

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«Название учебного заведения»**

Специальность **(специальность)**

Форма обучения – **(очная/заочная)**

**ДИПЛОМНЫЙ ПРОЕКТ**

на тему: **«Разработка программного модуля для учёта  
ремонта автомобилей»**

Выполнил:

студент группы **группа**

**ФИО**

---

(подпись)

РАБОТА ДОПУЩЕНА К ЗАЩИТЕ

Заместитель директора по УР

**ФИО**

---

(подпись)

Руководитель:

преподаватель колледжа,

**ФИО**

---

(подпись)

пос. Электроизолятор

2024

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1. АНАЛИТИЧЕСКАЯ ЧАСТЬ .....	5
1.1 Описание и анализ предметной области .....	5
1.2 Характеристика проблемы .....	7
1.3 Образ и сравнительный анализ .....	9
2. ПРОЕКТНАЯ ЧАСТЬ .....	10
2.1 Обзор и выбор средств проектирования решений .....	10
2.2 Проектирование архитектуры приложения .....	15
2.3 ER-диаграмма .....	18
2.4 Методы моделирования системы .....	24
2.5 Диаграмма переходов состояний .....	32
2.6 Макеты интерфейса приложения .....	36
3. СПЕЦИАЛЬНАЯ ЧАСТЬ .....	52
3.1 Формирование требований к ПО .....	52
3.2 Разработка интерфейса ПО .....	54
3.3 Описание разработки ПО .....	66
3.4 Разработка базы данных .....	68
3.5 Отладка программы .....	75
3.6 Руководство пользователя .....	78
ЗАКЛЮЧЕНИЕ .....	80
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	82
ПРИЛОЖЕНИЕ .....	83

## ВВЕДЕНИЕ

В современном мире автомобили играют значительную роль в повседневной жизни людей. В настоящее время он стал незаменимым средством передвижения для значительной части общества. Они не только предоставляют удобство и комфорт в передвижении, но и являются неотъемлемой частью экономики и инфраструктуры. С увеличением числа автопарков и ростом автомобильного рынка возрастают и потребности в обслуживании и ремонте транспортных средств для поддержания автомобилей в исправном состоянии и проведении регулярного технического обслуживания. Однако, эффективное управление процессом технического обслуживания и ремонта автомобилей представляет значительные вызовы для автосервисов и владельцев автотранспорта. Ведь одним из важных этапов обслуживания автомобиля является ремонт, который требует контроля и учета всех проведенных работ.

При этом, в сфере технического обслуживания автомобилей широко используются информационные технологии для улучшения эффективности и качества предоставляемых услуг. Разработка специализированных программных модулей для учета ремонта и обслуживания автомобилей становится все более востребованной задачей, поскольку они позволяют автосервисам и автовладельцам упростить процессы учета, планирования и контроля за ремонтом автомобилей.

Целью данной выпускной квалификационной работы является разработка программного модуля для учета ремонта автомобилей, который будет способствовать повышению эффективности работы автосервисов и облегчить процесс учета технического обслуживания и ремонта транспортных средств. Данный модуль будет представлять собой информационную систему, которая позволит автосервисам и владельцам автомобилей эффективно управлять

процессом ремонта, отслеживать выполненные работы, контролировать затраты на запчасти и услуги мастеров.

Для достижения данной цели необходимо решить следующие задачи:

1. Изучить существующие подходы к учету ремонта автомобилей и программные решения, используемые в автосервисах.
2. Определить требования к разрабатываемому программному модулю на основе анализа потребностей клиентов и особенностей процесса технического обслуживания автомобилей.
3. Спроектировать архитектуру программного модуля, включая функциональные возможности, интерфейсы пользователя, базу данных и алгоритмы обработки информации.
4. Разработать и реализовать программный модуль для учета ремонта автомобилей, учитывая специфику отрасли и требования клиентов.
5. Провести тестирование и оценку эффективности программного модуля на практике, сравнив результаты работы автосервиса до и после внедрения модуля.

Исходя из указанных целей и задач, разработка программного модуля для учета ремонта автомобилей представляет собой важную задачу, которая способствует оптимизации процесса обслуживания автомобилей и повышению качества предоставляемых услуг. Как результат работы ожидается получение программного модуля, который позволит оптимизировать процессы учета ремонта автомобилей, улучшить качество обслуживания и повысить удовлетворенность клиентов. Разработанный модуль будет иметь потенциал для расширения функционала и интеграции с другими информационными системами, что позволит эффективно управлять процессами ремонта автомобилей и повысить конкурентоспособность автосервисов в современном рыночном окружении.

# 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

## 1.1 Описание и анализ предметной области

Целью является рассмотреть информационную систему в сфере ремонта транспортных средств.

Задачи: проанализировать информационную систему, выяснить её актуальность и пригодность, создать диаграмму, помогающую в изучении системы, спроектировать базу данных и создать приложение на её основе.

Первый шаг в разработке программного модуля для учета ремонта автомобилей – это проведение анализа предметной области. Для достижения заданной цели необходимо на первом этапе подробно исследовать предметную область, используя для этого источники и средства получения информации. Далее из полученной информации выявить основные функции и задачи, организационную структуру и типичные сценарии работы, в удобной и понятной форме представить документооборот автосервиса. Для этого необходимо изучить все этапы и составляющие процесса ремонта автомобилей, включающие в себя следующие компоненты:

1. Прием автомобиля: Этот этап включает в себя заполнение заявки на ремонт, осмотр автомобиля, выявление неисправностей, установление контакта с клиентом. Программный модуль должен предоставлять возможность регистрации заявки, внесения информации о клиенте, описания проблемы с автомобилем.

2. Диагностика и оценка: на этом этапе производится тщательная диагностика неисправностей, определение списка работ и запчастей, а также оценка стоимости ремонта. Программный модуль должен позволять заносить данные о проведенной диагностике, определять необходимые запчасти и расчет стоимости ремонта.

3. Ремонтные работы: включает в себя выполнение всех необходимых работ по ремонту автомобиля, замену запчастей, регулировку, сборку и проверку исправности автомобиля. Программный модуль должен отражать процесс проведения работ, управлять складом запчастей, вести учет времени, затраченного на ремонт.

4. Финансовая отчетность: после завершения ремонта необходимо составить отчет о проделанных работах, запчастях, затратах и выставить счет клиенту. Программный модуль должен автоматизировать процесс формирования отчета и счета, а также вести учет финансовых операций.

5. Контроль качества: Важным элементом является контроль качества проведенных работ и обратная связь с клиентом. Программный модуль должен предоставлять возможность для оценки качества работы и получения обратной связи от клиента.

Разработка программного модуля для учета ремонта автомобилей позволит упростить и оптимизировать весь процесс работы автосервиса, уменьшить вероятность ошибок и повысить эффективность.

Программный модуль должен быть интуитивно понятным, легким в использовании, адаптированным для разных устройств и операционных систем. Важной частью модуля является база данных, в которой будут храниться все данные о клиентах, автомобилях, проделанных работах, использованных запчастях и т.д. Система должна поддерживать возможность генерации отчетов, статистики по выполненным работам, анализа прибыльности, управления складом запчастей и др.

Разработка программного модуля для учета ремонта автомобилей является сложным и многогранным процессом, требующим комплексного подхода к анализу, проектированию, разработке и внедрению. Однако правильно спроектированный и реализованный модуль поможет значительно улучшить управление автомобильным сервисом, повысить качество обслуживания клиентов, оптимизировать процессы и увеличить прибыльность бизнеса.

Одной из основных задач системы является возможность значительно сократить время, затрачиваемое на административные и технические задачи, на учет и обработку данных, а также на правильное формирование и постановку задач по ремонту автомобилей, что позволяет нацелить больше времени на улучшение качества выполняемых работ и совершенствование методов по оказанию ремонтных услуг, в связи повышения квалификации работников.

Однако при использовании любой информационной системы, возникают определенные риски, такие как возможность несанкционированного доступа к базе данных, её уничтожение или, при наихудшем сценарии, утечки личной информации и нарушения системы безопасности. Поэтому необходимо принимать меры по защите информации в системе от злоумышленников, обеспечивать регулярное обновление системы и её зависимостей, а также контролировать процесс администрирования и управления системой.

В целом, информационная система в сфере оказания услуг по ремонту транспортных средств является важным инструментом для современного предприятия, закладывающим принципы эффективного управления и качественно выполненных работ как на начальных этапах формирования предприятия по ремонту транспортных средств, так и для развитого и крупного предприятия, что делает ее очень востребованной и актуальной в сегодняшнее время.

## **1.2 Характеристика проблемы**

Характеристика проблемы, изучаемой в рамках текущего дипломного проекта, включает в себя ряд ключевых аспектов, которые требуют внимания и решения:

1. Отсутствие эффективной системы учёта ремонтных работ:

Владельцы автомастерских и автосервисов часто сталкиваются с проблемой необходимости ведения учёта всех ремонтных работ, затрат материалов и времени на ремонт. Отсутствие систематизированной программы для учёта данных приводит к потере информации, неоптимизированным процессам учёта и возможному увеличению издержек.

## 2. Низкая автоматизация и цифровизация процессов учёта:

Многие автосервисы до сих пор используют устаревшие методы учёта ремонта, такие как бумажные документы, электронные таблицы или устаревшие программы, что затрудняет взаимодействие и доступ к информации для сотрудников и клиентов.

## 3. Необходимость повышения прозрачности и надёжности информации:

Работа с данными о ремонте автомобилей требует высокой степени точности и надёжности. Ручной ввод информации может привести к ошибкам и потере ценных данных, что в свою очередь может отрицательно сказаться на доверии клиентов к сервисному центру.

## 4. Увеличение конкуренции на рынке автосервисов:

Сегодня автомобильный рынок насыщен множеством автосервисов, и владельцам необходимо по максимуму оптимизировать процессы, чтобы выделиться на фоне конкурентов и привлечь больше клиентов.

Разработка программного модуля для учёта ремонта автомобилей представляет собой ключевое решение этих проблем, позволяя автомастерским и автосервисам эффективно автоматизировать учёт производимых работ, улучшить качество обслуживания клиентов, повысить прозрачность и надёжность информации, а также повысить конкурентоспособность на рынке.



### 1.3 Образ и сравнительный анализ

Программный модуль для учета ремонта автомобилей – это инновационное информационное решение, разработанное с целью повышения эффективности управления процессом ремонта и технического обслуживания автотранспортных средств. В его основе лежит комплексный подход к автоматизации учета всех этапов технического обслуживания и ремонта автомобилей, начиная от регистрации заявки от клиента и заканчивая выдачей готового автомобиля.

Программный модуль представляет собой интуитивно понятный интерфейс, который позволяет пользователям удобно вводить информацию о каждом автомобиле, проводимых работах, расходных материалах, затратах времени и других аспектах технического обслуживания. Он обладает возможностью хранить и анализировать данные по каждому автомобилю, создавать отчеты о проделанных работах, запланированных и текущих ремонтах.

Этот модуль является надежным инструментом для автосервисов и автопредприятий, позволяя им проводить эффективный мониторинг состояния автомобилей, планировать профилактические работы и повышать уровень обслуживания для клиентов.

В рамках сравнительного анализа программный модуль для учета ремонта автомобилей будет сопоставлен с традиционными методами учета, такими как ручной ввод данных в таблицы Excel или использование бумажной документации.

1. Эффективность и точность данных: в отличие от ручного ввода, программный модуль обеспечивает автоматическое сохранение и структурирование данных, что исключает возможность ошибок при ручном ведении документации. Это повышает точность и актуальность информации.

2. Скорость обработки информации: Программный модуль значительно ускоряет процесс учета работ по ремонту и обслуживанию автомобилей за счет

автоматизации многих задач. Это позволяет сократить время на ведение учета и создание отчетов.

3. Доступ к данным: в отличие от бумажных документов, программный модуль обеспечивает быстрый доступ к необходимым данным, возможность быстрого поиска информации о предыдущих работах по автомобилю, а также создание аналитических отчетов.

4. Отчетность и аналитика: Программный модуль позволяет генерировать разнообразные отчеты о проделанных работах, затрате времени и материалов, анализировать данные по эффективности работы автосервиса.

Следовательно, разработка программного модуля для учета ремонта автомобилей предоставляет значительные преимущества по сравнению с традиционными методами учета, что делает его важным инструментом для современных автосервисов и предприятий автомобильной отрасли.

## **2. ПРОЕКТНАЯ ЧАСТЬ**

### **2.1 Обзор и выбор средств проектирования решений**

Прежде чем рассматривать инструментальные средства информационных систем, необходимо определить, что из себя представляет понятие инструментальные средства, для этого необходимо обратиться к понятию «информационная система». Информационная система (ИС) — это программно-аппаратный комплекс, который используется для сбора, хранения, обработки и выдачи информации. Традиционно информационные системы работают с огромными объемами сложно структурированных данных. Интеграция программного, аппаратного обеспечения и данных позволяет автоматизировать управление информацией. Иными словами, под определение информационной

системы представляют как программно-аппаратную систему, выполняющую в соответствии с запрограммированной в ней логикой (получение, обработку, хранение и вывод информации). При этом под инструментальными средствами информационных систем понимают совокупность аппаратных и программных средств, обеспечивающих функционирование.

В наиболее общем виде, информационную систему можно представить состоящей из следующих элементов:

- источника информации;
- аппаратной части ИС;
- программной части ИС;
- потребителя информации.

Источник информации для информационной системы может служить объект, идентифицирующий происхождение информации. Под эту категорию можно подставить человека, вносящего данные в информационную систему или же любой другой, доступный информационной системе, источник.

Аппаратные средства включают в себя средства вычислительной и коммуникационной техники. А программные средства разделяются на системное и прикладное ПО. Под программным обеспечением понимается совокупность программ и соответствующей документации, выполняемых вычислительной системой.

К программному обеспечению относится вся область деятельности по проектированию и разработке ПО:

- технология проектирования программ;
- методы тестирования программ;
- методы доказательства правильности программ;
- анализ качества работы программ;
- документирование программ;
- разработка и использование программных средств.

Для создания программного продукта необходимы средства создания приложений, которые включают в себя различные языки и системы программирования, а также инструментальную среду разработчика.

Язык программирования — это формализованный искусственный язык для записи программ — последовательностей действий на каком-либо устройстве-исполнителе (без вмешательства внешнего естественного или искусственного интеллекта в процесс исполнения). Первым языком программирования принято считать двоичные машинные коды, алфавит которых состоит из символов 0 и 1.

Языки программирования разделяют на следующие классы:

- машинные языки — языки программирования, воспринимаемые аппаратной частью компьютера (машинные коды);
- машинно-ориентированные языки — языки программирования, которые отражают структуру конкретного типа компьютера (ассемблеры);
- алгоритмические языки — не зависящие от архитектуры компьютера языки программирования для отражения структуры алгоритма (Паскаль, Фортран, Бейсик и др.);
- процедурно-ориентированные языки. Языки программирования, где имеется возможность описания программы как совокупности процедур, подпрограмм;
- проблемно-ориентированные языки — языки программирования, предназначенные для решения задач определенного класса.

Для разработки программного модуля, предназначенного для учета ремонта автомобилей, важно выбрать правильные средства проектирования, которые обеспечат эффективное функционирование и удовлетворение потребностей пользователей. В данном случае выделим основные этапы выбора средств проектирования и их применение.

#### 1. Выбор языка программирования:

Для разработки программного модуля учета ремонта автомобилей можно использовать различные языки программирования, такие как Python, Java, C#, PHP и другие. Однако, для более удобного и быстрого создания приложения, рекомендуется выбрать язык программирования C#, который обладает относительно простым синтаксисом и широкими возможностями для работы с базами данных, что является важным при проектировании приложения для учёта ремонта автомобилей.

## 2. Выбор инструментов разработки:

Для разработки программного модуля можно использовать интегрированные среды разработки (IDE), такие как Visual Studio. Visual Studio – отличный выбор для работы с C#, предоставляя широкий набор инструментов для отладки, автозаполнения кода и управления проектом. Также важно выбрать удобный инструмент для работы с базами данных, например, MySQL Workbench или SQLite Studio.

