ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc193489224)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ГЛАВА 6](#_Toc193489225)

[1.1. Общие сведения 6](#_Toc193489226)

[1.2. Обоснование выбора среды и языка программирования разработки. 11](#_Toc193489227)

[1.3. Обзор и выбор СУБД 14](#_Toc193489228)

[1.4 Описание этапа разработки 18](#_Toc193489229)

[ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 21](#_Toc193489230)

[2.1 Проектирование и нормализация базы данных 21](#_Toc193489231)

[2.2. Разработка информационной подсистемы 23](#_Toc193489232)

[ЗАКЛЮЧЕНИЕ 30](#_Toc193489233)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc193489234)

# ВВЕДЕНИЕ

В рамках данного курсового проекта рассматривается задача разработки специализированной системы для учёта заявок на техническое обслуживание оборудования. Основная цель проекта — создание программного обеспечения, способного эффективно управлять информацией о заявках, оборудовании, исполнителях и статусах выполнения работ. Это позволит компании, занимающейся техническим обслуживанием, более точно и оперативно управлять своими ресурсами, улучшая качество обслуживания клиентов и повышая уровень удовлетворённости заказчиков.

В процессе реализации проекта был осуществлён детальный выбор подходящего языка программирования и сопутствующих технологий. Для хранения и обработки данных о заявках на техническое обслуживание также был проведён анализ различных систем управления базами данных (СУБД). Это позволило определить наиболее эффективные инструменты для разработки системы, обеспечивающей удобное управление информацией и быстрый доступ к данным.

Актуальность данной темы обусловлена тем, что для успешной работы компании, занимающейся техническим обслуживанием оборудования, необходимо удобное и функциональное программное обеспечение, которое позволит автоматизировать процессы учёта заявок, управления задачами и контроля выполнения работ.

Практическая значимость работы заключается в том, что компания нуждается в удобном и функциональном программном решении, которое позволит повысить скорость и эффективность обработки заявок, минимизировать ошибки, связанные с ручным управлением данными, и улучшить взаимодействие с клиентами.

Программное обеспечение для учёта заявок на техническое обслуживание оборудования предназначено для хранения информации о заявках, оборудовании, исполнителях и статусах выполнения работ, а также для автоматизации процессов управления техническим обслуживанием.

Цель работы — разработка программного обеспечения для учёта и управления заявками на техническое обслуживание оборудования.

Задачи работы:

* произвести обзор среды разработки;
* рассмотреть и выбрать СУБД;
* описать область применения подсистемы;
* спроектировать UML диаграмму для информационной подсистемы;
* разработать программу подсистемы на языке программирования Python;
* подключить программу к СУБД SQL;
* протестировать информационную подсистему;
* сделать выводы на основе проведенного исследования.

Объект исследования — процесс управления арендой офисных помещений.

Предмет исследования — проектирование и разработка подсистемы управления заявками на техническое обслуживание оборудования.

Методы исследования, применяемые в данном курсовом проекте, включают теоретические, эмпирические и статистические подходы. Это позволяет всесторонне рассмотреть задачу разработки системы учёта заявок на техническое обслуживание оборудования.

Структура курсового проекта включает следующие основные разделы:

* **Введение**: В данном разделе обозначена актуальность темы, сформулированы цель и задачи исследования. Описаны причины, по которым разработка данной системы является важной и своевременной для компании, занимающейся техническим обслуживанием оборудования.
* **Теоретическая часть**: включает обзор различных языков программирования, среди которых был выбран один язык как наиболее подходящий для реализации проекта. Также рассматриваются различные системы управления базами данных (СУБД), и объясняется, почему для данного проекта была выбрана именно эта база данных. Описывается область применения разрабатываемой подсистемы, её функциональные возможности и преимущества.
* **Практическая часть**: включает создание UML-диаграммы, которая иллюстрирует структуру и взаимодействие компонентов разрабатываемой подсистемы. Описывается процесс проектирования и реализации подсистемы с использованием выбранного языка программирования, а также её интеграция с базой данных.

**Заключение**:

В данном разделе подводятся итоги выполненной работы, оценивается эффективность разработанной системы для учёта заявок на техническое обслуживание оборудования.

Разработанная система доказала свою эффективность в ходе тестирования, продемонстрировав стабильную работу, удобство использования и высокую производительность. Внедрение данной системы позволит улучшить качество предоставляемых услуг и повысить уровень удовлетворённости клиентов.

В процессе работы над курсовым проектом были изучены труды современных специалистов в области программирования и проектирования программных систем, а также авторитетные источники, что позволило обеспечить научную обоснованность исследования и высокое качество реализации проекта.

**Основные параметры работы:**

* Количество страниц: 34.
* Количество литературных источников: 18.
* Количество рисунков: 8.

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ГЛАВА

* 1. **Общие сведения**

В компании, которая занимается техническим обслуживанием оборудования, существует необходимость в усовершенствовании системы учёта заявок и управления процессами обслуживания. Основная цель проекта заключается в разработке программного обеспечения, которое позволит эффективно управлять информацией о заявках, оборудовании, исполнителях и статусах выполнения работ. Это поможет компании более оперативно и точно реагировать на запросы клиентов, что, в свою очередь, повысит качество обслуживания и уровень удовлетворённости заказчиков.

Сотрудники компании сталкиваются с трудностями при управлении данными о заявках и техническом обслуживании оборудования. Текущая система учёта неэффективна, что приводит к ошибкам, дублированию данных и замедлению процесса обработки заявок. Это негативно сказывается на качестве обслуживания и удовлетворённости клиентов. Новое программное обеспечение призвано устранить эти проблемы, обеспечив более удобный и надёжный инструмент для работы с базой данных о заявках, оборудовании и исполнителях.

**Требования к ПО:**

**Функциональные требования:**

* Учёт и хранение информации о заявках на техническое обслуживание (номер заявки, описание проблемы, дата создания, статус выполнения).
* Управление данными об оборудовании (тип, модель, серийный номер, местоположение).
* Контроль выполнения работ по заявкам (сроки, этапы выполнения, отчёты).
* Анализ статистики по заявкам и эффективности работы исполнителей.
* Интеграция с другими системами компании (например, CRM или системами управления ресурсами).
* Уведомления о важных событиях (например, изменение статуса заявки или просроченные задачи).

