Gestiunea activității campusului studențesc

TUIASI

Aplicație în C#

Coordonator, Studenți,

Dumitriu Tiberius Pavăl Mihaela-Irina

Prelipcean Dragoș-Iulian

Țaga Dan-Claudiu

Grupa 1307 A

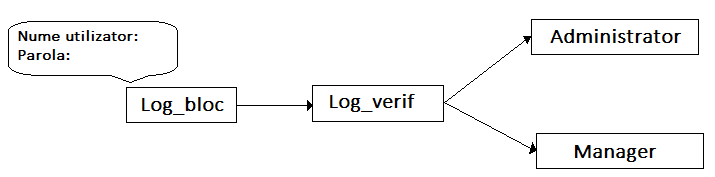
Iași, 2020

**Titlu proiect: Gestiunea activității campusului studențesc TUIASI**

Analiza, proiectarea și implementarea unei aplicații aferente care să modeleze activitatea unui campus studențesc cu privire la gestionarea studenților cazați pe perioada a 10 luni în cele 21 de cămine din campus.

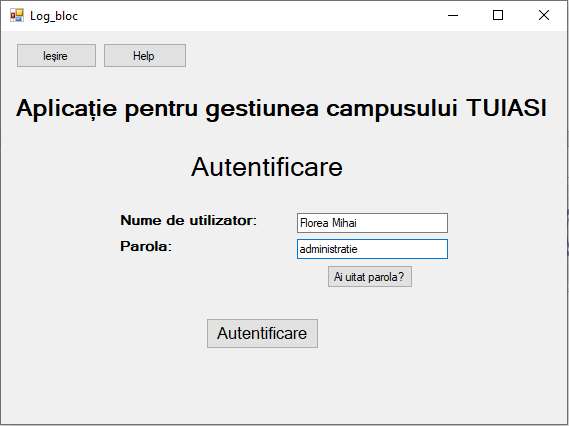
**Modul de utilizare al programului**

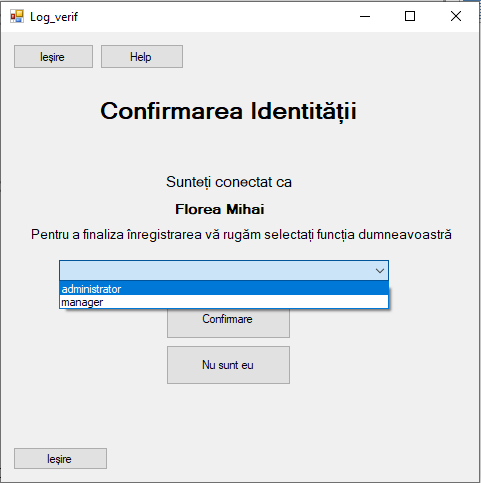
Aplicația de față este concepută după următoarea schemă de organizare împărțindu-se pe două ramuri manager respectiv administrator. Conceptual aplicația respectă următoarea schemă:

****

Împărțirea pe aceste ramuri a aplicației a fost necesară pentru că administratorul are alte drepturi în comparație cu managerul (persoană angajată în cadrul Direcției de Servicii Studențești), respectiv drepturi de modificare.

