

Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Platformă Web pentru gestionarea unei librării

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul Electronică și Telecomunicații
programul de studii de licență *Tehnologii și Sisteme de Telecomunicații*

Conducător științific

Conf. Dr. Ing. Dan GALAȚCHI

Absolvent

Cristina-Georgiana PETRACHE

2023

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **PETRACHE M. Cristina-Georgiana, 444C**

1. Titlul temei: Platformă Web pentru gestionarea unei librării

2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare):

Lucrarea are drept scop proiectarea și implementarea unei platforme web pentru gestionarea unei librării. Aplicația va fi formată din următoarele componente: interfețele utilizator, baza de date necesară gestionării și stocării informațiilor, API și logica de business. Ca funcționalități, platforma va conține sistem de autentificare și înregistrare a utilizatorilor cu verificare de e-mail și sistem de recuperare a parolei, secțiune de comentarii și recenzii ale utilizatorilor pentru produse, gestionarea coșului de cumpărături, posibilitatea adăugării produselor într-o listă de favorite, sistem complet de plasare și procesare a comenzilor, formular de retur, de închiriere și de donare, mecanism de promoții și recomandări bazate pe experiențele utilizatorului, secțiune de contact pentru utilizator, secțiune de căutare și sortare cu ajutorul unor filtre și gestionarea datelor personale. Pentru realizarea proiectului se va folosi ca limbaj de programare pentru partea de backend PHP prin intermediul frameworkului Laravel, iar pentru partea de frontend se vor folosi JavaScript, prin intermediul frameworkului Vue.js, HTML și CSS. Pentru implementarea bazei de date se va folosi MySQL.

3. Discipline necesare pt. proiect:
Baze de date, Programare obiect-orientată, Programarea calculatoarelor, Structuri de date și algoritmi

4. Data înregistrării temei: 2022-11-21 11:27:05

Conducător(i) lucrare,
Conf. dr. ing. Dan GALAȚCHI

Student,
PETRACHE M. Cristina-Georgiana

Director departament,
Conf. dr. ing. Șerban OBREJA

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **cb1a40af0d**

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul “*Platformă Web pentru gestionarea unei librării*”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică și Telecomunicații*, programul de studii *Tehnologii și Sisteme de Telecomunicații* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 27.06.2023

Absolvent *Cristina-Georgiana PETRACHE*



(semnătura în original)

Cuprins

Listă figuri	9
Listă acronime	11
Introducere	13
1. Descrierea problemei abordate	15
1.1. Descrierea domeniului din care face parte tema de licență	15
1.2. Formularea problemei	16
1.3. Studiu asupra realizărilor similare din domeniu	19
1.3.1. Funcționalități oferite de site-ul librăriei Libris ^[7]	19
1.3.2. Funcționalități oferite de site-ul librăriei CLB ^[8]	20
1.3.3. Funcționalități oferite de site-ul librăriei Mihai Eminescu ^[9]	21
1.3.4. Comparatie între cele trei realizări	21
1.4. Stabilirea cerințelor funcționale și nefuncționale ale aplicației	22
2. Stadiul actual în domeniu și selectarea soluției tehnice	25
2.1. Stadiul actual al tehnologiilor utilizate pentru dezvoltarea soluției	25
2.1.1. HTML	25
2.1.2. CSS	26
2.1.3. Bootstrap	26
2.1.4. Vue.js	27
2.1.5. Laravel	27
2.1.6. MySQL	29
2.1.7. API	30
2.1.8. Apache	30
2.2. Prezentarea tehnologiilor și mediilor de dezvoltare alese	31
2.2.1. HTML	31
2.2.2. CSS	32
2.2.3. Bootstrap	32
2.2.4. Vue.js	33
2.2.5. Laravel	34
2.2.6. MySQL	35
2.2.7. REST API	35
2.2.8. Apache	36
2.2.9. PhpStorm	36
2.2.10. DataGrip	37
3. Implementarea aplicației	39
3.1. Implementarea bazei de date	39
3.2. Implementarea funcționalităților	41

3.2.1. Controller.....	41
3.2.2. Service	52
3.2.3. Mail.....	53
3.3. Implementarea interfeței grafice	54
Concluzii	69
Bibliografie	71
Anexe	75
Anexa 1 – Migrări	75
Anexa 2 – Modele	83
Anexa 3 - Controllere.....	89
Anexa 4 – Servicii.....	113
Anexa 5 - Mail	117
Anexa 6 – Componente Vue	118
Anexa 7 – Configurări Javascript.....	160
Anexa 8 – Laravel Blade.....	161
Anexa 10 – Rute.....	166
Anexa 11 – Fișier environment	168

Listă figuri

Fig. 1.1 – Organigrama fluxului de activități în cazul utilizatorilor	18
Fig. 1.2 – Organigrama fluxului de activități în cazul administratorului.....	19
Fig. 2.1 – Structura unui document HTML ^[13]	25
Fig. 2.2 – Arhitectura MVC ^[26]	28
Fig. 2.3 – Modelul client-server ^[28]	29
Fig. 2.4 – Modul în care funcționează API ^[4]	30
Fig. 3.1 – Arhitectura bazei de date	39
Fig. 3.2 – Funcția index din CategoriaController	42
Fig. 3.3 – Funcția show din CategoriaController.....	43
Fig. 3.4 – Funcția store din CategoriaController	44
Fig. 3.5 – Funcția update din CategoriaController.....	45
Fig. 3.6 – Funcția destroy din CategoriaController	46
Fig. 3.7 – Funcția getFilters din CarteController	47
Fig. 3.8 – Funcția getRecommendations din CarteController	48
Fig. 3.9 – Funcția sendContactMail din Controller	49
Fig. 3.10 – Funcția changePassword din UserController.....	50
Fig. 3.11 – Funcțiile login și logout din UserController	51
Fig. 3.12 – Funcția setProperties din CarteComandaService.....	52
Fig. 3.13 – Funcția checkPassword din UserService	53
Fig. 3.14 – Clasa AutentificareEmail	54
Fig. 3.15 – Pagina de căutare	56
Fig. 3.16 – Pagină coș	57
Fig. 3.17 – Pagină favorite	58
Fig. 3.18 – Pagină profil.....	59
Fig. 3.19 – Formular de înregistrare	60
Fig. 3.20 – Template din BookCarousel.Vue.....	61
Fig. 3.21 – Import din BookCarousel.Vue	61
Fig. 3.22 – Numele, componentele și variabilele transmise din BookCarousel.vue.....	62
Fig. 3.23 – data() din BookCarousel.vue	62
Fig. 3.24 – computed din BookCarousel.vue.....	62
Fig. 3.25 – mounted() din BookCarousel.vue	63
Fig. 3.26 – Funcția addFavorite din BookCarousel.vue	63
Fig. 3.27 – watch din BookList.vue	64
Fig. 3.28 – Secțiunea style din BookList.vue	65
Fig. 3.29 – Header pagina-principală.....	65
Fig. 3.30 – Body pagina-principală.....	66
Fig. 3.31 – Style din pagina-principală	66

Listă acronime

API = Application Programming Interface (Interfață de programare a aplicației)

CLI = Command Line Interface

CRUD = Create-Read-Update-Delete

CSRF = Cross-Site Request Forgery

CSS = Cascading Style Sheets

CSV = Comma-separated values

DOM = Document Object Model

HTML = HyperText Markup Language

HTTP = Hypertext Transfer Protocol

MVC = Model-View-Controller

nr. = număr

ORM = Object-Relational Mapping

pg. = pagini

PHP = PHP: Hypertext Preprocessor

REST = Representational State Transfer

SQL = Structured Query Language (Limbaj de interogare structurat)

URL = Uniform Resource Locator

Introducere

Ceea ce m-a motivat să aleg această temă, aceea a dezvoltării unei platforme web comerciale, o reprezintă ideea traiului într-o eră a digitalizării în care încet, încet orice activitate de zi cu zi este trecută în mediul online. Datorită dorinței omului de a dispune de toate facilitățile necesare organizării și desfășurării activităților zilnice, cum ar fi realizarea cumpărăturilor, posibilitatea accesului la activități recreative, dar și ceea ce ține de activitatea profesională a indivizilor, internetul a devenit un mediu propice informării și promovării a ceea ce ne înconjoară. Pentru a putea beneficia de o vizibilitate și o reclamă mai bună, firmele sau micile întreprinderi recurg la realizarea de site-uri web sau platforme web, prin intermediul tehnologiilor web, pentru a putea deveni mai cunoscute în rândul oamenilor și de a beneficia de mai mult succes.

Proiectul își propune dezvoltarea unei platforme web comerciale destinată unei librării, care permite utilizatorilor să-și creeze cont în cadrul aplicației și să se conecteze pe contul deja creat pentru a cumpăra și dona cărți, cea din urmă fiind disponibilă prin completarea unui formular. După înregistrare, aceștia primesc, pe e-mailul pe care l-au folosit, un mesaj pentru activarea contului. Își vor putea recupera parola în cazul în care aceasta este uitată sau pierdută. În cadrul platformei, aceștia au posibilitatea să evalueze cărțile prin intermediul unei secțiuni unde pot lăsa comentarii și note acestora. De asemenea, pot introduce și șterge cărțile care prezintă interes pentru ei într-o listă de favorite sau le pot returna pe cele care nu corespund așteptărilor lor. Utilizatorii pot adăuga cărți în coșul de cumpărături în care pot ajusta cantitatea sau șterge produsele înainte de finalizarea comenzii, iar după plasarea unei comenzi, utilizatorii primesc pe e-mail un mesaj de confirmare al acesteia. Totodată, în cadrul platformei există o secțiune cu promoții unde se găsesc cărțile care dispun de reduceri, dar și o secțiune cu noutăți unde se găsesc ultimele cărți adăugate spre vânzare. Utilizatorii primesc recomandări de cărți, în funcție de autor și categorie, bazate pe paginile cărților pe care le accesează. O altă funcționalitate din cadrul aplicației o reprezintă zona de contact pentru utilizatori, unde aceștia pot pune întrebări în cazul în care au nelămuriri sau sugestii cu privire la îmbunătățirea experienței în cadrul platformei. Totodată, există și o secțiune de căutare, de sortare și de filtrare a cărților. De asemenea, clienții au o pagină cu datele lor personale, adresa de livrare, dar și cu parola de la cont, pe care le pot modifica, dar au și secțiuni care conțin comenzile pe care le-au efectuat anterior, produsele pe care le-au cumpărat, dar și produsele care se află în lista lor de favorite.

În dezvoltarea platformei se folosesc HTML, CSS, Bootstrap și Vue.js pe partea de frontend, pentru a crea interfața grafică a platformei, care conține pagini dinamice, care răspund acțiunilor întreprinse de utilizatori. Pentru partea de backend se folosește Laravel pentru a realiza logica din spatele acesteia: cu ajutorul migrărilor se creează schema bazei de date, cu ajutorul Eloquent ORM se creează modele pentru fiecare tabel din baza de date și se stabilesc relațiile dintre ele, în controllere se creează rute, care fac legătura între backend și frontend și care gestionează cererile utilizatorilor, se folosesc funcționalitățile pe care le are încorporate de autentificare, resetare a parolei și trimitere a e-mailurilor, se creează funcții care gestionează logica interacțiunilor utilizatorului cu platforma. MySQL se folosește pentru a crea schema bazei de date cu tabelele aferente și relațiile dintre acestea, dar și pentru a crea interogări cu ajutorul cărora se manipulează datele din interiorul bazei de date. REST API se folosește pentru a lega interfața grafică a aplicației de informațiile din baza de date, cu ajutorul protocolului HTTP. Apache HTTP Server se folosește pentru a putea rula platforma web pe un server local.

Lucrarea este împărțită în trei capitole, după cum urmează:

Primul capitol prezintă descrierea temei de licență din mai multe puncte de vedere. În primul subcapitol este descris domeniul din care face parte tema de licență, mai exact ce implică fiecare ramură de dezvoltare web. În al doilea subcapitol se descriu clasele de utilizatori și cazurile de utilizare ale platformei. În al treilea subcapitol se prezintă trei aplicații similare cu cea care urmează a fi implementată și o comparație între cele trei din punct de vedere al funcționalităților, iar în cel de-al patrulea subcapitol se prezintă funcționalitățile ce vor fi prezente și în aplicația propusă.

Al doilea capitol prezintă stadiul actual în domeniu și selectarea soluției tehnice, în primul subcapitol fiind descrise tehnologiile utilizate pentru dezvoltarea platformei, iar în cel de-al doilea subcapitol se detaliază scopul folosirii fiecărei tehnologii și mediu de dezvoltare ales în realizarea aplicației, dar și avantajele și dezavantajele acestora.

Al treilea capitol prezintă considerentele legate de implementarea soluției tehnice.

1. Descrierea problemei abordate

1.1. Descrierea domeniului din care face parte tema de licență

Dezvoltarea web implică activități precum proiectarea, realizarea și întreținerea site-urilor și aplicațiilor web care pot fi accesate prin intermediul unui browser. Scrierea efectivă a codului pentru realizarea funcționalităților și design-ului site-ului web, punând în practică schițele pe care un client le dorește sau o echipă de proiectare le-a realizat, sunt responsabilitățile pe care un dezvoltator web le are. Acesta trebuie să stăpânească o multitudine de limbaje de programare cuprinzând HTML, CSS, JavaScript, PHP, dar și framework-uri.^[1]

Dezvoltarea frontend, dezvoltarea backend și dezvoltarea full-stack sunt cele trei ramuri care fac parte din dezvoltarea web.^[1]

Dezvoltarea frontend se ocupă de partea vizuală a site-ului web sau a aplicației, cea pe care utilizatorul o vede și interacționează direct cu ea. Aceasta implică transformarea ideilor echipei de proiectare sau ale clientului, cu privire la aspectul produsului, în linii de cod, realizând, totodată, site-uri web care se pot mula pe orice dispozitiv, fără a fi afectată experiența utilizatorului. Aceștia trebuie să dețină vaste cunoștințe în ceea ce privește limbaje de programare precum HTML, CSS și JavaScript, dar și framework-uri, printre care se pot enumera React, Bootstrap, Vue.js, AngularJS.^[1]

Dezvoltarea frontend se ocupă de două elemente, acestea fiind înfățișarea și raționamentul. Înfațișarea se referă la modul cum arată interfața grafică a site-ului pe care utilizatorul o vede, mai exact prezentarea și aranjarea componentelor în pagină, dar și modul în care utilizatorul poate interacționa cu ele. Pe de altă parte, logica include colectarea și transformarea datelor care vin prin rute, accesate de utilizatori, și transpunerea acestora în interfața grafică a site-ului, confirmarea datelor de intrare, precum și alte sarcini aferente.^[2]

Dezvoltarea backend presupune partea care se concentrează asupra operațiunilor sistemului. Interacțiunile cu baza de date, autentificarea utilizatorilor, setările serverului, precum și logica de business sunt fundația site-ului web pe care dezvoltatorii backend trebuie să o realizeze și să se asigure că funcționează cum ar trebui.^[1]

Backend-ul dispune de mai multe servicii web, însă REST API este cel mai utilizat în zilele noastre, întrucât protocolul HTTP este, de obicei, singurul de care depinde. REST API oferă un set de operații (GET, PUT, POST, DELETE) pe care aplicațiile frontend le pot folosi.^[3] Aceste servicii funcționează precum un liant între frontend și backend, prin care aplicațiile frontend pot interacționa ușor cu backend-ul fără să fie nevoie să se aibă cunoștințe despre operațiunile care se execută pe server.^[4]

Rolul persoanelor care lucrează pe backend este de a permite frontend-ului să interacționeze cu datele din baza de date ale unei aplicații prin operațiile de preluare, stocare, modificare și ștergere ale acestora. De asemenea, dezvoltatorii se confruntă cu problemele legate de performanță, pe care trebuie să le rezolve, atunci când dezvoltă operații de backend, deoarece ei trebuie să administreze un volum mare de cereri simultane. Atunci când se proiectează punctele de acces în backend-ul unui site sau aplicații web, trebuie avută în vedere optimizarea proiectului, pentru ca acesta să se poată adapta la schimbări survenite în viitor sau la integrarea sa cu alte servicii externe sau baze de date. Prin intermediul acestor puncte de acces, frontend-ul poate trimite cereri venite din partea utilizatorilor

către server și primi răspunsuri din partea acestuia care conțin informațiile cerute. Legătura dintre frontend și backend se realizează prin intermediul API, în care dezvoltatorii specifică metodele protocolului HTTP, și anume GET, POST, PUT, DELETE, și ruta pe care trebuie să ajungă cererile.^[5]

Partea de backend trebuie să asigure acces securizat către resursele ei prin mecanisme adecvate de autentificare și autorizare, acest lucru implicând definirea unor rute private sau publice. Rutele publice sunt accesibile oricui, fiind utilizate pentru a oferi informații generale, cum ar fi lista de produse a site-ului web, specificațiile produselor, recenziiile lăsate acestora de către utilizatori. Rutele private conțin funcționalități private, cum ar fi gestionarea datelor personale din contul de utilizator sau gestionarea conținutului pe care îl postează în cadrul site-ului din contul său. Totodată, se pot introduce măsuri de revocare a accesului, după caz. De asemenea, atunci când aplicația oferă servicii de cumpărături online sau acces la servicii pe bază de abonament, backend-ul joacă un rol important în manevrarea plăților. Acest lucru implică munca cu procesatori de plăți online, cum este PayPal, sau cu magazine online, precum Google sau Apple pentru a verifica și a procesa plățile. Acest lucru implică ca backend-ul aplicației să colaboreze cu backend-ul acestor furnizori de servicii de plată sau platforme de plată, pentru a le oferi utilizatorilor experiențe fără întreruperi.^[5]

Dezvoltatorii backend lucrează, în general, cu limbaje de programare pe partea de server, și anume PHP, Python și Ruby, precum și cu sisteme de gestionare a bazelor de date.^[1]

Un dezvoltator full-stack este o persoană care posedă atât abilități de dezvoltare frontend, cât și de dezvoltare backend, care le permite să lucreze pe partea de server, dar și pe partea de client a unei aplicații. Aceștia au vaste cunoștințe, în ceea ce privește abilitățile de codare, care le dau posibilitatea să contribuie la diverse aspecte ale dezvoltării software. Ca un dezvoltator full-stack, câteva responsabilități presupun scrierea unui cod ușor de urmărit atât pentru backend, cât și pentru frontend, proiectarea interfeței grafice cu care utilizatorul interacționează, garantând compatibilitatea acestuia pe mai multe dispozitive, dezvoltarea de servere și baze de date, colaborarea cu echipa de proiectare a design-ului, dezvoltarea de REST API pentru a lega interfața grafică de server și implicit de baza de date, testarea și depanarea codului pentru a menține codul optimizat. Totodată, ei trebuie să aibă în vedere factori precum securitatea, scalabilitatea și întreținerea sistemului.^[6]

1.2. Formularea problemei

1. Clasele de utilizatori:

- a) Cumpărători obișnuiți: Aceștia fac parte din categoria persoanelor care vizitează platforma web pentru a căuta cărți și a vizualiza specificațiile acestora împreună cu recenziiile lăsate de alte persoane, pentru a plasa comenzi, dar și pentru a pune întrebări sau a trimite sugestii prin completarea formularului de contact;
- b) Cumpărători înregistrați: Utilizatorii care au un cont pe platformă beneficiază de anumite avantaje față de cei fără, cum ar fi gestionarea datelor din cont, vizualizarea istoricului de comenzi plasate și a produselor cumpărate, posibilitatea de a adăuga recenzii cărților, dar au și o listă de favorite în care pot salva sau șterge produse;
- c) Administrator: Față de cele două clase prezentate anterior, acesta poate, în plus, gestiona informațiile din baza de date legate de clienți, cărți, review-uri, donații, categoriile cărților și comenzi.

2. Cazurile de utilizare:

a) Înregistrarea utilizatorilor:

- Utilizatorii își pot crea conturi noi pe platformă prin completarea unui formular cu datelor lor personale;
- Procesul de înregistrare este finalizat cu verificarea adresei de e-mail a utilizatorului prin accesarea link-ului din mesajul primit pe e-mail.

b) Autentificarea utilizatorilor:

- Utilizatorii deja înregistrați se pot autentifica în conturile lor prin completarea e-mailul și a parolei;
- Autentificarea este necesară pentru a beneficia de funcționalități precum gestionarea datelor din cont, vizualizarea istoricului de comenzi plasate și a produselor cumpărate, posibilitatea de a adăuga recenzii cărților, dar au și o listă de favorite în care pot salva sau șterge produse.

c) Resetarea parolei: Utilizatorii își pot reseta parola de la cont în cazul în care o uită sau o pierd;

d) Căutarea produselor: Utilizatorii pot căuta produse pe platformă prin intermediul unui sistem de căutare, în funcție de titlu, autor sau ISBN;

e) Filtrarea și sortarea produselor:

- Utilizatorii pot aplica filtre în funcție de autor, preț, editură, categorie și limbă pentru a filtra rezultatele în funcție de preferințele fiecăruia;
- Utilizatorii pot sorta cărțile, ordonându-le crescător sau descrescător în funcție de preț sau nume.

f) Secțiune de recenzii: Pe fiecare pagină a unei cărți va exista o secțiune cu recenziile cărții respective, pe care utilizatorii le pot citi sau, în cazul în care au cont, pot adăuga o recenzie nouă, specificând titlul, comentariul și nota acesteia;

g) Listă cu produse favorite: Utilizatorii dispun de o listă de produse favorite în care pot adăuga cărți pentru a le putea accesa mai ușor sau le pot șterge atunci când nu mai prezintă interes pentru ei;

h) Gestionarea coșului de cumpărături: Utilizatorii pot adăuga cărți în coșul de cumpărături în care pot ajusta cantitatea sau șterge produsele înainte de finalizarea comenzii;

i) Plasarea și confirmarea comenzii:

- Utilizatorii pot plasa comenzi prin completarea unui formular în care trebuie să specifice datele sale de contact, adresa de livrare, metoda de plată și cea de livrare;
- După finalizarea comenzii, utilizatorii vor primi pe e-mail un mesaj de confirmare a comenzii.

j) Formular de donare: Utilizatorii pot dona oricâte cărți doresc prin completarea unui formular, în care trebuie să specifice numele, prenumele, e-mailul și numărul lor de telefon, titlul, autorul și ISBN-ul cărților pe care doresc să le doneze, dar și adresa de la care vor fi ridicate cărțile;

k) Returnarea cărților: Utilizatorii pot returna cărțile achiziționate care nu corespund așteptărilor lor, intrând pe profilul lor, în secțiunea de comenzi și selectând cărțile respective;

- l) Secțiuni cu promoții și noutăți: Pe pagina principală a platformei o secțiune cu cărțile care dispun de reduceri, dar și o secțiune cu noutăți în care se găsesc ultimele cărți adăugate spre vânzare;
- m) Recomandări personalizate: Utilizatorii primesc recomandări de cărți, în funcție de autor și categorie, bazate pe paginile cărților pe care le accesează;
- n) Secțiune de contact: La finalul paginii principale se găsește un formular de contact, pe care utilizatorii îl pot completa, specificând e-mailul și un mesaj, în cazul în care au întrebări sau sugestii cu privire la îmbunătățirea experienței în cadrul platformei ;
- o) Gestionarea profilului utilizatorului:
 - Utilizatorii dispun de o pagină cu datele lor personale, adresa de livrare, dar și cu parola de la cont, pe care le pot schimba oricând doresc;
 - Utilizatorii au, de asemenea, și secțiuni care conțin comenzile pe care le-au efectuat în trecut, produsele pe care le-au cumpărat, dar și produsele care se află în lista lor de favorite.
- p) Administrarea datelor din baza de date de către administrator: gestionarea informațiilor despre clienți, cărți, review-uri, donații, categoriile cărților și comenzi.

Conform celor descrise anterior, există două cazuri de navigare și utilizare în cadrul platformei, care vor fi exemplificate în continuare.

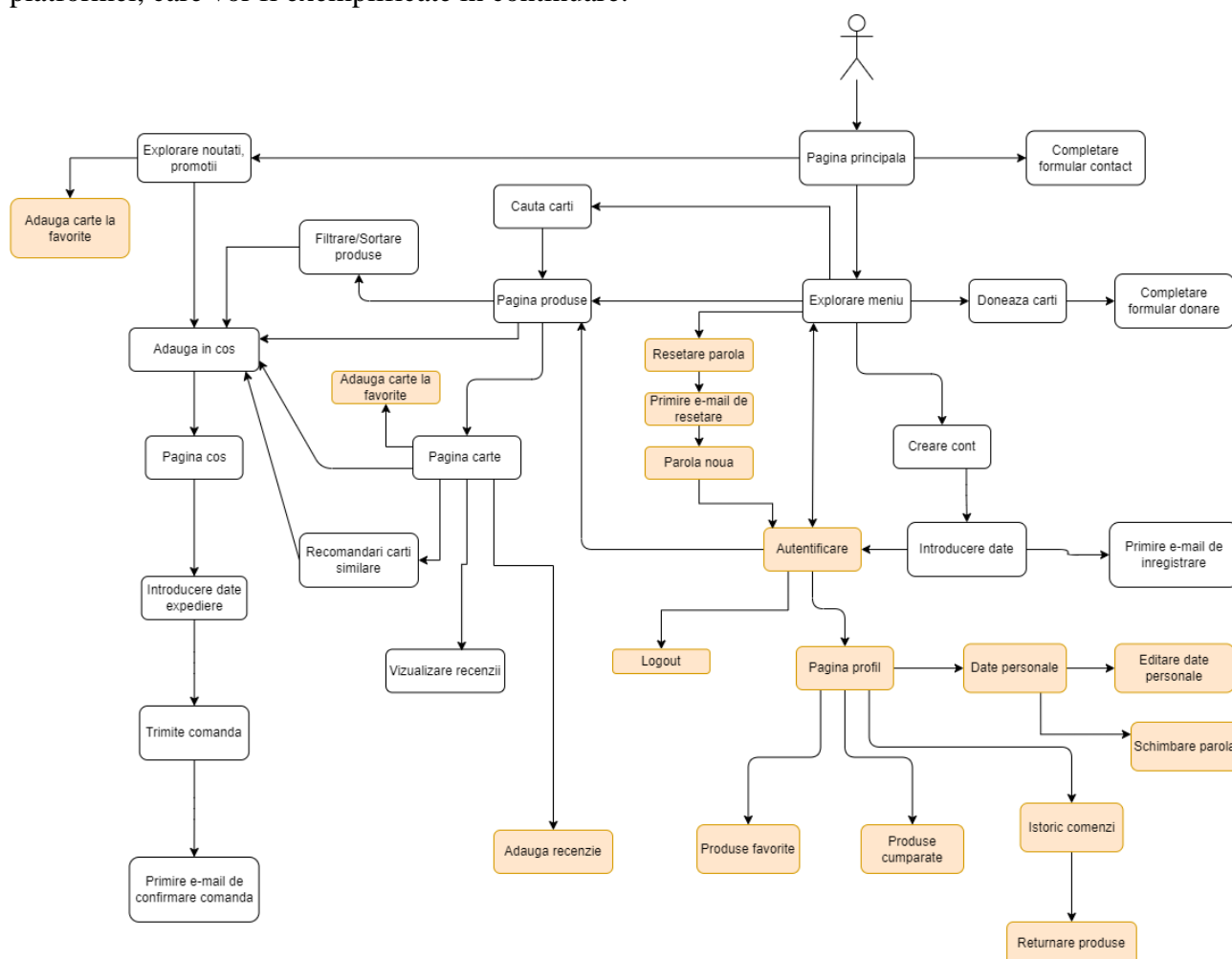


Fig. 1.1 – Organigrama fluxului de activități în cazul utilizatorilor

În Fig.1.1 se poate observa că utilizatorii care au cont în cadrul platformei beneficiază de funcționalitățile suplimentare, aceștia fiind reprezentați cu culoarea portocaliu.

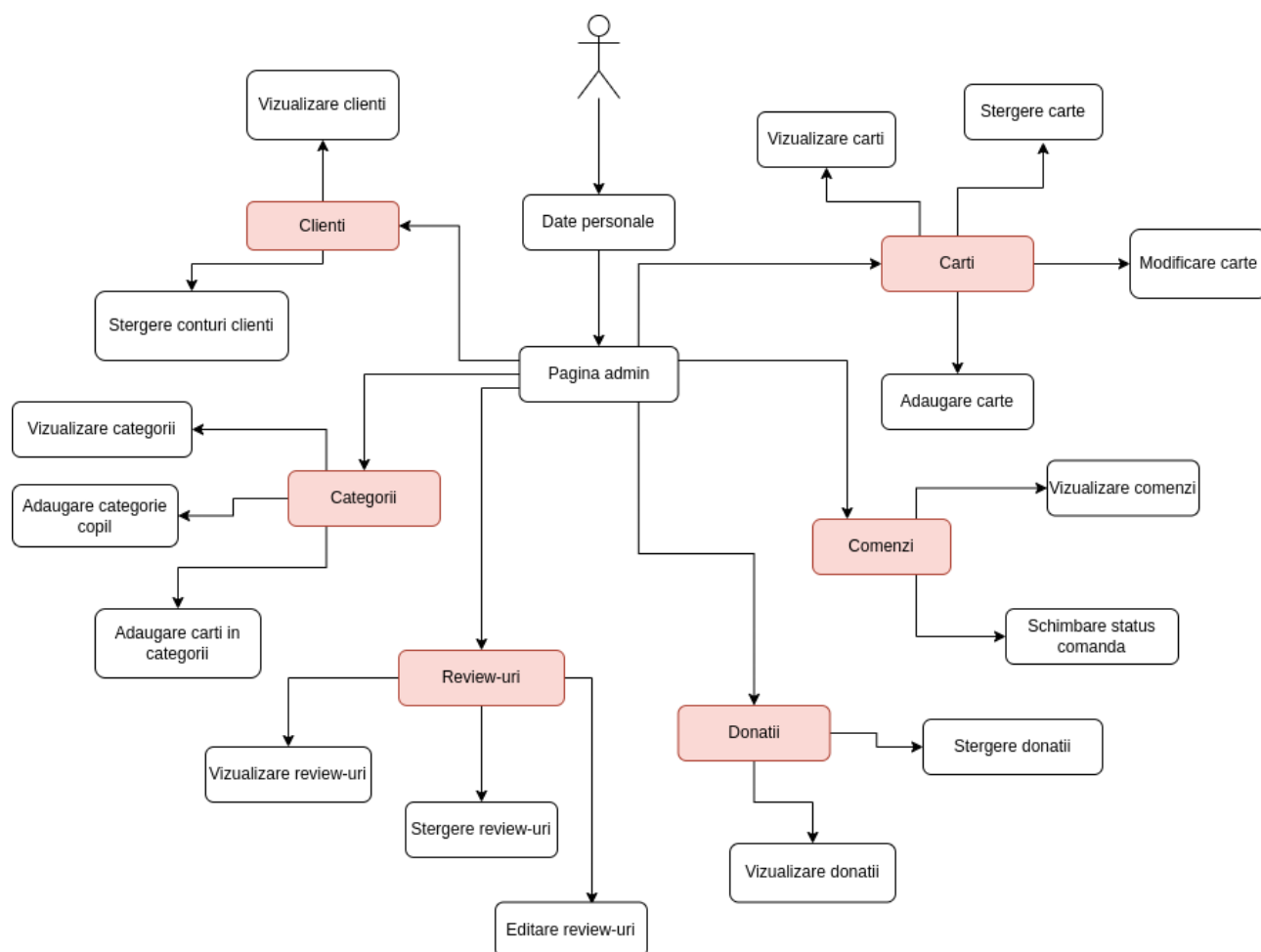


Fig. 1.2 – Organigrama fluxului de activități în cazul administratorului

Administratorul are acces asupra platformei ca un utilizator cu cont, și în plus, beneficiază de o pagină suplimentară, care poate fi accesată din secțiunea de date personale, de pe pagina de profil, în care poate gestiona date despre clienți, cărți, comenzi, categoriile cărților, review-uri și donații.

1.3. Studiu asupra realizărilor similare din domeniu

1.3.1. Funcționalități oferite de site-ul librăriei Libris^[7]

Utilizatorul își poate crea un cont nou pe site, iar când face acest lucru, primește pe e-mailul folosit pentru crearea contului un mesaj în care este înștiințat să acceseze link-ul din mesaj pentru a completa înregistrarea pe site. Utilizatorul se poate, după aceea, autentifica pe site. În cazul în care utilizatorul și-a uitat parola, acesta își poate introduce e-mailul pentru a-și confirma identitatea, putând apoi să-și aleagă noua parolă. În cadrul site-ului există o bară de căutare, în care utilizatorul poate căuta cărți în funcție de titlul, autorul, ISBN-ul, editura sau categoria acestora. Totodată există o bară laterală cu filtre, unde utilizatorul își poate filtra produsele în funcție de mai multe categorii.

Există, de asemenea, și posibilitatea ordonării cărților în funcție de mai multe criterii. Utilizatorul are posibilitatea de a se abona la newsletter pentru a primi pe e-mail notificări în legătură cu noutățile și reducerile de pe site. Utilizatorii dispun de o listă de dorințe în care pot adăuga sau șterge produse. Site-ul oferă recomandări personalizate în funcție de căutările anterioare, noutăți, promoții, istoricul de navigare al utilizatorilor, cele mai căutate produse, dar și cele mai comandate. Utilizatorii au posibilitatea de a citi primele pagini ale cărților. Utilizatorii pot lăsa review-uri produselor, în care să includă o recenzie scrisă și o notă. De asemenea utilizatorii pot aprecia sau lăsa un comentariu la un review al altui client. Clienții pot adăuga cărți sau alte produse în coșul de cumpărături în care pot ajusta cantitatea sau șterge produse înainte de finalizarea comenzii. După efectuarea unei comenzi, aceștia primesc un mesaj, pe e-mail, de confirmare a comenzii. Există mai multe opțiuni de plată și anume: plata cu cardul, plata în rate, plata ramburs, dar și mai multe opțiuni de livrare cum sunt livrarea prin curier sau ridicare personală. De asemenea, clienții pot folosi coduri de reducere sau puncte de fidelitate înainte de plasarea unei comenzi. Site-ul oferă un program de fidelitate în care utilizatorii pot acumula puncte în funcție de achizițiile pe care le efectuează. Utilizatorii au o pagină cu datele lor unde își pot edita datele personale, pot adăuga noi adrese de livrare, pot vedea ce comenzi au efectuat, ce review-uri au dat produselor, punctele lor de fidelitate și unde se pot deconecta de la cont.

1.3.2. Funcționalități oferite de site-ul librăriei CLB^[8]

Utilizatorul își poate crea un cont nou pe site, iar când face acest lucru, primește pe e-mailul folosit pentru crearea contului un mesaj în care este înștiințat să acceseze link-ul din mesaj pentru a completa înregistrarea pe site. Utilizatorul se poate, după aceea, autentifica pe site. În cazul în care utilizatorul și-a uitat parola, acesta își poate introduce e-mailul pentru a-și confirma identitatea, putând apoi să-și aleagă noua parolă. În cadrul site-ului există o bară de căutare, în care utilizatorul poate căuta cărți în funcție de titlul, autorul sau ISBN-ul acestora. Totodată există o bară laterală cu filtre, unde utilizatorul își poate filtra produsele în funcție de mai multe categorii. Există, de asemenea, și posibilitatea ordonării cărților în funcție de mai multe criterii, posibilitatea selectării numărului de produse pe o pagină, dar și a modului sub care pot apărea, sub formă de grilă sau listă. Utilizatorul are posibilitatea de a se abona la newsletter pentru a primi pe e-mail notificări în legătură cu noutățile și reducerile de pe site. Utilizatorii dispun de o listă de favorite în care pot adăuga sau șterge produse. Site-ul oferă recomandări în funcție de căutările anterioare, noutăți, promoții, istoricul de navigare al utilizatorilor, dar și cele mai vândute cărți. De asemenea, oferă și o secțiune de blog unde sunt postate articole, pe care utilizatorii le pot citi. Utilizatorii pot completa un formular de contact în care pot trimite comentarii și sugestii pentru îmbunătățirea experienței pe site. Aceștia au posibilitatea de a compara mai multe produse între ele în funcție de anumite categorii. Utilizatorii pot lăsa review-uri produselor, în care să includă o recenzie scrisă și o notă. Clienții pot adăuga cărți sau alte produse în coșul de cumpărături în care pot ajusta cantitatea sau șterge produse înainte de finalizarea comenzii. După efectuarea unei comenzi, aceștia primesc un mesaj pe e-mail de confirmare a comenzii. Există mai multe opțiuni de plată și anume plata cu cardul sau plata ramburs, dar și mai multe opțiuni de livrare cum sunt livrarea prin curier sau ridicare personală. De asemenea, clienții pot folosi coduri de reducere înainte de plasarea comenzilor. Utilizatorii au o pagină cu datele lor unde își pot edita datele

personale, își pot schimba parola contului, pot schimba adresa de livrare, pot vedea ce comenzi au efectuat și unde se pot deconecta de la cont.

1.3.3. Funcționalități oferite de site-ul librăriei Mihai Eminescu^[9]

Utilizatorul își poate crea un cont nou pe site, iar după aceea se poate autentifica pe el. În cazul în care utilizatorul și-a uitat parola, acesta își poate introduce e-mailul pentru a-și confirma identitatea, putând apoi să-și aleagă noua parolă. În cadrul site-ului există o bară de căutare, în care utilizatorul poate căuta cărți în funcție de titlul, autorul sau ISBN-ul acestora. În cadrul site-ului există posibilitatea ordonării cărților în funcție de mai multe criterii. Utilizatorul are posibilitatea de a se abona la newsletter pentru a primi pe e-mail notificări în legătură cu noutățile și reducerile de pe site. Site-ul oferă recomandări în funcție de căutările anterioare, noutăți, dar și cele mai vândute cărți. De asemenea, oferă și o secțiune de evenimente unde sunt postate lansări de carte, conferințe, recitaluri de muzică și poezie. Utilizatorii pot lăsa review-uri produselor, în care să includă o recenzie scrisă și o notă. Clienții pot adăuga cărți sau alte produse în coșul de cumpărături în care pot ajusta cantitatea sau șterge produse înainte de finalizarea comenzii. După efectuarea unei comenzi, aceștia primesc un mesaj pe e-mail de confirmare a comenzii. Există mai multe opțiuni de plată și anume plata cu cardul sau plata ramburs, dar și mai multe opțiuni de livrare cum sunt livrarea prin curier, ridicare personală, livrare prin poștă sau rezervare în magazin. De asemenea clienții pot folosi coduri promoționale înainte de plasarea comenzii. Site-ul oferă un program de fidelitate în care utilizatorii pot acumula puncte în funcție de achizițiile pe care le efectuează. Utilizatorii au o pagină cu datele lor unde își pot edita datele personale, își pot schimba parola contului, pot adăuga noi adrese de livrare, pot vedea ce comenzi au efectuat, pot vedea ce cupoane și ce puncte de loialitate au și unde se pot deconecta de la cont.

1.3.4. Comparație între cele trei realizări

Toate cele trei site-uri le permit utilizatorilor să-și creeze conturi și să se autentifice pe acestea, dar și să-și recupereze parola, cu ajutorul adresei de e-mail, în cazul în care este uitată. Se pot căuta produse în funcție de mai multe criterii, cum sunt titlu, autor, ISBN, iar în cazul site-ului librăriei Libris, și în funcție de editura sau categoria cărților. Toate site-urile pun la dispoziție utilizatorilor posibilitatea de a se abona la newsletter pentru a primi notificări pe e-mail în legătură cu noutățile și reducerile de pe site. Site-urile librăriilor CLB și Libris pun la dispoziție filtre pentru a face o selecție a produselor în funcție de mai multe categorii, o listă de favorite unde se pot salva sau șterge produse, dar beneficiază și de un sistem prin care se trimit, pe e-mailul noilor clienți, mesaje cu link-ul, prin care își pot definitiva înregistrarea pe site. Toate site-urile beneficiază de posibilitatea ordonării produselor în funcție de mai multe criterii, iar pe cel al librăriei CLB poți alege numărul de cărți afișate pe o pagină, dar și modul de afișare, sub formă de grilă sau listă. Pe site-ul celor de la Libris, utilizatorii au posibilitatea de a citi primele pagini ale cărților, dar au și o pagină unde pot vedea toate review-urile pe care le-au lăsat produselor. În cadrul tuturor site-urilor, utilizatorii pot lăsa o recenzie scrisă și o notă produselor. Clienții pot adăuga cărți sau alte produse în coșul de cumpărături în care pot ajusta cantitatea sau pot șterge produse din coș înainte de finalizarea comenzii și, de asemenea, pot adăuga coduri promoționale înainte de a da comanda. Există mai multe opțiuni de plată și anume

plata cu cardul, ramburs la curier, plata în rate (librăria Libris), dar și mai multe metode de livrare, cum sunt livrare prin curier, ridicare personală, livrare prin poștă (Librăria Mihai Eminescu) sau rezervare în magazin (Librăria Mihai Eminescu). După efectuarea unei comenzi, clienții primesc un mesaj de confirmare a comenzii pe e-mail. Site-urile librăriilor Mihai Eminescu și Libris oferă un program de fidelitate, în care utilizatorii pot acumula puncte în funcție de achizițiile efectuate. Toate site-urile oferă recomandări în funcție de căutările anterioare, noutăți și cele mai vândute cărți, iar cele de la CLB și Libris oferă, în plus, secțiune de promoții și istoricul de navigare al utilizatorilor, iar Libris oferă și o secțiune cu cele mai căutate cărți. Site-ul Librăriei Mihai Eminescu, spre deosebire de celelalte, oferă o secțiune de evenimente unde sunt postate lansări de carte, conferințe, recitaluri de muzică și poezie. Site-ul librăriei CLB, spre deosebire de celelalte, oferă utilizatorilor posibilitatea de a completa un formular de contact în care pot trimite comentarii și sugestii pentru îmbunătățirea experienței pe site, posibilitatea de a compara mai multe produse între ele în funcție de anumite categorii, dar și o secțiune de blog unde sunt postate articole. În cazul tuturor site-urilor există o pagină cu datele utilizatorilor, unde își pot edita datele personale, pot vedea comenzile efectuate, pot schimba/adăuga adrese de livrare și se pot deconecta de la cont, iar pe site-urile celor de la Libris și Librăria Mihai Eminescu pot vedea ce puncte de fidelitate au acumulat. Pe site-urile celor de la CLB și Librăria Mihai Eminescu, utilizatorii își pot schimba parola, iar pe cel din urmă pot vedea și de ce cupoane dispun.

1.4. Stabilirea cerințelor funcționale și nefuncționale ale aplicației

Funcționalitățile ce vor fi implementate și în site-ul ce urmează a fi dezvoltat:

- sistem de înregistrare cu verificare de e-mail – utilizatorul își va completa datele personale într-un formular pentru a-și crea cont în cadrul site-ului, iar pentru a finaliza tot procesul de înregistrare, va primi pe e-mail un link pe care va trebui să-l acceseze pentru a-și confirma identitatea;
- sistem de autentificare – utilizatorul având deja cont pe platforma, se poate autentifica pe acesta completând un formular în care își va scrie e-mailul și parola;
- sistem de resetare a parolei – utilizatorii își pot schimba parola de la cont în cazul în care este uitată, introducând e-mailul pentru a-și confirma identitatea, putând apoi să își aleagă noua parolă;
- sistem de căutare – utilizatorii au la dispoziție o bară de căutare în care pot căuta cărți în funcție de titlul, autorul și ISBN-ul acestora;
- catalog de produse – în cadrul platformei, va exista o pagină unde se vor afla toate cărțile care pot fi cumpărate;
- opțiuni de filtrare și sortare a cărților – utilizatorii au posibilitatea să-și filtreze cărțile în funcție de autor, preț, editură, categorie și limbă, dar și să le ordoneze crescător sau descrescător în funcție de preț sau nume;
- secțiune de recenzii pentru cărți – în cadrul platformei, va exista pe pagina fiecărei cărți o secțiune cu recenziiile cărții respective, lăsate de către utilizatorii, aceștia putând lăsa un comentariu însoțit de titlul acestuia și o notă de la 1 la 5;

- gestionarea coșului de cumpărături – utilizatorii pot adăuga cărți în coșul de cumpărături în care pot ajusta cantitatea sau șterge produsele înainte de finalizarea comenzii;
- proces de înregistrare a comenzii – după ce utilizatorul și-a adăugat în coș toate produsele pe care dorește să le comande, acesta va completa un formular în care își va introduce datele sale de contact, adresa de livrare, dar și metoda de plată și cea de livrare;
- listă cu produsele favorite – utilizatorii dispun de o listă de favorite în care pot adăuga sau șterge cărți care prezintă interes pentru ei;
- formular de donare – utilizatorii pot dona oricâte cărți doresc prin completarea unui formular, în care trebuie să specifice numele, prenumele, e-mailul și numărul lor de telefon, titlul, autorul și ISBN-ul cărților pe care vor să le doneze, dar și adresa de la care vor fi ridicate cărțile;
- confirmare de comandă prin e-mail – atunci când utilizatorul va finaliza o comandă, acesta va primi pe e-mail un mesaj de confirmare a comenzii;
- posibilitatea returnării cărților – utilizatorii au posibilitatea de a returna cărțile achiziționate care nu corespund așteptărilor lor, intrând pe profilul lor, în secțiunea de comenzi și selectând cărțile respective;
- secțiune cu promoții – pe pagina principală a platformei există o secțiune cu cărțile care dispun de reduceri;
- secțiune cu noutăți – pe pagina principală a platformei există o secțiune cu ultimele cărți adăugate spre vânzare;
- recomandări bazate pe căutările utilizatorului – utilizatorii primesc recomandări de cărți, în funcție de autor și categorie, bazate pe paginile cărților pe care le accesează;
- secțiune de contact – utilizatorii găsesc, la finalul paginii principale, un formular pe care îl pot completa cu e-mailul și un mesaj care poate conține întrebări, în cazul în care au nelămuriri, sau pot lăsa sugestii pentru îmbunătățirea experienței în cadrul platformei;
- gestionarea profilului utilizatorului – clienții dispun de o pagină cu datele lor personale, adresa de livrare, dar și cu parola de la cont, pe care le pot schimba; de asemenea, au o secțiune cu comenzile pe care le-au efectuat în trecut, cu produsele pe care le-au cumpărat, dar și cu produsele care se află în lista lor de favorite.

2. Stadiul actual în domeniu și selectarea soluției tehnice

2.1. Stadiul actual al tehnologiilor utilizate pentru dezvoltarea soluției

2.1.1. HTML

HTML este un limbaj folosit pentru a realiza structura și conținutul unei pagini web adică de a diviza textele și imaginile de pe aceasta pentru a ne spune care este titlul, subtitlul, din ce este format un paragraf, unde se găsește o listă, un tabel, care elemente sunt imagini și așa mai departe.^[10] De asemenea, cu ajutorul său paginile web pot avea în structura lor conținut video, audio, link-uri către alte pagini sau site-uri web. În aceeași ordine de idei, poate fi folosit pentru a crea formulare, pentru a efectua rezervări sau pentru a căuta informații.^[11] Acest limbaj nu prezintă operații logice sau comenzi care pot fi executate și nici nu este tradus de către un compilator, ci de către un browser web.^[10]

HTML folosește tag-uri pentru a alcătui structura paginii web. Aceste tag-uri sunt cuprinse între paranteze unghiulare “< >” și vin doar în pereche, una la începutul structurii ce urmează a fi scrisă și una la sfârșitul acesteia.^[12]

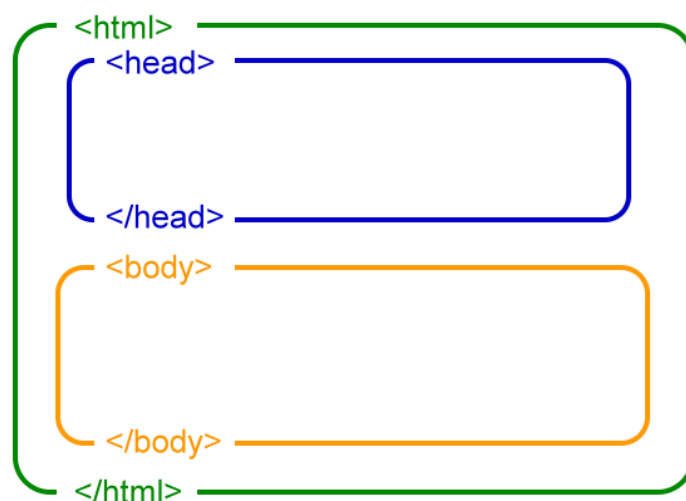


Fig. 2.1 – Structura unui document HTML^[13]

Notăția <html> se află la începutul oricărui document HTML, iar notația </html> la sfârșitul acestuia. Între cele două notații există două secțiuni, și anume antetul, <head>, și corpul documentului, <body>.^[13] Antetul înglobează informațiile care nu apar efectiv pe pagina web, precum elementele: <style> - ne îngăduie să stilizăm paginile web, <title> - sugerează titlul paginii web, care apare în bara de titlu a browser-ului, <base> - menționează adresa URL de bază, <noscript> - asigură inserarea unei alternative de conținut în cazul în care limbajul JavaScript a fost dezactivat, <script> - permite utilizarea limbajului JavaScript, <meta> - oferă informații despre conținutul unei paginii web, precum descrierea, cuvintele cheie, autorul sau alte detalii ale acesteia, <link> - leagă la documentul HTML alte resurse externe.^[14] Corpul documentului cuprinde informațiile paginii web care sunt vizibile utilizatorului și este în general alcătuit din antet, care conține titlul și logo-ul site-

ului, dar și bara de navigație, conținutul principal al paginii web, barele laterale care pot conține filtre după categorii și subsolul care prezintă date de contact, link-uri către rețele de socializare și drepturi de autor.^[11]

2.1.2. CSS

CSS este un limbaj folosit pentru a stiliza paginile web, modul în care conținutul acestora sunt prezentate către utilizator, prin definirea design-ului, aspectului și modificărilor care intervin atunci când site-ul este deschis pe diferite dispozitive. CSS oferă ajutor pentru a defini culori, font-uri, alineate și alte aspecte importante în definirea paginilor online. Acesta este esențial, pentru că ușurează navigarea clientului, dar oferă și informații clare, ușor de urmărit.^[15]

Acesta permite stilizarea unui document scris într-un limbaj de marcare, cum este HTML, îmbunătățind felul în care informațiile sunt oferite publicului, dar este util, pentru că reduce efortul depus prin faptul că secvențele de cod pot fi repetate în diferite pagini HTML, deoarece dezvoltatorii web au posibilitatea de a salva într-un fișier separat codul aferent de CSS, astfel se poate modifica vizual pagina, fără a umbla la structura sa.^{[15][16]} De asemenea, CSS poate utiliza o ierarhie a stilurilor, în care un element copil poate moșteni proprietăți de la un element părinte.

