

UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENȚĂ

Booktopia

Aplicație web de gestionare a unui magazin online de cărți

COORDONATOR ȘTIINȚIFIC

Lect. dr. Carmen Chiriță

ABSOLVENT

Pîrvu Daniel - Cătălin

BUCUREȘTI

2019

Cuprins

1	Introducere	4
1.1	Tema lucrării	4
1.2	Aplicația Web	4
1.3	Motivația tehnologiilor	5
2	Noțiuni teoretice	7
2.1	Arhitectura	7
2.2	Design pattern - MVC	9
2.3	Back-end	10
2.3.1	Baza de date – MySQL	10
2.3.2	PHP	10
2.4	Front-end	11
2.4.1	HTML	11
2.4.2	CSS & Bootstrap	11
2.4.3	JavaScript & jQuery	12
2.4.4	AJAX	15
2.4.5	Plugins	16
3	Framework-ul Laravel	18
3.1	Introducere	18
3.2	De ce?	18
3.3	Structura fișierelor	19
3.4	Configurarea mediului de lucru	20
3.5	Email	20
3.6	Autentificare	20
3.7	Testabilitate	20
3.8	Routing	21
3.9	Sistemul de șabloane Blade	22
3.10	Sistemul de construcție a bazei de date	24
3.11	Construcția query-urilor & Eloquent ORM	24
4	Proiectarea aplicației	27
4.1	Conceperea bazei de date	27

4.2	Autentificarea	27
4.3	Formatul aplicației	28
4.4	Maginul	29
4.5	Pagina de prezentare a unei cărți	30
4.6	Coșul de cumpărături	31
4.7	Profilul personal	33
4.8	Panoul de control	34
4.8.1	Gestionarea utilizatorilor	35
4.8.2	Gestionarea cărților	36
4.8.3	Gestionarea categoriilor	39
4.8.4	Gestionarea stocul produselor	39
4.8.5	Gestionarea cererilor partenerilor	40
4.8.6	Vizualizarea comenzilor efectuate	42
5	Concluzii	43
6	Bibliografie	44
7	ANEXE	45

1 Introducere

1.1 Tema lucrării

S-a întâmplat vreodată să petreci minute bune pe un magazin online în căutarea unei cărți pe gustul tău? Aplicația Booktopia vine în ajutorul tău. Ea își propune să surprindă prin fragmente axate pe cerințele esențiale ale utilizatorilor în ceea ce privește alegerea unei cărți bune și contextul de unde aceasta poate fi procurată.

Afișează un design simplist, dar în același timp atractiv clienților ce doresc o imediată înțelegere a elementelor vizuale. Experiența petrecută pe aplicație dorește a fi una intuitivă, fiind totodată și motivația construirii în scopul eficientizării timpului petrecut.



Fig. 1.1 - Logo Cărturești



Fig. 1.2 - Logo Libris

Ea vine ca o alternativă a magazinelor online deja cunoscute ce activează precum Cărturești, Libris, ș.a., a căror parcurgere devine din ce în ce mai îngreunată de aglomerarea elementelor în pagina.

1.2 Aplicația Web

Aplicația web este formată din:

- Baza de date, în care este păstrată o cantitate mare de date de care aplicația are nevoie, precum datele utilizatorilor, produsele afișate, dar și lucruri ce țin de partea administrativă, de exemplu stocul produselor, ofertele, cererile efectuate de către partenerii magazinului și date necesare în realizarea unor rapoarte periodice.
- Platforma web, care afișează diverse elemente în pagină în funcție de rolul pe care îl joacă utilizatorul, de la listarea produselor ce pot fi achiziționate la configurarea diferitelor aspecte interne de către un administrator.

Tehnologiile folosite:

- Baza de date: MySQL
- Platforma Web: Aplicație separată în două mari componente: back-end și front-end dezvoltate în paralel. Front-end-ul a fost realizat cu ajutorul tehnologiilor HTML, CSS, JavaScript, la care se adaugă elemente ajutătoare precum jQuery și librării de jQuery, iar back-end-ul a fost realizat folosind PHP, framework-ul Laravel și, de asemenea, API-uri ce pun la dispoziție funcționalități variate menite să creeze o mai bună interacțiune a clienților cu aplicația.

1.3 Motivația tehnologiilor

O lucrare de licență are ca scop demonstrarea de către absolvent că este un specialist ce poate proiecta, implementa și gestiona un produs software de la cap la coadă.

Avantajul acestei lucrări este acela că include o serie de provocări printre care cele mai semnificative sunt capacitatea de a pune în valoare un framework modern și de a coda într-o manieră ce permite pe viitor o foarte ușoară îmbunătățire și mentenanță a aplicației.

Am decis să folosesc pentru implementare framework-ul de PHP numit Laravel. Acesta pune la dispoziție o gamă largă de facilități, de la un foarte bun sistem de organizare a fișierelor la o ușurință în scrierea și înțelegerea de către viitoare persoane a codului, de la o bună integrare a structurilor de date folosite în varii zone ale proiectului la un nivel de securitate ridicat.

Este un framework aflat mereu într-o continuă dezvoltare ce satisface nevoile și standardele software ale timpului prezent. Astfel din ce în ce mai multe companii ce livrează produse software aleg să îl folosească, ajungând treptat să ocupe locul I în topul celor mai folosite framework-uri de PHP și locul VII în topul framework-urilor folosite la nivel global.

PHP Framework Used for Project Use

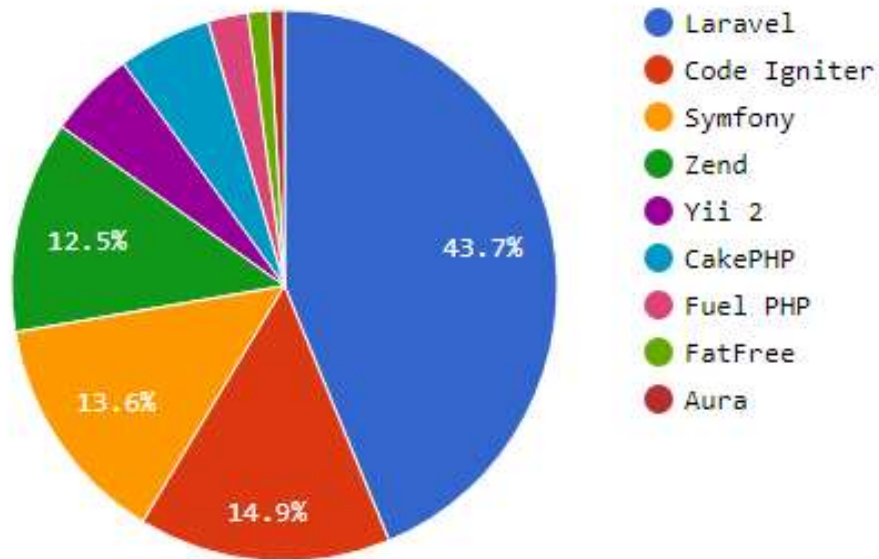


Fig. 1.3 - Topul framework-urilor web în PHP

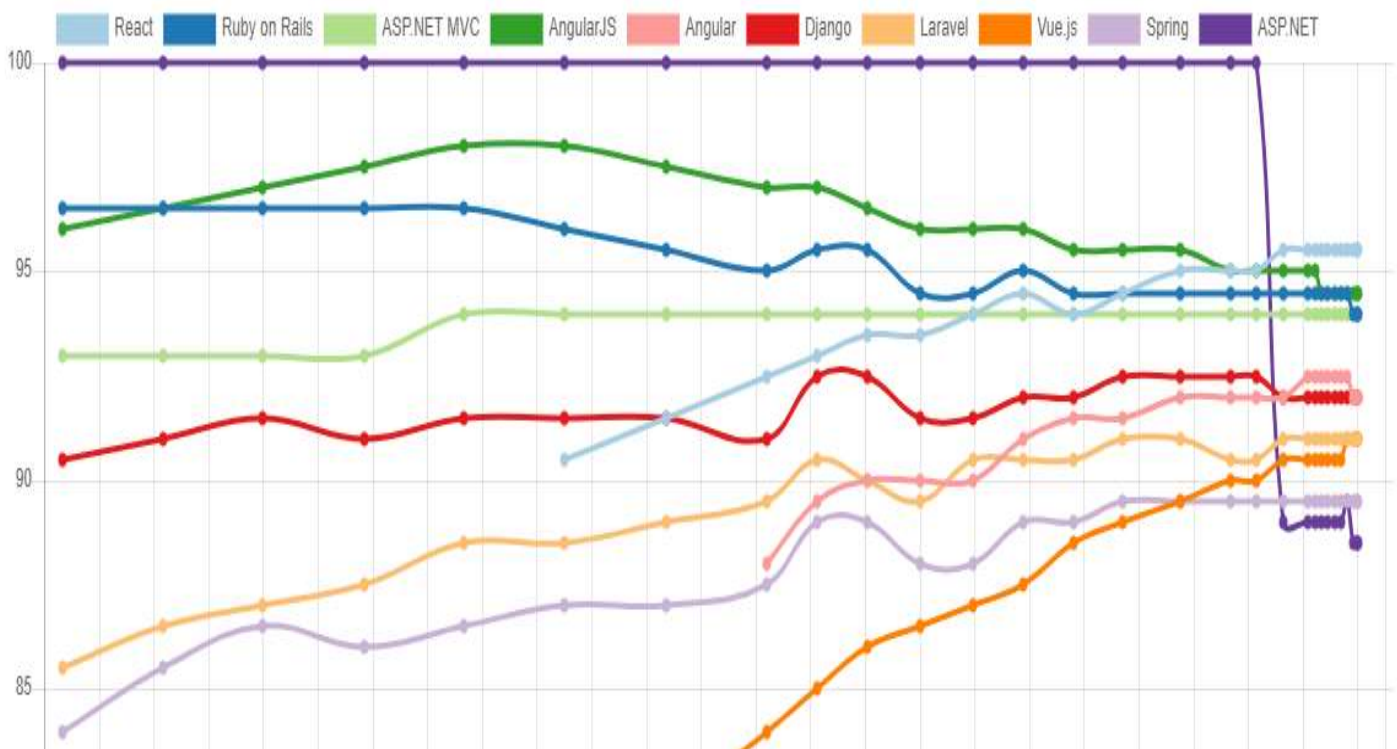


Fig. 1.4 - Topul framework-urilor web în general

2 Noțiuni teoretice

2.1 Arhitectura

În construcția aplicației am decis să folosesc o arhitectură pe niveluri (layered architecture). Aceasta este cea mai des folosită alternativă de către arhitecți și dezvoltatori, fiind unul din standardele întâlnite în companiile ce produc produse IT la nivel global.^[Vezi 1]

Este un sistem arhitectural ce asigură o organizare pe direcție verticală a nivelurilor conceptuale menite să performeze un rol specific în aplicație (e.g. prezentare sau procesare).

Deși arhitectura nu impune un număr specific de niveluri în componența sa, adesea acestea se rezumă la patru: prezentare, procesare, preluare, nivelul bazei de date. Există însă și cazuri în care nivelul de procesare este contopit cu nivelul de preluare, formând un singur nivel de procesare atunci când preluarea (e.g. SQL) este prezentă în componentele nivelului de procesare.

Responsabilitățile nivelurilor sunt foarte bine definite. Nivelul de procesare trebuie să fie capabil să execute instrucțiuni bazate pe contextul cererii formate de utilizator prin diferitele interacțiuni cu aplicația și cu browser-ul, interacțiuni de care nivelul prezentare este responsabil să le perceapă și să le trimită mai departe în ierarhie.

Prelucrarea unei cereri sugerează o abstractizare și o independență a nivelurilor arhitecturale în fiecare moment de timp. De exemplu, nivelul de prezentare este strict responsabil de afișarea pe ecran, într-un format specific, a datelor primite de la nivelul de procesare, fără a fi interesat de cum acestea au fost obținute. Similar, nivelul de procesare este strict responsabil de preluarea datelor de la nivelul corespunzător, de efectuare a operațiilor pe baza acestora și de pasare a rezultatelor la nivelul superior.

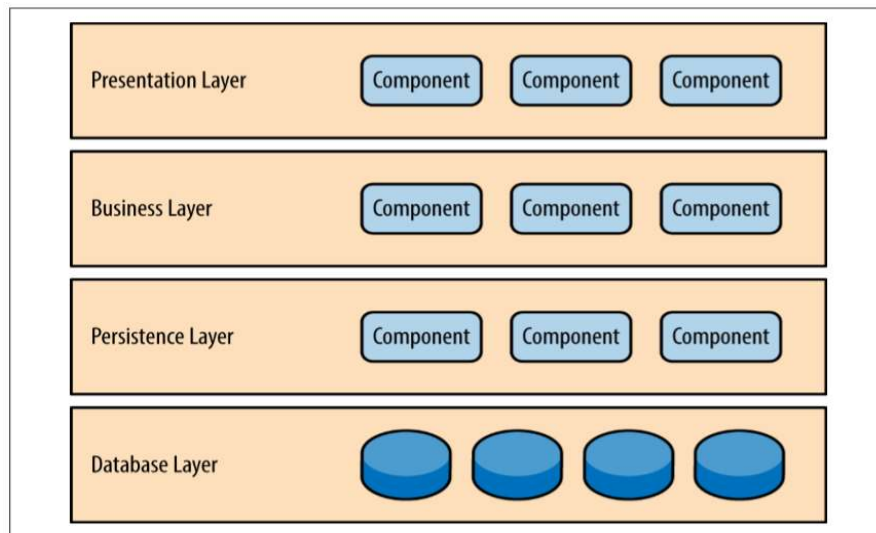


Fig. 2.1 - Structura arhitecturii pe niveluri

Un puternic avantaj al acestei arhitecturi îl constituie separarea sarcinilor în cadrul componentelor. Astfel fiecare componentă a unui nivel se preocupă doar de o mică parte din logica nivelului din care face parte.

Cheia arhitecturii este aceea că nivelurile sunt “închise” sau “izolate”. Conceptul arată că pentru o cerere, care necesită parcurgerea de la un nivel superior la unul inferior, este nevoie să trecem treptat prin fiecare nivel intermediar. Acest lucru asigură o bună mentenanță, astfel o schimbare într-o componentă a unui nivel nu afectează componentele oricărui alt nivel.

