Laboratorul 5

Lucrul cu baze de date în Visual C#

Ce ne propunem astăzi?

În laboratorul de astăzi ne propunem să realizăm o aplicație de gestiune a datelor studenților, date care sunt stocate într-o bază de date Microsoft Access (Figura 1).

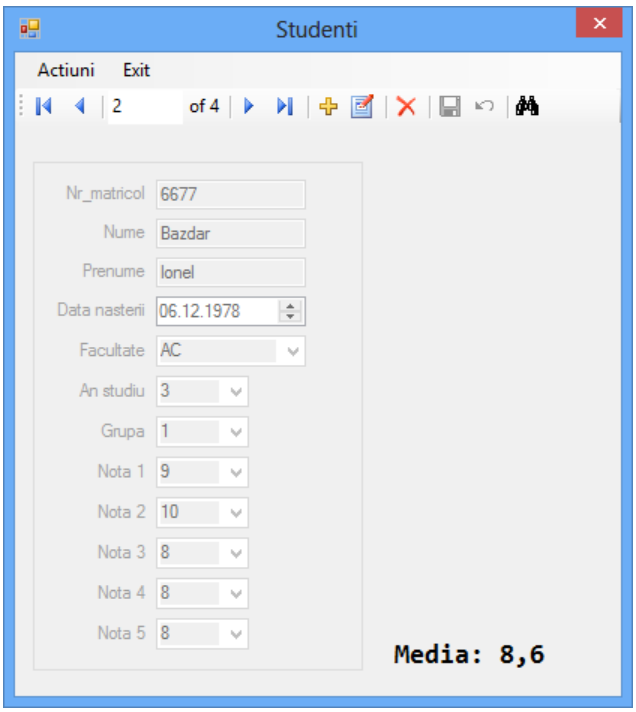


Figura 1. Fereastra principală a aplicației.

Mai pe larg, vom proceda astfel…

Primul pas în realizarea acestei aplicații este crearea bazei de date propriu-zise. Pentru aceasta, se va deschide aplicația Microsoft Access și se va crea o bază de date goală (Blank desktop database) care va fi salvată în directorul *bin\Debug* al proiectului.

Odată creată baza de date, comutați vizualizarea pe „Design view” și construiți structura tabelului, adăugând câmpurile ce se pot vedea în Figura 2. Câmpul `Nr_matricol` va fi ales cheie primară (click dreapta pe numele câmpului → Primary key). Un câmp sau un set de câmpuri de tip cheie primară identifică în mod unic fiecare înregistrare din tabel. Odată declarat cheie primară, un câmp nu poate conține duplicate sau valori NULL. Se aleg chei primare câmpuri de tip *AutoNumber* (câmpuri numerice cu auto-incrementare), câmpuri de care suntem siguri că nu vor conține valori duplicate (de exemplu CNP sau un număr unic de identificare), sau mai multe câmpuri care prin combinare dau o valoare unică.

După ce au fost introduse și stabilite proprietățile câmpurilor, se va salva tabelul cu numele „Studenti” apoi se vor adăuga câteva înregistrări.

