Web aplikacija za dokumentiranje i praćenje profesionalnog napredovanja nastavnika

Slijepčević, Borna

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

WEB APLIKACIJA ZA DOKUMENTIRANJE I PRAĆENJE PROFESIONALNOG NAPREDOVANJA NASTAVNIKA

Završni rad br. 16/RAČ/2023

Borna Slijepčević

Bjelovar, listopad 2023.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Borna Slijepčević** JMBAG: **0314023040**

Naslov rada (tema): **Web aplikacija za dokumentiranje i praćenje profesionalnog napredovanja nastavnika**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Dario Vidić, mag. ing. el. techn. inf.** zvanje: **viši predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Tomislav Adamović, mag. ing. el., predsjednik**
2. **Dario Vidić, mag. ing. el. techn. inf., mentor**
3. **Ivan Sekovanić, mag. ing. inf. et comm. techn., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 16/RAČ/2023

U sklopu završnog rada potrebno je:
1. razviti web aplikaciju koristeći react i TypeScript tehnologije na modularan i konzistentan način s react tehnikama kontekst i prilagođene kuke za upravljanje stanjem autentikacije i autorizacije namijenjeno korisnicima različitih razina ovlasti i funkcija
2. izraditi responzivno sučelje s konzistentnim i minimalističkim dizajnom i modalima za fokus na dodatne informacije i interaktivne elemente koristeći Ant Design komponente i Font Awesome ikone
3. omogućiti povezivanje aplikacije na Azure podatkovne servise sa osiguravajućim elementom JSON Web Token
4. implementirati sigurnosne protokole i komunikaciju s MySQL bazom podataka preko običnih i parametriziranih axios api poziva i pregleda isteka sesije preko JSON Web Tokena te dodatnih kontrola unosa od strane korisnika preko provjera unosa i povratnih informacija korisniku kroz Ant Design alert prozora

Datum: 29.09.2023. godine

Mentor: **Dario Vidić, mag. ing. el. techn. inf.**

Zahvala

Prvenstveno želio bi se zahvaliti mentoru Dariu Vidiću i svim profesorima na dragocjenim savjetima i stručnom usmjeravanju. Također, želim zahvaliti kolegama i prijateljima koji su pružili podršku tijekom tjekom dugih radnih sati i stresnih trenutaka. Hvala i mojoj obitelji na kontinuiranoj podršci, ljubavi i vjerovanju u mene.

Sadržaj

1. UVOD

Svrha ovog rada je opisati proces izrade web aplikacije koje omogućuje nastavnicima dokumentiranje i praćenje svog profesionalnog razvoja. Aplikacija omogućuje praćenje napretka putem prilaganja potrebnih dokumenata definiranih u pravilniku o napredovanju učitelja, nastavnika, stručnih suradnika i ravnatelja u osnovnim i srednjim školama i učeničkim domovima. Tijekom prilaganja datoteka dodjeljuju se odgovarajući bodovi i zahtjeva se odobrenje ovlaštene osobe ustanove kojoj nastavnik pripada kako bi dokument bio valjan za napredak.

U drugom poglavlju spominje se alat Visual Studio Code, još poznat kao VS Code u kojem se vodi cjelokupan razvoj aplikacije. Uz njega dolaze vanjski alati za dizajn, od kojih je potrebno instalirati neke, a neke samo implementirati. Osim alata navode se korištene tehnologije, uključujući programske jezike i biblioteke. Aplikacija koristi Azure kontejner s Azure Blob Storage za pohranu korisničkih datoteka koje se pristupa putem API poziva.

U trećem poglavlju osvrt je na opisivanje funkcionalnosti aplikacije za sve razine korisnika. Zatim opisivanje uspostavljanja radnog okruženja i arhitektura mapa s datotekama nakon čega prolazi se kroz postavljanje rasporeda na sučelju kroz CSS i navode se načini upravljanja tokom informacija između komponenti i prelazak iz komponente u komponentu na sučelju. Naposlijetku pokazuju se primjeri komponenti, njihova struktura, način izrade i na koji način se mogu prenijeti u druge komponente. Na kraju rada navodi se zaključak u kojem je ukratko rezimiran cijeli rad.

2. KORIŠTENI ALATI I TEHNOLOGIJE

Prilikom izrade web aplikacije koriste se brojne tehnologije i alati. U ovom poglavlju se navode i pojašnjavaju pojedinačno. Tehnologije i alati koji su korišteni tijekom izrade ovog rada su:

- Visual Studio Code
- React
- Typescript
- Azure servisi – Azure kontejner i Azure Blob Storage
- HTML
- CSS
- Dizajn alati – Ant Design, UIVerse, Font Awesome

2.1 Visual Studio Code

Visual Studio Code je besplatan i uređivač otvorenog koda (engl. *open-source)* razvijen od strane Microsoft-a [1].

Podržava mnoge programske neki od kojih su: JavaScript, TypeScript, Java, Python, C++, C#, Go, Rust, Ruby, PHP, Kotlin, Swift i mnogi drugi. Također podržava sustave za verzioniranje kao što su Git i Subversion te debugiranje. Dolazi uz pametno nadopunjavanje koda, korisnik također može mijenjati temu, prečace i brojne postavke i instalirati mnoštvo ekstenzija što od strane Microsofta a što od strane zajednice. Uz sve funkcionalnosti, važna odlika je što je dostupan u različitim operativnim sustavima Windows, macOS i Linux.

2.2 React

React je popularna JavaScript biblioteka otvorenog koda za izgradnju korisničkog sučelja razvijena od strane Facebook-a, danas poznat kao Meta [2].

Ono što React ističe od drugih tehnologija za razvoj korisničkih sučelja jest da može biti korišten za izradu jednostraničnih (engl. *single-page)* web aplikacija koristeći ponovno upotrebljive UI komponente koje se mogu kombinirati.

Koristi JSX sintaksu što je spajanje JavaScript i XML tehnologije što omogućuje pisanje naizgled sličnog HTML koda u JavaScript-u. U programskom kodu 2.1. vidljivo je korištenje

XML elementa <button> i JavaScript atributa className i type te običnog JavaScript teksta Prijavi se.

Programski kod 2.1: JSX sintaksa

```
<button className="button" type="submit">
     Prijavi se
</button>
```

Podržava i virtualni DOM koji čini renderiranje komponenti i sadržaja brzo i učinkovito izradom i pohranom pravog DOM-a, usporedi ga nakon promjena (npr. korisničke interakcije) s virtualnim DOM-om i ažurira pravi DOM samo gdje pronađe razlike.

React s inačicom 16.8 unosi velike promjene poput stanja i Hooks API. Stanja omogućuju dijeljenje podataka kroz stablo komponenata kako bi se izbjeglo ručno prosljeđivanje podataka (engl. *prop-drilling*) zbog potencijalnog dupliciranja koda. Dok Hooks API omogućuje praćenje stanja, efekata i drugih React koncepata unutar funkcija te razdvajanje, prenošenje i ponovno korištenje dijelova logike.

React rijetko za optimalan rad treba dodatne biblioteke poput React Router koja omogućuje navigaciju između stranica i Axios koja pruža jednostavnu sintaksu za izvođenje HTTP zahtjeva poput GET, POST, PUT i DELETE prema poslužitelju.

