Задание на производственную практику

по направлению “Аналитика”

от компании Медиасофт

Выполнил Васильев Степан Алексеевич ИСдо-33

2025

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

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

**Общее описание:** Мобильное приложение «Список покупок в магазинах» позволяет пользователям создавать и управлять списками покупок, добавлять продукты, указывать их количество и цену, распределять продукты по магазинам, а также отмечать товары как купленные.

**1. Объекты (Entity):**

* Пользователь (User):
  + Идентификатор (UserID)
  + Имя (Username)
  + Пароль (Password)
  + Email
* Список покупок (ShoppingList):
  + Идентификатор (ListID)
  + Название списка (ListName)
  + UserID (Внешний ключ к пользователю)
  + Дата создания (CreationDate)
  + Дата последнего изменения (LastModifiedDate)
* Магазин (Store):
  + Идентификатор (StoreID)
  + Название магазина (StoreName)
  + UserID (Внешний ключ к пользователю)
* Продукт (Product):
  + Идентификатор (ProductID)
  + Название продукта (ProductName)
  + Описание (Description)
* Элемент списка покупок (ShoppingListItem):
  + ListID (Внешний ключ к ShoppingList)
  + ProductID (Внешний ключ к Product)
  + StoreID (Внешний ключ к Store)
  + Количество (Quantity)
  + Цена (Price)
  + Куплено (IsPurchased)
  + Дата добавления (AddedDate)

**2. Бизнес-процессы (User Stories):**

**2.1 Аутентификация и авторизация:**

1. Регистрация пользователя:
   * Пользователь вводит данные (имя, email, пароль).
   * Приложение валидирует данные (формат email, сложность пароля).
   * Приложение отправляет данные на сервер.
   * Сервер проверяет уникальность email.
   * Сервер создает новую запись пользователя в базе данных.
   * Сервер возвращает подтверждение успешной регистрации.
2. Вход пользователя:
   * Пользователь вводит email и пароль.
   * Приложение отправляет данные на сервер.
   * Сервер проверяет наличие пользователя с указанным email и паролем.
   * Если данные верны, сервер возвращает токен аутентификации.
   * Приложение сохраняет токен для последующих запросов.
3. Выход пользователя:
   * Приложение удаляет токен аутентификации.

**2.2 Управление списками покупок:**

1. Создание списка покупок:
   * Пользователь нажимает кнопку “Создать список”.
   * Пользователь вводит название списка.
   * Приложение отправляет данные на сервер.
   * Сервер создает новую запись списка покупок в базе данных, связывая ее с UserID.
   * Сервер возвращает ID созданного списка.
2. Просмотр списка покупок:
   * Пользователь выбирает список из списка доступных.
   * Приложение запрашивает данные списка с сервера (название, элементы списка).
   * Сервер возвращает данные списка.
   * Приложение отображает название списка и элементы списка.
3. Редактирование названия списка:
   * Пользователь выбирает опцию “Редактировать название”.
   * Пользователь вводит новое название.
   * Приложение отправляет данные на сервер.
   * Сервер обновляет название списка в базе данных.
   * Сервер возвращает подтверждение успешного обновления.
4. Удаление списка покупок:
   * Пользователь выбирает опцию “Удалить список”.
   * Приложение запрашивает подтверждение удаления.
   * После подтверждения приложение отправляет запрос на сервер.
   * Сервер удаляет список из базы данных.
   * Сервер возвращает подтверждение успешного удаления.

**2.3 Управление элементами списка покупок:**

1. Добавление продукта в список:
   * Пользователь выбирает опцию “Добавить продукт”.
   * Пользователь вводит название продукта или выбирает из списка существующих (поиск по названию).
   * Пользователь указывает количество и цену (опционально).
   * Пользователь выбирает магазин (если магазины созданы).
   * Приложение отправляет данные на сервер.
   * Сервер создает новую запись элемента списка покупок в базе данных, связывая ее с ListID, ProductID, StoreID, количеством и ценой.
   * Сервер возвращает подтверждение успешного добавления.
2. Редактирование элемента списка:
   * Пользователь выбирает элемент списка.
   * Пользователь изменяет количество, цену или магазин.
   * Приложение отправляет данные на сервер.
   * Сервер обновляет запись элемента списка покупок в базе данных.
   * Сервер возвращает подтверждение успешного обновления.
3. Отметка продукта как купленного:
   * Пользователь нажимает на чекбокс рядом с продуктом.
   * Приложение отправляет данные на сервер.
   * Сервер обновляет поле “IsPurchased” в базе данных.
   * Сервер возвращает подтверждение успешного обновления.
4. Удаление продукта из списка:
   * Пользователь выбирает опцию “Удалить” для продукта.
   * Приложение запрашивает подтверждение удаления.
   * После подтверждения приложение отправляет запрос на сервер.
   * Сервер удаляет элемент списка покупок из базы данных.
   * Сервер возвращает подтверждение успешного удаления.

**2.4 Управление магазинами:**

1. Создание магазина:
   * Пользователь переходит в раздел “Магазины”.
   * Пользователь нажимает кнопку “Создать магазин”.
   * Пользователь вводит название магазина.
   * Приложение отправляет данные на сервер.
   * Сервер создает новую запись магазина в базе данных, связывая ее с UserID.
   * Сервер возвращает ID созданного магазина.
2. Просмотр списка магазинов:
   * Пользователь переходит в раздел “Магазины”.
   * Приложение запрашивает список магазинов с сервера.
   * Сервер возвращает список магазинов, связанных с UserID.
   * Приложение отображает список магазинов.
3. Редактирование названия магазина:
   * Пользователь выбирает магазин из списка.
   * Пользователь выбирает опцию “Редактировать название”.
   * Пользователь вводит новое название.
   * Приложение отправляет данные на сервер.
   * Сервер обновляет название магазина в базе данных.
   * Сервер возвращает подтверждение успешного обновления.
4. Удаление магазина:
   * Пользователь выбирает магазин из списка.
   * Пользователь выбирает опцию “Удалить магазин”.
   * Приложение запрашивает подтверждение удаления.
   * После подтверждения приложение отправляет запрос на сервер.
   * Сервер удаляет магазин из базы данных.
   * Сервер возвращает подтверждение успешного удаления.

