Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “ Об’єктно - орієнтоване програмування”

“Автосалон”

Керівник , асистент каф. ПІ Ляпота В.М.

Студент гр. ПІ-15-2 Назаренко Д.С.

Комісія: проф.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

проф.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-2* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Назаренка Дмитра Сергійовича*

1. Тема проекту: «Автосалон»
2. Термін здачі студентом закінченого проекту: ***“28” - травня - 2016 р***.
3. Вихідні дані до проекту: База нових і старих вітчизняних та іноземних автомобілів: марка, рік випуску, технічні характеристики, особливості виконання, технічний стан, запитувана ціна. База покупців: контактні координати, вимоги до марки, технічним характеристикам і технічним станом, фінансові можливості. Автоматизація підбору варіантів для покупця, формування заявки.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Формулювання вимог до програми | 1-03-2016 – 6-03-2016 р. |
| 3 | Розробка підсистеми збору статистики | 7-03-2016 – 20-03-2016 р. |
| 4 | Розробка підсистеми пошуку за критеріями автомобілів . | 21-03-2016 – 30-03-2016 р. |
| 5 | Розробка функцій формування переліку заявок про куплю-продажу. | 1-04-2016 – 3-04-2016 р. |
| 6 | Розробка функцій зберігання та завантаження даних | 4-04-2016 – 9-04-2016 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 10-04-2016 – 29-04-2016 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 1-05-2016 – 31-05-2016 р. |
| 9 | Захист | 28-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Назаренко Д.С.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В. М.

« 1 »\_березня\_\_\_\_\_\_\_\_\_\_ 2016 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 42 с., 24 рис., 1 додаток, 5 джерел.

Метою роботи є розробка програми “Автосалон” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2015, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

Профіль на github: <https://github.com/DmytroNazarenko>

В результаті отримана программа під назвою “Автосалон”, яка дозволяє зберігати список автомобілів та користувачів системою, характеристики кожного автомобілю, дозволяє робити покупки та аналізувати авторинок. Є можливість додавати, видаляти та редагувати автомобілі. Також, є функція формування та перегляду замовлення для кожного користувача окремо.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, XML, АВТОМОБІЛІ, JSON.

СОДЕРЖАНИЕ

Введение 6

1 Спецификация программы 7

1.1 Список задач 7

1.2 Функции программы 7

1.3 Интерфейс пользователя 8

2 Проектная спецификация. 17

2.1 Архитектурный стиль 17

2.2. Объектная структура программы 17

2.3 Формат данных 19

3 Инструкция пользователя 21

ВЫВОДЫ 22

СПИСОК ЛИТЕРАТУРЫ.. 23

Приложение А – Код программы.. 24

ВВЕДЕНИЕ

В наше время процессы автоматизации охватывают все больше отраслей. Одной из таких отраслей является автомобильный рынок. Мы сталкиваемся с автомобилями регулярно в современном мире. Поэтому целью проекта было усовершенствовать и облегчить покупку-продажу автомобилей для обычных пользователей. Обычному покупателю, который ищет автомобиль

Исследовав существующие аналоги было принято решение разработать прикладную программу для вышеперечисленных целей.

Электронный автосалон вмещает в себя хранение, редактирование автомобилей, удобный интерфейс подбора автомобилей для пользователей, а также функцию сбора статистики и составление заявок на продажу автомобилей.

Благодаря этому, любой пользователь сможет быстро подобрать себе автомобиль и оформить его покупку, а продавец сможет без потери времени выставлять свое предложение.

1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ

1.1 Список задач

Целью курсового проекта является создание программы «Автосалон».

Данная программа должна содержать в себе детальную информацию про автомобили, а именно: марку, модель, пробег, стоимость, цвет. Также программа должна хранить данные про ее пользователей: как покупателей, так и продавцов. Она должна работать в двух режимах: для клиента и для продавца.

Программа должна иметь функцию удаления, добавления, изменения автомобилей для продавцов, а также функцию подбора по критериям для пользователей. Функционал также должен включать в себя возможность оформление заказа на покупку автомобиля и просмотра уже сделанных заказов. Сбор статистики для анализа также является одной из задач, поставленных перед нами.

Не менее важным является интерфейс программы. Он должен быть интуитивно понятным и простым в использовании. Одной из важнейших задач является правильная реакция программы на любые действия пользователя.

1.2 Функции программы

Разработанная программа должна иметь следующие функции:

1. регистрация новых пользователей;
2. авторизация уже существующих пользователей;
3. отображение автомобилей пользователя;
4. добавление нового автомобиля;
5. редактирование автомобиля;
6. удаление автомобиля;
7. подбор автомобиля по критериям;
8. оформление заказа;
9. просмотр информации про автомобиль и его владельца для пользователя;
10. анализ стоимости автомобиля на рынке;
11. печать оформленных заказов в отдельный файл.

1.3 Интерфейс пользователя

После запуска программы пользователь увидит форму авторизации (см.рис. 1.1).

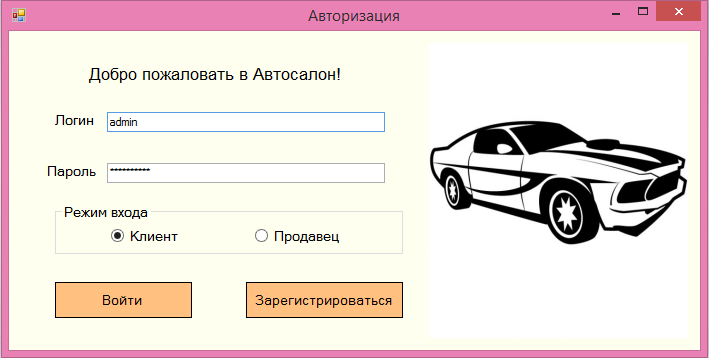


Рисунок 1.1 – Форма авторизации

На форме расположены 2 текстовых поля: для логина и пароля, кнопки для входа в аккаунт и регистрации, а также режим авторизации: как покупатель или как продавец. Пароль скрывается при наборе. В случае отсутствия логина и пароля в базе, на экране появится сообщение об ошибке (см.рис. 1.2).

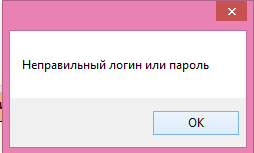


Рисунок 1.2 – Сообщение об ошибке

Если пользователя еще нет в базе, ему предоставляется возможность зарегистрироваться нажав на соответствующую кнопку. В этом случае перед ним откроется форма для регистрации (см.рис 1.3).