**Нефункциональные требования:**

* Высокая производительность и скорость обработки данных.
* Удобный и интуитивно понятный интерфейс для пользователей.
* Надёжность и отказоустойчивость системы.
* Защита данных (соблюдение требований GDPR или других нормативов).
* Масштабируемость системы для будущего расширения функционала.
* Кроссплатформенность (возможность работы на разных операционных системах).

**Основание разработки:**

Основанием для разработки программного обеспечения является индивидуальное задание к курсовому проекту, которое предусматривает создание системы для автоматизации учёта заявок на техническое обслуживание оборудования и управления процессами их выполнения.

**Что автоматизируется с помощью ПО:**

Учёт заявок: автоматизация сбора, хранения и обработки данных о заявках на техническое обслуживание.

Управление оборудованием: ведение базы данных по оборудованию, включая его характеристики и местоположение.

Анализ выполнения заявок: автоматический анализ данных для выявления задержек и формирования отчётов о выполнении работ.

Уведомления: автоматические напоминания о важных событиях (например, изменение статуса заявки или просроченные задачи).

Поиск и фильтрация данных: быстрый доступ к необходимой информации о заявках, оборудовании и исполнителях.

Разбор предметной области

Информационная подсистема представляет собой совокупность программных средств и данных, обеспечивающих на предприятии информационное сопровождение системы учёта и контроля заявок на техническое обслуживание оборудования.

**Виды информационных подсистем:**

* **Структурированная подсистема:** Это задача, в которой известны все элементы и их взаимосвязи. В структурированной задаче содержание можно выразить в форме математической модели с чётким алгоритмом решения. Такие задачи носят рутинный характер и решаются многократно. Цель использования информационной подсистемы для решения структурированных задач — полная автоматизация процесса, минимизирующая участие человека.
* **Неструктурированная подсистема:** Это задача, в которой невозможно выделить элементы и установить между ними связи. Решение неструктурированных задач связано с трудностями из-за отсутствия математического описания и чёткого алгоритма. Возможности использования информационной подсистемы в таких случаях ограничены и основываются на косвенной информации из различных источников.
* **Экспертная подсистема:** Это программа, которая ведёт себя подобно эксперту в узкой прикладной области. Типичные применения включают медицинскую диагностику и локализацию неисправностей в оборудовании. Такие системы используют базу знаний и логические правила для анализа данных и принятия решений, что позволяет им имитировать процесс мышления специалиста.

**Классификация подсистем по степени автоматизации:**

* **Ручные информационные подсистемы:** Характеризуются отсутствием современных технических средств обработки информации, все операции выполняются вручную человеком. Это приводит к низкой скорости выполнения задач, высокой вероятности ошибок и ограниченным возможностям для анализа данных.
* **Автоматические информационные подсистемы:** выполняют все операции по переработке информации без участия человека.
* **Автоматизированные информационные подсистемы:** предполагают участие как человека, так и технических средств, при этом главная роль отводится компьютеру.

**Классификация подсистем по характеру использования информации:**

* **Информационно-поисковые системы:** Осуществляют ввод, систематизацию, хранение и выдачу информации по запросу пользователя без сложных преобразований данных. Примеры включают системы в библиотеках, кассах продажи билетов или базах данных для учёта клиентов. Такие системы предназначены для работы с большими объёмами информации, обеспечивая быстрый доступ к ней и удобное управление.
* **Информационно-решающие системы:** выполняют все операции переработки информации по определенному алгоритму. Среди них выделяют управляющие и советующие системы в зависимости от степени воздействия выработанной информации на процесс принятия решений.

**Разработка программы**

После анализа различных языков программирования будет выбран подходящий язык программирования. На этом языке будет разработана программа база данных «Учет клиентов и их потребностей» с использованием СУБД для взаимодействия SQL.

**Принцип действия информационной подсистемы:**

Программа позволяет по имени клиента выводить всю информацию о нем, включая:

* Подразделение, в котором он обслуживается.
* Количество и тип страховых полисов.
* Контактную информацию.
* Общую стоимость страховых услуг.

Эта система обеспечивает эффективное управление клиентскими данными, улучшая качество обслуживания и повышая оперативность работы сотрудников страхового агентства.

**Аналоги систем учёта заявок на техническое обслуживание оборудования включают:**

* Системы управления сервисными заявками (Service Request Management) Примеры: Zendesk, Freshdesk, ServiceNow. Эти системы позволяют регистрировать, отслеживать и управлять заявками на обслуживание, а также автоматизируют процессы взаимодействия с клиентами.
* CMMS (Computerized Maintenance Management Systems) Примеры: Maintenance Connection, UpKeep, Fiix. CMMS системы предназначены для управления техническим обслуживанием оборудования, включая учёт заявок, планирование работ и контроль выполнения.
* ERP системы с модулями для управления обслуживанием Примеры: SAP ERP, Oracle ERP, 1C:ERP. Эти системы включают функционал для учёта заявок, управления ресурсами и анализа данных по техническому обслуживанию.
* Специализированные системы для учёта заявок Примеры: OTRS, Jira Service Management. Такие системы предоставляют инструменты для создания, обработки и мониторинга заявок, а также интеграции с другими бизнес процессами.
* CRM системы с функциями управления обслуживанием Примеры: Salesforce, HubSpot, Bitrix24. CRM системы могут быть адаптированы для учёта заявок на техническое обслуживание, обеспечивая удобное взаимодействие с клиентами и контроль выполнения задач.

