

UNIVERSITATEA DIN BUCUREŞTI



**FACULTATEA DE
MATEMATICĂ ȘI INFORMATICĂ**



SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENȚĂ
Platforma Hygieia - Îngrijire personalizată
și consiliere pentru sănătatea mintală

COORDONATOR ȘTIINȚIFIC

Lect. dr. Sorina-Nicoleta Preduț

ABSOLVENT

Talida-Marina Dobre

BUCUREŞTI

2023

Rezumat

În prezent, sănătatea mentală a devenit o problemă recunoscută la nivel global, iar metodele de prevenire și asistență pentru persoanele care se confruntă cu aceste provocări devin tot mai diverse și adaptate nevoilor individuale. Industria medicală s-a dezvoltat într-un format mai digitalizat și electronic în ultimele decenii, ceea ce înseamnă că pacienților li se prezintă diferite instrumente de asistență medicală la distanță prin intermediul platformelor web și aplicațiilor mobile. Aplicațiile de sănătate mentală sunt create pentru a ne ajuta să ne îmbunătățim starea emoțională, oferindu-ne diverse metode de a face acest lucru. Unele aplicații furnizează resurse informative, în timp ce altele oferă posibilitatea de a căuta ajutor specializat sau de a se implica în metode interactive de autodescoperire și abordare a anumitor traume. De asemenea, acestea pot furniza și modalități de a împărtăși emoții copleșitoare cu comunități de oameni care pot înțelege și empatiza cu noi.

Abstract

Currently, mental health has become a globally recognized issue, and the methods of prevention and assistance for individuals facing these challenges are becoming increasingly diverse and tailored to different needs. The medical industry has evolved into a more digitized and electronic format in recent decades, which means that patients are being introduced to various remote healthcare tools through different web platforms and mobile applications. Mental health apps are designed to help improve our emotional well-being by providing us with different methods to do so. Some apps provide informational resources, while others offer the ability to seek specialized help or engage in interactive methods of self-discovery and coping with certain traumas. They also provide a platform for sharing overwhelming emotions with communities of people who can understand and empathize with us.

CUPRINS

1. INTRODUCERE.....	5
1.1. Motivație.....	5
1.2 Soluții Existente.....	6
1.2.1. talkspace.....	6
1.2.2. I'm fine.....	6
1.2.3. Mental Health For Romania.....	7
1.3. Soluție Propusă.....	7
1.4 Structura lucrării.....	8
2. DESCRIERE TEHNOLOGII UTILIZATE.....	9
2.1. ASP.NET.....	9
2.1.1 .NET.....	9
2.1.2 ASP .NET MVC.....	10
2.2 Entity Framework.....	12
2.3. Razor Views.....	13
2.4. SQL Server.....	15
2.5 Stilizare.....	16
2.5.1. CSS.....	16
2.5.2. BOOTSTRAP.....	17
2.6. Javascript și jQuery.....	17
2.7 WebRTC.....	19
2.8 SignalR.....	20
2.9 Integrări.....	20
2.9.1 Google Maps API.....	21
2.9.2 API Registrul Medicilor din România [10].....	21
2.9.3 PayPal.....	23
3. DETALII IMPLEMENTARE.....	24
3.1 Structura bazei de date.....	24
3.1.1 IdentityUser - Address - Specialist/Pacient.....	28
3.1.2 Specialist - Pacient - Appointment - PaymentDetails.....	28
3.2 Structura aplicației.....	29
3.2.1 Layered Architecture.....	29
3.2.2 Persistence Layer.....	31
3.2.3 Domain Layer.....	32
3.2.3.i Domain Entities Layer.....	32
3.2.4 Business Layer.....	33
3.2.5 Presentation layer.....	35
4. EXPERIENȚA UTILIZATORULUI.....	37
4.1 Perspectiva vizitator.....	37
4.2 Perspectiva pacient.....	42

4.3 Perspectiva specialistului.....	50
4.4 Perspectiva administratorului.....	55
Concluzii și Direcții Viitoare de Cercetare.....	63
1. Concluzii.....	63
2. Direcții Viitoare de Cercetare.....	63
Bibliografie.....	65

1. INTRODUCERE

Deoarece totul ia naștere cu un motiv, în introducere o să prezint rațiunea din spatele alegerii acestei teme de proiect, precum și de ce au devenit atât de utile aplicațiile de sănătate mintală. De asemenea, o să prezint în acest capitol câteva soluții deja existente pe piață, alături de propria soluție propusă.

1.1. Motivație

Fiind tabu de zeci de ani, dacă nu de secole, sănătatea mintală este acum discutată pe scară largă. În aceste vremuri exceptionale, pandemii, război și alte situații naturale, mulți dintre noi ne confruntăm cu probleme psihologice legate de izolare, anxietate, singurătate, frustrări, dependențe, nesiguranță, depresie, frici și îngrijorare – probleme abordate în mod regulat. Conștientizarea că singurătatea cuiva, de exemplu, este comună întregii omeniri, este esențială pentru a o înțelege, a fi liber de ea și a învăța ce înseamnă să fii în siguranță și bine într-o lume incertă. Deoarece multora le este încă teamă să pășească în cabinetul unui medic specializat sau cum a fost în cazul epidemiei de coronavirus cu care s-a confruntat întreaga planetă, starea de alertă nu ne-a permis să ieșim din case, aplicațiile de mental health au devenit o modalitate sigură și confortabilă de a discuta cu cineva de specialitate din propriul cămin.

Datele ne spun că pandemia cu care ne-am confruntat recent a exacerbat afecțiunile de sănătate mintală, agravându-le în mod semnificativ și a declanșat declinuri ale bunăstării, cu o creștere dramatică a prevalenței problemelor precum depresia, anxietatea, simptomele de stres post-traumatic și stresul. Aproximativ patru din zece adulți din Statele Unite, de exemplu, au raportat simptome de anxietate și depresie în perioada iunie 2020 - martie 2021, comparativ cu o pondere mult mai mică a celor care au raportat aceste simptome în perioada ianuarie - iunie 2019. Unii profesioniști medicali și cercetători sugerează că pandemia i-a ajutat pe oameni să vorbească mai deschis despre sănătatea lor mentală și a făcut mai acceptabil accesul la tratament - acesta evidențiază prevalența problemelor pe care aplicațiile de sănătate mintală le abordează. Aplicațiile pot ajuta nu numai să abordeze volumul de nevoie pentru

sprijinul sănătății mentale, ci și să facă acel sprijin mai accesibil. Cercetările arată că aplicațiile de sănătate mentală au avantaje clinice clare pentru utilizatorii lor. Meta-analizele studiilor care acoperă mai mult de 20 de aplicații mobile au descoperit că utilizarea acestora pentru a ameliora simptomele și a gestiona singuri depresia reduce semnificativ simptomele depresive. O analiză similară a aplicațiilor de tratament a anxietății a descoperit că utilizatorii au experimentat o reducere a simptomelor de anxietate după utilizare, cea mai mare reducere survenind atunci când aplicațiile au fost combinate cu terapii de față în față sau pe internet [1].

1.2 Soluții Existente

La momentul actual există peste 10000 de aplicații de mental health și wellness, numărul crescând continuu [2]. Fiecare aplicație folosește diferite metode prin care să ofere ajutorul de care are nevoie un utilizator.

1.2.1. talkspace

Un exemplu de platformă de mental health care pune la dispoziție servicii online alături de un specialist este talkspace. Aceasta oferă în doar trei pași un terapeut potrivit pentru fiecare pacient din SUA. Aplicația oferă resurse video, telefonice și schimbul de mesaje indiferent de oră.

Platforma are o interfață grafică prietenoasă, quizz-ul pentru personalizarea serviciului este rapid și eficient, la final oferind mai multe rezultate, lăsând astfel decizia finală în mâinile utilizatorului. Pentru a vedea ofertele primite un user este nevoit să achiziționeze pachetul, fiind oferite mai multe planuri de subsecție. Un utilizator obișnuit poate accesa din meniu pagina de business unde se regăsesc articole, rapoarte, eBooks și studii de caz cu diferite topicuri.

1.2.2. I'm fine

Un alt exemplu este I'm fine, o platformă românească pentru psihoterapeuți, care se conectează cu aplicația mobilă pentru cei care au nevoie de suport [3]. Platforma dispune de un număr mare de filtre și o hartă interactivă, care contribuie la optimizarea experienței utilizatorilor în găsirea unui psiholog potrivit. Cu o interfață la fel de user friendly precum talkspace, aceasta dispune de o pagină cu articole despre sănătatea

mintală, în funcție de categorie, are o secțiune specială pentru psihoterapeuți, dar și pentru companii, oferind ateliere sau pachete de ședință de terapie. În cadrul platformei poți programa o ședință cu un psihoterapeut, restul funcționalităților, precum păstrarea unui jurnal, teste despre emoții și notificări din partea specialistului sunt accesibile doar în aplicația de mobil.

1.2.3. Mental Health For Romania

Asemănătoare cu I'm fine, Mental Health For România oferă o modalitate simplă de a găsi un specialist, fie că este vorba de un psihoterapeut, logoped, consilier sau psihiatru, existând diferite filtre și o hartă interactivă. Spre deosebire de aplicațiile anterioare nu poți programa un consult în aplicație, dar sunt puse la dispoziție diferite metode de contact. Spre deosebire de platformele anterioare, Mental Health For România dispune de un serviciu de hotlines, non-stop sau cu un program fixat. Apelurile sunt gratuite și oferă consiliere în situații de criză suicidară, suport emoțional pentru adolescenți și nu numai. Platforma mai dispune de asemenea de o pagină cu podcasturi, un blog conținând diferite topicuri de discuție.

1.3. Soluție Propusă

Cu soluția propusă de mine, am intenționat să preiau o parte din funcționalitățile existente în platformele prezентate mai sus și nu numai, dar am dorit să adaug propria viziune despre cum cred eu că ar trebui să fie configurată o aplicație de mental health. Foarte multe soluții se rezumă doar la căutarea unui psiholog/psihoterapeut și integrarea unei secțiuni de articole, ceea ce nu mi se pare întotdeauna suficient.

Astfel, pe lângă posibilitatea căutării unui specialist și programării unei consultații, am încercat să implementez un mediu de videoconferință integrat în aplicație, eliminând dependența de alte medii și oferind tuturor utilizatorilor același spațiu de exprimare. În acest mod, persoanele care nu au acces la un medic specialist pot primi ajutorul de care au nevoie indiferent de mediul din care provin.

Un alt serviciu introdus în aplicație este cel al unui forum în care utilizatorii pot împărtăși propriile experiențe într-un mediu perfect sigur, în care poveștile lor vor fi auzite. Prin intermediul forumului, ne propunem să normalizăm discuțiile despre sănătatea mentală și să încurajăm o deschidere sinceră în ceea ce privește sentimentele

interioare. În momentul în care împărtășim cu restul lumii ceea ce ne provoacă suferință, un pas spre vindecare este deja făcut.

În plus, am introdus un serviciu de raportare a consultanțelor, care permite crearea unui sumar al ședinței după fiecare sesiune online sau fizică. Această facilitate contribuie la îmbunătățirea comunicării dintre utilizatori și specialiști, permitându-le să păstreze înregistrarea momentelor importante din interacțiunea lor într-un singur loc, accesibil ambelor părți.

De asemenea, utilizatorul beneficiază de posibilitatea de a crea un jurnal personal în cadrul aplicației, oferindu-i o modalitate de a-și înregistra gândurile și trăirile. Acest jurnal poate fi păstrat în mod privat, oferindu-i utilizatorului intimitatea și confidențialitatea dorite. În plus, utilizatorul poate opta să facă publice anumite pagini din jurnal medicului său asociat, facilitând comunicarea și colaborarea în cadrul procesului de asistență medicală.

1.4 Structura lucrării

Această lucrare este structurată în trei capitole principale, concluzii, bibliografie. În Capitolul 2, tema abordată se referă la tehnologiile și integrările utilizate în dezvoltarea soluției. Următorul capitol va prezenta în detaliu structura proiectului, baza de date și arhitectura utilizate pentru a crea o aplicație funcțională, având ca scop satisfacerea utilizatorilor. În Capitolul 4, vom efectua un tur detaliat al modului în care utilizatorii interacționează cu interfața aplicației, iar capturile de ecran din soluție vor oferi o înțelegere mai amplă a explicațiilor. Capitolul de concluzii va furniza un rezumat al aspectelor prezentate anterior, precum și posibile direcții viitoare de cercetare și dezvoltare care pot fi aduse soluției. În cadrul capitolului bibliografic vor fi enumerate referințele utilizate în elaborarea lucrării.

2. DESCRIERE TEHNOLOGII UTILIZATE

În acest capitol se va discuta despre ce tehnologii au fost folosite în procesul de implementare a soluției mele, motivul alegerii și rolul lor pe piața actuală.

2.1. ASP.NET

2.1.1 .NET

.NET este o infrastructură open-source destinată creării aplicațiilor desktop, web și mobile care asigură compatibilitate cu toate sistemele de operare disponibile. Sistemul .NET include instrumente, librării și limbaje care susțin dezvoltarea de software modern, scalabil și cu performanțe ridicate. O comunitate activă de dezvoltatori menține și susține platforma .NET.

.NET Framework este o versiune autentică a platformei .NET, permitând executarea diferitelor tipuri de aplicații. Microsoft a lansat .NET Framework la începutul anilor '90.

Microsoft a lansat .NET Core la sfârșitul anului 2014 pentru a permite suportul cross-platform pentru dezvoltatorii .NET. Compania a lansat cea mai nouă versiune a .NET Core, .NET 5.0, în noiembrie 2020 și au redenumit-o .NET.

.NET Standard reprezintă o definiție oficială a diverse funcționalități (numite API-uri). Diferitele implementări .NET pot reutiliza același cod și librării. Fiecare implementare utilizează atât API-uri standard .NET, cât și API-uri unice. O comunitate activă de dezvoltatori menține și îmbunătățește software-ul .NET. Fundația .NET este o organizație non-profit independentă înființată pentru a sprijini comunitatea .NET. Aceasta oferă resurse de învățare, proiecte open-source .NET și diverse evenimente pentru dezvoltatorii .NET [4].

ASP.NET este succesorul tehnologiei ASP (Active Server Pages) și a reprezentat o modernizare semnificativă în ceea ce privește flexibilitatea și puterea. Aceasta reprezintă o expansiune a platformei .NET, furnizând un set de unelte și biblioteci suplimentare special concepute pentru a dezvolta aplicații web, inclusiv aplicații și site-uri web.

Acstea sunt câteva lucruri pe care ASP.NET le adaugă la platforma .NET:

- Cadru fundamental pentru a manipula cereri web în limbajele de programare F# sau C#;
- Sintaxa de templating (Razor), folosită în consolidarea de pagini web dinamice prin intermediul limbajului C#;
- Librării pentru facilitarea dezvoltării modelelor web comune, precum ar fi arhitectura MVC (Model View Controller);
- Extensii editoriale care oferă facilitatea punerii în evidență a sintaxei, a completării codului și altor întrebuișări utile dezvoltării de pagini web [5].

2.1.2 ASP .NET MVC

În soluția implementată de mine am adoptat o arhitectură bazată pe Model-View-Controller (MVC). Această abordare îmi permite să separ logica de afișare (View), logica de afaceri (Model) și logica de control (Controller), ceea ce conduce la o dezvoltare modulară și organizată. Am ales acest framework datorită beneficiilor semnificative pe care le aduce în dezvoltarea unei aplicații web, precum gestionarea datelor, separarea responsabilităților și creșterea scalabilității.

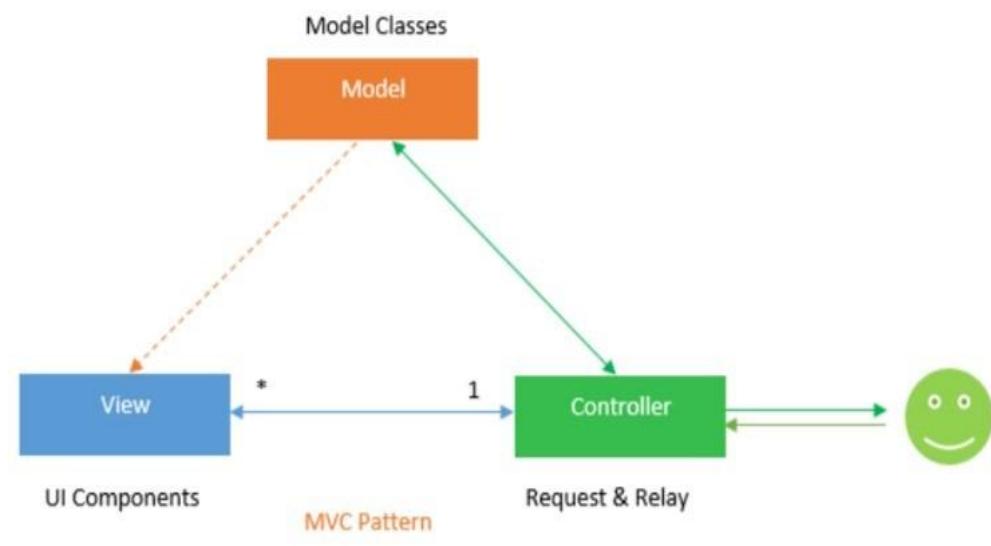


Figura 2.1 MVC Pattern

În proiectul WebApp, am structurat aplicația în jurul controller-elor, view-urilor și fișierelor de bază. Acest lucru asigură o funcționalitate comună pentru toate controller-ele și permite injectarea de dependențe necesare.

View-urile sunt componente responsabile de afișarea interfeței cu utilizatorul și sunt create de obicei pe baza datelor modelului. De exemplu, o vizualizare a produsului „Edit” poate fi creată pentru a afișa câmpuri de text, liste derulante și casete de selectare în funcție de starea curentă a obiectului „Product”.

Controller-urile sunt componente responsabile de manipularea interacțiunii cu utilizatorul, manipularea modelului și alegerea unei vizualizări pentru a afișa interfața cu utilizatorul. Într-o aplicație MVC, vizualizarea este doar despre afișarea informațiilor - este controller-ul care manipulează și răspunde la intrările și interacțiunile utilizatorului.

Această separare a preocupărilor între modele, vizualizări și controlere s-a dovedit a fi o modalitate utilă de structurare a unei aplicații web. Prin separarea clară a vizualizărilor de restul aplicației web, se permite redesenarea aspectului aplicației fără a atinge logica de bază. De exemplu, într-o zi, dacă voi decide să reimplementez vizualizările din aplicația mea, acest lucru va fi mai ușor de realizat prin separarea logică a vizualizării de restul logicii [6].

În cadrul folderului „Infrastructure”, am inclus proiectul „DataAccess” pentru gestionarea contextului aplicației și unității de lucru (Unit Of Work). Acest lucru facilitează interacțiunea cu baza de date și permite manipularea eficientă a entităților.

În folderul „Domain”, am concentrat logica de afaceri a aplicației. Proiectul „BusinessLogic” conține modelele de vizualizare (view models) și serviciile (services), care gestionează procesarea și manipularea datelor, aplicând regulile de afaceri specifice.

Într-o aplicație bazată pe MVC, modelele sunt componente responsabile de menținerea stării aplicației și sunt adesea persistente într-o bază de date.

De asemenea, am împărțit entitățile bazei de date și fișierele care conțin enums în proiectul „Entities”. Aceasta ne permite să definim structura datelor și relațiile în baza de date.

