BAZA ZNANJA OBRAZOVANJA U NASTAVNOM PROCESU

Marušić, Ivan

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni diplomski studij Primijenjeno računarstvo

IVAN MARUŠIĆ

ZAVRŠNI RAD

BAZA ZNANJA OBRAZOVANJA U NASTAVNOM PROCESU

Split, rujan 2024.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni diplomski studij Primijenjeno računarstvo

Predmet: Programsko inženjerstvo

ZAVRŠNI RAD

Kandidat: Ivan Marušić

Naslov rada: Baza znanja obrazovanja u nastavnom procesu

Mentor: mr. sc. Karmen Klarin, viši predavač

Split, rujan 2024.

Sadržaj

Sažetak

Ovaj završni rad opisuje izradu web aplikacije za upravljanje bazom podataka u nastavnom procesu. Cilj rada je omogućiti osoblju nastavne institucije uređivanje, upravljanje i analizu podataka učilišta te grafički prikaz analiziranih rezultata. Projekt vizualno prikazuje u kojoj mjeri studenti ostvaruju ishode učenja predmeta koje su postavili nastavnici ili učilište. S obzirom na to da većina učilišta ima vlastite web stranice za osoblje i studente, poput Moodlea, ova aplikacija dodatno ističe probleme u ishodima učenja i njihovu implementaciju kroz kolokvije, ispite i seminare.

Rad je izrađen korištenjem MERN tehnologija (MongoDB, ExpressJS, ReactJS, NodeJS) radi jednostavnog i brzog razvoja aplikacije. U radu su definirani arhitektura aplikacije, logika, funkcionalnosti i ostale postavke projekta. Detaljno su objašnjene sve tehnologije implementirane u projektu te njihove prednosti.

Ključne riječi: baza znanja, ishod učenja, MERN, tehnološki stog

Summary

Knowledge base of education in the teaching process

This final paper includes a description of the creation of a web application for database management in the teaching process. The goal of this work is to enable the staff of the educational institution to edit, manage, and analyze the data of the college and to graphically display the analyzed results. This project visually shows whether students meet the learning outcomes of the subjects as expected by the teachers/school. As most universities have their own websites for their staff and students, for example, Moodle, this application takes it to the next level by identifying problems in learning outcomes and their implementation through colloquia, exams, and seminars.

The entire work was created using the MERN (MongoDB, ExpressJS, ReactJS, NodeJS) technology stack for simple and fast application development. The paper defines the architecture of the developed application, the application logic, functionality, and other project settings. In detail, I will explain each technology implemented in the project and its benefits.

Keywords: knowledge base, learning outcome, MERN, technology stack

1. Uvod

U današnjem tehnološkom svijetu, digitalizacija je zahvatila svaki sektor društva. Svake godine nova tehnološka napredovanja omogućavaju daljnju automatizaciju i unapređivanje tradicionalnih metoda rada. Digitalna transformacija ključna je za implementaciju ovih promjena jer predstavlja poboljšanje procesa i praksi korištenjem tehnologija i podataka. Nakon nekoliko godina, softveri implementirani u edukacijskom sektoru postaju zastarjeli ili se ne koriste do punog potencijala.

Tijekom studiranja na Sveučilišnom odjelu za stručne studije (SOSS) Sveučilišta u Splitu, korištena je aplikacija Moodle za svakodnevne studentske aktivnosti. Aplikacija je omogućavala prijave predmeta, pristup materijalima, polaganje ispita i druge aktivnosti. Također je osiguravala izravan pristup svim važnim studentskim aplikacijama, kao što su *webmail* za studente i prijava za praksu. Moodle je bio dostupan i nastavnicima te zaposlenicima fakulteta. Nastavnici su imali izravan pristup predmetima u kojima su nositelji ili suradnici, s mogućnošću uređivanja svih materijala i izravnog slanja poruka studentima upisanima na predmet. Najvažnija funkcionalnost koju su nastavnici koristili bila je pohranjivanje i procjena ocjena, s arhivom rezultata ispita, ocjena i prosječnih podataka studenata, uključujući ukupne rezultate i prosjeke studenata po predmetu.