Prin utilizarea de CSS se pot realiza mai multe aspecte legate de partea vizuală a unei pagini, printre care se pot enumera: definirea culorii fundalului paginii sau a textului prezent pe aceasta, aranjarea tabelelor, butoanelor sau a imaginilor prin posibilitatea ajustării poziției, bordurii, culorii, adăugarea de animații elementelor prezente, definirea alineatelor și distanțelor între elemente, precizarea font-ului, dimensiunii și stilului unui text și poziționarea elementelor în pagină.^[15]

În limbajul CSS, cei mai însemnați termeni sunt selectorii, proprietățile și valorile. Selectorii determină la ce componentă din codul HTML se face referire pentru a fi stilizată ulterior, fiind urmați de acolade, în interiorul cărora se află proprietățile, care se referă la ce aspect urmează să se facă referire (mărime, culoare, font etc.). Aceste proprietăți urmează să fie definite cu ajutorul valorilor, de exemplu, cu privire la culoare se poate atribui valoarea “roșu”.^[15]

Există trei metode de a scrie CSS într-un fișier HTML:

- Inline CSS – se definește stilul direct în elementul HTML dorit, utilizând atributul “style”;^{[16][17]}
- Internal CSS – se definește în antetul documentului HTML, <head>, și este utilizat cu ajutorul etichetei <style> pentru a stiliza întregul fișier;^{[16][17]}
- External CSS – se definește ca un fișier separat de cel HTML și poate fi folosit la mai multe fișiere HTML, utilizând eticheta <link>.^{[16][17]}

2.1.3. Bootstrap

Bootstrap este un framework de CSS folosit, mai ales, pentru a crea site-uri și aplicații web care se pot adapta, din punct de vedere al interfeței grafice, pe orice dispozitiv ar folosit utilizatorul. Dispune de componente predefinite care pot fi folosite direct în proiect, fiind deja scrise și stilizate, care ulterior se pot adapta cu ajutorul CSS-ului, pentru a se mula pe nevoile fiecăruia, economisindu-se, astfel, timp, deoarece codul nu mai trebuie implementat de la zero.^[18] Bootstrap oferă o gamă

largă de componente printre care se numără bare de navigare, dropdown-uri, formulare, butoane, carduri, dar și containere care sunt folosite pentru împărțirea ecranului în linii și coloane. Sistemul de grilă este compus din două clase de containere: un container de lățime fixă (.container) și un container cu o lățime care se întinde pe toată lățimea ecranului (.container-fluid) care permite site-ului sau aplicației să se ajusteze în funcție de dispozitiv.^[19] Totodată acest framework are o compatibilitate universală, fiind compatibil cu toate versiunile moderne ale browserelor, făcând paginile web să fie afișate corect indiferent de platforma și dispozitivul folosit.^[18]

2.1.4. Vue.js

Vue.js este un framework de JavaScript care este utilizat pentru a crea interfețe cu utilizatorul. Se fundamentează pe HTML, CSS și JavaScript și este utilizat pentru a crea componente care pot fi încorporate ulterior într-o pagină web. Vue.js descrie structura componentelor prin intermediul unui șablon și extinde, astfel, standardul HTML, această proprietate numindu-se declarativitate. De asemenea, o altă caracteristică de care dispune este reactivitatea, care înseamnă că orice schimbare a stării unei componente este detectată imediat și interfețele utilizator sunt actualizate pentru a sugera aceste modificări.^[20]

Vue.js a fost creat de Evan You și lansat în 2014, acesta dorind să păstreze fragmentele care îi stârneau interesul din AngularJS și să creeze un cadru nou mult mai simplificat și mai adaptabil.^[21] Vue.js urmărește arhitectura MVC, care permite dezvoltatorilor să observe interfața proiectului lor, indiferent că vorbim despre o aplicație pentru mobil, desktop sau un site web.^[22]

Vue.js acoperă cele mai multe caracteristici necesare în dezvoltarea frontend, fiind flexibil și adaptabil în funcție de scopul utilizării acestuia. Vue.js poate fi folosit în următoarele situații:^[20]

- înglobarea ca și componente în orice pagină web;^[20]
- aplicații pe o singură pagină;^[20]
- redare Fullstack/Server-Side – aplicațiile pot fi redare și pe server;^[20]
- îmbunătățire HTML static, prin adăugare de interactivitate paginii web fără a fi nevoie de a crea o pagină nouă;^[20]
- poate fi utilizat pentru a crea aplicații destinate desktop-ului sau mobilului;^[20]
- pe prototipuri pentru a demonstra funcționalitatea acestora înainte de a fi făcute sau pentru a adăuga funcționalități aplicațiilor deja existente.^[21]

2.1.5. Laravel

Laravel este un framework de PHP folosit pentru a realiza aplicații web, pe partea de server. Are la bază arhitectura MVC pentru gestionarea datelor și pentru a diviza backend-ul în bucăți logice separate. Acesta conține funcționalități deja programate care ajută dezvoltatorii să creeze aplicațiile mai rapid.^[23]

Laravel a fost creat de Taylor Otwell pentru a înlocui framework-ul CodeIgniter, căruia îi lipseau anumite funcționalități cum ar fi suportul integrat de autentificare și autorizare a utilizatorilor.^[23]

Acest framework este utilizat frecvent în rândul dezvoltatorilor, datorită versatilității și diversității aplicațiilor în care poate fi inclus, cu ajutorul său se pot crea:[23]

- site-uri pentru rețele de socializare;[23]
- site-uri de comerț electronic;[23]
- aplicații formate dintr-o pagină sau pagini multiple;[23]
- site-uri web statice și dinamice;[23]
- aplicații pentru întreprinderi.[23]

Laravel este echipat cu un set de instrumente utile care le permite dezvoltatorilor să economisească timp, scriind mai puține linii de cod, ceea ce duce la reducerea apariției erorilor. În continuare se vor prezenta componentele Laravel-ului:[24]

- Sistem de șabloane Blade – facilitează scrierea și organizarea codului HTML, însă pot fi inserate și componente de Vue.js;
- Eloquent ORM – ușurează interacțiunea cu baza de date, fiecare tabel din aceasta având un model corespunzător prin care se poate interacționa cu tabelul, putându-se realiza operațiile de adăugare, afișare, editare, ștergere;[25]

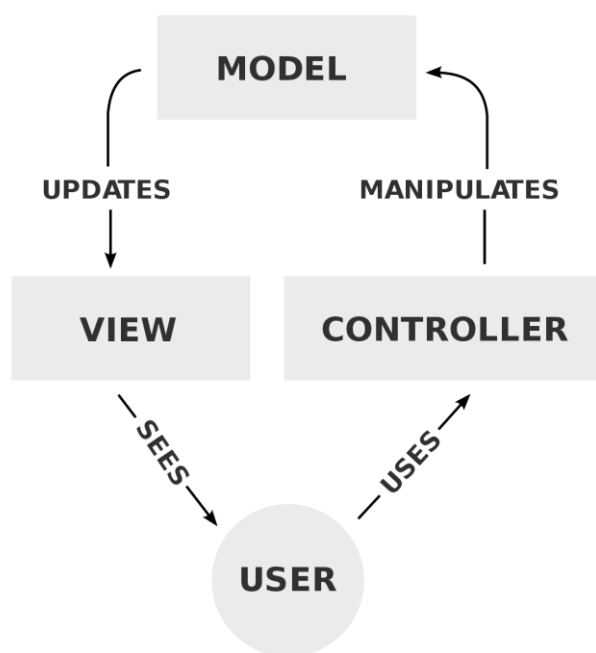


Fig. 2.2 – Arhitectura MVC[26]

- Arhitectură MVC – modelul corespunde unui tabel din baza de date; datele din model se actualizează în view; view-ul permite utilizatorului să vizualizeze datele din model; prin acțiunile utilizatorului, controllerul colectează cererile acestuia și manipulează datele din model care apoi sunt trimise în view;[26]
- Artisan CLI – este folosit pentru a gestiona baza de date, pentru a crea controllere, modele, liste și alte secțiuni ale unei aplicații;[23]
- Redis – poate fi configurat ca un sistem de gestionare a cache-ului, pentru a păstra în memorie datele des utilizate;[23][25]

- Autentificare încorporată – Laravel pune la dispoziție dezvoltatorilor funcționalități de autentificare și autorizare a utilizatorilor deja implementate;^[25]
- Securitate încorporată – folosește algoritmi de hashing deja implementați pentru a securiza parolele utilizatorilor, salvând-o în baza de date sub formă criptată, nu ca text simplu.^[24]

2.1.6. MySQL

MySQL este un sistem de gestiune a bazelor de date relaționale, deținut în prezent de Oracle, care permite datelor să fie stocate și organizate sub formă de tabele. Este open source adică permite oricui să acceseze și să modifice software-ul și este un model client-server, în care clientul este aplicația sau componenta care comunică cu serverul MySQL pentru a accesa sau a manipula datele din baza de date. Serverul MySQL primește comenzi de la clienți pentru a afișa, a insera, a modifica sau a șterge datele din acesta.^{[27][28][29]}

MySQL este una dintre cele mai cunoscute mărci de software de gestiune a bazelor de date, care folosește arhitectura client-server. Pentru a face conexiunea între client și server se utilizează un limbaj caracteristic și anume Structured Query Language. Prin instrucțiunile acestuia se poate învăța server-ul să realizeze anumite operații.^[28]

- interogarea datelor – permite extragerea de date dintr-o bază de date deja existentă, putându-se efectua filtrarea sau sortarea acestora;^[28]
- manipularea datelor – adăugarea, actualizarea, ștergerea și alte operații de modificare a datelor cum sunt combinarea datelor din mai multe tabele sau agregarea datelor;^[28]
- crearea sau modificarea structurii – crearea unei scheme sau a unui tabel, modificarea unui tabel, definirea tipurilor de data, definirea constrângerilor;^[28]
- controlul accesului și securității datelor – gestionarea utilizatorilor și privilegiilor acestora.^[28]

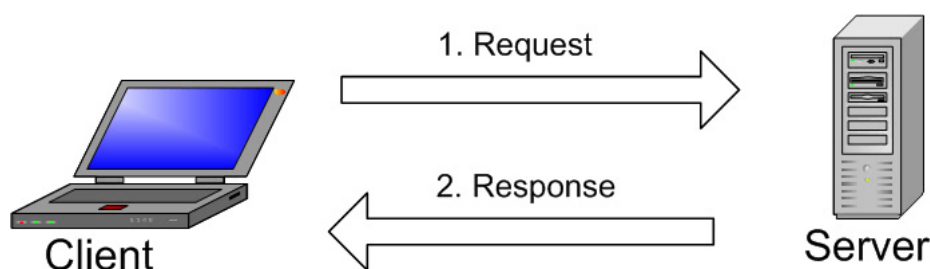


Fig. 2.3 – Modelul client-server^[28]

Clientul, care poate fi o aplicație dezvoltată într-un limbaj de programare, se leagă la server prin intermediul unei rețele. Fiecare client poate face o cerere către server pentru a primi datele dorite, cu ajutorul unei interfețe grafice, iar serverul va da la rândul său datele dorite de client, atâta timp cât ambele părți implicate formulează problema corect. În cazul MySQL, acesta creează o bază de date unde stochează și manipulează datele din fiecare tabel, clientul scrie o instrucțiune pentru a primi datele pe care le dorește, iar server-ul trimite datele cerute care vor apărea pe partea clientului.^[28]

MySQL este des folosit pentru: comerț electronic, adică pentru gestionarea tranzacțiilor, profilului clienților și conținutul acestora, rețele de socializare, jocuri online, marketing digital și altele.^[27]

2.1.7. API

API este un mecanism care facilitează comunicarea între două componente software cu ajutorul unui set de definiții și protocoale. Aceste componente comunică prin intermediul cererilor și răspunsurilor.^[30]

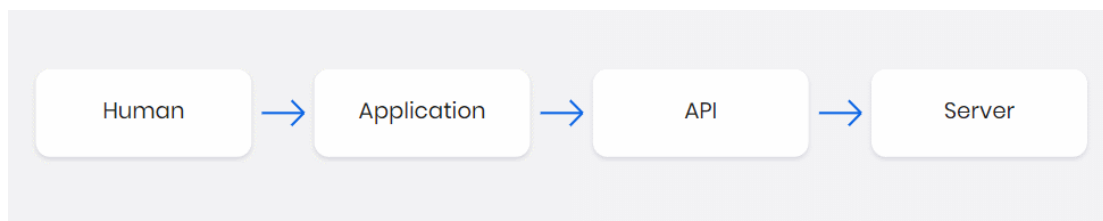


Fig. 2.4 – Modul în care funcționează API^[4]

Conform Fig. 2.4, utilizatorul are acces la aplicație prin intermediul interfeței grafice, unde poate executa diferite comenzi. Utilizatorul inițiază o cerere în cadrul aplicației, care, la rândul său, va iniția cu ajutorul API o cerere către server, API reprezentând intermediarul dintre aplicație și server.^[4]

REST API este un API web care face legătura între un browser web și un server web prin intermediul protocolului HTTP, care definește un set de metode, cum sunt GET, POST, PUT și DELETE, pentru a le permite utilizatorilor să acceseze datele din server. Clientul face o cerere către server prin intermediul metodelor respective, iar serverul îi va oferi un răspuns.^[30]

2.1.8. Apache

Apache HTTP Server este un software de server web responsabil de aprobarea cererilor HTTP primite din partea utilizatorilor și trimiterea răspunsurilor în care se află informațiile cerute sub formă de pagini web, adică le dă posibilitatea utilizatorilor să vadă conținutul paginilor de pe site-urile web. Cu alte cuvinte, Apache este răspunzător pentru ca serverul pe care este stocat site-ul web să facă legătura cu dispozitivul folosit de utilizator.^[31]

Apache este software-ul care rulează pe serverul web și care realizează o legătură între server și browserele web, pentru a le permite utilizatorilor, care folosesc acele browsere, să trimită cereri către server, iar Apache este răspunzător pentru gestionarea acestor cereri. Atunci când un utilizator realizează o cerere pentru a vizualiza pagina principală a unui site web, browserul web pe care îl folosește trimite acea cerere către server, Apache acceptă cererea și apoi trimite informațiile necesare de pe server către browser, astfel utilizatorul ajunge să vadă conținutul paginii web principale a site-ului web pe care l-a accesat.^[31]

2.2. Prezentarea tehnologiilor și mediilor de dezvoltare alese

2.2.1. HTML

HTML se va utiliza pe partea de frontend pentru a crea structura și conținutul paginilor web. Cu ajutorul acestuia se vor crea elemente precum paragrafe, câmpuri pentru introducerea datelor, liste, formulare, butoane, link-uri de legătură între paginile site-ului, secțiune de subsol a paginii.

Avantaje:

- este ușor de înțeles și implementat – stă la baza dezvoltării site-urilor web pe partea frontend, sintaxa sa este una simplă care constă din existența mai multor etichete, fiecare cu un scop specific; fiind ușor de interpretat, se pot lua bucăți de cod și se pot ajusta în funcție de necesități, permițând utilizatorilor să creeze și să structureze într-un mod rapid conținutul paginilor web;^[32]
- este gratuit – oricine îl poate utiliza fără a fi necesară instalarea altor software-uri;^[32]
- are compatibilitate universală – este compatibil cu majoritatea browserelor din toată lumea, cum sunt Google Chrome, Opera, Mozilla Firefox etc. Acest lucru sugerează că paginile web vor fi afișate corect indiferent de platforma și dispozitivul folosit;^[32]
- este ușor de editat – nu necesită existența unei platforme speciale, putând fi editat în orice editor de text;^[32]
- este ușor de integrat cu alte limbaje – limbaje precum CSS și JavaScript pot fi scrise direct în interiorul documentelor de HTML, fără a cauza probleme de compatibilitate;^[32]
- afișează modificările făcute imediat – modificările făcute în cod pot fi observate imediat după salvarea și reîncărcarea paginii la care se lucrează;^[32]
- este acceptat de cele mai multe instrumente de dezvoltare – majoritatea instrumentelor de dezvoltare web permit scrierea de cod HTML;^[32]
- este ușor de utilizat – fiind un limbaj simplu, nu necesită cunoștințe de alte limbaje de programare.^[32]

Dezavantaje:

- este un limbaj de marcare static – necesită ajutorul altor limbaje de programare pentru a crea pagini web dinamice și interactive; de asemenea, paginile vor rămâne la nivel de structură, fără ajutorul unui limbaj de stilizare care să îl transforme vizual;^[32]
- lipsa securității – HTML este limitat în ceea ce privește caracteristicile de securitate, site-urile devenind nesigure în fața atacurilor hackerilor, acestea având nevoie de alte limbaje de programare pentru a oferi securitate complementară;^[32]
- este nevoie de multe rânduri de cod chiar și pentru a realiza pagini web de complexitate redusă, ceea ce poate deveni greu de urmărit și gestionat;^[32]
- dezvoltatorii trebuie să folosească sintaxa implicită a HTML-ului, fără a putea devia de la aceasta.^[33]

2.2.2. CSS

CSS se va utiliza pe partea de frontend pentru a stiliza paginile web. Cu ajutorul său se vor defini aspecte ce țin de design-ul site-ului, cum ar fi definirea culorii fundalului, butoanelor, textelor, poziționarea elementelor în pagină, specificarea font-ului, dimensiunii și stilului textelor, definirea alineatelor și distanțelor între elemente, adăugarea de elemente textului cum ar fi sublinierea acestuia, definirea bordurilor elementelor, adăugarea de tranziții elementelor paginilor.

Avantaje:

- îmbunătățește experiența și atracția vizuală a utilizatorului – HTML nu este de ajuns pentru a face o pagină web atractivă pentru utilizatori, de aceea este nevoie de CSS pentru a aranja în pagină toate componentele, pentru a adăuga tranziții și animații, oferind, astfel, interactivitate și efecte dinamice, care vor face ca utilizatorii să rămână în continuare pe site-ul respectiv;^{[34][35]}
- este compatibil cu toate dispozitivele – cu ajutorul CSS-ului se pot crea pagini web care să se muleze pe orice dispozitiv ar folosi utilizatorul;^[34]
- viteza de încărcare a paginilor este mai mare – CSS este mai eficient decât dacă s-ar stiliza paginile web în HTML utilizând atributul “style”; totodată paginile încep să se încarce mai rapid atunci când browser-ul stochează în memoria cache stilurile acestora;^[36]
- este flexibil – permite schimbarea design-ului paginii web fără a fi nevoie să intervenim în codul de HTML;^[35]
- se pot face modificări asupra formatelor mai multor pagini, doar prin schimbarea codului unui singur fișier CSS, astfel, salvându-se mult timp și efort.^[34]

Dezavantaje:

- lipsa securității – CSS este limitat în ceea ce privește caracteristicile de securitate, site-urile devenind nesigure în fața atacurilor hackerilor, acestea având nevoie de alte limbaje de programare pentru a oferi securitate complementară;^[37]
- există probleme în cazul compatibilității cu browser-ele – stilurile CSS pot fi interpretate diferit în funcție de browser-ul care se utilizează, ceea ce poate duce la afișarea necorespunzătoare a paginilor.^[37]

2.2.3. Bootstrap

Bootstrap se va folosi pe partea de frontend, pentru a permite interfeței grafice a site-ului să se afișeze corect, indiferent de dimensiunea ecranului folosit.

Avantaje:

- este compatibil cu toate dispozitivele și implicit cu diferite dimensiuni ale ecranului – se pot crea pagini web care să se adapteze în funcție de dispozitivul folosit de utilizator;^[38]
- are o compatibilitate universală, fiind compatibil cu toate versiunile moderne ale browserelor, făcând paginile web să fie afișate corect indiferent de platforma folosită;^[18]
- dispune de componente predefinite care pot fi folosite direct în proiect, fiind deja scrise și stilizate, care ulterior se pot adapta cu ajutorul CSS-ului, în funcție de nevoi, nefiind nevoie să se implementeze codul de la zero;^[18]

- dispune de o documentație detaliată și ușor de citit, care facilitează scrierea de cod;^[18]
- cu ajutorul său, imaginile din interfața grafică a proiectului se pot redimensiona în funcție de dimensiunile ecranului pe care este deschis site-ul web.^[19]

Dezavantaje:

- datorită componentelor sale predefinite, este nevoie de multă stilizare, cu ajutorul CSS-ului, pentru a face proiecte diferite ca aspect, unele de celelalte;^[19]
- deși are compatibilitate universală cu versiunile noi ale browserelor, acest lucru nu este valabil când vine vorba de versiunile mai vechi ale acestora, ceea ce înseamnă că ceea ce vede utilizatorul depinde de versiunea browserului pe care o folosește.^[19]

2.2.4. Vue.js

Vue.js se va folosi pe partea de frontend pentru a adăuga interactivitate paginilor web, pentru a crea pagini dinamice care răspund la acțiunile întreprinse de utilizatori, dar și pentru a apela rute, care le va permite utilizatorilor să acceseze datele din baza de date și, astfel, se vor face modificări în interfața grafică a platformei. Cu ajutorul său se vor crea componente care conțin HTML, CSS și JavaScript (Vue.js) și care vor alcătui paginile web ale platformei.

Avantaje:

- este ușor de înțeles și folosit – are o structură simplă ceea ce îl face ușor de înțeles, îngăduindu-le dezvoltatorilor să adauge, fără prea mare greutate, Vue.js proiectelor lor web; totodată le permite acestora să păstreze arhitectura veche a proiectelor peste care pot adăuga componentele noi scrise în Vue.js;^[39]
- este alcătuit din componente care se pot reutiliza, un singur fișier conținând HTML, CSS și JavaScript;^[40]
- are o documentație detaliată care ajută orice dezvoltator să învețe și să înțeleagă limbajul cu ușurință, permițându-le să capete cunoștințe de bază pentru a crea o aplicație;^[40]
- necesită cunoștințe de bază de HTML, CSS și JavaScript pentru a putea fi folosit, spre deosebire de AngularJS sau React, care necesită cunoștințe și de alte limbaje de programare;^[39]
- este folosit pentru a adăga interactivitate site-urilor; totodată este răspunzător de logica care este vizibilă utilizatorilor.

Dezavantaje:

- este limitat în ceea ce privește bibliotecile și pachetele externe de care dispune, pentru a extinde funcționalitățile acestuia, spre deosebire de celelalte framework-uri de JavaScript, AngularJS și React;^[21]
- deoarece oferă o flexibilitate generoasă, le dă posibilitatea dezvoltatorilor să utilizeze diferite abordări, ceea ce poate deveni o problemă în cazul proiectelor mari, deoarece duce la o inconsecvență a codului.^[41]

2.2.5. Laravel

Laravel se va folosi pentru partea de backend a aplicației, pentru a realiza logica din spatele acesteia. Se vor folosi funcționalitățile, pe care le are încorporate, de autentificare a utilizatorilor, de resetare a parolei și de trimitere a e-mail-urilor către utilizatori. Se va crea schema bazei de date cu ajutorul migrărilor, se vor crea modele pentru fiecare tabel din baza de date unde se vor mapa datele acestora și se vor stabili și relațiile dintre tabele cu ajutorul Eloquent ORM. Cu ajutorul algoritmilor de criptare pe care îi are, parolele utilizatorilor se vor salva în baza de date sub formă criptată. În controllere se vor crea rutele, care vor face legătura între backend și frontend și care vor gestiona cererile utilizatorilor. În șabloanele Blade se vor adăuga componentele de Vue.js, care mai apoi vor fi afișate în interfața grafică. Totodată, se vor crea funcții care vor gestiona logica interacțiunilor utilizatorului cu site-ul.

Avantaje:

- vine cu funcționalități încorporate pentru autentificarea și autorizarea utilizatorilor, acestea fiind compuse din înregistrarea și autentificarea utilizatorilor, dar și din resetarea parolei;^[42]
- migrarea ușoară a datelor – permite dezvoltatorilor să gestioneze schema bazei de date, oferind o sintaxă simplă pentru crearea, modificarea și ștergerea tabelelor și coloanelor acestora din baza de date;^[43]
- cu ajutorul Eloquent ORM se pot proiecta modele care se corelează cu tabelele din baza de date, relațiile dintre tabele se stochează în ORM, iar în cazul în care apar modificări în structura tabelelor, datele asociate se vor actualiza;^[44]
- are o documentație detaliată care ajută orice dezvoltator să învețe și să înțeleagă limbajul cu ușurință;^[44]
- este compus dintr-o arhitectură MVC, care oferă performanțe îmbunătățite, prin împărțirea codului în partea de logică, cea de gestionare a cererilor utilizatorilor și cea de prezentare și afișare a datelor;^[42]
- oferă suport pentru gestionarea cache-ului, pentru a păstra în memorie datele des utilizate și pentru a spori, astfel, performanțele sistemului;^{[23][25][42]}
- are funcție de e-mail integrată pentru a putea trimite e-mail-uri, facilitând comunicarea cu utilizatorii;^[42]
- oferă securitate încorporată, folosește algoritmi pentru criptarea parolelor utilizatorilor, salvând-o în baza de date criptată și nu ca o formă simplă de text;^[42]
- acceptă testarea automată, oferind posibilitatea testării pe bucăți a codului;^[42]
- permite definirea și gestionarea într-un mod simplu a rutelor URL ale aplicației, acest sistem oferind flexibilitate în definirea rutelor și în asocierea acestora cu controllerele specifice, astfel se facilitează gestionarea fluxului de navigație pentru ca utilizatorii să gasească cu ușurință conținutul sau pagina pe care și-o dorește;^[42]
- prezintă sistemul de șablonare Blade, care facilitează scrierea și organizarea codului HTML;
- are propria interfață de linie de comanda, Artisan, care este folosit pentru a gestiona baza de date, pentru a crea controllere, modele, liste și alte secțiuni ale unei aplicații;^[23]

- are un sistem integrat de gestionare a cozilor, care elimină sarcinile care nu mai sunt relevante sau esențiale și le include într-o coadă.^[44]

Dezavantaje:

- nu oferă suport direct pentru procesarea plăților, fiind nevoie de procesoare de plată, un exemplu fiind PayPal;^[43]
- la prima vedere este un framework complex, cu o mulțime de funcționalități și caracteristici, care, la început, poate fi greu de înțeles.^[43]

2.2.6. MySQL

MySQL se va folosi pentru a crea schema bazei de date cu tabelele aferente și relațiile dintre acestea, dar și pentru a crea interogări cu ajutorul cărora se vor manipula datele din interiorul bazei de date.

Avantaje:

- este compatibil cu o mulțime de platforme, cum sunt Windows, Linux, macOS, ceea ce îl face să poată fi folosit pe orice dispozitiv, fiind foarte utilizat pentru dezvoltarea aplicațiilor web;^{[45][46]}
- oferă o securitate sporită, oferind suport pentru autentificarea utilizatorilor, criptare și pentru gestionarea accesului personalului la baza de date, protejând datele de persoanele care nu sunt autorizate;^[45]
- este ușor de învățat și folosit pentru că utilizează limbajul SQL care permite manipularea datelor, având o sintaxă simplă și ușor de aplicat;^[45]
- este realizat să poată gestiona cantități mari de date fără a-i scădea performanțele;^[45]
- este ușor de integrat cu alte limbaje de programare pentru a realiza partea de backend a unei aplicații.^[47]

Dezavantaje:

- deși poate gestiona o cantitate mare de date, atunci când se realizează un transfer mare de date sau interogări complexe, viteza sa se reduce.^{[46][48]}

2.2.7. REST API

REST API se va folosi pentru a lega interfața grafică a aplicației de informațiile din baza de date, cu ajutorul protocolului HTTP care dispune de metodele GET, POST, PUT, DELETE, care le permit utilizatorilor să facă cereri care accesează serverul legat la datele din baza de date, acesta oferindu-le răspunsuri la cererile respective.

Avantaje:

- este utilizat pentru a adăuga aplicații noi la funcționalitățile unei aplicații web deja existente;^[30]
- este independent de platforma sau de limbajul de programare folosit, ceea ce permite să fie utilizate în diferite sisteme software;^[49]
- este folosit pentru a face o separare între client și server, astfel modificările făcute unei componente nu o va afecta pe cealaltă;^[30]

- permite reutilizarea codului și funcționalităților din aplicațiile existente, dezvoltatorii economisind timp și reducând munca depusă;^[49]
- poate fi utilizat indiferent de limbajul de programare sau framework-ul folosit în realizarea aplicației.^[49]

Dezavantaje:

- poate prezenta o vulnerabilitate în cazul securității, ceea ce implică implementarea unor măsuri de securitate corespunzătoare;^[49]
- poate avea o configurare și o gestionare complexă în cazul aplicațiilor complexe;^[49]
- poate prezenta probleme de compatibilitate cu diverse versiuni ale aceluiași sistem software, rezultând un sistem cu erori.^[49]

2.2.8. Apache

Apache HTTP Server se va folosi pentru a putea rula platforma web pe un server local. Pentru a rula serverul web Apache se folosește WampServer, care este un mediu de dezvoltare pentru Windows.

Avantaje:

- este un software stabil, deoarece este folosit la scară largă de multă vreme;^[50]
- este flexibil grație structurii sale împărțită pe module care le permite dezvoltatorilor să facă configurări în funcție de nevoile site-ului;^[50]
- este simplu de configurat;^[50]
- este disponibil pentru o mulțime de sisteme de operare, inclusiv Linux, Windows, macOS.^[50]

Dezavantaje:

- pot apărea probleme legate de performanță în cazul site-urilor web cu o cantitate mare de trafic;^[50]
- o flexibilitate prea mare în ceea ce privește configurările poate duce la vulnerabilități cu privire la securitate.^[50]

2.2.9. PhpStorm

PhpStorm este un mediu integrat de dezvoltare în care se va scrie codul aferent realizării logicii și interfeței grafice a site-ului web.

Avantaje:

- se pot realiza redenumiri și mutări de fișiere, funcții, variabile pentru a ajuta la refactorizarea codului unui proiect, actualizându-se automat referințele către elementele care au fost modificate fără să apară erori;^[51]
- oferă funcționalități de autocompletare și sugestii, atunci când sunt scrise linii de cod, tot procesul de scriere a codului devenind mai rapid și mai precis;^[51]
- conține formatare de cod încorporată, care ajută la aranjarea codului și la colorarea elementelor acestuia, în mod distinct, pentru a fi ușor de citit și de înțeles;^[51]

- efectuează analiza codului pentru a detecta erori și oferă sugestii de remediere a acestora;^[51]
- permite integrarea cu baze de date și oferă posibilitatea modificării tabelor bazei de date, dar și a interogării datelor.

2.2.10. DataGrip

DataGrip este un mediu integrat de dezvoltare care se va folosi pentru a vizualiza și gestiona mai ușor datele din baza de date.

Avantaje:

- dispune de un editor care permite adăugarea, editarea, eliminarea sau clonarea liniilor cu date din tabele;^[52]
- are un editor de text, unde se pot scrie interogări, care oferă sugestii de completare a codului în funcție de datele care se află în baza de date;^[52]
- conține formatare de cod încorporată, care ajută la aranjarea codului și la colorarea elementelor acestuia, în mod distinct, pentru a fi ușor de citit și de înțeles;^[52]
- efectuează analiza codului scris pentru a detecta erori și oferă sugestii de remediere a acestora;^[52]
- atunci când se realizează redenumiri ale variabilelor sau tabelor, toate referințele din cod se actualizează automat fără să apară erori;^[52]
- oferă funcționalități de import și export de date sub diferite formate.^[52]

3. Implementarea aplicației

3.1. Implementarea bazei de date

Întreaga structură a bazei de date este reprezentată în Fig. 3.1. Codul utilizat pentru a crea tabelele structurii se regăsește în *Anexa 1*, iar relațiile dintre acestea se găsesc în *Anexa 2*, la finalul acestei lucrări.

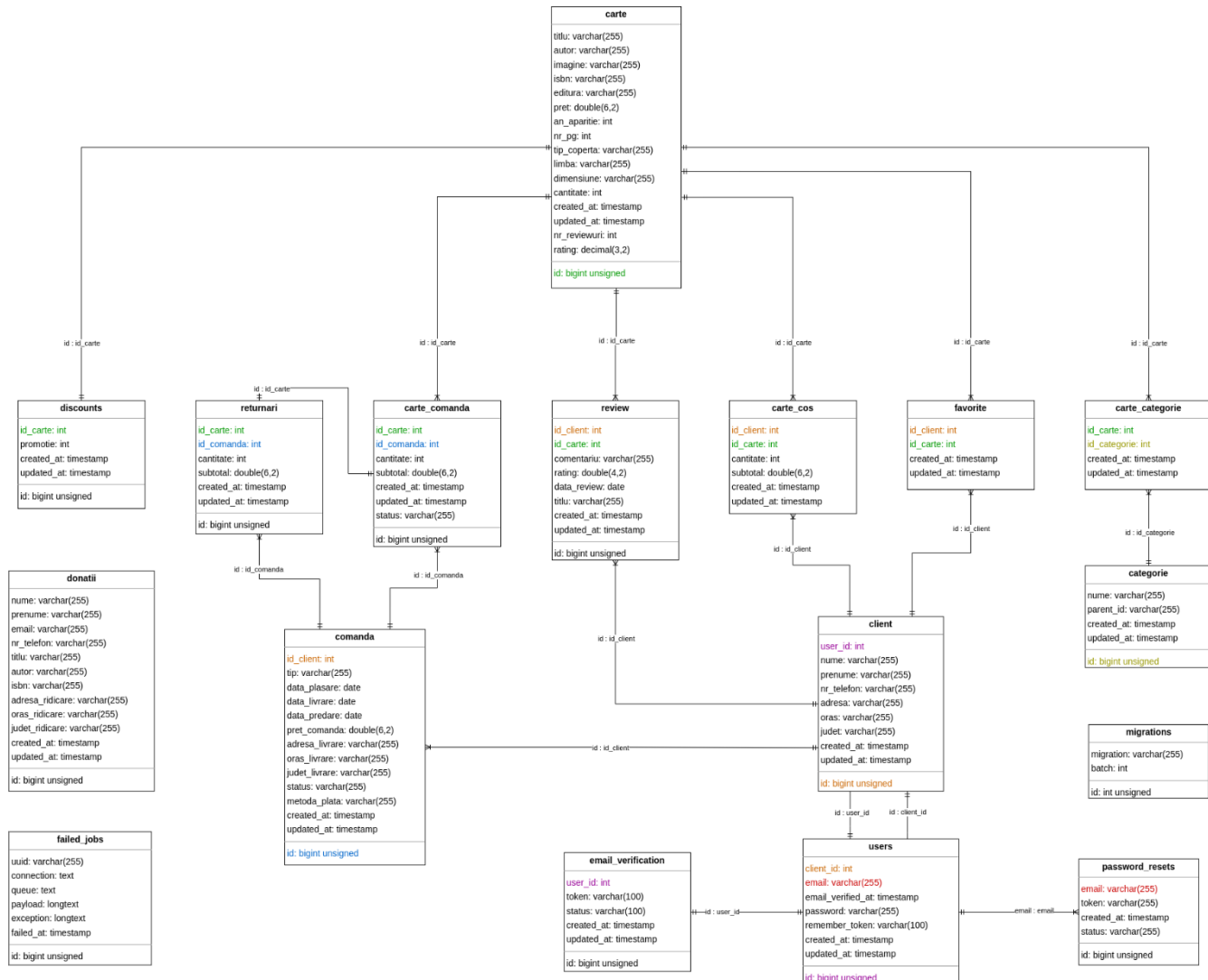


Fig. 3.1 – Arhitectura bazei de date

Pentru a crea toate funcționalitățile platformei, astfel încât să poată fi folosită de către utilizatori, este nevoie, în primul rând, de a crea schema bazei de date, care să conțină toate tabelele necesare, dar și relațiile dintre acestea. Pentru proiectarea întregii platforme, este nevoie de realizarea a șaptesprezece tabele, după cum urmează:

- Entitatea „carte”, care are următoarele atribute: id, titlu, autor, imagine, isbn, editură, preț, an_apariție, nr_pg, tip_copertă, limbă, dimensiune, cantitate, nr_reviewuri, rating, created_at, updated_at, conține toate detaliile necesare pentru a înștiința utilizatorul de aspectele și conținutul unei cărți;

- Entitatea „discounts”, care are următoarele attribute: id, id_carte, promoție, created_at, updated_at, prezintă procentul de reducere al unei cărți; această tabelă prezintă o relație 1:1 cu tabela „carte”, care se leagă la ea prin cheia străină id_carte;
- Entitatea „returnari”, care are următoarele attribute: id, id_carte, id_comandă, cantitate, subtotal, created_at, updated_at, prezintă cărțile care sunt returnate dintr-o comandă, fiind într-o relație de 1:1 cu tabela „carte-comandă” ;
- Entitatea „carte_comanda”, care are următoarele attribute: id, id_carte, id_comandă, cantitate, subtotal, created_at, updated_at, status, este un tabel pivot între *carte* și *comandă*, deoarece între cele două există o relație M:N, o carte putându-se afla în mai multe comenzi, iar o comandă poate conține mai multe cărți;
- Entitatea „review”, care are următoarele attribute: id, id_client, id_carte, comentariu, rating, data_review, titlu, created_at, updated_at, se referă la recenziile date de către client unei cărți; această tabelă se află într-o relație N:1 cu tabela „carte”, dar și cu tabela „client”, deoarece o carte poate avea mai multe recenzii, iar un client poate lăsa recenzii mai multor cărți;
- Entitatea „carte_cos”, care are următoarele attribute: id_client, id_carte, cantitate, subtotal, created_at, updated_at, este un tabel pivot între *carte* și *client*, deoarece între cele două există o relație M:N, o carte putându-se afla în coșul mai multor clienți, iar un client poate avea mai multe cărți în coș;
- Entitatea „favorite”, care are următoarele attribute: id_client, id_carte, created_at, updated_at, reprezintă cărțile favorite ale unui client care există în lista sa de dorințe; între tabela „carte” și tabela „favorite” există o relație 1:N, deoarece o carte se poate afla în mai multe liste de favorite;
- Entitatea „carte_categorie”, care are următoarele attribute: id_carte, id_categorie, created_at, updated_at, este un tabel pivot între *carte* și *categorie*, deoarece între cele două există o relație M:N, o carte putând avea mai multe categorii, iar o categorie conține mai multe cărți;
- Entitatea „categorie”, care are următoarele attribute: id, id_carte, id_categorie, created_at, updated_at, se referă la categoriile pe care le pot avea cărțile;
- Entitatea „comanda”, care are următoarele attribute: id, id_client, tip, dată_plasare, dată_livrare, dată_predare, preț_comandă, adresă_livrare, oraș_livrare, județ_livrare, status, metodă_plată, created_at, updated_at, prezintă toate detaliile unei comenzi care a fost plasată; attributele „tip” și „data_predare” se referă la posibilitatea implementării funcției de închiriere a cărților; tabela „comanda” se află într-o relație 1:N cu tabela „returnari”, deoarece o comandă poate avea mai multe cărți returnate;
- Entitatea „client”, care are următoarele attribute: id, user_id, nume, prenume, nr_telefon, adresă, oraș, județ, created_at, updated_at, conține datele personale ale utilizatorilor și se află într-o relație 1:1 cu tabela „users”, care se leagă la ea prin cheia străină „user_id”;
- Entitatea „users”, care conține următoarele attribute: id, client_id, email, email_verified_at, password, remember_token, created_at, updated_at, conține datele de autentificare ale utilizatorilor și se află într-o relație 1:1 cu tabela „client”, prin cheia străină „client_id”;
- Entitatea „email_verification”, care are următoarele attribute: id, user_id, token, status, created_at, updated_at, reprezintă e-mailul pe care utilizatorul îl primește la crearea unui cont pentru a-și confirma identitatea; această tabelă se află într-o relație 1:1 cu tabela „users”, care se leagă la ea prin cheia străină „user_id”;

- Entitatea „password_reset”, care conține următoarele atribute: id, email, token, created_at, status, se referă la posibilitatea utilizatorilor de a-și schimba parola, în cazul în care o uită sau o pierde; se află într-o relație N:1 cu tabela „users”, deoarece un utilizator își poate reseta parola de mai multe ori;
- Entitatea „donatii”, care conține următoarele atribute: id, nume, prenume, email, nr_telefon, titlu, autor, isbn, adresă_ridicare, oraș_ridicare, județ_ridicare, created_at, updated_at, se referă la datele pe care le conține o donare: datele donatorului, ale cărților donate și ale adresei de preluare a cărților;
- Entitatea „failed_jobs”, care conține următoarele atribute: id, uuid, connection, queue, payload, exception, failed_at, este creată de Laravel pentru monitorizarea listelor în cadrul cărora au apărut erori;
- Entitatea „migrations”, care are următoarele atribute: migration, batch, id, este creată de Laravel pentru monitorizarea statusului migrărilor efectuate.

Atributele created_at și updated_at sunt create implicit de Laravel, pentru a arată momentul în care s-au adăugat înregistrări noi în tabele, dar și momentul în care s-au făcut modificări asupra înregistrărilor existente.

3.2. Implementarea funcționalităților

Funcționalitățile platformei sunt implementate folosind o parte din componentele puse la dispoziție de framework-ul Laravel.

3.2.1. Controller

În cadrul acestor componente sunt implementate funcții, ce sunt accesate prin intermediul rutelor web. Asemănător modelelor, fiecare tabel din baza de date are un controller, unde există funcții pentru manipularea datelor și pentru organizarea acestora, în vederea transmiterii lor în partea de frontend. Printre aceste funcții, se enumeră implementări standard RESTful API:

- funcția *index*: acest tip de funcție are ca scop preluarea tuturor datelor dintr-un tabel și transmiterea acestora ca răspuns de tip JSON;

```

/**
 * @OA\Get(
 *   path="/categorii",
 *   summary="Returneaza toate categoriile",
 *   tags={"Categorii"},
 *   description="Returneaza toate categoriile",
 *   @OA\Response(
 *     response=200,
 *     description="Success"
 *   )
 * )
 */
public function index(): JsonResponse
{
    $data = Categorie::all();
    return response()->json($data, status: Response::HTTP_OK);
}

```

Fig. 3.2 – Funcția index din CategorieController

În Fig. 3.2 este evidențiată implementarea unei funcții index. De asemenea, este vizibilă implementarea unor comentarii folosite de Swagger UI, o unealtă care permite testarea ușoară a rutelor implementate.

- funcția *show*: acestei funcții îi este transmis un parametru, în general cheia primară a tabelului, însă acest lucru nu este obligatoriu (de exemplu, se poate transmite o cheie străină, dacă căutarea după această cheie este mai relevantă). În final, se va extrage o singură linie (înregistrare) din tabel, aceea în care se regăsește cheia transmisă ca parametru;

```

/**
 * @OA\Get(
 *   path="/categorii/{id}",
 *   summary="Returnează categoria cu id-ul dat",
 *   tags={"Categorii"},
 *   description="Returnează categoria cu id-ul dat",
 *   @OA\Parameter(
 *     name="id",
 *     description="Id-ul categoriei dorite",
 *     required=true,
 *     in="path",
 *     example=3,
 *     @OA\Schema(
 *       type="integer"
 *     )
 *   ),
 *   @OA\Response(
 *     response=200,
 *     description="Success"
 *   )
 * )
 */
public function show($id): JsonResponse
{
    $data = Categorie::where('id', '=', $id)->first();
    return response()->json($data, status: Response::HTTP_OK);
}

```

Fig. 3.3 – Funcția show din CategorieController

În Fig. 3.3 este prezentată implementarea funcției show, care primește ca parametru un id, acesta reprezentând cheia primară din tabelul categorie. Prin intermediul modelului Categorie se realizează legătura cu tabelul dorit, iar sintaxa de Eloquent ORM a Laravel-ului este transformată într-o sintaxă MySQL ce va avea ca rezultat o linie din tabelul „categorie”.

- funcția *store*: prin intermediul acestei funcții se pot adăuga date într-un tabel. Datele sunt transmise printr-o rută POST, acestea făcând parte din Payload. Acestea sunt preluate prin intermediul variabilei de tip Request *\$request* și sunt transmise mai departe unei funcții dintr-un serviciu, alături de o instanță nouă a modelului dorit. În urma apelului, vom avea instanța de model în care se vor regăsi datele din *\$request*, organizate corespunzător. În final, se va folosi funcția *save()*, de care dispune Laravel, pentru a crea o nouă intrare în baza de date, în tabelul dorit, cu datele salvate în instanța de model;

```

/**
 * @OA\Post(
 *   path="/categorii",
 *   summary="Adauga categoria",
 *   tags={"Categorii"},
 *   description="Adauga categorii",
 *   @OA\RequestBody(
 *     required=true,
 *     @OA\JsonContent(
 *       @OA\Property(
 *         property="parent_id",
 *         type="integer",
 *         description="",
 *         example=1,
 *       ),
 *       @OA\Property(
 *         property="nume",
 *         type="string",
 *         description="",
 *         example="",
 *       )
 *     )
 *   ),
 *   @OA\Response(
 *     response=200,
 *     description="Success"
 *   )
 * )
 */
public function store(Request $request): JsonResponse
{
    $data = new Categorie;
    $this->categorieService->setProperties($data, $request->all());
    $data->save();
    return response()->json($data, status: Response::HTTP_OK);
}

```

Fig. 3.4 – Funcția store din CategorieController

În Fig. 3.4 se poate observa implementarea unei funcții store, în mod specific, pentru CategorieController.

- funcția *update*: această funcție are ca scop actualizarea datelor dintr-un tabel. Datele sunt transmise prin intermediul unei rute PUT. Spre deosebire de funcția store, în cazul actual, trebuie transmisă și o cheie primară. Datele noi sunt organizate apelând o funcție dintr-un serviciu și apoi salvate;

```

/**
 * @OA\Put(
 *   path="/categorii",
 *   summary="Update categoria cu id-ul dat",
 *   tags={"Categorii"},
 *   description="Update categorii cu id-ul dat",
 *   @OA\RequestBody(
 *     required=true,
 *     @OA\JsonContent(
 *       @OA\Property(
 *         property="id",
 *         type="integer",
 *         description="",
 *         example=1,
 *       ),
 *       @OA\Property(
 *         property="parent_id",
 *         type="integer",
 *         description="",
 *         example=1,
 *       ),
 *       @OA\Property(
 *         property="nume",
 *         type="string",
 *         description="",
 *         example="",
 *       )
 *     )
 *   ),
 *   @OA\Response(
 *     response=200,
 *     description="Success"
 *   )
 * )
 */
public function update(Request $request): JsonResponse
{
    $data = Categoriae::where('id', '=', $request['id'])->first();
    $this->categoriaService->setProperties($data, $request->all());
    $data->save();
    return response()->json($data, status: Response::HTTP_OK);
}

```

Fig. 3.5 – Funcția update din CategoriaeController

În Fig. 3.5 este prezentată implementarea unei funcții update. De asemenea, se poate observa documentația Swagger UI, care permite testarea, cu ușurință, a metodelor implementate.

- funcția *destroy*: această metodă este utilizată în cazul în care se dorește ștergerea unei linii dintr-o tabelă. Aceasta primește ca parametru un id, reprezentând cheia primară a tabelului.

```

/**
 * @OA\Delete(
 *   path="/categorii/{id}",
 *   summary="Sterge categoria cu id-ul dat",
 *   tags={"Categorii"},
 *   description="Sterge categoria cu id-ul dat",
 *   @OA\Parameter(
 *     name="id",
 *     description="Id-ul categoriei dorite",
 *     required=true,
 *     in="path",
 *     example=3,
 *     @OA\Schema(
 *       type="integer"
 *     )
 *   ),
 *   @OA\Response(
 *     response=200,
 *     description="Success"
 *   )
 * )
 */
public function destroy($id): JsonResponse
{
    $data = Categorie::where('id', '=', $id)->first();
    $data->delete();
    return response()->json( data: "DELETED", status: Response::HTTP_OK);
}

```

Fig. 3.6 – Funcția destroy din CategorieController

Așa cum se poate observa în Fig. 3.6 , implementarea unei funcții destroy constă în preluarea liniei dintr-un tabel, prin intermediul modelului, și ștergerea acesteia cu ajutorul funcției *delete()* pe care ne-o pune la îndemână Laravel.

Pe lângă aceste funcții, a fost necesară implementarea mai multor metode care asigură funcționalități cheie în cadrul proiectului. Printre aceste funcționalități se numără:

1. Căutare: căutarea se poate face după numele cărții, autor sau ISBN. Pentru a face posibil acest lucru, am implementat funcțiile *search* și *paginaCautare* din CarteController, care se găsesc în Anexa 3. Funcția *search* primește, prin intermediul unei rute POST, textul după care utilizatorul dorește să facă căutarea. Prin intermediul unui interogări, se extrag toate cărțile care au numele, autorul sau ISBN-ul asemănătoare cu textul primit. Datele sunt organizate într-o matrice care este transmisă mai departe printr-o rută. Această rută apelează funcția *paginaCautare*, de unde se redirecționează către pagina de căutare, alături de setul de date rezultat în urma interogării din funcția *search*. Funcția *paginaCautare* este, de asemenea, apelată atunci când se accesează pagina de căutare prin intermediul navigației, în acest caz preluându-se toate cărțile existente;
2. Preluarea datelor din filtre: în pagina de căutare, utilizatorul are posibilitatea de a filtra rezultatele care să îi fie afișate. Categoriile de filtre existente sunt preț, categorie, autor, editură și limbă. Atunci când pagina de căutare este accesată, componenta *BookFilter.vue* trimite o cerere pentru preluarea datelor ce urmează a fi folosite pentru filtrare (de

exemplu, numele categoriilor existente, numele autorilor existenți etc.). Funcția care se ocupă de preluarea și prelucrarea datelor este *getFilters* din *CarteController*. Aceasta preia date din tabelele necesare prin intermediul unor interogări și le organizează într-o matrice ce poate fi folosită cu ușurință în componenta de frontend;

```
/**
 * @return JsonResponse
 */
public function getFilters()
{
    $autori = DB::table( table: 'carte' )->select( columns: 'autor' )->distinct()->get();
    $edituri = DB::table( table: 'carte' )->select( columns: 'editura' )->distinct()->get();
    $categorii = DB::table( table: 'categorii' )->select( columns: 'nume' )->distinct()->get();
    $limbi = DB::table( table: 'carte' )->select( columns: 'limba' )->distinct()->get();

    $data = [
        'autori' => $autori,
        'edituri' => $edituri,
        'categorii' => $categorii,
        'limbi' => $limbi,
    ];

    return response()->json($data, status: Response::HTTP_OK);
}
```

Fig. 3.7 – Funcția *getFilters* din *CarteController*

Așa cum se poate observa în Fig. 3.7, tabelele accesate în cadrul funcției *getFilters* sunt *carte* și *categorii*.