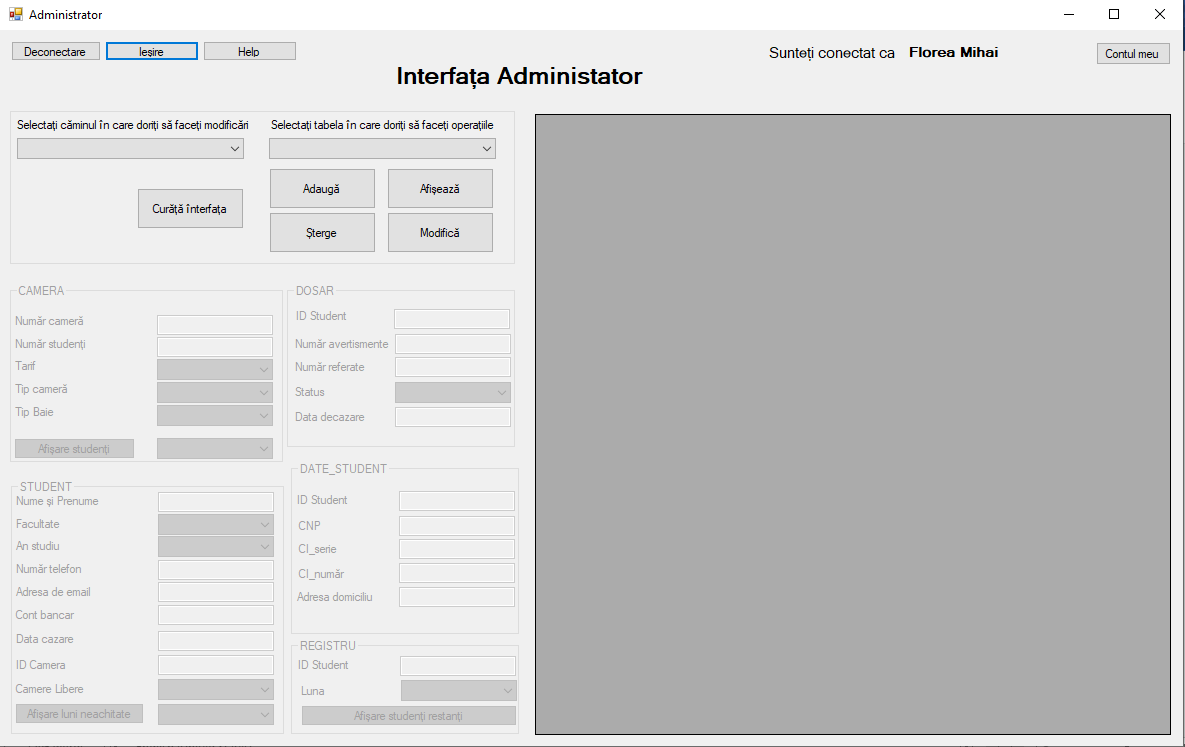
Folosind bibloteca ***System.Data.OleDb*** am creat o conexiune (***OleDbConnection***) cu baza de date specificând anumite detalii de legatură în paranteze. Folosid numele din TextBox-ul *username* se caută în baza de date (tabela **Login**) care salvează datele de logare (**username** si **parola**) și se identifică nivelul de securitate *(administrator, manager*), după care se trece la interfața de validare a identității (user-ul introduce funcția pe care acesta o deține urmând ca aceasta să fie validată sau nu printr-o conexiune nouă cu baza de date prin intermediul căreia se citește câmpul **rol** din entitatea **Login**) .



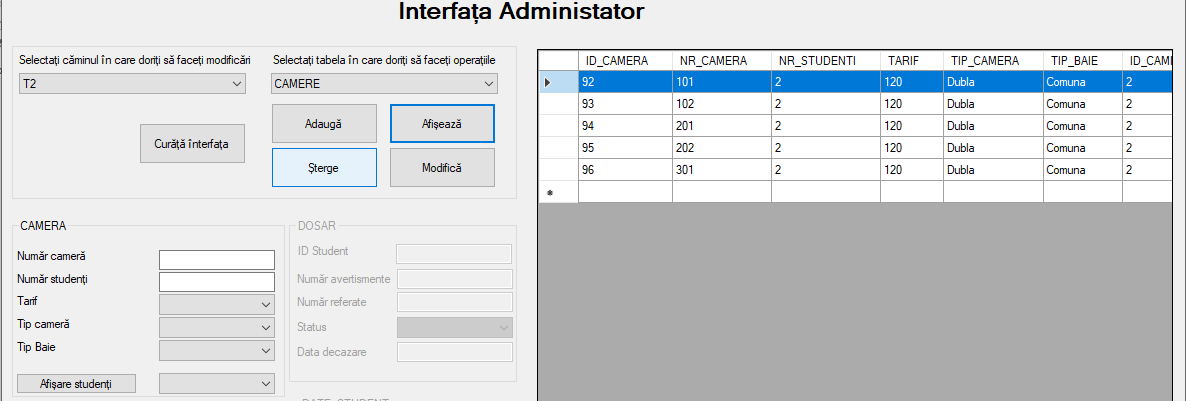


Odată stabilit tipul de utilizator, aplicația deschide fie interfața Admin, în cazul unui **administrator**, fie interfața destinată unui **manager**.

**Interfața Administrator**

 Această interfață permite introducerea, modificarea, ștergerea și afișarea datelor din baza de date aferentă aplicației selectând din aceasta tabele de interes pentru administrator (*CAMERE*, *STUDENTI*, *DATE\_STUDENTI*, *DOSARE*, *REGISTRE*). Aplicația, atât în modul administrator cât și în modul manager, este complet circulară permitând navigarea între interfețe prin intermediul butoanelor **Deconectare** și **Ieșire**.

