Aplikacija za organizaciju rada knjižnice

Špoljarić, Bernarda

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij Informatika

APLIKACIJA ZA ORGANIZACIJU RADA KNJIŽNICE

Završni rad

Bernarda Špoljarić

Osijek, 2020.

SADRŽAJ

1. UVOD

Iako većina ljudi govori kako knjige u fizičkom smislu polako nestaju, profesori i studenti raznih fakulteta i dalje ih koriste kao primarnu literaturu. Knjižnice, same po sebi, imaju veliki utjecaj na sve ljude, ali pogotovo na njihovo znanje i kulturu. Opseg knjiga u današnje vrijeme je poprilično velik, te se mnoge knjižnice susreću s problemima, kako knjiga tako i samih korisnika. Iz tog razloga postoje brojni programi koji takvu komunikaciju i rad pokušavaju učiniti jednostavnijim. Upravo je to cilj ovoga završnoga rada, odnosno napraviti aplikaciju koja će pomoći u organizaciji same knjižnice. Kako bi se omogućila i određena vrsta komunikacije između knjižnice i samog korisnika, u završni rad bilo je potrebno uključiti i korisnički profil, a ne samo administratorski. Prijavom putem korisničkog profila omogućuje se korisnicima pregled svih knjiga koje se u knjižnici nalaze kao i određene informacije o pojedinoj knjizi. Također, omogućuje im pregled svih trenutno posuđenih knjiga te samim time i pregled zakašnjelih knjiga. Slično, postoji mogućnost i pregleda uplaćenih godišnjih ili mjesečnih članarina za pojedinog korisnika. Za razliku od korisničkog profila, administratorski profil ima drugačiju ulogu. Prilikom prijave administratora, odnosno knjižničara, otvara se administratorski profil koji sadrži mnogo više funkcionalnosti kao potvrda posudbe knjige za pojedinog korisnika, plaćanje članarine za pojedinog korisnika, pregled i pretraga svih korisnika knjižnice te njihovo brisanje. Administrator, također, ima mogućnost pregleda svih knjiga koje se u knjižnici nalaze, ali ima i mogućnost dodavanja novih ukoliko knjižnica odluči proširiti ili smanjiti svoj repertoar. Izrada ove aplikacije pojednostavljuje rad knjižničara, ali i samih korisnika tako što mogu izvršiti rezervaciju knjige i plaćanje članarine preko same aplikacije. Kako je aplikacija osmišljena te implementirana ukratko je navedeno u nastavku, ali detaljnije objašnjeno u daljnjim poglavljima.

Glavni dio završnog rada sastoji se od tri velika poglavlja:

1. Korištene tehnologije i programski jezici prilikom izrade završnog rada: kratki opis korištenih programa i programskih jezika
2. Realizacija aplikacije: prikaz i detaljniji opis stvaranja funkcionalnosti aplikacije za administratorski i korisnički profil
3. Izgled korisničkog sučelja: prikaz načina stvaranja izgleda aplikacije za administratorski i korisnički profil

1.1. Zadatak završnog rada

Zadatak završnog rada je izraditi aplikaciju koja će služiti kao pomoć za rad knjižnice. Potrebno je kreirati najmanje dva korisnička profila koja se razlikuju po ovlastima. Administratorski profil treba imati mogućnost manipulacije informacija o knjigama i informacija o pojedinom korisniku. Korisničkom profilu omogućiti evidenciju plaćanja članarina, rezerviranih knjiga, posuđenih knjiga te zakašnjelih knjiga.

2. KORIŠTENE TEHNOLOGIJE I ALATI

Prilikom izrade ove aplikacije korišteni skriptni jezici su *HTML*, *JavaScript* te *CSS*, a alati koji su potrebni za ispravan rad aplikacije su *Firebase* i *Material Design*. Funkcija i primjena navedenih tehnologija i alata detaljnije su opisane u nastavku.

2.1. Firebase

Firebase predstavlja platformu za izradu softverskih rješenja, ponajviše mobilnih i web aplikacija. Platforma je razvijena od strane *James*-a *Tamplin*-a i *Andrew*-a *Lee*-a kao nova posebna inačica *Envolve*[1] platforme, a danas pripada *Google*-u. *Firebase* je s vremenom postao široko rasprostranjena platforma koja nudi programska rješenja za Android i iOS aplikacije, Web ili C++ aplikacije te ima mogućnost jednostavnog povezivanja s *Unity* projektima. Prilikom izrade aplikacija *Firebase* nudi više različitih usluga kao što su *„Cloud Firestore“*, *„ML Kit“*, *„Cloud Functions“*, *„Authentication“*, *„Hosting“*, *„Cloud Storage“* i *„Real Time Database“*. *„Cloud Firestore“* je fleksibilna baza podataka koja se koristi za mobilne, web i računalne aplikacije te izradu poslužitelja. Usklađena je s *NoSQL* modelom podataka gdje se pojedinom polju pridružuje vrijednost određenog tipa (tekst, broj, objekt i slično). Omogućuje dohvaćanje i postavljanje pojedinog podatka na bazu, ali moguće je i ostvariti sinkronizaciju podataka između strane poslužitelja i strane klijenta pomoću *„real-time listener-a“*. *„ML Kit“* predstavlja mobilni *SDK*[2] (engl. S*oftware development kit*) koji omogućava strojno učenje (prepoznavanje teksta, detekcija lica, skeniranje barkoda, prevođenje teksta i slično) Android i iOS aplikacijama. *„Cloud Functions“* je usluga za pohranjivanje *JavaScript* i *TypeScript* funkcija na *Google*-ov oblak. Takve funkcije mogu se koristiti i u izvan-mrežnom načinu rada pomoću direktnog *HTTP* zahtijeva ili pomoću događaja u samom kodu aplikacije. *„Authentication“* omogućava identifikaciju korisnika prilikom prijave u određenu aplikaciju putem lozinke, telefonskog broja, elektroničke pošte ili društvenih mreža (*Facebook*, *Twitter* i slično) kako bi se sa sigurnošću mogli pohranjivati potrebni podaci vezani uz sam korisnički račun. *„Hosting“* predstavlja uslugu koja omogućava hosting web aplikacije ili web stranice koja može biti statična ili dinamična. *„Cloud Storage“* je još jedna usluga koja omogućava pohranjivanje i preuzimanje podataka, ali ovdje su u pitanju podaci kao što su slika ili video, od strane klijenta. Posljednja usluga jest *„Real Time Database“* koja je zapravo i razlog nastanka *Firebase*-a, a ona predstavlja bazu podataka za pohranjivanje

[1] Envolve je platforma koja je omogućavala web stranicama integraciju funkcionalnosti online razgovora
[2] Pribor za razvijanje softvera

informacija koje će biti dostupne svim prijavljenim korisnicima u isto vrijeme. U ovoj aplikaciji od navedenih usluga korišteni su *„Cloud Firestore“* za pohranu podataka o knjigama koje se nalaze u knjižnici, detaljnije informacije o pojedinom korisniku, informacije o plaćenim članarinama pojedinog korisnika, trenutno rezervirane i posuđene knjige od strane pojedinog korisnika, *„Cloud Functions“* za funkcije kreiranja administratora te brisanja pojedinog korisnika, *„Authentication“* usluga prilikom kreiranja korisničkog računa i *„Hosting“* usluga. *Firebase* nudi i usluge za poboljšanje kvalitete aplikacije tako što korisniku daje uvid u performanse i stabilnost aplikacije te usluge za proširenje poslovanja na način da korisniku daje uvid u statistički povoljnija područja koja odgovaraju poslu koji aplikacija izvršava. Ranije navedene *Firebase* usluge nisu korištene u ovoj aplikaciji, ali su općenito vrlo korisne u svijetu izrade aplikacija. Također, još jedna važna usluga koju *Firebase* omogućava i korištena je u ovoj aplikaciji jest „*Admin*“ usluga. Ta usluga omogućava kreiranje administratorskog računa na klijentskoj strani koji ima posebne i proširene mogućnosti za razliku od običnog korisničkog računa. Pomoću tako kreiranog administratorskog računa povećava se sigurnost podataka te korištenje same aplikacije jer se može ograničiti mogućnost pisanja i preuzimanja podataka na ili s baze podataka samo administratorskom računu. [1] [2] [3] [4] [5] [6] [7] [8]

2.2. Javascript

Javascript, skraćeno *JS*, predstavlja široko rasprostranjeni, ali vrlo jednostavan programski jezik koji se najčešće koristi pri izradi web stranica, odnosno aplikacija. *Javascript* je skriptni jezik koji se izvršava na klijentskoj strani te se koristi za dizajniranje, odnosno određivanje kako će se web stranica ponašati ukoliko se dogodi ili pokrene određeni događaj (npr. klik na gumb). Zbog osobina koje opisuju ovaj programski jezik on predstavlja glavni alat korišten prilikom izrade ovog završnog rada. [9]

2.3. HTML

HTML (engl. Hyper Text Markup Language) je skriptni opisni jezik u kojemu je pisana većina statičkih i dinamičkih web stranica. *HTML* koristi metodu organiziranja i prikazivanja informacija u obliku teksta koje dovode do stvaranja veza. Te veze su hiperveze koje omogućuju prelazak na druge dokumente, odnosno web stranice. Na taj način korisnik stvara vlastiti put pretraživanja što web stranicu čini dinamičnijom. *HTML* se isključivo koristi za prikaz i obradu podataka, a kako bi se to izvelo koristi se oznakama (engl. *Tags*) i atributima (engl. *Atributes*).

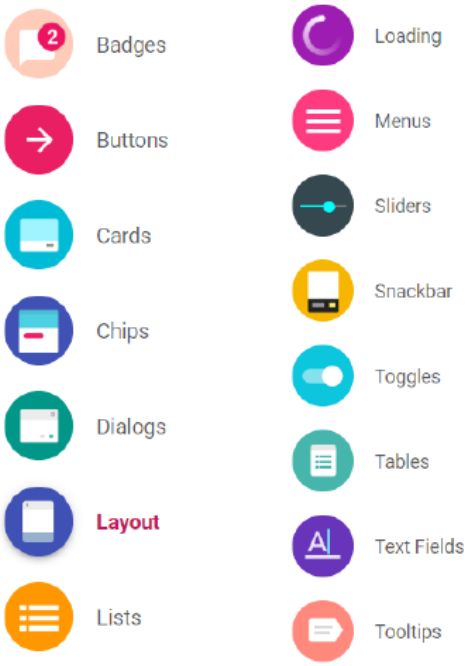
Oznake se koriste za određivanje što određeni podatak predstavlja te kako se on prikazuje. Oznake se prepoznaju putem šiljastih zagrada, a unutar tih zagrada se nalazi određeni *HTML* element. Neke od najčešće korištenih oznaka su oznake naslova (<h1>, <h2>, <h3>, itd.), oznake paragrafa ( ), prelazak u novi red (
) te oznaka sekcije ( ). Atributima se definiraju karakteristike *HTML* elementa i oni se navode u početnoj oznaci pojedinog elementa (pr. * This is left aligned *). Atribut se sastoji od imena svojstva koje se želi karakterizirati i vrijednosti koju će to svojstvo imati. Određenim načinom grupacije *HTML* elemenata i uporabom različitih atributa dolazi se do praktičnog i jednostavnog izgleda web stranice. [10]

2.4. CSS

Iako se izgled web stranice, odnosno aplikacije može postići i uporabom samo *HTML* oznaka i atributa, za detaljniji i privlačniji izgled koristi se *CSS*. *CSS* (engl. *Cascading Style Sheets*) predstavlja stilski jezik koji opisuje izgled *HTML* dokumenta, te se njime štedi vrijeme jer ima mogućnost manipulacije izgleda više različitih dokumenata u isto vrijeme. *CSS* ima široki raspon primjene, ali se najčešće koristi za način rasporeda elemenata unutar web stranice (način prikaza, margine, pozicija i slično), definiranje izgleda teksta (odabir fonta, razmak, debljina slova, boja, sjene i slično), dodavanje animacija, promjena izgleda kursora i ostalih vidljivih elemenata web stranice. Koristeći *CSS* mogućnosti definiran je ugodan i jednostavan izgled ove aplikacije te samim time povećava iskoristivost web aplikacije. [11]

