Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

К защите допустить:

Заведующий кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовому проекту  
на тему

**ПРОГРАМНОЕ СРЕДСТВО ДЛЯ ТОРГОВЛИ МУЖСКОЙ БИЖУТЕРИЕЙ**

БГУИР

БГУИР КП 1–40 04 01 073 ПЗ

Студент М. Ю. Пухов

Руководитель А. В. Давыдчик

Минск 2024

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc152964550)

[1 Архитектура вычислительной системы 6](#_Toc152964551)

[1.1 Структура и архитектура вычислительной системы 6](#_Toc152964552)

[1.2 История, версии и достоинства 9](#_Toc152964553)

[1.3 Обоснование выбора вычислительной системы 10](#_Toc152964554)

[2 Платформа программного обеспечения 12](#_Toc152964555)

[2.1 Структура и архитектура платформы 12](#_Toc152964556)

[2.2 Выбор платформы для написания программы. 14](#_Toc152964558)

[3 Теоретическое обоснование разработки програмного продукта 15](#_Toc152964560)

[3.1 Обоснование необходимости разработки 15](#_Toc152964561)

[3.2 Технологии, используемые для решения поставленных задач 16](#_Toc152964562)

[4 Проектирование функциональных возможностей программы 17](#_Toc152964564)

[4.1 Подключение к базе данных 17](#_Toc152964565)

[4.2 Регистрация и авторизация пользователей 18](#_Toc152964566)

[4.3 Управление пользователями 21](#_Toc152964567)

[4.4 Взаимодействие с сущностями приложения 22](#_Toc152964568)

[4.5 Общее описание системы, руководство пользователя, описание базы данных. 23](#_Toc152964569)

[5. Проектирование разрабатываемой базы данных программного   
 обеспечения 25](#_Toc152964570)

[5.1 Общая структура программы 25](#_Toc152964571)

[Заключение 26](#_Toc152964574)

[Список литературных источников 27](#_Toc152964575)

[Приложение А (обязательное) Листинг программного кода 28](#_Toc152964576)

[Приложение Б (обязательное) Конечная схема базы данных 29](#_Toc152964577)

[Приложение В (обязательное) Ведомость курсового проекта 30](#_Toc152964580)

**ВВЕДЕНИЕ**

Современные торговые компании, занимающиеся продажей мужской бижутерии, сталкиваются с необходимостью координации множества процессов: от управления ассортиментом и учета продаж до анализа спроса и прогнозирования поставок. Эти процессы требуют быстрой и точной обработки больших объемов данных, а также эффективного взаимодействия между различными участниками торговых операций. Введение автоматизированной системы управления может существенно повысить эффективность работы компании, снизить вероятность ошибок и сократить временные затраты.

Целью данной курсовой работы является создание программного средства для организации работы торговой компании, занимающейся продажей мужской бижутерии, на языке Python. Программное обеспечение будет включать в себя базу данных для хранения и управления информацией об ассортименте товаров, продажах, поставщиках, покупателях и других элементах торгового процесса. Основная задача — облегчить администрирование и улучшить координацию на всех этапах торговой цепочки, что особенно важно для своевременного пополнения запасов и качественного обслуживания клиентов.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Определить и обосновать перечень ключевых сущностей и их взаимосвязи в рамках торговли мужской бижутерией.
2. Разработать структуру базы данных, охватывающую все необходимые аспекты работы торговой компании, такие как учет товаров, анализ спроса и управление поставками.
3. Реализовать механизмы взаимодействия с данными и их обработки в соответствии с требованиями компании.
4. Создать приложение на Python, использующее разработанную базу данных для централизованного управления торговыми процессами.

В ходе разработки программного средства будут применяться современные подходы к проектированию и технологии, что позволит создать надежное и безопасное приложение для организации работы торговой компании. Итоговая система должна упростить рабочие процессы, минимизировать влияние человеческого фактора и повысить качество обслуживания клиентов, способствуя улучшению эффективности торговых операций.

# **1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ**

## 1.1 Структура и архитектура вычислительной системы

PostgreSQL – это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных. С помощью PostgreSQL можно создавать, хранить базы данных и работать с данными с помощью запросов на языке SQL.

Одной из наиболее сильных сторон СУБД PostgreSQL является архитектура. Как и в случаях со многими коммерческими СУБД, PostgreSQL можно применять в среде клиент-сервер – это предоставляет множество преимуществ и пользователям, и разработчикам.

В основе PostgreSQL – серверный процесс базы данных, выполняемый на одном сервере. Доступ из приложений к данным базы PostgreSQL производится с помощью специального процесса базы данных. То есть клиентские программы не могут получать самостоятельный доступ к данным даже в том случае, если они функционируют на том же ПК, на котором осуществляется серверный процесс.пользователям, и разработчикам.

