**Iulica Mariana Enăchescu**

**Academiei de IT Wantsome**

**Proiect final .NET**

**SCHOOL MANAGEMENT APP**

Proiectul School Management App este conceput pentru a realiza o aplicație web pentru managementul și organizarea activităților educaționale din cadrul unei școli de limbi străine.

Acest proiect are un caracter didactic și are ca scop implementarea noțiunilor de programare orientată obiect și a noțiunilor de programare .NET, acumulate pe parcursul orelor desfășurate în cadrul Academiei de IT Wantsome (pornind de la noțiuni elementare de programare și evoluând spre abordarea unor topicuri de programare avansată).

Aplicația dezvoltată în cadrul proiectului este o aplicație de tip Client-Server, în care comunicarea de de date se face după modelul request-service, prin intermediul unei interfețe. Aceasta a fost scrisă în .Net Framework, limbajul C#.

Aplicația utilizează și o componentă de stocare de date pe server (persistence layer), pentru care am apelat la serverul de date Microsoft SQL Server Management.

Pentru organizarea și dezvoltarea codului m-am bazat pe modelul arhitectural Three Layer Architecture, care are ca scop separarea programului pe trei nivele (3 Tier), după cum urmează:

* Nivelul data-layer -pentru gestionarea exclusive a conexiunii cu baza de date și a operațiilor CRUD (create, read, update, delete) pentru fiecare tabel de date stocat pe serverul de date fiind concepută o clasă specifică pentru realizarea operațiilor SQl prin intermediul codului .Net
* Nivelul business-logic – care are ca rol coordonarea aplicației și implementarea de calcule și de decizii logice. În acest modul au fost implementate diferite operații de prelucrare a datelor preluate de pe server prin operațiile CRUD. La acest nivel sunt definite entitățile de bază și sunt mapate entitățile în modele pentru afișarea pe interfața utilizatorului (ModelView)
* Nivelul prezentare (presentation layer), în care este implementată interfața grafică pentru utilizator.

La nivelul Data-Layer conexiunea la baza de date este realizată prin intermediul tehnologiei ADO Net. Pentru asigurarea unei singure conexiuni cu baza de date, care este reutilizată pentru fiecare operație Sql a fost folosit patternul de design Singleton (prin transferarea responsabilității creării și gestionării conexiunii unei clase special construite cu acest scop și care prezintă o singură instanță, prin realizarea unor constrângeri logice prin intermediul constructorului). În vederea securizării accesului la baza de date și a evitării unor incidente de tip SQL Injection, toate operațiile query au fost trasmise către serverul SQL prin parametri.

Pentru a încapsula logica necesară care asigură accesul la baza de date de restul aplicației, dar și pentru a spori modularitatea aplicației, am utilizat patternul Repository Design Pattern**,** prin proiectarea unor interfețe specifice pentru a fi implementate de fiecare clasa asociată DataAccess. De asemenea, am fost conceput un **generic repository,** cu scopul de a maximiza reutilizarea codului prin care se realizează operațiile CRUD, în concordanță cu principiul de programare DRY.

Operațiile de business logic și de prelucrare și agregare a datelor preluate din baza de date au fost realizate cu ajutorul tehnologiei LINQ-Language-Integrated Query.

În vederea asigurării unui nivel optim de modularitate și al reducerii cuplării între componentele aplicației, am utilizat de asemenea si paternul dependency injection, care a fost implementat prin injectarea dependențelor prin constructor, în clasele de tip Service și în controlleri.

Pentru dezvoltarea interfeței web cu utilizatorul am utilizat tehnologia ASP.Net-MVC (oferită de Microsoft în cadrul mediului de dezvoltare Visual Studio .Net). Această tehnologie se bazează pe modelul Model-View-Controller, care este cel mai utilizat în dezvoltarea de aplicații web. Avantajul acestui patern este acela că permite decuplarea dintre partea vizuală (respective interfața cu utilizatorul) de date și de logica aplicației, precum și o bună organizare a codului, care permite dezvoltarea de aplicații care au un nivel de complexitate ridicat. Modului în care este implementat componenta APS .Net nu conține decât componentele Controller și View, responsabilitățile componentei Model fiind preluate de către controller de la celelalte două nivele ale aplicației (respectiv DataAccess și BusinessLogic). În interfața grafică, datele sunt preluate de la utilizatorii prin formulare, care au integrate validări și care transmit datele către controller prin Http-Post.

