Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Inženjerstvo informacionih sistema
Veb orijentisane tehnologije i sistemi

Dokumentacija za MEAN 2 web aplikaciju

Novi Sad, 2018. god.

Sadržaj

1. Uvod

1.1 Opis aplikacije

U pitanju je web aplikacija za trgovanje knjigama.

Registrovani korisnici imaju mogućnost postavljanja knjiga za prodaju, kupovine knjiga od strane drugih korisnika, učestvovanje na forumu, ocenjivanja i komentarisanja kako knjiga, tako i iskustva sa određenim korisnicima u trgovini.

Posetioci koji nisu registrovani imaju mogućnost pregleda dostupnih knjiga, objava na forumu i objava od strane drugih korisnika.

Administator ima sve mogućnosti kao i registrovani korisnik, s tim da može još da: briše i menja sve knjige, vidi listu svih registrovanih korisnika sajta, briše i menja sve objave na forumu.

1.2 Tehnologije

Za izradu aplikacije su korišćene sledeće tehnologije:
Angular, HTML; CSS i Bootstrap za Frontend i
NodeJS, ExpressJS, Moongose i MongoDB za Backend.

Za izradu aplikacije je korišćeno razvojno okruženje Visual Studio Code
(https://code.visualstudio.com/).

Aplikacija je hostovana na besplatni servis Heroku (https://www.heroku.com/), dok se baza podataka nalazi na platformi Mlab (https://mlab.com/).

1.3 Način korišćenja

Za lokalno pokretanje aplikacije neophodno je imati instaliran NodeJS(https://nodejs.org/en/), Node Package Manager (NPM) i Mongodb bazu podataka (https://www.mongodb.com/download-center?jmp=nav#community).

Prvo je neophodno klonirati sledeći repozitorijum:
https://github.com/Vukan-Markovic/Book-trading-club.

Nakon toga je potrebno izvršiti sledeće komande (u command prompt-u, u folderu gde se nalazi projekat):
1.npm install - za instaliranje zavisnosti
2. npm run-script build - za pokretanje frontend dela
3. npm start - za pokretanje backend dela

Aplikacija se takođe nalazi na sajtu: https://bookclubtrading.herokuapp.com/.
Kompletan kod je dostupan na sledećem linku:
https://github.com/Vukan-Markovic/Book-trading-club.

1.4 Osnovna struktura aplikacije

Osnovna struktura aplikacije je kreirana upotrebom ExpressJS generatora (https://expressjs.com/en/starter/generator.html).

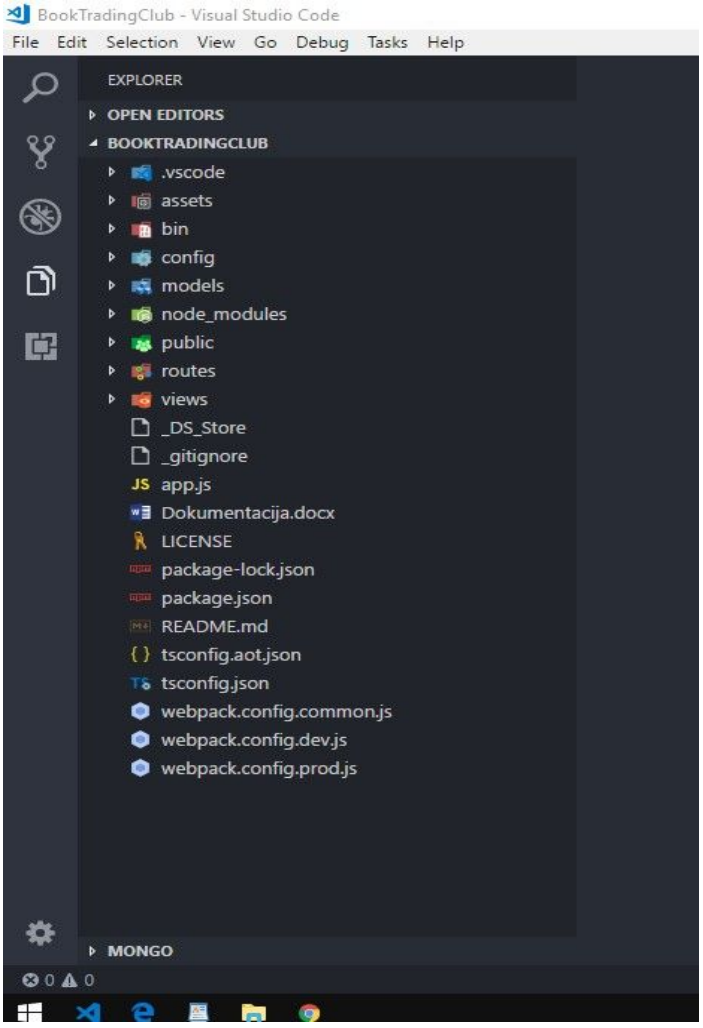
Za generisanje strukture aplikacije ExpressJS generatorom potrebno je izvršiti sledeće komande:

1. npm install -g express-generator - instaliranje express js framework-a
2. express mojaPrvaAplikacija - generisanje aplikacije sa Jade templating-om:
3. npm install - instaliranje zavisnosti.

Glavne celine aplikacije:

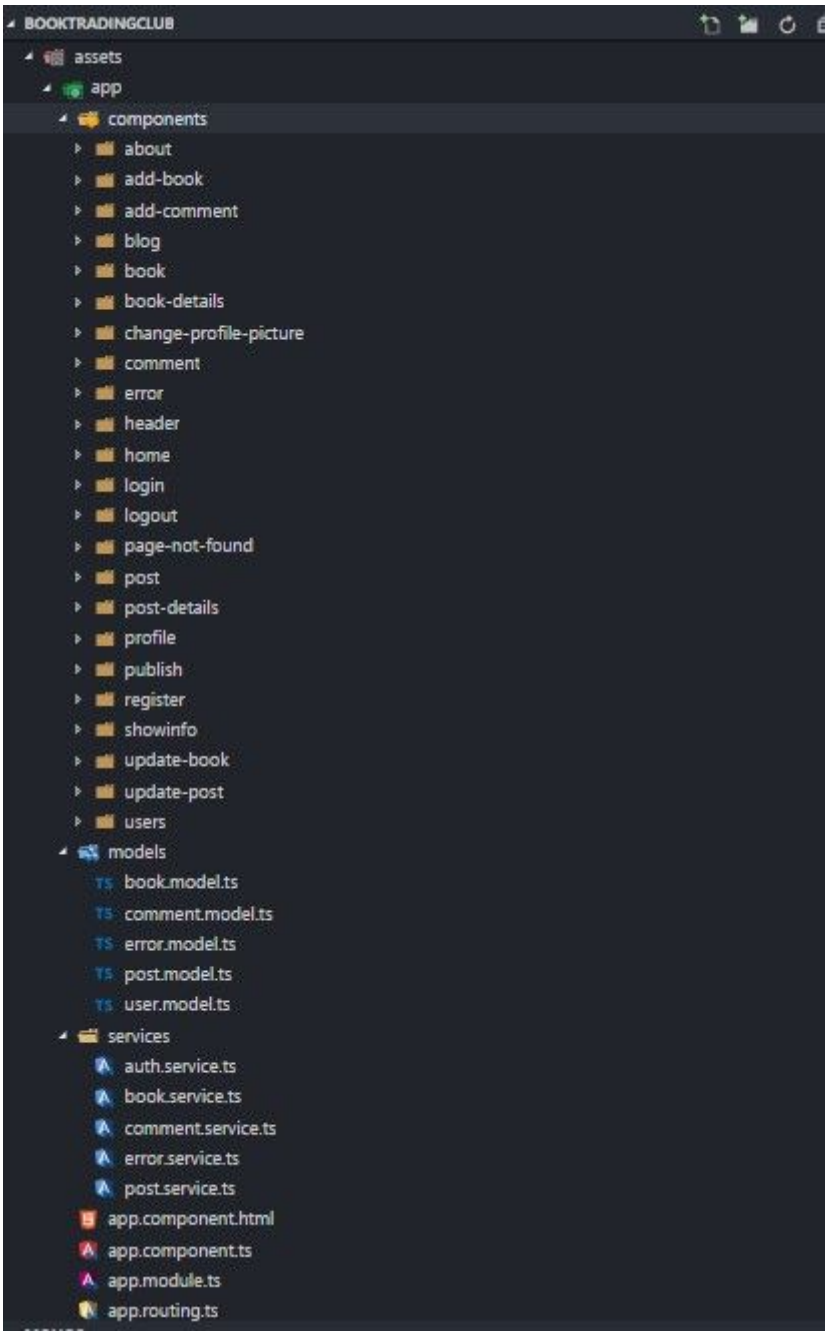
-Backend:
• Osnovna Express i Serverska konfiguracija
• Modeli
• Rute i kontroleri vezani za njih – odgovaraju HTTP zahtevima
• Autentifikacija

-Frontend:
• Modeli
• Komponente
• Servisi
• Rute
• Stilizacija
• Autentifikacija



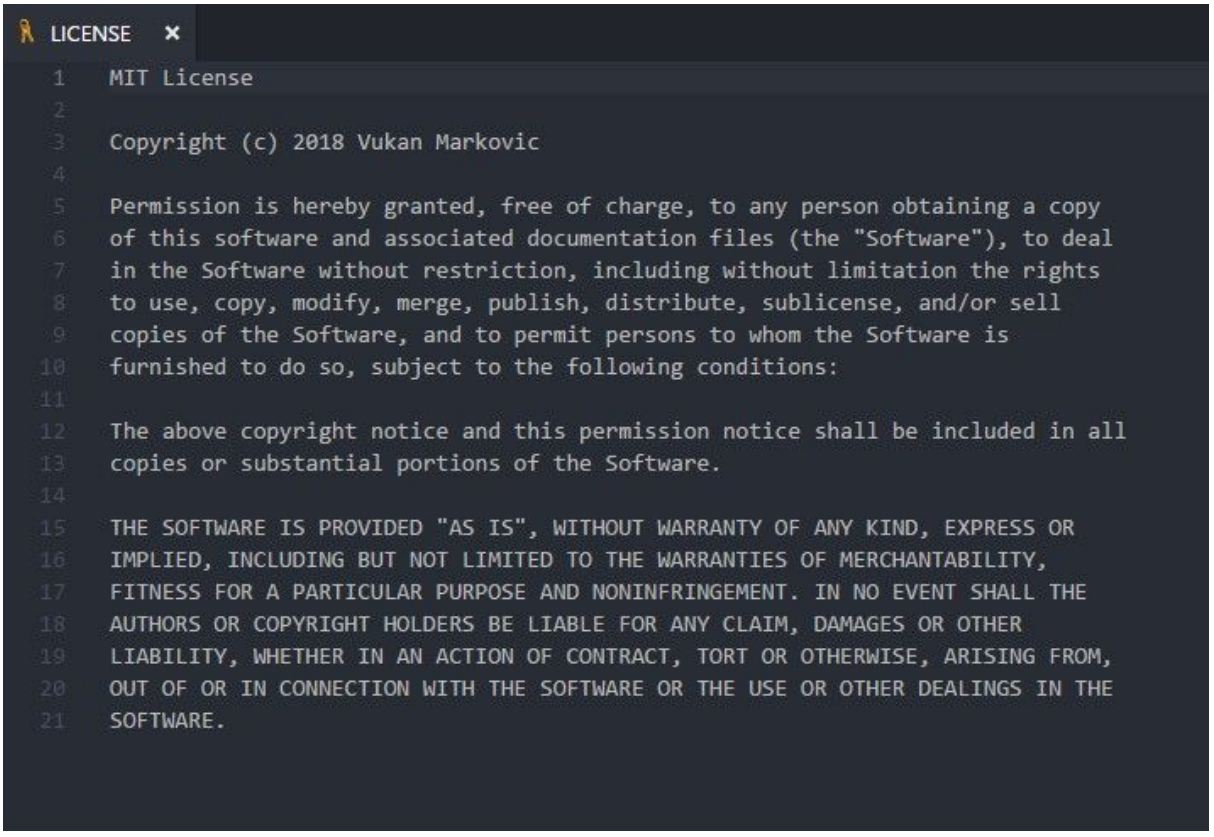
Slika 1. Glavna struktura aplikacije

Slika 2. Backend struktura aplikacije



Slika 3. Frontend struktura aplikacije

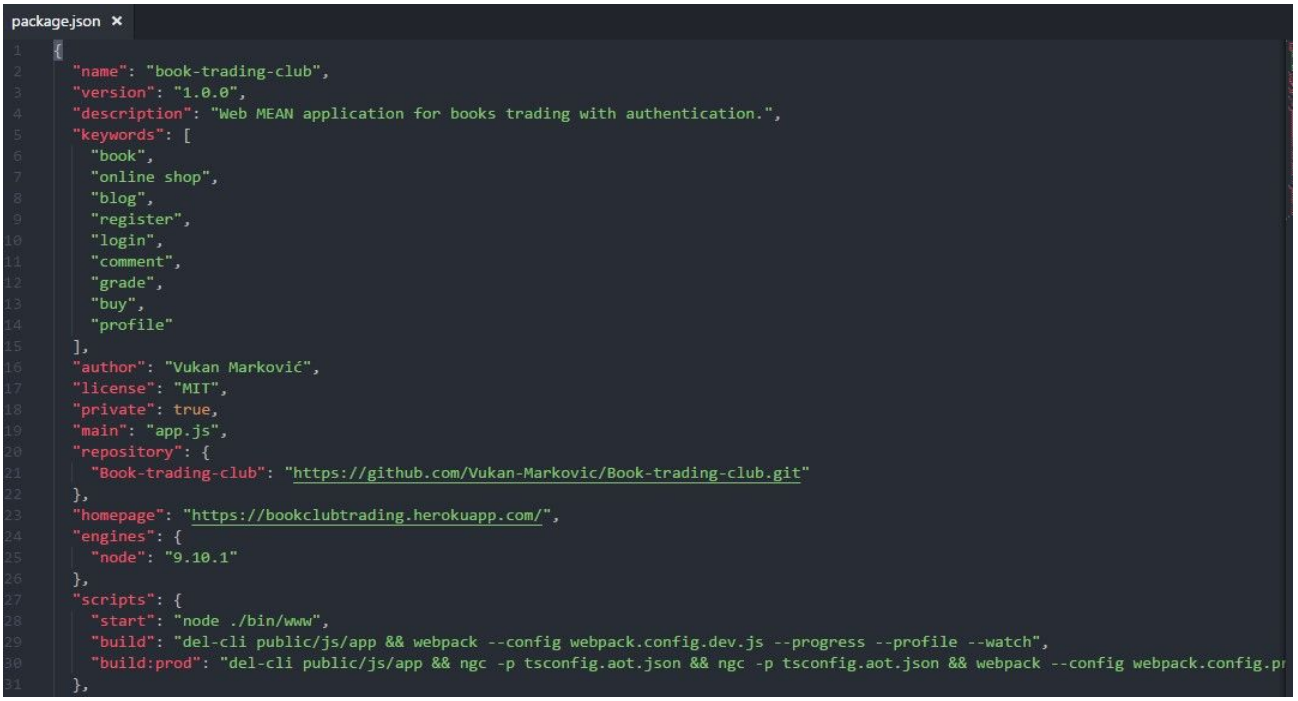
Aplikacija je razvijana pod MIT licencom.



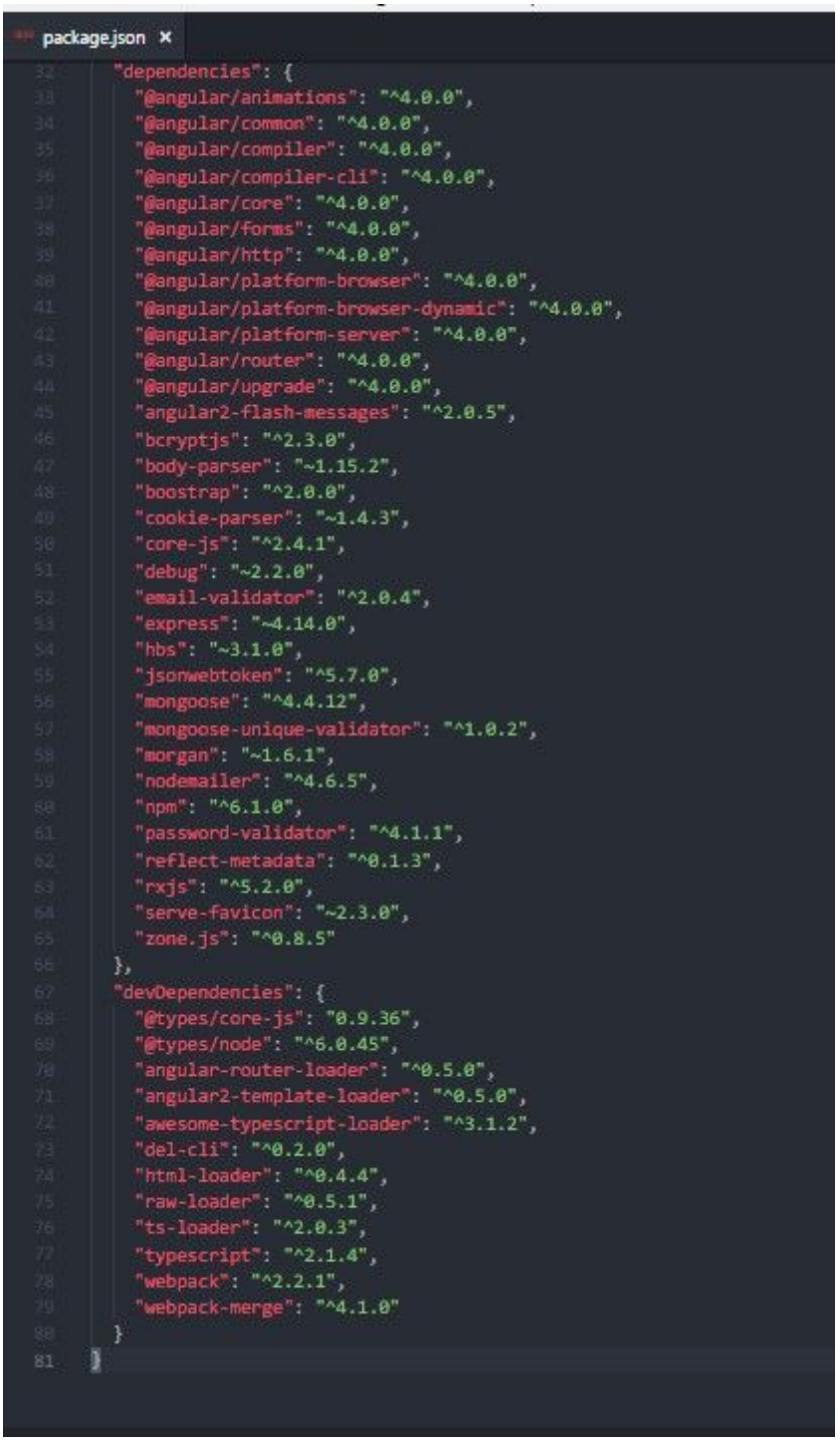
Slika 4. LICENSE fajl

package.json fajl je prisutan je u root direktorijumu u svakoj Node aplikaciji ili modulu i koristi se za definisanje svojstva paketa.

package.json



```
{
  "name": "book-trading-club",
  "version": "1.0.0",
  "description": "Web MEAN application for books trading with authentication.",
  "keywords": [
    "book",
    "online shop",
    "blog",
    "register",
    "login",
    "comment",
    "grade",
    "buy",
    "profile"
  ],
  "author": "Vukan Marković",
  "license": "MIT",
  "private": true,
  "main": "app.js",
  "repository": {
    "Book-trading-club": "https://github.com/Vukan-Markovic/Book-trading-club.git"
  },
  "homepage": "https://bookclubtrading.herokuapp.com/",
  "engines": {
    "node": "9.10.1"
  },
  "scripts": {
    "start": "node ./bin/www",
    "build": "del-cli public/js/app && webpack --config webpack.config.dev.js --progress --profile --watch",
    "build:prod": "del-cli public/js/app && ngc -p tsconfig.aot.json && ngc -p tsconfig.aot.json && webpack --config webpack.config.pr
  },
```



Slika 6. Zavisnosti u package.json fajlu

2. Backend

U www.js fajlu se vrši kreiranje servera i kreiranje i podešavanje porta. Ovaj kod je automatski generisan od strane ExpressJS generatora.

```
JS www
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('angular2-nodejs:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
```

Slika 8. Serverski www.js fajl

app.js predstavlja glavni serverski fajl aplikacije. U njemu se prvo se importuju svi potrebni moduli a zatim definišu sve rute koje će se koristiti u aplikaciji. Nakon toga se vrši konektovanje na bazu podataka. Ostala serverska podešavanja su automatski generisana od strane ExpressJS generatora.

Slika 9. Serverski app.js fajl

```
// set up body parser and cookie parser modules
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());


// set up headers
app.use((req, res, next) => {
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');
  res.setHeader('Access-Control-Allow-Methods', 'POST, GET, PATCH, DELETE, OPTIONS');
  next();
});

// define routes
app.use('/book', bookRoutes);
app.use('/user', userRoutes);
app.use('/post', postRoutes);
app.use('/comment', commentRoutes);
app.use('/', appRoutes);

// catch 404 and forward to error handler
app.use(function(req, res) {
  return res.render('index');
});

// export app module
module.exports = app;
```

Slika 10. Serverski app.js fajl

U database.js fajlu se nalazi modul sa linkom na Mlab bazu podataka.

JS database.js ×

```
// define and export module for mlab database url
module.exports = {
    url: 'mongodb://Vukan:srbija1@ds159400.mlab.com:59400/booktradingclub'
    // local database url: 'mongodb://localhost:27017/BookTradingClub'
};
```

Slika 11. database.js fajl

2.1 Modeli

MongoDB rukuje jednostavnim JSON dokumentima. Poslovna logika je smeštena u aplikativni sloj. Mongoose ODM nam omogućuje jednostavnu konverziju između dokumenata i JavaScript objekata

Mongoose nam služi za struktuiranje podataka. Ne može se skladištiti ništa što ne odgovara šemi. Šema sluzi kao deklaracija kako treba da izgledaju podaci.
Modeli su prave klase kreirane u JavaScript-i.

U aplikaciji postoje četiri modela i to:

1. user.js - model koji reprezentuje korisnika
2. post.js - model koji reprezentuje objavu na forumu
3. comment.js - model koji reprezentuje komentar o knjizi
4. book.js - model koji reprezentuje knjigu

book.js

```
// importing moongose module and define schema
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

// Book Schema
var Book = new Schema({
    title: {type: String},
    price: {type: Number},
    genre: {type: String},
    author: {type: String},
    publishDate: {type: Date},
    user: {type: Schema.Types.ObjectId, ref: 'User'},
    about: {type: String},
    solded: {type: Boolean},
    userBuyerId: {type: String},
    username: {type: String},
    image: {type: String, default:
        "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTitaeh4NFdB9ev1-t8zG4a71jJgce64lkvEHLCFSHaqWpT1bA"}
},
{
    usePushEach: true
});

// export Book model
module.exports = mongoose.model('Book', Book);
```

Slika 12. model (šema) knjige

JS comment.js

```
// importing moongose module and define schema
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

// Comment Schema
var Comment = new Schema({
    content: {type: String},
    publishDate: {type: Date},
    username: {type: String},
    book: {type: Schema.Types.ObjectId, ref: 'Book'},
    grade: {type: Number},
    tag: {type: String}
},
{
    usePushEach: true
});

// export Comment model
module.exports = mongoose.model('Comment', Comment);
```

Slika 13. model (šema) komentara o knjizi

JS post.js

```
// importing moongose module and define schema
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

// Post Schema
var Post = new Schema({
    title: {type: String, required: true},
    content: {type: String, required: true},
    user: {type: Schema.Types.ObjectId, ref: 'User'},
    username: { type: String }
},
{
    usePushEach: true
});

// export Post model
module.exports = mongoose.model('Post', Post);
```

Slika 14. model (šema) objave na forumu

Slika 15. Model (šema) registrovanog korisnika sajta

2.2 Rute

Rutiranje se odnosi na to kako aplikacija odgovra na klijentske zahteve u zavisnotsti od endpoint-a, HTTP zahteva (GET, POST,…), URI-a…

Vrši se parsiranje tela zahteva i preuzimanje parametara (parsiranje se vrši ukoliko je content-type application/JSON ili application/x-www-form-urlencoded).

Upiti ka bazi su zahtevni prema resursima. Zato se koriste kroz Callback-ove. Šalje se zahtev, obrađuju se drugi poslovi, i kada se dobije povratna vrednost upita, izvrsava se callback funkcija. Upiti su uvek u formatu: function(error, result) {...}.

Express framework omogucuje integrisanje middleware-a za odgovor na HTTP zahteve, definisanje ruting tabele radi implementacije raznih akcija u zavisnosti od URL-a i HTTP metoda kao i dinamičko renderovanje HTML strana bazirano na prosleđivanju argumenata templejtu.

Express aplikacije koriste callback funkcije čiji su parametri request i response objekti.

Request objekat predstavlja HTTP zahtev i poseduje propertije za query string, pararametre, body, HTTP hedere… Response objekat predstavlja HTTP odgovor koji Express aplikacija šalje kao odgovor na zahtev.

Pri tome body-parse middleware nam služi za rukovanje JSON, Text, URL enkodovanim i sirovim podacima.

2.2.1 Autentifikacija

Svi zahtevi sa klijenta stižu preko tokena. Korisiti se JWT - JSON web tokens

Svi zahtevi su autentifikovani. Prilikom logovanja token se šalje korisniku sa servera, pri čemu se čuva u LocalStorage na klijentu. U token-u se čuvaju enkriptovane informacije o korisniku. Vreme trajanja tokena je proizvoljno, u aplikaciji je 10800s.

Za svaki zahtev prema bazi, korisnik se verifikuje preko tokena. Ako je token verifikovan, iz njega se izvlače informacije o korisniku I akcije ka bazi vezane su samo za korisnikove podatke.