В основе PostgreSQL – серверный процесс базы данных, выполняемый на одном сервере. Доступ из приложений к данным базы PostgreSQL производится с помощью специального процесса базы данных. То есть клиентские программы не могут получать самостоятельный доступ к данным даже в том случае, если они функционируют на том же ПК, на котором осуществляется серверный процесс. [ <https://otus.ru/nest/post/1584/> ]

Типичная модель распределенного приложения СУБД PostgreSQL (рисунок 1.1):

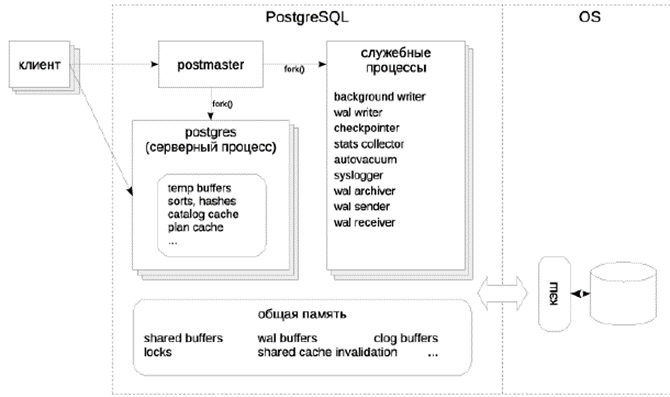


Рисунок 1.1 – Схема СУБД PostgreSQL

СУБД PostgreSQL ориентирована на протокол TCP/IP (локальная сеть либо Интернет), при этом каждый клиент соединён с главным серверным процессом БД (на рисунке 1.1 этот процесс обозначен Postmaster). Именно Postmaster создает новый серверный процесс специально в целях обслуживания запросов на доступ к данным определенного клиента. [ <https://habr.com/ru/companies/otus/articles/706346/> ]

Сервер PostgreSQL может обрабатывать несколько одновременных подключений от клиентов. Для этого он запускает новый процесс для каждого соединения. С этого момента клиент и новый серверный процесс обмениваются данными без вмешательства исходного процесса postgres. Таким образом, процесс сервера-супервизора всегда работает, ожидая клиентских подключений, в то время как клиентские и связанные серверные процессы приходят и уходят.[ <https://otus.ru/nest/post/1584/> ]

Данные, которыми управляет PostgreSQL, хранятся в базах данных. Один экземпляр PostgreSQL одновременно работает с несколькими базами, которые вместе называются кластером баз данных.

Каталог, в котором размещаются все файлы, относящиеся к кластеру, обычно называют словом PGDATA, по имени переменной окружения, указывающей на этот каталог.

При инициализации в PGDATA создаются три одинаковые базы данных (рисунок 1.2):

1 template0 используется, например, для восстановления из логической резервной копии или для создания базы в другой кодировке и никогда не должна меняться;

2 template1 служит шаблоном для всех остальных баз данных, которые может создать пользователь в этом кластере;

3 postgres представляет собой обычную базу данных, которую можно использовать по своему усмотрению.

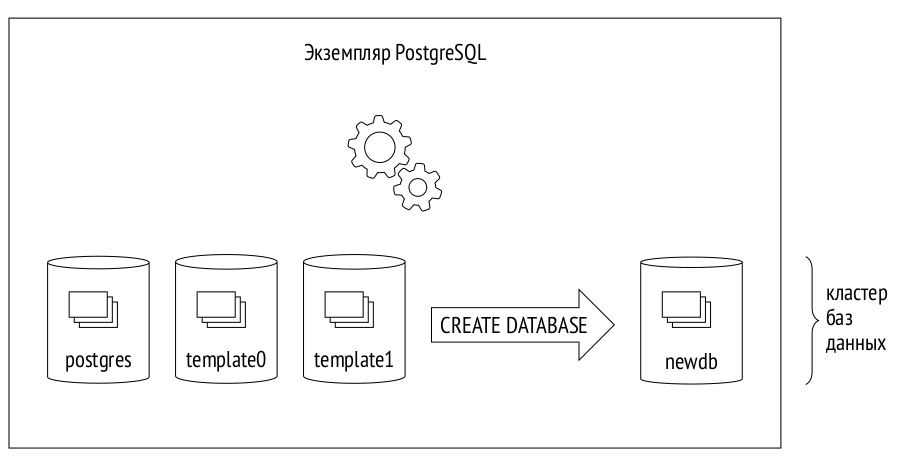


Рисунок 1.2 – Кластер PostgreSQL

Метаинформация обо всех объектах кластера (таких как таблицы, индексы, типы данных или функции) хранится в таблицах, относящихся к системному каталогу. В каждой базе данных имеется собственный набор таблиц (и представлений), описывающих объекты этой конкретной базы. Существует также несколько таблиц системного каталога, общих для всего кластера, которые не принадлежат какой-либо определенной базе данных и доступны в любой из них. [postgresql tutor]

## 1.2 История, версии и достоинства

Ранние версии системы были основаны на старой программе POSTGRES University, созданной университетом Беркли: так появилось название PostgreSQL. И сейчас СУБД иногда называют «Постгрес». Существуют сокращения PSQL и PgSQL – они тоже обозначают PostgreSQL.

По состоянию на июнь 2024 года PostgreSQL занимает четвертое место в общемировом рейтинге популярных СУБД (рисунок 1.3). [<https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/> ]

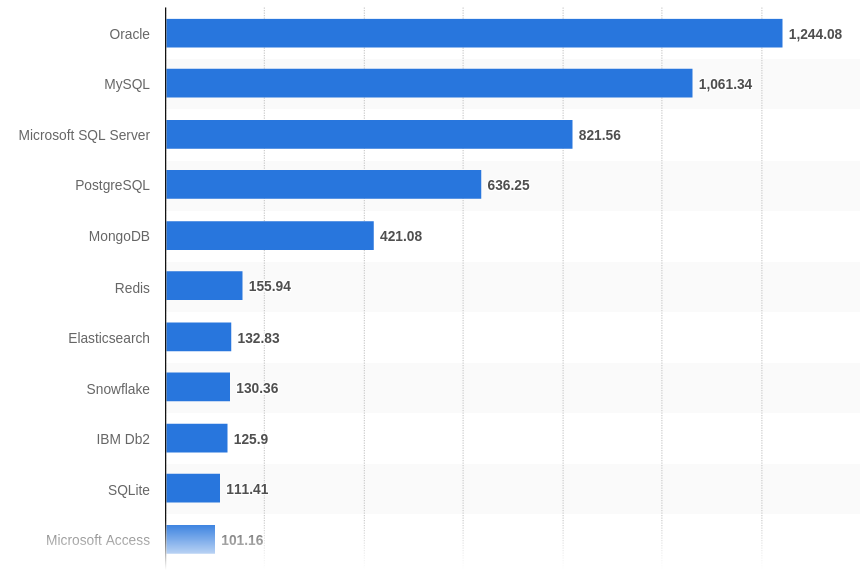


Рисунок 1.3 – Рейтинг популярности СУБД в июне 2024 года

У СУБД PostgreSQL много преимуществ, которые продолжают повышать ее популярность:

1 Любой специалист может бесплатно скачать, установить СУБД и сразу начать работу с базами данных.

2 PostgreSQL подходит для работы в любой операционной системе: Linux, macOS, Windows. Пользователь получает систему «из коробки» – чтобы установить и использовать программу, не нужны дополнительные инструменты.

3 PostgreSQL поддерживает много разных типов и структур данных, в том числе сетевые адреса, данные в текстовом формате JSON и геометрические данные для координат геопозиций. Все эти форматы можно хранить и обрабатывать в СУБД. Также при работе с PostgreSQL можно создавать собственные типы данных, их называют пользовательскими.

4 Размер базы данных в PostgreSQL не ограничен и зависит от того, сколько свободной памяти есть в месте хранения: на сервере, локальном компьютере или в облаке.

5 PostgreSQL реализует принципы ACID. Это четыре требования для надежной работы систем, которые обрабатывают данные в режиме реального времени. Если все требования выполняются, данные не будут теряться из-за технических ошибок или сбоев в работе оборудования.

6 PostgreSQL поддерживает все современные функции баз данных: оконные функции, вложенные транзакции, триггеры.

7 Хотя большинство операций в PostgreSQL и используют классический стандарт языка SQL, помимо него поддерживается и свой отдельный диалект, позволяющий еще комфортнее писать запросы.

8 Поддерживается репликация «из коробки». Репликация – это сохранение копии базы данных. Копия может находиться на другом сервере.

9 PostgreSQL позволяет быстро без потерь перенести данные из другой СУБД. [ <https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/> ]

10 Возможность одновременного доступа к базе с нескольких устройств. В СУБД реализована клиент-серверная архитектура, когда база данных хранится на сервере, а доступ к ней осуществляется с клиентских компьютеров. Для ситуаций, когда несколько человек одновременно модифицируют базу используется технология MVCC – Multiversion Concurrency Control, многоверсионное управление параллельным доступом.

Благодаря перечисленным выше преимуществам иногда PostgreSQL называют бесплатным аналогом Oracle Database. Обе системы адаптированы под большие проекты и высокую нагрузку. Но есть разница: они по-разному хранят данные, предоставляют разные инструменты и различаются возможностями. Важная особенность PostgreSQL в том, что эта система – feature-rich: так называют проекты с широким функционалом. [ <https://blog.skillfactory.ru/glossary/postgresql/> ]

## 1.3 Обоснование выбора вычислительной системы

PostgreSQL выбрана для разработки приложения для логистического центра, поскольку она сочетает в себе мощные функциональные возможности, надежность и высокую популярность среди реляционных СУБД. К июню 2024 года PostgreSQL занимает четвертое место в мировом рейтинге популярных СУБД, что подтверждает ее востребованность в сфере информационных технологий.

Одно из ключевых преимуществ PostgreSQL – это свободная лицензия, которая позволяет загружать и использовать систему без затрат. Она поддерживается на всех популярных операционных системах, включая Linux, macOS и Windows, что делает ее доступной для самых разных приложений и сред. Система предоставляет «из коробки» все необходимые инструменты, позволяя сразу приступить к работе с базами данных, не требуя дополнительных программ или надстроек.

PostgreSQL выделяется поддержкой различных типов и структур данных, что особенно важно для приложений, которые обрабатывают сложные данные. В дополнение к стандартным типам данных она позволяет работать с JSON, геометрическими данными и сетевыми адресами. В данном проекте это облегчает хранение медицинской информации и расширяет возможности ее обработки.

Другой важный аспект – масштабируемость PostgreSQL. Размер базы данных ограничен лишь объемом доступной памяти на сервере или в облаке, что позволяет хранить большие объемы медицинских данных и обеспечивает длительное использование системы без необходимости миграции.

Надежность системы обусловлена ее соответствием стандартам ACID, что важно для приложений, которые требуют целостности и безопасности данных. Медицинские данные должны быть защищены от потерь и технических сбоев, и PostgreSQL предлагает стабильную платформу для их обработки и хранения.

Также PostgreSQL реализует многоверсионное управление параллельным доступом (MVCC), что позволяет нескольким пользователям одновременно работать с базой данных без блокировки доступа. Это ключевое преимущество для логистического центра, где несколько сотрудников могут одновременно обращаться к базе для записи, просмотра и обновления данных пациентов.

Для торгового приложения, предназначенного для работы с мужской бижутерией, PostgreSQL также является подходящим выбором, поскольку она поддерживает мощные функции для обработки транзакций, что особенно важно для управления складскими операциями и отслеживания продаж. PostgreSQL позволяет эффективно обрабатывать запросы на обновление остатков товаров, регистрацию заказов и аналитические запросы, что способствует оперативному управлению ассортиментом и выполнению заказов в реальном времени.

