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](#_Toc158316729)

[Opis sistema 2](#_Toc158316730)

[Opis slučajeva korišćenja 2](#_Toc158316731)

[Prijava korisnika 2](#_Toc158316732)

[Kreiranje zaposlenog 3](#_Toc158316733)

[Brisanje zaposlenog 4](#_Toc158316734)

[Azuriranje zaposlenog 4](#_Toc158316735)

[Opis arihtekture aplikacije 5](#_Toc158316736)

[Opis procesa slučajeva korišćenja 6](#_Toc158316737)

[Dijagram sekvenci – prijava korisnika 6](#_Toc158316738)

[Dijagram sekvenci – kreiranje zaposlenog 7](#_Toc158316739)

[Dijagram sekvenci – brisanje zaposlenog 8](#_Toc158316740)

[Dijagram sekvenci – azuriranje zaposlenog 8](#_Toc158316741)

[Model podataka 9](#_Toc158316742)

[Specifikacija REST API-ja 11](#_Toc158316743)

[Opis tehnologija korišćenih u aplikaciji 21](#_Toc158316744)

[Specifikacija REST API-ja 22](#_Toc158316745)

[Korisničko uputstvo 31](#_Toc158316746)

[Prikaz delova koda 36](#_Toc158316747)

[Link ka github repozitorijumu 62](#_Toc158316748)

[Reference 63](#_Toc158316749)

# Korisnički zahtev

Potrebno je razviti web aplikaciju koja omogućava firmama da efikasno upravljaju svojim zaposlenima, dokumentima i zadacima. Glavna svrha aplikacije je da pruži organizovano i centralizovano rešenje za sve aspekte upravljanja ljudskim resursima i dokumentacijom unutar firme.

U segmentu registracije i upravljanja zaposlenima, aplikacija omogućava registraciju zaposlenih sa osnovnim informacijama kao što su pozicija, odeljenje, datum početka rada, datum kraja ugovora, plata i pripadnost firmi. Svaki zaposleni ima mogućnost pristupa svojim zadacima i dokumentima firme, što omogućava bolju organizaciju i efikasnost u radu.

Što se tiče upravljanja firmama, aplikacija omogućava kreiranje i upravljanje detaljima firmi kao što su naziv, PIB, matični broj, adresa, kontakt telefon i email. Firme kroz ovu aplikaciju mogu imati centralizovan pristup informacijama o svojim zaposlenima i dokumentima koji su sa njima povezani.

Upravljanje dokumentima je još jedna važna funkcionalnost aplikacije. Dokumenti, ili fajlovi, mogu se dodavati, uređivati i pregledavati unutar firme. Svaki dokument sadrži naziv, opis, putanju čuvanja i povezan je sa određenom firmom, što doprinosi boljem upravljanju dokumentacijom i informacijama unutar firme.

Za zaposlene se mogu kreirati zadaci sa specifičnim informacijama kao što su naziv, opis, rok i status, uključujući opcije kao što su završeno, otkazano ili u izradi. Svaki zadatak je direktno povezan sa određenim zaposlenim, što omogućava efikasno praćenje i upravljanje radnim zadacima unutar firme.

Upravljanje privilegijama je ključno za bezbednost i pravilno funkcionisanje aplikacije. Aplikacija pruža mogućnost dodeljivanja različitih nivoa pristupa dokumentima zaposlenima, na osnovu njihovih privilegija. Ovo omogućava da se definiše koji zaposleni ima pristup kojim dokumentima, čime se obezbeđuje sigurnost i privatnost podataka.

Na kraju, aplikacija treba da podrži osnovne funkcionalnosti kao što su registracija, prijava i odjava korisnika, sa posebnim naglaskom na bezbednost podataka i pristup sistemu. Ovo uključuje zaštitu podataka korisnika i sigurno upravljanje pristupom, što je od vitalnog značaja za integritet i pouzdanost sistema.

# Opis sistema

## Opis slučajeva korišćenja

### 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 zaposlenog

Opis:

Ovaj slučaj korišćenja opisuje osnovne korake za kreiranje zaposlenog.

Akteri:

Administrator (HR menadžer)

Sistem (HR softver)

Učesnici:

Administrator (HR menadžer)

Sistem (HR softver)

Preduslovi:

Administrator ima pristup funkcionalnosti za upravljanje zaposlenima u sistemu.

Sistem je uključen i funkcionalan.

Osnovni scenario:

1.1. Administrator otvara softver i pristupa opciji za kreiranje novog zaposlenog.

1.2. Sistem prikazuje formu za unos podataka o novom zaposlenom, kao što su ime, prezime, pozicija, departman, datum zaposlenja, plata itd.

1.3. Administrator unosi sve relevantne podatke o novom zaposlenom u formu.

1.4. Administrator potvrđuje unos podataka pritiskom na dugme "Kreiraj" ili sličnu opciju.

1.5. Sistem proverava unete podatke i kreira novi zapis o zaposlenom u bazi podataka.

1.6. Sistem prikazuje potvrdu o uspešnom kreiranju zaposlenog.

### Brisanje zaposlenog

Opis:

Ovaj slučaj korišćenja opisuje osnovne korake za brisanje zaposlenog.

Akteri:

Administrator (HR menadžer)

Sistem (HR softver)

Učesnici:

Administrator (HR menadžer)

Sistem (HR softver)

Preduslovi:

Administrator ima pristup funkcionalnosti za upravljanje zaposlenima u sistemu.

Sistem je uključen i funkcionalan.

Osnovni scenario:

1. Administrator otvara HR softver i pristupa opciji za pregled liste zaposlenih.

2. Administrator pronalazi zaposlenog koji treba da bude obrisan iz liste.

3. Administrator bira opciju za brisanje zaposlenog.

4. Sistem prikazuje potvrdu ili pita administratora da potvrdi brisanje zaposlenog.

5. Administrator potvrđuje brisanje zaposlenog.

6. Sistem uklanja zaposlenog iz baze podataka i ažurira listu zaposlenih.

7. Sistem prikazuje potvrdu o uspešnom brisanju zaposlenog.

### Azuriranje zaposlenog

Opis:

Ovaj slučaj korišćenja opisuje osnovne korake za azuriranje zaposlenog.

Akteri:

Administrator (HR menadžer)

Sistem (HR softver)

Učesnici:

Administrator (HR menadžer)

Sistem (HR softver)

Preduslovi: Administrator ima pristup funkcionalnosti za upravljanje zaposlenima u sistemu.

Sistem je uključen i funkcionalan.

Osnovni scenario:

1. Administrator otvara HR softver i pristupa opciji za izmenu podataka o zaposlenom.

2. Administrator pronalazi zaposlenog čije podatke želi da izmeni.

3. Administrator vrši izmene željenih podataka o zaposlenom.

4. Administrator potvrđuje izmene pritiskom na dugme "Sačuvaj".

5. Sistem proverava unete podatke i ažurira zapis o zaposlenom u bazi podataka.

6. Sistem prikazuje potvrdu o uspešnoj izmeni podataka o zaposlenom.

## Opis arihtekture aplikacije

Naša aplikacija je razvijena koristeći moderni web razvojni stack koji kombinuje Laravel, React i Axios, omogućavajući brz i efikasan razvoj kompleksnih web aplikacija. Ovaj segment detaljno objašnjava kako se ove tehnologije međusobno dopunjuju u arhitekturi naše aplikacije.