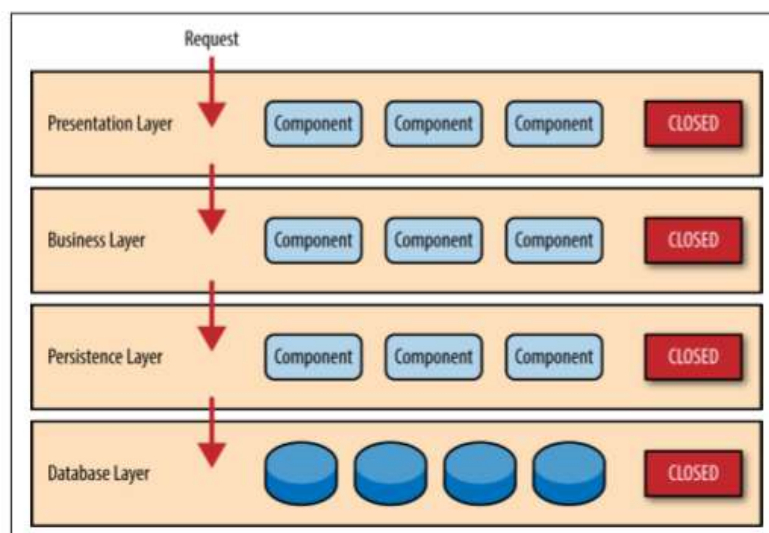


Fig. 2.2 - Niveluri închise și parcursul cererii

2.2 Design pattern - MVC

Adaptat la arhitectura pe niveluri, menționată mai devreme, este conceptul MVC. Acronimul vine de la Model – View – Controller ce definește cele 3 niveluri ale arhitecturii.

Gândit inițial cu scopul de a servi interfețelor grafice pentru aplicații desktop, a început să se extindă fiind în prezent popular în rândul aplicațiilor web, ba mai mult vine deja implementat în diferite framework-uri pentru limbaje ca JavaScript, Python, Ruby, PHP, etc.

Modelul este creierul aplicației, el asigură comunicarea cu nivelul bazei de date și este cel care impune o anumită structură a codului.

View-ul reprezintă componenta din prim plan pe care utilizatorul o vede și cu care poate interacționa în mod direct. Rolul esențial este de a creiona un aspect frumos în jurul unor date abstracte. În general este construit folosind limbaje precum HTML, CSS, JavaScript. El cuprinde de asemenea diferite implementări de funcții, dezvoltate în JavaScript, ce ajută la o bună experiență a utilizatorului pe platformă, însă fără a fi nevoie de date adiționale, deoarece ar implica o nouă accesare a bazei de date.

Controller-ul este sistemul nervos al aplicației, de aceea este responsabil de tot ce înseamnă preluarea cererilor utilizatorilor captate la nivelul superior, procesarea lor, accesul la eventuale noi informații necesare din baza de date prin intermediul modelului și trimiterea lor înapoi la nivelul de prezentare.

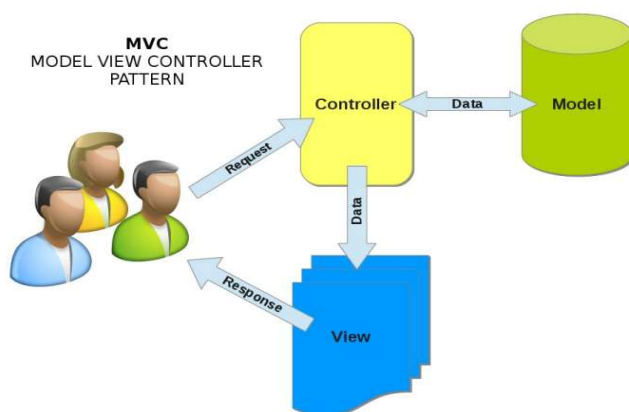


Fig. 2.3 - Structura Model – View – Controller

De ce MVC? Pentru că este un concept ce are sens doar în contextul aplicațiilor mari, cu o necesitate de procesare a unei cantități mari de date în diverse contexte. Ce are de oferit? Organizare, dezvoltare rapidă și paralelă, flexibilitate.^[Vezi RO 8]

2.3 Back-end

2.3.1 Baza de date – MySQL



Fig. 2.4 - Logo MySQL

În ceea ce privește implementările de stocare a datelor, sistemul de gestiune a bazelor de date MySQL este soluția. El este un sistem gratuit ce a ajuns să ocupe locul întâi în topul celor mai populare sisteme de gestiune a bazelor de date relaționale.

Deși poate fi regăsit în orice aplicație scrisă într-un limbaj major, este cel mai des întâlnit în aplicații implementate în PHP, fiind componenta esențială în tuplul LAMP (Linux, Apache, MySQL, PHP).

De asemenea, alegerea acestui tuplu pune la dispoziție dezvoltatorilor o unealtă de management al bazelor de date numită phpMyAdmin, a cărei scop este de a crea o cât mai ușoară interacțiune cu elementele prin interfața proprie.



Fig. 2.5 - Logo PHP

2.3.2 PHP

Limbajul PHP este un limbaj de scripting pe cât de simplu pe atât de puternic destinat să creeze conținut HTML. Originar conceput să genereze conținut web dinamic el este și astăzi o soluție în acest sens. Pentru a genera cod HTML, sunt necesare două elemente: un interpretator PHP și un server web prin care să trimitem fișiere de cod.^[Vezi 2]

Este un limbaj ce rulează pe toate sistemele de operare majore, de la suita Unix la Windows și Mac OS și poate fi folosit în relație cu toate serverele web precum Apache și Microsoft IIS.

Totodată este flexibil și asigură suport pentru toate sistemele de baze de date relaționale precum MySQL, PostgreSQL, Oracle și chiar și pentru cele non-relaționale precum mongoDB.

2.4 Front-end

2.4.1 HTML

Acronim pentru Hypertext Markup Language, acesta este un limbaj de marcare cu rolul de a defini structura elementelor sale componente. Aceste elemente (tag-uri) pot fi plasate în pagină în mod divers, putând astfel genera un comportament sau un aspect diferit de la situație la situație.

Elementele paginii au un schelet standard, fiind compuse din 3 mari părți: tag-urile de deschidere și închidere, ce marchează împreună tipul elementului folosit, dimensiunea blocului în care acesta are efect, atributele sale cu structura simplă `nume_atribut = "valoare_atribut"` și conținutul, ce poate lua diferite forme precum text clar sau alte elemente ale paginii creând efectul de imbricare. Există însă și cazuri în care ne abatem de la scheletul standard putând crea elemente ce au doar tag-ul de deschidere (e.g. elementul imagine sau elementele de introducere de valori existente în formulare).

Câteva exemple de elemente regăsite în componența oricărei pagini HTML ar fi:

- Elementul `<html>` ce reprezintă rădăcina paginii, în componența sa existând toate celelalte elemente.
- Elementul `<head>` este zona în care se află, în general, lucrurile pe care vrei să le incluzi în pagină dar fără a fi afișate. Aici sunt incluse fișierele CSS și JavaScript ce modelează elementele declarate.



Fig. 2.6 - Logo Bootstrap

2.4.2 CSS & Bootstrap

Cascading Style Sheets sau CSS este componenta ce ne permite să oferim un aspect plăcut paginilor web prezentate utilizatorilor. Un astfel de fișier CSS este alcătuit dintr-o serie de reguli cu o structură bine definită. Ele sunt formate din 2 mari părți: un selector, ce face referire la elementul sau elementele intenționate a fi stilizate și o serie de proprietăți aplicate acestora. Agentul responsabil de aplicarea CSS în document este browser-ul.

```
p {  
  color: red;  
}
```

Fig. 2.7 - Exemplu CSS

De exemplu, poza alăturată arată felul în care sunt selectate toate paragrafele din pagină și asupra conținutului lor se aplică culoarea roșie.

Voi prezenta în continuare framework-ul Bootstrap, fiind unul din cele mai des folosite noțiuni atunci când vorbim de stilizarea paginilor web.

Trecând prin multe schimbări, de la crearea lui, a ajuns astăzi să fie unul din cele mai stabile și adaptate framework-uri. Pentru dezvoltatorii web de toate nivelurile este o unealtă indispensabilă făcând capabilă crearea unui design atractiv și funcțional în câteva minute. Este ușor de folosit și ideal în condițiile impuse de timpul prezent, în care un website ar trebui să fie modern, adaptabil și axat în primă instanță vizualizării pe telefonul mobil.^[Vezi 4]

Bootstrap este cunoscut ca având unul din cele mai bune sisteme de gestionare a elementelor în pagină. El este construit cu ajutorul Flexbox și ajută la scalarea unui singur aspect pentru toate tipurile de rezoluții ale diferitelor dispozitive, de la cel mai mic telefon mobil la ecrane de dimensiuni mari. Esența lui este o împărțire a ecranului în 12 coloane și permiterea dezvoltatorului să aleagă câte din acestea să ocupe un anumit element.

Eficientizează munca oricui venind la pachet cu multe componente CSS și JavaScript reutilizabile, cu funcționalități necesare în mai toate website-urile.

Este un proiect open-source compatibil cu toate browser-ele, ce permite cutomizarea elementelor ce vin preinstalate.



Fig. 2.8 - Logo jQuery

2.4.3 JavaScript & jQuery

Deși face parte din grupul mare de limbaje de programare interpretate, JavaScript admite multe similitudi sintactice cu limbaje cunoscute precum C, C++ și Java. Cu toate acestea nu poate fi întru totul asemuit lor tinzând către comportamente precum cele întâlnite în Perl. Este un limbaj cu o slabă putere a tipurilor de date, ceea ce înseamnă că o variabilă nu are un tip de date stabil, iar accesarea și memorarea proprietăților unui obiect trimite cu gândul la hash tables din Perl în pofida structurilor din C++ și Java.^[Vezi 5]

Mediul comun în care activează este pe browser, de aceea scopul general este de a asigura interacțiunea utilizatorului cu website-ul, de a controla browser-ul și de a remodela conținutul ferestrei afișate. Acest mediu asigură rularea scripturilor legate și cu cele HTML direct pe calculatorul clientului și nu pe server-ul web.

Numit JavaScript client-side acesta combină abilitățile interpretatorului de JavaScript cu Document Object Model (DOM) definit de browser. Putem spune că fișierele ce conțin cod JavaScript folosesc DOM-ul pentru a modifica datele sau controlează browser-ul ce afișează conținutul, dând astfel un comportament unor obiecte web statice.

```
<html>
<head><title>Factorials</title></head>
<body>
<h2>Table of Factorials</h2>
<script>
var fact = 1;
for(i = 1; i < 10; i++) {
    fact = fact*i;
    document.write(i + "! = " + fact + "<br>");
}
</script>
</body>
</html>
```

Fig. 2.9 - Exemplu simplu de program JavaScript

Pentru o mai bună scriere, înțelegere și reducere a codului aplicației am decis să folosesc librăria de JavaScript numită jQuery, acesta fiind menită să faciliteze parsarea și manipularea conținutului HTML, gestiunea evenimentelor de browser, efectul animațiilor și interacțiunea prin intermediul AJAX-urilor.^[Vezi 6]

De ce jQuery? Pentru că este incredibil de popular, crescând pe zi ce trece comunitatea de dezvoltatori, săritori în ajutorul tău oricând e nevoie. Are o documentație foarte bine pusă la punct, disponibilă oricui și armonios explicată pentru toți dezvoltatorii indiferent de experiența proprie.

Este un punct de start pentru multe plugin-uri, fapt ce îl face responsabil și de rezolvarea conflictelor între diferite librării ce exista concomitent, dar și de posibilele conflicte de interpretare a codului de către diferite browser-e. Este o librărie solidă, consistentă, astfel îmbunătățirile nu întârzie să apară fără frica de modifica în mod eronat ce exista înainte.

Nu în ultimul rând elenaganța și filozofia de funcționare a schimbat modul de scriere a codului JavaScript devenind în sine un standard. Această filozosfie este “Scrie puțin, fă multe” ce poate fi concepută în 3 situații: preluarea elementelor pe baza selectorilor CSS și aplicarea unor funcții lor, înlănțuirea metodelor și folosirea obiectului jQuery în iterații implicate.

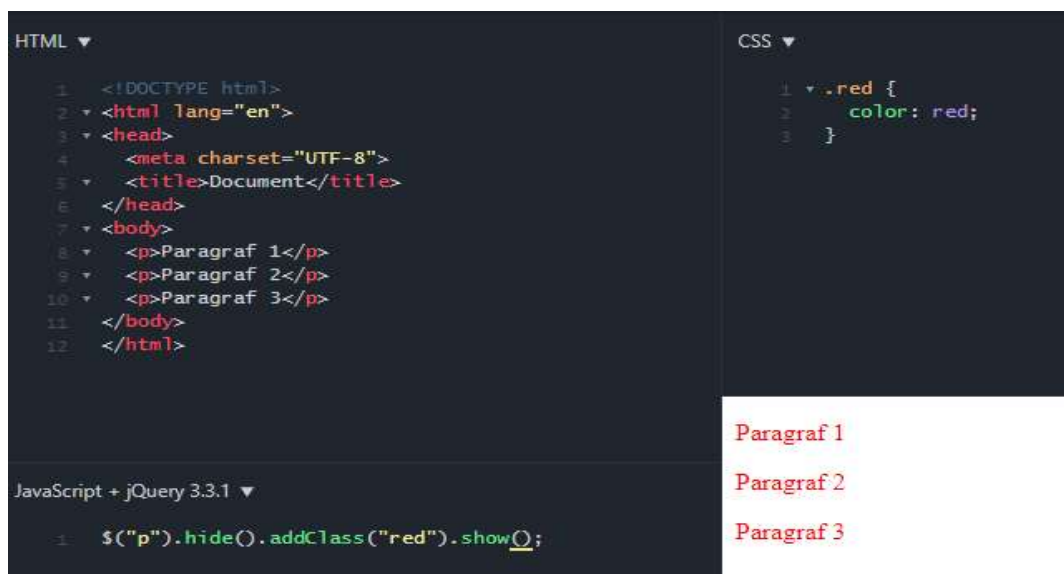


Fig. 2.10 - Exemplu cu HTML, CSS și jQuery

În acest exemplu se văd cele trei componente de bază ale librăriei. Cele 3 paragrafe sunt preluate cu ajutorul selectorului specific ("p"), apoi asupra lor sunt aplicate trei metode înlănțuite: una care ascunde elementele, una care adaugă clasa ce va colora textul roșu și ultima care va reafixa pe ecran cele 3 elemente preluate stilizate corespunzător. Deși am afirmat că efectele celor 3 metode se aplică tuturor paragrafelor, nu am folosit nicio structură repetitivă, acest lucru fiind întru totul asigurat de jQuery.

Printre metodele des întâlnite în mai toate fișierele jQuery se regăsesc:

- `.attr(nume_atribut) / .attr(nume_atribut, valoare_atribut)` : Această metodă este folosită cu dublu rol, în funcție de numărul de parametri, astfel execuția cu un parametru asigură întoarcerea valorii atributului cerut pe când cea cu doi setează valoarea atributului dat ca prim parametru cu cea din al doilea parametru. O adaptare a acestei metode este `.val(...)`, situație în care câmpurilor formularelor li se preia sau seta numele atributului "value".
- `.show() / .hide` : Perechea este des întâlnită în situații în care vrem să afișăm/ascundem elemente în pagină în funcție de contextul în care ne aflăm, dar fără a elimina elementul HTML din pagină. Un exemplu ar fi ascunderea diferitelor butoane după finalizarea tratării diferitelor evenimente.
- `.on(event, funcție)` : Metoda atașează colecției selectate o funcționalitate definită de programator în momentul declanșării evenimentului dat ca parametru.

