Технология разработки программного обеспечения

КП.084.

Министерство цифрового развития, связи и массовых коммуникаций РФ

Уральский технический институт связи и информатики (филиал)

ФГБОУ ВО «Сибирский государственный университет

телекоммуникаций и информатики» в г. Екатеринбурге

(УрТИСИ СибГУТИ)

КУРСОВОЙ ПРОЕКТ

по междисциплинарному курсу: «Технология разработки программного обеспечения»

на тему: «Разработать программу для автоматизации работы автосервиса ИП «Кирьянов»»

Вариант № 5

Выполнил:

Студент группы 084

Вотинов Ю.А.

Руководитель:

Ермоленко О.М.

Екатеринбург 2023

Отзыв руководителя

Министерство цифрового развития, связи и массовых коммуникаций РФ

Уральский технический институт связи и информатики (филиал)

ФГБОУ ВО «Сибирский государственный университет

телекоммуникаций и информатики» в г. Екатеринбурге

(УрТИСИ СибГУТИ)

Цикловая комиссия ИТ и АСУ кафедры ИСТ

ЗАДАНИЕ ДЛЯ КУРСОВОГО ПРОЕКТИРОВАНИЯ

По междисциплинарному курсу Технология разработки программного обеспечения

Студента 4 курса 084 группы

Вотинова Юрия Александровича

ТЕМА ЗАДАНИЯ Разработать программу для автоматизации работы автосервиса ИП «Кирьянов».

ИСХОДНЫЕ ДАННЫЕ

вариант 5

Составить программу, которая автоматизирует работу автосервиса, программа предназначена для учета клиентов. Сформировать и натроить базу данных, содержащую необходимые данные для корректной работы программы. Разработать систему авторизации администратора, а также дружественный интерфейс. Реализовать запросы по удалению, добавлению, обновлению пользователей, сотрудников, услуг.

При выполнении курсового проекта на указанную тему должны быть представлены:

1. Пояснительная записка

Введение; Анализ предметной области; Аналоги программного продукта; Требования к программному продукту; Выбор инструментальных средств; Разработка приложения; Разработка блок-схемы программы; Разработка базы данных; Проектирования графического интерфейса; Иструкция по работе с программой; Заключение; Библиография; Приложение А

2. Графическая часть

Дата выдачи Срок окончания

Председатель ЦК ИТиАСУ кафедры ИСТ Ермоленко О.М.

Преподаватель – руководитель Ермоленко О.М.

Студент Вотинов Ю.А.

Содержание

Введение

1. Обоснование темы курсового проекта 6

1.1 Анализ предметной области 7

[1.2 Аналоги программного продукта 7](file:///K:\КУРСАЧ\Отчет%20по%20курсовому%20(СМО-характеристики).docx#_Toc26617250)

[1.3 Требования к программному продукту 12](file:///K:\КУРСАЧ\Отчет%20по%20курсовому%20(СМО-характеристики).docx#_Toc26617250)

1.4 Выбор инструментальных средств разработки 13

2 Разработка приложения                     16

2.1 Разработка блок-схемы программы        16

2.2 Проектирование базы данных                                16

2.3 Проектирование графического интерфейса 19

2.4 Инструкция по работе с программой            22

Заключение       27

Библиография       28

Приложение A       29

# Введение

В современном мире, где технологии все больше влияют на наш повседневный опыт, автосервисы стремятся не только предоставлять высококачественные услуги по обслуживанию и ремонту автомобилей, но и внедрять современные технологии для оптимизации своей работы. Эффективное управление бизнес-процессами, точное ведение учета заказов и клиентской базы, а также оперативный анализ предоставленных услуг — все эти аспекты становятся ключевыми для успешного функционирования автосервиса.

Программа автоматизации автосервиса будет ориентирована на простоту использования и гибкость в настройке, чтобы соответствовать различным потребностям автосервисов разного масштаба. Результаты этого проекта обещают значительное сокращение времени на рутинные операции, минимизацию ошибок и улучшение общей производительности автосервиса.

Целью данного курсового проекта является разработка программного обеспечения, предназначенного для автоматизации работы автосервиса. Программа будет охватывать широкий спектр задач, начиная от учета клиентов и заказов, заканчивая ведением базы данных по обслуживанию и ремонту автомобилей. Использование современных информационных технологий позволит повысить эффективность бизнес-процессов, улучшить взаимодействие с клиентами и обеспечить точный учет всех операций.

Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть теоретический материал по теме;

- произвести анализ предметной области;

- разработать блок-схему программы;

- произвести выбор инструментальных средств разработки;

- разработать интерфейс программы;

- написать инструкцию по работе с программой;

- оформить пояснительную записку.

1 Обоснование темы курсового проекта

* 1. Анализ предметной области

ИП «Кирьянов» — небольшое предприятие, специализирующееся на обслуживании и ремонте автомобилей. Основные виды деятельности включают в себя техническое обслуживание, диагностику, а также капитальный и текущий ремонт транспортных средств. Владелец, ИП «Кирьянов», осуществляет свою деятельность в условиях конкурентного рынка, где эффективное управление процессами и качественное предоставление услуг становятся ключевыми компонентами успешного бизнеса.

В условиях современного бизнеса, где конкуренция на рынке автосервисных услуг значительно высока, эффективное управление всеми аспектами бизнеса становится стратегически важным. Внедрение программного обеспечения для автоматизации автосервиса имеет ряд весомых преимуществ:

- программа позволит оптимизировать учет заказов, управление запчастями, планирование работ и обслуживание клиентов, что повысит эффективность работы автосервиса.

- автоматизация позволит легко вести учет клиентов, предоставлять им оперативную информацию о состоянии ремонта и обслуживания, а также повысит общую удовлетворенность клиентов.

- программа сэкономит время и снизит риск ошибок при ведении учета заказов, запчастей, рабочего времени и финансовых операций.

В своей роли разработчика программного обеспечения, я обладаю опытом в создании индивидуальных решений для бизнеса. Мой опыт в области разработки программ, в том числе для автосервисов, позволяет мне предоставить качественное и индивидуализированное программное решение для ИП «Кирьянов». Кроме того, я способен адаптировать программу под конкретные потребности бизнеса, учитывая особенности автосервиса ИП «Кирьянов».

1.2 Рассмотрение аналогичных web-сервиосов

На сегодняшний день существует множество программных продуктов, предназначенных для автоматизации работы автосервисов. Рассмотрим несколько популярных аналогов, чтобы лучше понять контекст и конкурентное окружение.

1. autoSoft service manager (Рисунок 1.1).

