**Содержание**

[ВВЕДЕНИЕ 5](#_Toc164179424)

[1. Аналитическая часть 7](#_Toc164179425)

[1.1. Описание предметной области 7](#_Toc164179426)

[1.2. Анализ существующих разработок 8](#_Toc164179427)

[1.3. Требования, предъявляемые к разрабатываемой ИС 12](#_Toc164179428)

[1.4. Инфологическая модель предметной области 12](#_Toc164179429)

[1.5. Модель предметной области в стандарте IDEF0 15](#_Toc164179430)

[1.6. Диаграмма DFD предметной области 22](#_Toc164179431)

[1.7. Прототипирование интерфейса 29](#_Toc164179433)

[2. Практическая часть 32](#_Toc164179434)

[2.1. Обзор инструментальных средств для разработки ИС 32](#_Toc164179435)

[2.2. Обоснование и выбор программного обеспечения для разработки ИС 38](#_Toc164179436)

[2.3. Описание логики функционирования ПО 40](#_Toc164179437)

[2.4. Реализация интерфейса ИС 47](#_Toc164179438)

[2.5. Написание программного кода 51](#_Toc164179439)

[2.6. Защита информации 51](#_Toc164179440)

[2.7. Тестирование ИС 53](#_Toc164179441)

[2.8. Руководство пользователя ИС 53](#_Toc164179442)

[ЗАКЛЮЧЕНИЕ 54](#_Toc164179443)

[Список использованных источников 55](#_Toc164179444)

[ПРИЛОЖЕНИЯ 56](#_Toc164179445)

[Описание обозначений базы данных 56](#_Toc164179446)

[Фрагменты листинга программы 57](#_Toc164179447)

**ВВЕДЕНИЕ**

Станция технического обслуживания – это специализированное предприятие, занимающееся ремонтом и обслуживанием автомобилей. Несмотря на наличие множества автосервисов в каждом населённом пункте, всё ещё существует значительное количество клиентов, которые предпочитают получать эксклюзивные услуги, специально подходящие их потребностям. Уникальные запросы клиентов включают в себя необходимость подгонки автомобилей под их индивидуальные требования или выполнение мелких ремонтных работ. По оценкам экспертов, спрос на такие услуги превышает предложение на рынке. Именно поэтому открытие автосервиса является прекрасным способом начать собственный бизнес.

Современная промышленность и технические системы требуют высокой степени эффективности и точности в процессах обслуживания и технического ухода. В этом контексте разработка информационных систем для станций технического обслуживания становится необходимостью, направленной на оптимизацию рабочих процессов, улучшение контроля за оборудованием и повышение качества предоставляемых услуг. В данной работе будет рассмотрена разработка информационной системы для станции технического обслуживания, данная информационная система поможет улучшить качество работы предприятия и повысить скорость рассмотрения обращений клиентов, упростит процесс получения информации, связанной с организацией.

Также потребность в информационных системах обусловлена в постоянной автоматизации процессов, которая позволяет в значительной степени повысить уровень обслуживания клиентов, увеличить качество оказываемых услуг, оптимизировать производительность труда сотрудников.

Актуальность дипломного проекта – потребность в автоматизации и упрощения процесса работы с заявками, хранения информации, управления персоналом, формирование и введения учета.

Цель дипломного проекта – разработка информационной системы для станции технического обслуживании с целью провести оптимизацию и автоматизацию для упрощения деятельность ее сотрудников.

Объект работы – автоматизация и оптимизация процессов регистрации клиентов на обслуживание и введения учета и отчетности внутри сервиса.

Предмет исследования – процесс оказания технического обслуживания транспортных средств.

В соответствии с выше поставленной целью, для ее достижения поставлены следующиезадачи:

1. Проектирование системы;
2. Разработка DFD и IDEF0 диаграмм;
3. Произвести анализ уже существующих подобных систем;
4. Изучить средства создания информационной системы и выбрать из них более подходящие;
5. Провести разработку базы данных и информационной системы, осуществить подключение базы данных к информационной системе;
6. Обеспечить защиту информации;
7. Провести тестирование информационной системы.

Проект будет состоять из двух частей, аналитической и практической:

В аналитической части будет рассмотрена структура и порядок работы, проведён анализ схожих информационных системы, приведены диаграммы, спроектированы прототипы информационной системы.

В практической части будут описаны выбор и его обоснование средств создания информационной системы, процесс разработки информационной системы, описание функционала, проведено тестирование информационной системы и будет предоставлен итоговый результат

1. **Аналитическая часть**
   1. **Описание предметной области**

Сервисные технические центры (СТО) — это места, где автовладельцы обращаются, чтобы поддерживать свои машины в отличном состоянии. Здесь работают опытные механики, которые занимаются всем: от замены масла до ремонта двигателя. В СТО можно получить техническое обслуживание, диагностику неисправностей и качественный ремонт. Это помогает автомобилям быть безопасными и надежными на дороге, а их владельцам — спокойно передвигаться, зная, что машина в надежных руках.

Информационные системы для СТО разрабатываются с учетом особенностей этой сферы деятельности. Они объединяют в себе функционал учета и анализа данных о клиентах, автомобилях, запасных частях, проведенных работах, а также планирования и контроля процесса обслуживания. Эти системы помогают автоматизировать и оптимизировать все этапы работы СТО, повышая качество обслуживания, сокращая время на выполнение заказов и управляя ресурсами центра более эффективно.

Основные компоненты информационной системы для СТО включают в себя модуль учета клиентов и автомобилей, который позволяет вести базу данных клиентов и их автомобилей, хранить информацию о техническом состоянии, предыдущих работах и обслуживании. Модуль планирования и расписания работ обеспечивает возможность составления расписания приема автомобилей на обслуживание и ремонт, оптимизирует загрузку мастерской и сокращает время ожидания клиентов.

* 1. **Анализ существующих разработок**

В ходе анализа существующих разработок были найдены следующие информационный системы:

1. Мини-Гараж (https://www.kors-soft.ru/opgar1.htm)

Мини-Гараж представляет собой программу для автосервисов, разработанную компанией «KORSOFT». Она ориентирована на удобство использования и включает в себя широкий спектр функций, необходимых для эффективной работы автомастерской. Интерфейс программы представлен на рисунке 1.1.

Основные возможности программы:

* Учет клиентов и их автомобилей.
* Формирование заказов на услуги.
* Планирование работы и контроль исполнения заказов.
* Интеграция с кассовым оборудованием.
* Возможность ведения электронного документооборота.

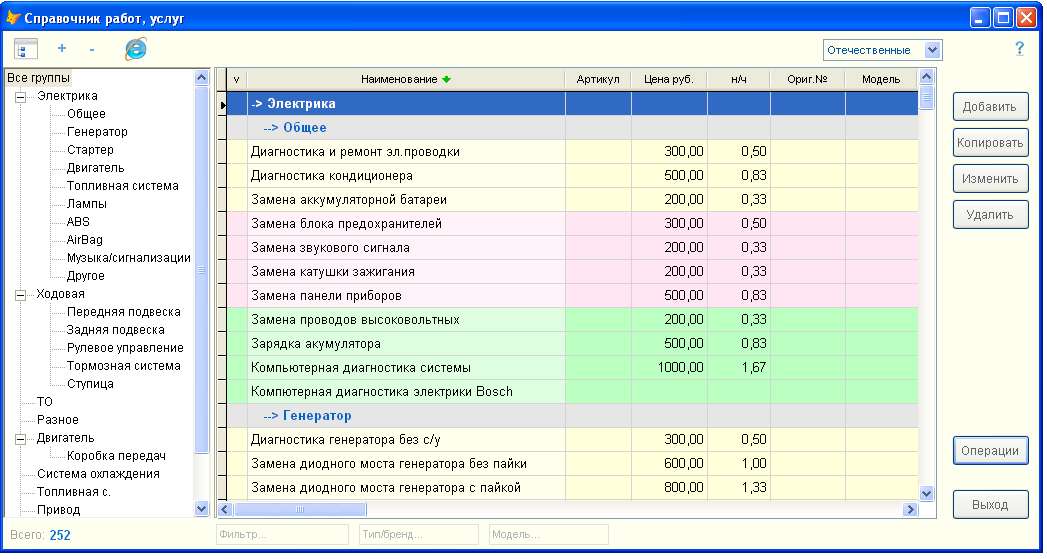


Рисунок 1.1 Интерфейс программы «Мини-Гараж»

1. АСУ-Автосервис Lite (http://surl.li/qzvkw)

АСУ-Автосервис Lite – это программное обеспечение от компании ToolHand, предназначенное для автомастерских и сервисных центров. Оно ориентировано на простоту использования и включает в себя базовый набор функций для учета работ и клиентов. Интерфейс программы представлен на рисунке 1.2.

Основные возможности программы:

* Учет автомобилей и клиентов;
* Удобство работы со списком клиентов;
* Формирование заказов на работы и услуги;
* Возможность редактирование введенных данных по клиентам, заказам и автомобилям;
* Просмотр истории обслуживания;
* Отчетность по выполненным работам и финансовая статистика.

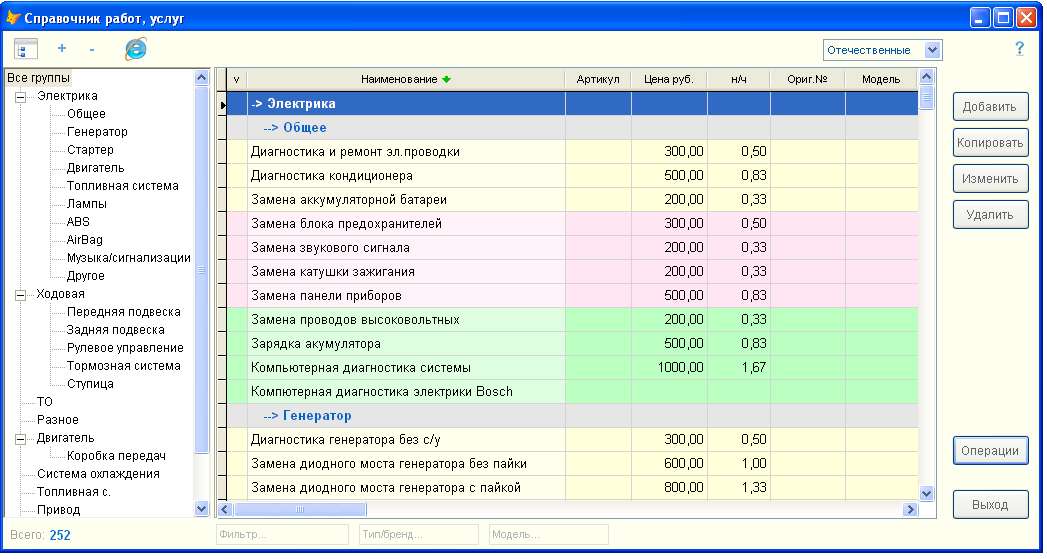


Рисунок 1.2 Интерфейс программы «АСУ-Автосервис Lite»

1. ProsoSoft (https://prostoysoft.ru/AutoService.htm)

ProsoSoft AutoService – это программное обеспечение для автосервисов от компании ProstoySoft. Оно создано для автоматизации процессов управления автосервисом и включает в себя ряд функций для удобства работы персонала и учета клиентов. Интерфейс программы представлен на рисунке 1.3.

Основные возможности программы:

* Учет клиентов и автомобилей.
* Формирование заказов и расписаний работ.
* Планирование ресурсов (мастеров, запчастей и оборудования).
* Отчетность по финансовым показателям и выполненным работам.
* Интеграция с электронными платежными системами.

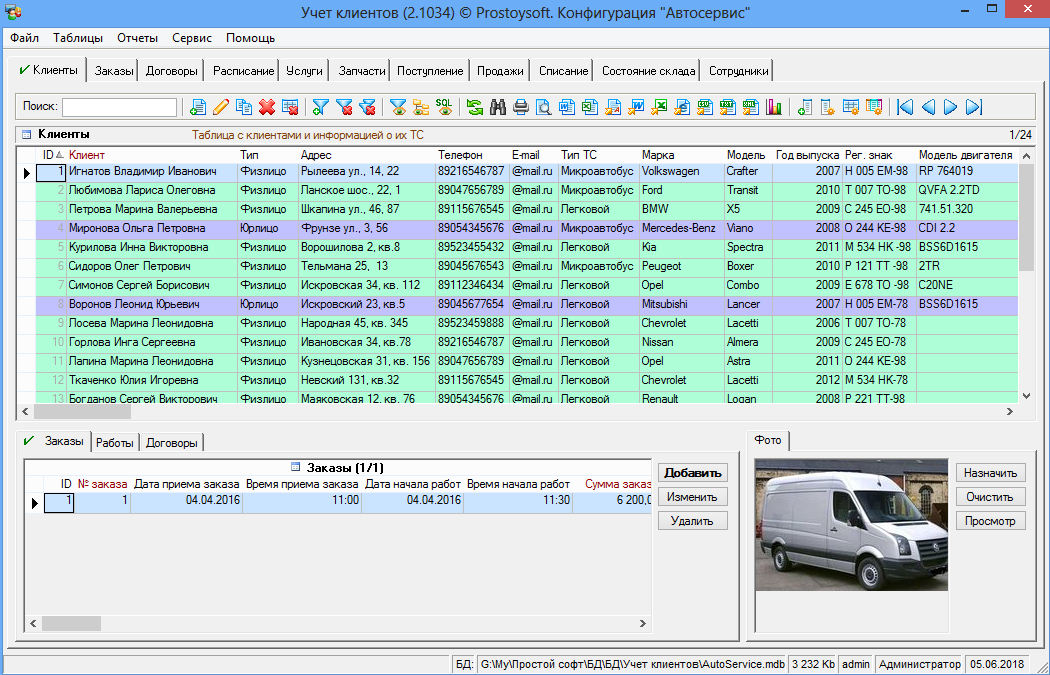


Рисунок 1.3 Интерфейс программы «ProsoSoft»

При сравнении трех программных продуктов для СТО – «Мини-Гараж», «АСУ-Автосервис Lite» и «ProsoSoft» - выявляются ряд важных аспектов, которые могут быть решающими при выборе и разработке подобного приложения.

Во-первых, функциональность играет ключевую роль. «Мини-Гараж» предлагает широкий спектр возможностей для учета заказов, работ с клиентами, и составления отчетности. «АСУ-Автосервис Lite» сосредотачивается на базовых функциях учета заказов и клиентов, но может быть более простым в освоении и использовании. «ProsoSoft» также ориентирован на базовые функции учета заказов, клиентов и автопарка, предлагая понятный интерфейс.

Во-вторых, интерфейс играет важную роль в удобстве использования. «Мини-Гараж» и «ProsoSoft» предлагают сравнительно схожие интерфейсы, с уклоном в стандартный для автосервисов дизайн. «АСУ-Автосервис Lite» может выделиться простым и понятным интерфейсом, что делает его хорошим выбором для пользователей, предпочитающих легкость в использовании.

Третий аспект, технические требования и интеграция. Важно учитывать, какая операционная система требуется для работы программы, а также возможности интеграции с другими системами (например, бухгалтерскими). «Мини-Гараж» и «ProsoSoft» предлагают гибкость в интеграции, в то время как «АСУ-Автосервис Lite» может иметь более ограниченные возможности в этом плане.

Исходя из сравнение данных приложений я сделала вывод, что для реализации информационной системы стоит уделить внимание на следующие аспекты:

* Адаптивность и мобильность интерфейса;
* Безопасность данных и защита конфиденциальной информации клиентов;
* Удобство использования, автоматизация и интеграция с другими сервисами.
  1. **Требования, предъявляемые к разрабатываемой ИС**

Требования к математическому обеспечению

В основе разрабатываемого программного обеспечения должна лежать реляционная модель базы данных. Все таблицы должны находиться в третьей или четвертой нормальной форме. Связи между таблицами должны быть вида:   
«один к одному» или «один ко многим». При проектировании используется метод «сущность – связь».

