Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

TEMA

Seminarski rad iz predmeta Internet tehnologije

Mentor: Studenti:

Beograd, 2023.

Sadržaj

[Korisnički zahtev 1](#_Toc158325455)

[Opis sistema 2](#_Toc158325456)

[Opis slučajeva korišćenja 2](#_Toc158325457)

[Registracija korisnika 2](#_Toc158325458)

[Prijava korisnika 3](#_Toc158325459)

[Kreiranje obavestenja 4](#_Toc158325460)

[Opis arihtekture aplikacije 5](#_Toc158325461)

[Opis procesa slučajeva korišćenja 6](#_Toc158325462)

[Dijagram sekvenci – registracija korisnika 6](#_Toc158325463)

[Dijagram sekvenci – prijava korisnika 7](#_Toc158325464)

[Dijagram sekvenci – kreiranje obavestenja 8](#_Toc158325465)

[Model podataka 9](#_Toc158325466)

[Specifikacija REST API-ja 11](#_Toc158325467)

[Opis tehnologija korišćenih u aplikaciji 18](#_Toc158325468)

[React 18](#_Toc158325469)

[Laravel 18](#_Toc158325470)

[Axios 18](#_Toc158325471)

[Korisničko uputstvo 19](#_Toc158325472)

[Prikaz delova koda 27](#_Toc158325473)

[Laravel 27](#_Toc158325474)

[React 34](#_Toc158325475)

[Link ka github repozitorijumu 59](#_Toc158325476)

[Literatura 60](#_Toc158325477)

# Korisnički zahtev

Potrebno je kreirati aplikaciju koja imitira servis euprave. Potrebno je obezbediti tri korisničke uloge, a prema tome i određene funkcionalnosti:

* Neulogovani korisnik ima mogućnost pregleda početne stranice, stranice sa COVID statistikama, kao i stranice na kojoj se građani obaveštavaju o nestanku vode i struje.
* Ulogovani korisnik sa ulogom „korisnik“ ima mogućnost kreiranja zahteva za određenu uslugu koju servis euprave nudi (obnova dokumenata, rezervacija vozila i slično).
* Administratori imaju mogućnost kreiranja obaveštenja o nestanku vode ili struje. Oni su odgovorni za upravljanje uslugama koje nudi euprava. Ovo podrazumeva prikaz, ažuriranje, kreiranje i brisanje usluga. Takođe, administratori imaju mogućnost pregleda statistika vezanih za usluge.

I za administratore i za obične korisnike je potrebno obezbediti rad sa sesijom, a za korisnike je potrebno uvesti i mogućnost registracije.

# Opis sistema

## Opis slučajeva korišćenja

### Registracija korisnika

Slučaj korišćenja: Registracija korisnika na sistem Akteri:

* Korisnik
* Sistem (program) Preduslov:
* Sistem je uključen. Osnovni scenario:

1. Korisnik pristupa opciji za registraciju na sistem.
2. Sistem prikazuje formu za registraciju koja zahteva unos korisničkog imena, e-mail adrese, lozinke i potvrdu lozinke.
3. Korisnik unosi tražene podatke u odgovarajuća polja forme.
4. Korisnik potvrđuje unos i šalje podatke sistemu klikom na dugme "Registracija".
5. Sistem proverava validnost unetih podataka, uključujući proveru da li je e-mail adresa validna i da li je lozinka dovoljno bezbedna.
6. Ukoliko su svi podaci ispravni, sistem kreira nalog za korisnika i šalje potvrdu registracije na unetu e-mail adresu.
7. Sistem prikazuje poruku "Uspešno ste se registrovali na sistem. Molimo proverite svoj e-mail radi potvrde registracije." i preusmerava korisnika na početnu stranu. Alternativni scenariji: 5.1 Ukoliko su uneti podaci neispravni ili nedovoljno bezbedni:

* Sistem prikazuje odgovarajuću poruku o grešci i traži od korisnika da ispravi podatke pre ponovnog pokušaja registracije.

### Prijava korisnika

Slučaj korišćenja: Prijava korisnika na sistem

Akteri:

* Korisnik
* Sistem (program)Preduslov:
* Sistem je uključen, a korisnik ima kreiran i verifikovan nalog za pristup.

Osnovni scenario:

1. Korisnik bira opciju za prijavu na sistem.
2. Sistem prikazuje formu za prijavu koja traži korisničko ime i lozinku.
3. Korisnik unosi svoje korisničko ime i lozinku u predviđena polja.
4. Korisnik potvrđuje unos i šalje podatke sistemu klikom na dugme "Prijavi se".
5. Sistem proverava da li su uneti podaci validni i da li se podudaraju sa podacima u bazi.
6. Ukoliko su podaci ispravni, sistem autorizuje korisnika i daje mu pristup odgovarajućim funkcionalnostima na osnovu njegovog nivoa pristupa.
7. Sistem prikazuje poruku "Uspešno ste prijavljeni na sistem" i preusmerava korisnika na početnu stranu.

Alternativni scenariji: 5.1 Ukoliko korisničko ime i/ili lozinka nisu ispravni:

* Sistem prikazuje poruku o grešci: "Neispravno korisničko ime ili lozinka. Molimo pokušajte ponovo."

### Kreiranje obavestenja

Slučaj korišćenja: Kreiranje obaveštenja Akteri:

* Administrator sistema
* Sistem (program)

Preduslov: Administrator sistema je prijavljen na sistem i ima odgovarajuće privilegije za kreiranje obaveštenja.

Osnovni scenario:

1. Administrator sistema pristupa opciji za kreiranje obaveštenja.
2. Sistem prikazuje formu za unos detalja obaveštenja, uključujući naziv opštine, vrstu prekida (vode ili struje), datum i vreme početka i završetka prekida.
3. Administrator sistema unosi sve potrebne detalje u predviđena polja forme.
4. Administrator sistema potvrđuje unos i šalje podatke sistemu klikom na dugme "Kreiraj obaveštenje".
5. Sistem proverava validnost unetih podataka.
6. Ukoliko su svi podaci ispravni, sistem kreira novo obaveštenje sa unetim detaljima i čuva ih u bazi podataka.
7. Sistem prikazuje poruku "Obaveštenje uspešno kreirano" i vraća administratora sistema na prethodnu stranicu.

Alternativni scenariji:

5.1 Ukoliko su uneti podaci neispravni ili nedovoljno precizni:

* Sistem prikazuje odgovarajuću poruku o grešci i traži od administratora sistema da ispravi podatke pre nego što obaveštenje bude kreirano.

## Opis arihtekture aplikacije

Ova aplikacija je razvijena koriščenjem sledećih tehnologija: Laravel, React i Axios, omogućavajući brz i efikasan razvoj kompleksnih web aplikacija.

Laravel je odabran kao backend okvir zbog svoje visoke performanse, sigurnosti i lakoće korišćenja. Služi kao temelj naše aplikacije, upravljajući bazom podataka, autentifikacijom korisnika, logikom aplikacije i API endpointima. Laravelova arhitektura MVC (Model-View-Controller) omogućava jasnu separaciju logike i prezentacije, čime se olakšava održavanje koda i unapređenje aplikacije. API resursi su kreirani koristeći Laravel, što omogućava aplikaciji da komunicira sa frontendom putem JSON odgovora, pružajući efikasnu razmenu podataka.

Za izgradnju korisničkog interfejsa koristimo React, JavaScript biblioteku za izgradnju korisničkih interfejsa, koja nam omogućava da kreiramo dinamične i interaktivne web stranice. React se koristi za izgradnju komponenata koje definišu elemente i logiku korisničkog interfejsa, omogućavajući brzu i efikasnu izradu single-page aplikacija (SPA). Korišćenjem Reacta, aplikacija može da ažurira i prikazuje pravilne komponente na korisničkom interfejsu u realnom vremenu bez potrebe za ponovnim učitavanjem stranice, što znatno poboljšava korisničko iskustvo.