**Преимущества разработанной системы**

1. **Индивидуальная настройка** – Возможность адаптации под специфические нужды конкретной компании.
2. **Локальное размещение** – Хранение данных локально обеспечивает дополнительный уровень контроля и безопасности.
3. **Простота использования** – Интуитивно понятный интерфейс снижает время на обучение и повышает эффективность работы.
4. **Гибкость и масштабируемость** – Легкость добавления новых функций и масштабирования системы по мере роста компании.
5. **Экономическая эффективность** – отсутствие необходимости в дорогостоящих лицензиях и подписках.
6. **Высокий уровень автоматизации** – Возможности для автоматизации рутинных задач, позволяющие сотрудникам сосредоточиться на более важных аспектах работы.
7. **Безопасность данных –** Встроенные меры безопасности для защиты конфиденциальной информации клиентов.

## **Обоснование выбора среды и языка программирования разработки.**

Для работы с базами данных применяются специализированные языки. Среди них:

SQL («язык структурированных запросов») — предназначен для работы с реляционными базами данных. Появился в 1974 году и с тех пор претерпел множество изменений, но основные принципы остались неизменными.

Преимущества:

* эффективен при работе с запросами, обновлениями и обработкой реляционных баз данных;
* декларативный синтаксис делает SQL читаемым языком.

Недостатки:

* синтаксис SQL может показаться сложным для тех, кто привык к императивному программированию;
* существует множество различных версий SQL, таких как PostgreSQL, SQL, MariaDB. Они несовместимы между собой.

R язык программирования с открытым исходным кодом, который используется для анализа данных и работы с графикой. Изначально R был разработан сотрудниками Оклендского университета Россом Айхэкой и Робертом Джентлменом. Они хотели создать язык, который помог бы им в процессе обучения студентов основам статистики. R был создан под влиянием языка S, который был разработан компанией Bell Labs в 1970е годы.

R считается предметно ориентированным языком, поскольку он был создан преимущественно для анализа данных. R можно использовать как в виде интерактивной среды, так и в виде скриптов и моделей, которые могут быть встроены в пакеты и интегрированы с другими программными модулями.

Язык R можно использовать для анализа данных из различных источников, включая внешние файлы или базы данных.

Преимущества:

* R имеет широкий набор высококачественных предметно ориентированных пакетов с открытым исходным кодом. R предоставляет пакеты для любого количественного и статистического применения, включая нейронные сети, нелинейную регрессию, филогенетику, построение сложных диаграмм и графиков.
* вместе с базовой установкой предоставляется возможность установки обширных встроенных функций и методов. Кроме того, R обрабатывает данные матричной алгебры.
* возможность визуализации данных и различных библиотек, например ggplot2.

Недостатки:

* низкая производительность;
* специфичность. R подходит для статистических исследований и науки о данных, но не подходит для общего программирования.

Julia — высокоуровневый, высокопроизводительный язык программирования с динамической типизацией, предназначенный для математических вычислений. Его синтаксис похож на синтаксис других сред технических вычислений, таких как Matlab и Octave. Язык написан на Си, C++ и Scheme.

Преимущества:

* Julia — это скомпилированный язык JIT, который может обеспечить высокую производительность. Язык поддерживает возможности динамической типизации и сценариев интерпретируемого языка, такого как Python.
* Хорошо подходит для численного анализа и может рассматриваться как язык программирования общего назначения.

Недостатки:

* Julia — новый язык, поэтому разработчики могут столкнуться с проблемами при работе с его пакетами.
* Ограниченное количество пакетов программ.

Java — популярный язык общего назначения, который работает на виртуальной машине Java Virtual Machine (JVM). Это абстрактная вычислительная система, которая обеспечивает переносимость между платформами. В настоящее время поддерживается корпорацией Oracle.

Преимущества:

* Универсальность. Многие современные системы и приложения разработаны с использованием Java. Преимущество такого языка заключается в том, что он позволяет интегрировать методы науки о данных непосредственно в существующую кодовую базу.
* Строгая типизация. Обеспечение типобезопасности для Java особенно важно при разработке важных приложений для работы с большими данными.

Недостатки:

* Java не подходит для специальных анализов и разработки специализированных статистических приложений.
* Java не имеет большого количества библиотек для передовых статистических методов по сравнению с некоторыми предметно ориентированными языками, такими как R.

Python — это универсальный язык программирования высокого уровня, который широко используется для различных проектов разработки. Он был создан и выпущен в 1991 году после разработки в конце 80х годов. Python используется в различных приложениях и программах, включая веб и мобильную разработку, Анализ данных и машинное обучение**,** Системное администрирование. Это объектно-ориентированный язык с открытым исходным кодом, который был разработан с помощью языка C.

Scala — это язык программирования, который был разработан для создания компонентного программного обеспечения. Он был создан для того, чтобы обеспечить лёгкость и быстроту разработки, а также для того, чтобы сделать код более читаемым и понятным. В Scala используются как функциональные, так и объектно-ориентированные парадигмы программирования.

Преимущества:

* Использование Scala и Spark позволяет работать с высокопроизводительными кластерными вычислениями.
* Мультипарадигматический. В Scala доступны как объектноориентированные, так и функциональные парадигмы программирования
  1. **Обзор и выбор СУБД**

Рассмотрим программное обеспечение – базу данных для хранения информации о заявках на техническое обслуживание, а также для создания базы данных об оборудовании, исполнителях и истории выполнения работ на протяжении всего периода взаимодействия с клиентами. Для проектирования этой системы проведём анализ различных программных решений и выберем наиболее подходящее.

Среди наиболее популярных систем управления базами данных (СУБД) для решения подобных задач выделяются следующие:

1. SQL – это встраиваемая реляционная база данных, разработанная на языке программирования C. Она предоставляет широкий набор команд SQL. В отличие от большинства других баз данных, работающих по схеме клиент/сервер, SQL использует динамическую типизацию данных. Это означает, что тип данных в столбце не ограничивает тип значения, которое может быть в него записано.