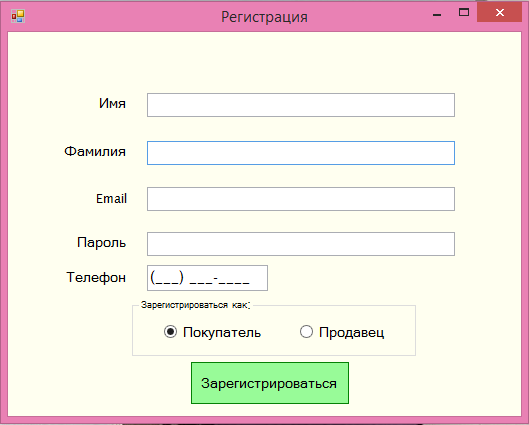


Рисунок 1.3 – Форма регистрации

Пользователь должен ввести свои данные и придумать пароль. Ему предоставляется возможность выбрать режим регистрации: для продавца или покупателя. В случае если такой логин уже существует, то выведется ошибка (см.рис. 1.5 слева). Если пользователь произвел некорректный ввод данных, то появится другая ошибка (см.рис. 1.5 справа).

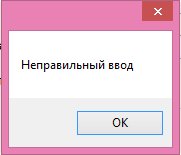
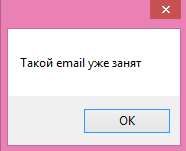
 

Рисунок 1.5 – Сообщения об ошибках в регистрации

После регистрации ползователю снова нужно будет авторизироваться. Дальнейшая форма появится в зависимости от выбора режима пользователем. Форма для продавца имеет 3 кнопки: для редактирования, удаления и добавления автомобиля.

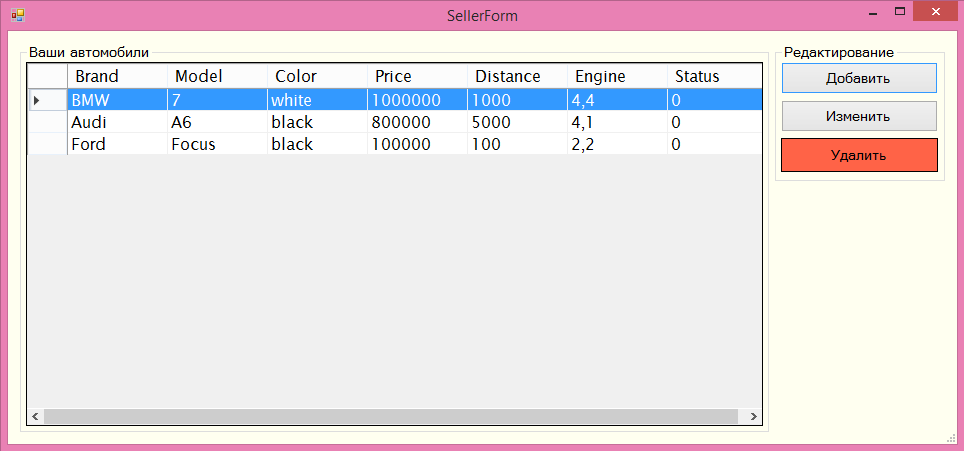


Рисунок 1.7 – Форма регистрации

Большую часть формы занимает элемент для вывода данных на экран: DataGridView. По нажатию на ячейку, мы выбираем всю строку, в которой она расположена. Автомобили представлены в виде строк, а их характеристики – столбцы. При входе в форму, элемент сразу же заполняется данными для данного пользователя.

Для того чтобы редактировать или добавить автомобиль, нужно щелкнуть по соответствующей кнопку. В случае выбора удаления, будет удален автомобиль, который был выбран в DataGridView. Если же пользователь захочет изменить или добавить автомобиль, перед ним появится форма для редактирования автомобиля. Форма для добавления (см.рис. 1.8) отличается от формы для изменения (см.рис. 1.7) редактированием всех полей.

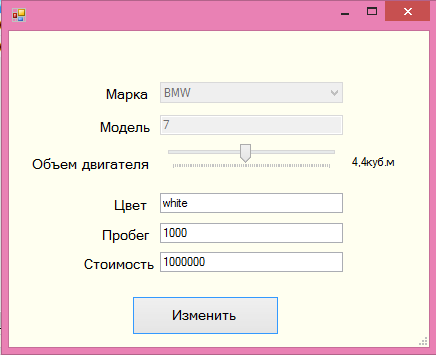


Рисунок 1.7 – Форма для изменения

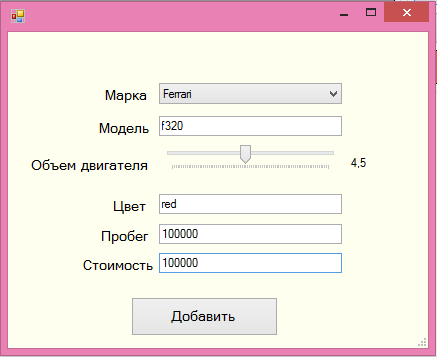


Рисунок 1.8 – Форма для добавления

В случае ошибки, поле в котором была допущена ошибка подсветится красным (см.рис. 1.9).

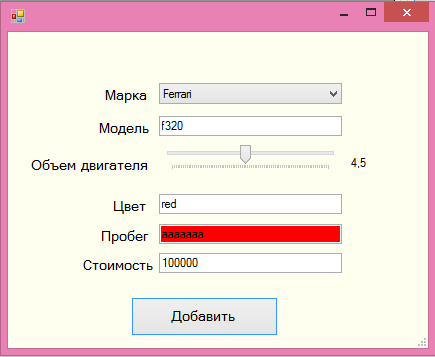


Рисунок 1.9 – Форма с ошибкой

Для покупателя предусмотрен другой интерфейс, представляющий, прежде всего, подбор автомобилей по заданным критериям (см.рис. 1.10).

Сама форма разделена на 3 группы : селекторы подбора по параметрам, отображение результатов, кнопки дополнительных действий.

