

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

APLICAȚIE WEB DE BLOGURI

PROIECT DE DIPLOMĂ

Autor: **Alexandru LEŞAN**

Conducător științific: **S.L. Dr. Ing. Radu DAN**

2022



Vizat,

DECAN

Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT AUTOMATICĂ

Prof. dr. ing. Honoriu VĂLEAN

Autor: **Alexandru LEŞAN**

Aplicație web de bloguri

1. **Enunțul temei:** Aplicația curentă este o aplicație web dedicată împărtășirii gândurilor, ideilor, cunoștințelor sau întâmplărilor personale. Practic, aceasta aplicație este un jurnal unde fiecare utilizator poate scrie ce vrea iar cei interesați pot citi.
2. **Conținutul proiectului:** Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Introducere, Studiu bibliografic, Analiza, proiectare, implementare, Concluzii, Bibliografie, Anexe.
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca
4. **Constanți:** S.L. Dr. Ing. Radu DAN
5. **Data emiterii temei:**
6. **Data predării:** 29.08.2022

Semnătura autorului

Semnătura conducerii științific



**Declarație pe proprie răspundere privind
autenticitatea proiectului de diplomă**

Subsemnatul(a) Alexandru LESAN, legitimat(ă) cu CI/BI _____ seria PE
nr.061835410, CNP 1980828410045,

autorul lucrării:

Aplicatie web de bloguri

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată**, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Septembrie 2022 a anului universitar 2021-2022, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

29.08.2022

Prenume NUME

Alexandru LESAN

(semnătura)



SINTEZA

proiectului de diplomă cu titlul:

Aplicație web de bloguri

Autor: **Alexandru LEŞAN**

Conducător științific: **S.L. Dr. Ing. Radu DAN**

1. Cerințele temei: Realizarea unei aplicații web care permite crearea, vizualizarea, editarea, ștergerea articolelor centralizat pentru mai mulți autori cat și alte funcționalități ca: înregistrarea, logarea, recuperarea parolei, editarea profilului, căutarea, filtrarea și sortarea articolelor după diferite criterii, crearea, vizualizarea, editarea și ștergerea comentariilor, aprecierea cu like/dislike a articolelor dar și salvarea lor.
2. Soluții alese: Pentru implementarea cerinței stabilite s-au folosit următoarele tehnologii:
 - Back-end: PHP, Laravel, Restify, MySQL
 - Front-end: HTML, JavaScript, CSS, TailwindCSS, TailwindUI, HeadlessUI
3. Rezultate obținute: O aplicație web care realizeaza toate cerințele stabilite inițial, și poate fi pusa la dispozitia publicului pentru folosire.
4. Testări și verificări: Testarea aplicației s-a realizat atât live prin simularea cazurilor reale pornind de la înregistrarea unui nou utilizator pana la executarea tuturor funcționalităților puse la dispoziție de interfața web. Totodată, s-a realizat și testarea endpoint-urilor disponibile prin programul Postman, în care au fost introduse date atât corecte cat și incorecte pentru testarea algoritmului de funcționare. Aplicația a fost supusă testării de către un grup alcătuit de 3 persoane, în urma careia fiecare participant și-a expus părerea, a lăsat un feedback și idei pentru dezvoltare.



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

5. Contribuții personale: Dezvoltarea integrală a aplicatiei și a modului de funcționare a acesteia.

6. Surse de documentare: Articole științifice, documentații, tutoriale video/scrisse, cursuri disponibile online.

Semnătura autorului

Legean

Semnătura conducătorului științific

DRadu

Cuprins

1	INTRODUCERE	3
1.1	CONTEXT GENERAL.....	3
1.2	OBIECTIVE	3
1.3	SPECIFICAȚII.....	4
2	STUDIU BIBLIOGRAFIC.....	6
2.1	BLOG.....	6
2.1.1	<i>Structura blogului.....</i>	6
2.2	SOLUȚII EXISTENTE	7
2.2.1	<i>Aplicații similare.....</i>	7
2.2.1.1	Medium	7
2.2.1.2	News Break.....	7
2.2.1.3	Substack.....	7
2.2.1.4	Vocal Media	7
2.2.2	<i>Tehnologii folosite.....</i>	8
2.2.3	<i>Funcții oferite</i>	9
2.2.3.1	Încorporarea (Embedding) - Medium	9
2.2.3.2	Etichete (Tags)	9
2.2.3.3	Evidențiere (Highlights).....	10
2.2.3.4	Partajarea schițelor (Sharing Drafts).....	10
2.2.3.5	Carduri sociale (Social Cards)	10
2.2.3.6	Comentarii (Notes)	11
2.2.3.7	WYSIWYG – HTML Editor	11
2.3	PROTOCOLUL HTTP	11
2.3.1	<i>Diferenta între HTTP si HTTPS</i>	13
2.4	APLICAȚIE CU O SINGURĂ PAGINĂ (SPA)	13
2.4.1	<i>Avantaje</i>	14
2.4.2	<i>Dezavantaje.....</i>	14
2.5	HTML.....	14
2.5.1	<i>HTML - definitie</i>	14
2.5.2	<i>Exemplu pagina HTML</i>	15
2.5.3	<i>Caracteristicile HTML</i>	16
2.6	CSS (CASCADING STYLE SHEETS)	16
2.6.1	<i>Avantajele CSS.....</i>	16
2.7	JAVASCRIPT	17
2.7.1	<i>La ce se foloseste JavaScript.....</i>	17
2.7.2	<i>Avantajele utilizarii JavaScript</i>	18
2.8	VUE.JS	18
2.8.1	<i>Caracteristici Vue.JS:</i>	18
2.8.2	<i>Cine îl folosește.....</i>	19
2.9	LARAVEL.....	19
2.9.1	<i>Avantajele Laravel.....</i>	20
2.10	MySQL	21
2.10.1	<i>Cum funcționează baza de date MySQL.....</i>	21
2.10.2	<i>Cum funcționează relațiile cu bazele de date</i>	22
2.10.3	<i>Exemplu de comenzi MySQL:.....</i>	22
3	ANALIZĂ, PROIECTARE, IMPLEMENTARE	23

3.1	PREZENTAREA GENERALĂ.....	23
3.2	PROIECTARE.....	25
3.2.1	<i>Diagrama Use case.....</i>	25
3.3	IMPLEMENTARE.....	27
3.3.1	<i>Prezentarea operatiilor</i>	27
3.3.2	<i>Arhitectura bazei de date.....</i>	29
3.3.3	<i>Limbaje, framework-uri, pachete și servicii folosite la dezvoltarea aplicației</i>	32
3.4	TESTARE.....	48
3.4.1	<i>Testarea securizării funcționalitătilor</i>	48
3.4.2	<i>Testarea paginilor de autentificare.....</i>	50
3.6	PREZENTAREA PAGINILOR ȘI FUNCȚIONALITĂȚILOR PRINCIPALE.....	55
4	CONCLUZII	62
4.1	REZULTATE OBȚINUTE	62
4.2	DIRECȚII DE DEZVOLTARE	63
5	BIBLIOGRAFIE	64
6	ANEXE	66

1 Introducere

1.1 Context general

Una dintre concepțiile greșite despre începerea unui blog este că trebuie să fii un scriitor grozav pentru a avea succes. Ceva departe de adevăr. Oamenii citesc site-uri de blog pentru a obține o perspectivă personală asupra lucrurilor, aşa că majoritatea bloggerilor scriu într-un stil foarte informal și conversațional. Datorită formatului, mulți bloggeri de succes vor scrie despre o varietate de subiecte pe același blog.

În plus, nu trebuie să fii un expert în niciunul dintre subiectele despre care scrii pentru a avea un blog de succes. De exemplu, vizitatorii unui blog de gătit nu vor să citească un manual de la un om de știință alimentar, ei vor să audă experiențele cuiva care a gătit de fapt niște mese adevărate, greșeli și tot restul.

Problema actuală cu lumea blogging-ului este că fiecare autor sau utilizator care dorește să-și creeze o pagină pentru împărtășirea ideilor sale trebuie să parcurga mai mulți pași complicați pentru majoritatea utilizatorilor, cum ar fi: alegerea unui nume de pagină, căutarea platformei, înregistrarea site-ului și obținerea gazduirii, alegerea și modificarea şablonului gratuit de design sau crearea și modificarea uneia de la 0, promovarea paginii...

Soluția acestei probleme este crearea unei platforme centralizate, cu un design simplu, bine aranjat, și o interfata prietenoasa și ușoară de utilizat pentru toți utilizatorii. Aplicația are ca scop principal îmbunătățirea accesibilității atât pentru cititori cât și pentru autori, eficientizarea procesului de scrierea a articolelor cât și gestionarea și procesarea ulterioară a lor. Astfel, pentru a avea succes ca blogger, va fi nevoie într-adevăr doar de o singură cerință: o pasiune pentru subiectul tău.

În esență, blogging-ul înseamnă împărtășirea cunoștințelor tale cu lumea. A scrie despre lucruri care vă pasionează face procesul de începere a unui blog de succes mult mai ușor. Atât timp cât scrii despre lucruri care te interesează cu adevărat, pasiunea ta va străluci și va menține vizitatorii interesați fără a mai fi nevoie să pierzi timpul cu activități tehnice ca: indexarea articolelor, crearea paginilor, promovarea lor, etc...

1.2 Obiective

Primul dintre cele mai importante obiective reprezintă realizarea unei aplicații cu o interfață și un design sprietenos care să fie accesibil, clar și ușor de utilizat pentru orice utilizator.

Un alt scop este ca aplicația dezvoltată să fie în centrul său, interactivă și centralizată. Să permită oricui să se înregistreze, să posteze sau să interacționeze cu articolele altor autori, lasând comentarii. Sa nu aiba limitări de subiecte sau teme care pot fi abordate. Astfel indiferent de subiectul care îl interesează pe utilizator, să aibă posibilitatea să acceseze un singur site și doar să filtreze în dependență de

preferințe(categorii și autori) articolele pe care ar vrea să le vadă. Ca autor, să fie nevoie doar de înregistrare, după care să poată scri un articol iar ceilalți utilizatori să aibă posibilitatea de a interacționa cu ușurință cu el, de a-l citi, a da like sau dislike, a-l salva în contul sau pentru a avea acces ușor la el sau de a lăsa comentarii pentru a interacționa cu alții utilizatori interesați de articolul respectiv sau chiar cu autorul articolului. Aceasta este o modalitate bună de a intra în legătură cu oameni care sunt interesați de aceleași subiecte. Astfel aplicația va permite cititorului să învețe pe baza experiențele unui autor, dar totodată oferă și autorului posibilitatea de a învăța și de la cititorii săi.

Astfel, obiective de baza propuse sunt:

- Procesul de manageriere a articolelor cat mai simplu și rapid: un utilizator să poată crea/ edita/ șterge/ vizualiza un articol cu usurinta
- Efectuarea unei aplicații care va îndeplini cerințele publicului real.
- Design plăcut, prietenos și o interfata intuitiva
- Afisarea și actualizarea datelor în timp real, fără o reținere observabila pentru a nu crea neplăceri utilizatorilor
- Asigurarea functionalitatilor ce facilitează filtrarea, sortarea, căutarea și afisarea corespunzătoare a articolelor
- Stocarea și gestionarea eficientă a datelor intr-o bază de date bine organizată, fără date repetitive.
- Procesul de autentificare sau înregistrare cat mai simplu și rapid, logare cu email și parola pe baza unui token de autentificare (JWT), functionalitate confirmarea adresei de email.
- Recuperarea parolei prin intermediul adresei de email, cu token de restabilire.
- Număr vizite/citiri per articol
- Evaluarea articolelor prin atribuirea a două calitative, apreciere/dezapreciere.
- Modificare date profil: nume, prenume, email, descriere, poza de profil, link-uri către profilele rețelelor sociale.
- Împărțirea articolelor în 2 tipuri, publice și private, cele private fiind vizibile doar pentru utilizatorii autentificați.
- Implementarea statisticilor pentru utilizatorilor, fiecare utilizator va avea reprezentate diferite date ca: numărul de articole scrise, numerele de citiri, numărul de vizualizări și numărul de aprecieri ale articolelor scrise
- Implementarea functionalitatii de salvare a articolelor, fiecare utilizator va putea salva articolele dorite pentru a avea acces rapid la ele dintr-o pagina separată.
- Implementarea functionalitatii de comentarii pe articole, fiecare utilizator autentificat va putea scrie/ edita/ șterge comentarii către orice articol disponibil.

1.3 Specificații

Se dorește realizarea unei aplicații care simplifică procesul de creare/citire/editare/ștergere a articolelor precum și executarea altor functionalitati pe baza lor pentru a se crea un mediu plăcut și eficient atât pentru autori cât și pentru cititori.

Aplicația va avea 2 moduri de utilizarea importante. Unul pentru utilizatorii neautentificați și celălalt pentru utilizatorii autentificați. Astfel, ca utilizatorii neautentificați au posibilitatea de a intra pe site și a vizualiza toate articolele scrise și a le ordona după data publicării. Articolele vor fi divizate în 2 tipuri, private și publice, statut care-l poate atribui autorul la crearea sau modificarea sa. Utilizatorii neautentificați vor vedea doar o mică parte din conținutul articolelor private și vor avea un buton la dispoziție "Citește mai mult" care la apasare le va afișa o fereastră în care le vor fi prezentate avantajele detinerii unui cont pe platforma astfel motivându-i să se înregistreze.

Utilizatorii autentificați vor avea următoarele posibilități:

- Filtrarea articolelor după categorii și autori
- Sortarea articolelor după vizualizări sau numărul de citiri
- Vizualizarea tuturor autorilor și statisticile acestora
- Citirea completă a articolelor private
- Crearea și editarea articolelor
- Crearea, editarea și ștergerea comentariilor și vizualizarea comentariilor lăsate de alții utilizatori
- Salvarea articolelor din colecția personală pentru a le putea accesa mai ușor ulterior
- Evaluarea articolelor, atribuindu-le calificative ca "Like" sau "Dislike"

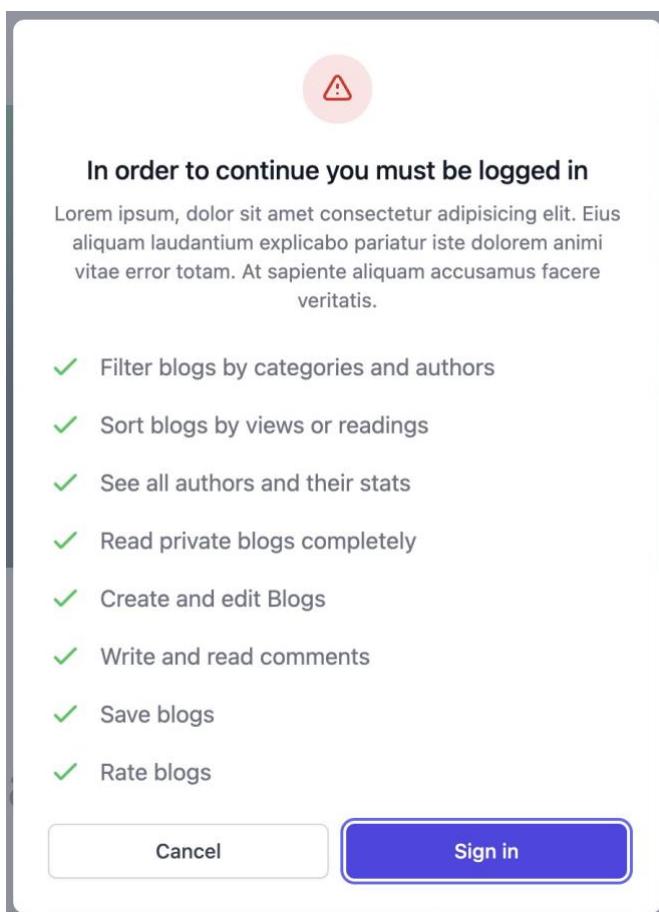


Figura 1.3.1 Fereastra de restricționare ce prezintă avantajele utilizatorilor autentificați

Aplicația va fi realizata folosind limbaje de programare, framework-uri ale limbajelor, pachete și tehnologii destinate realizarii aplicațiilor web.

Se va urmări faptul ca structura aplicației și a codului să fie cat mai clara și ușor de înțeles oricărui dezvoltator de aplicații, denumirea folderelor, numelor de fișiere, funcțiilor, variabilelor va fi cat mai sugestiva. Formatarea codului se va face astfel incat el să fie cat mai lizibil și se vor respecta recomandările ghidurilor stilistice a tehnologiile folosite. Respectand toate mentiuni, proiectul va fi realizat la un nivel cat mai înalt și profesionist, astfel facilitând dezvoltarea sa ulterioară.

2 Studiu bibliografic

2.1 Blog

Un blog (o versiune prescurtată a „weblog”) este un jurnal online sau un site web informativ care afișează informații în ordine cronologică inversă, cu cele mai recente postări apărând primele, [1] în partea de sus. Este o platformă în care un scriitor sau un grup de scriitori își împărtășesc opiniile asupra unui subiect individual.

2.1.1 Structura blogului

Aspectul blogurilor s-a schimbat de-a lungul timpului, iar în zilele noastre blogurile includ o mare varietate de articole și widget-uri. Cu toate acestea, majoritatea blogurilor includ încă unele caracteristici și structuri standard. [2]

Caracteristicile comune pe care le va include un blog tipic:

- Antet cu meniul sau bara de navigare.
- Zona principală de conținut cu postări evidențiate sau cele mai recente pe blog.
- Bară laterală cu profiluri sociale, conținut preferat sau îndemn.
- Subsol cu linkuri relevante, cum ar fi o declinare a răspunderii, politică de confidențialitate, pagina de contact etc.

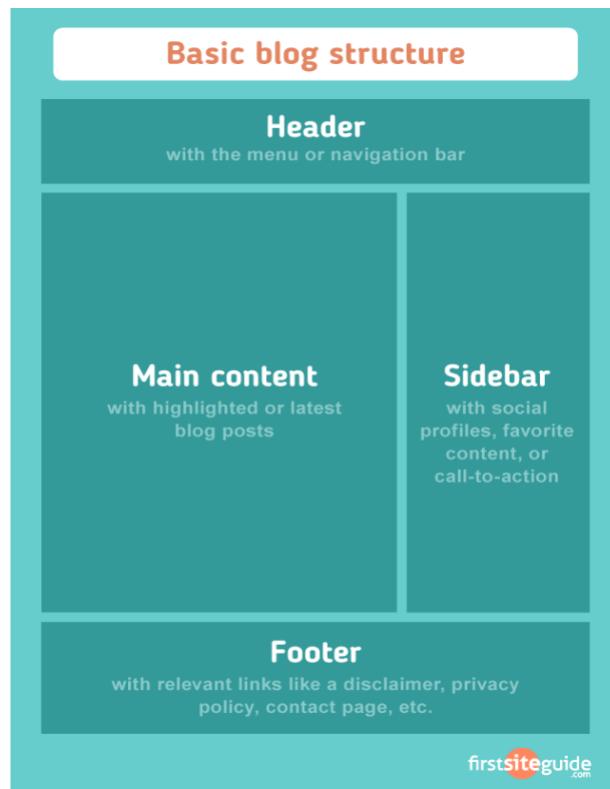


Figura 2.1.1 Structura unui blog [1]

2.2 Soluții existente

2.2.1 Aplicații similare

2.2.1.1 Medium

Este o platformă deschisă publicului în care cititorii pot găsi multiple articole ce dezvoltă diferite teme și idei,[4] este locul unde scriitorii de conținut profesioniști, dar totodată și vocile nedescoperite își pot împărtăși ideile despre orice subiect.

2.2.1.2 News Break

Este o revistă online de știri și actualitate publicată în Filipine. A început ca o revistă tipărită săptămânală care a fost publicată din 24 ianuarie 2001 până în 2006. [7] Site-ul web Newsbreak, lansat în 2006, funcționează acum ca componentă de investigare și cercetare a organizației de știri online Rappler.

2.2.1.3 Substack

Este o platformă online americană care oferă infrastructură de publicare, plată, analiză și proiectare pentru a sprijini buletinele informative cu abonament. [6] Le permite scriitorilor să trimită posturi informative digitale abonaților direct prin email. Fondată în 2017, Substack are sediul în San Francisco, California.

2.2.1.4 Vocal Media

Este o rețea de scris care îi ajută pe oameni câștigă din posturile lor printr-o metodă foarte simplă, doar creezi conținut și-l publici în rețeaua lor de milioane de cititori.[8] Au diferite comunități cărora le poți trimite direct scriurile tale. Rețeaua lor

este construită astfel încât să se poată câștiga bani scriind sau publicând propriile articole de blog.

Toate aceste aplicații existente prezentate anterior sunt bune și functionale, în schimb problema comună tuturor e ca ofera accesul doar pe baza unor subscrîptii contra cost, iar unele dintre ele se bazează mai mult doar pe anumite tipuri de conținut, ca buletine informative, știri sau sunt dedicate în special scriitorilor profesioniști care vor să genereze venit din articolele sale. [5] Ceea ce ne dorim de fapt prin realizarea acestei aplicații și reprezinta unul din principalele scopuri este să facilităm și să ajutăm autorii aflați la început de drum sau care au o pasiune în a scrie și a împărtăși cunoștințele personale cu alți utilizatori în scopul de a-i ajuta și a crea o comunitate puternica și inovatoare.

2.2.2 Tehnologii folosite

Toate soluțiile prezentate anterior sunt aplicații web de tip SPA(Single Page Application), partea de front-end al acestora este implementat folosind javascript, typescript dar și librării și framework-uri ale acestora, ca jQuery, Next.js, ReactJs, Angular etc.

Un **SPA** sau o aplicație cu o singura pagina, este o aplicație (sau un website) care interacționează cu browserul web prin rescrierea dinamica, a DOM-ului current, cu date noi generate fie din client, fie din server. În acest caz, browserul nu este nevoie să incarce o pagina nouă.

Atunci cand navigam pe aplicațiile respective, nu mergem niciodată pe o pagină nouă. Aplicațiile pur și simplu încarcă „pagina” într-un nou bloc div din container, setează pagina veche la proprietate „display: none” și actualizează adresa URL și titlul.

Fiecare dintre acele blocuri Html conține conținutul paginii. Iar când dorim să navigăm la pagina anterioară, Medium afișează pur și simplu vechiul div, în loc să creeze unul nou. De asemenea, pe „Medium” pare să existe un time-out pe paginile site-ului. După 5-10 minute, se șterg pentru a menține un DOM ordonat.

