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](#_Toc158393013)

[Opis sistema 2](#_Toc158393014)

[Opis slučajeva korišćenja 2](#_Toc158393015)

[Prijava korisnika 2](#_Toc158393016)

[Kreiranje artikla 3](#_Toc158393017)

[Pretraga artikla 4](#_Toc158393018)

[Opis arihtekture aplikacije 5](#_Toc158393019)

[Back-end: Laravel 5](#_Toc158393020)

[Front-end: React 5](#_Toc158393021)

[Baza Podataka 6](#_Toc158393022)

[Opis procesa slučajeva korišćenja 7](#_Toc158393023)

[Dijagram sekvenci – prijava korisnika 7](#_Toc158393024)

[Dijagram sekvenci – kreiranje artikla 8](#_Toc158393025)

[Dijagram sekvenci – pretraga artikla 9](#_Toc158393026)

[Model podataka 10](#_Toc158393027)

[Specifikacija REST API-ja 12](#_Toc158393028)

[Opis tehnologija korišćenih u aplikaciji 23](#_Toc158393029)

[Laravel (Back-end) 23](#_Toc158393030)

[React (Front-end) 23](#_Toc158393031)

[Korisničko uputstvo 25](#_Toc158393032)

[Prikaz delova koda 26](#_Toc158393033)

[Backend 26](#_Toc158393034)

[Frontend 37](#_Toc158393035)

[Link ka github repozitorijumu 45](#_Toc158393036)

[https://github.com/elab-development/internet-tehnologije-projekat-istrazivackalaboratorija\_2020\_0248 45](#_Toc158393037)

[Reference 46](#_Toc158393038)

# Korisnički zahtev

Aplikacija koja je razvijena služi kao platforma za upravljanje i deljenje akademskih publikacija u okviru istraživačkih i obrazovnih sredina, poput univerziteta ili naučnih institucija. Glavni fokus aplikacije je olakšavanje skladištenja, preuzimanja i deljenja naučnih radova i istraživačkih materijala.

U osnovi, aplikacija nudi intuitivan interfejs za postavljanje istraživačkih dokumenata. Korisnici mogu podnositi svoje publikacije, uključujući bitne detalje kao što su naslov, autori, sažetak i ključne reči. Svaka publikacija može biti dodatno opremljena priloženim fajlovima, koji se mogu preuzeti .

Jedna od ključnih funkcionalnosti aplikacije je mogućnost pretraživanja publikacija. Korisnici mogu pretraživati sadržaj na osnovu različitih parametara, uključujući naslov, autore, sažetak i ključne reči, što omogućava lako pronalaženje specifičnih istraživačkih radova. Takođe, aplikacija omogućava sortiranje publikacija po abecednom redu, što doprinosi boljoj organizaciji i lakšem snalaženju.

Administratori sajta imaju mogućnost odobravanja publikacija koje su istraživači postavili, kao i mogućnost pregleda raznih statistika vezanih za publikacije.

# Opis sistema

U ovom poglavlju ćemo prikazati tri slučaja korišćenja.

## 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 artikla

Slučaj korišćenja: Kreiranje artikla

Akteri:

* Autor (istraživač)
* Sistem (program)

Preduslov:

* Sistem je uključen, a autor ima pristup funkcionalnostima za kreiranje novih artikala.

Osnovni scenario:

1. Autor bira opciju za kreiranje novog artikla unutar sistema.
2. Sistem prikazuje formu za unos podataka o artiklu, uključujući naslov, sadržaj, apstrakt, ključne reči, datoteku sa sadržajem, i druge relevantne informacije.
3. Autor unosi tražene informacije o artiklu u formu.
4. Autor pregleda unete podatke i proverava da li su sve informacije ispravne i potpune.
5. Autor potvrđuje unos klikom na dugme "Kreiraj artikal" ili slično.
6. Sistem proverava unete podatke na validnost i potpunost.
7. Ukoliko su svi podaci ispravni, sistem kreira novi artikal i registruje ga u bazi.
8. Sistem obaveštava autora porukom: "Novi artikal je uspešno kreiran."

### Pretraga artikla

Slučaj korišćenja: Pretraga artikala

Akteri:

* Korisnik sistema (istraživač, student, posetilac)
* Sistem (program)

Preduslovi:

* Sistem je uključen i omogućen je pristup funkcionalnostima za pretragu artikala.

Osnovni scenario:

1. Korisnik unosi ključne reči ili druge parametre pretrage u odgovarajuće polje na interfejsu pretrage.
2. Korisnik pokreće pretragu klikom na dugme "Pretraži" ili slično.
3. Sistem vrši pretragu baze podataka koristeći unete parametre.
4. Sistem prikazuje rezultate pretrage korisniku u obliku liste artikala koji odgovaraju unetim kriterijumima.
5. Korisnik pregleda rezultate pretrage i bira artikal koji želi detaljnije da istraži.
6. Sistem prikazuje detaljne informacije o izabranom artiklu, uključujući naslov, autora, sadržaj, ključne reči, datum objavljivanja, i druge relevantne informacije.
7. Korisnik ima mogućnost da pročita sadržaj artikla, pregleda reference, ili preuzme datoteku sa sadržajem ako je dostupna.
8. Korisnik može da se vrati na prethodni korak pretrage ili da izvrši novu pretragu ako je potrebno.

## Opis arihtekture aplikacije

U ovom poglavlju, detaljno će biti opisana arhitektura aplikacije koja je razvijena koristeći Laravel kao back-end i React kao front-end tehnologiju. Ova moderna web aplikacija kombinuje moć Laravela, PHP frameworka za izradu robustnih web aplikacija, sa fleksibilnošću Reacta, JavaScript biblioteke za izgradnju korisničkih interfejsa, pružajući korisnicima visokokvalitetno i interaktivno korisničko iskustvo.

### Back-end: Laravel

Laravel je izabran za back-end segment aplikacije zbog svoje visoke performanse, sigurnosti i lakoće za upotrebu. Framework donosi bogat set funkcionalnosti, uključujući ali ne ograničavajući se na autentifikaciju korisnika, autorizaciju, upravljanje sesijama, rutiranje, kao i rad sa bazama podataka kroz ORM (Object-Relational Mapping) sistem nazvan Eloquent.

* **Struktura projekta**: Laravel aplikacija je strukturirana na način koji promoviše jasno razdvajanje logike i prezentacije, uz MVC (Model-View-Controller) arhitekturu kao osnovu. Modeli se koriste za interakciju sa bazom podataka; kontroleri obrađuju zahteve korisnika i vraćaju odgovore; dok su view-ovi zamijenjeni React komponentama na front-endu.
* **API**: Za komunikaciju sa React front-endom, Laravel je korišćen za izgradnju RESTful API-ja. Ovo omogućava aplikaciji da obradi CRUD (Create, Read, Update, Delete) operacije preko HTTP zahteva, pružajući JSON odgovore.
* **Bezbednost**: Laravel nudi integrisane mehanizme za zaštitu od uobičajenih web ranjivosti, uključujući XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) zaštite i SQL injekcija, osiguravajući da je aplikacija sigurna od napada.

### Front-end: React

React je korišćen za razvoj klijentskog dela aplikacije zbog svoje efikasnosti, komponentno orijentisane arhitekture i sposobnosti za izgradnju bogatih, interaktivnih korisničkih interfejsa.

