**MINISTERUL EDUCAŢIEI**

**UNIVERSITATEA „1 DECEMBRIE 1918” DIN ALBA IULIA**

**SPECIALIZAREA INFORMATICĂ**

**FORMA DE ÎNVĂŢĂMÂNT ZI**

**REȚEA DE SOCIALIZARE PENTRU STUDENȚI, PROFESORI, UNIVERSITĂȚI**

**ABSOLVENT:**

**COJOCARU AUGUSTIN-STELIAN**

**COORDONATOR ȘTIINȚIFIC:**

**CONF. UNIV. DR. ROTAR CORINA**

**ALBA IULIA**

**Iulie, 2022**

Contents

[Capitolul 1 Introducere 3](#_Toc107183344)

[1.1. Scopul proiectului si viziunea 3](#_Toc107183345)

[1.2. Task-uri propuse 3](#_Toc107183346)

[Capitolul 2 Requirements engineering 4](#_Toc107183347)

[2.1. Requirements (functional si non-functional) 4](#_Toc107183348)

[2.1.1. Functional requirements (lista) 4](#_Toc107183349)

[2.1.2. Non-functional requirements (lista) 4](#_Toc107183350)

[2.1.3. Traceability matrix 4](#_Toc107183351)

[2.1.4. Procesul de proritizare a requirement-urilor 4](#_Toc107183352)

[2.2. Resurse necesare (technology requirements) 4](#_Toc107183353)

[2.3. Timeline 4](#_Toc107183354)

[2.4. Diagrame –use-case, UML 4](#_Toc107183355)

[Capitolul 3 Descrierea tehnologiilor utilizate 4](#_Toc107183356)

[3.1. Generalizari 4](#_Toc107183357)

[3.2. Partea de Front-end 6](#_Toc107183358)

[3.2.1. HTML & CSS 6](#_Toc107183359)

[3.2.2. JavaScript 6](#_Toc107183360)

[3.3. Partea de Back-end 9](#_Toc107183361)

[Capitolul 4 Procesul tehnologic 10](#_Toc107183362)

[4.1. Front-end 10](#_Toc107183363)

[4.1.1. CSS 10](#_Toc107183364)

[4.1.2. JavaScript 13](#_Toc107183365)

[4.2. Back-end 15](#_Toc107183366)

[Capitolul 5 Proces de instalare 15](#_Toc107183367)

[Concluzii și propuneri 15](#_Toc107183368)

[Lista anexelor 15](#_Toc107183369)

[Anexe 15](#_Toc107183370)

# Introducere

## Scopul proiectului si viziunea

Scopul proiectului este crearea unei rețele orientata pe studenți ce sunt in căutare de o instituție unde a-si continua studiile, pe studenți ce deja studiază in vreo instituție de învățământ sau au studiat cândva, pe profesori ce activează sau au activat in vreo instituție de învățământ, precum si pe universități ce funcționează in prezent sau au funcționat. Principalul obiectiv in crearea acestei retele este de a rezolva problemele studentilor in de-a-si alege continuarea potrivita si pe plac a studiilor. Astfel, cu ajutorul retelei date, userii vor putea interactiona intre ei pentru a se ajuta. Pe langa obiectivul dat, reteaua poate fi folosita si ca un simplu instrument de comunicare si interactiune sociala intre useri, oferindu-se urmatoarele funcții pentru aceasta:

* Mesagerie,
* Postari,
* Recenzii catre user,
* Comentarii la postari si recenzii,
* Like-uri, dislike-uri la postari, recenzii si comentarii,
* Friendship mecanism, etc.

## Task-uri propuse

Principalele task-uri propuse la faza initiala de proiectare a proiectului propus sunt:

* Crearea unui site web
* Utilizarea tehnologiilor HTML5, CSS, Javascript, SQL, PHP pentru a crea site-ul dat
* Alegerea unui mediu confortabil de programare
* Alegerea unor framework-uri potrivite pentru fiecare din tehnologiile mentionate mai sus
* Crearea in prim plan a funcțiilor specifice unei retele de socializare
* Crearea unei interfate atractive, easy-to-use si intuitive pentru acest site
* Creare unui spatiu pentru administratorul site-ului pentru a facilita managementul acestui site

