UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

Meditlo - aplicație web pentru meditații

Absolvent

Munteanu André

Coordonator științific

Conf. Dr. Mureșan Claudia

București, iunie-iulie 2025

Rezumat

În prezent, majoritatea conexiunilor dintre elevi și meditatori se realizează prin recomandări, prin grupuri de Facebook sau prin platforme ca OLX, iar distribuirea de materiale didactice și comunicarea sunt realizate prin aplicații ca WhatsApp sau Teams.

Această lucrare propune o aplicație web care unifică aceste procese într-o singură platformă, simplificând interacțiunea dintre elevi și meditatori.

Aplicația web a fost implementata cu Laravel [1] pentru partea de backend, Blade [2] pentru frontend, iar Livewire [3] a fost utilizat pentru crearea de componente interactive reactive, oferind o experiență fluidă. Pentru stilizarea interfeței grafice, vom folosi TailwindCSS [4], împreună cu daisyUI [5], o bibliotecă de componente care faciliteaza realizarea interfetelor grafice pentru aplicatii web. Vom folosi și Vite [6], deoarece oferă atât Hot Module Replacement (HMR) [7] în timpul dezvoltării, cât și un build optimizat.

Abstract

Currently, most connections between students and tutors are made through recommendations, Facebook groups, or platforms like OLX, while the distribution of educational materials and communication take place via applications such as WhatsApp or Teams.

This paper proposes a web application that unifies these processes into a single platform, simplifying the interaction between students and tutors.

The web application was developed using Laravel [1] for the backend, Blade [2] for the frontend, and Livewire [3] for creating reactive interactive components, providing a smooth user experience. For styling the graphical interface, TailwindCSS [4] was used together with daisyUI [5], a component library that accelerates development. I also used Vite [6], as it offers both Hot Module Replacement (HMR) [7] during development and an optimized build.

Cuprins

1. Introducere

Conform unui raport publicat de Institutul de Științe ale Educației în 2024, „Liceenii din România: Implicarea și autonomia liceenilor” [8], 43% dintre elevii respondenti au declarat că participa la meditații particulare pentru bacalaureat, ceea ce este un procent semnificativ. Acest lucru evidențiază o realitate bine cunoscută în sistemul educațional românesc, necesitatea sprijinului educațional suplimentar la pregătirea pentru examenele importante, cum ar fi bacalaureatul.

Într-o lume ideală, sistemul de învățământ ar trebui să asigure elevilor suficiente resurse și suport pentru a se pregăti eficient în timpul orelor de curs. Însă, volumul mare de materie, împreună cu, de multe ori, calitatea procesului educațional îi determină pe mulți elevi să apeleze la meditații.

În prezent, majoritatea conexiunilor dintre elevi și meditatori se realizează prin recomandări, grupuri de Facebook și prin platforme precum OLX sau Meditații.ro, iar distribuirea materialelor didactice și comunicarea se fac prin aplicații separate, cum ar fi WhatsApp sau Teams. Elevii trebuie să navigheze între mai multe platforme pentru a găsi un meditator potrivit, pentru a primi și trimite materiale și pentru a comunica, ceea ce duce la o experiență mai puțin coerenta.

În acest context, lucrarea propune o platformă unitară care să faciliteze întregul proces de conectare și comunicare intre elevi și meditatori, precum și distribuirea materialelor, simplificând astfel interacțiunea și eficientizând procesul educațional.

2. Preliminarii

1. Metode actuale de conectare între elevi si meditatori

În prezent, există mai multe metode prin care elevii își găsesc meditatori pentru pregătirea examenelor, și prin care meditatorii își găsesc elevi, fiecare având dezavantaje și avantaje. Aceste metode includ recomandările personale, utilizarea grupurilor de Facebook și platformele de anunțuri și aplicațiile de comunicare. În continuare, vom analiza pe rând aceste modalități, evidențiind avantajele și dezavantajele fiecăreia.

1. **Recomandările personale**

Recomandările reprezintă una dintre cele mai frecvente și de încredere metode prin care elevii își găsesc meditatori. Acestea provin, de obicei, din cercurile apropiate – prieteni, familie, colegi de școală. Avantajul principal este încrederea, deoarece elevii se pot baza pe opiniile celor cunoscuți. Ca dezavantaje, avem limitarea opțiunilor la ceea ce oferă cercul personal și lipsa unui sistem organizat de căutare.

2. **Grupurile de Facebook**

Acestea reprezintă o modalitate populară pentru elevi de a găsi meditatori. Avantajul major este varietatea ofertelor disponibile. Dezavantajele includ dificultatea de a filtra opțiunile relevante și siguranța limitată în privința calității serviciilor oferite. De asemenea, comunicarea și distribuirea materialelor se realizează pe aplicații diferite.

3. **Platformele de anunțuri (ex. OLX, Meditatii.ro)**

Platformele de anunțuri precum OLX sau specializate precum Meditații.ro permit meditatorilor să își promoveze serviciile și elevilor să le caute într-un mod mai centralizat. Acestea oferă adesea filtre pentru a ajuta la selectarea meditatorilor în funcție de materie sau nivelul de învățământ. Dezavantajele ar fi faptul că multe nu dispun de funcționalități integrate de comunicare directă sau de distribuire de materiale, ceea ce impune folosirea unor aplicații suplimentare.

4. **Aplicații de comunicare (ex. WhatsApp, Teams)**

Pentru comunicare și schimb de materiale, elevii și meditatorii folosesc de obicei aplicații de mesagerie ca WhatsApp și Teams. Acestea sunt eficiente pentru convorbiri și trimiterea de documente, însă nu oferă funcționalități de căutare sau potrivire a meditatorilor cu elevii.

2. Comparație cu alte aplicații

În acest capitol este realizată o comparație între aplicațiile existente pe piață și aplicația dezvoltată, evidențiind diferențele semnificative în ceea ce privește funcționalitatea și experiența utilizatorului.

1. Preply este una dintre cele mai populare platforme internaționale, nișată pe învățarea limbilor străine. Are un proces de onboarding asemănător cu aplicația dezvoltată, un sistem de rating, profiluri, filtrare și sortare, mesagerie și notificări în timp real. Totuși, nu include un sistem de încărcare și gestionare al materialelor educaționale.

2. Superprof este o platformă de anunțuri pentru meditatori. Dispune de un sistem de rating, profiluri și mesagerie. Diferențele sunt lipsa unui proces de onboarding și, implicit, de sortare personalizată a meditatorilor. De asemenea, nu oferă funcționalități legate de încărcarea de materiale sau notificări.

3. Meditatii.ro este o platformă românească de anunțuri pentru meditații. Aplicația prezintă dashboard-uri separate pentru studenți și meditatori, sistem de mesagerie. O particularitate este că permite și meditatorilor să caute studenți. Aplicația nu prezintă niciun sistem de încărcare de materiale sau un proces de onboarding.

4. Cambly este o platformă axată pe învățarea limbii engleze prin conversații video cu vorbitori nativi. Oferă funcționalități de evaluare a meditatorilor, mesagerie în timp real și profiluri personalizate. Nu oferă un proces de onboarding personalizat sau încărcarea și organizarea de materiale educaționale.

Aplicația dezvoltată se diferențiază prin funcționalitatea de încărcare și gestionare a materialelor educaționale, controlul accesului la conținut, procesul personalizat de onboarding și logica de potrivire bazată pe scoruri de relevanță.

3. Tehnologii folosite

În acest capitol voi prezenta tehnologiile utilizate, împărțite în 3 categorii: partea de backend (logica aplicației și baza de date), frontend (interfața cu utilizatorul) și instrumente auxiliare.

1. Backend

1. Laravel

Laravel este un framework PHP modern, open-source, care pune accent pe simplitatea și rapiditatea dezvoltării aplicațiilor web. Acesta oferă un ecosistem vast de funcționalități integrate, precum rutare, autentificare, validare, cozi de joburi, middleware și protecție împotriva atacurilor CSRF.