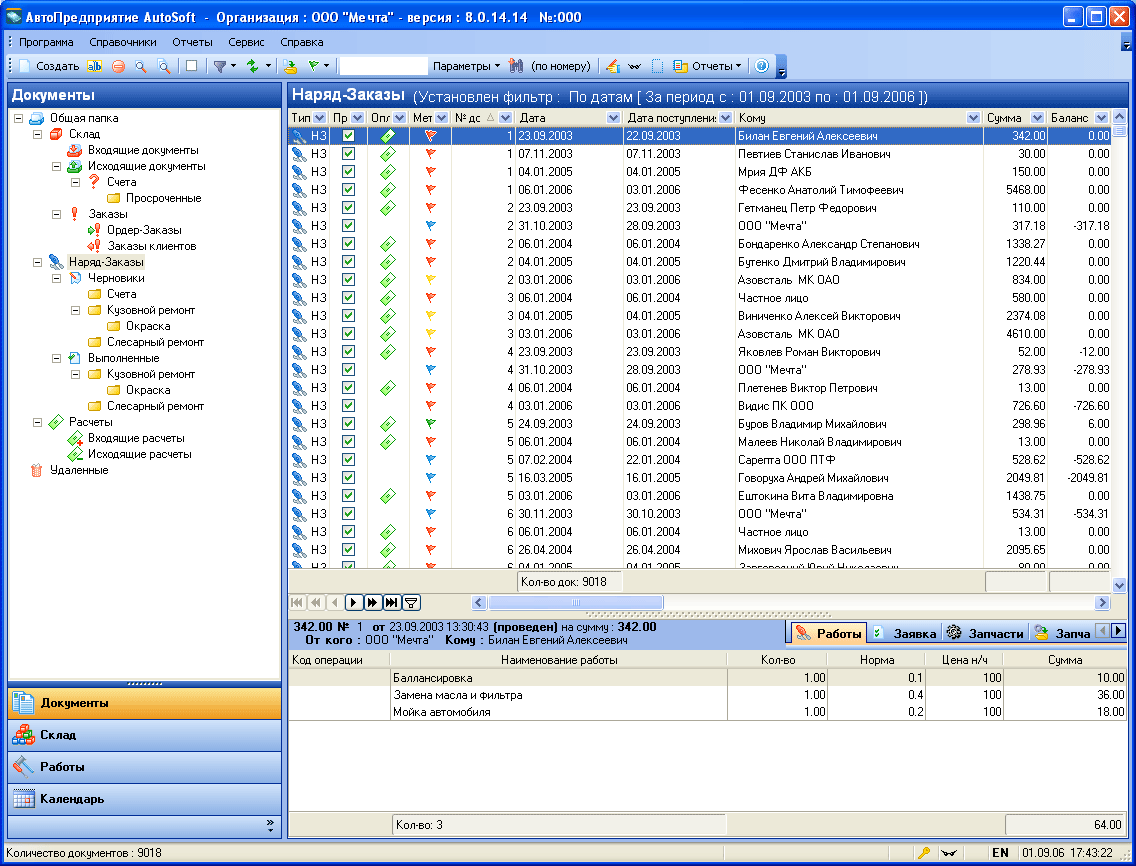


Рисунок 1.1- Вид программного продукта autosoft

* особенности, предоставляет полный набор инструментов для управления клиентскими заказами, учета запчастей, графика работы сотрудников и финансов (Рисунгок 1.2, 1.3, 1.4, 1.5).