* **Komponente**: Aplikacija je podeljena na ponovno upotrebljive komponente, što omogućava lakše održavanje i razvoj aplikacije. Svaka komponenta ima svoj vlastiti set stanja i propertija, omogućavajući modularan i efikasan način razvoja.
* **Stanje Aplikacije**: Za upravljanje stanjem aplikacije koristi se kontekst API ili biblioteke poput Reduxa, omogućavajući komponentama da dele i upravljaju globalnim stanjem bez potrebe za prosleđivanjem propertija kroz više nivoa.
* **Komunikacija sa Back-endom**: Axios je korišćen za asinhrono komuniciranje sa Laravel API-jem, omogućavajući aplikaciji da efikasno šalje zahteve i prima odgovore od servera. React Query se može koristiti za fetching, caching i sinhronizaciju podataka sa back-endom, poboljšavajući performanse i korisničko iskustvo.

### Baza Podataka

Baza podataka je integralni deo aplikacije, koristeći MySQL za skladištenje podataka. Laravelova migracija i seeder funkcionalnosti omogućavaju jednostavno verzioniranje baze podataka i populisanje testnim podacima, što olakšava razvoj i testiranje.

## Opis procesa slučajeva korišćenja

### Dijagram sekvenci – prijava korisnika

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

Opis je automatski generisanSlika 1 ds - prijava korisnika

### Dijagram sekvenci – kreiranje artikla

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

Opis je automatski generisan

Slika 2 ds -kreiranje artikla

### Dijagram sekvenci – pretraga artikla

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

Opis je automatski generisan

Slika 3 ds -pretraga artikla

## Model podataka

Aplikacija koristi niz modela koji čine osnovu sistema za upravljanje naučnim sadržajem. Ključni modeli uključuju:

1. Model Article: Predstavlja srce aplikacije. Svaki članak je povezan sa korisnikom (autorom) i može sadržavati višestruke komentare i tagove. Model čuva osnovne informacije o članku, kao što su naslov, sadržaj i datum objavljivanja.
2. Model User: Ovaj model upravlja registracijom korisnika, njihovim profilima i autentifikacijom. Korisnici mogu biti autori članaka i ostavljati komentare.
3. Model Comment: Omogućava korisnicima da komentarišu članke. Svaki komentar je povezan sa specifičnim člankom i korisnikom
4. Model Tag: Tagovi služe za kategorizaciju članaka. Jedan članak može imati više tagova, što olakšava pretragu i organizaciju sadržaja.
5. Model Role: Ovaj model definiše različite uloge korisnika u sistemu, kao što su administrator, autor ili običan korisnik, pružajući različite nivoe pristupa i dozvola.

Ovi modeli su međusobno povezani i formiraju osnovu za interakciju korisnika sa aplikacijom, omogućavajući efikasno upravljanje, pretragu i interakciju sa naučnim sadržajem.