Pentru comunicarea cu baza de date, Laravel utilizează Eloquent ORM (Object-Relational Mapping), care permite lucrul cu datele într-un mod intuitiv, bazat pe obiecte. În plus, Eloquent contribuie la securitatea aplicației prin protecție împotriva atacurilor de tip SQL injection, datorită utilizării interogărilor pregătite (prepared statements).

De asemenea, Laravel facilitează implementarea comunicării în timp real prin suportul nativ pentru event broadcasting, folosit pentru implementarea notificărilor în timp real și a sistemului de mesagerie.

2. Livewire

Livewire este un framework fullstack pentru Laravel, care permite construirea de componente interactive fără a fi necesară scrierea de cod JavaScript. Practic, face legătura dintre backend și frontend într-un mod transparent, folosind AJAX pentru a actualiza automat porțiuni din pagină atunci când utilizatorul interacționează cu componentele.

Avantajul principal este simplitatea. Dezvoltatorul poate crea funcționalități dinamice (liste filtrabile etc.) scriind doar cod PHP, fără a fi nevoie să implementeze logica în JavaScript. Ne putem gândi astfel: într-o componentă Livewire, dacă o proprietate publică se schimbă pe partea de backend, această schimbare se reflectă automat și în interfața frontend.

3. AJAX

AJAX (asynchronous JavaScript and XML) este o tehnică web care permite încărcarea asincronă a datelor de pe server fără a reîncărca pagina. Astfel, paginile

devin mai rapide și interactive.

4. **Bunny.net**

Bunny.net este o rețea de livrare a conținutului (CDN – Content Delivery Network) rapidă și accesibilă. În acest proiect este folosit pentru Bunny.net Stream, o platformă de stocare și livrare video.

5. **Pusher**

Pusher este un serviciu care permite aplicațiilor să transmită și să primească evenimente în timp real prin WebSockets, facilitând construirea de funcționalități interactive precum notificări.

6. **MySQL**

Pentru gestionarea datelor aplicației, folosim MySQL, unul dintre cele mai populare sisteme de gestionare a bazelor de date relaționale. Am ales o bază de date relațională deoarece structura datelor reflectă în mod natural relațiile dintre entități, precum cea dintre elevi și meditatori, sau dintre meditatori și materiale etc.

2. Frontend

1. **Blade**

Blade este motorul de templating implicit în Laravel, care permite crearea de interfețe frontend dinamice și ușor de întreținut. Acesta permite reutilizarea componentelor și optimizarea codului HTML prin utilizarea structurilor condiționale și a buclelor.

2. **TailwindCSS**

TailwindCSS este un framework CSS care permite stilizarea rapidă a interfețelor web prin clase predefinite, evitând scrierea de CSS personalizat pentru fiecare element. Un dezavantaj al acestui framework este faptul că utilizarea extinsă a claselor utilitare poate duce la un cod HTML mai lung și mai greu de citit, dar care poate fi gestionat prin folosirea claselor personalizate.

3. **Plyr**

Plyr este un player media simplu, accesibil și personalizabil pentru video și audio HTML5, precum și conținut YouTube.

4. **Swiper.js**

Swiper.js este o bibliotecă JavaScript pentru crearea de carusele touch-friendly.

5. **Laravel Echo**

Laravel Echo este o bibliotecă JavaScript care facilitează abonarea la canale și

ascultarea evenimentelor transmise în timp real de Laravel prin WebSockets, folosind tehnologii precum Pusher.

3. Tehnologii auxiliare

1. **Vite**

Vite este un bundler modern și rapid utilizat pentru dezvoltarea aplicațiilor frontend. Acesta oferă un timp de pornire aproape instant și reîncărcare automată a paginii la fiecare modificare a codului, folosind Hot Module Replacement (HMR), ceea ce accelerează semnificativ procesul de dezvoltare. În plus, Vite a fost folosit pentru a compila și livra eficient resursele frontend.

2. **Git**

Pentru versionarea codului sursă am folosit Git, un sistem de control al versiunilor extrem de popular în dezvoltarea de software. Git permite urmărirea modificărilor aduse codului de-a lungul timpului și revenirea la versiuni anterioare când este necesar.

3. **Visual Studio Code**

Visual Studio Code este un editor de cod gratuit dezvoltat de Microsoft.

4. Arhitectura aplicației și implementarea

4.1. Arhitectura generală

Aplicația este construită pe o arhitectură bazată pe componente (component-based architecture – CBA). Aceasta permite separarea logicii aplicației în componente independente, fiecare responsabilă pentru o anumită funcționalitate (de exemplu, componenta pentru sistemul de notificări, componenta pentru sistemul de mesagerie).

4.1.1. Arhitectura MVC (Model-View-Controller)

Modelul MVC este una dintre cele mai răspândite arhitecturi în dezvoltarea aplicațiilor web. Așa cum se poate observa în Figura 1, arhitectura MVC separă clar logica aplicației în 3 componente:

- **Modelul** – nucleul aplicației este responsabil cu gestionarea datelor. Modelul accesează și manipulează datele, de regulă prin interacțiunea cu baza de date, și răspunde cererilor venite din partea controller-ului.
- **Controller-ul** – actionează ca un intermediar între Model și View. El primește cererile din partea utilizatorului, procesează logica necesară, de obicei apelând unul sau mai multe modele, și transmite rezultatele către view-ul corespunzător.
- **View-ul** – se ocupă cu prezentarea informației către utilizator. Nu conține logică de procesare, ci doar de afișare.

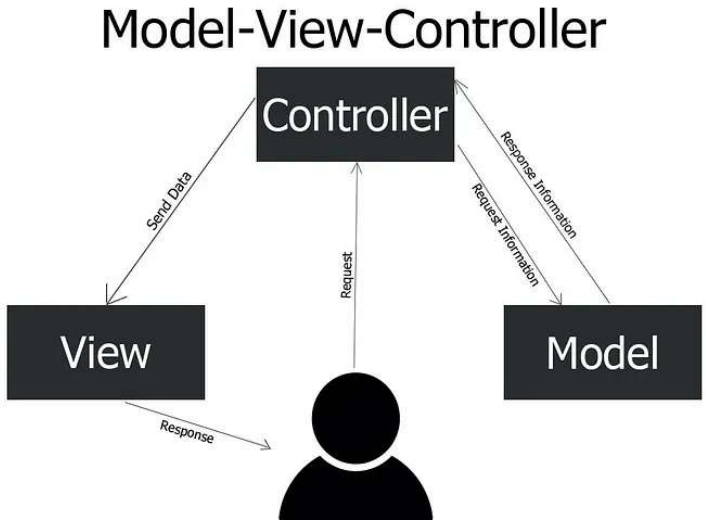

Figura 1 - Arhitectura MVC (Model-View-Controller) [9]

Printre avantajele arhitecturii MVC se numără:

- Separarea clară a responsabilităților, ceea ce duce la o structură de cod bine organizată și ușor de întreținut.
- Facilitează dezvoltarea colaborativă, deoarece diferite echipe pot lucra în paralel pe componente diferite ale aplicației.
- Testare mai ușoară, deoarece componentele aplicației pot fi testate individual.

Totuși, această arhitectură prezintă și anumite dezavantaje, cum ar fi:

- Complexitatea suplimentară în aplicații simple, unde separarea strictă a componentelor poate fi considerată excesivă.
- Volumul mai mare de cod necesar, întrucât chiar și funcționalitățile simple presupun crearea și conectarea celor trei componente.

Este important de menționat că Laravel este construit nativ pe arhitectura MVC.

4.1.2. Arhitectura aplicației dezvoltate

Aplicația este construită pe o arhitectură bazată pe componente (Component-Based Architecture – CBA), care poate fi privită ca o extensie a modelului clasic MVC. În această arhitectură, logica aplicației este împărțită în componente independente (vezi Figura 9), fiecare responsabilă pentru o funcționalitate specifică sau un grup de funcționalități conexe. De exemplu, există componente distincte pentru sistemul de notificări și pentru sistemul de mesagerie.

