1 Аналитический раздел

1.1 Анализ предметной области

Компьютерный магазин предлагает разнообразный ассортимент продукции, включая компьютеры, комплектующие, периферийные устройства, программное обеспечение и аксессуары. Для эффективного управления ассортиментом необходимо учитывать такие характеристики товаров, как название, описание, производитель, цена, количество на складе и уникальный идентификатор товара. [2]

Одной из ключевых задач является поддержание актуальной информации о количестве товаров на складе.

Система должна поддерживать функции создания, редактирования и отслеживания заказов. Это включает информацию о клиентах, деталях заказа (список товаров, количество, стоимость) и статусах заказов (оформлен, обработан, доставлен и т.д.). Кроме того, важно отслеживать статусы заказов, такие как "оформлен", "обработан", "в пути"и "доставлен" чтобы обеспечить своевременную и точную обработку каждого заказа. [3]

Для повышения качества обслуживания необходимо вести учет данных о клиентах, таких как имя, контактная информация, история покупок. Это поможет персонализировать предложения и улучшить клиентский сервис. [3]

1.2 Постановка задачи

Разработка удобного программного обеспечения для управления данными компьютерного магазина. Пользователи могут просматривать товары, размещать заказы и отслеживать информацию о заказе, а владельцы магазинов — управлять данными о товарах и заказах.

На рисунке 1.1 приведена IDEF0-схема для поставленной задачи.

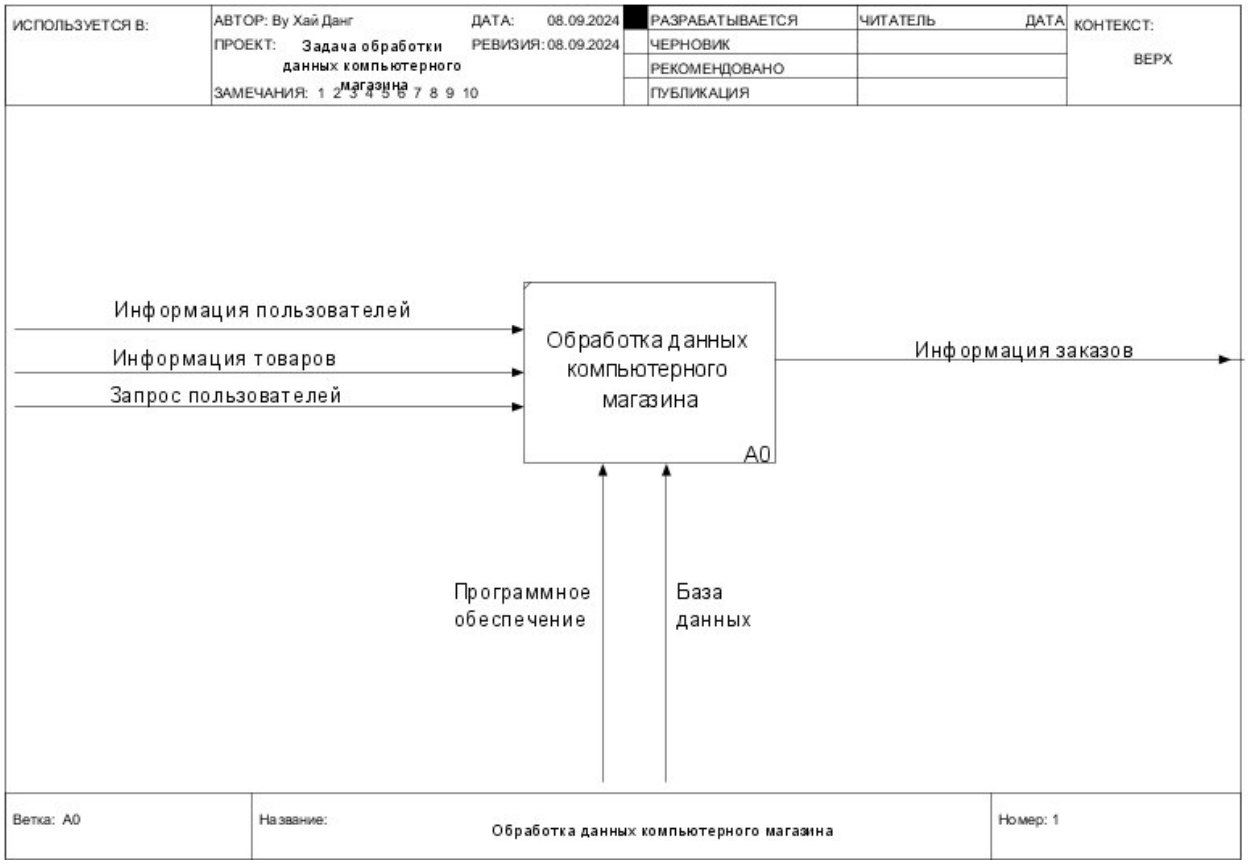

Рисунок 1.1 – Функциональная модель поставленной задачи

1.3 Формализация данных

В базе данных хранится информация о:

— пользователях;

— товарах;

— заказах;

— корзинах

— деталях корзины

— деталях заказы

— промокодах

В таблице 1.1 приведены информации о каждой сущности.

Таблица 1.1 – Категории и сведения о данных

Категория	**Сведения**
Пользователь	ID, имя, номер телефона, адрес, почта, логин, пароль, права доступа.
Товар	ID, название, цена, количество, производитель, описание.
Промокод	ID, код, акция, дата начала, дата конца.
Корзина	ID, дата создания, ID пользователя.
Деталь корзины	ID, ID товары, ID корзины, количество.
Заказ	ID, ID пользователя, ID промокода, статус, дата создания.
Деталь заказа	ID, ID пользователя, ID заказа, количество.