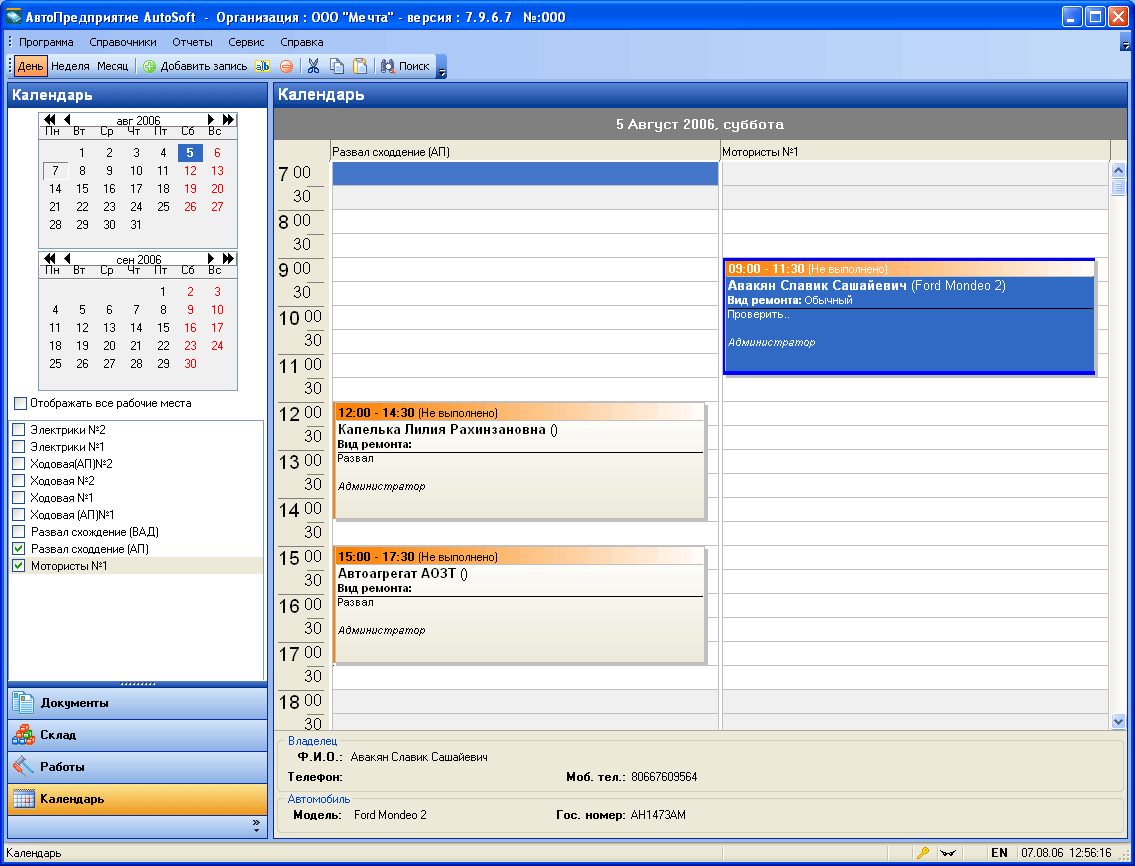


Рисунок 1.2 – Управление клиентскими заказами

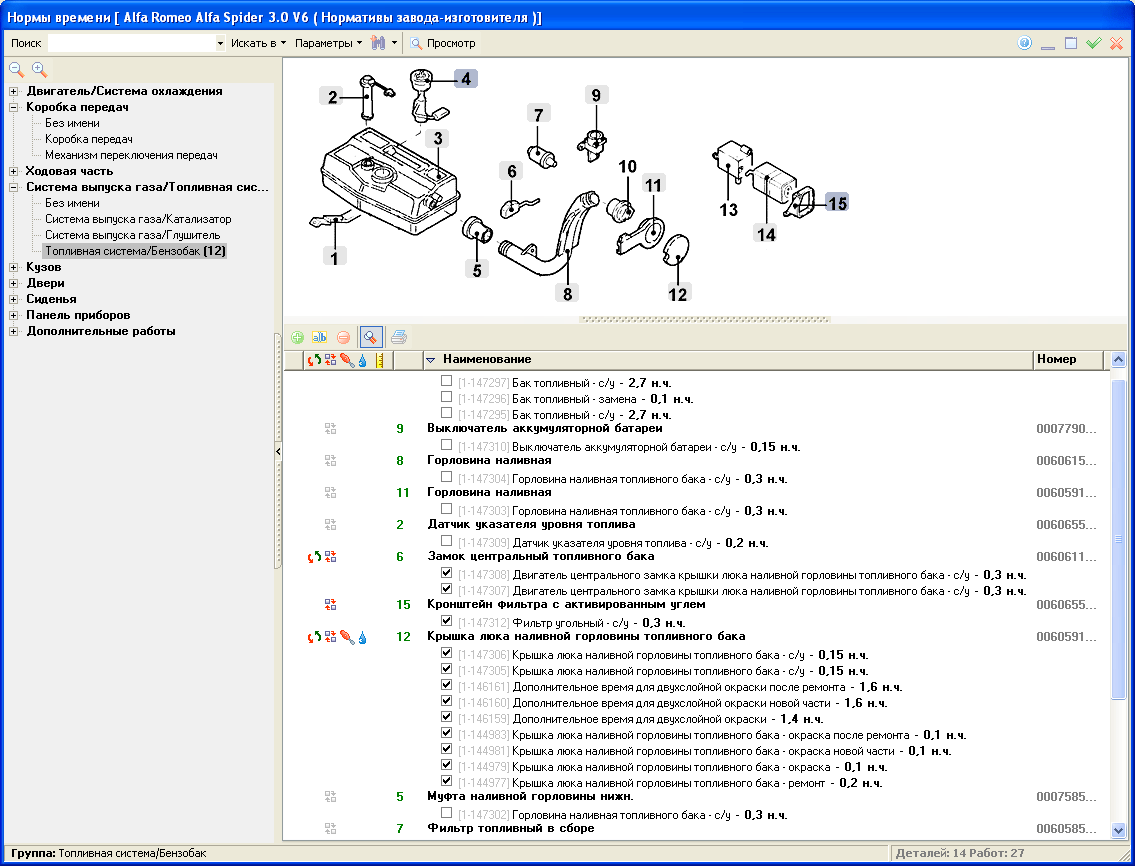


Рисунок 1.3 – Учет запчастей

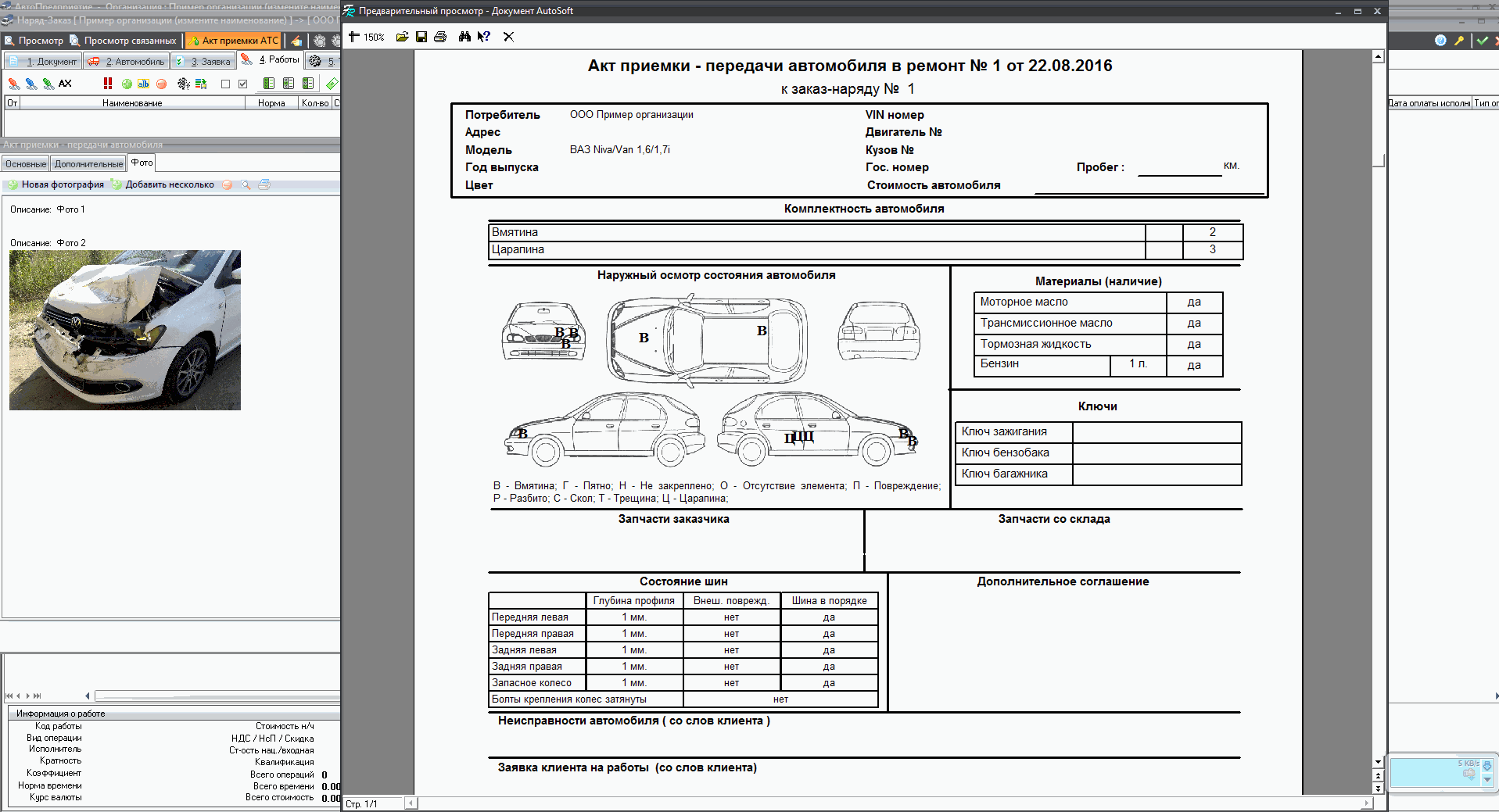
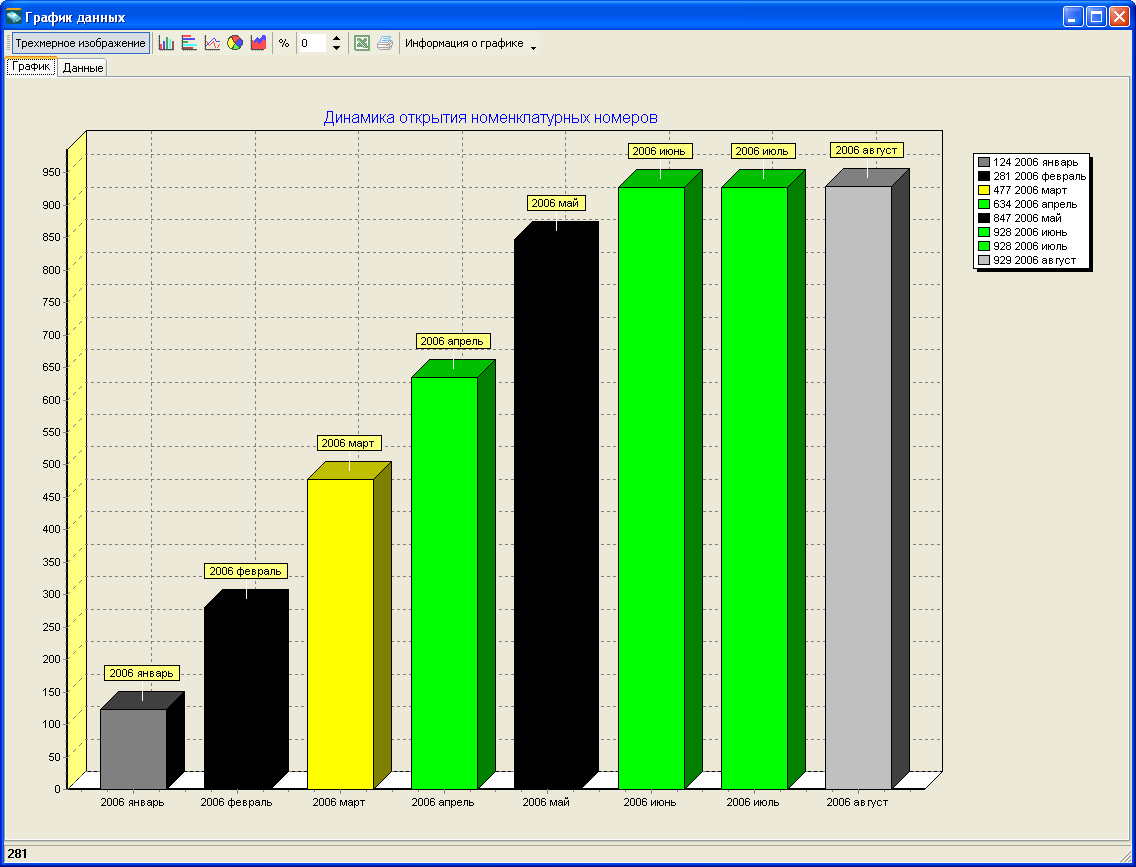


Рисунок 1.4 – Наряд заказы

Рисунок 1.5 – График данных

* преимущества, простота использования, подробные отчеты, многопользовательская поддержка.
* недостатки, ограниченные возможности по настройке под конкретные потребности.

2. Garage Keeper.

* особенности, включает в себя учет заказов, складского учета, ведение электронных чеков и генерацию отчетов.
* преимущества, интуитивно понятный интерфейс, поддержка мобильных устройств, управление персоналом.
* недостатки, не всегда предоставляет глубокую настройку под уникальные потребности.

3. WiseAuto.

* особенности, покрывает широкий спектр функций, включая учет заказов, ведение клиентской базы, управление складом и аналитику.
* преимущества, интеграция с бухгалтерскими системами, гибкая настройка, онлайн-поддержка.
* недостатки, требует времени для освоения, более высокая стоимость.

AutoSoft Service Manager и Garage Keeper обеспечивают более простой пользовательский интерфейс, что может быть важным для быстрого внедрения программы.

Гибкость настройки: WiseAuto выделяется гибкостью настройки, что делает его предпочтительным выбором для бизнесов с особыми требованиями.

Разные программы предлагают разные планы ценообразования. Важно учитывать бюджет ИП «Кирьянов» при выборе.

1.3 Требования к программному продукту

Требования к функциональным характеристикам.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций.

