



Министерство науки и высшего образования Российской
Федерации
Федеральное государственное бюджетное образовательное
учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Международных образовательных программ»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

КУРСОВАЯ РАБОТА

НА ТЕМУ:

«Разработка базы данных компьютерного магазина»

Студент группы ИУ7и-62Б

(Подпись, дата)

Бу Х.Д.

(И.О. Фамилия)

Руководитель

(Подпись, дата)

Кивва К.А.

(И.О. Фамилия)

2024 г.

РЕФЕРАТ

Расчетно-пояснительная записка 52 страница, 21 рисунков, 9 таблицы.

Целью данного курсового проекта является разработка базы данных для компьютерного магазина.

В ходе выполнения работы были установлены основные роли пользователей, такие как администраторы, поставщики и клиенты. Определены ключевые сущности, которые должны храниться в базе данных, включая информацию о товарах, заказах, клиентах, поставщиках и промокодах.

Для оптимизации процесса обработки заказов был реализован триггер, автоматически обновляющий остаток товаров на складе при размещении и оформлении заказов. Был разработан пользовательский интерфейс, который обеспечивает удобный доступ и взаимодействие с базой данных.

Ключевые слова: базы данных, СУБД, C#, PostgreSQL.52

СОДЕРЖАНИЕ

РЕФЕРАТ	2
Введение	5
1 Аналитический раздел	7
1.1 Анализ предметной области	7
1.2 Постановка задачи	9
1.3 Формализация данных	10
1.4 Системные пользователи	12
1.5 Модель баз данных	13
1.5.1 Дореляционные модели данных	14
1.5.2 Реляционные модели данных	15
1.5.3 Постреляционные модели данных	16
1.5.4 Выбор модели данных	17
2 Конструкторский раздел	18
2.1 Проектирование базы данных	18
2.1.1 Таблицы базы данных	18
2.1.2 Ролевая модель	21
2.2 Триггеры	22
2.3 Декомпозиция разрабатываемого программного обеспечения	24
3 Технологический раздел	26

3.1	Выбор языка программирования и среды разработки	26
3.2	Выбор СУБД	27
3.3	Создание объектов баз данных	28
3.4	Реализация программы	30
3.5	Интерфейс программы	31
3.6	Тестирование триггера	31
4	Исследовательский раздел	33
4.1	Технические характеристики	33
4.2	Описание эксперимента и результаты исследования	34
	Заключение	36
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	37
	ПРИЛОЖЕНИЕ А	38

Введение

В условиях быстрого развития технологий и увеличения ассортимента товаров, компьютерные магазины сталкиваются с необходимостью оптимизации управления своими запасами и процессами продаж. Ручное ведение учета, которое использовалось ранее, стало неэффективным и требует значительных временных и трудовых ресурсов, что увеличивает вероятность ошибок и усложняет оперативное принятие решений. Внедрение системы управления ассортиментом на основе базы данных позволяет автоматизировать ключевые процессы, такие как отслеживание складских остатков, обработка транзакций. Такая система обеспечивает более точное управление запасами, снижает риск ошибок, ускоряет обработку информации и улучшает взаимодействие с клиентами. В результате, магазины могут значительно повысить свою производительность, снизить затраты времени и ресурсов, а также предоставить более высокий уровень обслуживания, что в конечном итоге способствует укреплению их конкурентных преимуществ на рынке. [1]

Цель данной работы — разработка базы данных для компьютерного магазина.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- проанализировать предметной области, требования к базе данных и приложению;
- определить основные роли пользователей;
- описать структуру базы данных, включая основные сущности и их атрибуты. Описать связи между сущностями и обеспечить целостность данных;
- разработать программное обеспечение для взаимодействия с базой данных, который позволит пользователям выполнять свои задачи;
- определить средства программной реализации;

- реализовать программное обеспечение для взаимодействия с базой данных, которое позволит пользователям выполнять свои задачи;
- провести экспериментальные замеры временных характеристик разработанного программного обеспечения.

1 Аналитический раздел

1.1 Анализ предметной области

Компьютерный магазин предлагает широкий ассортимент продукции, включающий компьютеры, комплектующие, периферийные устройства, программное обеспечение и разнообразные аксессуары. Чтобы эффективно управлять таким объемом товаров, требуется систематизированный подход к учету и обработке информации о каждом товаре. Необходимо учитывать важные характеристики, такие как название продукта, его подробное описание, производитель, цена, количество на складе и уникальный идентификатор товара. Эти данные обеспечивают прозрачность и контроль за наличием товаров, что позволяет избегать ошибок при учете и управлении запасами. Одной из наиболее важных задач в рамках системы управления является поддержание актуальной информации о количестве товаров на складе. Это помогает магазину оперативно реагировать на изменения спроса, пополнять запасы вовремя и избежать ситуаций, когда какие-либо позиции оказываются недоступными для покупателей. [2]

Кроме того, система управления ассортиментом должна поддерживать такие функции, как создание, редактирование и отслеживание заказов. Каждый заказ должен содержать информацию о клиенте, включая его контактные данные, а также подробности о заказе, такие как список выбранных товаров, их количество, общая стоимость и статусы выполнения заказа. Эти статусы, такие как "оформлен", "обработан", "в пути" и "доставлен", позволяют магазину и клиентам отслеживать текущий этап выполнения заказа. Это не только упрощает контроль за логистикой, но и помогает улучшить взаимодействие с клиентом, предоставляя ему более точную и актуальную информацию о его покупке. [3]

Для повышения уровня обслуживания необходимо вести детализированный учет данных о клиентах. Сохранение информации, такой как имя,

контактные данные и история покупок, позволит магазину персонализировать свои предложения и лучше понимать потребности каждого покупателя. Это поможет не только предложить более релевантные товары и услуги, но и улучшить качество клиентского обслуживания в целом, что положительно скажется на лояльности покупателей. Таким образом, грамотно разработанная система управления ассортиментом и заказами обеспечивает не только эффективное управление бизнес-процессами, но и создает условия для улучшения качества обслуживания клиентов и повышения конкурентоспособности магазина на рынке. [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.

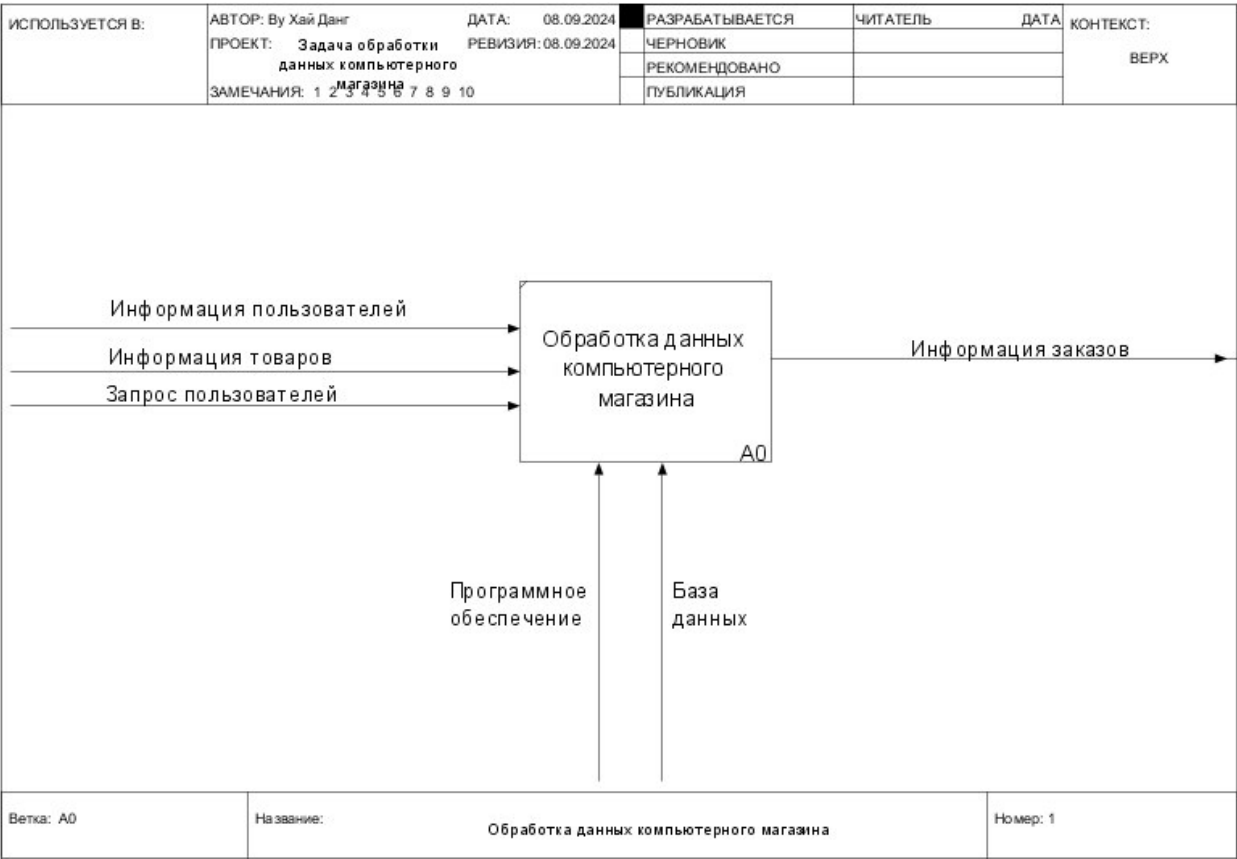


Рисунок 1.1 – Функциональная модель поставленной задачи

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	Имя, номер телефона, адрес, почта, логин, пароль, права доступа.
Товар	Название, цена, количество, производитель, описание.
Промокод	Код, акция, дата начала, дата конца.
Корзина	Дата создания, Пользователя.
Деталь корзины	Товары, корзины, количество.
Заказ	Пользователя, промокода, статус, дата создания.
Деталь заказа	Пользователя, заказа, количество.

