Izrada edukacijske platforme temeljene na MERN-u

Ivanišević, Anđela

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Anđela Ivanišević

IZRADA EDUKACIJSKE PLATFORME TEMELJENE NA MERN-U

Završni rad

Split,2024

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za Informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Izrada edukacijske platforme temeljene na MERN-u

Anđela Ivanišević

SAŽETAK

U ovom završnom radu opisani su ključni pojmovi vezani za MERN stack. MERN stack je popularna tehnologija koja se sastoji od MongoDB (baza podataka), Express.js(serverski okvir), React (frontend dio) i Node.js (backend). U radu su opisane ključne funkcionalnosti edukacijske platforme, kao što su pregled kolegija, pregled lekcija, slanje poruka i pohranjivanje datoteka.

Ključne riječi: web aplikacija, React, Node.js, MongoDB, Express

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 29 stranica, 19 grafičkih prikaza, 0 tablica i 7 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Prof. Dr. sc. Marko Rosić,** *redoviti profesor u trajnom zvanju Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Prof. Dr. sc. Marko Rosić,** *redoviti profesor u trajnom zvanju Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Doc. Dr. sc. Jelena Nakić, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Mirna Marić, *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: **rujan 2024.**

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of Computer Science
Ruđera Boškovića 33, 21000 Split, Croatia

Development of an education platform based on MERN

Anđela Ivanišević

ABSTRACT

In this thesis, key concepts related to the MERN stack are described. MERN stack is a popular technology consisting of MongoDB (database), Express.js (server framework), React (frontend), and Node.js (backend). The paper outlines the key functionalities of the educational platform, such as course overview, lesson review, messaging, and file storage.

Key words: web application, React, Node.js, MongoDB, Express

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 29 pages, 19 figures, 0 tables and 7 references

Original language: Croatian

Mentor: **Marko Rosić, Ph.D.,** *Professor of Faculty of Science, University of Split*

Reviewers: **Marko Rosić, Ph.D.,** *Professor of Faculty of Science, University of Split*

Jelena Nakić, Ph.D. *Assistant Professor of Faculty of Science, University of Split*

Mirna Marić, Ph.D. *Assistant Professor of Faculty of Science, University of Split*

Thesis accepted: **September 2024**.

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom Izrada edukacijske platforme temeljene na MERN-u izradila samostalno pod voditeljstvom prof. dr. sc. Marka Rosića. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući navela u završnom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Studentica

Anđela Ivanišević

SADRŽAJ

1. UVOD

U suvremenom društvu, Internet predstavlja neizostavni aspekt svakodnevnog života, a njegova upotreba kontinuirano raste. Internet omogućuje velik broj raznih informacija i sadržaja te povezuje milijarde korisnika širom svijeta. Prisutnost interneta u našoj svakodnevnici je neosporiva, a samim razvojem dolazimo do sve većeg broja različitih internetskih stranica i web aplikacija. Unatoč činjenici da korisnici redovito posjećuju ove platforme većina korisnika nema uvid u složenost samog procesa izrade web aplikacija.

Web aplikacije su programska rješenja kojima se pristupa putem interneta, a temelje se najčešće na komunikaciji između klijenta i servera.

Komunikacija između klijenta i servera obavlja se najčešće u ovih nekoliko jednostavnih koraka. Za početak korisnik odnosno klijent šalje neki zahtjev na server, nakon čega server prima zahtjev te ga obrađuje. Nakon što server obradi neki zahtjev server šalje odgovor na postavljeni zahtjev odnosno upit ili grešku u izvršavanju zahtjeva. Greška u zahtjevu se može dogoditi na klijentskoj strani, ukoliko korisnik nema pravo pristupa nekoj informaciji ili sadržaju ili server nije mogao pronaći traženi resurs, te na server strani što obuhvaća neku grešku u kodu, preopterećenost servera i slično.

Većina web aplikacija je najčešće razvijena u okruženjima JavaScripta, HTML-a, XML, Reacta i slično. Ovaj završni rad se sastoji od praktičnog i teoretskog dijela. U ovom radu opisati ću sam proces izrade aplikacije korištenjem MERN (MongoDB, Express, React i Node.js) tehnologije.

Prije same izrade završnog rada proučavala sam različite tehnologije pomoću kojih je moguće izraditi web aplikaciju, no odabrala sam upravo MERN tehnologiju zbog praktičnog iskustva kojeg ću steći izradom edukacijske aplikacije i zbog njegove same popularnosti i široke primjene. Do same izrade završnog rada, imala sam vrlo malo iskustva rada s MERN tehnologijom, smatram da mi je ovaj projekt i završni rad omogućio da proširim svoje znanje i razvijem vještine koje ću moći koristiti u budućnosti i koje će mi uvelike pomoći.

Također, uz izradu samih funkcionalnosti trebalo je razmišljati i o potencijalnim korisnicima te aplikacije. Intuitivnost znači da će korisnici na jednostavan način koristiti aplikaciju i lako se snalaziti bez većih potreba za uputama. Na taj način se postiže bolje korisničko iskustvo što je od velike važnosti. Korištenjem MERN tehnologije, odnosno Reacta, omogućeno je kreiranje

intuitivnih korisničkih sučelja. Elementi su napravljeni na način da budu lako prepoznatljivi i razumljivi korištenjem izbornika i dugmadi s kojom se većina korisnika redovito susreće.

2. WEB APLIKACIJA

Web aplikacija je aplikacija koja radi na internetu ili intranetu koristeći web preglednik. Drugim riječima, web aplikacija je aplikacija napisana na web jeziku (kao što su HTML, JavaScript, Java itd.) koju treba pokrenuti kroz preglednik [Alfred, 2019]. Većina web aplikacija se sastoji od tri glavne komponente, tj. tri sloja – prezentacijski, poslovni i podatkovni.

Prezentacijski sloj predstavlja onaj dio koji korisnik vidi dok koristi web aplikaciju ili stranicu. Za izradu prezentacijskog sloja programeri koriste HTML, CSS i JavaScript. U HTML dijelu određujemo kako će stranica izgledati, odnosno koje će sve elemente sadržavati i kako će oni biti raspoređeni na stranici. Pomoću CSS dijela određujemo izgled elemenata koje smo odredili u HTML dijelu. Pravila unutar CSS određujemo pomoću selektora i deklariranjem pravila unutar vitičastih zagrada. Te za kraj, JavaScript dio omogućuje da stranica „odgovara“ na zahtjeve korisnika, to jest unutar JavaScript dijela se nalaze sve naredbe koje će program izvršavati na korisnikov zahtjev. Za dinamičnost same aplikacije koriste se okviri kao što su React, Angular, Vue.

Poslovni sloj prima korisnikove zahtjeve te ih obrađuje. Obično je ovaj sloj podijeljen na više manjih dijelova kojima je lakše upravljati. Poslovni sloj uključuje sve aplikacijske funkcionalnosti, npr. upravljanje predmetima u košarici za kupnju u online trgovini(Grozev, Buyya, 2013). Poslovni sloj ima ulogu da djeluje između korisničkog sučelja, odnosno korisnika i njegovih zahtjeva i servera.

Podatkovni sloj je sloj koji je usko povezan s poslovnim slojem te pruža pristup podacima koji se koriste u sklopu aplikacije – baza podataka. U prethodnim godina studiranja susreli smo se s bazom podataka upravljanom MYSQL sustavom i MongoDB bazom podataka. MongoDB i SQL su dva potpuno različita sustava za upravljanje bazom podataka. MongoDB je nerelacijska baza podataka što znači da se podaci unutar ove vrste baze podataka spremaju u kolekcije, a svaka kolekcija se sastoji od više dokumenata. Za razliku od nerelacijski baza, u relacijskim bazama podaci se spremaju u tablice i retke. Primjer relacijske baze su upravo baze podataka izrađene u MYSQL-u, a primjer nerelacijskih baza su one baze podataka izrađene u Mongo DB. Uz sam Mongo DB sustav valja i spomenuti Mongo DB Atlas koja ima slične funkcionalnosti kao i Mongo DB, ali je dostupna online.

U ovom radu koristiti ćemo upravo troslojnu arhitekturu, no također postoje i druge arhitekture i pristupi pomoću kojih možemo izraditi web stranicu. Neki od njih su:

1. MVC – model-view-controller
2. PWA – progresivne web aplikacije
3. MVVM – model-view-viewmodel
4. SPA

2.1 Izrada web aplikacije

U ovom radu bavit ćemo se izradom edukacijske platforme temeljene na MERN-u. Za početak, upoznat ćemo se sa samim procesom izrade web aplikacije. Proces izrade web aplikacije možemo podijeliti u dva dijela: izrada backend dijela i izrada frontend dijela (slika 1).

