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

**Разработка клиент-серверного приложения для мониторинга работы удаленных сотрудников**

БГУИР КП 1-40 04 01 031 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А. А. Талай |
| 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**

В условиях современного рынка, характеризующегося высокой конкуренцией и активным развитием технологий, бизнесу становится необходимым эффективное управление данными и автоматизация процессов. Это особенно актуально для предприятий, где требуется гибкая система управления продуктами, заказами, пользователями и отзывами. Разработка информационных систем для таких целей становится важной задачей, так как позволяет упростить работу сотрудников, облегчить взаимодействие с клиентами и повысить общую эффективность компании.

Целью данной курсовой работы является создание информационной системы для магазина фруктов, которая позволит автоматизировать основные процессы управления продуктами, заказами и пользователями, а также обеспечит удобное взаимодействие между клиентами и сотрудниками. В рамках системы будут реализованы функциональные модули для авторизации и управления пользователями, администрирования фруктов и производителей, контроля отзывов, а также ведения журнала действий пользователей.

Система предусматривает ролевую модель, где права доступа дифференцируются в зависимости от роли пользователя: клиент, сотрудник или администратор. Клиенты смогут просматривать каталог фруктов, заказывать продукцию и оставлять отзывы. Сотрудники будут управлять ассортиментом и производителями, а администраторы получат полный контроль над всеми моделями данных, включая возможность работы с отзывами. В системе также предусмотрен модуль журналирования действий, который позволит отслеживать и анализировать основные операции, такие как покупки и оставленные отзывы, что обеспечит прозрачность и безопасность работы.

В процессе разработки информационной системы потребуется спроектировать структуру базы данных, включающую таблицы для хранения данных о фруктах, производителях, заказах, клиентах, отзывах и пользователях. Данная структура обеспечит удобное хранение и быстрое извлечение данных, необходимых для выполнения различных функций. Кроме того, система должна включать интерфейс для пользователей с разными правами доступа, чтобы каждый мог быстро находить нужную информацию и выполнять необходимые операции в зависимости от своей роли.

Создаваемая система будет способствовать повышению уровня сервиса, удобству взаимодействия с клиентами и оптимизации процессов управления.

# 1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

## 1.1 Структура и архитектура вычислительной системы

*PostgreSQL* – это объектно-реляционная система управления базами данных (ORDBMS), являющаяся одной из самых продвинутых и популярных открытых СУБД в мире. Она распространяется с открытым исходным кодом и представляет собой мощную альтернативу коммерческим решениям для работы с базами данных. *PostgreSQL* позволяет создавать и управлять базами данных, используя *SQL* -запросы для работы с данными.

Одним из ключевых преимуществ *PostgreSQL* является её архитектура. Как и многие коммерческие СУБД, *PostgreSQL* поддерживает клиент-серверную модель, что даёт разработчикам и пользователям значительные преимущества. В основе *PostgreSQL* лежит серверный процесс, выполняющийся на одном сервере и обеспечивающий централизованный доступ к базе данных.

Клиентские приложения взаимодействуют с базой данных *PostgreSQL* через этот серверный процесс и не могут напрямую обращаться к данным, даже если находятся на том же устройстве, что и сервер. Это позволяет поддерживать высокую безопасность и целостность данных, контролируя все запросы и взаимодействия через централизованный процесс базы данных.

Типичная модель распределенного приложения СУБД *PostgreSQL* (рисунок 1.1):

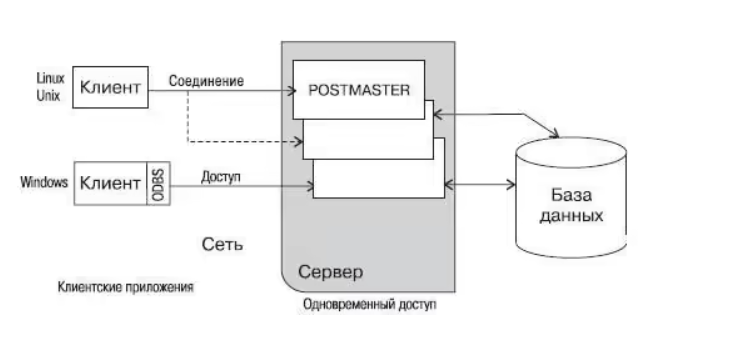


Рисунок 1.1 – Схема СУБД PostgreSQL

СУБД *PostgreSQL* ориентирована на протокол *TCP/IP* (локальная сеть либо Интернет), при этом каждый клиент соединён с главным серверным процессом БД. Именно *Postmaster* создает новый серверный процесс специально в целях обслуживания запросов на доступ к данным определенного клиента. 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.

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;

