Universitatea Tehnică „Gheorghe Asachi” din IAȘI
Facultatea de Automatică și Calculatoare,
Calculatoare și tehnologia informației

Proiect Ingineria Programării
Aplicație pentru gestionarea unui magazin cu produse cosmentice

Echipa: Baciu Claudia
Chelariu Emilia
Lungu Stefania
Poleac Alexandra

Curpins

I. Descrierea Aplicației

Aplicația noastră se numește „Rare Beauty” și are ca scop gestionarea unui magazin cu produse cosmetice. Am venit cu acest nume pentru că în opinia noastră fiecare persoană este unică și specială în felul ei, iar produsele de make-up pun în lumină și evidențiază frumusețea fiecăruia. Aplicația noastră poate fi utilizată atât de clienți (pentru a vedea produsele si prețurile lor, pentru a plasa o comandă), cât și de administratorul acestui magazin (pentru a modifica produsele, a șterge sau a adăuga in listă). Drepturile de admin sunt diferite de cele oferite clientului, iar aplicația deosebește utilizatorii in functie de datele de logare introduse.

Consideram că aplicația noastră ar putea fi un produs important de scos pe piață pentru că ar putea oferi suport în desfășurarea activității a mai multor magazine. Este ușor de utilizat, foarte utilă și cu o interfață cu utilizatorul plăcută. Aceasta dispune de un help care face mai simplă utilizarea butoanelor din aplicație.

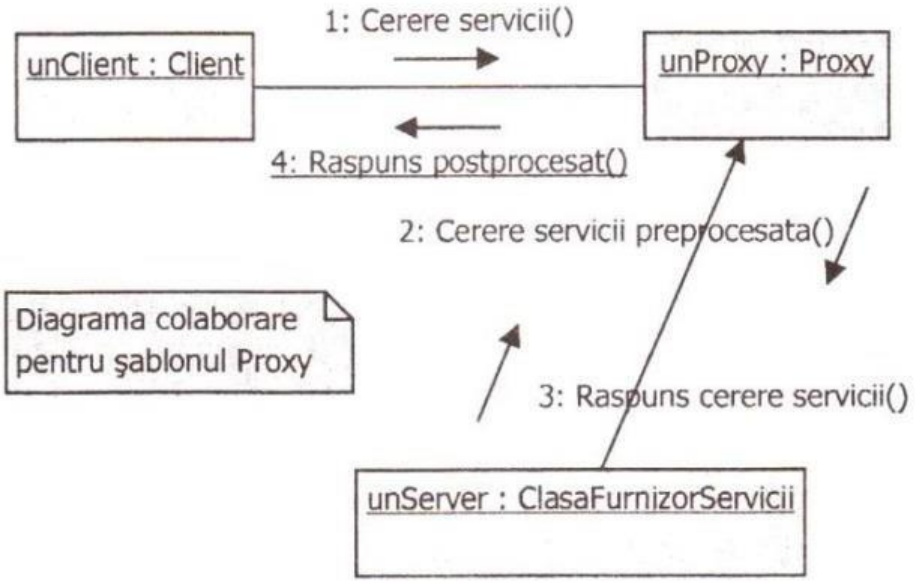
Logarea se face pe baza unui username și a unei parole. Proiectul nostru dispune de un fișier clienti.txt unde se găsesc datele disponibile de conectare la calea: MagazinCosmetice\bin\Debug\Clienti.

Butonul de Login a fost implementat astfel încât să fie tratată fiecare situație în parte. Dacă username-ul sau parola nu sunt corecte, acesta afișează „Eroare la conectare! Verificați user-ul și parola!”, dacă user-ul este “admin”, utilizatorul aplicației va fi adminul, iar dacă user-ul este un altul din fișierul crienti.txt, atunci utilizatorul va fi un client. Interfața dispune și de un buton de delogare, unul de adăugare în coș, de plasare comandă(în cazul în care coșul este gol, clientul este rugat să adauge produse în coș) , de ștergere produse, de ieșire din cont client, de adăugare produs nou, de help pentru client/admin.

II. Sablonul de proiectare

Șablonul de proiectare utilizat în proiectul nostru este Proxy . Am luat în considerare varianta proxy-ului de protecție pentru stabilirea drepturilor de acces la niște documente protejate.

Obiectul Proxy acţionează ca un intermediar între codul-client şi obiectul care furnizează serviciile cerute de client. Principalul model de interacţiune specific şablonului Proxy este reprezentat grafic în următoarea diagramă care subliniază rolul de intermediar al obiectului Proxy între entitatea-client şi obiectul furnizor de servicii.



