

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий
Кафедра Программной инженерии
Специальность 1-40 01 01 Программное обеспечение информационных технологий
Направление специальности 1-40 01 01 Программное обеспечение информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА:

по дисциплине «Объектно-ориентированные технологии проектирования и стандарты проектирования»
Тема Программное средство «Лизинг авто»

Исполнитель

студент 2 курса группы 6 Яхимчик Д.Ю
(Ф.И.О.)

Руководитель работы ассистент Мущук А.Н
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой _____
Председатель Пацей Н.В.
(подпись)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»
Факультет информационных технологий
Кафедра программной инженерии

Утверждаю
Заведующий кафедрой ПИ
Н.В. Пацей
подпись инициалы и фамилия

“ ____ ” 2022г.

ЗАДАНИЕ
к курсовому проектированию
по дисциплине "Объектно-ориентированные технологии
программирования и стандарты проектирования"

Специальность: 1-40 01 01 Программное обеспечение информационных технологий Группа: 6

Студент: Яхимчик Д.Ю.

Тема: Программное средство «Лизинг авто»

1. Срок сдачи студентом законченной работы: "20 мая 2022 г."

2. Исходные данные к проекту:

2.1. Функционально ПС поддерживает:

- Функции администратора сервиса:
 - Поддерживать работу с базой данных;
 - Добавление и удаление авто;
 - Редактировать данные об автомобиле;
 - Просмотр информации о клиентах;
- Функции клиента:
 - Выполнять регистрацию и авторизацию;
 - Поиск и фильтрация авто;
 - Оставить заявку на оформления лизинга ;
 - Отмечать понравившиеся авто;

2.2. При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес-логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

3. Содержание расчетно-пояснительной записи

- Введение
- Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
- Проектирование архитектуры проекта (структура модулей, классов).

- Разработка функциональной модели и модели данных ПС (выполняемые функции)
- Тестирование
- Заключение
- Список используемых источников
- Приложения

4. Форма представления выполненной курсовой работы:

- Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записи должно быть согласно выданным правилам.
- Листинги программы представляются частично в приложении.
- Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

Календарный план

№ п/п	Наименование этапов курсового проекта	Срок выполнения этапов проекта	Примечание
1	Введение	19.02.2022	
2	Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования	12.03.2022	
3	Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных)	26.03.2022	
4	Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса	2.04.2022	
5	Кодирование программного средства	23.04.2022	
6	Тестирование и отладка программного средства	30.04.2022	
7	Оформление пояснительной записи	7.05.2022	
9	Сдача проекта	20.05.2022	

5. Дата выдачи задания 12.02.2022

Руководитель A.H. Муциук
(подпись)

Задание принял к исполнению _____
(дата и подпись студента)

Содержание

Содержание.....	4
Введение	5
1 Анализ прототипов и литературных источников	6
2 Анализ требований к программному средству и разработка функциональных требований	7
3 Проектирование программного средства	10
3.1 Проектирование архитектуры приложения	10
3.2 Проектирование базы данных	11
3.3 Проектирование вариантов использования	12
4 Реализация программного средства.....	13
4.1 Реализация сущностей.....	13
4.2 Реализация паттерна проектирования MVVM	14
4.3 Реализация авторизации и регистрации	16
4.4 Реализация перехода между интерфейсами.....	17
5 Тестирование	18
6 Руководство по использованию	21
6.1 Руководство по использованию пользователем	21
6.2 Руководство по использованию администратором.....	24
Заключение	27
Список используемых источников.....	28

Введение

Сфера информационных технологий является одной из наиважнейших в современном обществе. Но не все процессы можно перенести в виртуальный мир. Однако информационные технологии могут помочь сэкономить время.

Приложение «Лизинг авто» предназначено для экономии времени, и помощи работы с клиентами. Программа помогает людям выбрать подходящее им авто, не посещая лично лизинговое агентство, и не взаимодействуя с сотрудниками лично. Тем самым экономя личное время, и время сотрудников. Программа интуитивно понятна для пользователей, так как не требует квалифицируемых ИТ-ресурсов.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — C#. Проект реализован на платформе .NET Core.

1 Анализ прототипов и литературных источников

Одним из первых этапов в создании программного продукта является анализ прототипов и литературных источников. В данном разделе рассматриваются программные средства с аналогичной тематикой с целью выделения преимуществ и недостатков. Разработка своего программного продукта будет основываться на их анализе. Большинство приложений реализованы как веб-приложения, либо для мобильных устройств.

ilz.by

Это сайт лизингового агентства в городе Минск. Имеет простой интерфейс и малый функционал. Интерфейс представлен на рисунке 1.1.

