Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ОРГАНИЗАЦИИ РАБОТЫ ЛОГИСТИЧЕСКОГОЦЕНТРА**

БГУИР КП 1-40 04 01 073 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | Е. В. Ласевич |
| Руководитель |  | А. В. Давыдчик |
|  |  |  |

Минск 2024

СОДЕРЖАНИЕ

[Введение 4](#__RefHeading___Toc12078_602624272)

[1 Архитектура вычислительной системы 5](#__RefHeading___Toc11247_602624272)

[1.1 Структура и архитектура вычислительной системы 5](#__RefHeading___Toc11249_602624272)

[1.2 История, версии и достоинства 7](#__RefHeading___Toc11251_602624272)

[1.3 Обоснование выбора вычислительной системы 9](#__RefHeading___Toc11253_602624272)

[2 Платформа программного обеспечения 11](#__RefHeading___Toc11259_602624272)

[2.1 Выбор операционной системы 11](#__RefHeading___Toc11261_602624272)

[2.2 Выбор платформы для написания программы 12](#__RefHeading___Toc11263_602624272)

[3 Теоретическое обоснование разработки программного продукта 13](#__RefHeading___Toc11265_602624272)

[3.1 Обоснование необходимости разработки 13](#__RefHeading___Toc11267_602624272)

[3.2 Технологии программирования, используемые для решения](#__RefHeading___Toc11269_602624272) [поставленных задач 13](#__RefHeading___Toc11271_602624272)

[4 Проектирование функциональных](#__RefHeading___Toc11277_602624272) [возможностей программы 14](#__RefHeading___Toc11279_602624272)

[4.1 Подключение к базе данных 14](#__RefHeading___Toc11281_602624272)

[4.2 Регистрация и авторизация пользователей 14](#__RefHeading___Toc11283_602624272)

[4.3 Управление пользователями 14](#__RefHeading___Toc11285_602624272)

[4.4 Взаимодействие с сущностями приложения 14](#__RefHeading___Toc11287_602624272)

[4.5 Общее описание системы 14](#__RefHeading___Toc11289_602624272)

[4.6 Руководство пользователя 14](#__RefHeading___Toc11291_602624272)

5 Проектирование разрабатываемой базы данных программного   
обеспечения 15

[5.1 Разработка информационной модели 15](#__RefHeading___Toc11281_6026242721)

[5.2 ER-диаграмма базы данных 15](#__RefHeading___Toc11283_6026242721)

[5.3 Оптимизация структуры разработанной базы данных 15](#__RefHeading___Toc11285_6026242721)

[5.4 Описание базы данных 15](#__RefHeading___Toc11287_6026242721)

[Заключение 16](#__RefHeading___Toc11307_602624272)

[Список литературных источников 17](#__RefHeading___Toc11309_602624272)

[Приложение А](#__RefHeading___Toc11311_602624272) [(обязательное)](#__RefHeading___Toc11313_602624272) [Листинг программного кода 18](#__RefHeading___Toc11315_602624272)

[Приложение Б](#__RefHeading___Toc11317_602624272) [(обязательное)](#__RefHeading___Toc11319_602624272) [Конечная схема базы данных 19](#__RefHeading___Toc11321_602624272)

[Приложение В](#__RefHeading___Toc11333_602624272) [(обязательное)](#__RefHeading___Toc11335_602624272) [Ведомость курсового проекта 20](#__RefHeading___Toc11337_602624272)

# **ВВЕДЕНИЕ**

Современные логистические центры ежедневно сталкиваются с необходимостью координации множества процессов: от управления запасами и складским учетом до планирования маршрутов и отслеживания перемещения грузов. Эти процессы требуют быстрой и точной обработки больших объемов данных, а также эффективного взаимодействия между различными участниками логистических операций. Введение автоматизированной системы управления может существенно повысить эффективность работы центра, снизить вероятность ошибок и сократить временные затраты.

Целью данной курсовой работы является создание программного средства для организации работы логистического центра на языке Python. Программное обеспечение будет включать в себя базу данных для хранения и управления информацией о товарных запасах, маршрутах, водителях, транспорте и других элементах логистического процесса. Основная задача — облегчить администрирование и улучшить координацию на всех этапах логистической цепочки, что особенно важно для своевременного и качественного выполнения поставок.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Определить и обосновать перечень ключевых сущностей и их взаимосвязи в рамках логистической деятельности.
2. Разработать структуру базы данных, охватывающую все необходимые аспекты работы логистического центра, такие как учет товаров, планирование маршрутов и управление транспортом.
3. Реализовать механизмы взаимодействия с данными и их обработки в соответствии с требованиями центра.
4. Создать приложение на Python, использующее разработанную базу данных для централизованного управления процессами.
5. Спроектировать и реализовать графический интерфейс для удобного взаимодействия с системой пользователями логистического центра.