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 – prijava korisnika

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.

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

Opis je automatski generisan

Slika 1 ds - prijava korisnika

### Dijagram sekvenci – kreiranje zaposlenog

1.1. Administrator otvara softver i pristupa opciji za kreiranje novog zaposlenog.

1.2. Sistem prikazuje formu za unos podataka o novom zaposlenom, kao što su ime, prezime, pozicija, departman, datum zaposlenja, plata itd.

1.3. Administrator unosi sve relevantne podatke o novom zaposlenom u formu.

1.4. Administrator potvrđuje unos podataka pritiskom na dugme "Kreiraj" ili sličnu opciju.

1.5. Sistem proverava unete podatke i kreira novi zapis o zaposlenom u bazi podataka.

1.6. Sistem prikazuje potvrdu o uspešnom kreiranju zaposlenog.

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

Opis je automatski generisan

Slika 2 ds -kreiranje zaposlenog

### Dijagram sekvenci – brisanje zaposlenog

1. Administrator otvara HR softver i pristupa opciji za pregled liste zaposlenih.

2. Administrator pronalazi zaposlenog koji treba da bude obrisan iz liste.

3. Administrator bira opciju za brisanje zaposlenog.

4. Sistem prikazuje potvrdu ili pita administratora da potvrdi brisanje zaposlenog.

5. Administrator potvrđuje brisanje zaposlenog.

6. Sistem uklanja zaposlenog iz baze podataka i ažurira listu zaposlenih.

7. Sistem prikazuje potvrdu o uspešnom brisanju zaposlenog.

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

Opis je automatski generisan

Slika 3 ds -brisanje zaposlenog

### Dijagram sekvenci – azuriranje zaposlenog

1. Administrator otvara HR softver i pristupa opciji za izmenu podataka o zaposlenom.

2. Administrator pronalazi zaposlenog čije podatke želi da izmeni.

3. Administrator vrši izmene željenih podataka o zaposlenom.

4. Administrator potvrđuje izmene pritiskom na dugme "Sačuvaj".

5. Sistem proverava unete podatke i ažurira zapis o zaposlenom u bazi podataka.

6. Sistem prikazuje potvrdu o uspešnoj izmeni podataka o zaposlenom.

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

Opis je automatski generisan

Slika 4 dijagram sekvenci azuriranje zaposlenog

## Model podataka

Model Firma predstavlja osnovni entitet u aplikaciji. Sadrži osnovne informacije o firmi kao što su naziv, PIB, matični broj, adresa, kontakt telefon i email. Ovaj model je centralna tačka oko koje se grupišu ostali entiteti poput zaposlenih i dokumenata.

Zaposleni, koji je izveden iz User modela, sadrži detalje o zaposlenima unutar firme, uključujući poziciju, odeljenje, datum početka rada, datum kraja ugovora, platu i pripadnost firmi. Model User služi kao osnova za autentifikaciju i autorizaciju, dok Zaposleni dodaje specifične poslovne informacije. Kroz ovaj model, aplikacija može efikasno upravljati korisničkim nalogom i pravima pristupa.

Model Fajl je namenjen upravljanju dokumentima unutar firme. Svaki fajl ima naziv, opis, putanju na kojoj je sačuvan i povezan je sa firmom. Ovaj model omogućava firmama da organizuju i pristupaju svojim dokumentima na strukturiran i bezbedan način.

Model Privilegija igra vitalnu ulogu u definisanju prava pristupa i odgovornosti zaposlenih. Ovaj model određuje koji zaposleni ima pristup određenim dokumentima ili zadacima, čime se obezbeđuje bezbednost i integritet poslovnih informacija.

Task model je ključan za upravljanje zadacima unutar firme. Svaki zadatak je povezan sa zaposlenim i sadrži informacije kao što su naziv, opis, rok i status zadatka (npr. završeno, otkazano, u izradi). To omogućava efikasno praćenje i upravljanje radnim zadacima i projektima.

