

Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

Veb aplikacija za elektronsku evidenciju seminarskih radova studenata

Seminarski rad iz predmeta Internet tehnologije

Studenti:

Andrija Protić 2019/0142

Irena Živković 2019/0082

Anastasija Golijanin 2019/0301

Beograd, 2023.

## Sadržaj

Sadržaj.....	1
1. Korisnički zahtev.....	1
2. Opis sistema.....	2
1. Opis slučajeva korišćenja .....	2
1.1. SK1: Slučaj korišćenja – Prijava korisnika.....	2
1.2. SK2: Slučaj korišćenja – Kreiranje seminarskog rada.....	3
1.3. SK3: Slučaj korišćenja – Registracija korisnika .....	4
2. Opis arhitekture aplikacije .....	5
3. Model podataka .....	6
4. Specifikacija REST API-ja .....	7
3. Opis tehnologija korišćenih u aplikaciji .....	14
1. JavaScript .....	14
2. React JS .....	15
3. PHP.....	15
4. Bootstrap .....	16
5. Laravel.....	16
4. Korisničko uputstvo .....	18
5. Prikaz reprezentativnih delova koda .....	22
6. GitHub link ka repozitorijumu .....	41
7. Reference .....	41

## **1. Korisnički zahtev**

Potrebno je kreirati web aplikaciju za evidenciju seminarskih radova studenata. Aplikacija treba da omogući profesorima da kreiraju zahteve/zadatke koje studenti treba da ispune. Za svaki zadatak je potrebno prikazati temu, opis i rok za izradu rada. Potrebno je omogućiti profesorima mogućnost pregleda radova, kao i mogućnost ocenjivanja radova studenata. Kako bi profesor poboljšao svoj način rada potrebno je prikazati mu određene statistike poput broja predatih radova po zadatku, broja ocenjenih radova i slično. Takođe, potrebno je omogućiti profesoru dodavanje novih zadataka, ažuriranje i brisanje postojećih.

Za studente je potrebno omogućiti prikaz svih zadataka koje treba da predaju. Kako bi se studenti lakše snalazili, potrebno je omogućiti im sortiranje i pretragu zadataka. Kada student završi svoj seminarski rad, može da ga preda u pdf formatu.

## **2. Opis sistema**

### **1. Opis slučaja korišćenja**

Aplikacija se sastoji od velikog broja SK, pa ćemo izdvojiti nekoliko:

1. Prijava korisnika
2. Registracija korisnika
3. Kreiranje seminarskog rada

#### **1.1. SK1: Slučaj korišćenja – Prijava korisnika**

1. Korisnik otvara stranicu za prijavu i unosi svoje korisničko ime i lozinku.
2. Aplikacija proverava da li su uneseni podaci ispravni.
3. Ukoliko su podaci ispravni, korisniku se prikazuje glavna stranica aplikacije.
4. Ukoliko su podaci neispravni, korisniku se prikazuje poruka o grešci i zahteva se da unese ispravne podatke.
5. Nakon uspešne prijave, korisnik može da koristi sve funkcije aplikacije koje su mu dostupne.
6. Ako korisnik želi da se odjavi, može to uraditi klikom na dugme za odjavu koje se obično nalazi na glavnoj stranici aplikacije.
7. Nakon odjave, korisnik se vraća na stranicu za prijavu i mora da se ponovo prijavi ako želi da koristi aplikaciju.

## 1.2. SK2: Slučaj korišćenja – Kreiranje seminarskog rada

Korisnik želi da unese novi seminarski rad u sistem.

1. Korisnik pristupa delu aplikacije za unos novog seminarskog rada.
2. Aplikacija prikazuje formu za unos podataka o seminarskom radu.
3. Korisnik unosi naslov seminarskog rada.
4. Korisnik unosi autora seminarskog rada.
5. Korisnik prilaže datoteku seminarskog rada.
6. Korisnik proverava unete podatke i datoteku seminarskog rada.
7. Korisnik potvrđuje unos novog seminarskog rada.
8. Aplikacija čuva podatke o seminarskom radu i datoteku u bazu podataka.
9. Aplikacija prikazuje poruku o uspešnom unosu novog seminarskog rada.
10. Korisnik se vraća na početnu stranicu ili ostaje na stranici za unos novog seminarskog rada.

### 1.3. SK3: Slučaj korišćenja – Registracija korisnika

Koraci:

Korisnik pristupa stranici za registraciju.

1. Sistem prikazuje formular za registraciju koji sadrži polja za unos korisničkog imena, email adrese i lozinke.
2. Korisnik unosi svoje podatke u formular.
3. Sistem proverava unete podatke i vrši validaciju unosa (npr. proveru da li je email adresa ispravno uneta i da li je lozinka dovoljno jaka).
4. Ako su svi uneti podaci ispravni, sistem kreira novi nalog za korisnika i šalje mu potvrdni email sa instrukcijama za aktiviranje naloga.
5. Korisnik potvrđuje svoj nalog klikom na link u potvrdnom email-u.
6. Nakon uspešne verifikacije naloga, sistem korisnika redirektuje na početnu stranicu aplikacije, a korisnik se može prijaviti koristeći svoje korisničko ime i lozinku.

Alternativni tok:

- Ako neki od unetih podataka nisu ispravni, sistem prikazuje korisniku poruku sa opisom greške i traži ga da ispravi svoje podatke.
- Ako korisnik ne potvrdi svoj nalog u roku od određenog vremenskog perioda, sistem ga obaveštava da njegov nalog nije aktiviran i da će biti automatski izbrisan nakon određenog perioda neaktivnosti.

