Universitatea Ștefan cel Mare Suceava

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Programarea Interfețelor Utilizator

Proiect

Cazare Studenți

Student: Beilic Maria

Specializarea: Calculatoare

Grupa: 3122B

An: II

2019/2020

Cuprins

I. Tema proiectului

Tema proiectului o reprezintă un sistem de cazare al studenților în cămin. Acesta suprinde două perspective, student și administrator. Aplicația oferă posibilitatea administratorului de a executa operații de tip inserare a studenților în listă, căutarea, modificarea si ștergerea acestora utilizând fișiere text. Proiectul se numește „Cazare Studenți” și este realizat în Visual Studio 2019 utilizând limbajul de programare C#. Scopul aplicației este organizarea mai bună a procedeului de cazare a studenților în cămine.

II. Descrierea proiectului

1. Student

Pentru a accesa aplicația din punctul de vedere al studentului avem nevoie de un utilizator “maria” și de o parola “student2020”. Pentru versiunea aceasta am ales să adaug opțiunile în limba română și limba engleză. Dacă utilizatorul introduce alte date, greșite, va primi un mesaj de avertizare.

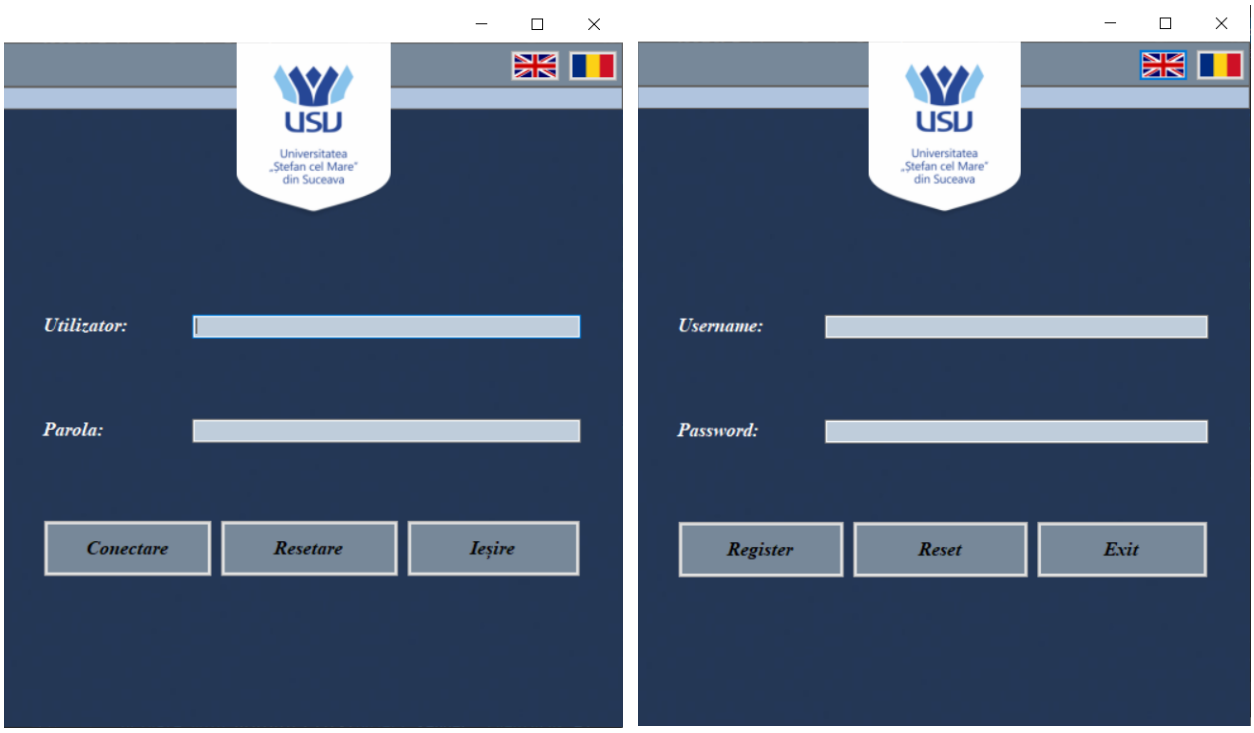

Figura II.1 Formă de autentificare

Studentul poate accesa lista cu persoanele care au optat pentru cazarea în căminele USV și îndeplinesc condițiile de cazare și informații despre cămine.