В ходе разработки программного средства будут применяться современные подходы к проектированию и технологиям, что позволит создать надежное и безопасное приложение для организации работы логистического центра. Итоговая система должна упростить рабочие процессы, минимизировать влияние человеческого фактора и повысить качество обслуживания клиентов, тем самым способствуя улучшению эффективности логистических 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):

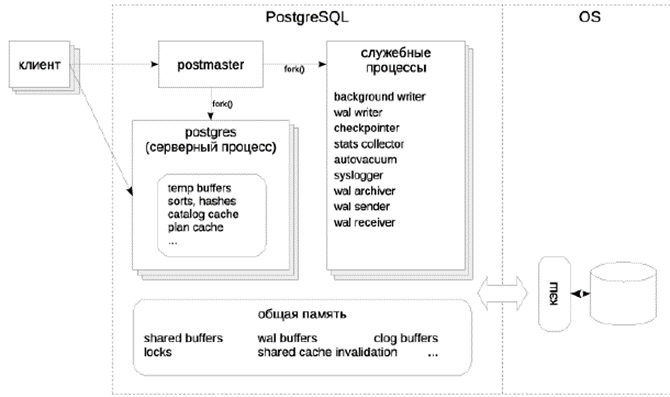


Рисунок 1.1 – Схема СУБД PostgreSQL

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.

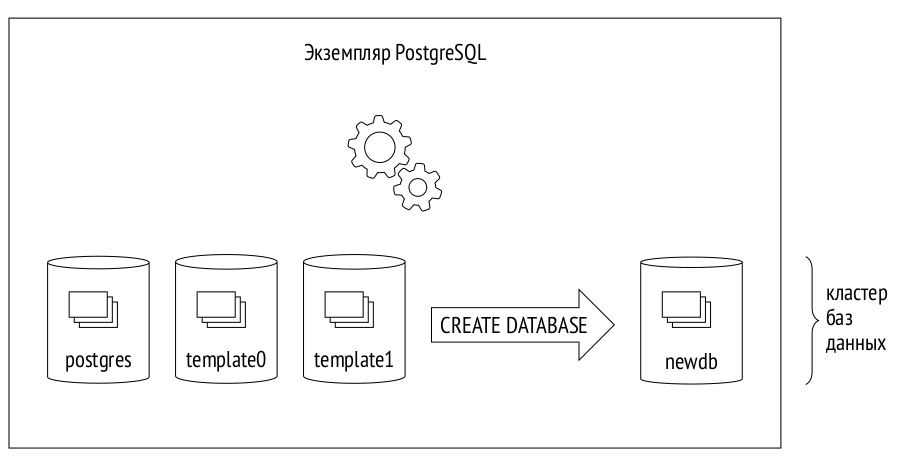


Рисунок 1.2 – Кластер PostgreSQL

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. [ книга postgresql изнутри ]

## 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 1.1). [ <https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/> ]