2.5. Material Design Lite

Material design lite predstavlja vrstu vizualnog jezika koji koristi principe dobrog dizajna kako bi se kreirale prilagodljive i jednostavne komponente i alati koji se koriste prilikom izrade korisničkog sučelja. Suradnjom dizajnera i programera nastale su komponente otvorenog koda koje se mogu koristiti unutar aplikacije bez dodatnog korištenja *CSS* koda. Za pojedinu komponentu nudi se više različitih vrsta dizajna kako bi korisnik mogao birati po svojim stavovima i interesima. Popis komponenata koje *Material design lite* nudi mogu se pogledati na slici 2.1. Uporabom *Material design lite* komponenata smanjuje se vrijeme potrebno za dizajn aplikacije te daje veći prostor za programiranje funkcionalnosti same aplikacije. Upravo zbog takve jednostavnosti i praktičnosti za izgled glavnih vizualnih elemenata ove aplikacije korištene su komponente: gumb, dijalog, raspored, oznaka učitavanja te polje za unos teksta koje omogućava *Material design lite*. [12]



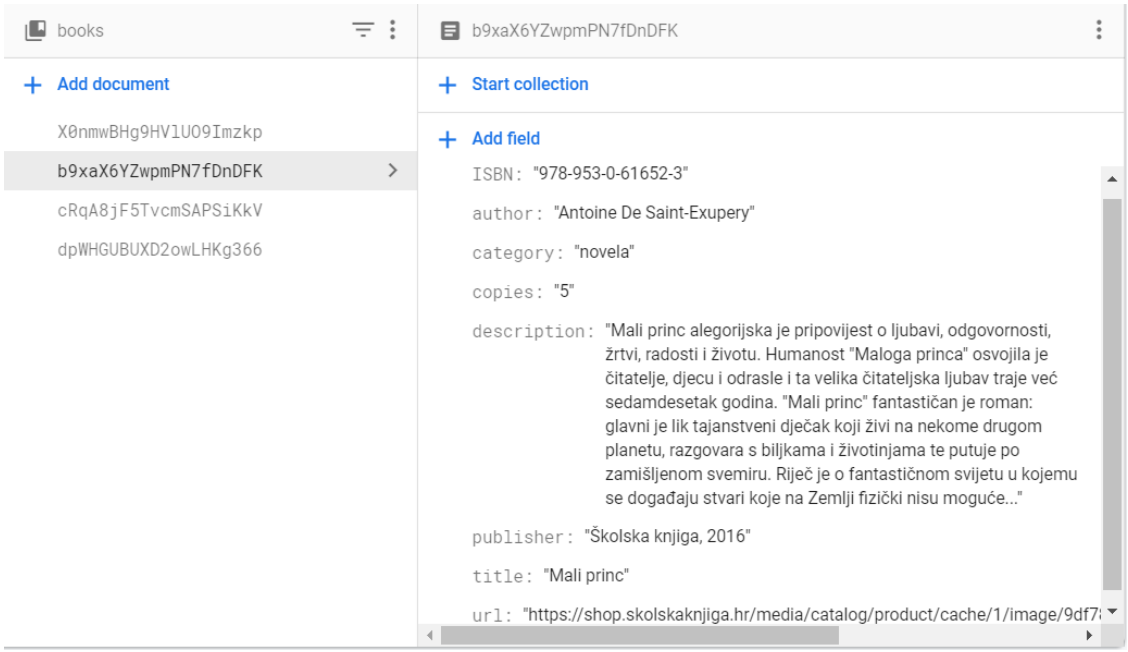
Slika 2.1. *Popis komponenata koje omogućava Material Design Lite.*[12]

3. REALIZACIJA APLIKACIJE

Ovo poglavlje daje uvid u detaljniju strukturu web aplikacije. Opisuje način na koji je baza podataka osmišljena i ostvarena te funkcija koje su zaslužne za korištenje podataka iz same baze kako bi aplikacija bila funkcionalna.

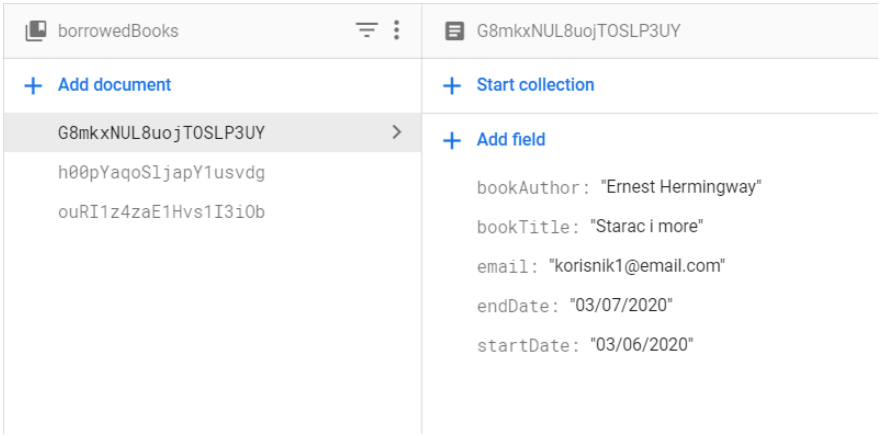
3.1. Baza podataka

Za ostvarenje baze podataka aplikacije korištena je *Firebase* usluga *„CloudFirestore“*. *„CloudFirestore“* organizira podatke u kolekcije (engl. *Collection*) koje sadržavaju dokumente vezane isključivo za tu kolekciju. Svaki dokument ima svoj jedinstveni identitet unutar kojeg se pohranjuju vrijednosti. Ova aplikacija sastoji se od ukupno pet kolekcija: *„books“*, *„borrowedBooks“*, *„membership“*, *„reservedBooks“* i *„users“*. Kolekcija *„books“* sadrži dokumente s informacijama svih knjiga u knjižnici. Informacije koje su pohranjene za svaku od knjiga su: naslov knjige, naziv autora, *URL* (engl. *Uniform Resource Locator*) naslovne stranice, nakladnik, ISBN, kategorija kojoj knjiga pripada, kratki sadržaj ili opis knjige te broj dostupnih primjeraka. Primjer dokumenta jedne knjige prikazan je na slici 3.1.



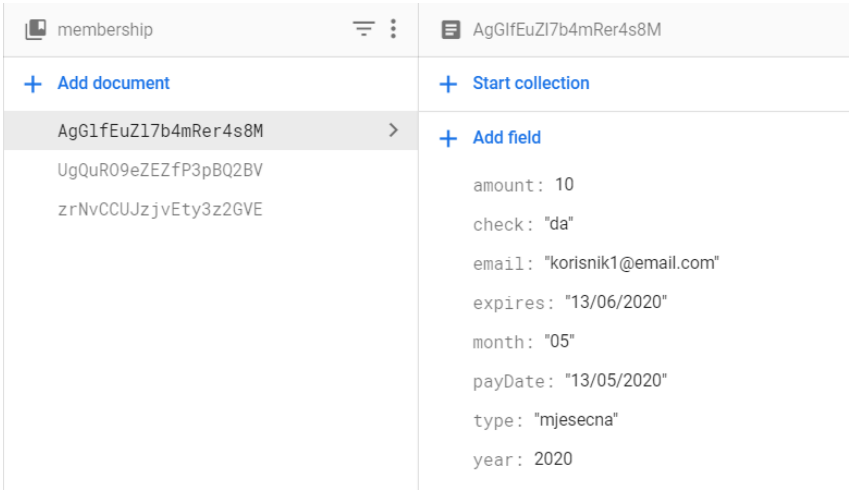
Slika 3.1. *Primjer izgleda dokumenta knjige.*

Kolekcija *„borrowedBooks“* sadrži dokumente s podacima svih trenutno posuđenih knjiga. Svaki dokument te kolekcije sadrži elektroničku poštu korisnika koji je posudio knjigu, naziv knjige, naziv autora, datum početka posudbe i datum isteka posudbe. Primjer dokumenta *„borrowedBooks“* kolekcije nalazi se na slici 3.2.



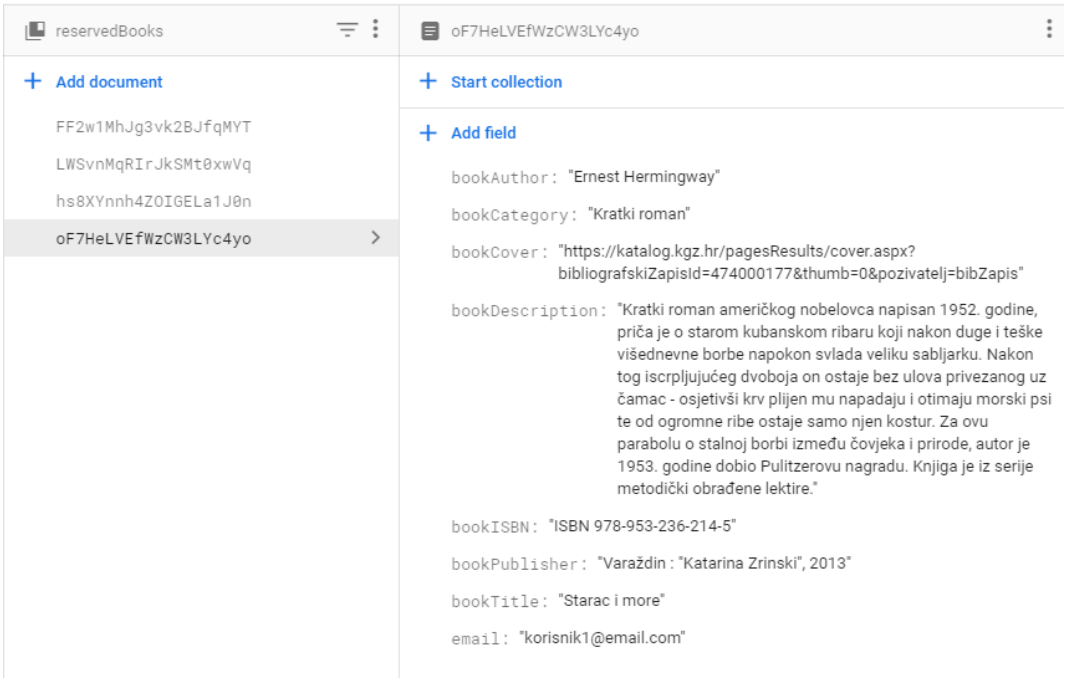
Slika 3.2. *Primjer izgleda dokumenta posuđenih knjiga.*

„Membership“ kolekcija sadrži jedinstvene dokumente s informacijama o plaćanju članarine za pojedinog korisnika. Svako novo plaćanje članarine postaje novi dokument koji sadrži informacije o elektroničkoj pošti korisnika koji je platio članarinu, vrstu članarine (mjesečna ili godišnja), iznos, datum plaćanja članarine, datum dospijeća članarine, mjesec i godina kada je plaćena članarina te podatak o tome treba li aplikacija provjeravati taj dokument ili ne. Primjer dokumenta kolekcije nalazi se na slici 3.3.