Backend dio zapravo predstavlja dio aplikacije koju sam korisnik ne vidi, odnosno predstavlja tehnologije koje su koriste u web razvoju za stvaranje i upravljanje web stranicom odnosno aplikacijom. Neke od ključnih komponenti i tehnologija koje se koriste za razvoj servera su: Python, JavaScript(Node.js), PHP, Java, Express.js, Django, Laravel, MySQL, MongoDB, Redis i slično.

Za razliku od backend dijela, frontend dio je vidljiv korisniku. Frontend dio predstavlja sve ono što korisnik vidi dok koristi web aplikaciju. Iako je možda za programere važnija funkcionalnost same aplikacije nego njen izgled, korisnici web aplikaciju uvijek prvo vizualno ocijene i uvjere se da je intuitivna za korištenje, te na taj način odluče hoće li ju koristiti u budućnosti ili ne. Neki od programskih jezika i „knjižica“ koje koriste frontend programeru si HTML, CSS, JavaScript, React, jQuery, Python i slično.

Također, uz backend i frontend dio, za funkcionalan rad same aplikacije potrebna je i baza podataka. Baza podataka pruža pristup podacima koje korisnik koristi u sklopu same aplikacije.

Ova tri sloja (backend,frontend i baza) zapravo čine troslojnu arhitekturu web aplikacije.

Svako od nas redovito posjećuje i koristi različite web aplikacije ne razmišljajući o načinu na koji oni zapravo rade tj. o onome što se događa u pozadini.

Komunikacija web aplikacije se događa između klijenta i servera. Klijent predstavlja korisnika te prilikom pokretanja same web aplikacije šalje zahtjev (request – PUT; POST; GET; DELETE)

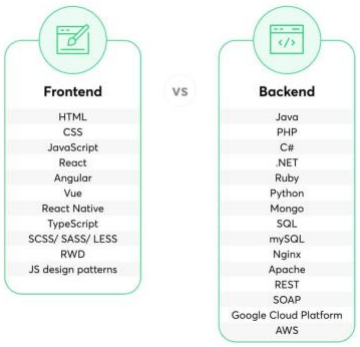
serveru koristeći neki od web protokola – HTTP ili HTTPS. Kada zahtjev dođe do servere on će ga obraditi, te nakon toga poslati odgovor klijentu (response). Nakon što server pošalje odgovor, odgovor će se ispisati klijentu na sučelje.

Get zahtjeve koristimo u slučajevima kada dohvaćamo neke podatke sa servera i prikazujemo ih korisniku. Neki od primjera kada koristimo get zahtjeve su npr. dohvaćanje informacije o nekom proizvodu, dohvaćanje nekih popisa, id podataka, poruka i slično.

Post zahtjeve koristimo kada želimo poslati neku informaciju na poslužitelja, npr. informacije o prijavi u sustav, prenijeti neku datoteku i slično.

Put zahtjevi se koriste onda kada već postoji neka informacija na serveru, a korisnik je želi promijeniti. Primjer korištenja put zahtijeva je bilo koja izmjena, npr. promjena lozinke ili korisničkog imena.

Delete zahtjeve koristimo kada želimo izbrisati neku informaciju iz baze podataka, npr. korisnik želi izbrisati neku poruku, korisnik želi izbrisati neku rezervaciju ili slično.



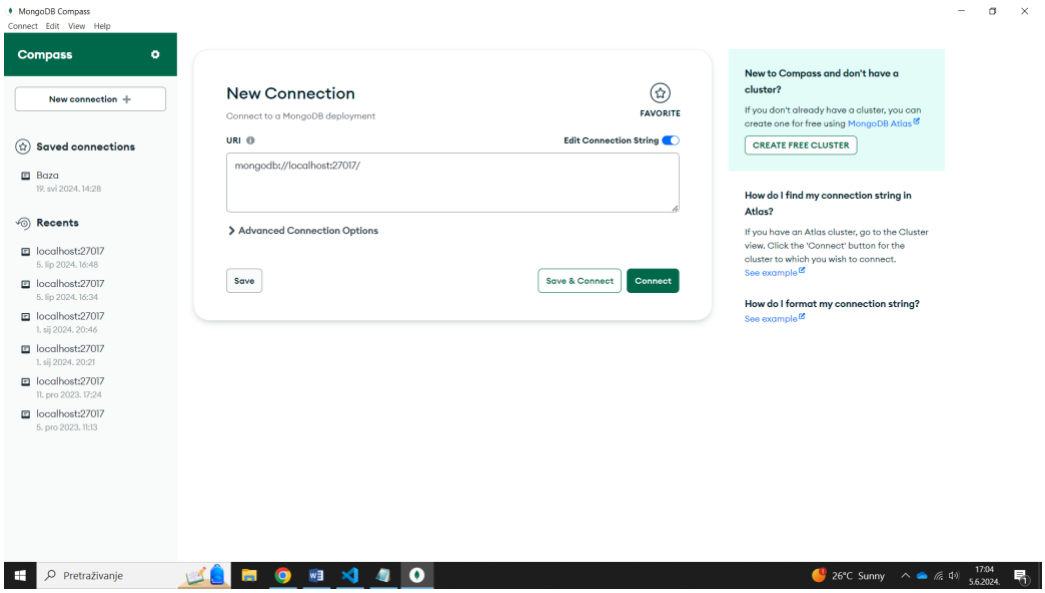
Slika 1. Backend i frontend tehnologije

Prilikom izrade web aplikacije, odabir tehnologije korištene u razvoju bio je temelj izrade. Kao tehnologiju izrade edukacijske platforme odabrala sam MERN tehnologiju. MERN predstavlja četiri dijela, odnosno tehnologije, pomoću kojih ćemo izraditi web aplikaciju. Kratica MERN predstavlja MongoDB, Express, React i Node.js.

2.2 MongoDB

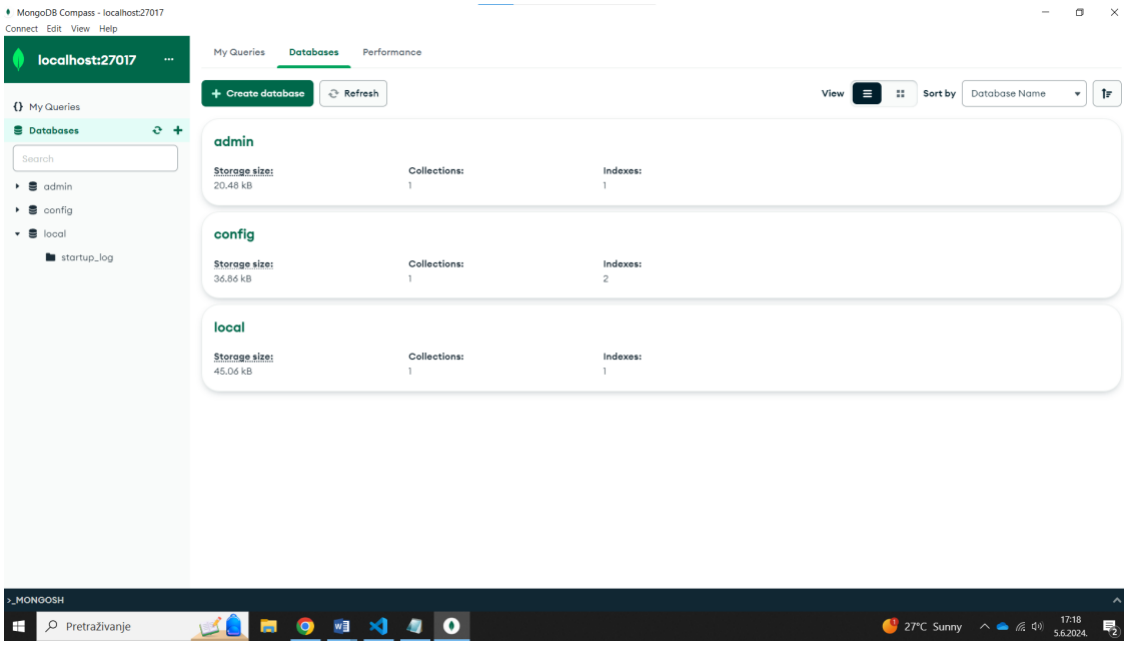
MongoDB je NoSQL baza podataka otvorenog koda koja je kreirana od tvrtke 10gen. Razvijena je u C++-u. MongoDB dokumenti su pohranjeni u binarnom obliku JSON-a pod nazivom BSON-

format. [Divya, Bansal, 2017). Za razliku od relacijskih baza podataka koje sadrže tablice i redove, MongoDB sadrži kolekcije i dokumente. Sada ćemo se ukratko upoznati s načinom rada MongoDB.