Еще одним значимым фактором в пользу выбора PostgreSQL является ее возможность интеграции с внешними аналитическими и CRM-системами. Для торговых приложений это открывает доступ к аналитическим данным, которые помогут выявлять популярные товары, прогнозировать спрос и оптимизировать ассортимент. PostgreSQL также поддерживает расширяемую систему индексов и позволяет создавать специальные индексы для ускоренной обработки запросов, что особенно полезно для ускорения поиска по большим каталогам товаров.

Таким образом, PostgreSQL была выбрана для этого проекта благодаря ее широким функциональным возможностям, высокой надежности, поддержке разнообразных типов данных и оптимальной масштабируемости. Это делает PostgreSQL идеальной СУБД для логистическогоцентра, нуждающегося в эффективной и надежной системе управления данными.

# **2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

## 2.1 Структура и архитектура платформы

При выборе операционной системы для проекта было принято решение в пользу *Windows* 10 из-за ряда ключевых преимуществ и характеристик, которые делают ее идеальным выбором для нашего контекста.

Основные свойства *Windows* 10 определяются главными составляющими: 1 *Windows* 10 известна своей пользовательской дружелюбностью, что делает ее доступной для широкого круга пользователей. Интуитивно понятный интерфейс и простота в использовании позволяют пользователям быстро освоиться с системой, что особенно важно в нашем проекте, ориентированном на широкую аудиторию;

2 *Windows* 10 имеет высокий уровень совместимости с различным аппаратным обеспечением и программным обеспечением, что позволяет нам легко интегрировать необходимые приложения и компоненты. Кроме того, компания *Microsoft* предоставляет обширную техническую поддержку, что обеспечивает стабильную работу системы;

3 *Windows* 10 обладает мощными средствами безопасности, включая встроенный антивирус и брандмауэр, что обеспечивает защиту от вирусов, вредоносного ПО и других угроз. Регулярные обновления и патчи обеспечивают надежную защиту от новых угроз;

Ядро *Windows* 10 представляет следующую архитектуру:

1 Ядро *Windows* 10 включает в себя монолитные элементы, что обеспечивает эффективное взаимодействие между компонентами системы. Это способствует высокой производительности и быстрому отклику на пользовательские команды;

2 Одновременно ядро *Windows* 10 включает микроядерные принципы, разделяя некоторые функции на отдельные модули. Это облегчает обновления и обеспечивает более высокую стабильность системы. Например, драйверы и подсистемы могут работать в собственных привилегированных пространствах пользователя, увеличивая безопасность и стабильность системы;

3 Ядро операционной системы *Windows* 10 обладает универсальной совместимостью, позволяя без проблем работать на компьютерах, ноутбуках, планшетах, смартфонах и других устройствах, что делает его идеальным выбором для нашего многофункционального проекта. Кроме того, *Windows* 10 предоставляет обширную поддержку драйверов, облегчая подключение и использование различных устройств для наших пользователей.

В области безопасности, ядро *Windows* 10 включает в себя встроенные средства безопасности, такие как *Windows Defender*, обеспечивая базовую защиту от вирусов и вредоносных программ. Дополнительные функции, такие как *Secure Boot* и *Windows Hello*, значительно усиливают уровень безопасности системы. Кроме того, *Microsoft* регулярно выпускает обновления и патчи для ядра *Windows* 10, что обеспечивает быструю реакцию на новые угрозы безопасности и поддерживает систему в актуальном состоянии. Эта непрерывная поддержка обновлений гарантирует, что наша платформа будет защищена от последних угроз и обеспечит стабильную и безопасную среду для пользователей.

*Windows Hello* – это технология биометрической аутентификации, встроенная в операционную систему *Windows* 10. Она позволяет пользователям входить в систему и авторизовываться в приложениях и онлайн-сервисах, используя биометрические данные вместо традиционных паролей. *Windows Hello* поддерживает различные методы биометрической идентификации, включая сканеры отпечатков пальцев, камеры для распознавания лиц и даже датчики радужки глаза. Это обеспечивает удобство для пользователей, так как им больше не нужно помнить или вводить сложные пароли, что также усиливает безопасность системы.

*Secure Boot* – это функция, встроенная в *UEFI* (*Unified Extensible Firmware Interface*), представляющую собой программный механизм, который обеспечивает проверку цифровых подписей загрузочных компонентов перед их запуском на компьютере. Когда *Secure Boot* включен, он обеспечивает доверенное начальное загрузочное окружение, что помогает предотвратить внедрение вредоносных программ или изменение системных файлов в процессе загрузки операционной системы. Это усиливает безопасность компьютера, предотвращая попытки запуска неподписанных или подделанных загрузочных компонентов. *Secure Boot* стал обязательным для всех устройств, имеющих логотип «Сертифицировано для *Windows* 8 и выше», и помогает создать более безопасное окружение загрузки для операционных систем, включая *Windows* 10.

## 2.2 Выбор платформы для написания программы

В качестве языка программирования для написания программы используется Python. Python имеет несколько преимуществ для разработки приложения логистического центра:

1 Python отличается простой и понятной синтаксической структурой, что сокращает время на написание и поддержку кода. Это позволяет разработчику быстрее перейти от идеи к работающему продукту, а также легко поддерживать и обновлять приложение.

2 Python предлагает широкий выбор библиотек и фреймворков для разработки серверных приложений, таких как Django и FastAPI. Эти фреймворки упрощают настройку и структуру серверной части, обеспечивая готовые решения для обработки запросов, аутентификации и работы с базами данных. Например, Django ORM позволяет упростить взаимодействие с PostgreSQL, делая его интуитивно понятным и надежным.