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

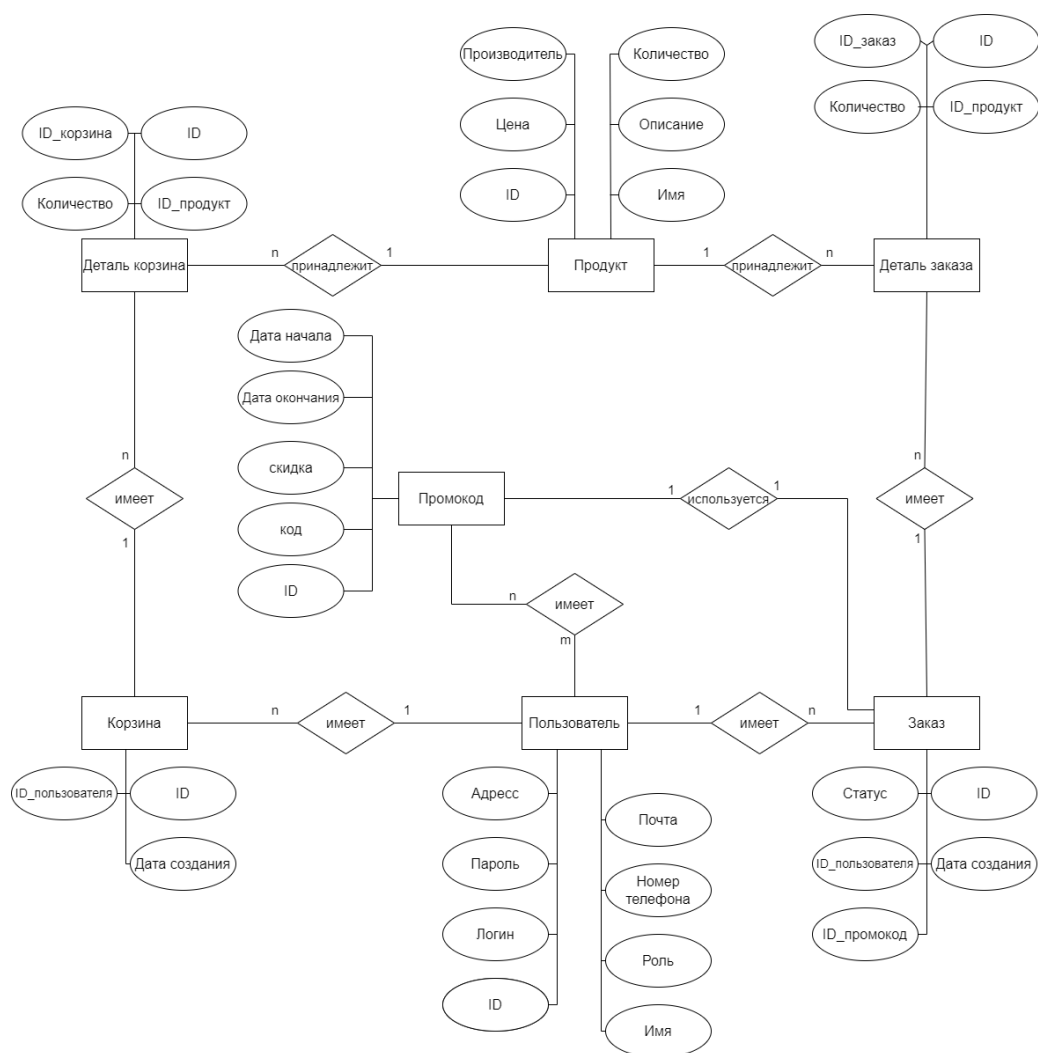


Рисунок 1.2 – ER-диаграмма

1.4 Системные пользователи

Для взаимодействия с программным обеспечением было выделено четыре вида ролей пользователей: гость, администратор, поставщик и клиент. Каждая из ролей обладает различными правами доступа и функциональностью, адаптированной к их задачам.

- 1) Гость: имеет доступ к просмотру доступных товаров, но не может совершать действия, требующие авторизации, такие как оформление заказов или редактирование данных;
- 2) Администратор: обладает максимальными правами доступа. В его компетенции управление всем приложением, включая редактирование и удаление товаров, управление пользователями, редактирование заказов;
- 3) Поставщик: ответственен за добавление новых товаров и управление их количеством на складе. Он может обновлять информацию о продуктах, включая цены, описания и наличие на складе;
- 4) Клиент: имеет возможность оформлять заказы, просматривать свою историю покупок, редактировать личные данные. Клиенты могут взаимодействовать с корзиной товаров, осуществлять оплату и отслеживать статус своих заказов.

На рисунке 1.3 представлена Use-Case диаграмма пользователей.

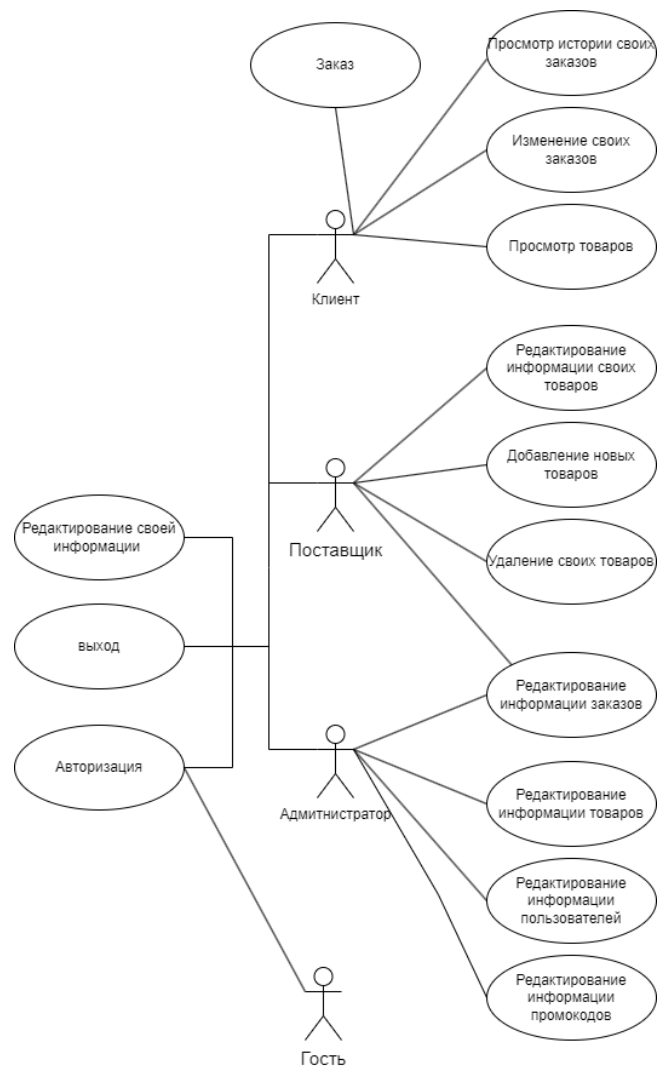


Рисунок 1.3 – Use-Case диаграмма

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 разделяются 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]

Одним из основных ограничений иерархических баз данных является трудность представления более сложных взаимосвязей, таких как отношения «многие-ко-многим». В такой структуре данные жёстко связаны, что усложняет внесение изменений или расширение системы, особенно если данные не вписываются в строгую иерархию. Также требуется дублирование данных, если одни и те же «потомки» должны быть связаны с несколькими «родителями», что приводит к избыточности.

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]

Одним из ключевых преимуществ сетевой модели является её высокая производительность при выполнении сложных запросов, так как она позволяет напрямую переходить между связанными узлами без необходимости прохода через промежуточные уровни, как это происходит в иерархической модели. Это делает её эффективной для обработки больших объемов данных с множеством взаимосвязей.

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]

Одной из ключевых особенностей реляционных баз данных является возможность их масштабирования и высокой совместимости с различными приложениями и системами. В то же время, для сложных структур данных, таких как графы или документы, реляционная модель может быть менее эффективной, так как требует преобразования таких данных в таблицы, что иногда приводит к усложнению запросов и снижению производительности.

