**Содержание**

[Введение 4](#_Toc101254385)

[1. Аналитическая часть 5](#_Toc101254386)

[1.1. Описание предметной области 5](#_Toc101254387)

[1.2. Инфологическая модель предметной области 6](#_Toc101254388)

[1.3. Модель предметной области в стандарте IDEF0 7](#_Toc101254389)

[1.4. Диаграмма потоков данных 9](#_Toc101254390)

[1.5. Прототипирование интерфейса 9](#_Toc101254390)

[2. Проектная часть 10](#_Toc101254391)

[2.1. Обзор инструментальных средств создания продукта 10](#_Toc101254392)

[2.2. Обоснование выбора средств реализации ИС 13](#_Toc101254393)

[2.3. Описание логики функционирования ПО 14](#_Toc101254394)

[2.4. Реализация интерфейса ПО 17](#_Toc101254395)

[2.5. Написание программного кода 19](#_Toc101254396)

[2.6. Защита информации 21](#_Toc101254396)

[2.7. Тестирование 23](#_Toc101254396)

[Заключение 25](#_Toc101254397)

[Список использованных источников 27](#_Toc101254398)

[Приложения 29](#_Toc101254399)

# Введение

Сегодня все больше людей используют ресурсы интернета. Ведь это действительно очень удобно. Теперь, когда тебе нужно что-то заказать, это можно сделать даже в домашних условиях, используя для этого телефон или компьютер, подключенный в глобальную сеть.

Теперь большинство систем в той или иной степени связаны с функциями хранения и обработки информации. Для этого существуют веб-сайты и приложения. Они очень сильно облегчают жизнь, ведь теперь нужная информация стала в разы доступней.

**Актуальность темы**: потребность автоматизации процесса получения информации об автосалоне

**Цель:** разработать приложение для автосалона

**Объект:** деятельность автосалона

**Предмет:** приложение для автосалона

**Для достижения цели, требуется выполнить следующие задачи:**

1. Описать предметную область;
2. Составить инфологическую модель предметной области;
3. Составить модель предметной области в стандарте IDEF0;
4. Составить DFD диаграмму предметной области;
5. Создать прототипы интерфейса;
6. Произвести обзор инструментальных средств создания продукта;
7. Обосновать выбор средств реализации ИС;
8. Описать логику функционирования ПО;
9. Реализовать интерфейсы ПО;
10. Написать программный код;
11. Описать механизмы защиты информации;
12. Произвести тестирование готового продукта.

# Аналитическая часть

* 1. **Описание предметной области**

Основным видом деятельности автосалона является реализация автомобилей на рынке.

Автосалон имеет зарегистрированный товарный знак, расчётный и иные счета в рублях и иностранной валюте в учреждениях банков. Автосалон имеет в своём составе администрацию, бухгалтерию, отдел продаж, юридический отдел, рекламный отдел, кадровый отдел и магазин.

Общее руководство предприятием осуществляет генеральный директор. Основными подразделениями являются:

Бухгалтерия - осуществляет приём платежей от покупателей, расчет и выдачу зарплаты, налоговую отчётность.

Отдел продаж - занимается непосредственно обслуживанием клиентов, оформлением заказов, оформлением покупок.

* 1. **Инфологическая модель предметной области**

Инфологическая модель — это описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных, называют **инфологической моделью данных. (рисунок 1.1)**

**Она показывает, как связаны экземпляры сущностей между собой. Если связь устанавливается между двумя сущностями, то она определяет взаимосвязь между экземплярами одной и другой сущности.**

**Комплексная и оптимизированная модель данных помогает создать упрощенную логическую базу данных, которая устраняет избыточность, снижает требования к хранилищу и обеспечивает эффективное извлечение.**

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой БД.



Рисунок 1.1 - Информационно-логическая модель данных.

Данная модель данных указывает, что клиент оформляет заказ на покупку автомобиля. Позиции показывают какая стоимость и какая марка будет у автомобиля. Чтобы осуществить заказ клиент должен оставить паспортные данные и номер телефона. В дальнейшем указывается стоимость автомобиля. В соответствии с данной моделью, можно создать бесконечное количество позиций.

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

IDEF0 — это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов, так же она указывает, что между функциями, которые входят в различные подсистемы, должно быть, как можно меньшей связей. На одном уровне должно быть не меньше 5 и не больше 3 функций. (рис. 1.2).

IDEF0 обозначает определение интеграции для моделирования процессов — методологию создания общедоступных доменов, которая используется для моделирования предприятий и их процессов, чтобы их можно было понять и улучшить.



Рисунок 1.2 – Контекстная диаграмма IDEF0

Принцип декомпозиции IDEF0 применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели. (рис. 1.3)



Рисунок 1.3 – Декомпозиция IDEF0

Таблица 1.1 - Характеристика объектов модели IDEF0

|  |  |  |  |
| --- | --- | --- | --- |
| Объект | Название | Определение | Описание |
| Функция А1 | Оформить заказ автомобиля у поставщика | Заказ | Автосалон заказывает у поставщика необходимую модель автомобиля для дальнейшей продажи |
| Функция А2 | Получение заказа от поставщика | Получение | Нужная модель автомобиля получена |
| Функция А3 | Оформление продажи | Оформление отчёта об обработке | Подготовка отчётных документов, подтверждающий продажу |
| Вход А1 | Обращение покупателя | Приведение желаний  покупателя в соответствие с возможностями  автосалона | Обращение может быть представлено в устной или письменной форме. |
| Выход А1 | Поставка | Документы, подтверждающие поставку модели авто, которых нет в  наличии. | Документы, подтверждающие поставку модели авто, которых нет в  наличии, но были заказаны |
| Выход А2 | Автомобили | Автомобили, которые будут оформлять на продажу | Автомобили, оформляемые на продажу |
| Выход А3 | Проданное авто | Проданный автомобиль | Автомобиль, приобретённый покупателем |
| Контроль А1 | Перечень поставщиков | Данные о поставщиках | Поставщики, к которым автосалон обращается при отсутствии товара |

* 1. **Диаграмма потоков данных**

Диаграмма в нотации DFD (диаграмма потоков данных) – это специальная нотация, предназначенная для моделирования информационных систем с точки зрения хранения, обработки и передачи данных (рис. 1.4 и рис. 1.5)



Рисунок 1.4 – Контекстная диаграмма потоков данных



Рисунок 1.5 – Декомпозиция потоков данных.

Диаграмма показывает оформление договора о купли-продажи в автосалоне. Так же показывает процесс заказа автомобиля, опираясь на данные об автомобилях и документов клиента.

**1.5 Прототипирование интерфейса**

Я спроектировал как будет выглядеть окно регистрации (рисунок 1.6). Здесь осуществляется вход в программу, где в дальнейшем можно работать с информацией из таблиц.



Рисунок 1.6 - Прототип окна авторизации

Прототип экранной формы (рисунок 1.7). На этом моменте можно смотреть, изменять или удалять данные из таблицы. Причем изменять или удалять данные может только администратор

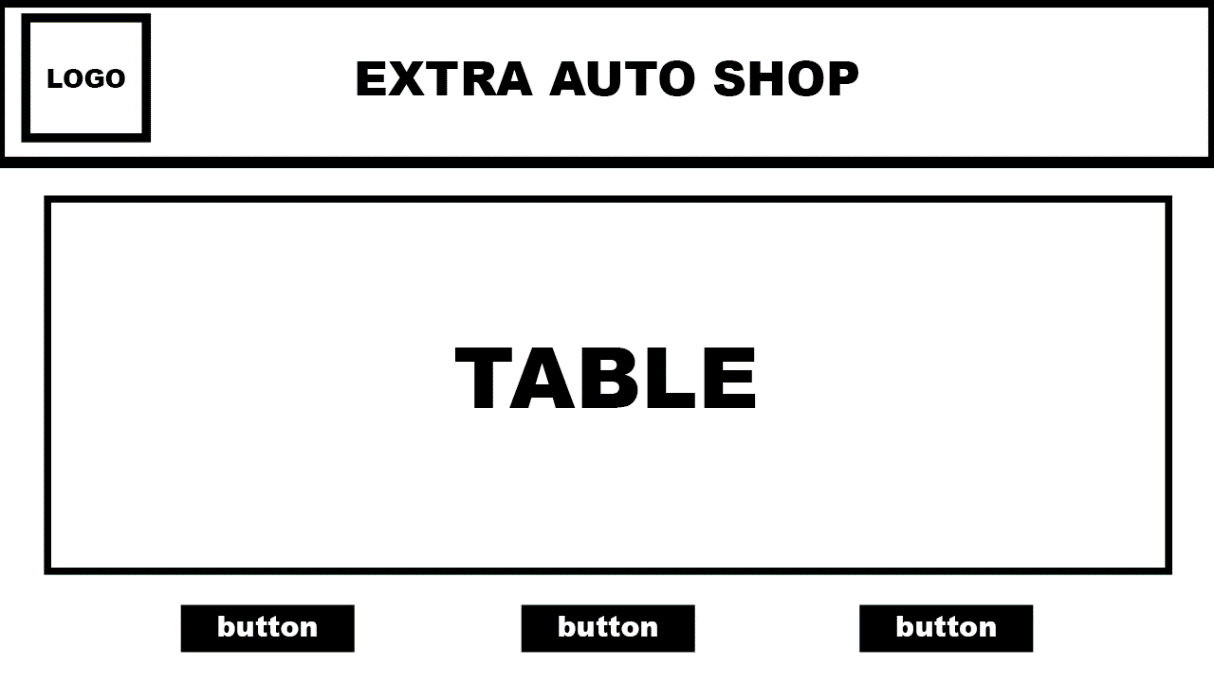


Рисунок 1.7 - Прототип главной экранной формы

Прототип окна редактирование таблицы (рисунок 1.8). Здесь можно редактировать информацию в таблице.

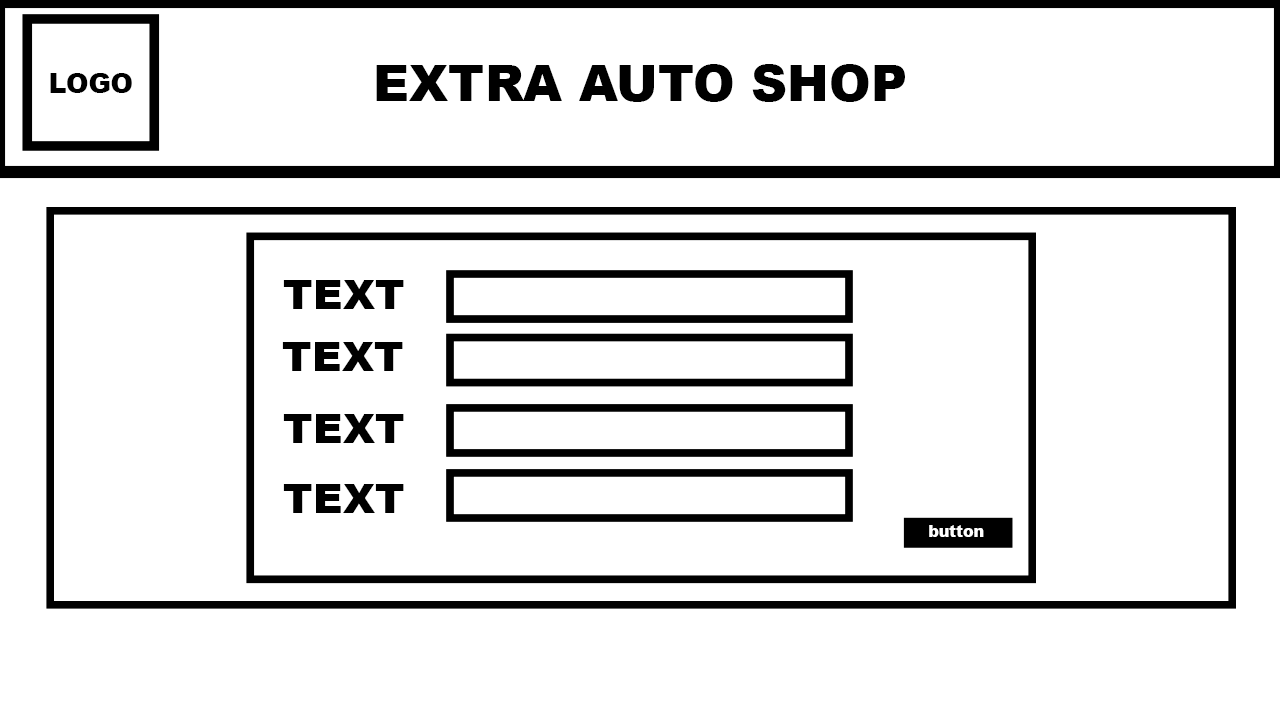


Рисунок 1.8 - Прототип окна редактирования таблицы

# 2. Проектная часть

**2.1 Обзор инструментальных средств создания продукта**

Я решил рассмотреть такие инструменты как Net Framework, Microsoft SQL Server, 1С: Предприятие, 1C в Облаке, Java Script, Python, C++, MySQL, Eclipse.

Преимущества Net Framework: кроссплатформенность

NET Framework - среда исполнения под Windows. Она полезна в создании классических Windows-приложений и веб-приложений ASP.NET. Большое количество уже написанных библиотек кода, более широкие возможности по сравнению с .NET Core. На данный момент Microsoft очень активно развивает Core ветку, и быстрыми темпами наращивает функционал.

Недостатки Net Framework: Ограниченность в объектно-реляционной поддержке. Обновления происходят только в новых версиях. Это значит, если в текущую версию внесли изменения, поддерживаться будет только новая версия, но не исходная. Поскольку .NET Framework является продуктом Microsoft, разработчику приложения необходимо будет придерживаться правил поставщика. Высокая стоимость лицензирования.

