РЕФЕРАТ

Отчет 52 с., 9 рис., 24 табл., 24 источн., 1 прил.

ЭЛЕКТРОННАЯ ТОРГОВАЯ ПЛОЩАДКА, БАЗА ДАННЫХ, АРХИТЕКТУРА ПРИЛОЖЕНИЯ, API, POSTGRESQL, LOCUST

Цель работы — разработка базы данных для электронной торговой площадки.

В данной работе был проведен анализ предметной области электронных торговых площадок. Были формализованы требования к разрабатываемой базе данных и приложению. Спроектированы сущности базы данных, ролевая модель и ограничения целостности. Выбраны средства реализации базы данных и приложения. Разработано программное обеспечение, обеспечивающее интерфейс для доступа к базе данных. Проведено исследование зависимости времени выполнения запросов от их количества в секунду.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В эпоху цифровизации и глобализации электронной коммерции значимость электронных торговых площадок трудно переоценить. Они не только упрощают процесс купли-продажи, но и предлагают целый ряд преимуществ, таких как снижение затрат, расширение клиентской базы, улучшение логистики и маркетинга.

Электронные торговые площадки стали неотъемлемой частью современной экономики, радикально изменив способ осуществления покупки и продажи товаров и услуг. Этому также способствуют растущее доверие потребителей к интернет-торговле и стремительный рост числа сервисов доставки.

Цель работы — разработка базы данных для электронной торговой площадки.

Для достижения поставленной цели, необходимо решить следующие задачи:

— провести анализ предметной области электронных торговых площадок;

— формализовать требования к разрабатываемой базе данных и приложению;

— спроектировать сущности базы данных, ролевую модель и ограничения целостности;

— выбрать средства реализации базы данных и приложения;

— разработать программное обеспечение, обеспечивающее интерфейс для доступа к базе данных;

— провести исследование зависимости времени выполнения запросов от их количества в секунду.

1 Аналитический раздел

В данном разделе будет проведен анализ предметной области электронных торговых площадок, формализованы требования к разрабатываемой базе данных и приложению, выбрана модель данных среди рассмотреных существующих моделей.

1.1 Анализ предметной области

Электронные торговые площадки представляют собой веб-приложения, которые позволяют потребителям и продавцам взаимодействовать друг с другом, осуществляя сделки онлайн. Такие платформы предлагают пользователям удобный интерфейс для поиска, сравнения и приобретения товаров, а также делают возможным участие продавцов в глобальной торговле, превышая географические и временные ограничения.

При анализе известных решений были выделены следующие критерии:

— поиск продуктов по категориям;

— личный кабинет магазина;

— доступность оплаты в России;

— Объединение товаров из нескольких магазинов в один заказ.

В таблице 1 представлено сравнение известных решений.

Таблица 1 – Сравнение известных решений

Известное решение	**Поиск**	**Личный кабинет магазина**	**Доступность оплаты**	**Объединение товаров**
Ozon	+	+	+	-
Wildbreries	+	+	+	-
Aliexpress	+	+	+	-
Amazon	+	+	-	-

Ни одна из рассмотренных электронных торговых площадок не обладает полным набором функциональных возможностей, отвечающим всем выделенным критериям.

1.2 Формализация требований к приложению

Необходимо спроектировать и разработать приложение, позволяющее работать с базой данных для электронной торговой площадки. Разрабатываемое решение должно соответствовать всем критериям, выдвинутым в предыдущем пункте. Электронная торговая площадка должна предоставлять возможность взаимодействия с пользователями различных ролей:

— гость;

— продавец;

— покупатель;

— модератор.

Функциональные требования для каждой из ролей представлены на рисунке 1.

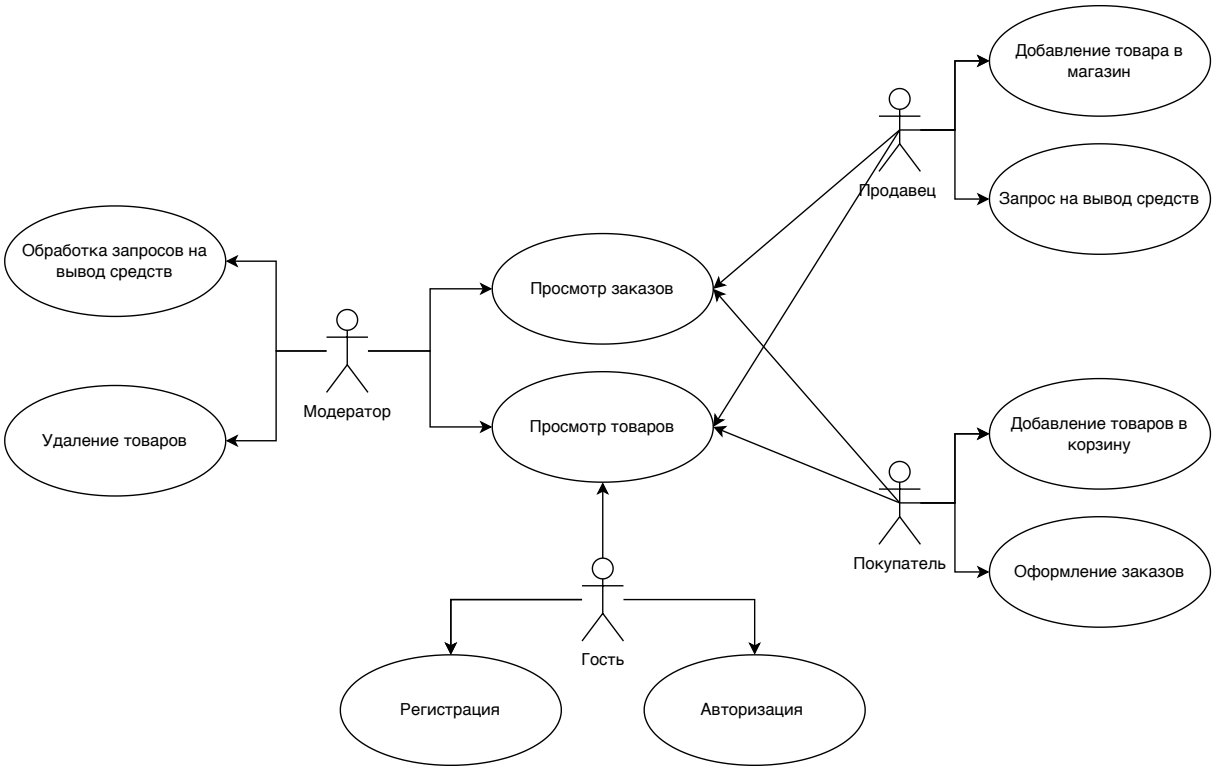

Рисунок 1 – Диаграмма использования приложения

1.3 Формализация требований к базе данных

База данных должна иметь возможность содержать данные о:

— пользователях;

— корзинах;

— магазинах;

— товарах;

— заявках о выводе средств;

— заказах магазинов;

— заказах покупателей.

Сущности, их атрибуты и связи между ними представлены в ER-диаграмме на рисунке 2.