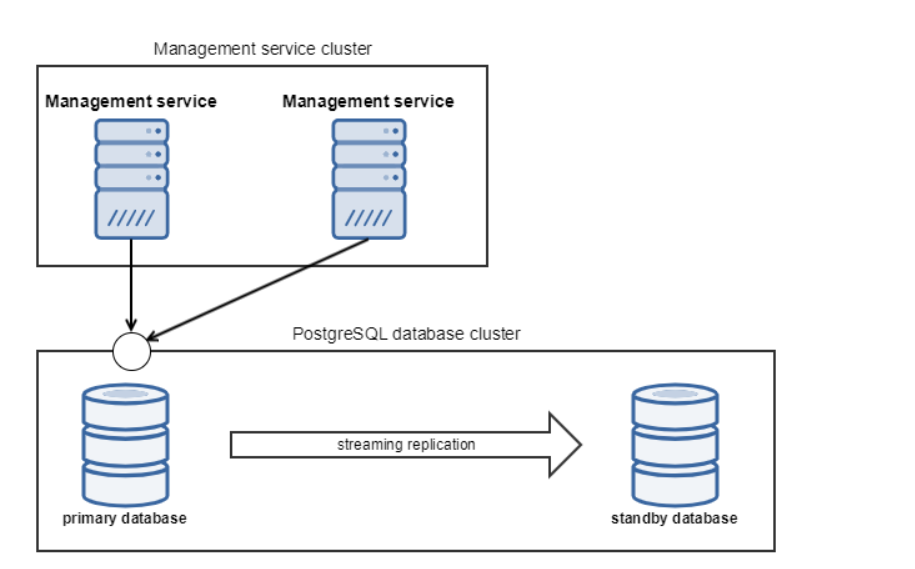


Рисунок 1.2 – Кластер PostgreSQL

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.

## 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*.

*PostgreSQL* занимает второе место в общемировом рейтинге популярных СУБД (рисунок 1.3).

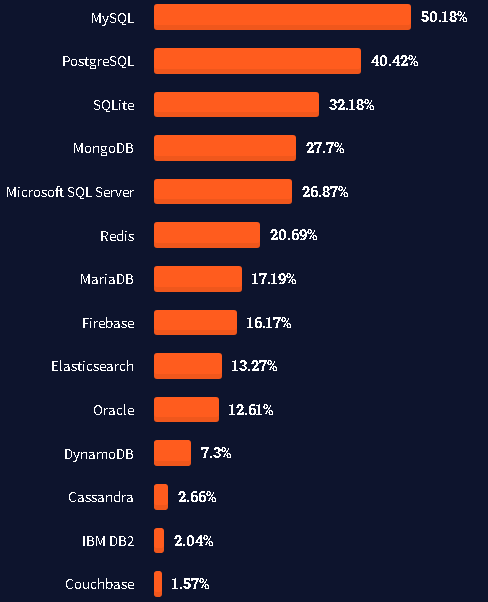


Рисунок 1.3 – Рейтинг популярности СУБД

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.

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.

## 1.3 Обоснование выбора вычислительной системы

Выбор *PostgreSQL* для создания данной информационной системы основан на её многочисленных преимуществах, которые делают её одной из наиболее подходящих баз данных как для средних, так и для крупных проектов. *PostgreSQL* является объектно-реляционной системой управления базами данных (ORDBMS) с открытым исходным кодом, что предоставляет огромные возможности для адаптации и настройки под конкретные требования. Это особенно ценно для проектов с уникальными бизнес-логиками, поскольку *PostgreSQL* позволяет глубоко кастомизировать систему и интегрировать её с другими сервисами, при этом не требуя затрат на лицензии, что существенно снижает общие издержки.

Одной из ключевых особенностей *PostgreSQL* является её способность работать с объектами, что позволяет хранить сложные структуры данных, такие как массивы, *JSON*-данные и даже пользовательские типы данных. Это даёт возможность создавать многоуровневые иерархии данных и работать с ними максимально эффективно, что немаловажно для приложений, где необходимо организовать взаимодействие между различными сущностями, как, например, в системе управления заказами, пользователями и производителями.

Кроме того, *PostgreSQL* поддерживает *ACID*-транзакции, что обеспечивает надёжную целостность данных даже в случае сбоев или ошибок. Это свойство делает *PostgreSQL* идеальным решением для систем, требующих строгого контроля за изменением информации, например, в системах учёта заказов, отзывов или финансовых данных, где каждая транзакция должна быть надёжно зафиксирована и доступна для анализа. Высокий уровень согласованности данных также важен для ролевых систем, в которых требуется точное разграничение прав доступа и гарантированное сохранение изменений в реальном времени.

Архитектура *PostgreSQL* построена по принципу клиент-серверного взаимодействия, что предоставляет множество преимуществ. Во-первых, это позволяет централизованно управлять доступом к данным и гарантировать безопасность и контроль над каждым клиентским запросом. Все клиентские программы и приложения взаимодействуют с базой данных исключительно через специальный процесс, что исключает возможность несанкционированного доступа и упрощает администрирование системы. Во-вторых, модель клиент-сервер даёт возможность легко масштабировать приложение, подключать множество клиентов и распределять нагрузку между серверами, что повышает устойчивость системы и её производительность.

Помимо всего прочего, *PostgreSQL* поддерживает большинство стандартов *SQL* и может быть интегрирована с различными языками программирования и фреймворками, что делает её удобной и гибкой для разработчиков. Это обеспечивает лёгкость интеграции *PostgreSQL* с существующими и новыми приложениями, минимизируя сложность перехода на другую СУБД или разработки новой инфраструктуры. Благодаря поддержке разных языков запросов и гибкой настройке индексов, *PostgreSQL* позволяет оптимизировать работу с данными, что особенно важно в высоконагруженных системах, где время отклика и производительность имеют решающее значение.

Дополнительно стоит отметить, что сообщество *PostgreSQL* активно поддерживает и развивает систему, что позволяет регулярно получать обновления и доступ к новым функциям, улучшая безопасность, производительность и удобство работы с базой данных. Благодаря этому *PostgreSQL* остаётся актуальной на фоне изменений в ИТ-отрасли и продолжает успешно конкурировать с коммерческими СУБД. Большое количество расширений и плагинов, доступных для *PostgreSQL*, позволяет добавлять специализированные функции, такие как работа с географическими данными (*PostGIS*) или анализ временных рядов, что делает её универсальным решением для широкого спектра приложений. Все эти факторы вместе делают *PostgreSQL* оптимальным выбором для создания комплексной информационной системы, которая будет надёжно работать в долгосрочной перспективе и эффективно поддерживать все бизнес-процессы.

