СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc152791695)

[1 ЦЕЛЬ И ЗАДАЧИ УЧЕБНОЙ ПРАКТИКИ 4](#_Toc152791696)

[1.1 Основные требования по технике безопасности во время прохождения учебной практики 5](#_Toc152791697)

[1.2 Нормативно-правовые документы 6](#_Toc152791698)

[2 ПРЕДМЕТНАЯ ОБЛАСТЬ. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ 8](#_Toc152791699)

[2.1 Описание предметной области 8](#_Toc152791700)

[2.2 Группы пользователей информационной системы 8](#_Toc152791701)

[2.3 Основные требования, предъявляемые к информационной системе 9](#_Toc152791702)

[3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 11](#_Toc152791703)

[3.1 Моделирование некоторых динамических аспектов системы 11](#_Toc152791704)

[3.1.1 Создание диаграммы вариантов использования 12](#_Toc152791705)

[3.1.2 Создание диаграммы деятельности 12](#_Toc152791706)

[3.2 Разработка функциональной структуры информационной системы. Карта навигации по информационной системе 16](#_Toc152791707)

[3.3 Разработка макета информационной системы 17](#_Toc152791708)

[4 РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ 23](#_Toc152791709)

[4.1 Моделирование и разработка базы данных 23](#_Toc152791710)

[4.2 Разработка программных модулей информационной системы 27](#_Toc152791711)

[5 РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ 70](#_Toc152791712)

[5.1 Назначение эксперимента. Выбор и обоснование методики проведения тестирования 70](#_Toc152791713)

[5.2 Технология тестирования 71](#_Toc152791714)

[5.3 Результаты проведения тестирования 72](#_Toc152791715)

[ЗАКЛЮЧЕНИЕ 76](#_Toc152791716)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 77](#_Toc152791717)

[ПРИЛОЖЕНИЕ А 79](#_Toc152791718)

[ПРИЛОЖЕНИЯ Б 83](#_Toc152791719)

# ВВЕДЕНИЕ

В современном автомобильном рынке эффективное управление деятельностью предприятий, занимающихся продажей и обслуживанием автомобилей, является неотъемлемой частью успешного функционирования. Уникальные особенности бизнеса компании LADA-сервис, специализирующейся на реализации автомобилей марки ВАЗ, предполагают комплексный подход к разработке прикладного программного обеспечения.

LADA-сервис — крупная корпорация с филиалами в различных районах нашего города, предоставляющая широкий спектр услуг по продаже автомобилей как собственного производства, так и поддержанных моделей. Отличительной особенностью является гибкий подход к заказам, где покупатель имеет возможность выбора не только модели и цвета, но и дополнительных опций, включая тюнинг, а также установку индивидуальных параметров и оговорку сроков поставки.

Акцент делается на создании информационной системы, способной эффективно управлять процессами продаж, учета подержанных автомобилей, обработки заказов и взаимодействия с клиентами, включая как физические лица, так и организации. Разработанные модули программного обеспечения направлены на повышение оперативности и точности управленческих решений, а также на обеспечение прозрачности и надежности финансовых транзакций.

В рамках модуля ПМ 01 Разработка модулей программного обеспечения для компьютерных систем, будет разработана и спроектирована информационная система “Автосервис”, которая будет включать в себя несколько видов профессиональных компетенций.

# ЦЕЛЬ И ЗАДАЧИ УЧЕБНОЙ ПРАКТИКИ

Целью является создание информационной системы, способствующей оптимизации процессов управления автомобильным ассортиментом компании, а также в улучшении информационной базы для покупателей.

Задачи:

* Закрепление и углубление знаний и умений;
* Формирование необходимых навыков профессиональной компетенции;
* Развить готовность выполнять поставленные профессиональные задачи;
* Разработка программных модулей¸ оптимизированных для работы;
* Тестирование разработанных программных модулей на соответствие функциональным требованиям;

Цель и задачи в совокупности направлены на повышение эффективности обслуживания через современные информационные технологии.

Для достижения цели и задач будет разработана и спроектирована информационная система “Автосервис” нацеленная на оптимизацию процесса внесения и хранения данных в этой предметной области, в рамках учебной практики по модулю ПМ 01 (Разработка модулей программного обеспечения для компьютерных систем), овладевая профессиональными компетенциями и задачами:

* систематизация, обобщение, закрепление и углубление знаний и умений;
* формирование знаний, умений и навыков, профессиональных компетенций, профессионально значимых личностных качеств;
* развитие профессионального интереса, формирование мотивационно-целостного отношения к профессиональной деятельности, готовности к выполнению профессиональных задач в соответствии с нормами морали, профессиональной этики и служебного этикета.

## Основные требования по технике безопасности во время прохождения учебной практики

К работе в компьютерном классе допускаются лица, ознакомленные с данной инструкцией по технике безопасности и правилам поведения. Необходимо неукоснительно соблюдать правила по технике безопасности, т.к. нарушение этих правил может привести к поражению электрическим током, вызвать возгорание и навредить здоровью.

- Работа, обучающегося в компьютерном классе разрешается только в присутствии преподавателя (лаборанта).

- Во время занятий посторонние лица могут находиться в классе только с разрешения преподавателя.

- Во время перемен между уроками проводится обязательное проветривание компьютерного кабинета с обязательным выходом обучающихся из класса.

Каждый обучающийся в ответе за состояние своего рабочего места и сохранность размещенного на нем оборудования.

Обучающийся допустивший нарушение по технике безопасности и правилам поведения в компьютерном классе, может быть привлечен к дисциплинарной ответственности. Если нарушение техники безопасности связано с причинением имущественного ущерба, обучающийся несет и материальную ответственность в установленном законом порядке.

## Нормативно-правовые документы

Оформление и содержание технического задания соответствует требованиям стандарта ГОСТ 19.201-78. ЕСПД. Общие положения.

Техническое задание оформлено в соответствии с ГОСТ 19.106-78 на листах формата 11 и 12 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставлены в верхней части листа над текстом.

Лист утверждения и титульный лист оформлен в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

Для внесения изменений или дополнений в техническое задание на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему.

Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Техническое задание должно содержать следующие разделы:

* введение;
* основания для разработки;
* назначение разработки;
* требования к программе или программному изделию;
* требования к программной документации;
* технико-экономические показатели;
* стадии и этапы разработки;
* порядок контроля и приемки;
* в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

Вывод по главе: были описаны, техника безопасности во время прохождения учебной практики; нормативно-правовые документы, которые отвечают за стандарты разработки; цели и задачи учебной практики, включая вводную часть, где описана тематика отрасли и её тенденции развития.

# ПРЕДМЕТНАЯ ОБЛАСТЬ. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ

Предметная область информационной системы — это материальная система или система, характеризующая элементы материального мира, информация о которой хранится и обрабатывается. Предметная область рассматривается как некоторая совокупность реальных объектов и связей между ними.

## Описание предметной области

Необходимо разработать и спроектировать прикладное программное обеспечение для эффективного управления продажами автомобилей.

Данная компания занимается продажей автомобилей марки ВАЗ в нашем городе. Она имеет несколько филиалов в разных районах. Автомобиль может быть продан как со склада компании, так и на заказ с завода-изготовителя по предоплате. Покупатель может заказать модель, цвет, тюнинг и оговорить срок поставки заказанного автомобиля. Одновременно с новыми авто на площадках компании имеется большой выбор подержанных автомобилей, как отечественных, так и иностранных. Покупателем может быть как физическое лицо, так и организация. В первом случае – расчет наличными, во втором – через банк. Расчет производится в рублях.

## Группы пользователей информационной системы

Администратор - имеет полный доступ ко всей информационной системе. Может авторизоваться, просмотреть или изменить данные во всех таблицах. Также администратор может добавлять, удалять, изменять и просматривать учётные записи пользователей.

Пользователь - имеет ограниченные возможности в использовании информационной системы. Может авторизоваться, а также просматривать и изменять записи в таблицах Автомобиль, Двигатели, Заказы, Клиент, Кузов и Марки.

Покупатель - ограничен в использованиях информационной системы, может только авторизоваться и просматривать данные из таблиц Клиент и Заказы, которые ассоциируются с его профильем, а также оформлять заказы на автомобиль и изменять свои данные в таблице Клиент.

Гость - ограничен в использованиях информационной системы, может только авторизоваться и создавать профиль клиента, чтобы получить доступ к роли покупателя.

## Основные требования, предъявляемые к информационной системе

*Гибкость:*

* Система должна обладать высокой степенью конфигурируемости, чтобы пользователи могли адаптировать её под различные потребности селекционного института;
* Возможность легкого масштабирования системы при увеличении объема данных или числа пользователей.

*Надёжность:*

* Обеспечение бесперебойной работы системы с минимальным количеством сбоев и перерывов;
* Регулярное создание резервных копий данных для предотвращения потери информации в случае сбоев.

*Эффективность:*

* Обеспечение высокой производительности при обработке данных, особенно в условиях одновременного использования множества пользователей
* Обеспечение высокой производительности при обработке данных, особенно в условиях одновременного использования множества пользователей

*Безопасность:*

* Реализация надежной системы аутентификации для проверки личности пользователей и установление строгих прав доступа.
* Использование шифрования для защиты передаваемой и хранимой чувствительной информации.
* Ведение журнала аудита для отслеживания действий пользователей и обнаружения возможных угроз безопасности.

Вывод по главе: было описано, что такое предметная область и её описание; группы пользователей, которые будут пользоваться информационной системой «Продажа сортов семян» и основные требования к ней.

# ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Проектированием информационных систем называется многоступенчатый процесс их создания и/или модернизации путём применения упорядоченной совокупности методологий и инструментария. Проектирование (в отличие от моделирования) предполагает работу с пока несуществующим объектом и направлено на создание информационной системы в области:

* обработки объектов будущей базы данных;
* написания программ (в том числе – отчётных и экранных форм), обеспечивающих выполнение запросов к данным,
* выполнения учёта функционирования конкретной среды (технологии).

Если выделять стадию проектирования информационных систем в качестве отдельного этапа, то его можно разместить между этапами анализа и разработки. Однако на практике чёткое разделение на этапы, как правило, затруднено или невозможно, поскольку проектирование, формально начинаясь с определения цели проекта, часто продолжается на стадиях тестирования и реализации.

## Моделирование некоторых динамических аспектов системы

Диаграммы деятельности применяются для моделирования динамических аспектов поведения системы. Эти динамические аспекты могут включать деятельность на любом уровне абстракции в любом представлении системной архитектуры, включая классы (в том числе активные), интерфейсы, компоненты и узлы.

Использовать диаграммы деятельности для моделирования некоторых динамических аспектов системы допускается в контексте почти любого моделируемого элемента. Однако чаще они применяются в контексте всей системы, подсистемы, операции или класса. Диаграмму деятельности можно присоединить к варианту использования, чтобы моделировать сценарий, и к кооперации, чтобы моделировать динамические аспекты поведения совокупности объектов.

При разработке программного обеспечения заблаговременное планирование и моделирование значительно упрощают программирование. На этапе создания концептуальной модели для описания функционального назначения используется диаграмма вариантов использования, а для описания логики процедур и бизнес-процессов используется диаграмма деятельности.

### Создание диаграммы вариантов использования

В приложении А, на рисунке 1 представлен пример диаграммы вариантов использования информационной системы для продажи автомобилей.

Рисунок 1 – Диаграмма вариантов использования

### Создание диаграммы деятельности

На рисунках 2 – 5 представлена диаграмма деятельности (по ролям) информационной системы «Автосервис».