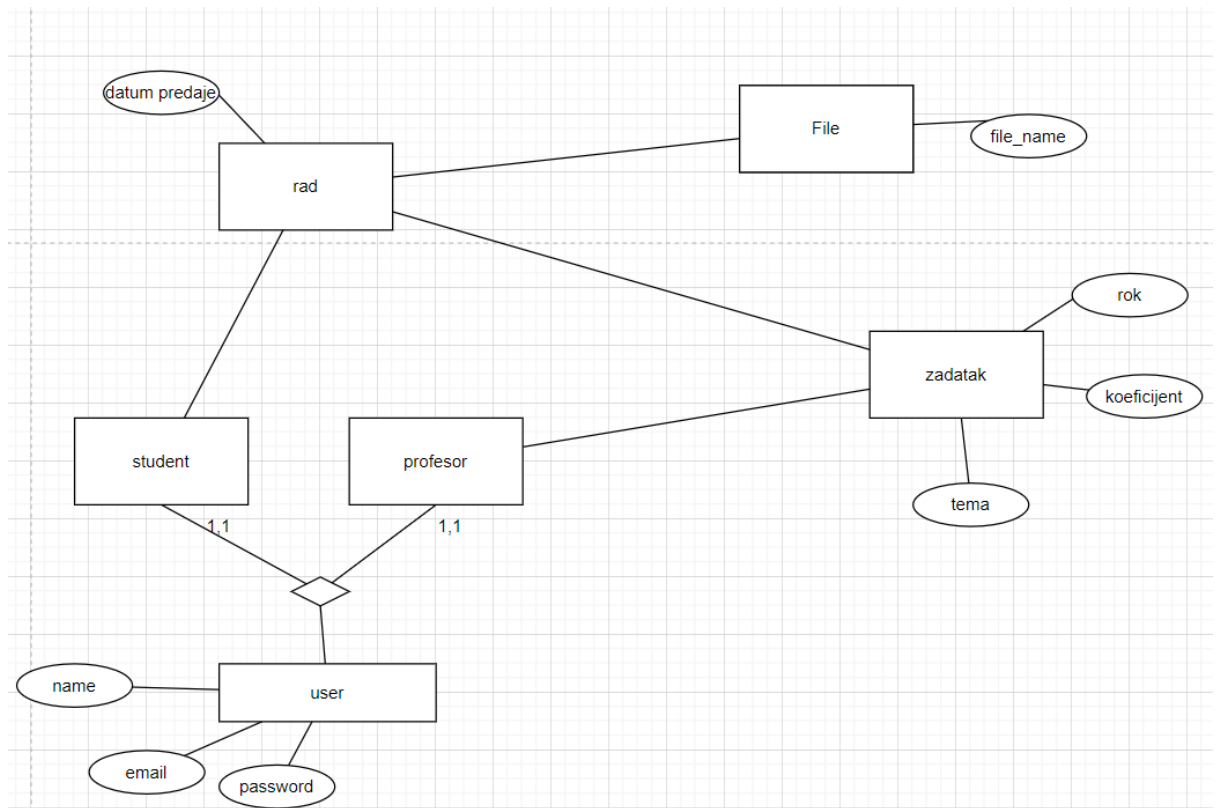
## 2. Opis arhitekture aplikacije

React se koristi kao glavni frontend okvir za kreiranje korisničkog interfejsa i klijentske logike. React koristi JavaScript biblioteke i alate za upravljanje stanjem, a takođe koristi različite biblioteke i komponente za stilizovanje korisničkog interfejsa.

Laravel se koristi kao backend okvir za kreiranje RESTful API-ja koji se koristi za povezivanje sa React aplikacijom. Laravel koristi MySQL bazu podataka za čuvanje podataka i pruža višeslojnu arhitekturu koja se sastoji od ruta, kontrolera, modela i pružača usluga za upravljanje podacima. Laravel pruža ugrađenu autentifikaciju koja se može koristiti za sigurno prijavljivanje i upravljanje korisničkim nalogima. Autentifikacija se zasniva na JWT-u (JSON Web Token) i omogućava sigurnu razmenu tokena između klijenta i servera.

Komponente se povezuju sa bazom podataka i komuniciraju uz korišćenje axios-a. Axios je popularna biblioteka za rad sa HTTP zahtevima u JavaScript okruženju, koja omogućava slanje zahteva za dobijanje podataka sa servera i upisivanje podataka na server. Biblioteka podržava različite HTTP metode kao što su GET, POST, PUT, DELETE, a takođe pruža mogućnosti za intersertovanje zahteva i odgovora, rukovanje greškama i podešavanje konfiguracije zahteva. Axios se može koristiti u kombinaciji sa različitim JavaScript okruženjima kao što su Node.js i React, kao i u browserima. Biblioteka je lako integrisati u postojeće projekte i pomaže u efikasnom i lakom radu sa HTTP zahtevima.

### 3. Model podataka



Slika 1. PMOV



#### 4. Specifikacija REST API-ja

Opis funkcije	Registracija
HTTP metoda	POST
URL	/api/register
URL parametri	?name=student&email= student @gmail.com&password= student
HTTP body parametri	<pre>{   "name": " student ",   "email": " student @gmail.com",   "password": " student ", }</pre>
Format HTTP body parametara	JSON
Izlazni parametri	<pre>{   "data": {     "name": " student ",     "email": " student @gmail.com",     "updated_at": "2022-12-22T21:05:37.000000Z",     "created_at": "2022-12-22T21:05:37.000000Z",     "id": 12   },   "access_token": "3 ReEm0L0o1yoHbyKfcrcFIfBm1CUQsPhxS7yjEnKS",   "token_type": "Bearer" }</pre>
Format izlaznih parametara	application/json
Opis funkcije	Login
HTTP metoda	POST
URL	/api/login
URL parametri	? email= student @gmail.com&password= student
HTTP body parametri	<pre>{   "email": " student @gmail.com",   "password": " student ", }</pre>

Format HTTP body parametara	JSON
Izlazni parametri	<pre>{   "user": {     "id": 12,     "name": " student ",     "email": " student @gmail.com",     "email_verified_at": null,     "created_at": "2022-12-22T21:05:37.000000Z",     "updated_at": "2022-12-22T21:05:37.000000Z"   },   "token": "4 KkCj615AzlcQcjtisSGGFICuyPFb8bJ7dTRs2j3I" }</pre>
Format izlaznih parametara	application/json

Opis funkcije	Odjava
HTTP metoda	POST
URL	/api/logout
URL parametri	nema
HTTP body parametri	(nema)
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>{   "status": "success", },</pre>
Format izlaznih parametara	application/json

Opis funkcije	Dodavanje novog zadatka
HTTP metoda	POST
URL	/api/ zadatak
URL parametri	?rok=2023-04-17&tema=tema1&koeficijent=10&user_id=2
HTTP body parametri	<pre>{   "tema": "tema1",   "user_id": "2",   "koeficijent": "10",   "rok": "2023-04-17", }</pre>
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>[   " kreirano!",   {     "tema": "tema1",     "user_id": "2",     "koeficijent": "10",     "rok": "2023-04-17",     "updated_at": "2023-02-21T16:46:31.000000Z",     "created_at": "2023-02-21T16:46:31.000000Z",     "id": 6   } ]</pre>
Format izlaznih parametara	application/json

Opis funkcije	Brisanje zadatka
HTTP metoda	DELETE
URL	/api/ zadatak /{id}
URL parametri	6
HTTP body parametri	nema
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>[   "Uspesno obrisano!",   {     "id": 6,     "user_id": 2,     "rok": "2023-04-17",     "koeficijent": 10,     "tema": "tema12",     "created_at": "2023-02-21T16:46:31.000000Z",     "updated_at": "2023-02-21T16:47:02.000000Z"   } ]</pre>
Format izlaznih parametara	application/json

Opis funkcije	Izmena zadatka
HTTP metoda	PUT
URL	/api/ zadatak /{id}
URL parametri	6?rok=2023-04-17&tema=tema12&koeficijent=10&user_id=2
HTTP body parametri	<pre>"user_id": "2", "rok": "2023-04-17", "koeficijent": "10", "tema": "tema12",</pre>
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>[   "Uspesno izmenjeno!",</pre>