Pentru componente comune, am creat proiectul „Common” care include DTO-uri (Data Transfer Objects), manipularea excepțiilor, interfețe IRepository și IEntity. Acestea facilitează schimbul de date și gestionarea erorilor în întreaga aplicație.

Beneficiile aduse de folosirea arhitecturii MVC pe care le-am considerat când am ales utilizarea acesteia în propria mea soluție au fost:

1. Prin faptul că Modelul (Business logic), Vizualizarea (UI logic) și Controller-ul (Input logic) sunt entități separate, este posibilă dezvoltarea bazată pe teste.

2. Nu are viewstates (ceea ce este pozitiv și negativ).
3. Business logic este stocată într-o singură stare numită model, în timp ce în platforma tradițională ASP.NET, fiecare pagină are propria sa logică de afaceri, ceea ce o face complexă .
4. Am obținut un cod mai organizat și inteligibil, ceea ce facilitează dezvoltarea și evoluția ulterioară a aplicației.

Cu toate acestea, arhitectura MVC poate prezenta și dezavantaje, cum ar fi complexitatea adăugată, mai ales în aplicații mari și complexe. De asemenea, separarea strictă a componentelor poate duce la o creștere a numărului de fișiere și clase, ceea ce poate afecta înțelegerea întregului proiect [7].

2.2 Entity Framework

Cu scopul de a lucra la un grad mai înalt de abstractizare pentru prelucrarea datelor am ales să folosesc Entity Framework, un ORM (Object-relational mapping) open-source pentru aplicații .NET susținut de Microsoft, ce are ca scop principal permiterea lucrului cu baze de date folosind concepte familiare și orientate pe obiecte, în locul interogărilor clasice dintr-o bază de date (precum SQL Server în cazul meu).

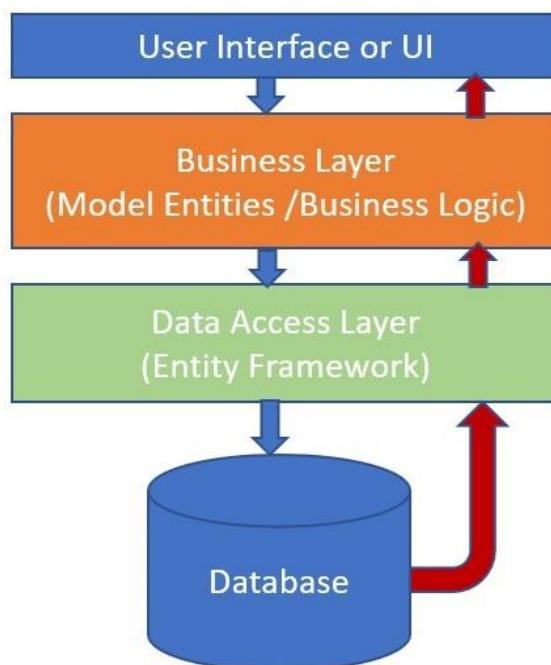


Figura 2.2 Data procession structure [8]

Entity Framework oferă opțiunea de a scrie interogări de tip SQL sub formă de metode LINQ. În spate, acesta va traduce query-ul nostru LINQ în cod pe care să îl înțeleagă baza de date folosită, execută interogarea și utilizează rezultatele pentru a popula instanțele obiectelor noastre. Logica Entity Framework-ului, query-urile și call-urile pentru salvarea modificărilor se află în clasa DBContext (regăsit de obicei în data layer-ul aplicației).

Unul dintre avantajele utilizării Entity Framework este flexibilitatea oferită între schema bazei de date și a claselor definite în domeniu. Pentru a ne asigura că proiectarea claselor din domeniu este cât mai eficientă pentru domeniul nostru iar cea a schemei bazei de date este cât mai avantajoasă pentru baza de date folosită, Entity Framework folosește mapări prin intermediul EDM-ului (Entity Data Model) între clase și structura bazei de date.

Merită menționat, faptul ca Entity Framework este un framework cross platform, acesta are suport pentru gestiunea tranzacțiilor și a salvării datelor (prin apelarea metodei asincrone SaveChanges() cunoscută și ca SaveChangesAsync()), oferă un layer de caching și ne pune la dispoziție un mod de configurare pentru modelul clasei sau entității [9].

2.3. Razor Views.

În patternul MVC, view-ul se ocupă de modul în care datele sunt prezentate în aplicație și interacțiunea cu userul. În aplicația mea, am folosit pentru partea de UI Razer Views, care este o sintaxă de marcare folosită în scrierea de cod HTML pe partea server-ului, în pagini web folosind C#. Razor este avantajos deoarece este compact, expresiv și fluid, ajutând dezvoltatorul în a scrie mai puțin cod oferindu-i de asemenea un workflow al codului mai rapid.

Sintaxa Razor folosește la început simbolul '@'. După '@' putem avea expresii C# sau declarări de variabile, iar dacă este nevoie de mai mult de o linie de cod, acesta se adaugă în blocuri de cod Razor '{}', precum în exemplul de mai jos.

```

@{
    foreach (var article in Model.TopReads)
    {
        var
base64=Convert.ToBase64String(@article.ArticleImage);
        var imgSrc =
String.Format("data:image/gif;base64,{0}", base64);
        <div class="slide">
            <div class="top-section">
                
                <div class="name">
                    <span>@article.ArticleTitle</span>
                    <br />
                </div>
            </div>
            <div class="info-section">
                <span class="date">@article.Author</span>
                <p>@article.Content</p>
                <span class="auteur">Citeste mai departe...
</span>
            </div>
        </div>
    }
}

```

Un alt feature al sintaxei Razor este folosirea directivei @model pentru a pasa într-un view instanțele unui.viewmodel, cum este prezentat în codul următor:

```

@using System.Security.Claims
@model MentalHealthApp.BusinessLogic.Implementation.Forum.
ViewModels.CreateDiscussionVM
@inject MentalHealthApp.Common.DTOs.CurrentUserDto
CurrentUser ...
@foreach(var discutie in @Model.DiscussionVMs)
{ ... }

```

Pentru a transmite anumite date în și din controller în views se folosesc proprietăți precum ViewData sau ViewBag. În soluția mea am folosit ViewBag, care este o încapsulare pentru ViewData. Ambele sunt rezolvate dinamic la rulare, având nevoie de mai multă atenție atunci când lucrăm cu ele datorită tipului de date necunoscut la compile-time, fiind mai predispușe la erori decât dacă am lucra cu un.viewmodel [11].

Mai jos se regăsește un exemplu de utilizare a unui ViewBag din soluția mea:

```

//Implementare din controller
List<ChartModel> data2 = new List<ChartModel>();
var result2 = _chartService.GetAppointmentsInfoChartData();
foreach (var res in result2)
{
    data2.Add(new ChartModel(res.Name, res.Info));
}
ViewBag.AppointmentsInfo JsonConvert.SerializeObject(data2);

//Implementare in JavaScript:
console.log(@Html.Raw(ViewBag.RatingInf));
console.log(@Html.Raw(ViewBag.AppointmentsInfo));
var data = @Html.Raw(ViewBag.AppointmentsInfo);
var dataPoints = @Html.Raw(ViewBag.AppointmentsInfo);

```

2.4. SQL Server

SQL Server reprezinta o modalitate de gestionare a unei baze de date relaționale (RDBMS), fiind dezvoltată și comercializată de Microsoft. Se bazează pe limbajul de programare SQL și utilizează Transact-SQL (T-SQL), implementarea proprie a Microsoft, care propune includerea unui ansamblu de elemente de programare dezvoltate exclusiv pentru acest proiect.

Am ales sa folosesc SQL Server ca mod de salvare a datelor, pentru a beneficia de un sistem robust și scalabil de administrare eficientă a bazelor de date relaționale. Este oferit un set bogat de funcționalități, cum ar fi gestionarea eficientă a interogărilor, securitatea avansată a datelor și suportul pentru gestionarea tranzacțiilor. Alegerea SQL Server mă ajută să dezvolt și să mențin aplicația într-un mod fiabil și performant.

Componentele principale ale SQL Server sunt Motorul de Baze de Date (Database Engine) și Motorul Relațional. Motorul de Baze de Date este format dintr-un motor relațional care procesează interogările și un motor de stocare care se ocupă de gestionarea fișierelor bazei de date, paginile, indecșii, etc. Motorul Relațional se ocupă de procesarea interogărilor, gestionarea memoriei, gestionarea firelor de execuție și a sarcinilor, gestionarea buffer-elor și procesarea interogărilor distribuite [12].

Arhitectura SQL Server:

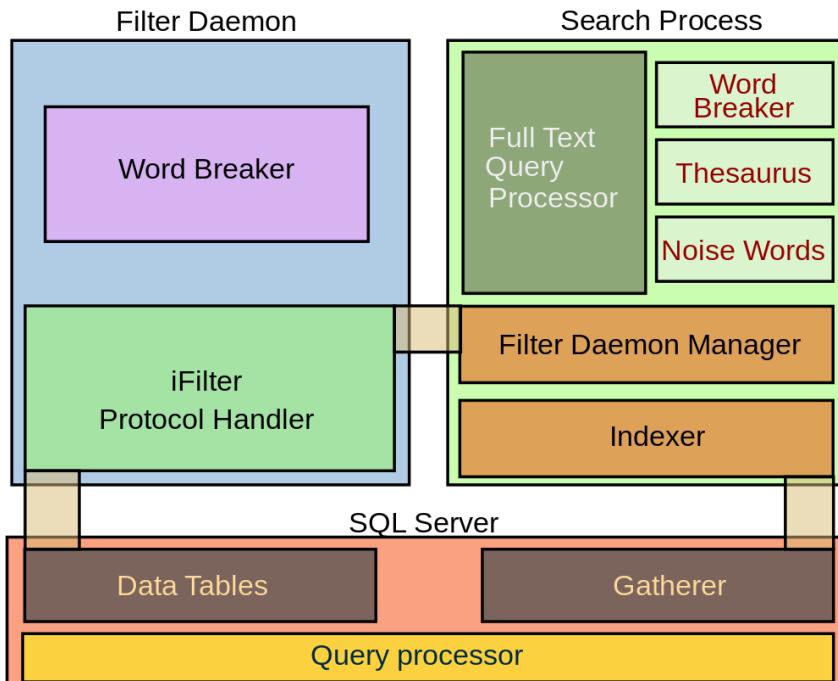


Figura 2.4 The SQL Server Full Text Search service architecture [13]

2.5 Stilizare

2.5.1. CSS

În aplicația mea, am utilizat CSS (Cascading Style Sheets) cu scopul de a crea o prezentare vizuală atractivă a paginilor web. Am folosit CSS pentru a realiza diverse lucruri, cum ar fi animații, poziționare, slide-uri cu imagini, carduri pentru medici, culori etc.

CSS este eficient pentru design, fiind creat cu scopul de a facilita procesul de stilizare a unei aplicații web. Prin intermediul CSS, am putut personaliza aspectul textelor și paragrafelor prin diferite culori, am stabilit spațierea, am ajustat modul de dimensionare și aliniere pe coloană, am adăugat imagini de fundal și am creat un design atrăgător pentru întreg proiectul. CSS permite adaptarea modului de afișare în funcție de dispozitivele și dimensiunile ecranului utilizatorilor, oferind o experiență vizuală optimă. De obicei, CSS este utilizat împreună cu HTML sau XHTML, care sunt limbaje de marcare [14].

Prin utilizarea CSS în aplicația mea, am putut crea interfețe web atractive și coerente, cu un aspect profesional. CSS mi-a permis să personalizez aspectul paginilor și să le adaptez la nevoile și cerințele specifice ale aplicației mele.

2.5.2. BOOTSTRAP

În aplicația mea, am utilizat Bootstrap, un framework front-end gratuit, pentru a dezvolta mai rapid și mai ușor partea vizuală a aplicației. Bootstrap oferă şabloane de design care au la bază HTML și CSS pentru diverse elemente precum tipografia, formularele, butoanele, tabelele, navigația, modalele, caruselele de imagini și multe altele. De asemenea, framework-ul vine și cu plugin-uri optionale de JavaScript.

Un avantaj al folosirii Bootstrap este faptul că are funcționalități responsive, ceea ce înseamnă că design-ul se adaptează automat la diferite dispozitive, precum telefoane mobile, tablete și desktop-uri [15].

În aplicația mea, am utilizat Bootstrap din dorința de a obține o interfață atrăgătoare și intuitivă. Am folosit şabloane predefinite pentru a structura paginile, am implementat stiluri responsive pentru a asigura o experiență plăcută pe diferite dispozitive, am utilizat componente precum butoane, formulare și carusele de imagini oferite de Bootstrap pentru a adăuga funcționalități și interactivitate.

2.6. Javascript și jQuery

Am ales să folosesc în aplicația mea tehnologiile Javascript și jQuery datorită valorilor adăugate prin îmbunătățirea interactivității, eficienței și extensibilității aplicației. Aceste tehnologii au permis crearea unei aplicații mai dinamice, interactive și atractive, cu o experiență îmbunătățită pentru utilizatori.

JavaScript a luat naștere din dorința de a anima paginile web. S-a creat denumirea de „scripturi” pentru programele care sunt scrise folosind acest limbaj. La început, JavaScript avea un alt nume: „LiveScript”. Dar la acea vreme, Java era foarte popular, aşa că s-a decis că poziționarea unui nou limbaj ca „frate mai Tânăr” al lui Java ar ajuta. Cu trecerea timpului, JavaScript a evoluat într-un limbaj de programare independent, având propria specificație cunoscută sub denumirea de ECMAScript. În prezent, nu

mai este asociat în niciun fel cu Java. Există cel puțin trei lucruri remarcabile despre JavaScript:

1. Integrare completă cu HTML/CSS.
2. Lucrurile simple se fac în mod simplu.
3. Este acceptat de toate browser-ele majore și este activat în mod implicit.

JavaScript este singura tehnologie de browser care îmbină aceste trei aspecte. Acesta este motivul pentru care JavaScript este instrumentul cel mai răspândit pentru crearea interfețelor browser-ului. [16].

În aplicația mea, am utilizat JavaScript pentru a implementa funcționalități precum căutarea în bara de căutare, filtrarea doctorilor, implementarea paginilor în jurnalul meu, integrarea cu Google Maps și paginare. JavaScript a jucat un rol crucial în crearea unei experiențe interactive și plăcute pentru utilizatori. Prin intermediul JavaScript, am putut realiza funcționalități complexe și personalizate, adaptându-le nevoilor și cerințelor produsului meu.

De asemenea, am folosit jQuery pentru a adăuga funcționalități interactive și pentru a simplifica manipularea și gestionarea elementelor DOM din HTML.

jQuery este o librărie de JavaScript, care se remarcă prin ușurința sa, avându-l ca și creator pe John Resig în anul 2006, oferind performanță ridicată și un cod concis. Această bibliotecă a fost dezvoltată pentru a ușura procesul de navigare și manipulare a arborelui DOM al documentului HTML, precum și pentru a administra diferite evenimente, animații și cereri Ajax.

Cu ajutorul jQuery, pot găsi elemente specifice în documentul HTML folosind ID-ul, clasa sau atributul corespunzător, iar apoi pot modifica unul sau mai multe atrbute ale aceluiași element, cum ar fi culoarea sau vizibilitatea. De asemenea, jQuery permite interacțiunea utilizatorului cu pagina web prin răspunsul la evenimente, cum ar fi un clic de mouse. jQuery este gratuit și open-source. La data de aprilie 2021, aproximativ 77,8% dintre cele mai accesate 10 milioane de site-uri web foloseau jQuery. [17]

Prin utilizarea jQuery în soluția propusă de mine, am obținut un cod mai concis și mai ușor de întreținut, îmbunătățind în același timp interacțiunea utilizatorului cu paginile web și oferind o experiență mai plăcută și mai dinamică.

2.7 WebRTC

Am utilizat WebRTC pentru a implementa o conferință video între medici și pacienți. WebRTC este o tehnologie open-source care permite comunicarea în timp real între browser și dispozitive prin intermediul apelurilor audio, video și text.

Cu ajutorul WebRTC, transferul de date are loc în timp real, fără a fi necesare interfețe personalizate, plugin-uri suplimentare sau software special pentru integrarea în browser. Pur și simplu, deschizând o pagină web, utilizatorii pot comunica în timp real prin voce și video. WebRTC implementează trei API-uri principale: MediaStream (cunoscut și sub numele de getUserMedia), RTCPeerConnection și RTCDATAChannel. De obicei, WebRTC conectează utilizatorii prin transferul audio, video și de date în timp real între dispozitive, utilizând comunicare P2P. În cazul în care utilizatorii se află în rețele IP diferite, care au firewall-uri NAT (Network Address Translation) ce pot împiedica comunicarea în timp real (RTC), WebRTC poate fi utilizat împreună cu server-ele STUN (Session Traversal Utilities for NAT). Aceasta permite traducerea unei adrese IP într-o adresă publică de internet, pentru a se putea stabili conexiuni între perechi.

WebRTC oferă o serie de avantaje în implementarea funcționalităților de comunicare în timp real în aplicația mea precum:

- Funcționează pe orice sistem de operare, atât timp cât browser-ul suportă WebRTC;
- Nu necesită componente terțe sau plugin-uri suplimentare;
- Este un software gratuit, open-source.

Cu toate acestea, există și unele dezavantaje asociate utilizării WebRTC:

- Fiecare utilizator trebuie să stabilească o conexiune de browser P2P, ceea ce poate duce la probleme de lățime de bandă;
- Nu există standarde definitive de calitate a serviciului, ceea ce înseamnă că calitatea video sau audio pe internet poate fi inconsistentă [18].

Utilizarea WebRTC în proiectul meu mi-a permis să implementez în mod eficient și interactiv funcționalitatea de videoconferință între medici și pacienți. Am folosit API-urile oferite de WebRTC pentru a gestiona conexiunile în timp real și transferul de date între participanți, oferind astfel o experiență de comunicare fluidă și interactivă.

2.8 SignalR

Alături de WebRTC am utilizat SignalR pentru implementarea conferinței video, oferind astfel o funcționalitate de timp real. SignalR este o librărie destinată programatorilor ASP.NET, care are ca scop facilitarea integrării posibilității de comunicare în timp real în aplicații web.

Prin utilizarea SignalR, server-ul poate trimite conținut către clienți conectați în timp real, fără ca acesta să fie nevoie să aștepte solicitările clientilor pentru a furniza date noi. Aceasta înseamnă că pacienții și medicul pot interacționa în timp real prin conferință video, fără întârzieri semnificative.

Implementarea mea în cadrul aplicației a implicat utilizarea unei conexiuni SignalR pentru a gestiona comunicarea între server și client. Am folosit SignalR pentru a trimite și a recepta mesaje între părți. Prin intermediul conexiunii SignalR, am putut crea o legătură bidirectională între medic și pacient pentru a partaja fluxuri video și audio.

Teoria din spatele SignalR constă în faptul că biblioteca furnizează un API care oferă o soluție ușor de utilizat pentru a crea apeluri de procedură remote (RPC) de la server la client, permitându-le aplicațiilor să trimită și să primească informații în timp real. Aceasta înseamnă că server-ul poate apela funcții JavaScript din browser-ul clientului și viceversa, facilitând comunicarea între cele două părți. SignalR extrage detaliile tehnice legate de transportul datelor în timp real, încercând să stabilească în primul rând o conexiune WebSocket pentru cea mai bună performanță și eficiență [19].

Astfel, prin utilizarea SignalR în soluție, am reușit să creez o experiență de conferință video interactivă și în timp real între medic și pacienți, facilitând comunicarea și îmbunătățind calitatea îngrijirii medicale.