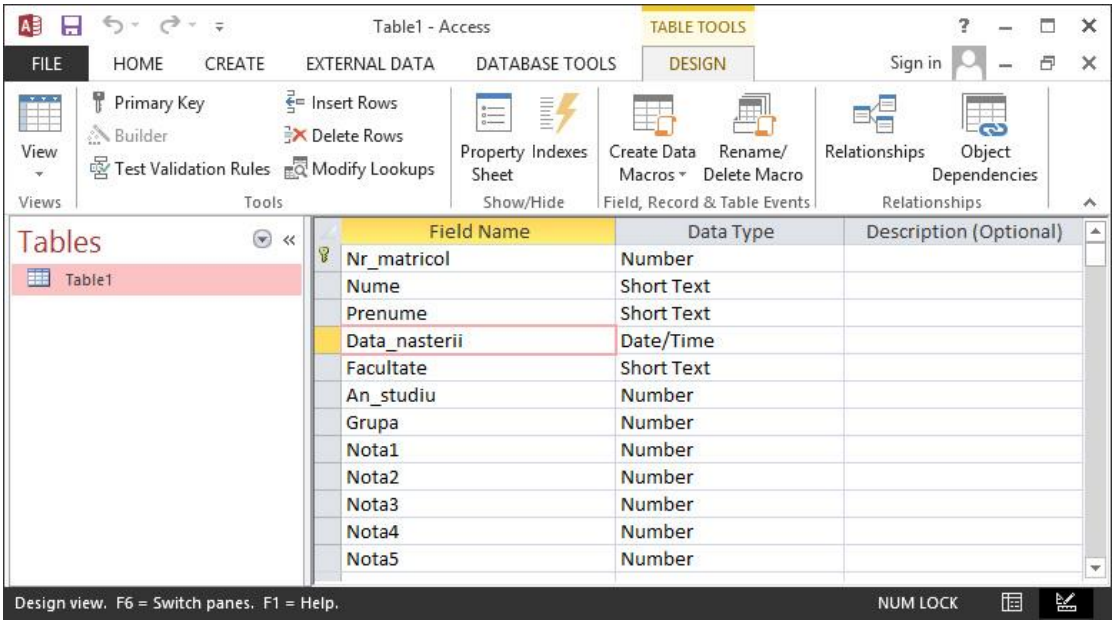


Figura 2. Structura tabelului

Atenție!

Se recomandă ca în numele tabelelor și a câmpurilor dintr-o bază de date să nu folosiți caracterul spațiu! Astfel, veți evita apariția erori la legarea controalelor la câmpurile bazei de date și la operațiile de salvare a datelor în baza de date.

În continuare se crea o aplicație Visual C# .NET de tip *Windows Forms Application* care va conține o fereastră de dimensiuni fixe precum cea din Figura 1.

Fereastra va conține un meniu care va implementa acțiunile din Figura 3 și o bară de unelte care va permite navigarea printre înregistrările din baza de date și apelarea rapidă a acțiunilor din meniu.

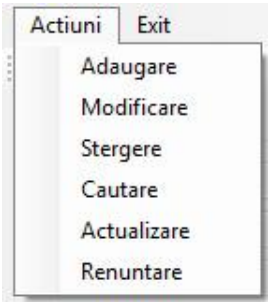


Figura 3. Meniul ferestrei principale.

Pe fereastră se vor mai adăuga: un control *BindingNavigator*, o componentă *BindingSource*, 12 etichete (*Label*) cu numele celor 12 câmpuri din tabelul Studenti, iar alăturat lor, 12 controale după cum urmează:

- Nr_matricol: TextBox.
- Nume: TextBox.
- Prenume: TextBox
- Data_nasterii: DateTimePicker (cu proprietățile Format = Short, ShowUpDown = True).
- Facultate: ComboBox (având valori posibile AC, ETC, MEC, CT și CI).
- An_studiu și Grupa: ComboBox (având valori posibile: 1, 2, 3, 4).
- Nota 1, Nota 2, Nota 3, Nota 4, Nota 5: ComboBox (cu valori posibile: 5, 6, 7, 8, 9, 10).

Controlul *BindingNavigator* vă pune la dispoziție comenzi pentru parcurgerea înregistrărilor din baza de date. Obiectele de tip *BindingSource* încapsulează datele din baza de date la care vă legați prin intermediul aplicației și oferă funcții pentru manipularea acestora. *BindingNavigator* va naviga printre datele din *BindingSource*, iar controale de pe formular (cum ar fi casetele de text și ComboBox-urile) vor fi legate la acest *BindingSource*.

În continuare se dorește editarea codului sursă al programului. În Visual Studio .NET lucrul cu baze de date se poate realiza utilizând tehnologiile OLEDB sau ADO.