* аутентификация.
* управление аккаунтом.
* офформление заказа.
* управление заказами.
* административный функционал.
* безопасность.
* масштабируемость.
* хранение данных.
* мониторинг и логирование.

Требования к составу и параметрам технических средств.

В состав технических средств должен входить IВМ-совместимый персональный компьютер (ПЭВМ):

* процессор Pentium-2.0Hz, не менее;
* оперативная память объемом, 1Гигабайт, не менее;
* HDD, 40 Гигабайт, не менее;
* операционную систему Windows 7, не менее;
* пакет .NET Framework.

Требования к запросам пользователей.

Пользователи работают с программой через графический интерфейс.

Конкретные требования не предъявляются.

Дополнительные и специальные требования.

Доступ к высокоскоростному Интернет-соединению.

Эти требования обеспечат создание надежной, безопасной и удобной для использования серверной части web-приложения для покупки и продажи цифровых предметов.

1.4 Выбор инструментальных средств разработки

Изучая инструментальные средства разработки, был выбран язык программирования C#. Язык программирования C# имеет объектно-ориентированные функции.

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, LINQ, исключения, комментарии в формате XML и др. Все эти функции очень удобны при работе над разработкой какого-либо продукта.

Переняв многое от своих предшественников – языков C++, Delphi, Модула, Smalltalk и, в особенности, Java, – С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем. Например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

Достоинства и недостатки языка.

Можно выделить следующие достоинства C#:

- независимость от аппаратного функционала. Программу не нужно адаптировать под многочисленные платформы и операционные системы. Виртуальная машина .NET Framework сама выполняет эту задачу. В конечном итоге программист может использовать один и тот же код на самых разных устройствах: смартфонах, компьютерах, серверах, банкоматах, умных часах и т.д;

- поддержка. Как уже было упомянуто, данный язык был создан компанией Microsoft. Именно поэтому C# лучше всего подходит для работы с экосистемой Windows;

- управление памятью. Если необходимо, чтобы программа работал как можно стабильнее, необходимо выполнять ее очистку от различного мусора. Язык программирования C# выполняет эту задачу в автоматическом режиме. Таким образом, разработчику не нужно контролировать расход памяти, устранять её утечки или удалять ненужные блоки кода;

- строгая типизация. Если нужно объявить переменную в C#, необходимо сначала указать, что в ней лежит – строка, число или массив. Разработка таким методом выполняется немного дольше. Однако написанный код становится более предсказуемым, так как числа взаимодействуют с числами, строки со строками и т.д. Да, языки со слабой типизацией предоставляют большую свободу действий, но вероятность допустить ошибку при создании или проверке кода также будет выше;

- крупное сообщество. С# используется миллионами программистов. Можно найти в социальных сетях многочисленные чаты и сообщества программистов. Там есть ответы на все важные вопросы по поводу работы с языком. Более того, можно найти ментора, который сможет обучить данному языку;

- «синтаксический сахар». Используя С#, можно сократить код без вреда для логики программы. Для этого нужно применить один из нескольких способов, которые принято называть «синтаксическим сахаром». Данные приемы позволяют сделать код более легким и понятным. Однако эти способы не следует применять на постоянной основе. В противном случае может развиться так называемый синтаксический диабет, при котором код будет слишком зашифрован.

Недостатки языка:

- невысокая скорость. При открытии программы на C#, код сначала адаптируется под конкретное аппаратное обеспечение, а уже потом исполняется. Таким образом, скорость загрузки становится значительно ниже. При первом запуске интерфейсы на базе С# могут даже подтормаживать;

- слабое взаимодействие с аппаратным обеспечением. С# является высокоуровневым языком, поэтому он редко используется для разработки, предполагающей полное взаимодействие с аппаратным обеспечением (игровые движки, операционные системы, авиационный софт и т.д.).

Эти отдельные недостатки носят субъективный, локальный, временный характер.

Данные недостатки незначительны для выполнения данного проекта, так как программа не выполняет очень важных по времени функций и не нуждается в максимальном ускорении выполнения программы, а также не работает на прямую с аппаратным обеспечением компьютера, поэтому выбор остается на С#.

Кроме всего на языке C# можно написать приложение с современным графическим интерфейсом с использованием фреймворка WPF и поддержкой XAML, что очень улучшает внешний вид приложения. Данный фреймворк позволяет создавать более гибкий графический интерфейс и упрощает процесс его редактирования.

Для написания кода программы необходима среда разработки, которая оптимально подходит для решения поставленной задачи.

При выборе среды разработки и отталкиваясь от ее функционала, быстродействия, удобства, гибкости и популярности, выбор пал на IDE Visual Studio. Данная среда разработки уступает по быстродействию многим другим, но при этом ее удобство и функционал на достаточно высоком уровне по сравнению с конкурентами.

Visual Studio – это популярная стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Достоинства.

В число ее преимуществ входит следующее:

- поддержка C#, VB, XAML, HTML, JavaScript, TS и других языков;

- отлично подходит для создания различного программного обеспечения;

- мощная поддержка навигации и рефакторинга;

- прекрасно реализованная поддержка интеллектуальных сочетаний клавиш;

- интеграция с Visual Studio и Unity;

- хорошая визуализация процессов и кода.

Недостатки.

Недостатками среды разработки являются: невысокое быстродействие из-за концепции «всё в одном», некоторые плагины (в том числе для разработки приложений для Android) имеют существенные ограничения функциональности.

При своих незначительных недостатках, среда разработки Visual Studio отлично подходит для реализации различных проектов любой сложности. Исходя из достоинств выбор остается за Visual Studio.

Исходя из того, что функционирование программы подразумевает использование базы данных, необходимо подобрать систему управления базами данных (СУБД) подходящую для эффективного решения поставленных задач.

Благодаря тому, что Visual Studio и язык программирования C# хорошо подходят для работы с СУБД MySQL, а также многие хостинги поддерживают данную СУБД, была выбрана система MySQL для дальнейшей работы.

MySQL представляет собой реляционную СУБД, которая относится к свободному программному обеспечению: она распространяется на условиях GNU Public License. Как правило, эту систему управления базами данных определяют, как хорошую, быструю и гибкую, рекомендованную к применению в небольших или средних проектах.

База данных должна хранится на каком-либо хостинге, сервере, или же на компьютере, как в данном случае. В связи с этим необходимо подобрать соответствующую сборку веб-сервера для работы с локальными базами данных. С помощью него будет производится работа с базой данных для приложения.

Проведя анализ доступных и подходящих веб-серверов, был выбран локальный сервер «WampServer». Данный локальный сервер работает с реляционной СУБД MySQL, а также содержит Apache, интерпретатор скриптов PHP, phpMyAdmin и другие дополнения.

Данный локальный сервер имеет достаточно высокий показатель качества обслуживания. «WampServer» приобрел доверие уже у многих пользователей и является одним из ведущих программных средств в данной сфере.

«WampServer» зарекомендовал себя, как качественный и производительный продукт, который подойдет для тестирования баз данных для различных проектов.

Процесс работы с базой данных в «WampServer» очень прост и не занимает много времени. Шансы возникновения каких-либо проблем с программой минимальны в связи с ее достаточной надежностью и производительностью.

Данный локальный сервер бесплатный и может использоваться под различные нужды и задачи.

Веб-сервер «WampServer» отлично подходит для решения поставленных задач и будет использоваться для дальнейшей работы.

2 Разработка приложения

2.1Разработка блок схемы программы

Блок-схема работы программы представлена на рисунке 2.1

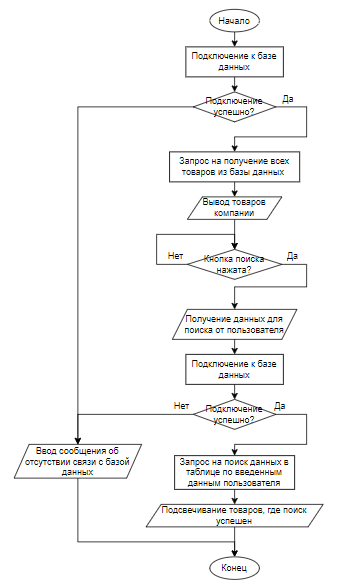


Рисунок 2.1– Блок схема программы

2.2 Проектирование базы данных

На рисунке 2.2 представлена схема доступа различных типов пользователей базы данных, к определенным действиям.



