Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

Тема проекта «Продажа автомобилей’LADA’»

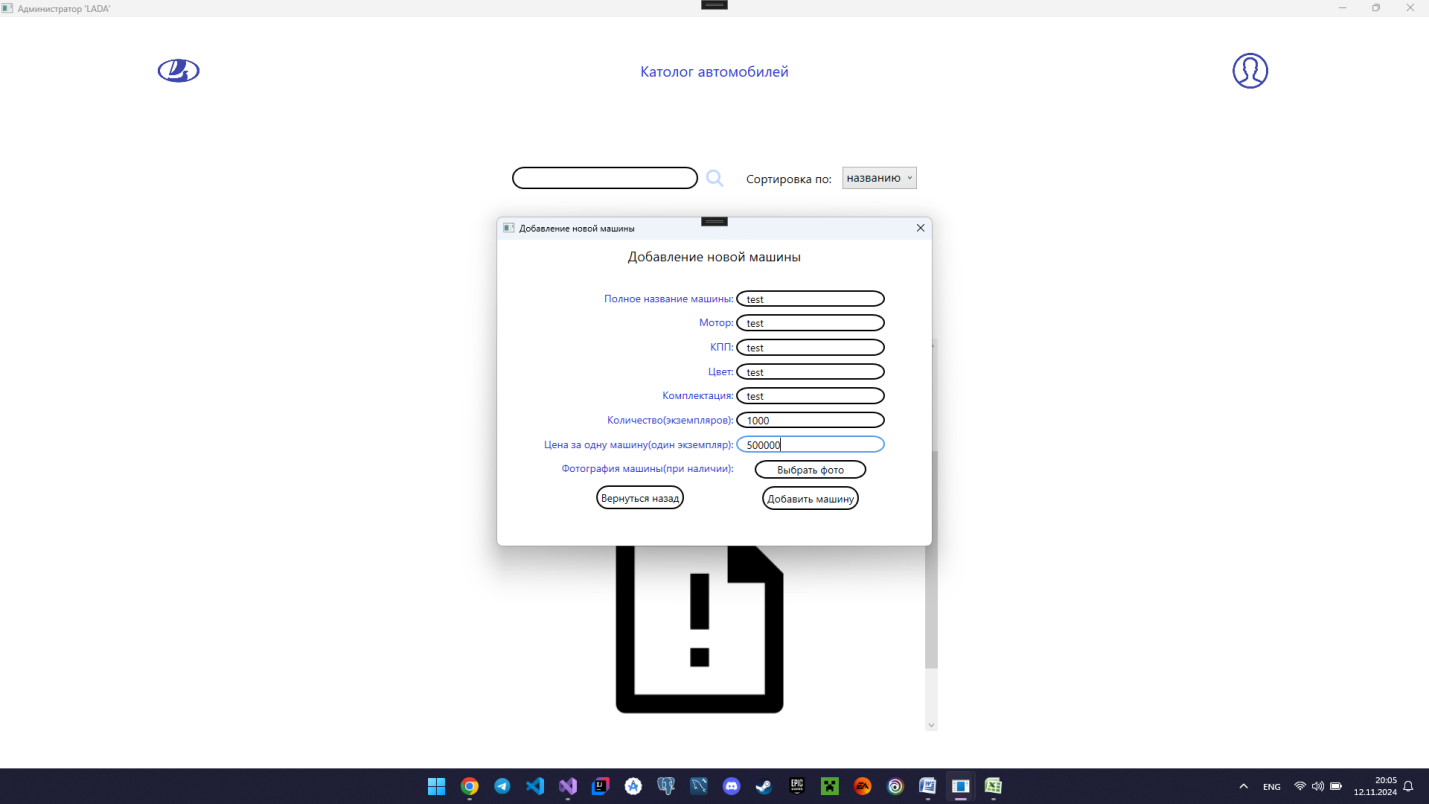
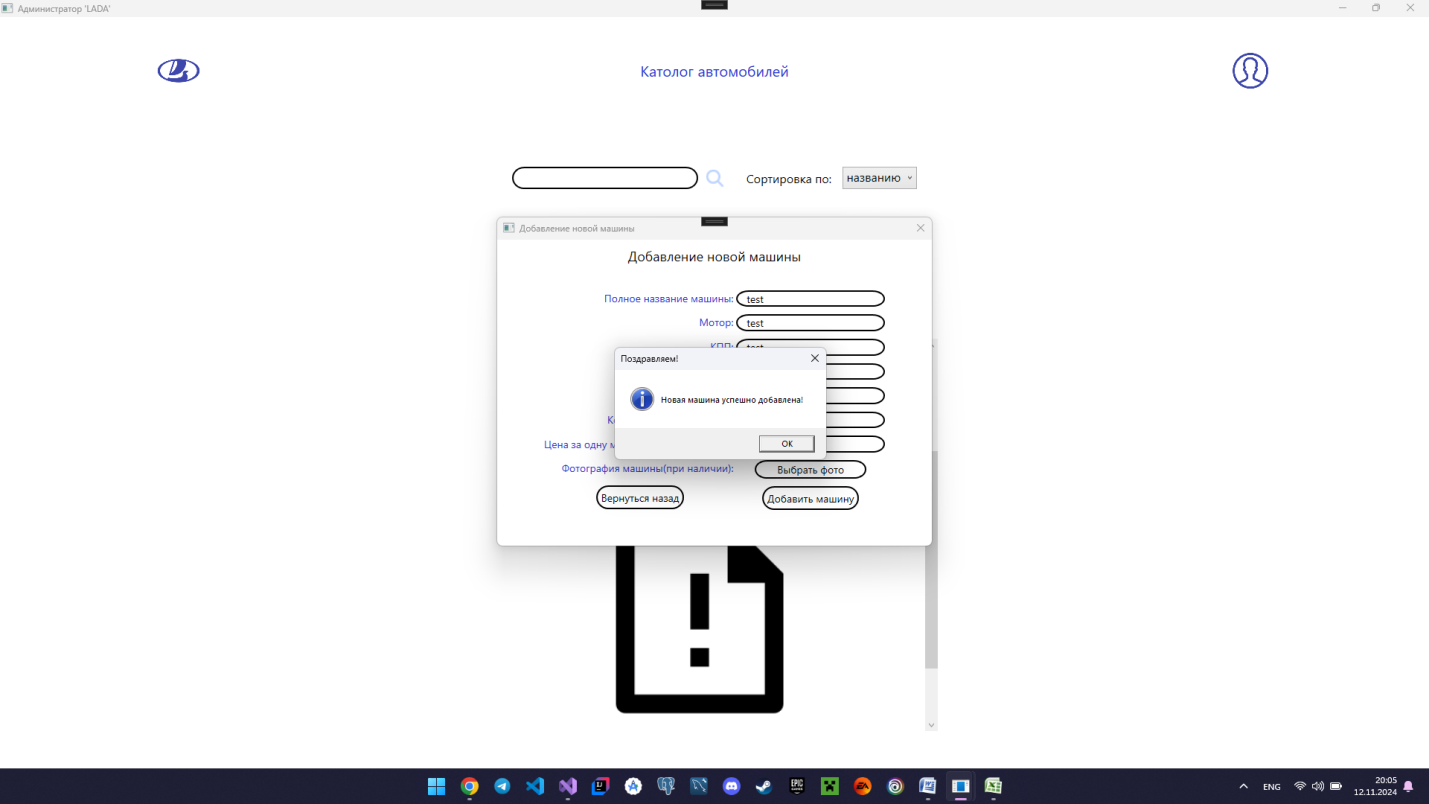
Практическая работа №3 «Вывод товаров из JSON файла в UserControl VS C# с функциональностью добавления, удаления, редактирования, обновления, поиска, сортировки и фильтрации»

МДК.01.02 Поддержка и тестирование программных модулей

|  |  |
| --- | --- |
|  | Преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Л.Р. Набиева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  | Студент гр. 22П-3  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Семенов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

2024

Описание теста: Успешное добавление товара.

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.ConstrainedExecution;

using System.Text;

using System.Text.Json;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Authorization

{

/// <summary>

/// Логика взаимодействия для AddNewItem.xaml

/// </summary>

public partial class AddNewItem : Window

{

string pathToJsonFileCars;

List<Car> cars;

string pathToDirectory = @"../../img/cars/";

string oldPathPhoto = @"";

string pathToPhoto = @"";

public AddNewItem(List<Car> cars, string pathToJsonFileCars)

{

InitializeComponent();

this.cars = cars;

this.pathToJsonFileCars = pathToJsonFileCars;

}

private void backButton\_Click(object sender, RoutedEventArgs e)

{

this.DialogResult = false;

}

private void addNewCarButton\_Click(object sender, RoutedEventArgs e)

{

List<TextBox> textBoxes = new List<TextBox>() { fullNameCarTextBox, motorCarTextBox, gearBoxTextBox, colorCarTextBox, configurationCarTextBox, amountCarTextBox, pricePerCarTextBox };

AddNewCar(textBoxes);

}

private string CheckToAddNewCar(List<TextBox> textBoxes)

{

foreach (TextBox textBox in textBoxes)

{

if (textBox.Text.Length == 0) return "Не все поля введены!";

if (!Regex.IsMatch(textBox.Text.ToLower(), @"^[a-zа-яё0-9.\s]+$")) return "Введены не корректные символы в полях!";

}

int res1 = 0;

double res = 0;

if (!int.TryParse(textBoxes[5].Text, out res1) || !Double.TryParse(textBoxes[6].Text, out res)) return "Введены не корректные символы в количестве или цене!";

return null;

}

private void AddNewCar(List<TextBox> textBoxes)

{

var error = CheckToAddNewCar(textBoxes);

if (error != null)

{

MessageBox.Show(error, "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var car = FindExistingCar(textBoxes[0].Text, textBoxes[1].Text, textBoxes[2].Text, textBoxes[3].Text, textBoxes[4].Text, double.Parse(textBoxes[6].Text));

if (car != null) AddCars(car, int.Parse(textBoxes[5].Text));

else CreateNewClassCar(textBoxes);

SaveDataToJson();

MessageBox.Show("Новая машина успешно добавлена!", "Поздравляем!", MessageBoxButton.OK, MessageBoxImage.Information);

this.DialogResult = true;

}

private Car FindExistingCar(string fullNameCar, string motorCar, string gearBoxCar, string colorCar, string configurationCar, double pricePerCar)

{

int id = -1;

foreach(Car car in cars)

{

if (car.FullName == fullNameCar && car.Motor == motorCar && car.Gearbox == gearBoxCar && car.Color == colorCar && car.Configuration == configurationCar && car.PricePerCar == pricePerCar) return car;

}

return null;

}

private void AddCars(Car car, int amount)

{

car.Amount += amount;

}

private void CreateNewClassCar(List<TextBox> textBoxes)

{

if (pathToPhoto.Length > 0) SavePhotoToBaseApp();

Car car = new Car(textBoxes[0].Text, textBoxes[1].Text, textBoxes[2].Text, textBoxes[3].Text, textBoxes[4].Text, int.Parse(textBoxes[5].Text), double.Parse(textBoxes[6].Text), pathToPhoto);

car.FindUniqueId(cars);

cars.Add(car);

}

private void SaveDataToJson()

{

var jsonObj = JsonSerializer.Serialize(cars);

File.WriteAllText(pathToJsonFileCars, jsonObj);

}

private void imageCarPath\_Click(object sender, RoutedEventArgs e)

{

SelectImageCar();

}

private void SavePhotoToBaseApp()

{

using (FileStream fs = new FileStream(pathToPhoto, FileMode.Create)) { }

File.Copy(oldPathPhoto, pathToPhoto, true);

}

private void SelectImageCar()

{

var openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Image Files(\*.jpg, \*.png, \*.jpeg)|\*.jpg|\*.png|\*.jpeg";

if (openFileDialog.ShowDialog() == false)

{

MessageBox.Show("Фотография не добавлена!", "Внимание!", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

oldPathPhoto = openFileDialog.FileName;

pathToPhoto = CreateFileName();

MessageBox.Show("Фотография добавлена", "Информация!", MessageBoxButton.OK, MessageBoxImage.Information);

}

private string CreateFileName()

{

string path;

path = $"{pathToDirectory}{oldPathPhoto.Split('\\')[oldPathPhoto.Split('\\').Length - 1]}";

if (!File.Exists(path)) return path;

int index = 2;

do

{

var lst = oldPathPhoto.Split('\\')[oldPathPhoto.Split('\\').Length - 1];

path = $"{pathToDirectory}({index}){lst}";

index++;

}

while (File.Exists(path));

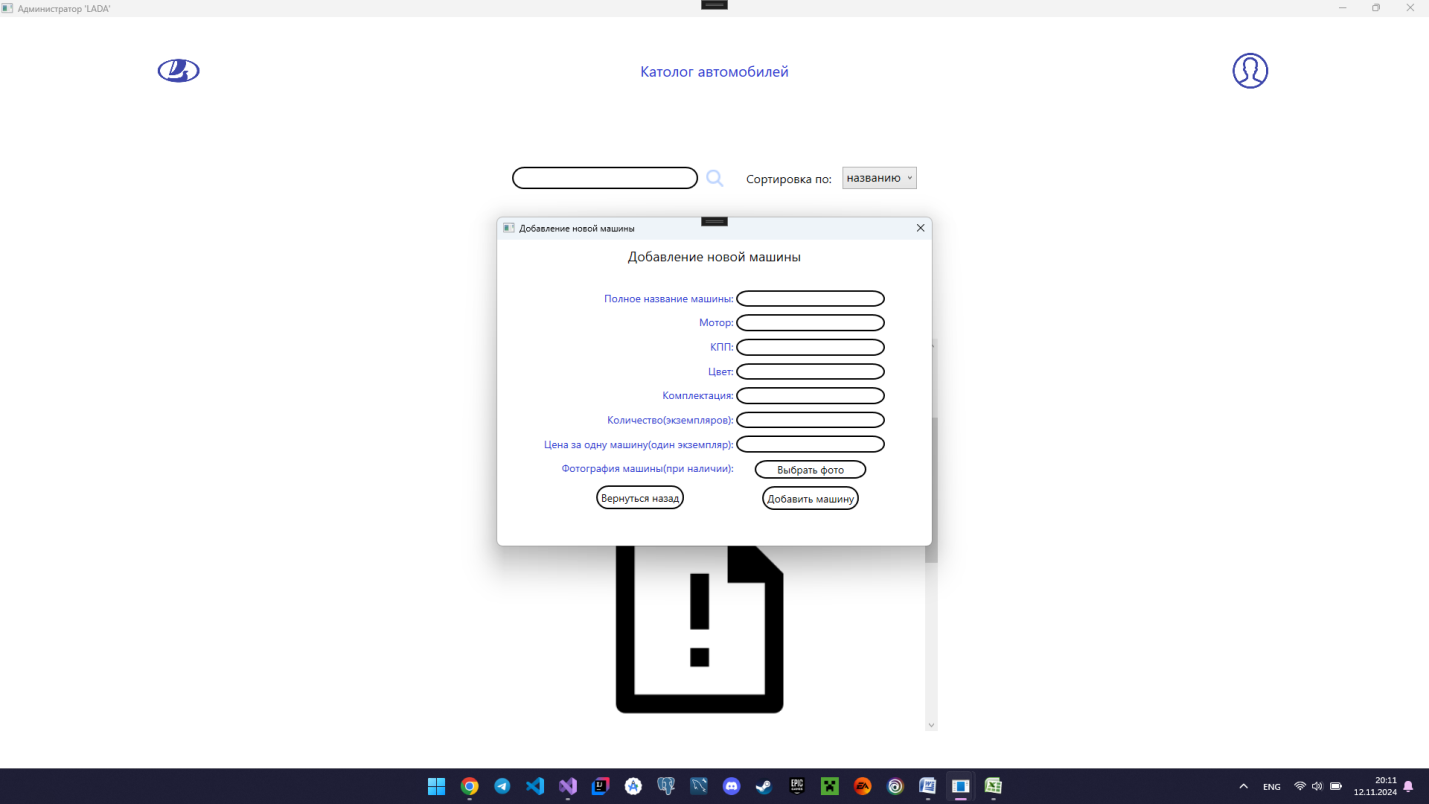
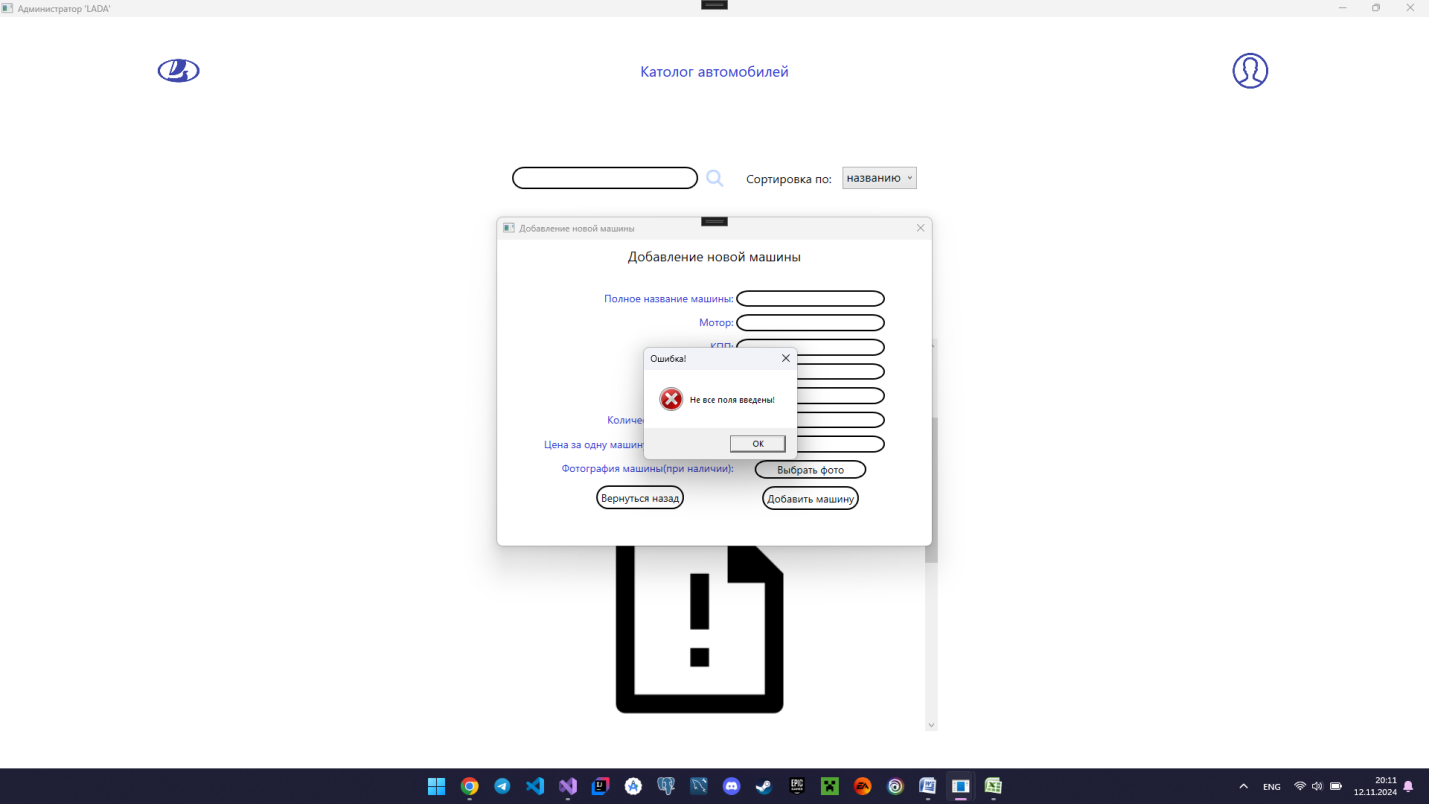
return path;