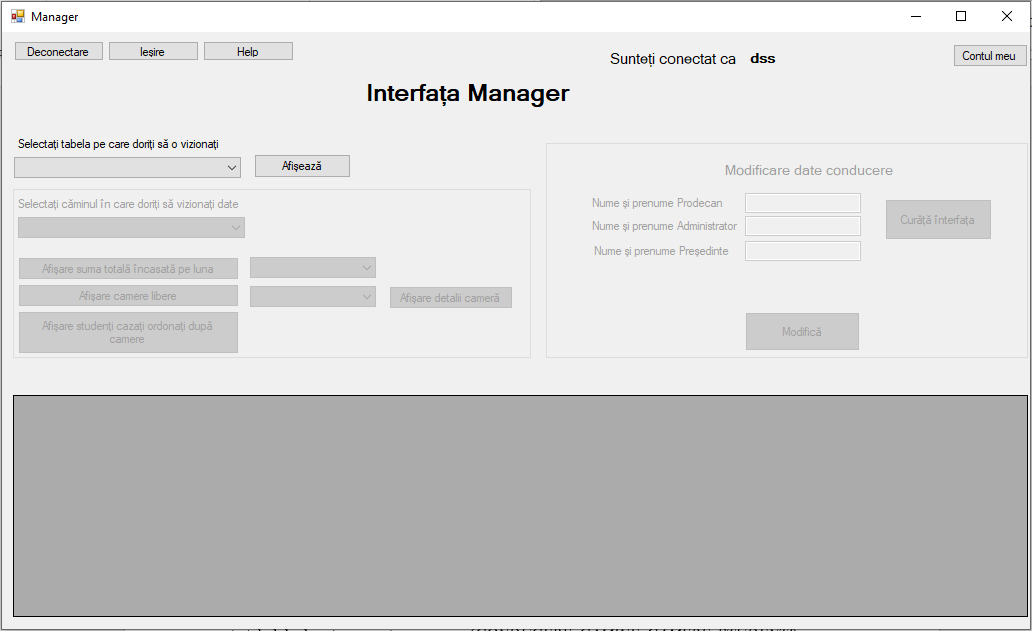
Pentru afișarea datelor s-a utilizat un control de tip **DataGridView** pentru care s-a definit următoarea funcționalitate, respectiv autocompletarea textbox-urilor (vezi mai jos) la selecția unui rând afișat (**SelectionMode=FullRowSelect**).



În funcție de căminul selectat (va fi posibilă doar selectarea unui cămin pentru care utilizatorul este administrator) se pot face modificări în baza de date, mai exact în tabela de interes selectată de utilizator. În continuare se vor prezenta funcționalitățile de care dispune utilizatorul, în plus față de cele amintite anterior, pentru fiecare tabelă în parte.

* Tabela **Camere**: utilizatorului îi este permisă vizualizarea studenților cazați în camera selectată de pe **DataGridView.**
* Tabela **Studenti**: în cazul în care utilizatorul dorește introducerea unui student într-o cameră a cărei număr maxim de studenți ce pot fi cazați a fost deja atins va primi un mesaj de eroare și se va face posibilă vizualizarea camerelor libere. De asemenea va fi posibilă și vizualizarea lunilor neachitate de studentul selectat de pe **DataGridView.** La adăugarea unui nou student se va face automat inserarea unui dosar și a unui registru corespunzător noului student, în mod similar se procedează și la ștergerea unui student.
* Tabela **Dosare**: în momentul actualizării numărului de avertismente pentru un anumit student se verifică daca este egal cu 3. În caz pozitiv, se face o creștere a numărului de referate cu 1 și o actualizare a numărului de avertismente cu 0, iar dacă se atinge un număr de 3 referate statusul studentului se modifică ca fiind *Decazat* iar data decazării respectivului student va fi *SYSDATE*. Se poate actualiza data decazării studentului la o dată dorită.
* Tabela **Registre**: se permite actualizarea achitării tarifului pentru luna selectată de utilizator. Dacă studentul nu are achitată luna anterioară, modificarea nu va avea loc, însă se va incrementa numărul de referate, moment în care se reiau verificările, respectiv actualizările, de la tabela Dosare. De asemenea este posibilă vizualizarea studenților ce nu au achitat luna selectată de utilizator.