2.3 TypeScript

TypeScript je visoki programski jezik besplatnog i otvorenog koda razvijen od strane Microsoft-a koji se izvršava u pretraživaču (engl. *browser*). Proširuje JavaScript tako što dodaje tipove u jezik omogućujući otkrivanje grešaka prije pokretanja koda (engl. *runtime*) i na taj način ubrzava razvojno iskustvo [3].

Upotrebljiv je u bilo kojem operativnom sustavu gdje se izvršava JavaScript i dostupan je na GitHub-u. TypeScript podržava sučelja (engl. *interface*) prikazanog u programskom kodu 2.2. koja omogućavaju definiranje strukture pojedinih objekata koju je moguće prenijeti u druge komponente koristeći ključne riječi *export* za izvoz te *import* za uvoz. Zatim klasa koje su zaslužne za bolju organizaciju koda iako korištenje klasa u React-u je izbačeno uvođenjem stanja i Hooks API-ja. Zatim enumeracija koji omogućuju definiranje skupa

imena za numeričke vrijednosti i naposlijetku generičke tipove koji omogućavaju stvaranje klasa i funkcija s različitim tipovima podataka pritom pazeći na sigurnost provjerom ispravnosti tipova prilikom prevođenja izvornog koda.

Programski kod 2.2: TypeScript interface

```
export interface User {
  id: number;
  ime: string;
  prezime: string;
  email: string;
  brojMobitela: string;
  ovlast: string;
  aktivan: boolean;
}
```

2.4 Azure

Kao što je spomenuto u uvodu, u svrhu realizacije rada korištene su plaćene Azure usluge: Azure kontejner i Azure Blob Storage. Napomena da se usluge kontejnera plaćaju po načelu pretplate pod nazivom “Pay as you go” koja besplatno nudi Azure servise u ograničenom obliku do dvanaest mjeseci. Azure kontejner u ovom slučaju nije bio besplatan, ali Azure Blob Storage je, prvih dvanaest mjeseci pod uvjetom da se ne prelazi iskorišteni prostor za pohranu od dvanaest gigabajta.

Azure je jedna od najvećih javno dostupnih platformi za računalstvo u oblaku (engl. cloud) na svijetu napravljen od strane Microsoft-a [9]. Nudi obilan raspon usluga i alata za razvijanje, implementiranje i upravljanje aplikacijama te infrastrukturom u oblaku. Pod to spadaju i resursi računala, baze podataka, usluge interneta, alati za analizu, umjetnu inteligenciju i ostale resurse koji omogućuju tvrtkama i pojedinim korisnicima ubrzavanje i skaliranje aplikacija i usluga.

2.4.1 **Azure kontejner**

Azure kontejner je usluga za implementaciju te upravljanje aplikacijama u kontejnerima u oblaku. Kontejneri su jedinični spremnici koji omogućuju izolaciju aplikacija i time osiguravajući njihovo brže stvaranje, izvođenje, upravljanje i skaliranje. Potrebni su dodatni kako bi se kontejneri realizirali poput Docker kontejnera ili Kubernetesa. U ovom radu korišten je Docker kontejner.

Za potrebe aplikacije, Azure kontejner omogućuje javnu dostupnost putem API poziva preko URL-a. Korisnici aplikaciji mogu pristupiti putem internetskog preglednika ili drugih klijentskih aplikacija.

2.4.2 **Azure Blob Storage**

Azure Blob Storage je usluga za pohranu strukturiranih i nestrukturiranih podataka u oblaku. Podržava razne vrste podataka poput slika, videozapisa, cijelih bazi podataka i ostalih podataka. Kako je online i skalabilno, omogućuje visoku dostupnost ali i sigurnost kroz autorizaciju i autentifikaciju, šifriranje podataka u mirovanju i tijekom prijenosa, postavljenog zaštitnog mehanizma protiv napada poput DDoS i pružanja opcija za sigurnosno kopiranje i oporavak [10].

U ovom radu svrha mu je pohraniti korisnikove datoteke te im omogućiti pristup prilikom preuzimanja.

2.5 HTML

HTML je prezentacijski jezik za označavanje sadržaja web aplikacija održavan od strane World Wide Web Consorrtium (W3C). Koristi se za stvaranje elemenata i struktura na web aplikacijama te omogućava prikaz teksta, slike, videa i drugih medija pomoću osnovnog građevnog elementa (tag) koji ujedno govori kako nešto na aplikaciji treba biti prikazano [4].

HTML podržava i poveznice koji su osnova za navigaciju na internetu, web obrasce koji služe za prikupljanje korisničkih i ostalih informacija i najčešće se za stil kombinira s CSS-om detaljnije pojašnjenim u poglavlju 2.6.

2.6 CSS

CSS je stilski jezik za web aplikacije. Preciznije koristi se kako bi se definirale boje, fontovi, izgled i raspored elemenata i struktura [5].

Obično se odvaja u zasebne datoteke s ekstenzijom .css i istog naziva radne datoteke u ovom projektu s .tsx ekstenzijom prema konvenciji kako bi dizajn bio odvojen od logike i pri tom bolje čitljiv, iako je moguće pisati stil direktno u HTML elemente koristeći atribut '*style'*. Kako bi se CSS primijenio na određeni HTML element koristi se selektori unutar CSS datoteke. Uz selektore dolaze i svojstva i vrijednosti koji određuju kako će se element prikazati na aplikaciji. Dodatno podržane su mogućnosti korištenja pseudo elemenata i pseudo klasa, animacija i tranzicija, podrška za fontove i tehnika za stvaranje složenih rasporeda, neki aspekti CSS-a prikazani su u slijedećem programskom kodu.

Programski kod 2.3: CSS gumb

```css
.button {
  z-index: 2;
  position: relative;
  width: 100%;
  border: none;
  background-color: #3f72af;
  height: 30px;
  color: white;
  font-size: 1em;
  font-weight: 500;
  letter-spacing: 1px;
  margin: 10px;
  cursor: pointer;
}


.button:hover {
  background-color: #112d4e;
}
```

U programskom kodu 2.3. vidi se dohvaćanje elementa "button" pomoću selektora "class", napisan kao točka, i imena zadanog u HTML elementu pomoću atributa '*className'*. Zatim dohvaćenom gumbu dodajemo svojstva i vrijednosti kao što je širina (npr. 'width: 100%') što znači da će širina ovog gumba će biti 100% u odnosu na roditelja u kojem se nalazi. Važno je napomenuti da se CSS pravila primjenjuju u kaskadnom redoslijedu pri čemu se kasnija pravila primjenjuju prema vrsti i važnosti što ujedno omogućuje nasljednost svojstava. Naposlijetku vidi se i korištenje pseudo klase ':hover' koja

omogućuje izvršavanje CSS svojstva i vrijednosti nakon korisnikovog prelaska kursorom preko elementa “button”, u ovom slučaju mijenjamo pozadinsku boju gumba.

2.7 Alati za dizajn

U ovom potpoglavlju opisuje se kako su dodani i korišteni alati prilikom izrade ovog rada.