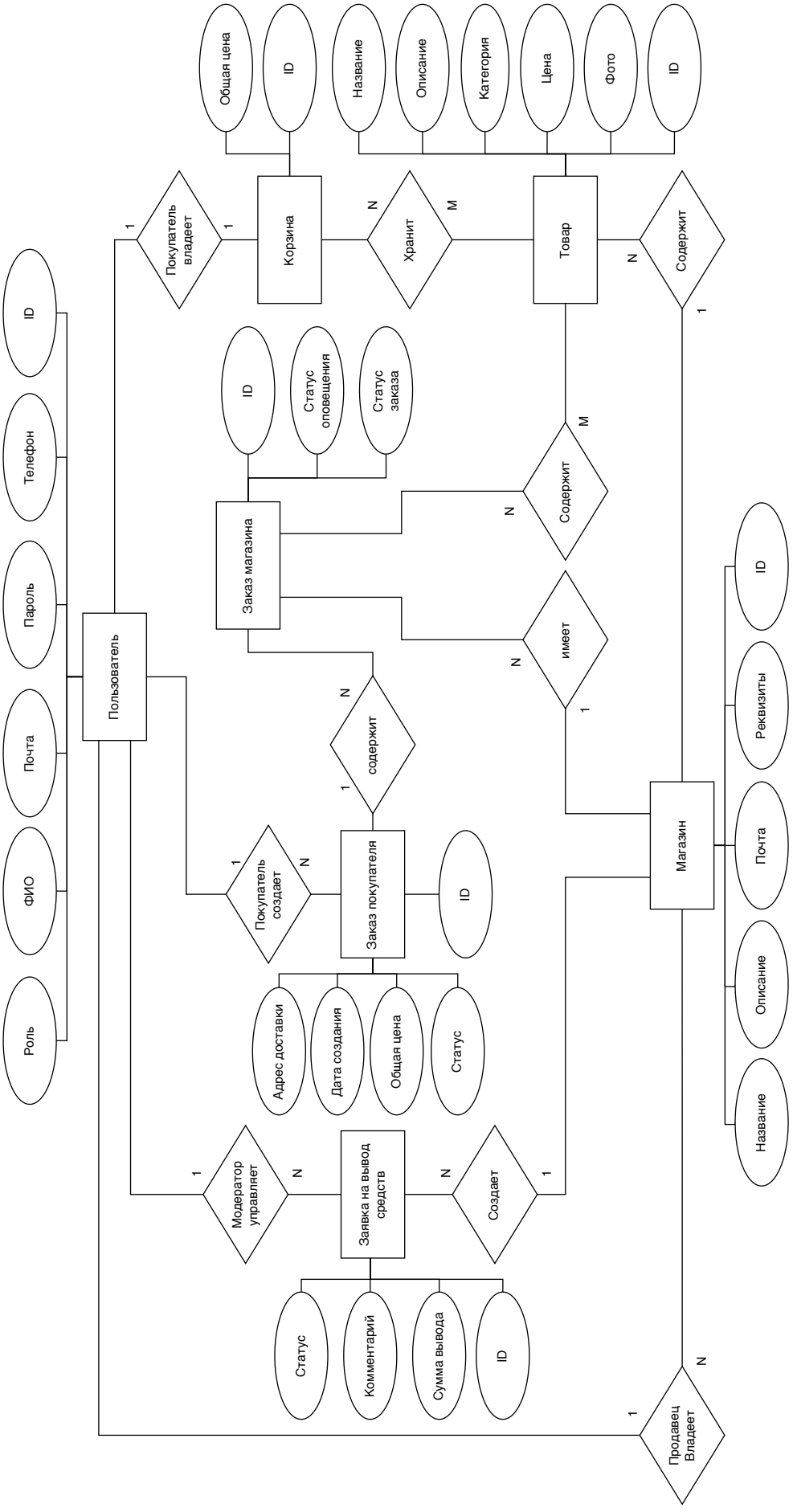

Рисунок 2 – ER-модель в нотации Чена

1.4 Модели данных

Модель данных — набор концепций, которые могут быть использованы для описания структуры базы данных — обеспечивает необходимые средства для достижения этой абстракции. Под структурой базы данных принято подразумевать типы данных, связи и ограничения, которые применяются к данным [1].

Базы данных по модели хранения делятся три основные группы [2]:

— дореляционные;

— реляционные;

— постреляционные.

Каждая модель хранения данных предоставляет различные механизмы для взаимодествия с данными. При выборе модели данных следует учитывать конкретную задачу, для которой она предназначена.

1.4.1 Дореляционные базы данных

Существует несколько типов дореляционных баз данных, наиболее распространенными из которых являются [2]:

— иерархические;

— сетевые;

— инвертированные списки.

В иерархической модели данные организованы в древовидную структуру, где каждый узел может иметь несколько дочерних узлов, но только один родительский узел. Это создает проблемы при попытке отразить отношения «многие ко многим» [3].

В сетевой модели данные организованы в графоподобную структуру, где узлы могут иметь несколько родительских и дочерних узлов, создавая более гибкие отношения между данными. Языки запросов для сетевой модели часто не так гибки, как SQL для реляционных баз данных, что усложняет выполнение сложных запросов [3].

Инвертированные списки позволяют индексировать данные по значениям ключей, что позволяет быстро находить записи по указанному ключу. Данная модель не всегда подходит для работы с отношениями «многие ко многим». Она предназначена для быстрого поиска по ключевым словам, а не для сложного моделирования данных [3].

1.4.2 Реляционные базы данных

Основателем теории реляционных баз данных является британский ученый Эдгар Кодд, который в 1970 году представил свою первую работу, посвященную реляционной модели данных. Наиболее распространенное определение реляционной модели принадлежит Кристоферу Дейту. Согласно Дейту, реляционная модель включает три составные части [2]:

— структурную;

— целостностную;

— манипуляционную.

Структурная составляющая реляционной модели определяет, из каких элементов она состоит. К основным концепциям этой части модели относятся тип данных, домен, атрибуты, схема отношения, схема базы данных, кортеж, отношение, а также потенциальные, первичные и альтернативные ключи реляционной базы данных [2].

Внутри целостностной части реляционной модели выделяются два основных принципа целостности, которые должны соблюдаться во всех отношениях в реляционных базах данных. Это целостность сущностей и ссылочная целостность [2].

Манипуляционная часть реляционной модели описывает два эквивалентных метода управления реляционными данными — реляционную алгебру и реляционное исчисление [2].

1.4.3 Постреляционные базы данных

Постреляционные базы данных представляют собой современные системы управления базами данных, которые развились из реляционных моделей

в ответ на растущие требования к обработке больших объемов данных, масштабируемости и гибкости хранения. Эти базы данных предлагают альтернативный подход к организации, хранению и извлечению данных, что делает их особенно популярными в средах, где традиционные реляционные базы данных могут испытывать сложности [4].

Постреляционные базы данных могут не поддерживать такие строгие ограничения целостности, как реляционные, что может привести к проблеме неконсистентности данных [4].

1.5 Выбор модели данных

С учетом особенности задачи была выбрана реляционная модель хранения данных. Дореляционные модели испытывают проблемы при попытке отразить в них сложные отношения, также манипуляционная часть данных моделей не является достаточно гибкой. Постреляционные базы данных могут не поддерживать требуемые ограничения целостности.

2 Конструкторский раздел

В данном разделе будет представлена схема проектируемой базы данных, будут описаны сущности базы данных, ограничения целостности, ролевая модель и используемый триггер.

2.1 Описание сущностей базы данных

На рисунке 3 представлена схема проектируемой базы данных.