**Interfața Manager**

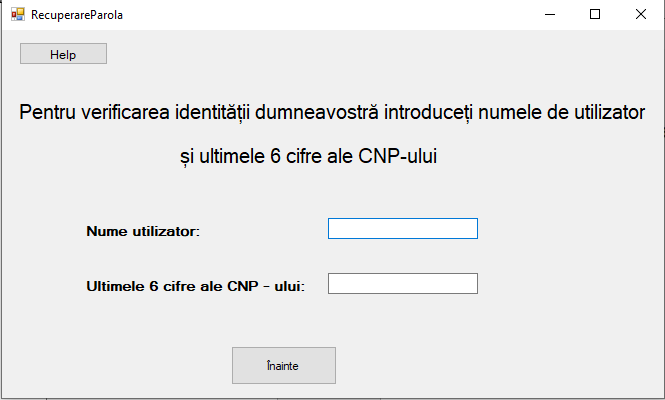


Această interfață permite afișarea datelor din baza de date aferentă aplicației selectând din aceasta tabelele de interes pentru manager (*CONDUCERE*, *CAMINE*, *CAMERE*, *STUDENTI*). Modificarea este posibilă doar pentru tabela Conducere. Pentru tabela Camine există posibilitatea de afișare a unor date utile pentru manager în funcție de căminul selectat de către utilizator. Aplicația, atât în modul administrator cât și în modul manager, este complet circulară permițând navigarea între interfețe prin intermediul butoanelor **Deconectare** și **Ieșire.**

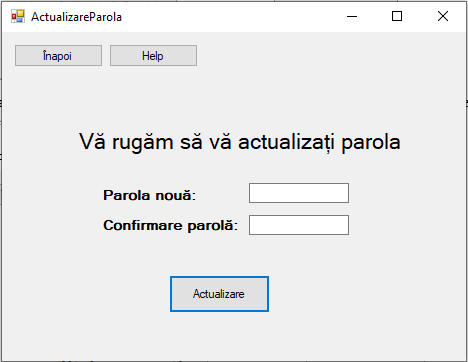
Tabela **Login** se actualizează automat la orice modificare realizată în tabela **Conducere**.

Pentru afișarea datelor s-a utilizat un control de tip **DataGridView** pentru care s-a definit funcționalitatea în aceași manieră ca la interfața Administrator.

Aplicația îi dă posibilitatea utilizatorului să iși reseteze parola în situația în care acesta nu și-o mai amintește. În fereastra de autentificare se află un buton „Ai uitat parola?”, iar la accesarea acestuia, utilizatorul este redirecționat către o pagină în care va fi nevoit să introducă numele de utilizator și ultimele 6 cifre din CNP pentru verificarea identității precum în figura următoare.