Требования к информационному обеспечению

Данные в системе должны быть организованны в виде связанных таблиц. Хранение и доступ к данным осуществляется с применением реляционные систем управления базами данных. Информационный обмен между компонентами системы осуществляется по технологии клиент-сервер. Желательно обеспечить возможность резервного копирования и восстановления данных.

Требования к структуре и функционированию информационной системы

Система должна состоять из сервера базы данных и информационной системы. Сервер базы данных отвечает за хранение данных. Программное обеспечение организует работу и обмен данными с сервером базы данных. Программное обеспечение – является основной частью системы, так как на его стороне происходит обработках данных и их получение от сервера базы данных. В последующим информационная система выводит их в понятном для конечного пользователя виде, а также формирует запросы от пользователя к серверу базы данных и последующим вывод информации из базы данных.

Требования к численности и квалификации персонала

Пользователями информационной системы являются сотрудник станции технического обслуживания. Пользователи используют систему через стационарный компьютер, поэтому количество людей, желающих просмотреть или воспользоваться информационной системой ограничено лишь количество устройств в организации.

Обслуживающий персонал:

• Системный администратор – 1 человек;

Пользователи должны иметь минимальные навыки работы за компьютером. Системный администратор должен иметь навыки работы иметь с библиотекой NET Framework 4.8 и выше, знать язык программирования C# на базовом уровне, быть ознакомленным с принципом работы документации XAML, уметь работать в среде SQL Server и знать T-SQL.

Требования к надежности

Программное обеспечение системы должно функционировать и не приводить к зависанию к аварийному, критическим ошибкам системы при любых ошибочных действиях в данной информационной системе.

Требования к эргономике и технической эстетике

Программное обеспечение и технические средства должны иметь эстетически привлекательный вид и быть удобными в использовании, а также иметь интуитивно понятный интерфейс.

Требования к эксплуатации и техническому обслуживанию

Для нормальной работы информационный системы необходимо произвести ряд действий на устройстве:

• Установить лицензионную копию ОС;

• Запустить программу установки информационной систем;

• Установить сервер SQL Server для базы данных;

• Импортировать базу данных.

Требования к защите информации от несанкционированного доступа

Защита информационной системы и информации, с которой она работает, от несанкционированного доступа осуществляется за счёт разграничения доступ по ролям возлагается, что частично возлагается на операционную систему и специализированные программные и программно–аппаратные решения. Доступ к данным предоставляется только легальным пользователям.

Требования по сохранности информации при авариях

Сохранность данных должна быть обеспеченна:

• При сбоях в работе оборудования сети в любой момент времени;

• При сбоях электропитания или программного обеспечения не ведущих к физическому уничтожению или повреждению информации.

Требования к стандартизации и унификации

Информационная система должен предоставлять пользователю привычный, общепринятый интуитивном понятный интерфейс. Программная документация, поставляемая с программой, должна быть оформлена в соответствии со стандартом ЕСПД.

Требования к лингвистическому обеспечению

В качестве языка программирования используются – C#, XML и T-SQL, взаимодействие пользователя и программного обеспечения осуществляется оконный интерфейс информационной системы.

Требования к программному обеспечению

Система должна иметь следующее программное обеспечение: ОС Windows 10 или выше, SQL Server и библиотеку NET Framework 4.8.

Требования к техническому обеспечению

Для работоспособности информационной системы необходимы следующие программные и технические средства:

• клавиатура;

• манипулятор мышь;

• процессор Intel Xeon 1300 и выше;

• оперативная память 4 Гб и больше;

• видеокарта 2 Гб;

• монитор, поддерживающего разрешение 1024х768 точек и больше при глубине цвета 16 бит и более;

• дисковое пространство 10 Гб и больше.

* 1. **Инфологическая модель предметной области**

Инфологическая модель представляет семантику (смысловое содержание предметной области в модели базы данных). Она должна включать в себя емкое описание предметной области, в котором можно было бы оценить корректность разработки проекта.

В инфологической модели содержится информация о сущностях системы (объектах, имеющих важность в предметной области), атрибуты сущностей (свойства) и связями (отношениями) их с другими сущностями.

При разработке инфологической модели она должна соответствовать некоторым требованиям, таким как:

• Легкая расширяемость, которая позволяет ввести новые данные, не изменяя существующие;

• Возможность композиции и декомпозиции модели;

• Легкое восприятие различными классами пользователей (естественный язык при описании модели) и т.п.

Спроектированная ниже модель описывает последующие функции:

• Клиент может существовать без обращения и без ТС, но обращение невозможно зарегистрировать без клиента и ТС;

• Пользователь не может существовать без указания кем именно является этот пользователь, что именно это за работник, но работник может существует без пользователя.

Инфологическая модель для станции технического обслуживания представляет собой описание данных и их взаимосвязей, которые используются в рамках организации для управления ее деятельностью.

Описание инфологической модели:

Клиент оставляет обращение на станции технического обслуживания, работник заносит его обращение в информационную систему, указывает данные клиента, данные его ТС, данные обращения и список услуг, которые потребовал клиент. Обращение включает в себя информацию об объекте недвижимости и контактных данных клиента.

Инфологическая модель представлена на рисунке 1.4.

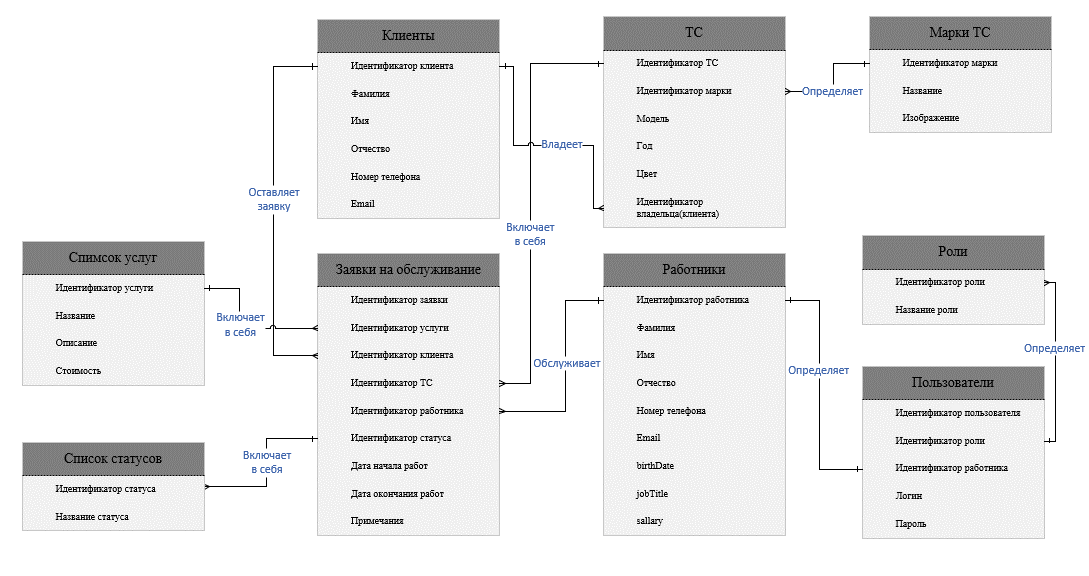


Рисунок .4 Инфологическая модель

* 1. **Модель предметной области в стандарте IDEF0**

Графический язык описания процессов является основой методологии IDEF0. Модель представляет из себя совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Модель IDEF0 предполагает наличие четкой формулировки цели единственного субъекта моделирования и одной точки зрения.

Моделирование IDEF0 – это первый этап изучения системы.

Процесс встроен в общую модель с помощью нескольких типов: вход, выход, контроль, механизм осуществления.

Вход – информация или объект, которые используются для получения выхода. Входом может быть как материальный объект (в данном случае – заявка пользователя).

Выход – результат выполнения функции (обработанная заявки с данными).

Контроль (управление) – нормативные или регламентирующие данные, с помощью которых осуществляется контроль над функцией. Этими данными является документы, которыми руководствуется функция, чтобы получить исправный результат. Контролем могут быть нормативы, стандарты, правила, и прочие руководствующие документы (правила приём и обработки заявок).

Механизм осуществления – ресурсы, средства, которые проводят работу функции для получения результата. Чаще всего механизмом выступает персонал предприятия и технические средства (специалисты).

Стратегии и процедуры, которыми руководствуется процесс (управление) – это правила приема, обработки и выполнения заявок.

Выход – результат выполнения функции (заявки с данными).

Модель для рассматриваемой предметной области реализована в среде MS Visio. Диаграмма представлена на рисунке 1.5.

На модели осуществляется следующее:

1. Обращение клиента вместе с информационей о нем и его трансопртном средстве, передаются работника СТО и заносятся в информационную систему, а с клиентом заключется акт приемки-сдачи.

2. Далее работники осуществляют технический осмотр транспортного средства и только после этого оказывают вабранные клиент улсуги.

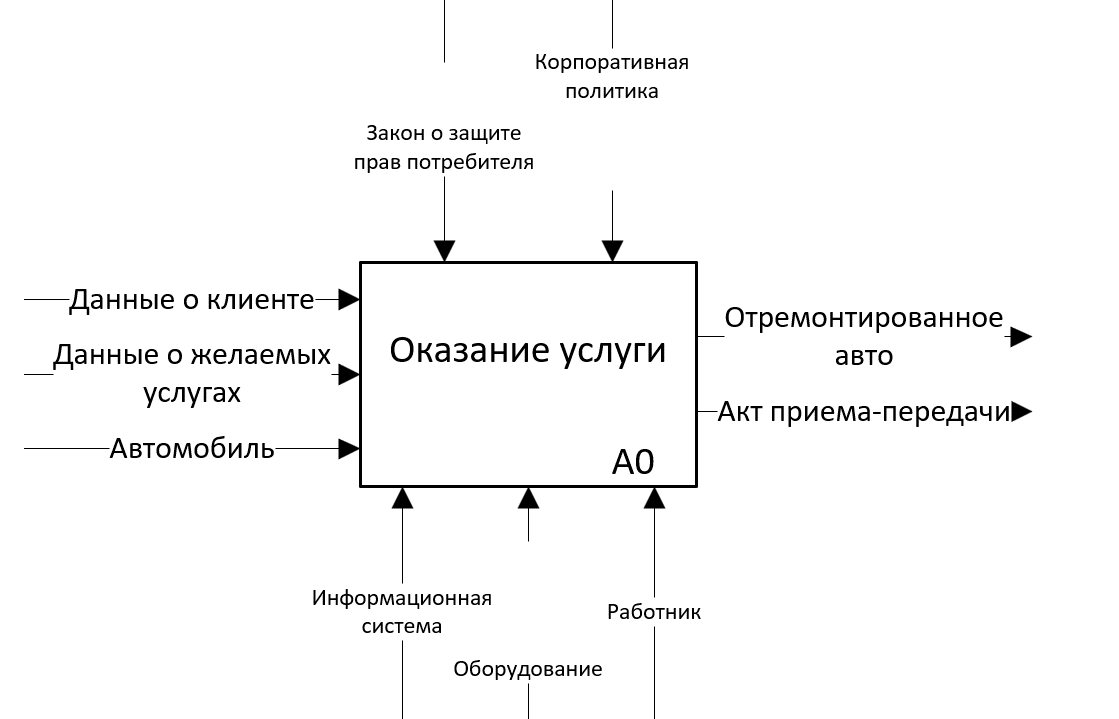


Рисунок 1.5 Контекстная диаграмма в стандарте IDEF0

Таблица 1 – Характеристики объектов первого уровня функциональной модели обработки заявок

|  |  |  |
| --- | --- | --- |
| Объект | Название | Описание |
| Функция А0 | Оказание услуги | Обработка обращения клиента в СТО и последующие обслуживание его траспортного средства |
| Вход А0 | Данные о клиенте | Данные о клиенте, которые идут вместе с его обращением и сохраняются в базе данных |
| Вход А1 | Данные о желаемых услугах | Перечень выбранных услуг клиентом |
| Вход А2 | Автомобиль | Сам автомобиль и информаиця о нем (ПТС) |
| Управление А0 | Закон о защите прав потребителя | Закон Российской Федерации, регулирующий отношения, возникающие между потребителями и изготовителями, исполнителями |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Управление А1 | Корпоративная политика | Правила, рекмондации и обязаности установленные в рамках организации СТО |
| Механизм А0 | Информационная система | Система внутнри которой происходят все операция и является связующим звеном |
| Механизм А1 | Оборудование | Оборудование за счёт которого осуществляется ремонт траспортного средтва |
| Механизм А2 | Работник | Человек осуществляющий работ на СТО и оказание услуг |
| Выход А0 | Отремонтированное авто | Автомобиль клиента после оказание услуг |
| Выход А1 | Акт приема-передачи | Документ, который подтверждает передачу недвижимости от собственника к новому владельцу |

Модель реаизована в среде программы MS Visio. Диаграммы декомпозиции представлена на рсиунке 1.6.

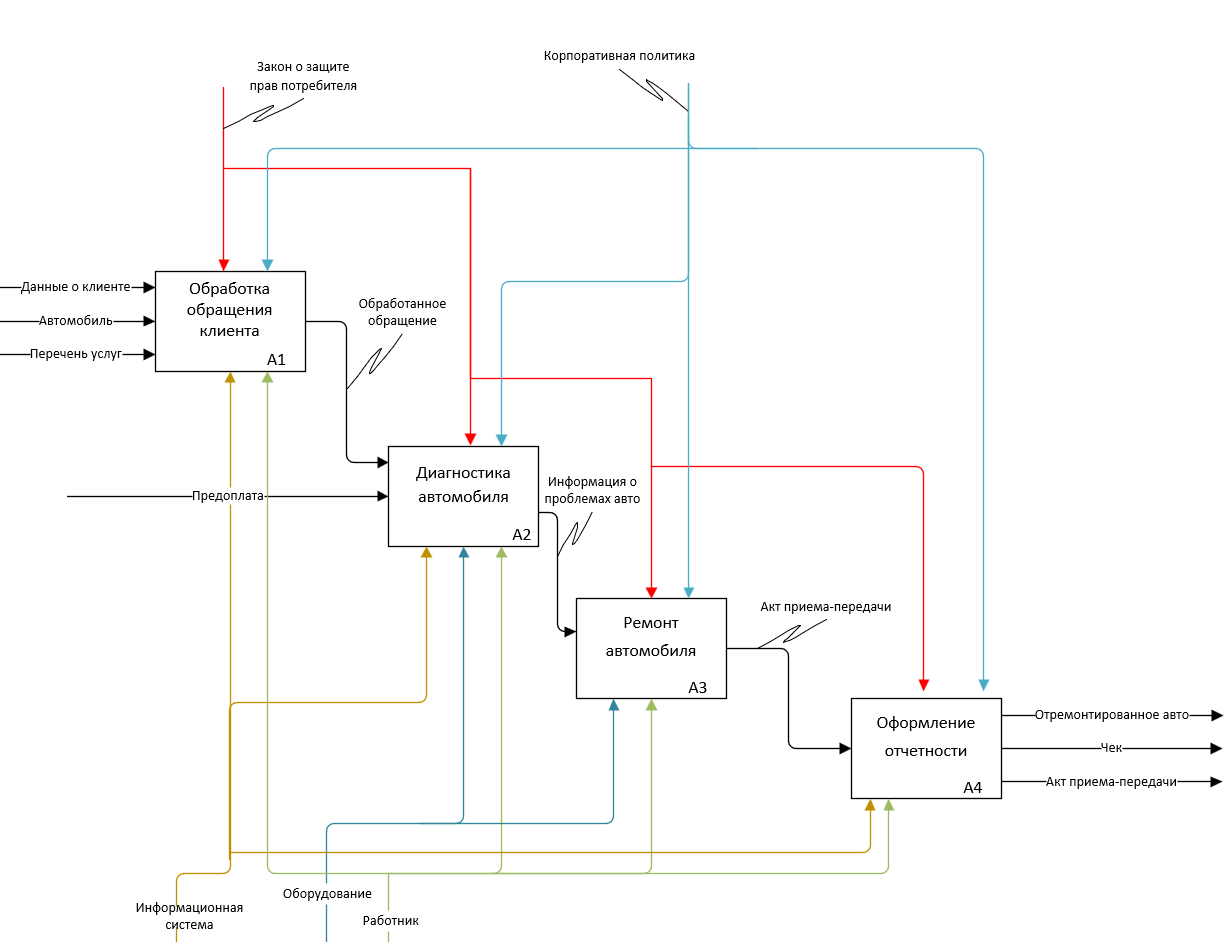


Рисунок 1.6 Диаграмма декомпозиции в стандарте IDEF0

Декомпозиция функциональной модели – это декомпозиция, выполненная в соответствии с функциями, выполняющимися организациями или людьми. Декомпозицию применяют при разборе процесса на функции, которые его составляют. Характеристики обьекто декомпозиции представлены в таблице 1.2.

