ОТЧЕТ ПО ПРАКТИКЕ

По теме «Аналитика данных»

Сроки прохождения практики: с *01.07.2024* г. по *13.07.2024* г.

Выполнил практику:

Студент Самарского университета

Группы 6302-010302D

Максимова Алина Вадимовна

Самара 2024

СОДЕРЖАНИЕ

[1 Моделирование бизнес-процесса 3](#_Toc171277753)

[2 Описание процесса синхронизации 5](#_Toc171277754)

[3 Прототип экрана мобильного приложения 12](#_Toc171277755)

[4 Описание функцию редактирования заказа 15](#_Toc171277756)

[4.1 Аудитория, использующая данную функцию 15](#_Toc171277757)

[4.2 Аудитория, которой предназначен документ 15](#_Toc171277758)

[4.3 Решаемая задача 15](#_Toc171277759)

[4.4 Глоссарий 16](#_Toc171277760)

[4.5 Критерии качества на входе(DoR) 16](#_Toc171277761)

[4.6 Ограничения 17](#_Toc171277762)

[4.7 Прототип 17](#_Toc171277763)

[4.8 Концепция решения 17](#_Toc171277764)

[4.9 Функциональные решения 17](#_Toc171277765)

[4.10 Нефункциональные требования 18](#_Toc171277766)

[4.11 Диаграмма Sequence 19](#_Toc171277767)

[4.12 Бизнес-процесс в нотации BPMN 21](#_Toc171277768)

[4.13 ER-модель 21](#_Toc171277769)

[4.14 Use Cases 21](#_Toc171277770)

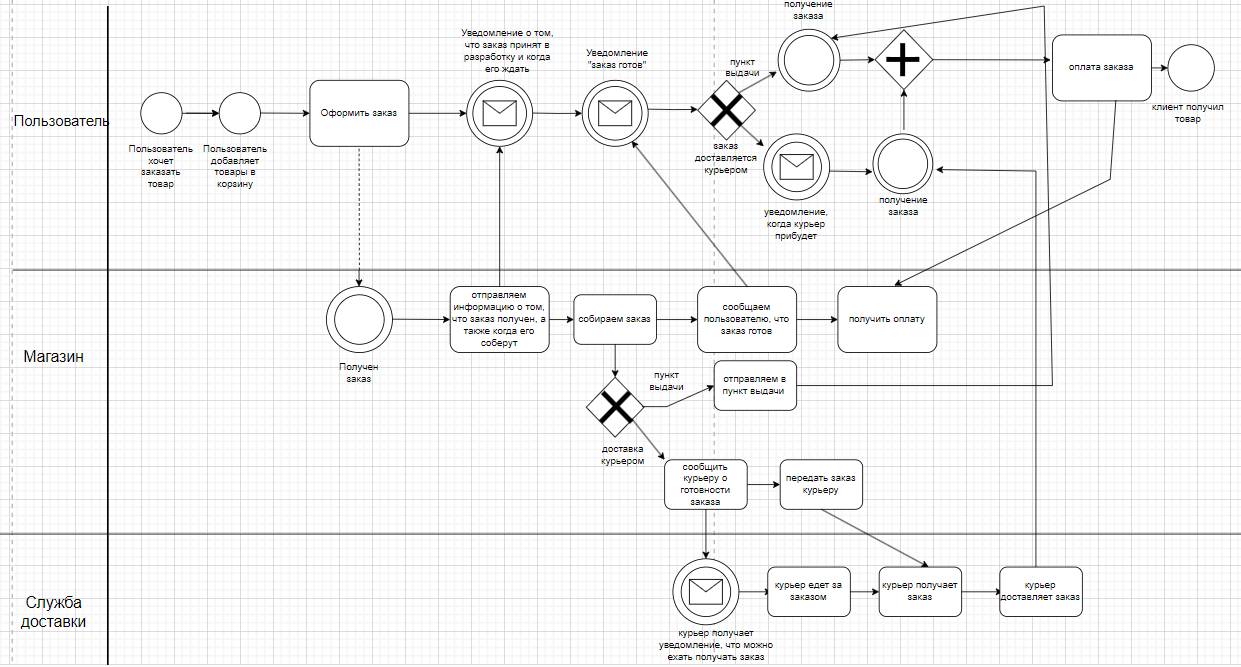
[4.15 Описание структуры данных 22](#_Toc171277771)

[5 Запросы к базе данных 24](#_Toc171277772)

1. Моделирование бизнес-процесса

Задание: Бизнес-процесс создания заказа (использовать любую удобную нотацию моделирования бизнес-процессов).

Моделирование бизнес-процесса проводится в нотации BPMN.



В данной модели бизнес-процесса представлены шаги, необходимые для создания заказа в интернет-магазине "Магазин уходовой косметики".

Для пользователя процесс выглядит следующим образом: сначала он выбирает и добавляет товары в корзину, затем оформляет заказ и получает уведомления о статусе заказа. После подтверждения готовности заказа клиент получает его в соответствии с выбранным способом доставки, оплачивает. По итогу, пользователь получает заказ, который хотел.

С точки зрения магазина процесс выглядит следующим образом: сотрудники получают уведомление о заказе, отправляют уведомление пользователю о получении заказа, начинают его сборку, затем передают заказ для доставки (либо в пункт выдачи, либо в курьерскую службу). Отправляют уведомление пользователю о готовности товара. После получения оплаты они завершают процесс.

Для курьерской службы процесс включает получение уведомления о необходимости забрать заказ, получение заказа, его доставку клиенту после получения в магазине.

1. Описание процесса синхронизации

Задание: Опишите процесс синхронизации данных между клиентом и сервером (создание, редактирование и отмена заказа, изменение персональных данных, оплата заказа и т.д.). Представить все в диаграммах UML, API методах и других представлениях, также составить ER-диаграмму сущностей.

Первоначально определим, какие сущности можно выделить в интернет магазине «Магазин уходовой косметики». Можно выделить следующие сущности: клиент магазина, товар, заказ, бренд, отзыв и оплата.

У каждой сущности определим атрибуты.

Клиент магазина имеет следующие атрибуты: ID клиента, ФИО клиента, дата рождения, адрес доставки, электронная почта, история покупок.

