Государственное бюджетное профессиональное образовательное учреждение Новосибирской области

«Новосибирский колледж электроники и вычислительной техники»

# Учебная практика

Внедрение и поддержка КС

## Тема: Создание программного продукта «Продажа автомобилей»

Выполнили студенты группы 9ИС-380:

Бычков Андрей

Ерыгин Максим

Преподаватель Усольцев А.А

Новосибирск – 2023

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение | 3 |
| 1 Теоретический обзор | 5 |
| 1.1 Обзор Технологий WPF и C# | 5 |
| 1.2 Основы проектирования базы данных | 6 |
| 2 Проектирование приложения | 7 |
| 3 Реализация приложения | 17 |
| 3.1 Описание кода различных экранов приложения | 17 |
| 3.2 Работа с базой данных | 29 |
| 3.3 Тестирование и отладка | 29 |
| Заключение | 31 |
| Список использованной литературы | 33 |

Введение

Автомобильная индустрия играет значительную роль в мировой экономике. Автомобили представляют собой важный и распространенный вид транспорта, обеспечивающий мобильность и удобство в повседневной жизни. Автомобильные дилерские центры и компании, занимающиеся продажей автомобилей, сталкиваются с управлением огромными объемами информации о клиентах, автомобилях, продавцах и продажах. В этом контексте становится критически важным иметь эффективную систему управления данными.

Актуальность данного проекта определяется следующими факторами:

- автоматизация процессов: автомобильные дилерские центры имеют сложные и многокомпонентные системы управления данными. Автоматизированная система управления продажами и клиентами упрощает процессы и снижает риск ошибок;

- улучшение обслуживание клиентов: более точная и актуальная информация о клиентах и их потребностях позволяет дилерам предоставлять более высокое качество обслуживания.

- эффективное управление складом: приложение позволит дилерам легко отслеживать доступность автомобилей и управлять своим автопарком.

- улучшение уровня довольства клиентов: удобство и прозрачность взаимодействия с дилером способствует увеличению уровня довольства клиентов и повышению лояльности.

Цель данной работы - разработать и исследовать приложение для управления продажей автомобилей. Это приложение предоставит удобный и эффективный инструмент для автомобильных дилеров, который поможет им легко управлять данными о клиентах, автомобилях, продавцах и продажах. Приложение будет разработано с использованием технологий WPF (Windows Presentation Foundation) и C#.

Задачи исследования:

1. Разработать графический интерфейс пользователя (GUI) для приложения на платформе WPF.
2. Создать базу данных для хранения информации о клиентах, продавцах, автомобилях и продажах.
3. Реализовать функциональность для регистрации и аутентификации пользователей в системе.
4. Разработать функциональность для добавления, редактирования и удаления данных о клиентах, продавцах и автомобилях.
5. Предоставить возможность регистрации продаж и отслеживания данных о проданных автомобилях.
6. Реализовать отчеты и статистику по продажам и доступным автомобилям.
7. Провести тестирование и оптимизацию приложения.

В данной работе будет подробно рассмотрен процесс проектирования, реализации и исследования приложения для управления продажей автомобилей.

1 Теоретический обзор

1.1 Обзор технологий WPF и C#:

1. Windows Presentation Foundation (WPF) - это технология, разработанная Microsoft, предназначенная для создания приложений с графическим интерфейсом пользователя (GUI). Она предоставляет богатые возможности для разработки приложений с современным и интерактивным дизайном.

Основные характеристики WPF включают:

- векторную графику: WPF использует векторную графику, что позволяет создавать гибкий и масштабируемый дизайн интерфейса;

- стили и шаблоны: WPF поддерживает стили и шаблоны, что облегчает создание и применение дизайнерских элементов в приложении;

**-** привязку данных: WPF обеспечивает мощные средства для привязки данных к элементам интерфейса, что делает работу с данными более эффективной;

- анимацию и трехмерную графику: WPF поддерживает анимацию и трехмерную графику, что позволяет создавать более интерактивные и привлекательные приложения.

2. Язык программирования C# - это объектно-ориентированный язык программирования, разработанный Microsoft. C# широко используется для разработки приложений под Windows, включая WPF-приложения. Он предоставляет мощные инструменты для разработки, отладки и тестирования приложений.

Основные особенности C#:

**-** синтаксис: C# предлагает чистый и удобочитаемый синтаксис, что делает его привлекательным для разработчиков;

**-** объектно-ориентированность: C# поддерживает объектно-ориентированное программирование, что упрощает организацию и управление кодом;

**-** управляемый код: C# создает управляемый код, что повышает безопасность и стабильность приложений;

- большая стандартная библиотека: C# имеет обширную стандартную библиотеку классов, которая включает множество полезных классов и методов для решения различных задач.

1.2 Основы проектирования базы данных.

Проектирование базы данных - это важный этап разработки приложений, где определяется структура и организация хранения данных. В данном проекте используется база данных CarRentalDb, включающая следующие таблицы:

CarTbl: Таблица для хранения информации о автомобилях. Включает в себя бренд, модель, государственный номер, цену и статус доступности.

CustomerTbl: Таблица для хранения информации о клиентах, включая имя, адрес и номер телефона.

FreeTbl: Таблица, аналогичная CarTbl, но содержит данные о свободных автомобилях.

SellersTbl: Таблица для хранения информации о продавцах, включая имя, адрес и номер телефона.

SellTbl: Таблица для регистрации продаж автомобилей, включая номер автомобиля, имя клиента, имя продавца и дату продажи.

UserTbl: Таблица для хранения информации о пользователях системы, включая логин и пароль.

При проектировании базы данных учитываются следующие аспекты:

- нормализация данных: база данных должна быть нормализована, чтобы избежать дублирования данных и обеспечить целостность информации;

- индексы: использование индексов ускоряет выполнение запросов к базе данных;

**-** связи между таблицами: определение связей между таблицами позволяет объединять данные из разных таблиц при необходимости;

**-** безопасность данных: обеспечение безопасности данных с использованием правил доступа и проверки подлинности.

Создание базы данных является критически важным для эффективной работы приложения, так как оно влияет на быстродействие, целостность и надежность хранения данных.

2 Проектирование приложения

При разработке приложения студенты группы 9ИС-380 Бычков Андрей и Ерыгин Максим осуществили следующие действия:

1. Разработка структуры базы данных:

Для продажи автомобилей на платформе WPF с использованием C# необходимо определить структуру базы данных, которая будет хранить информацию о машинах, клиентах, продажах, продавцах и пользователях системы. Структура базы данных CarRentalDb уже определена и перечислена в предыдущем пункте, она была разработана с учетом потребностей приложения и обеспечивает удобное хранение и взаимодействие с данными.