3. Filtrarea și ordonarea rezultatelor: atunci când un filtru este selectat de către utilizator, componenta *BookFilter.vue* face apel la backend cu scopul de a actualiza lista de cărți pentru a satisface dorințele utilizatorului. De acest lucru se ocupă funcția *queryBuilder* din *CarteController*. Aceasta primește lista de filtre pe care utilizatorul le-a selectat și compune o interogare în funcție de valoarea fiecărui filtru. Categoriile de filtre, unde utilizatorul nu a selectat nici o opțiune vor transmite valoarea null, astfel se pot ignora filtrele neselectate. De asemenea, funcția primește și opțiunea de ordonare care poate avea valoarea null, în cazul în care nu a fost selectată. După ce interogarea a fost construită, se extrag cărțile. Pentru fiecare carte obținută se recalculează media recenziilor și numărul acestora, pentru ca rezultatele să fie în permanență de actualitate;
4. Căutarea unei cărți după ISBN: atunci când o carte este selectată, utilizatorul va fi redirecționat către pagina cărții respective. URL-ul pe care acesta va fi redirecționat va conține la final ISBN-ul cărții respective. De asemenea, pagina unei cărți poate fi accesată direct, prin același URL. Astfel, pentru a putea prelua datele cărții respective, ISBN-ul este transmis ca parametru funcției *getBookByIsbn* din *CarteController*, care preia datele cărții din baza de date și le transmite către componenta frontend, pentru afișare;
5. Crearea listei de cărți recomandate: atunci când pagina unei cărți este accesată, utilizatorului îi este prezentată o listă de cărți recomandate. Aceste cărți fac parte din

aceeași categorie sau au același autor cu cartea accesată. De această funcționalitate se ocupă metoda *getRecommendations* din *CarteController*;

```
/**
 * @param int $id
 * @return JsonResponse
 */
public function getRecommendations(int $id)
{
    $carte = Carte::find($id);

    $categorieIds = $carte->categoriai()->pluck('id')->toArray();

    $recommendations = Carte::where('autor', $carte->autor)
        ->orWhereHas('categoriai', function ($query) use ($categorieIds) {
            $query->whereIn('id', $categorieIds);
        })
        ->get();

    $recommendations = $recommendations->reject(function ($recommendation) use ($carte) {
        return $recommendation->titlu === $carte->titlu;
    });

    return response()->json($recommendations, status: Response::HTTP_OK);
}
```

Fig. 3.8 – Funcția *getRecommendations* din *CarteController*

Așa cum se poate vedea în Fig. 3.8 , funcția este compusă din câțiva pași simpli: se caută cartea a cărei pagină a fost accesată și se extrag ID-urile categoriilor în care aceasta se încadrează. Apoi se caută cărțile care au același autor sau fac parte din una dintre categoriile găsite. În final, se șterge din colecție cartea inițială.

6. Listarea cărților cu reduceri: pe pagina principală se găsește o listă de cărți asupra cărora se aplică o reducere. Această funcționalitate este implementată în funcția *getDiscountedCarti* din *CarteController*. Aici se extrage lista ID-urilor cărților asupra cărora se aplică o reducere, iar apoi se extrag cărțile ale căror ID-uri se găsesc în acea listă;
7. Lista celor mai noi cărți: de asemenea, pe pagina principală se găsește și o secțiune cu cele mai noi cărți. Funcționalitatea se găsește în metoda *getLatestCarti* din *CarteController*. În cadrul acesteia se folosește o interogare care sortează toate cărțile după data în care au fost introduse în baza de date și le extrage pe cele mai recente 20;
8. Returnarea unei cărți: un utilizator își poate vedea lista cu comenzile plasate în secțiunea „Comenzile mele”. În cadrul fiecărei comenzi, utilizatorul poate să listeze, prin apăsarea unui buton, cărțile care fac parte din acea comandă. Fiecare carte are în dreptul său un buton de retur. Odată ce butonul de retur este apăsător, statusul cărții din comandă este schimbat. Acest lucru se realizează prin intermediul funcției *returnOrder* din *CarteComandaController*. Mai întâi se verifică ca utilizatorul să fie autentificat, apoi se extrage cartea dorită din comanda selectată și statusul ei este schimbat în „returnată”;
9. Afișarea cărților din comenzi: De asemenea, în pagina de profil există o secțiune în care utilizatorul poate vedea ce cărți a comandat. Aceste cărți nu sunt legate de o comandă, este

o listă comună în care fiecare carte apare o singură dată. Pentru a face posibilă această funcționalitate, am implementat metoda *carti* din *ComandaController*. În cadrul acesteia, se verifică ca utilizatorul să fie autentificat și, dacă acesta este, se preiau datele de utilizator și de client ale acestuia, urmând ca datele să fie prelucrate în vederea obținerii unei liste de ID-uri ale cărților comandate, aceasta fiind folosită într-o interogare care are ca rezultat o colecție de cărți distincte;

10. Afișarea cărților dintr-o comandă: pentru crearea listei de cărți care aparțin unei comenzi, am implementat funcția *cartiComanda* din *ComandaController*. Aceasta are ca scop preluarea, din baza de date, a tuturor cărților care fac parte din comanda cu ID-ul transmis ca parametru;
11. Trimiterea comenzii: pentru ca o comandă să fie salvată în baza de date, este necesar, ca inițial, să se creeze o intrare nouă în tabelul comandă, unde sunt salvate datele de facturare, livrare și alte informații necesare, urmând ca fiecare carte care există în coșul utilizatorului să fie atașată comenzii, prin adăugarea acestora în tabelul „carte_comanda”. Pe măsură ce cărțile sunt atașate comenzii, acestea se șterg din coșul utilizatorului. După ce comanda a fost salvată cu succes, utilizatorul va primi un e-mail de confirmare care va conține ID-ul comenzii. Această funcționalitate este asigurată de metoda *checkout* din *ComandaController*;
12. Primirea mesajelor din formularul de contact: în pagina principală, utilizatorul are la îndemână un formular prin intermediul căruia poate transmite un mesaj echipei Ink&Paper. Trimiterea formularului este posibilă datorită funcției *sendContactMail* din *Controller*. Aceasta primește ca parametrii e-mailul utilizatorului și mesajul acestuia și transmite aceste informații într-un șablon de e-mail care este apoi trimis la adresa `inkpaper2023@hotmail.com`;

```
/**
 * @param Request $request
 * @return void
 */
1 usage
public function sendContactMail(Request $request)
{
    $validatedData = $request->validate([
        'message' => 'required|string',
        'email' => 'required|email'
    ]);

    $message = $validatedData['message'];
    $email = $validatedData['email'];

    Mail::to( users: 'inkpaper2023@hotmail.com' )->send(new ContactMail($message, $email));
}
```

Fig. 3.9 – Funcția *sendContactMail* din Controller

În Fig. 3.9 se poate observa faptul că e-mailul este trimis adresei specificate, apelând șablonul *ContactMail*.

13. Importarea unor date de test: pentru a ușura procesele de dezvoltare și testare ale proiectului, am creat mai multe funcții pentru importarea unor date de testare care au fost salvate în format CSV;
14. Recenziile unei cărți: atunci când pagina unei cărți este accesată, se vor afișa date referitoare la recenziile pe care alți utilizatori le-au dat cărții respective. De aceea am implementat funcția *getByBookId* din *ReviewController*. Aceasta procesează datele preluate din baza de date referitoare la recenziile cărții, astfel obținând o clasificare în funcție de numărul de stele, dar și media notelor acordate, numărul recenziilor și o matrice în care se organizează datele din tabelul review cu datele clientului care l-a scris. În final, se vor salva numărul și media notelor în tabelul carte;
15. Schimbarea parolei: un utilizator are posibilitatea de a-și schimba parola. Cu acest scop a fost implementată funcția *changePassword* din *UserController*. Mai întâi se verifică dacă utilizatorul este autentificat și se extrag datele acestuia, urmând apoi să se verifice dacă parola nouă corespunde cerințelor impuse, prin apelarea unei funcții din *UserService*. Dacă funcția nu întoarce erori, parola nouă este criptată și salvată;

```
public function changePassword(Request $request): JsonResponse
{
    if (Auth::check()) {
        $user = User::findOrFail(Auth::user()->id);
    } else {
        return response()->json( data: 'User-ul nu este autentificat', status: Response::HTTP_UNAUTHORIZED);
    }

    if (!empty($this->userService->checkPassword($request->input( key: 'parola')))) {
        return response()->json($this->userService->checkPassword($request->input( key: 'parola')), status: Response::HTTP_UNAUTHORIZED);
    }

    $user->password = Hash::make($request->input( key: 'parola'));

    $user->save();
    return response()->json( data: "Parola schimbata!", status: Response::HTTP_OK);
}
```

Fig. 3.10 – Funcția *changePassword* din *UserController*

Se poate observa în Fig. 3.10 că, pentru criptarea parolei, am folosit o componentă Laravel *Hash*, astfel parola fiind mereu în siguranță, deoarece aceasta nu mai poate fi decriptată.

16. Logarea utilizatorilor: pentru realizarea acestei funcționalități, am folosit o dependență numită JWT. Pentru logare am creat funcția *login* din *UserController*, unde se verifică dacă datele introduse sunt corecte, iar dacă sunt, JWT generează un token distinct fiecărui utilizator, care este folosit apoi pentru verificarea autentificării. Pentru delogare, am creat funcția *logout*, unde se șterge token-ul utilizatorului;

```

public function login(Request $request): JsonResponse
{
    $validatedData = $request->validate([
        'email' => 'required|string',
        'password' => 'required|string'
    ]);

    $user = User::where('email', $validatedData['email'])->first();

    if (!$user) {
        return response()->json(['error' => 'User-ul nu exista!'], status: Response::HTTP_NOT_FOUND);
    }

    if (Hash::check($validatedData['password'], $user->password)) {
        JWTAuth::factory()->setTTL(60 * 24 * 7);
        $token = JWTAuth::fromUser($user);

        return response()->json([
            'user' => $user,
            'token' => $token,
        ]);
    }

    return response()->json(['error' => 'Unauthorized'], status: Response::HTTP_UNAUTHORIZED);
}

/**
 * @return JsonResponse
 */
public function logout(): JsonResponse
{
    JWTAuth::invalidate(JWTAuth::getToken());

    return response()->json(['data' => 'Te-ai delogat cu succes!'], status: Response::HTTP_OK);
}

```

Fig. 3.11 – Funcțiile login și logout din UserController

Se poate observa în Fig. 3.11 cum parola utilizatorului poate fi verificată doar prin intermediul Hash. De asemenea, un alt lucru important îl reprezintă faptul că token-ul generat este valabil doar pentru 7 zile (60 minute · 24 ore · 7 zile).

17. Resetarea parolei pierdute: dacă un utilizator își uită parola, acesta o poate schimba, accesând componenta „Ți-ai uitat parola?”. Aceasta va deschide un formular în care utilizatorul scrie adresa de e-mail cu care a fost creat contul, urmând să primească un e-mail care conține un link de resetare a parolei. Odată accesat, acesta va fi redirecționat către un formular în care trebuie să introducă o parolă conform standardelor impuse și să o confirme. Acest lucru este realizat prin intermediul funcțiilor *sendPasswordRequest*, care adaugă o intrare nouă în tabelul *password_resets* pentru monitorizarea procesului, generează un link de recuperare și trimite e-mailul folosind un șablon, *resetPasswordRedirect*, care preia datele din *password_reset* și redirecționează utilizatorul către formularul de recuperare și *resetPassword*, care verifică parola nouă. În cazul în care aceasta respectă toate cerințele, se criptează și salvează în tabel, iar statusul din *password_resets* se modifică în „verified”;

18. Redirecționarea în taburile de profil: dacă un utilizator dorește să acceseze un anumit tab din pagina de profil, a fost necesară crearea unei funcții care să preia numele tab-ului din URL și să-l transmită mai departe în frontend. Această funcție este denumită *getProfileRef* și se găsește în UserController.

3.2.2. Service

Aceste clase sunt folosite pentru a implementa funcții ce vor fi folosite în contextul a mai multe fișiere și pentru a reduce complexitatea funcțiilor din Controller, prin mutarea și gruparea unor porțiuni de cod care au ca scop obținerea unui rezultat. În contextul Ink&Paper, sunt implementate funcții folosite pentru organizarea datelor primite într-un model transmis și verificarea parolei.

Pentru serviciile care au ca scop doar organizarea datelor, acestea primesc ca parametrii proprietățile ce urmează a fi organizate și o instanță de model. Scopul funcției este de a verifica ce proprietăți există și a le atribui parametrilor din model.

```
public function setProperties(CarteComanda $cartecomanda, array $properties): void
{
    if (!empty($properties['id_carte'])) {
        $cartecomanda->id_carte = $properties['id_carte'];
    }

    if (!empty($properties['id_comanda'])) {
        $cartecomanda->id_comanda = $properties['id_comanda'];
    }

    if (!empty($properties['cantitate'])) {
        $cartecomanda->cantitate = $properties['cantitate'];
    }

    if (!empty($properties['subtotal'])) {
        $cartecomanda->subtotal = $properties['subtotal'];
    }
}
```

Fig. 3.12 – Funcția setProperties din CarteComandaService

După cum se poate observa în Fig. 3.12, se verifică anumite proprietăți din matricea „\$properties” și se atribuie modelului, acolo unde sunt disponibile.

Pentru serviciile care au ca scop verificarea parolei, acestea primesc ca parametru o parolă, urmând ca aceasta să fie verificată după mai multe criterii, printre care: numărul de caractere să fie minim 8, minim o majusculă, minim o minusculă, minim o cifră și minim un caracter special. La început se inițializează o matrice în care se vor stoca erorile apărute. În cazul în care nu apar erori, se va transmite o matrice goală.

```

/**
 * @param $password
 * @return array
 */
6 usages
public function checkPassword($password): array
{
    $passwordErrors = [];

    if (strlen($password) < 8) {
        $passwordErrors[] = 'Parola trebuie sa aiba cel putin 8 caractere';
    }

    if (!preg_match('/[A-Z]/', $password)) {
        $passwordErrors[] = 'Parola trebuie sa contina cel putin o majuscula';
    }

    if (!preg_match('/[a-z]/', $password)) {
        $passwordErrors[] = 'Parola trebuie sa contina cel putin o minuscula';
    }

    if (!preg_match('/\d/', $password)) {
        $passwordErrors[] = 'Parola trebuie sa contina cel putin o cifra';
    }

    if (!preg_match('/[!@#%&*()\-_=+{};:,<.>]/', $password)) {
        $passwordErrors[] = 'Parola trebuie sa contina cel putin un caracter special';
    }

    return $passwordErrors;
}

```

Fig. 3.13 – Funcția checkPassword din UserService

Se poate observa în Fig. 3.13 cum mesajele de eroare sunt compuse în funcție de criteriul care nu este îndeplinit și faptul că acestea se adună, adică în final vom avea o listă cu mai multe erori, dacă nu sunt respectate mai multe criterii.

3.2.3. Mail

Această componentă este folosită pentru a transmite e-mailuri cu ajutorul unor șabloane. Datele sunt transmise din controller către aceste clase cu anumiți parametri care urmează să fie organizați și transmiși către un șablon de e-mail. În cadrul proiectului sunt implementate patru șabloane: e-mail pentru confirmarea contului, e-mail pentru resetarea parolei, e-mail pentru confirmarea comenzii și e-mail pentru formularul de contact.

```

class AutenticareEmail extends Mailable
{
    use Queueable, SerializesModels;

    protected User $user;

    protected string $url;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    2 usages
    public function __construct(User $user, string $url)
    {
        $this->user = $user;
        $this->url = $url;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->subject( subject: 'Creare cont Ink&Paper')
            ->markdown( view: 'emails.welcome', [
                'user' => $this->user,
                'url' => $this->url
            ]);
    }
}

```

Fig. 3.14 – Clasa AutenticareEmail

Conform Fig. 3.14, clasa de Mail *AutenticareEmail* este construită folosind o instanță de model *User* și un string *url*, reprezentând datele userului căruia i se transmite e-mailul și un link cu ajutorul căruia se poate verifica contul. Acei doi parametri sunt transmiși mai departe către șablonul „welcome”.

3.3. Implementarea interfeței grafice

Pentru implementarea interfeței grafice am folosit Vue.js pentru crearea de componente și Laravel Blades pentru structurarea paginii. Printre componentele Vue.js implementate, se numără:

1. BookCarousel.vue: această componentă reprezintă o listă de cărți afișate sub formă de carusel. Implementarea acestuia a fost realizată folosind o librărie *vue-carousel*, alături de Bootstrap. Aceasta primește date de la alte componente (lista de cărți ce urmează să fie

afișată). Fiecare card, reprezentând o carte, prezintă trei funcționalități: adăugare la favorite, adăugare în coș și redirecționare către pagina cărții, dacă se apasă imaginea sau titlul;

2. BookDetails.vue: reprezintă prima parte din pagina unei cărți în care se găsesc imaginea cărții, datele acesteia și butoanele de adăugare la favorite și adăugare în coș;
3. BookFilter.vue: prin intermediul acestei componente, utilizatorul poate selecta unul sau mai multe filtre pentru afișarea cărților sau să le ordoneze, după preferințe. Această componentă are implementate mai multe funcții prin care comunică cu partea de backend, cum ar fi preluarea de date ce urmează să populeze filtrele și trimiterea filtrelor pentru a obține rezultatele dorite. De asemenea, această componentă comunică și cu alte componente, aceasta trimițând mai departe lista de cărți filtrată sau ordonată;
4. BookList.vue: această componentă este asemănătoare cu BookCarousel.vue, prezentând același comportament. Diferența între cele două o constituie modul de aranjare a cardurilor. În cazul componentei BookList.vue, cărțile sunt aranjate sub formă tabelară;
5. BookPage.vue: aceasta are rolul de a lega mai multe componente între ele, pentru a facilita transmiterea datelor între acestea. Ca și componente care se regăsesc în BookPage.vue, se numără BookDetails.vue, BookCarousel.vue și Review.vue;
6. Books.vue: asemeni BookPage.vue, această componentă are rolul doar de a face posibilă transmiterea datelor între BookList.vue (care primește lista de cărți) și BookFilter.vue (care trimite lista de cărți);

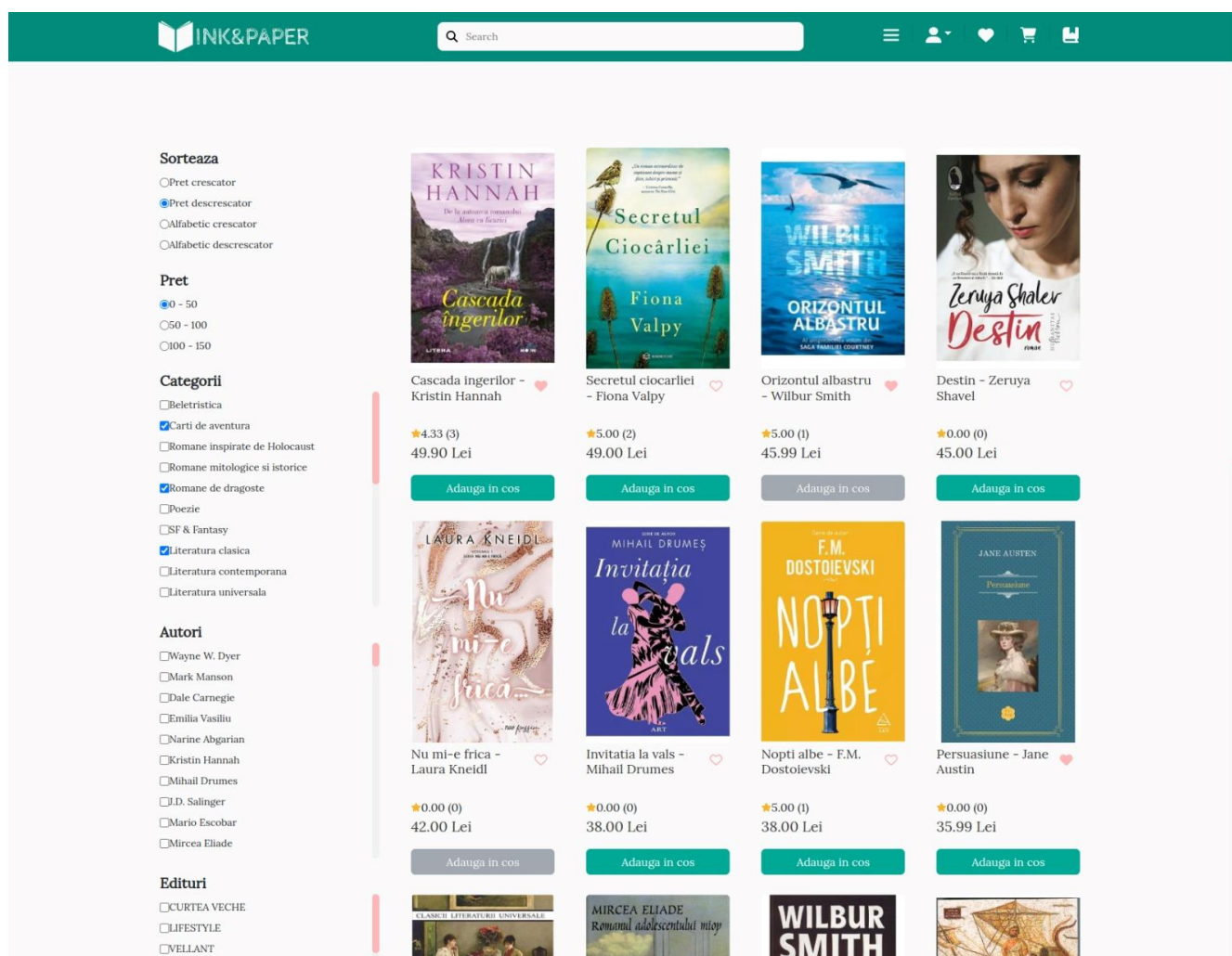


Fig. 3.15 – Pagina de căutare

7. ContactForm.vue: reprezintă formularul de contact disponibil pe pagina principală, acesta este o componentă simplă care trimite datele din formular prin intermediul unei rute către backend, unde urmează să fie procesate;
8. Cos.vue: componenta folosită în pagina în care se poate gestiona coșul și trimite comanda. Aceasta prezintă mai multe funcționalități, precum preluarea datelor utilizatorului în cazul în care acesta este autentificat, pentru autocompletarea formularului cu scopul trimiterii comenzii, preluarea produselor din coș și afișarea acestora, împreună cu calcularea prețului total pentru fiecare produs și salvarea cantității în coș în cazul modificării acesteia, calculul subtotalului și a totalului (subtotal și costuri de transport), ștergerea unui produs din coș și trimiterea unei comenzi;

INK&PAPER

Q

Search

</

9. `DonationForm.vue`: reprezintă componenta folosită pentru formularul de donare. Această componentă prezintă un formular în trei pași, realizat prin intermediul unor funcții JavaScript și trimite către backend datele pe care utilizatorul le-a completat în formular;
10. `Login.vue`: este folosit pentru formularul de logare. Aici se afișează erori în cazul în care datele pe care utilizatorul le introduce nu sunt corecte, redirecționează utilizatorul către formularul de creare al contului sau deschide formularul pentru parolă pierdută;
11. `MainPage.vue`: asemănător `Books.vue` și `BookPage.vue`, doar că în cadrul acestei componente se preiau listele cu cele mai noi cărți și cărțile cu promoții. Acestea sunt transmise către `BookCarousel.vue`. De asemenea, aici este accesat `ContactForm.vue`;
12. `MyFavorites.vue`: reprezintă una dintre componentele apelate în pagina de profil. Aceasta preia cărțile din lista de favorite ale utilizatorului și le transmite mai departe către o componentă `BookList.vue`;

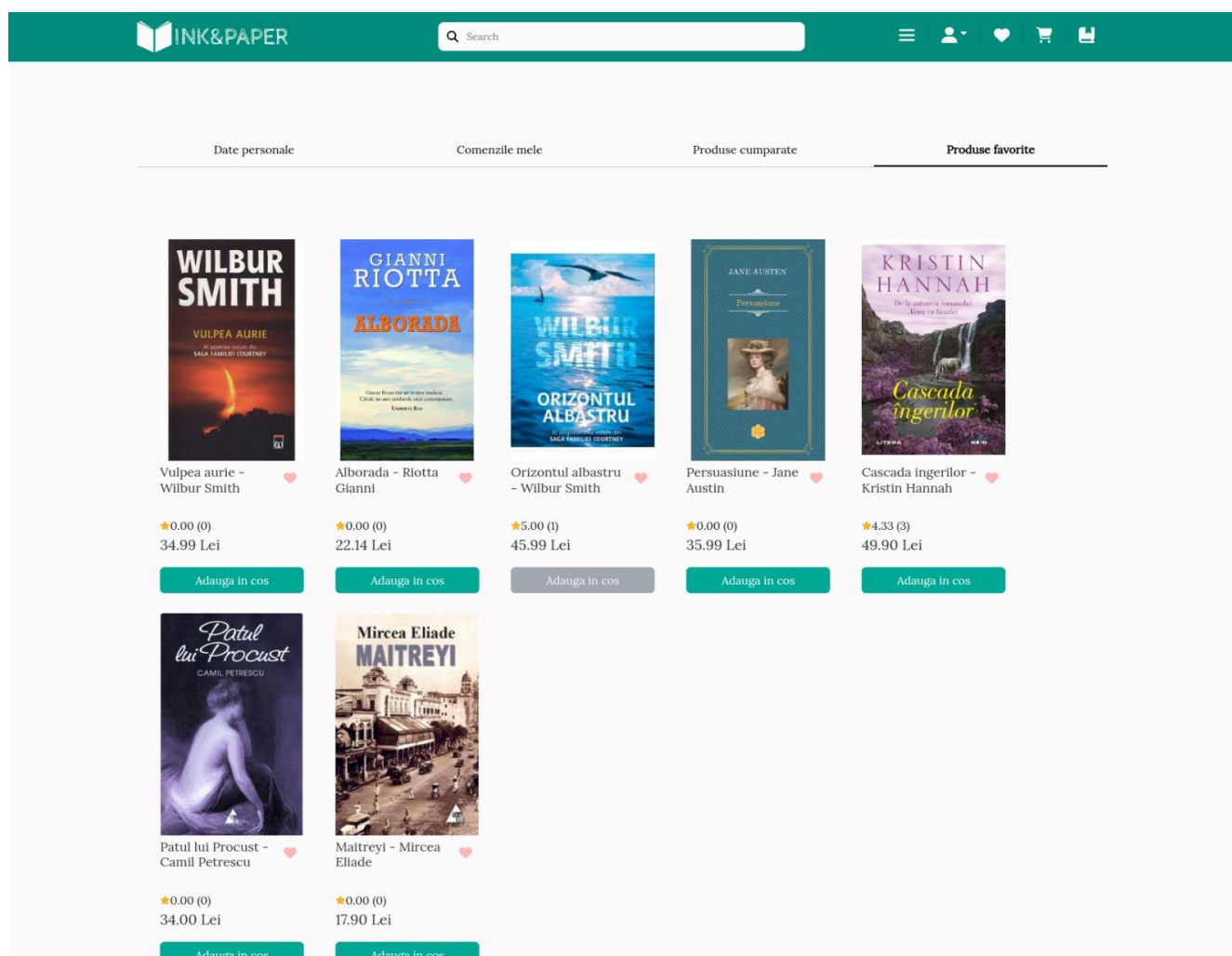


Fig. 3.17 – Pagină favorite

13. MyOrders.vue: este o componentă apelată în pagina de profil în care sunt afișate, sub formă de listă, comenzile utilizatorului. În cadrul acestora se pot vedea cărțile din fiecare comandă și se pot returna;
14. MyProducts: altă componentă apelată în pagina de profil care conține lista cărților achiziționate de utilizator. Aceasta preia lista prin intermediul unei rute și le transmite către o componentă BookList.vue;
15. MyProfile: componenta principală din pagina de profil în care utilizatorul își poate vedea și edita datele personale;

The screenshot shows the 'Date personale' (Personal Data) tab of a user profile on the INK&PAPER website. The form contains the following information:

Field	Value
Nume	Popeasca
Prenume	Maria
Email	popescu_maria01@gmail.com
Numar Telefon	0744055645
Adresa	Bd. Iuliu Maniu, Nr. 54
Localitate	Bucuresti
Judet	Bucuresti

Buttons visible: Salveaza, Schimba parola, Deschide formular.

Fig. 3.18 – Pagină profil

16. `Navigation.vue`: componenta folosită în toate paginile, reprezentând bara de navigare din partea de sus a paginii. Aceasta are încorporată bara de căutare, unde utilizatorii pot obține ca rezultat cărțile cu numele, autorul sau ISBN-ul asemănătoare cu textul introdus de ei, care va face redirectionarea către pagina de căutare. De asemenea, apăsând logo-ul din partea stângă, utilizatorul va fi redirectionat către pagina principală. În partea dreaptă, sunt puse la dispoziție scurtături pentru ca utilizatorul să poată naviga cu ușurință către paginile de favorite (în cazul în care utilizatorul nu este autentificat, acesta va fi redirectionat către pagina de autentificare), pagina de coș, formularul de donare, pagina de profil și logout în cazul în care este autentificat, sau paginile de login sau creare cont, în cazul în care nu este logat;
17. `Profile.vue`: reprezintă, asemănător `Books.vue`, o componentă în care se accesează alte componente, în funcție de tabul pe care utilizatorul dorește să-l afișeze din pagina de profil. Aici sunt accesate `MyFavorites.vue`, `MyOrders.vue`, `MyProducts.vue` și `MyProfile.vue`. De asemenea, această componentă primește codul tabului ce urmează să fie afișat;
18. `ProfileNavigation.vue`: componenta folosită pentru navigarea pe tab-uri în pagina de profil. Aceasta primește, de asemenea, codul tab-ului activ atunci când pagina de profil este accesată prin intermediul unei rute către un tab specific. Atunci când utilizatorul schimbă un tab, acesta va trimite către profil codul tabului nou accesat;
19. `ResetPassword.vue`: această componentă este folosită în cadrul paginii de resetare a parolei, reprezentând un formular în care utilizatorul introduce parola nouă și confirmarea acesteia. Ca și în restul formularelor, utilizatorul va primi mesaje de eroare în cazul în care parolele nu sunt identice sau dacă nu respectă normele impuse. Această componentă apelează o rută pentru a trimite datele completate de utilizator pentru prelucrare în backend;
20. `Review.vue`: reprezintă componenta din pagina unei cărți în care utilizatorul poate vedea recenziile și notele cărții, acordate de alți utilizatori. Aici se poate regăsi o diagramă în care se pot vedea numărul de recenzii în funcție de notele acordate, media generală și

recenziile fiecărui utilizator. Dacă este autentificat, utilizatorul poate, de asemenea, să scrie o recenzie;

21. `SignUp.vue`: folosit pentru formularul de creare a contului. Aici se fac verificări pentru ca fiecare câmp din formular să fie completat și să respecte normele impuse;

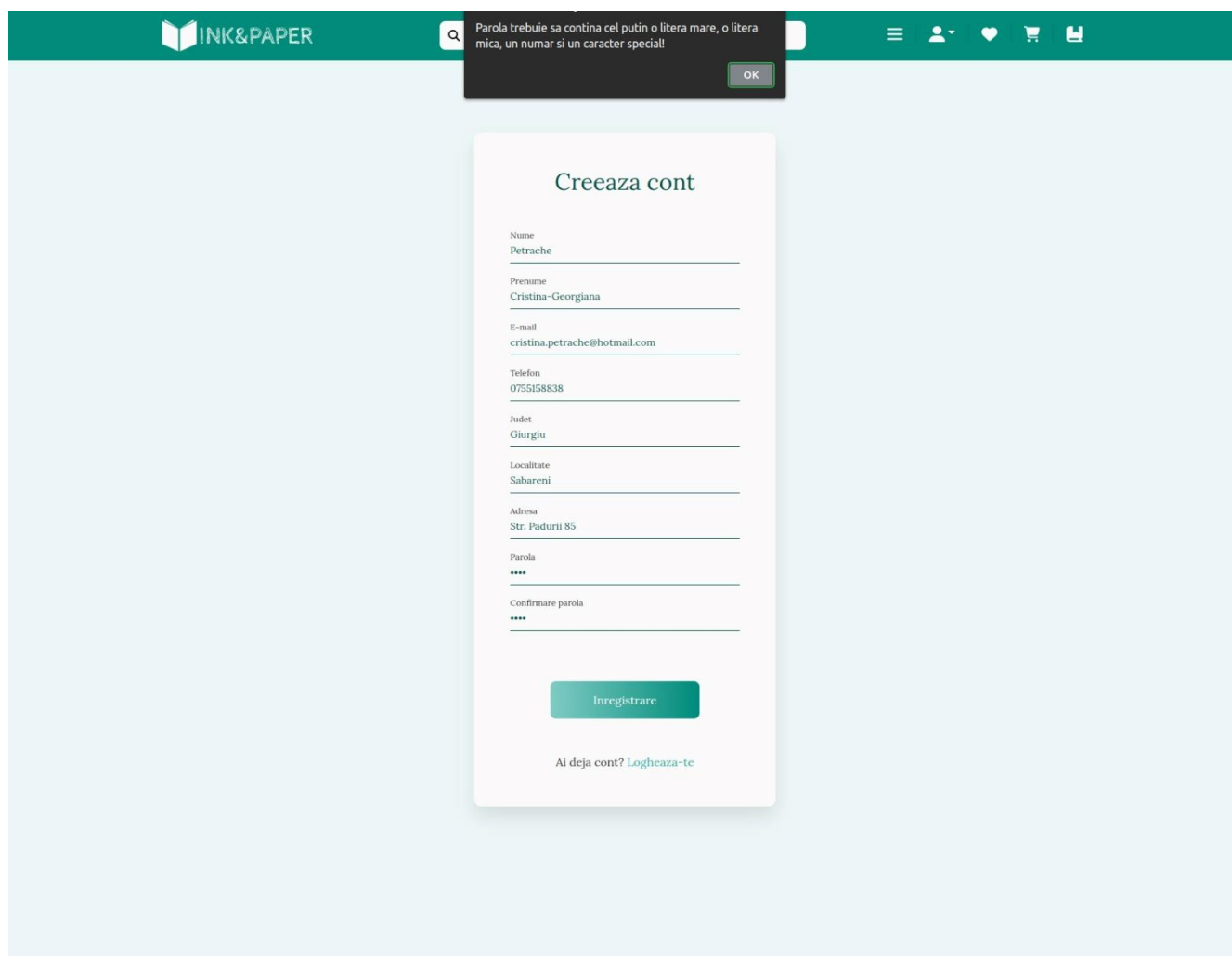


Fig. 3.19 – Formular de înregistrare

22. `CarteTable.vue`, `CategorieTable.vue`, `ComandaTable.vue`, `DonatieTable.vue`, `ReviewTable.vue`, `UserTable.vue`: folosite pentru a gestiona informațiile din baza de date.

Pentru verificarea autentificării utilizatorului și apelarea rutelor protejate, s-a folosit Vuex pentru stocarea token-ului primit la autentificare și transmiterea acestuia ca parametru de configurare în rutele protejate. Atunci când un utilizator se deloghează, cheia este ștearsă din stocarea locală.

O componenta Vue.js poate fi compusă din mai multe funcționalități, în funcție de necesități:

- **template**: reprezintă partea în care se scrie cod HTML, alături de CSS, după nevoie. Prin intermediul claselor se pot folosi componente Bootstrap. Vue.js pune la dispoziție mai multe instrumente care facilitează scrierea de cod simplu și ordonat, alături de funcționalități puternice;

```

<template>
  <div>
    <h4 class="titlu"><strong>{{ this.title }}</strong></h4>
    <carousel :per-page="5" :navigation-enabled="true">
      <slide v-for="book in books" :key="book.id">
        <div class="card-wrapper">
          <div ref="card" class="card mb-2">
            
            <span v-if="isDiscounted(book)" class="position-absolute top-0 start-100 translate-middle badge rounded-pill bg-danger" style="font-size: 0.8em; font-weight: normal; padding: 2px 5px; margin-left: 10px;">
              {{ '-' + mappedDiscountedBooks[book.id] + '%' }}
            </span>
            <div class="card-body">
              <div class="card">
                <div class="card-body d-flex">
                  <h5 class="flex-grow-1 card-title" @click="redirectBook(book)">{{ book.titlu }} - {{ book.autor }}</h5>
                  <div style="margin-left: 10px;">
                    
                    
                  </div>
                </div>
              </div>
            </div>
            <div class="d-flex">
              
              <p class="card-review">{{ book.rating }} ({{ book.nr_reviewuri }})</p>
            </div>
            <p v-if="!isDiscounted(book)" class="card-text card-price">{{ book.pret.toFixed(2) }} Lei</p>
            <p v-else class="card-text card-price">{{ (book.pret.toFixed(2) - ((mappedDiscountedBooks[book.id] * book.pret) / 100)).toFixed(2) }} Lei</p>
            <a v-if="book.cantitate !== 0" @click="addToCart(book)" class="btn btn-primary">Adauga in cos</a>
            <a v-else class="btn btn-primary disabled" style="background-color: #6c757d; color: #fff;">Adauga in cos</a>
          </div>
        </slide>
      </carousel>
    </div>
  </template>

```

Fig. 3.20 – Template din BookCarousel.Vue

Așa cum se poate observa în Fig. 3.20, în cadrul unui template am folosit cod HTML, alături de componente din librării JavaScript (carousel – componenta din vue-carousel), clase de Bootstrap, se apelează metode din script, se folosesc date din variabile, se folosesc stilizări CSS și se apelează clase personalizate, se folosesc instrumente ale Vue.js (v-for, v-if, v-model etc).

- script – import: în cadrul componentelor Vue.js, se pot importa librării sau alte componente Vue.js pentru a putea fi folosite în cadrul template-ului;

```

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
import { Carousel, Slide } from 'vue-carousel';
const csrfToken = document.querySelector( selectors: 'meta[name="csrf-token"]' ).content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};

```

Fig. 3.21 – Import din BookCarousel.Vue

În Fig. 3.21 se poate observa faptul că se importă componente de Vue.js (Carousel, Slide), funcționalități din Vuex, în mod specific funcția cu care se verifică dacă utilizatorul este autentificat, axios, care este utilizat pentru conexiunea cu partea de backend prin apelarea de rute și csrfToken.

- script – export: în această secțiune sunt prezente majoritatea funcționalităților componente. În prima parte se realizează listarea componentelor ce urmează să fie folosite în cadrul template-ului, dar și variabilele primite de la alte componente;

```
export default {
  name: 'BookCarousel',
  components: {
    Carousel,
    Slide
  },
  props: ['books', 'title'],
```

Fig. 3.22 – Numele, componentele și variabilele transmise din BookCarousel.vue

Așa cum se poate observa, componenta BookCarousel.vue folosește componentele Carousel și Slide care au fost importate precedent și primește variabilele books și title, acestea reprezentând lista de cărți și titlul ce urmează să fie afișate în secțiune.

- data(): aici se definesc variabilele globale ce urmează să fie folosite în cadrul componente. Aceste variabile pot fi folosite în cadrul funcțiilor, stilizărilor, pot fi afișate și lista poate continua;

```
data() {
  return {
    favorite: [],
    discountedBooks: [],
    mappedDiscountedBooks: [],
  }
},
```

Fig. 3.23 – data() din BookCarousel.vue

Conform Fig. 3.23 , în cadrul componente BookCarousel.vue sunt inițializate 3 variabile globale, reprezentând 3 matrici ce vor fi folosite pentru prelucrarea datelor.

- computed: aici se definesc proprietăți noi care se calculează într-un mod dinamic. Aceste proprietăți se calculează pe baza altor proprietăți existente. Practic, aici se definesc funcții care vor returna valori în funcție de alte valori sau proprietăți deja existente în componentă. Una dintre proprietățile acestui instrument este că valorile sunt recalulate la modificarea valorilor deja existente;

```
computed: {
  ...mapGetters(['isAuthenticated']),
},
```

Fig. 3.24 – computed din BookCarousel.vue

După cum reiese din Fig. 3.24, în cadrul componentei BookCarousel.vue se folosește funcția importată din Vuex pentru preluarea token-ului de autentificare al utilizatorului.

- `mounted()`: această opțiune este apelată în momentul în care componenta este montată în DOM;

```
mounted() {  
  this.getFavorite();  
  this.getDiscounted();  
},
```

Fig. 3.25 – `mounted()` din BookCarousel.vue

În componenta BookCarousel.vue, în cadrul opțiunii `mounted()` se apelează funcțiile pentru preluarea cărților pe care utilizatorul le are în lista de favorite, prin intermediul funcției `getFavorite()` și preluarea listei de cărți care au reduceri, prin intermediul funcției `getDiscounted()`, după cum reiese din Fig. 3.25. Astfel, atunci când componenta este încărcată în DOM, acele funcții vor fi apelate pentru preluarea datelor necesare.

- `methods`: aici se implementează funcțiile necesare în cadrul componentei. Aceste funcții pot avea diverse scopuri, de la preluarea de date, la apelarea de rute, manipularea variabilelor globale, redirecționare etc;

```
methods: {  
  addFavorite(carte) {  
    if (this.isAuthenticated === null) {  
      window.location.href = '/autentificare';  
    }  
    axios.post( url: '/favorite',  
      data: {  
        id_carte: carte.id,  
      },  
      config: {  
        headers: {  
          'Authorization': 'Bearer ' + this.isAuthenticated  
        }  
      })  
    .then(response => {  
      this.getFavorite();  
    })  
    .catch(error => {  
      console.log(error);  
    });  
  },  
},
```

Fig. 3.26 – Funcția `addFavorite` din BookCarousel.vue

În Fig. 3.26 este prezentată implementarea unei funcții în cadrul `methods`. Funcția `addFavorite` primește ca parametru un obiect de tip `carte`, apoi se verifică ca userul să fie autentificat, în caz contrar fiind redirecționat către pagina de autentificare, și se trimite ID-ul cărții prin intermediul rutei `/favorite` pentru a fi salvată în lista de favorite a utilizatorului curent. În cazul în care operațiunea

se execută cu succes, se va actualiza lista produselor favorite din componentă prin apelarea altei metode, *getFavorite()*, iar dacă apar erori pe parcursul executării, se va afișa în linia de comandă un mesaj de eroare.

- **watch**: această opțiune urmărește modificarea unei proprietăți și execută anumite acțiuni, în funcție de acele modificări;

```
watch: {  
  books: {  
    immediate: true,  
    handler() {  
      this.getCardWidth();  
    }  
  }  
},
```

Fig. 3.27 – watch din BookList.vue

Asa cum se poate observa din Fig. 3.27, această opțiune urmărește colecția de cărți și modifică lățimea fiecărui card de carte, în funcție de numărul lor.

- **style**: în cadrul secțiunii de style se pot personaliza clase de HTML sau crea clase noi, în vederea obținerii unor proprietăți conforme cu design-ul dorit.


```

<style scoped>
@font-face {
  font-family: 'Lora';
  src: url('/Fonts/static/Lora-Regular.ttf');
}

.book-list {
  background-color: #FAFAFA;
  padding-top: 20px;
}

.card-wrapper {
  width: 200px;
  margin-right: 5px;
  margin-left: 5px;
  margin-bottom: 10px;
  display: inline-block;
  vertical-align: top;
}

.row {
  display: flex;
  flex-wrap: wrap;
}

.col {
  flex: 0 0 200px;
  margin-right: 5px;
  margin-left: 5px;
  margin-bottom: 10px;
}

```

Fig. 3.28 – Secțiunea style din BookList.vue

În Fig. 3.28 se poate observa că se introduce un stil de font nou, în cadrul componentei, care urmează să fie folosit în cadrul unor clase, se creează componente noi, personalizate și se modifică clase de Bootstrap existente.

Paginile sunt create prin intermediul Laravel Blade. Printre acestea, se numără: pagină-autentificare, pagină-carte, pagină-căutare, pagină-coș, pagină-donare, pagină-favorite, pagină-înregistrare, pagină-principală, pagină-profil, pagină-resetare, pagină-admin. Toate paginile prezintă o structură asemănătoare.

```

<!DOCTYPE html>
<meta name="csrf-token" content="{{ csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha3
</head>

```

Fig. 3.29 – Header pagina-principală

Din Fig. 3.29 putem observa faptul că în antetul fișierului blade se importă componentele necesare utilizării Bootstrap, se include un token CSRF în cadrul paginii și se modifică titlul paginii în „Ink&Paper”.

```
<body>
<div id="app">
  <navigation class="padding-main"></navigation>
  <main-page class="margin-main"></main-page>
</div>
</body>
```

Fig. 3.30 – Body pagina-principală

Așa cum reiese din Fig. 3.30, în corpul fișierului se include fișierul de configurare pentru componentele Vue.js și se apelează două dintre acestea: componenta de navigare, care este apelată cu o clasă personalizată pentru padding și componenta main-page, care este apelată cu o clasă personalizată pentru margine.

```
<style>
  body{
    padding: 0;
    font-family: 'Lora', serif;
    background: #FAFAFA;
  }
  .margin-main {
    margin-left: calc(50% - 645px);
    margin-right: calc(50% - 645px);
  }
  .padding-main {
    padding-left: calc(50% - 645px);
    padding-right: calc(50% - 655px);
  }

  ::-webkit-scrollbar-track {
    background-color: #F0F1F2;
  }

  ::-webkit-scrollbar-thumb {
    background-color: #00A896;
    border: 3px solid #00A896;
  }

  ::-webkit-scrollbar-thumb:hover {
    background-color: #00A896;
  }

  ::-webkit-scrollbar-corner {
    background-color: #00A896;
  }

  ::-webkit-scrollbar {
    width: 10px;
    height: 10px;
  }
</style>
```

Fig. 3.31 – Style din pagina-principală

Din Fig. 3.31, observăm că se introduce în cadrul paginii un stil de font nou, care va fi folosit implicit în toate componentele existente. Apoi sunt create și personalizate clasele pentru margine și padding, acestea fiind calculate dinamic, astfel pagina va răspunde redimensionărilor. În final, sunt personalizate clasele specifice scrollbar-ului, pentru a obține un design plăcut.

Concluzii

Sumarizând, resursele tehnice pe care le folosim pentru implementarea proiectului sunt reprezentate de:

- HTML, CSS, Bootstrap și Vue.js pe partea de frontend, pentru a crea interfața platformei;
- Laravel pe partea de backend, pentru a realiza logica din spatele acesteia: cu ajutorul migrărilor am creat schema bazei de date, iar cu ajutorul Eloquent ORM am creat modele pentru fiecare tabel din baza de date și am stabilit relațiile dintre ele, în controllere am creat rute, care fac legătura între backend și frontend și care gestionează cererile utilizatorilor, am folosit funcționalitățile pe care le are încorporate de autentificare, resetare a parolei și trimitere a e-mailurilor și am creat funcții care gestionează logica interacțiunilor utilizatorului cu platforma;
- MySQL - pentru a crea schema bazei de date cu tabelele aferente și relațiile dintre acestea, dar și interogări pentru a gestiona datele;
- REST API - pentru a lega interfața grafică a aplicației de informațiile din baza de date, cu ajutorul protocolului HTTP;
- Apache HTTP Server - pentru a rula platforma pe un server local.

Cu ajutorul acestora, reușim să configurăm o librărie online funcțională care se remarcă prin următoarele trăsături inițiale:

- Utilizatorii creează un cont care le permite accesul în librăria virtuală;
- Înregistrarea necesită utilizarea unei adrese de e-mail care ajută nu doar la crearea contului, ci și pentru a recupera parola uitată sau pierdută, dar și pentru a primi confirmări în privința comenzilor date;
- În cadrul librăriei, utilizatorii pot atât să achiziționeze cărți, cât și să le doneze;
- În ceea ce privește cărțile achiziționate, utilizatorii au posibilitatea de a le returna, în anumite condiții;
- Utilizatorii pot evalua cărțile printr-o notă însoțită de un comentariu;
- Utilizatorii pot crea liste de favorite;
- Utilizatorii au la dispoziție un coș de cumpărături care centralizează selecțiile cumpărătorului înainte de finalizarea comenzii;
- Utilizatorii primesc recomandări în funcție de autor și categorie, în funcție de interesele manifestate;
- Utilizatorii au o pagină cu datele lor personale, pe care le pot actualiza/modifica, precum și un istoric al comenzilor;
- Platforma include o secțiune cu promoții care include cărțile care dispun de reduceri;
- Platforma include o secțiune cu noutăți pentru ultimele cărți adăugate spre vânzare.

Potențialele direcții de dezvoltare ulterioară implică:

- posibilitatea clienților de a închiria cărți;
- înștiințarea clienților, pe e-mail, cu privire la statusul în care se află comanda;
- adăugarea unei secțiuni în care utilizatorii pot vedea istoricul de navigare al cărților pe care le accesează;

- posibilitatea utilizatorilor de a se abona la newsletter pentru a primi înștiințări, pe e-mail, cu privire la reducerile și noutățile de pe platformă;
- stilizarea interfeței grafice a platformei pentru a oferi utilizatorilor o experiență îmbunătățită în cadrul său;
- dezvoltarea paginii administratorului platformei, care îi va permite să gestioneze mai multe date din baza de date.

