Laboratorul 4

Operații cu fișiere și directoare în aplicațiile Visual C# .NET

Ce ne propunem astăzi?

Aplicația propusă pentru acest laborator are rolul de a ține evidența jucătorilor de fotbal de la cluburile afiliate Ligii Profesioniste de Fotbal. Vor putea fi adăugate echipe noi și jucători noi, iar datele acestora vor fi validate înainte de salvare.

Aplicația propusă (Figura 1) operează cu date salvate pe disc sub forma unei ierarhii de directoare și fișiere. Astfel, un director reprezintă numele unei echipe. El conține fișiere text cu datele jucătorilor legitimați la echipă, câte un fișier distinct pentru fiecare jucător.

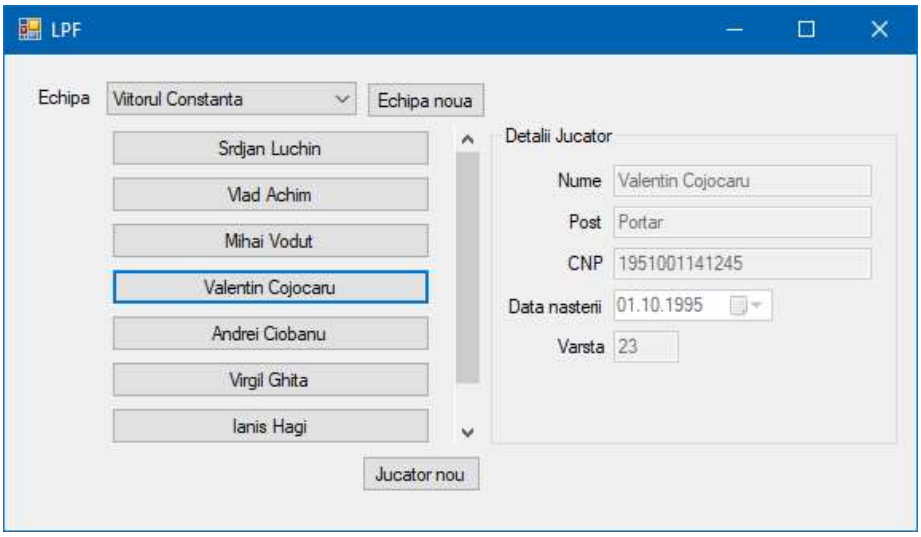

Figura 1. *Fereastra principală a aplicației*

Datele unui jucător sunt:

- Nume (numele complet)
- CNP
- Post (postul pe care joacă: enumerare cu valorile Portar, Fundaș, Mijlocaș, Atacant)

Fișierul cu datele unui jucător are numele identic cu CNP-ul jucătorului, extensia „.lpf” și conține pe linii diferite numele și postul jucătorului (Figura 2).

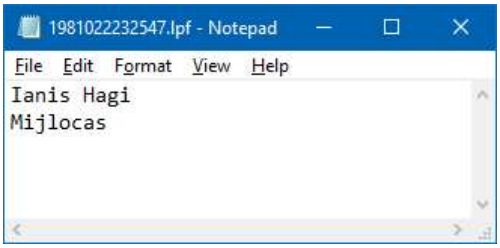

Figura 2. *Fișierul cu datele unui jucător*

Mod de lucru sugerat

Pentru încărcarea numelor echipelor în aplicație trebuie să aflați numele subdirectoarelor din directorul de lucru.

La selectarea unei echipe se vor citi toate fișierele cu extensia „.lpf" din directorul asociat echipei selectate și se vor construi dinamic controale de tip Button, câte unul pentru fiecare jucător, și se va adăuga un handler pentru evenimentul Click.

La click pe un buton asociat unui jucător vor fi afișate datele acestuia. În această etapă va fi extrasă din CNP data nașterii, iar pe baza acesteia va fi calculată vârsta în ani împliniți la data curentă.

Operații cu fișiere și directoare

Pentru a încărca datele despre echipe și jucători va fi nevoie să operați cu fișiere și directoare, astfel că trebuie importat spațiul de nume System.IO. Iată o listă de operații care vă vor fi utile:

- Aflarea tuturor directoarelor existente la calea indicată:

```
Directory.EnumerateDirectories("D:\\LPF")
```

- Extragerea numelui unui director (eliminare căii complete):

```
new DirectoryInfo("D:\\LPF\\FCSB").Name
```

- Aflarea tuturor fișierelor dintr-un director (se pot filtra fișierele după extensie):

```
Directory.EnumerateFiles("D:\\LPF\\FCSB", "*.lpf")
```

- Extragerea numelui unui fișier (eliminarea căii complete și a extensiei):

```
Path.GetFileNameWithoutExtension("D:\\LPF\\FCSB\\1850512457852.lpf")
```

- Verificarea existenței unui fișier:

```
File.Exists("D:\\LPF\\FCSB\\1931223234567.lpf")
```

- Verificarea existenței unui director:

```
Directory.Exists("D:\\LPF\\FCSB")
```

- Crearea unui director

```
Directory.CreateDirectory("D:\\LPF\\FCSB")
```

Validarea datelor

Datele trebuie să fie validate înainte de salvarea lor. În acest sens folosiți componenta ErrorProvider.

Pentru a semnala o eroare în dreptul unui control (textBox1) apelați metoda SetError și indicați mesajul de eroare:

```
errorProvider1.SetError(textBox1, "Numele trebuie completat");
```

Pentru a reseta o eroare apelați metoda Clear:

```
errorProvider1.Clear();
```

Sunt obligatorii urătoarele validări ale datelor în aplicație:

- Numele, postul și CNP-ul jucătorului să fie completate (Figura 3).
- CNP să conțină fix 13 caractere.
- Data de naștere extrasă din CNP să fie validă. Pentru validarea datei de naștere puteți folosi constructorul structurii DateTime cu parametrii (an, lună zi). În caz de dată invalidă va genera o eroare pe care o puteți trata într-un bloc try – catch.
- Numele echipei să fie completat și acea echipă să nu fie deja introdusă în aplicație.

