**UNIVERSITATEA TEHNICĂ ”GHEORGHE ASACHI” IAŞI**

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

Configurator Auto

* Iliescu Ștefan-Adrian 1310A
* Dimitriu Emilian 1310A
* Cristea Adrian 1310A
* Rusu Mitică-Eusebiu 1309B

# Documentul specificațiilor cerințelor software (Software Requirements Specifications, SRS)

1. Introducere

1.1 Scopul

Scopul aplicației noastre este de a ușura decizia cumpărătorului când dorește să achiziționeze o mașină. Utilizatorii pot folosi aplicația pentru a vizualiza modelul de mașină, putând configura marca, culoare și jantele mașinii.

1.2 Referințe

* Ingineria programării - Laborator 3 (Faza de analiză)
* Ingineria programării - Laborator 4 (Documentarea unui proiect. Fișiere de ajutor)
* Ingineria programării - Laborator 5 (Diagrame UML)
* Ingineria programării - Laborator 6 (Arhitectura Model-View-Controller)
* Ingineria programării - Laborator 12 (Testare unitara)

1.3 Prezentare generală

Aplicația dezvoltată în cadrul proiectului este menită în a ușura alegerea persoanelor ce doresc să își comande o mașină online. Aceștia o pot modifica după bunul plac astfel încât să se hotărască ce model de mașină, ce culoare, cât și modelul de jante ale acestora. Programul este unul simplist, pe înțelesul tuturor utilizatorilor.

2. Descrierea programului

2.1 Perspectiva produsului

Configuratorul Auto creat reprezintă ajutorul atât al cumpărătorilor cât și al vânzătorilor de mașini.

2.2 Funcționalități ale produsului

* Selectarea mărcii de mașină
* Selectarea jantelor
* Selectarea culorii
* Comandarea unei mașini
* Afișarea comenzii căutate
* Istoricul comenzilor

2.3 Caracterici ale utilizatorului

Aplicația poate fi utilizată de către oricine.

2.4 Constrângeri generale

Aplicația poate rula pe sistemul de operare Windows.

2.5 Ipoteze și dependențe

Aplicația poate rula pe sistemul de operare Windows și are nevoie de Visual Studio.

3. Cerințe specifice

3.1 Cerințele interfeței externe

3.1.1 Interfețe cu utilizatorul

Interfața este realizată cu Windows Forms App (.NET Framework).

3.1.2 Interfețe Hardware

Aplicația rulează pe orice CPU.

3.1.3 Interfețe Software

Sistemul de operare folosit este Windows deoarece este foarte practic, de aceea este și mai utilizat. Ca mediu de dezvoltare am folosit Visual Studio 2019 Community, limbajul folosit fiind C#.

3.2 Cerințe funcționale

După alegerea tipului de mașină din interfață, acesta poate alege jantele și culoarea după care poate comanda produsul final pe care și-l dorește. Afișează mașina, culoarea și jantele și comanda înregistrată.

3.3 Cerințe de performanță

3.3.1 Securitate

Nu au fost utilizate logările prin nume și parolă.

3.3.2 Performanță

Este o aplicație simplistă, nu necesită cerințe ale calculatorului mari. Aceasta funcționează pe orice computer cu sistem de operare Windows.

3.4 Constrângeri de proiectare

Pentru că interfața a fost realizată cu Windows Forms App (.NET Framework), aceasta este una simplistă dar care conține suficiente detalii astfel încât poate să ofere utilizatorului acces la toate operațiile de care are nevoie și ocupă un spațiu de 18 MB.

3.5 Atribute

3.5.1 Valabilitate

Produsele adaugate in fișiere pot fi accesate de la o sesiune la alta.

3.5.2 Mentenanță

Aplicația este ușor de extins. Codul oferă posibilitatea implementării unor noi funcții prin derivare.

3.5.3 Corectitudine

Aplicația oferă date corecte către utilizator. În funcție de operația dorită, va primi în interfață informații despre rezultatul așteptat.

Testări:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metoda testată** | **Parametri** | **Rezultat așteptat** | **Rezultat obținut** | **Stare test** |
| Car.GetOptions | new Car(new List<string> { "blue", "17" }) | { "blue", "17" } | { "blue", "17" } | Passed |
| MainForm.onlyNumbers | "123" | true | true | Passed |
| MainForm.onlyNumbers | "1a3" | false | false | Passed |
| Compare | CarFactory.makeCar("Audi",{ "blue", "17" }), new Audi(",{ "blue", "17" }) | true | true | Passed |
| CarFactory.makeCar | "Fiat",{ "blue", "17" } | exc | exc | Passed |
| Car.GetOrderNumber | { "blue", "17" } | true | true | Passed |

