Laboratorul 3

Lucrul cu baze de date relaționate în Visual C# .NET

Ce ne propunem astăzi?

În acest laborator ne propunem să implementăm în Visual C# .NET o aplicație destinată unui cabinet medical de radiologie, care va trebui să păstreze o evidență a pacienților și a imaginilor radiologice ale acestora. Aplicația va permite adăugarea și vizualizarea tuturor imaginilor radiologice ale unui pacient. Toate aceste date vor fi păstrate într-o bază de date relațională.

Aplicația va opera cu datele dintr-o bază de date care conține două tabele relaționate (tabelele Pacienti și Radiografii). Pe fereastra principală va fi afișată sub formă tabelară o listă a tuturor pacienților care au efectuat consultații în cadrul cabinetului medical. Se va prevedea posibilitatea de căutare a pacienților după nume. Operațiile de adăugare de pacienți noi sau de modificare a datelor pacienților existenți se vor efectua direct în controlul tabelar (DataGridView). Pentru fiecare pacient se vor putea adăuga imagini radiologice care vor putea fi apoi vizualizate sub formă de miniaturi într-o listă. La click pe o miniatură, imaginea radiologică se va încărca la dimensiune completă într-un control de tip PictureBox, iar celelalte detalii vor fi afișate într-o casetă text (Figura 1).

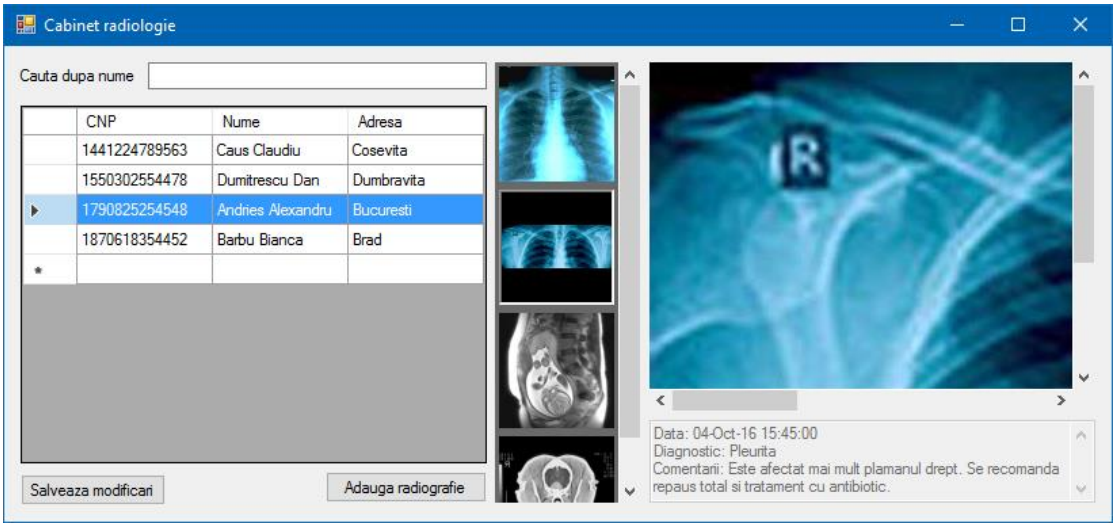

***Figura 1**. Interfața aplicației pentru cabinete medicale de radiologie*

Mai pe larg, vom proceda astfel…

Baza de date cu care vom lucra va fi de tip SQL Server și poate fi creată direct din Visual Studio (vă va fi totuși mai ușor dacă aveți instalat SQL Server Management Studio). În panoul *Server Explorer* dați click dreapta pe nodul *Data Connections*, alegeți comanda *Add Connection* și creați o nouă bază de date locală SQL Server (Figura 2).

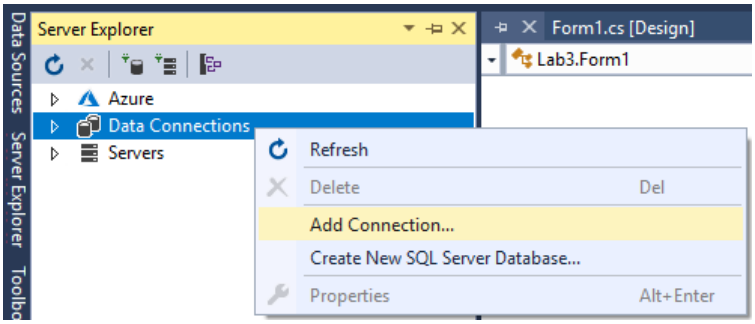

Figura 2. *Inițierea operației de creare a unei noi baze de date locale SQL Server.*

În fereastra dialog care se va deschide selectați serverul SQL și introduceți numele noii baze de date (Figura 3).

