UNIVERSITATEA TEHNICĂ "GHEORGHE ASACHI" IAȘI
FACULTATEA AUTOMATICĂ ȘI CALCULATOARE
SPECIALIZAREA CALCULATOAREA ȘI TEHNOLOGIA INFORMAȚIEI
DISCIPLINA INGINERIA PROGRAMARII

Aplicație de gestiune a unei reprezentanțe auto

~Documentație~

Profesor: Dumitriu Tiberius

Studenți:

- **Chiper Ioan-Alexandru**
- **Dima Raul Andrei**
- **Melinte Alexandru-Gicu**
- **Tizu Matei-Victor**

Grupa: 1306A

Cuprins

I. Software Requirements Specification 1
1. Introducere 1
1.1 Obiectivul 1
1.2 Standardele documentatiei 1
1.3 Audiența țintă 1
1.4 Scopul produsului 1
2. Descriere generală 1
2.1 Perspectiva produsului 1
2.2 Funcțiile produsului 2
2.3 Clase de utilizatori și caracteristici 2
2.4 Mediu de operare 2
2.5 Constrângeri de design implementare 2
2.6 Documentația utilizatorului 2
2.7 Ipoteze si dependențe 2
3. Cerinte externe pentru interfață 3
3.1 Interfața utilizatorului 3
3.2 Interfața hardware 3
3.3 Interfața software 3
3.4 Interfața comunicației 3
4. Caracteristicile sistemului 3
5. Alte cerințe 4
5.1 Cerințe pentru performanță 4
5.2 Cerințe pentru siguranță 5
5.3 Cerințe pentru securitate 5
5.4 Calitățile Soft-ului 5
5.5 Reguli de business 5
6. Anexe 5
II. Diagrame UML 9
III. Mod de utilizare 11

I. Software Requirements Specification

1.Introducere

1.1 Obiectivul

Obiectivul echipei noastre este acela de a crea o aplicație pentru a facilita gestiunea unei reprezentanțe auto, utilizând limbajul de programare C#, IDE-ul Microsoft Visual Studio, baze de date Oracle SQL, iar acel produs software să aibă cerințe de sistem minime.

1.2 Standardele documentației

Acest document urmează formatul standard IEEE pentru dezvoltare software. Standardele urmăresc să definească un format ușor de citit, rapid si cu informațiile importante, cât și detaliile din proiectul nostru.

1.3 Audiența țintă

Acest document este destinat persoanelor care dețin cunoștințe în programare și doresc să contribuie la dezvoltarea/mentenanța programului deoarece conține detalii legate de implementarea software a aplicației.

1.4 Scopul produsului

Scopul aplicației este acela de a facilita/ușura gestiunea unei reprezentanțe auto prin conceperea unei aplicații care să asigure fiabilitate, portabilitate si un cost minim de implementare.

Aplicația va funcționa fară conexiune la internet, va fi conectată la o bază de date Oracle SQL și va fi utilizată la operațiile de bază care se realizează în interiorul unui showroom auto: vânzare autoturism, generare facturi, gestiune bază de date clienți.

2. Descriere generală

2.1 Perspectiva produsului

Aplicația dezvoltată de noi nu este unică pe piață dar îmbină niște elemente ce ușurează semnificativ munca utilizatorului.

2.2 Funcțiile produsului

Funcțiile pe care aplicația trebuie să le îndeplinească sunt:

- Aplicația trebuie să fie portabilă, adică să ruleze cu success și pe sisteme cu specificații slabe;
- Aplicația trebuie să ofere posibilitatea de a stoca mașinile vândute;
- Aplicația trebuie să fie capabilă să genereze facturile aferente unei tranzacții;
- Este necesar ca produsul software să înregistreze detalii legate atât de garanție, cât și de servisare/mententanță;
- Aplicația să fie stabilă: să asigure coerența datelor și să nu genereze întreruperi cauzate de potențiale erori de proiectare.

2.3 Clase de utilizatori și caracteristici

Aplicația acceptă două tipuri de utilizatori: user normal (agent de vânzări) și admin. Un user este acel angajat care încheie contractele și generează facturile, iar administratorul este cel ce are dreptul de a adăuga diferite opțiuni în baza de date.

2.4 Mediul de operare

Deoarece rularea programului depinde de IDE-ul Microsoft Visual Studio, IDE disponibil pe majoritatea sistemelor de operare, aplicația mai sus descrisă va rula pe toate sistemele de operare pe care este disponibil VS.

2.5 Constrângeri de design implementare

Constrângerile legate de implementare sunt puține deoarece programul utilizează Visual Studio și Oracle Databases.

2.6 Documentația utilizatorului

Utilizatorul va avea în meniu un buton de help care îl va ajuta pe utilizator să înțeleagă funcționalitatea aplicației.

2.7 Ipoteze si dependențe

Se pleacă de la ipoteza că aplicația creată va funcționa și pe versiunile ulterioare de Microsoft Visual Studio, dar se recomandă testări suplimentare.

3. Cerințe externe pentru interfață

3.1 Interfața utilizatorului

Aplicația se deschide cu interfața de logare în care utilizatorul își introduce contul pentru a putea folosi aplicația. Dacă după logare, contul este unul valid se deschide pagina principală de unde utilizatorul va avea de ales din trei opțiuni: *Vinde, Istoric, Modele*. Fiecare opțiune deschide o subinterfață, butonul *Vinde* deschide interfața de vânzare a unei mașini cu butonul de generare factură, butonul *Istoric Vânzări* deschide o interfață în care se pot vedea tranzacțiile anterioare, iar butonul *Modele* afișează modele de autoturisme aflate în stocul reprezentanței.

3.2 Interfața hardware

Interfața hardaware este una simplă deoarece aplicația necesită doar un mouse și o tastatură pentru a introduce datele despre clienți si mașini.

3.3 Interfața software

Aplicația noastră poate rula pe orice versiune de Visual Studio mai nouă de VS 2017. Codul aplicației este scris in limbajul C# versiunea 7.3. Pentru a putea rula corect programul, este nevoie de folosirea unui program de gestiune a unei baze de date în care se stochează toate informațiile necesare utilizatorului. Noi am folosit Oracle SQL Developer versiunea 18c, iar conectarea la baza de date se face cu ajutorul librăriei Oracle.ManagedDataAccess.Client.

3.4 Interfața comunicației

Folosirea aplicației necesită doar un calculator cu un sistem de operare ce poate rula Visual Studio 2017 +.