Za komunikaciju između React frontend-a i Laravel backend-a koristimo Axios, popularnu JavaScript biblioteku za izvršavanje HTTP zahteva. Axios nam omogućava da jednostavno upravljamo zahtevima i odgovorima između klijenta i servera, uključujući konfigurisanje zahteva, prenos tokena za autentifikaciju i obradu odgovora servera. Zahvaljujući Axiosu, naša aplikacija može efikasno da dohvati podatke, šalje podatke na server i implementira kompleksne funkcionalnosti poput CRUD operacija (Create, Read, Update, Delete) na siguran i pouzdan način.

## Opis procesa slučajeva korišćenja

### Dijagram sekvenci – registracija korisnika

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 1 ds registracija

### Dijagram sekvenci – prijava korisnika

Slika na kojoj se nalazi tekst, snimak ekrana, broj, Font

Opis je automatski generisan

Slika 2 ds - prijava korisnika

### Dijagram sekvenci – kreiranje obavestenja

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 3 ds -kreiranje obavestenja

## Model podataka

Model "Usluga" predstavlja entitet usluga sa atributima kao što su naziv, opis, cena, kategorija, vreme obrade i drugi. Ovaj model ima i relaciju sa modelom "Zahtev", omogućavajući povezivanje usluga sa zahtevima korisnika.

Model "User" je odgovoran za upravljanje korisničkim podacima. On sadrži informacije kao što su ime, prezime, email adresa, uloga i drugi lični podaci korisnika. Ovaj model omogućava autentifikaciju i autorizaciju korisnika putem sanctum paketa.

Model "Dokument" se koristi za čuvanje informacija o različitim vrstama dokumenata, kao što su lične karte, pasoši ili izvodi. Ovaj model povezan je sa korisničkim modelom kako bi se omogućila veza između korisnika i njihovih dokumenata.

Model "Zahtev" predstavlja zahtev korisnika za određenom uslugom. Sadrži informacije o statusu zahteva, vremenu podnošenja zahteva, prioritetu i dodatnim napomenama. Ovaj model takođe ima veze sa korisničkim modelom kako bi se pratila povezanost sa konkretnim korisnikom, kao i sa modelom "Usluga" kako bi se odredila usluga na koju se zahtev odnosi.

Model Obaveštenje služi za kreiranje obaveštenja o nestanku vode ili struje. Ovaj model nije povezan sa drugim modelima.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, dijagram

Opis je automatski generisan

Slika 4 dijagram klasa

## Specifikacija REST API-ja

U ovom poglavlju ce biti prikazani detalji o api rutama za kontrolere.

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih usluga |
| HTTP metoda | GET |
| URL | /api/usluge |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | [      {          "id": 1,          "naziv": "cupiditate maxime quis",          "opis": "Voluptatibus sit fugiat et reprehenderit tempora dolores et.",          "cena": "538.00",          "kategorija": "commodi",          "vreme\_obrade": 9,          "dokumentacija\_potrebna": "Qui ex corrupti minus alias vitae sit non.",          "digitalni\_potpis\_potreban": 1,          "prioritet": "low"      },      {          "id": 2,          "naziv": "excepturi iure facilis",          "opis": "Quo eum aliquam et aperiam et fugiat culpa.",          "cena": "52.00",          "kategorija": "unde",          "vreme\_obrade": 7,          "dokumentacija\_potrebna": "Recusandae ut tenetur autem aut qui voluptatem.",          "digitalni\_potpis\_potreban": 0,          "prioritet": "high"      },  …  **}** |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje usluge |
| HTTP metoda | POST |
| URL | /api/usluge |
| URL parametri | ?naziv=dadsa&opis=dsada&cena=111&kategorija=dsadas&vreme\_obrade=5&digitalni\_potpis\_potreban=0&prioritet=low |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 9,      "naziv": "dadsa",      "opis": "dsada",      "cena": "111",      "kategorija": "dsadas",      "vreme\_obrade": "5",      "dokumentacija\_potrebna": **null**,      "digitalni\_potpis\_potreban": "0",      "prioritet": "low"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje usluge |
| HTTP metoda | PUT |
| URL | /api/usluge |
| URL parametri | 9?naziv=USLUGA 1&opis=dsada&cena=111&kategorija=dsadas&vreme\_obrade=5&digitalni\_potpis\_potreban=0&prioritet=low |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 9,      "naziv": "USLUGA 1",      "opis": "dsada",      "cena": "111",      "kategorija": "dsadas",      "vreme\_obrade": "5",      "dokumentacija\_potrebna": **null**,      "digitalni\_potpis\_potreban": "0",      "prioritet": "low"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje usluge |
| HTTP metoda | DELETE |
| URL | /api/usluge |
| URL parametri | 12 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Usluga deleted successfully"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih zahteva |
| HTTP metoda | GET |
| URL | /api/zahtevi |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {          "id": 1,          "status": **null**,          "submitted\_at": "2024-01-01 04:44:06",          "processed\_at": **null**,          "request\_priority": "medium",          "additional\_notes": "Placeat hic eum consequatur illum ea ex.",          "processing\_deadline": "2024-01-05 17:54:10"      },      {          "id": 2,          "status": **null**,          "submitted\_at": "2024-01-07 12:06:29",          "processed\_at": "2024-01-06 06:26:15",          "request\_priority": "high",          "additional\_notes": "Quas blanditiis magnam non ab quod dolore.",          "processing\_deadline": "2024-01-06 11:39:25"      },  … |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje zahteva |
| HTTP metoda | POST |
| URL | /api/zahtevi |
| URL parametri | ?korisnik\_id=1&usluga\_id=1&status=sda&submitted\_at=2024-01-01 04:44:06&request\_priority=0 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "ime": "ImeKorisnika",      "prezime": "PrezimeKorisnika",      "imeOca": "ImeOcaKorisnika",      "email": "emaila@primer.com",      "password": "sifra123",      "uloga": "korisnik",      "jmbg": "1234167890123",      "brLicneKarte": "0121456789",      "datumRodjenja": "2000-01-01",      "mestoRodjenja": "GradRodjenja",      "br\_lk": "01113456789",      "adresaPrebivalista": "Adresa 123",      "opstinaPrebivalista": "Opstina",      "korisnik\_id": "1",      "usluga\_id": "1",      "status": "sda",      "submitted\_at": "2024-01-01 04:44:06",      "request\_priority": "0"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje zahteva |
| HTTP metoda | PUT |
| URL | /api/ zahtevi |
| URL parametri | /2/?korisnik\_id=1&usluga\_id=1&status\_zahteva=sda&submitted\_at=2024-01-01 04:44:06&request\_priority=0 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 2,      "status": **null**,      "submitted\_at": "2024-01-01 04:44:06",      "processed\_at": "2024-01-06 06:26:15",      "request\_priority": "0",      "additional\_notes": "Quas blanditiis magnam non ab quod dolore.",      "processing\_deadline": "2024-01-06 11:39:25"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje zahteva |
| HTTP metoda | DELETE |
| URL | /api/ zahtevi |
| URL parametri | 2 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Zahtev deleted successfully"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | nema |
| HTTP body parametri | {      "ime": "ImeKorisnika",      "prezime": "PrezimeKorisnika",      "imeOca": "ImeOcaKorisnika",      "email": "emaila@primer.com",      "password": "sifra123",      "uloga": "korisnik",      "jmbg": "1234167890123",      "brLicneKarte": "0121456789",      "datumRodjenja": "2000-01-01",      "mestoRodjenja": "GradRodjenja",      "br\_lk": "01113456789",      "adresaPrebivalista": "Adresa 123",      "opstinaPrebivalista": "Opstina"  } |
| Format HTTP body parametara | json |
| Izlazni parametri | {      "access\_token": "2|t9brjG2nA34DMf7v6xfYxmYAliNBRDZ00hrdAM2U81135eb8",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | ?email=emaila@primer.com&password=sifra123 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "access\_token": "3|k0Y3REv5XD3P045deOlCsS8uxWqRcMaEGCbX8qjsc1aa8c91",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Odjava korisnika |
| HTTP metoda | POST |
| URL | /api/logout |
| URL parametri | nema |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Logged out successfully"  } |
| Format izlaznih parametara | application/json |

# Opis tehnologija korišćenih u aplikaciji

U ovom poglavlju opisaćemo ključne tehnologije koje su korišćene u izgradnji naše aplikacije.