III. Diagrame UML

Diagrama UML: cazuri de utilizare

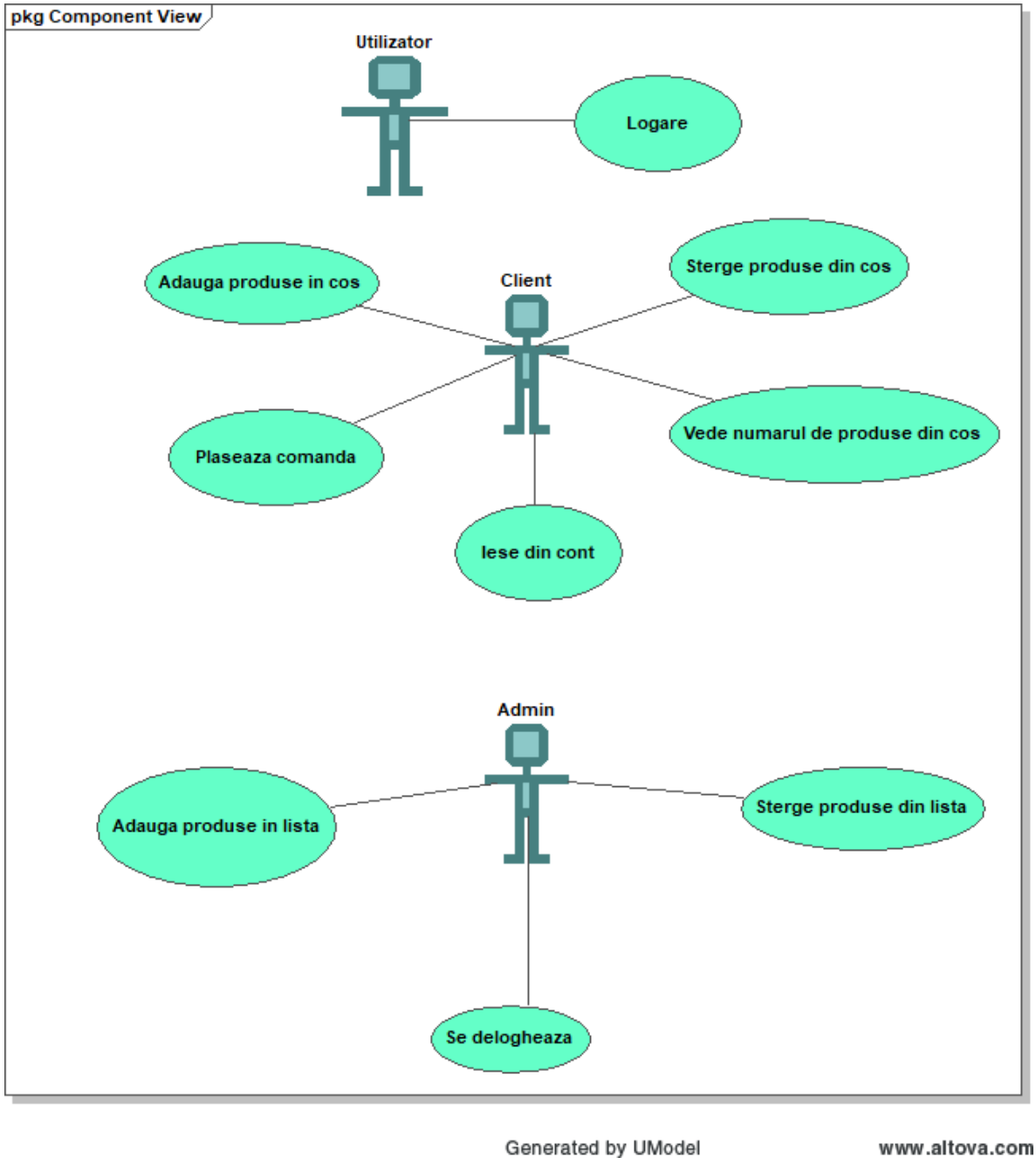
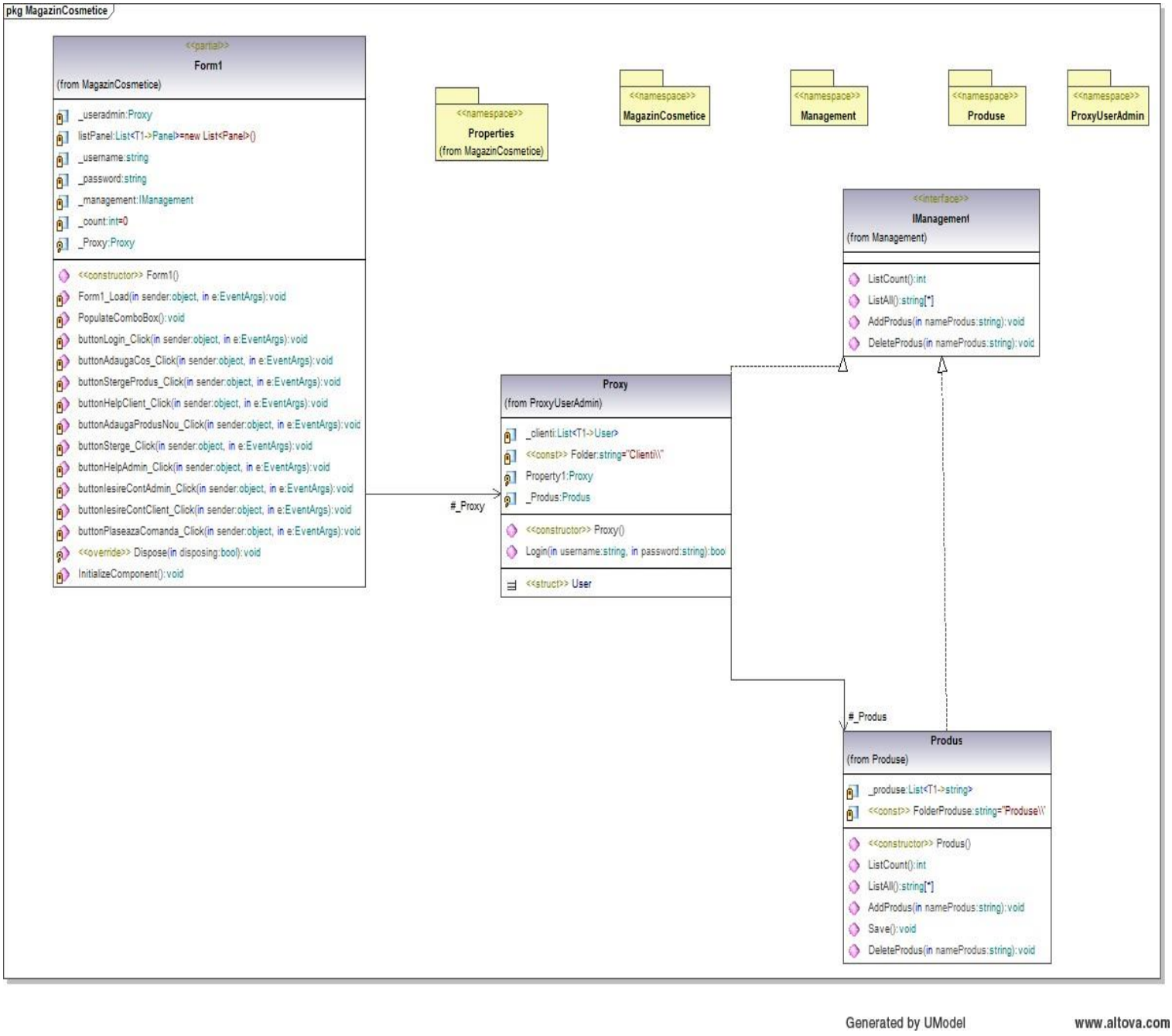


Diagrama UML: clase



Generated by UModel www.altova.com

Diagrama UML: activități

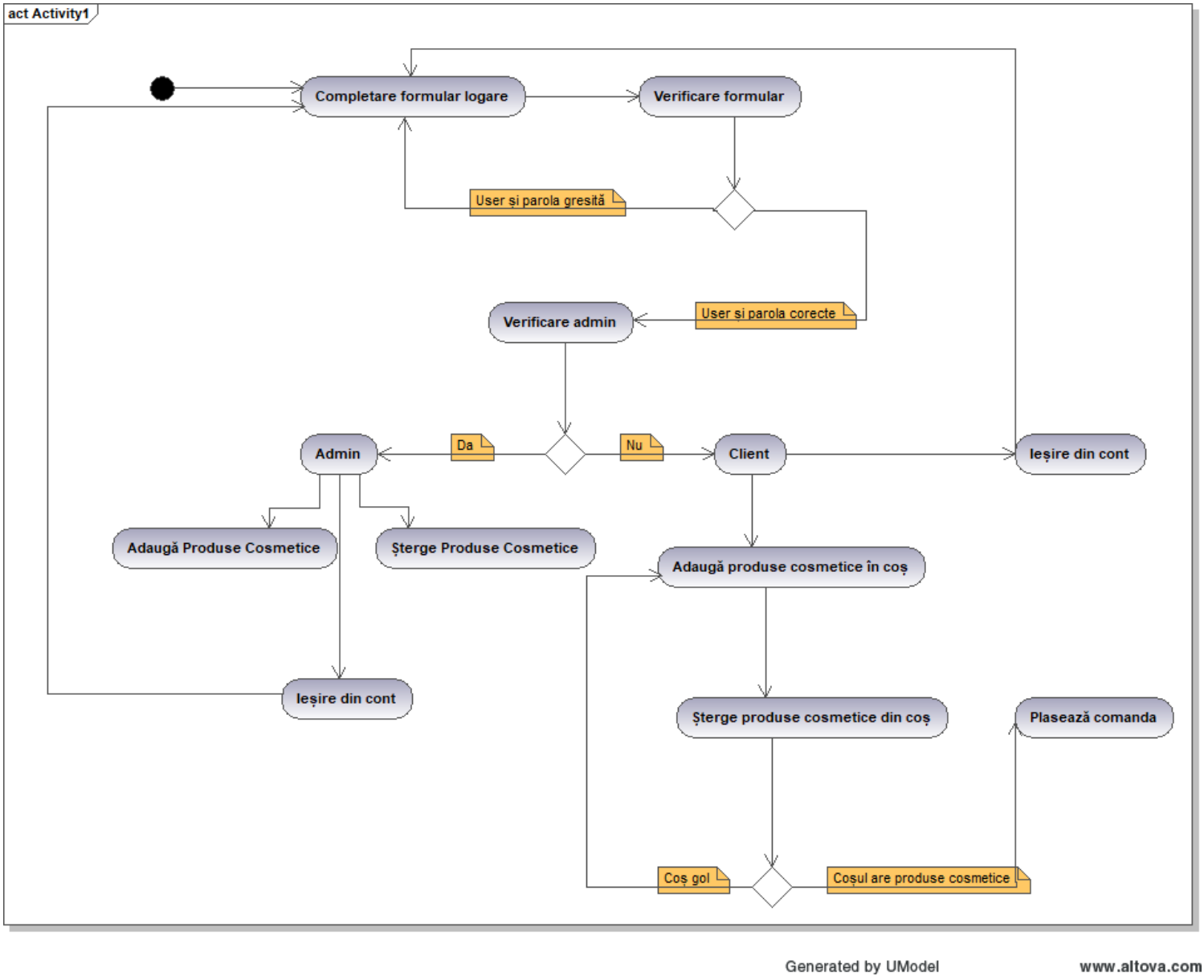
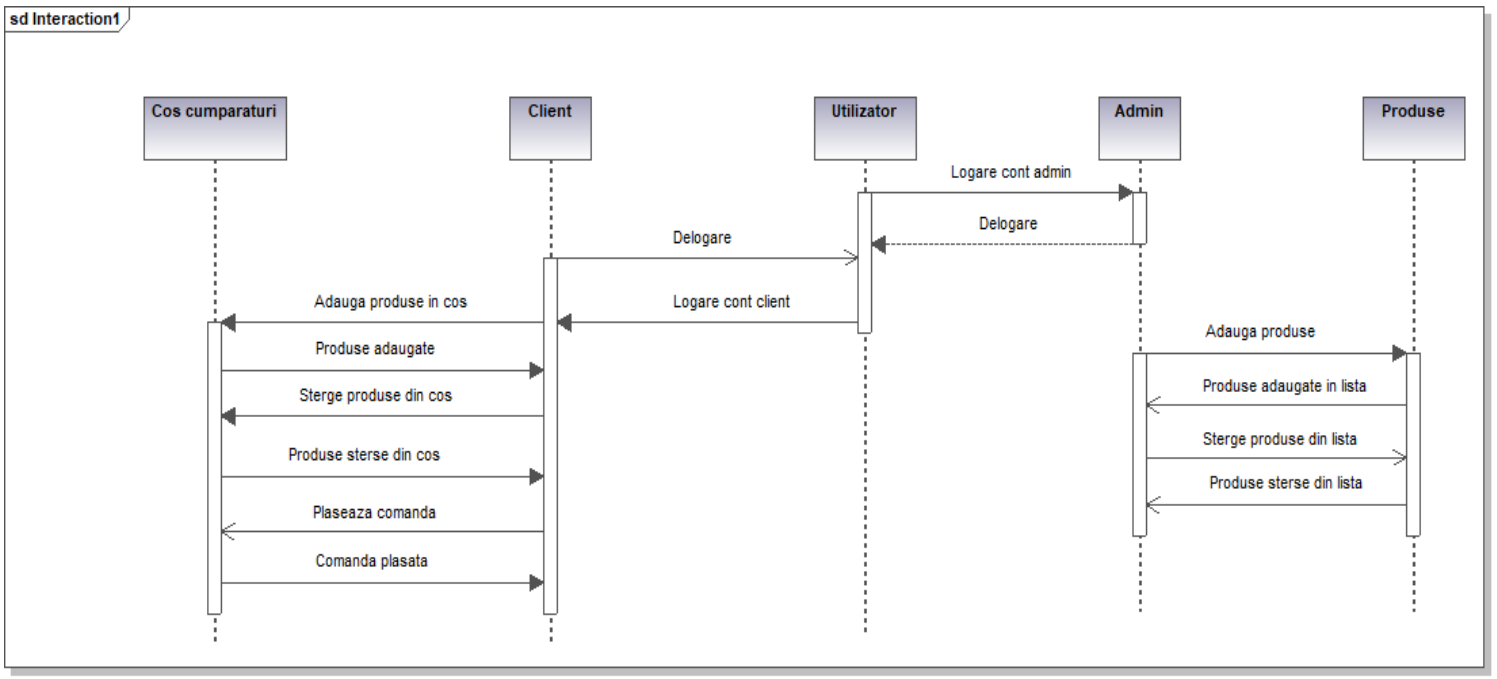