Slika na kojoj se nalazi tekst, dijagram, Plan, Tehnički crtež

Opis je automatski generisan

Slika 4 ERD

## Specifikacija REST API-ja

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih artikala |
| HTTP metoda | GET |
| URL | /api/ articles |
| URL parametri | ?page=2 |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "current\_page": 2,      "data": [          {              "id": 4,              "user\_id": 14,              "title": "Modi ea eos et ut magnam placeat dolore.",              "content": "Et voluptatem aut deleniti. Officia nostrum dolores expedita non. Iusto tempora explicabo aperiam ipsa.",              "image\_path": "example.jpg",              "published\_at": "2023-12-29T07:52:45.000000Z",              "deleted\_at": **null**,              "created\_at": "2024-01-11T23:08:39.000000Z",              "updated\_at": "2024-01-11T23:08:39.000000Z"          },          {              "id": 5,              "user\_id": 15,              "title": "Non illum facilis at.",              "content": "Aut aliquid consequatur aliquam ex quaerat sunt. Qui aut reprehenderit laborum consequuntur voluptas. Ducimus voluptatum facilis et magni.",              "image\_path": "example.jpg",              "published\_at": "2023-12-14T08:17:12.000000Z",              "deleted\_at": **null**,              "created\_at": "2024-01-11T23:08:39.000000Z",              "updated\_at": "2024-01-11T23:08:39.000000Z"          },          {              "id": 6,              "user\_id": 16,              "title": "Nulla velit sed voluptatum corporis consequatur eaque.",              "content": "Reiciendis sed dignissimos placeat provident molestiae tempora qui. Laboriosam harum architecto aut ut sit alias. Atque ab ut dolore blanditiis maxime cupiditate.",              "image\_path": "example.jpg",              "published\_at": "2024-01-03T17:36:23.000000Z",              "deleted\_at": **null**,              "created\_at": "2024-01-11T23:08:39.000000Z",              "updated\_at": "2024-01-11T23:08:39.000000Z"          }      ],      "first\_page\_url": "http://127.0.0.1:8000/api/articles?page=1",      "from": 4,      "last\_page": 7,      "last\_page\_url": "http://127.0.0.1:8000/api/articles?page=7",      "links": [          {              "url": "http://127.0.0.1:8000/api/articles?page=1",              "label": "&laquo; Previous",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=1",              "label": "1",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=2",              "label": "2",              "active": **true**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=3",              "label": "3",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=4",              "label": "4",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=5",              "label": "5",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=6",              "label": "6",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=7",              "label": "7",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/articles?page=3",              "label": "Next &raquo;",              "active": **false**          }      ],      "next\_page\_url": "http://127.0.0.1:8000/api/articles?page=3",      "path": "http://127.0.0.1:8000/api/articles",      "per\_page": 3,      "prev\_page\_url": "http://127.0.0.1:8000/api/articles?page=1",      "to": 6,      "total": 20  } |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje artikla |
| HTTP metoda | POST |
| URL | /api/ articles |
| URL parametri | ?title=artikal 1&content=sadryaj artikla 1&user\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "title": "artikal 1",      "content": "sadryaj artikla 1",      "user\_id": "1",      "updated\_at": "2024-01-12T02:56:26.000000Z",      "created\_at": "2024-01-12T02:56:26.000000Z",      "id": 21  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje artikla |
| HTTP metoda | PUT |
| URL | /api/ articles |
| URL parametri | 21?title=artikal 1 novi naslov&content=sadryaj artikla 1&user\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 21,      "user\_id": "1",      "title": "artikal 1 novi naslov",      "content": "sadryaj artikla 1",      "image\_path": **null**,      "published\_at": **null**,      "deleted\_at": **null**,      "created\_at": "2024-01-12T02:56:26.000000Z",      "updated\_at": "2024-01-12T02:57:44.000000Z"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje artikla |
| HTTP metoda | DELETE |
| URL | /api/ articles |
| URL parametri | 21 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Article deleted"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih naucnih dogadjaja |
| HTTP metoda | GET |
| URL | /api/fetchScienceEvents |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | "\_embedded": {          "events": [              {                  "name": "Rude Science",                  "type": "event",                  "id": "G5dbZ9VJN6wgD",                  "test": **false**,                  "url": "https://www.ticketmaster.co.uk/rude-science-scunthorpe-17-02-2024/event/36005F49A31F2D7B",                  "locale": "en-us",                  "images": [                      {                          "ratio": "3\_2",                          "url": "https://s1.ticketm.net/dam/c/2b5/5a0a3487-492b-4072-9d8d-1fcda02bd2b5\_781101\_ARTIST\_PAGE\_3\_2.jpg",                          "width": 305,                          "height": 203,                          "fallback": **true**                      },                      {                          "ratio": "4\_3",                          "url": "https://s1.ticketm.net/dam/c/2b5/5a0a3487-492b-4072-9d8d-1fcda02bd2b5\_781101\_CUSTOM.jpg",                          "width": 305,                          "height": 225,                          "fallback": **true**                      },                      {                          "ratio": "16\_9",                          "url": "https://s1.ticketm.net/dam/c/2b5/5a0a3487-492b-4072-9d8d-1fcda02bd2b5\_781101\_RETINA\_PORTRAIT\_16\_9.jpg",                          "width": 640,                          "height": 360,                          "fallback": **true**                      },                      {                          "ratio": "16\_9",                          "url": "https://s1.ticketm.net/dam/c/2b5/5a0a3487-492b-4072-9d8d-1fcda02bd2b5\_781101\_TABLET\_LANDSCAPE\_LARGE\_16\_9.jpg",                          "width": 2048,                          "height": 1152,                          "fallback": **true**                      },                      {                          "ratio": "3\_2",                          "url": "https://s1.ticketm.net/dam/c/2b5/5a0a3487-492b-4072-9d8d-1fcda02bd2b5\_781101\_RETINA\_PORTRAIT\_3\_2.jpg",                          "width": 640,                          "height": 427,                          "fallback": **true**                      },                      {                          "ratio": "16\_9",                          "url": "https://s1.ticketm.net/dam/c/2b5/5a0a3487-492b-4072-9d8d-1fcda02bd2b5\_781101\_RECOMENDATION\_16\_9.jpg",                          "width": 100,                          "height": 56,                          "fallback": **true**                      },                      {  … |
| Format izlaznih parametara | Application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih komentara |
| HTTP metoda | GET |
| URL | /api/comments |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "current\_page": 1,      "data": [          {              "id": 1,              "user\_id": 21,              "article\_id": 11,              "content": "Corporis aut animi et officiis ipsa esse. Quibusdam facere placeat magni illum animi tempore. Vel impedit voluptas at voluptatum ab.",              "status": "published",              "created\_at": "2024-01-11T23:08:40.000000Z",              "updated\_at": "2024-01-11T23:08:40.000000Z"          },          {              "id": 2,              "user\_id": 23,              "article\_id": 12,              "content": "Molestias magni eos cum atque aspernatur libero consequuntur. Iusto est dignissimos perferendis ad quae maiores officiis. Maxime ab magnam voluptas sint earum laudantium tempore. Eum in qui maxime fugiat laborum.",              "status": "published",              "created\_at": "2024-01-11T23:08:40.000000Z",              "updated\_at": "2024-01-11T23:08:40.000000Z"          }      ],      "first\_page\_url": "http://127.0.0.1:8000/api/comments?page=1",      "from": 1,      "last\_page": 5,      "last\_page\_url": "http://127.0.0.1:8000/api/comments?page=5",      "links": [          {              "url": **null**,              "label": "&laquo; Previous",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/comments?page=1",              "label": "1",              "active": **true**          },          {              "url": "http://127.0.0.1:8000/api/comments?page=2",              "label": "2",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/comments?page=3",              "label": "3",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/comments?page=4",              "label": "4",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/comments?page=5",              "label": "5",              "active": **false**          },          {              "url": "http://127.0.0.1:8000/api/comments?page=2",              "label": "Next &raquo;",              "active": **false**          }      ],      "next\_page\_url": "http://127.0.0.1:8000/api/comments?page=2",      "path": "http://127.0.0.1:8000/api/comments",      "per\_page": 2,      "prev\_page\_url": **null**,      "to": 2,      "total": 10  } |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje komentara |
| HTTP metoda | POST |
| URL | /api/comments |
| URL parametri | ? user\_id=1&content=sadrzaj prvog komentara&article\_id=1&status=status |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "content": "sadrzaj prvog komentara",      "user\_id": "1",      "article\_id": "1",      "status": "status",      "updated\_at": "2024-01-12T03:03:48.000000Z",      "created\_at": "2024-01-12T03:03:48.000000Z",      "id": 11  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje komentara |
| HTTP metoda | PUT |
| URL | /api/comments |
| URL parametri | 11?user\_id=1&content=sadrzaj prvog komentara&article\_id=1&status=status |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "id": 11,      "user\_id": "1",      "article\_id": "1",      "content": "sadrzaj prvog komentara",      "status": "status",      "created\_at": "2024-01-12T03:03:48.000000Z",      "updated\_at": "2024-01-12T03:03:48.000000Z"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje komentara |
| HTTP metoda | DELETE |
| URL | /api/comments |
| URL parametri | 11 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Comment deleted"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | name=Milica&email=milica@gmail.com&password=milica123 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "user": {          "name": "Milica",          "email": "milica@gmail.com",          "role\_id": 1,          "updated\_at": "2024-01-12T02:53:16.000000Z",          "created\_at": "2024-01-12T02:53:16.000000Z",          "id": 41      },      "token": "1|XUJ9I5StVzuAkvhWS7rbF52ao3QnwDnXTVybv3tzf2619251"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | email=milica@gmail.com&password=milica123 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "user": {          "id": 41,          "name": "Milica",          "email": "milica@gmail.com",          "email\_verified\_at": **null**,          "profile\_photo": **null**,          "created\_at": "2024-01-12T02:53:16.000000Z",          "updated\_at": "2024-01-12T02:53:16.000000Z",          "role\_id": 1,          "bio": **null**,          "last\_login\_at": **null**      },      "token": "2|7sxTE4myg54HbGZarWSAToYO4oGu2Q9AOnM2rqpq938b8ed1"  } |
| 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

Razvoj naše web aplikacije podrazumeva upotrebu savremenih tehnologija koje omogućavaju efikasan razvoj, visoke performanse i odlično korisničko iskustvo. U ovom poglavlju, detaljno će biti opisane ključne tehnologije koje su korišćene u izradi aplikacije, uključujući Laravel za back-end, React za front-end, kao i dodatne alate i biblioteke koje doprinose funkcionalnosti i efikasnosti aplikacije.

## Laravel (Back-end)

Laravel je PHP web framework visokog kvaliteta koji pruža elegantan nacin za razvoj web aplikacija. Poznat je po svojoj robusnosti, sigurnosti i nizu karakteristika koje ubrzavaju razvojni proces:

* **MVC Arhitektura**: Laravel koristi Model-View-Controller arhitekturu, omogućavajući jasno razdvajanje logike aplikacije od korisničkog interfejsa.
* **Eloquent ORM**: Integrirani Object-Relational Mapping alat olakšava interakciju sa bazama podataka kroz elegantan i intuitivan API.
* **Migracije i Seederi**: Sistem za verzionisanje baze podataka omogućava lako upravljanje strukturom baze i testnim podacima.
* **Autentifikacija i Autorizacija**: Ugrađeni sistemi za kontrolu pristupa i identifikaciju korisnika pružaju visok nivo bezbednosti.
* **Artisan CLI**: Komandna linija alata koja omogućava automatizaciju razvojnih zadataka i povećava produktivnost.

## React (Front-end)

React je JavaScript biblioteka za izgradnju korisničkih interfejsa, koju razvija Facebook. Omogućava razvoj interaktivnih UI-ova kroz komponentno baziran pristup:

* **Komponente**: React komponente omogućavaju izolaciju i ponovnu upotrebu koda, što olakšava održavanje i razvoj aplikacije.
* **Virtual DOM**: React koristi Virtual DOM za optimizaciju promena u DOM-u, što rezultira brzim ažuriranjem korisničkog interfejsa.
* **Hooks**: Hooks omogućavaju korišćenje stanja i drugih React karakteristika bez pisanja klasa, čineći kod jednostavnijim i čistijim.
* **Jednosmerna protok podataka**: Promoviše predvidljivo ponašanje aplikacije i olakšava razumevanje kako se podaci prenose kroz aplikaciju.