Diagramele UML

Diagrama de clase

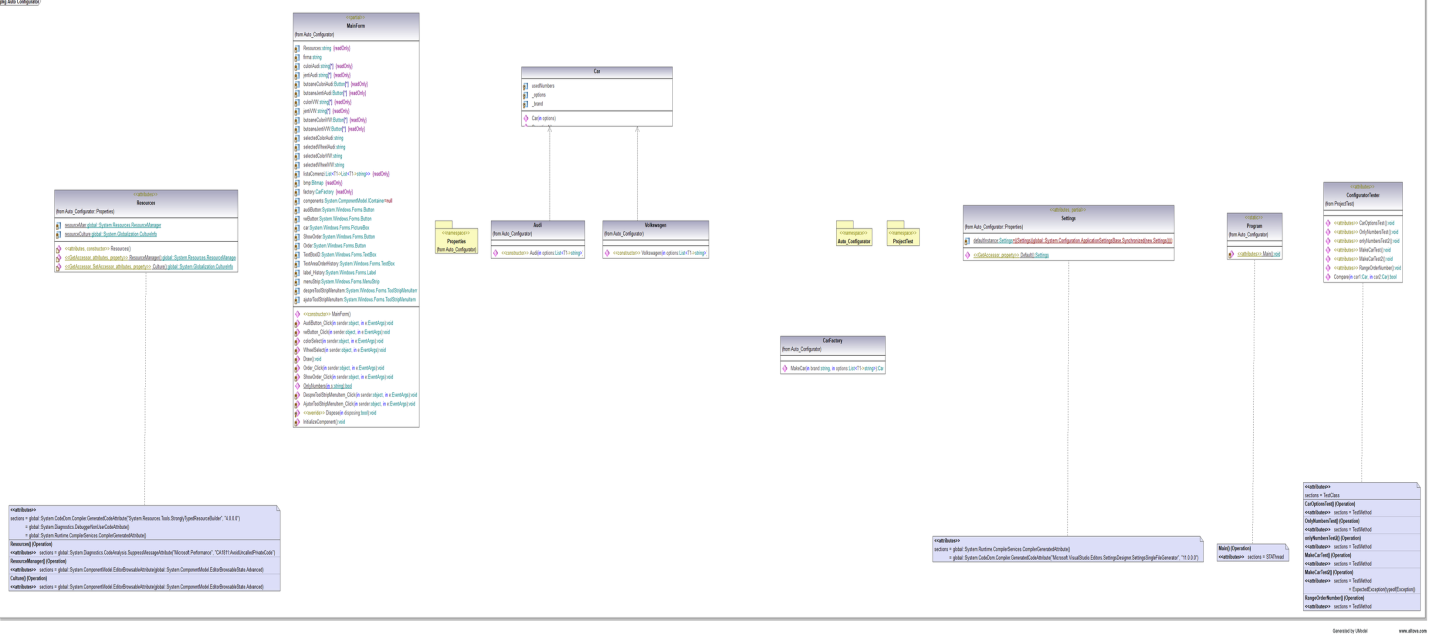