Diagrama UML: secvențe



Generated by UModel www.altova.com

IV. Modul de utilizare al programului

Programul realizat în cadrul echipei are scopul de a servi atât administratorului magazinului cu produse cosmetice, cât și clienților acestuia, de aceea modul de utilizare al proiectului trebuie sa fie simplu și clar pentru că vor participa și persoane cu o arie redusă de cunoștințe în domeniul tehnologiei.

Primul contact cu această aplicație este logarea:

• Cu ajutorul username-ului și al unei parole, persoana respectivă se va conecta pentru a avea acces la informații sau pentru a efectua diferite operații.

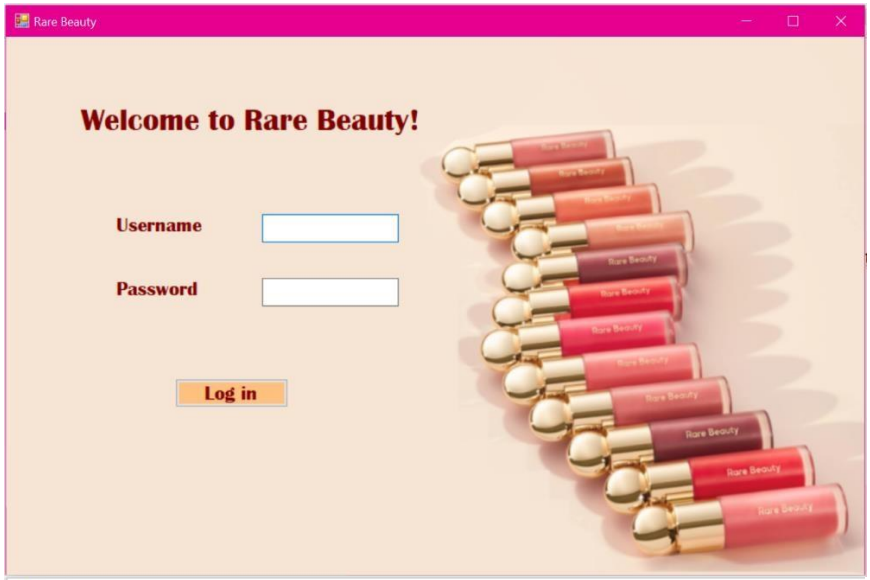
În funcție de user și parolă se stabilește funcția pe care o are utilizatorul:

• Dacă utilizatorul este client, acesta va avea posibilitatea de a vedea produsele disponibile, a adaugă produse în coș, a vedea câte produse are în coș și a plasa comenzi.Pentru a putea plasa comenzi, clientul trebuie să introducă mai întâi produsele în coș.

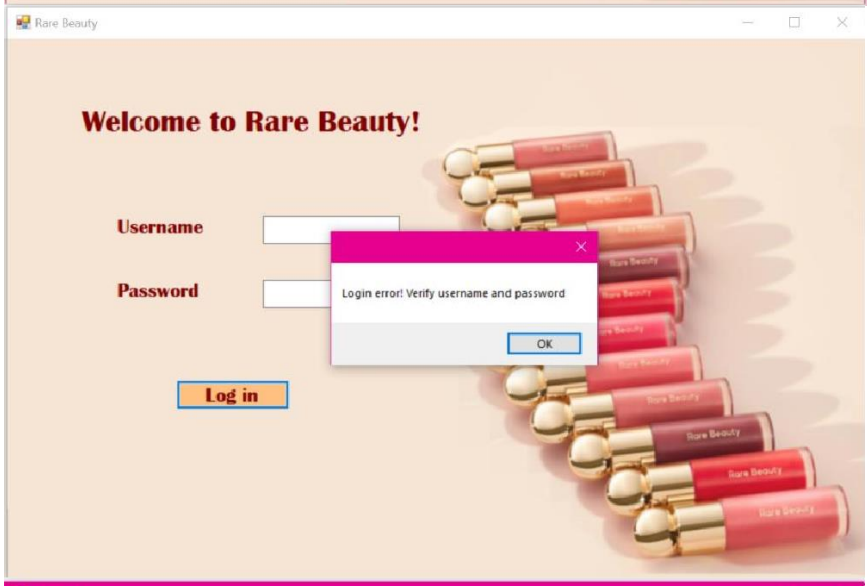
• Dacă utilizatorul este admin, acesta are dreptul să editeze lista cu produse disponibile sau prețul acestora. El poate să șteargă produse sau să adauge altele noi.

• Pentru fiecare utilizator este disponibil un help care pune la dispoziție informații cu privire la fiecare buton din interfață.