Таким образом, *PostgreSQL* представляет собой полноценное решение для создания надёжной, масштабируемой и гибкой информационной системы. Она сочетает в себе стабильность и высокую производительность коммерческих баз данных с доступностью и свободой, характерной для *open*-*source* решений.

# 2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**2.1** **Структура и архитектура платформы**

**2.1.1** Выбранная платформа

Выбор платформы разработки является критическим шагом в процессе создания программного обеспечения. Эта платформа определяет инструменты, языки программирования и технологии, которые будут использоваться в проекте. Возможны различные варианты, включая языки программирования, такие как Java, Python, JavaScript, а также фреймворки и инструменты разработки, такие как .NET, Ruby on Rails, Django и другие.

Выбор платформы может быть обусловлен несколькими факторами, включая:

1 Требования проекта: Некоторые платформы могут быть более подходящими для конкретных типов приложений или задач.

2 Опыт команды разработчиков: Уровень знаний и опыта команды разработчиков может влиять на выбор платформы.

3 Совместимость с другими системами: Важно учитывать совместимость выбранной платформы с существующими системами и технологиями.

4 Сообщество и поддержка: Наличие активного сообщества и хорошей поддержки со стороны разработчиков может значительно облегчить процесс разработки и решение проблем.

Важно провести тщательный анализ требований проекта и основных критериев выбора, чтобы выбрать наиболее подходящую платформу, которая обеспечит успешное выполнение задач и достижение поставленных целей.

**2.1.2** Язык программирования

Язык программирования – формальный язык, предназначенный   
для записи компьютерных программ. Язык программирования определяет набор лексических, синтактических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель   
под ее управлением.

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами и прочее. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего,   
для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

**2.1.3** Интегрированная среда разработки

Интегрированная среда разработки (*IDE*) играет ключевую роль в процессе создания программного обеспечения. Важно выбрать такую среду, которая соответствует потребностям разработчика и характеру проекта. *IDE* обеспечивает:

Удобный и интуитивно понятный интерфейс, облегчающий работу с проектами и файлами кода.

Интеграцию с другими инструментами разработки, такими как системы управления версиями, средства сборки и автоматизации тестирования.

Широкий выбор расширений и плагинов для расширения функциональности *IDE* в соответствии с потребностями проекта и предпочтениями разработчика.

Совместимость с различными операционными системами, обеспечивая возможность работы на *Windows*, *macOS* и *Linux*.

В итоге выбор *IDE* зависит от конкретных задач и предпочтений разработчика. Важно подобрать такую среду разработки, которая обеспечит комфортную и эффективную работу над проектом на платформе C#.

**2.1.4** Операционная система

Современный компьютер организован в виде иерархии уровней, каждый из которых добавляет определенные функции к нижележащему уровню.

С точки зрения программиста, операционная система – это программа, добавляющая ряд команд и функций к командам и функциям, предлагаемым уровнем архитектуры команд. Обычно операционная система организуется программно, но нет никаких веских причин, по которым ее,   
как микропрограммы, нельзя было реализовать.

Все команды уровня операционной системы доступны для прикладных программистов. Это практически все команды более низкого уровня, а также новые команды, добавленные операционной системой. Новые команды называются системными вызовами. Они вызывают предопределенную службу операционной системы, в частности одну из ее команд. Например, типичный системный вызов может читать данные из файла.

**2.2 История, версии и достоинства**

**2.2.1** Язык программирования *Python*

*Python* — это мощный инструмент для разработки программного обеспечения, подходящий для широкого круга задач: от простых скриптов до комплексных веб-приложений и научных исследований. Ниже представлены ключевые причины выбора платформы *Python*:

1. Универсальность: *Python* поддерживает множество парадигм программирования, таких как объектно-ориентированное, функциональное и процедурное программирование. Это делает его подходящим для самых разных областей, включая веб-разработку, анализ данных, машинное обучение и автоматизацию задач.

2. Обширная экосистема и поддержка библиотек: *Python* обладает богатой коллекцией библиотек и фреймворков, таких как *Django* и *Flask* для веб-разработки, *NumPy* и *Pandas* для работы с данными, а также *TensorFlow* и *PyTorch* для машинного обучения. Это позволяет решать широкий спектр задач без необходимости разработки с нуля.

3. Простота и читабельность: Синтаксис *Python* прост и понятен, что делает его удобным для начинающих разработчиков, а также облегчает поддержку и развитие кода. Код на *Python* легко читается, что способствует быстрой разработке и совместной работе в команде.

4. Поддержка современных технологий и сообществом: *Python* активно развивается, обеспечивая поддержку актуальных технологий и трендов, таких как искусственный интеллект и облачные вычисления. Широкое сообщество Python разрабатывает и поддерживает тысячи библиотек, что делает его одним из наиболее адаптивных и гибких языков.

5. Кроссплатформенность: Программы на *Python* могут выполняться на разных операционных системах, включая *Windows*, *macOS* и *Linux*. Это делает его удобным выбором для кроссплатформенных приложений и интеграционных решений.

Выбор *Python* обеспечивает разработчикам мощный набор инструментов для создания современных приложений с высокой производительностью, кроссплатформенностью и поддержкой актуальных технологий.

**2.2.2** Среда разработки *Visual Studio*

*Visual Studio* - это интегрированная среда разработки (*IDE*) от *Microsoft*, предназначенная для создания разнообразных приложений, включая настольные, веб-приложения, мобильные приложения и игры.