## 3. Выбор архитектуры приложения:

Для разработки программного модуля для учета ремонта автомобилей рекомендуется выбрать модульную архитектуру, такую как MVC (Model-View-Controller) или MVVM (Model-View-ViewModel). MVC обеспечит четкое разделение бизнес-логики, пользовательского интерфейса и управления данными, что упростит поддержку и последующие изменения в приложении.

## 4. Выбор базы данных:

Для учета ремонта автомобилей необходимо выбрать подходящую базу данных. Рекомендуется использовать реляционную базу данных, такую как MySQL или PostgreSQL, для удобного хранения информации о клиентах, автомобилях, выполненных работах и т.д. Также можно рассмотреть возможность использования ORM (Object-Relational Mapping) для более удобного взаимодействия с базой данных.

## 5. Средства для разработки пользовательского интерфейса:

Для создания удобного и интуитивно понятного пользовательского интерфейса можно использовать средства визуального программирования, такие как WPF или Windows Forms для C#. Кроме того, важно уделить внимание дизайну интерфейса и удобству навигации для пользователей.

#### 6. Тестирование и отладка:

Для обеспечения качественной работы программного модуля необходимо провести тестирование и отладку. Для автоматизации тестирования можно использовать встроенные в среду разработки Visual Studio инструменты тестирования или NUnit для C#. Рекомендуется также использовать инструменты для отслеживания ошибок, например, Sentry или BugSnag.

При выборе средств для проектирования решений для программного модуля учета ремонта автомобилей важно учитывать требования к функциональности, удобству использования и эффективности работы приложения. Вышеперечисленные средства и инструменты могут быть полезны при разработке данного модуля, обеспечивая удобство работы программиста и удобство использования конечными пользователями.

В первом рассмотрении всё программное обеспечение, в частности программы, работающие на компьютере, можно разделить на три категории:

1. Прикладные программы, непосредственно обеспечивающие выполнение необходимых пользователям работ.

2. Системные программы и утилиты, выполняющие различные вспомогательные и системные функции, к примеру:

- Разбивка жёсткого диска на разделы.
- Файловые менеджеры: Total Commander, Free Commander, Finder (macOS), Krusader (Linux).
- Проверка диска.
- Антивирусные программы.

- Программы-архиваторы.
- Программы для проверки устройств: AIDA, HDDScan и другие.
- Сетевые утилиты для проверки связи, например, ping (Windows).
- Программы для сканирования и распознавания текста, например, Foxit Reader.

3. Инструментальные программные системы, облегчающие процесс создания новых программ для компьютера. Инструментальные программные средства — это программы, которые используются в ходе разработки, корректировки или развития других прикладных или системных программ. По своему назначению они близки системам программирования.

## **2.2 Проектирование архитектуры приложения**

Архитектура информационной системы — это концептуальное описание её структуры и компонентов, а также принципов их взаимодействия. Она определяет, как система будет функционировать, какие технологии будут использоваться и как они будут интегрированы друг с другом.

Основные компоненты архитектуры информационной системы:

- Логическая архитектура: описывает функциональность системы, включая бизнес-процессы, правила обработки данных и алгоритмы.
- Физическая архитектура: определяет аппаратные и программные ресурсы, необходимые для реализации логической архитектуры, такие как серверы, базы данных, сетевые устройства и операционные системы.
- Программная архитектура: включает в себя выбор технологий и инструментов разработки, таких как языки программирования, фреймворки, библиотеки и т. д.

- Сетевая архитектура: описывает топологию сети, протоколы обмена данными и механизмы безопасности.
- Архитектура данных: определяет структуру и организацию данных в системе, включая базы данных, файлы и хранилища.

При разработке архитектуры информационной системы необходимо учитывать следующие аспекты:

- Масштабируемость: возможность расширения системы при увеличении нагрузки или объёма данных.
- Гибкость: способность к адаптации к изменяющимся требованиям бизнеса.
- Безопасность: обеспечение защиты данных от несанкционированного доступа и атак.
- Производительность: обеспечение быстрой и эффективной работы системы.
- Отказоустойчивость: способность системы продолжать работу при сбоях оборудования или программного обеспечения.

Для определения архитектуры информационной системы можно использовать различные подходы, такие как:

- Анализ требований: определение функциональных и нефункциональных требований к системе.
- Выбор технологий: выбор подходящих технологий для реализации системы на основе требований и ограничений проекта.
- Проектирование: разработка детального плана реализации системы с учётом выбранных технологий.
- Тестирование: проверка работоспособности системы и её соответствия требованиям.



После определения архитектуры информационной системы необходимо разработать документацию, которая будет служить основой для дальнейшей разработки и поддержки системы. Документация должна содержать подробное описание всех компонентов архитектуры, их функций и взаимодействий.

Проектирование архитектуры приложения — это процесс создания структуры и организации компонентов программного обеспечения, который определяет взаимодействие между различными частями системы.

Основные этапы проектирования архитектуры приложения:

1. Анализ требований: определение целей и задач приложения, а также его основных функций и возможностей.
2. Выбор архитектурных шаблонов: выбор подходящих шаблонов для реализации приложения (например, MVC, MVVM, микросервисы и т. д.).
3. Определение компонентов: разделение приложения на отдельные компоненты (модули, сервисы, библиотеки и т. п.), каждый из которых выполняет свою функцию.
4. Разработка интерфейсов: определение интерфейсов взаимодействия между компонентами.
5. Распределение обязанностей: назначение ролей и обязанностей каждому компоненту.
6. Обеспечение безопасности: разработка мер по защите данных и предотвращению несанкционированного доступа.
7. Масштабируемость и гибкость: проектирование с учётом возможности расширения функциональности и адаптации к изменяющимся требованиям.
8. Тестирование: проведение тестирования для проверки соответствия разработанной архитектуры требованиям и стандартам качества.
9. Документация: создание документации, описывающей архитектуру приложения.

Принципы проектирования архитектуры приложения включают в себя:

Разделение ответственности: каждый компонент должен выполнять только свои функции.

Слабая связанность: компоненты должны быть слабо связаны друг с другом, чтобы обеспечить гибкость и возможность изменения одного компонента без влияния на другие.

Инверсия зависимостей: зависимости должны быть направлены от абстракций к конкретным реализациям, что позволяет легко заменять одни компоненты другими.

Принцип открытости/закрытости: система должна быть открыта для расширения, но закрыта для изменений.

Единообразие: использование единых стандартов и подходов во всей системе.

Для проектирования архитектуры приложения можно использовать различные инструменты и методологии, такие как UML, BPMN, ArchiMate и другие. Выбор инструмента зависит от сложности проекта, требований к документации и других факторов.

Важно помнить, что проектирование архитектуры является непрерывным процессом, который продолжается на протяжении всего жизненного цикла разработки приложения.

## **2.3 ER-диаграмма**

ER-диаграмма (Entity-Relationship Diagram) — это графический способ представления структуры данных, который используется для проектирования баз данных. ER-диаграммы показывают отношения между сущностями и атрибутами в базе данных.

ER-диаграммы состоят из трёх основных элементов:

Сущности (Entity) — это основные элементы, которые представляют собой объекты или концепции в системе. Они могут быть реальными объектами, такими как клиенты, продукты или заказы, или абстрактными понятиями, такими как события или действия.

Атрибуты (Attribute) — это свойства сущностей. Атрибуты могут быть простыми, например, «имя» или «возраст», или сложными, состоящими из нескольких полей, таких как «адрес» или «номер телефона».

Связи (Relationship) — это отношения между сущностями. Связи определяют, как сущности взаимодействуют друг с другом и какие данные можно получить, объединив их. В ER-диаграммах используются три типа связей: один к одному (1:1), один ко многим (1:M) и многие ко многим (M:N).

В ER-диаграммах используются три основных типа связей:

«Один к одному» (1:1) — это связь, при которой одной записи в одной сущности соответствует только одна запись в другой сущности. Например, у одного студента может быть только один паспорт.

«Один ко многим» (1:M) — это связь, когда одной записи в первой сущности может соответствовать несколько записей во второй сущности. Пример: у одной группы студентов может быть много преподавателей.

«Многие ко многим» (M:N) — эта связь возникает, когда множественным записям из одной сущности соответствуют множественные записи из другой сущности и наоборот. Пример: студенты могут изучать несколько предметов, а предметы могут изучаться несколькими студентами.

Эти связи помогают определить, как данные будут храниться и обрабатываться в базе данных.

ER-диаграммы используются для моделирования данных на концептуальном уровне, прежде чем перейти к физическому проектированию базы данных. Они помогают понять структуру данных и связи между ними, что важно для разработки эффективной и надёжной системы.

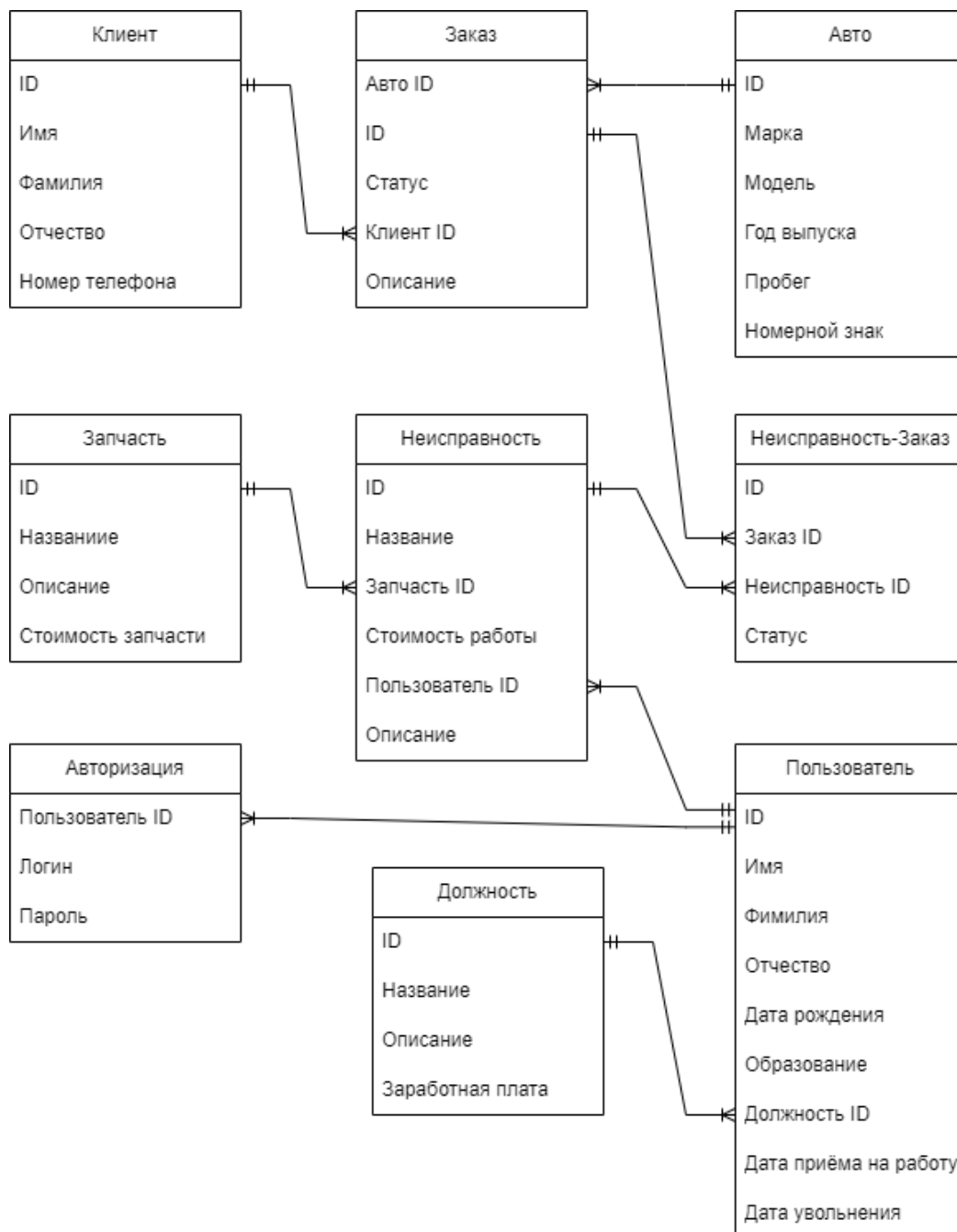


Рис. 1 «ER-диаграмма»

В ER-диаграмме (Рис.1) представлена структура программы для учёта ремонта автомобилей.

Сущности:

- Заказ

Атрибуты сущности:

- ID заказа
- ID авто
- ID клиента
- Статус
- Описание

- Клиент

Атрибуты сущности:

- ID клиента
- Имя
- Фамилия
- Отчество
- Номер телефона

- Авто

Атрибуты сущности:

- ID авто
- Марка
- Модель
- Год выпуска
- Пробег
- Номерной знак

- Неисправность

Атрибуты сущности:

- ID неисправности

- ID запчасти
- ID пользователя
- Название
- Описание
- Стоимость работы

- Неисправность-Заказ

Атрибуты сущности:

- ID неисправность-заказ
- ID заказа
- ID неисправности
- Статус

- Запчасть

Атрибуты сущности:

- ID запчасти
- Название
- Описание
- Стоимость

- Авторизация

Атрибуты сущности:

- ID пользователя
- Логин
- Пароль

- Должность

Атрибуты сущности:

- ID должности
- Название
- Описание
- Заработная плата

- Пользователь

#### Атрибуты сущности:

- ID пользователя
- ID должности
- Имя
- Фамилия
- Отчество
- Дата рождения
- Образование
- Дата приёма на работу
- Дата увольнения

#### Связи сущностей:

- Пользователь →Один ко многим→ Авторизация;
- Пользователь →Один ко многим→ Неисправность;
- Должность →Один ко многим→ Пользователь;
- Запчасть →Один ко многим→ Неисправность;
- Неисправность→Один ко многим→ Неисправность-Заказ;
- Заказ→Один ко многим→ Неисправность-Заказ;
- Авто→Один ко многим→ Заказ;
- Клиент→Один ко многим→ Заказ.

Впоследствии эта диаграмма поможет в создании баз данных, можно сказать, что ег-диаграмма является макетом для реально существующих таблиц и атрибутов в БД.

## 2.4. Методы моделирования системы

IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов.

IDEF0 позволяет наглядно представить структуру и функции системы, а также информационные и материальные потоки, которые проходят через неё. Это один из наиболее часто используемых методов создания функциональной модели, которая представляет собой структурированное изображение функций производственной системы или среды, информации и объектов, связывающих эти функции.

Основные элементы диаграммы IDEF0:

- Функциональный блок (Activity Box) — это основной элемент диаграммы IDEF0, который представляет собой функцию или процесс. Он преобразует входные данные в выходные и является ключевым элементом анализа бизнес-процессов.
- Вход (Input) — данные или материалы, которые необходимы для выполнения функции. Они поступают в функциональный блок и используются для создания выходных данных.
- Выход (Output) — результат выполнения функции, представленный в виде данных или продукции. Это то, что получается после преобразования входных данных функциональным блоком.
- Управление (Control) — правила, стандарты или процедуры, которыми руководствуется функция. Они определяют, как выполняется функция и какие результаты должны быть получены.
- Механизм (Mechanism) — ресурсы, необходимые для выполнения функции (персонал, оборудование, программное обеспечение и т. д.). Они обеспечивают выполнение функции и получение результатов.



Диаграммы IDEF0 позволяют наглядно представить структуру и функции системы, а также информационные и материальные потоки, проходящие через неё. Они широко используются в различных областях, таких как бизнес-анализ, проектирование систем и управление проектами. Диаграммы помогают понять и улучшить процессы, выявить проблемы и оптимизировать работу системы.