Dodatne Tehnologije i Alati

* **Axios**: Korišćen za upravljanje HTTP zahtevima iz Reacta, Axios olakšava komunikaciju sa back-endom i obradu asinhronih operacija.
* **MySQL**: Za skladištenje podataka koristi se relacijska baza podataka MySQL
* **Bootstrap**: Za stilizovanje aplikacije koriste se popularni CSS frameworki kao što su Bootstrap, koji pružaju bogat set komponenti i stilova.
* **Git**: Za verzionisanje koda i kolaboraciju među razvojnim timom koristi se Git, što omogućava efikasno upravljanje izmenama koda i projektom.

# Korisničko uputstvo

Priliko pokretanja aplikacije korisnicima se otvara početna stranica koja izgleda kao na slici ispod.

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

Opis je automatski generisan

Slika 5 početna

Korisnici imaju mogućnost prijave ili registracije, kao na slici ispod.

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

Opis je automatski generisan

Slika 6 prijava

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

Opis je automatski generisan

Slika 7 registracija

Kada se korisnik prijavi, njemu se otvara stranica na kojoj može da unese svoje publikacije.

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

Opis je automatski generisan

Slika 8 unos publikacije

Ulogovani korisnik ima mogućnost pregleda publikacija koje je on uneo. Ovo je prikazano na slici ispod.

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

Opis je automatski generisan

Slika 9 pregled publikacija

Kada se u aplikaciju prijavi administrator, on ima mogućnost pregleda raznik statistika, kao na stranici ispod.

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

Opis je automatski generisan

Slika 10 admin stranica

Aministrator ima mogućnost pregleda svih publikacija, kreiranih od strane svakog korisnika, kao I mogućnost odobravanja određenih publikacija.

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

Opis je automatski generisan

Slika 11 odobravanje publikacija

# Prikaz delova koda

## Backend

U ovom radu su kreirani sledeci kontroleri

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 12 prikaz kreiranih kontrolera

U ovom AuthControlleru, imamo tri glavne metode za rad sa korisnicima: register, login i logout.

* Register: Ova metoda prima zahtev za registraciju korisnika. zahtev mora sadrzati korisnikovo ime, email, lozinku, datum rođenja i biografiju. Metoda koristi validaciju da bude sigurna da su svi potrebni podaci uneti i da su validni. Ako validacija ne prodje, vraca se odgovor sa greškama. Ako validacija prodje, korisnik se kreira u bazi podataka sa unetim podacima.
* Login: Ova metoda prima zahtev za prijavu korisnika koji mora sadrzati email i lozinku. Metoda koristi validaciju da bude sigurna da su svi potrebni podaci uneti i da su validni. Ako validacija ne prodje, vraca se odgovor sa greškama. Ako validacija prodje, korisnik se prijavljuje u aplikaciju. Ako prijava ne uspe, vraca se odgovor sa greškom. Ako prijava uspe, kreira se token za autentifikaciju i vraca se zajedno sa korisnikovim podacima.
* Logout: Ova metoda brise token za autentifikaciju trenutno prijavljenog korisnika. Nakon brisanja tokena, korisnik se automatski odjavljuje iz aplikacije.

<?php

namespace App\Http\Controllers;

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(), [

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

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

            'password' => 'required|string|min:8|confirmed',

            'date\_of\_birth' => 'required|date',

            'bio' => 'required|string'

        ]);

        if ($validator->fails()) {

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

        }

        $user = User::create([

            'name' => $request->name,

            'email' => $request->email,

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

            'date\_of\_birth' => $request->date\_of\_birth,

            'bio' => $request->bio

        ]);

        return response()->json(['message' => 'User successfully registered', 'user' => $user], 201);

    }

    public function login(Request $request)

    {

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

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

            'password' => 'required|string|min:8'

        ]);

        if ($validator->fails()) {

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

        }

        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(['message' => 'Login successful', 'token' => $token,'user'=> $user]);

    }

    public function logout(Request $request)

    {

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

        return response()->json(['message' => 'Logout successful']);

    }

    public function changePassword(Request $request)

    {

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

            'current\_password' => 'required|string|min:8',

            'new\_password' => 'required|string|min:8|confirmed',

        ]);

        if ($validator->fails()) {

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

        }

        if (!Hash::check($request->current\_password, $request->user()->password)) {

            return response()->json(['message' => 'The current password is incorrect.'], 403);

        }

        $request->user()->update([

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

        ]);

        return response()->json(['message' => 'Password successfully changed.']);

    }

}

Dodatno, u ovom AuthControlleru imamo i metodu za promenu lozinke. Ova metoda prima zahtev za promenu lozinke.

Kontroler PasswordResetController ima dve funkcije koje omogućavaju korisnicima da resetuju svoju lozinku. Prva funkcija sendResetLinkEmail služi za slanje linka za resetovanje lozinke na email koji korisnik unese. Ova funkcija prvo validira email korisnika, a zatim koristi Laravelovu funkciju Password::sendResetLink za slanje linka. U zavisnosti od rezultata, vraća odgovarajući JSON odgovor sa porukom uspeha ili greške. Druga funkcija reset omogućava korisniku da postavi novu lozinku koristeći token dobijen putem emaila. Nakon validacije ulaznih podataka, koristi se Password::reset za promenu lozinke. Ako je resetovanje uspešno, korisnikov remember\_token se ažurira, a lozinka se čuva u hashiranom obliku u bazi podataka, nakon čega se vraća odgovor uspeha ili greške.

<?php

namespace App\Http\Controllers;

use App\Models\User;

use Carbon\Carbon;

use Illuminate\Auth\Events\PasswordReset;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Password;

use Illuminate\Support\Str;

class PasswordResetController extends Controller

{

    public function sendResetLinkEmail(Request $request)

    {

      $request->validate(['email' => 'required|email']);

        $status = Password::sendResetLink(

            $request->only('email')

        );

        return $status === Password::RESET\_LINK\_SENT

                    ? response()->json(['message' => \_\_($status)])

                    : response()->json(['email' => \_\_($status)], 422);

    }

    public function reset(Request $request)

    {

        $request->validate([

            'token' => 'required',

            'email' => 'required|email',

            'password' => 'required|min:8|confirmed',

        ]);

        $status = Password::reset(

            $request->only('email', 'password', 'password\_confirmation', 'token'),

            function ($user, $password) {

                $user->forceFill([

                    'password' => Hash::make($password)

                ])->setRememberToken(Str::random(60));

                $user->save();

                event(new PasswordReset($user));

            }

        );

        return $status === Password::PASSWORD\_RESET

                    ? response()->json(['message' => \_\_($status)])

                    : response()->json(['email' => \_\_($status)], 422);

    }

}

DocumentController sadrži nekoliko metoda za upravljanje dokumentima. Metoda index vraća sve dokumente iz baze podataka. show vraća određeni dokument na osnovu ID-a. store služi za kreiranje novog dokumenta, gde se prvo vrši validacija podataka, a zatim čuva dokument u bazi sa opcionalnim uploadom fajla. Metoda update omogućava ažuriranje postojećeg dokumenta, uključujući i zamenu fajla ukoliko je novi fajl poslat. destroy briše dokument, uz proveru da li je dokument već soft-deleted. search omogućava filtriranje dokumenata po različitim kriterijumima kao što su naslov, sadržaj, autor, kategorija i tagovi. download omogućava preuzimanje fajla dokumenta, uz proveru da li fajl postoji. Takođe, postoji pomoćna funkcija getMimeType koja vraća MIME tip fajla na osnovu njegove ekstenzije.

