Web aplikacija za pomoć u organizaciji čitanja lektire

Hudolin, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Prijediplomski sveučilišni studij

WEB APLIKACIJA ZA POMOĆ U ORGANIZACIJI ČITANJA LEKTIRE

Završni rad

Josip Hudolin

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Josip Hudolin
Studij, smjer:	Računalno inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4497, 27.07.2020.
OIB Pristupnika:	34595245109
Mentor:	prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za pomoć u organizaciji čitanja lektire
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Opisati mogućnosti web aplikacije koja olakšava organizaciju prilikom čitanja školeske lektire i vođenje bilješki o pročitanom djelu. Obrazložiti dizajn baze podataka koju će koristiti web aplikacija. Potrebno je omogućiti registraciju korisnika pri čemu se upisuje razred koji korisnik pogađa zbog dohvaćanja popisa obvezne lektire preko API-ja. Omogućiti registriranom korisniku dodavanje na popis pročitanog djela ili
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	07.09.2023.
Datum potvrde ocjene od strane Odbora:	

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 08.09.2023.

Ime i prezime studenta:	Josip Hudolin
Studij:	Računalno inženjerstvo
Mat. br. studenta, godina upisa:	R4497, 27.07.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za pomoć u organizaciji čitanja lektire**

izrađen pod vodstvom mentora prof. dr. sc. Krešimir Nenadić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Motivacija je ključni čimbenik koji potiče razvoj bilo kojeg projekta, uključujući i web aplikacije. Kada je riječ o vođenju bilješki o čitanju lektire, motivacija igra posebno važnu ulogu. Suočeni s obvezama školskog kurikuluma i mnogobrojnim knjigama koje učenici dobiju u zadatak pročitati, učenici su nerijetko suočeni s izazovom zadržavanja informacija o onome što je pročitano. Upravo je iz tog razloga važno imati alat koji će ima olakšati proces vođenja bilješki i pružati strukturirani način organizacije informacija. Web aplikacija za vođenje bilješki o čitanju lektire upravo to omogućava. Cilj je ove aplikacije olakšati učenicima i studentima proces čitanja, razumijevanja i analize lektire. Pomaže korisnicima da bilježe ključne teme, likove, citate i svoje misli tijekom čitanja, čime se stvara cjelovit pregled literature koju su proučavali. Kroz ovu web aplikaciju, želja je potaknuti korisnike da razviju svoje analitičke vještine, kritičko razmišljanje i dublje razumijevanje književnih djela. Cilj izrade je pružanje korisnicima alata koji će im olakšati školski rad, ali i potaknuti ljubav prema čitanju i književnosti.

U ovome radu se nalaze tri glavna poglavlja: Web aplikacije za vođenje bilješki, Postupak izrade aplikacije i Prikaz rada aplikacije. U poglavlju Web aplikacije za vođenje bilješki uspoređena je aplikacija „Leptira“ s pet bliskih aplikacija. U Postupku izrade aplikacije detaljno je opisano korištenje različitih programskih jezika i biblioteka čiji je rezultat upravo aplikacija „Leptira“. U Prikazu rada aplikacije vidljiv je način rada i izgled same aplikacije.

1.1. Zadatak rada

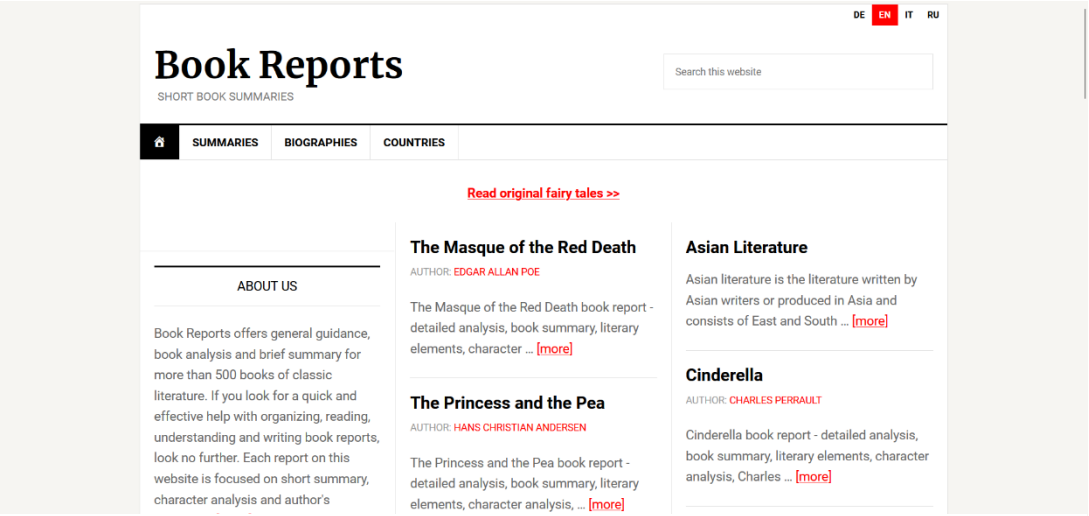
Opisati mogućnosti web aplikacije koja olakšava organizaciju prilikom čitanja školske lektire i vođenje bilješki o pročitanom djelu. Obrazložiti dizajn baze podataka koju će koristiti web aplikacija. Potrebno je omogućiti registraciju korisnika pri čemu se upisuje razred koji korisnik pogađa zbog dohvaćanja popisa obvezne lektire preko API-ja. Omogućiti registriranom korisniku dodavanje na popis pročitanog djela ili djela u postupku čitanja uz mogućnost vođenja nekih važnijih bilješki o samome djelu. Potrebno je omogućiti registriranom korisniku dodavanje djela koje nije na popisu lektire. Omogućiti korisnicima mogućnost promjene osobnih podataka zbog upisa u viši razred. Opisati tehnologije i postupak izrade web aplikacije.

2. PREGLED SLIČNIH RJEŠENJA

Glavna ideja same web aplikacije nije unikatna. Glavna funkcionalnost aplikacije je pohranjivanje podataka u bazu podataka i njeno pregledavanje. Kako bi se pregledala slična rješenja potrebno je pronaći aplikacije sličnih funkcionalnosti ili slične tematike. Aplikacije trebaju imati funkcionalnosti spremanja i čitanja bilješki ili se baviti temama o književnosti ili lektirama za osnovne i srednje škole. Pronađene aplikacije sličnih tematika ili funkcionalnosti su: Book Reports, Lektire.hr, Goodreads, Bookreporter i Sparknotes

2.1. Web aplikacija Book Reports

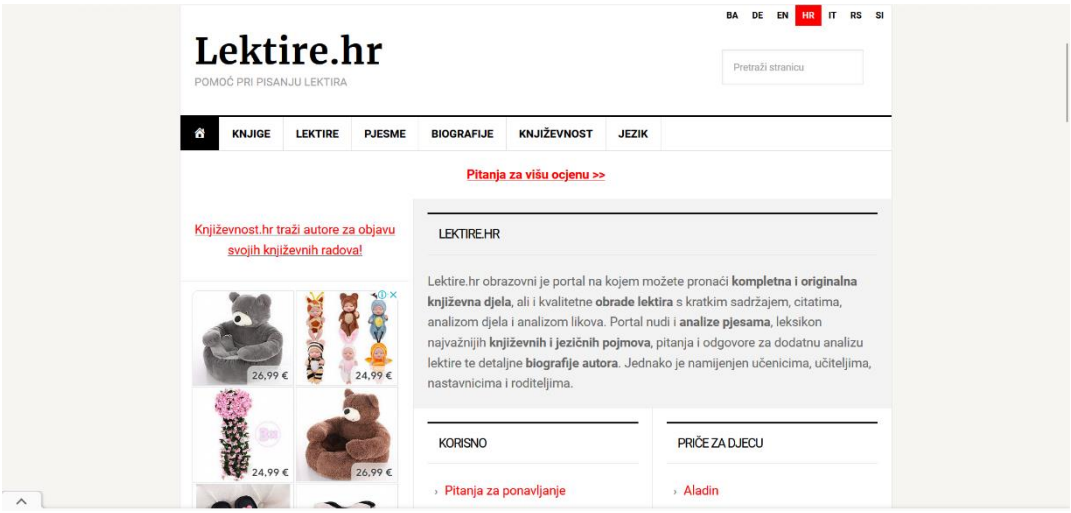
Aplikacija Book Reports [1] omogućuje korisnicima pregled već napisanih analiza lektira koje mogu pomoći pri izradi vlastite analize. Veliki je nedostatak ove aplikacije nemogućnost čitanja na hrvatskome jeziku (moguće je samo na ruskom, njemačkom, engleskom i talijanskom jeziku) te nisu obrađena djela iz hrvatske književnosti. Izgled je korisničkog sučelja aplikacije Book Reports prikazan na slici 2.1.