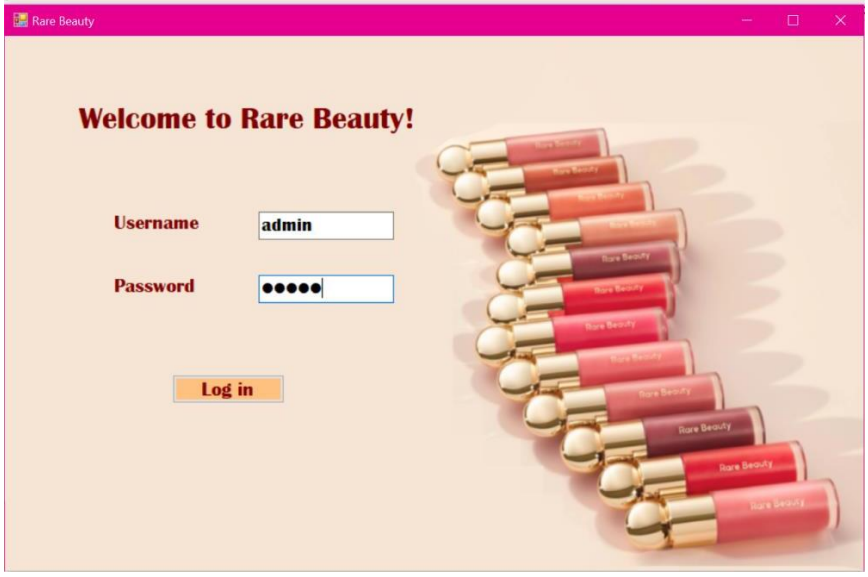
V. Programul in execuție



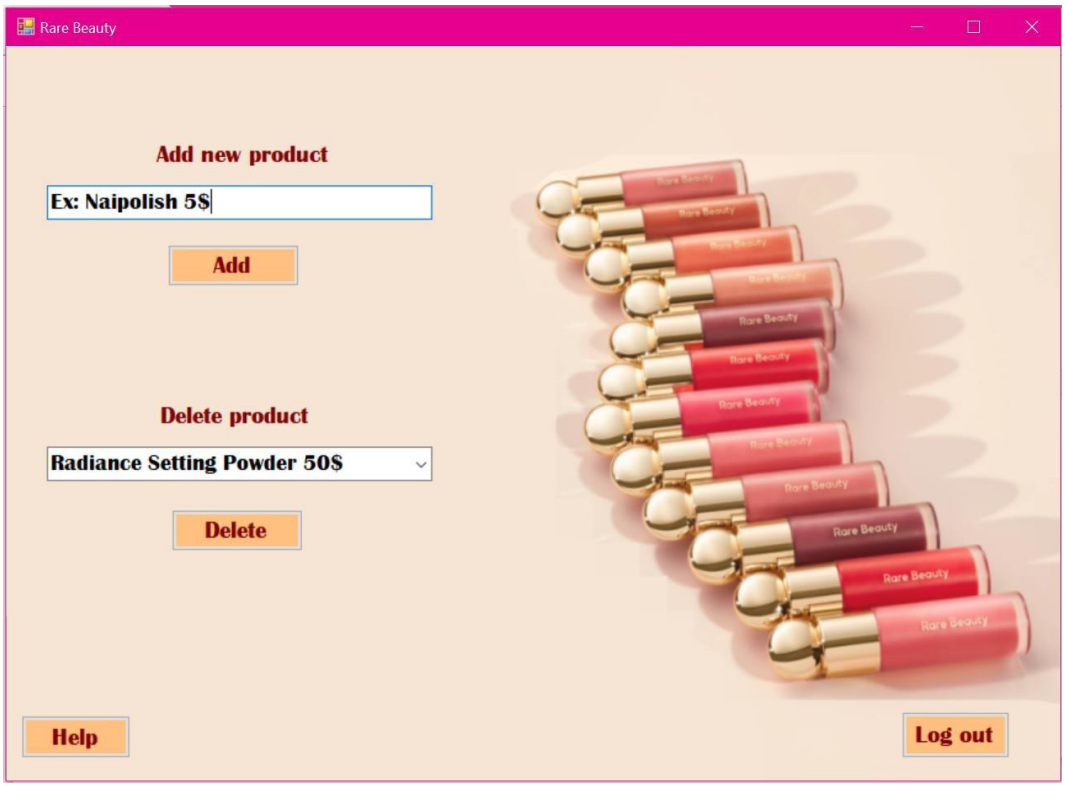
Acesta este primul contact cu interfața grafică unde trebuie introdus user-ul și parola.



În cazul în care user-ul și parola nu au fost introduse sau au fost introduse greșit, la apăsarea butonului de *Log in*, programul va afișa un mesaj de eroare.

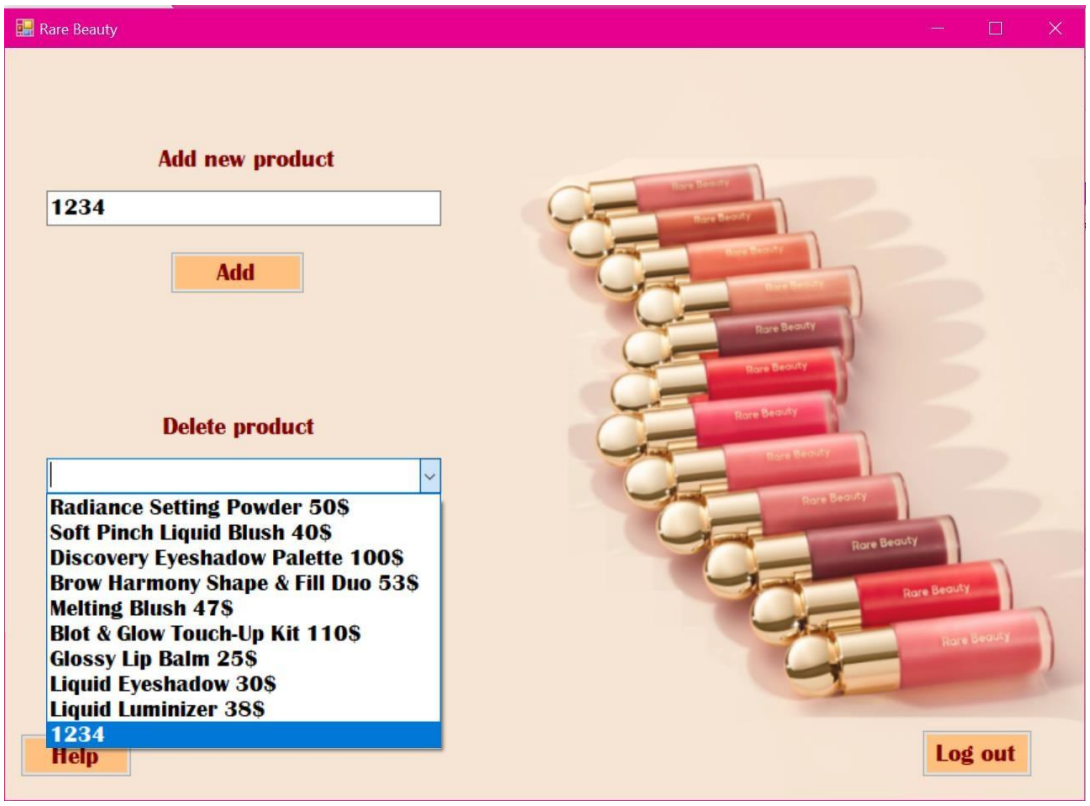


În acest mod se loghează o persoană pe contul de admin.