**3. Модель данных:**

classDiagram

class User {

UserID : int

Username : string

Password : string

Email : string

}

class ShoppingList {

ListID : int

ListName : string

UserID : int

CreationDate : datetime

LastModifiedDate : datetime

}

class Store {

StoreID : int

StoreName : string

UserID : int

}

class Product {

ProductID : int

ProductName : string

Description : string

}

class ShoppingListItem {

ListID : int

ProductID : int

StoreID : int

Quantity : int

Price : decimal

IsPurchased : bool

AddedDate : datetime

}

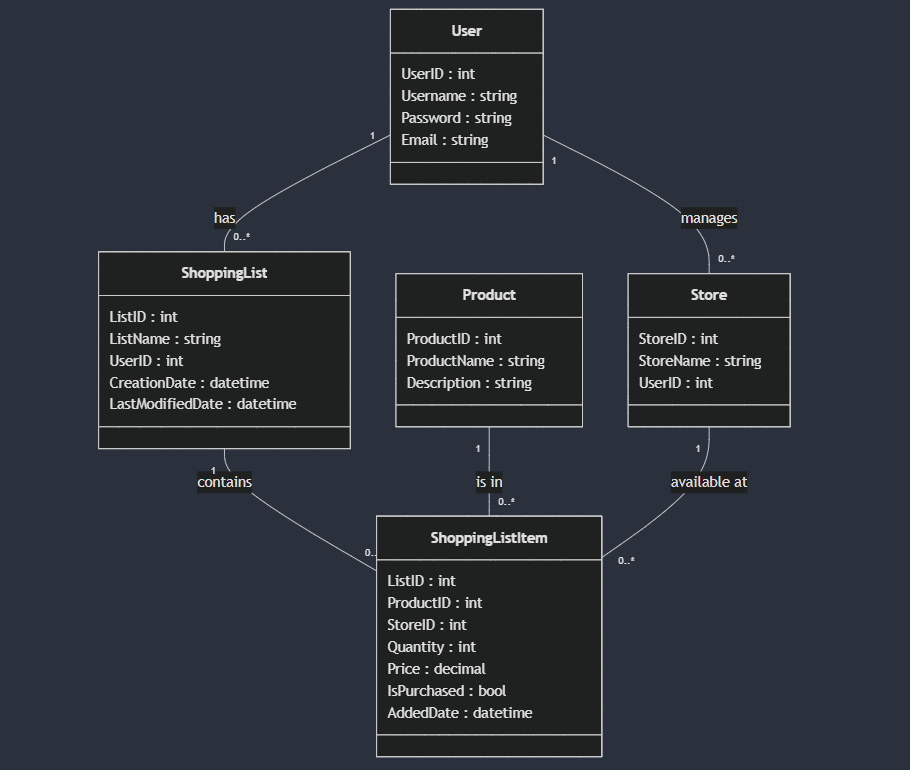
User "1" -- "0..\*" ShoppingList : has

User "1" -- "0..\*" Store : manages

ShoppingList "1" -- "0..\*" ShoppingListItem : contains

Product "1" -- "0..\*" ShoppingListItem : is in

Store "1" -- "0..\*" ShoppingListItem : available at

[](https://mermaid.live/edit#pako:eNqNVD1vwyAQ_SuIsUqidmWr6g6R2ipV1KXycgFioxqwALeKovz3HsRR7NpOzeLj3dfzO-BIuRWSMsor8D5TUDjQuSG4EkI-vHTkeEbiivt1RhhRJvRRA1oi7oNTpri6Nljlxzox4nrWoKo_-KnbfFvaukbXi_KhSyLuhyQi-jZOYpz1k5MQlDUZhJgk8BOUlp2K4MOrFWqvpBgN6rMN1skuzQQM2yZ4NtFei42zouE9LVpo2KZ1TDTKpOdO1fH3505gHaT-fwqTfCbUeG_ABBUOwzqKJ8ElVxqqq2PtN43jJXgZz9TO2o7vUYjpSZ2NdKBz-pBTslyicb9a3aHdO2qMYPnb4WnWjGgwUMg2tlfjdoukJSPcmgDKtPmX6c5KVR71avsmMrOy4BuvHOwqSQC1pguqpcNbKPAFSJPNaSglakYZmgLcV05zc8I4aILdHgynLLhGLqizTVFStofK466po9rt83EJqcF8WqvboNMvXoxUUA)

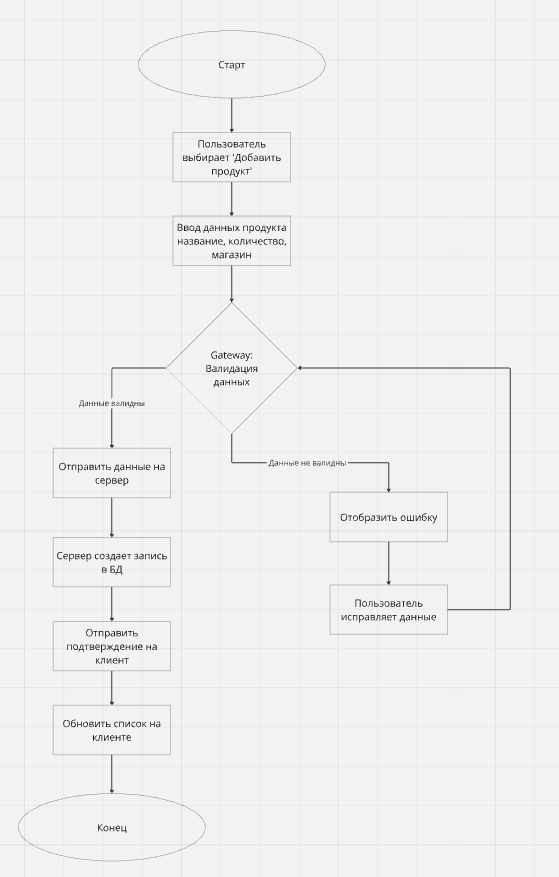
Такого рода диаграммы я делаю при помощи онлайн редактора по этой ссылке: [https://mermaid.live](https://mermaid.live/). Если хотите увидеть такие диаграммы не только в файле, просто вставьте код, который располагается выше или перейдите по ссылке прикрепленную на картинку.

**4. API (RESTful) Взаимодействие с сервером:**

* /users/register (POST): Регистрация пользователя.
* /users/login (POST): Вход пользователя.
* /users/logout (POST): Выход пользователя.
* /lists (GET): Получение списка списков покупок для пользователя.
* /lists (POST): Создание нового списка покупок.
* /lists/{listID} (GET): Получение данных конкретного списка покупок.
* /lists/{listID} (PUT): Обновление названия списка покупок.
* /lists/{listID} (DELETE): Удаление списка покупок.
* /lists/{listID}/items (POST): Добавление продукта в список покупок.
* /lists/{listID}/items/{itemID} (PUT): Обновление элемента списка покупок (количество, цена, магазин).
* /lists/{listID}/items/{itemID} (DELETE): Удаление продукта из списка покупок.
* /lists/{listID}/items/{itemID}/purchased (PUT): Отметка продукта как купленного.
* /stores (GET): Получение списка магазинов для пользователя.
* /stores (POST): Создание нового магазина.
* /stores/{storeID} (PUT): Обновление названия магазина.
* /stores/{storeID} (DELETE): Удаление магазина.
* /products (GET): Поиск продуктов (возвращает список подходящих продуктов).