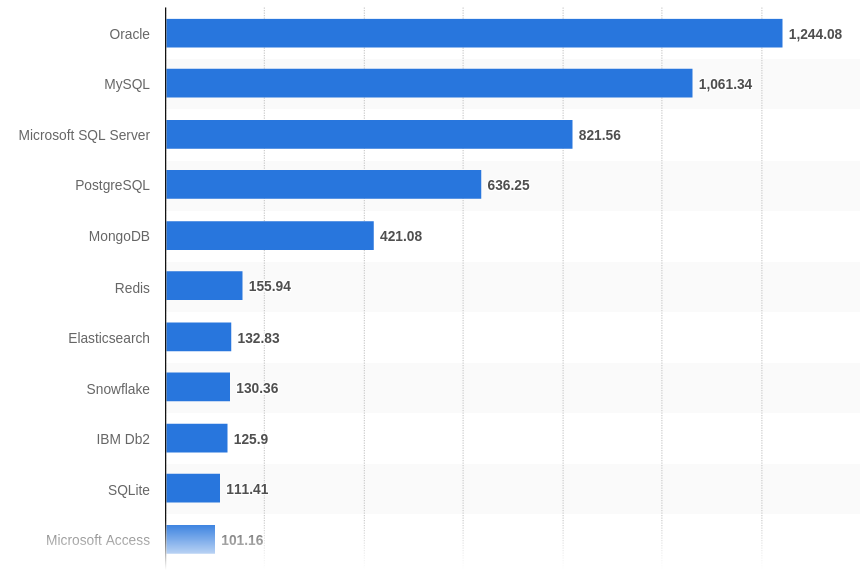


Рисунок 1.1 – Рейтинг популярности СУБД в июне 2024 года

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.

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 Выбор операционной системы

Linux – оптимальный выбор для приложения логистического центра, поскольку данная операционная система обеспечивает надежную, безопасную и гибкую платформу для серверных решений, особенно тех, которые требуют высокой производительности и стабильности.

Linux известен своей высокой стабильностью, особенно на серверных платформах. В медицинском центре работа с данными пациентов и организация процесса требует круглосуточной доступности и минимального времени простоя. Linux стабильно работает даже при значительных нагрузках, что делает его идеальным для приложений с высоким объемом запросов и большим количеством данных.

Логистические данные требуют особого уровня защиты, так как они содержат персональные данные и конфиденциальную информацию. Сильная встроенная система прав доступа (основанная на ролях) и возможность регулярных обновлений безопасности делают Linux безопасной ОС для хранения и управления конфиденциальными данными.

Также Linux позволяет оптимально использовать память и процессорные мощности, что особенно важно для приложений, обрабатывающих большое количество запросов и данных, как это требуется в медицинском центре. Благодаря тому, что Linux занимает минимум системных ресурсов, оставляя больше мощности для самого приложения и базы данных PostgreSQL, приложение будет работать быстрее и устойчивее.

Поскольку Linux является системой с открытым исходным кодом, она предоставляет экономически выгодное решение для логистического центра. Установка и использование Linux не требуют затрат на лицензии, что снижает затраты на внедрение и эксплуатацию.

Кроме того, Linux предлагает множество инструментов для автоматизации задач: скрипты на Bash, планировщики задач (например, cron) и возможности контейнеризации (например, Docker). Эти инструменты можно использовать для регулярного резервного копирования базы данных, обновлений системы и настройки мониторинга. Возможность автоматизации поможет логистическому центру обеспечить бесперебойную работу и уменьшить затраты на обслуживание системы.

Благодаря этим преимуществам, Linux является надежной и эффективной платформой для приложений логистического центра, требующих высокой безопасности, производительности и стабильности.

Таким образом, в качестве операционной системы для проведения сравнения используется Linux (дистрибутив Ubuntu).

## 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 отслеживания ее состояния.

# 3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

## 3.1 Обоснование необходимости разработки

В современных условиях логистические центры играют ключевую роль в цепочке поставок, обеспечивая управление запасами, распределение грузов и их своевременную доставку к конечным потребителям. Учитывая масштабы и сложность операций, возрастающую конкуренцию и высокие ожидания клиентов по скорости и точности выполнения заказов, логистические центры сталкиваются с рядом вызовов, таких как оптимизация маршрутов, снижение издержек и эффективное управление складскими запасами. Поэтому внедрение современных программных решений становится неотъемлемой частью их успешного функционирования.

Существующие системы зачастую обладают ограниченным функционалом или плохо адаптированы к специфическим требованиям конкретных центров. Многие из них требуют значительных финансовых вложений на покупку лицензий и техническое обслуживание, а также не всегда обладают достаточной гибкостью для быстрого внесения изменений под нужды растущего предприятия. В результате многие логистические компании сталкиваются с необходимостью внедрения специализированных программных средств, которые были бы не только экономичными, но и более гибкими, легко настраиваемыми и управляемыми.

Разработка программного продукта для логистического центра, ориентированного на автоматизацию основных процессов, позволит решить несколько важных задач:

1. Централизовать и автоматизировать управление складскими запасами, транспортными средствами и маршрутами;
2. Обеспечить оптимизацию использования ресурсов для снижения издержек;
3. Повысить скорость и точность выполнения заказов, что напрямую влияет на удовлетворенность клиентов;
4. Минимизировать риски, связанные с человеческим фактором, а также ошибки в учете и планировании.