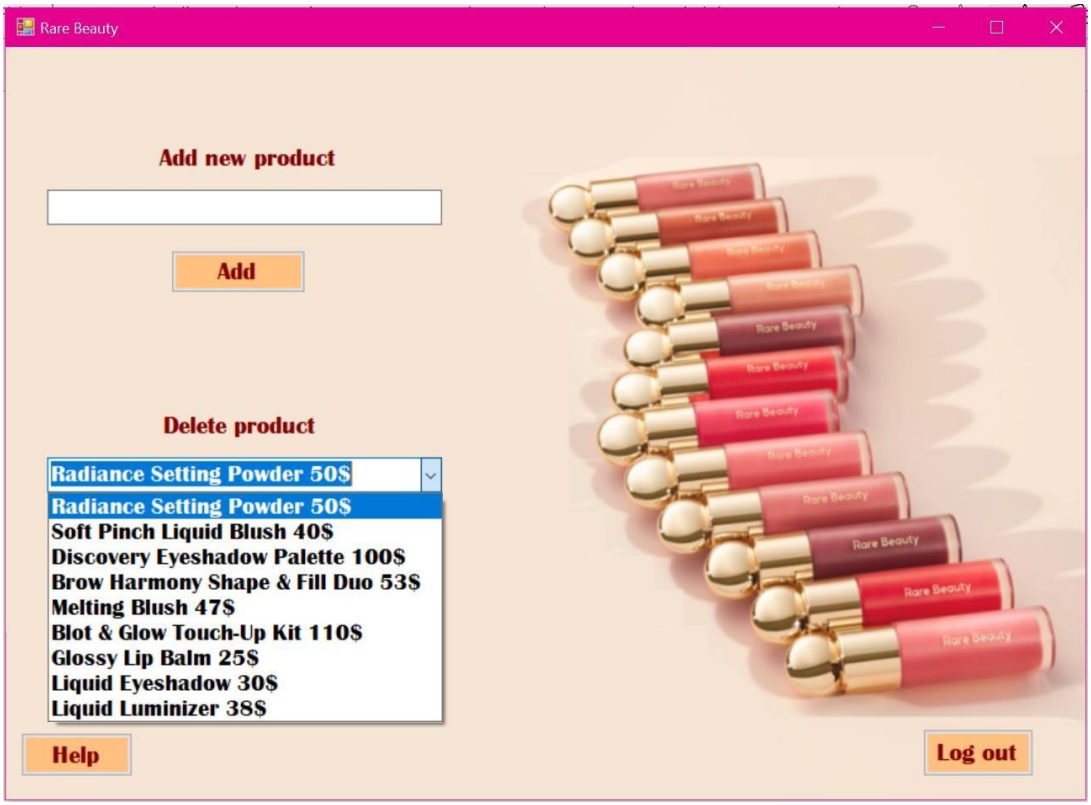


În urma logării cu contul de admin, în această interfață se pot adauga produse vizibile clientului și șterge produse din listă.

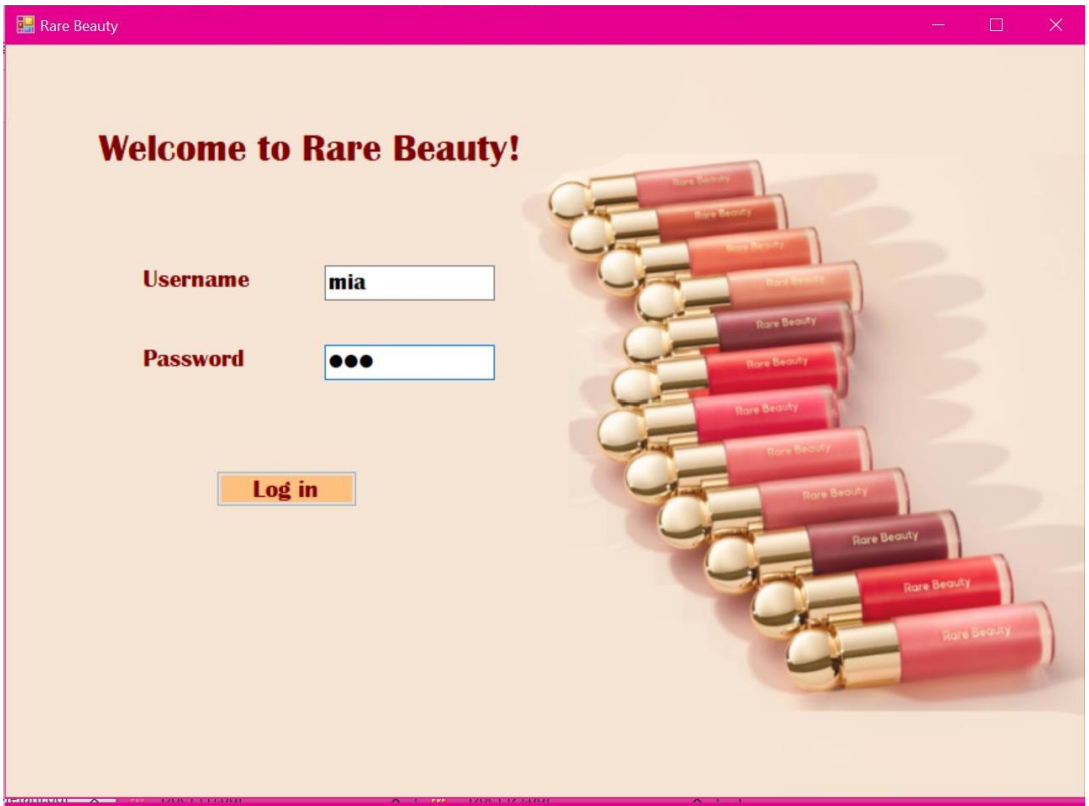
Dacă la apăsarea butonului *Add* nu este niciun produs introdus, programul va returna un mesaj de eroare.



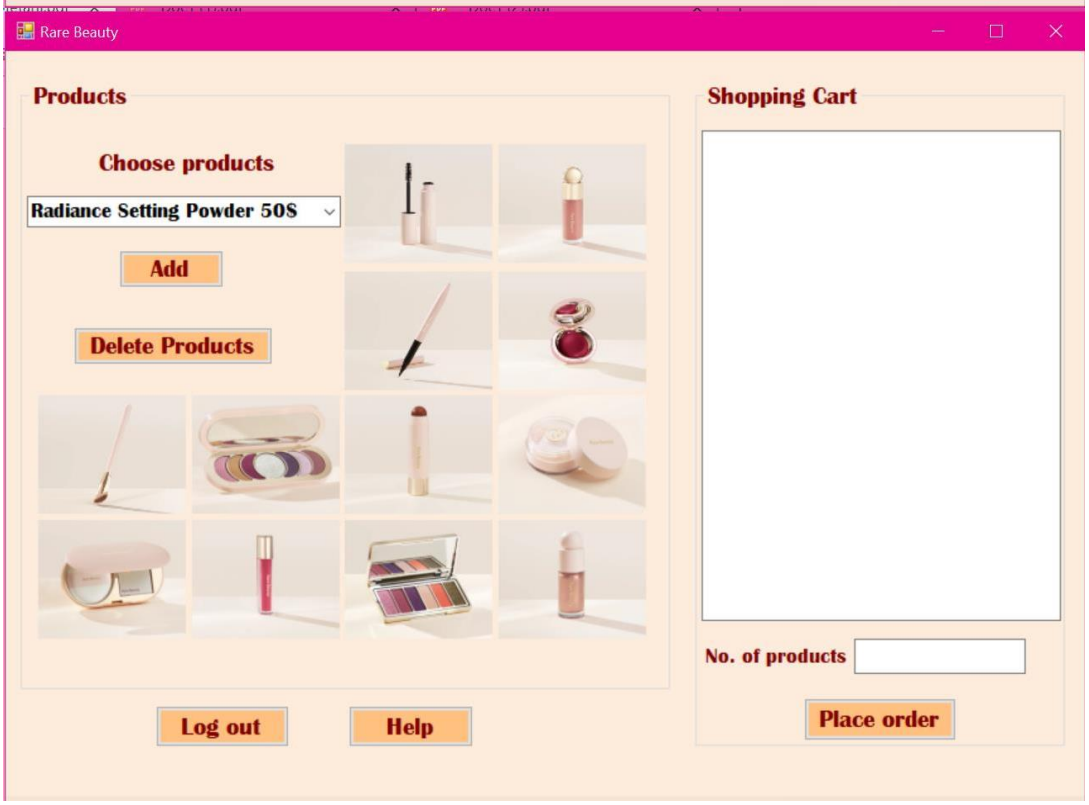
Dacă din lista *Delete product* selectezi un produs, în urma apăsării butonului *Delete*, acel produs va fi șters.