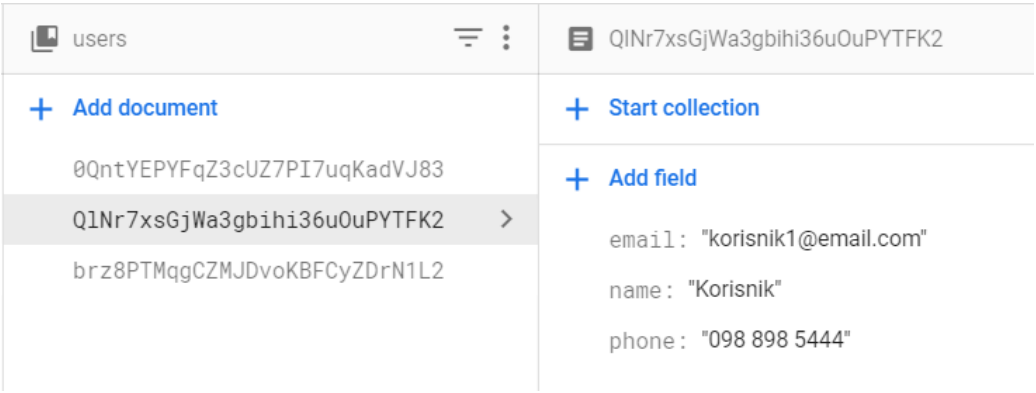
Slika 3.3. *Primjer izgleda dokumenta za praćenje članarina.*

Kolekcija *„reservedBooks“* sadrži dokumente sa svim trenutno rezerviranim knjigama od strane bilo kojeg korisnika. Svaki dokument sadrži elektroničku poštu korisnika koji je rezervirao pojedinu knjigu te sve informacije o knjizi koje se nalaze u dokumentima *„books“* kolekcije. Na slici 3.4. je prikazan izgled jednog od dokumenata.



Slika 3.4. *Primjer izgleda dokumenta rezervirane knjige.*

Posljednja kolekcija je *„users“* koja sadrži dokumente s informacijama pojedinog korisnika potrebne za njegovu identifikaciju i prijavu na aplikaciju te lakše pristupanje informacijama pohranjenim u ostale kolekcije. Svaki dokument te kolekcije sadrži korisničko ime, elektroničku poštu koja je potrebna za prijavu u aplikaciju te telefonski broj kao dodatna mogućnost komunikacije između administratora i samog korisnika. Izgled dokumenta za korisnika prikazan je na slici 3.5.



Slika 3.5. *Primjer izgleda dokumenta s informacijama korisnika.*

Kako baza podataka sadrži i vrstu povjerljivih informacija *„Cloud Firestore“* usluga omogućava i postavljanje pravila tko može čitati, a tko pisati na bazu podataka što je u ovom slučaju postavljano na administratora ili registriranog korisnika u ovisnosti o kojoj je kolekciji riječ. Dodatna sigurnost informacija o pojedinom korisniku postiže se uslugom autentifikacije koja je detaljnije opisana u nastavku.

3.2. Identifikacija korisnika

Da bi se moglo pristupiti administratorskom ili korisničkom profilu potrebno je imati korisnički račun putem kojega se korisnik prijavljuje u aplikaciju. Prilikom kreiranja korisničkog računa koristi se *Firebase* usluga koja omogućava identifikaciju korisnika. Autentifikacijska usluga omogućava provjeru identiteta korisnika koji trenutno vrši registraciju tako da šalje poruku na elektroničku poštu korisnika kako bi mogao potvrditi da je to zapravo on. Ukoliko korisnik ne potvrdi da je to on, neće moći pristupiti aplikaciji. Autentifikacijskom uslugom, također, onemogućena je registracija više korisnika s istom *e-mail* adresom te je prilikom pohrane informacija o samom korisniku nemoguće otkriti korisnikovu zaporku zbog korištenja *Firebase* metode prijave putem elektroničke pošte i zaporke. Na taj način zaporka je čvrsto vezana uz elektroničku poštu te je potpuno nevidljiva prilikom pristupa bazi podataka (vidi slika 3.6.).

Slika 3.6. *Prikaz baze podataka s informacijama za prijavu u aplikaciju.*

Programski kôd registracije korisnika može se pogledati na slici 3.7.

```
signupForm.addEventListener('submit', (e) => {
  e.preventDefault();

  const email = signupForm['signupEmail'].value;
  const password = signupForm['signupPass'].value;
  const password2 = signupForm['signupPass2'].value;

  if (password == password2) {
    firebase.auth().createUserWithEmailAndPassword(email, password).then(cred => {
      return db.collection('users').doc(cred.user.uid).set({
        name: signupForm['signupUsername'].value,
        phone: signupForm['signupPhone'].value,
        email: signupForm['signupEmail'].value
      });
    }).then(() => {
      var dialog = document.querySelector('#signUpDialog');
      dialog.close();
      signupForm.reset();
      var user = firebase.auth().currentUser;
      user.sendEmailVerification().then(function() {
        // Email sent.
      }).catch(function(error) {
        // An error happened.
      });
    });
  } else {
    alert("Lozinke nisu jednake!");
  }
});
```

Programski kôd 3.7. *Registracija korisnika.*

U slučaju da je korisnički račun uspješno kreiran, prilikom pristupa aplikaciji korisnik se mora prijaviti. Prijava se viši pomoću *Firebase* metode *„signInWithEmailAndPassword“* čija je primjena prikazana na slici 3.8.

```
signinForm.addEventListener('submit', (e) => {
  e.preventDefault();

  // get user info
  const email = signinForm['signinEmail'].value;
  const password = signinForm['signinPass'].value;
  $("#signinBtn").hide();

  // sign up the user
  firebase.auth().signInWithEmailAndPassword(email, password).catch(function (error) {
    var errorCode = error.code;
    var errorMessage = error.message;

    if (errorCode === 'auth/wrong-password') {
      alert('Lozinka nije točna');
    } else {
      alert(errorMessage);
    }
  });
  $("#signinBtn").show();
  signinForm.reset();
});
```

Programski kôd 3.8. *Prijava korisnika.*

Ukoliko je prijava korisnika u aplikaciju uspješna otvara se profil ovisno o tome jeli prijavu izvršio korisnik aplikacije ili zaposlenik knjižnice.

3.3. Programska rješenja korisničkog računa

Aplikacija omogućava dvije vrste korisničkog računa koji se razlikuju po ovlastima. Jedna vrsta korisničkog računa jest administratorski račun kojemu je omogućena manipulacija podataka aplikacije i pripada zaposleniku knjižnice, te obični korisnički račun kojim se koriste osobe koje žele koristiti usluge knjižnice. Detaljniji opis pojedine vrste korisničkog računa slijedi u nastavku.

3.1.1. Administratorski račun

Administratorski račun predstavlja vrstu korisničkog računa koji pripada zaposleniku knjižnice. Kako bi se ovakva vrsta korisničkog računa razlikovala od običnog korisničkog računa korištena je *Firebase* „*admin*“ usluga. Tom uslugom samo određenim *e-mail* adresama je dopuštena manipulacija informacija koje se nalaze u bazi podataka, te se na taj način povećava i sigurnost same baze podataka. Tako administratorski račun ima mogućnost pregleda i uređivanja informacija svih knjiga knjižnice kao i mogućnost dodavanja nove knjige, pregled i manipulacija korisnika, pregled i manipulacija informacija o posuđenim i rezerviranim knjigama, pregled zakasnina te evidencija plaćanja članarine za pojedinog korisnika. Slikom 3.9. prikazan je programski kôd koji omogućava pregled svih knjiga povezivanjem na bazu podataka i korištenjem *Firebase „real-time listener-a“*. Pomoću njega vrši se dohvaćanje svih dokumenata pohranjenih u „*books“* kolekciji organiziranih po abecednom redu u stvarnom vremenu.

```
db.collection('books').orderBy("title", "asc").onSnapshot(snapshot => {
  const bookList = document.querySelector('.books');

  let html = '';
  snapshot.docs.forEach(doc => {
    const book = doc.data();
    const div =`
    <div id ="${doc.id}" class="mdl-dialog__content" style=" box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.2);padding: 2%; margin-bottom: 20px;">
      <div class="row" style="padding-bottom: 2%;">
        <div class="col-sm-2" style:"text-align:center;"> <img id="bookCover" src="${book.cover}" ></img></div>
        <div class="lighten-4 col-sm-10 " style="font-size: 21px;font-weight: bold;">
          <div id="${book.title}">${book.title}</div>
          <div id="${book.author}">${book.author}</div></div>
        </div>
        <div id="${book.publisher}">
          Nakladnik: ${book.publisher}
        </div>
        <div>
          ISBN : ${book.ISBN}
        </div>
        <div>
          Kategorija: ${book.category}
        </div>
        <div>
          Opis: ${book.description}
        </div>
        <div>
        Dostupne kopije: ${book.copies}
      </div>
      <div>

      </div>
      <div style="text-align: end;">
        <button id="" onClick="openEditBook()" class="mdl-button mdl-js-button mdl-button--raised mdl-button--colored">Promijeni</button>
      </div>
    </div>`;
    html += div;
  });
  bookList.innerHTML = html;
});
```

Programski kôd 3.9. *Pregled svih knjiga u knjižnici.*

Samo povezivanje na bazu podataka vrši se pomoću varijable „*db*“ kojoj je pridružena vrijednost metode za pristup bazi podataka (Slika 3.10.) nakon čega je potrebno odrediti kojoj kolekciji se pristupa za čitanje ili pisanje podataka.

```
const db = firebase.firestore();
```

Programski kôd 3.10. *Povezivanje na bazu podataka.*

Za razliku od pregleda knjiga u knjižnici koje zahtjeva čitanje iz baze podataka, uređivanje informacija postojeće knjige i dodavanje nove knjige zahtijeva pisanje na bazu podataka. Kod uređivanja informacija knjige za pisanje na bazu podataka koristi se metoda *„update“* (Slika 3.11.).

```
db.collection('books').doc(doc.id).update({
  title: updateBookForm.titleEdit.value,
  author: updateBookForm.authorEdit.value,
  cover: updateBookForm.coverEdit.value,
  publisher: updateBookForm.publisherEdit.value,
  ISBN: updateBookForm.isbnEdit.value,
  category: updateBookForm.categoryEdit.value,
  description: updateBookForm.descriptionEdit.value,
  copies: updateBookForm.copiesEdit.value
}).then(() => {
  updateBookForm.reset();
  alert("Knjiga uspiješno uređena!");
  dialogEditBook.close();
  $(".modal-edit-book").hide();

}).catch(err => {
  console.log(err.message);
});
```

Programski kôd 3.11. *Uređivanje informacija knjige metodom „update“.*

Dok se kod dodavanja nove knjige metoda *„add“* (Slika 3.12.).

```
db.collection('books').add({
  title: addBookForm.title.value,
  author: addBookForm.author.value,
  cover: addBookForm.cover.value,
  publisher: addBookForm.publisher.value,
  ISBN: addBookForm.isbn.value,
  category: addBookForm.category.value,
  description: addBookForm.description.value,
  copies: addBookForm.copies.value
}).then(() => {
  addBookForm.reset();
  alert("Dodavanje nove knjige uspiješno!");
}).catch(err => {
  console.log(err.message);
});
```

Programski kôd 3.12. *Dodavanje nove knjige metodom „add“.*

Razlika između te dvije metode pisanja na bazu podataka je u tome što *„update“* metoda mijenja podatak dokumenta koji se već nalazi u bazi te je potreban *id* dokumenta kojemu se mijenjaju značajke. Dok „*add“* metoda kreira potpuno novi dokument unutar kolekcije. Osim što aplikacija nudi prikaz podataka o knjigama u knjižnici, administratorskom računu omogućava se i prikaz korisnika. Pregled svih korisnika postignut je na jednaki način kao i pregled knjiga, samo što se u ovome slučaju pristupa *„users“* kolekciji te se koriste njeni dokumenti (Slika 3.13.).

```
db.collection('users').orderBy("name", "asc").onSnapshot(snapshot => {
  const userList = document.querySelector('#userUl');
  let html = '';
  snapshot.docs.forEach(doc => {
    const users = doc.data();

    if( users.email == "admin@admin.com"){
      const div = ``;
      html += div;
    } else{
      const div = `
      <li  id= "${doc.id}">
      <div class="mdl-dialog__content users-container">
        <div  id= "${doc.id}" class="lighten-4 " style="font-size: 21px;font-weight: bold;margin-bottom: 10px;">${users.name}<br></div>
        <div id= "${users.email}" class ="userInfo">
        <span>
          <div>
           E-mail : ${users.email}
          </div>
          <div>
           Tel : ${users.phone}
          </div>
          </span>
          </div>
          <div style="text-align: end;">
          <button id="membershipBtn" onClick="showMembership()" class="mdl-button mdl-js-button mdl-button--raised mdl-button--colored">Članarina</button>
          <button id="showMore2Btn" onClick="showBorrowed()" class=" show-more-button mdl-button mdl-js-button mdl-button--raised mdl-button--colored">Posuđene knjige</button>
          <button id="showMoreBtn" onClick="showReserved()" class=" show-more-button mdl-button mdl-js-button mdl-button--raised mdl-button--colored">Zahtjevi</button>
          <button onClick="deleteUser()" class=" delete-button mdl-button mdl-js-button mdl-button--raised mdl-button--colored">Obriši</button>
          </div>
      </div>
      </li>`
      ;
      html += div;
    }
  });
  userList.innerHTML = html;
});
```

Programski kôd 3.13. *Prikaz svih korisnika*.