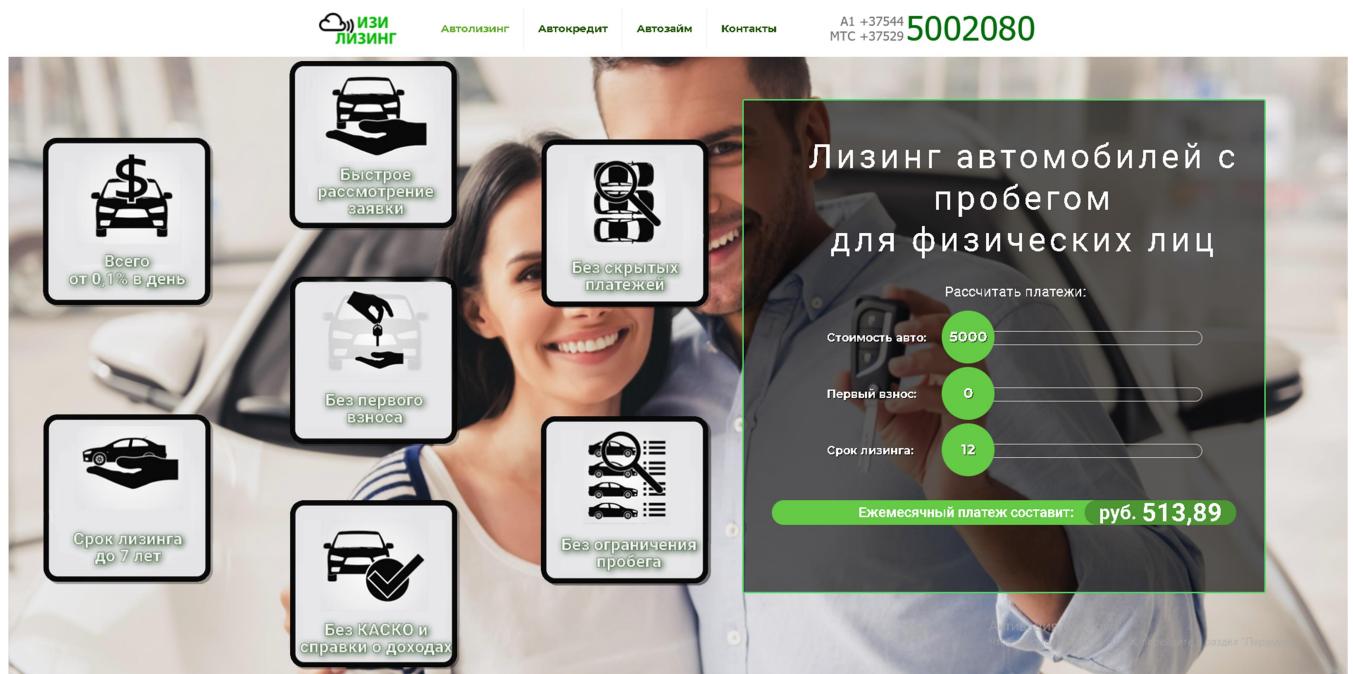


Рисунок 1.1 — Интерфейс «Изи лизинг»

Проанализировав «Изи лизинг», можно выделить минусы и плюсы данного программного средства.

Основные минусы:

- Нельзя посмотреть авто;
- Нельзя оставить заявку на сайте;
- Малый общий функционал;
- Неудобная навигация.

Основные плюсы:

- Простой и понятный интерфейс;

2 Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

- бизнес-требования;
- пользовательские требования;
- функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

- простота и лёгкость интерфейса;
- использование принципов объектно-ориентированного программирования;
- использование архитектурных шаблонов проектирования;
- использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность:

- регистрация и вход в систему;
- просматривать каталог авто;
- выполнять поиск по каталогу;

- фильтровать каталог;
- добавлять авто в понравившиеся;
- оставить заявку на лизинг.

Администратор имеет возможность:

- Просматривать заявки;
- Удалять заявки;
- Добавить/удалить авто;
- Просматривать информацию о клиенте;

После проведения анализа были выявлены следующие функциональные требования:

- архитектура приложения должна соответствовать шаблону проектирования, такому как MVVM;
- вся информация должна храниться в базе данных;
- приложение должно производить валидацию вводимых пользователем данных;
- приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
- приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
- приложение должно предоставлять пользователям возможность аутентификации, используя введенные данные;

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

В разработке приложения были использованы нижеперечисленные технологии.

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С WPF можно создавать широкий спектр как автономных приложений, так и приложений, размещенных в веб-обозревателе. В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений. Именно использование WPF позволило гибко управлять дизайном интерфейса, также стало возможным подключение различных сторонних пакетов и использование паттернов.

Entity Framework Core — представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отобра-

жения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

3 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

3.1 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

- выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
- соединение выбранных элементов структуры и поведения во всё более крупные системы;
- архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

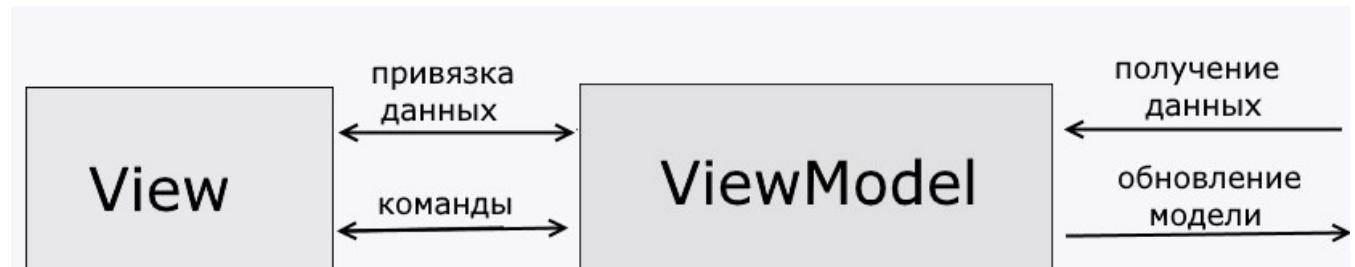


Рисунок 3.1 — Структура шаблона MVVM

View — визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

ViewModel — модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса `INotifyPropertyChanged` автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также ViewModel определяет логику по обновлению данных в модели. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

3.2 Проектирование базы данных

Проект включает в себя 4 сущностей:

- пользователи;
- авто;
- Понравившиеся авто;
- Заявки на лизинг;

Для создания базы данных использовался подход Database First технологии Entity Framework Core. Логическая модель базы данных представлена на рисунке 3.2.

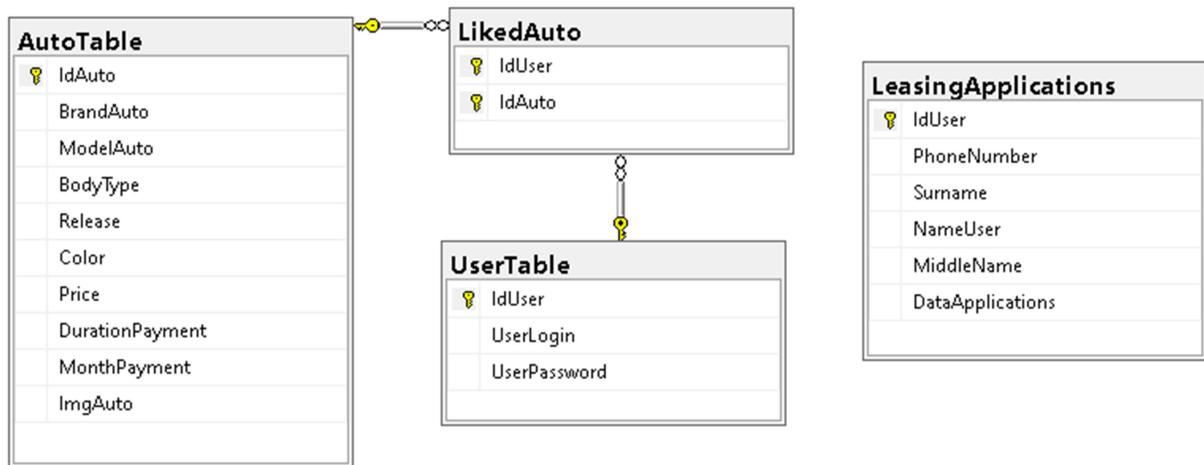


Рисунок 3.2 — Диаграмма базы данных

Таблица `UsersTable`:

- `IdUser` — Id пользователя
- `UserLogin` — логин пользователя

- UserPassword — пароль пользователя

Таблица LikedAuto (служит для осуществления связи многие-ко-многим):

- IdUser — Id пользователя
- IdAuto — Id авто

Таблица AutoTable:

- IdAuto — Id авто
- BrandAuto — Марка машины
- ModelAuto — Модель машины
- BodyType — Тип кузова
- Release — Год выпуска
- Color — Цвет авто
- Price — Цена авто
- DurationPayment — Продолжительность оплаты
- MonthPayment — Длительность оплаты
- ImgAuto — Изображение авто

Таблица LeasingApplications:

- IdUser — ключевое поле
- PhoneNumber — номер телефона
- Surname — Фамилия
- NameUser — Имя
- MiddleName — Отчество
- DataApplications — Дата заявки

3.3 Проектирование вариантов использования

На рисунке 3.3 представлена диаграмма использования приложения для пользователя и администратора. Данная диаграмма представлена в приложении А.