2.9 Integrări

În acest capitol, voi prezenta câteva API-uri externe care au fost utilizate în proiectul meu pentru a implementa funcționalități complete, care oferă o experiență detaliată și satisfăcătoare utilizatorilor.

2.9.1 Google Maps API

Pentru a oferi pacienților posibilitatea de a vizualiza locația cabinetului medicului pe o hartă am ales integrarea API-ului Google Maps în soluția mea. Implementarea include caracteristici precum geocodificarea, plasarea marajelor și funcționalitatea de căutare, care îmbunătățesc experiența utilizatorului și furnizează informații valoroase pentru navigarea către locația dorită. Implementarea a constat pentru început prin preluarea din baza de date a locației specialistului. Se vor determina coordonatele de latitudine și longitudine, utilizând serviciul de geocodificare al API-ului Google Maps. Cheia API-ului este furnizată în variabila "apiKey", iar adresa este inclusă în URL-ul cererii. Răspunsul de la API este preluat asincron utilizând funcția "fetch" și parsat ca JSON. După obținerea coordonatelor de latitudine și longitudine, se creează o hartă utilizând API-ul Google Maps. Un maraj este plasat pe hartă pentru a indica locația cabinetului medicului. Utilizatorii pot căuta locuri pe hartă folosind funcționalitatea de căutare, iar rezultatele sunt afișate ca maraje. Harta interacționează automat cu utilizatorul și se ajustează pentru a afișa toate marajele și locațiile relevante.

2.9.2 API Registrul Medicilor din România [10]

Pentru a mă asigura că identitatea medicului psihiatru care se înregistrează este reală și că acesta are dreptul de a practica, am ales să introduc în aplicația mea API-ul de la Registrul Medicilor, care prin căutarea după nume-prenume/prenume-nume a unui medic sunt furnizate date precum domeniul medical în care lucrează, locația, statusul, gradul medical și dreptul de practică.

Așadar, în implementarea mea am început prin a efectua un GET către API-ul respectiv pentru a căuta medicul folosind numele și prenumele furnizate la înregistrare. Dacă răspunsul este null, se va face o nouă cerere GET, dar de data aceasta după prenume și nume. Căutarea se face doar în cazul în care medicul are specialitatea psihiatrie. Ne vom asigura ca API-ul va oferi un răspuns de tip succes, pentru ca apoi să parsăm răspunsul în format JSON și accesa datele de care avem nevoie. Se verifică statusul doctorului din primul rezultat returnat. Dacă este diferit de null, se parcurg toate rezultatele obținute. Informații precum județul și statusul fiecărui doctor sunt extrase și se verifică dacă doctorul este psihiatru. Dacă statusul este „Activ” și județul corespunde cu cel din model sau utilizatorul curent are rol de administrator, doctorul este înregistrat

și utilizatorul este redirecționat către pagina principală. Dacă API-ul nu oferă niciun rezultat, se afișează o vizualizare care indică faptul că doctorul nu există în arhivele medicale. Mai jos este prezentat codul explicitat anterior:

```
[HttpPost]
public async Task RegisterDoctor(RegisterDoctorModel model) {
    HttpResponseMessage response = await
client.GetAsync("https://regmed.cmr.ro/api/v2/public/medic/cauta
re/" + model.LastName + model.FirstName);
    if (response == null) {
        response=await client.GetAsync
("https://regmed.cmr.ro/api/v2/public/medic/cautare/" +
model.FirstName + model.LastName);
    }
    if(model.Specialty == "Psihoterapie" || model.Specialty
=="Psihologie")
    {
        _doctorAccountService.RegisterNewDoctor(model);
        return RedirectToAction("Index", "Home");
    }
    response.EnsureSuccessStatusCode();
    string responseBody = await response.Content.
ReadAsStringAsync();
    var responseJson = (JObject) JsonConvert.
DeserializeObject(responseBody);
    var responseData = responseJson["data"];
    if (responseData["results"][0]["status"].Value() != null)
{
    for (int i = 0; i < responseData["total"].Value();
i++) {
        var judet = responseData["results"][i]["judet"]
.Value();
        var status = responseData["results"][i]["status"]
.Value();
        var isPsihiatru =
(responseData["results"][i]["specialitati"]
.Value() as JArray)
.Select(s=> (string)s.Value().nume)
.Any(s=>s.ToLower() == "psiatrie");
        if((status == "Activ" && judet.NormalizeString()
.Equals(model.County)) || CurrentUser.isAdmin == true) {
            _doctorAccountService.RegisterNewDoctor(model);
            return RedirectToAction("Index", "Home");
        }
    }
}
    return View("Status is not active, he can not work in
```

```
present as a doctor");
}

else { return View("Error, doctor does not exist"); } }
```

2.9.3 PayPal

Deoarece în cadrul aplicației există posibilitatea ca medicii să își țină ședințele cu pacienții în mediul online, am integrat posibilitatea plății cu cardul, pentru a simplifica astfel metoda de a achita costul programării. Înainte de a accesa conferința video, pacientul este redirecționat către pagina de PayPal unde se face plata. Am ales integrarea cu PayPal pentru facilitarea procesului de plată, pentru a oferi utilizatorilor o modalitate sigură și convenabilă de a efectua tranzacțiile. Pentru început în implementare, am creat un obiect "Payment", în care am stocat datele necesare despre tranzacție (valoare, monedă, descriere) urmând să utilizăm API-ul PayPal pentru a crea plata și obține URL-ul de aprobată pentru plată. Utilizăm acest URL pentru a redirecționa utilizatorul către pagina de aprobată a plății PayPal. Efectuăm plata propriu-zisă, iar dacă aceasta a fost aprobată vom salva în baza de date informațiile despre tranzacție.

3. DETALII IMPLEMENTARE

În acest capitol, doresc să prezint în detaliu implementarea soluției mele, începând de la componenta de bază, respectiv structura bazei de date, și progresând până la vârful piramidei, adică interfața utilizatorului.

3.1 Structura bazei de date

În figurile de mai jos vor fi prezentate diagrama ER și diagrama bazei de date generată în SSMS, pe structura căreia a fost construită soluția mea.

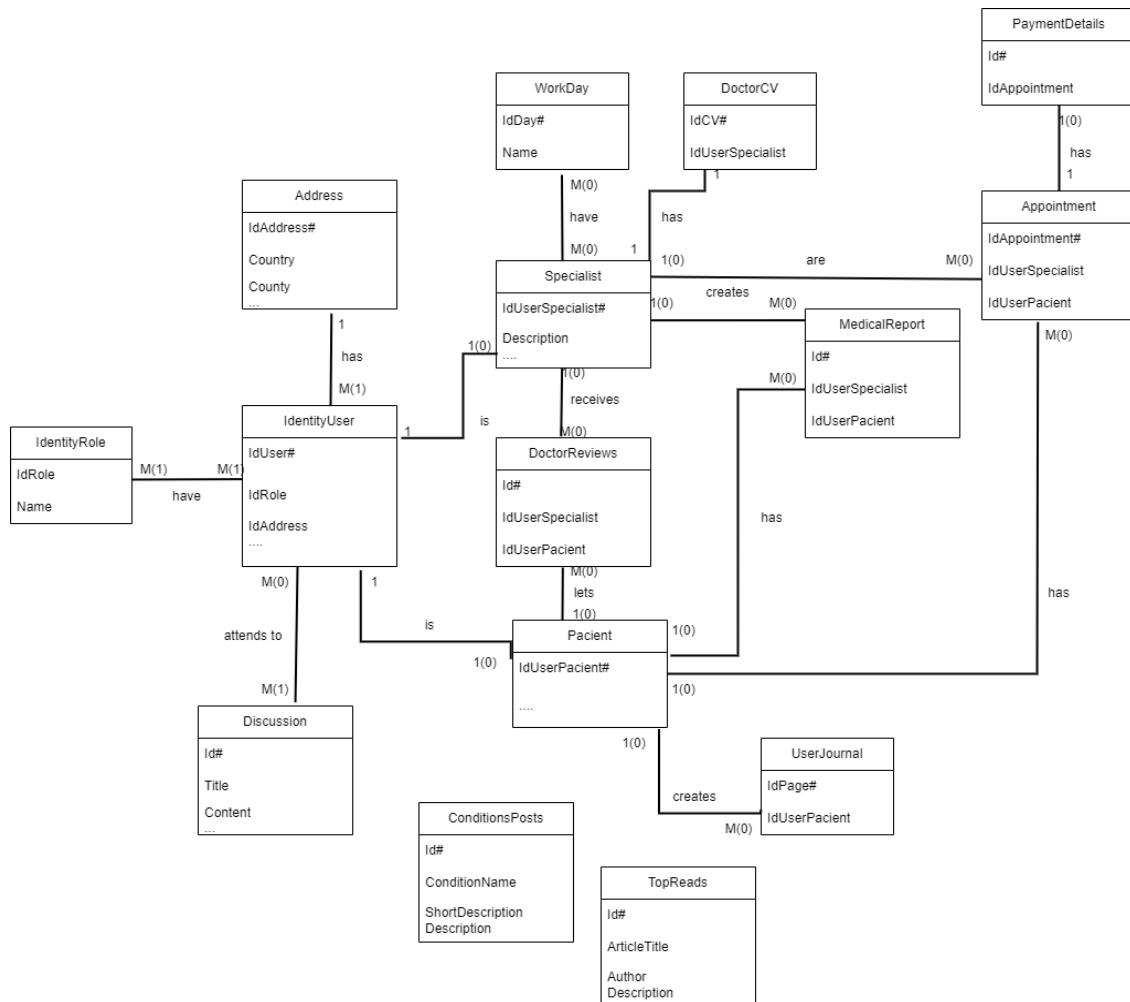


Figura 3.0 Diagrama ER

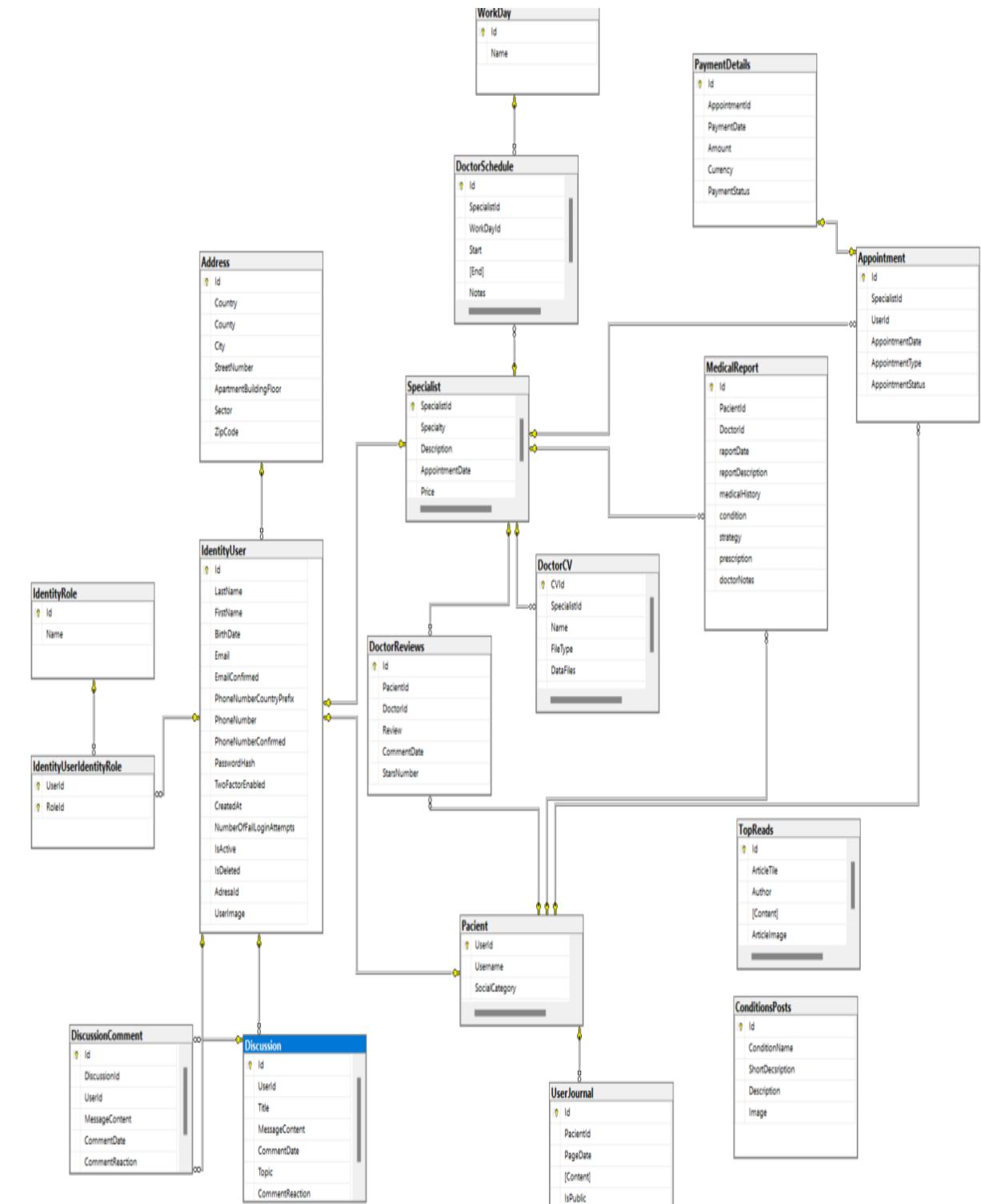


Figura 3.1 Diagrama bazei de date generata de SSMS

Sunt de părere că baza de date reprezintă primul pas și un aspect foarte important în procesul de creare a unei aplicații. O bază de date fragilă poate afecta întreaga structură și partea de business a proiectului, conducând la rezultate diferite față de ceea ce ne-am propus inițial. Astfel, prima etapă în procesul dezvoltării aplicației mele a fost gândirea

și construirea unei baze de date cu ajutorul căreia să pot construi funcționalități care să ofere o experiență pozitivă utilizatorului.

Baza de date a fost proiectată în stilul „database-first”, utilizând SQL Server Management Studio. Astfel, am început prin definirea schemelor, tabelelor și relațiilor între acestea în cadrul mediului de dezvoltare. Apoi, am folosit Entity Framework, o componentă cheie a aplicației mele, pentru a crea modelele și entitățile corespunzătoare bazei de date. Au fost generate automat clasele și mapările necesare pentru a accesa și manipula datele din baza de date. Astfel, tabelele din baza mea de date au fost transformate în clase corespunzătoare în codul sursă al aplicației mele.

Această abordare „database-first” mi-a permis să beneficiez de avantajele oferite de Entity Framework, precum abstracția și gestionarea ușoară a accesului la date, fără a fi nevoie să rescriu manual codul pentru a accesa fiecare tabel sau să creez scheme complexe de interogare a datelor.

În timpul implementării am decis să adaug câteva funcționalități rezultând o expansiune a bazei de date care a fost realizată prin „code-first”, modificările ajungând în baza de date prin migrări.

Mai jos este prezentată o parte din definirea claselor de context și entităților care sunt mapate pe tabelele bazei de date. De asemenea, se stabilește configurația folosită pentru conectarea cu baza de date și se definesc relațiile între entități prin intermediul cheilor străine (foreign keys). Pentru fiecare tabel, am creat o entitate corespunzătoare care moștenește clasa `DbSet<T>`. Proprietățile de tip `DbSet<T>` sunt utilizate pentru a defini tabelele din baza de date. Fiecare `DbSet<T>` reprezintă o entitate din modelul de date al aplicației (de exemplu, `IdentityUserToken`, `Appointment`, `DoctorSchedule`, etc.).

De exemplu, pentru clasa `UserJournal` se definește tabelul „`UserJournal`” și se specifică proprietățile coloanelor, cum ar fi `Id`, `PageDate`, `Content` și `IsPublic`. Se stabilește, de asemenea, relația cu clasa `Pacient` prin intermediul cheii străine `PacientId`.

```

public partial class MentalHealthApplicationContext : DbContext
{
    public MentalHealthApplicationContext() { }
    public MentalHealthApplicationContext(DbContextOptions<MentalHealthApplicationContext> options) : base(options) {}

    public virtual DbSet<Address> Addresses { get; set; } = null!;
    public virtual DbSet<IdentityRole> IdentityRoles{get;set;}= null!;
    public virtual DbSet<IdentityUser> IdentityUsers { get; set; } =
    null!;
    public virtual DbSet<IdentityUserToken> IdentityUserTokens { get; set; } = null!;
    public virtual DbSet<IdentityUserTokenConfirmation>
    IdentityUserTokenConfirmations { get; set; } = null!;
    public virtual DbSet<Specialist> Specialists { get; set; } =
    null;
    public virtual DbSet<Pacient> Pacients { get; set; } = null!;
    public virtual DbSet<UserJournal> UserJournals { get; set; } =
    null!;
    public virtual DbSet<DoctorCV> DoctorCVs { get; set; } = null!;

    ....
    modelBuilder.Entity<UserJournal>(entity =>
    {
        entity.ToTable("UserJournal");
        entity.Property(e => e.Id).ValueGeneratedNever();
        entity.Property(e => e.PageDate)
            .HasColumnType("datetime") .HasColumnName("PageDate");
        entity.Property(e => e.Content)
            .HasColumnName("Content");
        entity.Property(e => e.IsPublic)
            .HasColumnName("IsPublic")
            .HasColumnType("bit");
        entity.HasOne(d => d.Pacient)
            .WithMany(p => p.UserJournals)
            .HasForeignKey(d => d.PacientId)
            .OnDelete(DeleteBehavior.Cascade)
            .HasConstraintName("FK_UserJournal_PacientId");
    }) ; ...
});

```

În continuare o să prezint câteva tabele și relații importante din structura bazei de date.

3.1.1 IdentityUser - Address - Specialist/Pacient

Tabela IdentityUser este folosit pentru a stoca informații esențiale despre un utilizator precum nume, prenume, email, parola hash-uită. Regăsim aici legătura de tipul one-to-one către tabela Address. Aplicația aduce în prim plan două tipuri de utilizatori: pacient și specialist (exceptând adminul).

Un specialist dispune pe lângă datele din tabelul principal IdentityUser de care este legat prin cheie externă și de o relație de tip one-to-one pentru anumite informații specifice rolului său: specialitatea sa (psihologie, psihiatrie sau psihoterapie), descrierea profilului care poate ajuta un utilizator în a decide dacă medicul respectiv este potrivit, „AppointmentDate” reprezentând durata unei ședințe și „Price” prețul unei programări. Asemenea unui specialist, pacientul este legat prin foreign key de IdentityUser în relație one-to-one. Pe lângă atrbutele din user un pacient are un „UserName” folosit în principiu în partea de forum, un „SocialCategory” pentru a diferenția categoriile sociale: student, angajat, pensionar, elev și coloana „HasInsurance” prin care un medic află dacă pacientul său are sau nu asigurare medicală.

3.1.2 Specialist - Pacient - Appointment - PaymentDetails

Scopul aplicației este de a furniza pacienților oportunitatea de a consulta medici prin intermediul programărilor online sau prin programări fizice la cabinetul medicului. Astfel în baza de date am introdus tabela Appointment, legată prin foreign key atât de Specialist cât și de Pacient, ambele relații fiind one-to-many (atât un pacient cât și un specialist pot avea înregistrate în aplicație mai multe programări). O programare salvată în baza de date va avea următoarele informații, pe lângă id-ul său propriu și cele ale utilizatorilor, „AppointmentDate” care oferă informații despre data și ora la care are loc programarea, „AppointmentType” reprezintă tipul programării, online sau fizice, și „AppointmentStatus” este un atribut care va semnala statusul unei ședințe:

1. În Așteptare - statusul incipient până în momentul în care medicul va decide dacă o acceptă sau nu;
2. Programare Acceptată - programarea este acceptată de către medic și va avea loc la data stabilită;
3. Programare Respinsă - programarea nu este acceptată de către medic din diverse motive;

4. Programare Realizată - programarea a avut loc la data stabilită;
5. Programare Anulată - programarea a fost anulată de către medic sau nu a avut loc în decursul zilei la care a fost programată.