Kako je svaki korisnik povezan s ostalim kolekcijama u bazi podataka ovisno o tome ima li posuđenih i/ili rezerviranih knjiga te plaćene članarine, administratorski račun ima mogućnost manipulacije i tih podataka. Prikaz pojedinih informacija vezanih uz plaćene članarine, rezervirane knjige i posuđene knjige postignut je programskim kodovima jednakim prikazu knjiga i korisnika, jedina je razlika u kolekciji kojoj se pristupa. Manipulaciju podataka vezanih uz evidenciju plaćanja članarine predstavlja sama mogućnost plaćanja članarine korištenjem „*add*“ metode (Slika 3.14.).

```
const payMembershipFormAdmin = document.querySelector('#payMembershipFormAdmin');
payMembershipFormAdmin.addEventListener('submit', (e) => {
  e.preventDefault();

  var amountAdmin = "";
  changeCheckmarkAdmin();
  var todayAdmin = new Date();
  var ddAdmin = String(todayAdmin.getDate()).padStart(2, '0');
  var mmAdmin = String(todayAdmin.getMonth() + 1).padStart(2, '0');
  var yyyyAdmin = todayAdmin.getFullYear();
  dateAdmin = ddAdmin + '/' + mmAdmin + '/' + yyyyAdmin;

  var type = document.querySelector('input[name = option-admin]:checked').value;
  if(type == "mjesecna"){
     amountAdmin = 10;
     var todayAdmin = new Date();
     var ddExpiresAdmin = String(todayAdmin.getDate()).padStart(2, '0');
     var mmExpiresAdmin = String(todayAdmin.getMonth() + 2).padStart(2, '0');
     var yyyyExpiresAdmin = todayAdmin.getFullYear();
     expiresDateAdmin = ddExpiresAdmin + '/' + mmExpiresAdmin + '/' + yyyyExpiresAdmin;
  }else {
     amountAdmin = 120;
     var expiresDateAdmin = new Date();
     var ddExpiresAdmin = String(expiresDateAdmin.getDate()).padStart(2, '0');
     var mmExpiresAdmin = String(expiresDateAdmin.getMonth() + 1).padStart(2, '0');
     var yyyyExpiresAdmin = String(expiresDateAdmin.getFullYear() + 1)
     expiresDateAdmin = ddExpiresAdmin + '/' + mmExpiresAdmin + '/' + yyyyExpiresAdmin;
  }

  db.collection('membership').add({
    type: type,
    month: mmAdmin,
    payDate: dateAdmin,
    amount: amountAdmin,
    year: yyyyAdmin,
    email: userEmail,
    expires: expiresDateAdmin,
    check: "da"
  }).then(() => {
    payMembershipForm.reset();
    closePayMembershipAdmin();
    alert("Članarina uspiješno plaćena!");
    $("#modal-MembershipMsgAdmin").hide();
    document.getElementById("payMembershipBtn").disabled = true;
  }).catch(err => {
    console.log(err.message);
  });
});
```

Programski kôd 3.14. *Plaćanje članarine putem administratorskog računa*.

Plaćanje članarine moguće je na dva načina, mjesečno ili godišnje, te je plaćanje dopušteno samo ukoliko je članarina prethodnog mjeseca ili godine istekla. Ta, provjera vrši se pomoću funkcije *„checkMembershipAdmin()“* prikazane na slici 3.15.

```
function checkMembershipAdmin(){
  const email = userEmail;
  var today = new Date();
  var dd = String(today.getDate()).padStart(2, '0');
  var mm = String(today.getMonth() + 1).padStart(2, '0');
  var yyyy = today.getFullYear();

  db.collection('membership').onSnapshot(snapshot => {
     snapshot.docs.forEach(doc => {
      const membership = doc.data();
      datePayed = membership.payDate;
      arrPayed = datePayed.split("/");
      dateExpires = membership.expires;
      arrExpires = dateExpires.split("/");
      if(membership.email == email){
        if(membership.check == "da"){
          if(membership.type == "mjesecna"){
            if(arrExpires[2] < yyyy){
              console.log("mjesecna-godina")
              document.getElementById("payMembershipBtn").disabled = false;
              $("#modal-MembershipMsgAdmin").show();
            }else if(arrExpires[1] == mm){
              if(arrPayed[0] < dd){
              console.log("mjesecna-dan")
              document.getElementById("payMembershipBtn").disabled = false;
              $("#modal-MembershipMsgAdmin").show();
              }else{
                console.log("mjesecna-else")
                document.getElementById("payMembershipBtn").disabled = true;
              }
              }else if (arrExpires[1] < mm){
                console.log("mjesecna-mjesec")
                document.getElementById("payMembershipBtn").disabled = false;
              $("#modal-MembershipMsgAdmin").show();
              }
          }else{
            if(arrExpires[2] < yyyy){
              console.log("godišnja-godina")
              document.getElementById("payMembershipBtn").disabled = false;
              $("#modal-MembershipMsgAdmin").show();
            }else if(arrExpires[1] == mm){
              if(arrPayed[0] < dd){
                console.log("god-dan")
                document.getElementById("payMembershipBtn").disabled = false;
              $("#modal-MembershipMsgAdmin").show();
              }
              }else if (arrExpires[1] < mm){
                console.log("god-mjesec")
                document.getElementById("payMembershipBtn").disabled = false;
              $("#modal-MembershipMsgAdmin").show();
              }
           }
           }else{
            document.getElementById("payMembershipBtn").disabled = true;
           }
        }
      });
```

Programski kôd 3.15. *Sadržaj funkcije „checkMembershipAdmin()“* .

Funkcija *„checkMembershipAdmin()“* provjerava jeli je korisnikov dokument označen (*„check“*, Slika 3.3.) s „*da*“ ili „*ne*“. Ukoliko je dokument označen s „*da*“ znači da je to zadnje plaćena članarina tog korisnika te se vrši usporedba današnjeg datuma s datumom isteka članarine, dok u suprotnom ne. Prilikom svakog novog plaćanja članarine u dokument se pohranjuje *„check: da“*, te se prethodni dokument mijenja u *„check: ne“*. Funkcija koja to omogućava naziva se *„changeCheckmarkAdmin()“*, te je njen sadržaj prikazan na slici 3.16.

```
function changeCheckmarkAdmin(){
  db.collection('membership').get().then(snapshot => {
    snapshot.docs.forEach(doc => {
      const membership = doc.data();
      if(membership.email == userEmail){
        if(membership.check == "da"){
          db.collection('membership').doc(doc.id).update({
            check: "ne"
          });
        }
      }
    });
  });
}
```

Programski kôd 3.16. *Sadržaj funkcije „changeCheckmarkAdmin()“.*

Prilikom pregleda posuđenih knjiga pojedinog korisnika, administratorski račun ima mogućnost označavanja da je knjiga vraćena i pregled zakasnine za pojedinu knjigu. Vraćanje knjige postiže se metodom *„delete“* koja se koristi nad točno određenim dokumentom kolekcije *„borrowedBooks“* (Slika 3.17.).

```
function returnBook() {
 let id = event.target.parentElement.parentElement.getAttribute("id");
 db.collection("borrowedBooks").doc(id).delete();
 alert("Knjiga je vraćena!");
 $("#modal-additional-info").hide();
 dialogAdditionalInfo.close();
}
```

Programski kôd 3.17. *Sadržaj funkcije za vraćanje knjiga.*

Dok se obavijest o zakašnjenju vraćanja knjige javlja tek ukoliko je datum isteka posudbe prošao te je određeno da zakasnina iznosi dvije kune po zakašnjelom danu. Provjera datuma vrši se svaki puta pri pokretanju koda za pregled posuđenih knjiga te se „*update*“ metodom, ukoliko je datum isteka prošao, u određeni dokument postavlja koliko dana korisnik kasni s vraćanjem knjige (Slika 3.18.).

```
function checkIfLateAdmin() {
  db.collection('borrowedBooks').orderBy("bookTitle", "asc").onSnapshot(snapshot => {
    snapshot.docs.forEach(doc => {
     const borrowedBook = doc.data();

     var today = new Date();
     var dd = String(today.getDate()).padStart(2, '0');
     var mm = String(today.getMonth() + 1).padStart(2, '0');
     var yyyy = today.getFullYear();
     var date = yyyy + '-' + mm + '-' + dd;

     var dateStart = borrowedBook.startDate;
     arrStart = dateStart.split("/");
     dateEnd = borrowedBook.endDate;
     arrEnd = dateEnd.split("/");
     var end = arrEnd[2] + '-' + arrEnd[1] + '-' + arrEnd[0];
     const timeDiff  = (new Date(date)) - (new Date(end));
     var days = timeDiff / (1000 * 60 * 60 * 24);
     var late = days*2;
     if(borrowedBook.email == userEmailInfo){
       if(arrEnd[2] < yyyy){
         db.collection('borrowedBooks').doc(doc.id).update({
         late: late
         });
     } else if (arrEnd[1] <= mm ){
       if(arrEnd[0] < dd){
         db.collection('borrowedBooks').doc(doc.id).update({
         late: late
         });
       }else{
         db.collection('borrowedBooks').doc(doc.id).update({
         late: late
         });
       }

     }
     }
   });
 });
}
```

Programski kôd 3.18. *Sadržaj funkcije za provjeru zakasnine.*

Prilikom pregleda trenutno rezerviranih knjiga pojedinog korisnika, administratorski račun ima mogućnost potvrde posudbe određene knjige. Potvrdom posudbe knjiga koja se prethodno nalazila u dokumentu kolekcije *„reservedBooks“* postaje novi dokument u kolekciji *„borrowedBooks“* te se dokument kolekcije *„reservedBooks“* briše. Funkcija za potvrdu posudbe prikazana je na slici 3.19.

```
function confirmBorrow(){
  let userEmail = event.target.parentElement.parentElement.children[2].getAttribute("id");
  let borrowedBookTitile = event.target.parentElement.parentElement.children[0].getAttribute("id");
  let borrowedBookAuthor = event.target.parentElement.parentElement.children[1].getAttribute("id");
  let reservedBookId = event.target.parentElement.parentElement.getAttribute("id");

  var today = new Date();
  var dd = String(today.getDate()).padStart(2, '0');
  var mm = String(today.getMonth() + 1).padStart(2, '0');
  var yyyy = today.getFullYear();
  startDate = dd + '/' + mm + '/' + yyyy;

  endDate = new Date(today.setDate(today.getDate() + 30))
  var ddEnd = String(today.getDate()).padStart(2, '0');
  var mmEnd = String(today.getMonth() + 1).padStart(2, '0');
  var yyyyEnd = today.getFullYear();
  endDate = ddEnd + '/' + mmEnd + '/' + yyyyEnd;

  db.collection("borrowedBooks").add({
    email: userEmail ,
    bookTitle: borrowedBookTitile,
    bookAuthor: borrowedBookAuthor,
    startDate: startDate,
    endDate: endDate
})
.then(function() {
    alert("Posudba knjige uspiješna!");
    $("#modal-additional-info").hide();
})
.catch(function(error) {
    console.error("Error writing document: ", error);
});
db.collection("reservedBooks").doc(reservedBookId).delete();
dialogAdditionalInfo.close();
}
```