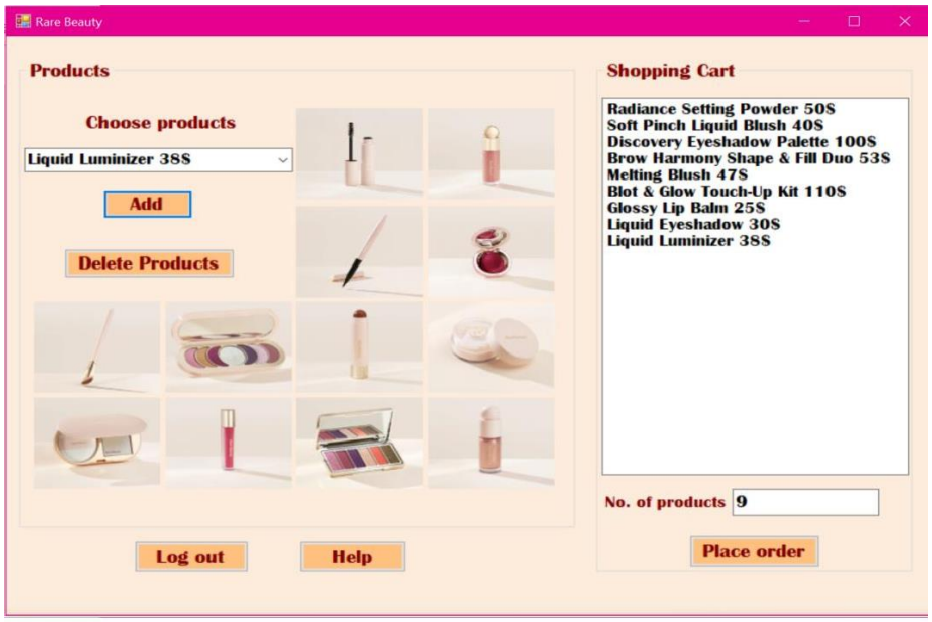
Produsul “1234” a fost șters!



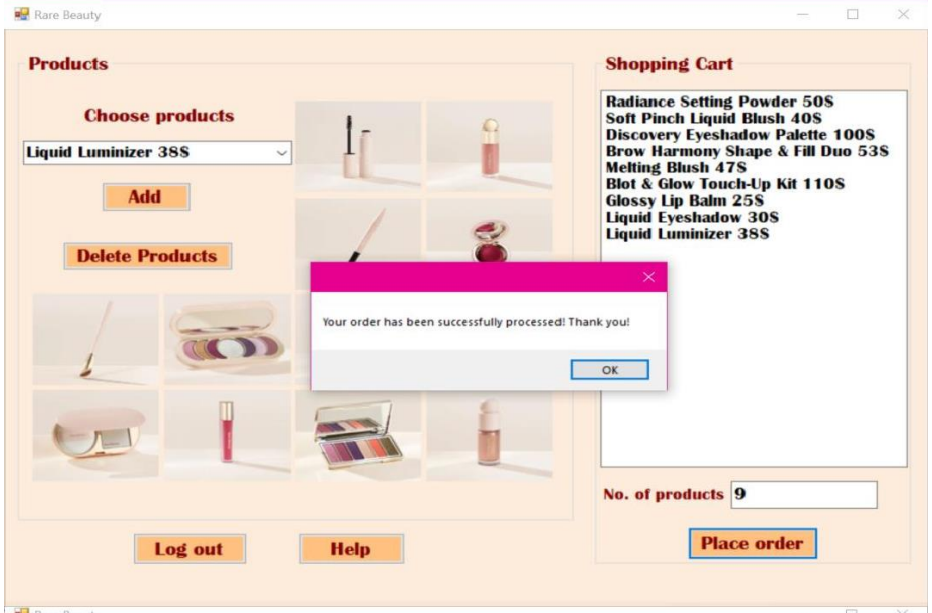
Acesta este un exemplu de logare pentru un client.



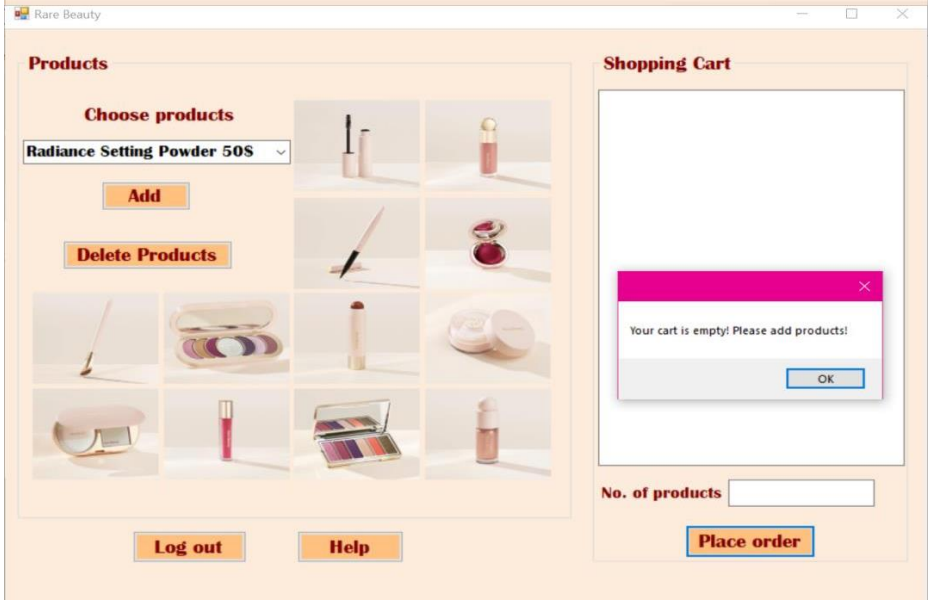
Dacă te înregistrezi cu un cont client, vei avea acces la aceasta interfață. Aici ai posibilitatea de a vedea lista de produse, de a adăuga produse in coș, de a le șterge, de a vedea cate produse ai adăugat în coș și de a plasa comanda .



În urma apăsării butonului *Add*, în stânga vei avea acces la coșul de cumpărături, iar mai jos în *No. Of products* vei vedea câte produse ai adăugat la momentul respectiv.

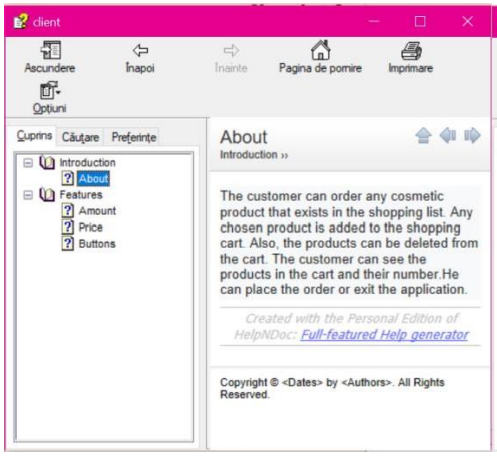


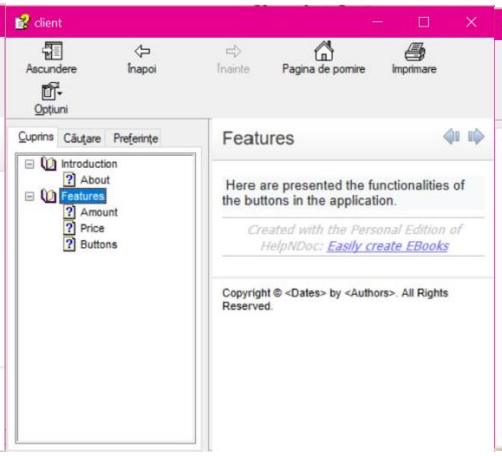
Dacă utilizatorul are produse în coș, acesta poate plasa comanda apăsând *Place order*.

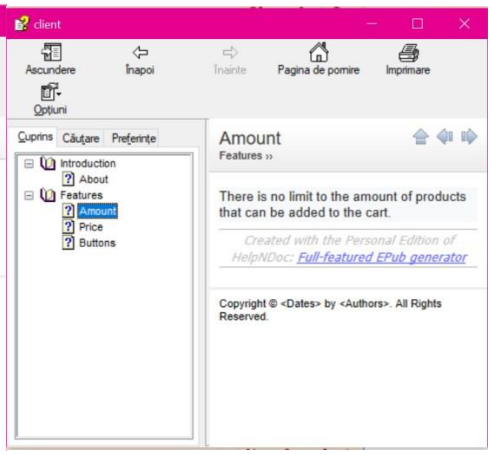


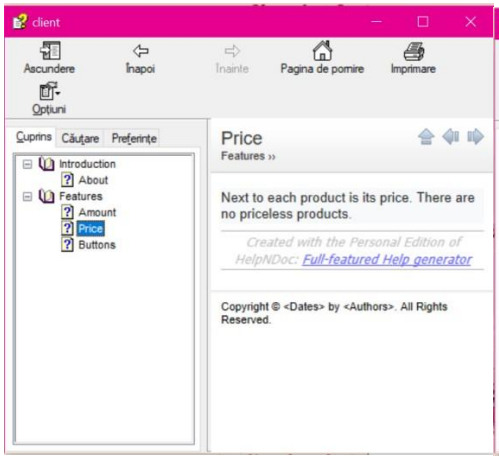
Dacă utilizatorul nu are produse în coș, dar acesta apasa *Place order*, va primi un mesaj de eroare.