OLEDB (Object Linking and Embedding, Database) este un set de interfețe proiectat de Microsoft pentru accesarea a diferite tipuri de date stocate într-o manieră uniformă. A fost proiectată ca un înlocuitor de un nivel mai înalt, fiind de fapt succesorul lui ODBC, extinzându-i caracteristicile. OLEDB separă, printr-un set de abstractizări, datele de aplicația care are nevoie să le acceseze. OLEDB este divizată la nivel conceptual între consumatori și furnizori. Consumatorii sunt aplicațiile care au nevoie să acceseze datele iar furnizorul este componenta software care implementează interfața și drept urmare care redă datele consumatorului.

ADO (ActiveX Data Objects) reprezintă un set de obiecte utilizate pentru accesarea surselor de date. ADO furnizează un nivel de abstractizare între client și interfețele OLEDB. ADO permite dezvoltatorului să scrie programe care accesează datele fără să știe cum este implementată baza de date. Sunt necesare cunoștințe privind baza de date doar pentru conectare. Nu sunt necesare cunoștințe de SQL pentru a accesa baza de date, deși se poate folosi ADO pentru a executa comenzi SQL.

ADO este bazat pe OLEDB, cele două nu sunt separate, nu sunt tehnologii distincte. OLEDB, fiind la un nivel mai jos decât ADO, este mai rapid. ADO prezintă un subset din capabilitățile OLEDB și abstractizează mult din funcționarea OLEDB.

Dezvoltarea aplicației utilizând OLEDB

Pentru a avea acces ușor la obiectele OLEDB este necesară importarea spațiului de nume OleDb:

```
using System.Data.OleDb;
```

Pentru dezvoltarea aplicației sunt necesare:

```
String string_conectare = "Provider=Microsoft.ACE.OLEDB.12.0; Data source=" +
  Application.StartupPath + "\\studenti.accdb;User ID=Admin;Password=;";
string sql = "SELECT * FROM Studenti";
OleDbConnection conexiune;
OleDbCommand comanda;
OleDbDataAdapter adaptor;
OleDbCommandBuilder construire_comenzi;
DataTable dt;
```

Obiectul de tip *OleDbConnection*, permite conectarea la baza de date prin intermediul stringului de conectare:

```
conexiune = new OleDbConnection(string_conectare);
conexiune.Open();
```

Obiectul de tip *OleDbCommand* permite efectuarea unei interogări asupra bazei de date, dar și executarea unor comenzi SQL NonQuery cum ar fi INSERT, UPDATE, DELETE etc.:

```
comanda = new OleDbCommand(sql, conexiune);
```

Un *DataSet* reprezintă o copie în memorie a datelor extrase dintr-o sursă de date. Un *DataSet* constă dintr-o colecție de tabele (*DataTable*), un tabel dintr-o colecție de înregistrări (*DataRow*), iar o înregistrare dintr-o colecție de câmpuri sau coloane (*DataColumn*).

```
dt = new DataTable();
```

Clasa *OleDbAdapter* reprezintă adaptorul de servicii utilizate pentru a efectua interogări asupra surselor de date OLEDB. *OleDbAdapter* reprezintă o punte între un *DataTable* sau *DataSet* și o sursă de date (cum ar fi o tabelă a unei baze de date) pentru extragerea datelor și salvarea lor. Adaptorul furnizează această punte prin utilizarea metodei `Fill` pentru a încărca datele din sursa de date (tabela) într-un *DataTable*, respectiv `Update` pentru a trimite schimbările făcute în *DataTable* înapoi în sursa de date.

```
adaptor = new OleDbDataAdapter(comanda);
adaptor.Fill(dt);
```