2.4.4 AJAX



Fig. 2.11 - Logo AJAX

O altă structură ideală atunci când vorbim despre dezvoltare web separată pe cele două componente este AJAX. Prescurtare pentru Asynchronous JavaScript and XML, este o tehnică pentru crearea paginilor mult mai interactive și ușoare în utilizare. Tot odată viteza crește prin eliminarea nevoii de reîncărcare a paginii și prin adăugarea comunicării cu server-ul prin cantități mici de date.

Librăria jQuery oferă și aici un format standardizat și ușor adaptabil la nevoile dezvoltatorilor.

Astfel un obiect de tip AJAX se reduce la un obiect de tip JavaScript (cheie – valoare) cu diferite proprietăți și metode. Printre acestea se numără:

- **headers:** Este, la rândul lui, un obiect ce conține perechi cheie - valoare, ce se adaugă în prefața datelor trimise către server.
- **data:** Este blocul principal ce cuprinde câmpurile și valorile acestora ce doresc a fi procesate.
- **datatype:** Este tipul de date așteptat a fi întors. Dacă nu este specificat, jQuery va face o apreciere în această sens, alegând unul din posibilele structuri: "xml", "html", "script", "json", "text".
- **type:** Marchează tipul cererii și poate lua fi unul din lista: "GET", "POST", "PUT/PATCH", "DELETE". Valoarea predefinită este GET.
- **url:** Marchează calea în care se îndreaptă cererea.
- **success:** Este o funcție apelată, după ce rolul cererii a fost atins cu succes. Această funcție poate primi și un parametru ce reprezintă colecția de date cu care serverul răspunde.
- **error:** Funcție opusă celei precedente, fiind apelată atunci în decursul procesării cererii a intervenit o eroare.

```
$.ajax({
  headers: {'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')},
  type: "POST",
  url: "/users/get-user-role",
  data: $data,
  success: function (response) {
    $('.small-view .form input[type=hidden]').val(response.id);
    $('.small-view .form input[name=name]').val(response.name);
    $('.small-view .form input[name=email]').val(response.email);
    $('.small-view .form select#roleSelect option').toArray().forEach(function (elem) {
      if ($(elem).text().trim() === response.roles[0].name)
        $(elem).attr('selected', true);
    });

    $('.users_page_right_panel_img').hide('slow');
    $('.users_page_right_panel').show('slow');
  },
  error: function () {
    toastr.error('Error on getting user role!');
  }
});
```

Fig. 2.12 - Exemplu AJAX post

Figura precedentă arată un exemplu de AJAX folosit pentru a prelua date despre utilizator și pentru a popula, pe baza lor, un formular. Asupra câmpurilor necesare, regăsite în obiectul data, se va adauga token-ul CSRF pentru a crea siguranță împotriva atacurilor de tip Cross-Site Request Forgery. Cererea va fi trimisă folosind metoda POST, ce asigură o transparență și o securitate cererii, către url-ul specificat, după care, dacă serverul nu întâmpină probleme, se vor întoarce datele cerute și se vor seta valorile câmpurilor din formular sau se va afișa un mesaj.

2.4.5 Plugins



DataTables

Fig. 2.13 - Logo DataTables

DataTables este o librărie, bazată pe jQuery, ce dă dezvoltatorilor posibilitatea de a construi un aspect avansat instantaneu. Este o unealtă foarte bine documentată, flexibilă ce ușurează enorm munca depusă în stilizarea și funcționalitatea unui tabel. Aceasta oferă posibilitatea generării pe baza datelor din tabel a mai multor elemente precum paginație, search-bar adaptiv, sortarea datelor în funcție de criterii puse asupra coloanelor. Este simplu, dar în același timp plăcut vizual, indiferent de aspectul ecranului de pe dispozitivul accesat, pentru acestea colaborând și cu ultimele versiuni de Bootstrap. Inițializarea este simplă, fiind nevoie doar de selectarea tabelului și aplicarea asupra lui a metodei *.DataTable()*.

Toastr este una din multele soluții atunci când vorbim de notificări interactive. Este conceput să îmbrace elegant simplele notificări la care avem acces cu ajutorul browser-ului. Oferă patru opțiuni de notificări, ce pot fi alese în funcție de contextul mesajului urmat a fi afișat – succes, informație, avertisment, eroare.



Fig. 2.14 - Logo Leaflet

Leaflet este cea mai simplă și performantă librărie în ceea ce privesc hărțile și informațiile cu care poți lucra. Un competitor capabil pentru Google Maps, el dorește a fi, în primă instanță, ușor de înțeles și aplicat, având o structură de inițializare fixă.

```
var mymap = L.map("leaflet-map")
    .setView([44.4306476, 26.051922699999977], 14);
```

Fig. 2.15 - Exemplu de inițializare a hărții

Selectarea elementului de tip div se face pe baza id-ului său cu ajutorul metodei *map* după care metoda *setView* inițializează punctul central dat de cele două coordonate și cât de aproape, pe scala înălțimii, este privită harta.

De menționat sunt și *jQuery.cookie* și *jQuery.session*, librării ce permit un mai bun management al acestor două spații de stocare ale browser-ului.

3 Framework-ul Laravel



Fig. 3.1 - Logo Laravel

3.1 Introducere

Laravel refolosește și assemblează componente existente pentru a oferi un sistem coerent cu care se pot contrui aplicații web mult mai pragmatic și structurat. Inspirat din framework-uri populare scrise nu numai în PHP, precum Yii, ASP.NET MVC, Ruby on Rails, încorporează multe din lucrurile ce face un framework o unealtă ideală atunci când începem un nou proiect.

Nu se abate nici el de la paradigma MVC și reușește să fie un veteran în rândul framework-urilor ce regândesc ideea manipulării cererilor și răspunsurilor într-un format independent de baza lor, opus trecutului în care dezvoltatorii construiau metode adaptate la contextul aplicațiilor lor.

Este un framework cu foarte multe rădăcini în paleta Symfony, dar nu numai, cuprinzând varii librării precum SwiftMailer pentru o ușurință sporită în lucrul cu poșta electronică, Carbon pentru o mai bună prelucrare a datelor calendaristice și a timpului și multe altele cu scopuri clare, ca de exemplu autentificare sau raportare de erori.^[Vezi 3]

3.2 De ce?

Este un framework capabil să furnizeze funcționalități precum:

- Numiri: Ca și celelalte limbaje, precum Java și C#, acesta rezolvă problema coliziunilor prin numiri intuitive ale fișierelor de lucru și a celor incluse în ele.
O numire, de exemplu *namespace Illuminate\Database\Eloquent* marchează, în partea superioară a fișierului, calea către acesta, fiind urmat de importul librăriilor ce se realizează ca în exemplul *use Illuminate\Database\Eloquent\Model*.
- Funcții anonime: Sunt asemănătoare cu cele din JavaScript menite să aibe o construcție scurtă și să folosească tratării unor evenimente, filtre sau accesări de rute.
- Supraîncărcare: Permite metode precum *whereUsernameOrEmail(\$name, \$email)* ce nu sunt explicit definite în clasă, de ele fiind responsabilă metoda *__call()* ce le descompune în metodele componente, în cazul nostru *where('username', \$username)* și *orWhere('email', \$email)*.

3.3 Structura fișierelor

Framework-ul vine la pachet cu o modularizare, ce ajută la mentenanța pe termen lung a aplicației.

Este construit cu peste 20 de librării diferite de a căror integrare se ocupă Composer. Acesta este un manager al dependențelor ce are putere în întreaga aplicație și asigură o foarte ușoară actualizare a modulelor folosite.

Pe lângă librăriile pe care le manageriază Composer și care sunt stocate în directorul *vendor*, Laravel impune o anumită zonă standard și pentru fișiere de alt tip:

- În directorul *app* se regăsesc subdirectoarele *Controllers* și *Models* în care se află fișierele corespunzătoare celor două componente ale MVC.
- Subdirectoarele accesate frecvent din directorul *resources* sunt *views*, ce conține fișierele necesare componentei a treia din trioul MVC și *lang*, ce conține fișiere folosite pentru internaționalizarea aplicației.
- În directorul *public* se găsesc elementele dorite a fi accesate în componenta View precum imagini, fișiere de stilizare CSS și fișiere generatoare de comportament în pagină JavaScript.
- Directorul *routes* cuprinde fișierele de cofigurare a căilor de acces prin aplicație prin intermediul browser-ului, iar directoarele *config* și *database* setează o parte din mediul de lucru al aplicației.

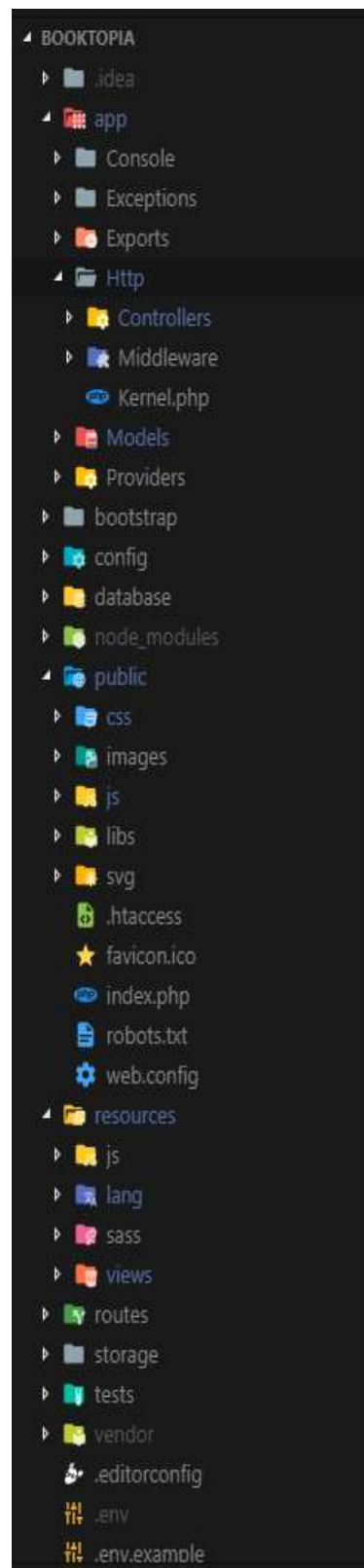


Fig. 3.2 - Structura fișierelor

3.4 Configurarea mediului de lucru

De mare ajutor în configurarea mediului în care aplicația rulează este fișierul `.env`. Acesta permite dezvoltatorilor să creeze diferite scenarii precum dezvoltare, producție, testare, scenarii în care se pot defini diferit baza de date folosită, sistemul de email și multe altele.

Are un format cheie-valoare, fiecare rând fiind interpretat ca o inițializare a unei variabile globale, vizibilă în orice loc din aplicație.

```
.env.example x
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=homestead
13 DB_USERNAME=homestead
14 DB_PASSWORD=secret
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_DRIVER=smtp
27 MAIL_HOST=smtp.mailtrap.io
28 MAIL_PORT=2525
29 MAIL_USERNAME=null
30 MAIL_PASSWORD=null
31 MAIL_ENCRYPTION=null
```

Fig. 3.3 - Fișier de configurare

3.5 Email

Trimiterea de email-uri este redusă la ceva banal, chiar și în situații în care mesajul este mare și poate conține atașamente, datorită clasei Mail, ce înfășoară librăria SwiftMailer.

```
Mail::send('helpers.email_format', ['text' => $text], function ($message) use ($subject, $email) {
    $message->subject($subject)
        ->from($email)
        ->to('booktopia.contact@gmail.com', 'Booktopia');
});
```

Fig. 3.4 - Trimitere de email

3.6 Autentificare

Fiind un lucru esențial în cadrul aplicațiilor web, Laravel vine cu o implementare predefinită pentru înregistrare, accesare a contului și chiar a posibilității de resetare a parolei.

3.7 Testabilitate

Pe parte de testare, framework-ul este construit să ofere lucruri ajutătoare menite să îți faciliteze accesul la rutere testate, colecționarea rezultatelor HTML, siguranța că metodele din diferite clase sunt apelate corespunzător și chiar posibilitatea de a juca rolul unui utilizator și de a verifica că aplicația răspunde corect, bazat pe atributele lui.

3.8 Routing

Laravel acordă flexibilitate dezvoltatorilor în ceea ce privește definirea rutelor. Acestea ajută, de asemenea, la o mai bună modularizare a întregii aplicații stabilind căile de acces și modul prin care un utilizator poate ajunge la o anumită pagină web.

Este inspirat din framework-uri mai mici ale altor limbaje, precum Sinatra (Ruby) sau Silex (PHP) și este elementul ce face, în primă fază, legătura între ce dorește utilizatorul și ce i se oferă.

```
Auth::routes();
Route::get('/notAdmin', function () {
    return view('helpers.NotAdmin');
});
Route::get('/home', function () {
    return redirect()->route('shop');
});

// Shop
Route::get('/', 'ShopController@index')->name('shop');
Route::post('shop/filters', 'ShopController@filters')->name('shop.filters');
Route::get('shop/filters', 'ShopController@filters')->name('shop.filters');
Route::get('shopping-cart', 'ShopController@shopping_cart')->name('shopping-cart');

Route::get('control-panel', 'ShopController@control_panel')->name('control-panel');
Route::post('contact-email', 'ShopController@contactemail')->name('contact-email');

// Users
Route::get('users/index', 'UserController@index')->name('users.index');
Route::post('users/get-user-role', 'UserController@get_user_role')->name('users.get-user-role');
Route::post('users/update-role', 'UserController@update_role')->name('users.update-role');

// Books
Route::post('books/drag-and-drop-upload', 'BookController@drag_and_drop_upload')->name('books.drag-ar
Route::get('books/import-from-CSV', 'BookController@import_from_CSV')->name('books.import-from-CSV');
Route::resource('books', 'BookController');
Route::post('books/rate', 'BookController@rate')->name('books.rate');
```

Fig. 3.5 - Rute de acces

O rută este formată din mai multe elemente fiecare cu rolul specific:^[Vezi RO 1]

- Verb: Acesta poate lua unul din cele 4 cunoscute GET, POST, PUT, DELETE sau resource, acesta din urmă creând rute pentru toate metodele controller-ului de tip REST specificat.
- URL: Este calea pe care utilizatorul trebuie să o acceseze prin intermediul browser-ului.
- Controller: Marchează clasa de tip controller în care se va cauta metoda.
- Acțiunea: Este metodă a controller-ului specificat anterior și reprezintă comportamentul pe care îl va avea server-ul.
- Funcția anonimă: Apare atunci când comportamentul nu este definit printr-o acțiune din controller.

3.9 Sistemul de șabloane Blade

Inspirat din Razor, sistemul de șabloane al framework-ului ASP.NET MVC, Laravel pune la dispoziția programatorilor Blade, un sistem similar cu care se pot crea diferite formate ierarhice ce conțin la rândul lor diferite blocuri inserate dinamic.

Este o soluție elegantă și eficientă prin care se evită codul duplicat și prin care se pot îmbina alternativ, în funcție de context, anumite elemente pentru a crea partea vizuală necesară utilizatorului.

