Tema: Stripovi  
Predmet: Internet Tehnologije

76/18 Danica Zdravković

45/18 Željana Grujić

241/18 Marko Vukosavljević

# Sadržaj

[Sadržaj 2](#_Toc95324966)

[Korisnički zahtev 4](#_Toc95324967)

[Opis tehnologija 4](#_Toc95324968)

[Laravel 4](#_Toc95324969)

[React 4](#_Toc95324970)

[Bootstrap 4](#_Toc95324971)

[Opis sistema 5](#_Toc95324972)

[Relacioni model: 5](#_Toc95324973)

[Funkcionalnosti 6](#_Toc95324974)

[korisnik 6](#_Toc95324975)

[Administrator 6](#_Toc95324976)

[Neprijavljen korisnik 6](#_Toc95324977)

[Dokumentacija REST API-ja 7](#_Toc95324978)

[knjiga 7](#_Toc95324979)

[App\Http\Controllers\PageController@books 7](#_Toc95324980)

[App\Http\Controllers\BookController@getBooksByIds 7](#_Toc95324981)

[App\Http\Controllers\BookController@deleteBook 7](#_Toc95324982)

[App\Http\Controllers\BookController@deleteBook 8](#_Toc95324983)

[author 8](#_Toc95324984)

[App\Http\Controllers\AuthorController@createAuthor 8](#_Toc95324985)

[App\Http\Controllers\AuthorController@getAuthors 8](#_Toc95324986)

[order 8](#_Toc95324987)

[App\Http\Controllers\OrderController@getAllOrders 8](#_Toc95324988)

[App\Http\Controllers\OrderController@getMyOrders 9](#_Toc95324989)

[App\Http\Controllers\OrderController@updateOrderStatus 9](#_Toc95324990)

[App\Http\Controllers\OrderController@createOrder 9](#_Toc95324991)

[Order Status 10](#_Toc95324992)

[App\Http\Controllers\OrderController@getOrderStatuses 10](#_Toc95324993)

[Languages 10](#_Toc95324994)

[App\Http\Controllers\LanguageController@getLanguages 10](#_Toc95324995)

[Categories 11](#_Toc95324996)

[App\Http\Controllers\CategoryController@getCategories 11](#_Toc95324997)

[Korisničko uputstvo 11](#_Toc95324998)

[Uputstvo za korišćenje 11](#_Toc95324999)

[Kao korisnik 11](#_Toc95325000)

[Kao administrator 17](#_Toc95325001)

[Karakteristični delovi koda 20](#_Toc95325002)

[Statistika prodaja jedne knjige 20](#_Toc95325003)

[Filtriranje i paginacija 22](#_Toc95325004)

[Filtriranje 22](#_Toc95325005)

[Paginacija 24](#_Toc95325006)

[Github 25](#_Toc95325007)

[Literatura 25](#_Toc95325008)

# Korisnički zahtev

Potrebno je napraviti aplikaciju koja će omogućiti prodaju stripova odnosno knjiga kao i praćenje prodaja tj. upravljanje porudžbinama. Pri ulasku na aplikaciju, korisnik bi trebalo da vidi knjige sa sajta, filtere kao i paginaciju zbog lakšeg pretraživanja određene knjige. Takođe, potrebno je da korisnik ima mogućnost dodavanja knjige u korpu, ali samo ako je prethodno prijavljen na sistem. Ako korisnik nije prijavljen na sistem, moći će da puni korpu, ali tek kada se prijavi ili registruje će moći da izvrši pravljenje pordužbine tj. da poruči željene proizvode. Takođe, treba prikazati količinu prodaja u poslednjih sedam dana, za svaki dan po proizvodu, kako bi korisnici mogli da vide

popularnost određene knjige.

Sa druge strane, osim korisnika, koji predstavlja jednu ulogu u sistemu, postoji i administrator koji predstavlja drugu ulogu u sistemu. Uloga administratora je da upravlja porudžbinama, odnosno da prati porudžbine, menja im status kada one postanu pripremljene, kreirane, isporučene i sl. Takođe, administrator ima mogućnosti upravljanja knjigama i autorima.

Određenu porudžbinu mogu videti samo korisnik koji je poručio proizvod i svaki administrator, kako ne bi došlo do mogućih zloupotreba privatnosti.

# Opis tehnologija

## Laravel

Laravel je MVC (model view controller) framework koji služi za brzu izradu skalabilnih veb aplikacija. Njegova snaga leži u veoma intuitivnom kodu, intuitivnim metodama i veoma lakom povezivanju sa bazom. Laravel takođe nudi veoma jak eloquent model za vršenje operacija nad bazom, povezivanje više tabela kroz relacije.