Многофункциональность: *Visual Studio* предоставляет широкий набор инструментов для разработки, включая редактор кода, отладчик, дизайнер форм, инструменты для тестирования и т.д.

Поддержка других языков и платформ: В *Visual Studio* есть поддержка различных языков программирования и платформ, включая *JavaScript, TypeScript, Python, C++,* и многие другие.

Инструменты для командной разработки: *Visual Studio* обеспечивает возможность совместной работы над проектами, включая систему контроля версий, возможность обмена кодом и совместного редактирования.

Обширная документация и сообщество: *Visual Studio* имеет обширную документацию, руководства и обучающие материалы, а также активное сообщество разработчиков, которые могут помочь в решении проблем и обмене опытом.

*Visual Studio* является одной из наиболее популярных и мощных сред разработки, обладающей всеми необходимыми инструментами для создания высококачественного программного обеспечения.

**2.2.3** Операционная система *Windows*

*Windows* обладает рядом значительных преимуществ и достоинств:

1 Популярность и совместимость: *Windows* является самой популярной операционной системой, что обеспечивает широкую совместимость   
с программным обеспечением и аппаратным обеспечением.

2 Игровая платформа: *Windows* является ведущей платформой для игр   
и обеспечивает доступ к широкому спектру видеоигр.

3 Простота использования: интерфейс *Windows* обычно считается более интуитивным и легким для освоения, особенно для новых пользователей.

4 Множество функций: *Windows* предоставляет множество функций, включая встроенные инструменты для работы с офисными приложениями, веб-браузерами и мультимедиа

5 Обновления и поддержка: *Microsoft* регулярно выпускает обновления и обеспечивает техническую поддержку для *Windows*, обеспечивая безопасность и стабильность системы.

6 Широкий спектр аппаратных устройств: *Windows* поддерживает множество устройств, включая настольные ПК, ноутбуки, планшеты   
и смартфоны.

Для разработки и запуска нашей утилиты мы выбрали операционную систему *Windows* 11.

Совместимость между *Python*, *Visual Studio* и Windows 11 обеспечивает безпроблемную разработку приложений на новой операционной системе компании *Microsoft*. Язык программирования *Python* является основным инструментом для создания широкого спектра приложений, а *Visual Studio* предоставляет разработчикам мощную интегрированную среду для написания, отладки и развертывания приложений.

*Windows* 11, как последняя версия операционной системы от *Microsoft*, обеспечивает поддержку разработки на *Python* и инструментов *Visual Studio*. Это означает, что разработчики могут использовать последние функции и возможности *Windows* 11 при создании своих приложений, таких как новый пользовательский интерфейс, улучшенная безопасность и поддержка новых технологий.

*Visual Studio*, в свою очередь, адаптирован для работы на *Windows* 11 и обеспечивает все необходимые инструменты и функции для разработки приложений на *Python*. Разработчики могут использовать мощные инструменты редактирования кода, отладки, а также интеграцию с системами контроля версий и другими сервисами разработки.

Таким образом, совместимость между *Python*, *Visual Studio* и *Windows* 11 обеспечивает разработчикам удобную и эффективную среду для создания современных приложений, соответствующих последним требованиям и возможностям операционной системы *Windows* 11.

**2.3 Обоснование выбора платформы**

Обоснование выбора платформы является ключевым этапом в процессе разработки программного обеспечения, поскольку от этого выбора зависят многие аспекты проекта, включая производительность, доступность, совместимость и т.д. Обоснования выбора платформы, связанных с использованием *Python* и *Visual Studio*:

1 Широкие возможности разработки: язык программирования *Python* предоставляют разработчикам широкие возможности для создания различных типов приложений, начиная от настольных и веб-приложений и заканчивая мобильными приложениями. Это обеспечивает гибкость и масштабируемость проекта.

2 Мощная среда разработки: *Visual Studio* является одной из наиболее мощных и продвинутых интегрированных сред разработки (*IDE*) для работы с языком *Python*. Она обеспечивает широкий набор инструментов, функций и возможностей, упрощающих разработку и обеспечивающих высокую производительность разработчиков.

3 Большое сообщество и поддержка: *Python* и *Visual Studio* имеют большое и активное сообщество разработчиков, что обеспечивает доступ к обширным ресурсам, обучающим материалам, форумам и блогам, где можно получить помощь, поддержку и советы по решению проблем.

4 Поддержка новейших технологий и платформ: *Python* и *Visual Studio* постоянно обновляются и развиваются, обеспечивая поддержку новейших технологий и платформ, что позволяет разработчикам быть в тренде и создавать современные приложения.

5 Интеграция с экосистемой *Microsoft*: Использование *Python* и *Visual Studio* в качестве основной платформы разработки обеспечивает интеграцию с другими продуктами и сервисами компании *Microsoft*, такими как *Azure*, *Office* 365, *Windows* и другими, что упрощает разработку, развертывание и масштабирование приложений.

6 Масштабируемость и поддержка облачных технологий: *Python* и *Visual* *Studio* отлично интегрируются с облачными сервисами, что упрощает создание масштабируемых приложений с высокой производительностью и позволяет эффективно использовать облачную инфраструктуру, например, *Microsoft* *Azure*, для хранения и обработки данных.

7 Простота тестирования и отладки: *Visual* *Studio* предоставляет мощные инструменты для тестирования и отладки, что делает разработку более контролируемой и предсказуемой. Наличие встроенных возможностей для тестирования, таких как *unittest* и *pytest* для *Python*, позволяет сократить время на выявление и исправление ошибок и значительно улучшить качество создаваемого ПО.