4. Caracteristicile sistemului

Clasa **Cryptography.cs** este clasa ce realizeaza encriptarea datelor legate de utilizatori. Cu ajutorul claselor Encrypt si Decrypt, parola este convertită și stocată sub o altă formă decât cea inițială. Această clasă este contruită cu ajutorul librăriei System.Security.Cryptography.

Clasa **DatabaseConnection.cs** este clasa ce realizeaza conectarea la baza de date cu ajutorul metodei GetConnectionInstance() și cu datele bazei de date necesare (data source, user ID, parola).

Clasa **Program.cs** este clasa principală ce pornește aplicația.

Clasa **ProxyInterfaceManager.cs** este clasa ce se ocupă cu logarea utilizatorilor. Aici avem o structură User ce conține informații despre utilizator: numele, parola și tipul contului. Tot aici

avem și metoda Login, ce verifică corectitudinea datelor introduse de utilizator atunci când se loghează.

Clasa **UnitTest1.cs** este clasa ce realizează corectitudinea rulării programului prin diferite teste:

-verificare date utilizator;

-verificare dacă un client a fost introdus corect în baza de date;

-verificări pentru constrângeri etc.

Clasele pentru interfețe:

Clasa **Interfata.cs** este clasa corespunzatoare interfeței principale. Aceasta conține 4 butoane: unul de schimbare al utilizatorului și trei pentru selecția operației dorite: vânzare, istoric, modele disponibile. De asemenea, în partea superioară a interfeței se află butonul de help, ce poate fi accesat de utilizator/admin.

Clasa **AdaugaClient.cs** este clasa corespunzatoare interfeței de adăugare a noilor clienți. Aceasta are câmpuri în care se completează datele esențiale ale clienților cum ar fi: numele, telefonul, tipul de plată, mail-ul. Fiecare câmp are constrângerile sale pentru a asigura o introducere corectă a datelor: numele trebuie sa fie alcătuit doar din litere, numărul de telefon trebuie să conțină 10 cifre, mail-ul trebuie să respecte formatul.

Asemenea sunt structurate clasele **AdaugaModel.cs** și **AdaugaProducator.cs**.

Clasa **Vanzare.cs** este clasa în care este implementată interfața de vânzare a unui autoturism. Aceasta conține 3 combobox-uri de unde se pot selecta numele clientului, producătorul și modelul de mașină dorit. De asemenea, aici se poate selecta tipul clientului: persoană fizică sau juridică. În partea inferioară, se completează detalii finale aparținând mașinii pentru generarea facturii (data achiziției, seria șasiu, costul).

Clasa **Istoric.cs** este clasa în care se poate vedea istoricul vânzărilor. Utilizatorul are la dispoziție două butoane: *Afișează* și *Șterge* prin care poate vedea istoricul sau șterge o anumită factură.

Clasa **Modele.cs** este clasa în care se pot vedea mașinile disponibile în reprezentanță. Utilizatorul are la dispoziție două butoane: *Afișează* și *Șterge* prin care poate vedea autoturismele sau șterge un anumit autoturism care a fost vândut sau retras.

5. Alte cerințe

5.1 Cerințe pentru performanță

Aplicația poate rula pe un sistem de operare mai vechi (Windows 7), dar și pe cele mai noi (Windows 10, Ubuntu etc.). Nu necesită o placă grafică sau o memorie ram mai mare de 2GB.

5.2 Cerințe pentru siguranță

Nici o măsură de siguranță nu trebuie luată pentru folosirea aplicației noastre.

5.3 Cerințe pentru securitate

Prin ecriptarea parolei, aplicația este sigură și poate fi folosită de către orice reprezentanță. De asemenea datele clienților sunt stocate local în baza de date.

5.4 Calitățile Soft-ului

Aplicație este foarte ușor de învățat și oferă o interfață prietenoasă cu utilizatorul. Nu are cerințe soft foarte mari.

5.5 Reguli de business

Echipa noastră a urmărit să îndeplinească standardele de conduită stabilite de către Universitatea Tehnică Gheorghe Asachi.

6. Anexe

Părți semnificative ale codului sursă, comentate:

Partea de login din ProxyInterfaceManager.cs

```
/// <summary>
/// Metoda de Login in interfata
/// </summary>
/// <param name="username">Numele utilizatorului</param>
/// <param name="password">Parola utilizatorului</param>
/// <returns></returns>
public bool Login(in string username, in string password)
{
	foreach (User u in _users)
	{
		if (u.Name == username && u.PassHash == Cryptography.Cryptography.HashString(password))
		{
			_currentUser = u;
			return true;
		}
	}

	return false;
}
```

Metoda de încărcare a interfeței de vânzare din Vanzare.cs:

```
/// <summary>
/// Metoda apelata la incarcarea interfetei Vanzare
/// </summary>
/// <param name="sender">Sender-ul evenimentului</param>
/// <param name="e">Argumentele evenimentului</param>
private void Vanzare_Load(object sender, EventArgs e)
{
        comboBoxModel.SelectedIndex = -1;
        comboBoxProducator.SelectedIndex = -1;
        comboBoxClient.SelectedIndex = -1;
        try
        {
                OracleCommand cmd = new OracleCommand(); ;
                OracleDataReader dr;
                cmd.CommandText = "SELECT nume_marca FROM producator";
          cmd.Connection = _dbConn;
          _dbConn.Open();
          dr = cmd.ExecuteReader();
          if (dr.HasRows)
          {
                List<Object> prod = new List<Object>();
                  while (dr.Read())
                  {
                        prod.Add(dr["nume_marca"].ToString());
                  }
                  object[] prodArray = prod.ConvertAll<object>(item => (object)item).ToArray();
                  comboBoxProducator.Items.Clear();
                  comboBoxProducator.Items.AddRange(prodArray);

          }
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message + "\nInterfata nu a putut fi incarcata. Marcile auto nu au putut
fi incarcate.", "Eroare-incarcare marca auto");
        }
        finally
        {
            _dbConn.Close();
        }
        try
        {
            OracleCommand cmd = new OracleCommand(); ;
            OracleDataReader dr;
            cmd.CommandText = "SELECT nume_client FROM client";
            cmd.Connection = _dbConn;
            _dbConn.Open();
            dr = cmd.ExecuteReader();
            if (dr.HasRows)
            {
                List<Object> prod = new List<Object>();
                while (dr.Read())
                {
                    prod.Add(dr["nume_client"].ToString());
                }
                object[] prodArray = prod.ConvertAll<object>(item => (object)item).ToArray();
```

```
                comboBoxClient.Items.Clear();
                comboBoxClient.Items.AddRange(prodArray);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message + "\nInterfata nu a putut fi incarcata. Clientii nu au putut fi
incarcati.", "Eroare-incarcare clienti");
        }
        finally
        {
            _dbConn.Close();
        }
    }
```