## React

React je frontend javascript biblioteka koja omogućava pravljenje izgleda aplikacija kroz izgradnju komponenata. Komponenta se sastoji od nekog HTML koda i neke logike komponente odnosno javascripta. Veliki broj aplikacija je urađen uz pomoć Reacta i njegova snaga leži u lakoći komunikacije između komponenti.

## Bootstrap

Bootstrap je framework koji se oslanja na CSS I delove JavaScript-a koji se odnose na logiku CSS komponenti. Omogućava brže I lakše pravljenje veb aplikacija.

# Opis sistema

## Relacioni model:

Graphical user interface, text, application

Description automatically generated  
***Users***tabela čuva sve podatke o registrovanim korisnicima I njihovim ulogama.

***Roles*** tabela sadrži sve postojeće uloge u sistemu.

***Books*** je tabela koja sadrzi sve knjige na stanju

***Languages*** jetabela koja sardži postojeće jezike knjiga u sistemu

***Categories*** je tabela koja sadrzi postojece kategorije knjiga u sistemu

***Authors*** je tabela koja sadrzi sve postojece autore knjiga u sistemu

***Orders*** je tabela koja sadrzi sve porudzbine knjiga koje su se ikada desile

**Order\_statuses** je tabela koja sadrzi postojece statuse porudzbina u sistemu

***Books\_orders*** je tabela koja sadrzi sve knjige povezane sa odgovarajucom porudzbinom

***Authors\_books*** je tabela koja povezuje sve autore koji su radili na odgovarajucoj knjizi

## Funkcionalnosti

### korisnik

* Registracija
* Login
* Pregled svih knjiga
* Filtriranje knjiga
* Dodavanje knjiga u korpu
* Kreiranje porudzbine
* Pregled svojih porudzbina
* Pregled jedne porudzbine (ako je korisnik napravio)

### Administrator

* Login
* Pregled svih knjiga
* Filtriranje knjiga
* Dodavanje novog autora
* Dodavanje nove knjige
* Brisanje knjiga
* Pregled svih porudzbina
* Pregled jedne porudzbine

### Neprijavljen korisnik

* Pregled svih knjiga
* Dodavanje knjige u korpu

## Dokumentacija REST API-ja

### knjiga

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\PageController@books |
| Opis: | Ovaj API endpoint vraca sve ili filtriranje knjige u objektu paginacije. |
| Endpoint: | /**api/books** |
| Metoda: | *GET* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Query:   * Languages[] * Categories[] * Authors[] |
| Izlaz: | Laravel Pagination Objekat |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\BookController@getBooksByIds |
| Opis: | Vraca sve knjige po prosledjenim idevima |
| Endpoint: | /**api/books-by-ids** |
| Metoda: | *GET* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Query   * Ids[] |
| Izlaz: | {  books: Array<Book> } |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\BookController@deleteBook |
| Opis: | Brisanje postojeće knjige, koristi AdminMiddleware |
| Endpoint: | /**api/admin/books/{id}** |
| Metoda: | *DELETE* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Path:  *id –* knjige za brisanje |
| Izlaz: | {  message: Knjiga uspesno obrisana.'  } |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\BookController@deleteBook |
| Opis: | Omogućava doktoru da se prijavi za postojeće prijave, ako već nemaju svoga doktora. |
| Endpoint: | /**api/prijave/{prijava}/update** |
| Metoda: | *GET|HEAD* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Path:  *prijava –* id prijave |
| Izlaz: | {  poruka: 'Uspesno ste dodelili pacijenta sebi.'  } |

### author

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\AuthorController@createAuthor |
| Opis: | Pravi novog autora u bazi |
| Endpoint: | /**api/admin/authors** |
| Metoda: | *POST* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Form data   * Author\_name * Author\_description * Author\_year\_of\_birth * Author\_year\_of\_death |
| Izlaz: | {  Message: Autor uspesno dodat  } |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\AuthorController@getAuthors |
| Opis: | Pravi novog autora u bazi |
| Endpoint: | /**api/authors** |
| Metoda: | *GET* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | / |
| Izlaz: | Pagination Objekat Autor |

### order

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\OrderController@getAllOrders |
| Opis: | Dohvata sve postojece porudzbine iz baze |
| Endpoint: | /**api/admin/orders** |
| Metoda: | *GET|HEAD* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | / |
| Izlaz: | Datatable Objekat Porudzbine |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\OrderController@getMyOrders |
| Opis: | Dohvata sve postojece korisnikove porudzbine iz baze |
| Endpoint: | /**api/my-orders** |
| Metoda: | *GET|HEAD* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | / |
| Izlaz: | Datatable Objekat Porudzbine |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\OrderController@updateOrderStatus |
| Opis: | Menja status porudzbine na zeljeni |
| Endpoint: | /**api/admin/orders/{id}** |
| Metoda: | *PUT* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Putanja:   * Id – id porudzbine   Form data:   * order\_status\_id |
| Izlaz: | Ako je uspesno izmenjen status: {  'message': 'Uspesno promenjen status porudzbine.'  }  Ako nije uspesno izmenjen status:  {  'message': Greska prilikom promene statusa porudzbine.'  } |