Преимущества:

* Файловая структура: Одним из ключевых преимуществ SQL является то, что вся база данных хранится в одном файле. Это упрощает переносимость, так как файл можно легко копировать между устройствами. Резервное копирование также становится простым — достаточно сделать копию файла для сохранения всех данных.
* Кроме того, отсутствие необходимости в отдельном сервере делает SQL удобным для разработки и использования в мобильных приложениях и встроенных системах.
* Стандарты: SQL поддерживает большинство стандартных SQLоператоров и конструкций, что позволяет разработчикам использовать знакомый синтаксис для работы с базами данных.
* Удобство для разработки и тестирования: Благодаря своей простоте (один файл и библиотека на C), SQL идеально подходит для разработки и тестирования приложений, особенно когда требуется масштабирование и быстрая настройка среды. Эта легковесная база данных позволяет разработчикам сосредоточиться на создании функциональности без необходимости в сложной конфигурации. Кроме того, она обеспечивает высокую производительность и простоту в использовании, что делает ее отличным выбором для проектов любого размера.

Недостатки:

* Отсутствие системы пользователей: Нет возможности настройки прав доступа для разных пользователей.
* Ограниченная производительность: Нет механизмов для увеличения производительности при работе с большими объемами данных.

1. MySQL – это реляционная СУБД с открытым исходным кодом, которая поддерживает табличные базы данных как с простой, так и со сложной структурой. Она отличается высокой скоростью обработки данных, гибкостью и простотой интерфейса. MySQL широко используется в ecommerce, IT и финтехе для построения прогностических моделей.

Преимущества:

* Простота: Легко устанавливается и настраивается. Существует множество сторонних инструментов, упрощающих работу с базой данных.
* Функциональность: Поддерживает большую часть стандартных функций SQL.
* Безопасность: Встроенные механизмы безопасности обеспечивают защиту данных.
* Масштабируемость: MySQL обладает высокой производительностью и способна эффективно обрабатывать большие объемы данных. Эта система управления базами данных (СУБД) идеально подходит для разработки масштабируемых приложений, которые могут расти и развиваться с течением времени. Благодаря своей надежности и поддержке множества пользователей одновременно, MySQL становится отличным выбором для проектов, требующих стабильности и высокой доступности.

Недостатки:

* Ненадежность: Некоторые операции менее надежны по сравнению с другими СУБД.

1. Microsoft SQL Server – это СУБД, разработанная компанией Microsoft, которая поддерживает операционные системы Windows и Linux. Она характеризуется простым интерфейсом, высокой надежностью и совместимостью с другими продуктами Microsoft, такими как Excel и Access.

Преимущества:

* Масштабируемость: Подходит как для портативных устройств, так и для мощных серверов.
* Высокая производительность: Быстрое извлечение данных благодаря размеру страниц до 8 КБ.
* Автоматизация: Рутинные задачи, такие как управление памятью и блокировками, автоматизированы.
* Интеграция: Поддерживает работу с другими продуктами Microsoft.

Недостатки:

* Высокая стоимость: Использование в коммерческих целях требует оплаты лицензии.

1. Oracle – это объектнореляционная СУБД, известная своей надежностью, практичностью и высокой производительностью. Она широко используется в Data Mining для анализа эффективности продаж и построения моделей потребностей клиентов.

Недостатки:

* Сложность: Требует глубоких знаний для настройки и управления.
* Высокая стоимость: Приобретение и поддержка Oracle обходятся дорого.

1. Redis (Remote Dictionary Server) – это СУБД с открытым исходным кодом, которая поддерживает хранение данных в памяти. Она используется для кэширования, хранения сессий пользователей, очередей и промежуточных результатов вычислений.

Преимущества:

* Высокая скорость: Все операции выполняются в памяти, что обеспечивает минимальные задержки.
* Гибкость: Поддерживает различные структуры данных, такие как строки, списки, хэши и наборы.
* Простота использования: Удобные форматы запросов и поддержка скриптов на LUA.

Недостатки:

* Ограниченный объем данных: Размер базы данных ограничен объемом доступной памяти.
* Отсутствие контроля доступа: Нет разделения прав доступа для пользователей.

В результате анализа была выбрана СУБД SQL по следующим причинам:

1. Отсутствие необходимости в администрировании: SQL не требует сложной настройки и управления.
2. Простота подключения: Легко интегрируется в приложения.
3. Высокая производительность: При достаточном объеме оперативной памяти обеспечивает быструю обработку данных.
4. Бесплатность: SQL распространяется бесплатно, что делает её экономически выгодным решением.

**Использование библиотеки и технологии:**

Для разработки информационной подсистемы на языке C# были использованы следующие библиотеки и технологии:

* **SQL** – библиотека для работы с базой данных SQL, обеспечивающая хранение и управление данными о заявках, оборудовании и исполнителях. Для работы с SQL в C# используется библиотека System.Data.SQL.
* **Windows Forms (WinForms)** – технология для создания графического интерфейса пользователя (GUI), которая была использована для разработки удобного и интуитивно понятного интерфейса.
* **Entity Framework (EF)** – ORMбиблиотека для работы с базами данных, которая упрощает взаимодействие с данными и позволяет использовать LINQ для запросов.
* **Chart Control (System.Windows.Forms.DataVisualization)** – библиотека для визуализации данных, которая может быть применена для создания графиков и диаграмм в аналитических отчётах.
* **NUnit/xUnit** – библиотеки для модульного и интеграционного тестирования, обеспечивающие проверку корректности работы системы.
* **ClickOnce/InstallShield** – инструменты для упаковки программы в установочный пакет, что упрощает её распространение и использование.
* **System.DateTime** – встроенная библиотека для работы с датами и временем, которая может быть использована для уведомлений о событиях (например, изменение статуса заявки или просроченные задачи).
* **System.Net.Mail** – библиотека для реализации функционала отправки уведомлений клиентам по электронной почте.
* **System.Security.Cryptography** – библиотека для шифрования данных, которая может быть использована для защиты конфиденциальной информации.