2.7.1 Ant Design

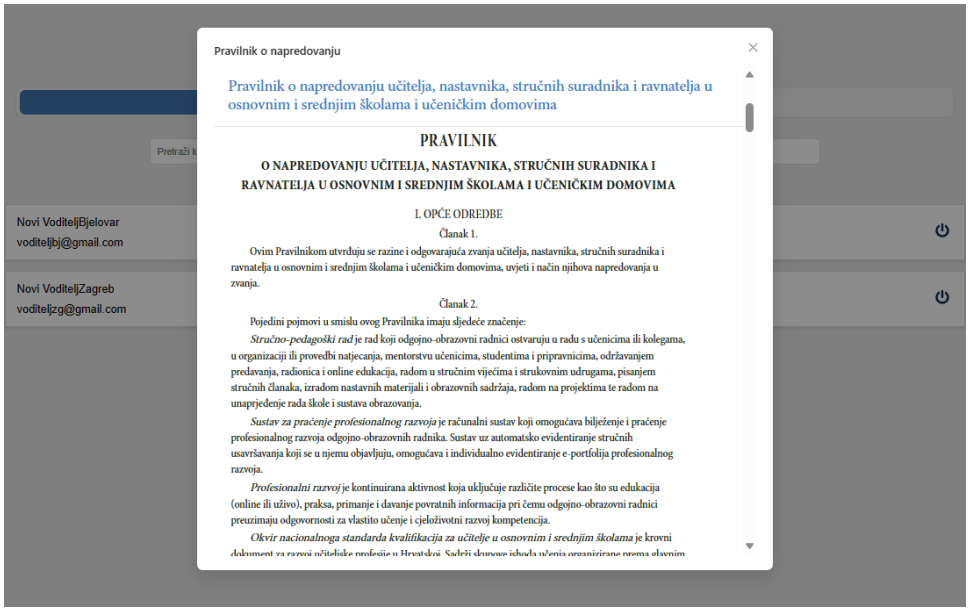
Ant Design je popularna biblioteka za dizajn i komponente u Reactu razvijena od strane Alibabe Group. Pruža raznolik izbor gotovih komponenata i stilova kao što su gumbi, obrasci, tablice, kartice, *modali* itd. Na taj način znatno ubrzava razvoj web aplikacija [6].

Programski kod 2.4: Uvoz i korištenje Modal komponente

```
import { Modal } from "antd";

...
<Modal
        title="Pravilnik o napredovanju"
        visible={isModalOpen}
        onOk={handleOk}
        onCancel={handleCancel}
        width={820}
        bodyStyle={{ height: "700px" }}
        footer={null}
      >
        <iframe
          src="https://narodne-
novine.nn.hr/clanci/sluzbeni/2019_07_68_1372.html"
          width="100%"
          height="100%"
          frameBorder="0"
          style={{ pointerEvents: "auto" }}
        />
</Modal>
```

Iako su stilovi unaprijed definirani, moguće ih je prepraviti s vlastitim stilovima. U ovaj projekt Ant Design paket dodao je naredbom ‘npm install antd’ preko naredbenog retka unutar Visual Studio Codea. U komponenti sučelja za korištenje Ant Design komponenti

potrebno ih je prvo uvesti ključnom riječi ‘*import’* zatim imenom komponente koju želimo koristiti (npr. ‘{ Button }’) zatim ključna riječ ‘*from’* i naziv biblioteke ‘antd’. U prethodnom programskom kodu prikazuje se uvoz i korištenje gotove Ant Design komponente “Modal”. U programskom kodu 2.4. vidi se naredba za unos *modala* iz biblioteke ‘antd’ i iskorištenu komponentu <Modal />. Unutar poziva komponente vidi se neke atribute i njihove vrijednosti koje su odgovorne za određene funkcionalnosti (npr. ‘OnCancel={handleCancel}’ će prilikom korisnikovog zatvaranja *modala* pozvati metodu “handleClose” koja će zatvoriti *modal*). U navedenom kodu mogu se primijetiti atributi poput ‘width=”100%”’ koji su nešto drugačije sintakse od CSS-a, ali jednako tako odgovorni za izgled *modala*. Unutar *modala* imamo i komponentu <iframe> sa svojim atributima koja služi za uključivanje web stranice s URL-om ‘https://narodne-novine.nn.hr/clanci/sluzbeni/2019_07_68_1372.html’ unutar *modala*. Krajnji rezultat vidljiv je na slici 2.1.



Slika 2.1: Modal sa stranicom pravilnikom o napredovanju

2.7.2 **UIVerse**

UIVerse je internet stranica s brojnim gotovim UI elementima otvorenog koda kreirana i održavana od strane Pixel Galaxies [7].

Struktura elemenata je napravljena u HTML-u, a stilovi su napravljeni ili s CSS-om ili Tailwind-om. Neki od elemenata koje pružaju su *checkboxes*, *cards*, *inputs*, *forms*, *buttons*, *toggle switches*, *loaders* i *radio buttons*.

Elemente nije potrebno instalirati putem naredbenog retka već je dovoljno samo kopirati pruženi kod. Elemente je također moguće prilagoditi po želji.

2.7.3 **Font Awesome**

Font Awesome je popularna i besplatna biblioteka ikona koja omogućuje implementaciju ikona u web stranicama i aplikacijama napravljena od strane Fonticons [8]. Ikone su skalabilne i lako prilagodljive putem CSS-a.

U ovom radu, Font Awesome ikone dodane su u Visual Studio Code preko naredbe ‘npm install react-icons' što nam omogućuje da u željenoj komponenti uvezemo ikonu koju želimo preko ključne riječi ‘*import*’ zatim imena ikone (npr. ‘{FaHome}’) zatim ključna riječ ‘*from*‘ te naposlijetu biblioteku ‘react-icons/fa’. U nastavku koda, pozove se u komponenti slijedećom sintaksom <FaHome /> što rezultira ikonom kućice.

3. REALIZACIJA APLIKACIJE

U ovom poglavlju detaljno je opisan proces izrade web aplikacije ovog rada kroz podnaslove.

U potpoglavlju 3.1. opisana je specifikacija aplikacije koja uključuje funkcionalne, nefunkcionalne zahtjeve i željene ciljeve.

Potpoglavlje 3.2. opisuje kako je projekt inicijalno postavljen i kako je odabrana i organizirana struktura direktorija i mapa u svrhu organizacije koda.

Potpoglavlje 3.3. opisuje komponente korisničkog sučelja kao što su *login*, *register*, ostale komponente bazirane na ulozi korisnika i druge relevantne komponente.

Potpoglavlje 3.4. govori kako je oblikovana CSS struktura kako bi se lakše primijenila kroz cijelu aplikaciju i time održala konzistentan izgled.

Potpoglavlje 3.5. opisuje način autentifikacije korisnika prilikom prijave ili registracije u aplikaciju te kako se korisnika temeljeno na ovlasti ograničava na uporabu samo onih komponenti sučelja za koje ima dopuštenje. Opisuje kako se upravlja stanjem i uporabom informacija kroz cijelu aplikaciju koristeći stanja i *Hook*-ova. Dodatno govori kako je implementirana navigacija koristeći navigacijske *Hook*-e i *React Router*.

3.1 Specifikacija

Cilj rada bio je izraditi repozitorij u obliku web aplikacije koja prvenstveno omogućuje svim korisnicima kreiranje računa te prijavu u repozitorij pod uvjetima da email i OIB već ne postoje te lozinka zadovoljava određen standard sigurnosti. Nadalje tri su razine korisnika: nastavnik, voditelj ustanove i administrator aplikacije. Različite razine korisnika imaju pristup različitim resursima i funkcionalnostima.