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\OrderController@createOrder |
| Opis: | Menja status porudzbine na zeljeni |
| Endpoint: | /**api/orders** |
| Metoda: | *POST* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | Form data:   * books |
| Izlaz: | {  “message”: “Uspesno ste narucili proizvode”  } |

### Order Status

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\OrderController@getOrderStatuses |
| Opis: | Dohvata trenutno dostupne statuse poruzbine |
| Endpoint: | /**api/order-statuses** |
| Metoda: | *GET* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | / |
| Izlaz: | {  orderStatuses: Array<OrderStatus>  } |

### Languages

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\LanguageController@getLanguages |
| Opis: | Dohvata trenutno dostupne jezike |
| Endpoint: | /**api/languages** |
| Metoda: | *GET* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | / |
| Izlaz: | {  languages: Array<Language>  } |

### Categories

|  |  |
| --- | --- |
| Kontroler: | App\Http\Controllers\CategoryController@getCategories |
| Opis: | Dohvata trenutno dostupne kategorije |
| Endpoint: | /**api/categories** |
| Metoda: | *GET* |
| Zaglavlje: | X-XSRF-TOKEN, laravel\_token (cookie) |
| Parametri: | / |
| Izlaz: | {  categories: Array<Category>  } |

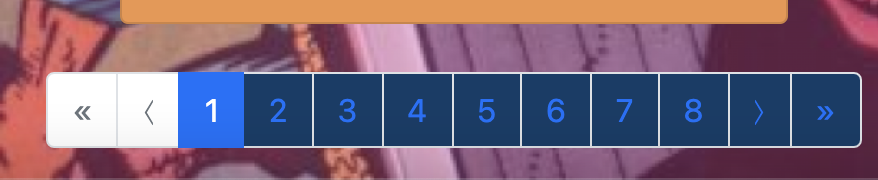
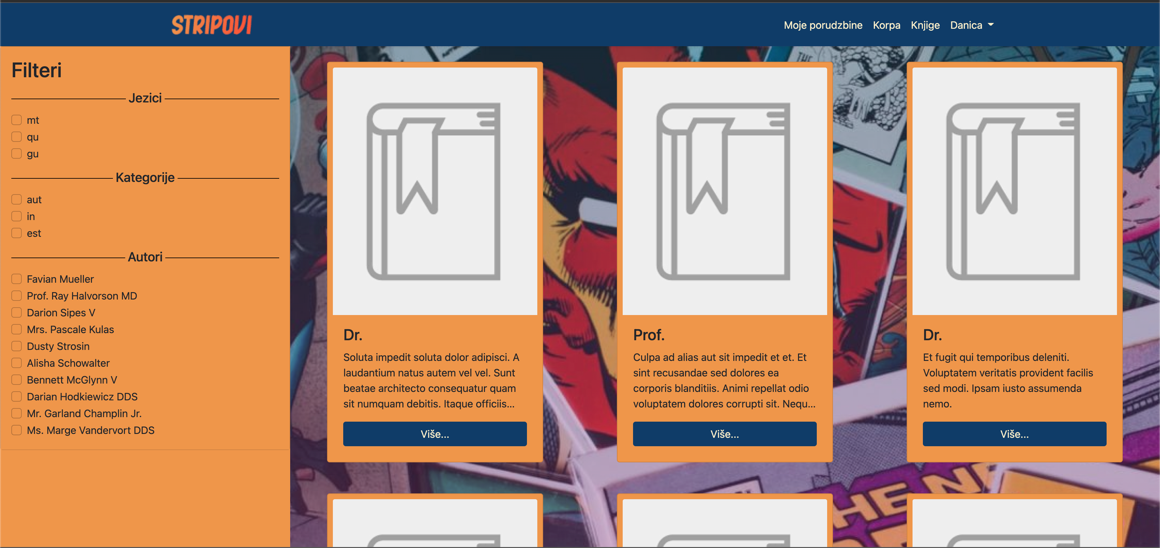
# Korisničko uputstvo

Stripovi su aplikacija koja je napravljena kao projektni rad iz predmeta Internet tehnologije. Korišćene su tehnologije Laravel i React. Poenta aplikacije je da omoguci online kupovinu knjiga i stripova kroz jednostavan I jasan korisnicki interfejs.