Fiecare componentă este structurată în două părți principale:

- Logica de procesare, implementată într-o clasă PHP dedicată, care gestionează starea componentei, procesează datele și răspunde evenimentelor generate de utilizator sau de aplicațietie.
- Partea de prezentare, realizată printr-un fișier Blade, care conține template-ul HTML și directivele specifice Laravel pentru afișarea dinamică a datelor și interacțiunea cu utilizatorul.

Această separare în cadrul componentei permite o organizare clară și modulară, în care logica de procesare și prezentarea sunt strâns legate, dar totuși distincte.

Avantaje ale arhitecturii alese:

- Componentele pot fi dezvoltate și testate independent, ceea ce asigură o modularitate ridicată.

- Prin combinarea logicii și prezentării în componente autonome, aplicația poate oferi interactivitate și reactivitate fără nevoia de a folosi JavaScript explicit, deoarece actualizările dinamice sunt gestionate automat pe server și sincronizate cu interfața grafică.

Totuși, arhitectura prezintă și unele limitări:

- Fiindcă reactivitatea este gestionată în întregime pe server, scalabilitatea poate deveni o provocare când numărul utilizatorilor și al interacțiunilor crește, ceea ce duce la o încărcare mai mare a serverului.
- Deoarece fiecare interacțiune a utilizatorului implică comunicarea cu serverul pentru procesare și actualizare, performanța aplicației poate fi influențată de viteza conexiunii la internet, ceea ce poate afecta experiența utilizatorului.

În proiectarea și implementarea aplicației au fost luate în calcul aceste limitări, iar mecanismele de actualizare dinamică a interfeței au fost aplicate doar în acele zone în care au contribuit la îmbunătățirea experienței utilizatorului, cu scopul de a optimiza performanța.

4.2. Arhitectura bazei de date

Baza de date reprezintă componenta fundamentală a aplicației, având rolul de a stoca și organiza informațiile necesare funcționării platformei. Prin baza de date gestionăm datele despre utilizatori, materiale didactice, mesaje, notificări și alte elemente esențiale pentru desfășurarea activităților în cadrul platformei. Structura bazei de date este proiectată astfel încât să asigure scalabilitatea sistemului.

Baza de date este relațională și este organizată în mai multe tabele conectate prin relații de tip cheie primară – cheie externă.

4.2.1. Structura bazei de date

Acest subcapitol prezintă tabelele principale ale bazei de date. Fiecare tabel este conceput pentru a răspunde unei funcționalități specifice ale aplicației:

1. **Users**: Reprezintă utilizatorii platformei. Sunt identificați prin adrese de email și pot avea roluri diferite (student, meditator, administrator).

2. **UserProfiles**: Conține informații adiționale despre utilizatori, separate de datele de autentificare din tabela Users. Include date precum numărul de telefon, calea către poza de profil și o descriere pe scurt, afișate pe pagina de profil.

3. **PossibleAnswers**: Reprezintă variantele de răspuns posibile la întrebările

adresate utilizatorilor în etapa de creare a contului. Fiecărui răspuns îi este asociat un număr de întrebare, permițând gruparea logică.

4. **Answers**: Reprezintă răspunsurile selectate de utilizatori în timpul onboarding-ului, fiind asociate atât cu tabela Users, cât și cu răspunsul ales din tabela PossibleAnswers.

5. **TutorsStudents**: Reprezintă relația de tip *many-to-many* dintre meditatori și studenți. Fiecare înregistrare din acest tabel face legătura între un utilizator cu rol de meditator și unul cu rol de student. Tabelul conține două chei externe către tabela Users, ambele referindu-se la ID-uri de utilizatori, dar cu roluri diferite.

6. **Content**: Reprezintă conținutul educațional adăugat de fiecare meditator pe platformă. Fiecare material este asociat cu un utilizator din tabela Users (autorul conținutului) și este încadrat într-o categorie definită în tabela Categories.

7. **Categories**: Reprezintă categoriile definite de meditatori pentru organizarea conținutului. Acestea pot varia de la concepte generale (Integrale, Ecuații diferențiale) până la grupe specifice (Informatică clasa a 10-a). Fiecare categorie este creată de un meditator și este folosită pentru clasificarea materialelor din tabela Content. De asemenea, meditatorii pot acorda acces selectiv la anumite categorii studenților lor, facilitând astfel o distribuție personalizată a resurselor educaționale.

8. **Conversations**: Reprezintă canalele de comunicare între mai mulți utilizatori, fie în relații unu-la-unu între student și meditator, fie între grupuri care includ un meditator și mai mulți studenți. Fiecare conversație are un identificator unic și este asociată cu participanții implicați.

9. **Messages**: Stochează mesajele trimise între utilizatori în cadrul conversațiilor definite în tabela Conversations. Fiecare mesaj este asociat cu un utilizator (expeditorul), o conversație, conține textul mesajului și informații precum momentul trimiterii.

10. **TutorRatings**: Conține evaluările oferite de studenți meditatorilor. Fiecare evaluare este asociată cu un utilizator cu rol de student și unul cu rol de meditator, incluzând un scor de la 1 la 5 stele și opțional un comentariu. Acest tabel permite monitorizarea calității serviciilor oferite de meditatori pe platformă.

4.2.2. Diagrama bazei de date

Pentru o înțelegere mai clară a modului în care sunt structurate și interconectate tabelele bazei de date, mai jos este prezentată diagrama entitate – relație (ERD – Entity Relationship Diagram). Aceasta oferă o reprezentare vizuală a entităților din sistem, a

câmpurilor cheie și a relațiilor dintre ele.

Diagrama reflectă organizarea logică a datelor în cadrul aplicației și evidențiază legăturile dintre entități, cum ar fi cele între utilizatori și profilurile lor, între meditatori și studenți, sau între conținutul educațional și categoriile aferente.