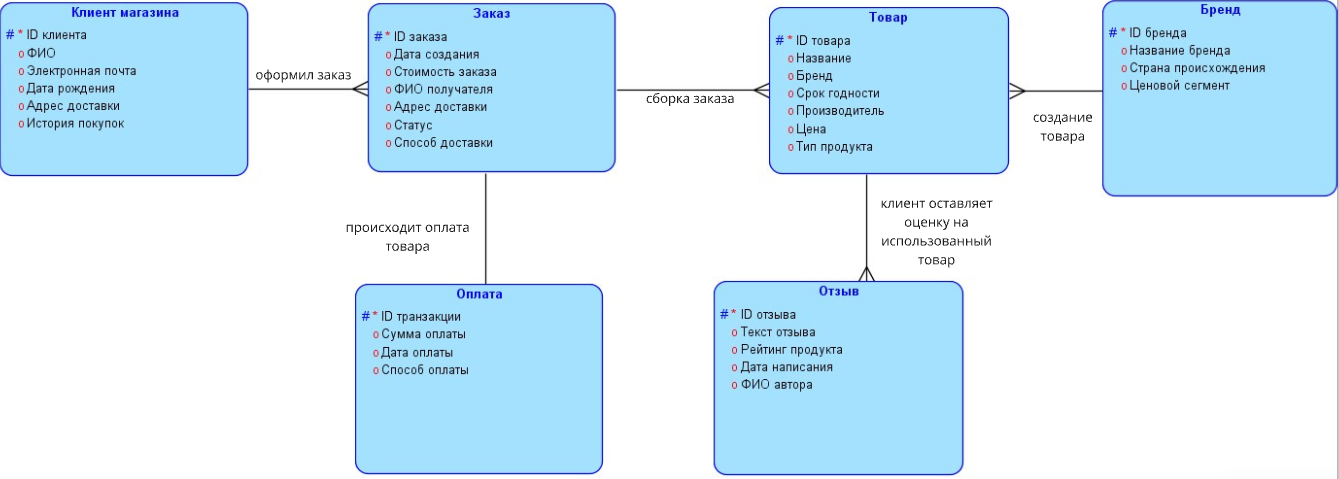
Заказ: ID заказа, дата создания, стоимость заказа, ФИО получателя, адрес доставки, статус, способ доставки.

Товар: ID товара, бренд, название, срок годности, цена, тип продукта.

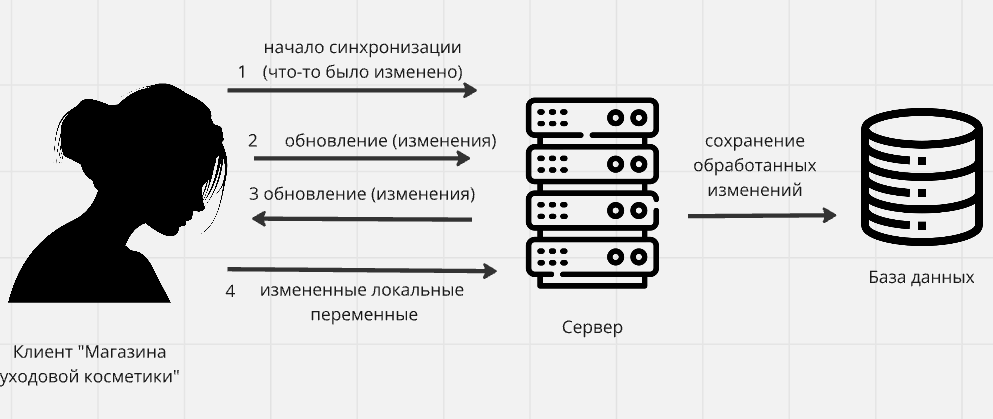
Отзыв: ID отзыва, текст отзыва, дата создания, рейтинг продукта, ФИО автора.

Оплата: ID транзакции, сумма оплаты, дата оплаты, способ оплаты.

Бренд: ID бренда, название бренда, страна происхождения, ценовой сегмент.

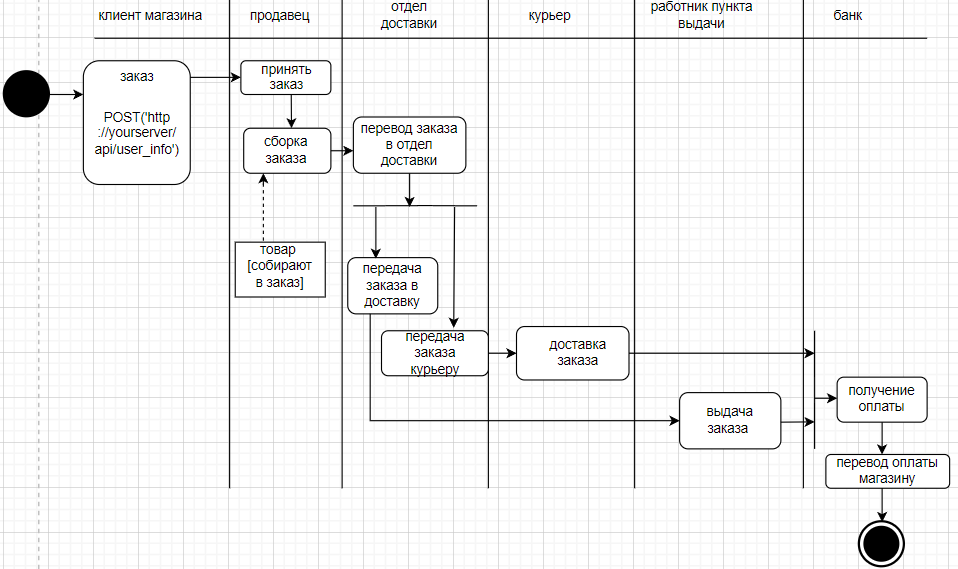


Опишем процесс синхронизации. Проблема синхронизации может состоять, например, в том, что доступ в интернет нестабилен или вовсе отсутствует у клиента. Диаграмма же необходима для общего представления о функционировании приложения при синхронизации данных. Рассмотрим составленную диаграмму.