Slika 2.1. *Web aplikacija Book Reports*

2.2. Web aplikacija Lektire.hr

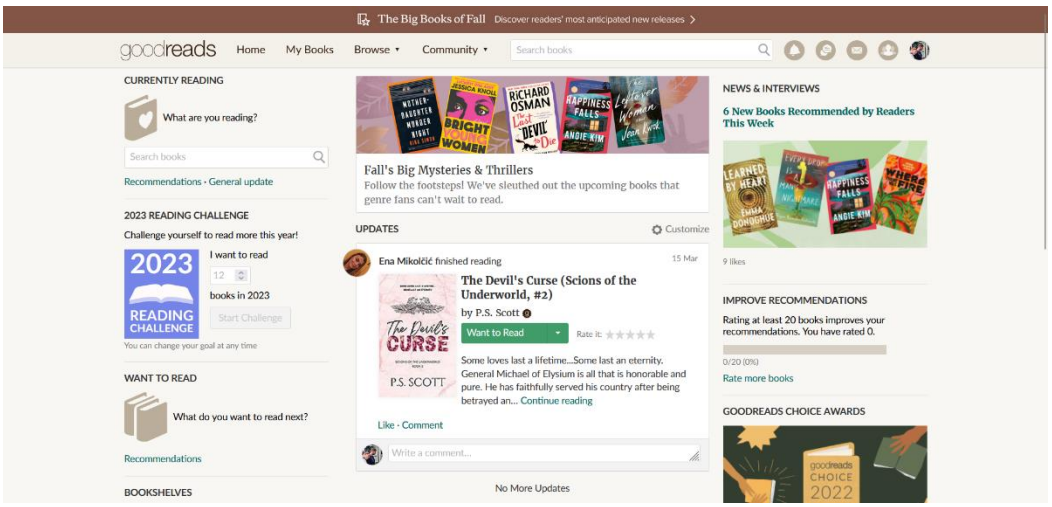
Aplikacija Lektire.hr [2] hrvatska je inačica aplikacije Book Reports. Baza je podataka ove aplikacije mnogo siromašnija, no nudi većinu djela koja su potrebna učenicima hrvatskih škola. Ne postoji nikakva mogućnost unosa vlastitih analiza niti mogućnost registracije korisnika. Dizajn web aplikacije Lektire.hr prikazan je na slici 2.2. te se da zaključiti kako su dizajni vrlo slični.



Slika 2.2. *Web aplikacija Lektire.hr*

2.3. Web aplikacija Goodreads

Goodreads [3] omogućuje praćenje vlastitog čitanja djela, ali kroz način dodavanja knjiga na listu želja, na listu knjiga koje su trenutno u stanju čitanja i pročitanih knjiga. Aplikacija također nudi i usluge društvene mreže tako što se mogu dodavati recenzije i komentari na djela. Ukoliko korisnik želi pročitati knjigu koju je pronašao na stranici, postoji i mogućnost kupovine, no ne preko aplikacije, nego aplikacija sadrži poveznice na Amazon koji prodaje određene knjige. Izgled korisničkog sučelja prikazan na slici 2.3. mnogo je jednostavniji korisnici za upotrebu nego što je sučelje aplikacije Lektire.hr.



Slika 2.3. *Web aplikacija Goodreads*

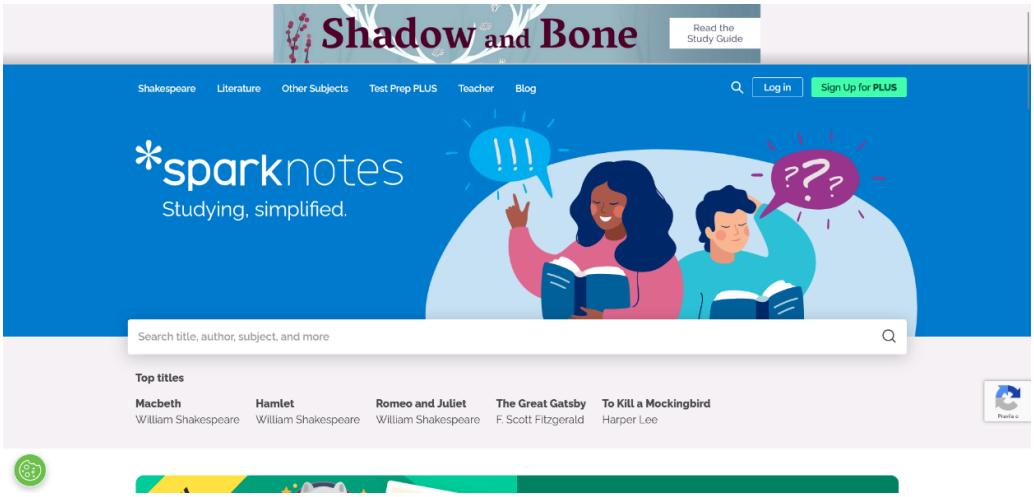
2.4. Web aplikacija Bookreporter

Aplikacija Bookreporter [4] nudi različite recenzije na veliki broj knjiga, autora, obavještava o izlascima novih knjiga, ali je također na engleskom jeziku te ne nudi veliki broj djela koji su na službenim popisima lektira. Izgled korisničkog sučelja prikazan na slici 2.4. jednostavnijeg je dizajna od aplikacije Goodreads, ali je također jednostavna za korištenje.

Slika 2.4. *Web aplikacija Bookreporter*

2.5. Web aplikacija Sparknotes

Sparknotes [5] nudi alat koji može poslužiti pripremanju ispita i provjeri osobnog znanja iz različitih školskih predmeta kao što su biologija, kemija, matematika i slično. Ne nudi mogućnost spremanja svojih bilješki i zapisa te nije specijalizirana za pripremanje lektira. Sučelje web aplikacije prikazano je na slici 2.5.



Slika 2.5. *Web aplikacija Sparknotes*

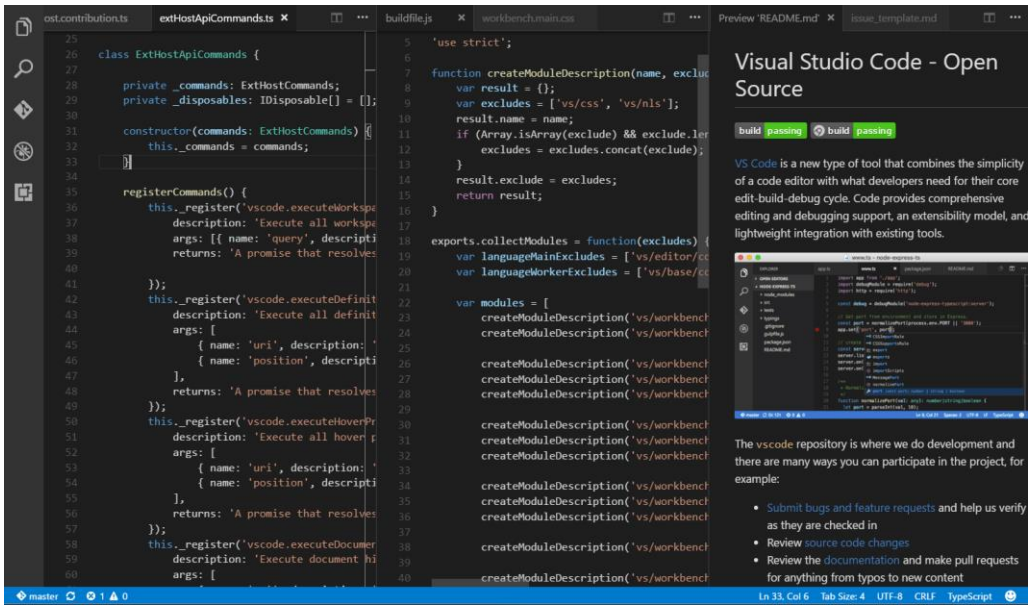
Kako bi se sjedinilo najbolje iz navedenih aplikacija, sačinjena je nova aplikacija Leptira koja pruža mogućnost poboljšati i olakšati način čitanja lektira i pripremanje istih za daljnja ispitivanja na nastavi u osnovnim i srednjim školama.

3. POSTUPAK IZRADE APLIKACIJE

Kako bi se napravila funkcionalna web aplikacija, potrebno je poznavati više programskih jezika i njihovih biblioteka. Za potrebe izrade aplikacije ovoga rada korišten je programski jezik JavaScript koji je najpopularniji za izradu web aplikacija zbog svoje sposobnosti lake interakcije s HTML-om (engl. *HyperText Markup Language*) i CSS-om (engl. *Cascading Style Sheets*), i njegov okvir (engl. *framework*) React te Express okvir Node.js alata.