1. Проектирование пользовательского интерфейса (GUI):

Пользовательский интерфейс (GUI) приложения разрабатывается с использованием технологии WPF. В интерфейсе предусмотрены следующие экраны (рис.1-9):



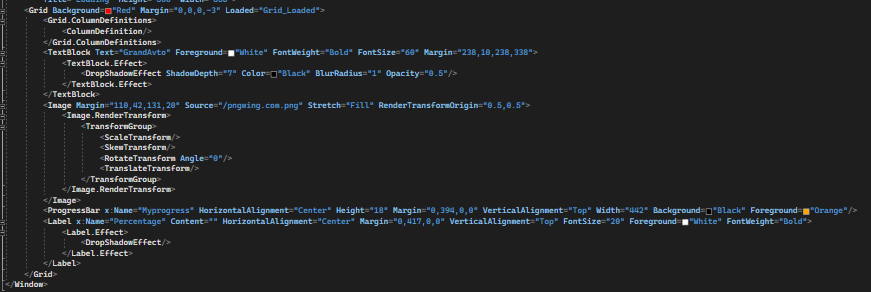
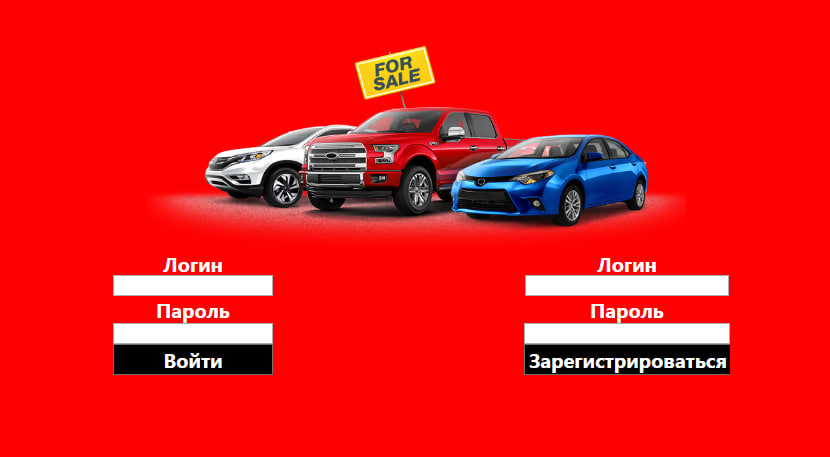


Рисунок 1 - Загрузочный экран (Loading)

Этот экран отображается при запуске приложения и предоставляет пользователю информацию о процессе загрузки. После загрузки приложения происходит переход на экран авторизации.



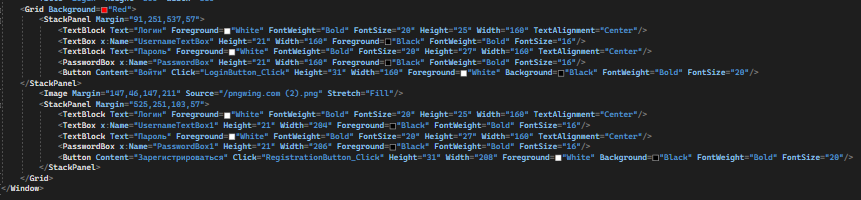


Рисунок 2 - Экран авторизации (Login)

Пользователь вводит свой логин и пароль. Эти данные сверяются с информацией из таблицы UserTbl базы данных. В случае успешной аутентификации пользователь получает доступ к главному экрану приложения.



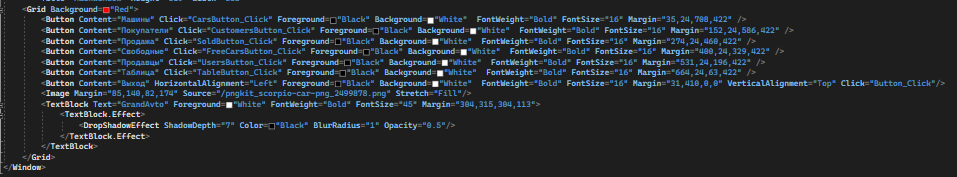


Рисунок 3 - Главный экран (MainWindow)

Этот экран предоставляет пользователю доступ к основным функциональным модулям приложения, таким как управление машинами, клиентами, продажами, продавцами, просмотр списка свободных машин и просмотр общей статистики продаж.



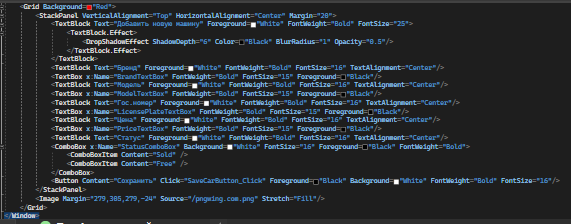


Рисунок 4 - Экран управления машинами (Cars)

Здесь пользователь может добавлять новые автомобили, обновлять их информацию, указывая бренд, модель, государственный номер, цену и статус доступности.



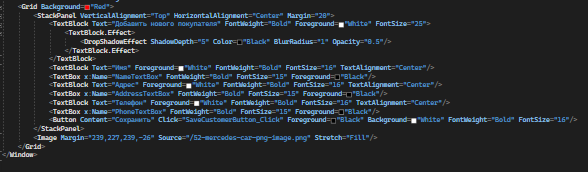


Рисунок 5 - Экран управления клиентами (Customers)

На этом экране пользователь может добавлять, обновлять информацию о клиентах, включая имя, адрес и номер телефона.



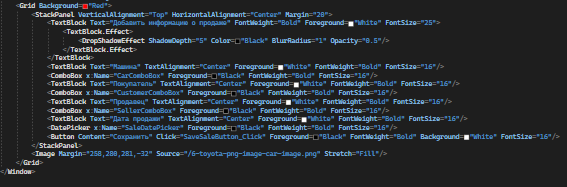


Рисунок 6 - Экран управления продажами (Prodani)

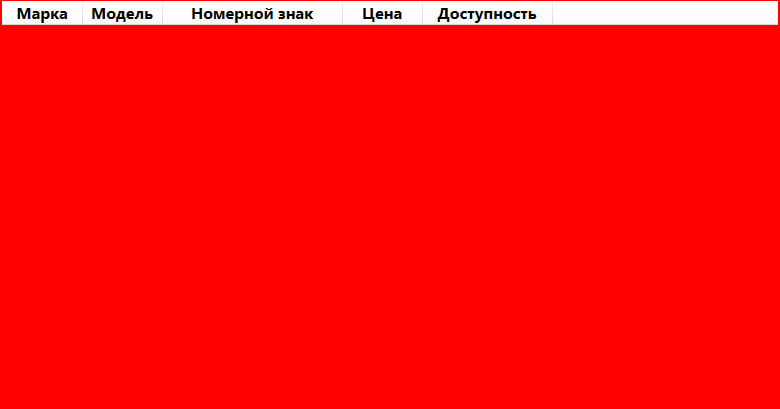
Здесь пользователь может регистрировать, обновлять продажи, выбирая автомобиль, клиента и продавца, а также указывая дату продажи.