}

}

}

Описание теста: Неуспешное добавление товара с пустыми полями.

  private string CheckToAddNewCar(List<TextBox> textBoxes)

{

foreach (TextBox textBox in textBoxes)

{

if (textBox.Text.Length == 0) return "Не все поля введены!";

if (!Regex.IsMatch(textBox.Text.ToLower(), @"^[a-zа-яё0-9.\s]+$")) return "Введены не корректные символы в полях!";

}

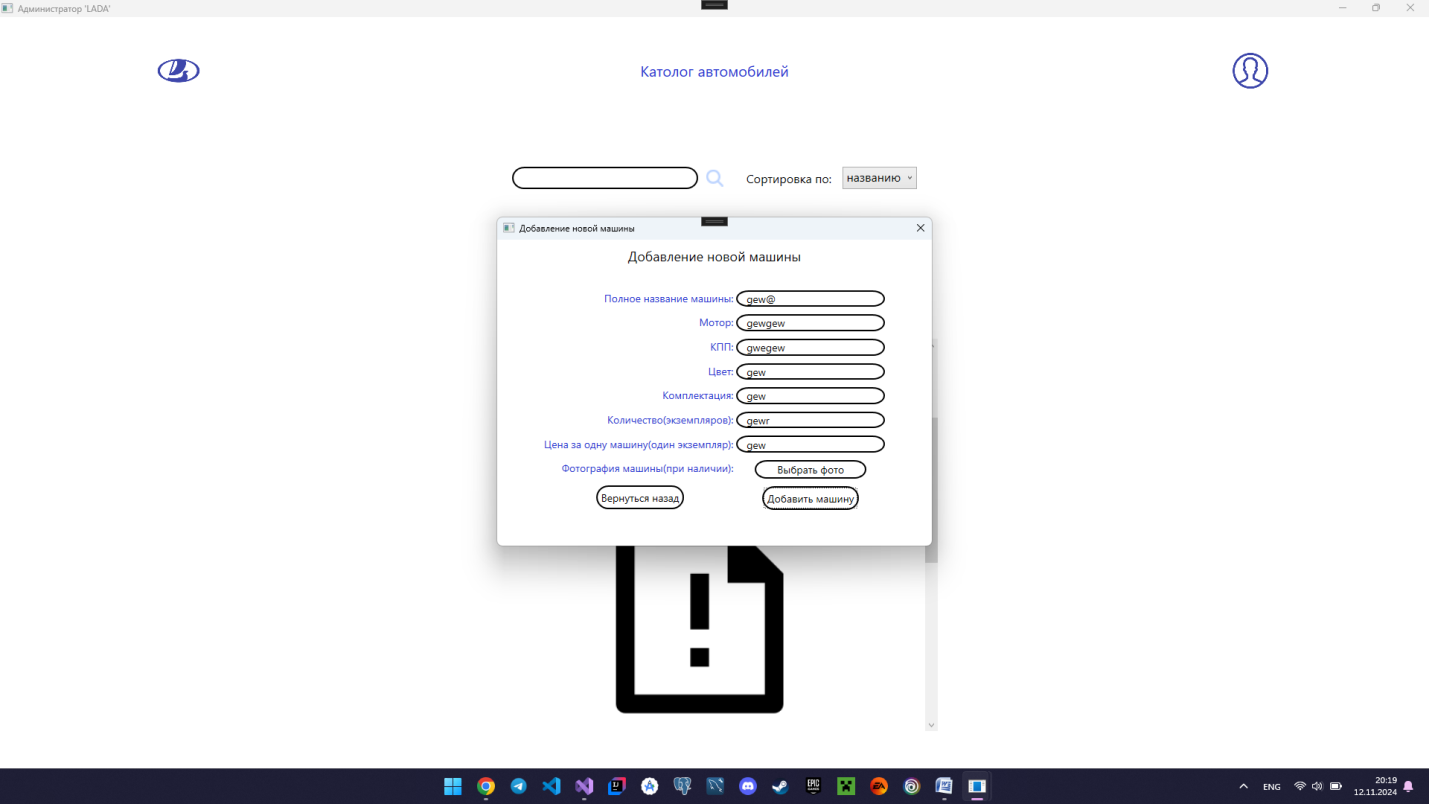
int res1 = 0;

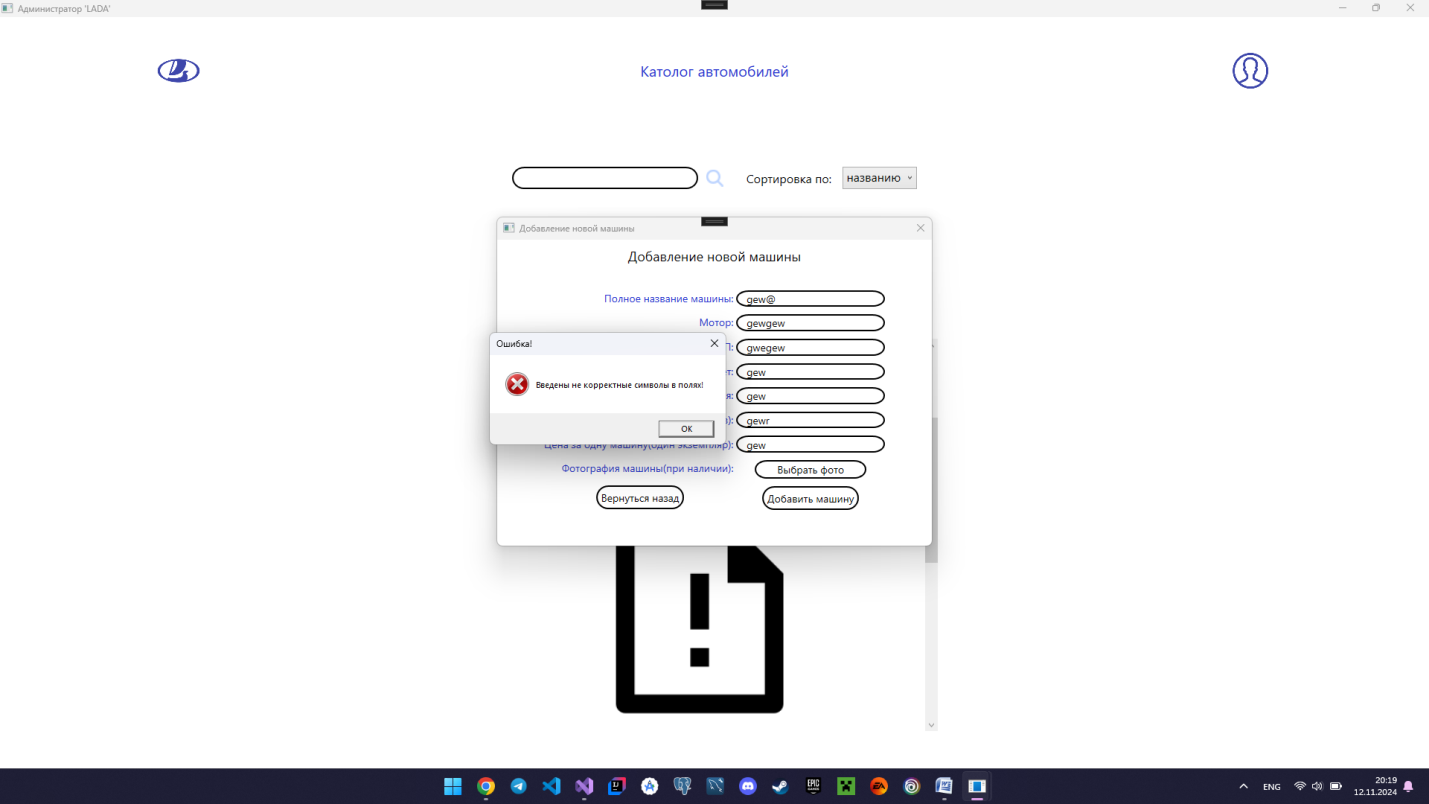
double res = 0;

if (!int.TryParse(textBoxes[5].Text, out res1) || !Double.TryParse(textBoxes[6].Text, out res)) return "Введены не корректные символы в количестве или цене!";

return null;

}

Описание теста: Неуспешное добавление товара с некорректными данными. 

 private string CheckToAddNewCar(List<TextBox> textBoxes)

{

foreach (TextBox textBox in textBoxes)

{

if (textBox.Text.Length == 0) return "Не все поля введены!";

if (!Regex.IsMatch(textBox.Text.ToLower(), @"^[a-zа-яё0-9.\s]+$")) return "Введены не корректные символы в полях!";

}

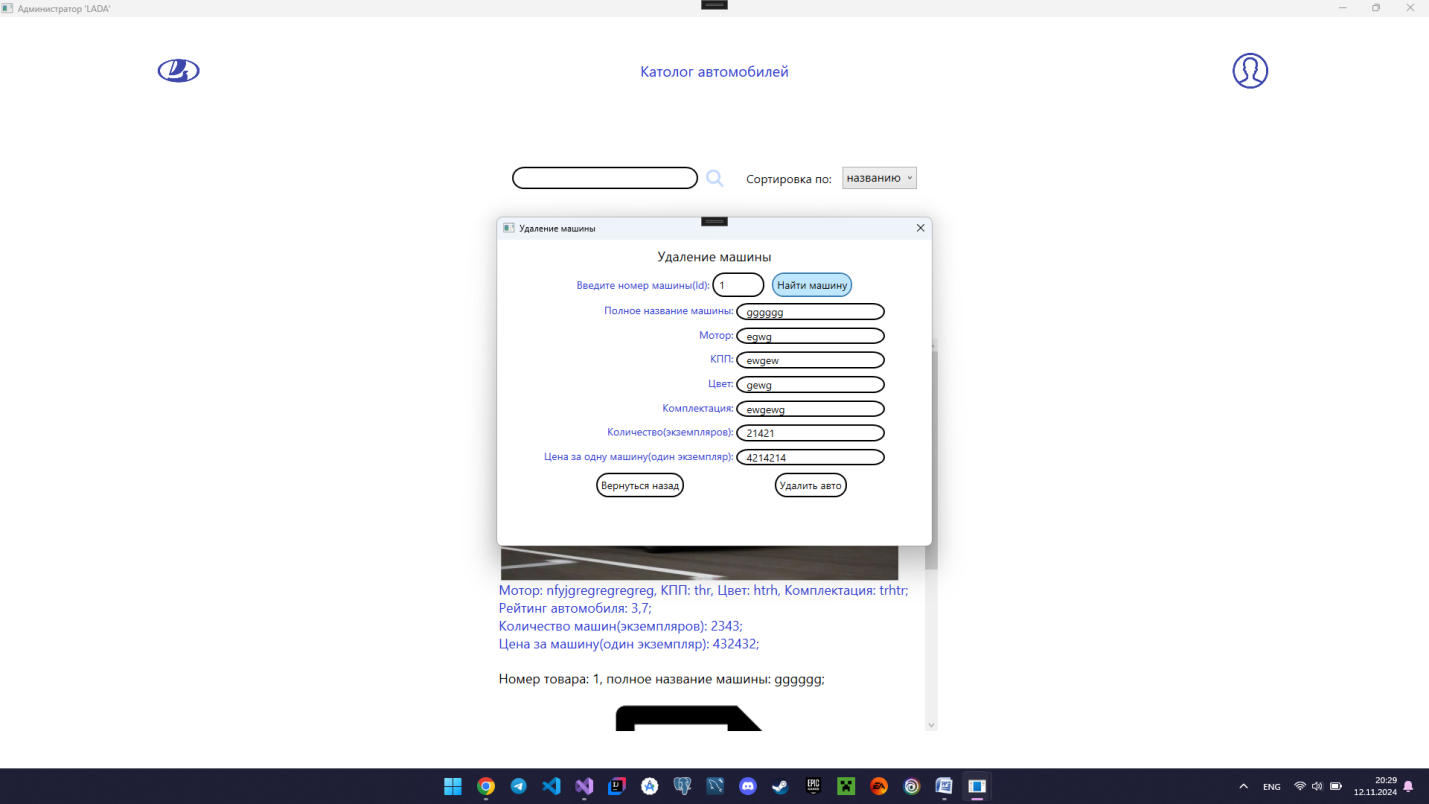
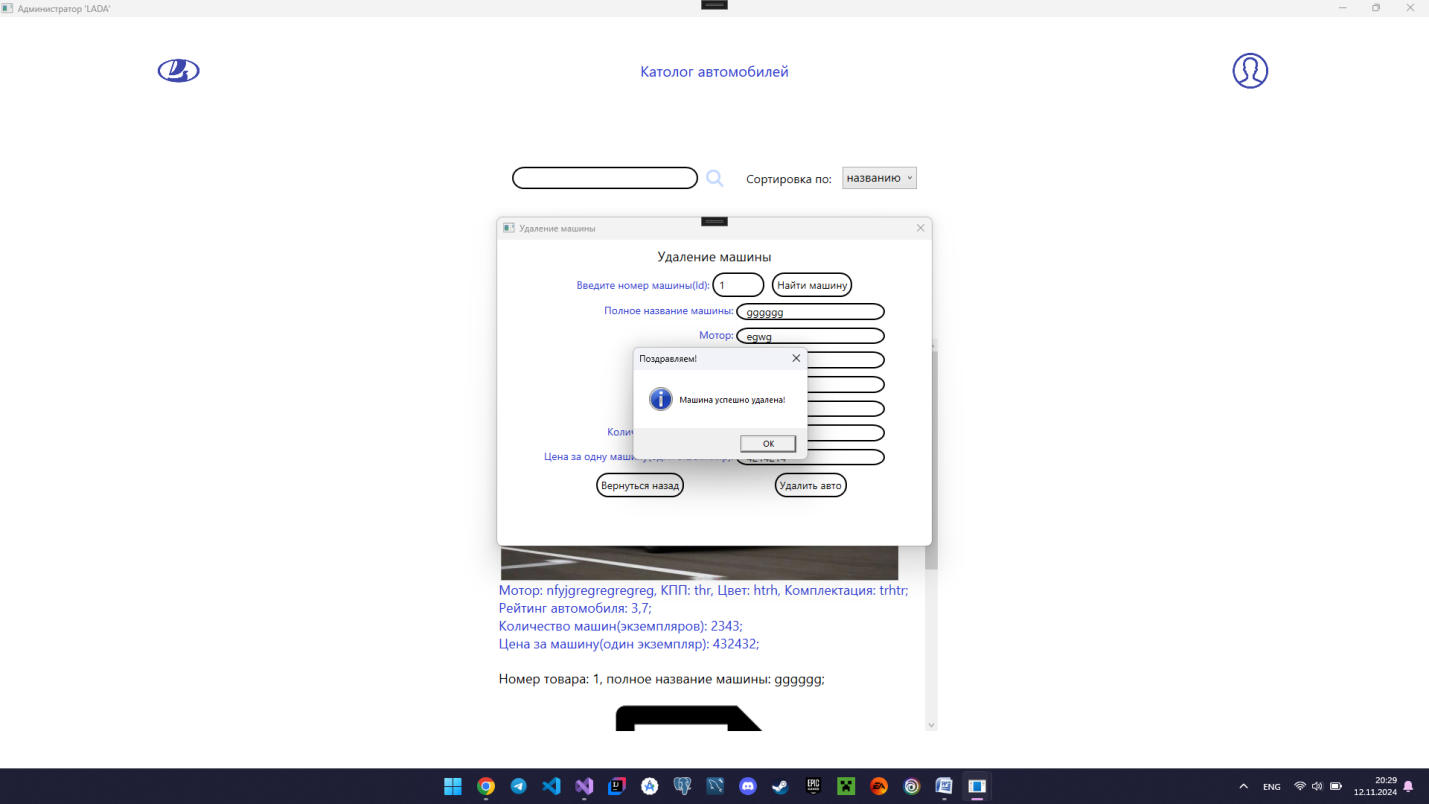
int res1 = 0;

double res = 0;

if (!int.TryParse(textBoxes[5].Text, out res1) || !Double.TryParse(textBoxes[6].Text, out res)) return "Введены не корректные символы в количестве или цене!";

return null;