Рисунок 2 – Диаграмма деятельности (Администратор)

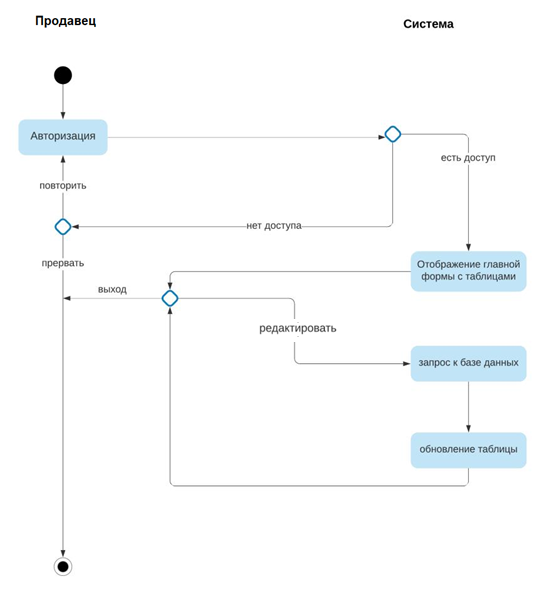


Рисунок 3 – Диаграмма деятельности (Продавец)

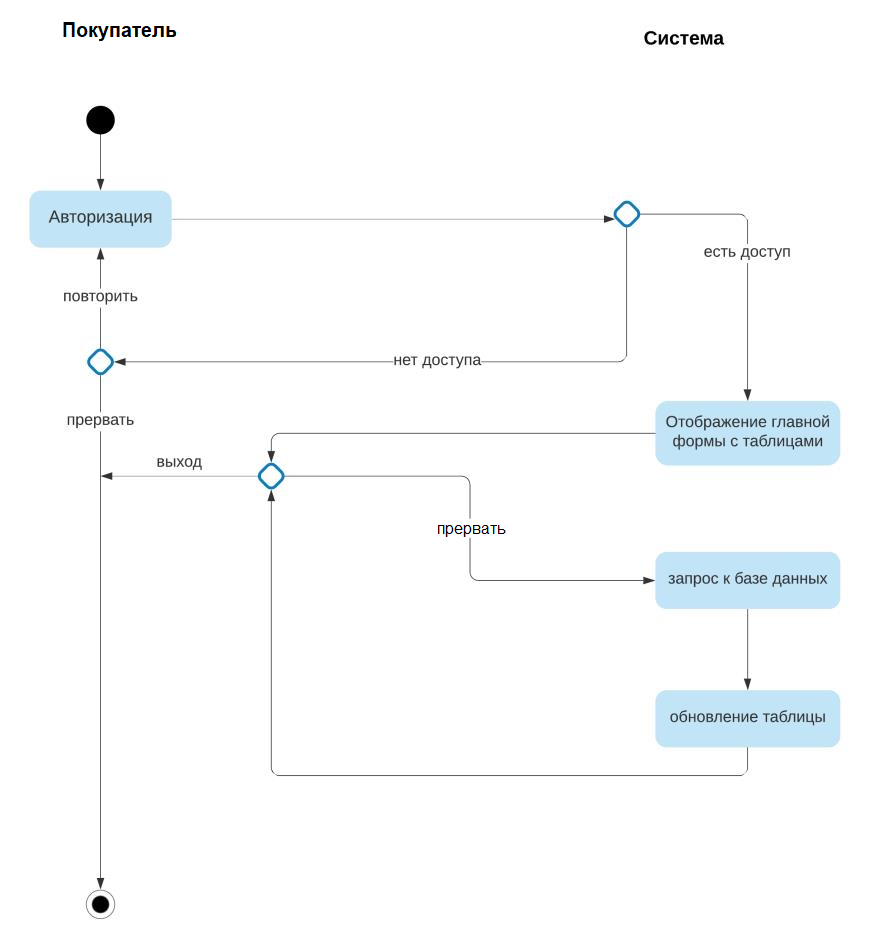


Рисунок 4 – Диаграмма деятельности (Покупатель)



Рисунок 5 – Диаграмма деятельности (Гость)

## Разработка функциональной структуры информационной системы. Карта навигации по информационной системе

В приложении А на рисунке 6 представлена карта переходов по информационной системе «Автосервис»

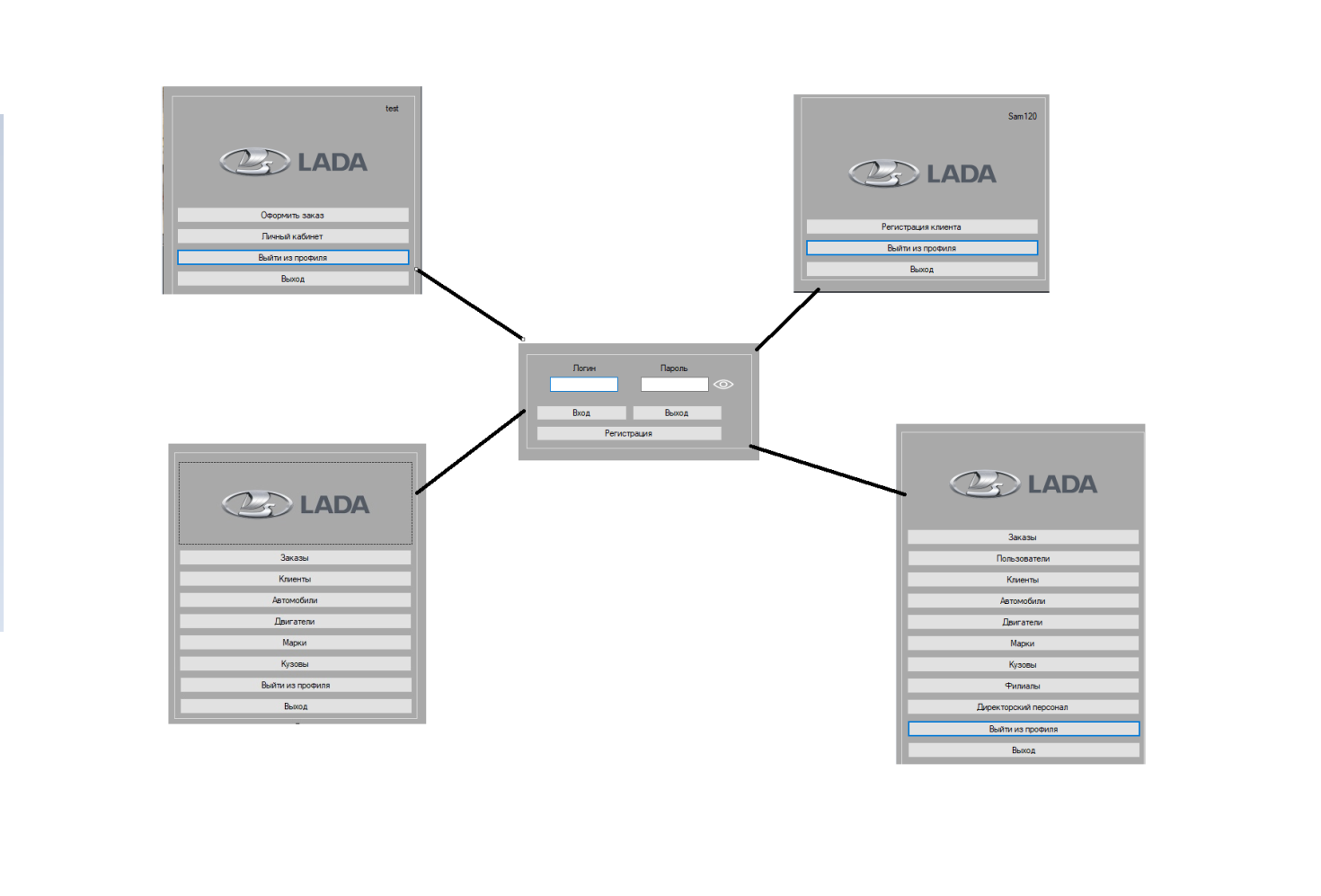


Рисунок 6 – Карта переходов

## Разработка макета информационной системы

На рисунке 7 предоставленно окно авторизации



Рисунок 7 – Окно авторизации

На этапе авторизации пользователь может ввести логин и пароль, чтобы продолжить работу в информационной системе.

В случае если пользователь еще не зарегистрирован в системе, он может сделать это, нажав на кнопку «Регистрация». В этом случае откроется окно регистрации, представленное в приложении 8 на рисунке 8.



Рисунок 8 – Окно регистрации

На этапе регистрации пользователю необходимо придумать свой логин и пароль.

Если сотрудник с введенными данными не зарегистрирован в базе

данных, в качестве сотрудника, то пользователь не сможет зарегистрироваться, и, соответственно, продолжить работу, так как данная информационная система предназначена только для сотрудников образовательной организации.

После того, как пользователь успешно прошел авторизацию, он может продолжить работу с таблицами на главной форме. На рисунке 9 – 12 представлена главная форма, которая будет отображена, в зависимости от прав пользователя в системе.



Рисунок 9 – Форма «Администратора»

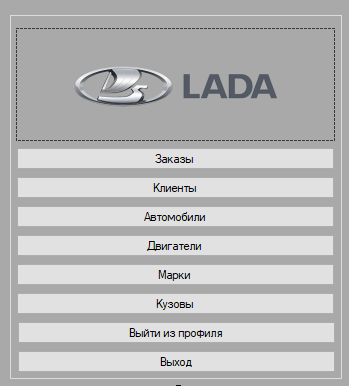


Рисунок 10 – Форма «Продавца»

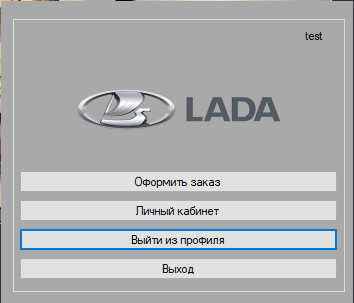


Рисунок 11 – Форма «Покупателя»

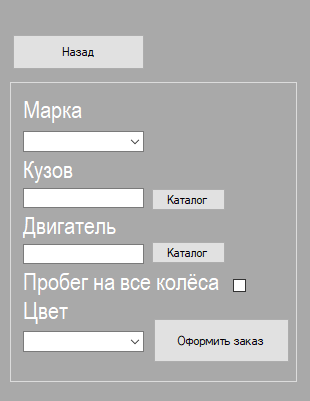
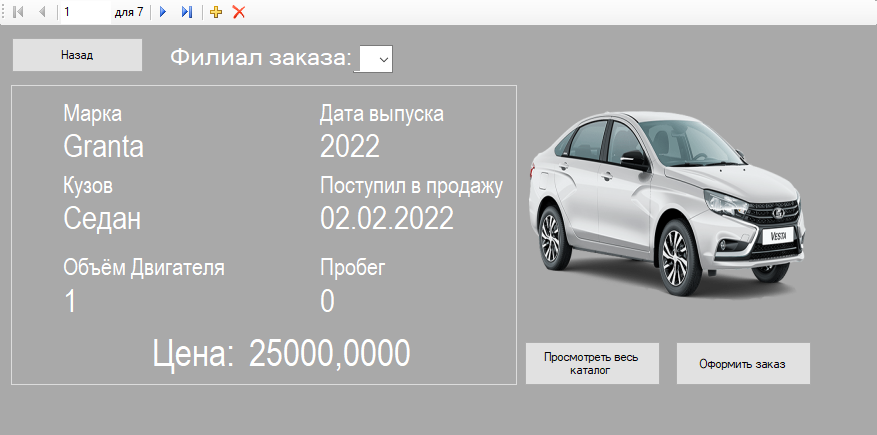
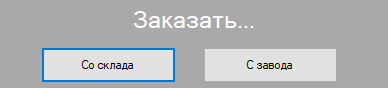


Рисунок 12 – Форма «Покупателя»

Аналогичным образом в зависимости от роли пользователя выглядит главная форма, на которой расположены все необходимые таблицы. Например, продавец сможет изменять только определённые таблицы, а для гостя будет недоступны функции редактирования, добавления и удаления записи, так как данной группе пользователей доступен только просмотр данных.

*Вывод по главе*: была спроектирована диаграмма вариантов использования, диаграмма деятельности, карта переходов и макеты будущей информационной системы.

# РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ

Прототип — это предварительная модель или образец продукта или системы, созданный для тестирования, оценки и демонстрации его возможностей перед полным производством или разработкой.

Прототипирование — это процесс создания прототипа, который может включать в себя создание физической модели продукта, разработку интерактивного прототипа на компьютере или создание макета интерфейса пользовательского опыта.

## Моделирование и разработка базы данных

Исходя из анализа предметной области, можно выделить восемь сущностей: «Users», «Автомобиль», «Двигатель», «ДиректорскийПерсонал», «Заказы», «Клиент»,«Кузов», «Марки», «Филиал».

В приложении А на рисунке 13 представлена логическая модель данных информационной системы «Автосервис».

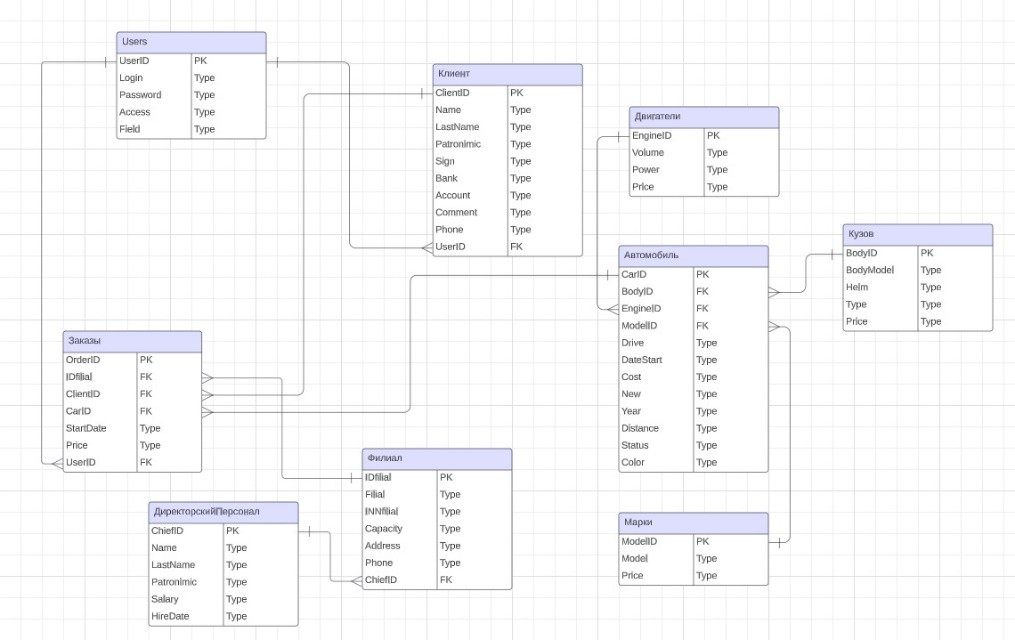


Рисунок 13 – Логическая модель данных

Создание базы данных происходит в MS SQL, в примере представлено создание базы данных в MS SQL Server.