# Requirements engineering

## Requirements (functional si non-functional)

### Functional requirements (lista)

### Non-functional requirements (lista)

### Traceability matrix

### Procesul de proritizare a requirement-urilor

## Resurse necesare (technology requirements)

## Timeline

## Diagrame –use-case, UML

# Descrierea tehnologiilor utilizate

## Generalizare

Ca mediu de programare s-a utilizat Visual Studio. De ce? Pentru ca:

* Este gratis;
* Suporta mai multe limbaje de programare;
* Automat identifica limbajul de programare din fisier, dar utilizatorul poate si el sa schimbe limbajul corespunzator fisierului in care lucreaza;
* Integrare git încorporată;
* IntelliSense;
* Posibilitatea instalarii unui numar mare de extensii;
* Prezenta indexarii.spatiului de lucru.

Inițial elaborarea site-ului Studbud era orientata pe JQuery si PHP curat, insa au aparut mari dificultăți in realizarea unor funcții mai complexe, si respectiv o mare complexitate in munca din cauza numărului mare de funcții ce necesitau o efectuare fără refresh la pagina (partea de Front-end) si funcțiilor complexe de pe Back-end. Realizarea La partea de Back-end, din cauza unui număr mare de tabele SQL, se necesita instrumente ce ar usura gestiunea informatiei din tabele, si mai mult in cazul in care tabelele au o relatie intre ele. De asemenea pe partea de securitatea ee necesita multa munca, si anume indeplinirea anumitor funcții in dependenta de tipul user-ului, ce permisiuni are acesta asupra site-ului etc. Evident, utilizarea PHP curat marea complexitatea realizarii acestor chestii cu mult.

Pentru problema de Front-end s-a ajuns la alegerea unui framework pentru reactivitatea site-ului nostru si o alegere s-a facut pentru Vue JS, care pare a fi foarte satisfacator si foarte flexibil fata de concurentii sai asemenatori (React sau Angular). Utilizarea JQuery-ului in cazul nostru necesita dupa fiecare gestiune a intormatiei din Back-end un upload de cod HTML pentru realizarea unor efecte dupa efectuarea acestor funcții, ce nu este foarte comod in cazul unui numar mare de funcții, plus. Deci, cea mai buna alegere este utilizarea anumitor state-uri, variabile, ce ar permite rendering-ul anumitor efecte in baza de informatia din Back-end (primita de asemenea in forma de variabile). Vue js-ul de asmenea ne pune la dispozitie si instrumente ca MIXINS, care ne permite reutilizarea unui cod JS in cazurile in care se necesita aceeasi solutie de rezolvare. Totodata Vue JS ne pune la dispoziție numeroase directive pentru diferite situatii, care reprezinta niste componente cu solutii deja gata (exemple: tabele de gestiune a informatiei pentru admin panel, hover directive etc)…

Deci, in realizarea site-ului nostru ne-am axat:

*Pentru mediul de dezvoltare:*

Visual Studio Code

*Pentru partea de Front-end:*

CSS – Bootstrap

Javascript – Vue js, JQuery

*Pentru partea de Back-end:*

PHP – Laravel

*Pentru baza de date:*

Localhost – XAMPP

Sistem de gestiune a bazei de date - MySQL



Img 3‑1 Logo-urile: 1) Laravel; 2) Bootstrap; 3) Vue JS

## Partea de Front-end

### HTML & CSS

Pentru HTML am utilizat versiunea 5 a acestuia (HTML 5), care este cea mai actuala.

Aici s-a ales libraria Bootstrap (versiunea 5.1), deoarece este gratisa si contine multe componente necesare site-ului pe care l-am ales sa-l implementam. Plus la aceasta, Vue JS-ul, care s-a ales ca framework javascript pentru Front-end, contine si el componente bootstrap facute sub tehnologia Vue JS, majoritatea fiind mult mai simplu de utilizat in Vue JS decat in versiunea originala clase de CSS. De asemenea, utilizarea librariei Bootstrap simplifica mult lucrul cu partea responsive a site-ului.

### JavaScript

Pentru o interfata easy-to-use este nevoie de o reactivitate intre actiunile user-ului si informatia ce se rendereaza pe web-site. Pentru reactivitate s-a ales sa se foloseasca Vue JS (versiunea 2). Plusul cel mai mare la Vue JS fata de alte framework-uri asemanatoare este ca acesta se poate utiliza foarte simplu si intr-o programare mixta dintre componente reactive si un web-site care nu este single page.