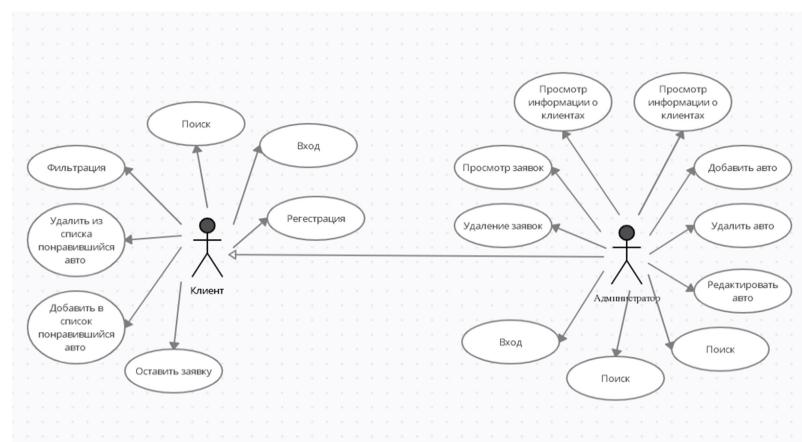


Рисунок 3.3 — Диаграмма использования

Данная диаграмма представляет общий функционал Администратора и Клиента.

4 Реализация программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Database-First. При данном подходе модель EDMX не используется. Создание базы данных происходит из созданной вручную базы данных. Диаграмма UML для сущностных классов представлена на рисунке 4.1.

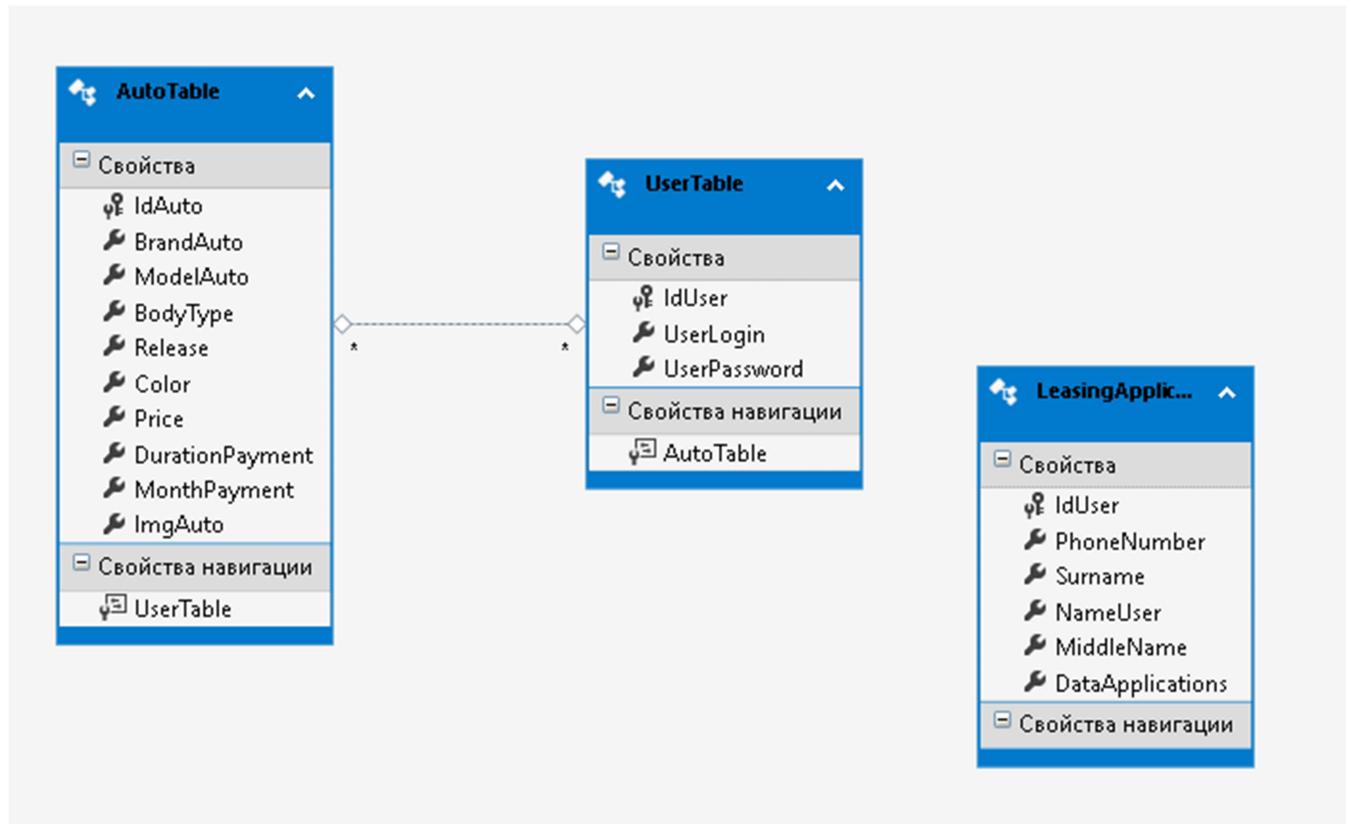


Рисунок 4.1 — Диаграмма классов сущностей БД

Таблицы, используемые в Базе данных проекта.

4.2 Реализация паттерна проектирования MVVM

Для облегчения навигации по файлам, проект был разделен на одноименные папки (рисунок 4.2).