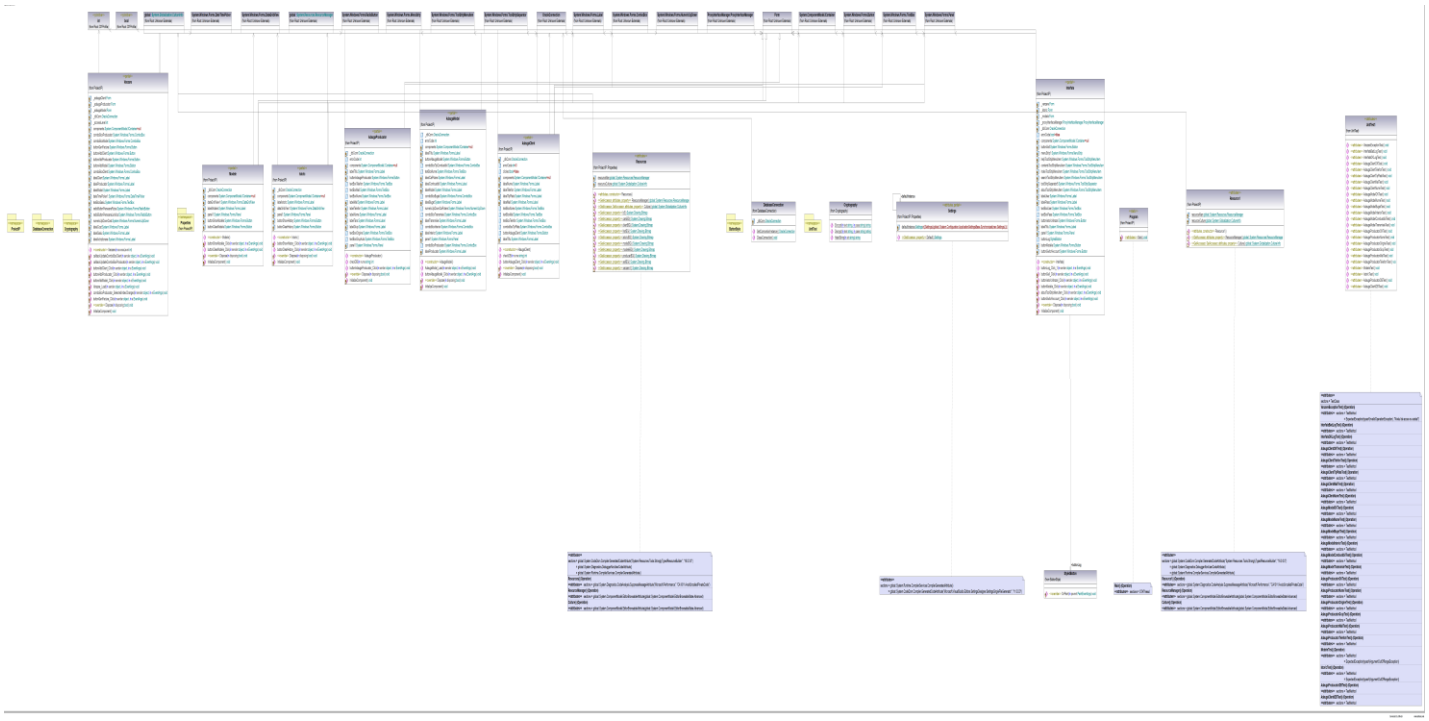
Adăugarea unui client în baza de date din AdaugaClient.cs

```
    /// <summary>
        /// Metoda ce adauga un client in baza de date daca datele acestuia respecta
    constrangerile inpuse
        /// </summary>
        /// <param name="sender">Sender-ul evenimentului</param>
        /// <param name="e">Argumentele evenimentului</param>
        public void buttonAdaugaClient_Click(object sender, EventArgs e)
        {
            clicked = true;
            if(!(new Regex(@"^[A-Za-z ]+$")).IsMatch(textBoxNume.Text) ||
    textBoxNume.Text.Length <= 4)
            {
                errorCode = 1;
                if(this.Visible)
                    MessageBox.Show("Formatul numelui clientului nu este potrivit!");
                return;
            }

            if (!(new Regex(@"[a-z0-9._%-]+@[a-z0-9._%-]+\.[a-
    z]{2,4}")).IsMatch(textBoxMail.Text) || textBoxMail.Text.Length < 1)
            {
                errorCode = 2;
                if(this.Visible)
                    MessageBox.Show("Formatul adresei de mail nu este potrivit!");
                return;
            }

            if(comboBoxTipPlata.SelectedIndex < 0)
            {
                errorCode = 3;
                if(this.Visible)
                    MessageBox.Show("Tipul de plata nu este selectat!");
                return;
            }

            if (!(new Regex(@"[0-9]+$")).IsMatch(textBoxTelefon.Text) ||
    textBoxTelefon.Text.Length != 10)
            {
                errorCode = 4;
```