С помощью раздела «Создание» и конструктора таблиц MS SQL созданы все таблицы и поля, представленные на логической модели данных.

Названия полей и их типы данных представлены в таблицах 1-9.

Таблица 1 – Поля таблицы «Users»

| Название поля | Тип данных |
| --- | --- |
| UserID | Числовой |
| Login | Текстовый |
| Password | Текстовый |
| Access | Текстовый |

Таблица 2 – Поля таблицы «Автомобиль»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| CarID | Числовой |
| BodyID | Числовой |
| EngineID | Числовой |
| Drive | Числовой |
| DateStart | Дата/время |
| Cost | Числовой |
| New | Числовой |
| Year | Числовой |
| Distance | Числовой |
| ModelID | Числовой |
| Status | Числовой |
| Color | Текстовый |

Таблица 3 – Поля таблицы «Двигатели»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| EngineID | Числовой |
| Volume | Числовой |
| Power | Числовой |
| Price | Числовой |

Таблица 4 – Поля таблицы «ДиректорскийПерсонал»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| ChiefID | Числовой |
| Name | Текстовый |
| LastName | Текстовый |
| Patronimic | Текстовый |
| Salary | Числовой |
| HireDate | Дата/время |

Таблица 5 – Поля таблицы «Заказы»

| Название поля | Тип данных |
| --- | --- |
|  |  |
| OrderID | Числовой |
| IDfilial | Числовой |
| ClientID | Числовой |
| CarID | Числовой |
| StartDate | Дата/время |
| Price | Числовой |
| UserID | Числовой |

Таблица 6 – Поля таблицы «Клиент»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| ClientID | Числовой |
| Name | Текстовый |
| LastName | Текстовый |
| Patronimic | Текстовый |
| Sign | Числовой |
| Bank | Текстовый |
| Account | Числовой |
| Comment | Текстовый |
| Phone | Числовой |
| UserID | Числовой |

Таблица 7 – Поля таблицы «Кузов»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| BodyID | Числовой |
| BodyModel | Текстовый |
| Helm | Числовой |
| Type | Текстовый |
| Price | Числовой |

Таблица 8 – Поля таблицы «Пользователи»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| ModelID | Числовой |
| Model | Текстовый |
| Photo | Photo |

Таблица 9 – Поля таблицы «Филиал»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Название поля | Тип данных |
| IDfilial | Текстовый |
| Filial | Числовой |
| INNfilial | Числовой |
| Capacity | Числовой |
| Address | Текстовый |
| Phone | Числовой |
| ChiefID | Числовой |

На рисунке 14 представлены все созданные таблицы в обозревателе объектов базы данных.

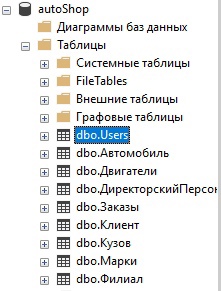


Рисунок 14 – Созданные таблицы

После того, как таблицы созданы, необходимо создать между ними связи.

В приложении А на рисунке 15 представлена физическая модель данных, отражающая все таблицы и их поля, а также связи между таблицами.

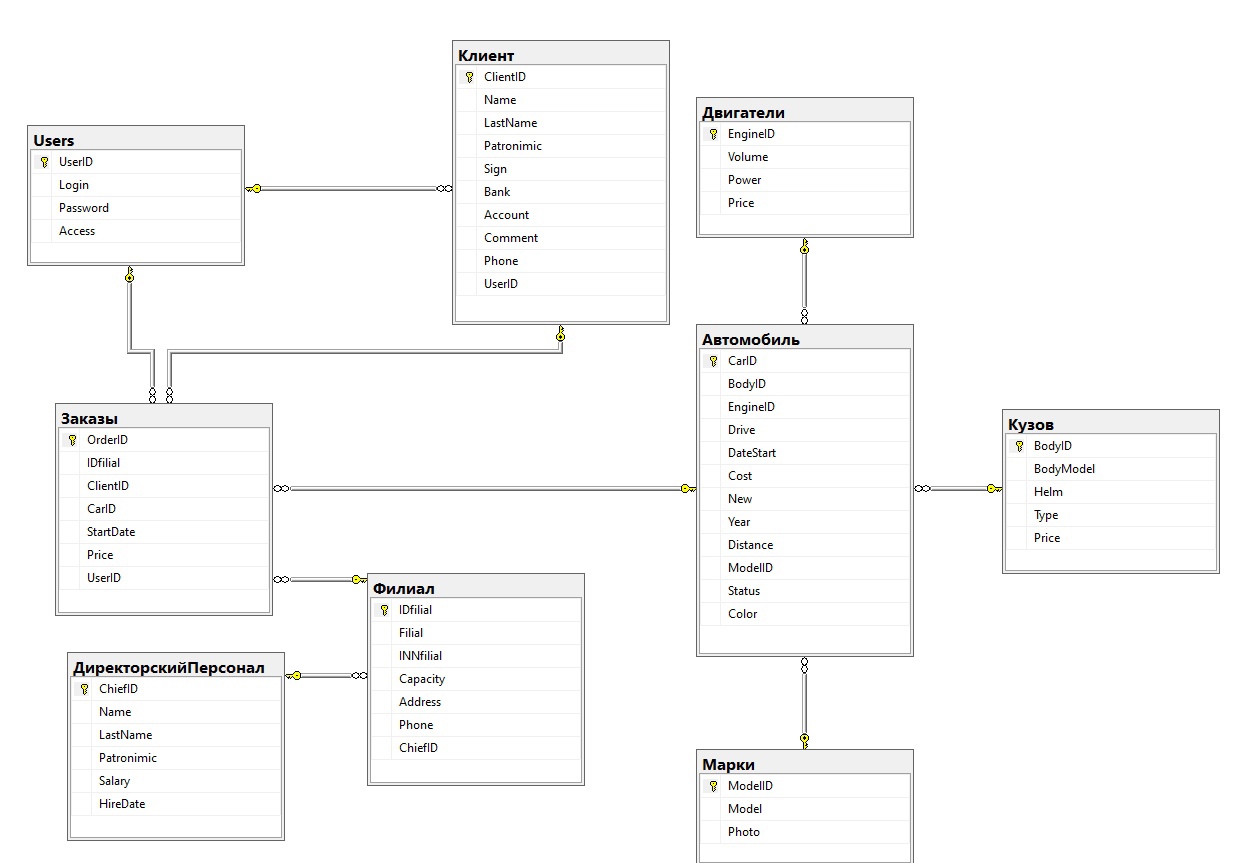


Рисунок 15 – Физическая модель базы данных

## Разработка программных модулей информационной системы

Разработанные в MS Visual Studio формы продемонстрированы в приложении А на рисунках 16-54.

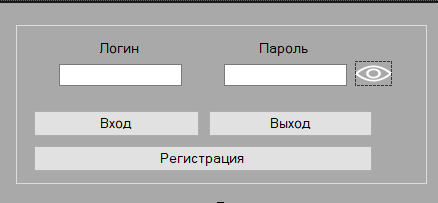


Рисунок 16 – Форма “Авторизация”

Свойства элементов формы Авторизация представлены в таблице 10.

Таблица 10 – Свойства элементов формы Авторизация

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Авторизация\_Form2 | FormBorderStyle | None |
| Text | Авторизация |
| StartPosition | CenterScreen |
| BackColor | DarkGray |
| Size | 440; 201 |
| Логин\_ label1 | Text | Логин |
| Font | Microsoft Sans Serif; 7,8pt |
| Пароль\_ label2 | Text | Пароль |
| Font | Microsoft Sans Serif; 7,8pt |
| Вход\_ button1 | Size | 167; 27 |
| Font | Microsoft Sans Serif; 7,8pt |
| Text | Вход |
| Выход\_button2 | Size | 165; 27 |
| Text | Выход |
| Font | Microsoft Sans Serif; 7,8pt |
| Регистрция\_button3 | Size | 340; 27 |
| Text | Регистрация |
| Font | Microsoft Sans Serif; 7,8pt |
| Скрыть\_пароль\_PictureBox3 | Size | 37; 25 |
| SizeMode | StretchImage |

Событийно-управляемые процедуры в форме авторизации представлены в таблице 11

Таблица 11 – Событийно-управляемые процедуры в форме «Авторизация»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Авторизация | public partial class Form2 : Form  {    public Form2()  {  InitializeComponent();  pictureBox1.Image = pictureBox1.InitialImage;  } |
| Кнопка\_Вход\_Click | string login = textBox1.Text.Trim();  string password = textBox2.Text;  if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))  {  MessageBox.Show("Не введён логин или пароль");  return;  }  if (loginAttempts <= 0)  {  MessageBox.Show("Превышено количество попыток входа. Обратитесь к администратору.");  this.Close();  return;  }  SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog = autoShop; Integrated Security = True");  try  {  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter("select \* from Users where Login = @login", sqlConnect);  da.SelectCommand.Parameters.AddWithValue("@login", login);  DataTable dt = new DataTable();  da.Fill(dt);  if (dt.Rows.Count == 0)  {  MessageBox.Show("Пользователь с таким логином не существует.");  loginAttempts--;  }  else  {  string dbPassword = dt.Rows[0]["Password"].ToString();  string role = dt.Rows[0]["Access"].ToString();  RegisteredUser currentUser = userLog(login);  if (password == dbPassword)  {  switch (role)  {  case "admin":  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Hide();  break;  case "user":  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Hide();  break;  case "buyer":  МенюПокупателя brForm = new МенюПокупателя(currentUser);  brForm.Show();  this.Hide();  break;  case "guest":  МенюГостя gstForm = new МенюГостя(currentUser);  gstForm.Show();  this.Hide();  break;  }  }  else  {  MessageBox.Show("Неверный Пароль.");  loginAttempts--;  }  }  }  catch (Exception ex)  {  MessageBox.Show("Опибка при входе: " + ex.Message);  }  } |
| Кнопка\_Выход\_Click | private void button2\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| Кнопка\_Войти\_как\_гость\_Click | private void Кнопка\_Войти\_как\_гость\_Click(object sender, EventArgs e) //Переходит на форму 'Гость'  {  this.Hide();  Гость UsrFrm3 = new Гость();  UsrFrm3.Show();  } |
| Кнопка\_Регестрации\_Click | private void button3\_Click(object sender, EventArgs e)  {  Регистрация regForm = new Регистрация();  regForm.Show();  this.Hide();  } |
| Показать\_пароль\_Click | private void pictureBox1\_Click(object sender, EventArgs e)  {  if (hidden == false)  {  pictureBox1.Image = Properties.Resources.glazik;  textBox2.PasswordChar = '\*';  hidden = true;  }  else  {  pictureBox1.Image = Properties.Resources.glazik\_zakryt;  textBox2.PasswordChar = '\0';  hidden = false;  }  } |

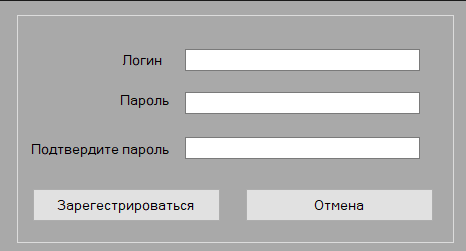


Рисунок 17 – Форма “Регистрация”

Свойства элементов формы Регистрация представлены в таблице 13.