Programski kôd 3.19. *Sadržaj funkcije za potvrdu posudbe knjige.*

I naposljetku administratorski račun ima mogućnost brisanja samih korisnika što se postiže korištenjem *„delete“* metode nad dokumentom kolekcije *„users“* te korištenjem *Firebase „Cloud Functions“* usluge (Slika 3.20.).

```
function deleteUser() {
  var id = event.target.parentElement.parentElement.children[0].getAttribute("id");
  var userEmail = event.target.parentElement.parentElement.children[1].getAttribute("id");

  const userDelete = functions.httpsCallable('deleteUser');
  userDelete({ email: userEmail }).then(result => {
    alert("Korisnik je uspiješno obrisan !");
  });
  db.collection("users").doc(id).delete();
}
```

Programski kôd 3.20. *Sadržaj funkcije za brisanje korisnika.*

Pomoću *„Cloud Functions“* usluge dopušteno je brisanje korisnika u potpunosti iz aplikacije samo korisničkom računu koji ima oznaku „*admin*“, te se na taj način briše sav trag o postojanju tog računa. Sadržaj *„Cloud Functions“* funkcije za brisanje korisnika nalazi se na slici 3.21.

```
exports.deleteUser = functions.https.onCall((data, context) => {
    return admin.auth().getUserByEmail(data.email).then(user => {
        const uid = user.uid;
        admin.auth().deleteUser(uid).then(function() {
    console.log('Successfully deleted user');
  })
  .catch(function(error) {
    console.log('Error deleting user:', error);
  });
    });
});
```

Programski kôd 3.21. *Sadržaj funkcije za brisanje korisnika pomoću „Cloud Functions“ usluge.*

3.1.2. Korisnički račun

Korisnički račun predstavlja sve profile korisnika koji nemaju oznaku „*admin*“, te za razliku od administratorskog računa imaju manje ovlasti. Korisnički račun ima mogućnost pregleda knjiga u knjižnici, mogućnost rezervacije pojedine knjige, pregled rezerviranih i posuđenih knjiga, pregled mogućih zakasnina te evidencija o plaćanju članarine i sama mogućnost plaćanja. Prikaz knjiga na korisničkom računu vrši se istim kodom kao i na strani administratorskog računa (Slika 3.9.). Jedina je razlika u izgledu prikaza jer se prilikom pregleda knjiga nudi mogućnost rezervacije pojedine knjige. Sadržaj funkcije za rezervaciju knjige nalazi se na slici 3.22. Određenu knjigu moguće je rezervirati samo ako korisnik ima manje od pet rezerviranih i posuđenih knjiga, te ukoliko ima dostupnih primjeraka. Prilikom svake rezervacije broj dostupnih primjeraka smanjuje se za jedan, što se *„update“* metodom postavlja u odgovarajući dokument *„books“* kolekcije. Sama rezervacija postiže se korištenjem metode *„add“* pomoću koje se kreira novi dokument u *„reservedBooks“* kolekciji.

```
function borrowBook() {
  let borrowTitle = event.target.parentElement.parentElement.children[0].children[1].children[0].getAttribute("id");
  let borrowAuthor = event.target.parentElement.parentElement.children[0].children[1].children[1].getAttribute("id");

  const uEmail = firebase.auth().currentUser.email;

  db.collection('books').get().then(snapshot => {
     snapshot.docs.forEach(doc => {
      const book = doc.data();

      if(book.title == borrowTitle){
        var nula = 0;
        if(book.copies == nula){
          alert("Nije moguće posuditi knjigu jer nema slobodnih kopija.");
        }else{
          const cop = book.copies - 1;
          var bookId = doc.id;
          db.collection('books').doc(bookId).update({
            copies : cop
          }).then(function(){
            db.collection("reservedBooks").add({
              email: uEmail ,
              bookTitle: borrowTitle,
              bookAuthor: borrowAuthor,
              bookPublisher: book.publisher,
              bookISBN: book.ISBN,
              bookCategory: book.category,
              bookDescription: book.description,
              bookCover: book.cover,
            }).then(function() {
                  alert("Knjiga je rezervirana za posudbu. Molimo dođite u knjižnicu kako biste je preuzeli. Hvala!");
                  location.reload();
              })
              .catch(function(error) {
                  console.error("Error writing document: ", error);
              });
            });
          }

        }
      });
  });
  };
```

Programski kôd 3.22. *Sadržaj funkcije za rezervaciju knjige.*

Za prikaz rezerviranih i posuđenih knjige te plaćenih članarina, također, se koristi jednaka metoda kao i u administratorskom dijelu za navedene akcije, ali je u ovom slučaju razlika u načinu dohvaćanja *e-mail* adrese korisnika za koju se vrši čitanje podataka te u samom prikazu podataka. Na korisničkom računu za dohvaćanje *e-mail* adrese koristi se *Firebase* metoda *„currentUser“* koja predstavlja trenutno prijavljenog korisnika (Slika 3.23.) dok se u administratorskom računu to postizalo pješke iz pročitanih podataka prilikom prikaza svih korisnika („*userEmail*“, Slika 3.19).

```
firebase.auth().currentUser.email;
```

Programski kôd 3.23. *Dohvaćanje e-mail adrese korisnika.*

Nadalje, prilikom pregleda posuđenih knjiga nudi se i mogućnost pregleda zakasnina, što je također jednako metodi iz administratorskog dijela. Isto vrijedi i za evidenciju plaćenih članarina gdje se nudi i sama mogućnost plaćanja članarine. Prilikom prikaza rezerviranih knjiga, korisničkom računu nudi se mogućnost odustajanja o rezervacije (Slika 3.24.). Odustajanjem od rezervacije broj dostupnih primjeraka knjige povećava se za jedan te se *„update“* metodom pohranjuje u odgovarajući dokument *„books“* kolekcije, te se dokument te rezervirane knjige briše iz *„reservedBooks“* kolekcije.

```
function cancelReservation(){
  let bookId = event.target.parentElement.parentElement.getAttribute("id");
  let bookTitleReserved = event.target.parentElement.parentElement.children[0].children[1].children[0].getAttribute("id");

  db.collection('books').get().then(snapshot => {
    snapshot.docs.forEach(doc => {
     const book = doc.data();

     if(book.title == bookTitleReserved){
      const cop = book.copies + 1;

      db.collection('books').doc(doc.id).update({
          copies : cop
       })
       }
   });
 });
db.collection("reservedBooks").doc(bookId).delete();
}
```

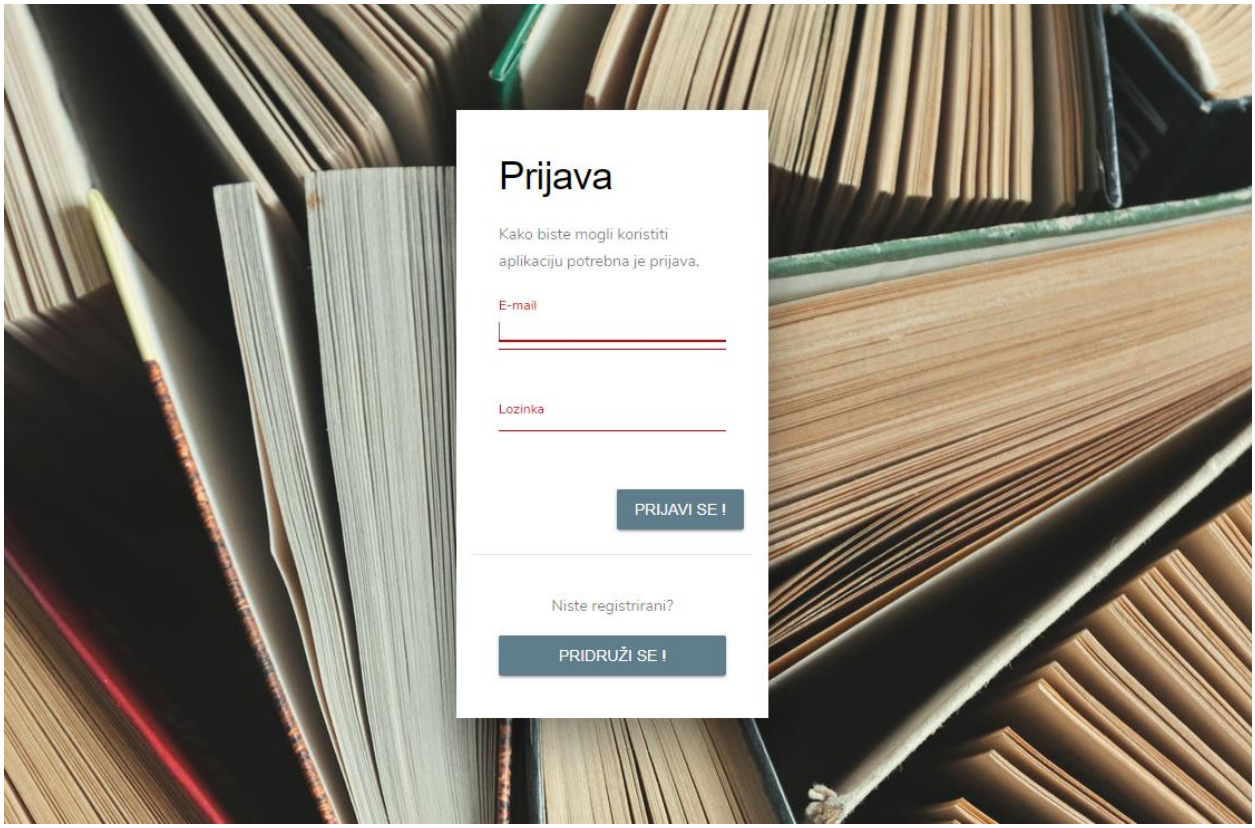
Programski kôd 3.24. *Funkcija odustajanja od rezervacije.*

4. IZGLED KORISNIČKOG SUČELJA

Korisničko sučelje ove aplikacije izrađeno je korištenjem ranije spomenutih alata: *HTML*-a, *CSS*-a i *Material Design*-a. Izgled korisničkog sučelja aplikacije može se podijeliti u tri grupe ovisno o funkciji koju obavlja i korisničkom računu kojem pripada: početni zaslon, odnosno prijava, administratorski profil i korisnički profil.