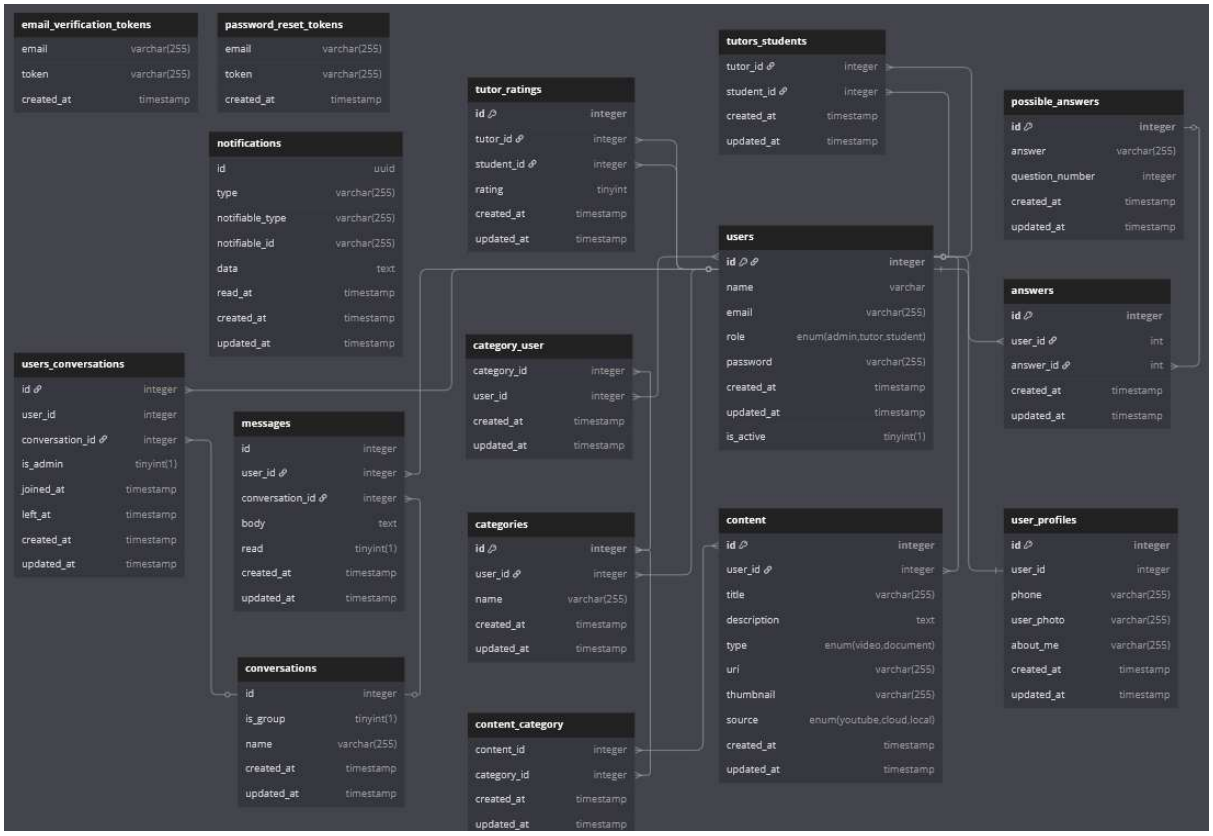

Figura 2- Diagrama entitate-relatie

5. Implementarea aplicației

Implementarea aplicației a fost realizată pe baza arhitecturii bazate pe componente, care poate fi văzută ca o extensie a modelului MVC tradițional. În această abordare, logica este împărțită în module independente, fiecare responsabil pentru o funcționalitate specifică. Pentru crearea componentelor reactive, care gestionează atât interfața, cât și logica aferentă, a fost folosit Livewire. O componentă Livewire este o clasă PHP care moștenește clasa Component din namespace-ul Livewire.

Împărțirea pe componente a permis separarea logicii aplicației în unități mai mici, reutilizabile, fiecare cu propriul ciclu de viață.

5.1. Structura directoarelor

Laravel oferă o structură a directoarelor clară și bine definită la crearea proiectului, încurajând organizarea eficientă a codului. Structura principalelor directoare este următoarea:

- app/Models/ include modelele Eloquent ce reprezintă entitățile principale ale aplicației, cum ar fi User, Category, Content sau Rating. Acestea definesc relațiile între entități.
- app/Http/Livewire conține componentele Livewire. Acestea sunt grupate în subdirectoare tematice (Dashboards, Auth) pentru o mai bună organizare.
- resources/views stochează fișierele Blade, responsabile pentru partea vizuală a aplicației. Fiecare componentă Livewire are un view asociat, care este așezat în subdirectorul resources/views/livewire.
- app/Notifications include notificările personalizate, cum ar fi MatchRequestNotification.
- app/Services este folosit pentru a organiza logica de business complexă, care nu își are locul direct într-o componentă sau într-un model. De exemplu, RatingService gestionează procesul de calcul și actualizare al evaluărilor.
- app/Events conține evenimentele din sistem, cum ar fi MessageSent, folosit la funcționalitatea de mesagerie.
- routes/ conține rutele aplicației, având 4 fișiere: web.php definește rutele web ale aplicației, api.php conține rutele API-ului, console.php este folosit pentru definirea de comenzi artisan din linia de comandă, iar channels.php definește canalele de broadcasting pentru Pusher.

• public/ este folderul accesibil extern, destinat fișierelor statice, cum ar fi imaginile de profil.

• config/ conține fișierele de configurare pentru diferite servicii, de exemplu în fișierul broadcasting.php este definit serviciul Pusher.

Această structură clară susține dezvoltarea pe termen lung, facilitând organizarea codului.

5.2. Pagina Welcome si sistemul de autentificare

Pagina Welcome reprezintă primul punct de contact al utilizatorului cu aplicația, oferind atât opțiunea de înregistrare, cât și posibilitatea de autentificare pentru utilizatorii existenți. Aceasta oferă acces rapid și intuitiv la funcționalitățile principale legate de autentificare și înregistrare:

- **Formularul de înregistrare** este afișat direct pe pagină, permițând utilizatorilor să creeze un cont nou prin completarea câmpurilor necesare (nume, email, parolă, confirmare parolă).
- **Butonul „Login”** deschide o fereastră modală care conține formularul de autentificare.
- **Link-ul „Forgot your password?”** oferă acces la mecanismul de resetare a parolei.

5.2.1. Sistemul de înregistrare

Procesul de înregistrare presupune atenție sporită la securitate și validarea datelor. Toate datele sunt verificate pe server, pentru a preveni orice tentativă de introducere a datelor incorecte sau malițioase. În figura 3 avem variabila $validated, care este utilizată pentru a valida datele, și mai apoi folosită pentru introducerea acestora în baza de date.

```
$validated = $this->validate(rules: [
    'name' => 'required|string|max:30',
    'email' => 'required|email|unique:users,email',
    'password' => 'required|string|min:8|confirmed',
    'password_confirmation' => 'required|string|min:8',
]);
```

Figura 3 – validarea datelor

În partea de model a aplicației, parolele sunt apoi protejate prin hash-uire cu algoritmul bcrypt înainte de a fi stocate în baza de date, asigurând astfel un nivel ridicat

de securitate în cazul unui eventual acces neautorizat la date. În plus, formularele sunt protejate prin token-uri CSRF pentru a evita eventualele atacuri de tip cross-site request forgery. După validarea și procesarea datelor, contul nou este salvat în baza de date, apoi utilizatorului i se creează automat un profil gol și introdus în baza de date. Figura 4 reprezintă funcția de creare a utilizatorului din cadrul modelului.

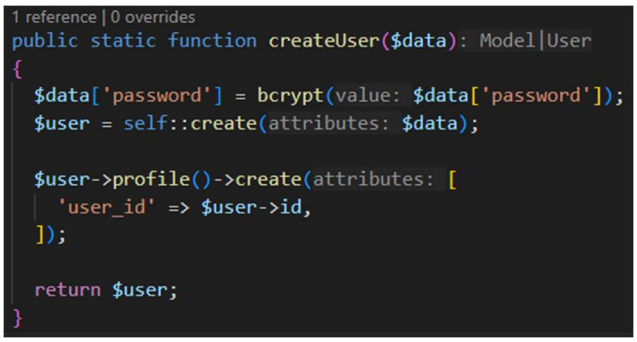

Figura 4 – crearea utilizatorului

Înregistrarea nu este considerată finalizată până la confirmarea adresei de email. După trimiterea formularului, un token unic este generat și inserat în baza de date, în tabela email_verification_tokens. Utilizatorului i se trimite automat un email cu un link de forma **/confirm/{token}**, care după ce este accesat validează adresa de email și activează contul, permițând astfel autentificarea.

5.2.2. Sistemul de autentificare

După confirmarea adresei de email, utilizatorii pot accesa aplicația prin sistemul de autentificare disponibil prin fereastra modală care se deschide la apăsarea butonului **Login**. Procesul de autentificare verifică dacă utilizatorul există, dacă parola introdusă este corectă și dacă adresa de email a fost validată.

Parola introdusă de utilizator este comparată cu hash-ul stocat în baza de date, folosind API-ul Auth::attempt() oferit de Laravel. În cazul unei autentificări reușite, este inițiată o sesiune de utilizator și este generat un token CSRF pentru securizarea interacțiunilor ulterioare cu aplicația.

5.2.3. Resetarea parolei

Funcționalitatea de resetare a parolei este accesibilă prin link-ul „Forgot your password?” din cadrul formularului de înregistrare. La accesarea link-ului, o fereastră modală este deschisă cu un singur câmp, adresa de email a utilizatorului.

Dacă adresa introdusă este validă și corespunde unui cont existent, aplicația generează un token unic care este salvat în baza de date și trimis pe email sub forma unui link de resetare **/reset-password/{token}**. Acest link deschide o pagină unde utilizatorul

își poate introduce o nouă parolă.