Таблица 13 – Свойства элементов формы Регистрация

| Объект | Свойство | Значение |
| --- | --- | --- |
| Регистрация | FormBorderStyle | None |
| Text | Регистрация |
| StartPosition | WindowsDefaultLocation |
| BackColor | DarkGray |
| Size | 468; 252 |
| ShowIcon | False |
| Логин\_label1 | Text | Логин |
| Font | Microsoft Sans Serif; 7,8pt |
| Пароль\_label2 | Size | Пароль |
| Font | Microsoft Sans Serif; 7,8pt |
| Подтвердите пароль\_label3 | Size | Подтвердите пароль |
| Font | Microsoft Sans Serif; 7,8pt |
| Логин\_textBox1 | Font | Microsoft Sans Serif; 7,8pt |
| Size | 235; 22 |
| Пароль\_textBox2 | Font | Microsoft Sans Serif; 7,8pt |
| Size | 235; 22 |
| Подтвердите пароль\_textBox3 | Font | Microsoft Sans Serif; 7,8pt |
| Size | 235; 22 |
| Зарегестрироваться\_ button1 | Size | 340; 27 |
| Text | Зарегестрироваться |
| Font | Microsoft Sans Serif; 7,8pt |
| Отмена\_button 2 | Size | 340; 27 |
| Text | Отмена |
| Font | Microsoft Sans Serif; 7,8pt |

Событийно-управляемые процедуры в форме Регистрация представлены в таблице 14

Таблица 14 – Событийно-управляемые процедуры в форме «Регистрация»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Регистрация | public partial class Регистрация : Form  {  public Регистрация()  {  InitializeComponent();  } |
| Кнопка\_Зарегестрироваться\_Click | private void button1\_Click(object sender, EventArgs e)  {  string login = textBox1.Text.Trim();  string password = textBox2.Text;  if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))  {  MessageBox.Show("Не введён логин или пароль");  return;  }  SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog = autoShop; Integrated Security = True");  SqlDataAdapter da = new SqlDataAdapter("select \* from Users where Login = @login", sqlConnect);  da.SelectCommand.Parameters.AddWithValue("@login", login);  DataTable dt = new DataTable();  da.Fill(dt);  if (dt.Rows.Count != 0)  {  MessageBox.Show("Пользователь с таким логином уже существует.");  }  else  {  if (textBox2.Text != textBox3.Text)  {  MessageBox.Show("Пароли не совпадают");  }  else  {  using (SqlCommand cmd = new SqlCommand("Insert Into Users (Login, Password, Access) Values (@login, @password, 'guest')", sqlConnect))  {  cmd.Parameters.AddWithValue("@login", login);  cmd.Parameters.AddWithValue("@password", password);  try  {  sqlConnect.Open();  int rowsAffected = cmd.ExecuteNonQuery();  if (rowsAffected > 0)  {  // Регистрация успешно завершена  MessageBox.Show("Регистрация успешно завершена!");  Form2 logForm = new Form2();  logForm.Show();  this.Close();  }  else  {  // Вставка не удалась, обработайте это по необходимости  MessageBox.Show("Ошибка при регистрации. Пожалуйста, попробуйте снова.");  }  }  catch (Exception ex)  {  // Обработка ошибок при выполнении SQL-запроса  MessageBox.Show("Ошибка при регистрации: " + ex.Message);  }  }  }  }  } |
| Отмена\_Click | private void button2\_Click(object sender, EventArgs e)  {  Form2 logForm = new Form2();  logForm.Show();  this.Close();  } |

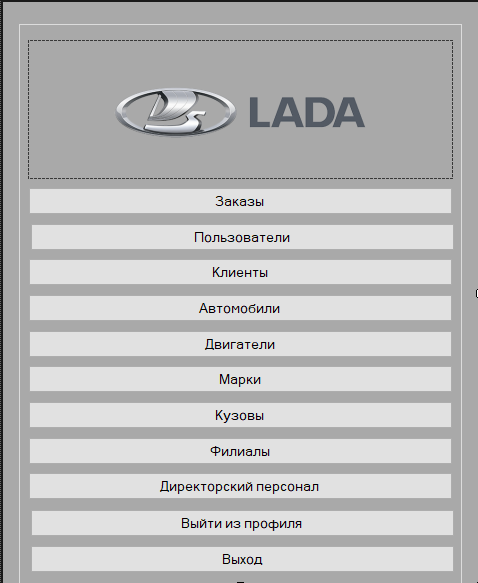


Рисунок 18 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 13.

Таблица 18 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Администратор | FormBorderStyle | None |
| Text | Администратор |
| StartPosition | CenterScreen |
| BackColor | DarkGrey |
| Size | 356; 483 |
| pictureBox1 | Size | 319; 113 |
| SizeMode | StretchImage |
| button1 | Size | 142; 28 |
| Text | Заказы |
| Font | Microsoft Sans Serif; 8,25pt |
| button5 | Size | 142; 28 |
| Text | Пользователи |
| Font | Microsoft Sans Serif; 8,25pt |
| button6 | Size | 142; 28 |
| Text | Клиенты |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 142; 28 |
| Text | Автомобили |
| Font | Microsoft Sans Serif; 8,25pt |
| button7 | Size | 142; 28 |
| Text | Двигатели |
| Font | Microsoft Sans Serif; 8,25pt |
| button8 | Size | 142; 28 |
| Text | Марки |
| Font | Microsoft Sans Serif; 8,25pt |
| button9 | Size | 142; 28 |
| Text | Кузовы |
| Font | Microsoft Sans Serif; 8,25pt |
| button10 | Size | 142; 28 |
| Text | Филиалы |
| Font | Microsoft Sans Serif; 8,25pt |
| button11 | Size | 142; 28 |
| Text | Директорский персонал |
| Font | Microsoft Sans Serif; 8,25pt |
| button3 | Size | 142; 28 |
| Text | Выйти из профиля |
| Font | Microsoft Sans Serif; 8,25pt |
| button4 | Size | 142; 28 |
| Text | Выход |
| Font | Microsoft Sans Serif; 8,25pt |
| groupBox1 | Location | 12; 12 |
| Size | 332; 462 |
| AutoSizeMode | GrowOnly |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 19

Таблица 19 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class МенюАдмина | public МенюАдмина()  {  InitializeComponent();  } |
| button5\_Click | private void button5\_Click(object sender, EventArgs e)  {  УправлениеПользователями userLog = new УправлениеПользователями();  userLog.Show();  this.Close();  } |
| button4\_Click | private void button4\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| button3\_Click | private void button3\_Click(object sender, EventArgs e)  {  Form2 logForm = new Form2();  logForm.Show();  this.Close();  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  ПросмотрЗаказовADM order = new ПросмотрЗаказовADM();  order.Show();  this.Close();  } |
| button6\_Click | private void button6\_Click(object sender, EventArgs e)  {  КлиентыADM client = new КлиентыADM();  client.Show();  this.Close();  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  АвтомобилиADM car = new АвтомобилиADM();  car.Show();  this.Close();  } |
| button7\_Click | private void button7\_Click(object sender, EventArgs e)  {  ДвигателиADM engine = new ДвигателиADM();  engine.Show();  this.Close();  } |
| button8\_Click | private void button8\_Click(object sender, EventArgs e)  {  МаркиADM model = new МаркиADM();  model.Show();  this.Close();  } |
| button9\_Click | private void button9\_Click(object sender, EventArgs e)  {  КузовыADM body = new КузовыADM();  body.Show();  this.Close();  } |
| button10\_Click | private void button10\_Click(object sender, EventArgs e)  {  Филиалы branch = new Филиалы();  branch.Show();  this.Close();  } |
| button11\_Click | private void button11\_Click(object sender, EventArgs e)  {  ДиректорскийПерсонал staff = new ДиректорскийПерсонал();  staff.Show();  this.Close();  } |

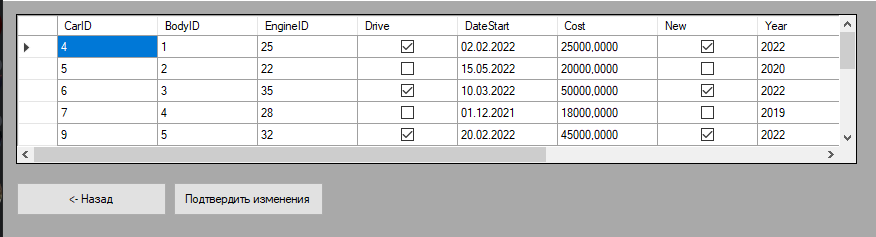


Рисунок 19 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 13.

Таблица 20 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| АвтомобильADM | FormBorderStyle | None |
| Text | АвтомобильADM |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 21

Таблица 21 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class АвтомобилиADM | public АвтомобилиADM()  {  InitializeComponent();  } |
| АвтомобилиADM\_Load | private void АвтомобилиADM\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Автомобиль". При необходимости она может быть перемещена или удалена.  this.автомобильTableAdapter.Fill(this.autoShopDataSet.Автомобиль);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  автомобильBindingSource.EndEdit();  автомобильTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

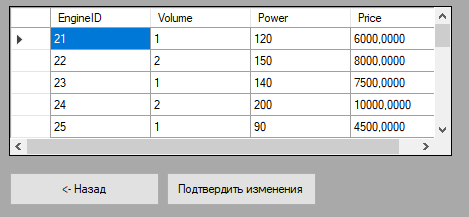


Рисунок 20 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 13.

Таблица 22 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| ДвигателиADM | FormBorderStyle | None |
| Text | ДвигателиADM |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 23

Таблица 24 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class ДвигателиADM | public ДвигателиADM()  {  InitializeComponent();  } |
| Двигатели\_Load | private void Двигатели\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Двигатели". При необходимости она может быть перемещена или удалена.  this.двигателиTableAdapter.Fill(this.autoShopDataSet.Двигатели);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  двигателиBindingSource.EndEdit();  двигателиTableAdapter.Update(autoShopDataSet); } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

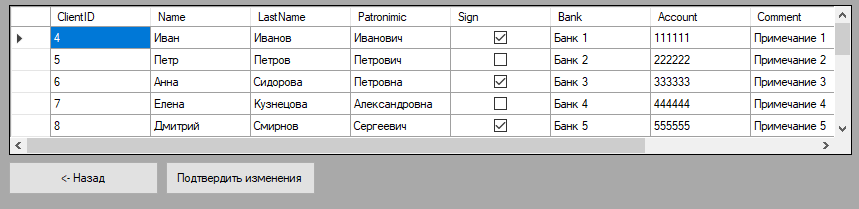


Рисунок 21 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 13.

Таблица 25 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| КлиентыADM | FormBorderStyle | None |
| Text | КлиентыADM |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 26.

Таблица 27 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class КлиентыADM | public КлиентыADM ()  {  InitializeComponent();  } |
| КлиентыADM\_Load | private void КлиентыADM\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Клиент". При необходимости она может быть перемещена или удалена.  this.клиентTableAdapter.Fill(this.autoShopDataSet.Клиент);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  клиентBindingSource.EndEdit();  клиентTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

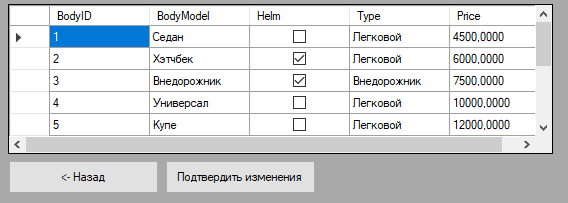


Рисунок 22 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 13.

Таблица 28 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| КузовыADM | FormBorderStyle | None |
| Text | КузовыADM |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 29

Таблица 29 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class КузовыADM | public КузовыADM ()  {  InitializeComponent();  } |
| КузовыADM\_Load | private void Кузовы\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Кузов". При необходимости она может быть перемещена или удалена.  this.кузовTableAdapter.Fill(this.autoShopDataSet.Кузов);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  кузовBindingSource.EndEdit();  кузовTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

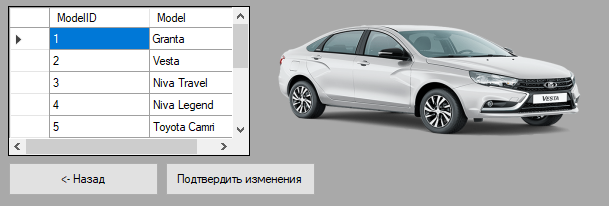


Рисунок 23 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 30.

Таблица 30 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| МаркиADM | FormBorderStyle | None |
| Text | МаркиADM |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 31.

Таблица 31 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class МаркиADM | public МаркиADM ()  {  InitializeComponent();  } |
| МаркиADM\_Load | private void МаркиADM\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Марки". При необходимости она может быть перемещена или удалена.  this.маркиTableAdapter.Fill(this.autoShopDataSet.Марки);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  маркиBindingSource.EndEdit();  маркиTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

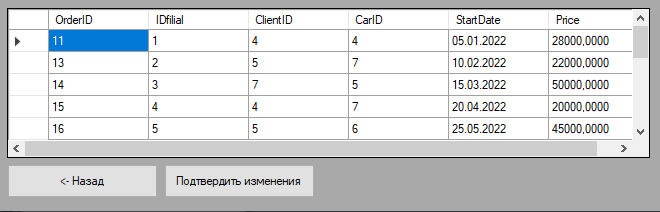


Рисунок 24 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 32.

Таблица 32 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| ПросмотрЗаказовADM | FormBorderStyle | None |
| Text | ПросмотрЗаказовADM |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 33.

Таблица 33 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class ПросмотрЗаказовADM | public ПросмотрЗаказовADM ()  {  InitializeComponent();  } |
| ПросмотрЗаказовADM\_Load | private void ПросмотрЗаказовADM\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Заказы". При необходимости она может быть перемещена или удалена.  this.заказыTableAdapter.Fill(this.autoShopDataSet.Заказы);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  заказыBindingSource.EndEdit();  заказыTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

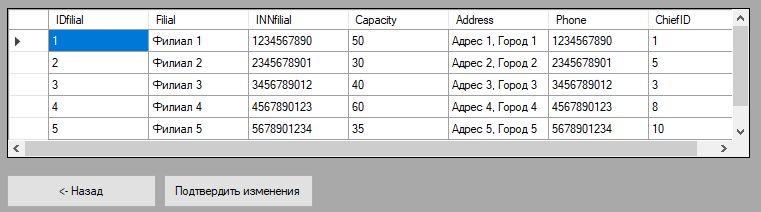


Рисунок 25 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 34.

Таблица 34 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Филиалы | FormBorderStyle | None |
| Text | Филиалы |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 35.