На рисунке 1.6 представлен пример реляционной модели данных

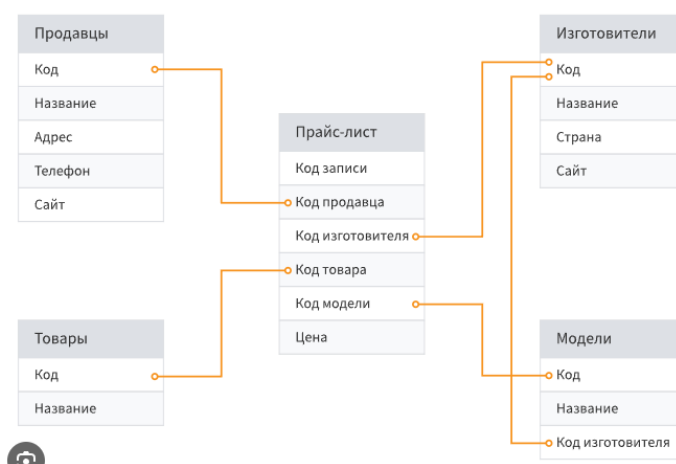


Рисунок 1.6 – Пример реляционных моделей данных

1.5.3 Постреляционные модели данных

Современные постреляционные СУБД обладают гибкостью, позволяющей хранить и обрабатывать разнообразные типы данных, включая документы, графы и мультимедийные объекты. Это делает их особенно полезными для приложений, требующих работы с большими объемами неструктурированных данных, таких как социальные сети, системы рекомендаций и аналитика больших данных. Важной особенностью постреляционных СУБД является поддержка расширяемых схем данных, что позволяет динамически изменять структуру базы данных без значительных затрат на её перестройку.

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.

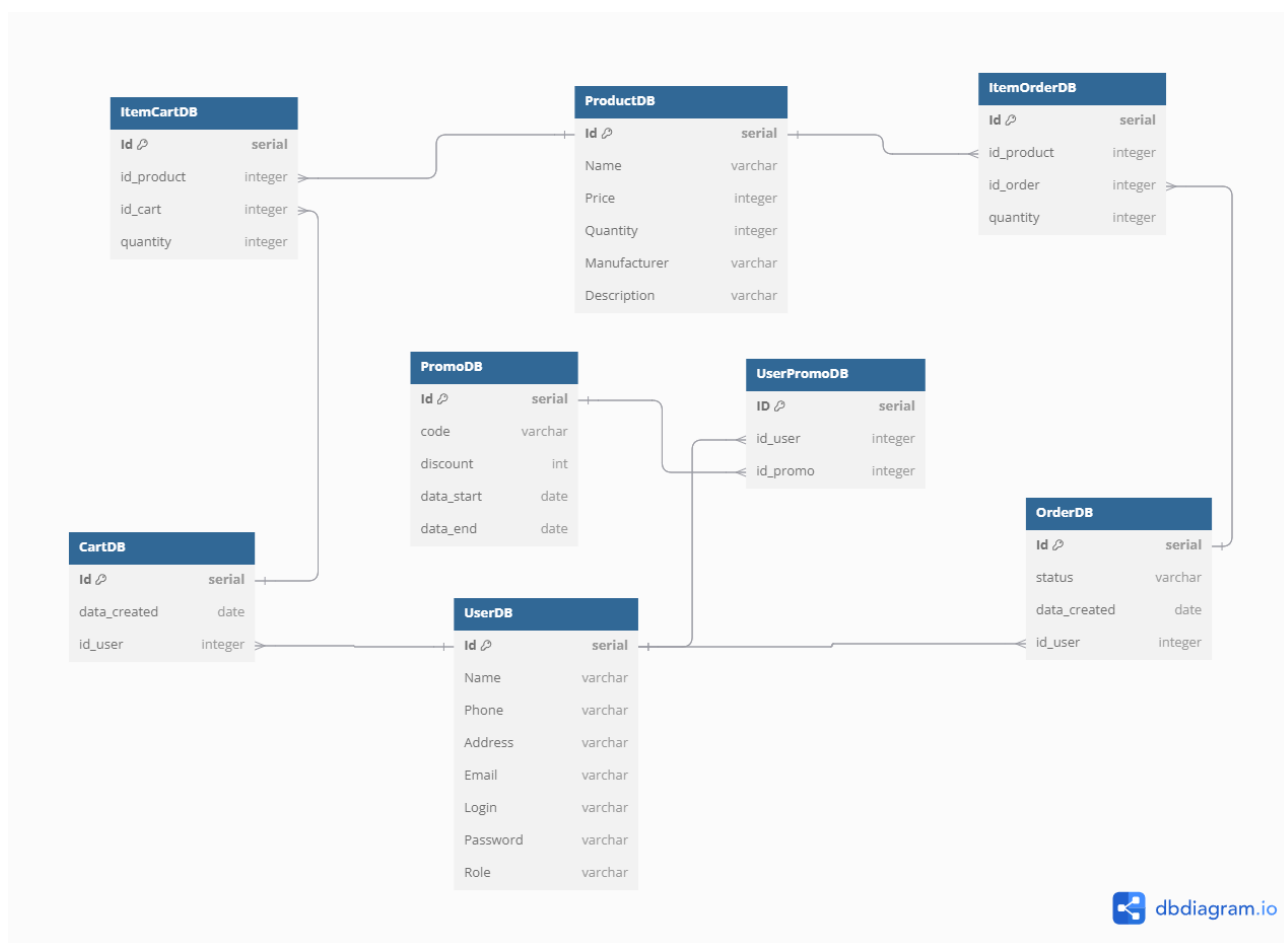


Рисунок 2.1 – Диаграмма базы данных

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

Таблица 2.1 – Сведение о таблице userdb

Столбец	Тип данных	Значение
id	serial	Идентификатор
name	VARCHAR(50)	имя
phone	VARCHAR(50)	Номер телефона
address	VARCHAR(50)	Адрес
email	VARCHAR(50)	Почта
login	VARCHAR(50)	Логин
password	VARCHAR(50)	Пароль
role	VARCHAR(50)	Права доступа

Таблица 2.2 – Сведение о таблице promodb

Столбец	Тип данных	Значение
id	serial	Идентификатор
code	VARCHAR(50)	Код
discount	INT	Акция
data_start	DATE	Дата начала
data_end	DATE	Дата конца

Таблица 2.3 – Сведение о таблице productdb

Столбец	Тип данных	Значение
id	serial	Идентификатор
name	VARCHAR(50)	Название
price	INT	Цена
quantity	VARCHAR(50)	Количество на складе
manufacturer	VARCHAR(50)	Производитель
description	VARCHAR(50)	Описание

Таблица 2.4 – Сведение о таблице Cartdb

Столбец	Тип данных	Значение
id	serial	Идентификатор
data_created	DATE	Дата создания
id_user	INT	Идентификатор пользователя

Таблица 2.5 – Сведение о таблице ItemCartdb

Столбец	Тип данных	Значение
id	serial	Идентификатор
id_product	VARCHAR(50)	Идентификатор товары
id_cart	INT	Идентификатор корзины
Quantity	INT	Количество

Таблица 2.6 – Сведение о таблице Orderdb

Столбец	Тип данных	Значение
id	serial	Идентификатор
status	VARCHAR(50)	Дата создания
data_created	DATE	Дата создания
id_user	INT	Идентификатор пользователя
id_promo	INT	Идентификатор промокода

Таблица 2.7 – Сведение о таблице ItemOrderdb

Столбец	Тип данных	Значение
id	serial	Идентификатор
id_product	INT	Идентификатор товары
id_order	INT	Идентификатор заказа
quantity	INT	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 просмотра всех таблиц, добавления, изменения, удаления из всех таблиц.

2.2 Триггеры

Триггеры в PostgreSQL — это инструмент для автоматизации определённых действий в базе данных, которые выполняются автоматически в ответ на события, такие как вставка, обновление или удаление данных [7]. Основные особенности и преимущества использования триггеров в PostgreSQL:

- 1) Триггеры позволяют автоматизировать задачи, такие как обновление связанных таблиц, ведение логов изменений или валидация данных, без необходимости вручную писать код в приложении для выполнения этих действий.
- 2) PostgreSQL триггеры могут срабатывать на различные операции (INSERT, UPDATE, DELETE), а также на уровне строк или таблиц. Это позволяет гибко настраивать, когда и как они должны выполняться.
- 3) Триггеры могут быть настроены на выполнение как до выполнения операции (BEFORE), так и после неё (AFTER). Это даёт возможность, например, проверять или изменять данные до того, как они будут записаны в базу.
- 4) Триггер проверяет каждую введенную команду, насколько она достоверна и реальна. Эта процедура помогает избежать ошибок в базе данных.

В этом проекте используются триггеры, когда клиент размещает заказ, количество товара на складе автоматически уменьшается на количество товара в заказе.

Ниже, на рисунке 2.2, представлена схема вышеуказанного триггера.