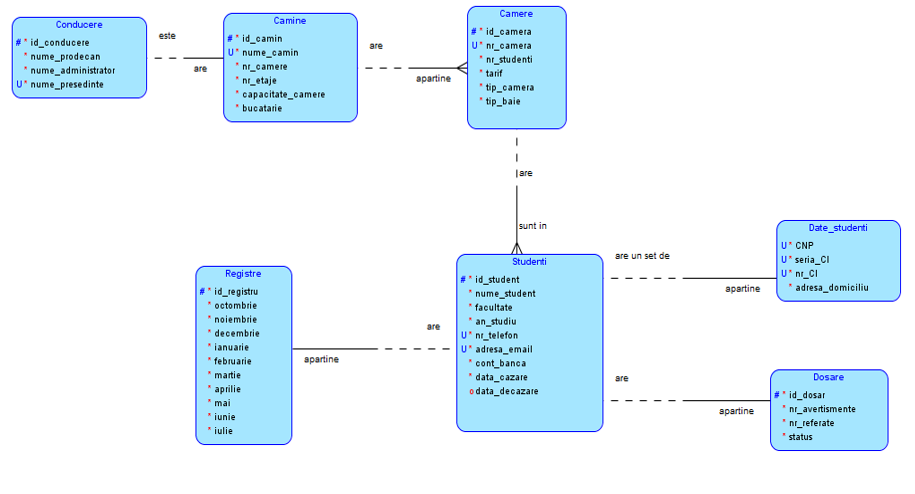


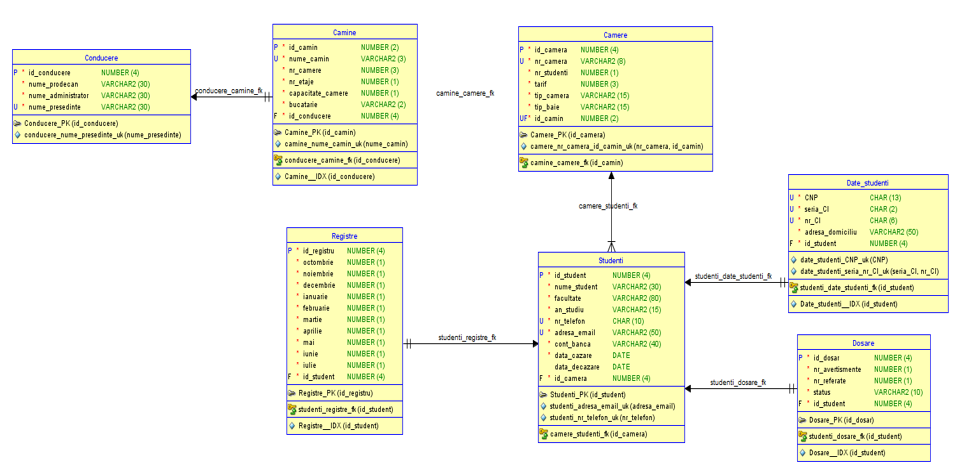
Odată realizat acest pas, va apărea posibilitatea de a introduce noua parolă și confirmarea acesteia, realizându-se astfel actualizarea parolei.



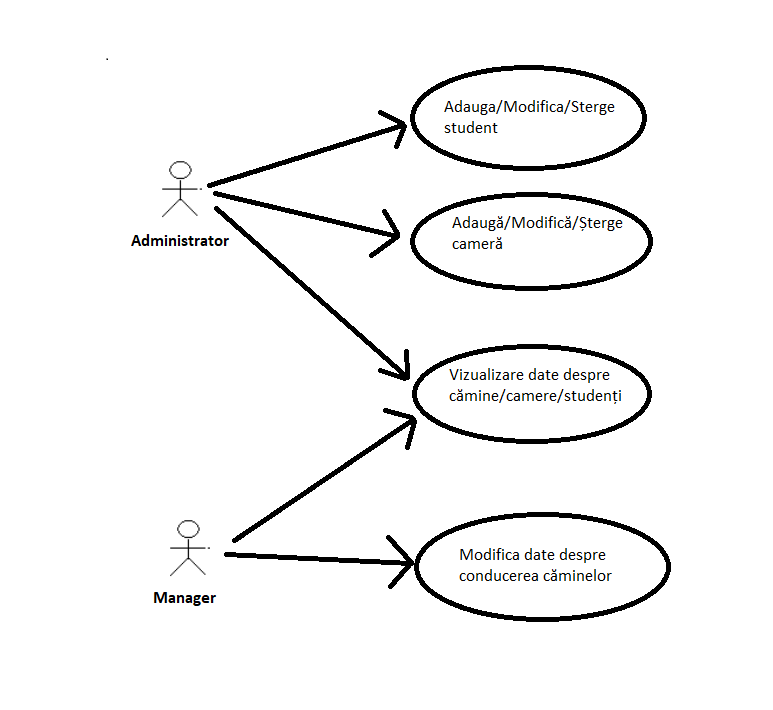
**Modelul relațional și logic**

Pentru o intelegere mai bună a aplicației se prezintă în continuare modelul relațional și logic.