3.1. Priprema lokalne okoline

Na samom je početku izrade odrađena instalacija alata *Microsoft Visual Studio Code*. Preuzimanje je moguće odraditi besplatno. Nakon instalacije alat je spreman, čije sučelje izgleda kao na slici 3.1. Kako bi se mogao pokrenuti kod pisan u JavaScript jeziku, potrebno je instalirati Node.js što je također besplatno i dostupno na web stranici *https://nodejs.org/en*. Nakon izvršene instalacije potrebno je kreirati React projekt nazvan ***leptira*** u kojem će se razvijati web aplikacija. Također je instaliran *bootstrap* za lakše i brže razvijanje prototipa web aplikacije.



Slika 3.1. *Sučelje alata VS Code*

3.2. Izrada stranica i routera

U mapi projekta automatski su izrađene datoteke i mape potrebne za razvijanje React projekata. U mapi *src* napravljena je mapa *pages* u koju su spremljene sve datoteke web aplikacije vezane za funkcionalnosti i izgled. Napravljene stranice su *Home*, *Login*, *MyBook*, *NewBook*, *Profile* i *Register*. U glavnoj datoteci projekta *App.js* sadržani su svi elementi potrebni za ispravan rad web aplikacije Leptira. Programski kod datoteke *App.js* prikazan je na slici 3.2. Prvi korišteni JSX (engl. *JavaScript Extensible Markup Language*) element [6] je <*UserProvider*> koji omogućuje omatanje cjelokupne aplikacije u takozvani kontekst (u ovom slučaju *UserContext*). Kontekst omogućuje pristup podacima svakoj komponenti i stranici koju obavija bez prosljeđivanja podataka putem svojstava (engl. *props*). U konkretnom slučaju, *UserContext* omogućuje pristup podacima prijavljenog korisnika na svim stranicama i komponentama što je detaljnije objašnjeno u potpoglavlju Konteksti. Ovdje se također nalazi i element <*ErrorProvider*> koji daje pristup ispisu grešaka preko *GlobalErrorContexta.* Sljedeći je element <*BrowserRouter*> [7] koji omogućuje brže usmjeravanje s jedne stranice na drugu na način da definira putanje za navigaciju kroz aplikaciju. React funkcionira na način da je cjelokupna aplikacija jedan dokument čiji se sadržaj mijenja ovisno o korisnikovim akcijama. Nakon omotavanja *BrowserRouter*, slijedi element <*Redirect*> u kojem su napisane sve rute. Unutar elementa <*BrowserRouter*> se također nalaze i pozivi elemenata *Navbar*, *GlobalMessage* i *GlobalError* potrebnih za prikaz na svakoj stranici (klasičan izgled navigacijske trake na vrhu stranice i obavijest o izvršenoj radnji ili grešci). U svakom trenutku se može dogoditi greška ili nešto što nije bilo korisnikova namjera. Za takve situacije je potrebno obavijestiti i samog korisnika aplikacije.

```
<StyleSheetManager shouldForwardProp={isPropValid}>
  <UserProvider>
    <ErrorProvider>
      <MessageProvider>
        <BrowserRouter>
          <GlobalStyle />
          <Navbar />
          <GlobalError />
          <GlobalMessage />
          <Redirect disabled>
            <Routes>
              <Route path="/" element={<Home />} />
              <Route path="/login" element={<Login />} />
              <Route path="/newbook" element={<NewBook />} />
              <Route path="/profile" element={<Profile />} />
              <Route path="/register" element={<Register />} />
              <Route path="/mybook/:bookId" element={<MyBook />} />
            </Routes>
          </Redirect>
        </BrowserRouter>
      </MessageProvider>
    </ErrorProvider>
  </UserProvider>
</StyleSheetManager>
```

Slika 3.2. *App.js*

3.3. Konteksti

3.3.1. GlobalErrorContext.js

Kako bi se osiguralo da se greška ispiše na zaslonu, napravljena je datoteka *GlobalError.js* koja ispisuje poruku greške. Kako bi se omogućio pristup poruci o grešci potreban je kontekst. U kontekstu se koristi stanje *globalError* koje je proslijeđeno svim komponentama potomcima koje taj kontekst obavija. Poruka o grešci će se pojaviti ispod navigacijske trake na 10 sekundi i nestati. Poruka koju ispisuje je upravo poruka greške same aplikacije (*error.message*). Programski kod konteksta prikazan na slici 3.3. prikazuje način kreiranja konteksta [9].

```
1  import React, { createContext, useState, useEffect } from "react";
2
3  export const GlobalErrorContext = createContext("");
4
5  const ErrorProvider = ({ children }) => {
6    const [globalError, setGlobalError] = useState("");
7
8    useEffect(() => {
9      if (!globalError) return;
10     setTimeout(() => {
11       setGlobalError("");
12     }, 10 * 1000);
13   }, [globalError]);
14
15   return (
16     <GlobalErrorContext.Provider value={{ globalError, setGlobalError }}>
17       {children}
18     </GlobalErrorContext.Provider>
19   );
20 };
21
22 export default ErrorProvider;
```

Slika 3.3. *Programski kod GlobalErrorContext.js*

3.3.2 GlobalMessageContext.js

Kao što *GlobalErrorContext* omogućava ispisivanje poruka o greškama, *GlobalMessageContext* ispisuje ostale poruke te je ona identična poruci o greškama. Jedina je razlika u imenima varijabli i elemenata programskog koda. Funkcija poruke jeste obavijestiti korisnika o ostalim radnjama u kojima može doći do greške. U aplikaciji Leptira se koristi kako bi se korisnik obavijestio da je e-mail za promjenu lozinke uspješno poslan.

3.3.3. UserContext.js

Najvažniji kontekst koji je potreban na svakoj stranici je upravo *UserContext* koji sadrži podatke o prijavljenom korisniku i omogućava svakoj stranici pristup korisničkim podacima. Ti su podaci potrebni na *Home* stranici, na *MyBook* stranici i slično. Preko ugrađenih funkcija koje nudi Firebase, vrlo je jednostavno postaviti te podatke, ali također i pristupiti im, čitati ih i uređivati. Kontekst uzima preko ugrađene funkcije *onAuthStateChanged* [10] trenutnog korisnika pomoću Furebase autentikacije i njega postavlja kao stanje *user*. Preko konteksta omogućen je pristup tom stanju i sa svake stranice je moguće pristupiti korisničkim podacima. Slika 3.4. prikazuje cjelokupan kod datoteke *UserContext.js*.

```
1  import { onAuthStateChanged } from "firebase/auth";
2  import React, { createContext, useState } from "react";
3  import { auth } from "../config";
4
5  export const UserContext = createContext(null);
6
7  const UserProvider = ({ children }) => {
8    onAuthStateChanged(auth, (user) => {
9      if (user) setUser(user);
10     else setUser(null);
11   });
12   const [user, setUser] = useState(null);
13   return <UserContext.Provider value={user}>{children}</UserContext.Provider>;
14 };
15
16 export default UserProvider;
17
```

Slika 3.4. *Programski kod UserContext.js*

3.4. Komponente

3.4.1. BookCard.js

Datoteka *BookCard.js* napravljena je za prikaz kartica na stranici *Home* i to u obliku kartice koja ukratko sadrži podatke o lektiri. Kartica se sastoji od naslova knjige, imena autora te u svom sadržaju prikazuje temu djela. Kartica također sadrži i dvije kontrole: *Pregledaj* i *Ukloni*. Kontrola *Pregledaj* otvara stranicu *MyBook* koja prikazuje unesene podatke o lektiri i prosljeđuje joj svojstva o lektiri koju predstavlja kartica, odnosno ID knjige u bazi podataka. Kontrola *Ukloni* briše zapis lektire iz baze podataka. Brisanje je složenija funkcija koja mora nakon pritiska na kontrolu Ukloni postaviti polje knjiga prikazanih na stranici Home pomoću filtera na način da nakon brisanja iz baze podataka mora ponovno filtrirati knjige korisnika i njih prikazati. Filtriranje se odvija na način da se postave sve knjige koje su bile u polju knjiga

na stranici *Home* osim one na čijoj je kartici pritisnuta kontrola Ukloni. Funkcija za brisanje lektire je prikazana na slici 3.5.

```
const handleClickDelete = async (e) => {
    e.preventDefault();
    try {
      const bookRef = doc(db, "book", props.id);
      await deleteDoc(bookRef);
      const updatedBooks = props.books.filter(
        (book) => book.id !== props.book.id
      );
      props.setBooks(updatedBooks);
    } catch (error) {
      setGlobalError(error.message);
    }
  };
```