### React

React je popularna JavaScript biblioteka otvorenog koda koja se koristi za izgradnju korisničkih interfejsa, posebno za web aplikacije. Razvijen od strane Facebook-a, React je dizajniran za olakšavanje kreiranja visoko performantnih, skalabilnih i modularnih korisničkih interfejsa. Centralna ideja iza React-a je koncept komponenti, koje su samostalni delovi interfejsa koji se mogu ponovo koristiti i koji sadrže svoju logiku i prikaz. React koristi JSX (JavaScript XML) sintaksu za deklarativno opisivanje strukture korisničkog interfejsa u JavaScript-u, što olakšava razvoj i održavanje kompleksnih aplikacija. Osim toga, React se odlikuje efikasnim upravljanjem stanjem aplikacije, virtuelnim DOM-om za efikasno ažuriranje DOM-a, kao i bogatom ekosistemom dodataka i biblioteka koji proširuju funkcionalnosti React-a.

### Laravel

Laravel je popularni open-source PHP framework za razvoj web aplikacija. Nastao je 2011. godine i razvio se kao jedan od najpopularnijih alata u PHP zajednici. Laravel omogućava programerima da brzo razvijaju visoko kvalitetne web aplikacije koristeći elegantan i ekspresivan sintaksu. Nudi mnoge funkcije koje olakšavaju razvoj, uključujući rutiranje, upravljanje korisničkim sesijama, autentifikaciju, upravljanje bazama podataka putem Eloquent ORM, šabloniranje putem Blade motora, podršku za API-je, testiranje i još mnogo toga. Laravel takođe promoviše najbolje prakse razvoja, poput MVC (Model-View-Controller) arhitekture, što doprinosi čistijem i organizovanijem kodu.

### Axios

Axios je obećavajuća JavaScript biblioteka koja se koristi za izvršavanje HTTP zahteva iz browsera i Node.js aplikacija. Izabrali smo Axios zbog njegove jednostavnosti upotrebe, podrške za Promise API, i mogućnosti da se lako konfiguriše za potrebe naše aplikacije. Axios omogućava slanje asinhronih HTTP zahteva ka RESTful API endpointima, podržavajući globalne konfiguracije, presretanje zahteva i odgovora, te automatsko transformisanje JSON podataka. Korišćenje Axiosa u našoj aplikaciji olakšava komunikaciju između frontend-a i backend-a, čineći razmenu podataka brzom i efikasnom.

# Korisničko uputstvo

Početna stranica na kojoj korisnici mogu da vide osnovni set usluga koji se nudi, kao i razne slike građana.

Slika na kojoj se nalazi tekst, nebo, drvo, snimak ekrana

Opis je automatski generisan

Slika 5 početn

Korisnici koji nisu ulogovani imaju mogućnost pregleda COVID statistika, kao I obaveštenja o planiranim isključenjima vode ili struje.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 6 covid

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 7 obaveštenja

Korisnici imaju mogućnost prijave na system, kao I registracije. Ovo je prikazano na slikama ispod.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, dizajn

Opis je automatski generisan

Slika 8 login

Slika na kojoj se nalazi tekst, snimak ekrana, softver, broj

Opis je automatski generisan

Slika 9 register

Nakon prijave, korisnicima se otvara forma preko koje mogu da kreiraju zahtev za neki od servisa euprave.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Font

Opis je automatski generisan

Slika 10 kreiraj zahtev

Kada se uloguje Admin, prikazuje mu se stranica Usluge na kojoj korisnici mogu vrsiti filtritranje, brisanje ili izmenu usluga. Ovde je implemnetirana i paginacija.

Slika na kojoj se nalazi tekst, snimak ekrana, broj, Font

Opis je automatski generisan

Slika 11 stranica usluge

Klikom na dugme promeni otvara se forma za izmenu određene usluge

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Ikona na računaru

Opis je automatski generisan

Slika 12 izmena

Odabirom opcije Dodaj iz navigacionog menija, korisnicima se otvara forma za dodavanje novih usluge

Slika na kojoj se nalazi tekst, snimak ekrana, softver, broj

Opis je automatski generisan

Slika 13 kreiranje

Odabirom opcije KreirajObavestenje iz navigacionog menija, administrator može da kreira obaveštenje o nestanku vode ili struje.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Font

Opis je automatski generisan

Slika 14 kreiranje obaveštenja

Admin takođe ima I mogućnost pregleda statistika, kako bi znao koje usluge su traženije od strane građana.

Slika na kojoj se nalazi tekst, snimak ekrana, dijagram, Font

Opis je automatski generisan

Slika 15 statistike

# Prikaz delova koda

## Laravel

Kontroler "UslugaController" upravlja akcijama povezanim sa uslugama, uključujući prikazivanje svih usluga, kreiranje nove usluge, ažuriranje postojeće usluge i brisanje usluge. Koristi se za manipulaciju podacima o uslugama, proveru i validaciju unesenih podataka te vraćanje odgovarajućih odgovora.

<?php

namespace App\Http\Controllers;

use App\Http\Resources\UslugaResource;

use App\Models\Usluga;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

class UslugaController extends Controller

{

    public function index()

    {

        $usluge = Usluga::all();

        return response()->json(UslugaResource::collection($usluge));

    }

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'naziv' => 'required|string|max:255',

            'opis' => 'required|string|max:1000',

            'cena' => 'required|numeric|min:0',

            'kategorija' => 'required|string|max:255',

            'vreme\_obrade' => 'required|integer|min:0',

            'dokumentacija\_potrebna' => 'sometimes|string|max:1000',

            'digitalni\_potpis\_potreban' => 'required|boolean',

            'prioritet' => 'required|string|in:low,medium,high',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $usluga = Usluga::create($validator->validated());

        return response()->json(new UslugaResource($usluga), 201);

    }

    public function update(Request $request, $id)

    {

        $usluga = Usluga::find($id);

        if (!$usluga) {

            return response()->json(['message' => 'Usluga not found'], 404);

        }

        $validator = Validator::make($request->all(), [

            'naziv' => 'required|string|max:255',

            'opis' => 'required|string|max:1000',

            'cena' => 'required|numeric|min:0',

            'kategorija' => 'required|string|max:255',

            'vreme\_obrade' => 'required|integer|min:0',

            'dokumentacija\_potrebna' => 'sometimes|string|max:1000',

            'digitalni\_potpis\_potreban' => 'required|boolean',

            'prioritet' => 'required|string|in:low,medium,high',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $usluga->update($validator->validated());

        return response()->json(new UslugaResource($usluga));

    }

    public function destroy($id)

    {

        $usluga = Usluga::find($id);

        if (!$usluga) {

            return response()->json(['message' => 'Usluga not found'], 404);

        }

        $usluga->delete();

        return response()->json(['message' => 'Usluga deleted successfully']);

    }

}

Kontroler "ZahtevController" se bavi akcijama povezanim sa zahtevima korisnika za određenim uslugama. Ovo uključuje prikazivanje svih zahteva, kreiranje novih zahteva, ažuriranje postojećih zahteva i brisanje zahteva. Kontroler takođe obavlja validaciju unesenih podataka, proveru postojanja povezanih resursa i vraćanje odgovarajućih odgovora.

