

TEHNICI AVANSATE DE PROGRAMARE – Curs 3

Extenderea modelului simplu pt.vizualizarea/actualizarea unei tablele dintr-o baza de date

- Lucrul cu chei externe
- Protejarea la editare a unor coloane
- Obligatorietatea completarii unor campuri
- Completarea unor campuri utilizand fereastra de dialog pt. fisiere
- La renuntare sa nu se mai trateze evenimentul `DataRow`

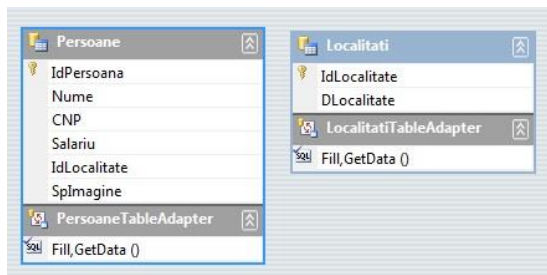
Vom pleca de la modelul prezentat in cursul 2 ce implementeaza urmatoarele functii:

- Controlul celor doua stari: vizualizare, actualizare
- Confirmarea stergerii unei inregistrari
- Verificarea duplicatelor pe cheia semantica (implementata printr-un index unic in baza de date).
- Verificarea integritatii referentiale la stergerea unei linii (daca este referita de o alta inregistrare).
- Validarea formatului numeric si data calendaristica

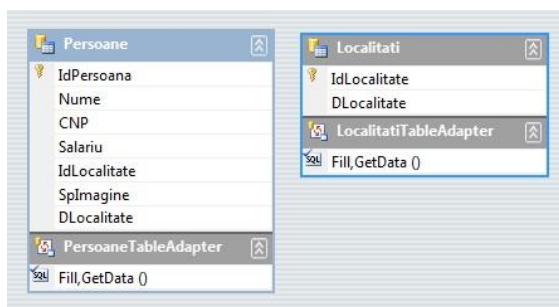
Consideram tabela *Persoane*:

Persoane (*IdPersoana*, Nume, CNP, Salariu, *IdLocalitate*, SpImagine)

Adaugam la `DataSet1` un nou `dataTable`, *Localitati*:



Adaugam un camp nou, *DLocalitate* in `DataTable`-ul *Persoane*:

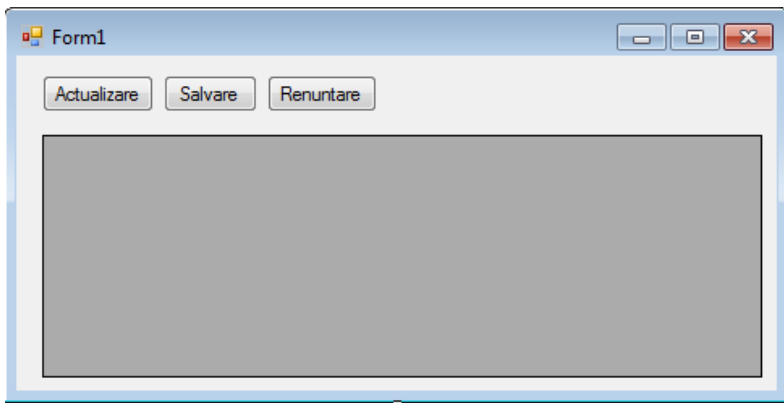


▪ Lucrul cu chei straine

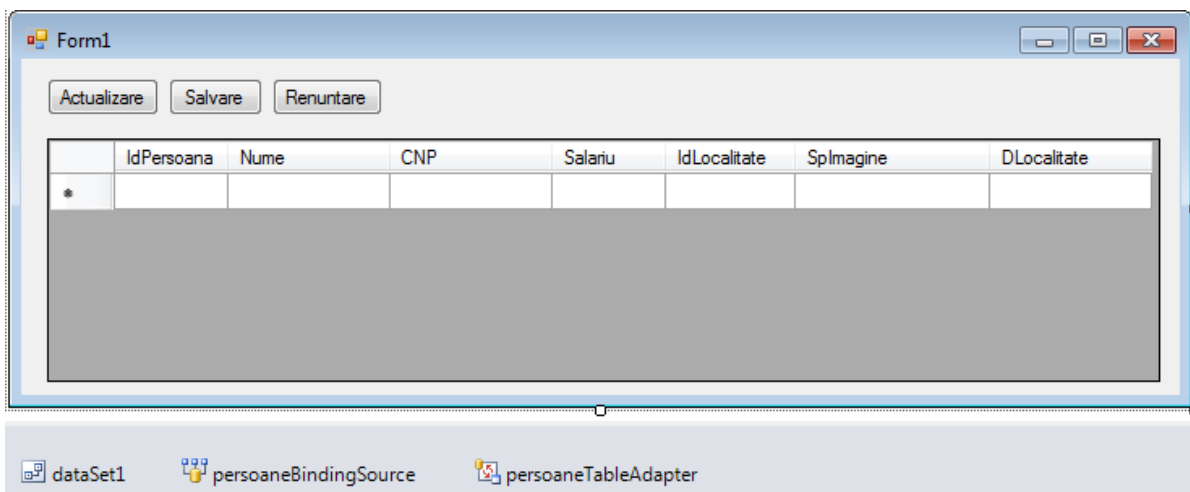
Tabela *Persoane* are o cheie straina, *IdLocalitate*. Pt. utilizator este dificil sa completeze *IdLocalitate*.

Mai convenabil ar fi sa selecteze numele localitatii dintr-un `comboBox`, chiar in grid, iar la selectia unei localitati, programul sa completeze automat *IdLocalitate* in grid.

Structuram form-ul astfel:

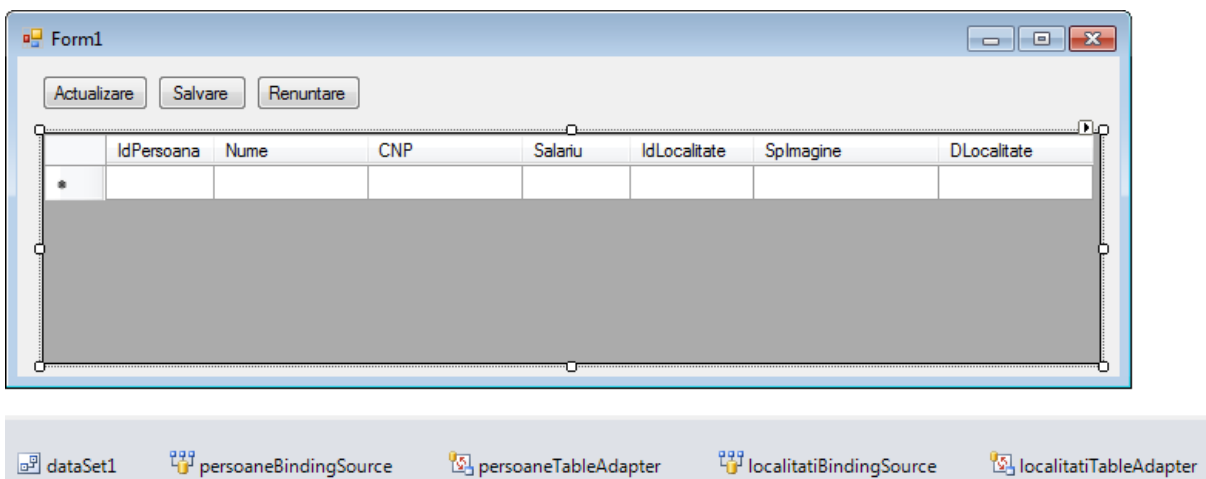


Legam grid-ul la dataTable-ul Persoane modificat.



Schimbam cu edit columns, tipul coloanei (column type) pt. DLocalitate din TextBox in ComboBox.
Legam DLocalitate din grid a dataTable-ul Localitati din DataSet1.
Setam Display member la DLocalitate. Nu setam Value member.
Setam deasemenea proprietatea DisplayStyleForCurrentCellOnly la true pt. campul DLocalitate.

Form-ul devine:



Completam codul cu cel din modelul anterior.

Extindem acum codul.