Рисунок 2.2 – Схема доступа для различных пользователей

На рисунке 2.3 показана ER-диаграмма или же инфологическая модель базы данных.

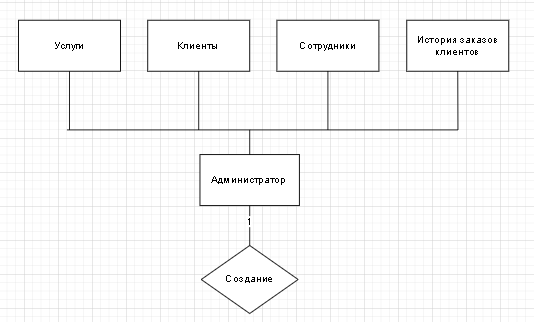


Рисунок 2.3 – ER- диаграмма базы данных

Даталогическая модель базы данных показана на рисунке 2.4, а также в таблицах 2.1-2.4 представлены атрибуты сущностей базы данных.

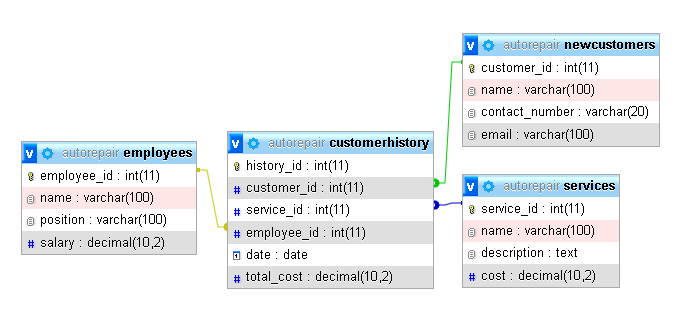


Рисунок 2.4 – Даталогическая модель базы данных

В таблицах 2.1-2.3 представлены атрибуты сущностей коллекций

Таблица 2.1 – Список атрибутов коллекции «Услуги» (services)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, Длина | Примечание |
| Идентификатор | Service\_id | int | Первичный ключ, уникальный |
| Название | name | varchar | Обязательное поле |
| Описание | description | text | Обязательное поле |
| Цена | cost | decimal | Обязательное поле |

Таблица 2.2 – Список атрибутов коллекции «Сотрудники» (Employees)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, Длина | Примечание |
| Идентификатор | employee\_id | int | Первичный ключ, уникальный |
| Имя | name | varchar | Обязательное поле |
| Должность | position | varchar | Обязательное поле |
| Зарплата | salary | decimal | Обязательное поле |

Таблица 2.3 – Списпок атрибутов коллекции «Корзина Пользователя» (Cart)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, Длина | Примечание |
| Идентификатор | customer\_id | int | Первичный ключ, уникальный |
| Имя | name | varchar | Обязательное поле |
| Номер телефона | contact\_number | varchar | Обязательное поле |
| Почта | email | varchar | Обязательное поле |

Таблица 2.4 – Списпок атрибутов коллекции «Корзина Пользователя» (Cart)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, Длина | Примечание |
| Идентификатор | history\_id | int | Первичный ключ, уникальный |
| Идентификатор пользователя | customer\_id | int | Обязательное поле, вторичный ключ |
| Идентификатор услуги | service\_id | int | Обязательное поле, вторичный ключ |
| Идентификатор сотрудника | employee\_id | int | Обязательное поле, вторичный ключ |
| Дата | date | date | Обязательное поле |
| Сумма | total\_cost | decimal | Обязательное поле |

2.3 Проектирование графического интерфейса

Макет графического интерфейса программы.

Стартовое окно программы (окно авторизации) представлено на рисунке 2.5.

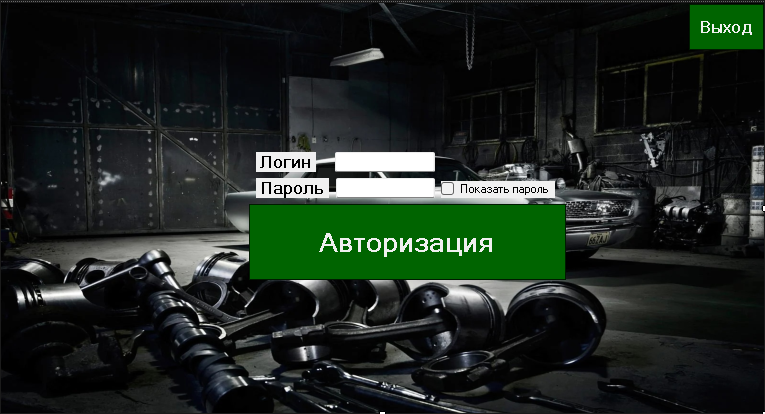


Рисунок 2.5 – Макет окна авторизации

На рисунке 2.6 представлен макет главной страницы администратора.



Рисунок 2.6 – Макет страницы администратора

На рисунке 2.7 показан макет страницы, на которой размещена информация о услагах предоставляемых предприятием.

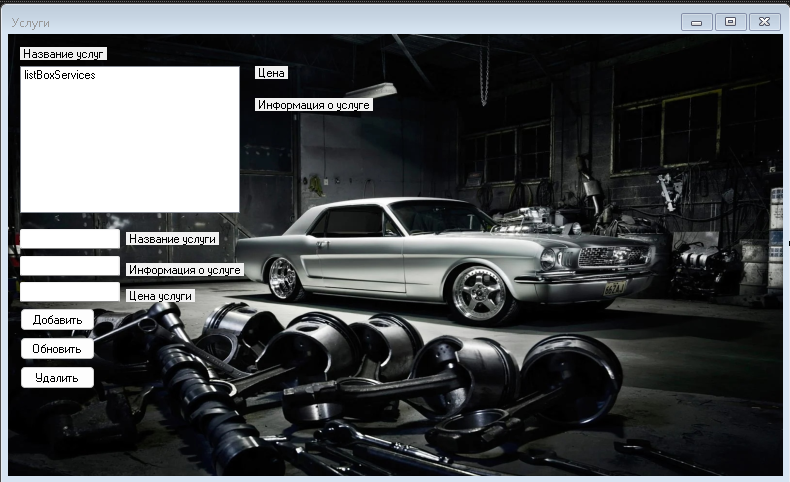


Рисунок 2.7 – Макет страницы услуги

На рисунке 2.8 показан макет страницы, на которой размещена информация о сотрудниках предприятия.



Рисунок 2.8 – Макет страницы сотрудники

На рисунке 2.9 показан макет страницы, на которой размещена информация о сотрудниках предприятия.

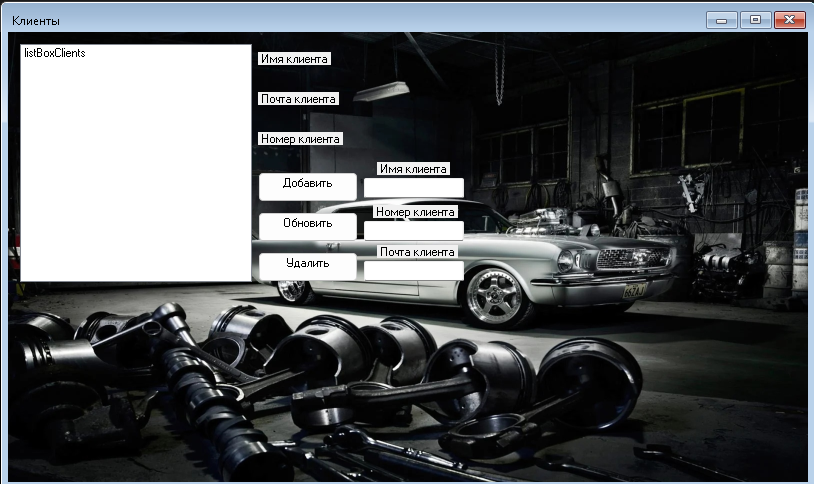


Рисунок 2.9 – Макет страницы клиенты

2.4 Инструкция по работе с программой

Инструкция для работы с программой «Автосервис ИП Кирьянов».

1.Авторизация (Рисунок 2.10).

* при запуске программы вас встретит форма авторизации.
* введите логин (admin) и пароль (password).
* нажмите кнопку «Войти».