Преимущества SQL Server:

СУБД масштабируется, поэтому работать с ней можно на портативных ПК или мощной мультипроцессорной технике. Процессор может одновременно обрабатывать большой объем запросов. Размер страниц – до 8 кб, поэтому данные извлекаются быстро, подробную и сложную информацию хранить удобнее. Система позволяет обрабатывать транзакции в интерактивном режиме, есть динамическая блокировка.

Недостатки SQL Server: Низкая безопасность — каждый, кто работает с каталогом, имеет доступ на «запись», а значит может сделать копию базы данных. Очень мало обучающей информации по данной среде.

Преимущества 1С: Предприятия:

Открытость конфигураций и гибкость настроек, наличие информационно-технологического сопровождения.

Недостатки 1С: Предприятие:

Недостаточный уровень безопасности;

Необходимость платить за обновления;

Сложность внедрения.

Преимущества 1С в Облаке:

Быстрый запуск покупка и настройка коробки занимает несколько дней, а облака − пару минут;

Простота обновлений, выполняется автоматически;

Высокая скорость работы за счёт мощного сервера;

Возможность работать удалённо или объединить базу разрозненных офисов; надёжная защита данных;

Ценовая доступность.

Недостатки 1С в Облаке:

Привязанность к интернету;

Зависимость от провайдера;

Отсутствие возможности доработать базовое решение.

Преимущества Java Script:

Концепция ООП (объектно-ориентированного программирования). Это значит, что программист сам определяет тип данных, его структуру и набор применяемых к нему функций. Это позволяет многократно использовать одни и те же объекты в разных программах, более удобно для организации структуры программ (особенно с большим объёмом кода), помогает избегать ошибок и упрощает поддержку и модернизацию старого кода;

Это язык высокого уровня, другими словами, он больше похож на человеческую речь, а не на машинный код. Следовательно, у него сравнительно простой синтаксис, что делает его быстрым для освоения и удобным для написания кода, его чтения и обслуживания. Есть и более простые варианты (например, Python), однако у человека с базовым пониманием основ программирования здесь не должно возникнуть сложностей;

Безопасность. У Java есть несколько функций, которые ликвидируют часто встречающиеся уязвимости. В частности, это Security Manager – создаваемая для каждого приложения политика безопасности, в которой можно прописать правила доступа;

Удобство для распределённого программирования. Этот язык изначально создавался для совместной работы (в том числе удалённой), поэтому он позволяет совместно использовать данные и программы несколькими компьютерами одновременно;

Принцип «написать один раз и использовать везде» — написанное на Java приложение можно запустить на любой поддерживающей его платформе;

Стабильное и постоянно развивающееся сообщество. По многочисленности и активности с ним мало кто может соперничать. В Сети есть масса ресурсов, где на любой вопрос по этой теме либо уже есть ответ, либо найдётся кто-нибудь, кто его подскажет, равно как и сотни курсов, семинаров и обучающих программ, как платных, так и бесплатных.

Недостатки Java Script:

Низкая скорость. Все высокоуровневые языки приходится компилировать с помощью виртуальной машины, что плохо сказывается на их производительности. Java – не исключение, кроме того, у него есть и некоторые собственные особенности, вызывающие дополнительные проблемы с производительностью;

Многословие (verbosity). Сходство с естественными языками делает Java проще для изучения и понимания, но также ведёт и к тому, что он содержит много лишней информации и довольно громоздок;

Платность для коммерческого использования (с 2019 года).

Преимущества Python:

Понятность кода. Синтаксическая особенность Python — выделение блоков кода отступами, что значительно упрощает зрительное восприятие программ, написанных на этом языке.

Интерпретируемость. Программы, написанные на языке программирования Python, не переводятся в машинный код, а сразу выполняются программой-интерпретатором. Это позволяет запускать код на любой платформе с установленным заранее интерпретатором.

Объектно-ориентированность. Python — это язык, созданный согласно парадигме объектно-ориентированного программирования (ООП). В ней основными являются понятия объекта и класса. Классы — это специальные типы данных, объекты — экземпляры классов. То есть любое значение является объектом конкретного класса. В Python вы можете не только использовать уже существующие классы, но и создавать свои собственные.

Динамическая типизация. В отличие от C-подобных языков программирования, в Python переменные связываются с типом в момент присваивания в них конкретных значений.

Простота синтаксиса и визуального восприятия кода;

Легкость изучения этого языка для новичков;

Изобилие различных библиотек и фреймворков;

Кроссплатформенность;

Динамическая типизация.

Недостатки Python:

Этот язык программирования является одним из самых медленных языков программирования.

Этот язык программирования не подходит для задач, которые требуют большого объема памяти.

Преимущества C++:

Нет потребления ресурсов на вспомогательные операции.

Метапрограммирование — возможность поручать большую часть работы компилятору.

C++ — строгий язык, в нём уже на этапе компиляции ясно, что будет делать программа. А значит, на этапе компиляции можно сделать ряд вычислений и оптимизаций, чтобы программа работала быстрее.

Этот язык программирования позволяет писать гибкие и быстро работающие программы. В других языках приходится платить скоростью за сложность, универсальность, красоту, понятность. В C++ сложные выразительные конструкции превращаются в простой и быстрый микрокод.

Недостатки C++:

Довольно сложный язык для изучения.

Чтобы начать правильно на нём программировать, придётся много чего изучить. Есть миф, что программист, работая на C++, «выстреливает себе в ногу»: легко написать неправильную программу и не заметить этого.

Но ситуация меняется, крупные обновления C++ выходят каждые три года, и многое проверяется на этапе компиляции. «Выстрелить в ногу» стало гораздо сложнее.

В C++ сложно работать с зависимостями.

Есть некоторые системы контроля за пакетами (например, Conan), но они менее удобны, чем аналоги для некоторых других языков.

А если что-то не работает или нужного пакета нет, то получение нужной зависимости часто превращается в головную боль. Необходимо вручную скачивать исходники нужной библиотеки со всеми зависимостями. При этом следить за параметрами компиляции. Разные зависимости могут собираться разными системами сборки. Если ошибиться в параметрах компилятора или версии компилятора, программа не соберётся или будет работать неправильно. Также может возникнуть конфликт версий — когда разные зависимости имеют несовместимые ревизии.

Плюсы MySQL:

Производительность

Для веб-приложений важна скорость работы. Пользователь не будет ждать загрузки страницы — он просто уйдет с неё, если приложение не будет работать достаточно быстро. В плане производительности приложения «болезненным» местом является база данных, и MySQL не подведет вас.

Кроссплатформенность

MySQL предоставляет кроссплатформенную базу данных: она работает на Linux, FreeBSD и конечно на Windows. Этот критерий нужно учитывать при выборе СУБД для проектов, нацеленных на несколько платформ, в частности веб-приложений. К слову, MySQL входит в состав популярных стеков для разработки веб-приложений — LAMP (Linux, Apache, MySQL, PHP) и WAMP (Windows, Apache, MySQL, PHP).

Бесплатная СУБД

Oracle развивает MySQL как проект с открытым исходным кодом, что позволяет сообществу вносить в него свой вклад и, при необходимости, адаптировать его под свои задачи. СУБД предоставляется для конечных пользователей бесплатно. Многие компании выбирают в качестве СУБД MySQL из-за того, что она бесплатна.

Oracle предлагает клиентам некоторые проприетарные модули с закрытым кодом, для использования которых потребуется денежное вложение. Но эти модули нужны не всем и не всегда.

Выбор движков

Движок базы данных (database engine) — это программный модуль, с помощью которого СУБД создает, читает и обновляет данные в базе. В MySQL есть два типа движков: транзакционные и нетранзакционные. Транзакционные движки хранения данных подразумевают, что операции в этих БД могут быть отменены, если они не завершены.

Всего MySQL поддерживает 8 движков:

InnoDB — это распространенный движок, который используется в MySQL по умолчанию. Он поддерживает ACID, блокировку на уровне строк, восстановление после сбоев и многопоточность. Также это единственный движок, который обеспечивает ограничение ссылочной целостности внешнего ключа. Oracle советует использовать именно InnoDB, за исключением ряда случаев.

MyISAM — этот движок использовался по умолчанию до появления InnoDB. И в отличие от него, MyISAM не поддерживает ACID. В этом движке реализована блокировка только на уровне таблиц, поэтому он небезопасен для транзакций. В тоже время MyISAM оптимизирован для чтения. Сфера применения MyISAM — это системы хранения с минимальными транзакционными данными.

Memory — этот движок реализуется хранение данных в памяти в виде кучи. Такая система обеспечивает очень быстрый доступ к данным. Memory поддерживает блокировку на уровне таблиц и хранит данные в оперативной памяти — при перезагрузке сервера они теряются. Этот движок подходит для временных таблиц.

CSV — этот движок хранит данные в файлах формата .csv. Не поддерживает индексирование и разделение. CSV используется для обмена данными между различными программами или приложениями.

Merge — движок работает с таблицами MyISAM. Он логически группирует ряд идентичных таблиц MyISAM и ссылается на них, как на один объект. Merge упрощает управление большими объемами данных и используется в системах хранения данных.

Archive — этот движок оптимизирован для высокоскоростной вставки. При вставке он сжимает данные. Archive не поддерживает транзакции, операции DELELE и UPDATE, но зато поддерживает блокировку на уровне строк. Отлично подходит для хранения больших объемов архивных данных, на которые редко ссылаются.

Federated используется создания одной БД на нескольких физических серверах. Подходит для распределенных систем.

Движок Blackhole принимает данные, но не сохраняет их. На первый взгляд этот движок бесполезен, но на самом деле он находит применение при репликации данных и тестировании. Blackhole можно использовать в качестве фильтра между серверами.

Безопасность

MySQL предлагает пользователям высокий уровень безопасности. В MySQL есть встроенные инструменты безопасности, которые поддерживают управление пользователями и их привилегиями. При недостатке стандартных инструментов пользователь всегда может установить дополнительные плагины

Недостатки MySQL:

Недостаточная надежность. В вопросах надежности некоторых процессов по работе с данными (например, связь, транзакции, аудит) MySQL уступает некоторым другим СУБД.

Низкая скорость разработки. Как и многим другим программным продуктам с открытым кодом, MySQL не достает некоторого технического совершенства, что порой сказывается на эффективности процессов разработки.

Преимущество Eclipse:

Гибкость в настройке. При помощи дополнительных плагинов из Eclipse можно сотворить очень функциональный инструмент. А если взять во внимание, что вы можете самостоятельно создать плагин и «навесить» его на эту IDE, тогда перед вами открываются просто безграничные возможности.

Цена. Отсутствие стоимости Эклипс — это приятный бонус, за такой первоклассный инструмент.

Кроссплатформенность. Среда разработки Eclipse разработана на Java, а это значит, что она может запуститься в любой операционной системе, которая поддерживает Java. А этот язык программирования поддерживают практически все современные операционные системы.

Мультипроеткность. В Эклипс можно работать сразу над несколькими проектами. Это удобно, особенно когда нужно из одного проекта в другой переносить программные модули.

Проблемы в работе. У Eclipse очень развита документация, но это не главное преимущество этой IDE. Она достаточно популярна во всем мире и обросла сообществами на разных языках. А это значит, что практически на каждый свой вопрос по Эклипс вы сможете найти правильный ответ.

Интерфейс. Минимальный интерфейс делает свое дело. Нет ничего лишнего, что отвлекало бы внимание от работы над кодом.

Недостатки Eclipse:

Плагины. Вроде плагины — это плюс, но они также могут быть и минусом. К сожалению, но частая проблема с плагинами — это несовместимость. Нет гарантии, что плагин, разработанный вами или кем-то другим, может стабильно работать в Eclipse и не вызывать проблем. Большинство плагинов разработаны энтузиастами, а не командой разработчиков, которая будет следить за их качеством и совместимостью.

Интуитивность. Изначально интерфейс и наличие функционала у Eclipse минимальны. Функционала не хватает для работы над крупными проектами, поэтому его нужно расширять с помощью плагинов. Обилие плагинов создает небольшой хаос в интерфейсе, поэтому большинство новичков не в состоянии разобраться как работать с этой средой разработки. Поэтому бытует мнение, что Эклипс — это среда разработки для профессиональных разработчиков, которые могут разрабатывать собственные плагины и настроить Eclipse «под себя».

**2.2 Обоснование выбора средств реализации ИС**

Я буду использовать среду SQL Server и среду .NET Framework, так как сделав собственный анализ я вычислил для себя главные достоинства, а именно:

Кроссплатформенность, NET Framework - среда исполнения под Windows. Создание классических Windows-приложений и веб-приложений.

Большое количество уже написанных библиотек кода, более широкие возможности по сравнению с .NET Core. На данный момент Microsoft очень активно развивает Core ветку, и быстрыми темпами наращивает функционал. Минусы: Framework. Среди недостатков нужно отметить ограниченность в объектно-реляционной поддержке. Обновления происходят только в новых версиях. Это значит, если в текущую версию внесли изменения, поддерживаться будет только новая версия, но не исходная. Поскольку .NET Framework является продуктом Microsoft, разработчику приложения необходимо будет придерживаться правил поставщика. Еще один минус – высокая стоимость лицензирования. SQL\_Server Плюсы: SQL Server. СУБД масштабируется, поэтому работать с ней можно на портативных ПК или мощной мультипроцессорной технике. Процессор может одновременно обрабатывать большой объем запросов. Размер страниц – до 8 кб, поэтому данные извлекаются быстро, подробную и сложную информацию хранить удобнее. Система позволяет обрабатывать транзакции в интерактивном режиме, есть динамическая блокировка.

В этой программной среле легко перепроектировать уже созданные приложения; достаточно большой выбор полезных инструментов, упрощающих создание приложений и сокращающих время их разработки; наличие кроссплатформенной функции.