Таблица 35 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Филиалы | public Филиалы()  {  InitializeComponent();  } |
| Филиалы \_Load | private void Филиалы\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Филиал". При необходимости она может быть перемещена или удалена.  this.филиалTableAdapter.Fill(this.autoShopDataSet.Филиал);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  филиалBindingSource.EndEdit();  филиалTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

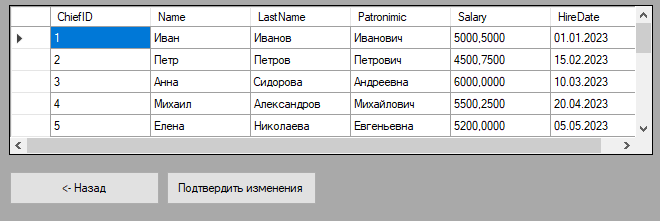


Рисунок 26 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 36.

Таблица 36 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| ДиректорскийПерсонал | FormBorderStyle | None |
| Text | ДиректорскийПерсонал |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 36.

Таблица 36 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class ДиректорскийПерсонал | public ДиректорскийПерсонал ()  {  InitializeComponent();  } |
| ДиректорскийПерсонал \_Load | private void ДиректорскийПерсонал\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.ДиректорскийПерсонал". При необходимости она может быть перемещена или удалена.  this.директорскийПерсоналTableAdapter.Fill(this.autoShopDataSet.ДиректорскийПерсонал);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  директорскийПерсоналBindingSource.EndEdit();  директорскийПерсоналTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюАдмина adminForm = new МенюАдмина();  adminForm.Show();  this.Close();  } |

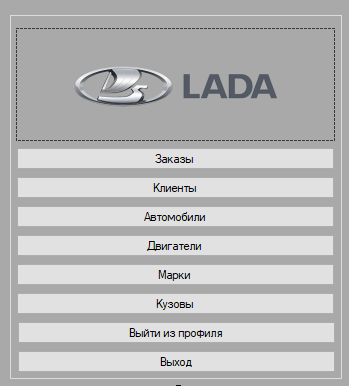


Рисунок 27 – Форма “Продавца”

Свойства элементов формы Продавец представлены в таблице 36.

Таблица 36 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Администратор | FormBorderStyle | None |
| Text | Администратор |
| StartPosition | CenterScreen |
| BackColor | DarkGrey |
| Size | 356; 483 |
| pictureBox1 | Size | 319; 113 |
| SizeMode | StretchImage |
| button1 | Size | 142; 28 |
| Text | Заказы |
| Font | Microsoft Sans Serif; 8,25pt |
| button6 | Size | 142; 28 |
| Text | Клиенты |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 142; 28 |
| Text | Автомобили |
| Font | Microsoft Sans Serif; 8,25pt |
| button7 | Size | 142; 28 |
| Text | Двигатели |
| Font | Microsoft Sans Serif; 8,25pt |
| button8 | Size | 142; 28 |
| Text | Марки |
| Font | Microsoft Sans Serif; 8,25pt |
| button9 | Size | 142; 28 |
| Text | Кузовы |
| Font | Microsoft Sans Serif; 8,25pt |
| button3 | Size | 142; 28 |
| Text | Выйти из профиля |
| Font | Microsoft Sans Serif; 8,25pt |
| button4 | Size | 142; 28 |
| Text | Выход |
| Font | Microsoft Sans Serif; 8,25pt |
| groupBox1 | Location | 12; 12 |
| Size | 332; 462 |
| AutoSizeMode | GrowOnly |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 37.

Таблица 37 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class МенюПользователя | public МенюПользователя()  {  InitializeComponent();  } |
| button4\_Click | private void button4\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| button3\_Click | private void button3\_Click(object sender, EventArgs e)  {  Form2 logForm = new Form2();  logForm.Show();  this.Close();  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  ПросмотрЗаказов order = new ПросмотрЗаказовADM();  order.Show();  this.Close();  } |
| button6\_Click | private void button6\_Click(object sender, EventArgs e)  {  Клиенты client = new КлиентыADM();  client.Show();  this.Close();  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  Автомобили car = new АвтомобилиADM();  car.Show();  this.Close();  } |
| button7\_Click | private void button7\_Click(object sender, EventArgs e)  {  Двигатели engine = new ДвигателиADM();  engine.Show();  this.Close();  } |
| button8\_Click | private void button8\_Click(object sender, EventArgs e)  {  Марки model = new МаркиADM();  model.Show();  this.Close();  } |
| button9\_Click | private void button9\_Click(object sender, EventArgs e)  {  Кузовы body = new КузовыADM();  body.Show();  this.Close();  } |

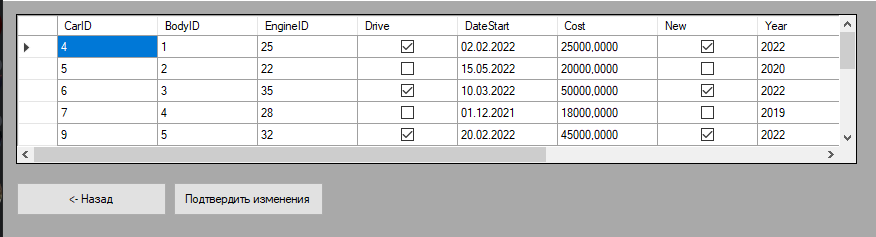


Рисунок 28 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 38.

Таблица 38 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Автомобиль | FormBorderStyle | None |
| Text | Автомобиль |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 39.

Таблица 39 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Автомобили | public Автомобили()  {  InitializeComponent();  } |
| Автомобили\_Load | private void Автомобили\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Автомобиль". При необходимости она может быть перемещена или удалена.  this.автомобильTableAdapter.Fill(this.autoShopDataSet.Автомобиль);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  автомобильBindingSource.EndEdit();  автомобильTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Close();  } |

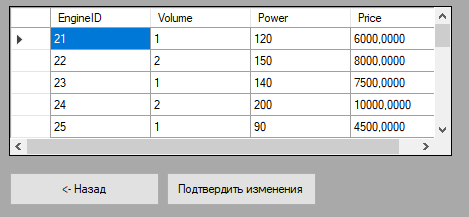


Рисунок 29 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 40.

Таблица 40 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Двигатели | FormBorderStyle | None |
| Text | Двигатели |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 41.

Таблица 41 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Двигатели | public Двигатели()  {  InitializeComponent();  } |
| Двигатели\_Load | private void Двигатели\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Двигатели". При необходимости она может быть перемещена или удалена.  this.двигателиTableAdapter.Fill(this.autoShopDataSet.Двигатели);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  двигателиBindingSource.EndEdit();  двигателиTableAdapter.Update(autoShopDataSet); } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Close();  } |

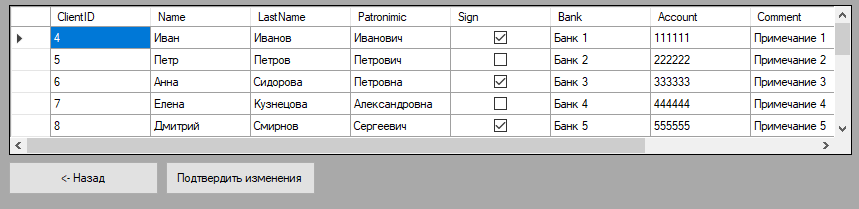


Рисунок 30 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 42.

Таблица 42 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Клиенты | FormBorderStyle | None |
| Text | Клиенты |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 43.

Таблица 43 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Клиенты | public Клиенты ()  {  InitializeComponent();  } |
| Клиенты\_Load | private void Клиенты\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Клиент". При необходимости она может быть перемещена или удалена.  this.клиентTableAdapter.Fill(this.autoShopDataSet.Клиент);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  клиентBindingSource.EndEdit();  клиентTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Close();  } |

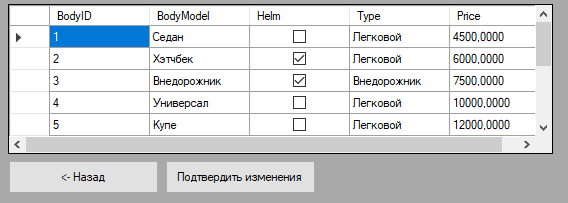


Рисунок 31 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 44.

Таблица 44 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| Кузовы | FormBorderStyle | None |
| Text | Кузовы |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 45.

Таблица 45 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Кузовы | public Кузовы ()  {  InitializeComponent();  } |
| Кузовы\_Load | private void Кузовы\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Кузов". При необходимости она может быть перемещена или удалена.  this.кузовTableAdapter.Fill(this.autoShopDataSet.Кузов);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  кузовBindingSource.EndEdit();  кузовTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Close();  } |

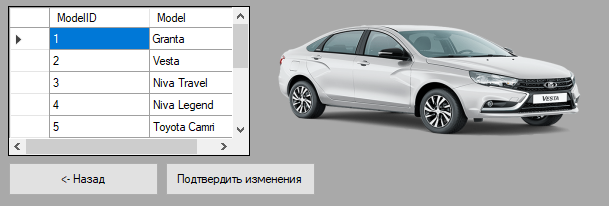


Рисунок 32 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 46.

Таблица 46 – Свойства элементов формы Администрато

| Объект | Свойство | Значение |
| --- | --- | --- |
| Марки | FormBorderStyle | None |
| Text | Марки |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 47.

Таблица 47 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class Марки | public Марки ()  {  InitializeComponent();  } |
| Марки\_Load | private void Марки\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Марки". При необходимости она может быть перемещена или удалена.  this.маркиTableAdapter.Fill(this.autoShopDataSet.Марки);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  маркиBindingSource.EndEdit();  маркиTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Close();  } |

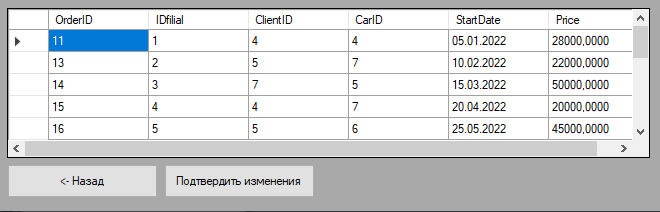


Рисунок 33 – Форма “ Администратора”

Свойства элементов формы Администратор представлены в таблице 48.

Таблица 48 – Свойства элементов формы Администратор

| Объект | Свойство | Значение |
| --- | --- | --- |
| ПросмотрЗаказов | FormBorderStyle | None |
| Text | ПросмотрЗаказов |
| StartPosition | CenterScreen |
| Size | 875; 237 |
| ShowIcon | False |
| dataGridView1 | Size | 841; 150 |
| BackColor | Control |
| button1 | Size | 151; 34 |
| Text | <- Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 151; 34 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |

Событийно-управляемые процедуры в форме Администратор представлены в таблице 49.

Таблица 49 – Событийно-управляемые процедуры в форме «Администратор»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class ПросмотрЗаказов | public ПросмотрЗаказов ()  {  InitializeComponent();  } |
| ПросмотрЗаказов\_Load | private void ПросмотрЗаказов\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "autoShopDataSet.Заказы". При необходимости она может быть перемещена или удалена.  this.заказыTableAdapter.Fill(this.autoShopDataSet.Заказы);  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  заказыBindingSource.EndEdit();  заказыTableAdapter.Update(autoShopDataSet);  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  МенюПользователя userForm = new МенюПользователя();  userForm.Show();  this.Close();  } |

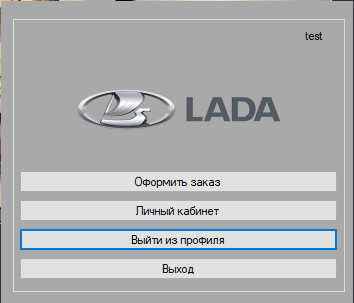


Рисунок 34 – Форма “Покупателя”

Свойства элементов формы Покупатель представлены в таблице 50.

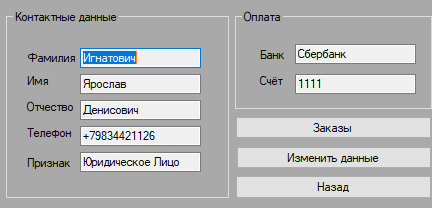
Таблица 50 – Свойства элементов формы Покупатель

| Объект | Свойство | Значение |
| --- | --- | --- |
| МенюПокупателя | FormBorderStyle | None |
| Text | МенюПокупателя |
| StartPosition | CenterScreen |
| Size | 352; 302 |
| ShowIcon | False |
| pictureBox1 | Size | 319; 113 |
| SizeMode | StretchImage |
| label1 | Size | 39; 13 |
| button1 | Size | 319; 23 |
| Text | Оформить заказ |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 319; 23 |
| Text | Личный кабинет |
| Font | Microsoft Sans Serif; 8,25pt |
| button3 | Size | 319; 23 |
| Text | Выйти из профиля |
| Font | Microsoft Sans Serif; 8,25pt |
| button4 | Size | 319; 23 |
| Text | Выход |
| Font | Microsoft Sans Serif; 8,25pt |
| groupBox1 | Location | 12; 12 |
| Size | 332; 280 |
| AutoSizeMode | GrowOnly |

Событийно-управляемые процедуры в форме Покупатель представлены в таблице 51.