Înainte de a accesa programarea online se va face plata cu cardul în aplicație printr-un third party - PayPal, iar informațiile legate de tranzacție vor fi salvate în baza de date, Tabela PaymentDetails este legată de Appointments (deoarece se salvează informații legate de plata programării respective) prin foreign key - id-ul programării, în relație de tip one-to-one (în cazul în care tranzacția eșuează și se va reface plata va fi apelat un update în baza de date cu noile informații). În tabela respectivă se vor salva date legate de data plății în „PaymentDate”, suma plătită în „Amount”, moneda în care s-a efectuat plata - „Currency” și statusul tranzacției „PaymentStatus”.

3.1.2 IdentityUser - Discussion - DiscussionComment

O altă funcționalitate importantă a aplicației este cea de forum, în care utilizatorii pot avea discuții legate de propriile experiențe. Discuțiile vor fi salvate astfel: topicul de discuție creat inițial va fi salvat în tabela Discussion legată de tabela IdentityUser prin cheie externă și va conține pe lângă id-ul discuției și al creatorului, „Title” - titlul discuției, „MessageContent” conținutul mesajului, „CommentDate” - data la care a fost creată discuția și „Topic” - tema centrală a discuției (un anumit simptom sau diagnostic).

Comentariile aduse unei discuții sunt salvate în DiscussionComment, care este legată de IdentityUser (persoana care scrie un comentariu la discuție) prin relație de tipul one-to-many (un user are posibilitatea de a avea mai mult de un comentariu) și de tabela Discussion tot prin relația one-to-many (o discuție poate fi alcătuită din minim un comentariu). Pe lângă aceste informații mai există „MessageContent” - conținutul comentariului și „CommentDate” - data exactă la care a fost postat comentariul.

3.2 Structura aplicației

3.2.1 Layered Architecture

Din punct de vedere tehnic, soluția a fost dezvoltată și gândită folosind „Layered Architecture” sau arhitectură pe niveluri. Fiecare „layer” fiind responsabil de transmiterea informației doar către nivelul inferior.

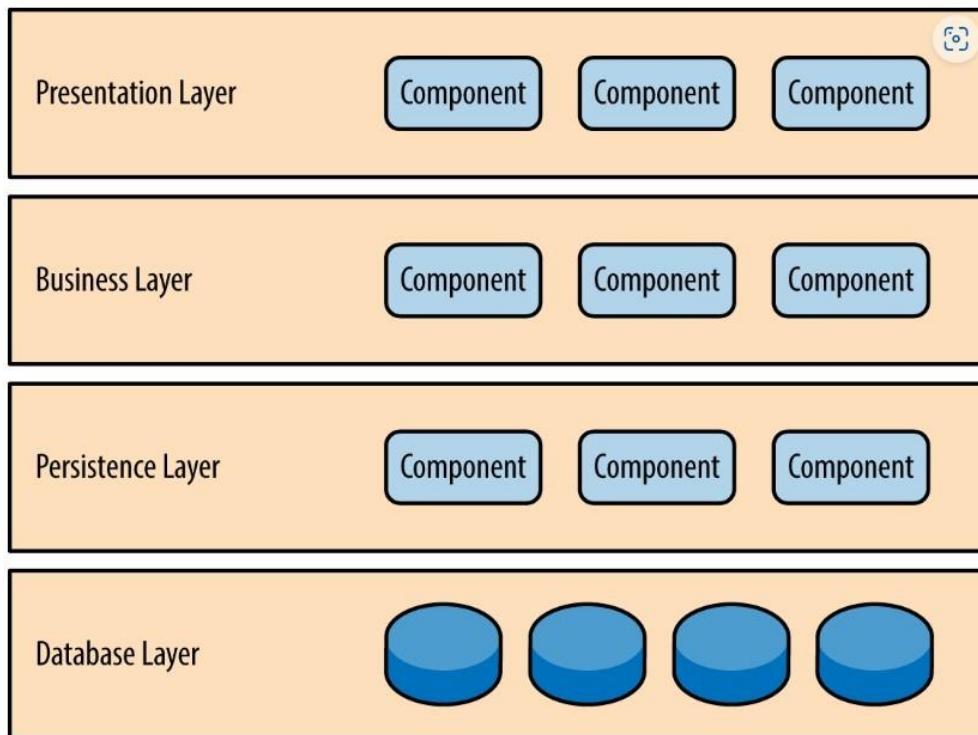


Figura 3.2.1 Layered architecture pattern

Această abordare este considerată standardul pentru majoritatea aplicațiilor Java EE și este larg cunoscută de către arhitecți, designeri și dezvoltatori. Arhitectura stratificată se potrivește în mod apropiat cu structurile tradiționale de comunicare și organizare IT întâlnite în majoritatea companiilor, fiind astfel o alegere naturală pentru majoritatea dezvoltatorilor de aplicații.

Unul dintre aspectele puternice ale arhitecturii stratificate constă în separarea responsabilităților între componente sale. Componentele dintr-un anumit strat se ocupă doar de logica specifică aceluia strat. De exemplu, componentele din stratul de prezentare se ocupă exclusiv de logica de prezentare, în timp ce componentele din stratul de afaceri se ocupă exclusiv de logica de afaceri. Această clasificare a componentelor facilitează definirea unor modele eficiente de roluri și responsabilități în cadrul arhitecturii, precum și dezvoltarea, testarea, gestionarea și întreținerea aplicațiilor utilizând această arhitectură, datorită interfețelor bine definite ale componentelor și a domeniului limitat al fiecărei componente. Pentru a ilustra modul în care funcționează arhitectura stratificată, luăm în considerare o cerere din partea unui utilizator de afaceri de a obține informații despre un client specific, aşa cum se poate vedea în imaginea prezentată în continuare. Săgețile negre arată cum cererea este

transmisă către bază pentru a primi informații despre client, iar săgețile roșii arată cum răspunsul este transmis înapoi la ecran pentru a reprezenta vizual datele.

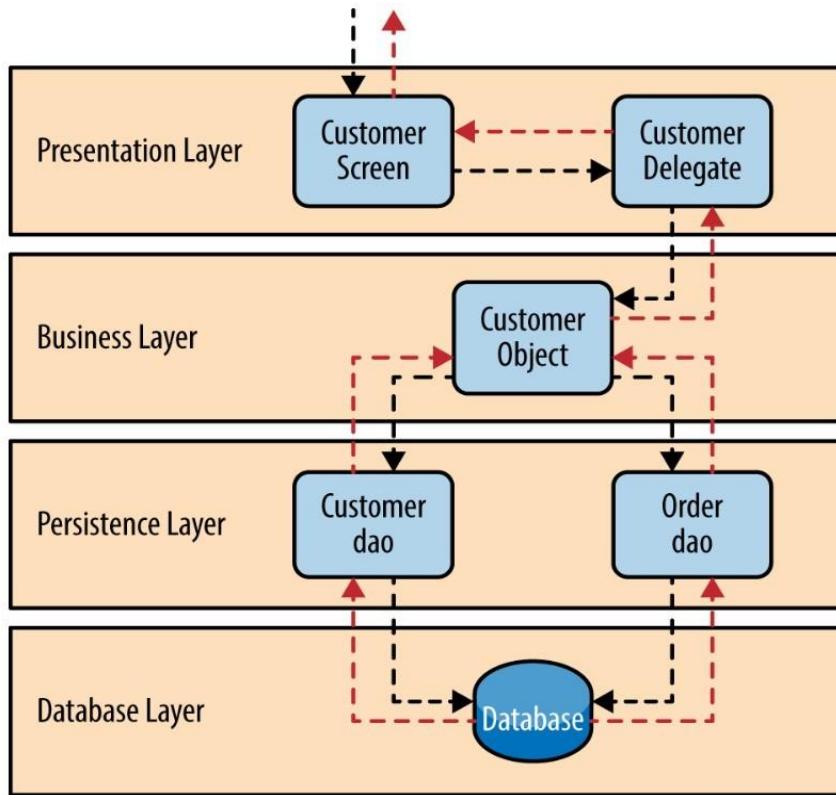


Figura 3.2.2 Layered architecture example

Arhitectura stratificată funcționează astfel: cererea pentru informații despre un client este transmisă prin straturi succesive, până la baza de date, iar răspunsul este transmis în sens invers către ecranul de afișare. Componentele din fiecare strat își îndeplinesc doar logica specifică, asigurând separarea responsabilităților și facilitând dezvoltarea, testarea și întreținerea aplicațiilor [20].

3.2.2 Persistence Layer

Layerul de persistență are rolul de a înconjura logica necesară pentru a ajuta comunicarea cu baza de date și efectuarea operațiilor standard de interogare și manipulare a datelor. Adesea, acest layer va simplifica aceste cerințe prin utilizarea unui ORM. Aceasta standardizează modul în care interacționează diferitele componente, eliminând redundanță din cod și accelerând procesul dezvoltării. ORM-ul poate fi comparat cu crearea unei "baze de date virtuale" care este reflectată în baze de date care sunt tradiționale, precum de exemplu SQL Server. [21], [22].

În proiectul meu Persistence Layer se află în interiorul proiectului de DataAccess, iar pe lângă ORM sunt integrate Repository-urile (în fișierul de BaseRepository) și Pattern-ul Unit Of Work.

Repository-ul este definit ca fiind o clasă echivalentă unei entități, în care se regăsesc toate posibilele operații asupra bazei de date, abstractizând astfel toate operațiile de CRUD. Pattern-ul Repository poate fi implementat în două moduri, fie utilizând pentru fiecare entitate un Repository separat, fie utilizând un singur Repository pentru toate entitățile [23]. În soluția mea am ales să creez un repository general pentru toate entitățile, astfel evitând scrierea de cod repetitiv ceea ce ar fi dus la încălcarea principiului DRY - „Don't repeat yourself”.

Unit of Work, un alt pattern inclus în dezvoltarea soluției mele, este utilizat pentru a grupa una sau mai multe operații (de obicei operații CRUD în baza de date) într-o singură tranzacție și pentru a le executa aplicând principiul „fă totul sau nu face nimic”. Astfel, se vor executa toate operațiile în baza de date ca o unitate [23].

În proiectul meu, am creat clasa „UnitOfWork” cu rolul de a implementa acest şablon. Aceasta conține instanțe ale claselor " IRepository" pentru diferite entități, cum ar fi „IdentityUser”, „Specialist”, „Pacient”, etc. Aceste instanțe sunt inițializate folosind clasa „BaseRepository” și contextul bazei de date „MentalHealthAppContext”.

De asemenea, clasa „UnitOfWork” conține o metodă „SaveChanges” care este responsabilă pentru salvarea modificărilor în baza de date prin apelul metodei „SaveChanges” a contextului. Prin utilizarea clasei „UnitOfWork”, este posibilă consolidarea mai multor operații într-o tranzacție unică, asigurând consistența datelor în baza de date.

3.2.3 Domain Layer

Domain Layer reprezintă nucleul întregii aplicații. Este stratul în care sunt incluse toate regulile de afaceri legate de problema ce urmează a fi rezolvată. În acest strat vor fi prezente entități, obiecte de valoare, aggregate, fabrici și interfețe. Acest strat ar trebui să fie cât mai independent de dependențe posibil.

3.2.3.i Domain Entities Layer

Domain Entities Layer este locul în care se află structurile de date și modelele care reflectă concepțele fundamentale ale domeniului aplicației.

Entitățile sunt extrem de importante în modelul de domeniu, deoarece reprezintă baza modelului. Acestea sunt definite în principal prin identitatea lor, continuitatea și persistența în timp, și nu doar prin atributele care le compun. Prin urmare, este important să le identificăm și să le proiectăm cu atenție. Domain entities layer ar trebui să nu prezinte legături cu baza de date, o astfel de dependență putând duce la complicații ale muncii în viitor [24].

3.2.4 Business Layer

Business Layer sau Application Layer este stratul în care sunt gestionate fluxurile de procese de afaceri. Capacitățile aplicației pot fi observate în acest strat. Entitățile din domeniu sunt create și supuse actualizării aici. În funcție de scenariile de utilizare, aspecte precum gestionarea tranzacțiilor sunt, de asemenea, rezolvate în acest strat. Acest strat definește în mod obișnuit serviciile aplicației care implementează scenariile de utilizare ale sistemului [25]. Aceste servicii coordonează fluxul de date folosind entitățile și tipurile din domeniu [26].

În soluția mea, în business layer se regăsește folder-ul „Base” în care se află „BaseService” care reprezintă serviciul de bază pe care îl implementează restul serviciilor, oferind un proces tranzacțional al execuției metodelor din aplicație.

O tranzacție reprezintă o singură unitate logică de lucru care accesează și, posibil, modifică conținutul unei baze de date. Tranzacțiile accesează date utilizând operațiuni de citire și scriere. Pentru a menține coerența într-o bază de date, înainte și după o tranzacție, se respectă anumite proprietăți. Acestea sunt denumite proprietăți ACID:

1. **Atomicitate** - toate operațiile din unit of work au loc odată sau nu se întâmplă deloc;
2. **Consistență** - Baza de date este consistentă înainte și după tranzacție;
3. **Izolare** - Multiple tranzacții au loc independent fără intervenție;
4. **Durabilitate** - Schimbările unei tranzacții efectuate cu succes se întâmplă chiar dacă sistemul întâmpină erori [27].

În cadrul mediului de business al proiectului meu, se regăsește folderul „Implementation”, care conține toate serviciile împreună cu logica funcționalităților și modelele soluției. Voi prezenta, cu ajutorul unui exemplu din cod, cum funcționează serviciile. Serviciul prezentat va fi cel care conține funcționalitățile

jurnalului creat de un utilizator iar exemplificarea va fi făcută pe metoda de modificare a confidențialității unei pagini de jurnal.

```
public class UserJournalService : BaseService
{
    public UserJournalService(ServiceDependencies dependencies) :
        base(dependencies) { }
    public bool ChangeJournalPublicField(bool isPublic,
        Guid id)
    {
        return ExecuteInTransaction(uow =>
        {
            var result = uow.UserJournals.Get()
                .Where(cd =>
cd.Id.Equals(id)).First();
            if (result == null)
            {
                return false;
            }
            result.IsPublic = isPublic;
            uow.UserJournals.Update(result);
            uow.SaveChanges();
            return true;
        });
    }
}
```

După cum poate fi văzut din codul prezentat mai sus, în acest serviciu interogăm în mai multe moduri baza de date cu ajutorul Design Pattern-ului descris anterior, Unit Of Work, și extragem informații ale entităților sub formele dorite. În situația dată ne dorim să obținem pagina de jurnal cu id-ul respectiv. Verificăm dacă din baza de date s-a extras rezultatul urmând, în cazul favorabil, să facem un update câmpului „IsPublic” și să salvăm modificările, salvarea fiind făcută prin metoda „SaveChanges”, care am văzut mai devreme, se află în clasă „UnitOfWork”.

Pentru a crea interogări asemănătoare query-urilor din SQL am folosit o componentă specifică .Net și anume LINQ (Language-Integrated Query). LINQ reprezintă un ansamblu de tehnologii care integrează capacitățile de interogare în limbajul C#. Prin intermediul LINQ, o interogare devine o componentă esențială a limbajului, similar cu cea a unei clase, metode și eveniment [28].

3.2.5 Presentation layer

Presentation layer are ca scop afișarea interfeței utilizatorului și facilitarea interacțiunii directe. Acesta gestionează toate datele necesare pentru partea clientului. Scopul principal al stratului de prezentare este de a prelua datele de intrare, trimițând solicitările utilizatorilor către servicii și furnizând rezultate mai departe. Acesta este accesibil într-un browser și conține elemente de UI care comunică cu stratul de sistem [29].

Am utilizat în acest layer al soluției mele design pattern-ul Model-View-Controller. În plus, în acest strat se află și logica necesară pentru autentificare și autorizare.

Model

Pentru a restricționa accesul direct la informații private venite din baza de date, am folosit modele, o componentă specifică unei aplicații MVC. Cu ajutorul unui model pot extrage doar anumite date din entitățile mele.

Controller

Controller-ele reprezintă în mod fundamental unitatea centrală a aplicației ASP.NET MVC. Acestea sunt primii destinatari care interacționează cu cererile HTTP primite. Astfel, controller-ul decide care model va fi selectat și apoi preia datele din model și le transmite către view-ul corespunzător. De fapt, controllerele controlează fluxul general al aplicației, preluând intrările și redând ieșiri adecvate. Pentru manipularea și recunoașterea request-urilor HTTP se folosesc anumite atrbute (de exemplu `HttpGet` recunoaște acțiunea de aducere din baza de date iar pentru `HttpPost` de adaugare în baza) [30]. În cazul în care dorim ca doar o anumită categorie de utilizatori să poată apela o metodă din controller, putem adăuga un atrbut de `[Authorize]` care indică faptul ca doar userul cu rolul indicat poate accesa acea funcționalitate.

În continuare, voi exemplifica ceea ce am explicat printr-un exemplu propriu din soluție:

```
[Authorize(Roles = "Admin")]
[HttpPost]
public ActionResult ActivateUser(Guid id)
{
    var result = _doctorAccountService.ActivateUser(id);
    if (result == true)
    {
        return RedirectToAction("GetCVPageVerification");
    }
    else
    {
        return View("Actiunea nu a avut succes");
    }
}
```

View

Un view reprezintă un document standard (X)HTML care poate conține scripturi. Acesta conține tot codul care alcătuiește pagina pe care un utilizator o vede în momentul deschiderii paginii. Un view este scris într-o proporție mare din cod html, existând posibilitatea de a utiliza secvențe de C#. Datele sunt aduse prin intermediul modelului, asigurând comunicarea cu controller-ul.

4. EXPERIENȚA UTILIZATORULUI

În cadrul acestui capitol se vor aborda funcționalitățile aplicației dintr-o perspectivă a utilizatorilor, având în considerare rolul pe care îl au: specialist, pacient sau administrator. De asemenea, vom oferi un tur detaliat al aplicației însorit de capturi de ecran relevante.

4.1 Perspectiva vizitator

Când un utilizator vizitează pentru prima dată aplicația noastră, va fi întâmpinat de o pagină de start, care îi prezintă scopul și beneficiile oferite de Hygieia. În aceeași pagină, se află și butonul de înregistrare pentru medici, astfel încât un specialist care accesează aplicația și dorește să se alăture comunității noastre trebuie să își creeze un cont în primul rând. Această pagină conține o animație preluată de pe CodePen și încorporată în aplicație pentru a oferi o estetică plăcută și pentru a se potrivi temei medicale [33].

Pe lângă pagina de start, un vizitator poate accesa pagina principală, unde va fi întâmpinat de un carusel interactiv, care, prin imaginile prezentate, evidențiază principiile fundamentale ale aplicației noastre. De asemenea, în pagina principală, vizitatorul are acces la articole informative despre diversi termeni și condiții medicale din domeniul sănătății mintale.