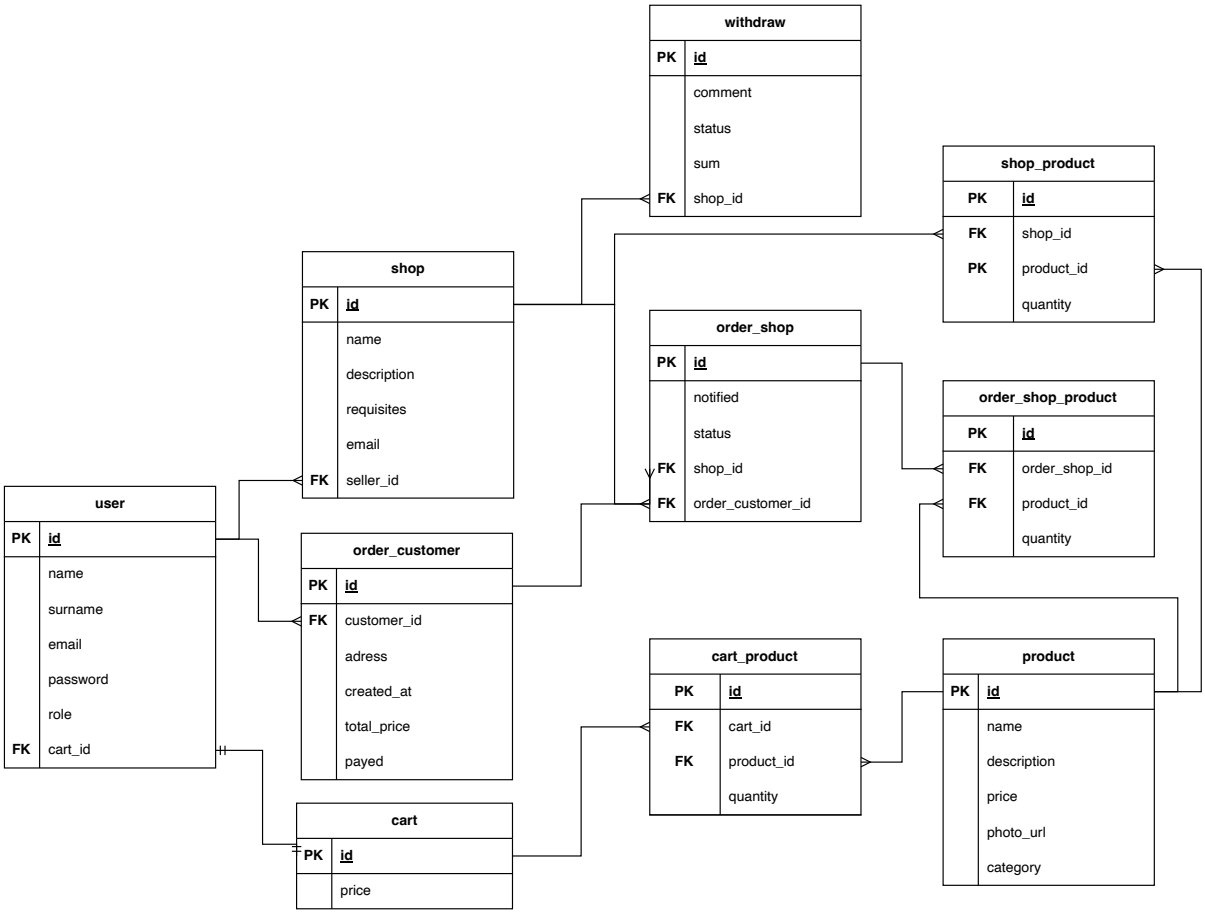

Рисунок 3 – Схема проектируемой базы данных

Необходимы следующие таблицы:

1) Таблица с пользователями (user).

2) Таблица с корзинами (cart).

3) Таблица с магазинами (shop).

4) Таблица с товарами (product).

5) Таблица с заказами покупателей (order_customer).

6) Таблица с заказами магазинов (order_shop).

7) Таблица с заявками на вывод средств (withdraw).

8) Развязочная таблица для корзин и их продуктов (cart_product).

9) Развязочная таблица для магазинов и их продуктов (shop_product).

10) Развязочная таблица для заказов магазинов и их продуктов (order_shop_product).

В таблицах 2 — 11 будут описаны поля таблиц схемы проектируемой базы данных.

Таблица 2 – Поля таблицы user

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор пользователя
cart_id	uuid	Идентификатор корзины
name	строка	Имя пользователя
surname	строка	Фамилия пользователя
email	строка	Электронная почта пользователя
password	строка	Пароль для авторизации
phone	строка	Номер телефона
role	перечисляемый	Роль пользователя

Таблица 3 – Поля таблицы cart

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор корзины
price	целое число	Общая цена товаров в корзине

Таблица 4 – Поля таблицы shop

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор магазина
seller_id	uuid	Идентификатор продавца
name	строка	Название магазина
description	строка	Описание магазина
requisites	строка	Реквизиты магазина
email	строка	Электронная почта магазина

Таблица 5 – Поля таблицы product

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор продукта
name	строка	Название продукта
description	строка	Описание продукта
price	целое число	Цена продукта
category	перечисляемый	Категория продукта
photo_url	строка	Ссылка на фотографию продукта

Таблица 6 – Поля таблицы order_customer

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор заказа
customer_id	uuid	Идентификатор клиента
address	строка	Адрес доставки
created_at	временная метка	Время создания заказа
total_price	целое число	Общая сумма заказа
payed	бинарный	Статус оплаты (оплачен или нет)

Таблица 7 – Поля таблицы Order_Shop

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор заказа магазина
shop_id	uuid	Идентификатор магазина
order_customer_id	uuid	Идентификатор заказа клиента
status	перечисляемый	Статус заказа
notified	бинарный	Был ли уведомлен магазин

Таблица 8 – Поля таблицы withdraw

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор вывода средств
shop_id	uuid	Идентификатор магазина
comment	строка	Комментарий к выводу средств
sum	целое число	Сумма для вывода
status	перечисляемый	Статус вывода

Таблица 9 – Поля таблицы cart_product

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор записи
cart_id	uuid	Идентификатор корзины
product_id	uuid	Идентификатор продукта
quantity	целое число	Количество товара в корзине

Таблица 10 – Поля таблицы shop_product

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор записи
shop_id	uuid	Идентификатор магазина
product_id	uuid	Идентификатор продукта
quantity	целое число	Количество товара в магазине

Таблица 11 – Поля таблицы order_shop_product

Поле	**Тип данных**	**Описание**
id	uuid	Идентификатор записи
order_shop_id	uuid	Идентификатор заказа магазина
product_id	uuid	Идентификатор продукта
quantity	целое число	Количество товара в заказе

2.2 Описание ограничений целостности базы данных

В таблицах 12 — 21 будут описаны ограничения целостности проектируемой базы данных.

Таблица 12 – Ограничения целостности таблицы user

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
cart_id	uuid	Внешний ключ на поле id таблицы cart, уникальный
name	строка	Не пустое значение
surname	строка	Не пустое значение
email	строка	Не пустое значение, уникальное
password	строка	Не пустое значение
phone	строка	—
role	перечисляемый	Не пустое значение

Таблица 13 – Ограничения целостности таблицы cart

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
price	целое число	Не пустое значение, значение больше или равно 0

Таблица 14 – Ограничения целостности таблицы shop

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
seller_id	uuid	Внешний ключ на поле id таблицы user
name	строка	Не пустое значение
description	строка	Не пустое значение
requisites	строка	Не пустое значение
email	строка	Не пустое значение, уникальное

Таблица 15 – Ограничения целостности таблицы product

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
name	varchar(255)	Не пустое значение
description	строка	Не пустое значение
price	целое число	Не пустое значение, больше 0
category	перечисляемый	Не пустое значение
photo_url	строка	—

Таблица 16 – Ограничения целостности таблицы order_customer

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
customer_id	uuid	Внешний ключ на поле id таблицы user
address	строка	Не пустое значение
created_at	временная метка	Не пустое значение
total_price	целое число	Не пустое значение, больше 0
payed	бинарный	Не пустое значение

Таблица 17 – Ограничения целостности таблицы order_shop

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
shop_id	uuid	Внешний ключ на поле id таблицы shop
order_customer_id	uuid	Внешний ключ на поле id таблицы order_customer
status	перечисляемый	Не пустое значение
notified	бинарный	Не пустое значение

Таблица 18 – Ограничения целостности таблицы withdraw

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
shop_id	uuid	Внешний ключ на поле id таблицы shop
comment	строка	Не пустое значение
sum	целое число	Не пустое значение, больше 0
status	перечисляемый	Не пустое значение

Таблица 19 – Ограничения целостности таблицы cart_product

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
cart_id	uuid	Внешний ключ на поле id таблицы cart
product_id	uuid	Внешний ключ на поле id таблицы product
quantity	целое число	Не пустое значение, больше 0

Таблица 20 – Ограничения целостности таблицы shop_product

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
shop_id	uuid	Внешний ключ на поле id таблицы shop
product_id	uuid	Внешний ключ на поле id таблицы product
quantity	целое число	Не пустое значение, больше или равно 0

Таблица 21 – Ограничения целостности order_shop_product

Поле	**Тип**	**Ограничение**
id	uuid	Первичный ключ
order_shop_id	uuid	Внешний ключ на поле id таблицы order_shop, не пустое значение
product_id	uuid	Внешний ключ на поле id таблицы product, не пустое значение
quantity	целое число	Не пустое значение, значение больше или равно 0

2.3 Описание ролевой модели

В данном подразделе отписана ролевая модель, исходя из сформулированных требований к приложению.