Рисунок 7 - Экран управления продавцами (Sellers)

На этом экране пользователь добавляет информацию о продавцах, включая имя, адрес и номер телефона.



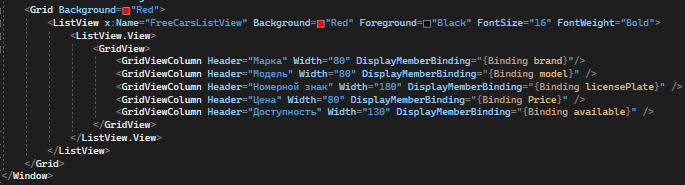
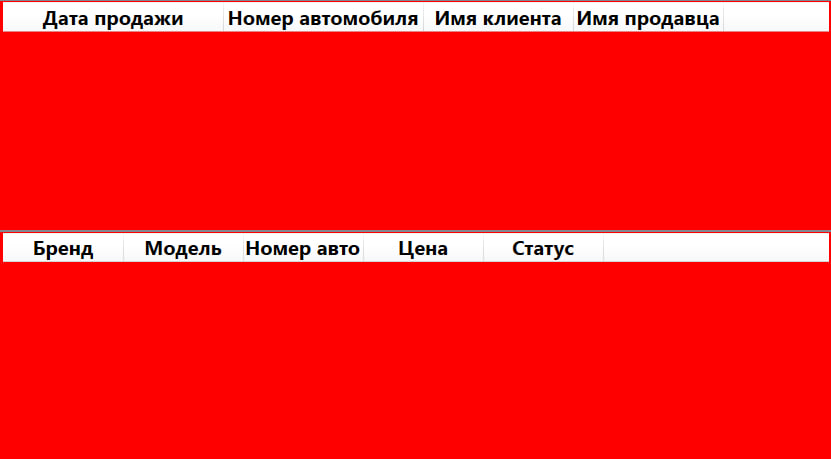


Рисунок 8 - Экран просмотра свободных машин (Free)

Позволяет просматривать список свободных автомобилей из таблицы FreeTbl.



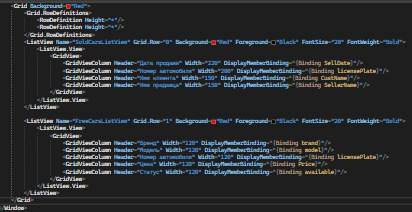


Рисунок 9 - Экран общей статистики продаж (Tablica)

Пользователь может просматривать общую статистику проданных автомобилей, включая дату продажи, номер автомобиля, имя клиента, имя продавца, бренд, модель, номер автомобиля, цену и статус.

Каждый из этих модулей обеспечивает удобное взаимодействие с данными в базе данных и позволяет пользователю выполнять необходимые операции в приложении.

3 Реализация приложения

3.1. Описание кода различных экранов приложения.

В конструкторе класса Loading, инициализируется объект timer типа DispatcherTimer, который будет использоваться для обновления индикатора прогресса (рис.10).

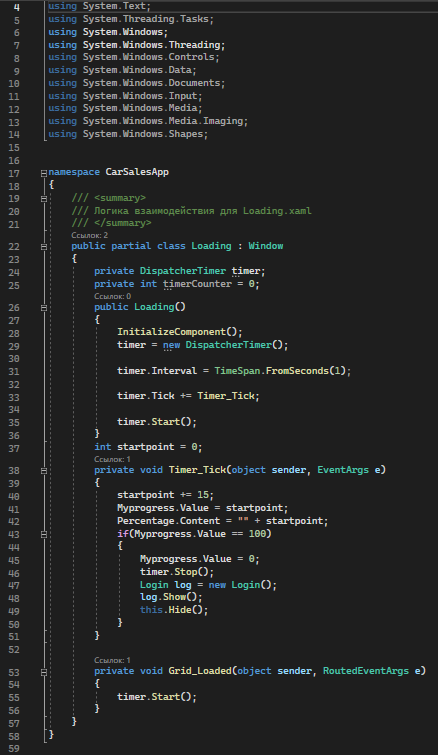
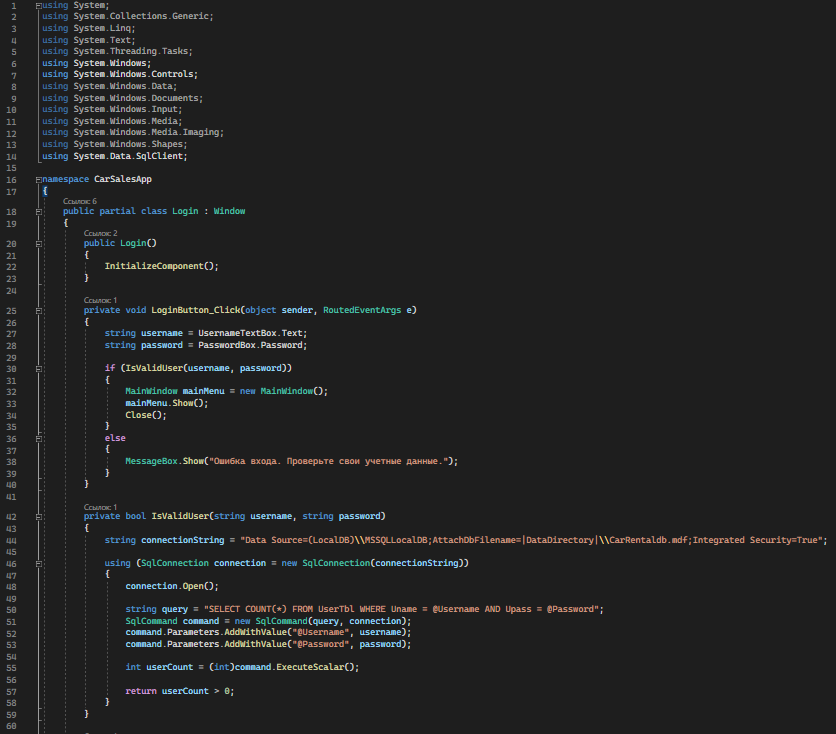


Рисунок 10 - Описание кода различных экранов приложения (Loading)

В методе Timer\_Tick, каждую секунду (timer.Interval = TimeSpan.FromSeconds(1)) значение startpoint увеличивается на 15, и это значение устанавливается в качестве значения индикатора прогресса Myprogress. Также это значение отображается в элементе Percentage.