Таким образом, разработка специализированного программного продукта позволит логистическим центрам не только повысить эффективность и снизить операционные затраты, но и создать конкурентное преимущество, благодаря точному и быстрому выполнению логистических операций.

## 3.2 Технологии программирования, 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 основным языком разработки благодаря его простоте, читаемости кода и обширной экосистеме библиотек и фреймворков. Python также широко используется в области автоматизации, обработки данных и создания веб-приложений, что делает его идеальным выбором для разработки логистического программного средства. 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;

SQLAlchemy для работы с базой данных, что упрощает взаимодействие с данными в объектно-ориентированном стиле.

2 Создание графического интерфейса

Для удобства взаимодействия с системой планируется создать графический интерфейс, с помощью которого пользователи смогут выполнять основные операции, такие как управление запасами, создание маршрутов и контроль перевозок. Возможные технологии для разработки интерфейса:

Tkinter – стандартная библиотека Python для создания простых графических интерфейсов;

PyQt или Kivy – библиотеки для создания более сложных интерфейсов с расширенной функциональностью и возможностью кроссплатформенной разработки;

3. Docker и системы виртуализации

Для развертывания и тестирования системы будет использоваться Docker, что позволит:

Обеспечить независимость приложения от окружения, упрощая развертывание и переносимость;

Управлять зависимостями и окружением проекта, создавая изолированные контейнеры для backend и базы данных;

Автоматизировать развёртывание, обеспечивая быстрый запуск и простоту тестирования.

4. Системы контроля версий и CI/CD

Для управления кодом и автоматизации процессов разработки и развертывания будут использоваться:

Git для контроля версий кода;

GitHub Actions или GitLab CI/CD для автоматизации процесса тестирования и развертывания, что ускорит выпуск обновлений и повысит надежность работы системы.

Использование этих технологий и инструментов позволит создать гибкую, производительную и устойчивую систему, которая удовлетворит потребности логистического центра в автоматизации процессов, обработке данных и обеспечении высокой точности операций.

.

# 4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ

# ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

Раздел "Проектирование функциональных возможностей программы" описывает ключевые компоненты системы, их назначение и логику взаимодействия. Это позволит создать четкую архитектуру, которая удовлетворяет потребности логистического центра и обеспечивает стабильную работу приложения.

#### 4.1 Подключение к базе данных

Программа будет использовать реляционную базу данных для хранения информации о пользователях, заказах, товарах, маршрутах и транспортных средствах. Подключение к базе данных будет реализовано с использованием библиотеки Cursor, что позволит эффективно управлять данными в объектно-ориентированном стиле. Конфигурация подключения будет включать указание адреса сервера, названия базы данных, учетных данных и параметров подключения. Основные функции модуля подключения:

1. Установка безопасного соединения с базой данных;
2. Обработка ошибок и управление соединениями;
3. Логирование запросов для диагностики.

#### 4.2 Регистрация и авторизация пользователей

Для обеспечения безопасности системы и контроля доступа к данным будет реализована регистрация и авторизация пользователей. Процесс регистрации включает валидацию данных и шифрование паролей перед их сохранением в базе данных:

1. Регистрация нового пользователя с проверкой уникальности данных;
2. Обработка ошибок (например, неверные данные или истекший токен).

#### 4.3 Управление пользователями

Для администрирования пользователей и контроля их доступа будет создан модуль управления пользователями, включающий:

1. Добавление, удаление и редактирование учетных записей;
2. Назначение ролей и прав доступа для управления уровнями доступа;
3. Просмотр статистики активности пользователя (например, последние действия, дата последнего входа).

#### 4.4 Взаимодействие с сущностями приложения

Программа должна обеспечивать доступ к основным сущностям логистического центра, таким как товары, заказы, транспортные средства и маршруты. Модуль взаимодействия с сущностями приложения включает следующие функции:

1. Управление запасами: добавление, редактирование и удаление информации о товарах;
2. Управление заказами: создание, изменение и отслеживание статуса заказов;
3. Управление транспортом: назначение транспортных средств на маршруты, отслеживание их состояния;
4. Создание и оптимизация маршрутов доставки с учетом текущих данных о заказах и местоположении транспортных средств.