Что касается SQL Server, то в нем возможно создать простые запросы. Это очень удобно, ведь можно использовать это для очень быстрого и эффективного извлечения большого объема данных из СУБД. SQL легко изучить, управлять СУБД с помощью SQL несложно. Что касается среды .NET Framework – оно дает дополнительное преимущество для начинающих специалистов – развитое .NET-комьюнити. На таких популярных ресурсах, как GitHub или Stack Overflow, можно найти ответы на неизбежно возникающие у начинающих специалистов вопросы и даже посмотреть примеры кода.

**2.3 Описание логики функционирования ПО**

Для разработки логистики приложения я составил ER-диаграмму (Рисунок 2.1)

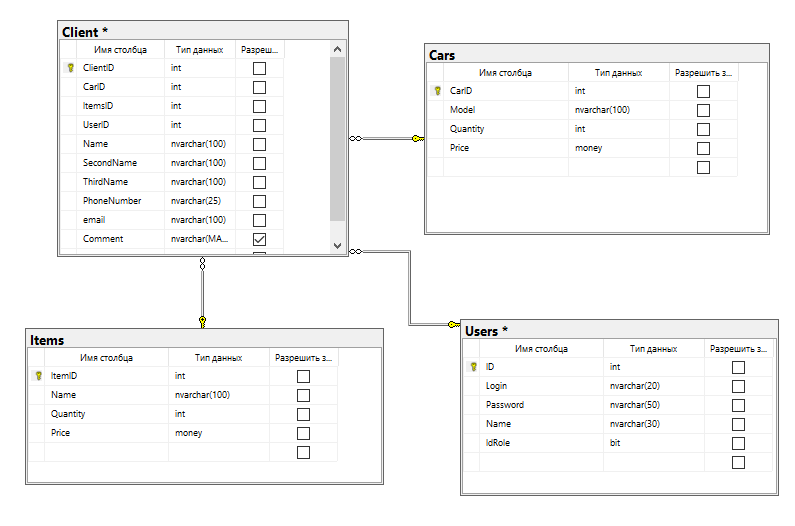


Рисунок 2.1 - ER диаграмма

Таблица со списком автомобилей (рисунок 2.2)

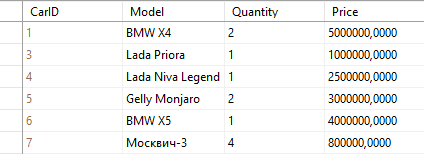


Рисунок 2.2 – Содержимое таблицы «Cars»

Таблица 2.1 Структура таблицы «Cars»

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| CarID | Int | Уникальный идентификатор |
| Model | NVARCHAR (100) | Название модели авто |
| Quantity | Int | Количество авто |
| Price | money | Стоимость авто |

Таблица со списком аксессуаров для автомобилей (рисунок 2.3)

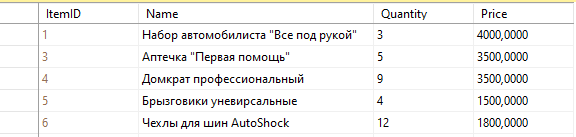


Рисунок 2.3 - Содержимое таблицы «Items»

Таблица 2.2 – Структура таблицы «Items»

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| ItemID | Int | Уникальный идентификатор |
| Name | NVARCHAR (100) | Название аксессуара |
| Quantity | Int | Количество авто |
| Price | money | Стоимость авто |

Таблица со списком клиентов (рисунок 2.4)

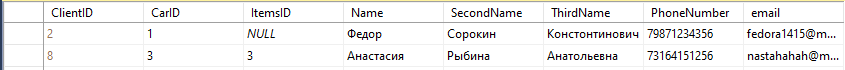


Рисунок 2.4 – Содержимое таблицы «Clients»

Таблица 2.2 – Структура таблицы «Clients»

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| ClientID | Int | Уникальный идентификатор |
| CarID | Int | Внешний ключ |
| ItemsID | Int | Внешний ключ |
| Name | Nvarchar(100) | Имя клиента |
| SecondName | Nvarchar(100) | Фамилия клиента |
| ThirdName | Nvarchar(100) | Отчество клиента |
| PhoneNumber | Nvarchar(25) | Номер телефона клиента |
| email | Nvarchar(100) | Электронная почта |

**2.4 Реализация интерфейса ПО**

При запуске приложения мы видим окно авторизации (рисунок 2.5).

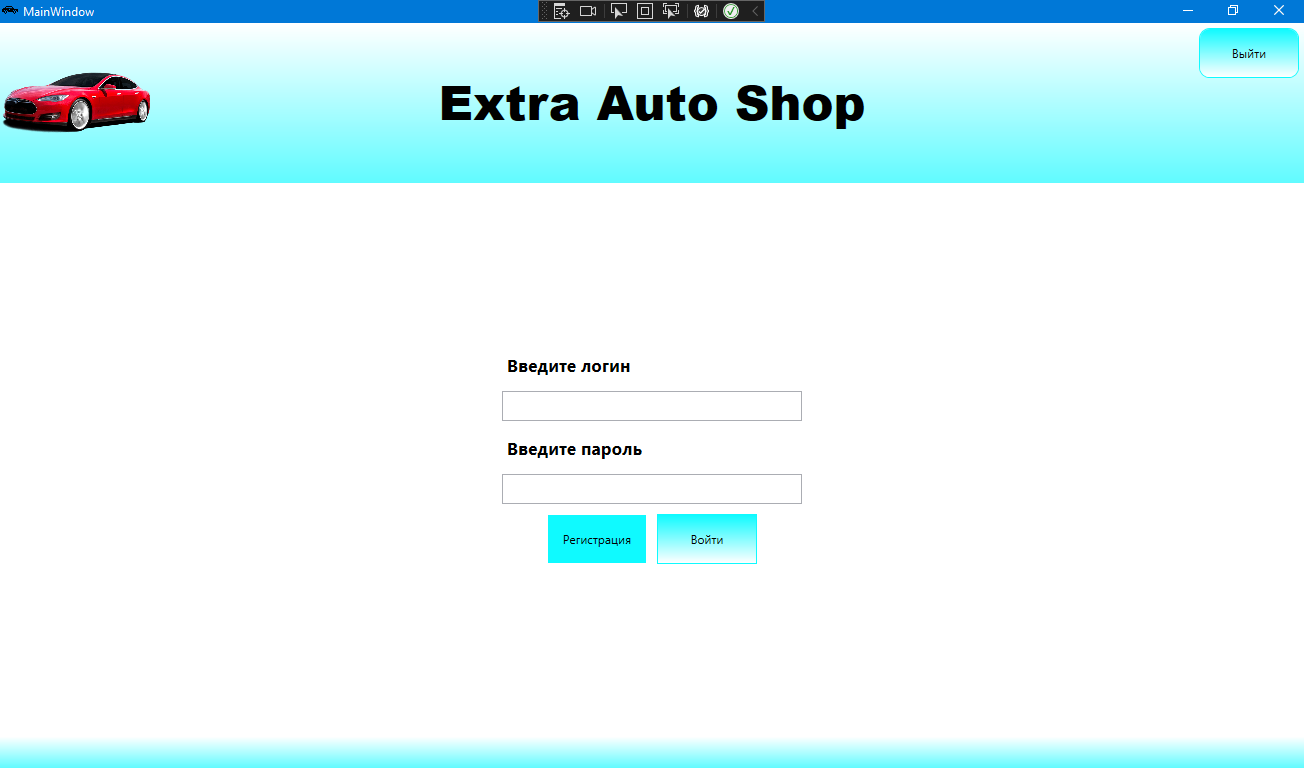


Рисунок 2.5 - Окно авторизации

Мы можем зайти как администратор и как обычный пользователь (рисунок 2.6). Если мы зайдем как администратор, можем изменять данные из таблиц.

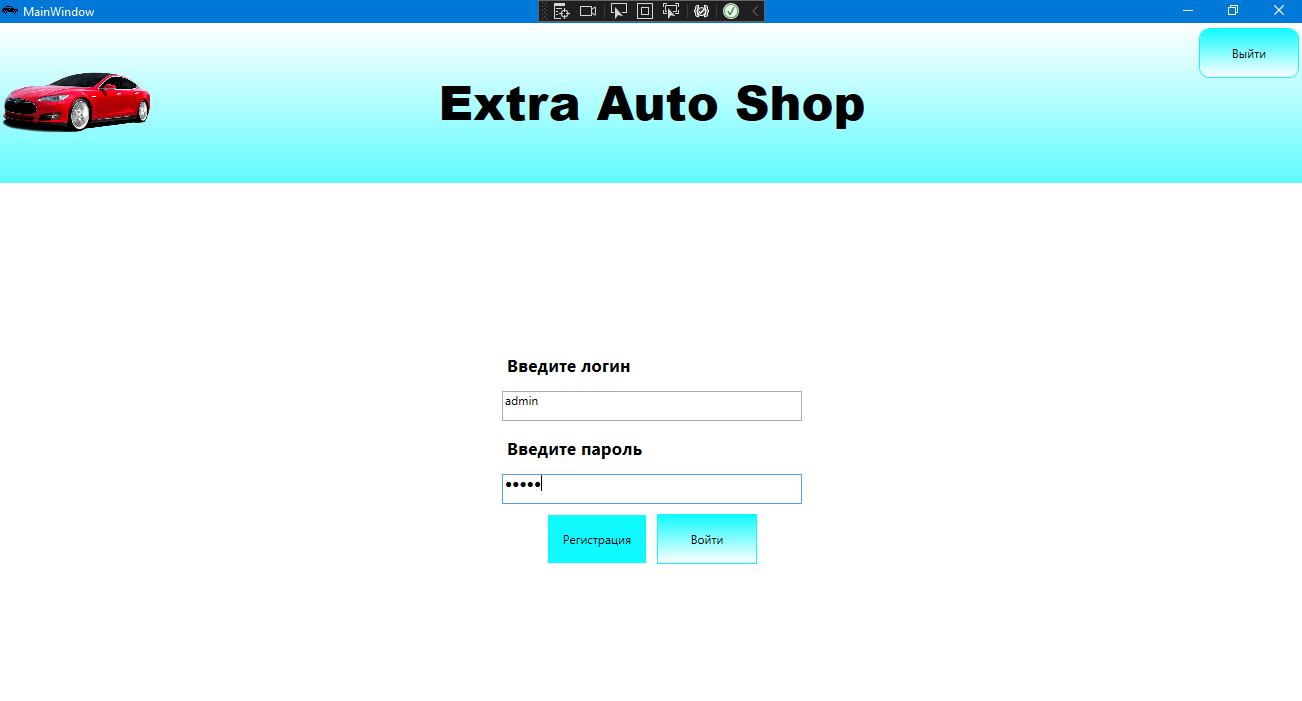


Рисунок 2.6 - Вход в учетную запись администратора

Если мы введем не корректные данные, то у нас появится соответствующее уведомление об ошибки (рисунок 2.7)

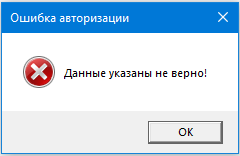


Рисунок 2.7 – Уведомление об ошибки авторизации

После того как мы осуществили вход как администратор, нас встречают ряд кнопок, в которой мы можем выбрать таблицу, с которой хотим работать (рисунок 2.7). В таблице мы можем изменять, удалять или добавлять данные.

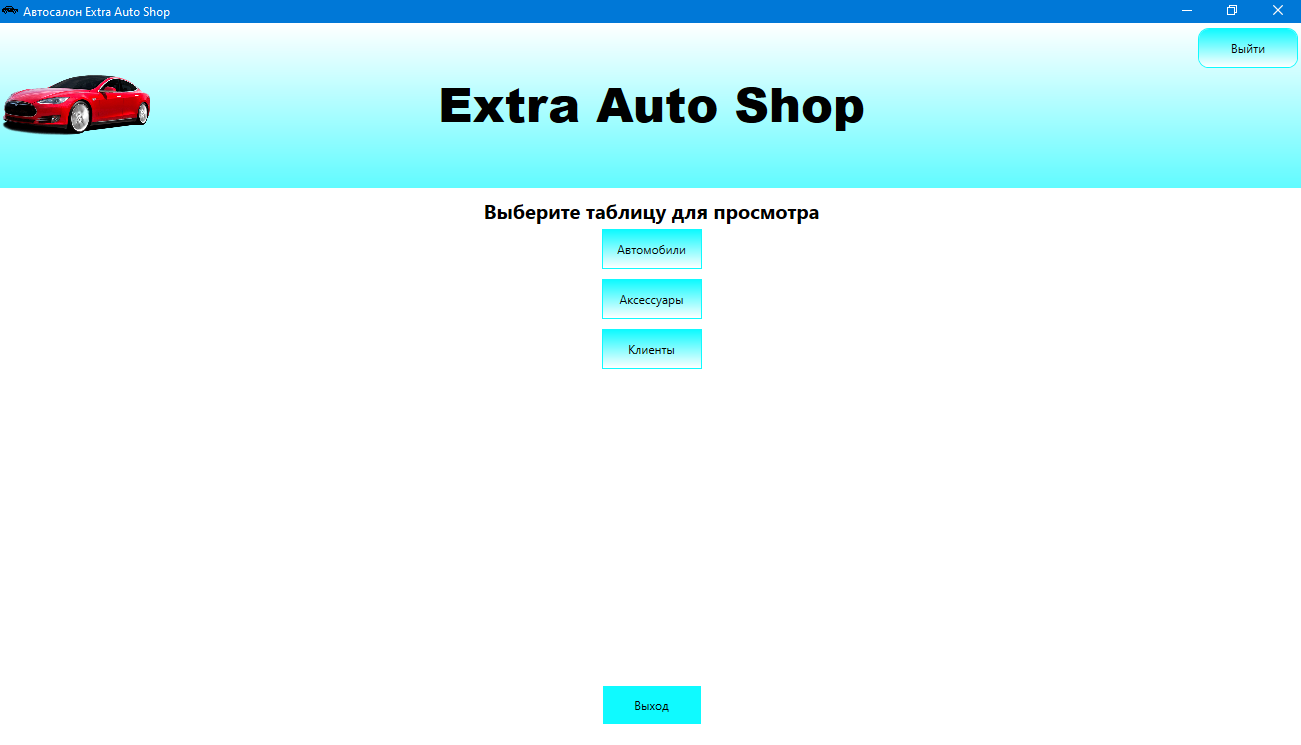


Рисунок 2.7 – Внешний вид страницы администратора

Нажав на кнопку «Автомобили» нас переместит на страницу с таблицей, со списком автомобилей (рисунок 2.8)