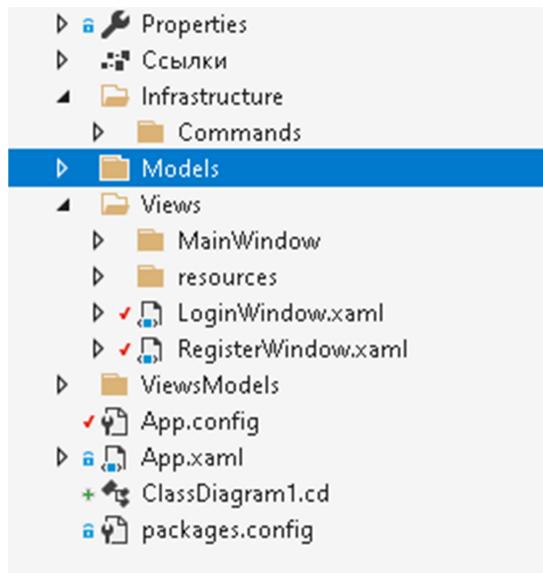


Рисунок 4.2 — Структура проекта

Папка resources содержит картинки.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. Данный класс представлен в листинге 4.1.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace leasingAgency.Infrastructure.Commands.Base
{
    ссылка: 1
    internal abstract class Command : ICommand
    {
        public event EventHandler CanExecuteChanged
        {
            add => CommandManager.RequerySuggested += value;
            remove => CommandManager.RequerySuggested -= value;
        }

        ссылка: 2
        public abstract bool CanExecute(object parameter);

        ссылка: 2
        public abstract void Execute(object parameter);
    }
}
```

Листинг 4.1 — Базовый класс Command

Класс реализует два метода:

- CanExecute: определяет, может ли команда выполняться;

- Execute: собственно, выполняет логику команды.

Для создания команд из ViewModel используется класс LambdaCommand, который наследуется от Command (листинг 4.2).

```

using leasingAgency.Infrastructure.Commands.Base;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace leasingAgency.Infrastructure.Commands
{
    Ссылок: 41
    internal class LambdaCommand : Command
    {
        private readonly Action<object> _Execute;
        private readonly Func<object, bool> _CanExecute;
        Ссылок: 40
        public LambdaCommand(Action<object> Execute, Func<object, bool> CanExecute = null)
        {
            _Execute = Execute;
            _CanExecute = CanExecute;
        }

        Ссылок: 0
        public Func<object, bool> CanExecute { get; }

        Ссылок: 2
        public override bool CanExecute(object parameter) => _CanExecute?.Invoke(parameter) ?? true;
        Ссылок: 2
        public override void Execute(object parameter) => _Execute(parameter);
    }
}

```

Листинг 4.2 — Класс LambdaCommand

Все классы ViewModel в проекте наследуются от абстрактного класса ViewModel (Листинг 4.3), который реализует интерфейс INotifyPropertyChanged. Это видно на UML диаграмме на рисунке 4.3.

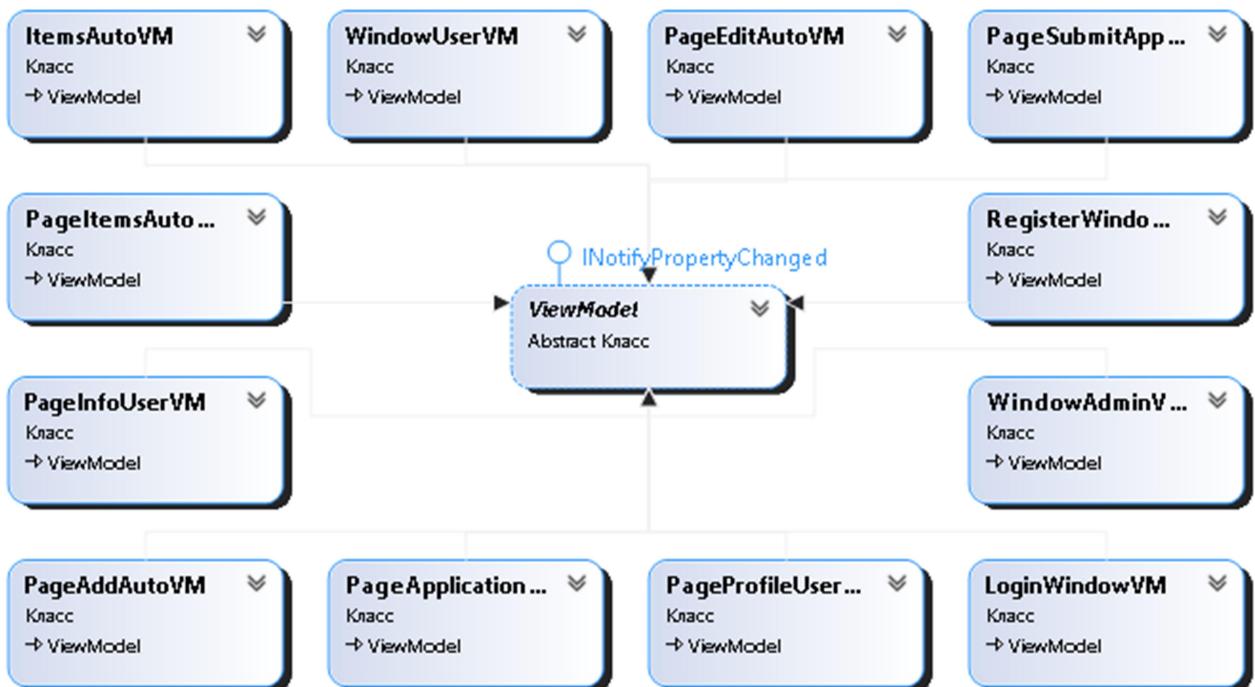


Рисунок 4.3 — Диаграмма классов производных от ViewModel

Класс ViewModel реализует метод Set, который применяет перегруженный метод OnPropertyChanged на свойство.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace leasingAgency.ViewsModels.Base
{
    //Базовый класс модели-представления

    Ссылка: 12
    internal abstract class ViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        Ссылка: 1
        protected virtual void OnPropertyChanged([CallerMemberName] stringPropertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(PropertyName));
        }

        Ссылка: 77
        protected virtual bool Set<T>(ref T field, T value, [CallerMemberName] stringPropertyName = null)
        {
            if(Equals(field, value)) return false;
            field = value;
            OnPropertyChanged(PropertyName);
            return true;
        }
    }
}
```

Листинг 4.3 — Класс ViewModel

4.3 Реализация авторизации и регистрации

Для того, чтобы пользоваться приложением, необходимо для начала зарегистрироваться в системе. В приложении Б представлен код, реализующий регистрацию.

Если пользователь уже зарегистрирован, то ему следует войти в систему. В приложении В представлен код реализующий вход в систему.