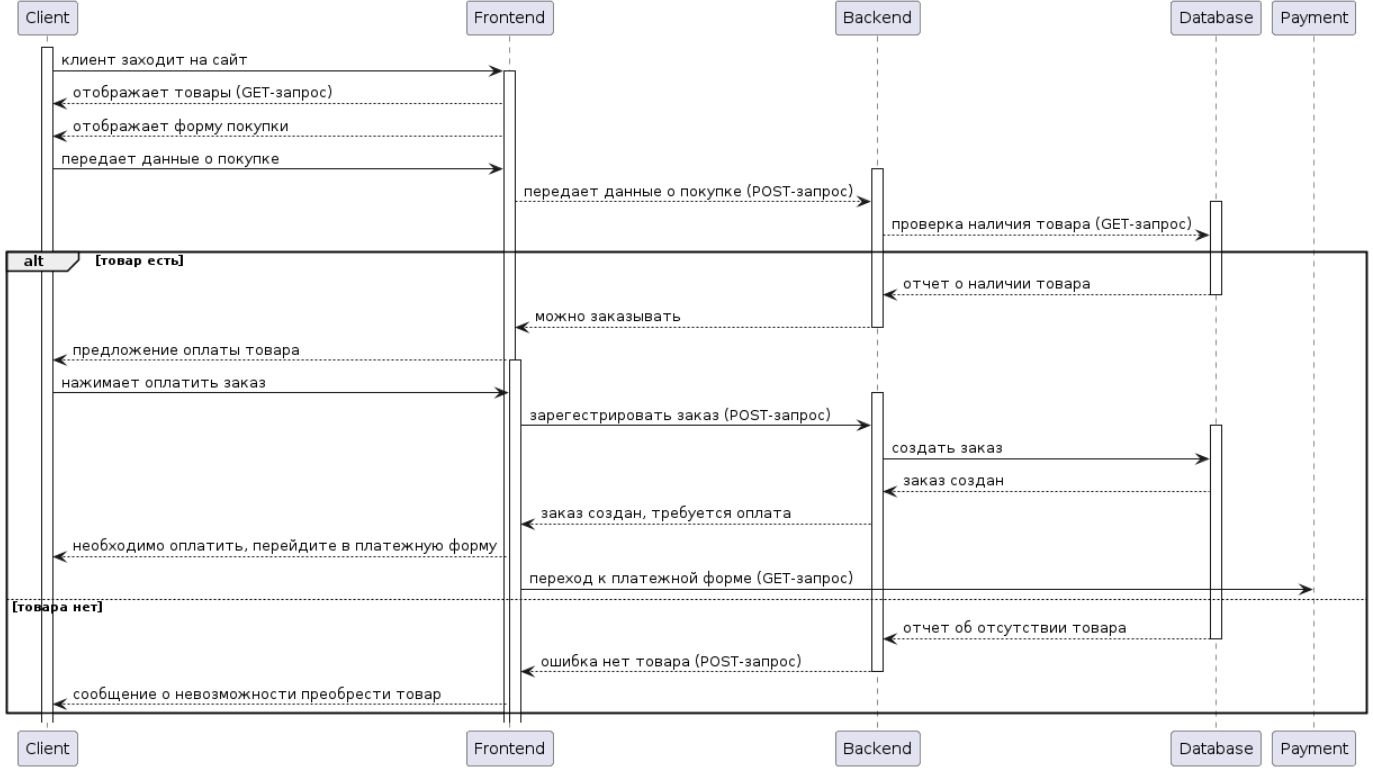
Кратко опишем то, что происходит на рисунке. Мы видим, что пользователь вносит изменения на своем приложении. Измененные данные отправляются на сервер через Интернет. Данная операция может быть выполнена через API метод, например. Далее сервер обрабатывает данные. Это может быть проверка данных, обновление базы данных и так далее. После обработки, сервер взаимодействует с базой данных для сохранения изменений. В дальнейшем пользователь видит те изменения, которые внес.

Создадим UML-диаграмму деятельности, которая будет описывать процесс оформления, сбора, доставки, а после получения и оплаты заказа.



Создадим UML-диаграммы последовательности для более детального описания действия между сервером и клиентской частью.

Далее предлагается рассмотреть диаграмму последовательности, которая нужна для описания создания, регистрации и оплаты заказа.



Код для данной диаграммы:

@startuml

participant Client as c

participant Frontend as f

participant Backend as b

participant Database as d

participant Payment as p

activate c

c -> f: клиент заходит на сайт

activate f

f --> c: отображает товары (GET-запрос)

f --> c: отображает форму покупки

c -> f: передает данные о покупке

activate b

f --> b: передает данные о покупке (POST-запрос)

activate d

b --> d: проверка наличия товара (GET-запрос)

alt товар есть

d --> b: отчет о наличии товара

deactivate d

b --> f: можно заказывать

deactivate b

f --> c: предложение оплаты товара

activate f

c -> f: нажимает оплатить заказ

activate b

f -> b: зарегестрировать заказ (POST-запрос)

activate d

b -> d: создать заказ

d --> b: заказ создан

b --> f: заказ создан, требуется оплата

f --> c: необходимо оплатить, перейдите в платежную форму

f -> p: переход к платежной форме (GET-запрос)

else товара нет

d --> b: отчет об отсутствии товара

deactivate d

b --> f: ошибка нет товара (POST-запрос)