На рисунке 1.2 отображена ER-диаграмма системы, основанная на приведенной выше таблице.

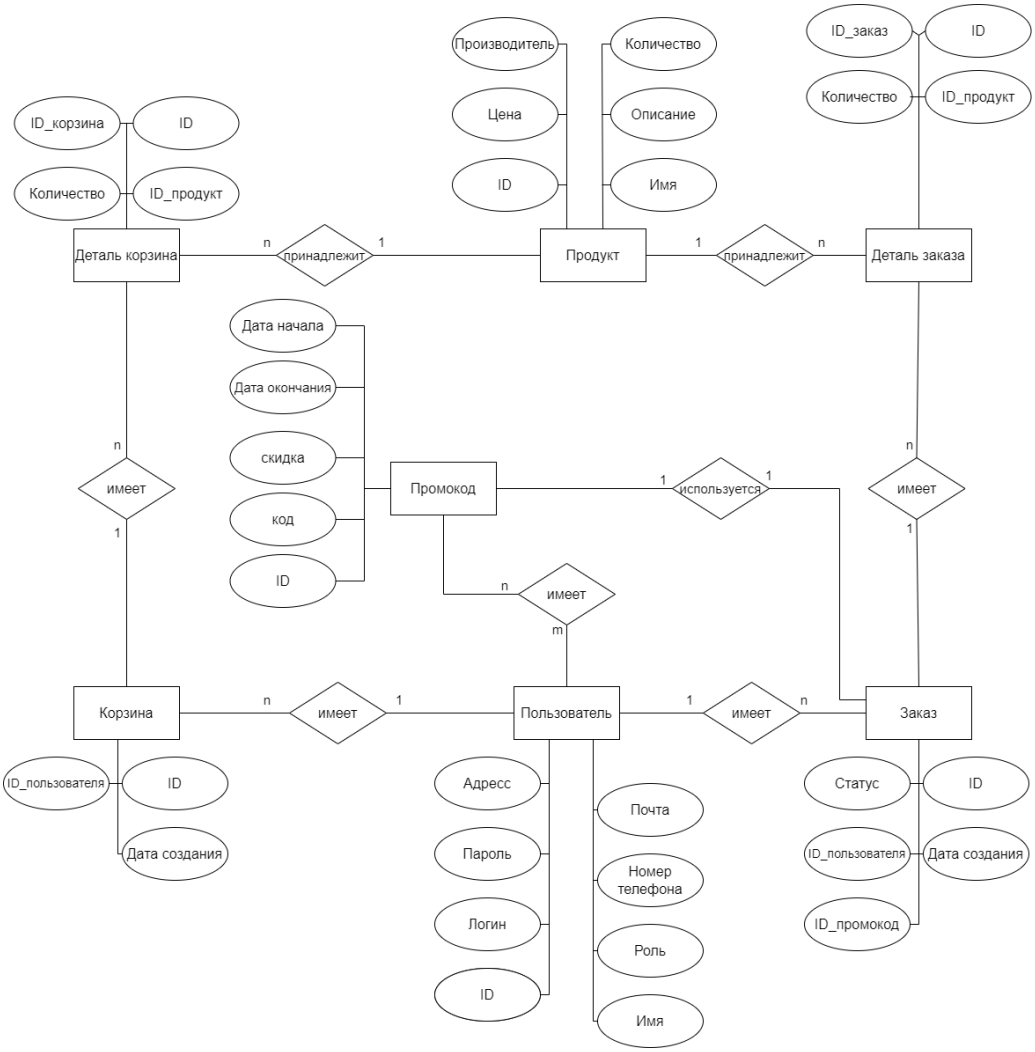

Рисунок 1.2 – ER-диаграмма

1.4 Системные пользователи

Для взаимодействия с программным обеспечением было выделено четыре вида ролей: гость, администатор, поставщик, клиент.

Ниже, в таблице 1.2 приведены роли пользователей и их набор возможностей. На рисунке 1.3 представлена Use-Case диаграмма пользователей.

Таблица 1.2 – Типы пользователей и их набор возможностей

Тип пользователя	**Возможности**
Гость	Авторизация.
Клиент	Авторизация. Редактирование своей информации. Просмотр товаров. Добавление товаров в корзину. Заказ. Просмотр истории своих заказов. Выход.
Поставщик	Авторизация. Редактирование своей информации. Редактирование товаров и заказов. Выход.
Администратор	Авторизация. Редактирование своей информации. Редактирование информации об аккаунтах других пользователей. Редактирование товаров и заказов. Выход.