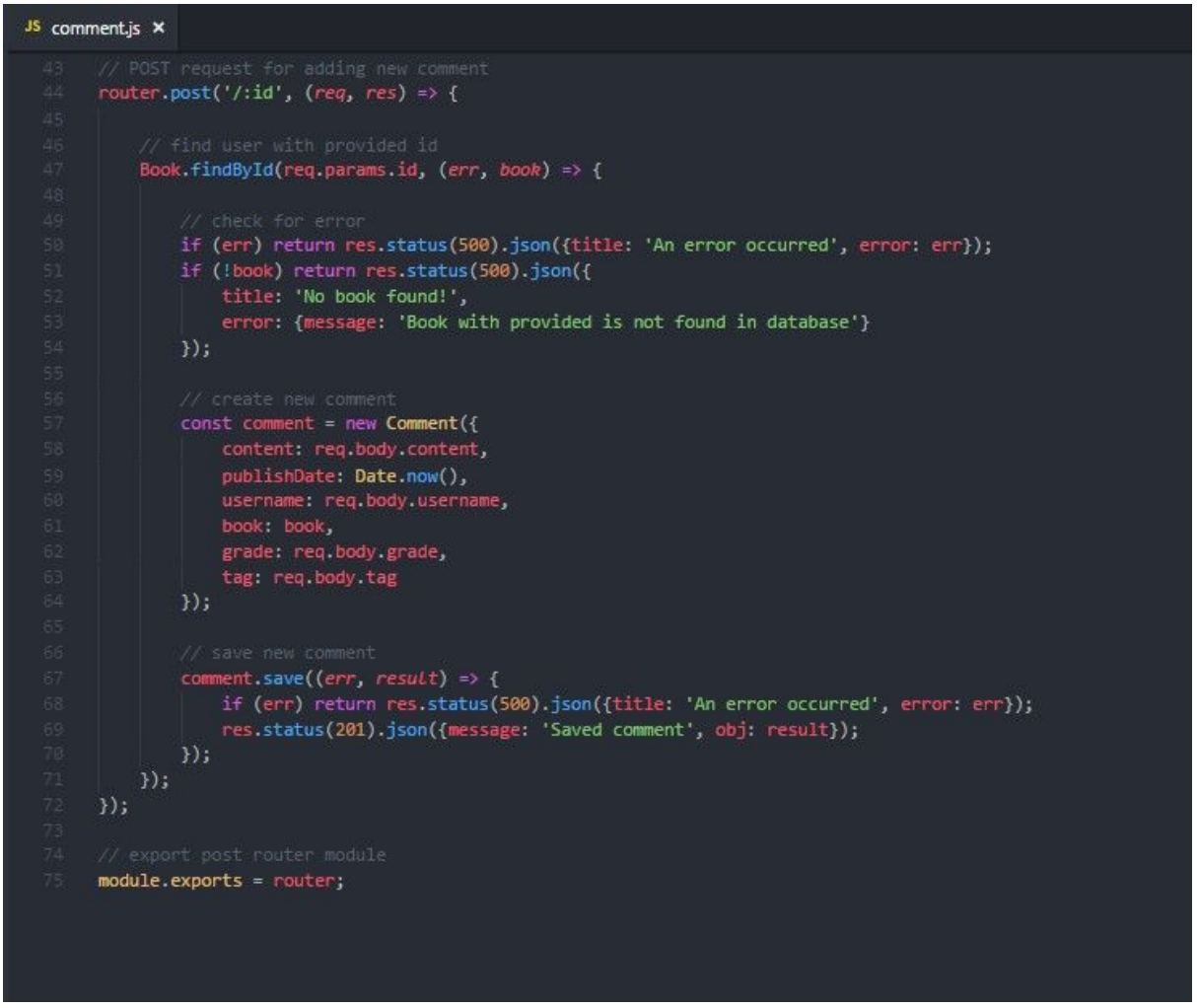
U aplikaciji postoje četiri rute i to:

1. user.js - ruta za sve zahteve vezane za korisnike (registracija, logovanje, promena profilne slike)
2. post.js - ruta za sve zahteve za objave (dodavanje novih objava, njihovo brisanje i ažuriranje od strane administratora)
3. comment.js - ruta koja reprezentuje sve zahteve vezane za komentare komentar o knjigama (dodavanje novog komentara)
4. book.js - ruta koja reprezentuje sve zahteve vezane za knjige (dodavanje knjige, brisanje, ažuriranje, pregled knjige, kao i brisanje i ažuriranje od strane administratora koji može da briše i ažurira sve knjige).

comment.js

```
// import express module
const express = require('express');

// define express router
const router = express.Router();

// import jsonwebtoke for authentication
const jwt = require('jsonwebtoken');

// importing models
const Book = require('../models/book');
const Comment = require('../models/comment');

// GET request for all comments for book
router.get('/:id', (req, res) => {

    // find book with provided id in database
    Book.findById(req.params.id, (err, book) => {
        if (err) return res.status(500).json({
            title: 'An error occurred on finding comment',
            error: err
        });

        // find and return all comments for that book
        Comment.find({book: book}, (err, comments) => {
            if (err) return res.status(500).json({
                title: 'An error occurred on finding comment',
                error: err
            });
            res.status(200).json({message: 'Success on getting book comments', obj: comments});
        });
    });
});

// check if user is logged in for all future routes
router.use('/', (req, res, next) => {
    jwt.verify(req.query.token, 'secret', (err) => {
        if (err) return res.status(401).json({title: 'Not Authenticated', error: err});
        next();
    })
});

```

Slika 16. GET zahtev za sve komentare za određenu knjigu



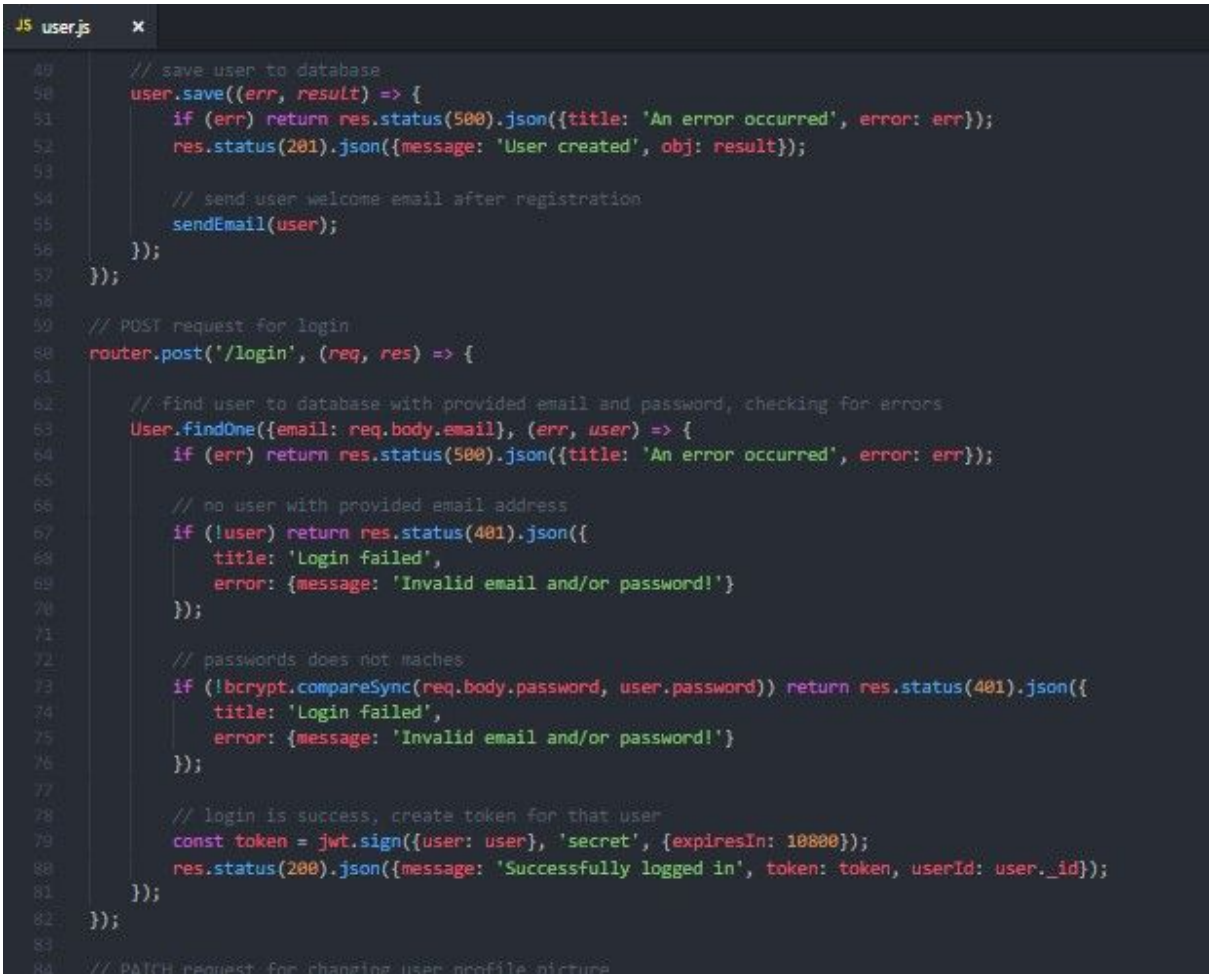
Slika 17. POST zahtev za dodavanje novog komentara o knjizi ili korisniku

JS app.js

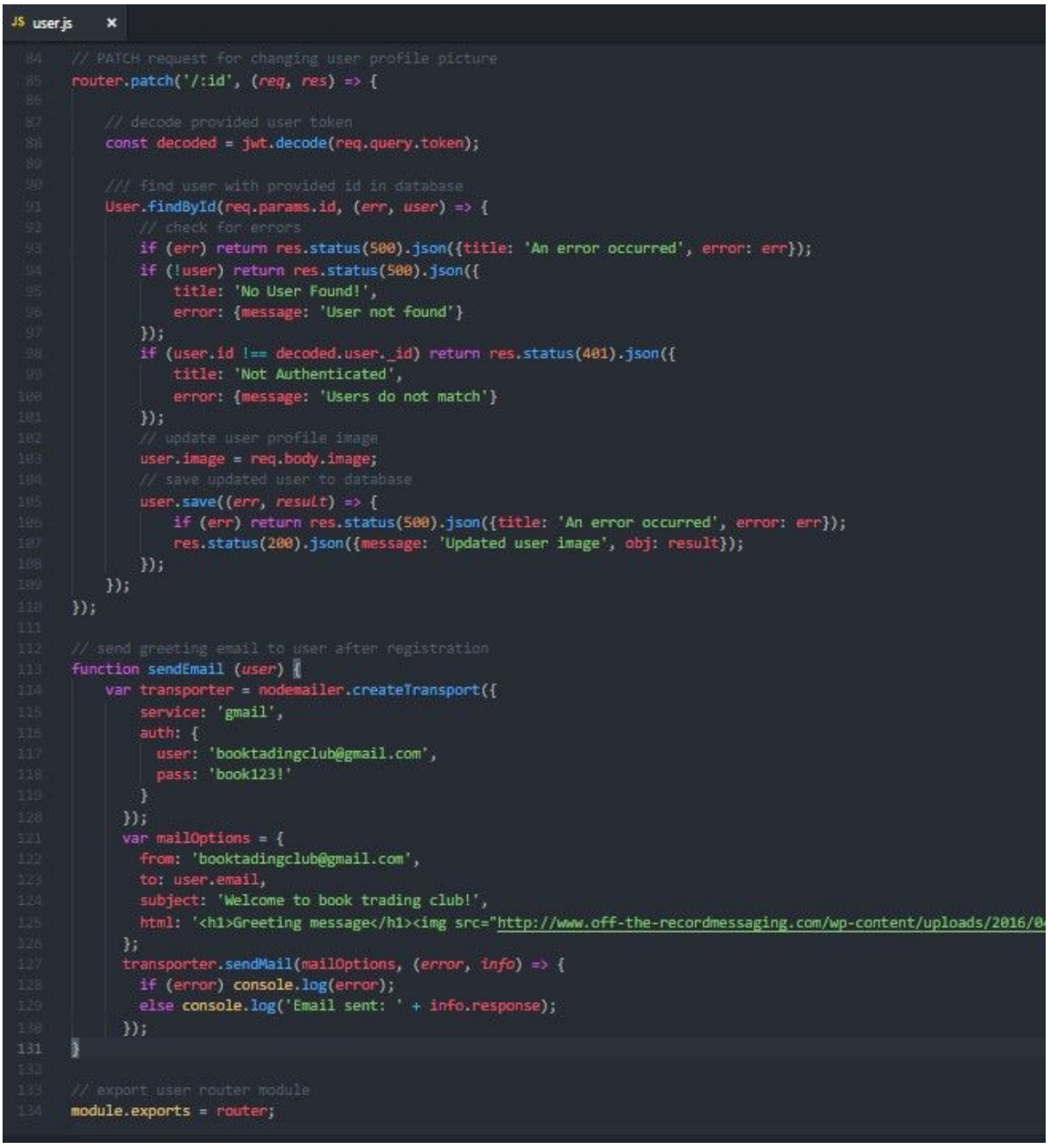
```
// importing express module
const express = require('express');

// define express router
const router = express.Router();

// import book model
const Book = require('../models/book');

// GET request for home page, render index page
router.get('/', (req, res) => {
  res.render('index', { title: 'Book trading club' });
});

// GET request for home page, get and return all books from database
router.get('/home', (req, res) => {
  Book.find({}, ((err, books) => {
    if (err) return res.status(500).json({title: 'An error occurred', error: err});
    res.status(200).json({message: 'Success', obj: books});
  }));
});

// export app router module
module.exports = router;
```

Slika 18. Osnovna ruta home stranice, prikazuju se sve knjige svih korisnika

Slika 19. GET zahtevi za sve korisnike i za pojedinačnog korisnika, kao i POST zahtev za registraciju



Slika 20. POST zahtev za logovanje



Slika 21. PATCH zahtev za ažuriranjem profile slike, prilikom registracije korisniku se šalje mejl dobrodošlice koristeći nodemailer module (https://www.w3schools.com/nodejs/nodejs_email.asp)

post.js

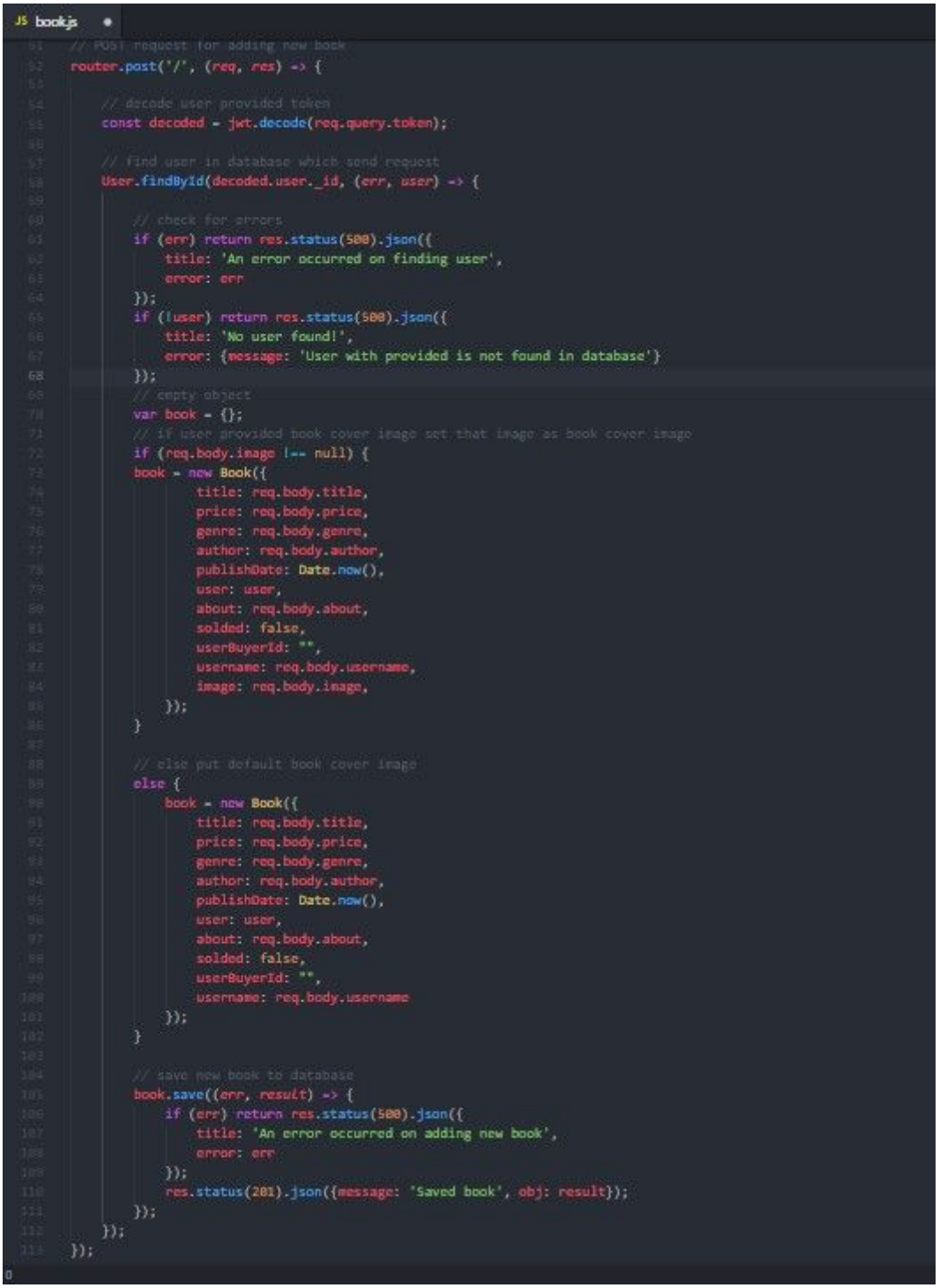
```
// POST request for adding new post
router.post('/', (req, res) => {
    // decode provided user token
    const decoded = jwt.decode(req.query.token);
    // find user with provided id
    User.findById(decoded.user._id, (err, user) => {

        // check for error
        if (err) return res.status(500).json({title: 'An error occurred', error: err});
        // create new post
        const post = new Post({
            title: req.body.title,
            content: req.body.content,
            user: user,
            username: req.body.username
        });

        // save new post to database
        post.save((err, result) => {
            if (err) return res.status(500).json({title: 'An error occurred', error: err});
            res.status(201).json({message: 'Saved post', obj: result});
        });
    });
});

// PATCH request for updating post, posts can be updated only by admin
router.patch('/:id', (req, res) => {
    // find post in database with provided id
    Post.findById(req.params.id, (err, post) => {
        // check for errors
        if (err) return res.status(500).json({
            title: 'An error occurred',
            error: err
        });
        if (!post) return res.status(500).json({
            title: 'No Post Found!',
            error: {message: 'Post not found'}
        });
        // update post
        post.title = req.body.title;
        post.content = req.body.content;
        // save updated post to database
        post.save((err, result) => {
            if (err) return res.status(500).json({
                title: 'An error occurred on updating post',
                error: err
            });
            res.status(200).json({message: 'Updated post', obj: result});
        });
    });
});
```

Slika 22. GET zahtev za sve objave i za pojedinačnu objavu

book.js

```
// import express module
const express = require('express');

// define express router
const router = express.Router();

// import jsonwebtoke for authentication
const jwt = require('jsonwebtoken');

// importing models
const User = require('../models/user');
const Book = require('../models/book');

// GET request for all users books
router.get('/', function (req, res) {
    // find all books for specified user
    Book.find()
        .populate('user', 'firstName')
        .exec((err, books) => {
            if (err) return res.status(500).json({
                title: 'An error occurred on getting user\'s books',
                error: err
            });
            res.status(200).json({message: 'Success on getting user\'s books', obj: books});
    });
});

// GET request for one book
router.get('/:id', function (req, res) {

    // find book in database with provided id
    Book.findById(req.params.id, (err, book) => {

        // check for error
        if (err) return res.status(500).json({
            title: 'An error occurred on getting book',
            error: err
        });
        res.status(200).json({message: 'Success on getting book', obj: book});
    });
});

// verify that user is logged in for all future routes
router.use('/', (req, res, next) => {
    jwt.verify(req.query.token, 'secret', (err) => {
        if (err) return res.status(401).json({title: 'Not Authenticated', error: err});
        next();
    })
});
```

Slika 23. GET zahtevi za knjige koje pripadaju određenom korisniku za i pojedinačnu knjigu



Slika 24. POST zahtev za dodavanje nove knjige

Slika 25. PATCH zahtev za izmenju knjige i DELETE zahtev za brisanje knjige

3. Frontend

Angular 2 je JavaScript Framework za ‘’Single Page’ veb aplikacije.
Koriste se asinhroni zahtevi, šalje se manji broj zahteva ka serveru, što kao za posledicu ima brži odziv UX.

Kod se piše u TypeScript-u - nadskup JavaScript-a kompajliran u cist JavaScript.

index.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <base href="/">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="keywords" content="Book, Trading, Online shop, Forum, Profile, Authentication">
    <meta name="description" content="Web application for book trading club">
    <meta name="author" content="Vukan Markovic">
    <title>Book Trading Club</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" integrity="sha384-WskhaSGF
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootswatch/4.1.1/sketchy/bootstrap.min.css">
    <link href='https://fonts.googleapis.com/css?family=EB+Garamond&subset=latin,latin-ext' rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="/stylesheets/style.css">
</head>
<body>
<my-app><span class="loader"></span></my-app>
<script src="/js/app/bundle.js"></script>
</body>
</html>
```

Slika 26. index.hbs html stranica

index.hbs stranica je kreirana putem Jade template-a i omogućava renderovanje templejta.

style.css

```css
body {
  padding: 2em;
  font-family: 'EB Garamond';
  background-image: url(http://evonyunites.com/wp-content/uploads/2011/09/old-paper2.jpg);
}
a {
  color: #00B7FF;
  cursor: pointer;
}
a:hover, a:active {
  color: #00B7FF;
  text-decoration: underline;
}
a.danger {
  color: red;
}

.spacing {
  margin-bottom: 20px;
}

input.ng-invalid.ng-touched {
  border: 1px solid red;
}

.loader {
  border: 16px solid green;
  border-top: 16px solid greenyellow;
  border-radius: 50%;
  width: 40px;
  height: 40px;
  animation: spin 2s linear infinite;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
@keyframes spin {
  0% {
      transform: rotate(0deg);
  }
  100% {
      transform: rotate(360deg);
  }
}

.content {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translateX(-50%) translateY(-50%);
}
```

Slika 27. style.css fajl

style.css fajl predstavlja glavni stysheet aplikacije, primenjuje se na sve komponente

app.routing.ts

```
import { Routes, RouterModule } from '@angular/router';
import { RegisterComponent } from './components/register/register.component';
import { ProfileComponent } from './components//profile/profile.component';
import { LoginComponent } from './components//login/login.component';
import { LogoutComponent } from './components//logout/logout.component';
import { HomeComponent } from './components//home/home.component';
import { AddBookComponent } from './components//add-book/add-book.component';
import { BlogComponent } from './components//blog/blog.component';
import { PageNotFoundComponent } from './components//page-not-found/page-not-found.component';
import { PublishComponent } from './components/publish/publish.component';
import { BookDetailsComponent } from './components/book-details/book-details.component';
import { UpdateBookComponent } from './components/update-book/update-book.component';
import { PostDetailsComponent } from './components/post-details/post-details.component';
import { AboutComponent } from './components/about/about.component';
import { ChangeProfilePictureComponent } from './components/change-profile-picture/change-profile-picture.component';
import { AddCommentComponent } from './components/add-comment/add-comment.component';
import { UsersComponent } from './components/users/users.component';
import { UpdatePostComponent } from './components/update-post/update-post.component';
import { ShowInfoComponent } from './components/showinfo/show-info.component';

const APP_ROUTES: Routes = [
    { path: '', component: HomeComponent, pathMatch: 'full', data: { title: 'Home' }},
    { path: 'about', component: AboutComponent, data: { title: 'About'} },
    { path: 'add-book', component: AddBookComponent, data: { title: 'Add book'}},
    { path: 'blog', component: BlogComponent, data: { title: 'Blog'}},
    { path: 'users', component: UsersComponent, data: { title: 'Users'}},
    { path: 'book-details/:id', component: BookDetailsComponent, data: { title: 'Book details'}},
    { path: 'login', component: LoginComponent, data: { title: 'Login'}},
    { path: 'logout', component: LogoutComponent, data: { title: 'Logout'} },
    { path: 'post-details/:id', component: PostDetailsComponent, data: { title: 'Post details'}},
    { path: 'profile', component: ProfileComponent, data: { title: 'Profile' }},
    { path: 'publish', component: PublishComponent, data: { title: 'Publish'}},
    { path: 'register', component: RegisterComponent, data: { title: 'Register'}},
    { path: 'update-book/:id', component: UpdateBookComponent, data: { title: 'Update book'}},
    { path: 'update-post/:id', component: UpdatePostComponent, data: { title: 'Update post'}},
    { path: 'change-profile-picture', component: ChangeProfilePictureComponent, data: { title: 'Change profile picture'}},
    { path: 'add-comment/:id', component: AddCommentComponent, data: { title: 'Add comment'}},
    { path: 'show-info/:id', component: ShowInfoComponent, data: { title: 'Show info'}},
    { path: '**', component: PageNotFoundComponent , data: { title: 'Invalid path'}}
];

export const routing = RouterModule.forRoot(APP_ROUTES);
```

Slika 28. app.routing.ts fajl

U app.routing.ts fajlu je kreiran Angular ruter i definisane su sve moguće putanje (rute) sa odgovarajućim komponentama koje je potrebno učitati.

app.component.ts

```
import { Component } from '@angular/core';
import { BookService } from './services/book.service';
import { PostService } from './services/post.service';
import { CommentService } from './services/comment.service';

@Component({
    selector: 'my-app',
    templateUrl: './app.component.html',
    providers: [BookService, PostService, CommentService]
})

export class AppComponent {}
```

Slika 29. app.component.ts fajl

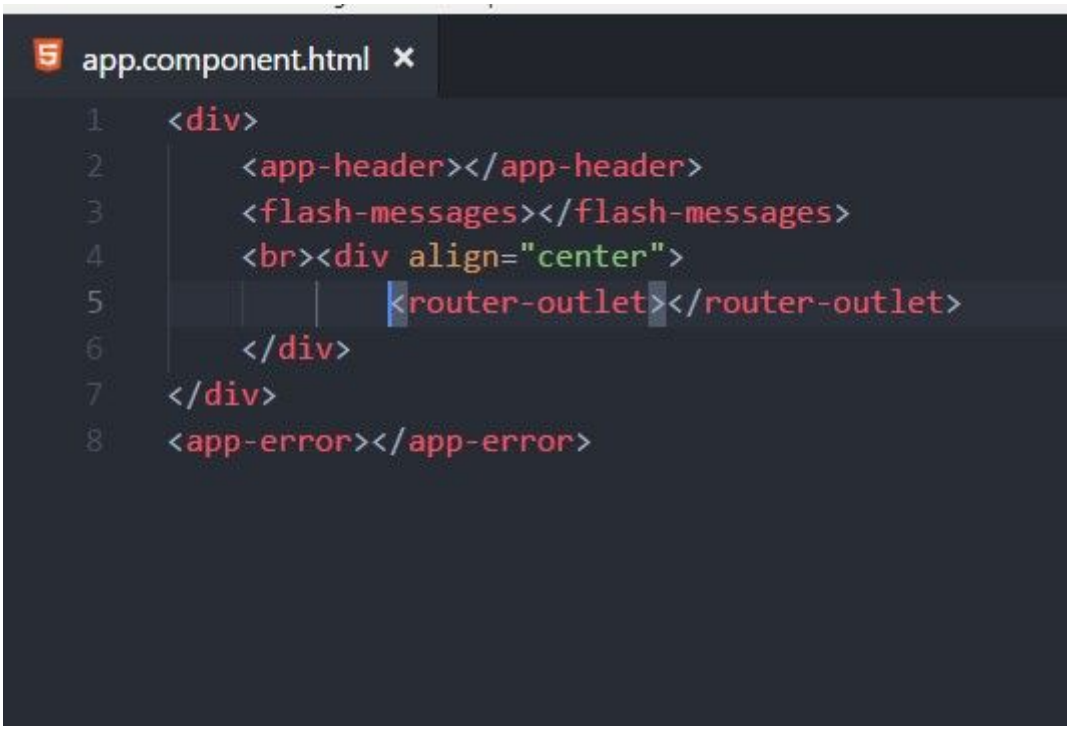
app.component.ts fajl predstavlja glavni typescript fajl. U njemu definisemo servise BookService, PostService i CommentService kako bi bili dostupni svim ostalim komponentama.

app.module.ts

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule, ReactiveFormsModule } from "@angular/forms";
import { HttpModule } from "@angular/http";
import { FlashMessagesModule } from 'angular2-flash-messages';
import { AppComponent } from "./app.component";
import { HeaderComponent } from './components/header/header.component';
import { LoginComponent } from './components/login/login.component';
import { RegisterComponent } from './components/register/register.component';
import { ProfileComponent } from './components/profile/profile.component';
import { AboutComponent } from './components/about/about.component';
import { PageNotFoundComponent } from './components/page-not-found/page-not-found.component';
import { AddBookComponent } from './components/add-book/add-book.component';
import { HomeComponent } from './components/home/home.component';
import { ErrorComponent } from './components/error/error.component';
import { LogoutComponent } from './components/logout/logout.component';
import { BlogComponent } from './components/blog/blog.component';
import { PostComponent } from './components/post/post.component';
import { PostDetailsComponent } from './components/post-details/post-details.component';
import { PublishComponent } from './components/publish/publish.component';
import { BookDetailsComponent } from './components/book-details/book-details.component';
import { UpdateBookComponent } from './components/update-book/update-book.component';
import { BookComponent } from './components/book/book.component';
import { ChangeProfilePictureComponent } from './components/change-profile-picture/change-profile-picture.component';
import { AddCommentComponent } from './components/add-comment/add-comment.component';
import { CommentComponent } from './components/comment/comment.component';
import { UsersComponent } from './components/users/users.component';
import { UpdatePostComponent } from './components/update-post/update-post.component';
import { ShowInfoComponent } from './components/showinfo/show-info.component';
import { AuthService } from './services/auth.service';
import { ErrorService } from './services/error.service';
import { routing } from './app.routing';

@NgModule({
    declarations: [
        AppComponent,
        AboutComponent,
        AddBookComponent,
        BlogComponent,
        BookComponent,
        BookDetailsComponent,
        ErrorComponent,
        HeaderComponent,
        HomeComponent,
        LoginComponent,
        LogoutComponent,
        PageNotFoundComponent,
        PostComponent,
        PostDetailsComponent,
        ProfileComponent,
        PublishComponent,
        RegisterComponent,
        UpdateBookComponent,
        ChangeProfilePictureComponent,
        CommentComponent,
        AddCommentComponent,
        UsersComponent,
        UpdatePostComponent,
        ShowInfoComponent
    ],
    imports: [
        BrowserModule,
        FormsModule,
        ReactiveFormsModule,
        HttpModule,
        routing,
        FlashMessagesModule.forRoot()
    ],
    providers: [AuthService, ErrorService],
    bootstrap: [AppComponent]
})

export class AppModule {}
```

Slika 30. app.module.ts fajl

U app.module.ts fajlu definisemo sve komponente aplikacije, module i service AuthService i ErrorService koji će na taj način biti dostupni na nivou cele aplikacije. Definise se root komponenta unutar AppModule unutar boostrap niza. Ne ucitavaju se sve komponente prilikom pokretanja Angular 2 aplikacije kako se ne bi narušile performanse.