Printre cele mai utilizate directive se numără: [Vezi RO 1]

- `@yield(" nume_secțiune ")`: Marchează o zonă în fișierul interpretat a fi o interfață în care va putea fi inclusă o secvență de cod dintr-un fișier diferit ce extinde interfața.
- `@extends(" cale_fișier ")`: Marchează numele interfeței pe care o extinde și pe care o poate suplimenta cu un conținut nou sau nu. De regulă fișierele tip interfață au simbolul ‘_’ în fața numelui și se găsesc în subfolderul *layouts* din *resources/views*.
- `@section(" nume_secțiune ")`: Marchează zona din interfața extinsă în care va ajunge codul scris în secțiune.
- `@include(" cale_fișier ")`: Marchează includerea codului existent în fișierul specificat în zona în care directiva a fost declarată.

Tot sistemul de șablonare ne permite să interacționăm direct prin cod PHP asupra conținutului paginilor HTML în momentul în care acestea vor fi randate de către server.

El asigură interpretarea diferitelor instrucțiuni și completarea pe baza lor a conținutului în pagina web. Se pot folosi structuri precum `if`, `for`, `foreach` ce ajută la o reducere a dimensiunilor fișierelor de cod și o mai bună înțelegere a imaginii de ansamblu.

Cea mai utilă componentă rămâne însă capacitatea de a afișa datele transmise de către controller și nu numai. Acest lucru se realizează într-o manieră simplă, cuprinzând secvența de cod sau variabila dorită a fi afișată între două seturi de acolade ca în exemplu.

```
Hello, {{ $name }}.
```

Fig. 3.6 - Text escapat

Siguranța este asigurată de framework, astfel că structura menționată mai devreme trece prin escaparea conținutului. Totuși dacă dorim anularea acelui efect putem folosi o structură similară.

```
Hello, {!! $name !!}.
```

Fig. 3.7 - Text neescapat

De asemenea, în componența blocurilor și a paginilor HTML se pot internaționaliza anumite secvențe. Acest lucru este realizat cu ajutorul fișierelor regăsite în subdirectorul *lang* din directorul *resources*. Acestea sunt redactate sub forma unor vectori asociativi și curpind vaste colecții de date cu traducerea aferentă lor. Se pot ierarhiza, fapt ce ajută la o mai bună înțelegere a contextului în care ele acționează, iar referențierea către ele se face sub forma *lang*("nume_fișier.cheie1.cheieN"). Limba în care conținutul aplicației este afișat este dată de proprietatea *locale* aflată în fișierul de configurare *app.php*.

```
@extends('layouts.app')

@section('scripts')
    <script type="text/javascript"
        src="{!! asset('js/general/control-panel.js') !!}" defer></script>
@endsection

@section('styles')
    <link rel="stylesheet" type="text/css"
        href="{!! asset('css/general/control-panel.css') !!}">
@endsection

@section('content')
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-12">
                <div class="card">
                    <div class="card-body">
                        @if(Auth::user()->isAdmin() || Auth::user()->isPartner())
                            <a class="card panel" href="{!! route('shop') !!}">
                                <i class="fas fa-3x fa-shopping-cart"></i><br>
                                @lang('dictionary.control-panel.shop')
                            </a>
                        @endif
                    </div>
                </div>
            </div>
        </div>
    </div>
```

Fig. 3.8 - Utilitatea Blade în HTML

În exemplu, se vede cum pagina curentă extinde interfața *app* din subfolderul *layouts*, curpind două secțiuni pentru cele două tipuri de fișiere adiționale celor HTML, a căror conținut va fi poziționat în zonele numite *scripts*, respectiv *styles* și secțiunea de bază numită *content* a cărei conținut definește pagina.

În aceasta se găsește structura if, condiționată de rolul pe care îl joacă utilizatorul actual, de a cărei decizie depinde randarea în pagină a ancorei. De asemenea, numele afișat în cadrul ancorei este preluat din fișierul *dictionary*, de la tuplul *control-panel*, cheia finală fiind *shop*.

3.10 Sistemul de construcție a bazei de date

Acesta ne dă posibilitatea de a construi schema bazei de date folosind cod PHP și de a păstra o evidență a diferitelor modificări ce apar prin intermediul migrărilor. Acestea reprezintă o metodă ușoară și reversibilă de a descrie un eveniment și impactul lui asupra schemei bazei de date.

3.11 Construcția query-urilor & Eloquent ORM

Laravel deține și un limbaj intern de relaționare cu baza de date, care scutește de mult efort în compunerea de query-uri lungi și complicate urmate a fi prelucrate prin sintaxă PHP. El se reduce la o înlănțuire de metode ce alcătuiesc într-un final query-ul analizat de baza de date.

Query-urile realizate cu ajutorul Laravel folosesc legarea parametrilor de tip PDO, astfel un dezvoltator este scutit de “curățarea” parametrilor trimiși acest lucru fiind realizat de framework ce se ocupă de înlăturarea oricărui risc de atac de tip SQL Injection.

Deși Laravel ne oferă posibilitatea de a crea query-uri sigure, acestea pot deveni extrem de mari și pe alocuri nefolositoare, de aceea în cazul unor aplicații mari ce lucrează cu multe tabele se utilizează sistemul Eloquent ORM.

Acesta aduce o implementare simplă și elegantă a unui obiect activ ce interacționează cu baza de date. Astfel fiecărui tabel îi corespunde un model, model pe baza căruia se vor construi query-urile și legăturile între elementele bazei de date.

Un model este reprezentat ca o clasă ce extinde *Illuminate\Database\Eloquent\Model* și având anumite proprietăți specifice.

- Convenția de nume a modelului este CamelCase în schimb ce numele tabelului este snake_case. Pentru un nume al tabelului diferit sistemul trebuie înștiințat prin setarea atributului *table*.
- În ceea ce privește numele coloanei de cheie primară, Laravel se așteaptă ca aceasta să fie numită *id*, în caz contrar, în model se va seta proprietatea *primaryKey*.
- Framework-ul caută în fiecare tabel două coloane *created_at* și *updated_at*, pe care să le gestioneze, comportament ce poate fi stopat prin setarea atributului *timestamps* fals.

Pentru o transpunere relațiilor de tip foreign key din baza de date, în Laravel se pot configura de asemenea relații între modele.

Ele pot apărea în trei situații:

➤ One To One

În modelul de bază ce conține foreign key-ul se implementează o metodă cu nume sugestiv care conține *hasOne(model2)*.

În modelul al doilea, se face legătura inversă cu o metodă ce conține *belongsTo(model1)*.

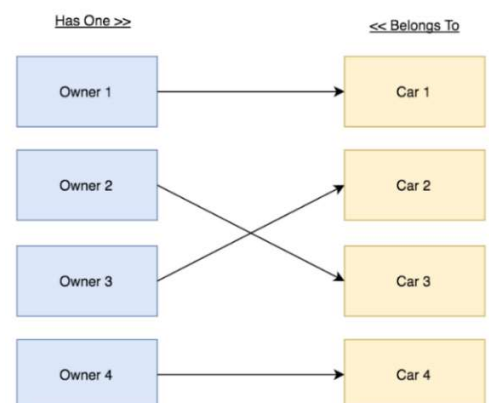


Fig. 3.9 - Relația între modele One to One

➤ One To Many

În modelul de bază ce conține foreign key-ul se implementează o metodă cu nume sugestiv care conține *hasMany(model2)*.

În modelul al doilea, se face legătura inversă cu o metodă ce conține *belongsTo(model1)*.

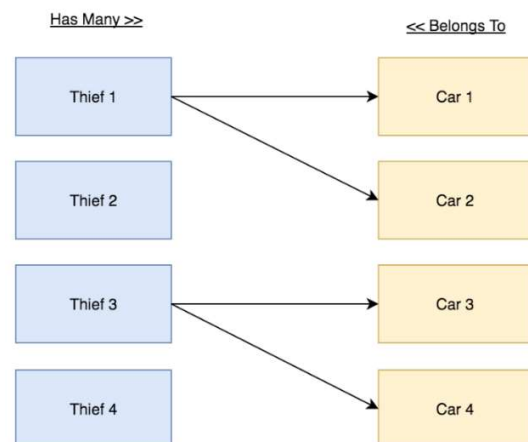


Fig. 3.10 - Relația între modele One to Many

➤ Many To Many

În ambele modele se creează câte o metodă care va conține *belongsToMany(modelOpus, tabelulPivot)*.

Al doilea parametru reprezintă acel tabel asociativ necesar în relațiile many-to-many, responsabil de legarea cheilor din cele două tabele.

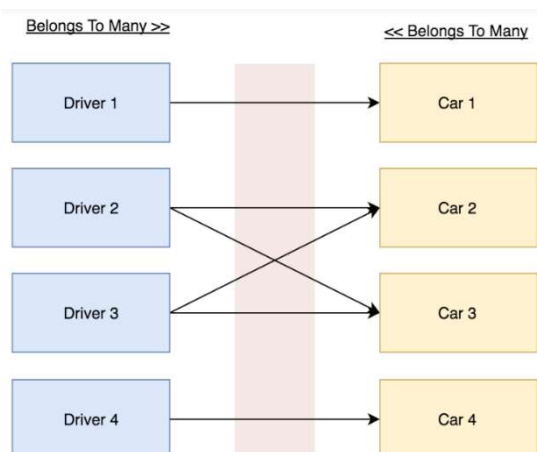


Fig. 3.11 - Relația între modele Many to Many

4 Proiectarea aplicatiei

4.1 Conceperea bazei de date

Aplicația lucrează pe mai multe ramuri și de aceea necesită o serie vastă de tabele corespunzător interconectate. Aproape fiecărui tabel îi corespunde un model, cu ajutorul căruia aplicația manageriază foarte ușor stocarea și preluarea valorilor necesare.

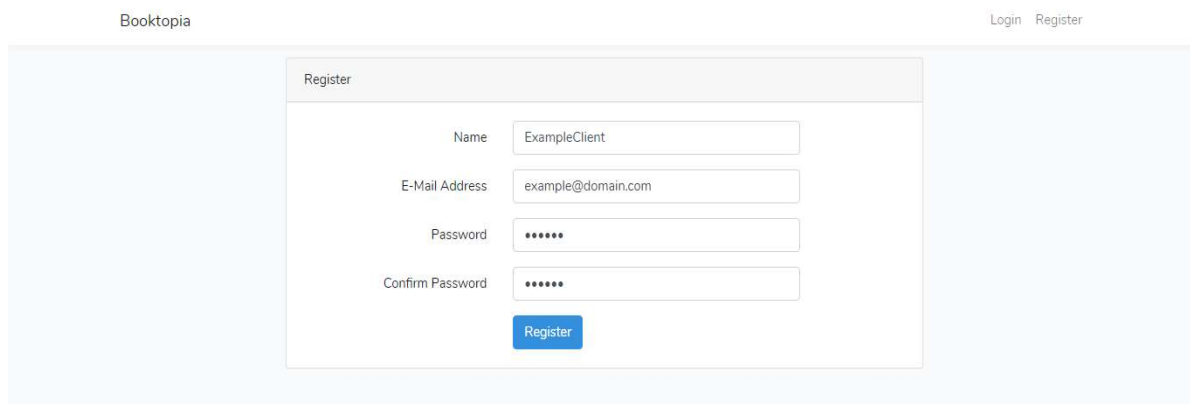
Cele două modele definitorii sunt *Book* și *User*. Pe lângă acestea se mai adaugă noi altele, în legătură directă sau nu, crând o oarecare imaginea de ansamblu a ce poate aplicația oferi.

Printre contextele de utilizare ale website-ului ce necesită un nou obiect central rezumat în model se numără crearea unui profil personal, acordarea părerilor asupra unei cărți, modificarea stocului produselor sau vizualizarea comenzilor efectuate.

4.2 Autentificarea

În ceea ce privește sistemul de autentificare, framework-ul permite generarea acestui sistem, atât din punct de vedere funcțional cât și din punct de vedere vizual, putând fi apoi modificat. De asemenea, validarea formularelor de acces (înregistrare sau accesare) este asigurată de framework. Generarea se realizează folosind comanda specifică *php artisan make:auth*.

Pentru înregistrare se apasă butonul *Register* aflat în partea dreaptă a bării de navigație, urmând ca apoi să fie completat formularul și trimis spre validare.



The screenshot shows a web application interface for a registration form. At the top left, the text "Booktopia" is visible. At the top right, there are links for "Login" and "Register". The main content area features a light blue background with a white rectangular box titled "Register". Inside this box, there are four input fields: "Name" with the value "ExampleClient", "E-Mail Address" with the value "example@domain.com", "Password" with masked characters "*****", and "Confirm Password" with masked characters "*****". Below these fields is a blue button labeled "Register".

Fig. 4.1 – Formularul de înregistrare

Dacă toate câmpurile respectă condițiile de validare, utilizatorul nou creat este totodată logat pe platformă.

Pentru accesare se va apăsa butonul *Login* aflat, de asemenea, în partea dreaptă a bării de navigație, și se vor folosi adresa de email și parola asociată contului.

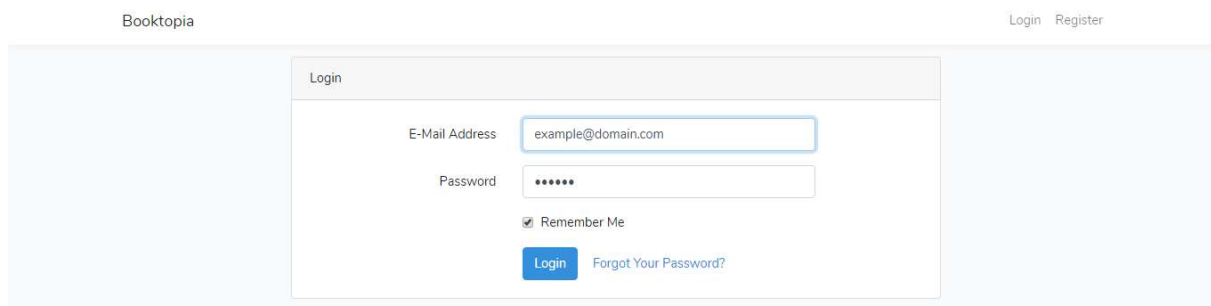
The image shows a web page for 'Booktopia'. In the top right corner, there are links for 'Login' and 'Register'. The main content area features a 'Login' form. The form has two input fields: 'E-Mail Address' with the value 'example@domain.com' and 'Password' with masked characters '*****'. Below these fields is a checkbox labeled 'Remember Me' which is checked. At the bottom of the form are two buttons: a blue 'Login' button and a link 'Forgot Your Password?'.

Fig. 4.2 – Formular de accesare a contului

4.3 Formatul aplicației

Design-ul aplicației este unul alcătuit din trei componente:

- Header-ul magazinului: Acesta cuprinde search bar-ul și butonul ce trimite la vizualizarea produselor din coș.