2.3.1 Гость

Гость имеет права доступа SELECT к таблицам:

— user;

— shop_product;

— cart;

— product.

Также имеет права INSERT к таблицам:

— user;

— cart.

Гость может просматривать данные пользователей, продуктов, корзин и товаров, а также добавлять записи в таблицы пользователей и корзин.

2.3.2 Покупатель

Покупатель имеет права доступа SELECT к таблицам:

— user;

— shop_product;

— cart;

— cart_product;

— order_customer;

— product.

Также имеет права INSERT к таблицам:

— cart_product;

— order_customer.

Имеет права UPDATE к таблицам:

— user;

— cart;

— cart_product;

И права DELETE к таблице:

— cart_product.

Покупатель может просматривать данные пользователей, корзин, товаров, продуктов и заказов, добавлять товары в корзину и заказы, редактировать свои данные, корзину и товары в корзине, а также удалять товары из своей корзины.

2.3.3 Продавец

Продавец имеет права доступа SELECT к таблицам:

— user;

— shop_product;

— shop;

— order_shop;

— withdraw;

— product.

Также имеет права UPDATE к таблицам:

— user;

— orde_shop;

— shop_product.

Имеет права INSERT к таблицам:

— shop;

— withdraw;

— shop_product;

— product.

Продавец может просматривать данные пользователей, товаров, магазинов, заказов и заявок на вывод средств, обновлять информацию о товарах и заказах, а также создавать новые магазины, товары и заявки на вывод средств.

2.3.4 Модератор

Модератор имеет права доступа SELECT к таблицам:

— user;

— shop_product;

— withdraw;

— product.

Также имеет права UPDATE к таблицам:

— withdraw;

— user.

Имеет права DELETE к таблице:

— shop_product.

Модератор может просматривать данные пользователей, товаров и заявок на вывод средств, обновлять информацию о пользователях и заявках, а также удалять товары магазинов из базы данных.

2.4 Описание триггера базы данных

При добавлениии пользователем товара в корзину, необходимо добавить новую запись в развязочную таблицу для корзин и их продуктов (cart_product). При этом общая цена товаров в корзине хранится в таблице корзин (cart) в поле price. Поэтому нобходим триггер, реализующий обновление общей цены товаров в корзине, при добавлении, обновлении и удалении товаров из нее. При попытке добавления новой записи, ее удаления или изменения в таблицу cart_product вызывается функция, схема алгоритма работы которой представлена на рисунке 4.

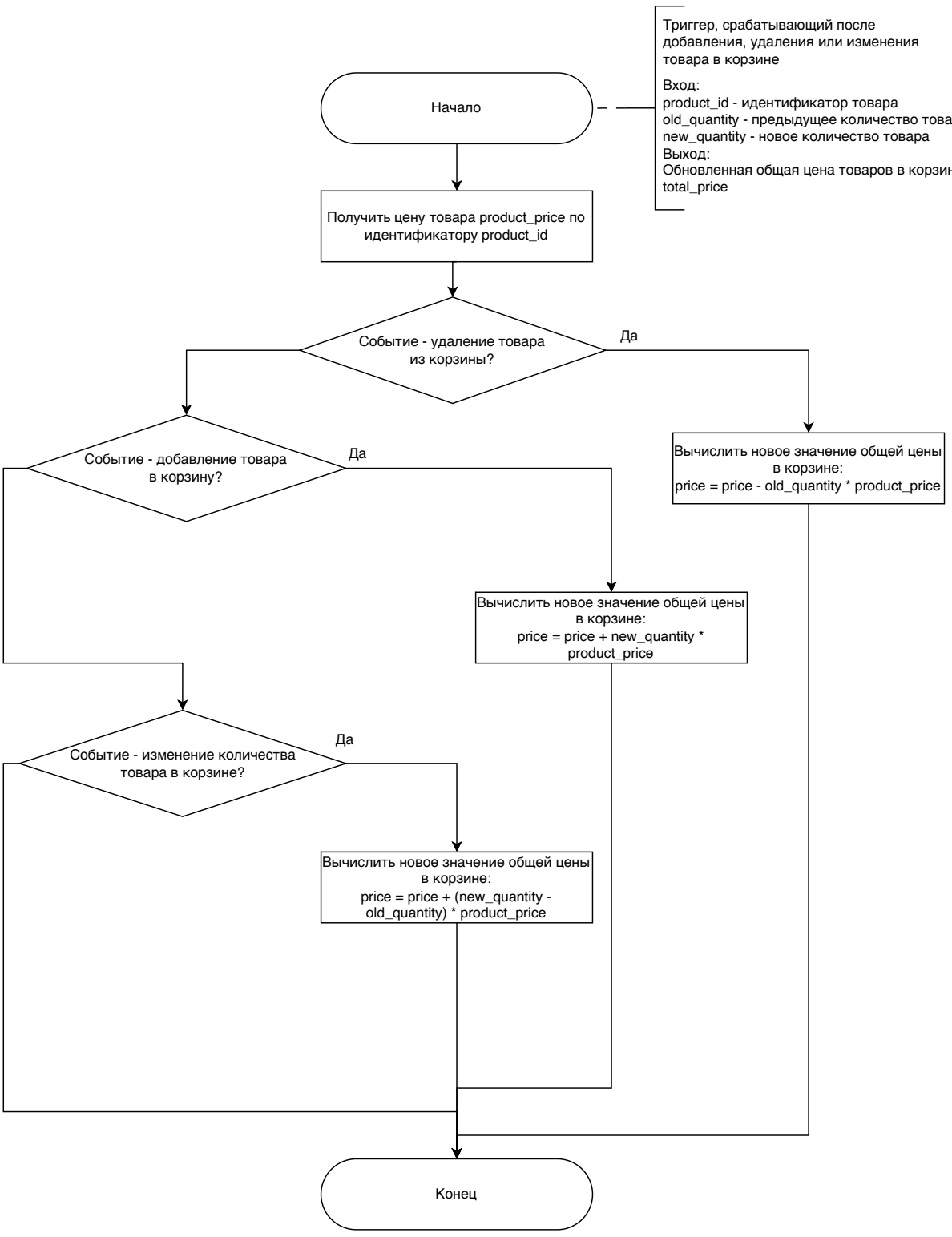

Рисунок 4 – Схема алгоритма работы триггера, отвечающего за обновление общей цены товаров в корзине

Вывод

В данном разделе была представлена представлена схема проектируемой базы данных, были описаны сущности базы данных, ограничения целостности, ролевая модель и используемый триггер.

3 Технологический раздел

В данном разделе будут выбораны и описаны средства реализации, описаны архитектура приложения и детали реализации, представлены описание интерфейса доступа к базе данных, а также метод тестирования компонента уровня доступа к базе данных.

3.1 Выбор средств реализации

В подразделах ниже будет проведен выбор средств реализации базы данных и приложения.

3.1.1 Выбор системы управления базами данных

На данный момент существует огромное множество различных реляционных систем управления базами данных с поддержкой ролевой модели. При выборе системы управления базами данных были выделены критерии:

— бесплатное и открытое программное обеспечение;

— поддержка процедурных расширений языка SQL;

— поддержка многопользовательского доступа.

В таблице 22 представлено сравнение систем управления базами данных.

Таблица 22 – Сравнение систем управления базами данных

Система управления базами данных	**бесплатное и открытое программное обеспечение**	**поддержка процедурных расширений языка SQL**	**поддержка многопользовательского доступа**
PostgreSQL [5]	+	+	+
SQLite [6]	+	+	-
MySQL [7]	+	-	+
Oracle [8]	-	+	+

Так как PostgreSQL удовлетворяет всем перечисленным критериям, качестве системы управления базами данных в данной работе будет использоваться PostgreSQL.