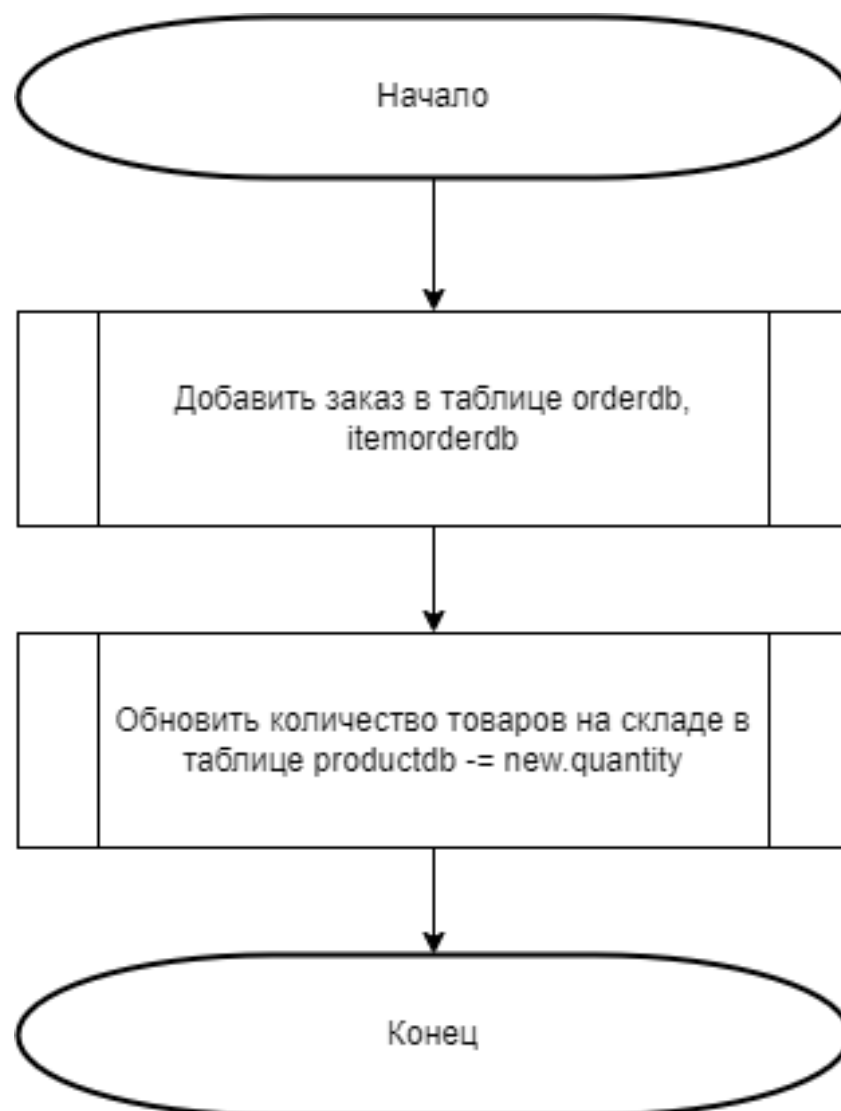


Рисунок 2.2 – Схема триггера

2.3 Декомпозиция разрабатываемого программного обеспечения

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

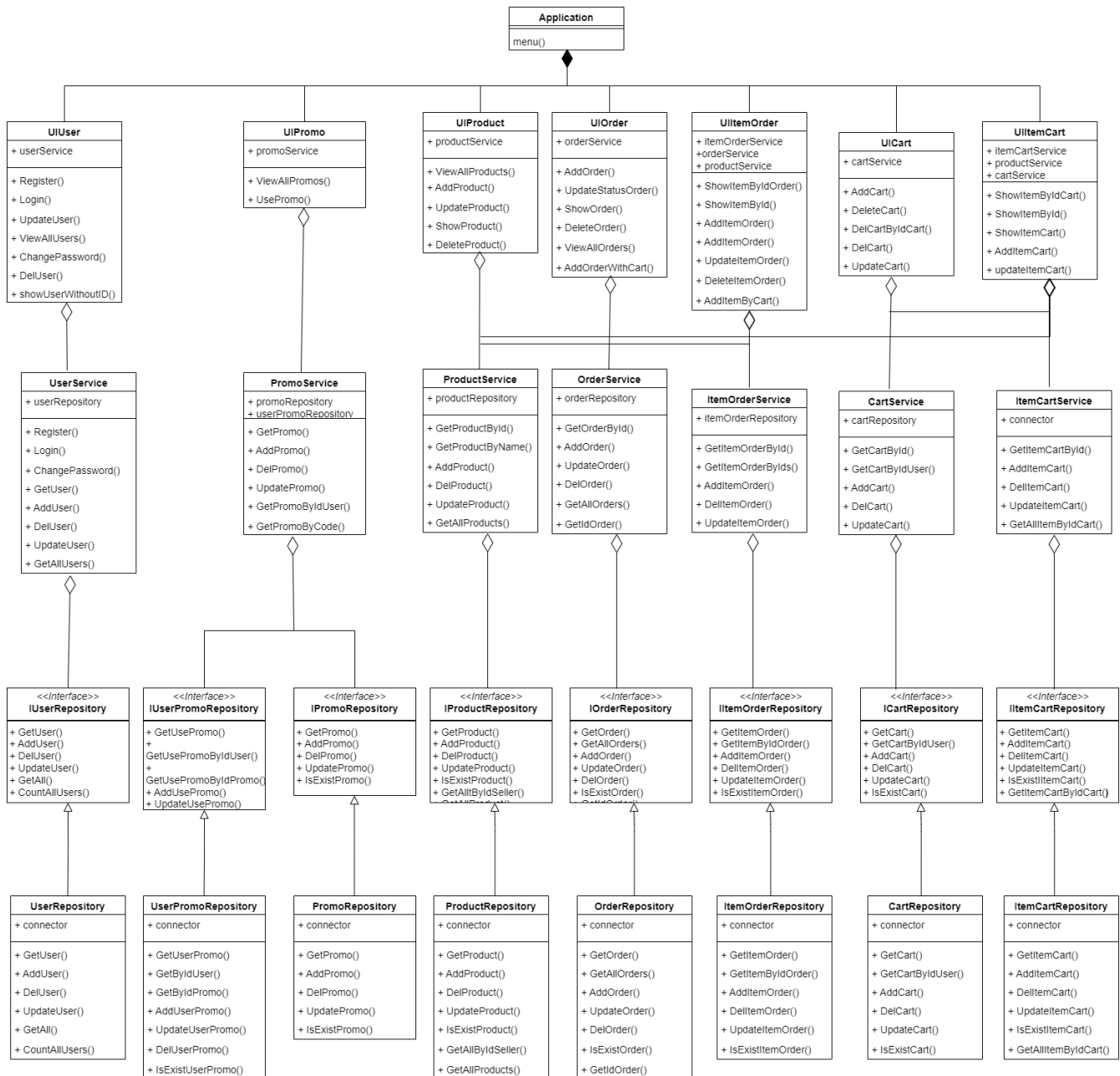


Рисунок 2.3 – Диаграмма разработанного программного обеспечения

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 CartRepository, ICartRepository - 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: пользователь, промокод, товар, заказ, деталь заказа, корзина, деталь корзины;
- class UIUser, UIUserPromo, UIPromo, UIProduct, UIOrder, UIItemOrder, UICart, UIItemCart – эти классы графического пользовательского интерфейса соответствующий сущностей: пользователь, 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

3.1 Выбор языка программирования и среды разработки

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

- Обширная стандартная библиотека: C# предоставляет мощный набор встроенных структур данных (коллекции, массивы, списки, словари и др.), которые можно легко адаптировать для работы с большими объемами данных. Это упрощает манипуляции с данными в приложении и делает код более понятным и поддерживаемым;
- Совместимость с PostgreSQL: C# имеет отличные библиотеки для работы с базами данных, включая PostgreSQL. Библиотека Npgsql позволяет легко подключаться к PostgreSQL, выполнять запросы и управлять данными. Это обеспечивает надежное взаимодействие с базой данных без необходимости написания сложного кода;
- В C# широко используется ORM-библиотека Entity Framework, которая упрощает работу с реляционными базами данных, предоставляя возможность манипулировать данными с использованием объектно-ориентированного подхода. Это повышает продуктивность разработки, так как разработчикам не нужно писать сложные SQL-запросы вручную;
- C# поддерживает асинхронное выполнение операций, что особенно важно при работе с базами данных. Асинхронные методы позволяют избежать блокировки потоков при выполнении долгих запросов, тем самым улучшая общую производительность приложения;

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

3.2 Выбор СУБД

PostgreSQL была выбрана в качестве СУБД [8]:

- PostgreSQL распространяется с открытым исходным кодом, что означает отсутствие лицензионных затрат и свободу в настройке и расширении системы под нужды конкретного проекта. Это делает её идеальным выбором для стартапов и малых предприятий, а также крупных организаций;
- PostgreSQL может работать как с реляционными (структурированными) данными, так и с нереляционными (неструктурированными), такими как JSON, XML и HSTORE. Это делает её идеальной для гибридных систем, требующих универсальности в работе с данными;
- Поддерживает основные платформы, включая Linux, Windows и macOS, что делает её универсальной для различных операционных систем;
- Встроенные функции безопасности, такие как поддержка DBMS_SESSION для управления сессиями пользователей, сложные механизмы контроля доступа, а также возможность шифрования данных на уровне транспортного слоя (SSL), делают PostgreSQL надёжным выбором для работы с конфиденциальной информацией;