În general, execuția front-end ului pe aceste aplicații este la un nivel înalt și duce la o experiență de utilizator cu adevărat încântătoare.

La partea de server, aplicațiile principale sunt create folosind node.js(toate), express(medium, substack), nginx(newsbreak), Next.js(newsbreak, vocal.media), Golang, Python, Spark. [4]

Medium, folosește propriul cadru MVC numit Matador, pe care l-au deschis cu o perioadă în urmă: Medium/matador.

Iar la partea de bază de date, aplicațiile folosesc: MySQL, Redis și MongoDB(newsbreak), Amazon DynamoDB, S3 and Redshift(medium). [4]

Toate activele acestor aplicații sunt deservite prin CloudFlare. CloudFlare protejează și accelerează orice website online. Odată ce un website este o parte a comunității CloudFlare, traficul acestuia va trece prin intermediul infrastructurii acestei rețele, la nivel mondial. Aceasta optimizează timpul de răspuns al paginilor web, astfel încât vizitatorii să obțină cel mai bun timp posibil pentru încărcarea unei pagini web. De asemenea, Cloudflare blochează traficul roboților sau accesările abuzive pentru a nu irosi resursele serverului.

2.2.3 Funcții oferite

Câteva dintre funcțiile interesante ce le oferă utilizatorilor aplicațiile prezentate anterior.

2.2.3.1 Încorporarea (*Embedding*) - Medium

Există modalități mai bune de a prezenta o altă lucrare decât hyperlink-ul albastru subliniat. Videoclipurile YouTube și Vimeo, esențele GitHub, Tweeturile, Vines, rezumatele campaniei Kickstarter, fotografiile Instagram și înregistrările SoundCloud și Rd.io sunt toate încorporabile personalizat.

Puteți chiar să încorporați alte povești Medium care fac parte dintr-o serie sau să se potrivească bine cu o altă poveste. De exemplu, atunci când oamenii cumpără The Ultimate Guide To Medium — o carte electronică publicată chiar pe Medium — li se trimitе un link către un cuprins, care nu este altceva decât o poveste Medium cu cele 9 capitole încorporate în ordine.[9]



Figura 2.2.3.1.1 Încorporarea articolelor în Medium[9]

2.2.3.2 Etichete (Tags)

Pentru fiecare poveste pe care o publicați, puteți selecta până la 3 etichete. Etichetele vor ajuta un public interesat să vă descopere povestea și să îmbunătățească clasarea rezultatelor căutării în Medium. [9]

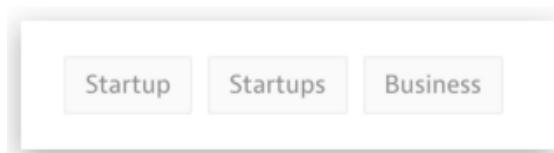


Figura 2.2.3.2.1 Etichetele articolelor în Medium[9]

Puteți reveni oricând la poveștile publicate pentru a vă actualiza sau a elimina etichetele. Folosiți etichetele relevante și veți crește probabilitatea ca cititorii să vă găsească povestea. [9]

2.2.3.3 Evidențiere (Highlights)

Este o mică funcție grozavă pentru cititori, dar și scriitorii pot folosi această funcție în avantajul lor.

Utilizați evidențierea ca o altă modalitate de a adăuga accent unui anumit text din propria poveste.[8]



Figura 2.2.3.3.1 Evidențierea textului în Medium[9]

2.2.3.4 Partajarea schițelor (Sharing Drafts)

După cum se spune, două capete sunt mai bune decât unul. Înainte de a publica următoarea povestire, trimiteți versiunea nefinalizată unui prieten sau coleg pentru o privire rapidă. În cel mai rău caz - vor prinde câteva greșeli de scriere. Cel mai bun caz - vor lumina unele puncte aspre din postarea dvs. care ar putea necesita puțină lustruire.

Partajați linkul specific schiței prin e-mail, Twitter, Facebook, iMessage etc. Oricine are linkul vă poate vizualiza schița - nu este necesar un cont Medium - dar cei care sunt autentificați pot lăsa comentarii. [9]

2.2.3.5 Carduri sociale (Social Cards)

Distribuirea pe rețelele sociale este o funcționalitate inevitabilă într-o aplicație de acest gen și este probabil o cauză principală a virilității. Poveștile sunt distribuite pe Twitter și Facebook folosind „carduri”, care sunt rezumate scurte care includ un titlu, teaser și o versiune în miniatură a imaginii de copertă. [9]

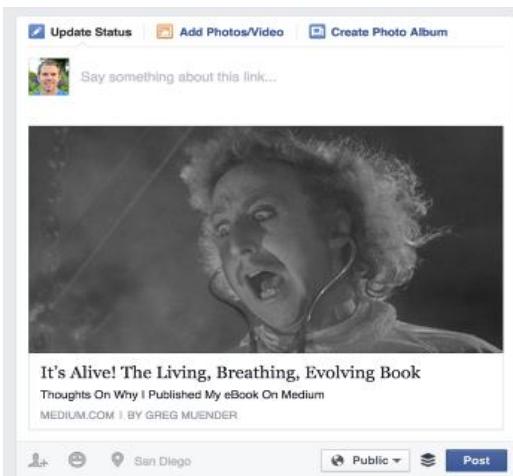


Figura 2.2.3.5.1 Carduri sociale de distribuire a articolelor[9]

2.2.3.6 Comentarii (Notes)

Lăsați comentarii despre propria poveste pentru a clarifica un punct, pentru a oferi resurse suplimentare sau pentru a înștiința o actualizare sau o modificare a textului original. Lăsați note care pun întrebări și încurajează o conversație sau o dezbatere între cititori. Făcând acest lucru, povestea va fi mult mai vie și se va crea o relație mai puternică între creatori și publicul acestora. [9]

2.2.3.7 WYSIWYG – HTML Editor

WYSIWYG este un instrument de editare a conținutului. Abrevierea oarecum criptată vine de la fraza în engleză „What You See Is What You Get”. În editorul WYSIWYG, conținutul editat, fie text sau grafică, apare într-o formă apropiată de produsul final. Deci, în loc să scrieți manual codul sursă, aveți de-a face cu un editor de text îmbogațit convenabil în care manipulați elementele de design. Editorul WYSIWYG vă permite să vedeți exact cum va arăta. Editorul HTML WYSIWYG este un instrument extrem de util pentru a oferi clienților posibilitatea de a-și edita și actualiza site-ul web fără a-i deranja pe programatori. [9]

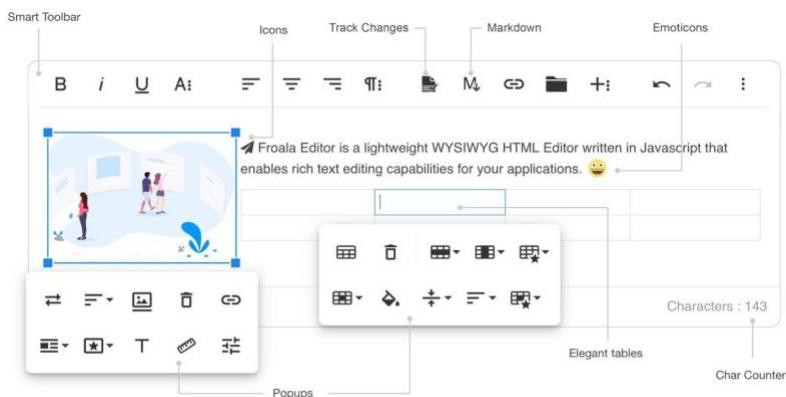


Figura 2.2.3.7.1 Editor HTML în aplicație WEB [9]

2.3 Protocolul HTTP

HTTP este un protocol pentru transferul de informații pe Internet, care înseamnă HyperText Transfer Protocol. De exemplu, browserul trimite o singură cerere către server, care la rândul său o prelucrează, generează un răspuns și partajează acest răspuns cu browserul - resurse sub formă de date.

Există două roluri diferite în protocolul HTTP: server și client. Cererea este întotdeauna inițiată de client, iar serverul îi răspunde. Clientul poate fi atât un browser, cât și, de exemplu, un robot de căutare care vizualizează pagini de pe Internet și le indexează în funcție de relevanța unei interogări cheie. HTTP este bazat pe text - mesajele dintre un client și un server sunt în esență fragmente de text, deși pot exista și alte elemente în corpul mesajului: video, fotografie, audio etc. [10]

Fiecare cerere individuală este trimisă către server, care o procesează și oferă un răspuns. Există multe obiecte între client și server, care se numesc proxy. Acestea oferă

diferite niveluri de funcționalitate, securitate și confidențialitate, în funcție de nevoile dvs. sau de politică companiei.

HTTP conține trei elemente principale:

- **Client:** este orice instrument care acționează în numele unui utilizator. Acest rol este îndeplinit în mare parte de browserul web, dar în afară de browser, acestea sunt programe folosite de ingineri sau dezvoltatori web pentru a-și depana aplicațiile. Clientul inițiază întotdeauna cererea, serverul nu o face niciodată.
- **Server:** Pe cealaltă parte a canalului de comunicație se află un server care servește documentul la cererea clientului. Deși serverul arată ca o singură mașină virtuală pentru utilizator, poate fi de fapt o colecție de servere care împart sarcina. Pe de altă parte, mai multe servere pot fi amplasate pe aceeași mașină. Cu HTTP/1.1 și un antet Host, ei pot folosi chiar și aceeași adresă IP.
- **Server proxy:** Proxy-urile sunt servere, computere sau alte mașini de nivel de aplicație care se află între dispozitivul client și serverul însuși. Ele transmit cereri și răspunsuri HTTP. De obicei, unul sau mai mulți proxy sunt utilizați pentru fiecare interacțiune client-server. [11]

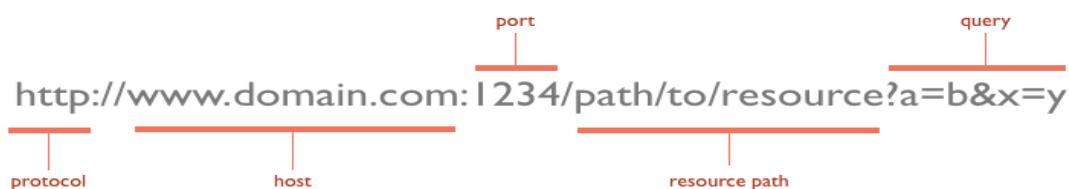


Figura 2.3.1 Protocolul HTTP

Folosind URL-ul, se poate determina domeniul cu care se dorește efectuarea comunicării, iar acțiunea se poate determina folosind metoda HTTP a cererii.

Cele mai des întâlnite și utilizate metode HTTP:

- **GET:** accesează și returnează datele unei înregistrări existente. E indicat ca adresa URL să includă informațiile necesare pentru ca serverul să poată găsi înregistrarea dorită. Solicitările GET nu pot avea un corp de date. Totuși ele se pot trimite către server folosind parametrii URL. În acest caz, există o limită de dimensiunea maximă a URL-ului, care este de aproximativ 2000 de caractere (depinde de browser).
- **POST:** este folosită pentru a crea sau adăuga o resursă pe server. De obicei, cererea POST adaugă o nouă resursă la server. Spre deosebire de solicitările GET și HEAD, cererile POST pot schimba starea serverului.
- **PUT:** actualizează o înregistrare deja existentă. Cererile de tip PUT pot conține datele care urmează a fi actualizate.

- **DELETE:** se folosește pentru a șterge o înregistrare din server. La fel ca metoda post, poate schimba starea serverului [12]

* În anumite cazuri, solicitările de tip PUT și DELETE pot fi trimise ca cerere POST, iar în conținutul de date se poate indica acțiunea care trebuie efectuată (crearea, actualizarea sau ștergerea) și resursa. Totodată, id-ul sau alta caracteristică care o poate diferenția resursa asupra careia urmează să se execute acțiunea față de alte înregistrări poate fi inclusa ca query parameter în url-ul cererii.

2.3.1 Diferența între HTTP și HTTPS

HTTPS (HyperText Transfer Protocol Secure) este același cu HTTP, dar adaugă criptarea datelor transmise folosind protocole criptografice precum SSL sau TLS. Serverul găzduiește un certificat special și o cheie privată, care sunt utilizate pentru a autentifica, cripta și decriptă datele.

Dacă explicați cu propriile cuvinte și nu intrați în detaliile tehnice ale creării „secretelor de sesiune” în timpul „străngerilor de mâna” ale clientului și serverului, atunci transferul de date prin HTTPS are loc după cum urmează: atunci când vizitați un site web, browserul face o solicitarea serverului de a obține informații despre certificat; după ce primește o copie a certificatului SSL cu o cheie de criptare specială, verifică autenticitatea certificatului. Dacă verificarea are succes, atunci folosind cheia primită, criptează și trimit datele către server, unde sunt decriptate. Cheia este valabilă numai pentru sesiunea curentă și este distrusă după ce se încheie.

HTTP transmite datele într-o formă „brută” fără a le cripta în niciun fel, prin urmare, prin interceptarea lor, un atacator poate intra în posesia datelor confidențiale, iar dacă acestea sunt criptate, atunci chiar și interceptând datele fără a avea o cheie, va fi aproape imposibil de decriptat. HTTP utilizează implicit portul TCP 80, iar HTTPS utilizează 443. [13]

2.4 Aplicație cu o singură pagină (SPA)

O aplicație cu o singură pagină (SPA) este un termen general larg pentru aplicațiile care sunt redate atunci când clientul le solicită. SPA-urile sunt structurate ca o singură pagină HTML care nu are conținut preîncărcat. Conținutul este încărcat prin fișiere Javascript pentru întreaga aplicație și găzduit într-o singură pagină HTML. Fișierele Javascript găzduiesc toate datele referitoare la logica aplicației, interfața de utilizare și comunicarea cu serverul. Cadrele și bibliotecile Javascript populare pentru construirea de SPA includ toți suspecții obișnuiți ai React, AngularJS, Vue.js, Ember.js și Svelte, printre alții.

Când utilizatorii navează prin diferitele părți ale SPA, nu va exista niciun timp suplimentar de încărcare între diferitele elemente ale aplicației. SSG-urile pot intra și ele în această categorie odată ce au fost încărcate în browser. Deoarece totul este încărcat pe partea clientului, echipele trebuie să țină cont de o gamă largă de clienți, asigurând în

același timp o experiență de utilizator rapidă și fără întreruperi. Cu cadrele moderne, divizarea codului permite încărcarea unor elemente la cerere, ceea ce poate ajuta la eliminarea acestei probleme. [14]

2.4.1 Avantaje

- În timp ce încărcarea inițială poate mai lungă, odată ce aplicația s-a încărcat complet, nu va fi necesară încărcare suplimentară.
- Alegere bună pentru experiențele dinamice în care echipele au nevoie de un sentiment personalizat pentru experiența lor de utilizator
- Echipele au foarte mult control asupra arhitecturilor lor și pot folosi cadre web moderne [15]
- Poate fi folosit în tandem cu alte tehnologii

2.4.2 Dezavantaje

Pe măsură ce aplicația crește în dimensiune și complexitate, aceasta poate afecta grav timpul inițial de încărcare, ceea ce poate duce la o deteriorare a experienței utilizatorului.

- Menținerea unui SEO bun este aproape imposibilă din cauza timpilor de încărcare și a lipsei conținutului inițial pe HTML
- Fișierele mari pentru aplicații web complexe pot deveni dificil de întreținut și organizat
- Provocările legate de abordarea SPA necesită soluții care pot fi costisitoare și consumatoare de timp [16]

2.5 HTML

2.5.1 HTML - definitie

HTML este un acronim care înseamnă **Hyper Text Markup Language**, care este folosit pentru a crea pagini web și aplicații web.

Hypertext: înseamnă pur și simplu „Text în text”. Un text are o legătură în el, este un hipertext. De fiecare dată când faceți clic pe un link care vă duce la o nouă pagină web, ați făcut clic pe un hipertext. HyperText este o modalitate de a lega două sau mai multe pagini web (documente HTML) între ele.

Limbajul de marcare: un limbaj de marcare este un limbaj de computer care este utilizat pentru a aplica convențiile de aspect și formatare unui document text. Limbajul de marcare face textul mai interactiv și mai dinamic. Poate transforma textul în imagini, tabele, link-uri etc. [17]

Pagina web: o pagină web este un document care este scris în mod obișnuit în HTML și tradus de un browser web. O pagină web poate fi identificată prin introducerea unei adrese URL. O pagină Web poate fi de tip static sau dinamic. Numai cu ajutorul HTML, putem crea pagini web statice.

Prin urmare, HTML este un limbaj de marcare care este folosit pentru a crea pagini web atractive cu ajutorul stilului și care arată într-un format frumos pe un browser web. Un document HTML este format din multe etichete HTML și fiecare etichetă HTML conține conținut diferit. [18]

2.5.2 Exemplu pagina HTML

```
<html>
  <head>
    <title>Titlu pagina html</title>
  </head>
  <body>
    <h1>Titlul din pagina</h1>
  </body>
</html>
```

Figura 2.5.2.1 Exemplu pagina HTML [19]

<!DOCTYPE>: definește tipul documentului sau informează browserul despre versiunea HTML.

<html>: Această etichetă informează browserul că este un document HTML. Textul dintre eticheta html descrie documentul web. Este un container pentru toate celelalte elemente ale HTML, cu excepția <!DOCTYPE>

<head>: ar trebui să fie primul element din interiorul elementului <html>, care conține metadatele (informații despre document). Trebuie să fie închis înainte ca eticheta pentru corp să se deschidă.

<title>: După cum sugerează și numele, este folosit pentru a adăuga titlul acelei pagini HTML care apare în partea de sus a ferestrei browserului. Trebuie plasat în interiorul etichetei de cap și ar trebui să se închidă imediat. (Optional)

<body> : textul dintre eticheta body descrie conținutul corpului paginii care este vizibil pentru utilizatorul final. Această etichetă conține conținutul principal al documentului HTML.

<h1> : Textul dintre eticheta <h1> descrie titlul de prim nivel al paginii web.

<p> : textul dintre eticheta <p> descrie paragraful paginii web. [20]

2.5.3 Caracteristicile HTML

- Este un limbaj foarte ușor și simplu . Poate fi ușor de înțeles și modificat.
- Este foarte ușor să faci o prezentare eficientă cu HTML, deoarece are o mulțime de etichete de formatare.
- Este un limbaj de marcăre , deci oferă o modalitate flexibilă de a proiecta pagini web împreună cu textul.
- Facilitează programatorilor să adauge un link pe paginile web (prin html anchor tag), astfel încât sporește interesul de navigare al utilizatorului.
- Este independent de platformă, deoarece poate fi afișat pe orice platformă precum Windows, Linux și Macintosh etc.
- Facilitează programatorului să adauge grafică, videoclipuri și sunet la paginile web, ceea ce îl face mai atractiv și mai interactiv.
- HTML este un limbaj care nu ține seama de majuscule, ceea ce înseamnă că putem folosi etichete fie cu litere mici, fie cu litere mari. [18]

2.6 CSS (Cascading Style Sheets)

Este un limbaj de design simplu menit să simplifice procesul de a face paginile web prezentabile.

Folosind CSS, puteți controla culoarea textului, stilul fonturilor, distanța dintre paragrafe, modul în care sunt dimensionate și aranjate coloanele, ce imagini sau culori de fundal sunt folosite, modelele de aspect, variațiile de afișare pentru diferite dispozitive și dimensiuni de ecran precum și o varietate de alte efecte.

CSS este ușor de învățat și de înțeles, dar oferă un control puternic asupra prezentării unui document HTML. Cel mai frecvent, CSS este combinat cu limbajele de marcăre HTML sau XHTML. [20]

2.6.1 Avantajele CSS

CSS economisește timp – Stilizările CSS pot fi scrise odată și apoi se poate reutiliza aceeași clasa în mai multe pagini HTML. Se pot defini stilizări pentru fiecare element HTML iar ulterior se pot aplica pe orice pagina.

Paginile se încarcă mai repede - Se scrie doar o singura data stilizarea css și se atribuie unei etichete. Deci, rezultă că avem mai puțin cod, respectiv timpi mai rapizi de descărcare.

Întreținere ușoară - Pentru a face o schimbare globală, pur și simplu se schimba stilul etichetei respective și toate elementele din toate paginile web vor fi actualizate automat.

Compatibilitate cu mai multe dispozitive - Foile de stil permit optimizarea conținutului pentru mai multe tipuri de dispozitive. Folosind același document HTML, pot fi prezentate versiuni diferite ale unui site web pentru diferite dispozitive cum ar fi PDA-uri și telefoane mobile sau pentru imprimare.

Standarde web globale - Acum atributele HTML sunt depreciate și se recomandă utilizarea CSS. Deci, este o idee bună să se folosească CSS în toate paginile HTML pentru a le face compatibile cu viitoarele browsere. [21]

2.7 JavaScript

JavaScript este un limbaj de programare bazat pe text, folosit atât pe partea client, cât și pe partea serverului, care permite crearea de pagini web interactive. În cazul în care HTML și CSS sunt limbaje care conferă structură și stil paginilor web, JavaScript oferă paginilor web elemente interactive care implică un utilizator.[22] Exemplele obișnuite de JavaScript care sunt întâlnite în viața de zi cu zi includ caseta de căutare de pe Google, un videoclip rezumat de știri încorporat în YouTube sau reîmprospătarea feedului pe FaceBook.

Încorporarea JavaScript îmbunătățește experiența utilizatorului a paginii web prin conversia acesteia dintr-o pagină statică într-o interactivă.[23]

2.7.1 La ce se foloseste JavaScript

JavaScript este utilizat în principal pentru aplicații bazate pe web și browsere web. Totodată, JavaScript este folosit și dincolo de Web în software-ul, serverele și controalele hardware încorporate. Câteva lucruri de bază pentru care este folosit JavaScript:

- **Adăugarea comportamentului interactiv paginilor web**

JavaScript permite utilizatorilor să interacționeze cu paginile web. Aproape că nu există limite pentru lucrurile pe care le puteți face cu JavaScript pe o pagină web – acestea sunt doar câteva exemple[25]:

- Afisarea sau ascunderea informațiilor cu un clic pe un buton
- Schimbarea culorii unui buton când mouse-ul trece peste el
- Glisarea printr-un carusel de imagini
- Mărirea sau micșorarea unei imagini
- Afisarea unui cronometru sau numărătoare inversă pe un site web
- Redarea audio și video într-o pagină web
- Afisarea animațiilor

- **Construirea de servere web și dezvoltarea aplicațiilor server**

Dincolo de site-uri web și aplicații, dezvoltatorii pot folosi JavaScript și pentru a construi servere web simple și pentru a dezvolta infrastructura back-end folosind Node.js.