Bibliografie

- [1] *What Is Web Development? – BrainStation*, <https://brainstation.io/career-guides/what-is-web-development>, accesat la data: 19.06.2023.
- [2] Filipova, O., Vilão, R. (2018). *Software Development from A to Z: A Deep Dive into all the Roles Involved in the Creation of Software*, Editura Apress, p. 22.
- [3] Filipova, O., Vilão, R. (2018). *Software Development from A to Z: A Deep Dive into all the Roles Involved in the Creation of Software*, Editura Apress, p. 20.
- [4] *How do APIs work? An in-depth guide | Tray.io*, <https://tray.io/blog/how-do-apis-work>, accesat la data: 16.06.2023.
- [5] Filipova, O., Vilão, R. (2018). *Software Development from A to Z: A Deep Dive into all the Roles Involved in the Creation of Software*, Editura Apress, p. 21.
- [6] *Full Stack Developer Job Profile - What Does A Full Stack Developer Do? | Le Wagon*, <https://www.lewagon.com/tech-jobs/web-development/full-stack-developer>, accesat la data: 19.06.2023.
- [7] *Librarie online - Carti, Jocuri, Muzica*, <https://www.libris.ro/>, accesat la data: 18.06.2023.
- [8] W. (n.d.). *Librarie Online - Compania de Librarii Bucuresti*, <https://www.clb.ro/>, accesat la data: 19.06.2023.
- [9] *Libraria Eminescu*, <https://librariaeminescu.ro/>, accesat la data: 19.06.2023.
- [10] *Is HTML a programming language? - 5 tips if you use markup languages - CodeBerry*. (2021, August 14), <https://codeberryschool.com/blog/en/html-a-programming-language/>, accesat la data: 31.05.2023.
- [11] *HTML*. (n.d.). Influencer Marketing Hub, <https://influencermarketinghub.com/glossary/html/>, accesat la data: 31.05.2023.
- [12] *What is HTML? Hypertext Markup Language Basics*. (2023, February 2). Intellipaat Blog. <https://intellipaat.com/blog/what-is-html/>, accesat la data: 02.06.2023.
- [13] *Limbaajul HTML*. (n.d.). <https://web.ceiti.md/lesson.php?id=1>, accesat la data: 02.06.2023
- [14] *HTML Introduction – GeeksforGeeks (2018, September 11)*, <https://www.geeksforgeeks.org/html-introduction/>, accesat la data: 02.06.2023.
- [15] *Ce este CSS si pentru ce se poate utiliza - Blog THC.ro*. (2021, August 5)., <https://www.thc.ro/blog/ce-este-css-si-pentru-ce-se-poate-utiliza/>, accesat la data: 04.06.2023.
- [16] S, C. (2021, September 17). *Ce este CSS (Cascading Style Sheets)? - Blog HostX.ro*, <https://blog.hostx.ro/utile/ce-este-css-cascading-style-sheets/>, accesat la data: 04.06.2023.
- [17] *Stiluri CSS*. (2010, January 1)., <https://web.ceiti.md/lesson.php?id=2#c2>, accesat la data: 04.06.2023.
- [18] Success, C. (2021, October 24). *What Is Bootstrap And Advantages of Bootstrap in Web Development*. Cyber Success, <https://www.cybersuccess.biz/advantages-of-bootstrap/>, accesat la data: 17.06.2023.
- [19] A., J. (2021, August 3). *What Is Bootstrap?* Hostinger Tutorials, <https://www.hostinger.com/tutorials/what-is-bootstrap/>, accesat la data: 17.06.2023.
- [20] *Introduction | Vue.js*. (n.d.), <https://vuejs.org/guide/introduction.html#what-is-vue>, accesat la data: 05.06.2023.

- [21] Brewster, C. (n.d.). *What Is Vue.js? The Pros and Cons of Vue.js in 2023 - Trio Developers.*, <https://www.trio.dev/blog/why-use-vue-js/>, accesat la data: 05.06.2023.
- [22] Tuama, D. (2022, February 21). *What is Vue.js? - Code Institute Global.*, <https://codeinstitute.net/global/blog/what-is-vue-js/>, accesat la data: 05.06.2023.
- [23] *A brief guide through Laravel.* (2022, January 12)., <https://mdevelopers.com/blog/a-brief-guide-through-laravel>, accesat la data: 05.06.2023.
- [24] *Laravel Tutorial: What It is, Framework, Features - Javatpoint.* (n.d.), <https://www.javatpoint.com/laravel>, accesat la data: 05.06.2023.
- [25] *What Is Laravel?* (n.d.). Built In. <https://builtin.com/software-engineering-perspectives/laravel>, accesat la data: 05.06.2023.
- [26] *The Laravel PHP Framework – Web App Construction for Everyone.* (2023, February 21). Kinsta®. <https://kinsta.com/knowledgebase/what-is-laravel/>, accesat la data: 05.06.2023.
- [27] *What is MySQL?* (n.d.). What Is MySQL? | Oracle., <https://www.oracle.com/mysql/what-is-mysql/>, accesat la data: 11.06.2023.
- [28] B., R. (2018, December 14). *What is MySQL: MySQL Explained For Beginners.* Hostinger Tutorials., <https://www.hostinger.com/tutorials/what-is-mysql>, accesat la data: 11.06.2023.
- [29] *What Is MySQL? A Beginner-Friendly Explanation.* (2022, April 11). Kinsta®, <https://kinsta.com/knowledgebase/what-is-mysql/>, accesat la data: 12.06.2023.
- [30] *What is an API? - Application Programming Interfaces Explained - AWS.* (n.d.). Amazon Web Services, Inc, <https://aws.amazon.com/what-is/api/>, accesat la data: 16.06.2023.
- [31] Giaquinto, R., Holcombe, J., & Phillips, M. (2022, February 15). *What is Apache and What Does it Do for Website Development?* GreenGeeks Blog. <https://www.greengeeks.com/blog/what-is-apache/>, accesat la data: 23.06.2023.
- [32] Singhal, P. (2022, October 17). *What Are the Advantages of HTML?- Scaler Topics*, <https://www.scaler.com/topics/advantages-of-html/>, accesat la data: 12.06.2023.
- [33] *Unstop - Competitions, Quizzes, Hackathons, Scholarships and Internships for Students and Corporates*, <https://unstop.com/blog/advantages-and-disadvantages-of-html>, accesat la data: 12.06.2023.
- [34] S. (2022, January 3). *11 Advantages and Disadvantages of CSS You Should Heed - Tech Quintal*, <https://www.techquintal.com/advantages-and-disadvantages-of-css/>, accesat la data: 13.06.2023.
- [35] Smith, D. (2022, December 24). *What is CSS, and what are its advantages and disadvantages? & NetworkUstad*, <https://networkustad.com/2022/12/24/what-is-css-and-what-are-its-advantages-and-disadvantages/>, accesat la data: 13.06.2023.
- [36] Sahu, P. (2021, October 12). *Advantages and Disadvantages of CSS Everyone Should Know – Motocms*, <https://www.motocms.com/blog/en/advantages-and-disadvantages-of-css/>, accesat la data: 13.06.2023.
- [37] *Unstop - Competitions, Quizzes, Hackathons, Scholarships and Internships for Students and Corporates.* (n.d.), <https://unstop.com/blog/advantages-and-disadvantages-of-css>, accesat la data: 13.06.2023.
- [38] *Bootstrap Tutorial.* (n.d.). GeeksforGeeks. <https://www.geeksforgeeks.org/bootstrap/>, accesat la data: 17.06.2023.

- [39] Smit, M. (2021, November 16). *Pros And Cons Of Using Vue.js for Web Development / Nordic APIs*, <https://nordicapis.com/pros-and-cons-of-using-vue-js-for-web-development/>, accesat la data: 13.06.2023.
- [40] *Advantages and Disadvantages of Using Vue.js - Webuters*. (2019, September 2), <https://www.webuters.com/advantages-and-disadvantages-of-using-vuejs>, accesat la data: 13.06.2023.
- [41] *Using Vue: Pros and Cons*. (n.d.), <https://thecodest.co/blog/pros-and-cons-of-vue/>, accesat la data: 13.06.2023.
- [42] *Advantages and disadvantages of Laravel Framework for web Development*. (n.d.). DDI Development, <https://ddi-dev.com/blog/programming/pros-and-cons-of-laravel-framework-for-web-app-development/>, accesat la data: 15.06.2023.
- [43] *Top 6 Advantages and Disadvantages of Laravel for web Development*, <https://dgaps.com/advantages-disadvantages-of-laravel-519>, accesat la data: 15.06.2023.
- [44] Technolabs, L. (2021, May 14). *Pros and Cons of Laravel PHP Framework 2021*. Medium. <https://latitudetechlabs.medium.com/pros-and-cons-of-laravel-php-framework-2021-e2027e22f2b8>, accesat la data: 15.06.2023.
- [45] *Advantages and Disadvantages of MySQL*. (n.d.). AspiringYouths, <https://aspiringyouths.com/advantages-disadvantages/mysql/>, accesat la data: 15.06.2023.
- [46] Munasingha, D. (2021, May 21). *Advantages and Disadvantages of using MySQL*. Medium, <https://diliru.medium.com/advantages-and-disadvantages-of-using-mysql-36f6ffce3fa3>, accesat la data: 15.06.2023.
- [47] *Advantages And Disadvantages Of MySQL* (2022, October 28). WatanLema, <https://watanlema.com/advantages-and-disadvantages-of-mysql/>, accesat la data: 15.06.2023.
- [48] Roomi, M. (2022, October 29). *5 Advantages and Disadvantages of MySQL / Limitations & Benefits of MySQL*. HitechWhizz - the Ultimate Tech Experience, <https://www.hitechwhizz.com/2022/10/5-advantages-and-disadvantages-limitations-benefits-of-mysql1.html>, accesat la data: 15.06.2023.
- [49] *The Advantages & Disadvantages of API Solutions / Healthie*, <https://www.gethealthie.com/blog/api-advantages-disadvantages>, accesat la data: 16.06.2023.
- [50] B., R. (2018, June 20). *What Is Apache? An In-Depth Overview of Apache Web Server*. Hostinger Tutorials, <https://www.hostinger.com/tutorials/what-is-apache>, accesat la data: 23.06.2023.
- [51] *PHP Code Editor and Debugger - Features / PhpStorm*. (n.d.). JetBrains, https://www.jetbrains.com/phpstorm/features/php_code_editor.html, accesat la data: 17.06.2023.
- [52] *Features and Screenshots – DataGrip*, JetBrains, <https://www.jetbrains.com/datagrip/features/>, accesat la data: 17.06.2023.

Anexe

Anexa 1 – Migrări

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function
(Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table-
>timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();

        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePasswordResetsTable extends
Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets',
function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')-
>nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {

```

```

        Schema::dropIfExists('password_resets');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateFailedJobsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('failed_jobs', function
(Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')-
>useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePersonalAccessTokensTable extends
Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('personal_access_tokens',
function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)-
>unique();
            $table->text('abilities')-
>nullable();
            $table->timestamp('last_used_at')-
>nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {

```

```

    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('personal_access_tokens')
;
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateClientTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('client', function
(Blueprint $table) {
            $table->id();
            $table->string('nume');
            $table->string('prenume');
            $table->string('email');
            $table->string('parola');
            $table->string('nr_telefon');
            $table->string('adresa');
            $table->string('oras');
            $table->string('judet');
            $table->string('favorite');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('client');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCarteTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('carte', function
(Blueprint $table) {
            $table->id();
            $table->string('titlu');
            $table->string('autor');
            $table->string('isbn');

```

```

            $table->string('editura');
            $table->float('pret',6,2);
            $table->integer('an_aparitie');
            $table->integer('nr_pg');
            $table->string('tip_coperta');
            $table->string('limba');
            $table->string('dimensiune');
            $table->integer('cantitate');
            $table->string('categorii');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('carte');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCarteComandaTable extends
Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('carte_comanda',
function (Blueprint $table) {
            $table->integer('id_carte');
            $table->integer('id_comanda');
            $table->integer('cantitate');
            $table->float('subtotal',6,2);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('carte_comanda');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateComandaTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('comanda', function
(Blueprint $table) {
            $table->id();

```

```

        $table->integer('id_client');
        $table->date('data_plasare');
        $table->date('data_livrare');
        $table->float('pret_comanda',6,2);
        $table->string('adresa_livrare');
        $table->string('oras_livrare');
        $table->string('judet_livrare');
        $table->string('status');
        $table->string('metoda_plata');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('comanda');
}
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateReviewTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('review', function
(Blueprint $table) {
            $table->id();
            $table->integer('id_client');
            $table->integer('id_carte');
            $table->string('comentariu');
            $table->float('rating',4,2);
            $table->date('data_review');
            $table->string('titlu');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('review');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCategorieTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categorie', function

```

```

(Blueprint $table) {
            $table->id();
            $table->string('nume');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categorie');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteTableAddImagineColumn extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->string('imagine')->
>after('autor');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->dropColumn('imagine');
        });
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterClientTableChangeFavoriteType extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('client', function
(Blueprint $table) {
            $table->json('favorite')->
>change();
        });
    }

    /**

```

```

        * Reverse the migrations.
        *
        * @return void
        */
        public function down()
        {
            Schema::table('client', function
(Blueprint $table) {
                $table->string('favorite')-
>change();
            });
        }
    }

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteTableChangeCategorieType
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->json('categorie')-
>change();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->string('categorie')-
>change();
        });
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterClientTableDropFavoriteColumn
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('client', function
(Blueprint $table) {
            $table->dropColumn('email');
            $table->dropColumn('parola');
            $table->dropColumn('favorite');
        });
    }

    /**
     * Reverse the migrations.
     *

```

```

        * @return void
        */
        public function down()
        {
            //
        }
    }

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterUsersTableDropNameColumn extends
Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function
(Blueprint $table) {
            $table->dropColumn('name');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateFavoriteTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('favorite', function
(Blueprint $table) {
            $table->integer('id_client');
            $table->integer('id_carte');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('favorite');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;

```

```

use Illuminate\Support\Facades\Schema;

class CreateCarteCategorieTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('carte_categorie',
function (Blueprint $table) {
            $table->integer('id_carte');
            $table->integer('id_categorie');
            $table->timestamps();
        });

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('carte_categorie');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateDonatiiTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('donatii', function
(Blueprint $table) {
            $table->id();
            $table->string('nume');
            $table->string('prenume');
            $table->string('email');
            $table->string('nr_telefon');
            $table->string('titlu');
            $table->string('autor');
            $table->string('isbn');
            $table->string('adresa_ridicare');
            $table->string('oras_ridicare');
            $table->string('judet_ridicare');
            $table->timestamps();
        });

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('donatii');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;

```

```

use Illuminate\Support\Facades\Schema;

class CreateReturnariTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('returnari', function
(Blueprint $table) {
            $table->id();
            $table->integer('id_carte');
            $table->integer('id_comanda');
            $table->integer('cantitate');
            $table->float('subtotal',6,2);
            $table->timestamps();
        });

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('returnari');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCarteCosTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('carte_cos', function
(Blueprint $table) {
            $table->integer('id_client');
            $table->integer('id_carte');
            $table->integer('cantitate');
            $table->float('subtotal',6,2);
            $table->timestamps();
        });

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('carte_cos');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteTableDropCategorieColumn
extends Migration
{
    /**

```

```

        * Run the migrations.
        *
        * @return void
        */
        public function up()
        {
            Schema::table('carte', function
(Blueprint $table) {
                $table->dropColumn('categorie');
            });
        }

        /**
         * Reverse the migrations.
         *
         * @return void
         */
        public function down()
        {
            //
        }
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterComandaTableAddTipDataPredareColumns
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('comanda', function
(Blueprint $table) {
            $table->string('tip')-
>after('id_client');
            $table->date('data_predare')-
>after('data_livrare')->default(null);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteTableAddCategorieIdColumn
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->string('categorie_id')-

```

```

>after('editura');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteTableDropCategorieIdColumn
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table-
>dropColumn('categorie_id');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCategorieTableAddParentIdColumn
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('categorie', function
(Blueprint $table) {
            $table->string('parent_id')-
>after('nume');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {

```



```

        //
    }
}

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddReviewDataToCarteTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->integer('nr_reviewuri')-
>default(0);
            $table->decimal('rating')-
>default(0.00);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('carte', function
(Blueprint $table) {
            //
        });
    }
}

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddClientIdColumnToUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function
(Blueprint $table) {
            $table->integer("client_id")-
>after("id")->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('users', function
(Blueprint $table) {
            //
        });
    }
}

```

```

}

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddUserIdColumnToClientTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('client', function
(Blueprint $table) {
            $table->integer("user_id")-
>after("id")->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('client', function
(Blueprint $table) {
            //
        });
    }
}

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateEmailVerificationTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('email_verification',
function (Blueprint $table) {
            $table->id();
            $table->integer('user_id');
            $table->string('token', 100)-
>unique();
            $table->string('status', 100)-
>default('pending');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('email_verification');
    }
}

```

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateDiscountsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('discounts', function
(Blueprint $table) {
            $table->id();
            $table->integer('id_carte');
            $table->integer('promotie');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('discounts');
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteCosTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte_cos', function
(Blueprint $table) {
            $table->integer('id_client')-
>nullable()->change();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteTable extends Migration
{
    /**
     * Run the migrations.
     *

```

```

     * @return void
     */
    public function up()
    {
        Schema::table('carte', function
(Blueprint $table) {
            $table->decimal('rating', 3, 2)-
>default(0.00)->change();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterPasswordResetsTable extends
Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('password_resets',
function (Blueprint $table) {
            $table->id();
            $table->string('status')-
>default('pending');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AlterCarteComandaTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte_comanda',
function (Blueprint $table) {
            $table->id();
        });
    }

    /**

```

```

        * Reverse the migrations.
        *
        * @return void
        */
        public function down()
        {
            //
        }
    }

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddStatusColumnToCarteComandaTable
extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('carte_comanda',
function (Blueprint $table) {
            $table->string('status')->
default('achizitionat')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('carte_comanda',
function (Blueprint $table) {
            //
        });
    }
}

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class DropUnusedTables extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::dropIfExists('personal_access_tokens')
;
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}

```

Anexa 2 – Model

```

<?php

namespace App\Models;

use DateTime;
use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;
use
Illuminate\Database\Eloquent\Relations\Belongs
ToMany;
use
Illuminate\Database\Eloquent\Relations\HasMany
;

/**
 * App\Models\Carte
 * @property int $id
 * @property string $titlu
 * @property string $autor
 * @property string $imagine
 * @property string $isbn
 * @property string $editura
 * @property float $pret
 * @property int $an_aparitie
 * @property int $nr_pg
 * @property string $tip_coperta
 * @property string $limba
 * @property string $dimensiune
 * @property string $cantitate
 * @property int $nr_reviewuri
 * @property float $rating
 */

class Carte extends Model
{
    use HasFactory;

    protected $table = 'carte';

    protected $columns = [
        'id' => [
            'label' => 'id'
        ],
        'titlu' => [
            'label' => 'titlu'
        ],
        'autor' => [
            'label' => 'autor'
        ],
        'imagine'=> [
            'label' => 'imagine'
        ],
        'isbn' => [
            'label' => 'isbn'
        ],
        'editura' => [
            'label' => 'editura'
        ],
        'pret' => [
            'label' => 'pret'
        ],
        'an_aparitie' => [
            'label' => 'an_aparitie'
        ],
        'nr_pg' => [
            'label' => 'nr_pg'
        ],
        'tip_coperta' => [
            'label' => 'tip_coperta'
        ],
        'limba' => [
            'label' => 'limba'
        ],
    ],

```

```

        'dimensiune' => [
            'label' => 'dimensiune'
        ],
        'cantitate' => [
            'label' => 'cantitate'
        ]
    ];

    protected $primaryKey = 'id';

    /**
     * @return HasMany
     */
    public function reviewuri(): HasMany
    {
        return $this->hasMany(
            Review::class,
            'id_carte',
            'id'
        );
    }

    /**
     * @return BelongsToMany
     */
    public function comenzi(): BelongsToMany
    {
        return $this->belongsToMany(
            Comanda::class,
            'carte_comanda',
            'id_carte',
            'id_comanda'
        );
    }

    /**
     * @return BelongsToMany
     */
    public function categorii(): BelongsToMany
    {
        return $this->belongsToMany(
            Categorie::class,
            'carte_categorie',
            'id_carte',
            'id_categorie'
        );
    }
}

<?php
namespace App\Models;

use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\CarteComanda
 * @property int $id_carte
 * @property int $id_comanda
 * @property int $cantitate
 * @property float $subtotal
 */

class CarteComanda extends Model
{
    use HasFactory;

    protected $table = 'carte_comanda';

    protected $columns = [
        'id_carte' => [
            'label' => 'id_carte'
        ],
        'id_comanda' => [
            'label' => 'id_comanda'
        ],
    ],

```

```

        'cantitate' => [
            'label' => 'cantitate'
        ],
        'subtotal' => [
            'label' => 'subtotal'
        ]
    ];

    protected $primaryKey = 'id';
}

<?php
namespace App\Models;

use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;
use
Illuminate\Database\Eloquent\Relations\HasMany
;

/**
 * App\Models\CarteCos
 * @property int $id_client
 * @property int $id_carte
 * @property int $cantitate
 * @property float $subtotal
 */

class CarteCos extends Model
{
    use HasFactory;

    protected $table = 'carte_cos';

    protected $columns = [
        'id_carte' => [
            'label' => 'id_carte'
        ],
        'id_client' => [
            'label' => 'id_client'
        ],
        'cantitate' => [
            'label' => 'cantitate'
        ],
        'subtotal' => [
            'label' => 'subtotal'
        ]
    ];

    protected $primaryKey = 'id_carte';

    public function carti(): HasMany
    {
        return $this->hasMany(
            Carte::class,
            'id',
            'id_carte'
        );
    }
}

<?php
namespace App\Models;

use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;
use
Illuminate\Database\Eloquent\Relations\Belongs
ToMany;

/**
 * App\Models\Categorie

```

```

* @property int $id
* @property string $nume
* @property int $parent_id
*/

class Categorie extends Model
{
    use HasFactory;

    protected $table = 'categorie';

    protected $columns = [
        'id' => [
            'label' => 'id'
        ],
        'nume' => [
            'label' => 'nume'
        ],
        'parent_id' => [
            'label' => 'parent_id'
        ]
    ];

    protected $primaryKey = 'id';

    /**
     * @return BelongsToMany
     */
    public function carti(): BelongsToMany
    {
        return $this->belongsToMany(
            Carte::class,
            'carte_categorie',
            'id_categorie',
            'id_carte'
        );
    }

    public function parent()
    {
        return $this->belongsTo(Categorie::class, 'parent_id');
    }

    public function children()
    {
        return $this->hasMany(Categorie::class, 'parent_id', 'id');
    }
}

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\Authenticatable;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Support\Facades\Auth;

/**
 * App\Models\Client
 * @property int $id
 * @property string $nume
 * @property string $prenume
 * @property string $nr_telefon
 * @property string $adresa
 * @property string $oras

```

```

* @property string $judet
*/

class Client extends Model
{
    use HasFactory;

    protected $table = 'client';

    protected $columns = [
        'id' => [
            'label' => 'id'
        ],
        'nume' => [
            'label' => 'nume'
        ],
        'prenume' => [
            'label' => 'prenume'
        ],
        'nr_telefon' => [
            'label' => 'nr_telefon'
        ],
        'adresa' => [
            'label' => 'adresa'
        ],
        'oras' => [
            'label' => 'oras'
        ],
        'judet' => [
            'label' => 'judet'
        ]
    ];

    protected $primaryKey = 'id';

    /**
     * @return HasMany
     */
    public function comenzi(): HasMany
    {
        return $this->hasMany(
            Comanda::class,
            'id_client',
            'id'
        );
    }

    /**
     * @return HasMany
     */
    public function reviewuri(): HasMany
    {
        return $this->hasMany(
            Review::class,
            'id_client',
            'id'
        );
    }

    /**
     * @return BelongsTo
     */
    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }

    /**
     * @return HasMany
     */
    public function cartiCos(): HasMany
    {
        return $this->hasMany(
            CarteCos::class,
            'id_client',
            'id'
        );
    }
}

```

```

/**
 * @return HasMany
 */
public function favorite(): HasMany
{
    return $this->HasMany(
        Favorit::class,
        'id_client',
        'id'
    );
}
}

<?php
namespace App\Models;

use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;
use
Illuminate\Database\Eloquent\Relations\Belongs
To;
use
Illuminate\Database\Eloquent\Relations\Belongs
ToMany;
use DateTime;

/**
 * App\Models\Comanda
 * @property int $id
 * @property int $id_client
 * @property string $tip
 * @property DateTime $data_plasare
 * @property DateTime $data_livrare
 * @property DateTime $data_predare
 * @property float $pret_comanda
 * @property string $adresa_livrare
 * @property string $oras_livrare
 * @property string $judet_livrare
 * @property string $status
 * @property string $metoda_plata
 */

class Comanda extends Model
{
    use HasFactory;

    protected $table = 'comanda';

    protected $columns = [
        'id' => [
            'label' => 'id'
        ],
        'id_client' => [
            'label' => 'id_client'
        ],
        'tip' => [
            'label' => 'tip'
        ],
        'data_plasare' => [
            'label' => 'data_plasare'
        ],
        'data_livrare' => [
            'label' => 'data_livrare'
        ],
        'data_predare' => [
            'label' => 'data_predare'
        ],
        'pret_comanda' => [
            'label' => 'pret_comanda'
        ],
        'adresa_livrare' => [
            'label' => 'adresa_livrare'
        ],
        'oras_livrare' => [
            'label' => 'oras_livrare'
        ],
    ],

```

```

        'judet_livrare' => [
            'label' => 'judet_livrare'
        ],
        'status' => [
            'label' => 'status'
        ],
        'metoda_plata' => [
            'label' => 'metoda_plata'
        ]
    ];

    protected $primaryKey = 'id';

    /**
     * @return BelongsToMany
     */
    public function carti(): BelongsToMany
    {
        return $this->belongsToMany(
            Carte::class,
            'carte_comanda',
            'id_comanda',
            'id_carte'
        );
    }
}

<?php
namespace App\Models;

use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;

/**
 * @property int $id
 * @property int $id_carte
 * @property int $promotie
 */
class Discount extends Model
{
    use HasFactory;

    protected $table = 'discounts';
} <?php

namespace App\Models;

use
Illuminate\Database\Eloquent\Factories\HasFact
ory;
use Illuminate\Database\Eloquent\Model;

/**
 * App\Models\Donatie
 * @property int $id
 * @property string $nume
 * @property string $prenume
 * @property string $nr_telefon
 * @property string $email
 * @property string $titlu
 * @property string $autor
 * @property string $isbn
 * @property string $adresa_ridicare
 * @property string $oras_ridicare
 * @property string $judet_ridicare
 */

class Donatie extends Model
{
    use HasFactory;

    protected $table = 'donatii';

    protected $columns = [
        'id' => [

```

```

        'label' => 'id'
    ],
    'nume' => [
        'label' => 'nume'
    ],
    'prenume' => [
        'label' => 'prenume'
    ],
    'nr_telefon' => [
        'label' => 'nr_telefon'
    ],
    'email' => [
        'label' => 'email'
    ],
    'titlu' => [
        'label' => 'titlu'
    ],
    'autor' => [
        'label' => 'autor'
    ],
    'isbn' => [
        'label' => 'isbn'
    ],
    'adresa_ridicare' => [
        'label' => 'adresa_ridicare'
    ],
    'oras_ridicare' => [
        'label' => 'oras_ridicare'
    ],
    'judet_ridicare' => [
        'label' => 'judet_ridicare'
    ]
];

protected $primaryKey = 'id';
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

/**
 * App\Models\Favorit
 * @property int $id_client
 * @property int $id_carte
 */

class Favorit extends Model
{
    use HasFactory;

    protected $table = 'favorite';

    protected $columns = [
        'id_carte' => [
            'label' => 'id_carte'
        ],
        'id_client' => [
            'label' => 'id_client'
        ]
    ];

    protected $primaryKey = 'id_carte';

    public function carti(): HasMany
    {
        return $this->hasMany(
            Carte::class,
            'id',
            'id_carte'
        );
    }
}

```

```

    }
}
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

/**
 * App\Models\Returnare
 * @property int $id
 * @property int $id_carte
 * @property int $id_comanda
 * @property int $cantitate
 * @property float $subtotal
 */

class Returnare extends Model
{
    use HasFactory;

    protected $table = 'returnari';

    protected $columns = [
        'id' => [
            'label' => 'id'
        ],
        'id_carte' => [
            'label' => 'id_carte'
        ],
        'id_comanda' => [
            'label' => 'id_comanda'
        ],
        'cantitate' => [
            'label' => 'cantitate'
        ],
        'subtotal' => [
            'label' => 'subtotal'
        ]
    ];

    protected $primaryKey = ['id'];

    public function cartiComanda(): HasMany
    {
        return $this->hasMany(
            CarteComanda::class
        );
    }
}
<?php

namespace App\Models;

use DateTime;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

/**
 * App\Models\Review
 * @property int $id
 * @property int $id_client
 * @property int $id_carte
 * @property string $comentariu
 * @property float $rating
 * @property DateTime $data_review
 */

```

```

* @property string $titlu
*/

class Review extends Model
{
    use HasFactory;

    protected $table = 'review';

    protected $columns = [
        'id' => [
            'label' => 'id'
        ],
        'id_client' => [
            'label' => 'id_client'
        ],
        'id_carte' => [
            'label' => 'id_carte'
        ],
        'comentariu' => [
            'label' => 'comentariu'
        ],
        'rating' => [
            'label' => 'rating'
        ],
        'data_review' => [
            'label' => 'data_review'
        ],
        'titlu' => [
            'label' => 'titlu'
        ]
    ];

    protected $primaryKey = 'id';

    /**
     * @return BelongsTo
     */
    public function client(): BelongsTo
    {
        return $this->belongsTo(
            Client::class,
            'id_client',
            'id'
        );
    }

    /**
     * @return BelongsTo
     */
    public function carte(): BelongsTo
    {
        return $this->belongsTo(
            Carte::class,
            'id_carte',
            'id'
        );
    }
}

<?php

namespace App\Models;

use DateTime;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Tymon\JWTAuth\Contracts\JWTSubject;

/**

```

```

* App\Models\User
* @property int $id
* @property string $email
* @property DateTime $email_verified_at
* @property string $password
* @property string $remember_token
*/

class User extends Authenticatable implements
MustVerifyEmail, JWTSubject
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass
     assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'email',
        'password',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function getJWTIdentifier()
    {
        return $this->getKey();
    }

    public function getJWTCustomClaims()
    {
        return [];
    }

    /**
     * @return BelongsTo
     */
    public function client(): BelongsTo
    {
        return $this->belongsTo(Client::class);
    }
}

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

/**
* @property int $id_carte
* @property int $id_categorie
*/
class CarteCategorie extends Model
{
    use HasFactory;

    protected $table='carte_categorie';

    protected $primaryKey = 'id_carte';
}

```


Anexa 3 - Controllere

```
<?php

namespace App\Http\Controllers;

use App\Models\CarteCategorie;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class CarteCategorieController extends Controller
{
    /**
     * @return JsonResponse
     */
    public function index()
    {
        $data = CarteCategorie::all();

        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @param Request $request
     * @return JsonResponse
     */
    public function store(Request $request)
    {
        $data = new CarteCategorie;

        $data->id_carte = $request-
>input('id_carte');
        $data->id_categorie = $request-
>input('id_categorie');

        $data->save();

        return response()->json($data,
Response::HTTP_CREATED);
    }

    /**
     * @param Request $request
     * @return JsonResponse
     */
    public function update(Request $request)
    {
        $data =
CarteCategorie::findOrFail($request-
>input('id'));

        $data->id_carte = $request-
>input('id_carte');
        $data->id_categorie = $request-
>input('id_categorie');

        $data->save();

        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @param $id
     * @return JsonResponse
     */
    public function destroy(Request $request)
    {
        $data =
CarteCategorie::findOrFail($request-
>input('id_carte'));
```

```
$data->delete();

        return response()->json("DELETED",
Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\CarteComanda;
use App\Services\CarteComandaService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;

class CarteComandaController extends Controller
{
    private CarteComandaService
$cartecomandaService;
    public function
_construct(CarteComandaService
$cartecomandaService)
    {
        $this->cartecomandaService =
$cartecomandaService;
    }

    /**
     * @OA\Get(
     *     path="/carti-comanda",
     *     summary="Returneaza toate cartile
comenzii",
     *     tags={"Carti-comanda"},
     *     description="Returneaza toate
cartile comenzii",
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function index(): JsonResponse
    {
        $data = CarteComanda::all();
        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @OA\Get(
     *     path="/carti-comanda/{id}",
     *     summary="Returneaza cartea comenzii
cu id-ul dat",
     *     tags={"Carti-comanda"},
     *     description="Returneaza cartea
comenzii cu id-ul dat",
     *     @OA\Parameter(
     *         name="id",
     *         description="Id-ul cartii
comenzii dorite",
     *         required=true,
     *         in="path",
     *         example=3,
     *         @OA\Schema(
     *             type="integer"
     *         )
     *     ),
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function show($id): JsonResponse
    {
```

```

        $data = CarteComanda::where('id', '=',
$idid)->first();
        return response()->json($data,
Response::HTTP_OK);
    }

/**
 * @OA\Put (
 *     path="/carti-comanda",
 *     summary="Update cartea comenzii cu
id-ul dat",
 *     tags={"Carti-comanda"},
 *     description="Update cartea comenzii
cu id-ul dat",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="id_carte",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="id_comanda",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="cantitate",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="subtotal",
 *                 type="float",
 *                 description="",
 *                 example=1.0,
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 */
public function update(Request $request):
JsonResponse
{
    $data = CarteComanda::where('id', '=',
$request['id'])->firstOrFail();
    $this->cartecomandaService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Post (
 *     path="/carti-comanda",
 *     summary="Adauga cartea comenzii",
 *     tags={"Carti-comanda"},
 *     description="Adauga cartea
comenzii",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="id_carte",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="id_comanda",

```

```

 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="cantitate",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="subtotal",
 *                 type="float",
 *                 description="",
 *                 example=1.0,
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 */
public function store(Request $request):
JsonResponse
{
    $data = new CarteComanda;
    $this->cartecomandaService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Delete (
 *     path="/carti-comanda/{id}",
 *     summary="Sterge cartea comenzii cu
id-ul dat",
 *     tags={"Carti-comanda"},
 *     description="Sterge cartea comenzii
cu id-ul dat",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul cartii
comenzii dorite",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 */
public function destroy($id): JsonResponse
{
    $data = CarteComanda::where('id', '=',
$idid)->firstOrFail();
    $data->delete();
    return response()->json(['message' =>
'Sters'], Response::HTTP_OK);
}

/**
 * @param Request $request
 * @return JsonResponse
 */
public function returnOrder(Request
$request): JsonResponse
{
    if (!Auth::check()) {
        return response()->json(['message'
=> 'Neautorizat'],

```

```

Response::HTTP_UNAUTHORIZED);
    }

    $carteComanda =
CarteComanda::where('id_comanda', '=',
$request['id_comanda'])
->where('id_carte', '=',
$request['id_carte'])
->first();

    $carteComanda->status = 'returnata';
    $carteComanda->save();

    return response()->json(['message' =>
'Salvat'], Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Carte;
use App\Models\Discount;
use App\Services\CarteService;
use
Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\DB;

class CarteController extends Controller
{
    private CarteService $carteService;
    public function __construct(CarteService
$carteService)
    {
        $this->carteService = $carteService;
    }

    /**
     * @OA\Get (
     *     path="/carti",
     *     summary="Returneaza toate cartile",
     *     tags={"Carti"},
     *     description="Returneaza toate
cartile",
     *     @OA\Response (
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function index(): JsonResponse
    {
        $data = Carte::all();
        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @OA\Get (
     *     path="/carti/{id}",
     *     summary="Returneaza cartea cu id-ul
dat",
     *     tags={"Carti"},
     *     description="Returneaza cartea cu
id-ul dat",
     *     @OA\Parameter (
     *         name="id",
     *         description="Id-ul cartii
dorate",
     *         required=true,
     *         in="path",
     *         example=3,
     *         @OA\Schema (

```

```

         type="integer"
     *     )
     * ),
     * @OA\Response (
     *     response=200,
     *     description="Success"
     * )
     * )
     */
    public function show($id): JsonResponse
    {
        $data = Carte::where('id', '=', $id)-
>firstOrFail();
        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @OA\Put (
     *     path="/carti",
     *     summary="Update cartea cu id-ul
dat",
     *     tags={"Carti"},
     *     description="Update cartea cu id-ul
dat",
     *     @OA\RequestBody (
     *         required=true,
     *         @OA\JsonContent (
     *             @OA\Property (
     *                 property="id",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="titlu",
     *                 type="string",
     *                 description="",
     *                 example="",
     *             ),
     *             @OA\Property (
     *                 property="autor",
     *                 type="string",
     *                 description="",
     *                 example="",
     *             ),
     *             @OA\Property (
     *                 property="imagine",
     *                 type="string",
     *                 description="",
     *                 example="",
     *             ),
     *             @OA\Property (
     *                 property="isbn",
     *                 type="string",
     *                 description="",
     *                 example="",
     *             ),
     *             @OA\Property (
     *                 property="editura",
     *                 type="string",
     *                 description="",
     *                 example="",
     *             ),
     *             @OA\Property (
     *                 property="pret",
     *                 type="float",
     *                 description="",
     *                 example=1.0,
     *             ),
     *             @OA\Property (
     *                 property="an_aparitie",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="nr_pg",

```

```

*         type="integer",
*         description="",
*         example=1,
*     ),
*     @OA\Property(
property="tip_coperta",
*         type="string",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="limba",
*         type="string",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="dimensiune",
*         type="string",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="cantitate",
*         type="integer",
*         description="",
*         example=1,
*     )
* ),
* @OA\Response(
*     response=200,
*     description="Success"
* )
*/
public function update(Request $request):
JsonResponse
{
    $data = Carte::where('id', '=',
$request['id'])->firstOrFail();
    $this->carteService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Post(
 *     path="/carti",
 *     summary="Adauga cartea",
 *     tags={"Carti"},
 *     description="Adauga cartea",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="titlu",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property(
 *                 property="autor",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property(
 *                 property="imagine",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property(
 *                 property="isbn",
 *                 type="string",

```

```

*                 description="",
*                 example="",
*             ),
*             @OA\Property(
*                 property="editura",
*                 type="string",
*                 description="",
*                 example="",
*             ),
*             @OA\Property(
*                 property="pret",
*                 type="float",
*                 description="",
*                 example=1.0,
*             ),
*             @OA\Property(
property="an_aparitie",
*                 type="integer",
*                 description="",
*                 example=1,
*             ),
*             @OA\Property(
*                 property="nr_pg",
*                 type="integer",
*                 description="",
*                 example=1,
*             ),
*             @OA\Property(
property="tip_coperta",
*                 type="string",
*                 description="",
*                 example="",
*             ),
*             @OA\Property(
*                 property="limba",
*                 type="string",
*                 description="",
*                 example="",
*             ),
*             @OA\Property(
*                 property="dimensiune",
*                 type="string",
*                 description="",
*                 example="",
*             ),
*             @OA\Property(
*                 property="cantitate",
*                 type="integer",
*                 description="",
*                 example=1,
*             ),
*             @OA\Property(
*                 property="categorii",
*                 type="string",
*                 description="",
*                 example="Beletristica",
*             )
*         ),
*     @OA\Response(
*         response=200,
*         description="Success"
*     )
* )
*/
public function store(Request $request):
JsonResponse
{
    $data = new Carte;
    $this->carteService-
>setProperties($data, $request->all());
    return response()->json($data,
Response::HTTP_OK);
}

/**

```

```

* @OA\Delete(
*   path="/carti/{id}",
*   summary="Sterge cartea cu id-ul
dat",
*   tags={"Carti"},
*   description="Sterge cartea cu id-ul
dat",
*   @OA\Parameter(
*     name="id",
*     description="Id-ul cartii
dorite",
*     required=true,
*     in="path",
*     example=3,
*     @OA\Schema(
*       type="integer"
*     )
*   ),
*   @OA\Response(
*     response=200,
*     description="Success"
*   )
* )
*/
public function destroy($id): JsonResponse
{
    $data = Carte::where('id', '=', $id)-
>firstOrFail();
    $data->delete();
    return response()->json("DELETED",
Response::HTTP_OK);
}

/**
* @OA\Get(
*   path="/carti/categorii/{id}",
*   summary="Returneaza categoriile
cartii cu id-ul dat",
*   tags={"Carti"},
*   description="Returneaza categoriile
cartii cu id-ul dat",
*   @OA\Parameter(
*     name="id",
*     description="Id-ul cartii
dorite",
*     required=true,
*     in="path",
*     example=3,
*     @OA\Schema(
*       type="integer"
*     )
*   ),
*   @OA\Response(
*     response=200,
*     description="Success"
*   )
* )
*/
public function categorii($id):
JsonResponse
{
    $data = Carte::where('id', '=', $id)-
>firstOrFail();
    $data = $data->categorii()->get();

    return response()->json($data,
Response::HTTP_OK);
}

/**
* @OA\Get(
*   path="/carti/reviewuri/{id}",
*   summary="Returneaza reviewurile
cartii cu id-ul dat",
*   tags={"Carti"},
*   description="Returneaza reviewurile
cartii cu id-ul dat",
*   @OA\Parameter(
*     name="id",

```

```

*     description="Id-ul cartii
dorite",
*     required=true,
*     in="path",
*     example=3,
*     @OA\Schema(
*       type="integer"
*     )
*   ),
*   @OA\Response(
*     response=200,
*     description="Success"
*   )
* )
*/
public function reviewuri($id):
JsonResponse
{
    $data = Carte::where('id', '=', $id)-
>firstOrFail();
    $data = $data->reviewuri()->get();

    return response()->json($data,
Response::HTTP_OK);
}

/**
* @param Request $request
* @return string
*/
public function search(Request $request)
{
    $validatedData = $request->validate([
        'search' => 'required',
    ]);

    $carti = Carte::where('titlu', 'like',
'%' . $validatedData['search'] . '%')
->orWhere('autor', 'like', '%' .
$validatedData['search'] . '%')
->orWhere('isbn', 'like', '%' .
$validatedData['search'] . '%')
->get();

    $data = [];
    foreach ($carti as $carte) {
        $data[] = [
            'titlu' => $carte->titlu,
            'image' => $carte->image,
            'autor' => $carte->autor,
            'isbn' => $carte->isbn,
            'pret' => $carte->pret,
            'id' => $carte->id,
            'cantitate' => $carte-
>cantitate,
            'nr_reviewuri' => $carte-
>nr_reviewuri,
            'rating' =>
number_format($carte->rating, 2, '.', ''),
        ];
    }

    $data = json_encode($data);

    return route('search', ['data' =>
$data]);
}

/**
* @param Request $request
* @return Application|Factory|View
*/
public function paginaCautare(Request
$request)
{
    $data = $request->input('data');

    if (empty($data)) {
        $data = Carte::all();
    }
}

```

```

    }

    return view('pagina-cautare',
compact('data'));
}

/**
 * @param Request $request
 * @return JsonResponse
 */
public function queryBuilder(Request
$request): JsonResponse
{
    $validatedData = $request->validate([
        'autor' => 'nullable',
        'editura' => 'nullable',
        'categorii' => 'nullable',
        'pret' => 'nullable',
        'limba' => 'nullable',
        'ordine' => 'nullable',
    ]);

    $query = DB::table('carte');

    if ($validatedData['autor'] != null) {
        $query->whereIn('autor',
$validatedData['autor']);
    }

    if ($validatedData['pret'] != null) {
        $query->whereBetween('pret',
$validatedData['pret']);
    }

    if ($validatedData['limba'] != null) {
        $query->whereIn('limba',
$validatedData['limba']);
    }

    if ($validatedData['editura'] != null)
    {
        $query->whereIn('editura',
$validatedData['editura']);
    }

    if ($validatedData['categorii'] !=
null) {
        $query->join('carte_categorii',
'carte.id', '=', 'carte_categorii.id_carte')
->join('categorii',
'carte_categorii.id_categorie', '=',
'categorii.id')
->whereIn('categorii.nume',
$validatedData['categorii'])
->select('carte.*');
    }

    if ($validatedData['ordine'] != null)
    {
        if ($validatedData['ordine'] ===
'pret_crescator') {
            $query->orderBy('pret',
'asc');
        } else {
            if ($validatedData['ordine']
=== 'pret_descrescator') {
                $query->orderBy('pret',
'desc');
            } else {
                if
($validatedData['ordine'] ===
'alfabetic_crescator') {
                    $query-
>orderBy('titlu', 'asc');
                } else {
                    if
($validatedData['ordine'] ===
'alfabetic_descrescator') {

```

```

                        $query-
>orderBy('titlu', 'desc');
                    }
                }
            }
        }

        $results = $query->distinct()->get();
        $carte_ids = [];

        if (!empty($results)) {
            foreach ($results as $carte) {
                if (empty($carte_ids)) {
                    $carte_ids[] = $carte->id;
                } else {
                    if (array_search($carte-
>id, $carte_ids)) {
                        $results-
>forget($carte);
                        continue;
                    }
                }

                $reviews =
DB::table('review')->where('id_carte', '=',
$carte->id)->get();
                if (!empty($reviews)) {
                    $carte->nr_reviewuri =
count($reviews);

                    $carte->rating = 0;
                    foreach ($reviews as
$review) {
                        $carte->rating +=
$review->rating;
                    }

                    if ($carte->nr_reviewuri
!= 0) {
                        $carte->rating =
$carte->rating / $carte->nr_reviewuri;
                    } else {
                        $carte->rating = 0.00;
                    }
                } else {
                    $carte->nr_reviewuri = 0;
                    $carte->rating = 0.00;
                }

                DB::table('carte')
->where('id', $carte->id)
->update([
                    'nr_reviewuri' =>
$carte->nr_reviewuri,
                    'rating' => $carte-
>rating,
                ]);
            }
        }

        return response()->json($results,
Response::HTTP_OK);
    }

    /**
     * @return JsonResponse
     */
    public function getFilters()
    {
        $autori = DB::table('carte')-
>select('autor')->distinct()->get();
        $edituri = DB::table('carte')-
>select('editura')->distinct()->get();
        $categorii = DB::table('categorii')-
>select('nume')->distinct()->get();
        $limbi = DB::table('carte')-
>select('limba')->distinct()->get();
    }
}

```

```

        $data = [
            'autori' => $autori,
            'edituri' => $edituri,
            'categorii' => $categorii,
            'limbi' => $limbi,
        ];

        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @param $isbn
     * @return Application|Factory|View
     */
    public function getBookByIsbn($isbn)
    {
        $data = Carte::where('isbn', '=',
$isbn)->firstOrFail();

        return view('pagina-carte',
compact('data'));
    }

    /**
     * @param int $id
     * @return JsonResponse
     */
    public function getRecommandations(int
$id)
    {
        $carte = Carte::find($id);

        $categorieIds = $carte->categorii()-
>pluck('id')->toArray();

        $recommandations =
Carte::where('autor', $carte->autor)
->orWhereHas('categorii', function
($query) use ($categorieIds) {
            $query->whereIn('id',
$categorieIds);
        })
->get();

        $recommandations = $recommandations-
>reject(function ($recommandation) use
($carte) {
            return $recommandation->titlu ==
$carte->titlu;
        });

        return response()-
>json($recommandations, Response::HTTP_OK);
    }

    /**
     * @return JsonResponse
     */
    public function getLatestCarti()
    {
        $latest = Carte::orderBy('created_at',
'desc')->take(20)->get();

        return response()->json($latest,
Response::HTTP_OK);
    }

    /**
     * @return JsonResponse
     */
    public function getDiscountedCarti()
    {
        $cartiId =
Discount::select('id_carte')->get();
        $carti = Carte::whereIn('id',
$cartiId)->get();

```

```

        return response()->json($carti,
Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Carte;
use App\Models\CarteCos;
use App\Models\User;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;

class CarteCosController extends Controller
{
    /**
     * @OA\Post (
     *     path="/cos",
     *     summary="Adauga o carte in cos",
     *     tags={"CarteCos"},
     *     description="Adauga o carte in cos",
     *     @OA\RequestBody(
     *         required=true,
     *         @OA\JsonContent (
     *             @OA\Property (
     *                 property="id_carte",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="cantitate",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="subtotal",
     *                 type="float",
     *                 description="",
     *                 example=1.0,
     *             )
     *         )
     *     ),
     *     @OA\Response (
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function store(Request $request):
JsonResponse
    {
        $data = new CarteCos;

        if (Auth::check()) {
            $user =
User::findOrFail(Auth::user()->id);
            $client = $user->client;
            $data->id_client = $client->id;
            $carteCos =
CarteCos::where('id_carte', '=', $request-
>input('id_carte'))
->where('id_client', '=',
$client->id)
->first();
        } else {
            $data->id_client = null;
            $carteCos =
CarteCos::where('id_carte', '=', $request-
>input('id_carte'))->first();
        }

        if (!empty($carteCos)) {

```

```

        $carteCos->cantitate += $request-
>get('cantitate');
        $carteCos->subtotal += $request-
>get('subtotal');
        $carteCos->save();
        return response()->json($carteCos,
Response::HTTP_OK);
    }

    $data->id_carte = $request-
>get('id_carte');
    $data->cantitate = $request-
>get('cantitate');
    $data->subtotal = $request-
>get('subtotal');

    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Put (
 *     path="/cos",
 *     summary="Update cantitatea cartii
din cos",
 *     tags={"CarteCos"},
 *     description="Update cantitatea
cartii din cos",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="cantitate",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function update(Request $request):
JsonResponse
{
    if (Auth::check()) {
        $user =
User::findOrFail(Auth::user()->id);
        $client = $user->client;
        $carteCos =
CarteCos::where('id_carte', '=', $request-
>input('id_carte'))
            ->where('id_client', '=',
$client->id)
            ->first();
    } else {
        $carteCos =
CarteCos::where('id_carte', '=', $request-
>input('id_carte'))
            ->first();
    }

    $carteCos->cantitate = $request-
>get('cantitate');

    $carteCos->save();
    return response()->json("Cantitate
schimbata!", Response::HTTP_OK);
}

/**
 * @OA>Delete(
 *     path="/cos/{id}",
 *     summary="Sterge carte cu id-ul dat

```

```

din cos",
 *     tags={"CarteCos"},
 *     description="Sterge carte cu id-ul
dat din cos",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul cartii
dorite",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function destroy($id): JsonResponse
{
    if (Auth::check()) {
        $user =
User::findOrFail(Auth::user()->id);
        $client = $user->client;
        $carteCos =
CarteCos::where('id_carte', '=', $id)
            ->where('id_client', '=',
$client->id)
            ->first();
    } else {
        $carteCos =
CarteCos::where('id_carte', '=', $id)-
>first();
    }

    $carteCos->delete();
    return response()->json("DELETED",
Response::HTTP_OK);
}

/**
 * @return JsonResponse
 */
public function index(): JsonResponse
{
    $userLogged = Auth::user();
    $data = [];
    if (!empty($userLogged)) {
        $user =
User::findOrFail($userLogged->id);
        $client = $user->client;
        $cartiId =
CarteCos::where('id_client', '=', $client-
>id)->get();
        foreach ($cartiId as $carteCos) {
            $carte = Carte::where('id',
'=', $carteCos->id_carte)->first();

            $data[] = [
                'titlu' => $carte->titlu,
                'imagine' => $carte-
>imagine,
                'autor' => $carte->autor,
                'pret' => $carte->pret,
                'id' => $carte->id,
                'cantitate' => $carte-
>cantitate,
                'cos' => $carteCos-
>cantitate,
            ];
        }
    } else {
        $cartiId =
CarteCos::where('id_client', '=', null)-
>get();
    }
}