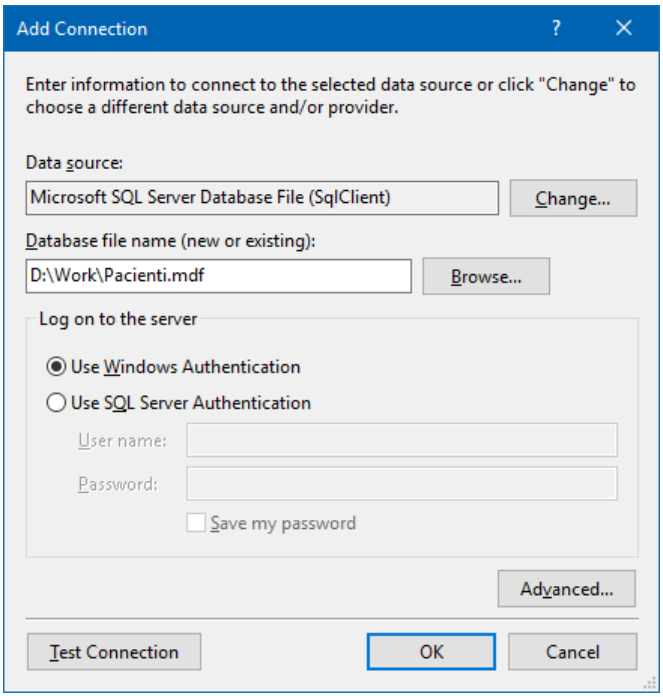

Figura 3. *Alegerea numelui bazei de date*

Dacă acel fișier de bază de date nu există deja, alegeți ca el să fie creat.

Se poate întâmpla ca modificările efectuate asupra structurii bazei de date să nu fie vizibile în panoul *Server Explorer* decât după comanda *Refresh*. Imediat după creare, baza de date nu conține niciun tabel, astfel că vom adăuga două tabele noi (vezi Figura 4).

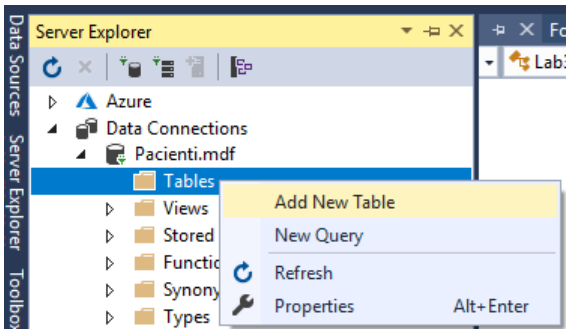

Figura 4. *Adăugarea unui tabel nou*

Cele două tabele pe care le vom adăuga sunt: Pacienti și Radiografii. Structurile lor sunt prezentate în Figura 5, respectiv Figura 6.

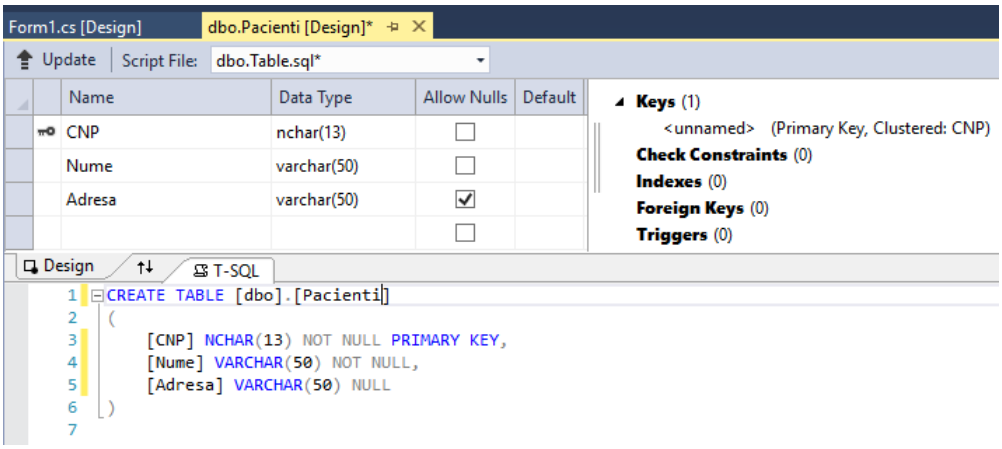

Figura 5. *Structura tabelului Pacienti*

După definirea în acest fel a structurii fiecărui tabel se va da comanda *Update* (plasată în colțul stânga-sus al ferestrei) apoi, în fereastra dialog care se va deschide, comanda *Update Database*.