<?php

namespace App\Http\Controllers;

use App\Models\Document;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Storage;

use Illuminate\Support\Facades\Validator;

class DocumentController extends Controller

{

    public function index()

    {

        $documents = Document::all();

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

    }

    public function show($id)

    {

        $document = Document::findOrFail($id);

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

    }

    public function store(Request $request)

    {

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

            'title' => 'required|max:255',

            'content' => 'required',

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

            'category\_id' => 'required|exists:categories,id',

            'tags' => 'required|array',      //Ovo pravilo zahteva da polje 'tags' bude prisutno u zahtevu (ne smije biti prazno ili null) i da bude niz (array).

            'tags.\*' => 'distinct|exists:tags,id',  //Ovo pravilo se primenjuje na svaki element niza 'tags'.  //distinct' proverava da svaki tag u nizu bude jedinstven, tj. da se ne ponavlja više puta.

            'file' => 'required|file', // Pravilo za validaciju uploadovanog fajla

         //   'is\_public' => 'required',

            'downloads' => 'required|numeric',

        ]);

        if ($validator->fails()) {

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

        }

        $documentData = $validator->validated();

        // Obrada uploada fajla

        if ($request->hasFile('file')) {

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

            $filePath = $file->store('documents', 'public'); // Sačuvaj fajl i vrati putanju

            $documentData['file\_path'] = $filePath;

        }

        $documentData['is\_public'] = 0;

        // Kreiraj dokument i postavi kategoriju

        $document = Document::create($documentData);

        // Kako su tagovi predstavljeni kao niz, treba ih konvertovati u string pre snimanja

       // $document->tags()->sync($request->tags);   //Ova linija koda se koristi za uspostavljanje veze many-to-many između Document i Tag. Metoda sync prima niz ID-ova tagova i ažurira odgovarajuću pivot tabelu, uklanjajući sve postojeće veze i dodajući nove. S obzirom na vašu strukturu, ovo je ispravno za rad sa tagovima.

        return response()->json($document, 201);

    }

    public function update(Request $request, $id)

    {

        $document = Document::findOrFail($id);

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

            'title' => 'sometimes|required|max:255',

            'content' => 'sometimes|required',

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

            'category\_id' => 'sometimes|required|exists:categories,id',

            'tags' => 'sometimes|required|array',

            'tags.\*' => 'distinct|exists:tags,id',

            'file' => 'sometimes|file',

            'is\_public' => 'sometimes|boolean',

            'downloads' => 'sometimes|numeric',

        ]);

        if ($validator->fails()) {

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

        }

        $documentData = $validator->validated();

        // Obrada uploada fajla

        if ($request->hasFile('file')) {

            // Pre nego što sačuvamo novi fajl, trebalo bi da obrisemo stari

            Storage::delete($document->file\_path);

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

            $filePath = $file->store('documents', 'public'); // Sačuvaj fajl i vrati putanju

            $documentData['file\_path'] = $filePath;

        }

        // Konvertovanje niza tagova u string pre snimanja, ako su tagovi prosleđeni

        if (isset($documentData['tags'])) {

            $documentData['tags'] = implode(',', $request->tags);

        }

        $document->update($documentData);

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

    }

    public function destroy($id)  //posto nas model koristi softDeletes, onda metoda destroy mora da bude implementirana na malo drugaciji nacin

    {

        $document = Document::withTrashed()->findOrFail($id);

        if ($document->trashed()) {

            //  dokument je već obrisan

            return response()->json(['message' => 'Document has already been deleted.'], 404);

        } else {

            $document->delete();

            return response()->json(null, 204);

        }

    }

    public function search(Request $request)

{

    // Počinjemo sa kreiranjem osnovnog upita

    $query = Document::query();

    // Filtriramo po naslovu ako je prosleđen kao parametar

    if ($request->has('title')) {

        $query->where('title', 'like', '%' . $request->title . '%');

    }

    // Filtriramo po sadržaju ako je prosleđen kao parametar

    if ($request->has('content')) {

        $query->where('content', 'like', '%' . $request->content . '%');

    }

    // Filtriramo po autoru ako je prosleđen kao parametar

    if ($request->has('author\_id')) {

        $query->where('author\_id', $request->author\_id);

    }

    // Filtriramo po kategoriji ako je prosleđen kao parametar

    if ($request->has('category\_id')) {

        $query->where('category\_id', $request->category\_id);

    }

    // Filtriramo po tagovima ako su prosleđeni kao parametar

    if ($request->has('tags')) {

        $tags = $request->tags;

        $query->whereHas('tags', function ($query) use ($tags) {

            $query->whereIn('id', $tags);

        });

    }

    // Filtriramo po dostupnosti ako je prosleđen kao parametar

    if ($request->has('is\_public')) {

        $query->where('is\_public', $request->is\_public);

    }

    // Dobijamo filtrirane dokumente

    $documents = $query->get();

    // Vraćamo dokumente kao JSON odgovor

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

    }

    public function download($id)

    {

        $document = Document::findOrFail($id);

        $filePath = storage\_path('app/public/' . $document->file\_path);

        if (!file\_exists($filePath)) {

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

        }

        $fileName = basename($filePath);

        $fileExtension = pathinfo($filePath, PATHINFO\_EXTENSION);

        // Određivanje MIME tipa na osnovu ekstenzije fajla

        $mimeType = $this->getMimeType($fileExtension);

        // Preuzimanje fajla sa ispravnim MIME tipom

        return response()->download($filePath, $fileName, [

            'Content-Type' => $mimeType,

            'Content-Disposition' => 'attachment; filename="' . $fileName . '"'

        ]);

    }

    // Pomoćna funkcija za dobijanje MIME tipa

    private function getMimeType($extension)

    {

        $mimeTypes = [

            'pdf' => 'application/pdf',

            'jpg' => 'image/jpeg',

            'jpeg' => 'image/jpeg',

            'png' => 'image/png',

            'txt' => 'text/plain',

        ];

        return $mimeTypes[$extension] ?? 'application/octet-stream';

    }

}

Prva ruta je GET /api/documents/search, koja koristi DocumentController za pretragu dokumenata po kriterijumima kao što su naslov, kategorija i oznake. Druga ruta, GET /api/documents/{document}/download, omogućava preuzimanje dokumenta i takođe koristi DocumentController.

Za resetovanje lozinke postoje dve rute: POST /api/forgot-password i POST /api/reset-password, koje koriste PasswordResetController za slanje linka za resetovanje lozinke i sam proces resetovanja lozinke.

Takođe, postoje rute za autentifikaciju korisnika: POST /api/register i POST /api/login koje koriste AuthController za registraciju novih korisnika i prijavljivanje postojećih korisnika.

Unutar grupe koja zahteva autentifikaciju korisnika (auth:sanctum), nalaze se dodatne rute. Postoji ruta POST /api/logout za odjavljivanje korisnika, kao i više ruta za upravljanje dokumentima, komentarima, oznakama i kategorijama kroz odgovarajuće kontrolere (DocumentController, CommentController, TagController, CategoryController). Na kraju, postoji ruta POST /api/changePassword za promenu korisničke lozinke. Ove rute omogućavaju kreiranje, čitanje, ažuriranje i brisanje (CRUD operacije) nad pomenutim resursima.