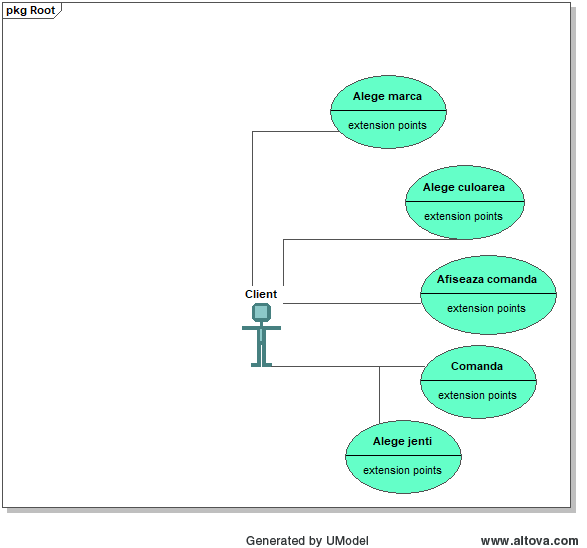
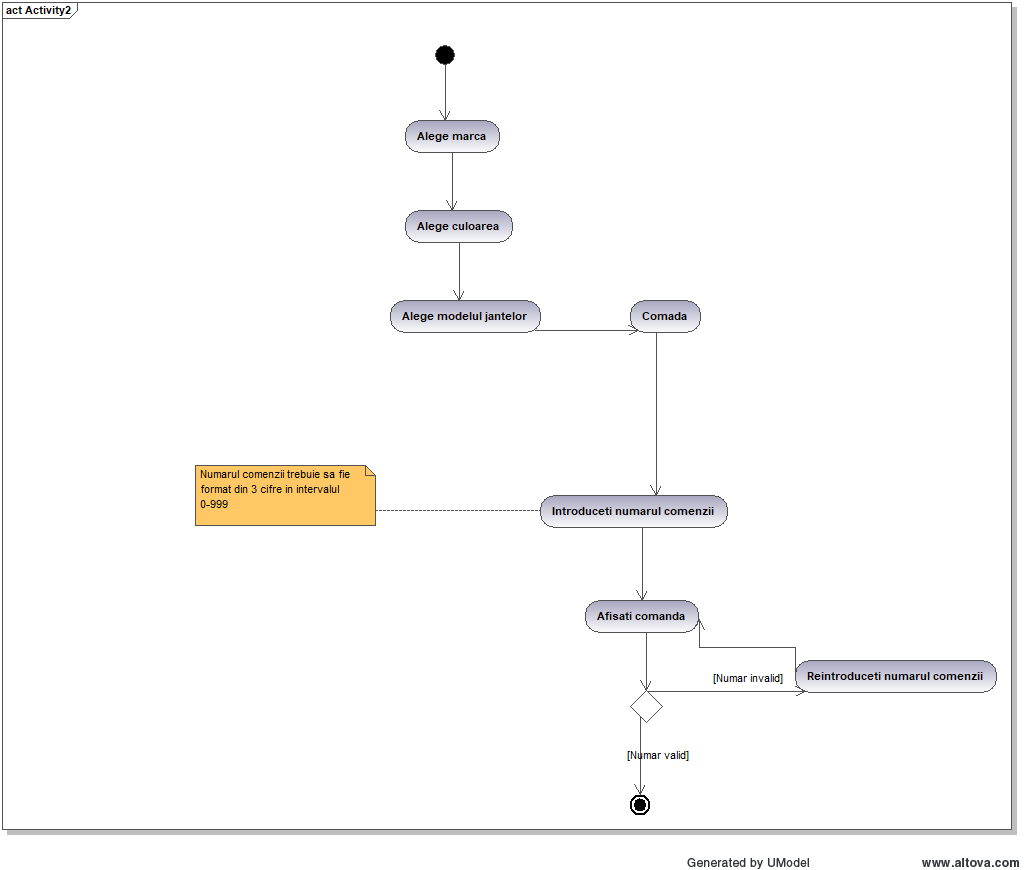
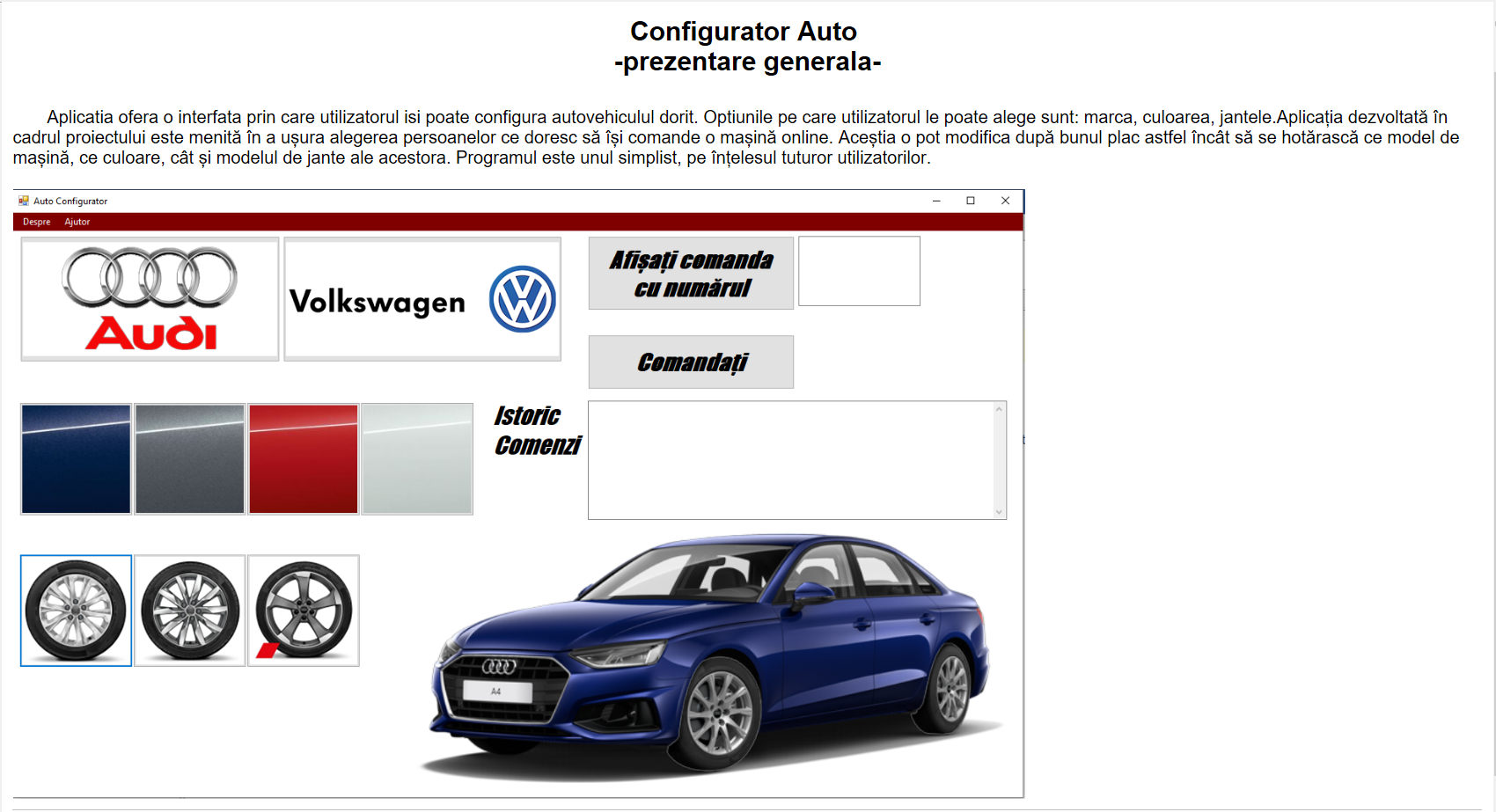