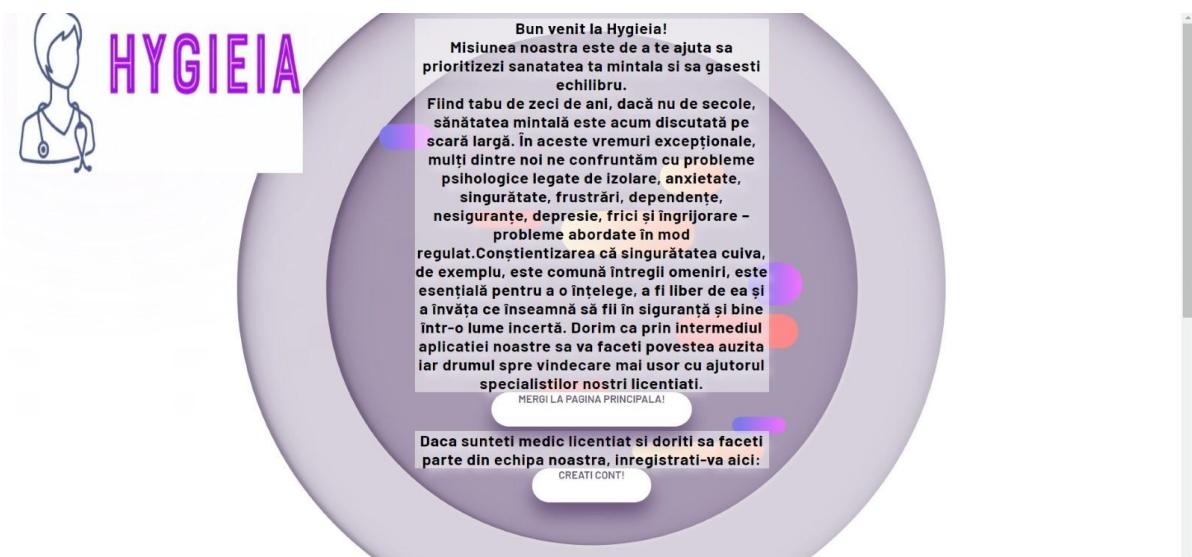


Figura 4.0 Pagina de start

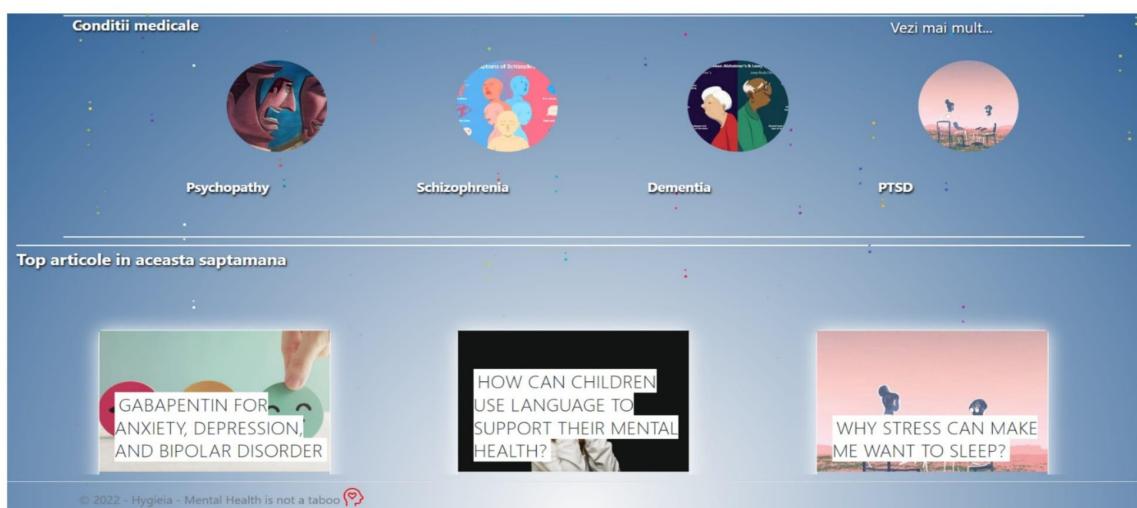
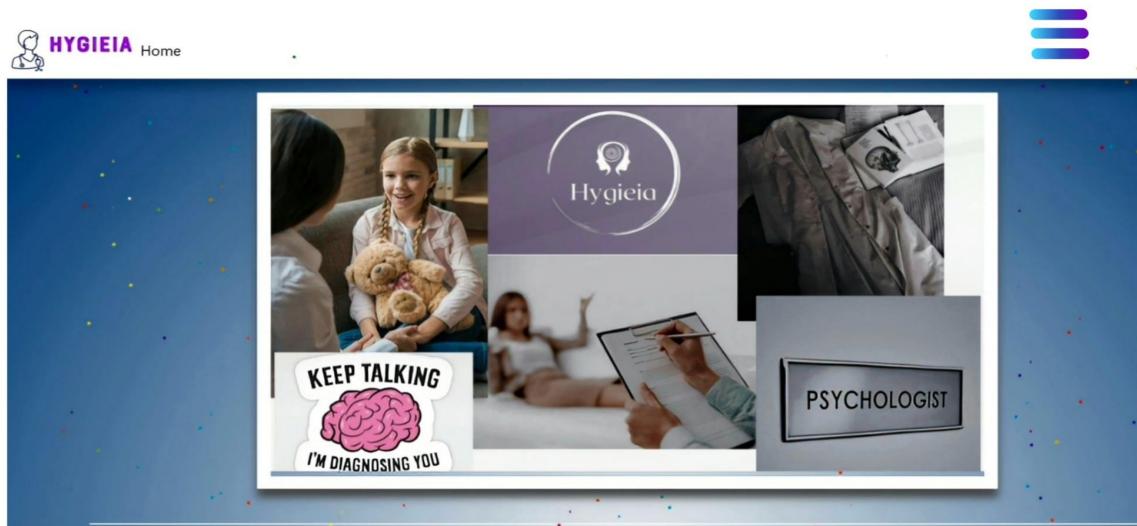


Figura 4.1 Pagina de Home

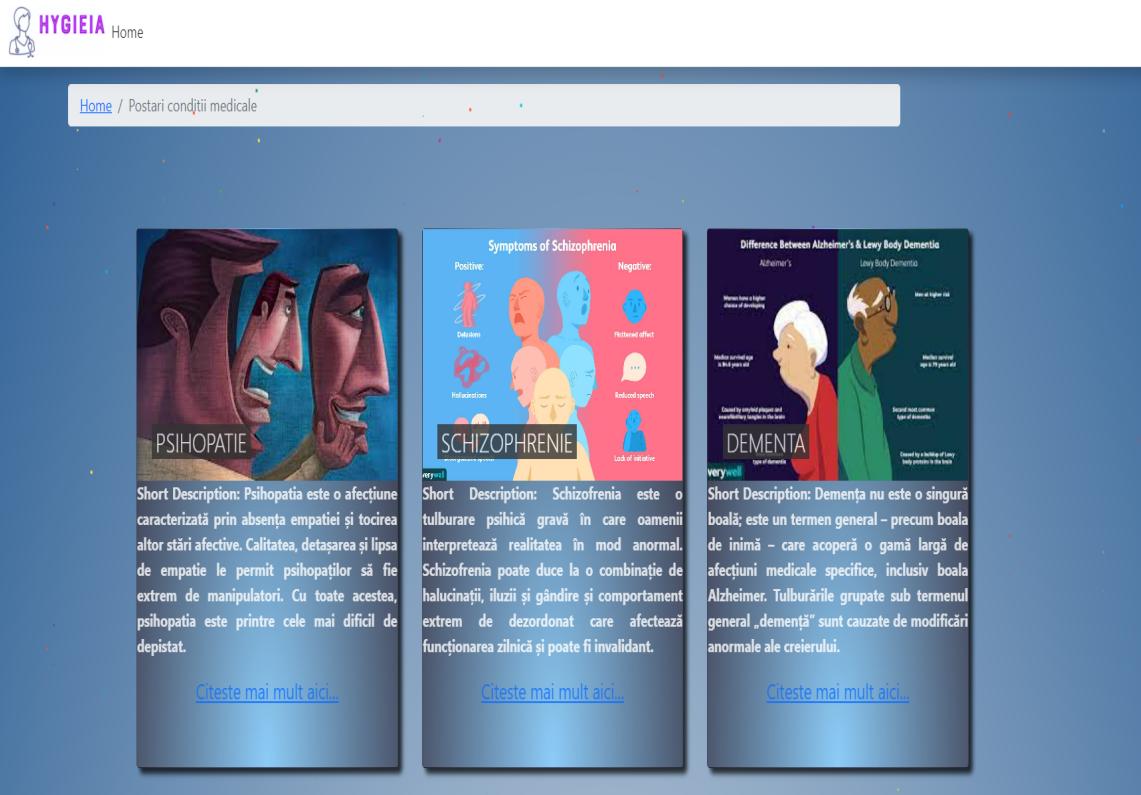


Figura 4.2 Pagina de condiții medical

Demența nu este o singură boală; este un termen general – precum boala de înimă – care acoperă o gamă largă de afecțiuni medicale specifice; inclusiv boala Alzheimer. Tulburările grupate sub termenul general „demență” sunt cauzate de modificări anormale ale creierului. Aceste schimbări declinază o scădere a abilităților de gândire, cunoscute și sub numele de abilități cognitive, suficient de sever pentru a afecta viața de zi cu zi și funcția independentă. Ele afectează, de asemenea, comportamentul, sentimentele și relațiile. Boala Alzheimer reprezintă 60-80% din cazuri. Demența vasculară, care apare din cauza săngerărilor microscopice și a blocării vaselor de sânge din creier, este a doua cea mai frecventă cauză a demenței. Cel care experimentează simultan modificări ale creierului mai multor tipuri de demență au demență mixtă. Există multe alte afecțiuni care pot provoca simptome de demență, inclusiv unele care sunt reversibile, cum ar fi problemele tiroïdiene și deficiențele de vitamine. Demența este adesea denumită în mod incorrect „semnătate” sau „demență senilă”, ceea ce reflectă credința larg răspândită, dar incorrectă, că declinul psihic grav este o parte normală a îmbâtrânerii. Simptomele și semnele demenței Semnele demenței pot varia foarte mult. Exemplul include probleme cu: Memoria pe termen scurt. Urmărirea unei poșete sau portofel. A plăti chitante. Planificarea și pregătirea meseelor. Amintirea întâlnirilor. Călătorind în afara cartierului. Multe afecțiuni sunt progresive, ceea ce înseamnă că semnele demenței încep încet și se agravează treptat. Dacă tu sau cineva pe care îl cunoști întâmpiniți dificultăți de memorie sau alte modificări ale abilităților de gândire, nu le ignorați. Consultați un medic în curând pentru a determina cauza. Evaluarea profesională poate detecta o afecțiune tratabilă. și chiar dacă simptomele sugerează demență, diagnosticarea precoce permite unei persoane să obțină beneficii maxime de pe urma tratamentelor disponibile și oferă o oportunitate de a se oferi voluntar pentru studii sau studii clinice. De asemenea, oferă timp pentru a planifica viitorul. Afiați mai multe: 10 semne de avertizare, etape ale bolii Alzheimer. Demența este cauzată de deteriorarea celulelor creierului. Această afecție interferează cu capacitatea celulelor creierului de a comunica între ele. Atunci când celulele creierului nu pot comunica în mod normal, gândirea, comportamentul și sentimentele pot fi afectate. Creierul are multe regiuni distincte, fiecare dintre acestea fiind responsabilă pentru diferite funcții (de exemplu, memoria, judecata și mișcarea). Când celulele dintr-o anumită regiune sunt deteriorate, acea regiune nu își poate îndeplini funcțiile în mod normal. Urmează cursul nostru gratuit de învățare electronică înțelegerea bolii Alzheimer și a demenței subliniază diferența dintre Alzheimer și demență, simptome, stadii, factori de risc și multe altele. Diferite tipuri de demență sunt asociate cu anumite tipuri de leziuni ale celulelor creierului în anumite regiuni ale creierului. De exemplu, în boala Alzheimer, nivelurile ridicate de anumite proteine din interiorul și din exteriorul celulelor creierului îngreunează celulele creierului să rămână sănătoase și să comunice între ele. Regiunea creierului numită hipocamp este central de învățare și memorie în creier, iar celulele creierului din această regiune sunt adesea primele care sunt afectate. De aceea, pierderea memoriei este adesea unul dintre primele simptome ale bolii Alzheimer. În timp ce majoritatea modificărilor din creier care provoacă demență sunt permanente și se agravează în timp, problemele de gândire și memorie cauzate de următoarele afecțiuni se pot îmbunătăți atunci când afecțiunea este tratată sau abordată: 1. Depresie. 2. Efecte secundare ale medicamentelor. 3. Consumul excesiv de alcool. 4. Probleme cu tiroida. 5. Deficiențe de vitamine.

Figura 4.3 Articol condiție medicală

În momentul de față, articolele medicale sunt preluate de pe internet și traduse în limba română, urmând ca pe viitor să fie scrise de membrii ai comunității sau preluate din cărți de specialitate.

În plus, vizitatorul poate accesa și paginile de autentificare și înregistrare, unde are posibilitatea de a crea un cont nou sau, dacă este deja utilizator al aplicației, de a se autentifica.

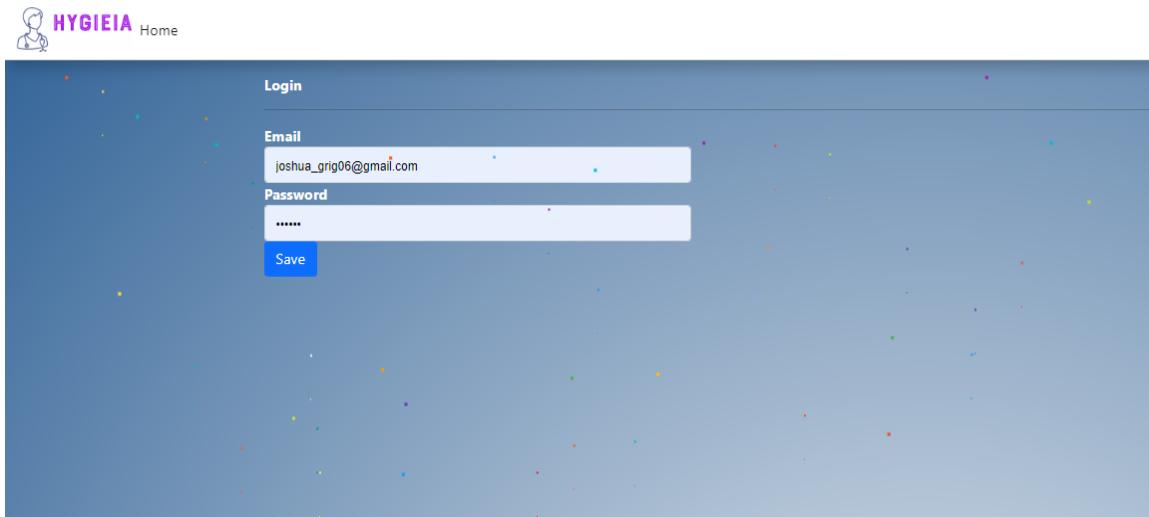


Figura 4.4 Autentificare utilizator

The screenshot shows a user registration form with the following fields:

- Nume
- Prenume
- Data nastere: 01/01/0001 12:00 AM (with a calendar icon)
- Email
- Prefix telefonic
- Numar Contact
- Username: jošhua_grig06@gmail.com
- Categorie Socială: Elev (dropdown menu)
- Asigurare medicală(Da/Nu): Da (dropdown menu)
- Parola: (password field)
- Country
- County
- City
- Strada, Numar
- Bloc, Scara, Apartament
- Sector
- Cod Postal
- Imagine Profil: Choose File (input field) - No file chosen
- Creaza Cont (Create Account button)

Figura 4.5 Înregistrare utilizator

Așa cum am discutat mai devreme, pe pagina de start avem posibilitatea de a fi redirecționați către pagina de înregistrare a unui specialist. În acest caz, specialistul va trebui să completeze toate câmpurile disponibile. Atunci când apasă butonul "Creează cont", se vor verifica unele dintre informațiile furnizate, în Registrul Medicilor din România, doar în cazul în care utilizatorul are specializarea în psihiatrie. Dacă medicul este găsit în registru, contul va fi creat. În cazul în care specializarea este diferită, nu se va efectua verificarea folosind API-ul menționat. Contul specialistului nu va fi complet activat în acel moment (nu va putea primi cereri de programare sau interacționa cu pacienții), și nu va apărea în lista publică a medicilor. În schimb, va fi necesară verificarea CV-ului de către administrator și, eventual, un interviu telefonic. Aceste metode de verificare ne asigură că comunitatea noastră va include doar cei mai buni specialiști.

Alatura-te comunitatii noastre de specialisti!

Nume

Prenume

Email

Numar Contact

Specialitate Psihiatrie ▾

Descriere

Durata Sedinta(minute)
0

Parola
.....

Locatie Cabinet

Tara

Judet

Oras

Strada, Numar

Bloc, Scara, Apartament

Sector

Cod Postal

Imagine Profil
Choose File No file chosen

Adaugati CV (Acesta va fi verificat cat mai repede posibil de catre admin, pentru a va activa contul.)
Choose File No file chosen

Creaza cont

Figura 4.6 Înregistrare medic specialist

4.2 Perspectiva pacient

Pentru utilizatorul cu rolul de Pacient, aplicația oferă o gamă variată de pagini interactive, menite să îi ofere o experiență satisfăcătoare și să îl îndrume într-o călătorie

înfloritoare către îngrijirea de sine. Dacă utilizatorul dorește să împărtășească gândurile și experiențele sale cu întreaga comunitate, el are acces la un forum, unde poate găsi sprijinul și alinarea de care are nevoie. Pagina forumului este compusă dintr-un buton ce va deschide un modal prin care se poate crea o nouă discuție, iar discuția creată este vizibilă pe pagină. De asemenea, se pot căuta discuții în funcție de topic (topicul reprezintă subiectul principal al discuției, reprezentat printr-un simptom sau diagnostic).