## Uputstvo za korišćenje

### Kao korisnik

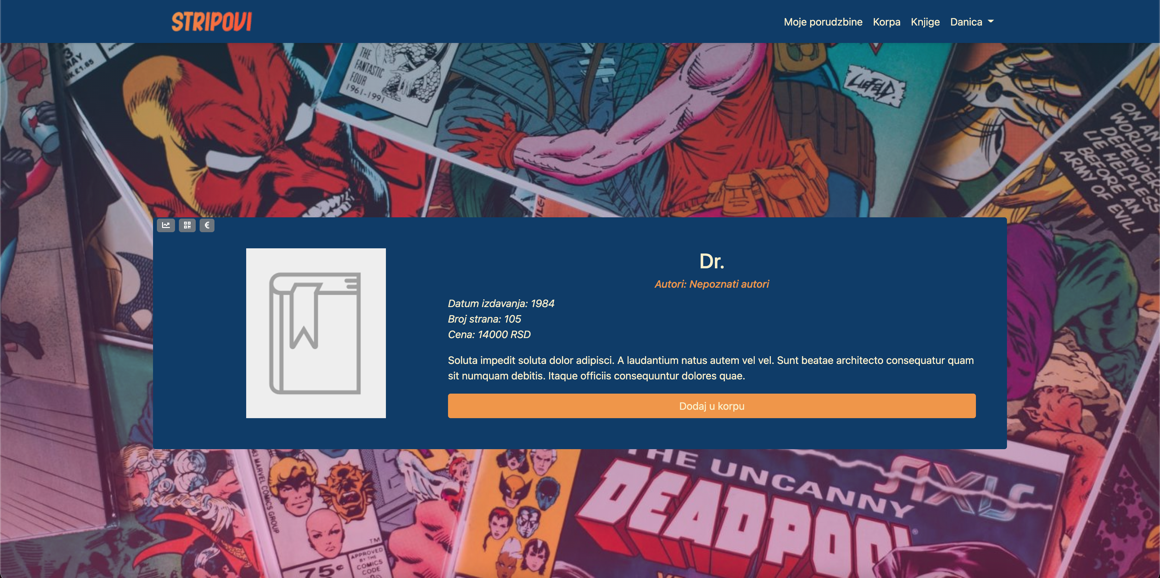
Po ulasku na aplikaciju, korisnik će biti prebačen na stranicu dobrodošlice. Klikom na dugme „Knjige“ će biti prebačen na stranicu na kojoj može videti sve knjige. Postoji opcija filtriranja i paginacija knjiga kako bi lakše našli omiljenu knjigu.