Adaptabilitatea Vue.js este ceea ce atrage în primul rând dezvoltatorii și face ca lucrul cu acesta să fie plăcut și mai eficient. Vue.js controlează nivelul fundamental al MAC-ului, oferă o interfață de utilizator mai bună și o integrare ușoară a aplicațiilor.

Pentru a facilita înțelegerea framework-ului, iată câteva dintre cele mai importante avantaje ale utilizării acestuia ca platformă de dezvoltare.

Framework-ul Vue.js se bazează pe șablonul MVVM (Model-View-ViewModel), derivat din șablonul Clasic Model-View-Controller (MVC). Aspectul său a contribuit la separarea dezvoltării interfeței interfeței grafice a utilizatorului și a logicii interne de afaceri, ceea ce crește semnificativ eficiența creării aplicațiilor. Nucleul MVVM este un strat ViewModel, oarecum similar cu un convertor de valoare responsabil pentru schimbarea obiectelor de date din model.

Ușor de utilizat și integrare ușoară. Acesta este unul dintre instrumentele cele mai prietenoase pentru începători, cu o curbă de învățare scăzută în comparație cu alte cadre. De asemenea, va fi intuitiv pentru dezvoltatorii care au lucrat anterior cu jQuery și Angular. De asemenea, este ușor să îl integrați în alte biblioteci sau să îl utilizați ca proiect separat.

Se compune din componente. Componenta este o parte importantă a Vue.js. Acestea sunt blocuri de cod care pot fi utilizate în mod repetat, incluzând atât o descriere a interfeței aplicației, cât și implementarea capabilităților sale. Dezvoltatorul are nevoie de ele pentru a crea o bază de cod modulară ușor de întreținut pentru crearea de programe la scară largă.

Eficiență. Vue.js este un cadru mic și rapid din punct de vedere al performanței, ceea ce îl face unul dintre cele mai importante instrumente pentru dezvoltarea aplicațiilor mobile.

Legarea reactivă a datelor. Am vorbit deja despre asta la început. De fiecare dată când datele se schimbă, DOM este actualizat automat.

Dom Virtual. Lumea Vue.js se învârte în jurul DOM. De fapt, servește ca o interfață între scripturi și marcare, ceea ce facilitează trimiterea și primirea datelor din HTML. La schimbarea elementelor CSS, în unele cazuri este necesară reconstruirea întregului copac. Vue.js creează o copie în Dom virtual, nu în Dom browser-ul. Prin eliminarea nevoii de duplicare a informațiilor, productivitatea generală crește.

Principalul element cheie in cadrul Vue.js este componenta. O componentă este închisă, reutilizabilă, cel mai adesea singura parte a logicii interfeței cu utilizatorul.

În ciuda faptului că folosim expresia "componentă Vue" aici, este important să știm că componentele nu sunt o definiție specifică pentru acest framework special. Putem vedea aceeași structură în React și în blocul Cadru Svelte.

Abordarea "componentă" este o modalitate de a structura dezvoltarea frontend fără a umfla foarte mult proiectele și a economisi timp prețios. În centrul majorității cadrelor componente se află reactivitatea, care contribuie la experiența utilizatorului fără a compromite dezvoltatorul. Fiecare componentă are propria stare, marcare și stil, permițând programatorului să creeze aplicații organizate într-un copac.

Până în prezent, există multe soluții (componente) gata făcute care pot fi utilizate pentru integrarea lor nedureroasă în orice proiect.

Când creați o componentă în .în fișierul vue, la nivelul de bază veți găsi trei secțiuni pentru introducerea codului:

* **Șablon (șablon).**

Scris într-o versiune extinsă a limbajului șablonului (HTML), acesta servește ca directivă. Acestea sunt un fel de reguli pentru crearea marcajului final al unei componente pe baza stării sale interne.

* **Scr ipt (script).**

Iată logica aplicației. Această secțiune include:

Caracteristici. Un set de variabile de intrare utilizate pentru a configura comportamentul componentei. Acestea pot fi filtre de date, ordine de sortare, etichete, comutatoare de vizibilitate etc.

Stare. Acest lucru este opțional, dar este adesea prezent. Aceasta este o structură de date care asigură starea componentei la un moment dat. Se va schimba în timp, în funcție de evenimentele care au loc (Faceți clic, derulați, mutații DOM).