2.7.2 Avantajele utilizării JavaScript

Pe lângă posibilitățile nelimitate, există multe motive pentru care dezvoltatorii web să folosească JavaScript peste alte limbaje de programare[24]:

- Este singurul limbaj de programare nativ pentru browserul web
- Este cel mai popular limbaj
- Există un prag scăzut pentru a începe
- Este o limbă distractiv de învățat

2.8 Vue.JS

VueJS este un framework progresiv al limbajului JavaScript, este open source și este utilizat pentru a dezvolta interfețe web interactive. Este unul dintre cele mai mari framework-uri folosite pentru a simplifica dezvoltarea web. VueJS se concentrează pe stratul de vizualizare.[25] Poate fi integrat cu ușurință în proiecte mari pentru dezvoltarea părții de front-end.

2.8.1 Caracteristici Vue.JS:

DOM virtual: VueJS folosește DOM virtual, care de altfel este folosit și de alte framework-uri precum React, Ember etc. Modificările nu sunt făcute în DOM, în schimb este creată o copie a DOM-ului care este salvată sub formă de structuri de date JavaScript. . Ori de câte ori sunt făcute modificări, acestea sunt executate asupra structurilor de date JavaScript, iar acestea din urmă sunt comparate cu structura de date inițială.[26] Modificările finale sunt apoi actualizate la DOM-ul real. Acest lucru este bun în ceea ce privește optimizarea, este mai puțin costisitor și modificările pot fi făcute într-un ritm mai rapid.

- **Transmiterea datelor:** în Vue are loc atribuirea de valori prin intermediul atributelor HTML, schimbarea stilului, atribuirea claselor cu ajutorul directivei de legare numită v-bind.
- **Componentele :**sunt una dintre caracteristicile importante ale VueJS care ajută la crearea elementelor personalizate, și care pot fi ulterior reutilizate în HTML.

Gestionarea evenimentelor: v-on este atributul adăugat elementelor DOM pentru a urmări evenimentele în VueJS.

- **Animație/Tranzitie:** VueJS oferă diferite moduri de aplicare a tranzițiilor la elementele HTML atunci când acestea sunt adăugate/actualizate sau eliminate din DOM. VueJS are o componentă de tranzitie încorporată care trebuie să fie înfășurată în jurul elementului pentru efectul de tranzitie. [27] Totodată se pot adăuga cu ușurință biblioteci de animație străine, dar și mai multă interactivitate la interfață.
- **Proprietățile „Computed”:** este una dintre caracteristicile importante ale VueJS-ului. Ajută să urmărească modificările aduse elementelor UI și să efectueze calculele necesar, fără a fi nevoie de codare suplimentară pentru aceasta.

- **Şabloane („Templates”):** VueJS oferă şabloane bazate pe HTML care leagă DOM-ul cu datele instanței Vue. Vue compilează şabloanele în funcții virtuale „DOM Render”.
- **Directive:** VueJS are directive încorporate, cum ar fi v-if, v-else, v-show, v-on, v-bind și v-model, care sunt folosite pentru a efectua diverse acțiuni pe front-end.
- **Watchers:** sunt funcții ce se aplică odată cu modificarea unor date. De exemplu, elementele de intrare din formular. Aici, nu trebuie să adăugăm niciun eveniment suplimentar.[28] Watcher se ocupă de gestionarea oricărora modificări ale datelor, făcând codul simplu și rapid.
- **Navigarea:** navigarea între pagini se realizează cu ajutorul vue router-ului.
- **Rapiditate:** este foarte ușor și performanța sa este de asemenea una foarte mare.

2.8.2 Cine îl folosește

Din ce în ce mai multe companii încep să folosească VueJS pentru a-și construi interfețele de utilizator front-end și site-urile web. Nintendo, Louis Vuitton, Adobe, BMW, Upwork, Alibaba, Gitlab și Xiaomi sunt toți utilizatori VueJS. Chiar și Google și-a construit platforma “Careers” bazată pe VueJS și nu pe cadrul lor nativ Angular, iar Apple și-a construit site-ul web de tutoriale cu VueJS.

VueJS a cunoscut această creștere a popularității, deoarece este al naibii de ușor pentru dezvoltatori să adapteze și să implementeze elementele de bază de care fiecare aplicație sau site web le necesită pentru a funcționa ca interfață de utilizator.

Gradul de dificultate pentru a începe lucrul în cadrul VueJS este minim, iar cei familiarizați cu limbajele de codare HTML, CSS și JavaScript de bază pot sări literalmente chiar în capăt fără prea multă grijă cu privire la încul în bazin.

Principalul avantaj este că uneltele și suportul se vor îmbunătăți doar cu o utilizare mai răspândită, iar faptul că Vue se poate rula cu băieții mari precum React și Angular, în timp ce nu au investiții uriașe, arată cât de capabil este cadrul și că poate eclipsa concurenții săi în următorii ani.

2.9 Laravel

Laravel este un framework PHP open-source, care este robust și ușor de înțeles. Urmează un model de design MVC (model-vedere-controler). Laravel reutilizează componente existente ale diferitelor cadre, ceea ce ajută la crearea unei aplicații web. Aplicația web astfel concepută este mai structurată și mai pragmatică.

Laravel oferă un set bogat de funcționalități care încorporează caracteristicile de bază ale cadrelor PHP precum CodeIgniter, Yii și alte limbaje de programare precum Ruby on Rails.[29] Laravel are un set foarte bogat de caracteristici care vor crește viteza dezvoltării web.

Dacă sunteți familiarizat cu Core PHP și Advanced PHP, Laravel vă va ușura sarcina. Economisește mult timp dacă intenționați să dezvoltați un site web de la zero. Mai mult, un site web construit în Laravel este sigur și previne mai multe atacuri web.

2.9.1 Avantajele Laravel

Laravel oferă următoarele avantaje, la proiectarea unei aplicații cu ajutorul său.

- Aplicația web devine mai scalabilă, datorită cadrului Laravel.
- Se economisește timp considerabil în proiectarea aplicației web, deoarece Laravel reutiliza componentele din alt framework în dezvoltarea aplicației web.
- Include spații de nume și interfețe, astfel ajută la organizarea și gestionarea resurselor.
- **Composer:** este un instrument care include toate dependențele și bibliotecile. Permite unui utilizator să creeze un proiect în raport cu cadrul menționat (de exemplu, cele utilizate în instalarea Laravel).[30] Bibliotecile terțe pot fi instalate cu ușurință cu ajutorul compozitorului. Toate dependențele sunt notate în fișierul composer.json care este plasat în folderul sursă.
- **Artisan:** Interfața de linie de comandă folosită în Laravel se numește Artisan. Include un set de comenzi care ajută la construirea unei aplicații web. Aceste comenzi sunt încorporate din cadrul Symphony.

Laravel oferă următoarele caracteristici cheie, ceea ce îl face o alegere ideală pentru proiectarea aplicațiilor web:

- **Modularitate:** ofera peste 20 de biblioteci și module integrate care ajută la îmbunătățirea aplicației. Fiecare modul este integrat cu managerul de dependențe Composer, care ușurează actualizările.
- **Testabilitate:** Laravel include funcții și ajutoare care ajută la testarea prin diferite cazuri de testare. Această caracteristică ajută la menținerea codului conform cerințelor.
- **Dirijare:** Laravel oferă utilizatorului o abordare flexibilă pentru a defini rutele în aplicația web. Rutarea ajută la scalarea aplicației într-un mod mai bun și crește performanța acesteia.
- **Managementul configurației:** O aplicație web proiectată în Laravel va rula în diferite medii, ceea ce înseamnă că va exista o schimbare constantă în configurația sa.[31] Laravel oferă o abordare consecventă pentru a gestiona configurația într-un mod eficient.
- **Generator de interogări și ORM:** Laravel încorporează un generator de interogări care ajută la interogarea bazelor de date folosind diferite metode simple în lanț. Oferă implementare ORM (Object Relational Mapper) și ActiveRecord numită Eloquent.
- **Generator de scheme:** Schema Builder menține definițiile și schema bazei de date în cod PHP. De asemenea, menține o evidență a modificărilor în ceea ce privește migrarea bazelor de date.
- **Motor de şabloane:** Laravel folosește motorul Blade Template, un limbaj de şablon ușor folosit pentru a proiecta blocuri ierarhice și machete cu blocuri predefinite care includ conținut dinamic.

- **E-mail:** Laravel include o clasă de e-mail care ajută la trimiterea de e-mailuri cu conținut bogat și atașamente din aplicația web.
- **Autentificare:** Autentificarea utilizatorului este o caracteristică comună în aplicațiile web.[31] Laravel ușurează proiectarea autentificării, deoarece include funcții precum înregistrarea, parola uitată și trimiterea mementourilor de parolă.
- **Redis:** Laravel folosește Redis pentru a se conecta la o sesiune existentă și la un cache de uz general. Redis interacționează direct cu sesiunea.
- **Cozile:** Laravel include servicii de coadă, cum ar fi trimiterea prin e-mail a unui număr mare de utilizatori sau un anumit job Cron.[31] Aceste cozi ajută la finalizarea sarcinilor într-un mod mai ușor, fără a aștepta ca sarcina anterioară să fie finalizată.
- **Command Bus:** Laravel include Command Bus care ajută la executarea comenziilor și la expedierea evenimentelor într-un mod simplu. Comenzile din Laravel acționează conform ciclului de viață al aplicației.

2.10 MySQL

MySQL este un sistem de management al bazelor de date relaționale (DBMS) care este distribuit ca software gratuit. Este unul dintre cele mai populare, deoarece este flexibil, ușor, ușor de utilizat.

Cuvântul „relațional” înseamnă că bazele de date sunt prezentate ca informații conexe și sunt descrise ca un set de relații. MySQL funcționează cu limbajul de interogare SQL , care este folosit în mod tradițional în bazele de date.

2.10.1 Cum funcționează baza de date MySQL

MySQL are o arhitectură client-server. Aceasta înseamnă că baza de date este stocată într-o singură sursă - pe server. și clienții - dispozitivele terțe pot comunica cu acesta. Clienții trimit interogări la baza de date și apoi primesc informații de la server.

Site-urile funcționează aproximativ în acest fel: în partea lor „externă”, vizibilă, este posibilă trimiterea unei cereri către server. Acesta este orice formular de trimitere: introducerea contului dvs. personal, postarea unui comentariu sau căutarea pe site.

Arhitectura client-server face stocarea datelor mai sigură: computerele client nu pot obține acces necontrolat la ele. Aceștia deschid doar o parte din datele care pot fi obținute la cerere. Toate informațiile se află pe server, iar clienții nu sunt supraîncărcați, deci nu au nevoie de putere de calcul mare.

Expresia „server MySQL” este comună, ceea ce înseamnă doar o bază de date situată pe server sub controlul acestui DBMS.

2.10.2 Cum funcționează relațiile cu bazele de date

Relațiile dintre tabele arată cum unele date pot depinde de altele. În funcție de modul în care sunt configurate relațiile, baza de date poate obține rezultate diferite și poate căuta date. O conexiune poate implica o înregistrare sau mai multe simultan.

Unu la unu (One to one) Aceasta este cel mai simplu tip de relație, care spune: o înregistrare din acest tabel corespunde doar unei înregistrări dintr-un alt tabel. Dacă facem un tabel nou cu fotografii ale clienților, atunci fiecărei fotografii îi va corespunde un singur client și invers.

Unu la mulți (One to many) Când avem o masă cu clienții și o masă cu achizițiile lor, aici funcționează o relație unu-la-mulți. Aceasta înseamnă că avem mai multe înregistrări despre achizițiile lor corespunzătoare unei singure evidențe de client, de exemplu, dacă le-a făcut în momente diferite. Datorită acestei conexiuni, putem afișa toate achizițiile pentru fiecare client în mod individual.

Un alt exemplu sunt artiștii și picturile. Fiecare pictură aparține unui singur artist, dar un artist poate deține multe picturi diferite.

Mulți la mulți (Many to many). Aceasta este o conexiune dificilă - are nevoie de un tabel separat. Sensul este acesta: există un tabel intermediar în care sunt combinate datele dintr-un tabel cu datele dintr-un altul. În noul tabel, așa cum ar fi, nu există date - are doar adrese.

2.10.3 Exemplu de comenzi MySQL:

Să creăm baza de date THOUGHTS_STORE:

```
CREATE DATABASE THOUGHTS_STORE;
```

Să spunem că vom continua să lucrăm cu această bază specială:

```
USE THOUGHTS_STORE;
```

Să creăm un tabel cu titlurile articolelor, autorilor și numărul de citiri pe lună:

```
CREATE TABLE THOUGHTS (name VARCHAR(200), author VARCHAR(20), readers INT);
```

Să încărcăm date gata făcute dintr-un fișier în tabel:

```
LOAD DATA LOCAL INFILE 'thoughts/readers.txt' INTO TABLE THOUGHTS;
```

Acum să le afișăm pe ecran:

```
SELECT * FROM THOUGHTS;
```

Alte exemple:

- Adăugarea unei înregistrări: `insert into mytable set fld1='val1', fld2='val2';`
- Ștergerea unei înregistrări: `delete from mytable where fld1='val1';`
- Editarea unei înregistrări: `update mytable set fld1='val3' where fld2='val2';`

3 Analiză, proiectare, implementare

3.1 Prezentarea generală

Aplicația are ca scop principal îmbunătățirea procesului de creare, manipulare și promovare a articolelor autorilor mici și mijlocii, care vor să-și promoveze articolele, să crească într-o comunitate sau să ajute alte persoane prin împărțirea cunoștințelor, ideilor sau experiențelor personale. Prin aceasta aplicație se urmărește îmbunătățirea accesibilității atât pentru cititori cat și pentru autori, eficientizarea procesului de scrierea a articolelor cat și gestionarea și procesarea ulterioară a lor. Astfel, pentru a avea succes ca blogger, va fi nevoie într-adevăr doar de o singură cerință: o pasiune pentru subiectul tău.

Problema actuală cu lumea blogging-ului este ca fiecare autor sau utilizator care dorește să-și creeze o pagină pentru împărtășirea ideilor sale trebuie să parcurga mai mulți pași complicați pentru majoritatea utilizatorilor, cum ar fi: alegerea unui nume de pagină, căutarea platformei, înregistrarea site-ului, obținerea gazduirei, alegerea și modificarea şablonului gratuit de design sau crearea și modificarea unuia de la 0, promovarea paginii, etc.

Soluția acestei probleme este crearea unei platforme centralizate, cu un design simplu, bine aranjat, o interfață prietenoasă și ușoară de utilizat pentru toți utilizatorii.

Aplicația va purta numele de “ThoughtStore”, acesta fiind sugestiv, ușor de memorat și reprezintă în totalitate ideea pe care o promovează proiectul, în traducere liberă “Depozit de Ganduri”.

Designul și interfața prin intermediul căreia interacționează utilizatorii cu aplicația s-a întîrit să fie cat mai accesibile și intuitive atât pentru autorii articolelor cat și pentru cititori. S-a încercat realizarea unui design într-o gama de culori optime și confortabile la vizualizare. Pe partea de funcționalitate s-a dorit crearea unei structuri cat mai modulară, clară și scrisă corect în conformitate cu convențiile programării. Astfel ca atât utilizatorii cat și programatorii care ar putea să participe la dezvoltarea ulterioară proiectului să aibă parte de o experiență plăcută.

Totodată, s-a pus accent și pe partea de securitate, astfel s-a restrictionat accesul public pentru paginile care sunt destinate doar utilizatorilor autentificați.

În realizarea aplicației s-au folosit cele mai recente și actuale tehnologii, limbi de programare, pachete și framework-uri ale limbajelor. Componentele și interfața au fost create de la 0 sau personalizate astfel încât să corespundă cerințelor aplicației. Pentru realizarea obiectivelor propuse a fost necesara cunoașterea și studierea diferitor tehnologii noi și modul lor de utilizare.

Întregul proiect este alcătuit din 2 aplicații, una de front-end și una de back-end, prima are rolul de interacționa cu clientul, de a colecta, a procesa, și a trimite date către server. Iar cea de a doua are rolul de a receptiona, prelucra, stoca și de a trimite datele către client.

În cazul aplicației avem 2 servere locale, care rulează pe 2 adrese și porturi diferite, unul fiind responsabil de aplicația de front-end și iar celălalt de aplicația de back-end.

Solicitările către server de tip post și put executate în aplicație trimit date prin intermediul unor date de format JSON.

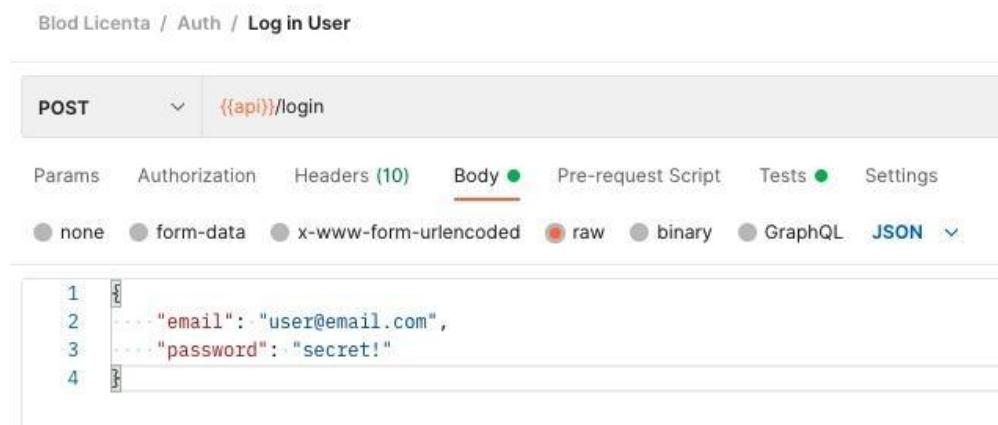


Figura 3.1.1 Exemplu request logare

JSON (JavaScript Object Notation) este un format de text folosit pentru schimb de date, bazat pe JavaScript, dar poate fi folosit în orice limbaj de programare. Este ușor de citit de către om și calculator. Folosit adesea în API-urile REST (mai des decât XML datorită lizibilității mai bune și greutății mai mici).

Valori JSON valide:

- Obiect JSON - un set neordonat de perechi , cuprinse între paranteze
`{ "cheie": valoare }`
- o matrice(array) -> un set ordonat de valori separate prin virgule. Este între paranteze pătrate [];
- număr (întreg sau real);
- true (boolean adevărat), false (boolean fals) și null;
- string

```

1 //Student JSON Object
2 {
3     "rollNumber" : 11,           ← Json for number values
4     "firstName" : "Saurabh",    ← without doublequote
5     "lastName" : "Gupta",       ← JSON for String values
6     "permanent" : false,      ← with double quote
7     "address" : {              ← JSON for boolean
8         "addressLine" : "Lake Union Hill Way",   allow values true/false
9         "city" : "Atlanta",          ← JSON for address object
10        "zipCode" : 50005          ← with in curly bracket
11    },
12    "phoneNumbers" : [ 2233445566, 3344556677 ], ← Array of Numeric
13    "cities" : [ "Dallas", "San Antonio", "Irving" ], ← Values
14    "properties" : {           ← Array of String values
15        "play" : "Badminton",   ← JSON to represent map
16        "interest" : "Math",    ← in key value pairs
17        "age" : "34 years"
18    }
19 }

```

Figura 3.1.2 Exemplu cod JSON

Pentru a functiona corect un cod JSON trebuie să fie “well formed”, adică corect din punct de vedere sintactic.

Reguli JSON “well formed”:

- Datele sunt scrise ca perechi cheie:valoare
- Date separate prin virgule
- Obiectul se află în acolade {}
- Matrice(array) - se află în paranteze pătrate []

3.2 Proiectare

3.2.1 Diagrama Use case

Inițial s-au analizat obiectivele și funcionalitatile propuse pentru dezvoltarea aplicației în prima fază. Pe baza cercetărilor efectuate s-a creat o diagramă a cazurilor de utilizare ce va prezenta funcionalitatile sistemului.

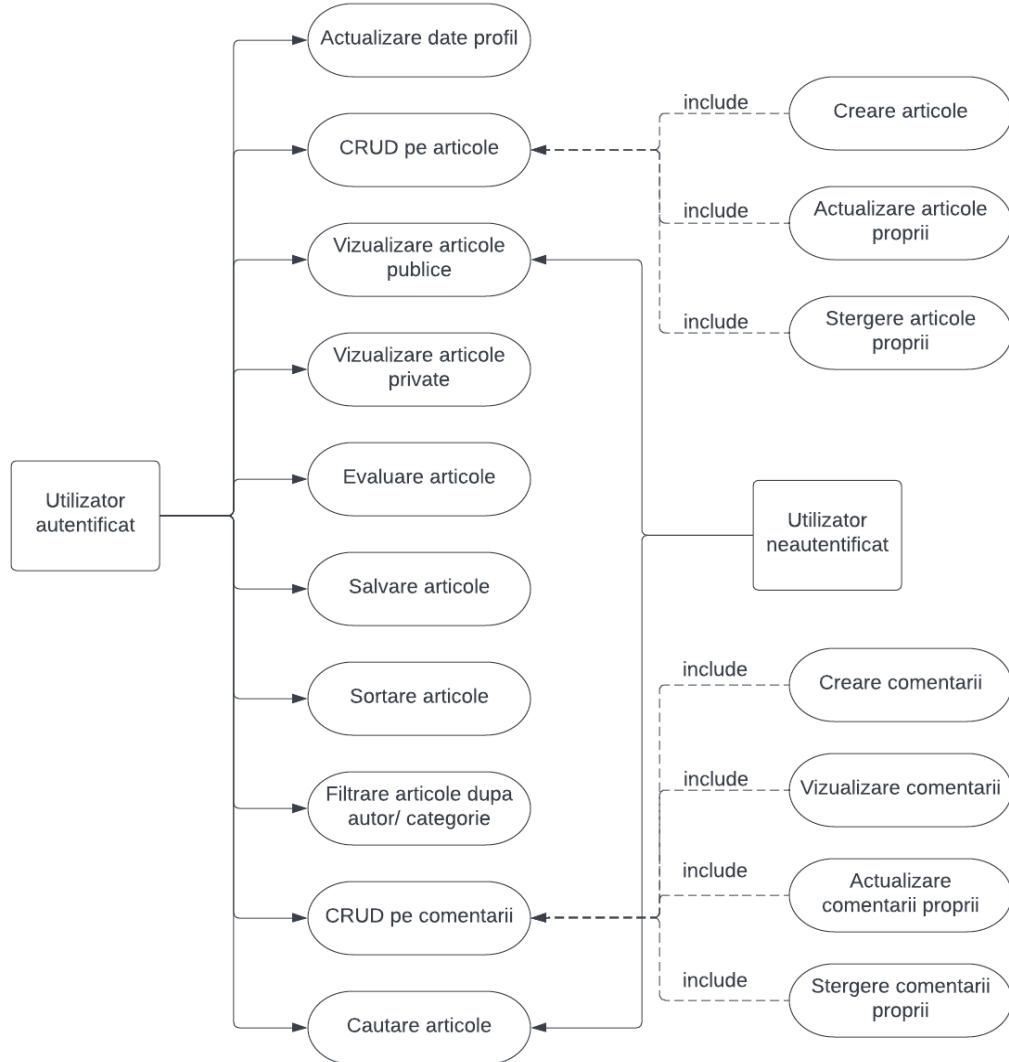


Figura 3.2.1.1 Diagrama Use Case a aplicației

O elipsă reprezintă funcționalitate sau un caz de utilizare, iar numele său este scris în interiorul acesteia. În cazul diagramei noastre avem 2 actori, primul reprezentând un "Utilizator autentificat" iar al doilea "Utilizator neautentificat". Un actor poate invoca un caz de utilizare. Diagrama este folosită pentru descrierea unui set de acțiuni pe care pe care sistemul le poate efectua în colaborare cu unul sau mai mulți utilizatori ale aplicației. Fiecare caz de utilizare oferă un rezultat actorilor cat și efectuează schimbări în baza de date a sistemului.