<?php

use App\Http\Controllers\AuthController;

use App\Http\Controllers\CategoryController;

use App\Http\Controllers\CommentController;

use App\Http\Controllers\DocumentController;

use App\Http\Controllers\PasswordResetController;

use App\Http\Controllers\TagController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| API Routes

|--------------------------------------------------------------------------

|

| Here is where you can register API routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| is assigned the "api" middleware group. Enjoy building your API!

|

\*/

Route::get('/documents/search', [DocumentController::class, 'search']);  //GET /api/documents/search?title=izveštaj&category\_id=3&tags[]=1&tags[]=5&is\_public=1

Route::get('/documents/{document}/download', [DocumentController::class, 'download']);

Route::post('/forgot-password', [PasswordResetController::class, 'sendResetLinkEmail']);

Route::post('/reset-password', [PasswordResetController::class, 'reset']);

Route::post('/register', [AuthController::class, 'register']);

Route::post('/login', [AuthController::class, 'login']);

Route::middleware('auth:sanctum')->group(function () {

    Route::post('/logout', [AuthController::class, 'logout']);

    Route::apiResource('documents', DocumentController::class);

        Route::apiResource('comments', CommentController::class);

        Route::apiResource('tags', TagController::class);

        Route::apiResource('categories', CategoryController::class);

    Route::post('/changePassword', [AuthController::class, 'changePassword']);

});

## Frontend

Kod ispod predstavlja komponentu "HomePage" u React aplikaciji.

Prvo, uvezene su potrebne biblioteke i komponente, uključujući React, useState i useEffect za upravljanje stanjem i efektima, axios za izvršavanje HTTP zahtjeva, te različite ikone iz biblioteke "react-icons" za prikazivanje ikona.

Zatim se definira funkcionalna komponenta "HomePage". U ovoj komponenti postoje delovi koda koji su namenjeni za prikazivanje i paginaciju "zanimljivih činjenica" koje su preuzete sa spoljnog API-ja.

Nakon toga, u samom HTML delu komponente, prikazuju se informacije o laboratoriji, ikone koje predstavljaju različite naučne oblasti, i nekoliko sekcija sa tekstom o značaju nauke i zanimljivim činjenicama.

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

import axios from 'axios';

import { FaMicroscope, FaFlask, FaAtom, FaBrain, FaLaughBeam, FaRegLaughBeam } from 'react-icons/fa';

import FunFact from './FunFact';

const HomePage = () => {

    // const [facts, setFacts] = useState([]);

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

    // const [factsPerPage] = useState(10);

    // useEffect(() => {

    //   const fetchFacts = async () => {

    //     try {

    //       const response = await axios.get('https://api.api-ninjas.com/v1/facts?limit=50', {

    //         headers: { 'X-Api-Key': '7xiJG3ZG/DVXBFQcpnUANw==DCKsOuWEdluVhptV' }

    //       });

    //       setFacts(response.data);

    //     } catch (error) {

    //       console.error('Error fetching facts:', error);

    //     }

    //   };

    //   fetchFacts();

    // }, []);

    // const lastFactIndex = currentPage \* factsPerPage;

    // const firstFactIndex = lastFactIndex - factsPerPage;

    // const currentFacts = facts.slice(firstFactIndex, lastFactIndex);

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

    // // Calculate the total number of pages

    // const pageCount = Math.ceil(facts.length / factsPerPage);

    // // Render pagination dots

    // const PaginationDots = () => {

    //   let dots = [];

    //   for (let i = 1; i < pageCount; i++) {

    //     dots.push(

    //       <span

    //         key={i}

    //         className={`dot ${i === currentPage ? 'active' : ''}`}

    //         onClick={() => paginate(i)}

    //       />

    //     );

    //   }

    //   return <div className="pagination">{dots}</div>;

    // };

  return (

    <div className="homepage">

     <h1><FaAtom /> Dobrodošli u našu naučno-istraživačku laboratoriju</h1>

      <p>

        Otkrijte svijet mikroskopski malih čuda i naprednih istraživanja. Naš tim

        stručnjaka je posvećen razumijevanju temelja znanosti kako bi

        unaprijedili medicinu, tehnologiju i naše razumijevanje svemira.

      </p>

      <div className="icons">

        <div><FaMicroscope /> Mikroskopija</div>

        <div><FaFlask /> Hemija</div>

        <div><FaAtom /> Fizika</div>

        <div><FaBrain /> Neuroscience</div>

      </div>

      <p>

        Otkrijte svijet mikroskopski malih čuda i naprednih istraživanja. Naš tim

        stručnjaka je posvećen razumijevanju temelja znanosti kako bi

        unaprijedili medicinu, tehnologiju i naše razumijevanje svemira.

      </p>

      <section className="science-importance">

        <h2><FaLaughBeam /> Nauka je važna i zabavna</h2>

        <p>

          U svijetu gdje se granice znanja neprestano pomjeraju, nauka nudi

          alat za razumijevanje složenosti oko nas. Osim što doprinosi našem

          svakodnevnom životu, nauka nas uči kritičkom razmišljanju i rješavanju problema.

          A istraživanje novih koncepta može biti izuzetno zabavno!

        </p>

      </section>

      <section className="fun-facts">

        <h2><FaRegLaughBeam /> Zanimljive činjenice</h2>

        <ul>

          {/\* {currentFacts.map((fact, index) => (

            <FunFact key={index} fact={fact.fact} />

          ))} \*/}

        </ul>

        {/\* <PaginationDots /> \*/}

      </section>

    </div>

  );

};

export default HomePage;

import React from 'react';

const FunFact = ({ fact }) => {

  return (

    <li className="fun-fact">

      <p>{fact}</p>

    </li>

  );

};

export default FunFact;

Dalje, imamo komponentu "PublicationUploadForm" koja se koristi za unos informacija o publikaciji i njeno preuzimanje kao datoteke.

Na početku se uvoze React i komponente: "InputField" i "TextareaField". Komponenta "InputField" koristi se za unos teksta u polje, dok "TextareaField" omogućava unos teksta u veće polje za unos teksta. Ove dve komponente predstavljaju reusable komponente.

Zatim se definiše funkcionalna komponenta "PublicationUploadForm". Unutar nje, koristi se React Hook "useState" kako bi se inicijalizovao objekat "publication" koji će sadržati podatke o publikaciji, uključujući naslov, autore, apstrakt, ključne reci i datoteku.