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

Opis je automatski generisan

Slika 5 dijagram klasa

## Specifikacija REST API-ja

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih firmi |
| HTTP metoda | GET |
| URL | api/firme |
| URL parametri |  |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | [  {  "id": 1,  "naziv": "Jacobi-Goodwin",  "PIB": "308249669",  "maticniBroj": "399498864",  "adresa": "72998 Lonie Island\nJudahview, RI 28740-8878",  "kontaktTelefon": "623-456-7536",  "email": "sadie75@bode.net",  "created\_at": "2024-01-05T20:01:09.000000Z",  "updated\_at": "2024-01-05T20:01:09.000000Z"  },  {  "id": 2,  "naziv": "Kreiger-Johns",  "PIB": "877651191",  "maticniBroj": "458700029",  "adresa": "87566 Myah Island Apt. 579\nSouth Francesfurt, OH 17132",  "kontaktTelefon": "1-682-349-3990",  "email": "audrey.bogan@gerlach.org",  "created\_at": "2024-01-05T20:01:09.000000Z",  "updated\_at": "2024-01-05T20:01:09.000000Z"  },  ..  ] |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje firme |
| HTTP metoda | POST |
| URL | /api/firme |
| URL parametri | ?naziv=firma1&PIB=1234232&maticniBroj=45432&adresa=adresa 1&kontaktTelefon=1111111&email=firma1@gmail.com |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Firma uspešno kreirana",      "firma": {          "naziv": "firma1",          "PIB": "1234232",          "maticniBroj": "45432",          "adresa": "adresa 1",          "kontaktTelefon": "1111111",          "email": "firma1@gmail.com",          "updated\_at": "2024-01-05T21:06:58.000000Z",          "created\_at": "2024-01-05T21:06:58.000000Z",          "id": 6      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje firme |
| HTTP metoda | PUT |
| URL | /api/ firme |
| URL parametri | 6?naziv=firma11111&PIB=1234232&maticniBroj=45432&adresa=adresa 1&kontaktTelefon=1111111&email=firma1@gmail.com |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Firma uspešno azurirana",      "firma": {          "id": 6,          "naziv": "firma11111",          "PIB": "1234232",          "maticniBroj": "45432",          "adresa": "adresa 1",          "kontaktTelefon": "1111111",          "email": "firma1@gmail.com",          "created\_at": "2024-01-05T21:06:58.000000Z",          "updated\_at": "2024-01-05T21:09:19.000000Z"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje firme |
| HTTP metoda | DELETE |
| URL | /api/ firme |
| URL parametri | 7 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Firma obrisana"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih taskova |
| HTTP metoda | GET |
| URL | /api/task |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | "data": [          {              "id": 1,              "naziv": "Reprehenderit magnam assumenda blanditiis rem est est occaecati amet.",              "opis": "Sapiente totam sunt et labore voluptatem laboriosam dolorum. Totam omnis et aliquid. Voluptatibus ea expedita esse animi animi ut.",              "rok": "1991-03-06",              "status": "u izradi",              "zaposleni": {                  "id": 4,                  "name": "Jameson Pfeffer",                  "email": "paula.leannon@example.com",                  "pozicija": "Benefits Specialist",                  "odeljenje": "repellat",                  "datum\_pocetka\_rada": "1988-11-22",                  "datum\_kraja\_ugovora": "1970-05-17",                  "plata": "8915.90",                  "firma": {                      "id": 1,                      "naziv": "Jacobi-Goodwin",                      "PIB": "308249669",                      "maticniBroj": "399498864",                      "adresa": "72998 Lonie Island\nJudahview, RI 28740-8878",                      "kontaktTelefon": "623-456-7536",                      "email": "sadie75@bode.net",                      "created\_at": "2024-01-05T20:01:09.000000Z",                      "updated\_at": "2024-01-05T20:01:09.000000Z"                  }              }          },          {              "id": 2,              "naziv": "Unde optio et iste itaque consequatur.",              "opis": "Ut id qui laudantium. Alias rerum necessitatibus qui aperiam. Fugiat omnis atque est placeat. Sint id eos non incidunt.",              "rok": "2005-11-08",              "status": "otkazano",              "zaposleni": {                  "id": 7,                  "name": "Brando Lebsack",                  "email": "owitting@example.net",                  "pozicija": "Business Development Manager",                  "odeljenje": "dolorem",                  "datum\_pocetka\_rada": "2009-07-21",                  "datum\_kraja\_ugovora": "1990-06-14",                  "plata": "5064.61",                  "firma": {                      "id": 3,                      "naziv": "Kunde, Gerlach and O'Keefe",                      "PIB": "768579133",                      "maticniBroj": "686542762",                      "adresa": "592 Rippin Land Suite 917\nNew Barney, NM 87843-3733",                      "kontaktTelefon": "339.670.4321",                      "email": "isom75@kilback.com",                      "created\_at": "2024-01-05T20:01:09.000000Z",                      "updated\_at": "2024-01-05T20:01:09.000000Z"                  }              }          }, |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje taska |
| HTTP metoda | POST |
| URL | /api/task |
| URL parametri | ?zaposleni\_id=1&naziv=task1&rok=2024-01-10&status=zavrseno |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Task uspešno kreiran",      "task": {          "id": 11,          "naziv": "task1",          "opis": **null**,          "rok": "2024-01-10",          "status": "zavrseno",          "zaposleni": {              "id": 1,              "name": "Mr. Hugh Stroman PhD",              "email": "yvolkman@example.com",              "pozicija": "Photographic Restorer",              "odeljenje": "illo",              "datum\_pocetka\_rada": "1995-08-24",              "datum\_kraja\_ugovora": "2015-07-10",              "plata": "2566.68",              "firma": {                  "id": 1,                  "naziv": "Jacobi-Goodwin",                  "PIB": "308249669",                  "maticniBroj": "399498864",                  "adresa": "72998 Lonie Island\nJudahview, RI 28740-8878",                  "kontaktTelefon": "623-456-7536",                  "email": "sadie75@bode.net",                  "created\_at": "2024-01-05T20:01:09.000000Z",                  "updated\_at": "2024-01-05T20:01:09.000000Z"              }          }      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje taska |
| HTTP metoda | PUT |
| URL | /api/task |
| URL parametri | 11?zaposleni\_id=1&naziv=task11111&rok=2024-01-10&status=zavrseno |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Task uspešno ažuriran",      "task": {          "id": 11,          "naziv": "task11111",          "opis": **null**,          "rok": "2024-01-10",          "status": "zavrseno",          "zaposleni": {              "id": 1,              "name": "Mr. Hugh Stroman PhD",              "email": "yvolkman@example.com",              "pozicija": "Photographic Restorer",              "odeljenje": "illo",              "datum\_pocetka\_rada": "1995-08-24",              "datum\_kraja\_ugovora": "2015-07-10",              "plata": "2566.68",              "firma": {                  "id": 1,                  "naziv": "Jacobi-Goodwin",                  "PIB": "308249669",                  "maticniBroj": "399498864",                  "adresa": "72998 Lonie Island\nJudahview, RI 28740-8878",                  "kontaktTelefon": "623-456-7536",                  "email": "sadie75@bode.net",                  "created\_at": "2024-01-05T20:01:09.000000Z",                  "updated\_at": "2024-01-05T20:01:09.000000Z"              }          }      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje taska |
| HTTP metoda | DELETE |
| URL | /api/task |
| URL parametri | 11 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Task obrisan"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | name=pera&email=peric@gmail.com&password=peraperapera |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "access\_token": "1|pV79bHyuYLO5UpcORAEouBBqGvCYZ0z3FBnuOoMn",      "token\_type": "Bearer",      "user": {          "name": "pera",          "email": "peric@gmail.com",          "updated\_at": "2024-01-05T21:18:36.000000Z",          "created\_at": "2024-01-05T21:18:36.000000Z",          "id": 11      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | email=peric@gmail.com&password=peraperapera |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "access\_token": "2|2ZHQEDdeds0A0dMjAUgvYbbHnCgPKkHLs4gWnuMi",      "token\_type": "Bearer",      "user": {          "id": 11,          "name": "pera",          "email": "peric@gmail.com",          "email\_verified\_at": **null**,          "created\_at": "2024-01-05T21:18:36.000000Z",          "updated\_at": "2024-01-05T21:18:36.000000Z",          "pozicija": **null**,          "odeljenje": **null**,          "datum\_pocetka\_rada": **null**,          "datum\_kraja\_ugovora": **null**,          "plata": **null**,          "firma\_id": **null**      }  } |
| 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": "Successfully logged out"  } |
| Format izlaznih parametara | application/json |

# Opis tehnologija korišćenih u aplikaciji

Laravel je PHP framework koji je korišćen za razvoj serverske strane aplikacije. On omogućava brzo i efikasno kreiranje robustnih web aplikacija. Laravel pruža različite alate i funkcionalnosti za upravljanje rutama, bazom podataka, autentifikacijom, validacijom, sesijama i još mnogo toga. Ovaj framework je odabran zbog svoje jednostavnosti korišćenja, jasne dokumentacije i velike zajednice korisnika koja pruža podršku i resurse za učenje.

React je JavaScript biblioteka za izgradnju korisničkih interfejsa. Korišćen je za razvoj klijentske strane aplikacije. React omogućava kreiranje reusable komponenti, dinamičko upravljanje stanjem, i efikasno upravljanje prikazom podataka. Zahvaljujući Virtual DOM-u, React pruža brz i efikasan način renderovanja korisničkog interfejsa.

Axios je JavaScript biblioteka koja omogućava slanje HTTP zahteva iz browser-a ili Node.js okruženja. Korišćen je u React komponentama za komunikaciju sa serverskom stranom aplikacije. Axios pruža jednostavnu i čistu sintaksu za obavljanje AJAX zahteva i rukovanje odgovorima od servera.