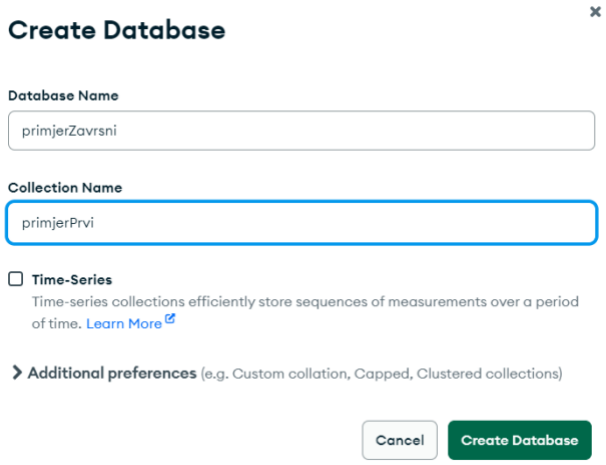
Slika 2. Početna stranica MongoDB

Kada otvorimo MongoDB sučelje aplikacije potrebno je upisati potrebnu URI adresu (ukoliko se npr. želimo povezati s bazom kreiranom u MongoDB Atlasu) ili pritisnuti connect ukoliko se želimo povezati lokalno na naše računalo (slika 2). Nakon toga, otvorit će se prozor koji već sadrži baze admin, config i login (slika 3). Unutar kartice local nazale se informacije vezane za pokretanje servera.

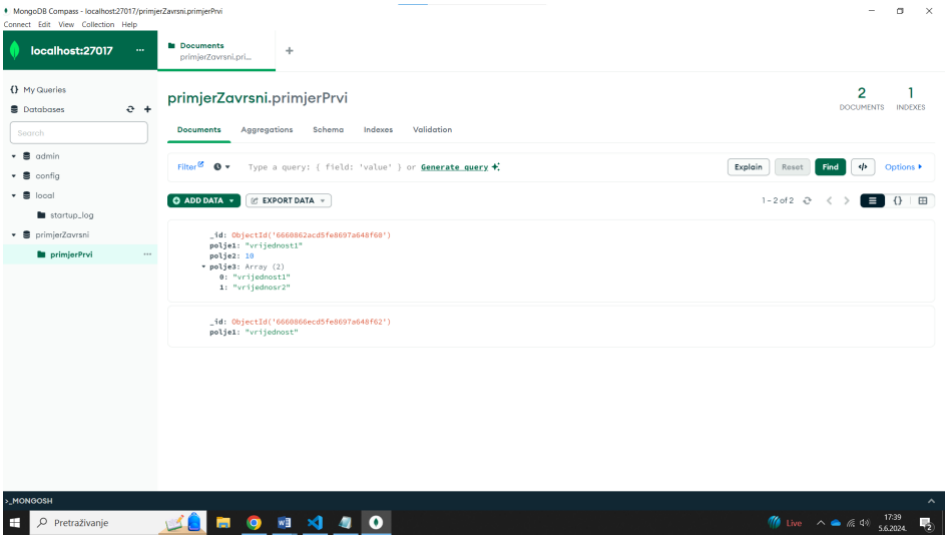


Slika 3. Sadržaj MongoDB

Sada ćemo napravi jedan jednostavan primjer da pobliže opišem i objasnim kolekcije odnosno dokumente. Novu bazu podataka kreiramo na jednostavan način – pritiskom na Create database. Nakon toga, upišemo željeni naziv naše baze i naziv prve kolekcije (slika 4).



Slika 4. Kreiranje prve tablice



Slika 5. Kolekcije i dokumenti

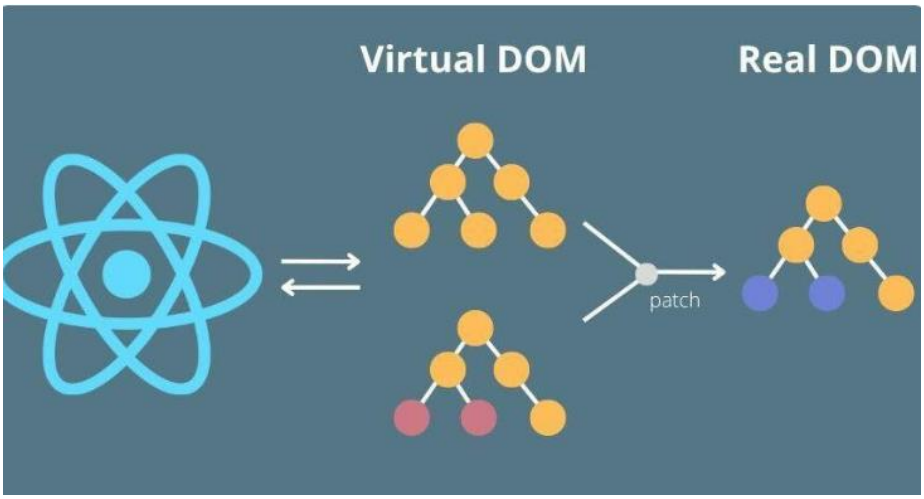
U primjeru na slici (slika 5), možemo vidjeti da smo kreirali bazu primjerZavrsni unutar koje se nalazi kolekcija primjerPrvi koja sadrži više dokumenata zapisanih u BSON formatu.

2.3 Node.js

Node.js je prvi put objavljen 2009 godine od strane Ryan Dahl-a kao reakciju na to koliko su web serveri sporo radili u to vrijeme. Node.js uključuje ugrađene HTTP biblioteke koje nam pomažu pri konfiguraciji React aplikacije i za pokretanje lokalnog razvojnog poslužitelja. Koristi se u aplikacijama koje zahtijevaju brze odgovore, kao na primjer chat i input/output operacije. Također Node.js omogućuje korištenje knjižica kao što su Moongoose koja se koristi za povezivanje i rad s MongoDB bazom podataka. On omogućuje definiranje same sheme i pruža funkcionalnosti za obavljanje CRUD operacija.

2.4 React

React je JavaScript biblioteka koja je razvijena od strane Facebook-a, ali je dostupna kao open-source kod. Funkcionira na temeljnu virtualnog DOM-a (Document Object Model). DOM je način prikaza Html dokumenta kao stabla gdje je svaki element prikazan kao čvor. Virtualni DOM funkcionira na način da se stvara „kopija“ stvarnog DOM-a i svaka promjena koja se napravi, događa se na virtualnom DOM-u. Nakon toga, virtualni DOM se uspoređuje sa stvarnim DOM-om te će React osvježiti samo one promjene koje je pronašao u usporedbi sa stvarnim DOM-om(slika 6).



Slika 6. Vizualni prikaz virtualnog DOM-a

Također, uz React aplikacije vežemo pojam React hooks. React hooks su posebne funkcije za upravljanje stanjima komponenti koje su prvi put predstavljene i demonstrirane na React Conf 2018. godine od strane Sophie Alpert, Dan Abramova i Ryan Florencea.

Hookovi omogućuju funkcijskim komponentama pristup stanjima varijabli što znači da se klasne komponente više ne koriste. U prošlosti, prije no što su uvedeni hookovi, postojala je razlika između klasnih i funkcijskih komponenti. Funkcijske komponente su se koristile samo onda kada komponenta nije treba imati nikakvu unutarnju logiku (usmjeravanje zahtjeva, vraćanje podataka, čuvanje podataka i slično).

Za razliku od funkcijskih, klasne komponente su imale mogućnost pamćenja unutarnjih stanja i izvršavanje složenijih operacija. Pojavom hookova funkcijske komponente su postale standard upravo zbog jednostavnosti same sintakse. Najčešće korišteni i najpoznatiji hook je useState. No, uz useState hook također treba spomenuti i useEffect, useNavigate, useContext, useRef, useReducer, useCallback. Sada ćemo detaljnije hook koji se najčešće koristi i s kojim sam se najviše susretala.

