



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

INGINERIE SOFTWARE

Documentație proiect

Gestiunea unei săli de fitness

Student: Adela Iosif

Grupa: 30233

2023-2024

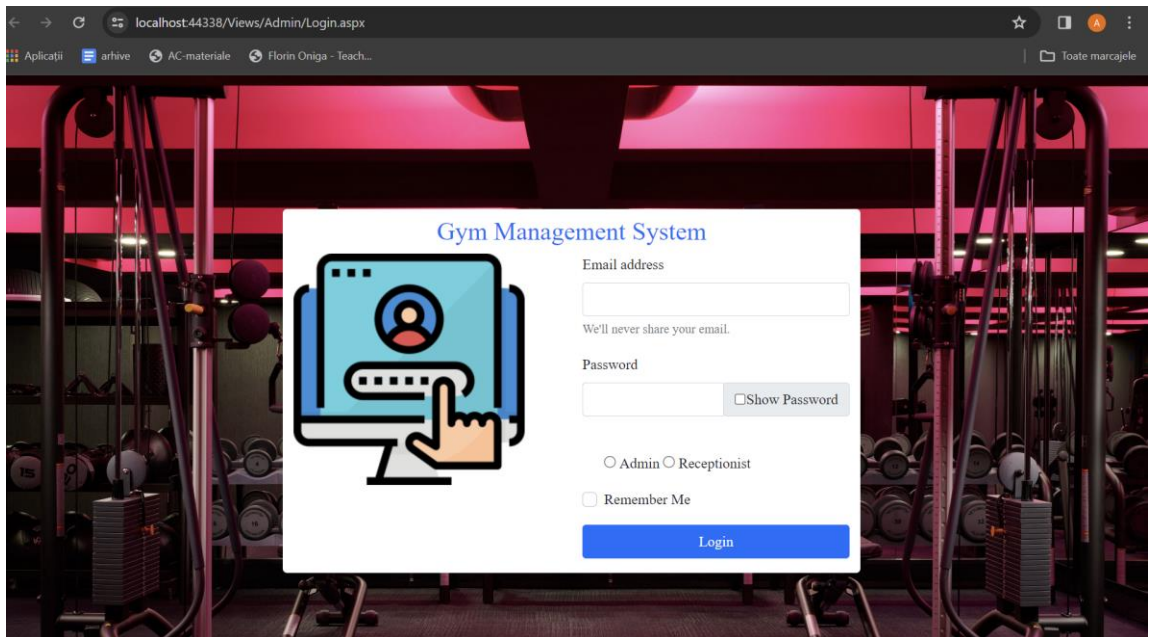
Cuprins

1. Introducere.....	3
1.1. Funcționalități.....	3
1.2. Diagramă use case	5
2. Modelare interacțiuni și comportament – diagrama de secvențiere	6
3. Design Pattern	6
4. Proiectare.....	7
4.1. Tehnologii.....	7
4.2. Baza de date.....	8
5. Ghid de utilizare	9
6. Concluzii, bibliografie și dezvoltări ulterioare.....	9