<?php

namespace App\Http\Controllers;

use App\Http\Resources\ZahtevResource;

use App\Models\Zahtev;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

class ZahtevController extends Controller

{

    public function index()

    {

        $zahtevi = Zahtev::all();

        return response()->json(ZahtevResource::collection($zahtevi));

    }

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'korisnik\_id' => 'required|exists:users,id',

            'usluga\_id' => 'required|exists:usluge,id',

            'status' => 'required|string|max:255',

            'submitted\_at' => 'required|date',

            'processed\_at' => 'nullable|date',

            'request\_priority' => 'required|string|max:255',

            'additional\_notes' => 'nullable|string|max:1000',

            'processing\_deadline' => 'nullable|date',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $zahtev = Zahtev::create($validator->validated());

        return response()->json(new ZahtevResource($zahtev), 201);

    }

    public function update(Request $request, $id)

    {

        $zahtev = Zahtev::find($id);

        if (!$zahtev) {

            return response()->json(['message' => 'Zahtev not found'], 404);

        }

        $validator = Validator::make($request->all(), [

            'korisnik\_id' => 'required|exists:users,id',

            'usluga\_id' => 'required|exists:usluge,id',

            'status' => 'required|string|max:255',

            'submitted\_at' => 'required|date',

            'processed\_at' => 'nullable|date',

            'request\_priority' => 'required|string|max:255',

            'additional\_notes' => 'nullable|string|max:1000',

            'processing\_deadline' => 'nullable|date',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 400);

        }

        $zahtev->update($validator->validated());

        return response()->json(new ZahtevResource($zahtev));

    }

    public function destroy($id)

    {

        $zahtev = Zahtev::find($id);

        if (!$zahtev) {

            return response()->json(['message' => 'Zahtev not found'], 404);

        }

        $zahtev->delete();

        return response()->json(['message' => 'Zahtev deleted successfully']);

    }

}

U oba kontrolera su implementirane sledece metode

* "index": Služi za prikazivanje liste resursa.
* "store": Služi za kreiranje novog resursa.
* "update": Služi za ažuriranje postojećeg resursa.
* "destroy": Služi za brisanje postojećeg resursa.
* "show": Služi za prikazivanje detalja određenog resursa.

Kontroler "AuthController" odgovoran je za autentifikaciju i registraciju korisnika u aplikaciji. Metoda "register" omogućava registraciju novih korisnika sa validacijom unesenih podataka, a nakon uspešne registracije generiše se i šalje JWT token za autentifikaciju. Metoda "login" omogućava prijavu postojećih korisnika, takođe uz upotrebu JWT tokena. Kontroler takođe ima metodu "logout" za odjavu korisnika, "searchUsers" za pretragu korisnika prema različitim kriterijumima i "exportUsersToCsv" za izvoz podataka o korisnicima u CSV formatu. Takođe, pruža i mogućnost dobijanja informacija o trenutno prijavljenom korisniku putem metode "getCurrentUser".

<?php

namespace App\Http\Controllers;

use App\Http\Resources\UserResource;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

class AuthController extends Controller

{

    public function register(Request $request)

{

    $validator = Validator::make($request->all(), [

        'ime' => 'required|string|max:255',

        'prezime' => 'required|string|max:255',

        'imeOca' => 'nullable|string|max:255',

        'email' => 'required|string|email|max:255|unique:users',

        'password' => 'required|string|min:6',

        'jmbg' => 'required|string|max:13|unique:users',

        'brLicneKarte' => 'required|string|max:20|unique:users',

        'datumRodjenja' => 'required|date',

        'mestoRodjenja' => 'required|string|max:255',

        'adresaPrebivalista' => 'required|string|max:255',

        'opstinaPrebivalista' => 'required|string|max:255'

    ]);

    if ($validator->fails()) {

        return response()->json($validator->errors(), 400);

    }

    $user = User::create([

        'ime' => $request->ime,

        'prezime' => $request->prezime,

        'imeOca' => $request->imeOca,

        'email' => $request->email,

        'password' => Hash::make($request->password),

        'jmbg' => $request->jmbg,

        'brLicneKarte' => $request->brLicneKarte,

        'datumRodjenja' => $request->datumRodjenja,

        'mestoRodjenja' => $request->mestoRodjenja,

        'adresaPrebivalista' => $request->adresaPrebivalista,

        'opstinaPrebivalista' => $request->opstinaPrebivalista

    ]);

    $token = $user->createToken('auth\_token')->plainTextToken;

    return response()->json(['access\_token' => $token, 'token\_type' => 'Bearer']);

}

    public function login(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'email' => 'required|email',

            'password' => 'required',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 401);

        }

        if (!Auth::attempt($request->only('email', 'password'))) {

            return response()->json(['message' => 'Unauthorized'], 401);

        }

        $user = User::where('email', $request['email'])->firstOrFail();

        $token = $user->createToken('auth\_token')->plainTextToken;

        return response()->json(['access\_token' => $token, 'token\_type' => 'Bearer']);

    }

    public function logout(Request $request)

    {

        $request->user()->currentAccessToken()->delete();

        return response()->json(['message' => 'Logged out successfully']);

    }

    public function searchUsers(Request $request)

    {

        $query = User::query();

        if ($request->has('opstinaPrebivalista')) {

            $query->where('opstinaPrebivalista', $request->opstinaPrebivalista);

        }

        if ($request->has('mestoRodjenja')) {

            $query->where('mestoRodjenja', $request->mestoRodjenja);

        }

        if ($request->has('datumRodjenjaStart') && $request->has('datumRodjenjaEnd')) {

            $query->whereBetween('datumRodjenja', [$request->datumRodjenjaStart, $request->datumRodjenjaEnd]);

        } elseif ($request->has('datumRodjenjaStart')) {

            $query->where('datumRodjenja', '>=', $request->datumRodjenjaStart);

        } elseif ($request->has('datumRodjenjaEnd')) {

            $query->where('datumRodjenja', '<=', $request->datumRodjenjaEnd);

        }

        $users = $query->get();

        return response()->json($users);

    }

    public function exportUsersToCsv()

    {

        $fileName = 'users\_' . date('Y-m-d\_His') . '.csv';

        $filePath = storage\_path('app/public/' . $fileName);

        $users = User::all();

        $columns = array\_keys($users->first()->toArray());

        $file = fopen($filePath, 'w');

        fputcsv($file, $columns);

        foreach ($users as $user) {

            fputcsv($file, $user->toArray());

        }

        fclose($file);

        return response()->json(['message' => 'File saved successfully', 'file' => $fileName]);

    }

    public function getCurrentUser(Request $request)

    {

        $user = $request->user();

        return new UserResource($user);

    }

}

## React

U ovoj React aplikaciji, implementirani su koncepti rutiranja koristeći react-router-dom biblioteku, koja omogućava navigaciju između različitih komponenata bez potrebe za ponovnim učitavanjem cele stranice. Glavna komponenta App koristi BrowserRouter za obuhvatanje celokupne aplikacije, omogućavajući rutiranje na klijentskoj strani. Unutar BrowserRouter, komponenta Navbar pruža navigacione linkove, koristeći Link komponentu iz react-router-dom za kreiranje veza ka različitim stranicama unutar aplikacije, poput početne stranice, stranice za dodavanje usluga, pregleda usluga i stranice za ažuriranje usluga. Routes komponenta definiše različite rute (Route), svaka sa svojom putanjom (path) i elementom koji treba renderovati, povezujući URL putanje sa odgovarajućim React komponentama kao što su PocetnaStranica, Usluge, NovaUslugaForma i AzurirajUslugu. Ovaj pristup rutiranju omogućava efikasnu organizaciju i navigaciju unutar aplikacije, uz lako upravljanje sadržajem koji se prikazuje korisniku u zavisnosti od trenutne URL putanje.

import React from 'react';

import { BrowserRouter, Routes, Route } from 'react-router-dom';

import PocetnaStranica from './components/home/Pocetna';

import Usluge from './components/usluge/Usluge';

import Navbar from './components/navbar/Navbar';

import NovaUslugaForma from './components/usluge/NovaUslugaForma';

import AzurirajUslugu from './components/usluge/AzurirajUslugu';

function App() {

  return (

    <BrowserRouter>

    <Navbar></Navbar>

      <Routes>

        <Route path="/" element={<PocetnaStranica />} />

        <Route path="usluge/promeni/:id" element={<AzurirajUslugu />} />

        <Route path="/usluge/dodaj" element={<NovaUslugaForma />} />

        <Route path="/usluge" element={<Usluge />} />

      </Routes>

    </BrowserRouter>

  );

}

export default App;

import React from 'react';

import { Link } from 'react-router-dom';

import './Navbar.css';

function Navbar() {

  return (

    <nav className="navbar">

      <div className="nav-container">

        <div className="nav-links">

          <Link to="/" className="nav-item">Početna</Link>

          <Link to="/usluge/dodaj" className="nav-item">Dodaj</Link>

          <Link to="/usluge" className="nav-item">Usluge</Link>

        </div>

      </div>

    </nav>

  );

