Capitolul 4 - Prezentarea aplicației

4.1. Arhitectura aplicației

Aplicația este alcătuită din două proiecte mari și unul mai mic care oferă o interfață în WPF (Windows Presentation Foundation) pentru autentificarea utilizatorului (**Figura 4.1).**

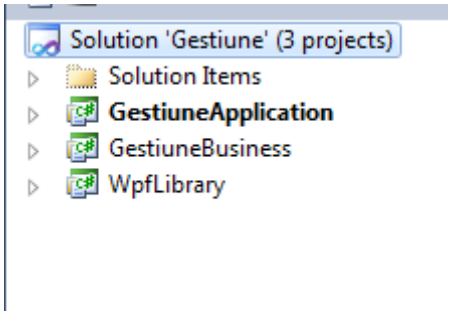

Figura 4.1. *Proiectele aplicației*

Proiectul care conține toată logica aplicației se numește *GestiuneBusiness* și conține toate clasele necesare pentru lucrul cu baza de date și obiectele sinonime din baza de date.

Fiecare tabelă are asociată câte o clasă în proiectul *GestiuneBusiness*, fiecare astfel de clasă are aceleași proprietăți ca și tabela pe care o reprezintă, dar mai moștenește clasa abstractă *GestiuneObject* al cărei cod este următorul:

```
public abstract class GestiuneObject : IDataPersistence
  {
    public int ID { get; set; }

    public abstract string NumeCompact { get; }

    public abstract PersistenceResult Save();

    public bool Contains(string text)
    {
      // verifică dacă atributele clasei moștenitoare conține textul 'text'
    }
```

```
    public string GetErrorString()
    {
       // returnează un text în care specifică dacă atributele au valoare sau nu
    }

    public abstract List<DbObject> PropertiesNamesWithValues { get; }

    protected const string StringSaveSuccess = "Salvare efectuata cu succes!";

    protected const string StringSaveFail = "Au aparut erori in timpul salvarii! Verificati
datele!";
  }
```

ID reprezintă cheia primară a fiecărei tabele, ea se moștenește în fiecare clasă care extinde clasa *GestiuneObject***.**

PropertiesNamesWithValues este o proprietate abstractă care lasă clasa moștenitoare să decidă care este lista atributelor ei. De exemplu, în cazul unui delegat, această proprietate va returna Nume, CNP și Serie.

Toate aceste clase inspirate după tabele din baza de date se salvează și se actualizează prin intermediul claselor derivate din clasa abstractă *GestiuneDataHelper:*

```
 internal abstract class GestiuneDataHelper
  {
    protected string selectAllStoredProcedureName = "";

    protected string insertStoredProcedureName = "";

    protected string updateStoredProcedureName = "";

    private const string StringProcedureFail = "Stored procedure failed: ";
```

```
    private const string StringDatabaseFail = "Connection to database failed.";

    public List<GestiuneObject> GetAll()
    {
       // metodă care se leagă la baza de date și aduce toate înregistrările dintr-o tabelă
    }

    public int Create(List<DbObject> dbObjects)
    {
       // metodă care se leagă la baza de date și salvează o înregistrare dintr-o tabelă
    }

    public void Update(List<DbObject> dbObjects, int id)
    {
        // metodă care se leagă la baza de date și actualizează o înregistrare dintr-o tabelă
    }

    protected abstract GestiuneObject ToPocoObject(SqlDataReader reader);
}
```

Clasele care derivează din *GestiuneDataHelper* se ocupă cu persistența bazei de date și transformarea entităților citite în clase sinonime și sunt singleton deoarece instanțierea lor de mai multe lor nu ar avea sens, este de preferabil să există o singură instanță a fiecărei clase *DataHelper* pe întreaga aplicație. Pentru fiecare clasă sinonimă cu o tabelă din baza de date există o clasă de tip *DataHelper.*

Metoda *protected abstract GestiuneObject ToPocoObject* este abstractă deoarece lasă clasa moștenitoare să transforme datele citite din baza de date în obiectul care se dorește a fi creat.

Lucrul cu clasele data helper se întâmplă exclusiv în clasele care derivă *GestiuneObject*, ele sunt complet transparente în afara proiectului. Acest proiect utilizează

încărcarea datelor înainte de a fi folosite în niște liste, care rețin, ca un cache, toate datele. Această abordare reduce semnificativ numărul apelurilor claselor care se ocupă cu citirea datelor din baza de date și transformarea lor în obiecte, mai exact apelul claselor *DataHelper*. Această încărcare are loc o singură dată după prima autentificare a unui utilizator prin apelul funcției GetAll() care încarcă cache-ul cu datele din baza de date. Iar pe măsură ce obiectele se salvează cu succes ele se adaugă automat în acest cache.