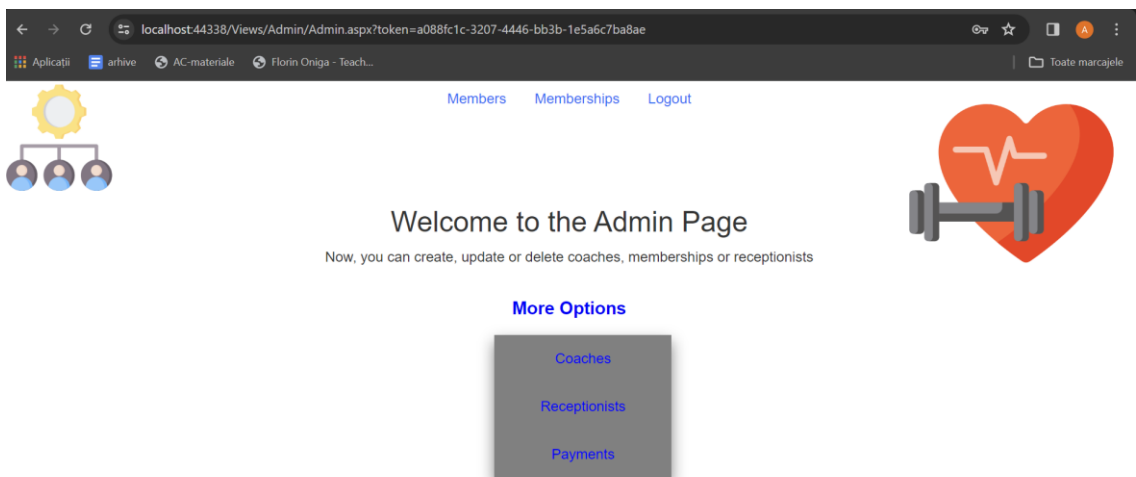
1. Introducere

1.1. Funcționalități

Aplicația web dezvoltată de mine se ocupă de gestionarea unei săli de fitness. Astfel, există posibilitatea logării în cont ca administrator sau recepționar, bifând opțiunea corespunzătoare cu ajutorul unui radio button. Am creat un singur cont principal de administrator, acesta având doar posibilitatea creării unor noi recepționeri (aceștia nu își pot crea singuri cont).

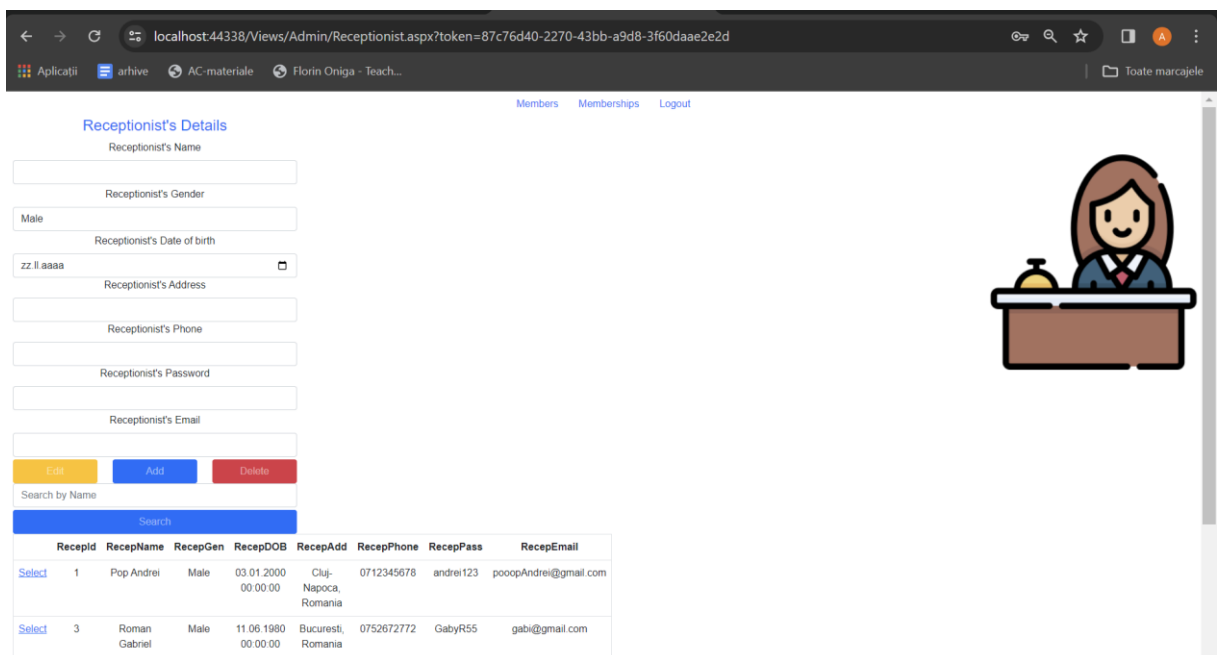


Administratorul poate să adauge, să editeze sau să ștergă antrenori, planuri de antrenament, membri sau recepționeri. Astfel, în cazul în care unul dintre recepționeri sau un antrenor va părăsi firma, adminul îl poate șterge, și deci, respectivul recepționar/antrenor nu va mai avea acces la cont. Totodată, administratorul poate adăuga plăți.



Mai mult, pentru a-i ușura munca și pentru că există posibilitatea existenței unui număr foarte mare de antrenori sau recepționeri, am creat opțiunea de căutare. Astfel, introducând numele recepționarului/antrenorului/membrului în câmpul de „search” din fereastra corespunzătoare, se va găsi persoana căutată în cel mai scurt timp. De asemenea, există și opțiunea de delogare (ieșire din cont).