В базе данных хранятся хеш паролей пользователей. Алгоритмом хеширования был выбран MD5, его реализацию можно видеть в листинге 4.4.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace leasingAgency.Models
{
    Ссылка: 2
    public static class PasswordManager
    {
        Ссылка: 2
        public static string GetHash(string input)
        {
            var md5 = MD5.Create();
            var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

            return Convert.ToBase64String(hash);
        }
    }
}
```

Листинг 4.4 — Реализация хеширование MD5

В приложении много полей для ввода. Валидация в них реализована с помощью регулярных выражений. Данные выражение собраны в классе RegexForm. Он представлен на рисунке 4.5.

```
Ссылок: 20
public static class RegexForm
{
    static private string regexModel = @"^(?!^[0-9]*$)([a-zA-Z0-9]{0,20})$";
    Ссылок: 0
    static public bool RegexModelFunc(string str)
    {
        return Regex.IsMatch(str, regexModel, RegexOptions.None);
    }

    static private string regexLogin = @"^(?!^[0-9]*$)([a-zA-Z0-9]{3,15})$";
    Ссылок: 2
    static public bool RegexLoginFunc(string str) {
        return Regex.IsMatch(str, regexLogin, RegexOptions.None);
    }

    static private string regexPassword = @"^(?!^[0-9]*$)(?!^[a-zA-Z]*$)^([a-zA-Z0-9]{6,20})$";
    Ссылок: 2
    static public bool RegexPasswordFunc(string str){
        return Regex.IsMatch(str, regexPassword, RegexOptions.None);
    }

    static private string regexOnlyNumber = @"^-[0-9]+$";
    Ссылок: 8
    static public bool RegexOnlyNumberFunc(string str)
    {
        return Regex.IsMatch(str, regexOnlyNumber, RegexOptions.None);
    }

    static private string regexPhoneNumber = @"^-[0-9]{9}$";
    Ссылок: 1
    static public bool RegexPhoneNumberFunc(string str)
    {
        return Regex.IsMatch(str, regexPhoneNumber, RegexOptions.None);
    }

    static private string regexName = @"^([a-zA-Zа-яА-Я]{1,20})$";
    Ссылок: 7
    static public bool RegexNameFunc(string str)
    {
        return Regex.IsMatch(str, regexName, RegexOptions.None);
    }
}
```

Листинг 4.5 — Регулярные выражения

4.4 Реализация перехода между интерфейсами

Приложение включает в себя 4 окна и 8 страниц.

Для перехода между окнами и страницами используется класс WindowsManager, представленный Приложении Г.

Для работы с блоками View используется привязка к своему ViewModel (листинг 4.7).



```
<TextBox Grid.Row="1"
         Text="{Binding Path=TextBoxLogin, UpdateSourceTrigger=PropertyChanged}"
         TextWrapping="Wrap" BorderThickness="2,2,2,2" Margin="20,0"/>
```

Листинг 4.7 — Привязка View к ViewModel

Далее привязывать определенный объект ViewModel к View. Продемонстрировано на листинге 4.8.

```
#region Логин

/// <summary> Логин </summary>

private bool flagLogin = false;

private string _TextBoxLogin;

Ссылок: 3
public string TextBoxLogin
{
    get => _TextBoxLogin;

    set {
        regexCheckLogin(value);
        Set(ref _TextBoxLogin, value);
    }
}
```

Листинг 4.8 — ViewModel

5 Тестирование

В курсовом проекте проверяются данные которые вводятся пользователем, непосредственно в момент ввода данных. Кроме полей для пароля, так как привязка не используется к данным полям для безопасности пользователя. Пример представлен на рисунке 5.1.

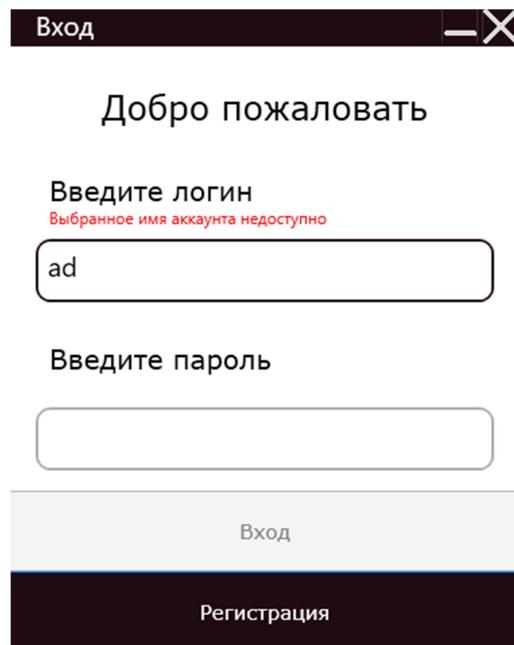


Рисунок 5.1 — Ошибка валидации

Также сообщения о валидации выводятся прямо в интерфейс (рисунок 5.2).

Регистрация

Зарегистрируйтесь

Логин
Выбранное имя аккаунта недоступно

Пароль
Выбранный пароль недоступен

Пароль должен состоять из комбинации цифр и букв, и состоять не менее чем из 6 символов и не более чем из 20 символов.

Повтор пароля
Пароли не совпали

Зарегистрироваться

Вернуться к входу

Рисунок 5.2 — Ошибка валидации

Если все требования не выполнены, то кнопка, на которую привязана команда, будет неактивна (рисунок 5.3).

Админ панель

Заявки

Авто Добавить авто Выход

Марка	Модель
Тип кузова	Год выпуска
Цвет	Цена BYN
Время оплаты (мес)	Изображение

Кузов:
Цвет:
Год выпуска:
Цена:

Добавить авто

Очистить поля

Рисунок 5.3 — Неактивная кнопка

Это реализуется благодаря условию, указанному в CanExecute команды.
Пример реализации расположен в листинге 5.1.

```
#region AddAuto
 ссылка:1
public ICommand AddAuto { get; }

ссылка:1
private bool CanAddAuto(object p) => flagBrandAuto && flagModelAuto && flagBodyType && flagRelease && flagColor && flagPrice && flagMonth && AutoImage != null;