Un obiect *OleDbCommandBuilder* generează automat comenzile necesare pentru ca schimbările făcute într-un tabel *DataTable* (sau într-un *DataSet*) să se reflecte și în baza de date din care a fost încărcat *DataTable*-ul (*DataSet*-ul). Pentru ca un adaptor sa poată aplica schimbările făcute într-un *DataTable* (sau *DataSet*) asupra bazei de date (metoda `Update`), este necesară asocierea lui cu un obiect *OleDbCommandBuilder*. Asocierea dintre obiectul *OleDbCommandBuilder* și adaptor se poate face prin intermediul constructorului:

```
construire_comenzi = new OleDbCommandBuilder(adaptor);
```

În continuare urmează specificarea sursei de date a obiectului *BindingSource*, realizarea legăturii dintre controale din formular și *BindingSource*, precum și dintre *BindingNavigator* și *BindingSource*.

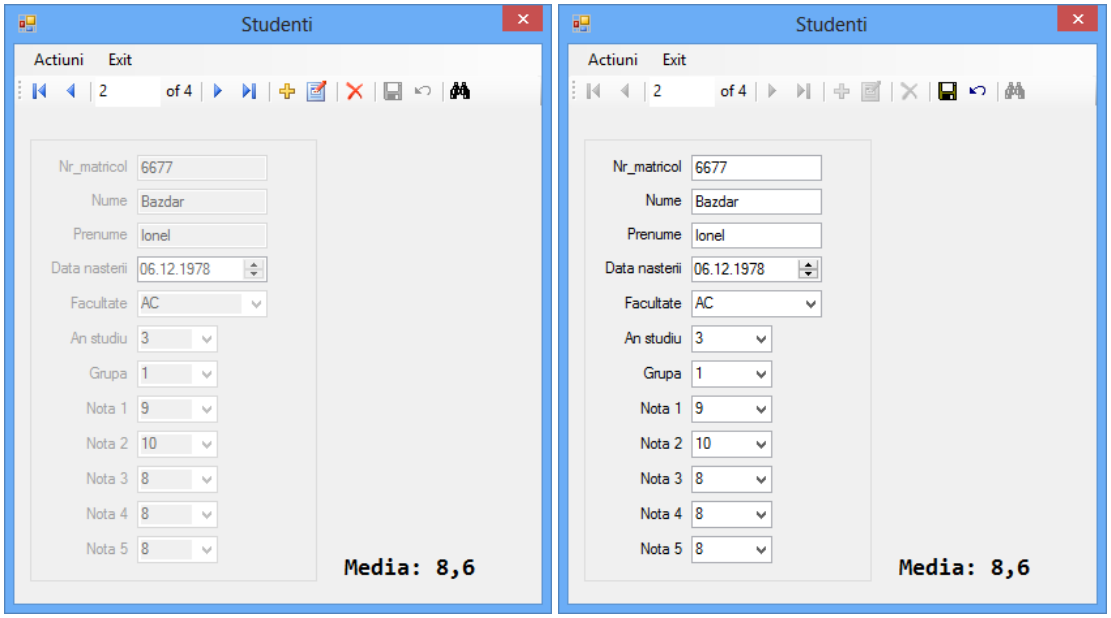
```
BindingSource1.DataSource = dt;
BindingSource1.Position = 0;

// legarea proprietății Text a casutei txtMatricol la campul Nr_matricol
txtMatricol.DataBindings.Add(new Binding("Text",BindingSource1,"Nr_matricol",true));
// in mod similar se va proceda si cu celelalte controale...
BindingNavigator1.BindingSource = BindingSource1;
```

Până în acest punct aplicația va permite vizualizarea înregistrărilor din baza de date și navigarea printre aceste înregistrări.

Aplicația va avea 2 stări de funcționare, care vor fi implementate în continuare:

- prima stare, în care se va putea naviga printre înregistrările din baza de date, acestea putând fi doar vizualizate nu și modificate. Butonul de Adăugare va fi activ, iar cele de Salvare și Renunțare vor fi inactive. Butoanele Editare, Ștergere și Căutare vor fi active doar dacă există înregistrări în baza de date. (Figura 4a)
- a doua stare se instaurează atunci când se apasă Adăugare sau Editare. Se vor dezactiva controalele de navigare și comenzile Adăugare, Editare, Ștergere, Căutare, singurele acțiuni posibile fiind Salvare sau Renunțare (Figura 4b).