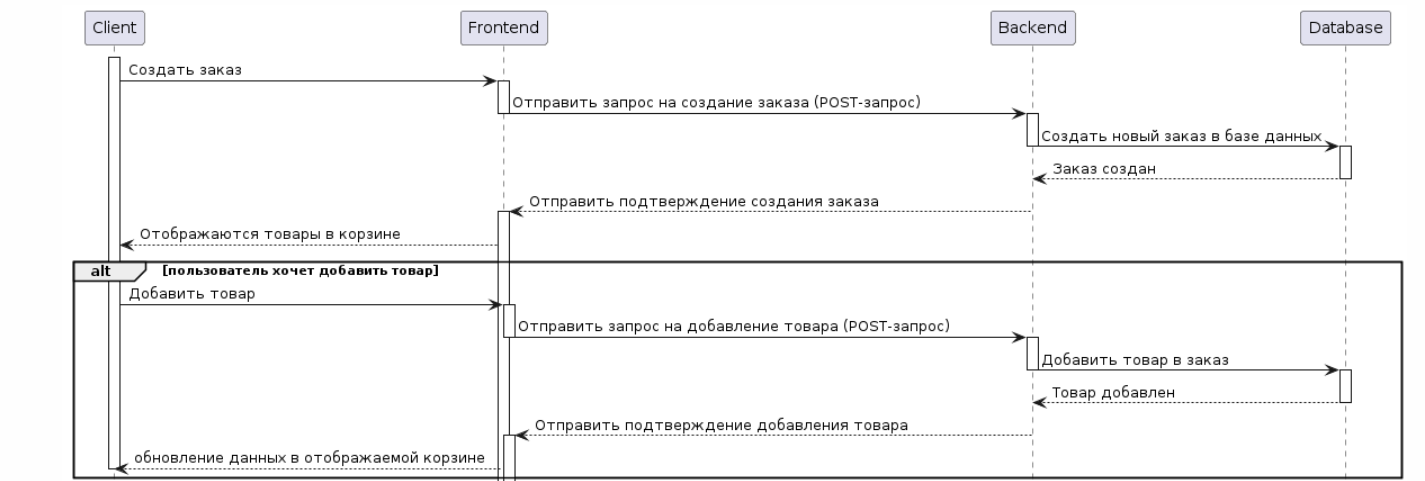
deactivate b

f --> c: сообщение о невозможности преобрести товар

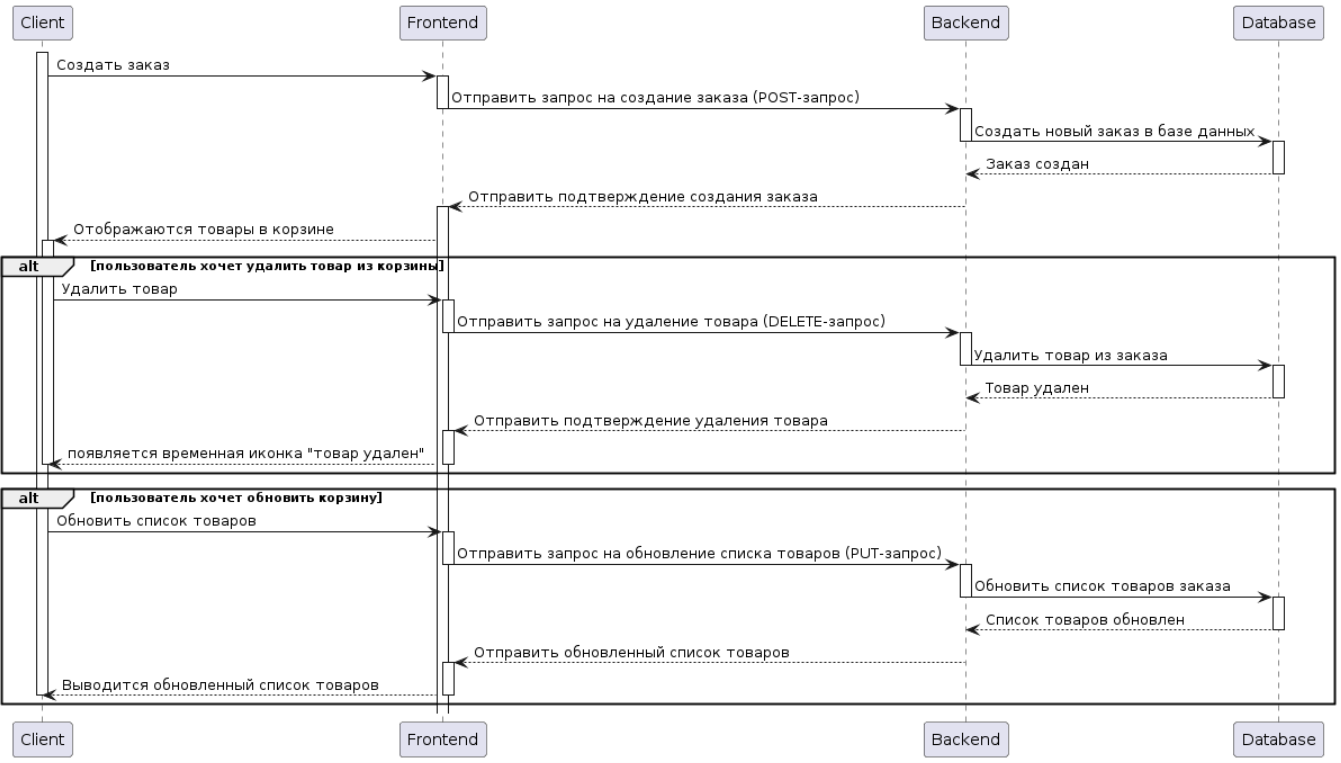
end

@enduml

Далее предлагается рассмотреть диаграмму последовательности, которая нужна для описания создания заказа и добавления товаров.



Ниже расположена диаграмма, которая описывает создание заказа, удаление из него товаров и обновления списка товаров в корзине.



Код для диаграммы:

@startuml

participant Client as c

participant Frontend as f

participant Backend as b

participant Database as d

activate c

c -> f: Создать заказ

activate f

f -> b: Отправить запрос на создание заказа (POST-запрос)

deactivate f

activate b

b -> d: Создать новый заказ в базе данных

deactivate b

activate d

d --> b: Заказ создан

deactivate d

b --> f: Отправить подтверждение создания заказа

deactivate b

activate f

f --> c: Отображаются товары в корзине

alt пользователь хочет добавить товар

c -> f: Добавить товар

activate f

f -> b: Отправить запрос на добавление товара (POST-запрос)

deactivate f

activate b

b -> d: Добавить товар в заказ

deactivate b

activate d

d --> b: Товар добавлен

deactivate d

b --> f: Отправить подтверждение добавления товара

deactivate b

activate f

f --> c: обновление данных в отображаемой корзине

deactivate c

end

alt пользователь хочет удалить товар из корзины

activate c

c -> f: Удалить товар

activate f

f -> b: Отправить запрос на удаление товара (DELETE-запрос)

deactivate f

activate b

b -> d: Удалить товар из заказа

deactivate b

activate d

d --> b: Товар удален

deactivate d

b --> f: Отправить подтверждение удаления товара

deactivate b

activate f

f --> c: появляется временная иконка "товар удален"

deactivate f

deactivate c

end

alt пользователь хочет обновить корзину

activate c

c -> f: Обновить список товаров

activate f

f -> b: Отправить запрос на обновление списка товаров (PUT-запрос)

deactivate f

activate b

b -> d: Обновить список товаров заказа

deactivate b

activate d

d --> b: Список товаров обновлен

deactivate d

b --> f: Отправить обновленный список товаров

deactivate b

activate f

f --> c: Выводится обновленный список товаров

deactivate f

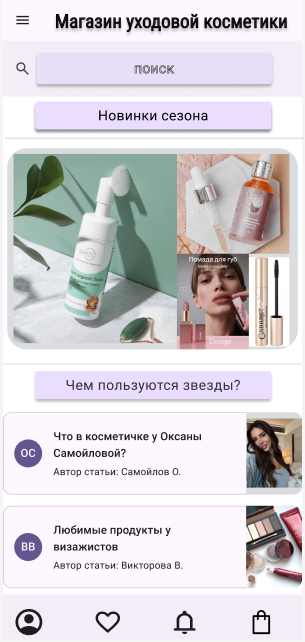
deactivate c

end

@enduml

1. Прототип экрана мобильного приложения

Задание: Подготовить прототип одного из экранов данного мобильного приложения и описать пользовательский интерфейс для данного экрана (например, создание заказа).