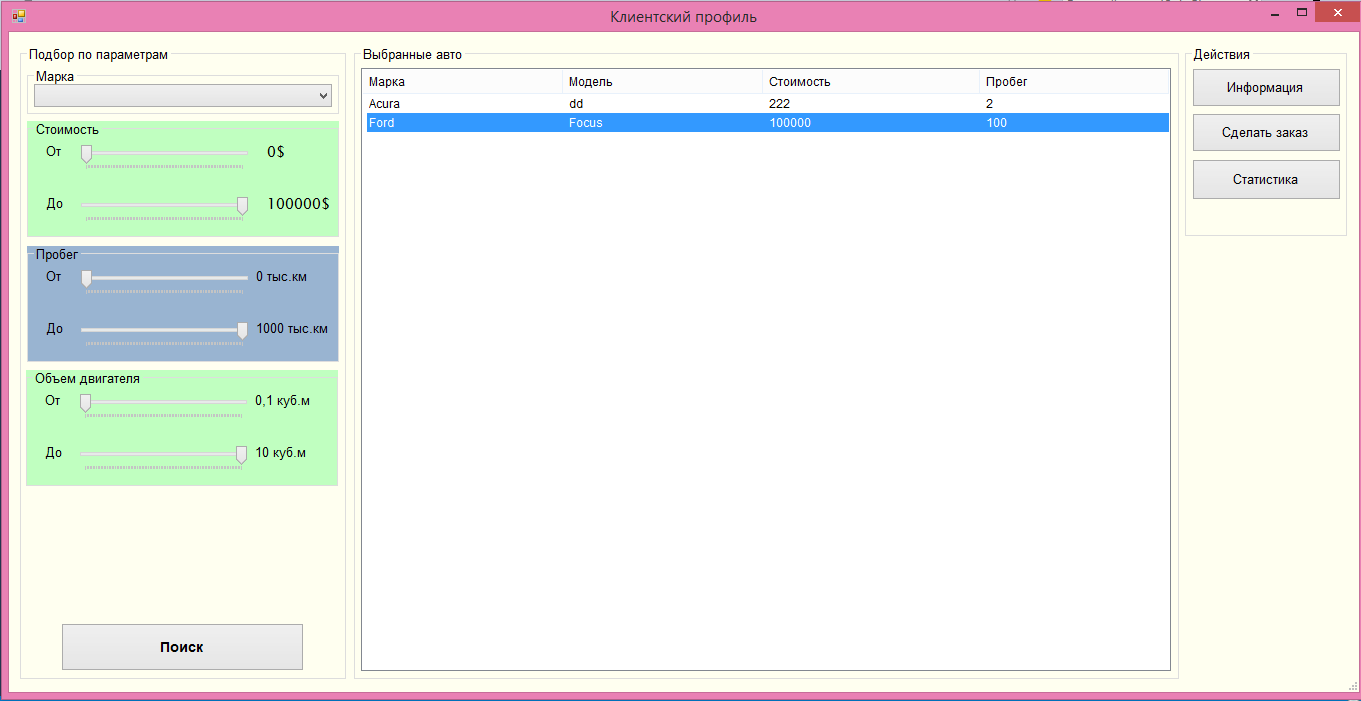


Рисунок 1.10 – Клиентская форма

В центре формы расположен элемент ListView для отоборажения выбранных автомобилей по критериям. В нем отображены модель, марка, стоимость и пробег автомобиля. Чтобы выбрать автомобиль, достаточно кликнуть по строке, в которой он находится.

Слева находятся селекторы подбора (см.рис. 1.11). Пользователь может подобрать авто по марке, по стоимости, по пробегу и по объему двигателя. Все элементы подбора, кроме выбора марки, представлены сдвоенными элементами TrackBar. Для того чтобы осуществить поиск нужно нажать на кнопку внизу.

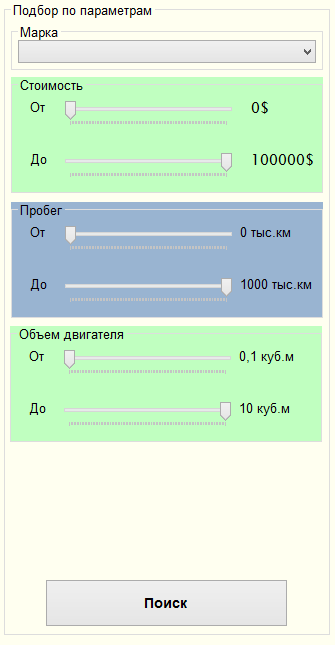


Рисунок 1.11 – Селекторы подбора по критериям

Справа расположены клавиши дополнительных функций (см.рис. 1.12). Кнопка «информация» показывает информацию про автомобиль и его владельца, чтобы пользователь мог связаться с ним (см.рис. 1.13). Кнопка «сделать заказ» формирует покупку автомобиля, который выбран в ListView, и сохраняет его в файле (см.рис. 1.14).

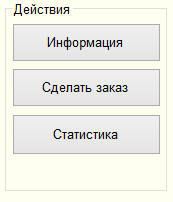


Рисунок 1.12 – Кнопки дополнительных функций

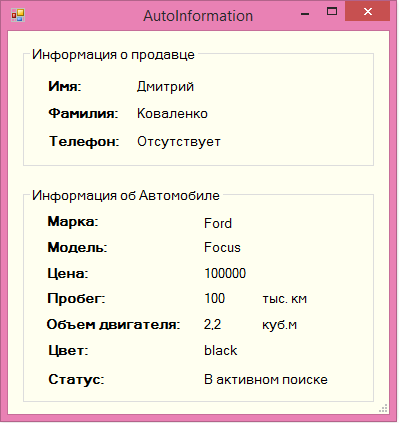


Рисунок 1.13 – Окно дополнительной информации

При нажатии на кнопку «статистика» появится форма статистики, в которой находится график изменения средней цены данного автомобиля по месяцам (см.рис. 1.15).

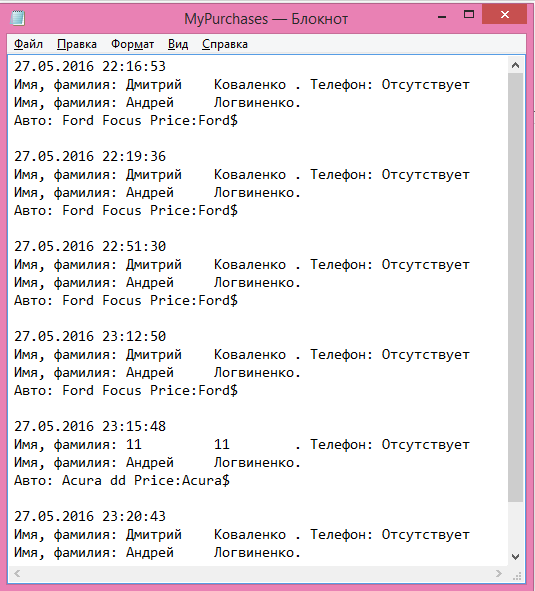


Рисунок 1.14 – Текстовый файл с заявками пользователя

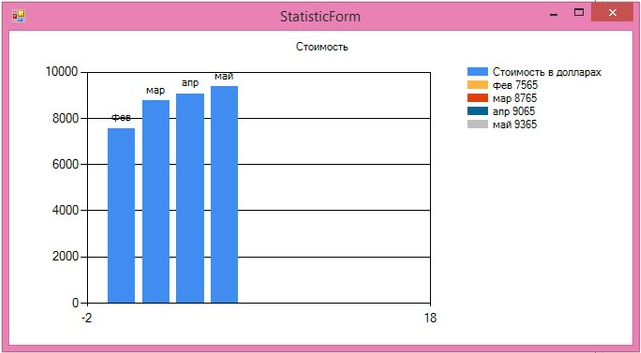


Рисунок 1.15 – Форма со статистикой

2 ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ

 2.1 Архитектурный стиль

Для реализации данного проекта лучше всего подходит архитектура MVC. В ней вся логика делится на три отдельных компонента: Model, View, Controller.

Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать

В компоненте View (изображение) происходит отображение данных модели.

Controller (контроллер) реагирует на внешние события и модель. В нашей программе мы использовали «толстый» контроллер, тоесть основная логика приложения находится в контроллере.

В нашем приложении мы использовали реляционную базу данных MSSQL. Модель получала данные из базы, а контроллер уже обрабатывал эти данные, и передавал их View, которое отображало их в пользовательском интерфейсе.

2.2. Объектная структура программы

Программа представлена набором классов, взаимодействующих между собой (рисунок 2.1).