ссылка:1
private void OnAddAuto(object p)
{
    if (Auto.AddAuto(textBoxBrandAuto, textBoxModelAuto, textBoxBodyType, textBoxRelease, textBoxColor, textBoxPrice, textBoxMonth, base64Image))
    {
        clearTextBox();
    }
}
#endregion
```

Листинг 5.1 — CanExecute команды

6 Руководство по использованию

Для того, чтобы пользоваться приложением, необходимо для начала зарегистрироваться в системе. Если у пользователя уже есть аккаунт, то нужно войти в систему. Окно регистрации и входа показано на рисунке 6.1.

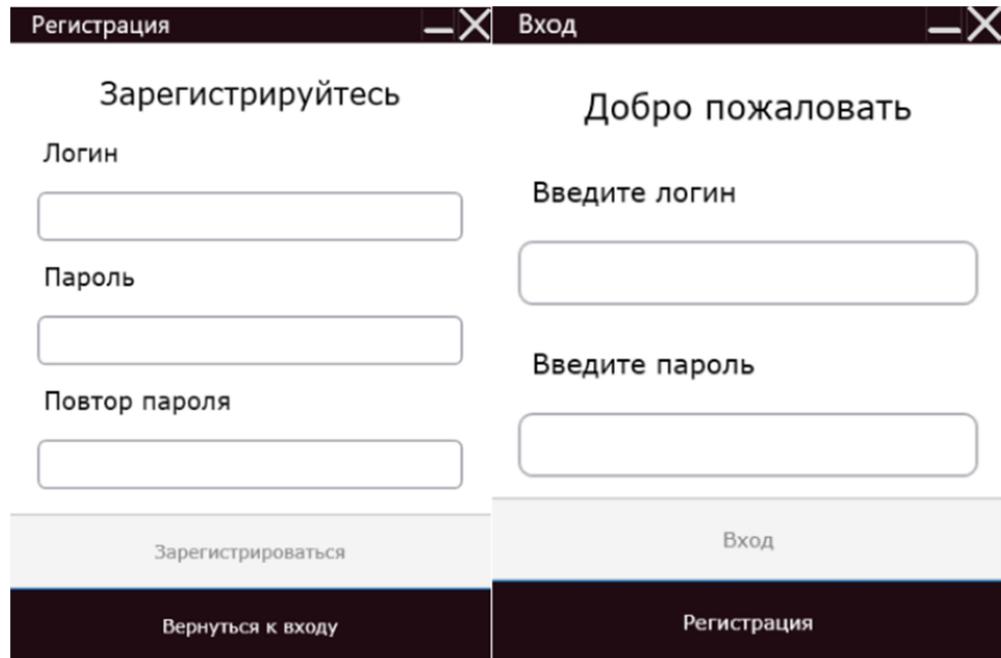


Рисунок 6.1 — Окно регистрации и входа

6.1 Руководство по использованию пользователем

Пользователь может добавлять понравившиеся автомобили в свой список. На каждом элементе списка имеется кнопка в виде сердце, при нажатие на которую авто добавляется в ваш список. Пример на рисунке 6.2.

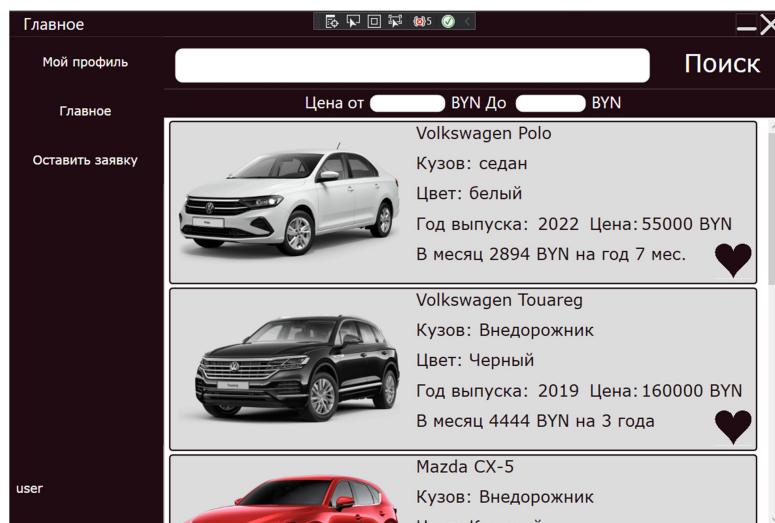


Рисунок 6.2 — Пример добавления в список понравившихся авто

Чтобы избавиться от этого авто в вашем списке, нужно зайти в ваш профиль. И как показано на (рисунок 6.3), удалить авто из списка,

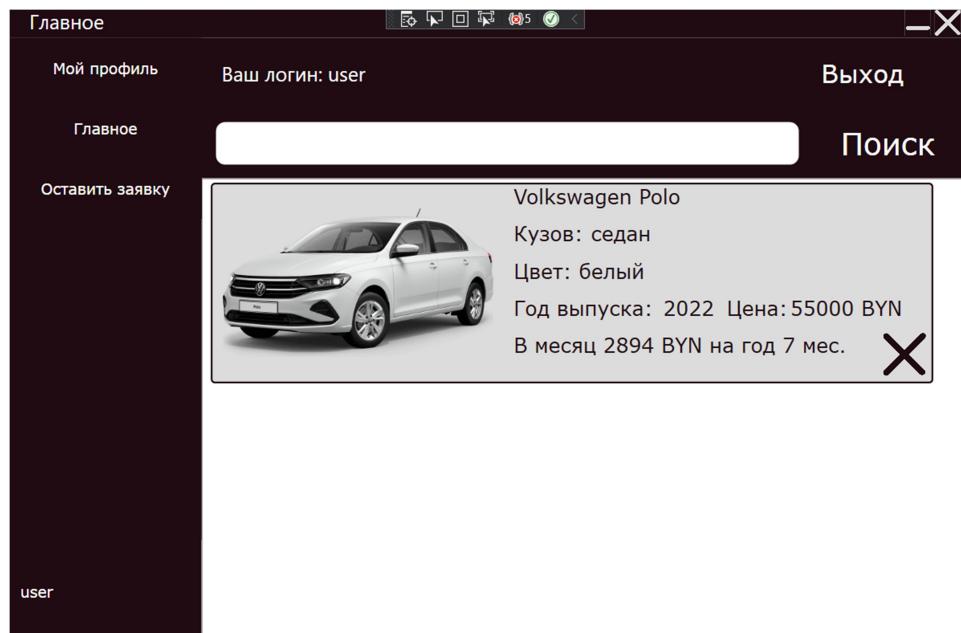


Рисунок 6.3 — Удаление авто из списка

Поиск и фильтрация работают как вместе, так и по независимо от друг друга (рисунок 6.4).

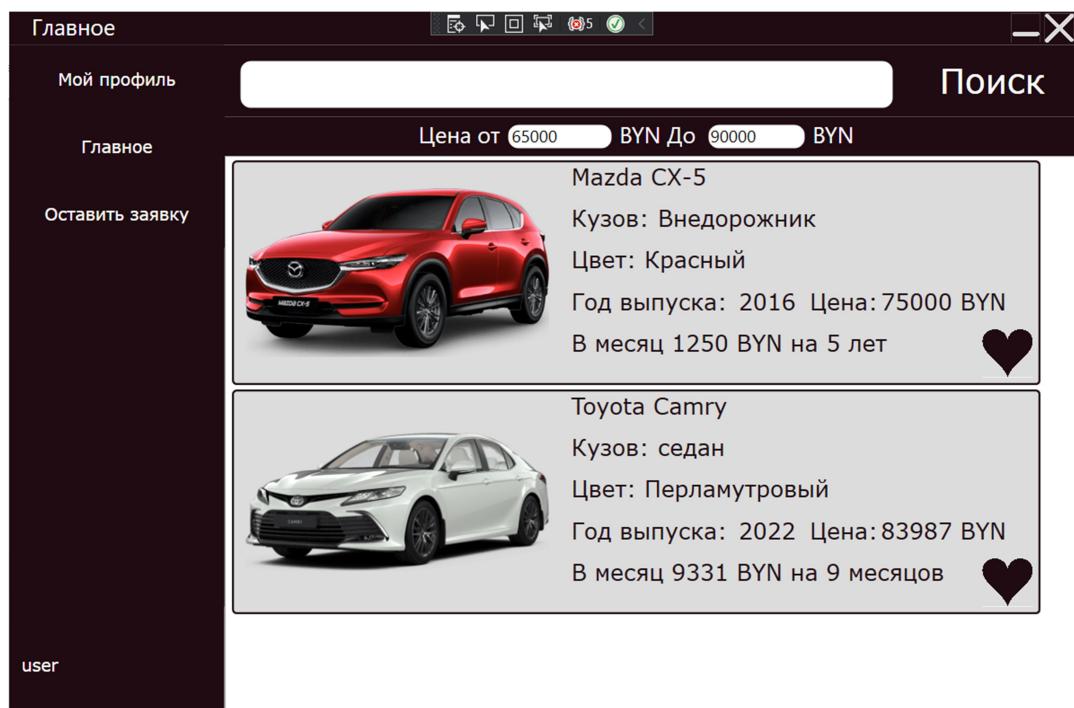


Рисунок 6.4 — Пример поиска и фильтрации

Пользователь может отправлять только одну заявку от своего лица (Рисунок 6.5.)

Главное

Мой профиль

Главное

Оставить заявку

Фамилия

Яхимчик

Имя

Даниил

Отчество

Юрьевич

Ваш номер телефона

295876951

user

Отправить заявку

