### UNIVERSITATEA DIN BUCUREȘTI

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

**Lucrare de licență**

**Aplicație web pentru meditatii**

### 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]](#Laravel) pentru partea de backend, Blade[2] pentru frontend, iar Livewire[3] va fi utilizat pentru crearea de componente interactive reactive, oferind o experiență fluidă. Pentru stilizarea interfeței grafice, vom folosi TailwindCSS[4], împreună cu daisyUI, o bibliotecă de componente care faciliteaza realizarea interfetelor grafice pentru aplicatii web. Vom folosi și Vite, deoarece oferă atât Hot Module Replacement (HMR) 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 will be 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, we will use TailwindCSS[4] together with daisyUI[5], a component library that accelerates development. We will also use Vite[6], as it offers both Hot Module Replacement (HMR)[7] during development and an optimized build.

[1. Introducere 4](#_Toc199605417)

[2. Preliminarii 5](#_Toc199605418)

[Metode actuale de conectare intre elevi si meditatori 5](#_Toc199605419)

[3. Tehnologii folosite 7](#_Toc199605420)

[4. Arhitectura aplicatiei si implementarea 10](#_Toc199605421)

[Bibliografie 11](#_Toc199605422)

[Bibliografie 11](#_Toc199605423)

# Introducere

Conform unui raport publicat de Institutul de Stiinte ale Educatiei in 2024, „Liceenii din Romania: Implicarea si autonomia liceenilor”[8], 43% dintre elevii respondenti au declarat ca participa la meditatii particulare pentru bacalaureat, un procent semnificativ. Acest lucru evidentiaza o realitate bine cunoscuta in sistemul educational romanesc, necesitatea sprijinului educational suplimentar la pregatirea pentru examenele importante, cum ar fi bacalaureatul.

Intr-o lume ideala, sistemul de invatamant ar trebui sa asigure elevilor suficiente resurse si suport pentru a se pregati eficient in timpul orelor de curs. Insa, volumul mare de materie, presiunea unei note mari, impreuna cu, de multe ori, calitatea actului educational ii determina pe multi elevi sa apeleze la meditatii.

In prezent, majoritatea conexiunilor dintre elevi si meditatori se realizeaza prin recomandari, grupuri de Facebook si prin platforme precum OLX sau Meditatii.ro, iar distribuirea materialelor didactice si comunicarea se fac prin aplicatii separate, cum ar fi WhatsApp sau Teams. Elevii trebuie sa navigheze intre mai multe platforme pentru a gasi un meditator potrivit, pentru a primi si trimite materiale si pentru a comunica, ceea ce duce la o experienta mai putin coerenta.

In acest context, lucrarea propune o platforma unitara care sa faciliteze intregul proces de conectare si comunicare intre elevi si meditatori, precum si distribuirea materialelor, simplificand astfel interactiunea si eficientizand procesul educational.

# Preliminarii

## Metode actuale de conectare intre elevi si meditatori

In prezent, exista mai multe metode prin care elevii isi gasesc meditatori pentru pregatirea examenelor, si prin care meditatorii isi gasesc elevi, fiecare avand dezavantaje si avantaje specifice. Aceste metode includ recomandarile personale, utilizarea grupurilor de Facebook si platformele de anunturi si aplicatiile de comunicare. In continuare, vom analiza pe rand aceste modalitati, evidentiind avantajele si dezavantajele fiecareia.

1. **Recomandarile personale**

Recomandarile reprezinta una dintre cele mai frecvente si de incredere metode prin care elevii isi gasesc meditatori. Acestea provin, de obicei, din cercurile apropiate – prieteni, familie, colegi de scoala. Avantajul principal este increderea, deoarece elevii se pot baza pe opiniile celor cunoscuti. Ca dezavantaje, avem limitarea optiunilor la ceea ce ofera cercul personal si lipsa unui sistem organizat de cautare.