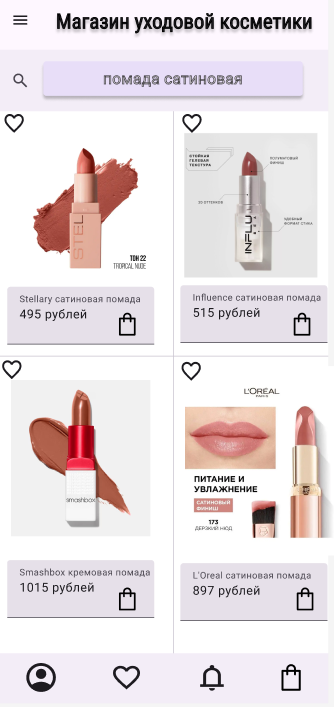


На экране инициализации приложения отображается верхний колонтитул с кнопкой меню (три горизонтальные линии) и названием магазина "Магазин уходовой косметики". На следующем уровне расположено поле для поиска товаров по характеристикам, где пользователь может ввести желаемую характеристику и нажать кнопку поиска (изображение лупы) для поиска соответствующих товаров.

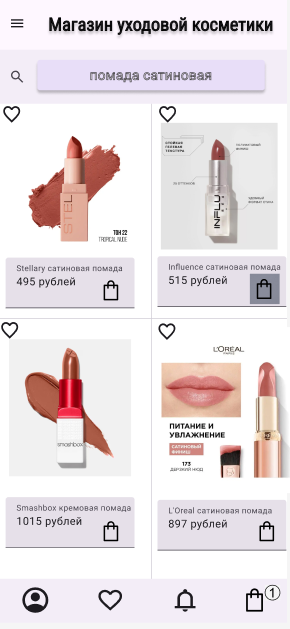
Продвигаясь далее по странице, пользователь обнаружит раздел "новинки сезона". Здесь представлена карусель с различными товарами, которые автоматически меняются каждые 15 секунд. Карусель является циклической и возвращается к первоначальному макету после определенного числа "слайдов". При нажатии на любое изображение в карусели пользователь переходит к описанию товара в магазине.

Далее на странице следует раздел "Чем пользуются звезды?", за которым расположены статьи. При нажатии на название статьи открывается статья, а все названия продуктов, фигурирующие в ней, являются ссылками на информацию о данных товарах магазина.

В нижнем колонтитуле страницы находятся четыре иконки: иконка человека для доступа к аккаунту пользователя, иконка сердца для избранных товаров, иконка звонка для уведомлений о проводимых акциях в магазине и иконка пакета для просмотра товаров, добавленных в корзину.



Рассмотрим пример, когда пользователь ищет «помада сатиновая». Нам высвечиваются (под знаком лупы и полем для поиска) товары, которые подходят под данную характеристику. На экране отображается фото товара, его название, цена, иконка пакет для добавления заказа в корзину, иконка сердце для добавления товара в избранное.



В данном случае пользователь хочет добавить «Influence сатиновая помада» в корзину. Иконка корзины, которая определена к данному товару, подсвечивается серым при нажатии, а у иконки пакета в нижнем колонтитуле страницы появляется значок, который отображает количество товаров в корзине (в примере это количество равно 1).

1. Описание функции редактирования заказа

Задание: Подготовить подробное описание функции редактирования заказа, 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).

* 1. Аудитория, использующая данную функцию

Данная функция предназначена для женщин всех возрастов, мужчин (выросла популярность косметики для мужчин), любителей брендов, любителей натуральной косметики.

* 1. Аудитория, которой предназначен документ

Данный документ предназначен для архитекторов решений, тимлидов и разработчиков.

* 1. Решаемая задача

Написать функцию редактирования заказа, которая позволит клиентам менять товары в корзине (добавлять или же удалить их из нее).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код требования | Как кто | Я хочу | Для чего | Приоритет требования |
| 1.3.1-1 | Клиент | иметь возможность редактировать заказ в корзине перед оформлением покупки | чтобы вносить изменения в свой заказ, добавлять или удалять товары, изменять их количество или варианты | Высокий |
| 1.3.1-2 | Администратор | иметь возможность просматривать историю изменений заказа в корзин | для отслеживания действий пользователей, связанных с редактированием заказов, и предоставления поддержки при необходимости | Средний |
| 1.3.1-3 | Разработчик | обеспечить безопасность и целостность данных при редактировании заказа в корзине | чтобы избежать утечек данных, ошибок или несанкционированного доступа при изменении заказов пользователей | Высокий |

* 1. Глоссарий

|  |  |
| --- | --- |
| Термин | Определение |
| Редактирование заказа | процесс изменения параметров или содержания заказа, который совершает пользователь в интернет-магазине. |
| История изменений | это запись или список всех произошедших изменений или действий, выполненных пользователем в отношении заказа в интернет-магазине. |
| Несанкционированный доступ | доступ к закрытой для публичного доступа информации со стороны лиц, не имеющих разрешения на доступ к этой информации. |
| Целостность данных | состояние информации, при котором отсутствует любое её изменение либо изменение осуществляется только преднамеренно субъектами, имеющими на него право. |

* 1. Критерии качества на входе(DoR)

1. Бизнес-требования согласованы с заказчиком и руководителем проекта;

2. Системные требования согласованы с лидером разработки и руководителем проекта;

3. Задачи поставлены ясно и понятны для всех участников команды, нет открытых вопросов;

4. У задачи описаны адекватные критерии приемки;

5. К задачам приложена диаграмма последовательности;

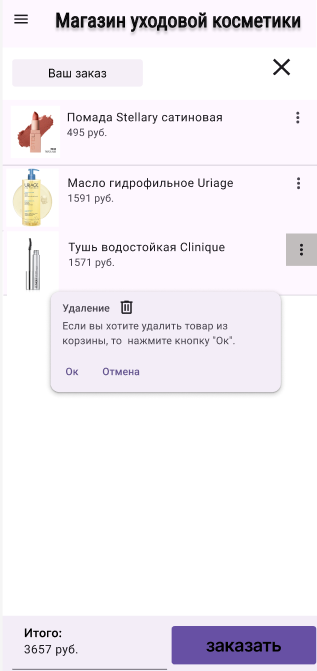
6. Для задач установлены зависимости.

* 1. Ограничения

1. Пользователь обязательно авторизован;

2. Редактирование заказа пользователем никак не влияет на работу процесса регистрации администраторов или пользователей административной панели интернет-магазина.

* 1. Прототип



* 1. Концепция решения

Результат будут использовать клиенты для редактирования (удаления или добавления товаров) в корзине.

