, необходимая для начала работы.
2. Добавление товаров в список
   * Указание названия, количества, единицы измерения и заметки к каждому товару.
   * Позволяет пользователю организовать процесс покупок.
3. Отметка товаров как купленных
   * Функция отметки купленных товаров (например, установка галочки).
   * Удобство отслеживания, что уже приобретено.
4. Редактирование списка
   * Возможность изменять название списка, удалять товары или редактировать их параметры.
   * Полезно при изменении планов покупок.
5. Синхронизация данных между устройствами
   * Автоматическая передача данных на сервер и обратно.
   * Гарантирует доступ к актуальной информации с разных устройств.
6. Удаление списков и товаров
   * Удаление ненужных списков или отдельных товаров из них.
   * Позволяет поддерживать порядок в приложении.
7. Поиск по списку
   * Функция поиска товаров в большом списке.
   * Экономит время, особенно при работе с длинными списками.
8. Фильтрация товаров
   * Удобство управления списком в процессе покупок.
9. Поддержка совместного использования списков
   * Опциональная функция, позволяющая делиться списками с другими пользователями.
   * Удобно для семейного или группового использования.

Задание 3.





API Методы:

Действия со списком:

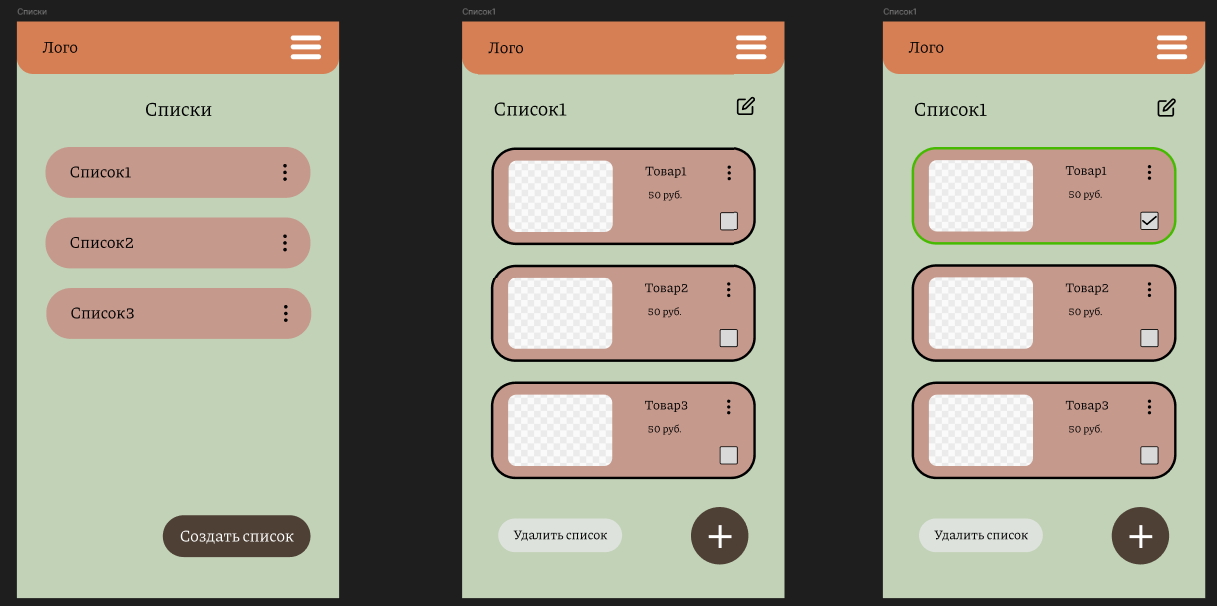
* Создать список: POST /lists
* Обновить название списка: PUT /lists/{id}
* Удалить список DELETE: /lists/{id}

Действия с товаром:

* Добавить товар в список: POST /lists/{id}/items
* Обновить товар: PUT /lists/{id}/items/{item\_id}
* Отметить товар купленным: PATCH /lists/{id}/items/{item\_id}
* Удалить товар из списка: DELETE /lists/{id}/items/{item\_id}

Задание 4.

<https://www.figma.com/design/svEFrVuS8LeMkf46MD5RBj/%D0%97%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-3.-MediaSoftPractice?node-id=1-2&p=f&t=2tReyyrxyXG3xjgQ-0>



Экран: "Просмотр всех списков"

**Описание интерфейса:**

1. Верхняя часть:
   * Логотип.
   * Кнопка меню-бургер.
2. Основная часть:
   * Название страницы.
   * Существующие списки:
     + Название списка.
     + Кнопка меню действий.
3. Нижняя часть:
   * Кнопка "Создать список".

Экран: "Просмотр списка покупок"

**Описание интерфейса:**

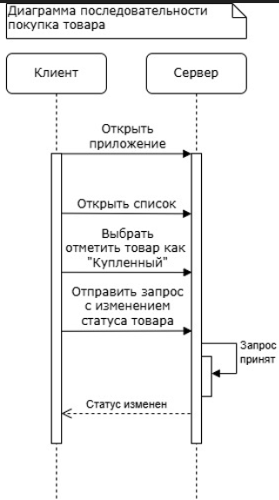
1. Верхняя часть:
   * Логотип.
   * Кнопка меню-бургер.
2. Основная часть:
   * Название списка (редактируемое поле).
   * Список товаров:
     + Название товара.
     + Цена товара.
     + Кнопка меню действий.
     + Кнопка для отметки "куплено" (чекбокс).
3. Нижняя часть:
   * Кнопка "Удалить список".
   * Кнопка добавления нового товара ("+").

Задание 5.