Dacă intrarea în cont se face în rolul de recepționar, va exista posibilitatea vizualizării, editării, adăugării și ștergerii unui recepționar, membru sau abonament. De asemenea, există posibilitatea de căutare după nume, atât pentru recepționeri, cât și pentru membri.



The screenshot shows a web browser window with the URL `localhost:44338/Views/Admin/Receptionist.aspx?token=87c76d40-2270-43bb-a9d8-3f60daae2e2d`. The page title is "Receptionist's Details". On the right, there is a cartoon illustration of a receptionist at a desk. The main content area contains a form for adding or editing a receptionist's details, with fields for Name, Gender, Date of birth, Address, Phone, Password, and Email. Below the form are buttons for "Edit", "Add", and "Delete". A search bar labeled "Search by Name" is also present. At the bottom, there is a table with the following columns: "RecepId", "RecepName", "RecepGen", "RecepDOB", "RecepAdd", "RecepPhone", "RecepPass", and "RecepEmail". The table contains two rows of data.

	RecepId	RecepName	RecepGen	RecepDOB	RecepAdd	RecepPhone	RecepPass	RecepEmail
Select	1	Pop Andrei	Male	03.01.2000 00:00:00	Cluj-Napoca, Romania	0712345678	andrei123	popoandrei@gmail.com
Select	3	Roman Gabriel	Male	11.06.1980 00:00:00	Bucuresti, Romania	0752672772	GabyR55	gabi@gmail.com

În plus, am încercat securizarea procesului de logare, prin token unic, care se creează după intrarea cu succes în cont a utilizatorului (admin sau recepționar). Am reușit să generez un token unic pentru fiecare logare reușită.

```
// Generate a unique session token for the receptionist user
string authToken = Guid.NewGuid().ToString();

// Store the token in the session with a prefix indicating the user type
Session["RecepAuthToken"] = authToken;

// Redirect to the receptionist page with the token in the URL
Response.Redirect("/Views/Admin/Receptionist.aspx?token=" + authToken);
```

Ceea ce aş fi vrut să fac, de fapt, este salvarea acestui url unic în local storage, iar la încercarea copierii link-ului într-un alt browser, să redirecţioneze utilizatorul la pagina iniţială de login. Nu am reuşit acestea, dar algoritmul spre obţinerea securităţii, pare să fie bine gândit.

```
function SetTokenInLocalStorage(token) {
    localStorage.setItem("AuthToken", token);
}

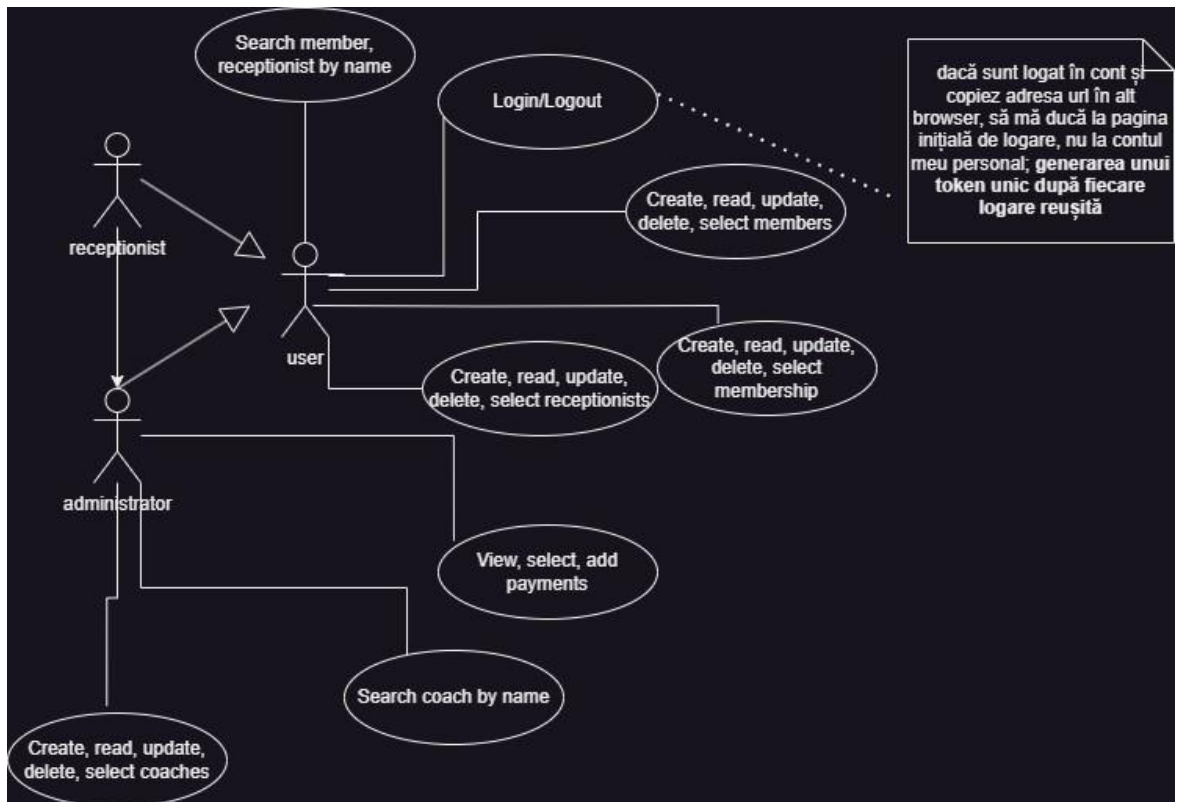
private void CheckAccess()
{
    // Get the user type (Admin or Recep)
    string userType = GetUserType(); // Implement a method to determine the user type

    // Get the session token from the URL or session
    string requestToken = GetRequestToken();

    // Get the currently authenticated user's token
    string authenticatedToken = Session[userType + "AuthToken"] as string;

    // Check if the tokens match
    if (string.IsNullOrEmpty(requestToken) || requestToken != authenticatedToken)
    {
        // Redirect to the main login page
        Response.Redirect("/Views/Admin/Login.aspx");
    }
}
```