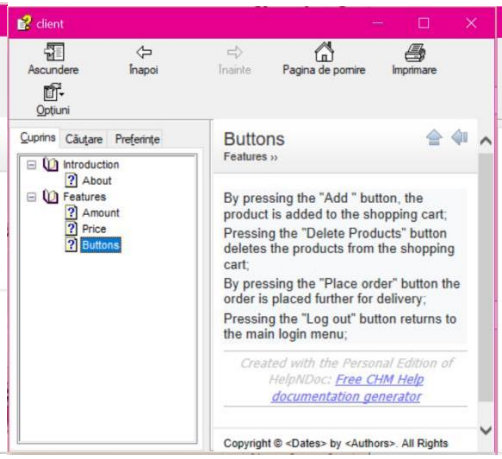
Mai jos sunt help-urile ajutătoare pentru admin, respectiv client.

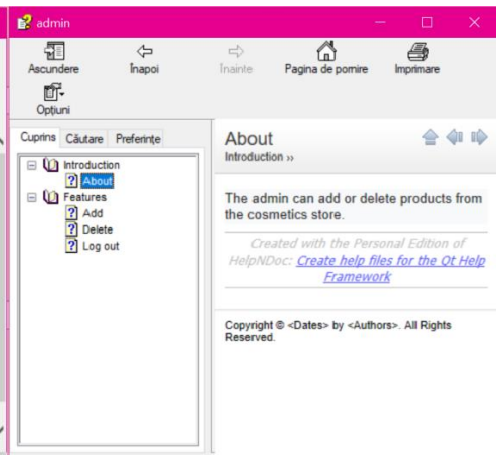


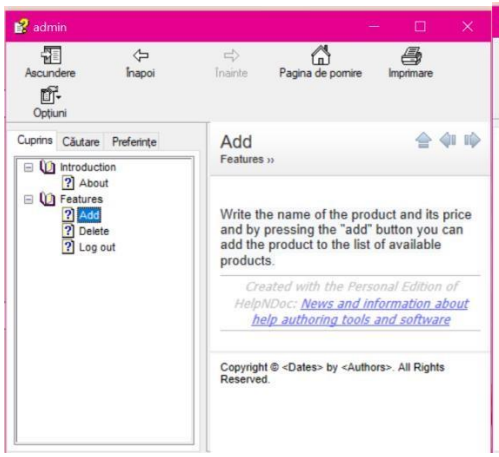


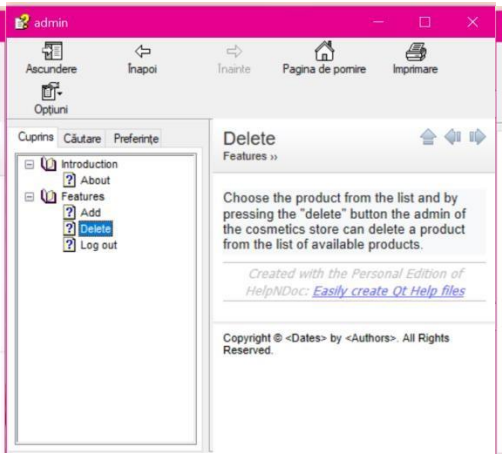


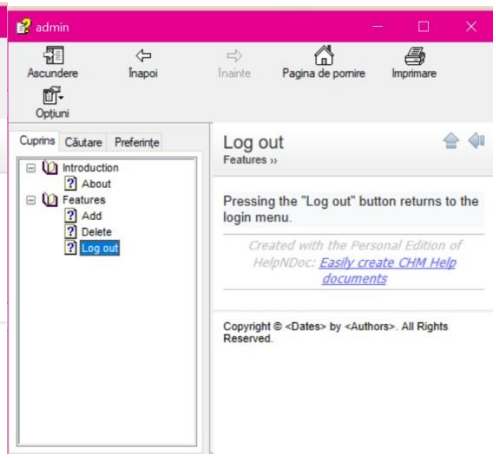












VI. Testarea unităților

În cadrul acestui capitol ne-am dorit să testăm aplicația noastră pe baza mai multor scenarii ce ar putea apărea în timpul folosirii de către utilizator.

Scenariul 1: Logarea

a) atunci când utilizatorul introduce un username corect și o parolă corectă

```
[TestMethod]
public void TestLogin_4()
{
    Proxy login = new Proxy();
    Assert.AreEqual(true, login.Login("alexandra", "alexandra"));

}
```

b) atunci când adminul introduce un username corect și o parolă corectă

```
[TestMethod]
public void TestLogin_3()
{
    Proxy login = new Proxy();
    Assert.AreEqual(true, login.Login("admin", "admin"));

}
```

a) atunci când utilizatorul introduce un username greșit sau o parolă greșită

```
[TestMethod]
public void TestLogin_5()
{
    Proxy login = new Proxy();
    Assert.AreEqual(false, login.Login("", "emilia"));

}

[TestMethod]
public void TestLogin_1()
{
    Proxy login = new Proxy();
    Assert.AreEqual(false, login.Login("", ""));

}

[TestMethod]
public void TestLogin_7()
{
    Proxy login = new Proxy();
    Assert.AreEqual(false, login.Login("1234-1", "1234-"));

}
```

b) atunci când adminul introduce un username greșit sau o parolă greșită

```
[TestMethod]
public void TestLogin_8()
{
    Proxy login = new Proxy();
    Assert.AreEqual(false, login.Login("", "ADMIN"));

}
[TestMethod]
public void TestLogin_2()
{
    Proxy login = new Proxy();
    Assert.AreEqual(false, login.Login("admin", ""));

}
```

```
[TestMethod]
public void TestLogin_10()
{
   Proxy login = new Proxy();
   Assert.AreEqual(false, login.Login(".Admin", "Admin"));

}
```

```
[TestMethod]
public void TestLogin_6()
{
   Proxy login = new Proxy();
   Assert.AreEqual(false, login.Login("adm3n..", "ADMIN"));

}
```

```
[TestMethod]
public void TestLogin_9()
{
    Proxy login = new Proxy();
    Assert.AreEqual(true, login.Login("stefania", "STEFANIA"));

}
```

Scenariul 2: Logat ca admin

a) adăugarea unui nou produs în listă

```
[TestMethod]
public void TestProdus_4()
{
    Produs produs = new Produs();
    produs.AddProdus("Blush");
}
```

```
public void TestProdus_9()
{
   Produs produs = new Produs();
   produs.AddProdus("1234");
}
```

```
[TestMethod]
public void TestProdus_6()
{
    Produs produs = new Produs();
    produs.AddProdus("");
}
```

b) ștergerea unui produs din listă

```
[TestMethod]
public void TestProdus_5()
{
    Produs produs = new Produs();
    produs.DeleteProdus("Blush");
}
```

```
[TestMethod]
public void TestProdus_10()
{
    Produs produs = new Produs();
    produs.DeleteProdus("1234");
}
```

```
[TestMethod]
public void TestProdus_7()
{
    Produs produs = new Produs();
    produs.DeleteProdus("");
}
```

Scenariul 3: Logat ca utilizator (client)

a) afișarea produselor disponible

```
[TestMethod]
public void TestProdus_3()
{
    Produs produs = new Produs();
    produs.ListAll();
}
```

```
[TestMethod]
public void TestProdus_1()
{
    Produs produs = new Produs();
    Assert.AreEqual(1, produs.ListCount() - 3);
}
```

```
[TestMethod]
public void TestProdus_2()
{
    Produs produs = new Produs();
    Assert.AreEqual(4, produs.ListCount());
}
```