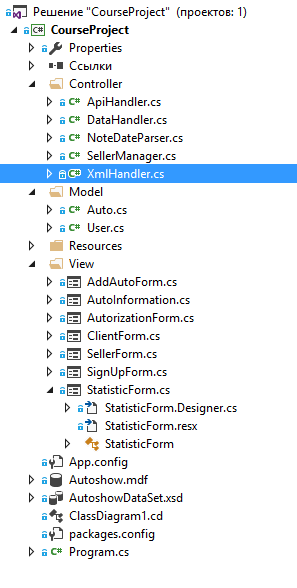


Рисунок 2.1 – Архитектура проекта

В программе реализованы такие классы: ApiHandler, XmlHandler, SellerManager, NoteDateParser, DataHandler, Auto, User. Диаграмма классов представлена на рисунке 2.2. Классы Auto и User входят в модель и служат для представления данных в коде. Классы ApiHandler, XmlHandler, SellerManager, NoteDateParser, DataHandler содержат в себе всю логику приложения.

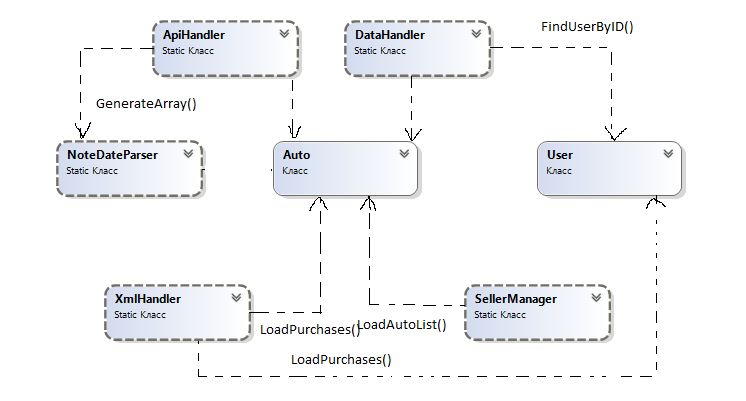


Рисунок 2.2 – Диаграмма классов проекта

2.3. Формат данных

Для хранения информации используется база данных. Она состоит из 2 таблиц: Subject, где хранятся пользователи, и Auto, где хранятся автомобили.

Также, данные про заявки на продажу мы храним в формате xml, как показано на рисунке 2.4. Затем при помощи linq to xml и чтения потока, мы считываем данные.

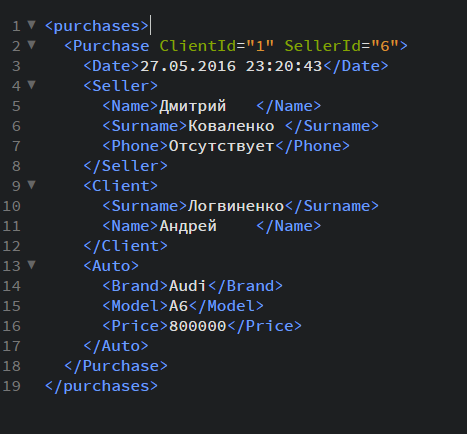


Рисунок 2.4 – Данные в формате xml

3. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Установка программы:

а) Запускаем «Мой компьютер».

б) Заходим на диск с программным продуктом.

в) Распаковываем архив в желаемую папку на компьютере.

г) Можно запускать ".exe" файл и начинать работу с программой.

При входе попадаем на форму авторизации, если вы уже были зарегистрированы, впишите свой логин и пароль в соответствующие поля (см.рис. 3.1). В противном случае, пройдите по клику на кнопку «зарегистрироваться» на форму регистрации.

В случае успешной регистрации вы попадете опять на форму авторизации, где введете необходимые данные.

Выберите режим входа для дальнейшего пользования, по умолчанию стоит «клиент».

Если вы зашли в режиме продавца (рисунок 3.2), то чтобы добавить/ редактировать/ удалить автомобиль, щелкните на соответствующую кнопку сбоку. В случае удаления, произойдет удаление выбраного авто, иначе высветится форма редактирования (см.рис. 3.3), где введите данные, которые хотите изменить. После этого нажмите кнопку добавить / изменить.

Если вы зашли в режиме покупателя, то чтобы оуществить поиск по критериям, передвиньте соответствующие Trackbarы для изменения параметром поиска (см.рис. 3.4) или выберете марку из выпадающего списка и щелкните кнопку Поиск.

Результат появится в центре формы, в виде списка машин. Вы можете выбрать их, щелкнув по ним кнопкой мыши. Чтобы просмотреть информацию, сделать заказ или просмотреть статистику, щелкните по соответствующей клавише сбоку (см.рис. 3.5). Список заявок хранится в файле “MyPurchases.txt” (см.рис. 3.6).

ВЫВОДЫ

Во время выполнения курсового проекта и сопровождение программного продукта мной были пройдены все пути разработки данной программы, начиная с постановки задачи до тестирования и отладки программы.

В ходе разработки проекта была создана приложение «Автосалон» на платформе .Net с использованием языка C #.

В результате получена программа под названием "Автосалон", которая позволяет сохранять список автомобилей и пользователей системы, характеристики каждому автомобилю, позволяет делать покупки и анализировать авторынок. Есть возможность добавлять, удалять и редактировать автомобили. Также, есть функция формирования и просмотра заказ для каждого пользователя.

 Мною были подробно рассмотрены такие аспекты .Net: основные принципы объектно-ориентированного программирования, SQL, LINQ запросы, XML, JSON, делегаты, события, итераторы и индексаторы. Был получен опыт в проектировании, реализации, документировании полноценной программы.

В будущем планируется выполнение некоторых пунктов для улучшения программы, а именно: переработка дизайна и усовершенствование статистики.

СПИСОК ЛИТЕРАТУРЫ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е издание= C# Programming Language (Covering C# 4.0), 4th Ed. – СПб.: «Питер», 2012. – 784 с.
2. Бондарев В. М. «Объектно-ориентированное программирование на С#» [Текст]: учеб. пособ / В. М. Бондарев. – Х.: Смит, 2009. – 224 с.;
3. Дейт, К. Дж. Введення у системи баз даних [Текст] / К. Дж. Дейт.- 7-е вид. – М. : Вид. дім «Вільямс», 2001. – 846 с.
4. Троелсен: «Мова програмування С# 2005 та платформа .NET 2.0» [Текст]: учеб. пособ, 3е видання. : Пер. з англ. – Москва. : ООО «И.Д. Вільямс», 2007. – 1168с.
5. Ульман, Дж. Д. Основи систем баз даних [Текст] / Дж. Д. Ульман - М. : Фінанси і статистика, 1983. – 334 с.