Когда индикатор прогресса достигает значения 100, таймер останавливается, и открывается новое окно Login. Текущее окно Loading скрывается (не закрывается, так как это может вызвать проблемы с переходом обратно).

Этот код реализует простой механизм загрузки перед переходом к основному окну приложения. Вам также следует убедиться, что в вашем проекте есть окно с именем Login, которое открывается после окончания загрузки (рис.11).



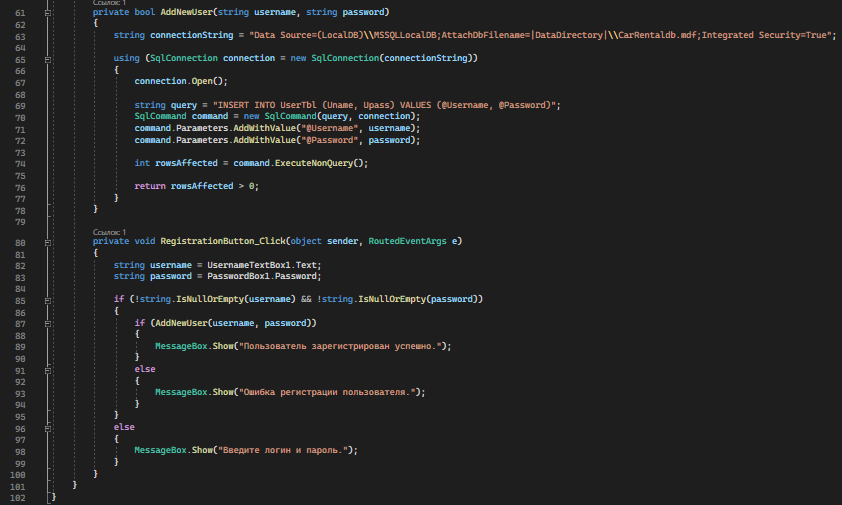


Рисунок 11 – Окно Login

В конструкторе класса Login, происходит инициализация окна.

Метод LoginButton\_Click вызывается при нажатии кнопки «Вход» и получает введенные пользователем логин и пароль. Затем он вызывает метод IsValidUser, чтобы проверить, существует ли пользователь с такими учетными данными. Если пользователь существует, открывается главное окно MainWindow, и текущее окно Login закрывается.

Метод IsValidUser проверяет, есть ли пользователь с указанным логином и паролем в базе данных. Для этого он использует соединение с базой данных через ADO.NET и SQL-запрос.

Метод AddNewUser вызывается при нажатии кнопки "Регистрация". Он пытается добавить нового пользователя с указанным логином и паролем в базу данных. Если операция успешна, отображается сообщение об успешной регистрации, иначе выводится сообщение об ошибке.

Валидация перед добавлением нового пользователя важна для обеспечения безопасности и предотвращения дублирования пользователей.

Этот код обеспечивает функциональность входа и регистрации пользователей в приложении для продажи автомобилей. Пользователю нужно также удостовериться, что у него есть таблица UserTbl в базе данных, которая содержит информацию о нем.

В конструкторе класса MainWindow, происходит инициализация главного окна приложения (рис.12).

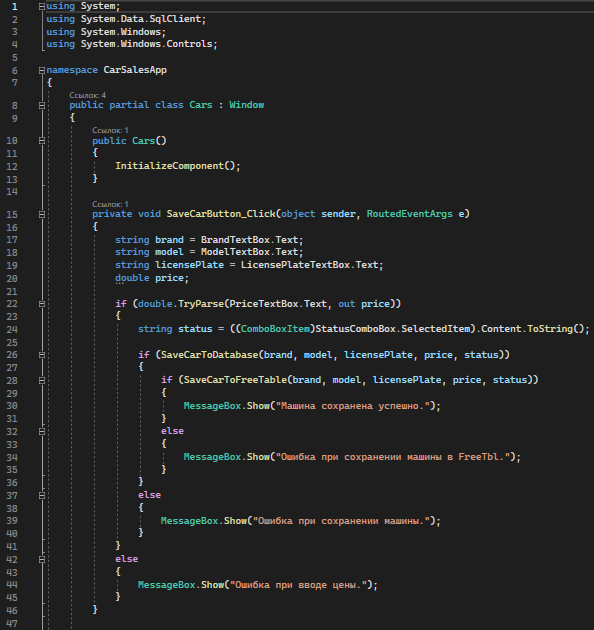


Рисунок 12 – Окно MainWindow

В методах, связанных с кнопками (например, CarsButton\_Click, CustomersButton\_Click, и так далее), создаются новые окна для каждого раздела вашего приложения (например, «Машины», «Покупатели», и так далее), и эти окна открываются с помощью ShowDialog(). Это позволяет пользователям взаимодействовать с этими окнами, а главное окно остается открытым и доступным для навигации.

Вы также имеете кнопку «Выйти» (которая вызывает окно Login), чтобы пользователь мог выйти из текущей сессии.

Этот код управляет навигацией в главном окне вашего приложения, позволяя пользователям переходить между различными разделами, такими как «Машины», «Покупатели», «Проданные», «Свободные», «Продавцы» и «Таблица», а также выходить из приложения.