}

Описание теста: Успешное удаление товара.   using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Authorization

{

/// <summary>

/// Логика взаимодействия для DeleteCar.xaml

/// </summary>

public partial class DeleteCar : Window

{

string IdCar;

Car Car;

List<Car> Cars;

string PathToFileJsonCars;

public DeleteCar(List<Car> cars, string pathToFileJsonCars)

{

InitializeComponent();

this.Cars = cars;

this.PathToFileJsonCars = pathToFileJsonCars;

}

private void deleteCarButton\_Click(object sender, RoutedEventArgs e)

{

DeleteCarOnId();

}

private void findCarOnIdButton\_Click(object sender, RoutedEventArgs e)

{

FindCar();

}

private Car OutpuCar()

{

foreach (Car car in Cars)

{

if (car.Id.ToString() == inputIdCarTextBox.Text) return car;

}

return null;

}

private void FindCar()

{

Car = OutpuCar();

if (Car == null)

{

MessageBox.Show("Машина с данным номером(Id) не найдена!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

IdCar = inputIdCarTextBox.Text;

AddDataToTextBox();

}

private void AddDataToTextBox()

{

fullNameCarTextBox.Text = Car.FullName;

motorCarTextBox.Text = Car.Motor;

gearBoxTextBox.Text = Car.Gearbox;

colorCarTextBox.Text = Car.Color;

configurationCarTextBox.Text = Car.Configuration;

amountCarTextBox.Text = Car.Amount.ToString();

pricePerCarTextBox.Text = Car.PricePerCar.ToString();

}

private void backButton\_Click(object sender, RoutedEventArgs e)

{

this.DialogResult = false;

}

private void DeleteCarOnId()

{

if (IdCar != inputIdCarTextBox.Text)

{

MessageBox.Show("Номер(Id) машины поменялся!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (MessageBox.Show("Вы точно хотите удалить эту машину?", "", MessageBoxButton.OKCancel) == MessageBoxResult.Cancel) return;

Cars.Remove(Car);

SaveDataToJson();

MessageBox.Show("Машина успешно удалена!", "Поздравляем!", MessageBoxButton.OK, MessageBoxImage.Information);

this.DialogResult = true;

}

private void SaveDataToJson()

{

var jsonObj = JsonSerializer.Serialize(Cars);

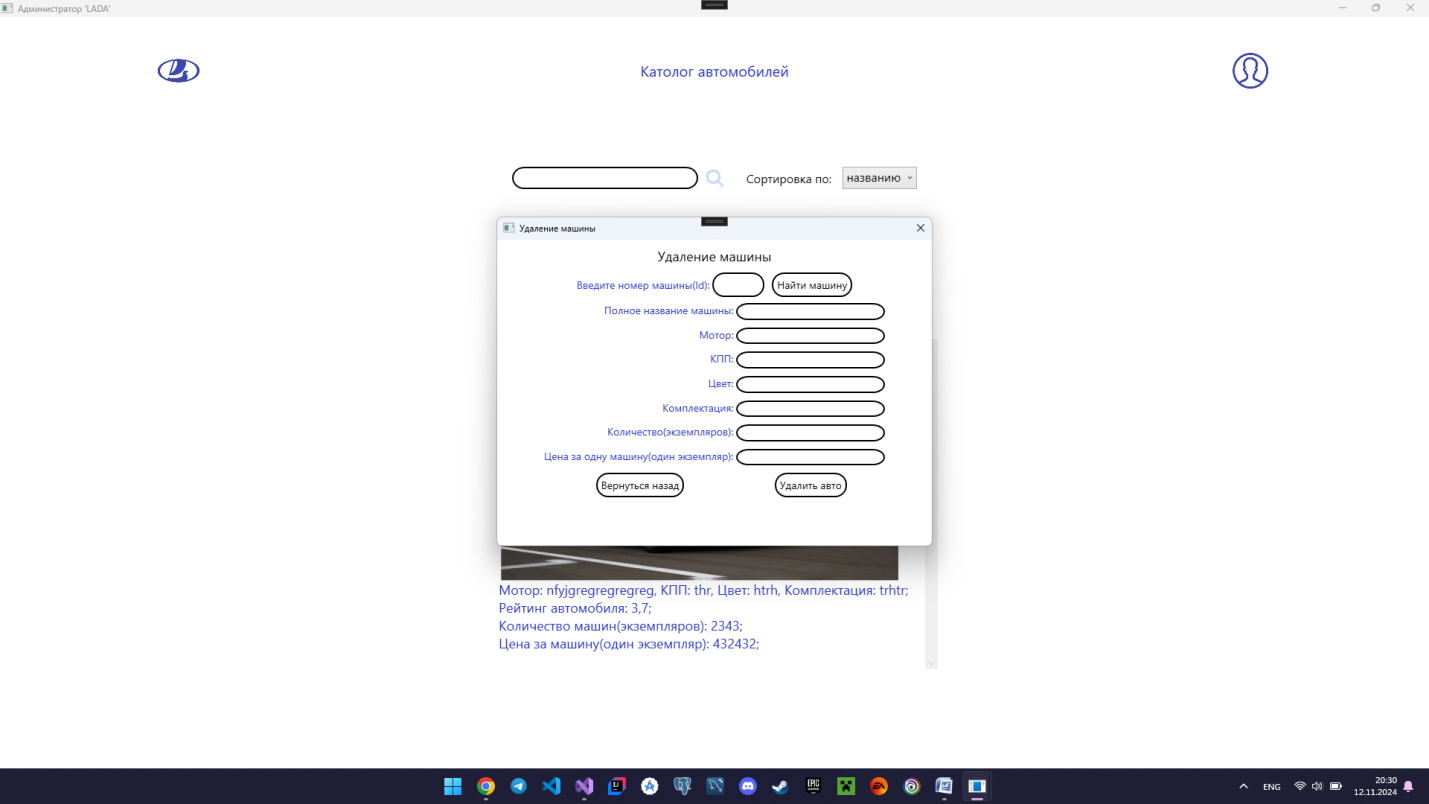
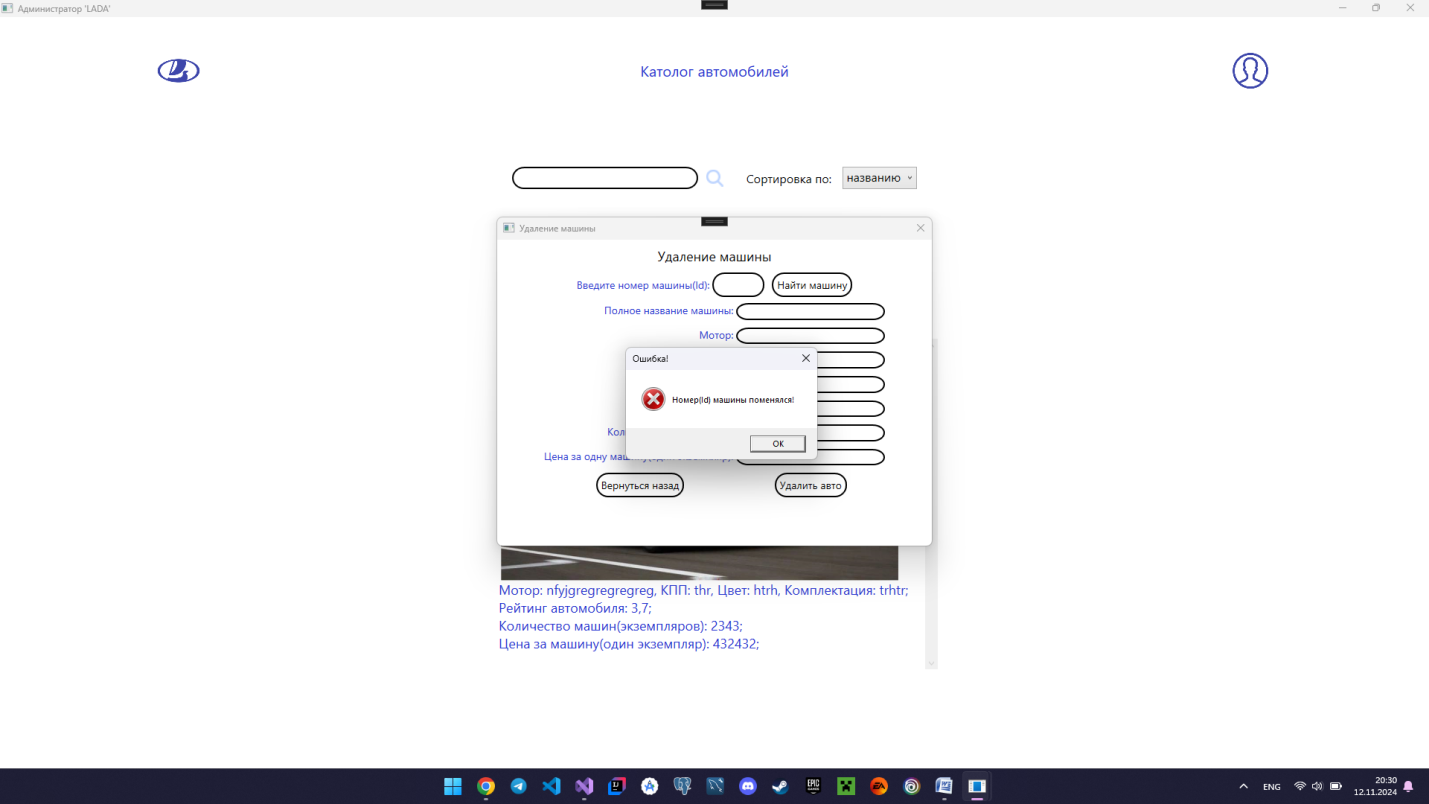
File.WriteAllText(PathToFileJsonCars, jsonObj);

}

}

}

Описание теста: Неуспешное удаление товара (нет выбранного товара).

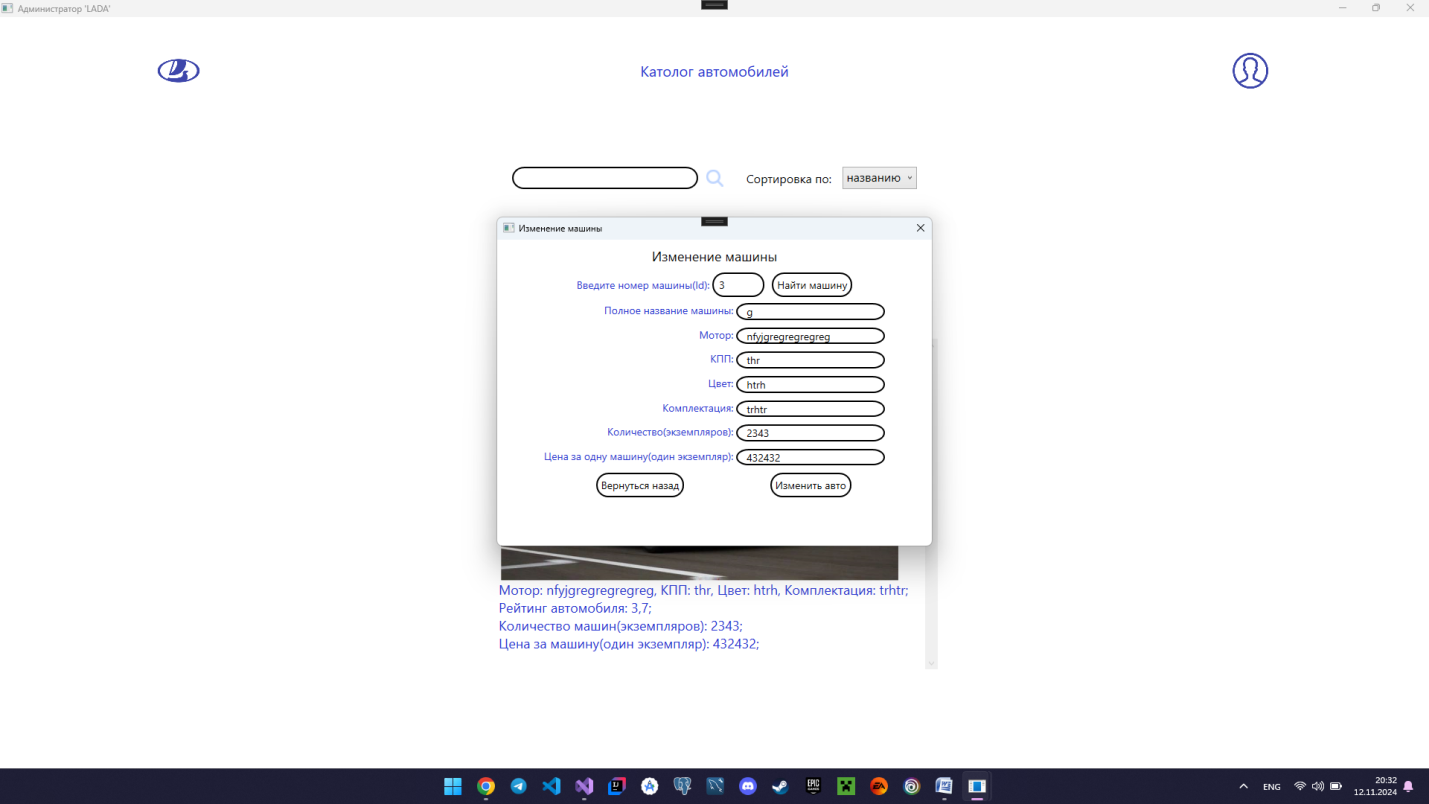
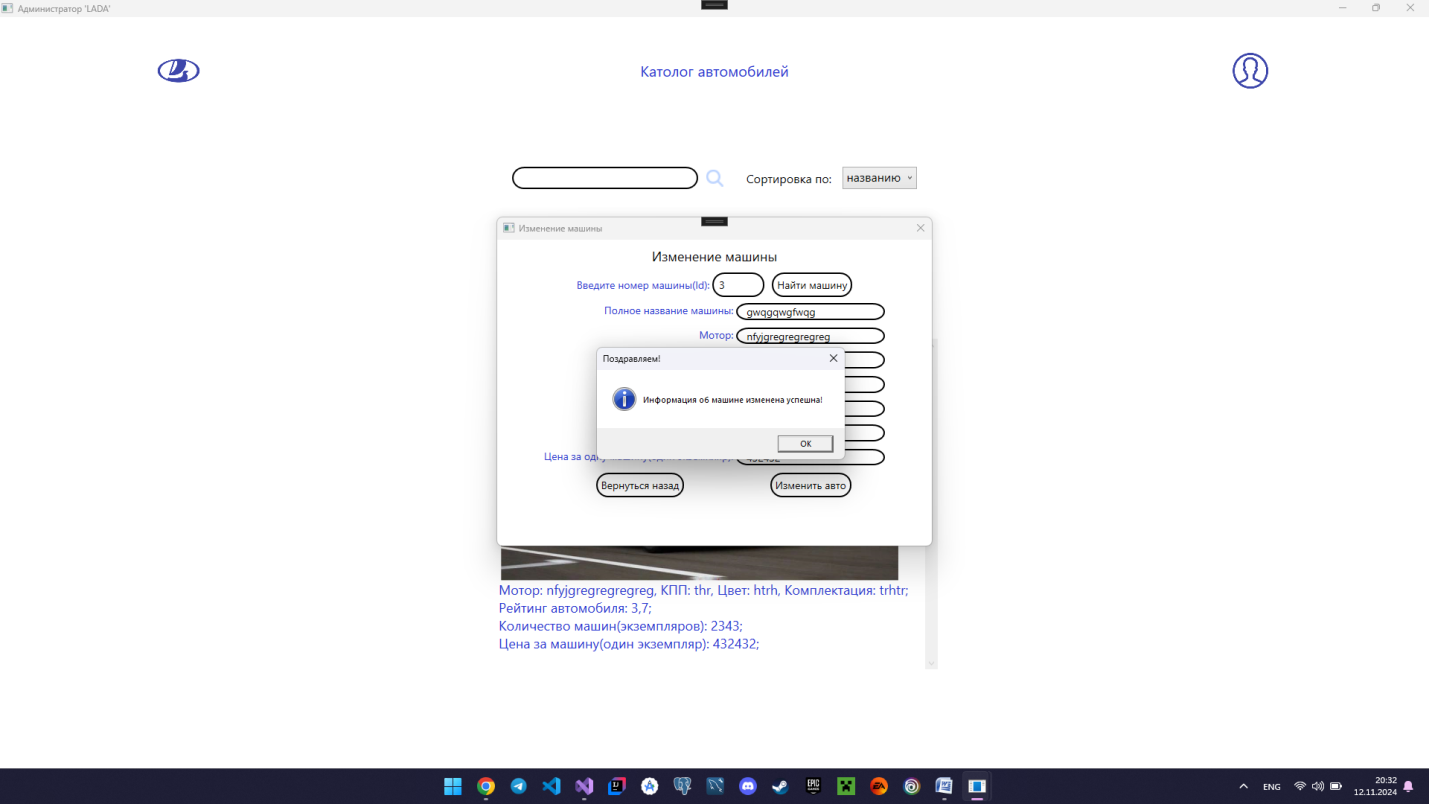
  if (IdCar != inputIdCarTextBox.Text)

{

MessageBox.Show("Номер(Id) машины поменялся!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

Описание теста: Успешное редактирование товара.   using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.Json;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Authorization

{

/// <summary>

/// Логика взаимодействия для ChanheCar.xaml

/// </summary>

public partial class ChanheCar : Window

{

string IdCar;

Car Car;

List<Car> Cars;

string PathToFileJsonCars;

public ChanheCar(List<Car> cars, string pathToFileJsonCars)

{

InitializeComponent();

this.Cars = cars;

this.PathToFileJsonCars = pathToFileJsonCars;

}

private void backButton\_Click(object sender, RoutedEventArgs e)