Diagrama are următoarele elemente:

- cazuri de utilizare (use cases) sau scenarii: descrie o secvență de acțiuni facute de către un actor
- actor: o entitate care interacționează cu sistemul nostru
- relații: între scenarii și actori, aceste asocieri există atunci când un actor interacționează cu sistemul prin intermediul unui scenariu
- Generalizare: identifică relații dintre actori și cazuri de utilizare

- relații de tip include: identifica un scenariu reutilizabil care este încorporat în execuția unui alt scenariu (reprezentate prin linii întretrerupte)

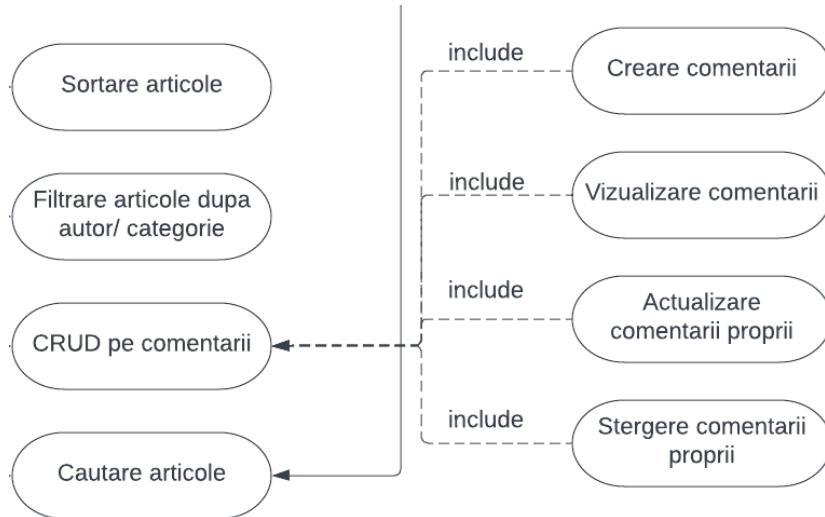


Figura 3.2.1.2 Diagrama use case exemplu relații tip include

3.3 Implementare

3.3.1 Prezentarea operațiilor

- **Înregistrare:** procesul de înregistrare constă în completarea unui formular cu câteva dintre datele pe care trebuie să le aibă fiecare utilizator (Nume, Prenume, adresa email, titlul job-ului pe care-l are și parola contului) și trimiterea acestuia într-o solicitare de tip POST către server, care returnează ca răspuns un mesaj ce reprezintă statusul solicitării, iar în caz de succes utilizatorul va primi pe adresa să poștală un email de confirmare
- **Confirmarea înregistrării** are loc prin accesarea unui link primit într-un mesaj pe adresa de email introdusă, linkul dat conține ca parametri e-mailul utilizatorului și un token de autentificare astfel încât odată ce utilizatorul îl accesează va avea loc automat logarea acestuia în aplicație.
- **Recuperarea parolei:** are loc prin completarea unei casute ce reprezintă adresa de e-mail a utilizatorului și trimitera acesteia într-o solicitare de tip POST către server, care returnează ca răspuns un mesaj ce reprezintă statusul solicitării, iar în caz de succes utilizatorul va primi pe adresa să poștală un email cu link-ul care-l poate accesa pentru resetarea parolei.
- **Resetarea parolei:** are loc prin accesarea unui link primit pe emailul utilizatorului care conține ca query parametri un token generat de backend și adresa de e-mail a utilizatorului. Odată accesat linkul utilizatorul are posibilitatea de a introduce nouă parola pe care dorește să o aibă. Datele completate se trimit

împreuna cu datele extrase din link printr-o solicitare de tip POST către server după care în caz ca sunt validate cu succes, parola va fi modificata.

- **Logarea:** are loc prin completarea unui formular format din adresa email și parola asociată contului și trimiterea acestuia într-o solicitare de tip POST către server, care returnează ca răspuns un token JWT de autentificare și datele profilului.
- **Vizualizarea articolele:** este disponibilă atât pentru utilizatorii neautentificati și pentru cei autentificati. Pentru aceasta functionalitate avem la dispoziție 2 endpointuri ambele de tip GET care returnează o lista de obiecte, fiecare obiect reprezentand un articol, un endpoint este public și celălalt securizat.
- **Vizualizarea utilizatorilor:** este o functionalitate disponibilă doar pentru utilizatorii autentificați, pentru aceasta avem la dispoziție un endpoint de tip GET care ne returnează o lista de obiecte, fiecare obiect reprezentand un utilizator împreună cu datele și statisticile sale ca autor (numărul de articole create, numărul de vizualizări, numărul de citiri și numărul de like-uri ale articolelor create de el)
- **CRUD articole:** orice utilizator autentificat poate crea/vizualiza/actualiza și șterge articolele proprii. Pentru aceste functionalitati sunt puse la dispoziție 4 endpointuri diferite: POST (pentru creare), PUT (pentru editare), GET (pentru vizualizare) și DELETE (pentru ștergere)
- **Filtrarea articolelor:** este o functionalitate dedicată utilizatorilor autentificați, prin intermediul ei, utilizatorii pot filtra articolele care le vad după autor și categorii.
- **Sortarea articolelor:** se poate face după 4 parametri diferenți: data creării, numărul de vizualizări, numărul de citiri și numărul de evaluări pozitive care le are articolul (reprezentând numărul de like-uri). Totodată pentru sortare se poate selecta și ordinea afișării, ascendent sau descendent, în mod implicit blogurile se afișează în ordine descrescătoare după data creării, adică de la cele mai recente la cele mai vechi.
- **Căutarea articolelor:** se realizează prin filtrarea articolelor ale căror titlu include valoarea introdusă de către utilizator în casuța dedicată căutării
- **Evaluarea articolelor:** orice articol existent poate fi evaluat de către un utilizator autentificat, evaluarea constă în atribuirea unui calificativ din cele 2 existente (like/dislike) articolului dorit, pentru aceasta functionalitate avem la dispoziție un endpoint de care acceptă solicitări de tip POST în care trimitem valoarea corespunzătoare calificativului, iar aceasta se înregistrează în baza de date.
- **Salvarea articolelor:** orice utilizator autentificat poate salva un articol existent, pentru ca ulterior să poată avea acces la el mai simplu fără a mai fi nevoie să-l caute prin toata lista de articole.
- **CRUD comentarii:** orice utilizator autentificat poate crea/vizualiza/actualiza și șterge comentariile proprii care la randul lor sunt atribuite unui articol. Pentru aceste functionalitati sunt puse la dispoziție 4 endpointuri diferite: POST (pentru creare), PUT (pentru editare), GET (pentru vizualizare) și DELETE (pentru ștergere).

* Observație: Orice endpoint la care are acces doar utilizatorii autentificați este securizat, aceasta înseamnă că trebuie să contină în “Headers”(antetul solicitării) tokenul de autorizare care a fost generat după logare, în timp ce endpointurile publice poate fi accesat fără a fi nevoie date suplimentare.

3.3.2 Arhitectura bazei de date

În aceasta secțiune sunt prezentate entitățile și relațiile bazei de date a aplicației web implementate. La crearea bazei de date s-au folosit următoarele tehnici de normalizare:

- Prima formă normală (1NF)
- A doua formă normală (2NF)
- A treia formă normală (3NF – Third Normal Form)
- Forma normală Boyce-Codd (BCNF)
- A patra formă normală (4NF)

Cu alte cuvinte, s-a rezolvat problema datelor repetitive (1NF) prin adăugarea relațiilor One-To-One, One-To-Many, Many-To-Many. Un exemplu este relația tabelelor users, blogs și blog_pinned. Această relație este de tip Many-To-Many. Tabela blog_pinned este tabela de legătură care are atributele user_id și blog_id care sunt chei externe ce pointează către tabela blogs și tabela users. În acest mod datele nu se repeta. În figura de mai jos sunt prezentate toate entitățile și relațiile lor.

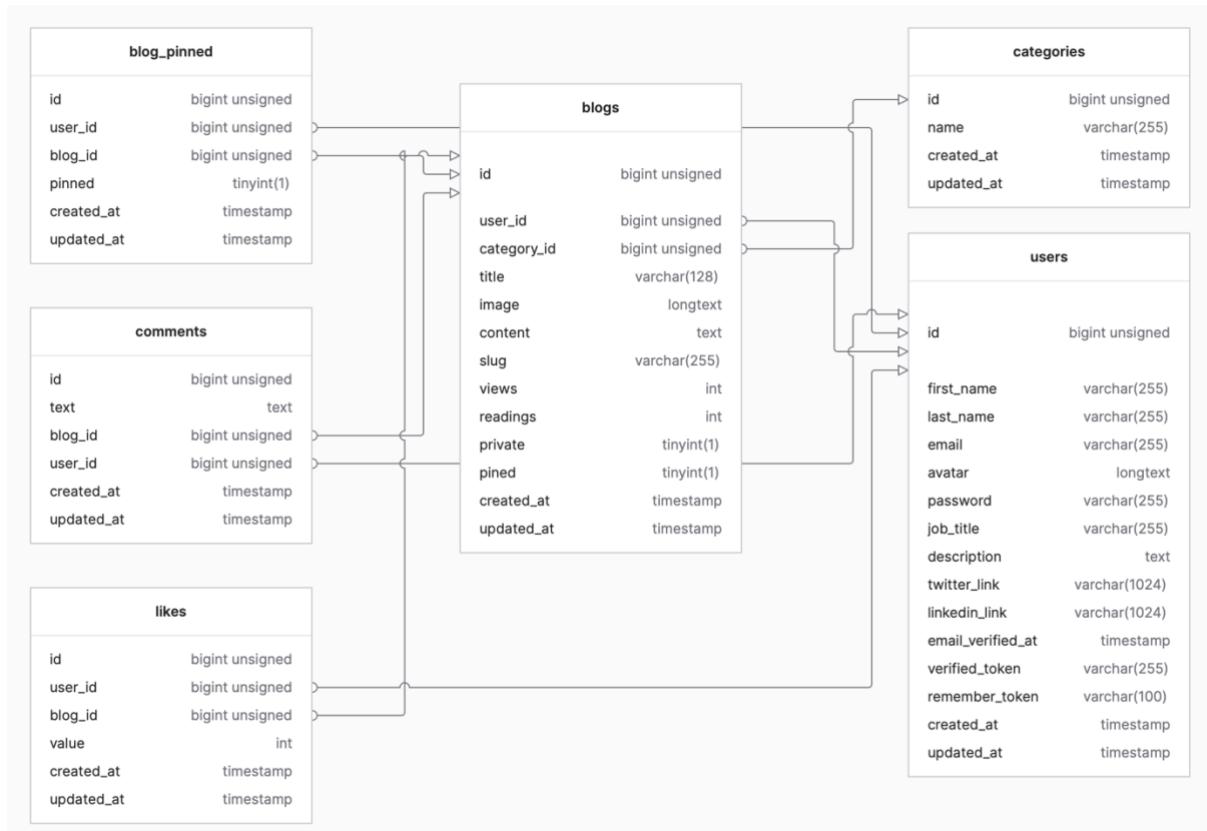


Figura 3.3.2.1 Diagrama bazei de date a aplicatiei

Backend-ul aplicatiei a fost implementat cu ajutorului pachetului Laravel Restify.

Laravel Restify ne ajută să cream REST API-uri mai rapid, mai ușor și cu mai multă consistență. Este un constructor Laravel JSON:API complet personalizabil. După instalarea și configurarea pachetului, putem începe rapid utilizarea lui. Repository-ul este nucleul acestui pachet.

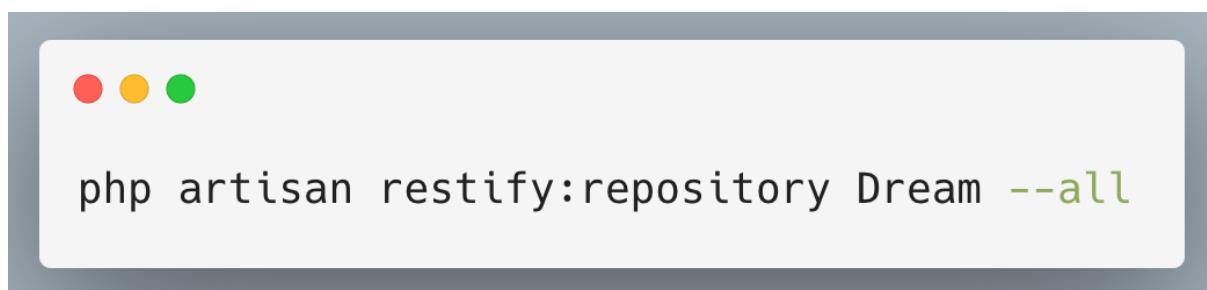


Figura 3.3.2.2 Comanda restify pentru crearea repozitoriului

Exemplul de comandă prezentată mai sus ar genera un repository gol la care putem adăuga câmpuri:

```
namespace App\Restify;

use App\Models\Dream;
use Binaryk\LaravelRestify\Http\Requests\RestifyRequest;

class DreamRepository extends Repository
{
    public static string $model = Dream::class;

    public function fields(RestifyRequest $request): array
    {
        return [
            id(),
            field('title')->required(),
            field('description'),
            field('image')->image(),
        ];
    }
}
```

Figura 3.3.2.3 Adaugarea campurilor în repozitoriu Laravel

Astfel cu ajutorul acestui pachet putem implementa foarte rapid toate acțiunile CRUD (Create/ Read/ Update/ Delete) pe diferite tabele. După rularea comenziilor și stabilirea campurilor avem acces la următoarele endpointuri, care pot fi testate în PostMan:

```
GET: http://laravel.test/api/restify/dreams
POST: http://laravel.test/api/restify/dreams
GET: http://laravel.test/api/restify/dreams/1
PUT: http://laravel.test/api/restify/dreams/1
DELETE: http://laravel.test/api/restify/dreams/1
```

Figura 3.3.2.4 Exemplu endpoint-uri generate prin restify

Acest pachet ne ajută de asemenea și în procesul de autentificare, paginare, filtrare avansată și multe altele!

Astfel pentru aplicația curentă, s-a creat o colecție Postman prin intermediul căreia putem testa și verifica endpointurile disponibile:

Blog	User	Category
Actions	GET Get users	GET Get categories
POST Add views on blogs	Auth	Comments
POST Add reading on blogs	POST Log in User	POST Create Comment
POST Add like to blog	POST Profile	PUT Update Comment
POST Pinn blog to user	POST Register new user	GET Get comments
POST Unpin blog from user	POST Forgot password	
GET Get public blogs	POST Reset password	
GET Get blogs	POST Verify User	
GET Get blog by slug	POST Change User password	DEL Delete comment
Blogs		
POST Delete blog		

Figura 3.3.2.5 Lista endpointuri aplicatie (din Postman)

3.3.3 Limbaje, framework-uri, pachete și servicii folosite la dezvoltarea aplicatiei

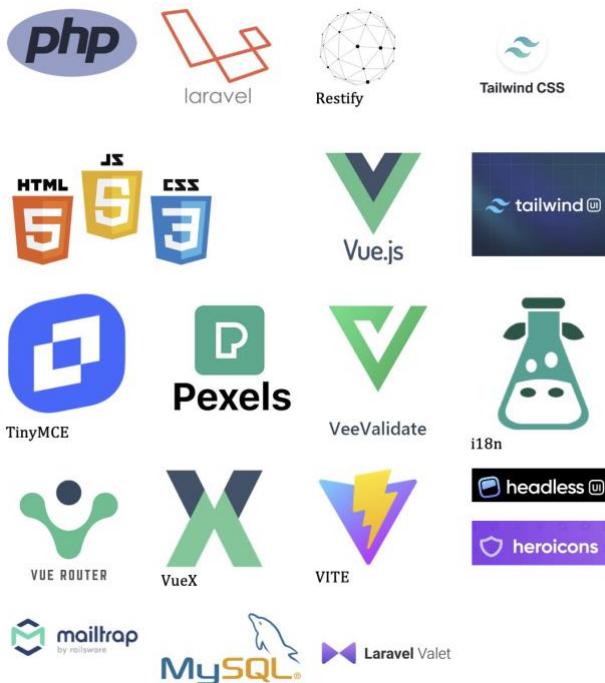


Figura 3.3.3.1 Logo-urile serviciilor folosite la aplicatie

- **Vite** este folosit ca un server de dezvoltare și un instrument de construcție(build) al aplicației.
- **Vue Router** este folosit pentru rutare. Vue router-ul ajută la conectarea între URL-uri/Istoricul browserului și componentele Vue, permitand ca pentru anumite căi(path-uri) să se randeze componentele necesare și paginile asociate cu acestea.
- **Vite Plugin Routes** este folosit pentru a genera automat rute după locația fișierelor în proiectul aplicației de front-end.

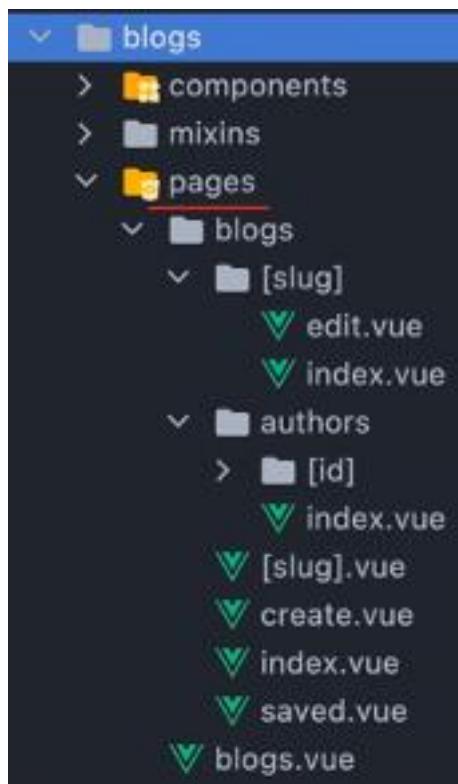


Figura 3.3.3.2 Modulul “blogs” din aplicație front-end

Pentru modulul de “blogs” există dosarul “pages” în care sunt toate paginile disponibile modulului curent. După cum se poate observa există dosarul principal “blogs” având ca echivalent ruta “/blogs”, în fișierul “blogs.vue” e implementat template-ul modulului, modul de afișare și acțiunile ce se executa odată cu încărcarea sa. De asemenea în dosarul “blogs” mai avem 2 subdosare: “slug”(combinatie de cuvinte generate din titlul unui blog, folosita la identificarea acestuia) și “authors”, faptul ca “[slug]” se află între paranteze patrate semnifica că acesta este un parametru dinamic, are ca echivalent ruta “/blogs/[slug]”. În dosarul “slug” mai avem 2 fișiere care la randul lor reprezinta alte 2 subrute: conținutul din “index.vue” se va randa în mod implicit pentru ruta “/blogs/[slug]”, iar conținutul din “edit.vue” se va randa în cazul rutei: “/blogs/[slug]/edit”. Astfel se poate înțelege modul de funcționare al plugin-ului curent. Fiecare nivel reprezinta o subrute, fișierele ce au același nume ca al dosarului parinte reprezinta template-urile rutei, fișierele sau dosarele ale căror nume sunt în interiorul parantezelor patrate reprezinta rute determinate cu ajutorul unui parametru dinamic.

- **Vuex** este un pachet de gestionare a stării și a pentru aplicațiile Vue.js. Modulele sunt încărcate automat în “src/store.ts”. Pentru ca un modul vuex să fie încărcat, acesta trebuie să fie plasat în folderul “stores” și numele acestuia să se termine cu “Module.ts”. Ca exemplu: “authModule.ts”.

```

export type State = {
  blogs: [];
  publicBlogs: [];
  categories: [];
  authors: [];
  pinnedBlogs: [];
  categoryColorsClass: {};
}

export type State = {
  isLoggedIn: boolean;
  user: object;
}
  
```

Figura 3.3.3.3 Declararea variabilelor pentru modulelor vue-store

În aplicația realizată avem 2 module (authModule și blogsModule). În modulul de autentificare(authModule) se pastrează toate datele utilizatorului. În blogsModule se stochează datele despre articole, categoriile disponibile, autori/utilizatori, articole salvate... Pe lângă date, în modulele de vuex se stochează și acțiunile necesare acestora (obținere, actualizare, ștergere, logare, delogare...)

- **Vue I18n** este un plugin folosit pentru traduceri și afișarea corectă a conținutului dinamic în funcție de diferite variabile.

```

{
  "Welcome Message": "Welcome Message Translation",
  "locales": "ro",
  "minutes_read": "minute to read | {n} to minutes read"
}
  
```

Figura 3.3.3.4 Declarație a traducerilor

```

<span class="reading-time">
  {{ time }} $t('minutes_read', time)
</span>
  
```

Figura 3.3.3.5 Utilizarea traducerilor

Pentru fiecare limbă în care se dorește traducerea conținutului aplicației există cate un fisier json, în care sunt stocate toate textele din aplicația, traduse în limba respectivă sub forma unui obiect. Astfel, prin afișarea conținutului în acest mod: \$t("cheia conținutului") se verifică care limbă este activă, și se cauta conținutul corespunzător "cheiei" în fisierul necesar după care se afișează textul corect. Dacă nu există o traducere cu o astfel de cheie, se afișează însuși textul cheiei.