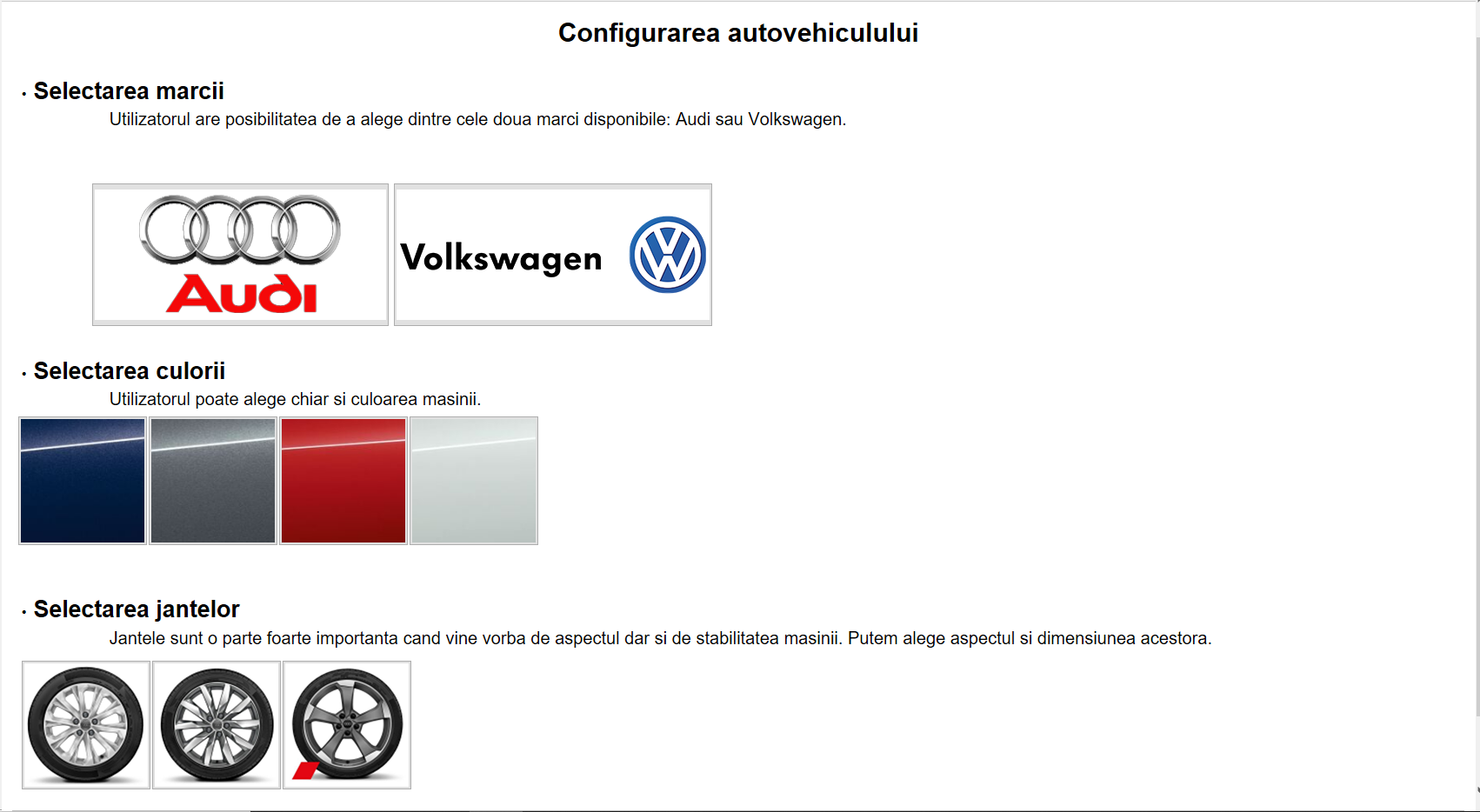
Diagrama Cazurilor de utilizare 