Додаток А – Код програми

namespace CourseProject.Model

{

class User

//Класс пользователя

{

public int ID { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Phone { get; set; }

public bool Client { get; set; }

public User(string name, string surname, string phone, bool client, int id = 0)

{

ID = id;

Name = name;

Surname = surname;

Phone = phone;

Client = client;

}

public User() : this(null, null, null, false) {}

}

}

namespace CourseProject.Model

{

public class Auto

//Класс автомобиль

{

public string Brand { get; set; }

public string Model { get; set; }

public string Color { get; set; }

public int Price { get; set; }

public int Distance { get; set; }

public double Engine { get; set; }

public byte Status { get; set; }

public int ID { get; set; }

public int SellerID { get; set; }

public Auto(string brand, string model, string color, int price, int distance, double engine, byte status, int sellerid, int id = 0)

{

ID = id;

Brand = brand;

Model = model;

Color = color;

Price = price;

Distance = distance;

Engine = engine;

Status = status;

SellerID = sellerid;

}

public Auto():this(null,null,null,0,0,0,0,0) { }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CourseProject.Model;

using System.Data.SqlClient;

using System.Data;

using System.Windows.Forms;

namespace CourseProject.Controller

{

static class DataHandler

//Класс для работы с данными

{

static SqlConnection cn = new SqlConnection(@"Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename='C:\Users\Дмитрий\documents\visual studio 2015\Projects\CourseProject\CourseProject\Autoshow.mdf';Integrated Security = True; Connect Timeout = 30");

static SqlCommand cmd = new SqlCommand("",cn);

public static List<Auto> SelectAutosByParameter(int price1, int price2, string brand, int dist1, int dist2, double engine1, double engine2)

//Метод подбора авто по критериям

{

List<Auto> autoList = new List<Auto>();

string price = (price1 == 0 && price2 == 0)? "" :" and Price >=" + price1 + " and Price <= " + price2;

string dist = (dist1 == 0 && dist2 == 0) ? "" : " and Distance >=" + dist1 + " and Distance <= " + dist2;

string engine = (engine1 == 0.1 && engine2 == 0.1) ? "" : " and EngineCapacity >=" + engine1.ToString(System.Globalization.CultureInfo.InvariantCulture) + " and EngineCapacity <= " + engine2.ToString(System.Globalization.CultureInfo.InvariantCulture);

string sql\_query;

if (brand == null || brand == "")

{

sql\_query = "select \* from Auto where Brand is not null" + price + dist + engine;

}

else

{

sql\_query = "select \* from Auto where Brand = '" + brand +"'"+price + dist + engine;

}

cn.Open();

SqlDataAdapter SDA = new SqlDataAdapter(sql\_query , cn);

DataTable dt = new DataTable();

SDA.Fill(dt);

cn.Close();

autoList = dt.AsEnumerable().Select(r => new Auto

{

Brand = r.Field<string>("Brand"),

Model = r.Field<string>("Model"),

Price = (r.IsNull("Price")) ? 0 : r.Field<int>("Price"),

Color = r.Field<string>("Color"),

Engine = (r.IsNull("EngineCapacity")) ? 0.0 : (double)r.Field<Decimal>("EngineCapacity"),

Distance = (r.IsNull("Distance")) ? 0 : r.Field<int>("Distance"),

SellerID = r.Field<int>("SubjectID"),

//Status = r.Field<byte>("Status"),

}).ToList();

return autoList;

}

public static User GetUserById(int id)

//Метод поиска пользователя по ID

{

string sql\_query = "select \* from Subject where id = " + id;

cn.Open();

var SDA = new SqlDataAdapter(sql\_query, cn);

var dt = new DataTable();

SDA.Fill(dt);

cn.Close();

return dt.AsEnumerable().Select(x => new User(x.Field<string>("Name"), x.Field<string>("Surname"), x.Field<string>("Phone"), false, x.Field<int>("Id"))).Single();

}

public static bool UserRegistration(string name, string surname, string email, string password, string phone, bool client)

//Регистрация пользователя в БД

{

bool b = true;

if (password != "" && email != "" && phone != ""

&& name != "" && surname != "")

{

cn.Open();

cmd.CommandText = "select Count(\*) from Subject

where email = '" + email +"'";

if (cmd.ExecuteScalar().Equals(0))

{

try

{

cn.Open();

cmd.CommandText = "insert into Subject (Name, Surname, Email, Password, Client, Phone) VALUES " +

"('" + name + "','" + surname + "','" +

email + "','" + password + "'," +

Convert.ToByte(client) + ",'" + phone + "')";

cmd.Connection = cn;

cmd.ExecuteNonQuery();

}

catch

{

b = false;

}

finally

{

cn.Close();

}

}

else

{

MessageBox.Show("Такой email уже занят");

return true;

}

}

else b = false;

return b;

}

public static int IsUserExist(string login, string password, bool client)

//Проверка на существование логина и пароля

{

int id = 0;

//try

//{

cmd.CommandText = "select ID from Subject where Email = '" + login + "' and Password = '" + password + "' and Client = " + Convert.ToByte(client);

cn.Open();

try

{

id = (int)cmd.ExecuteScalar();

}

//}

catch {

}

finally

{

cn.Close();

}

return id;

}

}

}

using CourseProject.Model;

using System;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

namespace CourseProject.Controller

{

static class SellerManager

//Класс для работы с продавцом

{

static SqlConnection cn = new SqlConnection(@"Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename='C:\Users\Дмитрий\documents\visual studio 2015\Projects\CourseProject\CourseProject\Autoshow.mdf';Integrated Security = True; Connect Timeout = 30");

static SqlCommand cmd = new SqlCommand("", cn);

public static void DeleteAuto(int autoId)

//Удаление авто из бд

{

cn.Open();

cmd.CommandText = "delete from Auto where id = " + autoId;

cmd.ExecuteNonQuery();

cn.Close();

}

public static BindingList<Auto> LoadAutoList(int userid)

//Формирование списка автомобилей продавца

{

cn.Open();

string sqlQuery= "select \* from Auto where SubjectID =" + userid;

SqlDataAdapter SDA = new SqlDataAdapter(sqlQuery, cn);

DataTable dt = new DataTable();

SDA.Fill(dt);

cn.Close();

BindingList<Auto> res = new BindingList<Auto>(dt.AsEnumerable().Select(r => new Auto

{

ID = r.Field<int>("ID"),

Brand = r.Field<string>("Brand"),

Model = r.Field<string>("Model"),

Price = (r.IsNull("Price")) ? 0 : r.Field<int>("Price"),

Color = r.Field<string>("Color"),

Distance = (r.IsNull("Distance")) ? 0 : r.Field<int>("Distance"),

Engine = (r.IsNull("EngineCapacity")) ? 0.0 : (double)r.Field<Decimal>("EngineCapacity"),

SellerID = r.Field<int>("SubjectID"),

//Status = r.Field<byte>("Status"),

}).ToList());

return res;

}

public static void UpdateAuto(Auto auto)

//Изменение автомобиля в бд

{

cn.Open();

cmd.CommandText = "update Auto set Price = '"+auto.Price+"', Color = '"+auto.Color+"', Distance = "+auto.Distance+", EngineCapacity = "+auto.Engine.ToString(System.Globalization.CultureInfo.InvariantCulture) + " where Id ="+auto.ID;

cmd.ExecuteNonQuery();

cn.Close();