```



```

        foreach ($cartiId as $carteCos) {
            $carte = Carte::where('id',
            '=', $carteCos->id_carte)->first();

            $data[] = [
                'titlu' => $carte->titlu,
                'image' => $carte-
            >image,
                'autor' => $carte->autor,
                'pret' => $carte->pret,
                'id' => $carte->id,
                'cantitate' => $carte-
            >cantitate,
                'cos' => $carteCos-
            >cantitate,
            ];
        }

        return response()->json($data,
        Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Categorie;
use App\Services\CategorieService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class CategorieController extends Controller
{
    private CategorieService
    $categorieService;

    public function
    __construct(CategorieService
    $categorieService)
    {
        $this->categorieService =
        $categorieService;
    }

    /**
     * @OA\Get(
     *     path="/categorii",
     *     summary="Returneaza toate
    categoriile",
     *     tags={"Categorii"},
     *     description="Returneaza toate
    categoriile",
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function index(): JsonResponse
    {
        $data = Categorie::all();
        return response()->json($data,
        Response::HTTP_OK);
    }

    /**
     * @OA\Get(
     *     path="/categorii/{id}",
     *     summary="Returneaza categoria cu id-
    ul dat",
     *     tags={"Categorii"},
     *     description="Returneaza categoria cu
    id-ul dat",
     *     @OA\Parameter(
     *         name="id",
     *         description="Id-ul categoriei
    dorite",

```

```

     *         required=true,
     *         in="path",
     *         example=3,
     *         @OA\Schema(
     *             type="integer"
     *         )
     *     ),
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function show($id): JsonResponse
    {
        $data = Categorie::where('id', '=',
        $id)->first();
        return response()->json($data,
        Response::HTTP_OK);
    }

    /**
     * @OA\Put(
     *     path="/categorii",
     *     summary="Update categoria cu id-ul
    dat",
     *     tags={"Categorii"},
     *     description="Update categorii cu id-
    ul dat",
     *     @OA\RequestBody(
     *         required=true,
     *         @OA\JsonContent(
     *             @OA\Property(
     *                 property="id",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property(
     *                 property="parent_id",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property(
     *                 property="nume",
     *                 type="string",
     *                 description="",
     *                 example="",
     *             )
     *         )
     *     ),
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function update(Request $request):
    JsonResponse
    {
        $data = Categorie::where('id', '=',
        $request['id']->first();
        $this->categorieService-
        >setProperties($data, $request->all());
        $data->save();
        return response()->json($data,
        Response::HTTP_OK);
    }

    /**
     * @OA\Post(
     *     path="/categorii",
     *     summary="Adauga categoria",
     *     tags={"Categorii"},
     *     description="Adauga categorii",
     *     @OA\RequestBody(
     *         required=true,
     *         @OA\JsonContent(

```

```

*         @OA\Property(
*             property="parent_id",
*             type="integer",
*             description="",
*             example=1,
*         ),
*         @OA\Property(
*             property="nume",
*             type="string",
*             description="",
*             example="",
*         )
*     ),
*     @OA\Response(
*         response=200,
*         description="Success"
*     )
* )
*/
public function store(Request $request):
JsonResponse
{
    $data = new Categorie;
    $this->categorieService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Delete(
 *     path="/categorii/{id}",
 *     summary="Sterge categoria cu id-ul
dat",
 *     tags={"Categorii"},
 *     description="Sterge categoria cu id-
ul dat",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul categoriei
dorite",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function destroy($id): JsonResponse
{
    $data = Categorie::where('id', '=',
$idid)->first();
    $data->delete();
    return response()->json("DELETED",
Response::HTTP_OK);
}

/**
 * @OA\Get(
 *     path="/categorii/{id}",
 *     summary="Returneaza cartile din
categoria cu id-ul dat",
 *     tags={"Categorii"},
 *     description="Returneaza cartile din
categoria cu id-ul dat",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul categoriei
dorite",
 *         required=true,
 *         in="path",

```

```

*         example=3,
*         @OA\Schema(
*             type="integer"
*         )
*     ),
*     @OA\Response(
*         response=200,
*         description="Success"
*     )
* )
*/
public function carti($id): JsonResponse
{
    $data = Categorie::where('id', '=',
$idid)->first();
    $data = $data->carti()->get();

    return response()->json($data,
Response::HTTP_OK);
}
<?php

namespace App\Http\Controllers;

use App\Models\Carte;
use App\Models\Client;
use App\Models\User;
use App\Services\ClientService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;

class ClientController extends Controller
{
    private ClientService $clientService;
    public function __construct(ClientService
$clientService)
    {
        $this->clientService = $clientService;
    }

    /**
     * @OA\Get(
     *     path="/clienti",
     *     summary="Returneaza toti clientii",
     *     tags={"Clienti"},
     *     description="Returneaza toti
clientii",
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function index(): JsonResponse
    {
        $data = Client::all();
        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @OA\Get(
     *     path="/clienti/{id}",
     *     summary="Returneaza clientul cu id-
ul dat",
     *     tags={"Clienti"},
     *     description="Returneaza clientul cu
id-ul dat",
     *     @OA\Parameter(
     *         name="id",
     *         description="Id-ul clientului
dorit",
     *         required=true,
     *         in="path",
     *         example=3,
     *         @OA\Schema(

```

```

*         type="integer"
*     )
* ),
* @OA\Response(
*     response=200,
*     description="Success"
* )
*/
public function show($id): JsonResponse
{
    $data = Client::where('id', '=', $id)-
>firstOrFail();

    $data->email = $data->user()->get()-
>email;

    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Put(
 *     path="/clienti",
 *     summary="Update clientul cu id-ul
dat",
 *     tags={"Clienti"},
 *     description="Update clientul cu id-
ul dat",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="id",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="nume",
 *                 type="string",
 *                 description="",
 *                 example="Petrache",
 *             ),
 *             @OA\Property(
 *                 property="prenume",
 *                 type="string",
 *                 description="",
 *                 example="Cristina",
 *             ),
 *             @OA\Property(
 *                 property="nr_telefon",
 *                 type="string",
 *                 description="",
 *                 example="0755158838",
 *             ),
 *             @OA\Property(
 *                 property="adresa",
 *                 type="string",
 *                 description="",
 *                 example="Str. Padurii,
Nr. 85",
 *             ),
 *             @OA\Property(
 *                 property="oras",
 *                 type="string",
 *                 description="",
 *                 example="Sabareni",
 *             ),
 *             @OA\Property(
 *                 property="judet",
 *                 type="string",
 *                 description="",
 *                 example="Giurgiu",
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,

```

```

*         description="Success"
*     )
* )
*/
public function update(Request $request):
JsonResponse
{
    $data = Client::where('id', '=',
$request['id']->firstOrFail());
    $this->clientService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Post(
 *     path="/clienti",
 *     summary="Adauga clientul",
 *     tags={"Clienti"},
 *     description="Adauga clientul",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="nume",
 *                 type="string",
 *                 description="",
 *                 example="Petrache",
 *             ),
 *             @OA\Property(
 *                 property="prenume",
 *                 type="string",
 *                 description="",
 *                 example="Cristina",
 *             ),
 *             @OA\Property(
 *                 property="nr_telefon",
 *                 type="string",
 *                 description="",
 *                 example="0755158838",
 *             ),
 *             @OA\Property(
 *                 property="adresa",
 *                 type="string",
 *                 description="",
 *                 example="Str. Padurii,
Nr. 85",
 *             ),
 *             @OA\Property(
 *                 property="oras",
 *                 type="string",
 *                 description="",
 *                 example="Sabareni",
 *             ),
 *             @OA\Property(
 *                 property="judet",
 *                 type="string",
 *                 description="",
 *                 example="Giurgiu",
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
*/
public function store(Request $request):
JsonResponse
{
    $data = new Client;
    $this->clientService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

```

```

    }

    /**
     * @OA\Delete(
     *   path="/clienti/{id}",
     *   summary="Sterge clientul cu id-ul
dat",
     *   tags={"Clienti"},
     *   description="Sterge clientul cu id-
ul dat",
     *   @OA\Parameter(
     *     name="id",
     *     description="Id-ul clientului
dorit",
     *     required=true,
     *     in="path",
     *     example=3,
     *     @OA\Schema(
     *       type="integer"
     *     )
     *   ),
     *   @OA\Response(
     *     response=200,
     *     description="Success"
     *   )
     *)
    */
    public function destroy($id): JsonResponse
    {
        $data = Client::where('id', '=', $id)-
>firstOrFail();
        $data->delete();
        return response()->json("DELETED",
Response::HTTP_OK);
    }

    /**
     * @OA\Get(
     *   path="/client-autentificat",
     *   summary="Returneaza clientul
autentificat",
     *   tags={"Clienti"},
     *   description="Returneaza clientul
autentificat",
     *   @OA\Response(
     *     response=200,
     *     description="Success"
     *   )
     *)
    */
    public function getUserData():
JsonResponse
    {
        $user = auth()->user();
        $client = $user->client()->get();
        return response()->json($client,
Response::HTTP_OK);
    }

    public function addFavoriteBook(Request
$request)
    {
        if (Auth::check()) {
            $user =
User::findOrFail(Auth::user()->id);
        } else {
            return redirect()-
>intended(route('login'));
        }

        $carte = Carte::where('id', '=',
$request->get('id_carte'))
            ->select('titlu',
'autor', 'imagine', 'pret')
            ->first();

        $client = $user->client();
        $client->favorite[] = $carte;
        $client->save();
    }

```

```

        return response()->json("Carte
adaugata", Response::HTTP_OK);
    }

    /**
     * @OA\Get(
     *   path="/clienti/comenzi",
     *   summary="Returneaza comenzile
clientului cu id-ul dat",
     *   tags={"Clienti"},
     *   description="Returneaza comenzile
clientului cu id-ul dat",
     *   @OA\Parameter(
     *     name="id",
     *     description="Id-ul clientului
dorit",
     *     required=true,
     *     in="path",
     *     example=3,
     *     @OA\Schema(
     *       type="integer"
     *     )
     *   ),
     *   @OA\Response(
     *     response=200,
     *     description="Success"
     *   )
     *)
    */
    public function comenzi($id): JsonResponse
    {
        if (Auth::check()) {
            $user =
User::findOrFail(Auth::user()->id);
        } else {
            return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
        }
        $data = $user->client()->comenzi()-
>get();

        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @OA\Get(
     *   path="/clienti/favorite",
     *   summary="Returneaza cartile favorite
ale clientului cu id-ul dat",
     *   tags={"Clienti"},
     *   description="Returneaza cartile
favorite ale clientului cu id-ul dat",
     *   @OA\Parameter(
     *     name="id",
     *     description="Id-ul clientului
dorit",
     *     required=true,
     *     in="path",
     *     example=3,
     *     @OA\Schema(
     *       type="integer"
     *     )
     *   ),
     *   @OA\Response(
     *     response=200,
     *     description="Success"
     *   )
     *)
    */
    public function favorite($id):
JsonResponse
    {
        if (Auth::check()) {
            $user =
User::findOrFail(Auth::user()->id);
        } else {

```

```

        return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
    }
    $data = $user->client()->favorite()-
>get();

    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Get(
 *     path="/clienti/cos",
 *     summary="Returneaza cartile din cos
ale clientului cu id-ul dat",
 *     tags={"Clienti"},
 *     description="Returneaza cartile din
cos ale clientului cu id-ul dat",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul clientului
dorit",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function cos($id): JsonResponse
{
    if (Auth::check()) {
        $user =
User::findOrFail(Auth::user()->id);
    } else {
        return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
    }
    $data = $user->client()->cartiCos()-
>get();

    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Get(
 *     path="/clienti/user",
 *     summary="Returneaza datele
clientului autentificat",
 *     tags={"Clienti"},
 *     description="Returneaza datele
clientului autentificat",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul clientului
dorit",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function userData(): JsonResponse

```

```

    {
        if (Auth::check()) {
            $user =
User::findOrFail(Auth::user()->id);
        } else {
            return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
        }

        $data = $user->client()->get();

        $data->email = $user->email;

        return response()->json($data,
Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Mail\ConfirmOrder;
use App\Models\Carte;
use App\Models\CarteComanda;
use App\Models\CarteCos;
use App\Models\Comanda;
use App\Models\Discount;
use App\Models\User;
use App\Services\ComandaService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Mail;

class ComandaController extends Controller
{
    private ComandaService $comandaService;
    public function __construct(ComandaService
$comandaService)
    {
        $this->comandaService =
$comandaService;
    }

    /**
     * @OA\Get(
     *     path="/comenzi",
     *     summary="Returneaza toate
comenzile",
     *     tags={"Clienti"},
     *     description="Returneaza toate
comenzile",
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function index(): JsonResponse
    {
        if (Auth::check()) {
            $user = User::find(Auth::id());
            $client = $user->client;
        } else {
            return response()->json(['message'
=> 'Nu sunteti logat'],
Response::HTTP_UNAUTHORIZED);
        }

        $data = Comanda::where('id_client',
'=', $client->id)->get();
        return response()->json($data,
Response::HTTP_OK);
    }
}

/**

```

```

* @return JsonResponse
*/
public function adminIndex()
{
    $data = Comanda::all();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Get (
 *     path="/comenzi/{id}",
 *     summary="Returneaza comanda cu id-ul
dat",
 *     tags={"Comenzi"},
 *     description="Returneaza comanda cu
id-ul dat",
 *     @OA\Parameter (
 *         name="id",
 *         description="Id-ul comenzii
dorate",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema (
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response (
 *         response=200,
 *         description="Success"
 *     )
 * )
*/
public function show($id): JsonResponse
{
    $data = Comanda::where('id', '=',
$idid)->firstOrFail();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Put (
 *     path="/comenzi",
 *     summary="Update comanda cu id-ul
dat",
 *     tags={"Comenzi"},
 *     description="Update comanda cu id-ul
dat",
 *     @OA\RequestBody (
 *         required=true,
 *         @OA\JsonContent (
 *             @OA\Property (
 *                 property="id",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property (
 *                 property="id_client",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property (
 *                 property="tip",
 *                 type="string",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property (
 *
property="data_plasare",
 *                 type="date",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (

```

```

*
property="data_livrare",
 *                 type="date",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (
 *
property="data_predare",
 *                 type="date",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (
 *
property="pret_comanda",
 *                 type="float",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property (
 *
property="adresa_livrare",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (
 *
property="oras_livrare",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (
 *
property="judet_livrare",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (
 *                 property="status",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             ),
 *             @OA\Property (
 *
property="metoda_plata",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             )
 *         ),
 *         @OA\Response (
 *             response=200,
 *             description="Success"
 *         )
 *     )
*/
public function update(Request $request):
JsonResponse
{
    $data = Comanda::where('id', '=',
$request['id'])->firstOrFail();
    $this->comandaService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Post (
 *     path="/comenzi",
 *     summary="Adauga comanda",
 *     tags={"Comenzi"},

```

```

*      description="Adauga comanda",
*      @OA\RequestBody(
*          required=true,
*          @OA\JsonContent(
*              @OA\Property(
*                  property="id_client",
*                  type="integer",
*                  description="",
*                  example=1,
*              ),
*              @OA\Property(
*                  property="tip",
*                  type="string",
*                  description="",
*                  example=1,
*              ),
*              @OA\Property(
*                  property="data_plasare",
*                  type="date",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="data_livrare",
*                  type="date",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="data_predare",
*                  type="date",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="pret_comanda",
*                  type="float",
*                  description="",
*                  example=1,
*              ),
*              @OA\Property(
*                  property="adresa_livrare",
*                  type="string",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="oras_livrare",
*                  type="string",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="judet_livrare",
*                  type="string",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="status",
*                  type="string",
*                  description="",
*                  example="",
*              ),
*              @OA\Property(
*                  property="metoda_plata",
*                  type="string",
*                  description="",
*                  example="",
*              )
*          )
*      )

```

```

*      ),
*      @OA\Response(
*          response=200,
*          description="Success"
*      )
*  )
*/
public function store(Request $request):
JsonResponse
{
    $data = new Comanda;
    $this->comandaService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
* @OA\Delete(
*     path="/comenzi/{id}",
*     summary="Sterge comanda cu id-ul
dat",
*     tags={"Comenzi"},
*     description="Sterge comanda cu id-ul
dat",
*     @OA\Parameter(
*         name="id",
*         description="Id-ul comenzii
dorite",
*         required=true,
*         in="path",
*         example=3,
*         @OA\Schema(
*             type="integer"
*         )
*     ),
*     @OA\Response(
*         response=200,
*         description="Success"
*     )
* )
*/
public function destroy($id): JsonResponse
{
    $data = Comanda::where('id', '=',
$id)->firstOrFail();
    $data->delete();
    return response()->json("DELETED",
Response::HTTP_OK);
}

/**
* @OA\Get(
*     path="/comenzi/carti/{id}",
*     summary="Returneaza cartile din
comanda cu id-ul dat",
*     tags={"Comenzi"},
*     description="Returneaza cartile din
comanda cu id-ul dat",
*     @OA\Parameter(
*         name="id",
*         description="Id-ul comenzii
dorite",
*         required=true,
*         in="path",
*         example=3,
*         @OA\Schema(
*             type="integer"
*         )
*     ),
*     @OA\Response(
*         response=200,
*         description="Success"
*     )
* )
*/
public function carti(): JsonResponse
{

```

```

        if (Auth::check()) {
            $user = User::find(Auth::id());
            $client = $user->client;
        } else {
            return response()->json(['message'
=> 'Nu sunteti logat'],
Response::HTTP_UNAUTHORIZED);
        }
        $data = Comanda::where('id_client',
            '=', $client->id)->select('id')->get();
        $cartiComanda = [];

        foreach ($data as $comanda) {
            $cartiComanda[] =
CarteComanda::where('id_comanda', '=',
$comanda->id)->get();
        }

        $cartiId = [];

        foreach ($cartiComanda as
$carteComanda) {
            foreach ($carteComanda as $carte)
            {
                $cartiId[] = $carte->id_carte;
            }
        }

        $carti = Carte::whereIn('id',
$cartiId)->distinct()->get();

        return response()->json($carti,
Response::HTTP_OK);
    }

/**
 * @param int $id_comanda
 * @return JsonResponse
 */
public function cartiComanda(int
$id_comanda): JsonResponse
{
    if (Auth::check()) {
        $user = User::find(Auth::id());
        $client = $user->client;
    } else {
        return response()->json(['message'
=> 'Nu sunteti logat'],
Response::HTTP_UNAUTHORIZED);
    }

    $cartiComanda =
CarteComanda::where('id_comanda', '=',
$id_comanda)->get();

    $cartiId = [];

    foreach ($cartiComanda as
$carteComanda) {
        $cartiId[] = $carteComanda-
>id_carte;
    }

    $carti = Carte::whereIn('id',
$cartiId)->distinct()->get();

    return response()->json($carti,
Response::HTTP_OK);
}

/**
 * @param Request $request
 * @return JsonResponse
 */
public function checkout(Request $request)
{
    $data = new Comanda;

```

```

        $this->comandaService-
>setProperties($data, $request->all());
        $data->save();

        $discounts = Discount::all();

        foreach ($request->input('carti') as
$carte) {
            $carteComanda = new CarteComanda;

            $carteComanda->id_comanda = $data-
>id;
            $carteComanda->id_carte =
$carte['id'];
            $carteComanda->cantitate =
$carte['cos'];

            $checkDiscount = 0;
            foreach ($discounts as $discount)
            {
                if ($discount->id_carte ==
$carte['id']) {
                    $checkDiscount = 1;
                    $carteComanda->subtotal =
($carte['pret'] * $carte['cos']) -
($carte['pret'] * $carte['cos']) * $discount-
>discount / 100;
                    break;
                }
            }

            if ($checkDiscount == 0) {
                $carteComanda->subtotal =
$carte['pret'] * $carte['cos'];
            }

            $carteComanda->save();

            if ($request->input('id_client')
== 0) {
                $carteCos =
CarteCos::where('id_carte', '=', $carte['id'])
->first();
                $carteCos->delete();
            } else {
                $carteCos =
CarteCos::where('id_carte', '=', $carte['id'])
->where('id_client', '=',
$request->input('id_client'))
->first();
                $carteCos->delete();
            }
        }

        Mail::to($request->input('email'))-
>send(new ConfirmOrder($data->id));

        return response()->json("DONE",
Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Mail>ContactMail;
use Illuminate\Foundation\Auth\Access\AuthorizesRe
quests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequ
ests;
use Illuminate\Http\Request;
use Illuminate\Routing\Controller as
BaseController;
use Illuminate\Support\Facades\Mail;

```



```

/**
 * @OA\Info(title="Librarie API",
version="0.1")
 */

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs,
ValidatesRequests;

    /**
     * @param Request $request
     * @return void
     */
    public function sendContactMail(Request
$request)
    {
        $validatedData = $request->validate([
            'message' => 'required|string',
            'email' => 'required|email'
        ]);

        $message = $validatedData['message'];
        $email = $validatedData['email'];

        Mail::to('inkpaper2023@hotmail.com')->
send(new ContactMail($message, $email));
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Discount;
use Illuminate\Http\Response;

class DiscountController extends Controller
{
    /**
     * @return \Illuminate\Http\JsonResponse
     */
    public function index()
    {
        $discounts = Discount::all();

        return response()->json($discounts,
Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Client;
use App\Models\Donatie;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class DonatieController extends Controller
{
    /**
     * @OA\Post(
     *     path="/donatii",
     *     summary="Adauga donatia",
     *     tags={"Donatii"},
     *     description="Adauga donatia",
     *     @OA\RequestBody(
     *         required=true,
     *         @OA\JsonContent(
     *             @OA\Property(
     *                 property="nume",
     *                 type="string",
     *                 description="",
     *                 example="Petrache",

```

```

     *     ),
     *     @OA\Property(
     *         property="prenume",
     *         type="string",
     *         description="",
     *         example="Cristina",
     *     ),
     *     @OA\Property(
     *         property="email",
     *         type="string",
     *         description="",
     *         example="cristina.petrache@hotmail.com",
     *     ),
     *     @OA\Property(
     *         property="nr_telefon",
     *         type="string",
     *         description="",
     *         example="0755158838",
     *     ),
     *     @OA\Property(
     *         property="titlu",
     *         type="string",
     *         description="",
     *         example="",
     *     ),
     *     @OA\Property(
     *         property="autor",
     *         type="string",
     *         description="",
     *         example="",
     *     ),
     *     @OA\Property(
     *         property="isbn",
     *         type="string",
     *         description="",
     *         example="",
     *     ),
     *     @OA\Property(
     *         property="adresa_ridicare",
     *         type="string",
     *         description="",
     *         example="Str. Padurii,
Nr. 85",
     *     ),
     *     @OA\Property(
     *         property="oras_ridicare",
     *         type="string",
     *         description="",
     *         example="Sabareni",
     *     ),
     *     @OA\Property(
     *         property="judet_ridicare",
     *         type="string",
     *         description="",
     *         example="Giurgiu",
     *     )
     * )
     * ),
     * @OA\Response(
     *     response=200,
     *     description="Success"
     * )
     */
    public function store(Request $request):
JsonResponse
    {
        foreach ($request->get('books') as
$book) {
            $data = new Donatie;

            $data->nume = $request-
>get('nume');
            $data->prenume = $request-
>get('prenume');

```

```

        $data->email = $request->get('email');
        $data->nr_telefon = $request->get('nr_telefon');

        $data->titlu = $book['titlu'];
        $data->autor = $book['autor'];
        $data->isbn = $book['isbn'];

        $data->adresa_ridicare = $request->get('adresa_ridicare');
        $data->oras_ridicare = $request->get('oras_ridicare');
        $data->judet_ridicare = $request->get('judet_ridicare');
        $data->save();

    }

    return response()->json("DONE",
Response::HTTP_OK);
    }

/**
 * @return JsonResponse
 */
public function index()
{
    $data = Donatie::all();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @param int $id
 * @return JsonResponse
 */
public function destroy(int $id)
{
    $data = Donatie::where('id', '=',
$id)->first();
    $data->delete();
    return response()->json("DELETED",
Response::HTTP_OK);
}
}

<?php

namespace App\Http\Controllers;

use App\Models\Carte;
use App\Models\Favorit;
use App\Models\User;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;

class FavoritController extends Controller
{
    /**
     * @OA\Post(
     *     path="/favorite",
     *     summary="Adauga cartea favorita",
     *     tags={"Favorite"},
     *     description="Adauga cartea
favorita",
     *     @OA\RequestBody(
     *         required=true,
     *         @OA\JsonContent(
     *             @OA\Property(
     *                 property="id_client",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property(

```

```

     *                 property="id_carte",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             )
     *         )
     *     ),
     *     @OA\Response(
     *         response=200,
     *         description="Success"
     *     )
     */
    public function store(Request $request):
JsonResponse
    {
        if (!Auth::check()) {
            return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
        } else {
            $userLogged = Auth::user();
            $user =
User::findOrFail($userLogged->id);

            if (Favorit::where('id_carte',
$request->get('id_carte'))
->where('id_client', $user-
>client->id)
->exists()) {
                $this->destroy($request-
>get('id_carte'));
                return response()->json('Cartea a
fost stearsa din favorite',
Response::HTTP_OK);
            }

            $data = new Favorit;

            $carte = Carte::find($request-
>get('id_carte'));

            $data->id_carte = $carte->id;
            $data->id_client = $user->client->id;

            $data->save();
            return response()->json($data,
Response::HTTP_OK);
        }
    }

    public function index()
    {
        $userLogged = Auth::user();
        $user = User::findOrFail($userLogged-
>id);

        $data = Favorit::where('id_client',
$user->client->id)->get();
        return response()->json($data,
Response::HTTP_OK);
    }

    public function show($id_carte):
JsonResponse
    {
        $userLogged = Auth::user();
        $user = User::findOrFail($userLogged-
>id);

        $data = Favorit::where('id_carte',
$id_carte)
->where('id_client', $user-
>client->id)
->first();
        if (!empty($data)) {
            return response()->json(['data' =>
true], Response::HTTP_OK);
        } else {

```

```

        return response()->json(['data' =>
false], Response::HTTP_OK);
    }
}

/**
 * @OA\Delete(
 *     path="/favorit/{id}",
 *     summary="Sterge cartea cu id-ul dat
din favorite",
 *     tags={"CarteCos"},
 *     description="Sterge cartea cu id-ul
dat din favorite",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul cartii
dorite",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function destroy($id): JsonResponse
{
    if (Auth::check()) {
        $user =
User::findOrFail(Auth::user()->id);
    } else {
        return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
    }

    $client = $user->client;
    $favorite = $client->favorite()-
>where('id_carte', '=', $id)->first();
    $favorite->delete();

    return response()->json("DELETED",
Response::HTTP_OK);
}
<?php

namespace App\Http\Controllers;

use App\Services\ImportService;
use Exception;
use Illuminate\Http\JsonResponse;

class ImportController extends Controller
{
    private ImportService $importService;
    public function __construct(ImportService
$importService)
    {
        $this->importService = $importService;
    }

    /**
     * @return JsonResponse
     */
    public function importCategorii():
JsonResponse
    {
        try {
            $this->importService-
>import('categorii.csv');
            return response()->json(['message'
=> 'Importul a fost realizat cu succes!']);
        } catch (Exception $e) {

```

```

        return response()->json(['message'
=> $e->getMessage()]);
    }
}

/**
 * @return JsonResponse
 */
public function importCarti():
JsonResponse
{
    try {
        $this->importService-
>import('carti.csv');
        return response()->json(['message'
=> 'Importul a fost realizat cu succes!']);
    } catch (Exception $e) {
        return response()->json(['message'
=> $e->getMessage()]);
    }
}

/**
 * @return JsonResponse
 */
public function importUseri():
JsonResponse
{
    try {
        $this->importService-
>import('useri.csv');
        $this->importService-
>import('clienti.csv');
        return response()->json(['message'
=> 'Importul a fost realizat cu succes!']);
    } catch (Exception $e) {
        return response()->json(['message'
=> $e->getMessage()]);
    }
}

/**
 * @return JsonResponse
 */
public function importReviews():
JsonResponse
{
    try {
        $this->importService-
>import('review.csv');
        return response()->json(['message'
=> 'Importul a fost realizat cu succes!']);
    } catch (Exception $e) {
        return response()->json(['message'
=> $e->getMessage()]);
    }
}

/**
 * @return JsonResponse
 */
public function importDiscounts():
JsonResponse
{
    try {
        $this->importService-
>import('promotii.csv');
        return response()->json(['message'
=> 'Importul a fost realizat cu succes!']);
    } catch (Exception $e) {
        return response()->json(['message'
=> $e->getMessage()]);
    }
}
}
<?php

namespace App\Http\Controllers;

```

```

use App\Models\Returnare;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;

class ReturnareController extends Controller
{
    /**
     * @OA\Post (
     *     path="/returnari",
     *     summary="Adauga returnarea",
     *     tags={"Returnari"},
     *     description="Adauga returnarea",
     *     @OA\RequestBody (
     *         required=true,
     *         @OA\JsonContent (
     *             @OA\Property (
     *                 property="id_carte",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="id_comanda",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="cantitate",
     *                 type="integer",
     *                 description="",
     *                 example=1,
     *             ),
     *             @OA\Property (
     *                 property="subtotal",
     *                 type="float",
     *                 description="",
     *                 example=1.0,
     *             )
     *         )
     *     ),
     *     @OA\Response (
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function store(Request $request):
    JsonResponse
    {
        $data = new Returnare;

        $data->id_carte = $request-
    >get('id_carte');
        $data->id_comanda = $request-
    >get('id_comanda');
        $data->cantitate = $request-
    >get('cantitate');
        $data->subtotal = $request-
    >get('subtotal');

        $data->save();
        return response()->json($data,
    Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Carte;
use App\Models\Client;
use App\Models\Review;
use App\Models\User;
use App\Services\ReviewService;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;

```

```

use Illuminate\Http\Response;
use Illuminate\Support\Carbon;
use Illuminate\Support\Facades\Auth;

class ReviewController extends Controller
{
    private ReviewService $reviewService;
    public function __construct(ReviewService
    $reviewService)
    {
        $this->reviewService = $reviewService;
    }

    /**
     * @OA\Get (
     *     path="/reviewuri",
     *     summary="Returneaza toate review-
    urile",
     *     tags={"Review-uri"},
     *     description="Returneaza toate
    review-urile",
     *     @OA\Response (
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function index(): JsonResponse
    {
        $data = Review::all();
        return response()->json($data,
    Response::HTTP_OK);
    }

    /**
     * @OA\Get (
     *     path="/reviewuri/{id}",
     *     summary="Returneaza review-ul cu id-
    ul dat",
     *     tags={"Review-uri"},
     *     description="Returneaza review-ul cu
    id-ul dat",
     *     @OA\Parameter (
     *         name="id",
     *         description="Id-ul review-ului
    dorit",
     *         required=true,
     *         in="path",
     *         example=3,
     *         @OA\Schema (
     *             type="integer"
     *         )
     *     ),
     *     @OA\Response (
     *         response=200,
     *         description="Success"
     *     )
     * )
     */
    public function show($id): JsonResponse
    {
        $data = Review::where('id', '=', $id)-
    >first();
        return response()->json($data,
    Response::HTTP_OK);
    }

    /**
     * @OA\Put (
     *     path="/reviewuri",
     *     summary="Update review-ul cu id-ul
    dat",
     *     tags={"Review-uri"},
     *     description="Update review-ul cu id-
    ul dat",
     *     @OA\RequestBody (
     *         required=true,
     *         @OA\JsonContent (
     *             @OA\Property (

```

```

*         property="id",
*         type="integer",
*         description="",
*         example=1,
*     ),
*     @OA\Property(
*         property="id_client",
*         type="integer",
*         description="",
*         example=1,
*     ),
*     @OA\Property(
*         property="id_carte",
*         type="integer",
*         description="",
*         example=1,
*     ),
*     @OA\Property(
*         property="comentariu",
*         type="string",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="rating",
*         type="float",
*         description="",
*         example=1.0,
*     ),
*     @OA\Property(
property="data_review",
*         type="date",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="titlu",
*         type="string",
*         description="",
*         example="",
*     )
* )
* )
* /
public function update(Request $request):
JsonResponse
{
    $data = Review::where('id', '=',
$request['id'])->first();
    $this->reviewService-
>setProperties($data, $request->all());
    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Post(
 *     path="/reviewuri",
 *     summary="Adauga review-ul",
 *     tags={"Review-uri"},
 *     description="Adauga review-ul",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="id_client",
 *                 type="integer",
 *                 description="",
 *                 example=1,
 *             ),
 *             @OA\Property(
 *                 property="id_carte",

```

```

*         type="integer",
*         description="",
*         example=1,
*     ),
*     @OA\Property(
*         property="comentariu",
*         type="string",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="rating",
*         type="float",
*         description="",
*         example=1.0,
*     ),
*     @OA\Property(
property="data_review",
*         type="date",
*         description="",
*         example="",
*     ),
*     @OA\Property(
*         property="titlu",
*         type="string",
*         description="",
*         example="",
*     )
* )
* )
* /
* @OA\Response(
*     response=200,
*     description="Success"
* )
* )
* /
public function store(Request $request):
JsonResponse
{
    $userLogged = Auth::user();
    $user = User::where('id', '=',
$userLogged->id)->first();
    $client = $user->client;

    $data = new Review;
    $this->reviewService-
>setProperties($data, $request->all());
    $data->id_client = $client->id;
    $data->data_review = Carbon::now()-
>format('Y-m-d');

    $data->save();
    return response()->json($data,
Response::HTTP_OK);
}

/**
 * @OA\Delete(
 *     path="/reviewuri/{id}",
 *     summary="Sterge review-ul cu id-ul
dat",
 *     tags={"Review-uri"},
 *     description="Sterge review-ul cu id-
ul dat",
 *     @OA\Parameter(
 *         name="id",
 *         description="Id-ul review-ului
dorit",
 *         required=true,
 *         in="path",
 *         example=3,
 *         @OA\Schema(
 *             type="integer"
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"

```

```

        * )
        * )
        */
    public function destroy($id): JsonResponse
    {
        $data = Review::where('id', '=', $id)-
>first();
        $data->delete();
        return response()->json("DELETED",
Response::HTTP_OK);
    }

    /**
     * @param $id
     * @return JsonResponse
     */
    public function getByBookId($id):
JsonResponse
    {
        $data = Review::where('id_carte', '=',
$idid)->get();

        $reviews = [];
        $count = [
            '1' => '0',
            '2' => '0',
            '3' => '0',
            '4' => '0',
            '5' => '0',
        ];

        $carte = Carte::where('id', '=', $id)-
>first();
        $carte->nr_reviewuri = count($data);

        $avg = 0.0;

        foreach ($data as $review) {
            $avg = $avg + $review->rating;
            $client = Client::where('id', '=',
$review->id_client)->first();
            $count[$review->rating]++;
            $reviews[] = [
                'id' => $review->id,
                'id_client' => $review-
>id_client,
                'id_carte' => $review-
>id_carte,
                'comentariu' => $review-
>comentariu,
                'rating' => $review->rating,
                'data_review' => $review-
>data_review,
                'titlu' => $review->titlu,
                'nume_client' => $client-
>nume,
                'prenume_client' => $client-
>prenume,
            ];
        }

        $carte->rating = count($data) != 0 ?
$avg / count($data) : 0.00;
        $carte->save();

        return response()->json(['reviews' =>
$reviews, 'statistics' => $count],
Response::HTTP_OK);
    }
}

<?php

namespace App\Http\Controllers;

use App\Mail\AutentificareEmail;
use App\Mail\ResetareParolaEmail;
use App\Models\Client;
use App\Models\User;

```

```

use App\Services\ClientService;
use App\Services\UserService;
use
Illuminate\Contracts\Foundation\Application;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Mail;
use Illuminate\Support\Str;
use Tymon\JWTAuth\Facades\JWTAuth;

class UserController
{
    private ClientService $clientService;
    private UserService $userService;
    public function __construct(ClientService
$clientService, UserService $userService)
    {
        $this->clientService = $clientService;
        $this->userService = $userService;
    }

    /**
     * @return JsonResponse
     */
    public function index()
    {
        $data = User::all();
        return response()->json($data,
Response::HTTP_OK);
    }

    /**
     * @return JsonResponse
     */
    public function show()
    {
        $user = Auth::user();
        $client = $user->client;
        return response()->json([
            'user' => $user,
            'client' => $client
        ], Response::HTTP_OK);
    }

    /**
     * @OA\Post(
     *     path="/user",
     *     summary="Adauga clientul",
     *     tags={"User"},
     *     description="Adauga clientul",
     *     @OA\RequestBody(
     *         required=true,
     *         @OA\JsonContent(
     *             @OA\Property(
     *                 property="nume",
     *                 type="string",
     *                 description="",
     *                 example="Petrache",
     *             ),
     *             @OA\Property(
     *                 property="prenume",
     *                 type="string",
     *                 description="",
     *                 example="Cristina",
     *             ),
     *             @OA\Property(
     *                 property="email",
     *                 type="string",
     *                 description="",
     *             ),
     *         ),
     *     ),
     *     example="cristina.petrache@hotmail.com",
     * )
     */
    public function create()
    {
        $data = request()->json();
        $user = $this->userService->create($data);
        $client = $this->clientService->create($user);
        return response()->json($client,
Response::HTTP_CREATED);
    }
}

```

```

*         property="parola",
*         type="string",
*         description="",
*         example="asdl23",
*     ),
*     @OA\Property(
*         property="nr_telefon",
*         type="string",
*         description="",
*         example="0755158838",
*     ),
*     @OA\Property(
*         property="adresa",
*         type="string",
*         description="",
*         example="Str. Padurii",
*     ),
*     @OA\Property(
*         property="oras",
*         type="string",
*         description="",
*         example="Sabareni",
*     ),
*     @OA\Property(
*         property="judet",
*         type="string",
*         description="",
*         example="Giurgiu",
*     )
* )
* ),
* @OA\Response(
*     response=200,
*     description="Success"
* )
*/
public function store(Request $request):
JsonResponse
{
    $user = new User;
    $user->email = $request-
>input('email');
    $password = $request->input('parola');

    if (!empty($this->userService-
>checkPassword($password))) {
        return response()->json($this-
>userService->checkPassword($password),
Response::HTTP_NOT_ACCEPTABLE);
    }

    $user->password =
Hash::make($password);

    $user->save();

    $data = new Client;
    $this->clientService-
>setProperties($data, $request->all());
    $data->save();

    $data->user()->associate($user);
    $data->save();
    $user->client()->associate($data);
    $user->save();

    $emailVerificationToken =
Str::random(60);

    DB::table('email_verification')-
>insert([
        'user_id' => $user->id,
        'token' => $emailVerificationToken
    ]);

```

```

        $emailVerificationToken =
'http://127.0.0.1:8000/verify/' .
$emailVerificationToken;

        Mail::to($user->email)->send(new
AuthenticateEmail($user,
$emailVerificationToken));

        $token = JWTAuth::fromUser($user);

        return response()->json([ 'token' =>
$token ], Response::HTTP_OK);
    }

    public function verifyMail(string $token)
    {
        $emailVerification =
DB::table('email_verification')-
>where('token', '=', $token)->first();
        if ($emailVerification) {
            $user =
User::findOrFail($emailVerification->user_id);
            $user->email_verified_at = now();
            $user->save();
            DB::table('email_verification')-
>where('token', '=', $token)->update(['status'
=> 'verified']);
            return response()->json('Email-ul
a fost verificat cu succes!',
Response::HTTP_OK);
        }
        return response()->json('Token-ul nu
exista!', Response::HTTP_NOT_FOUND);
    }

/**
 * @OA\Put(
 *     path="/user/parola",
 *     summary="Update parola user-ului
autenticat",
 *     tags={"User"},
 *     description="Update parola user-ului
autenticat",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="parola",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function changePassword(Request
$request): JsonResponse
{
    if (Auth::check()) {
        $user =
User::findOrFail(Auth::user()->id);
    } else {
        return response()->json('User-ul
nu este autentificat',
Response::HTTP_UNAUTHORIZED);
    }

    if (!empty($this->userService-
>checkPassword($request->input('parola')))) {
        return response()->json($this-
>userService->checkPassword($request-
>input('parola')),
Response::HTTP_NOT_ACCEPTABLE);
    }
}

```

```

        $user->password = Hash::make($request-
>input('parola'));

        $user->save();
        return response()->json("Parola
schimbata!", Response::HTTP_OK);
    }

/**
 * @param int $id
 * @return JsonResponse
 */
public function destroy(int $id)
{
    $user = User::findOrFail($id);
    $user->delete();
    return response()->json('User sters cu
succes!', Response::HTTP_OK);
}

/**
 * @OA\Put(
 *     path="/user",
 *     summary="Update email-ul user-ului
autenticat",
 *     tags={"User"},
 *     description="Update email-ul user-
ului autenticat",
 *     @OA\RequestBody(
 *         required=true,
 *         @OA\JsonContent(
 *             @OA\Property(
 *                 property="email",
 *                 type="string",
 *                 description="",
 *                 example="",
 *             )
 *         )
 *     ),
 *     @OA\Response(
 *         response=200,
 *         description="Success"
 *     )
 * )
 */
public function update(Request $request):
JsonResponse
{
    if (Auth::check()) {
        $user =
User::findOrFail(Auth::user()->id);
    } else {
        return response()->json('User-ul
nu este autenticat',
Response::HTTP_UNAUTHORIZED);
    }

    $user->email = $request-
>input('email');
    $client = $user->client;

    $this->clientService-
>setProperties($client, $request->all());

    $client->save();
    $user->save();
    return response()->json("Email
schimbat!", Response::HTTP_OK);
}

/**
 * @param Request $request
 * @return JsonResponse
 */
public function login(Request $request):
JsonResponse
{
    $validatedData = $request->validate([

```

```

        'email' => 'required|string',
        'password' => 'required|string'
    ]]);

    $user = User::where('email',
$validatedData['email']->first());

    if (!$user) {
        return response()->json(['error'
=> 'User-ul nu exista!'],
Response::HTTP_NOT_FOUND);
    }

    if
(Hash::check($validatedData['password'],
$user->password)) {
        JWTAuth::factory()->setTTL(60 * 24
* 7);

        $token = JWTAuth::fromUser($user);

        return response()->json([
            'user' => $user,
            'token' => $token,
        ]);
    }

    return response()->json(['error' =>
'Unauthorized'], Response::HTTP_UNAUTHORIZED);
}

/**
 * @return JsonResponse
 */
public function logout(): JsonResponse
{
    JWTAuth::invalidate(JWTAuth::getToken());

    return response()->json('Te-ai delogat
cu succes!', Response::HTTP_OK);
}

/**
 * @param Request $request
 * @return void
 */
public function
sendPasswordRequest(Request $request)
{
    $user = User::where('email', '=',
$request->input('email')->first());
    if ($user) {
        $resetPasswordToken =
Str::random(60);
        DB::table('password_resets')->
insert([
            'email' => $user->email,
            'token' => $resetPasswordToken
        ]);
        $resetPasswordToken =
'http://127.0.0.1:8000/reset/' .
$resetPasswordToken;
        Mail::to($user->email)->send(new
ResetareParolaEmail($user,
$resetPasswordToken));
    }
}

/**
 * @param string $token
 * @return Application|Factory|View
 */
public function
resetPasswordRedirect(string $token)
{
    $passwordReset =
DB::table('password_resets')->where('token',
'=', $token)->first();

```



```

        return view('pagina-resetare',
compact('passwordReset'));
    }

    /**
     * @param Request $request
     * @return JsonResponse
     */
    public function resetPassword(Request
$request): JsonResponse
    {
        $user = User::where('email', '=',
$request->input('email'))->first();

        if (!empty($this->userService-
>checkPassword($request->input('parola')))) {
            return response()->json($this-
>userService->checkPassword($request-
>input('parola')),
Response::HTTP_NOT_ACCEPTABLE);
        }

        $user->password = Hash::make($request-
>input('parola'));
        $user->save();

        DB::table('password_resets')-
>where('email', '=', $user->email)-
>update(['status' => 'verified']);

        return response()->json("Parola
schimbata!", Response::HTTP_OK);
    }

    /**
     * @param string $ref
     * @return Application|Factory|View
     */
    public function getProfileRef(string $ref)
    {
        $ref = json_encode($ref);
        return view('pagina-profil',
compact('ref'));
    }
}

```

Anexa 4 – Servicii

```

<?php

namespace App\Services;

use App\Models\CarteComanda;

class CarteComandaService
{
    public function __construct()
    {
        //
    }

    public function setProperties(CarteComanda
$cartecomanda, array $properties): void
    {
        if (!empty($properties['id_carte'])) {
            $cartecomanda->id_carte =
$properties['id_carte'];
        }

        if (!empty($properties['id_comanda']))
        {
            $cartecomanda->id_comanda =

```

```

$properties['id_comanda'];
        }

        if (!empty($properties['cantitate']))
        {
            $cartecomanda->cantitate =
$properties['cantitate'];
        }

        if (!empty($properties['subtotal'])) {
            $cartecomanda->subtotal =
$properties['subtotal'];
        }
    }
}

<?php

namespace App\Services;

use App\Models\Carte;
use App\Models\Categorie;

class CarteService
{
    public function __construct()
    {
        //
    }

    public function setProperties(Carte
$carte, array $properties): void
    {
        if (!empty($properties['titlu'])) {
            $carte->titlu =
$properties['titlu'];
        }

        if (!empty($properties['autor'])) {
            $carte->autor =
$properties['autor'];
        }

        if (!empty($properties['imagine'])) {
            $carte->imagine =
$properties['imagine'];
        }

        if (!empty($properties['isbn'])) {
            $carte->isbn =
$properties['isbn'];
        }

        if (!empty($properties['editura'])) {
            $carte->editura =
$properties['editura'];
        }

        if (!empty($properties['pret'])) {
            $carte->pret =
$properties['pret'];
        }

        if
(!empty($properties['an_aparitie'])) {
            $carte->an_aparitie =
$properties['an_aparitie'];
        }

        if (!empty($properties['nr_pg'])) {
            $carte->nr_pg =
$properties['nr_pg'];
        }

        if
(!empty($properties['tip_coperta'])) {
            $carte->tip_coperta =
$properties['tip_coperta'];
        }
    }
}

```

```

        if (!empty($properties['limba'])) {
            $carte->limba = $properties['limba'];
        }

        if (!empty($properties['dimensiune']))
        {
            $carte->dimensiune =
            $properties['dimensiune'];
        }

        if (!empty($properties['cantitate']))
        {
            $carte->cantitate =
            $properties['cantitate'];
        }

        $carte->save();

        foreach (explode(',',
$properties['categorii']) as $categorii) {
            if (count(explode(',',
$properties['categorii']) > 1) {
                $categorii =
                ltrim($categorii);
            }
            $carte->category($categorii)-
            >attach(Category::where('nume', $categorii)-
            >first());
            $carte->save();
        }
    }
}

```

```

<?php
namespace App\Services;

use App\Models\Category;

class CategoryService
{
    public function __construct()
    {
        //
    }

    public function setProperties(Category
$category, array $properties): void
    {
        if (!empty($properties['nume'])) {
            $category->nume =
            $properties['nume'];
        }

        if (!empty($properties['parent_id']))
        {
            $category->parent_id =
            $properties['parent_id'];
        }
    }
}

<?php
namespace App\Services;

use App\Models\Client;

class ClientService
{
    public function __construct()
    {
        //
    }

    public function setProperties(Client
$client, array $properties): void
    {
        if (!empty($properties['nume'])) {

```

```

            $client->nume =
            $properties['nume'];
        }

        if (!empty($properties['prenume'])) {
            $client->prenume =
            $properties['prenume'];
        }

        if (!empty($properties['nr_telefon']))
        {
            $client->nr_telefon =
            $properties['nr_telefon'];
        }

        if (!empty($properties['adresa'])) {
            $client->adresa =
            $properties['adresa'];
        }

        if (!empty($properties['oras'])) {
            $client->oras =
            $properties['oras'];
        }

        if (!empty($properties['judet'])) {
            $client->judet =
            $properties['judet'];
        }
    }
}

```

```

<?php
namespace App\Services;

use App\Models\Command;
use Illuminate\Support\Carbon;

class CommandService
{
    public function __construct()
    {
        //
    }

    public function setProperties(Command
$command, array $properties): void
    {
        if (!empty($properties['id_client']))
        {
            $command->id_client =
            $properties['id_client'];
        }

        if (!empty($properties['tip'])) {
            $command->tip =
            $properties['tip'];
        }

        $currentDate = Carbon::now();
        $futureDate = $currentDate-
        >addDays(7)->format('Y-m-d');

        $command->data_plasare =
        Carbon::now()->format('Y-m-d');

        $command->data_livrare = $futureDate;

        $command->data_predare = $futureDate;

        if
(!empty($properties['pret_comanda'])) {
            $command->pret_comanda =
            $properties['pret_comanda'];
        }

        if
(!empty($properties['adresa_livrare'])) {

```

```

        $comanda->adresa_livrare =
$properties['adresa_livrare'];
    }

    if
(!empty($properties['oras_livrare'])) {
        $comanda->oras_livrare =
$properties['oras_livrare'];
    }

    if
(!empty($properties['judet_livrare'])) {
        $comanda->judet_livrare =
$properties['judet_livrare'];
    }

    if (!empty($properties['status'])) {
        $comanda->status =
$properties['status'];
    }

    if
(!empty($properties['metoda_plata'])) {
        $comanda->metoda_plata =
$properties['metoda_plata'];
    }
}
<?php

namespace App\Services;

use App\Models\Carte;
use App\Models\Categorie;
use App\Models\Client;
use App\Models\Discount;
use App\Models\Review;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class ImportService
{
    public function import($fileName)
    {
        $filePath = public_path('import/' .
$fileName);
        $file = fopen($filePath, 'r');

        $header = fgetcsv($file);
        while ($row = fgetcsv($file)) {

            $data = array_combine($header,
$row);

            switch ($fileName) {
                case 'useri.csv':
                    $user = new User();
                    $user->email =
$data['email'];
                    $user->password =
Hash::make($data['parola']);
                    $user->save();

                    break;

                case 'clienti.csv':
                    $client = new Client();
                    $client->nume =
$data['nume'];
                    $client->prenume =
$data['prenume'];
                    $client->nr_telefon =
$data['nr_telefon'];
                    $client->adresa =
$data['adresa'];
                    $client->oras =
$data['localitate'];
                    $client->judet =
$data['judet'];

```

```

                    $client->save();

                    $user = User::where(fn
($query) =>
                        $query->where('email',
'like', '%' . $client->nume . '%')
                        -
                        >orWhere('email', 'like', '%' . $client-
>prenume . '%')
                        )->first();

                    $client->user()-
>associate($user);
                    $client->save();

                    $user->client()-
>associate($client);
                    $user->save();

                    break;

                    case 'carti.csv':
                        $carte = new Carte();
                        $carte->titlu =
$data['titlu'];
                        $carte->autor =
$data['autor'];
                        $carte->imagine =
$data['isbn'] . '.jpg';
                        $carte->isbn =
$data['isbn'];
                        $carte->editura =
$data['editura'];
                        $carte->pret =
$data['pret'];
                        $carte->an_aparitie =
$data['an_aparitie'];
                        $carte->nr_pg =
$data['nr_pg'];
                        $carte->tip_coperta =
$data['tip_coperta'];
                        $carte->limba =
$data['limba'];
                        $carte->dimensiune =
$data['dimensiune'];
                        $carte->cantitate =
$data['cantitate'];

                        $carte->save();

                        foreach (explode(',',
$data['categorii_id']) as $categorie) {
                            if (count(explode(',',
$data['categorii_id'])) > 1) {
                                $categorie =
ltrim($categorie);

                                $carte->categoria()-
>attach(Categorie::where('nume', $categorie)-
>first());

                                $carte->save();
                            }
                        }
                        break;

                    case 'categorii.csv':
                        $categorie = new
Categorie();
                        $categorie->nume =
$data['nume'];

                        if ($data['parent_id'] ==
"0") {
                            $categorie->parent_id
= $data['parent_id'];
                        } else {
                            $categorie->parent()-
>associate(Categorie::where('nume', '=',
$data['parent_id'])->first());