Noua parolă este validată, apoi hash-uită și salvată în baza de date, înlocuind parola veche. După resetare, utilizatorul este trimis către pagina de Welcome, pentru autentificare.

5.2. Onboarding – pagina Answer Questions

După autentificare, utilizatorul este trimis către pagina de onboarding, denumită **Answer Questions**. Această etapă este obligatorie și are rolul de a colecta informații esențiale, afișate pe profilul utilizatorului. Răspunsurile oferite la acest pas sunt folosite la calcularea scorului de relevanță, care influențează ordinea în care meditatorii apar în lista disponibilă pentru studenți pe pagina **Home**. Scorul reflectă gradul de potrivire dintre preferințele studentului și profilurile meditatorilor.

Accesul la celelalte secțiuni ale aplicației este restricționat până la completarea onboarding-ului. Acest lucru este realizat prin intermediul unui middleware care verifică dacă utilizatorul are toate întrebările completate. În caz contrar, orice încercare de a accesa alte rute va redirecționa utilizatorul înapoi la pagina de **Answer Questions**.

Răspunsurile la întrebări sunt procesate și salvate în baza de date imediat ce utilizatorul apasă butonul de salvare. Acestea sunt asociate cu utilizatorul și pot fi modificate ulterior de pe pagina de profil.

5.3. Pagina Home

După finalizarea procesului de onboarding, utilizatorul este redirecționat către pagina principală a aplicației, **Home**. Această pagină reprezintă primul punct de interacțiune între utilizatori și afișează o listă de profiluri (vezi Figura 8), în funcție de rolul utilizatorului autentificat (meditatori pentru studenți și studenți pentru meditatori). Funcționalitățile acestei pagini includ filtrarea, sortarea, accesul la profiluri și trimiterea cererilor de potrivire.

5.3.1. Filtrarea și sortarea

Utilizatorii pot filtra lista afișată după nume sau după oricare dintre răspunsurile oferite în procesul de onboarding, ceea ce permite o căutare specifică și relevantă. În plus, lista poate fi sortată după următoarele criterii:

- **Relevance** – scor calculat pe baza răspunsurilor din onboarding și, pentru lista de meditatori, și în funcție de recenziile lor.
- **Rating** – media evaluarilor primite de fiecare meditator.

- **Rating count** – pentru a favoriza meditatorii cu mai multe evaluări.
- **Newest** – utilizatorii adăugați recent, pentru a oferi vizibilitate profilurilor noi.

Filtrarea și sortarea listei de utilizatori se realizează dinamic, fără reîncărcarea paginii, cu ajutorul Livewire. La încărcarea inițială a paginii, Livewire atașează ascultători de evenimente pentru fiecare element relevant (ex. butonul „Filter”), care ascultă modificările utilizatorului. Livewire facilitează legarea bidirecțională a proprietăților componentei cu elementele de formular, folosind directive precum **wire:model**.

În acest caz, câmpurile de filtrare folosesc **wire:model.defer**, ceea ce înseamnă că modificările introduse de utilizator nu sunt trimise imediat către server, ca în cazul folosirii wire:model, ci sunt amânate până la apăsarea butonului Filter. În schimb, câmpul pentru sortare folosește **wire:model**, astfel încât orice modificare a valorii este trimisă instantaneu către server, iar lista este actualizată imediat.

Astfel, filtrarea are loc doar la acțiunea explicită a utilizatorului, în timp ce sortarea este aplicată instant. La momentul actualizării, serverul procesează criteriile și returnează un set de date filtrat și sortat.

Această abordare contribuie la optimizarea traficului de rețea și a performanței, reducând numărul de cereri către server și evitând actualizările inutile, păstrând în același timp o experiență de utilizare fluentă.

5.3.2. Scorul de relevanță

Scorul de relevanță este utilizat pentru a ordona rezultatele afișate în funcție de gradul de potrivire între student și meditator. Acest scor este calculat pe baza mai multor factori:

- Compatibilitatea între cerințele studentului și ofertele meditatorului.
- În cazul listei de meditatori, ratingul calculat pe baza recenziilor primite de la studenți.

Ratingul contribuie semnificativ și este calculat folosind o combinație între **media bayesiană** [9] și un factor de **exponential decay** [10] (descreștere exponențială), pentru a reflecta atât calitatea generală a serviciilor oferite până în prezent, cât și actualitatea acestora.

Media bayesiană este utilizată pentru a evita supraevaluarea utilizatorilor cu un număr redus de recenzii. De exemplu, un utilizator cu 3 recenzii de 5 stele nu ar trebui să fie poziționat mai sus decât un utilizator cu 300 de recenzii și media 4,7. Media

bayesiană combină media recenziilor unui utilizator cu media globală a tuturor recenziilor din platformă, în funcție de numărul de evaluări primite. Formula generală folosită este:

$$scor = \frac{v}{v+m} * R + \frac{m}{v+m} * C$$

unde:

- R este media recenziilor pentru utilizatorul respectiv
- V este numărul recenziilor acelui utilizator
- M este pragul minim de recenzii pentru a considera un rating stabil (ales ca 10)
- C este media generală a tuturor recenziilor din sistem (rating global)

Valoarea lui C este calculată periodic, odată la 3 ore, si este stocată in cache, pentru a evita recalcularea la fiecare sortare după scorul de relevanță.

Pentru a reflecta cât mai fidel performanța recentă, fiecărei recenzii i se aplică un factor de exponential decay, astfel încât cele vechi au o influență redusă. Formula utilizată este:

$$scor = scor * e^{-\lambda t}$$

unde:

- t = timpul parcurs în luni de la data recenziei
- λ = rata de decay

Deci, formula finală este:

$$scor\ final = \frac{\sum_{i=1}^{N} \text{rating}_i * e^{-\lambda t_i}}{\sum_{i=1}^{N} e^{-\lambda t_i}}$$

Factorul λ a fost ales ca 0.1, valoare pentru care în aproximativ 7 luni valoarea unei recenzii este înjumătățită. Prin această abordare, scorul rezultat este mai stabil și mai realist, motivând meditatorii să ofere constant servicii de calitate.

5.3.3. Trimiterea cererilor de potrivire si notificările în timp real

Prin intermediul butonului **Send request**, utilizatorii pot trimite cereri de potrivire către alți utilizatori. La apăsarea butonului, se deschide un modal care afișează un profil prescurtat al destinatarului, iar la confirmare, cererea este trimisă către server și salvată în baza de date.

Cererile sunt înregistrate în tabela notifications, care are o structură polimorfică

pentru a putea asocia notificările cu mai multe tipuri de modele. Coloana **data** din tabela stochează detaliile notificării în format JSON (JavaScript Object Notation), oferind flexibilitate în conținutul acesteia.

După salvarea notificării, este emis un eveniment de tip **broadcast** care conține datele notificării în format JSON și specifică canalul privat destinat utilizatorului țintă, identificat prin ID-ul său. Acest eveniment este trimis către serviciul Pusher, care acționează ca un WebSocket.

Un WebSocket este un protocol care permite o comunicare bidirecțională și permanentă între client și server. Aceasta înseamnă că atât serverul, cât și clientul pot trimite mesaje în timp real, fără a fi nevoie de reîncărcarea paginii sau realizarea unor cereri HTTP repetate (polling).

Pusher primește cererea HTTP de la aplicația Laravel, care conține payload-ul JSON al notificării și identificatorul canalului, în acest caz **private-Models.App.User.{idDestinatar}**. Pusher gestionează apoi conexiunile WebSocket deschise de clienții frontend și transmite notificarea în timp real către toți abonații acelui canal.

Pe partea de client, aplicația folosește Laravel Echo, o bibliotecă JavaScript care facilitează conectarea la canale WebSocket și ascultarea evenimentelor. Laravel Echo stabilește conexiunea Pusher și ascultă pentru evenimente pe canalul privat al utilizatorului curent.