Таблица 51 – Событийно-управляемые процедуры в форме «Покупатель»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class МенюПокупателя | RegisteredUser currentUser;  public МенюПокупателя(RegisteredUser user)  {  InitializeComponent();  currentUser = user;  label1.Text = currentUser.UserName;  } |
| button4\_Click | private void button4\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| button3\_Click | private void button3\_Click(object sender, EventArgs e)  {  Form2 logForm = new Form2();  logForm.Show();  this.Close();  } |
| button5\_Click | private void button5\_Click(object sender, EventArgs e)  {  ЛичныйКабинет profile = new ЛичныйКабинет(currentUser);  profile.Show();  this.Close();  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  ОформлениеЗаказа createOrder = new ОформлениеЗаказа(currentUser);  createOrder.Show();  this.Close();  } |



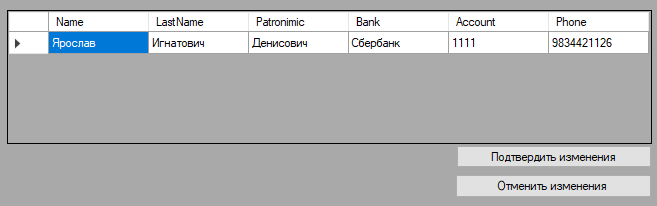


Рисунок 35 – Форма “Покупателя”

Свойства элементов формы Покупатель представлены в таблице 52.

Таблица 52 – Свойства элементов формы Покупатель

| Объект | Свойство | Значение |
| --- | --- | --- |
| ЛичныйКабинет | FormBorderStyle | None |
| Text | ЛичныйКабинет |
| StartPosition | CenterScreen |
| Size | 444; 213 |
| ShowIcon | False |
| pictureBox1 | Size | 319; 113 |
| SizeMode | StretchImage |
| label1 | Size | 39; 13 |
| Text | Фамилия |
| label2 | Size | 39; 13 |
| Text | Имя |
| label3 | Size | 39; 13 |
| Text | Отчество |
| label4 | Size | 39; 13 |
| Text | Телефон |
| label5 | Size | 39; 13 |
| Text | Признак |
| label6 | Size | 39; 13 |
| Text | Банк |
| label7 | Size | 39; 13 |
| Text | Счёт |
| textBox1 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| textBox2 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| textBox3 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| textBox4 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| textBox5 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| textBox6 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| textBox7 | Font | Microsoft Sans Serif; 8,25pt |
| Size | 121; 20 |
| ReadOnly | True |
| button1 | Size | 319; 23 |
| Text | Заказы |
| Font | Microsoft Sans Serif; 8,25pt |
| button2 | Size | 319; 23 |
| Text | Изменить данные |
| Font | Microsoft Sans Serif; 8,25pt |
| button3 | Size | 319; 23 |
| Text | Назад |
| Font | Microsoft Sans Serif; 8,25pt |
| dataGridView1 | Size | 645; 134 |
| BackColor | Control |
| button4 | Size | 319; 23 |
| Text | Подтвердить изменения |
| Font | Microsoft Sans Serif; 8,25pt |
| button5 | Size | 319; 23 |
| Text | Отменить изменения |
| Font | Microsoft Sans Serif; 8,25pt |
| groupBox1 | Location | 12; 12 |
| Size | 332; 280 |
| AutoSizeMode | GrowOnly |
| Text | Контактные данные |
| groupBox2 | Location | 12; 12 |
| Size | 332; 280 |
| AutoSizeMode | GrowOnly |
| Text | Оплата |

Событийно-управляемые процедуры в форме Покупатель представлены в таблице 54.

Таблица 54 – Событийно-управляемые процедуры в форме «Покупатель»

| Объект | Событийно-управляемые процедуры |
| --- | --- |
| class ЛичныйКабинет | RegisteredUser CurrentUser;  public ЛичныйКабинет(RegisteredUser currentUser)  {  InitializeComponent();  button4.Visible = false;  button5.Visible = false;  dataGridView1.Visible = false;  CurrentUser = currentUser;  LoadUserData();  LoadDataGridViewData();  } |
| button2\_Click | private void button2\_Click(object sender, EventArgs e)  {  МенюПокупателя brForm = new МенюПокупателя(CurrentUser);  brForm.Show();  this.Close();  } |
| button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  Заказы orders = new Заказы(CurrentUser);  orders.Show();  this.Close();  } |
| button3\_Click | private void button3\_Click(object sender, EventArgs e)  {  button1.Visible = false;  button2.Visible = false;  button3.Visible = false;  button4.Visible = true;  button5.Visible = true;  groupBox1.Visible = false;  groupBox2.Visible = false;  dataGridView1.Visible = true;  this.Size = new Size(671, 213);  } |
| button5\_Click | private void button5\_Click(object sender, EventArgs e)  {    button1.Visible = true;  button2.Visible = true;  button3.Visible = true;  button4.Visible = false;  button5.Visible = false;  groupBox1.Visible = true;  groupBox2.Visible = true;  dataGridView1.Visible = false;  this.Size = new Size(445, 213);  } |
| button4\_Click | private void button4\_Click(object sender, EventArgs e)  {  using (SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog=autoShop;Integrated Security=True"))  {  try  {  sqlConnect.Open();  SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Клиент WHERE UserID = @login", sqlConnect);  // Добавляем параметр к командам  adapter.UpdateCommand = new SqlCommand("UPDATE Клиент SET LastName = @LastName, Name = @Name, Patronimic = @Patronimic, Phone = @Phone, Bank = @Bank, Account = @Account WHERE UserID = @login", sqlConnect);  adapter.UpdateCommand.Parameters.AddWithValue("@login", CurrentUser.UserID);  adapter.UpdateCommand.Parameters.Add("@LastName", SqlDbType.NVarChar, 50, "LastName");  adapter.UpdateCommand.Parameters.Add("@Name", SqlDbType.NVarChar, 50, "Name");  adapter.UpdateCommand.Parameters.Add("@Patronimic", SqlDbType.NVarChar, 50, "Patronimic");  adapter.UpdateCommand.Parameters.Add("@Phone", SqlDbType.NVarChar, 50, "Phone");  adapter.UpdateCommand.Parameters.Add("@Bank", SqlDbType.NVarChar, 50, "Bank");  adapter.UpdateCommand.Parameters.Add("@Account", SqlDbType.NVarChar, 50, "Account");  adapter.InsertCommand = new SqlCommand("INSERT INTO Клиент (UserID, LastName, Name, Patronimic, Phone, Bank, Account) VALUES (@login, @LastName, @Name, @Patronimic, @Phone, @Bank, @Account)", sqlConnect);  adapter.InsertCommand.Parameters.AddWithValue("@login", CurrentUser.UserID);  adapter.InsertCommand.Parameters.Add("@LastName", SqlDbType.NVarChar, 50, "LastName");  adapter.InsertCommand.Parameters.Add("@Name", SqlDbType.NVarChar, 50, "Name");  adapter.InsertCommand.Parameters.Add("@Patronimic", SqlDbType.NVarChar, 50, "Patronimic");  adapter.InsertCommand.Parameters.Add("@Phone", SqlDbType.NVarChar, 50, "Phone");  adapter.InsertCommand.Parameters.Add("@Bank", SqlDbType.NVarChar, 50, "Bank");  adapter.InsertCommand.Parameters.Add("@Account", SqlDbType.NVarChar, 50, "Account");  adapter.DeleteCommand = new SqlCommand("DELETE FROM Клиент WHERE UserID = @login", sqlConnect);  adapter.DeleteCommand.Parameters.AddWithValue("@login", CurrentUser.UserID);  // Применяем изменения  adapter.Update((DataTable)dataGridView1.DataSource);  MessageBox.Show("Данные успешно обновлены");  МенюПокупателя brForm = new МенюПокупателя(CurrentUser);  brForm.Show();  this.Close();  }  catch (Exception ex)  {  MessageBox.Show("Не удалось обновить данные: " + ex.ToString());  }  }  } |
| void LoadUserData() | private void LoadUserData()  {  using (SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog=autoShop;Integrated Security=True"))  {  try  {  sqlConnect.Open();  SqlCommand cmd = new SqlCommand("SELECT \* FROM Клиент WHERE UserID = @login", sqlConnect);  cmd.Parameters.AddWithValue("@login", CurrentUser.UserID);  using (SqlDataReader reader = cmd.ExecuteReader())  {  if (reader.Read())  {  textBox1.Text = reader["LastName"].ToString();  textBox2.Text = reader["Name"].ToString();  textBox3.Text = reader["Patronimic"].ToString();  textBox4.Text = "+7" + reader["Phone"].ToString();  textBox5.Text = reader["Bank"].ToString();  textBox6.Text = reader["Account"].ToString();  string sign = reader["Sign"].ToString();  textBox7.Text = (sign == "1") ? "Физическое Лицо" : "Юридическое Лицо";  reader.Close();  }  }  }  catch (Exception ex)  {  MessageBox.Show("Ошибка: " + ex.Message);  }  }  } |
| void LoadDataGridViewData() | private void LoadDataGridViewData()  {  using (SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog=autoShop;Integrated Security=True"))  {  try  {  sqlConnect.Open();  SqlCommand cmd = new SqlCommand("SELECT \* FROM Клиент WHERE UserID = @login", sqlConnect);  cmd.Parameters.AddWithValue("@login", CurrentUser.UserID);  SqlDataAdapter adapter = new SqlDataAdapter(cmd);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  // Привязываем данные к DataGridView  dataGridView1.DataSource = dataTable;  }  catch (Exception ex)  {  MessageBox.Show("Ошибка: " + ex.Message);  }  }  } |

# РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестовый сценарий – это неавтоматизированный или автоматизированный сценарий, содержащий инструкции по реализации тестового набора. Тестовый сценарий может быть написан вручную (для выполнения человеком) либо полностью или частично автоматизирован.

## Назначение эксперимента. Выбор и обоснование методики проведения тестирования

В данном разделе проведены два вида тестирования. Провести анализ тестирования и выбрать, как будет проходить тестирование ИС.

Модульное тестирование Цель: проверить, что код работает именно так, как должен (при заданных входных параметрах выдает предполагаемый результат).

Анализ покрытия кода тестами. Цель: проверить, что весь наш код отрабатывает при модульном тестировании, что нет не участвующих в тестировании участков кода.

Анализ стилистики кода. Цель: проверка кода на соблюдение стандартов разработки (отраслевых, корпоративных, принятых на проекте).

Анализ производительности. Цель: проанализировать производительность ключевых операций, потребление памяти, утечку памяти.

## Технология тестирования

Рассмотрим, какие технологии можно применить для модульного тестирования и покрытия кода тестами.

Как в ходе выполнения проекта, так и при развитии системы, изменения в ее алгоритмы вносятся в соответствии с техническими заданиями, хотя данный документ может иметь различные названия. В технических заданиях, как правило, описываются критерии приемки: как мы и заказчик убедимся в том, что заказанный функционал работает верно.

В качестве примера можно привести такой упрощенный критерий:

* Исходные данные: в систему внесен сорт растения, который в дальнейшем приобретёт покупатель.
* Действия пользователя: пользователь ищет в таблице упаковку семян с определённым сортом растений и сколько он будет стоить и т п.
* В результате в результате отображена цена упаковки семян 500 руб.
* Вручную такое тестирование, как правило, выполняется единожды, поскольку выполнять каждый раз полный комплект тестов очень трудоемко.
* Нет гарантии того, что тесты, описанные в техническом задании, покрывают функциональность полностью. Т.е. даже если тесты описаны и проводятся регулярно, вероятно наличие не протестированного функционала.

Автоматизированное тестирование лишено данных недостатков. Конечно, возникает отдельная задача кодирования данного теста на основании критериев приемки, но это скорее дисциплинирующий фактор…

Прежде чем начинать юзабилити-тестирование необходимо собрать группу пользователей, которые будут тестировать данную информационную систему. Количество привлеченных пользователей должно быть не менее пяти человек. Пользователям будет представлен список некоторых вопросов, относящихся к графическому интерфейсу.

В таблице 55 представлены сведения о пользователях.

Таблица 55 – Информация о пользователях

|  |  |  |
| --- | --- | --- |
| ФИО | Пол | Образование |
| Аверьянов Николай Владимирович | Мужской | Среднее специальное |
| Туркина Надежда Михайловна | Женский | Среднее специальное |
| Окладников Антон Александрович | Мужской | Среднее специальное |
| Петрова Надежда Михайловна | Женский | Среднее специальное |
| Фурман Ксения Михайловна | Женский | Среднее специальное |

## Результаты проведения тестирования

В ходе тестирования был применен метод нагрузочного тестирования и юзабилити-тестирования, которые проверяют работу при нагрузке системы и сделать простым и удобным в пользовании информационной системы.

Суть тестирования заключается в проверке работы всех обязательных функций.

Результаты проведения нагрузочного тестирования

Для проведения нагрузочного тестирования составлена таблица 22, в которой указано количество запущенных копий программного продукта, нагрузка на процесс и оперативную память.

Таблица 56 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| Число копий | Нагрузка на процессор (%) | Нагрузка на память (%) |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

Исходя из результатов тестирования, можно прийти к выводу, что нагрузка, оказываемая на центральный процессор и память, не имеет критичных значений. Это значит, что показатель производительности информационной системы на приемлемом уровне, то есть сбои и длительная загрузка не должны помешать пользователю при работе с данным программным продуктом.