Slika 31. app.component.html fajl

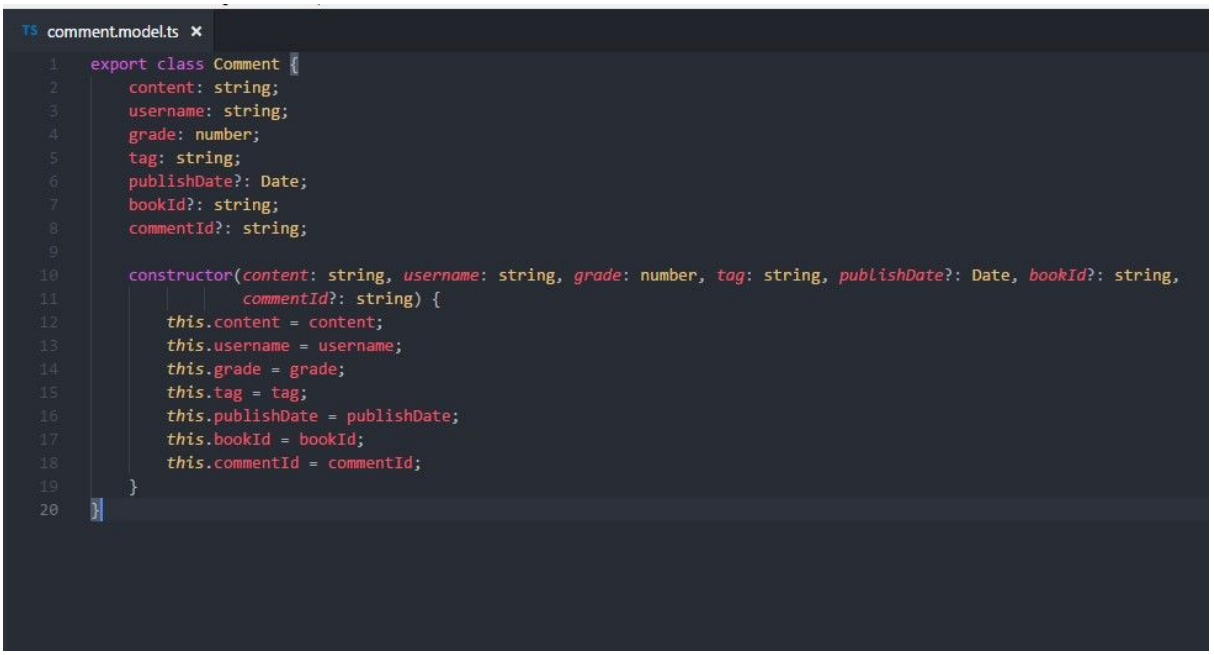
Ovaj fajl će prikazivati renderovane komponente u zavisnosti od rute. Takođe definiše osnovni raspored komponenti.

3.1 Modeli

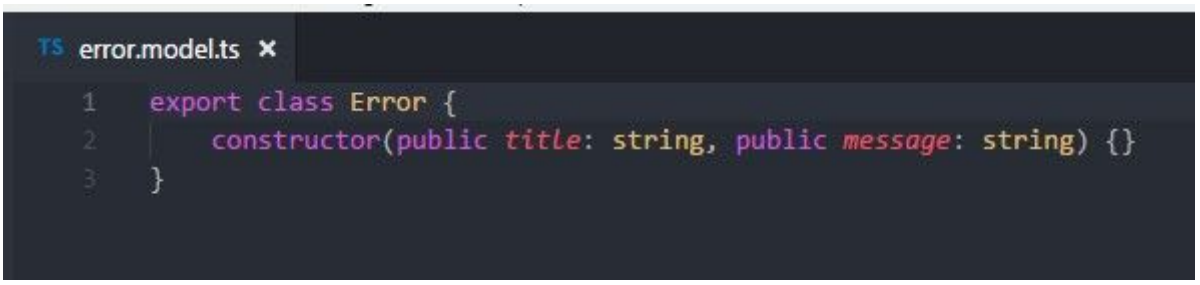
U aplikaciji postoje 5 modela i to:

1. book.model.ts - model koji reprezentuje knjigu
2. user.model.ts - model koji reprezentuje korisnika
3. post.model.ts - model koji reprezentuje objavu
4. comment.model.ts - model koji reprezentuje komentar
5. error.model.ts - model koji reprezentuje grešku

Slika 32. model knjige



Slika 33. model komentara



Slika 34. model greške

post.model.ts

```ts
export class Post {
    title: string;
    content: string;
    username: string;
    userId?: string;
    postId?: string;

    constructor(title: string, content: string, username: string, postId?: string, userId?: string) {
        this.title = title;
        this.content = content;
        this.username = username;
        this.userId = userId;
        this.postId = postId;
    }
}
```

Slika 35. model objave

user.model.ts

```ts
export class User {
    constructor(public email: string, public password: string, public firstName?: string,
                public lastName?: string, public country?: string, public city?: string,
                public postalCode?: string, public address?: string, public phoneNumber?: string,
                public image?: string) {}
}
```

Slika 36. model korisnika

3.2 Servisi

Servisi služe za centralizaciju funkcionalnosti. Smanjuje se dupliciranje koda kroz centralizaciju, prosleđuju se samo instance servisa specijalizovanog za određenu logiku unutar aplikacije. Servisi se često koriste za rukovanje podacima – centralizujemo logiku za rukovanje podacima kroz servis.

3.2.1 HTTP Zahtevi

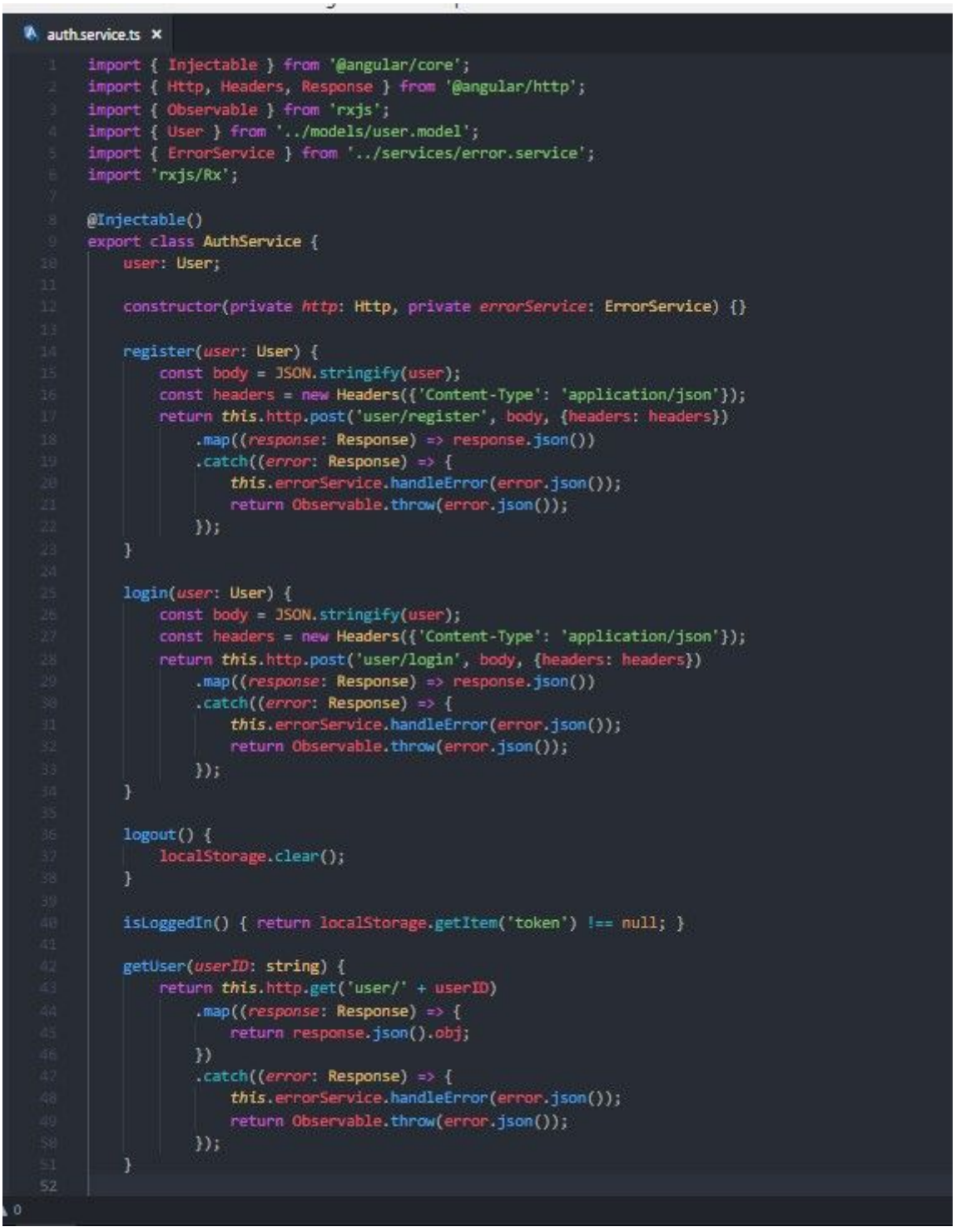
Ne renderuje se serverski nova stranica, podaci se šalju koriz XMLHttpRequest objekat-AJAX zahtevi. Koriste se Observables i Observable patern. Observable služi za asinhrone zahteve, osluškuje različite izvore. Javlja Observer-u kada dođe do promene od interesa. Angular koristi observables u pozadini za zahteve - http.post kreira observable. subscribe na http observable - rukujemo podacima poslanim od Observable-a što je slično kao callbacks ili promises.

Observable HTTP zahtevi – RxJS Svi zahtevi ka backend-u, šalju token kao query parametar.

U aplikaciji postoje pet servisa i to:

1. auth.service.ts - servis za rukovanje autentifikacijom korisnika
2. post.service.ts - servis za rukovanje objavama
3. comment.service.ts - servis za rukovanje komentarima
4. book.service.ts - servis za rukovanje knjigama
5. error.service.ts - zaseban servis i komponenta za rukovanje greskama na nivou aplikacije

Svaki servis poseduje odgovarajuće operacije koje su već navedene prilikom definisanja ruta za korisnika, knjigu, objavu i komentar.



Slika 37. servis korisnika

Metoda logout() vrši odjavljivanje korisnika brisanje njegovih informacija(tokena i id) iz localstorage-a.

Metoda isLoggedIn() vrši proveru da li je korisnik trenutno ulogovan proveravajući da li je token različit od null.

Metoda register() vrši registrovanje korisnika, metoda login prijavljivanje, dok metoda getUser vraća specifičnog korisnika.

auth.service.ts

```
    getUsers() {
        return this.http.get('user/')
            .map((response: Response) => {
                return response.json().obj;
            })
            .catch((error: Response) => {
                this.errorService.handleError(error.json());
                return Observable.throw(error.json());
            });
    }

    changeProfilePicture(user: User) {
        const body = JSON.stringify(user);
        const headers = new Headers({'Content-Type': 'application/json'});
        const token = localStorage.getItem('token') ? '?token=' + localStorage.getItem('token') : '';
        return this.http.patch('user/' + localStorage.getItem('userId') + token, body, {headers: headers})
            .map((response: Response) => response.json())
            .catch((error: Response) => {
                this.errorService.handleError(error.json());
                return Observable.throw(error.json());
            });
    }
}
```

Slika 38. servis korisnika

Metoda getUsers() služi za prikaz svih korisnika administratoru.

Metoda changeProflePicture() služi za promenu profilne slike.

book.service.ts

```
import { Http, Response, Headers } from '@angular/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Book } from '../models/book.model';
import { ErrorService } from '../services/error.service';
import 'rxjs/Rx';

@Injectable()
export class BookService {
    public books: Book[] = [];

    constructor(private http: Http, private errorService: ErrorService) {}

    addBook(book: Book) {
        const body = JSON.stringify(book);
        const headers = new Headers({'Content-Type': 'application/json'});
        const token = localStorage.getItem('token') ? '?token=' + localStorage.getItem('token') : '';
        return this.http.post('book/' + token, body, {headers: headers})
            .map((response: Response) => {
                const result = response.json();
                const book = new Book(
                    result.obj.title,
                    result.obj.price,
                    result.obj.genre,
                    result.obj.author,
                    result.obj.image,
                    result.obj.about,
                    result.obj.username,
                    result.obj.publishDate,
                    result.obj.user._id,
                    result.obj.userBuyerId,
                    result.obj._id,
                    result.obj.solded);
                this.books.push(book);
                return book;
            })
            .catch((error: Response) => {
                console.log(error);
                this.errorService.handleError(error.json());
                return Observable.throw(error.json());
            });
    }

```

Slika 39. servis knjiga

Metoda addBook() služi za dodavanje nove knjige.

book.service.ts

```
getBooks() {
    return this.http.get('book/')
        .map((response: Response) => {
            const books = response.json().obj;
            let transformedBooks: Book[] = [];
            for (let book of books) {
                transformedBooks.push(new Book(
                    book.title,
                    book.price,
                    book.genre,
                    book.author,
                    book.image,
                    book.about,
                    book.username,
                    book.publishDate,
                    book.user._id,
                    book.userBuyerId,
                    book._id,
                    book.solded)
                );
            }
            this.books = transformedBooks;
            return transformedBooks;
        })
        .catch((error: Response) => {
            this.errorService.handleError(error.json());
            return Observable.throw(error.json());
        });
}
```

Slika 40. servis knjiga

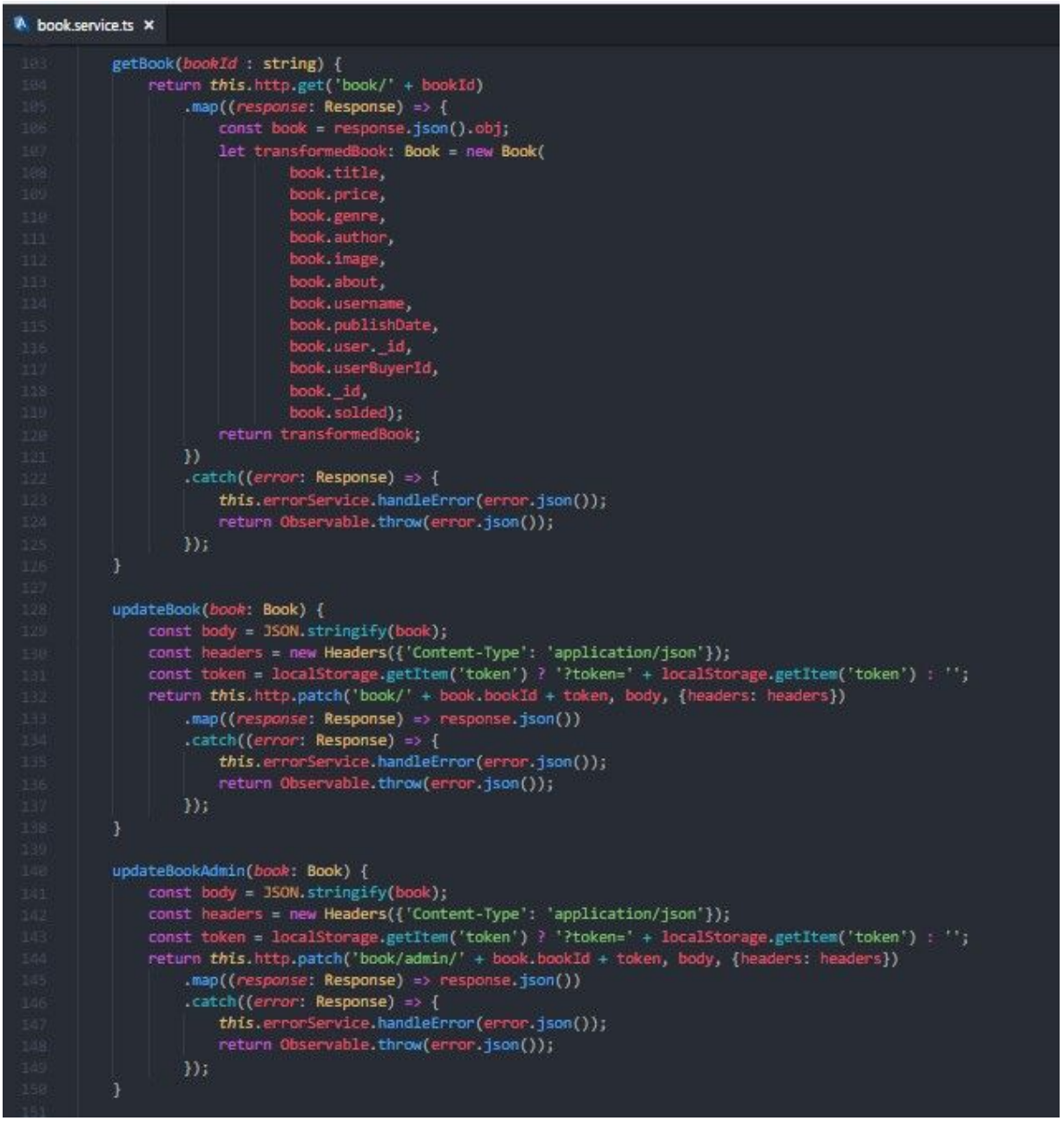
Metoda getBooks() služi za prikaz svih korisnikovih knjiga.

book.service.ts

```
getBooksHome() {
    return this.http.get('home/')
        .map((response: Response) => {
            const books = response.json().obj;
            let transformedBooks: Book[] = [];
            for (let book of books) {
                transformedBooks.push(new Book(
                    book.title,
                    book.price,
                    book.genre,
                    book.author,
                    book.image,
                    book.about,
                    book.username,
                    book.publishDate,
                    book.user._id,
                    book.userBuyerId,
                    book._id,
                    book.solded));
            }
            this.books = transformedBooks;
            return transformedBooks;
        })
        .catch((error: Response) => {
            this.errorService.handleError(error.json());
            return Observable.throw(error.json());
        });
}
```

Slika 41. servis knjiga

Metoda getBooksHome() služi za prikaz svih knjiga svih korinika na početno strani.

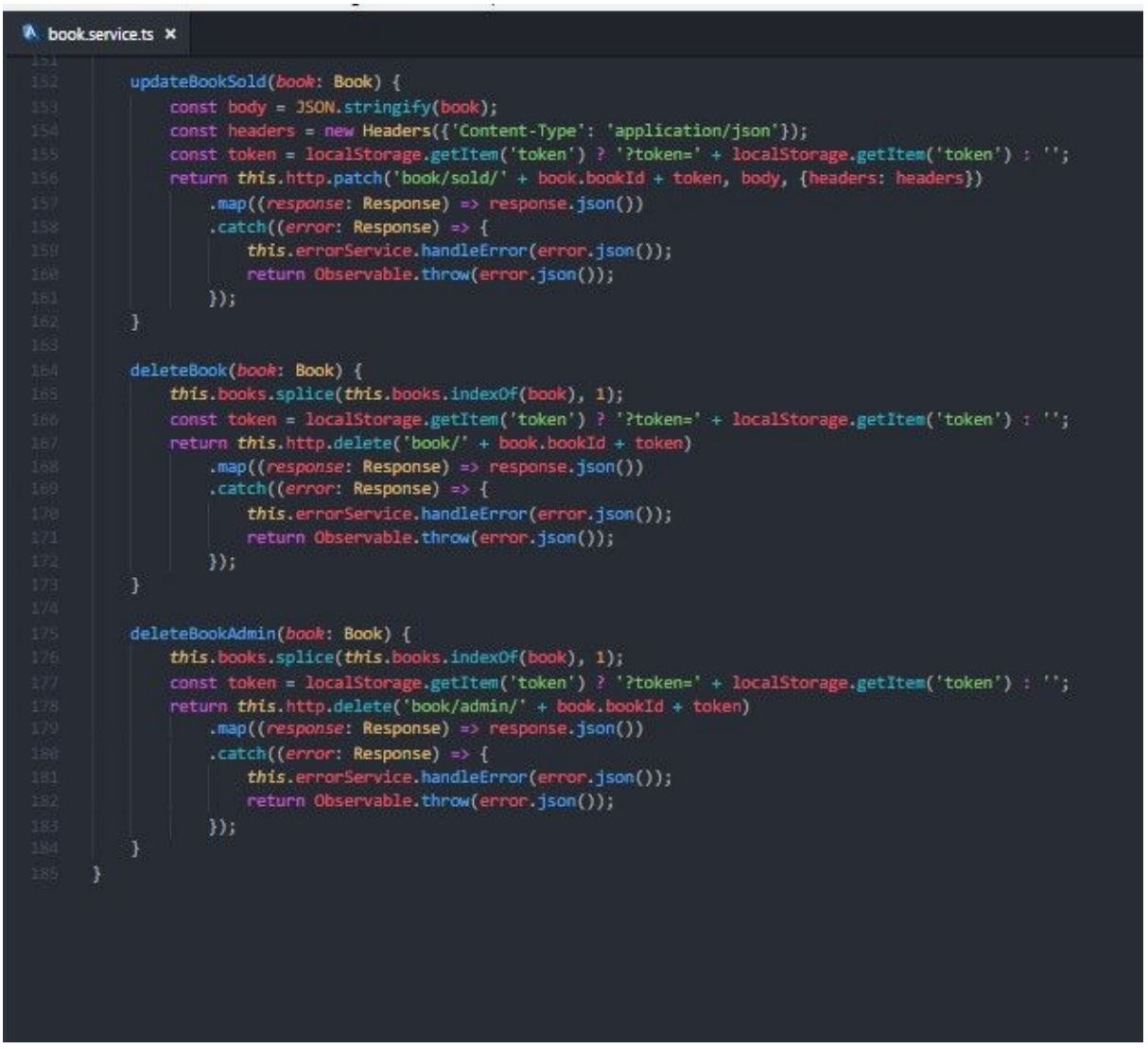


Slika 42. servis knjiga

Metoda getBook() služi za prikaz određene knjige.

Metoda updateBook() služi za ažuriranje knjige od strane korisnika. Korisnik može da ažurira samo svoje knjige ukoliko je registrovan.

Metoda updateBookAdmin() služi za ažuriranje knjige od strane admina. Admin može da ažurira sve knjige.



Slika 43. servis knjiga

Metoda updateSold() služi za ažuriranje knjige prilikom njene kupovine od strane nekog korisnika.

Metoda deleteBook() služi za brisanje knjige od strane nekog korisnika. Svaki korisnik može da briše jedino svoje knjige (knjige koje je on postavio).

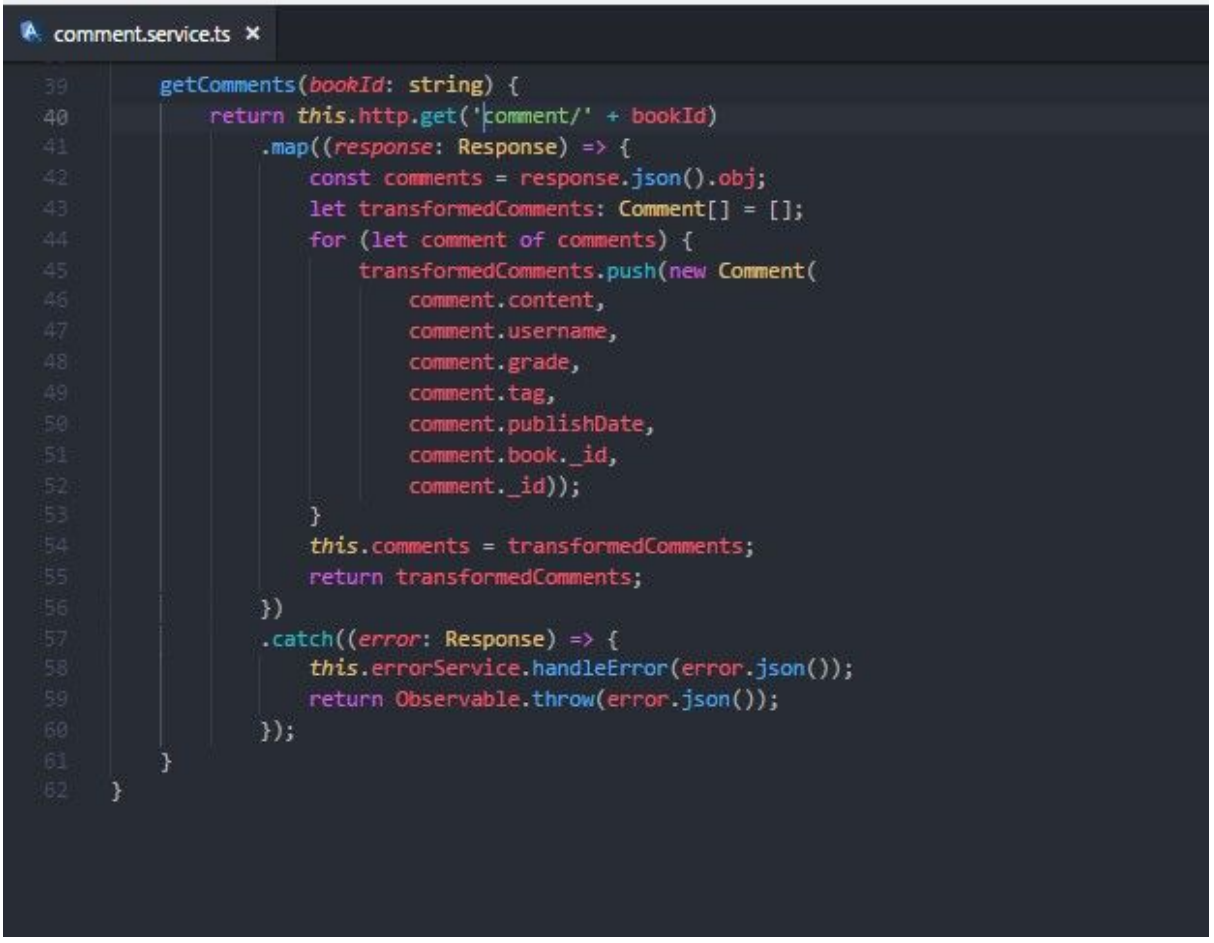
Metoda deleteBookAdmin() služi za brisanje knjige od strane administratora. Administrator može da briše knjige svih korisnika.

comment.service.ts

```
import { Http, Response, Headers } from '@angular/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Comment } from '../models/comment.model';
import { ErrorService } from '../services/error.service';
import 'rxjs/Rx';

@Injectable()
export class CommentService {
    private comments: Comment[] = [];
    comment: Comment;

    constructor(private http: Http, private errorService: ErrorService) {}

    addComment(comment: Comment, bookId: string) {
        const body = JSON.stringify(comment);
        const headers = new Headers({'Content-Type': 'application/json'});
        const token = localStorage.getItem('token') ? '?token=' + localStorage.getItem('token') : '';
        return this.http.post('comment/' + bookId + token, body, {headers: headers})
            .map((response: Response) => {
                const result = response.json();
                const comment = new Comment(
                    result.obj.content,
                    result.obj.username,
                    result.obj.grade,
                    result.obj.tag,
                    result.obj.publishDate,
                    result.obj.book._id,
                    result.obj._id);
                this.comments.push(comment);
                return comment;
            })
            .catch((error: Response) => {
                this.errorService.handleError(error.json());
                return Observable.throw(error.json());
            });
    }
```

44. servis komentara

Metoda addComment() služi za dodavanje komentara na neku knjigu, o knjizi ili korisniku koji ju je postavio. Samo registrovani korisnici mogu komentarisati.



Slika 45. servis komentara

Metoda getComments() služi za prikaz svih komentara neke knjige.

error.service.ts

```typescript
import { EventEmitter, Injectable } from '@angular/core';
import { Error } from '../models/error.model';

@Injectable()
export class ErrorService {
    errorOccurred = new EventEmitter<Error>();

    handleError(error: any) {
        const errorData = new Error(error.title, error.error.message);
        this.errorOccurred.emit(errorData);
    }
}
```

Slika 46. servis za rukovanje greškama

dodavanje novog Slika 47. Slika 47. servis Slika

post.service.ts

```
import { Http, Response, Headers } from '@angular/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import 'rxjs/Rx';
import { Post } from '../models/post.model';
import { ErrorService } from '../services/error.service';

@Injectable()
export class PostService {
    private posts: Post[] = [];
    post: Post;

    constructor(private http: Http, private errorService: ErrorService) {}

    addPost(post: Post) {
        const body = JSON.stringify(post);
        const headers = new Headers({'Content-Type': 'application/json'});
        const token = localStorage.getItem('token') ? '?token=' + localStorage.getItem('token') : '';
        return this.http.post('post/' + token, body, {headers: headers})
            .map((response: Response) => {
                const result = response.json();
                const post = new Post(
                    result.obj.title,
                    result.obj.content,
                    result.obj.username,
                    result.obj._id,
                    result.obj.user._id);
                this.posts.push(post);
                return post;
            })
            .catch((error: Response) => {
                this.errorService.handleError(error.json());
                return Observable.throw(error.json());
            });
    }
```

Slika 47. servis objava

Metoda addPost() služi za objavljivanje novih postova na blogu. Samo registrovani korisnici mogu objavljivati na blogu.