Când Pusher transmite notificarea, Laravel Echo o recepționează și emite un semnal intern către componenta Livewire. Aceasta ascultă semnalul și, ca răspuns, apelează metoda **updateNotifications()**, care actualizează lista de notificări afișate, actualizând astfel interfața utilizatorului în timp real. Procesul este asemănător pentru sistemul de mesagerie (vezi Figura 4).

Cererea poate fi acceptată printr-un click pe notificare (vezi Figura 5), care deschide un modal cu un formular de confirmare. Prin apăsarea butonului **Accept**, se realizează asocierea între utilizatori.

5.4. Pagina de profil

Pagina de profil (vezi Figura 7) reprezintă spațiul dedicat gestionării și vizualizării informațiilor personale ale utilizatorilor, precum și explorării profilurilor altor membri ai platformei. Structura acestei pagini este adaptivă în funcție de utilizatorul curent: dacă pagina este accesată de proprietarul contului, anumite secțiuni devin

editabile, iar în cazul unui utilizator cu rol de meditator, acesta are posibilitatea de a adăuga sau elimina categorii de conținut educațional.

Pagina este compusă din următoarele secțiuni principale:

- **Informatiile personale**: nume, poză de profil, email, număr de telefon.
- **Descriere personala**: un câmp în care utilizatorul își poate completa un profil descriptiv.
- **Raspunsurile din onboarding**: afișate într-o secțiune separată, acestea oferă vizitatorilor o imagine de ansamblu asupra preferințelor utilizatorului.
- **Formularul de evaluare**: disponibil exclusiv pe paginile de profil ale meditatorilor. Este reprezentat vizual de 5 stele. În cazul în care vizitatorul nu a evaluat meditatorul, sunt evidențiate stelele corespunzătoare mediei recenziilor utilizatorului. Studenții asociați meditatorului pot interacționa cu formularul, selectând un număr de stele. Sistemul permite o singură evaluare per asociere, care poate fi modificată ulterior.
- **Continutul educational**: materialele publicate de meditator, împărțite în categorii.
- **Lista de evaluari ale studentilor**: include toate evaluările primite de meditator, fiecare conținând numele evaluatorului, comentariul și scorul acordat.

Secțiunea de conținut afișează materialele încărcate de meditator, organizate în categorii definite de acesta. Conținutul este împărțit în două tipuri: videoclipuri și documente, ambele afișate în carusele interactive implementate cu ajutorul bibliotecii Swiper.js.

Videoclipurile sunt redate cu ajutorul playerului Plyr, care asigură o experiență consistentă, indiferent de sursa videoclipului. Documentele, pe de altă parte, sunt afișate ca linkuri descărcabile.

5.5. Dashboard-uri

5.5.1. Admin Dashboard

Dashboard-ul administratorilor oferă un set de funcționalități pentru gestionarea platformei.

Una dintre componente este secțiunea dedicată întrebărilor din procesul de onboarding. Administratorii au posibilitatea de a adăuga sau elimina opțiuni de răspuns, astfel adaptând răspunsurile posibile la nevoile reale ale utilizatorului.

O altă funcționalitate este tabelul cu toți utilizatorii înregistrați pe platformă. Din

această interfață, administratorii pot inspecta profilurile utilizatorilor și aplica acțiuni administrative ca blocarea contului de utilizator.

De asemenea, dashboard-ul conține un buton dedicat declanșării manuale a procesului de recalculare a mediei globale necesare necesară în formula de calcul a mediei bayesiene a evaluărilor meditatorilor, contribuind astfel la o evaluare mai echitabilă în cadrul scorului de relevanță.

5.5.2. Tutor Dashboard

Dashboard-ul meditatorilor reprezintă spațiul în care aceștia pot gestiona conținutul educațional și accesul studenților la materiale. Prin intermediul acestui dashboard, meditatorii au posibilitatea de a adăuga, modifica sau șterge videoclipuri și documente, de a ajusta drepturile de acces ale studenților la materiale, precum și de a încheia colaborarea cu aceștia.

Pentru adăugarea de materiale video, meditatorii folosesc un formular accesibil dintr-un modal care se deschide la apăsarea butonului Add video (vezi Figura 10). Formularul conține un câmp denumit Video Source, care permite selectarea sursei videoclipului: „File” sau „Youtube”. În funcție de opțiunea aleasă, formularul afișează câmpurile corespunzătoare, pentru „File”, un input de tip fișier, unde utilizatorul poate încărca un fișier video local, iar pentru „Youtube”, un input de tip text pentru introducerea URL-ului.

Fișierele video încărcate sunt stocate folosind CDN-ul Bunny.net, și livrate către utilizator prin Bunny Stream, cu un sistem de fallback care salvează fișierele local în caz de eșec, cauza principală fiind indisponibilitatea Bunny.net. Datele despre fișier cum ar fi numele, descrierea, URL-ul sunt salvate în tabela Content. Videoclipurile adăugate apar într-un tabel, unde pot fi editate ulterior titlul, descrierea, și categoriile atribuite.

Pentru documente, procesul este similar, diferența fiind că documentele sunt salvate local.

Această arhitectură oferă meditatorilor control complet asupra conținutului educațional și a accesului studenților la acesta, într-o interfață intuitivă și ușor de utilizat.

5.5.3. Student Dashboard

Dashboard-ul studentului are o structură simplă, conținând un singur tabel cu meditatorii asociați. Prin intermediul acestuia, studentul poate accesa direct profilurile meditatorilor sau poate întrerupe asocierea cu oricare dintre ei.

6. Concluzii si direcții viitoare

Lucrarea a urmărit proiectarea și implementarea unei aplicații web care facilitează interacțiunea dintre studenți și meditatori. Platforma a fost construită folosind Laravel și Livewire, tehnologii care au permis dezvoltarea unei interfețe reactive, cu actualizări în timp real și o experiență fluidă pentru utilizator.

Ca direcții viitoare de dezvoltare, se pot lua în considerare următoarele idei:

- Aplicația poate fi extinsă prin a oferi suport pentru lecții colective. În primul rând, pentru programarea lecțiilor ar putea fi folosit un calendar, în care meditatorul ar putea să marcheze zilele și intervalele orare în care este disponibil pentru lecții individuale, și zile și intervale orare în care organizează o lecție colectivă, iar numărul acestora să fie afișat și actualizat pe măsură ce se înscriu noi studenți. Meditatorul ar trebui să poată stabili un număr maxim de studenți pentru fiecare lecție, iar disponibilitatea pentru noi înscrieri să dispară din calendar în cazul în care numărul maxim de studenți este atins.
- O funcționalitate de plată online care să acompanieze acest calendar poate fi implementată folosind Stripe. Meditatorul ar trebui să poată specifica metoda de plată și suma pentru fiecare lecție sau să stabilească o sumă totală pentru intreaga săptămână, calculată în funcție de numărul de lecții stabilite pentru acea săptămână. O idee bună ar fi opțiunea de abonament, care să ofere studentului un număr de lecții săptămânal sau lunar.
- Funcționalitatea pentru conferințe video online în cadrul platformei nu este o prioritate, deoarece întâlnirile online se pot realiza folosind platforme precum Zoom sau Microsoft Teams.
- Un sistem de caching pentru a reduce timpii de încărcare. De exemplu, pentru lista de utilizatori de pe pagina Home, calculul scorului de relevanță se efectuează pentru fiecare utilizator în parte, ceea ce poate ridica timpii de încărcare pentru liste mari.
- Tabele de analytics pentru administratori si meditatori.

În concluzie, aplicația oferă o bază solidă atât pentru facilitarea interacțiunii dintre studenți și meditatori, cât și pentru extinderea viitoare a funcționalităților.

Bibliografie

[1] Laravel, „Laravel documentation,”. Disponibil la: https://laravel.com/docs/12.x. [Accesat Mai 2025].

[2] Laravel, „Blade Templating Engine Documentation,”. Disponibil la: https://laravel.com/docs/12.x/blade. [Accesat Mai 2025].