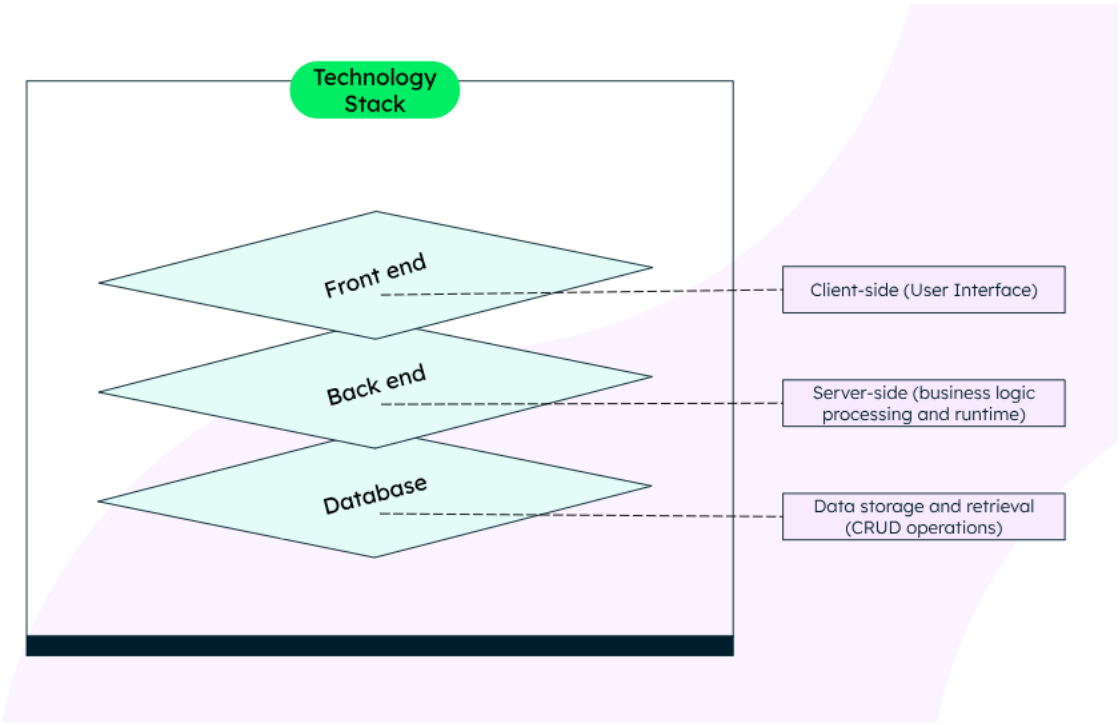
Ovaj projekt ide korak dalje jer vizualno prikazuje statističke rezultate ispita studenata u odnosu na ishode učenja. Svaki predmet ima svoje ishode učenja, koji predstavljaju očekivanja o znanju, razumijevanju i vrednovanju koje student mora ispuniti. Ovi kriteriji moraju biti zadovoljeni da bi student uspješno položio predmet, a provjeravaju se putem ispita, usmenih provjera, projekata ili kolokvija. Projekt povezuje ispite i pitanja s ishodima učenja, omogućujući profesorima da uoče i unaprijede dijelove gdje studenti zaostaju ili imaju nedostatke u znanju. Glavni cilj aplikacije je grafički prikazati korelaciju između rezultata na pitanjima i ishodima učenja u ispitima i kolokvijima te omogućiti profesorima da komentiraju rezultate za buduće potrebe.

Ovaj projekt je kompletna aplikacija za sve korisnike na sveučilištu, uključujući administrativno osoblje, nastavnike i studente, pri čemu svaki korisnik ima svoje ovlasti u sustavu. Sljedeći predlošci detaljno predstavljaju dodatne funkcionalnosti projekta.

Rad je podijeljen u nekoliko poglavlja radi boljeg razumijevanja i objašnjenja procesa razvoja. U prvom poglavlju, „Korištene tehnologije i alati“, objašnjena je sva relevantna tehnološka oprema u projektu. U drugom poglavlju objašnjene su funkcionalnosti projekta u dubokim detaljima. Pojasnit će se svaki korak i interakcija korisnika unutar sučelja projekta. U trećem poglavlju detaljno se razjašnjava implementacija projekta i unutarnje funkcionalnosti najvažnijih dijelova kôda koji se koriste u projektu, a koji su ključni za osnovne funkcionalnosti rada. U posljednjem poglavlju sažeto je opisan osnovni cilj projekta i zaključci doneseni tijekom rada na projektu.