Slika 48. servis objava

Metoda getPosts() služi za prika svih objava na forumu.

Metoda getPost() služi za prikaz jedne određene objave.

Metoda deletePost() služi za brisanje postova. Samo administrator može brisati postove (i to sve).

Slika 49. servis objava

Metoda updatePost() služi za izmenu objava na forumu. Samo administrator sistema može vršiti izmenu objava(i to svih) ukoliko uoči neki neprikladan sadržaj.

3.3 Komponente

Aplikacija se sastoji od ukupno 23 komponente i te komponente su sledeće:

- about - služi za prikaz stranice sa nekim osnovnim informacijama o sajtu
- add-book - reprezentuje formu za dodavanje nove knjige
- add-comment - reprezentuje formu za dodavanje novog komentara
- blog - služi za prikaz stranice za blog
- book - gradivna komponenta koja reprezentuje knjigu
- book-details - služi za prikaz stranice sa detaljima o odabranom knjizi
- change-profile-picture - služi za prikaz stranice koja omogućuje promenu profilne slike
- comment - gradivna komponenta koja reprezentuje komentar
- error - komponenta koja je namenjena za obradu grešaka
- header - služi za prikaz zaglavlja stranice
- home - služi za prikaz naslovne strana sajta
- login - reprezentuje formu za logovanje
- logout - služi za odjavljivanje sa sajta
- page-not-found - komponenta koja se prikazuje u slučaju pogrešno URL-a
- post - gradivna komponenta koja reprezentuje objavu
- post-details - služi za prikaz pojedinačne objave
- profile - služi za prikaz profilne stanice registrovanog korisnika
- publish - služi za objavljivanje novih postova
- register - reprezentuje formu za registraciju
- showinfo - služi za prikaz informacija o kupcu
- update-book - reprezentuje formu za ažuriranje knjige
- update-post - reprezentuje formu za ažuriranje posta
- users - služi za prikaz svih registrovanih korisnika (ova mogućnost je dostupna samo adminu)

U nastavku je dat prikaz svih navedenih komponenti koje se sastoje iz tri dela - htm, css i ts fajl.

about.component.html

```
<div id="about" class="content text-center">
<div class="row">
<div class="col-sm-12">
<img width="35%" height="45%" src="http://blog.onclickinnovations.com/wp-content/uploads/2016/12/mean.png">
<hr>
<p>
The application is an example of a MEAN 2 web application with demonstrated CRUD operations and
authentication. It was developed for the needs of a student project and the purchase and sale of books
is not real.
</p>
</div>
</div>
<hr>
<div class="row">
      <div class="col-sm-4">
          Made by Vukan Markovic.<br>
          Code released under the MIT License.
      </div>
      <div class="col-sm-4">
          <br>
          Book Trading Club App&copy;
      </div>
      <div class="col-sm-4">
          <br>
          <img width="40%" height="60%" src="https://cdn1.iconfinder.com/data/icons/simple-icons/256/github-256-black.png">
          <a href="https://github.com/Vukan-Markovic/Book-trading-club" target="blank" title="Github" style="text-decoration: none">
            See code here</a>
        </div>
  </div>
</div>
```

Slika 50. about.component.hmtl

about.component.ts

```typescript
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})

export class AboutComponent implements OnInit {
  constructor() {}
  ngOnInit() {}
}
```

Slika 51. about.component.ts

add-book.component.html

```html
<form id="f" (ngSubmit)="onSubmit()" #f="ngForm">
  <fieldset>
    <legend>Add a new book</legend>
    <
      <legend>
      The legend element represents a caption for the rest of the contents of the legend element's parent fieldset element, if any.
    <div class="form-group">
      <label for="price">Price:</label>
      <input id="price" name="price" [ngModel]="book?.price" class="form-control" placeholder="Enter the price of the book" type="number" required min="0">
    </div>
    <div class="form-group">
      <label for="genres">Genre:</label>
      <select id="genre" name="genre" [ngModel]="book?.genre" class="form-control" required>
        <option selected>Arts &amp;and; Photography</option>
        <option>Biographies &amp;and; Memoirs</option>
        <option>Business &amp;and; Investing</option>
        <option>Children's Books</option>
        <option>History</option>
        <option>Literature &amp;and; Suspense</option>
        <option>Romance</option>
        <option>Sci-Fi &amp;and; Fantasy</option>
        <option>Teens &amp;and; Young Adult</option>
      </select>
    </div>
    <div class="form-group">
        <label for="author">Author:</label>
        <input name="author"  id="author" [ngModel]="book?.author" class="form-control" placeholder="Enter the author of the book" type="text" required>
    </div>
    <div class="form-group">
      <label for="about">Write a couple of words about the book if you like:</label>
      <textarea id="about" name="about" [ngModel]="book?.about" class="form-control" rows="3" placeholder="Your comment about book" maxlength="80"></textarea>
    </div>
    <div class="form-group">
      <label for="author">Image:</label>
      <input id="image" name="image" [ngModel]="book?.image" class="form-control" placeholder="Enter the image url of the book cover" type="url">
      <a href="https://www.google.rs/search?tbm=isch&q={{f.value.title}}+book+cover" target="_blank">Get book cover</a>
    </div>
    <button type="button" class="btn btn-lg btn-danger" (click)="onClear()">Clear</button>
    <button class="btn btn-lg btn-success" type="submit" [disabled]="!form.valid">Add a book</button>
    </fieldset>
</form>
```

Slika 52. add-book.component.html

add-book.component.ts

```ts
import { Component, OnInit, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { BookService } from '../../services/book.service';
import { Book } from '../../models/book.model';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';
import { FlashMessagesService } from 'angular2-flash-messages';

@Component({
  selector: 'app-add-book',
  templateUrl: './add-book.component.html',
  styleUrls: ['./add-book.component.css']
})

export class AddBookComponent implements OnInit {
  @ViewChild('f') form : NgForm;
  book: Book;
  user: any = {};

  constructor(private bookService: BookService, private router: Router,
              private authService: AuthService, private flashMessage: FlashMessagesService) {}

  ngOnInit() {
    if (!this.authService.isLoggedIn())
      return this.router.navigate(['/login']);
    this.authService.getUser(localStorage.getItem('userId'))
      .subscribe((user: User) => {
          this.user = user;
      }
    );
  }

  onSubmit() {
    const book = new Book(this.form.value.title, this.form.value.price, this.form.value.genre, this.form.value.author, this.form.value.image, this.form.value.about, this.user.firstName);

    this.bookService.addBook(book)
      .subscribe(
        data => {
          console.log(data),
          this.flashMessage.show('New book is added!', { cssClass: 'alert-success', timeout: 3000 });
        },
        error => console.error(error));
    this.form.resetForm();
    this.router.navigate(['/profile']);
  }

  onClear() {
      this.book = null;
      this.form.resetForm();
  }
}
```

Slika 53. add-book.component.ts

add-comment.component.html

```
<form id="f" (ngSubmit)="onSubmit()" #f="ngForm">
  <fieldset>
    <legend>Add a new comment</legend>
    <div class="form-group">
      <label for="author">This comment is about:</label>
      <select [ngModel]="comment?.tag" class="form-control" name="tag" id="tag" required>
        <option selected>Book</option>
        <option>User</option>
      </select>
    </div>
    <div class="form-group">
      <label for="content">Content:</label>
      <input id="content" name="content" [ngModel]="comment?.content" class="form-control" placeholder="Enter the content of the comment" type="text" required
      max="40">
    </div>
    <div class="form-group">
      <label for="grade">Grade:</label>
      <input id="grade" name="grade" [ngModel]="comment?.grade" class="form-control" placeholder="Enter the grade" type="number" required min="0" max="10">
    </div>
    <button type="button" class="btn btn-lg btn-danger" (click)="onClear()">Clear</button>
    <button class="btn btn-lg btn-success" type="submit" [disabled]="!form.valid">Add a comment</button>
  </fieldset>
</form>
```

Slika 54. add-comment.component.html

add-comment.component.ts

```ts
import { Component, OnInit, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { CommentService } from '../../services/comment.service';
import { Router, ActivatedRoute } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';
import { Comment } from '../../models/comment.model';
import { FlashMessagesService } from 'angular2-flash-messages';

@Component({
  selector: 'app-add-comment',
  templateUrl: './add-comment.component.html',
  styleUrls: ['./add-comment.component.css']
})

export class AddCommentComponent implements OnInit {
  @ViewChild('f') form : NgForm;
  comment: Comment;
  user: any = {};
  book: any = {}

  constructor(private commentService: CommentService, private router: Router,
              private authService: AuthService, private flashMessage: FlashMessagesService,
              private route: ActivatedRoute) {}


  ngOnInit() {
    if (!this.authService.isLoggedIn())
      return this.router.navigate(['/login']);

    this.authService.getUser(localStorage.getItem('userId'))
      .subscribe((user: User) => {
          this.user = user;
      }
    );
  }

  onSubmit() {
    const comment = new Comment(
      this.form.value.content,
      this.user.firstName,
      this.form.value.grade,
      this.form.value.tag);

    this.commentService.addComment(comment, this.route.snapshot.params['id'])
      .subscribe(
        data => {
          console.log(data),
          this.flashMessage.show('New comment is added!', { cssClass: 'alert-success', timeout: 3000 });
        },
        error => console.error(error));

    this.form.resetForm();
    this.router.navigate(['/book-details/', this.route.snapshot.params['id']]);
  }

  onClear() {
      this.comment = null;
      this.form.resetForm();
  }
}
```

Slika 55. add-comment.component.ts

blog.component.html

```
<div class="jumbotron">
  <h1 class="display-3">Welcome to our forum!</h1>
  <p class="lead">Here people interested in literature exchange their opinions</p>
  <hr class="my-4">
  <div class="row">
    <div class="col-sm-4"></div>
    <div *ngIf="!isLoggedIn()" class="col-sm-4">
      <p>If you want to join us in discussion you need to be registered!</p>
      <button (click)="onRegister()" class="btn btn-primary btn-lg">Register</button>
    </div>
    <div *ngIf="isLoggedIn()" class="col-sm-4">
        <p>Share your experiences with us! Publish a new post and take part at our forum</p>
        <button (click)="onPost()" class="btn btn-primary btn-lg">Post</button>
    </div>
    <div class="col-sm-4"></div>
  </div>
</div>
<br>
<br>
<div class="row">
  <div class="col-sm-12">
    <app-post [post]="post" *ngFor="let post of posts" (click)="postDetails(post)"></app-post>
  </div>
</div>
```

Slika 56.blog.component.html

blog.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { PostService } from '../../services/post.service';
import { Post } from '../../models/post.model';

@Component({
  selector: 'app-blog',
  templateUrl: './blog.component.html'
})

export class BlogComponent implements OnInit {
  posts: Post [] = [];

  constructor(private router: Router, private authService: AuthService, private postService: PostService) { }

  ngOnInit() {
    this.postService.getPosts()
      .subscribe((posts: Post[]) => {
          this.posts = posts;
        }
      );
  }

  onPost() {
    this.router.navigate(['/publish']);
  }

  onRegister() {
    this.router.navigate(['/register']);
  }

  postDetails(post: Post) {
    this.router.navigate(['/post-details/', post.postId]);
  }

  isLoggedIn () {
    return this.authService.isLoggedIn();
  }
}
```

Slika 58. blog.component.ts

book.component.html

```html
<div id="book" (click)="bookDetails()" *ngIf="belongsToUser()">
    <span class="badge badge-success">{{ book.title }}</span>
    <span class="badge badge-info">{{ book.price | currency: "usd": 3 }}</span>
    <span class="badge badge-success">{{ book.author }}</span>
    <span class="badge badge-danger">{{ book.genre }}</span>
    <span class="badge badge-info">{{ book.publishDate | date:"d/M/y" }}</span>
    <div class="btn-group">
          <button class="btn btn-warning" (click)="onUpdate(book)">Update</button>
        <button class="btn btn-danger" (click)="onDelete()">Delete</button>
        <button (click)="showInfo()" class="btn btn-info" *ngIf='isBookSolded()'>Solded</button>
    </div>
</div>
```

Slika 59. book.component.html

book.component.ts

```typescript
import { Component, OnInit, Input } from '@angular/core';
import { Book } from '../../models/book.model';
import { BookService } from '../../services/book.service';
import { Router} from '@angular/router';
import { FlashMessagesService } from 'angular2-flash-messages';
import { AuthService } from '../../services/auth.service';

@Component({
  selector: 'app-book',
  templateUrl: './book.component.html',
  styleUrls: ['./book.component.css'],
})

export class BookComponent implements OnInit{
    @Input() book: Book;

    constructor(private bookService: BookService, private router: Router, private flashMessage: FlashMessagesService, private authService: AuthService) {}

    ngOnInit(){}

    onUpdate(book : Book) {
        this.router.navigate(['/update-book', book.bookId]);
    }

    onDelete() {
        this.bookService.deleteBook(this.book)
            .subscribe(
                result => {
                    console.log(result)
                    this.flashMessage.show('Book is deleted!', { cssClass: 'alert-danger', timeout: 3000 });
                },
                error => console.error(error));
    }

    belongsToUser() {
        if (this.authService.isLoggedIn())
            return localStorage.getItem('userId') === this.book.userId;
        return false;
    }

    isBookSolded() {
        return this.book.solded;
    }

    bookDetails() {
        this.router.navigate(['/book-details/', this.book.bookId]);
    }

    showInfo() {
        this.router.navigate(['/show-info/', this.book.userBuyerId]);
    }
}
```

Slika 60. book.component.ts

book-details.component.html

```
<article>
<div class="card mb-3">
        <h3 class="card-header"><span id="text">Title:</span> {{ book.title | uppercase }} &reg;</h3>
        <div class="card-body">
          <h5 class="card-title"><span id="text">Author:</span> {{ book.author }}</h5>
          <h6 class="card-subtitle text-muted"><span id="text">Price:</span> {{ book.price | currency: "usd": 3 }} </h6>
        </div>
        <img style="height: 200px; width: 12%; display: block;" src= "{{ book.image }}" alt="{{ book.title }}" title="{{ book.title }}">
        <div class="card-body">
          <p class="card-text">
            <span id="text">Genre:</span> {{ book.genre }}<br>
            <span id="text" *ngIf="isDefined()">About:</span> {{ book.about }}<br>
            <span id="text">User:</span> {{ book.username }}
          </p>
        </div>
        <div class="card-body">
            <button *ngIf="!belongsToUser()" class="card-link btn btn-lg btn-success" (click)="onBuy()">Buy</button>
            <button *ngIf="isAdmin()" class="card-link btn btn-lg btn-warning" (click)="onUpdate()">Update</button>
            <button *ngIf="isAdmin()" class="card-link btn btn-lg btn-danger" (click)="onDelete()">Delete</button>
            <button *ngIf="!belongsToUser()" class="card-link btn btn-lg btn-danger" (click)="onComment()">Add comment</button>
          </div>
        <div class="card-footer text-muted">
          <span id="text">Published:</span> {{ book.publishDate | date:"d/M/y" }}
                        {{ book.publishDate | date:"shortTime" }}
        </div>
</div>
</article>
<div class="row">
  <div class="col-sm-12">
    <br><h2>Comments:</h2>
    <app-comment [comment]="comment" *ngFor="let comment of comments"></app-comment>
  </div>
</div>
```

Slika 61. book-details.component.html

book-details.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Book } from '../../models/book.model';
import { Comment } from '../../models/comment.model';
import { BookService } from '../../services/book.service';
import { AuthService } from '../../services/auth.service';
import { Router, ActivatedRoute } from '@angular/router';
import { FlashMessagesService } from 'angular2-flash-messages';
import { CommentService } from '../../services/comment.service';
import { User } from '../../models/user.model';

@Component({
  selector: 'app-book-details',
  templateUrl: './book-details.component.html',
  styleUrls: ['./book-details.component.css'],
})
export class BookDetailsComponent implements OnInit {
    book: any = {};
    user: any = {};
    comments: Comment[] = [];

    constructor(private bookService: BookService, private authService: AuthService,
                private router: Router, private flashMessage: FlashMessagesService,
                private route: ActivatedRoute, private commentService: CommentService) {}

    ngOnInit () {
        this.bookService.getBook(this.route.snapshot.params['id'])
            .subscribe(
                (book: Book) => {
                    this.book = book;
                }
            );

        this.commentService.getComments(this.route.snapshot.params['id'])
            .subscribe((comments: Comment[]) => {
                this.comments= comments;
            }
        );

        if(this.authService.isLoggedIn()) {
            this.authService.getUser(localStorage.getItem('userId'))
                .subscribe((user: User) => {
                        this.user = user;
                    }
                );
        }
    }

    onBuy () {
        this.book.userBuyerId = localStorage.getItem('userId');

        this.bookService.updateBookSold(this.book)
            .subscribe(result => console.log(result), error => console.error(error));

        this.router.navigate(['/']);
        this.flashMessage.show('Book is bought!', { cssClass: 'alert-success', timeout: 3000 });
    }

    belongsToUser () {
        if(this.authService.isLoggedIn())
            return this.user.firstName === this.book.username;
        return true;
    }
```

book-details.component.ts

```
    isAdmin () {
        if(this.authService.isLoggedIn())
            return this.user.email === 'vukan.markovic97@gmail.com';
        return false;
    }

    onDelete () {
        this.bookService.deleteBookAdmin(this.book)
            .subscribe(
                result => {
                    console.log(result)
                    this.flashMessage.show('Book is deleted!', { cssClass: 'alert-danger', timeout: 3000 });
                    this.router.navigate(['/']);
                },
                error => console.error(error));
    }

    onComment () {
        this.router.navigate(['/add-comment/', this.book.bookId]);
    }

    onUpdate () {
        this.router.navigate(['/update-book', this.book.bookId]);
    }

    isDefined () {
        return this.book.about !== null;
    }
}
```

Slika 62. book-details.component.ts

change-profile-picture.component.html

```
<form id="f" (ngSubmit)="onSubmit()" #f="ngForm">
  <fieldset>
    <legend>Update a profile picture</legend>
    <div class="form-group">
      <label for="title">Picture:</label>
      <input [ngModel]="user?.image" class="form-control" id="image" name="image" placeholder="Enter the url of the image" type="url" required>
      <br><br><button  class="btn btn-lg btn-warning" type="submit">Change</button>
    </div>
    </fieldset>
</form>
```

Slika 63. change-profile-picture.component.html

change-profile-picture.component.ts

```ts
import { Component, OnInit, ViewChild } from '@angular/core';;
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';
import { FlashMessagesService } from 'angular2-flash-messages';
import { NgForm } from '@angular/forms';

@Component({
  selector: 'app-change-profile-picture',
  templateUrl: './change-profile-picture.component.html',
  styleUrls: ['./change-profile-picture.component.css']
})

export class ChangeProfilePictureComponent implements OnInit {
    user: any = {};
    @ViewChild('f') form: NgForm;

    constructor(private router: Router, private authService: AuthService,
                private flashMessage: FlashMessagesService) {}

    ngOnInit () {
        if (this.authService.isLoggedIn())
        {
            this.authService.getUser(localStorage.getItem('userId'))
                .subscribe((user: User) => {
                    this.user = user;
                }
            );
        }
        else this.router.navigate(['/login']);
    }

    onSubmit () {
        this.user.image = this.form.value.image;
        this.authService.changeProfilePicture(this.user)
                .subscribe((user: User) => {
                        this.user = user;
                        this.router.navigate(['/profile']);
                        this.flashMessage.show('Profile picture changed', { cssClass: 'alert-warning', timeout: 3000 });
                    }
                );
    }
}
```

Slika 64.change-profile-picture.component.ts

comment.component.html

```html
<div id="comment" class="card text-white mb-3 flex-container bg-warning">
    <div class="card-header flex-item"><span class="badge badge-success">{{ comment.tag }}</span></div>
    <div class="card-body flex-item">
      <h4 class="card-title flex-item">{{ comment.content }} </h4>
      <p class="card-text flex-item">User: {{ comment.username }}</p>
      <p class="card-text flex-item">Grade: {{ comment.grade }}</p>
      <p class="card-text flex-item">Published: {{ comment.publishDate | date }}</p>
    </div>
</div>
```

Slika 65. comment.component.html

comment.component.ts

```ts
import { Component, OnInit, Input } from '@angular/core';
import { Comment } from '../../models/comment.model';

@Component({
  selector: 'app-comment',
  templateUrl: './comment.component.html',
  styleUrls: ['./comment.component.css'],
})

export class CommentComponent implements OnInit{
    @Input() comment: Comment;
    constructor() {}
    ngOnInit(){}
}
```

Slika 66. comment.component.ts

error.component.html

```html
<div class="backdrop" [ngStyle]="{'display': display}"></div>
<div class="modal" tabindex="-1" role="dialog" [ngStyle]="{'display': display}">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" aria-label="Close" (click)="onErrorHandled()"><span aria-hidden="true">
                    &times;</span></button>
                <h4 class="modal-title">{{ error?.title }}</h4>
            </div>
            <div class="modal-body">
                <p>{{ error?.message }}</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" (click)="onErrorHandled()">Close</button>
            </div>
        </div>
    </div>
</div>
```

Slika 67. error.component.html

error.component.ts

```typescript
import { Component, OnInit } from '@angular/core';
import { Error } from '../../models/error.model';
import { ErrorService } from '../../services/error.service';

@Component({
    selector: 'app-error',
    templateUrl: './error.component.html',
    styleUrls: ['./error.component.css']
})

export class ErrorComponent implements OnInit {
    error: Error;
    display = 'none';

    constructor(private errorService: ErrorService) {}

    onErrorHandled() {
        this.display = 'none';
    }

    ngOnInit() {
        this.errorService.errorOccurred
            .subscribe(
                (error: Error) => {
                    this.error = error;
                    this.display = 'block';
                }
            );
    }
}
```

Slika 68. error.component.ts

header.component.html

```
<header>
  <nav class="col-md-12 navbar navbar-expand-lg navbar-dark bg-dark nav-default">
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" ariacontrols="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span></button>
    <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="nav nav-tabs mr-auto">
          <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}"><a class="nav-link" [routerLink]="['/']">Home</a></li>
          <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}" *ngIf="isLoggedIn()"><a class="nav-link" [routerLink]="['/profile']">Profile</a></li>
          <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}"><a class="nav-link" [routerLink]="['/blog']">Blog</a></li>
          <li *ngIf="isAdmin()" role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}"><a class="nav-link" [routerLink]="['/users']">Users</a></li>
        </ul>
      <ul class="nav nav-tabs">
          <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}" *ngIf="!isLoggedIn()"><a class="nav-link" [routerLink]="['/register']">Register</a></li>
          <li role="presentation" routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}" *ngIf="!isLoggedIn()"><a class="nav-link" [routerLink]="['/login']">Login</a></li>
          <li role="presentation" routerLinkActive="active"  [routerLinkActiveOptions]="{exact:true}" *ngIf="isLoggedIn()"><a class="nav-link" [routerLink]="['/logout']">Logout</a></li>
          <li role="presentation" routerLinkActive="active"  [routerLinkActiveOptions]="{exact:true}"><a class="nav-link" [routerLink]="['/about']">About</a></li>
        </ul>
    </div>
  </nav>
</header>
```

Slika 69. header.component.html

header.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../../services/auth.service';
import { Book } from '../../models/book.model';
import { User } from '../../models/user.model';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html'
})

export class HeaderComponent implements OnInit {

  constructor(private authService: AuthService) {}

  user: any = {};
  book: Book;

  ngOnInit() {
    if (this.authService.isLoggedIn()) {
      this.authService.getUser(localStorage.getItem('userId'))
        .subscribe((user: User) => {
            this.user = user;
        }
        );
    }
  }

  isLoggedIn() {
    return this.authService.isLoggedIn();
  }

  isAdmin() {
    if (this.authService.isLoggedIn())
        return this.user.email === 'vukan.markovic97@gmail.com';
    return false;
  }
}
```

Slika 70. header.component.ts

home.component.html

```
<div class="jumbotron">
  <div class="text-center">
    <h1>Online Book Store</h1>
    <p>Buy or sell your favorite book</p>
  </div>
</div>
<div class="row">
<div class="col-md-12">
<div *ngFor="let book of books" (click)="bookDetails(book)">
  <div id="book" *ngIf="!book.solded" class="card text-white mb-3 flex-container bg-info">
    <div class="card-header flex-item">{{ book.title }}</div>
    <div class="card-body flex-item">
      <h4 class="card-title flex-item">{{ book.price | currency : "usd" : 3 }}</h4>
      <p class="card-text flex-item"><img style="height: 200px; width: 40%; display: block;" src= "{{ book.image }}" alt="Book cover"
        title="{{ book.title }}"></p>
    </div>
  </div>
</div>
</div>
</div>
```

Slika 71. home.component.html

home.component.ts

```
import { Component, OnInit } from '@angular/core';
import { BookService } from '../../services/book.service';
import { Book } from '../../models/book.model';
import { Router } from '@angular/router';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css'],
})

export class HomeComponent implements OnInit {
  public books: Book[] = [];

  constructor(private bookService: BookService, private router: Router) {}

  ngOnInit () {
    this.bookService.getBooksHome()
      .subscribe((books: Book[]) => {
          this.books = books;
        }
      );
  }

  bookDetails(book: Book) {
    this.router.navigate(['/book-details/', book.bookId]);
  }
}
```

Slika 72. home.component.ts

login.component.html

```
<div class="col-md-3 col-md-offset-2 content">
    <form [formGroup]="form" (ngSubmit)="onSubmit()" class="form-signin">
        <h2 class="form-signin-heading">Login</h2>
        <div class="form-group">
            <label for="email">Email</label>
            <input name="email" type="email" id="email" class="form-control" formControlName="email" required autocomplete="email">
        </div>
        <div class="form-group">
            <label for="password">Password</label>
            <input type="password" id="password" class="form-control" formControlName="password" required
            autocomplete="current-password">
        </div>
        <button class="btn btn-success btn-lg" type="submit" [disabled]="!form.valid">Submit</button>
    </form>
    <br>Not registered. Click <span (click)="register()">here</span> to join us!
</div>
```

Slika 73. login.component.html

login.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { User } from '../../models/user.model';
import { AuthService } from '../../services/auth.service';
import { FlashMessagesService } from 'angular2-flash-messages';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})

export class LoginComponent implements OnInit {
    form: FormGroup;

    constructor(private authService: AuthService, private router: Router, private flashMessage: FlashMessagesService) {}

    ngOnInit() {
        this.form = new FormGroup({
            email: new FormControl(null, [
                Validators.required,
                Validators.pattern("[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?")
            ]),
            password: new FormControl(null, Validators.required)
        });
    }


    onSubmit() {
        const user = new User(this.form.value.email, this.form.value.password);
        this.authService.login(user)
            .subscribe(
                data => {
                    localStorage.setItem('token', data.token);
                    localStorage.setItem('userId', data.userId);
                    this.flashMessage.show('Logged in!', { cssClass: 'alert-success', timeout: 3000 });
                    this.router.navigateByUrl('/');
                },
                error => {
                    this.flashMessage.show('Invalid email or/and password!', { cssClass: 'alert-danger', timeout: 3000 });
                    console.error(error)
                }
            );
        this.form.reset();
    }

    register() {
        this.router.navigate(['/register']);
    }
}
```

Slika 74. login.component.ts

logout.component.html

```html
<div class="col-md-8 col-md-offset-2 content">
    <button class="btn btn-lg btn-danger" (click)="onLogout()">Logout</button>
</div>
```

Slika 75. logout.component.html

logout.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { FlashMessagesService } from 'angular2-flash-messages';

@Component({
  selector: 'app-logout',
  templateUrl: './logout.component.html'
})
export class LogoutComponent implements OnInit {

  constructor(private authService: AuthService, private router: Router, private flashMessage: FlashMessagesService) { }

  ngOnInit() {
    if (!this.authService.isLoggedIn())
      this.router.navigate(['/login']);
  }

  onLogout() {
    this.authService.logout();
    this.flashMessage.show('Logged out!', { cssClass: 'alert-danger', timeout: 3000 });
    this.router.navigate(['/']);
  }
}
```

Slika 76. logout.component.ts

page-not-found.component.html

```html
<div class="content">
  <div align="center">
    <img src="https://juststickers.in/wp-content/uploads/2016/12/404-error-not-found.png">
  </div>
</div>
```

Slika.77. page-not-found.component.html

page-not-found.component.ts