UseState hook se koristi za pohranu stanja komponenti. Prije korištenja same naredbe useState potrebno ju je importati iz React-a. Nakon toga deklariramo varijablu. Varijabla se sastoji od trenutnog stanja komponenti i funkcije za promjenu stanja komponenti. Na slici ispod je prikazan kod pomoću kojega uvećavamo vrijednost(slika 7). Početna vrijednost je postavljena na 0, a svakim

pritiskom na button poziva se funkcija uvecajVrijednost koja povećava stanje trenutne vrijednosti za jedan. Drugim riječima, poziv funkcije postaviVrijednost uzrokuje ponovno renderiranje komponente i postavljanje nove vrijednosti.

Slika 7. Jednostavan primjer upotrebe useState-a

Uz useState hook još jedan popularno i često upotrebljavam hook je useEffect. Primjer korištenja useEffect hooka je dohvaćanje podataka iz baze. Također, jedan od bitnih obilježja ovog hooka su upravo ovisnosti odnosno definiranje dependenies-a. Definiranjem ovisnosti određujemo kada će se kod unutar ovog hooka izvršiti. Na primjer, ukoliko u ovisnosti postavimo prazan niz, kod će se izvršiti samo prilikom prvog renderiranja komponente. Ukoliko unutar ovisnosti definiramo neku varijablu, kod će se izvršiti svaki put kada se vrijednost varijable promijeni.

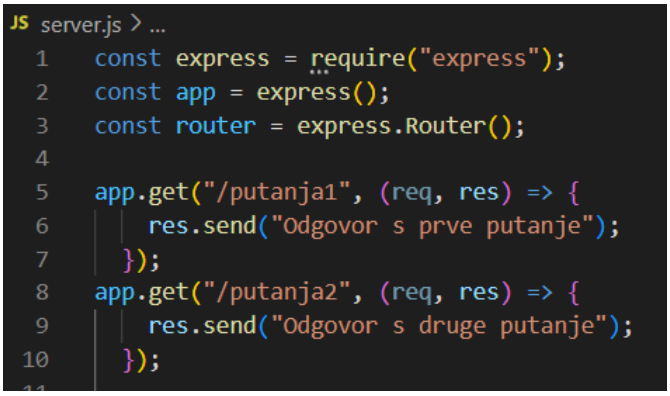
UseNavigate hook služi za preusmjeravanje korisnika na neku drugu rutu. Primjer korištenja je prijava korisnika. Nakon ispravnog unosa email adrese i lozinke, korisnika pomoću useNavigate hooka preusmjeravamo na novu rutu. Sličnu funkcionalnost kao useNavigate ima Link komponenta koja se također koristi za preusmjeravanje korisnika na različite rute definirane unutar same aplikacije. Glavna razlika između useNavigate hooka i Linka je ta što se useNavigate koristi kada želimo programski preusmjeriti korisnika nakon uspješno obavljene prijave, unosa podataka ili neke druge radnje, dok Link služi kao vidljiva navigacija korisniku između različitih komponenti aplikacije.

2.5 Express

U MERN-u, Express olakšava komunikaciju između frontenda ili React-a i backenda ili baze podataka (u ovom slučaju MongoDB). Ovaj okvir ima tri osnovne funkcije:

1. Routing

Route koristimo kako bismo definirali način na koji aplikacija odgovara na zahtjeve koji su upućeni na određenu adresu.



Slika 8. Routing

Ruta sadrži ključni pojam app (instanca za kreiranje nove express aplikacije), metodu GET, i callback funkciju koja će se pozvati kada poslužitelj dobije zahtjev za traženu putanju (slika 8).

2. Upravljanje HTTP zahtjevima

Upravljanje HTPP zahtjevima omogućuje komunikaciju između klijenta ili korisnika i servera. Osnovni HTTP zahtjevi su GET (služi za dohvat podataka) ,POST (služi za slanje podataka na poslužitelja), DELETE (služi za brisanje određenog podatka) i PUT/PATCH (služi za zamjenu ili ažuriranje postojećeg podatka). Zahtjeve najčešće šaljemo pomoću biblioteka kao što su axios i fetch. Svaki zahtjev koji klijent šalje serveru sastoji se od osnovnih metoda, URL-a, domene odnosno localhosta, broj porta kojeg server sluša i putanji koja odgovara određenoj ruti na serveru koji obrađuje zahtjev. Također, zahtjev može sadržavati i tijelo zahtjeva koje sadrži podatke koje klijent šalje serveru. Primjer tijela zahtjeva su informacije poput emaila i lozinke koje se u zahtjevu šalju serveru koji provjera točnost i uspoređuje ih s onim podacima koji se već nalaze u bazi podataka. Nakon što zahtjev pošalje odgovor koristi se then() i catch() ovisno odgovoru servera.

Catch funkcija se koristi onda kada je došli do pogreške, bilo to tijekom slanja zahtjeva ili njegove obrade, a then se koristi onda kada je server uspješno obradio korisnikov zahtjev i poslao traženi odgovor.

3. Middleware

Middleware funkcije su funkcije koje imaju pristup req i res i sljedećoj middleware funkciji. Ove funkcije su uključuju u kod pozivom metode app.use(). Middleware funkcije mogu obavljati sljedeće zadatke:

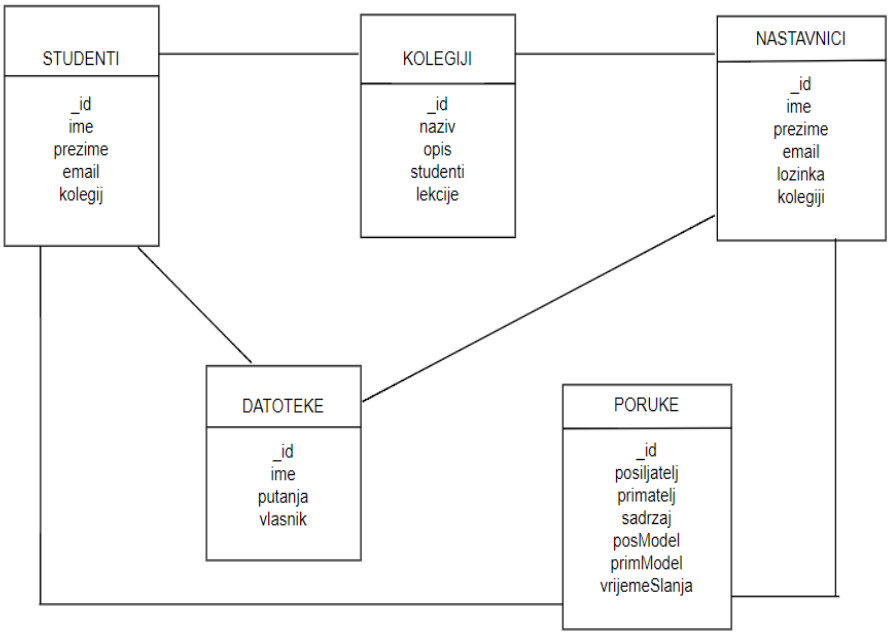
- Izvršiti bilo koji kod – izvođenje asinkronih operacija, dohvat podataka iz baze podataka ili slanje drugih zahtjeva
- Napraviti promjene na zahtjevu i odgovoru – dodavanje novih podataka koji će kasnije biti dostupni u drugim middleware funkcijama
- Završiti ciklus zahtjeva i odgovora – može završiti slanjem odgovora čime se funkcija završava i nema potrebe za pozivanjem druge middleware funkcije
- Pozvati sljedeću middleware funkciju – funkcija next() omogućuje da Express.js nastavi s obradom drugih zahtjeva tako što prenese kontrolu na drugu middleware funkciju koja se nalazi u nizu

3. RAZRADA TEME

U ovom dijelu rada proći ću kroz korake izrade odabrane web aplikacije. Ideja za praktični dio rada bila je izrada aplikacije (pomoću MERN tehnologija) koju može koristiti fakultet ili neka druga odgojno-obrazovna ustanova.

Aplikacija je osmišljena na način da ima korisničke uloge. Korisnik se može u aplikaciju prijaviti kao nastavnik ili kao student. Ukoliko je korisnik prijavljen u aplikaciji kao student, on ima mogućnost pregleda kolegija koje sluša te ima pristup lekcijama tog kolegija. Uz to, student ima mogućnost slanja poruka drugim studentima ili nastavnicima. Također, studenti i nastavnici mogu pohranjivati neke datoteke na svoj korisnički račun.