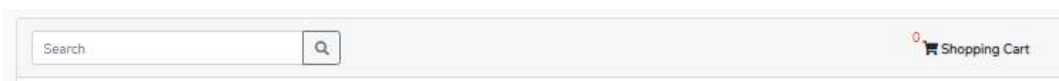
The image shows a horizontal navigation bar. On the left, there is a search bar with the placeholder text 'Search' and a magnifying glass icon. On the right, there is a shopping cart icon with a red circle containing the number '0' and the text 'Shopping Cart'.

Fig. 4.3 – Bara interioară de navigație

- Footer-ul magazinului: Acesta cuprinde programul de funcționare al magazinelor fizice, localizarea pe hartă, butonul de trimitere către pagina în care se poate determina cel mai apropiat magazine de locația în care se face căutare și formularul de contact de trimite pe email neregulile sesizate de clienți.

Fig. 4.4 – Panoul din josul paginii

- Conținutul: Acesta este elementul maleabil ce dă semnificație paginii curente.

4.4 Magazinul

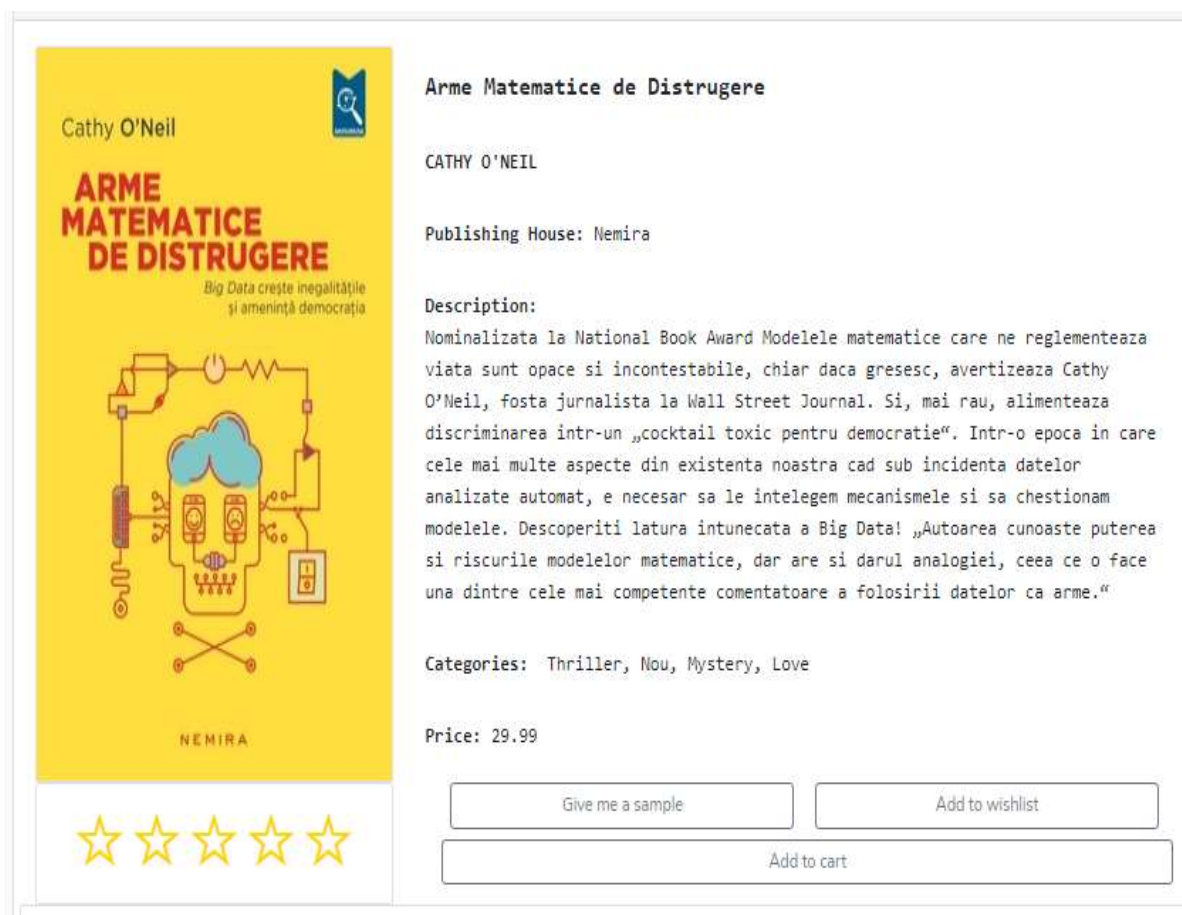
În primă instanță se afișează întreaga listă a cărților existente în magazin cu titlu, poză și preț eferente fiecăreia, acestea putând fi reduse prin selectarea filtrelor din panou și apăsarea butonului de aplicare a lor.

Fig. 4.5 – Afișarea cărților din magazin

4.5 Pagina de prezentare a unei cărți

Un utilizator poate vedea pagina de prezentare a oricărei cărți existente în magazin fără a fi nevoie de accesarea contului făcut pe platformă. În acest mod se pot citi proprietățile, se pot vedea cărți asemănătoare și se pot activa funcționalități precum adăugarea produsului în coșul de cumpărături prin apăsarea butonului *Add to Cart* sau obținerea unei mostre prin apăsarea butonului *Give me a sample*. Acesta din urmă creează un apel către API-ul Google Books ce va răspunde cu câteva pagini, suficiente pentru ca utilizatorul să își formeze o parere despre ce va cumpăra.

Adăugarea cărții pe lista de dorințe se poate face doar dacă utilizatorul s-a identificat prin contul deținut pe platformă. Se realizează prin apăsarea butonului *Add to wishlist* și va rezulta în inserarea în lista asociată profilului a cărții curente.



The screenshot displays a book detail page for "Arme Matematice de Distrugere" by Cathy O'Neil, published by Nemira. The page layout includes a book cover on the left, a title and author section at the top right, a detailed description in the middle right, category tags, the price, and three interactive buttons at the bottom right. The book cover features a yellow background with a circuit diagram of a skull and the text "Big Data crește inegalitățile și amenință democrația". The description discusses the impact of mathematical models on society and the dangers of Big Data. The price is listed as 29.99. The buttons are "Give me a sample", "Add to wishlist", and "Add to cart".

Arme Matematice de Distrugere

CATHY O'NEIL

Publishing House: Nemira

Description:

Nominalizata la National Book Award Modelele matematice care ne reglementeaza viata sunt opace si incontestabile, chiar daca gresesc, avertizeaza Cathy O'Neil, fosta jurnalista la Wall Street Journal. Si, mai rau, alimenteaza discriminarea intr-un „cocktail toxic pentru democratie”. Intr-o epoca in care cele mai multe aspecte din existenta noastra cad sub incidenta datelor analizate automat, e necesar sa le intelegem mecanismele si sa chestionam modelele. Descoperiti latura intunecata a Big Data! „Autoarea cunoaste puterea si riscurile modelelor matematice, dar are si darul analogiei, ceea ce o face una dintre cele mai competente comentatoare a folosirii datelor ca arme.”

Categories: Thriller, Nou, Mystery, Love

Price: 29.99

Give me a sample Add to wishlist

Add to cart

Fig. 4.6 – Formatul de afișare al detaliilor unei cărți

De asemenea, după identificare, un utilizator poate da notă și poate scrie comentarii ce vin în ajutorul viitorilor clienți. Tot aici, un comentariu poate fi apreciat sau dezagreat, compunând o părere colectivă a mai multor clienți. Editarea și ștergerea comentariilor se poate realiza doar de către un administrator sau de către autorul lui.

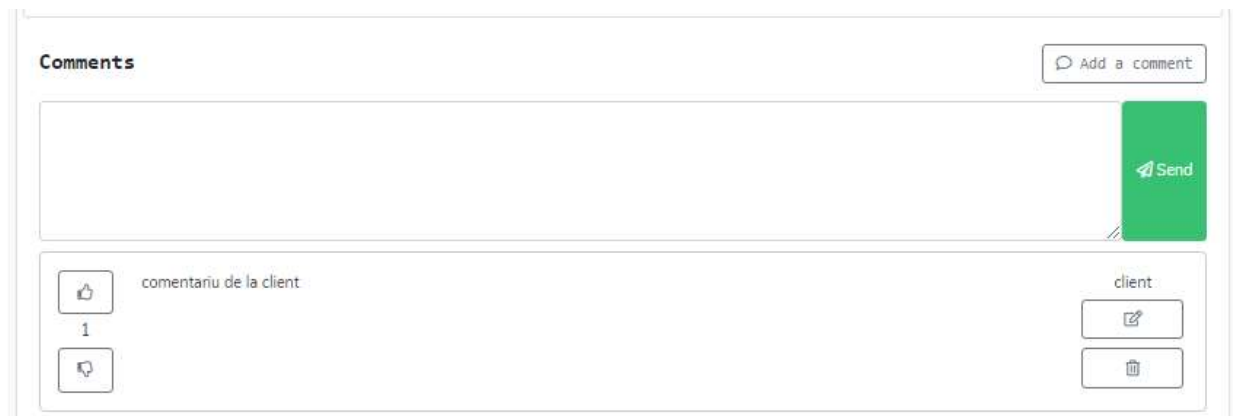


Fig. 4.7 – Panoul comentariilor

4.6 Coșul de cumpărături


Lista produselor curente aflate în coșul de cumpărături se poate vedea prin apăsarea butonului *Shop* aflat în bara interioară de navigație specifică interfeței magazinului.

În cadrul acestei pagini se poate modifica conținutul coșului prin adăugarea sau eliminarea produselor de același tip, prin apăsarea butoanelor specifice de incrementare, respectiv decrementare asociate fiecărei cărți din coș.

Dacă utilizatorul este un client fidel al magazinului înseamnă că are un număr considerabil de puncte în portofelul atașat profilului personal. Aceste puncte de fidelitate se pot folosi în cazul în care numărul lor depășește prețul total, fapt ce reduce costul pachetului la 0, acțiunea fiind ireversibilă. Nefolosirea lor duce la plata prețului afișat al coletului și adăugarea în portofel a 4% din costul total echivalat în număr de puncte.

Your shopping cart:

ID:	Title	Quantity	Price
118	Harry Potter and the Philosopher's Stone	<input type="button" value="-1"/> <input type="text" value="1"/> <input type="button" value="+1"/>	572.2
12	Arme Matematice de Distrugere	<input type="button" value="-1"/> <input type="text" value="2"/> <input type="button" value="+1"/>	59.98
117	The Hunger Games	<input type="button" value="-1"/> <input type="text" value="1"/> <input type="button" value="+1"/>	430.01

Points: 256 

Total: 1062.19 RON

Fig. 4.8 – Conținutul coșului de cumpărături

Plata impune apăsarea butonului *Checkout* urmată de completarea unui formular specific.

Billing address ▲

☐ Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium aliquam, dignissimos, harum incidunt molestiae natus nisi numquam perferendis. ☐

Delivery address ▲

☐ Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium aliquam, dignissimos, harum incidunt molestiae natus nisi numquam perferendis. ☐

Payment method ▲

☐ Ramburs ☐ Card (PayU)

Price ▲

Fig. 4.9 – Formular folosit pentru achiziționarea unei cărți

4.7 Profilul personal

Acesta poate fi accesat din meniul ce se deschide la apăsarea butonului cu numele utilizatorului aflat în colțul din dreapta sus al ferestrei.

Un profil predefinit se creează la prima încercare de accesare a lui, după care, pentru modificare se folosește butonul *Edit* ce deblochează câmpurile. Salvarea se face prin apăsarea butonului *Save* ce va înlocui în timpul editării butonul anterior.

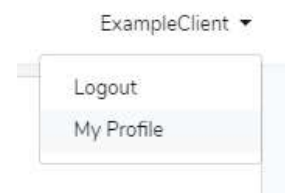


Fig. 4.10 – Meniu asociat numelui de utilizator

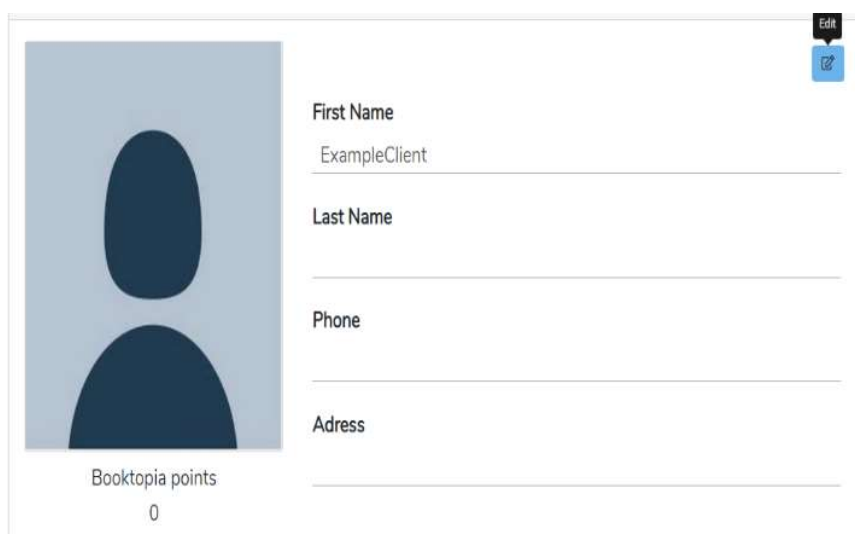


Fig. 4.11 - Pagina profilului personal când nu se pot modifica câmpurile sale

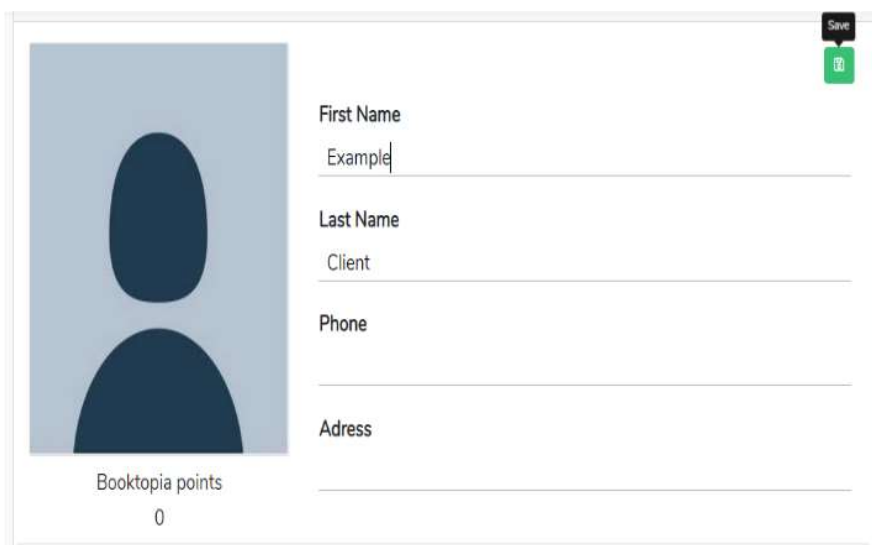


Fig. 4.12 – Pagina profilului personal când se pot modifica câmpurile sale

4.8 Panoul de control

Aplicația este gândită din trei perspective ierarhice corespunzătoare celor trei roluri posibile pe care un utilizator le poate lua. Acestea sunt client, partener și administrator, ultimele două roluri având drept de modificare parțială sau totală a conținutului magazinului. Tocmai de aceea ei au acces la panoul de control.