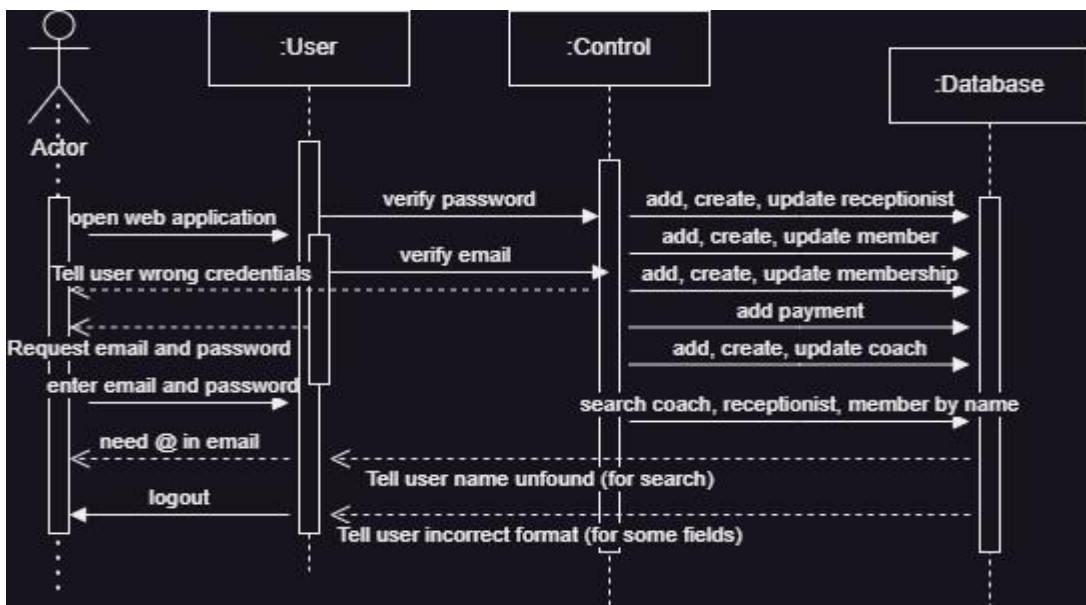
1.2.Diagramă use case



2. Modelare interacțiuni și comportament – diagrama de secvențiere

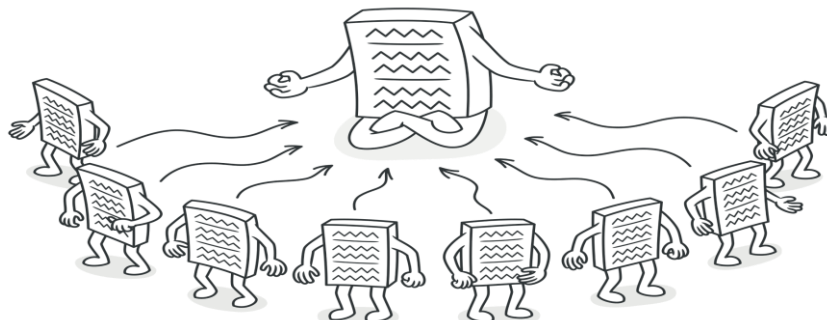
Diagramele de interacțiune vizualizează modul în care un grup de obiecte și actori comunică pentru a efectua acțiunile corespunzătoare: unui caz de utilizare sau unui alt fragment de funcționalitate. Aceste diagrame vizualizează secvențe de mesaje. Elementele diagramelor de interacțiune sunt: obiecte, actori, mesaje – reprezentate sub formă de săgeți (etichete). Pentru vizualizarea interacțiunilor se utilizează două tipuri de diagrame: diagrame de secvențiere și diagrame de comunicare.

În continuare, eu am ales folosirea unei diagrame de secvențiere. O diagramă de secvențiere vizualizează secvența de mesaje schimbate de un grup de obiecte cu scopul efectuării unei anumite sarcini de calcul.



3. Design Pattern

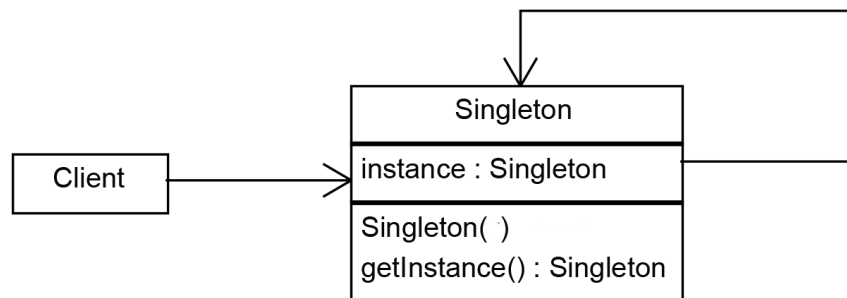
Design pattern-urile sunt moduri de structurare a codului pentru a rezolva o problemă generală în programare. Singleton este un șablon de proiectare care se folosește atunci când dorim să avem o singură instanță la nivelul aplicației, acesta furnizând un punct global de acces la instanța respectivă.



Deci, Singleton rezolvă două probleme:

- se asigură că o clasă are o singură instanță – în general, pentru a controla accesul la niște resurse comune
- oferă un punct de acces global la această instanță – la fel ca și o variabilă globală, șablonul de proiectare Singleton oferă acces unor obiecte, de oriunde din program; totuși, protejează instanța de a fi suprascrisă

Soluția propusă de Singleton presupune declararea constructorului implicit privat, pentru a împiedica alte obiecte să folosească noul operator cu clasa Singleton.



4. Proiectare

4.1. Tehnologii

Am dezvoltat o aplicație web cu framework-ul ASP.NET WebForms, acesta folosind modelul de programare eveniment-driven (interacțiunea și fluxul logic al programului sunt determinate de evenimente externe; evenimente care apar în timpul execuției), bazat pe conceptul de postback-uri. În acest context, evenimentele pot fi acțiuni precum apăsarea unui buton, trimiterea unui formular sau alte interacțiuni făcute utilizator. Astfel, în ASP.NET WebForms, atunci când are loc un eveniment, întregul formular este trimis la server (prin intermediul unui postback), iar codul de server procesează acțiunea și returnează o nouă versiune a paginii la client.

Ca și limbaj de programare principal pentru backend am folosit C#: pentru logica serverului, manipularea evenimentelor și interacțiunea cu baza de date.