4.1. Početni zaslon

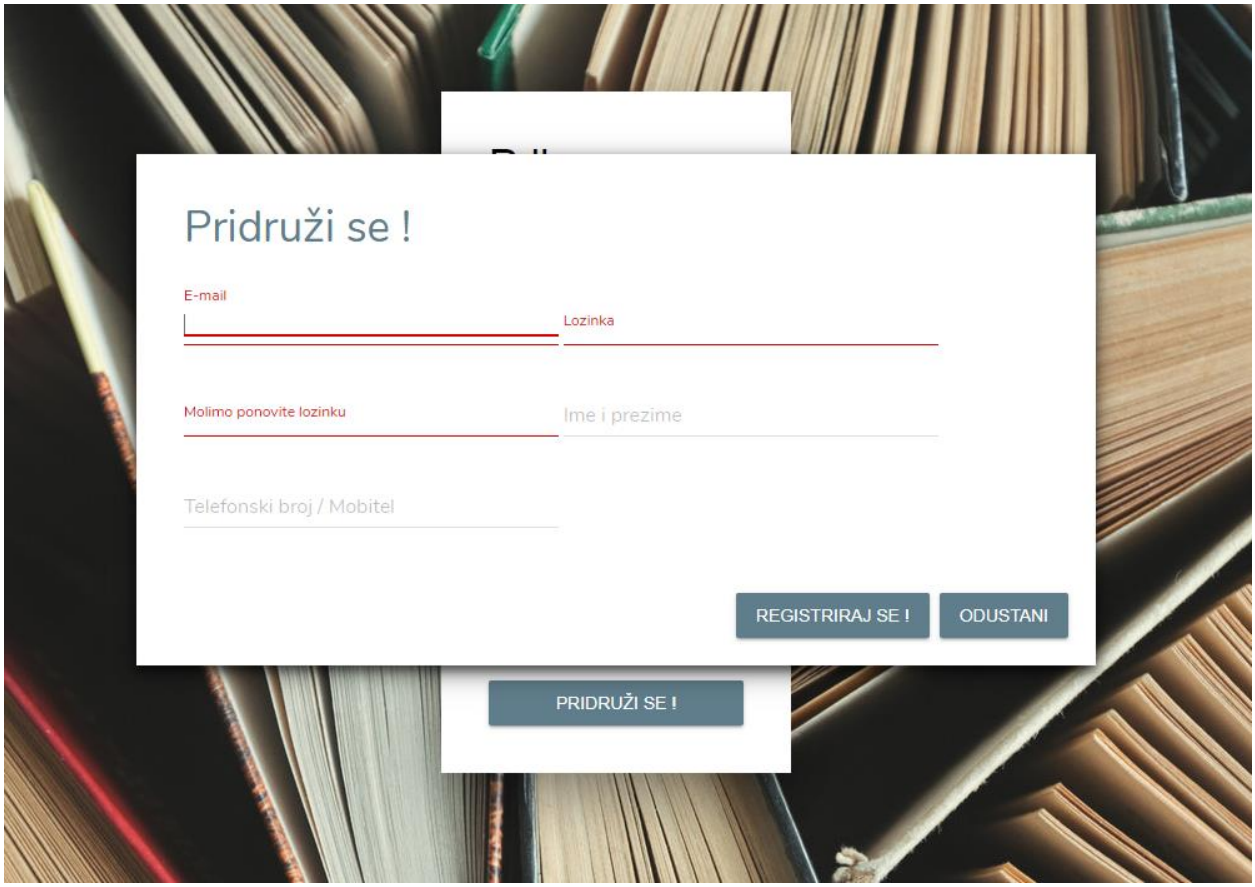
Prilikom pokretanja aplikacije otvara se početni zaslon, koji ujedno predstavlja prijavu korisnika u aplikaciju. Izgled početnog zaslona prikazan je na slici 4.1.



Slika 4.1. *Početni zaslon aplikacije.*

Dizajn početnog zaslona vrlo je jednostavan, a sastoji se od dva polja za unos teksta i dva gumba. Klikom na gumb „*Pridruži se*“ otvara se novi dijalog koji predstavlja formu za registraciju korisnika. Sadržaj i izgled forme nalazi se na slici 4.2. Klikom na gumb „*Registriraj se*“ kreira se novi korisnički račun ako su unesene tražene vrijednosti ispravne. Kako bi se korisnik mogao u

potpunosti prijaviti u aplikaciju potrebno je preko elektroničke pošte izvršiti potvrdu *e-mail* adrese, odnosno identifikaciju korisnika.

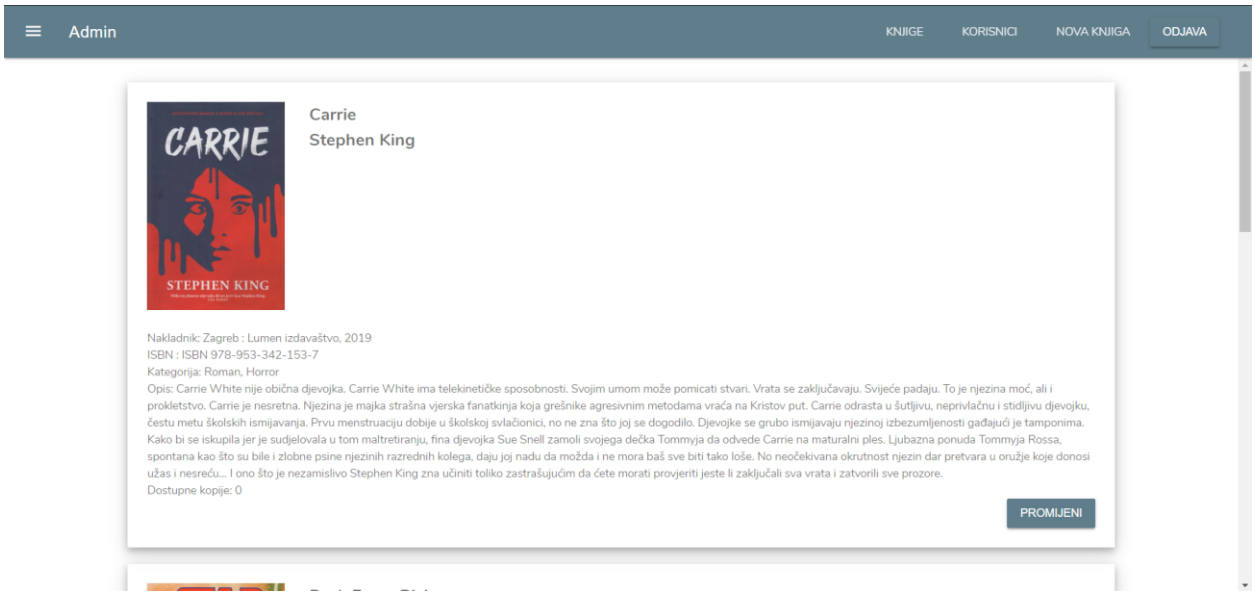


Slika 4.2. *Forma za registraciju korisnika.*

No, na početnom zaslonu prilikom klika na gumb „*Prijavi se*“, ukoliko su unesene potrebne i valjane vrijednosti (*e-mail* adresa i lozinka) otvara se, ovisno o vrsti korisničkog računa, početni zaslon administratorskog ili korisničkog profila.

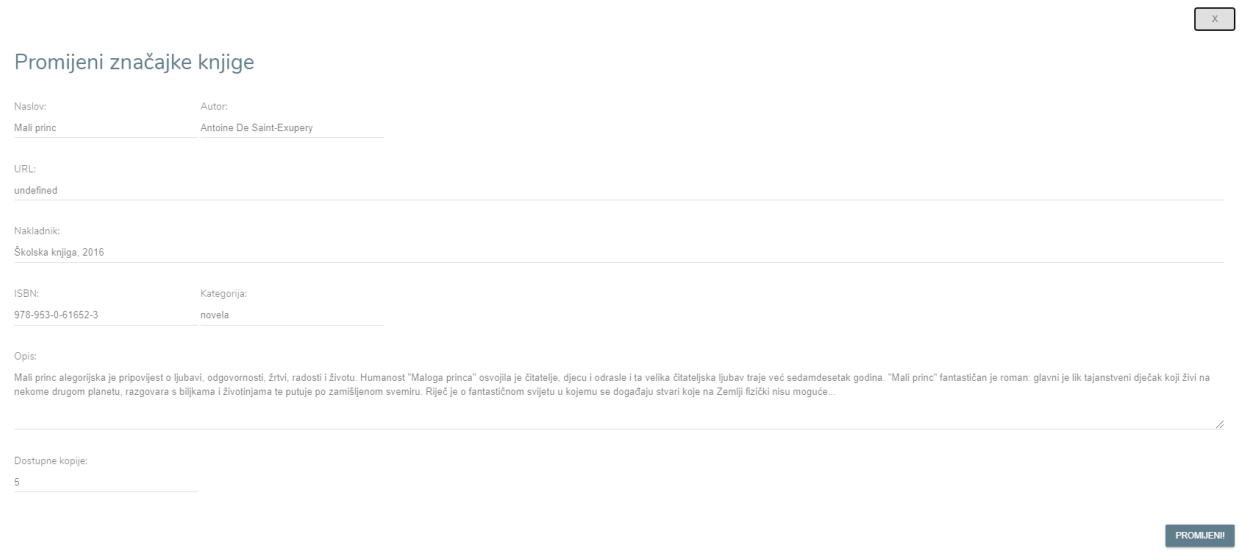
4.2. Administratorski profil

Prilikom otvaranja administratorskog profila pojavljuje se zaslon prikazan na slici 4.3. Taj zaslon ujedno predstavlja početni zaslon administratorskog profila i zaslon prikaza svih knjiga u knjižnici. Izgled svakog zaslona u administratorskom profilu sastoji se od navigacijske trake i dijela prikaza podataka. Navigacijska traka sastoji se od polja za prikaz imena korisnika, tri navigacijska gumba koji mijenjaju podatke ovisno o tome što se traži: „*Knjige*“, „*Korisnici*“ i „*Nova knjiga*“ te gumba za odjavu iz aplikacije.



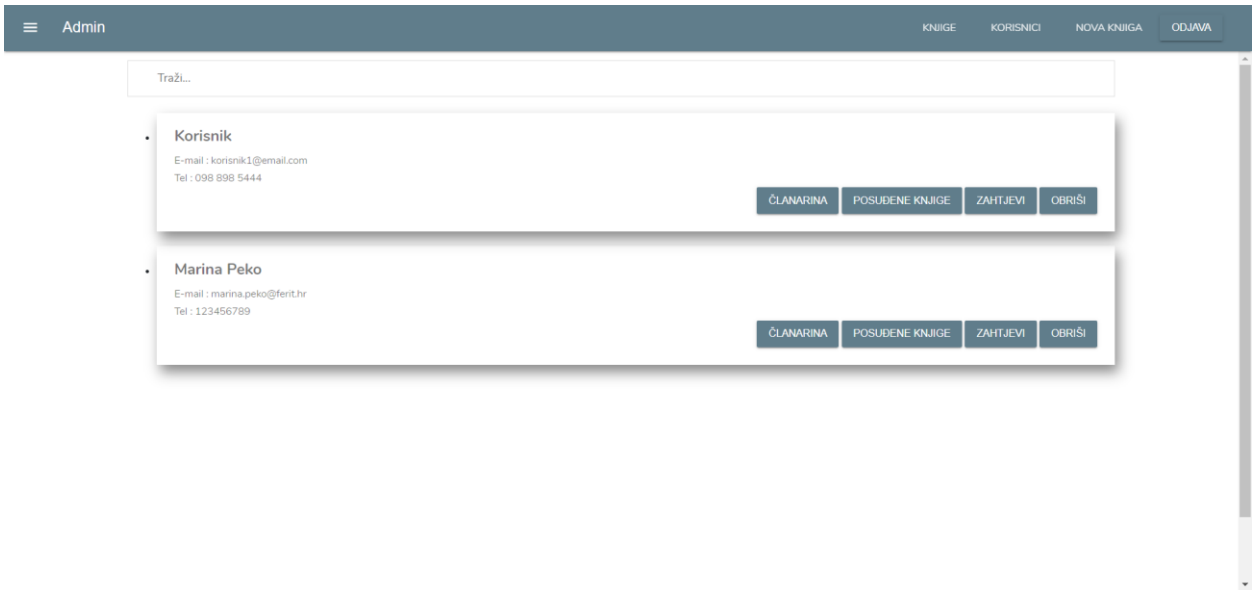
Slika 4.3. *Administratorski profil: prikaz knjiga u knjižnici.*

Prikaz podataka zaslona knjiga u knjižnici sastoji se slike naslovne stranice knjige, naziva knjige, naziva autora knjige, nakladnika, ISBN-a, kategorije, kratkog opisa, dostupnih primjeraka i gumba za promjenu informacija za svaku pojedinu knjigu u knjižnici. Prilikom klika na gumb „*Promijeni*“ otvara se posebni dijalog koji omogućuje promjenu informacija o pojedinoj knjizi (Slika 4.4.).