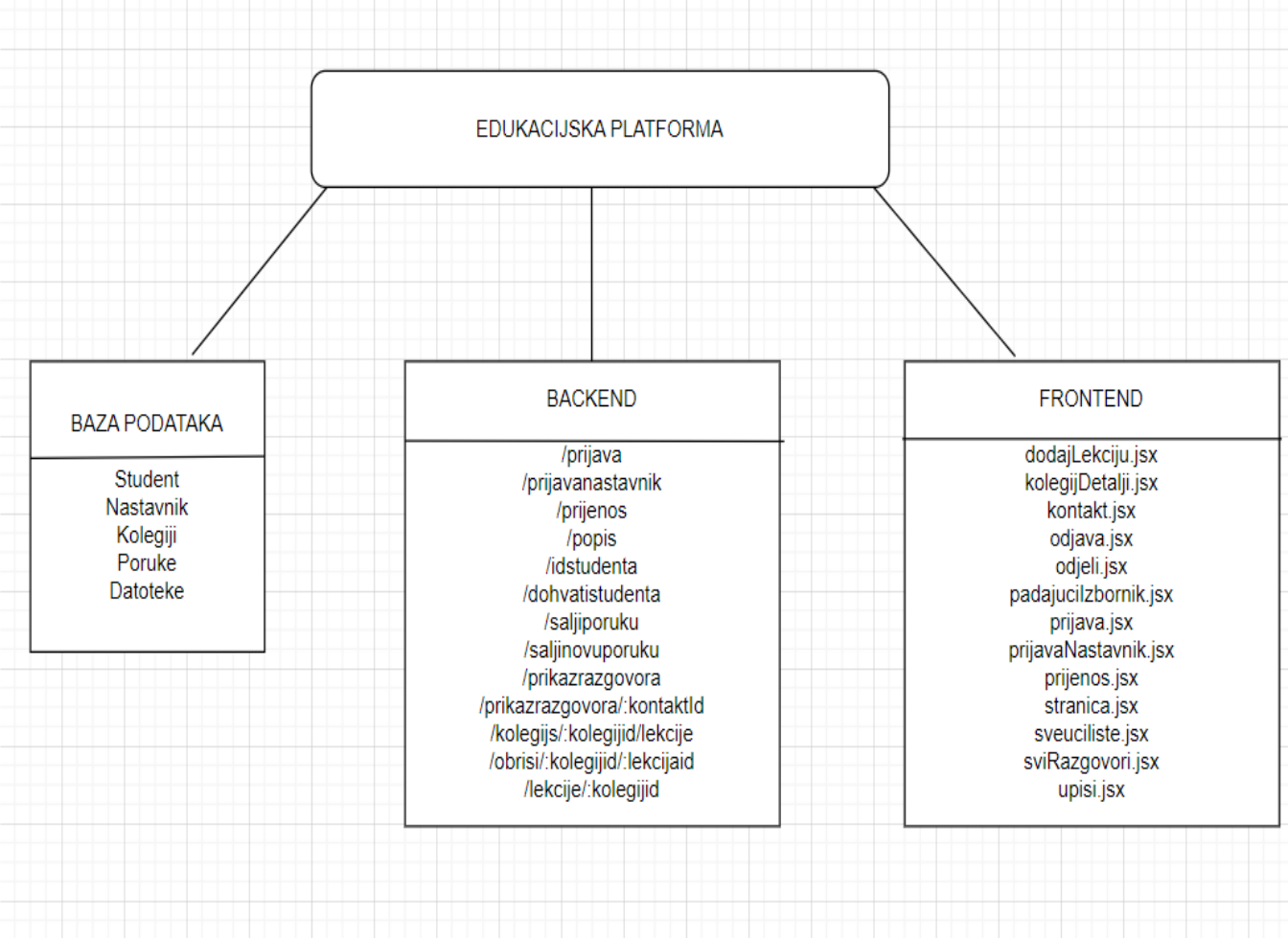
Baza podataka se sastoji od više tablica, odnosno dokumenata koji sadrže podatke o studentima i nastavnicima, kolegijima, pohranjenim datoteka i porukama koje korisnici međusobno razmjenjuju (slika 9).



Slika 9. Shema baze podataka

Uz već opisane funkcionalnosti nastavnik ima mogućnost dodavanja novih lekcija koje će biti vidljive studentima i brisanja već postojećih lekcija iz nekog kolegija. Aplikacija je osmišljena na način da bude jednostavna i intuitivna svim korisnicima koji će ju svakodnevno upotrebljavati.

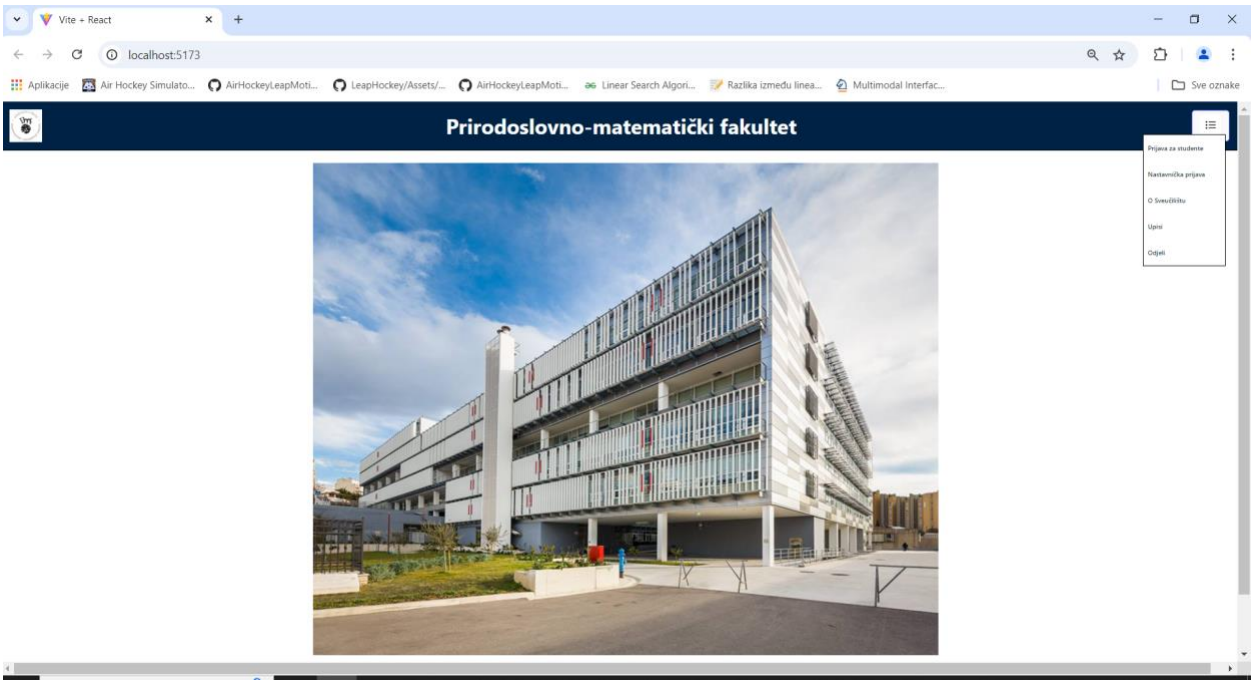
Kako bi aplikacija bila funkcionalna i upotrebljiva potrebno je da sadrži bazu podataka, server i frontend, odnosno intuitivno korisničko sučelje (slika 10) .



Slika 10. Shema aplikacije

Stranica se sastoji od tijela i zaglavalja. U tijelu stranice se samo nalazi slika ustanove koja koristi tu aplikaciju (u ovom slučaju slika Prirodoslovno-matematičkog fakulteta u Split). Zaglavlje stranice se sastoji od loga ustanove, imena ustanove te padajućeg izbornika koji nam nudi i koji

nas vodi na sve funkcionalnosti ove stranice (slika 11). Unutar padajućeg izbornika nalazi se nekoliko opcija koje nudi stranica, a to su: prijava za studente, nastavnička prijava, općenite informacije o sveučilištu, o upisima i o samim odjelima fakulteta. Nakon što sam osmislila što će naslovna strana sve nuditi, prvo sam napravila funkcionalan padajući izbornik koji vodi na daljnje stranice ovisno o korisnikovim potrebama i željama. Nakon toga, uredila sam stranicu na način da bude intuitivna i jednostavna za korištenje.



Slika 11. Naslovna stranica

3.1 Postupak prijave u aplikaciju

Nakon što je naslovna stranica razvijena, fokusirala sam se na razvoj stranice za prijavu studenta u aplikaciju, te kasnije s istim funkcionalnostima za prijavu nastavnika (samo sa različitim izgledom stranice). S obzirom da za prijavu moramo imati podatke u bazi kreirala sam sami model i shemu koja određuju strukturu dokumenta unutar MongoDB kolekcije za studente.