{

this.DialogResult = false;

}

private void findCarOnIdButton\_Click(object sender, RoutedEventArgs e)

{

FindCar();

}

private Car OutpuCar()

{

foreach(Car car in Cars)

{

if(car.Id.ToString() == inputIdCarTextBox.Text) return car;

}

return null;

}

private void FindCar()

{

Car = OutpuCar();

if (Car == null)

{

MessageBox.Show("Машина с данным номером(Id) не найдена!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

IdCar = inputIdCarTextBox.Text;

AddDataToTextBox();

}

private void AddDataToTextBox()

{

fullNameCarTextBox.Text = Car.FullName;

motorCarTextBox.Text = Car.Motor;

gearBoxTextBox.Text = Car.Gearbox;

colorCarTextBox.Text = Car.Color;

configurationCarTextBox.Text = Car.Configuration;

amountCarTextBox.Text = Car.Amount.ToString();

pricePerCarTextBox.Text = Car.PricePerCar.ToString();

}

private void changeCarButton\_Click(object sender, RoutedEventArgs e)

{

ChangeCar();

}

private string CheckToChangeCar(List<TextBox> textBoxes)

{

if (!CheckId())

{

return "Номер(Id) машины поменялся!";

}

foreach (TextBox textBox in textBoxes)

{

if (textBox.Text.Length == 0) return "Не все поля введены!";

if (!Regex.IsMatch(textBox.Text.ToLower(), @"^[a-zа-яё0-9.\s]+$")) return "Введены не корректные символы в полях!";

}

double res = 0;

if (!Double.TryParse(textBoxes[5].Text, out res) || !Double.TryParse(textBoxes[6].Text, out res)) return "Введены не корректные символы в количестве или цене!";

return null;

}

private bool CheckId()

{

return IdCar == inputIdCarTextBox.Text;

}

private void ChangeCar()

{

string error = CheckToChangeCar(new List<TextBox>() { fullNameCarTextBox, motorCarTextBox, gearBoxTextBox, colorCarTextBox, configurationCarTextBox, amountCarTextBox, pricePerCarTextBox });

if (error != null)

{

MessageBox.Show(error, "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

UpdateData();

MessageBox.Show("Информация об машине изменена успешна!", "Поздравляем!", MessageBoxButton.OK, MessageBoxImage.Information);

this.DialogResult = true;

}

private void UpdateData()

{

Cars.Remove(Car);

Car.FullName = fullNameCarTextBox.Text;

Car.Motor = motorCarTextBox.Text;

Car.Gearbox = gearBoxTextBox.Text;

Car.Color = colorCarTextBox.Text;

Car.Configuration = configurationCarTextBox.Text;

Car.Amount = int.Parse(amountCarTextBox.Text);

Car.PricePerCar = double.Parse(pricePerCarTextBox.Text);

Cars.Add(Car);

SaveDataToJson();

}

private void SaveDataToJson()

{

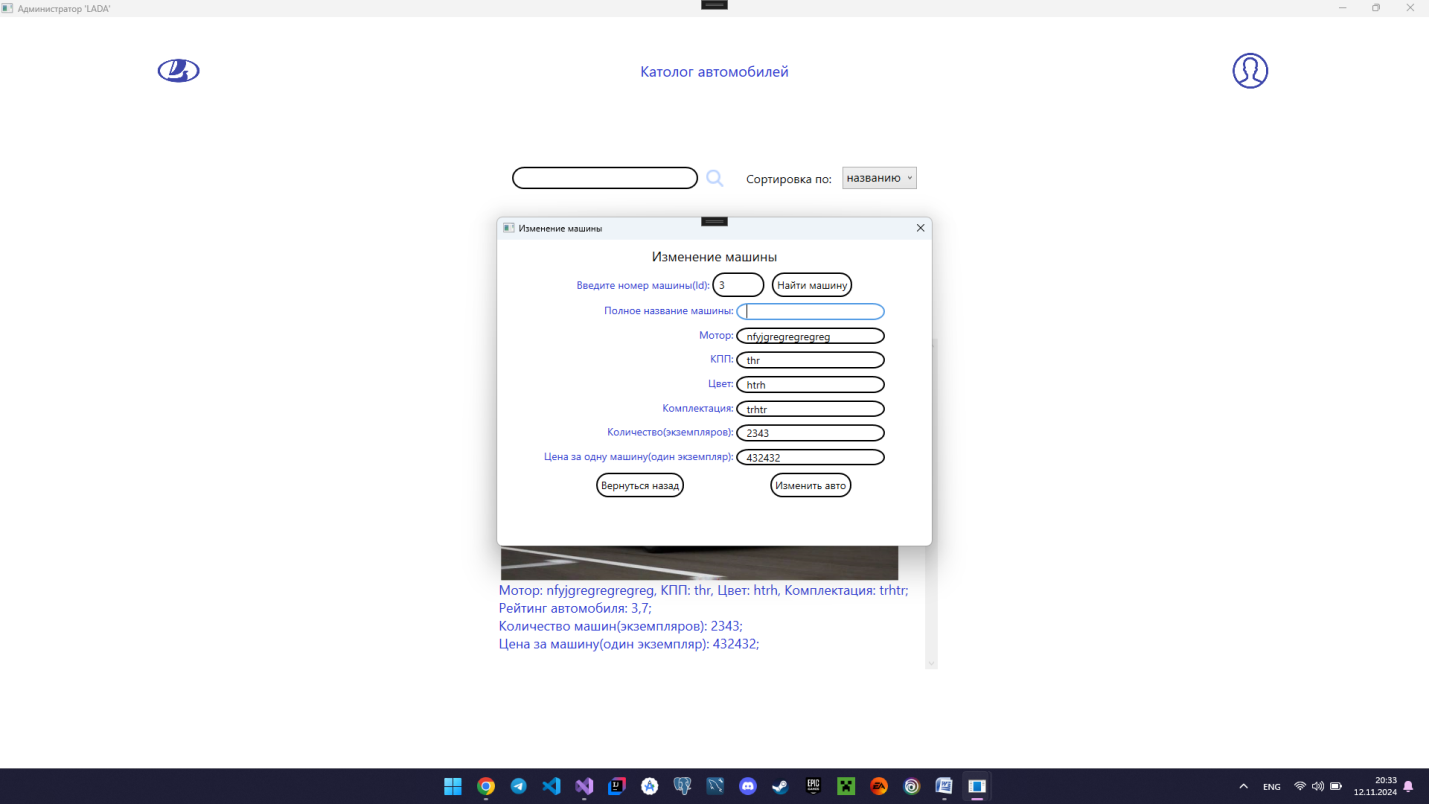
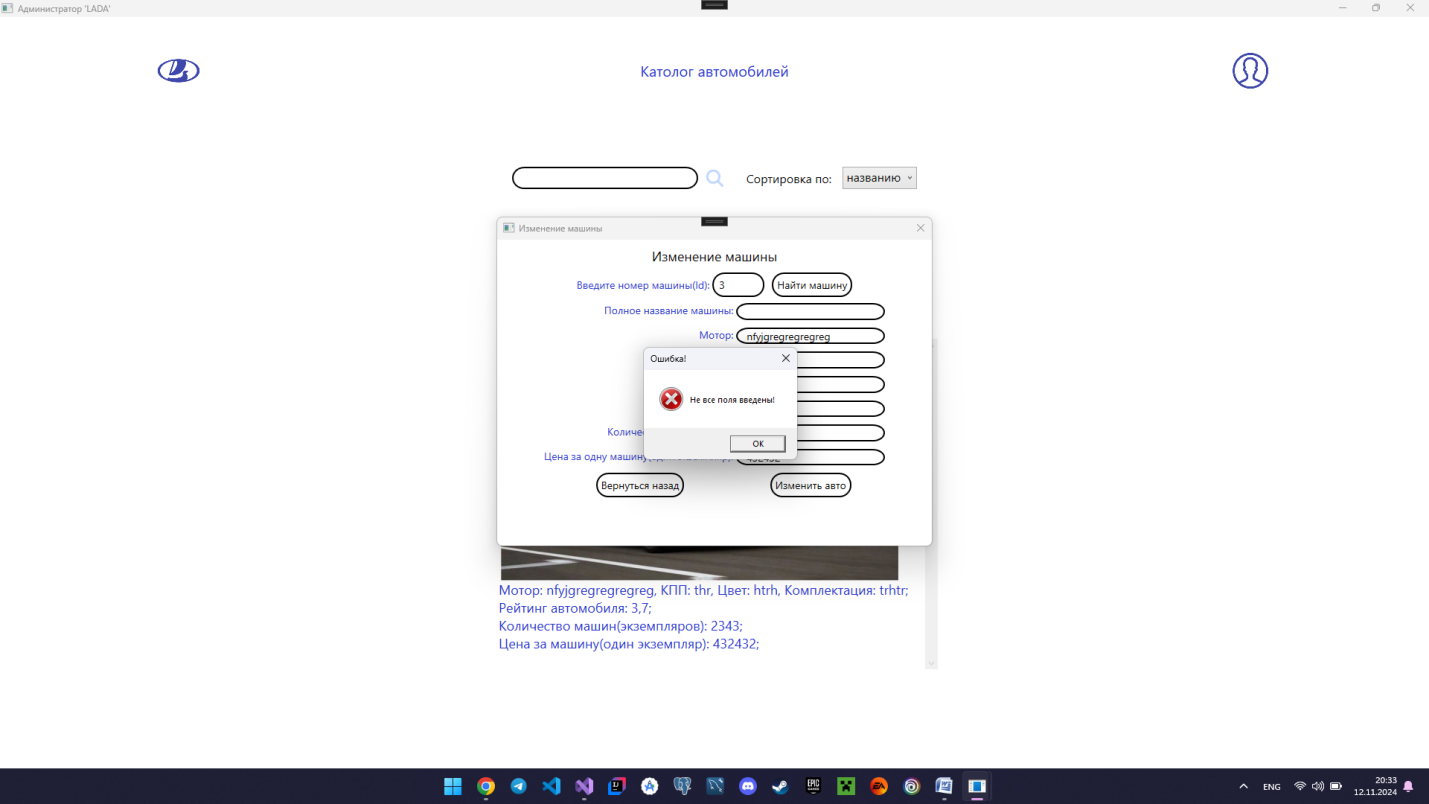
var jsonObj = JsonSerializer.Serialize(Cars);

File.WriteAllText(PathToFileJsonCars, jsonObj);

}

}

}

Описание теста: Неуспешное редактирование товара с пустыми полями.   private string CheckToChangeCar(List<TextBox> textBoxes)

{

if (!CheckId())

{

return "Номер(Id) машины поменялся!";

}

foreach (TextBox textBox in textBoxes)

{

if (textBox.Text.Length == 0) return "Не все поля введены!";

if (!Regex.IsMatch(textBox.Text.ToLower(), @"^[a-zа-яё0-9.\s]+$")) return "Введены не корректные символы в полях!";

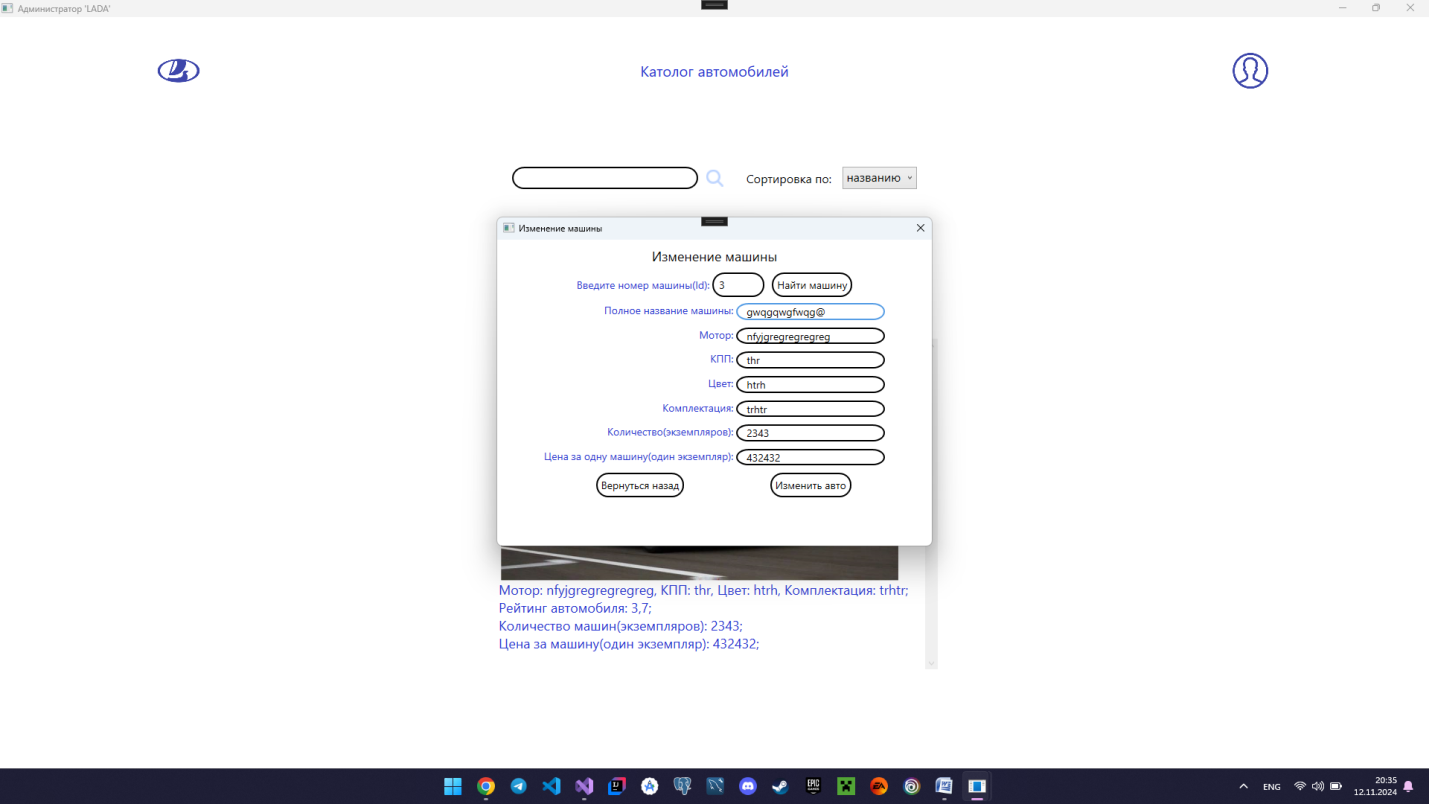
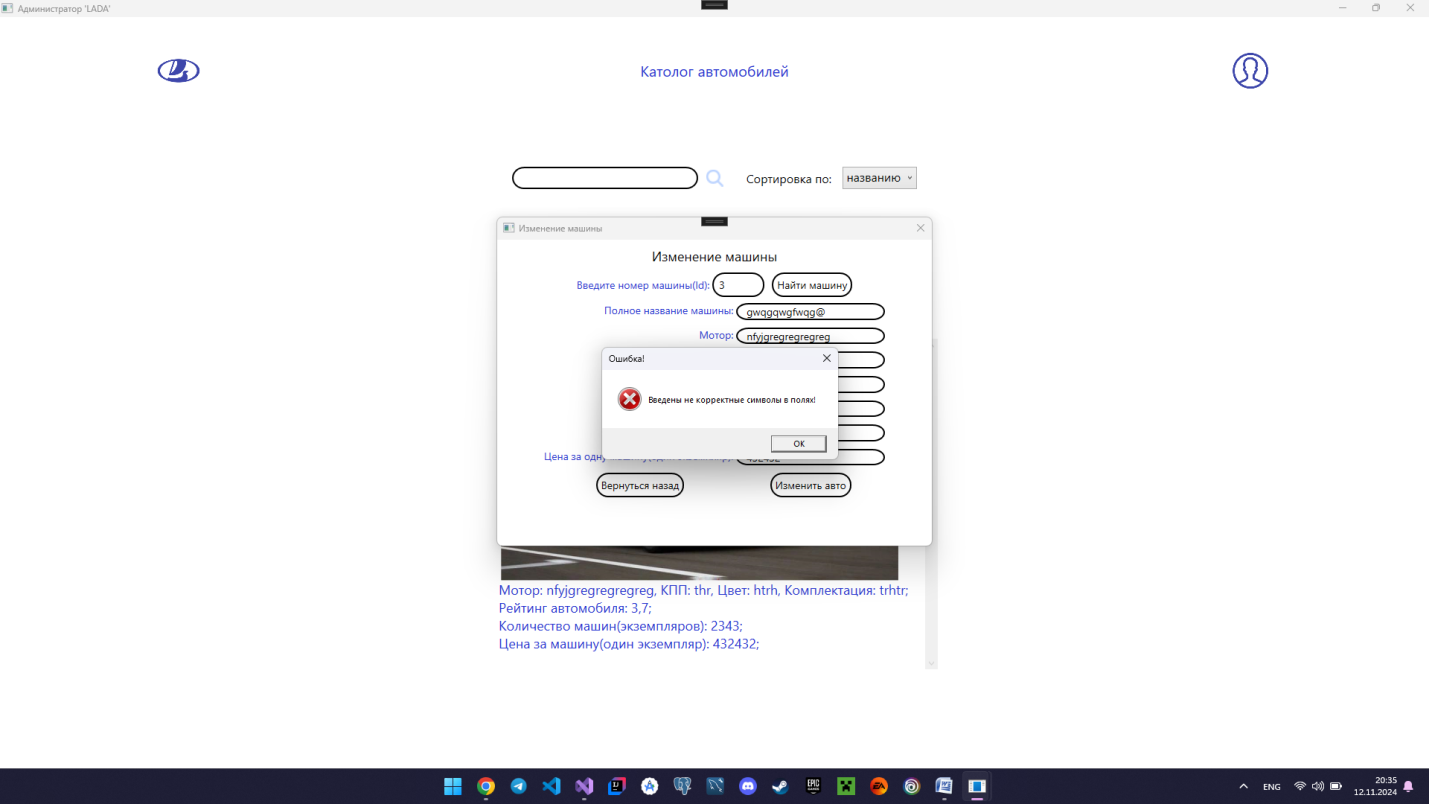
}

double res = 0;

if (!Double.TryParse(textBoxes[5].Text, out res) || !Double.TryParse(textBoxes[6].Text, out res)) return "Введены не корректные символы в количестве или цене!";

return null;