	<pre>{   "id": 6,   "user_id": "2",   "rok": "2023-04-17",   "koeficijent": "10",   "tema": "tema12",   "created_at": "2023-02-21T16:46:31.000000Z",   "updated_at": "2023-02-21T16:47:02.000000Z" }</pre>
Format izlaznih parametara	application/json

Opis funkcije	Prikaz svih zadataka
HTTP metoda	GET
URL	/api/ zadatak
URL parametri	nema
HTTP body parametri	nema
Format HTTP body parametara	(nema)
Izlazni parametri	<pre>[   {     "id": 1,     "tema": "Istraživanje implementacije blockchain tehnologije",     "rok": "2023-03-31",     "koeficijent": 20,     "profesor": {       "id": 12,       "name": "profesor",       "email": "profesor@gmail.com",       "profesor": 1,       "email_verified_at": null,       "created_at": "2023-02-18T21:47:22.000000Z",       "updated_at": "2023-02-18T21:47:22.000000Z"     }   },   {     "id": 2,     "tema": "Analiza uticaja društvenih mreža na ponašanje korisnika", </pre>

	<pre> "rok": "2023-04-15", "koeficijent": 15, "profesor": {   "id": 12,   "name": "profesor",   "email": "profesor@gmail.com",   "profesor": 1,   "email_verified_at": null,   "created_at": "2023-02-18T21:47:22.000000Z",   "updated_at": "2023-02-18T21:47:22.000000Z" }, ... ]</pre>
Format izlaznih parametara	application/json

Opis funkcije	Prikaz svih radova
HTTP metoda	GET
URL	/api/ radovi
URL parametri	nema
HTTP body parametri	nema
Format HTTP body parametara	(nema)
Izlazni parametri	<pre> [   {     "id": 4,     "student": {       "id": 1,       "name": "Willard Sanford",       "email": "zhowe@example.org",       "profesor": 0,       "email_verified_at": "2023-02-18T21:47:22.000000Z",       "created_at": "2023-02-18T21:47:22.000000Z",       "updated_at": "2023-02-18T21:47:22.000000Z"     },     "zadatak": {       "id": 1,       "tema": "Istraživanje implementacije blockchain tehnologije", </pre>

	<pre>         "rok": "2023-03-31",         "koeficijent": 20,         "profesor": {             "id": 12,             "name": "profesor",             "email": "profesor@gmail.com",             "profesor": 1,             "email_verified_at": null,             "created_at": "2023-02- 18T21:47:22.000000Z",             "updated_at": "2023-02- 18T21:47:22.000000Z"         }     },     "datum_predaje": "2023-02-17",     "file": {         "id": 1,         "file_name": "bDlwBxVM5EKgfV62bTU5bSVyBXbIED6G .pdf",         "created_at": "2023-02-18T22:04:26.000000Z",         "updated_at": "2023-02-18T22:04:26.000000Z"     } }, {     "id": 5,     "student": {         "id": 11,         "name": "student",         "email": "student@gmail.com",         "profesor": 0,         "email_verified_at": null,         "created_at": "2023-02-18T21:47:22.000000Z",         "updated_at": "2023-02-18T21:47:22.000000Z"     },     "zadatak": {         "id": 1,         "tema": "Istraživanje implementacije blockchai n tehnologije",         "rok": "2023-03-31",         "koeficijent": 20,         "profesor": {             "id": 12,             "name": "profesor",             "email": "profesor@gmail.com",             "profesor": 1,             "email_verified_at": null,             "created_at": "2023-02- 18T21:47:22.000000Z",             "updated_at": "2023-02- 18T21:47:22.000000Z"         }     } } </pre>
--	---

	<pre>         },         "datum_predaje": "2023-02-18",         "file": {             "id": 2,             "file_name": "sPLDYQSFwcCttuHZE0Z4NAw3NNwLD2oj .pdf",             "created_at": "2023-02-18T22:10:59.000000Z",             "updated_at": "2023-02-18T22:10:59.000000Z"         }     },     ... ] </pre>
Format izlaznih parametara	application/json

### 3. Opis tehnologija korišćenih u aplikaciji

#### 1. JavaScript

JavaScript je popularan programski jezik koji se danas koristi ne samo u web razvoju, već i u razvoju desktop i mobilnih aplikacija, server-side aplikacija te čak i u IoT (Internet of Things) projektima. Razvijan je kontinuirano od 1995. godine, a danas postoje mnoge biblioteke i frameworkovi koji olakšavaju razvoj složenih aplikacija.

JavaScript ima široku primjenu u različitim tehnologijama kao što su React, Angular, Vue.js, Node.js, Express, te mnogim drugim. Takođe, zajednica programera pridonosi razvoju i popularnosti JavaScripta, nudeći mnoštvo rešenja za različite probleme, i pružajući podršku u online forumima i zajednicama.

JavaScript omogućuje web developerima da dinamički mijenjaju sadržaj web stranica te pruža jednostavan način za validaciju podataka koje korisnici unose u formularima. Takođe JavaScript se može koristiti za animaciju elemenata na stranici, validaciju korisničkih akcija te za stvaranje bogatih korisničkih interakcija koje korisnicima pružaju bolje iskustvo korištenja web aplikacija.



## 2. React JS

React (poznat i kao React.js ili ReactJS) je Javaskript biblioteka otvorenog koda koja se koristi za izgradnju korisničkih interfejsa. React primenjuje koncept virtuelnog DOM-a (Document Object Model) kako bi se omogućilo efikasno ažuriranje HTML dokumenta pri promeni podataka i olakšala izgradnja interaktivnih korisničkih interfejsa. Virtuelni DOM je brži i efikasniji jer omogućava minimalne izmene na stvarnom DOM-u, što rezultira poboljšanom performansom aplikacije.

React pregledi su obično organizovani korišćenjem komponenti koje predstavljaju delove korisničkog interfejsa i sadrže dodatne komponente definisane kao prilagođene HTML oznake. React obezbeđuje programeru model u kojem podkomponente ne mogu direktno da utiču na spoljašnje komponente, što olakšava održavanje i testiranje koda.

React je posebno popularan u izgradnji jednostraničnih aplikacija (SPA - Single Page Applications) koje omogućavaju bolje korisničko iskustvo i bržu interakciju sa korisnikom bez potrebe za preusmeravanjem na nove stranice. Takođe, React se često koristi u kombinaciji sa drugim bibliotekama i okvirima.

## 3. PHP

PHP (Hypertext Preprocessor) je široko rasprostranjen jezik za skriptovanje opšte namene otvorenog koda koji je posebno pogodan za web razvoj i može se ugraditi u HTML.