2. Korištene tehnologije i alati

Prije početka rada na projektu, bilo je potrebno istražiti i odabrati tehnologije koje će zadovoljiti sve zahtjeve. Za razvoj *fullstack* web aplikacije, odabran je tehnološki stog (engl. *stack*), koji predstavlja skup tehnologija za izgradnju web, mobilnih ili sličnih aplikacija. Tijekom odabira tehnologija za tehnološki stog, uzeta je u obzir njihova kompatibilnost i funkcionalnost. Da bi tehnološki stog bio učinkovit, mora pružati besprijekorno korisničko iskustvo te također biti skalabilan i isplativ za korištenje. Tehnološki stog tipično se sastoji od *frontend*, *backend* i baze podataka, zbog čega se naziva puni tehnološki stog (engl. *full technology stack*). Na slici 1 prikazana je tipična arhitektura tehnološkog stoga [1].



Slika 1: Primjer arhitekture tehnološkog stoga [2]

Uobičajeno, u *frontend* ili korisničkom sučelju (engl. *user interface*) koriste se tehnologije poput HTML-a (engl. *Hypertext Markup Language*), CSS-a (engl. *Cascading Style Sheets*) i JavaScripta. Ovisno o potrebama projekta, dostupne su biblioteke (engl. *libraries*) i okviri (engl. *frameworks*) poput Reacta ili Angulara, koji su dizajnirani za razvoj korisničkog sučelja.

Backend se sastoji od poslužitelja koji obavlja svu logiku aplikacije. Najčešće je napisan u jednom ili više programskih jezika, poput JavaScripta, Jave ili Pythona, ili koristi okvire (engl. *frameworks*) kao što su Node.js ili JRE (engl. *Java Runtime Environment*).

Baza podataka služi za pohranu podataka aplikacije. Podaci se mogu pohraniti u tabličnoj strukturi (relacijska baza podataka) ili u nestrukturiranom obliku, poput NoSQL-a (engl. *Not only Structured Query Language*), koji koristi dokumentne ili grafičke strukture. Tipične baze podataka su MongoDB, Oracle i MySQL (engl. *My Structured Query Language*).

Pri odabiru tehnološkog stoga, moguće je sastaviti ga prema specifičnim zahtjevima ili koristiti već postojeće rješenje. Potrebno je uzeti u obzir krivulju učenja tehnologija i programskih jezika koji se koriste. Također je važno provjeriti kompatibilnost tih tehnologija za izradu *fullstack* aplikacija, kao i njihovu učinkovitost i jednostavnost. Nakon analize ponuđenih tehnologija i njihovih funkcionalnosti, odabran je MERN stog [2].

2.1 Visual Studio Code

Za kodiranje ili implementaciju projekta potrebno je odabrati *open-source* editor kôda. U ovom slučaju korišten je Visual Studio Code, poznat i kao VS Code. Visual Studio Code je jedan od najpopularnijih editora kôda na svijetu, razvijen od strane Microsofta. Ovaj editor nudi brojne funkcionalnosti i integraciju s različitim programskim jezicima i razvojnim alatima. Podržava širok spektar programskih jezika, a za one koji nisu podržani mogu se instalirati ekstenzije. Također omogućuje provjeru sintakse kôda i debugiranje pogrešaka. Integriran je s Gitom, sustavom za upravljanje izvornim kodom. Ima moćan CLI (engl. *Command Line*

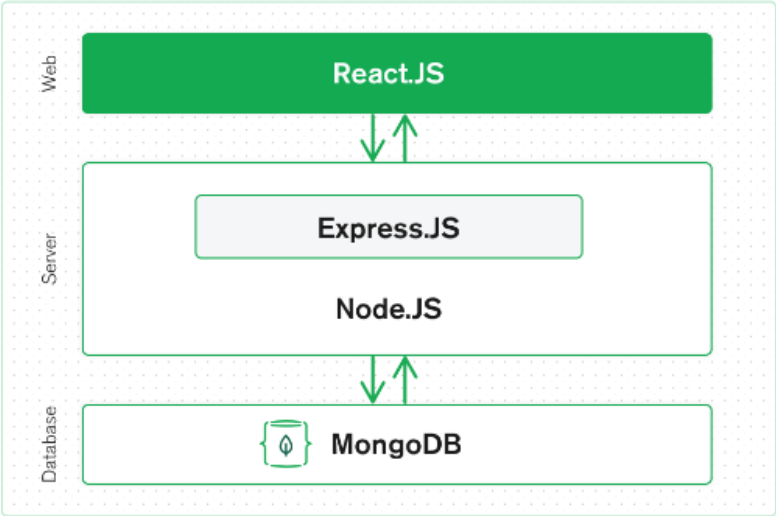
Interface) koji omogućuje kontrolu nad pokretanjem editora, kao što su otvaranje datoteka, instaliranje ekstenzija i dijagnosticiranje informacija [3].