```ts
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-page-not-found',
  templateUrl: './page-not-found.component.html'
})

export class PageNotFoundComponent implements OnInit {
  constructor() { }
  ngOnInit() {}
}
```

Slika 78. page-not-found.component.ts

post.component.html

```
<div id="post" class="card text-white mb-3 flex-container bg-success">
    <div class="card-header flex-item">{{ post.title }}</div>
    <div class="card-body flex-item">
      <h4 class="card-title flex-item">By: {{ post.username }}</h4>
      <p class="card-text flex-item"></p>
    </div>
</div>
```

Slika 79. post.component.html

post.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Post } from '../../models/post.model';

@Component({
  selector: 'app-post',
  templateUrl: './post.component.html',
  styleUrls: ['./post.component.css'],
})

export class PostComponent implements OnInit{
    @Input() post: Post;
    constructor() {}
    ngOnInit(){}
}
```

Slika 80. post.component.ts

post-details.component.html

```
<article>
  <div class="card mb-3 bg-danger">
    <h3 class="card-header"><span id="text">Title:</span> {{ post.title | uppercase }}</h3>
    <div class="card-body">
      <h5 class="card-title"><span id="text">By:</span> {{ post.username }}</h5>
      <h6 class="card-subtitle text-muted"></h6>
    </div>
    <div class="card-body">
      <p class="card-text"></p>
    </div>
    <div class="card-body">
      {{ post.content }}
    </div>
    <div class="card-footer text-muted">
      <button *ngIf="isAdmin()" class="card-link btn btn-lg btn-success" (click)="onDelete()">Delete</button>
      <button *ngIf="isAdmin()" class="card-link btn btn-lg btn-success" (click)="onUpdate()">Update</button>
    </div>
  </div>
</article>
```

Slika 81. post-details.component.html

post-details.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Post } from '../../models/post.model';
import { PostService } from '../../services/post.service';
import { ActivatedRoute, Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { FlashMessagesService } from 'angular2-flash-messages';
import { User } from '../../models/user.model';

@Component({
  selector: 'app-post-details',
  templateUrl: './post-details.component.html',
  styleUrls: ['./post-details.component.css'],
})

export class PostDetailsComponent implements OnInit{
    post: any = {};
    user: any = {};

    constructor(private postService: PostService, private route: ActivatedRoute,
                private authService: AuthService, private flashMessage: FlashMessagesService,
                private router: Router) {}

    ngOnInit () {
        this.postService.getPost(this.route.snapshot.params['id'])
            .subscribe((post: Post) => {
                    this.post = post;
                }
            );

        if(this.authService.isLoggedIn()) {
            this.authService.getUser(localStorage.getItem('userId'))
                .subscribe((user: User) => {
                        this.user = user;
                    }
                );
        }
    }

    isAdmin() {
        if(this.authService.isLoggedIn())
            return this.user.email === 'vukan.markovic97@gmail.com';
        return false;
    }

    onDelete() {
        this.postService.deletePost(this.post)
            .subscribe(
                result => {
                    console.log(result);
                    this.router.navigate(['/blog']);
                    this.flashMessage.show('Post is deleted!', { cssClass: 'alert-danger', timeout: 3000 });
                },
                error => console.error(error));
    }

    onUpdate() {
        this.router.navigate(['/update-post', this.post.postId]);
    }
}
```

Slika 82. post-details.component.ts

profile.component.html

```
<div class="row">
  <div class="col-sm-12">
    <div class="jumbotron">
      <img id="profilePicture" width="40%" height="60%" src="{{ user.image }}" alt="Profile picture" title="Profile picture">
      <br><br><button id="changeProfilePicture" class="btn btn-info btn-sm btn-block" (click)=changeProfilePicture()>Change profile picture</button>
      <br><br><div><span class="text-info">{{ user.firstName }}</span></div>
      <div><span class="text-danger">{{ user.lastName }}</span></div>
      <div><span class="text-success">{{ user.email }}</span></div>
    </div>
    <br><button id="btnAddBook" class="btn btn-primary btn-lg btn-block" (click)=addBook()>Add a book for sale</button>
    <button id="btnAddBook" class="btn btn-default btn-lg btn-block" (click)=buyBook()>Buy book</button>
  </div>
</div>
<div class="row">
  <div class="col-sm-12">
    <br><h2>Books:</h2>
    <app-book [book]="book" *ngFor="let book of books"></app-book>
  </div>
</div>
```

Slika 83. profil.component.html

profile.component.ts

```ts
import { Component, OnInit} from '@angular/core';
import { Book } from '../../models/book.model';
import { BookService } from '../../services/book.service';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';

@Component({
  selector: 'app-profile',
  templateUrl: './profile.component.html',
  styleUrls: ['./profile.component.css']
})

export class ProfileComponent implements OnInit {
    books: Book[];
    user: any = {};

    constructor(private bookService: BookService, private router: Router,
                private authService: AuthService) {}

    ngOnInit () {
        if (this.authService.isLoggedIn())
        {
            this.authService.getUser(localStorage.getItem('userId'))
                .subscribe((user: User) => {
                    this.user = user;
                }
            );

            this.bookService.getBooks()
                .subscribe((books: Book[]) => {
                    this.books = books;
                }
            );
        }
        else this.router.navigate(['/login']);
    }

    addBook () {
        this.router.navigate(['/add-book']);
    }

    buyBook () {
        this.router.navigate(['/']);
    }

    changeProfilePicture () {
        this.router.navigate(['/change-profile-picture']);
    }
}
```

Slika 84. profil.component.ts

publish.component.html

```html
<form (ngSubmit)="onSubmit()" #f="ngForm">
  <fieldset>
  <legend>Publish a new post</legend>
  <div class="form-group">
    <label for="title">Title of the post:</label>
    <input type="text" id="title" name="title" class="form-control" [ngModel]="post?.title" rows="14" required>
  </div>
  <div class="form-group">
      <label for="content">Shares with us something interesting</label>
    <textarea id="content" name="content" class="form-control" [ngModel]="post?.content" rows="14" required></textarea>
  </div>
  <p id="warning" class="form-text text-muted">We want just to notify you that this is forum about literature and books and all non-topic posts will be deleted!</p>
  <button class="btn btn-lg btn-success" type="submit" [disabled]="!form.valid">Post</button>
  <button type="button" class="btn btn-danger btn-lg" (click)="onClear()">Clear</button>
  </fieldset>
</form>
```

Slika 85. publish.component.html

publish.component.ts

```typescript
import { Component, OnInit, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { PostService } from '../../services/post.service';
import { Post } from '../../models/post.model';
import { AuthService } from '../../services/auth.service';
import { Router } from '@angular/router';
import { User } from '../../models/user.model';

@Component({
    selector: 'app-publish',
    templateUrl: './publish.component.html'
})

export class PublishComponent implements OnInit {
    @ViewChild('f') form : NgForm;
    post: Post;
    user: any = {};

    constructor(private postService: PostService, private authService: AuthService, private router: Router) {}

    ngOnInit() {
        if (!this.authService.isLoggedIn())
            this.router.navigate(['/login']);
        this.authService.getUser(localStorage.getItem('userId'))
            .subscribe((user: User) => {
                    this.user = user;
                }
            );
    }

    onSubmit() {
        const post = new Post(this.form.value.title, this.form.value.content, this.user.firstName);
        this.postService.addPost(post)
            .subscribe(data => console.log(data), error => console.error(error));
        this.form.resetForm();
        this.router.navigate(['/blog']);
    }

    onClear() {
        this.post = null;
        this.form.resetForm();
    }
}
```

Slika 86. publish.component.ts

register.component.html

```
<div class="col-md-4 col-md-offset-2 content">
    <form [formGroup]="form" (ngSubmit)="onSubmit()">
        <h2>Register</h2>
        <div class="form-group">
            <label for="firstName">First Name</label>
            <input type="text" id="firstName" class="form-control" formControlName="firstName" autocomplete="name">
        </div>
        <div class="form-group">
            <label for="lastName">Last Name</label>
            <input type="text" id="lastName" class="form-control" formControlName="lastName" autocomplete="family-name">
        </div>
        <div class="form-group">
            <label for="email">Email</label>
            <input type="email" id="email" class="form-control" formControlName="email" autocomplete="email">
        </div>
        <div class="form-group">
            <label for="password">Password</label>
            <input type="password" id="password" class="form-control" formControlName="password" autocomplete="current-password">
        </div>
        <div class="form-group">
            <label for="country">Country</label>
            <input type="text" id="country" class="form-control" formControlName="country" autocomplete="country">
        </div>
        <div class="form-group">
            <label for="city">City</label>
            <input type="text" id="city" class="form-control" formControlName="city">
        </div>
        <div class="form-group">
            <label for="postalCode">Postal code</label>
            <input type="text" id="postalCode" class="form-control" formControlName="postalCode" autocomplete="postal-code">
        </div>
        <div class="form-group">
            <label for="address">Address</label>
            <input type="text" id="address" class="form-control" formControlName="address" autocomplete="address-level2">
        </div>
        <div class="form-group">
            <label for="phoneNumber">Phone number</label>
            <input type="tel" id="phoneNumber" class="form-control" formControlName="phoneNumber" autocomplete="tel">
        </div>
        <button class="btn btn-lg btn-info" type="submit" [disabled]="!form.valid">Submit</button>
        <br><br>Already have account. <span (click)="login()">Log in.</span>
    </form>
</div>
```

Slika 87. register.component.html

register.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';
import { FlashMessagesService } from 'angular2-flash-messages';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})

export class RegisterComponent implements OnInit {
  form: FormGroup;
  re: any;

  constructor(private authService: AuthService, private router: Router, private flashMessage: FlashMessagesService) {}

  ngOnInit() {
    this.re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
    this.form = new FormGroup({
        firstName: new FormControl(null, [
          Validators.required,
          Validators.pattern("^(?=.{2,20}$)(?![_.])(?!.*[_.]{2})[a-zA-Z0-9._]+(?<![_.])$")
        ]),
        lastName: new FormControl(null, [
            Validators.required,
            Validators.pattern("^(?=.{4,20}$)(?![_.])(?!.*[_.]{2})[a-zA-Z0-9._]+(?<![_.])$")
        ]),
        email: new FormControl(null, [
            Validators.required,
            Validators.pattern(this.re)
        ]),
        password: new FormControl(null, Validators.required),
        country: new FormControl(null, Validators.required),
        city: new FormControl(null, Validators.required),
        postalCode: new FormControl(null, Validators.required),
        address: new FormControl(null, Validators.required),
        phoneNumber: new FormControl(null, Validators.required)
    });
  }

  onSubmit() {
    const user = new User(
        this.form.value.email,
        this.form.value.password,
        this.form.value.firstName,
        this.form.value.lastName,
        this.form.value.country,
        this.form.value.city,
        this.form.value.postalCode,
        this.form.value.address,
        this.form.value.phoneNumber
    );

    this.authService.register(user)
        .subscribe(data => {
                console.log(data)
                this.flashMessage.show('Registered!', { cssClass: 'alert-success', timeout: 3000 });
                this.router.navigate(['/login']);
              },
            error => console.error(error));

    this.form.reset();
  }

  login () {
      this.router.navigate(['/login']);
  }
}
```

Slika 88. register.component.ts

show-info.component.html

```
<article>
  <div class="card mb-3 bg-danger">
    <h3 class="card-header"><span id="text">User: </span> {{ user.firstName }} {{user.lastName}}</h3>
      <div class="card-body">
        <h5 class="card-title"></h5>
        <h6 class="card-subtitle text-muted"></h6>
      </div>
    <div class="card-body">
      <p class="card-text">
        <img id="profilePicture" width="40%" height="60%" src="{{ user.image }}" alt="Profile picture" title="Profile picture">
      </p>
    </div>
    <div class="card-body">
    </div>
    <div class="card-footer text-muted">
      {{ user.country }}  
      {{ user.city }}  
      {{ user.postalCode }}  
      {{ user.address }}  
      {{ user.phoneNumber }}
    </div>
  </div>
</article>
```

Slika 89. show-info.component.html

show-info.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-show-info',
  templateUrl: './show-info.component.html'
})

export class ShowInfoComponent implements OnInit {
  public user: any = {};

  constructor(private authService: AuthService, private route: ActivatedRoute) {}

  ngOnInit () {
    if(this.authService.isLoggedIn()) {
    this.authService.getUser(this.route.snapshot.params['id'])
      .subscribe((user: User) => {
          this.user = user;
       }
      );
    }
  }
}
```

Slika 90. show-info.component.ts

```
m id="updateForm" (ngSubmit)="onSubmit()" #f="ngForm">
ieldset>
<legend>Update a book</legend>
<div class="form-group">
  <label for="title">Title:</label>
  <input [ngModel]="book?.title" class="form-control" id="title" name="title" placeholder="Enter the title of the book" type="text" required>
</div>
<div class="form-group">
  <label for="price">Price:</label>
  <input [ngModel]="book?.price" class="form-control" id="price" name="price" placeholder="Enter the price of the book" type="number" required>
</div>
<div class="form-group">
  <label for="genres">Genre:</label>
  <select [ngModel]="book?.genre" class="form-control" name="genre" id="genres" required>
    <option>Arts &amp;and; Photography</option>
    <option>Biographies &amp;and; Memoirs</option>
    <option>Business &amp;and; Investing</option>
    <option>Children's Books</option>
    <option>History</option>
    <option>Literature &amp;and; Suspense</option>
    <option>Romance</option>
    <option>Sci-Fi &amp;and; Fantasy</option>
    <option>Teens &amp;and; Young Adult</option>
  </select>
</div>
<div class="form-group">
    <label for="author">Author:</label>
    <input [ngModel]="book?.author" name="author" class="form-control" id="author" placeholder="Enter the author of the book" type="text" required>
</div>
<div class="form-group">
  <label for="about">Write a couple of words about the book if you like:</label>
  <textarea [ngModel]="book?.about" name="about" class="form-control" id="about" rows="3" placeholder="Your comment about book"></textarea>
</div>
<div class="form-group">
  <label for="author">Image:</label>
  <input [ngModel]="book?.image" name="image" class="form-control" id="image" placeholder="Enter the image url of the book cover" type="text">
div>
<button class="btn btn-lg btn-warning" type="submit">Update a book</button>
</fieldset>
rm>
```

Slika 91. update-book.component.html

update-book.component.ts

```
  onSubmit() {
    this.book.title = this.form.value.title;
    this.book.author = this.form.value.author;
    this.book.price = this.form.value.price;
    this.book.genre = this.form.value.genre;
    this.book.about = this.form.value.about;

    if (this.user.email == "vukan.markovic97@gmail.com") {
      this.bookService.updateBookAdmin(this.book)
        .subscribe(data => console.log(data), error => console.error(error));

    this.book = null;
    this.form.resetForm();
    this.flashMessage.show('Book is updated!', { cssClass: 'alert-warning', timeout: 3000 });
    this.router.navigate(['/']);

    } else {

      this.bookService.updateBook(this.book)
        .subscribe(data => console.log(data), error => console.error(error));
          this.book = null;

      this.form.resetForm();
      this.flashMessage.show('Book is updated!', { cssClass: 'alert-warning', timeout: 3000 });
      this.router.navigate(['/profile']);
      }
    }

    onClear(form: NgForm) {
        this.book = null;
        form.resetForm();
    }
  }
```

Slika 92. update-book.component.ts

update-post.component.html

```
<form id="updateForm" (ngSubmit)="onSubmit()" #f="ngForm">
  <fieldset>
    <legend>Update a post</legend>
    <div class="form-group">
      <label for="title">Title:</label>
      <input [ngModel]="post?.title" id="title" name="title" class="form-control" placeholder="Enter the title of the post"
      type="text" required>
    </div>
    <div class="form-group">
      <label for="content">Content:</label>
      <textarea id="content" name="content" class="form-control" [ngModel]="post?.content" rows="14"></textarea>
    </div>
    <button class="btn btn-lg btn-warning" type="submit">Update a post</button>
    </fieldset>
</form>
```

Slika 93. update-post.component.html

update-post.component.ts

```ts
import { Component, OnInit, ViewChild } from '@angular/core';
import { NgForm } from '@angular/forms';
import { PostService } from '../../services/post.service';
import { Post } from '../../models/post.model';
import { Router, ActivatedRoute } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { FlashMessagesService } from 'angular2-flash-messages';

@Component({
  selector: 'app-update-post',
  templateUrl: './update-post.component.html',
  styleUrls: ['./update-post.component.css']
})

export class UpdatePostComponent implements OnInit {
  post: any = {};
  @ViewChild('f') form: NgForm;

  constructor(private postService: PostService, private router: Router, private authService: AuthService, private route: ActivatedRoute, private flashMessage: FlashMessagesService) {}

  ngOnInit() {
    if (this.authService.isLoggedIn()) {
      this.postService.getPost(this.route.snapshot.params['id'])
        .subscribe((post: Post) => {
          this.post = post;
          this.form.setValue({
            title: post.title,
            content: post.content
          })
        });
    }
    else this.router.navigate(['/login']);
  }

  onSubmit() {
    this.post.title = this.form.value.title;
    this.post.content = this.form.value.content;

    this.postService.updatePost(this.post)
      .subscribe(data => console.log(data), error => console.error(error));

    this.post = null;
    this.form.resetForm();
    this.flashMessage.show('Post is updated!', { cssClass: 'alert-warning', timeout: 3000 });
    this.router.navigate(['/blog']);
  }

  onClear(form: NgForm) {
      this.post = null;
      form.resetForm();
  }
}
```

Slika 94. update-post.component.ts

users.component.html

```html
<div class="row">
<div class="col-md-12">
<div *ngFor="let user of users">
  <div *ngIf="!isAdmin(user)" id="user" class="card text-white mb-3 flex-container bg-info">
    <div class="card-header flex-item">{{ user.firstName }} {{ user.lastName }}</div>
    <div class="card-body flex-item">
      <h4 class="card-title flex-item">{{ user.email }}</h4>
      <p class="card-text flex-item"><img style="height: 160x; width: 40%; display: block;" src= "{{ user.image }}" alt="Book cover"
        title="{{ user.firstName }}"></p>
    </div>
  </div>
</div>
```

Slika 95. users.component.html

users.component.ts

```typescript
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../../services/auth.service';
import { User } from '../../models/user.model';
import { Router } from '@angular/router';

@Component({
  selector: 'app-users',
  templateUrl: './users.component.html',
  styleUrls: ['./users.component.css'],
})

export class UsersComponent implements OnInit {
  public users: User[] = [];

  constructor(private authService: AuthService, private router: Router) {}

  ngOnInit () {
    if (!this.authService.isLoggedIn()) {
      this.router.navigate(['/login']);
    }
    else {
      this.authService.getUsers()
        .subscribe((users: User[]) => {
            this.users = users;
        }
        );
    }
  }