1. **Grupurile de Facebook**

Acestea reprezinta o modalitate populara pentru elevi de a gasi meditatori. Avantajul major este varietatea ofertelor disponibile. Dezavantajele includ dificultatea de a filtra optiunile relevante si siguranta limitata in privinta calitatii serviciilor oferite. De asemenea, comunicarea si distribuirea materialelor se realizeaza pe aplicatii diferite.

1. **Platformele de anunturi (ex. OLX, Meditatii.ro)**

Platformele de anunturi precum OLX sau specializate precum Meditatii.ro permit meditatorilor sa isi promoveze serviciile si elevilor sa le caute intr-un mod mai centralizat. Acesetea ofera adesea filtre pentru a ajuta la selectarea meditatorilor in functie de materie sau nivelul de invatamant. Dezavantajele ar fi faptul ca multe nu dispun de functionalitati integrate de comunicare directa sau de distribuire de materiale, ceea ce impune folosirea unor aplicatii suplimentare.

1. **Aplicatii de comunicare (ex. WhatsApp, Teams)**

Pentru comunicare si schimb de materiale, elevii si meditatorii folosesc de obicei aplicatii de mesagerie ca WhatsApp si Teams. Acestea sunt eficiente pentru convorbiri si trimiterea de documente, insa nu ofera functionalitati de cautare sau potrivire a meditatorilor cu elevii.

# 

# 3. Tehnologii folosite

Dezvoltarea unei platforme eficiente si intuitive pentru conectarea elevilor cu meditatorii incepe de la alegerea tehnologiilor. Acestea ar trebui sa eficientizeze procesul de dezvoltare, sa permita scalabilitate si intretinere usoara.

In acest capitol voi prezenta tehnologiile utilizate, impartite in 3 categorii: partea de backend (logica aplicatiei si baza de date), frontend (interfata cu utilizatorul) si instrumente auxiliare.

1. **Backend**
2. **Laravel**

Laravel este un framework PHP modern, open-source, care pune accent pe simplitatea si rapiditatea dezvoltarii aplicatiilor web. Acesta ofera un ecosistem vast de functionalitati integrate, precum rutare, autentificare, validare, cozi de joburi, middleware, si protectie impotriva atacurilor CSRF.

Pentru comunicarea cu baza de date, Laravel utilizeaza Eloquent ORM (Object-Relational Mapping), care permite lucrul cu datele intr-un mod intuitiv, bazat pe obiecte. In plus, Eloquent contribuie la securitatea aplicatiei prin protectie impotriva atacurilor de tip SQL injection, datorita utilizarii interogarilor pregatite.

De asemenea, Laravel faciliteaza implementarea comunicarii in timp real prin suportul nativ pentru event broadcasting, folosit pentru implementarea notificarilor in timp real si a sistemului de mesagerie.

1. **Livewire**

Livewire este un framework fullstack pentru Laravel, care permite construirea de componente interactive in interfata utilizatorului, fara a fi necesara scrierea de cod JavaScript. Practic, face legatura dintre backend si frontend intr-un mod transparent, folosind AJAX pentru a actualiza automat portiuni din pagina atunci cand utilizatorul interactioneaza cu componentele.

Avantajul principal este simplitatea. Dezvoltatorul poate crea functionalitati dinamice (liste filtrabile etc.) scriind doar cod PHP, fara a fi nevoie sa implementeze logica in JavaScript. Ne putem gandi astfel: intr-o componenta Livewire, daca o valoare se schimba pe partea de backend, aceasta schimbare se reflecta automat si in interfata frontend.

1. **AJAX**
2. **Bunny.net**
3. **Pusher**
4. **MySQL**

Pentru gestionarea datelor aplicatiei, folosim MySQL, unul dintre cele mai populare sisteme de gestionare a bazelor de date relationale. Am ales o baza de date relationala deoarece structura datelor reflecta in mod natural relatiile dintre entitati, precum cea dintre elevi si meditatori, sau dintre meditatori si materiale etc.