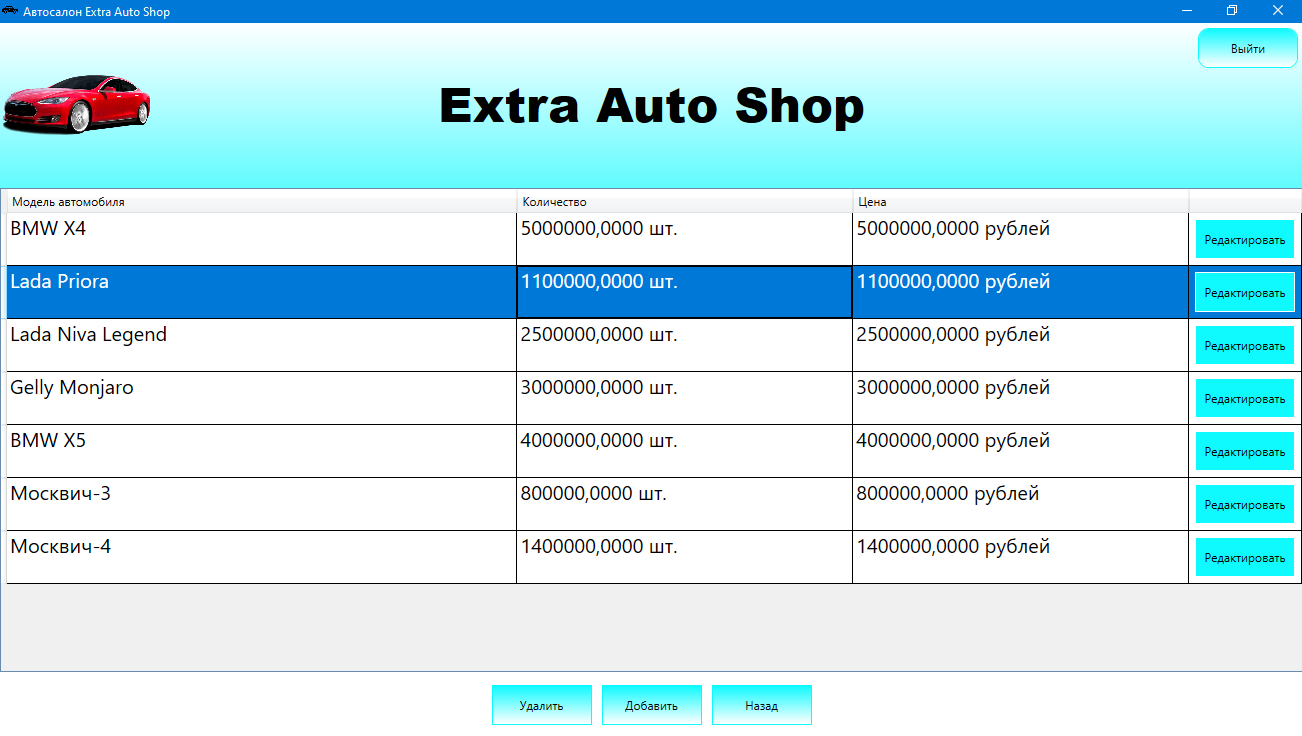


Рисунок 2.8 – Страница с таблицей «Автомобили»

Нажав на кнопку «Редактировать», мы можем изменить данные из таблицы (рисунок 2.8)

# 

Рисунок 2.8 – Окно изменения данных из таблицы «Cars»

Нажав на кнопку «Добавить», мы сможем обновить таблицу новой записью (рисунок 2.9)

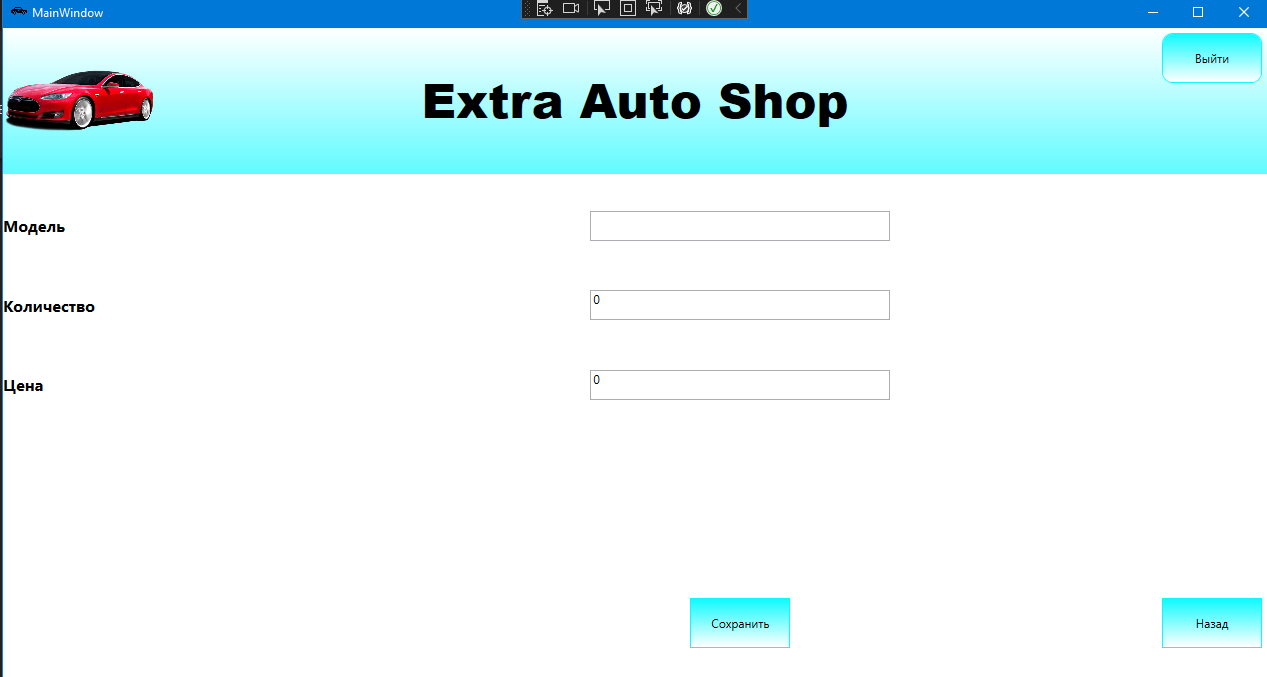


Рисунок 2.9 - Окно добавления новой записи в таблицу

Если мы попытаемся сохранить не корректные данные, то программа не даст нам этого сделать (рисунок 2.10)

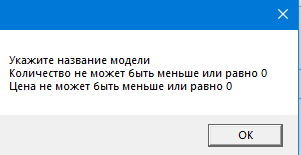


Рисунок 2.10 – Уведомление об ошибки целостности данных

Нажав на кнопку «Аксессуары» (на странице администратора) нас переместит на страницу с таблицей, со списком аксессуаров для автомобилей (рисунок 2.11)

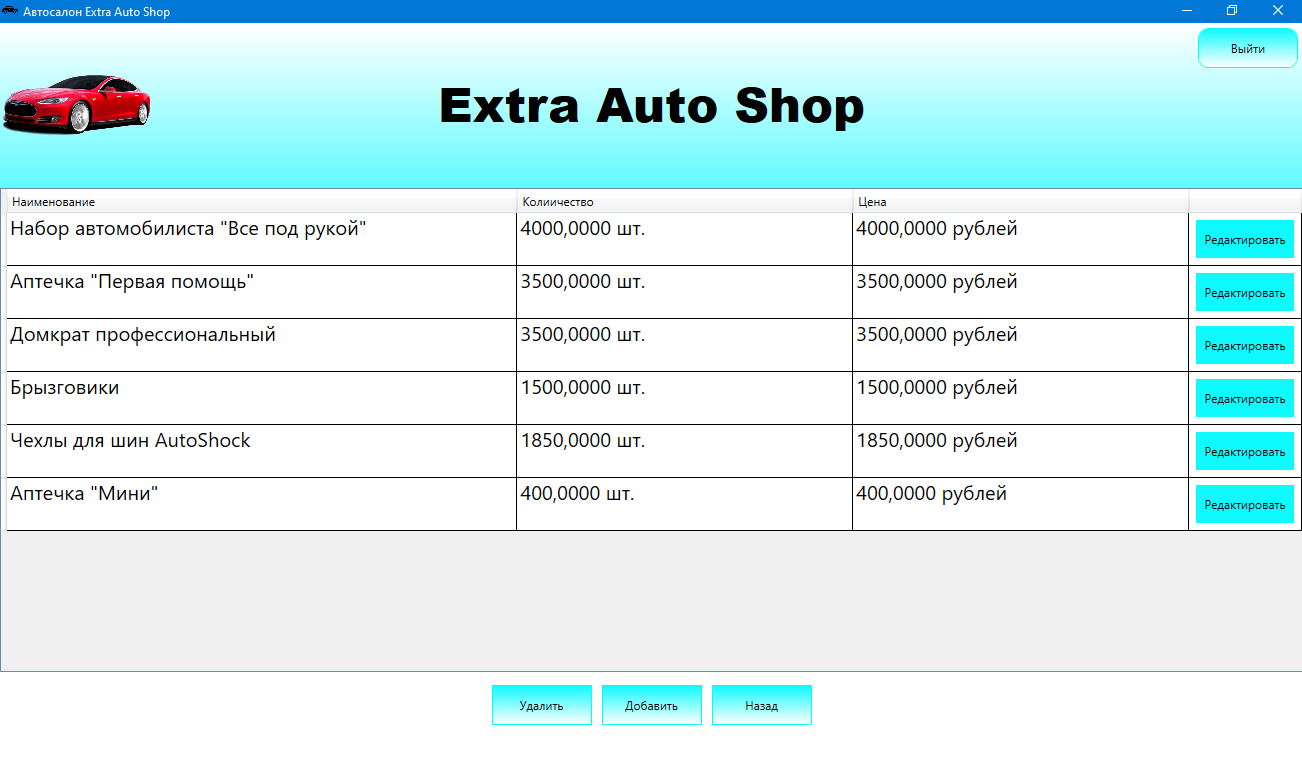


Рисунок 2.11 – Страница с таблицей «Аксессуары»

Когда мы выделим какую-либо запись, чтобы удалить ее, программа сначала попросит подтвердить действие. (рисунок 2.12)

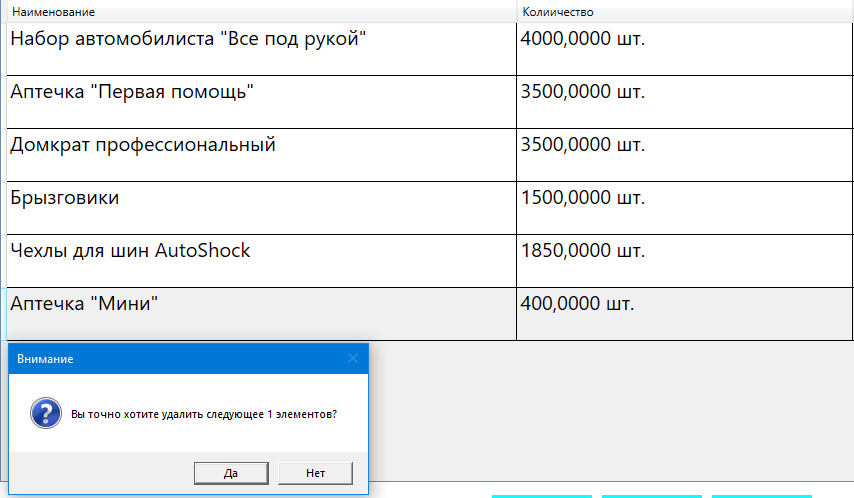


Рисунок 2.12 – Уведомление о подтверждении удаления записи

Нажав на кнопку «Добавить», мы сможем обновить таблицу новой записью (рисунок 2.13)