3 Python имеет обширные возможности для работы с базами данных. Библиотеки, такие как psycopg2 и SQLAlchemy, облегчают взаимодействие с PostgreSQL, позволяя выполнять как простые, так и сложные запросы к базе данных. Эти библиотеки предоставляют как интерфейс ORM, так и позволяют отправлять базы чистые запросы через так называемый «raw SQL».

В качестве платформы для разработки был выбран PyCharm. Главная причина этому то, что PyCharm является одной из наиболее популярных и мощных IDE для разработки на Python, предоставляя разработчикам множество инструментов и удобств для повышения производительности и качества кода, и в том числе для удобной интеграции с базой данных и

# **3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ i ПРОГРАММНОГО ПРОДУКТА**

## 3.1 Обоснование необходимости разработки

## 

Современный рынок мужской бижутерии демонстрирует растущий спрос на разнообразные и уникальные аксессуары, а вместе с этим — необходимость удобного и эффективного управления продажами, ассортиментом и данными о клиентах. В условиях увеличения количества заказов и расширения ассортимента компаниям, занимающимся продажей мужской бижутерии, требуется надежный и функциональный инструмент для координации всех этапов торгового процесса. Без автоматизированного решения возрастает риск ошибок в учете товаров, задержек в обработке заказов, сложностей в анализе продаж и неудобства для клиентов. Это может приводить к потерям прибыли и снижению лояльности покупателей.

Разработка специализированного программного продукта для управления продажами и логистикой в сфере мужской бижутерии позволит автоматизировать рутинные процессы, повысить точность учета данных и упростить взаимодействие с клиентами. Такой продукт станет универсальным решением для ведения учета остатков, управления заказами, обработки клиентской информации и получения аналитических отчетов. Внедрение автоматизированной системы управления также позволит компании быстрее адаптироваться к изменениям на рынке, гибко корректируя ассортимент в зависимости от спроса и предпочтений клиентов.

Кроме того, использование программного продукта для анализа продаж и управления товарными запасами обеспечивает оптимизацию затрат на хранение и закупки. Это помогает компании экономить ресурсы и улучшать логистические процессы, избегая излишних запасов или, наоборот, дефицита товаров. В конечном итоге, автоматизация и улучшение точности учета способствуют увеличению продаж и удовлетворенности клиентов, создавая конкурентное преимущество на насыщенном рынке мужской бижутерии.

## 3.2 Технологии, используемые для решения поставленных задач

Для разработки программного средства, направленного на автоматизацию работы логистического центра, важно выбрать подходящие технологии программирования, которые обеспечат надежность, масштабируемость и производительность системы. Основные технологии, используемые в данном проекте, включают язык программирования Python, системы управления базами данных (СУБД), а также инструменты для создания графического интерфейса и разработки API. Рассмотрим ключевые технологии подробнее.

1 Язык программирования Python

Python был выбран как основной язык разработки благодаря его читаемости, простоте и поддержке большого числа библиотек, что ускоряет создание функционала. Python также обладает отличными возможностями для работы с базами данных, что необходимо для учета товаров и управления данными о продажах и клиентах. В проекте будут использоваться библиотеки, такие как:

Pandas для обработки и анализа данных, что полезно для анализа запасов, маршрутов и других логистических данных;

NumPy для быстрой обработки числовых данных;

2 Система управления базами данных (СУБД)

PostgreSQL используется для хранения данных о товарах, заказах, клиентах и поставщиках. Эта СУБД обеспечивает высокую производительность, поддерживает транзакции и гарантирует целостность данных, что важно для торговли. Поддержка сложных запросов позволяет эффективно работать с ассортиментом и анализировать данные по продажам. В данном проекте планируется использовать реляционную базу данных, такую как PostgreSQL или MySQL, которые обеспечивают:

Надежность и устойчивость к сбоям, что важно для непрерывной работы системы;

Высокую производительность при выполнении запросов к большому объему данных, что необходимо для обработки информации о запасах, маршрутах и транспорте;

Гибкость и масштабируемость, позволяющие легко наращивать объемы данных.

3 Система управления версиями Git.

Для управления версиями кода и отслеживания изменений используется Git. Это помогает хранить и контролировать весь процесс разработки, обеспечивая безопасность кода и возможность отката к предыдущим версиям при необходимости.

# **4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ iВОЗМОЖНОСТЕЙ ПРОГРАММЫ**

## 4.1 Подключение к базе данных

Подключение к базе данных — это один из ключевых этапов настройки приложения, поскольку оно позволяет приложению взаимодействовать с данными, такими как товары, заказы, отзывы, информация о пользователях и других сущностях, хранящихся в базе данных. В данном проекте используется база данных PostgreSQL, популярная СУБД, подходящая для обработки сложных данных и обеспечения надежности транзакций.

Для подключения к базе данных приложение использует библиотеку psycopg2, которая предоставляет интерфейс для взаимодействия с PostgreSQL на языке Python. Библиотека выполняет роль посредника, позволяя отправлять SQL-запросы к базе данных и обрабатывать возвращаемые результаты.

Подключение к базе данных требует следующих параметров:

* Адрес сервера — IP-адрес или локальный хост, на котором работает сервер базы данных. Для локальной работы может быть указан localhost.
* Порт — номер порта, на котором PostgreSQL ожидает подключения (по умолчанию это 5432, но в проекте используется порт 5433).
* Имя пользователя и пароль — учетные данные, используемые для аутентификации. Они подтверждают право приложения на выполнение операций с данными. В данном случае имя пользователя — postgres, а пароль — rootroot.
* Название базы данных — имя конкретной базы данных на сервере, с которой будет работать приложение. В проекте эта база данных называется chain\_shop.