* 1. Функциональные решения

1. Пользователь должен иметь возможность изменить количество товаров в заказе.

2. Пользователь должен иметь возможность добавить новый товар в заказ.

3. Пользователь должен иметь возможность удалить товар из заказа.

4. При изменении заказа, система должна автоматически пересчитать общую стоимость заказа.

5. Пользователь должен иметь возможность сохранить изменения в заказе или отменить их.

6. Система должна предоставлять возможность просмотра истории изменений заказа.

7. Возможность смены адреса, на который будет доставлен заказ.

8. Возможность восстановить аккаунт в магазине через номер телефона или электронную подпись, для восстановления заказа, который лежит в корзине.

9. При ошибке, возникающей в работе приложения, присылать пользователю сообщение об ошибке, а также указание причины (неисправность приложения/неверные данные для входа/плохой интернет у пользователя)

10. Пользователь не может заказать более 50 позиций.

* 1. Нефункциональные требования

1. Система должна обеспечивать оперативное обновление информации о заказе без задержек.

2. Интерфейс функции редактирования заказа должен быть интуитивно понятным и удобным для пользователя.

3. Должна быть возможность увеличения масштаба шрифта в заказе для плохо видящих пользователей магазина.

4. Система должна обеспечивать безопасность данных пользователя при редактировании заказа.

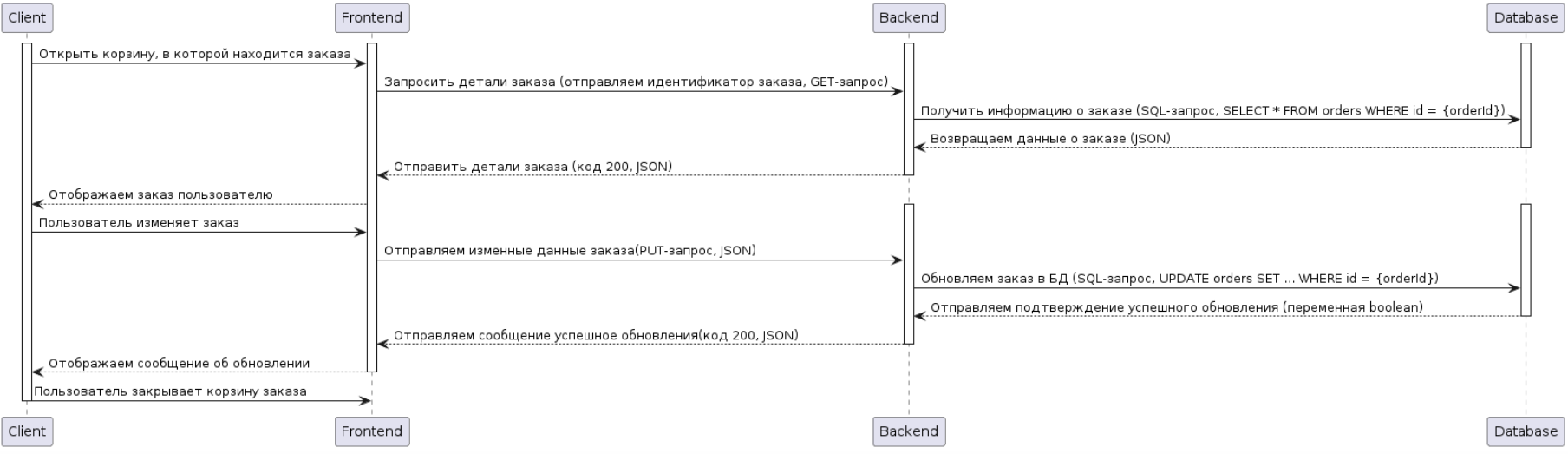
5. Время ответа системы на действия пользователя при редактировании заказа не должно превышать определенного значения (3 секунды).

6. Система должна быть доступна для использования на различных устройствах (компьютер, планшет, мобильный телефон).

7. Система должна быть способной обрабатывать большое количество одновременных запросов на редактирование заказов.

8. Система должна быть совместима с разными операционными системами.

* 1. Диаграмма Sequence



На данной диаграмме в общем виде (более подробно удаление, добавление и обновление заказа показаны на диаграммах в задании 2) показано редактирование заказа. Пользователь заходит в корзину заказа, происходит GET-запрос от фронтенда к бекенду приложения. Бекенд в свою очередь через SQL-запрос запрашивает у базы данных необходимую информацию. База возвращает данные в JSON, бекенд принимает эти данные и отправляет с кодом 200 и данными в формате JSON на фронтенд, который отображает все пользователю. Таким образом, пользователь видит товары в корзине, и начинает редактирование.

Пользователь изменяет заказ, данные изменения в виде PUT-запроса и данными в формате JSON отправляются с фронтенда на бекенд приложения. Бекенд отправляет SQL-запрос базе данных, которая должна обновить данные. База данных обновляет заказ и отправляет подтверждение об успешном исполнении операции бекенду, который отправляет на фронтенд код 200 и данные в формате JSON. На фронтенде отображается инфрмация об успешном обновлении и измененные данные.

На этом пользователь заканчивает обновление и закрывает корзину интернет магазина.

Код диаграммы:

participant Client

participant Frontend

participant Backend

participant Database

activate Client

Client->Frontend: Открыть корзину, в которой находится заказ

activate Frontend

Frontend->Backend: Запросить детали заказа (отправляем идентификатор заказа, GET-запрос)

activate Backend

Backend->Database: Получить информацию о заказе (SQL-запрос: SELECT \* FROM orders WHERE id = {orderId})

activate Database

Database-->Backend: Возвращаем данные о заказе (JSON)

deactivate Database

Backend-->Frontend Отправить детали заказа (код 200, JSON)deactivate Backend

Frontend-->Client: Отображается заказ пользователя

activate Backend

activate Database

Client->Frontend: Пользователь изменяет заказ

Frontend->Backend: Отправляем измененные данные заказа (PUT-запрос, JSON)

Backend->Database: Обновляем заказ в БД (SQL-запрос: UPDATE orders SET ... WHERE id = {orderId})