}

Описание теста: Неуспешное редактирование товара с некорректными данными.   private string CheckToChangeCar(List<TextBox> textBoxes)

{

if (!CheckId())

{

return "Номер(Id) машины поменялся!";

}

foreach (TextBox textBox in textBoxes)

{

if (textBox.Text.Length == 0) return "Не все поля введены!";

if (!Regex.IsMatch(textBox.Text.ToLower(), @"^[a-zа-яё0-9.\s]+$")) return "Введены не корректные символы в полях!";

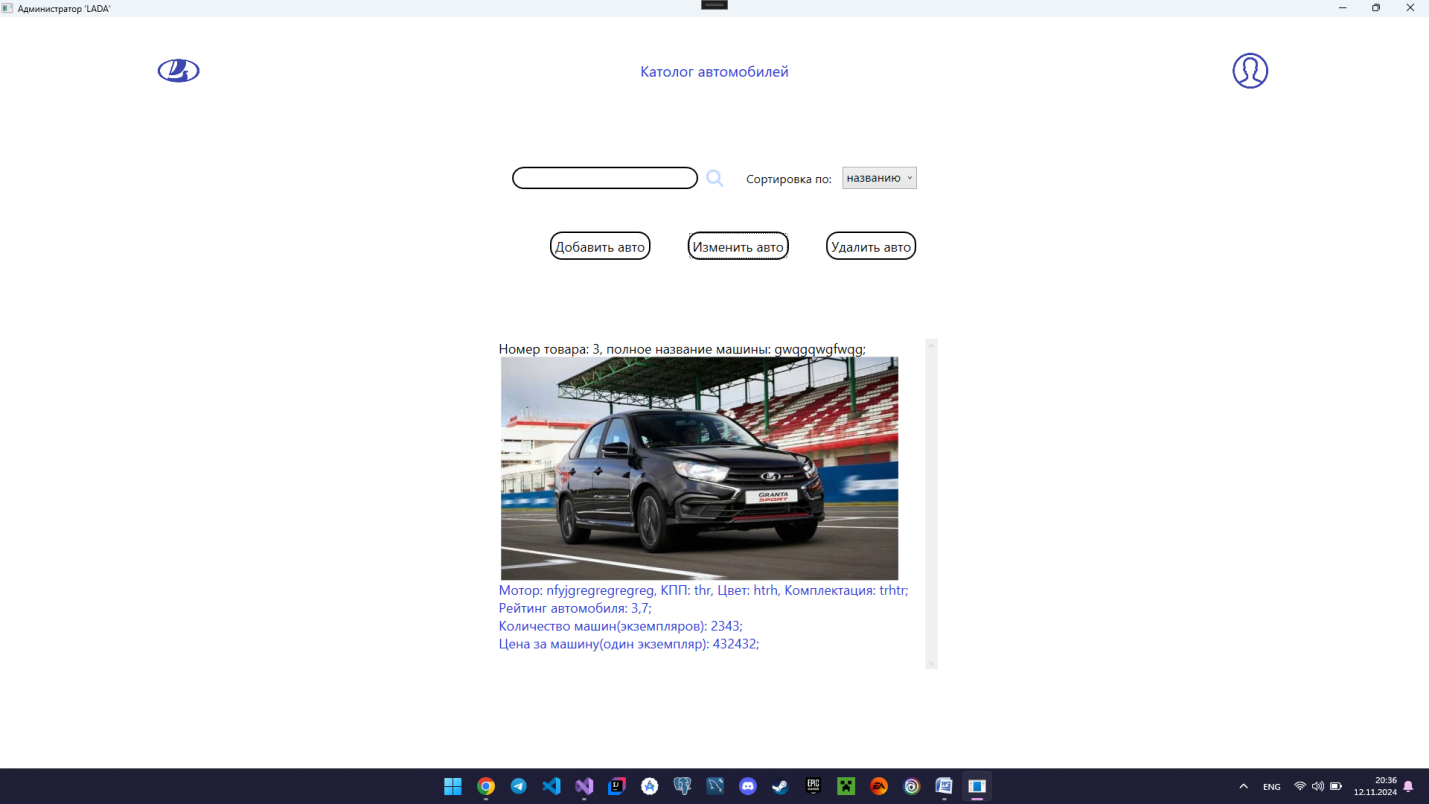
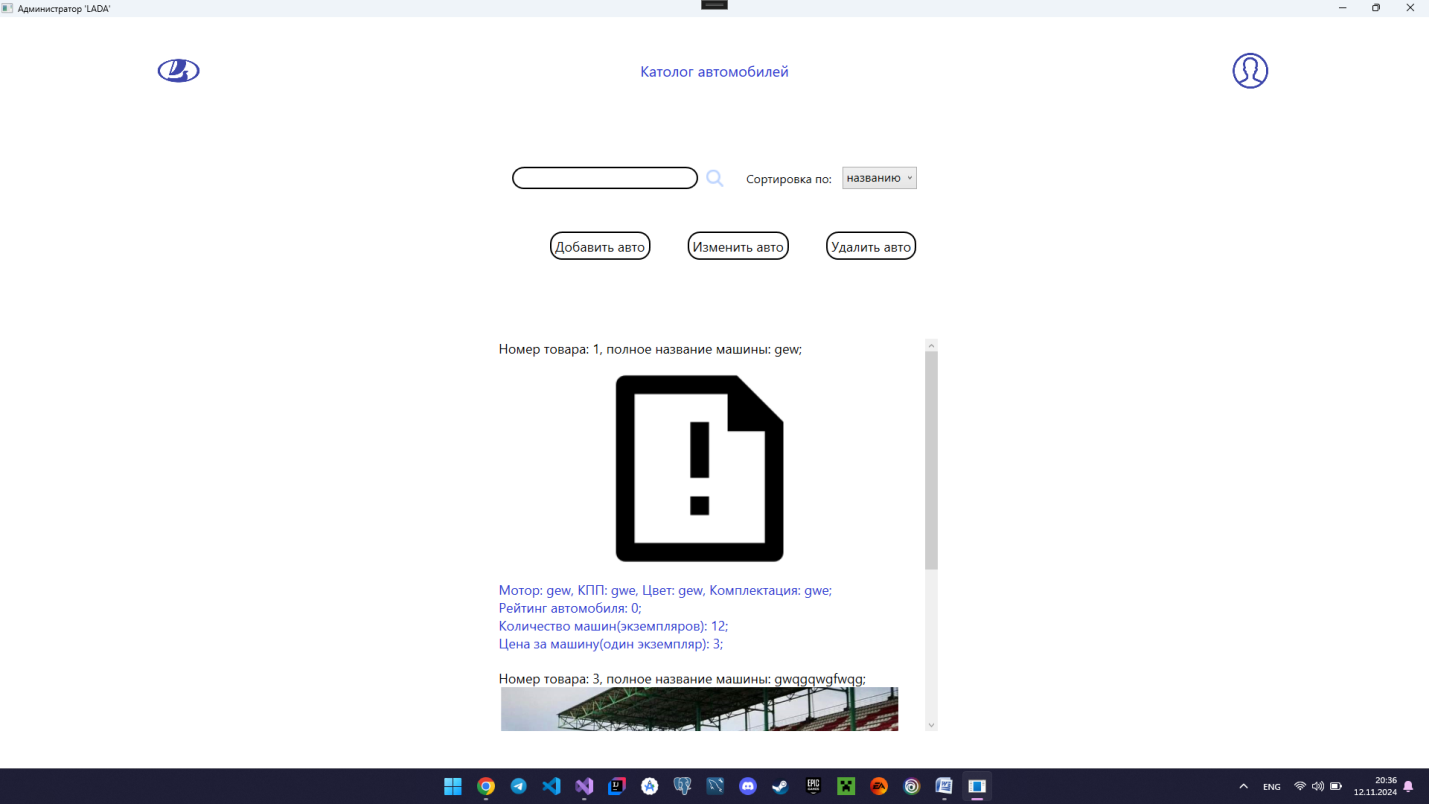
}

double res = 0;

if (!Double.TryParse(textBoxes[5].Text, out res) || !Double.TryParse(textBoxes[6].Text, out res)) return "Введены не корректные символы в количестве или цене!";

return null;

}

Описание теста: Обновление списка товаров после добавления.   private void AddCarsToCard()

{

carsContainer.Items.Clear();

foreach (var car in cars)

{

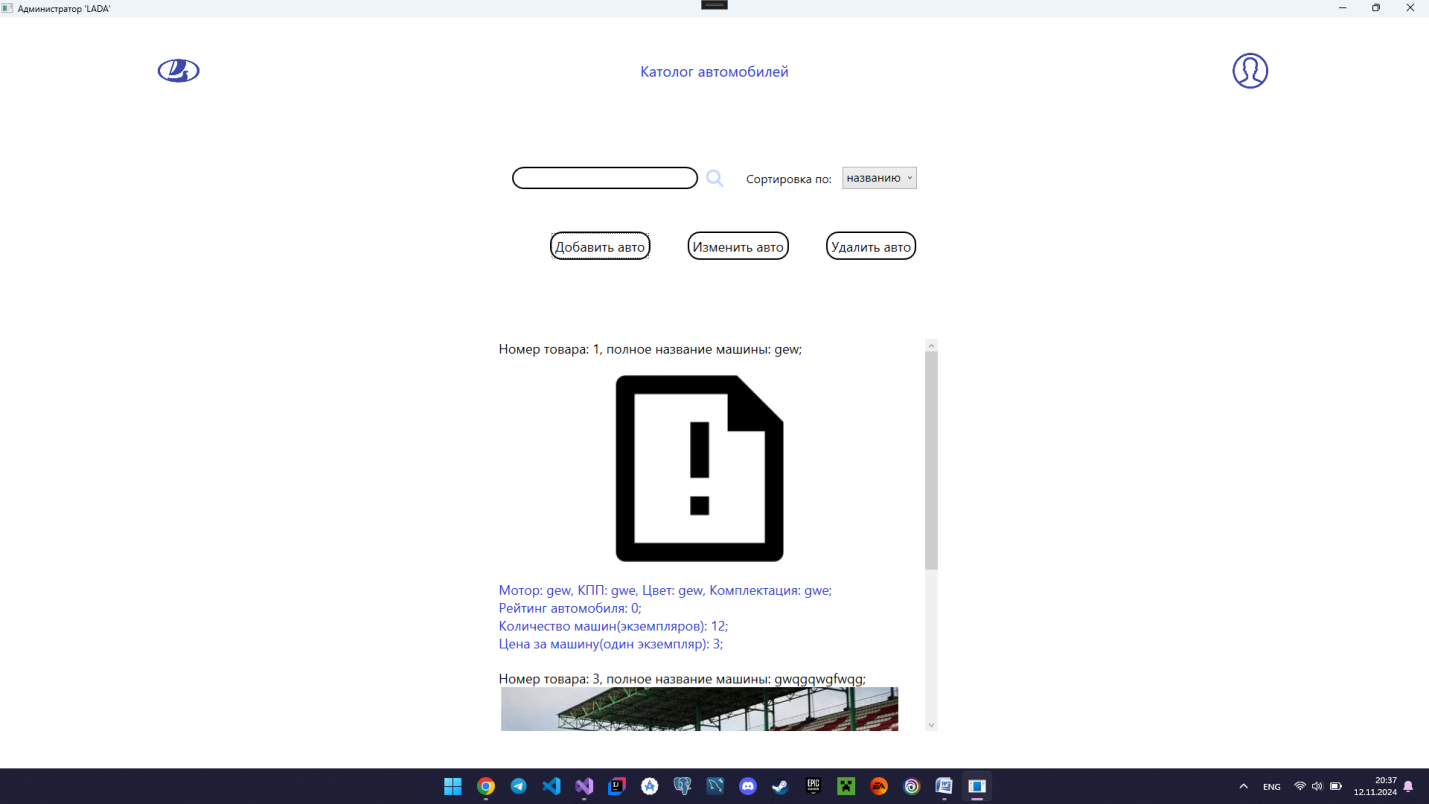
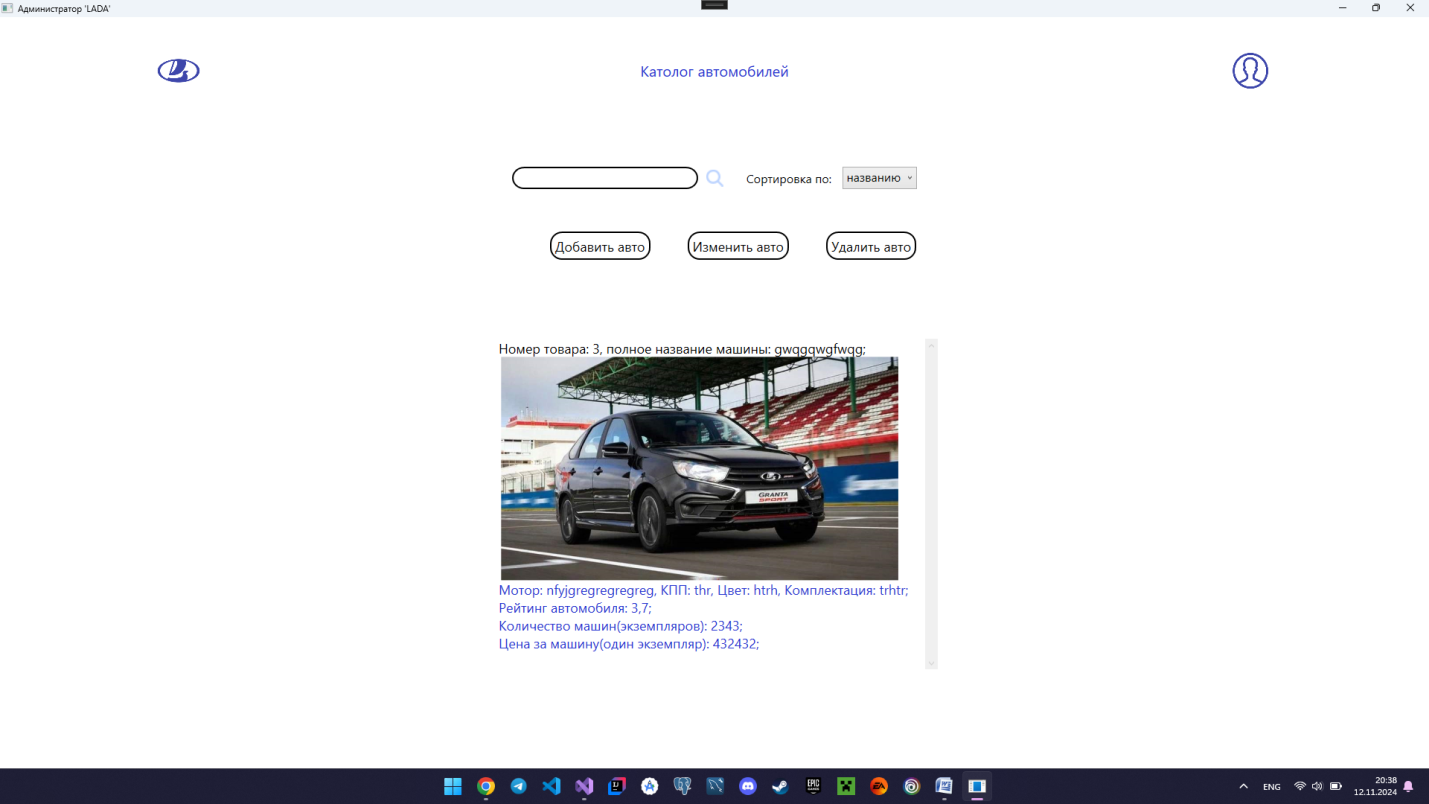
var CarItem = new CarCard();

CarItem.SetCarData(car);

carsContainer.Items.Add(CarItem);

}

}

Описание теста: Обновление списка товаров после удаления.   private void AddCarsToCard()

{

carsContainer.Items.Clear();

foreach (var car in cars)