}

public static void AddAuto(Auto auto)

//Добавление авто в бд

{

cn.Open();

cmd.CommandText = "insert into Auto(Brand,model,price,color,distance,enginecapacity,SubjectId) Values ('" + auto.Brand + "','" + auto.Model + "', " + auto.Price + ", '" + auto.Color + "', " + auto.Distance + ", " + auto.Engine.ToString(System.Globalization.CultureInfo.InvariantCulture) + " ," + auto.SellerID + ")";

cmd.ExecuteNonQuery();

cn.Close();

}

}

}

using CourseProject.Model;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

namespace CourseProject.Controller

{

static class XmlHandler

//Класс для работы с xml

{

public static void Create()

//Создание xml файла

{

using (var writer = new XmlTextWriter("Purchases.xml", Encoding.UTF8))

{

writer.Formatting = Formatting.Indented;

writer.WriteStartElement("Purchases");

writer.WriteEndElement();

}

}

public static void Save(Auto \_auto, User \_client, User \_seller)

//Запись в xml файл

{

XmlDocument doc = new XmlDocument();

doc.Load("Purchases.xml");

var purchase = doc.CreateElement("Purchase");

purchase.SetAttribute("ClientId", \_client.ID.ToString());

purchase.SetAttribute("SellerId", \_seller.ID.ToString());

var date = doc.CreateElement("Date");

date.InnerText = DateTime.Now.ToString();

purchase.AppendChild(date);

var seller = doc.CreateElement("Seller");

var name = doc.CreateElement("Name");

name.InnerText = \_seller.Name;

var surname = doc.CreateElement("Surname");

surname.InnerText = \_seller.Surname;

var phone = doc.CreateElement("Phone");

phone.InnerText = \_seller.Phone;

seller.AppendChild(name);

seller.AppendChild(surname);

seller.AppendChild(phone);

purchase.AppendChild(seller);

var clientName = doc.CreateElement("Name");

clientName.InnerText = \_client.Name;

var clientSurname = doc.CreateElement("Surname");

clientSurname.InnerText = \_client.Surname;

var client = doc.CreateElement("Client");

client.AppendChild(clientSurname);

client.AppendChild(clientName);

purchase.AppendChild(client);

var auto = doc.CreateElement("Auto");

var autoBrand = doc.CreateElement("Brand");

autoBrand.InnerText = \_auto.Brand;

var autoModel = doc.CreateElement("Model");

autoModel.InnerText = \_auto.Model;

var autoPrice = doc.CreateElement("Price");

autoPrice.InnerText = \_auto.Price.ToString();

auto.AppendChild(autoBrand);

auto.AppendChild(autoModel);

auto.AppendChild(autoPrice);

purchase.AppendChild(auto);

doc.DocumentElement.AppendChild(purchase);

doc.Save("Purchases.xml");

}

static IEnumerable<XElement> LoadUserPurchasesInXML(int userID)

//Загрузка данных из файла в виде xml

{

XElement root = XElement.Load("Purchases.xml");

var x = from el in root.Elements("Purchase")

where (string)el.Attribute("SellerId") == userID.ToString() || (string)el.Attribute("ClientId") == userID.ToString()

select el;

return x;

}

public static void LoadUserPurchases(int userID)

{

//Обновление файла заявок

IEnumerable<XElement> xml = (IEnumerable<XElement>)LoadUserPurchasesInXML(userID);

using (var sw = new StreamWriter("MyPurchases.txt", false))

{

foreach (XElement x in xml)

{

sw.WriteLine(x.Element("Date").Value);

sw.WriteLine("Имя, фамилия: "+x.Element("Seller").Element("Name").Value + " " +

x.Element("Seller").Element("Surname").Value + ". " +

"Телефон: " + x.Element("Seller").Element("Phone").Value);

sw.WriteLine("Имя, фамилия: " + x.Element("Client").Element("Name").Value + " " +

x.Element("Client").Element("Surname").Value + ". ");

sw.WriteLine("Авто: " + x.Element("Auto").Element("Brand").Value + " "

+ x.Element("Auto").Element("Model").Value + " Price:" + x.Element("Auto").Element("Brand").Value+"$");

sw.WriteLine();

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using HtmlAgilityPack;

using System.Text.RegularExpressions;

namespace CourseProject.Controller

{

static class NoteDateParser

//Класс для извлечения даты с сайта

{

static DateTime parseForDate(int noteId)

//Парсинг сайта на дату заметки

{

HtmlWeb htmlWeb = new HtmlWeb();

HtmlDocument doc = htmlWeb.Load("https://auto.ria.com/auto\_\_\_"+noteId+".html");

HtmlNodeCollection elements = doc.DocumentNode.SelectNodes("//strong[@id='final\_page\_\_add\_date']");

string result = null;

if (elements != null) result = elements[0].InnerText.Split(' ')[0];

DateTime d = DateTime.Parse(result);

return d;

}

public static int MonthCount(int noteID)

//Расчет времени

{

DateTime cur = DateTime.Now;

DateTime res = parseForDate(noteID);

return (cur.Year - res.Year)\*12 + cur.Month - res.Month;

}

}

}

using System.Collections.Generic;

using System.Linq;

using Newtonsoft.Json;

using System.Net;

using System.IO;

using CourseProject.Model;

namespace CourseProject.Controller

{

/\*--------Классы-для-десериализации-----\*/

class JsonData

{

public int total { get; set; }

public double arithmeticMean { get; set; }

public double interQuartileMean { get; set; }

public Percentile percentiles { get; set; }

public double[] prices { get; set; }

public int[] classifieds { get; set; }

}

class Percentile

{

public Dictionary<string, double> a;

}

class Item

{

public string name;

public int value;

}

/\*--------------------------------------\*/

class ApiHandler

// Класс для работы с API сайта

{

public static double[] GenerateArray(Auto auto)

//Генерация массива для статистики из исходных данных

{

string url = ConvertToUrl(auto);

return ArrayCreating(GET(url));

}

static string ConvertToUrl(Auto auto)

//Создаем URL GET-запроса

{

int brandValue = 0, modelValue = 0;

try {

var brandJSON = GET("http://api.auto.ria.com/categories/1/marks");

Item[] data = JsonConvert.DeserializeObject<Item[]>(brandJSON);

foreach (Item item in data)

if (item.name == auto.Brand)

{

brandValue = item.value;

break;

}

//data.Single(x => x.name == auto.Brand);

}

catch { }

try {

var modelJson = GET("http://api.auto.ria.com/categories/1/marks/" + brandValue+"/models");

Item[] data1 = JsonConvert.DeserializeObject<Item[]>(modelJson);

foreach (Item item in data1)

if (item.name == auto.Model)

{

modelValue = item.value;

break;

}

//data1.Single(x => x.name == auto.Brand);

}

catch { }

int r1 = (int)(0.9\*auto.Distance);

int r2 = (int)(1.1\*auto.Distance);

string resultUrl = "http://api.auto.ria.com/average?marka\_id="+brandValue+"&model\_id="+modelValue+ "&raceInt="+r1+ "&raceInt="+r2;

return resultUrl;