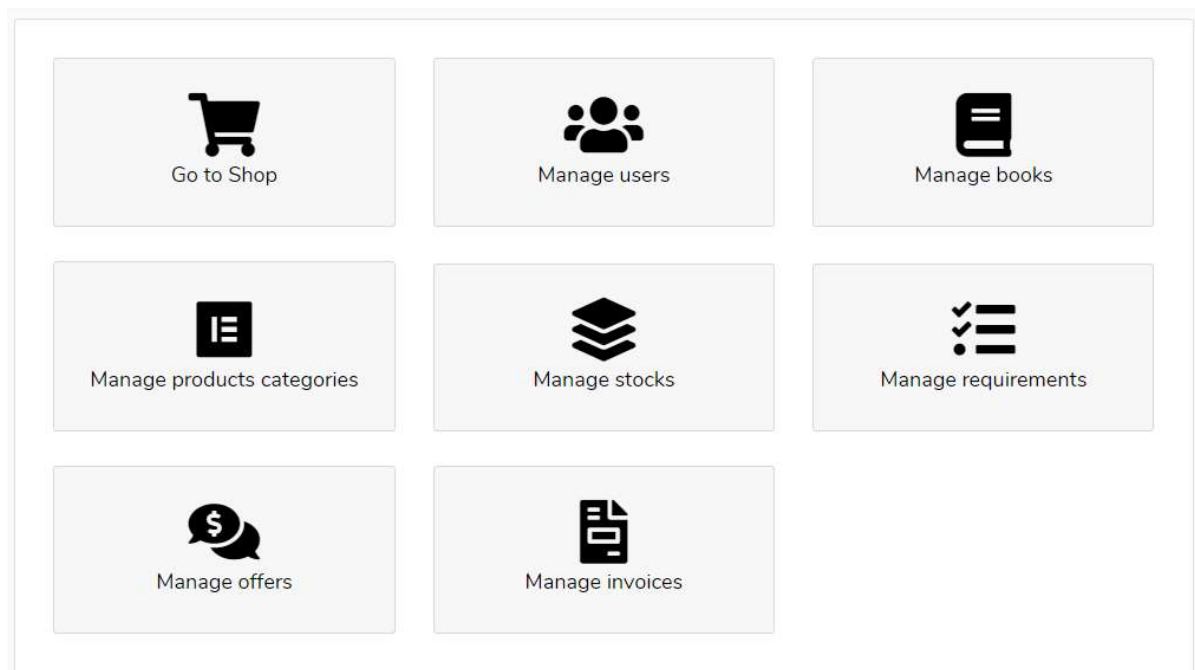


Fig. 4.13 – Panoul de control văzut din perspectiva unui administrator

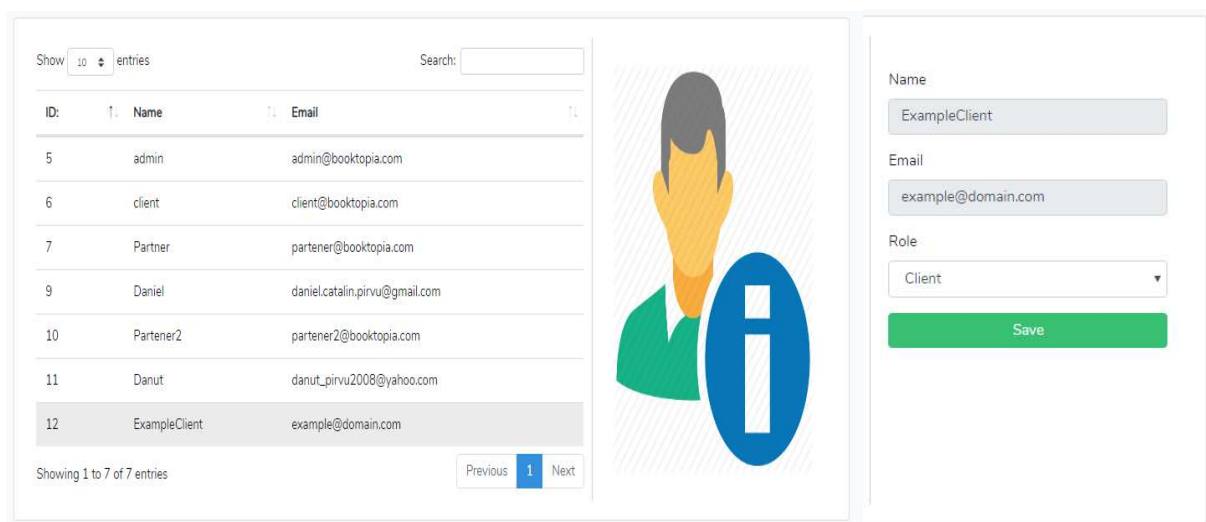
Aspectul este însă diferit, un partener putând accesa doar pagina de management al cărților, a stocului și a cererilor în timp ce un admin are acces la toate.

Pentru o mai bună siguranță, framework-ul dă posibilitatea creării unor porți – *middlewares* și setarea lor în cadrul rutelor de acces prin care trece cererea făcută către server. Astfel un client, ce nu are drepturi de acces asupra panoului de control sau a oricărei pagini la care trimite acesta, va fi oprit în situația în care acesta va vrea totuși să ajungă pe acestea și i se va afișa un mesaj corespunzător.

4.8.1 Gestionarea utilizatorilor

Panoul le oferă administratorilor posibilitatea de a vedea și modifica rolul utilizatorilor în aplicație.

La apăsarea pe linia corespunzătoare utilizatorului a cărui rol este dorit a fi modificat, se deschide panoul lateral în care sunt preluate datele și se poate seta un nou rol prin selectarea opțiunii dorite urmate de apăsarea butonului *Save*.



The screenshot displays a user management interface. On the left, a table lists users with columns for ID, Name, and Email. The user 'ExampleClient' (ID 12) is selected. To the right of the table is a large blue circular icon with a white lowercase 'i'. On the far right, a side panel shows the details for 'ExampleClient', including fields for Name, Email, and a Role dropdown menu set to 'Client'. A green 'Save' button is at the bottom of the side panel.

ID	Name	Email
5	admin	admin@booktopia.com
6	client	client@booktopia.com
7	Partner	partener@booktopia.com
9	Daniel	daniel.catalin.pirvu@gmail.com
10	Partener2	partener2@booktopia.com
11	Danut	danut_pirvu2008@yahoo.com
12	ExampleClient	example@domain.com

Showing 1 to 7 of 7 entries

Previous 1 Next

Name: ExampleClient

Email: example@domain.com

Role: Client

Save

Fig. 4.14 – Panoul de acces la rolurile utilizatorilor

Popularea cu date despre utilizator a panoului lateral și salvarea noului rol se face prin intermediul apelurilor de tip AJAX, astfel se asigură o performanță sporită în ceea ce privește viteza de răspuns și, în plus, elimină necesitatea, după oricare din cele două acțiuni, a reîmprospătării pagini.

4.8.2 Gestionarea cărților

Este panoul care are unul dintre cele mai importante roluri de ordin administrativ din întreaga aplicație.

Aici putem vorbi de două perspective, cea a partenerului și cea a administratorului.

Din perspectiva administratorului se pot vizualiza toate cărțile existente în baza de date, fie ele vizibile de către un client sau nu și se fac trimiteri către operațiile de tip CRUD efectuate asupra modelului Book.

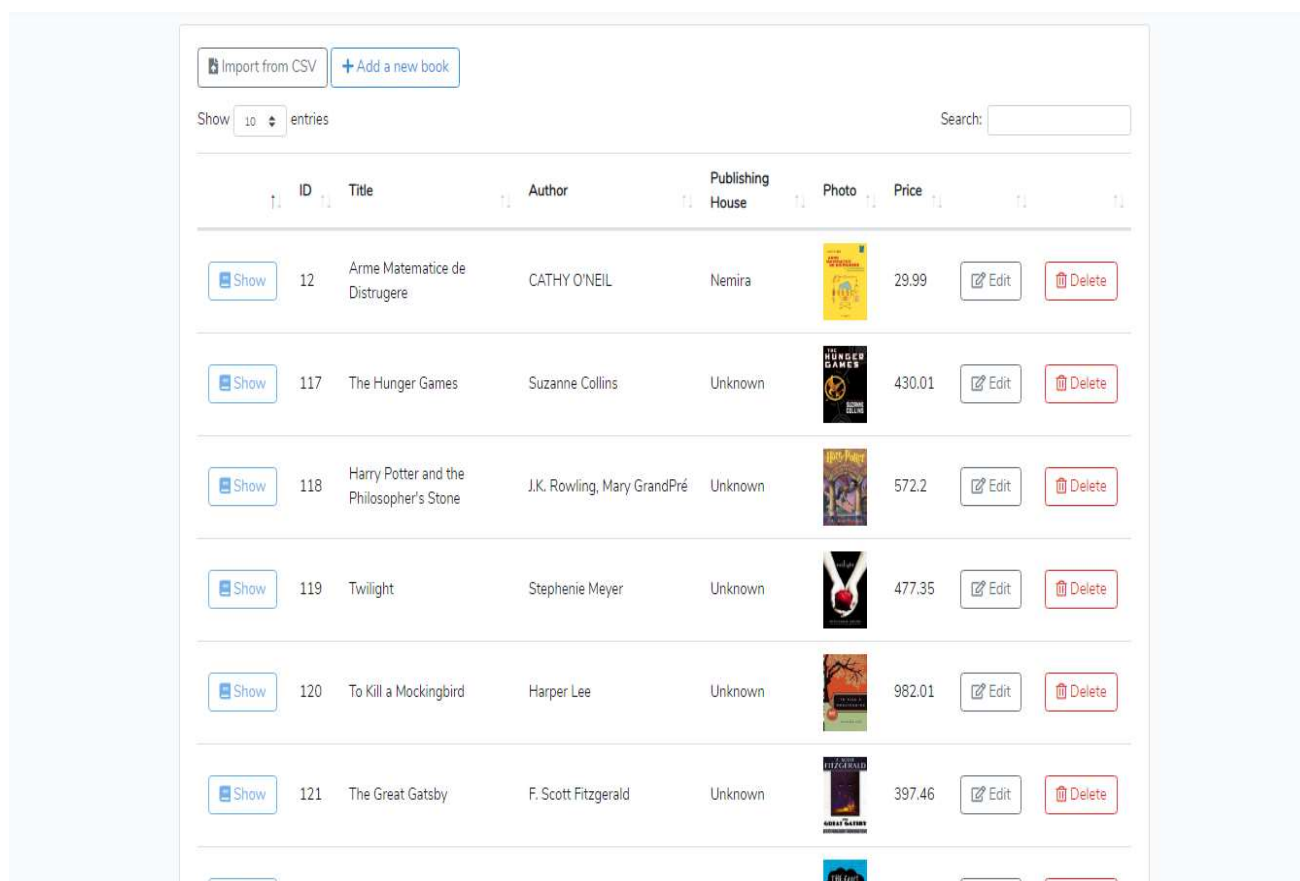


Fig. 4.15 – Panoul de control al cărților

Crearea unei noi cărți impune apăsarea butonul *Add a new book*, ce va trimite la pagina formularului în care vom putea seta toate proprietățile caracteristice, grupate pe patru mari zone.

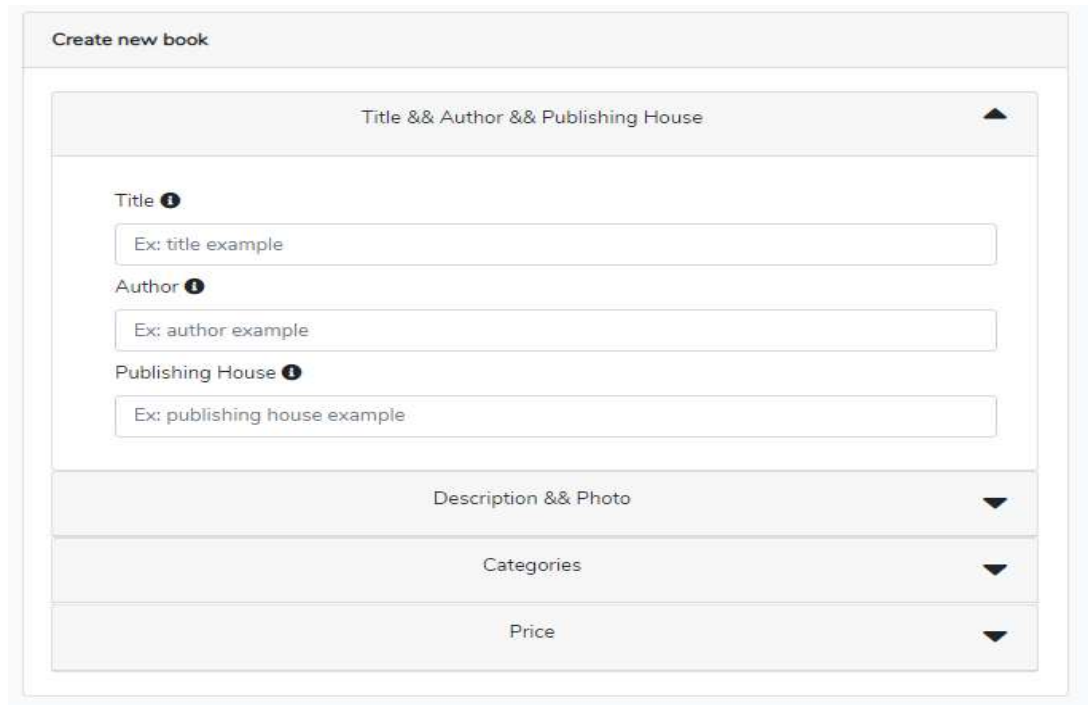


Fig. 4.16 – Formularul de creare a unei noi cărți – Secțiunea I

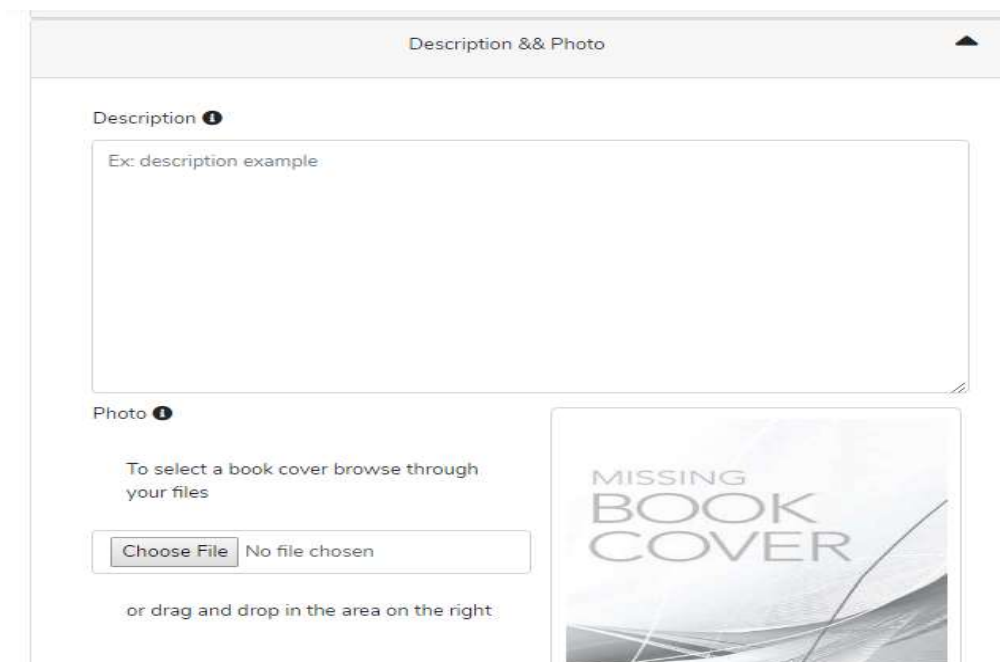


Fig. 4.17 - Formularul de creare a unei noi cărți – Secțiunea II

The form is divided into four main sections, each with a header bar and a content area:

- Title & Author & Publishing House:** A header bar with a downward arrow icon.
- Description & Photo:** A header bar with a downward arrow icon.
- Categories:** A header bar with an upward arrow icon. Below it, there are five buttons: "+ Thriller", "+ Nou", "+ Mystery", "+ Love", and "+ Sci Fi".
- Price:** A header bar with an upward arrow icon. Below it, there is a label "Price" with an information icon, a text input field containing "Ex: 99.99 Ron", and a green "Submit" button.