[3] Livewire, „Livewire Documentation,”. Disponibil la: https://laravel-livewire.com/docs. [Accesat 5 2025].

[4] Tailwind Labs, „Tailwind CSS Documentation,”. Disponibil la: https://tailwindcss.com/docs. [Accesat 5 2025].

[5] saadeghi, „daisyUI - Tailwind CSS Component Library,”. Disponibil la: https://daisyui.com. [Accesat 5 2025].

[6] Vite, „Vite Documentation,”. Disponibil la: https://vite.dev/guide/. [Accesat 5 2025].

[7] B. Lu, „Hot Module Replacement is Easy,”. Available: https://bjornlu.com/blog/hot-module-replacement-is-easy. [Accesat 5 2025].

[8] Institutul de Științe ale Educației, „Liceenii din România: Implicarea și autonomia liceenilor”, 12 2024. Disponibil la: https://www.ise.ro/wp-content/uploads/2024/12/Raport-4_Liceenii-din-Romania_Implicarea-si-autonomia-liceenilor.pdf. [Accesat Mai 2025].

[9] S. Bhujbal, „MVC Design Pattern”, 17 8 2023. Disponibil la: https://medium.com/@sandesh__30_/mvc-design-pattern-ff40d66990e3. [Accesat 6 2025].

[10] E. Miller, „Bayesian Average Ratings”, 6 11 2012. Disponibil la: https://www.evanmiller.org/bayesian-average-ratings.html. [Accesat 6 2025].

[11] G. Gundersen, „Exponential decay”, 17 5 2022. Disponibil la: https://gregorygundersen.com/blog/2022/05/17/exponential-decay/. [Accesat 6 2025].