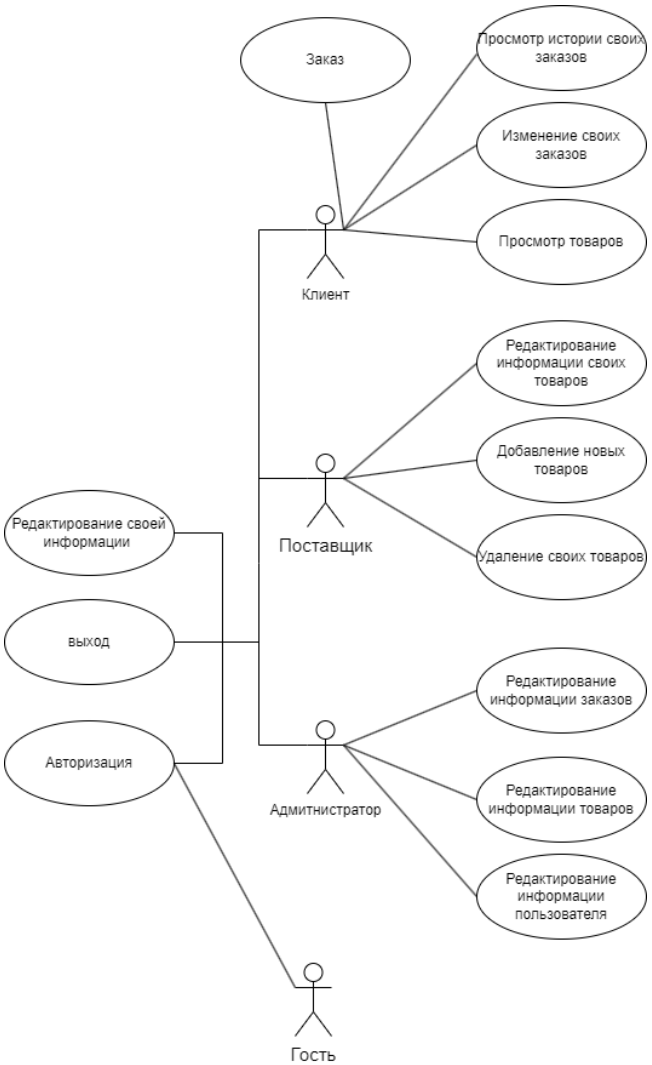

Рисунок 1.3 – Use-Case диаграмма

1.5 Модель баз данных

Модель данных — это совокупность допустимых структур данных и операций над ними, поддерживаемая компьютерной средой (в т. ч. СУБД), для определения логической структуры базы данных и динамического моделирования состояний предметной области. [4]

Модели данных действительно разделятся на три основных вида: дореляционные, реляционные и постреляционные. Каждый из этих видов моделей данных характеризуется своими особенностями, архитектурой и областями применения.

1.5.1 Дореляционные модели данных

Иерархическая база данных организована в виде множества деревьев, где каждый узел дерева представляет собой запись, содержащую именованные поля, соответствующие атрибутам объектов предметной области. В такой структуре данные упорядочены по строгой иерархии: каждая запись-«потомок» может иметь только одну запись-«родителя», исключая возможность наличия нескольких предков. Этот подход используется для представления отношений, когда данные имеют четкую и логичную иерархическую структуру. [5]

На рисунке 1.4 представлена иерархическая модель данных в базе данных интернет–провайдера.

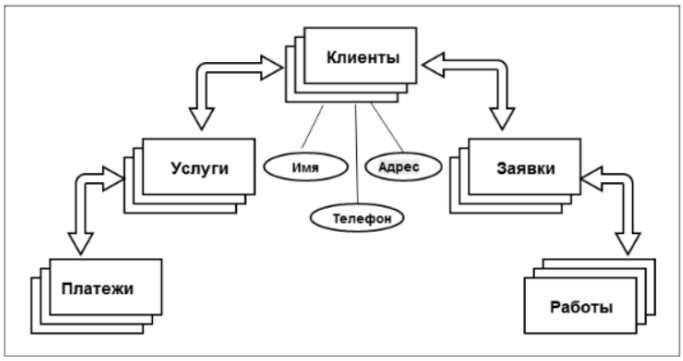

Рисунок 1.4 – Фрагмент иерархической модели данных

Сетевая модель данных организована в виде графа, где записи представляют собой узлы, соединенные между собой множественными отношениями. В отличие от иерархической модели, узел в сетевой базе данных может иметь несколько родительских и несколько дочерних узлов, что позволяет более гибко моделировать сложные взаимосвязи между данными. Эта структура особенно полезна для представления и управления данными, которые не вписываются в жесткую иерархическую структуру, позволяя создавать более сложные и разветвленные сети взаимосвязей. [5]

На рисунке 1.5 представлен фрагмент сетевой модели данных в базе данных интернет–провайдера.

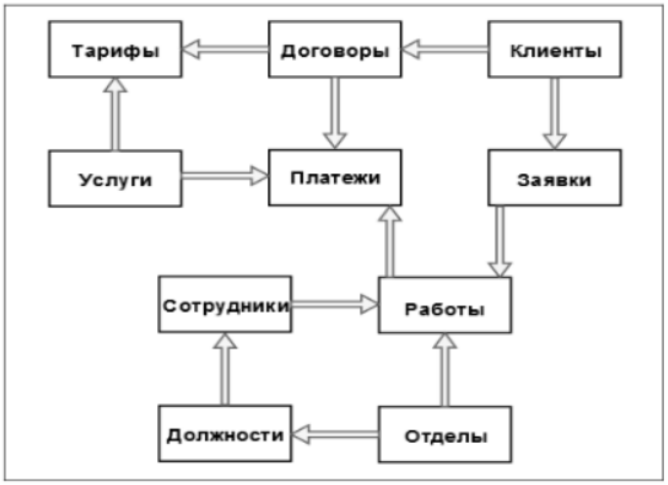

Рисунок 1.5 – Фрагмент иерархической модели данных

1.5.2 Реляционные модели данных

Реляционная база данных представляет собой набор взаимосвязанных таблиц, каждая из которых имеет уникальное имя. Таблицы отображают данные о реальных объектах или «сущностях» предметной области и состоят из строк, называемых кортежами, и столбцов, называемых атрибутами. Каждый кортеж представляет собой экземпляр сущности и описывается набором значений, соответствующих его атрибутам. Для установления связей между таблицами используется механизм внешних ключей: эти ключи содержат ссылки на соответствующие атрибуты других таблиц, обеспечивая целостность данных и возможность выполнения сложных запросов. [5]