**5. Нотация моделирования (BPMN):**

Для моделирования бизнес-процессов я использовал BPMN (Business Process Model and Notation). Процесс добавления продукта в список покупок:

[](https://miro.com/welcomeonboard/aGFkWmZvTnBlUnRtaVFhcUllaUEvYXo5YWNsZi8zMml3ZHBzV0lLSGFNS25objNnQkVkZWxXTkVBZU1XeGR1SkVSNEpOK0xXMmJ6b2VBLy9IVFJZbGZlN0dmMXpNbkwyd2JncG9XZENWSWFkakNKYUcrNWNjWnovTDgxa3ROZlJBS2NFMDFkcUNFSnM0d3FEN050ekl3PT0hdjE=?share_link_id=935003594438)

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

1. Создание и ведение списков покупок (сохранение, редактирование, удаление):
   * Обоснование: Это *самая* важная функция. Без возможности создания и ведения списков покупок, приложение теряет смысл.
   * Включает:
     + Создание новых списков покупок с возможностью присвоения имени.
     + Добавление элементов (продуктов) в список покупок.
     + Редактирование количества и других атрибутов продукта в списке.
     + Удаление элементов из списка покупок.
     + Удаление целых списков покупок.
     + Интуитивный и понятный интерфейс для этих действий.
2. Отметка товаров как купленных (ведение прогресса покупок):
   * Обоснование: Эта функция позволяет пользователям отслеживать, что уже куплено, а что еще нужно приобрести.
   * Включает:
     + Четкое визуальное обозначение купленных и некупленных товаров (например, зачеркивание, изменение цвета).
     + Простой и быстрый способ отметить товар как купленный (например, нажатие на чекбокс).
     + Возможность снять отметку о покупке, если товар был ошибочно отмечен.
3. Распределение продуктов по магазинам (организация покупок):
   * Обоснование: Возможность организации списка покупок по магазинам значительно упрощает процесс.
   * Включает:
     + Создание и ведение списка магазинов (пользовательские магазины).
     + Выбор магазина для каждого элемента списка покупок.
     + Автоматическую сортировку списка покупок по выбранным магазинам.
     + Возможность изменения магазина для элемента списка покупок.
4. Синхронизация списков между устройствами (доступность данных):
   * Обоснование: Синхронизация гарантирует, что списки всегда актуальны на всех устройствах.
   * Включает:
     + Автоматическую синхронизацию данных при наличии подключения к Интернету.
     + Возможность ручной синхронизации.
     + Обработку конфликтов при редактировании списков на разных устройствах.
5. Поиск продуктов (ускорение добавления товаров):
   * Обоснование: Быстрый поиск продуктов позволяет пользователям быстро добавлять товары в список, не тратя время на ручной ввод.
   * Включает:
     + Функцию поиска по названию продукта.
     + Возможность добавления результатов поиска непосредственно в список покупок.
     + (Дополнительно) Историю поиска.
6. Указание цены и количества товаров (контроль бюджета):
   * Обоснование: Указание цены и количества позволяет пользователям отслеживать общую стоимость покупок и контролировать свой бюджет.
   * Включает:
     + Возможность указания цены за единицу товара.
     + Возможность указания количества товара.
     + Автоматический расчет общей стоимости товара и общей стоимости списка покупок.
7. Аутентификация и авторизация пользователя (безопасность данных):
   * Обоснование: Обеспечивает доступ к спискам только авторизованным пользователям, защищая их данные.
   * Включает:
     + Регистрацию пользователя (создание учетной записи).
     + Вход пользователя в приложение.
     + Выход пользователя из приложения.
     + Восстановление пароля.
8. Возможность делиться списком покупок с другими пользователями (совместные покупки):
   * Обоснование: Эта функция позволяет пользователям делиться списками покупок с членами семьи, друзьями или коллегами, что упрощает процесс совместных покупок.
   * Включает:
     + Генерацию ссылки или кода для доступа к списку.
     + Управление правами доступа (например, только просмотр, редактирование).
     + Уведомления об изменениях в списке (опционально).
9. Настройка уведомлений (напоминания о покупках):
   * Обоснование: Позволяет пользователям устанавливать напоминания о покупках.
   * Включает:
     + Установку уведомлений на определенное время или дату.
     + Выбор типа уведомления (звук, вибрация).
     + Персонализацию текста уведомления.

**Почему я выбрал именно такой порядок?**

Приоритет функций основан на принципе “минимума жизнеспособного продукта” (MVP). Наиболее важные функции - это те, которые обеспечивают основную ценность приложения: возможность создавать и управлять списками покупок. Остальные функции улучшают опыт пользователя, добавляют удобство и расширяют возможности приложения, но они не являются критичными для его базовой функциональности.

**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.**

1. ER-диаграмма сущностей (Entity-Relationship Diagram):

erDiagram

User {

int UserID PK

string Username

string Password

string Email

}

ShoppingList {

int ListID PK

string ListName

int UserID FK

datetime CreationDate

datetime LastModifiedDate

}

Store {

int StoreID PK

string StoreName

int UserID FK

}

Product {

int ProductID PK

string ProductName

string Description

}

ShoppingListItem {

int ListID PK

int ProductID PK

int StoreID FK

decimal Quantity

decimal Price

boolean IsPurchased

datetime AddedDate

}

User ||--o{ ShoppingList : has

User ||--o{ Store : manages

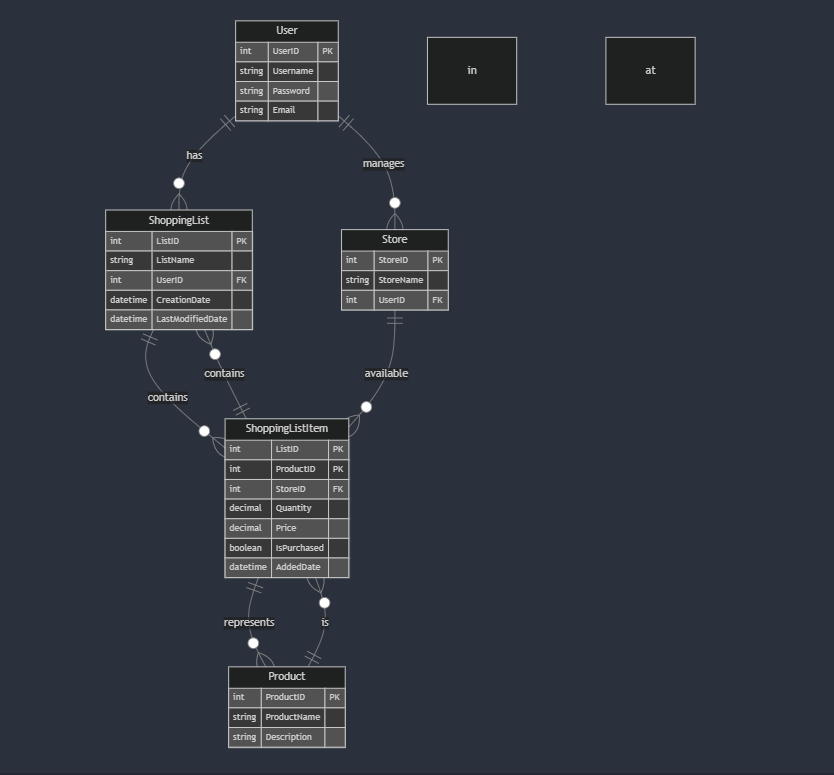
ShoppingList ||--o{ ShoppingListItem : contains