#### Преимущества IDEF0:

1. **Наглядность и простота восприятия.** Диаграммы IDEF0 позволяют наглядно представить структуру и функции системы, а также информационные и материальные потоки, проходящие через неё. Это упрощает понимание сложных процессов и систем.
2. **Универсальность.** IDEF0 может быть использован для моделирования различных типов систем, таких как бизнес-процессы, производственные процессы, информационные системы и т. д.
3. **Возможность анализа и оптимизации.** Диаграммы позволяют выявить проблемы и узкие места в системе, а также предложить пути их устранения. Это помогает улучшить работу системы и повысить её эффективность.
4. **Поддержка стандартизации.** IDEF0 является стандартом, что обеспечивает согласованность и понятность моделей, созданных разными специалистами.
5. **Простота документирования.** Диаграммы могут служить основой для создания документации по системе или процессу. Они содержат всю необходимую информацию о структуре, функциях и потоках данных.
6. **Обучение и передача знаний.** Модели IDEF0 могут использоваться для обучения новых сотрудников работе с системой или процессом. Они также могут быть использованы для передачи знаний между различными подразделениями компании.

7. Интеграция с другими методами. IDEF0 можно интегрировать с другими методами моделирования, такими как DFD (Data Flow Diagram) или ERD (Entity-Relationship Diagram), чтобы создать более полную модель системы.
8. Использование в различных областях. IDEF0 широко используется в бизнесе, промышленности, государственном секторе и других областях для анализа, проектирования и управления системами и процессами.

Основные этапы создания диаграммы IDEF0:

1. Определение границ системы. На этом этапе необходимо определить, какие функции и процессы будут включены в модель. Это может быть сделано на основе анализа требований к системе или её текущего состояния.
2. Идентификация основных функций. Необходимо определить основные функции, которые выполняет система. Эти функции будут представлены в виде функциональных блоков на диаграмме.
3. Анализ входов и выходов. Для каждой функции необходимо определить входные данные, которые поступают в функциональный блок, и выходные данные, которые получаются после выполнения функции.
4. Определение управления и механизма. Необходимо определить правила, стандарты или процедуры, которыми руководствуется функция, а также ресурсы, необходимые для её выполнения.
5. Создание диаграммы. На основе полученной информации необходимо создать диаграмму IDEF0. Диаграмма должна содержать функциональные блоки, входы, выходы, управление и механизм для каждой функции.
6. Проверка и уточнение. После создания диаграммы необходимо проверить её на соответствие требованиям и уточнить при необходимости.
7. Использование диаграммы. Диаграмма IDEF0 может использоваться для анализа, оптимизации и документирования системы. Она также может служить

основой для создания других моделей, таких как DFD (Data Flow Diagram) или ERD (Entity-Relationship Diagram).

Важно отметить, что создание диаграммы IDEF0 — это итеративный процесс. Возможно, потребуется несколько раз вернуться к предыдущим этапам, чтобы получить более точную и полную модель системы.

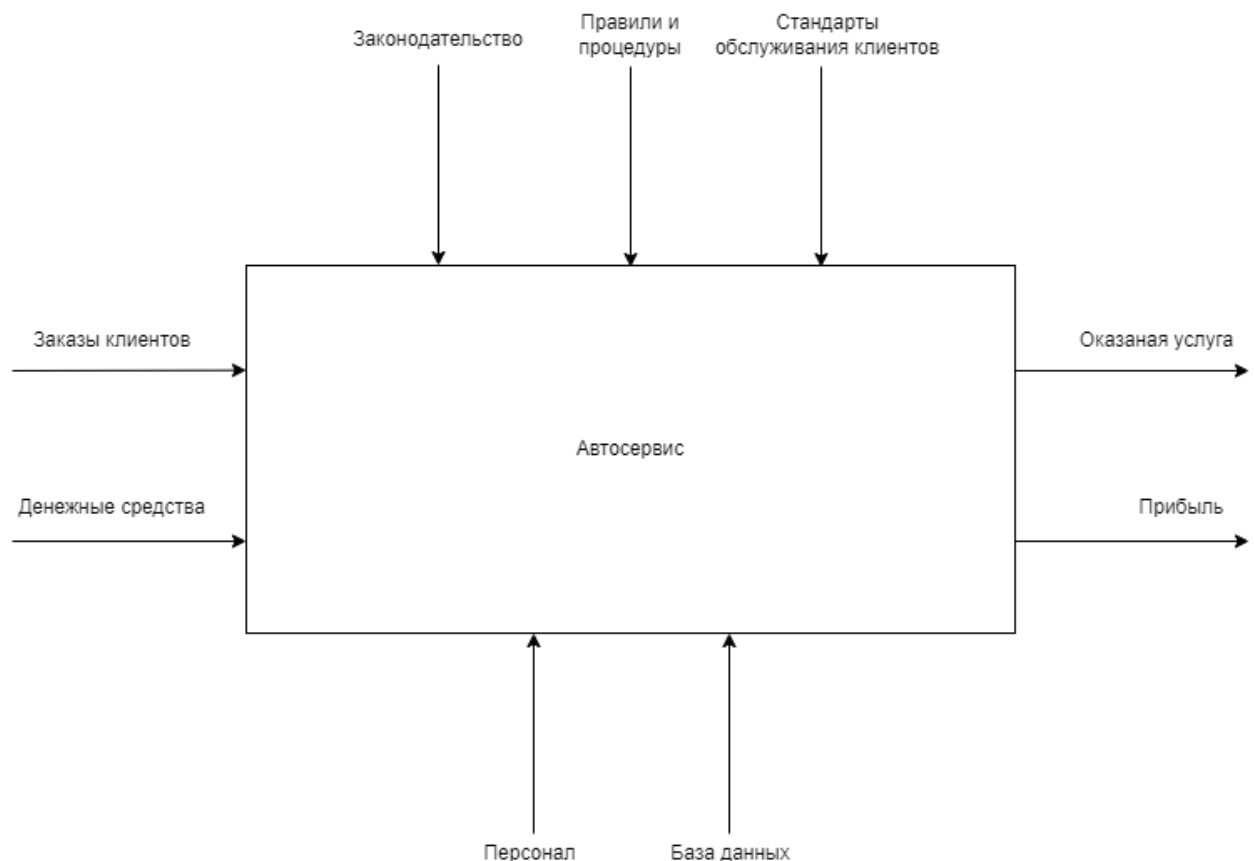


Рис. 2 «IDEF0 диаграмма»

Диаграмма вариантов использования (Use-Case) — это инструмент, который используется для описания взаимодействия между пользователями и системой. Она представляет собой графическое представление всех возможных вариантов использования системы.

Цель use-case: описать, как система будет использоваться в различных сценариях, чтобы обеспечить понимание требований к функциональности системы и её поведению.

Основные элементы use-case:

1. Название (Title) — краткое и понятное название, описывающее цель использования системы. Оно должно быть уникальным и легко запоминающимся. Например, «Оформление заказа», «Регистрация пользователя» или «Поиск товара».
2. Описание (Description) — подробное описание сценария использования, включая участников, их роли, действия и ожидаемые результаты. Описание должно быть понятным и не содержать технических деталей.
3. Участники: кто участвует в сценарии? Это могут быть пользователи, администраторы, разработчики и т. д.
4. Роли: какие роли выполняют участники? Например, пользователь может быть покупателем, а администратор — менеджером по продажам.
5. Действия: что делают участники в рамках сценария? Например, покупатель выбирает товары, добавляет их в корзину, заполняет форму оплаты и доставки и нажимает кнопку «Оформить заказ».
6. Ожидаемые результаты: какой результат должен быть достигнут в конце сценария? Например, система должна сформировать и отправить заказ на обработку.
7. Предусловия (Preconditions) — условия, которые должны быть выполнены перед началом сценария. Они определяют, при каких обстоятельствах сценарий может быть запущен. Например, для оформления заказа необходимо наличие товаров в корзине.
8. Постусловия (Postconditions) — результаты выполнения сценария. Они описывают, что происходит после завершения сценария. Например, после

оформления заказа система отправляет уведомление покупателю о статусе заказа.

9. Основной поток (Main flow) — последовательность действий, выполняемых в сценарии. Он описывает, как система реагирует на действия участников. Например, при оформлении заказа система проверяет наличие товаров, рассчитывает стоимость заказа с учётом скидок и налогов, формирует заказ и отправляет его на обработку.

10. Альтернативные потоки (Alternative flows) — возможные отклонения от основного потока, например, ошибки или исключения. Они описывают ситуации, когда сценарий не может быть выполнен успешно. Например, если пользователь вводит некорректные данные, система может выдать сообщение об ошибке и предложить исправить данные.

Эти элементы позволяют описать, как система будет использоваться в различных сценариях, чтобы обеспечить понимание требований к функциональности системы и её поведению.

Преимущества использования Use-Case:

1. Понимание требований к системе. Use-case позволяет описать, как система будет использоваться в различных сценариях, что помогает разработчикам и заказчикам лучше понять требования к функциональности системы и её поведению.

2. Простота и наглядность. Use-case представляет собой простой и понятный способ описания взаимодействия между участниками и системой. Это делает его удобным инструментом для обсуждения и согласования требований с заинтересованными сторонами.

3. Гибкость и масштабируемость. Use-case может быть адаптирован под различные бизнес-процессы и системы. Это позволяет использовать его для разработки новых проектов или модернизации существующих систем.

4. Поддержка тестирования. Use-case предоставляет основу для создания тестовых сценариев, которые могут быть использованы для проверки функциональности и поведения системы.
5. Улучшение коммуникации. Use-case способствует улучшению коммуникации между разработчиками, заказчиками и другими заинтересованными лицами, обеспечивая общее понимание требований и ожиданий от системы.
6. Снижение риска ошибок. Детальное описание сценариев использования помогает выявить потенциальные проблемы и риски на ранних этапах разработки, что может снизить вероятность ошибок и переделок в будущем.
7. Возможность повторного использования. Хорошо разработанные use-cases могут быть повторно использованы в других проектах, что экономит время и ресурсы при разработке новых систем.

Основные шаги создания Use-Case:

1. Определение цели проекта. Прежде чем начать создавать use-case, необходимо определить цель проекта и его основные задачи. Это поможет вам понять, какие сценарии использования системы будут наиболее важными.
2. Идентификация участников. Определите, кто будет взаимодействовать с системой. Это могут быть пользователи, администраторы, разработчики и т. д.
3. Описание сценариев использования. Для каждого участника опишите, как он будет использовать систему. Укажите, какие действия он будет выполнять и какие результаты ожидать.
4. Создание диаграмм. Используйте диаграммы для визуализации сценариев использования. Диаграммы помогут вам лучше понять структуру и логику системы.
5. Уточнение деталей. Добавьте в use-case дополнительные детали, такие как предусловия, постусловия, основной поток и альтернативные потоки.

6. **Согласование с заинтересованными сторонами.** Обсудите use-case с заказчиками, разработчиками и другими заинтересованными лицами. Убедитесь, что все понимают требования к системе.
7. **Документирование.** Задokumentируйте use-case в виде формального документа. Это позволит сохранить информацию о системе и обеспечить её доступность для всех участников проекта.
8. **Тестирование.** Создайте тестовые сценарии на основе use-case. Тестирование поможет убедиться, что система соответствует требованиям и ожиданиям пользователей.
9. **Обновление.** По мере развития проекта обновляйте use-case, чтобы отразить изменения в требованиях и функциональности системы.

Эти шаги помогут вам создать use-case, который будет служить основой для разработки и тестирования системы.

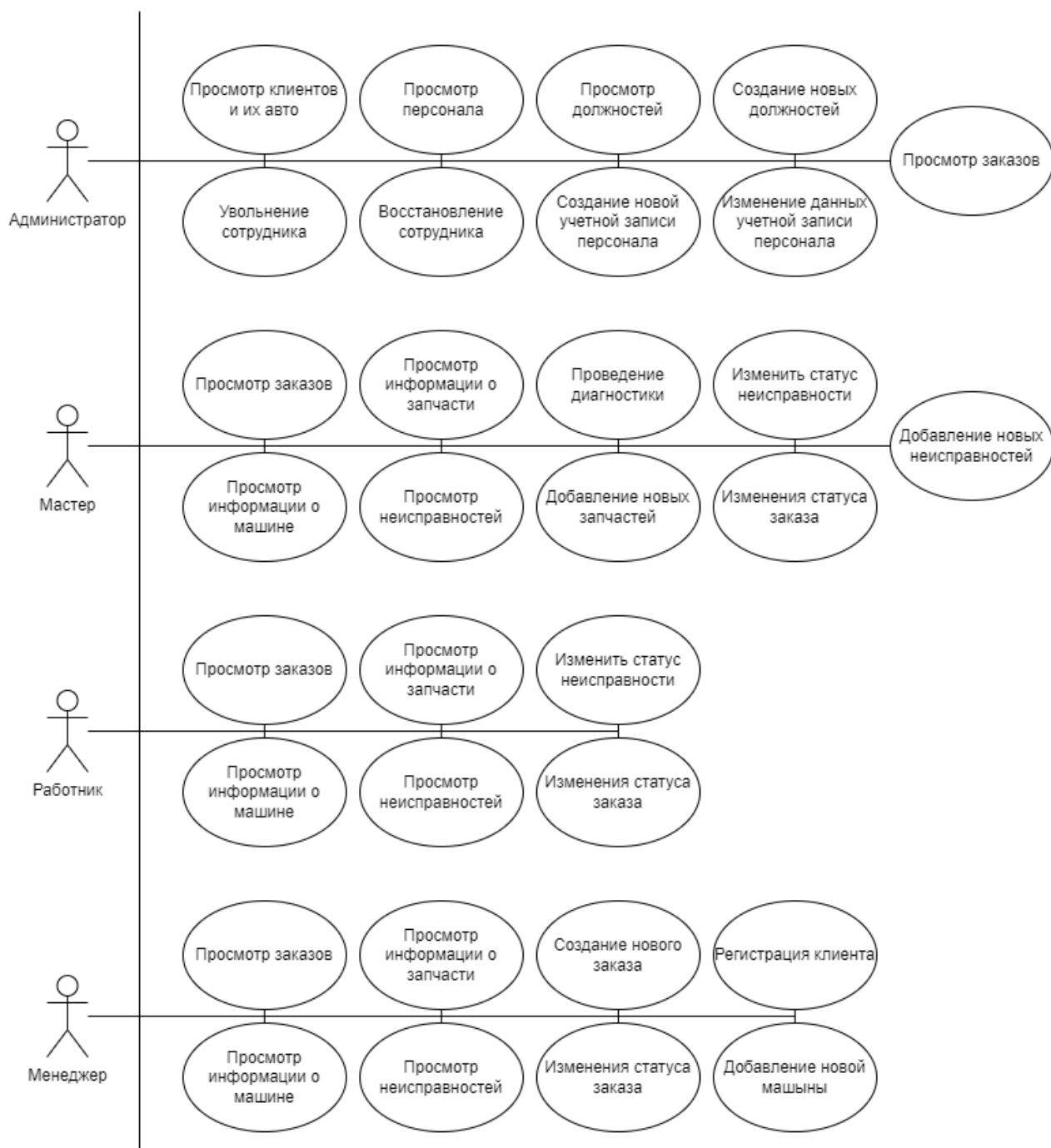


Рис. 3 «USE-CASE диаграмма»

## 2.5 Диаграмма переходов состояний

Диаграмма переходов состояний (UML state machine diagram) — это диаграмма, которая используется для моделирования поведения системы или объекта в зависимости от его состояния. Она показывает возможные переходы



между различными состояниями и условия, при которых эти переходы происходят.

Диаграммы переходов состояний используются для моделирования сложных систем с множеством состояний и переходов. Они помогают понять логику работы системы и выявить возможные проблемы.

Для создания диаграммы переходов состояний можно использовать инструменты UML-моделирования, такие как Visual Paradigm, Enterprise Architect, StarUML и другие.