Slika 3.5. *Otvaranje pregleda lektire*

3.4.2. GlobalError.js

Datoteka *GlobalError.js* obavlja prikaz greške na svakoj stranici. Element *<GlobalError>* ispisuje poruku *error.message* generiranu *try-catch* funkcijom [8]. Pristup elementu omogućuje *GlobalErrorContext*. Poruka je greške smještena u *bootstrap Alert* komponentu. Datoteka *GlobalError.js* prikazana je na slici 3.6.

```
const GlobalError = () => {
  const { globalError } = useContext(GlobalErrorContext);

  return globalError ? (
    <Container>
      <Alert variant="danger" className="mt-3">
        {globalError}
      </Alert>
    </Container>
  ) : null;
};
```

Slika 3.6. *GlobalError.js*

3.4.3. GlobalMessage.js

Datoteka *GlobalMessage.js* predstavlja poruku ostale vrste na svakoj stranici. Sadržaj ispisa određen je od strane programera koji tekst poruke postavlja u kodu aplikacije. Pristup elementu omogućuje *GlobalMessageContext*. Funkcionalnost datoteke *GlobalMessage.js* prikazana je na slici 3.7.

```
const { globalMessage } = useContext(GlobalMessageContext);

  return globalMessage ? (
    <Container>
      <Alert variant="success" className="mt-3">
        {globalMessage}
      </Alert>
    </Container>
  ) : null;
};
```

Slika 3.7. *GlobalMessage.js*

3.4.4. Navbar.js

Datoteka *Navbar.js* odrađuje funkcionalnosti koju svaka navigacijska traka ima, a to su navigacija između različitih stranica, ali i mogućnost odjave korisnika. Koristi *UserContext* te *GlobalErrorContext* kako bi se pritiskom na *Logout* poveznicu trenutni korisnik mogao odjaviti, ali i dobiti obavijest o mogućoj grešci. *Logout* je napisan u *try-catch* bloku i tako da ako ne uspije doći do valjane odjave poziva se funkcija *setGlobalError* iz *GlobalErrorContext* i ispisuje se poruka o grešci na ekranu što je vidljivo u kodu na slici 3.8.

```
const handleLogOut = async () => {
    try {
      await signOut(auth);
      navigate("/login");
    } catch (error) {
      setGlobalError(error.message);
    }
  };
```

Slika 3.8. *Funkcija za odjavu korisnika*

3.4.5. Redirect.js

Kako bi se omogućilo da, ukoliko korisnik nije prijavljen, svaki klik na poveznice na stranici odvode na stranicu *Login*, napravljena je datoteka *Redirect.js*. Ona koristi funkcije *useNavigate()*, *useLocation()*, *UserContext* kako bi bila dostupna informacija je li korisnik prijavljen ili ne i polje *allowedRoutes* kako se ne bi dogodilo prebacivanje ako je trenutna lokacija *Login* ili *Register* stranica.Koristi se stanje *firstLoad* koje gleda je li u pitanju prvo učitavanje stranice. Ako je *firstLoad* postavljen na *true*, *Redirect* je onemogućen i neće se

odvesti na stranicu *Login*. *Redirect* komponenta ne vraća nikakav HTML kod *(<>{children}</>)*. Programski je kod prikazan na slici 3.9.

```
useEffect(() => {
    if (disabled) return;

    if (firstLoad) {
      setFirstLoad(false);
      return;
    }
    if (!user && !allowedRoutes.includes(location.pathname)) navigate("/login");
    // eslint-disable-next-line
  }, [user, firstLoad, location]);
```

Slika 3.9. *Funkcija prebacivanja stranica na stranicu za prijavu*

3.5. Stranice

3.5.1. Home.js

Prva stranica aplikacije je *Home* stranica. Na njoj se, kao i na svakoj, nalazi navigacijska traka na vrhu stranice. Kontrola *Ukloni* koja se nalazi na kartici unesene knjige briše zapis o pročitanoj lektiri iz baze podataka. Preko gotovih funkcija *Firebasea* dolazi se do polja dokumenata kojima je naveden uvjet: *user* == *user.id*, što znači da uzima samo one dokumente koje je spremio trenutno prijavljeni korisnik. U *returnu* stranice se za svaki dokument polja *rendera* nova kartica s pripadajućim podacima. Funkcija kojom se uzima filtrirani popis korisnikovih knjiga i postavlja ih kao stanje *books* prikazana je na slici 3.10.

```
useEffect(() => {
    (async () => {
      if (!user) {
        navigate("/login");
      }
      try {
        const booksRef = collection(db, "book");
        const q = query(booksRef, where("user", "==", user.uid));
        const booksSnap = await getDocs(q);

        let tempBooks = [];
        booksSnap.forEach((doc) =>
          tempBooks.push({ id: doc.id, ...doc.data() })
        );
        setBooks(tempBooks);
      } catch (error) {
        console.log(error.message);
      }
    })();
  }, [user, navigate, setGlobalError]);
```

Slika 3.10. *Postavljanje polja knjiga korisnika*

3.5.2. Login.js

Kako bi se omogućilo korisniku prijavljivanje pri otvaranju aplikacije, mora postojati stranica za prijavu. Datoteka *Login.js* uzima vrijednosti iz dva tekstna okvira. Nakon unosa valjanih podataka i na pritisak kontrole odrađuje se funkcija prikazana na slici 3.11. i korisnik se prijavljuje u aplikaciju. Također se nalaze dvije poveznice: *REGISTRIRAJ SE!* i *Zaboravili ste lozinku?*. Pritiskom na poveznicu *REGISTRIRAJ SE!* otvara se stranica za registraciju *Register*, a pritiskom na *Zaboravili ste lozinku?* otvara se modal koji daje mogućnost unosa nove lozinke korisniku. Kako bi se omogućilo otvaranje modala, koristi se stanje *modalOpen* koji je postavljen kao *false*, a pritiskom na poveznicu se ono postavlja na *true* i prikaže se modal. Koriste se stanja *email* i *password* koji su postavljeni na vrijednosti unesena u polja. Pozivom funkcije *signInWithEmailAndPassword(auth, email, password)* [11] pritiskom na kontrolu *Prijavi se* odrađuje se prijava korisnika.

```
1  const handleSubmit = async (e) => {
2      e.preventDefault();
3
4      try {
5        await signInWithEmailAndPassword(auth, email, password);
6        navigate("/");
7      } catch (error) {
8        setError(error.message);
9      }
10   };
11
```

Slika 3.11. *Programski kod prijave korisnika*

3.5.3. Register.js

Stranica Register radi na sličan princip kao i stranica Login. Uzimaju se stanja email i password i poziva se funkcija createUserWithEmailAndPassword [12] koja kreira novog korisnika. Registracija se odrađuje tek nakon što je provjereno jesu li obje lozinke identične jer je pri registraciji potrebno unijeti lozinku, ali i potvrdu lozinke. Na slici 3.12. prikazana je funkcija registracije. Nakon što je korisnik registriran, u Firebase se unosi novi korisnik u kolekciju user s podacima name, surname i grade. Na stranici se također nalazi poveznica PRIJAVI SE! koja vraća na stranicu Login.js.

```
try {
      const result = await createUserWithEmailAndPassword(
        auth,
        email,
        password1
      );
      const userRef = doc(db, "user", result.user.uid);
      const data = {
        name,
        surname,
        grade,
      };
      await setDoc(userRef, data);

      navigate("/");
    } catch (error) {
      setGlobalError(error.message);
    }
```

Slika 3.12. *Programski kod registracije korisnika*

3.5.4. Profile.js

Potrebno je omogućiti izmjenu korisničkih podataka ako dođe do promjene razreda. Na stranici *Profile* to je i omogućeno. Na jednoj kartici su napisani podaci korisnika: ime, prezime razred i adresa elektroničke pošte. Na slici 3.13. prikazano je iščitavanje podataka iz baze podataka i njihovo spremanje u stanja čije su vrijednosti upravo vrijednosti tekstnih okvira. Sve je te podatke moguće izmijeniti. Postoji kontrola *Izmijeni* koji stavlja stanje *editable* na *true*. Dok je *editable* postavljen na *false*, nijedno polje za unos teksta nije omogućeno za izmjenu, ali promjenom na *true* sve se može izmijeniti. Pritiskom na *Izmijeni*, kontrola postaje kontrola *Spremi*. Nakon izmjene, na pritisak *Spremi*, dolazi do promjene korisničkih podataka u kolekciji *user*, te se ponovno podaci iščitaju iz baze podataka.

```
useEffect(() => {
    (async () => {
      if (!user) return;
      const userRef = doc(db, "user", user.uid);
      const userSnap = await getDoc(userRef);
      const userData = userSnap.data();
      console.log(userData);
      setName(userData.name);
      setSurname(userData.surname);
      setGrade(userData.grade);
      setEmail(auth.currentUser.email);
    })();
  }, [user]);
```