Product ||--o{ ShoppingListItem : is in

Store ||--o{ ShoppingListItem : available at

ShoppingListItem ||--o{ ShoppingList : contains

ShoppingListItem ||--o{ Product : represents

[](https://mermaid.live/edit#pako:eNqNVMFugzAM_ZUo5_YHuE1jk6p1FVO1y8TFJS5EgwQ5ZlPV9t8XKF2hBG25kLxn4pdnJ0eZWYUykkixhpygSo3w490hieNl3g5tuMNWsUhebrBj0ibvGAMVTogEnPu2pCbEUwW6vKDny2db2Lr21Fo7vs_cYuHMLbMZZR4ofR7EK2BkXaF4JATW1sQeCNBrcPxqld5rVLeQq0a2hPfiOjCsrqP-lNdvnpBVTTY5ew-HE_TkJuR9jC4jXbdnnTV6xVj9w-x5HUMDRnZjpisoxVsDhjUfpkxCOhuI3llbIhixcklDWQEOVaA6D0rdlWXQrqfTcmmP40aKhN8qENMVMhIVGMjRBTowsFlnViQyaxi0ceOqzcdr510a9s98KHz5WwG7EgXwTLHChxxrmvvpqjUShDWhQ8NOLmSF5G-j8o9A1wmp5AJ9P8nITxXQZypTc_Zx0LDdHkwmI6YGF5Jskxcy2kPp_Kqp2yr1j8gvWoP5sPa6Pv8AmaxnmQ)

2. API методы (RESTful):

* Общие принципы:
  + Все запросы требуют аутентификации (токен передается в заголовке Authorization: Bearer <token>).
  + Данные передаются в формате JSON.
  + Коды ответов: 200 (OK), 201 (Created), 204 (No Content), 400 (Bad Request), 401 (Unauthorized), 404 (Not Found), 500 (Internal Server Error).
* Списки покупок (/lists):
  + GET /lists: Получение списка списков покупок пользователя.
    - Ответ: [ {ListID, ListName, CreationDate, LastModifiedDate}, ... ]
  + POST /lists: Создание нового списка покупок.
    - Запрос: {ListName}
    - Ответ: {ListID}
  + GET /lists/{listID}: Получение информации о конкретном списке покупок.
    - Ответ: {ListID, ListName, CreationDate, LastModifiedDate}
  + PUT /lists/{listID}: Обновление названия списка покупок.
    - Запрос: {ListName}
    - Ответ: 204 No Content
  + DELETE /lists/{listID}: Удаление списка покупок.
    - Ответ: 204 No Content
* Элементы списка покупок (/lists/{listID}/items):
  + GET /lists/{listID}/items: Получение списка элементов списка покупок.
    - Ответ: [ {ProductID, ProductName, Quantity, Price, IsPurchased, StoreID, StoreName}, ... ]
  + POST /lists/{listID}/items: Добавление продукта в список покупок.
    - Запрос: {ProductID, Quantity, Price, StoreID}
    - Ответ: 201 Created
  + PUT /lists/{listID}/items/{productID}: Обновление элемента списка покупок.
    - Запрос: {Quantity, Price, StoreID}
    - Ответ: 204 No Content
  + DELETE /lists/{listID}/items/{productID}: Удаление продукта из списка покупок.
    - Ответ: 204 No Content
  + PATCH /lists/{listID}/items/{productID}: Обновление статуса “куплено” (IsPurchased).
    - Запрос: {IsPurchased}
    - Ответ: 204 No Content
* Магазины (/stores):
  + GET /stores: Получение списка магазинов пользователя.
    - Ответ: [ {StoreID, StoreName}, ... ]
  + POST /stores: Создание нового магазина.
    - Запрос: {StoreName}
    - Ответ: {StoreID}
  + PUT /stores/{storeID}: Обновление названия магазина.
    - Запрос: {StoreName}
    - Ответ: 204 No Content
  + DELETE /stores/{storeID}: Удаление магазина.
    - Ответ: 204 No Content
* Продукты (/products):
  + GET /products?search={query}: Поиск продуктов по названию.
    - Ответ: [ {ProductID, ProductName, Description}, ... ]

**3. UML-диаграммы:**

3.1 Добавление списка покупок:

sequenceDiagram

participant Client

participant Server

Client->>Server: POST /lists {ListName: "Новый список"}

activate Server

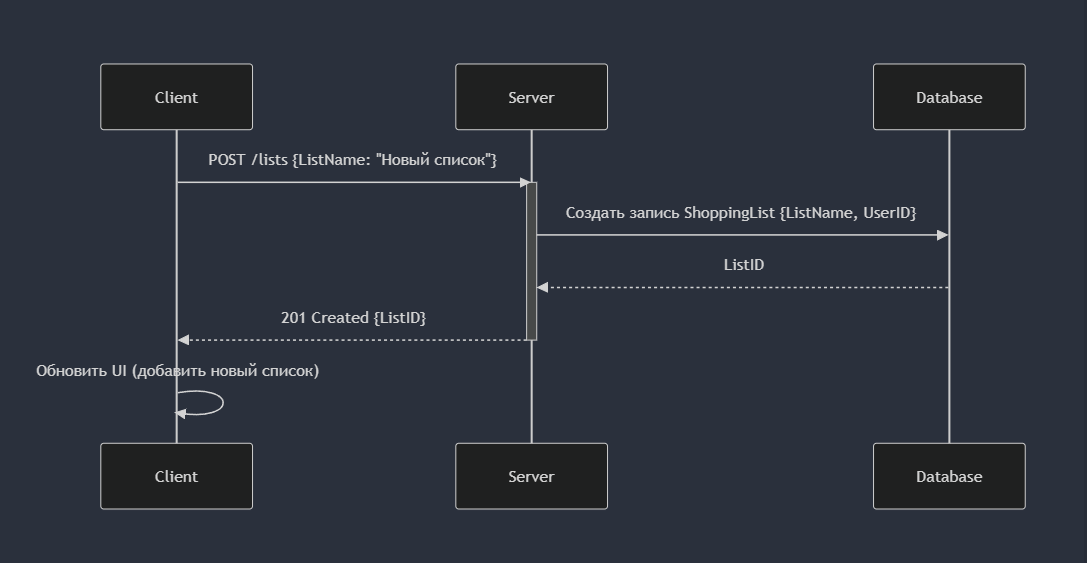
Server->>Database: Создать запись ShoppingList {ListName, UserID}

Database-->>Server: ListID

Server-->>Client: 201 Created {ListID}

deactivate Server

Client->>Client: Обновить UI (добавить новый список)