3.3 Создание объектов баз данных

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

Листинг 3.1 – Создание таблицы UserDB

```
1      create table UserDB(  
2          Id serial primary key,  
3          Name varchar(50) not null,  
4          Phone varchar(50) not null,  
5          Address varchar(50) not null,  
6          Email varchar(50) not null,  
7          Login varchar(50) unique,  
8          Password varchar(50) not null,  
9          Role varchar(50) not null check (role in ('admin',  
              'seller', 'client'))  
10     );
```

Листинг 3.2 – Создание таблицы PromoDB

```
1      create table PromoDB(  
2          Id serial primary key,  
3          code varchar(20) unique,  
4          discount int not null,  
5          data_start date not null,  
6          data_end date not null  
7      );
```

Листинг 3.3 – Создание таблицы ProductDB

```
1      create table ProductDB (  
2          Id serial primary key,  
3          Name varchar(50) not null,  
4          Price int not null,  
5          Quantity int not null,  
6          Manufacturer varchar(50) not null,  
7          Description varchar(50) not null  
8      );
```

Листинг 3.4 – Создание таблицы CartDB

```
1      create table CartDB(  
2          Id serial primary key,  
3          data_created date not null,  
4          id_user int references UserDB(Id) ON DELETE CASCADE  
5      );
```

Листинг 3.5 – Создание таблицы ItemCartDB

```
1      create table ItemCartDB(  
2          Id serial primary key,  
3          id_product int,  
4          id_cart int,  
5          quantity int not null,  
6          foreign key (id_cart) references CartDB(Id) ON DELETE  
          CASCADE,  
7          foreign key (id_product) references ProductDB(Id) ON  
          DELETE CASCADE  
8      );
```

Листинг 3.6 – Создание таблицы OrderDB

```
1      create table orderDB(  
2          Id serial primary key,
```

```

3      status varchar(20) not null ,
4      data_created date not null ,
5      id_user int not null ,
6      id_promo int references PromoDB(id) ON DELETE CASCADE,
7      foreign key (id_user) references UserDB(id) ON DELETE
      CASCADE
8  );

```

Листинг 3.7 – Создание таблицы ItemOrderDB

```

1      create table ItemOrderDB(
2      id serial primary key ,
3      id_product int not null ,
4      id_order int not null ,
5      quantity int not null ,
6      foreign key (id_order) references OrderDB(id) ON DELETE
      CASCADE,
7      foreign key (id_product) references ProductDB(id) ON
      DELETE CASCADE
8  );

```

3.4 Реализация программы

На листингах 4.8 - 4.9 в приложении А представлены примеры основных функций программы, реализующие ключевые действия пользователей, такие как редактирование товаров и редактирование информации о пользователях. Эти функции включают в себя обработку пользовательских данных, валидацию ввода и взаимодействие с базой данных для обновления соответствующих записей. Примеры кода демонстрируют архитектуру приложения и логику выполнения операций по управлению товарами и пользовательскими данными.

3.5 Интерфейс программы

На рисунках 4.2 – 4.12 в приложении А отображён интерфейс приложения. Эти изображения соответствуют каждой роли пользователя и их операции. Каждый рисунок иллюстрирует различные экраны и элементы интерфейса, соответствующие операциям, которые могут выполнять пользователи в зависимости от их роли.

Для каждой роли пользователя предусмотрены свои функции и права доступа. Интерфейс адаптирован под разные задачи, такие как редактирование товаров, управление учетными записями пользователей, просмотр и изменение данных, а также выполнение административных операций.

Изображения демонстрируют ключевые элементы взаимодействия, включая формы для ввода и редактирования данных, элементы навигации, кнопки действий, а также сообщения об успешных операциях или ошибках. Весь интерфейс разработан с учётом требований удобства и простоты использования, что делает приложение доступным для пользователей с любым уровнем технической подготовки. Кроме того, все операции сопровождаются соответствующей валидацией и обработкой ошибок, что повышает стабильность работы приложения и защищает его от некорректного ввода данных."

3.6 Тестирование триггера

На листинге 4.2 в приложении А представлено создание триггера, который автоматически рассчитывает количество товаров на складе когда клиент размещает заказ.

Проводится тестирование триггера, который автоматически обновляет значение поля `quantity` в таблице `productdb`, отражающее количество доступных товаров на складе. Триггер срабатывает каждый раз, когда клиент совершает заказ, уменьшает количество товаров на складе в соответствии с числом заказанных единиц, и записывает обновлённое значение в базу данных.

Тестирование включает в себя проверку корректности работы триггера в различных сценариях. Важно убедиться, что триггер правильно уменьшает quantity только после успешного оформления заказа и что нет расхождений между фактическим и заявленным количеством товаров.

На листинге 4.7 в приложении А представлена функция тестирования триггера.

На рисунке 3.1 представлена функция, предназначенная для тестирования триггера, который обновляет количество товаров на складе.

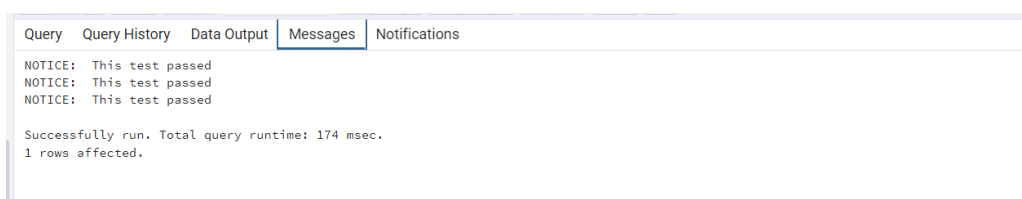


Рисунок 3.1 – Результаты тестов для триггера `after_itemorder_insert`.

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 достаточно для выполнения задач программирования, анализа данных ;
- центральный процессор Intel Core i7-9750H (2.6 ГГц, 6 ядер) поддерживает многопоточность и подходит для ресурсоемких вычислений, таких как обработка данных.

Во время проведения исследования устройство не подвергалось значительной дополнительной нагрузке, за исключением стандартных процессов операционной системы и встроенных приложений, которые выполнялись в фоновом режиме. Это позволило минимизировать влияние внешних факторов на результаты измерений. Также стоит отметить, что устройство было подключено к источнику электропитания, что позволило избежать возможных ограничений производительности, которые могли бы возникнуть при работе в режиме батареи.

4.2 Описание эксперимента и результаты исследования

Целью эксперимента является изучение влияния индексирования на время обработки запросов к таблицам базы данных.

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 количества 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.

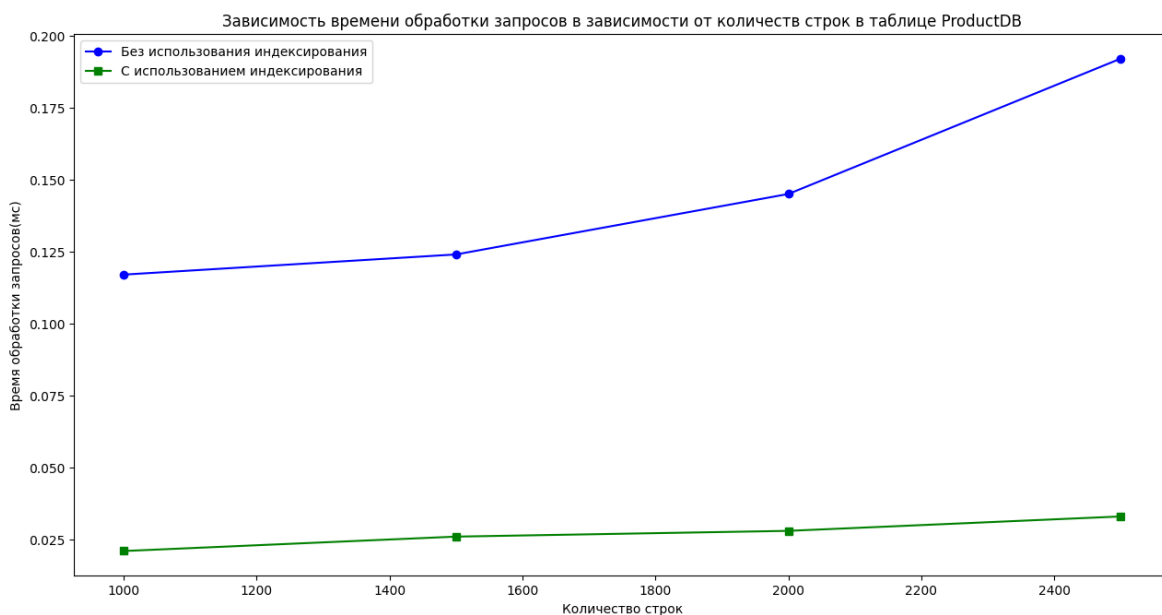


Рисунок 4.1 – Сравнение по времени обработки запросов.

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

Вывод

Эксперимент показал, что индексация заметно увеличивает скорость выполнения запросов SELECT. Поэтому внедрение индексов в данной системе приведет к улучшению её общей эффективности.

Заключение

В ходе выполнения курсовой работы 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. Спроектирована база данных и приложение для доступа к ней. Был спроектирован триггер, осуществляющий автоматический пересчет количества товаров на складе при добавлении заказов. Представлены средства разработки программного обеспечения, выбор языка программирования и описан интерфейс программы. Также рассмотрены примеры работы программы

В результате исследования было выяснено, что при использовании индексирования ускоряет выполнение запросов `SELECT` больше чем в 5 раз.