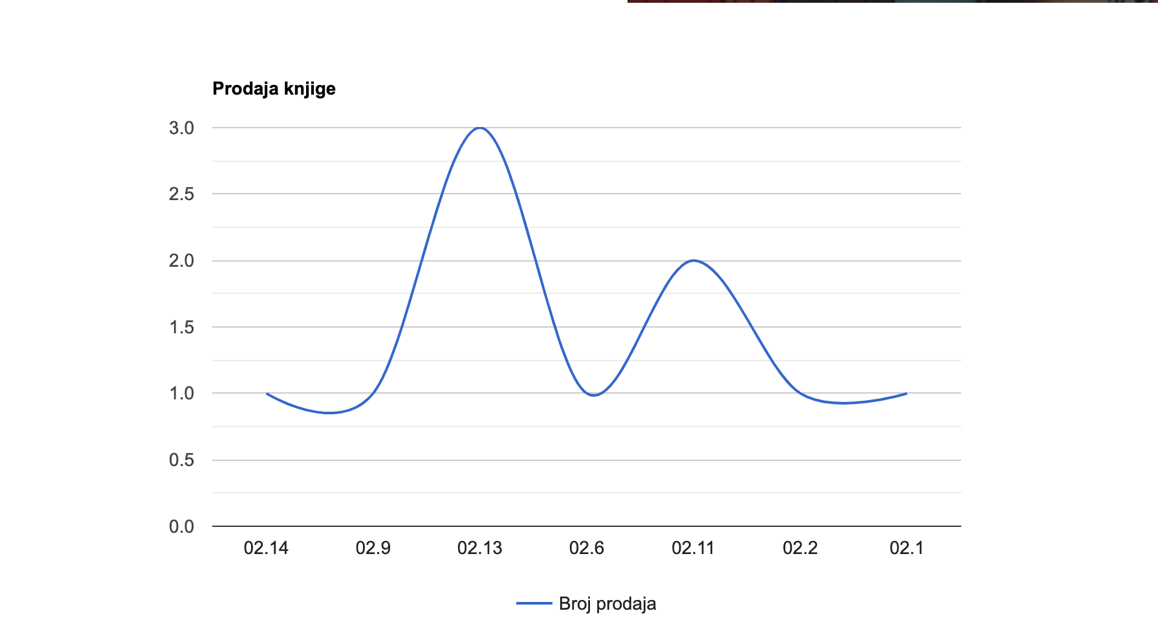


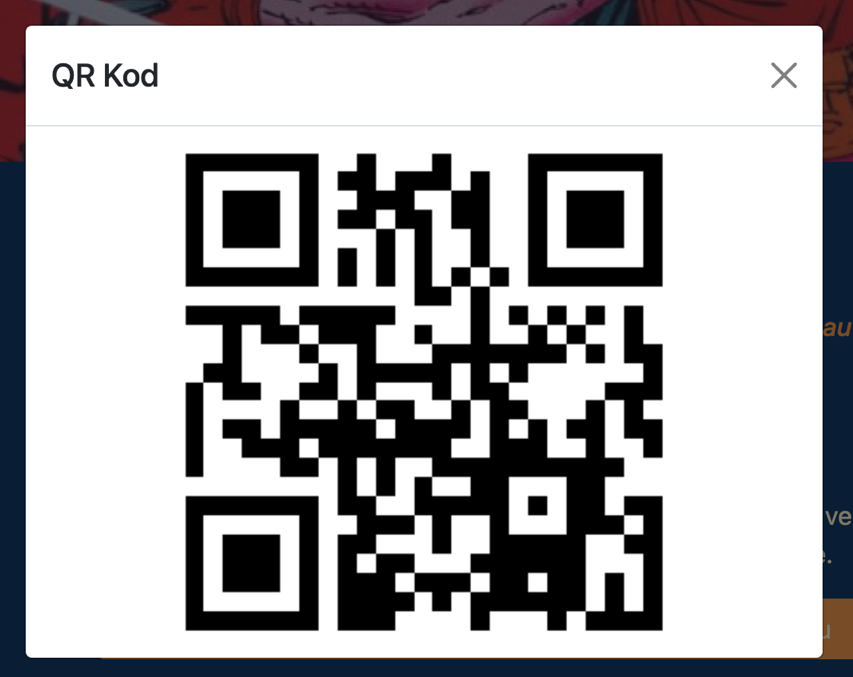
Kada konačno izabere knjigu koja mu se najviše sviđa, klikom na dugme “Više” koje se nalazi ispod knjige, može da vidi više informacija o toj knjizi. U uglu postoje tri ikonice, a to su za:

1. statistiku prodaja knjiga
2. QR Code
3. Cena u drugim valutama

Klikom na bilo koju ikonicu pojaviće se odgovarajući prozor koji nosi informacije vezane za izabranu opciju.

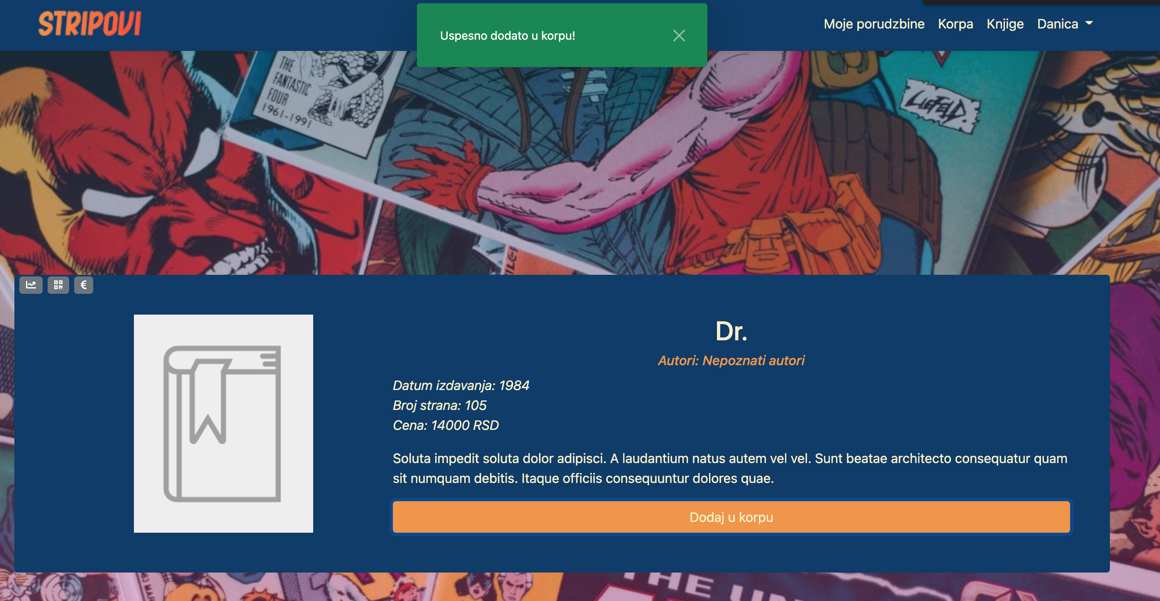


Klikom na prvu opciju, možemo videti količinu prodate knjige u poslednjih 7 dana. 