  isAdmin (user: User) {
    return user.email === "vukan.markovic97@gmail.com";
  }
}
```

Slika 96. users.component.ts

4. Ostalo

4.1 Linkovi

Tehnologije:
https://angular.io/
https://expressjs.com/
https://nodejs.org/en/
http://mongoosejs.com/

Tema: https://bootswatch.com/sketchy/

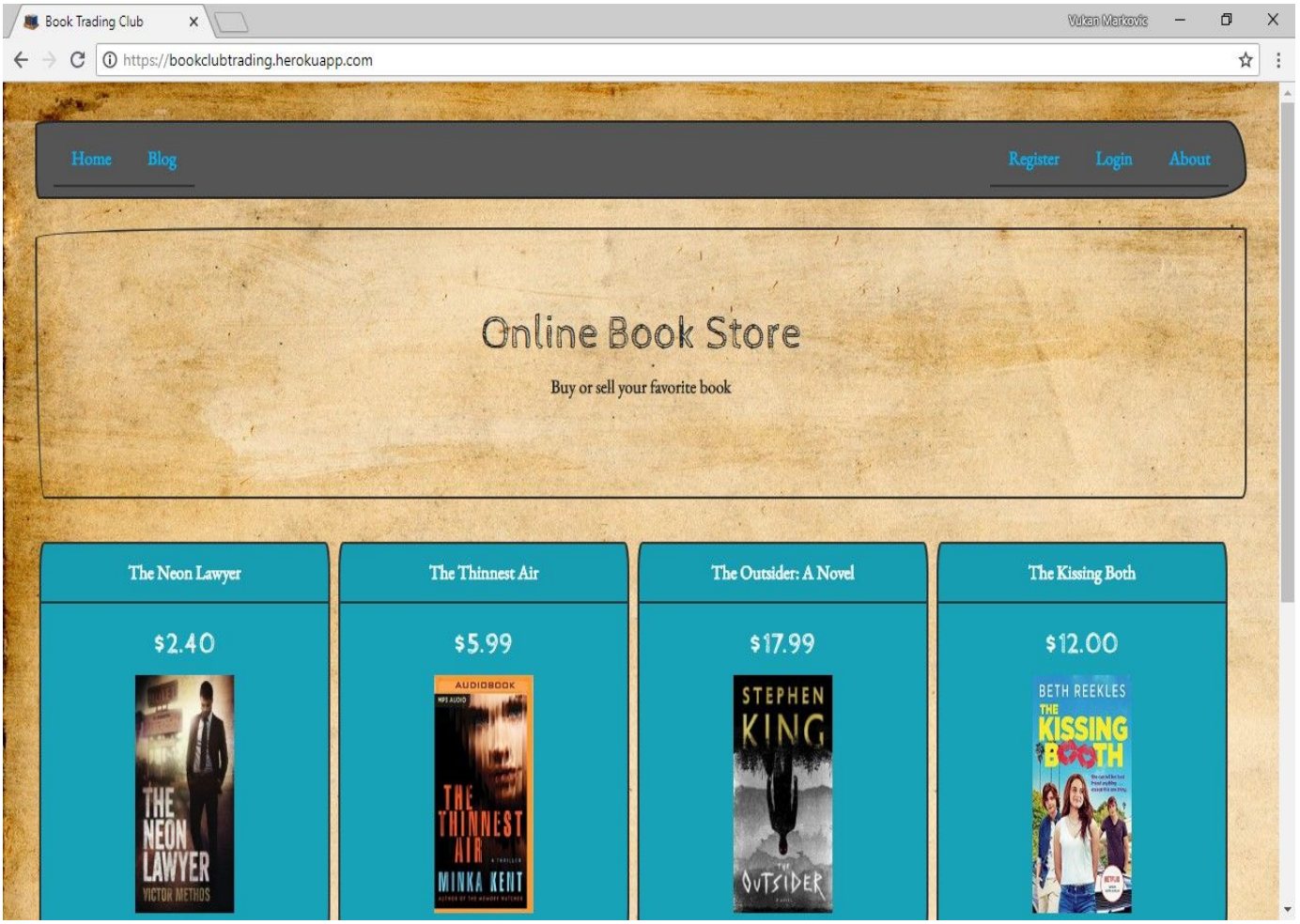
Hosting servis: https://www.heroku.com/

Sajt: https://bookclubtrading.herokuapp.com/.

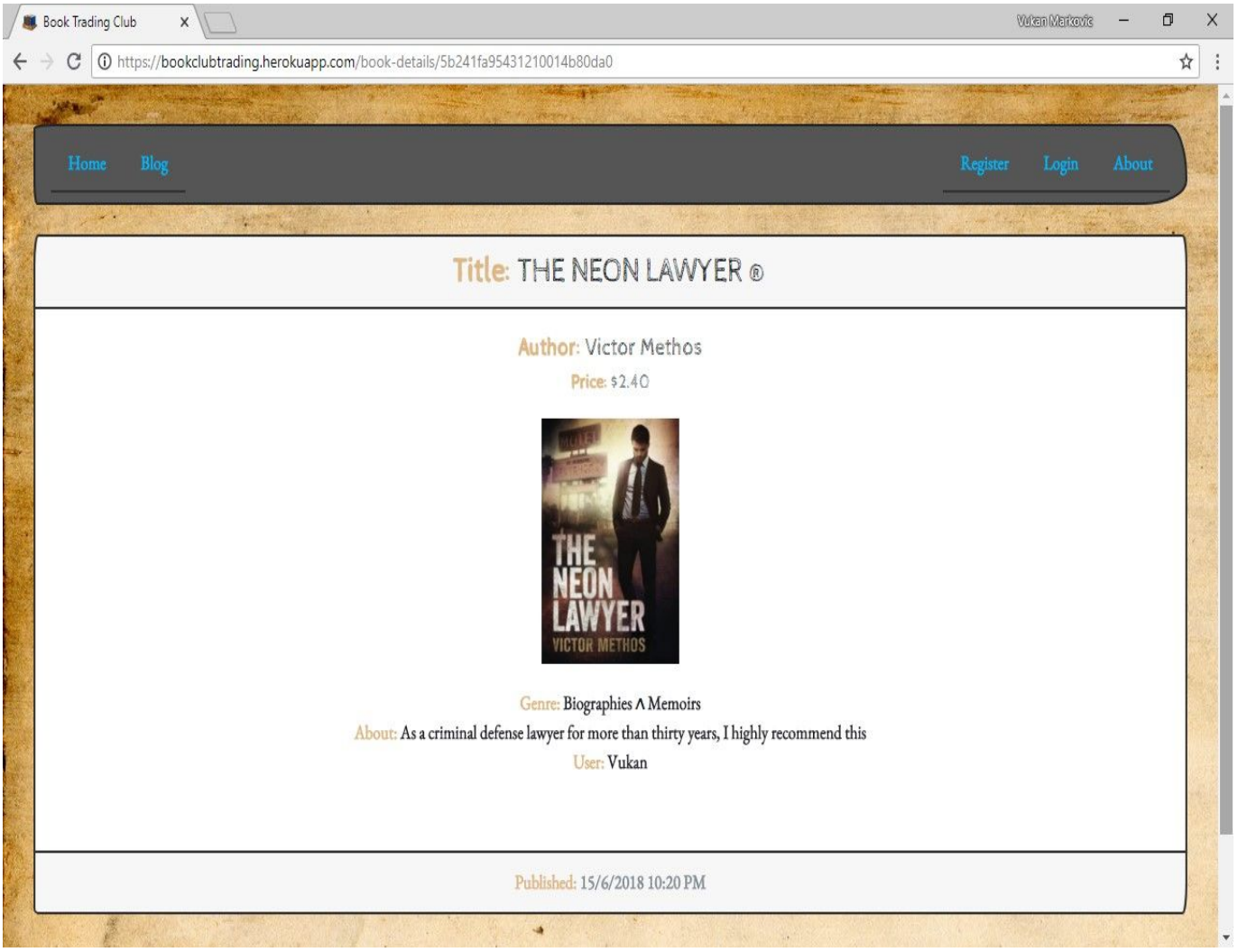
Baza podataka: https://mlab.com/

Repozitorijum: https://github.com/Vukan-Markovic/Book-trading-club

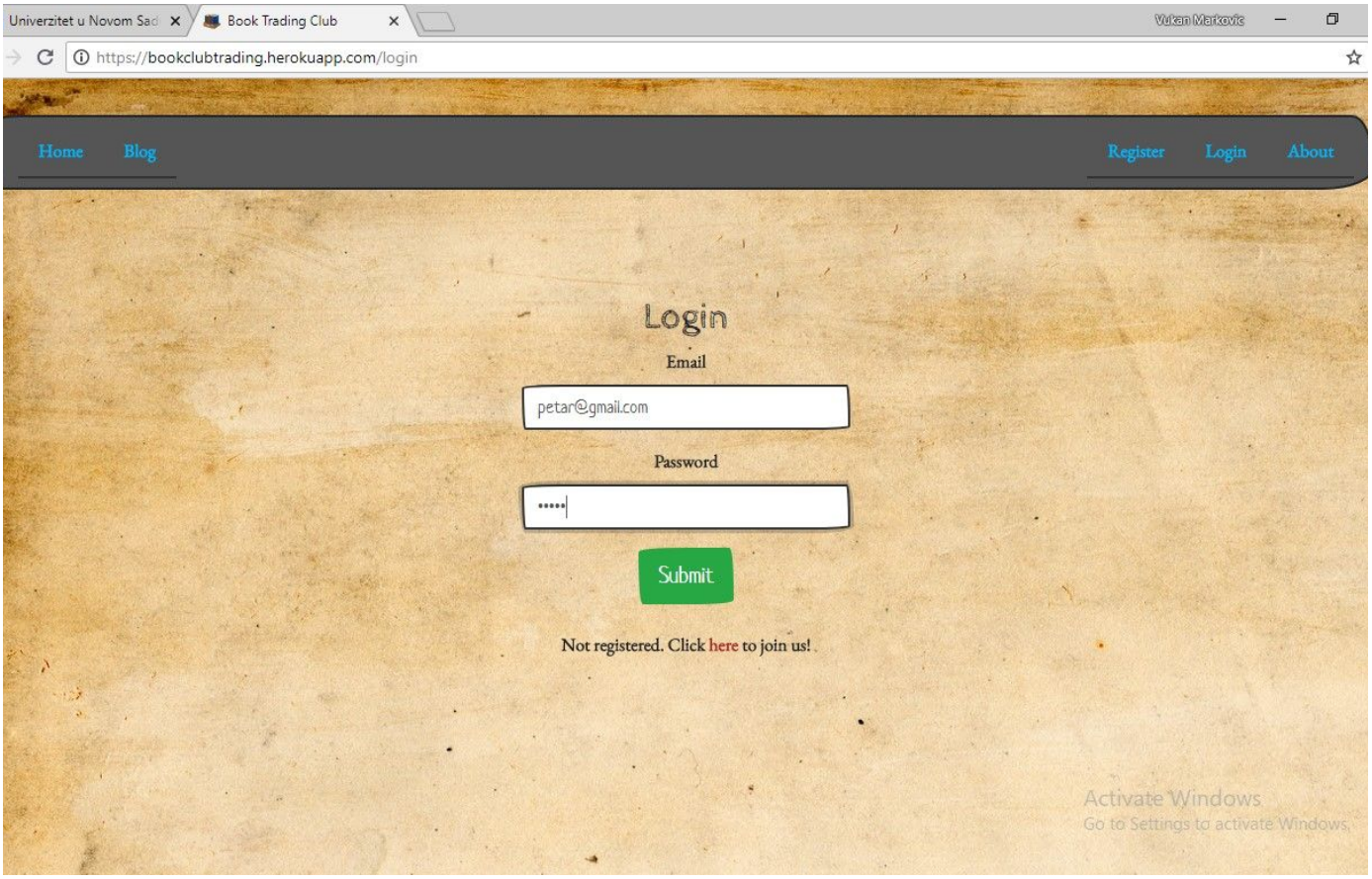
4.2 Sajt



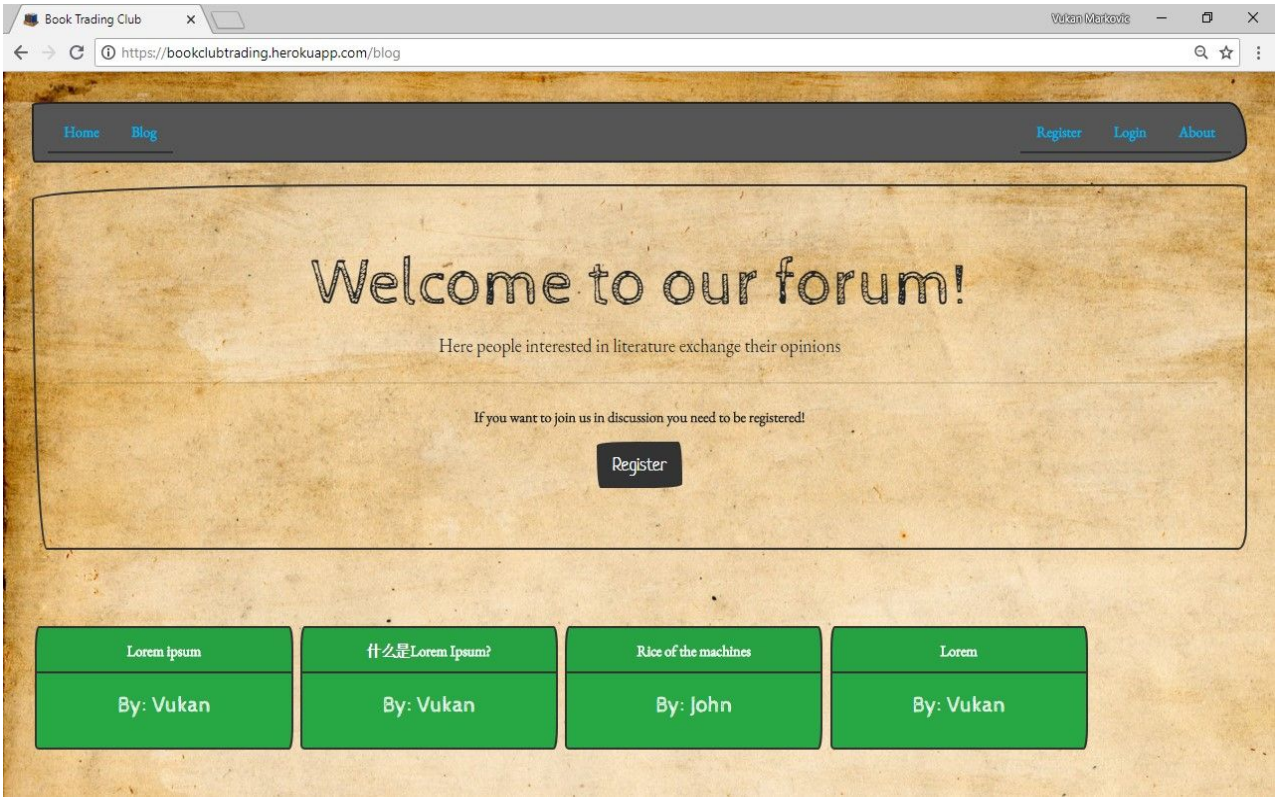
Slika 97. Naslovna strana sajta iz aspekta neregistrovanog korisnika



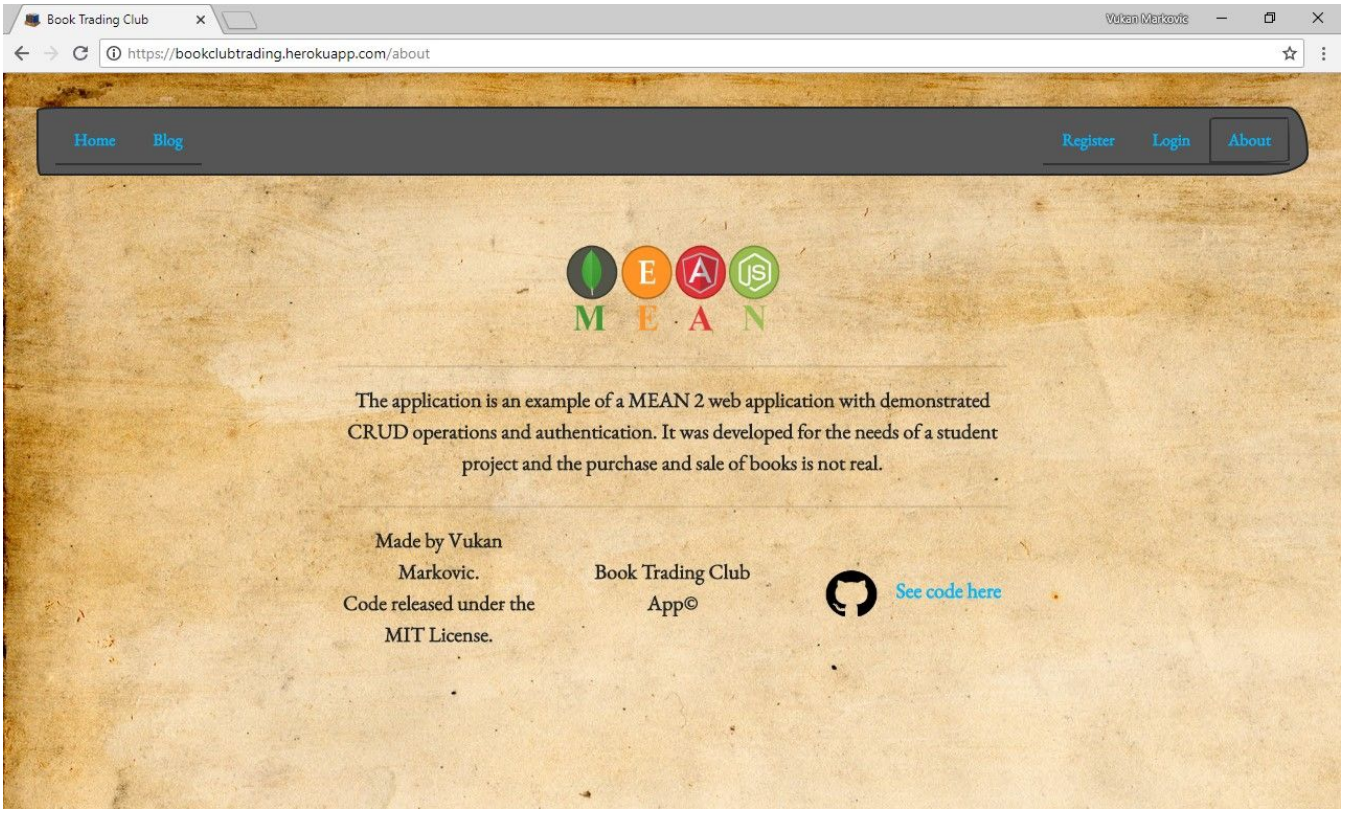
Slika 98. Neregistrovan korisnik može samo da pregleda dostupne knjige bez ostalih



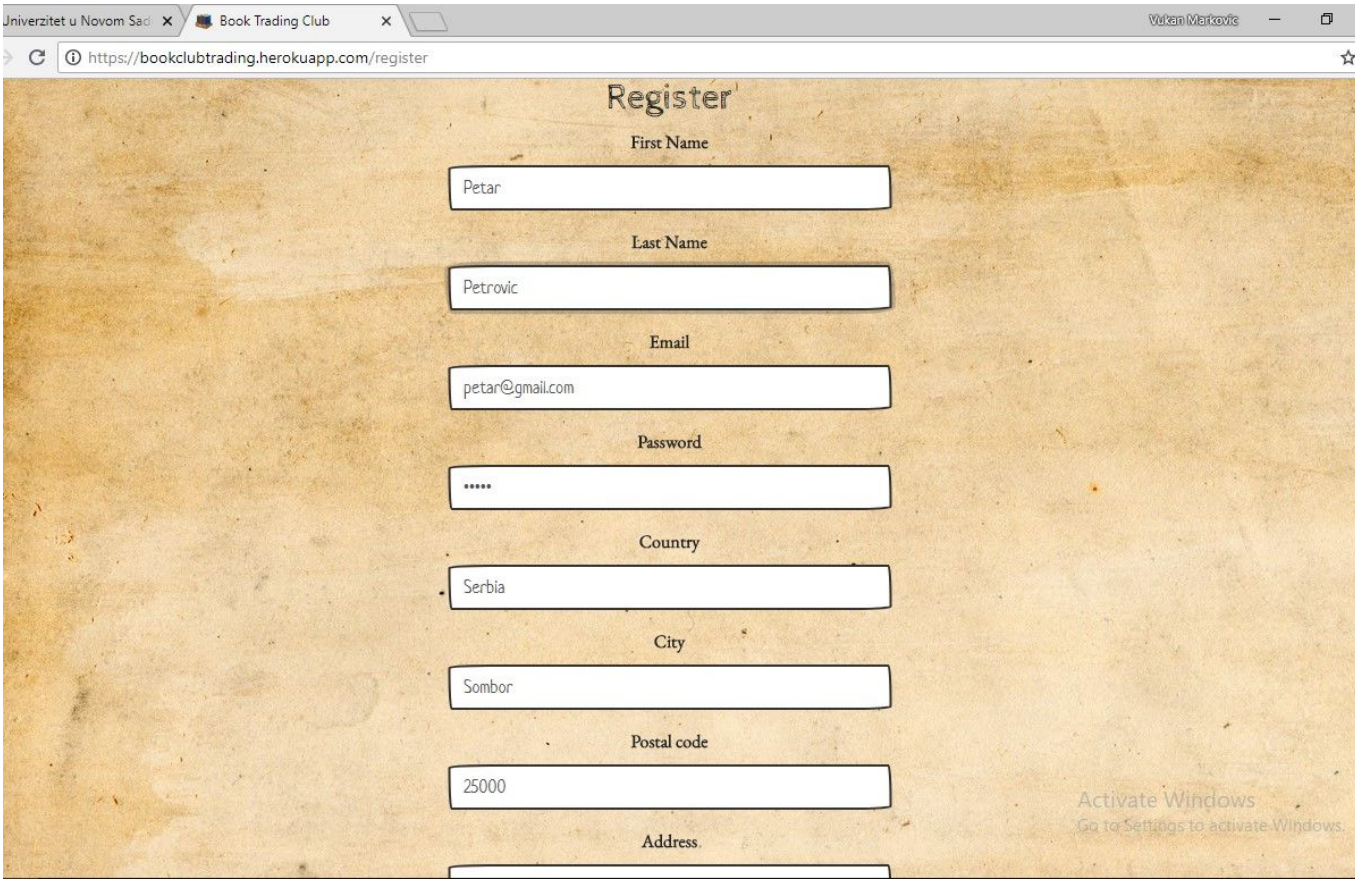
Slika 99. Login strana vidljiva samo ako korisnik nije već ulogovan

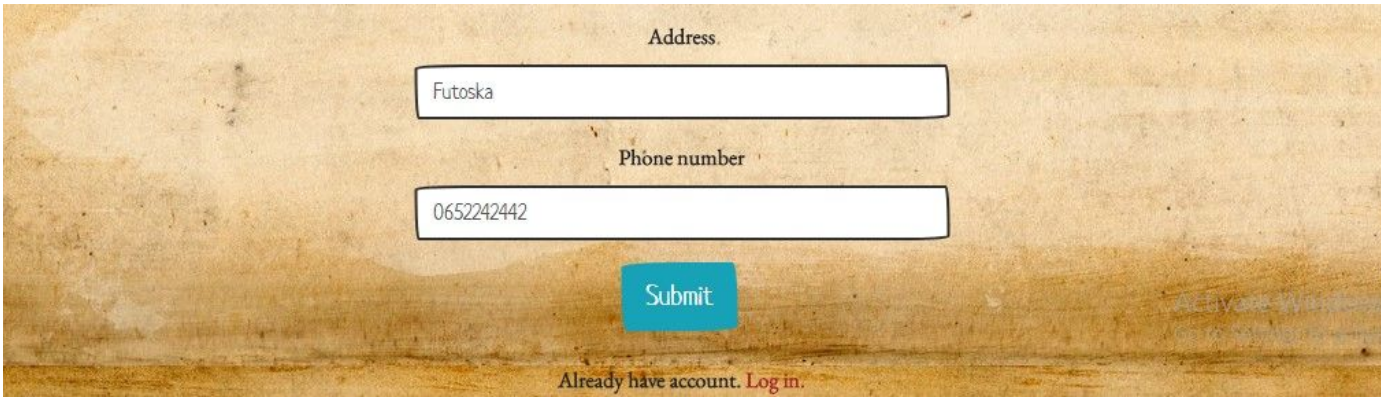


Slika 100. Blog, neregistrovan korisnik može samo da pregleda objave

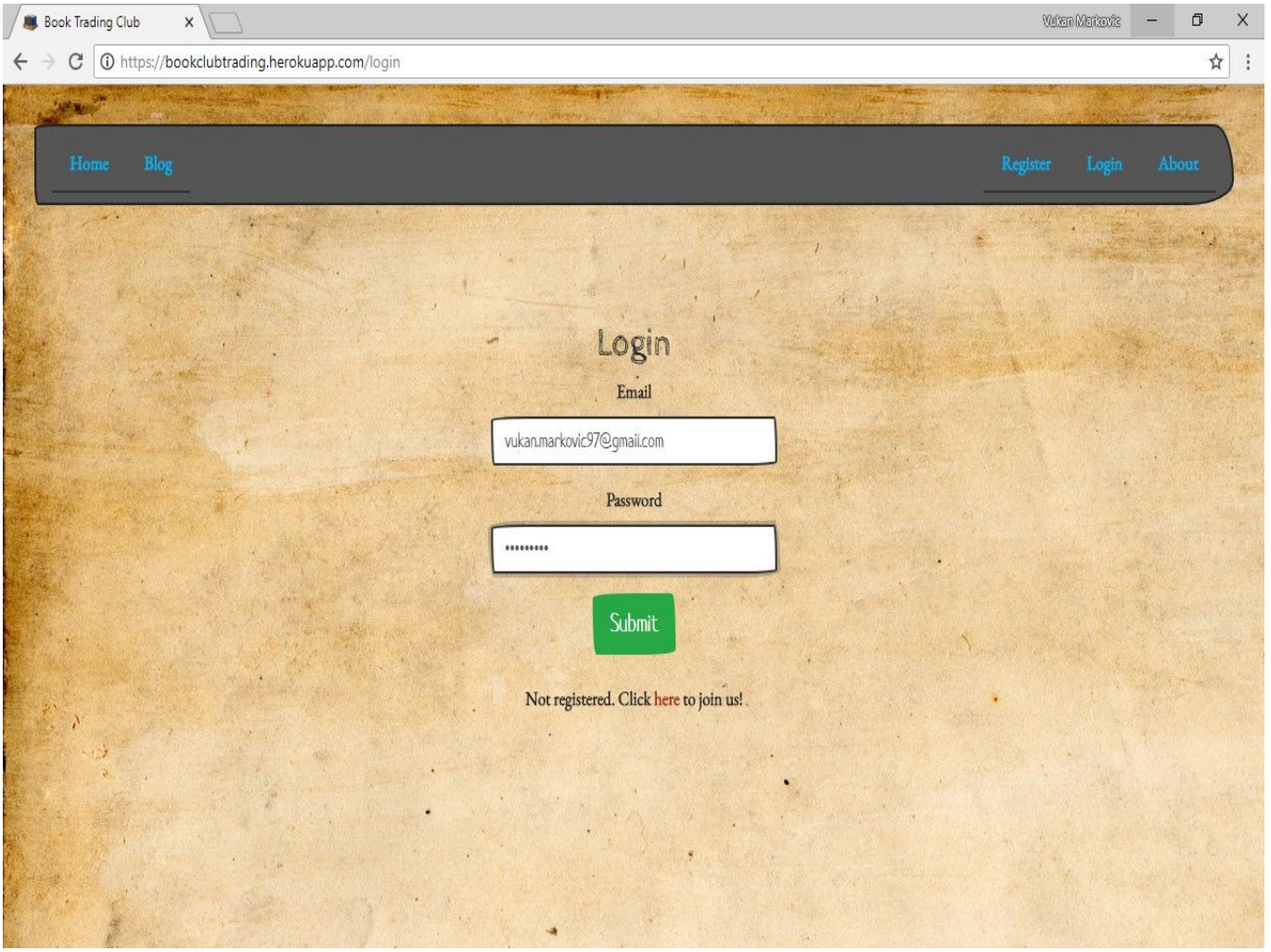


Slika 101. About stranica sa kratkim opisom sajta

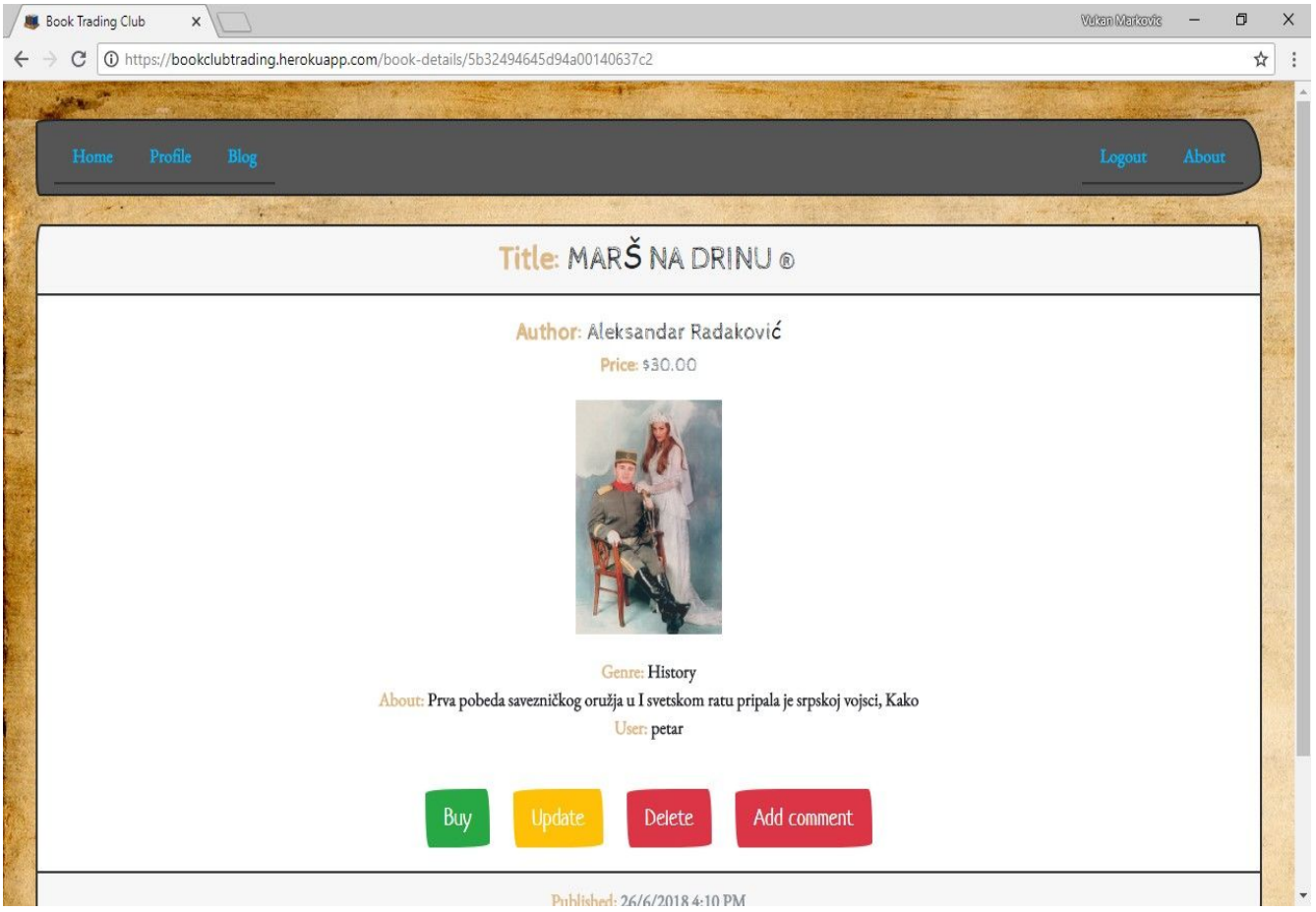




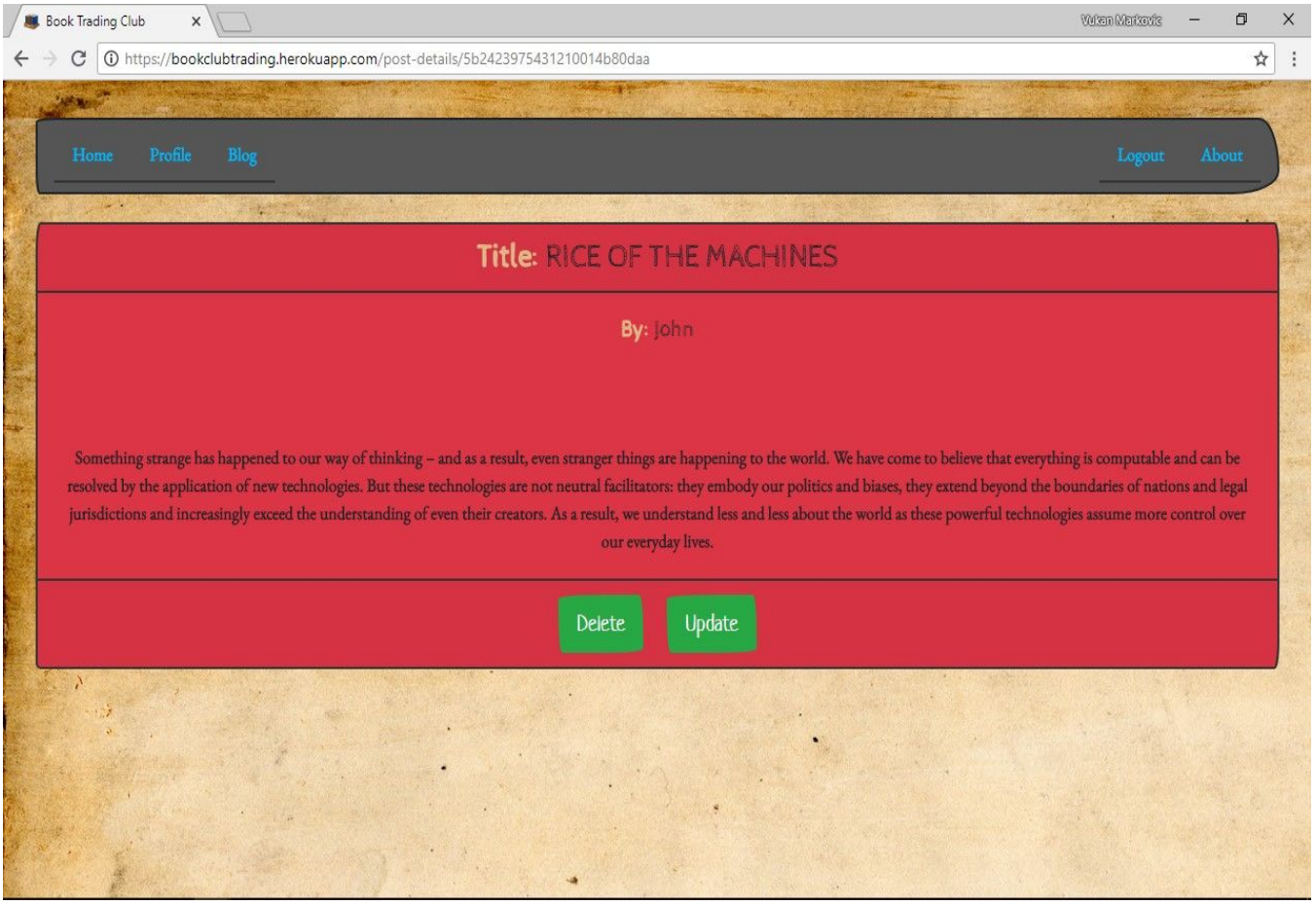
Slika 101. Register stranica vidljiva samo korisnicima koji nisu registrovani (ulogovani)



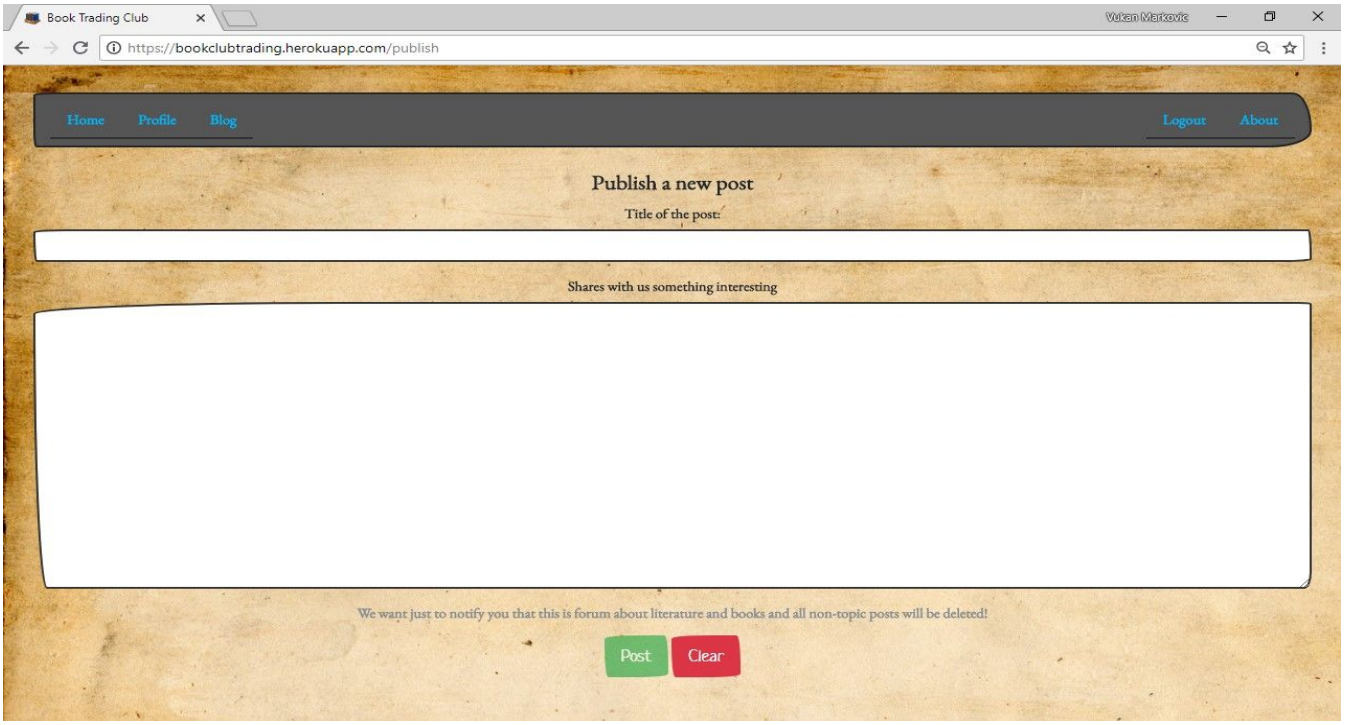
Slika 102. Administator se identifikuje preko email-a koji je jedinstven



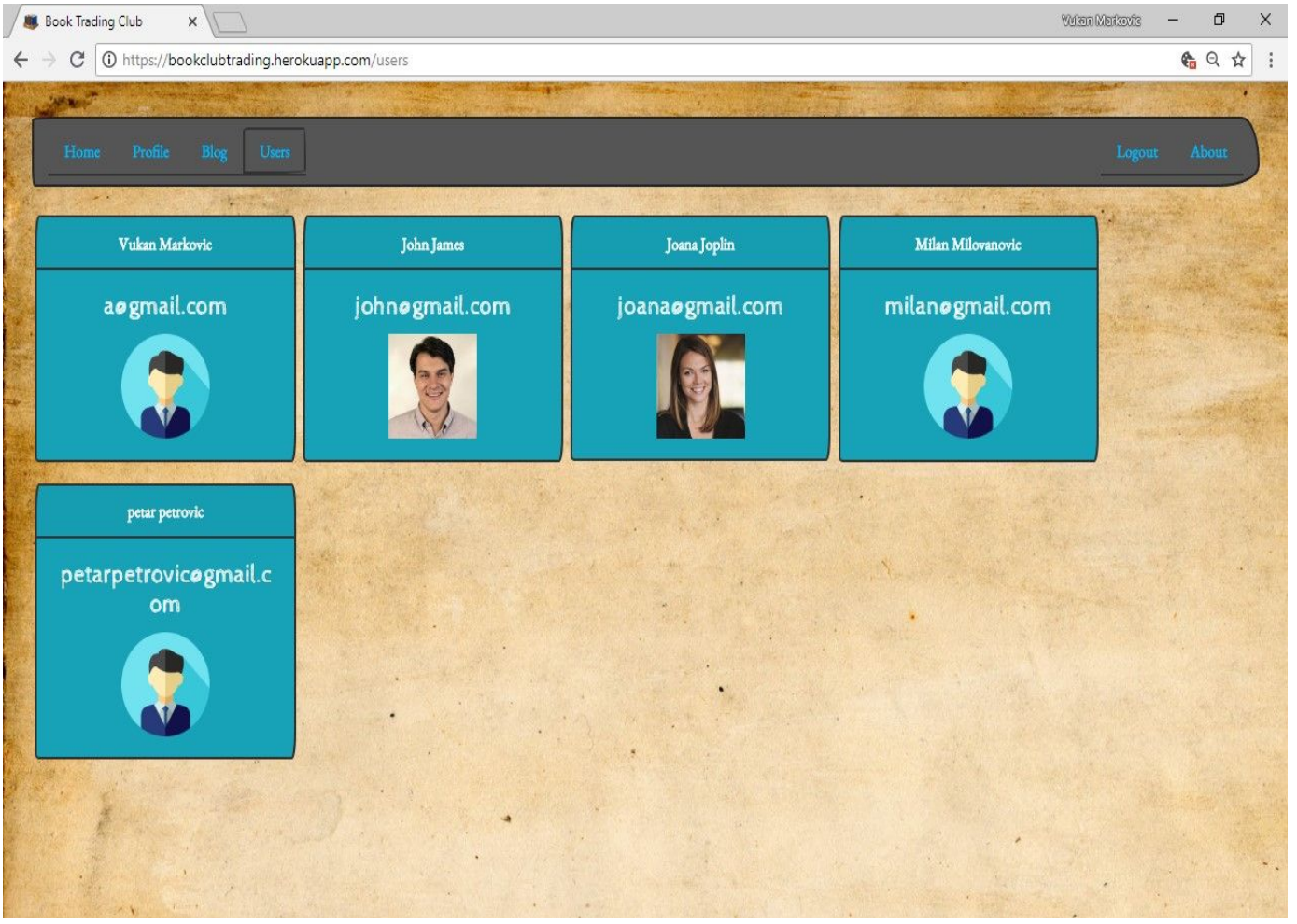
Slika 103. Detalji knjige sa aspekta administatora koji može da kupuje, briše, menja i komentriše sve knjige



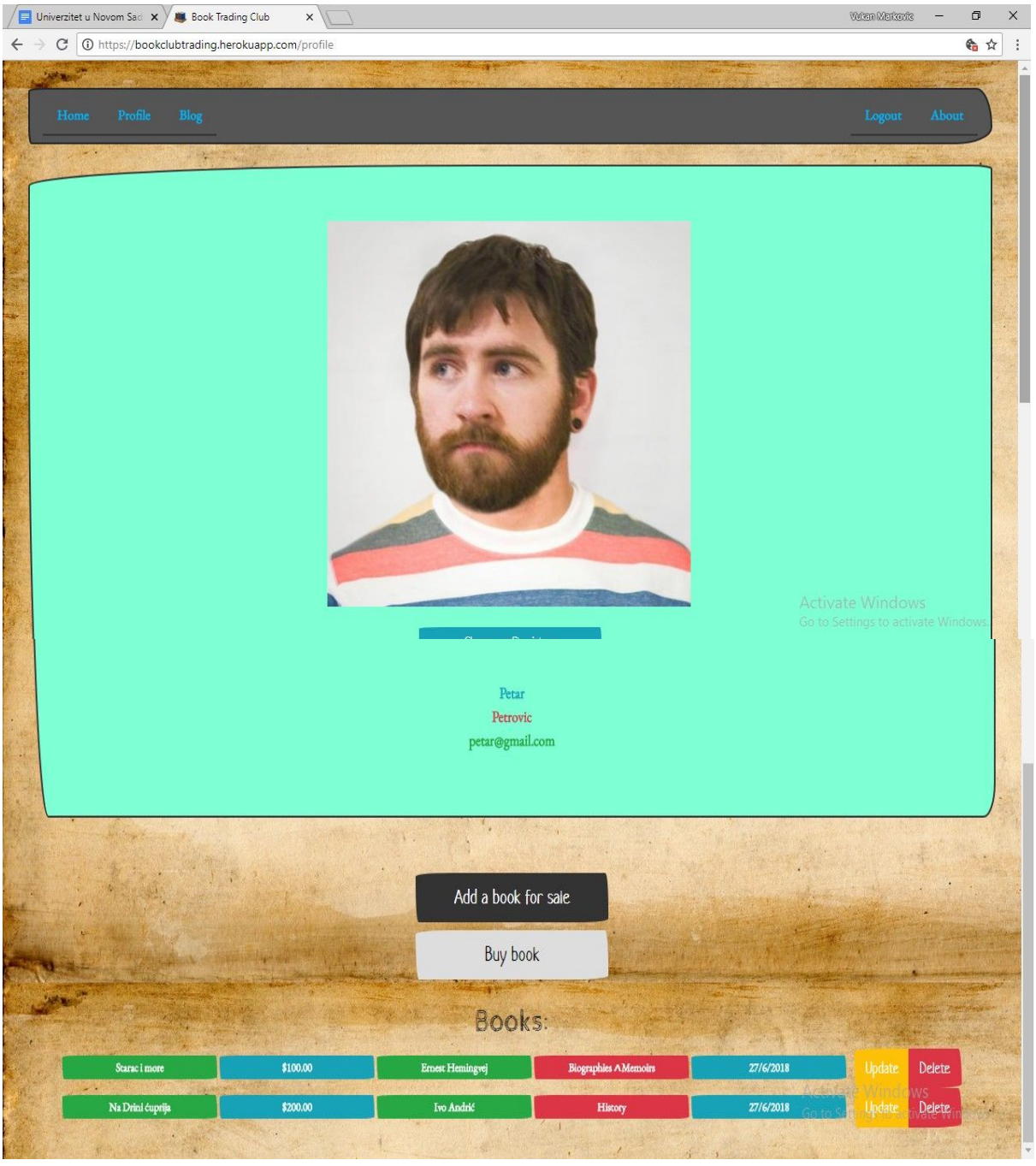
Slika 104. Objava na blogu sa aspekta administratora koji može da menja i briše sve objave



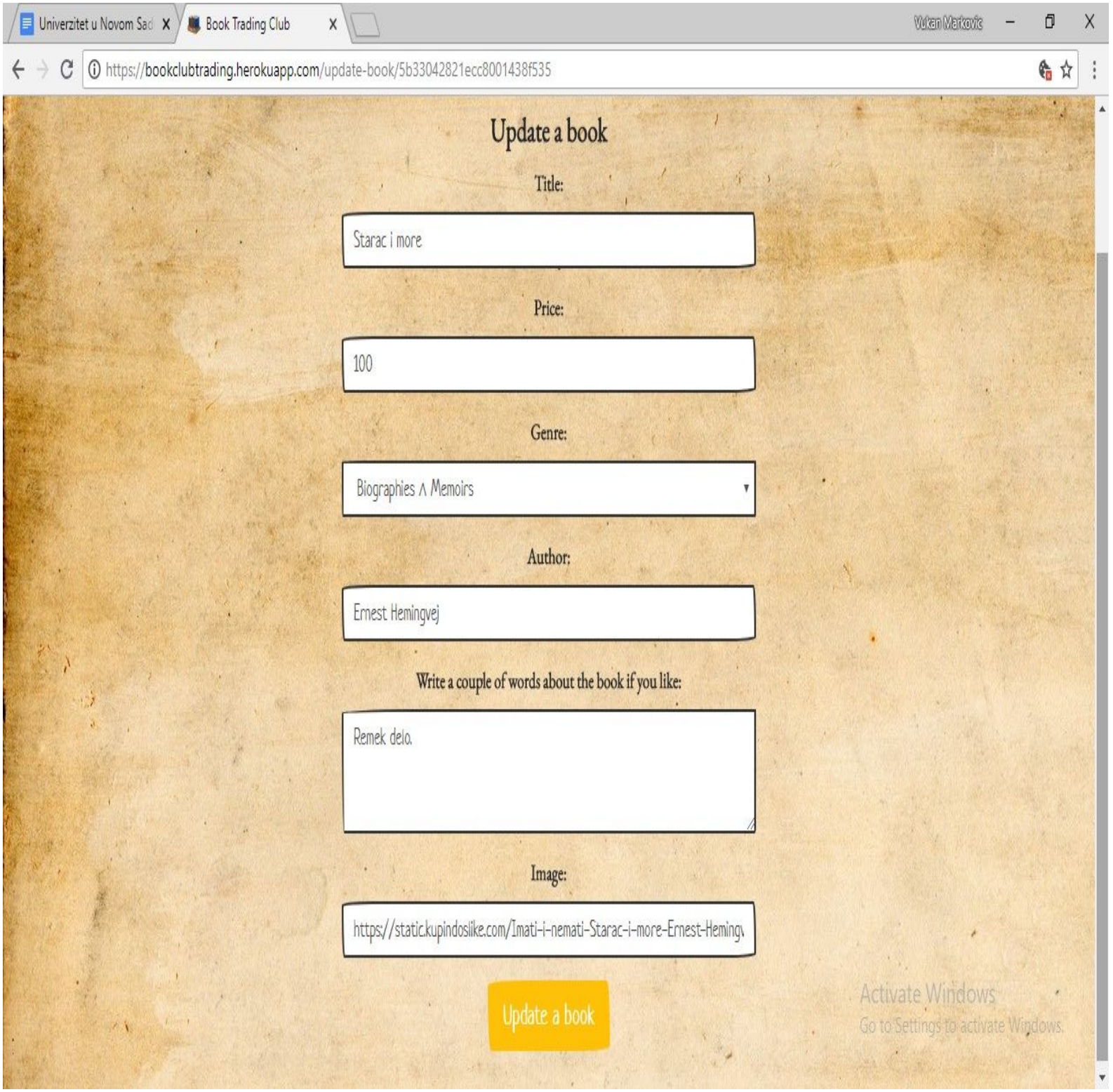
Slika 105. Objavljivanje novog posta, ovu mogućnost imaju samo registrovani korisnici



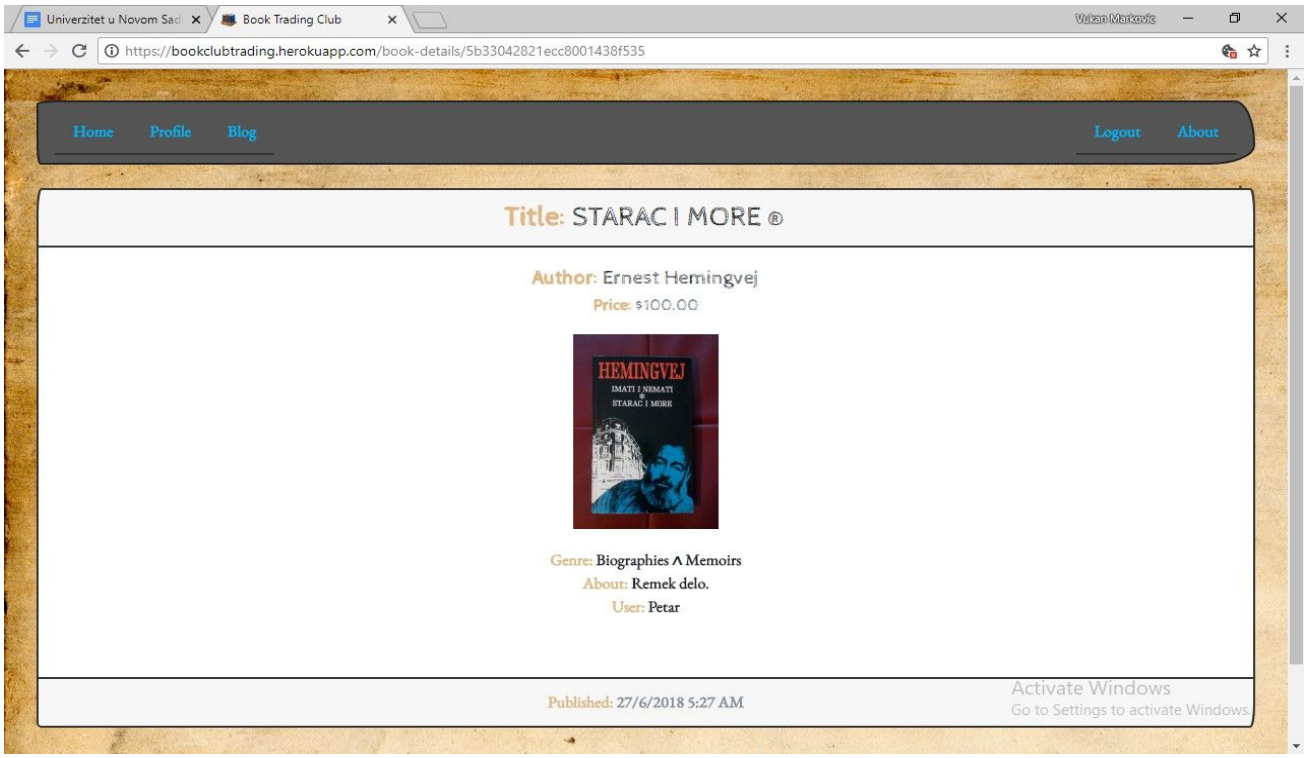
Slika 106. Administrator ima mogućnost pregleda svih registrovanih korisnika