2.2 MERN stog

MERN stog je skup unaprijed odabranih tehnologija koje pomažu developerima u izgradnji robusnih i skalabilnih web aplikacija koristeći programski jezik JavaScript. Akronim „MERN“ označava četiri ključne tehnologije koje čine ovaj stog: MongoDB, ExpressJS, ReactJS i NodeJS. Svaka komponenta ima specifičnu ulogu u razvoju web aplikacije i bazirana je na JavaScript tehnologiji. Komponente MERN stoga su:

- MongoDB – baza podataka zasnovana na dokumentima
- Express(.js) – Node.js web okvir
- React(.js) – korisnički JavaScript okvir
- Node(.js) – JavaScript web poslužitelj

Express i Node čine srednji sloj aplikacije. Express je *web framework* za poslužitelja, dok je Node popularna i moćna JavaScript poslužitelj platforma. Na slici 2 prikazan je primjer arhitekture tehnologija u MERN stogu [2].



Slika 2: Arhitekture tehnologija u MERN stogu [2]

2.3 React.js frontend

React.js je JavaScript *framework* za kreiranje dinamičnih *client-side* aplikacija u HTML-u i nalazi se na najvišem sloju MERN stoga. React služi za izgradnju kompleksnih sučelja pomoću jednostavnih komponenti, povezujući ih s podacima s *backend* poslužitelja i prikazujući ih kao HTML. React je odličan za rukovanje sučeljima uz minimalan kôd i složenost, te ima sve karakteristike modernog *web framework*. [4].

2.4 Razina Express.js i Node.js poslužitelja

Sljedeća razina u arhitekturi je Express.js *framework*, koji se pokreće unutar Node.js poslužitelja. Express.js je brz, jednostavan i minimalan web framework za Node.js. Ima moćne modele za URL (engl. *Uniform Resource Locator)* rutiranje, koji povezuju dolazne URL-ove s

funkcijama na poslužitelju i upravljaju HTTP (engl. *Hypertext Transfer Protocol)* zahtjevima i odgovorima. Svaki HTTP zahtjev s React *frontend* povezuje se s Express.js funkcijama koje napajaju aplikaciju. Express.js funkcije koriste MongoDB za pristup i uređivanje podataka u MongoDB bazi podataka. [2].

2.5 Razina MongoDB baze podataka

MongoDB je dokumentna baza podataka koja pruža skalabilnost i fleksibilnost za upite i indeksiranje. Nudi opciju skladištenja u oblaku ili na lokalnom poslužitelju. Jednostavan je za učenje i korištenje te omogućava ispunjavanje složenih zahtjeva na bilo kojoj skali. MongoDB skladišti podatke u fleksibilnim JSON (engl. *Javascript Object Notation*) dokumentima. Ovaj format omogućava da svaki dokument ima različita polja i da se struktura dokumenta može mijenjati tijekom vremena. JSON dokument se mapira na entitet objekta u aplikacijskom kodu. Svaki JSON dokument kreira se u React.js *frontendu* te se prosljeđuje Express.js poslužitelju, gdje se obrađuje i skladišti direktno u MongoDB za buduće potrebe. [5].

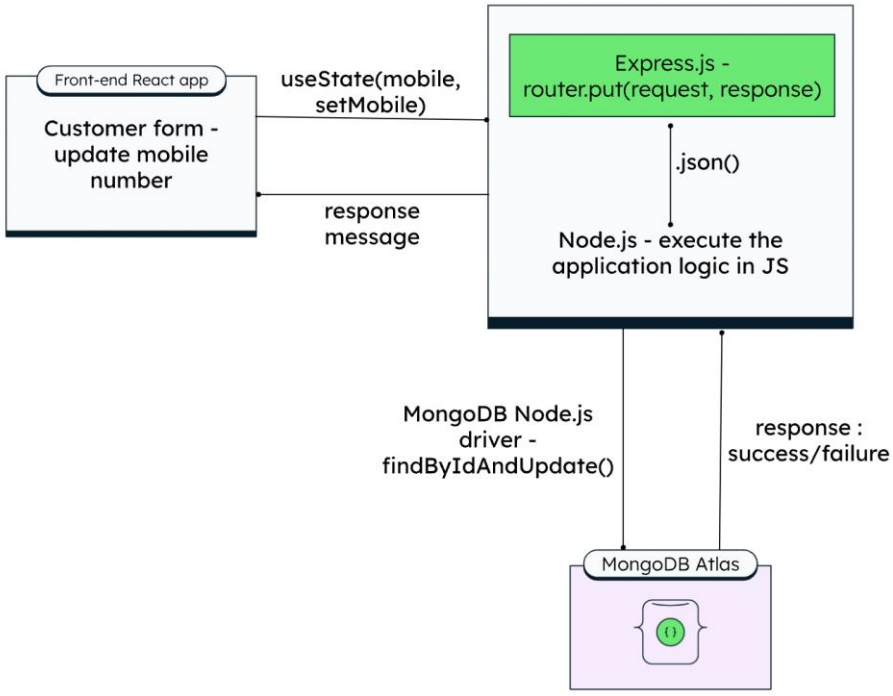
2.6 Jednostavan proces zahtjeva i odgovora u MERN stogu