В целом, выбор платформы *Python* и *Visual Studio* обоснован широкими возможностями разработки, мощными инструментами, поддержкой сообщества и интеграцией с экосистемой *Microsoft*, что делает их отличным выбором для создания современных и высококачественных приложений.

# 3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

## 3.1 Обоснование необходимости разработки

В условиях современного рынка, где конкуренция между компаниями становится все более ожесточенной, а требования клиентов к качеству обслуживания растут, предприятиям необходимо активно внедрять автоматизацию и оптимизацию своих бизнес-процессов. Для этого важно иметь эффективную информационную систему, которая позволяет управлять основными аспектами бизнеса, такими как учет товаров, обработка заказов, управление клиентскими данными и взаимодействие с клиентами. Особенно это актуально для розничных магазинов, которые занимаются продажей товаров с высокой оборотностью, таких как продукты питания.

Разработка информационной системы для магазина фруктов является необходимостью, так как позволяет значительно упростить и ускорить процессы, связанные с управлением ассортиментом, заказами и отзывами клиентов. Без автоматизации таких операций бизнес сталкивается с рядом проблем: высокими трудозатратами на обработку заказов вручную, возможными ошибками в учете продукции, трудностями в отслеживании отзывов и взаимодействии с клиентами. Внедрение системы, которая будет интегрировать все эти функции, значительно повысит эффективность работы как сотрудников магазина, так и улучшит клиентский опыт.

Автоматизация процессов управления заказами, клиентами, отзывами и товарами способствует снижению числа ошибок, ускорению обслуживания клиентов и улучшению качества работы с данными. Это, в свою очередь, позволяет сотрудникам магазина сосредоточиться на более важных задачах, таких как повышение качества обслуживания, оптимизация ассортимента и улучшение взаимодействия с клиентами. Разработка системы с удобным интерфейсом, где каждый пользователь будет иметь доступ к функциям в зависимости от своей роли, позволит не только повысить безопасность данных, но и обеспечить более гибкую и эффективную работу всего коллектива магазина.

Таким образом, необходимость разработки данной информационной системы для магазина фруктов обусловлена желанием повысить уровень сервиса, улучшить взаимодействие с клиентами, оптимизировать бизнес-процессы и повысить общую эффективность компании.

## 3.2 Технологии программирования, используемые для решения

## поставленных задач

Для разработки программного средства, направленного на автоматизацию работы логистического центра, необходимо выбрать технологии программирования, которые обеспечат надежность, масштабируемость и производительность системы. В качестве основного языка разработки выбран *Python* благодаря своей простоте, читаемости кода и обширной экосистеме библиотек и фреймворков. *Python* 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 и *SQLAlchemy* для работы с базой данных, что упрощает взаимодействие с данными в объектно-ориентированном стиле.

Для удобства взаимодействия с системой будет создан графический интерфейс, который позволит пользователям выполнять основные операции, такие как управление запасами, создание маршрутов и контроль перевозок. В качестве технологии для разработки интерфейса был выбран *Tkinter*, стандартная библиотека *Python* для создания простых графических интерфейсов. *Tkinter* предоставляет достаточно функционала для создания удобных окон, форм и кнопок, что идеально подходит для решения задач в рамках данного проекта. Этот инструмент обеспечит простоту разработки и поддержку графического интерфейса, не требуя сложных дополнительных зависимостей и позволяя сосредоточиться на функциональности.

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, а также автоматизировать развертывание, обеспечивая быстрый запуск и простоту тестирования.

Для управления кодом и автоматизации процессов разработки и развертывания будут использоваться *Git* для контроля версий кода и *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.

**3.3 Связь архитектуры вычислительной системы с**

**разрабатываемым программным обеспечением**

Архитектура вычислительной системы играет ключевую роль в создании эффективного и надежного программного обеспечения, поскольку от ее структуры зависит не только производительность системы, но и способность приложения справляться с растущими требованиями бизнеса. В контексте разработки информационной системы для автоматизации работы логистического центра, архитектура вычислительной системы должна учитывать несколько факторов, включая нагрузку на систему, взаимодействие различных компонентов, возможность масштабирования и поддержку различных пользовательских ролей. В этом разделе рассматриваются основные аспекты связи архитектуры вычислительной системы с разрабатываемым программным обеспечением.

Программное обеспечение будет разрабатываться с использованием клиент-серверной архитектуры, где серверная часть будет заниматься обработкой данных, взаимодействием с базой данных, выполнением бизнес-логики и обеспечением функциональности для различных ролей пользователей (администраторы, сотрудники и клиенты). Сервер будет обрабатывать запросы клиентов через *API* и предоставлять необходимые данные в ответах, что обеспечит гибкость и расширяемость системы.

Для управления данными, а также хранения и извлечения информации о запасах, маршрутах, заказах и других элементах, будет использована реляционная база данных, что позволяет обеспечить структурированное хранение информации и быстрый доступ к ней. База данных будет поддерживать таблицы для каждого из объектов системы, таких как фрукты, пользователи, заказы, маршруты и т.д. Важно, что архитектура вычислительной системы должна обеспечить высокую производительность при работе с большими объемами данных, поддерживая быстрое выполнение запросов.

Для взаимодействия с пользователем будет разработан графический интерфейс, построенный с использованием *Tkinter*. Архитектура системы будет таким образом настроена, чтобы клиентские приложения могли отправлять запросы на сервер, а сервер, в свою очередь, возвращал пользователю результаты в удобном виде. Это обеспечит высокую степень разделения бизнес-логики и пользовательского интерфейса, что повысит модульность и упрощение сопровождения системы в дальнейшем.