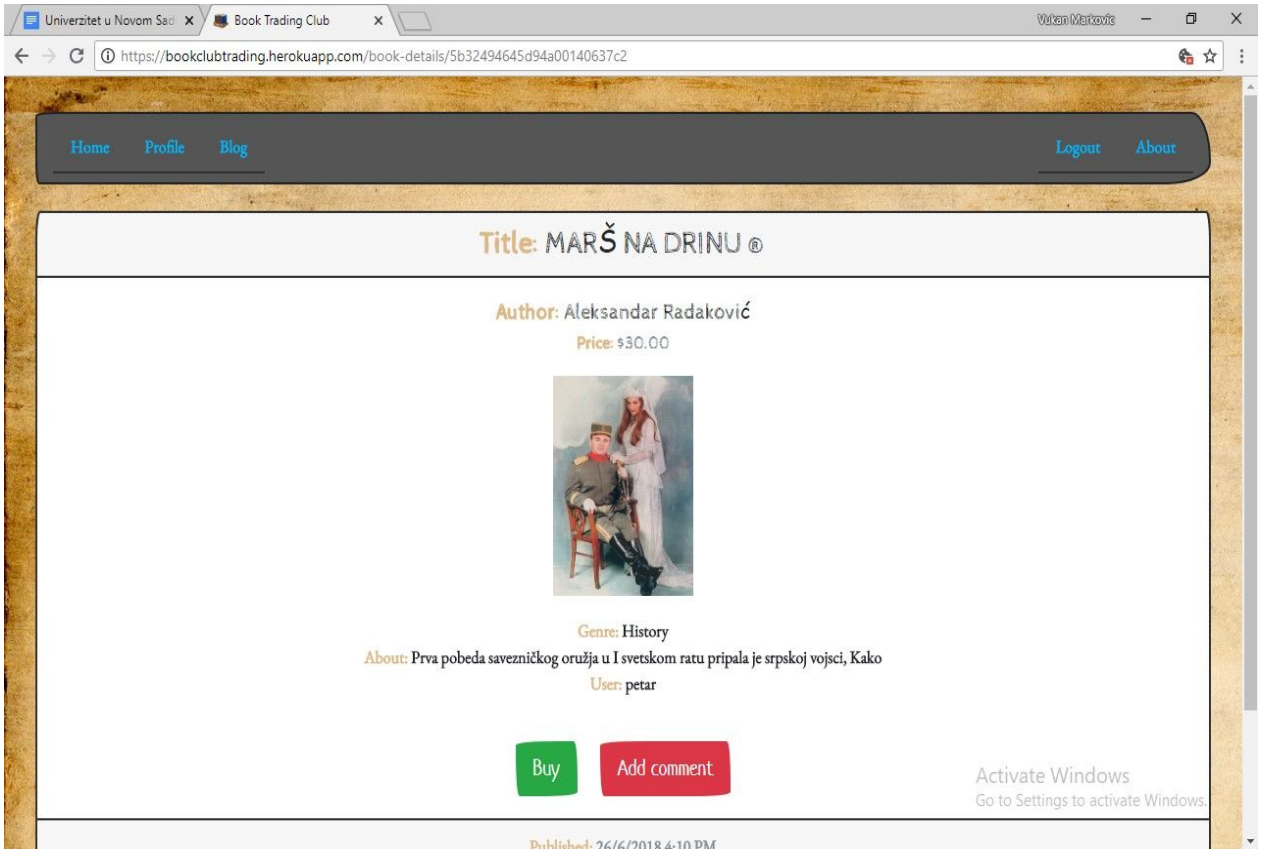
Slika 107. Profilna strana



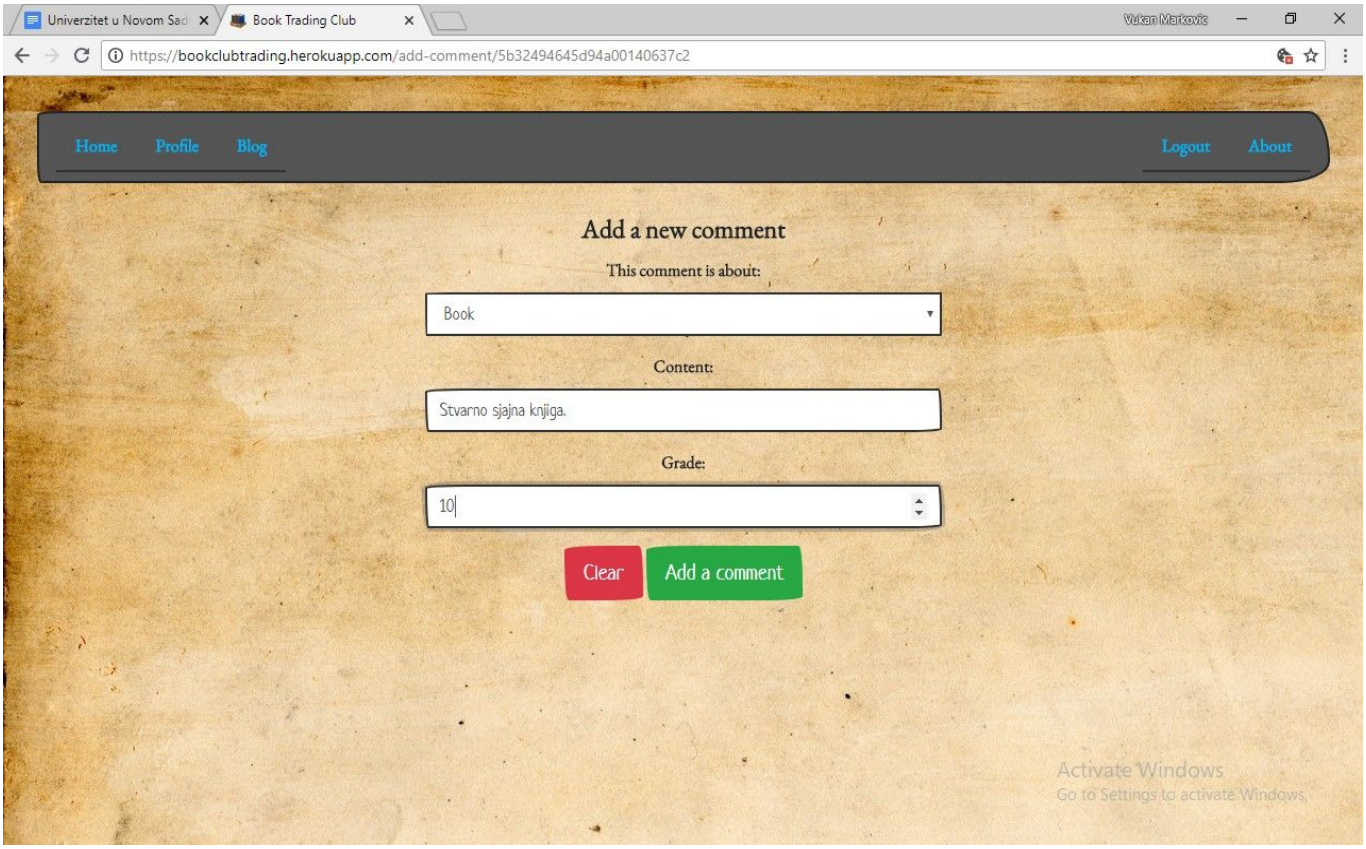
Slika 108. Ažuriranje knjige



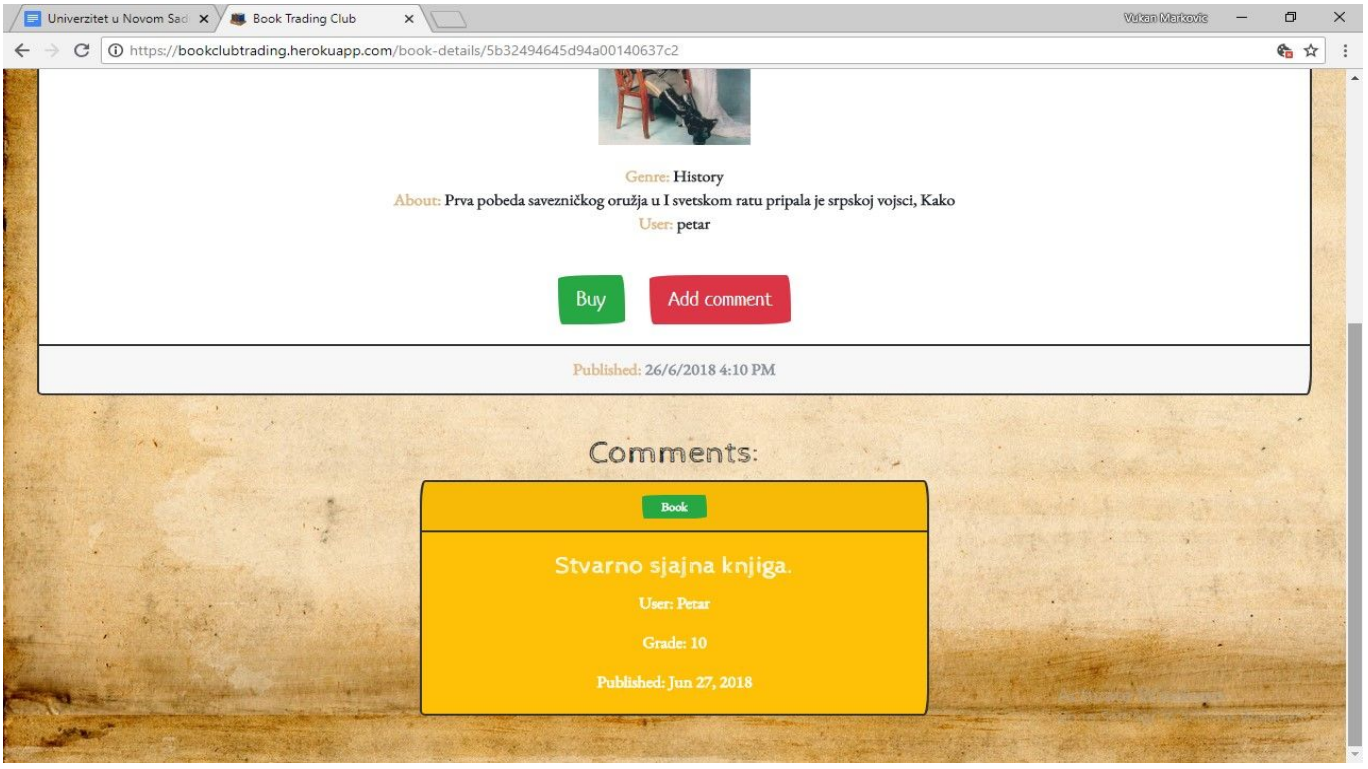
Slika 109. Pregled korisnikove knjige



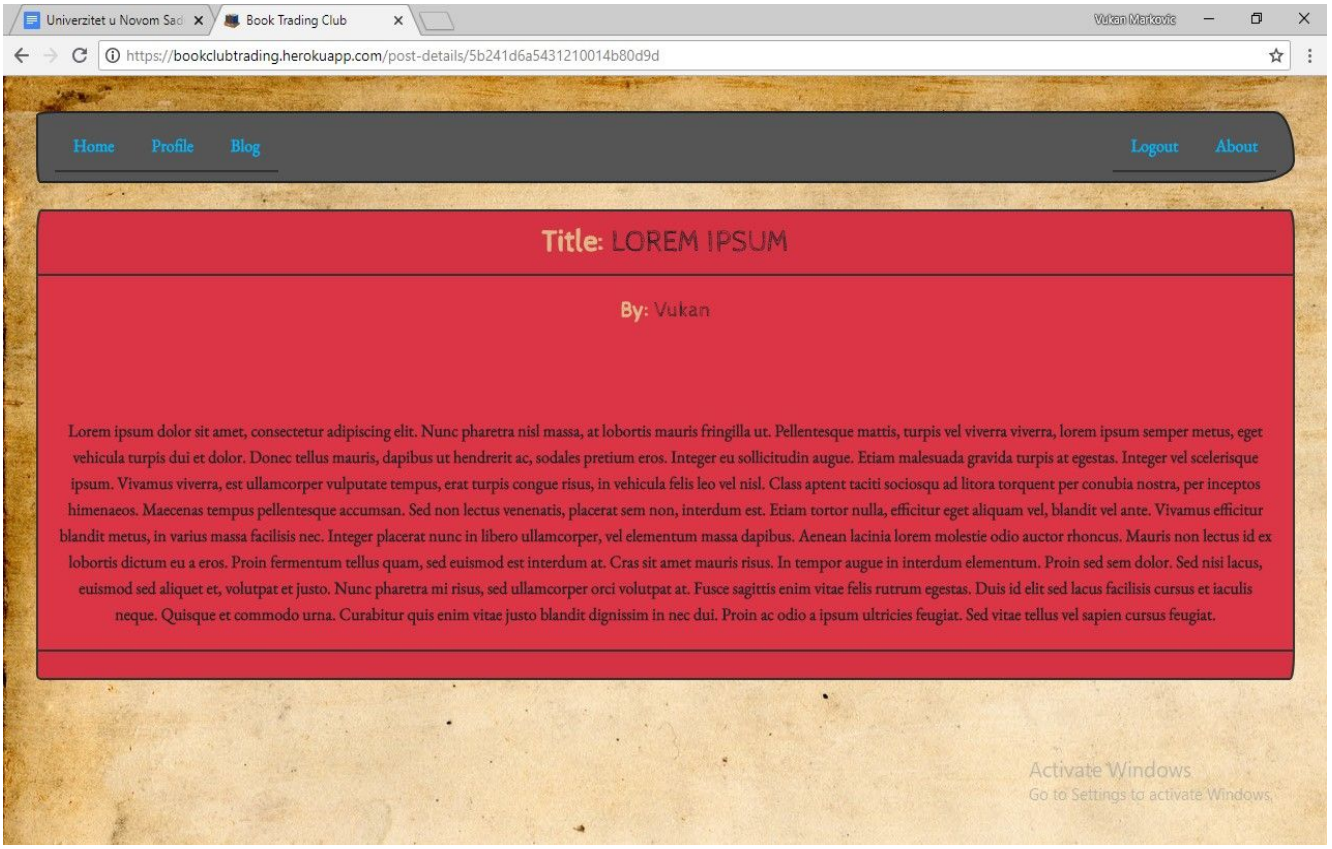
Slika 110. Pregled knjige od strane drugog korisnika



Slika 111. Dodavanje novog komentara

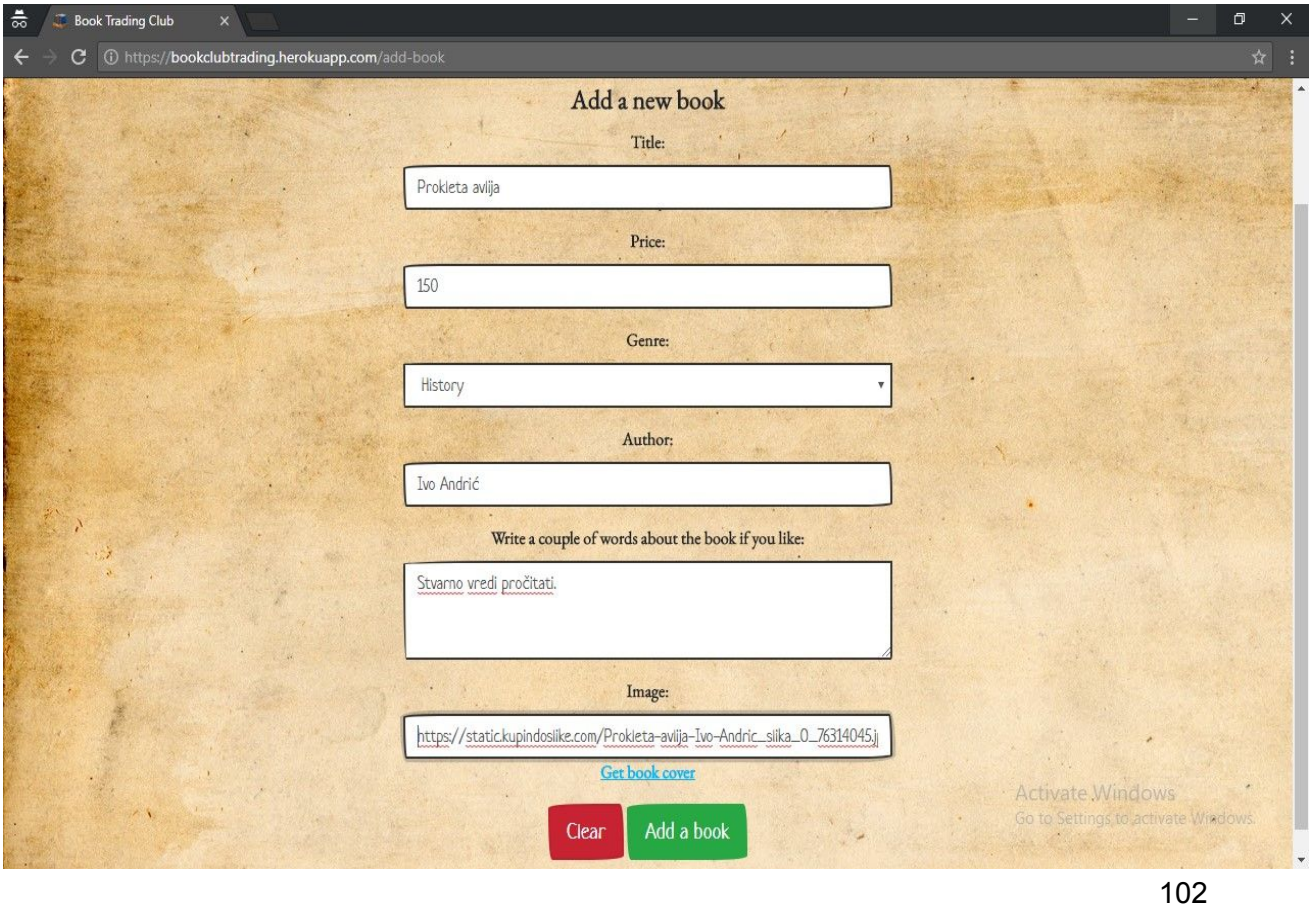


Slika 112. Dodani komentar

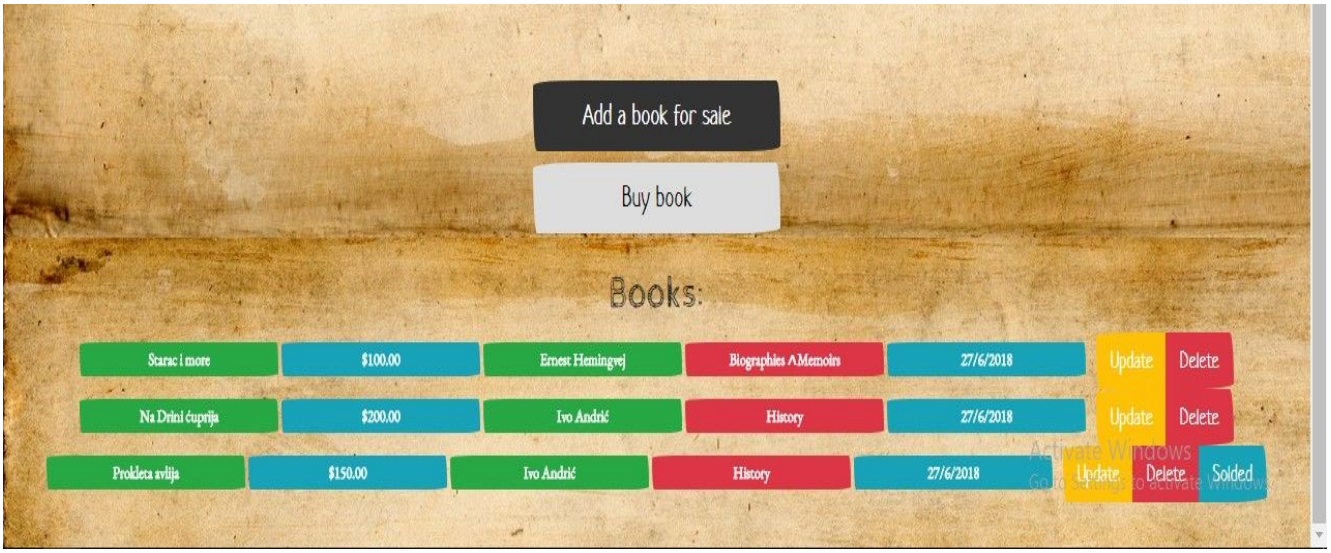


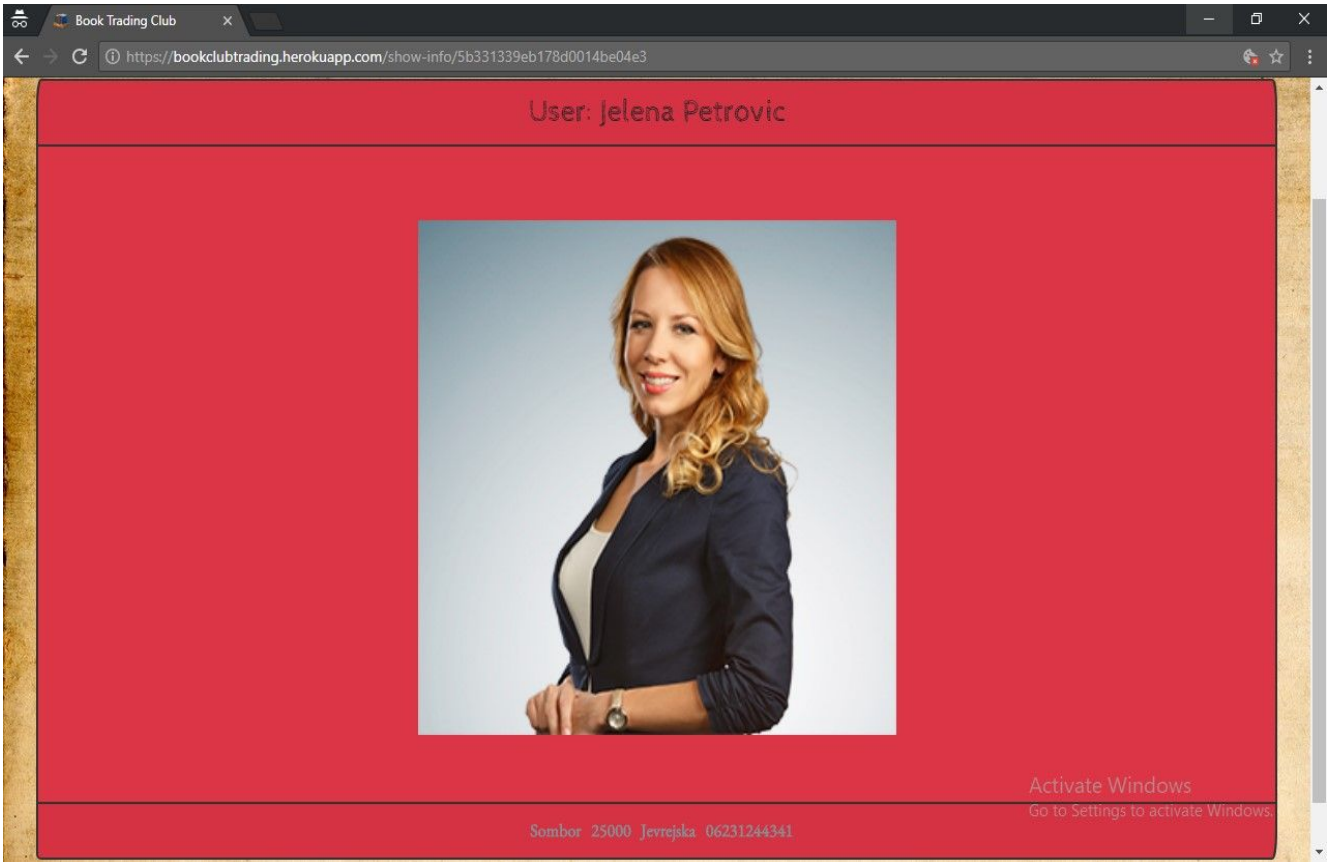
Slika 113. Pregled objave sa apseta korisnika koji nije administrator

Slika 114. Dodavanje nove knjige



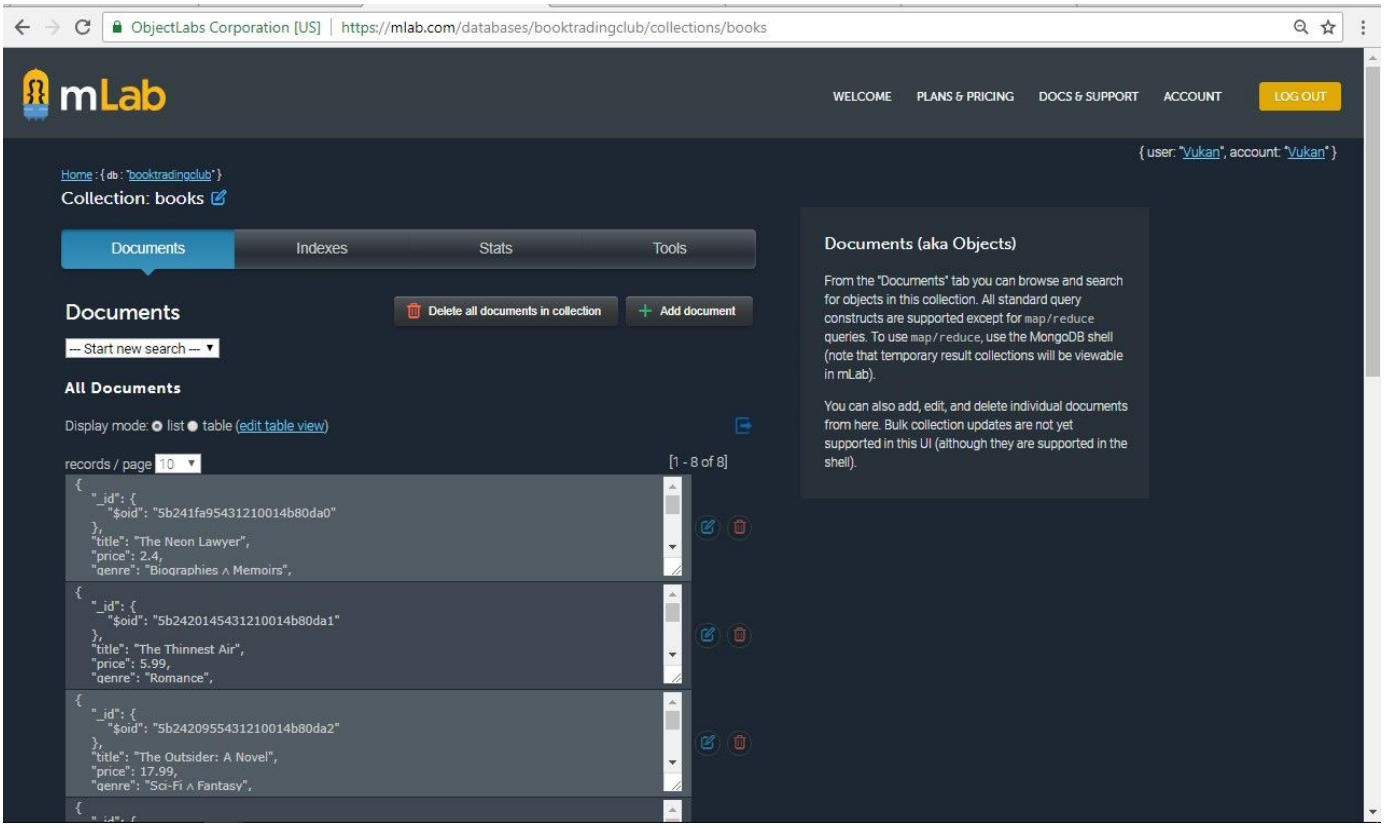
Slika 115. Korisnik može da vidi podatke neophodne za obavljanje kupovine...

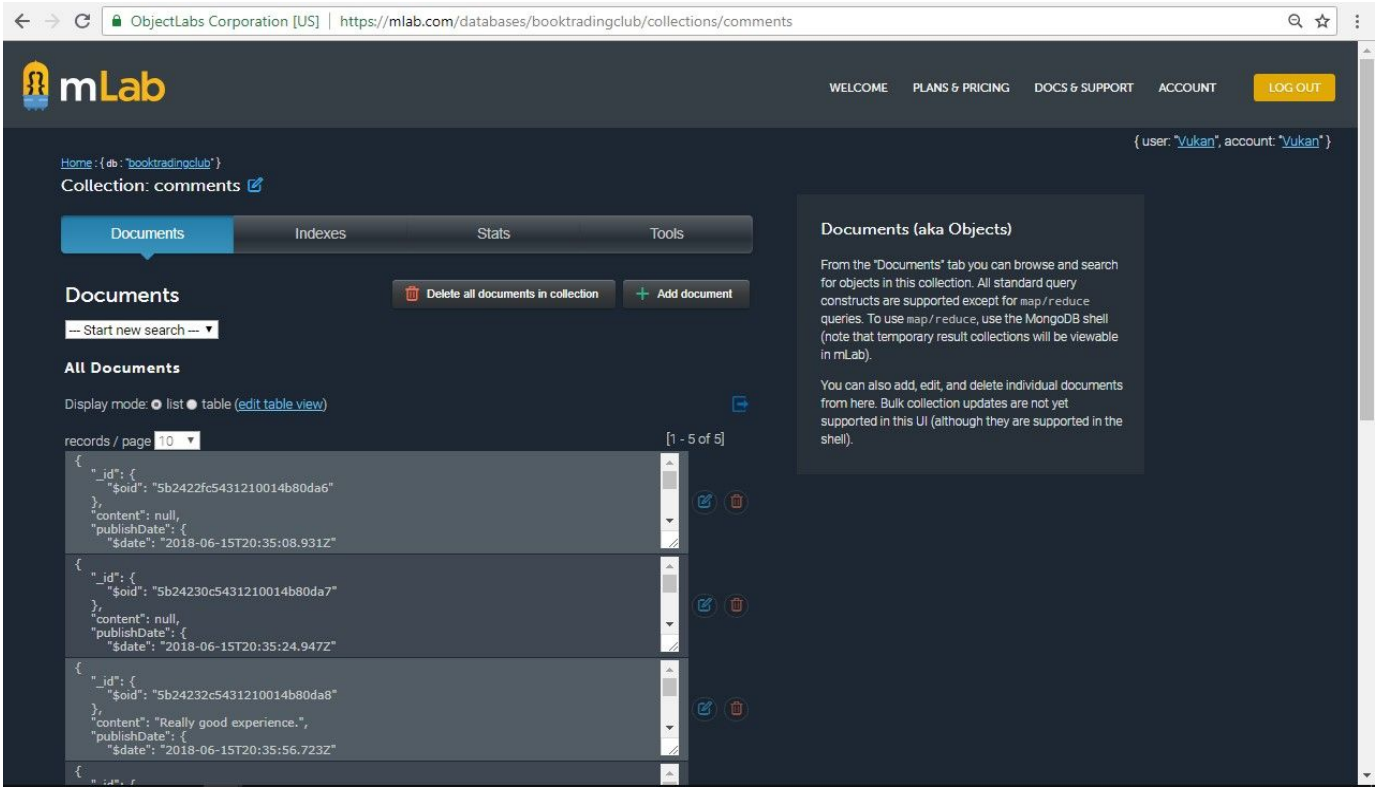


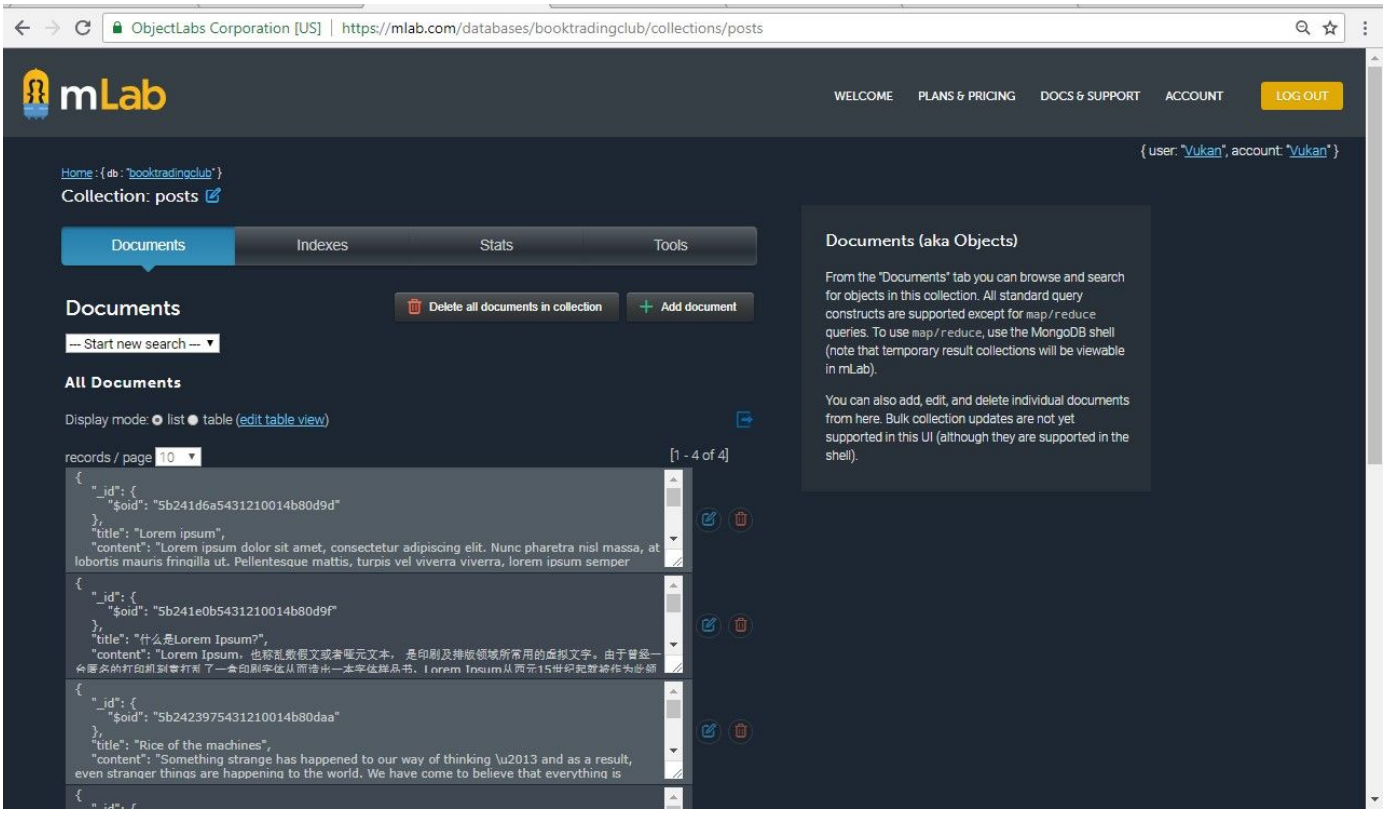


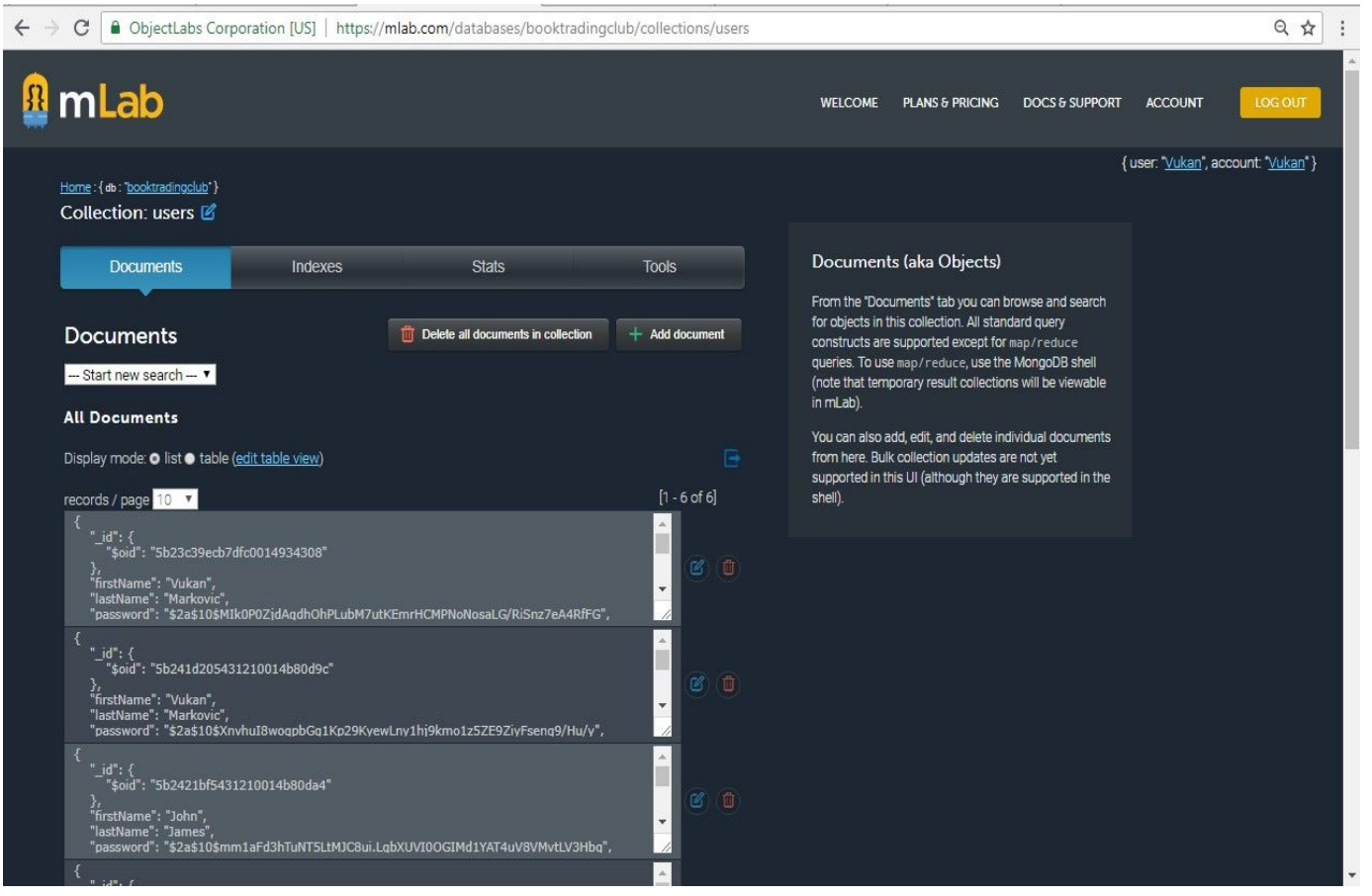
...onog korisnika koji ju je izvršio.

4.3Bazapodataka






ObjectLabs Corporation [US] | https://mlab.com/databases/booktradingclub/collections/posts
mLab
WELCOME
PLANS & PRICING
DOCS & SUPPORT
ACCOUNT
LOG OUT
{ user: "Vukan", account: "Vukan" }
Home : { db : "booktradingclub" }
Collection: posts
Documents
Indexes
Stats
Tools
Documents
Delete all documents in collection
Add document
--- Start new search ---
All Documents
Display mode: list table (edit table view)
records / page 10
[1 - 4 of 4]
{
"_id": {
"$oid": "5b241d6a5431210014b80d9d"
},
"title": "Lorem ipsum",
"content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc pharetra nisl massa, at lobortis mauris fringilla ut. Pellentesque mattis, turpis vel viverra viverra, lorem ipsum semper
{
"_id": {
"$oid": "5b241e0b5431210014b80d9f"
},
"title": "什么是Lorem Ipsum?",
"content": "Lorem Ipsum，也称乱数假文或者哑元文本， 是印刷及排版领域所常用的虚拟文字。由于曾经一
{
"_id": {
"$oid": "5b2423975431210014b80daa"
},
"title": "Rice of the machines",