CSS (Cascading Style Sheets) je jezik koji se koristi za stilizovanje HTML elemenata na web stranici. Korišćen je za definisanje izgleda i stilova korisničkog interfejsa aplikacije. CSS se koristi za definisanje boja, fontova, rasporeda elemenata, animacija i ostalih vizuelnih efekata.

Postman je alat za razvoj API-ja koji omogućava slanje HTTP zahteva i testiranje API-ja. Korišćen je za testiranje funkcionalnosti API endpoint-ova i rukovanje sa odgovorima od servera. Postman omogućava jednostavno slanje različitih vrsta zahteva (GET, POST, PUT, DELETE) i pregled odgovora od servera.

XAMPP je besplatan i open-source paket softvera koji se koristi za kreiranje lokalnog web servera. Uključuje Apache HTTP server, MySQL bazu podataka i interpretere za PHP i Perl. Korišćen je za postavljanje lokalnog razvojnog okruženja za Laravel aplikaciju.

Visual Studio Code je besplatan i open-source code editor koji je korišćen za razvoj aplikacije. On pruža bogat set funkcionalnosti za pisanje, uređivanje i debugiranje koda. Visual Studio Code podržava različite programske jezike i tehnologije, uključujući HTML, CSS, JavaScript, PHP i React.

# Korisničko uputstvo

Prilikom pokretanja aplikacije otvara se početna stranica kao na slici ispod

Slika na kojoj se nalazi tekst, snimak ekrana, dizajn

Opis je automatski generisan

Slika 6 home stranica

Odabirom opcije Firme iz navigacionog menija, korisniku se otvara tabelarni pregled svih firmi

Slika na kojoj se nalazi tekst, softver, Ikona na računaru, broj

Opis je automatski generisan

Slika 7 prikaz firmi

Na ovoj stranici korisnik može da pretražuje firme po nazivu ili PIBu, a klikom na Dugme detalji se prikazuju svi detalji o toj firmi

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

Opis je automatski generisan

Slika 8 prikaz detalja o firmi

Odabriom opcije Prijava iz navigacionog menija, korisniku se otvara stranica koja sadrži login formu

Slika na kojoj se nalazi snimak ekrana, tekst, dizajn

Opis je automatski generisan

Slika 9 login forma

Kada se korisnik prijavi otvara mu se stranica na kojoj može da vidi i filtrira taskove koji su namenjeni za njega

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

Opis je automatski generisan

Slika 10 todo lista ulogovanog korisnika

Odabirom opcije fajlovi korisniku se otvara stranica na kojoj može da pregleda sve fajlove koji se nalaze na dropbox nalogu firme

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

Opis je automatski generisan

Slika 11 prikaz svih fajlova firme

Klikom na dugme otvori korisniku se otvara taj fajl na dropbox platofrmi.

S obrzirom na to da firme imaju veliki broj fajlova, radi lakšeg snalaženja dodata je paginacija na tabelu.

Odabirom opcije upload, korisniku se otvara stranica preko koje može da otpremi fajl na dropbox platformu firme.

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

Opis je automatski generisan

Slika 12 upload fajla

Odabirom opcije Odjava, korisnici se odjavljuju iz aplikacije.

Kada se u aplikaciju prijavi admin, on ima pristup grafikonima, kao na slici ispod.

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

Opis je automatski generisan

Slika 13 grafikoni na admin stranici

Administrator ima mogućnost upravljana zaposlenima – dodavanje, brisanje, izmena.. Ovo je prikazano slikom ispod.

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

Opis je automatski generisan

Slika 14 CRUD operacije za zaposlenog od strane admina

# Prikaz delova koda

U ovom delu ćemo opisati samo najznačajnije kontrolere.

1. FajlController.php

Ovaj kontroler upravlja akcijama vezanim za dokumente (fajlove) unutar aplikacije. To uključuje funkcionalnosti kao što su dodavanje novih dokumenata, ažuriranje, dohvatanje detalja o specifičnim dokumentima, te brisanje dokumenata.

FajlController koristi DropBox API za upravljanje fajlovima ,a dizajniran je da omogući korisnicima da izvršavaju različite operacije sa fajlovima kao što su pregledanje, preuzimanje, otpremanje, ažuriranje i brisanje fajlova u DropBox-u.

U konstruktoru kontrolera, zavisnost DropBoxService se injektira, omogućavajući pristup funkcionalnostima DropBox API-ja. Ovaj servis je zatim dostupan kroz instancu $this->dropBox, koja se koristi u svim metodama kontrolera.

class FajlController extends Controller

{

    protected $dropBox;

    public function \_\_construct(DropBoxService $dropBox)

    {

        $this->dropBox = $dropBox;

    }

Metoda index koristi listFiles funkciju iz DropBoxService da dohvati listu svih fajlova sa DropBox naloga. Rezultat, koji je lista fajlova, vraća se kao JSON odgovor.

    public function index()

    {

        $files = $this->dropBox->listFiles();

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

    }

update metoda omogućava ažuriranje postojećeg fajla. U ovoj metodi, novi fajl se dohvata iz zahteva, a zatim se koristi updateFile metoda iz DropBoxService-a da se ažurira odabrani fajl na DropBox-u. Rezultat operacije se vraća kao JSON odgovor.

    public function upload(Request $request)

    {

        $file = $request->file('file');

        if (!$file) {

            return response()->json(['error' => 'Nema fajla za otpremanje.'], 400);

        }

        $path = '/public/files/' . $file->getClientOriginalName();

        $contents = file\_get\_contents($file->getRealPath());

        $result = $this->dropBox->uploadFile($path, $contents);

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

    }

Metoda destroy omogućava brisanje fajla sa DropBox naloga. ID fajla se dohvata iz URL-a, a deleteFile metoda DropBoxService-a se koristi za brisanje fajla. Metoda vraća JSON odgovor koji potvrđuje brisanje fajla.

    public function destroy($fileId)

    {

        $this->dropBox->deleteFile($fileId);

        return response()->json(['message' => 'Fajl obrisan']);

    }

1. FirmaController.php

FirmaController se bavi upravljanjem firmama unutar aplikacije. Osnovne operacije koje ovaj kontroler podržava uključuju kreiranje novih firmi, izmenu postojećih informacija o firmama, pregled detalja o firmama, i brisanje firmi iz sistema.