Slika 3.13. *Programski kod čitanja podataka*

3.5.5. MyBook.js

Pritiskom na karticu na stranici *Home* otvara se stranica *MyBook*. Na stranici *MyBook* nalaze se svi uneseni podaci za pročitanu lektiru. Pri otvaranju stranice svako polje za unos teksta je omogućeno za upisivanje, a sve unesene promjene se spremaju na pritisak kontrole *Spremi izmjene*. Podaci koji su ispisani na stranici se nakon pritiska spremaju i izmjenjuju na istom dokumentu, odnosno na istoj lektiri koja je otvorena na *MyBook*. Na slici 3.14. je prikazana funkcija spremanja novih uređenih podataka u isti dokument.

```
const handleSave = async (e) => {
    e.preventDefault();
    try {
      const bookFinal = {
        bookName,
        author,
        authorNote,
        year,
        period,
        language,
        mainCharacter,
        characters,
        place,
        time,
        theme,
        idea,
        summary,
        quotes,
        conclusion,
        user: user.uid,
      };
      const bookRef = doc(db, "book", bookId);
      await updateDoc(bookRef, bookFinal);
    } catch (error) {
      setGlobalError(error.message);
    }
```

Slika 3.14. *Programski kod izmjene podataka*

3.5.6. NewBook.js

Stranica *NewBook* malo je kompliciranija verzija stranice *MyBook*. Pri otvaranju stranice onemogućena su polja za unos naslova djela i autora. Prvo se treba izabrati jedan od 12 razreda (8 osnovne i 4 srednje škole). Nakon odabira razreda odabire se jedno djelo ponuđeno sa službenog popisa lektira za aktualnu školsku godinu. Kada se odabere jedna lektira naslov djela i autor se automatski popunjavaju kako je prikazano na slici 3.15. Postoji i opcija *Ostalo* gdje se može unijeti neka vlastita knjiga koja se ne nalazi na popisu i u tome trenutku polja za upis naslova knjige i imena autora butu omogućena. Na dnu stranice nalazi se kontrola *Pročitaj leptiru* koji stvara novi dokument u kolekciji *book* s novim ID-em. U novom dokumentu se također sprema i ID korisnika koji je unio lektiru kako bi se one mogle prikazati na stranici *Home*.

```
useEffect(() => {
    if (book === "ostalo") {
      setDisableFields(false);
      setBookName("");
      setAuthor("");
    } else if (book !== "") {
      setDisableFields(true);
      const filteredData =
        data &&
        data[grade] &&
        data[grade].filter((item) => item.name === book)[0];
      setBookName(filteredData?.name || "");
      setAuthor(filteredData?.author || "");
    }
  }, [book]);
```

Slika 3.15. *Postavljanje naslova i autora*

3.6 Server

Kako bi aplikacija imala mogućnost aktivne aktualizacije službenog popisa lektira, ona dohvaća podatke zahtjevom na serveru. Server u sebi ima *books.js* i *periods.js* koji u sebi sadrže sve knjige sa službenog popisa i sva književno-povijesna razdoblja. Pisano je Expressom koji

```
app.get("/knjige-za-razred", (req, res) => {
  res.json(books[req.query.razred]);
});
```

Slika 3.16. *get funkcija servera*

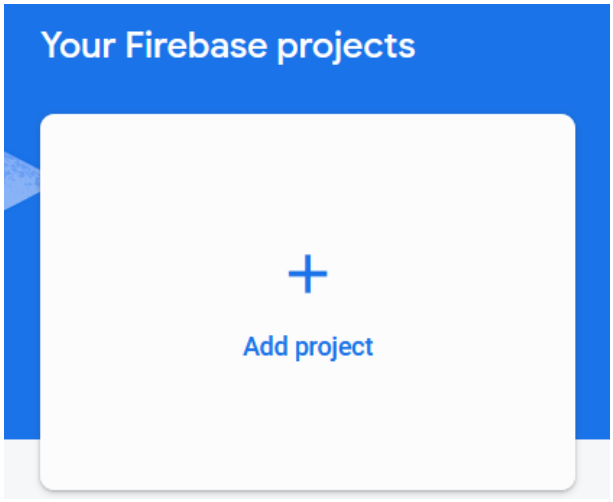
koristi ključne riječi req (request) i res (response) [13] kako je i prikazano na slici 3.16. Kako bi pristup, odnosno slanje zahtjeva serveru bilo brže i jednostavnije, upotrijebljene su unaprijed napisane funkcije u mapi *leptira* getAllBooks, getGradeBooks i getAllPeriods koje koriste Axios za komunikaciju sa serverom što je prikazano na slici 3.17.

```
export const getGradeBooks = async (grade) => {
  try {
    const endpoint = "http://localhost:8000/knjige-za-razred";

    const { data } = await axios.get(endpoint, {
      params: {
        razred: grade,
      },
    });

    return data;
  } catch (error) {
    console.log(error);
  }
};

```

Slika 3.17. *funkcija getGradeBooks*

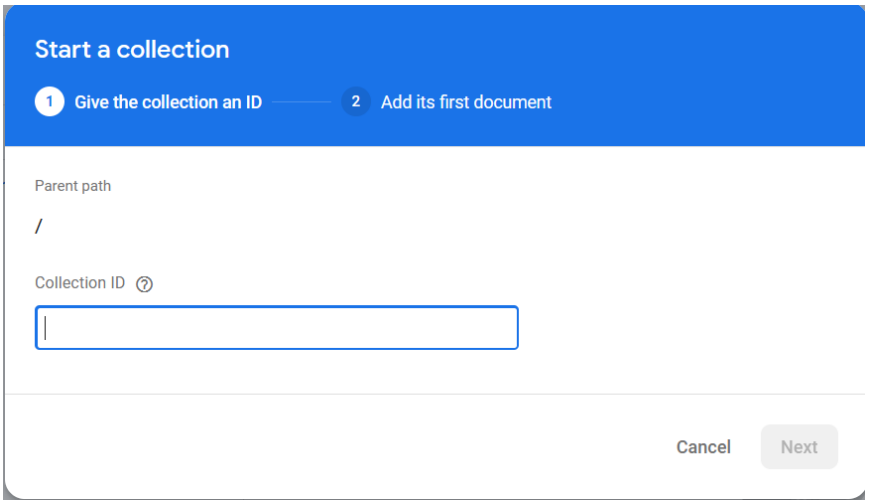
3.7. Firebase

Firebase je razvojni alat koji pruža obilje usluga i alata za razvoj mobilnih i web aplikacija. Ponuđene su razne ključne značajke kao što su baza podataka u stvarnom vremenu, autentifikacija korisnika, hosting, pohrana datoteka, analitika, upravljanje korisnicima i još mnogo toga. Kako bi se napravila baza na Firebase alatu, potrebno je otići na stranicu *https://firebase.google.com/* i prijaviti se. Nakon toga potrebno je izraditi novi projekt pritiskom na prozor prikazanom na slici 3.18.



Slika 3.18. *Izgled prozora za izradu projekta*

Nakon kreiranja projekta, izrađena je autentikacija na kojoj je odabrana usluga prijave preko e-mail adrese i lozinke. Također je izrađena Firestore database kako je prikazano na slici 3.19. Nakon toga potrebno je izraditi dvije kolekcije: book i user.

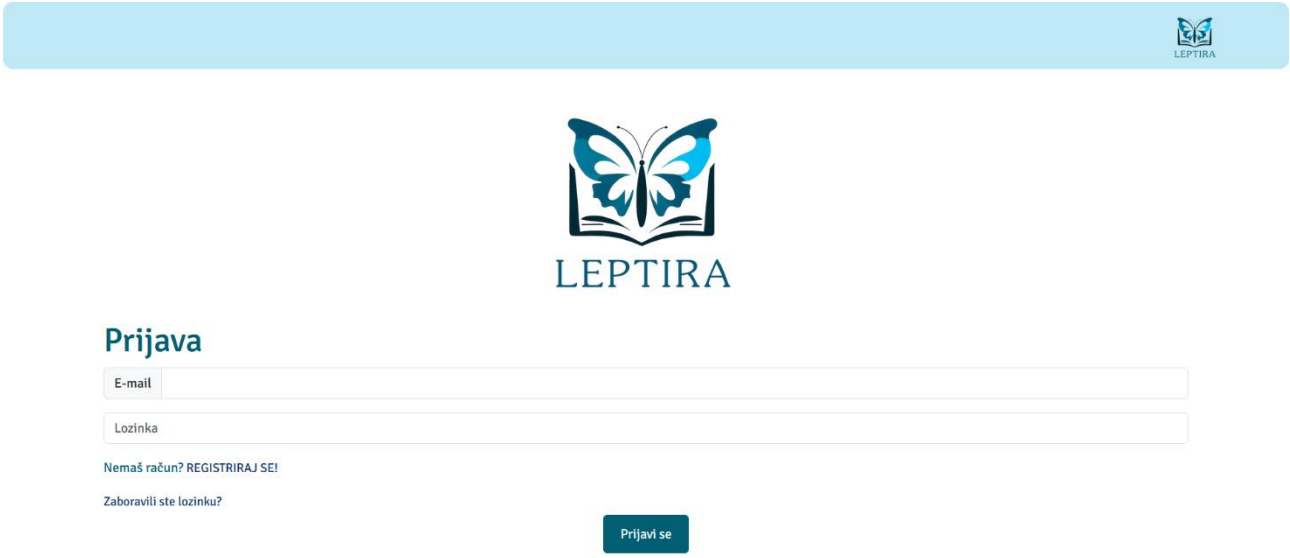


Slika 3.19. *Izrada kolekcije*

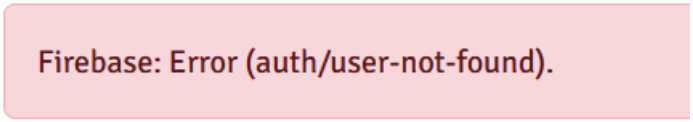
Nakon izrade kolekcija sve je spremno za registraciju korisnika i za spremanje novih pročitanih knjiga.