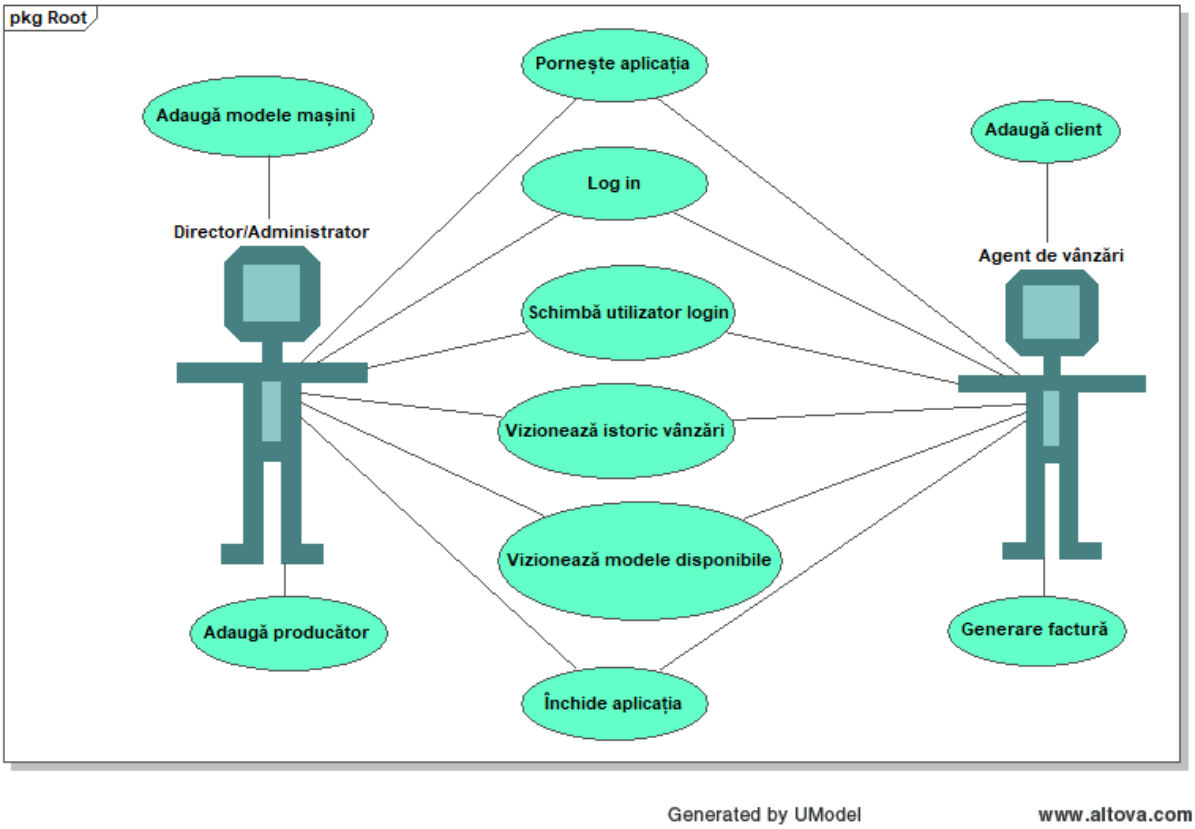
```
                if(this.Visible)
                    MessageBox.Show("Formatul numarului de telefon nu este potrivit!");
                return;
            }

            try
            {
                _dbConn.Open();
                string sql = "INSERT INTO client (nume_client, numar_telefon_client,
tip_plata, adresa_mail_client) VALUES ('" + textBoxNume.Text + "', '" + textBoxTelefon.Text
+ "', '" + comboBoxTipPlata.Text + "', '" + textBoxMail.Text + "')";
                OracleCommand cmd = new OracleCommand(sql, _dbConn);
                cmd.BindByName = true;
                OracleDataAdapter adapter = new OracleDataAdapter(cmd);
                DataSet ds = new DataSet();
                adapter.Fill(ds);
                this.Close();
            }
            catch (Exception ex)
            {
                errorCode = 5;
                MessageBox.Show(ex.Message + "\nClientul nu a putut fi introdus in baza de
date!", "Eroare - introducere client!");
            }
            finally
            {
                _dbConn.Close();
            }

        }
```