<?php

namespace App\Http\Controllers;

use App\Models\Firma;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

class FirmaController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index() //GET

    {

        $sveFirme = Firma::all();

        return $sveFirme;

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request) //POST

    {

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

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

            'PIB' => 'required|string|max:12|unique:firmas,PIB',

            'maticniBroj' => 'required|string|max:12|unique:firmas,maticniBroj',

            'adresa' => 'required|string',

            'kontaktTelefon' => 'required|string',

            'email' => 'required|email|unique:firmas,email',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $firma = new Firma;

        $firma->naziv = $request->naziv;

        $firma->PIB = $request->PIB;

        $firma->maticniBroj = $request->maticniBroj;

        $firma->adresa = $request->adresa;

        $firma->kontaktTelefon = $request->kontaktTelefon;

        $firma->email = $request->email;

        $firma->save();

        return response()->json(['message' => 'Firma uspešno kreirana', 'firma' => $firma], 201);

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  \App\Models\Firma  $firma

     \* @return \Illuminate\Http\Response

     \*/

    public function show($id) //GET

    {

        $f=Firma::find($id);

        if($f){

            return $f;

        }

        return response()->json(['error' => 'Ne postoji ta firma'], 404);

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*

     \* @param  \App\Models\Firma  $firma

     \* @return \Illuminate\Http\Response

     \*/

    public function edit(Firma $firma)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  \App\Models\Firma  $firma

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, $id)//PUT

    {

        $firma = Firma::find($id);

        if($firma){

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

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

                'PIB' => 'required|string|max:12',

                'maticniBroj' => 'required|string|max:12',

                'adresa' => 'required|string',

                'kontaktTelefon' => 'required|string',

                'email' => 'required|email',

            ]);

            if ($validator->fails()) {

                return response()->json($validator->errors(), 422);

            }

            $firma->naziv = $request->naziv;

            $firma->PIB = $request->PIB;

            $firma->maticniBroj = $request->maticniBroj;

            $firma->adresa = $request->adresa;

            $firma->kontaktTelefon = $request->kontaktTelefon;

            $firma->email = $request->email;

            $firma->save();

            return response()->json(['message' => 'Firma uspešno azurirana', 'firma' => $firma], 200);

        }else{

            return response()->json(['error' => 'Ne postoji ta firma'], 404);

        }

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*

     \* @param  \App\Models\Firma  $firma

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy($id) //DELETE

    {

        $f = Firma::with('zaposleni','fajlovi')->find($id);

        if($f){

            $f->delete();

            return response()->json(['message' => 'Firma obrisana'], 200);

        }else{

            return response()->json(['error' => 'Ne postoji ta firma'], 404);

        }

    }

}

1. TaskController.php

TaskController je zadužen za upravljanje zadacima u aplikaciji. Ovaj kontroler omogućava kreiranje zadataka, ažuriranje, dohvatanje informacija o zadacima, te brisanje zadataka.

<?php

namespace App\Http\Controllers;

use App\Http\Resources\TaskResource;

use App\Models\Task;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

class TaskController extends Controller

{

    public function index() //GET

    {

        $sviTaskovi = Task::all();

        return TaskResource::collection($sviTaskovi);

    }

    public function store(Request $request)

    {

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

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

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

            'opis' => 'nullable|string',

            'rok' => 'required|date',

            'status' => 'required|in:zavrseno,otkazano,u izradi',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $task = Task::create($request->all());

        return response()->json(['message' => 'Task uspešno kreiran', 'task' => new TaskResource($task)], 201);

    }

    public function show($id) //GET

    {

        $task = Task::find($id);

        if ($task) {

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

        }

        return response()->json(['error' => 'Task nije pronađen'], 404);

    }

    public function update(Request $request, $id)

    {

        $task = Task::find($id);

        if ($task) {

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

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

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

                'opis' => 'nullable|string',

                'rok' => 'required|date',

                'status' => 'required|in:zavrseno,otkazano,u izradi',

            ]);

            if ($validator->fails()) {

                return response()->json($validator->errors(), 422);

            }

            $task->update($request->all());

            return response()->json(['message' => 'Task uspešno ažuriran', 'task' => new TaskResource($task)], 200);

        } else {

            return response()->json(['error' => 'Task nije pronađen'], 404);

        }

    }

    public function destroy($id)

    {

        $task = Task::find($id);

        if ($task) {

            $task->delete();

            return response()->json(['message' => 'Task obrisan'], 200);

        } else {

            return response()->json(['error' => 'Task nije pronađen'], 404);

        }

    }

}

1. ZaposleniController.php

ZaposleniController upravlja svim aspektima vezanim za zaposlene unutar aplikacije. To uključuje registraciju novih zaposlenih, ažuriranje njihovih podataka, pregled informacija o zaposlenima, i uklanjanje zaposlenih iz sistema.

<?php

namespace App\Http\Controllers;

use App\Http\Resources\ZaposleniResource;

use App\Models\User;

use App\Models\Zaposleni;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

class ZaposleniController extends Controller

{

    public function index()

    {

        $sviZaposleni = User::paginate(5);

        return  ZaposleniResource::collection($sviZaposleni);

    }

    public function store(Request $request)

    {

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

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

            'email' => 'required|email|unique:users,email',

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

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

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

            'datum\_pocetka\_rada' => 'required|date',

            'datum\_kraja\_ugovora' => 'nullable|date',

            'plata' => 'required|numeric',

            'firma\_id' => 'required|exists:firmas,id',

        ]);

        if ($validator->fails()) {

            return response()->json($validator->errors(), 422);

        }

        $zaposleni = new User;

        $zaposleni->name = $request->name;

        $zaposleni->email = $request->email;

        $zaposleni->password = Hash::make($request->password);

        $zaposleni->pozicija = $request->pozicija;

        $zaposleni->odeljenje = $request->odeljenje;

        $zaposleni->datum\_pocetka\_rada = $request->datum\_pocetka\_rada;

        $zaposleni->datum\_kraja\_ugovora = $request->datum\_kraja\_ugovora;

        $zaposleni->plata = $request->plata;

        $zaposleni->firma\_id = $request->firma\_id;

        $zaposleni->save();

        return response()->json(['message' => 'Zaposleni uspešno kreiran', 'zaposleni' => $zaposleni], 201);

    }

    public function show($id)

    {

        $zaposleni = User::find($id);

        if ($zaposleni) {

            return new ZaposleniResource($zaposleni);

        }

        return response()->json(['error' => 'Zaposleni nije pronađen'], 404);

    }

    public function update(Request $request, $id) //PUT

    {

        $zaposleni = User::find($id);

        if ($zaposleni) {

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

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

                'email' => 'required|email',

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

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

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

                'datum\_pocetka\_rada' => 'required|date',

                'datum\_kraja\_ugovora' => 'nullable|date',

                'plata' => 'required|numeric',

                'firma\_id' => 'required|exists:firmas,id',

            ]);

            if ($validator->fails()) {

                return response()->json($validator->errors(), 422);

            }

            $zaposleni->update($request->all());

            return response()->json(['message' => 'Zaposleni uspešno ažuriran', 'zaposleni' => new ZaposleniResource($zaposleni)], 200);

        } else {

            return response()->json(['error' => 'Zaposleni nije pronađen'], 404);

        }

    }

    public function destroy($id)

    {

        $zaposleni = User::find($id);

        if ($zaposleni) {

            $zaposleni->delete();

            return response()->json(['message' => 'Zaposleni obrisan'], 200);

        } else {

            return response()->json(['error' => 'Zaposleni nije pronađen'], 404);

        }

    }

}

1. AuthController.php

AuthController je odgovoran za autentifikaciju korisnika u aplikaciji. Ovaj kontroler upravlja procesima kao što su registracija korisnika, prijava i odjava, te promena lozinki. AuthController je ključan za održavanje sigurnosti i integriteta korisničkih naloga unutar aplikacije, obezbeđujući da samo ovlašćeni korisnici imaju pristup određenim funkcionalnostima i podacima.