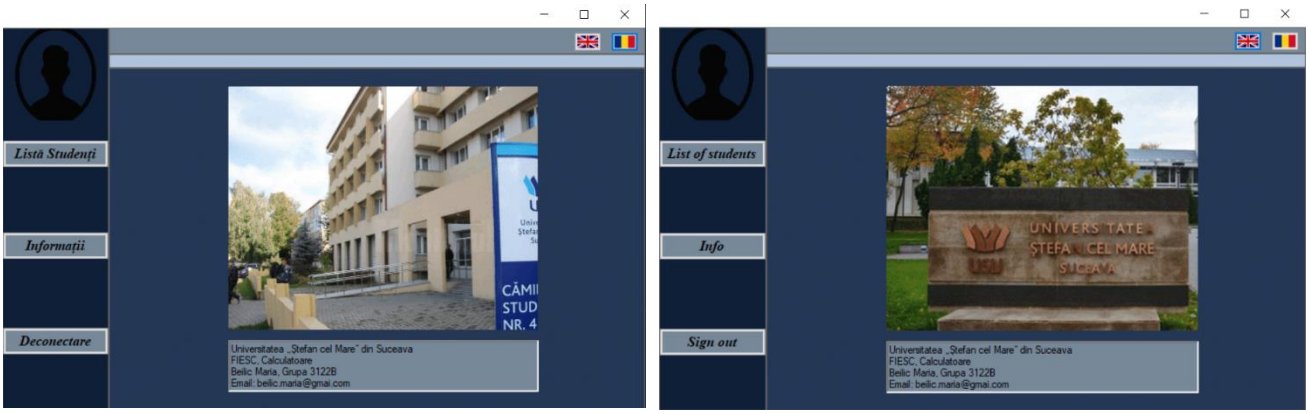

Figura II.2 Formă intermediară pentru utilizator

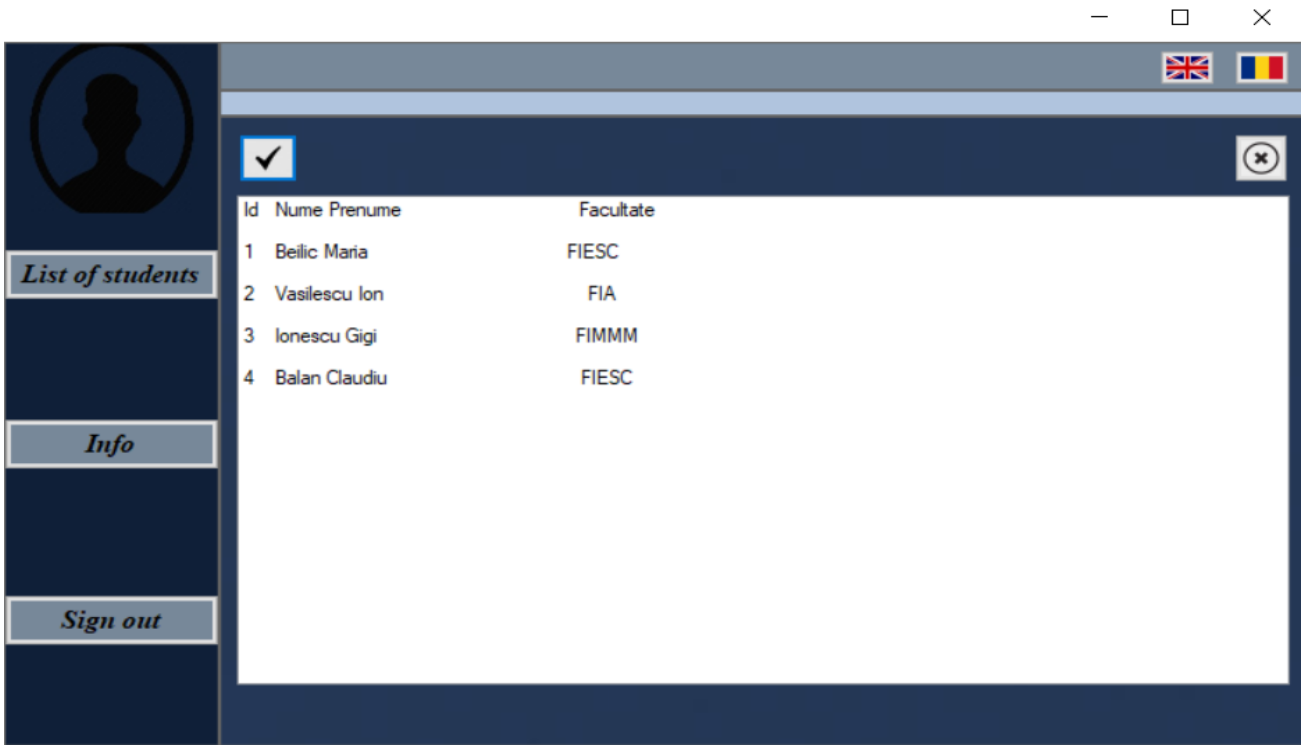

Figura II.3 Lista studenților ce îndeplinesc criteriile de cazare

În secțiunea “Informații” se încarcă dinamic detaliile despre fiecare cămin. Imaginile se regăsesc în folder-ul proiectului. La final, utilizatorul poate să se deconecteze fiind trimis pe forma inițială de autentificare.