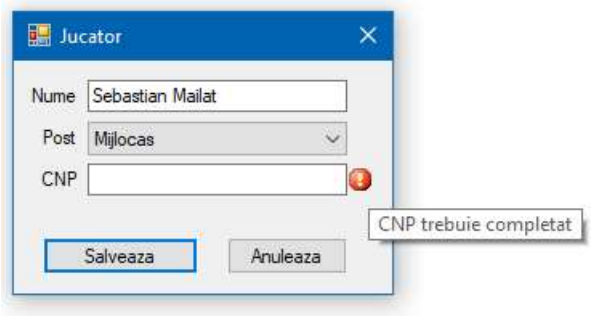

Figura 3. *Validarea CNP-ului unui jucător*

Accesul restricționat la datele aplicației

Considerăm că datele din aplicație sunt confidențiale, astfel că accesul la acestea ar trebui să fie protejat prin parolă. În mod obligatoriu, accesul în aplicație se va face prin autentificarea utilizatorului. În acest sens se va crea un nou formular de autentificare (vezi Figura 4).

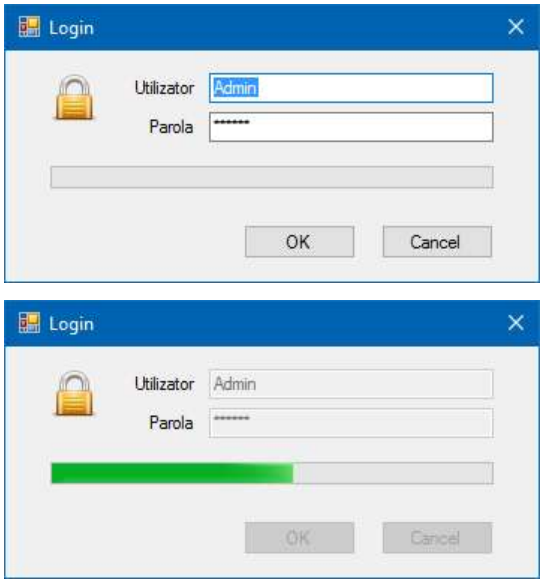

Figura 4. *Fereastra de autentificare a utilizatorilor în aplicație, înainte și după autentificare*

Pentru acest formular se vor seta proprietățile `MaximizeBox` și `MinimizeBox` pe *false*, iar proprietatea `FormBorderStyle` la valoarea *FormBorderStyle.FixedDialog*.

Pentru crearea și utilizarea acestei ferestre trebuie precizate câteva lucruri.

1. Pentru afișarea ferestrei de autentificare înaintea ferestrei principale a aplicației, instanțiați și deschideți fereastra în handlerul evenimentului Load al ferestrei principale:

```
FormLogin f = new FormLogin();
if (f.ShowDialog() == DialogResult.Cancel)
```

```
    this.Close();
```

2. Controlul TextBox asociat parolei va trebui să ascundă textul introdus prin setarea proprietății `PasswordChar` a acestuia.
3. Dacă s-a apăsat butonul OK, iar numele utilizatorului și parola de acces sunt corecte, atunci cele două casete text și butoanele OK și Cancel vor fi dezactivate, trecându-se la pasul următor.
4. Fereastra va utiliza o componentă Timer și un control de tip ProgressBar (indică grafic progresul unei operații). Se vor utiliza proprietățile `Minimum` (0), `Maximum` (20) și `Value` (inițial având valoarea 0). Se dorește ca, la autentificarea cu succes a utilizatorului, controlul ProgressBar să se „umple" treptat, indicând posibile operații efectuate în fundal (inițializări diverse, încărcare de date etc.)
5. Pentru componenta Timer va fi setată proprietatea `Interval` la valoarea 100. Aceasta indică numărul de milisecunde scurs între generarea a două evenimente Tick. La autentificarea utilizatorului, dacă numele utilizatorului și parola au fost introduse corect, se va activa componenta Timer (metoda `Start()` a acesteia), iar din acest moment, la fiecare 100 ms va fi generat câte un eveniment Tick. Handlerul acestui eveniment poate fi utilizat pentru a indica operații care trebuie efectuate periodic, cu o anumită temporizare. În cazul nostru, operația aceasta este cea de incrementare a valorii controlului ProgressBar, până la atingerea valorii maxime, situație în care componenta Timer va fi dezactivată (metoda `Stop()` a acesteia), fereastra de autentificare va fi închisă, iar fereastra principală a aplicației va fi afișată.
6. Pentru ca fereastra de autentificare să se comporte precum o fereastră dialog, trebuie să putem folosi valoarea returnată de metoda `ShowDialog()` a acesteia astfel că la părăsirea ferestrei veți seta corespunzător proprietatea `DialogResult` a formularului (acest lucru va avea ca efect și închiderea formularului).

```
this.DialogResult = DialogResult.OK;
```

Cu ce ne-am ales?

Prin aplicația dezvoltată în cadrul laboratorului de astăzi am reușit să ne familiarizăm cu lucrul cu fișiere și directoare, cu crearea dinamică a controalelor și a handler-elor de eveniment. Am învățat de asemenea să validăm datele și să le semnalăm pe cele eronate prin componenta ErrorProvider.

Bibliografie

[1] Codul Numeric Personal, https://ro.wikipedia.org/wiki/Cod_numeric_personal

[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