```

```

    }

    $categorii->save();

    break;

    case 'review.csv':
        $review = new Review();

        $carte =
Carte::find($data['id_carte']);
        $client =
Client::find($data['id_client']);

        $review->carte()-
>associate($carte);
        $review->client()-
>associate($client);

        $review->comentariu =
$data['comentariu'];
        $review->rating =
$data['rating'];
        $review->data_review =
$data['data_review'];
        $review->titlu =
$data['titlu'];

        $review->save();
        break;

    case 'promotii.csv':
        $discount = new
Discount();

        $discount->id_carte =
$data['id_carte'];
        $discount->promotie =
$data['promotie'];

        $discount->save();
        break;

    }

    fclose($file);
}
<?php

namespace App\Services;

use App\Models\Review;

class ReviewService
{
    public function __construct()
    {
        //
    }

    public function setProperties(Review
$review, array $properties): void
    {
        if (!empty($properties['id_client']))
        {
            $review->id_client =
$properties['id_client'];
        }

        if (!empty($properties['id_carte'])) {
            $review->id_carte =
$properties['id_carte'];
        }

        if (!empty($properties['comentariu']))
        {
            $review->comentariu =
$properties['comentariu'];
        }
    }
}

```

```

        if (!empty($properties['rating'])) {
            $review->rating =
$properties['rating'];
        }

        if
(!empty($properties['data_review'])) {
            $review->data_review =
$properties['data_review'];
        }

        if (!empty($properties['titlu'])) {
            $review->titlu =
$properties['titlu'];
        }
    }
}

<?php

namespace App\Services;

class UserService
{
    /**
     * @param $password
     * @return array
     */
    public function checkPassword($password):
array
    {
        $passwordErrors = [];

        if (strlen($password) < 8) {
            $passwordErrors[] = 'Parola
trebuie sa aiba cel putin 8 caractere';
        }

        if (!preg_match('/[A-Z]/', $password))
        {
            $passwordErrors[] = 'Parola
trebuie sa contina cel putin o majuscula';
        }

        if (!preg_match('/[a-z]/', $password))
        {
            $passwordErrors[] = 'Parola
trebuie sa contina cel putin o minuscula';
        }

        if (!preg_match('/\d/', $password)) {
            $passwordErrors[] = 'Parola
trebuie sa contina cel putin o cifra';
        }

        if (!preg_match('/[!@#%&*()\-
_+={};:;<.>]/', $password)) {
            $passwordErrors[] = 'Parola
trebuie sa contina cel putin un caracter
special';
        }

        return $passwordErrors;
    }

    /**
     * @param $phoneNumber
     * @return array
     */
    public function
checkPhoneNumber($phoneNumber): array
    {
        $phoneNumberErrors = [];

        if (preg_match('/[A-Z]/',
$phoneNumber) ||
preg_match('/[a-z]/',
$phoneNumber) ||

```

```
//      preg_match('/[!@#$$%^&*()\-
_={};:,<.>]\/', $phoneNumber) {
//      $phoneNumberErrors[] = 'Numarul
de telefon nu trebuie sa contina litere sau
caractere speciale';
//      }
//
//      return $phoneNumberErrors;
//      }
}
```

Anexa 5 - Mail

```
<?php

namespace App\Mail;

use App\Models\User;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class AutentificareEmail extends Mailable
{
    use Queueable, SerializesModels;

    protected User $user;

    protected string $url;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct(User $user,
    string $url)
    {
        $this->user = $user;
        $this->url = $url;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->subject('Creare cont
    Ink&Paper')
        ->markdown('emails.welcome', [
            'user' => $this->user,
            'url' => $this->url
        ]);
    }
}

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class ConfirmOrder extends Mailable
{
    use Queueable, SerializesModels;

    protected int $id;
```

```
/**
 * Create a new message instance.
 *
 * @return void
 */
public function __construct(int $id)
{
    $this->id = $id;
}

/**
 * Build the message.
 *
 * @return $this
 */
public function build()
{
    return $this->subject('Confirmare
    comanda')
    ->markdown('emails.confirm-order', [
        'id' => $this->id
    ]);
}

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class ContactMail extends Mailable
{
    use Queueable, SerializesModels;

    protected string $message;

    protected string $email;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct(string
    $message, string $email)
    {
        $this->message = $message;
        $this->email = $email;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->subject('Mesaj nou')
        ->markdown('emails.contact', [
            'message' => $this->message,
            'email' => $this->email
        ]);
    }
}

<?php

namespace App\Mail;

use App\Models\User;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;

class ResetareParolaEmail extends Mailable
```

```

{
    use Queueable, SerializesModels;

    protected User $user;

    protected string $url;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct(User $user,
        string $url)
    {
        $this->user = $user;
        $this->url = $url;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->subject('Resetare parola
            Ink&Paper')
            ->markdown('emails.password-
            reset', [
                'user' => $this->user,
                'url' => $this->url
            ]);
    }
}

```

Anexa 6 – Componente Vue

```

<template>
    <div>
        <div class="container">
            <h4 style="margin-top:
            50px">Carti</h4>
            <button v-if="showData"
            type="button" class="btn btn-primary"
            @click="show()">Ascunde</button>
            <button v-else type="button"
            class="btn btn-primary"
            @click="show()">Arata</button>
            <button type="button" class="btn
            btn-primary"
            @click="changePopupAdd()">Adauga</button>
            </div>
            <table v-if="!!dataLoaded && showData"
            class="table table-striped">
                <thead>
                    <tr>
                        <th scope="col">ID</th>
                        <th scope="col">Titlu</th>
                        <th scope="col">Autor</th>
                        <th scope="col">Imagine</th>
                        <th scope="col">ISBN</th>
                        <th scope="col">Editura</th>
                        <th scope="col">Pret</th>
                        <th scope="col">An
                        aparitie</th>
                        <th scope="col">Numar
                        pagini</th>
                        <th scope="col">Tip
                        coperta</th>
                        <th scope="col">Limba</th>
                        <th
                        scope="col">Dimensiune</th>

```

```

                        <th scope="col">Cantitate</th>
                        <th scope="col">Actiuni</th>
                    </tr>
                </thead>
                <tbody>
                    <tr v-for="carte in carti"
                    :key="carte.id">
                        <th scope="col">{{ carte.id
                        }}</th>
                        <th scope="col">{{ carte.titlu
                        }}</th>
                        <th scope="col">{{ carte.autor
                        }}</th>
                        <th scope="col">{{
                        carte.imagine }}</th>
                        <th scope="col">{{ carte.isbn
                        }}</th>
                        <th scope="col">{{
                        carte.editura }}</th>
                        <th scope="col">{{ carte.pret
                        }}</th>
                        <th scope="col">{{
                        carte.an_aparitie }}</th>
                        <th scope="col">{{ carte.nr_pg
                        }}</th>
                        <th scope="col">{{
                        carte.tip_coperta }}</th>
                        <th scope="col">{{ carte.limba
                        }}</th>
                        <th scope="col">{{
                        carte.dimensiune }}</th>
                        <th scope="col">{{
                        carte.cantitate }}</th>
                        <th scope="col">
                            
                            
                        </th>
                    </tr>
                </tbody>
            </table>
            <div v-if="popupAdd" class="popup-
            container">
                <div class="popup-content">
                    <form>
                        <div class="mb-3">
                            <label for="titlu"
                            class="form-label">Titlu</label>
                            <input type="text"
                            class="form-control" id="titlu" v-
                            model="carte.titlu">
                        </div>
                        <div class="mb-3">
                            <label for="autor"
                            class="form-label">Autor</label>
                            <input type="text"
                            class="form-control" id="autor" v-
                            model="carte.autor">
                        </div>
                        <div class="mb-3">
                            <label for="imagine"
                            class="form-label">Imagine</label>
                            <input type="text"
                            class="form-control" id="imagine" v-
                            model="carte.imagine">
                        </div>
                        <div class="mb-3">
                            <label for="isbn"
                            class="form-label">ISBN</label>
                            <input type="text"
                            class="form-control" id="isbn" v-
                            model="carte.isbn">
                        </div>
                    </form>
                </div>
            </div>

```

```

        <div class="mb-3">
            <label for="editura"
class="form-label">Editura</label>
            <input type="text"
class="form-control" id="editura" v-
model="carte.editura">
        </div>

        <div class="mb-3">
            <label for="pret"
class="form-label">Pret</label>
            <input type="text"
class="form-control" id="pret" v-
model="carte.pret">
        </div>

        <div class="mb-3">
            <label for="an-
apartitie" class="form-label">An
apartitie</label>
            <input type="text"
class="form-control" id="an-apartitie" v-
model="carte.an_apartitie">
        </div>

        <div class="mb-3">
            <label for="nr-pg"
class="form-label">Numar pagini</label>
            <input type="text"
class="form-control" id="nr-pg" v-
model="carte.nr_pg">
        </div>

        <div class="mb-3">
            <label for="coperta"
class="form-label">Tip coperta</label>
            <input type="text"
class="form-control" id="coperta" v-
model="carte.tip_coperta">
        </div>

        <div class="mb-3">
            <label for="limba"
class="form-label">Limba</label>
            <input type="text"
class="form-control" id="limba" v-
model="carte.limba">
        </div>

        <div class="mb-3">
            <label
for="dimensiune" class="form-
label">Dimensiune</label>
            <input type="text"
class="form-control" id="dimensiune" v-
model="carte.dimensiune">
        </div>

        <div class="mb-3">
            <label for="cantitate"
class="form-label">Cantitate</label>
            <input type="text"
class="form-control" id="cantitate" v-
model="carte.cantitate">
        </div>

        <div class="d-flex
justify-content-end">
            <button type="button"
class="btn btn-primary"
@click="addCarte()">Submit</button>
            <button type="button"
class="btn btn-secondary"
@click="changePopupAdd()">Close</button>
        </div>
    </form>
</div>
</div>

```

```

        <div v-if="popupEdit" class="popup-
container">
            <div class="popup-content">
                <form>
                    <div class="mb-3">
                        <label for="titlu"
class="form-label">Titlu</label>
                        <input type="text"
class="form-control" id="titlu" v-
model="carte.titlu">
                    </div>

                    <div class="mb-3">
                        <label for="autor"
class="form-label">Autor</label>
                        <input type="text"
class="form-control" id="autor" v-
model="carte.autor">
                    </div>

                    <div class="mb-3">
                        <label for="imagine"
class="form-label">Imagine</label>
                        <input type="text"
class="form-control" id="imagine" v-
model="carte.imagine">
                    </div>

                    <div class="mb-3">
                        <label for="isbn"
class="form-label">ISBN</label>
                        <input type="text"
class="form-control" id="isbn" v-
model="carte.isbn">
                    </div>

                    <div class="mb-3">
                        <label for="editura"
class="form-label">Editura</label>
                        <input type="text"
class="form-control" id="editura" v-
model="carte.editura">
                    </div>

                    <div class="mb-3">
                        <label for="pret"
class="form-label">Pret</label>
                        <input type="text"
class="form-control" id="pret" v-
model="carte.pret">
                    </div>

                    <div class="mb-3">
                        <label for="an-
apartitie" class="form-label">An
apartitie</label>
                        <input type="text"
class="form-control" id="an-apartitie" v-
model="carte.an_apartitie">
                    </div>

                    <div class="mb-3">
                        <label for="nr-pg"
class="form-label">Numar pagini</label>
                        <input type="text"
class="form-control" id="nr-pg" v-
model="carte.nr_pg">
                    </div>

                    <div class="mb-3">
                        <label for="coperta"
class="form-label">Tip coperta</label>
                        <input type="text"
class="form-control" id="coperta" v-
model="carte.tip_coperta">
                    </div>

                    <div class="mb-3">

```

```

        <label for="limba"
class="form-label">Limba</label>
        <input type="text"
class="form-control" id="limba" v-
model="carte.limba">
    </div>

    <div class="mb-3">
        <label
for="dimensiune" class="form-
label">Dimensiune</label>
        <input type="text"
class="form-control" id="dimensiune" v-
model="carte.dimensiune">
    </div>

    <div class="mb-3">
        <label for="cantitate"
class="form-label">Cantitate</label>
        <input type="text"
class="form-control" id="cantitate" v-
model="carte.cantitate">
    </div>

    <div class="d-flex
justify-content-end">
        <button type="button"
class="btn btn-primary"
@click="editCarte()">Submit</button>
        <button type="button"
class="btn btn-secondary"
@click="changePopupEdit()">Close</button>
    </div>
</form>
</div>
</div>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "CarteTable",
  data() {
    return {
      carti: [],
      adminUser: [],
      adminClient: [],
      dataLoaded: false,
      showData: false,
      popupAdd: false,
      popupEdit: false,
      carte: [
        {
          id: '',
          titlu: '',
          autor: '',
          imagine: '',
          isbn: '',
          editura: '',
          pret: '',
          an_aparitie: '',
          nr_pg: '',
          tip_coperta: '',
          limba: '',
          dimensiune: '',
          cantitate: '',
        }
      ]
    }
  },

```

```

    computed: {
      ...mapGetters(['isAuthenticated']),
    },
    mounted() {
      this.getData();
      this.getCarti();

      Promise.all([this.getData(),
this.getCarti()])
        .then(() => {
          this.checkAdmin();
          this.dataLoaded = true;
        });
    },
    methods: {
      async getData() {
        if (this.isAuthenticated) {
          await axios.get('/user', {
            headers: {
              'Authorization':
'Bearer ' + this.isAuthenticated
            }
          })
            .then(response => {
              this.adminClient =
response.data.client;
              this.adminUser =
response.data.user;
            })
            .catch(error => {
              console.log(error);
            });
        } else {
          window.location.href =
'/autenticare'
        }
      },
      async getCarti() {
        await axios.get('/carte', {
          headers: {
            'Authorization': 'Bearer '
+ this.isAuthenticated
          }
        })
          .then(response => {
            this.carti =
response.data;
          })
          .catch(error => {
            console.log(error);
          });
      },
      checkAdmin() {
        if (this.adminUser.email !==
'inkpaper2023@hotmail.com') {
          alert('Nu aveti acces la
aceasta pagina!');
          window.location.href =
'/autenticare';
        }
      },
      async deleteItem(carte) {
        await axios.delete('/carte/' +
carte.id, {
          headers: {
            'Authorization': 'Bearer '
+ this.isAuthenticated
          }
        })
          .then(response => {
            console.log(response);
          })
          .catch(error => {
            console.log(error);
          });
      },
      show() {
        this.showData = !this.showData;
      },

```



```

changePopupAdd() {
  this.popupAdd = !this.popupAdd;
},
changePopupEdit(carte) {
  this.carte = carte;
  this.popupEdit = !this.popupEdit;
},
async addCarte() {
  await axios.post('/carte', {
    titlu: this.carte.titlu,
    autor: this.carte.autor,
    imagine:
this.carte.imagine,
    isbn: this.carte.isbn,
    editura:
this.carte.editura,
    pret: this.carte.pret,
    an_aparitie:
this.carte.an_aparitie,
    nr_pg: this.carte.nr_pg,
    tip_coperta:
this.carte.tip_coperta,
    limba: this.carte.limba,
    dimensiune:
this.carte.dimensiune,
    cantitate:
this.carte.cantitate,
  },
  {
    headers: {
      'Authorization': 'Bearer '
+ this.isAuthenticated
    }
  })
  .then(response => {
    console.log(response);
  })
  .catch(error => {
    console.log(error);
  });
  this.popupAdd = !this.popupAdd;
},
async editCarte () {
  await axios.put('/carte', {
    id: this.carte.id,
    titlu: this.carte.titlu,
    autor: this.carte.autor,
    imagine:
this.carte.imagine,
    isbn: this.carte.isbn,
    editura:
this.carte.editura,
    pret: this.carte.pret,
    an_aparitie:
this.carte.an_aparitie,
    nr_pg: this.carte.nr_pg,
    tip_coperta:
this.carte.tip_coperta,
    limba: this.carte.limba,
    dimensiune:
this.carte.dimensiune,
    cantitate:
this.carte.cantitate,
  },
  {
    headers: {
      'Authorization':
'Bearer ' + this.isAuthenticated
    }
  })
  .then(response => {
    console.log(response);
  })
  .catch(error => {
    console.log(error);
  });
  this.popupEdit = !this.popupEdit;
}
}

```

```

}
</script>

<style scoped>
.table {
  width: 100%;
}

.btn-primary {
  display: block;
  background-color: #00A896;
  border: none;
  margin-bottom: 10px;
}

.popup-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
}

.popup-content {
  background-color: #fff;
  padding: 20px;
  border-radius: 4px;
}

.container {
  padding-left: 0;
}
</style>

<template>
  <div>
    <div class="container">
      <h4 style="margin-top:
50px">Categorii</h4>
      <button v-if="showData"
type="button" class="btn btn-primary"
@click="show()">Ascunde</button>
      <button v-else type="button"
class="btn btn-primary"
@click="show()">Arata</button>
      <button type="button" class="btn
btn-primary"
@click="changePopupAdd()">Adauga</button>
    </div>
    <table v-if="!!dataLoaded && showData"
class="table table-striped">
      <thead>
        <tr>
          <th scope="col">ID</th>
          <th scope="col">Nume</th>
          <th scope="col">ID
Parinte</th>
          <th scope="col">ID carti</th>
          <th scope="col">Actiuni</th>
        </tr>
      </thead>
      <tbody>
        <tr v-for="categorii in categorii"
:key="categorii.id">
          <th scope="col">{{
categorii.id }}</th>
          <th scope="col">{{
categorii.nume }}</th>
          <th scope="col">{{
categorii.parent_id }}</th>
          <th scope="col"><span v-
for="carte_categorii in categorii.carti">{{
carte_categorii }}, </span></th>
          <th scope="col">

```



```

        this.categorii =
response.data;
    })
    .catch(error => {
        console.log(error);
    });
    },
    async getCategoriiCarti() {
        await axios.get('/carte-
categorie', {
            headers: {
                'Authorization': 'Bearer '
+ this.isAuthenticated
            }
        })
        .then(response => {
            this.carti_categorii =
response.data;
        })
        .catch(error => {
            console.log(error);
        });
    },
    checkAdmin() {
        if (this.adminUser.email !==
'inkpaper2023@hotmail.com') {
            alert('Nu aveti acces la
aceasta pagina!');
            window.location.href =
'/autenticare';
        }
    },
    combineData() {
        this.categorii.forEach(categorie
=> {
            categorie.carti = [];

this.carti_categorii.forEach(carte_categorie
=> {
                if (categorie.id ===
carte_categorie.id_categorie) {

categorie.carti.push(carte_categorie.id_carte)
;

                }
            })
        },
        show() {
            this.showData = !this.showData;
        },
        changePopupAdd() {
            this.popupAdd = !this.popupAdd;
        },
        changePopupEdit(categorie) {
            this.categorie = categorie;
            this.popupEdit = !this.popupEdit;
        },
        async addCategorie() {
            await axios.post('/categorii', {
                nume: this.categorie.nume,
                parent_id:
this.categorie.parent_id,
            },
            {
                headers: {
                    'Authorization': 'Bearer '
+ this.isAuthenticated
                }
            })
            .then(response => {
                console.log(response);
            })
            .catch(error => {
                console.log(error);
            });
            this.popupAdd = !this.popupAdd;
        },
        async addCarte () {

```

```

            await axios.post('/carte-
categorie', {
                id_categorie:
this.categorie.id,
                id_carte: this.id,
            },
            {
                headers: {
                    'Authorization':
'Bearer ' + this.isAuthenticated
                }
            })
            .then(response => {
                console.log(response);
            })
            .catch(error => {
                console.log(error);
            });
            this.popupEdit = !this.popupEdit;
        }
    }
}
</script>

<style scoped>
.table {
    width: 100%;
}

.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
    margin-bottom: 10px;
}

.popup-container {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    display: flex;
    align-items: center;
    justify-content: center;
}

.popup-content {
    background-color: #fff;
    padding: 20px;
    border-radius: 4px;
}

.container {
    padding-left: 0;
}
</style>

<template>
    <div>
        <div class="container">
            <h4 style="margin-top:
50px">Comenzi</h4>
            <button v-if="showData"
type="button" class="btn btn-primary"
@click="show()">Ascunde</button>
            <button v-else type="button"
class="btn btn-primary"
@click="show()">Arata</button>
        </div>
        <table v-if="!!dataLoaded && showData"
class="table table-striped">
            <thead>
                <tr>
                    <th scope="col">ID</th>
                    <th scope="col">ID Client</th>
                    <th scope="col">Tip</th>

```

```

        <th scope="col">Data
plasare</th>
        <th scope="col">Data
livrare</th>
        <th scope="col">Total
comanda</th>
        <th scope="col">Adresa</th>
        <th
scope="col">Localitate</th>
        <th scope="col">Judet</th>
        <th scope="col">Status</th>
        <th scope="col">Metoda
plata</th>
        <th scope="col">Carti</th>
        <th scope="col">Actiuni</th>
    </tr>
</thead>
<tbody>
    <tr v-for="comanda in comenzi"
:key="comanda.id">
        <th scope="col">{{ comanda.id
}}</th>
        <th scope="col">{{
comanda.id_client }}</th>
        <th scope="col">{{ comanda.tip
}}</th>
        <th scope="col">{{
comanda.data_plasare }}</th>
        <th scope="col">{{
comanda.data_livrare }}</th>
        <th scope="col">{{
comanda.pret_comanda }}</th>
        <th scope="col">{{
comanda.adresa_livrare }}</th>
        <th scope="col">{{
comanda.oras_livrare }}</th>
        <th scope="col">{{
comanda.judet_livrare }}</th>
        <th scope="col">{{
comanda.status }}</th>
        <th scope="col">{{
comanda.metoda_plata }}</th>
        <th scope="col"><span v-
for="carte_comanda in comanda.carti">{{
carte_comanda }}, </span></th>
        <th scope="col">
            
        </th>
    </tr>
</tbody>
</table>
<div v-if="popupEdit" class="popup-
container">
    <div class="popup-content">
        <form>
            <div class="mb-3">
                <label for="comanda"
class="form-label">ID comanda </label>
                <input type="text"
class="form-control" id="comanda" v-
model="comanda.id">
            </div>
            <div class="mb-3">
                <label for="status"
class="form-label">Status</label>
                <input type="text"
class="form-control" id="status" v-
model="comanda.status">
            </div>
            <div class="d-flex
justify-content-end">
                <button type="button"
class="btn btn-primary"
@click="updateStatus()">Submit</button>

```

```

                <button type="button"
class="btn btn-secondary"
@click="changePopupEdit()">Close</button>
            </div>
        </form>
    </div>
</div>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: "CarteTable",
    data() {
        return {
            comenzi: [],
            carti_comenzi: [],
            adminUser: [],
            adminClient: [],
            dataLoaded: false,
            showData: false,
            popupEdit: false,
            comanda: [],
        }
    },
    computed: {
        ...mapGetters(['isAuthenticated']),
    },
    mounted() {
        this.getData();
        this.getComenzi();
        this.getComenziCarti();

        Promise.all([this.getData(),
this.getComenzi(), this.getComenziCarti()])
            .then(() => {
                this.checkAdmin();
                this.combineData();
                this.dataLoaded = true;
            });
    },
    methods: {
        async getData() {
            if (this.isAuthenticated) {
                await axios.get('/user', {
                    headers: {
                        'Authorization':
'Bearer ' + this.isAuthenticated
                    }
                })
                .then(response => {
                    this.adminClient =
response.data.client;
                    this.adminUser =
response.data.user;
                })
                .catch(error => {
                    console.log(error);
                });
            } else {
                window.location.href =
'/autenticare'
            }
        },
        async getComenzi() {
            await axios.get('/admin-comenzi',
{
                headers: {
                    'Authorization': 'Bearer '
+ this.isAuthenticated

```

```

    })
    .then(response => {
        this.comenzi =
response.data;
    })
    .catch(error => {
        console.log(error);
    });
},
async getComenziCarti() {
    await axios.get('/carti-comanda',
{
    headers: {
        'Authorization': 'Bearer '
+ this.isAuthenticated
    }
    })
    .then(response => {
        this.carti_comenzi =
response.data;
    })
    .catch(error => {
        console.log(error);
    });
},
checkAdmin() {
    if (this.adminUser.email !==
'inkpaper2023@hotmail.com') {
        alert('Nu aveti acces la
aceasta pagina!');
        window.location.href =
'/autentificare';
    }
},
combineData() {
    this.comenzi.forEach(comanda => {
        comanda.carti = [];

this.carti_comenzi.forEach(carte_comanda => {
    if (comanda.id ===
carte_comanda.id) {
        comanda.carti.push(carte_comanda.id);
    }
    })
    },
show() {
    this.showData = !this.showData;
},
changePopupEdit(comanda) {
    this.comanda = comanda;
    this.popupEdit = !this.popupEdit;
},
async updateStatus () {
    await axios.put('/admin-comenzi',
{
        id: this.comanda.id,
        status:
this.comanda.status,
    },
    {
        headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
        }
    })
    .then(response => {
        console.log(response);
    })
    .catch(error => {
        console.log(error);
    });
    this.popupEdit = !this.popupEdit;
}
}

```

```

}
</script>

<style scoped>
.table {
    width: 100%;
}

.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
    margin-bottom: 10px;
}

.popup-container {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    display: flex;
    align-items: center;
    justify-content: center;
}

.popup-content {
    background-color: #fff;
    padding: 20px;
    border-radius: 4px;
}

.container {
    padding-left: 0;
}
</style>

<template>
<div>
    <div class="container">
        <h4 style="margin-top:
50px">Donatii</h4>
        <button v-if="showData"
type="button" class="btn btn-primary"
@click="show()">Ascunde</button>
        <button v-else type="button"
class="btn btn-primary"
@click="show()">Arata</button>
    </div>
    <table v-if="!!dataLoaded && showData"
class="table table-striped">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Nume</th>
                <th scope="col">Prenume</th>
                <th scope="col">Email</th>
                <th scope="col">Numar
telefon</th>
                <th scope="col">Titlu</th>
                <th scope="col">Autor</th>
                <th scope="col">ISBN</th>
                <th scope="col">Adresa</th>
                <th
scope="col">Localitate</th>
                <th scope="col">Judet</th>
                <th scope="col">Actiuni</th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="donatie in donatii"
:key="donatie.id">
                <th scope="col">{{ donatie.id
}}</th>
                <th scope="col">{{
donatie.nume }}</th>
                <th scope="col">{{
donatie.prenume }}</th>

```

```

        <th scope="col">{{
donatie.email }}</th>
        <th scope="col">{{
donatie.nr_telefon }}</th>
        <th scope="col">{{
donatie.titlu }}</th>
        <th scope="col">{{
donatie.autor }}</th>
        <th scope="col">{{
donatie.isbn }}</th>
        <th scope="col">{{
donatie.adresa_ridicare }}</th>
        <th scope="col">{{
donatie.oras_ridicare }}</th>
        <th scope="col">{{
donatie.judet_ridicare }}</th>
        <th scope="col">
            
        </th>
    </tr>
</tbody>
</table>
</div>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "CarteTable",
  data() {
    return {
      donatii: [],
      adminUser: [],
      adminClient: [],
      dataLoaded: false,
      showData: false,
    }
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getData();
    this.getDonatii();

    Promise.all([this.getData(),
this.getDonatii()])
      .then(() => {
        this.checkAdmin();
        this.dataLoaded = true;
      });
  },
  methods: {
    async getData() {
      if (this.isAuthenticated) {
        await axios.get('/user', {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        })
        .then(response => {
          this.adminClient =
response.data.client;
          this.adminUser =
response.data.user;
        })
        .catch(error => {
          console.log(error);

```

```

        });
      } else {
        window.location.href =
'/autentificare'
      }
    },
    async getDonatii() {
      await axios.get('/donatie', {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        }
      })
      .then(response => {
        this.donatii =
response.data;
      })
      .catch(error => {
        console.log(error);
      });
    },
    checkAdmin() {
      if (this.adminUser.email !==
'inkpaper2023@hotmail.com') {
        alert('Nu aveti acces la
aceasta pagina!');
        window.location.href =
'/autentificare';
      }
    },
    async deleteItem(donatie) {
      await axios.delete('/donatie/' +
donatie.id, {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        }
      })
      .then(response => {
        console.log(response);
      })
      .catch(error => {
        console.log(error);
      });
    },
    show() {
      this.showData = !this.showData;
    },
  }
}
</script>

<style scoped>
.table {
  width: 100%;
}

.btn-primary {
  display: block;
  background-color: #00A896;
  border: none;
  margin-bottom: 10px;
}

.container {
  padding-left: 0;
}
</style>

<template>
  <div>
    <div class="container">
      <h4 style="margin-top:
50px">Reviewuri</h4>
      <button v-if="showData"
type="button" class="btn btn-primary"
@click="show()">Ascunde</button>

```

```

        <button v-else type="button"
class="btn btn-primary"
@click="show()">Arata</button>
    </div>
    <table v-if="!!dataLoaded && showData"
class="table table-striped">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">ID Client</th>
                <th scope="col">ID Carte</th>
                <th
scope="col">Comentariu</th>
                <th scope="col">Rating</th>
                <th scope="col">Data</th>
                <th scope="col">Titlu</th>
                <th scope="col">Actiuni</th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="review in reviwuri"
:key="review.id">
                <th scope="col">{{ review.id
}}</th>
                <th scope="col">{{
review.id_client }}</th>
                <th scope="col">{{
review.id_carte }}</th>
                <th scope="col">{{
review.comentariu }}</th>
                <th scope="col">{{
review.rating }}</th>
                <th scope="col">{{
review.data_review }}</th>
                <th scope="col">{{
review.titlu }}</th>
                <th scope="col">
                    
                    
                </th>
            </tr>
        </tbody>
    </table>
    <div v-if="popupEdit" class="popup-
container">
        <div class="popup-content">
            <form>
                <div class="mb-3">
                    <label for="review"
class="form-label">ID</label>
                    <input type="text"
class="form-control" id="review" v-
model="review.id">
                </div>
                <div class="mb-3">
                    <label for="client"
class="form-label">ID Client</label>
                    <input type="text"
class="form-control" id="client" v-
model="review.id_client">
                </div>
                <div class="mb-3">
                    <label for="carte"
class="form-label">ID Carte</label>
                    <input type="text"
class="form-control" id="carte" v-
model="review.id_carte">
                </div>
                <div class="mb-3">
                    <label
for="comentariu" class="form-
label">Comentariu</label>

```

```

                    <input type="text"
class="form-control" id="comentariu" v-
model="review.comentariu">
                </div>
            </div>
            <div class="mb-3">
                <label for="rating"
class="form-label">Rating</label>
                <input type="text"
class="form-control" id="rating" v-
model="review.rating">
            </div>
            <div class="mb-3">
                <label for="data"
class="form-label">Data</label>
                <input type="text"
class="form-control" id="data" v-
model="review.data_review">
            </div>
            <div class="mb-3">
                <label for="titlu"
class="form-label">Titlu</label>
                <input type="text"
class="form-control" id="titlu" v-
model="review.titlu">
            </div>
            <div class="d-flex
justify-content-end">
                <button type="button"
class="btn btn-primary"
@click="editReview()">Submit</button>
                <button type="button"
class="btn btn-secondary"
@click="changePopupEdit()">Close</button>
            </div>
        </form>
    </div>
</div>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: "CarteTable",
    data() {
        return {
            reviwuri: [],
            adminUser: [],
            adminClient: [],
            dataLoaded: false,
            showData: false,
            popupEdit: false,
            review: [],
        }
    },
    computed: {
        ...mapGetters(['isAuthenticated']),
    },
    mounted() {
        this.getData();
        this.getReviews();

        Promise.all([this.getData(),
this.getReviews()])
            .then(() => {
                this.checkAdmin();
                this.dataLoaded = true;
            })
    }
}

```

```

    });
  },
  methods: {
    async getData() {
      if (this.isAuthenticated) {
        await axios.get('/user', {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        })
        .then(response => {
          this.adminClient =
response.data.client;
          this.adminUser =
response.data.user;
        })
        .catch(error => {
          console.log(error);
        });
      } else {
        window.location.href =
'/autenticare'
      }
    },
    async getReviews() {
      await axios.get('/reviewuri', {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        }
      })
      .then(response => {
        this.reviwuri =
response.data;
      })
      .catch(error => {
        console.log(error);
      });
    },
    checkAdmin() {
      if (this.adminUser.email !==
'inkpaper2023@hotmail.com') {
        alert('Nu aveti acces la
aceasta pagina!');
        window.location.href =
'/autenticare';
      }
    },
    async deleteItem(review) {
      await axios.delete('/reviewuri/' +
review.id, {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        }
      })
      .then(response => {
        console.log(response);
      })
      .catch(error => {
        console.log(error);
      });
    },
    show() {
      this.showData = !this.showData;
    },
    changePopupEdit(review) {
      this.review = review;
      this.popupEdit = !this.popupEdit;
    },
    async editReview() {
      await axios.put('/reviewuri', {
        id: this.review.id,
        id_client:
this.review.id_client,
        id_carte:
this.review.id_carte,

```

```

        comentariu:
this.review.comentariu,
        rating:
this.review.rating,
        data_review:
this.review.data_review,
        titlu: this.review.titlu,
      },
      {
        headers: {
          'Authorization':
'Bearer ' + this.isAuthenticated
        }
      })
      .then(response => {
        console.log(response);
      })
      .catch(error => {
        console.log(error);
      });
    },
    this.popupEdit = !this.popupEdit;
  }
}
</script>

<style scoped>
.table {
  width: 100%;
}

.btn-primary {
  display: block;
  background-color: #00A896;
  border: none;
  margin-bottom: 10px;
}

.popup-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
}

.popup-content {
  background-color: #fff;
  padding: 20px;
  border-radius: 4px;
}

.container {
  padding-left: 0;
}
</style>

<template>
  <div>
    <div class="container">
      <h4 style="margin-top:
100px">Clienti</h4>
      <button v-if="showData"
type="button" class="btn btn-primary"
@click="show()">Ascunde</button>
      <button v-else type="button"
class="btn btn-primary"
@click="show()">Arata</button>
    </div>
    <table v-if="!!dataLoaded && showData"
class="table table-striped">
      <thead>
        <tr>
          <th scope="col">User
ID</th>

```



```

        <th scope="col">Client
ID</th>
        <th scope="col">Email</th>
        <th scope="col">Email
verificat</th>
        <th scope="col">Nume</th>
        <th
scope="col">Prenume</th>
        <th scope="col">Numar
telefon</th>
        <th
scope="col">Adresa</th>
        <th
scope="col">Localitate</th>
        <th scope="col">Judet</th>
        <th
scope="col">Actiuni</th>
      </tr>
    </thead>
    <tbody>
      <tr v-for="user in users"
:key="user.id">
        <th scope="col">{{ user.id
}}</th>
        <th scope="col">{{
user.client_id }}</th>
        <th scope="col">{{
user.email }}</th>
        <th scope="col">{{
user.email_verified_at }}</th>
        <th scope="col">{{
user.client.nume }}</th>
        <th scope="col">{{
user.client.prenume }}</th>
        <th scope="col">{{
user.client.nr_telefon }}</th>
        <th scope="col">{{
user.client.adresa }}</th>
        <th scope="col">{{
user.client.oras }}</th>
        <th scope="col">{{
user.client.judet }}</th>
        <th scope="col"></th>
      </tr>
    </tbody>
  </table>
</div>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "UserTable",
  data() {
    return {
      users: [],
      clients: [],
      adminUser: [],
      adminClient: [],
      dataLoaded: false,
      showData: false,
    }
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getData();

```

```

    this.getUsers();
    this.getClients();

    Promise.all([this.getData(),
this.getUsers(), this.getClients()])
      .then(() => {
        this.checkAdmin();

        this.combineData();
      });

    Promise.all([this.deleteItem()])
      .then(() => {
        this.getUsers();
        this.getClients();
      });
  },
  methods: {
    async getData() {
      if (this.isAuthenticated) {
        await axios.get('/user', {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        })
          .then(response => {
            this.adminClient =
response.data.client;
            this.adminUser =
response.data.user;
          })
          .catch(error => {
            console.log(error);
          });
      } else {
        window.location.href =
'/autenticare'
      }
    },
    async getUsers() {
      await axios.get('/user/index', {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        }
      })
        .then(response => {
          this.users =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
    },
    async getClients() {
      await axios.get('/clienti', {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        }
      })
        .then(response => {
          this.clients =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
    },
    checkAdmin() {
      if (this.adminUser.email !==
'inkpaper2023@hotmail.com') {
        alert('Nu aveti acces la
aceasta pagina!');
        window.location.href =
'/autenticare';
      }
    },

```

```

        combineData() {
            this.users.forEach(user => {
                this.clients.forEach(client => {
                    if (user.id ===
client.user_id) {
                        user.client = client;
                    }
                });
            });
            this.dataLoaded = true;
        },
        async deleteItem(user) {
            await axios.delete('/clienti/' +
user.client.id, {
                headers: {
                    'Authorization': 'Bearer '
+ this.isAuthenticated
                }
            })
                .then(response => {
                    console.log(response);
                })
                .catch(error => {
                    console.log(error);
                });

            await axios.delete('/user/' +
user.id, {
                headers: {
                    'Authorization': 'Bearer '
+ this.isAuthenticated
                }
            })
                .then(response => {
                    console.log(response);
                })
                .catch(error => {
                    console.log(error);
                });
        },
        show() {
            this.showData = !this.showData;
        },
    }
}
</script>

<style scoped>
.table {
    width: 100%;
}

.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
    margin-bottom: 10px;
}

.container {
    padding-left: 0;
}
</style>

<template>
    <div>
        <h4 class="titlu"><strong>{{
this.title }}</strong></h4>
        <carousel :per-page="5" :navigation-
enabled="true">
            <slide v-for="book in books"
:key="book.id">
                <div class="card-wrapper">
                    <div ref="card"
class="card mb-2">
                        

```

```

                        <span v-
if="isDiscounted(book)" class="position-
absolute top-0 start-100 translate-middle
badge rounded-pill bg-danger" style="margin-
top: 10px">
                            {{ '-'
+mappedDiscountedBooks[book.id] + '%' }}
                        </span>
                    <div class="card-
body">
                        <div class="card">
                            <div
class="card-body d-flex">
                                <h5
class="flex-grow-1 card-title"
@click="redirectBook(book)">{{ book.titlu }} -
{{ book.autor }}</h5>
                                <div
style="margin: 10px">
                                    
                                    
                                </div>
                            </div>
                        <div class="d-
flex">
                            
                            <p
class="card-review">{{ book.rating }} ({{
book.nr_reviewuri }})</p>
                            </div>
                            <p v-
if="!isDiscounted(book)" class="card-text
card-price">{{ book.pret.toFixed(2) }} Lei</p>
                            <p v-else
class="card-text card-price">{{
(book.pret.toFixed(2) -
((mappedDiscountedBooks[book.id] * book.pret)
/ 100)).toFixed(2) }} Lei <span class="text-
decoration-line-through reduced-price">{{
book.pret.toFixed(2) }} Lei</span></p>
                            <a v-
if="book.cantitate !== 0"
@click="addToCart(book)" class="btn btn-
primary">Adauga in cos</a>
                            <a v-else
class="btn btn-primary disabled"
style="background-color: rgba(23, 42, 58,
0.6)">Adauga in cos</a>
                        </div>
                    </div>
                </slide>
            </carousel>
        </div>
    </template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
import { Carousel, Slide } from 'vue-
carousel';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: 'BookCarousel',

```

```

components: {
  Carousel,
  Slide
},
props: ['books', 'title'],
data() {
  return {
    favorite: [],
    discountedBooks: [],
    mappedDiscountedBooks: [],
  }
},
computed: {
  ...mapGetters(['isAuthenticated']),
},
mounted() {
  this.getFavorite();
  this.getDiscounted();
},
methods: {
  addFavorite(carte) {
    if (this.isAuthenticated === null)
    {
      window.location.href =
'/autentificare';
    }
    axios.post('/favorite',
    {
      id_carte: carte.id,
    },
    {
      headers: {
        'Authorization':
'Bearer ' + this.isAuthenticated
      }
    })
    .then(response => {
      this.getFavorite();
    })
    .catch(error => {
      console.log(error);
    });
  },
  checkFavorite(carte) {
    if (this.isAuthenticated === null)
return false;

    for (let i = 0; i <
this.favorite.length; i++) {
      if (this.favorite[i].id_carte
=== carte.id) {
        return true;
      }
    }
    return false;
  },
  getFavorite() {
    if (this.isAuthenticated === null)
return;

    axios.get('/favorite', {
      headers: {
        'Authorization': 'Bearer '
+ this.isAuthenticated
      }
    })
    .then(response => {
      this.favorite =
response.data;
    })
    .catch(error => {
      console.log(error);
    });
  },
  redirectBook(carte) {
    window.location.href = '/carte/' +
carte.isbn;
  },
  addToCart(book) {
    if (this.isAuthenticated) {

```

```

      axios.post('/cos', {
        id_carte: book.id,
        cantitate: 1,
        subtotal:
this.isDiscounted(book) ?
(book.pret.toFixed(2) -
((this.mappedDiscountedBooks[book.id] *
book.pret) / 100)).toFixed(2) :
book.pret.toFixed(2),
      },
      {
        headers: {
          'Authorization':
'Bearer ' + this.isAuthenticated
        }
      })
    .then(response => {
      console.log(response.data);
    })
    .catch(error => {
      console.log(error);
    })
    else {
      axios.post('/cos', {
        id_carte: book.id,
        cantitate: 1,
        subtotal:
this.isDiscounted(book) ?
(book.pret.toFixed(2) -
((this.mappedDiscountedBooks[book.id] *
book.pret) / 100)).toFixed(2) :
book.pret.toFixed(2),
      })
    .then(response => {
      console.log(response.data);
    })
    .catch(error => {
      console.log(error);
    })
  },
  getDiscounted() {
    axios.get('/discounts')
    .then(response => {
      this.discountedBooks =
response.data;
      this.discountedBooks.forEach(book => {
        this.mappedDiscountedBooks[book.id_carte] =
book.promotie;
      })
    })
    .catch(error => {
      console.log(error);
    })
  },
  isDiscounted(book) {
    for (let i = 0; i <
this.discountedBooks.length; i++) {
      if
(this.discountedBooks[i].id_carte === book.id)
{
        return true;
      }
    }
    return false;
  },
  checkAvailability(book) {
    if (book.cantitate > 0) {
      if (book.cantitate < 5) {
        this.availability =
'Ultimul exemplar';
      } else {
        this.availability = 'In
stoc';
      }
    }
    else {

```

```

        this.availability =
'Indisponibil';
    }
    },
}
</script>

<style scoped>
@font-face {
    font-family: 'Lora';
    src: url('/Fonts/static/Lora-
Regular.ttf');
}

.card-wrapper {
    width: 200px;
    margin-right: 5px;
    margin-left: 5px;
    margin-bottom: 10px;
    display: inline-block;
    vertical-align: top;
}

.titlu {
    font-family: "Lora", serif;
    color: #333333;
    margin-left: 5px;
    margin-bottom: 30px;
}

.card {
    width: 100%;
    border: none;
    background-color: #FAFAFA;
}

.card-img-top {
    width: 100%;
    height: auto;
}

.card-body {
    padding: 0;
}

.card-title {
    display: -webkit-box;
    -webkit-box-orient: vertical;
    -webkit-line-clamp: 3;
    overflow: hidden;
    font-size: 18px;
    height: 70px;
    padding-top: 5px;
    font-family: "Lora", serif;
    color: #333333;
}

.card-price {
    display: -webkit-box;
    -webkit-box-orient: vertical;
    font-size: 20px;
    overflow: hidden;
    font-family: "Lora", serif;
    color: #333333;
}

.card-review {
    margin-bottom: 0;
    margin-left: 0;
    font-size: 16px;
    font-family: "Lora", serif;
}

.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
    margin-top: 1rem;

```

```

}

.reduced-price {
    font-size: 16px;
    margin-left: 10px;
    color: rgba(23, 42, 58, 0.6)
}
</style>

<template>
    <div class="d-flex justify-content-center"
style="margin-top: 100px">
        <div class="card mb-3" style="width:
100%">
            <div class="row">
                <div class="col-md-auto">
                    
                    <span v-
if="isDiscounted(book)" class="position-
absolute top-0 translate-middle badge
rounded-pill bg-danger" style="margin-top:
10px">
                        {{ '-'
+mappedDiscountedBooks[book.id] + '%' }}
                    </span>
                </div>
                <div class="col">
                    <div class="card-body">
                        <h4 class="card-
title">{{ book.titlu }}</h4>
                        <div class="container
d-flex align-content-center justify-content-
start" style="margin-bottom: 10px; padding-
left: 0">
                            
                            <p class="card-
text">{{ book.rating }} ({{ book.nr_reviewuri
}})</p>
                        </div>
                        <p class="card-
text">Autor: {{ book.autor}} </p>
                        <p class="card-
text">Editura: {{ book.editura }} </p>
                        <div class="container
d-flex align-content-center" style="padding-
left: 0">
                            <p class="card-
text">Categorie:&nbsp;<span class="card-text"
v-for="(categorie) in categorii"> {{
categorie.nume }}</span></p>
                        </div>
                        <p class="card-text"
style="margin-top: 10px">Anul aparitiei: {{
book.an_aparitie }} </p>
                        <p class="card-
text">Numar pagini: {{ book.nr_pg }} </p>
                        <p class="card-
text">ISBN: {{ book.isbn }} </p>
                        <p class="card-
text">Tip coperta: {{ book.tip_coperta }} </p>
                        <p class="card-
text">Limba: {{ book.limba }} </p>
                        <p class="card-
text">Dimensiuni: {{ book.dimensiune }} </p>
                    </div>
                </div>
            </div>
            <div class="col">
                <div class="card-body
price">
                    <h5 v-
if="!isDiscounted(book)" class="card-title
card-price">Pret: {{ book.pret.toFixed(2) }}
Lei</h5>

```

```

        <h5 v-else
class="card-title card-price">Pret: {{
(book.pret.toFixed(2) -
(mappedDiscountedBooks[book.id] * book.pret)
/ 100)).toFixed(2) }} Lei <span class="text-
decoration-line-through reduced-price">{{
book.pret.toFixed(2) }} Lei</span></h5>

        <p class="card-
text">Stoc: {{availability}}</p>
        <a v-if="availability
!== 'Indisponibil'" @click="addToCart(book)"
class="btn btn-primary">Adauga in cos</a>
        <a v-else class="btn
btn-primary disabled" style="background-color:
rgba(23, 42, 58, 0.6)">Adauga in cos</a>
        <a v-
if="!checkFavorite(book)"
@click="addFavorite(book)" class="btn btn-
primary-outline">Adauga la favorite</a>
        <a v-else
@click="addFavorite(book)" class="btn btn-
primary-full">Sterge de la favorite</a>
        </div>
    </div>
</div>
<!-- <book-carousel
:books="bookList"></book-carousel>-->
</div>
</template>

<script>
import BookCarousel from "../BookCarousel.vue";
import axios from 'axios';
import {mapGetters} from "vuex";
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "BookDetails",
  components: {
    BookCarousel,
  },
  props: ['book'],
  data() {
    return {
      availability: null,
      bookList: null,
      favorite: [],
      categorii: '',
      discountedBooks: [],
      mappedDiscountedBooks: [],
    }
  },
  mounted() {
    this.getCategorii();
    this.checkAvailability();
    this.getRecommandations();
    this.getFavorite();
    this.getDiscounted();
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  methods: {
    checkAvailability() {
      if (this.book.cantitate > 0) {
        if (this.book.cantitate < 5) {
          this.availability =
'Ultimele exemplare';
        } else {
          this.availability = 'In
stoc';
        }
      }
    }
  }
}

```

```

    } else {
      this.availability =
'Indisponibil';
    }
  },
  getRecommandations() {
    axios.get('/carti/recomandari/' +
this.book.id)
      .then(response => {
        console.log(response.data);
        this.bookList =
response.data;
        this.$emit('bookList',
this.bookList);
      })
      .catch(error => {
        console.log(error);
      })
  },
  checkFavorite(carte) {
    if (this.isAuthenticated === null)
return false;

    for (let i = 0; i <
this.favorite.length; i++) {
      if (this.favorite[i].id_carte
=== carte.id) {
        return true;
      }
    }
    return false;
  },
  getFavorite() {
    if (this.isAuthenticated === null)
return;
    axios.get('/favorite', {
      headers: {
        'Authorization': 'Bearer '
+ this.isAuthenticated
      }
    })
      .then(response => {
        this.favorite =
response.data;
      })
      .catch(error => {
        console.log(error);
      });
  },
  addFavorite(carte) {
    if (this.isAuthenticated === null)
{
      window.location.href =
'/autenticare';
    }
    axios.post('/favorite',
{
      id_carte: carte.id,
    },
    {
      headers: {
        'Authorization':
'Bearer ' + this.isAuthenticated
      }
    })
      .then(response => {
        this.getFavorite();
      })
      .catch(error => {
        console.log(error);
      });
  },
  getCategorii() {
    axios.get('/carti/categorii/' +
this.book.id)
      .then(response => {
        this.categorii =
response.data;
      })
  }
}

```

```

    })
    .catch(error => {
      console.log(error);
    })
  },
  addToCart(book) {
    if (this.isAuthenticated) {
      axios.post('/cos', {
        id_carte: book.id,
        cantitate: 1,
        subtotal:
this.isDiscounted(book) ?
(book.pret.toFixed(2) -
((this.mappedDiscountedBooks[book.id] *
book.pret) / 100)).toFixed(2) :
book.pret.toFixed(2),
      },
      {
        headers: {
          'Authorization':
'Bearer ' + this.isAuthenticated
        }
      })
      .then(response => {

console.log(response.data);
      })
      .catch(error => {
        console.log(error);
      })
      .else {
        axios.post('/cos', {
          id_carte: book.id,
          cantitate: 1,
          subtotal:
this.isDiscounted(book) ?
(book.pret.toFixed(2) -
((this.mappedDiscountedBooks[book.id] *
book.pret) / 100)).toFixed(2) :
book.pret.toFixed(2),
        })
        .then(response => {

console.log(response.data);
        })
        .catch(error => {
          console.log(error);
        })
      }
    },
    getDiscounted() {
      axios.get('/discounts')
      .then(response => {
        this.discountedBooks =
response.data;

this.discountedBooks.forEach(book => {

this.mappedDiscountedBooks[book.id_carte] =
book.promotie;
      })
    })
    .catch(error => {
      console.log(error);
    })
  },
  isDiscounted(book) {
    for (let i = 0; i <
this.discountedBooks.length; i++) {
      if
(this.discountedBooks[i].id_carte === book.id)
      {
        return true;
      }
    }
    return false;
  },
},
}
}
</script>