PHP je stekao popularnost zbog svoje jednostavnosti i sintakse nasleđene iz programskog jezika C. Tokom vremena jezik se proširivao i sticao mogućnosti za objektno orijentisano programiranje, naročito od verzije 5.0. Nalikuje jeziku C++ u smislu što dozvoljava i čisto-proceduralno programiranje, ali istovremeno omogućava i korišćenje klasa i drugih koncepata objektno orijentisanog programiranja (nasleđivanje, apstraktne klase i metode, interfejse itd.).

PHP se često koristi za kreiranje dinamičkih web stranica i web aplikacija, a omogućava interakciju sa bazama podataka, manipulaciju fajlovima na serveru, kao i slanje i primanje podataka preko HTTP protokola. Takođe, PHP ima veliku podršku zajednice, što znači da postoji mnogo gotovih biblioteka i alata koje programeri mogu koristiti u svojim projektima. PHP je podržan na svim glavnim operativnim sistemima i može se izvršavati na većini web servera, kao što su Apache i Nginx. Zbog svoje popularnosti i jednostavnosti, mnogi veliki sajtovi poput Facebooka, Wikipedije i WordPressa koriste PHP za svoje back-end procese.

#### 4. Bootstrap

Bootstrap je besplatni web frejmwork otvorenog koda, za kreiranje veb sajtova i veb aplikacija. Baziran je na HTML i CSS šablonima za tipografiju, kreiranju formulara, dugmadi, navigacionim i ostalim komponentama interfejsa, kao i opcionim JavaScript dodacima. Cilj Bootstrap-a je olakšavanje programiranja za veb. Koristi se i za web aplikacije, tj. podržava razvoj dinamičkih web sajtova i web aplikacija.

Razvila ga je Twitter ekipa i prvo je objavljen 2011. godine. Ovaj frejmwork se koristi za ubrzavanje razvoja modernih i respozivnih veb sajtova, čime se programerima omogućava brži i efikasniji proces dizajniranja i razvoja veb stranica. Bootstrap sadrži mnoge komponente kao što su dugmad, forme, navigacije, kartice, izbornike i mnoge druge koje se mogu koristiti za izradu respozivnih veb stranica koje se prilagođavaju različitim veličinama ekrana. Bootstrap takođe uključuje i mogućnost korišćenja JavaScripta za naprednije interakcije i efekte na veb stranici. Bootstrap je besplatan za korišćenje i dostupan je kao otvoreni kod na GitHub-u, što znači da se može besplatno preuzeti, koristiti i modifikovati.

#### 5. Laravel

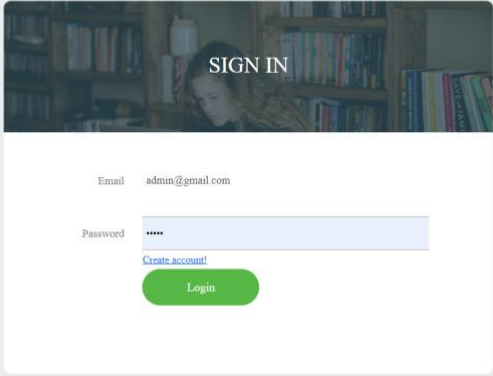
Laravel je besplatni PHP web okvir otvorenog koda, koji je namenjen razvoju web aplikacija koje prate arhitektonski obrazac model–pogled–kontrolor (MVC) i zasnovan na Symfony-ju. Neke od karakteristika Laravel-a su modularni sistem pakovanja sa namenskim menadžerom zavisnosti, različiti načini za pristup relacionim bazama podataka, uslužni programi koji pomažu u primeni i održavanju aplikacija, i njegova orijentacija ka syntactic sugar. Izvorni kod Laravel-a se nalazi na GitHub-u.

Laravel je popularan web okvir za PHP programski jezik, koji se koristi za razvoj moderne, dinamičke i skalabilne web aplikacije. On prati arhitektonski obrazac model–pogled–kontrolor (MVC), što omogućava programerima da jasno razdvoje logiku aplikacije od prezentacije i poslovnog sloja. Laravel je poznat po svom modularnom sistemu pakovanja i

menadžeru zavisnosti, koji olakšava dodavanje novih funkcionalnosti u aplikaciju. Takođe nudi mogućnost jednostavnog korišćenja različitih baza podataka, uključujući MySQL, PostgreSQL i SQLite, i ugrađeni sistem za migracije baza podataka. Laravel takođe sadrži različite alate za testiranje i debugging aplikacije, što pomaže programerima da brzo identifikuju i otklone greške. Osim toga, Laravel je poznat po svojoj orijentaciji ka syntactic sugar, što olakšava pisanje čistog i čitljivog koda.

## 4. Korisničko uputstvo

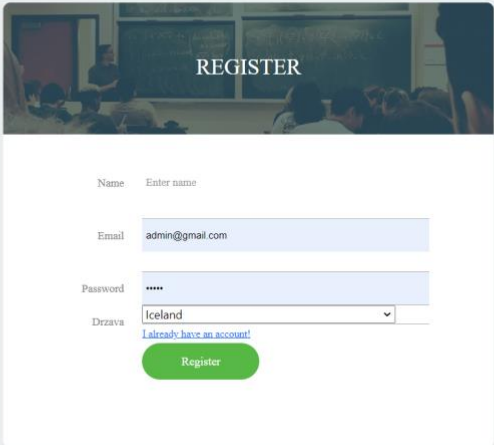
Prilikom pokretanja aplikacija korisniku se prikazuje login forma.



The screenshot shows a login form titled "SIGN IN" with a background image of a person reading. The form includes an "Email" field with the value "admin@gmail.com", a "Password" field with masked characters "\*\*\*\*", a "Create account!" link, and a green "Login" button.

Slika 2 login

Sa ove stranice korisnik može da dodje na stranicu za registraciju, ukoliko nema nalog, može da ga napravi.



The screenshot shows a registration form titled "REGISTER" with a background image of a classroom. The form includes a "Name" field with the placeholder "Enter name", an "Email" field with the value "admin@gmail.com", a "Password" field with masked characters "\*\*\*\*", a "Drzava" (Country) dropdown menu with "Iceland" selected, a "I already have an account!" link, and a green "Register" button.

Slika 3 register

Obe forme izgledaju identično, samo sa drugim poljima. Na vrhu svake forme se nalazi slika, svaki put kad se stranica osveži, na formi se prikazuje druga slika. Ovo je urađeno uz pomoć javnog apija. Takođe na strain za registarciju, imamo kombobox u kom je prikazan spisak svih država koji je takođe učitao pomoću javnog apija.

Nakon što se ulogujemo kao student, otvara nam se stranica na kojoj se nalazi spisak svih zadataka koje treba da obavimo (seminarski radovi koje treba da predamo). Ovo je prikazano na slici ispod.