Imamo sledeće tri funkcije:

* "handleInputChange" - Ova funkcija se poziva kada se promeni sadržaj polja za unos teksta ili teksta unutar komponenti "InputField" i "TextareaField". Ona ažurira stanje objekta "publication" s novim vrednostima na temelju imena polja koje se menja.
* "handleFileChange" - Ova funkcija se poziva kada se odabere datoteka za unos. Ona ažurira datoteku u objektu "publication" s odabranom datotekom.
* "handleSubmit" - Ova funkcija se poziva kada se obrazac za unos podataka pošalje. Ona prvo proverava da li je odabrana datoteka, a zatim čita tu datoteku kao Base64 string i sprema je u lokalno skladište zajedno s ostalim informacijama o publikaciji. Nakon toga, prikazuje se poruka o uspešnom unosu ili poruka o grešci ako dođe do problema.

import React, { useState } from 'react';

import InputField from './InputField';

import TextareaField from './TextareaField';

const PublicationUploadForm = () => {

  const [publication, setPublication] = useState({

    title: '',

    authors: '',

    abstract: '',

    keywords: '',

    file: null

  });

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setPublication({ ...publication, [name]: value });

  };

  const handleFileChange = (e) => {

    setPublication({ ...publication, file: e.target.files[0] });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (publication.file) {

      const reader = new FileReader();

      reader.onload = function (event) {

        try {

          const base64File = event.target.result;

          const existingFiles = JSON.parse(localStorage.getItem('uploadedFiles') || '[]');

          const newFile = {

            title: publication.title,

            authors: publication.authors,

            abstract: publication.abstract,

            keywords: publication.keywords,

            file: base64File

          };

          existingFiles.push(newFile);

          localStorage.setItem('uploadedFiles', JSON.stringify(existingFiles));

          alert('Publication uploaded successfully!');

        } catch (error) {

          console.error('Error saving the file:', error);

          alert('Failed to upload the publication.');

        }

      };

      reader.onerror = function (error) {

        console.error('Error reading the file:', error);

      };

      reader.readAsDataURL(publication.file);

    }

    setPublication({

      title: '',

      authors: '',

      abstract: '',

      keywords: '',

      file: null

    });

  };

  return (

    <form onSubmit={handleSubmit}>

      <InputField

        label="Title"

        name="title"

        value={publication.title}

        onChange={handleInputChange}

      />

      <InputField

        label="Authors"

        name="authors"

        value={publication.authors}

        onChange={handleInputChange}

      />

      <TextareaField

        label="Abstract"

        name="abstract"

        value={publication.abstract}

        onChange={handleInputChange}

      />

      <InputField

        label="Keywords"

        name="keywords"

        value={publication.keywords}

        onChange={handleInputChange}

      />

      <label>

        Upload File:

        <input

          type="file"

          name="file"

          onChange={handleFileChange}

          required

        />

      </label>

      <button type="submit">Submit Publication</button>

    </form>

  );

};

export default PublicationUploadForm;

const TextareaField = ({ label, name, value, onChange }) => {

    return (

      <label>

        {label}:

        <textarea

          name={name}

          value={value}

          onChange={onChange}

          required

        />

      </label>

    );

  };

  export default TextareaField;

const InputField = ({ label, name, value, onChange, type = 'text' }) => {

    return (

      <label>

        {label}:

        <input

          type={type}

          name={name}

          value={value}

          onChange={onChange}

          required

        />

      </label>

    );

  };

  export default InputField;

Poslednja komponenta, "PublicationsList", koja se koristi za prikazivanje liste publikacija, omogućava pretragu i sortiranje tih publikacija. Takođe, koristi se komponenta "PublicationItem" za prikaz pojedinačnih stavki na listi publikacija. Pored toga, kod uključuje i funkciju "useSearch" koja se koristi za filtriranje publikacija na osnovu pretrage. Kratak pregled koda:

* useState i useEffect: Komponenta "PublicationsList" koristi "useState" i "useEffect" hook-ove kako bi upravljala stanjem komponente. Stanje "publications" čuva listu publikacija, "searchTerm" čuva trenutni unos za pretragu, a "sortAscending" označava trenutni redosled sortiranja. Unutar "useEffect" hook-a, komponenta čita prethodno spremljene publikacije iz lokalnog skladišta i postavlja ih u stanje "publications" kad se komponenta montira.
* downloadFile funkcija: Ova funkcija se koristi za preuzimanje datoteke publikacije. Kada korisnik klikne na dugme "Download File" pored publikacije, funkcija kreira privremeni link za preuzimanje i pokreće preuzimanje datoteke.
* handleSearchChange funkcija: Ova funkcija se poziva kada se promeni unos za pretragu. Ona ažurira stanje "searchTerm" s trenutnim unosom korisnika.
* handleSortToggle funkcija: Ova funkcija se poziva kada korisnik klikne na dugme za promenu redosleda sortiranja.
* useSearch funkcija: Ova funkcija je prilagođena kuka koja se koristi za filtriranje publikacija na osnovu unosa za pretragu. Ona prima listu publikacija i trenutni unos za pretragu, a zatim filtrira publikacije koje odgovaraju kriterijumima pretrage, kao što su naslov, autori, apstrakt i ključne reci. Filtrirane publikacije se čuvaju u stanju "filteredItems" i ažuriraju se svaki put kad se unos za pretragu promeni

Komponentu MojiArtikli, koja se koristi za prikazivanje i upravljanje korisničkim artiklima.

Prvo se koriste useState i useEffect hook-ovi kako bi se definisala stanja komponente i izvršila asinhrona operacija dobijanja artikala sa servera pri prvom renderovanju komponente.

Nakon toga, implementirane su funkcije za pretragu, otvaranje, brisanje i ažuriranje artikala. Ove funkcije koriste Axios biblioteku za slanje HTTP zahteva ka odgovarajućim API endpointima na serveru.

Komponenta takođe prikazuje tabelu sa artiklima i omogućava korisniku interakciju sa njima putem dugmadi za otvaranje, brisanje i ažuriranje. Takođe je dodata kolona za prikaz statusa odobrenja artikla.

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

import axios from 'axios';

import './MojiArtikli.css';

const MojiArtikli = () => {

  const [articles, setArticles] = useState([]);

  const [editingArticle, setEditingArticle] = useState(null);

  const [editedData, setEditedData] = useState({});

  const [searchQuery, setSearchQuery] = useState('');

  useEffect(() => {

    const fetchArticles = async () => {

      try {

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

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

          headers: {

            Authorization: `Bearer ${token}`,

          },

        });

        setArticles(response.data);

      } catch (error) {

        console.error('Error fetching articles:', error);

      }

    };

    fetchArticles();

  }, []);

  const handleSearchChange = (e) => {

    setSearchQuery(e.target.value.toLowerCase());

  };

  const handleOpenArticle = async (id, filename) => {

    try {

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

      const response = await axios.get(`http://127.0.0.1:8000/api/articles/${id}`, {

        headers: {

          Authorization: `Bearer ${token}`,

        },

        responseType: 'blob',

      });

      const contentDisposition = response.headers['content-disposition'];

      let finalFilename = filename;

      if (contentDisposition) {

        const matches = /filename="([^"]+)"/.exec(contentDisposition);

        if (matches.length > 1) {

          finalFilename = matches[1];

        }

      }

      const url = window.URL.createObjectURL(new Blob([response.data], { type: response.headers['content-type'] }));

      const link = document.createElement('a');

      link.href = url;

      link.setAttribute('download', finalFilename);

      document.body.appendChild(link);

      link.click();

      document.body.removeChild(link);

      alert('Article downloaded successfully!');

    } catch (error) {

      console.error('Error downloading article:', error);

      alert('Failed to download article.');

    }

  };

  const handleDeleteArticle = async (id) => {

    try {

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

      await axios.delete(`http://127.0.0.1:8000/api/articles/${id}`, {

        headers: {

          Authorization: `Bearer ${token}`,

        },

      });

      setArticles(articles.filter(article => article.id !== id));

      alert('Article deleted successfully!');

    } catch (error) {

      console.error('Error deleting article:', error);

      alert('Failed to delete article.');

    }

  };

  const handleUpdateArticle = (article) => {

    setEditingArticle(article.id);

    setEditedData({ ...article });

  };

  const handleInputChange = (e) => {

    const { name, value } = e.target;

    setEditedData({ ...editedData, [name]: value });