Mai mult, pentru a structura și a defini elemente de interfață am folosit HTML, iar pentru stilizarea paginii web – definirea aspectului și prezentarea elementelor – am folosit CSS. Totodată, pentru furnizarea funcționalităților client-side și pentru gestionarea evenimentelor de interfață se poate observa utilizarea limbajului de programare JavaScript (limbaj OOP, bazat pe conceptul prototipurilor). Ar mai fi de menționat framework-ul (CSS) Bootstrap (framework

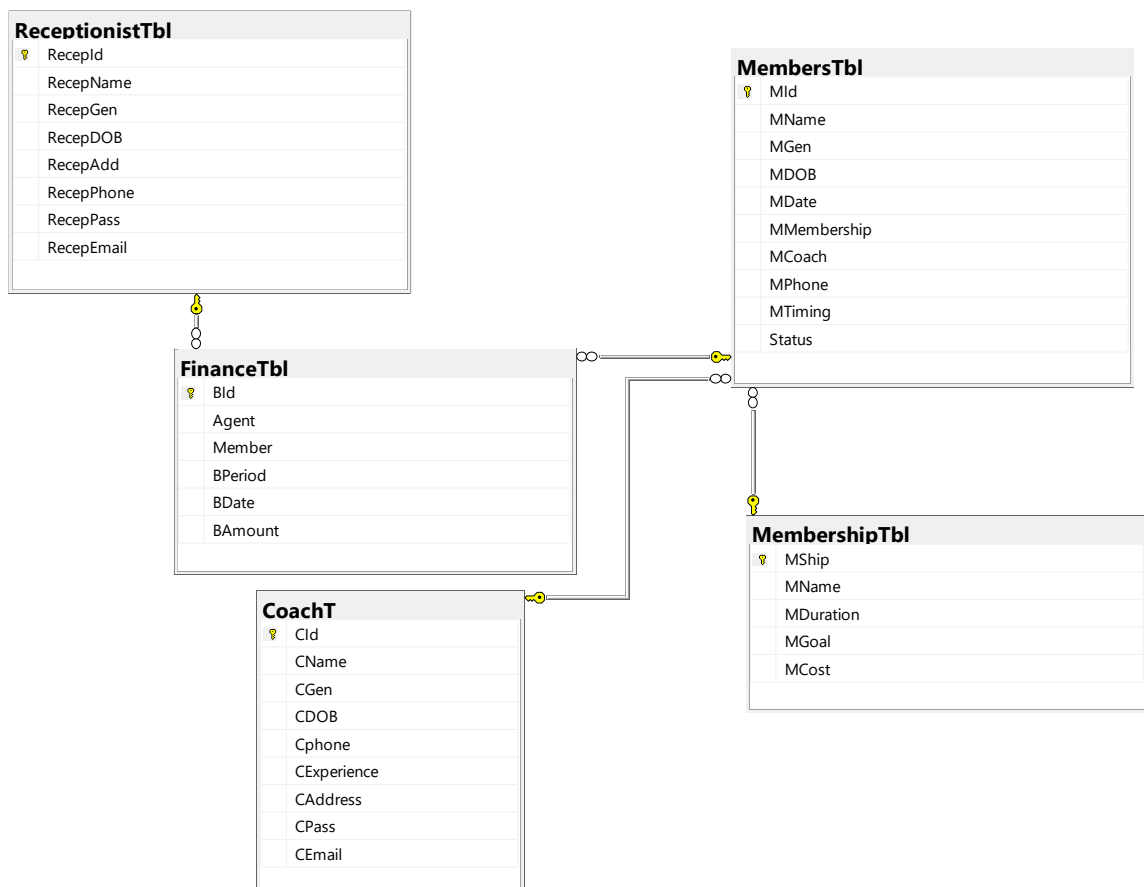
orientat către dezvoltarea web front-end receptivă și mobilă) care este folosit prin includerea unor fișiere (spre exemplu: „bootstrap.min.css”), pentru a adăuga un aspect modern și interactiv paginilor web.

4.2.Baza de date

Baza de date a fost creată în Microsoft SQL Server (sistem de gestionare a bazei de date relaționale produs de compania americană Microsoft Corp). Limbajul de interogare folosit este SQL, iar extensia procedurală este T-SQL (Transact-SQL) – include adiiții specifice furnizate de Microsoft pentru gestionarea tranzacțiilor și pentru a face interogări mai complexe. Spre exemplu, atributul IDENTITY este specific T-SQL, indicând popularea automată cu valori unice crescătoare a coloanei CId.