VII. Anexa

a) Metoda care încarcă interfața grafică

```
  /// <summary>
  /// Metoda care incarca interfata grafica
  /// </summary>
  /// <param name="sender">Referinta catre eveniment</param>
  /// <param name="e">Instanta unui eveniment</param>
private void Form1_Load(object sender, EventArgs e)
  {
      listPanel.Add(panelLogin);
      listPanel.Add(panel);
      listPanel.Add(panelAdmin);

      listPanel[0].BringToFront();
      PopulateComboBox();
  }
```

b) Funcționalitatea butonului de Log in

```
  /// <summary>
  /// Functionalitatea butonului de login
  /// </summary>
  /// <param name="sender">Referinta catre eveniment</param>
  /// <param name="e">Instanta unui eveniment</param>
private void buttonLogin_Click(object sender, EventArgs e)
  {
      _username = textBoxUsername.Text;
      _password = textBoxPassword.Text;

      bool ok = _useradmin.Login(_username, _password);
      if (!ok)
      {
          MessageBox.Show("Login error! Verify username and password");
          return;
      }

      if (_username.Equals("admin"))
      {
          listPanel[2].BringToFront();
      }
      else
      {
          listPanel[1].BringToFront();
      }
          }
```

c) Funcționalitatea butonului de Log out din contul de admin

```
   /// <summary>
  /// Metoda pentru logout din contul de admin
  /// </summary>
  /// <param name="sender">Referinta catre eveniment</param>
  /// <param name="e">Instanta unui eveniment</param>
private void buttonIesireContAdmin_Click(object sender, EventArgs e)
  {
      listPanel[0].BringToFront();
      textBoxPassword.Text = "";
      textBoxUsername.Text = ""; }
```

d) Funcționalitatea butonului de Log out din contul de client

```
   /// <summary>
   /// Metoda pentru logout din contul de client
   /// </summary>
   /// <param name="sender">Referinta catre eveniment</param>
   /// <param name="e">Instanta unui eveniment</param>
private void buttonIesireContClient_Click(object sender, EventArgs e)
   {
       listPanel[0].BringToFront();
       textBoxPassword.Text = "";
       textBoxUsername.Text = "";
       if (textBoxProduseCos.Text.Equals(""))
       {
           listPanel[0].BringToFront();

       }
       else
       {
           MessageBox.Show("Be careful you have products in the basket!");
           return;
       }
              }
```

e) Metoda ce afișează lista cu produse

```
    /// <summary>
   /// Metoda care afiseaza lista cu produse din interfata
   /// </summary>
private void PopulateComboBox()
   {
       string[] produse = _management.ListAll();
       comboBoxSterge.Items.Clear();
       comboBoxAdaugaCos.Items.Clear();
       for (int i = 0; i < _management.ListCount(); ++i)
       {
           comboBoxSterge.Items.Add(produse[i]);
           comboBoxAdaugaCos.Items.Add(produse[i]);

       }
       comboBoxAdaugaCos.SelectedIndex = 0;
       comboBoxSterge.SelectedIndex = 0;

   }
```

f) Metoda ce populează coșul de cumpărături cu produsele selectate

```
   /// <summary>
   /// Metoda care populeaza cosul de cumparaturi cu produsele selectate
   /// </summary>
   /// <param name="sender">Referinta catre eveniment</param>
   /// <param name="e">Instanta unui eveniment</param>
private void buttonAdaugaCos_Click(object sender, EventArgs e)
   {
       string produs = (String)(comboBoxAdaugaCos.SelectedItem);
       textBoxCos.AppendText(String.Format(produs + System.Environment.NewLine));
       _count++;
       textBoxProduseCos.Text = "" + _count; }
```

g) Funcționalitatea butonului de plasare a comenzii

```
    /// <summary>
    /// Metoda care plaseaza comanda din cosul de cumparaturi
    /// </summary>
    /// <param name="sender">Referinta catre eveniment</param>
    /// <param name="e">Instanta unui eveniment</param>
private void buttonPlaseazaComanda_Click(object sender, EventArgs e)
    {
        if (textBoxCos.Text.Equals(""))
        {
            MessageBox.Show("Your cart is empty! Please add products!");
        }
        else
        {
            MessageBox.Show("Your order has been successfully processed! Thank you! ");
            Close();
        }
    }
```

h) Funcționalitatea butonului de golire a coșului de cumpărături

```
    /// <summary>
    /// Metoda care goleste cosul de cumparaturi
    /// </summary>
    /// <param name="sender">Referinta catre eveniment</param>
    /// <param name="e">Instanta unui eveniment</param>
private void buttonStergeProdus_Click(object sender, EventArgs e)
    {
        textBoxCos.Text = "";
        _count = 0;
        textBoxProduseCos.Text = "" + _count;
    }
```

i) Funcționalitatea butonului de adăugare a unui nou produs

```
    /// <summary>
    /// Metoda care populeaza lista de produse cu unele noi
    /// </summary>
    /// <param name="sender">Referinta catre eveniment</param>
    /// <param name="e">Instanta unui eveniment</param>
private void buttonAdaugaProdusNou_Click(object sender, EventArgs e)
    {
        string produsNou = textBoxAdaugaProdusNou.Text;
        if (produsNou.Equals(""))
        {
            MessageBox.Show("Enter a product!");
            return;
        }
        _management.AddProdus(produsNou);
        PopulateComboBox();
        textBoxAdaugaProdusNou.Text = "";
    }
```

j) Funcționalitatea butonului de ștergere a unui produs din listă

```
   /// <summary>
   /// Metoda care sterge un produs din lista
   /// </summary>
   /// <param name="sender">Referinta catre eveniment</param>
    /// <param name="e">Instanta unui eveniment</param>
  private void buttonSterge_Click(object sender, EventArgs e)
   {
       string produsDeSters = (String)comboBoxSterge.SelectedItem;
       _management.DeleteProdus(produsDeSters);
       PopulateComboBox();
   }
```

k) Constructorul clasei Produs care actualizează lista cu produse din fișier

```
   /// <summary>
  /// Constructorul clasei Produs care are rolul de a actualiza lista cu produse din fisier
  /// </summary>
public Produs()
   {
       try
       {
           _produse = new List<string>();
           StreamReader sr = new StreamReader(FolderProduse + "produse.txt");
           string line;
           while ((line = sr.ReadLine()) != null)
           {

               _produse.Add(line);
           }
           sr.Close();

       }
       catch (Exception exc)
       {
           Console.WriteLine(exc);
       }
   }
```

l) Metoda care returnează lista de produse cosmetice

```
    /// <summary>
   /// Metoda care returneaza lista cu numele produselor din magazinul de cosmetice
   /// </summary>
   /// <returns>Lista cu numele produselor din magazinul de cosmetice</returns>
public string[] ListAll()
   {
       string[] list = new string[_produse.Count];
       if (_produse.Count == 0)
       {
           return null;
       }
       for (int i = 0; i < _produse.Count; ++i)
       {
           list[i] = _produse[i];

       }
       return list;
   }
```

m) Metoda care actualizează lista cu produse cosmetice

```
   /// <summary>
   /// Metoda care actualizeaza lista cu produse din magazinul de cosmetice
   /// </summary>
   /// <param name="nameProdus"></param>
public void AddProdus(string nameProdus)
   {
       foreach (String st in _produse)
       {
           if (st.Equals(nameProdus))
           {
               return;
           }
       }
       _produse.Add(nameProdus);
       Save();
   }
```

n) Metoda ce salvează produsul nou adăugat și actualizează fișierul produse.txt

```
   /// <summary>
   /// Metoda care salveaza produsul nou adaugat si rescrie fisierul respectiv
   /// </summary>
public void Save()
   {
       StreamWriter wr = new StreamWriter(FolderProduse + "produse.txt");
       foreach (String st in _produse)
       {
           wr.WriteLine(st);
       }
       wr.Close();
   }
```

o) Metoda ce șterge un produs din listă

```
  /// <summary>
   /// Metoda care sterge un anumit produs din lista
   /// </summary>
   /// <param name="nameProdus"></param>
public void DeleteProdus(string nameProdus)
   {
       int i;
       for (i = 0; i < _produse.Count; ++i)
       {
           if (_produse[i] == nameProdus)
           {
               _produse.RemoveAt(i);
           }
       }
       Save();
   }
```