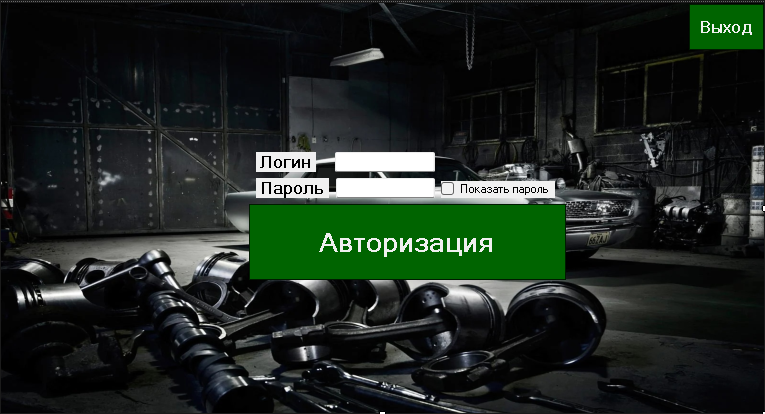


Рисунок 2.10 – Авторизация в программе

2. Главное меню (Рисунок 2.11).

* после успешной авторизации откроется главное меню с кнопками «Услуги», «Сотрудники» и «Клиенты».
* выберите необходимый раздел для работы.



Рисунок 2.11 – Главное меню программы

3. Услуги (Рисунок 2.12).

* на форме услуг вы можете просмотреть список доступных услуг автосервиса.
* добавить новую услугу: заполните поля «Название», «Стоимость» и нажмите «Добавить».
* обновить услугу: выберите услугу из списка, внесите изменения и нажмите «Обновить».
* удалить услугу: выберите услугу из списка и нажмите «Удалить».

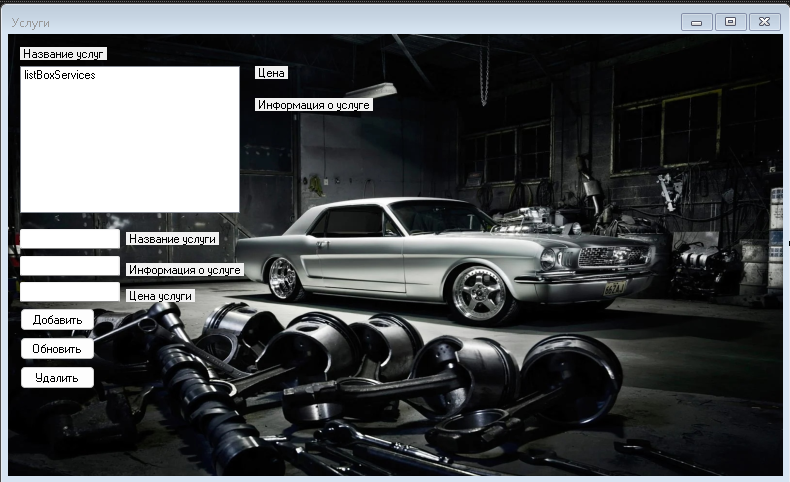


Рисунок 2.12 – Страница услуги

1. Сотрудники (Рисунок 2.13).

* на форме сотрудников представлен список сотрудников автосервиса.
* добавить нового сотрудника: заполните поля «Имя», «Должность», «Зарплата» и нажмите «Добавить».
* обновить данные сотрудника: выберите сотрудника из списка, внесите изменения и нажмите «Обновить».
* удалить сотрудника: выберите сотрудника из списка и нажмите «Удалить».



Рисунок 2.13 – Страница сотрудники

5. Клиенты (Рисунок 2.14).

* на форме клиентов отображается список клиентов автосервиса.
* добавить нового клиента: заполните поля «Имя клиента», «Номер телефона», «Почта» и нажмите «Добавить».
* обновить данные клиента: выберите клиента из списка, внесите изменения и нажмите «Обновить».
* удалить клиента: выберите клиента из списка и нажмите «Удалить».

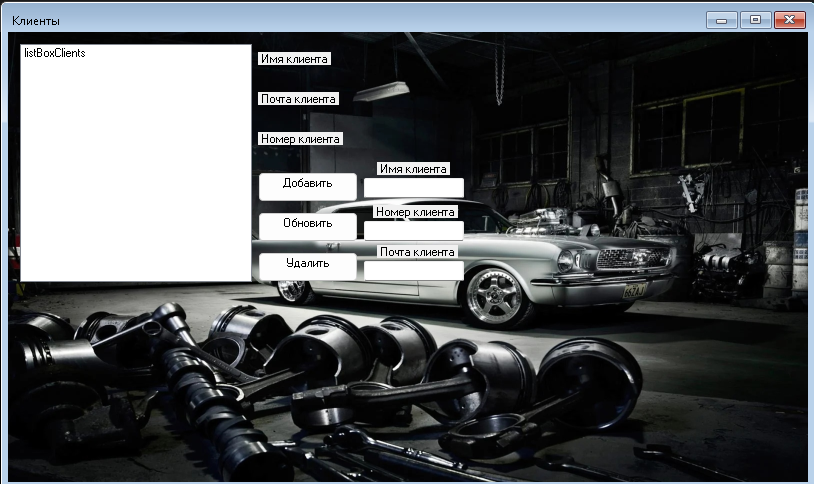


Рисунок 2.14 – Страница клиенты

6.  Выход.

* для выхода из программы нажмите «Выход» в верхнем правом углу.

Все операции добавления, обновления и удаления подлежат подтверждению.

Перед выполнением операций удостоверьтесь, что все обязательные поля заполнены корректно.

Для каждой формы предусмотрены проверки на наличие выбранных элементов перед выполнением операций.

Эта инструкция предоставляет основные указания для использования программы «Автосервис ИП Кирьянов».

# Заключение

Целью данного курсового проекта являлась разработка программного обеспечения, предназначенного для автоматизации работы автосервиса.

Для достижения поставленной цели были решины следующие задачи:

- рассмотрен теоретический материал по теме;

- произведен анализ предметной области;

- разработана блок-схема программы;

- произведен выбор инструментальных средств разработки;

- разработан интерфейс программы;

- написана инструкция по работе с программой;

- оформлена пояснительная записка.

Разработка программы для автоматизации работы автосервиса ИП «Кирьянов» представляет собой важный шаг в совершенствовании бизнес-процессов данной организации. Создание данного программного продукта обусловлено несколькими ключевыми факторами.

В ходе анализа предметной области было выявлено, что автосервис ИП «Кирьянов» является динамичным предприятием, специализирующимся на обслуживании и ремонте автотранспорта. С учетом увеличения объемов работы и клиентской базы, необходимо эффективно управлять информацией о предоставляемых услугах, сотрудниках и клиентах.

Проведенное исследование рынка программных продуктов выявило необходимость создания индивидуального решения, специально адаптированного к уникальным потребностям ИП «Кирьянов». Аналогичные программы либо не предоставляли всего необходимого функционала, либо были избыточными и сложными для использования.

Выбор в пользу разработки данной программы обусловлен моим пониманием конкретных потребностей заказчика, гибкостью внесения изменений и поддержкой в процессе эксплуатации. Программа ориентирована на упрощение рабочего процесса, улучшение качества обслуживания клиентов и повышение эффективности управления персоналом.

В результате успешной реализации программы для автосервиса ИП «Кирьянов» ожидается увеличение производительности, сокращение времени на выполнение операций и повышение уровня удовлетворенности как сотрудников, так и клиентов. Создание данного программного продукта - это шаг к современному, эффективному и конкурентоспособному автосервису.

# Библиография

1 Десять бесплатных систем управления [Электронный ресурс] – Режим доступа: https://habr.com/ru/companies/yougile/articles/537920/;

2 Двадцать российских сервисов для автоматизации бизнес-процессов [Электронный ресурс] – Режим доступа: https://vc.ru/services/410492-20-rossiyskih-servisov-dlya-avtomatizacii-biznes-processov;

3 Двадцать пять лучших программ для управления рабочими процессами для повышения эффективности в 2023 году [Электронный ресурс] – Режим доступа: https://xmldatafeed.com/25-luchshih-programm-dlya-upravleniya-rabochimi-proczessami-dlya-povysheniya-effektivnosti-v-2023-godu/;

4 Безопасность и защита данных пользователя приложения [Электронный ресурс] – Режим доступа: https://blog.advertmobile.net/bezopasnost-i-zashhita-dannyx/;

5 Многодокументный интерфейс [Электронный ресурс] – Режим доступа: https://goo.su/KJHh53p/;

6 Идентификация и аутентификация субъектов [Электронный ресурс] – Режим доступа: https://drive.google.com/drive/folders/1Qt9HLf9hZJ5a5H-TSz0wy-Y4qFwPu-kj;

7 Методы разграничения доступа [Электронный ресурс] – Режим доступа: https://spravochnick.ru/informacionnaya\_bezopasnost/sposoby\_i\_sredstva\_narusheniya\_konfidencialnosti\_informacii/metody\_razgranicheniya\_dostupa/.