* **Styles.**

Fișierele de stil CSS sunt plasate aici.

Componentele au întotdeauna un nume specific. Aceasta este ceea ce compilatorul Vue caută pentru a crea și monta instanțele sale. Pentru a utiliza o componentă într-un șablon, pur și simplu creați o etichetă HTML cu numele său în paranteze triunghiulare.

Cel mai simplu mod de a înregistra componente este să o faceți la nivel global folosind metoda Vue.component. Cu înregistrarea globală, acestea pot fi utilizate oriunde în arborele component al aplicației dvs., de exemplu:

// for pages

Vue.component('v-register', require('./pages/Register.vue').default);

Vue.component('v-login', require('./pages/Login.vue').default);

Vue.component('v-contact', require('./pages/Contact.vue').default);

Vue.component('v-search', require('./pages/Search.vue').default);

Vue.component('v-chat', require('./pages/Chat.vue').default);

Vue.component('v-friends', require('./pages/Friends.vue').default);

Vue.component('v-page-reviews', require('./pages/Reviews.vue').default);

Vue.component('v-page-feed', require('./pages/Feed.vue').default);

Vue.component('v-page-files', require('./pages/Files.vue').default);

Fragment de cod 3‑1 Exemplu de inregistrare globala intr-un fisier app.js a componentelor

Pe alocuri s-a utilizat si libraria JQuery, si anume acolo unde se necesita lucrul cu CSS sau cu unele lucruri pentru care nu exista un echivalent in Vue JS iar scrierea lor in Vanilla JS era sa fie cu mult mai complexa decat echivalentul lor in JQuery (spre ex. adaugarea unei clase unui grup de elemente, sau selectarea unui element dupa denumirea clasei, id-ului etc.).

## Partea de Back-end

Pentru realizarea Back-end-ului, s-a ales a se folosi libraria PHP – Laravel.

Laravel este un proiect 100% open source. Acest lucru deschide oportunități excelente de personalizare, modificare și extindere, iar aici Laravel depășește multe cadre populare. Hub, GitHub și BitBucket.

Laravel folosește biblioteci partajate cu Symfony. Aceste cadre urmează cele mai bune practici de dezvoltare și proiectare. Prin urmare, aceste proiecte sunt compatibile. Acest lucru este deosebit de important pentru software-ul de întreprindere, deoarece vă permite să mențineți o bază de cod curată, minimalistă și eficientă, ușor de modificat.

Aplicațiile Laravel oferă performanțe mai mari în comparație cu aplicațiile create folosind alte cadre. Acest lucru este posibil și datorită sistemului de cache. Driverul de cache a fișierelor stochează multe elemente în sistemul de fișiere. Acest lucru vă permite să dezvoltați rapid aplicații.

Laravel din cutie oferă un sistem de autentificare simplu și eficient. Oferă chiar și controlul accesului la resurse. Acest lucru vă permite să restricționați accesul utilizatorilor neautorizați la anumite resurse.

Laravel asigură securitatea aplicațiilor. Baza de cod a cadrului este protejată de amenințări, de exemplu, injecții SQL sau falsificarea interogării între site-uri (CSRF). Acest lucru protejează utilizatorii de pierderea datelor importante.

Iată câteva caracteristici tehnice care fac din Laravel cel mai bun cadru PHP:

* Built-in Eloquent ORM. Acest sistem vă permite să lucrați cu diferite baze de date utilizând implementarea șablonului ActiveRecord. Datorită Eloquent ORM, puteți lucra cu baze de date fără a fi nevoie să scrieți interogări SQL complexe.
* Șabloane Blade (Șabloane Blade). Laravel susține designul arhitectural MVC. Vă permite să separați interfața cu utilizatorul și logica de business. Laravel acceptă, de asemenea, motorul Blade template, care vă permite să utilizați codul PHP nativ.
* Viteză mare de dezvoltare. Laravel vă permite să creați rapid aplicații, deoarece dezvoltatorii pot face fără cod complex în acest proces. După cum sa menționat mai sus, framework-ul este construit pe arhitectura MVC, ceea ce oferă acces la toată infrastructura necesară pentru crearea unui site web și economisește timp.
* Gestionarea eficientă a traficului. Dacă site-ul este popular, aplicația trebuie să proceseze un număr mare de solicitări în fiecare secundă. Acest lucru crește încărcarea pe server și costul găzduirii. Mai mult, datorită încărcării mari, serverul poate să nu mai răspundă, ceea ce duce uneori la pierderea datelor. Cu Laravel, astfel de riscuri sunt reduse la minimum, deoarece framework-ul implementează un sistem unic de așteptare a mesajelor. Acest sistem echilibrează sarcina pe server, ceea ce asigură funcționarea neîntreruptă și securitatea datelor.