- **Vee-Validate**: este o bibliotecă de validare pentru Vue.js. Are o mulțime de reguli de validare prestabilite și de asemenea oferă suport pentru crearea regulilor personalizate. Este similar și familiarizat cu API-ul de validare HTML5. Cu ajutorul sau se pot valida intrările HTML5, precum și componentele Vue personalizate.

```
const emailRegex = /(^$)|^(([^<>()\\[\]\\.,;:\\s@"]+\\(.\\.)*|^(([^<>()\\[\]\\.,;:\\s@"]+\\(.\\.)*\\.)+\\(.\\.)*))$/;
```

```
defineRule( id: 'email', validator: (value: string) => {
  if (value && emailRegex.test(value)) {
    return true
  }
  return 'This field must be a valid email';
});
```

Figura 3.3.3.6 Declararea regulei de validare cu vee-validate

În exemplul de mai sus s-a definit o regula pentru a valida un email, validarea se face pe baza unei expresii regulate. Definim regulă prin intermediul funcției "defineRule" ce e importată din pachet. Funcția are 2 parametri, primul reprezentând numele regulei, iar al doilea reprezintă funcția prin care se realizează validarea("validator"). Funcția "validator" primește la randul ei un singur parametru, ce reprezintă valoarea ce urmează a fi validată. Dacă există valoarea și verifica expresia regulată introdusă se returnează valoarea booleană true, ceea ce semnifică că valoarea verificată respectă regula de validare, în caz contrar funcția returnează un text ce reprezintă mesajul de validare.

```
<BaseInput  
  v-model="form.email"  
  :label="$t( key: 'Email')"  
  rules="required|email"  
/>
```

Figura 3.3.3.7 Aplicarea regulilor de validare

În exemplul de mai sus se executa validarea unei variabilei asociate unei componente de baza folosită pentru inputuri, "form.input" fiind variabila. În atributul rules sunt scrise regulile necesare validării campului, în cazul de fata sunt prezente 2, "required" pentru a indica faptul că campul este obligatoriu și "email" pentru a indica faptul că câmpul trebuie să fie un e-mail, ele sunt despărțite prin bara verticală, acesta este un criteriu foarte important ce trebuie respectat pentru a asigura o funcționare corectă. În cazul în care variabila nu trece validarea sub elementul de "input" va fi afișat mesajul de eroare.

Email address *

The email field is required

Figura 3.3.3.8 Mesaj eroare camp obligatoriu

Email address *

This field must be a valid email

Figura 3.3.3.9 Mesaj eroare email invalid

- **HeadlessUI:** Este un set de componente UI complet nestilizate, complet accesibile, concepute pentru a se integra frumos cu Tailwind CSS. HeadlessUI este un instrument din categoria UI Components a unei stive de tehnologie, este un instrument open source cu 15,9K stele GitHub și 608 furci(copii) GitHub.

- **HeroIcons:** este un set de pictograme SVG gratuite creat de către fondatorii Tailwind CSS, ele vin în două dimensiuni diferite și sunt pre-optimizate pentru a fi stilizate cu clase CSS direct în HTML. Acest pachet oferă peste 200 de pictograme unice, fiecare în 2 variante, “outline”(conturate) și “solid” (pline de culoare)

Exemplu de utilizare:

```
import { CheckIcon, ExclamationIcon } from '@heroicons/vue/outline'
```

Figura 3.3.3.10 Importarea pictogramelor heroicons

În partea de script a componentei se importă pictogramele din pachetul heroicons, iar în partea de sablon HTML aceasta se folosește ca în exemplul de mai jos, de asemenea i se pot atribui clase css pt stilizare. Pictograma din exemplul de mai jos are setate prin intermediul claselor dimensiunile și culoarea.

```
<CheckIcon  
  aria-hidden="true"  
  class="flex-shrink-0 w-6 h-6 text-green-500"  
/>
```

Figura 3.3.3.11 Utilizarea pictogramelor heroicons

- **MailTrap:** este un serviciu pentru testarea în siguranță a e-mailurilor trimise din mediile de dezvoltarea aplicațiilor. Mailtrap trimit e-mailurile într-o căsuță de e-mail virtuală, astfel încât se oferă posibilitatea de a testa și a optimiza campaniile de e-mail înainte de a le trimite utilizatorilor reali.
- **Mailator:** este un programator de e-mail-uri pentru Laravel. Oferă un pachet ușor de configurat,[33] care poate la randul sau să configureze a programe de e-mail și a şabloanelor pe baza diferitor evenimente ale aplicației.

- **TinyMCE**: TinyMCE este un editor de text online cu sursă deschisă. Se integrează ușor cu javascript precum alte framework-uri ale acestuia. Deoarece a oferit o valoare excelentă la un preț scăzut, a devenit alegerea de preferat pentru editare. Are o mulțime de clienți high-end precum Evernote, Medium, Shopify și Atlassian.[38] În general, este un editor HTML WYSIWYG avansat care simplifică crearea de conținut.

TinyMCE permite utilizatorilor finali să adauge și să editeze conținut pe un site web. Un alt beneficiu pe care îl oferă este personalizarea de a adăuga butoane în bara de instrumente.[38] De asemenea este flexibil cu numeroase API-uri. TinyMCE are o colecție de peste 50 de pluginuri, precum și peste 100 de opțiuni de personalizare diferite.

```
editorOptions() {
    return this.options || {
        content_css: '/assets/css/juice-editor.css',
        menubar: ' edit insert format tools',
        toolbar_mode: 'wrap',
        max_height: 700,
        contextmenu: 'image',
        toolbar: [
            'styleselect | bold italic underline | ' +
            'bullist numlist | alignleft aligncenter alignright | ' +
            'link codesample code | insertImageBtn',
        ],
        plugins: 'autolink codesample link lists autosize ' +
            'wordcount image searchreplace paste code',
        imagedtools_cors_hosts: ['pexels.com', 'images.pexels.com'],
        mobile: {
            menubar: ' edit insert format tools',
            toolbar_sticky: true,
            toolbar_sticky_offset: 64,
            toolbar_mode: 'floating',
        },
        setup: (editor) => {
            this.editor = editor;

            editor.ui.registry.addButton( name: 'insertImageBtn', spec: {
                icon: 'image',
                onAction: () => this.toggleInsertImagePopup()
            });
        },
    }
}
```

Figura 3.3.3.12 Modul de configurare al editorului

- **Pexels**: este un site web gratuit pentru fotografii și videoclipuri și o aplicație care ajută designerii, bloggerii și toți cei care caută elemente vizuale să găsească fotografii și videoclipuri grozave care pot fi descărcate și utilizate gratuit. Toate fotografiile sunt frumos etichetate, pot fi căutate și, de asemenea, ușor de găsit.

Pixel oferă de asemenea un serviciu de API public prin intermediul căruia se pot integra serviciile sale în orice aplicație web. [32]

O astfel de integrare a fost implementată și în aplicația curentă, astfel în timpul editării sau creării unui articol nou, utilizatorul are posibilitatea de a selecta din meniul editorului opțiunea de inserare a imaginilor, după care poate ajuta imagini după cuvinte cheie iar ulterior imaginea selectată se poate introduce un articol. De asemenea a fost implementată funcționalitatea ca după orice imagine introdusă să fie plasat un text cu informații despre imagine și sursa acesteia pentru a fi respectate drepturile de autor.

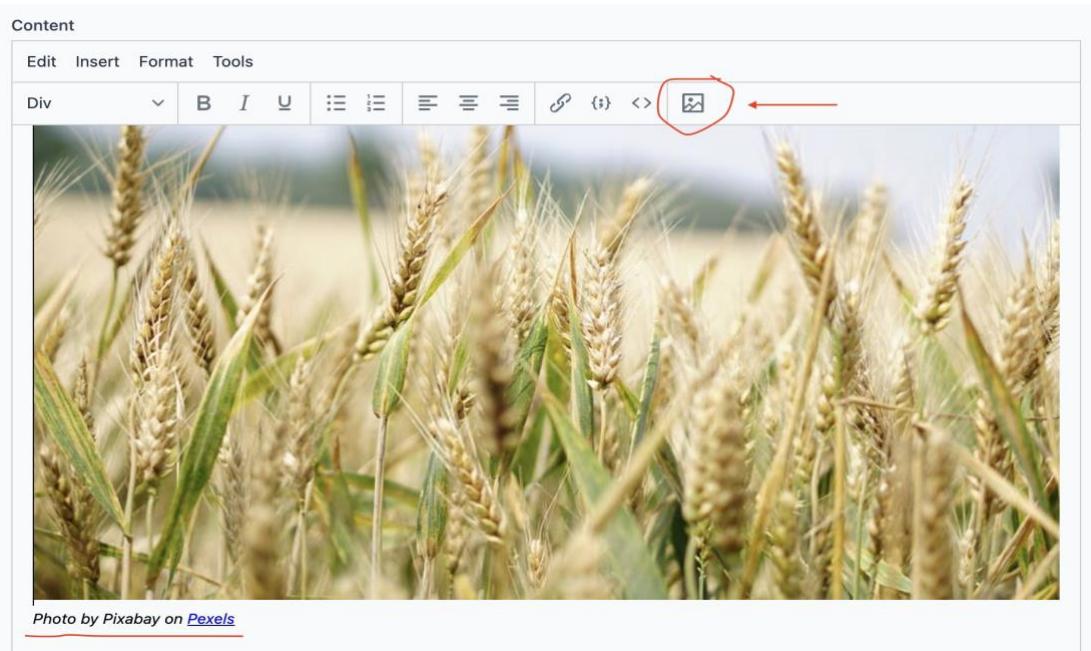


Figura 3.3.3.13 Integrarea Pexels în editorul HTML

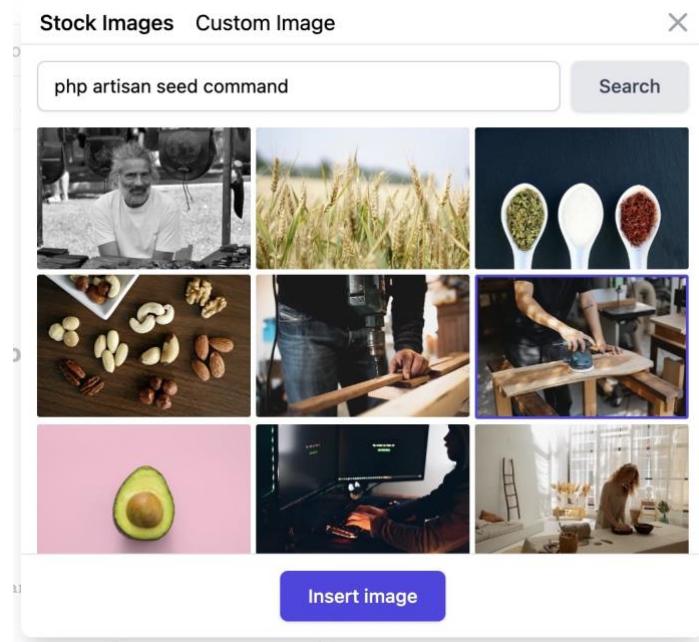


Figura 3.3.3.14 Fereastra de căutare și selectare a imaginilor

- **Axios** este folosit pentru a face request-uri. De asemenea au fost creați cativa interceptori de răspunsuri în “src/modules/common/apiConfig.ts”. Interceptorii de request-urilor sunt implementați pentru a simplifica modul de executare a cererilor către server. astfel dacă în stocarea locală a browserului există tokenul salvat de la autentificare acesta este introdus automat în antetul request-ului ce urmează să fie efectuat. Există de asemenea și interceptori pentru răspunsurile request-urilor, în cazul în care request-ul returnează o eroare, se afișează automat o notificare cu un mesaj de eroare ce descrie problema în funcție de codul de eroare returnat. În funcție de codul erorii, dacă requestul returnează un mesaj de eroare, acesta se afișează automat ca notificare pentru utilizator.

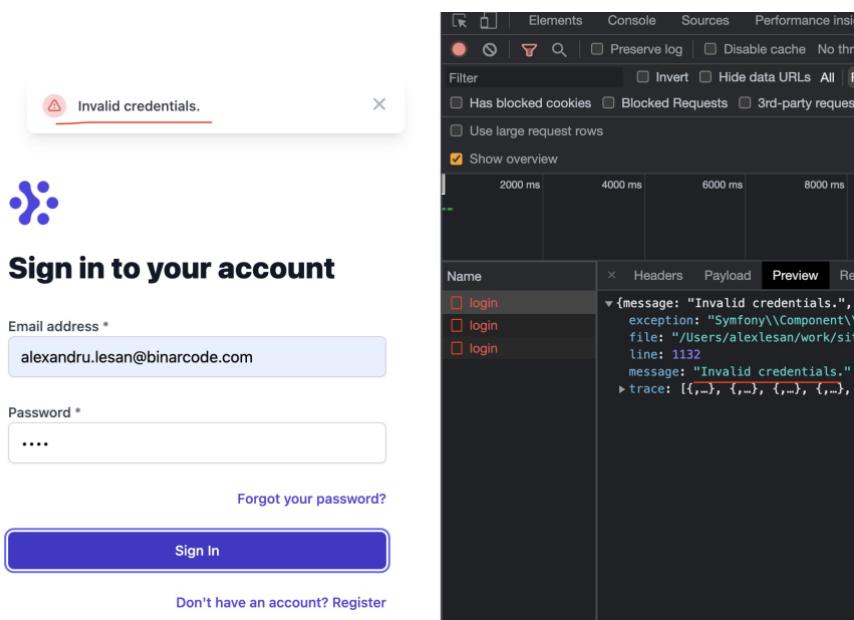


Figura 3.3.3.15 Afișarea notificării de eroare la răspunsul request-ului

- **Tailwind CSS** este folosit pentru stilizare. Tailwind CSS funcționează prin scanarea claselor din toate fișierele HTML, componentele JavaScript și oricare alte şablonane, generând stilariile corespunzătoare și apoi scriindu-le într-un fișier CSS static. Este rapid, flexibil și fiabil - cu timp de rulare zero. Configurația a fost extinsă în tailwind.config.ts.

```
theme: {
  extend: {
    colors: {},
    inset: {
      18: '4.5rem',
    },
    padding: {
      '1.75': '0.4375rem',
    }
  }
},
```

Figura 3.3.3.16 Exemplu de extindere a configurării TailwindCSS

În exemplul de mai sus a fost adaugata o marime aditionala pentru padding respectand convenția după care funcționează tailwind css, mărimea care va fi aplicată în clasă trebuie să fie de 4 ori mai mare mărimiile pe care o reprezintă în unitatea de masura "rem".

```
class="hidden md:flex md:gap-4 md:items-center"
```

Figura 3.3.3.17 Exemplu de utilizare a claselor TailwindCSS

Clasele de mai sus au fost atribuit unui element HTML, fiecare dintre ele reprezentând o stilizare css, astfel acestea pot fi interpretate în modul următor:

Clasa/ punct de rupere TailwindCss	Stilizarea CSS	Comentarii
hidden	display: none;	pentru controlul tipului de afișare a unui element.
flex	display: flex;	pentru controlul tipului de afișare a unui element.
gap-4	gap: 1rem;	pentru controlul distanței dintre elementele grid și flexbox.
items-center	align-items: center;	pentru a controla modul în care elementele flexibile și grile sunt poziționate de-a lungul axei transversale a containerului.
md:	@media (min-width: 768px) { ... }	pentru responsive design, stilizarile folosite împreună cu acest punct de rupere vor fi aplicate pentru ecranele cu lățimea minima de 768px

Tabelul 3.3.3.18 Tabelul cu explicații clasele TailwindCSS

- **Securizarea rutelor:** După cum sugerează și numele, securizarea rutelor de navigație furnizate de Vue-Router este utilizată în principal pentru a securiza navigarea între paginile aplicației prin redirecționarea sau anularea acestora. Există mai multe moduri de implementare a securizării rutelor: global, pe rută sau în componentă. Pentru aplicația curentă a fost creată o funcție globală de securizare a rutelor, numita "authMiddleware" ce are ca parametru pachetul "router" pus la dispoziție de către framework-ul Vue. În aplicație avem 2 tipuri de rute, publice și private, cele private necesită ca utilizatorul să fie autentificat, iar

verificarea se face pe baza unui token care este înregistrat în stocarea locală a browser-ului. De fiecare dată cand are loc navigarea pe o alta ruta se verifica dacă ruta următoare(“to”) necesită ca utilizatorul să fie autentificat. Verificarea se face pe baza a două cazuri generale. Dacă ruta necesită autentificare iar tokenul lipsește din stocarea locală utilizatorul va fi redirectionat pe pagina de “login”. Dacă ruta următoare aparține modulului de autentificare iar tokenul este prezent în stocarea locală a browserului, utilizatorul va fi redirectionat pe pagina implicită a aplicației(“/blogs” în cazul de fata, pagina în care se pot vizualiza toate articolele). În celelalte cazuri utilizatorul va fi redirectionat pe ruta pe care a intenționat să navigheze.

```

import {hasToken} from "@/modules/auth/utils/tokenUtils";

const routesPath = {
  login: `/auth`,
  default: `/blogs`
};

export default function authMiddleware(router) {
  router.beforeEach(async (to, from, next) => {
    const requiresAuth = to.matched.some(record => record.meta.requiresAuth);
    const isAuthModule = to.path.includes('/auth') && !to.path.includes('/authors');

    if (requiresAuth && !hasToken()) {
      return next(routesPath.login);
    }

    if (isAuthModule && hasToken()) {
      return next(routesPath.default);
    }

    return next();
  });
}

```

Figura 3.3.3.19 Codul sursa al funcției middleware

- **Componente de bază:** Toate componente ale căror nume încep cu “Base” și de asemenea pictogramele ale căror nume încep cu “Icon” sunt importate automat, prin urmare, nu este nevoie să fie importate în fiecare componentă în care urmează să fie folosite. Pentru aceasta s-a folosit un script ce operează cu ajutorul compilatorului “Vite”. Acesta salvează într-un vector(array) toate căile fișierelor ce respectă un anumit format, în cazul de fata, se extrag toate căile fișierelor ale căror nume respectă convenția menționată anterior(să înceapă cu “Base” sau “Icon” iar extensia să fie una din următoarele “.vue, .js, .svg”), totodată pentru a nu ingreuna rularea scriptului și pentru a respecta o ordine și o consistență în proiect fișierele au fost amplasate în dosare specifice, astfel căutarea are loc doar

în anumite dosare, fapt ce imbunatatește performanțele rulării. Odată găsite căile componentelor, cu ajutorul funcțiilor javaScript se extrage numele fișierelor, se formatează într-un mod corespunzător pentru a se putea executa înregistrarea componentei după care se înregistrează global în proiect folosit metoda .component(nume, compoenta) pusa la dispozitie de Vue.

```
const components = import.meta.globEager('../components/**/(Base|Icon)*.(vue|js)')
const icons = import.meta.globEager('../assets/icons/(Icon)*.(vue|svg)')

export default {
  install(Vue: App) {

    function importComponents(components: any) {
      for (let filePath in components) {

        const componentPath = filePath
          .split('/')
          .pop()
          .replace(/\.\.\w+$/, '')

        const componentName = upperFirst(camelCase(componentPath))
        const componentConfig = components[filePath]
        // Înregistrarea componentei global
        Vue.component(
          componentName,
          componentConfig.default || componentConfig
        )
      }
    }

    importComponents(components);
    importComponents(icons);

    Vue.component('BaseForm', VeeForm)
  }
}
```

Figura 3.3.3.20 Codul sursa al funcției de importare globală a componentelor de bază

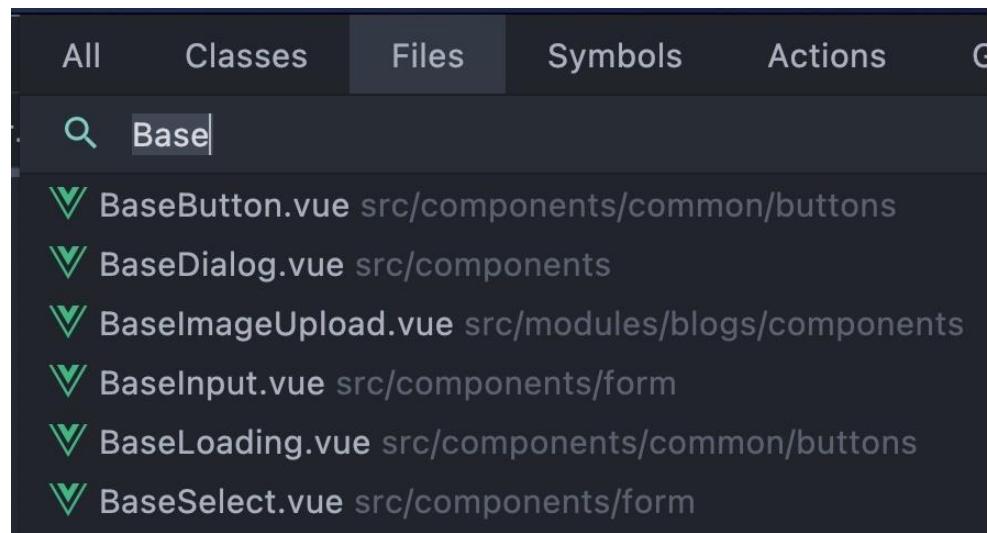


Figura 3.3.3.21 Lista componentelor de baza

În aplicație avem 6 componente de baza, care sunt utilizate foarte des, aceasta fiind unul dintre motivele principale din cauza cărora s-a ales să fie globale. Fiecare dintre aceste componente este complexă și poate fi setată și manipulată în diferite moduri doar aplicând diferite proprietăți.

```

<BaseButton
  :label="$t('Primary')"
/>
<BaseButton
  :loading="true"
  :label="$t('Primary loading')"
/>
<BaseButton
  :disabled="true"
  :label="$t('Primary disabled')"
/>
<BaseButton
  variant="secondary"
  :label="$t('Secondary')"
/>
<BaseButton
  variant="danger"
  :label="$t('Danger')"
/>

```



```

<BaseButton
  variant="secondary"
  :label="$t('Secondary xs')"
  size="xs"
/>
<BaseButton
  variant="secondary"
  :label="$t('Secondary sm')"
  size="sm"
/>
<BaseButton
  variant="danger"
  :label="$t('Danger lg')"
  size="lg"
/>
<BaseButton
  variant="danger"
  :label="$t('Danger xl')"
  size="xl"
/>

```

Figura 3.3.3.22 Codul folosirea pentru variantele de butoane



Figura 3.3.3.23 Afisarea butoanelor din aplicatii

● **Functii și plugin-uri globale:**

Aplicația are de asemenea cateva functii și plugin-uri înregistrate globale pentru a putea fi folosite cu ușurință din orice componentă

- **this.\$copyToClipboard(mesaj)**: poate fi folosit pentru a copia conținut în clipboard.

```
import { success } from "@/components/common/NotificationPlugin";

function copyToClipboard(value: string, message = 'Text copied to clipboard.') {
  const el = document.createElement('textarea'); // se creaza un element textarea

  el.value = value; // se introduce continutul in textarea

  document.body.appendChild(el); // se adauga elementul textarea in body
  el.select(); // se selecteaza tot continutul din textarea
  document.execCommand('copy'); // se executa comanda de copiere
  document.body.removeChild(el); // se sterge textarea din body
  success(message) // se afiseaza un mesaj de succes
}
```

Figura 3.3.3.24 Codul sursa al funcției de copiere a conținutului

Functia “copyToClipboard” primește 2 parametri, primul fiind “value” ce reprezinta continutul care urmeaza a fi copiat în clipboard, iar al doilea, “message” reprezinta mesajul afisat ca notificare popup în caz de success. Functia operează cu metode implicate puse la dispoziție de către limbajul javaScript.