```

```

<style scoped>
.price {
  border-left: rgba(0, 0, 0, 0.1) 1px solid;
  height: 100%;
}

.card {
  border: none;
  background-color: #FAFAFA;
}

.card-title {
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: 2;
  overflow: hidden;
  font-weight: 700;
  height: 50px;
  padding-top: 5px;
  font-family: "Lora", serif;
  color: #333333;
}

.card-text {
  display: -webkit-box;
  -webkit-box-orient: vertical;
  overflow: hidden;
  font-size: 18px;
  font-family: "Lora", serif;
  color: #333333;
}

.btn-primary {
  font-family: "Lora", serif;
  display: block;
  background-color: #00A896;
  border: none;
}

.btn-primary-outline {
  font-family: "Lora", serif;
  display: block;
  background-color: transparent;
  border: 1px solid #FFB6B9;
  margin-top: 10px;
  color: #FFB6B9
}

.btn-primary-full {
  font-family: "Lora", serif;
  display: block;
  background-color: #FFB6B9;
  border: none;
  margin-top: 10px;
  color: #fff
}

.reduced-price {
  font-size: 16px;
  margin-left: 10px;
  color: rgba(23, 42, 58, 0.6)
}
</style>

<template>
  <div class="container-fluid">

    <div class="ctg-title" style="margin-
top: 0; padding-top: 20px">
      <span>Sorteaza</span>
    </div>
    <div class="container-fluid ctg"
:style="{ height: 120 + 'px' }">
      <div class="items" v-for="order in
orders">

        <input type="radio"
:value="order.order" v-model="selectedOrder"
@change="applyFilters">

```

```

        <label :for="order.id">{{
order.name }}</label>
    </div>
</div>

    <div class="ctg-title">
        <span>Pret</span>
    </div>
    <div class="container-fluid ctg"
:style="{ height: 90 + 'px' }">
        <div class="items" v-for="price in
prices">
            <input type="radio"
:value="price.pret" v-model="selectedPrice"
@change="applyFilters">
                <label :for="price.id">{{
price.interval }}</label>
            </div>

            <div class="ctg-title">
                <span>Categorii</span>
            </div>
            <div class="container-fluid ctg"
:style="{ height: 300 + 'px' }">
                <div class="items" v-for="category
in categories">
                    <input type="checkbox"
:value="category.num" v-
model="selectedCategories"
@change="applyFilters">
                        <label :for="category.id">{{
category.num }}</label>
                    </div>

                    <div class="ctg-title">
                        <span>Autori</span>
                    </div>
                    <div class="container-fluid ctg"
:style="{ height: 300 + 'px' }">
                        <div class="items" v-for="author
in authors">
                            <input type="checkbox"
:value="author.autor" v-
model="selectedAuthors"
@change="applyFilters">
                                <label :for="author.id">{{
author.autor }}</label>
                            </div>

                            <div class="ctg-title">
                                <span>Edituri</span>
                            </div>
                            <div class="container-fluid ctg"
:style="{ height: 300 + 'px' }">
                                <div class="items" v-for="publish
in publishing">
                                    <input type="checkbox"
:value="publish.editura" v-
model="selectedPublishing"
@change="applyFilters">
                                        <label :for="publish.id">{{
publish.editura }}</label>
                                    </div>

                                    <div class="ctg-title">
                                        <span>Limbi</span>
                                    </div>
                                    <div class="container-fluid ctg"
:style="{ height: 90 + 'px' }">
                                        <div class="items" v-for="language
in languages">
                                            <input type="checkbox"
:value="language.limba" v-
model="selectedLanguages"
@change="applyFilters">

```

```

        <label :for="language.id">{{
language.limba }}</label>
    </div>
</div>
</template>

<script>
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: "BookFilter",
    data() {
        return {
            prices: null,
            categories: null,
            authors: null,
            publishing: null,
            languages: null,
            orders: null,

            categoriesExpanded: false,
            authorsExpanded: false,
            publishingExpanded: false,
            languageExpanded: false,
            priceExpanded: false,

            selectedOrder: [],
            selectedCategories: [],
            selectedAuthors: [],
            selectedPublishing: [],
            selectedLanguages: [],
            selectedPrice: [],
        };
    },
    mounted() {
        this.prices = [
            {id: 1, interval: '0 - 50', pret:
[0, 50]},
            {id: 2, interval: '50 - 100',
pret: [50, 100]},
            {id: 3, interval: '100 - 150',
pret: [100, 150]},
        ];
        this.orders = [
            {id: 1, name: 'Pret crescator',
order: 'pret_crescator'},
            {id: 2, name: 'Pret descrescator',
order: 'pret_descrescator'},
            {id: 3, name: 'Alfabetic
crescator', order: 'alfabetic_crescator'},
            {id: 4, name: 'Alfabetic
descrescator', order:
'alfabetic_descrescator'},
        ];
        this.getFilters();
    },
    beforeUnmount() {
        window.removeEventListener('resize',
this.getWindowDimensions);
    },
    methods: {
        getFilters() {
            axios.get('/filtre')
                .then(response => {
                    this.categories =
response.data.categorii;
                    this.authors =
response.data.autori;
                    this.publishing =
response.data.edituri;
                    this.languages =
response.data.limbi;

```

```

    })
    .catch(error => {
      console.log(error);
    });
  },
  applyFilters() {
    console.log(this.selectedOrder)
    axios.post('/filtre', {
      autor: this.selectedAuthors,
      categorie:
this.selectedCategories,
      editura:
this.selectedPublishing,
      limba: this.selectedLanguages,
      pret: this.selectedPrice,
      ordine: this.selectedOrder,
    })
    .then(response => {
      console.log(response.data);
      this.$emit('filtered',
response.data);
    })
    .catch(error => {
      console.log(error);
    });
  },
};
</script>
<style scoped>
@font-face {
  font-family: 'Lora';
  src: url('/Fonts/static/Lora-
Regular.ttf');
}
.ctg {
  overflow-y: auto;
  padding-left: 20px;
  padding-right: 20px;
}
::-webkit-scrollbar-track {
  background-color: #F0F1F2;
  border-radius: 10px;
}
::-webkit-scrollbar-thumb {
  background-color: #FFB6B9;
  border-radius: 10px;
  border: 3px solid #FFB6B9;
}
::-webkit-scrollbar-thumb:hover {
  background-color: #FFB6B9;
}
::-webkit-scrollbar-corner {
  background-color: #FFB6B9;
}
::-webkit-scrollbar {
  width: 10px;
  height: 10px;
}
.ctg-title {
  font-family: "Lora", serif;
  font-size: 20px;
  font-weight: bold;
  padding-left: 20px;
  margin-top: 20px;
}

```

```

.items {
  font-family: "Lora", serif;
  display: flex;
  font-size: 14px;
  padding-top: 8px;
}
</style>
<template>
  <div class="book-list">
    <div class="row" style="margin-left:
10px">
      <div class="col" v-for="book in
books" :key="book.id">
        <div class="card-wrapper">
          <div ref="card"
class="card mb-2">
            
            <span v-
if="isDiscounted(book)" class="position-
absolute top-0 start-100 translate-middle
badge rounded-pill bg-danger" style="margin-
top: 10px">
              {{ '-'
+mappedDiscountedBooks[book.id] + '%' }}
            </span>
            <div class="card-
body">
              <div class="card">
                <div
class="card-body d-flex">
                  <h5
class="flex-grow-1 card-title"
@click="redirectBook(book)">{{ book.titlu }} -
{{ book.autor }}</h5>
                  <div
style="margin: 10px">
                    
                    
                  </div>
                </div>
              </div>
              <div class="d-
flex">
                
                <p
class="card-review">{{
formatRating(book.rating) }} ({{
book.nr_reviewuri }})</p>
                </div>
                <p v-
if="!isDiscounted(book)" class="card-text
card-price">{{ book.pret.toFixed(2) }} Lei</p>
                <p v-else
class="card-text card-price">{{
(book.pret.toFixed(2) -
(mappedDiscountedBooks[book.id] * book.pret)
/ 100).toFixed(2) }} Lei <span class="text-
decoration-line-through reduced-price">{{
book.pret.toFixed(2) }} Lei</span></p>
                <a v-
if="book.cantitate !== 0"
@click="addToCart(book)" class="btn btn-
primary">Adauga in cos</a>
                <a v-else
class="btn btn-primary disabled"
style="background-color: rgba(23, 42, 58,
0.6)">Adauga in cos</a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </template>

```



```

        </div>
      </div>
    </div>
  </div>
</template>

<script>
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "BookList",
  props: ['books', 'token'],
  data() {
    return {
      cardElement: null,
      cardWidth: 0,
      favorite: [],
      discountedBooks: [],
      mappedDiscountedBooks: [],
    }
  },
  mounted() {
    this.getFavorite();
    this.getDiscounted();
  },
  watch: {
    books: {
      immediate: true,
      handler() {
        this.getCardWidth();
      }
    }
  },
  methods: {
    getCardWidth() {
      const cardElement =
document.querySelector('.card');
      if (cardElement) {
        const cardRect =
cardElement.getBoundingClientRect();
        this.cardWidth =
cardRect.width;
      }
    },
    getFavorite() {
      if (this.token === null) return;
      axios.get('/favorite', {
        headers: {
          'Authorization':
'Bearer ' + this.token
        }
      })
      .then(response => {
        this.favorite =
response.data;
      })
      .catch(error => {
        console.log(error);
      });
    },
    checkFavorite(carte) {
      if (this.token === null) return
false;

      for (let i = 0; i <
this.favorite.length; i++) {
        if (this.favorite[i].id_carte
=== carte.id) {
          return true;
        }
      }
      return false;
    }
  }
}

```

```

    },
    addFavorite(carte) {
      if (this.token === null) {
        window.location.href =
'/autenticare';
      }
      axios.post('/favorite',
        {
          id_carte: carte.id,
        },
        {
          headers: {
            'Authorization': 'Bearer '
+ this.token
          }
        })
      .then(response => {
        this.getFavorite();
      })
      .catch(error => {
        console.log(error);
      });
    },
    redirectBook(carte) {
      window.location.href = '/carte/' +
carte.isbn;
    },
    addToCart(book) {
      if (this.token) {
        axios.post('/cos', {
          id_carte: book.id,
          cantitate: 1,
          subtotal:
this.isDiscounted(book) ?
(book.pret.toFixed(2) -
((this.mappedDiscountedBooks[book.id] *
book.pret) / 100)).toFixed(2) :
book.pret.toFixed(2),
        },
        {
          headers: {
            'Authorization':
'Bearer ' + this.token
          }
        })
      .then(response => {
        console.log(response.data);
      })
      .catch(error => {
        console.log(error);
      })
    } else {
      axios.post('/cos', {
        id_carte: book.id,
        cantitate: 1,
        subtotal:
this.isDiscounted(book) ?
(book.pret.toFixed(2) -
((this.mappedDiscountedBooks[book.id] *
book.pret) / 100)).toFixed(2) :
book.pret.toFixed(2),
      })
      .then(response => {
        console.log(response.data);
      })
      .catch(error => {
        console.log(error);
      })
    }
  },
  getDiscounted() {
    axios.get('/discounts')
    .then(response => {
      this.discountedBooks =
response.data;
      this.discountedBooks.forEach(book => {

```

```

this.mappedDiscountedBooks[book.id_carte] =
book.promotie;
    })
    })
    .catch(error => {
        console.log(error);
    })
    },
    isDiscounted(book) {
        for (let i = 0; i <
this.discountedBooks.length; i++) {
            if
(this.discountedBooks[i].id_carte === book.id)
{
                return true;
            }
        }
        return false;
    },
    formatRating(rating) {
        if (typeof rating === 'number' &&
!isNaN(rating)) {
            return rating.toFixed(2);
        } else {
            return rating;
        }
    },
    },
}
</script>

<style scoped>
@font-face {
    font-family: 'Lora';
    src: url('/Fonts/static/Lora-
Regular.ttf');
}

.book-list {
    background-color: #FAFAFA;
    padding-top: 20px;
}

.card-wrapper {
    width: 200px;
    margin-right: 5px;
    margin-left: 5px;
    margin-bottom: 10px;
    display: inline-block;
    vertical-align: top;
}

.row {
    display: flex;
    flex-wrap: wrap;
}

.col {
    flex: 0 0 200px;
    margin-right: 5px;
    margin-left: 5px;
    margin-bottom: 10px;
}

.card {
    width: 100%;
    border: none;
    background-color: #FAFAFA;
}

.card-img-top {
    width: 100%;
    height: auto;
}

.card-body {
    padding: 0;
}

```

```

.card-title {
    display: -webkit-box;
    -webkit-box-orient: vertical;
    -webkit-line-clamp: 3;
    overflow: hidden;
    font-size: 18px;
    height: 70px;
    padding-top: 5px;
    font-family: "Lora", serif;
    color: #333333;
}

.card-price {
    display: -webkit-box;
    -webkit-box-orient: vertical;
    font-size: 20px;
    overflow: hidden;
    font-family: "Lora", serif;
    color: #333333;
}

.card-review {
    margin-bottom: 0;
    margin-left: 0;
    font-size: 16px;
    font-family: "Lora", serif;
}

.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
    margin-top: 1rem;
}

.reduced-price {
    font-size: 16px;
    margin-left: 10px;
    color: rgba(23, 42, 58, 0.6)
}

</style>

<template>
    <div>
        <book-details :book="book"
@bookList="redirectList"></book-details>
        <book-carousel :books="bookList"
:title="sectionTitle" :style="{ marginTop: 150
+ 'px' }"></book-carousel>
        <review :book="book" :style="{
marginTop: 150 + 'px' }"></review>
    </div>
</template>

<script>
import BookDetails from "./BookDetails.vue";
import BookCarousel from "./BookCarousel.vue";
import Review from "./Review.vue";
export default {
    name: "BookPage",
    props: ['book'],
    components: {
        BookDetails,
        BookCarousel,
        Review,
    },
    data() {
        return {
            bookList: null,
            sectionTitle: 'Te-ar putea
interesa si ...',
        }
    },
    methods: {
        redirectList(bookList) {
            this.bookList = bookList;
        }
    }
}

```

```

}
</script>

<style scoped>
<template>
  <div>
    <div :style="{ float: 'left', width:
350 + 'px', }">
      <book-filter
@filtered="handleData"></book-filter>
    </div>
    <div :style="{ marginLeft: 350 + 'px',
width: 1000 + 'px' }" v-if="!!emittedData">
      <book-list :books="emittedData"
:token="isAuthenticated"></book-list>
    </div>
  </div>
</template>

<script>
import BookList from "./BookList.vue";
import BookFilter from "./BookFilter.vue";
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "Books",
  props: ['books'],
  components: {
    BookList,
    BookFilter
  },
  data() {
    return {
      emittedData: null,
      windowHeight: 0,
      windowWidth: 0,
    };
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getWindowDimensions();
    window.addEventListener('resize',
this.getWindowDimensions);
    if (this.books) {
      this.emittedData = this.books;
    }
  },
  methods: {
    handleData(filtered) {
      this.emittedData = filtered;
    },
    getWindowDimensions() {
      this.windowHeight =
window.innerHeight;
      this.windowWidth =
window.innerWidth;
    },
  }
}
</script>

<style>

</style>

</style>

<template>

```

```

<div class="container" style="padding-top:
30px; padding-bottom: 30px">
  <h3 class="container d-flex justify-
content-center"><strong>Contact</strong></h3>
  <div class="mb-3">
    <label
for="exampleFormControlInput1" class="form-
label">Adresa e-mail</label>
    <input type="email" class="form-
control" id="exampleFormControlInput1" v-
model="email" placeholder="name@example.com">
  </div>
  <div class="mb-3">
    <label
for="exampleFormControlTextareal" class="form-
label">Mesaj</label>
    <textarea class="form-control"
id="exampleFormControlTextareal" v-
model="message" rows="3"></textarea>
  </div>
  <div class="d-flex justify-content-
end">
    <button type="button" class="btn
btn-primary"
@click="sendContact">Trimite</button>
  </div>
</div>
</template>

<script>
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "ContactForm",
  data() {
    return {
      message: null,
      email: null,
    }
  },
  methods: {
    sendContact() {
      axios.post('/contact', {
        email: this.email,
        message: this.message,
      })
      .then(response => {
        this.email = '';
        this.message = '';
      })
      .catch(error => {
        console.log(error);
      });
    }
  }
}
</script>

<style scoped>
.container {
  background: #EDF6F6;
  color: #09554c;
  font-family: "Lora", serif;
  margin-left: 0;
  margin-right: 0;
}

.btn-primary {
  font-family: "Lora", serif;
  display: block;
  background-color: #00A896;
  border: none;
}

```

```

</style>

<template>
  <div class="container d-flex justify-content-center align-content-center">
    <div class="col-md-6">
      <form v-if="!!client && !!user"
@submit.prevent="checkout" class="row g-3"
:style="{ width: 95 + '%', marginLeft: 10 + 'px' }">
        <h4 class="titlu-1"><strong>Date de contact</strong></h4>

        <div class="col-md-6">
          <label for="inputEmail" class="form-label">E-mail</label>
          <input type="email" class="form-control" id="inputEmail" v-model="user.email">
        </div>

        <div class="col-md-6">
          <label for="inputPhone" class="form-label">Telefon</label>
          <input type="text" class="form-control" id="inputPhone" v-model="client.nr_telefon">
        </div>

        <h4 class="titlu"><strong>Date de livrare</strong></h4>

        <div class="col-md-6">
          <label for="inputNume" class="form-label">Nume</label>
          <input type="text" class="form-control" id="inputNume" v-model="client.nume">
        </div>

        <div class="col-md-6">
          <label for="inputPrenume" class="form-label">Prenume</label>
          <input type="text" class="form-control" id="inputPrenume" v-model="client.prenume">
        </div>

        <div class="col-12">
          <label for="inputAdresa" class="form-label">Adresa</label>
          <input type="text" class="form-control" id="inputAdresa" v-model="client.adresa">
        </div>

        <div class="col-md-6">
          <label for="inputCity" class="form-label">Localitate</label>
          <input type="text" class="form-control" id="inputCity" v-model="client.oras">
        </div>

        <div class="col-md-6">
          <label for="inputState" class="form-label">Judet</label>
          <input type="text" class="form-control" id="inputState" v-model="client.judet">
        </div>

        <h4 class="titlu"><strong>Metode de livrare</strong></h4>

        <div class="form-check">
          <input type="radio" name="exampleRadios1"

```

```

id="exampleRadios1" value="fancourier" v-model="delivery" checked>
          <label class="form-check-label" for="exampleRadios1">
            Livrare prin FAN
          </label>
        </div>
        <div class="form-check">
          <input type="radio" name="exampleRadios2" id="exampleRadios2" value="sameday" v-model="delivery">
          <label class="form-check-label" for="exampleRadios2">
            Livrare prin Sameday
          </label>
        </div>
        <h4 class="titlu"><strong>Metode de plata</strong></h4>

        <div class="form-check">
          <input type="radio" name="exampleRadios3" id="exampleRadios3" value="card" v-model="payment" checked>
          <label class="form-check-label" for="exampleRadios3">
            Plata online cu cardul
          </label>
        </div>
        <div class="form-check">
          <input type="radio" name="exampleRadios4" id="exampleRadios4" value="ramburs" v-model="payment">
          <label class="form-check-label" for="exampleRadios4">
            Ramburs la livrare
          </label>
        </div>

        <div v-if="payment === 'card'" class="row g-3" style="margin-top: 20px">
          <div class="col-12">
            <label for="nrCard" class="form-label">Numarul cardului</label>
            <input type="text" class="form-control" id="nrCard">
          </div>
          <div class="col-12">
            <label for="numeCard" class="form-label">Numele de pe card</label>
            <input type="text" class="form-control" id="numeCard">
          </div>
          <div class="col-md-8">
            <label for="dataExp" class="form-label">Data de expirare</label>
            <input type="text" class="form-control" id="dataExp">
          </div>
          <div class="col-md-4">
            <label for="cvc" class="form-label">CVC</label>
            <input type="text" class="form-control" id="cvc">
          </div>
        </div>

        <div class="" style="margin-top: 40px; width: 100%; margin-bottom: 100px">
          <button type="submit" class="btn btn-primary" style="width: 100%">Trimite comanda</button>
        </div>
      </form>

```

```

    </div>
    <div class="col-md-5 bg-custom">
      <div v-if="!!carti"
class="container custom-container">
        <div v-for="carte in carti"
class="card custom mb-3" style="max-width:
540px;">
          <div class="row g-0">
            <div class="col-md-2">
              
            </div>
            <div class="col-md-7">
              <div class="card-
body">
                <h5
class="card-title">{{ carte.titlu }}</h5>
                <p
class="card-text">{{ carte.autor }}</p>
                <p v-
if="!isDiscounted(carte)" class="card-text
card-price">{{ (carte.pret *
carte.cos).toFixed(2) }} Lei</p>
                <p v-else
class="card-text card-price">{{ ((carte.pret *
carte.cos).toFixed(2) -
((mappedDiscountedBooks[carte.id] * carte.pret
* carte.cos) / 100)).toFixed(2) }} Lei <span
class="text-decoration-line-through reduced-
price">{{ (carte.pret * carte.cos).toFixed(2)
}} Lei</span></p>
              </div>
            </div>
            <div class="col-md-3">
              
              <select
class="end-quantity form-select" aria-
label="Default select example" v-
model="carte.cos" @change="updateCos(carte)">
                <option v-
for="i in carte.cantitate" :value="i"
:selected="i === carte.cos">{{ i }}</option>
              </select>
            </div>
          </div>
        </div>
        <div class="checkout-container">
          <div class="">Subtotal:</div>
          <div class="">{{ total }}
        </div>
        <div class="checkout-container">
          <div class="">Taxe de
transport:</div>
          <div class="">15.00 Lei</div>
          <div class="checkout-container-
total">
            <div class="">Total:</div>
            <div class="">{{ total + 15.00
}} Lei</div>
          </div>
        </div>
      </div>
    </div>
  </template>

  <script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',

```

```

  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "Cos",
  data() {
    return {
      client: null,
      user: null,
      carti: null,
      showPopup: false,
      discountedBooks: [],
      mappedDiscountedBooks: [],
      quantity: 0,
      quantityModified: [],
      total: 0.00,
      delivery: null,
      payment: null,
    }
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getClient();
    this.getCos();
    this.getDiscounted();
    setTimeout(() => {
      this.getTotal();
    }, 1000);
  },
  methods: {
    getClient() {
      if (this.isAuthenticated) {
        axios.get('/user', {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        })
        .then(response => {
          this.client =
response.data.client;
          this.user =
response.data.user;
        })
        .catch(error => {
          console.log(error);
        });
      } else {
        this.client = {
          nume: null,
          prenume: null,
          nr_telefon: null,
          adresa: null,
          oras: null,
          judet: null,
          id: 0,
        };
        this.user = {
          email: null
        };
      }
    },
    getCos() {
      if (this.isAuthenticated) {
        axios.get('/cos', {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        })
        .then(response => {
          this.carti =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
      } else {

```

```

        axios.get('/cos')
            .then(response => {
                this.carti =
response.data;
            })
            .catch(error => {
                console.log(error);
            });
    },
    getDiscounted() {
        axios.get('/discounts')
            .then(response => {
                this.discountedBooks =
response.data;

this.discountedBooks.forEach(book => {

this.mappedDiscountedBooks[book.id_carte] =
book.promotie;

            })
            .catch(error => {
                console.log(error);
            })
    },
    isDiscounted(book) {
        for (let i = 0; i <
this.discountedBooks.length; i++) {
            if
(this.discountedBooks[i].id_carte === book.id)
{
                return true;
            }
        }
        return false;
    },
    deleteItem(carte) {
        if (this.isAuthenticated) {
            axios.delete('/cos/' +
carte.id, {
                headers: {
                    'Authorization':
'Bearer ' + this.isAuthenticated
                }
            })
            .then(response => {

window.location.reload();
            })
            .catch(error => {
                console.log(error);
            });
        } else {
            axios.delete('/cos/' +
carte.id)
                .then(response => {

window.location.reload();
            })
            .catch(error => {
                console.log(error);
            });
        }
    },
    updateCos(carte) {
        if (this.isAuthenticated) {
            axios.put('/cos', {
                id_carte: carte.id,
                cantitate: carte.cos
            },
            {
                headers: {
                    'Authorization':
'Bearer ' + this.isAuthenticated
                }
            })
            .then(response => {

```

```

window.location.reload();
            })
            .catch(error => {
                console.log(error);
            });
        } else {
            axios.put('/cos', {
                id_carte: carte.id,
                cantitate: carte.cos
            })
            .then(response => {

window.location.reload();
            })
            .catch(error => {
                console.log(error);
            });
        }
    },
    getTotal() {
        this.carti.forEach(carte => {
            if (!this.isDiscounted(carte))
{
                this.total +=
parseFloat((carte.pret *
carte.cos).toFixed(2)); // Use += for numeric
addition
            } else {
                this.total +=
parseFloat(((carte.pret *
carte.cos).toFixed(2) -
((this.mappedDiscountedBooks[carte.id] *
carte.pret * carte.cos) / 100)).toFixed(2));
                // Use += for numeric addition
            }
        });
    },
    checkout() {
        axios.post('/checkout', {
            carti: this.carti,
            id_client: this.client.id,
            tip: 'comanda',
            email: this.user.email,
            pret_comanda: this.total +
15.00,
            adresa_livrare:
this.client.adresa,
            oras_livrare:
this.client.oras,
            judet_livrare:
this.client.judet,
            status: 'in curs de
pregatire',
            metoda_plata: this.payment,
        })
            .then(response => {
                window.location.href =
'/'
            })
            .catch(error => {
                console.log(error);
            });
    }
}
</script>

<style scoped>
.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
}

.reduced-price {
    font-size: 16px;
    margin-left: 10px;
    color: rgba(23, 42, 58, 0.6)
}

```

```

.form-select {
    width: 90%;
}

.custom {
    border: none;
    background-color: transparent;
    width: 100%;
    padding-bottom: 30px;
    border-bottom: 1px solid rgba(0, 0, 0, 0.2);
    border-radius: 0;
}

.custom-container {
    background-color: #fafafa;
}

.end-delete {
    position: absolute;
    right: 0;
    top: 15px;
}

.end-quantity {
    position: absolute;
    right: 0;
    bottom: 45px;
    width: 13%;
}

.checkout-container {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 10px;
    background-color: transparent;
}

.checkout-container-total {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 10px;
    background-color: transparent;
    margin-top: 20px;
    border-top: 1px solid rgba(0, 0, 0, 0.2);
    border-radius: 0;
}

.bg-custom {
    background-color: #fafafa;
}

.titlu {
    font-family: "Lora", serif;
    color: #333333;
    margin-bottom: 30px;
    margin-top: 50px;
}

.titlu-1 {
    font-family: "Lora", serif;
    color: #333333;
    margin-bottom: 30px;
}

/* Customize the appearance of the checked
radio button */
.form-check-input[type="radio"]:checked {
    background-color: #008b7a;
    border-color: #008b7a;
}

```

```

<div class="donation" :class="{
'donation-step1': step === 1, 'donation-
step2': step === 2, 'donation-step3': step
===3 }">
    <form action="">
        <div v-show="step===1">
            <h2>Informatii
donator</h2>
            <div class="donation-
input">
                <input type="text" v-
model="nume" required="">
                <label>Nume</label>
            </div>
            <div class="donation-
input">
                <input type="text" v-
model="prenume" required="">
                <label>Prenume</label>
            </div>
            <div class="donation-
input">
                <input type="text" v-
model="email" required="">
                <label>E-mail</label>
            </div>
            <div class="donation-
input">
                <input type="tel" v-
model="telefon" required="">
                <label>Telefon</label>
            </div>
            <button type="button"
class="donation-button-next"
@click="nextStep()">Continua</button>
        </div>
        <div v-show="step===2">
            <h2>Detalii donatie</h2>
            <div v-for="(book, index)
in books" :key="index">
                <div class="donation-
details">
                    <div
class="donation-input">
                        <input
type="text" v-model="book.titlu" required
title="">
                        <label>Titlu</label>
                    </div>
                    <div
class="donation-input">
                        <input
type="text" v-model="book.autor" required="">
                        <label>Autor</label>
                    </div>
                    <div
class="donation-input">
                        <input
type="text" v-model="book.isbn" required="">
                        <label>ISBN</label>
                    </div>
                </div>
                <div class="buttons-container">
                    <button
type="button" class="add-button"
@click="addBook()">+</button>
                    <button
type="button" class="remove-button"
@click="removeBook(index)">-</button>
                </div>
            </div>
        </div>
        <div class="navigation-
buttons-container">

```

```

        <button type="button"
class="donation-button-next-prev"
@click="prevStep()">Inapoi</button>
        <button type="button"
class="donation-button-next-prev"
@click="nextStep()">Continua</button>
    </div>
</div>

<div v-show="step===3">
    <h2>Adresa preluare</h2>
    <div class="donation-
input">
        <input type="text" v-
model="adresa" required title="">
        <label>Adresa</label>
    </div>
    <div class="donation-
input">
        <input type="text" v-
model="localitate" required="">
    </div>
    <label>Localitate</label>
    </div>
    <div class="donation-
input">
        <input type="text" v-
model="judet" required="">
        <label>Judet</label>
    </div>
    <div class="navigation-
buttons-container">
        <button type="button"
class="donation-button-next-prev"
@click="prevStep()">Inapoi</button>
        <button type="button"
class="donation-button-next-prev"
@click="send()">Trimite</button>
    </div>
</div>
</form>
</div>
</div>
</template>

<script>
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: "DonationForm",
    data() {
        return {
            nume: '',
            prenume: '',
            email: '',
            telefon: '',
            adresa: '',
            localitate: '',
            judet: '',
            step: 1,
            books: [
                { titlu: '', autor: '', isbn:
'' }
            ]
        };
    },
    methods: {
        nextStep() {
            if (this.step === 1 && this.nume
&& this.prenume && this.email && this.telefon)
{
                const specialCharRegex = /@/;

```

```

                if
(!specialCharRegex.test(this.email)) {
                    alert('E-mail invalid');
                    return;
                }
                this.step++;
                return;
            } else {
                if (this.step === 2 ||
this.step === 3) {
                    this.step++;
                } else {
                    alert('Completati toate
campurile');
                    return;
                }
            }
        },
        prevStep() {
            this.step--;
        },
        addBook() {
            this.books.push({ titlu: '',
autor: '', isbn: '' });
        },
        removeBook(index) {
            this.books.splice(index,1);
        },
        send() {
            axios.post('/donatie', {
                nume: this.nume,
                prenume: this.prenume,
                email: this.email,
                nr_telefon: this.telefon,
                adresa_ridicare: this.adresa,
                oras_ridicare:
this.localitate,
                judet_ridicare: this.judet,
                books: this.books
            })
                .then((response) => {
                    window.location.href =
'/';
                }, (error) => {
                    console.log(error);
                });
        }
    },
};
</script>

<style scoped>
.donation-container {
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 100px;
    margin-bottom: 100px;
}

.donation {
    /*width: 380px;*/
    position: relative;
    padding: 50px;
    background: #fafafa;
    box-sizing: border-box;
    box-shadow: 0 15px 25px rgba(0,0,0,0.1);
    border-radius: 10px;
}

.donation-step1 {
    width: 420px;
}

.donation-step2 {
    width: 950px;
}

.donation-step3 {

```



```

        width: 420px;
    }

    .donation-details{
        display: flex;
        flex-direction: row;
        align-items: center;
        gap: 100px;
    }

    .donation .donation-details .donation-input
    input{
        width: 140%;
    }

    .buttons-container {
        display: flex;
        margin-bottom: 20px;
        gap: 2px;
    }

    .add-button {
        width: 20px;
        height: 20px;
        border: none;
        border-radius: 2px;
        background-color: #FFB6B9;
        display: flex;
        justify-content: center;
        align-items: center;
        /*border-width: 1px;*/
        /*border-color: white;*/
    }

    .remove-button {
        width: 20px;
        height: 20px;
        border: none;
        border-radius: 2px;
        background-color: #FFB6B9;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .donation h2 {
        margin: 0 0 40px;
        padding-bottom: 20px;
        color: #09554c;
        text-align: center;
    }

    .donation .donation-input {
        position: relative;
    }

    .donation .donation-input input {
        width: 100%;
        padding: 6px 0;
        font-size: 14px;
        color: #09554c;
        margin-bottom: 30px;
        border: none;
        border-bottom: 1px solid #09554c;
        outline: none;
        background: transparent;
    }

    .donation .donation-input label {
        position: absolute;
        top: 0;
        left: 0;
        padding: 6px 0;
        font-size: 14px;
        color: #09554c;
        pointer-events: none;
        transition: .5s;
    }

    .donation .donation-input input:focus ~ label,
    .donation .donation-input input:valid ~ label
    {
        top: -20px;
        left: 0;
        color: #333333;
        font-size: 12px;
    }

    .navigation-buttons-container {
        display: flex;
        justify-content: space-between;
        margin-left: 0;
        margin-right: 0;
    }

    .donation-button-next {
        color: #fff;
        font: inherit;
        padding: 12px 25px;
        border-radius: 8px;
        cursor: pointer;
        background: linear-gradient(to left,
        #80CBC4, #008b7a);
        /*border: 2px solid rgba(0,0,0,0.5);*/
        border: none;
        /*margin: 40px 20px auto;*/
        margin-top: 40px;
        margin-right: 0;
        margin-left: 203px;
    }

    .donation-button-next-prev {
        color: #fff;
        font: inherit;
        padding: 12px 25px;
        border-radius: 8px;
        cursor: pointer;
        background: linear-gradient(to left,
        #80CBC4, #008b7a);
        /*border: 2px solid rgba(0,0,0,0.5);*/
        border: none;
        margin: 40px 0 0 0;
    }

    .donation-button-next:hover {
        background: linear-gradient(to right,
        #80CBC4, #008b7a);
    }

    .donation-button-next-prev:hover {
        background: linear-gradient(to right,
        #80CBC4, #008b7a);
    }

</style>

<template>
    <div class="log-in-container">
        <div class="log-in">
            <h2>Autenticare</h2>
            <form @submit.prevent="login()">
                <div class="log-in-input">
                    <input type="text" v-
                    model="email" required="">
                    <label>E-mail</label>
                </div>
                <div class="log-in-input">
                    <input type="password" v-
                    model="password" required="">
                    <label>Parola</label>
                </div>
                <span class="forget-password-
                switch">
                    <a @click="openPopUp">Ti-
                    ai uitat parola?</a>
                </span>
            </form>
        </div>
    </div>

```

```

        <div v-if="showPopup"
class="popup-container">
        <div class="popup-content"
:style="{ width: 600 + 'px' }">
            <form>
                <div class="mb-3">
                    <label
for="pass" class="form-label">Email</label>
                    <input
type="email" class="form-control" id="pass" v-
model="resetMail">
                </div>
                <div class="d-flex
justify-content-end" style="margin-top: 30px">
                    <button
type="button" class="btn btn-primary"
@click="sendRequest" style="margin-right:
10px">Submit</button>
                    <button
type="button" class="btn btn-secondary"
@click="closePopup">Close</button>
                </div>
            </form>
        </div>
        <button type="submit"
class="log-in-button">Conectare</button>
        <span class="log-in-switch">Nu
ai cont? <a href="/inregistrare">Creeaza
unul</a>
        </span>
    </form>
</div>
</div>
</template>

<script>
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: "LogIn",
    data() {
        return {
            email: null,
            password: null,
            showPopup: false,
            resetMail: null,
        }
    },
    methods: {
        login() {
            const specialCharRegex = /@/;
            if
(!specialCharRegex.test(this.email)) {
                alert('E-mail invalid');
                return;
            }
            axios.post('/login', {
                email: this.email,
                password: this.password
            })
                .then(response => {
                    const token =
response.data.token;

this.$store.commit('setToken', token);

localStorage.setItem('token', token);

                window.location.href =
'/cautare';
            })

```

```

                .catch(error => {
                    alert('Datele introduce nu
sunt corecte!')
                    console.error(error);
                });
            },
            openPopup() {
                this.showPopup = true;
            },
            closePopup() {
                this.showPopup = false;
            },
            sendRequest() {
                axios.post('/reset-request', {
                    email: this.resetMail
                })
                    .then(response => {
                        console.log(response.data);
                        this.closePopup();
                    })
                    .catch(error => {
                        console.error(error);
                    })
            }
        }
    }
</script>

<style scoped>
.log-in-container {
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 100px;
    margin-bottom: 100px;
}

.log-in {
    width: 420px;
    padding: 50px;
    background: #fafafa;
    box-sizing: border-box;
    box-shadow: 0 15px 25px rgba(0,0,0,0.1);
    border-radius: 10px;
}

.log-in h2 {
    margin: 0 0 40px;
    padding-bottom: 20px;
    color: #09554c;
    text-align: center;
}

.log-in .log-in-input {
    position: relative;
}

.log-in .log-in-input input {
    width: 100%;
    padding: 6px 0;
    font-size: 14px;
    color: #09554c;
    margin-bottom: 30px;
    border: none;
    border-bottom: 1px solid #09554c;
    outline: none;
    background: transparent;
}

.log-in .log-in-input label {
    position: absolute;
    top: 0;
    left: 0;
    padding: 6px 0;
    font-size: 14px;
    color: #09554c;
    pointer-events: none;
    transition: .5s;

```

```

}

.log-in .log-in-input input:focus ~ label,
.log-in .log-in-input input:valid ~ label {
  top: -20px;
  left: 0;
  color: #333333;
  font-size: 12px;
}

.forget-password-switch {
  display: block;
  margin-top: 0;
}

.forget-password-switch a {
  font-size: 14px;
  color: #333333;
}

.forget-password-switch a {
  text-decoration: none;
}

.forget-password-switch a:hover {
  text-decoration: underline;
}

.log-in-button {
  display: block;
  margin: 40px auto 48px;
  padding: 14px 60px;
  background: linear-gradient(to left,
#80CBC4, #008b7a);
  color: #ffffff;
  font: inherit;
  border: none;
  cursor: pointer;
  border-radius: 8px;
}

.log-in-button:hover {
  background: linear-gradient(to right,
#80CBC4, #008b7a);
}

.log-in-switch {
  display: block;
  text-align: center;
  font-size: 16px;
  color: #333333;
}

.log-in-switch a {
  text-decoration: none;
  color: #26A69A;
}

.log-in-switch a:hover {
  text-decoration: underline;
}

.popup-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
}

.popup-content {
  background-color: #fff;
  padding: 20px;
  border-radius: 4px;

```

```

}

.btn-primary {
  display: block;
  background-color: #00A896;
  border: none;
  width: auto;
}

</style>

<template>
  <div>
    <book-carousel :books="bookLatest"
:title="sectionLatest" :style="{ marginTop:
100 + 'px' }"></book-carousel>
    <book-carousel :books="bookDiscounted"
:title="sectionDiscounted" :style="{
marginTop: 100 + 'px' }"></book-carousel>
    <contact-form :style="{ marginTop: 100
+ 'px', marginBottom: 100 + 'px' }"></contact-
form>
  </div>
</template>

<script>
import BookCarousel from "./BookCarousel.vue";
import ContactForm from "./ContactForm.vue";
import axios from "axios";
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};

export default {
  name: "MainPage",
  data() {
    return {
      bookLatest: [],
      sectionLatest: 'Noutati',
      bookDiscounted: [],
      sectionDiscounted: 'Promotii',
    },
  },
  components: {
    BookCarousel,
    ContactForm,
  },
  mounted() {
    this.getLatest();
    this.getDiscounted();
  },
  methods: {
    getLatest() {
      axios.get('/latest')
        .then(response => {
          this.bookLatest =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
    },
    getDiscounted() {
      axios.get('/discounted')
        .then(response => {
          this.bookDiscounted =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
    },
  },
}

```

```

</script>

<style scoped>

</style>
<template>
  <div v-if="bookList !== []">
    <book-list :books="booklist"
:token="isAuthenticated"></book-list>
  </div>
</template>

<script>
import BookList from "./BookList.vue";
import axios from 'axios';
import {mapGetters} from "vuex";
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "MyFavorites",
  components: {
    BookList,
  },
  data() {
    return {
      booklist : [],
      favorite: [],
      windowHeight: 0,
      windowWidth: 0,
    }
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getWindowDimensions();
    this.getFavorite();
    setTimeout(() => {
      this.getCarti();
    }, 500);
  },
  methods: {
    getFavorite() {
      if (this.isAuthenticated === null)
{
          window.location.href =
'/autentificare';
        }
        axios.get('/favorite', {
          headers: {
            'Authorization': 'Bearer '
+ this.isAuthenticated
          }
        })
        .then(response => {
          this.favorite =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
      },
      getCarti() {
        if (this.favorite !== null) {
          this.favorite.forEach((item)
=> {
              axios.get('/carti/' +
item.id_carte)
                .then(response => {
this.booklist.push(response.data);
                })
                .catch(error => {

```

```

console.log(error);
            });
          })
        },
        getWindowDimensions() {
          this.windowHeight =
window.innerHeight;
          this.windowWidth =
window.innerWidth;
        },
      }
    }
  </script>

<style scoped>

</style>

<template>
  <div v-if="!!orders">
    <div v-for="order in orders"
:key="order.id">
      <div class="card">
        <p> <strong>ID comanda: {{
order.id }}</strong> </p>
        <p> Data plasare:
{{order.data_plasare}} </p>
        <div class="row">
          <p class="col-md-auto">
Suma achitata: {{order.pret_comanda}} </p>
          <p class="col-md-auto">
Metoda de plata: {{order.metoda_plata}} </p>
        </div>
        <div class="row">
          <p class="col-md-auto">
Data estimata de livrare:
{{order.data_livrare}} </p>
          <p class="col-md-auto">
Status: {{order.status}} </p>
        </div>
        <div class="row">
          <p class="col-md-auto">
Adresa livrare: {{order.adresa_livrare + ', '
+ order.oras_livrare + ', ' +
order.judet_livrare}} </p>
        </div>
        <button type="button"
class="btn btn-primary"
@click="showBooks">Arata cartile din
comanda</button>
        <div v-if="show"
style="margin-top: 30px">
          <div class="row" v-
for="book in mappedBooks[order.id]">
            <div class="card w-75
mb-3">
              <div class="card-
body">
                <h5
class="card-title">{{ book.titlu }}</h5>
                <p
class="card-text">{{ book.autor }}</p>
                <a href="#"
class="btn btn-primary"
@click="returnOrder(order,
book)">Returneaza</a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';

```

```

const csrfToken =
document.querySelector('meta[name="csrf-token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "MyOrders",
  data() {
    return {
      orders: null,
      mappedBooks: {},
      books: null,
      show: false,
    }
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getOrders();
    setTimeout(() => {
      this.orderBooks();
    }, 500);
  },
  methods: {
    getOrders() {
      axios.get('/comenzi', {
        headers: {
          'Authorization': 'Bearer '
+ this.isAuthenticated
        })
      .then(response => {
        this.orders =
response.data;
      })
      .catch(error => {
        console.log(error);
      });
    },
    orderBooks() {
      this.orders.forEach(async (order)
=> {
        const books = await
this.getOrderBooks(order);
        this.mappedBooks[order.id] =
books;
      });
      console.log(this.mappedBooks);
    },
    async getOrderBooks(order) {
      try {
        const response = await
axios.get('/comanda-carti/' + order.id, {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        });
        return response.data;
      } catch (error) {
        console.log(error);
        return null;
      }
    },
    showBooks() {
      this.show = !this.show;
    },
    returnOrder(order, book) {
      axios.post('/retur', {
        id_comanda: order.id,
        id_carte: book.id,
      },
      {
        headers: {
          'Authorization': 'Bearer '

```

```

+ this.isAuthenticated
      })
    })
    .then(response => {
      console.log(response);
    })
    .catch(error => {
      console.log(error);
    });
  },
}
</script>

<style scoped>
@font-face {
  font-family: 'Lora';
  src: url('/Fonts/static/Lora-Regular.ttf');
}

.card {
  margin-bottom: 30px;
  border: none;
  font-family: "Lora", serif;
  background-color: #FAFAFA;
}

.btn-primary {
  display: block;
  background-color: #00A896;
  border: none;
  width: 250px;
}

</style>

<template>
  <div v-if="products !== []">
    <book-list :books="products"
:token="isAuthenticated"></book-list>
  </div>
</template>

<script>
import BookList from "../BookList.vue";
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "MyProducts",
  components: {
    BookList,
  },
  data() {
    return {
      products: [],
      windowHeight: 0,
      windowWidth: 0,
    }
  },
  computed: {
    ...mapGetters(['isAuthenticated']),
  },
  mounted() {
    this.getWindowDimensions();
    setTimeout(() => {
      this.getProducts();
    }, 500);
  },
  methods: {
    getProducts() {
      if (this.isAuthenticated) {

```

```

        axios.get('/comenzi-carti', {
          headers: {
            'Authorization':
'Bearer ' + this.isAuthenticated
          }
        })
        .then(response => {
          this.products =
response.data;
        })
        .catch(error => {
          console.log(error);
        });
      } else {
        window.location.href =
'/autenticare';
      }
    },
    getWindowDimensions() {
      this.windowHeight =
window.innerHeight;
      this.windowWidth =
window.innerWidth;
    },
  },
}
</script>

<style scoped>
</style>

<template>
  <form v-if="!!user && !!client"
@submit.prevent="sendData" class="row g-3">
    <div class="col-md-6">
      <label for="inputNume"
class="form-label">Nume</label>
      <input type="text" class="form-
control" id="inputNume" v-model="client.numa">
    </div>

    <div class="col-md-6">
      <label for="inputPrenume"
class="form-label">Prenume</label>
      <input type="text" class="form-
control" id="inputPrenume" v-
model="client.prenume">
    </div>

    <div class="col-md-6">
      <label for="inputEmail"
class="form-label">Email</label>
      <input type="email" class="form-
control" id="inputEmail" v-model="user.email">
    </div>

    <div class="col-md-6">
      <label for="inputPhone"
class="form-label">Numar Telefon</label>
      <input type="text" class="form-
control" id="inputPhone" v-
model="client.nr_telefon">
    </div>

    <div class="col-12">
      <label for="inputAddress"
class="form-label">Adresa</label>
      <input type="text" class="form-
control" id="inputAddress" v-
model="client.adresa">
    </div>

    <div class="col-md-6">
      <label for="inputCity"
class="form-label">Localitate</label>
      <input type="text" class="form-
control" id="inputCity" v-model="client.oras">
    </div>
  </form>
</template>

```

```

    <div class="col-md-4">
      <label for="inputState"
class="form-label">Judet</label>
      <input type="text" class="form-
control" id="inputState" v-
model="client.judet">
    </div>

    <div class="col-md-2">
      <label for="formular" class="form-
label">Schimba parola</label>
      <button id="formular" class="btn
btn-primary form-control" type="button"
@click="openPopup">Deschide formular</button>
    </div>

    <div v-if="showPopup"
class="popup-container">
      <div class="popup-content">
        <form>
          <div class="mb-3">
            <label for="pass"
class="form-label">Parola noua</label>
            <input
type="password" class="form-control" id="pass"
v-model="password" @input="validatePassword">
          </div>
          <div class="mb-3">
            <label
for="confPass" class="form-label">Confirmare
parola</label>
            <input
type="password" class="form-control"
id="confPass" v-model="confirmPassword">
          </div>
          <div class="d-flex
justify-content-end">
            <button
type="button" class="btn btn-primary"
@click="submitForm">Submit</button>
            <button
type="button" class="btn btn-secondary"
@click="closePopup">Close</button>
          </div>
        </form>
      </div>
    </div>

    <div class="col-12">
      <button type="submit" class="btn
btn-primary">Salveaza</button>
    </div>

    <div v-if="user.email ===
'inkpaper2023@hotmail.com'" class="col-12">
      <button type="button" class="btn
btn-primary" @click="redirectAdmin()">Pagina
admin</button>
    </div>
  </form>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "MyProfile",
  data() {
    return {
      client: null,
      user: null,
      showPopup: false,

```

```

        password: '',
        confirmPassword: '',
        isPasswordValid: false,
    },
    computed: {
        ...mapGetters(['isAuthenticated']),
    },
    mounted() {
        this.getData();
    },
    methods: {
        getData() {
            if (this.isAuthenticated) {
                axios.get('/user', {
                    headers: {
                        'Authorization':
'Bearer ' + this.isAuthenticated
                    }
                })
                .then(response => {
                    this.client =
response.data.client;
                    this.user =
response.data.user;
                })
                .catch(error => {
                    console.log(error);
                });
            } else {
                window.location.href =
'/autenticare'
            }
        },
        sendData() {
            if (this.isAuthenticated) {
                axios.put('/user', {
                    email: this.user.email,
                    nume: this.client.nume,
                    prenume:
this.client.prenume,
                    nr_telefon:
this.client.nr_telefon,
                    adresa:
this.client.adresa,
                    oras: this.client.oras,
                    judet: this.client.judet,
                }, {
                    headers: {
                        'Authorization':
'Bearer ' + this.isAuthenticated
                    }
                })
                .catch(error => {
                    console.log(error);
                });
            }
        },
        openPopup() {
            this.showPopup = true;
        },
        closePopup() {
            this.showPopup = false;
        },
        submitForm() {
            if (this.password !==
this.confirmPassword) {
                alert('Parolele nu coincid!');
                return;
            }
            if (!this.parolaValidata) {
                alert('Parola trebuie sa
contina cel putin o litera mare, o litera
mica, un numar si un caracter special!');
                return;
            }
            axios.put('/user/parola',

```

```

        {
            parola: this.password,
        },
        {
            headers: {
                'Authorization':
'Bearer ' + this.isAuthenticated
            }
        })
        .then(response => {
            this.closePopup();
        })
        .catch(error => {
            console.log(error);
        });
    },
    validatePassword() {
        const uppercaseRegex = /[A-Z]/;
        const lowercaseRegex = /[a-z]/;
        const numberRegex = /[0-9]/;
        const specialCharRegex = /[-
!$%^&*()_+|~='{}[\]:";'<>?,.\|/]/;

        this.isPasswordValid =
uppercaseRegex.test(this.password) &&
lowercaseRegex.test(this.password) &&
numberRegex.test(this.password) &&
specialCharRegex.test(this.password);
    },
    redirectAdmin() {
        window.location.href = '/admin';
    }
}
</script>

<style scoped>
.popup-container {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    display: flex;
    align-items: center;
    justify-content: center;
}

.popup-content {
    background-color: #fff;
    padding: 20px;
    border-radius: 4px;
}

.btn-primary {
    display: block;
    background-color: #00A896;
    border: none;
}

.btn-secondary {
    display: block;
    border: none;
}
</style>

<template>
<nav class="navigation background" >
    <div class="left-section">
        <a href="/principala">
            
        </a>
    </div>
</nav>
</template>