U bazi podataka sam izradila više kolekcija koje se odnose na studente, nastavnike, kolegije, datoteke i same poruke koje razmjenjuju. Prethodno sam već objasnila same pojmove kolekcija i dokumenata u ne relacijskim bazama podataka, te se na slici nalazi jedan primjer izrade modela i sheme za kolekciju podataka (slika 12). Polja ime, prezime, email i lozinka su tipa string te

pohranjuju osnovne podatke o studentima, dok polje kolegiji služi za povezivanje studenta s kolegijima koji se nalaze u bazi podataka. Na taj način ne moramo za svakog studenta posebno pohranjivati sve informacije o kolegiju, već unutar dokumenta samo pohranjujemo njihov jedinstveni ključ odnosno ObjectId.

```
const studentShema = new mongoose.Schema({
    ime: String,
    prezime: String,
    email: String,
    lozinka: String,
    kolegiji: [{type: mongoose.Schema.Types.ObjectId, ref: "Kolegij"}],
});

const Student = mongoose.model("Student", studentShema);
```

Slika 12. Shema izrade dokumenta

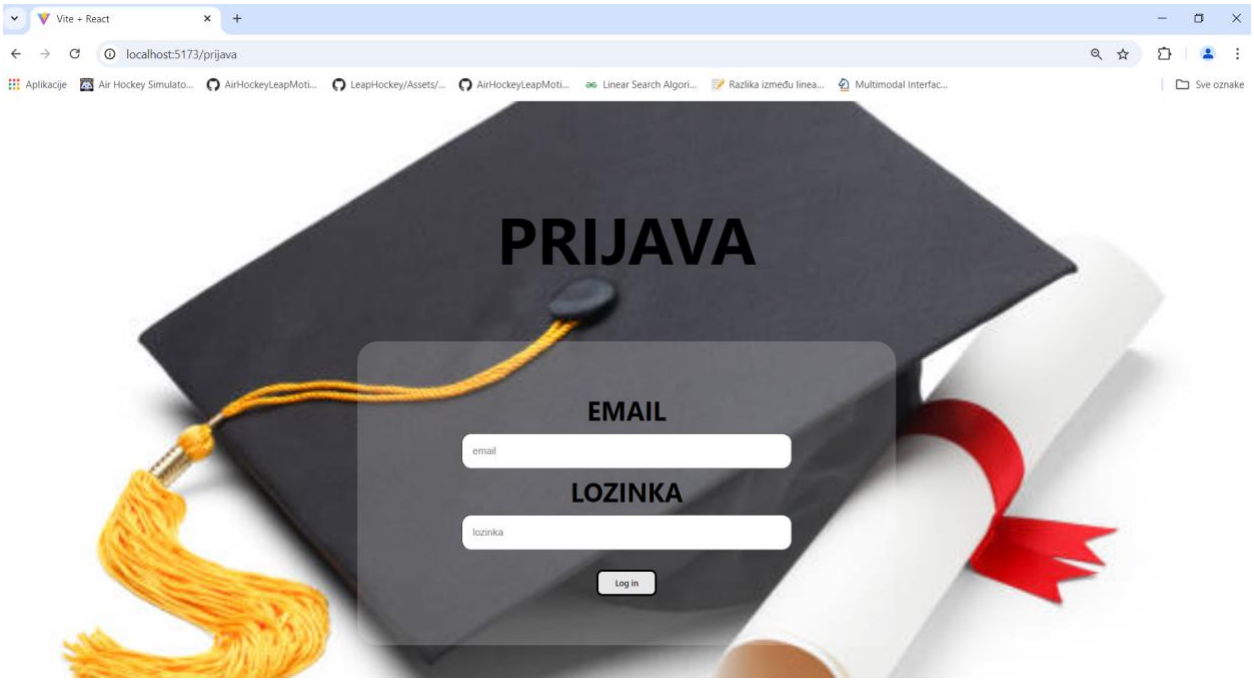
Stranica za prijavu studenta/nastavnika sastoji se od polja za unos emaila i lozinke (slika 15 i slika 16). Nakon što korisnik unese email i lozinku i pritisne dugme za prijavu, on šalje zahtjev na server. Na serveru se provjerava nalazi li se uneseni email u bazi te se nakon toga uspoređuje unesena lozinka s hashiranom lozinkom koja se nalazi u bazi (slika 14). Hashirana lozinka je zapravo lozinka koja se sastoji od jedinstvenog niza znakova. Hashirane lozinke zbog sigurnosti. Ukoliko dođe do napada na stranicu odnosno bazu podataka napadaču je dosta teško izvući stvarnu lozinku iz heshirane. U svom projektu za heshiranje lozinki koristila sam bcrypt funkciju (slika 13). Na slici ispod se nalazi primjer hashiranja lozinki. „Student1“ predstavlja stvarnu lozinku studenta, a broj 10 označava broj salt rounds-a odnosno koliko puta će se u stvarnu lozinku dodati neki slučajni podatak kako bi sama lozinka bila sigurnija.

```
const hashedLozinka1 = await bcrypt.hash("student1", 10);
const hashedLozinka2 = await bcrypt.hash("student2", 10);
const hashedLozinka3 = await bcrypt.hash("student3", 10);
const hashedLozinka4 = await bcrypt.hash("student4", 10);
const hashedLozinka5 = await bcrypt.hash("student5", 10);
```

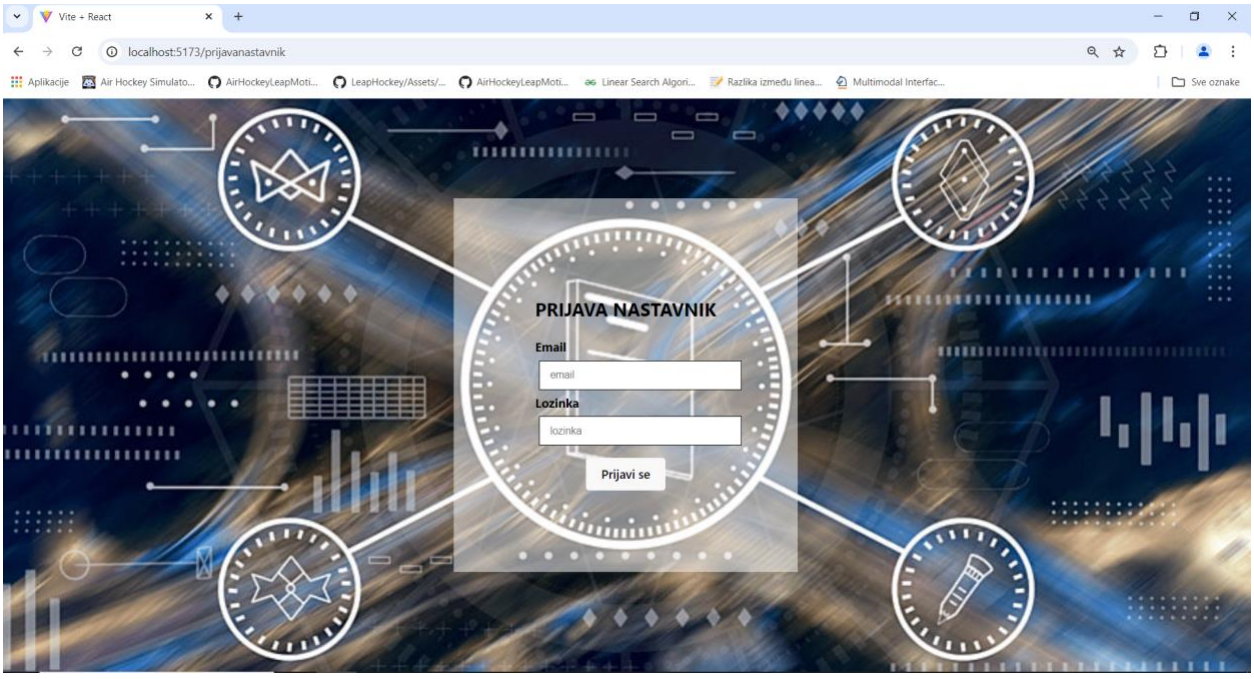
Slika 13. Hash lozinke

```
app.post("/prijava", async (req, res) => {
    const { email, lozinka } = req.body;

    try {
        const stud = await Student.findOne({ email }).populate('kolegiji');
        if (!stud) {
            return res.status(400).json({ poruka: 'Netočna lozinka ili email!' });
        }

        const tocnalozinka = await bcrypt.compare(lozinka, stud.lozinka);
        if (!tocnalozinka) {
            return res.status(400).json({ poruka: 'Netočna lozinka ili email!' });
        }

        const token = jwt.sign({ id: stud._id,role:"student" }, 'secretKey', { expiresIn: '1h' });
        //const kolegiji = await Kolegij.find({studenti: stud._id}).select('naziv opis');
        const kolegiji = await Kolegij.find({studenti: stud._id}).populate("lekcije");
        console.log(kolegiji);
        res.status(200).json({ poruka: 'Login successful', token,kolegiji, id:stud._id });
    } catch (err) {
        console.error('Greška tijekom prijave:', err);
        res.status(500).json({ poruka: 'Server error' });
    }
});
```