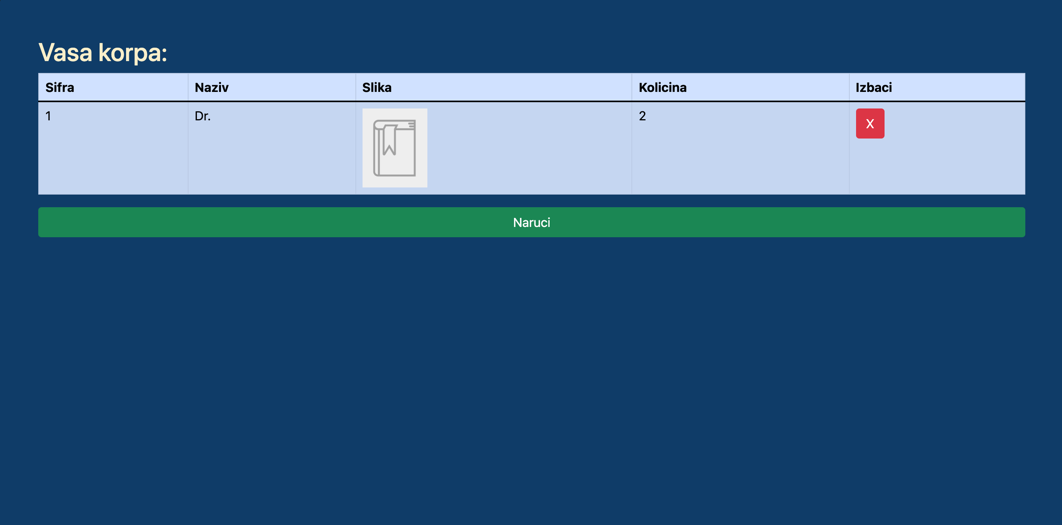
Klikom na drugu opciju, otvara se QR kod koji može da se odnese zaposlenom u prodavnici da bi zaposleni mnogo lakše došao do odgovarajuće knjige I uneo je u porudžbinu.  


Treća opcija nam pokazuje cenu knjiga u nekoj drugoj valuti. Da bismo se lakše snašli u ovoj tabeli, treba koristiti prečicu CTRL plus F A picture containing graphical user interface

Description automatically generated

Kada proveri sve željene informacije vezano za određenu knjigu, korisnik može da doda tu knjigu u korpu klikom na dugme “Dodaj u korpu”. 

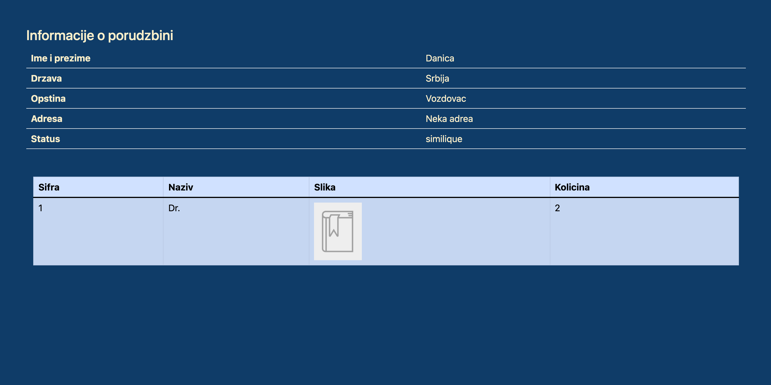
Zatim, sve knjige iz korpe može videti na jednom mestu klikom na dugme “Korpa”



Ako sve izgleda kako bi trebalo, i ništa ne treba da se izbaci, klikom na dugme “Naruči” može da naruči proizvode.A picture containing text, indoor, screenshot

Description automatically generated

Takođe, klikom na dugme “Moje Porudžbine” prebacuje se na stranicu gde ima uvid I pregled svih svojih porudžbina. Detaljni pregled se može videti klikom na oko u poslednjoj koloni tabele.  
A picture containing timeline

Description automatically generated

### Kao administrator

Administrator ima uvid u sve porudžbine klikom na dugme “Porudžbine”. Tu može pregledati i detaljno videti svaki porudžbinu.