Основные элементы диаграммы переходов состояний:

1. Состояние (State) — определённое состояние системы или объекта, в котором он может находиться. Состояние может быть активным или пассивным. Активное состояние означает, что система выполняет какие-либо действия, а пассивное — что она просто находится в ожидании.
2. Переход (Transition) — изменение состояния системы или объекта при выполнении определённых условий. Переход может быть инициирован событием или действием. Событие — это внешнее воздействие на систему, которое может вызвать переход из одного состояния в другое. Действие — это внутреннее событие, которое происходит внутри системы и может привести к переходу из одного состояния в другое.
3. Событие (Event) — событие, которое может вызвать переход из одного состояния в другое. События могут быть внутренними или внешними. Внутренние события происходят внутри системы, а внешние — вне её.
4. Действие (Action) — действие, которое выполняется при переходе из одного состояния в другое. Действия могут быть связаны с изменением состояния системы или с выполнением каких-либо операций.
5. Начальное состояние (Start State) — состояние, с которого начинается работа системы. Начальное состояние может быть единственным или одним из нескольких возможных начальных состояний.

6. Конечное состояние (Final State) — состояние, в которое система переходит после завершения работы. Конечное состояние также может быть единственным или одним из нескольких возможных конечных состояний.
7. Альтернативный переход (Alternative Transition) — переход, который может произойти при разных условиях. Альтернативные переходы используются для моделирования ситуаций, когда система может перейти в одно из нескольких возможных состояний в зависимости от условий.
8. Параллельный переход (Parallel Transition) — одновременный переход нескольких объектов в одно и то же состояние. Параллельные переходы используются для моделирования ситуаций, когда несколько объектов могут одновременно перейти в одно и то же состояние в результате одного и того же события.
9. Ортогональный переход (Orthogonal Transition) — переход между состояниями, которые не связаны друг с другом напрямую. Ортогональные переходы используются для моделирования сложных систем, в которых объекты могут переходить в различные состояния в результате различных событий.

Основные шаги создания диаграммы переходов состояний (UML state machine diagram):

1. Определение системы или объекта, для которого будет создаваться диаграмма. Необходимо чётко определить, что именно будет моделироваться с помощью диаграммы. Это может быть система, процесс, объект или что-то ещё.
2. Выделение основных состояний системы или объекта. Каждое состояние должно иметь чёткое определение и описание.
3. Определение возможных переходов между состояниями. Для каждого перехода необходимо указать событие или условие, которое его инициирует.
4. Добавление событий и действий к переходам. События могут быть внутренними или внешними, а действия могут быть связаны с изменением состояния системы или с выполнением каких-либо операций.

5. Указание начального и конечного состояний. Начальное состояние — это состояние, с которого начинается работа системы, а конечное состояние — это состояние, в которое система переходит после завершения работы.
6. Проверка диаграммы на полноту и непротиворечивость. Диаграмма должна полностью описывать поведение системы или объекта и не содержать противоречий.
7. Оформление диаграммы. Диаграмму можно оформить в соответствии с требованиями UML или в соответствии со стандартами вашей организации.
8. Использование диаграммы для анализа и проектирования системы. Диаграмма может использоваться для выявления проблем в системе, для разработки требований к системе или для проектирования архитектуры системы.

Инструменты для создания диаграмм переходов состояний:

- Visual Paradigm for UML;
- Enterprise Architect;
- StarUML;
- ArgoUML.

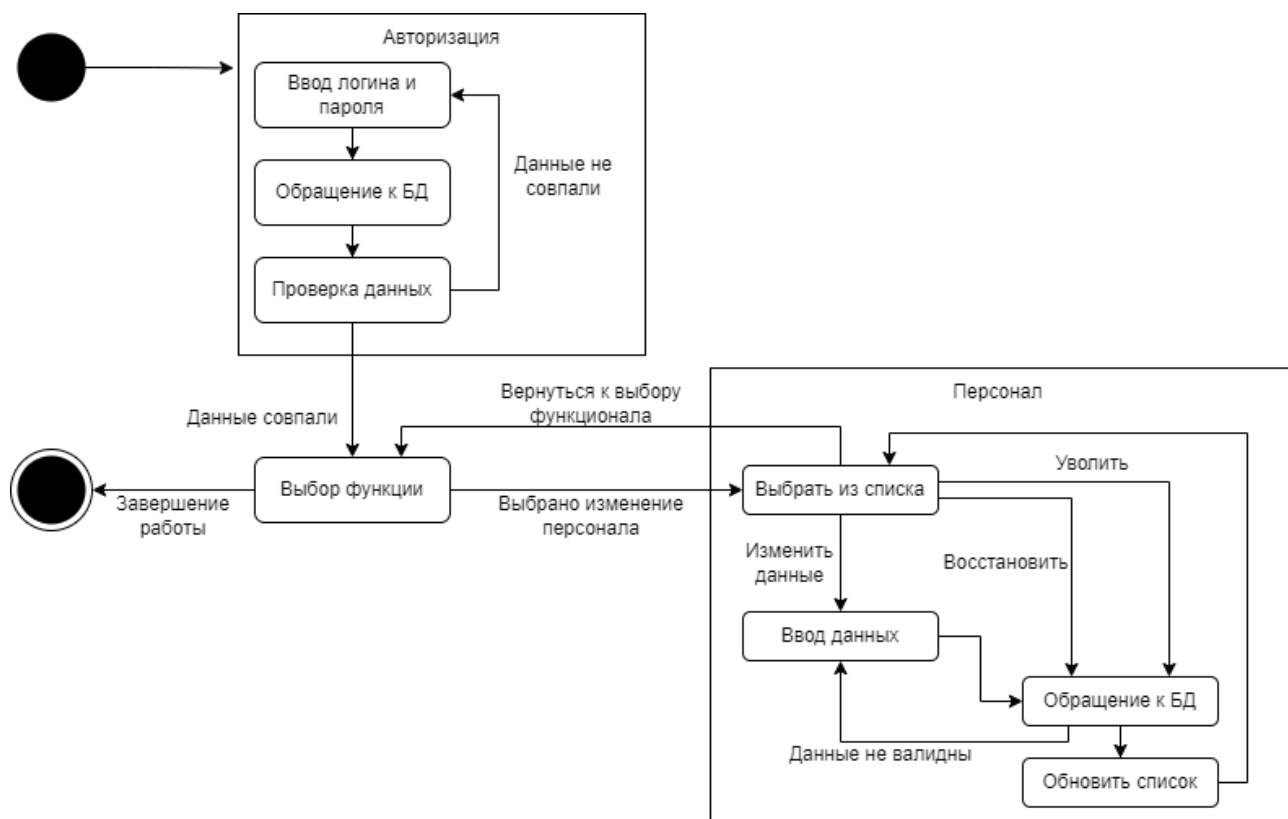


Рис. 4 « Диаграмма переходов состояния предметной области»

## 2.6 Макеты интерфейса приложения

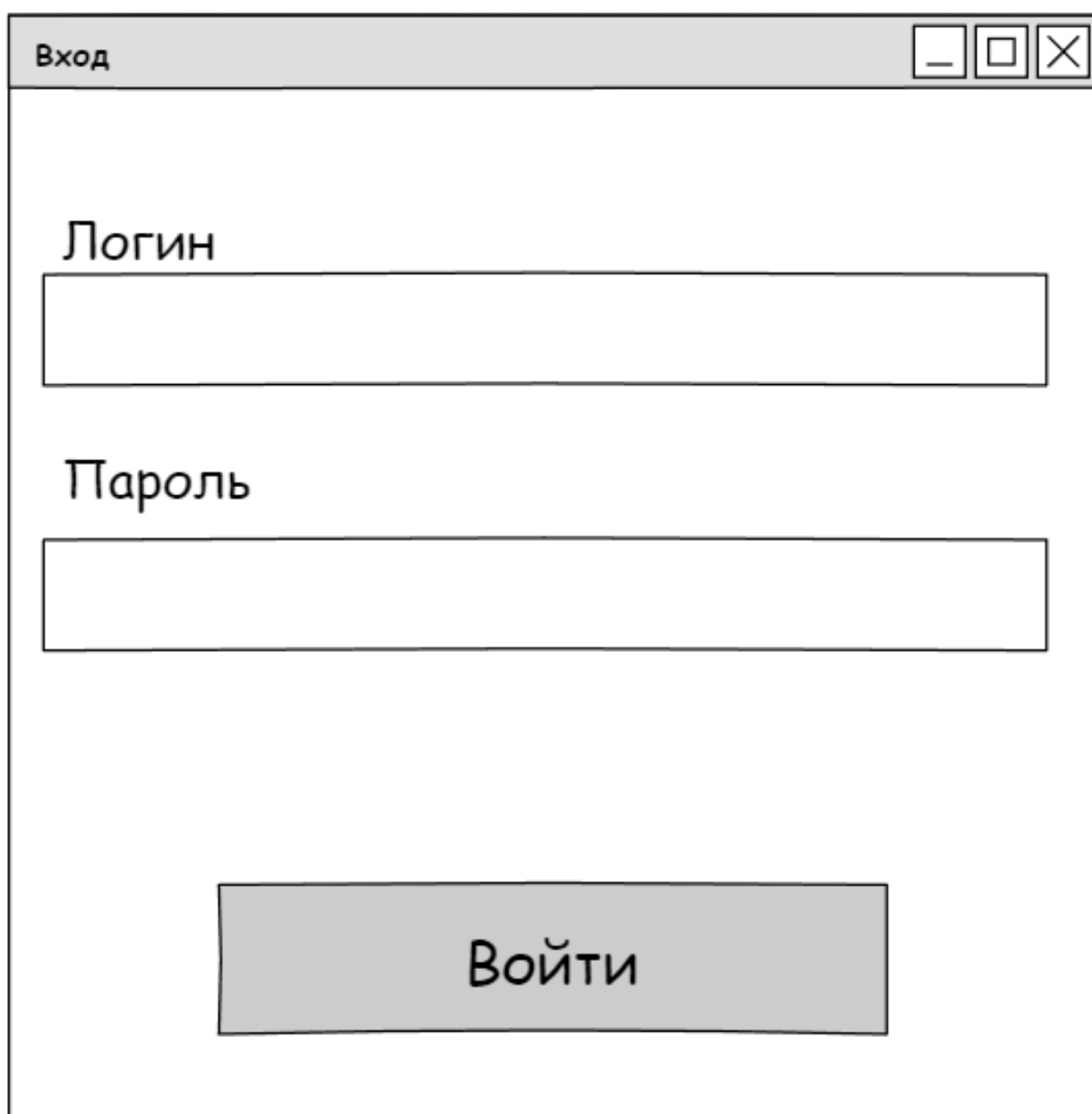
После проектирования архитектуры приложения и его ER-диаграммы, необходимо разработать пользовательский интерфейс программы.

Пользовательский интерфейс — средства, позволяющие пользователю эффективно взаимодействовать с информационной системой удобным для себя образом.

Визуально, пользовательский интерфейс представляет собой набор элементов управления (кнопок, к примеру) для навигации между окнами, в которых отображаются данные информационной системы. Современный интерфейс у многих программ визуально схож.

Разработанная информационная система будет содержать следующие окна:

Окно авторизации – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют пользователю ввести необходимые для авторизации в системе данные, такие как логин и пароль. Система авторизации является важной частью в системе безопасности в информационной системе, которая не позволяет получить доступ к определённым функциям или информации не зарегистрированным в системе пользователям.



The image shows a graphical user interface for a login window. At the top, there is a title bar with the text 'Вход' (Login) on the left and three standard window control buttons (minimize, maximize, close) on the right. The main area of the window contains two text input fields. The first field is preceded by the label 'Логин' (Login) and the second by 'Пароль' (Password). Below these fields, centered at the bottom, is a large rectangular button with the text 'Войти' (Login) in a bold, sans-serif font.

Рис. 5 «Окно авторизации»

Основное окно администратора– это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют администратору выбрать из перечня необходимый функционал. Интерфейс состоит из четырёх кнопок:

- Клиенты – просмотр базы клиентов;
- Персонал – просмотр и манипуляция с данными рабочего персонала;
- Заказы – просмотр базы заказов;
- Должности – просмотр и возможность добавления новых должностей.

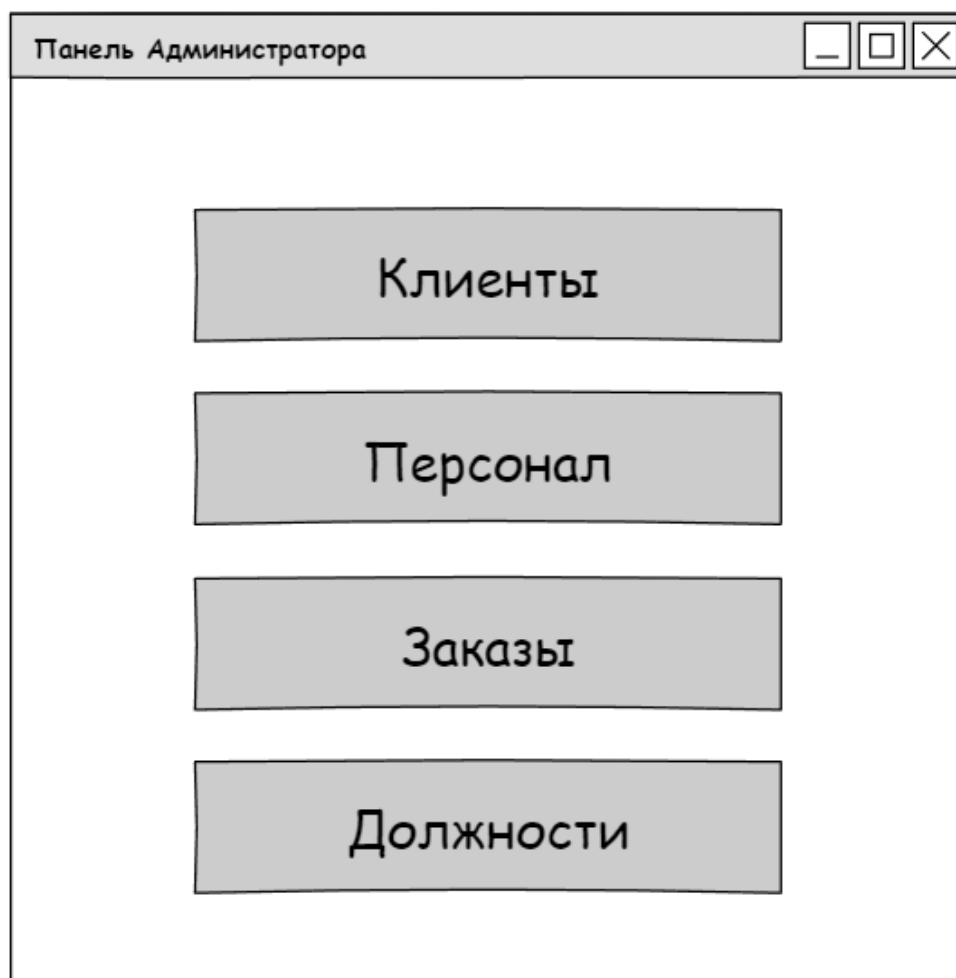
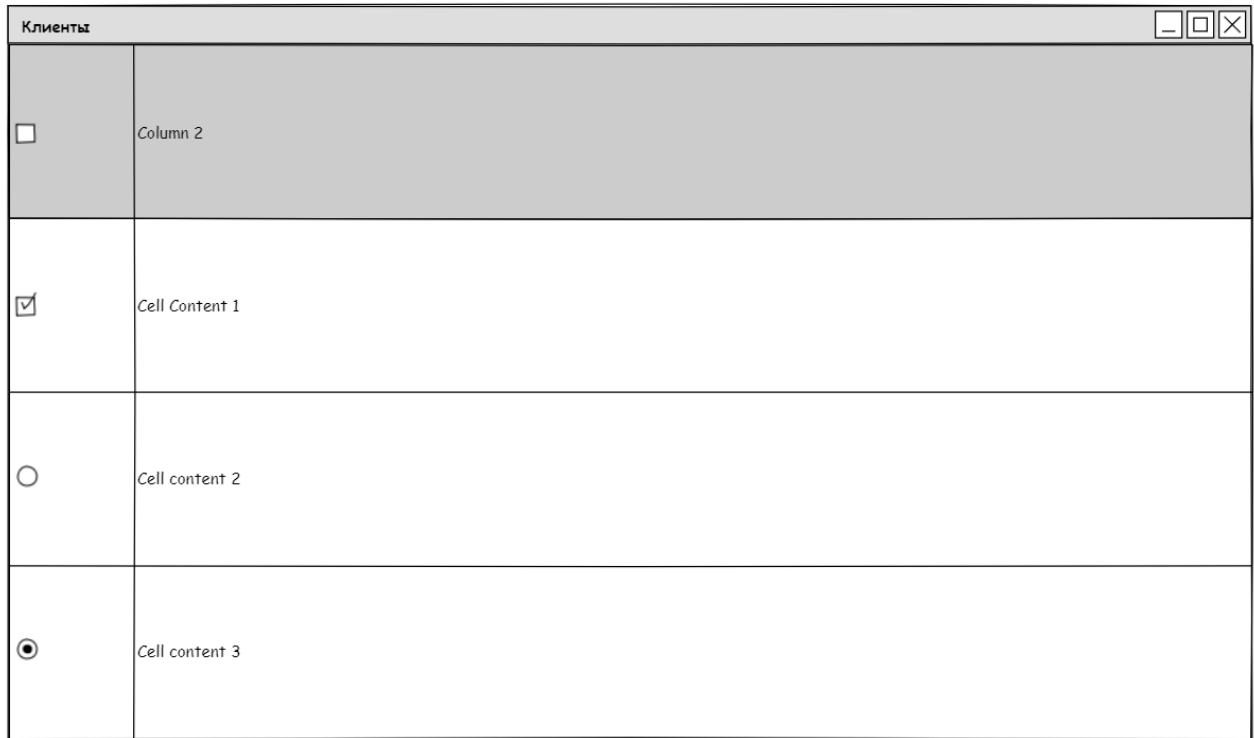


Рис. 6 «Окно администратора»

Окно со списком клиентов – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют просмотреть список всех клиентов сервисном центре в виде таблицы.