1. **Frontend**
2. **Blade**

Blade este motorul de templating implicit in Laravel, care permite crearea de interfete frontend dinamice si usor de intretinut. Acesta permite reutilizarea componentelor si optimizarea codului HTML prin utilizarea structurilor conditionale si a buclelor.

1. **TailwindCSS**

TailwindCSS este un framework CSS care permite stilizarea rapida a interfetelor web prin clase predefinite, evitand scrierea de CSS personalizat pentru fiecare element. Un dezavantaj al acestui framework este faptul ca utilizarea extinsa a claselor utilitare poate duce la un cod HTML mai lung si mai greu de citit, dar care poate fi gestionat prin folosirea claselor personalizate.

1. **Tehnologii auxiliare**
2. **Vite**

Vite este un bundler modern si rapid utilizat pentru dezvoltarea aplicatiilor frontend. Acesta ofera un timp de pornire aproape instant si reincarcare automata a paginii la fiecare modificare, folosind Hot Module Replacement (HMR), ceea ce accelereaza semnificativ procesul de dezvoltare. In plus, Vite a fost folosit pentru a compila si livra eficient resursele frontend.

1. **Git**

Pentru versionarea codului sursa am folosit Git, un sistem de control al versiunilor extrem de popular in dezvoltarea de software. Git permite urmarirea modificarilor aduse codului de-a lungul timpului si revenirea la versiuni anterioare cand este necesar.

1. **Visual Studio Code**

# 4. Arhitectura aplicatiei si implementarea

**4.1. Arhitectura generala**

Aplicatia este construita pe o arhitectura bazata pe componente (component-based architecture – CBA). Aceasta permite separarea logicii aplicatiei in componente independente, fiecare responsabila pentru o anumita functionalitate (de exemplu, componenta pentru sistemul de notificari, componenta pentru sistemul de mesagerie).

**4.1.1. Arhitectura MVC (Model-View-Controller)**

Modelul MVC este una dintre cele mai raspandite arhitecturi in dezvoltarea aplicatiilor web. Asa cum se poate observa in Figura 1 [9], arhitectura MVC separa clar logica aplicatiei in 3 componente:

1. Modelul – nucleul aplicatiei, este responsabil cu gestionarea datelor. Modelul acceseaza si manipuleaza datele, de regula prin interactiunea cu baza de date, si raspunde cererilor venite din partea controller-ului.
2. Controller-ul – actioneaza ca un intermediar intre Model si View. El primeste cererile din partea utilizatorului, proceseaza logica necesara, de obicei apeland unul sau mai multe modele, si transmite rezultatele catre view-ul corespunzator.
3. View-ul – se ocupa cu prezentarea informatiei catre utilizator. Nu contine logica de procesare, ci doar de afisare.

*Figura 1 – Arhitectura MVC (Model-View-Controller)*

Printre avantajele arhitecturii MVC se numara:

* Separarea clara a responsabilitatilor, ceea ce duce la o structura de cod bine organizata si usor de intretinut.
* Faciliteaza dezvoltarea colaborativa, deoarece diferite echipe pot lucra in paralel pe componente diferite ale aplicatiei
* Testare mai usoara, deoarece componentele aplicatiei pot fi testate individual

Totusi, aceasta arhitectura prezinta si anumite dezavantaje, cum ar fi:

* Complexitatea suplimentara in aplicatii simple, unde separarea stricta a componentelor poate fi considerata excesiva.
* Volumul mai mare de cod necesar, intrucat chiar si functionalitatile simple presupun crearea si conectarea celor trei componente.

Este important de mentionat ca Laravel este construit nativ pe arhitectura MVC.

**4.1.1. Arhitectura aplicatiei dezvoltate**

Aplicatia este construita pe o arhitectura bazata pe componente (Component-Based Architecture – CBA), care poate fi privita ca o extensie a modelului clasic MVC. In aceasta arhitectura, logica aplicatiei este impartita in componente independente, fiecare responsabila pentru o functionalitate specifica sau un grup de functionalitati strans legate. De exemplu, exista componente distincte pentru sistemul de notificari si pentru sistemul de mesagerie.