}

static double[] ArrayCreating(string json)

//Создание массива статистики из десериализованного JSON

{

JsonData data = JsonConvert.DeserializeObject<JsonData>(json);

List<double>[] list = new List<double>[13];

for (int i = 0; i < 13; i++) {

list[i] = new List<double>();

}

double[] resultArray = new double[13];

//var v = data.classifieds.Where(x => NoteDateParser.MonthCount(x) <= 12);

for (int i = 0; i < data.classifieds.Length; i++ )

{

int delta = NoteDateParser.MonthCount(data.classifieds[i]);

if (delta <= 12) list[delta].Add(data.prices[i]);

}

for (int i = 0; i < list.Length; i++)

if (list[i].Count != 0) resultArray[i] = list[i].Average();

return resultArray;

}

static string GET(string Url, string Data = null)

//Получение GET запроса в виде JSON

{

WebRequest req = (Data == null) ? WebRequest.Create(Url) : WebRequest.Create(Url + "?" + Data);

WebResponse resp = req.GetResponse();

Stream stream = resp.GetResponseStream();

string Out = "";

if (stream != null)

using (StreamReader sr = new StreamReader(stream))

{

Out = sr.ReadToEnd();

}

return Out;

}

}

}

using CourseProject.Controller;

using CourseProject.Model;

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 CourseProject.View

{

public partial class AddAutoForm : Form

//Форма добавления авто

{

Auto resAuto;

int SellerID;

bool Updating = false;

public AddAutoForm(int id, Auto auto = null)

{

InitializeComponent();

SellerID = id;

brandBox.DropDownStyle = ComboBoxStyle.DropDownList;

if (auto != null)

{

Updating = true;

colorBox.Text = auto.Color;

distanceBox.Text = auto.Distance.ToString();

priceBox.Text = auto.Price.ToString();

brandBox.Enabled = false;

modelBox.Text = auto.Model;

modelBox.Enabled = false;

brandBox.SelectedItem = auto.Brand;

engineCapacityTrackbar.Value = (int)(auto.Engine\*10);

engineValue.Text = auto.Engine.ToString() + "куб.м";

addButton.Text = "Изменить";

}

resAuto = auto;

}

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

{

if (!Updating && adding())

{

SellerManager.AddAuto(resAuto);

this.Close();

}

if (Updating)

{

if (updating())

{

SellerManager.UpdateAuto(resAuto);

this.Close();

}

}

}

private bool adding()

{

bool b = true;

string brand = null, model = null, color = null;

int price = 0, distance = 0;

double engine = 0;

if (brandBox.SelectedItem != null) brand = brandBox.SelectedItem.ToString();

else { brandBox.BackColor = Color.Red; b = false;}

if (modelBox.Text != "") model = modelBox.Text;

else { modelBox.BackColor = Color.Red; b = false; }

if (colorBox.Text != "") color = colorBox.Text;

else { colorBox.BackColor = Color.Red; b = false; }

try

{

price = Convert.ToInt32(priceBox.Text);

if (price < 0 || price > 5000000) throw new Exception();

}

catch { priceBox.BackColor = Color.Red; b = false; }

try

{

distance = Convert.ToInt32(distanceBox.Text);

if (distance < 0 || distance > 5000000) throw new Exception();

}

catch { distanceBox.BackColor = Color.Red; b = false; }

engine = (engineCapacityTrackbar.Value/10.0);

resAuto = new Auto(brand, model, color, price, distance, engine,0,SellerID);

return b;

}

private bool updating()

{

bool b = true;

if (colorBox.Text != "") resAuto.Color = colorBox.Text;

else { colorBox.BackColor = Color.Red; b = false; }

try

{

int price = Convert.ToInt32(priceBox.Text);

if (price < 0 || price > 5000000) throw new Exception();

resAuto.Price = price;

}

catch { priceBox.BackColor = Color.Red; b = false; }

try

{

int distance = Convert.ToInt32(distanceBox.Text);

if (distance < 0 || distance > 5000000) throw new Exception();

resAuto.Distance = distance;

}

catch { distanceBox.BackColor = Color.Red; b = false; }

resAuto.Engine = (engineCapacityTrackbar.Value / 10.0);

return b;

}

private void engineCapacityTrackbar\_Scroll(object sender, EventArgs e)

{

engineValue.Text = (engineCapacityTrackbar.Value / 10.0).ToString();

}

}

}

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;

using CourseProject.Controller;

using CourseProject.Model;

namespace CourseProject.View

{

//Форма клиента

public partial class ClientForm : Form

{

private int ID;

public ClientForm(int id)

{

InitializeComponent();

//XmlHandler.Create();

//XmlHandler.Save(new Auto(), new User(), new User());

//comboBox1.DataSource = DataHandler.LoadBrands();

ID = id;

brandBox.DropDownStyle = ComboBoxStyle.DropDownList;

selectedAutosLV.FullRowSelect = true;

minPrice.Value = minDist.Value = 0;

minEngine.Value = 1;

minEngineLabel.Text = "0.1 куб.м";

maxPrice.Value = maxEngine.Value = maxDist.Value = maxPrice.Maximum;

maxVal.Text = (maxPrice.Maximum\*1000).ToString() + '$';

maxEngineLabel.Text = (maxEngine.Maximum/10.0).ToString() + " куб.м";

maxDistLabel.Text = (maxDist.Maximum \* 10).ToString() + " тыс.км";

}

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

{

List<Auto> autos = new List<Auto>();

bool error = false;

try

{

string brand = (brandBox.SelectedItem == null) ? null : brandBox.SelectedItem.ToString();

autos = DataHandler.SelectAutosByParameter(minPrice.Value \* 1000, maxPrice.Value \* 1000,

brand, minDist.Value \* 10,

maxDist.Value \* 10, minEngine.Value / 10.0, maxEngine.Value / 10.0);

}

catch

{

error = true;

minPrice.Value = minDist.Value = 0;

minEngine.Value = 1;

maxPrice.Value = maxEngine.Value = maxDist.Value = maxPrice.Maximum;

MessageBox.Show("Введите корректные данные");

}

selectedAutosLV.Items.Clear();

if (autos.Count == 0 && !error) MessageBox.Show("По данному запросу ничего не найдено");

for (int i = 0; i < autos.Count; i++)

{

ListViewItem lv = new ListViewItem(autos[i].Brand);

lv.Tag = autos[i];

lv.SubItems.Add(autos[i].Model);

lv.SubItems.Add(autos[i].Price.ToString());

lv.SubItems.Add(autos[i].Distance.ToString());

selectedAutosLV.Items.Add(lv);

}

}

private void minPrice\_Scroll(object sender, EventArgs e)

{

minVal.Text = (minPrice.Value\*1000).ToString()+'$';

if (minPrice.Value > maxPrice.Value )

{

maxPrice.Value = minPrice.Value;

maxVal.Text = (minPrice.Value\*1000).ToString() + '$';