[](https://mermaid.live/edit#pako:eNptUUtOwzAQvYo1K5Ba8Vlm0U2zqYQAKXSDvBmSobFonOA4lVDVBWzhCiBxg4KQ-EgtV3BuxERuA1Xxwp55Hr_3PDOFOE8IAijpuiIdU6hwZDCTWvAq0FgVqwK1Ff2xIm238YjMhIzHfU231_NgIE5PojOxN1alLcX0iI9jzCgQEtyjW7rX-t59ivrWfbsP3pfuS8LMM2Fs1QQtbdD7mOlDtHiBJTO5Z3727t7cvL6rHwSHc8_GSZTmRaH0qNH9Ve-IYUlmEK6E1lTdP66b0kG4ocm3_nOBONw_EH1DbC7xrC1XQv_abruyZnBP7sUtmgaw08b2cCB2-A9Lhuct6Cu2WrQrNXQgI5OhSnhw00ZEgk0pIwlNbxM0VxKknnEdVjaPbnQMgTUVdcDk1SiF4BLHJWdVkbDZ1chblMd6nufrfPYD7UvfBg)

3.2 Удаление списка покупок:

sequenceDiagram

participant Client

participant Server

Client->>Server: DELETE /lists/{listID}

activate Server

Server->>Database: Удалить ShoppingList {ListID}

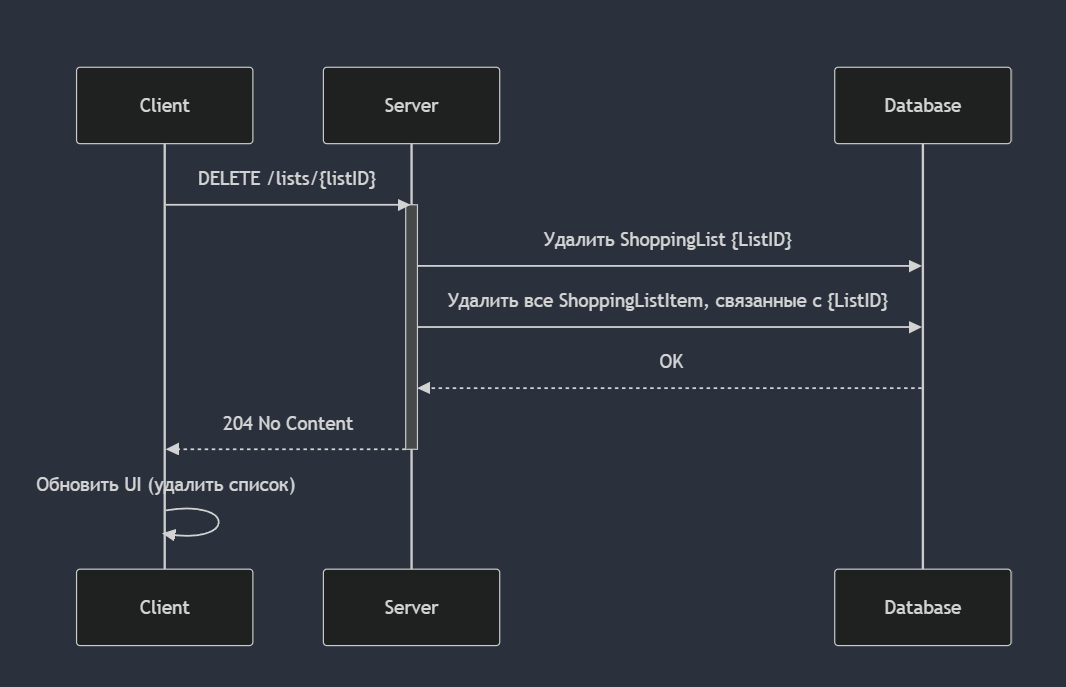
Server->>Database: Удалить все ShoppingListItem, связанные с {ListID}

Database-->>Server: OK

Server-->>Client: 204 No Content

deactivate Server

Client->>Client: Обновить UI (удалить список)

[](https://mermaid.live/edit#pako:eNqFUrFOwzAQ_RXLE0itihCThy5Nh4gKhsKCshzJ0Vo0dnCcSijqUBhB4g9Y-AGoQCBQ4RfsP-KqtFUDSHjx3fO753dnlzzWCXLBc7woUMUYSBgYSCPFaGVgrIxlBsqyzkiisr_xPpoxmgqvOM12uwIFC7q97lGXtUYyt3mrXGxhMKnIEFs5Bos1hSomhQAsnEKOgrkH9-we3Yd781f-lvWHOsukGvRIi5W9TcX_i93MT91LTSO0mDYYwTN_516JO3dzf0MkP_0hv1JtbjR4uF-7mk6qGQi2u7PHDjTraGXXc0vwz6bXY1vVunv3RD4-3Wzp-zhkW_661go5_qJwSqz37UjxBk_RpCATesxyoRtxO8QUIy4oTMCcRzxSE-JBYXX_UsVcWFNggxtdDIZcnMEop6zIEvK3_AZrlJ76ROtVPvkGtrbi3Q)

3.3 Добавление элемента в список покупок:

sequenceDiagram

participant Client

participant Server

Client->>Server: POST /lists/{listID}/items {ProductID, Quantity, Price, StoreID}

activate Server

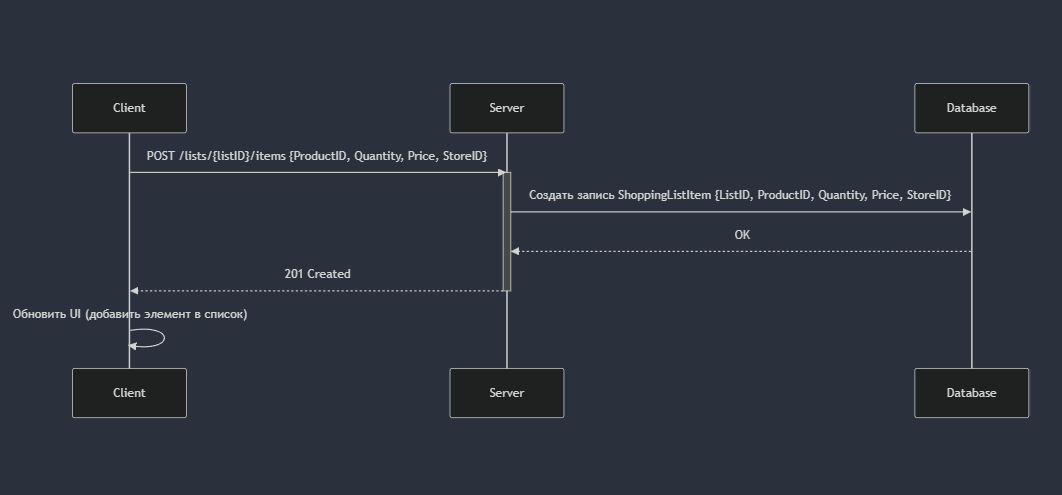
Server->>Database: Создать запись ShoppingListItem {ListID, ProductID, Quantity, Price, StoreID}

Database-->>Server: OK

Server-->>Client: 201 Created

deactivate Server

Client->>Client: Обновить UI (добавить элемент в список)