Таблица 2 – Характеристики объектов второго уровня функциональной модели обработки заявок

|  |  |  |
| --- | --- | --- |
| Объект | Название | Описание |
| Функция А1 | Обрабтка обращения клиента | Обработка обращения клиента в СТО и последующие обслуживание его траспортного средства |
| Функция А2 | Диагностика автомобиля | Диагностика автомобиля, поиск неисправностей и проблем |
| Функция А3 | Ремонт автомобиля | Решение проблем и неисправностей автомобиля |
| Функция А4 | Оформление отчетсности | Оформление докумнетов и отчет о проведенных работах и оказанных услугах |
| Вход А0 | Данные о клиенте | Данные о клиенте, которые идут вместе с его обращением и сохраняются в базе данных |
| Вход А1 | Данные о желаемых услугах | Перечень выбранных услуг клиентом |
| Вход А2 | Автомобиль | Сам автомобиль и информаиця о нем (ПТС) |
| Вход А3 | Предоплата | Оплата услуг клиентом заранее на осблуживание атво, затраты на запчасти и компоненты |
| Управление А0 | Закон о защите прав потребителя | Закон Российской Федерации, регулирующий отношения, возникающие между потребителями и изготовителями, исполнителями |
| Управление А1 | Корпоративная политика | Правила, рекмондации и обязаности установленные в рамках организации СТО |
| Механизм А0 | Информационная система | Система внутнри которой происходят все операция и является связующим звеном |
| Механизм А1 | Оборудование | Оборудование за счёт которого осуществляется ремонт траспортного средтва |
| Механизм А2 | Работник | Человек осуществляющий работ на СТО и оказание услуг |
| Выход А0 | Отремонтированное авто | Автомобиль клиента после оказание услуг |
| Выход А1 | Чек | Подтверждение оказанных услуг |
| Выход А2 | Акт приема-передачи | Документ, который подтверждает передачу недвижимости от собственника к новому владельцу |

* 1. **Диаграмма DFD предметной области**

Диаграмма потоков данных DFD – это нотация, которая является основным средством моделирования функциональных требований проектируемой системы. Для моделирования информационный систем с точки зрения хранения, обработки и передачи данных.

В этой нотации отображены данные, напрямую связанные с обработкой обращения клиента. Клиент передает данные о себе и своем транспортом средстве и желаемых услугах, в последующем эти данные заносятся в информационную системы данные, после этого они передаются работника для дальнейшего анализа, осмотра автомобиля и оказания услуг. Контекстная диаграмма потоков данных (DFD) изображена на рисунке 1.7.

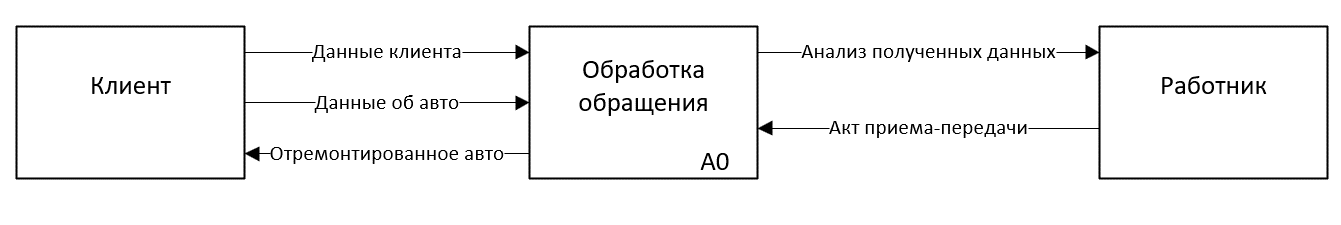


Рисунок 1.7 Декомпозиция DFD 1-го уровня.

На рисунке 1.8 представлена диаграмма декомпозиции в нотации DFD, описывающая деятельность по оформлению авиабилета.

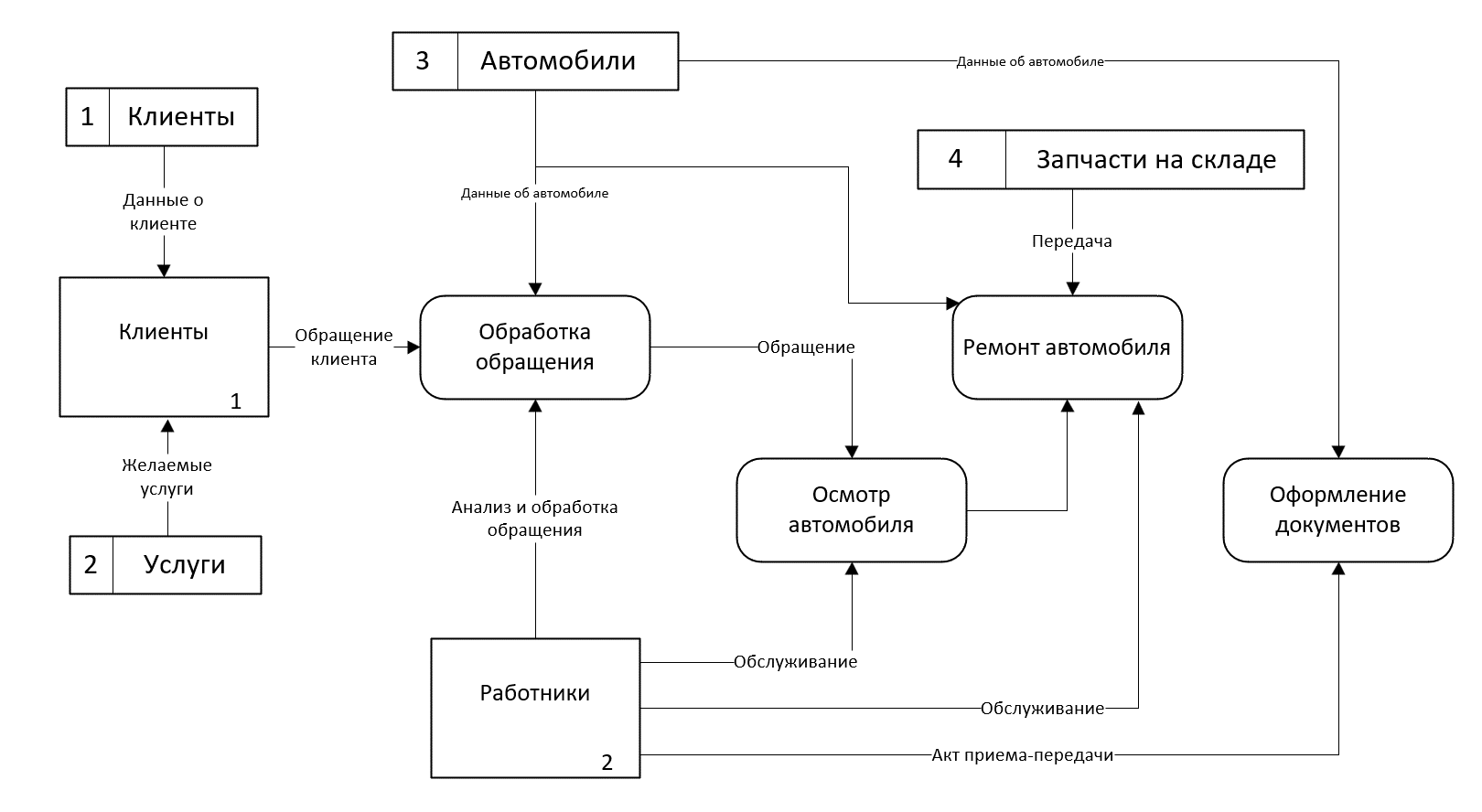


Рисунок 1.8 Декомпозиция DFD 2-го уровня.

* 1. **Прототипирование интерфейса**

Прототипирование является важным этапом в разработке продукта, так как позволяет проверить идеи и концепции до того, как будет написан код. Это экономит время и ресурсы, так как можно быстро определить, что работает и что не работает, и внести необходимые изменения.

Прототипирование экономит денежные затраты на разработку того, что может не сработать. И даёт возможность посмотреть на «рабочее приложение» до того, как программисты начнут писать код. Качество прототипирования влияет удобство использование и на качество связанного с разметкой кода. Высококачественная разработка прототипа — это хорошие вложения в программную часть.

Модуль информационной системы будет состоять из страницы авторизации, с которой пользователь будет переходить на главную страницу, на которой представлены основные разделы на который пользователь сможет перейти, это раздел с «заявки на обслуживание», с информацией о клиента – «наши клиенты» и «обслуживаемые машины».

Страница авторизации, представлена на рисунке 1.9. Содержит форму для авторизации.

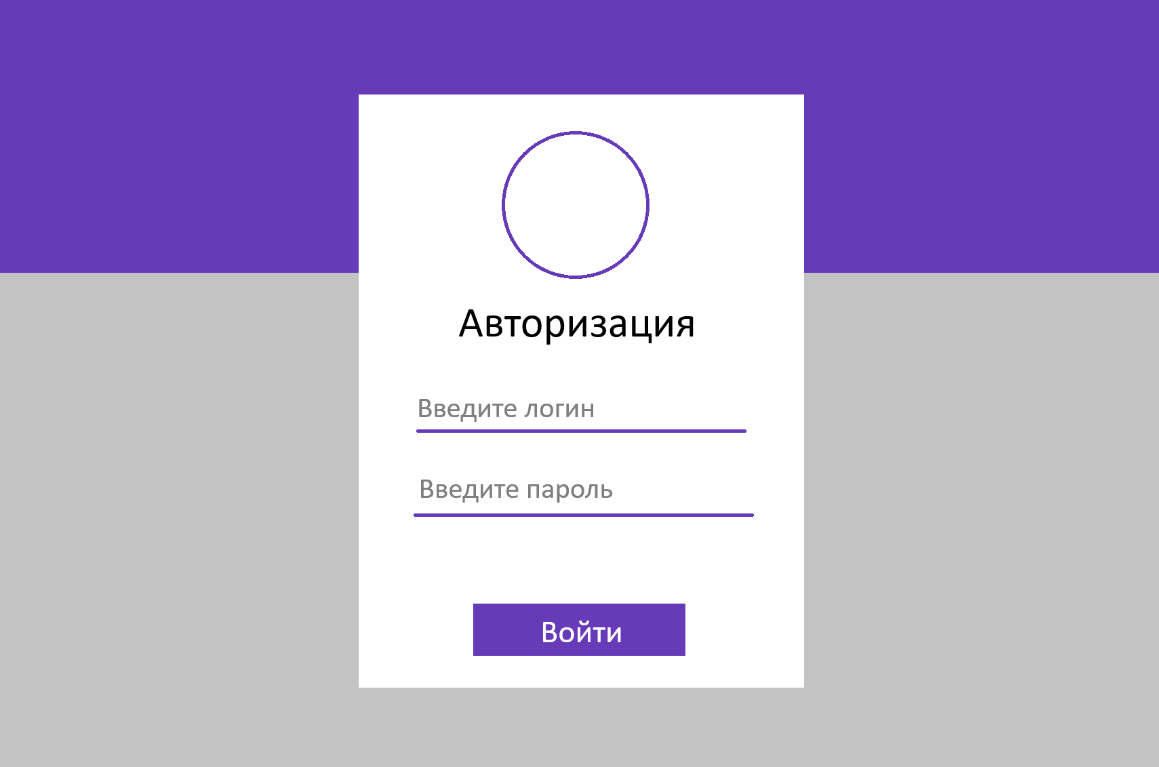


Рисунок 1.9 – «Прототип страницы авторизации»

Главная страница, представлена на рисунке 1.10. Содержит шапку, в которой располагается логотип организации, ее наименования и вспомогательная кнопка, а также в качестве содержимого страницы выступают кнопки для перехода по разделам.



Рисунок 1.10 – «Прототип главной страницы»

Страница с выводом информации о разделе, представлена на рисунке 1.11. Содержит информацию о выбранном разделе и кнопки для взаимодействия с информацией.

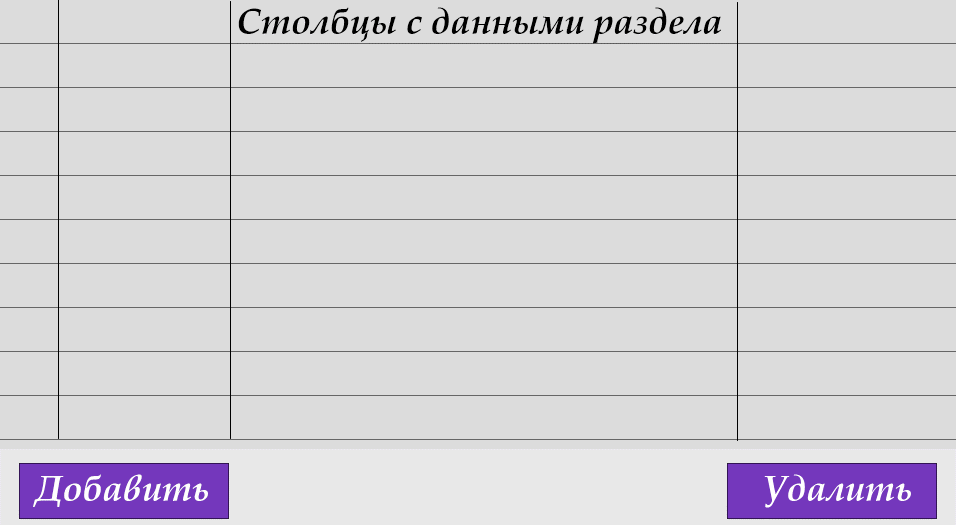


Рисунок 1.11 – «Прототип страницы с информацией о разделе»

Страница для редактирования и добавления записей содержит форму для записи. Страница представлена на рисунке 1.12.