На рисунке 1.6 представлен пример реляционной модели данных

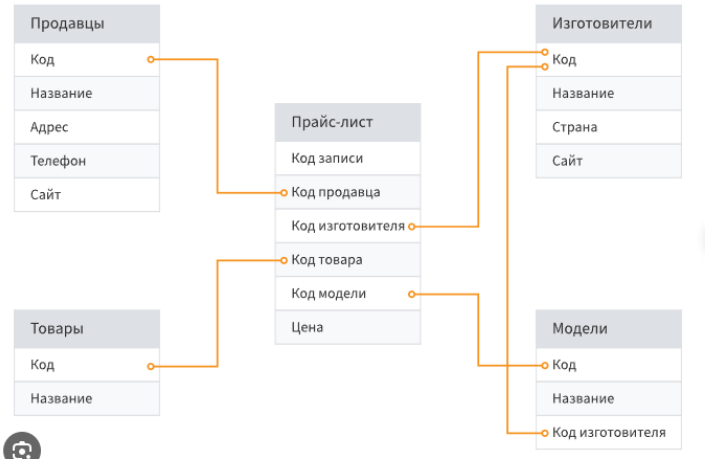

Рисунок 1.6 – Пример реляционных моделей данных

1.5.3 Постреляционные модели данных

Современный этап развития информационных систем включает использование постреляционных моделей данных, которые сочетают преимущества реляционных и объектно-ориентированных подходов. Эти модели поддерживают как объектные, так и объектно-реляционные структуры данных, обеспечивая разработчикам возможность использовать современные языки программирования, такие как C++, Java, Perl и Python. Это дает постреляционным системам управления базами данных (СУБД) конкурентные преимущества, позволяя им более эффективно обрабатывать сложные данные и улучшать производительность. [5]

1.5.4 Выбор модели данных

Реляционная модель данных была выбрана из-за её зрелости и надежности, что делает её стандартом для многих критически важных приложений. Она обеспечивает строгую целостность данных через механизмы ограничений и поддержку транзакций, что помогает избежать несоответствий и потери данных. Кроме того, реляционная модель позволяет обеспечить независимость данных от приложения, что упрощает модификацию структуры данных и делает систему более адаптируемой к изменяющимся требованиям бизнеса. [6]

Вывод

В данном разделе была проведена формализация задачи и данных, рассмотрены типы пользователей и требуемые функционалы. Также был проведен анализ существующих моделей баз данных и было решено использовать в данной работе реляционную СУБД.

2 Конструкторский раздел

2.1 Проектирование базы данных

2.1.1 Таблицы базы данных

В соответствии с ER-диаграммой системы, изображенной на рисунке 1.2, база данных приложения хранит следующие таблицы:

1. Таблица пользователей — UserDB;
2. Таблица промокодов — PromoDB;
3. Таблица товаров — ProductDB;
4. Таблица заказов — OrderDb;
5. Таблица деталей заказов — ItemOrderDB;
6. Таблица корзины — CartDB;
7. Таблица деталей корзины — ItemCartDB;
8. Таблица связи многие к многим — UserPromoDB;

На рисунке 2.1 представлена диаграмма разрабатываемой базы данных.