После того как приложение успешно подключается к базе данных, оно создает объект подключения, который используется всеми функциями для выполнения запросов. Например, функции для добавления или удаления данных, а также для их выборки, используют это подключение для получения доступа к таблицам базы данных.

Важно, что после завершения работы приложение закрывает подключение. Это предотвращает утечку ресурсов и блокировку базы данных. Если приложение не закрывает соединение, оно продолжит потреблять память и ресурсы на сервере базы данных, что может привести к снижению производительности как приложения, так и самого сервера.

## 4.2 Регистрация и авторизация пользователей

Регистрация и авторизация пользователей являются фундаментальными компонентами любого программного продукта, обеспечивая доступ к функционалу системы только для уполномоченных лиц. В контексте консольного приложения для торговли мужской бижутерией эти процессы играют ключевую роль в управлении доступом к различным функциям, основанным на ролях пользователей, таких как клиенты, сотрудники и администраторы.

Цель регистрации — предоставить новым пользователям возможность создать учетную запись в системе, что позволит им взаимодействовать с приложением, совершать покупки, оставлять отзывы и использовать другие доступные функции.

Процесс регистрации включает следующие шаги:

1 Сбор данных пользователя:

При регистрации пользователь вводит необходимые данные, такие как имя, фамилия, номер телефона, адрес и пароль. Эти данные являются основными для идентификации пользователя и обеспечения его доступа к системе.

2 Валидация введенных данных:

* Имя и фамилия:

Проверяется наличие хотя бы одного символа, чтобы гарантировать корректность введенной информации.

* Номер телефона:

Проверяется соответствие формату, например, телефон должен начинаться с кода страны и содержать определенное количество цифр (+375XXXXXXXXX).

* Пароль:

Проверяется наличие минимального количества цифр или других

критериев безопасности, чтобы обеспечить защиту учетной записи от несанкционированного доступа.

3 Создание учетной записи:

После успешной валидации введенных данных информация сохраняется в базе данных. Для этого используются две таблицы:

* Users, которая содержит общую информацию о пользователях, включая их роли (клиент, сотрудник, администратор).
* Clients, которая хранит специфическую информацию о клиентах, такую как адрес доставки.

4 Назначение роли:

По умолчанию новым зарегистрированным пользователям присваивается роль "клиент" (Role\_Id = 2). Это определяет доступный функционал и права внутри системы.

5 Автоматический вход в систему:

После успешной регистрации пользователь автоматически авторизуется в системе, что позволяет ему сразу начать использовать доступные функции без необходимости повторного ввода учетных данных.

Преимущества процесса регистрации заключаются в удобстве для пользователей, безопасности и организации данных. Простая и интуитивно понятная форма регистрации снижает барьеры для новых клиентов, что упрощает первый опыт работы с приложением. Защита учетных записей достигается за счет валидации данных и строгих критериев паролей, которые предотвращают несанкционированный доступ. Структурированное хранение информации в базе данных упрощает управление учетными записями и взаимодействие с пользовательскими данными.

Цель авторизации — предоставить безопасный доступ к системе, подтверждая личность пользователя и обеспечивая ему права и функции в зависимости от его роли. Процесс авторизации начинается с ввода учетных данных: пользователь вводит свои уникальные логин и пароль. После этого система проверяет подлинность, сверяя данные с записями в базе. Проверка включает поиск пользователя по имени и сопоставление пароля. Важно отметить, что в проекте пароли хранятся в открытом виде, что является потенциальным риском для безопасности; для повышения защиты рекомендуется использовать хеширование паролей перед их сохранением.

При успешной авторизации устанавливается сессия пользователя, в которой система фиксирует состояние входа и сохраняет идентификатор текущего пользователя. Это позволяет системе отслеживать активные сессии и предоставлять доступ к нужным функциям в зависимости от роли пользователя. Если учетные данные введены неверно, система уведомляет пользователя об ошибке и предлагает повторить попытку, что предотвращает несанкционированный доступ и повышает безопасность.

Роли пользователей определяют доступный функционал приложения. Клиенты могут просматривать ассортимент товаров, оформлять заказы, оставлять отзывы и просматривать историю заказов. Сотрудники, помимо возможностей клиентов, могут управлять товарами, производителями и заказами, а также просматривать информацию о клиентах и поставщиках. Администраторы имеют полный доступ к системе, включая управление пользователями (добавление и удаление сотрудников), просмотр логов, управление отзывами и выполнение других административных функций.

Безопасность авторизации достигается за счет валидации данных и разграничения прав доступа. Проверка формата вводимых данных (например, номера телефона и пароля) предотвращает ввод некорректной или вредоносной информации. Роли и права доступа позволяют ограничить доступ к определенным функциям, что способствует предотвращению несанкционированных действий. Возможность выхода из системы (logout) позволяет пользователям завершать сессию безопасным образом, защищая учетные данные при использовании общего или публичного компьютера.

Процессы регистрации и авторизации пользователей обеспечивают безопасный и организованный доступ к функционалу консольного приложения для торговли мужской бижутерией. Благодаря тщательно продуманной системе ролей и проверке учетных данных, приложение гарантирует, что только авторизованные пользователи могут выполнять определенные действия, что способствует надежности и безопасности всей системы.