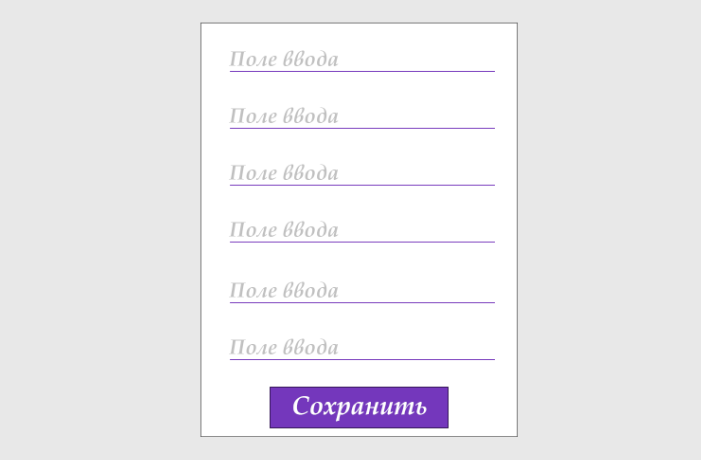


Рисунок 1.12 – «Прототип страницы для редактирования и добавления записей»

1. **Практическая часть**
   1. **Обзор инструментальных средств для разработки ИС**

В настоящее для создания приложений используется множество средств программирования. Это языки программирования, различные среды разработки, используются шаблоны и инструменты для базы данных и многое другое.

Наибольшей популярностью при написании приложений пользуются такие языки программирования, как:

* C# – объектно-ориентированный язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.
* C++ – компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.
* Java – кроссплатформенный, строго типизированный объектно-ориентированный язык программирования общего назначения. Зачастую используется для написания серверного программного обеспечения, драйверов, мобильных и мультимедийных приложений. Программы на Java транслируются в байт-код, который затем выполняется виртуальной машиной Java (JVM). JVM — это программа, которая обрабатывает байтовый код и передает инструкции оборудованию как интерпретатор. Достоинством подобной реализации является независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует JVM.
* Python – высокоуровневый язык программирования общего назначения, который используется в том числе и для разработки веб-приложений. Язык ориентирован на повышение производительности разработчика и читаемости кода. Поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. В языке присутствует динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Программный код на Python организовывается в функции и классы, которые могут объединяться в модули, а они в свою очередь могут быть объединены в пакеты.
* Delphi – императивный, структурированный, объектно-ориентированный, высокоуровневый язык программирования со строгой статической типизацией переменных. Основная область использования — написание прикладного программного обеспечения. Этот язык программирования является диалектом языка Object Pascal.

Управление базами данных:

* Oracle – система, которая функционирует на подавляющем большинстве платформ, даже таких как масштабные сервера и большие ЭВМ. Oracle поддерживает все возможные архитектуры. Эта системы будет полезна на больших предприятиях. Для чтения и записи данных требуется использовать Oracle NoSQL Database с Java/C API, что не является удобным для дальнейшего ее использования при разработке небольшой web-системы. Полностью поддерживает разработку схематичных приложений с помощью модели данных JSON для повышения производительности разработчиков.
* MySQL – СУБД, позволяющая упростить работу с реляционными базами данных. Одним из преимуществ является то, что это свободное программное обеспечение. Это одна из самых популярных СУБД, которой может воспользоваться любой человек, которого она заинтересует. Главной особенность MySQL считается её гибкость, которая обеспечивается поддержкой большого количества типов таблиц. Она включает в себя огромный набор инструментов, который будет полезен при разработке любого проекта. Зачастую является решением для малых и средних приложений.
* Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Также является одной из популярной СУБД. Эта СУБД поддерживает работу с другими программными продуктами от Microsoft: MS Excel, MS Access. Использует язык запросов – Transact-SQL, создан совместно Microsoft и Sybase, который является реализацией стандарта ANSI/ISO по структурированному языку запросов с расширениями. Имеет множество утилит и компонент для работы, например таких как SQL Server Management Studio (SSMS).
* MongoDB – СУБД, которая написана на C++. Она используется для хранения той информации, которую невозможно нормально структурировать в обычных реляционных базах данных. Это документно-ориентированная база данных. Может быть удобной при наличии на предприятии крупного документооборота. MongoDB кроссплатформенна, что позволяет портировать ее на различные платформы. Имеется возможность и загрузить исходный код, самостоятельно скомпилировав его, однако, потребуется использовать библиотеки с официального сайта.
* PostgreSQL – Масштабируемая объектно-реляционная база данных, работающая на Linux, Windows, OSX и некоторых других системах. В PostgreSQL 10 есть такие функции, как логическая репликация, декларативное разбиение таблиц, улучшенные параллельные запросы, более безопасная аутентификация по паролю на основе SCRAM-SHA-256 с поддержкой табличных пространств, а также хранимых процедур, объединений, представлений и триггеров.

Среды разработки:

* Visual Studio – представляет собой полностью интегрированную среду разработки. Она спроектирована таким образом, чтобы делать процесс написания кода, его отладки и компиляции в сборку для поставки конечным потребителям как можно более простым. Является наиболее функционально насыщенной IDE-средой, в которой можно разрабатывать приложения на C#. Данный продукт поставляется с конструкторами графических пользовательских интерфейсов, с возможностями работы с базами данных, с поддержкой фрагментов кода, с возможностями для просмотра всего проекта в целом, с просмотром свойств объектов.
* Standard Widget Toolkit – библиотека графических пользовательских элементов с открытым исходным кодом для разработки интерфейсов пользователя на языке Java. Предоставляет из себя переносимый API для разработчиков Java. Использует собственные библиотеки операционной системы через собственный интерфейс Java (Java Native Interface) для визуализации элементов графического интерфейса, аналогично программам, написанным с использованием собственного API операционной системы.
* PyCharm – это интегрированная среда разработки на языке Python, которая была разработана международной компанией JetBrains (да, и снова эти ребята). Эта IDE распространяется под несколькими лицензиями, в том числе как Community Edition, где чуть урезан функционал. Сами разработчики характеризуют свой продукт как «самую интеллектуальную Python IDE с полным набором средств для эффективной разработки на языке Python», но она не ограничивается на Python. PyCharm поддерживает Google App Engine; IronPython, Jython, Cython, PyPy wxPython, PyQt, PyGTK и др.
* Qt Creator – кроссплатформенная свободная IDE для разработки на С, С++ и QML с помощью фреймворка Qt. Разработана Trolltech для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML. Зачастую используется при написания программного обеспечения для роботов, автомобилей и кораблей, которые используются на производствах, в перевозке грузов и других похожих отраслях. Qt Creator позволяет добиться высокой производительности при прототипировании интерфейса в приложениях любой сложности.
* CLion – продукт JetBrains. Многофункциональная IDE. В которой поддерживается разработка кода на C и C++, а также web-разработка на HTML(5), CSS, JavaScript, XML. А также на некоторых других языках доступен в виде плагинов (например, Lua). Основным преимуществом является статический и динамический анализ с безопасный рефакторингом кода. Позволяет кастомизировать редактор так, как удобно пользователю.
* IntelliJ IDEA – ещё один продукт от JetBrains. Комплекс программных средств, который используется для написания, исполнения, отладки и оптимизации кода для многих языков программирования, в частности Java, JavaScript, Python и других языков программирования от компании JetBrains. Отличается обширным набором инструментов для рефакторинга (перепроектирования) и оптимизации кода. Интеграция с серверами приложений, включая Tomcat, TomEE, GlassFish и другие. Имеет набор инструментов для работы с базами данных, SQL файлами и инструменты для запуска тестов и анализа покрытия кода, включая поддержку всех популярных фреймворков для тестирования.
* Eclipse – то бесплатная опенсорсная среда разработки, которая хорошо подойдет как новичкам, так и опытным разработчикам. Помимо инструментов отладки и поддержки Git/CVS, Eclipse поставляется с Java и инструментом для создания плагинов. Изначально Eclipse использовалась только для Java, но сейчас, благодаря плагинам и расширениям, ее функции значительно расширились. Именно из-за возможности расширить Eclipse своими модулями эта платформа и завоевала свою популярность среди разработчиков. Функционал Eclipse не такой большой, как у IntelliJ IDEA, зато эта среда разработки распространяется с открытым исходным кодом. Основными преимуществами является возможность программировать сразу на нескольких языках и возможность интеграции JUnit.
  1. **Обоснование и выбор программного обеспечения для разработки ИС**

Для разработки программного обеспечения мной было принято решение реализовывать информационную системы с нуля в среде Visual Studio на платформе Windows Presentation Foundation (WPF) используя язык C#. Ниже описано каждое средство реализации, вместе с их преимуществами среди других.

В качестве среды разработки была выбрана «Visual Studio», потому что:

* Поддержка контроля версий GIT;
* Технология автодополнения IntelliSense, которая умеет исправлять ошибки разработчика, предлагая ряд вариантов, но, в отдельных случаях, может и сгенерировать фрагменты кода;
* Расширенный функционал, который можно дополнятся плагинами;
* Интегрированная среда разработки (IDE);
* Удобный и интуитивно понятный интерфейс, который можно кастомизировать под каждого пользователя индивидуально;
* Присутствует встроенный отладчик;
* Наличие встроенного средства для быстрого рефакторинга;
* Множество функций, которые позволяют изменять редактор в соответствии с использованием, что означает, что пользователь может загружать библиотеки из Интернета и интегрировать его с кодом в соответствии со своими требованиями.
* Автоматизированный процесс публикации и автоматизации сборки проекта и приложения;
* Наличие множество утилит, который упрощают работу с сторонними компонентами и приложениями, например таким как SQL Server.

В качестве СУБД был выбран «SQL Server», потому что:

* Обеспечивает практически неограниченный рост объемов хранения за счет увеличения надежности и масштабируемости системы, используя все преимущества мультипроцессорной обработки файлов;
* Возможность взаимодействия с публичным облаком Windows Azure;
* Облачное резервирование локальных систем и аварийное восстановление из облака повышает отказоустойчивость;
* Возможность интеграция структурированных и неструктурированных данных;
* Наличие интегрированная среды для управления любой инфраструктурой SQL.

В качестве языка программирования была выбран «C#», потому что:

* Объектно-ориентированный язык программирования;
* Работает на базе платформы .NET Framework за счёт чего платформа поддерживает несколько языков: наряду с C# это VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET;
* Наличие большого количества библиотек и шаблонов, которые позволяют не тратить время на изобретение “велосипеда”;
* Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык.
  1. **Описание логики функционирования ПО**

При разработке информационной системы станции технического обслуживания была создала база данных «koleso», в которой хранится вся необходимая информация для корректной работы информационной системы. Это информация о заявка на обслуживание, клиентах, о списке предоставляемых видов работы и услуг, о сотрудниках и их учетных данных, и о самих автомобилях. Также спроектирована   
схема базы данных, которая представлена на рисунке 2.1.

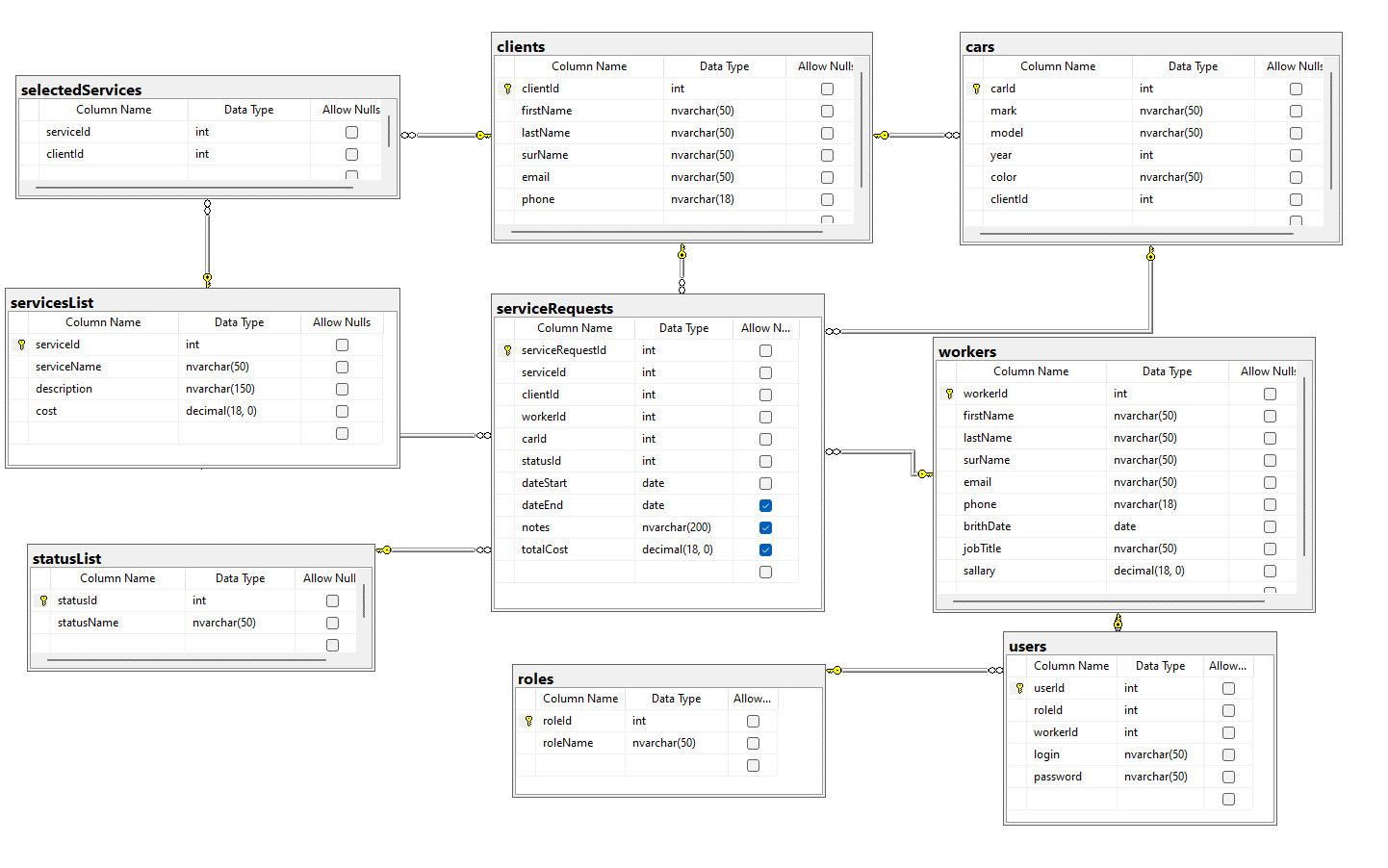


Рисунок 2.1 – «Схема базы данных»

В базе данных имеются следующие таблицы:

Таблица «serviceRequests» - является основной и содержит информацию о всех имеющихся обращениях клиентов, данные таблицы представлены на рисунке 2.2.

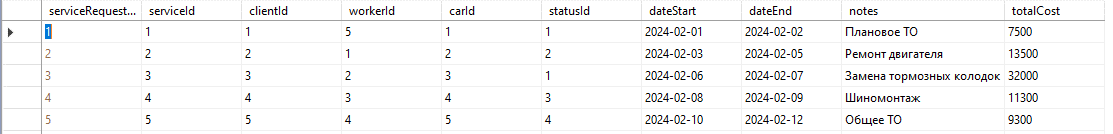


Рисунок 2.2 – Данные таблицы «serviceRequests»

Таблица 3 – Описание структуры «serviceRequests»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| serviceRequestId | int | Идентификатор обращения |
| serviceId | int | Идентификатор услуги |
| clientId | int | Идентификатор клиента |
| workerId | int | Идентификатор сотрудника |
| carId | int | Идентификатор машины |
| statusId | int | Идентификатор статуса заявки |
| dateStart | date | Дата начала работ |
| dateEnd | date | Дата окончания работ |
| notes | nvarchar(200) | Примечания |
| totalCost | decimal(18,0) | Общая стоимость работ |

Таблица «serviceList» - содержит информацию о предоставляемых услугах, данные таблицы представлены на рисунке 2.3.