Данная программа может иметь следующие перспективы развития:

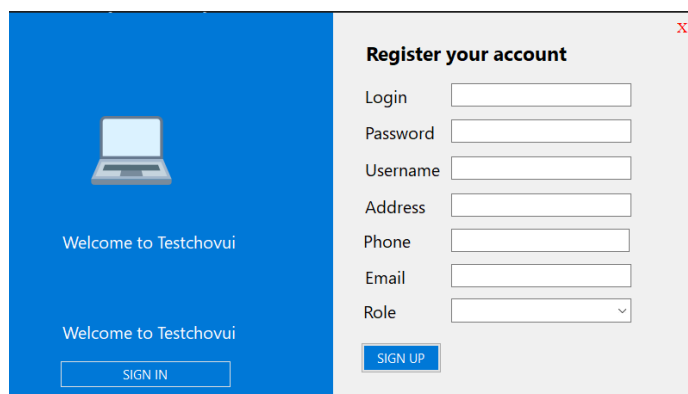
- анализ продаж: внедрение инструментов для анализа данных о продажах, тенденциях и предпочтениях клиентов;
- аналитические панели и отчеты: создание удобных интерфейсов для управления данными и генерации отчетов для аналитики;

Список использованных источников

1. Зачем нужен интернет-магазин? <https://integrion.biz/articles/zachem-nuzhen-internet-magazin>.
2. Бизнес-план магазина компьютерной техники.
<https://plan-pro.ru/torgovlya/raznye-magaziny/biznes-plan-magazina-kompyuternoj-tehniki/>.
3. Информационная система компьютерного магазина.
<https://cyberleninka.ru/article/n/informatsionnaya-sistema-kompyuternogo-magazina/viewer>.
4. КаРа-Ушанов В. Ю. SQL — Язык реляционных баз данных. 2016. — С. 6.
5. Документация по C#. <https://learn.microsoft.com/ru-ru/dotnet/csharp/>.
6. Что такое реляционная база данных. <https://help.reg.ru/support/server-y-vps/oblastnyye-bazy-dannykh/zakaz-i-upravleniye-uslugoy-oblastnyye-bazy-dannykh/relyatsionnyye-bazy-dannykh#0>.
7. Для чего используется триггер? <https://blog.skillfactory.ru/triggery-v-bazah-dannyh/>.
8. Сравнение современных СУБД. <https://drach.pro/blog/hi-tech/item/145-db-comparison>.

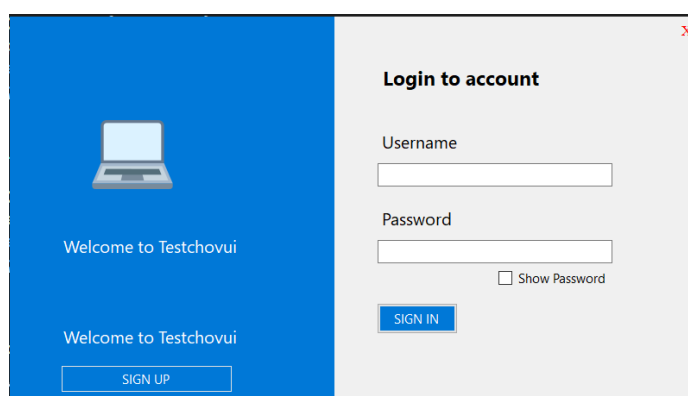
ПРИЛОЖЕНИЕ А

Описание интерфейсов



The screenshot shows a web interface for 'Testchovui'. On the left, a blue sidebar contains a laptop icon, the text 'Welcome to Testchovui', and a 'SIGN IN' button. On the right, a light gray panel titled 'Register your account' (with a red close button 'X' in the top right) contains several input fields: 'Login', 'Password', 'Username', 'Address', 'Phone', 'Email', and a 'Role' dropdown menu. A blue 'SIGN UP' button is located at the bottom of the registration form.

Рисунок 4.2 – Демонстрация работы программы при регистрации



The screenshot shows the same web interface as Figure 4.2, but the right panel is titled 'Login to account' (with a red close button 'X' in the top right). It contains input fields for 'Username' and 'Password', a 'Show Password' checkbox, and a blue 'SIGN IN' button. The blue sidebar on the left remains the same.