Таблица 57 – Результаты проведения юзабилити-тестирования

| Вопрос | Аверьянов Николай Владимирович | Окладников Антон Александрович | Туркина Надежда Михайловна | Петрова Надежда Михайловна | Фурман Ксения Михайловна |
| --- | --- | --- | --- | --- | --- |
| Присутствуют ли такие дизайн-элементы, которые неприятны для зрения? | нет | нет | нет | нет | нет |
| Присутствует ли отвлекающая информация? | нет | нет | нет | нет | нет |
| Понятна ли структура приложения на интуитивном уровне? | да | да | да | да | да |
| Смогли ли вы найти интересующую вас информацию без затруднений? | да | да | да | да | нет |
| Понятен ли смысл приложения? | да | да | да | да | да |
| Общая оценка | 7/10 | 10/10 | 8/10 | 9/10 | 6/10 |

Исходя из средней оценки информационной системы, рассчитанной на основе общих оценок всех пользователей с учетом ответов на все поставленные вопросы, можно сделать вывод о том, что внешние характеристики графического интерфейса находятся на довольно высоком уровне. Также в информационной системе нет помех для пользователя в виде отвлекающей информации, так как все отображаемые данные находятся в рамках предметной области.

Вывод по главе: были проведены тесты путём проведения модульного тестирования, анализа покрытия кода тестами, анализа стилистики кода и анализом производительности. Также было проведено нагрузочное тестирование, в результате чего нагрузка на процессор и память не имела критичных значений.

# ЗАКЛЮЧЕНИЕ

В результате учебной практики был проведен анализ предметной области, спроектирована и разработана информационная система «Автосалон», серверная часть которой была реализована в MS SQL, а клиентская часть в MS Visual Studio.

В MS SQL была создана база данных для хранения разработана и спроектировать прикладное программное обеспечение для эффективного управления продажами автомобилей.

Клиентская часть, реализованная в MS Visual Studio, отражает все хранящиеся в базе данных таблицы. Также разработаны некоторые запросы, позволяющие редактировать и удалять существующие данные, добавлять новые и осуществлять поиск по таблицам.

Цели учебной практики были достигнуты путем проведения анализа предметной области, проектирования информационной системы и разработки программных модулей.

Это позволяет сделать вывод, что поставленные цели и задачи выполнены в полном объеме.

В дальнейшем данная информационная система может быть модернизирована.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения [Электронный доступ]. — Режим доступа: [ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения (rostest.info)](https://rostest.info/gost/001.001.040.035/gost-20886-85/#:~:text=%D0%93%D0%9E%D0%A1%D0%A2%2020886%2D85.%20%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B2,%D0%BD%D0%B0%D1%83%D1%87%D0%BD%D0%BE%2D%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%) (Дата обращения: 01.12.2023 г.)
2. ГОСТ Р. 59793–2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания [Электронный доступ]. — Режим доступа: [ГОСТ Р. 59793-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания (internet-law.ru)](https://internet-law.ru/gosts/gost/77858/?ysclid=lifap277mg519765664) (Дата обращения: 01.12.2023 г.).
3. ГОСТ 19.201-78. ЕСПД. Общие положения. [Электронный доступ]. — Режим доступа: [ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению (swrit.ru)](https://www.swrit.ru/doc/espd/19.201-78.pdf?ysclid=lpi5u6naom706376064) (Дата обращения: 01.12.2023 г.).
4. Аллен, Г. Тейлор SQL для чайников — М.: Диалектика, Вильямс, 2015. — 416 c.
5. Грабер Мартин. SQL для простых смертных — М: «ЛОРИ», 2014. — 378 с.
6. Microsoft Visual Studio [Электронный доступ]. — Режим доступа: [Руководство по разработке модулей расширений на C# — Павел Еремеев (pvs-studio.ru)](https://pvs-studio.ru/ru/blog/posts/csharp/a0082/) (Дата обращения: 01.12.2023 г.).
7. Кариев Ч. А. Разработка Windows-приложений на основе Visual C# (+ CD-ROM) — М: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2018. — 768 c.
8. Microsoft SQL Server [Электронный доступ]. — Режим доступа: [Microsoft SQL Server — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server) (Дата обращения: 01.12.2023 г.).
9. Шаша Д., Бонне Ф. Оптимизация баз данных. Принципы, практика, решение проблем — М.: «КУДИЦ - Образ», 2004. — 586с.
10. Microsoft Visual Studios [Электронный доступ]. — Режим доступа: [Microsoft Visual Studios — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) (Дата обращения: 01.12.2023 г.).