Очистить поля

Рисунок 6.5 — пример заявки

Также все данные заявки проверяются на корректный ввод данных(Рисунок 6.6)

Главное

Мой профиль

Главное

Оставить заявку

Фамилия
Только символы

1213

Имя
Только символы

12

Отчество
Только символы

12

Ваш номер телефона
9 цифр вашего номера

1221

user1

Отправить заявку

Очистить поля

Рисунок 6.6 — Валидация заявки

6.2 Руководство по использованию администратором

Администратор может просматривать заявки клиентов, введённые ими данные, и дату заявки(Рисунок 6.7).

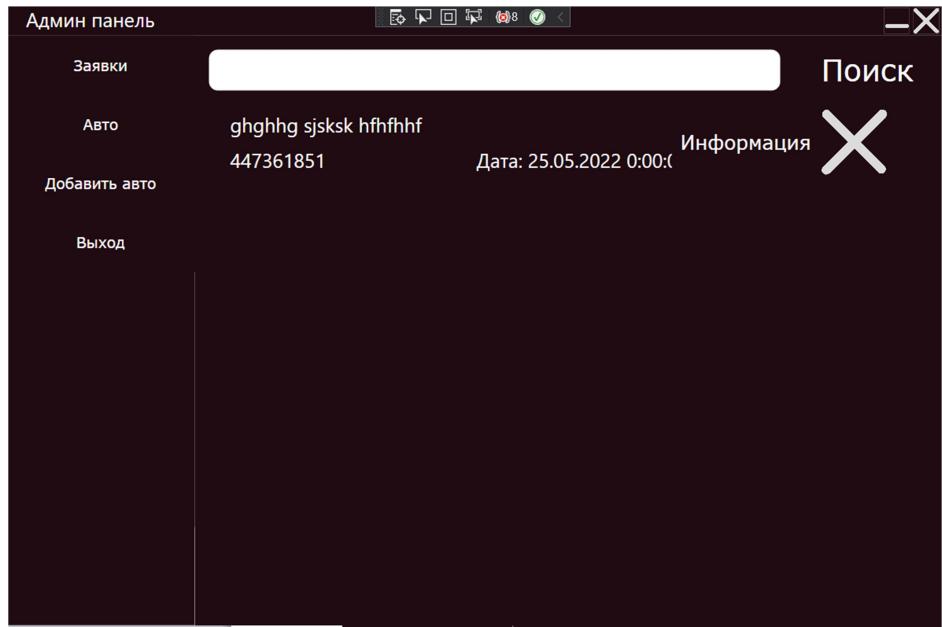


Рисунок 6.7 — Заявки клиентов

Мы можем удалить заявку, или просмотреть информацию о пользователе (рисунок 6.8).

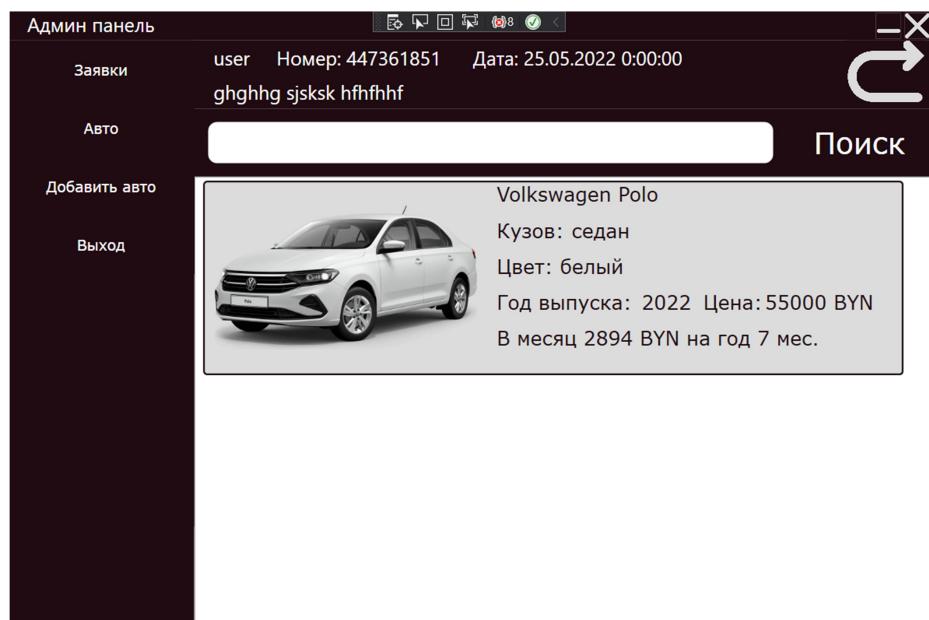


Рисунок 6.8 — Информация о пользователе

Мы можем добавить новый автомобиль. Администратор при добавлении нового авто может видеть как данные будут выглядеть у пользователя. Для добавле-

ния нового авто все поля должны быть указаны корректно в ином случае мы не сможем добавить новое авто(Рисунок 6.9).

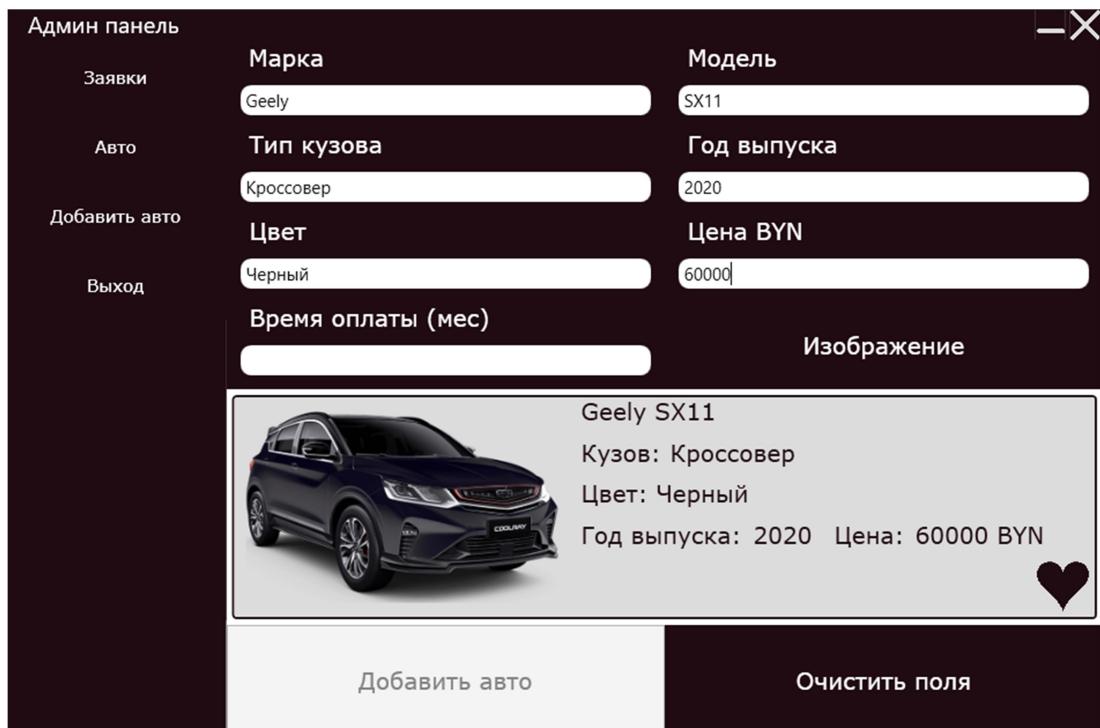


Рисунок 6.9 — Добавление авто

Список авто администратора отличается от пользовательского. У администратора вместо кнопки добавить понравившиеся авто, кнопка редактирования(рисунок 6.10).

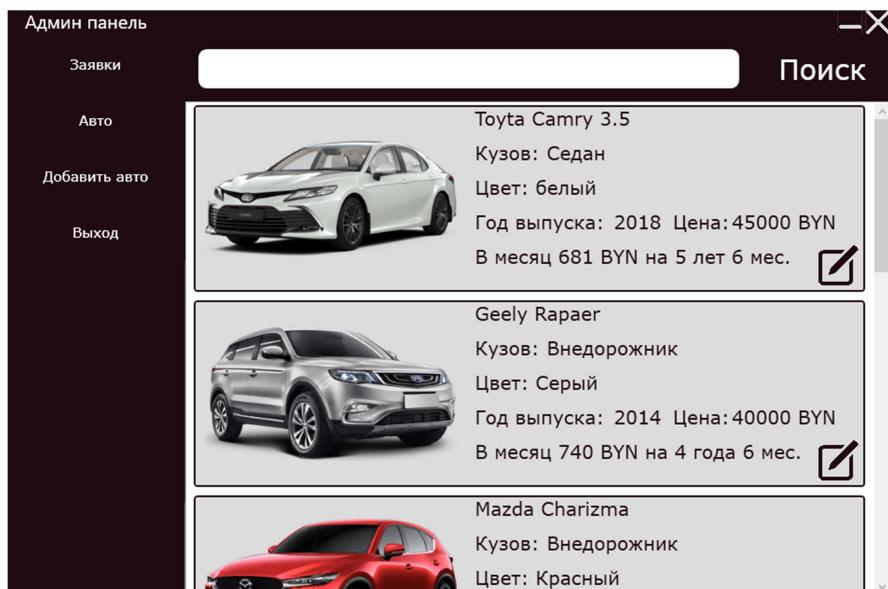


Рисунок 6.10 — Список авто администратора

Страница редактирования похоже на страницу добавления. С помощью данной страницы можно удалить или изменить данные об автомобиль(рисунок 6.11).

Админ панель

Заявки	Марка	Модель
	Toyota	Camry 3.5
Авто	Тип кузова	Год выпуска