[](https://mermaid.live/edit#pako:eNqNUsFKw0AQ_ZVhTwopVY859NJcikIr0YvksiZju9jsxs2mUEIP7Vl_QfAPalEUa-svbP7IiWuLRQ8GQmZe3s57b3dLFqsEmc9yvClQxhgI3tc8jSTQk3FtRCwyLg20hwKl-Y2HqEeoHe44jVbLgT70uuEZNIciN3mzrD-dYNIUBtMcyp5WSRET4sFpQZOEGXvQ0yJGD0KjNBLXjeWxESNucEfL1aQVcMMveY4-2Ae7ti_2yc6rWXULVM7th32tptSEA5VlQvZPahNkAMqvKqgl_-tjo9T4kbB7vGOH_rhN8OHo4BDaGsl34igJ_hlku2mbhfbePtoVRVmQ9zrIeQf2KNWa4PkWrO7s0j7bd3pX1QzsAqqpS0vEt_1IMo-lqFMuEjrestaKmBlgihHzqUy4vo5YJCfE44VR4VjGzDe6QI9pVfQHzL_iw5y6IkvI8_fF2KJ0-BdKbfrJJ2Bz5pk)

3.4 Редактирование элемента списка покупок:

sequenceDiagram

participant Client

participant Server

Client->>Server: PUT /lists/{listID}/items/{productID} {Quantity, Price, StoreID}

activate Server

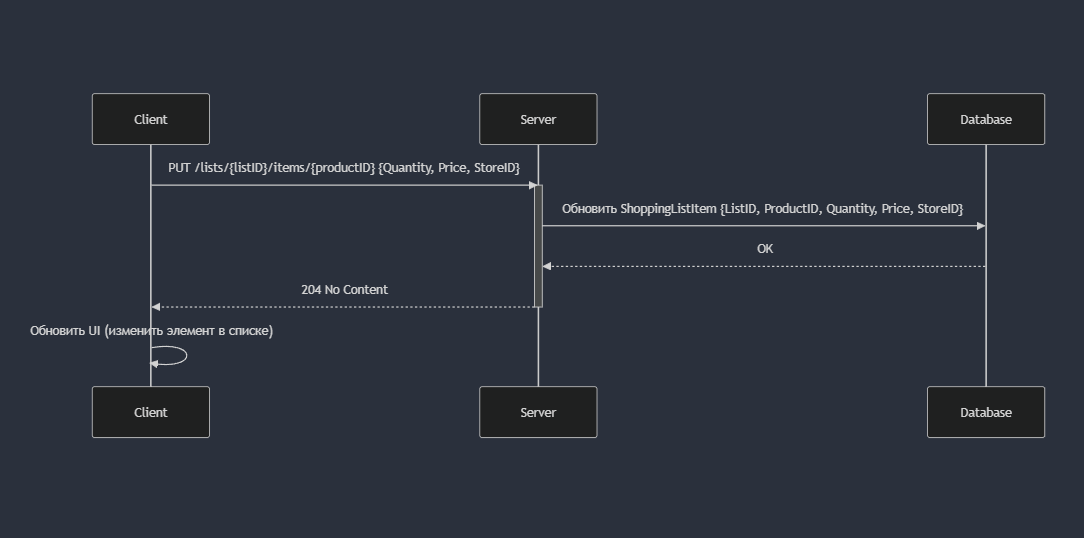
Server->>Database: Обновить ShoppingListItem {ListID, ProductID, Quantity, Price, StoreID}

Database-->>Server: OK

Server-->>Client: 204 No Content

deactivate Server

Client->>Client: Обновить UI (изменить элемент в списке)

[](https://mermaid.live/edit#pako:eNqFUs1OwkAQfpXJnjQpwRhPPXCBi9EopnIxvaztCBvpbt1uSUjDAc76Cr6DJETi7ytM38gpBSLBxL3sfN_Mzjc_W4jIxCh8keFDjjrCjpJ9K5NQA59UWqcilUrtoD1UqN0-H6Adoa35OqbRatWkD93eNTSHKnNZs6iu086kqRwmDFNr4jyqGCiucs6k3NiDrlURehA4Y5FddVoZOTWSDne0apu1OtLJW5mhD_RML_RJXzSnZTkrHyEYmDRVun9eSbMsFCurU-ms1T34R3yTvvGrrcuznRrYU3fuw_HRCVwYaBvtttOK8c8GtsPavN0rv3cKB7SkV_qgBTtqsnyid4YrqpwBzaGc0jc7p_RGi8NQC08kaBOpYl5rUWmFwg0wwVD4bMbS3oci1BOOk7kzwVhHwnc2R09Yk_cHwr-Tw4xRnsZc8_pDbFle-o0xGzz5ARPB4PY)

3.5 Покупка/«откупка» продукта:

sequenceDiagram

participant Client

participant Server

Client->>Server: PATCH /lists/{listID}/items/{productID} {IsPurchased: true/false}

activate Server

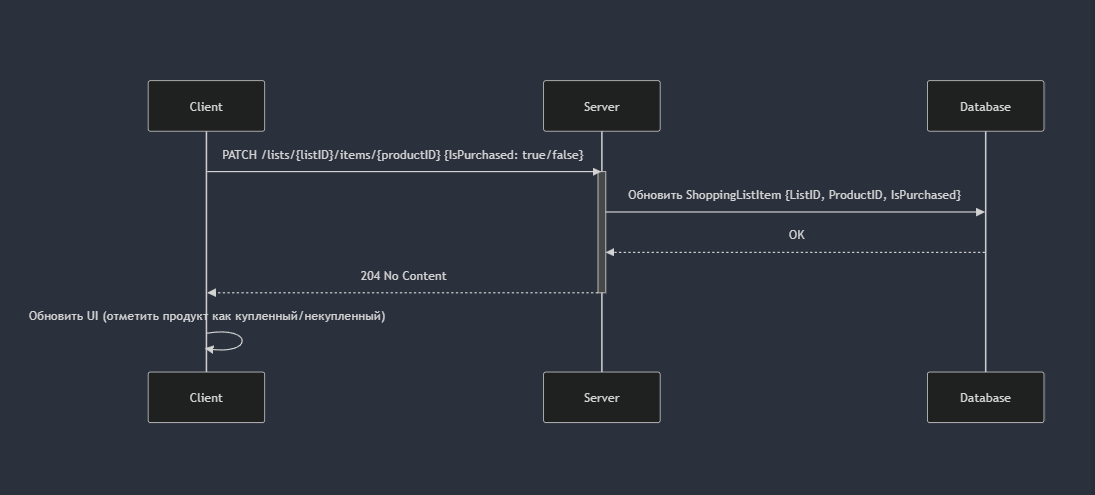
Server->>Database: Обновить ShoppingListItem {ListID, ProductID, IsPurchased}

Database-->>Server: OK

Server-->>Client: 204 No Content

deactivate Server

Client->>Client: Обновить UI (отметить продукт как купленный/некупленный)