**1.4 Описание этапа разработки**

На этапе анализа требований определяются ключевые задачи, которые должно решать программное обеспечение. Основные требования включают учёт и хранение данных о заявках, таких как номер заявки, описание проблемы, дата создания, статус выполнения и приоритет. Система должна обеспечивать управление информацией об оборудовании, включая тип, модель, серийный номер, местоположение и состояние. Также требуется управление данными об исполнителях, такими как ФИО, контактные данные, специализация и закреплённые задачи. Важным аспектом является контроль выполнения работ, включая сроки выполнения, этапы работ и отчёты о выполнении. Система должна предоставлять возможность анализа данных на основе истории заявок и выполненных работ, формировать отчёты по выполненным работам, загруженности исполнителей и состоянию оборудования. Для удобства сотрудников интерфейс должен быть интуитивно понятным, позволяющим быстро находить информацию, управлять заявками и отслеживать статусы. Надёжное хранение данных обеспечивается использованием SQL, а также высокой производительностью и минимальным количеством ошибок при обработке запросов.

Для реализации системы используются современные средства и технологии. В качестве языка программирования выбран C#, благодаря его широким возможностям и интеграции с платформой .NET. Для хранения и управления данными применяется база данных SQL. Графический интерфейс разрабатывается с использованием Windows Forms (WinForms) или WPF для создания десктопного приложения. В качестве среды разработки используется IDE Visual Studio, а для контроля версий — система Git (например, через GitHub или GitLab). Для работы с базами данных применяются библиотеки System.Data.SQL. Дополнительно используются инструменты для тестирования (например, NUnit или xUnit) и создания документации (например, Sandcastle или DocFX).

Программирование и реализация системы:

Проектирование архитектуры программы начинается с определения структуры, включающей классы, методы и модули. Разрабатывается схема базы данных, включая таблицы и связи между ними. Планируется интерфейс пользователя, определяются основные окна, кнопки и формы для удобного взаимодействия с системой.

Реализация функционала включает создание базы данных с использованием SQL-запросов для создания таблиц и управления данными. Разрабатываются функции для работы с базой данных, такие как добавление, удаление и редактирование данных о заявках, оборудовании и исполнителях. Реализуется логика анализа данных, например, на основе истории выполнения заявок. Создаётся графический интерфейс с использованием Windows Forms или WPF, включая главное окно с меню и кнопками для навигации, формы для добавления и редактирования данных, а также окно для просмотра отчётов и аналитики. Интерфейс связывается с базой данных для отображения данных в таблицах.

Тестирование и критерии готовности:

Тестирование системы включает несколько этапов. Модульное тестирование проверяет отдельные компоненты программы, такие как методы для работы с базой данных. Интеграционное тестирование оценивает взаимодействие между различными модулями программы, например, графическим интерфейсом и базой данных. Системное тестирование проверяет работу программы в целом, включая все её функции. Ручное тестирование проводится для выявления ошибок в пользовательском опыте и удобстве интерфейса.

Критерии готовности системы включают реализацию всех функциональных требований, стабильную работу без критических ошибок, удобный интерфейс для пользователей, корректное сохранение и извлечение данных из базы данных, успешное прохождение всех этапов тестирования и наличие документации, соответствующей реальному функционалу.

Таким образом, разрабатываемая система учёта заявок на техническое обслуживание оборудования представляет собой комплексное решение, которое автоматизирует ключевые процессы, повышает эффективность работы и обеспечивает надёжное хранение данных. Использование современных технологий и инструментов делает систему гибкой, масштабируемой и удобной для пользователей..

**ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА**

## **Проектирование и нормализация базы данных**

База данных (БД) – это инструмент для накопления и организации больших объёмов информации о заявках, оборудовании и исполнителях. Она должна отражать актуальные данные, накапливать и хранить информацию, а также предоставлять различным категориям пользователей быстрый и удобный доступ к данным. Это позволяет эффективно управлять процессами технического обслуживания, оперативно обрабатывать заявки и анализировать данные для улучшения качества работы.

Разработка базы данных была выполнена в системе управления реляционными базами данных SQL. База данных хранит всю необходимую информацию для работы системы, включая данные о пользователях, заявках и материалах. Ниже приведено описание таблиц, их назначение и связь между ними.

**Таблицы базы данных:**

Таблица Clients создана для работы с данными клиентов

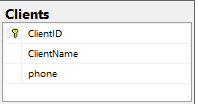
****

Рисунок 1 – Таблица client.

Таблица Requests создана для работы с заявками

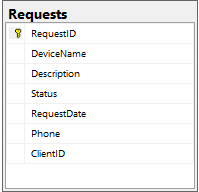


Рисунок 2 – Таблица Requests.

Таблица Users работает с данными авторизации.

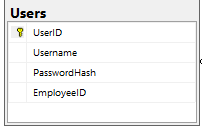
****

Рисунок 3 – Таблица Users.

Таблица employee работает с данными сотрудников.

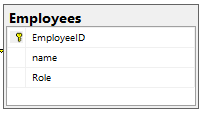


Рисунок 4 – Таблица Employees.

Таблица employee работает с данными сотрудников.

**Корректность данных:**

Схема данных обеспечивает целостность информации: например, нельзя удалить пользователя, если у него есть активные заявки.

Связи между таблицами позволяют быстро находить нужные данные, например, все заявки конкретного пользователя или помещения для определённой заявки.

Программа упрощает процесс работы с данными, минимизируя вероятность ошибок.

Проектирование информационной подсистемы будет происходить на языке программирования C#. Для начала нужен редактор кода Visual Studio, в котором и будет написан код.

В редакторе Visual Studio добавляется модуль для работы с SQL, а затем создается объект подключения к БД. Объект подключения создается с помощью метода connect():

Для выполнения операторов SQL, нужен объект курсора, создаваемый методом cursor(). Курсор – это метод объекта соединения. Для выполнения операторов сначала устанавливается соединение, а затем создается объект курсора с использованием объекта соединения следующим образом:

После создания соединения с SQL, файл БД создается автоматически, при условии его отсутствия. Этот файл создается на диске.

Далее с помощью функции CREATE TABLE IF NOT EXISTS автоматически создается таблица. Затем прописываются названия столбцов таблицы и их типы данных.

Для выборки данных используется оператор SELECT. Метод fetchall() объекта курсора подходит для сохранения значений в переменной. При этом переменная будет являться списком, где каждая строка из БД будет отдельным элементом списка.