# Приложение А

(справочное)

Листинг программы

Ниже представлен листинг программы:

using MySql.Data.MySqlClient;

using System;

using System.Windows.Forms;

namespace kyrs

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

string connectionString = "Server=localhost;Database=autorepair;UserID=root;Password=root;";

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

Console.WriteLine("Подключение к базе данных успешно!");

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка подключения к базе данных: {ex.Message}");

}

}

// Остальной код, который вы хотите выполнить при загрузке формы

}

}

}

using System;

using System.Drawing;

using System.Windows.Forms;

namespace kyrs

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

}

private void LoginForm\_Load(object sender, EventArgs e)

{

}

private void btnLogin\_Click(object sender, EventArgs e)

{

// Проверка имени пользователя и пароля

if (txtUsername.Text == "admin" && txtPassword.Text == "password")

{

// Если успешно, открываем форму администратора

AdminForm adminForm = new AdminForm();

adminForm.Show();

// Скрываем форму авторизации

this.Hide();

}

else

{

MessageBox.Show("Неверные имя пользователя или пароль.", "Ошибка входа", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void chkShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !chkShowPassword.Checked;

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void btnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

using System;

using System.Windows.Forms;

namespace kyrs

{

public partial class AdminForm : Form

{

public AdminForm()

{

InitializeComponent();

btnLogout.Click += new EventHandler(btnLogout\_Click\_1); // Подписываемся на событие нажатия кнопки выхода

}

private void btnServices\_Click(object sender, EventArgs e)

{

ServicesForm servicesForm = new ServicesForm();

servicesForm.Show();

}

private void btnEmployees\_Click(object sender, EventArgs e)

{

EmployeesForm employeesForm = new EmployeesForm();

employeesForm.Show();

}

private void btnClients\_Click(object sender, EventArgs e)

{

ClientsForm clientsForm = new ClientsForm();

clientsForm.Show();

}

private void btnLogout\_Click\_1(object sender, EventArgs e)

{

Application.Exit();

}

}

}

using MySql.Data.MySqlClient;

using System;

using System.Windows.Forms;

namespace kyrs

{

public partial class EmployeesForm : Form

{

private string connectionString = "Server=localhost;Database=autorepair;UserID=root;Password=root;";

public EmployeesForm()

{

InitializeComponent();

LoadEmployees();

}

private void LoadEmployees()

{

listBoxEmployees.Items.Clear();

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "SELECT \* FROM Employees";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

listBoxEmployees.Items.Add(reader["name"].ToString());

// Добавьте другие поля, если необходимо

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке информации о сотрудниках: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

listBoxEmployees.SelectedIndexChanged += new EventHandler(ListBoxEmployees\_SelectedIndexChanged);

}

}

private void ListBoxEmployees\_SelectedIndexChanged(object sender, EventArgs e)

{

ClearEmployeeInfo();

if (listBoxEmployees.SelectedIndex != -1)

{

int selectedEmployeeID = 0;

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = $"SELECT employee\_id FROM Employees WHERE name = '{listBoxEmployees.SelectedItem.ToString()}'";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

selectedEmployeeID = Convert.ToInt32(reader["employee\_id"]);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении ID выбранного сотрудника: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = $"SELECT name, position, salary FROM Employees WHERE employee\_id = {selectedEmployeeID}";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

lblEmployeeName.Text = reader["name"].ToString();

lblEmployeePosition.Text = reader["position"].ToString();

lblEmployeeSalary.Text = reader["salary"].ToString();

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении информации о выбранном сотруднике: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void ClearEmployeeInfo()

{

lblEmployeeName.Text = string.Empty;

lblEmployeePosition.Text = string.Empty;

}

private void btnAddEmployee\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(txtNewEmployeeName.Text) && !string.IsNullOrEmpty(txtNewEmployeePosition.Text) && !string.IsNullOrEmpty(txtNewEmployeeSalary.Text))

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "INSERT INTO Employees (name, position, salary) VALUES (@name, @position, @salary)";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@name", txtNewEmployeeName.Text);

command.Parameters.AddWithValue("@position", txtNewEmployeePosition.Text);

command.Parameters.AddWithValue("@salary", Convert.ToDecimal(txtNewEmployeeSalary.Text));

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении сотрудника: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadEmployees(); // Обновить список после добавления

}

else

{

MessageBox.Show("Заполните все поля для добавления сотрудника.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btnUpdateEmployee\_Click(object sender, EventArgs e)

{

if (listBoxEmployees.SelectedIndex != -1 && !string.IsNullOrEmpty(txtNewEmployeeName.Text) && !string.IsNullOrEmpty(txtNewEmployeePosition.Text) && !string.IsNullOrEmpty(txtNewEmployeeSalary.Text))

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "UPDATE Employees SET name = @name, position = @position, salary = @salary WHERE name = @selectedName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@name", txtNewEmployeeName.Text);

command.Parameters.AddWithValue("@position", txtNewEmployeePosition.Text);

command.Parameters.AddWithValue("@salary", Convert.ToDecimal(txtNewEmployeeSalary.Text));

command.Parameters.AddWithValue("@selectedName", listBoxEmployees.SelectedItem.ToString());

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении сотрудника: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadEmployees(); // Обновить список после обновления

}

else

{

MessageBox.Show("Выберите сотрудника и заполните все поля для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btnDeleteEmployee\_Click(object sender, EventArgs e)

{

if (listBoxEmployees.SelectedIndex != -1)

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "DELETE FROM Employees WHERE name = @selectedName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@selectedName", listBoxEmployees.SelectedItem.ToString());

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении сотрудника: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadEmployees(); // Обновить список после удаления

}

else

{

MessageBox.Show("Выберите сотрудника для удаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

}

using MySql.Data.MySqlClient;

using System;

using System.Windows.Forms;

namespace kyrs

{

public partial class ServicesForm : Form

{

string connectionString = "Server=localhost;Database=autorepair;UserID=root;Password=root;";

public ServicesForm()

{

InitializeComponent();

LoadServices();

}

private void ServicesForm\_Load(object sender, EventArgs e)

{

}

private void LoadServices()

{

// Подключение к базе данных и получение информации о услугах

listBoxServices.Items.Clear(); // Очистите список перед загрузкой обновленных данных

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

// Выполнение SQL-запроса для получения информации о услугах

string query = "SELECT \* FROM Services";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

// Вывод информации на форму

while (reader.Read())

{

listBoxServices.Items.Add(reader["name"].ToString());

// Добавьте другие поля, если необходимо

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке информации о услугах: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

listBoxServices.SelectedIndexChanged += new EventHandler(ListBoxServices\_SelectedIndexChanged);

}

}

private void ListBoxServices\_SelectedIndexChanged(object sender, EventArgs e)

{

// Очистка предыдущей информации

ClearServiceInfo();

// Проверка, выбрана ли какая-то услуга

if (listBoxServices.SelectedIndex != -1)

{

// Получение ID выбранной услуги

int selectedServiceID = 0; // Инициализация переменной

// Получение значения service\_id из базы данных

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

// Выполнение SQL-запроса для получения информации о выбранной услуге

string query = $"SELECT service\_id FROM Services WHERE name = '{listBoxServices.SelectedItem.ToString()}'";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

// Проверка, найдена ли услуга в базе данных

if (reader.Read())

{

selectedServiceID = Convert.ToInt32(reader["service\_id"]);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении ID выбранной услуги: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Получение информации о выбранной услуге

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

// Выполнение SQL-запроса для получения информации о выбранной услуге

string query = $"SELECT name, description, cost FROM Services WHERE service\_id = {selectedServiceID}";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