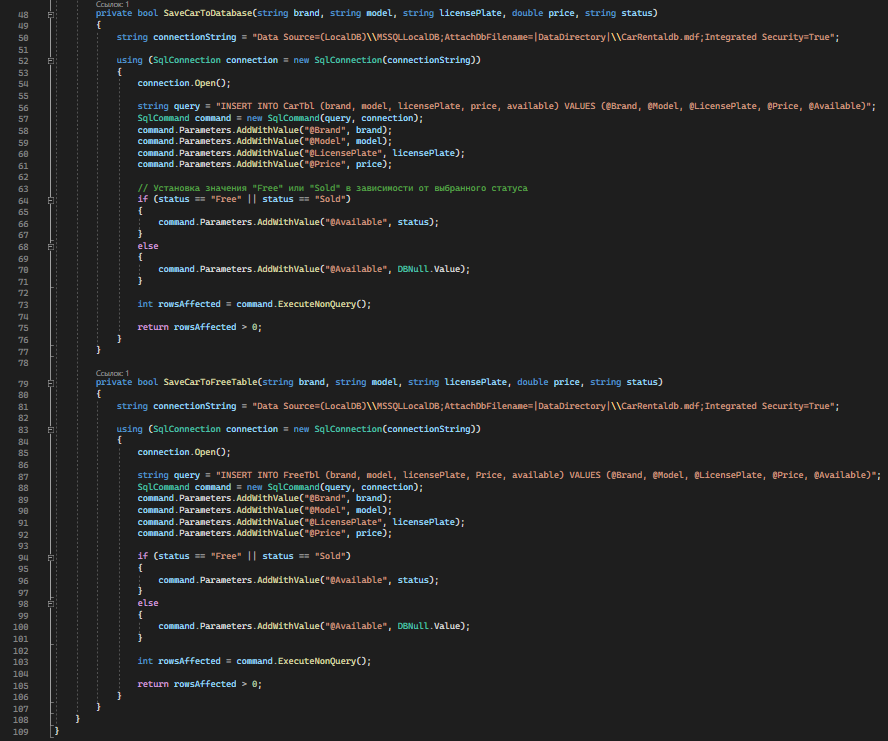


Рисунок 13 – Окно Cars

В конструкторе класса Cars, происходит инициализация окна (рис.13).

Метод SaveCarButton\_Click вызывается при нажатии кнопки «Сохранить». Он получает данные о машине (бренд, модель, гос. номер, цену и статус) из визуальных элементов на форме.

Затем происходит проверка цены на корректность. Если цена успешно парсится в double, код продолжает выполнение. В противном случае пользователю выводится сообщение об ошибке.

Далее вызываются два метода: SaveCarToDatabase и SaveCarToFreeTable. Первый метод сохраняет информацию о машине в таблице CarTbl, а второй метод сохраняет информацию о машине в таблице FreeTbl.

В обоих методах создается подключение к базе данных, и выполняется SQL-запрос для вставки данных. Строка подключения определена в переменной connectionString.

В зависимости от выбранного статуса (например, «Free» или «Sold»), значение Available устанавливается в соответствующее значение или null (при помощи DBNull.Value).

Если операция вставки данных в базу данных успешна (возвращается значение больше 0), выводится сообщение об успешном сохранении. В противном случае выводится сообщение об ошибке.

Этот код обеспечивает сохранение данных о машинах в вашей базе данных и в таблице FreeTbl, что позволяет отслеживать доступность машин для продажи.

В конструкторе класса Customers, происходит инициализация окна (рис.14).

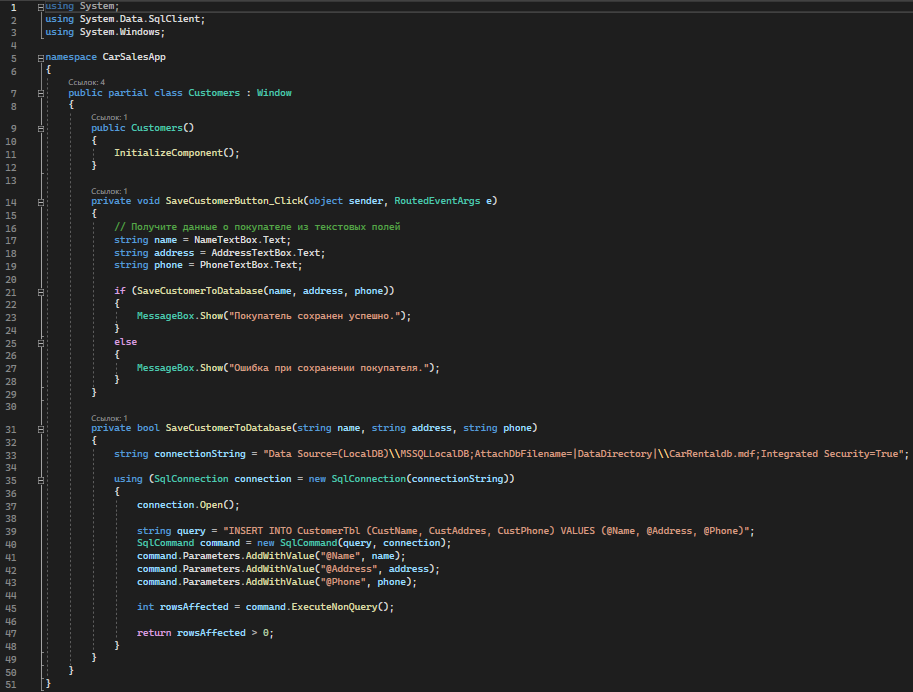


Рисунок 14 – Окно Customers

Метод SaveCustomerButton\_Click вызывается при нажатии кнопки «Сохранить». Он получает данные о покупателе (имя, адрес, номер телефона) из текстовых полей на форме.

Далее вызывается метод SaveCustomerToDatabase, который сохраняет информацию о покупателе в таблице CustomerTbl.

Создается подключение к базе данных с использованием строки подключения connectionString.

Выполняется SQL-запрос для вставки данных в таблицу CustomerTbl. Параметры @Name, @Address и @Phone заполняются данными из текстовых полей.

Если операция вставки данных в базу данных успешна (возвращается значение больше 0), выводится сообщение об успешном сохранении. В противном случае выводится сообщение об ошибке.

Этот код обеспечивает сохранение данных о покупателях в вашей базе данных, что позволяет вам отслеживать информацию о клиентах, которые приобрели автомобили.