II. Diagrame UML

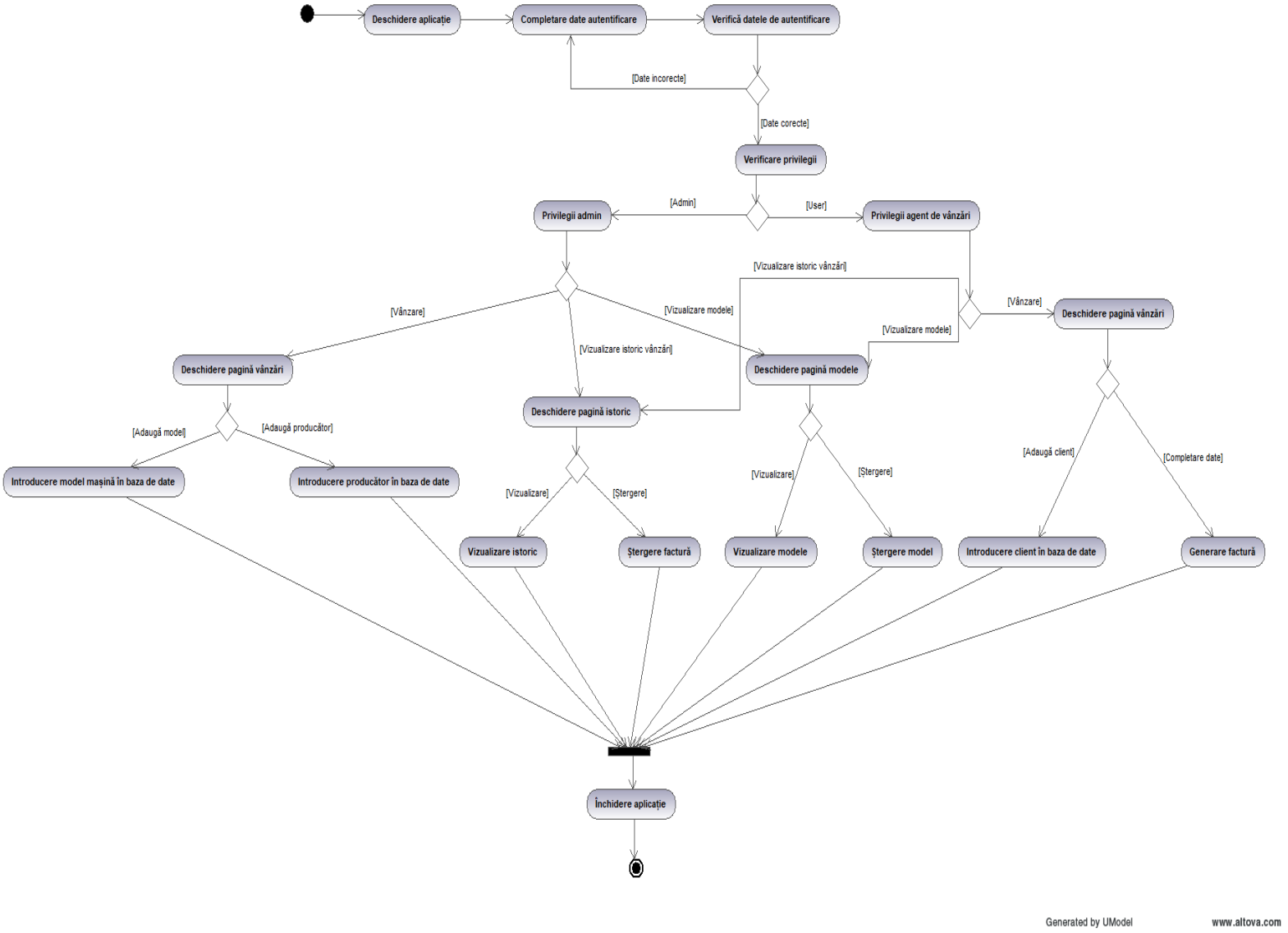
- **Diagrama de clase**



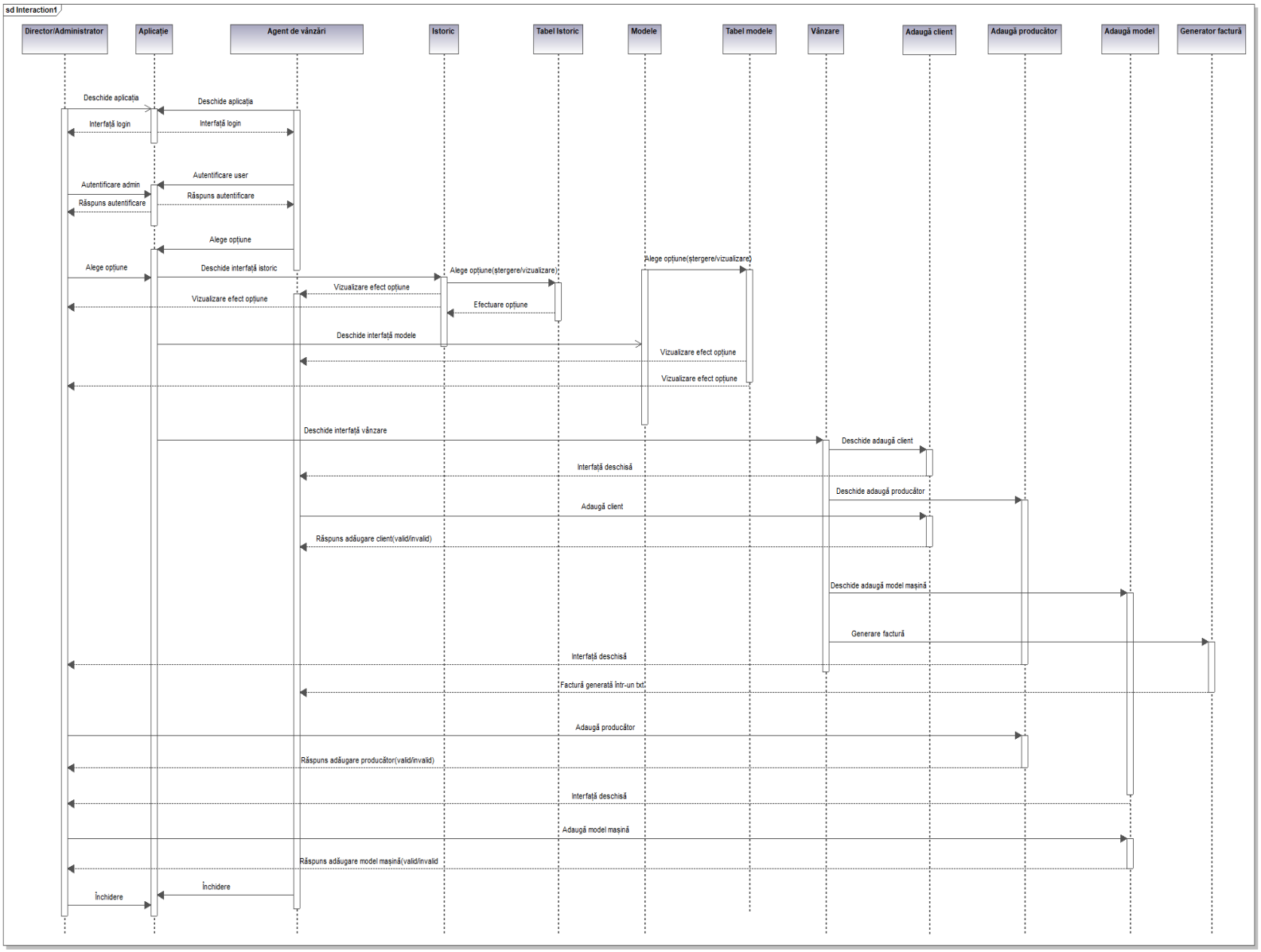
- **Diagrama cazuri de utilizare**



- **Diagrama de activități**



- **Diagrama de secvențe**



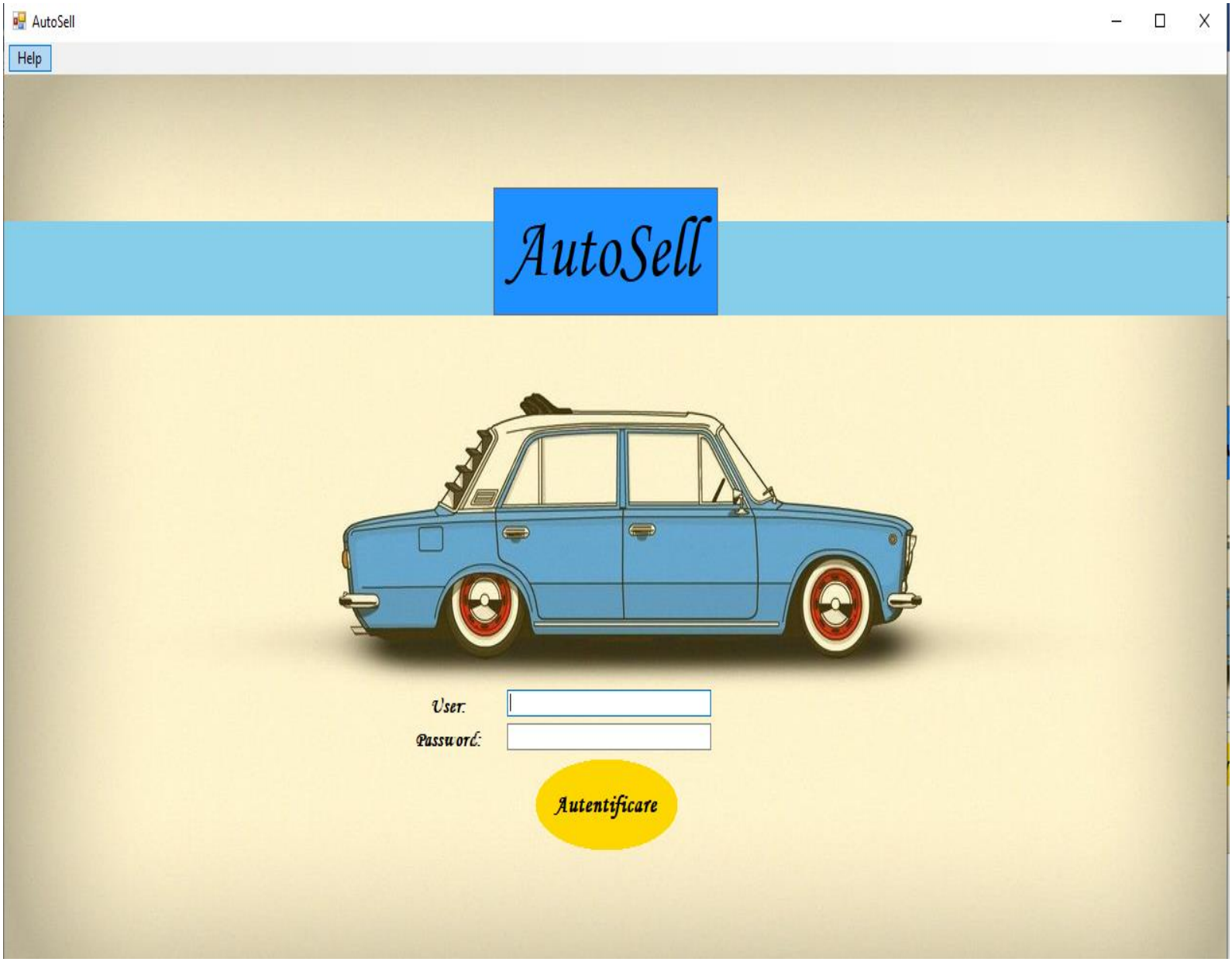