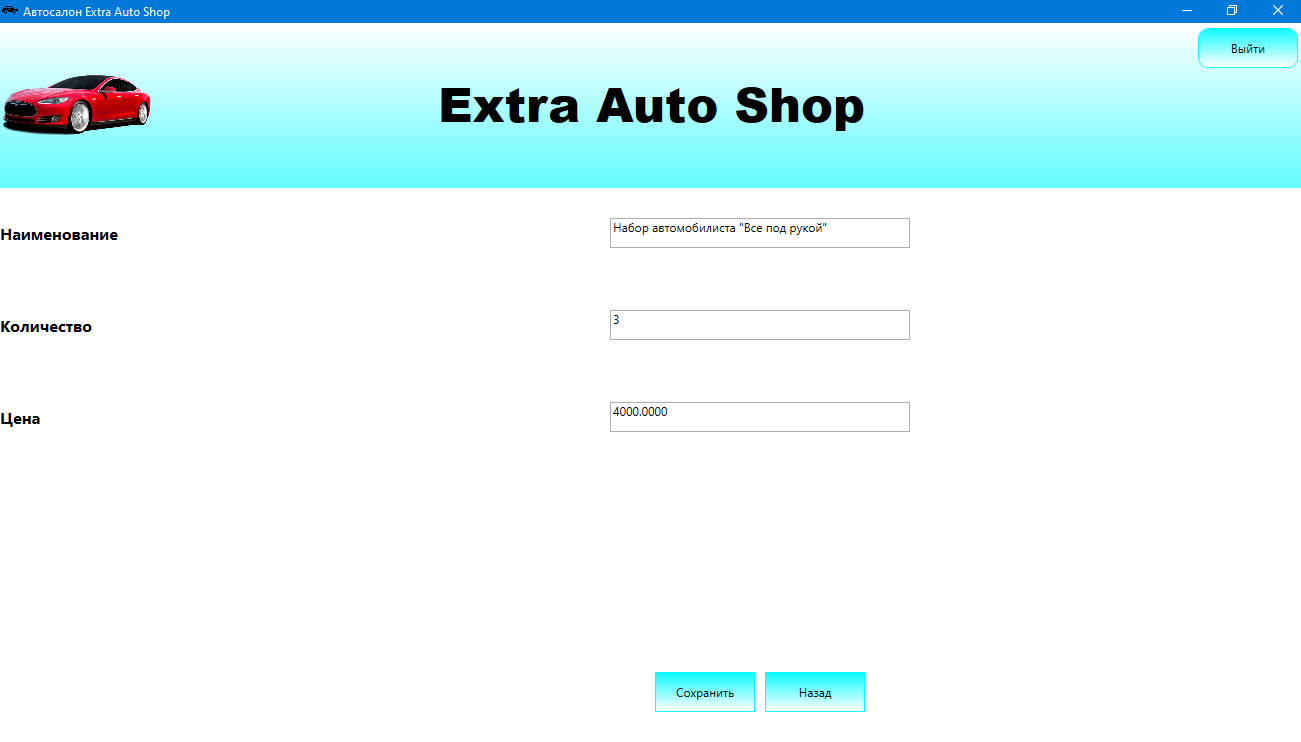


Рисунок 2.13 – Окно изменения данных из таблицы «Items»

Так же если мы попытаемся сохранить не корректные данные, то программа не даст нам этого сделать (рисунок 2.14)

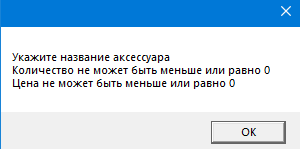


Рисунок 2.14 – Уведомление об ошибки целостности данных

Нажав на кнопку «Клиенты» (на странице администратора) нас переместит на страницу с таблицей, со списком клиентов (рисунок 2.15)

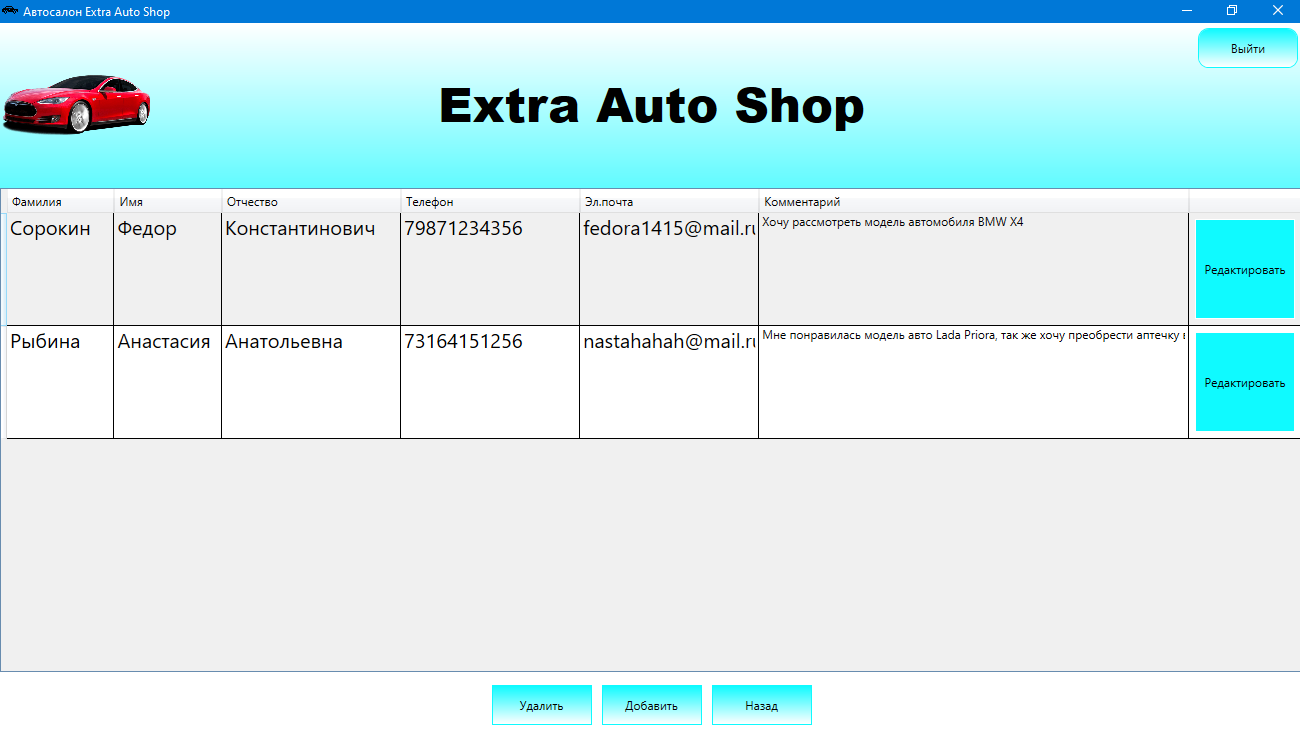


Рисунок 2.15 – Страница с таблицей «Клиенты»

Нажав на кнопку «Добавить», мы сможем обновить таблицу новой записью (рисунок 2.16)

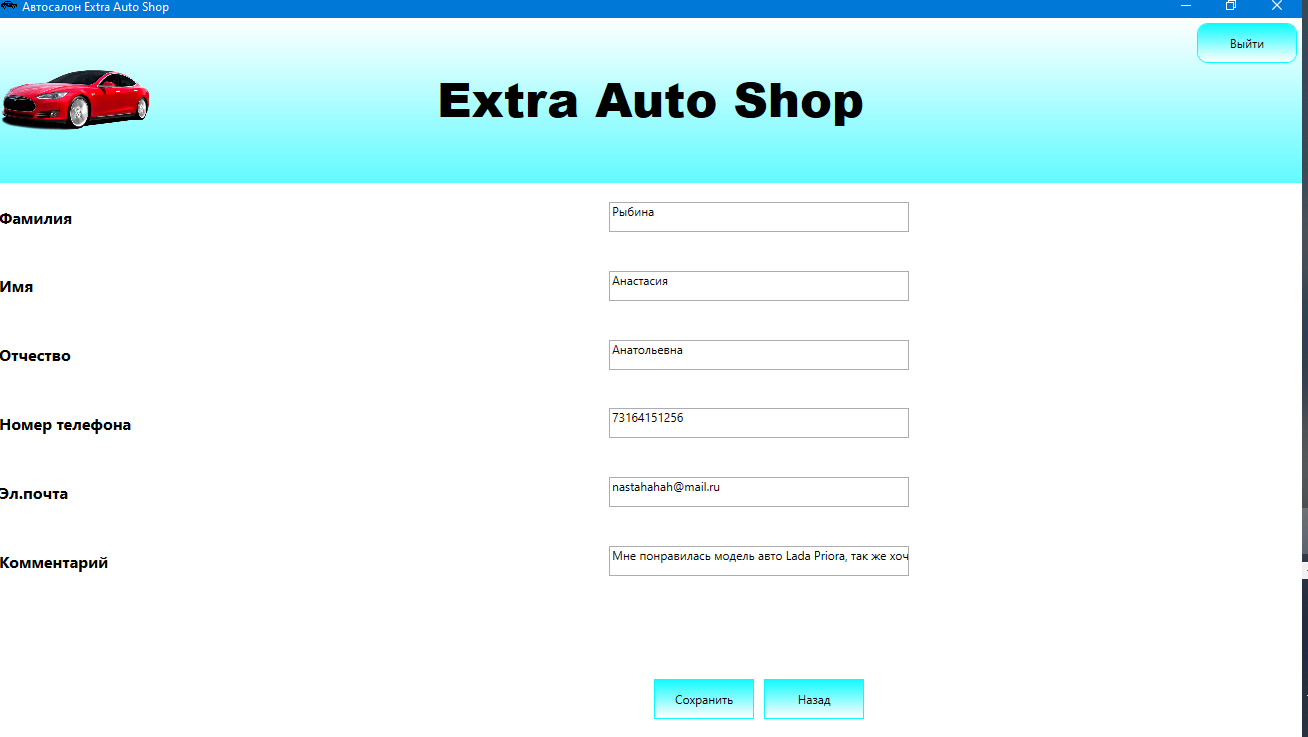


Рисунок 2.16 – Окно изменения данных из таблицы «Clients»

Так же если мы попытаемся сохранить не корректные данные, то программа не даст нам этого сделать (рисунок 2.17)

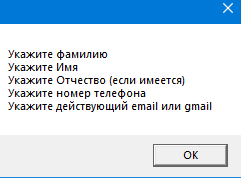


Рисунок 2.17 – Уведомление об ошибки целостности данных

Так же в окне авторизации мы можем зарегистрироваться как обычный пользователь (рисунок 2.18)

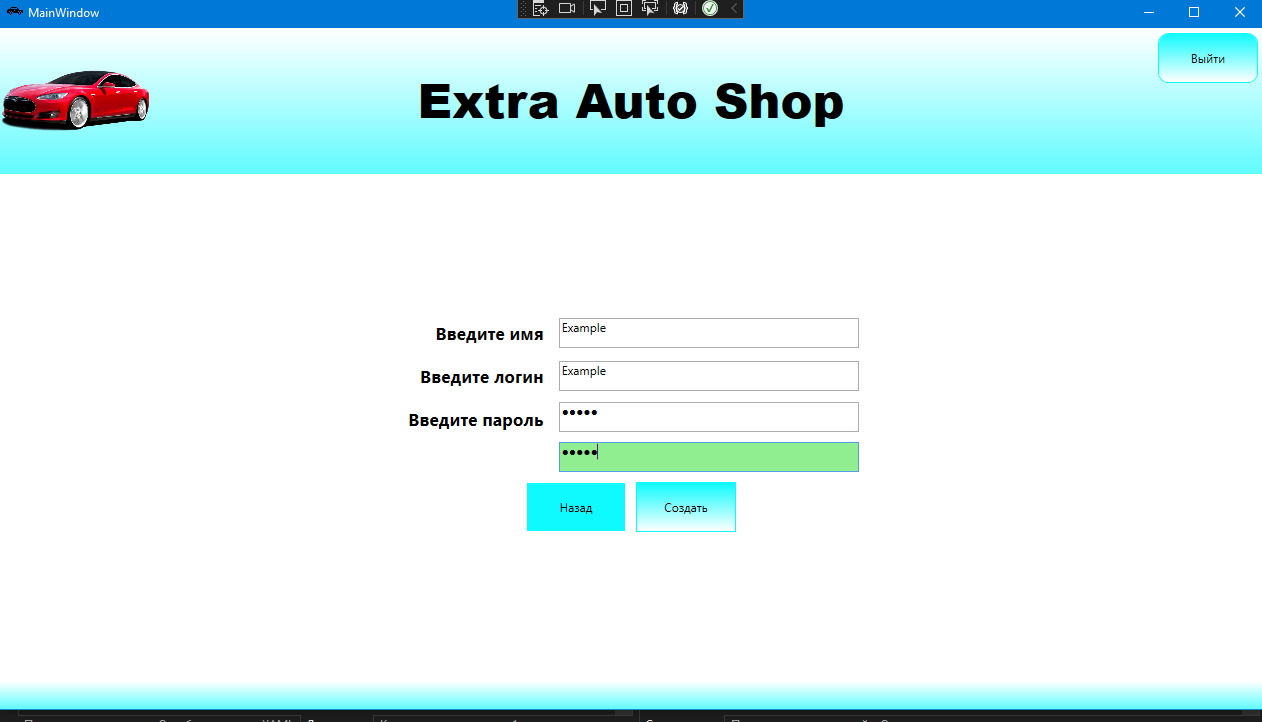


Рисунок 2.18 – Окно регистрации

Если мы попытаемся зарегистрироваться и ввести неверные данные, программа не даст нам этого (рисунок 2.19)

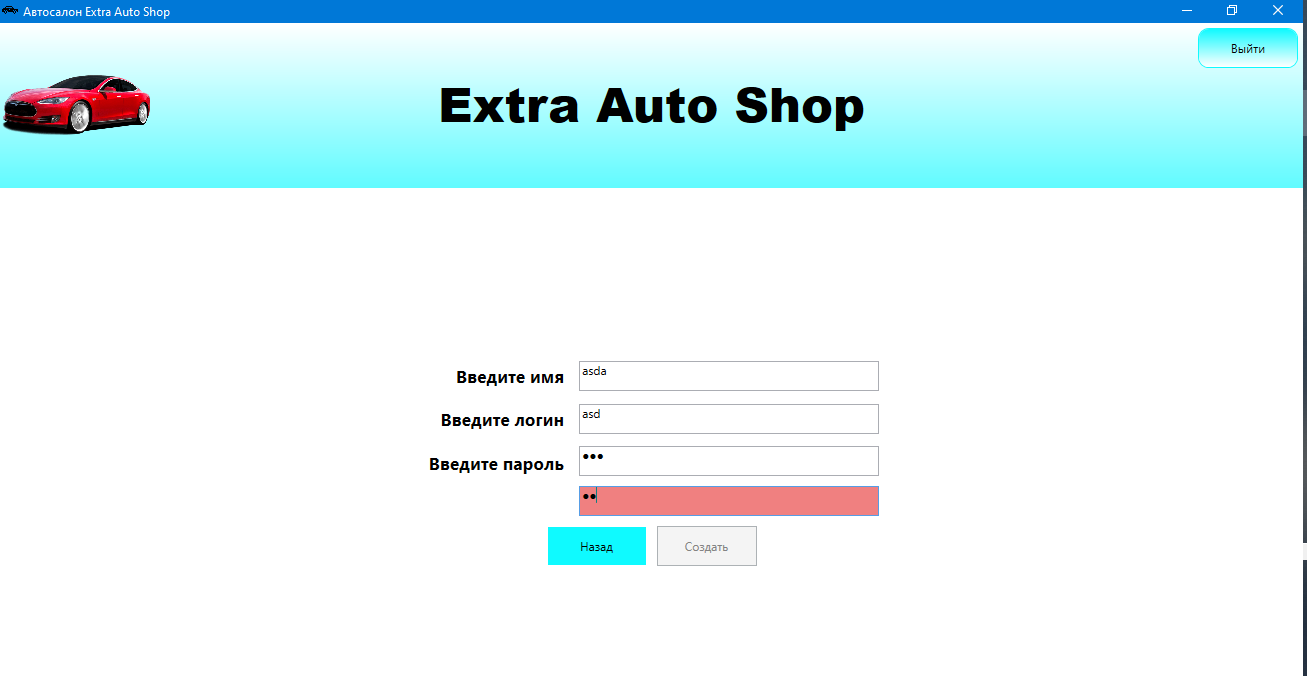


Рисунок 2.19 – Неверные данные в окне регистрации

Если пользователь зайдет как «User», то перед ним появится таблица со списком (рисунок 2.20)



Рисунок 2.20 – Таблица, в ней «User» просматривает список автомобилей.