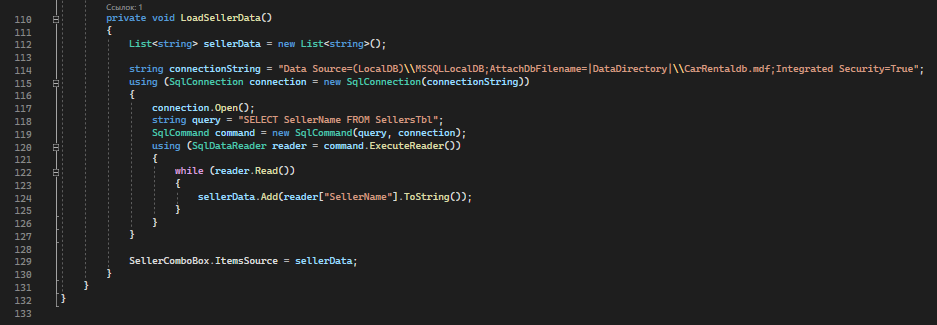
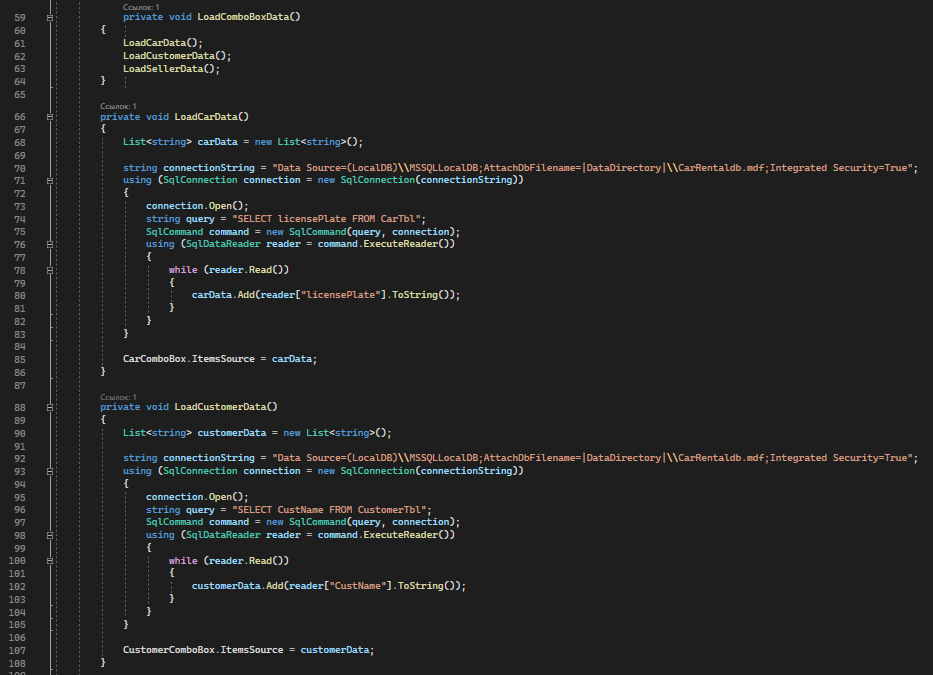
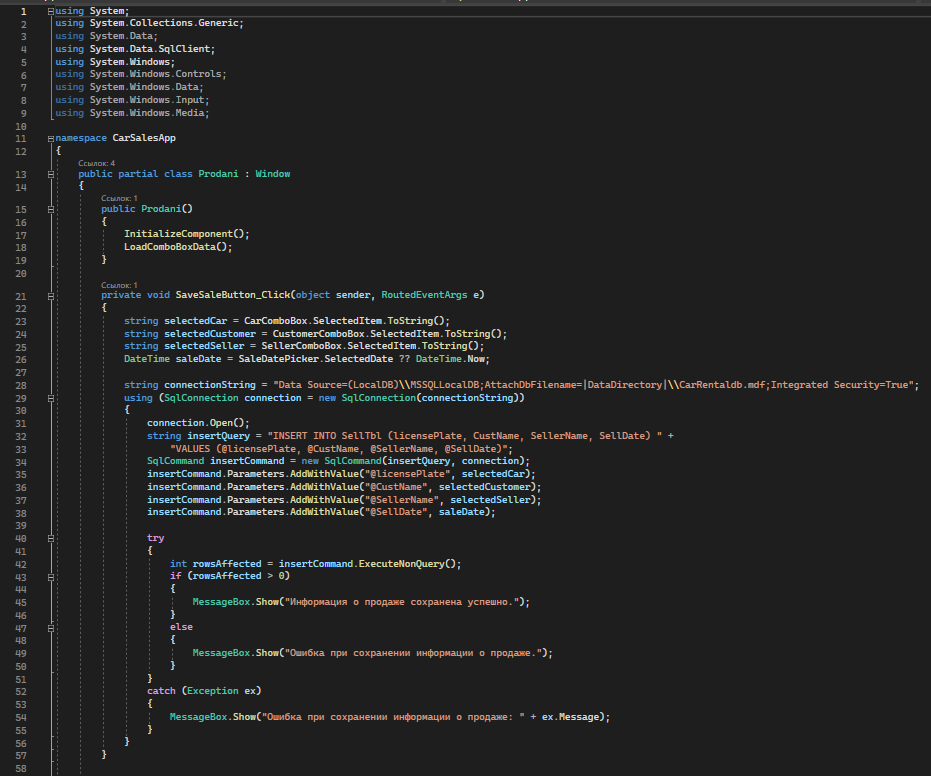


Рисунок 15 – Окно Prodani

В конструкторе класса Prodani, происходит инициализация окна и вызывается метод LoadComboBoxData, который загружает данные для выпадающих списков с данными о машинах, покупателях и продавцах (рис.15).

Метод SaveSaleButton\_Click вызывается при нажатии кнопки «Сохранить». Он получает выбранные данные о машине, покупателе и продавце, а также дату продажи. Затем он выполняет SQL-запрос для вставки информации о продаже в таблицу SellTbl в базе данных.

Методы LoadCarData, LoadCustomerData и LoadSellerData загружают данные о машинах, покупателях и продавцах из соответствующих таблиц в базе данных и заполняют выпадающие списки на форме данными.

Если операция вставки данных о продаже в базу данных успешна (возвращается значение больше 0), выводится сообщение об успешном сохранении. В случае возникновения ошибки, выводится сообщение об ошибке, включая сообщение об исключении, если таковое имеется.

Этот код обеспечивает ввод данных о продажах и сохранение информации о них в вашей базе данных.

В конструкторе класса Sellers, происходит инициализация окна (рис.16).

Метод SaveSellerButton\_Click вызывается при нажатии кнопки «Сохранить». Он получает данные о продавце (имя, адрес, номер телефона) из текстовых полей на форме.

Далее вызывается метод SaveSellerToDatabase, который сохраняет информацию о продавце в таблице SellersTbl.