Функция "Покупка товара" предназначена для изменения статуса товара в списке покупок на «Куплено». Это позволяет пользователю отслеживать уже приобретённые товары. Процесс включает взаимодействие клиента (мобильного приложения) с сервером для обновления данных в базе.

### **Бизнес-логика функции**

1. Пользователь отмечает товар как купленный (например, нажимает чекбокс рядом с названием товара).
2. Приложение отправляет запрос на сервер для обновления статуса конкретного товара.
3. Сервер обновляет статус товара в базе данных и возвращает подтверждение.
4. Интерфейс приложения обновляется (помечает товар как купленный, выделяет его визуально).



#### **Описание API методов**

##### **1. API Метод: Изменение статуса товара**

* **Метод:** PUT /items/{item\_id}
* **Описание:** Обновляет статус конкретного товара в списке.
* **Входные параметры:**

{

"status": "purchased"

}

* **Ответ:**

{

"item\_id": 12345,

"status": "purchased",

"updated\_at": "2025-02-11T14:23:00Z"

}

Задание 6.

### **Основные сложности разработки приложения:**

1. **Синхронизация данных**
   * **Описание:**  
     Обеспечение бесшовной синхронизации данных между клиентом и сервером, особенно в случае нестабильного интернет-соединения.
   * **Сложности:**
     + Конфликт изменений: например, если два пользователя одновременно редактируют один и тот же список.
     + Производительность при большом объёме данных (много списков/товаров).
     + Обработка ошибок синхронизации (восстановление после разрыва соединения).
2. **Поддержка работы оффлайн**
   * **Описание:**  
     Возможность работы с приложением в режиме оффлайн (добавление, редактирование и удаление данных), с последующей синхронизацией при восстановлении соединения.
   * **Сложности:**
     + Хранение локальной копии данных.
     + Обработка конфликтов между локальными и серверными данными.
     + Тестирование работы оффлайн/онлайн на различных устройствах.
3. **Интуитивно понятный интерфейс**
   * **Описание:**  
     Дизайн должен быть удобным для пользователя с минимальными действиями для выполнения основных операций.
   * **Сложности:**
     + Организация работы с большими списками.
     + Удобство взаимодействия при добавлении/редактировании товаров (например, быстрое добавление через автозаполнение или шаблоны).
4. **Поддержка совместного использования списков (опционально)**
   * **Описание:**  
     Возможность делиться списками с другими пользователями.
   * **Сложности:**
     + Управление доступами (просмотр, редактирование).
     + Отслеживание изменений, сделанных разными пользователями.
     + Обеспечение безопасности данных.
5. **Тестирование на различных устройствах**
   * **Описание:**  
     Приложение должно одинаково хорошо работать на разных операционных системах (iOS, Android) и экранах.
   * **Сложности:**
     + Адаптивность интерфейса.
     + Возможные различия в поведении на устройствах разных производителей.
6. **Обеспечение безопасности данных**
   * **Описание:**  
     Данные списков и товаров, особенно при совместном использовании, должны быть надёжно защищены.
   * **Сложности:**
     + Защита передаваемых данных (шифрование).
     + Безопасное хранение данных пользователя (например, паролей).
     + Устойчивость к атакам (например, SQL-инъекции).

### **Вопросы к заказчику:**

1. **О функционале:**
   * Требуется ли поддержка совместного использования списков между пользователями?
   * Нужно ли предусмотреть возможность дублирования списка?
   * Должно ли приложение отправлять уведомления (например, напоминания о покупках)?
2. **Оффлайн-режим:**
   * Должен ли оффлайн-режим поддерживать весь функционал (создание, редактирование, удаление), или достаточно минимальных функций?
   * Есть ли ограничения по объёму данных, которые можно хранить локально?
3. **О синхронизации данных:**
   * Какие действия должны выполняться автоматически (например, синхронизация при запуске приложения)?
   * Нужно ли предоставлять пользователю возможность выбора, какие изменения синхронизировать?
4. **О дизайне интерфейса:**
   * Нужен ли отдельный экран для просмотра всех товаров в списке?
   * Должен ли быть реализован поиск по товарам или спискам?
5. **О дополнительных функциях:**
   * Нужна ли система учёта расходов (сумма, подсчёт итоговой стоимости списка)?
   * Нужно ли отслеживать историю покупок?
6. **О производительности и масштабируемости:**
   * Есть ли ограничения на количество списков и товаров для одного пользователя?
   * Планируется ли в будущем расширение функционала?
7. **О безопасности:**
   * Какие конкретные требования по защите данных (шифрование, двухфакторная аутентификация)?
   * Нужно ли предусмотреть возможность восстановления данных?

Задание 7.

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

SELECT

a.Id AS AuthorId,

a.Name AS AuthorName,

SUM(b.Price) AS TotalPrice

FROM Authors a

JOIN Books b ON a.Id = b.AuthorId

GROUP BY a.Id, a.Name

ORDER BY TotalPrice DESC;

**2. Авторы, у которых стоимость книг превышает 1500.**

SELECT

a.Id AS AuthorId,

a.Name AS AuthorName,

SUM(b.Price) AS TotalPrice

FROM Authors a

JOIN Books b ON a.Id = b.AuthorId

GROUP BY a.Id, a.Name

HAVING SUM(b.Price) > 1500;

**3. Количество книг у каждого автора.**

SELECT

a.Id AS AuthorId,

a.Name AS AuthorName,

COUNT(b.Id) AS BookCount

FROM Authors a

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

GROUP BY a.Id, a.Name;

**4. Авторы, у которых нет книг**.

SELECT

a.Id AS AuthorId,

a.Name AS AuthorName

FROM Authors a

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

WHERE b.Id IS NULL;