}

export default Navbar;

Komponenta PocetnaStranica u React aplikaciji predstavlja početnu stranicu sa ključnim informacijama i uslugama. Korišćenjem React Hooks (useState i useEffect), komponenta upravlja prikazom slika i sadržaja. U useState se čuvaju slike koje se dinamički učitavaju iz Unsplash API-ja. useEffect se koristi za postavljanje ovih slika prilikom inicijalnog učitavanja komponente. Komponenta takođe uključuje Slider iz react-slick biblioteke za prikaz slika u obliku karusela, sa definisanim postavkama kao što su automatsko prelistavanje, brzina prelaza i sl. U okviru stranice, prikazane su različite kategorije usluga kao što su porodični život, vrtići, obrazovanje i zapošljavanje, svaka sa svojim opisom i identifikatorom.

import React, { useState, useEffect } from 'react';

import Slider from 'react-slick';

import './Pocetna.css';

import 'slick-carousel/slick/slick.css';

import 'slick-carousel/slick/slick-theme.css';

function PocetnaStranica() {

  const [images, setImages] = useState([]);

  useEffect(() => {

    // Here you would fetch images from the Unsplash API

    // For demonstration, we're using static image URLs.

    setImages([

      'https://source.unsplash.com/featured/?school',

      'https://source.unsplash.com/featured/?kindergarten',

      'https://source.unsplash.com/featured/?job',

      'https://source.unsplash.com/featured/?education',

      'https://source.unsplash.com/featured/?family'

    ]);

  }, []);

  const settings = {

    dots: true,

    infinite: true,

    speed: 500,

    slidesToShow: 1,

    slidesToScroll: 1,

    autoplay: true,

    autoplaySpeed: 3000

  };

  return (

    <div className="pocetna">

      <header className="pocetna-header">

        <h1>Dobro došli</h1>

        <p>U par koraka do svih informacija i elektronskih usluga u Republici Srbiji.</p>

      </header>

      <section className="pocetna-services">

        <div className="service-card" id="family">

          <h2>Porodični život</h2>

          <p>Trudničko i porodiljsko bolovanje, naknade, prijava deteta, lečenje lopatak...</p>

        </div>

        <div className="service-card" id="birth">

          <h2>Vrtic</h2>

          <p>Upis u vrtic, eVrtic...</p>

        </div>

        <div className="service-card" id="education">

          <h2>Osnovno obrazovanje</h2>

          <p>Proceduri dovarak, dokumentacija za upis deteta u osnovnu školu, elektronski dnevnik...</p>

        </div>

        <div className="service-card" id="employment">

          <h2>Tražim posao</h2>

          <p>Najlakši načini do posla.</p>

        </div>

      </section>

      <Slider {...settings}>

          {images.map((img, index) => (

            <div key={index}>

              <img src={img} alt="Slika" />

            </div>

          ))}

        </Slider>

    </div>

  );

}

export default PocetnaStranica;

U komponenti Usluge, implementirana je funkcionalnost za prikaz, pretragu, brisanje i ažuriranje usluga koristeći React i axios za upravljanje HTTP zahtevima. Pomoću kuke useFetchServices, podaci o uslugama se dinamički učitavaju sa zadatog URL-a. Ova kuka upravlja stanjima kao što su loading i error za efikasno upravljanje procesom učitavanja i grešaka. Nakon što se podaci učitaju, korisnici mogu pretraživati usluge po različitim kriterijumima kao što su naziv, kategorija i prioritet. Komponenta takođe omogućava paginaciju rezultata, prikazujući ograničen broj usluga po stranici za bolju preglednost.

RedTabele je odgovorna za prikaz pojedinačnih redova u tabeli usluga. Ova komponenta omogućava brisanje usluga putem axios DELETE zahteva i navigaciju do stranice za ažuriranje usluga, gde korisnici mogu izmeniti postojeće informacije. Svaki red u tabeli sadrži informacije o usluzi kao što su naziv, opis, cena, kategorija, i druge relevantne informacije, uz opcije za brisanje i ažuriranje.

import { useState, useEffect } from 'react';

import axios from 'axios';

const useFetchServices = (url) => {

  const [data, setData] = useState([]);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    setLoading(true);

    axios.get(url)

      .then(response => {

        setData(response.data);

        setLoading(false);

      })

      .catch(error => {

        setError(error);

        setLoading(false);

      });

  }, [url]);

  return { data, setData,loading, error };

};

export default useFetchServices;

import React from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

function RedTabele({ usluga, setUsluge }) {

  const navigate = useNavigate();

  const handleDelete = (id) => {

    if (window.confirm("Da li ste sigurni da želite da obrišete uslugu?")) {

      axios.delete(`http://127.0.0.1:8000/api/usluge/${id}`)

        .then(response => {

          alert('Usluga je uspešno obrisana.');

          setUsluge((prevUsluge) => prevUsluge.filter((usluga) => usluga.id !== id));

        })

        .catch(error => {

          console.error('There was an error deleting the usluga!', error);

        });

    }

  };

  const goToUpdatePage = (id) => {

    navigate(`promeni/${id}`);

  };

  return (

    <tr>

      <td>{usluga.naziv}</td>

      <td>{usluga.opis}</td>

      <td>{usluga.cena}</td>

      <td>{usluga.kategorija}</td>

      <td>{usluga.vreme\_obrade}</td>

      <td>{usluga.dokumentacija\_potrebna}</td>

      <td>{usluga.digitalni\_potpis\_potreban ? 'Da' : 'Ne'}</td>

      <td>{usluga.prioritet}</td>

      <td>

        <button onClick={() => goToUpdatePage(usluga.id)}>Promeni</button>

        <button onClick={() => handleDelete(usluga.id)}>Obriši</button>

      </td>

    </tr>

  );

}

export default RedTabele;

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import './Usluge.css';

import RedTabele from './RedTabele';

import useFetchServices from './useFetchServices ';

function Usluge() {

  const { data: usluge,setData:setUsluge, loading, error } = useFetchServices('http://127.0.0.1:8000/api/usluge');

  const [filteredUsluge, setFilteredUsluge] = useState([]);

  const [search, setSearch] = useState({

    naziv: '',

    kategorija: '',

    prioritet: '',

  });

  const [currentPage, setCurrentPage] = useState(1);

  const servicesPerPage = 5;

  useEffect(() => {

    setFilteredUsluge(usluge);

  }, [usluge]);

  useEffect(() => {

    const filteredData = usluge.filter(usluga =>

      (!search.naziv || usluga.naziv.toLowerCase().includes(search.naziv.toLowerCase())) &&

      (!search.kategorija || usluga.kategorija.toLowerCase().includes(search.kategorija.toLowerCase())) &&

      (!search.prioritet || usluga.prioritet.toLowerCase().includes(search.prioritet.toLowerCase()))

    );

    setFilteredUsluge(filteredData);

  }, [search, usluge]);

  const handleSearchChange = (e) => {

    const { name, value } = e.target;

    setSearch({ ...search, [name]: value });

    setCurrentPage(1); // Reset to first page when search changes

  };

  const handleSearch = (e) => {

    e.preventDefault();

  };

  // Get current services

  const indexOfLastService = currentPage \* servicesPerPage;

  const indexOfFirstService = indexOfLastService - servicesPerPage;

  const currentServices = filteredUsluge.slice(indexOfFirstService, indexOfLastService);

  const totalPages = Math.ceil(filteredUsluge.length / servicesPerPage);

  // Change page

  const paginate = (pageNumber) => setCurrentPage(pageNumber);

  if (loading) return <p>Loading...</p>;

  if (error) return <p>Error loading services: {error.message}</p>;

  return (

    <div className="usluge">

    <form onSubmit={handleSearch}>

      <input

          type="text"

          name="naziv"

          value={search.naziv}

          onChange={handleSearchChange}

          placeholder="Pretraži po nazivu"

        />

        <input

          type="text"

          name="kategorija"

          value={search.kategorija}

          onChange={handleSearchChange}

          placeholder="Pretraži po kategoriji"