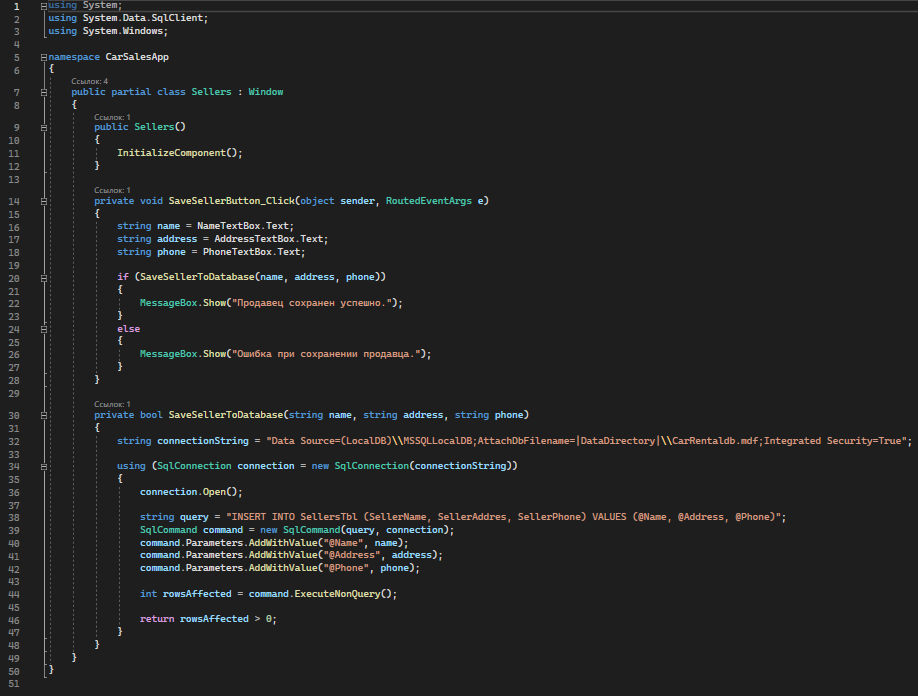


Рисунок 16 - Окно Sellers

Создается подключение к базе данных с использованием строки подключения connectionString.

Выполняется SQL-запрос для вставки данных в таблицу SellersTbl. Параметры @Name, @Address и @Phone заполняются данными из текстовых полей.

Если операция вставки данных в базу данных успешна (возвращается значение больше 0), выводится сообщение об успешном сохранении. В противном случае выводится сообщение об ошибке.

Этот код обеспечивает сохранение данных о продавцах в вашей базе данных, что позволяет вам отслеживать информацию о продавцах, работающих с продажей автомобилей.

В конструкторе класса Free (рис.17), происходит инициализация окна и вызывается метод LoadFreeCarsData, который загружает данные о свободных автомобилях из базы данных.

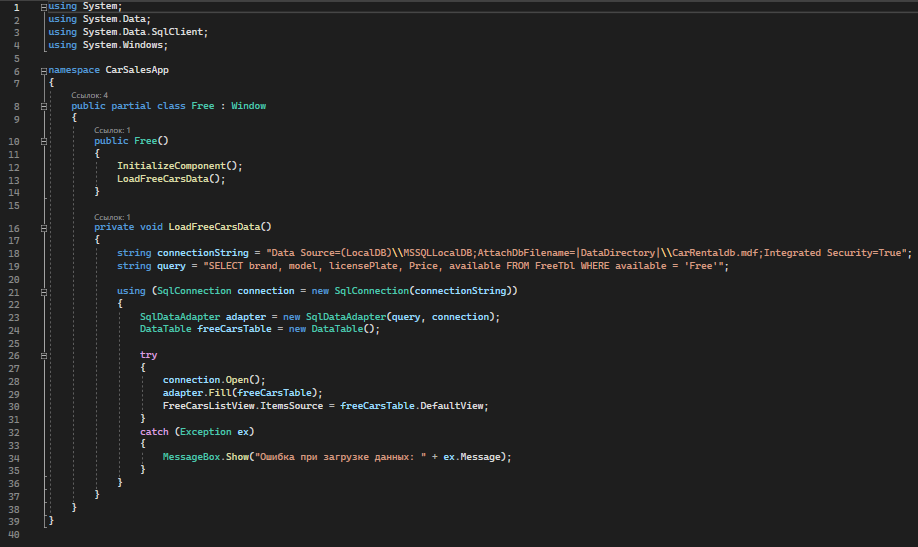


Рисунок 17 – Окно Free

Метод LoadFreeCarsData выполняет SQL-запрос, чтобы получить данные о свободных автомобилях из таблицы FreeTbl, где статус «available» равен «Free».

Используется объект SqlDataAdapter, чтобы выполнить запрос и заполнить результаты в DataTable. Затем данные из DataTable привязываются к FreeCarsListView, что позволяет отобразить данные на форме.

В случае возникновения ошибки при загрузке данных, выводится сообщение об ошибке, включая сообщение об исключении, если таковое имеется.

Этот код обеспечивает отображение списка свободных автомобилей из вашей базы данных на форме «Свободные».

В конструкторе класса Tablica, происходит инициализация окна и вызывается методы LoadSoldCarsData и LoadFreeCarsData, которые загружают данные о проданных и свободных автомобилях из базы данных (рис.18).

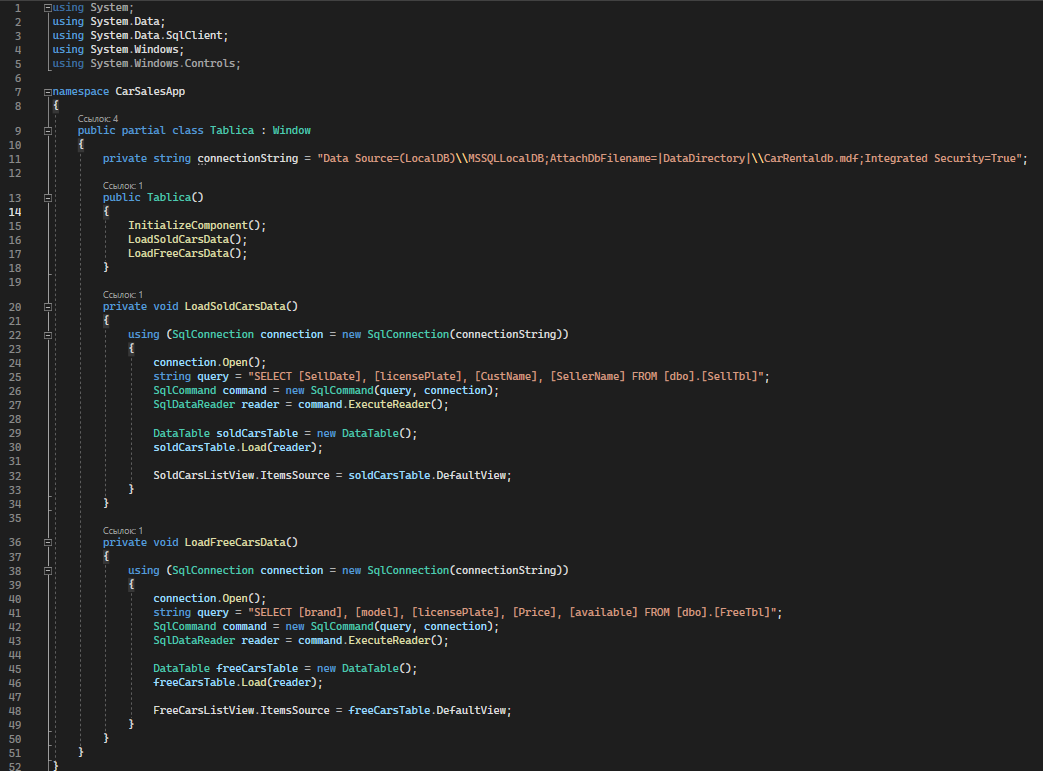


Рисунок 18 – Окно Tablica

Метод LoadSoldCarsData выполняет SQL-запрос для получения данных о проданных автомобилях из таблицы SellTbl. Результаты запроса загружаются в DataTable, который затем связывается с SoldCarsListView, что позволяет отобразить данные на форме.