	Седан	2018
Добавить авто	Цвет	Цена BYN
	белый	45000
Выход	Время оплаты (мес)	Изображение
	66	

Toyota Camry 3.5
Кузов: Седан
Цвет: белый
Год выпуска: 2018 Цена: 45000 BYN
В месяц 681 BYN на 5 лет 6 мес.



Сохранить  

Рисунок 6.11 — Страница редактирования

Заключение

В итоге выполнения данного курсового проекта было разработано программное средство «Лизинг авто». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения:

- архитектура приложения должна соответствовать шаблону проектирования, такому как MVVM;
- вся информация должна храниться в базе данных;
- все медиаресурсы должны хранится на сервере;
- приложение должно производить валидацию вводимых пользователем данных;
- приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
- приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
- приложение должно предоставлять пользователям возможность аутентификации, используя введенные данные;
- приложение должно предоставлять пользователем все основные функции библиотеки.

Также в процессе выполнения данного курсового проекта были закреплены навыки в программировании на языке C#, создании приложений на WPF, использование Entity Framework Core, работа с современным паттерном MVVM.

Список используемых источников

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> . Дата доступа: 18.05.2022
2. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 20.04.2022
3. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 29.04.2022
4. Форум для программистов или разработчиков [Электронный ресурс] – <https://stackoverflow.com/> – Дата доступа: 17.05.2022