Database-->Backend: Отправляем подтверждение успешного обновления(переменная Boolean)

deactivate Database

Backend-->Frontend: Отправляем сообщение «успешное обновление» (код 200, JSON)

deactivate Backend

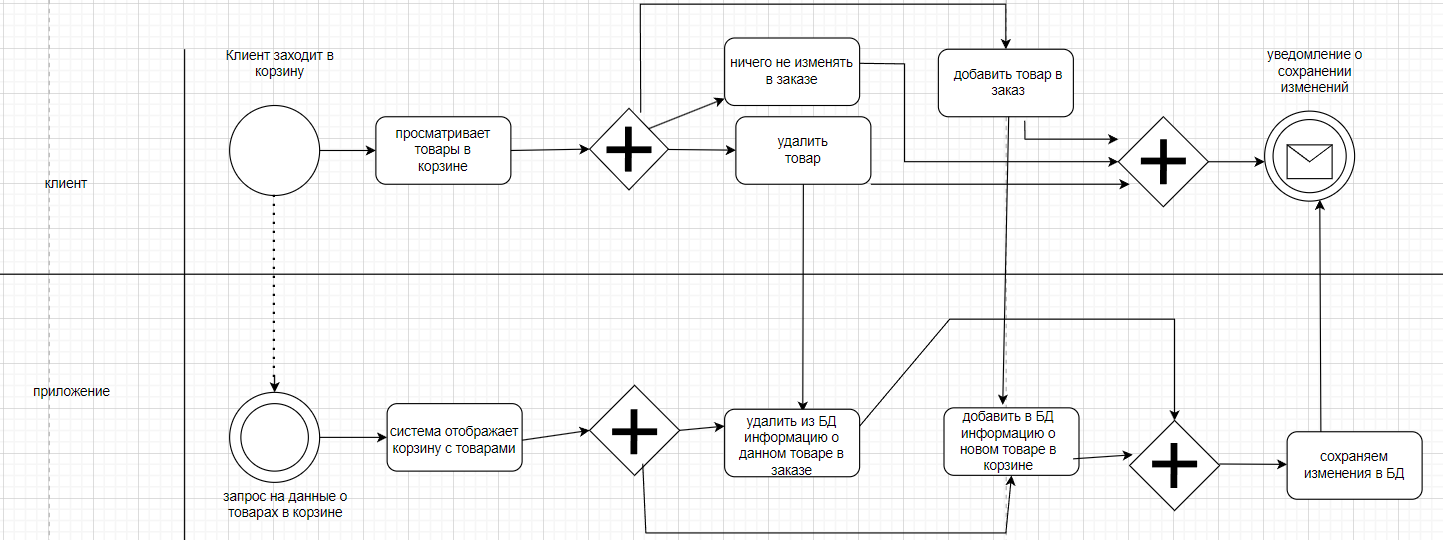
Frontend-->Client: Отображаем сообщение об обновалении

deactivate Frontend

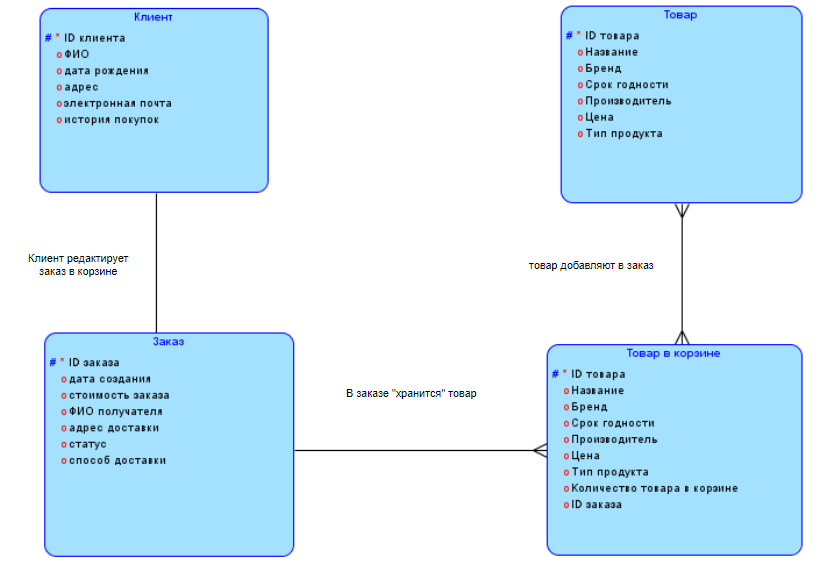
Client-->Frontend: Пользователь закрывает корзину заказа

deactivate Client

* 1. Бизнес-процесс в нотации BPMN



* 1. ER-модель



* 1. Use Cases

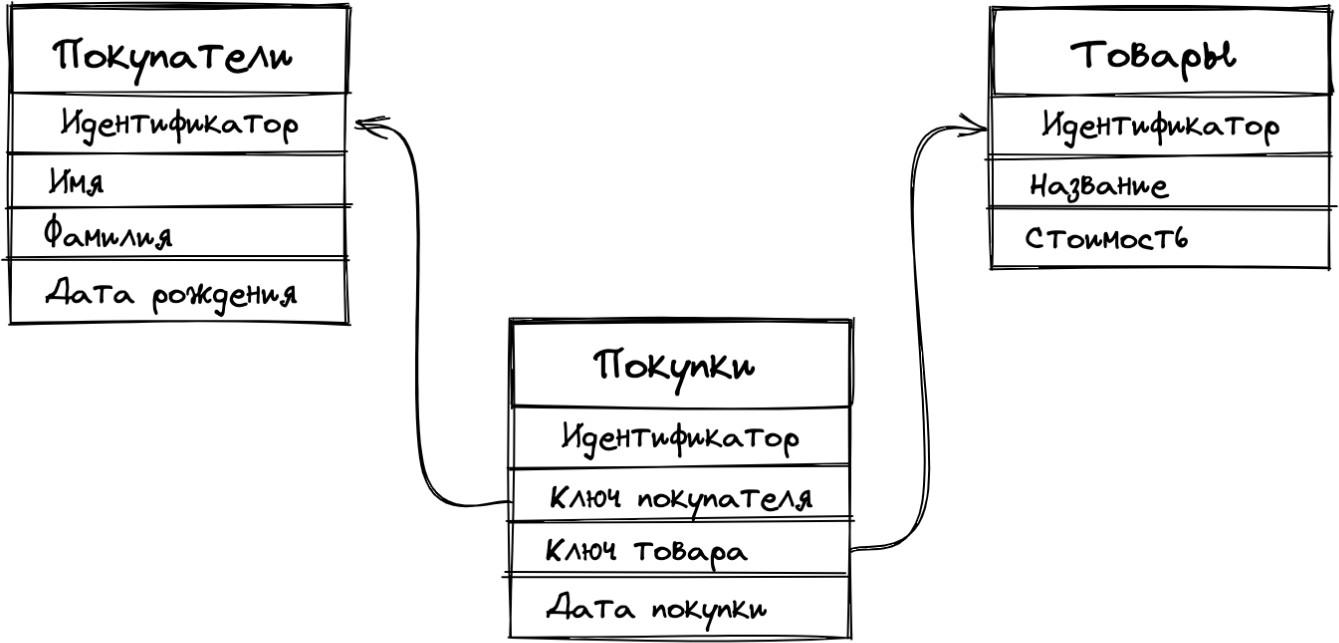
|  |  |
| --- | --- |
| Характеристика сценария | Возможные варианты действий |
| 1.Основной положительный | Пользователь редактирует заказ, выходит из системы система автоматически сохраняет изменения |
| 2.Негативный | Пользователь редактирует заказ, который в доставке система уведомляет пользователя о невозможности изменить заказ  Пользователь добавляет в заказ товар, который отсутствует на складе система уведомляет о невозможности добавить данный товар |
| 3.Дополнительный | Пользователь редактирует заказ после оплаты система пересчитывает заказ возвращает или просит доплатить деньги за заказ |