Метод LoadFreeCarsData выполняет SQL-запрос для получения данных о свободных автомобилях из таблицы FreeTbl. Результаты запроса также загружаются в DataTable и связываются с FreeCarsListView.

Для выполнения SQL-запросов используется объект SqlConnection. Результаты запросов загружаются в DataTable, который затем связывается с ListView, чтобы отобразить данные.

Этот код обеспечивает отображение данных о проданных и свободных автомобилях на форме «Таблица», что позволяет вам отслеживать информацию о состоянии автомобилей и продажах.

3.2 Работа с базой данных

Приложение использует базу данных CarRentalDb, которая хранит всю необходимую информацию. База данных локальная. В коде каждого экрана, связанного с базой данных, используются SQL-запросы для выполнения операций добавления, обновления, удаления и извлечения данных.

Примеры работы с базой данных:

Для сохранения информации о машинах в таблицы CarTbl и FreeTbl используются SQL-запросы INSERT INTO.

Для проверки аутентификации пользователя на экране авторизации используется SQL-запрос SELECT COUNT(\*) для проверки совпадения логина и пароля в таблице UserTbl.

Для загрузки данных на экран «Свободные» используется SQL-запрос, который выбирает только машины с доступностью «Free» из таблицы FreeTbl.

Для загрузки данных на экран «Таблица» используются SQL-запросы, которые выбирают информацию о проданных автомобилях из таблицы SellTbl.

В коде каждого экрана реализованы функциональности для взаимодействия с данными. Например, на экране «Машины» данные о машинах вводятся пользователем и сохраняются в базе данных. На экране «Продажа» данные о продаже регистрируются с указанием соответствующих автомобиля, клиента и продавца. Таким образом, каждый экран выполняет определенные функциональные задачи в приложении, обеспечивая полноценное взаимодействие с данными и пользователями.

3.4 Тестирование и отладка

В результате тестирования и отладки ожидается, что приложение будет работать без существенных ошибок, и его производительность будет удовлетворительной для пользователей.

Каждый функциональный модуль приложения был тщательно протестирован на соответствие его задачам. Например, на экране «Машины» были проведены тесты на добавление, обновление и удаление машин. Экран «Продажа» был протестирован на корректное регистрирование продаж и т.д.

Были проверены взаимодействия различных модулей приложения. Например, взаимодействие между экранами «Машины», «Продажа» и «Свободные» было протестировано для убеждения в правильной передаче данных между ними.

В процессе тестирования возникали такие ошибки как:

Логические ошибки, Ошибки ввода данных, Производительность.

В ходе оптимизации все перечисленные ошибки были исправлены, а приложение заново протестировано. К примеру длительная загрузка при запуске приложения была исправлена путем увеличения количества «%» до значения 15. А ошибка с синхронизацией сохранения данных машин сразу в 2 SQl-таблицы была исправлена путем добавления практически идентичных строк кода C#. Неполадка с некорректным появлением лишь «Free» машин в соответствующем окне-таблице была исправлена после добавления «binding» строк в Free.xaml.

Заключение

В ходе выполнения работы студентами Бычковым Андреем и Ерыгиным Максимом группы 9ИС-380 было разработано приложение для продажи автомобилей на платформе WPF с использованием C#. Приложение предоставляет функциональность для работы с данными о машинах, покупателях, продажах и продавцах. База данных была спроектирована и реализована для хранения необходимой информации.

В рамках данного проекта каждый из разработчиков внес огромный вклад во все аспекты разработки. Совместное усилие и взаимодействие команды были ключевыми факторами в достижении успешных результатов в разных областях разработки приложения. Ниже приведены основные аспекты, в которых каждый разработчик сыграл важную роль:

Дизайн и GUI: Разработчик Бычков Андрей активно участвовал в проектировании пользовательского интерфейса приложения. Это включало в себя создание макетов, выбор цветовой палитры, размещение элементов управления и обеспечение удобной навигации.

Написание кода: Каждый разработчик принимал участие в написании кода приложения. Создание классов, методов и обработчиков событий, а также взаимодействие с базой данных и реализацию бизнес-логики приложения.

Поиск литературы и источников: Разработчик Ерыгин Максим исследовал различные источники литературы и онлайн-ресурсы, чтобы получить информацию, необходимую для разработки. Это включало в себя официальную документацию, книги, учебники и академические статьи.

Тестирование и выявление ошибок: Все разработчики активно участвовали в тестировании приложения. Они создавали тестовые случаи, проводили отладку и выявляли ошибки. Затем ошибки анализировались и устранялись совместно.

Разработка приложений на платформе WPF с использованием C# позволяет создать современные и интуитивно понятные пользовательские интерфейсы для разнообразных приложений.

Основы проектирования базы данных критически важны для разработки приложений, требующих хранения и обработки данных. Грамотное проектирование базы данных помогает обеспечить эффективное взаимодействие с данными.

Проведение тестирования и отладки неотъемлемая часть разработки приложения. Тщательное тестирование помогает выявить и исправить ошибки, а также обеспечить высокую производительность и надежность приложения.

Для дальнейшего совершенствования приложения рекомендуется уделить внимание его интерфейсу, возможностям пользовательской настройки, а также обеспечить защиту данных и безопасность при работе с базой данных.

Учебная практическая работа демонстрирует применение технологий WPF, SQL и C# в создании функционального приложения с базой данных.

Списокиспользованнойлитературы

1. Andrew Troelsen и Philip Japikse. Pro C# 7: With .NET and .NET Core.
2. Adam Nathan. WPF 4.5 Unleashed.
3. Jon P. Smith. Entity Framework Core in Action.
4. Marek Chmel и Vladimir. SQL Server 2019 Administrator's Guide.
5. Randall J. Dyck. Practical Database Design for the Web.
6. Официальная документация Microsoft для C#, WPF и Entity Framework Core.