## 4.3 Управление пользователями

Управление пользователями в приложении — это система, которая позволяет администратору и другим уполномоченным ролям эффективно контролировать доступ, назначать права и поддерживать актуальность данных пользователей. В приложении для торговли мужской бижутерией управление пользователями необходимо для организации функций, связанных с назначением ролей, добавлением и удалением сотрудников и клиентов, а также с учетом особенностей доступа к данным.

Основные функции управления пользователями включают регистрацию новых пользователей, назначение ролей и удаление записей. В процессе регистрации каждому пользователю по умолчанию присваивается роль клиента, но администраторы имеют право изменять роль пользователя, назначая его сотрудником или администратором, в зависимости от потребностей компании. Назначение роли определяет уровень доступа пользователя к функциям приложения: клиенты могут оформлять заказы и оставлять отзывы, сотрудники дополнительно могут управлять товарами и заказами, а администраторы имеют полный доступ ко всем функциям, включая управление пользователями.

Для добавления сотрудников используется функция add\_employee, доступная только администраторам. При добавлении сотрудника администратор вводит данные пользователя и назначает его роль, зарплату, рабочее время и должности. Информация о сотруднике добавляется в соответствующие таблицы базы данных. Это позволяет разграничить функционал между различными категориями сотрудников и более гибко управлять организацией рабочего процесса.

Удаление пользователей также является важной функцией управления, поскольку оно позволяет администратору поддерживать актуальность данных и избегать переполнения системы неактуальной информацией. В зависимости от роли удаляемого пользователя система предоставляет возможность удаления записи из таблицы клиентов (delete\_client) или сотрудников (delete\_employee). Эти функции также обеспечивают защиту от случайного удаления критически важных данных и предотвращают несанкционированные действия.

## 4.4 Взаимодействие с сущностями приложения

Взаимодействие с сущностями приложения является ключевым аспектом разработки баз данных, так как оно определяет, как пользователи и другие системы будут взаимодействовать с данными. В контексте приложения, использующего реляционную базу данных, сущности могут представлять различные объекты, такие как клиенты, заказы, товары и производители. Каждая из этих сущностей имеет свои атрибуты и связи с другими сущностями, что формирует структуру базы данных.

При проектировании взаимодействия с сущностями необходимо учитывать несколько важных аспектов. Во-первых, каждое приложение должно обеспечивать четкий интерфейс для работы с данными. Это может включать функции для создания, чтения, обновления и удаления (CRUD) записей в базе данных. Например, для работы с сущностью "Товар" могут быть реализованы функции для добавления нового товара, получения информации о существующих товарах, обновления их цены или удаления из базы данных.

Во-вторых, необходимо реализовать механизмы валидации данных при взаимодействии с сущностями. Это поможет избежать ошибок и гарантировать целостность данных. Например, при добавлении нового клиента следует проверять корректность введенных данных, таких как телефонный номер или адрес электронной почты. Валидация может быть реализована через регулярные выражения или специальные функции проверки.

В-третьих, важно учитывать безопасность взаимодействия с сущностями. Это включает в себя аутентификацию пользователей и авторизацию их действий. Например, только администраторы могут добавлять или удалять товары, в то время как клиенты могут только делать заказы и оставлять отзывы. Реализация таких механизмов требует разработки соответствующих функций и логики на уровне приложения.

Кроме того, следует обратить внимание на производительность запросов к базе данных. Оптимизация запросов и использование индексов могут значительно ускорить взаимодействие с большими объемами данных. Например, при поиске товара по имени можно использовать индекс на поле имени для быстрого доступа к записям.

## 4.5 Общее описание системы, руководство пользователя, описание базы данных

Общее описание системы включает в себя описание функциональности и структуры базы данных, а также руководство пользователя, которое поможет пользователям эффективно взаимодействовать с системой. Данная база данных предназначена для управления информацией о производителях, цепочках, пользователях и заказах в контексте бизнеса, связанного с продажей ювелирных изделий.

Система состоит из нескольких ключевых сущностей, каждая из которых имеет свою таблицу в базе данных. Основные таблицы включают:

1 Producers, которая  хранит информацию о производителях, включая их название и страну.

2 Users, управляет данными пользователей системы, включая их имена, телефоны и роли.

3 Clients, представляет клиентов, связывая их с пользователями и их адресами.

4 Orders,  фиксирует заказы клиентов, включая дату создания, общую стоимость и количество товаров.

5 Reviews, позволяет клиентам оставлять отзывы о цепочках.

6 Employees, хранит информацию о сотрудниках и их зарплатах.

Каждая таблица имеет свои уникальные идентификаторы и связи с другими таблицами через внешние ключи. Это обеспечивает целостность данных и позволяет легко выполнять запросы для получения необходимой информации.

Руководство пользователя охватывает основные операции, которые могут выполнять пользователи системы. Пользователи могут регистрироваться в системе, создавать заказы на цепочки, оставлять отзывы и просматривать историю своих заказов. Для администраторов предусмотрены дополнительные функции управления пользователями и товарами. Каждое действие в системе сопровождается соответствующими проверками для обеспечения безопасности и целостности данных.

Описание базы данных включает в себя структуру таблиц с указанием типов данных для каждого поля. Например, таблица **Users** содержит поля для имени, фамилии, телефона и пароля, где телефон должен соответствовать определенному формату. Важно также отметить наличие триггеров и хранимых процедур для автоматизации некоторых процессов, таких как проверка существования клиента перед созданием заказа или автоматическое увеличение зарплаты сотрудников при получении положительных отзывов.

# **5. ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

## 5.1 Общая структура программы

Проектирование разрабатываемой базы данных программного обеспечения является важным этапом, который включает в себя создание общей структуры программы, определение сущностей, их атрибутов и взаимосвязей. В данном контексте база данных предназначена для управления информацией о производителях, цепочках, пользователях и заказах в сфере ювелирных изделий.