<?php

namespace App\Http\Controllers;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

class AuthController extends Controller

{

    public function register(Request $request)

    {

        $validatedData = $request->validate([

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

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

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

        ]);

        $user = User::create([

            'name' => $validatedData['name'],

            'email' => $validatedData['email'],

            'password' => Hash::make($validatedData['password']),

        ]);

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

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

    }

    public function login(Request $request)

    {

        $request->validate([

            'email' => 'required|email',

            'password' => 'required',

        ]);

        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', 'user' => $user]);

    }

    public function logout(Request $request)

    {

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

        return response()->json(['message' => 'Successfully logged out']);

    }

}

Komponenta LoginPage je dizajnirana da omogući korisnicima prijavljivanje u aplikaciju. Ona koristi stanje React-a (useState) za upravljanje korisničkim unosom e-pošte i lozinke, a takođe koristi axios za slanje asinhronog zahteva za prijavu na server. Nakon što korisnik unese svoje kredencijale i potvrdi formu, handleLogin funkcija obrađuje zahtev. U slučaju uspešne prijave, korisnički token se čuva u sessionStorage za dalju autentifikaciju tokom sesije, a korisnik se preusmerava na stranicu sa zadacima.

import React, { useState } from 'react';

import axios from 'axios';

import './LoginPage.css';

import InputField from './InputField';

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

const LoginPage = ({setToken}) => {

  const [email, setEmail] = useState('');

  const [password, setPassword] = useState('');

  let navigate = useNavigate();

  const handleLogin = async (event) => {

    event.preventDefault();

    try {

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

        email,

        password,

      });

      const { access\_token } = response.data;

      sessionStorage.setItem('authToken', access\_token);

      sessionStorage.setItem('userId', response.data.user.id);

      setToken(access\_token);

      navigate('/tasks');

    } catch (error) {

      console.error('Login error', error);

    }

  };

  return (

    <div className="login-page">

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

        <InputField

            type="email"

            value={email}

            onChange={(e) => setEmail(e.target.value)}

            placeholder="Email"

            />

            <InputField

            type="password"

            value={password}

            onChange={(e) => setPassword(e.target.value)}

            placeholder="Password"

            />

        <button type="submit" className="login-button">Login</button>

      </form>

    </div>

  );

};

export default LoginPage;

komponenta Firme

Komponenta Firme služi a prikazivanje i pretraživanje informacija o firmama. Ona koristi React hooks (useState i useEffect) za upravljanje stanjem i asinhrono dohvatanje podataka o firmama putem axios biblioteke. Korisnici mogu pretraživati firme po nazivu ili PIB-u kroz tekstualno polje, a filtrirani rezultati se prikazuju tabelarno. Svaki red tabele, koji predstavlja pojedinačnu firmu, generiše se korišćenjem reusable komponente FirmaRow. Ova komponenta prikazuje osnovne informacije o firmi i omogućava navigaciju do detaljnijih podataka klikom na dugme "Detalji", koristeći useNavigate hook iz react-router-dom.

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

import axios from 'axios';

import './Firme.css';

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

import FirmaRow from './FirmaRow';

const Firme = () => {

  const [firme, setFirme] = useState([]);

  const [searchTerm, setSearchTerm] = useState('');

  const navigate = useNavigate();

  useEffect(() => {

    const dohvatiFirme = async () => {

      try {

        const response = await axios.get('http://127.0.0.1:8000/api/firme');

        setFirme(response.data);

      } catch (error) {

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

      }

    };

    dohvatiFirme();

  }, []);

  const handleSearchChange = (e) => {

    setSearchTerm(e.target.value);

  };

  const filteredFirme = firme.filter(

    firma =>

      firma.naziv.toLowerCase().includes(searchTerm.toLowerCase()) ||

      firma.PIB.toLowerCase().includes(searchTerm.toLowerCase())

  );

  return (

    <div className="firme-container">

      <input

        type="text"

        placeholder="Pretraži po nazivu ili PIB-u"

        value={searchTerm}

        onChange={handleSearchChange}

        className="search-input"

      />

      <table className="firme-table">

        <thead>

          <tr>

            <th>ID</th>

            <th>Naziv</th>

            <th>PIB</th>

            <th>Detalji</th>

          </tr>

        </thead>

        <tbody>

          {filteredFirme.map((firma) => (

            <FirmaRow key={firma.id} firma={firma} onNavigate={navigate} />

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default Firme;

Komponenta Tasks igra ključnu ulogu u prikazivanju i organizovanju zadataka (taskova) dodeljenih korisnicima. Ova komponenta efikasno koristi prilagođenu React kuku useTasks za dohvatanje liste zadataka sa servera, uzimajući u obzir ID trenutno prijavljenog korisnika, čuvanog u sessionStorage. Kada se učita, komponenta Tasks dinamički prikazuje zadatke koristeći tabelarni format, gde svaki red tabele, kreiran kroz TableRow reusable komponentu, sadrži detalje kao što su ID zadatka, njegov naziv, opis, rok za završetak i trenutni status.

Dodatna funkcionalnost komponente uključuje mogućnost filtriranja zadataka na osnovu njihovog statusa, kao što su "završeno", "otkazano" i "u izradi". Ovaj filter omogućava korisnicima da brzo pronađu i fokusiraju se na zadatke koji su relevantni za njihov trenutni radni tok. Selekcija filtera se vrši kroz padajući meni, a filtrirani zadaci se odmah prikazuju u tabeli, čime se postiže interaktivnost i veća korisnička praktičnost.

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

import axios from 'axios';

import './Tasks.css';

import useTasks from './useTasks';

import TableRow from './TableRow';

const Tasks = () => {

    const userId = parseInt(sessionStorage.getItem('userId'));

    const tasks = useTasks('http://127.0.0.1:8000/api/task', userId);

    const [filter, setFilter] = useState('sve');

    const filteredTasks = filter === 'sve' ? tasks : tasks.filter(task => task.status === filter);

  return (

    <div className="tasks-container">

         <div className="filter-container">

            <label htmlFor="statusFilter">Filter po statusu: </label>

            <select id="statusFilter" onChange={(e) => setFilter(e.target.value)} value={filter}>

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

            <option value="zavrseno">Završeno</option>

            <option value="otkazano">Otkazano</option>

            <option value="u izradi">U izradi</option>

            </select>

        </div>

      <table className="tasks-table">

        <thead>

          <tr>

            <th>ID</th>

            <th>Naziv</th>

            <th>Opis</th>

            <th>Rok</th>

            <th>Status</th>

          </tr>

        </thead>

        <tbody>

          {filteredTasks.map(task => (

            <TableRow key={task.id} task = {task}/>

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default Tasks;

import { useState, useEffect } from 'react';

import axios from 'axios';

const useTasks = (url, userId) => {

  const [tasks, setTasks] = useState([]);

  useEffect(() => {

    const fetchTasks = async () => {

      try {

        const response = await axios.get(url, {

          headers: {

            Authorization: `Bearer ${sessionStorage.getItem('authToken')}`

          }

        });

        const filteredTasks = response.data.data.filter(task => task.zaposleni.id === userId);

        setTasks(filteredTasks);

      } catch (error) {

        console.error('Greška prilikom dohvatanja taskova', error);

      }

    };

    fetchTasks();

  }, [url, userId]);

  return tasks;

};

export default useTasks;

Ovaj kod predstavlja React komponentu nazvanu **AdminPage**, koja je odgovorna za prikaz statističkih podataka o firmama. Koristi se za dohvatanje podataka o broju zaposlenih i broju fajlova po firmi putem HTTP zahteva ka API-ju, a zatim ih prikazuje u obliku grafova pomoću biblioteke **react-chartjs-2**.