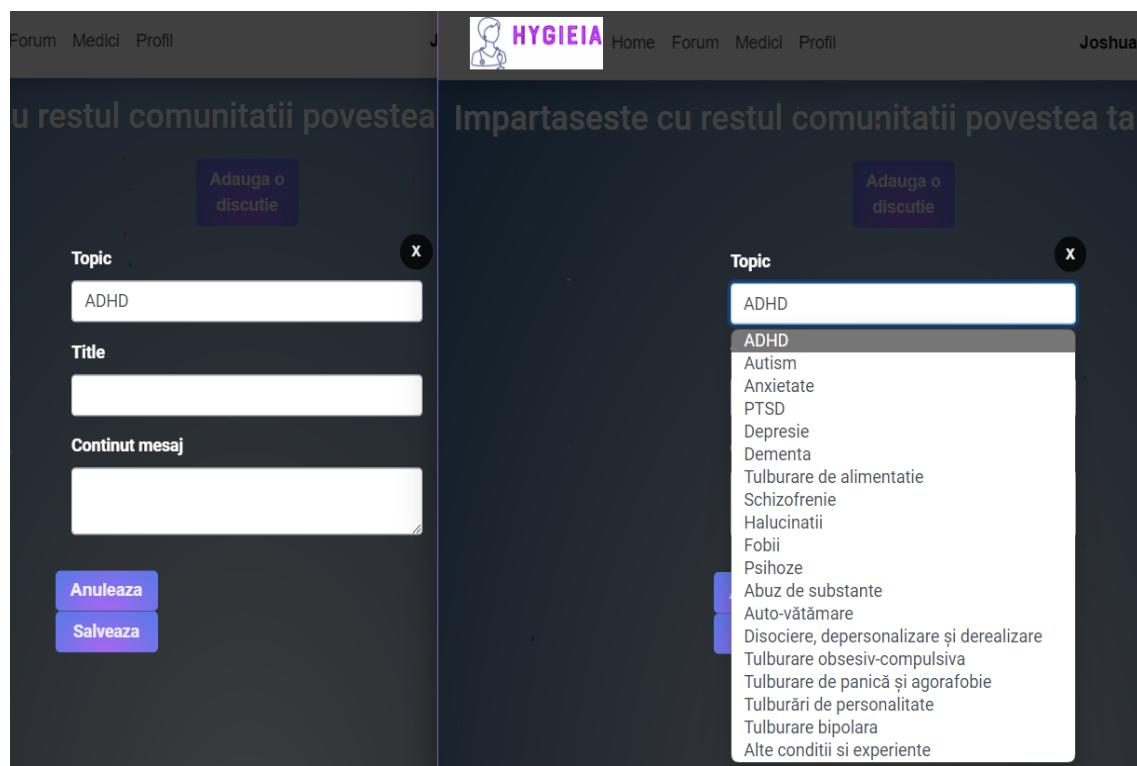


Figura 4.7 Modal creare discuție

Figura 4.8 Topicuri discuție

The screenshot shows a forum interface with a blue header bar. The header contains the text "Impartaseste cu restul comunitatii povestea ta" and a purple button labeled "Adauga o discutie". Below the header is a search bar with the placeholder "Cauta discutie dupa topic: ex: Depresie, ADHD...". The main content area is titled "Discutii" and displays six discussion posts in a grid:

- Grigorescu Joshua** (Profile picture of a man) - 5/30/2023 10:00:52 PM - Topic: Halucinatii
Lucrurile materiale au prins viata...
[Vezi Discutie...](#)
- Grigorescu Joshua** (Profile picture of a man) - 4/17/2023 7:47:31 PM - Topic: ADHD
test...
[Vezi Discutie...](#)
- Barbu Oliver** (Profile picture of a man) - 9/22/2022 11:22:52 AM - Topic: Schizophrenie
De ce sunt fantomele atât de preocupate
[Vezi Discutie...](#)
- Craciun Alexa** (Profile picture of a woman) - 9/11/2022 10:25:45 PM - Topic: Schizophrenie
Cred in ceea ce spun vocile si face ce
[Vezi Discutie...](#)
- Voinea Mira** (Profile picture of a woman) - 9/11/2022 10:22:58 PM - Topic: Tulburare bipolară
Teama ca cineva sa imi citeasca
[Vezi Discutie...](#)
- Barbu Oliver** (Profile picture of a man) - 9/11/2022 10:18:34 PM - Topic: Anxietate
Prefer să fiu mort decât în viață (nu)
[Vezi Discutie...](#)

Figura 4.9 Pagina forum

Se oferă posibilitatea de a accesa o discuție, iar dacă utilizatorul are intenția de a se alătura, acesta poate lăsa un comentariu, inițiind astfel o conversație cu autorul sau cu alții utilizatori.

Forum Page / Discussion Page

 Craciun Alexa
Topic: Schizophrenie

Sotul meu aude voci și crede că sunt oameni adevărați care îl spionează. A început să facă ceea ce îi spun el să facă și vrea și eu. El strigă la el să tacă. I-au spus să se culce și să tacă și vor vorbi între ei despre a-l lăsa singur și să nu-l mai privească. L-am observat scuipat mult, lucru pe care nu îl suport, dar nu a fost niciodată unul care să scuipe. Vociile lui i-au spus că atunci când înghețe, sună de parcă spune o curvă rasiabilă foarte urâtă. Din acest motiv, el doar scuipă în loc să înghețe. I-au spus că vor fi aiici în scurt timp să-l ia și să iasă afară, să se pună în genunchi și să aștepte. A stat așa afară ore întregi. Nu înțeleg de ce crede tot ce spun ei. Pot să spun sau să fac ceva pentru a-l ajuta? Sunt la capătul mintii! Îți jur că dacă mă linștește încă o dată, îmi pot pierde mintile.

9/11/2022 10:25:45 PM 2 Comments

Alatura-te discuției

Adaugă comentariu

 Craciun Alexa

Nu înțeleg de ce crede tot ce spun ei. Pot să spun sau să fac ceva pentru a-l ajuta? Sunt la capătul mintii! Îți jur că dacă mă linștește încă o dată, îmi pot pierde mintile.....

9/13/2022 10:58:56 AM

 Grigorescu Joshua

Consider ca ar trebui să vorbesc cu un medic psihiatru legat de problemele sotului dumneavoastră, pe mine m-a ajutat să înțeleag că cea ce vedeam nu este real iar medicamentația recomandată a pus capăt multor nopti nedormite datorate vocilor.

3/17/2023 1:09:23 AM

Figura 4.10 Pagina unei discuții

Pentru a facilita experiența de găsire a unui specialist (psihiatru, terapeut, psiholog) și de realizare a programărilor, atât online cât și la cabinet, am implementat în aplicație o listă cu medici disponibili. Pentru a asigura o experiență cât mai personalizată, am introdus diverse filtre de căutare a unui medic (precum locația, recenziile, prețul și specializarea acestuia). Pe pagina dedicată fiecărui medic, în afară de informațiile legate de activitatea sa și locația cabinetului, există și funcționalități suplimentare, precum posibilitatea de a adăuga recenzii și de a programa o consultare.

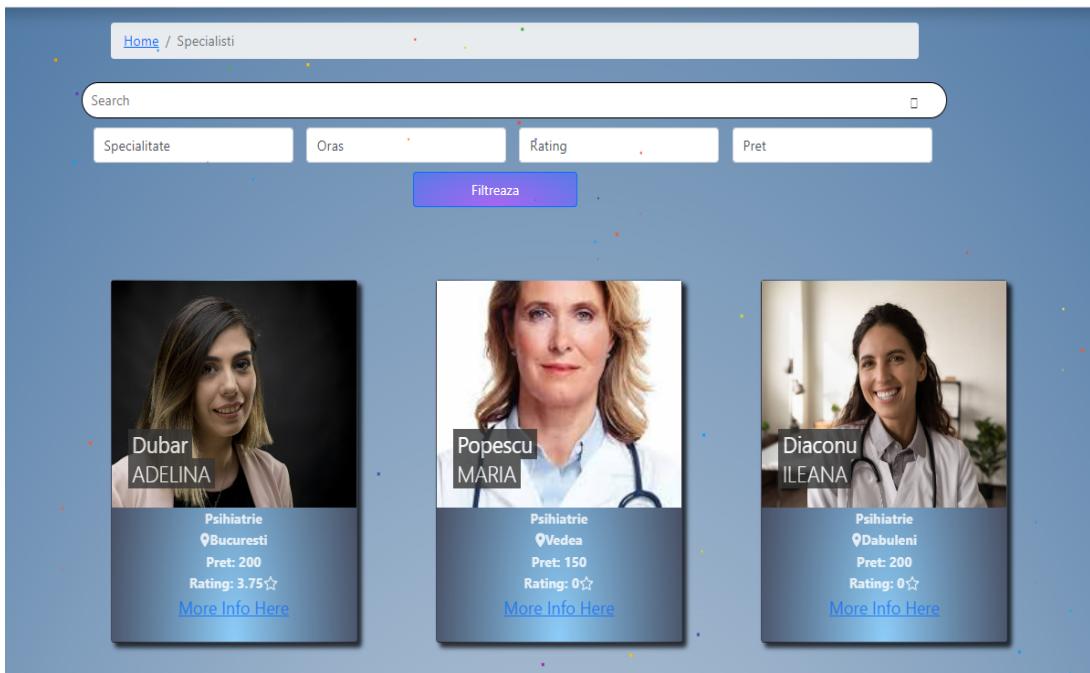


Figura 4.11 Pagina specialiști

Dr. Dubas Adelina-Raluca
Psihiatrie
Bucuresti, Romania
0723153213
Email:
dubas-adelinaR@yahoo.com
40 min durata sedintei
Pret sedinta: 200

★★★★★

[Descriere](#) [Locatie Cabinet](#) [Rating & Comentarii](#) [Programare Sedinta](#)

Locatie Clinica
Romania, Bucuresti, 1, Str. Jules Michelet nr. 7, et.1

0723153213

Map Satellite

Lasa un review medicului.

Submit

Craciun Alexandra
12/19/2022 11:52:53 PM
Doamna doctor a fost foarte rabdatoarea sa asculte problemele mele si tratamentul pe care mi l-a dat m-a ajutat foarte mult. Acum ma duc doar la control. Am recomandat-o si altor apropiati care au avut nevoie si au fost f. multumiti

Grigorescu Joshua
12/20/2022 8:15:00 PM
Am recomandat-o si altor apropiati care au avut nevoie si au fost multumiti

Figura 4.12 Pagina medic + pagina recenzii

Pentru a simplifica procesul de programare, am dezvoltat o pagină specială care oferă o serie de facilități esențiale. Această pagină permite pacienților să aibă acces la programul medicului pentru diverse zile și să completeze un formular pentru a realiza o programare. Utilizatorul poate alege ziua dorită, iar în funcție de aceasta, se vor afișa opțiuni de ore disponibile ale medicului (ore în care acesta nu are alte consultații). De asemenea, utilizatorul poate selecta tipul de consultatie dorit, iar în același loc i se furnizează detalii relevante privind sedința (prețul și durata). După ce programarea este creată, medicul primește o notificare și are posibilitatea de a o accepta sau respinge.

Work Day	Start	End	Notes
Monday	09:00 AM	16:00 PM	Pauza intre orele 12:00-13:00/16:00-17:00

Data și ora

Data: mm/dd/yyyy

Ora: 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00

Data și ora pot fi modificate de către specialist. Voi primi o notificare când apare o modificare.

Tipul sedintei:

Tip sedintă (Fizic/Online)

Fizic

Online

Detalii pret și durată sedintă:

Taxa Servicii: 200

Durata consultatie: 40

Trimite Programarea

Reservarea ta va fi confirmata numai dupa ce specialistul iti va accepta cererea (poate dura pana la 24 de ore).

Figura 4.13 Pagina de programări

În pagina de profil, utilizatorul își poate vizualiza sau edita informațiile personale, sau chiar pagini precum istoric medical, programările din ziua respectivă și jurnalul propriu creat în aplicație.

Istoricul medical reprezintă o pagină ce furnizează un rezumat al consultației, creat de către specialist în urma fiecărei programări. Acesta va conține un card cu o imagine sugestivă, numele medicului, data și diagnosticul. Pentru a accesa mai multe detalii despre acest rezumat, se poate apăsa cercul de meniu din interiorul său.

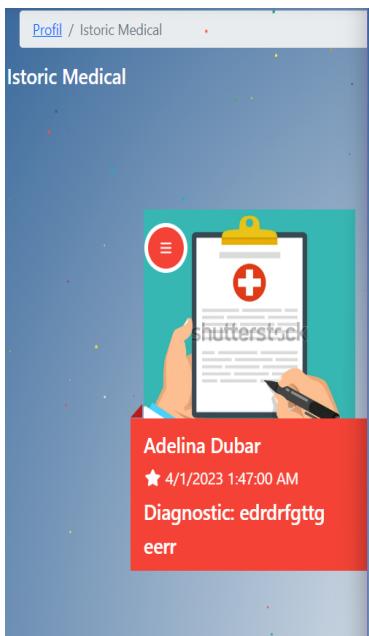


Figura 4.14 Pagina de istoric medical

Report Mrdical		Detalii	
Medic Specialist:	Adelina Dubar	Data Consult	4/1/2023 1:47:00 AM
Pacient:	Joshua Grigorescu	Diagnostic Rezultat	edrdrfttg eerr
Tip consultatie:	Online	Prescriptie	dfdfghghb
Raport Psihiatric			
Istoric Medical			sedrfgg
Sumar Consultatie			sadfvgvgb rf
Doctor			
> Dr. Adelina Dubar			
Semnatura			
Semnatura semneaza aici			

Figura 4.15 Pagina raport medical

Am menționat anterior despre accesarea paginii de jurnal. Cu scopul de a sprijini pacientul în a-și aduna toate gândurile într-un spațiu privat sau chiar de a facilita sesiunile deschise între specialist și pacient, am inclus posibilitatea ca utilizatorul să-și creeze pagini de jurnal, care pot rămâne private sau pot fi partajate cu medicul. Mai jos sunt atașate câteva capturi cu pagina de jurnal.

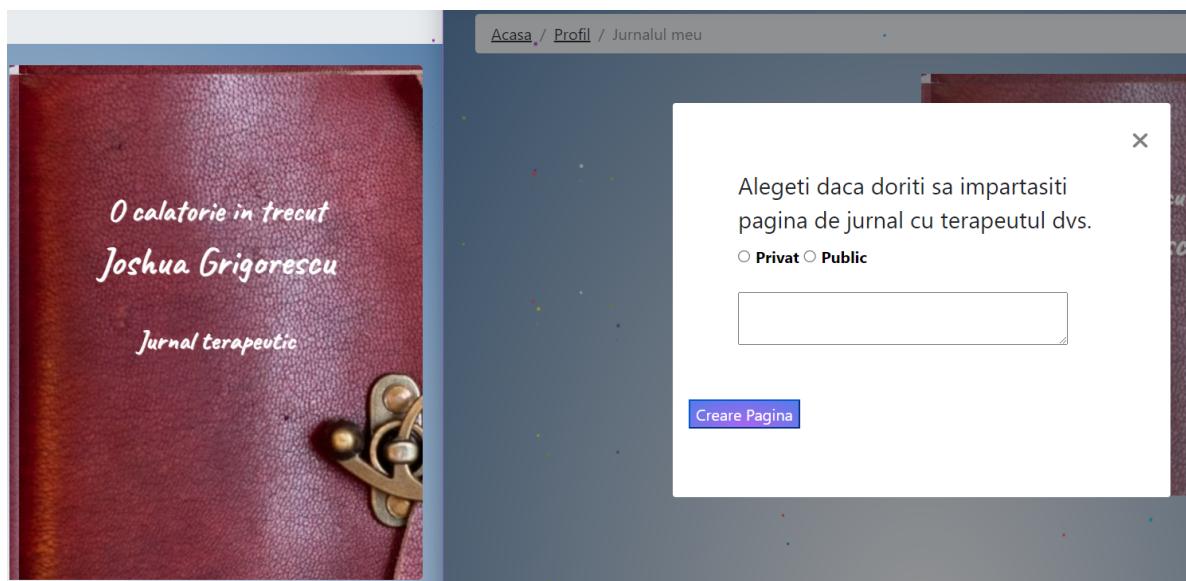


Figura 4.16 Jurnal

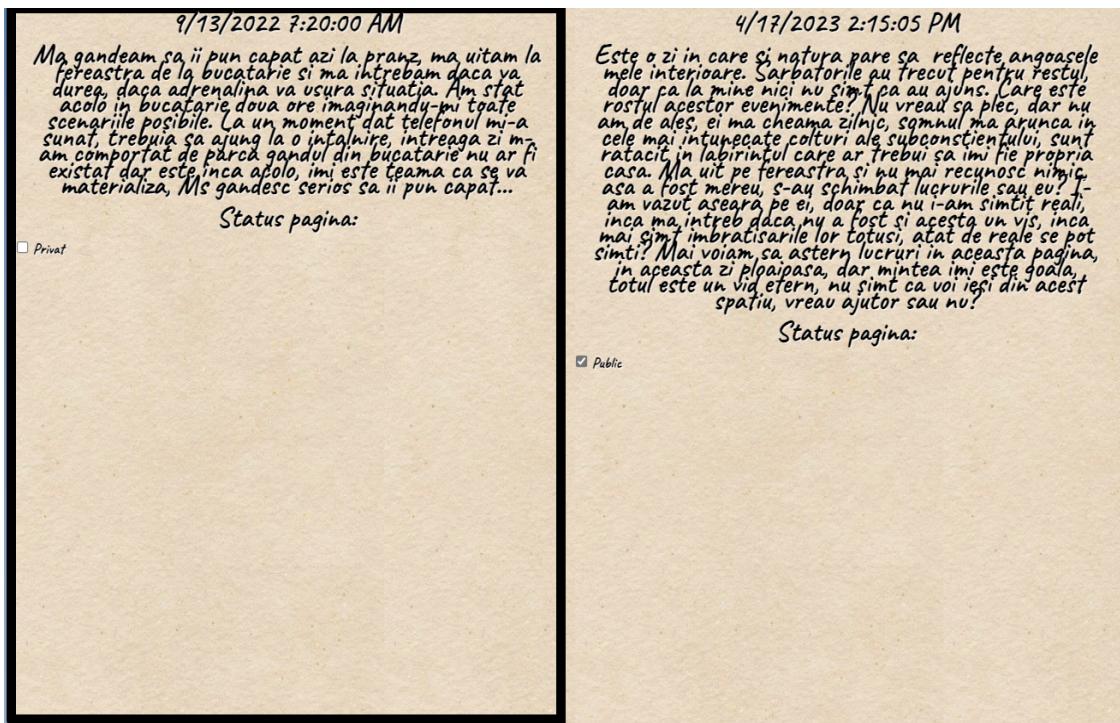


Figura 4.17 Pagini de jurnal

Tot în cadrul paginii de profil, se oferă posibilitatea accesării programărilor online din ziua respectivă, prezentate sub forma unor carduri. Aceste carduri conțin informații precum numele medicului, ora programării, numărul de telefon și un buton inițial pentru plată, care va redirecționa utilizatorul către pagina de PayPal. Prin intermediul acestei platforme se va efectua plata pentru ședința programată. După finalizarea procesului de plată, pacientul va fi redirecționat înapoi pe pagina programărilor, unde butonul de plată va fi înlocuit cu un buton pentru accesarea videoconferinței.