4. PRIKAZ RADA APLIKACIJE

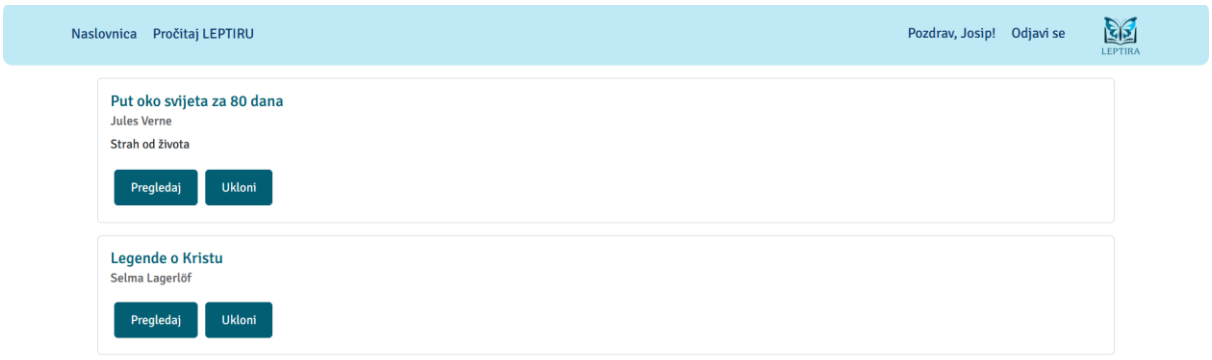
Pri učitavanju stranice otvara se *Login* stranica prikazana slikom 4.1. Stranica sadrži navigacijsku traku, logo aplikacije i te obrazac pod nazivom *Prijava*. Obrazac se sastoji od dva tekstna okvira, dvije poveznice i kontrolom za prijavu. Kako je vrijednost atributa tekstnog okvira za unos adrese elektroničke pošte postavljen na email, uneseni tekst mora bit u valjanom formatu adrese elektroničke pošte. Vrijednost atributa tekstnog okvira za unos lozinke je postavljena na password tako da je svaki uneseni znak prikazan kao točkica radi sigurnosti povjerljivih podataka. Unosom pogrešnih podataka prikazuje se poruka o grešci prikazana na slici 4.2. koje nestane nakon 10 sekundi. Unosom ispravnih podataka otvara se stranica čije je sučelje prikazano na slici 4.3.



Slika 4.1. *Sučelje stranice Login*

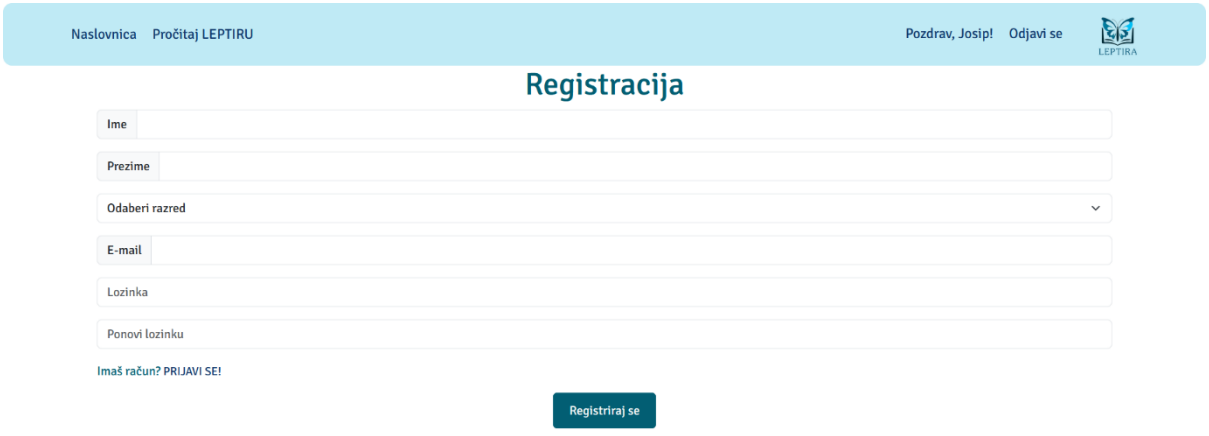


Slika 4.2. *Izgled poruke o grešci*



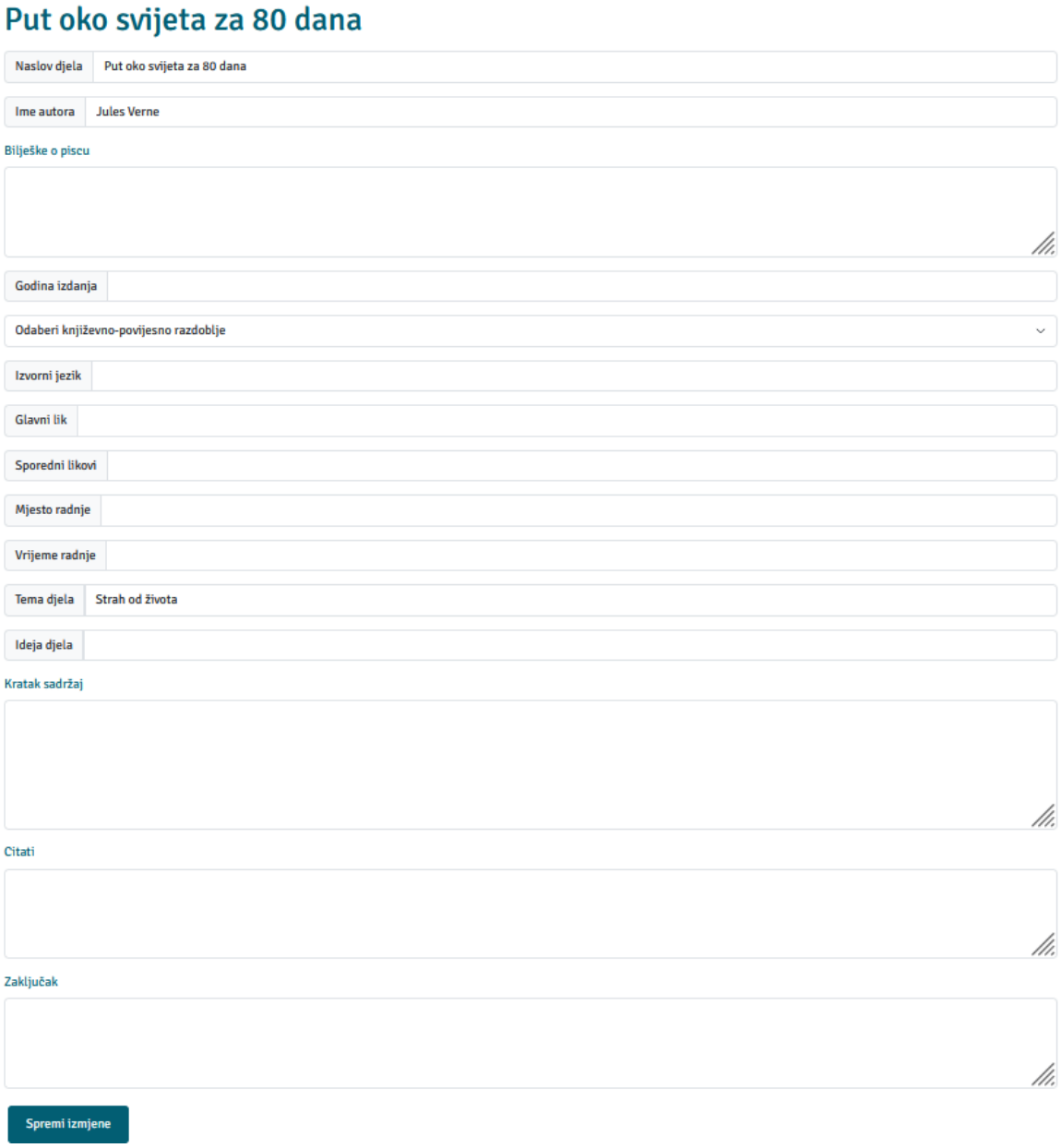
Slika 4.3. *Sučelje stranice Home*

Pritiskom na poveznicu *Registriraj se!* otvara se stranica *Register* koja prikazuje obrazac za registraciju korisnika. Obrazac prikazan na slici 4.4. sadrži polja za unos imena, prezimena, razreda, adrese elektroničke pošte, lozinke te ponovljene lozinke. Kao što *Login* ima poveznicu na stranicu za registraciju, tako i stranica za registraciju sadrži poveznicu na stranicu za prijavu. Pritiskom na kontrolu *Registriraj se* registrira se korisnik s podacima unesenim u obrazac.