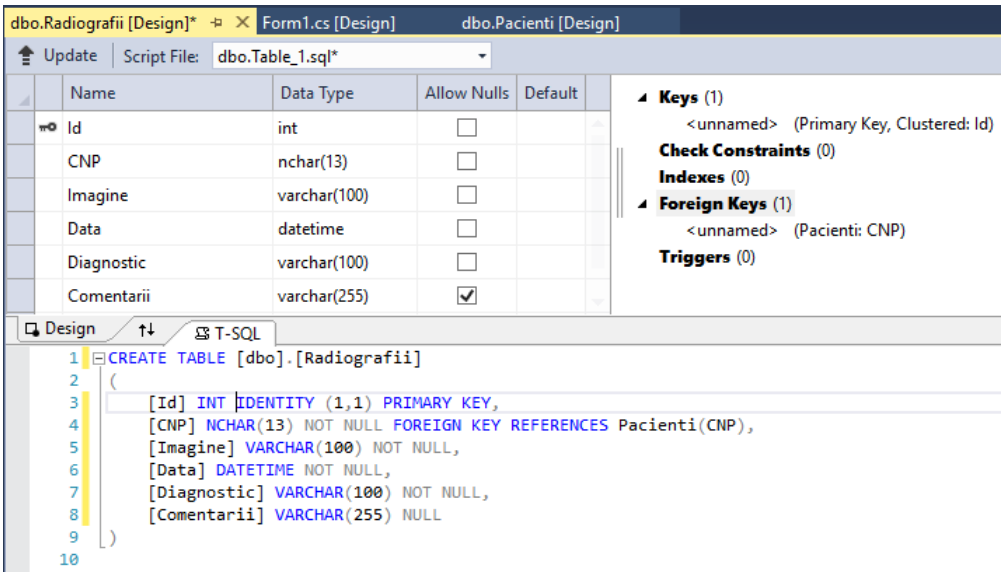

Figura 7. *Structura tabelului Radiografii*

Restricția de tip Foreign Key aplicată câmpului CNP impune relaționarea celor două tabele în sensul în care unei înregistrări din tabelul Pacienti (identificată prin CNP) îi poate corespunde oricâte înregistrări din tabelul Radiografii.

Cele două script-uri pentru crearea tabelelor sunt expuse în continuare:

```
CREATE TABLE [dbo].[Pacienti]
(
    [CNP] NCHAR(13) NOT NULL PRIMARY KEY,
    [Nume] VARCHAR(50) NOT NULL,
    [Adresa] VARCHAR(50) NULL
)
```

```
CREATE TABLE [dbo].[Radiografii]
(
    [Id] INT IDENTITY (1,1) PRIMARY KEY,
    [CNP] NCHAR(13) NOT NULL FOREIGN KEY REFERENCES Pacienti(CNP),
```

```
    [Imagine] VARCHAR(100) NOT NULL,
    [Data] DATETIME NOT NULL,
    [Diagnostic] VARCHAR(100) NOT NULL,
    [Comentarii] VARCHAR(255) NULL
)
```

Legarea aplicației la baza de date

Putem asigura legarea aplicației la baza de date fără a scrie nicio linie de cod urmând pașii descriși în continuare.

În Visual Studio selectați din fereastra flotantă *Data Sources* comanda *Add New Data Source*. Aici alegeți Data Source Type: Database și Database Model: Dataset (Figura 8).

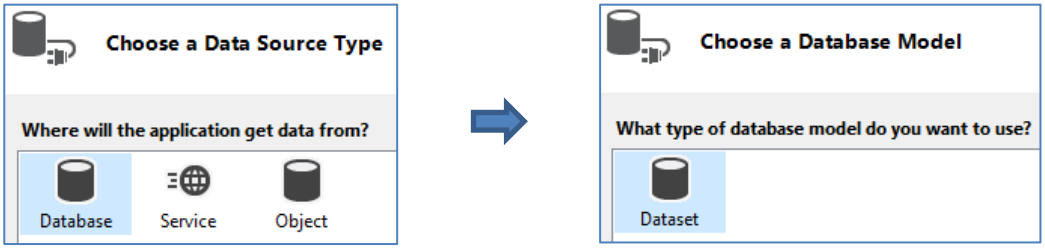

Figura 8. *Add new Data Source*

În continuare selectați baza de date creată anterior (Figura 9).