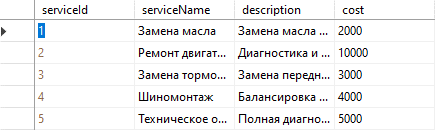


Рисунок 2.3 – Данные таблицы «serviceList»

Таблица 4 – Описание структуры «serviceList»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| serviced | int | Идентификатор услуги |
| serviceName | nvarchar(50) | Название |
| description | nvarchar(150) | Описание |
| cost | decimal(18,0) | Стоимость |

Таблица «statusList» - содержит информацию о статусе заявок, данные таблицы представлены на рисунке 2.4.

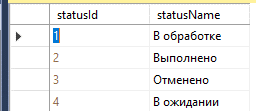


Рисунок 2.4 – Данные таблицы «statusList»

Таблица 5 – Описание структуры «statusList»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| statusId | int | Идентификатор статуса |
| statusName | nvarchar(50) | Название |

Таблица «cars» - содержит информацию о машинах клиентах, которые попадают в сервис, данные таблицы представлены на рисунке 2.5.

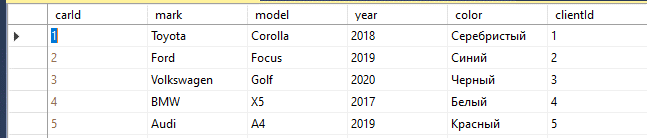


Рисунок 2.5 – Данные таблицы «cars»

Таблица 6 – Описание структуры «cars»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| carId | int | Идентификатор машины |
| mark | nvarchar(50) | Марка |
| model | nvarchar(50) | Модель |
| year | int | Год |
| color | nvarchar(50) | Цвет |
| clientId | int | Идентификатор клиента |

Таблица «clients» - содержит информацию о клиентах, которые воспользовались услугами, данные таблицы представлены на рисунке 2.6.

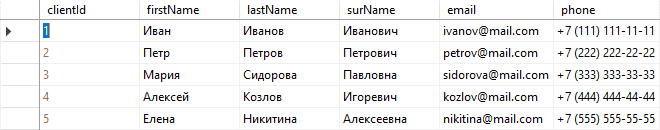


Рисунок 2.6 – Данные таблицы «clients»

Таблица 7 – Описание структуры «clients».

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| clientId | int | Идентификатор клиента |
| firstName | nvarchar(50) | Фамилия |
| lastName | nvarchar(50) | Имя |
| surName | nvarchar(50) | Отчество |
| email | nvarchar(50) | Почта |
| phone | nvarchar(50) | Номер телефона |

Таблица «workers» - содержит информацию о сотрудниках, работающих в организации, данные таблицы представлены на рисунке 2.7.

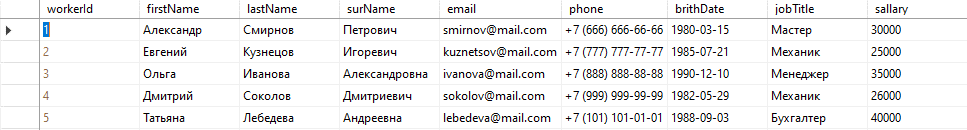


Рисунок 2.7 – Данные таблицы «workers»

Таблица 8 – Описание структуры «workers»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| workerId | int | Идентификатор сотрудника |
| firstName | nvarchar(50) | Фамилия |
| lastName | nvarchar(50) | Имя |
| surName | nvarchar(50) | Отчество |
| email | nvarchar(50) | Почта |
| phone | nvarchar(50) | Номер телефона |
| birthDate | date | День рождения |
| jobTitle | nvarchar(50) | Должность |
| salary | decimal(18,0) | Зарплата |

Таблица «users» - содержит информацию о пользователях информационной системы, данные таблицы представлены на рисунке 2.8.

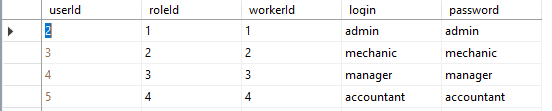


Рисунок 2.8 – Данные таблицы «users»

Таблица 9 – Описание структуры «users»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| userId | int | Идентификатор пользователя |
| roleId | int | Идентификатор роли |
| workerId | int | Идентификатор сотрудника |
| login | nvarchar(50) | Логин |
| password | nvarchar(50) | Пароль |

Таблица «roles» - содержит информацию о ролях для пользователей, данные таблицы представлены на рисунке 2.9.



Рисунок 2.9. – Данные таблицы «roles»

Таблица 10 – Описание структуры «roles»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| roledId | int | Идентификатор роли |
| roleName | nvarchar(50) | Название роли |

* 1. **Реализация интерфейса ИС**

В ходе разработки информационной системы был разработан логотип. Логотип был реализован в программе Adobe Photoshop CC. Логотип представлен на рисунке 2.8.



Рисунок 2.8. – Логотип информационной системы

Была выбрана цветовая палитра для информационной системы. Палитра представлена в таблице 11.

Таблица 11 – Цветовая палитра модуля информационной системы

|  |  |  |
| --- | --- | --- |
| Основной цвет | Дополнительный цвет | Акцентирования внимания |
| #FFF (255, 255, 255) | # DADADA (218,218,218) | #673AB7 (103,58,183) |
|  |  |  |

При запуске модуля информационной системы открывается страница авторизации на странице имеется форма для указания данных для входа, страница представлена на рисунке 2.9.

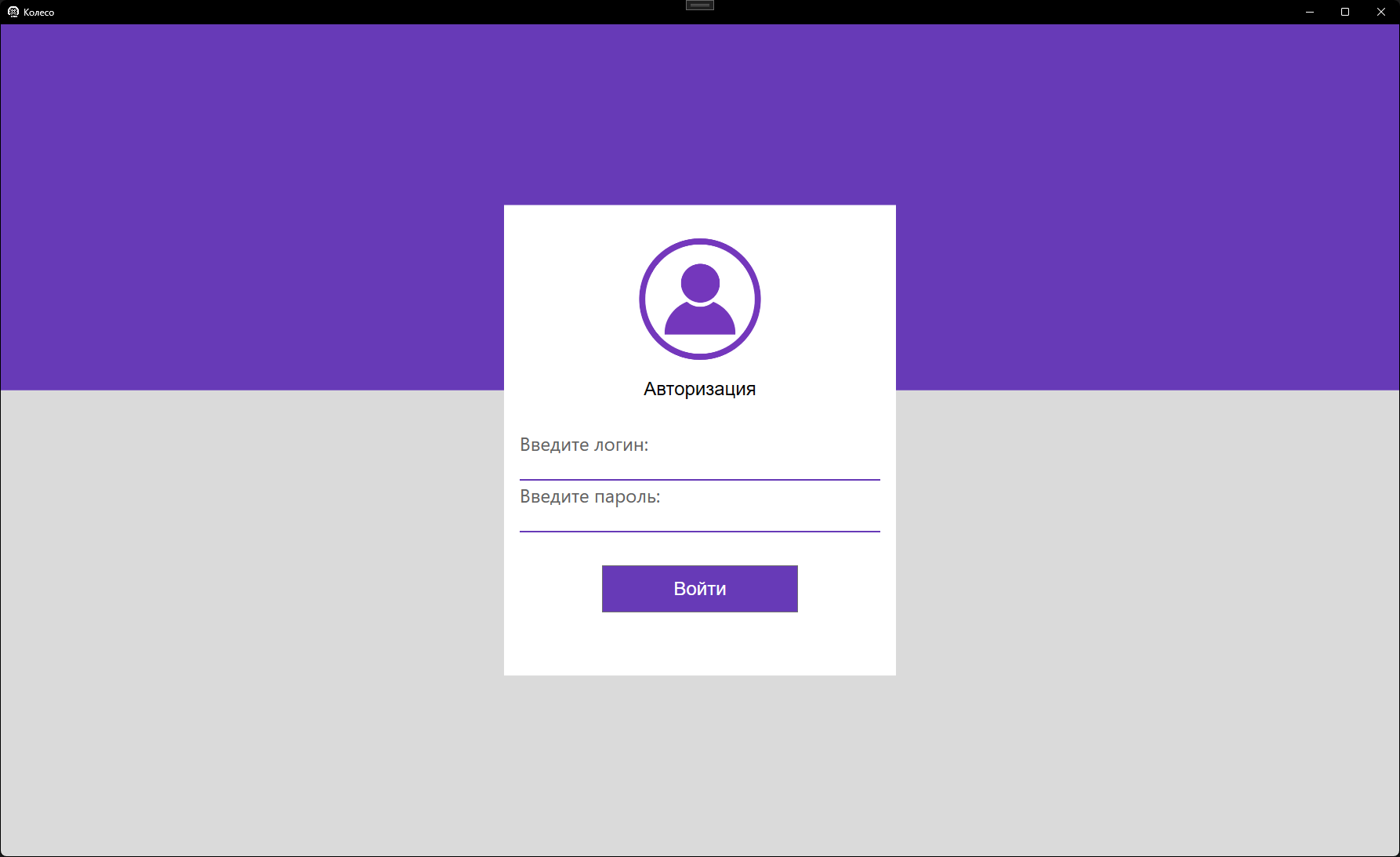


Рисунок 2.9 – Страница авторизации

После авторизации нас встречает главная страницу, на которой расположены основные разделы. Страница представлена на рисунке 2.10.



Рисунок 2.10 – Главная страница

Страница «Заявки на обслуживание» содержит информацию о всех заявках организации и кнопки для взаимодействия с этими записями. Страница представлена на рисунке 2.11.

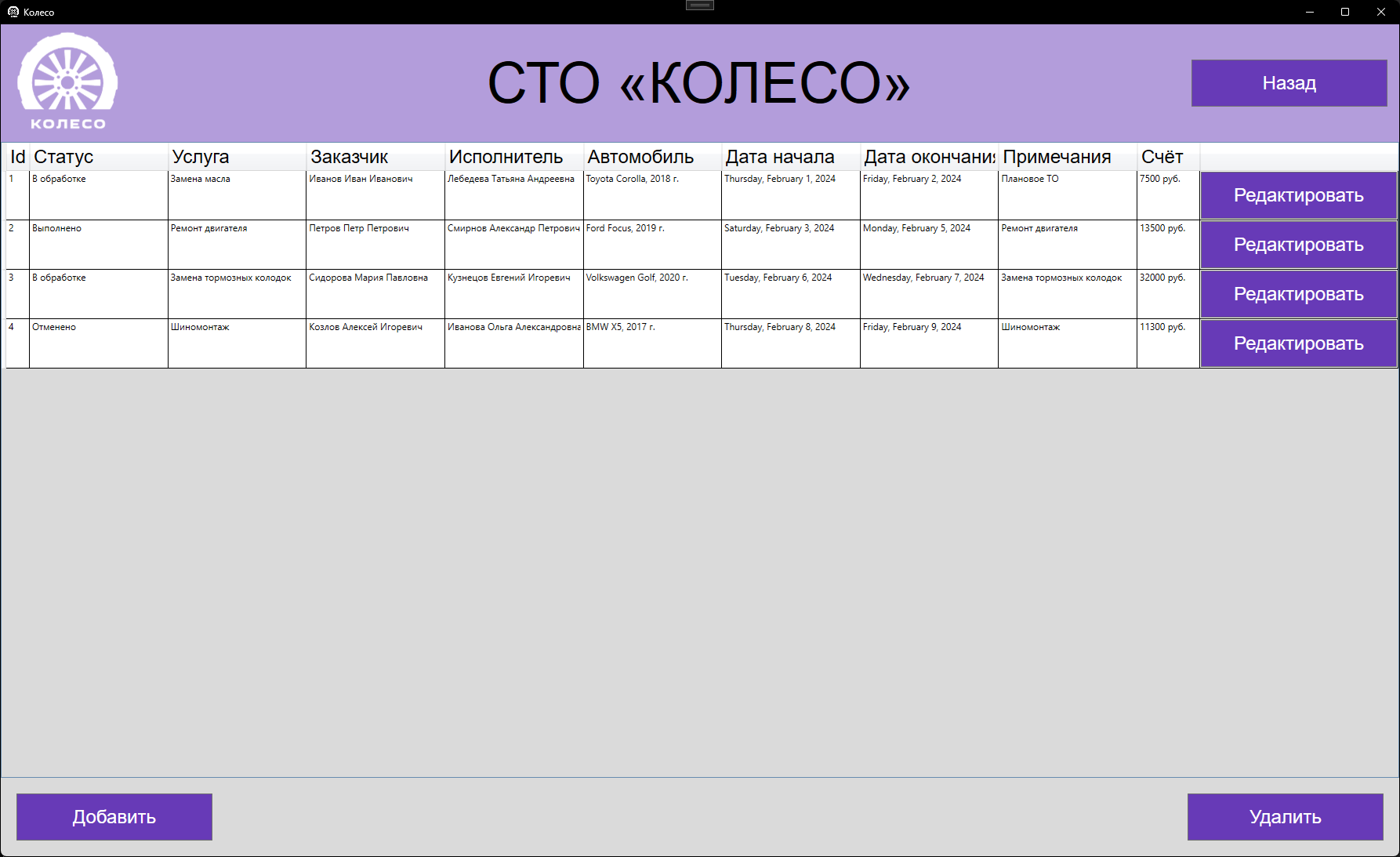


Рисунок 2.11 – Страница «Заявки на обслуживание»

Страница «Наши клиенты» содержит информацию о клиентах, обращавшихся в организацию и их контактные данные. Страница представлена на рисунке 2.12.

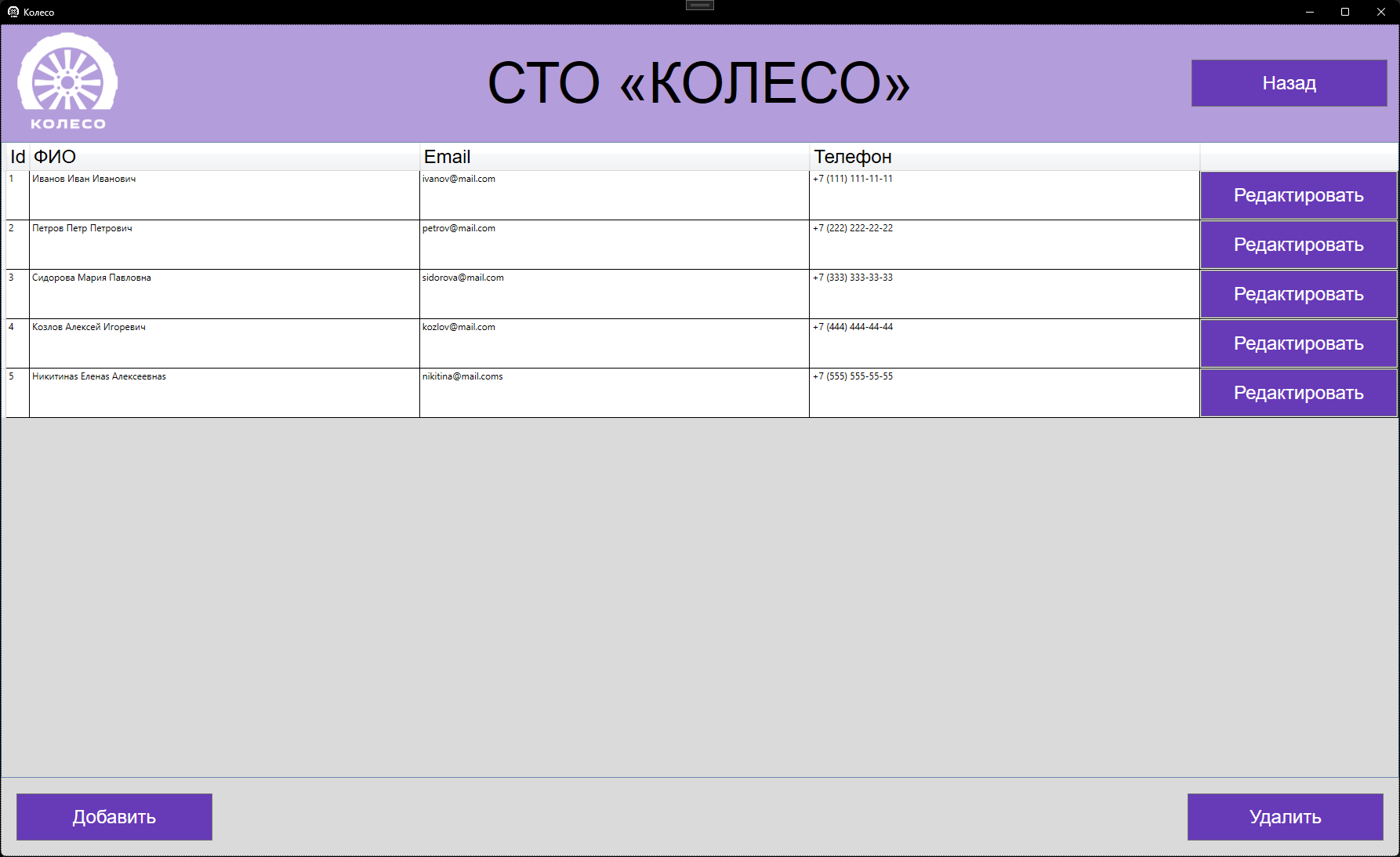


Рисунок 2.12 – Страница «Наши клиенты»

Также для добавления и редактирования данных разработана отдельная страница. Страница представлена на рисунке 2.13.