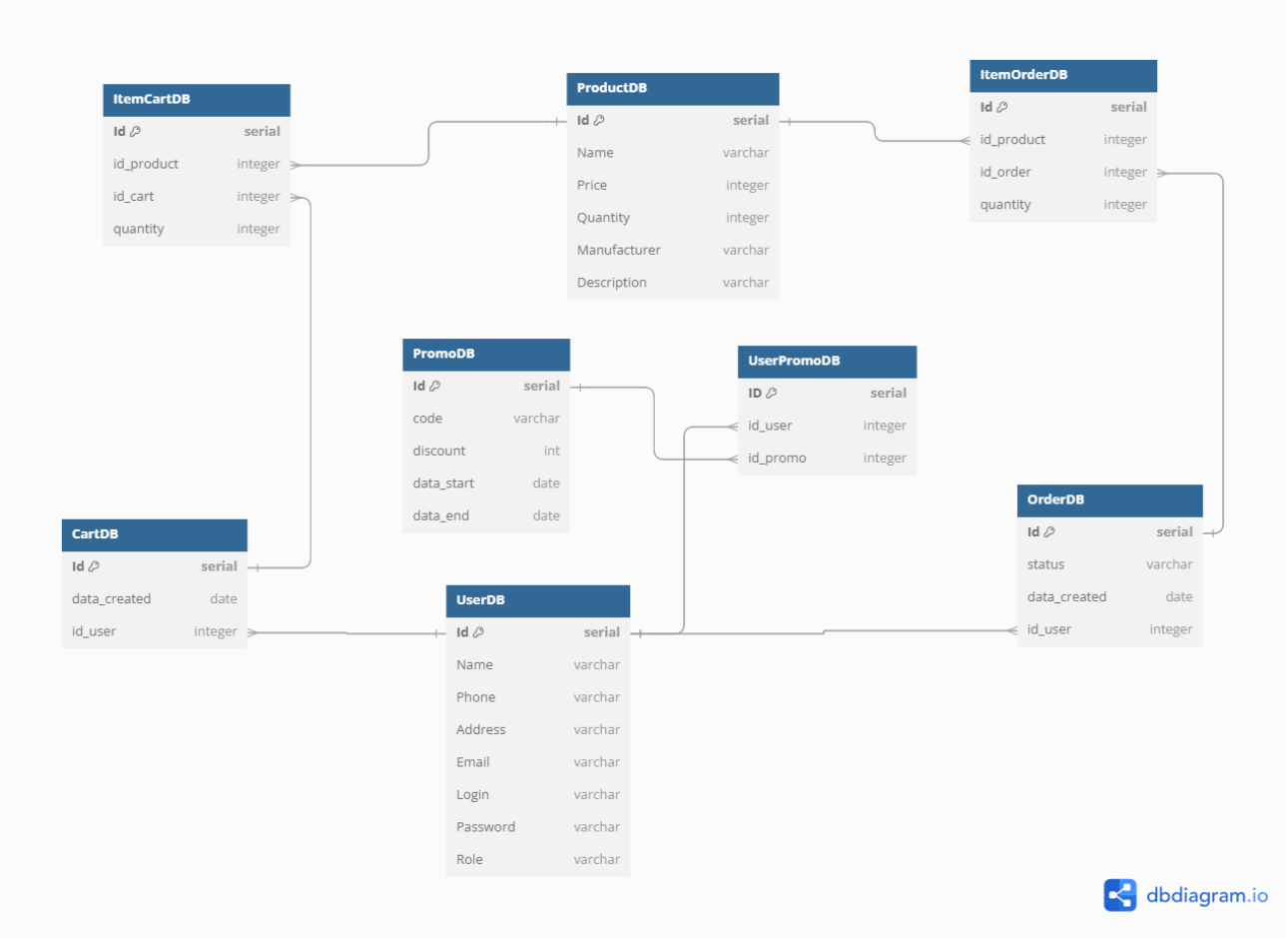

Рисунок 2.1 – Диаграмма базы данных

На основе диаграммы сущностей-связей, приведенной на рисунке 2.1, определяются стуктуры столбцов, их типы и органичения.

Таблица 2.1 – Сведение о таблице userdb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
name	VARCHAR(50)	NOT NULL	имя
phone	VARCHAR(50)	NOT NULL	Номер телефона
address	VARCHAR(50)	NOT NULL	Адрес
email	VARCHAR(50)	NOT NULL	Почта
login	VARCHAR(50)	NOT NULL	Логин
password	VARCHAR(50)	NOT NULL	Пароль
role	VARCHAR(50)	NOT NULL	Права доступа

Таблица 2.2 – Сведение о таблице promodb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
code	VARCHAR(50)	NOT NULL	Код
discount	INT	NOT NULL	Акция
data_start	DATE	NOT NULL	Дата начала
data_end	DATE	NOT NULL	Дата конца

Таблица 2.3 – Сведение о таблице productdb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
name	VARCHAR(50)	NOT NULL	Название
price	INT	NOT NULL	Цена
quantity	VARCHAR(50)	NOT NULL	Количество на складе
manufacturer	VARCHAR(50)	NOT NULL	Производитель
description	VARCHAR(50)	NOT NULL	Описание

Таблица 2.4 – Сведение о таблице Cartdb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
data_created	DATE	NOT NULL	Дата создания
id_user	INT	FK	Идентификатор пользователя

Таблица 2.5 – Сведение о таблице ItemCartdb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
id_product	VARCHAR(50)	FK	Идентификатор товары
id_cart	INT	FK	Идентификатор корзины
Quantity	INT	NOT NULL	Количество

Таблица 2.6 – Сведение о таблице Orderdb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
status	VARCHAR(50)	NOT NULL	Дата создания
data_created	DATE	NOT NULL	Дата создания
id_user	INT	FK	Идентификатор пользователя
id_promo	INT	FK	Идентификатор промокода

Таблица 2.7 – Сведение о таблице ItemOrderdb

Столбец	Тип данных	Ограничения	Значение
id	serial	PK	Идентификатор
id_product	INT	FK	Идентификатор товары
id_order	INT	FK	Идентификатор заказа
quantity	INT	NOT NULL	Количество

2.1.2 Ролевая модель

Создание ролей для каждой группы пользователей в базе данных

1. Гость имеет право добавления в таблицу userdb;
2. Клиент имеет право просмотра всех таблиц, добавления в таблицы cartdb, itemcartdb, orderdb, itemorderdb, удаления в таблицах cartdb, itemcartdb, изменения в таблице users;
3. Поставщик имеет право просмотра, добавления, удаления всех таблиц, кроме таблицы userdb, cartdb, itemcartdb, promodb;
4. Администратор имеет право просмора всех таблиц, добавления, изменения, удаления из всех таблиц.

2.2 Триггеры

Триггеры — это особые команды в коде языка SQL, которые реагируют и запускаются только при определенных событиях.[7]

Когда клиент размещает заказ, количество товара на складе автоматически уменьшается на количество товара в заказе.

Ниже, на рисунке 2.2, представлена схема вышеуказанного триггера.