// Отображение информации о выбранной услуге

if (reader.Read())

{

lblServiceName.Text = reader["name"].ToString();

lblServiceDescription.Text = reader["description"].ToString();

lblServicePrice.Text = $"Цена: {reader["cost"].ToString()} руб.";

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении информации о выбранной услуге: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

// Метод для очистки информации о выбранной услуге

private void ClearServiceInfo()

{

lblServiceName.Text = string.Empty;

lblServiceDescription.Text = string.Empty;

lblServicePrice.Text = string.Empty;

}

private void btnAddService\_Click(object sender, EventArgs e)

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "INSERT INTO Services (name, description, cost) VALUES (@name, @description, @cost)";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@name", txtNewServiceName.Text);

command.Parameters.AddWithValue("@description", txtNewServiceDescription.Text);

command.Parameters.AddWithValue("@cost", Convert.ToDecimal(txtNewServicePrice.Text));

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении услуги: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadServices(); // Обновить список после добавления

}

private void btnUpdateService\_Click(object sender, EventArgs e)

{

if (listBoxServices.SelectedIndex != -1)

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "UPDATE Services SET name = @name, description = @description, cost = @cost WHERE name = @selectedName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@name", txtNewServiceName.Text);

command.Parameters.AddWithValue("@description", txtNewServiceDescription.Text);

command.Parameters.AddWithValue("@cost", Convert.ToDecimal(txtNewServicePrice.Text));

command.Parameters.AddWithValue("@selectedName", listBoxServices.SelectedItem.ToString());

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении услуги: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadServices(); // Обновить список после обновления

}

else

{

MessageBox.Show("Выберите услугу для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btnDeleteService\_Click(object sender, EventArgs e)

{

if (listBoxServices.SelectedIndex != -1)

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "DELETE FROM Services WHERE name = @selectedName";

MySqlCommand command = new MySqlCommand(query, connection);

command.Parameters.AddWithValue("@selectedName", listBoxServices.SelectedItem.ToString());

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении услуги: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadServices(); // Обновить список после удаления

}

else

{

MessageBox.Show("Выберите услугу для удаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

}

using MySql.Data.MySqlClient;

using System;

using System.Windows.Forms;

namespace kyrs

{

public partial class ClientsForm : Form

{

private string connectionString = "Server=localhost;Database=autorepair;UserID=root;Password=root;";

public ClientsForm()

{

InitializeComponent();

LoadClients();

}

private void ClientsForm\_Load(object sender, EventArgs e)

{

}

private void LoadClients()

{

listBoxClients.Items.Clear();

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = "SELECT \* FROM newcustomers";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

listBoxClients.Items.Add(reader["name"].ToString());

// Добавьте другие поля, если необходимо

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке информации о клиентах: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

listBoxClients.SelectedIndexChanged += new EventHandler(ListBoxClients\_SelectedIndexChanged);

}

}

private void ListBoxClients\_SelectedIndexChanged(object sender, EventArgs e)

{

ClearClientInfo();

if (listBoxClients.SelectedIndex != -1)

{

int selectedClientID = 0;

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = $"SELECT customer\_id FROM newcustomers WHERE name = '{listBoxClients.SelectedItem.ToString()}'";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

selectedClientID = Convert.ToInt32(reader["customer\_id"]);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении ID выбранного клиента: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

string query = $"SELECT name, email, contact\_number FROM newcustomers WHERE customer\_id = {selectedClientID}";

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

lblClientName.Text = reader["name"].ToString();

lblClientEmail.Text = reader["email"].ToString();

lblClientPhone.Text = reader["contact\_number"].ToString();

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении информации о выбранном клиенте: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void ClearClientInfo()

{

lblClientName.Text = string.Empty;

lblClientEmail.Text = string.Empty;

lblClientPhone.Text = string.Empty;

}

private void btnAddClient\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(txtNewClientName.Text) && !string.IsNullOrEmpty(txtNewClientContact.Text) && !string.IsNullOrEmpty(txtNewClientEmail.Text))

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

// Проверяем, существует ли клиент с таким именем

string checkQuery = "SELECT COUNT(\*) FROM newcustomers WHERE name = @name";

MySqlCommand checkCommand = new MySqlCommand(checkQuery, connection);

checkCommand.Parameters.AddWithValue("@name", txtNewClientName.Text);

int existingClients = Convert.ToInt32(checkCommand.ExecuteScalar());

if (existingClients == 0)

{

// Если клиента с таким именем нет, выполняем добавление

string insertQuery = "INSERT INTO newcustomers (name, contact\_number, email) VALUES (@name, @contact\_number, @email)";

MySqlCommand insertCommand = new MySqlCommand(insertQuery, connection);

insertCommand.Parameters.AddWithValue("@name", txtNewClientName.Text);

insertCommand.Parameters.AddWithValue("@contact\_number", txtNewClientContact.Text);

insertCommand.Parameters.AddWithValue("@email", txtNewClientEmail.Text);

insertCommand.ExecuteNonQuery();

}

else

{

MessageBox.Show("Клиент с таким именем уже существует.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении клиента: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadClients(); // Обновить список после добавления

}

else

{

MessageBox.Show("Заполните все поля для добавления клиента.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btnUpdateClient\_Click(object sender, EventArgs e)

{

if (listBoxClients.SelectedIndex != -1 && !string.IsNullOrEmpty(txtNewClientName.Text) && !string.IsNullOrEmpty(txtNewClientContact.Text) && !string.IsNullOrEmpty(txtNewClientEmail.Text))

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

// Проверяем, существует ли клиент с таким именем, исключая выбранный клиент

string checkQuery = "SELECT COUNT(\*) FROM newcustomers WHERE name = @name AND name != @selectedName";

MySqlCommand checkCommand = new MySqlCommand(checkQuery, connection);

checkCommand.Parameters.AddWithValue("@name", txtNewClientName.Text);

checkCommand.Parameters.AddWithValue("@selectedName", listBoxClients.SelectedItem.ToString());

int existingClients = Convert.ToInt32(checkCommand.ExecuteScalar());

if (existingClients == 0)

{

// Если клиента с таким именем (не считая выбранного) нет, выполняем обновление

string updateQuery = "UPDATE newcustomers SET name = @name, contact\_number = @contact\_number, email = @email WHERE name = @selectedName";

MySqlCommand updateCommand = new MySqlCommand(updateQuery, connection);

updateCommand.Parameters.AddWithValue("@name", txtNewClientName.Text);

updateCommand.Parameters.AddWithValue("@contact\_number", txtNewClientContact.Text);

updateCommand.Parameters.AddWithValue("@email", txtNewClientEmail.Text);

updateCommand.Parameters.AddWithValue("@selectedName", listBoxClients.SelectedItem.ToString());

updateCommand.ExecuteNonQuery();

}

else

{

MessageBox.Show("Клиент с таким именем уже существует.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении клиента: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadClients(); // Обновить список после обновления

}

else

{

MessageBox.Show("Выберите клиента и заполните все поля для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void btnDeleteClient\_Click(object sender, EventArgs e)

{

if (listBoxClients.SelectedIndex != -1)

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

try

{

connection.Open();

// Получаем имя выбранного клиента

string selectedClientName = listBoxClients.SelectedItem.ToString();

// Проверяем, существует ли клиент с таким именем

string checkQuery = "SELECT COUNT(\*) FROM newcustomers WHERE name = @name";

MySqlCommand checkCommand = new MySqlCommand(checkQuery, connection);

checkCommand.Parameters.AddWithValue("@name", selectedClientName);

int existingClients = Convert.ToInt32(checkCommand.ExecuteScalar());

if (existingClients > 0)

{

// Если клиент с таким именем существует, выполняем удаление

string deleteQuery = "DELETE FROM newcustomers WHERE name = @name";

MySqlCommand deleteCommand = new MySqlCommand(deleteQuery, connection);

deleteCommand.Parameters.AddWithValue("@name", selectedClientName);

deleteCommand.ExecuteNonQuery();

}

else

{

MessageBox.Show("Клиент с таким именем не существует.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении клиента: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

LoadClients(); // Обновить список после удаления

}

else

{

MessageBox.Show("Выберите клиента для удаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void txtNewClientContact\_TextChanged(object sender, EventArgs e)

{

}

}

}