A screenshot of a computer

Description automatically generated with medium confidence

Klikom na trenutni status, otvara se padajuća lista sa mogućim statusima porudžbine pa na taj način, administrator može korisniku da “javi” kroz aplikaciju status neke njegove porudžbine. Takođe, kao I korisnik može da pristupi pojedinačnoj porudžbini, ali u ovom slučaju ima pravo da vidi sve pojedinačne porudžbine, a ne samo svoje.A screenshot of a computer

Description automatically generated with medium confidence

Na stranici “Knjige”, kao I kod korisnika, administrator može da filtrira kroz knjige. Dodatna funkcionalnost kod ove stranice je što administrator može da doda novu knjigu ili obrise postojeću ili doda novog autora. A picture containing text, screenshot

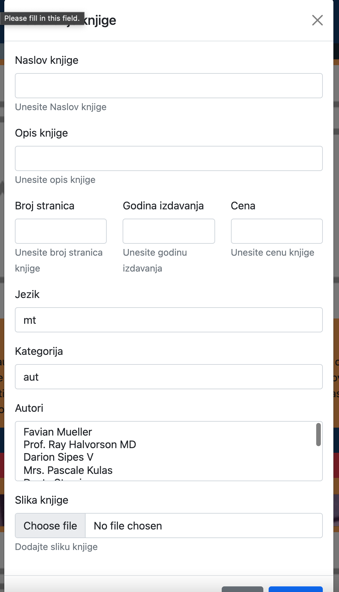
Description automatically generated

Brisanje knjige je veoma intuitivno i radi se klikom na dugme “Obriši” koje se nalazi ispod knjige.Graphical user interface

Description automatically generated

Klikom na ­+ koji se nalazi u donjem desnom ćošku ekrana, otvara se prozor sa opcijama za dodavanje novog autora ili nove knjige. Klikom na bilo koju od ovih opcija se otvara prozor koji omogućava određenu akciju.  
Graphical user interface

Description automatically generated  
  
Nakon popunjavanja forme za dodavanje autora ili knjige, dugme „Napravi“ će povrditi polja i knjiga će biti napravljena.  
Graphical user interface, application

Description automatically generated

# Karakteristični delovi koda

Možemo izdvojiti dva najkarakteristicnija dela koda:

1. Način na koji se generiše statistika prodaja jedne knjige
2. Filtriranje i paginacija knjiga

## Statistika prodaja jedne knjige

Kako bismo prikazali statistiku prodaja neke knjige, prvenstveno moramo da prikupimo informacije iz baze potrebne za našu knjigu. To uspevamo metodom *getBook* koja, kada pogodimo web endpoint za pregled jedne knjige, vraća stranicu te knjige (*pages.user.book-user*) i knjigu. Ta knjiga, uz pomoć metode *whereHas*, uz sebe ima i sve porudžbine u kojoj se nalazi. Unutrašnji upit ove metode nam govori da želimo da izvučemo samo one porudžbine koje nisu strarije od 7 dana.

public function getBook($id)

{

$book = Book::where('id', $id)->with(['language', 'category', 'authors'])->whereHas("orders", function ($query) {

$query->where("orders.created\_at", ">=", Carbon::now()->subDays(7));

})->first();

return view('pages.user.book-user', ['book' => $book]);

}

Kada smo izvukli sve potrebne informacije o porudžbinama, ostaje da u *dataset* nekog elementa (neka bude div I nazvaćemo ga *curve\_chart*)ubacimo potrebne informacije o porudžbinama i enkodiramo tako što ćemo ga pretvoriti u json string.

<div id="curve\_chart"

data-orders="{{ json\_encode($book->orders) }}"

style="width: 900px; height: 500px">

</div>

Sledeći korak predstavlja obradu i preslikavanje podataka iz standardnog oblika porudžbina u oblik neophodan za unos u google charts API. Neophodan oblik je matrica, odnosno niz u nizu, gde prvi element (prvi niz) predstavlja kriterijume, a ostali podatke koji će se na njima nalaziti. Poenta sledećeg segmenta koda je da u objekat „counts“ ubacimo ključ-vrednost atribute tako da odgovaraju sledećem: <dan>.<mesec>: <broj prodaja tog dana>

<script type="text/javascript">

google.charts.load('current', {

'packages': ['corechart']

});

google.charts.setOnLoadCallback(drawChart);

const orders = $('#curve\_chart').data('orders');

orders.forEach(o => {

o.formattedDate = `0${new Date(o.created\_at).getMonth() + 1}.${new Date(o.created\_at).getDate()}`

o.quantity = 0;