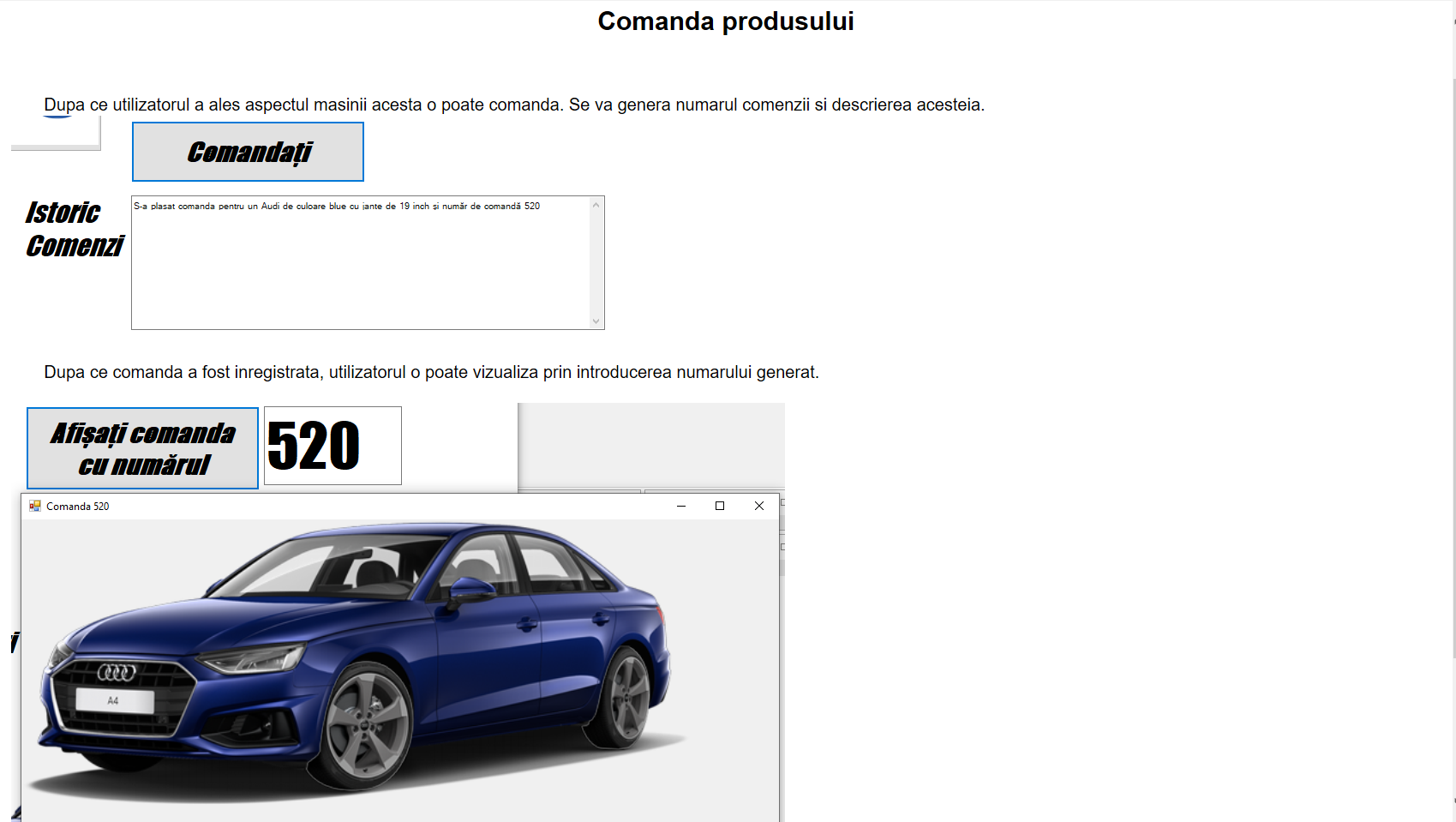
Diagrama de activități