{

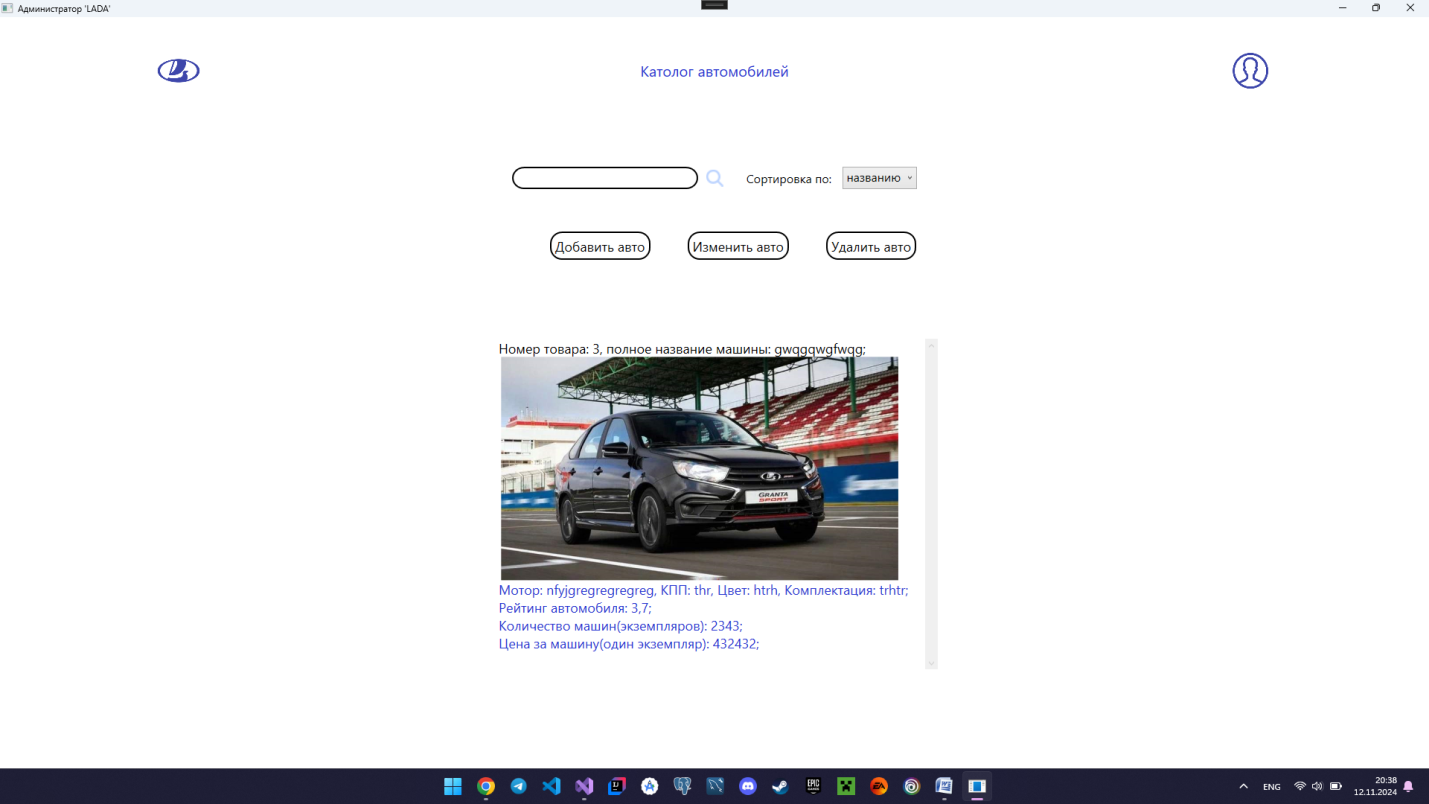
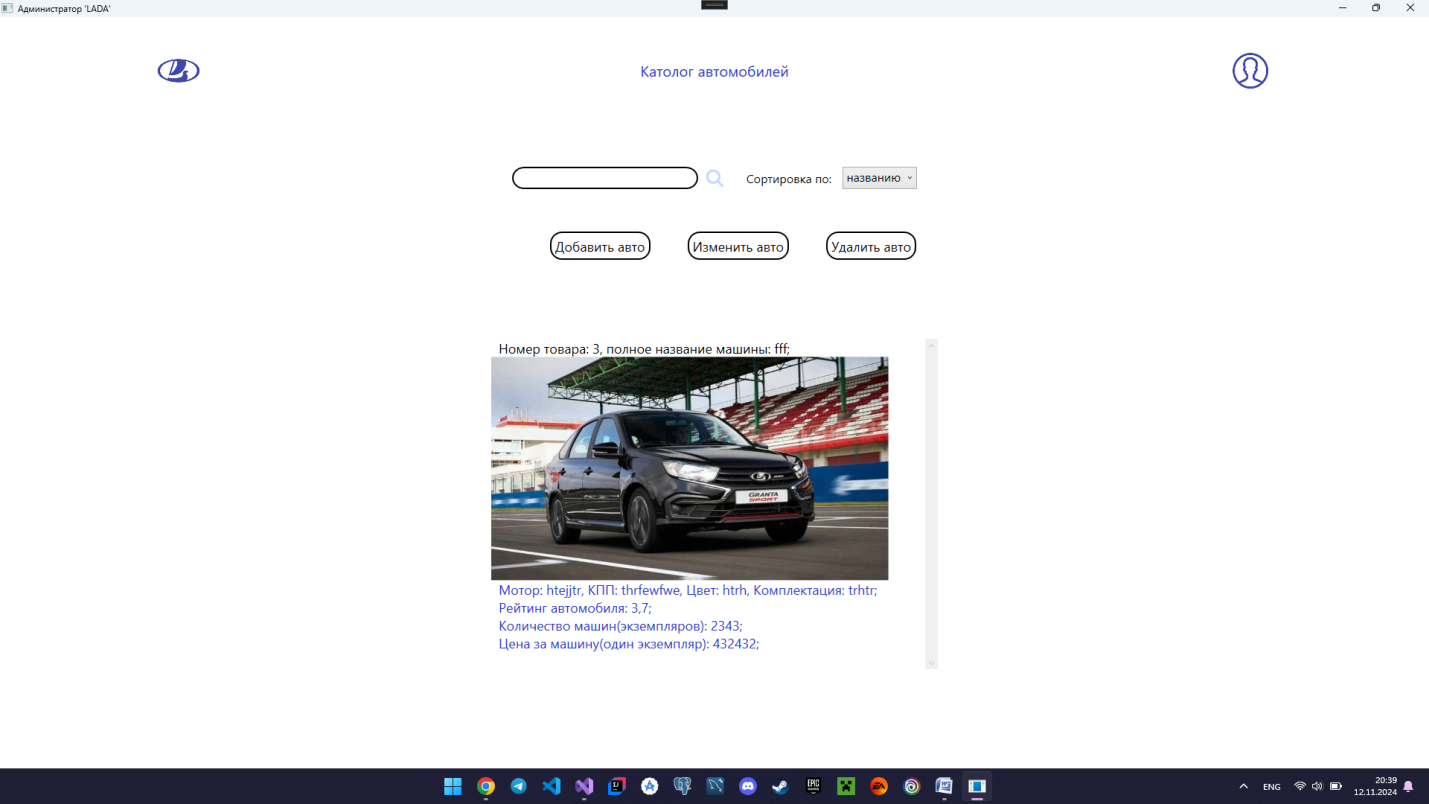
var CarItem = new CarCard();

CarItem.SetCarData(car);

carsContainer.Items.Add(CarItem);

}

}

Описание теста: Обновление списка товаров после редактирования.   private void AddCarsToCard()

{

carsContainer.Items.Clear();

foreach (var car in cars)

{

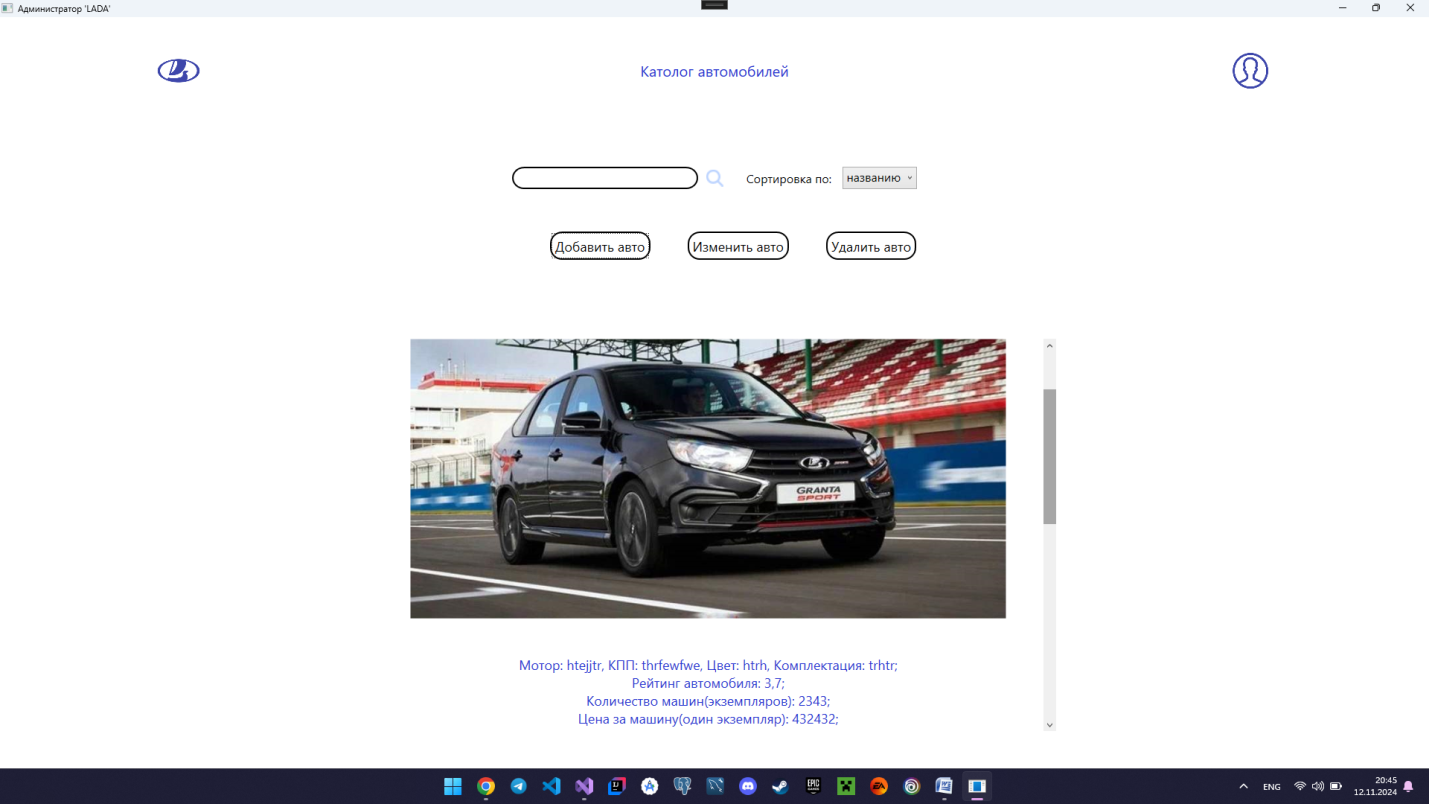
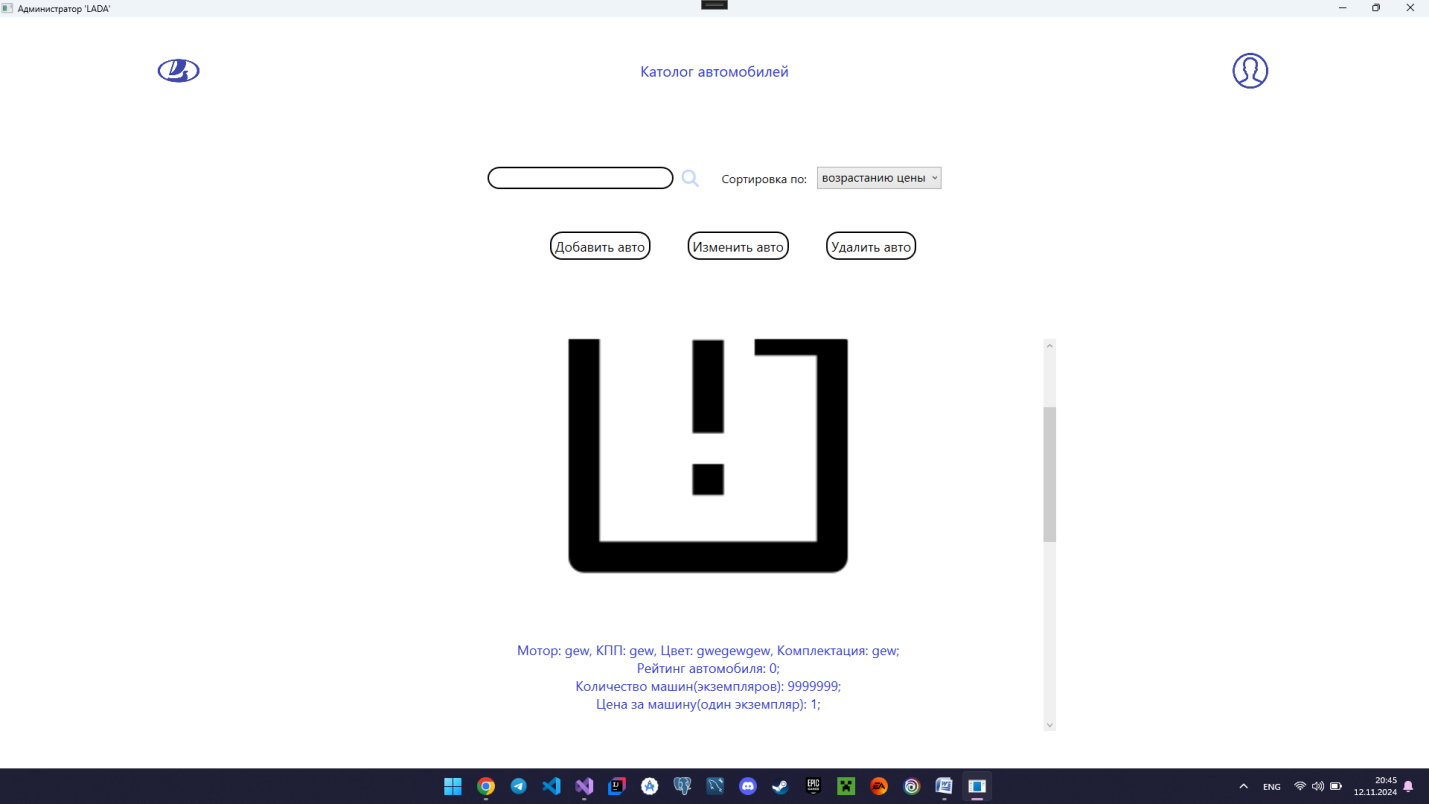
var CarItem = new CarCard();

CarItem.SetCarData(car);

carsContainer.Items.Add(CarItem);

}

}

Описание теста: Сортировка товаров по цене (по возрастанию).   private void SortingByPriceUp()

{

var lst = new List<double>();

var newLst = new List<Car>();

foreach (Car car in cars)

{

lst.Add(car.PricePerCar);

}

lst.Sort();

foreach (double pricePerCarUp in lst)

{

foreach (Car car in cars)

{

if (pricePerCarUp == car.PricePerCar)

{

newLst.Add(car);

break;

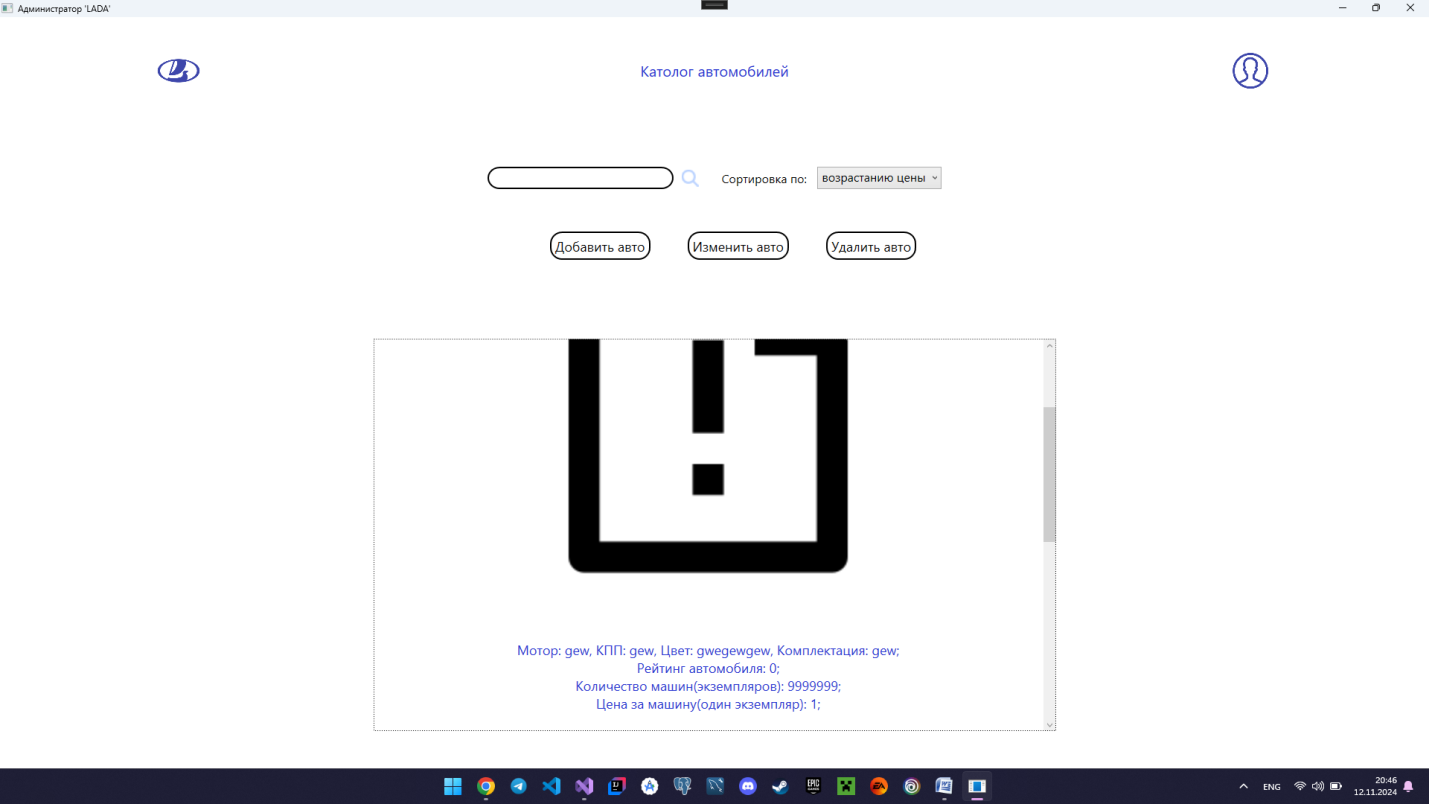
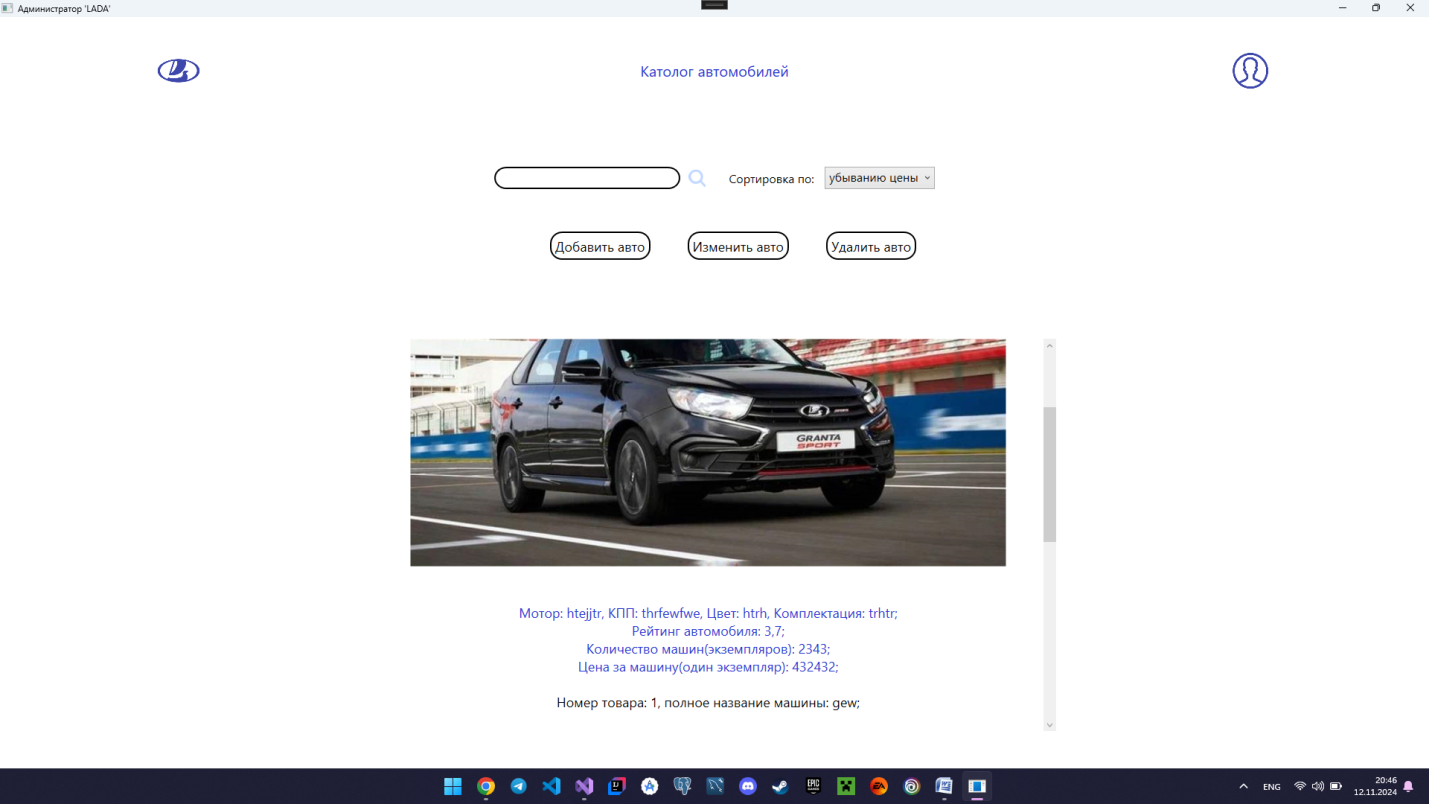
}