# Procesul tehnologic

## Generalizare

Pana a incepe a descrie procesul de lucru cu tehnologiile si instrumentele utilizate in proiectul nostru, ar fi bine de mentionat procesul initial de instalare a acestor instrumente (se presupune ca XAMPP este deja instalat).

Pentru a putea mediul Laravel pentru aplicatia noastra, este nevoie initial de instalat Composer. Procesul de instalare a acestuia se poate gasi pe: [*https://getcomposer.org/download/*](https://getcomposer.org/download/). Dupa instalarea acestuia putem recurge la instalarea proiectului nostru in Laravel cu urmatoarele comenzi in terminalul mediului nostru de programare VSCode:

1. composer global require laravel/installer
2. laravel new denumirea-aplicatiei (cazul nostru StudBud)
3. cd denumirea-aplicatiei (cazul nostru StudBud)
4. php artisan serve – pentru deschiderea serverului

Dupa s-a creat in folderul views deja prezent in proiectul Laravel un alt folder - layouts, in care s-a creat un fisier cu denumirea app.blade.php. Aici am creat un layout comun pentru toate paginile ulterioare ale site-ului in head-ul caruia vom atasa ulterior link-urile pentru instalarea celorlalte instrumente ce le-am folosit in proiect.

Dupa cum s-a mentionat anterior, am utilizat librarila Bootstrap pentru CSS. S-a folosit conectarea Bootstrap-ului prin amplasarea link-urilor CDN la head-ul site-ului. Libraria Bootstrap necesita conectarea mai multor link-uri pentru a face functionala utilizarea acesteea. Astfel au fost conectate urmatoarele link-uri (conectarea Bootstrap-ului pentru functionarea sa deplina necesita si conectarea librariei JQuery):

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"

        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

Fragment de cod 4‑1 CDN pentru CSS

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"

        integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"

        integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous">

    </script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"

        integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous">

    </script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"

        integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous">

    </script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"

        integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous">

    </script>

Fragment de cod 4‑2 CDN conectare pentru librariile js:Jquery, Popper, Bootstrap si, repsectiv, Bundle

## Front-end

### CSS

In cadrul realizarii CSS pentru site-ul nostru am utilizat libraria Bootstrap. Utilizarea Bootstrap-ului consta in atribuirea unor clase ce contin deja niste stiluri prestabilite de librarie, pentru diverse tag-uri HTML in scop-ul de a stiliza pagina web.

In cadrul site-ului nostru s-a utilizat mult sistemul grid oferit de Bootstrap. Plusul mare a acestui system este ca ne ofera si posibilitatea de a face concomitant si responsive site-ul ce-cream, continand clase pentru 5 diverse marimi ale ecranului:

(Bootstrap, Grid system · Bootstrap v5.1, 2021)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Extra small** <576px | **Small** ≥576px | **Medium** ≥768px | **Large** ≥992px | **Extra large** ≥1200px |
| **Max container width** | None (auto) | 540px | 720px | 960px | 1140px |
| **Class prefix** | .col- | .col-sm- | .col-md- | .col-lg- | .col-xl- |
| **# of columns** | 12 | | | | |
| **Gutter width** | 1.5rem (.75rem on left and right) | | | | |
| **Custom gutters** | Yes | | | | |
| **Nestable** | Yes | | | | |
| **Column ordering** | Yes | | | | |

Table 4‑1 Descrierea succinta a sistemului grid oferit de Bootstrap

<div class="row">

      <div class="col-12 col-md-8 col-lg-12 col-xl-8 left-wrapper">

...