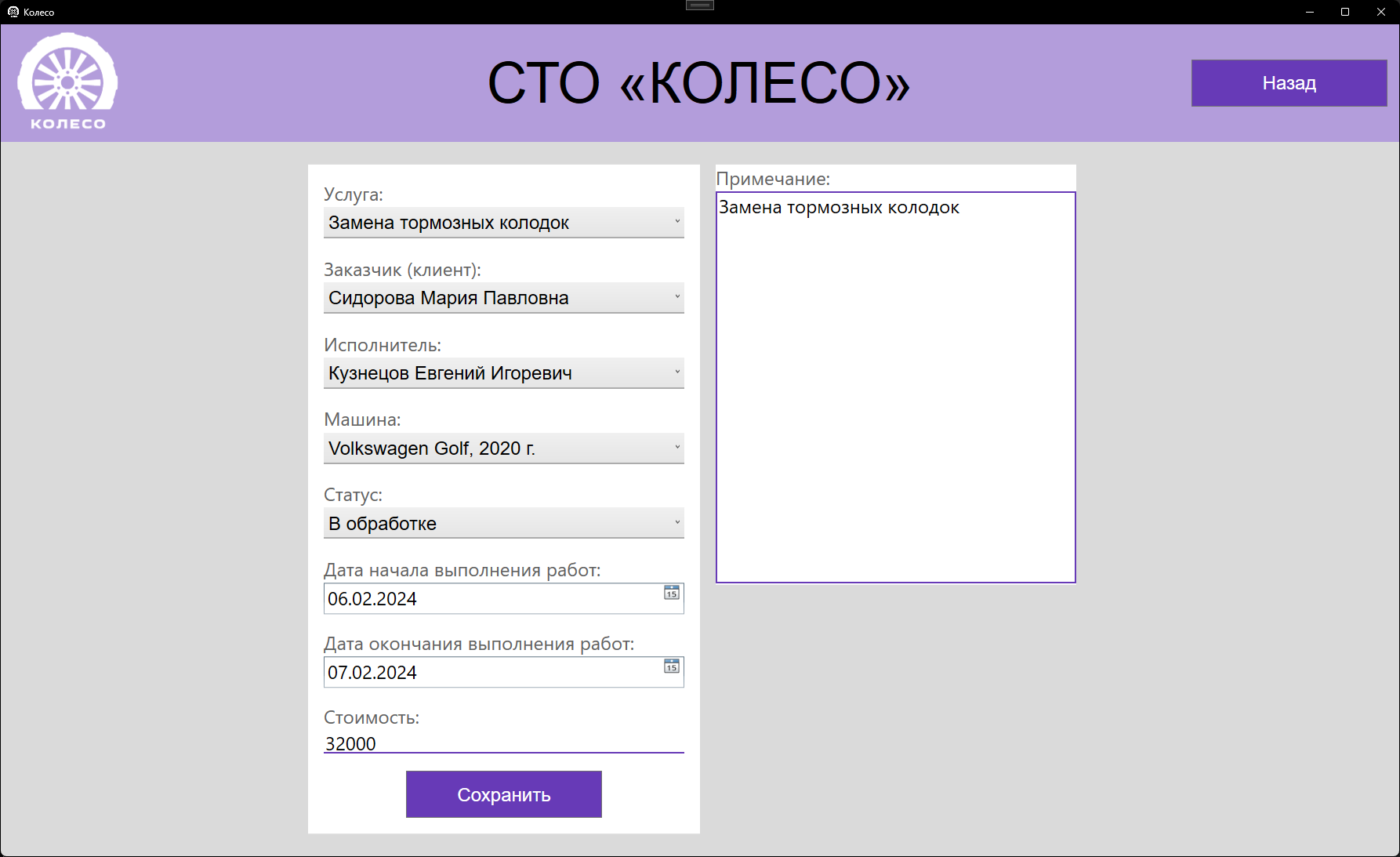


Рисунок 2.12 – Страница «Редактор клиента»

Страница «Услуг» содержит информацию о предоставляемых услугах в СТО. Страница представлена на рисунке 2.13.

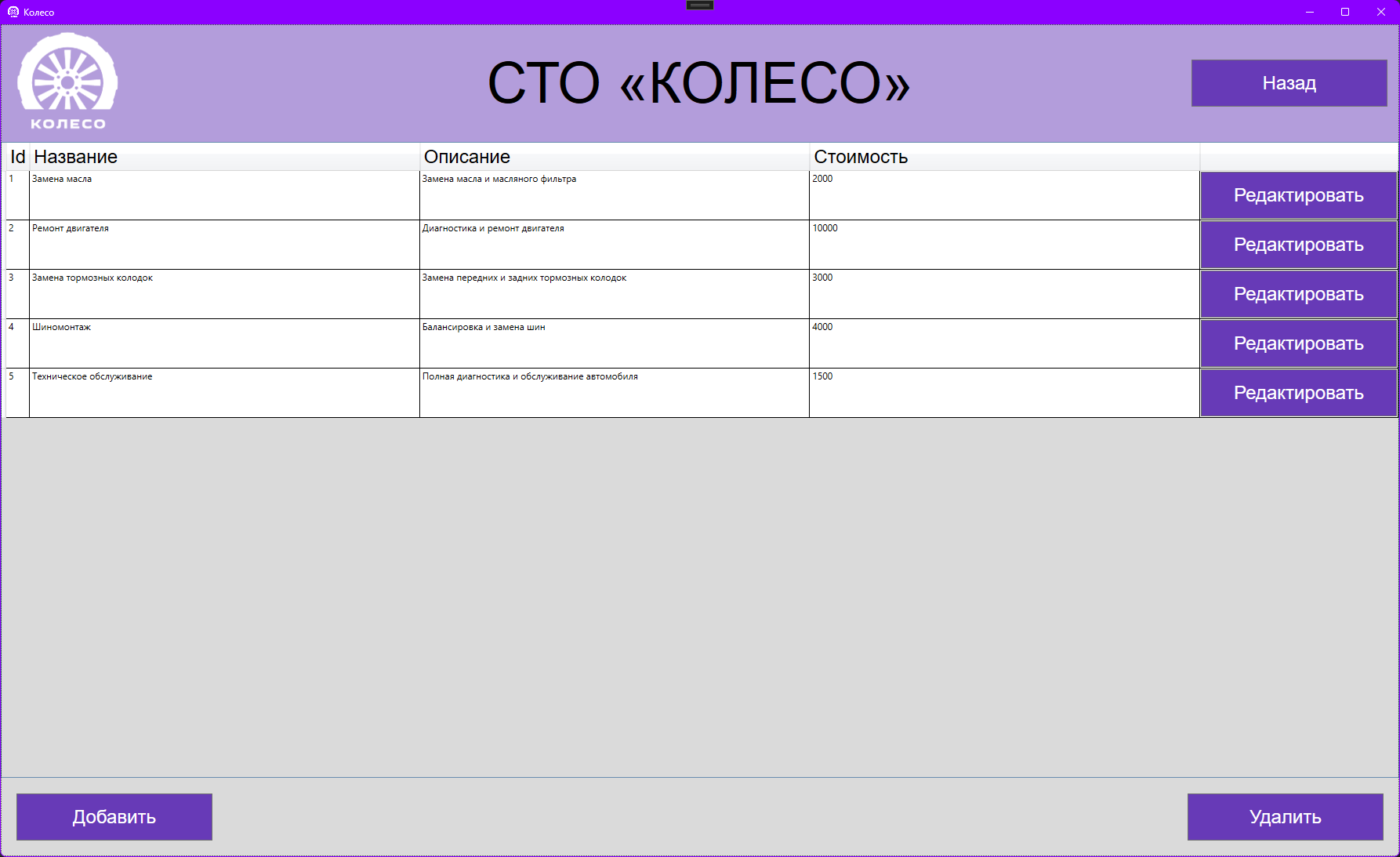


Рисунок 2.13 – Страница «Услуги»

Страница «Заявки на обслуживание» содержит информацию об предоставленных и актуальных заявках на оказание услуг. Страница представлена на рисунке 2.14.

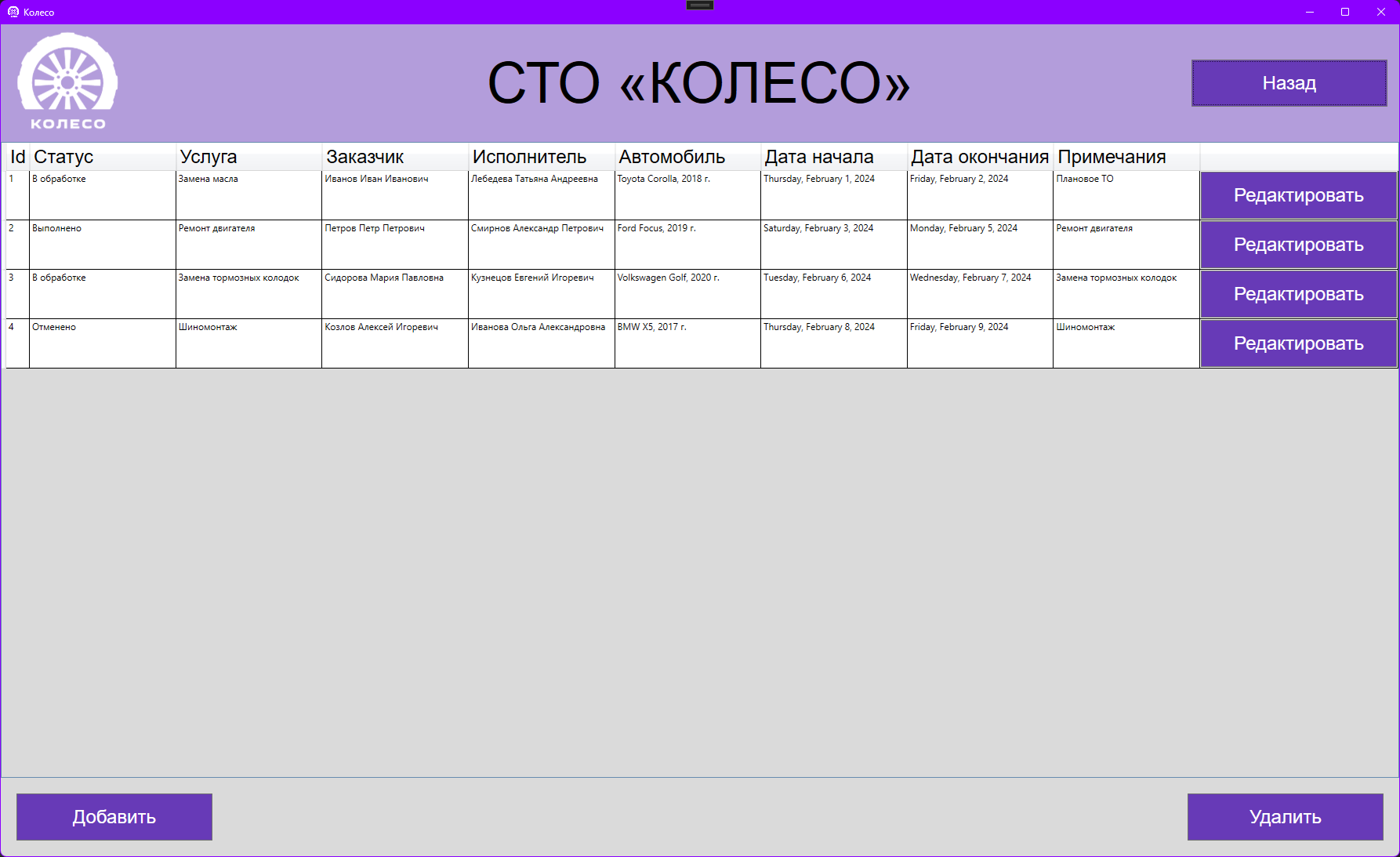


Рисунок 2.13 – Страница «Услуги»

* 1. **Написание программного кода**

Для авторизации пользователей и разделения их интерфейсов был разработан следующий код, который проверяет данные для входа и выдает соответствующий интерфейс в зависимости от роли пользователя. Код представлен на листинге 1.

Листинг 1 – Обработчик для авторизации пользователей

|  |
| --- |
| private void btnSignIn\_Click(object sender, RoutedEventArgs e) { StringBuilder errors = new StringBuilder(); if (string.IsNullOrEmpty(txtLogin.Text)) errors.AppendLine("Поле \"Логин\" должно быть заполнено!"); if (string.IsNullOrEmpty(pbPassword.Password)) errors.AppendLine("Поле \"Пароль\" должно быть заполнено!"); if (errors.Length > 0) { MessageBox.Show(errors.ToString(), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); pbPassword.Password = string.Empty; } else { users \_currentUser = DataBaseEntities.GetContext().users.FirstOrDefault(x => x.login == txtLogin.Text || x.password == pbPassword.Password); if (\_currentUser != null) { Manager.AuthUser = \_currentUser; MessageBox.Show($"Здраствуйте, {DataBaseEntities.GetContext().workers.FirstOrDefault(x => x.workerId == \_currentUser.workerId).FullName}!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); Manager.MainFrame.Navigate(new MainPage()); } else { MessageBox.Show("Пользователь с такими данными не найден!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } } } |

Перед внесение данных в БД, данные проверяются на корректность и их наличие в полях. Код для проверки данных представлен на листинге 2.

Листинг 2 – Обработчик на проверку корректности данных

|  |
| --- |
| StringBuilder errors = new StringBuilder(); if (string.IsNullOrEmpty(txtServiceName.Text)) errors.AppendLine("Укажите название услуги!"); if (string.IsNullOrEmpty(txtDescription.Text)) errors.AppendLine("Укажите описание услуги!"); if (string.IsNullOrEmpty(txtCost.Text)) errors.AppendLine("Укажите стоимость услуги!"); if (!decimal.TryParse(txtCost.Text, out decimal cost)) { errors.AppendLine("Стоимость услуги должна быть числом!"); } else if (cost < 0) { errors.AppendLine("Стоимость услуги не может быть отрицательной!"); } if (errors.Length > 0) { MessageBox.Show(errors.ToString(), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); return; } if (currentService.serviceId == 0) { try { DataBaseEntities.GetContext().servicesList.Add(currentService); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning); } } try { DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Информация сохранена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); Manager.MainFrame.GoBack(); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning); } |

Была реализована проверка автомобилей у клиента. При выборе клиента в заявки буду указываться только те транспортные средства, к которым он привязан, как владелец, а также при удалении клиента удаляются и данные об его транспортных средствах. Код свящи ТС и клиента представлен на листинге 3.