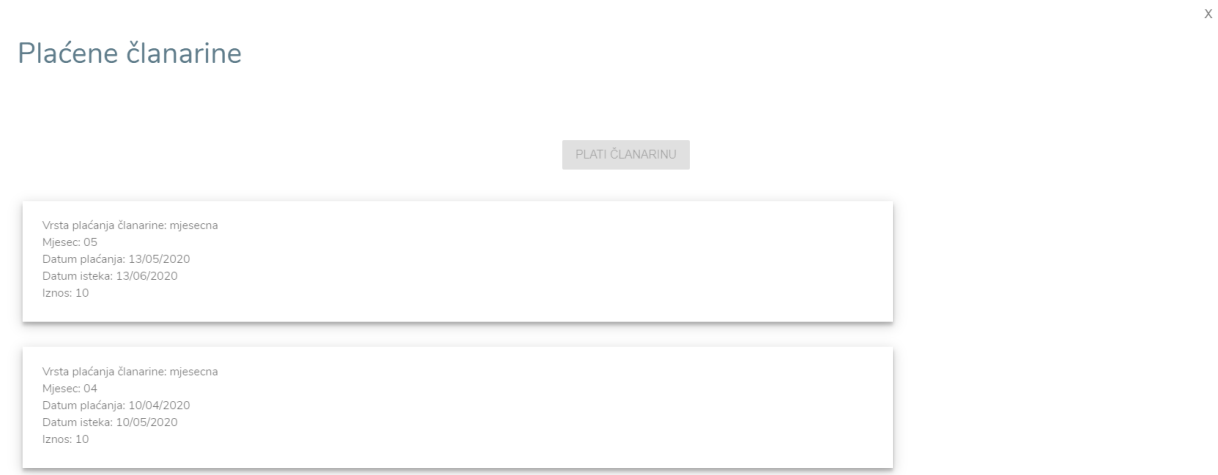
Slika 4.4. *Administratorski profil: promjena informacija o knjizi.*

Prateći navigacijske gumbe, sljedeći je zaslon prikaza svih korisnika knjižnice. Dizajn prikaza zaslona korisnika nalazi se na slici 4.5.



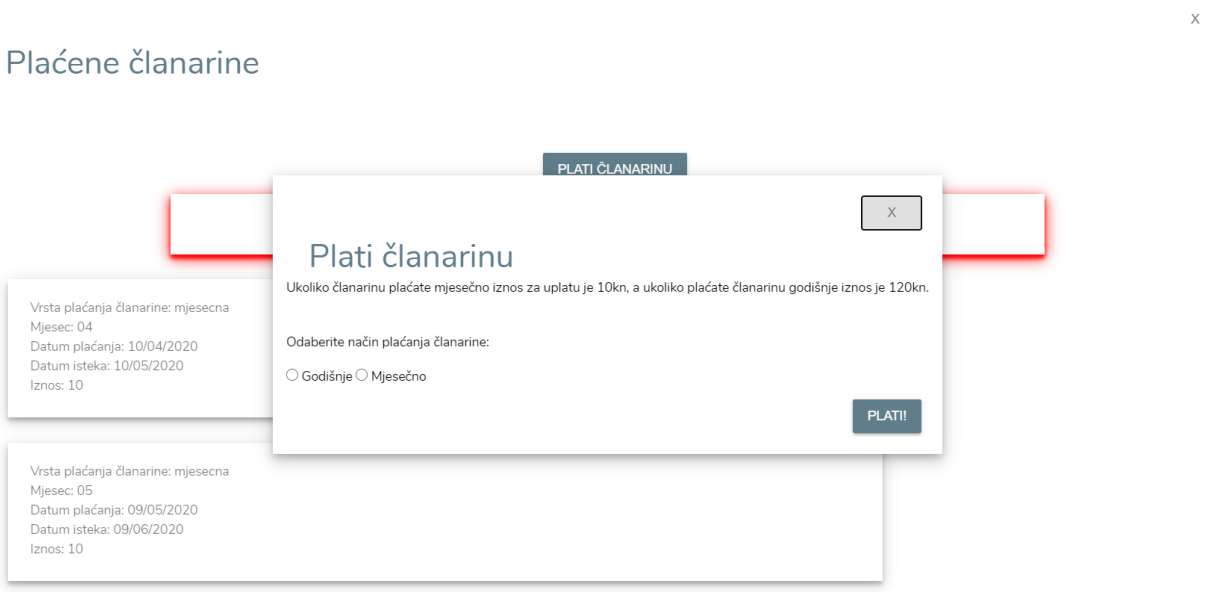
Slika 4.5. *Administratorski profil: prikaz svih korisnika.*

Dizajn dijaloga prikaza informacija pojedinog korisnika sastoji se od imena korisnika, korisnikove *e-mail* adrese, korisnikovog telefonskog broja te četiri gumba koji nude detaljnije informacije o aktivnosti korisnika unutar aplikacije. Klikom na prvi gumb („*Članarina*“) otvara se dijalog koji nudi uvid u evidenciju plaćanja članarine odabranog korisnika (Slika 4.6.).



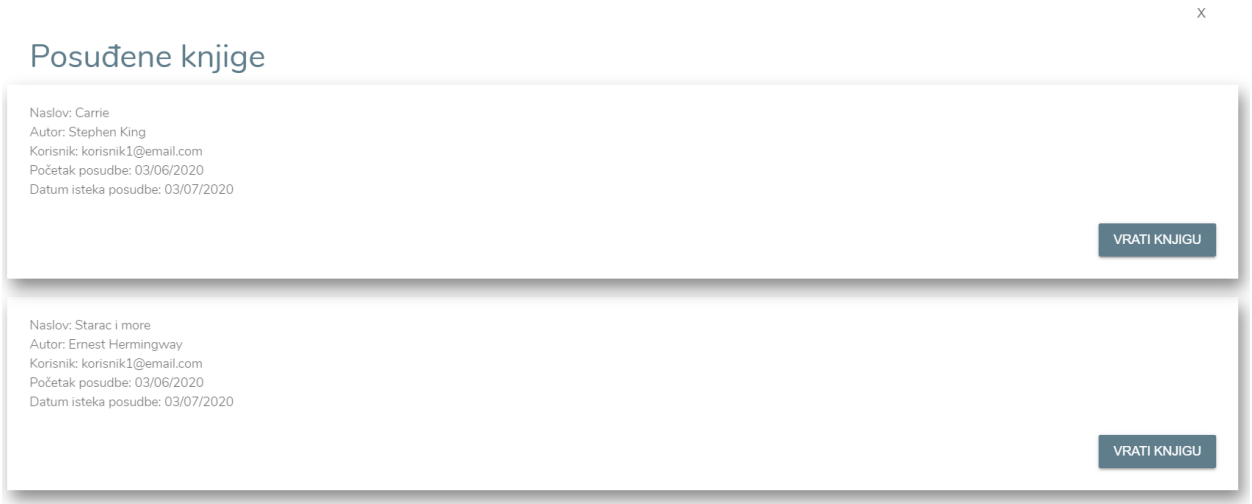
Slika 4.6. *Administratorski profil: evidencija plaćanja članarine.*

Kako je ranije rečeno, gumb „*Plati članarinu*“ postaje aktivan ukoliko je članarina posljednjeg mjeseca ili godine istekla. Klikom na gumb otvara se zaslon za plaćanje članarine čiji se sadržaj vidi na slici 4.7.



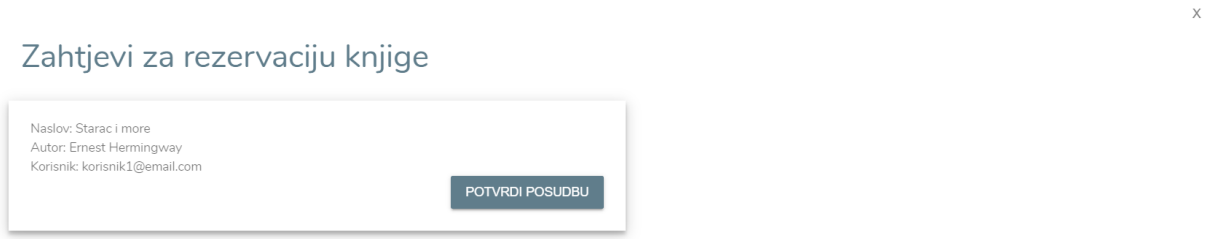
Slika 4.7. *Administratorski profil: plaćanje članarine.*

Klikom na gumb „*Posuđene knjige*“ administrator ima uvid u trenutno posuđene knjige pojedinog korisnika kao i uvid u moguće zakasnine, što je prikazano slikom 4.8.



Slika 4.8. *Administratorski profil: prikaz posuđenih knjiga.*

Klikom na gumb „*Zahtjevi*“ administrator ima mogućnost pregleda rezerviranih knjiga korisnika te potvrde posudbe ukoliko je korisnik došao posuditi knjigu. Slika 4.9. prikazuje dijalog prikaza rezerviranih knjiga.



Slika 4.9. *Administratorski profil: rezervirane knjige.*

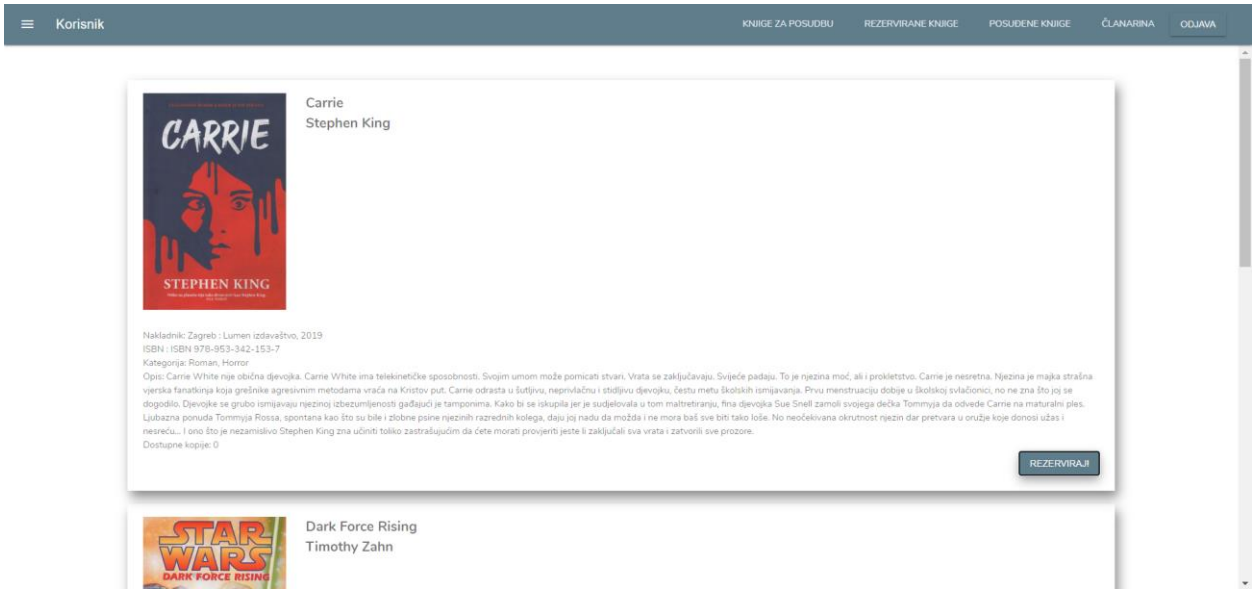
I naposljetku, administrator ima mogućnost brisanja korisnika što je moguće klikom na gumb „*Obriši*“. Osim što administrator može vidjeti i urediti informacije već postojeće knjige, on ima i mogućnost dodavanja potpuno nove knjige. Prikaz forme za dodavanje nove knjige postiže se klikom na posljednji navigacijski gumb „*Nova knjiga*“. Izgled forme prikazan je na slici 4.10.