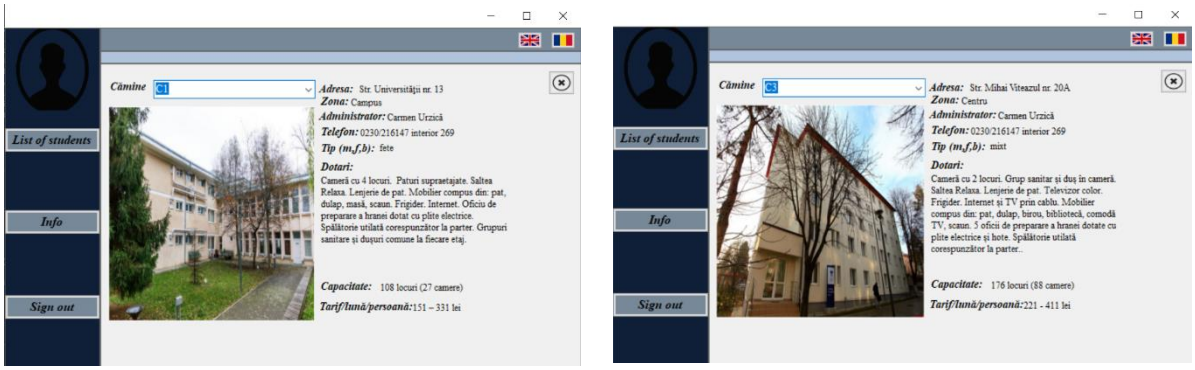

Figura II.4 Informații despre cămine

2. Administrator

Pentru a accesa aplicația din punctul de vedere al administratorului avem nevoie de un utilizator “admin” și de o parola “2020”. Dacă utilizatorul introduce alte date, greșite, va primi un mesaj de avertizare. Forma este prezentată în Figura II.1.

Administratorul poate alege înregistrarea sau căutarea studenților, opțiuni care includ alte posibilități în cadrul formelor aferente.

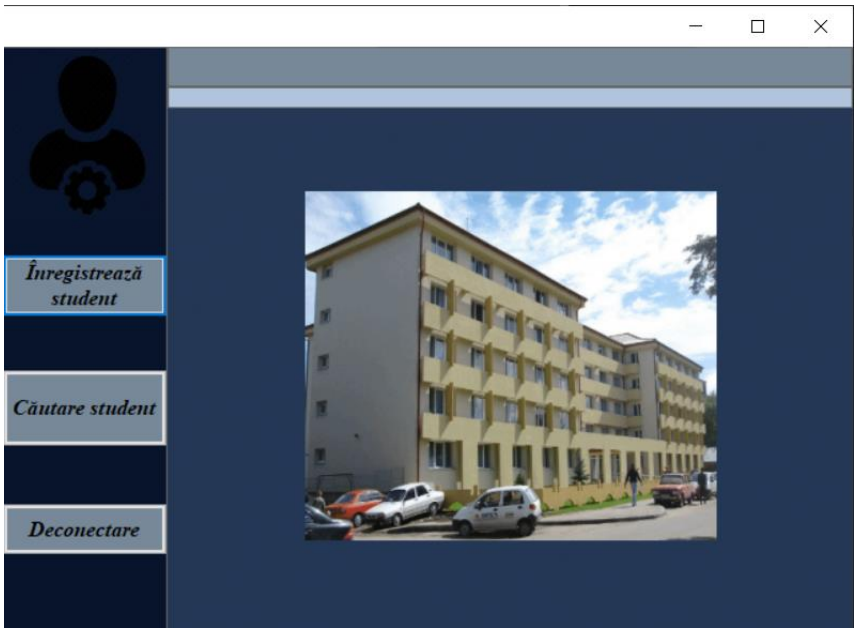

Figura II.5 Opțiunile administratorului

În forma asociată înregistrării unui utilizator, putem afișa lista cu toți studenți înregistrați. Punctajul de cazare se poate afișa prin selectarea unui student din listă și acționarea butonului “Punctaj”.

Pentru adăugarea unui student în lista de așteptare s-au solicitat următoarele date prezentate în Figura II.6 : nume, prenume, distanța dintre domiciliu și campus, programul de studii, anul universitar, facultatea, opțiunile pentru cămine și numărul de studenți în cameră.