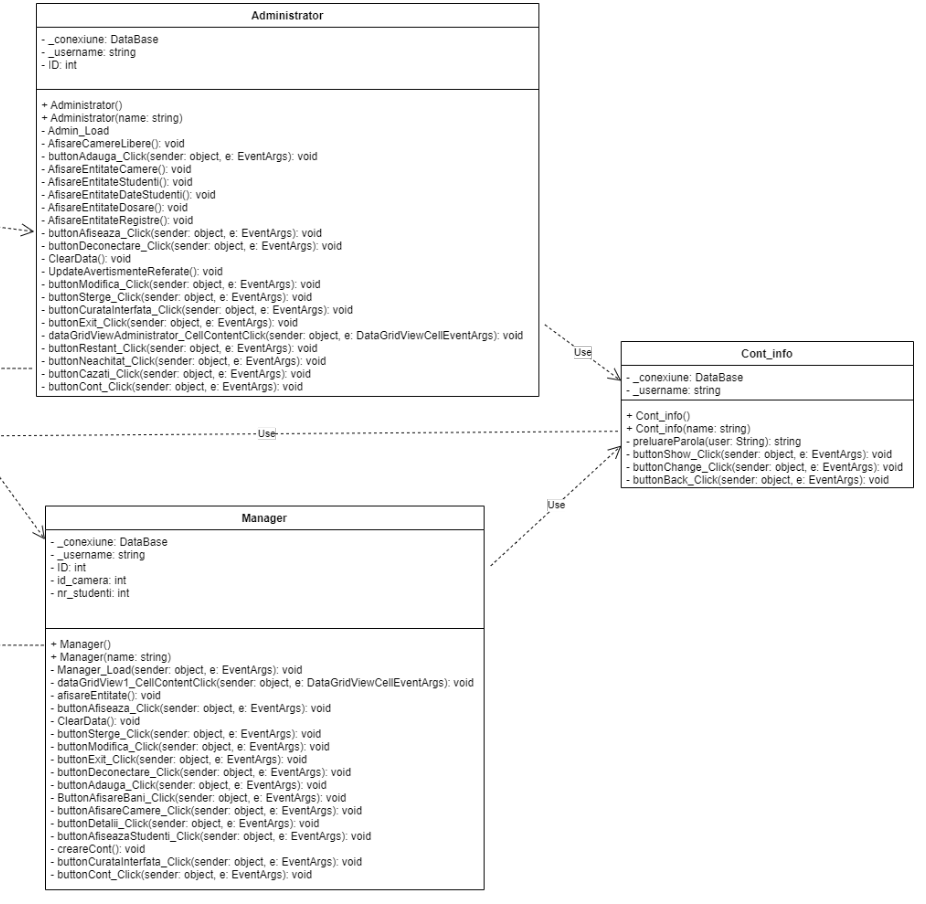
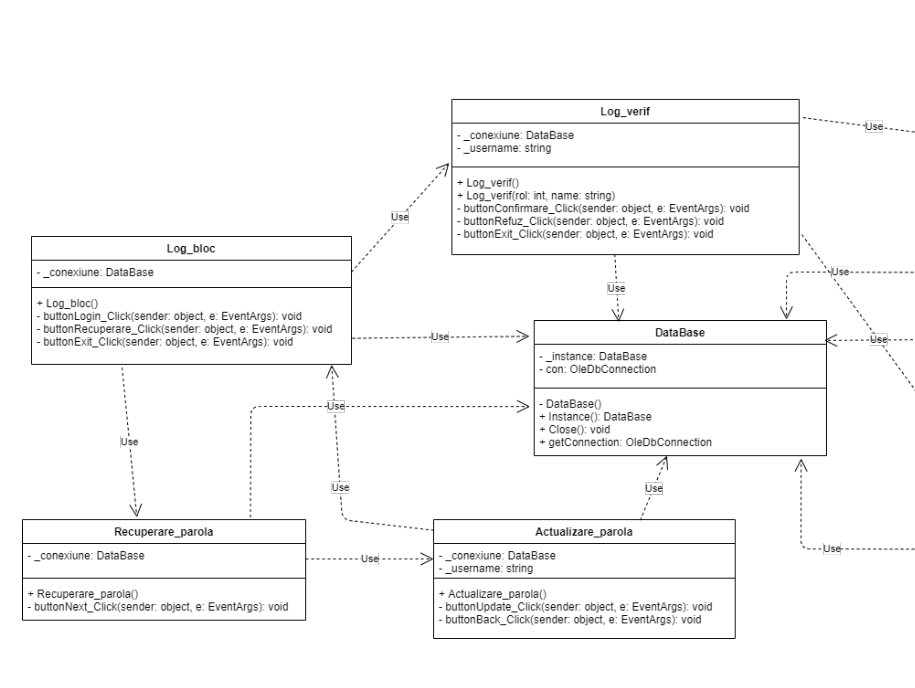