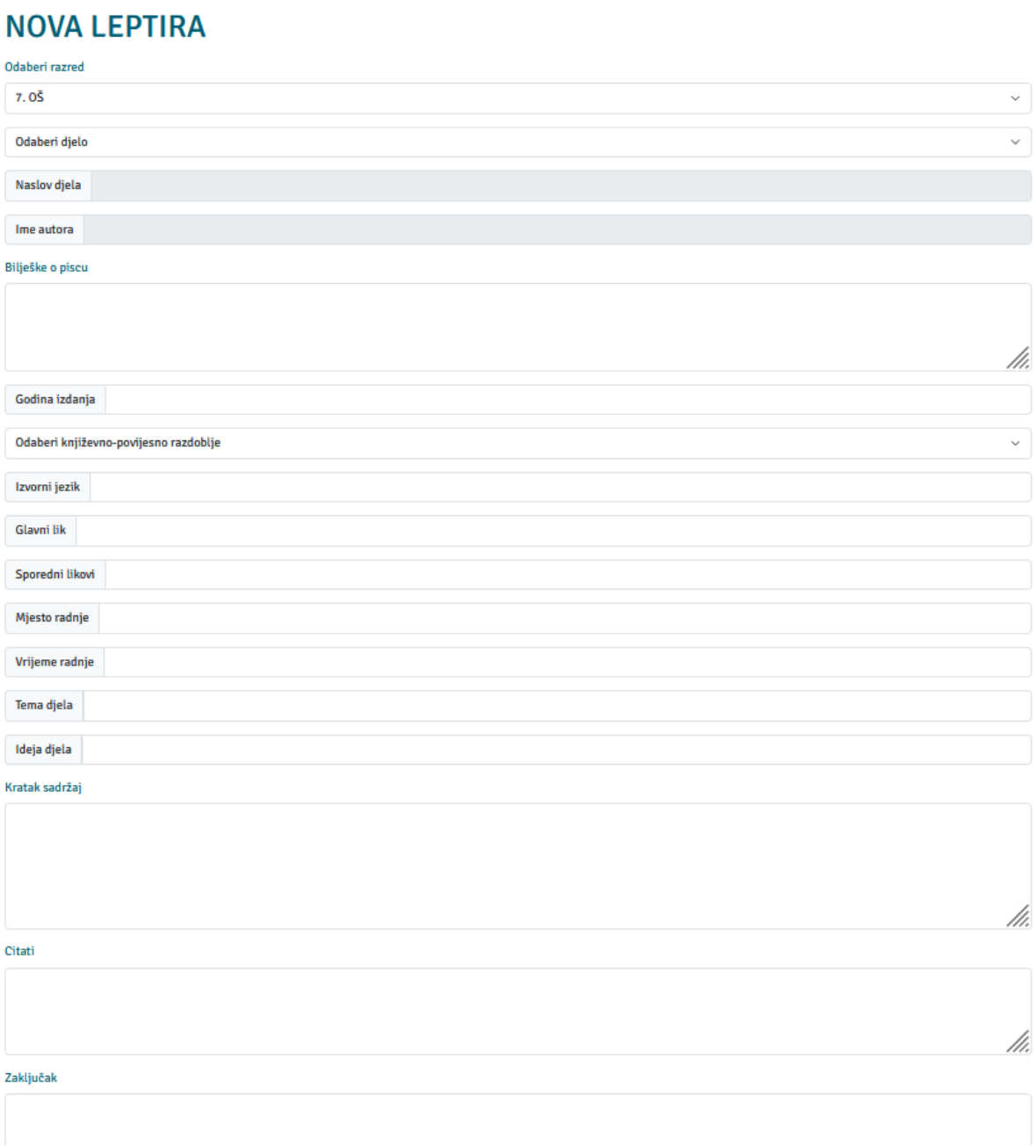
Slika 4.4. *Sučelje stranice Register*

Home stranica prikazuje navigacijsku traku s poveznicama na stranice aplikacije. Također se nalaze kartice za svaku unesenu knjigu prijavljenog korisnika. Pritiskom na kontrolu *Pregledaj* otvara se stranica *MyBook* koja ispisuje sve unesene podatke o odabranoj lektiri, a pritiskom na kontrolu *Ukloni* unesena knjiga se briše iz baze podataka. Sučelje stranice MyBook prikazano je na slici 4.5.



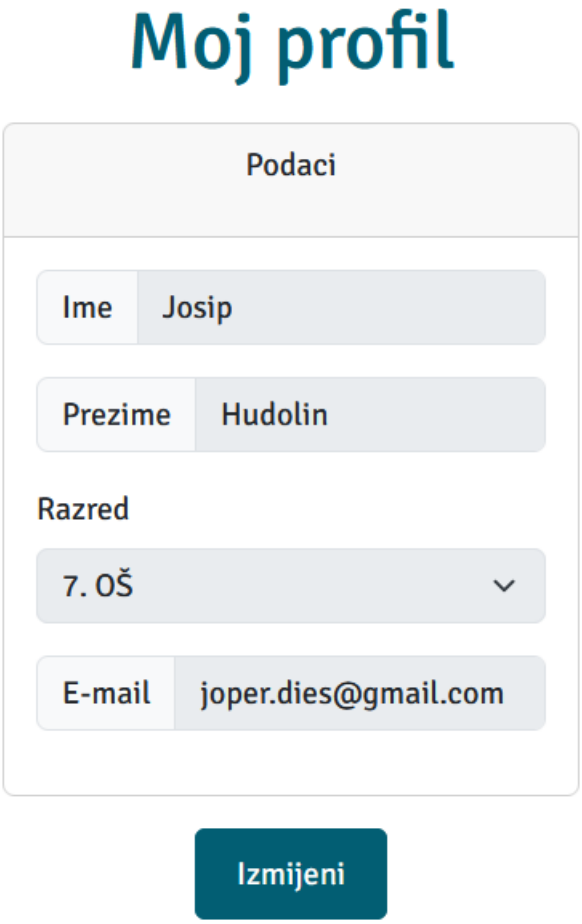
Slika 4.5. *Sučelje stranice MyBook*

Na pritisak kontrole *Spremi izmjene* sve promjene u podacima o lektiri spremaju se u isti dokument. Otvaranjem stranice *NewBook*, odnosno pritiskom na navigacijskoj traci na *Pročitaj LEPTIRU* otvara se obrazac prikazan na slici 4.6. koji je potrebno ispuniti. Ako je odabrano djelo s popisa lektire, polja za unos naslova djela i imena autora su postavljena na odabrano i onemogućena za uređivanje. Nakon unosa svih potrebnih podatak pritisne se kontrola *Pročitaj leptiru* koja sprema novi dokument u bazu podataka.



Slika 4.6. *Izgled stranice NewBook.js*

Odlaskom na stranicu *Moj profil* prikaže se kartica s podacima prijavljenog korisnika kao na slici 4.7. koje je moguće izmijeniti pritiskom na kontrolu *Izmijeni*. Do tada su sva polja onemogućena za uređivanje.



Slika 4.7. *Izgled stranice Profile.js*

Pritiskom na poveznicu *Logout* na navigacijskoj traci korisnik se odjavi i prebacuje se na *Login* stranicu.

4.1. Usporedba izrađene aplikacije sa sličnim rješenjima

Nakon što je web aplikacija Leptira izrađena, moguća je usporedba s aplikacijama opisanim u poglavlju Pregled sličnih područja.

Aplikacija Book Reports sadrži analize raznih djela koje su na popisima lektira diljem svijeta. Ne postoji nikakva mogućnost registracije i prijave korisnika dok je u aplikaciji Leptira ta mogućnost implementirana. Također, u aplikaciji Book Reports ne postoji mogućnost pohrane vlastitih analiza. Osnovna je funkcija aplikacije Leptira upravo spremanje vlastitih analiza kako bi se moglo pripremiti za ispitivanja lektira, ali i pripremu za esej državne mature.