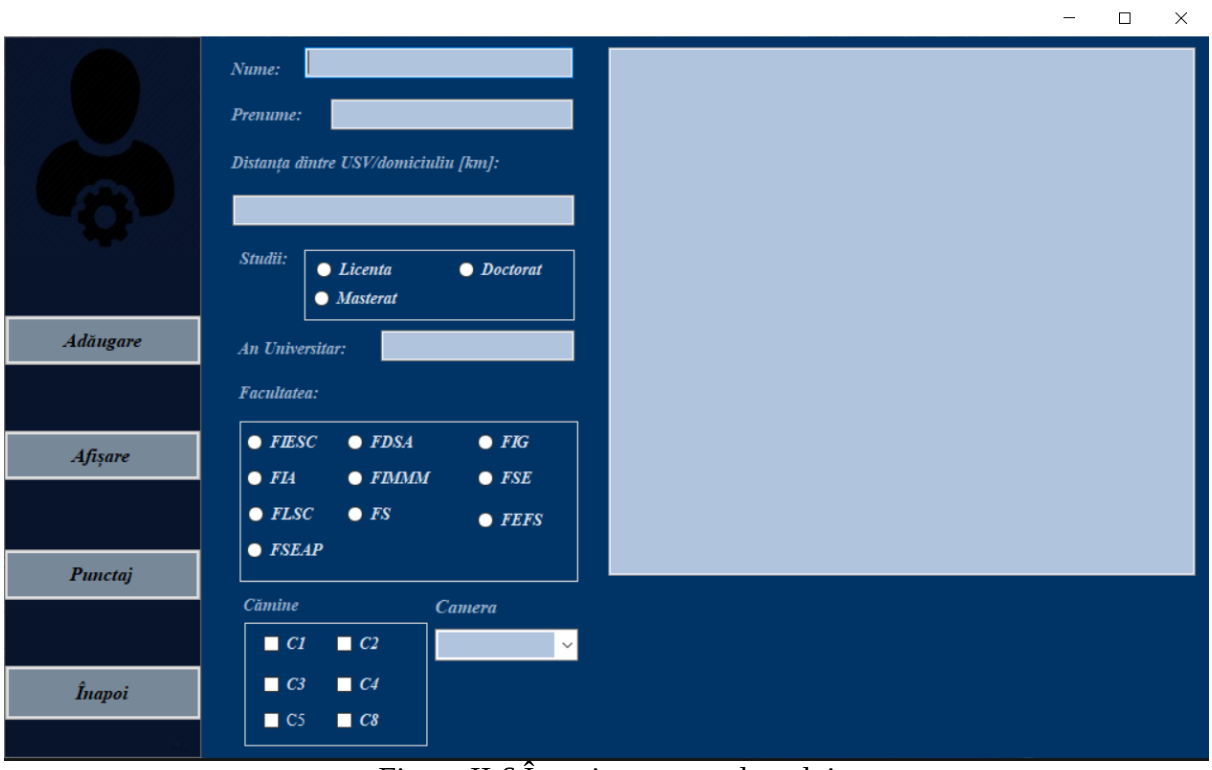

Figura II.6 Înregistrarea studentului

În primul rând, la acționarea butonului „Adăugare”, se resetează culorile etichetelor, se verifică datele introduse și dacă acestea sunt corecte se creează un obiect nou, de tip StudentCamin ce primește ca parametrii numele, prenumele, distanța și anul universitar. Proprietățile auto-implemented sunt setate cu valorile introduse. Cu ajutorul „contractului” se adauga studentul trimis ca parametru în fișier. Dacă studentul a fost adaugat, administratorul primește un mesaj de confirmare. Dacă datele sunt incorecte, etichetele asociate câmpurilor greșite își vor schimba culoarea și se va afișa un mesaj de avertisment.

```
private void BtnAdauga_Click(object sender, EventArgs e)
{
    ResetCuloareEtichete();
    CodEroare codValidare = Validare(txtNume.Text, txtPrenume.Text, txtDistanta.Text, txtAnUniversitar.Text);
    if (codValidare != CodEroare.CORECT)
    {
        MarcheazaControaleCuDateIncorecte(codValidare);
        MessageBox.Show("Trebuie sa verificati datele marcate cu rosu! Informatia lipseste sau a fost introdusa incorect.");
    }
    else
    {
        StudentCamin s = new StudentCamin(txtNume.Text, txtPrenume.Text, Convert.ToInt32(txtDistanta.Text), Convert.ToInt32(txtAnUniversitar.Text));


        s.ProgramSTD = GetSelectetFacultate();
        s.StudSTD = GetSelectedStudii();
        //set Discipline
        s.Camine = new ArrayList();
        s.Camine.AddRange(camineSelectate);
        s.Camera = Convert.ToInt32(cmbCamera.Text);

        adminStudenti.AddStudent(s);
        lblAdauga.Text = "Studentul a fost adaugat";

        //resetarea controalelor pentru a introduce datele unui student nou
        ResetareControale();

    }
}
```