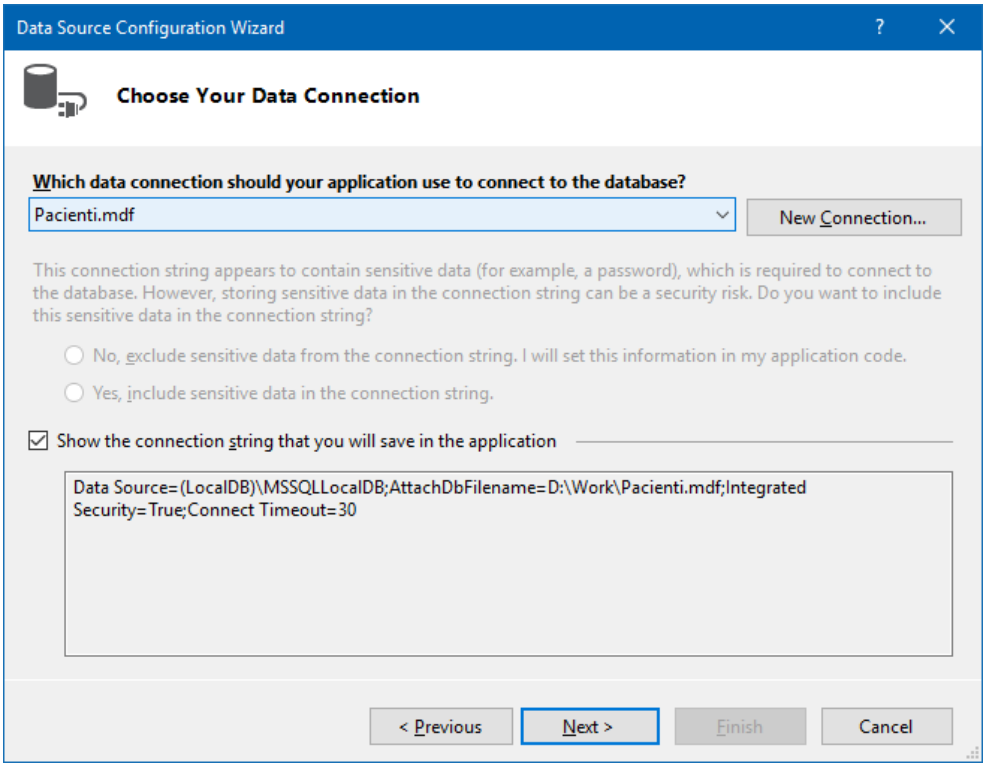

Figura 9. *Alegerea bazei de date*

În continuare, se parcurge wizard-ul până la dialogul „Choose your database objects”, în care vom selecta ambele tabele ale bazei de date (Figura 10).