U prvom delu koda definiše se komponenta **AdminPage** kao funkcionalni komponenta. U njoj se koriste React-ove Hook-ove **useState** i **useEffect** kako bi se omogućilo praćenje stanja komponente i izvršavanje efekata nakon što se komponenta montira.

Zatim se definiše funkcija **fetchData** koja koristi asinhroni pristup za dohvatanje podataka sa servera. Podaci se dohvaćaju putem HTTP GET zahteva na **http://127.0.0.1:8000/api/statistics**. Ovaj zahtev se autorizuje koristeći token koji je smešten u sessionStorage. Dobijeni podaci se smeštaju u stanje komponente kroz **setData**.

U render metodi komponente, prikazuju se dva grafa ukoliko postoje podaci (**data**). Prvi graf prikazuje broj zaposlenih po firmi, dok drugi graf prikazuje broj fajlova po firmi. Podaci za graf se dobijaju iz **data** objekta koji se prethodno dobio iz servera. Grafovi su implementirani kao Bar (stubasti) grafovi kroz komponentu **Bar** iz biblioteke **react-chartjs-2**. Svaki graf ima odgovarajuće oznake (labels) i podatke (data) koje su dobijene iz **data** objekta.

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

import axios from 'axios';

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

import { Chart, registerables } from 'chart.js';

Chart.register(...registerables);

const AdminPage = () => {

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

  useEffect(() => {

    fetchData();

  }, []);

  const fetchData = async () => {

    try {

      const token = sessionStorage.getItem('authToken');

      if (!token) {

        console.error('Token nije pronađen u sessionStorage.');

        return;

      }

      const config = {

        headers: {

          Authorization: `Bearer ${token}`

        }

      };

      const response = await axios.get('http://127.0.0.1:8000/api/statistics', config);

      setData(response.data);

    } catch (error) {

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

    }

  };

  return (

    <div>

      <h1>Statistika po firmama</h1>

      {data && (

        <>

          <div>

            <h2>Broj zaposlenih po firmi</h2>

            <Bar

              key="employees-chart"

              data={{

                labels: data.employees\_per\_company.map(item => item.naziv),

                datasets: [

                  {

                    label: 'Broj zaposlenih',

                    data: data.employees\_per\_company.map(item => item.broj\_zaposlenih),

                    backgroundColor: 'rgba(75, 192, 192, 0.2)',

                    borderColor: 'rgba(75, 192, 192, 1)',

                    borderWidth: 1,

                  },

                ],

              }}

              options={{

                scales: {

                  y: {

                    beginAtZero: true

                  }

                }

              }}

            />

          </div>

          <div>

            <h2>Broj fajlova po firmi</h2>

            <Bar

              key="files-chart"

              data={{

                labels: Object.keys(data.files\_per\_company),

                datasets: [

                  {

                    label: 'Broj fajlova',

                    data: Object.values(data.files\_per\_company),

                    backgroundColor: 'rgba(255, 99, 132, 0.2)',

                    borderColor: 'rgba(255, 99, 132, 1)',

                    borderWidth: 1,

                  },

                ],

              }}

              options={{

                scales: {

                  y: {

                    beginAtZero: true

                  }

                }

              }}

            />

          </div>

        </>

      )}

    </div>

  );

};

export default AdminPage;

Ovaj kod predstavlja React komponentu nazvanu **FileUploadForm**, koja omogućava korisniku da otpremi fajl na server.

Funkcija **handleFileChange** se poziva kada korisnik izabere fajl pomoću input polja tipa "file". Ta funkcija postavlja stanje fajla na izabrani fajl.

Funkcija **handleSubmit** se poziva kada korisnik potvrdi formu za otpremanje fajla. Ova funkcija prvo proverava da li je fajl izabran, a zatim kreira **FormData** objekat i dodaje izabrani fajl u njega. Takođe, dohvaća **firmaId** iz sessionStorage-a i dodaje ga u FormData objekat kako bi se fajl otpremio u okviru odgovarajuće firme.

Zatim se koristi **axios** za slanje POST zahteva ka serveru na odgovarajuću putanju za otpremanje fajlova. U zaglavljima zahteva se šalje token za autorizaciju i **Content-Type** se postavlja na **multipart/form-data** kako bi se pravilno poslali podaci. Nakon što se fajl uspešno otpremi, korisniku se prikazuje poruka o uspešnom otpremanju.

import React, { useState } from 'react';

import axios from 'axios';

const FileUploadForm = () => {

  const [file, setFile] = useState(null);

  const handleFileChange = (e) => {

    setFile(e.target.files[0]);

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!file) {

      alert('Morate izabrati fajl za otpremanje.');

      return;

    }

    const formData = new FormData();

    formData.append('file', file);

    const firmaId = sessionStorage.getItem('firmaId'); // Dohvatanje firmeId iz sessionStorage-a

    formData.append('firmaId', firmaId); // Dodavanje firmeId u formData

    try {

        const token = sessionStorage.getItem('authToken');

        const response = await axios.post('http://127.0.0.1:8000/api/fajlovi/upload', formData, {

            headers: {

              'Content-Type': 'multipart/form-data',

              'Authorization': `Bearer ${token}`

            }

          });

      console.log('Odgovor sa servera:', response.data);

      alert('Fajl je uspešno otpremljen.');

    } catch (error) {

      console.error('Greška prilikom otpremanja fajla:', error);

      alert('Došlo je do greške prilikom otpremanja fajla.');

    }

  };

  return (

  <> <h2 className="todo-list-title">UPLOADUJ FILE</h2>

    <div className="login-page">

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

        <input type="file" onChange={handleFileChange} />

        <button type="submit" className="login-button">Otpremi fajl</button>

      </form>

    </div></>

  );

};

export default FileUploadForm;

Ovaj kod predstavlja React komponentu nazvanu **Zaposleni**, koja omogućava administraciju zaposlenih u okviru sistema. Komponenta omogućava pregled, kreiranje, ažuriranje i brisanje podataka o zaposlenima.

Unutar nje se koriste **useState** Hook-ovi kako bi se pratilo stanje zaposlenih (**zaposleni**), stanje učitavanja (**loading**), novi zaposleni koji se unosi (**newZaposleni**) i ID izabranog zaposlenog za ažuriranje (**selectedZaposleniId**).

Komponenta sadrži niz funkcija za upravljanje promenama stanja, kao što su **handleInputChange** za praćenje promena u input poljima forme, **handleCreateClick** za kreiranje novog zaposlenog, **handleEditClick** za izmenu postojećeg zaposlenog, **handleUpdateClick** za ažuriranje izmenjenog zaposlenog i **handleDeleteClick** za brisanje zaposlenog.