Figura II.7 Butonul de adăugare

Adăugarea în fișier este prezentată în Figura II.8.

```
public void AddStudent(StudentCamin student)
{
    student.IdStudent = GetId();
    try
    {
        //instructiunea 'using' va apela la final swFisierText.Close();
        //al doilea parametru setat la 'true' al constructorului StreamWriter indica modul 'append' de deschidere al fisierului
        using (StreamWriter swFisierText = new StreamWriter(NumeFisier, true))
        {

            swFisierText.WriteLine(student.ConversieLaSir_PentruFisier());
        }
    }
    catch (IOException eIO)
    {
        throw new Exception("Eroare la deschiderea fisierului. Mesaj: " + eIO.Message);
    }
    catch (Exception eGen)
    {
        throw new Exception("Eroare generica. Mesaj: " + eGen.Message);
    }
}
```

Figura II.8 Adăugarea în fișier

La acționarea butonului „Afișare” se golește ListBox-ul, se introduce un antet ca linie principală ce conține id-ul, numele, prenumele, distanța și facultatea. Se creează o listă nouă de tipul StudentCamin, ce primește studenți din fișier. Prin parcurgerea listei se introduce câte o linie pentru fiecare student în ListBox. Preluarea datelor din fișier este prezentată în Figura II.10.

```
private void BtnAfisare_Click(object sender, EventArgs e)
{
    lBoxAfisare.Items.Clear();
    var antetTabel = String.Format("{0,-5}{1,-35}{2,20}{3,10}\n", "Id", "Nume Prenume", "Distanta", "Facultate");
    lBoxAfisare.Items.Add(antetTabel);

    List<StudentCamin> studenti = adminStudenti.GetStudenti();
    foreach (StudentCamin s in studenti)
    {
        var linieTabel = String.Format("{0,-5}{1,-35}{2,20}{3,10}\n", s.IdStudent, s.NumeComplet, s.Distanta, s.ProgramSTD.ToString());
        lBoxAfisare.Items.Add(linieTabel);
    }
}
```

Figura II.9 Butonul de afișare

```
public List<StudentCamin> GetStudenti()
{
    List<StudentCamin> studenti = new List<StudentCamin>();

    try
    {
        // instructiunea 'using' va apela sr.Close()
        using (StreamReader sr = new StreamReader(NumeFisier))
        {
            string line;

            //citeste cate o linie si creaza un obiect de tip Student pe baza datelor din linia citita, repta pana ajunge la finalul documentului
            while ((line = sr.ReadLine()) != null)
            {
                StudentCamin s = new StudentCamin(line);
                studenti.Add(s);
            }
        }
    }
    catch (IOException eIO)
    {
        throw new Exception("Eroare la deschiderea fisierului. Mesaj: " + eIO.Message);
    }
    catch (Exception eGen)
    {
        throw new Exception("Eroare generica. Mesaj: " + eGen.Message);
    }

    return studenti;
}
```

Figura II.10 Preluarea datelor din fișier

La acționarea butonului „Punctaj” se verifică datele introduse și dacă acestea sunt corecte se creează un obiect nou, de tip StudentCamin ce primește ca parametrii numele, prenumele, distanța și anul universitar. Prin noul obiect se apelează propietatea punctaj asociată unui obiect din clasa StudentCamin. Se resetează controalele și culorile etichetelor.

```
private void BtnCazare_Click(object sender, EventArgs e)
{
    CodEroare codValidare = Validare(txtNume.Text, txtPrenume.Text, txtDistanta.Text, txtAnUniversitar.Text);
    if (codValidare != CodEroare.CORECT)
    {
        MarcheazaControaleCuDateIncorecte(codValidare);
        MessageBox.Show("Selectati un student din lista!");
    }
    else
    {
        StudentCamin s = new StudentCamin(txtNume.Text, txtPrenume.Text, Convert.ToInt32(txtDistanta.Text), Convert.ToInt32(txtAnUniversitar.Text));
        lblCazare.Text = s.Punctaj();
        ResetareControale();
        ResetCuloareEtichete();
    }
}
```

Figura II.11 Butonul pentru punctaj

Prin acționarea butonului „Înapoi”, administratorul se va întoarce în forma intermediară prezentată în Figura II.5.

În forma asociată căutării unui student, putem salva lista studenților ce au îndeplinit crtiteriile necesare cazării. Căutarea unui student se face pe baza numelui și a prenumelui. După căutare putem opta pentru modificarea datelor unui student (anul universitar sau numărul de studenți în cameră) sau ștergerea unui student.