Далее будет выполняться перебор значений переменной и их вывод. SQL синтаксис для этого будет следующим:

Метод commit() сохраняет все сделанные изменения. Когда работа с БД будет завершена, рекомендуется закрыть соединение. Соединение может быть закрыто с помощью метода close(). Окончательный вариант программы будет иметь следующий вид:

Этот код обеспечивает создание, управление и выборку данных клиентов в страховом агентстве.

**2.2. Разработка информационной подсистемы**

**Описание пользовательского интерфейса**

Информационная подсистема для учёта заявок на техническое обслуживание оборудование

Информационная подсистема является ключевым компонентом программного обеспечения, предназначенного для учёта заявок на техническое обслуживание оборудование. Её основная задача — обеспечение эффективного управления данными, их хранения, обработки и анализа. Разработка подсистемы включает несколько этапов:

1.Определение структуры данных:

* Анализ требований к данным: Определение, какие данные необходимо хранить и обрабатывать (например, информация об арендаторах, офисных помещениях, договорах аренды и платежах).
* Определение сущностей и их атрибутов:
* Арендаторы (ФИО, контактные данные, история взаимодействий).
* Офисные помещения (адрес, площадь, стоимость аренды, доступность).
* Договоры аренды (сроки, условия, статус, платежи).
* Связи между арендаторами и помещениями (например, какие помещения арендованы конкретным арендатором).
* Разработка схемы базы данных: Создание структуры базы данных с учётом нормализации для минимизации дублирования данных и обеспечения целостности.

Проектирование базы данных:

* Таблица tenants для хранения информации об арендаторах.
* Таблица office\_spaces для хранения данных об офисных помещениях.
* Таблица rental\_agreements для связи арендаторов с помещениями и хранения данных о договорах аренды.
* Определение первичных и внешних ключей: Обеспечение целостности данных через связи между таблицами (например, арендатор связан с договором, а договор — с помещением).
* Разработка индексов: Ускорение поиска данных, например, по имени арендатора или адресу помещения.
* Реализация функций работы с данными:
* Разработка модуля для взаимодействия с базой данных:
* Функции для добавления, удаления и редактирования данных: Например, добавление нового арендатора или обновление информации о помещении.
* Функции для поиска и фильтрации данных: Например, поиск арендатора по имени или фильтрация помещений по доступности.
* Функции для формирования отчётов: Например, список арендаторов с просроченными платежами или отчёт по занятости помещений.
* Реализация механизма анализа данных: Например, выявление наиболее востребованных помещений или анализ финансовых поступлений.

Интеграция подсистемы с графическим интерфейсом:

* Отображение данных: Например, таблицы с информацией об арендаторах, помещениях и договорах.
* Формы для ввода и редактирования данных: Удобные интерфейсы для добавления новых арендаторов или изменения условий договоров.
* Отображение результатов анализа данных: Например, графики занятости помещений или диаграммы финансовых поступлений.
* Обработка пользовательских запросов: Например, поиск арендатора по номеру телефона или фильтрация помещений по площади.

Обеспечение безопасности данных:

* Шифрование конфиденциальной информации: Например, паспортных данных арендаторов или финансовой информации.
* Ограничение доступа к данным: Разделение прав доступа для разных ролей пользователей (например, администратор и арендатор).
* Соблюдение требований законодательства: Например, GDPR для защиты персональных данных.

Тестирование подсистемы:

* Модульное тестирование: Проверка отдельных функций, таких как добавление арендатора или поиск помещения.
* Проверка корректности SQLзапросов: Убедиться, что данные правильно сохраняются и извлекаются из базы.
* Тестирование производительности: Проверка скорости обработки запросов при большом объёме данных.
* Устранение ошибок и оптимизация: Исправление выявленных проблем и улучшение работы подсистемы.

**Разработка приложения в рамках курсовой работы**

Для реализации приложения был выбран язык программирования C#, который является одним из самых популярных и современных языков программирования, обладающим высокой производительностью, гибкостью и широким набором библиотек и фреймворков. Разработка велась в интегрированной среде разработки Visual Studio Code, предоставляющей удобные инструменты для создания и отладки программ. Для хранения данных использовалась система управления базами данных SQL, отличающаяся простотой использования и возможностью интеграции с другими продуктами.

**Функционал приложения**

Чтобы начать работу с приложением, нужно ввести логин и пароль пользователя. При открытии программы появляется страница входа в систему. В поле «Имя пользователя» клиент или администратор вводит свой логин, а в поле «Пароль» свой пароль.

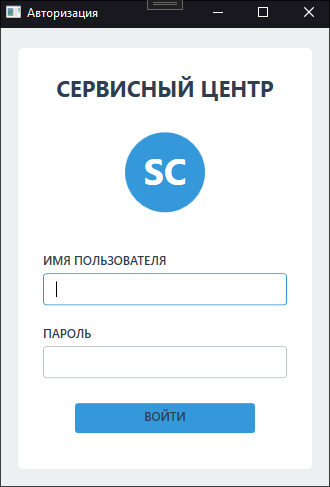
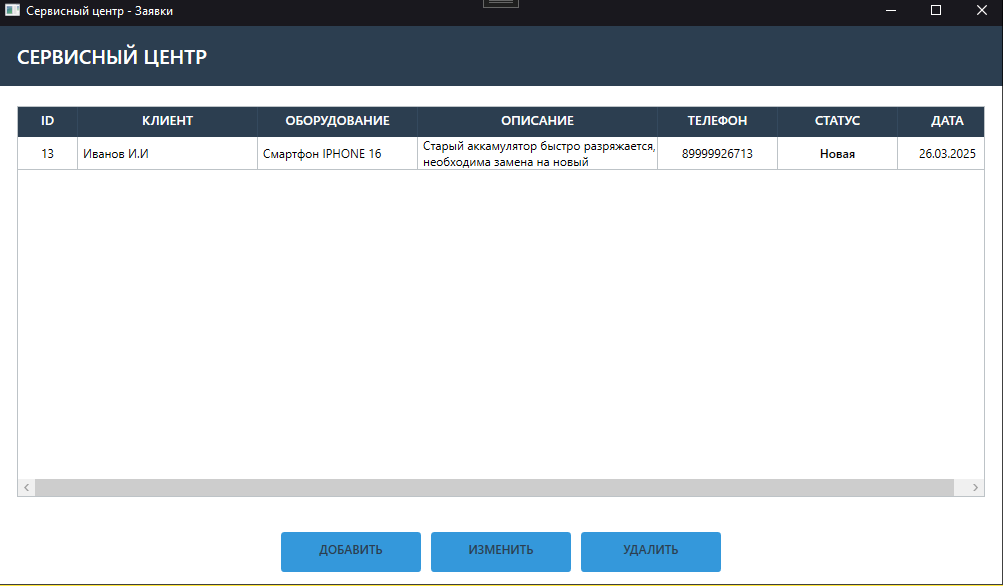


Рисунок 5 – Окно авторизации пользователя системы.