Anexa

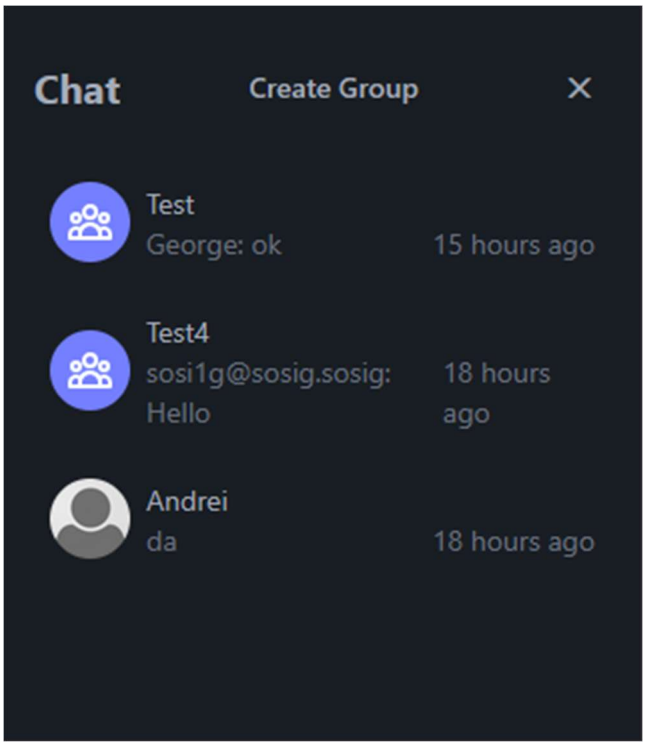

Figura 3 - Sistemul de mesagerie

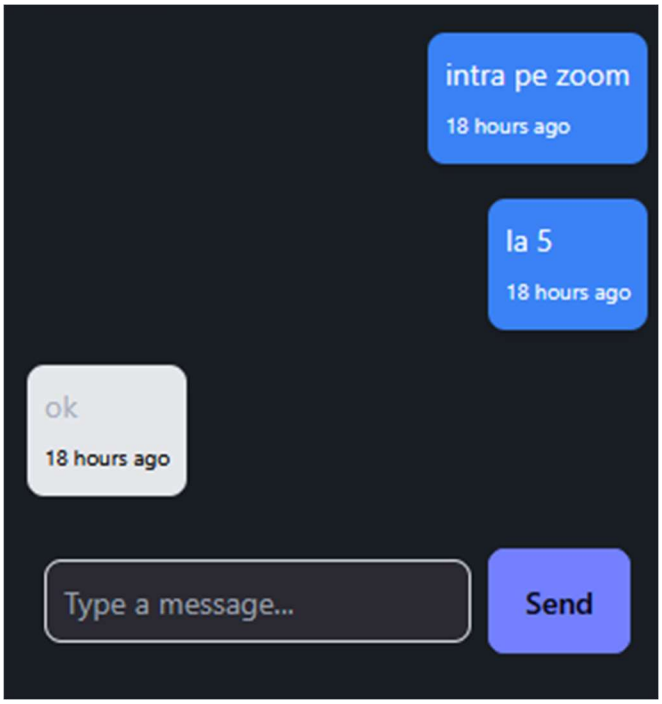

Figura 4 - Exemple de mesaje

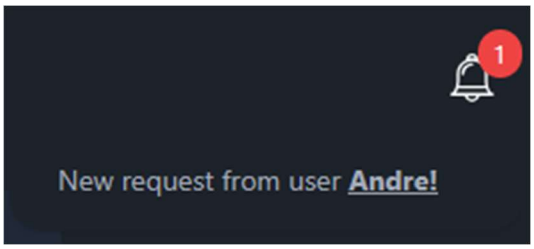

Figura 5 - Exemplu de notificare

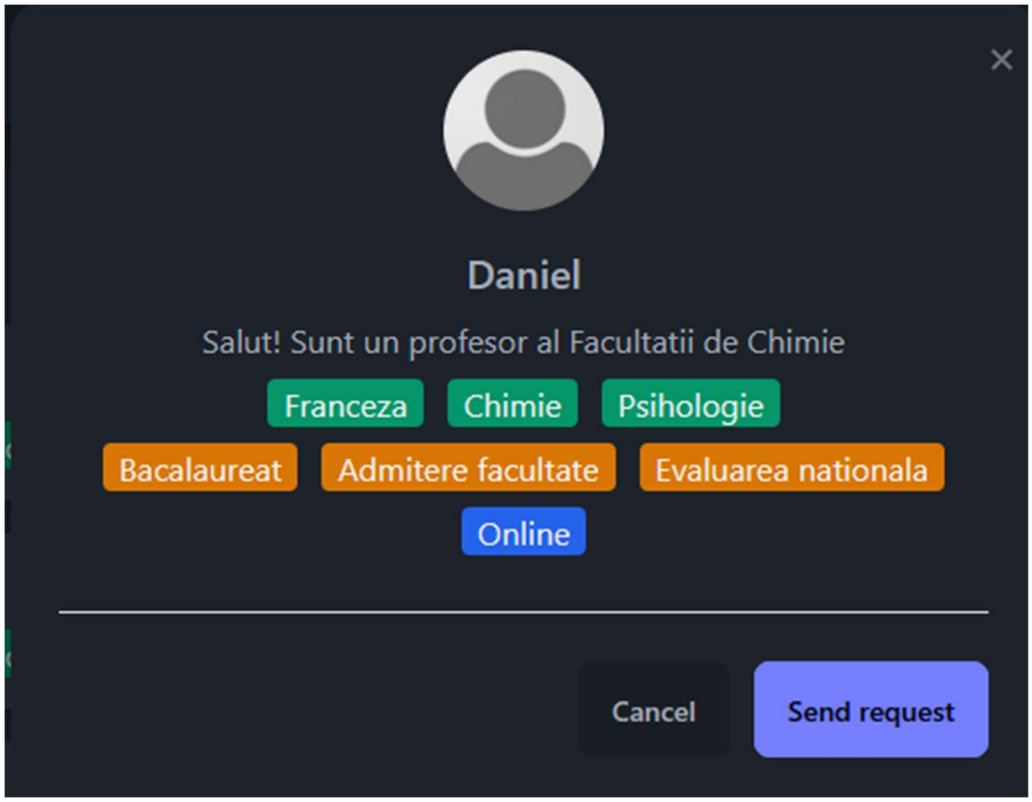

Figura 6 - Modalul pentru trimiterea cererii de potrivire

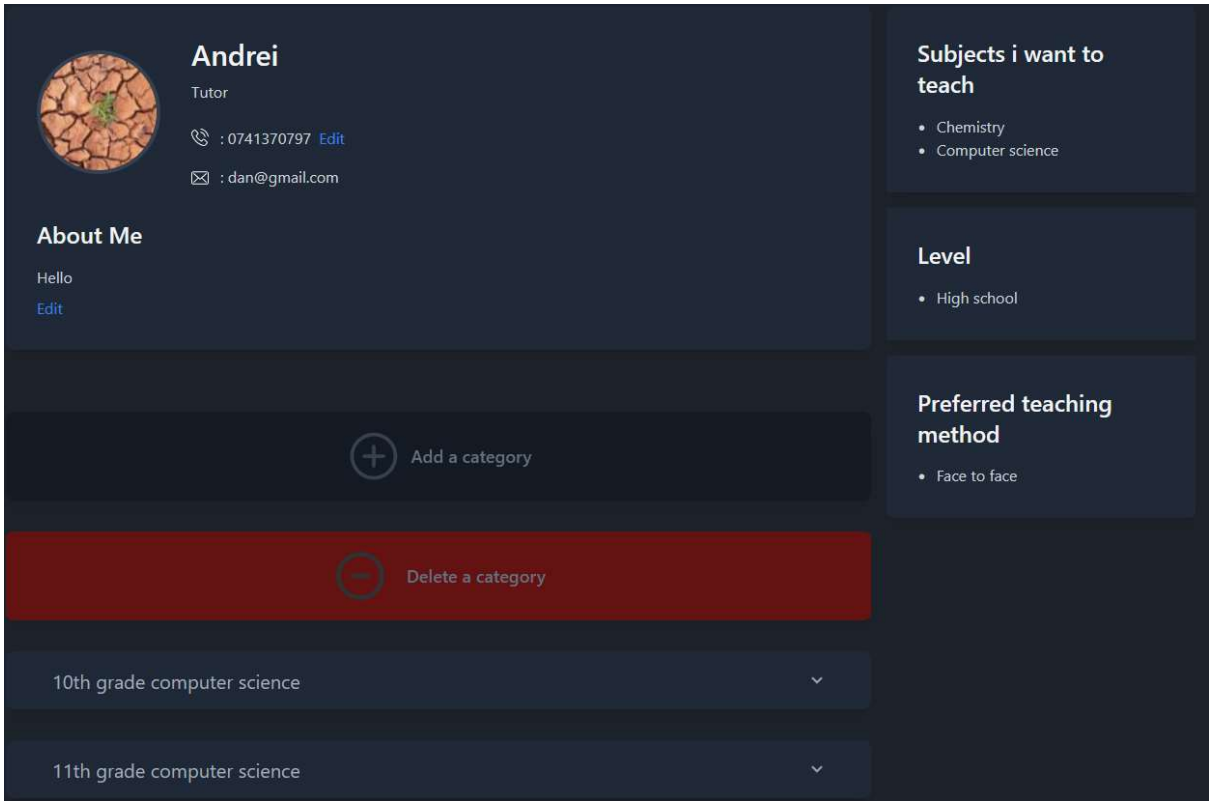

Figura 7 - Pagina de profil a unui meditator

Figura 8 - Lista de utilizatori

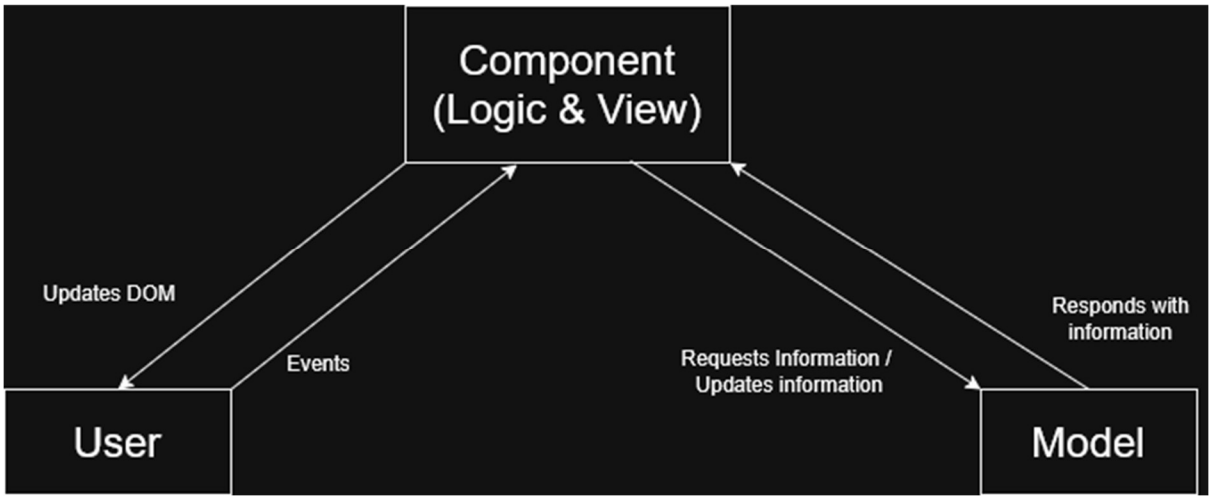

Figura 9 - Arhitectura aplicatiei

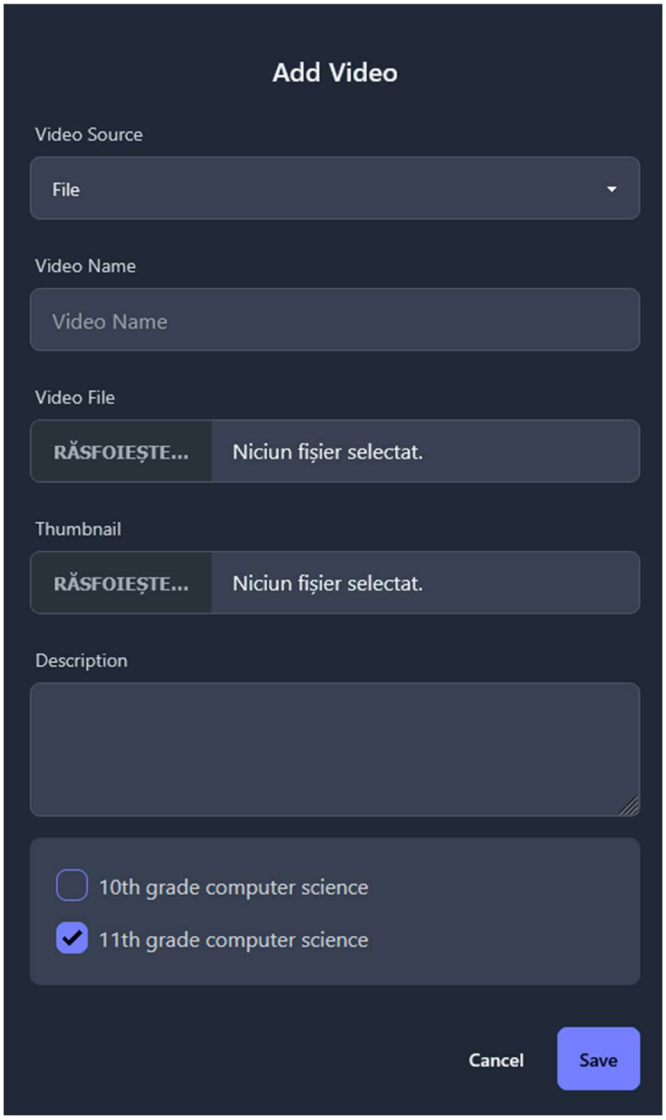

Figura 10 - Formularul pentru incărcarea de videoclipuri