U **useEffect** Hook-u se izvršava funkcija **fetchZaposleni** koja dohvata podatke o zaposlenima sa servera putem HTTP GET zahteva. Nakon što se podaci uspešno dohvate, stanje **zaposleni** se ažurira sa dobijenim podacima, a stanje **loading** se postavlja na **false**.

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

import axios from 'axios';

import './Zaposleni.css';

const Zaposleni = () => {

  const [zaposleni, setZaposleni] = useState([]);

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

  const [newZaposleni, setNewZaposleni] = useState({

    name: '',

    email: '',

    password: '',

    pozicija: '',

    odeljenje: '',

    datum\_pocetka\_rada: '',

    datum\_kraja\_ugovora: '',

    plata: '',

    firma\_id: '',

  });

  const [selectedZaposleniId, setSelectedZaposleniId] = useState(null);

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setNewZaposleni({ ...newZaposleni, [name]: value });

  };

  const handleCreateClick = async () => {

    try {

      const token = sessionStorage.getItem('authToken');

      const headers = {

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

      };

      const response = await axios.post('http://127.0.0.1:8000/api/zaposleni', newZaposleni, { headers });

      // Dodajte novog zaposlenog u niz zaposlenih

      setZaposleni([...zaposleni, response.data.zaposleni]);

      // Resetujte formu nakon uspešnog kreiranja

      setNewZaposleni({

        name: '',

        email: '',

        password: '',

        pozicija: '',

        odeljenje: '',

        datum\_pocetka\_rada: '',

        datum\_kraja\_ugovora: '',

        plata: '',

        firma\_id: '',

      });

    } catch (error) {

      console.error('Greška prilikom kreiranja zaposlenog:', error);

    }

  };

  const handleEditClick = (id) => {

    // Pronađite zaposlenog u nizu na osnovu ID-ja i postavite ga u formu za ažuriranje

    const selectedZaposleni = zaposleni.find((z) => z.id === id);

    if (selectedZaposleni) {

      setNewZaposleni(selectedZaposleni);

      setSelectedZaposleniId(id);

    }

  };

  const handleUpdateClick = async () => {

    try {

      const token = sessionStorage.getItem('authToken');

      const headers = {

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

      };

      const response = await axios.put(`http://127.0.0.1:8000/api/zaposleni/${selectedZaposleniId}`, newZaposleni, { headers });

      // Ažurirajte zaposlenog u nizu sa novim podacima

      const updatedZaposleni = zaposleni.map((z) => (z.id === selectedZaposleniId ? response.data.zaposleni : z));

      setZaposleni(updatedZaposleni);

      // Resetujte formu za ažuriranje

      setNewZaposleni({

        name: '',

        email: '',

        password: '',

        pozicija: '',

        odeljenje: '',

        datum\_pocetka\_rada: '',

        datum\_kraja\_ugovora: '',

        plata: '',

        firma\_id: '',

      });

      setSelectedZaposleniId(null);

    } catch (error) {

      console.error('Greška prilikom ažuriranja zaposlenog:', error);

    }

  };

  useEffect(() => {

    const fetchZaposleni = async () => {

      try {

        const token = sessionStorage.getItem('token');

        const headers = {

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

        };

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

        setZaposleni(response.data.data);

        setLoading(false);

      } catch (error) {

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

        setLoading(false);

      }

    };

    fetchZaposleni();

  }, []);

  const handleDeleteClick = async (id) => {

    try {

      const token = sessionStorage.getItem('authToken');

      const headers = {

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

      };

      await axios.delete(`http://127.0.0.1:8000/api/zaposleni/${id}`, { headers });

      // Uklonite zaposlenog iz niza

      setZaposleni(zaposleni.filter((z) => z.id !== id));

    } catch (error) {

      console.error('Greška prilikom brisanja zaposlenog:', error);

    }

  };

  if (loading) {

    return <p>Učitavanje...</p>;

  }

  return (

    <div className="zaposleni-container">

      <h1>Zaposleni</h1>

      <table className="zaposleni-table">

        <thead>

          <tr>

            <th>ID</th>

            <th>Ime</th>

            <th>Email</th>

            <th>Pozicija</th>

            <th>Odeljenje</th>

            <th>Datum početka rada</th>

            <th>Datum kraja ugovora</th>

            <th>Plata</th>

            <th>Akcije</th>

          </tr>

        </thead>

        <tbody>

          {zaposleni.map((zaposlen) => (

            <tr key={zaposlen.id}>

              <td>{zaposlen.id}</td>

              <td>{zaposlen.name}</td>

              <td>{zaposlen.email}</td>

              <td>{zaposlen.pozicija}</td>

              <td>{zaposlen.odeljenje}</td>

              <td>{zaposlen.datum\_pocetka\_rada}</td>

              <td>{zaposlen.datum\_kraja\_ugovora}</td>

              <td>{zaposlen.plata}</td>

              <td>

                <button onClick={() => handleEditClick(zaposlen.id)}>Izmeni</button>

                <button onClick={() => handleDeleteClick(zaposlen.id)}>Obriši</button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

      <div className="zaposleni-form">

        <h2>Kreiraj/Ažuriraj zaposlenog</h2>

        <div>

          <label>Ime:</label>

          <input type="text" name="name" value={newZaposleni.name} onChange={handleInputChange} />

        </div>

        <div>

          <label>Email:</label>

          <input type="email" name="email" value={newZaposleni.email} onChange={handleInputChange} />

        </div>

        <div>

          <label>Šifra:</label>

          <input type="password" name="password" value={newZaposleni.password} onChange={handleInputChange} />

        </div>

        <div>

          <label>Pozicija:</label>

          <input type="text" name="pozicija" value={newZaposleni.pozicija} onChange={handleInputChange} />

        </div>

        <div>

          <label>Odeljenje:</label>

          <input type="text" name="odeljenje" value={newZaposleni.odeljenje} onChange={handleInputChange} />

        </div>

        <div>

          <label>Datum početka rada:</label>

          <input type="date" name="datum\_pocetka\_rada" value={newZaposleni.datum\_pocetka\_rada} onChange={handleInputChange} />

        </div>

        <div>

          <label>Datum kraja ugovora:</label>

          <input type="date" name="datum\_kraja\_ugovora" value={newZaposleni.datum\_kraja\_ugovora} onChange={handleInputChange} />

        </div>

        <div>

          <label>Plata:</label>

          <input type="number" name="plata" value={newZaposleni.plata} onChange={handleInputChange} />

        </div>

        <div>

          <label>ID Firme:</label>

          <input type="number" name="firma\_id" value={newZaposleni.firma\_id} onChange={handleInputChange} />

        </div>

        {selectedZaposleniId ? (

          <button onClick={handleUpdateClick}>Ažuriraj</button>

        ) : (

          <button onClick={handleCreateClick}>Kreiraj</button>

        )}

      </div>

    </div>

  );

};

export default Zaposleni;

# Link ka github repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-podacinainternetu_2019_0227>

# Reference

* 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