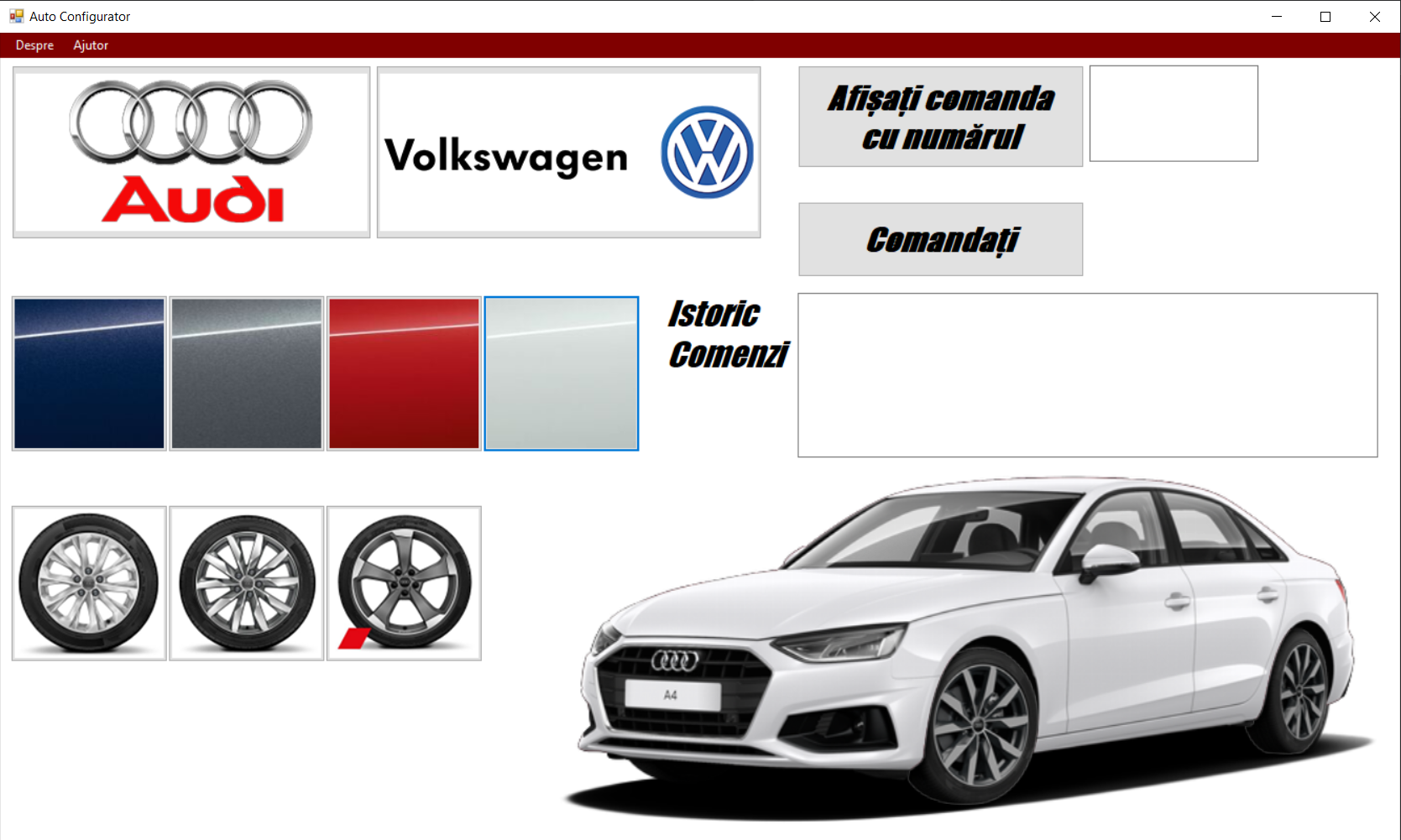
Modul de utilizare ( bazat pe conținutul help-ului):