Клиенты	
<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 7 «Окно клиенты (администратора)»

Окно со списком персонала – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют просмотреть список персонала, работающего в сервисном центре в виде таблицы. А также возможность добавить нового сотрудника, изменить данные существующего, уволить и восстановить сотрудника. Кнопка обновить позволяет обновить данные в таблице.

Персонал	
<div> <div>Добавить</div> <div>Изменить</div> <div>Уволить</div> <div>Восстановить</div> <div>Обновить</div> </div>	
<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 8 «Окно персонал»

Окно добавления нового сотрудника – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют ввести данные нового сотрудника для последующей проверки и занесения в базу. Содержит такие поля как:

- Фамилия;
- Имя;
- Отчество;
- Дата рождения;
- Образование;
- Должность;
- Логин;
- Пароль.



Добавить

Фамилия

Имя

Отчество

Дата рождения

Образование

Должность

Логин

Пароль

Применить

Рис. 9 «Окно добавления нового сотрудника»

Окно изменения данных сотрудника – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют ввести новые данные сотрудника для последующей проверки и занесения в базу. Содержит такие поля как:

- Фамилия;
- Имя;
- Отчество;
- Дата рождения;
- Образование;
- Должность;
- Логин;
- Пароль.

Изменить

Фамилия

Имя

Отчество

Дата рождения

Образование

Должность

Логин

Пароль

Применить

Рис. 10 «Окно изменения данных сотрудника»

Окно со списком заказов – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют администратору просмотреть все заказы.

<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 11 «Окно заказы (администратора)»

Окно со списком должностей – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют администратору просмотреть все существующие в системе должности, а так же перейти на окно добавления новой должности.

Должности	
<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 12 «Окно должности»

Окно добавления новой должности – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют администратору ввести необходимые данные для добавления новой должности в информационную систему. Содержит такие поля как:

- Название;
- Заработная плата;
- Технический индекс доступа;
- Описание.

Новая должность

Название

Зарботная плата

Технический индекс доступа

Описание

Добавить

Рис. 13 «Окно добавления новой должности»

Окно менеджера – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют менеджеру отслеживать статус заказов, а так же манипулировать ими. Интерфейс менеджера позволяет перейти к окну создания нового заказа. Так же менеджер может получить подробную информацию о неисправностях автомобиля определённого заказа, получить информацию о машине и обновить данные в таблице заказов.

<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 14 «Окно менеджера»

Окно добавления нового заказа – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют менеджеру ввести данные о клиенте, его автомобиле и жалобах клиента на автотранспорт. Интерфейс содержит такие поля как:

- Клиент:
  - Фамилия;
  - Имя;
  - Отчество;
  - Номер телефона;
- Машина:
  - Марка;
  - Модель;
  - Год выпуска;
  - Пробег;
  - Номерной знак;
- Проблема.

Добавить заказ

Клиент

Фамилия

Имя

Отчество

Номер телефона

Машина

Марка

Модель

Год выпуска

Пробег

Номерной знак

Проблема

Рис. 15 «Окно добавления нового заказа»

Окно мастера – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют мастеру вносить данные о проделанной работе изменяя статус неисправности и заказа. Интерфейс позволяет получить информацию о текущих заказах, неисправностях автомобиля выбранного заказа, получить информацию о машине заказа, получить информацию о запчасти необходимой для устранения неисправности. Также интерфейс окна позволяет принудительно обновить данные таблицы заказов, данные неисправностей автомобиля заказа в таблице обновляются автоматически при выборе заказа. Кнопка диагностика на интерфейсе окна позволяет перейти к окну диагностики.

Заказы - Мастер

Обновить

Диагностика

Информация о машине

Изменить статус на Готово

<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Информация о запчасти

Изменить статус на Готово

<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 16 «Окно мастера»

Окно диагностики – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют мастеру вносить данные диагностики выбирая их в списке всех неисправностей и двойным кликом переносить в список неисправностей диагностируемого автотранспорта. Интерфейс окна содержит в себе текстовое поле, которое помогает в поиске нужной неисправности среди всех, проще говоря это текстовый фильтр. Так же интерфейс окна отображает подробную информацию о, выделенной в списке всех неисправностей, неисправности.

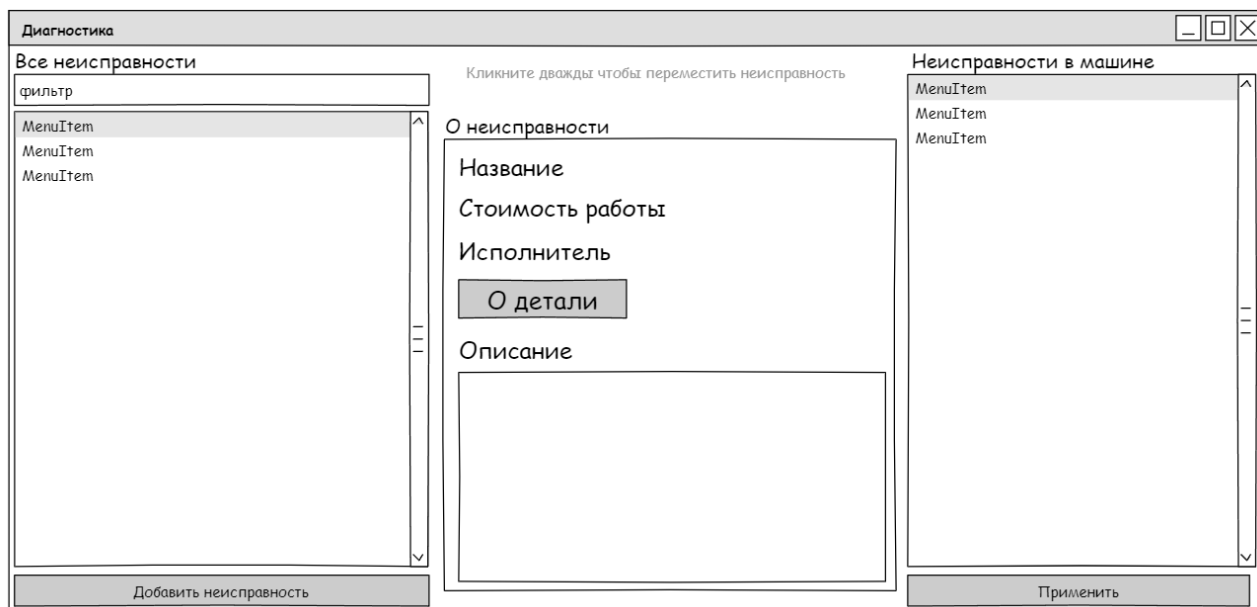


Рис. 17 «Окно диагностики»

Окно добавления новой неисправности – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют мастеру вводить информацию для добавления нового типа неисправности в систему. Интерфейс содержит такие поля как:

- Название;
- Описание;
- Стоимость работы;
- Запчасть;
- Исполнитель.



Новая неисправность

Название

Описание

Стоимость работы

Запчасть ▼

Исполнитель ▼

Добавить запчасть

Сохранить

Рис. 18 «Окно добавления новой неисправности»

Окно добавления новой запчасти – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют мастеру вводить информацию для добавления новой запчасти в базу данных информационной системы. Интерфейс содержит такие поля как:

- Название;
- Описание;
- Стоимость.

Новая запчасть

Название

Описание

Стоимость

Добавить

Рис. 19 «Окно добавления новой запчасти»

Окно исполнителя – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют работнику (исполнителю) вносить данные о проделанной работе изменяя статус неисправности и заказа. Интерфейс позволяет получить информацию о текущих заказах, неисправностях автомобиля выбранного заказа, получить информацию о машине заказа, получить информацию о запчасти необходимой для устранения неисправности. Также интерфейс окна позволяет принудительно обновить данные таблицы заказов, данные неисправностей автомобиля заказа в таблице обновляются автоматически при выборе заказа.

Заказы -	
<input type="button" value="Обновить"/>	
<input type="button" value="Информация о машине"/>	
<input type="button" value="Изменить статус на Готово"/>	
<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Информация о запчасти	
<input type="button" value="Изменить статус на Готово"/>	
<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 20 «Окно исполнителя»

Окно о неисправностях – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют получить информацию о неисправностях автомобиля определённого заказа. Интерфейс окна позволяет получить дополнительную информацию о запчасти необходимой для устранения неисправности.

О неисправностях	
Информация о запчасти	
<input type="checkbox"/>	Column 2
<input checked="" type="checkbox"/>	Cell Content 1
<input type="radio"/>	Cell content 2
<input checked="" type="radio"/>	Cell content 3

Рис. 21 «Окно о неисправностях»

Окно о запчасти – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют получить информацию об определённой запчасти.

О запчасти
<div>Название</div> <div>Стоимость</div> <div>Описание</div> <div></div>

Рис. 22 «Окно о запчасти»

Окно о машине – это интерфейс, который содержит в себе элементы интерфейса, которые в свою очередь позволяют получить информацию об определённой машине.

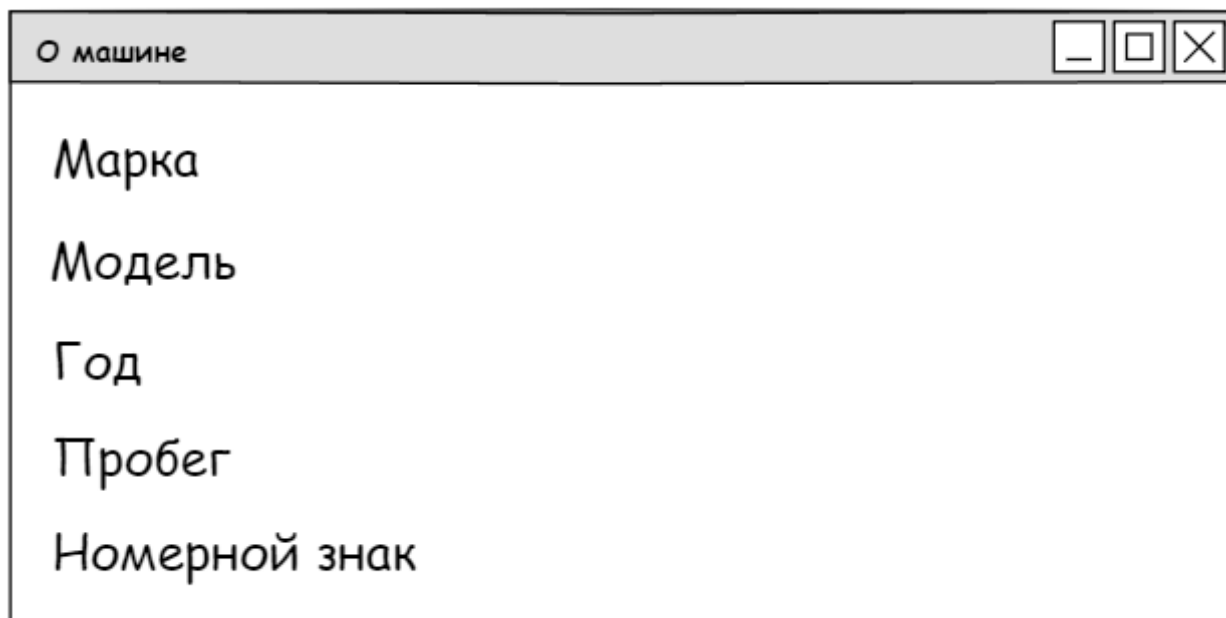


Рис. 23 «Окно о машине»

### **3. СПЕЦИАЛЬНАЯ ЧАСТЬ**

#### **3.1 Формирование требований к ПО**

Формирование требований к ПО — это процесс определения и документирования характеристик, функций и ограничений программного обеспечения (ПО), которые должны быть реализованы в соответствии с потребностями пользователей и бизнес-целями.

Основные этапы формирования требований:

Анализ потребностей: определение целей и задач проекта, выявление основных пользователей и их потребностей.

Сбор информации: проведение интервью, опросов, анализ существующих систем и документов.

Классификация требований: разделение требований на функциональные, нефункциональные и другие виды.

Определение приоритетов: ранжирование требований по важности и срочности.

Спецификация требований: создание документа, описывающего требования к ПО.

Проверка и согласование: проверка соответствия требований потребностям пользователей и целям проекта.

Утверждение: окончательное утверждение требований заказчиком или заинтересованными сторонами.

Управление изменениями: отслеживание изменений в требованиях в процессе разработки и после выпуска продукта.

Шаги для формирования требований к ПО:

Определить цели и задачи проекта: что должно делать ПО? Какие проблемы оно должно решать?

Выявить основные функции: какие действия должен выполнять пользователь с помощью ПО?

Установить ограничения: какие ресурсы доступны для разработки? Какие технические ограничения существуют?

Описать нефункциональные требования: производительность, безопасность, надёжность, удобство использования и т. д.

Провести анализ требований: проверить полноту, непротиворечивость и выполнимость требований.

Создать документ спецификации требований: описать все требования в структурированном виде.

Для успешного формирования требований необходимо вовлечь всех заинтересованных сторон, включая разработчиков, тестировщиков, пользователей и заказчиков. Это поможет создать чёткое и понятное описание того, что должно быть реализовано в ПО, и обеспечить его соответствие потребностям пользователей.

Важно помнить, что требования могут меняться в процессе разработки, поэтому необходимо регулярно пересматривать и обновлять их. Также стоит учитывать, что формирование требований — это итеративный процесс, который может потребовать нескольких циклов анализа, обсуждения и согласования.

### **3.2 Разработка интерфейса ПО**

Разработка интерфейса ПО на WPF C# — это процесс создания графического интерфейса пользователя (GUI) для программного обеспечения с использованием технологии Windows Presentation Foundation (WPF) и языка программирования C#.

Вот основные шаги, которые могут помочь вам разработать интерфейс ПО на WPF C#:

Изучение основ WPF: перед началом разработки необходимо изучить основы WPF, такие как элементы управления, привязка данных, стили и шаблоны. Это поможет вам понять, как работает технология и какие возможности она предоставляет.

Создание проекта: создайте новый проект в Visual Studio или другой среде разработки, выбрав шаблон WPF Application. Это создаст базовую структуру проекта, включая файл MainWindow.xaml, который будет содержать основной пользовательский интерфейс.