Рисунок 4.3 – Демонстрация работы программы при входе в систему

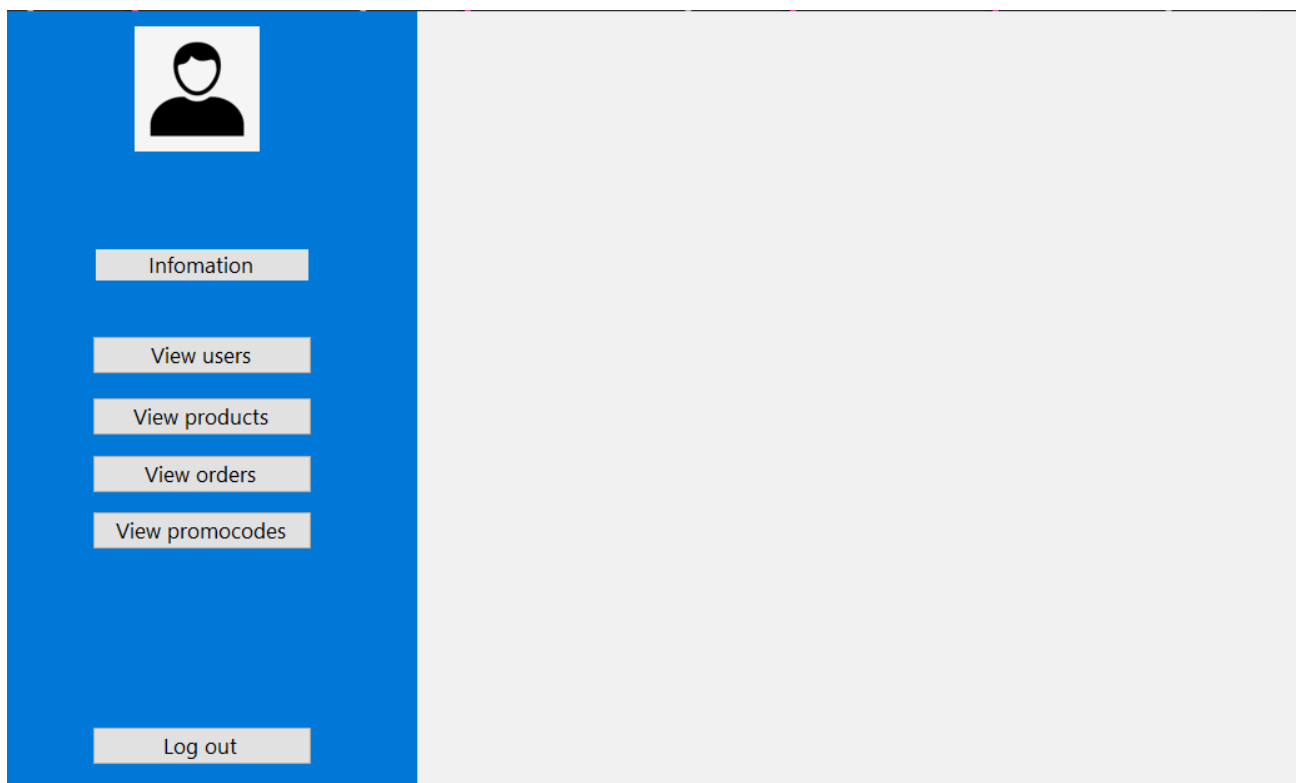


Рисунок 4.4 – Демонстрация главного экрана админстратора

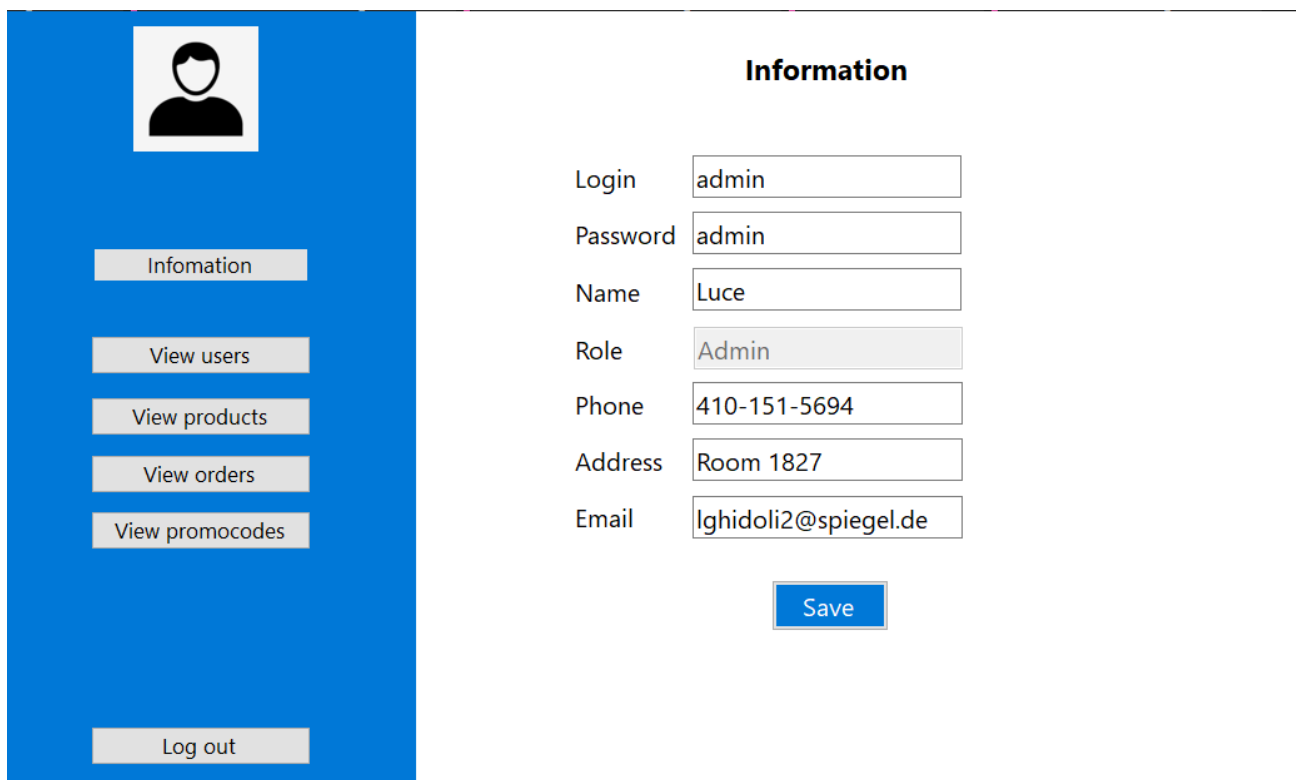



Рисунок 4.5 – Демонстрация экрана админстратора при просмотре информации



Information

View users

View products

View orders

View promocodes

Log out

ID	Login	Password	Name	Address	Phone
1	admin	admin	Luce	Room 1827	410
2	client	client	Lacey	PO Box 90...	37
4	Spriggs	ed	Lacy	Apt 708	98
5	Elden	123	Kaylyn	PO Box 61...	66
6	Longland	33	Krystal	8th Floor	59
7	Downing	fp5#4//D6	Trstram	PO Box 18...	79
9	Daughton	iD9~0Luqll...	Madelene	19th Floor	23
10	McKinna	rB9/\$W50j...	Borq	PO Box 85...	28
8	seller	seller	Michael	Apt 1109	84

Login
 Daughton
 Phone
 231-252-1575


Password
 iD9~0Luqll\$#K67
 Address
 19th Floor

Name
 Madelene
 Email
 mdaughton8@vkontakte

Role
 Client

Add
 Delete
 Update

Рисунок 4.6 – Демонстрация экрана администратора при просмотре пользователей



Information

View users

View products

View orders

View promocodes

Log out

ID	Name	Price	Quantity	Manufacturer
1	Galliano	12	75	Coqilith
2	Fireball Wh...	96	49	Browseblab
3	Cookies - A...	100	72	Jabbertype
4	Mix Pina C...	79	51	Myworks
5	Beef Cheek ...	99	28	Aqivu
6	Jolt Cola - ...	51	8	Roexo
7	Beer - Laba...	68	37	Zooveo
8	Tomatoes - ...	24	71	Thoughttwo...
9	Bacardi Bre...	26	29	Fiveclub
10	Pepper - Re...	88	74	Coqidoo

Name
 Beer - Labatt Blue
 Price
 68

Manufacturer
 Zooveo
 Quantity
 37

Description
 Insect, stick

Add
 Delete
 Update

Рисунок 4.7 – Демонстрация экрана администратора при просмотре товаров

ID	Data	Id_User	Id_promo	Status
1	6/10/24 12...	1	1	Init
2	10/23/23 1...	2	2	Init
4	2/9/24 12:...	4	4	Init
5	12/31/23 1...	5	5	Init
6	8/18/23 12...	6	6	Delivered
7	3/21/24 12...	7	7	Delivered
8	1/11/24 12...	8	8	Delivered
9	12/27/23 1...	10	9	Delivered
10	2/23/24 12...	10	10	Delivered
11	8/29/24 12...	2	1	Init
12	8/29/24 12...	2	2	Init


Status: Data:
 User Login: PromoCode:

Рисунок 4.8 – Демонстрация экрана администратора при просмотре заказов

ID	Code	Discount	Start	End
1	0	0	12/31/23 1...	11/2/23 12...
2	HSV	41	4/19/24 12...	7/10/24 12...
3	CIE	8	8/20/23 12...	7/1/24 12:...
4	OZC	78	2/28/24 12...	4/13/24 12...
5	YLV	35	9/29/23 12...	8/14/23 12...
6	CDU	30	7/25/24 12...	4/14/24 12...
7	MXK	99	12/8/23 12...	5/29/24 12...
8	AVB	89	9/8/23 12:...	11/23/23 1...
9	SUG	74	11/25/23 1...	11/1/23 12...
10	IKK	32	1/3/24 12:...	4/16/24 12...

Code: Discount:
 Start: End:

Рисунок 4.9 – Демонстрация экрана администратора при просмотре промокодов



Information

View products

View orders

Log out

Information

Login

seller

Password

seller

Name

Michael

Role

Seller

Phone

842-721-8242

Address


Apt 1109

Email

fsdf

Save

Рисунок 4.10 – Демонстрация главного экрана поставщика



Information

View users

View products

View orders

View promocodes

Log out

ID	Code	Discount	Start	End
1	0	0	12/31/23 1...	11/2/23 12...
2	HSV	41	4/19/24 12...	7/10/24 12...
3	CIE	8	8/20/23 12...	7/1/24 12:...
4	OZC	78	2/28/24 12...	4/13/24 12...
5	YLV	35	9/29/23 12...	8/14/23 12...
6	CDU	30	7/25/24 12...	4/14/24 12...
7	MXK	99	12/8/23 12...	5/29/24 12...
8	AVB	89	9/8/23 12:...	11/23/23 1...
9	SUG	74	11/25/23 1...	11/1/23 12...
10	IKK	32	1/3/24 12:...	4/16/24 12...
*				

Code

Discount

Start


End

Add

Delete

Update

Рисунок 4.11 – Демонстрация экрана клиентов при просмотре корзины



Information

View products

View carts

View orders

View promos

Log out

ID	Name	Price	Quantity	Manufacturer
1	Galliano	12	75	Cogilith
2	Fireball Whisky	96	49	Browseblab
3	Cookies - Assor...	100	72	Jabbertype
4	Mix Pina Colada	79	51	Myworks
5	Beef Cheek Fresh	99	28	Agivu
6	Jolt Cola - Elect...	51	8	Rooxo
7	Beer - Labatt Bl...	68	37	Zooveo
8	Tomatoes - Vin...	24	71	Thoughtworks
9	Bacardi Breezer...	26	29	Fiveclub
10	Pepper - Red C...	88	74	Cogidoo

Name

Mix Pina Colada

Price

79

Manufacturer

Myworks

Quantity

51

Description

Smith's bush squirrel

Choose Cart

Add to Cart

Рисунок 4.12 – Демонстрация экрана клиентов при добавления товаров в корзину

Создание базы данных

Листинг 4.1 – Создание всех таблиц

```
1      drop table userdb CASCADE;
2      drop table promodb CASCADE;
3      drop table productdb CASCADE;
4      drop table userpromodb CASCADE;
5      drop table cartdb CASCADE;
6      drop table itemcartdb CASCADE;
7      drop table orderdb CASCADE;
8      drop table itemorderdb CASCADE;
9
10     create table UserDB(
11         Id serial primary key,
12         Name varchar(50) not null,
13         Phone varchar(50) not null,
14         Address varchar(50) not null,
15         Email varchar(50) not null,
16         Login varchar(50) not null,
17         Password varchar(50) not null,
18         Role varchar(50) not null
19     );
20
21     create table PromoDB(
22         Id serial primary key,
23         code varchar(20) not null,
24         discount int not null,
25         data_start date not null,
26         data_end date not null
27     );
28     create table ProductDB (
29         Id serial primary key,
30         Name varchar(50) not null,
31         Price int not null,
```

```

32     Quantity int not null ,
33     Manufacturer varchar(50) not null ,
34     Description varchar(50) not null
35 );
36 create table UserPromoDB(
37     ID serial primary key ,
38     id_user int not null ,
39     id_promo int not null ,
40     foreign key (id_user) references UserDB(Id) ON DELETE CASCADE,
41     foreign key (id_promo) references PromoDB(Id) ON DELETE
        CASCADE
42 );
43
44 create table CartDB(
45     Id serial primary key ,
46     data_created date not null ,
47     id_user int references UserDB(Id) ON DELETE CASCADE
48 );
49
50 create table ItemCartDB(
51     Id serial primary key ,
52     id_product int not null ,
53     id_cart int not null ,
54     quantity int not null ,
55     foreign key (id_cart) references CartDB(Id) ON DELETE CASCADE,
56     foreign key (id_product) references ProductDB(Id) ON DELETE
        CASCADE
57 );
58
59 create table orderDB(
60     Id serial primary key ,
61     status varchar(20) not null ,
62     data_created date not null ,
63     id_user int not null ,
64     id_promo int references PromoDB(Id) ON DELETE CASCADE,