Nastavnici prilikom prijave u aplikaciju dobivaju zaseban ekran u kojem odmah imaju pregled na inicijalno praznu listu učitanih datoteka zajedno s mogućnošću pregleda detalja vezanih uz datoteke poput validiranosti ili imena, opciju preuzimanja, brisanja te pohranjivanja novih datoteka. Nakon učitavanja datoteke pojavljuje se izbornik iz kojeg je obavezno odabrati kategoriju datoteke kako bi se mogli dodijeliti pripadajući bodovi. Kao dodatni alati postoji navigacijska i alatna traka. U navigacijskoj traci mogu pronaći gumb za navigaciju na početni zaslon te padajući izbornik s opcijama odjave iz profila i opcijom

‘Informacije’ gdje mogu pronaći pravilnik o napredovanju nastavnika. U alatnoj traci mogu pronaći opciju ‘Profil’ koja nastavnicima omogućuje ažuriranje trenutnih osobnih informacija profila te brisanje računa.

Slijedeća uloga je voditelj ustanove. Svrha ove uloge je pregled svih pripadnika ustanove te provjera i potvrda datoteka nastavnika. Prilikom prijave dobivaju pripadajući ekran ekran. Način takvog rutiranja detaljnije je pojašnjen u programskom kodu 3.1. Početni sadržaj za voditelja ustanove sastoji se od liste korisnika koji pripadaju njegovoj ustanovi, što je omogućeno putem pozivanja komponente šaljući ID ustanove koju je voditelj odabrao prilikom registracije. U toj listi voditelj može pretraživati nastavnike prema njihovim imenima ili email-om putem tražilice (engl. s*earch bar*) te ih filtrirati putem ponuđenih gumbova ‘Aktivni’ i ‘Neaktivni’ što u listi prikazuje aktivne ili neaktivne nastavnike. Uz navedeno u listi korisnika voditelj ima ikonu mape iz biblioteke Font Awesome pojašnjene u potpoglavlju 2.6.3. koja kada se na nju pritisne izbaci *modal* sa svim datoteka odabranog korisnika. U tom *modalu* voditelju je omogućeno preuzimanje, validiranje te filtriranje datoteka prema statusu validnosti. Dodatno, voditelj također ima identičnu navigacijsku i alatnu traku kao i nastavnik s identičnim mogućnostima.

Posljednja uloga je administrator aplikacije. Svrha ove uloge je pregledati, brisati, unositi te ažurirati nastavnike, voditelje ustanova te same ustanove. Za početni ekran primijenjen je isti princip rutiranja i prikazivanja sadržaja prethodno spomenutog. Sadržaj se sastoji od četiri kartice: Ustanove, Admini, Profesori i Datoteke napravljene pomoću UIVerse alata pojašnjenog u potpoglavlju 2.6.2. Kartica Ustanove prebacuje administratora aplikacije na rutu /schools u kojem se nalazi lista ustanova. Svaki zapis te liste sadrži informacije o ustanovi te opcije za kontrolu poput brisanja, ažuriranja i pregledavanje članova ustanove. Opcija za pregledavanje članova ustanove dolazi s filterima za sve korisnike, samo voditelje ustanove i samo nastavnike te s dodatnim filterima za aktivne ili neaktivne. Naknadno moguće je korisnike ažurirati, obrisati i pregledati. Kartica Admini administora odvodi na stranicu /admins koja dolazi s popisom svih administratora, također s omogućenim CRUD operacijam te filterima za aktivne i neaktivne voditelje. Kartica Profesori vodi na stranicu /professors koja dolazi s listom nastavnika nad kojima su također omogućene CRUD operacije i filteri za aktivne ili neaktivne nastavnike. Naposlijetku kartica Datoteke vodi na stranice /files, koja nudi listu s pregledom svih datoteka s pravilnika o napredovanju nastavnika nad kojima su omogućene CRUD (*Create*, *Read*, *Update*, *Delete*) operacije. Uz svaku datoteku vidljivo je koliko bodova nosi te koji joj je naziv.

3.2 Postavljanje projekta i arhitektura

Aplikacija je kreirana pomoću naredbe u naredbenom retku VS code-a ‘create-react-app' koja automatski postavlja osnovni okvir za razvoj React aplikacije. Dijelovi osnovnog okvira su: src (*source*), public i node_modules direktoriji, webpack i babel konfiguracije, razvojni server, početni HTML, NPM skripte i datoteka package.json.

- Src - sadrži izvorni kod aplikacije
- Public - sadrži javno dostupne datoteke
- Node_modules - sadrži instalirane biblioteke i pakete
- Webpack konfiguracija - alat za spajanje JavaScript datoteka u jednu ili nekoliko datoteka što olakšava prenošenje i učitavanje JavaScript koda
- Babel konfiguracija - alat koji omogućava transpilaciju JavaScript koda
- Razvojni server - automatski osvježava aplikaciju prilikom promjene u kodu
- Početni HTML - pruža početnu stranicu kao početak aplikacije, ujedno programer može vidjeti kako funkcionira React aplikacija
- NPM skripte - sadrži skripte za upravljanje aplikacijom poput ‘npm start’ koja pokreće aplikaciju

Dodatni instalirani paketi su: typescript , react-router-dom (React router), react-icons, react-select i axios. React-select je paket s komponentom padajući izbornik, a ostali paketi opisani su u poglavlju 2. Svi instalirani paketi nalaze se u datoteci package što je vidljivo iz slike 3.1.

Slika 3.1: Paketi aplikacije

Direktorij „src“ sadrži poddirektorije components, context, css, hooks, modals, pages i datoteke App i index.

Poddirektorij components u kojoj se nalaze datoteke Nav, Sidebar, Notification, RequireAuth, SchoolList, UserList, FileList, služi za organiziranje komponenti unutar stranica.

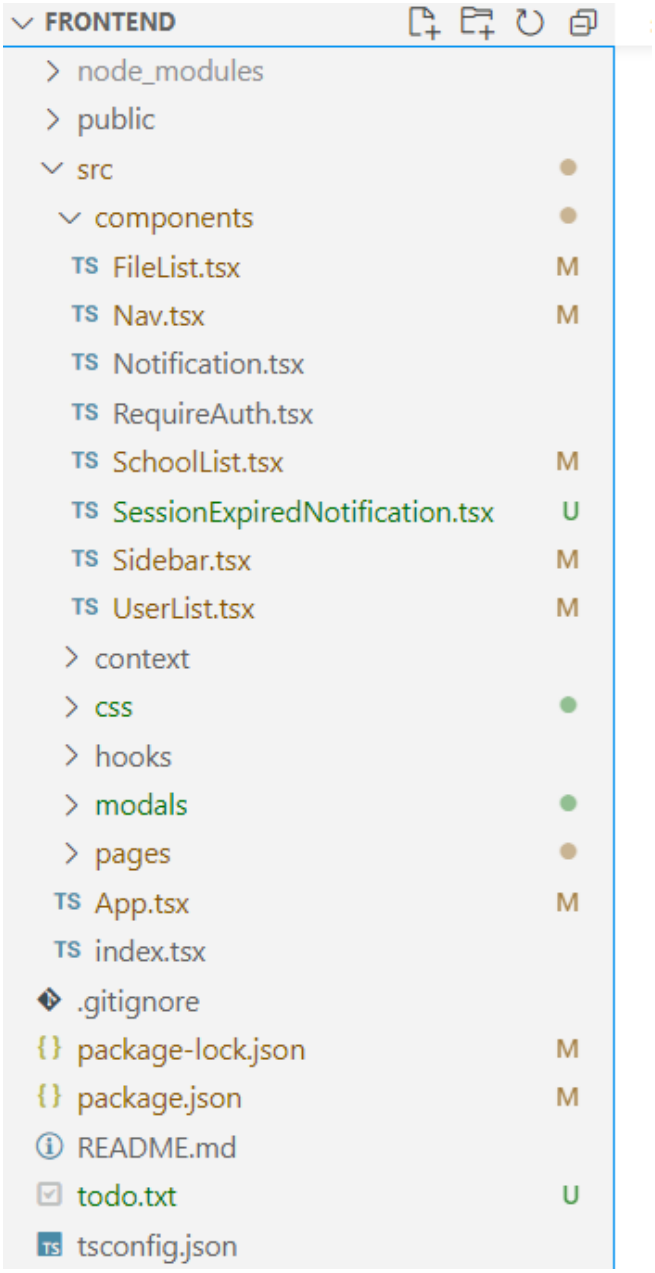
Poddirektorij context služi za postavljanje konteksta aplikacije, sadrži AuthProvider i SessionExpiredNotification.