### Zadaci za predaju

Search

SORTIRAJ PO ROKU

ID	Tema	Rok	Profesor	Predaj
1	Istraživanje implementacije blockchain tehnologije	2023-03-31	profesor	Predaj <input type="button" value="Choose File"/> No file chosen
2	Analiza uticaja društvenih mreža na ponašanje korisnika	2023-04-15	profesor	Predaj <input type="button" value="Choose File"/> No file chosen
3	Razvoj softverskog sistema za upravljanje zalihama	2023-05-01	profesor	Predaj <input type="button" value="Choose File"/> No file chosen
4	Analiza performansi različitih algoritama pretrage	2023-05-15	profesor	Predaj <input type="button" value="Choose File"/> No file chosen
5	Razvoj mobilne aplikacije za upravljanje projektima	2023-06-01	profesor	Predaj <input type="button" value="Choose File"/> No file chosen

Slika 4 zadaci

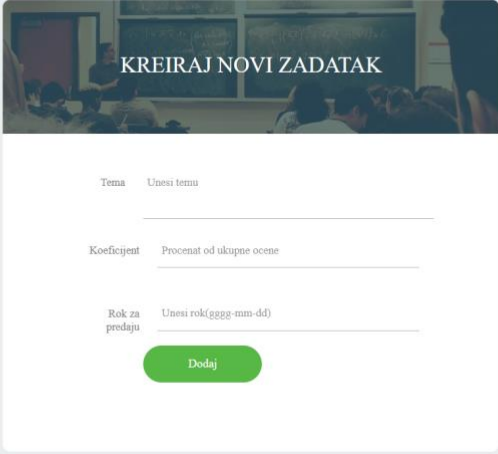
Na ovo stranici su prikazani svi zadaci koje student treba da obavi, radi lakšeg snalaženja korisnicima aplikacije je omogućeno pretraživanje zadataka, kao i sortiranje po roku. Kada student obavi rad klikom na opciju Choose file mu se otvara prozor preko kog može da odabere koji fajl želi da preda za određen zadatak, a zatim klikom na dugme predaj rad je predat profesoru.

Kako bismo videli sve predate radove, potrebno je da se ulogujemo kao profesor. Tada nam se otvara stranica kaon a slici ispod.

ID	Tema	Rok	Procenat od ukupne ocene	Opcije
1	Istraživanje implementacije blockchain tehnologije	2023-03-31	20	<input type="button" value="🗑️"/> <input type="button" value="✎️"/> <input type="button" value="VIDI RADOVE 📄"/>
2	Analiza uticaja društvenih mreža na ponašanje korisnika	2023-04-15	15	<input type="button" value="🗑️"/> <input type="button" value="✎️"/> <input type="button" value="VIDI RADOVE 📄"/>
3	Razvoj softverskog sistema za upravljanje zalihama	2023-05-01	25	<input type="button" value="🗑️"/> <input type="button" value="✎️"/> <input type="button" value="VIDI RADOVE 📄"/>
4	Analiza performansi različitih algoritama pretrage	2023-05-15	10	<input type="button" value="🗑️"/> <input type="button" value="✎️"/> <input type="button" value="VIDI RADOVE 📄"/>
5	Razvoj mobilne aplikacije za upravljanje projektima	2023-06-01	30	<input type="button" value="🗑️"/> <input type="button" value="✎️"/> <input type="button" value="VIDI RADOVE 📄"/>

Slika 5 admin


Profesor takođe može da vidi sve zadatke koji su postavljeni, može da vidi sve radove koji su predate, kao i da vrši CRUD operacije nad zadacima. Klikom na dugme +Dodaj, korisniku se otvara stranica sa formom preko koje može da kreira novi zadatak studentima.



The screenshot shows a web form titled "KREIRAJ NOVI ZADATAK" (Create New Task) in a classroom setting. The form has three input fields: "Tema" (Topic) with placeholder text "Unesi temu", "Koeficijent" (Coefficient) with placeholder text "Procenat od ukupne ocene", and "Rok za predaju" (Deadline) with placeholder text "Unesi rok(gggg-mm-dd)". A green "Dodaj" (Add) button is at the bottom.

Slika 6 dodaj

Klikom dugmeta sa ikonom olovke se otvara forma koja je već popunjena podacima o određenom zadatku. Na ovoj formi korisnik može da ažurira određen zadatak.



The screenshot shows a web form titled "IZMENI ZADATAK" (Edit Task) in a classroom setting. The form has three input fields: "Tema" (Topic) with the value "Istraživanje implementacije blockchain tehnologije", "Koeficijent" (Coefficient) with the value "20", and "Rok za predaju" (Deadline) with the value "2023-03-31". A green "Izmeni" (Edit) button is at the bottom.

Slika 7 ažuriraj

Klikom na dugme vidi radove, korisniku se otvara stranica na kojoj može da pregleda i oceni radove studenata.

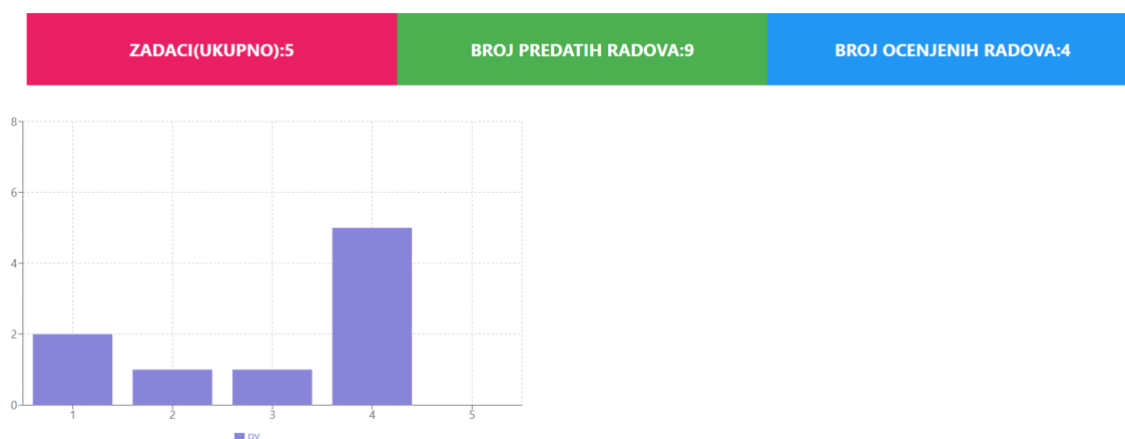
id	student	zadatak	datum_predaje	rad	oceni_rad
4	Willard Sanford	Istraživanje implementacije blockchain tehnologije	2023-02-17	<a href="#">OTVORI</a>	<button>OCENI</button>
5	student	Istraživanje implementacije blockchain tehnologije	2023-02-18	<a href="#">OTVORI</a>	<button>OCENI</button>

Rows per page:  1 - 2 of 2 < >

### Slika 8 radovi

Ovi radovi su prikazani uz pomoc DataTable tabele i professor može da sortira I pretražuje tabelu kako bi lakše došao do radova koje želi da ažurira.

Klikom na dugme statistike, profesoru se prikazuje stranica na kojoj može da vidi grafikom sa brojem predatih radova po zadatku.