3.1.2 Выбор языка программирования

Для разработки проекта был выбран язык программирования Golang [9]. Основные причины выбора Golang следующие:

— golang компилируется в машинный код, что обеспечивает высокую скорость выполнения программ и эффективное использование ресурсов системы;

— язык обладает минималистичным синтаксисом, что упрощает процесс разработки и снижает вероятность ошибок;

— golang имеет активное сообщество и широкую поддержку различных инструментов и библиотек.

3.2 Средства реализации

В рамках данной работы были выбраны следующие технологии:

1) язык программирования — Golang [9];

2) система управления базами данных — PostgreSQL [5];

3) для работы с СУБД была выбрана библиотека database/sql [10], представляющая универсальный интерфейс для реляционных баз данных, а также ее расширение — библиотека sqlx [11];

4) фреймворк для реализации API [12] — Gin [13];

5) для документации API был использован Swagger [14];

6) аутентификация была реализована на основе Bearer-токенов [15];

7) для сессионного хранилища использовался Redis [16];

8) для объектного хранилища использовался MinIO [17];

9) для изолирования приложения была выбрана платформа Docker [18].

3.3 Архитектура приложения

Для реализации приложения электронной торговой площадки была выбрана клиент-серверная архитектура. Доступ к приложению будет осуществляться через API. Также пользователь имеет возможность пользоваться приложением через технический интерфейс (консоль).

Для реализации серверной части приложения была использована гексагональная архитектура [19]. Она была разработана для сегрегации бизнес логики от внешних слоёв системы. Основной принцип гексагональной архитектуры заключается в разделении системы на ядро и внешние слои. Ядро (бизнес логика) не зависит от внешних слоёв. Взаимодействие с внешними системами осуществляется с помощью портов, к которым подключаются адаптеры, реализующие конкретную логику взаимодействия. Это позволяет добиться более гибкой и расширяемой архитектуры, а также упрощает тестирование и замену компонентов.

Приложение разделено на следующие компоненты верхнего уровня:

1) технический интерфейс;
2) бизнес логика;
3) уровень доступа к базе данных;
4) платежный шлюз;
5) email provider;
6) auth provider;
7) объектное хранилище;
8) HTTP API.

Технический интерфейс — это на данный момент основной интерфейс взаимодействия пользователя с приложением, предоставляющий доступ к функциям и операциям через консольные команды.

Бизнес логика — отвечает за реализацию основной функциональности приложения. Этот компонент обрабатывает запросы пользователей, выполняет все ключевые операции, связанные с логикой приложения.

Уровень доступа к базе данных — компонент, обеспечивающий взаимодействие с системой управления базами данных. Здесь происходит создание запросов к базе данных, получение данных и их обработка перед отправкой в бизнес-логику.

Платежный шлюз — отвечает за обработку платежей и взаимодействие с внешними платежными системами. Компонент реализует интеграцию с API платежных сервисов для обработки транзакций и управления платежными данными.

Email Provider — Компонент для отправки электронной почты с уведомлениями о заказах. Для для обеспечания гарантии доставки использовался архитектурный паттерн transactional outbox [20].

Auth Provider — отвечает за процессы аутентификации и авторизации пользователей. Он предоставляет Bearer токены доступа для дальнейших действий пользователей.

Объектное хранилище — компонент, предназначенный для хранения изображений. Объектное хранилище предоставляет интерфейс для сохранения и извлечения объектов данных.

HTTP API — этот компонент предоставляет интерфейс для взаимодействия с приложением через HTTP [21] запросы. Он обрабатывает запросы, поступающие от внешних клиентов или сервисов, и возвращает необходимые данные.

Верхнеуровневое разбиение на компоненты представлено на рисунке 5.

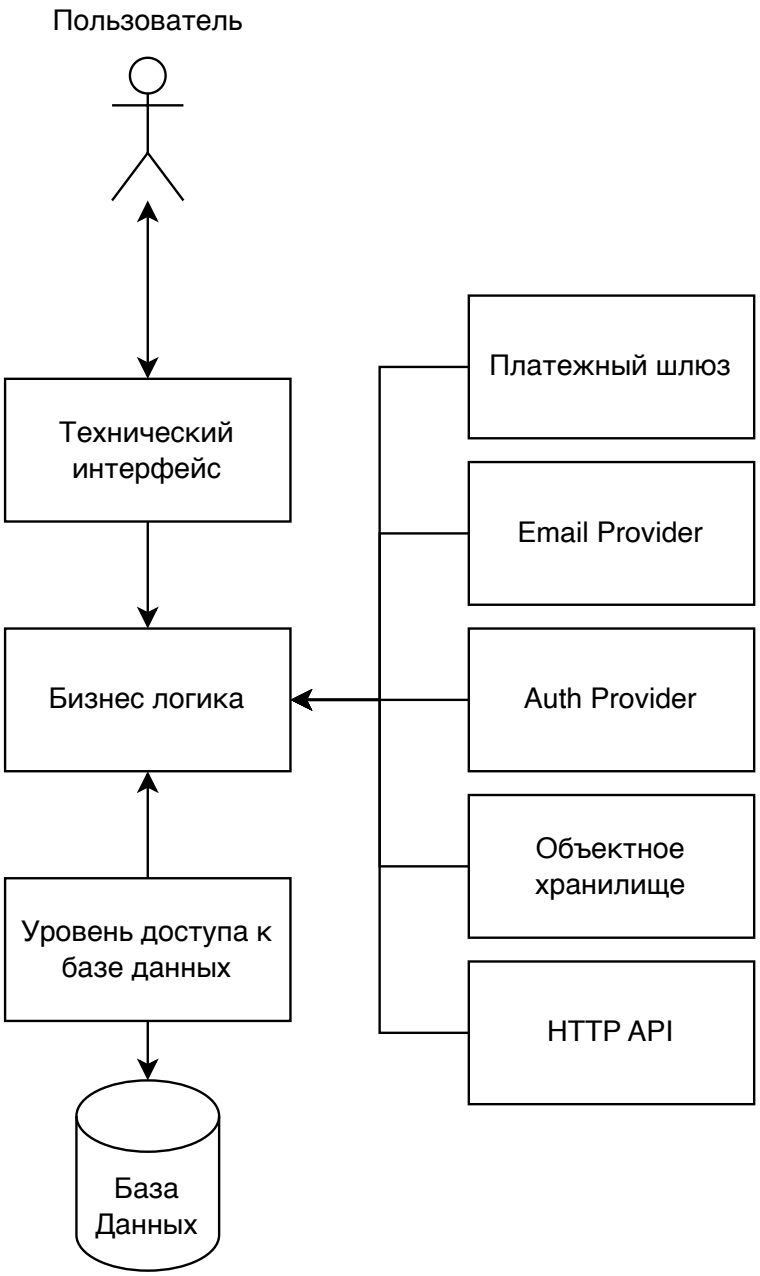

Рисунок 5 – Верхнеуровневое разбиение на компоненты

3.4 Детали реализации

В данном разделе будут описаны детали реализации базы данных.

3.4.1 Создание таблиц

В конструкторском разделе были описаны таблицы базы данных. В листинге 1 представлен соответствующий код создания таблиц.

Листинг 1 – Скрипт создания таблиц базы данных

```
create table public.cart (
    id uuid,
    price bigint
);
create type user_role as enum ('Customer', 'Seller',
   'Moderator');
create table public.user (
    id uuid,
    cart_id uuid,
    name varchar(255),
    surname varchar(255),
    email varchar(255),
    password varchar(255),
    phone varchar(32),
    role user_role
);
create type product_category as enum ('Electronic', 'Fashion',
   'Home', 'Health', 'Sport', 'Books');
create table public.product (
    id uuid,
    name varchar(255),
    description text,
    price bigint,
    category product_category,
    photo_url text
);
create table public.cart_product (
    id uuid,
    cart_id uuid not null,
    product_id uuid not null,
    quantity bigint not null
);
create table public.shop (
    id uuid,
    seller_id uuid,
    name varchar(255),
```

```
    description text,
    requisites text,
    email varchar(255)
);
create table public.shop_product (
    id uuid,
    shop_id uuid,
    product_id uuid,
    quantity bigint
);
create type withdraw_status as enum ('Start', 'Ready', 'Done');
create table public.withdraw (
    id uuid,
    shop_id uuid,
    comment text,
    sum bigint,
    status withdraw_status
);
create table public.order_customer (
    id uuid,
    customer_id uuid,
    address text,
    created_at timestamp,
    total_price bigint,
    payed boolean
);
create type order_shop_status as enum ('Start', 'Ready', 'Done');
create table public.order_shop (
    id uuid,
    shop_id uuid,
    order_customer_id uuid,
    status order_shop_status,
    notified boolean
);
create table public.order_shop_product (
    id uuid,
    order_shop_id uuid,
    product_id uuid,
    quantity bigint
);
```

3.4.2 Создание ограничений целостности

В конструкторском разделе были описаны ограничения целостности. В листинге 2 представлен соответствующий код создания ограничений целостности.