[](https://mermaid.live/edit#pako:eNptUs1Kw0AQfpVlTwopEfG0h4I0B4uihepFclmTtVlsduNmU5AQkHrUZ_AZVCxote0rTN7ISdOUlrqH3ZlvZ-abv5wGOhSU0VTcZ0IFwpN8YHjsK4In4cbKQCZcWdIZSqHsLt4XZiRMjdc2rXa7BhnpHV92Tog7lKlN3bx6ul7hSitiVBOjwyyoEJJ3015mgoinImTEmky4t3yYiqKOywMrR9yKLbJaRjKPW36DnozAK7zDDObwAV_luHwh_UgniVSDs4oZWUm-lDyH9Bpyh2xwr_iaiK2NUi5Ot2jxp66WkcODI3KuSUcru-5QKP7Ned2gxncn46su2YN5OYZfmOBdg7AoH9Hms3yCaTkmMIU3mOKD-gJ-YIIRZuUzfLsoTHbhfV9Rh8bCxFyGOOu8SsanNhKx8ClDMeTmzqe-KtCOZ1b3H1RAWTUHhxqdDSLKlvNwaJaEWNRqS9YobsK11o1e_AHfP_VM)

4. Детализация процесса синхронизации:

* Инициализация: При запуске приложения клиент проверяет наличие локальной базы данных. Если её нет, или если пользователь вошел в систему впервые, клиент запрашивает список списков покупок с сервера (GET /lists).
* Локальные изменения: Все изменения, сделанные пользователем (добавление, удаление, редактирование), сначала сохраняются в локальной базе данных.
* Синхронизация:
  + Периодическая синхронизация: Приложение автоматически синхронизирует данные с сервером через определенные промежутки времени (например, каждые 5 минут).
  + Ручная синхронизация: Пользователь может принудительно запустить синхронизацию.
  + Синхронизация при изменении: Приложение может пытаться синхронизировать данные сразу после внесения изменений (при наличии подключения к Интернету).
* Стратегия синхронизации:
  + Клиент отправляет на сервер информацию обо всех локальных изменениях, произошедших с момента последней успешной синхронизации (используя временные метки LastModifiedDate).
  + Сервер обрабатывает эти изменения и отправляет клиенту обновленные данные.
  + Клиент обновляет локальную базу данных с учетом полученных изменений.
* Обработка конфликтов:
  + Реализация стратегий разрешения конфликтов:
    - “Побеждает последнее изменение”: Последнее изменение, полученное сервером, применяется.
    - “Слияние изменений”: Применяются оба изменения, если это возможно.
    - “Предупреждение пользователя”: Пользователю предлагается выбрать, какое изменение следует применить.

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

[****](https://www.figma.com/design/Hq1ctxHdiqMCilamrX1QaC/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4-%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B8?node-id=0-1&t=BxNFfhZsjD4DpcD0-1)

**Описание прототипа:**

Логотип: В левом верхнем углу. Нажав на него возвращает на главную.

Кнопка в виде крестика: В правом верхнем углу. Позволяет вернуться к предыдущей странице.

Общее кол-во товаров: Показывает количество товаров, которые пользователи добавили.

Кнопка поиска: позволяет находить определенный товар, если список слишком большой.

Кнопка сброса: позволяет полностью удалить весь список.

Картинка товара: Небольшая картинка, представляющая товар. Это улучшает визуальное восприятие.

Название товара: Название товара из списка покупок(иногда указывается вес).

Количество: Количество товара, указанное в списке покупок, которую можно менять в большую или меньшую сторону при помощи “+” и ”-”.

Итоговая стоимость: Указывает сколько всего выйдет за определенный товар.

**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)**

Цель: Реализовать функцию, позволяющую пользователю отмечать товары в списке покупок как купленные, фиксировать факт покупки, и отражать изменения на всех устройствах пользователя.

1. Текстовое описание:

Пользователь, просматривая список покупок, должен иметь возможность легко и быстро отметить отдельный товар как купленный. Это действие должно:

* Визуально отобразиться в пользовательском интерфейсе (например, зачеркивание товара, изменение цвета).
* Сохранить информацию о факте покупки в базе данных на сервере.
* Синхронизировать изменения на всех устройствах, где пользователь авторизован.

Аналогично, пользователь должен иметь возможность снять отметку о покупке товара (например, если покупка была отменена).

2. Пользовательский интерфейс (UI):

* Каждый элемент списка покупок должен иметь интерактивный элемент, позволяющий отметить товар как купленный (чекбокс, переключатель).
* Визуальное представление:
  + Не куплен: Стандартное отображение товара в списке.
  + Куплен: Товар зачеркнут, цвет может быть приглушенным.
* Элемент управления должен быть интуитивно понятным и легко доступным.

3. API-метод:

* Метод: PATCH /lists/{listID}/items/{productID}
* Описание: Обновляет статус IsPurchased для указанного товара в списке покупок.
* Параметры URL:
  + {listID}: Идентификатор списка покупок.
  + {productID}: Идентификатор продукта в списке покупок (ProductID из сущности Product).
* Запрос (Request Body):
  + { "IsPurchased": true/false }
    - IsPurchased: Булево значение, указывающее статус товара (true - куплен, false - не куплен).
* Пример запроса:

PATCH /lists/123/items/456

Content-Type: application/json

Authorization: Bearer eyJhbGciOiJIUzI1Ni...

{

"IsPurchased": **true**

}

json

Ответ (Response):

* + Успех: 204 No Content (успешное обновление)
  + Ошибка:
    - 400 Bad Request: Неверный формат запроса, неверные данные.
    - 401 Unauthorized: Пользователь не аутентифицирован.
    - 404 Not Found: Список покупок или товар не найден.
    - 500 Internal Server Error: Ошибка на сервере.

4. UML-диаграмма последовательности (Sequence Diagram):

sequenceDiagram

participant Пользователь

participant Клиентское приложение

participant Сервер

participant База данных

Пользователь->>Клиентское приложение: Отмечает товар как "Куплен"

Клиентское приложение->>Сервер: PATCH /lists/{listID}/items/{productID} {IsPurchased: true}

activate Сервер

Сервер->>База данных: Обновить ShoppingListItem {ListID, ProductID, IsPurchased=true}

База данных-->>Сервер: Успешно

Сервер-->>Клиентское приложение: 204 No Content

deactivate Сервер

Клиентское приложение->>Пользователь: Обновить UI (товар зачеркнут)

Клиентское приложение->>Сервер: (Фоновая синхронизация) Запрос актуальных данных /lists/{listID}/items