**Добавление элементов управления:** начните добавлять элементы управления в пользовательский интерфейс, такие как кнопки, текстовые поля, списки и т.д. Вы можете использовать дизайнер XAML в Visual Studio для визуального редактирования интерфейса.

**Привязка данных:** используйте привязку данных для связывания элементов управления с данными из модели. Это позволит автоматически обновлять интерфейс при изменении данных.

**Стили и шаблоны:** создавайте стили и шаблоны для элементов управления, чтобы обеспечить единообразный внешний вид приложения. Это также может упростить процесс разработки.

**Тестирование:** регулярно проводите тестирование интерфейса, чтобы убедиться, что он работает правильно и соответствует требованиям.

**Оптимизация производительности:** оптимизируйте производительность интерфейса, используя такие методы, как кэширование данных и оптимизация рендеринга.

**Документация:** документируйте свой код и дизайн интерфейса, чтобы другие разработчики могли легко понять и поддерживать ваше решение.

**Отладка и устранение ошибок:** отлаживайте свой код, чтобы найти и исправить ошибки, а также улучшить функциональность интерфейса.

**Доработка и улучшение:** после завершения основных функций продолжайте дорабатывать и улучшать интерфейс, основываясь на обратной связи от пользователей и собственных наблюдениях.

Помните, что разработка интерфейса — это непрерывный процесс, требующий постоянного внимания и усилий. Важно постоянно анализировать и улучшать свой продукт, чтобы он соответствовал требованиям пользователей и рынка.

XAML (eXtensible Application Markup Language) — это язык разметки, который используется для создания пользовательских интерфейсов и декларативного описания объектов в приложениях. XAML является ключевым компонентом

платформы .NET и широко применяется в таких технологиях, как WPF (Windows Presentation Foundation), Xamarin и других.

Основные особенности XAML:

Декларативное описание интерфейса: XAML позволяет описывать внешний вид и поведение пользовательского интерфейса в виде XML-разметки. Это упрощает процесс разработки и делает его более наглядным.

Поддержка привязки данных: XAML поддерживает привязку данных между элементами управления и данными из модели. Это позволяет автоматически обновлять интерфейс при изменении данных.

Стили и шаблоны: XAML предоставляет возможность создавать стили и шаблоны для элементов управления, что обеспечивает единообразный внешний вид приложения.

Разделение логики и представления: XAML отделяет логику приложения от его представления, что упрощает разработку и поддержку кода.

Расширяемость: XAML можно расширять с помощью пользовательских элементов управления и тем самым настраивать внешний вид и функциональность приложения.

Интеграция с другими технологиями: XAML интегрируется с различными технологиями, такими как WPF, Xamarin, Silverlight и другими, что делает его универсальным инструментом для разработки пользовательских интерфейсов.

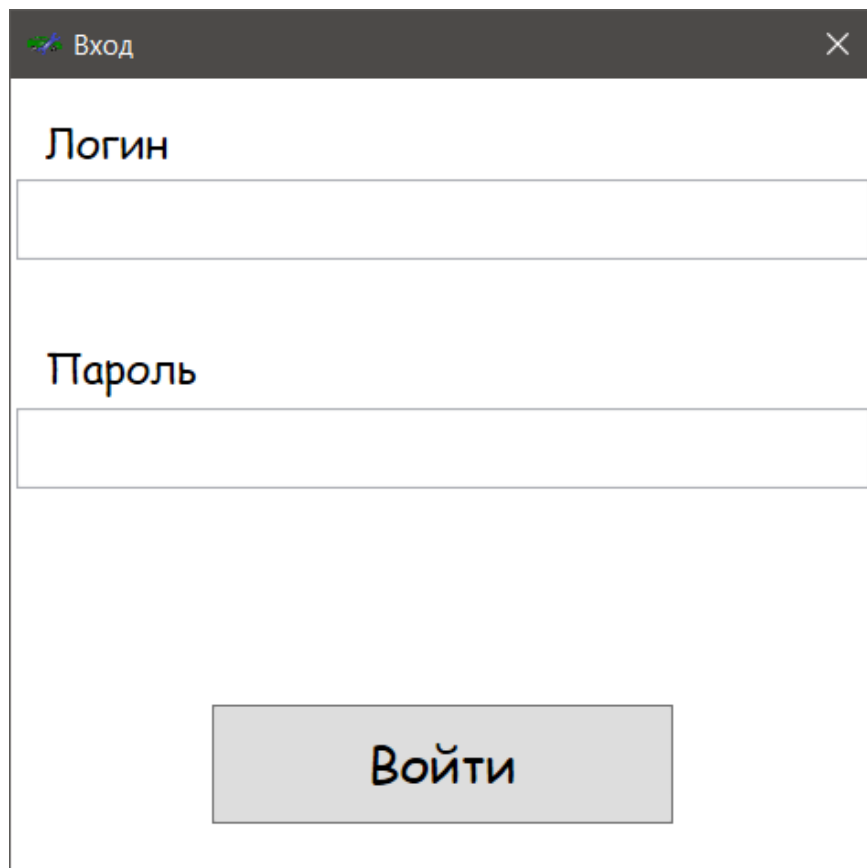
Интуитивно понятный синтаксис: XAML имеет интуитивно понятный синтаксис, основанный на XML, что облегчает его изучение и использование.

Возможность использования визуального редактора: многие среды разработки, такие как Visual Studio, предоставляют визуальный редактор для работы с XAML, что ускоряет процесс создания и редактирования пользовательского интерфейса.

Поддержка анимации и переходов: XAML позволяет создавать анимацию и переходы между состояниями пользовательского интерфейса, делая приложение более привлекательным и интерактивным.



В целом, XAML представляет собой мощный инструмент для создания современных и гибких пользовательских интерфейсов, которые легко поддерживать и развивать.



The image shows a screenshot of a login window. The window has a dark gray title bar with the text "Вход" (Login) and a close button (X). The main content area is white and contains two input fields. The first field is labeled "Логин" (Login) and the second field is labeled "Пароль" (Password). Below the input fields is a large, light gray button with the text "Войти" (Login).

Рис. 24 «Окно авторизации»

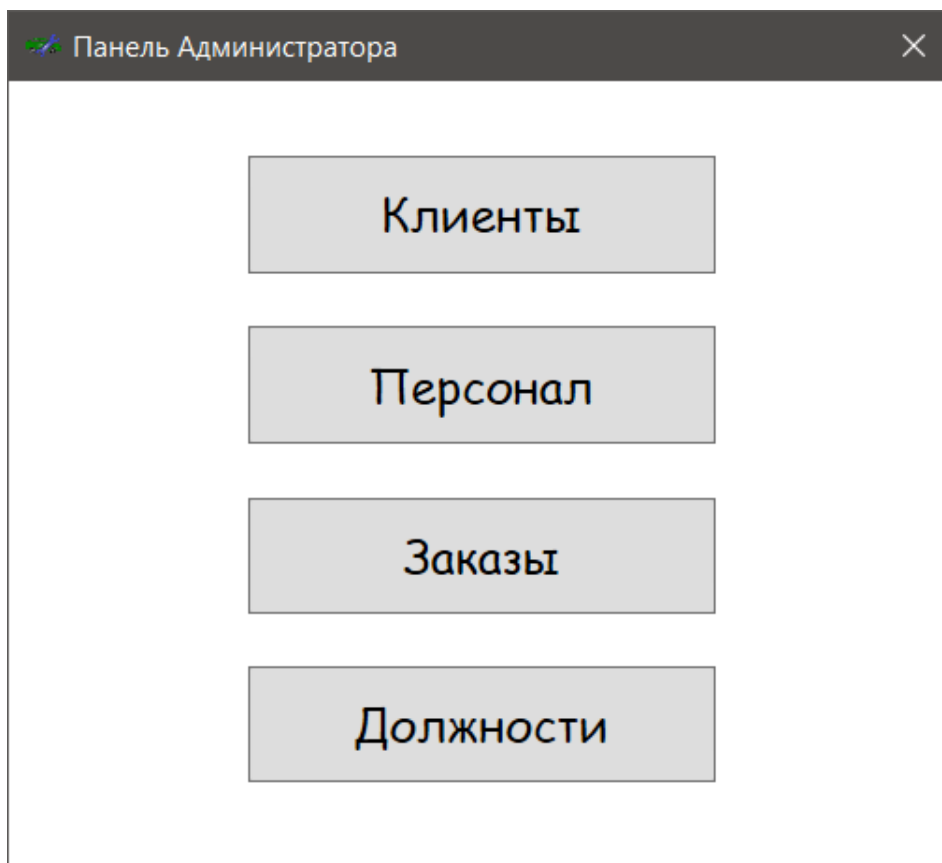


Рис. 25 «Окно панель администратора»

Клиенты								
Фамилия	Имя	Отчество	Номер телефона	Марка	Модель	Год производства	Пробег	Номерной знак
Петров	Иван	Яковлевич	+7 987 654 32-10	Audi	A8	1994	60937 km	P003BO37
Смирнов	Пётр	Дмитриевич	+7 942 532 34 29	Toyota	Corolla	2008	83886 km	A038EK66
Петров	Иван	Саватеевич	+7 984 665 44-00	ЗАЗ	968M	1982	125083 km	P353MO96
Сидоров	Сергей	Николаевич	+7 945 123 44 84	Kia	Sorento	2019	239003 km	O234TA799

Рис. 26 «Окно клиенты»

Персонал										
<div>Добавить</div> <div>Изменить</div> <div>Уволить</div> <div>Восстановить</div> <div>Обновить</div>										
Фамилия	Имя	Отчество	Дата рождения	Образование	Должность	Дата принятия на работу	Дата увольнения	Логин	Пароль	Зарботная плата
Иванов	Давид	Вячеславович	1978.01.04	Высшее	Администратор	2013.05.12	Не уволен	l1	p1	100000 руб.
Пеньков	Тарас	Петрович	2024.05.29	Среднее	Администратор	2024.05.29	2024.05.29	12345678	12345678	100000 руб.
Малышев	Юрий	Павлович	1984.03.21	Высшее	Менеджер	2018.08.09	Не уволен	l2	p2	80000 руб.
Панов	Алексей	Иванович	1992.02.05	Среднее	Мастер	2020.11.20	Не уволен	l3	p3	85000 руб.
Зубов	Роман	Кириллович	1985.07.01	Высшее	Мастер	2024.05.30	Не уволен	zubov1985	zubov1985	85000 руб.
Малинин	Даниил	Святославович	1987.09.23	Среднее	Кузовщик	2020.11.20	Не уволен	l4	p4	70000 руб.
Тарасов	Михаил	Павлович	1989.06.19	Высшее	Электрик	2017.03.18	Не уволен	l5	p5	70000 руб.

Рис. 27 «Окно персонал»

Заказы										
Статус заказа	Описание проблемы	Фамилия	Имя	Отчество	Номер телефона	Марка	Модель	Год производства	Пробег	Номерной знак
Диагностика завершена	помята	Петров	Иван	Яковлевич	+7 987 654 32-10	Audi	A8	1994	60937 km	P003BO37
Диагностика завершена	не работает магнитола	Смирнов	Петр	Дмитриевич	+7 942 532 34 29	Toyota	Corolla	2008	83886 km	A038EK66
Отменён	не едет	Петров	Иван	Саватеевич	+7 984 665 44-00	3A3	968M	1982	125083 km	P353MO96
Диагностика завершена	спустило колесо	Сидоров	Сергей	Николаевич	+7 945 123 44 84	Kia	Sorento	2019	239003 km	O234TA799

Рис. 28 «Окно заказы»

Должности			
Добавить			
Должность	Описание	Зарботная плата	Технический индекс доступа
Администратор	Администратор	100000 руб.	1
Менеджер	Менеджер	80000 руб.	2
Мастер	Мастер, механик может проводить диагностику авто	85000 руб.	4
Кузовщик	Кузовщик	70000 руб.	3
Электрик	Электрик	70000 руб.	3

Рис. 29 «Окно должности»

Новая должность		×
Название	<input type="text"/>	
Зарботная плата	<input type="text"/>	
Технический индекс доступа	<input type="text"/>	
Описание	<input type="text"/>	
		Добавить


Рис. 30 «Окно новая должность»

**Добавить**

Фамилия

Имя

Отчество

Дата рождения   15

Образование

Должность

Логин

Пароль

**Применить**

Рис. 31 «Окно добавить сотрудника»

**Менеджер**

Добавить заказ | Изменить статус на "Завершён" | Изменить статус на "Отменён" | Обновить | Информация о машине | Информация о неисправностях

Фамилия	Имя	Отчество	Статус	Номер телефона	Описание	Машина	Цена ремонта
Петров	Иван	Яковлевич	Диагностика завершена	+7 987 654 32-10	помята	Audi A8 1994	98000
Смирнов	Пётр	Дмитриевич	Диагностика завершена	+7 942 532 34 29	не работает магнитола	Toyota Corolla 2008	11000
Петров	Иван	Саватеевич	Отменён	+7 984 665 44-00	не едет	3A3 968M 1982	98600
Сидоров	Сергей	Николаевич	Диагностика завершена	+7 945 123 44 84	спустило колесо	Kia Sorento 2019	10400

Рис. 32 «Окно менеджера»

Добавить заказ

Клиент

Фамилия

Имя

Отчество

Номер телефона

Машина

Марка

Модель

Год выпуска

Номерной знак

Пробег

Проблема

Создать

Рис. 33 «Окно добавить заказ»

О машине

Марка	Audi
Модель	A8
Год	1994
Пробег	60937 km
Номерной знак	P003BO37

Рис. 34 «Окно о машине»

О неисправности					
Информация о запчасти					
Название	Описание	Запчасть	Стоимость работы	Исполнитель	Статус
Помята крыша (Audi A8)	Замена крыши на машине Audi A8	Крыша (Audi A8)	10000	Малинин Даниил Святославович	В процессе
Помятый капот Audi A8	Замена капота Audi A8	Капот Audi A8	5000	Малинин Даниил Святославович	В процессе
Помято переднее левое крыло Audi A8	Замена переднего левого крыла на Audi A8 (Audi A8 переднее левое крыло)	Переднее левое крыло Audi A8	13000	Зубов Роман Кириллович	В процессе

Рис. 35 «Окно о неисправностях»

О запчасти	
Название	Капот Audi A8
Стоимость	15000
Описание	Капот Audi A8

Рис. 36 «Окно о запчасти»

Заказы - Мастер: Панов Алексей Иванович

Обновить

Диагностика

Изменить статус на "Готово"

Информация о машине

Статус	Описание	Машина	
Диагностика завершена	помята	Audi A8 1994	
Диагностика завершена	не работает магнитола	Toyota Corolla 2008	
Диагностика завершена	спустило колесо	Kia Sorento 2019	

Информация о запчасти

Изменить статус на "Готово"

Название	Описание	Запчасть	Статус
Помята крыша (Audi A8)	Замена крыши на машине Audi A8	Крыша (Audi A8)	В процессе
Помятый капот Audi A8	Замена капота Audi A8	Капот Audi A8	В процессе
Помято переднее левое крыло Audi A8	Замена переднего левого крыла на Audi A8 (Audi A8 переднее левое крыло)	Переднее левое крыло Audi A8	В процессе

Рис. 37 «Окно мастера»

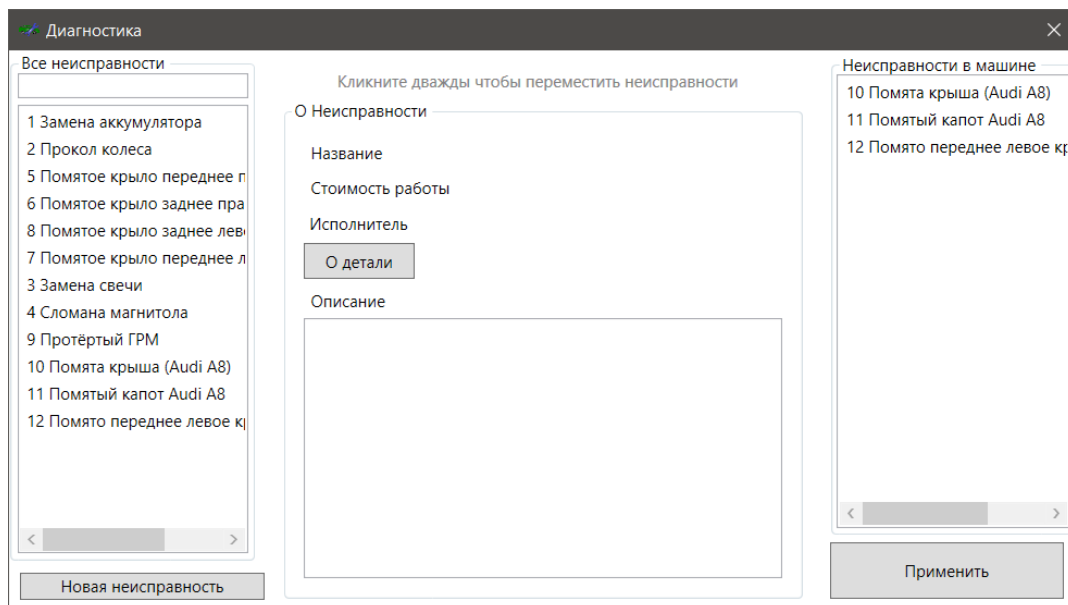


Рис. 38 «Окно диагностики»



Рис. 39 «Окно новая запчасть»



Новая неисправность

Название

Описание

Стоимость работы

Запчасть

Исполнитель

Рис. 40 «Окно новая неисправность»

Заказы - Кузовщик: Малинин Даниил Святославович

Информация о машине

Статус	Описание	Машина
Диагностика завершена	помята	Audi A8 1994

Информация о запчасти

Название	Описание	Запчасть	Статус
Помята крыша (Audi A8)	Замена крыши на машине Audi A8	Крыша (Audi A8)	В процессе
Помятый капот Audi A8	Замена капота Audi A8	Капот Audi A8	В процессе

Рис. 41 «Окно исполнителя»

### 3.3 Описание разработки ПО

Windows Presentation Foundation (WPF) — это технология, разработанная корпорацией Microsoft для построения клиентских приложений Windows с визуально привлекательными пользовательскими интерфейсами.