Листинг 2 – Скрипт создания ограничений целостности базы данных

```
alter table public.cart
    add constraint pk_cart_id primary key (id),
    alter column price set not null,
    add check (price >= 0);
alter table public.user
    add constraint pk_user_id primary key (id),
    add unique (cart_id),
    add constraint fk_user_cart_id
    foreign key (cart_id)
    references public.cart(id) on delete cascade,
    alter column name set not null,
    alter column surname set not null,
    alter column email set not null,
    add unique (email),
    alter column password set not null,
    alter column role set not null;
alter table public.product
    add constraint pk_product_id primary key (id),
    alter column name set not null,
    alter column description set not null,
    alter column price set not null,
    add check (price > 0),
    alter column category set not null;
create index idx_product_name on public.product (name);
alter table public.cart_product
    add constraint pk_cart_product_id primary key (id),
    alter column cart_id set not null,
    alter column product_id set not null,
    add constraint fk_cart_product_cart_id
    foreign key (cart_id)
    references public.cart(id) on delete cascade,
    add constraint fk_cart_product_product_id
    foreign key (product_id)
    references public.product(id) on delete cascade,
    add constraint uc_cart_product unique (cart_id,product_id),
```

```
    alter column quantity set not null,
    add check (quantity > 0);
alter table public.shop
    add constraint pk_shop_id primary key (id),
    alter column seller_id set not null,
    alter column name set not null,
    alter column description set not null,
    alter column requisites set not null,
    alter column email set not null,
    add unique (email),
    add constraint fk_shop_seller_id
    foreign key (seller_id)
    references public.user(id) on delete cascade;
alter table public.shop_product
    add constraint pk_shop_product_id primary key (id),
    alter column shop_id set not null,
    add constraint fk_shop_product_shop_id
    foreign key (shop_id)
    references public.shop(id) on delete cascade,
    alter column product_id set not null,
    add constraint fk_shop_product_product_id
    foreign key (product_id)
    references public.product(id) on delete cascade,
    alter column quantity set not null,
    add check (quantity >= 0),
    add constraint uc_shop_product unique (shop_id,product_id);
alter table public.withdraw
    add constraint pk_withdraw_id primary key (id),
    alter column shop_id set not null,
    add constraint fk_withdraw_shop_id
    foreign key (shop_id)
    references public.shop(id) on delete cascade,
    alter column comment set not null,
    alter column sum set not null,
    add check (sum > 0),
    alter column status set not null;
alter table public.order_customer
    add constraint pk_order_customer_id primary key (id),
    alter column customer_id set not null,
    add constraint fk_order_customer_customer_id
    foreign key (customer_id)
```

```
    references public.user(id) on delete cascade,
    alter column address set not null,
    alter column created_at set not null,
    alter column total_price set not null,
    add check (total_price > 0),
    alter column payed set not null;
alter table public.order_shop
    add constraint pk_order_shop_id primary key (id),
    alter column shop_id set not null,
    add constraint fk_order_shop_shop_id
    foreign key (shop_id)
    references public.shop(id) on delete cascade,
    alter column order_customer_id set not null,
    add constraint fk_order_shop_order_customer_id
    foreign key (order_customer_id)
    references public.order_customer(id) on delete cascade,
    alter column status set not null,
    alter column notified set not null,
    add constraint uc_order_shop unique
       (shop_id,order_customer_id);
alter table public.order_shop_product
    add constraint pk_order_shop_product_id primary key (id),
    alter column order_shop_id set not null,
    add constraint fk_order_shop_product_order_shop_id
    foreign key (order_shop_id)
    references public.order_shop(id) on delete cascade,
    alter column product_id set not null,
    add constraint fk_order_shop_product_product_id
    foreign key (product_id)
    references public.product(id) on delete cascade,
    alter column quantity set not null,
    add check (quantity >= 0),
    add constraint uc_order_shop_product unique
       (order_shop_id,product_id);
```

3.4.3 Создание ролевой модели

В конструкторском разделе была описана ролевая модель. В листинге 3 представлен соответствующий код создания ролевой модели.

Листинг 3 – Скрипт создания ролевой модели базы данных

```
create role guest with
    noinherit
    login
    password 'guest';
grant select on table
    public.user,
    public.shop_product,
    public.cart,
    public.product to guest;
grant insert on table
    public.user,
    public.cart to guest;
create role customer with
    noinherit
    login
    password 'customer';
grant select on table
    public.user,
    public.shop_product,
    public.cart,
    public.cart_product,
    public.order_customer,
    public.product to customer;
grant insert on table
    public.cart_product,
    public.order_customer to customer;
grant update on table
    public.user,
    public.cart,
    public.cart_product to customer;
grant delete on table
    public.cart_product to customer;
create role seller with
    noinherit
    login
    password 'seller';
```

```
grant select on table
    public.user,
    public.shop_product,
    public.shop,
    public.order_shop,
    public.withdraw,
    public.product to seller;
grant update on table
    public.user,
    public.order_shop,
    public.shop_product to seller;
grant insert on table
    public.shop,
    public.withdraw,
    public.shop_product,
    public.product to seller;
create role moderator with
    noinherit
    login
    password 'moderator';
grant select on table
    public.user,
    public.shop_product,
    public.withdraw,
    public.product to moderator;
grant update on table
    public.withdraw,
    public.user to moderator;
grant delete on table
    public.shop_product to moderator;
```

3.4.4 Создание триггера

В конструкторском разделе был описан триггер базы данных. В листинге 4 представлен соответствующий код создания триггера.

Листинг 4 – Скрипт создания триггера базы данных

```
create or replace function add_cart_item()
returns trigger as $$
declare
product_price numeric;
begin
    if (TG_OP = 'DELETE') then
        select price from public.product where id=old.product_id
           into product_price;
        update public.cart set price=(price - product_price *
           old.quantity) where id=old.cart_id;
    elseif (TG_OP = 'INSERT') then
        select price from public.product where id=new.product_id
           into product_price;
        update public.cart set price=(price + product_price *
           new.quantity) where id=new.cart_id;
    elseif (TG_OP = 'UPDATE') then
        select price from public.product where id=new.product_id
           into product_price;
        update public.cart set price=(price + product_price *
           (new.quantity - old.quantity)) where id=new.cart_id;
    end if;
    return  new;
end;
$$ language plpgsql;
create trigger update_cart_price
after insert or update or delete on public.cart_product
for each row
execute function add_cart_item();
```

3.5 Интерфейс доступа к базе данных

В листинге 5 представлено описание интерфейса доступа к базе данных.