```

```

65     foreign key (id_user) references UserDB(id) ON DELETE CASCADE
66 );
67
68     create table ItemOrderDB(
69     id serial primary key,
70     id_product int not null ,
71     id_order int not null ,
72     quantity int not null ,
73     foreign key (id_order) references OrderDB(id) ON DELETE
        CASCADE,
74     foreign key (id_product) references ProductDB(id) ON DELETE
        CASCADE
75 );

```

Листинг 4.2 – Реализация триггера

```

1     drop trigger after_itemorder_insert on itemorderdb;
2
3     create or replace function process_itemorder()
4     returns trigger
5     as
6     $$
7     begin
8     update productdb set quantity = (select quantity -
        new.quantity from productdb where id = new.id_product)
        where id = new.id_product;
9     RETURN NEW;
10
11     end;
12     $$ LANGUAGE plpgsql;
13
14     CREATE TRIGGER after_itemorder_insert
15     AFTER INSERT ON itemorderdb
16     for each row
17     EXECUTE FUNCTION process_itemorder();

```

Листинг 4.3 – Создание роли администратора и выдача права

```
1      create role Radmin with
2      connection limit -1
3      login
4      password 'admin';
5
6      grant all privileges
7      on all tables in schema public
8      to Radmin;
```

Листинг 4.4 – Создание роли поставщика и выдача права

```
1      create role Rseller with
2      connection limit 2
3      login
4      password 'seller';
5
6      grant select on
7      public."productdb",
8      public."orderdb",
9      public."userdb"
10     to Rseller;
11
12     grant insert on
13     public."productdb"
14     to Rseller;
15
16     grant update on
17     public."productdb",
18     public."orderdb",
19     public."itemorderdb",
20     public."userdb"
21     to Rseller;
22
23     grant delete on
```

```
24 public."productdb",
25 public."orderdb",
26 public."itemorderdb"
27 to Rseller;
```

Листинг 4.5 – Создание роли клиента и выдача права

```
1 create role Rclient with
2 connection limit 100
3 login
4 password 'client';
5
6 grant select on
7 public."productdb",
8 public."cartdb",
9 public."itemcartdb",
10 public."orderdb",
11 public."itemorderdb",
12 public."userdb"
13 to Rclient;
14
15 grant insert on
16 public."cartdb",
17 public."itemcartdb",
18 public."orderdb",
19 public."itemorderdb"
20 to Rclient;
21
22 grant update on
23 public."cartdb",
24 public."itemcartdb",
25 public."orderdb",
26 public."itemorderdb",
27 public."userdb"
28 to Rclient;
29
```



```

30 grant delete on
31 public."cartdb",
32 public."itemcartdb"
33 to Rclient;

```

Листинг 4.6 – Создание роли гостя и выдача права

```

1 create role Rguest with
2 connection limit 100
3 login
4 password 'guest';
5
6 grant insert on
7 public."userdb"
8 to Rguest;

```

Листинг 4.7 – Реализация тестирования для триггера after_itemorder_insert

```

1 CREATE OR REPLACE FUNCTION test_trigger(product_id int ,
2 order_id int , quantityl int)
3 RETURNS void AS $$
4 DECLARE
5 quantity_after int;
6 quantity_before int;
7 BEGIN
8 select quantity into quantity_before from productdb
9 where id = product_id;
10
11 INSERT INTO itemorderdb(id_product , id_order , quantity)
12 VALUES (product_id , order_id , quantityl);
13
14 SELECT quantity INTO quantity_after FROM
15 productdb WHERE id = product_id;
16
17 if (quantity_after = quantity_before - quantityl) then
18 RAISE NOTICE 'This test passed';
19 else

```

```

20 RAISE EXCEPTION 'Test failed';
21 end if;
22 END;
23 $$ LANGUAGE plpgsql;
24
25 — Запуск теста
26 SELECT test_trigger(1, 1, 5);
27 SELECT test_trigger(2, 1, 5);
28 SELECT test_trigger(3, 1, 5);

```

Листинг 4.8 – Функция входа в систему

```

1 private void btnSignIn_Click(object sender, EventArgs e)
2 {
3     try
4     {
5         string login = tbUsername.Text;
6         string password = tbPassword.Text;
7         if (login == "" || password == "")
8         {
9             throw new Exception("Input error");
10        }
11        User user = _userService.Login(login, password);
12        switch (user.Role)
13        {
14            case Role.Admin:
15                FormAdmin frm_admin = new FormAdmin(user.Id,
16                    _userService,
17                    _productService, _promoService, _orderService,
18                    _cartService,
19                    _itemOrderService, _itemCartService);
20                frm_admin.ShowDialog(this);
21                break;
22            case Role.Seller:
23                FormSeller frm_seller = new FormSeller(user.Id,
24                    _userService,

```

```

22         _productService , _promoService , _orderService ,
           _cartService ,
23         _itemOrderService , _itemCartService);
24     frm_seller.ShowDialog(this);
25     break;
26     case Role.Client:
27         FormClient frm_client = new FormClient(user.Id ,
           _userService ,
28         _productService , _promoService , _orderService ,
           _cartService ,
29         _itemOrderService , _itemCartService);
30         frm_client.ShowDialog(this);
31         break;
32     }
33     this.Close();
34 }
35 catch (Exception ex) { MessageBox.Show(ex.Message ,
           "Error" , MessageBoxButtons.OK ,
36         MessageBoxIcon.Error); }
37 }

```

Листинг 4.9 – Функция работы пользователей

```

1     public User GetUser(int id)
2     {
3         CheckConnection.checkConnection(Connector);
4         string query = queryGetUser(id);
5
6         NpgsqlCommand cmd = new NpgsqlCommand(query ,
           Connector.Connect);
7
8         User user = null;
9
10        NpgsqlDataReader reader = cmd.ExecuteReader();
11        if (reader.Read())
12        {

```

```

13         user = new User(reader.GetInt32(0),
14             reader.GetString(1), reader.GetString(2),
15             reader.GetString(3), reader.GetString(4),
16             reader.GetString(5), reader.GetString(6),
17             (Role)Enum.Parse(typeof(Role),
18                 reader.GetString(7)));
19     }
20     reader.Close();
21     return user;
22 }
23
24 public User GetUser(string login)
25 {
26     CheckConnection.checkConnection(Connector);
27     string query = queryGetUser(login);
28
29     NpgsqlCommand cmd = new NpgsqlCommand(query,
30         Connector.Connect);
31
32     User user = null;
33     NpgsqlDataReader reader = cmd.ExecuteReader();
34
35     if (reader.Read())
36     {
37         user = new User(reader.GetInt32(0),
38             reader.GetString(1), reader.GetString(2),
39             reader.GetString(3), reader.GetString(4),
40             reader.GetString(5), reader.GetString(6),
41             (Role)Enum.Parse(typeof(Role),
42                 reader.GetString(7)));
43     }
44     reader.Close();
45
46     return user;
47 }

```

```

39
40     public void AddUser(User user)
41     {
42         CheckConnection.checkConnection(Connector);
43         string query = queryAddUser(user);
44         NpgsqlCommand cmd = new NpgsqlCommand(query,
45             Connector.Connect);
46         cmd.ExecuteNonQuery();
47     }
48     public void DelUser(User user)
49     {
50         CheckConnection.checkConnection(Connector);
51         string query = queryDelUser(user);
52         NpgsqlCommand cmd = new NpgsqlCommand(query,
53             Connector.Connect);
54         cmd.ExecuteNonQuery();
55     }
56     public void UpdateUser(User user)
57     {
58         CheckConnection.checkConnection(Connector);
59         string query = queryUpdateUser(user);
60         NpgsqlCommand cmd = new NpgsqlCommand(query,
61             Connector.Connect);
62         cmd.ExecuteNonQuery();
63     }
64     public List<User> GetAll()
65     {
66         CheckConnection.checkConnection(Connector);
67         string query = queryGetAll();
68         List<User> allUser = new List<User>();
69         NpgsqlCommand cmd = new NpgsqlCommand(query,
70             Connector.Connect);
71         using (NpgsqlDataReader reader = cmd.ExecuteReader())
72         {

```

```

70         while (reader.Read())
71         {
72             allUser.Add(new User(reader.GetInt32(0),
73                                   reader.GetString(1), reader.GetString(2),
74                                   reader.GetString(3), reader.GetString(4),
75                                   reader.GetString(5), reader.GetString(6),
76                                   (Role)Enum.Parse(typeof(Role),
77                                   reader.GetString(7))));
78         }
79         reader.Close();
80     }
81     return allUser;
82 }
83
84 public int CountAllUsers()
85 {
86     CheckConnection.checkConnection(Connector);
87     string query = queryCountAllUsers();
88
89     NpgsqlCommand cmd = new NpgsqlCommand(query,
90                                             Connector.Connect);
91
92     return Convert.ToInt32(cmd.ExecuteScalar());
93 }

```