Poddirektorij hooks služi za pohranu hook-a useAuth i useFileHandler.

Poddirektorij modals sadrži *modale* koji se mogu smatrati kao stranica unutar stranice. Datoteke s *modalima* su: AddActivityModal, AddSchoolModal, AddAdminModal, EditActivityModal, EditSchoolModal, EditUserModal, FileUploadModal, SchoolUserModal i UserFileModal.

Poddirektorij pages sadrži stranice aplikacije kroz koje korisnik može navigirati osim Layout. Ostale stranice su: Account, Admins, Developer, Files, Home, Login, Missing, Professors, Register, SchoolAdmin, Schools i Unauthorized.

Naposlijetku poddirektorij css sadrži dizajn svih navedenih komponenti i stranica te u nastavku na slici 3.2. vidljiva je cijela arhitektura navedenih direktorija i poddirektorija aplikacije.



Slika 3.2: Arhitektura aplikacije

3.3 Komponente aplikacije

Ovo poglavlje započinje s početnom stranicom - Login. Login korisnicima omogućuje prijavu u aplikaciju putem unosa email-a i lozinke putem API poziva prema Azure kontejneru. Nakon uspješne prijave, korisnika se navigira na odgovarajuću početnu stranicu ovisno o njegovoj ulozi koristeći naredbu useNavigate() iz react-router-dom biblioteke.

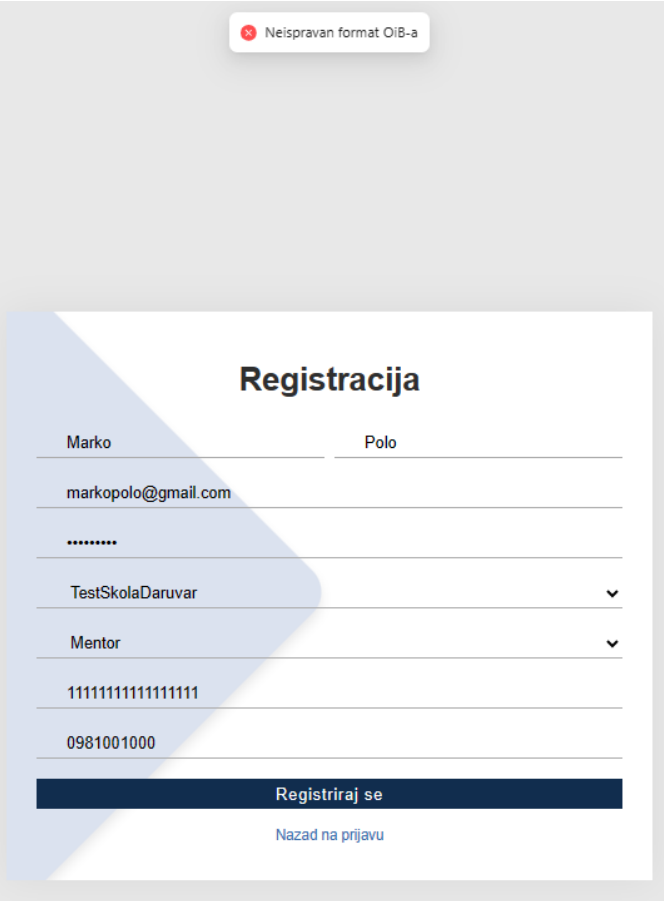
Programski kod 3.1: Funkcija za prijavu i pohranu informacija korisnika

```
import { Link, useNavigate, useLocation } from "react-router-dom";

const Login: React.FC = () => {
  const navigate = useNavigate();

const handleSubmit = async (event: React.FormEvent<HTMLFormElement>) => {
    event.preventDefault();

    // Login data object
    const loginData = {
      email: email,
      password: password,
    };

    try {
      const response = await axios.post(apiUrl, loginData, {
        headers: {
          "Content-Type": "application/json",
        },
      });

      if (response.status === 200) {
        const data = response.data;
        console.log("Data: ", data);

        const userToken = data.token;
        const roles = data.ovlast;
        const userId = data.id;
        const ustanovaId = data.skolskaUstanovaId;

        if (roles.includes("voditeljUstanove")) {
          login(userToken, roles, userId, ustanovaId);
          navigate("/admin", { replace: true, state: { userToken } });
        } else if (roles.includes("korisnik")) {
          login(userToken, roles, userId, ustanovaId);
          navigate("/", { replace: true, state: { userToken } });
```

```
        } else if (roles.includes("admin")) {
          login(userToken, roles, userId, ustanovaId);
          navigate("/developer", { replace: true, state: { userToken }});
        }
      } else {
        console.log("Login failed");
      }
    } ...  };
```

U programskom kodu 3.1. može se vidjeti uvoz useNavigate funkcije te ako je axios poziv prema Azure kontejneru bio uspješan, punimo Login funkciju detaljnije pojašnjenu u poglavlju 3.5. s podacima korisnika koji se prijavljuje. Zatim korisnika ovisno o njegovoj ulozi navigiramo na pripadajući ekran prosljeđujući userToken što je JWT zaslužan za autentifikaciju i autorizaciju korisnika. Kada userToken istekne, logika u App komponenti prebacuje korisnika na Login komponentu, detaljnije pojašnjeno u poglavlju 3.5.

Slijedeća komponenta je Register, gdje korisnik može kreirati svoj račun s informacijama koje se od njega traže informacije ime, prezime, email, lozinka, OIB, ustanova kojoj pripada (opcionalno), zvanje i broj mobitela. Nad podacima koje korisnik sam upisuje postavljene su sigurnosne provjere koje korisniku izbacuju poruku na sučelju ako je napravio unos koji krši pravilo provjere i potvrde. Na taj način minimizirani su potencijalni sigurnosni problemi poput *SQL injection*. Sa slike 3.3. vidljivo je kako izgleda komponenta Register i kako izgleda kada korisnik unese nešto što ne prolazi provjeru i potvrdu.