Листинг 5 – Интерфейс доступа к базе данных

```
--------------------------------
Меню (A - все; C - покупатель; S - продавец; M - модератор)
0  Закрыть приложение
1  Вывести меню
2  Войти (G)
3  Зарегистрироваться (G)
4  Выйти (M | C | S)
5  Получить информацию о пользователе (M | C | S)
6  Обновить информацию о пользователе (M | C | S)
7  Получить товар (G | M | C | S)
8  Получить список всех товаров (G | M | C | S)
9  Получить список товаров в магазине (G | M | C | S)
10 Получить список магазинов продавца (S)
11 Создать магазин (S)
12 Добавить товар в магазин (S)
13 Обновить товар в магазине (S)
14 Удалить товар из магазина (M)
15 Добавить товар в корзину (C)
16 Изменить количество товара в корзине (C)
17 Удалить товар из корзины (C)
18 Получить список товаров в корзине (C)
19 Создать заказ покупателя (C)
20 Получить список заказов покупателя (C)
21 Получить список заказов магазина (S)
22 Изменить статус заказа магазина (S)
23 Получить заявки на вывод средств (M)
24 Получить заявки на вывод средств для магазина (S)
25 Создать заявку на вывод средств (S)
26 Обработать заявку на вывод средств (M)
27 Оплатить заказ (C)
--------------------------------
Введите команду:
```

3.6 Тестирование

Компонент уровня доступа к базе данных был протестирован с использованием интеграционных тестов на основе библиотеки Testcontainers [22].

Testcontainers — это библиотека, которая предоставляет удобный API для запуска контейнеров Docker в рамках тестов. Она используется для интеграционного тестирования, где создаются изолированные среды с реальными сервисами, такими как базы данных, чтобы проверять взаимодействие приложения с ними. В рамках тестирования уровня доступа к базе данных были развернуты контейнеры с базой данных PostgreSQL, что позволило проверить корректность SQL-запросов, обработку транзакций и взаимодействие с реальной базой данных.

Процесс работы с Testcontainers заключается в следующем:

1) На этапе тестов запускается Docker-контейнер с необходимой базой данных.

2) Приложение подключается к этой базе данных, как если бы это была реальная база данных в рабочей среде.

3) Проводятся интеграционные тесты, в которых проверяется, как приложение взаимодействует с базой данных.

4) По завершении тестов контейнер автоматически удаляется, обеспечивая чистую среду для следующих тестов.

Такой подход позволяет создавать реалистичные сценарии тестирования, имитирующие рабочие условия, что значительно повышает точность тестов и позволяет выявлять потенциальные проблемы на ранних этапах разработки. В листинге 6 представлены результаты тестирования компонента уровня доступа к базе данных.

Листинг 6 – Результаты тестирования

```
PASS
ok      github.com/.../postgres/test     60.184s
```

Вывод

В данном разделе были выбораны и описаны средства реализации, были описаны архитектура приложения и детали реализации, были представлены описание интерфейса доступа к базе данных, а также метод тестирования компонента уровня доступа к базе данных.

4 Исследовательский раздел

В данном разделе будет приведено описание исследования зависимости времени выполнения запросов от их количества в секунду и представлены его результаты.

4.1 Цель и описание исследования

Целью является проведение исследования зависимости времени выполнения запросов от их количества в секунду.

Для электронной торговой площадки очень важно выдерживать высокую нагрузку для методов получения товаров, потому что в условиях высокой конкуренции на рынке электронной торговли, скорость и надежность доступа к информации о товарах могут стать решающим фактором для привлечения и удержания клиентов. Площадки, которые предоставляют информацию быстро и эффективно, имеют преимущество перед конкурентами.

По этой причине исследование проводилось на запросе получения товара по его имени с использованием индекса и без при 10000 строк в таблице товаров.

Исследование проводилось с использованием Locust [23]. Locust — это инструмент для проведения нагрузочного тестирования, который позволяет моделировать большое количество пользователей, взаимодействующих с системой одновременно. Инструмент позволяет создавать сценарии тестирования на Python [24] и предоставляет веб-интерфейс для мониторинга и анализа результатов тестирования в реальном времени.

На рисунках 6 — 7 представлены графики нагрузочного тестрирования в реальном времени.

Рисунок 6 – графики нагрузочного тестрирования без индекса в реальном времени

Рисунок 7 – графики нагрузочного тестрирования с индексом в реальном времени

4.2 Результаты исследования

Был выполнен сравнительный анализ времени выполнения запросов от их количества в секунду с использованием индекса и без.

Тестирование проводилось в течение 20 секунд с увеличением количества пользователей до 2000 с шагом 100 в секунду.

Значения в правых столбцах таблиц представляют собой усреднённые результаты из общего числа запросов, зафиксированных на момент, когда интенсивность запросов достигала значений, указанных в левой колонке. Например, в таблице 23 среднее время ответа с индеком 112 миллисекунд получено на основе 56 запросов, а время 116 миллисекунд — на основе 173 запросов.

В таблице 23 приведены результаты измерений среднего времени ответа от количества запросов.

Таблица 23 – Зависимость времени среднего времени ответа от количества запросов в секунду с индексом и без

Число запросов в секунду	**Среднее время ответа (мс)**	
	Без индекса	**С индексом**
56	202	112
117	205	116
179	208	120
240	214	126
302	217	132
376	222	141
451	237	147
525	243	151
600	253	159
675	259	162
750	265	169
825	280	173
900	297	179
985	313	187
1070	338	197
1155	378	210
1240	423	233

На рисунке 8 приведен график зависимости.

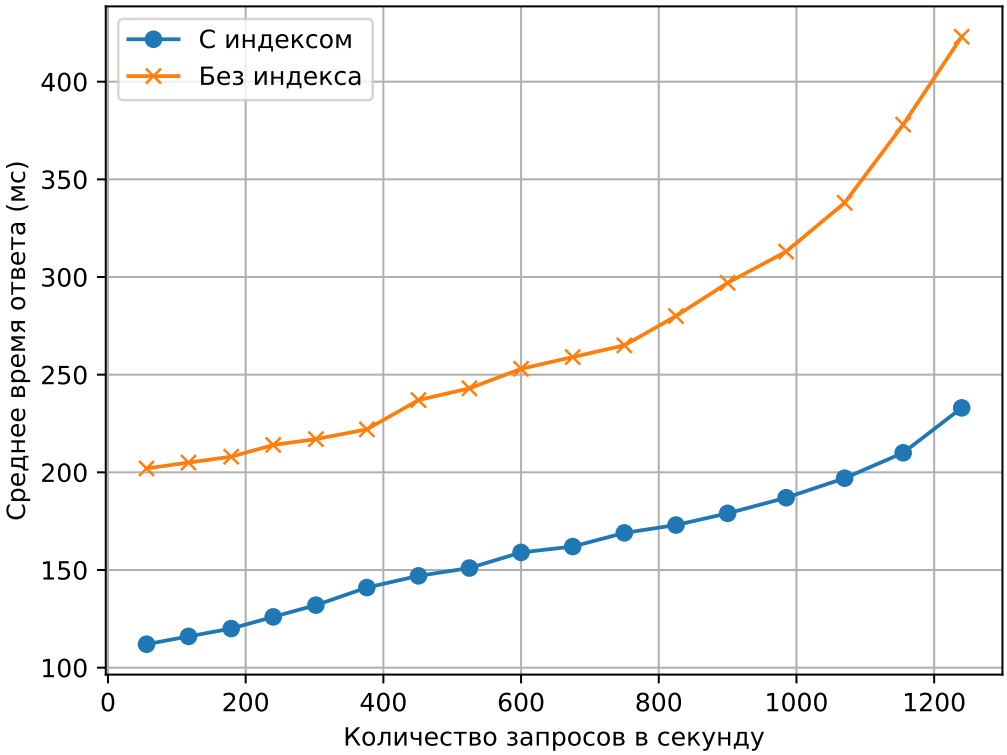

Рисунок 8 – График зависимости времени среднего времени ответа от количества запросов в секунду с индексом и без

По графику 8 видно, что по мере увеличения количества запросов в секунду, время ответа увеличивается как для запросов с индексом, так и без индекса. Также время ответа без индекса стабильно выше чем с индексом в среднем в 1.81 раза. Время ответа без индекса растет более стремительно. Таким образом, при увеличении числа запросов производительность системы без индекса ухудшается гораздо быстрее.

В таблице 24 приведены результаты измерений 95-го процентиля времени ответа от количества запросов в секунду.

Таблица 24 – Зависимость 95-го процентиля времени ответа от количества запросов в секунду с индексом и без

Число запросов в секунду	**Среднее время ответа (мс)**	
	Без индекса	**С индексом**
56	278	452
117	292	453
179	312	459
240	340	460
302	360	469
376	391	470
451	436	474
525	471	480
600	502	500
675	551	510
750	589	518
825	645	527
900	692	552
985	733	564
1070	778	572
1155	841	589
1240	902	609

На рисунке 9 приведен график зависимости.