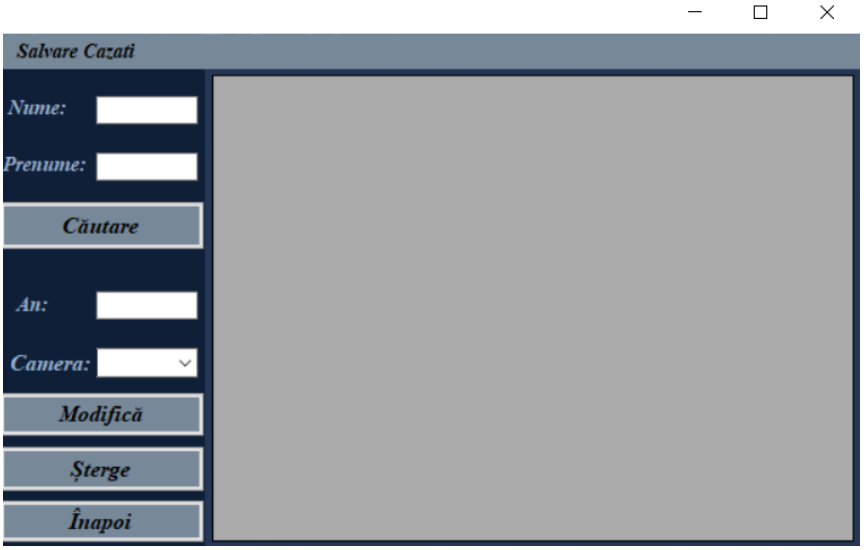

Figura II.12 Căutarea studentului

La acționarea controlului de tip MenuStrip „Salvare Cazați” se execută următori pași: se creează o listă nouă cu elemente de tipul StudentCamin care primește prin ajutorul „contractului” o listă de studenți ce indeplinesc condițiile de cazare. Preluarea studenților cazați este prezentată în Figura II.14. Pentru a scrie un fișier de tip text se utilizează o structură de tratare a excepțiilor. În blocul try sunt incluse instrucțiunile ce trebuie verificate pentru apariția excepțiilor. Blocurile catch pot intercepta posibilele excepții.

```
private void salvareCazati_Click(object sender, EventArgs e)
{

    List<StudentCamin> studenti_cazati = adminStudenti.GetStudentiCazati();
    saveFile.ShowDialog();
    string numeFisier = saveFile.FileName;

    try
    {
        //instructiunea 'using' va apela la final swFisierText.Close();
        //al doilea parametru setat la 'true' al constructorului StreamWriter indica modul 'append' de deschidere al fisierului
        using (StreamWriter swFisierText = new StreamWriter(numeFisier, true))
        {
            foreach (StudentCamin s in studenti_cazati)
                swFisierText.WriteLine(s.ConversieLaSir());
        }
    }
    catch (IOException eIO)
    {
        throw new Exception("Eroare la deschiderea fisierului. Mesaj: " + eIO.Message);
    }
    catch (Exception eGen)
    {
        throw new Exception("Eroare generica. Mesaj: " + eGen.Message);
    }

}
```

Figura II.13 Salvarea într-un fisier text

```
public List<StudentCamin> GetStudentiCazati()
{
    List<StudentCamin> studenti = GetStudenti();
    List<StudentCamin> studenti_cazati = new List<StudentCamin>();
    foreach (StudentCamin s in studenti)
    {
        if(s.Distanta>= MINIM_CAZARE)
        {
            studenti_cazati.Add(s);
        }
    }
    return studenti_cazati;
}
```

Figura II.14 Preluarea datelor filtrate

```
private void BtnCautare_Click(object sender, EventArgs e)
{
    contorcautare = 1;
    CodEroare codValidare = Validare_Cautare(txtNume.Text, txtPrenume.Text);
    ResetCuloareEtichete_Cauta();
    if (codValidare == CodEroare.CORECT)
    {
        StudentCamin s = adminStudenti.GetStudent(txtNume.Text, txtPrenume.Text);
        if (s != null)
        {
            if (s.Nume == txtNume.Text && s.Prenume == txtPrenume.Text)
            {
                lblMesaj.Text = string.Empty;
                lblMesaj.Text = "Studentul a fost gasit!";
                List<StudentCamin> stud = adminStudenti.GetStudenti();
                AdaugaStudentInControlDataGridView(stud);
            }
            else
            {
                lblMesaj.Text = string.Empty;
                lblMesaj.Text = "Studentul nu a fost găsit!";
                dataGridStudent.DataSource = null;
            }
            if (txtNume.Enabled == true && txtPrenume.Enabled == true)
            {
                txtNume.Enabled = false;
                txtPrenume.Enabled = false;
            }
            else
            {
                txtNume.Enabled = true;
                txtPrenume.Enabled = true;
            }
        }

        else
        {
            lblMesaj.Text = string.Empty;
            lblMesaj.Text = "Studentul nu a fost găsit!";
        }
    }
    else
        {
        MarcheazaControaleCuDateIncorecte(codValidare);
        lblMesaj.Text = "Introduceti date valide!";
        }
}
```