Особое внимание будет уделено безопасности данных и управлению правами доступа, что также тесно связано с архитектурой вычислительной системы. Серверная часть будет иметь возможность аутентификации и авторизации пользователей, обеспечивая безопасность работы с системой. В зависимости от роли пользователя (клиент, сотрудник или администратор), система будет предоставлять различный уровень доступа, что важно для обеспечения корректной работы и защиты данных.

Масштабируемость системы также является важным аспектом, который требует внимательного подхода при проектировании архитектуры. Система должна быть готова к расширению, например, добавлению новых функций или увеличению объема данных, без существенного ухудшения производительности. Использование *Docker* для контейнеризации приложения и базы данных, а также настройки автоматического развертывания и тестирования с помощью *CI/CD* инструментов, позволит легко масштабировать систему и разворачивать её в различных средах.

Таким образом, архитектура вычислительной системы должна быть спроектирована таким образом, чтобы обеспечивать стабильную работу всех компонентов, поддержку разных типов пользователей, высокую производительность и безопасность данных. Связь архитектуры с разрабатываемым программным обеспечением заключается в том, что архитектура служит основой для реализации всех требуемых функциональных возможностей, обеспечивая стабильную работу системы в условиях роста и изменений.

# 4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ

# ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

## 4.1 Обоснование и описание функций программного обеспечения

Проектирование функциональных возможностей программы должно учитывать все основные процессы, которые происходят в рамках логистической системы, и описывать их с учетом конкретной архитектуры базы данных и требований пользователей. В данном случае разрабатываемое программное обеспечение включает в себя несколько ключевых функций, которые обеспечивают эффективное управление данными и поддерживают различные роли пользователей. Рассмотрим эти функции и их связи с базой данных и системой.

Авторизация пользователей является важной частью системы, поскольку она определяет доступность функционала в зависимости от роли пользователя. Для этого в базе данных предусмотрена таблица "Пользователи", которая хранит данные для аутентификации, такие как имя, фамилия, телефон и пароль. После успешной авторизации, система использует поле "ID роли" для определения прав доступа пользователя: клиент, сотрудник или администратор. Авторизованный пользователь получает возможность выполнять действия, ограниченные его ролью.

Для управления пользователями и их ролями используется стандартный функционал CRUD (создание, чтение, обновление, удаление). Сотрудники и администраторы могут управлять данными пользователей в базе, изменяя информацию в таблице "Пользователи". Администратор, имея полный доступ к базе, может добавлять новых пользователей, изменять их данные, а также удалять их, что обеспечивается через интерфейсы CRUD.

Ролевая система регулирует доступ к функционалу в зависимости от того, какой роли принадлежит пользователь. В таблице "Роли" хранится информация о типах ролей, таких как "Клиент", "Сотрудник", и "Администратор". Эти роли обеспечивают разделение прав доступа. Клиенты могут только просматривать товары и оставлять отзывы, сотрудники могут управлять фруктами и производителями, а администраторы имеют полный доступ к системе, включая возможность редактировать и удалять любые данные.

Система отслеживает действия пользователей, что позволяет анализировать поведение клиентов, а также вести журнал действий для безопасности и аудита. Журнал действий хранит информацию о таких действиях, как покупки и оставленные отзывы. Для этого в базе данных предусмотрены таблицы "Заказы" и "Отзывы", которые связываются с пользователями и фруктах через внешний ключ. Все данные о действиях пользователя, такие как дата заказа, сумма и отзывы, будут фиксироваться и доступны для анализа.

Клиенты могут просматривать информацию о сотрудниках через таблицу "Сотрудники", которая хранит информацию о каждом сотруднике, включая его зарплату и рабочее время. Также предполагается наличие функции поиска, позволяющей клиентам фильтровать сотрудников по различным параметрам.

Сотрудники имеют доступ к таблицам "Фрукты" и "Производители", что позволяет им управлять информацией о товарах и их производителях. С помощью функций CRUD сотрудники могут добавлять новые фрукты, редактировать существующие, а также удалять их из системы. Сотрудники также могут обновлять информацию о производителях, что отражается в таблице "Производители".

Администраторы имеют полный доступ к таблице "Отзывы", что позволяет им управлять отзывами пользователей. С помощью функций *CRUD* администратор может добавлять, изменять или удалять отзывы, оставленные клиентами. Это обеспечивает контроль над качеством данных и поддержание актуальной информации о товарах.

Таблица "Фрукты" хранит информацию о фруктах, включая их название, цену, срок годности и дату изготовления. Каждое фруктовое изделие связано с производителем через внешний ключ.

Таблица "Заказы" фиксирует заказы клиентов. Она хранит информацию о сумме заказа, количестве товара и связывает заказ с конкретным клиентом и фруктом.

В таблице "Клиенты" хранятся данные клиентов, включая их адрес и ссылку на пользователя. Каждый клиент может иметь один заказ или несколько заказов.

Таблица "Роли" хранит список всех возможных ролей пользователей в системе, таких как клиент, сотрудник и администратор.

Таблица "Отзывы" хранит текстовые отзывы клиентов о фруктах, а также оценку, которую оставил клиент. Эти данные помогают клиентам принимать решения при покупке.

Таблица "Сотрудники" хранит информацию о сотрудниках, включая их зарплату и рабочее время. Сотрудники могут управлять продуктами и заказами.

Таблица "Доставка" содержит информацию о доставке заказов, связывая заказы с датой доставки.

В таблице "Производители" хранится информация о производителях фруктов, включая страну и название компании.

Таблица "Пользователи" хранит основную информацию о пользователях системы, включая их имя, фамилию, телефон и пароль. Каждый пользователь связан с конкретной ролью через поле "*ID* Роли".