}

}

private void maxPrice\_Scroll(object sender, EventArgs e)

{

maxVal.Text = (maxPrice.Value\*1000).ToString() + '$';

if (minPrice.Value > maxPrice.Value)

{

minPrice.Value = maxPrice.Value;

minVal.Text = (minPrice.Value\*1000).ToString() + '$';

}

}

private void minDist\_Scroll\_1(object sender, EventArgs e)

{

minDistLabel.Text = (minDist.Value \* 10).ToString() + " тыс.км";

if (minDist.Value > maxDist.Value)

{

maxDist.Value = minDist.Value;

maxDistLabel.Text = (minDist.Value \* 10).ToString() + " тыс.км";

}

}

private void maxDist\_Scroll(object sender, EventArgs e)

{

maxDistLabel.Text = (maxDist.Value \* 10).ToString() + " тыс.км";

if (minDist.Value > maxDist.Value)

{

minDist.Value = maxDist.Value;

minDistLabel.Text = (minDist.Value \* 10).ToString() + " тыс.км";

}

}

private void minEngine\_Scroll(object sender, EventArgs e)

{

minEngineLabel.Text = (minEngine.Value/10.0).ToString() + " куб.м";

if (minEngine.Value > maxEngine.Value)

{

maxEngine.Value = minEngine.Value;

maxEngineLabel.Text = (minEngine.Value/10.0).ToString() + " куб.м";

}

}

private void maxEngine\_Scroll(object sender, EventArgs e)

{

maxEngineLabel.Text = (maxEngine.Value/10.0).ToString() + " куб.м";

if (minEngine.Value > maxEngine.Value)

{

minEngine.Value = maxEngine.Value;

minEngineLabel.Text = (minEngine.Value/10.0).ToString() + " куб.м";

}

}

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

{

if (selectedAutosLV.SelectedItems.Count != 0)

{

var auto = (Auto) selectedAutosLV.SelectedItems[0].Tag;

var info = new AutoInformation((Auto)selectedAutosLV.SelectedItems[0].Tag);

info.ShowDialog();

}

}

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

{

if (selectedAutosLV.SelectedItems.Count != 0)

{

var auto = (Auto)selectedAutosLV.SelectedItems[0].Tag;

XmlHandler.Save(auto, DataHandler.GetUserById(ID), DataHandler.GetUserById(auto.SellerID));

XmlHandler.LoadUserPurchases(ID);

}

}

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

{

if (selectedAutosLV.SelectedItems.Count != 0)

{

var auto = (Auto)selectedAutosLV.SelectedItems[0].Tag;

MessageBox.Show(ApiHandler.GenerateArray(auto).ToString());

}

}

}

}

using CourseProject.Controller;

using CourseProject.Model;

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 CourseProject.View

{

public partial class SellerForm : Form

//Форма продавца

{

int ID;

public SellerForm(int id)

{

InitializeComponent();

ID = id;

//listView1.FullRowSelect = true;

dataGridView1.DataSource = SellerManager.LoadAutoList(id);

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.Columns["SellerID"].Visible = false;

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

dataGridView1.ReadOnly = true;

dataGridView1.AllowUserToAddRows = false;

// dataGridView1.Refresh();

//listView1. = SellerManager.LoadAutoList(ID);

//showAutos();

}

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

{

//SellerManager.AddAuto()

AddAutoForm form = new AddAutoForm(ID);

form.ShowDialog();

dataRefresh();

}

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

{

AddAutoForm form = new AddAutoForm(ID, (Auto)dataGridView1.SelectedRows[0].DataBoundItem);

form.ShowDialog();

dataRefresh();

}

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

{

if (dataGridView1.SelectedRows.Count != 0)

{

Auto auto = (Auto)dataGridView1.SelectedRows[0].DataBoundItem;

if (auto != null)

SellerManager.DeleteAuto(auto.ID);

dataRefresh();

}

//listView1.Refresh();

}

private void dataRefresh()

{

dataGridView1.DataSource = null;

dataGridView1.DataSource = SellerManager.LoadAutoList(ID);

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.Columns["SellerID"].Visible = false;

}

}

}

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;

using System.Data.SqlClient;

using CourseProject.Controller;

namespace CourseProject.View

{

public partial class AutorizationForm : Form

//Форма авторизации

{

SqlConnection cn = new SqlConnection(@"Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename='C:\Users\Дмитрий\documents\visual studio 2015\Projects\CourseProject\CourseProject\Autoshow.mdf';Integrated Security = True; Connect Timeout = 30");

public AutorizationForm()

{

InitializeComponent();

}

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

{

int userId = DataHandler.IsUserExist(loginTextBox.Text, passwordTextBox.Text, clientRadioButton.Checked);

if (userId != 0)

{

if (clientRadioButton.Checked)

{

ClientForm clientForm = new ClientForm(userId);

this.Hide();

clientForm.ShowDialog();

this.Close();

}

else

{

SellerForm sellerForm = new SellerForm(userId);

this.Hide();

sellerForm.ShowDialog();

this.Close();

}

}

else

{

MessageBox.Show("Неправильный логин или пароль");

loginTextBox.Clear();

passwordTextBox.Clear();

}

}

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

{

SignUpForm signUp = new SignUpForm();

//this.Hide();

this.Enabled = false;

signUp.Show();

signUp.FormClosed += SignUp\_FormClosed;

}

private void SignUp\_FormClosed(object sender, FormClosedEventArgs e)

{

this.Enabled = true;

}

}

}

using CourseProject.Model;

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 CourseProject.View

{

public partial class AutoInformation : Form

{

// Форма отображения информации об авто

public AutoInformation(Auto auto)

{

InitializeComponent();

var user = Controller.DataHandler.GetUserById(auto.SellerID);

nameLabel.Text = user.Name;

surnameLabel.Text = user.Surname;

phoneLabel.Text = user.Phone;

brandLabel.Text = auto.Brand;

modelLabel.Text = auto.Model;

distLabel.Text = auto.Distance.ToString();

engineLabel.Text = auto.Engine.ToString();

priceLabel.Text = auto.Price.ToString();

colorLabel.Text = auto.Color;

statusLabel.Text = (auto.Status == 0) ? "В активном поиске" : "Продано";

}

}

}

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;

using System.Data.SqlClient;

namespace CourseProject.View

{

public partial class SignUpForm : Form

//Форма регистрации

{

SqlConnection cn = new SqlConnection(@"Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename='C:\Users\Дмитрий\documents\visual studio 2015\Projects\CourseProject\CourseProject\Autoshow.mdf';Integrated Security = True; Connect Timeout = 30");

SqlCommand cmd = new SqlCommand();

public SignUpForm()

{

InitializeComponent();

}

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

{

if (Controller.DataHandler.UserRegistration(firstNameTextBox.Text, secondNameTextBox.Text, emailTextBox.Text,

passwordTextBox.Text,

phoneTextBox.Text, radioButton1.Checked))

{

this.Close();

}

else MessageBox.Show("Неправильный ввод");

}

}

}