activate Сервер

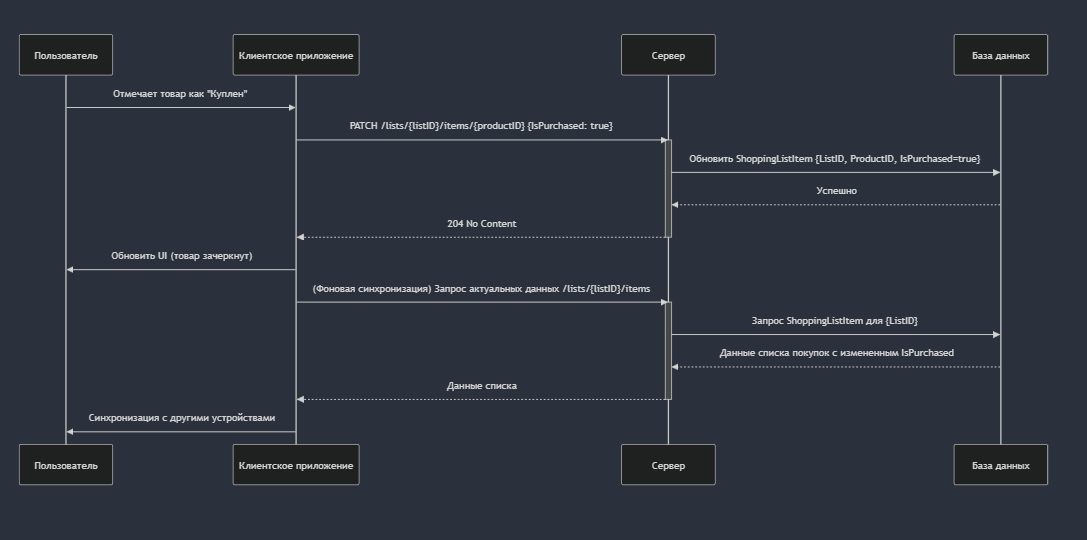
Сервер->>База данных: Запрос ShoppingListItem для {ListID}

База данных-->>Сервер: Данные списка покупок с измененным IsPurchased

Сервер-->>Клиентское приложение: Данные списка

deactivate Сервер

Клиентское приложение->>Пользователь: Синхронизация с другими устройствами

[](https://mermaid.live/edit#pako:eNq9VV2L00AU_SuXedqFLhXxKeCCbB8siBRWXyQvIRnb4DaJyUSQUugHq4ILiyAI-6As-gPqutXaz78w8488N20lsqlsWbGFztyZe-ece-Y0aQk39KSwRCKfpzJwZcV36rHTtAPCJ3Ji5bt-5ASK9Cc91xNzon9gvNAD09NDjgsyz_REj7A7Mz3T1WPkD0kvTAeLEwTfeYsTCkrP9RB5F_xbsPtODwA_IH2JyQznvzXHdsBfTt3EcG9__9qULNIfUTcFgdfAGJoeIVwe1yEUDvSYbKHPTF8vUIoyW6zgr4vBfHJ9WlS79-jgPpWP_EQl5RYP1Uq77CvZRBjFoZe6vEKtalJLY7fhJNKzSMWpbC-hHVf5LxwlC_TLrzBwkYTctP6KiPscgf0JHTbCKPKD-gMmAyLUymaVEtXWfEqUo3M3x6YQY-9K1_ozVFpg_oaRC9hud2-3b92hhyEdhIGSgVoe58m_KbPNfW2wVoFyj6u0k7cMhICVGHsMpL7p7d7ULjv6C_ZnK4hTQvUIpxwjY56VMOQr0DndJf0BV8CHzk2X2LwA62PkXrJ7-eOSij34rzyWp3LFX8idoJeVzbZ00vv1LiTMTDXKFEXdApKMs_8qRmINWJ5ppu1wVTPNG_nGPtxI5v9Z8nyTHzIBLk0HgnzD-lSPCI7sAoxTf_KMj-INOxAl0ZRx0_E9vB5aTNAWqiGb0hYWpp4TP-NHXxt5TqrCw5eBKyx-DJREHKb1hrCeOkcJojTy0O7qxfJ7FU_0J2G4jtu_AJstwck)

5. Хранение информации о покупках:

* Информация о том, куплен товар или нет, хранится в поле IsPurchased таблицы ShoppingListItem в базе данных.
* Тип данных поля: boolean (true/false).
* Дополнительные поля: Можно добавить поля PurchasedDate (дата и время покупки) и PurchasedByUserID (идентификатор пользователя, совершившего покупку) для более детальной информации.

6. Бизнес-логика на сервере:

1. Аутентификация: Проверка токена аутентификации пользователя в заголовке запроса (Authorization).
2. Авторизация: Проверка, имеет ли пользователь право изменять данный список покупок (является ли он владельцем списка или имеет права на редактирование).
3. Валидация: Проверка, что listID и productID существуют и соответствуют друг другу.
4. Обновление базы данных: Обновление поля IsPurchased в таблице ShoppingListItem для указанного listID и productID.
5. Регистрация действия: (Опционально) Запись информации о покупке в журнал аудита (для отслеживания действий пользователей).
6. Отправка уведомлений: (Опционально) Отправка уведомлений другим пользователям, имеющим доступ к списку (например, если это общий список).

7. Синхронизация данных:

* Изменение статуса IsPurchased автоматически инициирует процесс синхронизации, как описано в предыдущих ответах.
* Приложение периодически (или по запросу пользователя) запрашивает с сервера обновленные данные для списка покупок, получая актуальный статус IsPurchased для каждого элемента.

8. Технические требования:

* Необходимо реализовать обработку ошибок (неверные входные данные, отсутствие подключения к базе данных, и т.д.).
* Необходимо обеспечить безопасность API (защита от несанкционированного доступа, XSS, CSRF).
* Реализация должна быть оптимизирована для высокой производительности и масштабируемости.

**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: Обеспечение актуальности списков на разных устройствах с нестабильным интернетом. Сложность в разрешении конфликтов и масштабировании.
2. Оффлайн-режим: Работа приложения без интернета. Сложность в локальном хранении данных и последующей синхронизации.
3. Безопасность: Защита данных пользователей. Сложность в хранении паролей и защите API.
4. Производительность: Быстрая и плавная работа приложения. Сложность в оптимизации кода и базы данных.

Вопросы заказчику:

1. Целевая аудитория: Кто будет пользоваться приложением?
2. Монетизация: Как планируется зарабатывать на приложении?
3. Интеграция: Нужна ли интеграция с другими сервисами?
4. База продуктов: Где брать данные о продуктах?
5. Платформы: Под какие платформы разрабатываем?
6. Дизайн: Есть ли требования к дизайну?
7. Бюджет/Сроки: Какой бюджет и сроки?
8. Обработка ошибок: Как показывать ошибки пользователям?
9. Поддержка/Масштаб: Как планируется поддерживать и масштабировать приложение?

**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 и сортировка результата в порядке убывания;

**SELECT**

A.AuthorName,

SUM(B.Price) **AS** TotalBookPrice

**FROM**

Books B

**JOIN**

Authors A **ON** B.AuthorId = A.Id

**GROUP** **BY**

A.AuthorName

**ORDER** **BY**

TotalBookPrice **DESC**;

2. Нахождение стоимости книг автора, которые превышают 1500;

**SELECT**

**A.AuthorName,**

**SUM(B.Price) AS TotalBookPrice**

**FROM**

**Books B**

**JOIN**

**Authors A ON B.AuthorId = A.Id**

**GROUP BY**

**A.AuthorName**

**HAVING**

**SUM(B.Price) > 1500;**

3. Вывести авторов с количеством книг;

**SELECT**

A.AuthorName,

COUNT(B.Id) **AS** NumberOfBooks

**FROM**

Authors A

**LEFT** **JOIN**

Books B **ON** A.Id = B.AuthorId

**GROUP** **BY**

A.AuthorName

**ORDER** **BY**

NumberOfBooks **DESC**;

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**;