WPF предоставляет разработчикам широкий набор инструментов для создания приложений с графическим интерфейсом пользователя. Он включает в себя элементы управления, привязку данных, шаблоны, стили, анимацию и другие функции.

Технология WPF является частью .NET Framework и доступна для использования в приложениях на C#, Visual Basic .NET и других языках программирования, поддерживающих .NET. Она широко используется для разработки бизнес-приложений, игр и других типов программного обеспечения.

Основные особенности WPF:

- Графика с аппаратным ускорением;
- Векторная графика и анимация;
- Двухмерная и трёхмерная графика;
- Стилизация элементов управления;
- Использование тем оформления;
- Встроенная поддержка документов;
- Поддержка работы с мультимедиа;
- Интеграция с XAML.

Основные компоненты WPF:

- XAML — язык разметки, который используется для описания пользовательского интерфейса приложения. XAML позволяет разработчикам создавать сложные пользовательские интерфейсы с минимальными усилиями.

- Элементы управления — предоставляют разработчикам возможность создания интерактивных элементов пользовательского интерфейса, таких как кнопки, текстовые поля, списки и т. д. Элементы управления могут быть созданы с помощью XAML или кода.
- Привязка данных — механизм, который позволяет связать данные из источника с элементами пользовательского интерфейса. Это позволяет автоматически обновлять элементы пользовательского интерфейса при изменении данных.
- Шаблоны — позволяют повторно использовать код для создания похожих элементов пользовательского интерфейса. Шаблоны могут быть определены в XAML или коде.
- Стили — определяют внешний вид элементов пользовательского интерфейса. Стили могут быть применены к отдельным элементам или ко всему приложению.
- Анимация — позволяет анимировать элементы пользовательского интерфейса для улучшения взаимодействия с пользователем. Анимация может быть определена в XAML или коде.
- Графика — включает в себя двухмерную и трёхмерную графику, а также анимацию и эффекты. Графика может быть создана с использованием встроенных элементов управления или сторонних библиотек.
- Темы оформления — позволяют изменить внешний вид приложения без изменения его функциональности. Темы оформления могут быть выбраны пользователем или определены разработчиком.
- Поддержка документов — предоставляет возможность работы с документами различных форматов, такими как изображения, текст и таблицы. Документы могут быть отображены в приложении или сохранены в файл.

- Работа с мультимедиа — поддерживает работу с аудио, видео и другими мультимедийными данными. Мультимедиа может быть воспроизведено в приложении или сохранено в файл.

### 3.4 Разработка базы данных

Физическая база данных — это часть базы данных, которая описывает физическую организацию данных на диске или другом носителе информации. Она включает в себя информацию о структуре файлов, индексах, таблицах и других объектах базы данных.

Физическая база данных определяет:

- как данные будут храниться на физическом уровне;
- какие методы доступа будут использоваться для чтения и записи данных;
- каким образом будет обеспечиваться целостность и безопасность данных.

Основные компоненты физической базы данных:

Таблицы. Это основные объекты базы данных, которые содержат данные. Таблицы могут иметь различные структуры и типы данных.

Индексы. Индексы используются для ускорения поиска данных в таблицах. Они могут быть созданы на основе одного или нескольких столбцов таблицы.

Файлы. Файлы представляют собой физические файлы на диске, в которых хранятся данные таблиц и индексов.

Блоки. Блоки — это наименьшие единицы хранения данных в базе данных. Блоки могут содержать несколько записей таблицы или элементов индекса.

Схемы. Схемы определяют структуру базы данных и её объектов. В одной базе данных может быть несколько схем.

Триггеры. Триггеры — это процедуры, которые автоматически выполняются при определённых событиях, таких как вставка, обновление или удаление данных.

Ограничения. Ограничения используются для обеспечения целостности данных. Они могут ограничивать значения столбцов, отношения между таблицами и другие аспекты данных.

Транзакции. Транзакции обеспечивают атомарность, согласованность, изолированность и долговечность (ACID) операций с данными.

Журналирование. Журналирование используется для восстановления базы данных после сбоев. Оно записывает изменения данных в журнале транзакций.

Для создания и управления физической базой данных используются системы управления базами данных (СУБД). СУБД предоставляют инструменты для определения структуры базы данных, создания объектов, выполнения запросов к данным и обеспечения безопасности и целостности данных.

Основные элементы реляционной базы данных:

1. Таблица — основной элемент реляционной базы данных. Каждая таблица имеет уникальное имя и состоит из строк (записей) и столбцов (полей).
2. Строка — запись, содержащая информацию об одном объекте или событии. Каждая строка имеет уникальный первичный ключ.
3. Столбец — поле, содержащее информацию об одном атрибуте объекта или события.
4. Первичный ключ — уникальный идентификатор строки в таблице. Первичный ключ может состоять из одного или нескольких полей.
5. Внешний ключ — поле в таблице, которое ссылается на первичный ключ другой таблицы. Внешний ключ обеспечивает связь между таблицами.

Реляционные базы данных используются для хранения и обработки больших объёмов структурированных данных. Они широко применяются в

различных областях, таких как бизнес, наука, медицина и т. д. Для работы с реляционными базами данных существует множество СУБД.

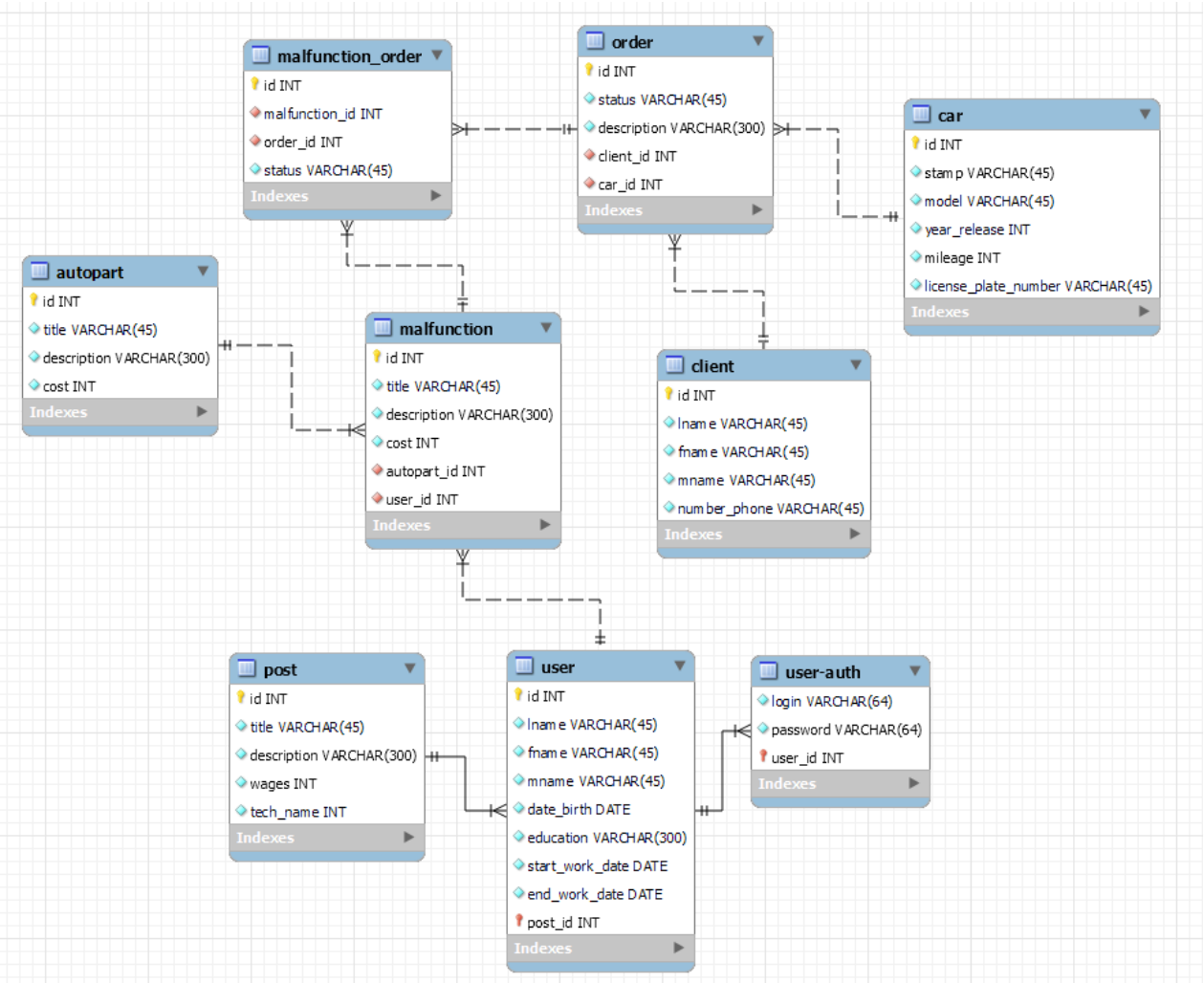


Рис. 42 «Физическая ER-диаграмма»

Имя в БД	Тип данных	Имя
id	int	ID
lname	varchar(45)	Фамилия
fname	varchar(45)	Имя
mname	varchar(45)	Отчество
date_birth	date	Дата рождения
education	varchar(300)	Образование
start_work_date	date	Дата приёма на работу
end_work_date	date	Дата увольнения

post_id	int	Должность ID
---------	-----	--------------

Табл. 1 «Таблица БД: “user”»

Имя в БД	Тип данных	Имя
login	varchar(64)	Логин
password	varchar(64)	Пароль
user_id	int	Пользователь ID

Табл. 2 «Таблица БД: “user-auth”»

Имя в БД	Тип данных	Имя
id	int	ID
status	varchar(45)	Статус
description	varchar(300)	Описание
client_id	int	Клиент ID
car_id	int	Автомобиль ID

Табл. 3 «Таблица БД: “order”»

Имя в БД	Тип данных	Имя
id	int	ID
title	varchar(45)	Название
description	varchar(300)	Описание
cost	int	Стоимость

Табл. 4 «Таблица БД: “autopart”»

Имя в БД	Тип данных	Имя
id	int	ID
lname	varchar(45)	Фамилия
fname	varchar(45)	Имя
mname	varchar(45)	Отчество
number_phone	varchar(45)	Номер телефона

Табл. 5 «Таблица БД: “client”»

Имя в БД	Тип данных	Имя
id	int	ID
title	varchar(45)	Название
description	varchar(300)	Описание
cost	int	Цена
autopart_id	int	Запчасть ID
user_id	int	Пользователь ID

Табл. 6 «Таблица БД: “malfunction”»

Имя в БД	Тип данных	Имя
id	int	ID
malfunction_id	int	Неисправность ID
order_id	int	Заказ ID
status	varchar(45)	Статус

Табл. 7 «Таблица БД: “malfunction\_order”»

Имя в БД	Тип данных	Имя
id	int	ID
title	varchar(45)	Название
description	varchar(300)	Описание
wages	int	Заработная плата
tech_name	int	Технический индекс доступа

Табл. 8 «Таблица БД: “post”»

Имя в БД	Тип данных	Имя
id	int	ID
stamp	varchar(45)	Марка
model	varchar(45)	Модель
year_release	int	Год выпуска
mileage	int	Пробег
license_plate_number	varchar(45)	Номерной знак

Табл. 9 «Таблица БД: “car”»



Примеры заполненных данными таблиц:

id	title	description	cost
1	Аккумулятор	Аккумулятор 12 вольт	5000
2	Колесо	Колесо	10000
3	Крыло (правое, переднее)	Крыло (правое, переднее)	8000
4	Крыло (правое, заднее)	Крыло (правое, переднее)	15000
5	Крыло (левое, заднее)	Крыло (правое, переднее)	15000
6	Крыло (левое, переднее)	Крыло (правое, переднее)	8000
7	Крыша	Крыша	30000
8	Капот	Капот	20000
9	Свеча	Свеча	800
10	Магнитола	Магнитола	10000
11	Ремень ГРМ	Ремень ГРМ	2000
12	Крыша (Audi A8)	Крыша (Audi A8)	30000
13	Капот Audi A8	Капот Audi A8	15000
14	Переднее левое крыло Audi A8	Переднее левое крыло A...	25000

Рис. 43 «Таблица “autopart”»

id	stamp	model	year_release	mileage	license_plate_number
1	Audi	A8	1994	60937	P003BO37
2	Toyota	Corolla	2008	83886	A038EK66
3	3A3	968M	1982	125083	P353MO96
4	Kia	Sorento	2019	239003	O234TA799

Рис. 44 «Таблица “car”»

id	lname	fname	mname	number_phone
1	Петров	Иван	Яковлевич	+7 987 654 32-10
2	Смирнов	Пётр	Дмитриевич	+7 942 532 34 29
3	Петров	Иван	Саватеевич	+7 984 665 44-00
4	Сидоров	Сергей	Николаевич	+7 945 123 44 84

Рис. 45 «Таблица “client”»

login	password	user_id
l1	p1	1
l2	p2	2
l3	p3	3
l4	p4	4
l5	p5	5
12345678	12345678	6
zubov1985	zubov1985	7

Рис. 46 «Таблица “user-auth”»

id	lname	fname	mname	date_birth	education	start_work_date	end_work_date	post_id
1	Иванов	Давид	Вячеславович	1978-01-04	Высшее	2013-05-12	1900-01-01	1
2	Малышев	Юрий	Павлович	1984-03-21	Высшее	2018-08-09	1900-01-01	2
3	Панов	Алексей	Иванович	1992-02-05	Среднее	2020-11-20	1900-01-01	3
4	Малинин	Даниил	Святославович	1987-09-23	Среднее	2020-11-20	1900-01-01	4
5	Тарасов	Михаил	Павлович	1989-06-19	Высшее	2017-03-18	1900-01-01	5
6	Пеньков	Тарас	Петрович	2024-05-29	Среднее	2024-05-29	2024-05-29	1
7	Зубов	Роман	Кириллович	1985-07-01	Высшее	2024-05-30	1900-01-01	3