Общая структура программы включает в себя несколько ключевых компонентов. Во-первых, это **пользовательский интерфейс**, который обеспечивает взаимодействие с пользователями системы. Интерфейс должен быть интуитивно понятным и удобным, позволяя пользователям легко выполнять основные операции, такие как регистрация, создание заказов и оставление отзывов.

Во-вторых, **серверная логика** отвечает за обработку запросов от пользовательского интерфейса и взаимодействие с базой данных. Серверная часть должна включать бизнес-логику, реализующую правила работы с данными, такие как валидация вводимых данных и управление правами доступа пользователей. Это может быть реализовано через использование языков программирования, таких как Python или Java, а также фреймворков для создания веб-приложений.

В-третьих, **база данных** представляет собой хранилище информации, организованное в виде таблиц. Каждая таблица соответствует определенной сущности и содержит атрибуты, которые описывают свойства этой сущности. Например, таблица "Producers" хранит информацию о производителях ювелирных изделий, включая их название и страну. Таблица "Chains" содержит данные о цепочках, такие как название, дата создания и цена. Связи между таблицами обеспечивают целостность данных и позволяют выполнять сложные запросы для извлечения необходимой информации.

Также важным компонентом является **система управления базами данных (СУБД)**. Выбор СУБД влияет на производительность приложения и его масштабируемость. В данном случае предпочтение может быть отдано реляционным СУБД, таким как PostgreSQL или MySQL, которые обеспечивают надежное хранение данных и поддержку сложных запросов.

Кроме того, необходимо учитывать аспекты **безопасности** системы. Это включает в себя аутентификацию пользователей и защиту данных от несанкционированного доступа. Реализация таких механизмов требует использования шифрования паролей и других методов защиты информации.

В заключение, проектирование базы данных программного обеспечения требует комплексного подхода к созданию структуры программы. Это включает в себя разработку удобного пользовательского интерфейса, реализацию серверной логики для обработки запросов и взаимодействия с базой данных, выбор подходящей СУБД и обеспечение безопасности системы. Все эти компоненты должны работать в гармонии для достижения эффективного управления данными и удовлетворения потребностей пользователей.

# **ЗАКЛЮЧЕНИЕ**

 Разработанная база данных обеспечивает эффективное управление информацией о производителях, бижутерии, пользователях и заказах, что является критически важным для успешного ведения бизнеса.

В процессе проектирования базы данных были определены основные сущности и их атрибуты, а также установлены взаимосвязи между ними. Это позволяет обеспечить целостность данных и упростить выполнение запросов на получение информации. Реализация таких таблиц, как Producers, Chains, Users, Orders и Reviews, создает основу для эффективного хранения и обработки данных.

Кроме того, внедрение триггеров и хранимых процедур значительно улучшает функциональность системы. Триггеры позволяют автоматизировать определенные процессы, такие как проверка существования клиента перед созданием заказа или автоматическое добавление записи о доставке. Хранимые процедуры обеспечивают удобный интерфейс для выполнения сложных операций с данными, таких как добавление новых пользователей или обновление цен на цепочки.

Система также учитывает аспекты безопасности и производительности. Реализованные механизмы аутентификации пользователей и проверки прав доступа защищают данные от несанкционированного доступа. Оптимизация запросов и использование индексов позволяют ускорить обработку больших объемов информации.

В заключение, разработанная база данных представляет собой мощный инструмент для управления бизнес-процессами в области ювелирной торговли. Она обеспечивает надежное хранение данных, удобный доступ к информации и высокую степень автоматизации процессов. Это позволяет бизнесу эффективно реагировать на потребности клиентов и поддерживать конкурентоспособность на рынке.

# **СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ**

[1] "NVIDIA CUDA Toolkit Documentation." Официальный ресурс NVIDIA. [Электронный ресурс]. – Режим доступа: https://docs.nvidia.com/cuda/; Дата доступа: 28.09.2022.

[2] Маслов А.В. "PostgreSQL. Настольная книга администратора баз данных." – Москва: Питер, 2019. – 512 с.

[3] "PostgreSQL Documentation." Официальный сайт PostgreSQL. [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/; Дата доступа: 28.09.2022..

[4] Лутц М. "Изучаем Python. Программирование на примерах." – 5-е изд. – Москва: ДМК Пресс, 2021. – 864 с.

[5] Дронов В.В. "PostgreSQL. Основы программирования и администрирования." – Москва: БХВ-Петербург, 2020. – 416 с.

[6] "Django Documentation." Официальный сайт Django. [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/; Дата доступа: 28.09.2022.

[7] Грибков А.Н., Соловьев А.А. "Разработка веб-приложений на Python и Django." – Москва: Питер, 2022. – 320 с.

[8] Beazley D.M., Jones B.K. "Python Cookbook: Recipes for Mastering Python 3." – 3rd Edition. – O’Reilly Media, 2018. – 706 p.

[9] "SQLAlchemy Documentation." Официальный сайт SQLAlchemy. [Электронный ресурс]. – Режим доступа: https://docs.sqlalchemy.org/; Дата доступа: 28.09.2022.

[10] "PostgreSQL and Python Integration." Официальный сайт Psycopg. [Электронный ресурс]. – Режим доступа: https://www.psycopg.org/docs/; Дата доступа: 28.09.2022.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Исходный код**

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Функциональная схема**

# 

# **ПРИЛОЖЕНИЕ Д**

**(обязательное)**

**Ведомость курсового проекта**

# 

# 