Figura II.15 Căutarea studentului

La acționarea butonului „Căutare” se execută următori pași: se creează un obiect de tip StudentCamin care primește studentul ce corespunde datelor introduse de administrator, se resetează culorile etichetelor, și se validează datele. Dacă datele sunt corecte se afișează lista cu toți studenți în DataGridView și un mesaj ce confirmă găsirea studentului, dacă datele sunt incorecte se afișează un mesaj ce anunța utilizatorul și etichetele asociate datelor greșite își schimbă culoarea. Preluarea studentului căutat din fișier este prezentată în Figura II.16.

```
public StudentCamin GetStudent(string nume, string prenume)
{
    try
    {
        // instructiunea 'using' va apela sr.Close()
        using (StreamReader sr = new StreamReader(NumeFisier))
        {
            string line;

            //citeste cate o linie si creaza un obiect de tip Student pe baza datelor din linia citita
            while ((line = sr.ReadLine()) != null)
            {
                StudentCamin student = new StudentCamin(line);
                if (student.Nume.Equals(nume) && student.Prenume.Equals(prenume))
                    return student;
            }
        }
    }
    catch (IOException eIO)
    {
        throw new Exception("Eroare la deschiderea fisierului. Mesaj: " + eIO.Message);
    }
    catch (Exception eGen)
    {
        throw new Exception("Eroare generica. Mesaj: " + eGen.Message);
    }
    return null;
}
```

Figura II.16 Preluarea studentului căutat din fișier

La acționarea butonului „Modifică” se verifică dacă a fost căutat un student, dacă această condiție este îndeplinită se trece la procesul de modificare. Procesul de modificare începe cu resetarea culorilor etichetelor urmată de validarea datelor. Dacă datele sunt incorecte se va afișa un mesaj de avertizare si etichetele corespunzătoare datelor incorecte își vor schimba culoarea. Pentru date corecte se creează un obiect de tip StudentCamin ce primește studentul căutat (Figura II.16), se preiau datele ce doresc a fi modificate și se rescrie fișierul cu datele actualizate (Figura II.18). După modificare se actualizează lista în DataGridView și administratorul este anunțat printr-un mesaj că acțiunea a fost executată cu succes. Dacă nu a fost căutat un student, utilizatorul va fi anunțat că această acțiune este necesară pentru a continua.

```
private void btnModifica_Click(object sender, EventArgs e)
{
    if (contorcautare == 1)
    {
        ResetCuloareEtichete_Modifica();
        CodEroare codValidare = Validare_Modificare(txtAn.Text);
        if (codValidare != CodEroare.CORECT)
        {
            MarcheazaControaleCuDateIncorecte(codValidare);
            MessageBox.Show("Trebuie sa verificati datele marcate cu rosu!\nInformatia lipseste sau a fost introdusa incorect.");
        }
        else
        {
            StudentCamin s = adminStudenti.GetStudent(txtNume.Text, txtPrenume.Text);
            if (s != null)
            {
                s.GetAn(txtAn.Text);
                s.Camera = int.Parse(cmbCamera.Text);
                s.DataActualizata = DateTime.Now;
                adminStudenti.UpdateStudent(s);
                List<StudentCamin> stud = adminStudenti.GetStudenti();
                AdaugaStudentInControlDataGridView(stud);
                lblModifica.Text = "Modificare efectuata";
                txtNume.Enabled = true;
                txtPrenume.Enabled = true;
            }
            else
            {
                lblModifica.Text = "Student inexistent!";
            }
        }
    }
    else
    {
        MessageBox.Show("Trebuie sa cautati un student!");
    }
    ResetareControale();
    contorcautare = 0;
}
```