Fig. 4.18 - Formularul de creare a unei noi cărți – Secțiunile III și IV

Modificarea proprietăților unei cărți existente implică apăsarea butonului *Edit* de pe linia asociată, acțiune ce va rezulta în deschiderea formularului precedent menționat dar care va avea câmpurile precompletate, urmând ca după efectuarea schimbărilor să se valideze și să se salveze noua stare în baza de date.

Ștergerea se face prin apăsarea butonului *Delete* de pe linia asociată urmată de confirmare.

Adminul are puterea de a crea direct cărți ce vor fi vizibile în magazin de către clienți, în timp ce un partener creează o cerere ce va trebui acceptată de către un admin. În ceea ce privește modificarea și ștergerea unei cărți, aceste două acțiuni pot fi desfășurate în întregime doar de către un administrator. Astfel rolul unui partener nu este altul decât de a eficientiza munca unui administrator.

4.8.3 Gestionarea categoriilor

În această pagină un administrator poate vedea întreaga listă de categorii și prin apăsarea oricăreia se va deschide panoul lateral cu datele relevante. Acțiunile posibile sunt schimbarea numelui categoriei, acest lucru realizându-se într-o manieră similară cu modificarea datelor din profilul personal, ștergerea sau eliminarea anumitor cărți ce aparțin categoriei curente.

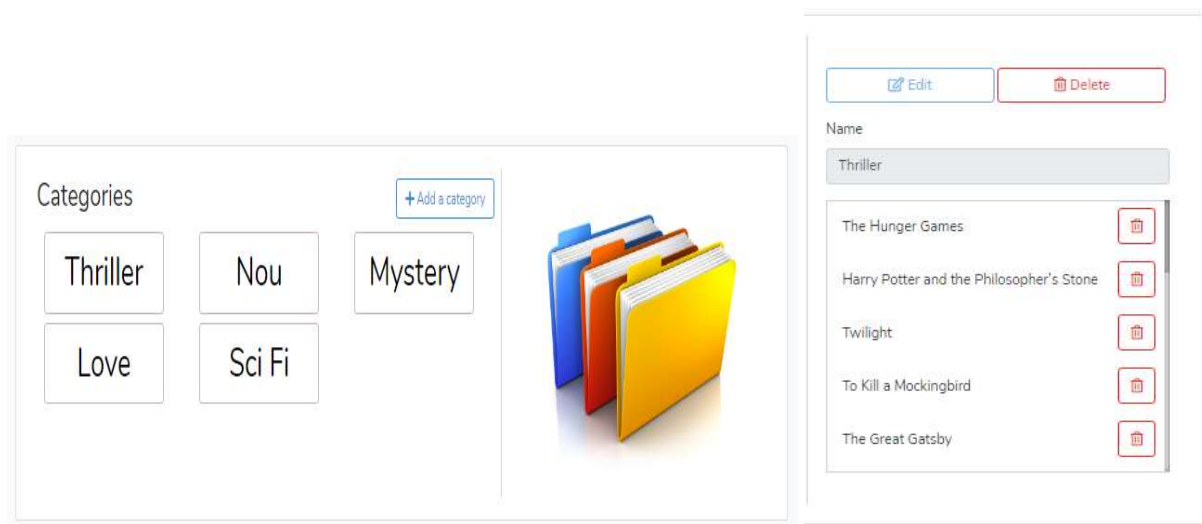


Fig. 4.19 – Panoul de control asupra categoriilor și cărților ce o dețin

4.8.4 Gestionarea stocul produselor

Un administrator ce accesează pagina stocurilor produselor vede toate cărțile existente în baza de date și prin apăsarea pe o linie a tabelului se va deschide panoul adițional în care se vor popula câmpurile cu informațiile asociate stocului curent.

Pentru a modifica stocului cărții curente se va apăsa butonul *Edit*, după care în câmpul, ce acum poate fi populat, se va introduce un număr întreg ce va urma a fi adăugat la valoarea curentă a stocului. Procesul se încheie prin apăsarea butonului *Save*.

Un administrator are dreptul de a salva în mod direct modificări efectuate, în timp ce un partener, ce poate vedea doar cărțile create de acesta și care va urma aceeași pași, va crea o nouă cerere ce va fi aprobată sau refuzată de către un administrator.

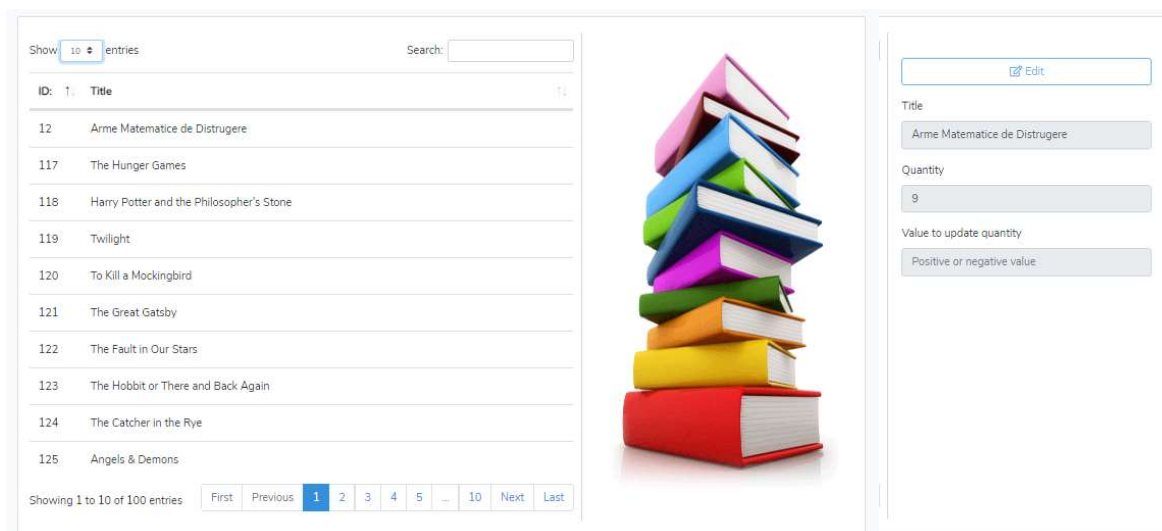


Fig. 4.20 – Panoul de control al stocului produselor

4.8.5 Gestionarea cererilor partenerilor

Acest panoul este, de asemenea, diferențiat între cele două roluri. Un administrator vede toate cererile făcute de toți partenerii, pe când un partener le vede doar pe cele ale sale.

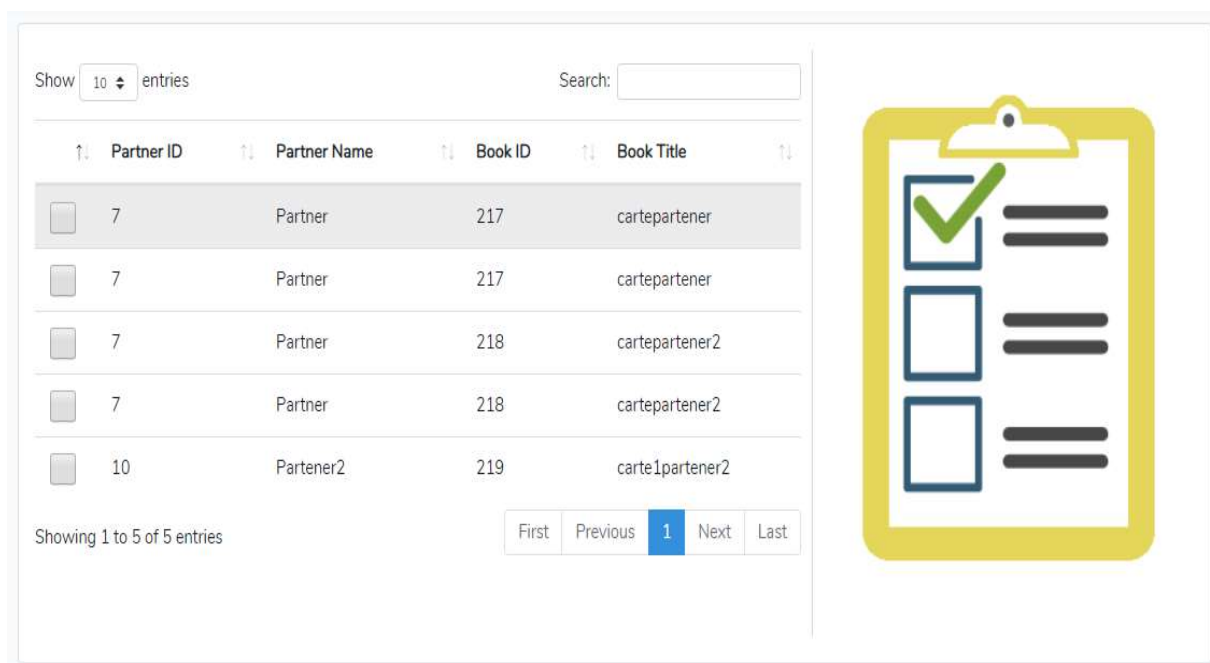
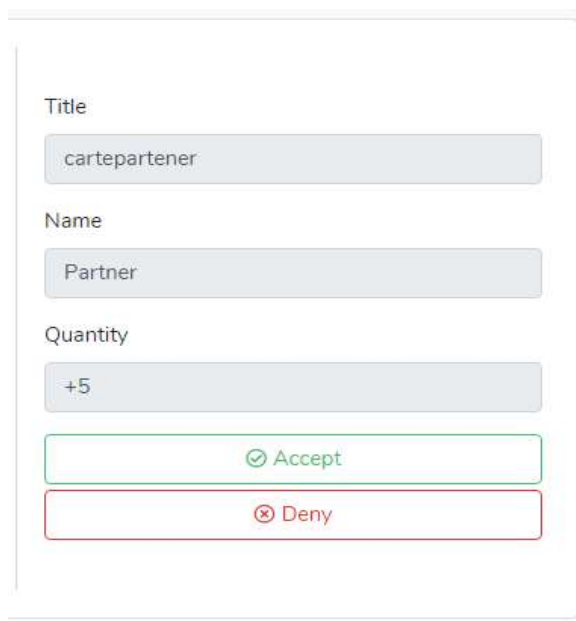


Fig. 4.21 – Panoul de control al cererilor partenerilor

Fiecare linie poate activa panoul lateral în care se pot desfășura activități specifice.

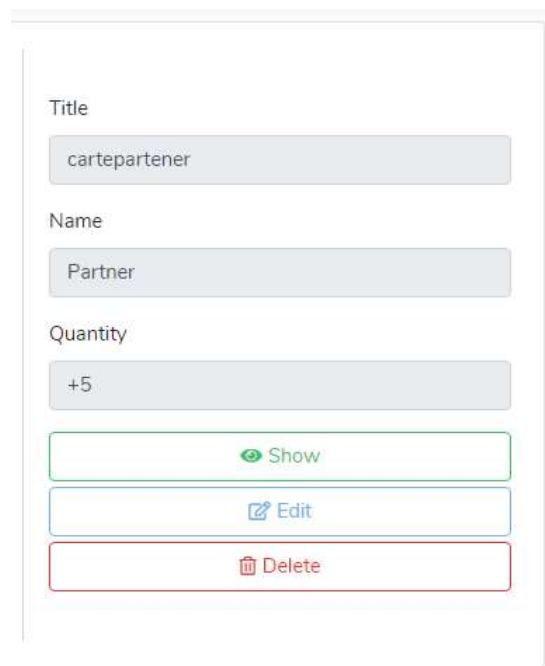
Un administrator poate doar accepta sau refuza o cerere, în timp ce un partener își poate vedea, modifica sau șterge propria cerere.

Pentru vizualizare și modificare, se vor prelua cererile selectate, urmând apoi ca un partener să fie redirectionat către pagina stocurilor, ce va avea, de această dată, un conținut personalizat. Cererile ce au un statut diferit de neutru nu mai pot fi întrebuințate celor trei acțiuni.



This screenshot shows a web form for managing a request. It contains three input fields: 'Title' with the value 'cartepartner', 'Name' with the value 'Partner', and 'Quantity' with the value '+5'. Below the fields are two buttons: a green 'Accept' button with a checkmark icon and a red 'Deny' button with a crossed-out circle icon.