Media notelor studentului curent va fi afișată în partea dreaptă a formularului.



(a) (b)

Figura 4. Starea 1 (a), respectiv starea 2 (b)

În continuare se vor implementa funcționalitățile controalelor Adăugare, Editare, Ștergere, Căutare, Salvare și Renunțare:

- Adăugare – butonul de adăugare poate să rămână cel implicit al obiectului *BindingNavigator*, sau se poate crea un buton nou. La apăsarea butonului adăugare se va trece din starea 1 în starea 2.
- Editare – singura acțiune care se va petrece la editare va fi cea de trecere din starea 1 în starea 2.
- Ștergerea nu va fi realizată fără o confirmare suplimentară, precum cea din Figura 5. Pentru a putea implementa aceasta funcționalitate se va selecta componenta `BindingNavigator1` și i se va seta proprietatea `DeleteItem` pe valoarea (none). Ștergerea se poate realiza prin comanda de mai jos și va fi urmată de salvare:

```
BindingSource1.RemoveCurrent();
```

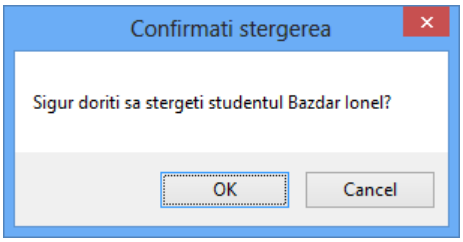


Figura 5. Confirmarea ștergerii

- Salvarea va presupune trecerea din Starea 2 în Starea 1 și actualizarea modificărilor făcute în *DataTable*, în tabela considerată:

```
BindingSource1.EndEdit();
adaptor.Update(dt);
```

- Renunțarea presupune trecerea din Starea 2 în Starea 1 și anularea modificărilor realizate în *BindingSource*:

```
BindingSource1.CancelEdit();
```

În continuare se va implementa fereastra pentru căutarea unui student (Figura 6). Aici, controlul *ComboBox* corespunzător numelui câmpului va fi completat prin cod cu numele tuturor câmpurilor din tabelul Studenti.