"content": "Something strange has happened to our way of thinking \u2013 and as a result, even stranger things are happening to the world. We have come to believe that everything is
Documents (aka Objects)
From the "Documents" tab you can browse and search for objects in this collection. All standard query constructs are supported except for map/reduce queries. To use map/reduce, use the MongoDB shell (note that temporary result collections will be viewable in mLab).
You can also add, edit, and delete individual documents from here. Bulk collection updates are not yet supported in this UI (although they are supported in the shell).


ObjectLabs Corporation [US] | https://mlab.com/databases/booktradingclub/collections/users
mLab
WELCOME
PLANS & PRICING
DOCS & SUPPORT
ACCOUNT
LOG OUT
{ user: "Vukan", account: "Vukan" }
Home : { db : "booktradingclub" }
Collection: users
Documents
Indexes
Stats
Tools
Documents
Delete all documents in collection
Add document
--- Start new search ---
All Documents
Display mode: list table (edit table view)
records / page 10
[1 - 6 of 6]
{
"_id": {
"$oid": "5b23c39ecb7dfc0014934308"
},
"firstName": "Vukan",
"lastName": "Markovic",
"password": "$2a$10$MIk0P0ZjdAqdhOhPLubM7utKEmrHCMPNoNosaLG/RiSnz7eA4RfFG",
{
"_id": {
"$oid": "5b241d205431210014b80d9c"
},
"firstName": "Vukan",
"lastName": "Markovic",
"password": "$2a$10$XnvhuI8woqpbGq1Kp29KyewLnv1hj9kmo1z5ZE9ZiyFsenq9/Hu/y",
{
"_id": {
"$oid": "5b2421bf5431210014b80da4"
},
"firstName": "John",
"lastName": "James",
"password": "$2a$10$mm1aFd3hTuNT5LtMJC8ui.LqbXUVI0OGIMd1YAT4uV8VMvtLV3Hbq",
Documents (aka Objects)
From the "Documents" tab you can browse and search for objects in this collection. All standard query constructs are supported except for map/reduce queries. To use map/reduce, use the MongoDB shell (note that temporary result collections will be viewable in mLab).
You can also add, edit, and delete individual documents from here. Bulk collection updates are not yet supported in this UI (although they are supported in the shell).