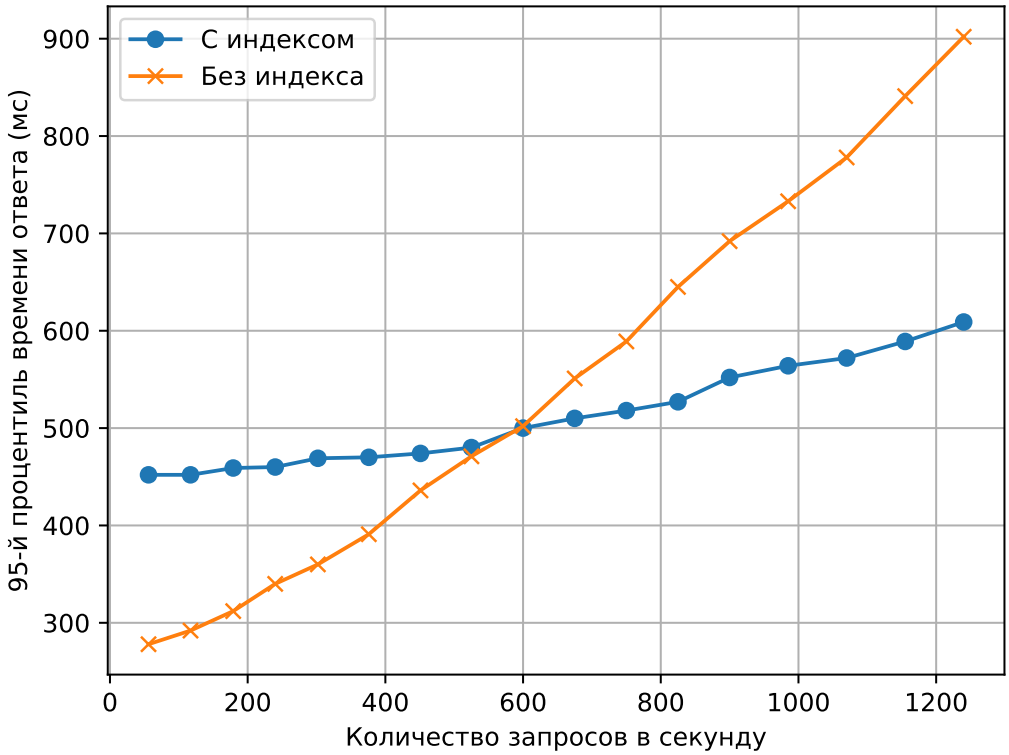

Рисунок 9 – График зависимости 95-го процентиля времени ответа от количества запросов в секунду с индексом и без

По графику 9 видно, что при 1240 запросах в секунду 95 процентов запросов к приложению при использовании индекса обрабатываются примерно в 1.48 раза быстрее. При этом 95-го процентиль времени ответа с использованием индекса больше чем без использования индекса вплоть до 600 запросов в секунду, но 95-й процентиль времени ответа без индекса растет более стремительно. Таким образом, система без использования индекса становится менее эффективной при высокой нагрузке.

Вывод

В данном разделе было приведено описание исследования зависимости времени выполнения запросов от их количества в секунду. Среднее время выполнения запросов увеличивается с ростом количества запросов в секунду.

Использование индекса позволяет ускорить среднее время ответа. Так при 1240 запросах в секунду использование индекса позволило ускорить среднее время выполнения запросов более чем в 1.81 раза.

ЗАКЛЮЧЕНИЕ

Цель данной работы была достигнута, а именно, была разработана база данных для электронной торговой площадки.

Для достижения поставленной цели, были решены все задачи:

— проведен анализ предметной области электронных торговых площадок;

— формализованы требования к разрабатываемой базе данных и приложению;

— спроектированы сущности базы данных, ролевая модель и ограничения целостности;

— выбраны средства реализации базы данных и приложения;

— разработано программное обеспечение, обеспечивающее интерфейс для доступа к базе данных.

— проведено исследоване зависимости времени выполнения запросов от их количества в секунду.

Разработанное программное обеспечение можно усовершенствовать, добавив кеширование и расширив сценарии использования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Elmasri R.*, *Navathe S. B.* Fundamentals of Database Systems. — 7th. — Pearson, 2016.
2. *Гаврилова Ю.* Конспект лекций по курсу «Базы Данных». — 2023.
3. *Baeza-Yates R.*, *Ribeiro-Neto B.* Modern Information Retrieval: The Concepts and Technology behind Search. — Addison-Wesley, 1999.
4. *Fowler M.* NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. — Addison-Wesley, 2012.
5. *Group P. G. D.* PostgreSQL Documentation. — Режим доступа: `https://www.postgresql.org/docs/` (дата обращения: 27.8.2024).
6. *SQLite Foundation.* SQLite Documentation. — Режим доступа: `https://www.sqlite.org/docs.html` (дата обращения: 27.8.2024).
7. *Oracle Corporation.* MySQL Documentation. — Режим доступа: `https://dev.mysql.com/doc/` (дата обращения: 27.8.2024).
8. *Oracle Corporation.* Oracle Database Documentation. — Режим доступа: `https://docs.oracle.com/en/database/` (дата обращения: 27.8.2024).
9. *Authors T. G.* The Go Programming Language Documentation. — 2024. — Режим доступа: `https://golang.org/doc/` (дата обращения: 27.8.2024).
10. *Authors T. G.* Database/sql Package Documentation. — 2024. — Режим доступа: `https://pkg.go.dev/database/sql` (дата обращения: 27.8.2024).
11. *Mattn.* Sqlx Documentation. — 2024. — Режим доступа: `https://pkg.go.dev/github.com/jmoiron/sqlx` (дата обращения: 27.8.2024).
12. *Mulesoft I.* What is an API? - Definition and Examples. — 2024. — Режим доступа: `https://www.mulesoft.com/resources/api/what-is-an-api` (дата обращения: 27.8.2024).
13. *Authors G.* Gin Framework Documentation. — 2024. — Режим доступа: `https://gin-gonic.com/docs/` (дата обращения: 27.8.2024).
14. *Swagger.* Swagger Documentation. — 2024. — Режим доступа: `https://swagger.io/docs/` (дата обращения: 27.8.2024).

15. *OAuth2*. Bearer Tokens. — 2024. — Режим доступа: https://oauth.net/2/bearer-tokens/ (дата обращения: 27.8.2024).

16. *Redis*. Redis Documentation. — 2024. — Режим доступа: https://redis.io/documentation (дата обращения: 27.8.2024).

17. *MinIO I.* MinIO Documentation. — 2024. — Режим доступа: https://docs.min.io/ (дата обращения: 27.8.2024).

18. *Docker I.* Docker Documentation. — 2024. — Режим доступа: https://docs.docker.com/ (дата обращения: 27.8.2024).

19. *Porras J.* Hexagonal Architecture with Golang. — 2021. — Режим доступа: https://dev.to/juliandavidmr/hexagonal-architecture-with-golang-4488 (дата обращения: 27.8.2024).

20. *Richardson C.* Transactional Outbox Pattern. — 2018. — Режим доступа: https://microservices.io/patterns/data/transactional-outbox.html (дата обращения: 27.8.2024).

21. *(IETF) I. E. T. F.* Hypertext Transfer Protocol – HTTP/1.1. — 1999. — Режим доступа: https://tools.ietf.org/html/rfc2616 (дата обращения: 27.8.2024).

22. *Testcontainers*. Testcontainers Documentation. — 2024. — Режим доступа: https://www.testcontainers.org/ (дата обращения: 27.8.2024).

23. Locust Documentation. — Режим доступа: https://docs.locust.io (дата обращения: 27.8.2024).

24. Python Documentation. — Режим доступа: https://docs.python.org (дата обращения: 27.8.2024).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе

Презентация состоит из 12 слайдов.