### Slika 9 admin statistike

## 5. Prikaz reprezentativnih delova koda

```
<?php

namespace App\Http\Controllers;

use App\Http\Resources\RadResource;
use App\Models\Rad;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class RadController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return RadResource::collection(Rad::all());
    }

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

            'naziv' => 'required|string',
            'student'=>'required|integer|exists:users,id',
            'zadatak_id'=>'required|integer|exists:zadataks,id',

            'datum_predaje' => 'required|date',
```



```

    });

    if ($validator->fails())
        return response()->json($validator->errors());
    $d = Rad::create([
        'naziv' => $request->naziv,
        'student' => $request->student,
        'zadatak_id' => $request->zadatak_id,

        'datum_predaje' => $request->datum_predaje,

    ]);
    $d->save();
    return response()->json([' kreirano!', $d]);
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\Rad $rad
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    return new RadResource(Rad::find($id));
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\Rad $rad
 * @return \Illuminate\Http\Response
 */
public function edit(Rad $rad)
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Models\Rad $rad
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, Rad $rad)

```

```

{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param \App\Models\Rad $rad
 * @return \Illuminate\Http\Response
 */
public function destroy(Rad $rad)
{
    //
}
}

<?php

namespace App\Http\Controllers;

use App\Models\File;
use App\Models\Rad;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Str;
class FileController extends Controller
{
    public function index() {
        $files = File::all();
        return response()->json(["status" => "success", "count" =>
count($files), "data" => $files]);
    }

    public function upload(Request $request) {
        $imagesName = [];
        $response = [];

        $validator = Validator::make($request->all(),
            [
                'files' => 'required',
                'files.*' => 'required|mimes:pdf|max:2048',
                'datum_predaje' => 'required|string|max:30',
                'student' => 'required|integer|exists:users,id',
                'zadatak_id' => 'required|integer|exists:zadataks,id',
            ]
        );

        if ($validator->fails())

```

```

        return response()->json([
            'validation_errors'=>$validator->errors(),
        ]);

        if($request->has('files')) {
            foreach($request->file('files') as $file) {
                $filename = Str::random(32)." ".$file-
>getClientOriginalExtension();
                $file->move('uploads/', $filename);

                $fajl= File::create([
                    'file_name' => $filename
                ]);
            }

            Rad::create([
                'datum_predaje' => $request->datum_predaje,
                'student' => $request->student,
                'zadatak_id' => $request->zadatak_id,
                'file_id' => $fajl->id,

            ]);
            $response["status"] = "200";
            $response["message"] = "Success!";

        }

        else {
            $response["status"] = "failed";
            $response["message"] = "Failed! image(s) not uploaded";
        }
        return response()->json($response);
    }
}

<?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 AutfController extends Controller
{
    public function register(Request $request)

```

```

{
    $validator = Validator::make(
        $request->all(),
        [
            'password' => 'required' ,
            'name' => 'required|string|max:100',
            'email' => 'required|string|max:100|email'
        ]
    );
    if ($validator->fails())
        return response()->json($validator->errors());

    $user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password)]);

    $token = $user->createToken('auth_token')->plainTextToken;
    return response()->json(['data' => $user, 'access_token' => $token,
'token_type' => 'Bearer']);
}
public function login(Request $request)
{
    $validator = Validator::make($request->all(), [
        'email' => 'required|email',
        'password' => 'required|string',
    ]);

    if ($validator->fails()) {
        return response()->json(['success'=>false]);
    }

    if (!Auth::attempt($request->only('email', 'password'))) {
        return response()->json(['poruka' => 'Pogresan email ili
password!']);
    }

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

    if(!$user){
        return response()->json([
            'status'=>401,
            'message'=>'Invalid credentials',
        ]);
    }else{
        if($user->profesor==1){
            $role='admin';

```

```

        $token = $user->createToken($user->email.'_AdminToken',['server:admin'])->plainTextToken;
        $response = [
            'status'=>200,
            'username'=>"Admin",
            'token' => $token,
            'role'=> $role,
            'id'=>$user->id
        ];
        return response()->json([$response,'success'=>true ]);
    }else{
        $role=' ';
        $token = $user->createToken($user->email.'_Token',[''])->plainTextToken;
        $response = [
            'status'=>200,
            'username'=>$user->name,
            'token' => $token,
            'role'=> $role,
            'id'=>$user->id
        ];
        return response()->json([$response,'success'=>true ]);
    }
}

```

```

    }

    public function logout()
    {
        auth()->user()->tokens()->delete();
        return response()->json(['Uspešno ste se izlogovali!']);
    }
}

```

<?php

```

namespace Database\Seeders;

use App\Models\Komentar;
use App\Models\Rad;
use App\Models\User;
use App\Models\Zadatak;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;

```

```

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        User::truncate();
        Komentar::truncate();
        Rad::truncate();
        Zadatak::truncate();

        User::factory(10)->create();

        User::create([
            'name' => 'student',
            'email' => 'student@gmail.com',
            'password' => Hash::make('student')]);
        User::create([
            'name' => 'profesor',
            'email' => 'profesor@gmail.com',
            'profesor' => 1,
            'password' => Hash::make('profesor')]);

        (new ZadatakSeeder())->run();
        (new RadSeeder())->run();
        (new KomentarSeeder())->run();

    }
}

<?php

use App\Http\Controllers\AutfController;
use App\Http\Controllers\FileController;
use App\Http\Controllers\KomentarController;
use App\Http\Controllers\RadController;
use App\Http\Controllers\ZadatakController;
use App\Models\Komentar;
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("/zadatak",[ZadatakController::class,'index']);

Route::get("/radovi",[RadController::class,'index']);

Route::post("/register",[AutfController::class,'register']);
Route::post("/login",[AutfController::class,'login']);


Route::get('/files', [FileController::class, 'index'])->name('files');
Route::post('/files', [FileController::class, 'upload'])->name('files');


Route::get("/komentar",[KomentarController::class,'index']);
Route::post("/komentar",[KomentarController::class,'store']);


Route::group(['middleware' => ['auth:sanctum']], function () {

    Route::post("/radovi",[RadController::class,'store']);

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

});

Route::post("/zadatak",[ZadatakController::class,'store']);
Route::put("/zadatak/{id}",[ZadatakController::class,'update']);
Route::middleware(['auth:sanctum','isAPIAdmin'])->group(function(){ //ako je
u logovan admin

    Route::post("/komentar",[KomentarController::class,'store']);

    Route::delete("/zadatak/{id}",[ZadatakController::class,'destroy']);