Slika 14. Ruta za prijavu



Slika 15. Izgled stranice za prijavu studenta



Slika 16. Izgled stranie za prijavu nastavnika

3.2 Izgled i funkcionalnosti aplikacije

Ukoliko je korisnik unio ispravne informacije, odnosno točan email i lozinku, otvara mu se stranica koja sadrži sve funkcionalnosti ove aplikacije. Za početak stranica sadrži popis kolegija koje student trenutno sluša (slika 17). Popis kolegija je prilagođen prema svakom studentu i informacija pohranjenim u bazi podataka. Popis kolegija prikazan je u obliku button elemenata. Ukoliko korisnik pritisne na button određenog kolegija otvara mu se nova stranica na kojem su prikaze same pojedinosti tog kolegija.

Također, stranica sadrži mogućnost pohranjivanja datoteka i dokumenata. Datoteke se učitavanju na način na se odabere gumb za pregled datoteka te korisnik odabere datoteku s računala koju želi prenijeti na svoj račun. Datoteka koju student ili nastavnik pohrani nije vidljiva javno, odnosno samo onaj korisnik koji ju je pohranio ima mogućnost pristupa. Datoteke se pohranjuju od najnovije prema najstarijoj te su prikazane jedna ispod druge. Datoteke se prikazuju nastavniku ili studentu u obliku liste.

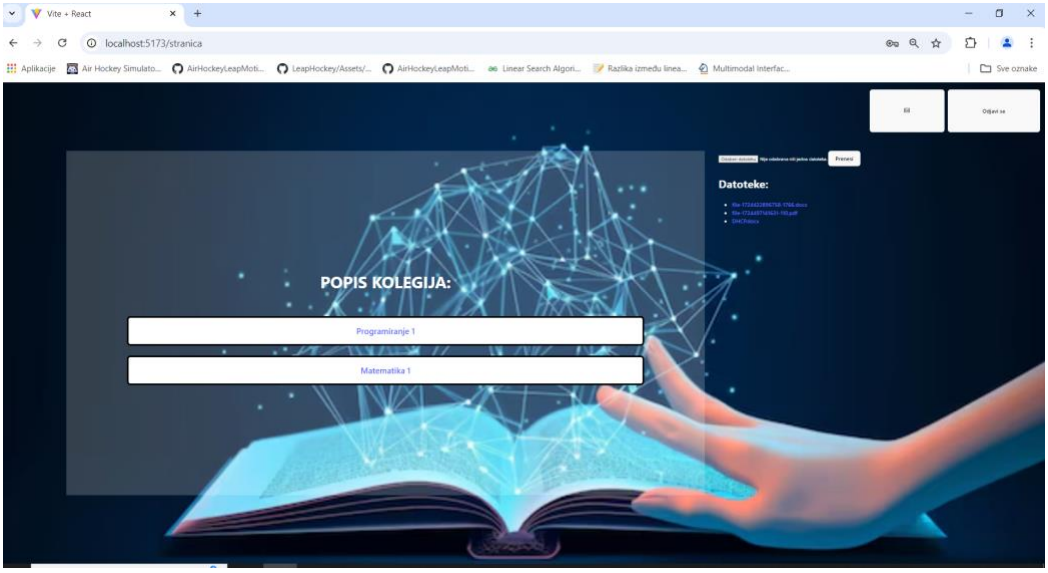
Student ili nastavnik ima opciju za slanje poruka drugom studentu ili nastavniku. Opcija za slanje poruka nalazi se u gornjem desnom kutu pored gumba za odjavu korisnika. Sučelje za slanje poruke je kreirano na jednostavan način. Sadrži dva input elementa. Prvi input element se odnosi na primatelja poruke. Korisnik u prvi input element upisuje email adresu primatelja. Drugi input

element odnosno textarea koristi se za sam sadržaj poruke. Korisnik u textBox upisuje poruku koju želi poslati.

Također, korisnik može vidjeti svoje prethodne razgovore. S lijeve strane nalaze se imena korisnika s kojima je u prošlosti komunicirao, te odabirimo nekog od njih može vidjeti prethodne poruke. Ukoliko korisnik prethodno nije vodio niti jedan razgovor, s lijeve strane umjesto popisa imena prikazuje se poruka kako korisnik trenutno nema niti jedan razgovor.

Gumb za odjavu iz sustava odnosno aplikacije nalazi se u gornjem desnom kutu. Pritiskom na odjava gumb korisnik se odjavljuje te se preusmjerava na početnu, odnosno naslovnu stranicu.

Ove ključne komponente osiguravaju da studenti i nastavnici imaju sve potrebne datoteke i informacije za praćenje nastavnog sadržaja i međusobnu komunikaciju s drugim kolegama i nastavnicima.



Slika 17. Stranica nakon uspješne prijave korisnika

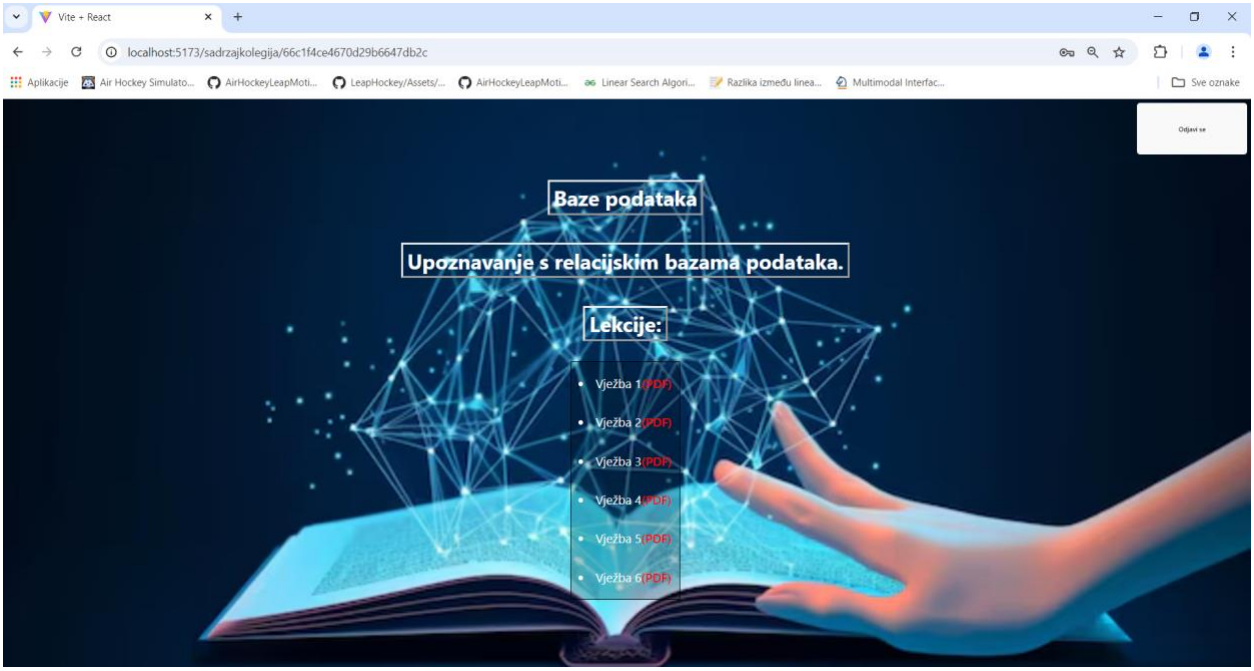
Nakon što korisnik odabere jedan od kolegija koje sluša otvori mu se stranica koja sadrži detalje tog kolegija (slika 18). Detaljni kolegija se dohvaćaju slanjem get zahtjeva na server, a tijelo zahtjeva sadrži token. Učitavanje kolegija i detalja događa se samo jednom upravo zbog korištenja useEffect hooka. Na stranici se nalaze podaci o kolegiju – naziv i opis kolegija. U slučaju da zbog pogreške kolegij nije pronađen, korisniku se ispisuje poruka na sučelju. Također stranica sadrži

naziv lekcije i pdf datoteku kojoj student može pristupiti i po želji preuzeti ju. Pdf datoteka sadrži funkcionalnost za otvaranje datoteke u novom prozoru ili na novoj kartici.