}

}

this.cars = newLst;

}

Описание теста: Сортировка товаров по цене (по убыванию).   private void SortingByPriceDown()

{

var lst = new List<double>();

var newLst = new List<Car>();

foreach (Car car in cars)

{

lst.Add(car.PricePerCar);

}

lst.Sort();

lst.Reverse();

foreach (double pricePerCarUp in lst)

{

foreach (Car car in cars)

{

if (pricePerCarUp == car.PricePerCar)

{

newLst.Add(car);

break;

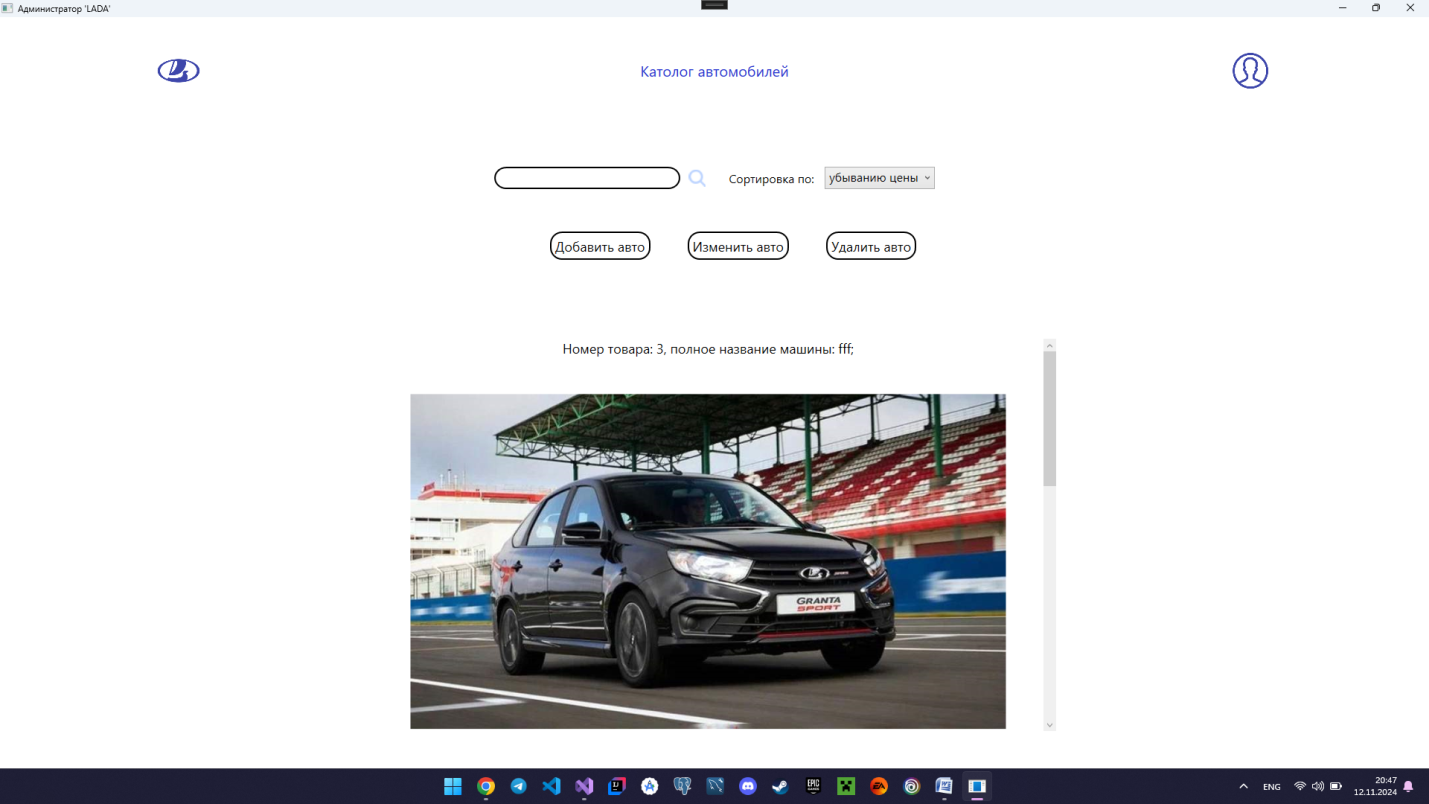
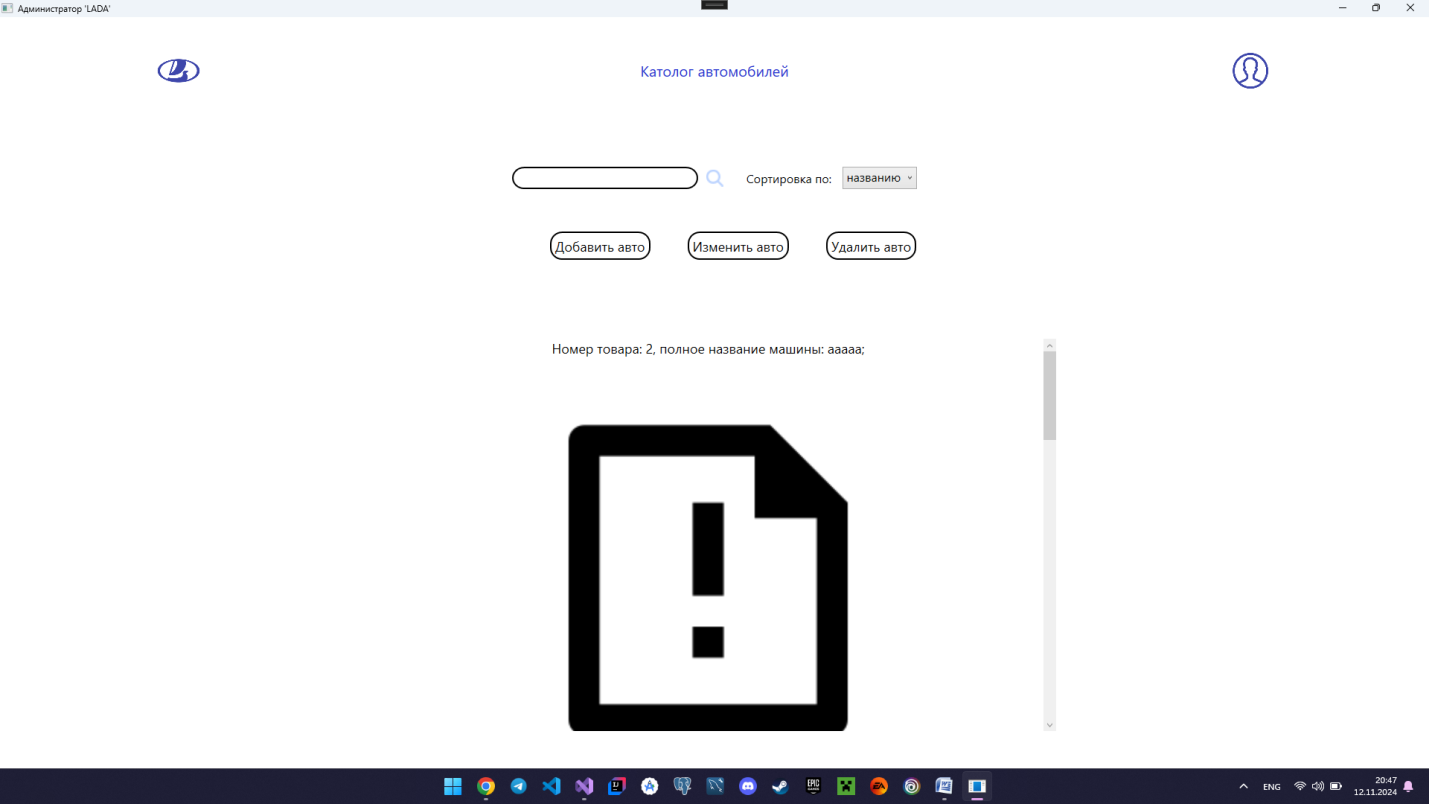
}

}

}

this.cars = newLst;

}

Описание теста: Сортировка товаров по названию.   private void SortingByName()

{

var lst = new List<string>();

var newLst = new List<Car>();

foreach (Car car in cars)

{

lst.Add(car.FullName);

}

lst.Sort();

foreach (string name in lst)

{

foreach(Car car in cars)

{

if (name == car.FullName)

{

newLst.Add(car);

break;

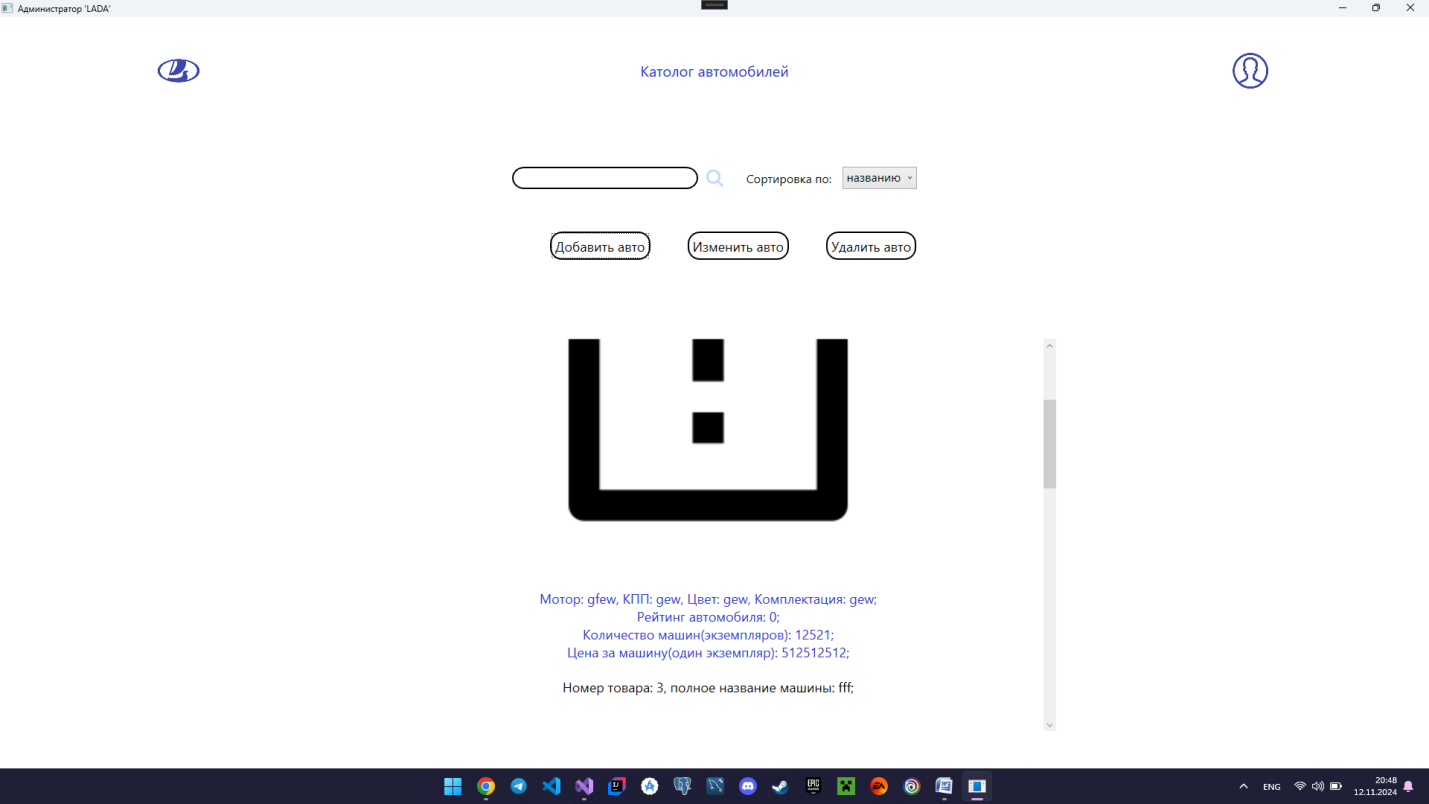
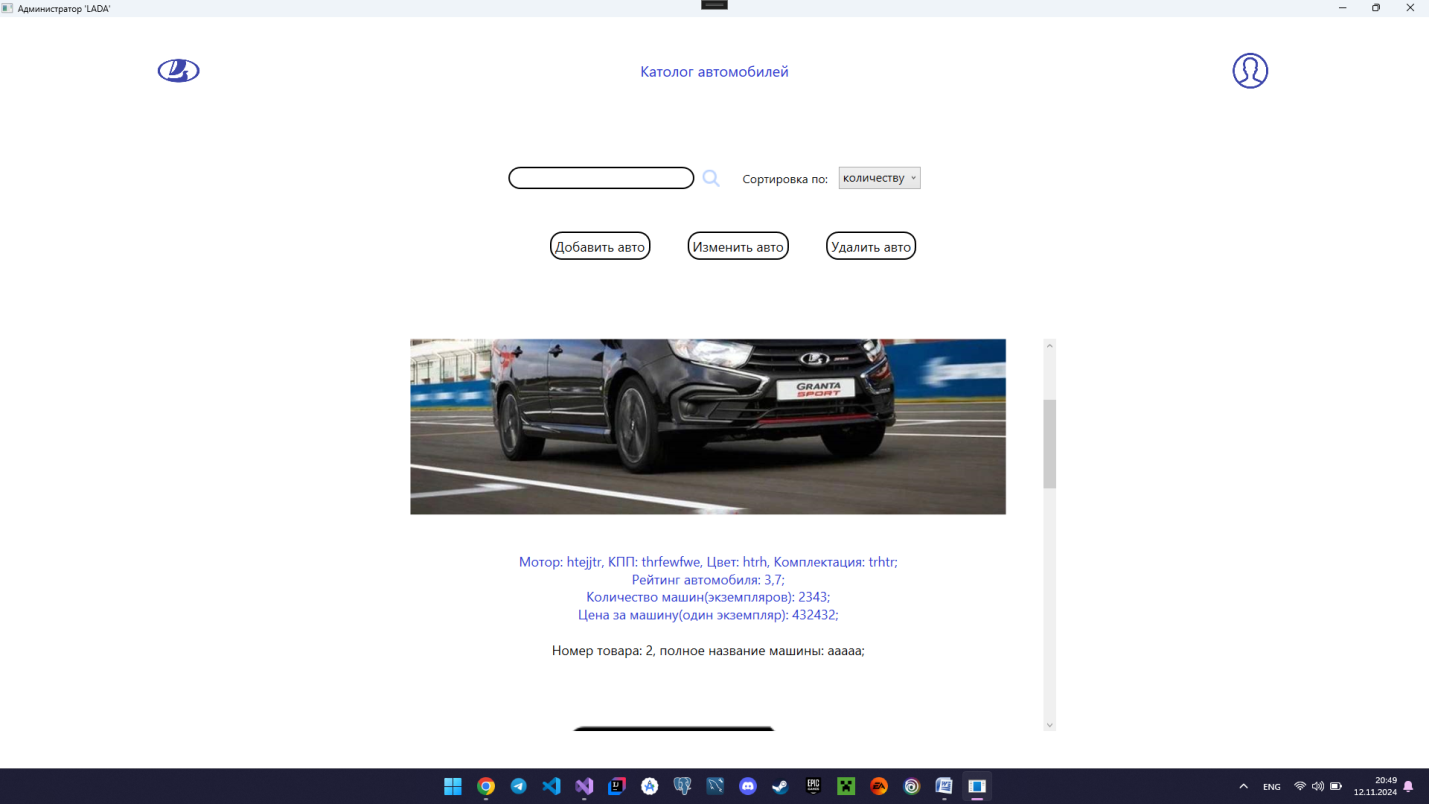
}