Листинг 3 – Обработчик привязки ТС к клиентам

|  |
| --- |
| var rowForRemoving = dgClients.SelectedItems.Cast<clients>().ToList(); if (rowForRemoving.Count == 0) { MessageBox.Show("Выберите клиентов для удаления!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } else if (rowForRemoving.Count > 1) { MessageBox.Show("Выберите не более одного клиента для удаления!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } else { StringBuilder message = new StringBuilder(); foreach (var client in rowForRemoving) { var clientCars = DataBaseEntities.GetContext().cars.Where(x => x.clientId == client.clientId).ToList(); if (clientCars.Any()) { message.AppendLine($"У клиента {client.FullName} есть следующие автомобили:"); foreach (var car in clientCars) { message.AppendLine($"- {car.FullInfo}"); } message.AppendLine("Вы уверены, что хотите удалить все данные об этом клиенте и его автомобилях?"); message.AppendLine(); } } if (message.Length > 0) { if (MessageBox.Show(message.ToString(), "Подтверждение удаления", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes) { try { foreach (var client in rowForRemoving) { var clientCars = DataBaseEntities.GetContext().cars.Where(x => x.clientId == client.clientId).ToList();  DataBaseEntities.GetContext().cars.RemoveRange(clientCars); }  DataBaseEntities.GetContext().clients.RemoveRange(rowForRemoving); |

Продолжение листинга 3

|  |
| --- |
| DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Данные удалены!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); DataBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(x => x.Reload()); dgClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } } } else { try { DataBaseEntities.GetContext().clients.RemoveRange(rowForRemoving); DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Данные удалены!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); DataBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(x => x.Reload()); dgClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } } } |

* 1. **Защита информации**

Защита информации – важный и обязательный пункт при создании программного обеспечения. С этой задачей рано или поздно придется столкнуться каждому разработчику приложений. Каждая составляющая должна быть надежно защищена.

Чтобы понять, как защищать, важно знать, откуда берутся вирусы. Основные пути заражения:

• Скачивание и установка непроверенных программ с внедренными вредоносными элементами, цель которых – перехват доступа к протоколам CMS и/или FTP с их дальнейшей обратной отправкой;

• Посещение зараженных порталов;

• Автоматический или ручной подбор логина и пароля злоумышленниками, используемые для входа на сервер или CMS, и заражение изнутри с помощью прописанного кода;

• Использование шаблонов, плагинов и других компонентов с уязвимыми местами и дырами, через которые можно управлять ресурсом;

• Действия пользователей, оставляющих вредоносный контент (ссылки, файлы);

• Размещение рекламы от непроверенных источников и партнеров.

Безопасность Windows

Ядро Windows предоставляет разные функции безопасности, которые формируют основу безопасности для всех приложений Windows, включая приложения, созданные с помощью WPF. Windows обеспечивает защиту путем перекомпиляции многих системных библиотек ядра, включая все зависимости, такие как CLR, чтобы снизить вероятность переполнения буфера. GS защищает от некоторых потенциальных переполнений буфера путем добавления специального файла cookie безопасности для защиты адреса функции, которая содержит буферы локальных строк. После возврата функции cookie безопасности сравнивается со своим предыдущим значением. Если значение изменилось, то возможно, произошло переполнение буфера, и процесс будет остановлен с состоянием ошибки. Остановка процесса предотвращает выполнение потенциально вредоносного кода.

Развертывание ClickOnce

ClickOnce — является инновационной технологией, которая позволяет развертывать приложения с максимальным уровнем безопасности. Она включена в состав .NET Framework и интегрируется с Visual Studio для удобства разработчиков.

Приложения, развернутые с помощью ClickOnce, приобретают дополнительный уровень безопасности посредством управления доступом для кода в самом приложении. Приложения, развернутые с помощью ClickOnce, запрашивают разрешения, которые им необходимы. Им предоставляются только разрешения, не расширяющие набор разрешений для зоны, в которой развертывается приложения. Таким образом, уменьшается вероятность повреждений на клиентском компьютере в случае перехвата приложения.

Технология ClickOnce также обеспечивает управление доступом для кода, что является дополнительным уровнем безопасности. Это означает, что приложения, использующие ClickOnce, запрашивают разрешения только на те действия, которые им необходимы, и не расширяют набор разрешений для зоны, в которой они работают.

Технология ClickOnce является важным инструментом для создания безопасных и надежных приложений. Она позволяет разработчикам управлять доступом для кода и уменьшить вероятность повреждений на клиентском компьютере.

* 1. **Тестирование ИС**

Тестирование – процесс выполнения программы с целью обнаружения ошибок. Тестирование обеспечивает:

– обнаружение ошибок;

– демонстрацию соответствия функций программы ее назначению;

– демонстрацию реализации требований к характеристикам программы;

– отображение надежности как индикатора качества программы.

Цель этапа тестирования – снижение риска, возникающего при вводе решения в промышленную эксплуатацию.

Пилотная эксплуатация – это тестирование решения в промышленной среде. Основная задача пилотной эксплуатации – продемонстрировать, что решение способно стабильно работать условиях промышленной эксплуатации и удовлетворяет требованиям бизнеса. В процессе пилотной эксплуатации решение испытывается в реальных условиях. Пилотная эксплуатация дает возможность пользователям высказать свое мнение о работе продукта. Руководствуясь этим мнением разработчиком, устраняются все возможные неполадки или создается план действий на случай непредвиденных обстоятельств. В конечном итоге, пилотная эксплуатация позволяет принять решение, стоит ли начинать полномасштабное развертывание или отложить до устранения неполадок, способных сорвать развертывание.

Для тестирования разработанной информационной системы были разработаны следующие unit-test’ы, который тестируют корректность и надежность пароля и логина веденных пользователем. Тесты проверяют минимальное и максимально количество символов, проверяют наличие специальных символов, а также верхний и нижний регистр. Код модульных тестов представлен на листингах 4 и 5.

Листинг 4 – Модульные тесты для логина

|  |
| --- |
| [TestMethod()] public void Check4Symbols\_ReturnsTrue() { // Arrange string login = "DADA"; bool expected = true; // Act bool actual = LoginCheck.ValidateLogin(login); // Assert Assert.AreEqual(expected, actual); } [TestMethod()] public void Check\_LoginWithSpecSymbols\_ReturnsFalse() { // Arrange string login = "DADA#"; bool expected = false; // Act bool actual = LoginCheck.ValidateLogin(login); // Assert Assert.AreEqual(expected, actual); } [TestMethod()] public void Check\_LoginWithOutSpecSymbols\_ReturnsTrue() { // Arrange string login = "Dada"; bool expected = true; // Act bool actual = LoginCheck.ValidateLogin(login); // Assert Assert.AreEqual(expected, actual); } |

Листинг 5 – Модульные тесты для пароля

|  |
| --- |
| [TestMethod()] public void CorrectPassword\_ReturnsTrue() { // Arange string Password = "Dasha2002##"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void CountSymbols8\_ReturnsTrue() { // Arange string Password = "ASDFGHHJ"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void CountSymbolsFrom30\_ReturnsFalse() { // Arange string Password = "DashaDashaDashaDashaDashaDasha"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void Check\_PasswordWithSpecSymbols\_ReturnsFalse() { // Arange string Password = "DADADADASD2231#"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void Check\_PassowordWithOutSpecSymbols\_ReturnsTrue() { // Arange string Password = "GHASYUGWYX"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void Check\_PasswordWithBigLetter\_ReturnsTrue() { // Arange string Password = "Dasha20241!"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void Check\_PasswordWithoutBigLetter\_ReturnsFalse() { // Arange string Password = "dasha20241!"; bool Expected = true; // Act bool Actual = PasswordCheck.ValidatePassword(Password); // Assert Assert.AreEqual(Expected, Actual); } [TestMethod()] public void Check\_PasswordWithSmallLetter\_ReturnsTrue() { // Arange string Password = "DASsD1123#"; bool Expected = true; Actual = PasswordCheck.ValidatePassword(Password); |

* 1. **Руководство пользователя ИС**

Область применения

Данное руководство пользователя предназначено для сотрудников станции технического обслуживания (СТО), включая администраторов, мастеров, механиков и других специалистов, которые будут использовать информационную систему для оптимизации рабочих процессов. Информационная система разработана для автоматизации и упрощения различных аспектов работы СТО, таких как регистрация клиентов, управление персоналом, учет и отчетность, а также улучшение качества обслуживания.

Краткое описание возможностей

1. Регистрация клиентов и автомобилей:

* Добавление новых клиентов: Внесение данных о новых клиентах и их транспортных средствах;
* Редактирование информации о клиентах: Обновление данных при необходимости;
* Управление списком клиентов: Просмотр и управление списком зарегистрированных клиентов и их автомобилей.

1. Запись на техническое обслуживание:

* Планирование услуг: Клиенты могут записываться на обслуживание в удобное для них время;
* Управление заявками: Отслеживание и управление текущими заявками на обслуживание и ремонт.

1. Управление персоналом:

* Редактирование информации о сотрудниках: Обновление данных при необходимости;
* Отслеживание рабочей деятельности сотрудников.

1. Обеспечение безопасности

* Защита данных: Использование механизмов шифрования для защиты конфиденциальной информации о клиентах и финансовых операциях;
* Управление доступом: Настройка прав доступа для сотрудников, чтобы обеспечить безопасный доступ к системе.

Основной функционал

1. Регистрация клиентов и их автомобилей
   1. Откройте модуль «Наши клиенты»;
   2. Нажмите кнопку «Добавить нового клиента»;
   3. Введите необходимые данные клиента (ФИО, контактная информация, данные об автомобиле и т.д.);
   4. Нажмите «Сохранить».
2. Редактирование информации о клиентах:
   1. Откройте модуль «Наши клиенты»;
   2. Найдите клиента в списке зарегистрированных;
   3. Нажмите кнопку «Редактировать»;
   4. Внесите необходимые изменения в данные;
   5. Нажмите «Сохранить изменения».
3. Редактирование информации о автомобилях:
   1. Откройте модуль «Обслуживаемые машины»;
   2. Найдите автомобиль в списке зарегистрированных;
   3. Нажмите кнопку «Редактировать»;
   4. Внесите необходимые изменения в данные;
   5. Нажмите «Сохранить изменения».
4. Запись на техническое обслуживание
   1. Откройте модуль «Заявки на обслуживание»;
   2. Выберите «Создать новую заявки»;
   3. Выберите клиента и автомобиль из списка;
   4. Выберите нужную услугу из предложенного списка;
   5. Выберите удобное время и дату для обслуживания;
   6. Нажмите "Записать".
5. Управление заявками
   1. Перейдите в модуль «Заявки на обслуживание»;
   2. Просматривайте все текущие и запланированные заявки;
   3. Обновляйте статус заявок (например, в процессе, завершено);
   4. Вносите изменения в заявки при необходимости.
6. Управление персоналом
   1. Откройте модуль «Сотрудника»;
   2. Найдите необходимого сотрудника в списке;
   3. Нажмите кнопку «Редактировать»;
   4. Внесите необходимые изменения в данные;
   5. Нажмите «Сохранить изменения».

Эта информационная система предназначена для использования внутри станции технического обслуживание и будет устанавливаться на компьютеры внутри технический помещений

**ЗАКЛЮЧЕНИЕ**

В данном дипломном проекте был представлен наглядный пример процесса разработки программного обеспечения для станции технического обслуживания «КОЛЕСО». В ходе проектирования информационной системы была построена инфологическая модель, модель предметной области в стандарте IDEF0, спроектирована и разработана база данных. Был определен состав информационной системы, спроектирован её интерфейс и в последующем реализован и протестирован. Разработанная информационная поможет автоматизировать и упростить процесс работы с заявками клиентов, хранения информации об автомобилях, клиентах, а также упростит процесс управления персоналом, формирование и введения учета и многое другое. Это позволит повысить эффективность работы станции технического обслуживания, улучшит качество обслуживания клиентов и увеличить прибыльность бизнеса. Кроме того, разработанная информационная система позволит хранить и обрабатывать большие объемы данных, что поможет агентству лучше понимать потребности своих клиентов и принимать более обоснованные и объективно выгодные решения.

Программное обеспечение полностью функционирует и оправдывает представления потенциальных пользователей.

При выполнении дипломного проекта была проанализирована заданная предметная область, изучены ее основные аспекты и базовые знания, которые могли помочь при разработке.

Созданное программное обеспечение имеет удобный и простой для восприятия интерфейс, выполнена в эстетически приятном минималистическом стиле, и отвечает нынешним современным стандартам. Данная система позволит организации облегчить работу, связанную с деятельностью сервиса технического обслуживания.

В рамках дипломного проекта были выполнены следующие ранее поставленные задачи:

* Спроектирована веб-система;
* Разработаны и спроектированы диаграммы DFD и IDEF0;
* Произведен анализ уже существующих схожих информационных систем;
* Изучены средства создания информационных систем и выбрана более подходящая;
* Произведена разработка базы данных и информационной системы, осуществлено подключение базы данных к информационной системе;
* Обеспечена защита информации;
* Произведено тестирование информационной системы.

Исходный код программы выложен на в систему контроля версий GitHub: https://github.com/Darya0207/diplom

**Список использованных источников**

1. **Волобуева, Т. В.** Информатика. Основы алгоритмизации: учебное пособие / Т. В. Волобуева. — Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2019. — 73 c. — ISBN 978-5-7731-0740-8. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROFобразование: [сайт]. — URL: https://profspo.ru/books/93316 (дата обращения: 28.02.2022). — Режим доступа: для авторизир. Пользователей
2. **Маркин, А. В.** Программирование на SQL: учебное пособие для среднего профессионального образования / А. В. Маркин. — Москва: Издательство Юрайт, 2023. — 435 с. — (Профессиональное образование). — ISBN 978-5-534-11093-7. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/518166
3. **Казанский, А. А.**  Программирование на Visual C#: учебное пособие для среднего профессионального образования / А. А. Казанский. — 2-е изд., перераб. и доп. — Москва: Издательство Юрайт, — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/513400
4. **Маркин, А. В.**  Программирование на SQL в 2 ч. Часть 1: учебник и практикум для вузов / А. В. Маркин. — 3-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 429 с. — (Высшее образование). — ISBN 978-5-534-15817-5. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/509818
5. **Подбельский, В. В.**  Программирование. Базовый курс С#: учебник для вузов / В. В. Подбельский. — Москва: Издательство Юрайт, 2023. — 369 с. — (Высшее образование). — ISBN 978-5-534-10616-9. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/511747

6. **Кудрявцева, И. А.**  Программирование: теория типов : учебное пособие для вузов / И. А. Кудрявцева, М. В. Швецкий. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 652 с. — (Высшее образование). — ISBN 978-5-534-11088-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/517823

7. **Маркин, А. В.**  Программирование на SQL в 2 ч. Часть 2 : учебник и практикум для вузов / А. В. Маркин. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 385 с. — (Высшее образование). — ISBN 978-5-534-15818-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/509819

8. **Маркин, А. В.**  Программирование на SQL : учебное пособие для среднего профессионального образования / А. В. Маркин. — Москва : Издательство Юрайт, 2023. — 435 с. — (Профессиональное образование). — ISBN 978-5-534-11093-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/518166

9. **Нестеров, С. А.**  Базы данных : учебник и практикум для среднего профессионального образования / С. А. Нестеров. — Москва : Издательство Юрайт, 2023. — 230 с. — (Профессиональное образование). — ISBN 978-5-534-11629-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/518507 (дата обращения: 12.05.2023).