Modificam metoda refresh: fill Localitati dataTable.

```
private void refresh() {  
    persoaneTableAdapter.Fill(dataSet1.Persoane);  
    localitatiTableAdapter.Fill(dataSet1.Localitati);  
}
```

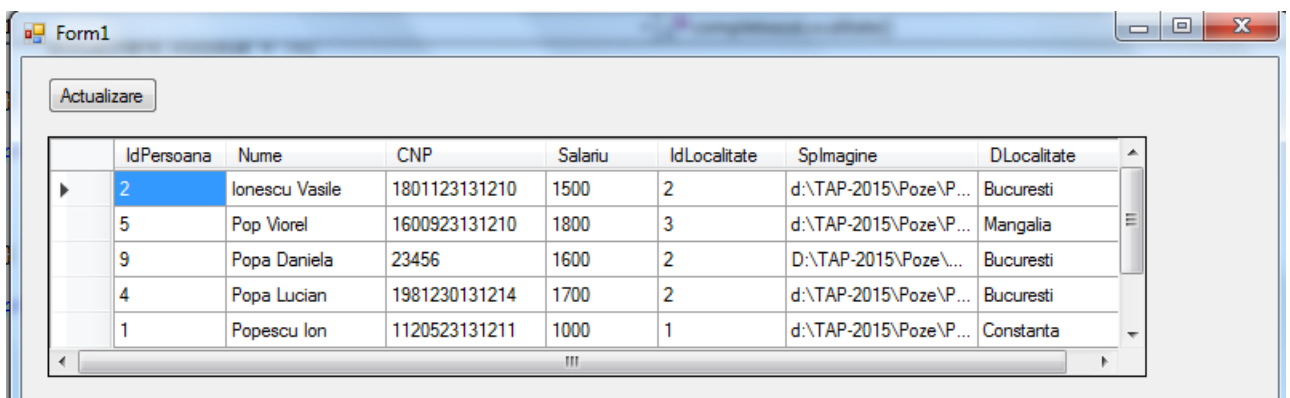
Trebuie sa completam campul DLocalitate prin program (in metoda refresh). Definim metoda completeazaLocalitate.

```
private void completeazaLocalitate() {  
    String idLoc;  
    int idLocalitate;  
    DataRow r;  
    foreach (DataGridViewRow x in persoaneBindingSource) {  
        idLoc = x["IdLocalitate"].ToString();  
        idLocalitate = Convert.ToInt32(idLoc);  
        r = dataSet1.Tables["Localitati"].Rows.Find(idLocalitate);  
        if (r != null) {  
            x["DLocalitate"] = r[1].ToString();  
        }  
        else {  
            MessageBox.Show("Nu exista localitate cu id " + idLocalitate);  
        }  
    }  
}
```

Adaugam apelul ei in metoda refresh:

```
private void refresh() {  
    persoaneTableAdapter.Fill(dataSet1.Persoane);  
    localitatiTableAdapter.Fill(dataSet1.Localitati);  
    completeazaLocalitate();  
}
```

Cand executam aplicatia, avem:



Trebuie acum sa completam campul IdLocalitate din grid, atunci cand utilizatorul selecteaza o localitate din comboBox.

Se creaza o noua metoda eveniment,

```
private void dataGridView1_CellValueChanged(object sender, DataGridViewCellEventArgs e) {  
    DataGridViewRow crt;  
    try {
```

```

        if (dataGridView1.CurrentCell.ColumnIndex == DLocalitateIndex){
            crt = (DataRowView)localitatiBindingSource.Current;
            dataGridView1.CurrentRow.Cells[IdLocalitateIndex].Value = crt["IdLocalitate"];
        }
    }
    catch { }
}

```

Avem nevoie de doua constante DLocalitateIndex si IdLocalitateIndex, indecsii din grid pt. aceste coloane.

```

const int DLocalitateIndex = 6;
const int IdLocalitateIndex = 4;

```

Vom adauga in plus:

```

const int IdPersoanaIndex = 0;
const int NumeIndex = 1;
const int CNPIndex = 2;
const int SalariuIndex = 3;
const int SpImagineIndex = 5;

```

Punem aceste declaratii la inceputul clasei.