# ПРИЛОЖЕНИЕ А

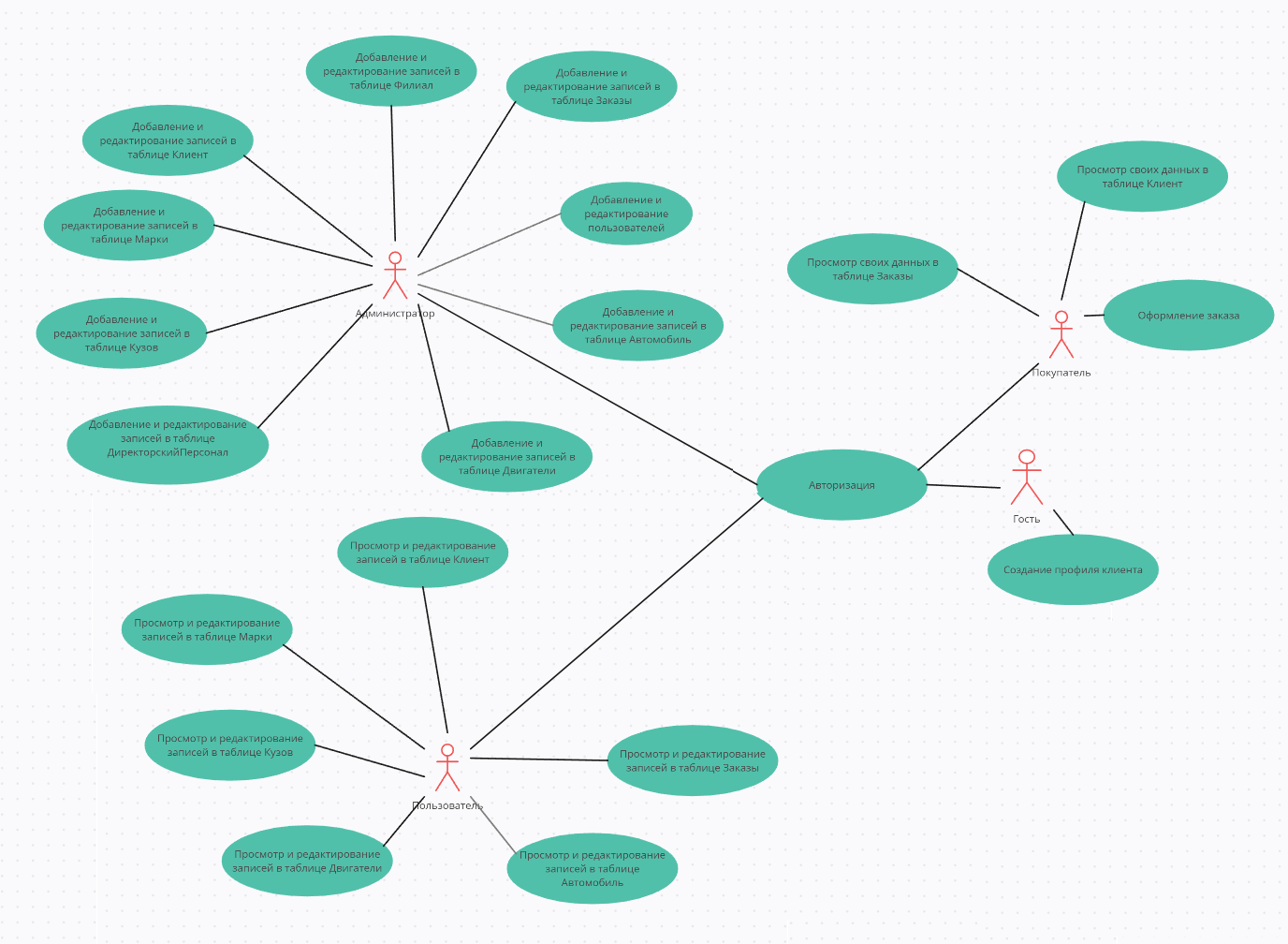


Рисунок 1 – Диаграмма вариантов использования

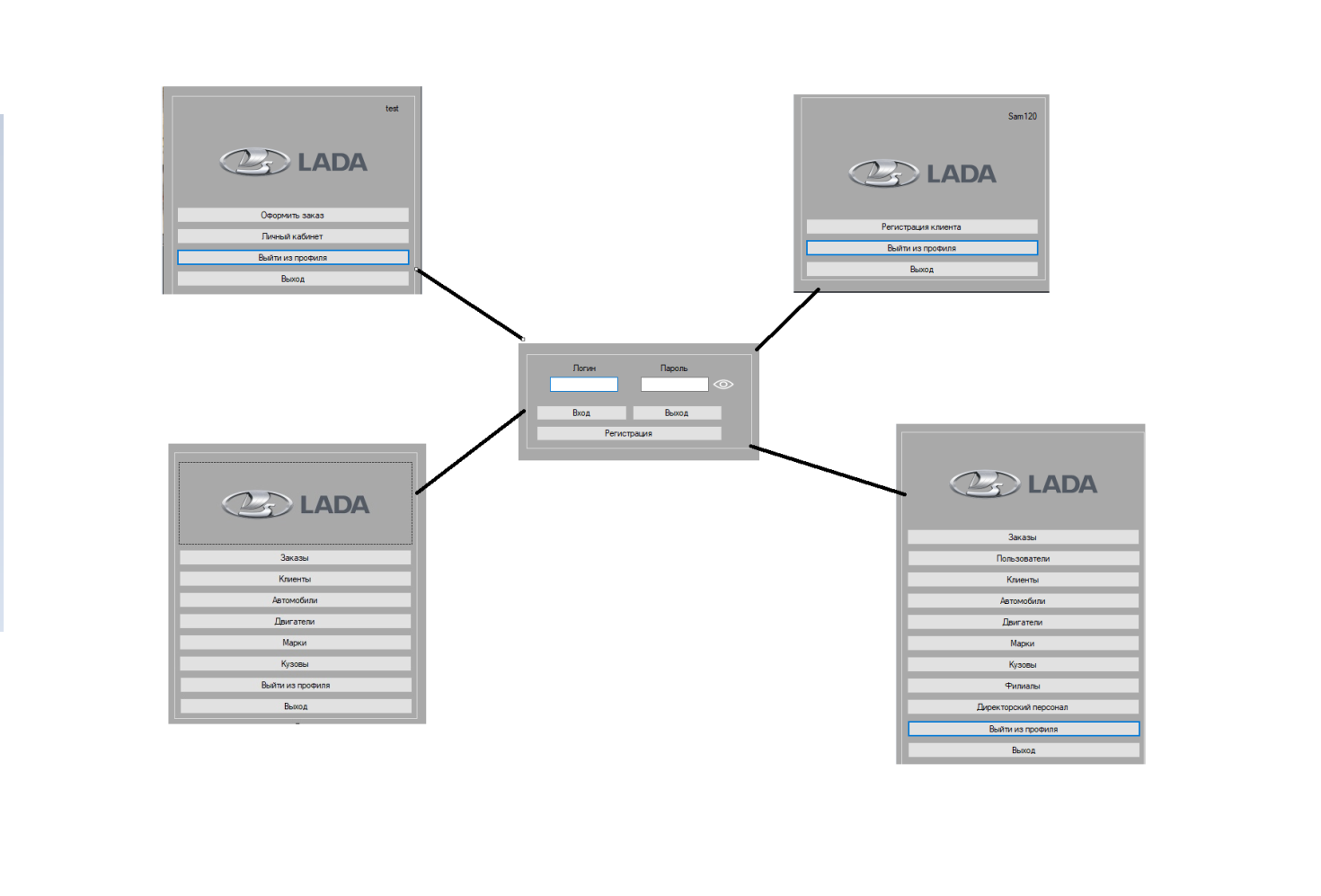


Рисунок 6 – Карта переходов

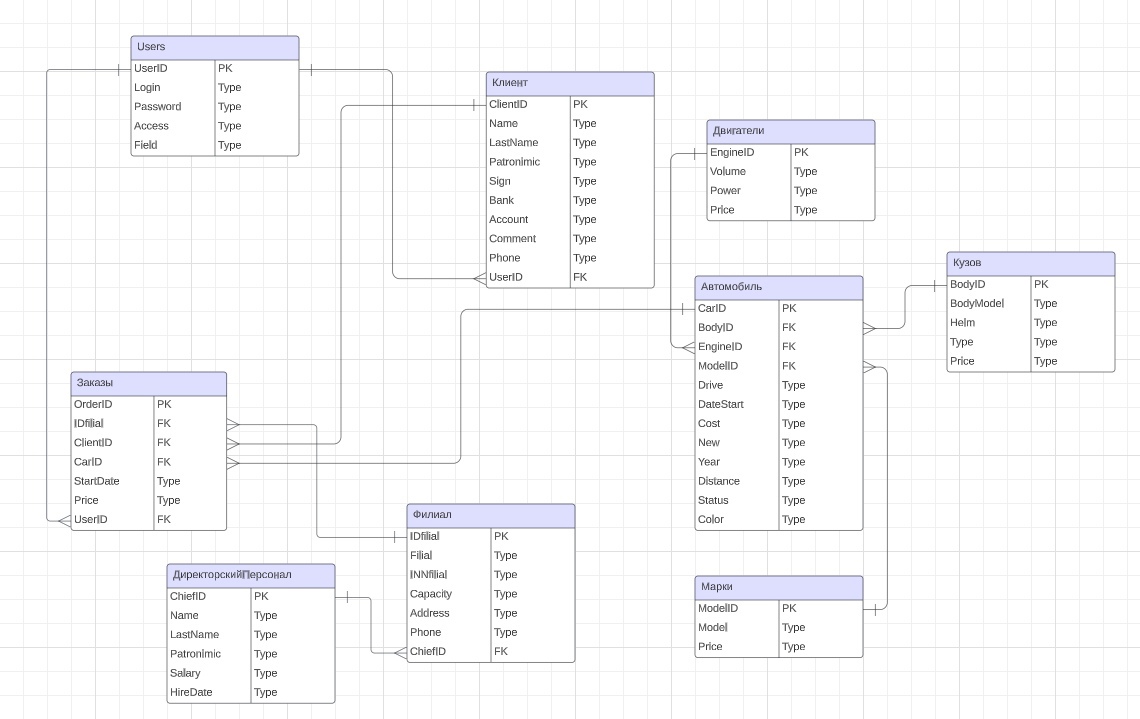


Рисунок 12 – Логическая модель данных

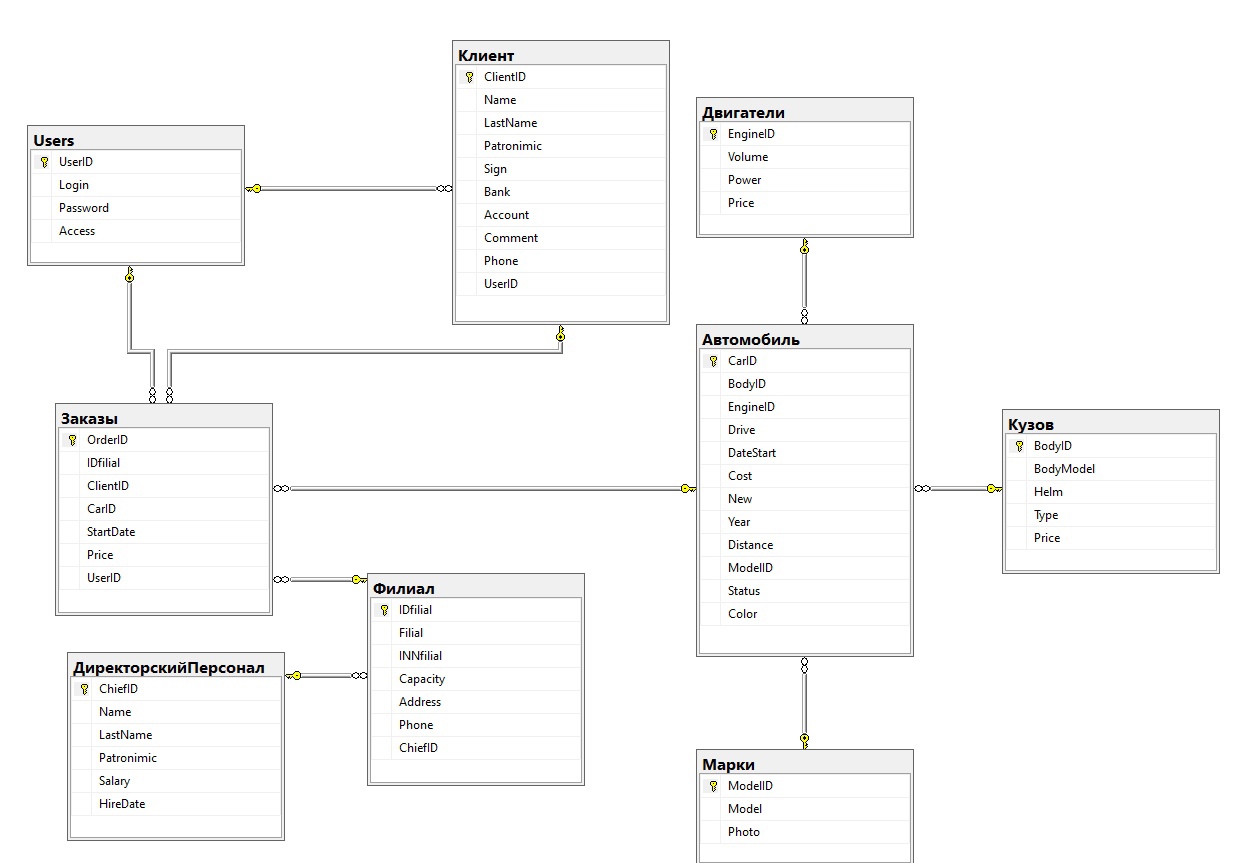


Рисунок 14 – Физическая модель базы данных

# ПРИЛОЖЕНИЯ Б

Листинг кода формы “Авторизация”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace autoShop

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

pictureBox1.Image = pictureBox1.InitialImage;

}

private int loginAttempts = 5;

bool hidden = true;

private void button1\_Click(object sender, EventArgs e)

{

string login = textBox1.Text.Trim();

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Не введён логин или пароль");

return;

}

if (loginAttempts <= 0)

{

MessageBox.Show("Превышено количество попыток входа. Обратитесь к администратору.");

this.Close();

return;

}

SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog = autoShop; Integrated Security = True");

try

{

sqlConnect.Open();

SqlDataAdapter da = new SqlDataAdapter("select \* from Users where Login = @login", sqlConnect);

da.SelectCommand.Parameters.AddWithValue("@login", login);

DataTable dt = new DataTable();

da.Fill(dt);

if (dt.Rows.Count == 0)

{

MessageBox.Show("Пользователь с таким логином не существует.");

loginAttempts--;

}

else

{

string dbPassword = dt.Rows[0]["Password"].ToString();

string role = dt.Rows[0]["Access"].ToString();

RegisteredUser currentUser = userLog(login);

if (password == dbPassword)

{

switch (role)

{

case "admin":

МенюАдмина adminForm = new МенюАдмина();

adminForm.Show();

this.Hide();

break;

case "user":

МенюПользователя userForm = new МенюПользователя();

userForm.Show();

this.Hide();

break;

case "buyer":

МенюПокупателя brForm = new МенюПокупателя(currentUser);

brForm.Show();

this.Hide();

break;

case "guest":

МенюГостя gstForm = new МенюГостя(currentUser);

gstForm.Show();

this.Hide();

break;

}

}

else

{

MessageBox.Show("Неверный Пароль.");

loginAttempts--;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Опибка при входе: " + ex.Message);

}

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

if (hidden == false)

{

pictureBox1.Image = Properties.Resources.glazik;

textBox2.PasswordChar = '\*';

hidden = true;

}

else

{

pictureBox1.Image = Properties.Resources.glazik\_zakryt;

textBox2.PasswordChar = '\0';

hidden = false;

}

}

private void button3\_Click(object sender, EventArgs e)

{

Регистрация regForm = new Регистрация();

regForm.Show();

this.Hide();

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private RegisteredUser userLog(string login)

{

RegisteredUser currentUser = new RegisteredUser();

using (SqlConnection connection = new SqlConnection("Data Source=localhost;Initial Catalog = autoShop; Integrated Security = True"))

{

try

{

// Открываем соединение

connection.Open();

string selectQuery = "SELECT UserID FROM Users WHERE Login = @EnteredLogin";

using (SqlCommand command = new SqlCommand(selectQuery, connection))

{

// Задаем параметр для безопасного использования в запросе

command.Parameters.AddWithValue("@EnteredLogin", login);

// Выполняем запрос и получаем результат

object result = command.ExecuteScalar();

// Проверяем, был ли найден UserID

if (result != null)

{

currentUser.UserID = Convert.ToInt32(result);

currentUser.UserName = login;

}

else

{

Console.WriteLine("Пользователь с таким логином не найден.");

}

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при выполнении запроса: " + ex.Message);

}

}

return currentUser;

}

}

}

Листинг кода формы “Регистрация”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace autoShop

{

public partial class Регистрация : Form

{

public Регистрация()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string login = textBox1.Text.Trim();

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Не введён логин или пароль");

return;

}

SqlConnection sqlConnect = new SqlConnection("Data Source=localhost;Initial Catalog = autoShop; Integrated Security = True");

SqlDataAdapter da = new SqlDataAdapter("select \* from Users where Login = @login", sqlConnect);

da.SelectCommand.Parameters.AddWithValue("@login", login);

DataTable dt = new DataTable();

da.Fill(dt);

if (dt.Rows.Count != 0)

{

MessageBox.Show("Пользователь с таким логином уже существует.");

}

else

{

if (textBox2.Text != textBox3.Text)

{

MessageBox.Show("Пароли не совпадают");

}

else

{

using (SqlCommand cmd = new SqlCommand("Insert Into Users (Login, Password, Access) Values (@login, @password, 'guest')", sqlConnect))

{

cmd.Parameters.AddWithValue("@login", login);

cmd.Parameters.AddWithValue("@password", password);

try

{

sqlConnect.Open();

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

// Регистрация успешно завершена

MessageBox.Show("Регистрация успешно завершена!");

Form2 logForm = new Form2();

logForm.Show();

this.Close();

}

else

{

// Вставка не удалась, обработайте это по необходимости

MessageBox.Show("Ошибка при регистрации. Пожалуйста, попробуйте снова.");

}

}

catch (Exception ex)

{

// Обработка ошибок при выполнении SQL-запроса

MessageBox.Show("Ошибка при регистрации: " + ex.Message);

}

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 logForm = new Form2();

logForm.Show();

this.Close();

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

Листинг кода формы “Администратор”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace autoShop

{

public partial class МенюАдмина : Form

{

public МенюАдмина()

{

InitializeComponent();

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

УправлениеПользователями userLog = new УправлениеПользователями();

userLog.Show();

this.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button3\_Click(object sender, EventArgs e)

{

Form2 logForm = new Form2();

logForm.Show();

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

ПросмотрЗаказовADM order = new ПросмотрЗаказовADM();

order.Show();

this.Close();

}

private void button6\_Click(object sender, EventArgs e)

{

КлиентыADM client = new КлиентыADM();

client.Show();

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

АвтомобилиADM car = new АвтомобилиADM();

car.Show();

this.Close();

}

private void button7\_Click(object sender, EventArgs e)

{

ДвигателиADM engine = new ДвигателиADM();

engine.Show();

this.Close();

}

private void button8\_Click(object sender, EventArgs e)

{

МаркиADM model = new МаркиADM();

model.Show();

this.Close();

}

private void button9\_Click(object sender, EventArgs e)

{

КузовыADM body = new КузовыADM();

body.Show();

this.Close();

}

private void button10\_Click(object sender, EventArgs e)

{

Филиалы branch = new Филиалы();

branch.Show();

this.Close();

}

private void button11\_Click(object sender, EventArgs e)

{

ДиректорскийПерсонал staff = new ДиректорскийПерсонал();

staff.Show();

this.Close();

}

}

}

Листинг кода формы “Продавец”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace autoShop

{

public partial class МенюПокупателя : Form

{

RegisteredUser currentUser;

public МенюПокупателя(RegisteredUser user)

{

InitializeComponent();

currentUser = user;

label1.Text = currentUser.UserName;

}

private void button4\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button3\_Click(object sender, EventArgs e)

{

Form2 logForm = new Form2();

logForm.Show();

this.Close();

}

private void button5\_Click(object sender, EventArgs e)

{

ЛичныйКабинет profile = new ЛичныйКабинет(currentUser);

profile.Show();

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

ОформлениеЗаказа createOrder = new ОформлениеЗаказа(currentUser);

createOrder.Show();

this.Close();

}

}

}

Листинг кода формы “Меню пользователя”:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace autoShop

{

public partial class МенюПользователя : Form

{

public МенюПользователя()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

ПросмотрЗаказовUSER order = new ПросмотрЗаказовUSER();

order.Show();

this.Close();

}

private void button6\_Click(object sender, EventArgs e)

{

КлиентUSER client = new КлиентUSER();

client.Show();

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

АвтомобилиUSER car = new АвтомобилиUSER();

car.Show();

this.Close();

}

private void button7\_Click(object sender, EventArgs e)

{

ДвигателиUSER engine = new ДвигателиUSER();

engine.Show();

this.Close();

}

private void button8\_Click(object sender, EventArgs e)

{

МаркиUSER model = new МаркиUSER();

model.Show();

this.Close();

}

private void button9\_Click(object sender, EventArgs e)

{

КузовыUSER body = new КузовыUSER();

body.Show();

this.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

Form2 logForm = new Form2();

logForm.Show();

this.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

Application.Exit();

}

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

{

}

}

}