Fiecare componenta este structurata in doua parti principale:

* Logica de procesare, implementata intr-o clasa PHP dedicata, care gestioneaza starea componentei, proceseaza datele si raspunde evenimentelor generate de utilizator sau de aplicatie.
* Partea de prezentare, realizata printr-un fisier Blade, care contine template-ul HTML si directivele specifice Laravel pentru afisarea dinamica a datelor si interactiunea cu utilizatorul.

Aceasta separare in cadrul componentei permite o organizare clara si modulara, in care logica de procesare si prezentarea sunt strans legate, dar totusi distincte.

Avantaje ale arhitecturii alese:

* Componentele pot fi dezvoltate si testate independent, ceea ce asigura o modularitate ridicata.
* Prin combinarea logicii si prezentarii in componente autonome, aplicatia poate oferi interactivitate si reactivitate fara nevoia de a folosi JavaScript explicit, deoarece actualizarile dinamice sunt gestionate automat pe server si sincronizate cu interfata grafica.

Totusi, arhitectura prezinta si unele dezavantaje:

* Deoarece reactivitatea este gestionata in intregime pe server, aceasta arhitectura poate intampina provocari legate de scalabilitate, in special cand numarul utilizatorilor si al interactiunilor cresc semnificativ. Procesarea centralizata poate duce la o incarcare mai mare a serverului.
* Deoarece fiecare interactiune a utilizatorului implica comunicarea cu serverul pentru procesare si actualizare, performanta aplicatiei poate fi influentata de viteza conexiunii la internet, ceea ce poate afecta experienta utilizatorului.

Prin urmare, in proiectarea si implementarea aplicatiei au fost luate in calcul aceste limitari, iar mecanismele de actualizare dinamica a interfetei au fost aplicate doar in acele zone in care au contribuit la imbunatatirea experientei utilizatorului, cu scopul de a optimiza performanta.

**4.2. Arhitectura bazei de date**

Baza de date reprezinta componenta fundamentala a aplicaitei, avand rolul de a stoca si organiza informatiile necesare functionarii platformei. Prin baza de date gestionam datele despre utilizatori, materialelor didactice, mesaje, notificari si alete elemente esentiale pentru desfasurarea activitatilor in cadrul platformei. Structura bazei de date este proiectata astfel incat sa asigure integritatea datelor si scalabilitatea sistemului.

Baza de date este relationala si este organizata in mai multe tabele conectate prin relatii de tip cheie primara – cheie externa.

**4.2.1. Structura bazei de date**

Acest subcapitol prezinta tabelele principale ale bazei de date. Fiecare tabel este conceput pentru a raspunde unei functionalitati specifice ale aplicatiei:

1. **Users**: Reprezinta utilizatorii platformei. Sunt identficati prin adrese de email si pot avea roluri diferite (student, meditator, administrator)
2. **UserProfiles**: Contine informatii aditionale despre utilizatori, separate de datele de autentificare din tabela Users. Include date cum ar fi numarul de telefon, calea catre poza de profil si o descriere pe scurt, afisate pe pagina de profil.
3. **PossibleAnswers**: Reprezinta variantele de raspuns posibile la intrebarile adresate utilizatorilor in etapa de creare a contului. Fiecarui raspuns ii este asociat un numar de intrebare, permitand gruparea logica.
4. **Answers**: Reprezinta raspunsurile selectate de utilizatori in timpul inregistrarii, fiind asociate atat cu tabela Users, cat si cu raspunsul ales din tabela PossibleAnswers.
5. **TutorsStudents**: Reprezinta relatia de tip *many-to-many* dintre meditatori si studenti. Fiecare intregistrare din acest tabel face legatura intre un utilizator cu rol de **meditator** si unul cu rol de **student**. Tabelul contine doua chei externa catre tabela Users, ambele referindu-se la ID-uri ale de utilizatori, dar cu roluri diferite.
6. **Content**: Reprezinta continutul educational adaugat de fiecare meditator pe platforma. Fiecare material este asociat cu un utilizator din tabela Users (autorul continutului) si este incadrat intr-o categorie definita in tabela Categories.
7. **Categories**: Reprezinta categoriile definite de meditatori pentru organizarea continutului. Acestea pot varia de la concepte generale (*Integrale, Ecuatii diferentiale*) pana la grupe specifice (*Informatica clasa a 10-a*). Fiecare categorie este creata de un meditator si este folosita pentru clasificarea materialelor din tabela Content. De asemenea, meditatorii pot acorda acces selectiv la anumite categorii studentilor lor, facilitand astfel o distributie personalizata a resurselor educationale.
8. **Conversations**: Reprezinta canalele de comunicare intre mai multi utilizatori fie in relatii unu-la-unu intre student si meditator, fie intre grupuri care includ un meditator si mai multi studenti. Fiecare conversatie are un identificator unic si este asociata cu patrticipantii implicati.
9. **Messages**: Stocheaza mesajele trimise intre util;izatori in cadrul conversatiilor definite in tabela Conversations. Fiecare mesaj este asociat cu un utilizator (expeditorul), o conversatie, contine textul mesajului si informatii precum momentul trimiterii.
10. **TutorRatings**: Contine evaluarile oferite de studenti meditatorilor. Fiecare evaluare este asociata cu un utilizator cu rol **student** si unul cu rol **meditator**, incluzand un scor de la 1 la 5 stele, si optional un comentariu. Acest tabel permite monitorizarea calitatii serviciilor oferite de meditatori pe platforma.

# Bibliografie

|  |  |
| --- | --- |
| [1] | Institutul de Stiinte ale Educatiei, „Liceenii din Romania,” [Interactiv]. Available: https://www.ise.ro/wp-content/uploads/2024/12/Raport-4\_Liceenii-din-Romania\_Implicarea-si-autonomia-liceenilor.pdf. [Accesat Mai 2025]. |
| [2] | Laravel, „Laravel documentation,” [Interactiv]. Available: https://laravel.com/docs/12.x. [Accesat Mai 2025]. |
| [3] | Laravel, „Blade Templating Engine Documentation,” [Interactiv]. Available: https://laravel.com/docs/12.x/blade. [Accesat Mai 2025]. |

# Bibliografie

[1] Laravel. *Laravel Documentation*, <https://laravel.com/docs/12.x>, [accesat: 05.2025].

[2] Laravel. *Blade Templating Engine Documentation*, <https://laravel.com/docs/12.x/blade>, [accesat: 05.2025].

[3] Livewire. *Livewire Documentation*, <https://laravel-livewire.com/docs>, [accesat: 05.2025].

[4] Tailwind Labs. *Tailwind CSS Documentation*, [https://tailwindcss.com/docs](https://tailwindcss.com/docs,), [accesat: 05.2025].

[5] saadeghi. *daisyUI – Tailwind CSS Component Library*, <https://daisyui.com/>, [accesat: 05.2025].

[6] Vite. *Vite Documentation*, <https://vite.dev/guide/>, [accesat: 05.2025].

[7] Bjorn Lu. *Hot Module Replacement is Easy*, <https://bjornlu.com/blog/hot-module-replacement-is-easy>, [accesat: 05.2025].

[8] Institutul de Științe ale Educației. *Liceenii din România: Implicarea și autonomia liceenilor*, <https://www.ise.ro/wp-content/uploads/2024/12/Raport-4_Liceenii-din-Romania_Implicarea-si-autonomia-liceenilor.pdf>, [accesat: 05.2025]

[9] https://developer.mozilla.org/en-US/docs/Glossary/MVC