Aplicația este împărțită în următoarele subproiecte:

1. **SchoolDBModel** -conține modelele utilizate pentru operațiile de acces la baza de date și de logică și prelucrare

2. **SchoolWebAppView-** conține ViewModels, utilizate pentru afișarea datelor pe interfața cu utilizatorul, precum și mapările între entitity-models și view-models

3. **SchoolWebAppData**- care are ca responsabilitate realizarea conexiunii cu baza de date și realizarea operațiilor CRUD la baza de date

4. **SchoolWebAppBusinessService**care are ca responsabilitate implementarea de operații de logică, de prelucrare și agregare date

5. **SchoolWebApp** care conține codul ASP .Net MVC, pentru realizarea interfeței cu utilizatorul

6. **SchoolWebAppTests** care conține UnitTests

Aplicația este proiectată a fi utilizată de trei tipuri de utilizatori, care vor beneficia, diferențiat de următoarele servicii:

**Administrator**:

* Adaugă, șterge, modifică, caută și vizualizează tipurile de cursuri
* Adaugă, șterge, modifică, caută și vizualizează cursurile în desfășurare.
* Vizualizează orarele cursurilor și mesajele transmise pentru fiecare curs
* Adaugă, șterge, modifică, caută și vizualizează cursanții. Manageriază plațile cursanților, istoricul fiecărui cursant. Vizualizează prezența studenților și evaluările cursanților
* Adaugă, șterge, modifică, caută și vizualizează persoanele de contact pentru fiecare cursant
* Adaugă, șterge, caută și vizualizează profesorii

**Profesor:**

* Caută și vizualizează tipurile de cursuri
* Caută și vizualizează cursurile în desfășurare
* Adaugă, șterge, modifică, caută și vizualizează orarele cursurilor
* Adaugă, șterge, modifică, caută și vizualizează mesajele transmise pentru fiecare curs
* Adaugă, șterge, modifică, caută evaluările participanților la curs
* Adaugă, șterge, modifică, caută prezența la cursuri, pentru fiecare cursan
* Caută și vizualiează cursanții
* Modifică parola personală

**Student:**

* Modifică parola personală
* Vizualizează informațiile personale
* Vizualizează informațiiloe școalare: istoric școlar, prezență, mesaje și evaluări
* Vizualizează situațiile plăților

**DBO TABLES:**

1.     User: Id(primary key, identity), UserName, LastName, FirstName, Password, DateBirth, CreateDate, City, Adress, Phone, Email

2.     Admin: Id(primary key, identity), UserId(foreign key to User by Id), Role

3. Student: Id(primary key, identity), UserId(foreign key to User by Id), Status active, ContactId (foreign key to ContactPerson by Id)

4. ContactPerson: Id(primary key, identity), Name, Adress, Phone, Email

5. Teacher: Id(primary key, identity), UserId(foreign key to User by Id), Status active

6. Course: Id(primary key, identity), NumberOfLessons, Description, Category, Language, Level, StatusActive

7. Class: Id(primary key, identity), CourseId(foreign key to Course by Id), TeacherId(foreign key to Teacher by Id), Name, Descriptin, StartDate, EndDate

8. StudentClass: Id(primary key, identity), ClassId(foreign key to Class by Id), StudentId(foreign key to Student by Id) -manage m-n relationship between Class and Student

9. ClassTimeTable: Id(primary key, identity), ClassId(foreign key to Class by Id), LessonNumber, Date

10. ClassMessage: Id(primary key, identity), ClassId(foreign key to Class by Id), Date, Message

11. StudentPresence: Id(primary key, identity), TimeTableId(foreign key to TimeTable by Id), Presence

12. StudentClassEvaluation: Id(primary key, identity), ClassId(foreign key to Class by Id), StudentId(foreign key to Student by Id), Description, Grade

13. StudentPayment: Id(primary key, identity), StudentId(foreign key to Student by Id), ClassId(foreign key to Class by Id), PaymentDate, Ammount