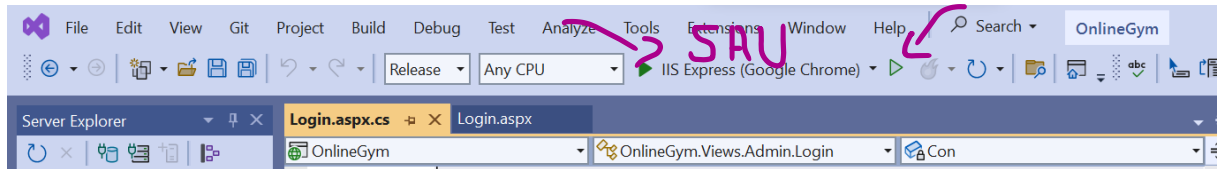
```
1 CREATE TABLE [dbo].[CoachT] (  
2     [CId] INT IDENTITY (1, 1) NOT NULL,  
3     [CName] VARCHAR (50) NOT NULL,  
4     [CGen] VARCHAR (10) NOT NULL
```

Diagrama bazei de date este următoarea:

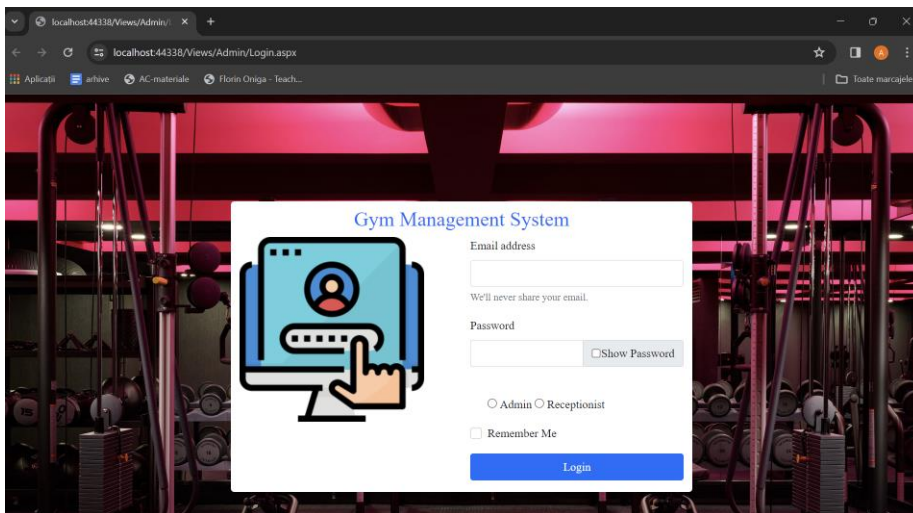


5. Ghid de utilizare

Pentru scrierea codului întregii aplicații am folosit IDE-ul Visual Studio, deci este recomandat ca testarea să fie făcută pe un astfel de IDE, preferabil mai nou de 2017. Prima pagină a aplicației este cea de login, deci pentru testarea tuturor funcționalităților, e de preferat rularea începând cu aceasta.



Tot ceea ce mai este necesar, ar fi existența unui browser și a unei conexiuni stabile la internet.



6. Concluzii, bibliografie și dezvoltări ulterioare

Având în vedere faptul că a fost prima dată când am dezvoltat o aplicație web și că nu am studiat deloc astfel de tehnologii la facultate, acest proiect a fost o provocare, dar și o oportunitate de a explora lucruri noi. De aceea, am întâmpinat dificultăți în alegerea unor limbaje de programare și framework-uri potrivite pentru proiectul meu. Totuși, ASP.NET a fost de un real folos, datorită paginilor interactive și resurselor diverse disponibile online.

Dintre multiplele resurse folosite în realizarea proiectului, menționez câteva:

- <https://help.syncfusion.com/aspnet/overview>
- <https://visualstudiomagazine.com/articles/2022/05/16/vs2022-web-forms-tip.aspx>

- <https://www.c-sharpcorner.com/UploadFile/2b481f/using-web-api-with-Asp-Net-web-forms/>

Ca dezvoltări posibile aş menţiona, în primul rând, finalizarea securităţii la logare, proces început de mine prin generarea de token-uri unice. Mai mult, am creat pagina de plăţi (la care are acces doar adminul), pentru ca în viitor să existe extragerea automată din contul firmei direct în acest formular – pentru vizualizarea mai optimă a plătitorilor de abonament. De asemenea, s-ar putea extinde aplicaţia şi la nivelul clienţilor care frecventează sala de fitness – crearea unui nou rol, care să îşi poată vedea abonamentele active, expirate şi, eventual, să poată face plăţi online.