Slika 3.3: Register komponenta te provjera i potvrda unosa

Ako se je prijavio nastavnik, on dolazi na stranicu Home gdje ima pregled nad svojim datotakama i mogućnost pohrane novih. Pregled datoteka omogućen je kroz komponentu FileList, a pohrana je omogućena kroz *modal* FileUploadModal. Komponentu i *modal* potrebno je samo pozvati uz prosljeđivanje informacija koje komponenta i *modal* zahtjevaju, unutar funkcije *return* koja je obavezna u svim komponentama.

U programskom kodu 3.2. vidi se uvoz komponenti FileUploadModal i FileList, koje se pozivaju uz parametre poput userId i userToken što komponentama daje do znanja o kojem korisniku se radi te ovisno o tome mogu povući informacije namijenjene za tog korisnika s Azure kontejnera ili Azure Blob Storage-a.

U FileList komponenti nalazi se axios API poziv prema Blob Storage-u za dohvaćanje za prikaz ili preuzimanje datoteka te axios API poziv prema Azure kontejneru za brisanje datoteka. Osim poziva imamo i sistem pretraživanja te filtracije kroz tražilicu i gumbe za odabir opcije filtracije ‘odobreno’ ili ‘neodobreno’ što se odnosi na provjeru i potvrdu datoteka.

Programski kod 3.2: Home component s pozivom drugih komponenata

```
import FileUploadModal from "../modals/FileUploadModal";
import FileList from "../components/FileList";
import useAuth from "../hooks/useAuth";

const Home = () => {

  const { userToken, userId } = useAuth();

  ...

  return (
    <main className="main">
      <FileList
        userId={userId}
        userToken={userToken}
        fileUploaded={fileUploaded}
      />
      <FileUploadModal
        isOpen={isFileUploadModalOpen}
        onClose={() => setIsFileUploadModalOpen(false)}
        userToken={userToken}
        onUploadSuccess={handleFileUploadSuccess}
      />

      <button
        type="button"
        className="toggle-button"
        onClick={toggleFileUploadModal}
      >
        Upload File
      </button>
    </main>
  );
};
```

Programski kod 3.3: Filtracija prikaza podataka

```
...
<input
        className="listSearch"
        type="text"
        placeholder="Pretraži korisnike"
      />

<ul className="fileList">
        {files.map((file, index) => (
          <li className="fileItem" key={index}>
            <div className="fileItemDetails">
              <div className="fileItemName">{file.naziv}</div>
              <div className="file-date">Created: {file.datumKreiranja}
</div>
            </div>
            <span
              className={file.odobreno ? "approved" : "not-approved"}
              title={file.odobreno ? "Odobreno" : "Na čekanju"}
            >
              {file.odobreno ? <FaCheck /> : <FaTimes />}
            </span>
...
```

U programskom kodu 3.3. prikazan je način filtracije prikaza datoteka. Dohvaćena datoteka u sebi sadrži informaciju da li je odobrena, ta se informacija stavlja u ternarnu (?) provjeru ako je istinito izvrši prvu opciju, a ako nije izvrši drugu opciju (:) Po istom principu funkcionira filtracija kroz cijelu aplikaciju.

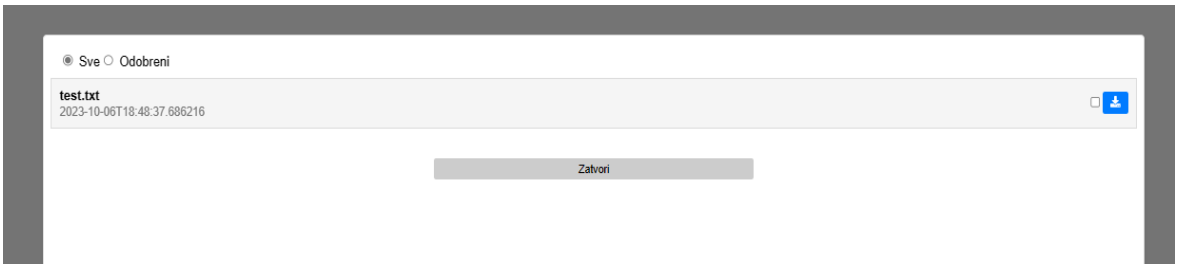
Nastavnik zatim ima pristup navigacijskoj traci iliti komponenti „Nav“. Ranije spomenuta kućica ima različitu ulogu za svaku razinu ovlasti, tj. vodi korisnike na pripadajući polazni ekran. Padajući izbornik nudi opciju odjave koja obriše spremljene podatke trenutnog korisnika i prebacuje ga na ekran za prijavu. Dodatno nudi opciju prikaza *modala* s pravilnikom sa slike 2.1. Naposlijetku izbornik je moguće zatvoriti pritiskom lijeve tipke miša bilo gdje na ekranu a da nije sam izbornik, takva funkcionalnost napravljena je i za sve *modale* aplikacije.

Dodatno s lijeve strane nalazi se alatna traka u kojoj stoji opcija ‘Profil’ koja vodi na stranicu Account. Prilikom ulaska u Account prikazan je obrazac sa poljima za unos (engl. *Input field*) koja su automatski popunjena s podacima korisnika. To je učinjeno kako bi korisnik mogao lakše ažurirati podatke bez da pamti što je unio prilikom registracije. Takav

princip za lakše ažuriranje informacija primijenjen je bilo gdje gdje je informacije moguće ažurirati, krajnji rezultat vidljiv je sa slike 3.4.

Slika 3.4: Komponenta Account i automatska popuna informacija

Prilikom prijave voditelja ustanove, pojavljuje se sadržaj komponente SchoolAdmin koja zove komponentu UserList s parametrima koji govore da treba prikazati samo one korisnike koji pripadaju ustanovi voditelja. U listi nalaze se opcije za deaktivaciju korisnika i *modal* UserFileModal. UserFileModal voditelju daje uvid u sve datoteke odabranog korisnika te pregled statusa datoteke zajedno s opcijom za preuzimanje. Voditelj može datoteku odobriti ili ostaviti na čekanju. Dodatno u listi s korisnicima nalaze se filteri za pretraživanje te pregled svih korisnika, voditelja ili nastavnika sa statusima aktivni ili neaktivni. Izgled *modala* za upravljanje provjerom i potvrdom datoteka vidljiv je na slici 3.5.



Slika 3.5: UserFileModal komponenta za provjeru i pregled datoteka

Prilikom prijave administratora aplikacije, prikazuje se sadržaj komponente Developer. Sadržaj se sastoji od četiri kartice UIVerse dizajna koje nude pregled škola, admina,

profesora i datoteka. Kartica škole vodi na sadržaj komponente Schools gdje se nalazi SchoolList komponenta koja sadrži popis svih škola s opcijama za ažuriranje, brisanje te otvaranje *modala* SchoolUserModal u kojem se nalazi popis svih pripadajućih voditelja ustanove te nastavnika i nastavnicima je moguće pregledati sve datoteke, odobriti ih i preuzeti i naposlijetku *modala* AddSchoolModal preko kojeg se može registrirati novu školu. Kartica Admini vodi na sadržaj komponente Admins koja sadrži popis svih administratora i tu ih je moguće pregledavati i deaktivirati. Kartica Profesori vodi na sadržaj komponente Professors u kojoj su prikazani svi nastavnici, ponovno s mogućnošću deaktivacije nastavnika, pregleda, preuzimanja i provjere datoteka. Kartica Datoteke vodi na sadržaj komponente Files gdje se nalazi popis datoteka potrebnih za napredovanje nastavnika. Iste je moguće ažurirati, brisati te dodavati nove preko pripadajućih *modala*, zajedno s opcijom pretraživanja preko tražilice. Svaka datoteka sadrži ime te iznos bodova.