        />

        <select name="prioritet" value={search.prioritet} onChange={handleSearchChange}>

          <option value="">Svi prioriteti</option>

          <option value="low">Nizak</option>

          <option value="medium">Srednji</option>

          <option value="high">Visok</option>

        </select>

      <button type="submit">Pretraži</button></form>

      <table>

        <thead>

          <tr>

            <th>Naziv</th>

            <th>Opis</th>

            <th>Cena</th>

            <th>Kategorija</th>

            <th>Vreme Obrade</th>

            <th>Dokumentacija Potrebna</th>

            <th>Digitalni Potpis Potreban</th>

            <th>Prioritet</th>

            <th>Obrisi</th>

          </tr>

        </thead>

        <tbody>

          {currentServices.map(usluga => (

            <RedTabele key={usluga.id} usluga={usluga} setUsluge={setUsluge} />

          ))}

        </tbody>

      </table>

      <div className="pagination">

        {Array.from({ length: totalPages }, (\_, index) => (

          <button

            key={index}

            onClick={() => paginate(index + 1)}

            className={currentPage === index + 1 ? 'active' : ''}

          >

            {index + 1}

          </button>

        ))}

      </div>

    </div>

  );

}

export default Usluge;

Komponenta NovaUslugaForma u React aplikaciji služi za dodavanje novih usluga kroz formu koja komunicira sa back-end serverom putem axios HTTP biblioteke. Korisnik može uneti podatke o usluzi, uključujući naziv, opis, cenu, kategoriju, vreme obrade, potrebnu dokumentaciju, informaciju da li je digitalni potpis potreban, kao i prioritet usluge. Kada se forma popuni i potvrdi, podaci se šalju back-endu putem POST zahteva na definisani URL. Stanje forme se upravlja korišćenjem React hook-a useState, a promene u input poljima se obrađuju putem handleInputChange funkcije, koja ažurira stanje forme. Po uspešnom dodavanju usluge, korisnik dobija obaveštenje, a forma se resetuje za eventualno dodavanje sledeće usluge. U slučaju greške pri slanju podataka, korisnik će biti obavešten o grešci.

import React, { useState } from 'react';

import axios from 'axios';

import './NovaUslugaForma.css';

function NovaUslugaForma() {

  const [formData, setFormData] = useState({

    naziv: '',

    opis: '',

    cena: '',

    kategorija: '',

    vreme\_obrade: '',

    dokumentacija\_potrebna: '',

    digitalni\_potpis\_potreban: false,

    prioritet: ''

  });

  const handleInputChange = (e) => {

    const { name, value, type, checked } = e.target;

    setFormData({

      ...formData,

      [name]: type === 'checkbox' ? checked : value

    });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    axios.post('http://127.0.0.1:8000/api/usluge', formData)

      .then(response => {

        alert('Usluga je uspešno dodata!');

        // Optionally reset form or handle response further

        setFormData({

          naziv: '',

          opis: '',

          cena: '',

          kategorija: '',

          vreme\_obrade: '',

          dokumentacija\_potrebna: '',

          digitalni\_potpis\_potreban: false,

          prioritet: ''

        });

      })

      .catch(error => {

        console.error('There was an error submitting the form!', error);

      });

  };

  return (

    <form className="nova-usluga-forma" onSubmit={handleSubmit}>

      <input

        type="text"

        name="naziv"

        value={formData.naziv}

        onChange={handleInputChange}

        placeholder="Naziv usluge"

        required

      />

      <textarea

        name="opis"

        value={formData.opis}

        onChange={handleInputChange}

        placeholder="Opis usluge"

        required

      />

      <input

        type="number"

        name="cena"

        value={formData.cena}

        onChange={handleInputChange}

        placeholder="Cena usluge"

        required

      />

      <input

        type="text"

        name="kategorija"

        value={formData.kategorija}

        onChange={handleInputChange}

        placeholder="Kategorija usluge"

        required

      />

      <input

        type="number"

        name="vreme\_obrade"

        value={formData.vreme\_obrade}

        onChange={handleInputChange}

        placeholder="Vreme obrade (u danima)"

        required

      />

      <input

        type="text"

        name="dokumentacija\_potrebna"

        value={formData.dokumentacija\_potrebna}

        onChange={handleInputChange}

        placeholder="Dokumentacija potrebna"

      />

      <label>

        Digitalni potpis potreban:

        <input

          type="checkbox"

          name="digitalni\_potpis\_potreban"

          checked={formData.digitalni\_potpis\_potreban}

          onChange={handleInputChange}

        />

      </label>

      <select name="prioritet" value={formData.prioritet} onChange={handleInputChange} required>

        <option value="">Izaberite prioritet</option>

        <option value="low">Nizak</option>

        <option value="medium">Srednji</option>

        <option value="high">Visok</option>

      </select>

      <button type="submit">Dodaj Uslugu</button>

    </form>

  );

}

export default NovaUslugaForma;

Komponenta AzurirajUslugu u React aplikaciji omogućava ažuriranje podataka o postojećoj usluzi. Nakon što korisnik odabere uslugu za ažuriranje, komponenta koristi ID usluge iz URL-a (dobijenog preko useParams kuke) za učitavanje detalja te usluge pomoću axios GET zahteva. Inicijalno stanje forme, koje upravlja podacima usluge, postavlja se na osnovu odgovora dobijenog sa servera. Forma koristi InputField komponente za unos podataka kao što su naziv, opis, cena, kategorija, vreme obrade i dokumentacija potrebna. Takođe postoji checkbox za digitalni potpis i selekt meni za odabir prioriteta. Promene u input poljima se obrađuju kroz handleInputChange funkciju koja ažurira stanje forme. Kada se forma pošalje, izvršava se axios PUT zahtev na server sa ažuriranim podacima usluge. U slučaju uspeha, korisnik dobija obaveštenje o uspešnom ažuriranju, dok se u slučaju greške prikazuje odgovarajuća poruka o grešci.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { useParams } from 'react-router-dom';

import './NovaUslugaForma.css';

import InputField from './InputField';

function AzurirajUslugu() {

  const [formData, setFormData] = useState({

    naziv: '',

    opis: '',

    cena: '',

    kategorija: '',

    vreme\_obrade: '',

    dokumentacija\_potrebna: '',

    digitalni\_potpis\_potreban: false,

    prioritet: ''

  });

  const { id } = useParams();

  useEffect(() => {

    axios.get(`http://127.0.0.1:8000/api/usluge/${id}`)

      .then(response => {

        setFormData(response.data);

      })

      .catch(error => {

        console.error('There was an error fetching the service details!', error);

      });

  }, [id]);

  const handleInputChange = (e) => {

    const { name, value, type, checked } = e.target;

    setFormData({

      ...formData,

      [name]: type === 'checkbox' ? checked : value

    });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    axios.put(`http://127.0.0.1:8000/api/usluge/${id}`, formData)

      .then(response => {

        alert('Usluga je uspešno ažurirana!');

      })

      .catch(error => {

        console.error('There was an error updating the service!', error);

      });

  };

  return (

    <form className="nova-usluga-forma" onSubmit={handleSubmit}>

      <label htmlFor="naziv">Naziv usluge</label>

      <InputField

        label="Naziv usluge"

        type="text"

        name="naziv"

        value={formData.naziv}

        onChange={handleInputChange}

        required

      />

      <InputField

        label="Opis usluge"

        type="text"

        name="opis"

        value={formData.opis}

        onChange={handleInputChange}

        required

      />

      <InputField

        label="Cena usluge"

        type="number"

        name="cena"

        value={formData.cena}

        onChange={handleInputChange}

        required

      />

      <InputField

        label="Kategorija usluge"

        type="text"

        name="kategorija"

        value={formData.kategorija}

        onChange={handleInputChange}

        required

      />

      <InputField

        label="Vreme obrade (u danima)"

        type="number"

        name="vreme\_obrade"

        value={formData.vreme\_obrade}

        onChange={handleInputChange}

        required

      />

      <InputField

        label="Dokumentacija potrebna"

        type="text"

        name="dokumentacija\_potrebna"

        value={formData.dokumentacija\_potrebna}

        onChange={handleInputChange}

      />

      <label htmlFor="digitalni\_potpis\_potreban">

        Digitalni potpis potreban:

        <input

          id="digitalni\_potpis\_potreban"

          type="checkbox"

          name="digitalni\_potpis\_potreban"

          checked={formData.digitalni\_potpis\_potreban}

          onChange={handleInputChange}

        />

      </label>

      <label htmlFor="prioritet">Prioritet</label>

      <select

        id="prioritet"

        name="prioritet"

        value={formData.prioritet}

        onChange={handleInputChange}

        required

      >

        <option value="">Izaberite prioritet</option>

        <option value="low">Nizak</option>

        <option value="medium">Srednji</option>

        <option value="high">Visok</option>

      </select>

      <button type="submit">Ažuriraj Uslugu</button>

    </form>

  );

}

export default AzurirajUslugu;

Komponentu KreirajObavestenje, koja omogućava administratoru sistema da unese novo obaveštenje u sistem. Unutar komponente se koristi Hook useState kako bi se inicijalizovalo stanje forme. formData je objekat koji sadrži četiri ključa: opstina, vrsta\_prekida, datum\_pocetka i datum\_zavrsetka. Svaki od ovih ključeva čuva vrednost polja forme koje korisnik unosi.

Funkcija handleChange se koristi za ažuriranje stanja forme kada korisnik unese ili izmeni vrednost polja forme. Koristi se pristup prethodnom stanju kako bi se očuvale prethodne vrednosti i zatim se ažurira samo polje koje je promenjeno.

Funkcija handleSubmit se poziva kada korisnik klikne na dugme "Pošalji" u formi. Ova funkcija obavlja HTTP POST zahtev ka API endpointu (http://127.0.0.1:8000/api/obavestenja) kako bi poslala unete podatke na server. Prilikom slanja zahteva, u zaglavlju se uključuje autorizacioni token iz lokalnog skladišta kako bi se osiguralo da samo autorizovani korisnici mogu da pristupe ovoj funkcionalnosti. Ukoliko se zahtev uspešno izvrši, korisnik se obaveštava alert-om da je obaveštenje uspešno dodato, a zatim se preusmerava na stranicu za administraciju (/admin). U suprotnom, ako se pojavi greška prilikom slanja podataka, greška se prikazuje u konzoli.

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

const KreirajObavestenje = () => {

    let navigate=useNavigate();

  const [formData, setFormData] = useState({

    opstina: '',

    vrsta\_prekida: '',

    datum\_pocetka: '',

    datum\_zavrsetka: ''

  });

  const handleChange = (e) => {

    setFormData({ ...formData, [e.target.name]: e.target.value });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const token = localStorage.getItem('accessToken');

      const headers = {

        'Authorization': `Bearer ${token}`,

      };

      await axios.post('http://127.0.0.1:8000/api/obavestenja', formData, { headers });

       alert("Uspesno dodato");

       navigate('/admin');

    } catch (error) {

      console.error('Greška prilikom slanja podataka:', error);

    }

  };

  return (

    <div className="register-container">

      <form className="register-form" onSubmit={handleSubmit}>

        <h2>Unesite novo obaveštenje</h2>

        <input

          type="text"

          name="opstina"

          placeholder="Opština"

          value={formData.opstina}

          onChange={handleChange}

          required

        />

        <input

          type="text"

          name="vrsta\_prekida"

          placeholder="Vrsta prekida (voda/struja)"

          value={formData.vrsta\_prekida}

          onChange={handleChange}

          required

        />

        <input

          type="datetime-local"

          name="datum\_pocetka"

          placeholder="Datum početka"

          value={formData.datum\_pocetka}

          onChange={handleChange}

          required

        />

        <input

          type="datetime-local"

          name="datum\_zavrsetka"

          placeholder="Datum završetka"

          value={formData.datum\_zavrsetka}

          onChange={handleChange}

          required

        />

        <button type="submit">Pošalji</button>

      </form>

    </div>

  );

};

export default KreirajObavestenje;

Komponenta Login, služi za upravljanje sesijom na klijentskoj strani.

Metoda handleInputChange ažurira stanje formData kada korisnik unese ili izmeni vrednosti u poljima za unos. Kada se forma pošalje (submit), metoda handleSubmit se poziva. Ova metoda prvo sprečava podrazumevano ponašanje forme (ponovno učitavanje stranice) pozivom e.preventDefault(). Zatim šalje POST zahtev na API endpoint /api/login sa unetim podacima za prijavu (formData). Ako su uneti podaci validni i prijava uspe, server vraća JWT token koji se zatim čuva u lokalnom skladištu (localStorage) pod ključem accessToken. Nakon toga se korisnik preusmerava na odgovarajuću stranicu u zavisnosti od njegove uloge (korisnik ili administrator) pomoću navigate funkcije. Takođe se postavlja token i uloga korisnika koristeći propse setToken i setUloga.

import React, { useState } from 'react';

import axios from 'axios';

import './Login.css';

import { useNavigate } from 'react-router-dom';

function Login({setToken,setUloga}) {

    let navigate=useNavigate();

  const [formData, setFormData] = useState({

    email: 'qcole@example.net',

    password: 'secret'

  });

  const [error, setError] = useState('');

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setFormData({

      ...formData,

      [name]: value

    });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post('http://127.0.0.1:8000/api/login', formData);

      const accessToken = response.data.access\_token;

      localStorage.setItem('accessToken', accessToken);

     if(response.data.user.role=='admin'){

        navigate('/usluge');

     }else{

        navigate('/kreirajzahtev');

     }

      setToken(accessToken)

      setUloga(response.data.user.role);

    } catch (error) {

      setError('Invalid email or password. Please try again.');

      console.error('Error logging in:', error);

    }

  };

  return (

    <div className="login-container">

      <form className="login-form" onSubmit={handleSubmit}>

        <h2>Login</h2>

        {error && <p className="error-message">{error}</p>}

        <input

          type="email"

          name="email"

          value={formData.email}

          onChange={handleInputChange}

          placeholder="Email"

          required

        />

        <input

          type="password"

          name="password"

          value={formData.password}

          onChange={handleInputChange}

          placeholder="Password"

          required

        />

        <button type="submit">Login</button>

      </form>

    </div>

  );

}

export default Login;

Slično kao I Login, postoji I komponenta za registraciju

import React, { useState } from 'react';

import axios from 'axios';

import './Register.css';

import { useNavigate } from 'react-router-dom';

function Register() {

    let navigate=useNavigate();

  const [formData, setFormData] = useState({

    ime: '',

    prezime: '',

    imeOca: '',

    email: '',

    password: '',

    jmbg: '',

    brLicneKarte: '',

    datumRodjenja: '',

    mestoRodjenja: '',

    adresaPrebivalista: '',

    opstinaPrebivalista: ''

  });

  const [error, setError] = useState('');

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setFormData({

      ...formData,

      [name]: value

    });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post('http://127.0.0.1:8000/api/register', formData);

      navigate('/login');

    } catch (error) {

      setError('Registration failed. Please check your input and try again.');

      console.error('Error registering:', error);

    }

  };

  return (

    <div className="register-container">

      <form className="register-form" onSubmit={handleSubmit}>

        <h2>Register</h2>

        {error && <p className="error-message">{error}</p>}

        <input

          type="text"

          name="ime"

          value={formData.ime}

          onChange={handleInputChange}

          placeholder="First Name"

          required

        />

        <input

          type="text"

          name="prezime"

          value={formData.prezime}

          onChange={handleInputChange}

          placeholder="Last Name"

          required

        />

        <input

          type="text"

          name="imeOca"

          value={formData.imeOca}

          onChange={handleInputChange}

          placeholder="Father's Name"