Fig. 4.22 – Panoul lateral de control asupra stării cererii din perspectiva unui administrator

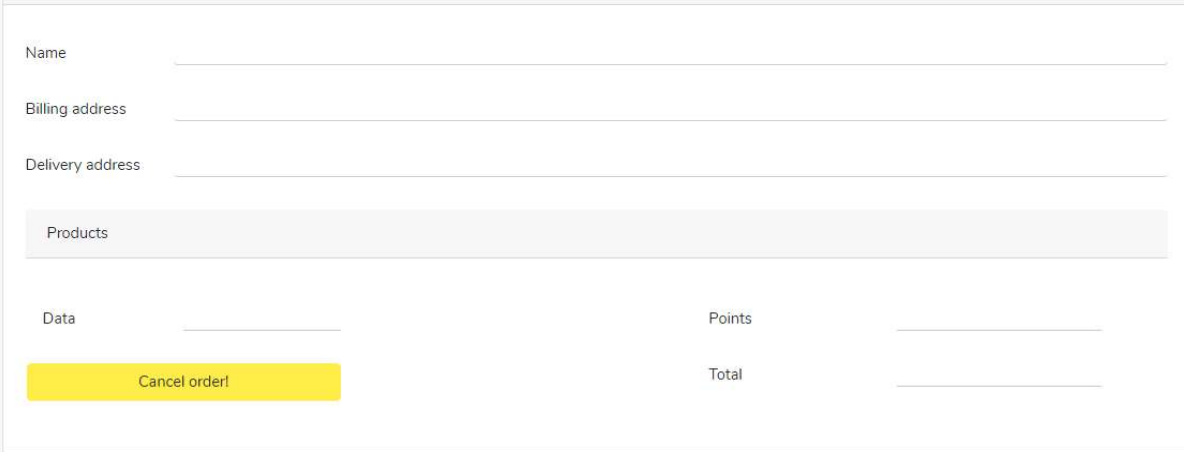


This screenshot shows a similar web form for managing a request, but with different action buttons. It contains the same three input fields: 'Title' with 'cartepartner', 'Name' with 'Partner', and 'Quantity' with '+5'. Below the fields are four buttons: a green 'Show' button with an eye icon, a blue 'Edit' button with a pencil icon, and a red 'Delete' button with a trash can icon.

Fig. 4.23 – Panoul lateral de control asupra stării cererii din perspectiva unui partener

4.8.6 Vizualizarea comenzilor efectuate

Istoricul comenzilor poate fi văzut în ultima parte a profilului personal, acesta fiind compus din mai multe panouri ce descriu sumar comanda. Pentru vizualizarea detaliată a unei comenzi se va apăsa pe zona panoului corespunzător ei, acțiune ce va rezulta în deschiderea unei ferestre pentru afișarea tuturor informațiilor.



The image shows a web form for displaying invoice details. It contains several input fields and a table structure.

Fields:

- Name:
- Billing address:
- Delivery address:
- Products:
- Data:
- Points:
- Total:

Buttons:

- Cancel order! (Yellow button)

Products	
Data	Points
Cancel order!	Total

Fig. 4.24 – Formularul de afișare a detaliilor unei facturi

5 Concluzii

Pot confirma că tehnologiile folosite sunt potrivite pentru dezvoltări rapide și o mentenanță pe termen lung a fiecărei componente existente în aplicație.

Folosind tehnologiile de front-end povestite am putut proiecta rapid imaginea de ansamblu a aplicației, după care back-end-ul a venit natural creând funcționalitate elementelor statice.

Framework-ul a ajutat în fiecare moment, în procesul dezvoltării aplicației pe toate ramurile sale, punând la dispoziție o serie de unelte. De asemenea, a permis realizarea unei aplicații complete din multe puncte de vedere, ce țin atât de contextul principal al aplicației cât și de cel secundar.

Pentru o imagine de viitor a aplicației propun implementarea mai multor elemente:

- Integrarea unui sistem inteligent de plată online
- Posibilitatea găsirii celui mai apropiat magazin raportat la locația de unde se face căutarea
- Integrarea autentificării cu rețelele sociale
- Crearea unui sistem ce folosește inteligență artificială pentru a sugera noi cărți
- Crearea unui sistem de reclame periodice prin email

6 Bibliografie

1. Mark Richards - *Software Architecture Patterns*
2. Rasmus Lerdorf, Kevin Tatroe, Bob Kaehms, Ric McGreedy - *Programming PHP*
3. Martin Bean - *Laravel 5 Essentials*
4. Maria A.P., Syed F.R., Ahmed B., Craig W., Rhiana H.- *Learn Bootstrap: The Collection*
5. David Flangan - *JavaScript: The Definitive Guide*
6. jQuery Community Experts - *jQuery Cookbook*

Resurse online

1. laravel.com/docs/5.7 - *Laravel 5.7 Documentantation*
2. goalkicker.com – *PHP . Notes for Professionals*
3. goalkicker.com - *MySQL . Notes for Professionals*
4. goalkicker.com - *HTML . Notes for Professionals*
5. goalkicker.com - *CSS . Notes for Professionals*
6. goalkicker.com - *jQuery . Notes for Professionals*
7. hackernoon.com/eloquent-relationships-cheat-sheet-5155498c209 - *Eloquent ORM Relations*
8. science.upm.ro/~traian/web_curs/Web_tech/lucr_stud/Micheu_Katalin.pdf - *MVC*
9. dev.to/williamragstad/how-to-use-ajax-3b5e - *AJAX*

7 ANEXE

Listă de figuri

Fig. 1.1 - Logo Cărturești [Sursa : carturesti.ro]	4
Fig. 2.2 - Logo Libris [Sursa : libris.ro]	4
Fig. 1.3 - Topul framework-urilor web în PHP [Sursa : coderseye.com]	6
Fig. 1.4 - Topul framework-urilor web în general [Sursa : hotframeworks.com]	6
Fig. 2.1 - Structura arhitecturii pe niveluri [Sursa : Bibliografie 1]	6
Fig. 2.2 - Niveluri închise și parcursul cererii [Sursa : Bibliografie 1]	6
Fig. 2.3 - Structura Model – View – Controller [Sursa : Bibliografie RO 8]	6
Fig. 2.4 – Logo MySQL [Sursa : mysql.com]	10
Fig. 2.5 - Logo PHP [Sursa : php.net]	10
Fig. 2.6 - Logo Bootstrap [Sursa : getbootstrap.com]	11
Fig. 2.7 - Exemplu CSS	11
Fig. 2.8 - Logo jQuery [Sursa : jquery.com]	12
Fig. 2.9 - Exemplu simplu de program JavaScript [Sursa : Bibliografie 5]	13
Fig. 2.10 - Exemplu cu HTML,CSS și jQuery	14
Fig. 2.11 - Logo AJAX [Sursa : Bibliografie RO 9]	15
Fig. 2.12 - Exemplu AJAX post	16
Fig. 2.13 - Logo DataTables [Sursa : datatables.net]	16
Fig. 2.14 - Logo Leaflet [Sursa : leafletjs.com]	17
Fig. 2.15 - Exemplu de inițializare a hărții	17
Fig. 3.1 - Logo Laravel [Sursa : Bibliografie RO 1]	18
Fig. 3.2 - Structura fișierelor	19
Fig. 3.3 - Fișier de configurare	20
Fig. 3.4 - Trimitere de email	20
Fig. 3.5 - Rute de acces	21
Fig. 3.6 - Text escapat [Sursa : Bibliografie RO 1]	22
Fig. 3.7 - Text neescapat [Sursa : Bibliografie RO 1]	23
Fig. 3.8 - Utilitatea Blade în HTML	23

Fig. 3.9 - Relația între modele One to One [Sursa : Bibliografie RO 7]	25
Fig. 3.10 - Relația între modele One to Many [Sursa : Bibliografie RO 7]	26
Fig. 3.11 - Relația între modele Many to Many [Sursa : Bibliografie RO 7]	26
Fig. 4.1 - Formularul de înregistrare	27
Fig. 4.2 - Formular de accesare a contului	28
Fig. 4.3 - Bara interioară de navigație	28
Fig. 4.4 - Panoul din josul paginii	29
Fig. 4.5 - Afișarea cărților din magazin	29
Fig. 4.6 - Formatul de afișare al detaliilor unei cărți	30
Fig. 4.7 - Panoul comentariilor	31
Fig. 4.8 - Conținutul coșului de cumpărături	32
Fig. 4.9 - Formular folosit pentru achiziționarea unei cărți	32
Fig. 4.10 - Meniul asociat numelui de utilizator	33
Fig. 4.11 - Pagina profilului personal când nu se pot modifica câmpurile sale	33
Fig. 4.12 - Pagina profilului personal când se pot modifica câmpurile sale	33
Fig. 4.13 - Panoul de control văzut din perspectiva unui administrator	34
Fig. 4.14 - Panoul de acces la rolurile utilizatorilor	35
Fig. 4.15 - Panoul de control al cărților	36
Fig. 4.16 - Formularul de creare a unei noi cărți – Secțiunea I	37
Fig. 4.17 - Formularul de creare a unei noi cărți – Secțiunea II	37
Fig. 4.18 - Formularul de creare a unei noi cărți – Secțiunile III și IV	38
Fig. 4.19 - Panoul de control asupra categoriilor și cărților ce o dețin	39
Fig. 4.20 - Panoul de control al stocului produselor	40
Fig. 4.21 - Panoul de control al cererilor partenerilor	40
Fig. 4.22 - Panoul lateral de control asupra stării cererii din perspectiva unui administrator	41
Fig. 4.23 - Panoul lateral de control asupra stării cererii din perspectiva unui partener	41
Fig. 4.24 - Formularul de afișare a detaliilor unei facturi	42

Crearea unei noi cărți

```
/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */

public function store(Request $request)
{
    $validator = Validator::make($request->all(),
    [
        'title' => 'required|string|max:101|unique:books,title',
        'author' => 'required|string|max:101',
        'publishing_house' => 'required|string|max:101',
        'description' => 'required|string|min:30|max:2000',
        'photo' => 'image',
        'price' => 'required|numeric|min:1',
    ], [
        'required' => 'The :attribute field is required! ',
        'string' => 'The :attribute field must be a string! ',
        'min' => [
            'string' => 'The :attribute field must have at least :min characters!',
            'numeric' => 'The :attribute field must be at least :min! '
        ],
        'max' => [
            'string' => 'The :attribute field may not be greater than :max characters!',
        ],
        'unique' => 'The :attribute field value has already been used! ',
        'image' => 'The :attribute field must be an image! ',
        'numeric' => 'The :attribute field must be a number! ',
    ]);

    if ($validator->fails()) {
        return back()->with('errors', $validator->errors());
    } else {
```

```

$book = Book::create([
    'title' => $request->title,
    'author' => $request->author,
    'publishing_house' => $request->publishing_house,
    'description' => $request->description,
    'price' => $request->price,
]);

if ($request->has('photo')) {
    $image = $request->file('photo');
    $filename = $book->book_id . '.' . $image->getClientOriginalExtension();
    $location = public_path('images/books-covers/');
    Image::make($image->resize(250, 400)->save($location . $filename);
    $book->photo = $filename;
    $book->save();
} else
    if (file_exists(public_path('images/helpers/DraggedAndDropped.jpg'))) {
        $old_path = public_path('images/helpers/DraggedAndDropped.jpg');
        $filename = $book->book_id . '.jpg';
        $new_path = public_path('images/books-covers/' . $filename);
        rename($old_path, $new_path);
        $book->photo = $filename;
        $book->save();
    }

if ($request->categories)
    foreach ($request->categories as $category_name) {
        $category = Category::where('name', $category_name)->first();
        $book->categories()->attach($category);
    }

$book->inShop = (Auth::user()->isAdmin()) ? 1 : 0;
$book->save();

$stock = Stock::create([
    'book_id' => $book->book_id,
    'amount' => 1
]);

```



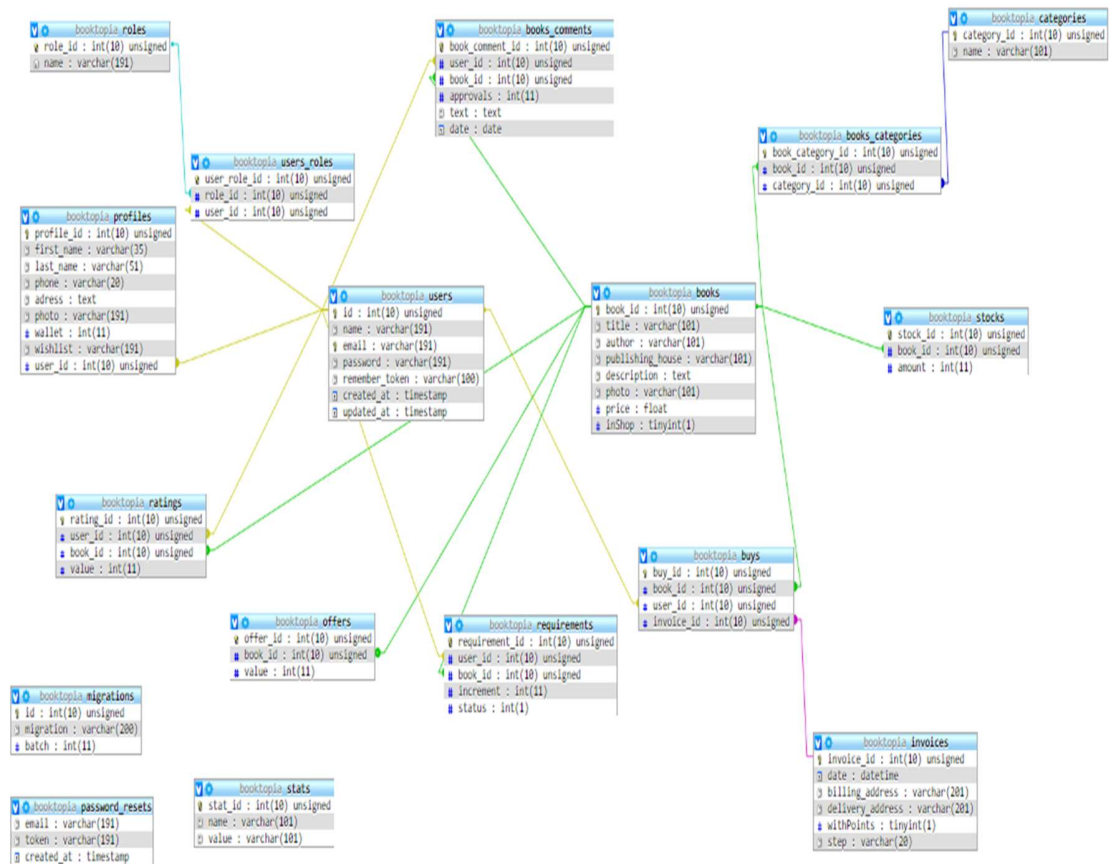
```

if(Auth::user()->isPartner())
{
    $requirement = Requirement::create([
        'user_id' => Auth::user()->id,
        'book_id' => $book->book_id,
        'increment' => 0,
        'status' => 0,
    ]);
}

Session::flash('success', Lang::get('dictionary.book.add-success'));
return redirect()->route('books.index');
}
}

```

Schema bazei de date



Dicționar de termeni și acronime

Framework	Structură conceptuală
MVC	Model-view-controller
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
PHP	Php: Hypertext Preprocessor
MySQL	Michael Widenius Structured Query Language
LAMP	Linux, Apache, MySQL, PHP
Middleware	Mecanism de filtrare a cererilor HTTP
HTTP	Hypertext Transfer Protocol
Query	Instrucțiune pentru baza de date
Website	Colecție de elemente web relaționate
URL	Uniform Resource Locator
AJAX	Asynchronous JavaScript and XML
DOM	Document Object Model
XML	Extensible Markup Language
JSON	JavaScript Object Notation
CSRF	Cross-Site Request Forgery
PDO	PHP Data Objects
ORM	Object-relational mapping
API	Application Programming Interface