```

```

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

});

import './App.css';
import Navbar from './komponente/Navbar';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import { useEffect, useState } from 'react';
import LoginPage from './komponente/LoginPage';
import React from 'react';
import RegisterPage from './komponente/RegisterPage';
import Zadaci from './komponente/Zadaci';
import axios from 'axios';
import AdminPocetna from './komponente/AdminPocetna';
import Dodaj from './komponente/Dodaj';
import Azuriraj from './komponente/Azuriraj';
import Radovi from './komponente/Radovi';
import Statistike from './komponente/Statistike';

const axiosInstance = axios.create({
  baseURL: process.env.REACT_APP_API_URL,
});

function App() {
  const [token,setToken] = useState(null);
  const [zadaci,setZadaci] = useState([ ])
  const [radovi,setRadovi] = useState([ ])
  const [komentari,setKomentari] = useState([ ])

  useEffect(() => {
    const getZadaci = async () => {
      try {
        const res = await axiosInstance.get(
"http://127.0.0.1:8000/api/zadatak",
        {
          headers:{'Authorization': `Bearer ${
window.sessionStorage.getItem('auth_token')}`},
        }
      );
      setZadaci(res.data.data);
      console.log(res.data.data)
    } catch (err) {
      console.log(err);
    }
  });
};

```



```

    getZadaci();
  }, [ axiosInstance]);
  useEffect(() => {
    const getRadovi = async () => {
      try {
        const res = await axiosInstance.get(
"http://127.0.0.1:8000/api/radovi",
        {
          headers: {'Authorization': `Bearer ${
window.sessionStorage.getItem('auth_token')}`},
        }
      );
      setRadovi(res.data.data);
      console.log(radovi)
    } catch (err) {
      console.log(err);
    }
  };
  getRadovi();
}, [ axiosInstance]);
  useEffect(() => {
    const getKomentari = async () => {
      try {
        const res = await axiosInstance.get(
"http://127.0.0.1:8000/api/komentar",
        {
          headers: {'Authorization': `Bearer ${
window.sessionStorage.getItem('auth_token')}`},
        }
      );
      setKomentari(res.data.data);
      console.log(res.data)
    } catch (err) {
      console.log(err);
    }
  };
  getKomentari();
}, [ axiosInstance]);

function obrisi(id){

  axios
    .delete("http://127.0.0.1:8000/api/zadatak/"+id,{headers: {'Authorization':
`Bearer ${ window.sessionStorage.getItem('auth_token')}` } } )
    .then((res)=>{
      console.log(res.data);
      alert("OBRISANO")
    })

```

```

        .catch(function (error) {
            if (error.response) {
                // Request made and server responded
                console.log(error.response.data);

                console.log(error.response.status);
                console.log(error.response.headers);
            } else if (error.request) {
                // The request was made but no response was received
                console.log(error.request);
            } else {
                // Something happened in setting up the request that triggered an
Error
                console.log('Error', error.message);
            }
        });
    });
}

const [zadatakZaAzuriranj,setZAzuriraj] = useState(null)
const [zadatakZaPrikazRadova,setzRadovi] = useState(null)

function setZadatakAzuriraj(zadatak){
    setZAzuriraj(zadatak)
}
function setZadatakRadovi(zadatak){
    setzRadovi(zadatak)
}
return (
    <div className="App">
        <BrowserRouter >
            <Navbar token={token} setToken={setToken} ></Navbar>
            <Routes>
                <Route path="/" element={<LoginPage
addToken={setToken}></LoginPage>}></Route>
                <Route path="/register"
element={<RegisterPage ></RegisterPage>}></Route>
                <Route path="/zadaci"
element={<Zadaci zadaci={zadaci}></Zadaci>}></Route>

                <Route path="/admin/radovi" element={<Radovi
radovi={radovi} zadatak={zadatakZaPrikazRadova}></Radovi>}></Route>

                <Route path="/admin/azuriraj" element={<Azuriraj
zadatak={zadatakZaAzuriranj}></Azuriraj>}></Route>
                <Route path="/admin/statistike"
element={<Statistike radovi={radovi} zadaci={zadaci}
komentar={komentar}></Statistike>}></Route>
            </Routes>
        </BrowserRouter>
    </div>
);

```

```

        <Route path="/admin/dodaj" element={<Dodaj></Dodaj>}></Route>
        <Route path="/admin" element={<AdminPocetna zadaci={zadaci}
obrisi={obrisi} setZadatakAzuriraj={setZadatakAzuriraj}
setZadatakRadovi={setZadatakRadovi}></AdminPocetna>}></Route>

    </Routes>

</BrowserRouter>
</div>
);
}

export default App;

import React from 'react';
import {useState} from "react";
import './LoginStyle.css';
import axios from "axios";
import { useNavigate } from 'react-router-dom';
function LoginPage({addToken}) {

    const [userData,setUserData]=useState({
        email:"",
        password:""
    });
    function handleInput(e){

        let newUserData = userData;

        newUserData[e.target.name]=e.target.value;
        console.log(newUserData)
        setUserData(newUserData);

    }
    let navigate = useNavigate();
    function handleLogin(e){

        e.preventDefault();

        axios
            .post("http://127.0.0.1:8000/api/login", userData )
            .then((res)=>{
                console.log(res.data[0]);
                if(res.data.success===true){

```

```

        window.sessionStorage.setItem("auth_token",res.data[0].token);
        window.sessionStorage.setItem("auth_name",res.data[0].username);
        window.sessionStorage.setItem("auth_id",res.data[0].id);

        addToken(res.data[0].token);
        console.log(res.data[0].token);
        if(res.data[0].role === 'admin')
        {

            navigate("/admin")
        }
        else{
            navigate("/zadaci");
        }

    }else{
        alert("NEUSPESNO");
    }
});

}
return (
    <div className="limiter">
    <div className="container-login100">
        <div className="wrap-login100">
            <div className="login100-form-title" >
                <span className="login100-form-title-1">
                    Sign In
                </span>
            </div>

            <form className="login100-form validate-form" method="post"
onSubmit={handleLogin}>
                <div className="wrap-input100 validate-input m-b-26" data-
validate="Email is required">
                    <span className="label-input100">Email</span>
                    <input className="input100" type="email" name="email"
id="email" placeholder="Enter email" onInput={handleInput}/>
                    <span className="focus-input100"></span>
                </div>
                <br /><br /><br />
                <div className="wrap-input100 validate-input m-b-18" data-
validate = "Password is required">
                    <span className="label-input100">Password</span>

```

```

        <input className="input100" type="password"
name="password" id="password" placeholder="Enter password"
onInput={handleInput}/>
        <span className="focus-input100"></span>
      </div>

      <div className="flex-sb-m w-full p-b-30">
        <div className="contact100-form-checkbox">
          <a href="/register">Create account!</a>
        </div>

      </div>

    </div>

    <div className="container-login100-form-btn">
      <button className="login100-form-btn" id="login"
name="login"> Login </button>
    </div>
  </form>
</div>
</div>
</div>
</div>
);
}

export default LoginPage;

import axios from 'axios';
import { MDBDataTableV5 } from 'mdbreact';

import React, { useState } from 'react';
function Radovi({radovi}) {
  const [komentarData,setKomentarData]=useState({

    profesor_id:window.sessionStorage.getItem("auth_id"),
    ocena:51,
    opis:"Excepturi officia voluptatem facere. Modi eveniet alias totam
dignissimos et labore. Voluptate exercitationem rerum quo.",

  });
  function handleInput(e){

    let newKomentarData = komentarData;

    newKomentarData[e.target.name]=e.target.value;