Таблица "Должности" содержит список должностей для сотрудников, таких как менеджер, продавец и т.д.

Неавторизованные пользователи могут просматривать каталог фруктов, отзывы о фруктах и список сотрудников. Также они могут пройти процедуру авторизации.

Клиенты могут совершать заказы, оставлять отзывы о фруктах и просматривать свою историю заказов. Авторизованные пользователи имеют доступ к функционалу, который позволяет им взаимодействовать с товарами.

Сотрудники могут выполнять все действия клиента, а также управлять данными о фруктах и производителях, используя функции *CRUD*.

Администраторы имеют полный доступ ко всем функциям системы, включая возможность *CRUD* операций со всеми моделями (пользователи, фрукты, производители, заказы, отзывы и т.д.).

Таким образом, функциональные возможности программы обеспечивают гибкость и расширяемость системы с учетом потребностей разных типов пользователей, при этом поддерживается эффективное управление данными с помощью архитектуры базы данных.

**5 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Для эффективного управления данными была разработана реляционная база данных, включающая ключевые сущности и их взаимосвязи. Разработанная база данных позволяет учитывать данные о клиентах, заказах, товарах, и других объектах, связанных с операционной деятельностью магазина. 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 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, необходимых для эффективного функционирования системы. В ходе проектирования были выделены следующие основные сущности:

1. Users (Пользователи): хранит информацию о пользователях системы, включая имя, фамилию, контактные данные и роль.

2. Orders (Заказы): содержит данные о заказах, включая дату создания, сумму заказа, количество товаров, информацию о клиенте и фрукте.

3. Fruits (Фрукты): хранит информацию о фруктах, включая название, дату производства, цену, срок годности и ID производителя.

4. Manufacturers (Производители): включает данные о производителях фруктов, такие как название компании и страна.

5. Reviews (Отзывы): содержит отзывы клиентов о фруктах, включая текст отзыва, оценку, а также связь с фруктом и клиентом.

6. Employees (Сотрудники): хранит данные о сотрудниках, включая зарплату, а также связь с пользователем и рабочим временем.

7. EmployeeWorkingTime (Рабочее время сотрудников): хранит информацию о рабочем времени сотрудников, включая дату и время начала и окончания работы.

8. Deliveries (Доставки): содержит информацию о доставках заказов, включая ID заказа и дату доставки.

Эта модель обеспечивает хранение данных о пользователях, заказах, фруктах, отзывах, сотрудниках и доставках, что позволяет эффективно управлять процессами в системе и поддерживать целостность и полноту данных.

**5.2 ER-диаграмма базы данных**

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

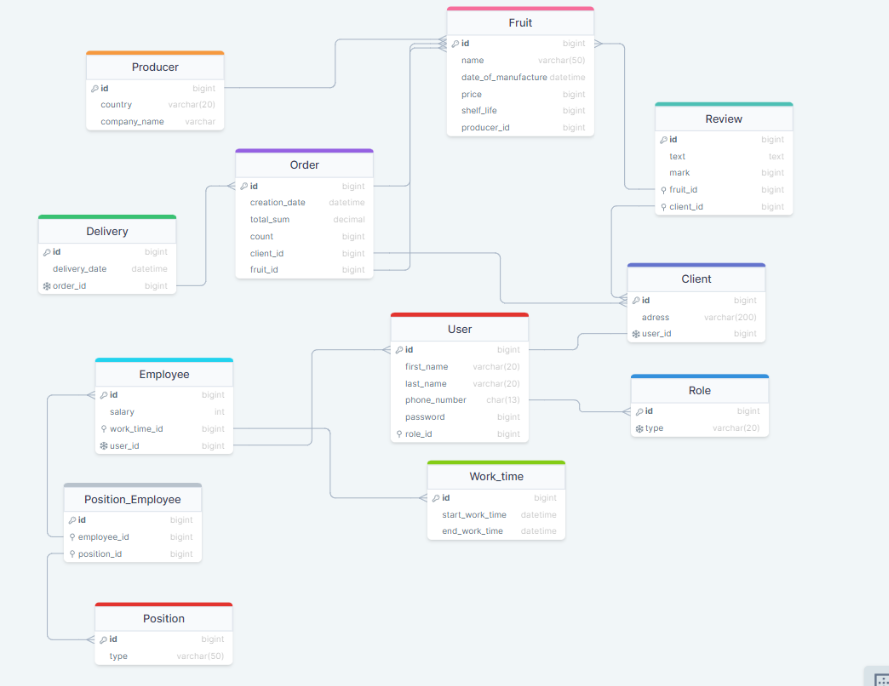


Рисунок 5.1 – ER-диаграмма магазина фруктов

**5.3 Оптимизация структуры разработанной базы данных**

Оптимизация структуры базы данных включала несколько ключевых этапов:

1. Нормализация данных: для минимизации избыточности и предотвращения аномалий при обновлениях была проведена нормализация. Каждая сущность была спроектирована с минимальным количеством избыточных данных, что способствует более эффективному использованию и обновлению информации в системе.

2. Использование индексов: для ускорения операций выборки были добавлены индексы на ключевые поля, которые часто используются в запросах, такие как `customer\_id` и `fruit\_id` в таблице отзывов. Это позволило значительно повысить скорость выполнения поисковых запросов.

3. Оптимизация связей: связи между таблицами были организованы с учетом минимизации избыточности, что позволило повысить целостность данных и улучшить общую производительность базы данных.

В результате этих шагов была создана база данных, которая обеспечивает высокую производительность, целостность и надежность, что критически важно для эффективного функционирования системы.