        />

        <input

          type="email"

          name="email"

          value={formData.email}

          onChange={handleInputChange}

          placeholder="Email"

          required

        />

        <input

          type="password"

          name="password"

          value={formData.password}

          onChange={handleInputChange}

          placeholder="Password"

          required

        />

        <input

          type="text"

          name="jmbg"

          value={formData.jmbg}

          onChange={handleInputChange}

          placeholder="JMBG"

          required

        />

        <input

          type="text"

          name="brLicneKarte"

          value={formData.brLicneKarte}

          onChange={handleInputChange}

          placeholder="ID Card Number"

          required

        />

        <input

          type="date"

          name="datumRodjenja"

          value={formData.datumRodjenja}

          onChange={handleInputChange}

          placeholder="Date of Birth"

          required

        />

        <input

          type="text"

          name="mestoRodjenja"

          value={formData.mestoRodjenja}

          onChange={handleInputChange}

          placeholder="Place of Birth"

          required

        />

        <input

          type="text"

          name="adresaPrebivalista"

          value={formData.adresaPrebivalista}

          onChange={handleInputChange}

          placeholder="Address"

          required

        />

        <input

          type="text"

          name="opstinaPrebivalista"

          value={formData.opstinaPrebivalista}

          onChange={handleInputChange}

          placeholder="Municipality"

          required

        />

        <button type="submit">Register</button>

      </form>

    </div>

  );

}

export default Register;

Ova komponenta Covid se koristi za prikazivanje statistika vezanih za COVID-19 situaciju u Srbiji. Koristi se React zajedno sa hook-ovima useState i useEffect radi praćenja stanja komponente i izvršavanja efekata kao što je dohvatanje podataka sa eksternog API-ja. Takođe koristi axios za slanje HTTP zahteva ka API-ju koji pruža COVID-19 statistike.

Komponenta počinje definisanjem stanja pomoću useState Hook-a. Stanja covidData i lastUpdated se koriste za čuvanje podataka o COVID-19 statistikama i vremenu poslednjeg osvežavanja podataka.

Zatim se koristi useEffect Hook kako bi se definisalo ponašanje komponente prilikom montiranja i demontiranja. Unutar useEffect-a, definiše se funkcija fetchCovidData koja se poziva prilikom prvog učitavanja komponente i svakih 30 minuta (1800000 milisekundi) nakon toga. Ova funkcija koristi axios za slanje GET zahteva ka COVID-19 API-ju i dohvatanje statistika za Srbiju. Dobijeni podaci se zatim parsiraju i čuvaju u stanju covidData, dok se vreme poslednjeg osvežavanja podataka čuva u stanju lastUpdated.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import './Covid.css';

function Covid() {

    const [covidData, setCovidData] = useState({});

    const [lastUpdated, setLastUpdated] = useState('');

    useEffect(() => {

        fetchCovidData();

        const intervalId = setInterval(fetchCovidData, 1800000);

        return () => clearInterval(intervalId);

    }, []);

    const fetchCovidData = async () => {

        try {

            const response = await axios.get('https://covid-193.p.rapidapi.com/statistics?country=Serbia', {

                headers: {

                    'X-RapidAPI-Key': 'bbb2a722e5msh49d347b4b22fbedp1bb4b9jsn34bcd74d42ae',

                    'X-RapidAPI-Host': 'covid-193.p.rapidapi.com'

                }

            });

            if (response.data && response.data.response && response.data.response.length > 0) {

                const data = response.data.response[0];

                setCovidData({

                    cases: data.cases.total,

                    deaths: data.deaths.total,

                    recovered: data.cases.recovered,

                    active: data.cases.active,

                    critical: data.cases.critical,

                    todayCases: data.cases.new,

                    todayDeaths: data.deaths.new,

                    tests: data.tests.total

                });

                setLastUpdated(new Date().toLocaleString());

            }

        } catch (error) {

            console.error('Došlo je do greške prilikom dohvatanja podataka', error);

        }

    };

    return (

        <div className="covid-container">

            <div id="last-updated">Poslednji put osveženo: {lastUpdated}</div>

            <div id="covid-data">

                <div className="card"><span className="number">{covidData.cases}</span><div className="label"> Ukupno slučajeva</div></div>

                <div className="card"><span className="number">{covidData.deaths}</span> Ukupno smrtnih slučajeva</div>

                <div className="card"><span className="number">{covidData.recovered}</span> Ukupno oporavljenih</div>

                <div className="card"><span className="number">{covidData.active}</span> Aktivni slučajevi</div>

                <div className="card"><span className="number">{covidData.critical || 'N/A'}</span> Kritični slučajevi</div>

                <div className="card"><span className="number">{covidData.todayCases || 'N/A'}</span> Slučajevi danas</div>

                <div className="card"><span className="number">{covidData.todayDeaths || 'N/A'}</span> Smrtni slučajevi danas</div>

                <div className="card"><span className="number">{covidData.tests}</span> Ukupno testova</div>

            </div>

        </div>

    );

}

export default Covid;

Ova komponenta Statistics se koristi za prikazivanje statistika vezanih za zahteve za uslugama.

Koristi se useEffect Hook kako bi se definisalo ponašanje komponente prilikom prvog renderovanja. Unutar useEffect-a, definiše se funkcija fetchStatistics koja se poziva kako bi se dohvatili podaci sa servera o statistikama. Ovi podaci uključuju broj zahteva po uslugama, ukupan broj usluga i ukupan broj korisnika koji nisu administratori.

U JSX-u se definiše struktura komponente gde se prikazuju statistike u obliku naslova, paragrafa i pie chart-a koji prikazuje broj zahteva po uslugama. Kako bi se grafikon pravilno prikazao, postavljen je containerStyle objekat sa stilovima za širinu, visinu i marginu. Osim toga, postavljene su opcije za grafikon kao što je maintainAspectRatio: false kako bi se osiguralo da se grafikon pravilno prikaže bez obzira na dimenzije kontejnera.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { Chart as ChartJS, ArcElement, Tooltip, Legend } from 'chart.js'; // Import necessary components

import { Pie } from 'react-chartjs-2';

// Register the components

ChartJS.register(ArcElement, Tooltip, Legend);

const Statistics = () => {

  const [statistics, setStatistics] = useState(null);

  useEffect(() => {

    const fetchStatistics = async () => {

      try {

        const token = localStorage.getItem('accessToken');

        const headers = {

          'Authorization': `Bearer ${token}`,

        };

        const response = await axios.get('http://127.0.0.1:8000/api/zahtevi/statistics', { headers });

        setStatistics(response.data);

      } catch (error) {

        console.error('Greška prilikom dobijanja statistika:', error);

      }

    };

    fetchStatistics();

  }, []);

  if (!statistics) {

    return <p>Učitavanje statistika...</p>;

  }

  const { zahtevi\_po\_uslugama, ukupan\_broj\_usluga, ukupan\_broj\_korisnika } = statistics;

  // Priprema podataka za pie chart

  const chartData = {

    labels: Object.keys(zahtevi\_po\_uslugama),

    datasets: [

      {

        label: 'Broj zahteva',

        data: Object.values(zahtevi\_po\_uslugama),

        backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0', '#9966FF'],

      },

    ],

  };

  const containerStyle = {

    width: '600px',

    height: '400px',

    margin: '0 auto', // Center the container if desired

  };

  return (

    <div className="statistics-container" style={containerStyle}>

      <h2>Statistike</h2>

      <p>Ukupan broj usluga: {ukupan\_broj\_usluga}</p>

      <p>Ukupan broj korisnika koji nisu admin: {ukupan\_broj\_korisnika}</p>

      <div style={{ position: 'relative', height: '100%' }}>

        <Pie

          data={chartData}

          options={{ maintainAspectRatio: false }}

          id="pie-chart"

        />

      </div>

    </div>

  );

};

export default Statistics;

# Link ka github repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-euprava_2020_0079>

# Literatura

* W3C. (s.d.). HTML i CSS specifikacije. https://www.w3.org/standards/webdesign/htmlcss
* MySQL. (s.d.). Zvanična web stranica MySQL baze podataka. https://www.mysql.com/
* <https://medium.com/@ITesic/uvod-u-react-ekosistem-8ccfad0a1030>
* React. (n.d.). Learn React [Web stranica]. <https://react.dev/learn>
* Laravel. (n.d.). Laravel 10.x Documentation [Web stranica]. https://laravel.com/docs/10.x