Ukoliko je korisnik u aplikaciju prijavljen kao nastavnik on ima neke dodatne funkcionalnosti kada su sami kolegiji u pitanju.

Nastavnik ima mogućnost dodavanja novih lekcija. Nastavnik prvo upisuje ime same lekcije te nakon toga odabire gumb za odabir lekcije. Nakon toga otvaramo mu se popis dokumenata i datoteka koje se nalaze na računalu i nastavnik odabire željenu lekciju. Novo dodane lekcije odmah su vidljive i nastavniku i studentima koji pohađaju taj kolegij.

Također, nastavnik ima mogućnost i brisanja lekcija. Pokraj svake lekcije, korisnik koji je prijavljen kao nastavnik, vidi gumb za brisanje lekcije. Ukoliko nastavnik odabere taj gumb lekcija se briše iz baze, stranica i popis lekcija se osvježava, a izbrisana lekcija više nije vidljiva ni studentima ni nastavnicima.



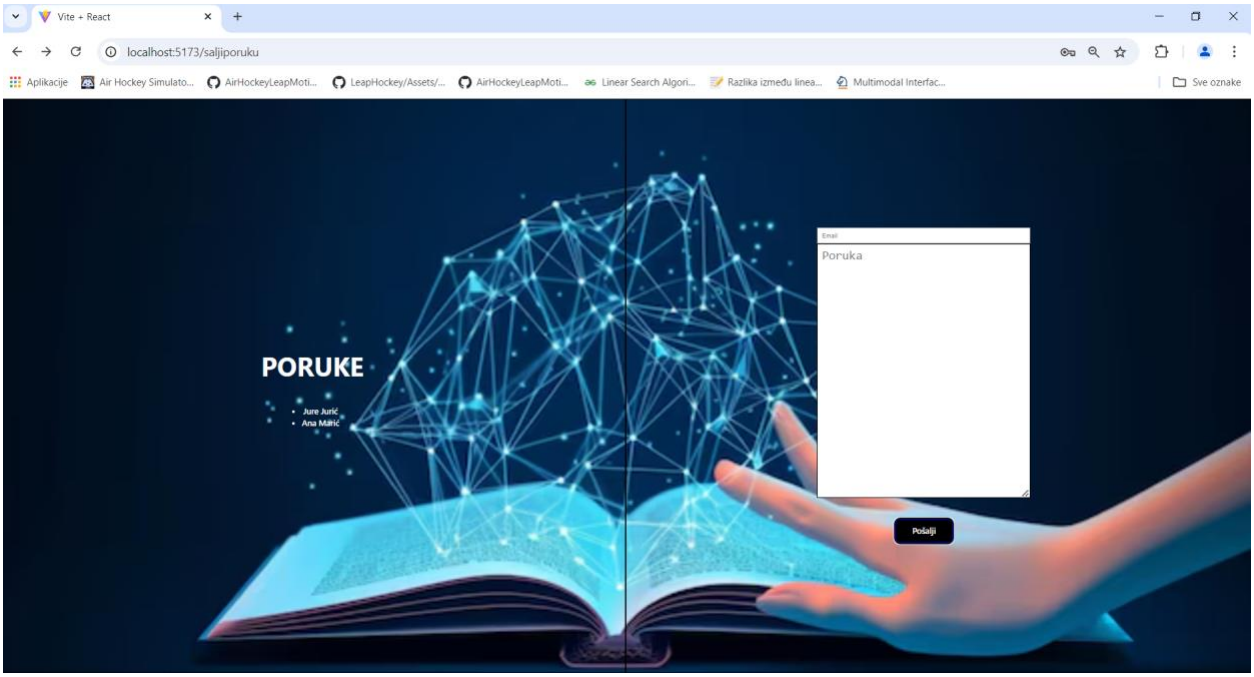
Slika 18. Detalji kolegija

Uz opciju prikaza kolegija, student ima i mogućnost komunikacije s drugim studentima i nastavnicima.

Kada student ili nastavnik otvori stranicu s poruka na server se šalje get zahtjev koji dohvaćanja korisnikove prethodne razgovore. Ukoliko je korisnik imao prethodne razgovore s lijeve strane

prikazana su imena osoba s kojima je vodio razgovore odnosno razmjenjivao poruke. Ako korisnik još nema niti jedan razgovor s lijeve strane ispisuje se poruka da još nema niti jedan razgovor. S desne strane nalazi se opcija za slanje poruka.

Nakon što korisnik upiše email primatelja i željenu poruku, odabirom gumba Pošalji poruku na server se šalje post zahtjev koji unutar tijela funkcije sadrži email primatelja i sadržaj poruke. Nakon što je poruka uspješno poslana, korisničko sučelje se osvježava i korisniku su vidljive najnovije informacije odnosno svi razgovori i poruke (slika 19) .



Slika 19. Prikaz poruka

Uz opciju za prikaz i slanje poruka i prikaz kolegija i njihovih lekcija, korisnik ima mogućnost pohranjivanja i čuvanja datoteka na svom računu.

Pomoću useEffect hooka šaljemo get zahtjev pomoću kojega dohvaćamo datoteke iz baze. U tijelu zahtjeva nalazi se token koji služi za autorizaciju. Nakon što dohvatimo datoteke sa servera, prikazujemo ih korisniku. Na taj način sve prethodno pohranjene datoteke vidljive su korisniku.

Ukoliko nastavnik ili student želi pohraniti neku novu datoteku odabere opciju Odaberi datoteku te nakon toga bira dokument ili datoteku za pohranu. Kada je obarao datoteku na server šaljemo post zahtjev. U tijelu zahtjeva šaljemo novu datoteku i token. Nakon uspješnog prijenosa datoteke se ponovno dohvaćaju i prikazuju studentu ili nastavniku.

4. ZAKLJUČAK

Informacijsko – komunikacijska tehnologija iz dana u dan napreduje sve više i više. Neke stvari s kojima se danas susrećemo i koje uzimamo kao „normalne“ bile su nezamislive prije svega nekoliko godina. Iako tehnologija ima dosta prednosti, te uvelike olakšava našu svakodnevnicu i život bez nje je postao nezamisliv, Tehnologija pruža veliku moć u današnjem svijetu te je treba znati iskoristiti na pravi način. Mogućnosti interneta i samo istraživanje raznih tehnologiju su mi uvelike pomogle u odabiru same teme. U ovom radu, predstavila sam jednu od tehnologija za izradu web aplikacija i izradila aplikaciju koja bi se mogla biti temelje aplikacije koja bi se svakodnevno upotrebljavati u nekoj obrazovnoj ustanovi. U samom uvodu objašnjeni su neki temeljeni pojmovi vezani općenito za web aplikacije. Nakon toga, opisan je sam pojam MERN tehnologije i način na koji radi. Nakon teorijskog dijela, opisan je praktični dio rada, odnosno sam proces izrade web aplikacije. Razvoj tehnologije neprekidan je proces koji zahtijeva konstantno učenje i prilagodbu.

5. LITERATURA

1. Alfred Tan Yik Ern, Web Applications, https://www.researchgate.net/publication/337224940_Web_Applications
2. Divya Chauhan, kartik Bansal: Using the Advantages of NOSQL: A Case Study on MongoDB,
3. Express, Writing middleware for use in Express apps, https://expressjs.com/en/guide/writing-middleware.html - pristupljeno 8.6.2024
4. https://www.researchgate.net/profile/Divya-Chauhan-4/publication/349110376_Using_the_Advantages_of_NOSQL_A_Case_Study_on_MongoDB/links/6021154d92851c4ed5580298/Using-the-Advantages-of-NOSQL-A-Case-Study-on-MongoDB.pdf
5. Karl Seguin: The Little MongoDB Book, https://www.openmymind.net/mongodb.pdf
6. Web Application Architecture: How the Web Works, https://www.altexsoft.com/blog/web-application-architecture-how-the-web-works/ , pristupljeno 10.6.2024
7. Web Application vs Website, https://digitalya.co/blog/web-application-vs-website/#1, pristpuljeno 31.5.2024

6. TABLICA SLIKA