#### 4.5 Общее описание системы

Система предназначена для автоматизации процессов логистического центра, включая управление складом, планирование маршрутов и отслеживание доставки. Программа взаимодействует с пользователями через графический интерфейс, поддерживая регистрацию и авторизацию, а также различные роли, такие как менеджеры, операторы и администраторы. Архитектура системы строится по принципу клиент-серверного приложения, где backend отвечает за обработку данных и взаимодействие с базой данных, а frontend предоставляет интерфейс для взаимодействия с пользователями.

#### 4.6 Руководство пользователя

Руководство пользователя описывает основные функции программы и порядок работы с системой. Включает инструкции по регистрации, авторизации, управлению сущностями и настройке системы. Подробные шаги будут описаны в отдельных разделах, например:

1. Пошаговое руководство по работе с товарами и заказами;
2. Описание процесса создания и оптимизации маршрутов;
3. Инструкции по управлению учетными записями и настройке прав доступа.

**5 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Для эффективного управления данными в логистическом центре была разработана реляционная база данных, включающая ключевые сущности и их взаимосвязи. Разработанная база данных позволяет учитывать данные о клиентах, заказах, товарах, складах, поставщиках и других объектах, связанных с операционной деятельностью центра. В результате разработки создана целостная и оптимизированная структура данных, способствующая ускорению обработки данных и упрощению работы с ними.

Таким образом, созданная база данных обеспечивает возможность эффективного управления всеми аспектами логистического процесса, улучшая организацию и обработку данных для повышения качества обслуживания клиентов.

**5.1 Разработка информационной модели**

Разработка информационной модели включала анализ предметной области и определение ключевых сущностей и атрибутов, которые необходимы для ведения работы логистического центра. В ходе проектирования были выделены следующие основные сущности:

1. Customers (Клиенты): хранит данные о клиентах, такие как имя, фамилия, контактные данные и ID заказа.
2. Orders (Заказы): содержит информацию о заказах, включая пункт выдачи, товары, статус и стоимость.
3. Goods (Товары): хранит данные о товарах, включая цену, количество, описание, ID склада и ID поставщика.
4. Warehouses (Склады): содержит информацию о складах, включая название и адрес.
5. Suppliers (Поставщики): включает данные о поставщиках, такие как имя, телефон и электронная почта.
6. Transport (Транспорт): хранит данные о транспортных средствах, которые принадлежат поставщикам.
7. Reviews (Отзывы): хранит отзывы клиентов о товарах, включая текст отзыва и рейтинг.
8. PickupPoints (Пункты выдачи): содержит информацию о пунктах выдачи заказов, включая название и адрес.

Эта модель позволяет эффективно управлять данными о клиентах, заказах и товарах, обеспечивая полноту и целостность данных.

**5.2 ER-диаграмма базы данных**

ER-диаграмма базы данных представляет визуализацию сущностей и связей между ними. (рисунок 5.1).