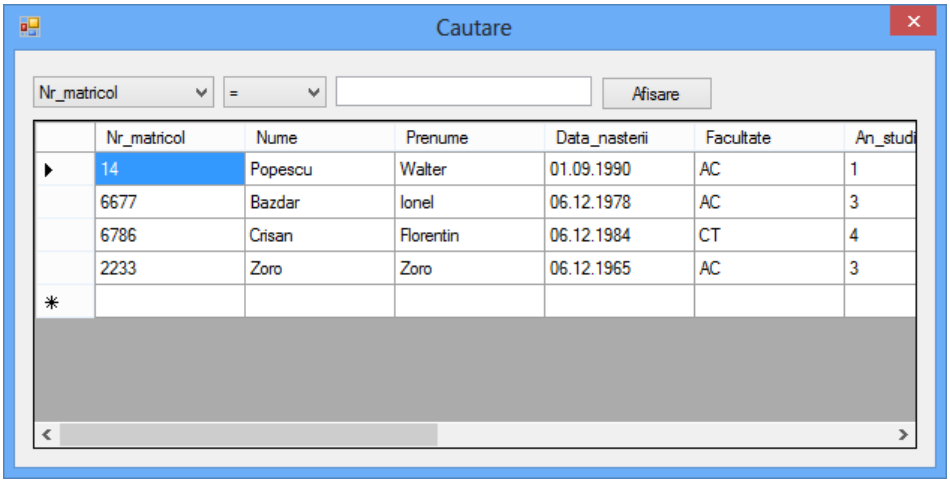


Figura 6. Căutarea după un câmp

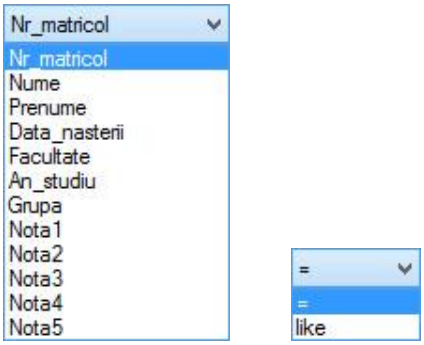


Figura 7. Valori luate de cele două controale ComboBox în vederea efectuării unei căutări

Fereastra va conține un control *DataGridView*, care inițial va afișa toate înregistrările din baza de date. În urma apăsării butonului Afișare se vor afișa în *DataGridView* doar înregistrările care respectă condițiile precizate ("câmp = valoare" sau „câmp like valoare").

Pentru încărcarea controlului *DataGridView* cu toate datele din tabela bazei de date se va proceda în mod similar exemplului de mai sus cu mențiunea că obiectele de tip *BindingNavigator*, *BindingSource* și *OleDbCommandBuilder* nu mai sunt necesare.

Pentru încărcarea datelor din *DataTable* în *DataGridView* se va proceda în felul următor:

```
DataGridView1.DataSource = dt;
DataGridView1.ReadOnly = true;
```

Inițial *DataGridView* va conține înregistrările din întreaga tabelă. Interogarea SQL va fi construită dinamic, astfel încât dacă se apasă butonul afișare și nu este introdus nimic în câmpul pentru valoare, în *DataGridView* va fi afișat conținutul întregii tabele, iar altfel se vor afișa în *DataGridView* doar înregistrările filtrate prin interogare. În cazul în care interogarea SQL nu este corectă se va afișa un mesaj de eroare, urmat de afișarea comenzii în clar. Pentru aceasta se va folosi blocul `try…catch`.

Sfaturi utile

1. *Pentru a asigura comutarea între cele două stări complementare se recomandă crearea unei funcții de dezactivare/activare a controalelor, care are ca și parametru o variabilă boolean, corespunzătoare stării.*
2. *Pentru includerea unei confirmări suplimentare la operația de ștergere, se recomandă renunțarea la acțiunea implicită pe care o are butonul de ștergere în cadrul BindingNavigator-ului și editarea manuală a codului necesar ștergerii.*
3. *În fereastra de căutare, pentru completarea numelor tuturor câmpurilor din tabelul Studenti în controlul ComboBox se recomandă parcurgerea prin intermediul unei bucle `foreach` a tuturor coloanelor din tabelul creat și adăugarea prin cod a denumirilor acestora.*
4. *Se recomandă transmiterea obiectului dt din fereastra principala către fereastra de căutare prin constructorul acesteia.*

Salvarea în tabelă a modificărilor făcute se poate realiza și fără utilizarea unui obiect *OleDbCommandBuilder* și a metodei `Update` a adaptorului, prin construirea dinamică a comenzii SQL (INSERT, DELETE, UPDATE) și executarea ei, la fel ca în exemplul de mai jos:

```
// se construieste interogarea insert, update, sau delete
command.CommandText = sql
command.ExecuteNonQuery()
```

Cu ce ne-am ales?

Prin aplicația dezvoltată am învățat să realizăm aplicații desktop pentru lucrul cu baze de date și să implementăm principalele operații care apar în aplicațiile cu baze de date: adăugare, modificare, ștergere sau căutare. Aplicațiile cu baze de date sunt utile și necesare în toate domeniile. Aplicația dezvoltată operează asupra unei baze de date Access dar tehnologia OLEDB poate fi aplicată și altor baze de date precum Oracle, Microsoft SQL Server, MySQL etc.

Bibliografie

[1] Visual C# Resources, http://msdn.microsoft.com/en-us/vstudio/hh341490%28v=msdn.10%29.aspx