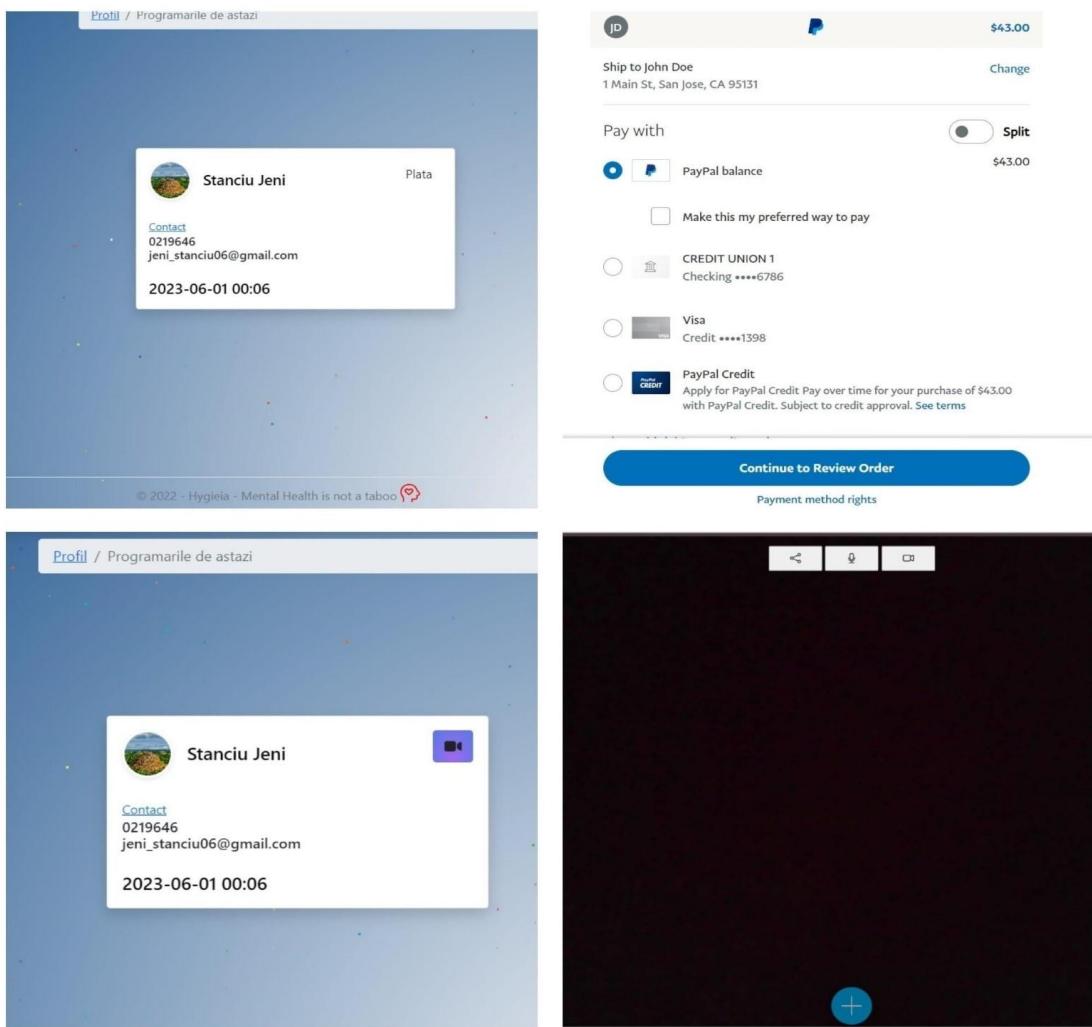


Figura 4.18 Flow-ul de accesare consultatie

4.3 Perspectiva specialistului

Pentru utilizatorul cu rolul de Specialist, aplicația pune la dispoziție pagini interactive, concepute special pentru a ajuta medicul specialist să ofere servicii medicale de înaltă calitate.

În momentul inițial al conectării în cont, până când administratorul finalizează activarea contului specialistului, pe pagina principală un mesaj de atenționare va fi afișat pentru a informa despre această stare. În captura de mai jos putem vedea pagina de home a unui cont care nu este încă activat:

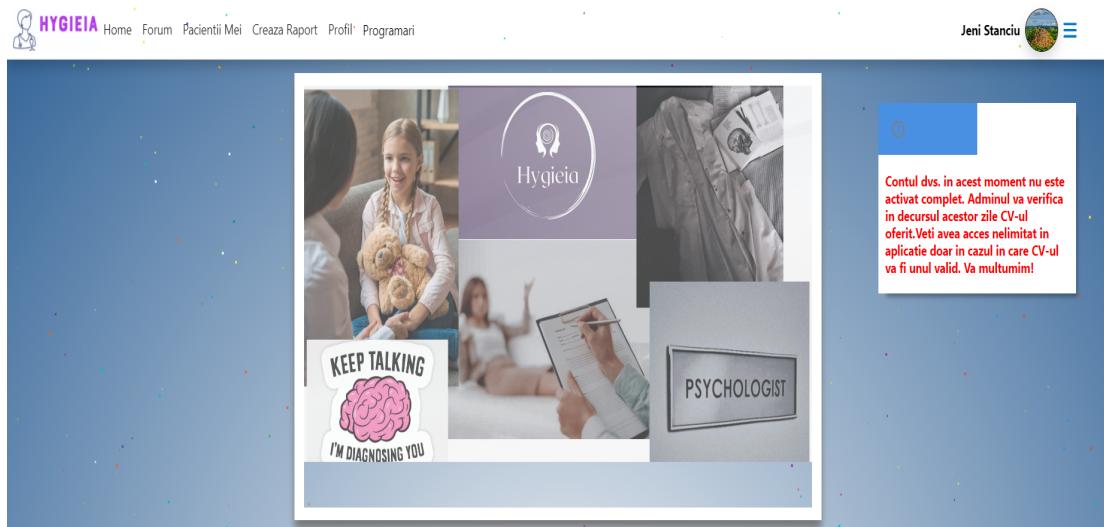


Figura 4.19 Pagina Home Cont Inactiv

Prin intermediul aplicației, specialistul poate accesa o pagină dedicată pacienților, pe care sunt listati toți pacienții săi. Odată accesat profilul unui pacient, medicul va fi redirecționat către un dashboard, unde poate consulta rapoartele medicale ale pacientului (create atât de acesta, cât și de alți medici) și paginile publice din jurnalul pacientului.

The screenshot shows the HYGIEIA platform's patient list page. At the top, there is a navigation bar with links: Home, Forum, Pacientii Mei, Creaza Raport, Profil, Programari, and a user profile for Adelina Dubar. Below the navigation bar is a breadcrumb trail: Home / Pacientii mei. There is also a search bar with the placeholder 'Caută Pacient...'. The main content area displays two patient profiles in cards:

- Nume:** Barbu Oliver
Categorie: Student
Asigurat: Nu
- Nume:** Grigorescu Joshua
Categorie: Adolescent
Asigurat: Nu

Each card includes contact information: phone number (0789967332, 0777898008), email (oliver.b001@gmail.com, joshua_grig06@gmail.com), and a link 'Mai multe informații'.

Figura 4.20 Pagina Pacienți

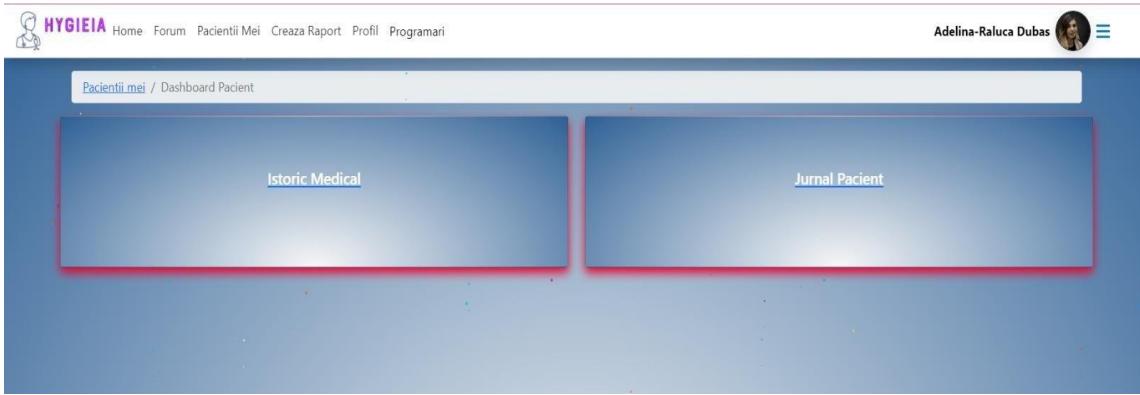


Figura 4.21 Dashboard Pacient

The screenshot shows a split-screen view. On the left, there is a large area for handwriting, with a sample note written in Romanian: "Este o zi în care și natura pare să reflecte angoscile mele interioare. Sarbatorile nu trecut pentru festul doar că la mine încă nu simt să au ajuns. Cine este roșul acestor evenimente? Nu vreau să plec, dar nu am nicio altă moștenire să îl duc cu mine în urmă, cel mai întunecate fătu în subconștiință, care s-a rătăcit în labirintul care ar trebui să îmi fie propria casa. Ma uit pe ferestre, și îmi mai recunoșc nimic, asa a fost mereu, să-ău schimbat lucrurile sau eu? Î-am vazut aşteptă pe ei, doar că nu i-am simțit reali, încă nu, întreb dacă nu a fost și acesta un vis, încă nu simt înțelegerile lor totuși, atât de reală și pot simți înțelegerile lor, și îmi doresc să îl întâlnesc pe el, în aceasta zilă logodna, dar întâlnea îmi este golia, totul este un viață eternă, nu simt ca voi fi din acest spațiu, vreau ajutor sau nu?"

On the right, there is a digital summary of the report, titled "Istoric Medical". It includes a placeholder image of a medical clipboard, the author's name "Joshua Grigorescu", the date "4/1/2023 1:47:00 AM", and the diagnostic code "edrdrfgrttg eerr".

Figura 4.22 Jurnal Pacient

Figura 4.23 Raport Pacient

Din bara de meniu, un specialist are opțiunea de a accesa pagina destinată creării unui raport medical. Acest raport reprezintă un rezumat exhaustiv al consultației finalizate și va include informații esențiale precum numele pacientului, diagnosticul stabilit, strategia de consult și alte date relevante ce caracterizează întâlnirea medicală.

Raport Medical

Pacient

Email pacient

Medic Specialist

Data Raport

mm/dd/yyyy --::--

Descriere Raport

Istoric medical al pacientului

Diagnostic rezultat în urma consultului

Strategie de consult

Prescriptie medicala

Notite

The screenshot shows a user interface for creating a medical report. The top bar is dark blue with the title "Raport Medical". Below it, there are several input fields with labels in white text. The first field is "Pacient" with a dropdown arrow. The second is "Email pacient". The third is "Medic Specialist" with a dropdown arrow. The fourth is "Data Raport" with a date input field and a calendar icon. The fifth is "Descriere Raport". The sixth is "Istoric medical al pacientului". The seventh is "Diagnostic rezultat în urma consultului". The eighth is "Strategie de consult". The ninth is "Prescriptie medicala". The tenth is "Notite". At the bottom left are two buttons: "Save" (dark blue) and "Cancel" (purple).

Figura 4.24 Pagina Creare Raport

Pe pagina de profil a specialistului, pe lângă opțiunea de a vizualiza și manipula informațiile personale, există și opțiunea de a accesa pagina de programări pentru ziua respectivă, pagina cu programul medicului și cererile de programare.

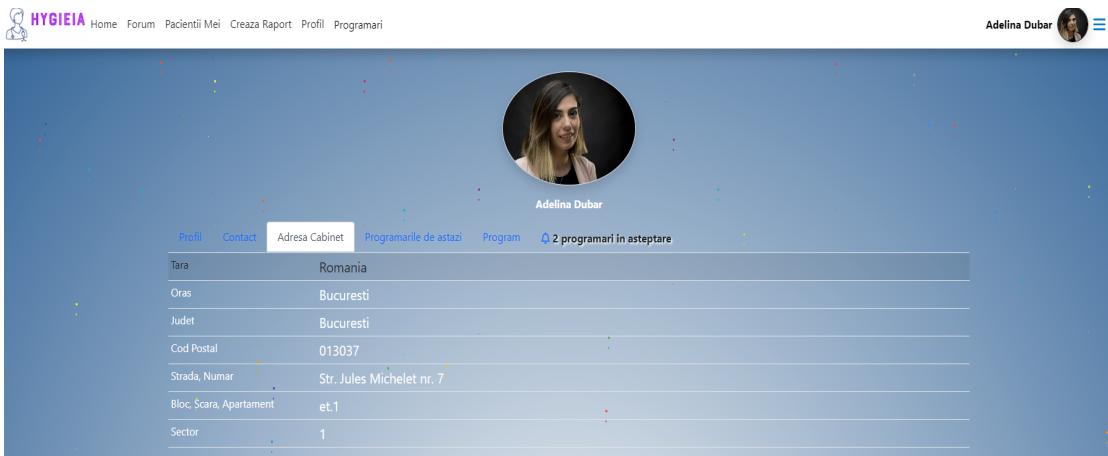


Figura 4.25 Pagina Profil Medic

Pe pagina "Program", specialistul are posibilitatea de a adăuga propriul program de lucru, pentru a-și organiza mai eficient activitățile și pentru a-l face cunoscut utilizatorilor care doresc să își programeze o consultăție. Programul cuprinde informații precum ora de începere și încheiere a activității, precum și alte detalii relevante, cum ar fi perioada de pauză de masă în fiecare zi a săptămânii.

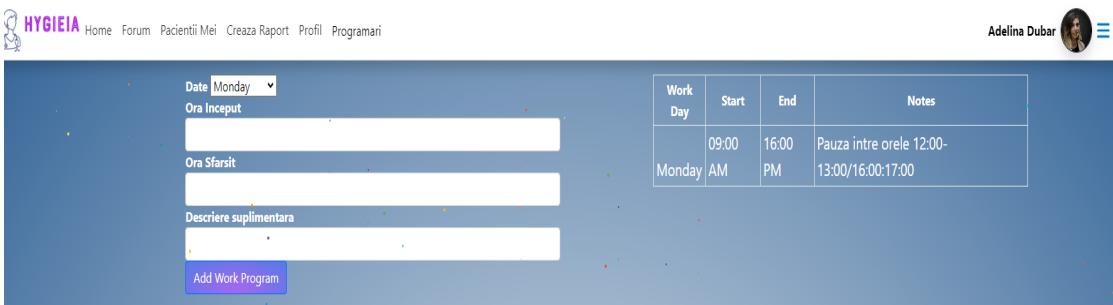


Figura 4.26 Pagina Program Medic

Atunci când un pacient înregistrează o programare, numărul de programări în așteptare în pagina de profil a medicului va crește, iar în tabel va apărea cererea de programare, al cărei status poate fi modificat, permitând acceptarea sau refuzarea programării în cauză.

Programari online în așteptare				
ID Programare	Data	Pacient	Status	Acțiune
1055	2023-03-22 12:00	Grigorescu Joshua	În Așteptare	
1059	2023-05-25	Grigorescu Joshua	În Așteptare	

Figura 4.27 Pagina Cereri Programări

De asemenea un specialist mai are posibilitatea de a accesa lista programărilor fizice sau online, programările din ziua respectivă, din care va accesa videoconferința. În plus, acesta poate interacționa prin intermediul forumului, oferind ajutor și împărtășind experiențe valoroase cu utilizatorii.

4.4 Perspectiva administratorului

Aplicația dispune de un singur cont cu rolul de Admin, creat în ideea că se pot înregistra mai mulți administratori cu el. Administratorul dispune de o pagină separată de login (alți utilizatori nu se pot autentifica prin ea), a cărei rută nu este afișată în aplicație, sporind astfel securitatea contului. Odată autentificat în aplicație cu rolul de admin, în bara de navigare va apărea posibilitatea accesării paginii de „Administrare”.



Figura 4.28 Dashboard Admin

Rolul administratorului constă în gestionarea utilizatorilor, având opțiunea de a șterge sau manipula informațiile ce țin de propria persoana, precum și de a modifica starea programărilor. De asemenea, administratorul are atribuția de a gestiona articolele de pe pagina principală și discuțiile utilizatorilor. Două alte funcționalități importante ale administratorului includ vizualizarea graficelor medicilor, care cuprind informații legate

de recenziile și programările acestora, și verificarea CV-urilor specialiștilor, în vederea activării complete a conturilor acestora.

În ceea ce privește gestionarea pacienților și specialiștilor, există două pagini distincte, fiecare oferind o modalitate diferită de organizare a informațiilor: fie sub formă de tabele, fie prin opțiunea de vizualizare simplificată, care implică rotirea tabelului și eliminarea anumitor coloane. De asemenea, în ultima coloană a paginilor respective se găsesc butoanele de editare și ștergere a utilizatorului, care funcționează prin intermediul librăriei AJAX pentru a apela metodele corespunzătoare din controller și pentru a oferi un caracter dinamic, asigurând dispariția liniei asociate utilizatorului înainte ca cererea să fie procesată de server.

The screenshot shows a web-based application interface titled "Informatii Specialisti". At the top left, there is a breadcrumb navigation: "Dashboard / Doctors Page". On the left side, there is a vertical sidebar with a "Register Doctor" button. The main content area has a search bar at the top right. Below the search bar is a table with the following columns: Nume, Prenume, Email, Prefix telefonic, Contact, Specialitate, Descriere, Durata Sedinta, Tara, Judet, Oras, Strada, Numar, Bloc, Scara, Apartament, and Sect. There are three rows of data in the table:

Nume	Prenume	Email	Prefix telefonic	Contact	Specialitate	Descriere	Durata Sedinta	Tara	Judet	Oras	Strada, Numar	Bloc, Scara, Apartament	Sect
Dubar	Adelina	dubar-adelin...		0723153213	Psihiatrie	Medic Specia...	40	Romania	Bucuresti	Bucuresti	Str. Jules Mic...	et.1	1
Stanciu	Jeni	jeni_stanciu0...		0219646	Psihiatrie	-	50	Romania	Iasi	Iasi	Str. Vasile Lu...	-	-
Popescu	Maria	popescuMari...		0789967332	Psihiatrie	-	25	Romania	Arges	Vedea	DN67B	-	-

At the bottom of the table, there are navigation buttons: "Anterior" and "Urmator", and a page indicator "Page 0".

Figura 4.29 Informații Specialiști tabel

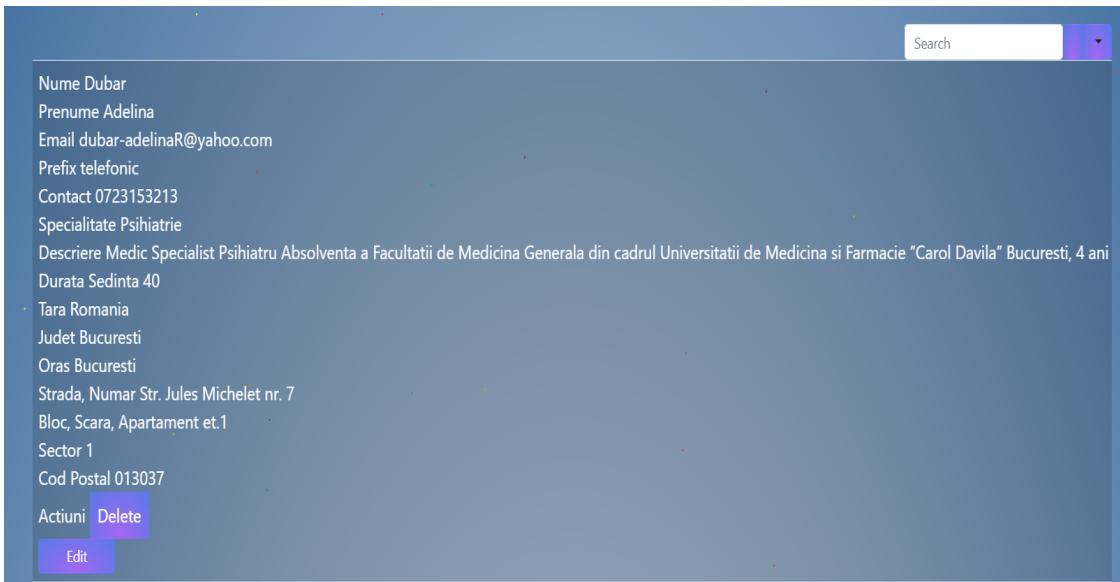


Figura 4.30 Informații Specialiști vizualizare fără coloane

În cazul programărilor vizualizarea este tot sub forma de tabel, oferind posibilitatea de a filtra după numele medicilor. Acțiunea din pagina programărilor este cea de a edita statusul, iar în momentul apăsării pe butonul de editare, coloana cu status a respectivei sesiuni va fi transformată într-un meniu derulant, unde administratorul poate alege un nou status, cum poate fi observat în următoarea captură de ecran.