Slika 4.10. *Administratorski profil: dodavanje nove knjige.*

4.3. Korisnički profil

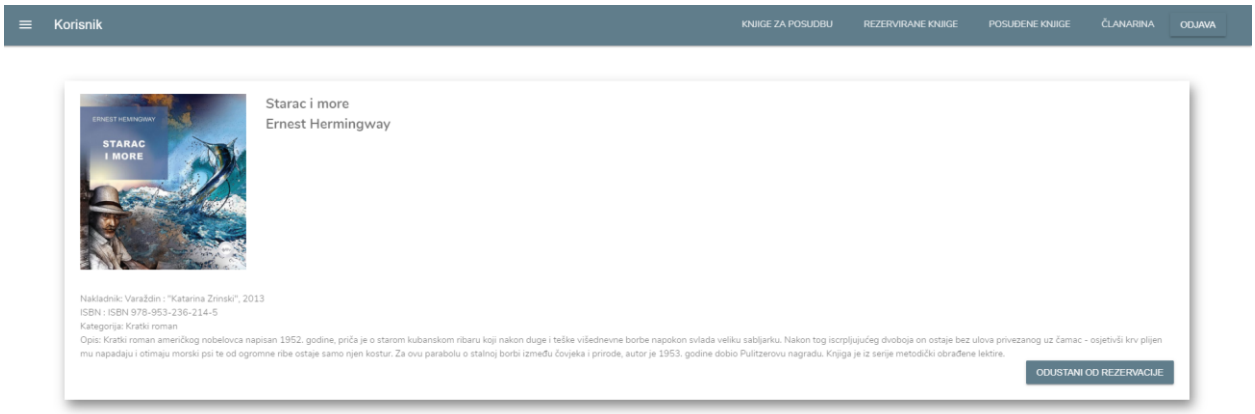
Slično kao i kod administratorskog profila, prilikom prijave na korisnički profil pojavljuje se zaslon s popisom knjiga u knjižnici (Slika 4.11.). Razlika je u sadržaju navigacijske trake koja sada sadrži četiri navigacijska gumba: „*Knjige*“, „*Rezervirane knjige*“, „*Posuđene knjige*“ i

„*Članarina*“. Kod prikaza informacija pojedine knjige, jedina je razlika u gumbu „*Rezerviraj*“ koji korisniku omogućuje rezervaciju određene knjige.



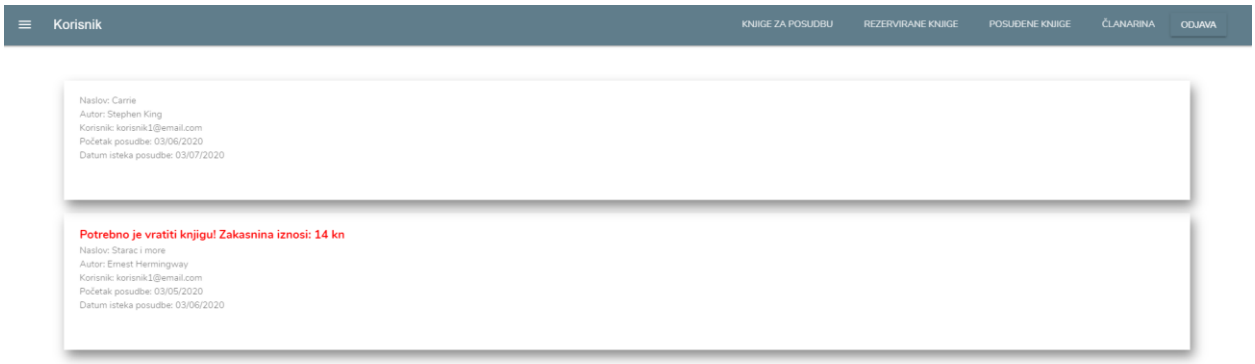
Slika 4.11. *Korisnički profil: prikaz knjiga.*

Slijedeći navigacijski gumb omogućuje korisniku pregled svih rezerviranih knjiga (Slika 4.12.) te prilikom pregleda korisnik ima mogućnost odustajanja od rezervacije.



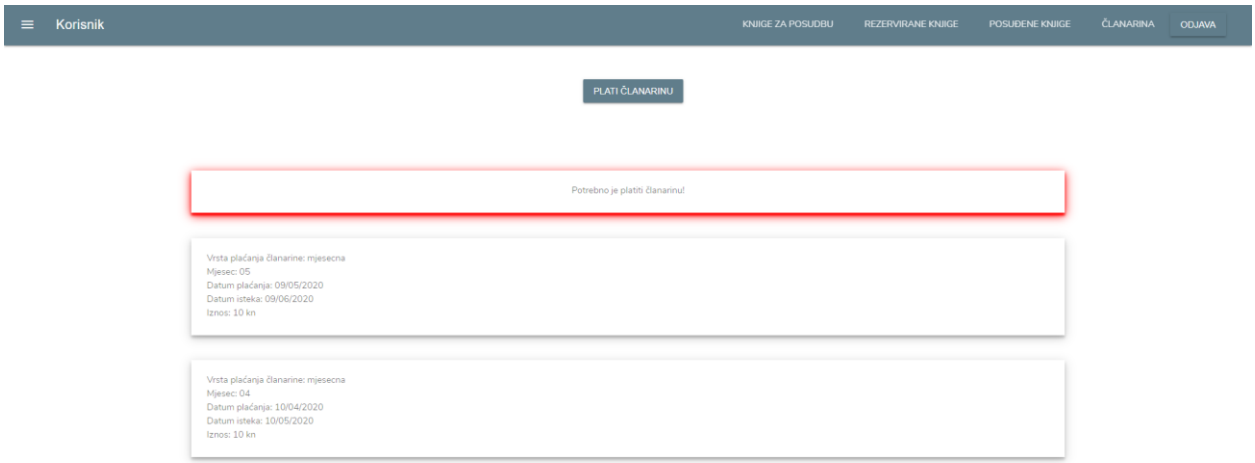
Slika 4.12. *Korisnički profil: prikaz rezerviranih knjiga.*

Klikom na gumb „*Posuđene knjige*“ korisnik ima uvid u trenutno posuđene knjige kao i pregled mogućih zakasnina što je prikazano slikom 4.13.



Slika 4.13. *Korisnički profil: prikaz posuđenih knjiga.*

I naposljetku svaki korisnik ima uvid u evidenciju plaćenih članarina klikom na gumb „*Članarina*“. Dizajn zaslona evidencije članarina za korisnički profil (Slika 4.14.) kao i dizajn dijaloga koji se otvara prilikom klika na gumb „*Plati članarinu*“ (Slika 4.7.) jednaki su onima na administratorskom profilu.



Slika 4.14. *Korisnički profil: evidencija članarina.*

5. ZAKLJUČAK

Rad knjižnice nije ni malo jednostavan posao. Knjižnice sadrže mnogo podataka koji se moraju evidentirati i koji ne smiju biti izgubljeni. Sve funkcije za održavanje rada knjižnice uglavnom mora obavljati administrator. Neke od tih funkcija nalaze se i u ovoj aplikaciji, poput mijenjanja i brisanje podataka o korisniku, unos podataka nove knjige u bazu podataka kao i uređivanje postojećih knjiga, vođenje evidencije o plaćanju godišnje ili mjesečne članarine, plaćanje članarine te praćenje trenutno posuđenih i rezerviranih knjiga od strane pojedinog korisnika. No, prilikom izrade aplikacije bilo je potrebno omogućiti i korisniku jednostavnije rukovanje posudbom knjiga. Funkcije koje korisnik može obavljati putem aplikacije su pregled dostupnih knjiga u knjižnici, mogućnost rezervacije knjige, pregled rezerviranih knjiga, odustajanje od rezervacije, pregled posuđenih knjiga te pregled uplaćenih članarina i mogućnost plaćanja. Putem aplikacije korisnik, također, dobiva obavijesti ukoliko je određenu knjigu potrebno vratiti ili ako je potrebno platiti članarinu. Iako zasloni za prikaz pojedinih podataka izgledaju slično na administratorskom i korisničkom dijelu, kodovi koje omogućuju funkcionalnost su različiti i moraju biti dobro definirani. Zato su za izradu aplikacije najbitnije *Firebase* usluge i *JavaScript* programski jezik, dok su *HTML*, *CSS* i *Material design lite* korišteni za dizajn korisničkog sučelja. Aplikacija je osmišljena tako da se administratorima olakša rad korištenjem samo jedne aplikacije za sve potrebne funkcionalnosti, a korisniku približi sadržaj knjižnice i omogući način provjere ispravnosti podataka vezanih uz njihovo korištenje usluga knjižnice.

LITERATURA

[1] Google Developers, Firebase: Products, 2020, dostupno na: https://firebase.google.com/products

[2] Google Developers, Firebase: Docs: Guides: Cloud Firestore, 2020, dostupno na: https://firebase.google.com/docs/firestore

[3] Google Developers, Firebase: Docs: Guides: ML Kit for Firebase, 2020, dostupno na: https://firebase.google.com/docs/ml-kit

[4] Google Developers, Firebase: Docs: Guides: Cloud Functions for Firebase, 2020, dostupno na: https://firebase.google.com/docs/functions

[5] Google Developers, Firebase: Docs: Guides: Firebase Authentication, 2020, dostupno na: https://firebase.google.com/docs/auth

[6] Google Developers, Firebase: Docs: Guides: Firebase Hosting, 2020, dostupno na: https://firebase.google.com/docs/hosting

[7] Google Developers, Firebase: Docs: Guides: Cloud Storage, 2020, dostupno na: https://firebase.google.com/docs/storage

[8] Google Developers, Firebase: Docs: Guides: Firebase Realtime Database, 2020, dostupno na: https://firebase.google.com/docs/database

[9] Mozilla and individual contributors, MDN web docs: Javascript, 2005-2020, dostupno na: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[10] HTML.com, Learn HTML Code, Tags & CSS: HTML For Beginners The Easy Way: Start Learning HTML & CSS Today, 2015-2020, dostupno na: https://html.com/#What_is_HTML

[11] w3schools.com, CSS Tutorial, 1999-2020, dostupno na: https://www.w3schools.com/css/default.asp

[12] Material Design Lite, Components, dostupno na: https://getmdl.io/components/index.html

SAŽETAK

Naslov: Aplikacija za organizaciju rada knjižnice

U ovom završnom radu razvijena je web aplikacija za organizaciju rada knjižnice. Bilo je potrebno napraviti dva računa koja se razlikuju po ovlastima: administratorski račun i korisnički račun. Administratorski račun pripada zaposleniku knjižnice te ima mogućnost pregleda svih knjiga u knjižnici, mijenjanje informacija o pojedinoj knjizi, mogućnost dodavanja nove knjige te pregled i manipulacija podataka vezanih uz korisnika i njegovu aktivnost unutar aplikacije. Korisnički račun pripada osobi koja želi koristiti usluge knjižnice. Mogućnosti korisničkog računa su pregled knjiga u knjižnici, mogućnost rezervacije pojedine knjige, pregled rezerviranih i trenutno posuđenih knjiga te uvid u evidenciju plaćanja članarine i plaćanje mjesečne ili godišnje članarine. Korištene tehnologije u radu opisane su u drugom poglavlju, funkcionalnost aplikacije objašnjena je u trećem, a izgled korisničkog sučelja prikazan je u četvrtom poglavlju.

Ključne riječi: administrator, *Firebase, Javascript*, knjiga, knjižnica, korisnik

ABSTRACT

Title: Application for the library business organization

In this final paper, a web application for the library business organization has been developed. It was required to create two accounts that differ in authority: the administrator account and the user account. The administrator account belongs to the library employee and it has the ability to review all books in the library repertoire, the ability to change a specific book data, the ability to add a new book and the ability to review and manipulate data related to the user and their activity within the application. The user account belongs to the person who wants to use library services. The user account capabilities are the ability to review books in library repertoire, the ability to reserve a specific book, the ability to review all reserved and currently borrowed books, the ability to insight into membership fee payment data, and the ability to pay monthly or annual membership fees. The technologies used in this final paper are described in the second chapter, the application functionality is explained in the third chapter and the user interface is presented in the fourth chapter.

Key words: administrator, book, Firebase, JavaScript, Library, user

ŽIVOTOPIS

Bernarda Špoljarić rođena je 16. listopada 1997. godine u Novoj Gradiški. Završava Osnovnu školu Matija Gubec u Cerniku, te nakon toga upisuje opću gimnaziju Nova Gradiška. Nakon završetka srednje škole, 2016. godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, stručni smjer Informatika.