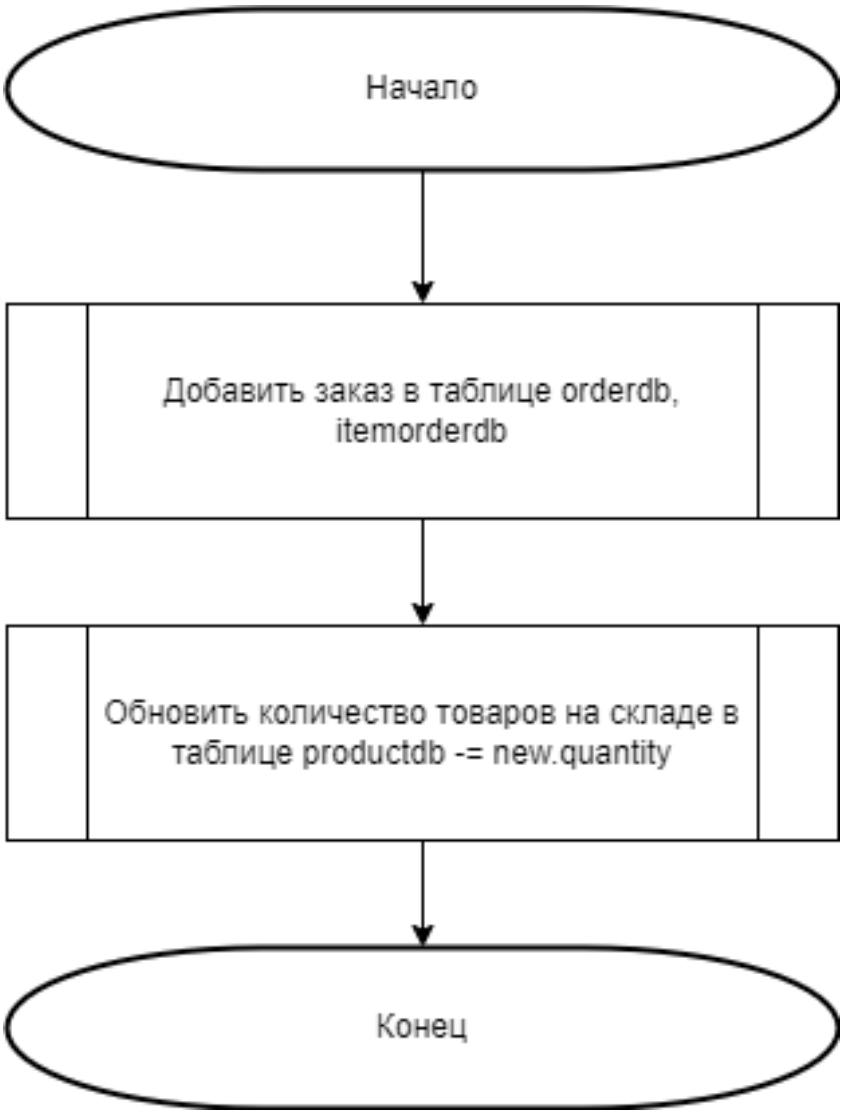

Рисунок 2.2 – Схема тригерра

2.3 Декомпозиция разрабатываемого программного обеспечения

На рисунке 2.3, представлена схема UML-диаграмма классов приложения.

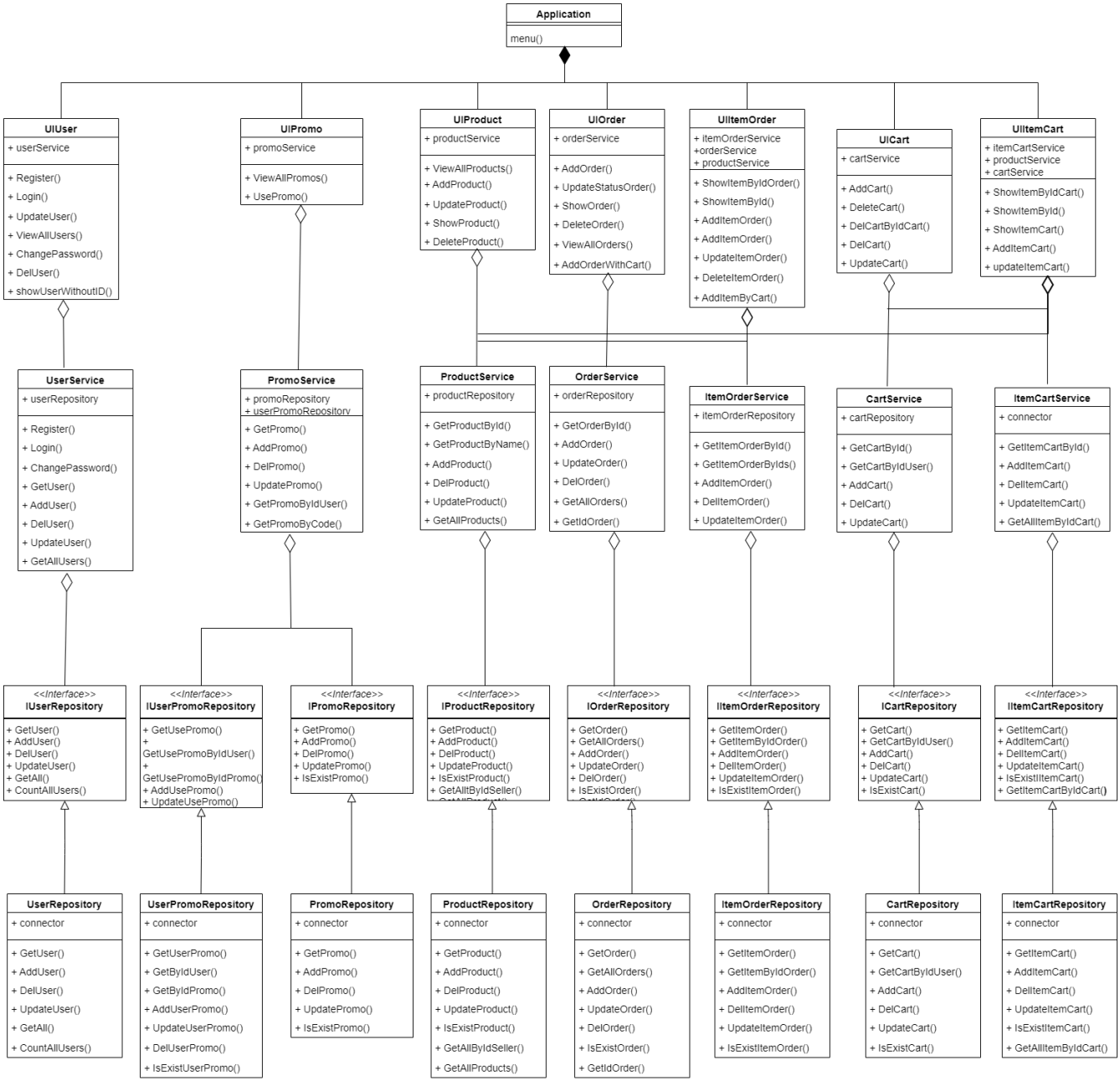

Рисунок 2.3 – Диаграмма разработанного программного обеспечения

В программе реализованы следующие классы:

— class UserRepository, IUserRepository - это класс и интерфейс класса компонента для доступа к данным для сущности пользователей;

— class PromoRepository, IPromoRepository - это класс и интерфейс класса компонента для доступа к данным для сущности промокодов;

— class ProductRepository, IProductRepository – это класс и интерфейс класса компонента для доступа к данным для сущности товаров;

— class OrderRepository, IOrderRepository- это класс и интерфейс класса компонента для доступа к данным для сущности заказов;

— class ItemOrderRepository, IItemOrderRepository - это класс и интерфейс класса компонента для доступа к данным для сущности деталей заказа;

— class CartReposiotory, ICartReposiotory - это класс и интерфейс класса компонента для доступа к данным для сущности корзин;

— class ItemCartRepository, IItemCartRepository - это класс и интерфейс класса компонента для доступа к данным для сущности деталей корзины;

— class UserService, UserPromoService, PromoService, ProductService, OrderService, ItemOrderService, CartService, ItemCartService: – эти классы компонента бизнес-логики соответствующий сущностей: пользовотель, промокод, товар, заказ, деталь заказа, корзина, деталь корзины;

— class UIUser, UIUserPromo, UIPromo, UIProduct, UIOrder, UIItemOrder, UICart, UIItemCart – эти классы графического пользовательского интерфейса соответствующий сущностей: пользовотель, промокод, товар, заказ, деталь заказа, корзина, деталь корзины.

Вывод

В этом разделе спроектирована база данных и приложение для доступа к ней. Был спроектирован триггер, осуществляющие автоматически пересчитывать количество товаров на складе при добавления заказов.

3 Технологический раздел

В данном разделе представлены средства разработки программного обеспечения, выбор языка программирования и описан интерфейс программы. Также будут рассмотрены примеры работы программы.

3.1 Выбор языка программирования

В качестве языка программирования был выбран C# в силу следующих причин:

— В стандартной библиотеке языка присутствует поддержка всех структур данных, выбранных по результатам проектирования;

— C# обладает обширной экосистемой библиотек, которые упрощают работу с базой данных PostgreSQL, обеспечивая удобное подключение, выполнение запросов и управление данными.;

3.2 Выбор среды разработки

В качестве среды разработки была выбрана Microsoft Visual Studio 2022 как версия единственной на данный момент среды, полностью поддерживающей последний стандарт языка C#

3.3 Выбор СУБД

PostgreSQL была выбрана в качестве СУБД благодаря своей бесплатной и открытой природе, что делает её доступным и популярным решением для управления данными. Она поддерживает как структурированные, так и неструктурированные данные, обеспечивая универсальность и гибкость в работе с различными типами информации. Совместимость с основными платформами, включая Linux, и отличная производительность делают PostgreSQL идеальным выбором для различных сред, будь то виртуальные, физические

или облачные. Кроме того, PostgreSQL предоставляет мощные инструменты для импорта данных и встроенные функции для повышения безопасности, такие как поддержка DBMS_SESSION, что делает её надёжным и безопасным решением для работы с большими объёмами данных. [8]

3.4 Создание объектов баз данных

3.4.1 Создание таблиц сущностей

Ниже, на листинге 3.1 - 3.7, представлено создание всех таблиц.

Листинг 3.1 – Создание таблицы UserDB

```
create table UserDB(
  Id serial primary key,
  Name varchar(50) not null,
  Phone varchar(50) not null,
  Address varchar(50) not null,
  Email varchar(50) not null,
  Login varchar(50) unique,
  Password varchar(50) not null,
  Role varchar(50) not null check (role in ('admin', 'seller', 'client'))
);
```

Листинг 3.2 – Создание таблицы PromoDB

```
create table PromoDB(
  Id serial primary key,
  code varchar(20) unique,
  discount int not null,
  data_start date not null,
  data_end date not null
);
```

Листинг 3.3 – Создание таблицы ProductDB

```
create table ProductDB (
  Id serial primary key,
  Name varchar(50) not null,
  Price int not null,
  Quantity int not null,
  Manufacturer varchar(50) not null,
  Description varchar(50) not null
);
```

Листинг 3.4 – Создание таблицы CartDB

```
create table CartDB(
  Id serial primary key,
  data_created date not null,
  id_user int references UserDB(Id) ON DELETE CASCADE
);
```

Листинг 3.5 – Создание таблицы ItemCartDB

```
create table ItemCartDB(
  Id serial primary key,
  id_product int,
  id_cart int,
  quantity int not null,
  foreign key (id_cart) references CartDB(Id) ON DELETE CASCADE,
  foreign key (id_product) references ProductDB(Id) ON DELETE CASCADE
);
```

Листинг 3.6 – Создание таблицы OrderDB

```
create table orderDB(
  Id serial primary key,
  status varchar(20) not null,
  data_created date not null,
  id_user int not null,
  id_promo int references PromoDB(Id) ON DELETE CASCADE,
  foreign key (id_user) references UserDB(Id) ON DELETE CASCADE
);
```

Листинг 3.7 – Создание таблицы ItemOrderDB

```
create table ItemOrderDB(
  Id serial primary key,
  id_product int not null,
  id_order int not null,
  quantity int  not null,
  foreign key (id_order) references OrderDB(Id) ON DELETE CASCADE,
  foreign key (id_product) references ProductDB(Id) ON DELETE CASCADE
);
```

3.4.2 Создание тригерра

На листинге 4.2 в приложении А представлено создание триггера в соответствии с рисунком 2.2.