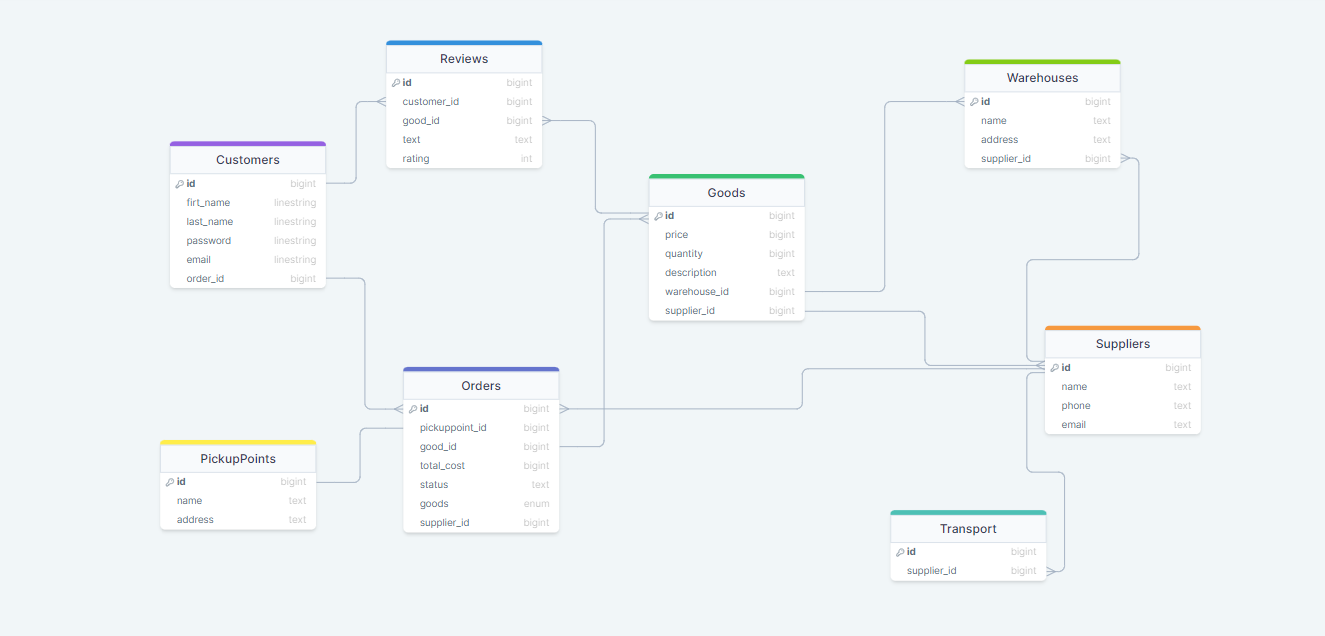


Рисунок 5.1 – ER-диаграмма логистического центра

Основные связи в базе данных включают:

1. Связь между клиентами и отзывами (каждый клиент может оставить несколько отзывов).
2. Связь между заказами и пунктами выдачи (заказ может быть назначен к конкретному пункту выдачи).
3. Связь между товарами и складами, где каждый товар имеет ссылку на определённый склад, а также на поставщика.
4. Связь между заказами и товарами, что позволяет указать, какие товары входят в заказ.
5. Связь между транспортом и поставщиками, которая показывает, какой транспорт принадлежит конкретному поставщику.

**5.3 Оптимизация структуры разработанной базы данных**

Оптимизация структуры базы данных включала следующие шаги:

1. Нормализация: была проведена нормализация данных, чтобы минимизировать избыточность и устранить аномалии обновления. Каждая сущность была разработана с минимальным количеством избыточных данных, что упрощает её дальнейшее использование и обновление.
2. Использование индексов: были добавлены индексы к ключевым полям, которые часто используются в поисковых запросах (например, customer\_id и good\_id в таблице Reviews), что позволило ускорить операции выборки.
3. Оптимизация связей: связи между таблицами были оптимизированы таким образом, чтобы избежать избыточности и улучшить целостность данных.

Эти шаги позволили создать базу данных, которая обеспечивает высокую производительность и целостность данных, необходимую для операционной деятельности логистического центра.

**5.4 Описание базы данных**

Описание основных таблиц базы данных:

1. Customers: содержит информацию о клиентах логистического центра. Основные поля — first\_name, last\_name, email, password, order\_id.
2. Orders: информация о заказах, включая такие поля, как pickup\_id, good\_id, status, cost, goods.
3. Goods: хранит данные о товарах, включая price (цена), quantity (количество), description (описание), warehouse\_id, supplier\_id.
4. Warehouses: таблица с данными о складах, включая name (название) и address (адрес).
5. Suppliers: информация о поставщиках, включая name (имя), phone (телефон), email.
6. Transport: содержит информацию о транспортных средствах поставщиков. Основное поле — supplier\_id.
7. Reviews: отзывы клиентов о товарах, включающие customer\_id, good\_id, text (текст отзыва), rating.
8. PickupPoints: таблица с информацией о пунктах выдачи, включая name и address.

Эта структура обеспечивает полноценное и упорядоченное хранение данных, что позволяет эффективно организовать процессы логистического центра, а также автоматизировать обработку и анализ данных для улучшения бизнес-процессов.

# ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта … .

Таким образом, … .

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] Архитектура Zen: сколько поколений продержится главная технология AMD [Электронный ресурс]. – Режим доступа: https://club.dns-shop.ru/blog/t-100-protsessoryi/61416-arhitektura-zen-skolko-pokolenii-proderjitsya-glavnaya-tehnologi/ – Дата доступа: 01.10.2023.

[2] Поколения процессоров AMD Ryzen [Электронный ресурс]. – Режим доступа: https://te4h.ru/pokoleniya-protsessorov-amd-ryzen – Дата доступа: 01.10.2023.

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Листинг программного кода

To be continued

# ПРИЛОЖЕНИЕ Б

# (обязательное)

# Конечная схема базы данных

# ПРИЛОЖЕНИЕ В

# (обязательное)

# Ведомость курсового проекта