Здесь же он может оставить здесь заявку, заполнив данные (рисунок 2.21)

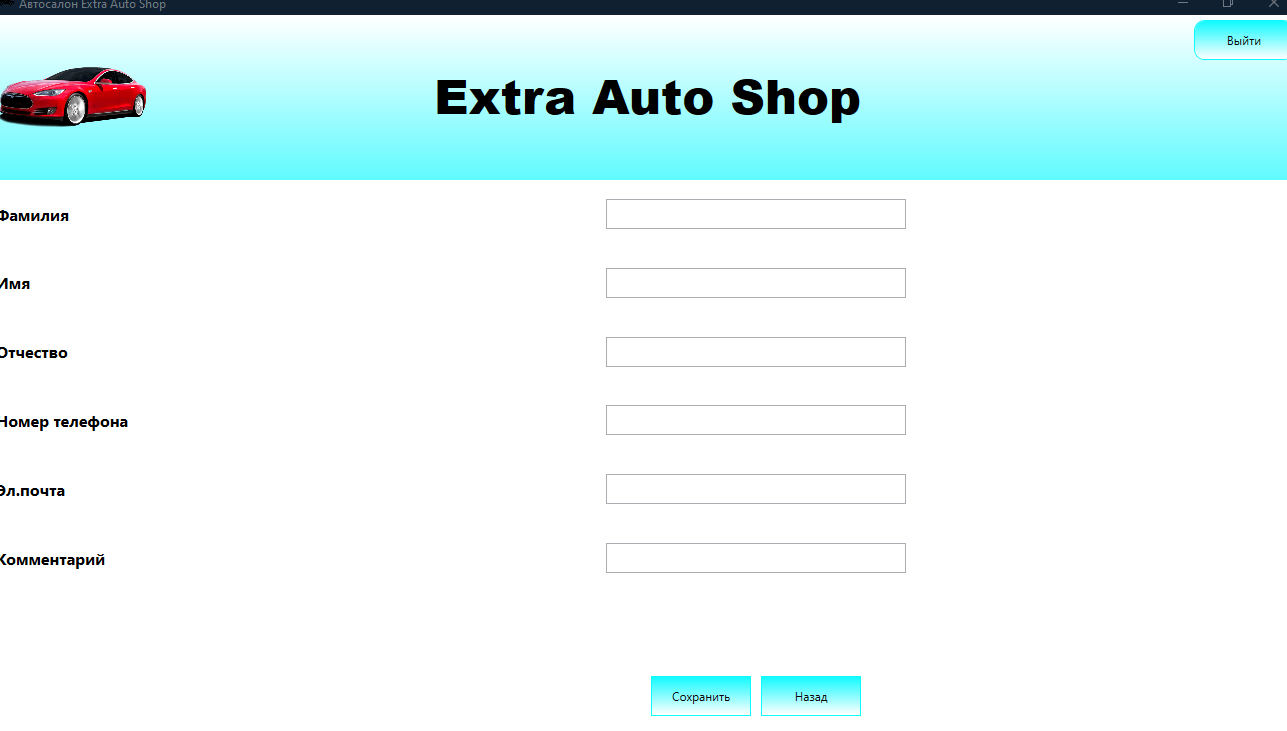


Рисунок 2.21 – Форма заявки для клиента

**2.5 Написание программного кода**

Стартовое окно «MainWindow»

Стартовая окно, через которую происходит вход в нашу программу. Обработчик событий для данного окна выглядит следующим образом (MainWindow.cs):

public MainWindow() { InitializeComponent(); AppConnect.modelOdb = new diplomEntities(); AppFrame.FrameMain = myFrame; myFrame.Navigate(new PageLogin()); } private void btnExit\_Click(object sender, RoutedEventArgs e) { Close(); }

В данном обработчике, настраиваем страницу, которая будет выводится при запуске. В данном случае это страница «PageLogin».

Класс «AppFrame.cs»

Данный класс предназначен для вывода страницы «PageLogin» на главное окно «MainFindow»

Обработчик событий для этого класса выглядит так:

using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Controls; namespace WPF\_DIPLOM.AplicationData { internal class AppFrame { public static Frame FrameMain; }

Страница «PageLogin»

Страница авторизации. Авторизация предоставляется для доступа к данным и выполнения различных действий. Управление ей осуществляется с помощью ролей базы данных и явных разрешений. Авторизация необходима для разрешений, которые назначены пользователю и которые определяют, какие действия доступны этому пользователю. Внешний вид окна регистрации в форме кода, будет выглядеть следующим образом (PageLogin xaml)

<StackPanel Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Center"> <Label Content="Введите логин" Style="{StaticResource Label}"/> <TextBox Text="" Style="{StaticResource TextBox}" x:Name="txbLogin"/> <Label Content="Введите пароль" Style="{StaticResource Label}"/> <PasswordBox Style="{StaticResource PasswordBox}" x:Name="psbPassword"/> <StackPanel Orientation="Horizontal" HorizontalAlignment="Center"> <Button Content="Регистрация" Style="{StaticResource ButtonY}" Click="Button\_Click"/> <Button Content="Войти" Style="{StaticResource ButtonG}" Click="Button\_Click\_1"/> </StackPanel> </StackPanel>

Обработчик событий для окна регистрации выглядит следующим образом (PageLogin.cs):

private void Button\_Click\_1(object sender, RoutedEventArgs e) { try { var userObj = AppConnect.modelOdb.Users.FirstOrDefault(x => x.Login == txbLogin.Text && x.Password == psbPassword.Password); if (userObj == null) { MessageBox.Show("Данные указаны не верно!", "Ошибка авторизации", MessageBoxButton.OK, MessageBoxImage.Error); } else { switch (userObj.IdRole) { case true: MessageBox.Show("Здраствуйте, Администратор " + userObj.Name + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information); NavigationService.Navigate(new PagesList()); break; case false: MessageBox.Show("Здраствуйте, Пользователь " + userObj.Name + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information); NavigationService.Navigate(new PageUser()); break; } } } catch (Exception Ex) { MessageBox.Show("Ошибка" + Ex.Message.ToString() + "Критическая ошибка приложения", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning); } }

Страница «PageList»

Страница для администратора. На этой странице он может выбирать любую таблицу и работать с информацией в этой таблице. Внешний вид этой странице выглядит следующим образом (PageList xaml):

<Grid.RowDefinitions> <RowDefinition Height="10"/> <RowDefinition Height="193\*"/> <RowDefinition Height="27\*"/> </Grid.RowDefinitions> <StackPanel Grid.Row="1" Grid.RowSpan="2"> <TextBlock Style="{StaticResource TextBlock}" Text="Выберите таблицу для просмотра" HorizontalAlignment="Center" FontSize="20"/> <Button Style="{StaticResource ButtonG}" Content="Автомобили" HorizontalAlignment="Center" Click="Button\_Click"/> <Button Style="{StaticResource ButtonG}" Content="Аксессуары" HorizontalAlignment="Center" Click="Button\_Click\_1"/> <Button Style="{StaticResource ButtonG}" Content="Клиенты" HorizontalAlignment="Center" Click="Button\_Click\_2"/> </StackPanel> <Button Grid.Row="2" x:Name="btExit" Style="{StaticResource ButtonY}" Content="Выход" HorizontalAlignment="Center" Click="btExit\_Click"/>

Обработчик события для страницы администратора будет выглядеть следующим образом (PageList.cs):

public partial class PagesList : Page { public PagesList() { InitializeComponent(); } private void Button\_Click(object sender, RoutedEventArgs e) { NavigationService.Navigate(new PageAdmin()); } private void Button\_Click\_1(object sender, RoutedEventArgs e) { NavigationService.Navigate(new PageItems()); } private void btExit\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PageLogin()); } private void Button\_Click\_2(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PageClient()); } }

Страница «PageUser»

Страница для пользователя. На этой странице он может просмотреть данные из таблицы с автомобилями и оформить заявку.

Обработчик события выглядит следующим образом:

public partial class PageUser : Page { public PageUser() { InitializeComponent(); DtGridTovar.ItemsSource = diplomEntities.GetContext().Cars.ToList(); } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PageLogin()); } private void btnAdd\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PageClientAdd(null)); }

**2.6 Защита информации**

Базы данных (БД) до сих пор является основным решением для хранения информации. При всем своем удобстве, практичности использования БД отличаются немалым числом уязвимостей, которые требуют привлечения дополнительных решений для обеспечения безопасности и защиты базы данных. Утечка информации считаются одной из главных угроз стабильности, конкурентоспособности компании.

Под угрозой безопасности понимают все способы вторжения, провоцирующие причинение ущерба информации (искажение или разрушение), хищение информационных ресурсов, в том числе с целью сбыта. Возможны как случайные, так и преднамеренные угрозы.

Под случайными понимаются технические сбои, не зависящие от пользователя, – ошибки, повреждения компьютеров или носителей, природные бедствия. Ошибки по невниманию, как правило, связаны с нарушением процесса выполнения программ, и происходят чаще всего по вине сотрудников.

Преднамеренные угрозы зачастую реализуют внешние источники вторжения, то есть не имеющие отношения к организации – источнику информации. Таких «специалистов» может заинтересовать как программное обеспечение, так и обрабатываемые им сведения. Причем цели присвоения ресурсов могут быть разными: от банальной наживы до саботажа и дискриминации всего предприятия и впоследствии – разлада рабочего процесса.

Что касается сохранности содержимого баз, стоит выделить два основных подхода, которые призваны обеспечить безопасность данных:

1. Избирательное управление данными. Достаточно гибкий способ, при использовании которого разные контрагенты могут быть наделены различными полномочиями по отношению к одному и тому же элементу.

2. Обязательное управление данными. Этот способ имеет обратную схему действия, то есть база данных и все объекты библиотек распределены по уровням секретности, которые подразумевают допуск к просмотру и модификации пользователей с соответствующим уровнем доступа или выше. Дабы не нарушать всю структуру секретности, способ не дает возможности субъекту с наивысшей классификацией полномочий вносить данные в категории низшей секретности.

На обеспечение целостности и конфиденциальности сведений предприятия и средств их хранения направлена определенная система действий, исключающая несанкционированные манипуляции с информацией. Защита является надежной, если допуск в хранилище осуществляется через многоступенчатую проверку прав доступа и при этом имеется достаточный набор программных средств. В идеале система должна фиксировать любые изменения, связанные с безопасностью.

Первостепенные критерии надежности:

Политика безопасности данных. Направлена на мониторинг потенциальных угроз и поиск средств для их устранения. Содержит правила и нормы пользования содержимым хранилища, свойственные конкретной организации. На основании редакции политики безопасности определяются отдельные механизмы организации защиты системы.

Гарантированность данных. Предполагает оправданность выбранных программных средств. Эта категория воплощает те механизмы, которые дают ход политике безопасности конкретной системы.

Вычислительная часть системы управления безопасностью контролирует допуск контрагентов к тем или иным операциям с данными, находящимися под защитой. Функции мониторинга при обращении субъекта соотносят допустимые для него действия.

Наиболее распространенной проблемой безопасности хранилища является несанкционированный распределенный доступ к ПК, поэтому защита персонального рабочего места – отдельный важный аспект данной статьи. К слову, изначальное назначение персонального компьютера не было ориентировано на процесс защиты информации, а, скорее, на личное пользование. Риск подобной угрозы возрастает именно для содержимого ПК по следующим причинам:

Доступность ПК для посторонних. Как известно, не все владельцы конфиденциальных сведений склонны оберегать свое рабочее место. Зачастую, отсутствуя, они оставляют систему работающей, чем легко воспользоваться для извлечения конфиденциальной информации или ее повреждения.

Коллективные средства. Во многих организациях программными инструментами пользуются на корпоративной основе. Данный фактор позволяет осуществить доступ, не подразумевающий ответственности конкретного пользователя, что ведет к безнаказанности нарушителя.

Стационарные накопители также являются рискованным решением, так как сохраняют весь объем сведений даже вне питания.

Стоит ознакомиться с основными факторами, повышающими риск похищения или уничтожения сведений, циркулирующих в библиотеках информации:

увеличение объема баз при помощи автоматизированного учета;

«разнокалиберные» данные для разного использования, собранные в одной библиотеке;

недостаточный контроль за расширением круга лиц, имеющих доступ;

недостаточный контроль механизмов дистанционного обмена информацией.