▪ **Protjarea la editare a unor coloane**

Vom folosi proprietatea ReadOnly pt. coloanele grid-ului (in loc de grid).

Modificam metoda Form_Load:

```

private void Form1_Load(object sender, EventArgs e) {
    //A1
    config(true);
    refresh();
    dataGridView1.Columns[IdPersoanaIndex].ReadOnly = true;
    dataGridView1.Columns[IdLocalitateIndex].ReadOnly = true;
}

```

Modificam deasemenea metoda config:

```

private void config(bool v) {
    dataGridView1.AllowUserToAddRows = !v;
    dataGridView1.AllowUserToDeleteRows = !v;
    dataGridView1.Columns[NumeIndex].ReadOnly = v;
    dataGridView1.Columns[CNPIndex].ReadOnly = v;
    dataGridView1.Columns[SalariuIndex].ReadOnly = v;
    dataGridView1.Columns[SpImagineIndex].ReadOnly = v;
    dataGridView1.Columns[DLocalitateIndex].ReadOnly = v;
    btnActualizare.Enabled = v;
    btnSalvare.Visible = !v;
    btnRenuntare.Visible = !v;
}

```

▪ **Obligativitatea introducerii datelor pt. anumite campuri**

Sa consideram ca Nume si CNP sunt obligatoriu de completat. Vom face aceasta validare, analizand liniile din dataTable (din dataSet) inainte de a le salva in baza de date.

Vom define metoda *completareCampuri*, ce returneaza *true* daca sunt completate toate campurile obligatorii, sau fals altfel.

```

private bool completareCampuri() {
    bool raspuns = true;
    foreach (DataRow r in dataSet1.Persoane)

```

```

    {
        if (r.RowState == DataRowState.Deleted) continue;
        if (r["Nume"] == DBNull.Value)
        {
            MessageBox.Show("Completati Nume la linia cu Id " + r["IdPersoana"]);
            raspuns = false;
        }
        if (r["CNP"] == DBNull.Value)
        {
            MessageBox.Show("Completati CNP la linia cu Id " + r["IdPersoana"]);
            raspuns = false;
        }
    }
    return raspuns;
}

```

Aceasta metoda trebuie apelata la inceputul metodei btnSalvare_Click:

```

private void btnSalvare_Click(object sender, EventArgs e) {
    //A4
    if (!completareCampuri()) return;
    try
    {
        persoaneTableAdapter.Update(dataSet1.Persoane);
        config(true);
        refresh();
    }
    catch (Exception exc)
    {
        string s = exc.Message;
        if (s.IndexOf("duplicate values") > 0)
            MessageBox.Show("Inregistrare deja existenta !");
        else if (s.IndexOf("cannot be deleted") > 0)
            MessageBox.Show("Ati sters inregistrari referite in alte tabele !");
    }
}

```

▪ Completarea unor campuri utilizand fereastra open file dialog

Sa consideram adaugarea unui fisier imagine la fiecare persoana.

Vom adauga un PictureBox si un OpenFileDialog, form-ului.

Legam proprietatea ImageLocation la campul SpImagine din persoaneBindingSource.

Setam proprietatea SizeMode (de la PictureBox) la StretchImage.

Generam metoda eveniment *dataGridView1_MouseDoubleClick*:

```

private void dataGridView1_MouseDoubleClick(object sender, MouseEventArgs e) {
    if (btnActualizare.Enabled) return;

    if (dataGridView1.CurrentCell.ColumnIndex == SpImagineIndex )
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            string s = openFileDialog1.FileName; ;
            dataGridView1.CurrentRow.Cells[SpImagineIndex].Value = s;
            pictureBox1.ImageLocation = s;
        }
}

```

▪ La renuntare sa nu se mai trateze evenimentul DataError

Modificam metoda eveniment *dataGridView1_DataError*:

```
private void dataGridView1_DataError(object sender, DataGridViewDataErrorEventArgs e) {  
    if (btnRenuntare.Focused) {  
        dataGridView1.CancelEdit();  
        //A3  
        config(true);  
        refresh();  
        return;  
    }  
    MessageBox.Show("Format eronat");  
}
```