После успешного входа в учётную запись, пользователь видит стартовую страницу приложения. Главный экран позволяет добавить объявление и дополнить уже существующую, указав название, описание, телефон, Статус и дату.

Рисунок 6 – Главное окно.

Дальше продемонстрировано добавление заявки.

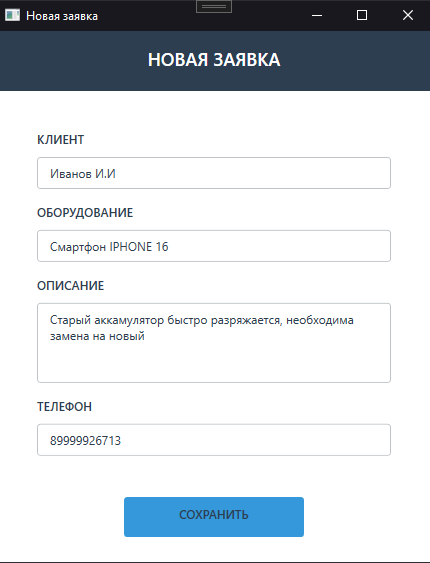


Рисунок 7 – Окно добавление заявки.

Таже есть возможность редактирования заявки

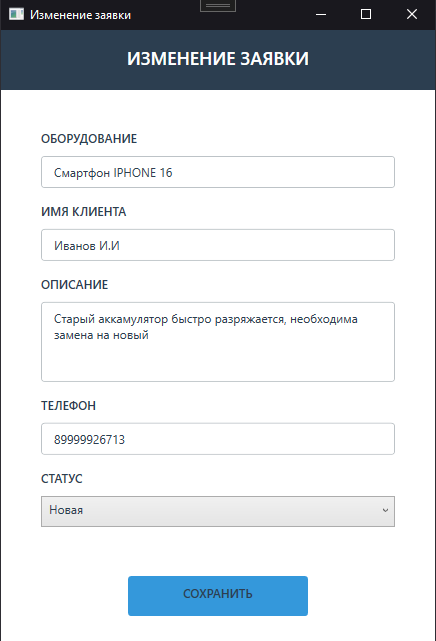


Рисунок 8 – Окно редактирования заявки.

**Тестирование созданной информационной подсистемы**

Тестирование информационной подсистемы является важным этапом разработки, направленным на обеспечение корректности работы всех её компонентов. Процесс тестирования включает следующие этапы:

Модульное тестирование направлено на проверку отдельных функций и модулей подсистемы. Оно включает тестирование функций для добавления, удаления и редактирования данных в базе данных, а также проверку корректности выполнения SQL-запросов, таких как выборка данных по определённым критериям. Кроме того, тестируются функции анализа данных, например, формирование отчётов, чтобы убедиться в их точности и корректности.

Интеграционное тестирование проверяет взаимодействие между различными компонентами подсистемы. Это включает тестирование связи между графическим интерфейсом и базой данных, а также проверку корректности передачи данных между модулями, например, от интерфейса к базе данных и обратно.

Системное тестирование оценивает работу подсистемы в целом. Оно охватывает тестирование всех сценариев использования, таких как добавление нового клиента, поиск страхового продукта или формирование отчёта. Также проверяется производительность подсистемы, включая скорость обработки запросов при работе с большими объёмами данных.

Тестирование безопасности направлено на проверку механизмов защиты данных. Это включает тестирование шифрования конфиденциальной информации и проверку ограничения доступа к данным для разных ролей пользователей, чтобы убедиться в надёжности системы.

Ручное тестирование фокусируется на проверке интерфейса подсистемы. Оно оценивает удобство использования форм для ввода и редактирования данных, корректность отображения данных в таблицах и списках, а также обработку ошибок, например, при вводе некорректных данных.

Подсистема считается готовой, если все её функции работают корректно, она стабильно функционирует без критических ошибок, данные корректно сохраняются и извлекаются из базы данных, а также если она успешно проходит все этапы тестирования без существенных замечаний.

**ЗАКЛЮЧЕНИЕ**

В рамках данного курсового проекта рассматривается задача создания специализированной программы для учёта заявок на техническое обслуживание оборудование. Система включает базу данных на основе SQL, функции для работы с данными и удобный графический интерфейс, созданный с использованием C#.

**Основные результаты проекта:**

* Создана эффективная система для управления заявок на техническое обслуживание оборудование.
* Разработан простой и удобный интерфейс, который облегчает работу сотрудников агентства.
* Проведено тестирование системы, подтвердившее её стабильность и готовность к использованию.

Благодаря этой системе можно:

* Повысить качество обслуживания клиентов.
* Сократить время обработки запросов.
* Уменьшить количество ошибок, связанных с ручным управлением данными.

**Планы по доработке системы:**

1. **Расширение функционала:**

* Добавить возможность интеграции с онлайн платформами, например, с вебсайтом агентства.
* Реализовать отправку уведомлений клиентам через SMS или электронную почту.
* Разработать мобильную версию приложения для удобства работы сотрудников вне офиса.

1. **Улучшение аналитических возможностей:**

* Внедрить более сложные алгоритмы для анализа данных, например, прогнозирование спроса на страховые продукты.
* Добавить визуализацию данных (графики, диаграммы) для более наглядного представления информации.