Самые распространенные уязвимости баз данных:

Любая БД обрастает во время использования дополнительными уязвимыми местами, которые способствуют обходу защиты базы данных, неправомерному использованию и утечке данных. Большинство таких инцидентов связано с поверхностным или полностью отсутствующим обслуживанием базы и внесением своевременных изменений во время эксплуатации сведений. Типичными уязвимостями БД считаются:

Присутствие в системе пользователей с завышенными или неиспользованными полномочиями. Такая ситуация чаще всего возникает при отсутствии контроля за отдельными сотрудниками обладающими неограниченными правами. Они могут менять и изменять отдельные права и полномочия других сотрудников или злоупотреблять собственными, что способно привести к инцидентам безопасности, снизить уровень защиты информации в базах данных или создать двусмысленные ситуации. Также нередки случаи, когда происходит накопление неиспользованных прав по причине увольнения персонала, перевода в другой отдел или на новую должность. Такие неиспользованные права представляют повышенную угрозу с точки зрения кибератак и интереса со стороны злоумышленников или бывших сотрудников.

Вводные инъекции. Подразумевают взлом через поле ввода путем интеграции вредоносного кода. На практике встречаются инъекции типов SQL и NoSQL. Первые характерны для классических СУБД, вторые затрагивают платформы на основе Big Data. SQL-инъекции более массовые и распространенные ввиду огромного числа СУБД упрощённого типа. NoSQL-инъекции носят более сложный и направленный характер, т.к. проникают через определенные входные точки или отдельные компоненты по типу Hive или MapReduce. В обоих случаях при взломе БД через вводные инъекции злоумышленники получают практически неограниченные возможности для манипуляций данными.

Хакерские атаки и вредоносное ПО. Атаки киберпреступников становятся все более сложными и изощрёнными: фишинг, подмена данных, вредоносные ссылки, коды. Многие из них носят замаскированный и неявный характер, что вводит в заблуждение пользователей, слабо ознакомленных с политиками и правилами информационной безопасности. Тем самым риски хакерских угроз носят непредсказуемый характер, способны привести к многочисленным инцидентам безопасности и самым тяжёлым последствиям для компании, если технология защиты баз данных отличается несовершенством.

Слабая защищенность носителей информации. Сами БД и их резервные копии зачастую хранятся на незащищенных носителях вроде флэш-карт или обычных жестких дисках. Такие данные легко выкрасть или подменить. Дополнительные риски создает отсутствие или игнорирование проведения аудита и мониторинга деятельности администраторов наделенным низкоуровневым доступом к сведениям повышенной важности.

Использование старых версий или неудачных конфигураций БД. Главной проблемой здесь являются морально устаревшие БД, которые эксплуатируются в виде как есть. Кроме того, отсутствуют регулярные обновления и патчи. Такой подход к обслуживанию информации делает ее легкодоступной и накапливает уязвимости со временем. Главными причинами такого отношения к БД и накоплению проблем являются необходимость отключения информации от рабочего процесса и длительность обновления.

Ввиду многообразия внешних угроз и внутренних уязвимостей защитные меры проводятся комплексно и направлено. Широко используемыми вариантами защиты БД считаются:

Шифрование. Применяется с целью подмены или преобразования исходных данных или отдельных информационных блоков, что затрудняет правильную идентификацию сведений и построение их в единое целое. Также в качестве дополнительной меры используют ключи шифрования без которых представляется невозможным использование украденной информации. Шифрование может быть симметричным и асимметричным, на уровне базы, отдельных приложений. Достоинством шифрования считается возможность создать защиту одновременно на нескольких уровнях: информация, алгоритмы, функционал. У такой защиты присутствует обратная сторона - усложнение управлением БД, увеличение сроков обработки сведений из-за необходимости проведения дешифровки.

Введение доступа через авторизацию. Способ, который используют, чтобы отсечь неавторизованный доступ пользователей. Лучше всего работает в паре с разграничением доступа. В таком случае важные данные защищены вдвойне: с одной стороны, правами доступа, с другой необходимостью авторизации. Применяется в тех случаях, когда с БД работает группа людей. Пароль доказал свою надёжность как инструмент защиты сведений, особенно если комбинация цифр, букв, знаков носит сложный характер и регулярно меняется. Подобрать пароль сложно, потому что он находится в зашифрованной базе данных и к нему нет прямого доступа извне.

Разграничение прав доступа между пользователями. Позволяет организовать контролируемый доступ к информации на основании статуса пользователя, его прав. Это удобно с точки зрения контроля конкретных пользователей. Благодаря такому подходу возможно назначить такие полномочия как просмотр, редактирование, использование баз данных. Тем самым одним информационным ресурсом одновременно пользуются разные пользователи, каждый на своем уровне. Такая защита информации в базах данных удобна своей гибкостью в использовании, т.к. позволяет хранить в одном месте сведения с разным уровнем доступа и важности, скрыть отдельные разделы от тех, кто не должен ими пользоваться, создать широкий круг лиц, могущих обращаться к информационному ресурсу по мере надобности.

Введение избирательного или общего доступа к БД. Создаёт условия, в которых информацией пользуется группа людей, объединенных общими интересами и обязанностями или отдельные персоны, обладающие привилегированными правами. Избирательный доступ может быть настроен исходя из групповых или индивидуальных разрешений доступа и уровня ценности сведений. Определяется изначально на внутреннем уровне путем создания дополнительных защитных мер или введения режима коммерческой тайны. Или возможен вариант введения многоуровневой аутентификации с отслеживанием действий пользователей. Реализуется посредством использования DLP-систем. Также действенной мерой в ходе использования избирательного доступа считается предварительная маркировка информации по типу "для общего пользования", "секретно", "коммерческая тайна". Аналогичным образом оформляются пропуски допуска или учетные записи пользователей.

Рассмотрев виды защиты информации в базе данных и сделав анализ своей предметной области, было принято решение использовать такую защиту как: Введение доступа через авторизацию, потому что это одна из самых эффективных и надежных способов защиты информации.

**2.7 Тестирование**

Тестирование — это проверка соответствия между реальным поведением программы и ее ожидаемым поведением. В широком смысле слова тестирование – это одна из техник контроля качества, включающая активности по планированию работ, разработке тестов

Модульное тестирование — это процесс тестирования, который позволяет проверить отдельные модули программного обеспечения направленный на оценку правильности работы. Это один из самых распространенных методов тестирования и является неотъемлемой частью процесса разработки программного обеспечения. Тесты помогают увидеть сильные и слабые стороны программы, найти причины ошибок и наметить пути их устранения.

Цель тестирования — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы. Тестирование проводится, потому что оно может проверить соответствие требованиям.

Тестирование программы полезно тем, что:

Улучшает качество кода. Программный продукт менее подвержен сбоям в работе поскольку тесты помогает написать более надёжный и хороший код, который менее подвержен ошибкам. Делает рефакторинг простым и безопасным. Создание программного обеспечения — это итеративный процесс.

Таким образом, тестирование программного обеспечения помогает повысить эффективность, стабильность, удобство использования и безопасность вашего IT-продукта. Оно также необходимо для измерения производительности программного решения.

Листинг теста программы:

namespace TestProject

{

[TestClass]

public class UnitTest1

{

[TestMethod()]

public void loginTest()

{

string psbPassword = "";

string txbLogin = "";

Assert.AreEqual(psbPassword, txbLogin);

}

}

}

Этот тест позволяет проверить «TextBox» и «PasswordBox» на отсутствие каких-либо данных или символов.

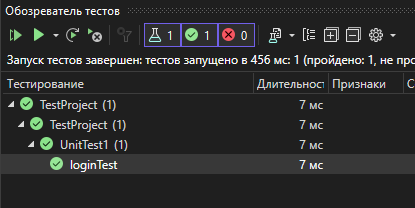


Рисунок 2.22 – Результат тестирование

# Заключение

В ходе дипломного проекта было разработано приложение для автосалона, позволяющий повысить эффективность выполнения работ, связанный с деятельностью данного предприятия.

Также закреплены теоретические знания и практические навыки работы с Microsoft SQL Server и Visual Studio.

Мною были поставлены задачи, которые успешно выполнены, а именно:

1. Описать предметную область;

2. Составить инфологическую модель предметной области;

3. Составить модель предметной области в стандарте IDEF0;

4. Составить DFD диаграмму предметной области;

5. Создать прототипы интерфейса;

6. Произвести обзор инструментальных средств создания продукта;

7. Обосновать выбор средств реализации ИС;

8. Описать логику функционирования ПО;

9. Реализовать интерфейсы ПО;

10. Написать программный код;

11. Описать механизмы защиты информации;

12. Произвести тестирование готового продукта.

Работа выложена на платформе GitHub <https://github.com/NyTipoMorozov/WPF_DIPLOM_MOROZOV.git>

# Список использованных источников

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд Язык программирования C#. Классика Computers Science. 4-е издание = C# Programming Language (Covering C# 4.0), 4th Ed. -- СПб.: «Питер», 2012. -- 784 с. -- ISBN 978-5-459-00283-6. URL: <https://urait.ru/bcode/492224>.

2. Шатрова, Т. И. Языковая игра в текстах комической направленности: процессы кодирования и декодирования [Текст]: канд. филол. наук / Т. И. Шатрова; Тульский гос. пед. ун-т им. Л. Н. Толстого. - Белгород, 2006. - 159 с. URL: <https://studbooks.net/2250190/informatika/zaklyuchenie>

3. Васильков, А. В. Информационные системы и их безопасность: Учебное пособие [Текст] / А. В. Васильков, А. А. Васильков, И. А. Васильков. - М.: Форум, 2023. - 528 c. 72

4. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем: Учебник для студентов экономических вузов, обучающихся по спец. "Прикладная информатика (по областям)" и "Прикладная математика и информатика" [Текст] / А. М. Вендров. - М.: Финансы и статистика, 2023.- 544 с.

5. Голицына, О.Л. Базы данных. Учебное пособие [Текст] / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – М.: Форму-Инфра-М, 2023. – 430 с.

6. Грекул, В. И. Проектирование информационных систем [Текст] / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. - М.: Интернет-университет информационных технологий, 2023. – 420 с.

7. Гришин, А. В. Промышленные информационные системы и сети: практическое руководство [Текст] / А. В. Гришин. - М.: Радио и связь, 2023. - 176 c

8. Гусятников, В.Н. Стандартизация и разработка программных систем [Текст] / В. Н. Гусятников, А. И. Безруков. – М.: Финансы и статистика, 2023. – 288с.

9. Дубейковский, В. И. Практика функционального моделирования с AllFusion Process Modeler 4.1. (BPwin) Где? Зачем? Как? [Текст] / В. И. Дубейковский. – М. : ДИАЛОГ-МИФИ, 2023. – 464 с

10. Душин, В. К.Теоретические основы информационных процессов и систем [Текст] / В. К. Душин – М.: Дашков и К, 2023. -350 с.

11. Емельянова, Н. З. Основы построения автоматизированных информационных систем. Учебное пособие [Текст] / Н. З. Емельянова, Т. Л. Партыка, И. И. Попов. – М.: Academia, 2023 – 220 с.

12. Ивашкевич, В. Б. Управленческий учет в информационной системе предприятия [Текст] / Н. З. Емельянова. - . – М. : ДИАЛОГ-МИФИ, 2023. – 102 с.

13. Исаев, Г. Н. Информационные системы в экономике. [Текст] / Г. Н. Исаев – М.: Омега-Л, 2023 - .430 с. 73

14. Йордан, Э. Объектно-ориентированный анализ и проектирование систем [Текст] / Э. Йордан, С. Аргила. - М.: ЛОРИ, 2023. - 264 с."

15. Карминский, A.M., Черников Б.В. Информационные системы в экономике. Методология создания. [Текст] / А. М. Карминский, Б. В. Черников. – М.: Финансы и статистика, 2023 – 330 с.

16. Карпова, Т.П. Управленческий учет: Учебник для вузов. - М.: Аудит, ЮНИТИ, 2023. - 237с

17. Когаловский, М. Р. Технология баз данных на персональных ЭВМ. [Текст] / М. Р. Когаловский. - М.:Финансы и статистика, 2023 г.- 123 с."

18. Кукукина, И. Г. Автоматизация управленческого учета на предприятиях. [Текст] / М.: Омега-Л, 2023 – 240 с.

19. Муромцев, В. В. Проектирование информационных систем: Учебное пособие для студентов вузов заочной формы обучения по спец. 010502 "Прикладная информатика в экономике". [Текст] / В. В. Муромцев – г. Белгород:БелГУ, 2023.-160 с.

20. Нигматулин, Г. Т. Программный комплекс «Сводная отчетность» [Электронный ресурс] / Г.Т. Нигматулин - НПО Компьютер: Информационные системы. – Режим доступа: <http://www.npo-comp.ru/1517366.aspx>.

21. Норенков, И. П. Автоматизированные информационные системы: Учебное пособие . [Текст] / И. П. Норенков. - М.: МГТУ им. Баумана, 2023. - 342 c.

22.Сысолетин, Е. Г. Разработка интернет-приложений: учебное пособие для среднего профессионального образования / Е. Г. Сысолетин, С. Д. Ростунцев. — Москва: Издательство Юрайт, 2023. — 90 с. — (Профессиональное образование). — ISBN 978-5-534-10015-0. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/517538> (дата обращения: 28.02.2023).

23. Полуэктова, Н. Р. Разработка веб-приложений: учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — Москва: Издательство Юрайт, 2023. — 204 с. — (Профессиональное образование). — ISBN 978-5-534-14744-5. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/519727> (дата обращения: 28.05.2023).

24 .Полуэктова, Н. Р. Разработка веб-приложений: учебное пособие для вузов / Н. Р. Полуэктова. — Москва: Издательство Юрайт, 2023. — 204 с. — (Высшее образование). — ISBN 978-5-534-13715-6. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/519714> (дата обращения: 28.05.2023).

25. Кукукина, И. Г. Автоматизация управленческого учета на предприятиях. [Текст] / М.: Омега-Л, 2023 – 240 с.