- **this.\$format Date(data, format)**: poate fi folosit pentru a formata dătile.

Functia “formatDate” primește 2 parametri, primul fiind “date” ce reprezinta data care urmeaza a fi formatata, iar al doilea, “dateFormat” reprezinta formatul in care trebuie formatata data . Funcția operează cu metode puse la dispoziție de către pachetul “date-fns”. “Date-fns” este un pachet ce oferă cel mai cuprinzător, simplu și consistent

set de instrumente pentru manipularea datelor JavaScript în browser dar și în limbaje/framework-uri asociate acestuia.

```

import en from 'date-fns/locale/en-US'
import { format, parseISO } from 'date-fns' // importam modulele necesare

export const DEFAULT_DATE_FORMAT = 'dd/MM/yy' // definirea formatului de data implicită

export function formatDate(date: Date | string, dateFormat = DEFAULT_DATE_FORMAT) {
  if (!date) {
    return '--' // dacă nu există data, returnăm un --
  }
  let dateToFormat: any = date // initializăm variabila dateToFormat, copie din date
  if (typeof date === 'string') { // dacă date este un string, parsăm date
    dateToFormat = parseISO(date)
  }
  if (!dateToFormat.getTime() || isNaN(dateToFormat.getTime())) {
    // dacă dateToFormat nu are getTime sau nu este număr, returnăm un --
    return '--'
  }

  return format(dateToFormat, dateFormat, { locale: en }) // returnăm data formatată
}

```

Figura 3.3.3.25 Codul sursa al funcției de formatare a dății

- this.\$success("mesaj") sau this.\$error("mesaj"): pot fi folosit pentru a declanșa notificări în aplicație, în componente, cât și în cod.

Pentru sistemul de notificari, s-a implementat un plugin personalizat, care de asemenea este importat global și poate fi folosit cu usurință din orice componentă a aplicației. Prestabilit există posibilitatea de a afișa notificari de 4 tipuri: succes (this.\$success("mesaj")), eroare (this.\$error("mesaj")), informativ (this.\$info("mesaj")), atentionare (this.\$warning("mesaj")). În mod implicit toate notificările prestabilite vin cu aceleași setări, de funcționare, cât și de afișare, astfel o notificare apare în colțul drept sus, îndată după o notificare anterioară și dispare automat după 5 secunde, sau în momentul în care utilizatorul da click pe ea sau apăsa pe butonul "X" din colțul acesteia. Fiecare mod prestabilit are asociată o iconă informatică.

Exemplu de utilizare a notificărilor din cod:

```
this.$success(this.$t('Notificare cu mesaj de success'));
this.$error(this.$t('Notificare cu mesaj de eroare'));
this.$info(this.$t('Notificare cu mesaj de informare'));
this.$warning(this.$t('Notificare cu mesaj de atentionare'));
```

Figura 3.3.3.26 Codul pentru afişarea notificărilor

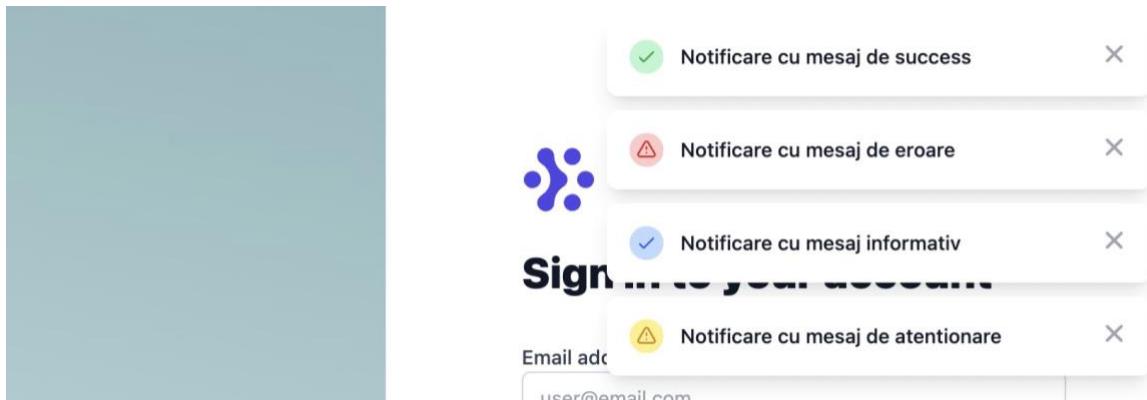


Figura 3.3.3.27 Afisarea notificarilor

Totodată există posibilitatea de a afișa notificări ce pot fi personalizate cu ajutorul mai multor parametri, folosind comanda globală `this.$notify({ proprietăți })`, de exemplu:

```
this.$notify({
  icon: 'mdi-check',
  message: this.$t('mesaj notificare personalizata'),
  timeout: 0,
  showClose: false,
  ...alteProprietati
});
```

Figura 3.3.3.28 Codul pentru afişarea notificărilor personalizate



Figura 3.3.3.29 Afisarea notificarilor personalizate

```

name: 'Notification',
props: {
  message: [String, Object],
  title: String,
  icon: String,
  verticalAlign: {
    type: String,
    default: 'top',
    validator: (value: string) => {
      let acceptedValues = ['top', 'bottom'];
      return acceptedValues.indexOf(value) !== -1;
    }
  },
  horizontalAlign: {
    type: String,
    default: 'right',
    validator: (value: string) => {
      let acceptedValues = ['left', 'center', 'right'];
      return acceptedValues.indexOf(value) !== -1;
    }
  },
  type: {
    type: String as PropType<NotificationType>,
    default: 'info',
  },
},
  timeout: {
    type: Number,
    default: 5000,
    validator: (value: number) => {
      return value >= 0;
    }
  },
  timestamp: {
    type: [Date, Number],
    default: () => new Date()
  },
  component: {
    type: [Object, Function]
  },
  showClose: {
    type: Boolean,
    default: true
  },
  closeOnClick: {
    type: Boolean,
    default: true
  },
}
,
```

Figura 3.3.3.30 Proprietățile disponibile pentru personalizarea notificărilor

Proprietatea	Semnificația
message	mesajul care va apărea împreuna cu notificarea
title	titlul notificării, aflat deasupra mesajului
icon	pictograma notificarii
verticalAlign	locul unde va fi afisata notificarea pe axa verticala (sus sau jos)
horizontalAlign	locul unde va fi afisata notificarea pe axa verticala (stanga, centru sau dreapta)
type	tipul notificării (succes, eroare, informativ, atenționare)
timeout	timpul reprezentat în milisecunde după care va dispărea notificarea, în caz ca are valoarea 0, notificarea va fi afisata tot timpul pana nu va fi închisă manual
closeOnClick	varianta booleană în dependență de care notificarea se executa funcția de ascundere sau nu a notificarii la apasare pe ea
showClose	valoare booleană (true sau false) în dependență de care se va afișa iconita "X" de ascundere a notificarii

Tabelul 3.3.3.31 Proprietățile disponibile pentru personalizarea notificărilor-explicate

3.4 Testare

3.4.1 Testarea securizării funcționalităților

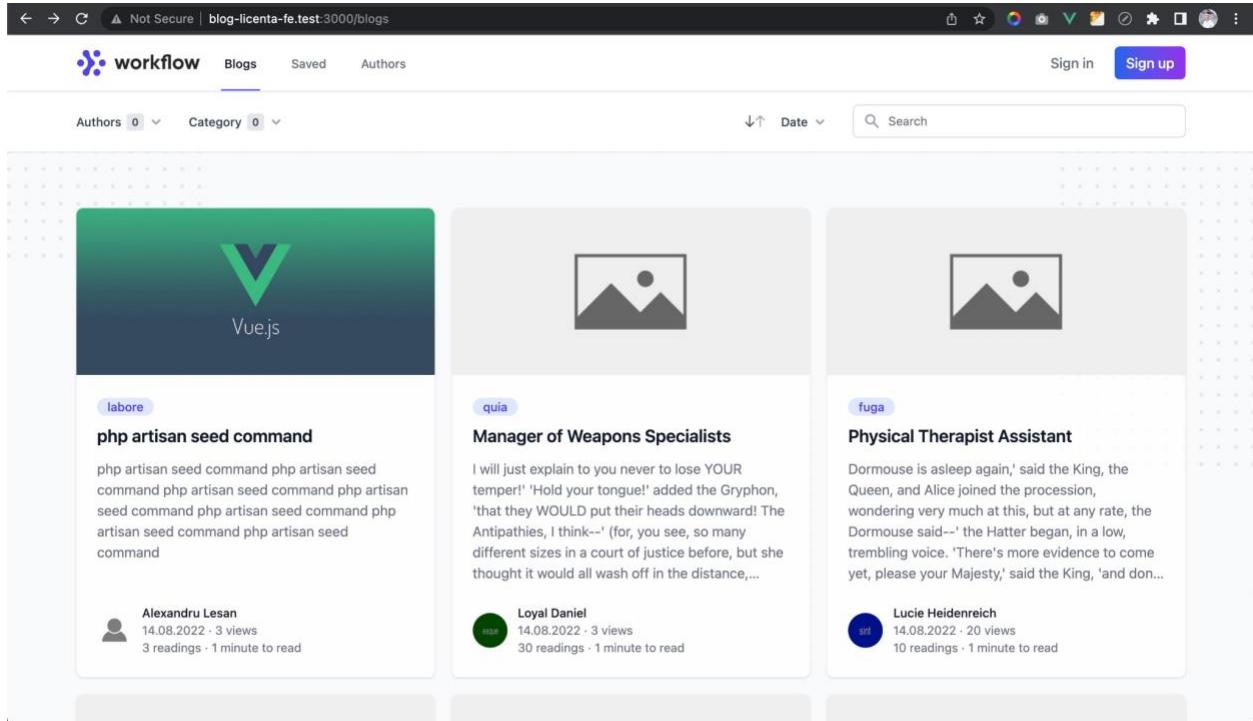


Figura 3.4.1.1 Pagina principala a aplicatiei

Pe pagina principala există o bara de instrumente ce oferă opțiuni de filtrare a blogurilor după categorie sau autor, totodată blogurile pot fi sortate ascendent sau DESCENDENT după diferite criterii ca data publicării, numărul de vizualizări, numărul de citiri și numărul de aprecieri. Aceste opțiuni sunt valabile și pe pagina cu articole salvate. De asemenea există și posibilitatea de a căuta articole, introducând text în căsuța aflată pe aceeași bara de instrumente lângă opțiunile de sortare.

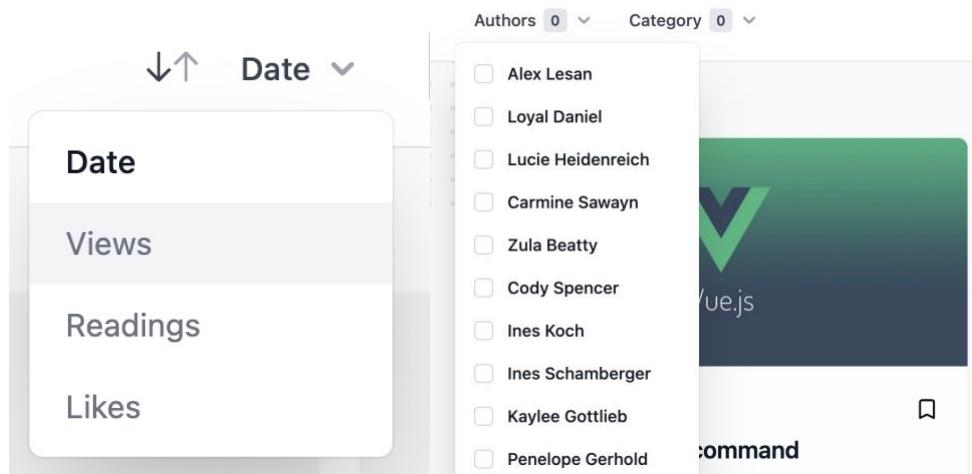


Figura 3.4.1.2 Metodele de filtrare și sortare a blogurilor

Metodele de filtrare și sortare, împreună cu cele de navigare pe paginile de bloguri salvate("Saved") sau pagina de prezentare a autorilor("Authors") sunt restrictionate

pentru utilizatorii neautentificați, astfel incat daca seincearca accesarea lor, utilizatorului îi va aparea o fereastra care-l instiinteaaza ca aceasta actiune este restrictionata și pentru a continua este necesar sa fie autentificat, iar pe langa asta i se prezinta o lista cu avantajele și alte optiuni ce le pot îndeplini utilizatorii autentificați.

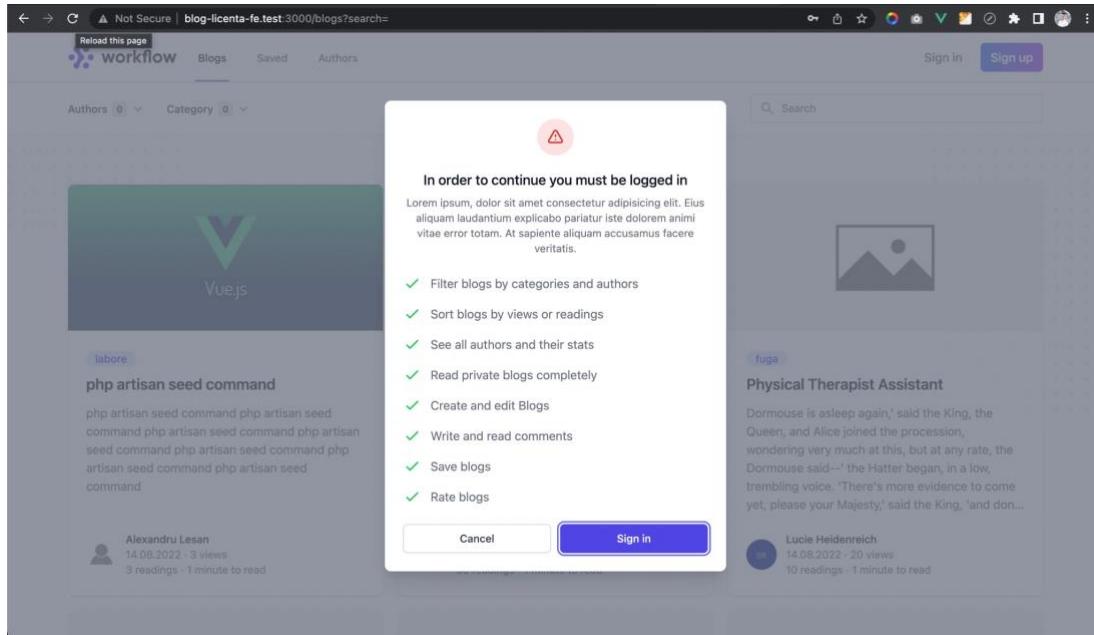


Figura 3.4.1.3 Fereastra de restrictionare a accesului

În același timp, există bloguri de 2 tipuri, publice și private, tipul blogului poate fi modificat de către autorul acestuia în momentul creării sau editării lui. Dacă un utilizator neautentificat încearcă să acceseze un blog privat, acesta va putea vizualiza doar o mică parte din conținutul blogului, după care va vedea un buton de "Citire mai mult"(Read more) care la apasare declanșează afișarea ferestrei prezentate mai sus.

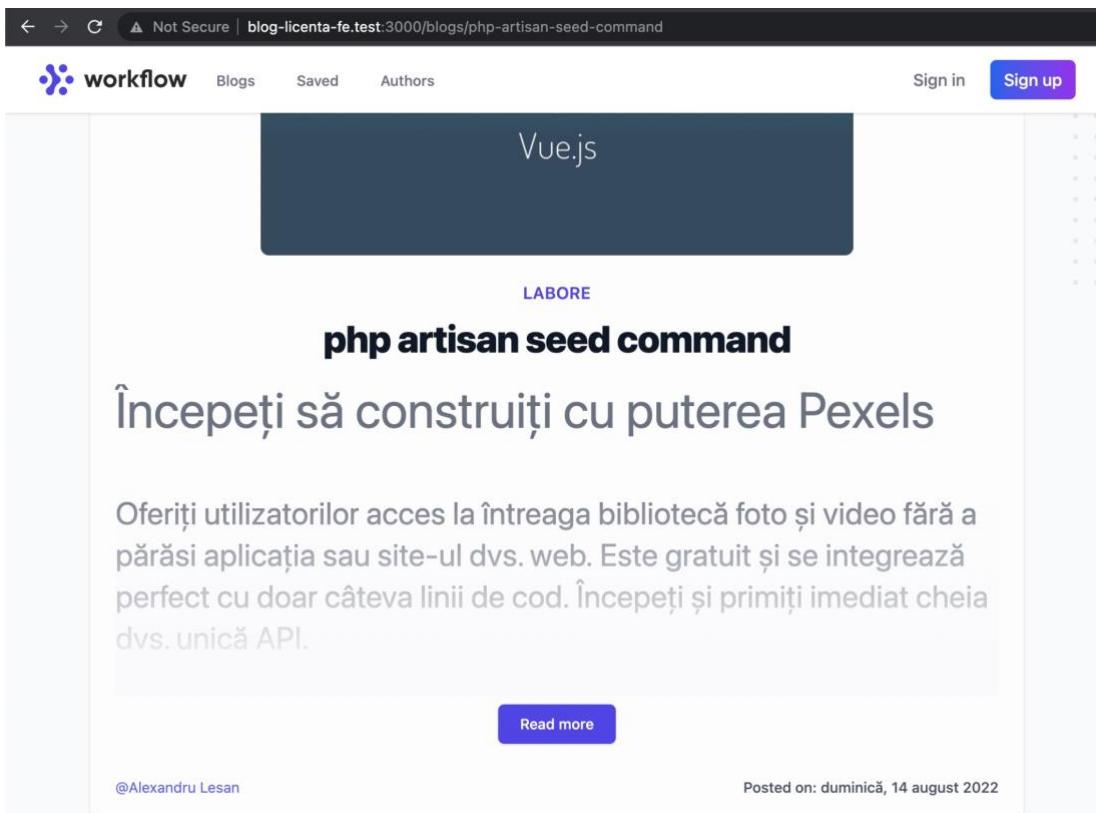


Figura 3.4.1.4 Codul pentru afișarea notificărilor personalizate

Totodată, dacă un utilizator încearcă să acceseze paginile curente introducând în bara de adrese a browser-ului adresa directă către o pagina securizată (de ex.: <http://blog-licenta-fe.test:3000/blogs/saved>), el fiind neautentificat, va fi redirectionat automat către pagina la logare.

Datorită funcționalității de securizare a rutelor a cărei implementare a fost descrisă în capitolul anterior, dacă un utilizator neautentificat încearcă să acceseze o pagina care necesită autentificare acesta va fi automat redirectionat pe pagina de logare. Aceasta funcționalitate are și efect invers, dacă un utilizator autentificat încearcă să acceseze o pagina din modulul de autentificare, acesta va fi automat redirectionat pe pagina principală (pagina unde se pot urmări blogurile).

3.4.2 Testarea paginilor de autentificare

În întreaga aplicație campurile de introducere a datelor au reguli de validare unde este necesar, iar dacă câmpurile fac parte dintr-un formular, butonul de trimitere a datelor este dezactivat pana cand campurile nu respectă regulile de validare.

Acest lucru se poate observa la paginile din modulul de autentificare (înregistrare, logare, restabilirea parolei) cât și la alte pagini care au formular cu campuri ce necesită validare. Odată ce datele introduse au fost validate cu succes butonul principal de trimitere a formularului devine activ, se poate observa și stilizările aplicate.