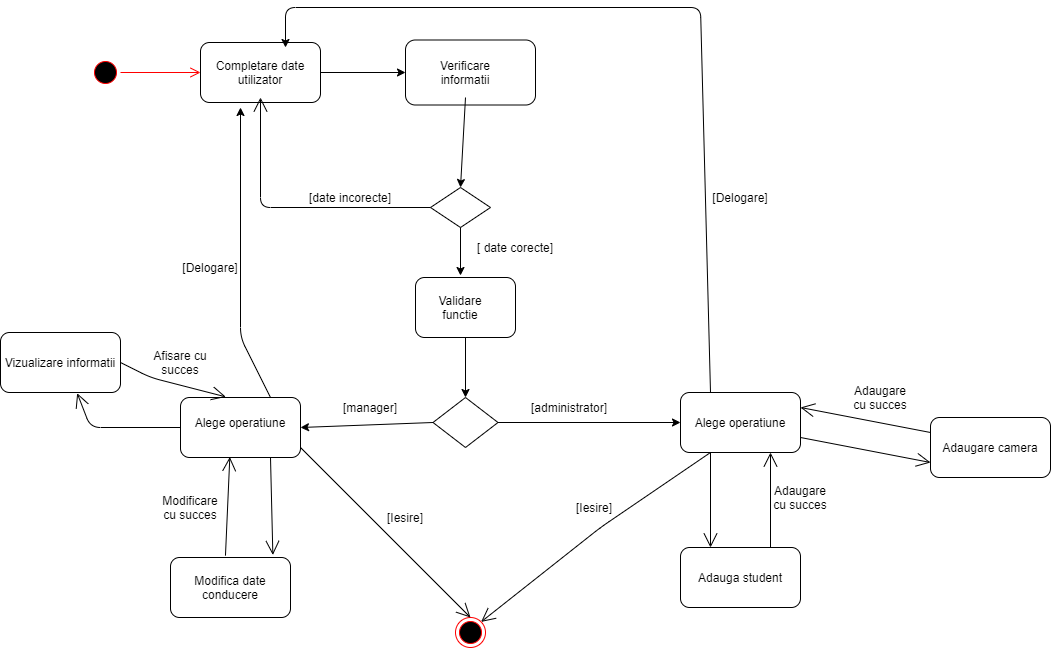
**Diagrama cazurilor de utilizare**



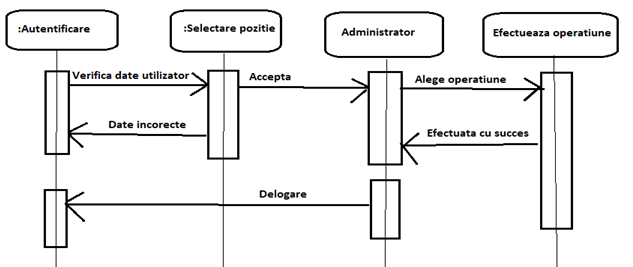
**Diagrama de clase**

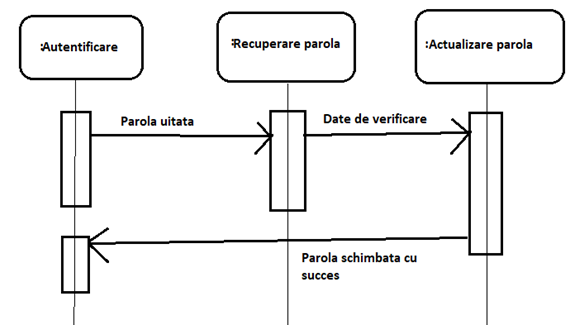


**Diagrama de activități**



**Diagrama de secvențe**





**Anexa la documentație**

**Listingul părților semnificative din cod**

Conexiunea cu baza de date:

class DataBase

{

private static DataBase \_instance;

private OleDbConnection con;

private DataBase()

{

con = new OleDbConnection("Provider=MSOLEDBSQL;Data Source=(localdb)\\MSSQLLocalDB;Catalog=IP\_Proiect;Persist Security Info=True;User ID=test;Password=test");

con.Open();

}

public static DataBase Instance()

{

if (\_instance == null)

\_instance = new DataBase();

return \_instance;

}

public void Close()

{

con.Close();

}

public OleDbConnection getConnection()

{

return con;

}

}

Interogare de tip **select** din Oracle

void AfisareEntitateCamere()

{

OleDbDataAdapter sda = new OleDbDataAdapter("SELECT \* FROM IP\_Proiect.dbo.camere WHERE id\_camin = (SELECT id\_camin FROM IP\_Proiect.dbo.camine WHERE nume\_camin = '" + comboBoxCamin.Text + "')",

\_conexiune.getConnection());

DataSet ds = new DataSet();

ds.Tables.Add(comboBoxTabele.Text);

sda.Fill(ds, comboBoxTabele.Text);

this.dataGridViewAdministrator.DataSource = ds;

this.dataGridViewAdministrator.DataMember = comboBoxTabele.Text;

}

Interogare de tip **delete** din Oracle

switch (comboBoxTabele.Text)

{

case "CAMERE":

OleDbCommand sters\_camera = new OleDbCommand("DELETE FROM CAMERE WHERE id\_camera= " + ID, \_conexiune.getConnection()); sters\_camera.Parameters.AddWithValue("@ID\_CAMERA", ID);

sters\_camera.ExecuteNonQuery();

MessageBox.Show("ACTIVITATE ÎNCHEIATĂ CU SUCCES! \n Ștergerea CAMEREI s-a realizat cu succes!");

AfisareEntitateCamere();

ClearData(); //goleste campuri

break;

Interogare de tip **insert** din Oracle

insert.CommandText = "INSERT INTO IP\_Proiect.dbo.Studenti (nume\_student, facultate, an\_studiu, nr\_telefon, adresa\_email, cont\_banca, data\_cazare, id\_camera) "

+ "VALUES('"