}

}

this.cars = newLst;

}

Описание теста: Сортировка товаров по количеству.   private void SortingByAmount()

{

var lst = new List<double>();

var newLst = new List<Car>();

foreach (Car car in cars)

{

lst.Add(car.Amount);

}

lst.Sort();

foreach (double amount in lst)

{

foreach (Car car in cars)

{

if (amount == car.Amount)

{

newLst.Add(car);

break;

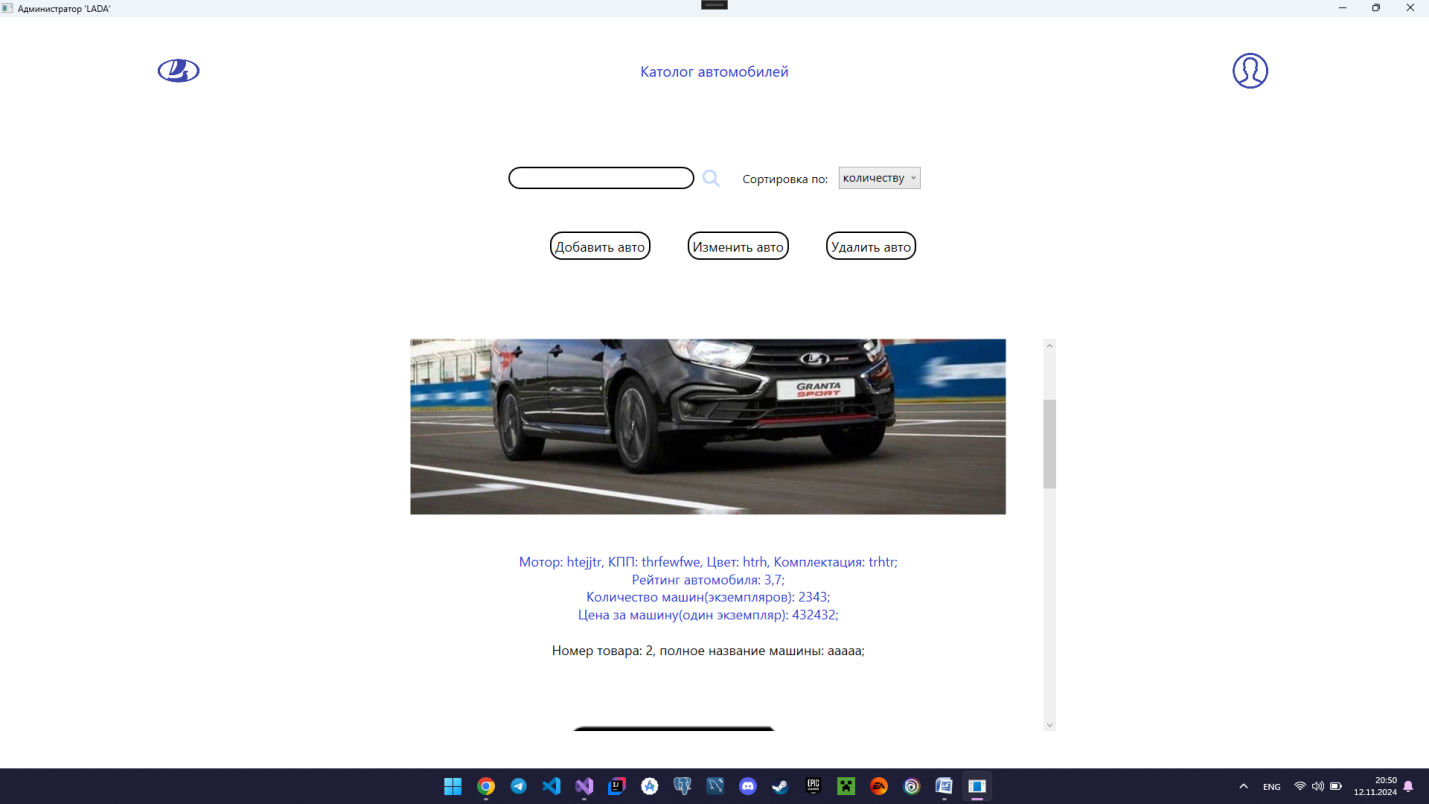
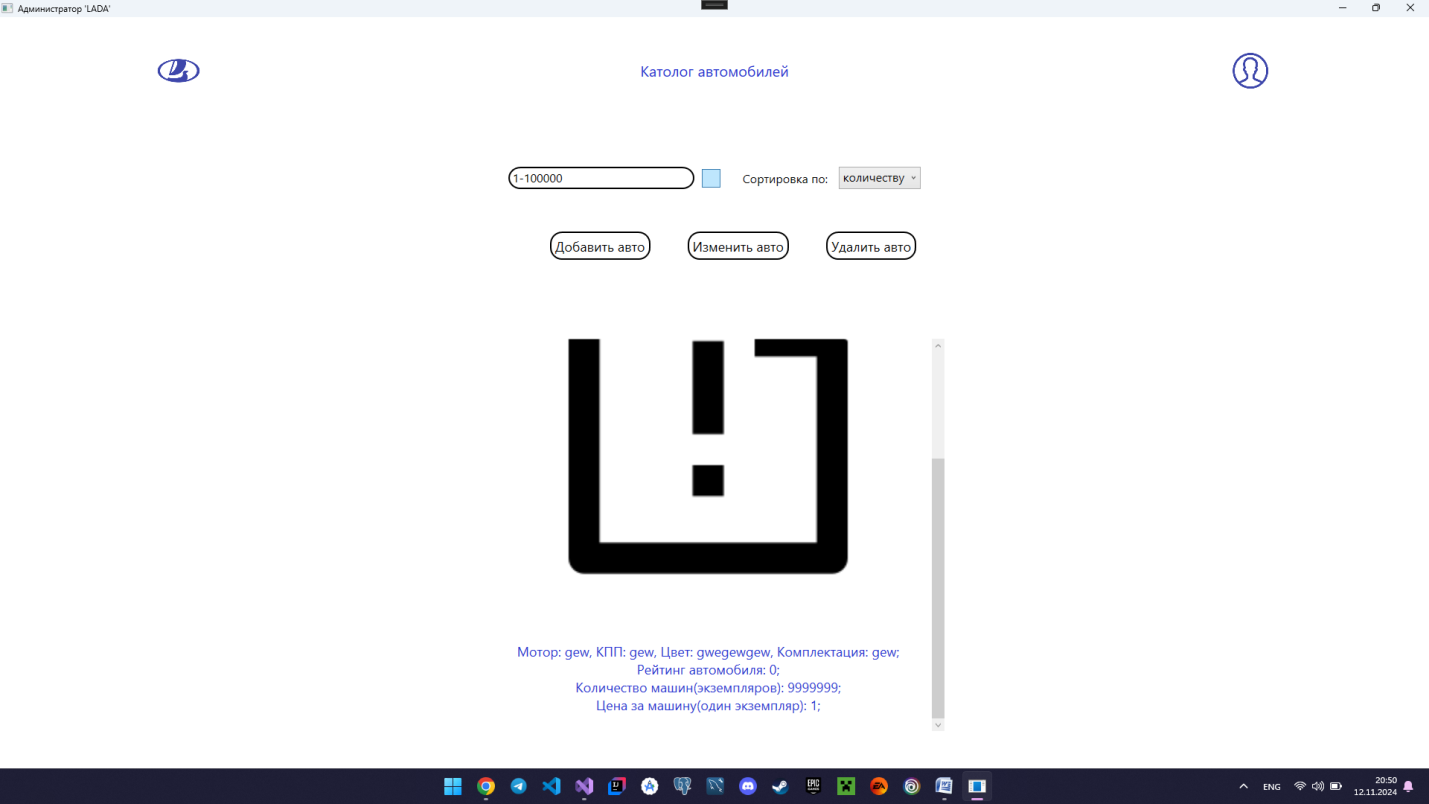
}

}

}

this.cars = newLst;

}

Описание теста: Фильтрация товаров по цене (диапазон).   if (text.Contains("-"))

{

var firstNum = text.Split('-')[0];

var secondNum = text.Split('-')[1];

double res1;

double res2;

if (!double.TryParse(firstNum, out res1) || !double.TryParse(secondNum, out res2))

{

MessageBox.Show("Не правильно введён диапозон!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

newLst = FindOnDiaposonPrice(res1, res2);

}

private List<Car> FindOnDiaposonPrice(double firstNum, double secondNum)

{

var newLst = new List<Car>();

foreach (Car car in this.cars)

{

if (car.PricePerCar >= firstNum && car.PricePerCar <= secondNum)

{

newLst.Add(car);

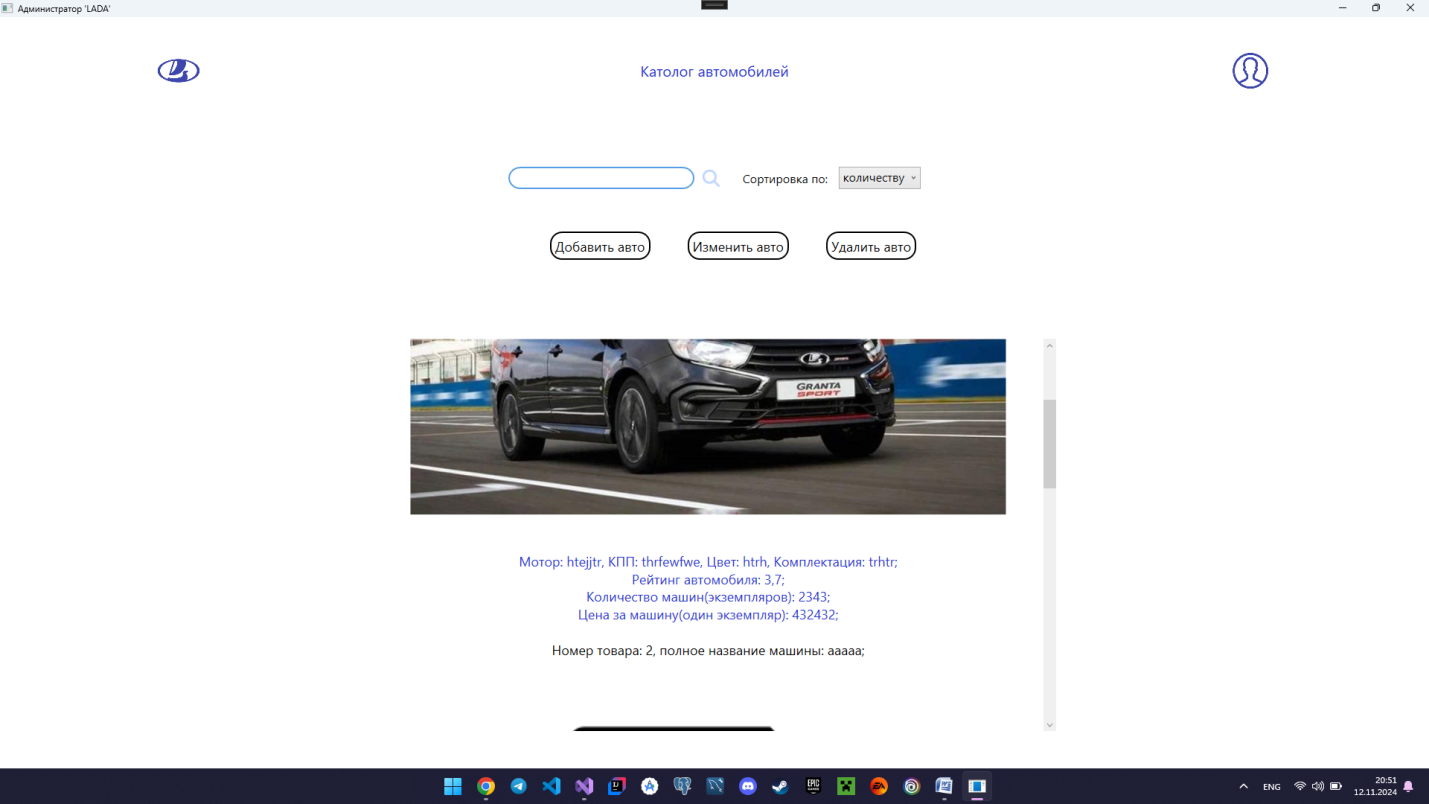
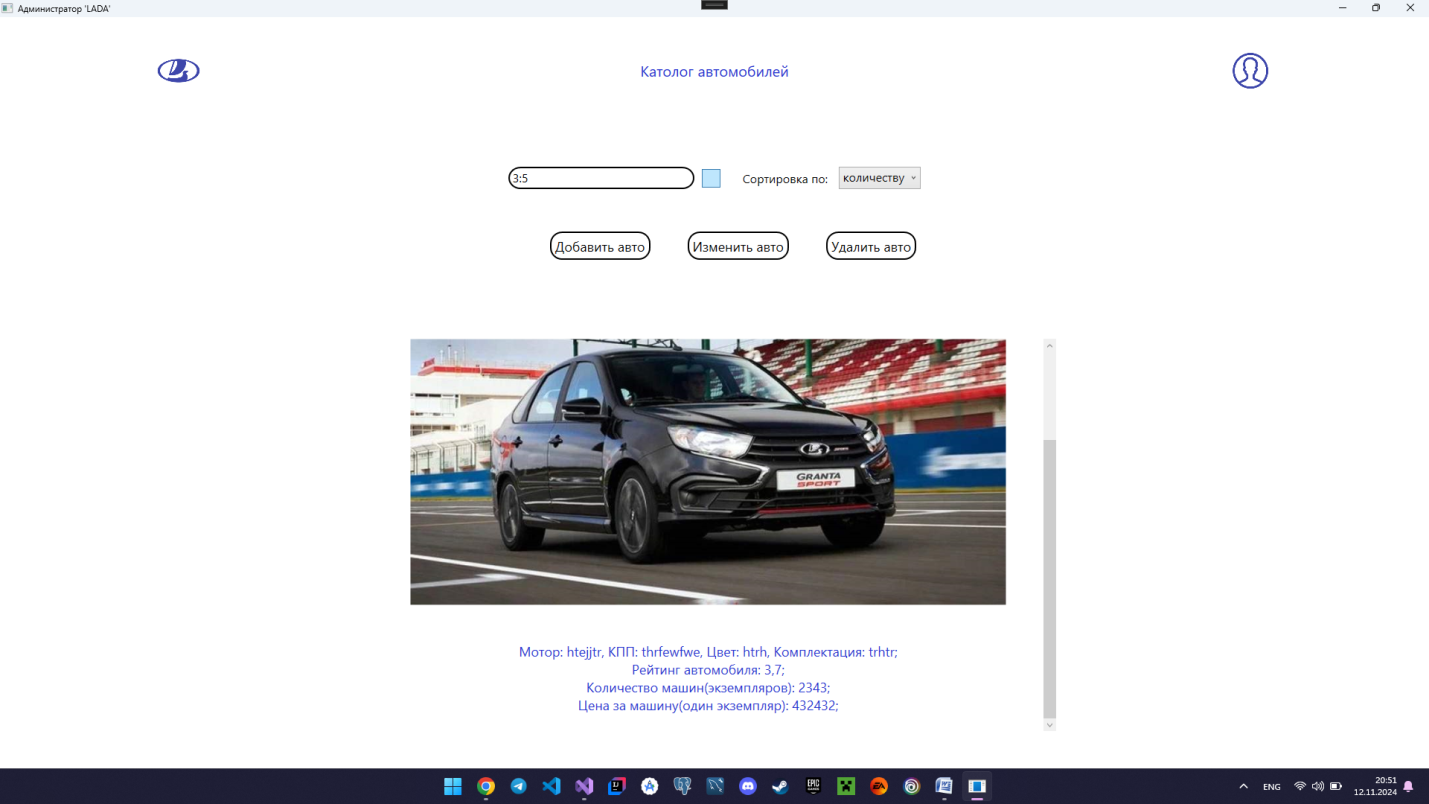
}

}

return newLst;

}

Описание теста: Фильтрация товаров по рейтингу (диапазон).

  else if (text.Contains(":"))

{

var firstNum = text.Split(':')[0];

var secondNum = text.Split(':')[1];

double res1;

double res2;

if (!double.TryParse(firstNum, out res1) || !double.TryParse(secondNum, out res2))

{

MessageBox.Show("Не правильно введён диапозон!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

newLst = FindOnDiaposonRating(res1, res2);

private List<Car> FindOnDiaposonRating(double firstNum, double secondNum)

{

var newLst = new List<Car>();

foreach (Car car in this.cars)

{

double rating = car.CalculationRating();

if (rating >= firstNum && rating <= secondNum)

{

newLst.Add(car);

}

}

return newLst;

}