10. **Тузовский, А. Ф.**  Объектно-ориентированное программирование : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2022. — 206 с. — (Высшее образование). — ISBN 978-5-534-00849-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/490369

**ПРИЛОЖЕНИЯ**

**Приложение А**

**Листинги программного кода**

**ControlMenuPage.xaml**

|  |
| --- |
| <Grid Background="#DADADA"> <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center"> <Border x:Name="brRequests" Style="{StaticResource clickElements}" MouseLeftButtonDown="brRequests\_MouseLeftButtonDown"> <StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center"> <Image Source="/Resources/requests.png" Width="150" Height="150" Margin="10 0 10 10"/> <TextBlock Text="Заявки на обслуживание" TextWrapping="Wrap" TextAlignment="Center" Width="320" Foreground="White" FontSize="40"/> </StackPanel> </Border> <Border x:Name="brClients" Style="{StaticResource clickElements}" MouseLeftButtonDown="brClients\_MouseLeftButtonDown"> <StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center"> <Image Source="/Resources/clients.png" Width="150" Height="150" Margin="10 0 10 10"/> <TextBlock Text="Наши клиенты" TextWrapping="Wrap" TextAlignment="Center" Width="200" Foreground="White" FontSize="40"/> </StackPanel> </Border> <Border x:Name="brCars" Style="{StaticResource clickElements}" MouseLeftButtonDown="brCars\_MouseLeftButtonDown"> <StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center"> <Image Source="/Resources/cars.png" Width="150" Height="150" Margin="10 0 10 10"/> <TextBlock Text="Обслуживаемые машины" TextWrapping="Wrap" TextAlignment="Center" Width="320" Foreground="White" FontSize="40"/> </StackPanel> </Border> </StackPanel> </Grid> |

**ControlMenuPage.xaml.cs**

|  |
| --- |
| public partial class ControlMenuPage : Page { public ControlMenuPage() { InitializeComponent(); } private void brRequests\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e) { Manager.SecondFrame.Navigate(new RequestsPage()); } private void brClients\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e) { Manager.SecondFrame.Navigate(new ClientsPage()); } private void brCars\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e) { Manager.SecondFrame.Navigate(new CarsPage()); } } |

**ClientsPage.xaml**

|  |
| --- |
| <Grid Background="#DADADA"> <Grid.RowDefinitions> <RowDefinition Height="\*"/> <RowDefinition Height="100"/> </Grid.RowDefinitions> <DataGrid x:Name="dgClients" Background="#DADADA" Grid.Row="0"> <DataGrid.Columns> <DataGridTextColumn Header="Id" Binding="{Binding clientId}" Width="auto"/> <DataGridTextColumn Header="ФИО" Binding="{Binding FullName}" Width="\*"/> <DataGridTextColumn Header="Email" Binding="{Binding email}" Width="\*"/> <DataGridTextColumn Header="Телефон" Binding="{Binding phone}" Width="\*"/> <DataGridTemplateColumn Width="auto"> <DataGridTemplateColumn.CellTemplate> <DataTemplate> <Button x:Name="btnEdit" Content="Редактировать" Click="btnEdit\_Click"/> </DataTemplate> </DataGridTemplateColumn.CellTemplate> </DataGridTemplateColumn> </DataGrid.Columns> </DataGrid> <Grid Grid.Row="1"> <Button x:Name="btnAdd" Content="Добавить" HorizontalAlignment="Left" Margin="20" Click="btnAdd\_Click"/> <Button x:Name="btnDel" Content="Удалить" HorizontalAlignment="Right" Margin="20" Click="btnDel\_Click"/> </Grid> </Grid> |

**ClientsPage.xaml.cs**

|  |
| --- |
| public partial class ClientsPage : Page { public ClientsPage() { InitializeComponent(); dgClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); } private void btnAdd\_Click(object sender, RoutedEventArgs e) { Manager.SecondFrame.Navigate(new AddEditClients(null)); } private void btnDel\_Click(object sender, RoutedEventArgs e) { var rowForRemoving = dgClients.SelectedItems.Cast<clients>().ToList(); if (rowForRemoving.Count == 0) { MessageBox.Show("Выберите клиентов для удаления!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } else if (rowForRemoving.Count > 1) { MessageBox.Show("Выберите не более одного клиента для удаления!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } else { StringBuilder message = new StringBuilder(); foreach (var client in rowForRemoving) { var clientCars = DataBaseEntities.GetContext().cars.Where(x => x.clientId == client.clientId).ToList(); if (clientCars.Any()) { message.AppendLine($"У клиента {client.FullName} есть следующие автомобили:"); foreach (var car in clientCars) { message.AppendLine($"- {car.FullInfo}"); } message.AppendLine("Вы уверены, что хотите удалить все данные об этом клиенте и его автомобилях?"); message.AppendLine(); } } if (message.Length > 0) { if (MessageBox.Show(message.ToString(), "Подтверждение удаления", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes) { try { foreach (var client in rowForRemoving) { var clientCars = DataBaseEntities.GetContext().cars.Where(x => x.clientId == client.clientId).ToList(); DataBaseEntities.GetContext().cars.RemoveRange(clientCars); } DataBaseEntities.GetContext().clients.RemoveRange(rowForRemoving); DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Данные удалены!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); DataBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(x => x.Reload()); dgClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } } } else { try { DataBaseEntities.GetContext().clients.RemoveRange(rowForRemoving); DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Данные удалены!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); DataBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(x => x.Reload()); dgClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } } } } private void btnEdit\_Click(object sender, RoutedEventArgs e) { Manager.SecondFrame.Navigate(new AddEditClients((sender as Button).DataContext as clients)); } private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e) { if (Visibility == Visibility.Visible) { DataBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(x => x.Reload()); dgClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); } } } |

**AddEditRequests.xaml**

|  |
| --- |
| <Grid Background="#DADADA"> <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center"> <StackPanel Orientation="Vertical" Background="White" Height="auto" Width="500" Grid.RowSpan="2"> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Услуга:" Style="{StaticResource HeaderBlock}"/> <ComboBox x:Name="cmbService" DisplayMemberPath="serviceName" SelectedItem="{Binding servicesList}"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Заказчик (клиент):" Style="{StaticResource HeaderBlock}"/> <ComboBox x:Name="cmbClients" DisplayMemberPath="FullName" SelectedItem="{Binding clients}" SelectionChanged="cmbClients\_SelectionChanged"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Исполнитель:" Style="{StaticResource HeaderBlock}"/> <ComboBox x:Name="cmbWorkers" DisplayMemberPath="FullName" SelectedItem="{Binding workers}"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Машина:" Style="{StaticResource HeaderBlock}"/> <ComboBox x:Name="cmbCars" DisplayMemberPath="FullInfo" SelectedItem="{Binding cars}"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Статус:" Style="{StaticResource HeaderBlock}"/> <ComboBox x:Name="cmbStatus" DisplayMemberPath="statusName" SelectedItem="{Binding statusList}"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Дата начала выполнения работ:" Style="{StaticResource HeaderBlock}"/> <DatePicker x:Name="dpDateStart" Loaded="DatePicker\_Loaded" SelectedDate="{Binding dateStart}"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Дата окончания выполнения работ:" Style="{StaticResource HeaderBlock}"/> <DatePicker x:Name="dpDateEnd" Loaded="DatePicker\_Loaded" SelectedDate="{Binding dateEnd}" /> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 20 20 0"> <TextBlock Text="Стоимость:" Style="{StaticResource HeaderBlock}"/> <TextBox x:Name="txtTotalCost" Text="{Binding totalCost}" Style="{StaticResource InputBox}"/> </StackPanel> <Button x:Name="btnSave" Content="Сохранить" Margin="20" VerticalAlignment="Bottom" Click="btnSave\_Click"/> </StackPanel> <StackPanel Orientation="Vertical" Margin="20 0 20 0" Background="White" VerticalAlignment="Top"> <TextBlock Text="Примечание:" Style="{StaticResource HeaderBlock}"/> <TextBox x:Name="txtNote" Text="{Binding notes}" Style="{StaticResource InputBox}" BorderThickness="2 2 2 2" Height="500" TextWrapping="Wrap" VerticalContentAlignment="Top"/> </StackPanel> </StackPanel> </Grid> |

**AddEditRequests.xaml.cs**

|  |
| --- |
| public partial class AddEditRequests : Page { serviceRequests currentRequest = new serviceRequests(); public AddEditRequests(serviceRequests \_selectedRequest) { InitializeComponent(); cmbService.ItemsSource = DataBaseEntities.GetContext().servicesList.ToList(); cmbClients.ItemsSource = DataBaseEntities.GetContext().clients.ToList(); cmbWorkers.ItemsSource = DataBaseEntities.GetContext().workers.ToList(); cmbStatus.ItemsSource = DataBaseEntities.GetContext().statusList.ToList(); cmbCars.IsEnabled = false; if (\_selectedRequest != null) currentRequest = \_selectedRequest; DataContext = currentRequest; } private void DatePicker\_Loaded(object sender, RoutedEventArgs e) { DatePicker datePicker = sender as DatePicker; if (datePicker != null && (datePicker.SelectedDate == null || datePicker.SelectedDate == DateTime.MinValue)) { datePicker.SelectedDate = DateTime.Today; } } private void btnSave\_Click(object sender, RoutedEventArgs e) { StringBuilder errors = new StringBuilder(); if (cmbService.SelectedItem == null) errors.AppendLine("Выберите услугу!"); if (cmbClients.SelectedItem == null) errors.AppendLine("Выберите заказчика (клиента)!"); if (cmbWorkers.SelectedItem == null) errors.AppendLine("Выберите исполнителя!"); if (cmbCars.SelectedItem == null) errors.AppendLine("Выберите машину!"); if (cmbStatus.SelectedItem == null) errors.AppendLine("Выберите статус!"); if (dpDateStart.SelectedDate == null) errors.AppendLine("Укажите дату начала работ!"); if (dpDateEnd.SelectedDate == null) errors.AppendLine("Укажите дату окончания работ!"); if (string.IsNullOrEmpty(txtTotalCost.Text)) { errors.AppendLine("Укажите стоимость!"); } else { // Пытаемся преобразовать введенный текст в число if (!decimal.TryParse(txtTotalCost.Text, out decimal totalCost)) { errors.AppendLine("Укажите корректную стоимость!"); } } if (errors.Length > 0) { MessageBox.Show(errors.ToString(), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); return; } if (currentRequest.clientId == 0) { try { DataBaseEntities.GetContext().serviceRequests.Add(currentRequest); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning); } } try { DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Информация сохранена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information); Manager.MainFrame.GoBack(); } catch (Exception ex) { MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning); } } private void cmbClients\_SelectionChanged(object sender, SelectionChangedEventArgs e) { clients selectedClient = cmbClients.SelectedItem as clients; cmbCars.ItemsSource = DataBaseEntities.GetContext().cars.Where(x => x.clientId == selectedClient.clientId).ToList(); cmbCars.Items.Refresh(); cmbCars.SelectedIndex = 0; cmbCars.IsEnabled = true; } } |

**App.xaml**

|  |
| --- |
| <Application.Resources> <Style TargetType="TextBlock"> <Setter Property="FontFamily" Value="Arial"/> <Setter Property="FontSize" Value="24"/> </Style> <Style TargetType="Button"> <Setter Property="Background" Value="#673AB7"/> <Setter Property="Foreground" Value="#FFF"/> <Setter Property="Height" Value="60"/> <Setter Property="Width" Value="250"/> <Setter Property="HorizontalAlignment" Value="Center"/> <Setter Property="VerticalAlignment" Value="Center"/> <Setter Property="VerticalContentAlignment" Value="Center"/> <Setter Property="Cursor" Value="Hand"/> </Style> <Style TargetType="TextBlock" x:Key="HeaderBlock"> <Setter Property="FontSize" Value="24"/> <Setter Property="Foreground" Value="#FF616161"/> <Setter Property="Margin" Value="0 0 0 2"/> </Style> <Style TargetType="TextBox" x:Key="InputBox"> <Setter Property="FontSize" Value="24"/> <Setter Property="Width" Value="460"/> <Setter Property="Height" Value="30"/> <Setter Property="Margin" Value="0 0 0 2"/> <Setter Property="HorizontalAlignment" Value="Left"/> <Setter Property="VerticalAlignment" Value="Center"/> <Setter Property="VerticalContentAlignment" Value="Center"/> <Setter Property="HorizontalContentAlignment" Value="Left"/> <Setter Property="BorderBrush" Value="#673AB7"/> <Setter Property="BorderThickness" Value="0 0 0 2"/> </Style> <Style TargetType="PasswordBox" x:Key="AuthPBox"> <Setter Property="FontSize" Value="24"/> <Setter Property="Width" Value="460"/> <Setter Property="Height" Value="30"/> <Setter Property="Margin" Value="0 0 0 2"/> <Setter Property="HorizontalAlignment" Value="Left"/> <Setter Property="VerticalAlignment" Value="Center"/> <Setter Property="VerticalContentAlignment" Value="Center"/> <Setter Property="HorizontalContentAlignment" Value="Left"/> <Setter Property="BorderBrush" Value="#673AB7"/> <Setter Property="BorderThickness" Value="0 0 0 2"/> </Style> <Style TargetType="Border" x:Key="clickElements"> <Setter Property="Width" Value="500"/> <Setter Property="Height" Value="500"/> <Setter Property="Margin" Value="10"/> <Setter Property="Cursor" Value="Hand"/> <Setter Property="BorderThickness" Value="2"/> <Setter Property="CornerRadius" Value="1000"/> <Setter Property="BorderBrush" Value="#673AB7"/> <Setter Property="Background"> <Setter.Value> <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0" SpreadMethod="Repeat" MappingMode="Absolute"> <GradientStop Color="#FF673AB7" Offset="0"/> <GradientStop Color="White" Offset="1"/> </LinearGradientBrush> </Setter.Value> </Setter> </Style> <Style TargetType="DataGrid"> <Setter Property="AutoGenerateColumns" Value="False"/> <Setter Property="IsReadOnly" Value="True"/> </Style> <Style TargetType="ComboBox"> <Setter Property="FontSize" Value="24"/> <Setter Property="Width" Value="460"/> <Setter Property="Height" Value="40"/> <Setter Property="Margin" Value="0 0 0 2"/> <Setter Property="HorizontalAlignment" Value="Left"/> <Setter Property="VerticalAlignment" Value="Center"/> <Setter Property="VerticalContentAlignment" Value="Center"/> <Setter Property="HorizontalContentAlignment" Value="Left"/> <Setter Property="BorderBrush" Value="#673AB7"/> <Setter Property="BorderThickness" Value="0 0 0 2"/> </Style> <Style TargetType="DatePicker"> <Setter Property="FontFamily" Value="Segoe UI"/> <Setter Property="FontSize" Value="24"/> <Setter Property="Height" Value="40"/> <Setter Property="Width" Value="460"/> <Setter Property="HorizontalAlignment" Value="Center"/> <Setter Property="VerticalContentAlignment" Value="Top"/> <Setter Property="Foreground" Value="Black"/> <Setter Property="VerticalContentAlignment" Value="Top"/> <Style.Resources> <Style TargetType="Button"> <Setter Property="VerticalAlignment" Value="Center" /> <Setter Property="VerticalContentAlignment" Value="Center" /> <Setter Property="VerticalContentAlignment" Value="Center" /> </Style> </Style.Resources> </Style> </Application.Resources> |