</div>

</div>

Fragment de cod 4‑3 Exemplu de utilizare a claselor pemtru sistemul grid

O alta clasa des utilizata in cadrul proiectului este flex-ul. Clasele pentru flex cu variantele responsive:

* .d-flex
* .d-inline-flex
* .d-sm-flex
* .d-sm-inline-flex
* .d-md-flex
* .d-md-inline-flex
* .d-lg-flex
* .d-lg-inline-flex
* .d-xl-flex
* .d-xl-inline-flex
* .d-xxl-flex
* .d-xxl-inline-flex

(Bootstrap, Flex · Bootstrap v5.1, 2021)

Pe langa clase, Bootstrap ofera secvente de tag-uri cu clase prestabilite in scopul crearii unor componente. Componente de acest gen utilizate in cadrul proiectului dat sunt:

* **Accordion** - element de colaps vertical pentru a afișa și ascunde conținutul prin modificări de clasă.
* **Alerte** – niste div-uri stilizate ce permite furnizarea unor mesaje pentru diverse actiuni ale user-ului.
* **Badge-uri** – componenta stilizat pentru etichete sau count componente.
* **Butoane** – componenta stilizat pentru butoane.
* **Card** - container de conținut flexibil și extensibil. Include opțiuni pentru anteturi și subsoluri, o mare varietate de conținut, culori de fundal contextuale și opțiuni puternice de afișare.
* **Carousel** - componentă slideshow pentru parcurgerea prin elemente-imagini sau diapozitive de text—ca un carusel.
* **Collapse** – o componenta alcatuita dintr-un button si informatia ce poate fi ascunsa sau aratata in functie de starea sa printr-un click pe button.
* **Dropdowns** – acelasi efect ca in cazul componentei collaspse, doar ca informatia gestionata nu are o pozitie relative, ci absoluta.
* **Liste** – componente stilizate pentru liste.
* **Navbar** – componenta bootstrap stilizata pentru navbar.
* **Placeholders** - pot fi utilizați pentru a îmbunătăți experiența aplicației dvs, atunci cand informatia nu este incarcata inca si este nevoie de tras atentia utilizatorului in ce parte aceasta va fi ulterior incarcata si totodata de a arata si momentul de incarcare a acestei informatii. Sunt construite doar cu HTML și CSS, ceea ce înseamnă că nu aveți nevoie de JavaScript pentru a le crea. Cu toate acestea, avem nevoie de un cod JS pentru a comuta vizibilitatea acestora.
* **Nav-uri si tab-uri** – componente stilizate pentru bare de navigare.
* **Modal** – un component similar cu card ce apare pozitionat deasupra altor elemente. S-au folosit mai mult in cazurile in care se necesita confirmarea unei actiuni de catre utilizator sau pentru galerie.
* **Offcanvas** - este o componentă bara laterală care poate fi comutata prin JavaScript să apară de la marginea din stânga, dreapta, sau de jos a viewport. Butoanele sau ancorele sunt folosite ca declanșatoare care sunt atașate la anumite elemente pe care le comutați, iar atributele de date sunt folosite pentru a invoca JavaScript-ul nostru. De obicei s-au utilizat in cadrul unor filtre sau a unui meniu ce poate fi afisat in cazul ecran-urilor mici (ex. sidebar la admin panel).

### JavaScript

Dupa cum s-a mentionat anterior, in proiectul dat partea cea mai majora in JS il ocupa framework-ul Vue.js. In construirea website-ului cu Vue.js s-au folosit foarte des built-in directive ale framework-ului, si anume:

* **v-if**. Această directivă este responsabilă pentru redarea condiționată a elementelor. Dacă controlați vizibilitatea unui element prin if, atunci elementul dvs. va dispărea și va apărea în arborele DOM în funcție de valoarea din directiva. v-if ia valori boolene. De asemenea, in aceasta directiva putem atribui si metode/functii care returneaza o anumita valoare booloeana.

<div

            class="hstack gap-1"

            v-if="im\_user == false || user\_login == false"

          >

            <template v-if="user.status == 'not\_friends'">

...

</template>

</div>

Fragment de cod 4‑4 Exemplu de utilizare a directivei v-if cu o variabila

* **v-else, v-else-if**. Directiva if este convenabilă deoarece poate fi utilizată împreună cu directivele v-else și v-else-if. În funcție de valori, va fi afișată una sau alta componentă.