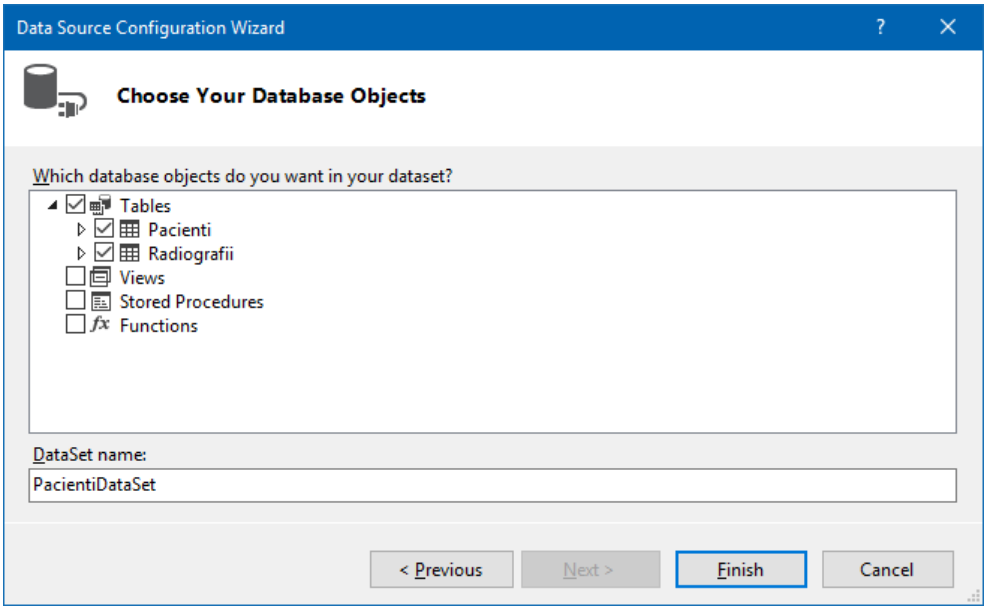

Figura 10. *Selectarea obiectelor pe care dorim să le includem în DataSource.*

După finalizarea wizard-ului, se poate observa în fereastra Data Sources că tabelul Radiografii este listat atât ca nod independent, cât și ca nod fiu al tabelului Pacienti, deoarece există o relație „părinte-copil“ între cele două (Figura 11).

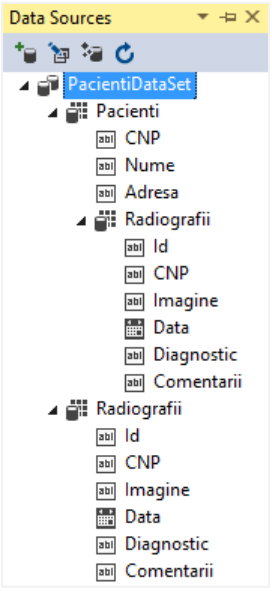

Figura 11. *Fereastra Data Sources.*

Dacă la comanda *Add New Data Source* ați optat ca sursa datelor să fie „Microsoft SQL Server Database File”, veți fi informați că baza de date este un fișier local și veți fi întrebați dacă vreți să o adăugați la proiect (Figura 12). Dacă selectați „Yes”, baza de date va fi inclusă în proiect și va apărea în fereastra *Project Explorer*, iar la fiecare compilare a proiectului baza de date va fi copiată în directorul care conține fișierul executabil. Dacă selectați „No”, atunci aplicația va lucra cu baza de date aflată într-o locație fixă.

Dacă la întrebare ați ales răspunsul „Yes”, atunci, pentru a evita problemele de sincronizare a datelor, selectați numele bazei de date (Pacienti.mdf) în *Solution Explorer* și selectați pentru proprietatea Copy to Output Directory valoarea „Copy if newer”.

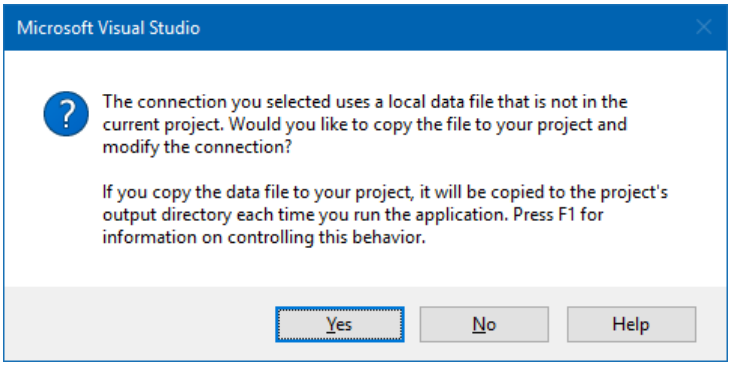

Figura 12. *Dialogul pentru adăugarea bazei de date la proiectul curent*

După ce am creat sursa de date putem construi extrem de ușor aplicații pentru lucrul cu baza de date. Trebuie doar să efectuăm operații drag–and–drop cu obiectele din fereastra Data Sources pe un formular.

Putem facem click pe un tabel din Data Sources apoi drag-and-drop pe un formular iar Visual C# va crea automat controalele BindingNavigator, DataGridView împreună cu alte controale și componente pentru afișarea datelor din tabel. Datele din tabel pot fi afișate în mod tabelar (control DataGridView) sau detaliat (datele din fiecare câmp sunt afișate în controale separate). Modul de afișare poate fi ales din meniul ce va fi afișat la click pe un tabel din fereastra Data Sources (Figura 13).