1. **Оптимизация производительности:**

* Улучшить скорость обработки запросов, особенно при работе с большими объёмами данных.
* Оптимизировать базу данных для более эффективного хранения и поиска информации.

1. **Усиление безопасности:**

* Внедрить двухфакторную аутентификацию для повышения безопасности доступа к системе.
* Реализовать шифрование данных при передаче между клиентом и сервером.

1. **Тестирование и доработка интерфейса:**

* Провести usability тестирование, чтобы выявить и устранить недостатки интерфейса.
* Адаптировать интерфейс под конкретные потребности сотрудников агентства.

Таким образом, цели и задачи курсового проекта были успешно достигнуты. Разработанная система готова к внедрению и может значительно улучшить работу страхового агентства. Дальнейшая доработка сделает её ещё более функциональной, удобной и безопасной, что повысит её ценность как для компании, так и для её клиентов.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Антонова, А. В. Компьютерное тестированиесовременная форма контроля учебных достижений учащихся в процессе обучения истории в школе / А. В. Антонова // В сборнике: Инновации и традиции педагогической науки2019. Сборник трудов ХIХ Международной научнопрактической конференции / под ред. Т. А. Макаренко, С. В. Паниной. — 2019. — С. 2327. — Текст : электронный // elibrary.ru [сайт]. — URL: https://www.elibrary.ru/download/elibrary\_37326820\_65482003.pdf (дата обращения: 04.02.2023).
2. Болье, А. Изучаем SQL. Генерация, выборка и обработка данных / А. Болье. — М.: Издательство ДиалектикаВильямс, 2020. — 423 с. — (Профессиональное образование). — Текст: электронный // ЭБС МГТУ им. Баумана [сайт]. — URL: https://elearning.bmstu.ru/iu6/pluginfile.php/18674/mod\_resource/content/1/Изучаем\_SQL\_Генерация%2C\_выборка\_и\_обработка\_данных%2C\_3\_изд.pdf (дата обращения: 01.02.2023).
3. Заельская, С. А. История отечества с древнейших времён и до наших дней в тестах и заданиях: электронное учебнометодическое пособие для бакалавров заочного отделения / С. А. Заельская. — Оренбург, 2019. — Текст : электронный // elibrary.ru [сайт]. — URL: https://lib.rucont.ru/efd/702462/info (дата обращения: 04.02.2023).
4. Зарапин, Р. В. Образовательный потенциал использования онлайнтестирования при изучении истории / Р. В. Зарапин // В сборнике: Актуальные вопросы гуманитарных наук: теория, методика, практика. Сборник научных статей IХ Всероссийской научнопрактической конференции с международным участием. — Москва, 2022. — С. 239244. — Текст : электронный // elibrary.ru [сайт]. — URL: https://www.elibrary.ru/download/elibrary\_50163440\_95910012.pdf (дата обращения: 04.02.2023).
5. Маркин, А. В. Программирование на SQL в 2 ч. Часть 1 : учебник и практикум для вузов / А. В. Маркин. — 3е изд., перераб. и доп. — М. : Издательство Юрайт, 2023. — 429 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/509818 (дата обращения: 04.02.2023).
6. Советов, Б. Я. Базы данных : учебник для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3е изд., перераб. и доп. — М. : Издательство Юрайт, 2023. — 420 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/510752 (дата обращения: 04.02.2023).
7. Стасышин, В. М. Базы данных: технологии доступа : учебное пособие для вузов / В. М. Стасышин, Т. Л. Стасышина. — 2е изд., испр. и доп. — М.: Издательство Юрайт, 2023. — 164 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/514252 (дата обращения: 04.02.2023).
8. MYSQL.COM — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.mysql.com/, свободный. (дата обращения: 01.02.2023).
9. Oracle Database Software — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.oracle.com/database/technologies/oracledatabasesoftwaredownloads.html, свободный. (дата обращения: 01.02.2023).
10. PostgreSQL: The World's Most Advanced Open Source Relational Database — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.postgresql.org/, свободный. (дата обращения: 01.02.2023).
11. SQL Server 2019. — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.microsoft.com/ruRU/download/details.aspx?id=101064, свободный. (дата обращения: 01.02.2023).
12. Васильев, А. Н. Программирование на Python в примерах и задачах / А. Н. Васильев. — 1е изд. — Москва: Эксмо, 2022. — 616 с.
13. Мэтиз, Э. Изучаем Python: программирование игр, визуализация данных, вебприложения / Э. Мэтиз. — 3е изд. — Питер: Издательский дом «Питер», 2022. — 560 с.
14. Любанович, Б. Простой Python. Современный стиль программирования / Б. Любанович. — 2е изд. — Питер: Издательский дом «Питер», 2025. — 592 с.
15. Иванов, И. И. Основы программирования на Java: учебное пособие для вузов / И. И. Иванов. — 2е изд., перераб. и доп. — М.: Издательство Юрайт, 2022. — 320 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/512345 (дата обращения: 05.02.2023).
16. Петрова, С. К. Вебразработка: современные технологии и инструменты / С. К. Петрова. — М.: Издательство ДиалектикаВильямс, 2021. — 480 с. — (Профессиональное образование). — Текст : электронный // ЭБС МГТУ им. Баумана [сайт]. — URL: https://elearning.bmstu.ru/iu6/pluginfile.php/18674/mod\_resource/content/1/Вебразработка\_Современные\_технологии\_и\_инструменты.pdf (дата обращения: 05.02.2023).
17. Сидоров, А. А. Машинное обучение: практическое руководство / А. А. Сидоров. — СПб.: Питер, 2023. — 400 с. — (Библиотека программиста). — Текст : электронный // ЭБС Лань [сайт]. — URL: https://e.lanbook.com/book/123456 (дата обращения: 05.02.2023).
18. Козлов, Д. В. Основы кибербезопасности: учебник для вузов / Д. В. Козлов. — М.: Издательство Юрайт, 2022. — 280 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/511234 (дата обращения: 05.02.2023).