Generated by UModel www.altova.com

III. Mod de utilizare

Modul de utilizare este descris în help-ul programului, acolo utilizatorul poate vedea la ce folosește fiecare interfață.

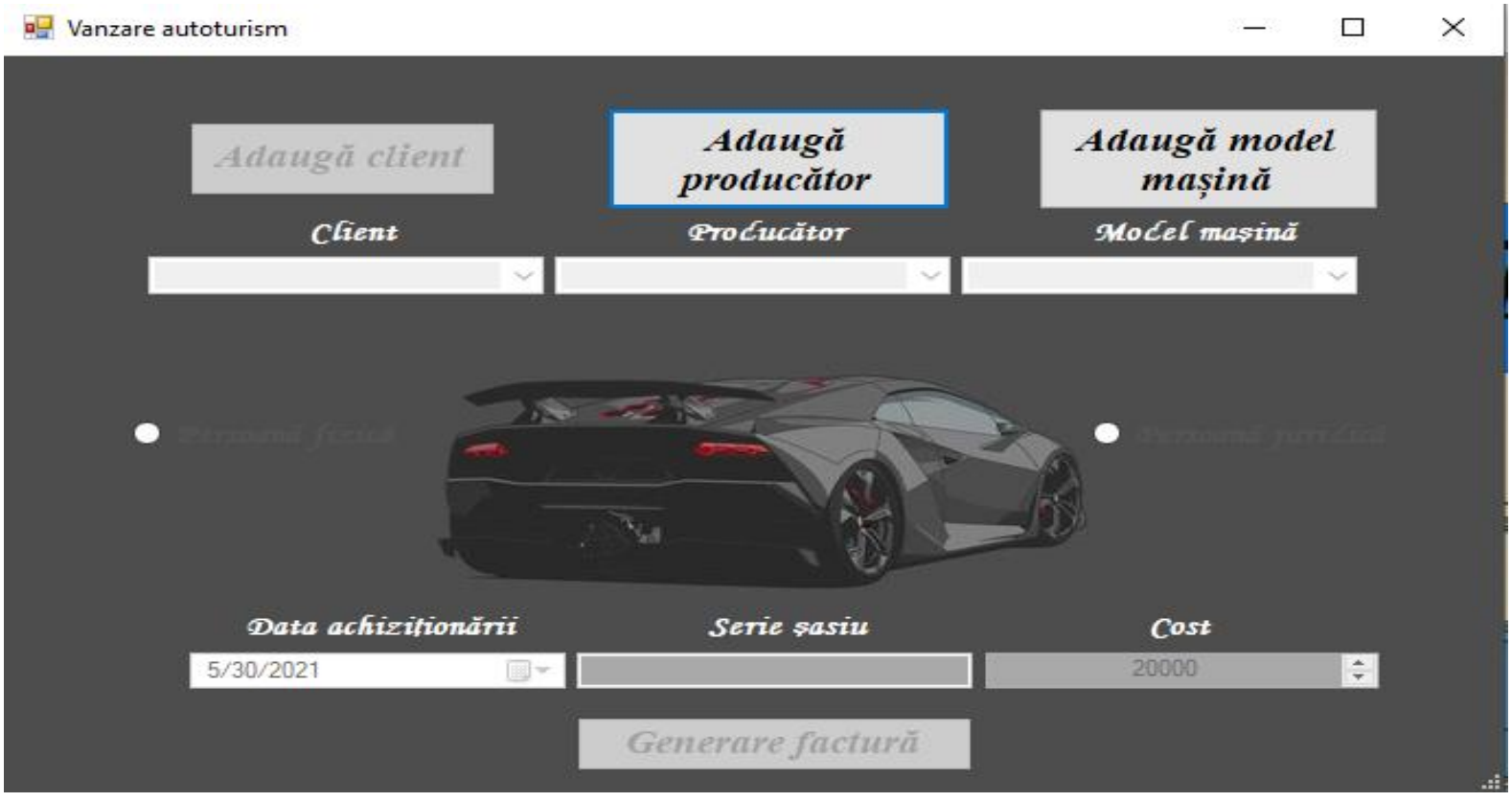
Aplicația se folosește urmarind pașii următori:

1. Agentul de vânzări/adminul se loghează.

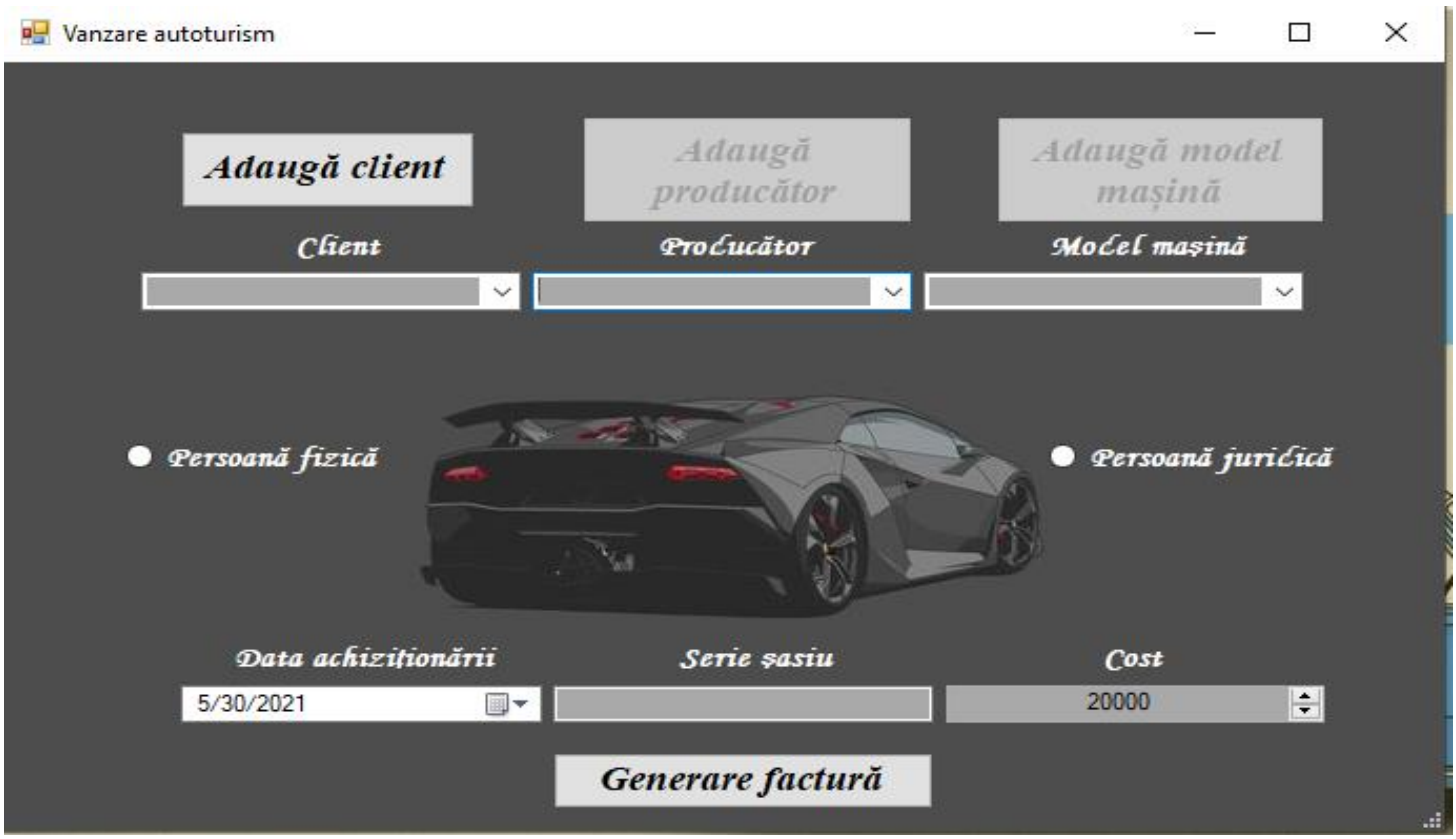


2. Dacă logarea s-a executat cu succes, se va deschide interfața principală a aplicației. Aici se poate schimba utilizatorul sau se pot accesa celelalte trei secțiuni ale programului.

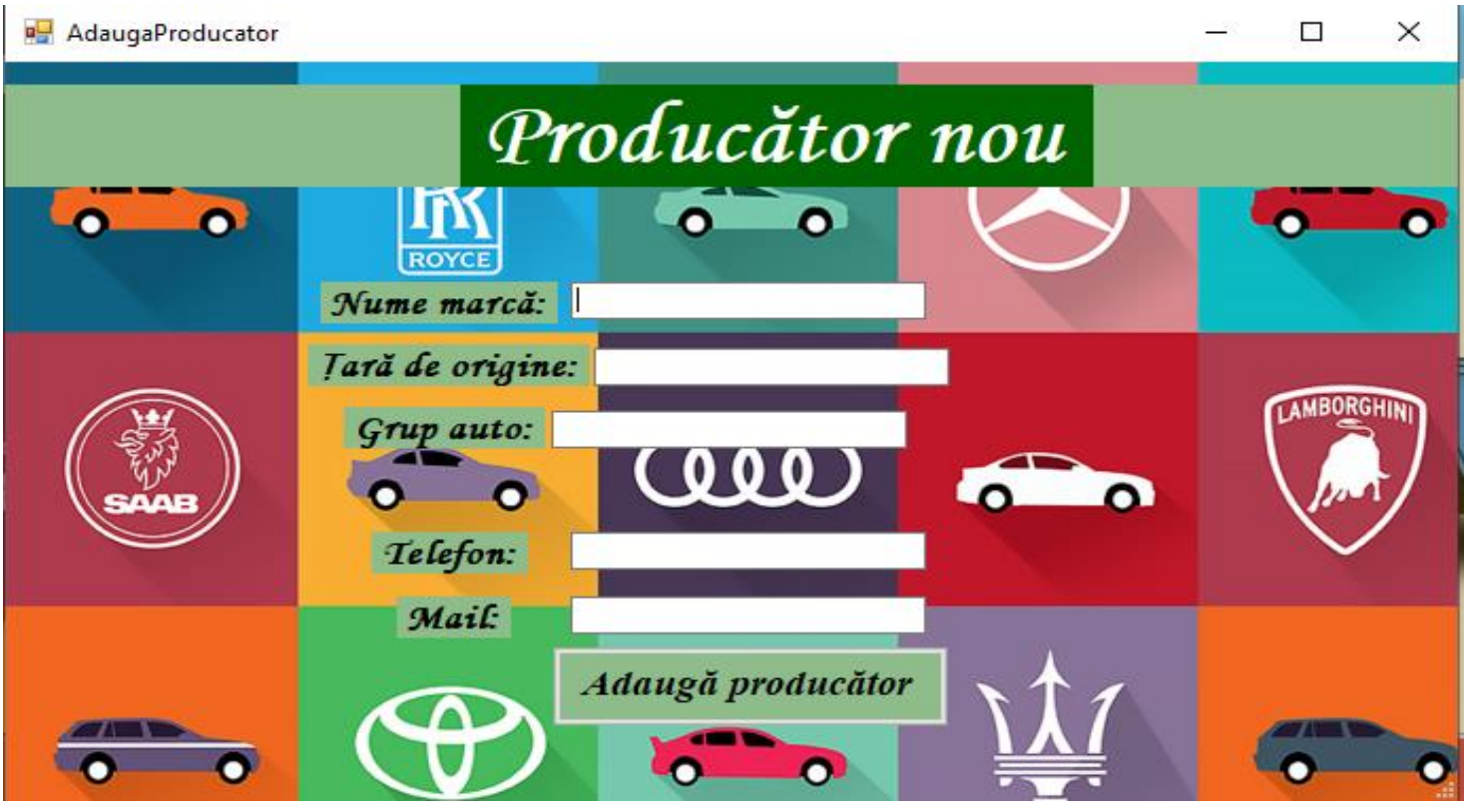
3. Dacă adminul apasă primul buton, *Vinde* se va deschide urmatoarea interfață:

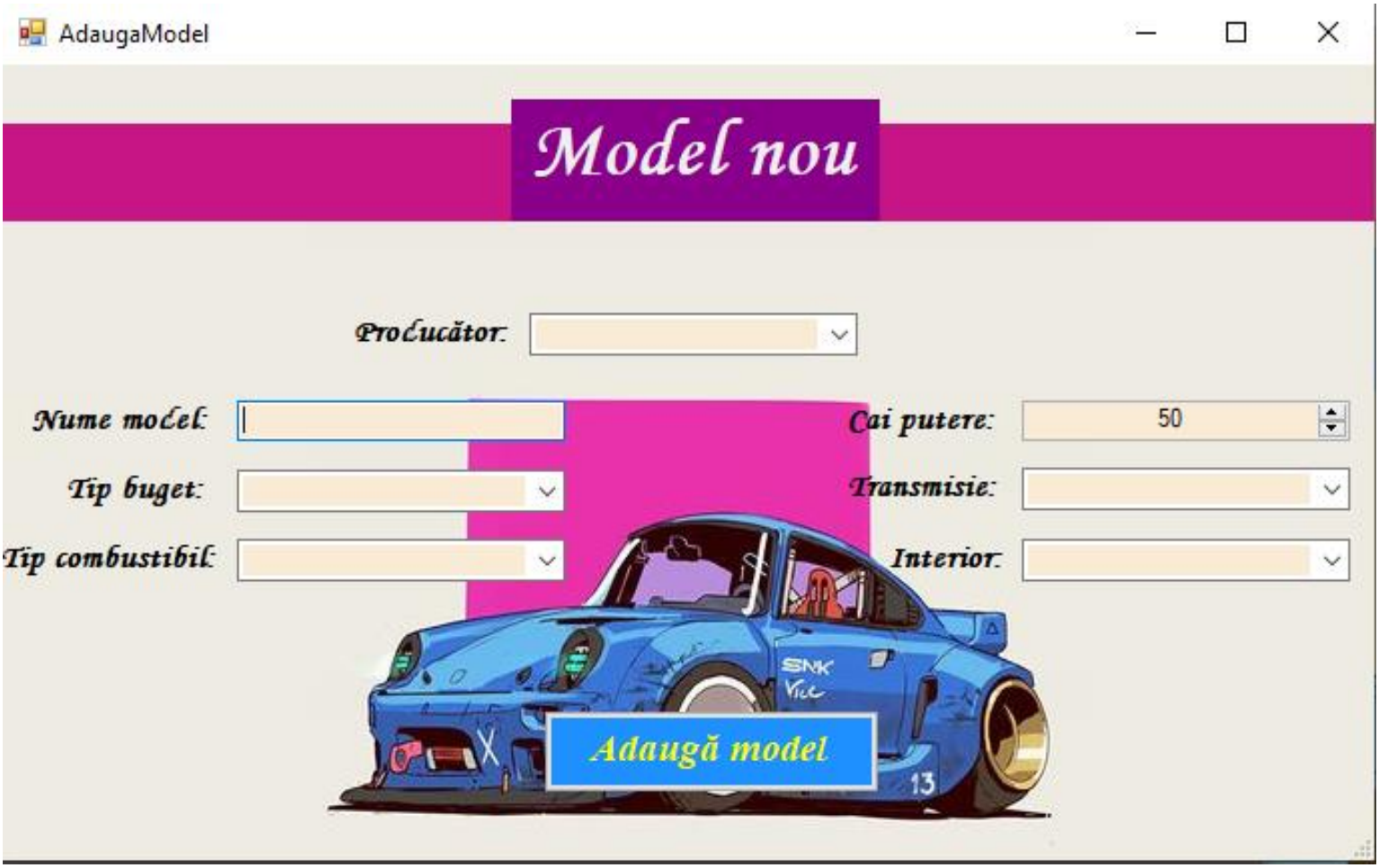


Dacă agentul de vânzări apasă primul buton, Vinde, se va deschide următoarea interfață:



4. Dacă logarea este făcută de către un administrator, acesta va avea acces la butoanele de adăugare producător/adăugare model mașină.





5. Dacă agentul de vânzări nu are drepturi de administrator, acesta va putea adăuga un client nou sau va putea completa câmpurile pentru a genera o factură într-un format txt.

6. Întorcându-ne la pagina principală, se poate selecta următorul buton, și anume *Istoric vânzări*, deschizând următoarea interfață unde se pot afișa facturile generate, până la acel moment sau se pot șterge chiar unele facturi.

	NR_FACTURA	TIP_PERSOANA	SERIE_SASIU	COST	DATA_ACHIZITIONARE	ID_CLIENT	MODEL_MASINA
▶	1000	fizica	1HGCM82633A004352	72000	3/2/2021	1	Q4 e-tron
	1001	juridica	2ABCD12343B001123	250000	10/1/2020	2	Mustang Mach-e
	1002	fizica	3HJKD10234K012462	80000	4/4/2021	3	Seria NX
	1003	fizica	5OIPL56723M021390	70000	11/21/2020	4	Land Cruiser
	1004	juridica	1BHJS24513T412134	12000	8/10/2020	5	Logan TCe 100
	1005	fizica	6CVBL44523D422456	200000	3/1/2019	6	X5 Xdrive
*							

7. Ultimul buton este cel de *Modele,* unde se poate vizualiza sau șterge orice model de mașina disponibil în reprezentanță sau retras.