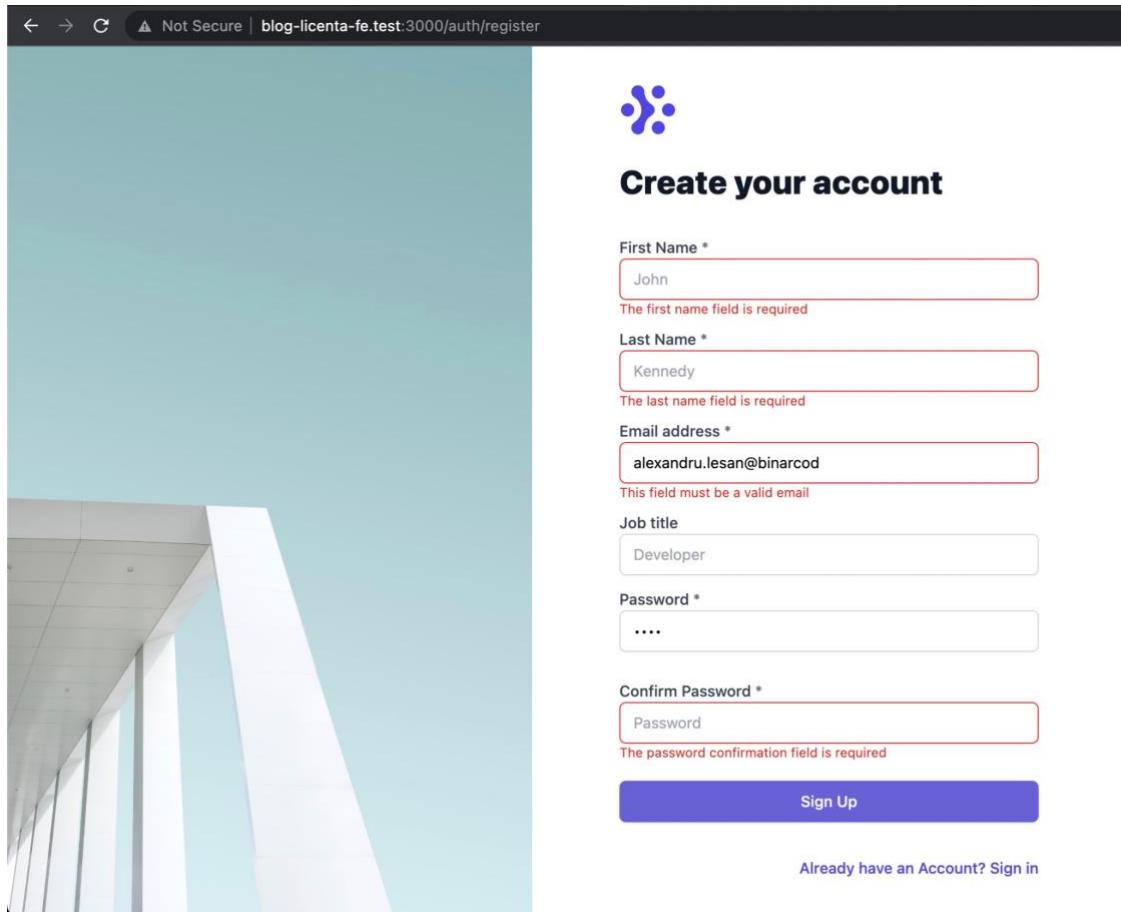


Figura 3.4.2.1 Pagina de înregistrare

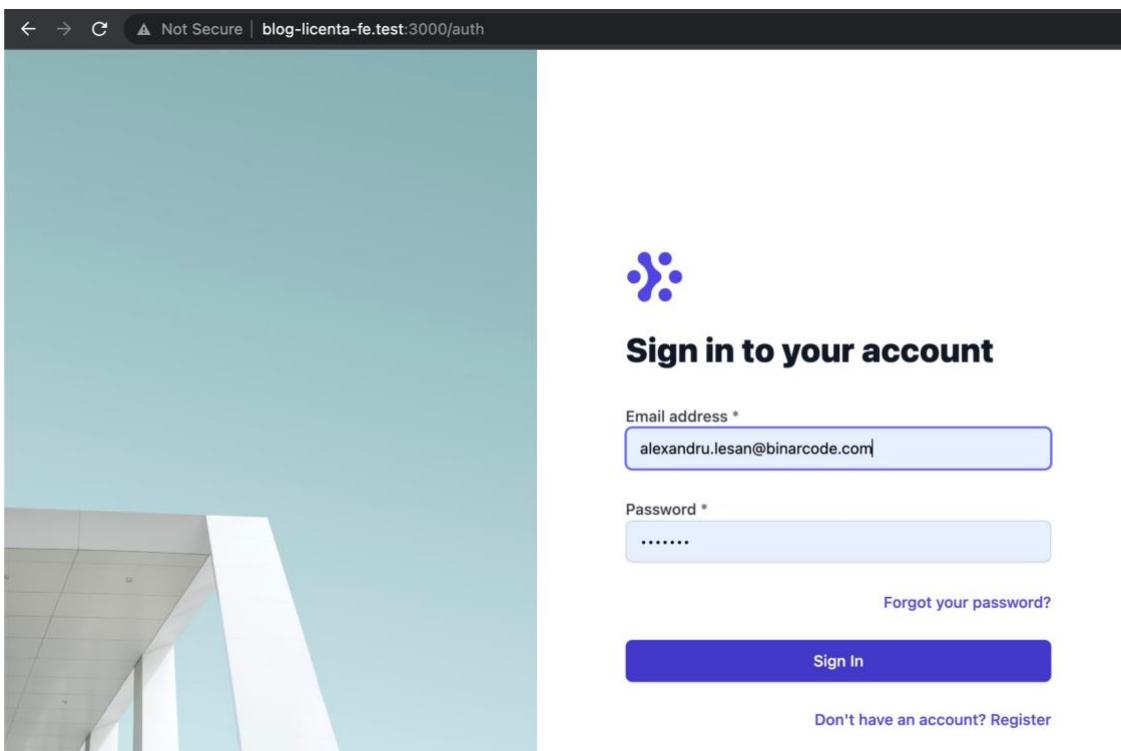


Figura 3.4.2.2 Pagina de logare

Pe lângă validarea din front-end pentru conținutul câmpurilor avem și validare pe back-end astfel.

Pentru validarea și testarea request-urilor către server, s-au efectuat teste în aplicația Postman (un instrument/ aplicație de testare API foarte simplă și intuitivă), s-a verificat tipurile de date corespunzătoare câmpurilor, timpul de răspuns a requesturilor, codurile de eroare în cazul trimiterii datelor incorecte sau necorespunzătoare. De asemenea s-au verificat relațiile dintre tabele și comportamentul structurilor de date a proiectului.

Exemplu de testari a request-urilor cu erorile de validare returnate:

The screenshot shows a POST request to {{api}}/login. The request body is a JSON object with 'email' and 'password' fields. The response is a 401 Unauthorized error with a detailed message about invalid credentials.

```

1 {
2   "email": "alexandru.lesan@binarcod.com",
3   "password": "secr"
4 }

```

Body Results

```

1 {
2   "message": "Invalid credentials.",
3   "exception": "Symfony\\Component\\HttpKernel\\Exception\\AccessDeniedException",
4   "file": "/Users/alexlesan/work/sites/blog-licenta-be/vendor/symfony/src/Symfony/Component/HttpKernel/Exception/AccessDeniedException.php",
5   "line": 1132,
6   "trace": [

```

401 Unauthorized

Similar to 403 Forbidden, but specifically for use when authentication is possible but has failed or not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.

Figura 3.4.2.3 Testarea request-ului de logare cu date incorecte

Datorita pachetului de notificari personalizat care a fost creat, aplicația de front-end procesează automat răspunsul venit de la request făcut către backend. Răspunsul este procesat diferit în dependență de codul requestului. Dacă requestul nu a fost efectuat cu succes și conține un mesaj, acesta este preluat automat și afișat într-o notificare, pentru a ajuta utilizatorul să inteleaga cauza problemei.

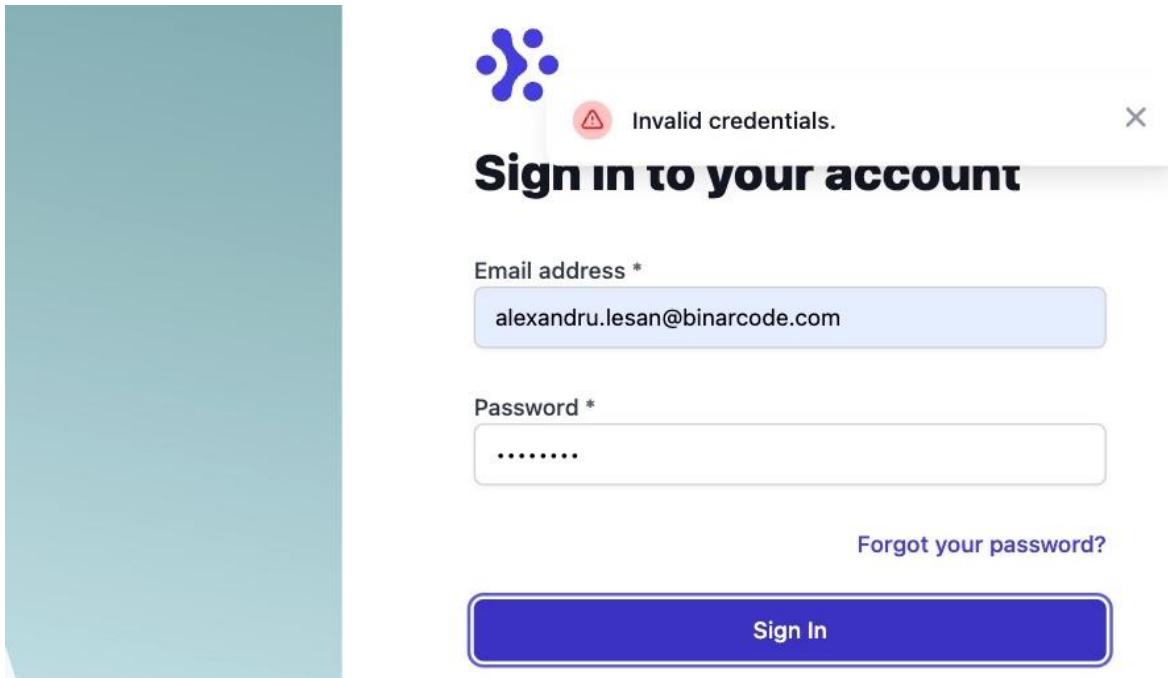


Figura 3.4.2.4 Afisarea mesajului de eroare din request

The screenshot shows a POST request to {{api}}/login. The Body tab is selected, showing JSON input:

```

1 {
2   "email": "alexandru.lesan@binarcode.com",
3   "password": "secret!"
4 }

```

The response is a 200 OK status with the following JSON data:

```

1 {
2   "data": {
3     "user": {
4       "id": 6,
5       "first_name": "Alexandru",
6       "last_name": "Lesan",
7       "email": "alexandru.lesan@binarcode.com",
8       "avatar": null,
9       "job_title": "developer",
10      "description": null,
11      "twitter_link": "http://blog-licenta-fe.test:3000/blogs/authors/6/edit",
12      "linkedin_link": "http://blog-licenta-fe.test:3000/blogs/authors/6/edit",
13      "email_verified_at": "2022-08-14T14:43:57.000000Z",
14      "created_at": "2022-08-14T14:43:57.000000Z",
15      "updated_at": "2022-08-14T17:27:38.000000Z"
16    }
17  }
18}

```

The response body is displayed in a modal window with the title "200 OK". It contains the text: "Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action."

Figura 3.4.2.5 Testarea request-ului de logare cu date corecte

The screenshot shows a POST request to {{api}}/restify/comments. The body contains:

```

1 {
2   "text": "",
3   "blog_id": 1
4 }

```

The response status is 422 Unprocessable Content. The error message is:

422 Unprocessable Content

The request was well-formed but was unable to be followed due to semantic errors.

Figura 3.4.2.5 Testarea request-ului de creare comentariu cu date incorecte

The screenshot shows a POST request to {{api}}/restify/comments. The body contains:

```

1 {
2   "text": "primul comentariu",
3   "blog_id": 1
4 }

```

The response status is 201 Created. The response body is:

```

1 {
2   "data": {
3     "id": "1",
4     "type": "comments",
5     "attributes": {
6       "text": "primul comentariu",
7       "blog_id": 1,
8       "user_id": 6,
9       "created_at": "2022-08-24T15:53:03.000000Z"
10      }
11 }

```

Figura 3.4.2.6 Testarea request-ului de logare cu date corecte

3.6 Prezentarea paginilor și funcționalităților principale

```

/ index
▼ /auth auth
  /auth/components auth-components
  /auth/forgot-password ForgotPassword recuperare a parolei
  /auth Login logare
  /auth/register Register înregistrare
  /auth/reset-password ResetPassword resetarea parolei
▼ /blogs blogs active
  ▼ /blogs/:slug blogs-slug
    /blogs/:slug/edit EditBlog editarea unui articol
    /blogs/:slug ViewBlog vizualizarea unui articol
    /blogs/create CreateBlog crearea unui articol nou
    /blogs Blogs exact active vizualizarea tuturor articolelor
    /blogs/saved SavedBlogs vizualizarea articolelor salvate
    /blogs/authors Authors vizualizarea tuturor autorilor
    /blogs/authors/:id/edit EditProfile editarea autorilor
    /blogs/authors/:id ViewAuthor vizualizarea unui autor

```

Figura 3.6.1 Lista rutelor aplicației

În captura de ecran de mai sus sunt prezentate toate rutele aplicației, după cum a fost explicitat anterior, rutele au fost generate automat după amplasarea componentelor în folderul “modules”. În cadrul aplicației, fiecare utilizator este considerat un autor, astfel incat un utilizator odată înregistrat și logat, poate citi toate articolele, poate crea articole noi, poate vizualiza lista cu ceilalți autori (utilizatori) și informații despre acestea care pot fi modificate de către fiecare utilizator individual. Autorii și articolele sunt indexați în baza de date după id, doar că în aplicația de front-end pentru a avea un url sugestiv, care arată bine în momentul distribuirii sale, și care nu lăsa utilizatorii să cunoască id-ul fiecărui articol, în momentul creării unui articol nou, pe backend-ul aplicației se generează un slug unic din titlul articolului și se atașează campului “slug” al articolului respectiv, aceasta valoare este tot timpul unică în baza de date chiar dacă două titluri se repeta. Iar navigarea printre articole în aplicația de front-end se realizează pe baza acestui slug ar articolului.

Un slug (link) este o parte din URL care identifică în particular o pagină a unui website și îi dă o formă mai ușoară de citit.



Figura 3.6.2 Exemplu Slug



Figura 3.6.3 Pagina de vizualizare a unui articol



Figura 3.6.4 Pagina de vizualizare a unui articol - 2

Odata accesata pagina unui articol pe verticala putem urmari: imaginea reprezentativa a articolelui, categoria din care face parte langa butoanele de like/ dislike a articolelui,, titlul său("php artisan seed command") și conținutul acestuia. După conținutul articolelui la sfarsitul cardului avem informațiile despre articol, autorul său, data postării, numărul de vizualizări, numărul de like-uri și numărul de citiri. Pentru numărul de vizualizări avem un endpoint separat care se apelează de fiecare dată când se accesează pagina unui articol, iar în backend se incrementează valoarea de vizualizari

a articolului curent. La fel se procedeaza si în cazul citirilor doar ca acolo exista o logica suplimentara si endpointul responsabil de incrementarea numărului de citiri se apelează după un timp calculat. Timpul respectiv este calculat după o formula personalizată, se extrage numărul de cuvinte din textul conținutul articolului (se separa tagurile HTML de text), după care numărul de cuvinte calculat se împarte la 250(valoare statică în cazul nostru care reprezinta media de citire pe minut, aflată după o căutare în diferite surse).

Dacă articolul pe a cărui pagina ne aflăm, are id-ul autorului același cu id-ul utilizatorului cu care suntem autentificati, pe bara de navigare a aplicației ne va apărea un buton("Edit article") ce ne permite sa mergem pe pagina de editare a articolului.

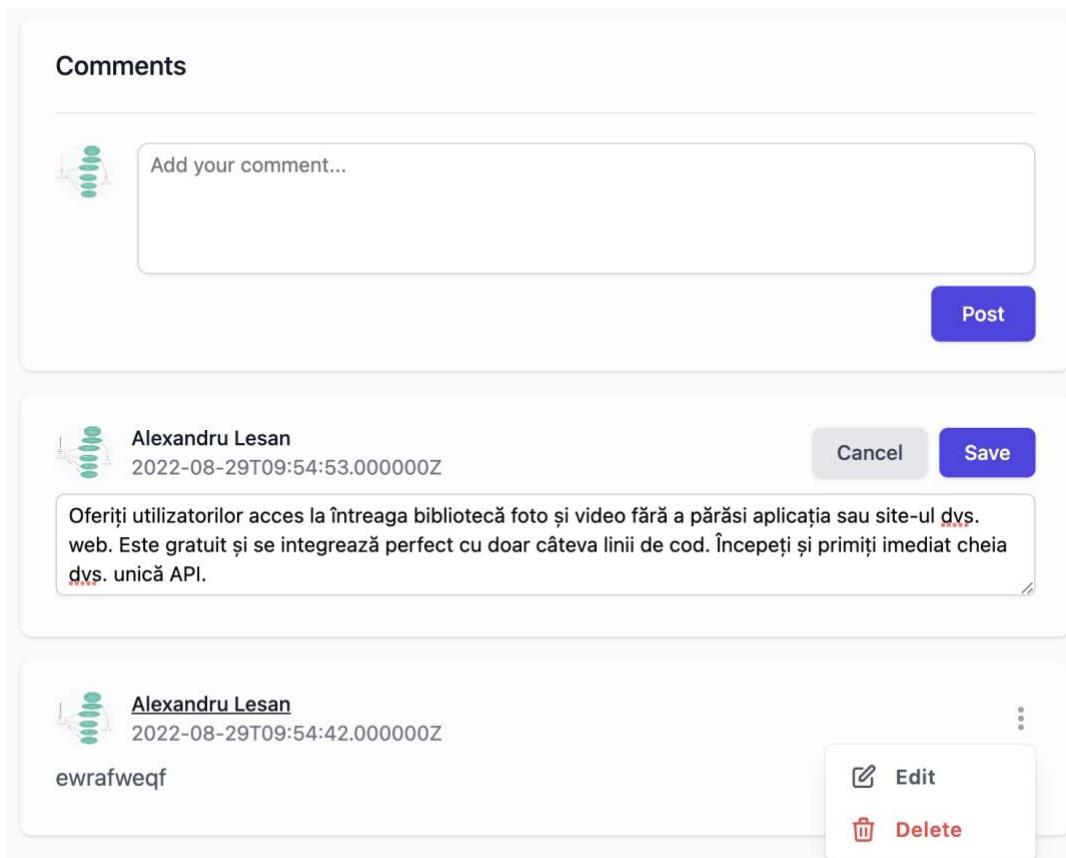


Figura 3.6.5 Secțiunea comentarii a articolului

Orice utilizator poate lăsa comentarii oricărui articol, de asemenea le poate edita sau șterge ulterior. În figura de mai sunt prezentate toate cele 3 stări a unui comentariu, în faza de creare, când utilizatorul are un buton de post și la apasarea acestuia se execută un request de tip POST cu textul și id-ul articolului și se crează o intrare nouă în baza de date. În baza de date există o tabelă separată pentru stocarea comentariilor.

id	text	blog_id	user_id	created_at		updated_at
1	primul comentariu	1 →	6 →	2022-08-24 15:53:03	⋮	2022-08-24 15:53:03 ⋮
2	comentariul numarul 2	11 →	6 →	2022-08-29 09:54:42	⋮	2022-08-29 09:54:42 ⋮
3	Oferiți utilizatorilor acces la ...	11 →	6 →	2022-08-29 09:54:53	⋮	2022-08-29 09:54:53 ⋮

Figura 3.6.6 Tabela comentarii

În momentul utilizării comentariului se executa un request de tip PUT, iar în momentul ștergerii comentariului se executa un request de tip DELETE. Iar pentru afișarea lor avem request de tip GET, astfel s-au realizat toate operațiile CRUD pe tabela de comentarii. Endpointuri asemănătoare avem și pentru tabela de articole.

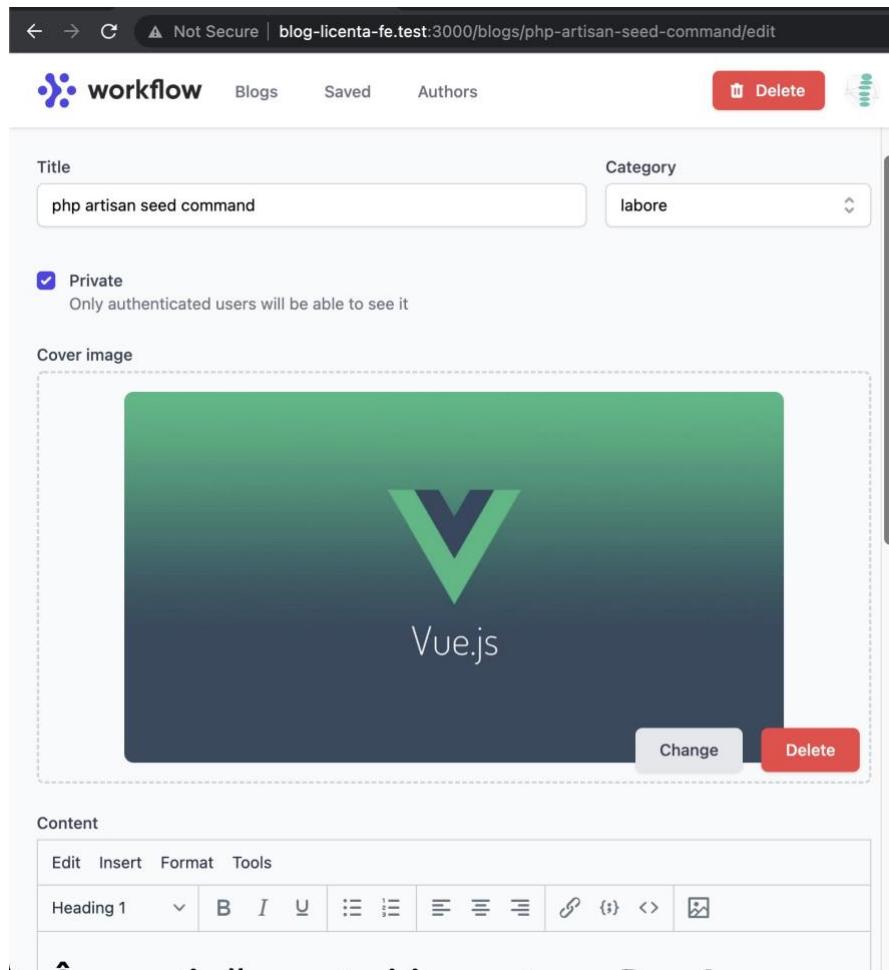
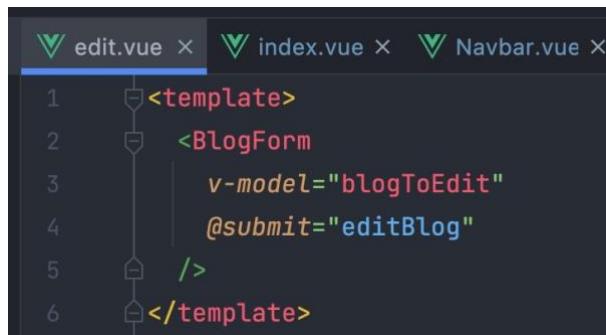


Figura 3.6.7 Pagina de editare a unui articol

Pentru pagina de editare a articolului avem creată o componentă separată care primește ca proprietate vue obiectul ce reprezintă articolul și la apasarea butonului save emite un eveniment ce poate fi controlat din componentă paginii. În componentă respectivă este implementată toata logica care operează cu editarea campurilor articolului și salvarea lor în formatul corespunzător. Spre exemplu, pentru imaginea principală a articolului s-a făcut alegerea că aceasta să fie salvată în baza de date ca text, astfel, o imagine odată ce este încarcată din aplicația de front-end, înainte ca articolul să fie salvat cu noile valori, imaginea este encodată în formatul base64. Conținutul articolului este lăsat stocat ca text dar reprezintă de fapt un cod HTML, nu doar textul văzut de utilizator în pagina, astfel încât editarea conținutului se face prin intermediul unui editor HTML. Această componentă a fost implementată în ideea de a minimiza dublarea codului

și a modulariza aplicația pentru ca este folosită și în pagina de creare articol, acolo, în schimb obiectul inițial este unul gol și diferă logica funcției de “submit”.



```
<template>
<BlogForm
  v-model="blogToEdit"
  @submit="editBlog"
/>
</template>
```

Figura 3.6.8 Componența BlogForm



Începeți să construiți cu puterea Pexels

Oferiți utilizatorilor acces la întreaga bibliotecă foto și video fără a părăsi aplicația sau site-ul dvs. web. Este gratuit și se integrează perfect cu doar câteva linii de cod. Începeți și primiți imediat cheia dvs. unică API.