})

let counts = {};

for (const o of orders) {

counts[o.formattedDate] = counts[o.formattedDate] ? counts[o.formattedDate] + 1 : 1;

}

Na kraju, ostaje da prikažemo krivu, to radimo sledećom metodom.

function drawChart() {

var data = google.visualization.arrayToDataTable([

['Dan', 'Broj prodaja'],

...Object.entries(counts)

]);

console.log('test')

var options = {

title: 'Prodaja knjige',

curveType: 'function',

legend: {

position: 'bottom'

}

};

var chart = new google.visualization.LineChart(document.getElementById('curve\_chart'));

chart.draw(data, options);

}

## Filtriranje i paginacija

Filtriranje i paginacija je takođe zanimljiv segment koda jer način na koji je implementiran ima mogućnost da bude skalabilan, odnosno može se dodati neograničen broj filtera veoma lako.

### Filtriranje

Proces počinje onog trenutka kada komponenta *BookContainer* pozove komponentu *FilterComponent* i tako joj prosledi *handleAddFilter* funkciju.

<FilterComponent

toggleFilter={handleAddFilter}

/>

Zatim, u filter komponenti, ta metoda se koristi svaki put kada je neki checkbox uključen ili isključen. Uzećemo primer za kategorije. Vidimo da se ovoj funkciji šalju 2 parametra – naziv i id.

<input

onClick={(e) => { props.toggleFilter('categories', e.target.id) }}

type="checkbox"

class="form-check-input" name=""

id={l.id} value="checkedValue"

/>

Kada se vrednost čekboksa promeni, poziva se metoda handleAddFilters koja izgleda ovako:

const handleAddFilter = (type, value) => {

const filter = `${type}[]=${value}`;

if (filters.includes(filter)) {

setFilters([...filters.filter(existingFilter => existingFilter !== filter)])

loadBooks(booksPagination.current\_page, [...filters.filter(existingFilter => existingFilter !== filter)]);

}

else {

setFilters([...filters, filter])

loadBooks(booksPagination.current\_page, [...filters, filter]);

}

}

Kao što vidimo, ova metoda spaja vrednost prvog i drugog parametra i podešava niz filtera na sada nove filtere. Prvo proveri da li postoji određeni filter, ako ne postoji dodaje ga u niz, ako postoji briše ga iz niza. Na primer, ako je u pitanju neka kategorija sa identifikatorom 4, novi filter bi bio *categories[]=4*. Zatim se poziva funkcija za učitavanje knjiga, odnosno *loadBooks.* Ona izgleda ovako:

const loadBooks = async (pageNumber, filters = null) => {

const books = (await getBooks(pageNumber, filters)).data.books

setBooksPagination({ ...books });

}

Ona zatim poziva asinhronu funkciju *getBooks* koja na osnovu filtera šalje zahtev ka bekendu.

export async function getBooks(pageNumber = 1, filters) {

if (filters)

return axios.get('/api/books?page=' + pageNumber + '&' + filters?.join('&'));

return axios.get('/api/books?page=' + pageNumber);

}

A na bekendu, filteri su implementirani na ovaj način:

public function getBooks(Request $request)

{

$builder = Book::query();

$term = $request->query();

if (!empty($term['languages'])) {

$builder->whereIn('language\_id', $term['languages']);

}

if (!empty($term['categories'])) {

$builder->whereIn('category\_id', $term['categories']);

}

if (!empty($term['authors'])) {

$builder->whereHas('authors', function ($q) use ($term) {

$q->whereIn('authors.id', $term['authors']);

});

}

$books = $builder->with(['language', 'category', 'authors'])->paginate(6);

return response()->json(['books' => $books], 200);

}

### Paginacija

Za paginaciju je iskorišćen moćan Laravel Paginate metod koji omogućava veoma laku implementaciju paginacije. Takođe, na frontendu je korišćen *Laravel-react-paginate* node paket. Na promenu stranice, gorepomenuta funkcija *loadBooks* se poziva sa parametrom *pageNumber* koji predstavlja broj strancie.

<Pagination

activePage={booksPagination?.current\_page ? booksPagination?.current\_page : 0}

itemsCountPerPage={booksPagination?.per\_page ? booksPagination?.per\_page : 0}

totalItemsCount={booksPagination?.total ? booksPagination?.total : 0}

onChange={(pageNumber) => {

loadBooks(pageNumber)

}}

pageRangeDisplayed={8}

itemClass="page-item"

linkClass="page-link bg-teget"

/>

# Github

https://github.com/ZeljanaGrujic/ITEHSeminarski

# Literatura

1. <https://stackoverflow.com/>
2. <https://laracasts.com/>,
3. https://laravel.com/