Komponenta Unauthorized pojavljuje se kada korisnik pokušava pristupiti resursu za kojeg nema odobrenje te mu je ponuđena opcija da se vrati nazad.

Komponenta Missing pojavljuje se kada korisnik pokušava pristupiti resursu koji ne postoji, najčešće putem prepravljanja ili dodavanja na postojeću URL te mu se pojave opcije za ponovnu prijavu ili povratak nazad ovisno o integritetu JWT.

Komponenta Layout odgovorna je za omatanje (engl. *wrap*) komponenti s komponentama Nav i Sidebar te sadrži kontejner u kojem se renderiraju djeca, odnosno pozvane komponente.

Komponenta SessionExpiredNotification pojavljuje se prilikom isteka JWT te daje korisniku do znanja da je potrebno ponovno se prijaviti.

Komponente RequireAuth i AuthProvider odgovorne su za autorizaciju i autentikaciju te su detaljnije pojašnjeni u poglavlju 3.5 skupa s komponentama App i indeks koje su odgovorne za način rutiranja.

3.4 Izrada CSS strukture

CSS datoteka App sadrži stil za korijenski element unutar indeks komponente. Označen je s #root i njegova struktura utiče na strukturu daljnjeg CSS-a zbog spomenutog roditeljskog i kaskadnog učinka u CSS-u.

Zatim u Layout css datoteci definirani su stilovi za kontejner kojeg djeca poprime i tu je namješteno da sav sadržaj bude centriran po x-osi stoga ne treba svaku komponentu pozicionirati na ekranu. Također sadrži stilove za navigacijsku i alatnu trake pošto su prisutni tamo gdje je i Layout komponenta što eliminira potrebu za dodatnim css datotekama.

Kod ispisa lista korisnika, datoteka i ustanova, korištena je vrlo slična struktura što omogućuje uporabu različitim komponentama. Unatoč sličnosti, napravljene su različite CSS datoteke zbog mogućnosti proširenja funkcionalnosti aplikacije u budućnosti čemu bi odvojeni CSS bolje odgovarao.

3.5 Upravljanje stanjima i rutama

Za upravljanje stanjima u aplikaciji korištene su komponente RequireAuth, AuthProvider.

AuthProvider komponenta u sebi sadrži login funkciju koja se puni podacima bitnim za korisnika poput njegovog ID-a, ID-a ustanove, uloge te JWT. Zadaća login funkcije je napuniti lokalnu pohranu (engl. *Local storage*) u web pregledniku sa spomenutim informacijama kako bi se mogli koristiti kroz aplikaciju. Lokalna pohrana koristi se kako bi podaci o korisniku bili ustrajni kroz nepredvidive promjene u aplikaciji ili mreži. Podaci su vidljivi preko *inspect* opcije unutar web preglednika što je za potrebe aplikacije u redu jer ti podaci nisu osjetljivi.

RequireAuth služi za dodjelu prava pristupa resursima ovisno o ulozi korisnika dobivenoj iz AuthProvider hook-e.

Programski kod 3.4: Funkcija za dodjelu prava pristupa resursima

```
import { useLocation, Navigate, Outlet } from "react-router-dom";
import useAuth from "../hooks/useAuth";

const RequireAuth = ({ allowedRoles }: { allowedRoles: string[] }) => {
  const { userToken, roles } = useAuth();
  const location = useLocation();

  // Check if the user's roles intersect with the allowed roles
  const isAuthorized = roles
```

```
    ? allowedRoles.some((role) => roles.includes(role))
    : false;

  return userToken && isAuthorized ? (
    <Outlet />
  ) : userToken ? (
    <Navigate to="/unauthorized" state={{ from: location }} replace />
  ) : (
    <Navigate to="/login" state={{ from: location }} replace />
  );
};

export default RequireAuth;
```

U programskom kodu 3.4. vidljiv je *import* useAuth *custom hook*. *Custom hook* služi kao privatni spremnik kojeg se može pozivati u bilo kojoj komponenti te uzimati njegov sadržaj, u ovom slučaju informacije o korisniku iz AuthProvider komponente. Uz lokalnu pohranu nije potrebno imati ovu hook-u ali napravljeno je u svrhu proširenja aplikacije u budućnosti gdje bi useAuth imala bolju primjenu. Nakon što useAuth dostavi informacije o korisniku, funkcija RequireAuth može provjeriti ulogu korisnika te mu dodijeliti pripadajuće dopuštenje. Ako se utvrdi da korisniku nije dopušten pristup resursu, navigira ga se na spomenutu Unauthorized komponentu.

Implementacija autoriziranog pristupa je u App kompomenti.

Programski kod 3.5: Glavna funkcija za prikaz sadržaja i rutiranje

```
import { Routes, Route, useNavigate } from "react-router-dom";
import { isExpired } from "react-jwt";
import { useEffect } from "react";
import RequireAuth from "./components/RequireAuth";
...

// Handle expired token
  useEffect(() => {
    if (isMyTokenExpired) {
      navigate("/login", { state: { isSessionExpired: true } });
      localStorage.clear();
    }
  }, [isMyTokenExpired]);

return (
```

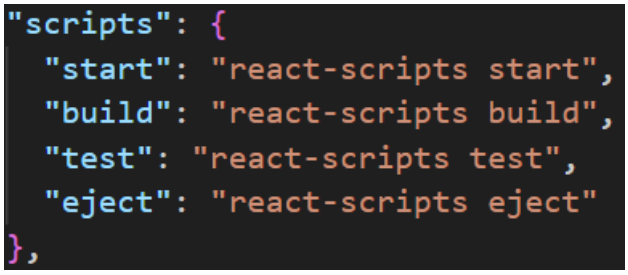
```
    <Routes>
      {/* public routes */}
      <Route path="login" element={<Login />} />
      <Route path="register" element={<Register />} />
      <Route path="unauthorized" element={<Unauthorized />} />

      {/* admin */}
      <Route element={<RequireAuth allowedRoles={["admin"]} />}>
        <Route element={<Layout />}>
          <Route path="/developer" element={<Developer />} />
          ...
        </Route>
      </Route>

      {/* admin and korisnik */}
      <Route element={<RequireAuth allowedRoles={["admin", "korisnik"]}
/>}>
        <Route element={<Layout />}>
          <Route path="/" element={<Home />} />
        </Route>
      </Route>
      ...
  );
}
```

U programskom kodu 3.5. vidljivo je korištenje react-jwt biblioteke koja pruža metodu isExpired() koja provjerava rok trajanja JWT i ukoliko je istekao pokreće se useEffect *hook* koji korisnika navigira na *login* stranicu. Zatim uvedena je biblioteka *react-router-dom* za potrebe rutiranja. Rute su kategorizirane po ulozi a preostale su javne. Prilikom pozivanja RequireAuth vidljivo je da se prosljeđuje ovlast korisnika koji ima pristup u omotanim rutama. Ako se ovlast ne podudara s rutom kojoj pripada, pristup je odbijen i korisnika se šalje na Unauthorized komponentu, isto princip vrijedi za ostale uloge.