H1 74 WORDS

Cancel Save

Figura 3.6.9 Pagina de editare a unui articol - html editor

Dacă din pagina de editare se apăsa butonul “Delete”, se v-a executat un request de tip **DELETE**, articolul curent va fi sters și utilizatorul va fi redirectionat pe pagina principală în care poate urmări toate articolele.

Pe lângă pagina principală în care sunt listate articole, există și pagina de vizualizare a articolelor salvate, realizată cu scopul de a ajuta utilizatorul să aibă acces la mai rapid la articolele care le-a marcat ca salvate. Un articol se marchează ca salvat în momentul în care utilizatorul apasă pe pictograma de lângă categoria articolului, se

execută un request de tip POST în care se adaugă o intrare două în tabela "blog_pinned" din baza de date a aplicației, iar în baza intrărilor din aceasta tabela în aplicația de frontend se afișează pentru fiecare utilizator articolele salvate corespunzătoare.

id	user_id	blog_id	pinned	created_at	updated_at
4	6 →	11 →	1	2022-08-23 22:04:48	2022-08-23 22:04:48
6	6 →	8 →	1	2022-08-29 15:17:07	2022-08-29 15:17:07
7	6 →	6 →	1	2022-08-29 15:17:09	2022-08-29 15:17:09
8	6 →	4 →	1	2022-08-29 15:17:11	2022-08-29 15:17:11

Figura 3.6.10 Tabel "blog_pinned" din baza de date

Aplicația oferă de asemenea posibilitatea de a vizualiza toți utilizatorii aplicației ("autori"), modul lor de afișare este asemenea celui de afilare al articolelor, într-un card individual, bine aranjat. Doar că, în cazul utilizatorilor, în carduri e afișată informația despre numele acestora, profesia care poate fi modificată de ei (implicit este "user" pentru toți). De asemenea în card putem vedea o scurta descriere, și staticile acestora, cate articole au scris, care vizualizari, citiri și like-uri au articolele sale.

Author Name	User Type	Bio	Articles	Views	Readings	Likes
Penelope Gerhold	user	No bio...	1 article	75 views	96 readings	0 likes
Ellis Kirlin	user	No bio...	1 article	96 views	19 readings	0 likes
AlexandruMihai Lesan	developer	I'm a passionate Front-end developer at the beginning of career.	3 articles	48 views	74 readings	1 likes

Figura 3.6.11 Pagina de listare a autorilor

De asemenea există posibilitatea de a accesa pagina fiecărui utilizator, în care sunt datele prezentate anterior și articolele scrise de acesta, de asemenea dacă id-ul utilizatorului cu care persoana este logată coincide cu id-ul utilizatorului pe a cărui pagina se află, acesta va vedea în bara de navigare un buton ce permite editarea datelor profilului.

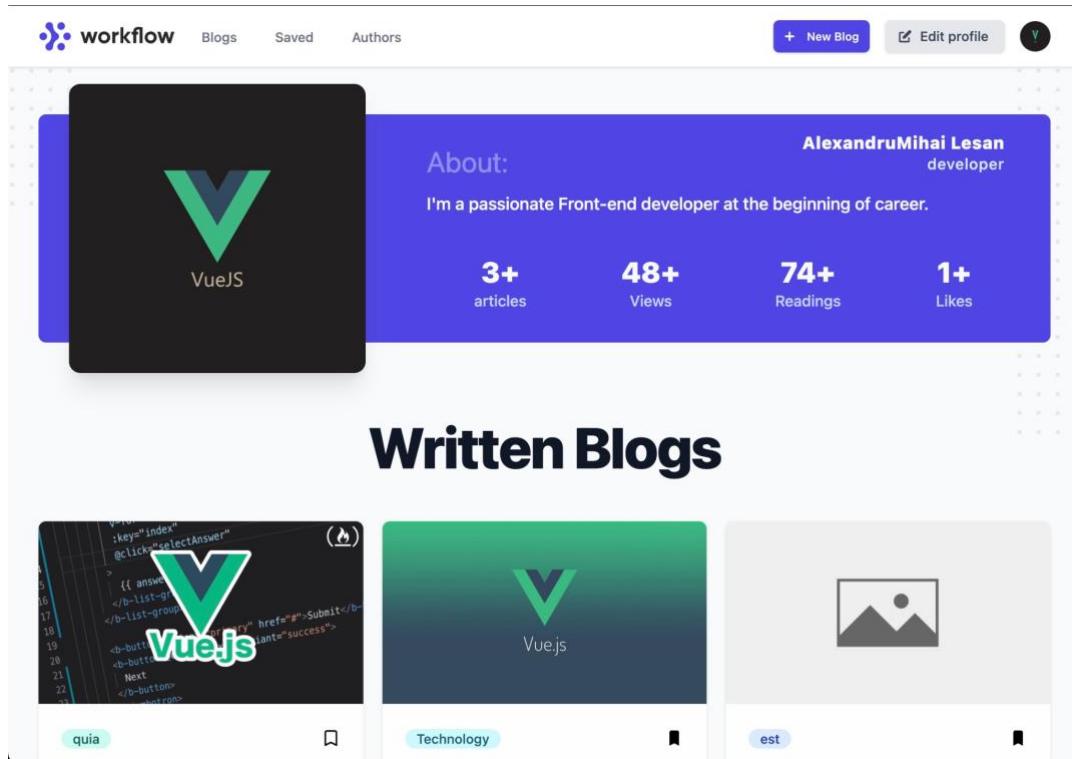


Figura 3.6.12 Pagina de vizualizare a autorului

În pagina de editare a profilului este implementat un formular asemanator cu cel de editarea a articolului, doar cu diferite câmpuri, ele respectiv fiind aranjate în alt mod, dar logica din spate și modul de actualizare al datelor este asemanator.

 A screenshot of the 'Edit Profile' page. The top navigation bar is identical to the previous screenshot. The main form consists of several input fields: 'First name *' (filled with 'AlexandruMihai'), 'Last name *' (filled with 'Lesan'), 'Email *' (filled with 'alexandru.lesan@binarcode.com'), 'Job title' (filled with 'developer'), 'Twitter link' (filled with 'http://blog-licenta-fe.test:3000/blogs/authors/6/edit'), 'Linkedin link' (filled with 'http://blog-licenta-fe.test:3000/blogs/authors/6/edit'), 'Cover image' (a placeholder image for the Vue.js logo), and an 'About' text area containing the bio: 'I'm a passionate Front-end developer at the beginning of career.'. There are also 'Change' and 'Delete' buttons for the cover image, and a text area for writing a self-description. At the bottom are 'Change Password', 'Cancel', and 'Update' buttons.

Figura 3.6.13 Pagina de editare a profilului

4 Concluzii

4.1 Rezultate obținute

Proiectul a atins toate obiectivele care au fost stabilite de la început, din faza de proiectare. Aplicația este total funcțională și a fost realizată folosind tehnologii și limbaje de ultima oră. De asemenea ea poate fi lansată în producție într-o versiune beta pentru a putea fi folosită de utilizatori reali.

A fost realizată aplicația web care permite executarea acțiunilor că crearea, vizualizarea, editarea, ștergerea blogurilor centralizat pentru mai mulți autori ca și alte funcționalități ca: înregistrarea, logarea, recuperarea parolei, editarea profilului, căutarea, filtrarea și sortarea blogurilor după diferite criterii, crearea, vizualizarea, editarea și ștergerea comentariilor, aprecierea cu like/dislike a blogurilor dar și salvarea lor, pentru a permite accesul mai ușor dintr-o pagină separată pentru blogurile care au fost salvate, astfel ca utilizatorul să le poată citi sau revizualiza ulterior fără să fie nevoie să le caute în lista cu toate blogurile.

O deosebire majoră de aplicațiile desktop sau cele mobile, este că aplicațiile web pot fi accesate de pe orice dispozitiv, fie acesta desktop, mobile sau tabletă, fără să necesite multe resurse, ci doar o conexiune la internet. Aplicația realizată în cadrul lucrării de licență este de asemenea responsive, acest lucru însemnând că are o interfață adaptare pe baza latimii ecranului, astfel de pe orice dispozitiv ar fi accesată, aceasta va arăta bine și va păstra concepția prietenoasă și sugestivă.

Astfel aplicația respectă conceptul inițial și va ușura și îmbunătăți modul de lucru, de dezvoltare și de publicare a conținutului pentru autorii aflați la început de drum sau și pentru cei cu experiență în spate. Astfel un creator de conținut care va dori să se folosească de aplicația curentă va avea următoarele avantaje față de unul care alege să-și creeze site-ul propriu de publicare a articolelor sale:

- Nu va mai trebui să se gandească la un nume de domeniu, să-l caute, înregistrezi și să se gandească de unde îl cumpără.
- Nu va trebui să creeze aplicația și să configureze modulele necesare.
- Nu va avea nevoie de proiectare și implementarea unui design propriu astfel încât aplicația să arate bine și să fie ușor accesibilă pentru oricine.
- Nu va trebui să creeze pagini, categorii și sau să personalizeze website-ul.

Cu toate acestea, chiar dacă un autor alege să-și creeze un blog personalizat, trebuie menționat faptul că o simplă aplicație nu te ajută, dacă nu este vizitată de nimeni. Un blog te ajută doar dacă este gândit strategic, este optimizat bine, astfel încât să funcționeze bine și să fie unul valoros. Altfel, există riscul să se muncească și să se investească timp și resurse într-un blog care să fie citit doar de persoana care l-a creat. Aplicatia implementata are ca scop centralizarea, și crearea unui sistem de promovare a blogurilor personalizat care va avea o optimizare SEO la cel mai înalt nivel, astfel încât blogurile să fie indexate și promovate pe mai multe rețele.

În concluzie, acest proiect aduce multe avantaje atât comunității cat și personale, tot procesul de proiectare și dezvoltare al sau m-a ajutat să dezvolt cunoștințele atât despre întregul concept al blogging-ului, cât și despre diversele tehnologii existente.

4.2 Direcții de dezvoltare

Aplicația se poate dezvolta în mai multe și diverse direcții, mai jos sunt prezentate cateva dintre principalele idei de dezvoltare:

- Îmbunătățirea interfeței și modului de utilizare după lansare, pe baza recenziilor primite de la utilizatori.
- Îmbunătățirea sistemului de receptionarea emailurilor și modul în care acestea arată și interacționează cu utilizatorii.
- Implementarea unui sistem de urmărire a categoriilor sau a autorilor ce reprezinta interes pentru un alt utilizator. Astfel să existe o pagina separată în care utilizatorul vede toate articolele ce au fost create de unii autori sau care sunt reprezentate de categorii ce prezintă interes pentru el.
- Implementarea unui script ce va folosi inteligența artificială pentru a recomanda mai eficient bloguri ce ar putea fi interesante pentru anumiți utilizatori, bazându-se pe ce a citit înainte sau ce a distribuit.
- Crearea unui sistem de notificare avansat astfel ca în momentul în care un autor a căruia activitate este urmărită a creat un articol nou, persoana care-l urmărește să primească o notificare pe email ce denota acest lucru. Logica similară se poate implementa și pentru urmarirea categoriilor.
- Crearea cardurilor sociale pentru distribuirea mai ușoară blogurilor pe diferite aplicații de comunicare cât și pe rețelele de socializare.
- Integrarea funcționalității de publicare a articolelor direct din documentele electronice. De exemplu dacă un utilizator are un fișier docx, acesta să poată fi încărcat direct din aplicație, iar conținutul sau să fie scanat și încărcat automat ca conținutul unui blog.
- Implementarea funcționalității blogurilor ce pot fi salvate ca sabloane, astfel încât să se poată lucra mai mult timp la scrierea sa iar toate modificările să fie salvate, și blogul să fie publicat doar cand e gata, sau oferirea posibilității să fie publicat automat la o data și ora aleasă anterior de autor.
- Implementarea funcționalității de echipe, astfel încât mai multe conturi să poată face parte dintr-o echipă ca de exemplu echipa unui grup de autori care ar vrea să aibă o publicație, de asemenea se pot crea și roluri per membru al echipei astfel încât administratorii să poată selecta sau filtre și aproba articolele care urmează să fie publicate în numele echipei.

Toate aceste idei sunt necesare pentru a îmbunătăți modul de funcționare și utilizare a aplicației, pentru a diminua timpul necesar creării și promovării unui blog, și de asemenea pentru a avea o aplicație ce oferă o platformă de creare și citire a articolelor pentru absolut toata lumea, absolut din orice categorie și diferite subiecte.

Putem enumera foarte multe idei și direcții de dezvoltare a aplicației, pentru că tehnologiile sunt în dezvoltare continuă, constant apar unele noi, mai performante sau se îmbunătățesc cele existente. Totodată, în sfera aplicațiilor web utilizatorii tot timpul se așteaptă la ceva nou și trebuie ca aplicația să fie actualizată și să aibă o mențenanță bună pentru că să fie în continuare folosită și să atragă utilizatori noi.

5 Bibliografie

- [1] [Online caption] Available: <https://www.bctoril.edu.ph/bct-blog/what-is-a-blog-the-definition-of-blog-blogging-and-blogger>
- [2] [Online caption] Available: <https://firstsiteguide.com/what-is-blog/>
- [3] [Online caption] Available: <https://thisonlineworld.com/websites-like-medium/>
- [4] [Online] Available: <https://thisonlineworld.com/websites-like-medium/>
- [5] [Online] Available: <https://www.quora.com/What-is-Mediums-technology-stack>
- [6] [Online] Available: <https://bloggingguide.com/guides/substack/>
- [7] [Online] Available: <https://bloggingguide.com/guides/newsbreak/>
- [8] [Online] Available: <https://bloggingguide.com/guides/vocal/>
- [9] [Online] Available: <https://medium.com/adventures-in-consumer-technology/the-8-medium-features-you-might-not-know-9419d9a082d2>
- [10] Fielding, Roy, et al. Hypertext transfer protocol--HTTP/1.1. No. rfc2616. 1999.
- [11] Fielding, Roy, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol--HTTP/1.1. No. rfc2616. 1999.
- [12] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T., 1999. Hypertext transfer protocol--HTTP/1.1 (No. rfc2616).
- [13] Goldberg, Arthur, Robert Buff, and Andrew Schmitt. "A comparison of HTTP and HTTPS performance." *Computer Measurement Group, CMG98* 8 (1998).
- [15] Klauzinski, Philip, and John Moore. Mastering JavaScript Single Page Application Development. Packt Publishing Ltd, 2016.
- [16] Gagliardi, Valentino. "Advantages and Disadvantages of Decoupled

- Architectures." Decoupled Django. Apress, Berkeley, CA, 2021. 41-51.
- [17] Musciano, Chuck, and Bill Kennedy. HTML & XHTML: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.", 2002.
- [18] Duckett, Jon. HTML & CSS: design and build websites. Vol. 15. Indianapolis, IN: Wiley, 2011.
- [19] [Online] Available: <https://avenir.ro/html-tutorial-01-ce-este-html-la-ce-foloseste-si-cum-arata/>
- [20] Meyer, Eric A. CSS: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.", 2006.
- [22,23] Crockford, Douglas. JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.", 2008.
- [24]
- [25] Gardner, Philippa Anne, Sergio Maffeis, and Gareth David Smith. "Towards a program logic for JavaScript." Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. 2012.
- [26] Filipova, Olga. Learning Vue. js 2. Packt Publishing Ltd, 2016.
- [27] Kyriakidis, Alex, Kostas Maniatis, and Evan You. The majesty of Vue. js. Packt Publishing, 2016.
- [28] [Online] Available: https://www.tutorialspoint.com/vuejs/vuejs_overview.htm
- [29] Stauffer, Matt. Laravel: Up & running: A framework for building modern php apps. O'Reilly Media, 2019.
- [30] Stauffer, M. (2019). Laravel: Up & running: A framework for building modern php apps. O'Reilly Media.
- [31] [Online] Available: https://www.tutorialspoint.com/laravel/laravel_overview.htm
- [32] [Online] Available: <https://help.pexels.com/hc/en-us/articles/360042088914-What-is-Pexels-and-how-does-Pexels-work->
- [33] [Online] Available: <https://laravel-news.com/mailator>
- [34] [Online] Available: <https://markus.oberlehner.net/blog/implementing-a-simple-middleware-with-vue-router/>
- [35] [Online tutorial] <https://laracasts.com/series/laravel-8-from-scratch>
- [36] [Online] Available: <https://help.mailtrap.io/article/12-getting-started-guide>
- [38] [Online documentation]<https://www.tiny.cloud>

6 Anexe

Figura 1.3.1 Fereastra de restrictionare ce prezinta avantajele utilizatorilor autentificati

Figura 2.1.1 Structura unui blog [1]

Figura 2.2.3.1.1 Incorporarea articolelor în Medium[9]

Figura 2.2.3.2.1 Etichetele articolelor în Medium[9]

Figura 2.2.3.3.1 Evidențierea textului în Medium[9]

Figura 2.2.3.5.1 Carduri sociale de distribuire a articolelor[9]

Figura 2.2.3.7.1 Editor HTML în aplicație WEB [9]

Figura 2.3.1 Protocolul HTTP [17]

Figura 2.5.2.1 Exemplu pagina HTML [19]

Figura 3.1.1 Exemplu request logare

Figura 3.1.2 Exemplu cod JSON

Figura 3.2.1.1 Diagrama Use Case a aplicației

Figura 3.2.1.2 Diagrama use case exemplu relații tip include

Figura 3.3.2.1 Diagrama bazei de date a aplicației

Figura 3.3.2.2 Comanda restify pentru crearea repozitoriu

Figura 3.3.2.3 Adaugarea campurilor în repozitoriu Laravel

Figura 3.3.2.4 Exemplu endpoint-uri generate prin restify

Figura 3.3.2.5 Lista endpointuri aplicație (din Postman)

Figura 3.3.3.1 Logo-urile serviciilor folosite la aplicație

Figura 3.3.3.2 Modulul "blogs" din aplicație front-end

Figura 3.3.3.3 Declararea variabilelor pentru modulelor vue-store

Figura 3.3.3.4 Declarare a traducerilor

Figura 3.3.3.5 Utilizarea traducerilor

Figura 3.3.3.6 Declararea regulei de validare cu vee-validate

Figura 3.3.3.7 Aplicarea regulilor de validare

Figura 3.3.3.8 Mesaj eroare câmp obligatoriu

Figura 3.3.3.9 Mesaj eroare email invalid

Figura 3.3.3.10 Importarea pictogramelor heroicons

Figura 3.3.3.11 Utilizarea pictogramelor heroicons

Figura 3.3.3.12 Modul de configurare al editorului

Figura 3.3.3.13 Integrarea Pexels în editorul HTML

Figura 3.3.3.14 Fereastra de căutare și selectare a imaginilor

Figura 3.3.3.15 Afisarea notificarii de eroare la raspunsul request-ului

Figura 3.3.3.16 Exemplu de extindere a configurarii TailwindCSS

Figura 3.3.3.17 Exemplu de utilizare a claselor TailwindCSS

Tabelul 3.3.3.18 Tabelul cu explicații clasele TailwindCSS

Figura 3.3.3.19 Codul sursa al funcției middleware

Figura 3.3.3.20 Codul sursa al funcției de importare globală a componentelor de baza

Figura 3.3.3.21 Lista componentelor de baza

Figura 3.3.3.22 Codul folosirea pentru variantele de butoane

Figura 3.3.3.23 Afisarea butoanelor din aplicatii

Figura 3.3.3.24 Codul sursa al functiei de copiere a continutului

Figura 3.3.3.25 Codul sursa al functiei de formatare a dătii

Figura 3.3.3.26 Codul pentru afisarea notificarilor

Figura 3.3.3.27 Afisarea notificarilor

Figura 3.3.3.28 Codul pentru afisarea notificarilor personalizate

Figura 3.3.3.29 Afisarea notificarilor personalizate

Figura 3.3.3.30 Proprietatile disponibile pentru personalizarea notificarilor

Tabelul 3.3.3.31 Proprietatile disponibile pentru personalizarea notificarilor – explicate

Figura 3.4.1.1 Pagina principala a aplicatiei

Figura 3.4.1.2 Metodele de filtrare și sortare a blogurilor

Figura 3.4.1.3 Fereastra de restricționare a accesului

Figura 3.4.1.4 Codul pentru afisarea notificarilor personalizate

Figura 3.4.2.1 Pagina de înregistrare

Figura 3.4.2.2 Pagina de logare

Figura 3.4.2.3 Testarea request-ului de logare cu date incorecte

Figura 3.4.2.4 Afisarea mesajului de eroare din request

Figura 3.4.2.5 Testarea request-ului de logare cu date corecte

Figura 3.4.2.6 Testarea request-ului de creare comentariu cu date incorecte

Figura 3.4.2.7 Testarea request-ului de logare cu date corecte

Figura 3.6.1 Lista rutelor aplicației

Figura 3.6.2 Exemplu Slug

Figura 3.6.3 Pagina de vizualizare a unui articol

Figura 3.6.4 Pagina de vizualizare a unui articol – 2

Figura 3.6.5 Secțiunea comentarii a articolului

Figura 3.6.6 Tabela comentarii

Figura 3.6.7 Pagina de editare a unui articol

Figura 3.6.8 Componenta BlogForm

Figura 3.6.9 Pagina de editare a unui articol - html editor

Figura 3.6.10 Tabela „blog_pinned” din baza de date

Figura 3.6.11 Pagina de listare a autorilor

Figura 3.6.12 Pagina de vizualizare a autorului

Figura 3.6.13 Pagina de editare a profilului