```

```

        console.log(newKomentarData)
        setKomentarData(newKomentarData);

    }

    function oceni(id){
        komentarData.rad_id=id;
        var config = {
            method: 'post',
            url: 'http://127.0.0.1:8000/api/komentar?rad?id='+{id},
            headers:{'Authorization': `Bearer ${
window.sessionStorage.getItem('auth_token')}`},
            data:komentarData
        };

        axios(config)
            .then(function (response) {

                console.log(response);
                alert("USPEH")

            })
            .catch(function (error) {

                console.log(error);

            });
    }

    const [datatable, setDatatable] = React.useState({

        columns: [
            {
                label: 'id',
                field: 'id',
                width: 150,
            },
            {
                label: 'student',
                field: 'student',
                width: 200,
            },
            {

```

```

        label: 'zadatak',
        field: 'zadatak',
        width: 270,
      },
      {
        label: 'datum_predaje',
        field: 'datum_predaje',
        width: 270,
      },
      {
        label: 'rad',
        field: 'rad',
        width: 270,
      },
      {
        label: 'oceni_rad',
        field: 'oceni_rad',
        width: 270,
      }
    ],
    rows: radovi.map(r=>{
      return{
        id: r.id,
        student:r.student.name,
        zadatak:r.zadatak.tema,
        datum_predaje: r.datum_predaje,
        rad: <a
href={`http://127.0.0.1:8000/uploads/${r.file.file_name}`} download
target={`'_blank'`} >OTVORI</a> ,
        oceni_rad: <><input type="text" name='ocena'
onInput={handleInput}/><button className='btn'
onClick={()=>oceni(r.id)}>Oceni</button></>
      }
    })
  })

  return <div className='container'><MDBDataTableV5 hover
entriesOptions={[5, 20, 25]} entries={5} pagesAmount={4} data={datatable}
/></div>;
}

export default Radovi;

```

```

import { BsFillTrashFill, BsPencilFill } from 'react-icons/bs';
import { useNavigate } from 'react-router-dom';
import { AiOutlinePlus ,AiFillFolder} from 'react-icons/ai';
import { BiStats } from 'react-icons/bi';

import React from 'react';
function AdminPocetna({zadaci,obrisi,setZadatakAzuriraj,setZadatakRadovi}) {
  let navigate = useNavigate();
  function dodaj(){
    navigate("/admin/dodaj/");
  }
  function azuriraj(zadatak){

    setZadatakAzuriraj(zadatak);

    navigate("/admin/azuriraj/");
  }
  function vidiRadove(zadatak){
    setZadatakRadovi(zadatak);

    navigate("/admin/radovi/");
  }
  function statistike(){

    navigate("/admin/statistike/");
  }
  console.log(zadaci.filter((z)=>z.profesor.id==window.sessionStorage.getItem("auth_id")))
  return (

    <div className='container'>
      <button className="btn btn-primary"
onClick={dodaj}><AiOutlinePlus></AiOutlinePlus>Dodaj</button>
      <button className="btn btn-primary"
onClick={statistike}><BiStats></BiStats>Statistike</button>

      <table id="dtBasicExample" className="table table-striped table-bordered table-sm" cellSpacing="0" width="100%">
        <thead>
          <tr>
            <th className="th-sm">ID
            </th>
            <th className="th-sm">Tema
            </th>
            <th className="th-sm">Rok
            </th>
            <th className="th-sm">Procenat od ukupne ocene
            </th>

```



```

        <th className="th-sm">Opcije
      </th>
    </tr>
  </thead>
  <tbody>
    {zadaci.filter((z)=>z.profesor.id==window.sessionStorage.getItem
("auth_id")).map((z)=><tr
key={z.id}><td>{z.id}</td><td>{z.tema}</td><td>{z.rok}</td><td>{z.koeficijent}
</td><td><button className="btn" onClick={() =>
obrisi(z.id)}><BsFillTrashFill></BsFillTrashFill></button><button
className="btn" onClick={() =>
azuriraj(z)}><BsPencilFill></BsPencilFill></button><button className='btn'
onClick={()=>vidiRadove(z)}>Vidi radove
<AiFillFolder></AiFillFolder></button></td></tr>))}}
  </tbody>

</table>
</div>

);
}

export default AdminPocetna;

import { BsPaperclip } from 'react-icons/bs';
import React, { useState } from 'react';
import { Modal } from 'bootstrap';
import axios from 'axios';
import BarChartComponent from './BarChart';

function Statistike({zadaci,radovi,komentari}) {

  return (

    <>
      <div className="wrapper">
        <div className="box pink">
          ZADACI(UKUPNO): <br/>
          <span>{zadaci.length}</span>
        </div>
        <div className="box green">
          BROJ PREDATIH RADOVA:<br/>
          <span>{radovi.length}</span>
        </div>
      </div>
    </>
  )
}

```

```

        <div className="box blue">
            BROJ OCENJENIH RADOVA:<br/>
            <span>{komentar.length}</span>
        </div>

    </div>
    <BarChartComponent radovi={radovi} zadaci={zadaci}
komentar={komentar}></BarChartComponent>
    </>

    );
}

export default Statistike;

import React from "react";
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend } from
"recharts";

const BarChartComponent = ({zadaci,radovi,komentar}) => {
    const data=[];
    function brojPredatihRadova(id){ //po zadatku
        var broj=0;
        for(var i=0;i<radovi.length;i++){
            if(radovi[i].zadatak.id==id){
                broj++;
            }
        }

        return broj;
    }

    for(var i=0;i<zadaci.length;i++){
        data.push({name:zadaci[i].id,pv:brojPredatihRadova(zadaci[i].id)})
    }

    return (
        <BarChart
            width={800}
            height={500}
            data={data}
            margin={{ top: 50, right: 30, left: 20, bottom: 5 }}
        >
            <CartesianGrid strokeDasharray="3 3" />
            <XAxis dataKey="name" />

```

```
    <YAxis />
    <Tooltip />
    <Legend />
    <Bar dataKey="pv" fill="#8884d8" />

  </BarChart>
);
};

export default BarChartComponent;
```

## 6. GitHub link ka repozitorijumu

<https://github.com/irenazivkovic/seminarskiRadovi>

## 7. Reference

- Vlajić S, „Projektovanje softvera“, FON, 2020.
- PHP skripta, Laravel – konačna skripta, React script
- <https://reactjs.org/blog/2013/06/05/why-react.html>
- <https://www.php.net/manual/en/intro-what-is.php>
- <https://www.w3.org/standards/webdesign/htmlcss>