3.6 Implementacija aplikacije

U sklopu projekta, aplikacija je razvijena kako bi omogućila komunikaciju s Azure servisima. Za lokalni razvoj i testiranje aplikacije, potrebno je osigurati pristup izvornom kodu aplikacije na vlastitom računalu. Nakon preuzimanja koda, potrebno je preuzeti i instalirati Node.js i npm. Zatim je potrebna instalacija svih zavisnosti koja se postiže izvršavanjem naredbe naredbenog retka '*npm install*' unutar direktorija React aplikacije. Nakom postavljanja svih zavisnosti, aplikacija se pokreće naredbom naredbenog retka '*npm*

start' koja automatski otvara web aplikaciju u pregledniku. U slučaju da se web aplikacija ne otvara, moguće ju je ručno otvoriti unosom slijedećeg URL-a u preglednik: 'http://localhost:3000'. Dodatno URL se može konfigurirati po želji korisnika u datoteci package.json prikazano na slikama 3.6 i 3.7 S konfiguracijom na slici 3.10 URL poprima oblik 'http://localhost:8080'.



Slika 3.6: Originalna konfiguracija URL-a

```
"scripts": {
  "start": "react-scripts start --host 0.0.0.0 --port 8080",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
```

Programski kod 3.7: Prilagođen URL aplikacije

4. ZAKLJUČAK

U radu su opisane tehnologije, alate, uspostavljanje radnog okruženja i njihov produkt u obliku web aplikacije. Cilj rada bio je omogućiti nastavnicima, voditeljima ustanova i administratoru aplikacije pregled i kontrolu nad profilima i datotekama potrebnih za praćenje napretka nastavnika.

Kao rezultat proizašla je intuitivna web aplikacija koja temeljem uloge korisnika pruža personalizirano radno okruženje. Aplikacija omogućuje nastavnicima praćenje i unaprjeđivanje rada, dok voditelji ustanove na raspolaganju imaju alat za pregled i odobravanje napretka nastavnika. Administratori aplikacije imaju potpunu kontrolu nad svim resursima.

Unatoč uspješnom rezultatu, aplikacija dolazi uz određena ograničenja. U trenutnoj fazi komunikacija s Azure servisom odvija se putem HTTP umjesto sigurnijeg HTTPS protokola te nije postavljena da bude javno dostupna već je u lokalnom pogonu. Vrijedi razmotriti korištenje drugih tehnika za upravljanje stanjem i tokom podataka kroz komponente te izradom više modularnih predložaka.

Aplikacija pruža čvrsti temelj za budući razvoj. U daljnjem razvoju, naglasak se stavlja na praćenje kontinuiranih tehnoloških napretka i trendova, kao i na korištenje već postojećih rješenja za razvoj web aplikacija.

5. LITERATURA

[1] Microsoft Corporation. Visual Studio Code Documentation [Online]. 2023. Dostupno na: https://code.visualstudio.com/docs. (10.10.2023)

[2] Meta Open Source. React [Online]. 2023. Dostupno na: https://react.dev. (11.10.2023)

[3] Microsoft Corporation. TypeScript Documentation [Online]. 2023. Dostupno na: https://www.typescriptlang.org/docs. (12.10.2023)

[4] Web Hypertext Application Technology Working Group. HTML Introduction [Online]. 2023.

Dostupno na: https://html.spec.whatwg.org/multipage/#toc-introduction (13.10.2023)

[5] World Wide Web Consortium (W3C). Cascading Style Sheetshome page [Online]. 2023.

Dostupno na: https://www.w3.org/Style/CSS/Overview.en.html. (13.10.2023)

[6] Alibaba Group Holding Limited. Ant Design [Online]. 2023. Dostupno na: https://ant.design. (13.10.2023)

[7] Pixel Galaxies. Open-Source UI elements for any project [Online]. 2023. Dostupno na: https://uiverse.io. (13.10.2023)

[8] Fonticons, Inc. Font Awesome Documentation [Online]. 2023. Dostupno na: https://fontawesome.com/docs. (13.10.2023)

[9] Microsoft Corporation. Azure [Online]. 2023. Dostupno na: https://azure.microsoft.com/en-us. (13.10.2023)

[10] Microsoft Corporation. Azure Blob Storage [Online]. 2023. Dostupno na: https://azure.microsoft.com/en-us/products/storage/blobs/. (13.10.2023)

6. OZNAKE I KRATICE

API - *Application Programming Interface*

CRUD - *Create, Read, Update, Delete*

CSS - *Cascading Style Sheets*

DDoS - *Distributed Denial of Service*

DOM - *Document Object Model*

HTML - *HyperText Markup Language*

HTTP - *HyperText Transfer Protocol*

HTTPS - *HyperText Transfer Protocol Secure*

ID - *Identifier*

JSX - *JavaScript XML*

JWT - *JSON Web Token*

NPM - *Node Package Manager*

OiB - *Osobni Identifikacijski Broj*

OS - *Operating System*

SQL - *Structured Query Language*

UI - *User Interface*

URL - *Uniform Resource Locator*

W3C - *World Wide Web Consortium*

XML - *eXtensible Markup Language*

7. SAŽETAK

Naslov: Web aplikacija za dokumentiranje i praćenje profesionalnog napredovanja nastavnika

Ovaj rad se bavi izradom web aplikacije za praćenje profesionalnog napretka nastavnika. Aplikacija omogućuje nastavnicima, voditeljima ustanova i administratorima kontrolu nad profilima i datotekama relevantnim za praćenje napretka nastavnika.

Aplikacija je strukturirana kroz tri ključne faze: definiranje ciljeva i zahtjeva, planiranje komponenti, stranica, toka informacija i dizajna, te konačno, implementaciju.

Rezultat rada je intuitivna web aplikacija koja pruža personalizirane radne površine ovisno o korisničkim ulogama. Nastavnici mogu pratiti i unapređivati svoj rad, dok voditelji ustanova mogu pregledavati i odobravati napredak nastavnika. Administratori imaju potpunu kontrolu nad resursima.

Cilj rada bio je stvoriti aplikaciju koja olakšava praćenje profesionalnog napretka. Različite razine korisnika imaju različite ovlasti i funkcionalnosti. Aplikacija olakšava administraciju i praćenje datoteka.

Ključne riječi: web aplikacija, napredak nastavnika, korisničke uloge, kontrola resursa.

8. ABSTRACT

Title: Web Application for Documenting and Monitoring Teacher Professional Development

This paper focuses on the development of a web application designed for tracking the professional progress of teachers. The application provides teachers, institution administrators, and system administrators with control over profiles and files relevant to monitoring teachers' advancements.

The application is structured through three key phases: defining goals and requirements, planning components, pages, information flow, and design, and finally, implementation.

The outcome of this project is an intuitive web application that offers personalized workspaces based on user roles. Teachers can monitor and enhance their work, while institution administrators can review and approve teachers' progress. System administrators have complete control over resources.

The primary objective of this paper was to create an application that simplifies the tracking of professional development. Different user levels have varying authorizations and functionalities. The application streamlines file management and administration.

Keywords: web application, teacher progress, user roles, resource control

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 13.10.2023	BORNA SLIJEPČEVIĆ	Borna Slijepčević

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

BORNA SLIJEPČEVIĆ
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon _____ (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 13. 10. 2023

Borna Slijepčević
potpis studenta/ice