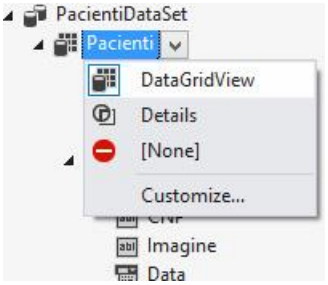

Figura 13. *Setarea modului de afișare a datelor dintr-un tabel*

În fereastra DataSource vom alege tabelul Pacienti (vizualizare tabelară) și îl vom trage peste formular (drag-and-drop), apoi vom face același lucru pentru tabelul Radiografii (vizualizare tabelară), cel afișat ca nod fiu al tabelului Pacienti. Împreună cu cele două controale de tip DataGridView vor fi adăugate automat pe formular un control de tip BindingNavigator și o serie de componente utile pentru lucrul cu datele din cele două tabele.

În acest moment se poate rula aplicația, putându-se observa legătura dintre cele două tabele ale bazei de date: la selectarea unui pacient din controlul `pacientiDataGridView` în controlul `radiografiiDataGridView` vor fi afișate numai radiografiile asociate pacientului selectat.

După aceasta vom șterge de pe formular controlul `pacientiBindingNavigator` (nu avem nevoie de el) și controlul `radiografiiDataGridView` (avem nevoie doar de componentele create odată cu acesta: `radiografiiBindingSource` și `radiografiiTableAdapter`). În final, pentru interacțiunea cu baza de date, vor rămâne pe formular următoarele controale și componente:

- `pacientiDataGridView`
- `pacientiDataSet`
- `pacientiBindingSource`
- `pacientiTableAdapter`
- `tableAdapterManager`
- `radiografiiBindingSource`
- `radiografiiTableAdapter`.

Reamintim faptul că atât operația de adăugare a unui pacient cât și cea de modificare a datelor acestuia se vor efectua direct din controlul `pacientiDataGridView`.

Iată o scurtă descriere a rolului pe care îl are fiecare tip de control/componentă utilizat pentru operarea cu datele din baza de date:

- Un obiect **DataSet** reprezintă o întreagă bază de date. Acesta conține obiecte de tip DataTable care reprezintă tabelele din baza de date. Fiecare obiect DataTable conține obiecte DataRow care reprezintă înregistrările din baza de date, iar fiecare obiect DataRow reprezintă câte o coloană a unei înregistrări.
- Obiectele de tip **TableAdapter** sunt utilizate pentru comunicația dintre aplicație (DataSet) și o bază de date. Acestea oferă metode pentru efectuarea operațiilor asupra bazei de date. Obiectele TableAdapter mai sunt utilizate pentru a trimite date actualizate de la aplicația curentă înapoi la baza de date.
- **TableAdapterManager** reprezintă o componentă nouă începând cu Visual Studio 2008 și oferă funcții de salvare a datelor în tabele relaționate.
- Un obiect **BindingSource** încapsulează toate datele din DataSet și oferă funcții pentru controlul acestora din cadrul programului.
- **BindingNavigator** oferă o interfață grafică pentru ca utilizatorul să poată controla BindingSource.

În mod implicit, controlul de tip DataGridView permite adăugarea, modificarea și ștergerea înregistrărilor. Totuși, pentru ca aceste modificări efectuate asupra datelor din control să fie reflectate și la nivelul bazei de date este nevoie de salvarea acestora. Codul asociat cu comanda *Salvează modificări* este următorul.

```
this.Validate();
this.pacientiBindingSource.EndEdit();
this.tableAdapterManager.UpdateAll(this.pacientiDataSet);
```

Afișarea imaginilor și crearea dinamică a controalelor

Afișarea imaginilor radiologice pentru pacientul selectat se va face la evenimentul PositionChanged asociat cu componenta `pacientiBindingSource`.

Pentru a putea afișa sub formă miniaturală imaginile radiologice asociate unui pacient va trebui să utilizăm un control container de tip FlowLayoutPanel și o modalitate de creare dinamică a unor controale de tip PictureBox, câte unul pentru fiecare imagine radiologică. Crearea dinamică a controalelor este folosită în special atunci când nu avem cum să știm cu exactitate numărul de controale de care vom avea nevoie pe timpul rulării aplicației.

Pentru controalele componente ale containerului FlowLayoutPanel nu trebuie precizată poziția, deoarece acest container distribuie automat controalele componente, în ordine, fie pe orizontală (de la stânga la dreapta și de sus în jos), fie pe verticală (de sus în jos și de la stânga la dreapta).