+ textBoxNumeStudent.Text + "', '"

+ comboBoxFacultate.Text + "', '"

+ comboBoxAnStudiu.Text + "', '"

+ textBoxNrTelefon.Text + "', '"

+ textBoxAdresaEmail.Text + "', '"

+ textBoxContBancar.Text + "', "

+ "TO\_DATE('" + textBoxDataCazare.Text + "', 'MM/DD/YYYY'), "

//+ "TO\_DATE('" + aux.ToString() + "', 'MM/DD/YYYY'), "

+ textBoxCameraStudenti.Text

+ ")";

insert.ExecuteNonQuery();

Pentru afișarea datelor s-a folosit un control de tip DataGridView :

DataGridViewRow row = dataGridViewAdministrator.Rows[e.RowIndex];

switch (comboBoxTabele.Text)

{

case "CAMERE":

textBoxNrCamera.Text = row.Cells["NR\_CAMERA"].Value.ToString();

textBoxNrStudenti.Text = row.Cells["NR\_STUDENTI"].Value.ToString();

comboBoxTarif.Text = row.Cells["TARIF"].Value.ToString();

comboBoxTipCamera.Text = row.Cells["TIP\_CAMERA"].Value.ToString();

comboBoxTipBaie.Text = row.Cells["TIP\_BAIE"].Value.ToString();

ID = Convert.ToInt32(row.Cells["ID\_CAMERA"].Value.ToString());

break;

case "STUDENTI":

textBoxNumeStudent.Text = row.Cells["NUME\_STUDENT"].Value.ToString();

comboBoxFacultate.Text = row.Cells["FACULTATE"].Value.ToString();

comboBoxAnStudiu.Text = row.Cells["AN\_STUDIU"].Value.ToString();

textBoxNrTelefon.Text = row.Cells["NR\_TELEFON"].Value.ToString();

textBoxAdresaEmail.Text = row.Cells["ADRESA\_EMAIL"].Value.ToString();

textBoxContBancar.Text = row.Cells["CONT\_BANCA"].Value.ToString();

textBoxDataCazare.Text = row.Cells["DATA\_CAZARE"].Value.ToString();

textBoxCameraStudenti.Text = row.Cells["ID\_CAMERA"].Value.ToString();

ID = Convert.ToInt32(row.Cells["ID\_STUDENT"].Value.ToString());

break;

case "DATE\_STUDENTI":

textBoxNumeDateStudent.Text = row.Cells["ID\_STUDENT"].Value.ToString();

textBoxCnp.Text = row.Cells["CNP"].Value.ToString();

textBoxCiSerie.Text = row.Cells["SERIA\_CI"].Value.ToString();

textBoxCiNumar.Text = row.Cells["NR\_CI"].Value.ToString();

textBoxAdresa.Text = row.Cells["ADRESA\_DOMICILIU"].Value.ToString();

ID = Convert.ToInt32(row.Cells["ID\_STUDENT"].Value.ToString());

break;

}

Pentru a valida funcționalitatea implementarilor am creat o serie de cazuri de testare în care am încercat diferite abordări.

Cazuri de teste:

[TestMethod]

public void Test\_Conectare\_manager()

{

var var\_login = new Log\_bloc();

var createControl = var\_login.GetType().GetMethod("CreateControl", BindingFlags.Instance | BindingFlags.NonPublic);

createControl.Invoke(var\_login, new object[] { true });

var input\_nume = "dss";

var input\_status = " ";

var rez = "administrator";

var test = var\_login.getUtilizatorBD(input\_nume, input\_status);

Assert.AreEqual(rez, test);

}

[TestMethod]

public void Test\_Conectare\_Florea\_Mihai()

{

var var\_login = new Log\_bloc();

var createControl = var\_login.GetType().GetMethod("CreateControl", BindingFlags.Instance | BindingFlags.NonPublic);

createControl.Invoke(var\_login, new object[] { true });

var input\_nume = "Florea Mihai";

var input\_status = " ";

var rez = "administrator";

var test = var\_login.getUtilizatorBD(input\_nume, input\_status);

Assert.AreEqual(rez, test);

}

[TestMethod]

[Ignore]

public void Test\_Conectare\_Cenzor\_Username() //va esua intodeauna

{

var var\_login = new Log\_bloc();

var createControl = var\_login.GetType().GetMethod("CreateControl", BindingFlags.Instance | BindingFlags.NonPublic);

createControl.Invoke(var\_login, new object[] { true });

var rez = "Florea Mihai";

var test = var\_login.getUtilizator();

Assert.AreEqual(rez, test);

}