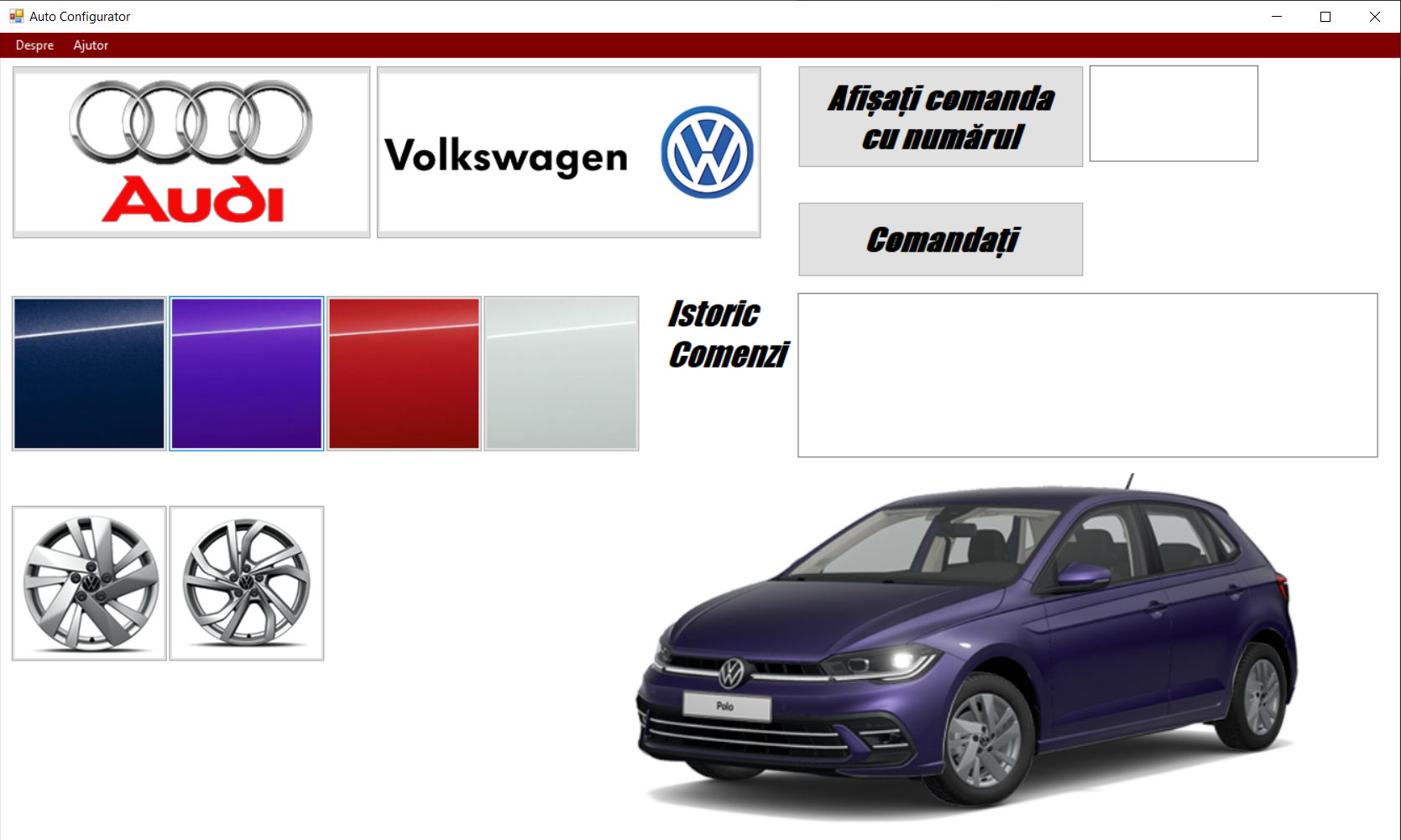




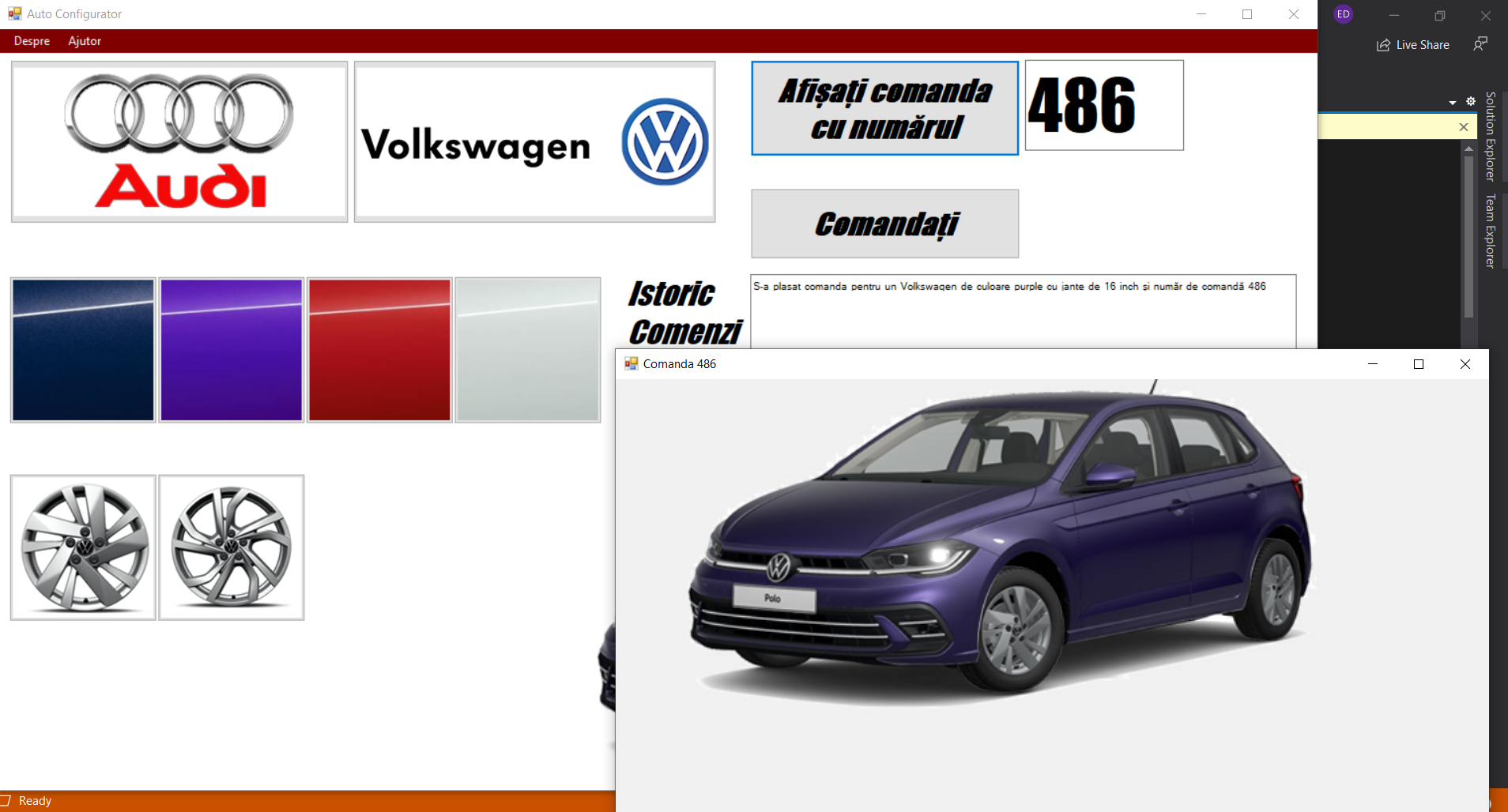


Capturi de ecran









Anexă:

Configurator Tester.cs

namespace ProjectTest

{

[TestClass]

public class ConfiguratorTester

{

/// <summary>

/// Metoda de testare a initierii optiunilor

/// </summary>

[TestMethod]

public void CarOptionsTest()

{

CollectionAssert.AreEqual(new List<string> { "blue", "17" }, new Car(new List<string> { "blue", "17" }).GetOptions());

}

/// <summary>

/// Metoda de testare pentru a verifica daca numarul de comanda este format doar din cifre

/// </summary>

[TestMethod]

public void OnlyNumbersTest()

{

Assert.AreEqual(true, MainForm.OnlyNumbers("123"));

}

/// <summary>

/// Metoda a doua de testare pentru a verifica daca numarul de comanda este format doar din cifre

/// </summary>

[TestMethod]

public void onlyNumbersTest2()

{

Assert.AreEqual(false, MainForm.OnlyNumbers("1a3"));

}

/// <summary>

/// Metoda de testare pentru crearea unui obiect de tip Car

/// </summary>

[TestMethod]

public void MakeCarTest()

{

List<string> options = new List<string> { "blue", "17" };

CarFactory cf = new CarFactory();

Car c = new Audi(options);

Assert.AreEqual(true, Compare(cf.MakeCar("Audi", options),c));

}

/// <summary>

/// Metoda de testare pentru crearea unui obiect de tip Car

/// </summary>

[TestMethod]

[ExpectedException(typeof(Exception))]

public void MakeCarTest2()

{

List<string> options = new List<string> { "blue", "17" };

CarFactory cf = new CarFactory();

\_ = cf.MakeCar("Fiat", options);

}

/// <summary>

/// Metoda de testare pentru generarea numerelor de comanda

/// </summary>

[TestMethod]

public void RangeOrderNumber()

{

Car car = new Car(new List<string> { "blue", "17" });

car.SetOrderNumber();

Assert.AreEqual(true, Enumerable.Range(100,900).Contains(car.GetOrderNumber()));

}

/// <summary>

/// Metoda de testare care compara crearea a doua obiecte

/// </summary>

/// <param name="car1">Obiect 1 tip Car</param>

/// <param name="car2">Obiect 2 tip Car</param>

/// <returns>Valoarea de adevar a operatiei de comparare</returns>

public bool Compare(Car car1,Car car2)

{

return car1.Brand.Equals(car2.Brand) && car1.GetOptions().SequenceEqual(car2.GetOptions());

}

}

}

Car.cs

namespace Auto\_Configurator

{

/// <summary>

/// Clasa de baza a masinilor

/// </summary>

public class Car

{

private static List<int> usedNumbers;//numere utilizate

private List<string> \_options;//proprietatile

private string \_brand; //marca masinii

/// <summary>

/// Getter pentru numele marcii

/// </summary>

/// <returns>Numele marcii</returns>

public string Brand => \_brand;

private int orderNumber;//numar de ordin

/// <summary>

/// Constructorul pentru clasa de baza Car

/// </summary>

/// <param name="options"></param>

public Car(List<string> options)

{

\_options = options;

usedNumbers = new List<int>(100);

}

/// <summary>

/// Getterul pentru optiuni

/// </summary>

/// <returns>Optiunile masinii</returns>

public List<string> GetOptions()

{

return \_options;

}

/// <summary>

/// Setterul pentru brandul masinii

/// </summary>

/// <param name="brand"></param>

public void SetBrand(string brand)

{

\_brand = brand;

}

/// <summary>

/// Seteaza numarul de ordin

/// </summary>

public void SetOrderNumber()

{

int random = (int)(100 + new Random().NextDouble() \* 900);//intre 100 si 999

if (usedNumbers.Contains(random))//verificam daca se afla in lista de numere utilizate

{

if (usedNumbers.Count == 100)//maximul de comenzi

MessageBox.Show("S-a ajuns la un numar limitat de comenzi");

SetOrderNumber();//apelam recursiv si poate primim numar care nu se afla in lista

}

else

{

orderNumber = random;

usedNumbers.Add(random);//adaugam numarul nou in lista;

}

}

/// <summary>

/// Getterul pentru numarul de ordin

/// </summary>

/// <returns>Numarul de ordin</returns>

public int GetOrderNumber()

{

return orderNumber;

}

}

}

* Iliescu Ștefan-Adrian 1310A - cod( interfață, funcționalitate)
* Dimitriu Emilian 1310A – documentație, SRS
* Cristea Adrian 1310A – cod ( testări )
* Rusu Mitică-Eusebiu 1309B – diagrame, doxygen