26. Муромцев, В. В. Проектирование информационных систем: Учебное пособие для студентов вузов заочной формы обучения по спец. 010502 "Прикладная информатика в экономике". [Текст] / В. В. Муромцев – г. Белгород:БелГУ, 2023.-160 с.

27. Нигматулин, Г. Т. Программный комплекс «Сводная отчетность» [Электронный ресурс] / Г.Т. Нигматулин - НПО Компьютер: Информационные системы. – Режим доступа: <http://www.npo-comp.ru/1517366.aspx>.

28. Норенков, И. П. Автоматизированные информационные системы: Учебное пособие . [Текст] / И. П. Норенков. - М.: МГТУ им. Баумана, 2023. - 342 c.

29. Олейников, А. Е. Автоматизированные информационные технологии. [Текст] / А. Е. Олейников. - М.: Феникс, 2023 – 420 с.

30 Осипов, В. И. Проблемы автоматизации управленческого учета. [Текст] / В. И. Осипов – М.: Феникс, 2023-76 с.

31 Палий, В. Ф. Управленческий учет - новое прочтение внутрихозяйственного расчета. Бухгалтерский учет. [Текст] / В. Ф. Палий, В. В. Палий – М.: Феникс, 2023 – 62 с. 74

32 Радченко, М. Г. 1С: Предприятие 8.0. Практическое пособие разработчика. Примеры и типовые приемы. [Текст] / М. Г. Радченко. – М.: ООО «1С-Паблишинг», 2023.– 656 с.

33 Рубичев, Н. А. Измерительные информационные системы: Учебное пособие [Текст] / Н. А. Рубичев. - М.: Дрофа, 2023. - 334 c

34 Уткин, В. Б. Информационные системы в экономике: Учебник для студентов высших учебных заведений [Текст] / В. Б. Уткин, К. В. Балдин. - М.: ИЦ Академия, 2023. - 288 c

# Приложение

**MainWindows xaml:**

<Windowx:Class="WPF\_DIPLOM.MainWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:local="clr-namespace:WPF\_DIPLOM" mc:Ignorable="d" Title="Автосалон Extra Auto Shop" MinHeight="400" Height="720" MaxHeight="1000" Width="1280" MinWidth="900" MaxWidth="2000"> <Grid> <Grid.ColumnDefinitions> <ColumnDefinition/> </Grid.ColumnDefinitions> <Grid.RowDefinitions> <RowDefinition Height="151\*"/> <RowDefinition Height="505\*"/> <RowDefinition Height="30"/> </Grid.RowDefinitions> <Label Content="" Grid.Row="0" Grid.Column="0"> <Label.Background> <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0"> <GradientStop Color="White"/> <GradientStop Color="#FF61FBFF" Offset="1"/> </LinearGradientBrush> </Label.Background> </Label> <Label Content="" Grid.Row="2"> </Label> <Label Content="Extra Auto Shop" Grid.Row="0" Grid.Column="0" Background="{x:Null}" VerticalContentAlignment="Center" HorizontalContentAlignment="Center" FontSize="48" FontFamily="Arial Black"/> <StackPanel Orientation="Horizontal"> <Image Source="images/icon.png" Width="150" /> </StackPanel> <Button Content="Выйти" Name="btnExit" Click="btnExit\_Click" Style="{StaticResource ButtonG}" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Right" VerticalAlignment="Top" > <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="10"/> </Style> </Button.Resources> </Button> <Frame x:Name="myFrame" Grid.Row="1" NavigationUIVisibility="Hidden"/> </Grid> </Window>

**MainWindow.cs:**

public partial class MainWindow : Window { public MainWindow() { InitializeComponent(); AppConnect.modelOdb = new diplomEntities(); AppFrame.FrameMain = myFrame; myFrame.Navigate(new PageLogin()); } private void btnExit\_Click(object sender, RoutedEventArgs e) { Close(); } } }

**PageLogin xaml**

<Grid>

<StackPanel Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Center">

<Label Content="Введите логин" Style="{StaticResource Label}"/>

<TextBox Text="" Style="{StaticResource TextBox}" x:Name="txbLogin"/>

<Label Content="Введите пароль" Style="{StaticResource Label}"/>

<PasswordBox Style="{StaticResource PasswordBox}" x:Name="psbPassword"/>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<Button Content="Регистрация" Style="{StaticResource ButtonY}" Click="Button\_Click"/>

<Button Content="Войти" Style="{StaticResource ButtonG}" Click="Button\_Click\_1"/>

</StackPanel>

</StackPanel>

</Grid>

**PageLogin.cs**

public partial class PageLogin : Page

{ public PageLogin() { InitializeComponent(); } private void Button\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PageСreateAcc()); } private void Button\_Click\_1(object sender, RoutedEventArgs e) { try { var userObj = AppConnect.modelOdb.Users.FirstOrDefault(x => x.Login == txbLogin.Text && x.Password == psbPassword.Password); if (userObj == null) { MessageBox.Show("Данные указаны не верно!", "Ошибка авторизации", MessageBoxButton.OK, MessageBoxImage.Error); } else { switch (userObj.IdRole) { case true: MessageBox.Show("Здраствуйте, Администратор " + userObj.Name + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information); NavigationService.Navigate(new PagesList()); break; case false: MessageBox.Show("Здраствуйте, Пользователь " + userObj.Name + "!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information); NavigationService.Navigate(new PageUser()); break; } } } catch (Exception Ex) { MessageBox.Show("Ошибка" + Ex.Message.ToString() + "Критическая ошибка приложения", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Warning); } } } }

**PageAdmin xaml**

<Grid.RowDefinitions> <RowDefinition Height="22\*"/> <RowDefinition Height="3\*"/> </Grid.RowDefinitions> <DataGrid x:Name="DtGridTovar" AutoGenerateColumns="False" IsReadOnly="True"> <DataGrid.Columns> <DataGridTextColumn Header="Модель автомобиля" Binding="{Binding Model}" Width="38\*" FontSize="20" ></DataGridTextColumn> <DataGridTextColumn Header="Количество" Binding="{Binding Quantity1}" FontSize="20" Width="25\*"/> <DataGridTextColumn Header="Цена" Binding="{Binding Valuta}" FontSize="20" Width="25\*"/> <DataGridTemplateColumn Width="auto"> <DataGridTemplateColumn.CellTemplate> <DataTemplate> <Button Content="Редактировать" Name="BtnEdit" Click="BtnEdit\_Click" Style="{StaticResource ButtonY}"></Button> </DataTemplate> </DataGridTemplateColumn.CellTemplate> </DataGridTemplateColumn> </DataGrid.Columns> </DataGrid> <StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center"> <Button Content="Удалить" Name="btnDel" Click="btnDel\_Click" Style="{StaticResource ButtonG}"/> <Button Content="Добавить" Name="btnAdd" Click="btnAdd\_Click" Style="{StaticResource ButtonG}"/> <Button Content="Назад" Name="btnBack" Click="btnBack\_Click" Style="{StaticResource ButtonG}"/> </StackPanel>

**PageAdmin.cs**

public partial class PageAdmin : Page { public PageAdmin() { InitializeComponent(); DtGridTovar.ItemsSource = diplomEntities.GetContext().Cars.ToList(); } private void btnAdd\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageCarsAdd(null)); } private void btnDel\_Click(object sender, RoutedEventArgs e) { var tovarForRemoning = DtGridTovar.SelectedItems.Cast<Cars>().ToList(); if (MessageBox.Show($"Вы точно хотите удалить следующее {tovarForRemoning.Count()} элементов?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes) { try { diplomEntities.GetContext().Cars.RemoveRange(tovarForRemoning); diplomEntities.GetContext().SaveChanges(); MessageBox.Show("Данные удалены"); DtGridTovar.ItemsSource = diplomEntities.GetContext().Cars.ToList(); } catch (Exception ex) { MessageBox.Show(ex.Message.ToString()); } } } private void BtnEdit\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageCarsAdd((sender as Button).DataContext as Cars)); } private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e) { if (Visibility == Visibility.Visible) { diplomEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload()); DtGridTovar.ItemsSource = diplomEntities.GetContext().Cars.ToList(); } } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PagesList()); } }

**PageCarsAdd xaml**

<Grid> <Grid.ColumnDefinitions> <ColumnDefinition Width="1\*"></ColumnDefinition> <ColumnDefinition Width="5\*"></ColumnDefinition> </Grid.ColumnDefinitions> <Grid.RowDefinitions> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> </Grid.RowDefinitions> <TextBlock Text="Модель" Grid.Row="0" Style="{StaticResource TextBlock}"/> <TextBlock Text="Количество" Grid.Row="1" Style="{StaticResource TextBlock}"/> <TextBlock Text="Цена" Grid.Row="2" Style="{StaticResource TextBlock}"/> <TextBox Grid.Row="0" Grid.Column="1" Style="{StaticResource TextBox}" MaxLength="50" Text="{Binding Model}"/> <TextBox Grid.Row="1" Grid.Column="1" Style="{StaticResource TextBox}" Text="{Binding Quantity}"/> <TextBox Grid.Row="2" Grid.Column="1" Style="{StaticResource TextBox}" Text="{Binding Price}"/> <StackPanel Grid.Row="5" Grid.Column="1" HorizontalAlignment="Center" Orientation="Horizontal"> <Button Content="Сохранить" Name="btnSave\_Click" Style="{StaticResource ButtonG}" Click="btnSave\_Click\_1"/> <Button Content="Назад" Name="btnBack\_Click" Style="{StaticResource ButtonG}" Click="btnBack\_Click\_1" HorizontalAlignment="Right"/> </StackPanel> </Grid>

**PageCarsAdd.cs**

private Cars \_currentCars = new Cars(); public PageCarsAdd(Cars selectedCars) { InitializeComponent(); if (selectedCars != null) \_currentCars = selectedCars; DataContext = \_currentCars; } private void btnSave\_Click\_1(object sender, RoutedEventArgs e) { StringBuilder errors = new StringBuilder(); if (string.IsNullOrWhiteSpace(\_currentCars.Model)) errors.AppendLine("Укажите название модели"); if (\_currentCars.Quantity <= 0) errors.AppendLine("Количество не может быть меньше или равно 0"); if (\_currentCars.Price <= 0) errors.AppendLine("Цена не может быть меньше или равно 0"); if (errors.Length > 0) { MessageBox.Show(errors.ToString()); return; } if (\_currentCars.CarID == 0) diplomEntities.GetContext().Cars.Add(\_currentCars); try { diplomEntities.GetContext().SaveChanges(); MessageBox.Show("Информация сохранена"); AppFrame.FrameMain.GoBack(); } catch (Exception ex) { MessageBox.Show(ex.Message.ToString()); } } private void btnBack\_Click\_1(object sender, RoutedEventArgs e) { AppFrame.FrameMain.GoBack(); }

**PageClient xaml**

<Grid.RowDefinitions> <RowDefinition Height="22\*"/> <RowDefinition Height="3\*"/> </Grid.RowDefinitions> <DataGrid x:Name="DtGridTovar" AutoGenerateColumns="False" IsReadOnly="True"> <DataGrid.Columns> <DataGridTextColumn Header="Фамилия" Binding="{Binding SecondName}" Width="15\*" FontSize="20"></DataGridTextColumn> <DataGridTextColumn Header="Имя" Binding="{Binding Name}" Width="15\*" FontSize="20"/> <DataGridTextColumn Header="Отчество" Binding="{Binding ThirdName}" Width="25\*" FontSize="20"/> <DataGridTextColumn Header="Телефон" Binding="{Binding PhoneNumber}" Width="25\*" FontSize="20"/> <DataGridTextColumn Header="Эл.почта" Binding="{Binding email}" Width="25\*" FontSize="20"/> <DataGridTextColumn Header="Комментарий" Binding="{Binding Comment}" Width="60\*"/> <DataGridTemplateColumn Width="auto"> <DataGridTemplateColumn.CellTemplate> <DataTemplate> <Button Content="Редактировать" x:Name="BtnEdit" Click="BtnEdit\_Click" Style="{StaticResource ButtonY}" Height="100"></Button> </DataTemplate> </DataGridTemplateColumn.CellTemplate> </DataGridTemplateColumn> </DataGrid.Columns> </DataGrid> <StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center"> <Button Content="Удалить" x:Name="btnDel" Click="btnDel\_Click" Style="{StaticResource ButtonG}"/> <Button Content="Добавить" x:Name="btnAdd" Click="btnAdd\_Click" Style="{StaticResource ButtonG}"/> <Button Content="Назад" x:Name="btnBack" Click="btnBack\_Click" Style="{StaticResource ButtonG}"/> </StackPanel>

**PageClient.cs**

public PageClient() { InitializeComponent(); DtGridTovar.ItemsSource = diplomEntities.GetContext().Client.ToList(); } private void BtnEdit\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageClientAdd((sender as Button).DataContext as Client)); } private void btnDel\_Click(object sender, RoutedEventArgs e) { var tovarForRemoning = DtGridTovar.SelectedItems.Cast<Client>().ToList(); if (MessageBox.Show($"Вы точно хотите удалить следующее {tovarForRemoning.Count()} элементов?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes) { try { diplomEntities.GetContext().Client.RemoveRange(tovarForRemoning); diplomEntities.GetContext().SaveChanges(); MessageBox.Show("Данные удалены"); DtGridTovar.ItemsSource = diplomEntities.GetContext().Client.ToList(); } catch (Exception ex) { MessageBox.Show(ex.Message.ToString()); } } } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageMain.PagesList()); } private void btnAdd\_Click(object sender, RoutedEventArgs e) { AppFrame.FrameMain.Navigate(new PageClientAdd(null)); } private void Page\_IsVisibleChanged\_1(object sender, DependencyPropertyChangedEventArgs e) { if (Visibility == Visibility.Visible) { diplomEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload()); DtGridTovar.ItemsSource = diplomEntities.GetContext().Client.ToList(); } } }