Lektira.hr sadrži iste funkcionalnosti kao i Book Reports, no razlika je što se u analizama ipak nalaze i djela hrvatske književnosti koja su obvezna za osnovne i srednje škole u Republici Hrvatskoj. Navedena aplikacija u svojoj bazi podataka ipak nema sva djela koja se čitaju u hrvatskim školama, dok Leptira omogućava unos svih djela sa službenog popisa, ali i bilo kojeg drugog djela.

Web aplikacija Goodreads, za razliku od prve dvije navedene aplikacije, ima mogućnost registracije korisnika, ali njezina funkcija nije zapis bilješki o pročitanim djelima. Glavna je svrha aplikacije evidencija pročitanih djela, ali i planiranja čitanja djela sa liste želja. Moguće je ostaviti recenzije o pročitanim djelima i postavljati statuse o knjigama, npr. „želim pročitati“, „čitam“ ili „pročitano“, ali i kupiti knjigu putem Amazona. Leptira nema mogućnost ostavljanja ocjene što bi bila poželjna funkcionalnost kao i mogućnost kupovine knjiga ili reklamiranja knjiga hrvatskih knjižara.

Bookreporter ima mogućnost registracije, no aplikacija je napravljena za praćenje zbivanja na književnoj sceni. Izlaze novosti o novoizdanim knjigama i recenzijama poznatih stručnjaka. Postoji i mogućnost dobivanja obavijesti putem e-maila. Leptira nije aplikacija za praćenje novosti već za spremanje vlastitih bilješki o pročitanim djelima.

Sparknotes je web aplikacija najsličnija aplikaciji Leptira. Specijalizirana je za spremanje bilješki ne samo iz književnosti, već i svih drugih školskih predmeta. Leptira također sadrži funkcionalnost spremanja, ali samo književnih djela. Sparkontes nudi i opciju čitanja književnih djela i raznih kvizova za vježbu i pripremu. Naravno, djela u bazi podataka su za učenike Sjedinjenih Američkih Država i Ujedinjenog Kraljevstva, dok je Leptira namijenjena hrvatskim učenicima.

6. ZAKLJUČAK

Proces razvoja i implementacije web aplikacije je vrlo složen i zahtjeva integraciju različitih tehnoloških komponenti. U ovome je radu opisan postupak izrade aplikacije koji započinje samom motivacijom za izradu aplikacije, nakon čega slijedi pregled područja ideje, odnosno analiza već postojećih bliskih rješenja na odabranu problematiku. Idući je korak odabir najboljih alata za razvoj aplikacija, ali i programskih jezika koji mogu pružiti mogućnosti ciljane aplikacije. Idući je korak pisanje samog prototipa, izrada funkcionalne aplikacije s najjednostavnijim rješenjima za same funkcije aplikacije. Kada je prototip završen, slijedi optimizacija aplikacije, tj. pronalazak efikasnijih rješenja za funkcionalnosti aplikacije. Nakon same optimizacije ostaje aplikacija grdog i jednostavnog izgleda. Potrebno je osmisliti dizajn i implementirati kako bi aplikacija bila zanimljiva korisnicima i što lakša za korištenje. Nakon izrade dizajna ostaje testiranje i otklanjanje mogućih grešaka koda aplikacije. Kao što je vidljivo, izrada aplikacije bilo koje vrste je složen posao. U poslovnom svijetu, projekti zahtijevaju timove ljudi od kojih svako vrši svoju ulogu i proces izrade bude puno brži, ali i efikasniji jer je svaki član specijaliziran za rad u svojem području.

Leptira će pronaći svoje korisnike u učenicima osnovnih, ali i srednjih škola. Kako bi se mogao pratiti kontinuirani rad, u aplikaciju se unose informacije o pročitanim djelima i njima se može pristupiti u bilo kojem trenutku. Aplikacija će posebno pomoći srednjoškolcima koji prijavljuju ispite državne mature. Ona će im olakšati pripremu za esej iz hrvatskog jezika jer je tema eseja upravo jedno od pročitanih djela tokom srednjoškolskog obrazovanja.

Nakon postavljanja aplikacije na tržište, potrebno ju je i održavati. Uz brz tehnološki napredak, važno je i kontinuirano pratiti nove trendove i tehnologije kako bi se aplikacija održavala relevantnom i konkurentnom na tržištu. Kroz ovaj rad, stečeno je dublje razumijevanje procesa izrade web aplikacija. Daljnja istraživanja mogu se fokusirati na detaljniju analizu specifičnih tehnologija i pristupa, kao i na kontinuirano unapređenje korisničkog iskustva kako bi se ostvarile visoke razine zadovoljstva korisnika. Postoji još mnogo različitih funkcionalnosti koje se mogu dodati u aplikaciju kao što su mogućnost objavljivanja vlastitih zapisa drugim korisnicima, kao i mogućnost čavrljanja s drugim korisnicima kako bi se moglo upitati autore izvještaja o pročitanoj lektiri za moguće nejasnoće, ali i neslaganja.

LITERATURA

[1] Book Reports, https://www.bookreports.info/ [Pristupljeno: 20.6.2023.]

[2] Lektire.hr, https://www.lektire.hr/ [Pristupljeno: 20.6.2023.]

[3] Goodreads, https://www.goodreads.com/ [Pristupljeno: 20.6.2023.]

[4] Bookreporter, https://www.bookreporter.com/ [Pristupljeno: 20.6.2023.]

[5] Sparknotes, https://www.sparknotes.com/ [Pristupljeno: 20.6.2023.]

[6] D. Flanagan, JavaScript: The Definitive Guide, Sixth Edition, O'Reilly Media, Inc., Sebastopol, 2011.

[7] React Router, https://reactrouter.com/en/main/router-components/browser-router [Pristupljeno: 23.6.2023.]

[8] FreeCodeCamp, https://www.freecodecamp.org/news/try-catch-in-javascript/ [Pristupljeno: 24.6.2023.]

[9] Medium, https://medium.com/swlh/understanding-context-in-js-eceb5ef1fa75 [Pristupljeno: 24.6.2023.]

[10] Firebase, https://firebase.google.com/docs/auth/web/manage-users#get_the_currently_signed-in_user [Pristupljeno: 25.6.2023.]

[11] Firebase, https://firebase.google.com/docs/auth/web/password-auth#sign_in_a_user_with_an_email_address_and_password [Pristupljeno: 25.6.2023.]

[12] Firebase, https://firebase.google.com/docs/auth/web/password-auth#create_a_password-based_account [Pristupljeno: 25.6.2023.]

[13] E. Brown, Web Development with Node and Express, O'Reilly Media, Inc., Sebastopol, 2014.

SAŽETAK

Zbog prevelike količine informacija, učenicima je potreban alat koji im olakšava pohranu tih istih informacija. Kao rješenje tom problemu, izrađena je aplikacija po imenu Leptira koja omogućuje spremanje i čitanje izvještaja o pročitanoj lektiri, ali i o bilo kojem drugom književnom djelu. Aplikacija je pisana u Reactu koji je danas najkorišteniji za izradu web aplikacija. Kako bi svaki korisnik mogao unositi vlastite analize, aplikacija ima mogućnost registracije i prijave korisnika. Aplikacija ima sve funkcionalnosti kako bi korisnicima pružila iskustvo jednostavne, ali vrlo korisne web aplikacije koja im olakšava vođenje bilješki o pročitanom djelu, ali i srednjoškolcima omogućava pripremu za esej državne mature iz hrvatskog jezika.

Ključne riječi: analiza, bilješke, čitanje, Firebase, knjiga, React, server

ABSTRACT

Web Application for Managing Mandatory Books

Due to the excessive amounts of information, students need a tool that facilitates the storage of these same pieces of information. To address this issue, an application named Leptira has been developed, enabling the storaging and reading of book reports, as well as any other literary work. The application is built using React, which is currently the most widely used framework for web application development. To allow each user to store their own analyses, the application features user registration and login capabilities. The application has all functionalities to provide users with a simple yet highly useful web application that eases the process of keeping notes on the read works. Moreover, it offers high school students the opportunity to prepare for their final exams.

Key words: analysis, book, Firebase, notes, React, reading, server

ŽIVOTOPIS

Josip Hudolin je rođen u Bošnjacima, selu pokraj Županje, 31. svibnja 2001. godine. Osnovnu školu je završio u rodnome selu, a prirodoslovno-matematičku gimnaziju u Županji. Nakon gimnazije odlazi u Osijek na Fakultet elektrotehnike, računarstva i informacijskih tehnologija. Kao student postaje stipendist uspješne hrvatske tvrtke „Comping d.o.o.“.