  };

  const handleSubmitUpdate = async (e) => {

    e.preventDefault();

    try {

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

      const response = await axios.put(`http://127.0.0.1:8000/api/articles/${editingArticle}`, editedData, {

        headers: {

          Authorization: `Bearer ${token}`,

        },

      });

      setArticles(articles.map((article) => (article.id === editingArticle ? response.data : article)));

      alert('Article updated successfully!');

      setEditingArticle(null);

      setEditedData({});

    } catch (error) {

      console.error('Error updating article:', error);

      alert('Failed to update article.');

    }

  };

  const closeModal = () => {

    setEditingArticle(null);

    setEditedData({});

  };

  const filteredArticles = articles.filter(article =>

    article.title.toLowerCase().includes(searchQuery)

  );

  return (

    <div className='moji-artikli'>

      <h2>Moji Artikli</h2>

      <input

        type="text"

        placeholder="Pretraži po naslovu..."

        value={searchQuery}

        onChange={handleSearchChange}

      />

      <table>

        <thead>

          <tr>

            <th>Title</th>

            <th>Content</th>

            <th>Published At</th>

            <th>Keywords</th>

            <th>Status</th> {/\* Dodata kolona za prikaz statusa odobrenja \*/}

            <th>Actions</th>

          </tr>

        </thead>

        <tbody>

          {filteredArticles.map((article) => (

            <tr key={article.id}>

              <td>{article.title}</td>

              <td>{article.content}</td>

              <td>{article.published\_at}</td>

              <td>{article.keywords}</td>

              <td>{article.odobreno ? 'Odobren' : 'Nije odobren'}</td> {/\* Prikazuje status odobrenja \*/}

              <td>

                <button onClick={() => handleOpenArticle(article.id)}>Otvori</button>

                <button onClick={() => handleDeleteArticle(article.id)}>Obrisi</button>

                <button onClick={() => handleUpdateArticle(article)}>Azuriraj</button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

      {editingArticle && (

        <div className="modal">

          <div className="modal-content">

            <span className="close" onClick={closeModal}>&times;</span>

            <h2>Edit Article</h2>

            <form onSubmit={handleSubmitUpdate}>

              <label>Title:</label>

              <input type="text" name="title" value={editedData.title || ''} onChange={handleInputChange} />

              <label>Content:</label>

              <textarea name="content" value={editedData.content || ''} onChange={handleInputChange}></textarea>

              <label>Published At:</label>

              <input type="text" name="published\_at" value={editedData.published\_at || ''} onChange={handleInputChange} />

              <label>Keywords:</label>

              <input type="text" name="keywords" value={editedData.keywords || ''} onChange={handleInputChange} />

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

            </form>

          </div>

        </div>

      )}

    </div>

  );

};

export default MojiArtikli;

Login je React funkcionalna komponenta koja omogućava korisniku da se prijavi na aplikaciju.

Na početku komponente, uvoze se potrebne biblioteke i hook-ovi iz React-a, kao što su useState i useNavigate za upravljanje rutama.

Zatim se definiše stanje komponente pomoću useState hook-a, koje sadrži podatke o emailu i lozinci korisnika. Podrazumevane vrednosti ovih polja postavljene su na test vrednosti radi olakšane demonstracije.

Funkcija handleChange se koristi za ažuriranje stanja komponente kada korisnik unese podatke u polja za email i lozinku.

Metoda handleSubmit se poziva kada korisnik klikne na dugme za prijavljivanje. Ova metoda šalje POST zahtev ka API endpointu za prijavljivanje sa unetim podacima. U slučaju uspešne prijave, dobijeni token se čuva u stanju komponente i u sessionStorage-u, a korisnik se preusmerava na stranicu za otpremanje fajlova. U slučaju greške pri prijavi, ispisuje se odgovarajuća poruka o grešci.

import React, { useState } from 'react';

import axios from 'axios';

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

const Login = ({setToken}) => {

  let navigate=useNavigate();

  const [formData, setFormData] = useState({

    email: 'marvin.aracely@example.org',

    password: 'password'

  });

  const handleChange = (e) => {

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

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

      console.log(response.data);

      setToken(response.data.token);

      sessionStorage.setItem("token",response.data.token);

      navigate('/upload')

    } catch (error) {

      console.error('Login error:', error.response.data.message);

    }

  };

  return (

    <div className="homepage">

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

        <h1>Login</h1>

        <div className="form-group">

          <label htmlFor="email">Email</label>

          <input

            type="email"

            id="email"

            name="email"

            placeholder="Enter your email"

            value={formData.email}

            onChange={handleChange}

            required

          />

        </div>

        <div className="form-group">

          <label htmlFor="password">Password</label>

          <input

            type="password"

            id="password"

            name="password"

            placeholder="Enter your password"

            value={formData.password}

            onChange={handleChange}

            required

          />

        </div>

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

      </form>

    </div>

  );

};

export default Login;

**Register** je React funkcionalna komponenta koja omogućava korisniku da se registruje na aplikaciju.

Metoda **handleChange** se koristi za ažuriranje stanja komponente kada korisnik unese podatke u polja za ime, email i lozinku.

Metoda **handleSubmit** se poziva kada korisnik klikne na dugme za registraciju. Ova metoda šalje POST zahtev ka API endpointu za registraciju sa unetim podacima. U slučaju uspešne registracije, korisnik se preusmerava na stranicu za prijavljivanje. U slučaju greške pri registraciji, ispisuje se odgovarajuća poruka o grešci.

import React, { useState } from 'react';

import axios from 'axios';

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

const Register = () => {

    let navigate= useNavigate();

  const [formData, setFormData] = useState({

    name: '',

    email: '',

    password: ''

  });

  const handleChange = (e) => {

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

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

      console.log(response.data);

      navigate('/login');

    } catch (error) {

      console.error('Registration error:', error.response.data.message);

    }

  };

  return (

    <div className="homepage">

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

        <h1>Register</h1>

        <div className="form-group">

          <label htmlFor="name">Name</label>

          <input

            type="text"

            id="name"

            name="name"

            placeholder="Enter your name"

            value={formData.name}

            onChange={handleChange}

            required

          />

        </div>

        <div className="form-group">

          <label htmlFor="email">Email</label>

          <input

            type="email"

            id="email"

            name="email"

            placeholder="Enter your email"

            value={formData.email}

            onChange={handleChange}

            required

          />

        </div>

        <div className="form-group">

          <label htmlFor="password">Password</label>

          <input

            type="password"

            id="password"

            name="password"

            placeholder="Enter your password"

            value={formData.password}

            onChange={handleChange}

            required

          />

        </div>

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

      </form>

    </div>

  );

};

export default Register;

# Link ka github repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-istrazivackalaboratorija_2020_0248>

# Reference

* Herbert, D. (2023, November 13). A Beginner's Guide to React.js. HubSpot. Retrieved from <https://blog.hubspot.com/website/react-js>
* Jalli, A. (2022, December 21). Laravel: A Comprehensive Guide. Built In. Retrieved from https://builtin.com/software-engineering-perspectives/laravel
* 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>
* Tešić, I. (2017, December 27). Introduction to the React Ecosystem. Medium. Retrieved from <https://medium.com/@ITesic/uvod-u-react-ekosistem-8ccfad0a1030>
* Heidi, E. (n.d.). A Practical Introduction to Laravel Eloquent ORM. DigitalOcean. Retrieved from <https://www.digitalocean.com/community/tutorial-series/a-practical-introduction-to-laravel-eloquent-orm>
* Kollegger, E. (2018, May 14). What is Axios.js and Why Should I Care? Medium. Retrieved from <https://medium.com/@MinimalGhost/what-is-axios-js-and-why-should-i-care-7eb72b111dc0>