 <template v-if="tip\_user == 'profesor'">

 Nr. de institutii in care ai lucrat

 </template>

 <template v-else-if="tip\_user == 'student1'">

 Nr. de institutii in care ai invatat

 </template>

Fragment de cod 4‑5 Exemplu de utilizare a directivei if in combinatie cu v-else

* **v-for.** Pentru a afisa datele unui array intr-o lista de elemente exista directiva v-for. Aceasta are urmatoarea sintaxa – v-for = „item in items” :key = item.id, sau o varianta alternativa – v-for = „(item, index) in items” :key = index. Ulterior in interiorul tag-ului asupra caruia aplicam acest v-for putem sa ne adresam catre elementul iterat.

<template>

  <ul class="students">

    <li v-for="student in students">{{student}}</li>

  </ul>

</template>

<script>

    export default {

        name: 'App',

        data() {

            return {

                students: [

                    'Alex',

                    'Robert',

                    'Marta'

                ]

            }

        }

    }

</script>

Fragment de cod 4‑6 Exemplu cu utilizarea v-for

**!Este important ca pentru „:key” sa fie introdusa o valoarea unica a listei pe care vrem sa o afisam.**

* **v-bind.** Directiva este utilizată pentru așa - numita legare a datelor. Pentru a transmite date dinamice în interiorul unui atribut html, este necesar să transmiteți acest atribut ca argument la directiva v-bind. Argumentele directivelor sunt transmise după caracterul de doua puncte. In loc de „v-bind:” se poate folosi doar „:”.

<a

          @click="modal.id = image.id"

          :href="'#image\_' + index\_gallery"

          data-bs-toggle="modal"

        >

          <img class="rounded w-100 h-100" :src="image.path" alt="" />

        </a>

Fragment de cod 4‑7 Pentru href si src se transmit niste variabile dinamice, care se pot schimba pe parcurs de utilizare a site-ului

* **v-model.** Directiva este intr-un fel similară cu v-bind și servește pentru legarea bidirecțională între elementele formularului de date și datele din obiectul de date (two-way binded). Directiva ignoră atributele html: *value, checked, selected*, care sunt prezente în elemente. Pentru valoarea reală, directiva ia datele pe care le transmitem de la obiectul *data*. In exemplul de mai jos am legat valoarea inputului cu variabila name.search din obiectul data al exemplarului vue. La introducerea unor caractere in inputul dat se vor inscrie aceste caractere in variabila *name.search*.

<input

type="text"

placeholder="Cauta persoana..."

class="form-control form-control-sm"

    v-model="name\_search"

/>

Fragment de cod 4‑8 Exemplu cu utilizarea directivei v-model

* **v-on.** Adăugarea unui eveniment la un element este implementată de directiva v-on. Numele evenimentului trebuie transmis ca argument al directivei după doua puncte. Numele evenimentelor sunt similare cu evenimentele JavaScript. Se poate omite scrierea v-on, lasand doar doua puncte si denumirea actiunii ce trebuie ascultata pe element.

<button

v-if="my\_id == selected\_user\_id"

@click="delete\_comment(comment, 'comment')"

class="btn btn-sm btn-danger"

>

<i class="bi bi-trash3-fill"></i>

</button>

Fragment de cod 4‑9 La click pe butonul din exemplu se va efectua functia "delete\_comment"

Pe langa click, vue oferă pseudonime si pentru alte cele mai frecvent utilizate coduri cheie:

* + .**enter**
  + **.tab**
  + **.delete (**captures both “Delete” and “Backspace” keys**)**
  + **.esc**
  + **.space**
  + **.up**
  + **.down**
  + **.left**
  + **.righ**

De asemenea se poate utiliza si keyCode atribute in definirea ce taste trebuie ascultate.

<input v-on:keyup.13="submit">

Fragment de cod 4‑10 Exemplu de utilizare a v-on cu keyCode

Totodata Vue ofera si posibilitatea definirii pseudonimului pentru diverse taste:

// enable `v-on:keyup.f1`

Vue.config.keyCodes.f1 = 112

Fragment de cod 4‑11 Definirea unui pseudonim pentru tasta cu codul 112

## Back-end

# Proces de instalare

# Concluzii și propuneri

**Bibliografie**

Bootstrap. (2021). *Flex · Bootstrap v5.1*. Preluat de pe getbootstrap.com: view-source:https://getbootstrap.com/docs/5.1/utilities/flex/

Bootstrap. (2021). *Grid system · Bootstrap v5.1*. Preluat de pe getbootstrap.com: https://getbootstrap.com/docs/5.1/layout/grid/

# Lista anexelor

# Anexe