Iată un **exemplu** de creare dinamică a unui control și de adăugare a unui handler de eveniment (procedură de tratare a unui eveniment).

```
PictureBox pic = new PictureBox();

//Setarea proprietatilor controlului:
pic.SetBounds(0, 0, latimeControl, inaltimeControl);
pic.BackColor = Color.Black;
pic.SizeMode = PictureBoxSizeMode.Zoom;
pic.Image = Bitmap.FromFile(numeFisier);
pic.Tag = "Alte informatii despre control";

//Adaugarea controlului nou creat la colectia de controale a containerului:
```

```
flowLayoutPanel1.Controls.Add(pic);
//Adaugarea unui handler pentru evenimentul Click:
pic.Click += pic_Click;
```

Procedura eveniment `pic_Click` va fi creată automat după tastarea operatorului `+=` urmată de TAB TAB, și va avea următoarea formă:

```
void pic_Click(object sender, EventArgs e)
{
    PictureBox pic = (PictureBox)sender;
    // ...
}
```

Parametrul `sender` poate oferi informații despre controlul PictureBox pentru care a fost apelat.

Proprietatea Tag a unui control vă permite să asociați cu acel control orice informații care vă pot fi de folos (cum ar fi, pentru un control PictureBox, detalii despre imaginea radiologică).

Accesul la datele din tabelul Radiografii se face astfel:

```
// parcurgerea tuturor radiografiilor unui pacient
foreach (DataRowView drv in radiografiiBindingSource.List)
{
    // obtinerea valorii din campul 'Imagine'
    string img = (string)drv["Imagine"];

    // crearea unui PictureBox pentru fiecare radiografie
    // ...
}
```

Controlul container FlowLayoutPanel permite adăugarea automată a unor bare de derulare în cazul în care conținutul acestuia depășește marginile controlului container (proprietatea `AutoScroll` setată pe *true*). Același mecanism va fi utilizat și în cazul afișării imaginii radiologice la dimensiune completă (un control PictureBox în interiorul unui control container de tip Panel).

Afișarea imaginilor sub formă de miniatură se face prin dimensionarea controlului PictureBox la lățimea și înălțimea dorite, apoi prin setarea proprietății `SizeMode` la valoarea *Zoom*. Astfel, imaginea încărcată în controlul PictureBox va fi afișată astfel încât să fie cuprinsă în întregime pe suprafața controlului, indiferent de dimensiunile acestuia (dacă imaginea are dimensiuni mai mari decât controlul, atunci aceasta va fi micșorată la dimensiunile controlului, iar dacă imaginea are dimensiuni mai mici decât controlul, atunci aceasta va fi mărită la dimensiunile controlului).

Afișarea imaginii radiologice la dimensiune completă (împreună cu celelalte detalii, vezi Fig. 1) se va face la click pe una dintre imaginile miniaturale. Pentru ca o imagine să fie afișată la dimensiunile ei normale, vom seta proprietatea `SizeMode` a controlului PictureBox la valoarea *AutoSize*.

Sfaturi utile

Dacă doriți ca la click pe imaginea miniaturală selectată aceasta să producă impresia de buton apăsat, setați proprietatea BorderStyle la valorile BorderStyle.None sau BorderStyle.Fixed3d.

După cum am amintit la început, aplicația va prevedea posibilitatea adăugării unei imagini radiologice noi pentru un pacient. La apăsarea butonului corespunzător se va deschide o fereastră dialog în care utilizatorul va putea indica toate detaliile asociate imaginii radiologice (Figura 14).

Presupunem că fereastra de introducere a datelor unei radiografii se numește Form2. Aceasta va fi afișată ca fereastră dialog (fereastră modală). O fereastră modală este afișată prioritar, deasupra interfeței aplicației și o dezactivează temporar, până la închiderea ferestrei modale. Afișarea ferestrei Form2 ca fereastră modală se face astfel:

```
Form2 f = new Form2();
f.ShowDialog();
```