Рис. 47 «Таблица “user”»

id	title	description	wages	tech_name
1	Администратор	Администратор	100000	1
2	Менеджер	Менеджер	80000	2
3	Мастер	Мастер, механик может проводить...	85000	4
4	Кузовщик	Кузовщик	70000	3
5	Электрик	Электрик	70000	3

Рис. 48 «Таблица “post”»

id	status	description	client_id	car_id
1	Диагностика завершена	помята	1	1
2	Диагностика завершена	не работает магнитола	2	2
3	Отменён	не едет	3	3
4	Диагностика завершена	спустило колесо	4	4

Рис. 49 «Таблица “order”»

id	malfunction_id	order_id	status
14	3	3	В процессе
15	7	3	В процессе
16	8	3	В процессе
17	6	3	В процессе
18	5	3	Готово
19	2	3	В процессе
20	1	3	В процессе
21	4	3	Готово
30	4	2	В процессе
31	2	4	В процессе
32	10	1	В процессе
33	11	1	В процессе
34	12	1	В процессе

Рис. 50 «Таблица “malfunction\_order”»

id	title	description	cost	autopart_id	user_id
1	Замена аккумулятора	Замена аккумулятора	1000	1	5
2	Прокол колеса	Замена колеса	400	2	3
3	Замена свечи	Замена свечи	400	9	3
4	Сломана магнитола	Замена магнитолы	1000	10	5
5	Помятое крыло переднее правое	Замена крыла	2000	3	4
6	Помятое крыло заднее правое	Замена крыла	10000	4	4
7	Помятое крыло переднее левое	Замена крыла	2000	6	4
8	Помятое крыло заднее левое	Замена крыла	10000	5	4
9	Протёртый ГРМ	Замена ремня ГРМ	1000	11	3
10	Помята крыша (Audi A8)	Замена крыши на машине Audi A8	10000	12	4
11	Помятый капот Audi A8	Замена капота Audi A8	5000	13	4
12	Помято переднее левое крыло Audi A8	Замена переднего левого крыла...	13000	14	7

Рис. 51 «Таблица “malfunction ”»

### 3.5 Отладка программы

Процесс отладки проводится в среде разработки Microsoft Visual Studio 2022 с использованием встроенного отладчика. Отладка в Visual Studio 2022 — это процесс поиска и исправления ошибок в коде. Отладка позволяет разработчикам выявлять и исправлять проблемы, которые могут привести к сбоям или неправильному поведению программы.

В Visual Studio есть несколько инструментов для отладки, включая:

Точки останова (Breakpoints) — позволяют остановить выполнение программы в определённой точке кода. Это полезно для проверки состояния программы и переменных в определённый момент времени.

Окно «Локальные» (Locals Window) — показывает значения переменных во время выполнения программы. Это помогает понять, как работает код и какие значения имеют переменные.

Окна «Стек вызовов» (Call Stack) и «Смотреть» (Watch) — показывают стек вызовов функций и значения выражений соответственно. Эти окна помогают понять, как выполняется программа и что происходит с данными.

Инструменты профилирования — позволяют анализировать производительность программы и находить узкие места.

Вот как можно начать отладку в Visual Studio:

Откройте проект в Visual Studio.

Установите точку останова, щёлкнув слева от строки кода.

Запустите программу, нажав F5 или выбрав пункт меню «Отладка → Начать отладку».

Программа остановится на точке останова. Вы можете проверить значения переменных и выполнить код пошагово, используя кнопки «Шаг с заходом» (F11), «Шаг с обходом» (F10) и «Шаг назад» (Shift+F11).

После завершения отладки нажмите кнопку «Остановить отладку» (Shift+F5) или выберите пункт меню «Отладка → Остановить отладку».

Это лишь некоторые из инструментов отладки в Visual Studio. Для более подробного изучения отладки рекомендуется обратиться к документации Visual Studio или пройти онлайн-курсы по отладке.

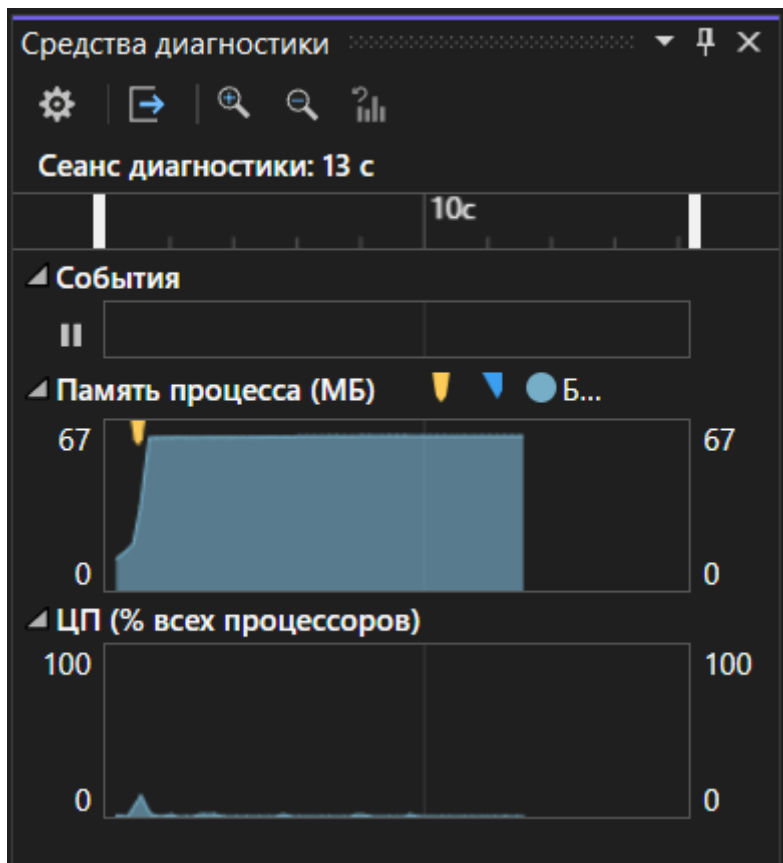


Рис. 52 «Средства диагностики»

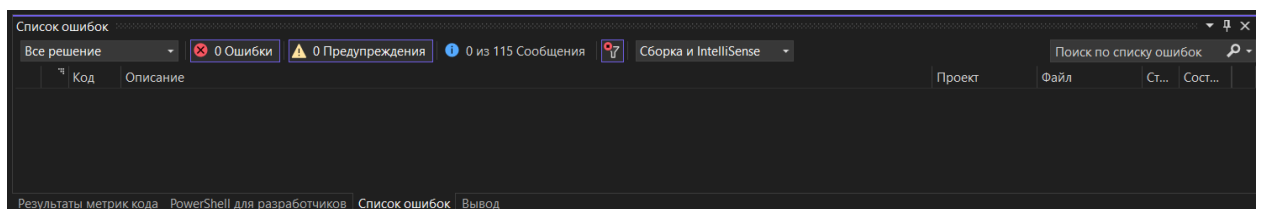


Рис. 53 «Окно ошибок»

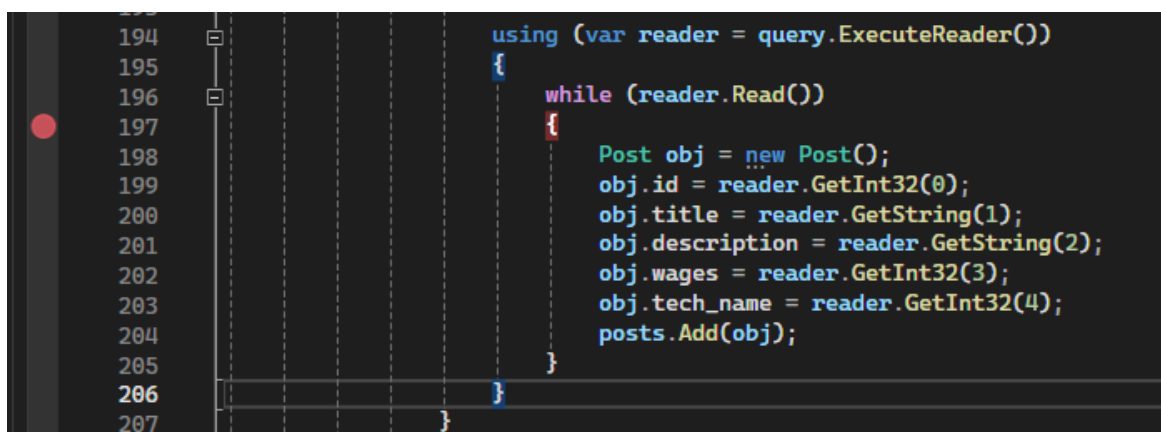


Рис. 54 «Отладка debug»

### 3.6 Руководство пользователя

Добро пожаловать в программу!

Программа предназначена повысить эффективность вашей работы. Её интерфейс максимально прост и интуитивно понятен.

Общий функционал:

Кнопки «Информация о машине», «Информация о запчастях» позволят просмотреть подробную информацию о выбранном автомобиле или запчасти.

Кнопка «Обновить» присутствует у большинства таблиц. Она позволит принудительно обновить данные в таблице. (При взаимодействии с данными таблицы обновляются автоматически)

Для администратора:

Окно администратора позволит переключаться между таблицей заказов, таблицей клиентов, таблицей должностей и таблицей сотрудников. Таблицы заказов и клиентов просты и не имеют возможности редактирования данных.

В окне персонала можно ознакомиться со всеми сотрудниками, в том числе уволенными. Для увольнения, восстановления или изменения данных сотрудника, необходимо выбрать сотрудника из таблицы, и нажать на кнопку необходимого действия. При добавлении нового сотрудника или изменении данных существующего заполните все поля формы. (При несоответствии данных ожидаемым, программа выведет окно с указанием на ошибку заполнения полей)

В окне должностей имеется возможность добавить новую должность сотрудников, для этого заполните все поля в появившемся окне. После этого можно назначить нового сотрудника на новую должность или переквалифицировать старого.

Для менеджера:

Главное окно менеджера содержит в себе таблицу всех заказов.

Кнопки «Изменить статус на «-»» - изменяет статус выбранного в таблице заказа на соответствующий названию кнопки.

Кнопка «Информация о неисправностях» переведет вас к окну в котором указаны неисправности для выбранного вами в таблице заказов автомобиля.

Для добавления нового заказ нажмите кнопку «Добавить заказ», в появившемся окне заполните данные о клиенте, его автомобиле и запишите его жалобы на автотранспорт.

Для рабочего:

В окне рабочего отображаются все доступные ему заказы. По нажатию на заказ в таблице справа появятся неисправности, соответствующие вашей квалификации.

Кнопки «Изменить статус на «Готово»» отвечают за изменение статуса заказа или неисправности, в зависимости от того над какой таблицей она (кнопка) находится. Изменяйте статус после проделанной работы по устранению неисправности.

Для мастера:

В окне мастера отображаются все актуальные заказы. По нажатию на заказ в таблице справа появятся все неисправности.

Кнопки «Изменить статус на «Готово»» отвечают за изменение статуса заказа или неисправности, в зависимости от того над какой таблицей она (кнопка) находится. Изменяйте статус после проделанной работы по устранению неисправности.

Кнопка диагностика позволит указать обнаруженные в ходе диагностики неисправности. Для этого выберите нужное транспортное средство из таблицы слева и нажмите «Диагностика».

Диагностика:

Слева отображается список всех имеющихся в базе неисправностей. Если нужной нет, вы можете её добавить, нажав кнопку «Новая неисправность» и заполнив поля в появившемся окне. Для быстрой навигации среди всех неисправностей над списком всех неисправностей есть текстовое поле – работает как поиск по тексту названия и описания неисправностей.

Для добавления продиагностируемой неисправности к списку неисправностей автомобиля, переместите его в список справа (кликнув дважды по нужной неисправности). Если вы добавили неисправность в список справа случайно, кликните по ней дважды в списке справа для удаления.

Теперь вы готовы к эффективной работе с данной программой, удачи!

При не понятных ситуациях обращайтесь к администратору базы данных.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы была достигнута основная цель – разработан программный модуль для учёта ремонта автомобилей, который может быть использован в работе автосервисов.

Для достижения поставленной цели были решены следующие задачи:

Изучены существующие подходы к учёту ремонта автомобилей и программные решения.

Определены требования к разрабатываемому программному модулю на основе анализа потребностей клиентов.

Спроектирована архитектура программного модуля, база данных и алгоритмы обработки информации.

Разработан и реализован программный модуль для учёта ремонта автомобилей с учётом специфики отрасли и требований клиентов.



Проведено тестирование и оценка эффективности программного модуля на практике, сравнив результаты работы автосервиса до и после внедрения модуля.

Программный модуль представляет собой удобный инструмент для автоматизации учёта ремонта автомобилей. Он позволяет вести учёт всех видов работ, выполняемых в автосервисе, отслеживать историю обслуживания каждого автомобиля, формировать отчёты о выполненных работах и затраченных материалах. Модуль также предоставляет возможность анализировать данные об обслуживании и ремонте автомобилей для выявления тенденций и проблем в работе автосервиса.

Разработанный программный модуль может быть адаптирован под конкретные потребности конкретного автосервиса и интегрирован с другими системами управления предприятием. Это позволит повысить эффективность работы автосервисов и улучшить качество обслуживания клиентов.

Таким образом, можно сделать вывод, что разработанный программный модуль является эффективным инструментом для учёта ремонта автомобилей в автосервисах. Его внедрение позволит автоматизировать процессы учёта технического обслуживания и ремонта транспортных средств, повысить точность и оперативность предоставления информации о состоянии автомобилей, а также оптимизировать работу автосервисов в целом.

Результаты данной работы могут быть использованы в практической деятельности автосервисов, а также в дальнейших исследованиях по автоматизации процессов учёта и управления в сфере технического обслуживания и ремонта автомобиле

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ГОСТ и иные нормативные документы:

1. ГОСТ 7.32-2001 "Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления".
2. ГОСТ Р 7.0.11-2011 "Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования".
3. ГОСТ 7.0.5-2008 "Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила".
4. ГОСТ Р 50779.42–99 (ИСО 8258–91). «Статистические методы. Контрольные карты Шухарта».
5. Федеральный закон от 06.12.2011 № 402-ФЗ (ред. от 28.12.2022) «О бухгалтерском учёте».

Книги и монографии:

1. В. Дунаев. Базы данных. Язык SQL для студента. Учеб. пособие – СПб.: БХВ-Петербург, 2012. – 320 с.
2. О.Л. Голицына, Т.Л. Партыка, И.И. Попов. Основы проектирования баз данных. Учеб. пособие – М.: Форум, 2012. – 416 с.
3. Волгин В. В. «Автосервис. Создание и компьютеризация». — М.: Дашков и К, 2023.
4. Карпов Б. М. «Основы автоматизации производства». — М.: Высшая школа, 2006.

Научные статьи:

1. Афанасьев М. Ю., Багриновский К. А., Матюшок В. М. «Прикладные задачи исследования операций». — М.: ИНФРА-М, 2005.

2. Бураков П. В., Петров В. Ю. «Введение в системы управления проектами». — СПб: Университет ИТМО, 2015.
3. Волков О. И. «Экономика предприятия». — М.: Инфра-М, 1997.

Электронный ресурс:

1. Автоматизированная информационная система // URL: <https://author24referat.ru> (дата обращения: 30.05.2024).
2. Разработка программы // URL: <http://www.tnu.in.ua> (дата обращения: 30.05.2024).
3. Разработка БД для АСУ // URL: <https://studfile.net> (дата обращения: 30.05.2024).
4. Портал «Автомастер» // URL: <https://automastera.net/> (дата обращения: 30.05.2024).
5. Сравнительный анализ готовых решений с информационной системой // URL: <https://vuzlit.com> (дата обращения: 30.05.2024).
6. Работа с Visual Studio // URL: <https://professorweb.ru> (дата обращения: 30.05.2024).

## ПРИЛОЖЕНИЕ