```

```

        <form @submit.prevent="search()"
class="search-container">
        <input type="search"
class="search" v-model="titlu"
placeholder="Search">
        </form>
    </div>
    <div class="right-section">
        <ul>
            <li class="item"><a
href="/cautare"></a></li>
            <li class="item">
                <div class="dropdown">
                    <a style="color:
#FAFAFA" class="dropdown-toggle" role="button"
@click="toggleDropdown" :aria-
expanded="isDropdownOpen.toString()">
                        
                    </a>
                    <ul v-if="isLogged"
class="dropdown-menu" v-show="isDropdownOpen"
@click="closeDropdown">
                        <li><a
class="dropdown-item"
@click="profile">Profil</a></li>
                        <li><a
class="dropdown-item"
@click="logout">Logout</a></li>
                    </ul>
                    <ul v-else
class="dropdown-menu" v-show="isDropdownOpen"
@click="closeDropdown">
                        <li><a
class="dropdown-item"
@click="login">Login</a></li>
                        <li><a
class="dropdown-item"
@click="register">Autentificare</a></li>
                    </ul>
                </div>
            </li>
            <li class="item"><a
href="/profil/produce_favorite"></a></li>
            <li class="item"><a
href="/cart"></a></li>
            <li class="item-last"><a
href="/donare"></a></li>
        </ul>
    </div>
</nav>
</template>

<script>
import { mapGetters } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-token"]').content;
axios.defaults.headers.common = {
    'X-Requested-With': 'XMLHttpRequest',
    'X-CSRF-TOKEN': csrfToken
};
export default {
    name: "Navigation",
    data() {
        return {
            titlu: null,
            isDropdownOpen: false,
            isLogged: false
        }
    },

```

```

        computed: {
            ...mapGetters(['isAuthenticated']),
        },
        mounted() {
            this.isLogged = this.isAuthenticated
            !== null;
        },
        methods: {
            search() {
                axios.post('/search', {
                    search: this.titlu,
                })
                    .then(response => {
                        window.location.href =
response.data;
                    })
                    .catch(error => {
                        console.error(error);
                    });
            },
            toggleDropdown() {
                this.isDropdownOpen =
!this.isDropdownOpen;
            },
            closeDropdown() {
                this.isDropdownOpen = false;
            },
            logout() {
                axios.post('/user/logout', {},
                    {
                        headers: {
                            'Authorization': 'Bearer '
+ this.isAuthenticated
                        }
                    })
                    .then(response => {
                        this.$store.commit('setToken', null);
                        localStorage.removeItem('token');
                        window.location.href =
'/cautare';
                    })
                    .catch(error => {
                        console.error(error);
                    });
            },
            login() {
                window.location.href =
'/autentificare';
            },
            register() {
                window.location.href =
'/inregistrare';
            },
            profile() {
                window.location.href = '/profil';
            },
            mail() {
                axios.post('/mail', {})
                    .then(response => {
                        console.log('done')
                    })
                    .catch(error => {
                        console.error(error);
                    });
            }
        }
    }
</script>

<style scoped>
@font-face {
    font-family: 'Lora';
    src: url('/Fonts/static/Lora-Regular.ttf');
}

.navigation {

```



```

    background-color: #008b7a;
    height: 70px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.left-section {
    width: 480px;
}

.right-section ul {
    font-family: "Lora", serif;
    display: flex;
    padding-top: 10px
}

.right-section ul li {
    list-style: none;
}

.right-section ul li a {
    font-size: 17px;
    font-weight: 700;
    text-decoration: none;
    padding: 10px;
    transition: 0.5s ease;
}

.search-container{
    position: absolute;
    left: 35%;
    top: 15px;
}

.search {
    height: 40px;
    width: 510px;
    background-color: #FAFAFA;
    font-family: "Lora", serif;
    font-size: 14px;
    border: none;
    border-radius: 8px;
    padding-left: 40px;
    background-image: url("/icons/magnifying-glass-solid.svg");
    background-size: 16px 16px;
    background-position-y: center;
    background-position-x: 12px;
    background-repeat: no-repeat;
    outline: none;
    box-shadow: 0 0 0 2px #008b7a;
}

.dropdown-menu {
    display: none;
    position: absolute;
    z-index: 1000;
    min-width: 10rem;
    padding: 0.5rem 0;
    margin: 0.125rem 0 0;
    font-size: 1rem;
    color: #212529;
    text-align: left;
    list-style: none;
    background-color: #fff;
    background-clip: padding-box;
    border: 1px solid rgba(0, 0, 0, 0.15);
    border-radius: 0.25rem;
    column-count: 1; /* Display items in a
single column */
    column-gap: 0;
}

.dropdown-menu li {
    padding: 0.25rem 1.5rem;
}

.dropdown-menu li:hover {

```

```

    background-color: #fff;
}

.dropdown-menu li a {
    color: #000000;
    text-decoration: none;
}

.dropdown-menu .dropdown-divider {
    margin: 0.5rem 0;
    border: none;
    border-top: 1px solid rgba(0, 0, 0, 0.15);
}

.dropdown-item {
    color: #000000;
}

.item {
    padding-left: 8px;
    padding-right: 8px;
    border-right: 1px solid rgba(0, 0, 0,
0.15);
}

.item-last {
    padding-left: 8px;
    padding-right: 8px;
}
}
</style>

<template>
  <div>
    <profile-navigation
@changeTab="activeTab"
:accessedTab="currentTab"></profile-
navigation>
    <my-profile v-if="currentTab ===
'date_personale'" class="top-margin"></my-
profile>
    <my-favorites v-if="currentTab ===
'produse_favorite'" class="top-margin"></my-
favorites>
    <my-orders v-if="currentTab ===
'comenzile_mele'" class="top-margin"></my-
orders>
    <my-products v-if="currentTab ===
'produse_cumparate'" class="top-margin"></my-
products>

  </div>
</template>

<script>
import ProfileNavigation from
"./ProfileNavigation.vue";
import MyProfile from "./MyProfile.vue";
import MyFavorites from "./MyFavorites.vue";
import MyOrders from "./MyOrders.vue";
import MyProducts from "./MyProducts.vue";
export default {
  name: "Profile",
  props: ['ref1'],
  components: {
    ProfileNavigation,
    MyProfile,
    MyFavorites,
    MyOrders,
    MyProducts,
  },
  data() {
    return {
      currentTab: null,
    }
  },
  mounted() {

```

```

        if (this.ref1 !== null) {
            this.currentTab = this.ref1;
        } else {
            this.currentTab =
' date_personale';
        }
    },
    methods: {
        activeTab(tab) {
            this.currentTab = tab;
        }
    }
}
</script>

<style scoped>
    .top-margin {
        margin-top: 80px;
    }
</style>

<template>
    <div class="nav-container">
        <ul class="nav nav-underline">
            <li class="nav-item">
                <a
                    :class="['nav-link', {
active: currentTab === 'date_personale' }]"
href="/profil/date_personale"
@click="changeTab('date_personale')"
                >
                    Date personale
                </a>
            </li>
            <li class="nav-item">
                <a
                    :class="['nav-link', {
active: currentTab === 'comenzile_mele' }]"
href="/profil/comenzile_mele"
@click="changeTab('comenzile_mele')"
                >
                    Comenzile mele
                </a>
            </li>
            <li class="nav-item">
                <a
                    :class="['nav-link', {
active: currentTab === 'produse_cumparate' }]"
href="/profil/produse_cumparate"
@click="changeTab('produse_cumparate')"
                >
                    Produse cumparate
                </a>
            </li>
            <li class="nav-item">
                <a
                    :class="['nav-link', {
active: currentTab === 'produse_favorite' }]"
href="/profil/produse_favorite"
@click="changeTab('produse_favorite')"
                >
                    Produse favorite
                </a>
            </li>
        </ul>
    </div>
</template>

<script>
export default {
    name: "ProfileNavigation",

```

```

        props: ['accessedTab'],
        data() {
            return {
                currentTab: null,
            },
            computed: {
                underlineStyle() {
                    const activeLink =
document.querySelector('.nav-link.active');
                    if (activeLink) {
                        const { left, width } =
activeLink.getBoundingClientRect();
                        return {
                            left: `${left}px`,
                            width: `${width}px`,
                        };
                    }
                    return {};
                },
            },
            mounted() {
                setTimeout(() => {
                    console.log(this.accessedTab)
                    if (this.accessedTab !== null) {
                        this.currentTab =
this.accessedTab;
                    } else {
                        this.currentTab =
'date_personale';
                    }
                }, 500);
            },
            methods: {
                changeTab(tab) {
                    this.currentTab = tab;
                    this.$emit('changeTab', tab);
                }
            }
        }
    </script>

<style scoped>
    .nav-container {
        border-bottom: 1px solid #ccc; /* Add a
border as a separator */
    }

    .nav-underline {
        display: flex; /* Display items in a row
*/
        justify-content: space-between; /* Add
space between items */
        padding: 0;
        list-style: none;
    }

    .nav-item {
        flex: 1; /* Distribute the width evenly
among items */
    }

    .nav-link {
        display: block;
        padding: 10px;
        text-align: center;
        text-decoration: none;
        color: #000;
    }

    .nav-link.active {
        font-weight: bold;
    }
</style>

<template>
<div class="sign-up-container">
    <div class="sign-up">

```

```

<h2>Reseteaza parola</h2>
<form @submit.prevent="reset">
  <div class="sign-up-input">
    <input type="password" v-
model="parola" @input="validatePassword"
required="">
    <label>Parola noua</label>
  </div>
  <div class="sign-up-input">
    <input type="password" v-
model="confirmare_parola" required="">
    <label>Confirmare
parola</label>
  </div>
  <button type="submit" class="sign-
up-button">Resetare</button>
</form>
</div>
</template>

```

```

<script>
import { mapMutations } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {
  name: "Signup",
  props: ['data'],
  data() {
    return {
      parola: null,
      confirmare_parola: null,
    },
  },
  mounted() {
    console.log(this.data);
  },
  methods: {
    reset() {
      if (!this.parolaValidata) {
        alert('Parola trebuie sa
continua cel putin o litera mare, o litera
mica, un numar si un caracter special!');
        return;
      }
      if (this.parola !==
this.confirmare_parola) {
        alert('Parolele nu coincid!');
        return;
      }
      axios.put('/reset-password', {
        email: this.data,
        parola: this.parola,
      })
        .then(response => {
          window.location.href =
'/principala';
        })
        .catch(error => {
          console.log(error);
        })
    },
    validatePassword() {
      const uppercaseRegex = /[A-Z]/;
      const lowercaseRegex = /[a-z]/;
      const numberRegex = /[0-9]/;
      const specialCharRegex = /[-
!$%^&*()_+|~='{}[\]:";'<>?,.\\"/];
      this.parolaValidata =
uppercaseRegex.test(this.parola) &&

```

```

lowercaseRegex.test(this.parola) &&
numberRegex.test(this.parola)
&&
specialCharRegex.test(this.parola);
    }
  }
}
</script>

<style scoped>
.sign-up-container {
  display: flex;
  justify-content: center;
  align-items: center;
  margin-top: 100px;
  margin-bottom: 100px;
}

.sign-up {
  width: 420px;
  padding: 50px;
  /*background: #F2EFE9;*/
  background: #fafafa;
  box-sizing: border-box;
  box-shadow: 0 15px 25px rgba(0,0,0,0.1);
  border-radius: 10px;
}

.sign-up h2 {
  margin: 0 0 40px;
  padding-bottom: 20px;
  color: #09554c;
  text-align: center;
}

.sign-up .sign-up-input {
  position: relative;
}

.sign-up .sign-up-input input {
  width: 100%;
  padding: 6px 0;
  font-size: 14px;
  color: #09554c;
  margin-bottom: 30px;
  border: none;
  border-bottom: 1px solid #09554c;
  outline: none;
  background: transparent;
}

.sign-up .sign-up-input label {
  position: absolute;
  top: 0;
  left: 0;
  padding: 6px 0;
  font-size: 14px;
  color: #09554c;
  pointer-events: none;
  transition: .5s;
}

.sign-up .sign-up-input input:focus ~ label,
.sign-up .sign-up-input input:valid ~ label {
  top: -20px;
  left: 0;
  color: #333333;
  font-size: 12px;
}

.sign-up-button {
  display: block;
  margin: 40px auto 48px;
  padding: 14px 60px;
  background: linear-gradient(to left,
#80CBC4, #008b7a);
  color: #ffffff;

```

```

font: inherit;
border: none;
cursor: pointer;
border-radius: 8px;
}

.sign-up-button:hover {
background: linear-gradient(to right,
#80CBC4, #008b7a);
}

.sign-up-switch {
display: block;
text-align: center;
font-size: 16px;
color: #333333;
}

.sign-up-switch a {
text-decoration: none;
color: #26A69A;
}

.sign-up-switch a:hover {
text-decoration: underline;
}

</style>

<template>
<div>
<h4
class="titlu"><strong>Rating</strong> {{
book.titlu }}</h4>
<div :style="{ width: 50 + '%',
marginBottom: 30 + 'px' }">
<h5 class="titlu"><strong>{{
book.rating }}</strong></h5>

<div class="row">
<div class="col-md-auto"
:style="{ width: 12 + 'px', marginBottom: 10 +
'px' }" v-for="i in 5" :key="i">

</div>
</div>
<div class="progress"
role="progressbar" aria-label="Basic example"
aria-valuenow="0" aria-valuemin="0" aria-
valuemax="100">
<div class="progress-bar"
:style="{ width: calculateWidth(5) + '%'}">
<span v-if="statistics[5]"
!= 0">{{ statistics[5] }}</span>
</div>
</div>

<div class="row">
<div class="col-md-auto"
:style="{ width: 12 + 'px', marginBottom: 10 +
'px' }" v-for="i in 5" :key="i">


</div>
</div>
<div class="progress"
role="progressbar" aria-label="Basic example"
aria-valuenow="25" aria-valuemin="0" aria-
valuemax="100">
<div class="progress-bar"
:style="{ width: calculateWidth(4) + '%'}">
<span v-if="statistics[4]"
!= 0">{{ statistics[4] }}</span>

```

```

</div>
</div>

<div class="row">
<div class="col-md-auto"
:style="{ width: 12 + 'px', marginBottom: 10 +
'px' }" v-for="i in 5" :key="i">


</div>
</div>
<div class="progress"
role="progressbar" aria-label="Basic example"
aria-valuenow="50" aria-valuemin="0" aria-
valuemax="100">
<div class="progress-bar"
:style="{ width: calculateWidth(3) + '%'}">
<span v-if="statistics[3]"
!= 0">{{ statistics[3] }}</span>
</div>
</div>

<div class="row">
<div class="col-md-auto"
:style="{ width: 12 + 'px', marginBottom: 10 +
'px' }" v-for="i in 5" :key="i">


</div>
</div>
<div class="progress"
role="progressbar" aria-label="Basic example"
aria-valuenow="75" aria-valuemin="0" aria-
valuemax="100">
<div class="progress-bar"
:style="{ width: calculateWidth(2) + '%'}">
<span v-if="statistics[2]"
!= 0">{{ statistics[2] }}</span>
</div>
</div>

<div class="row">
<div class="col-md-auto"
:style="{ width: 12 + 'px', marginBottom: 10 +
'px' }" v-for="i in 5" :key="i">


</div>
</div>
<div class="progress"
role="progressbar" aria-label="Basic example"
aria-valuenow="100" aria-valuemin="0" aria-
valuemax="100">
<div class="progress-bar"
:style="{ width: calculateWidth(1) + '%'}">
<span v-if="statistics[1]"
!= 0">{{ statistics[1] }}</span>
</div>
</div>
<div v-for="review in reviews"
:key="review.id">
<div class="card">
<p><strong>{{ review.titlu
}}</strong></p>
<p>{{ review.comentariu }}
</p>

```



```

        headers: {
          'Authorization':
'Bearer ' + this.isAuthenticated
        }
      })
      .then(response => {
        this.closePopup();
        this.getReview();
        window.location.reload()
      })
      .catch(error => {
        console.log(error);
      });
    },
    calculateWidth(index) {
      if (this.book.nr_reviewuri === 0)
      {
        return 0;
      } else {
        return this.statistics[index]
/ this.book.nr_reviewuri * 100;
      }
    }
  }
}
</script>

<style scoped>
@font-face {
  font-family: 'Lora';
  src: url('/Fonts/static/Lora-
Regular.ttf');
}

.card {
  margin-bottom: 30px;
  border: none;
  font-family: "Lora", serif;
  background-color: #FAFAFA;
}

.titlu {
  font-family: "Lora", serif;
  color: #333333;
  margin-left: 5px;
  margin-bottom: 30px;
}

.progress {
  margin-bottom: 10px;
}

.progress-bar {
  background-color: #008b7a;
}

.btn-primary {
  display: block;
  background-color: #00A896;
  border: none;
  width: auto;
}

.popup-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
}

.popup-content {
  background-color: #fff;
  padding: 20px;
  border-radius: 4px;

```

```

}
</style>

<template>
<div class="sign-up-container">
  <div class="sign-up">
    <h2>Creeaza cont</h2>
    <form @submit.prevent="signUp">
      <div class="sign-up-input">
        <input type="text" v-
model="nume" required title="">
        <label>Nume</label>
      </div>
      <div class="sign-up-input">
        <input type="text" v-
model="prenume" required="">
        <label>Prenume</label>
      </div>
      <div class="sign-up-input">
        <input type="email" v-
model="email" required="">
        <label>E-mail</label>
      </div>
      <div class="sign-up-input">
        <input type="tel" v-
model="telefon" required="">
        <label>Telefon</label>
      </div>
      <div class="sign-up-input">
        <input type="text" v-
model="judet" required="">
        <label>Judet</label>
      </div>
      <div class="sign-up-input">
        <input type="text" v-
model="oras" required="">
        <label>Localitate</label>
      </div>
      <div class="sign-up-input">
        <input type="text" v-
model="adresa" required="">
        <label>Adresa</label>
      </div>
      <div class="sign-up-input">
        <input type="password" v-
model="parola" @input="validatePassword"
required="">
        <label>Parola</label>
      </div>
      <div class="sign-up-input">
        <input type="password" v-
model="confirmare_parola" required="">
        <label>Confirmare
parola</label>
      </div>
      <button type="submit" class="sign-
up-button">Inregistrare</button>
      <span class="sign-up-switch">
        Ai deja cont? <a
href="/autentificare">Logheaza-te</a>
      </span>
    </form>
  </div>
</div>
</template>

<script>
import { mapMutations } from 'vuex';
import axios from 'axios';
const csrfToken =
document.querySelector('meta[name="csrf-
token"]').content;
axios.defaults.headers.common = {
  'X-Requested-With': 'XMLHttpRequest',
  'X-CSRF-TOKEN': csrfToken
};
export default {

```

```

name: "Signup",
data() {
  return {
    nume: null,
    prenume: null,
    email: null,
    telefon: null,
    judet: null,
    oras: null,
    adresa: null,
    parola: null,
    confirmare_parola: null,
    parolaValidata: false,
  },
  methods: {
    signUp() {
      if (this.parola !==
this.confirmare_parola) {
        alert('Parolele nu coincid!');
        return;
      }
      if (!this.parolaValidata) {
        alert('Parola trebuie sa
continua cel putin o litera mare, o litera
mica, un numar si un caracter special!');
        return;
      }
      console.log("ADS")
      axios.post('/user', {
        nume: this.nume,
        prenume: this.prenume,
        email: this.email,
        nr_telefon: this.telefon,
        judet: this.judet,
        oras: this.oras,
        adresa: this.adresa,
        parola: this.parola,
      })
        .then(response => {
          const token =
response.data.token;

this.$store.commit('setToken', token);

localStorage.setItem('token', token);
window.location.href =
'/cautare';
        })
        .catch(error => {
          console.log(error);
        })
      },
      validatePassword() {
        const uppercaseRegex = /[A-Z]/;
        const lowercaseRegex = /[a-z]/;
        const numberRegex = /[0-9]/;
        const specialCharRegex = /[-
!$%^&*()_+|~='{}[\]:";'<>?,.\|/];

        this.parolaValidata =
uppercaseRegex.test(this.parola) &&
lowercaseRegex.test(this.parola) &&
numberRegex.test(this.parola)
&&
specialCharRegex.test(this.parola);
      }
    }
  }
}
</script>

<style scoped>
.sign-up-container {
  display: flex;
  justify-content: center;

```

```

    align-items: center;
    margin-top: 100px;
    margin-bottom: 100px;
  }

.sign-up {
  width: 420px;
  padding: 50px;
  /*background: #F2EFE9;*/
  background: #fafafa;
  box-sizing: border-box;
  box-shadow: 0 15px 25px rgba(0,0,0,0.1);
  border-radius: 10px;
}

.sign-up h2 {
  margin: 0 0 40px;
  padding-bottom: 20px;
  color: #09554c;
  text-align: center;
}

.sign-up .sign-up-input {
  position: relative;
}

.sign-up .sign-up-input input {
  width: 100%;
  padding: 6px 0;
  font-size: 14px;
  color: #09554c;
  margin-bottom: 30px;
  border: none;
  border-bottom: 1px solid #09554c;
  outline: none;
  background: transparent;
}

.sign-up .sign-up-input label {
  position: absolute;
  top: 0;
  left: 0;
  padding: 6px 0;
  font-size: 14px;
  color: #09554c;
  pointer-events: none;
  transition: .5s;
}

.sign-up .sign-up-input input:focus ~ label,
.sign-up .sign-up-input input:valid ~ label {
  top: -20px;
  left: 0;
  color: #333333;
  font-size: 12px;
}

.sign-up-button {
  display: block;
  margin: 40px auto 48px;
  padding: 14px 60px;
  background: linear-gradient(to left,
#80CBC4, #008b7a);
  color: #ffffff;
  font: inherit;
  border: none;
  cursor: pointer;
  border-radius: 8px;
}

.sign-up-button:hover {
  background: linear-gradient(to right,
#80CBC4, #008b7a);
}

.sign-up-switch {
  display: block;
  text-align: center;

```

```

        font-size: 16px;
        color: #333333;
    }

    .sign-up-switch a {
        text-decoration: none;
        color: #26A69A;
    }

    .sign-up-switch a:hover {
        text-decoration: underline;
    }
</style>

```

Anexa 7 – Configurări Javascript

```

/**
 * First we will load all of this project's
 * JavaScript dependencies which
 * includes Vue and other libraries. It is a
 * great starting point when
 * building robust, powerful web applications
 * using Vue and Laravel.
 */

require('./bootstrap');

import store from './store';
import Carousel from 'vue-carousel';

window.Vue = require('vue').default;

/**
 * The following block of code may be used to
 * automatically register your
 * Vue components. It will recursively scan
 * this directory for the Vue
 * components and automatically register them
 * with their "basename".
 *
 * Eg. ./components/ExampleComponent.vue ->
 * <example-component></example-component>
 */

// const files = require.context('./', true,
// /\.vue$/i)
// files.keys().map(key =>
Vue.component(key.split('/')[1].pop().split('.')[0], files(key).default))

Vue.use(Carousel);

Vue.component('navigation',
require('./components/Navigation.vue').default
);
Vue.component('book-filter',
require('./components/BookFilter.vue').default
);
Vue.component('book-list',
require('./components/BookList.vue').default);
Vue.component('sign-up',
require('./components/SignUp.vue').default);
Vue.component('log-in',
require('./components/LogIn.vue').default);
Vue.component('donation',
require('./components/DonationForm.vue').default);
Vue.component('books',
require('./components/Books.vue').default);

```

```

Vue.component('profile',
require('./components/Profile.vue').default);
Vue.component('profile-navigation',
require('./components/ProfileNavigation.vue').default);
Vue.component('my-profile',
require('./components/MyProfile.vue').default);
;
Vue.component('book-details',
require('./components/BookDetails.vue').default);
Vue.component('book-carousel',
require('./components/BookCarousel.vue').default);
Vue.component('book-page',
require('./components/BookPage.vue').default);
Vue.component('review',
require('./components/Review.vue').default);
Vue.component('main-page',
require('./components/MainPage.vue').default);
Vue.component('reset-password',
require('./components/ResetPassword.vue').default);
Vue.component('my-favorites',
require('./components/MyFavorites.vue').default);
Vue.component('contact-form',
require('./components/ContactForm.vue').default);
Vue.component('cos',
require('./components/Cos.vue').default);
Vue.component('my-orders',
require('./components/MyOrders.vue').default);
Vue.component('my-products',
require('./components/MyProducts.vue').default);
);

/**
 * Next, we will create a fresh Vue
 * application instance and attach it to
 * the page. Then, you may begin adding
 * components to this application
 * or customize the JavaScript scaffolding to
 * fit your unique needs.
 */

const app = new Vue({
    el: '#app',
    store,
});

window._ = require('lodash');

try {
    require('bootstrap');
} catch (e) {}

/**
 * We'll load the axios HTTP library which
 * allows us to easily issue requests
 * to our Laravel back-end. This library
 * automatically handles sending the
 * CSRF token as a header based on the value
 * of the "XSRF" token cookie.
 */

window.axios = require('axios');

window.axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest';

/**
 * Echo exposes an expressive API for
 * subscribing to channels and listening
 * for events that are broadcast by Laravel.
 * Echo and event broadcasting
 * allows your team to easily build robust
 * real-time web applications.
 */

```



```
// import Echo from 'laravel-echo';

// window.Pusher = require('pusher-js');

// window.Echo = new Echo({
//     broadcaster: 'pusher',
//     key: process.env.MIX_PUSHER_APP_KEY,
//     cluster:
// process.env.MIX_PUSHER_APP_CLUSTER,
//     forceTLS: true
// });

import Vue from 'vue';
import Vuex from 'vuex';

Vue.use(Vuex);

const store = new Vuex.Store({
  state: {
    token: localStorage.getItem('token')
  || null, // Initialize token from localStorage
  or as null
  },
  mutations: {
    setToken(state, token) {
      state.token = token;
      localStorage.setItem('token',
token); // Store the token in localStorage
    },
    clearToken(state) {
      state.token = null;
      localStorage.removeItem('token');
    }
  },
  actions: {
    // You can define additional actions
    here if needed
  },
  getters: {
    isAuthenticated: (state) => {
      return state.token;
    }
  },
});

export default store;
```

Anexa 8 – Laravel Blade

```
<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRhO002iuK6FUUVM" crossorigin="anonymous">
</head>
<body>
<div id="app">
  <navigation class="padding-
main"></navigation>
```

```
<admin-user-table class="margin-main"
:style="{ width: 1350 + 'px' }"></admin-user-
table>
<admin-carte-table class="margin-main"
:style="{ width: 1350 + 'px' }"></admin-carte-
table>
<admin-categorie-table class="margin-main"
:style="{ width: 1350 + 'px' }"></admin-
categorie-table>
<admin-comanda-table class="margin-main"
:style="{ width: 1350 + 'px' }"></admin-
comanda-table>
<admin-donatie-table class="margin-main"
:style="{ width: 1350 + 'px' }"></admin-
donatie-table>
<admin-review-table class="margin-main"
:style="{ width: 1350 + 'px' }"></admin-
review-table>
</div>
</body>
```

```
<script src="{{ mix('js/app.js') }}"></script>
```

```
<style>
```

```
body {
  margin: 0;
  padding: 0;
  font-family: 'Lora', serif;
  /*background: #eaf3fa;*/
  /*background: #e8e5df;*/
  background: #EDF6F6;
  overflow-y: scroll;
}

.margin-main {
  margin-left: calc(50% - 675px);
  margin-right: calc(50% - 675px);
}

.padding-main {
  padding-left: calc(50% - 645px);
  padding-right: calc(50% - 655px);
}

::-webkit-scrollbar-track {
  background-color: #F0F1F2;
}

::-webkit-scrollbar-thumb {
  background-color: #00A896;
  border: 3px solid #00A896;
}

::-webkit-scrollbar-thumb:hover {
  background-color: #00A896;
}

::-webkit-scrollbar-corner {
  background-color: #00A896;
}
</style>
```

```
<title> Log In </title>
```

```
<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRhO002iuK6FUUVM" crossorigin="anonymous">
```

```

</head>
<body>
  <div id="app">
    <navigation class="padding-
main"></navigation>
    <log-in></log-in>
  </div>
</body>

<script src="{{ mix('js/app.js') }}"></script>

<style>

  body {
    margin: 0;
    padding: 0;
    font-family: 'Lora', serif;
    /*background: #eaf3fa;*/
    /*background: #e8e5df;*/
    background: #EDF6F6;
  }

  .margin-main {
    margin-left: calc(50% - 675px);
    margin-right: calc(50% - 675px);
  }

  .padding-main {
    padding-left: calc(50% - 645px);
    padding-right: calc(50% - 655px);
  }

  ::-webkit-scrollbar-track {
    background-color: #F0F1F2;
  }

  ::-webkit-scrollbar-thumb {
    background-color: #00A896;
    border: 3px solid #00A896;
  }

  ::-webkit-scrollbar-thumb:hover {
    background-color: #00A896;
  }

  ::-webkit-scrollbar-corner {
    background-color: #00A896;
  }
</style>

<title> Log In </title>

<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ
2cdeRh002iuK6FUUVM" crossorigin="anonymous">
</head>
<body>
<div id="app">
  <navigation class="padding-
main"></navigation>
  <book-page class="margin-main" :book="{{
$data }}"></book-page>
</div>
</body>

<script src="{{ mix('js/app.js') }}"></script>

```

```

<style>
  body{
    padding: 0;
    font-family: 'Lora', serif;
    background: #FAFAFA;
  }
  .margin-main {
    margin-left: calc(50% - 675px);
    margin-right: calc(50% - 675px);
  }
  .padding-main {
    padding-left: calc(50% - 645px);
    padding-right: calc(50% - 655px);
  }
  ::-webkit-scrollbar-track {
    background-color: #F0F1F2;
  }
  ::-webkit-scrollbar-thumb {
    background-color: #00A896;
    border: 3px solid #00A896;
  }
  ::-webkit-scrollbar-thumb:hover {
    background-color: #00A896;
  }
  ::-webkit-scrollbar-corner {
    background-color: #00A896;
  }
  ::-webkit-scrollbar {
    width: 10px;
    height: 10px;
  }
</style>

<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ
2cdeRh002iuK6FUUVM" crossorigin="anonymous">
</head>
<body>
<div id="app">
  <navigation class="padding-
main"></navigation>
  <books class="margin-main" :books="{{
$data }}"></books>
</div>
</body>

<script src="{{ mix('js/app.js') }}"></script>

<style>
  body{
    padding: 0;
    font-family: 'Lora', serif;
    background: #FAFAFA;
  }
  .margin-main {
    margin-left: calc(50% - 675px);
    margin-right: calc(50% - 675px);
    margin-top: 100px;
  }
  .padding-main {
    padding-left: calc(50% - 645px);

```

```

padding-right: calc(50% - 655px);
}

::-webkit-scrollbar-track {
background-color: #F0F1F2;
}

::-webkit-scrollbar-thumb {
background-color: #00A896;
border: 3px solid #00A896;
}

::-webkit-scrollbar-thumb:hover {
background-color: #00A896;
}

::-webkit-scrollbar-corner {
background-color: #00A896;
}

::-webkit-scrollbar {
width: 10px;
height: 10px;
}
</style>

<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
<title>Ink&Paper</title>
<meta charset="utf-8">
<meta name="viewport"
content="width=device-width, initial-scale=1">
<link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRh002iuK6FUUVM" crossorigin="anonymous">
</head>
<body>
<div id="app">
<navigation class="padding-
main"></navigation>
<cos class="margin-main"></cos>
</div>

</body>

<script src="{{ mix('js/app.js') }}"></script>

<style>
body{
padding: 0;
font-family: 'Lora', serif;
background: #FAFAFA;
}
.margin-main {
margin-left: calc(50% - 675px);
margin-right: calc(50% - 675px);
margin-top: 100px;
}
.padding-main {
padding-left: calc(50% - 645px);
padding-right: calc(50% - 655px);
}

::-webkit-scrollbar-track {
background-color: #F0F1F2;
}

::-webkit-scrollbar-thumb {
background-color: #00A896;
border: 3px solid #00A896;
}

```

```

::-webkit-scrollbar-thumb:hover {
background-color: #00A896;
}

::-webkit-scrollbar-corner {
background-color: #00A896;
}

::-webkit-scrollbar {
width: 10px;
height: 10px;
}
</style>

<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
<title>Ink&Paper</title>
<meta charset="utf-8">
<meta name="viewport"
content="width=device-width, initial-scale=1">
<link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRh002iuK6FUUVM" crossorigin="anonymous">
</head>
<body>

<div id="app">
<navigation class="padding-
main"></navigation>
<donation></donation>
</div>

</body>

<script src="{{ mix('js/app.js') }}"></script>

<style>

body {
margin: 0;
padding: 0;
font-family: 'Lora', serif;
/*background: #e8e5df;*/
background: #EDF6F6;
}

.margin-main {
margin-left: calc(50% - 675px);
margin-right: calc(50% - 675px);
}
.padding-main {
padding-left: calc(50% - 645px);
padding-right: calc(50% - 655px);
}

::-webkit-scrollbar-track {
background-color: #F0F1F2;
}

::-webkit-scrollbar-thumb {
background-color: #00A896;
border: 3px solid #00A896;
}

::-webkit-scrollbar-thumb:hover {
background-color: #00A896;
}

::-webkit-scrollbar-corner {
background-color: #00A896;
}

```

```

        ::-webkit-scrollbar {
            width: 10px;
            height: 10px;
        }
    </style>

<!DOCTYPE html>
<meta name="csrf-token" content="{ { csrf_token() } }">
<html>
<head>
    <title>Ink&Paper</title>
    <meta charset="utf-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
</head>
<body>
<div id="app">
    <navigation class="padding-
main"></navigation>
</div>

</body>

<script src="{ { mix('js/app.js') } }"></script>

<style>
    body{
        padding: 0;
        font-family: 'Lora', serif;
        background: #FAFAFA;
    }
    .margin-main {
        margin-left: calc(50% - 675px);
        margin-right: calc(50% - 675px);
    }
    .padding-main {
        padding-left: calc(50% - 645px);
        padding-right: calc(50% - 655px);
    }
    ::-webkit-scrollbar-track {
        background-color: #F0F1F2;
    }
    ::-webkit-scrollbar-thumb {
        background-color: #00A896;
        border: 3px solid #00A896;
    }
    ::-webkit-scrollbar-thumb:hover {
        background-color: #00A896;
    }
    ::-webkit-scrollbar-corner {
        background-color: #00A896;
    }
    ::-webkit-scrollbar {
        width: 10px;
        height: 10px;
    }
</style>
<!DOCTYPE html>
<meta name="csrf-token" content="{ { csrf_token() } }">
<html>
<head>
    <title>Ink&Paper</title>
    <meta charset="utf-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
</head>
<body>

```

```

href="https://cdn.jsdelivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
</head>
<body>

    <div id="app">
        <navigation class="padding-
main"></navigation>
        <sign-up></sign-up>
    </div>

</body>

<script src="{ { mix('js/app.js') } }"></script>

<style>

    body {
        margin: 0;
        padding: 0;
        font-family: 'Lora', serif;
        /*background: #e8e5df;*/
        background: #EDF6F6;
    }

    .margin-main {
        margin-left: calc(50% - 675px);
        margin-right: calc(50% - 675px);
    }
    .padding-main {
        padding-left: calc(50% - 645px);
        padding-right: calc(50% - 655px);
    }
    ::-webkit-scrollbar-track {
        background-color: #F0F1F2;
    }
    ::-webkit-scrollbar-thumb {
        background-color: #00A896;
        border: 3px solid #00A896;
    }
    ::-webkit-scrollbar-thumb:hover {
        background-color: #00A896;
    }
    ::-webkit-scrollbar-corner {
        background-color: #00A896;
    }
    ::-webkit-scrollbar {
        width: 10px;
        height: 10px;
    }
</style>

<!DOCTYPE html>
<meta name="csrf-token" content="{ { csrf_token() } }">
<html>
<head>
    <title>Ink&Paper</title>
    <meta charset="utf-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ
2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
</head>
<body>

```

```

<div id="app">
  <navigation class="padding-
main"></navigation>
  <main-page class="margin-main"></main-
page>
</div>

</body>

<script src="{{ mix('js/app.js') }}"></script>

<style>
  body{
    padding: 0;
    font-family: 'Lora', serif;
    background: #FAFAFA;
  }
  .margin-main {
    margin-left: calc(50% - 645px);
    margin-right: calc(50% - 645px);
  }
  .padding-main {
    padding-left: calc(50% - 645px);
    padding-right: calc(50% - 655px);
  }

  ::-webkit-scrollbar-track {
    background-color: #F0F1F2;
  }

  ::-webkit-scrollbar-thumb {
    background-color: #00A896;
    border: 3px solid #00A896;
  }

  ::-webkit-scrollbar-thumb:hover {
    background-color: #00A896;
  }

  ::-webkit-scrollbar-corner {
    background-color: #00A896;
  }

  ::-webkit-scrollbar {
    width: 10px;
    height: 10px;
  }
</style>

<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ
2cdeRh002iuK6FUUVM" crossorigin="anonymous">
</head>

<body>
<div id="app">
  <navigation class="padding-
main"></navigation>
  @if (!empty($ref))
    <profile class="margin-main"
style="margin-top: 100px" :refl="{{ $ref
}}"></profile>
  @else
    <profile class="margin-main"
style="margin-top: 100px"
:refl="null"></profile>
  @endif

```

```

</div>
</body>

<script>
</script>

<script src="{{ mix('js/app.js') }}">
  export default {
    data() {
      return {
        emittedData: null
      };
    },
    methods: {
      handleData(filtered) {
        this.emittedData = filtered;
      }
    }
  }
</script>

<style>
  body{
    padding: 0;
    font-family: 'Lora', serif;
    background: #FAFAFA;
  }
  .margin-main {
    margin-left: calc(50% - 675px);
    margin-right: calc(50% - 675px);
  }
  .padding-main {
    padding-left: calc(50% - 675px);
    padding-right: calc(50% - 675px);
  }

  ::-webkit-scrollbar-track {
    background-color: #F0F1F2;
  }

  ::-webkit-scrollbar-thumb {
    background-color: #00A896;
    border: 3px solid #00A896;
  }

  ::-webkit-scrollbar-thumb:hover {
    background-color: #00A896;
  }

  ::-webkit-scrollbar-corner {
    background-color: #00A896;
  }

  ::-webkit-scrollbar {
    width: 10px;
    height: 10px;
  }
</style>

<!DOCTYPE html>
<meta name="csrf-token" content="{{
csrf_token() }}">
<html>
<head>
  <title>Ink&Paper</title>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5
.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ
2cdeRh002iuK6FUUVM" crossorigin="anonymous">
</head>
<body>

```

```

<div id="app">
  <navigation class="padding-
main"></navigation>
  <reset-password :data="{{
$passwordReset->email }}"></reset-password>
</div>

</body>

<script src="{{ mix('js/app.js') }}"></script>

<style>
  body {
    margin: 0;
    padding: 0;
    font-family: 'Lora', serif;
    /*background: #e8e5df;*/
    background: #EDF6F6;
  }

  .margin-main {
    margin-left: calc(50% - 675px);
    margin-right: calc(50% - 675px);
  }

  .padding-main {
    padding-left: calc(50% - 645px);
    padding-right: calc(50% - 655px);
  }

  ::-webkit-scrollbar-track {
    background-color: #F0F1F2;
  }

  ::-webkit-scrollbar-thumb {
    background-color: #00A896;
    border: 3px solid #00A896;
  }

  ::-webkit-scrollbar-thumb:hover {
    background-color: #00A896;
  }

  ::-webkit-scrollbar-corner {
    background-color: #00A896;
  }

  ::-webkit-scrollbar {
    width: 10px;
    height: 10px;
  }

</style>

```

Anexa 9 – Laravel Mail Blade

```

@component('mail::message')
  # Va multumim pentru comanda!

  Va multumim pentru comanda plasata pe
site-ul nostru! Suntem incantati sa va avem
alaturi de noi.

  Comanda cu ID-ul {{ $id }} a fost plasata
cu succes.

  Daca aveti intrebari sau aveti nevoie de
asistenta, nu ezitati sa ne contactati.

  Multumim,
  Echipa Ink&Paper
@endcomponent

@component('mail::message')
  # CONTACT

```

```

Mesaj de la {{ $email }}

Mesajul:
{{ $message }}
@endcomponent

@component('mail::message')
  # Link pentru resetarea parolei

  Am primit o cerere de resetare a parolei
pentru contul dvs.

  {{ $user->email }}

  Pentru resetarea parolei, va rugam sa
accesati link-ul de mai jos:

  {{ $url }}

  Daca aveti intrebari sau aveti nevoie de
asistenta, nu ezitati sa ne contactati.

  Multumim,
  Echipa Ink&Paper
@endcomponent

@component('mail::message')
  # Bine ati venit pe site-ul nostru!

  Va multumim pentru inregistrarea pe site-
ul nostru! Suntem incantati sa va avem alaturi
de noi.

  Acestea sunt detaliile inregistrarii:

  {{ $user->email }}

  Pentru confirmarea contului, va rugam sa
accesati link-ul de mai jos:

  {{ $url }}

  Daca aveti intrebari sau aveti nevoie de
asistenta, nu ezitati sa ne contactati.

  Multumim,
  Echipa Ink&Paper
@endcomponent

```

Anexa 10 – Rute

```

<?php

use
App\Http\Controllers\CarteCategorieController;
use
App\Http\Controllers\CarteComandaController;
use App\Http\Controllers\CarteCosController;
use App\Http\Controllers\CategorieController;
use App\Http\Controllers\ClientController;
use App\Http\Controllers\ComandaController;
use App\Http\Controllers\Controller;
use App\Http\Controllers\DiscountController;
use App\Http\Controllers\DonatieController;
use App\Http\Controllers\FavoritController;
use App\Http\Controllers\ReviewController;
use App\Http\Controllers\ImportController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;

```

```

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes
| for your application. These
| routes are loaded by the
| RouteServiceProvider within a group which
| contains the "web" middleware group. Now
| create something great!
|
*/

Route::get('/', function () {
    return redirect('/principala');
});
Route::get('/inregistrare', function () {
    return view('pagina-inregistrare');
});
Route::get('/autentificare', function () {
    return view('pagina-autentificare');
});
Route::get('/donare', function () {
    return view('pagina-donare');
});
Route::get('/principala', function () {
    return view('pagina-principala');
});
Route::get('/profil', function () {
    return view('pagina-profil');
});
Route::get('/cart', function () {
    return view('pagina-cos');
});
Route::get('/admin', function () {
    return view('pagina-admin');
});

Route::group(['middleware' => ['auth:api']],
function () {

    Route::prefix('favorite')->group(function
() {
        Route::get('/{id}',
[FavoritController::class, 'show']);
        Route::get('/',
[FavoritController::class, 'index']);
        Route::post('/',
[FavoritController::class, 'store']);
        Route::delete('/{id}',
[FavoritController::class, 'destroy']);
    });

    Route::prefix('comenzi')->group(function
() {
        Route::get('/',
[ComandaController::class, 'index']);
        Route::get('/{id}',
[ComandaController::class, 'show']);
        Route::post('/',
[ComandaController::class, 'store']);
        Route::delete('/{id}',
[ComandaController::class, 'destroy']);
    });

    Route::prefix('reviewuri')->group(function
() {
        Route::put('/',
[ReviewController::class, 'update']);
        Route::post('/',
[ReviewController::class, 'store']);
        Route::delete('/{id}',
[ReviewController::class, 'destroy']);
    });

    Route::prefix('user')->group(function () {

        Route::get('/',
[UserController::class, 'show']);
        Route::put('/parola',
[UserController::class, 'changePassword']);
        Route::put('/',
[UserController::class, 'update']);
        Route::post('/logout',
[UserController::class, 'logout']);
    });

    Route::get('/comenzi-carti',
[ComandaController::class, 'carti']);
    Route::post('/return',
[CarteComandaController::class,
'returnOrder']);
    Route::get('/comanda-carti/{id_comanda}',
[ComandaController::class, 'cartiComanda']);
});

Route::get('/profil/{ref}',
[UserController::class, 'getProfileRef']);
Route::get('/carte/{isbn}',
[CarteController::class, 'getBookByIsbn']);
Route::get('/cautare',
[CarteController::class, 'paginaCautare'])->
name('search');
Route::post('/contact', [Controller::class,
'sendContactMail']);
Route::post('/checkout',
[ComandaController::class, 'checkout']);
Route::post('/mail', [UserController::class,
'sendMail']);
Route::get('/verify/{token}',
[UserController::class, 'verifyMail']);
Route::get('/reset/{token}',
[UserController::class,
'resetPasswordRedirect']);
Route::post('/login', [UserController::class,
'login']);
Route::post('/reset-request',
[UserController::class,
'sendPasswordRequest']);
Route::put('/reset-password',
[UserController::class, 'resetPassword']);
Route::post('/search',
[CarteController::class, 'search']);
Route::get('/client-autentificat',
[ClientController::class, 'getUserData']);
Route::get('/latest', [CarteController::class,
'getLatestCarti']);
Route::get('/discounted',
[CarteController::class,
'getDiscountedCarti']);
Route::get('/discounts',
[DiscountController::class, 'index']);
Route::post('/returnari',
[CategorieController::class, 'store']);
Route::post('/donatii',
[CategorieController::class, 'store']);

Route::prefix('carte-categorie')->
group(function () {
    Route::get('/',
[CarteCategorieController::class, 'index']);
    Route::post('/',
[CarteCategorieController::class, 'store']);
    Route::put('/',
[CarteCategorieController::class, 'update']);
    Route::delete('/{id}',
[CarteCategorieController::class, 'destroy']);
});

Route::prefix('user')->group(function () {
    Route::post('/', [UserController::class,
'store']);
    Route::get('/index',
[UserController::class, 'index']);
    Route::delete('/{id}',
[UserController::class, 'destroy']);
});

```

```

});

Route::prefix('filtre')->group(function () {
    Route::get('/', [CarteController::class,
'getFilters']);
    Route::post('/', [CarteController::class,
'queryBuilder']);
});

Route::prefix('clienti')->group(function () {
    Route::get('/', [ClientController::class,
'index']);
    Route::get('/{id}',
[ClientController::class, 'show']);
    Route::put('/', [ClientController::class,
'update']);
    Route::post('/', [ClientController::class,
'store']);
    Route::delete('/{id}',
[ClientController::class, 'destroy']);
    Route::get('/user',
[ClientController::class, 'userData']);
    Route::get('/comenzi',
[ClientController::class, 'comenzi']);
    Route::get('/favorite',
[ClientController::class, 'favorite']);
    Route::get('/cos',
[ClientController::class, 'userData']);
});

Route::prefix('reviewuri')->group(function ()
{
    Route::get('/', [ReviewController::class,
'index']);
    Route::get('/carte/{id}',
[ReviewController::class, 'getByBookId']);
    Route::get('/{id}',
[ReviewController::class, 'show']);
    Route::put('/', [ReviewController::class,
'update']);
    Route::post('/', [ReviewController::class,
'store']);
    Route::delete('/{id}',
[ReviewController::class, 'destroy']);
});

Route::get('/carte', [CarteController::class,
'index']);
Route::put('/carte', [CarteController::class,
'update']);
Route::post('/carte', [CarteController::class,
'store']);
Route::delete('/carte/{id}',
[CarteController::class, 'destroy']);

Route::prefix('carti')->group(function () {
    Route::get('/recomandari/{id}',
[CarteController::class,
'getRecommendations']);
    Route::get('/{id}',
[CarteController::class, 'show']);
    Route::get('/categorii/{id}',
[CarteController::class, 'categorii']);
    Route::get('/reviewuri/{id}',
[CarteController::class, 'reviewuri']);
});

Route::prefix('carti-comanda')->group(function
() {
    Route::get('/',
[CarteComandaController::class, 'index']);
    Route::get('/{id}',
[CarteComandaController::class, 'show']);
    Route::put('/',
[CarteComandaController::class, 'update']);
    Route::post('/',
[CarteComandaController::class, 'store']);
    Route::delete('/{id}',
[CarteComandaController::class, 'destroy']);
});

```

```

});

Route::prefix('categorii')->group(function ()
{
    Route::get('/',
[CategorieController::class, 'index']);
    Route::get('/{id}',
[CategorieController::class, 'show']);
    Route::put('/',
[CategorieController::class, 'update']);
    Route::post('/',
[CategorieController::class, 'store']);
    Route::put('/{id}',
[CategorieController::class, 'destroy']);
    Route::get('/carti/{id}',
[CategorieController::class, 'carti']);
});

Route::prefix('cos')->group(function () {
    Route::post('/',
[CarteCosController::class, 'store']);
    Route::put('/',
[CarteCosController::class, 'update']);
    Route::delete('/{id}',
[CarteCosController::class, 'destroy']);
    Route::get('/',
[CarteCosController::class, 'index']);
});

Route::prefix('import')->group(function () {
    Route::get('/carti',
[ImportController::class, 'importCarti']);
    Route::get('/categorii',
[ImportController::class, 'importCategorii']);
    Route::get('/useri',
[ImportController::class, 'importUseri']);
    Route::get('/reviews',
[ImportController::class, 'importReviews']);
    Route::get('/discounts',
[ImportController::class, 'importDiscounts']);
});

Route::prefix('admin-comenzi')->group(function
() {
    Route::get('/', [ComandaController::class,
'adminIndex']);
    Route::put('/', [ComandaController::class,
'update']);
});

Route::prefix('donatie')->group(function () {
    Route::get('/', [DonatieController::class,
'index']);
    Route::post('/',
[DonatieController::class, 'store']);
    Route::delete('/{id}',
[DonatieController::class, 'destroy']);
});

```

Anexa 11 – Fișier environment

```

APP_NAME="Ink&Paper"
APP_ENV=local
APP_KEY=base64:bM7ZKgrT0XLIg4hc6MekA87v52aEvss
HESfHGFRjWak=
APP_DEBUG=true
APP_URL=http://localhost

L5_SWAGGER_GENERATE_ALWAYS=true

LOG_CHANNEL=stack
LOG_DEPRECATEDATIONS_CHANNEL=null
LOG_LEVEL=debug

```



```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=librariandb
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.office365.com
MAIL_PORT=587
MAIL_USERNAME=inkpaper2023@hotmail.com
MAIL_PASSWORD=asd123ASD!
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=inkpaper2023@hotmail.com
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

JWT_SECRET=ImYfW7NxJaWEgkvMZPPzLNTgiF9xpvsvIXl
Bmpzmwa5CZXhs1ikMGf7lMWiYaleF
```