**5.4 Описание базы данных**

1. Users: содержит информацию о пользователях системы. Основные поля — `first\_name`, `last\_name`, `phone`, `password`, `role\_id`.

2. Roles: таблица с данными о ролях пользователей, включая такие поля, как `role\_name`.

3. Fruits: информация о фруктах, включая такие поля, как `name`, `manufacture\_date`, `price`, `expiry\_period`, `manufacturer\_id`.

4. Manufacturers: таблица с данными о производителях фруктов, включая `company\_name` и `country`.

5. Orders: таблица для хранения данных о заказах пользователей, включая поля `created\_at`, `total\_sum`, `quantity`, `user\_id`, `fruit\_id`.

6. Reviews: таблица с отзывами клиентов о фруктах, содержащая поля `text`, `rating`, `fruit\_id`, `user\_id`.

7. Employees: информация о сотрудниках системы, включая поля `salary`, `user\_id`.

8. EmployeeWorkingTime: таблица для хранения данных о рабочем времени сотрудников, включая `start\_time`, `end\_time`, `employee\_id`.

9. Deliveries: таблица для хранения данных о доставках заказов, включая поля `order\_id`, `delivery\_date`.

Эта структура обеспечивает хранение данных о пользователях, заказах, фруктах, отзывах и доставках, что позволяет эффективно управлять процессами в системе и автоматизировать обработку данных для улучшения бизнес-процессов.

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] Архитектура вычислительных систем [Электронный ресурс]: учебное пособие – Эл. изд. – Электрон. текстовые дан. (1 файл *pdf*: 77 с.). – Грейбо С.В., Новоселова Т.Е., Пронькин Н.Н., Семенычева И.Ф. 2019. – Режим доступа: <http://scipro.ru/conf/computerarchitecture.pdf> – Дата доступа: 30.09.2023

[2] Таненбаум, Э. Архитектура компьютера – 6-е изд. / Э. Таненбаум. СПб. : Питер, 2013. – 816 с.

[3] Петцольд, Ч. Код. Тайный язык информатики / Ч. Петцольд. – М. : Манн, Иванов и Фербер, 2021. – 448 с.

[4] Харрис, Д. М. Цифровая схемотехника и архитектура компьютера – 2-е изд. / Д. М. Харрис, С. Л. Харрис. – *NY* : *Elsevier Inc*, 2013. – 1662 c.

[5] Шитов, В. Н. *Windows* 10 : самый простой и понятный самоучитель / В. Шитов. – М. : Эксмо, 2023. – 464 с.

[6] Русинович, М. Внутреннее устройство *Windows* – 7-е изд. / М. Русинович [и др.]. – СПб. : Питер, 2018. – 944 с.

[7] Как работает *GPU* [Электронный ресурс]. – Режим доступа:  [https://coremission.net/gamedev/kak-rabotaet-gpu – Дата доступа: 30.09.2023](https://learn.microsoft.com/ru-ru/windows/win32/api%20–%20Дата%20доступа%2020.09.2023)

[8] Обзор процессора *Intel Core i*5*-*10300*H* [Электронный ресурс]. –https://askgeek.io/ru/cpus/Intel/Core-i5-10300H#google\_vignette – Дата доступа: 09.11.2023

[9] Обзор видеокарты *NVIDIA GeForce GTX* 3050 [Электронный ресурс]. – https://3dnews.ru/987707/obzor-nvidia-geforce-gtx-3050 – Дата доступа: 09.11.2023

[10] *SSD Micron* 2210 [Электронный ресурс]. – https://3dnews.ru/987707/obzor-nvidia-geforce-gtx-1650 – Дата доступа: 09.11.2023

[11] Попов, А. Администрирование *Windows* с помощью *WMI* и *WMIC*. / А. В. Попов, Е. А. Шикин. СПб. : БХВ-Петербург, 2004. – 752 с.

[12] *PyCharm* [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/help/pycharm/quick-start-guide –Дата доступа: 02.09.2023

[13] *Windows –* что это такое? [Электронный ресурс]. – Режим доступа: https://internet-lab.ru/windows\_os – Дата доступа: 02.09.2023

[14] Инструментарий управления *Windows*[Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/wmisdk/wmi-start-page> – Дата доступа: 14.11.2023

[15] *WMI* [Электронный ресурс]. – Режим доступа: <https://script-coding.com/WMI.html> – Дата доступа: 14.10.2023

[16] *Open Hardware Monitor* [Электронный ресурс]. – Режим доступа: <https://openhardwaremonitor.org/documentation/> – Дата доступа: 14.10.2023

[17] *Psutil* [Электронный ресурс]. – Режим доступа: <https://pypi.org/project/psutil/> – Дата доступа: 10.10.2023

[18] *GPUnit* [Электронный ресурс]. – Режим доступа: <https://github.com/williamyang1991/GP-UNIT> – Дата доступа: 10.10.2023

[19] *PyQt* [Электронный ресурс]. – Режим доступа: <https://wiki.python.org/moin/PyQt> – Дата доступа: 15.10.2023

[20] Получаем информацию о системе [Электронный ресурс]. – Режим доступа: [https://codeby.net/threads/poluchaem-informaciju-o-sisteme-s-pomoschju-python-chast-1.79797](https://codeby.net/threads/poluchaem-informaciju-o-sisteme-s-pomoschju-python-chast-1.79797/) – Дата доступа: 15.10.2023

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Листинг программного кода

# ПРИЛОЖЕНИЕ Б

# (обязательное)

# Конечная схема базы данных

# ПРИЛОЖЕНИЕ В

# (обязательное)

# Ведомость курсового проекта