Programari							
Dubar Adelina		Cauta					
Id Programare		Data Programare		Pacient	Medic	Status	Actiune
44		2022-09-12 01:00		Barbu Oliver	Dubar Adelina	Programare_Realizata	
39		2022-09-12 10:05		Craciun Alexa	Popescu Maria	In_Asteptare	
40		2022-09-12 10:52		Barbu Oliver	Diaconu Ileana	In_Asteptare	
42		2022-09-12 11:05		Barbu Oliver	Diaconu Ileana	Programare_Responsa Programare_Acceptata Programare_Anulata Programare_Realizata	
43		2022-09-12 13:00		Barbu Oliver	Dubar Adelina	Programare_Realizata	

Figura 4.31 Tabel Programări

În ceea ce privește pagina de grafice, aceasta are rolul de a informa adminul în legătură cu modul de gestionare a programărilor cât și despre relația specialist-pacient. Graficele sunt create folosind formule de calcul corespunzătoare - media alcătuită din jumătate din numărul total de evaluări plus scorul mediu al evaluărilor pentru a obține rezultatul final.

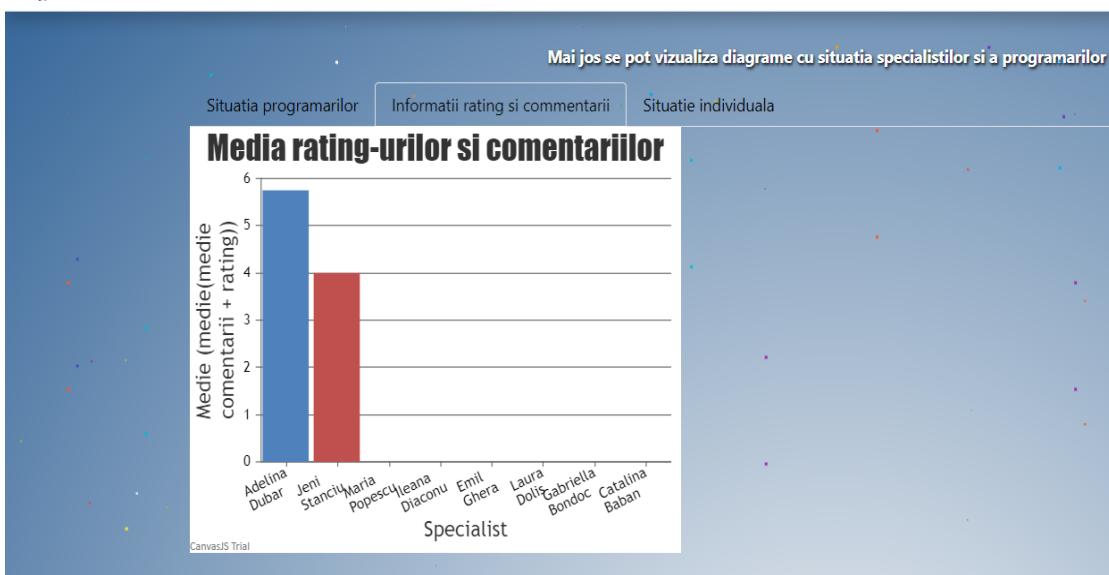


Figura 4.32 Diagrama recenziilor și comentariilor

În ceea ce privește situația programărilor, se calculează procentul programărilor realizate de către fiecare medic din totalul programărilor.



Figura 4.33 Pie Chart Programări

În plus, administratorul are posibilitatea de a accesa situația individuală a fiecărui medic. Secțiunea denumită "Situatie individuală", prezintă specialiștii sub forma unor carduri, iar prin selectarea unui card se va apărea o fereastră care conține două grafice în formă de cerc: unul referitor la starea programărilor fizice (realizate, acceptate, respinse) și unul referitor la starea programărilor online.

Situatia programarilor Informatii rating si comentarii Situatie individuala

Lista specialiști

 <p>Dubar Adelina dubar-adelinaR@yahoo.com.</p>	 <p>Stanciu Jeni jeni_stanciu06@gmail.com.</p>	 <p>Popescu Maria popescuMaria@yahoo.com.</p>
 <p>Diaconu Ileana diaconu_ileana20@yahoo.com.</p>	 <p>Gheră Emil ghera_emilS@gmail.com.</p>	 <p>Dolis Laura dolis@yahoo.com.</p>

Figura 4.34 Lista specialiști secțiunea grafice

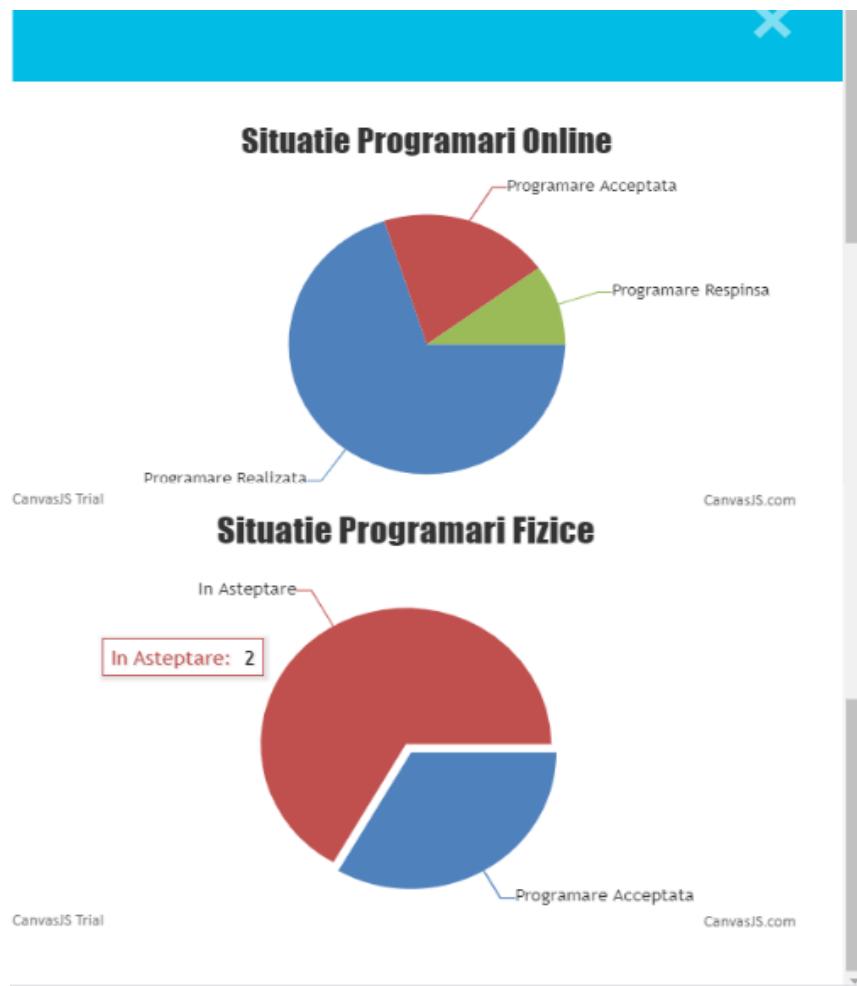


Figura 4.35 Pie Chart-uri Programări Online și Fizice

Deoarece am dorit să prezint situația specialiștilor într-un mod vizual atractiv și ușor de înțeles, pentru realizarea acestui lucru, am utilizat biblioteca CanvasJS.Chart iar ca sursă de inspirație am folosit site-ul oficial [32].

Întrucât procesul de activare a contului pentru specialiști necesită verificarea din partea administratorului, am dezvoltat o pagină specială dedicată acestui scop, care facilitează verificarea CV-urilor și activarea profilurilor. Informațiile sunt organizate într-un tabel, unde pe lângă câteva date despre specialist se regăsesc un link către CV-ul adăugat la momentul înregistrării și un buton pentru activarea profilului. Pentru a oferi o perspectivă mai clară, va fi prezentată o captură de ecran care ilustrează cum, după ce se apasă pe CV, acesta este descărcat în secțiunea de download a browserului.

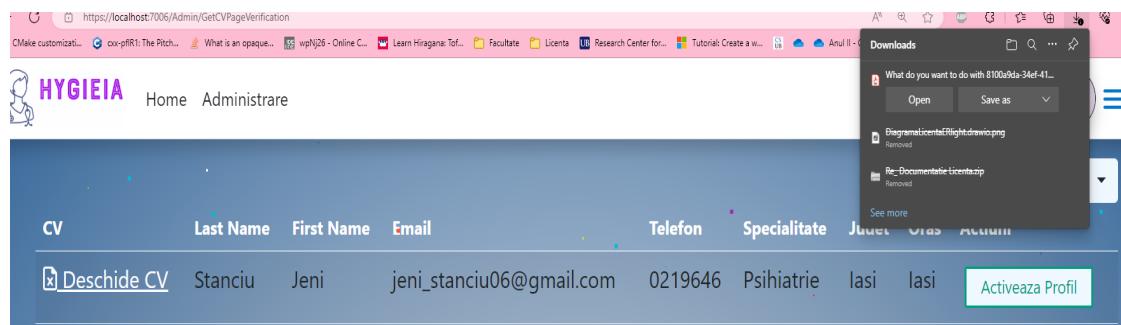


Figura 4.37 Pagina Verificare CV-uri

Majoritatea paginilor din aplicație sunt responsive, fiind concepute atât pentru dispozitive de tip desktop cât și dispozitive mobile.

Figura 4.38 Pagina forum responsive

Figura 4.39 Pagina specialiști responsive

[Doctori](#) / Profil Doctor



Dr. Dubar Adelina
Psihiatrie
📍 Bucuresti, Romania
0723153213
Email:
dubar-adelinaR@yahoo.com

⌚
40 min durata sedintei
Pret sedinta: 200



[Descriere](#) [Locatie Cabinet](#) [Rating & Comentarii](#)

[Programare Sedinta](#)

Despre

Medic Specialist Psihiatru Absolventa a Facultatii de Medicina Generala din cadrul Universitatii de Medicina si Farmacie "Carol Davila" Bucuresti, 4 ani de experienta in cadrul sectiei de psihiatrie adulti a Spitalului clinic de psihiatrie "Prof.Dr.Al.Obregia", experienta in specialitatea psihiatrie 6 ani. Participare la scoala de vara "ABCs of psychotherapy", Strasbourg 2017.

[Doctori](#) / [Profil Doctor](#) / Creaza o programare

Program medic

Work Day	Start	End	Notes
Monday AM	09:00	16:00	Pauza intre orele 12:00-13:00/16:00:17:00

Data si ora

Data
 📅

Ora

Data si ora pot fi modificate de catre specialist. Vei primi o notificare cand apare o modificar.

Tipul sedintei:

Tip sedinta (Fizic/Online)
 Fizic
 Online

Detalii pret si durata sedinta:

Taxa Servicii: 200

Durata consultatie: 40

[Trimite Programarea](#)

Rezervarea ta va fi confirmata numai dupa ce specialistul iti va accepta cererea (poate dura pana la 24 de ore).

Figura 4.40 Pagina specialist responsive

Figura 4.41 Pagina de programări responsive

Concluzii și Direcții Viitoare de Cercetare

1. Concluzii

În cadrul acestei lucrări, am elaborat o prezentare detaliată a soluției dezvoltate cu scopul de a oferi o platformă prietenoasă, informativă și utilă, atât pentru specialiștii în sănătatea mentală, cât și pentru utilizatorii obișnuiți. În ciuda faptului că unele funcționalități deja există în aplicații existente pe piață, precum am prezentat în capitolul I, consider că acestea nu sunt suficient de complete pentru a satisface nevoile utilizatorilor din mediul online. Prin urmare, având în vedere gama de funcționalități și arhitectura utilizată, soluția mea prezintă următoarele avantaje:

- Datorită structurii pe layer și utilizării arhitecturii MVC (Model-View-Controller), aplicația mea este ușor de întreținut pe termen lung, având de asemenea capacitatea de a se extinde foarte ușor.
- Soluția propusă oferă o varietate de funcționalități, care includ surse informative, socializare și consultanță medicală specializată. Prin intermediul acestei soluții, se facilitează o comunicare online directă între specialiști și pacienți, evitând intermediarii și creând astfel o interacțiune mai fluentă cu aplicația.
- Pentru a asigura siguranța utilizatorilor, soluția utilizează tehnici de hashing și salting pentru parole. De asemenea, s-a implementat un sistem de verificare a specialiștilor, cu scopul de a inspira încredere în comunitatea Hygieia.

2. Direcții Viitoare de Cercetare

Cu toate că soluția pe care am dezvoltat-o în prezent are calitățile necesare pentru a fi un produs utilizat pe piață, fiind vorba în special de domeniul medical, consider că există îmbunătățiri și dezvoltări posibile în continuare.

Pentru a crea o experiență îmbunătățită, consider că modul de găsire al unui medic ar putea fi personalizat într-o mai mare măsură. Pe viitor, se poate lua în considerare implementarea unui chestionar mai detaliat, atât pentru specialiști, cât și pentru pacienți, pentru a determina care medic ar fi cel mai potrivit pentru fiecare utilizator în

parte. În plus, integrarea unei componente de chat în platformă ar îmbunătăți comunicarea între pacient și medic.

Un aspect nerezolvat încă este partea de articole din domeniul sănătății mintale, prezентate pe pagina principală a aplicației. Deși există o pagină de creare a acestor articole în partea de administrare, paginile necesare pentru afișarea și accesarea articolelor nu au fost implementate încă. Deoarece doresc ca experiența articolelor să fie asemănătoare cu cea dintr-un blog, am dedicat o perioadă specială de implementare pentru a asigura un aspect vizual plăcut.

Un parteneriat cu instituții medicale specializate și integrarea aplicației în propriile lor soluții de gestionare a pacienților ar aduce beneficii semnificative, oferind un spațiu comun de interacțiune.

De asemenea, o funcționalitate similară raportului medical existent, prin care să se poată crea rețete medicale reale sau integrarea unui mod de scanare a unei rețete create de medic în cabinetul său, ar ajuta persoanele care aleg consultații online.

Cu toate că platforma este în mare parte scalabilă și compatibilă cu dispozitivele mobile, dezvoltarea unei aplicații mobile ar oferi accesul la servicii în orice moment al zilei.

Bibliografie

- [1]. *Mental health goes mobile: The mental health app market will keep on growing* [Mental health app market growth | Deloitte Insights](#) [Accesat 28 Aprilie 2023].
- [2]. *Sorting Through the Noise of Mental Health Apps* [Sorting Through the Noise of Mental Health Apps - UConn Today](#) [Accesat 28 Aprilie 2023].
- [3]. *I'm Fine* [I'm Fine | Ajutor pentru Depresie și Anxietate | Povestea Noastră \(im-fine.app\)](#) [Accesat 15 Iunie 2023].
- [4]. *What Is .NET?* [What is .Net? - Dotnet Explained - AWS \(amazon.com\)](#) [Accesat 15 Iunie 2023]
- [5]. *What is ASP.NET?* [What is ASP.NET? | .NET \(microsoft.com\)](#) [Accesat 01 Iunie 2023]
- [6]. *Introduction to ASP.NET MVC* [Introduction to ASP.NET MVC \(dotnettricks.com\)](#) [Accesat 14 Iunie 2023]
- [7]. *Chapter 1: An Introduction to ASP.NET MVC* [Chapter 1: An Introduction to ASP.NET MVC \(codemag.com\)](#) [Accesat 14 Iunie 2023]
- [8]. *ASP.NET Core And Entity Framework Core CRUD Operations* [ASP.NET Core And Entity Framework Core CRUD Operations \(dotnettricks.com\)](#) [Accesat 15 Iunie 2023]
- [9]. *Getting Started with Entity Framework 6* [Getting Started with Entity Framework 6 | Pluralsight](#) [Accesat 30 Mai 2023]
- [10]. *Registrul C.M.R.* [Registrul Medicilor din Romania \(cmr.ro\)](#) [Accesat 15 Iunie 2023]
- [11]. *Views in ASP.NET Core MVC* [Views in ASP.NET Core MVC | Microsoft Learn](#) [Accesat 02 Iunie 2023]
- [12]. *What is SQL Server* [What is SQL Server \(sqlservertutorial.net\)](#) [Accesat 02 Iunie 2023]
- [13]. *Microsoft SQL Server Management Studio 18* [Microsoft SQL Server - Wikipedia](#) [Accesat 01 Iunie 2023]
- [14]. *What is CSS?* [What is CSS? \(tutorialspoint.com\)](#) [Accesat 28 Mai 2023]
- [15]. *Bootstrap Get Started* [Bootstrap Get Started \(w3schools.com\)](#) [Accesat 28 Mai 2023]
- [16]. *An Introduction to JavaScript* [An Introduction to JavaScript](#) [Accesat 28 Mai 2023]
- [17]. *jQuery Tutorial* [jQuery Tutorial \(tutorialspoint.com\)](#) [Accesat 28 Mai 2023]
- [18]. *WebRTC (Web Real-Time Communications)* [What is WebRTC and how is it used? \(techttarget.com\)](#) [Accesat 29 Mai 2023]
- [19]. *Introduction to SignalR* [Introduction to SignalR | Microsoft Learn](#) [Accesat 15 Mai 2023]

- [20]. *Mark Richards, Software Architecture Patterns* O'Reilly Media, Inc., Chapter 1. [Layered Architecture](#) [1. Layered Architecture - Software Architecture Patterns \[Book\]](#) ([oreilly.com](#)) [Accesat 14 Iunie 2023]
- [21]. *Understanding Object-Relational Mapping: Pros, Cons, and Types* [Understanding Object-Relational Mapping | AltexSoft](#) [Accesat 14 Iunie 2023]
- [22]. *.NET Basics: ORM (Object Relational Mapping)* [.NET Basics: ORM \(Object Relational Mapping\) \(telerik.com\)](#) [Accesat 15 Iunie 2023]
- [23]. *Unit Of Work in Repository Pattern* [Unit Of Work in Repository Pattern - Dot Net Tutorials](#) [Accesat 01 Iunie 2023]
- [24]. *Badia Kharroubi, Microservices Architecture, Domain Entity pattern* [Domain Entity pattern · Microservices Architecture \(gitbooks.io\)](#) [Accesat 01 Iunie 2023]
- [25]. *How to Differentiate Business and Service Layers in Layered Architecture* [How to Implement Layered Architecture | Level Up Coding \(gitconnected.com\)](#) [Accesat 02 Iunie 2023]
- [26]. *Rules to Better Clean Architecture - 10 Rules* [SSW.Rules | Rules to Better Clean Architecture](#) [Accesat 14 Iunie 2023]
- [27]. *ACID Properties in DBMS* [ACID Properties in DBMS - GeeksforGeeks](#) [Accesat 01 Iunie 2023]
- [28]. *Language Integrated Query (LINQ) (C#)* [Language-Integrated Query \(LINQ\) \(C#\) | Microsoft Learn](#) [Accesat 01 Iunie 2023]
- [29]. *Web Applications Architectures: Components, Layers, and Types* [Web Application Architecture Fundamentals: The Full Roadmap \(cleveroad.com\)](#) [Accesat 01 Iunie 2023]
- [30]. *ASP.NET MVC - Controllers* [ASP.NET MVC - Controllers \(tutorialspoint.com\)](#) [Accesat 01 Iunie 2023]
- [31]. *ASP.NET MVC Pie Charts with Index / Data Labels placed Inside* <https://canvasjs.com/asp-net-mvc-charts/pie-chart-index-data-label-inside/> [Accesat 14 Iunie 2023]
- [32]. *Daily UI #008 - 404 Page* [Daily UI #008 - 404 Page \(codepen.io\)](#) [Accesat 15 Iunie 2023]