* 1. Описание структуры данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование поля | Тип поля | Размер | Уникаль-ность | Обязате-льность | Как используется |
| ID товара | ключ | 36 | Да | Да | Позволяет однозначно идентифицировать товар. |
| ID клиента | Ключ | 36 | Да | Да | Позволяет однозначно идентифицировать клиента. |
| ID заказа | ключ | 36 | Да | Да | Позволяет однозначно идентифицировать заказ. |
| ФИО/ФИО получателя | Строка | До 255 символов | Нет | Да | Обеспечивает правильную доставку заказа. Позволяет клиенту указать, кому нужно доставить заказ, если это не он сам. |
| Дата рождения | Дата | 8 | Нет | Нет | Может использоваться для персонализации маркетинговых сообщений. |
| Адрес/ адрес доставки | Строка | До 255 символов | Нет | Да | Обеспечивает правильную доставку заказа. |
| Электронная почта | Строка | До 70 символов | Да | Нет | Является стандартным способом идентификации клиента в интернет-магазинах. Позволяет легко отправлять массовые сообщения. |
| Дата создания | Дата | 8 | Нет | Да | Необходимо для расчета сроков доставки и выполнения заказов. |
| Стоимость заказа | Число | 36 | Нет | Да | Позволяет клиенту видеть общую стоимость заказа. |
| Статус | Строка | До 50 символов | Нет | Да | Позволяет отслеживать статус заказа в реальном времени. |
| Способ доставки | Строка | До 50 символов | Нет | Да | Позволяет клиенту выбрать наиболее удобный способ доставки. |
| Название | Строка | До 100 символов | Да | Да | Обеспечивает понимание, что именно продается. |
| Бренд | Строка | До 100 символов | Нет | Да | Позволяет проводить фильтрацию товаров по брендам. |
| Срок годности | Дата | 8 | Нет | Да | Позволяет магазину отслеживать сроки годности и предотвращать продажу просроченных товаров. |
| Производитель | Строка | До 100 символов | Нет | Да | Позволяет клиентам узнать, кто произвел товар. |
| Цена | Число | 36 | Нет | Да | Позволяет клиентам сравнивать цены различных товаров. |
| Тип продукта | Строка | До 100 символов | Нет | Да | Может быть использовано для проведения фильтрации товаров по типу. |
| Количество товара в корзине | Число | 36 | Нет | Да | Необходимо для расчета стоимости заказа. |

1. Запросы к базе данных

Задание: Перед вами реляционная модель данных. Необходимо написать SQL-запросы.



Вывести покупателей с числом осуществленных покупок

SELECT Покупатели.Имя, Покупатели.Фамилия, COUNT(Покупки.идентификатор)

FROM Покупатель

JOIN Покупки

ON Покупатели.идентификатор=Покупки.ключ\_покупателя

GROUP BY Покупатели.имя, Покупатели.Фамилия, Покупатели.идентификатор;

Вывести общую стоимость товаров для каждого покупателя и отсортировать результат в порядке убывания

* Общая стоимость товаров в конкретной покупке

SELECT Покупатели.Имя, Покупатели.Фамилия, SUM(Товары.стоимость)

FROM Покупатели

JOIN Покупки

ON Покупатели.идентификатор=Покупки.ключ\_покупателя

JOIN Товары

ON Товары.идентификатор=Покупки.ключ\_товара

GROUP BY Покупки.Идентификатор, Покупатели.Имя, Покупатели.Фамилия, Покупки.дата\_покупки

ORDER BY SUM(Товары.стоимость) DESC;

* Общая стоимость товаров за все покупки покупателя

SELECT Покупатели.Имя, Покупатели.Фамилия, SUM(Товары.стоимость)

FROM Покупатели

JOIN Покупки

ON Покупатели.идентификатор=Покупки.ключ\_покупателя

JOIN Товары

ON Товары.идентификатор=Покупки.ключ\_товара

GROUP BY Покупки.Идентификатор, Покупатели.Имя, Покупатели.Фамилия

ORDER BY SUM(Товары.стоимость) DESC;

**Получить покупателей, купивших только один товар**

* Покупатель мог покупать товар несколько раз, но это всегда была одна одинаковая позиция

SELECT DISTINCT Покупатели.Имя, Покупатели.Фамилия, Покупатели.Идентификатор

FROM Покупатели

JOIN

(SELECT Покупки.ключ\_покупателя

FROM Покупки

GROUP BY Покупки.ключ\_покупателя

HAVING COUNT(DISTINCT Покупки.ключ\_товара) = 1)

ON Покупатели.идентификатор=Покупки.ключ\_покупателя;

* Покупатель покупал только один раз в магазине только один товар

SELECT DISTINCT Покупатели.Имя, Покупатели.Фамилия, Покупатели.Идентификатор

FROM Покупатели

JOIN

(SELECT Покупки.ключ\_покупателя

FROM Покупки

GROUP BY Покупки.ключ\_покупателя

HAVING COUNT(DISTINCT Покупки.ключ\_товара) = 1 AND COUNT(Покупки.идентификатор) = 1)

ON Покупатели.идентификатор=Покупки.ключ\_покупателя;

* Покупатель хотя бы раз покупал только один товар в конкретной покупке

SELECT DISTINCT Покупатели.Имя, Покупатели.Фамилия, Покупатели.Идентификатор

FROM Покупатели

JOIN

(SELECT Покупки.ключ\_покупателя

FROM Покупки

GROUP BY Покупки.ключ\_покупателя, Покупки.дата\_покупки

HAVING COUNT(DISTINCT Покупки.ключ\_товара) = 1)

ON Покупатели.идентификатор=Покупки.ключ\_покупателя;