Figura II.17 Modificarea datelor

```
public bool UpdateStudent(StudentCamin studentActualizat)
{
    List<StudentCamin> studenti = GetStudenti();

    bool actualizareCuSucces = false;
    try
    {
        //instructiunea 'using' va apela la final swFisierText.Close();
        //al doilea parametru setat la 'false' al constructorului StreamWriter indica modul 'overwrite' de deschidere al fisierului
        using (StreamWriter swFisierText = new StreamWriter(NumeFisier, false))
        {
            foreach (StudentCamin stud in studenti)
            {
                //informatiile despre studentul actualizat vor fi preluate din parametrul "studentActualizat"
                if (stud.IdStudent != studentActualizat.IdStudent)
                {
                    swFisierText.WriteLine(stud.ConversieLaSir_PentruFisier());
                }
                else
                {
                    swFisierText.WriteLine(studentActualizat.ConversieLaSir_PentruFisier());


                }
            }
            actualizareCuSucces = true;
        }
    }
    catch (IOException eIO)
    {
        throw new Exception("Eroare la deschiderea fisierului. Mesaj: " + eIO.Message);
    }
    catch (Exception eGen)
    {
        throw new Exception("Eroare generica. Mesaj: " + eGen.Message);
    }

    return actualizareCuSucces;
}
```

Figura II.18 Rescrierea fișierului cu datele actualizate

La acționarea butonului „Șterge” se verifică dacă a fost căutat un student, dacă această condiție este îndeplinită se trece la procesul de ștergere. Se creează un student ce primește studentul căutat, dacă studentul există în listă vom rescrie fișierul fără acesta (Figura II.20). După ștergere se actualizează lista în DataGridView și administratorul este anunțat printr-un mesaj că acțiunea a fost executată cu succes.. Dacă nu a fost căutat un student, utilizatorul va fi anunțat că această acțiune este necesară pentru a continua.

```
private void btnSterge_Click(object sender, EventArgs e)
{
    if (contorcautare == 1)
    {

            StudentCamin s = adminStudenti.GetStudent(txtNume.Text, txtPrenume.Text);
            if (s != null)
            {
                adminStudenti.DeleteStudent(s);
                List<StudentCamin> stud = adminStudenti.GetStudenti();
                AdaugaStudentInControlDataGridView(stud);
                lblModifica.Text = "Modificare efectuata";
                txtNume.Enabled = true;
                txtPrenume.Enabled = true;
            }
            else
            {
                lblModifica.Text = "Student inexistent!";
            }
    }
    else
    {
        MessageBox.Show("Trebuie sa cautati un student!");
    }
    ResetareControale();
    contorcautare = 0;
}
```

Figura II.19 Ștergerea unui student din listă

```
public  bool DeleteStudent  (StudentCamin studentGresit)
{
    List<StudentCamin> studenti = GetStudenti();
    int idid = 1;

    bool actualizareCuSucces = false;
    try
    {
        //instructiunea 'using' va apela la final swFisierText.Close();
        //al doilea parametru setat la 'false' al constructorului StreamWriter indica modul 'overwrite' de deschidere al fisierului
        using (StreamWriter swFisierText = new StreamWriter(NumeFisier, false))
        {
            foreach (StudentCamin stud in studenti)
            {
                //informatiile despre studentul actualizat vor fi preluate din parametrul "studentActualizat"
                if (stud.IdStudent != studentGresit.IdStudent )
                {
                    stud.IdStudent = idid;
                    swFisierText.WriteLine(stud.ConversieLaSir_PentruFisier());
                    idid++;

                }


            }
            actualizareCuSucces = true;
        }
    }
    catch (IOException eIO)
    {
        throw new Exception("Eroare la deschiderea fisierului. Mesaj: " + eIO.Message);
    }
    catch (Exception eGen)
    {
        throw new Exception("Eroare generica. Mesaj: " + eGen.Message);
    }

    return actualizareCuSucces;
}
```

Figura II.20 Rescrierea fișierului fără studentul căutat

Prin acționarea butonului „Înapoi”, administratorul se va întoarce în forma intermediară prezentată în Figura II.5.

III. Observații

Pentru o funcționare corectă a aplicației se recomandă modificarea path-ului aferent imaginilor din folder-ul proiectului. Codul ce trebuie modificat se regăsește în „Info.cs”.

IV. Concluzii

În concluzie, această aplicație nu reprezintă un soft profesional, ea execută cu succes opțiunile puse la dispoziție fără erori majore dar există oricând loc de îmbunătățiri. Proiectul suprinde caracteristicile principale ale programării orientate pe obiecte: încapsularea, moștenirea, polimorfismul și reutilizarea.

V. Bibliografie

Am atașat câteva linkuri ale unor videoclipuri ce m-au ajutat în realizarea acestui proiect. Menționez că inspirația principală a fost activitatea susținută în cadrul laboratorului de Programarea interfețelor utilizator.

https://www.youtube.com/watch?v=K9Ps66GoD-k
https://www.youtube.com/watch?v=GhQdlIFylQ8&t=4093s
https://www.youtube.com/watch?v=gfkTfcpWqAY
https://www.youtube.com/watch?v=trkXsXIh0xo&t=577s
https://www.youtube.com/watch?v=JP5rgXO_5Sk