Proiectul aplicației principale se numește *GestiuneApplication*. Rolul ei este să apeleze funcționalitățile proiectului de business și să le dispună corespunzător pentru utilizator. Singurele clase care sunt complet dependente de acest proiect sunt cele pentru realizarea rapoartelor.

Cel de-al treilea proiect: *WpfLibrary* conține o interfață grafică pentru autentificarea utilizatorului. Această interfață este realizată folosind limbajul de marcare XAML:

```
<Border CornerRadius="10"  BorderThickness="3"  Background="White"  Padding="20"
Margin="4"  BorderBrush="Black">
    <Border.Effect>
      <DropShadowEffect  Color="Gray" Opacity=".50"  ShadowDepth="6"/>
    </Border.Effect>
    <Grid Background="White"
      FocusManager.FocusedElement="{Binding ElementName=usernameTbox}">
        <Label Grid.Row="0" Grid.Column="0"   Grid.ColumnSpan="2"  FontSize="24"
      HorizontalAlignment="Center"
     VerticalAlignment="Center">Introduceti datele pentru autentificare</Label>
     <Label Grid.Row="1"
            Grid.Column="0" FontSize="20"  HorizontalAlignment="Right"
     VerticalAlignment="Center">Utilizator</Label>
     <Label Grid.Row="2"
            Grid.Column="0" FontSize="20"  HorizontalAlignment="Right"
            VerticalAlignment="Center">Parola</Label>
     <TextBox Grid.Row="1"
            Grid.Column="1" Margin="10" Name="usernameTbox"
```

```
                ToolTip="Introduceti numele utilizatorului" TabIndex="0"/>
        <PasswordBox Grid.Row="2"
                Grid.Column="1" Margin="10" Name="pwdBox"
                ToolTip="Introduceti parola utilizatorului" TabIndex="1" />
        <Button Grid.Column="0" Grid.Row="3" Margin="10" Name="exitBtn"
IsCancel="True"
            Content="Iesire" Click="exitBtn_Click" TabIndex="3">
            <Button.Effect>
               <DropShadowEffect Color="Gray" Opacity=".50" ShadowDepth="8" />
            </Button.Effect>
        </Button>
        <Button Grid.Column="1" Grid.Row="3"  HorizontalAlignment="Right"
            Width="80" Margin="10" Name="loginBtn"
            Content="Intrare"
                Click="loginBtn_Click"
                IsDefault="True"
                IsCancel="False"
                TabIndex="2">
            <Button.Effect>
               <DropShadowEffect Color="Gray" Opacity=".50" ShadowDepth="8" />
            </Button.Effect>
        </Button>
      </Grid>
   </Border>
```

Această clasă conține și un eveniment care este generat în momentul în care se face click pe butonul de *Intrare* al interfeței (**Figura 4.2)**.

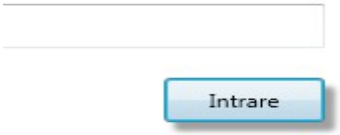

Figura 4.2. *Butonul de intrare de pe interfața de autentificare*

Declarerea evenimentului pentru generarea acestuia se face declarând mai întâi un delegat, după care evenimentul va fi de tipul acelui delegat, iar generarea acestuia se face apelând *OnLoginClick* cu parametrii corespunzători:

```
public delegate void ChangedEventHandler(string username, string password);

public event ChangedEventHandler OnLoginClick;

  private void loginBtn_Click(object sender, RoutedEventArgs e)
      {
         OnLoginClick(usernameTbox.Text, pwdBox.Password);
         pwdBox.Password = string.Empty;
      }
```

Evenimentul semnalizează faptul că s-a făcut click pe buton şi trimite numele utilizatorului și parola metodei care este înregistrată la acest eveniment. De exemplu în forma principală din proiectul *GestiuneApplication*:

```
       private void loginWindow_OnLoginClick(string username, string password)
      {
         // validarea utilizatorului
      }
```

Înregistrare unei metode la un eveniment se face prin intermediul unui delegat:

```
loginWindow.OnLoginClick += new ChangedEventHandler(loginWindow_OnLoginClick);
```

Prin înregistrarea metodei *loginWindow_OnLoginClick* la evenimentul de click pe butonul de intrare se asigură faptul că de fiecare dată când un utilizator va da click pe butonul de intrare se va apela metoda *loginWindow_OnLoginClick,* din proiectul principal, pentru a se face validarea acestuia.