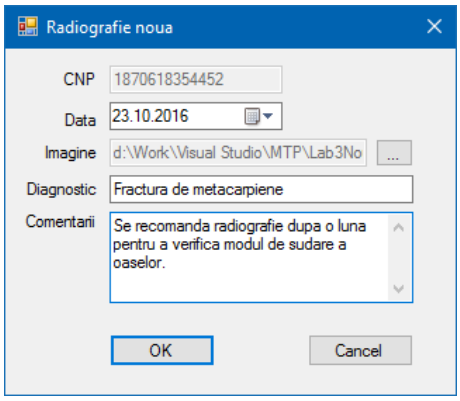

Figura 14. *Adăugarea detaliilor unei radiografii*

O fereastră, odată afișată modal (cu metoda `ShowDialog`), oprește chiar și fluxul de execuție a codului apelant până la închiderea acesteia. Metoda `ShowDialog` returnează un obiect enumerare (de tip `DialogResult`) care reprezintă rezultatul returnat în urma închiderii ferestrei. Valorile posibile ale unui obiect `DialogResult` sunt: `Abort`, `Cancel`, `Ignore`, `No`, `None`, `OK`, `Retry` sau `Yes`. Pentru cazul aplicației de față ne sunt suficiente valorile `DialogResult.OK` (la apăsarea butonului OK), respectiv `DialogResult.Cancel` (la apăsarea butonului Cancel).

Pentru ca fereastra de introducerea a datelor unei radiografii să se comporte precum o fereastră dialog, trebuie să putem folosi valoarea returnată de metoda `ShowDialog()` a acesteia. În acest sens, apăsarea butonul Ok setați proprietatea `DialogResult` a formularului la valoarea `DialogResult.OK`, iar la apăsarea butonului Cancel la valoarea `DialogResult.Cancel`. Acest lucru va avea ca efect închiderea formularului și returnarea valorii `DialogResult` alese.

Următoarea problemă de rezolvat este aceea a transmiterii datelor de la fereastra dialog la fereastra principală a aplicației și invers. Se recomandă crearea în clasa Form2 a proprietăților read-write: `CNP`, `Data`, `Imagine`, `Diagnostic` și `Comentarii`, care să opereze asupra conținutului controalelor de pe interfața ferestrei dialog.

```
Form2 f = new Form2();
f.CNP = (string)((DataRowView)pacientiBindingSource.Current)["CNP"];
if (f.ShowDialog()==DialogResult.OK)
{
    // Se preiau datele despre radiografie
    // Se salvează radiografia în baza de date (vezi mai jos)
}
```

Selectarea unui fișier pentru o imagine radiologica se face prin utilizarea clasei OpenFileDialog. Aceasta definește un dialog standard pentru alegerea unui fișier de pe disc și poate fi utilizat astfel:

```
OpenFileDialog dlg = new OpenFileDialog();
dlg.InitialDirectory = Application.StartupPath;
dlg.Filter = "*.jpg|*.jpg";
if (dlg.ShowDialog() == DialogResult.OK)
{
    // Fisierul ales este accesibil prin proprietatea dlg.FileName
}
```

Salvarea datelor

La apăsarea butonul OK va trebui ca în tabelul Radiografii să fie adăugată o nouă înregistrare. La operația de adăugare ne vom folosi de componenta `radiografiiTableAdapter`.

```
// Adaugarea datelor
radiografiiTableAdapter.Insert(cnp, imagine, data, diagnostic, comentarii);
// Salvarea datelor
tableAdapterManager.UpdateAll(pacientiDataSet);
// Reincarcarea datelor
radiografiiTableAdapter.Fill(pacientiDataSet.Radiografii);
```

Metoda `Insert()` a obiectului de tip TableAdapter acceptă ca parametri valorile care să fie introduse în câmpurile noii înregistrări. Acești parametri trebuie indicați în ordinea în care apar câmpurile din tabel și trebuie să aibă același tip de date pe care îl au câmpurile.

Căutarea unui pacient

Cea mai simplă modalitate de efectuare a căutării unui pacient din baza de date este setarea corespunzătoare a proprietății `Filter` a unei componente de tip BindingSource. Pentru anularea filtrului de căutare se va apela metoda `RemoveFilter()` a componentei de tip BindingSource.

Cu ce ne-am ales?

Prin aplicația dezvoltată în cadrul laboratorului de astăzi am reușit să ne familiarizăm cu utilizarea bazelor de date relaționate, să creăm dinamic controale și să le gestionăm cu ajutorul controalelor container.

Bibliografie

[1] Fill datasets by using TableAdapters, https://docs.microsoft.com/en-us/visualstudio/data-tools/fill-datasets-by-using-tableadapters?view=vs-2017

[2] How to programmatically add controls to Windows forms at run time by using Visual C#, https://support.microsoft.com/en-us/help/319266/how-to-programmatically-add-controls-to-windows-forms-at-run-time-by-u

[3] FlowLayoutPanel Class, https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.flowlayoutpanel?view=netframework-4.7.2