3.5 Реализация программы

На листингах 4.8 - 4.9 в приложении А представлены основные функции программы.

3.6 Интерфейс программы

Проект спроектирован через Desktop-приложение. На рисунках 4.2 – 4.12 в приложении А отображён интерфейс приложения. Эти изображения соответствуют каждой роли пользователя и их операции.

3.7 Тестирование триггера

Проводится тестирование триггера, который автоматически обновляет количество товаров на складе «quantity» в таблице productdb когда клиент делает заказ.

На листинге 4.7 в приложении А представлена функция тестирования триггера.

На рисунке 3.1 приведены результаты тестирования.

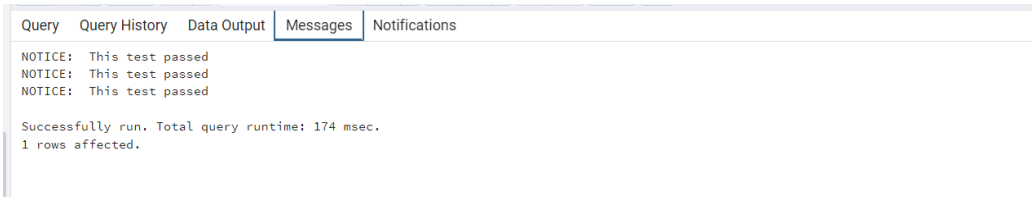

Рисунок 3.1 – Результаты тестов для триггера after_itemorder_insert.

Вывод

В данном разделе представлены средства разработки программного обеспечения, выбор языка программирования и описан интерфейс программы. Также рассмотрены примеры работы программы.

4 Исследовательский раздел

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось исследование:

— операционная система Windows 10 Домашняя 21H2;

— оперативная память 12 Гб;

— процессор Intel(R) Core(TM) i7-9750H CPU @ 2.6ГГц.

Во время проведения замеров времени устройство, помимо исследуемой программы, было нагружено только операционной системой и встроенными приложениями и он было включено в сеть электропитания.

4.2 Описание эксперимента и результаты исследования

Цель эксперимента заключается в выявлении зависимости времени обработки запросов к таблицам базы данных от использования индексирования. Эксперимент проводился на таблице ProductDB, содержащих информацию о товарах. Чтобы получить достаточно точное значение, производилось усреднение времени. Количество запусков замера времени для каждого случая — 10 раз.

Таблица 4.1 – Замеры времени обработки запросов в зависимости от количеств строк в таблице (с использованием индексирования и без использования индексирования)

Количество строк	Без индекса, мс	С индексом, мс
1000	0.117	0.021
1500	0.124	0.026
2000	0.145	0.028
2500	0.192	0.033

На рисунке 4.1 приведены графические результаты замеров по времени обработки запросов.

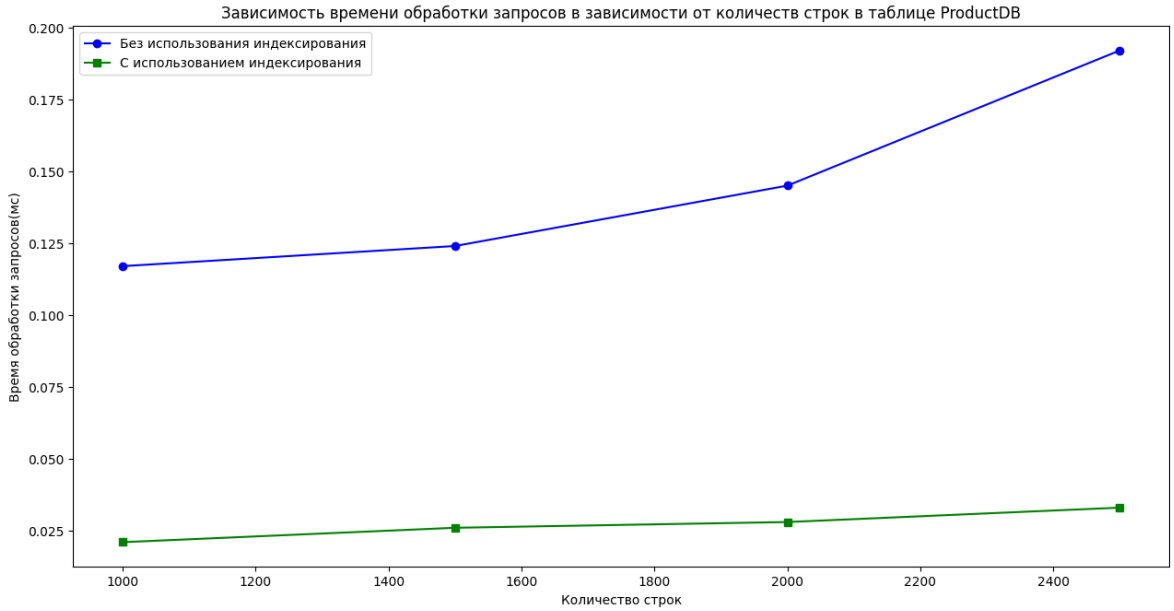

Рисунок 4.1 – Сравнение по времени обработки запросов.

В результате исследования было выяснено, что без использовании индексирования время обработки запросов медленнее, чем при использовании индексирования, в 5 раз.

Вывод

В ходе эксперимента было установлено, что использование индексов существенно ускоряет выполнение SELECT запросов. Поэтому применение индексов в данной работе будет способствовать повышению общей эффективности системы.