Za potpuno razumijevanje kako sve četiri tehnologije u MERN stogu komuniciraju i rade zajedno, potrebno je pregledati tipičan primjer uporabe. Ovaj odlomak objašnjava primjer HTTP zahtjeva i odgovora u MERN stogu.

HTTP koristi jedan od četiri osnovna zahtjeva – POST, GET, PUT, DELETE – koji su povezani s osnovnim operacijama baze podataka: stvaranje, čitanje, ažuriranje i brisanje. Da bi se udovoljilo tim zahtjevima, Express.js pruža objekte zahtjeva i odgovora koji pohranjuju

parametre. HTTP zahtjev pohranjuje podatke od korisnika, dok HTTP odgovor pohranjuje podatke preuzete iz baze podataka. Glavna značajka MERN-a stoga je što se podaci kroz sve tehnologije pohranjuju u istom formatu, što eliminira potrebu za konverzijom prilikom prijelaza između slojeva.

Na primjer, ako korisnik želi promijeniti adresu putem online portala, *frontend* izrađen u React.js imat će obrazac za unos nove adrese. React.js zatim postavlja uneseni podatak kao parametar zahtjeva. Express.js kôd preuzima tu adresu iz HTTP zahtjeva, mapira je na podatkovni model i primjenjuje odgovarajuću metodu za izvršenje operacije ažuriranja [2]. Na slici 3 prikazana je putanja zahtjeva korisnika kroz MERN stog.



Slika 3: Putanje zahtjeva kroz MERN stog [2]

3. Funkcionalnosti aplikacije

Definirani su svi alati i tehnologije implementirani u projektu, a sada se prelazi na funkcionalnosti web aplikacije. U ovom poglavlju fokus će biti na značajkama i sposobnostima aplikacije iz perspektive korisnika. Opisat će se glavne funkcije projekta, uključujući prijavu, unos podataka, grafičke vizualizacije itd.

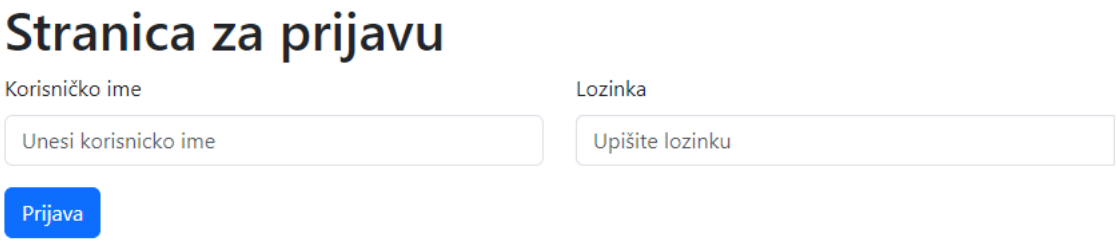
Za opis rada aplikacije potrebno je razmotriti različite korisničke uloge. U ovom slučaju, korisnici su podijeljeni u tri uloge: administrator, profesor i student. Pružit će se pregled primjernih funkcionalnosti aplikacije prema ulogama korisnika i njihovim specifičnim web sučeljima.

3.1 Funkcionalnost zasnovana na ulogama

Ovo potpoglavlje obuhvaća sva korisnička sučelja podijeljena prema ulogama. Počet će se s najvažnijom ulogom, administratorom, zatim će se prijeći na profesora, a na kraju na studenta. Detaljno će se razmotriti sučelja svake korisničke uloge, njihove funkcionalnosti i doprinos projektu.

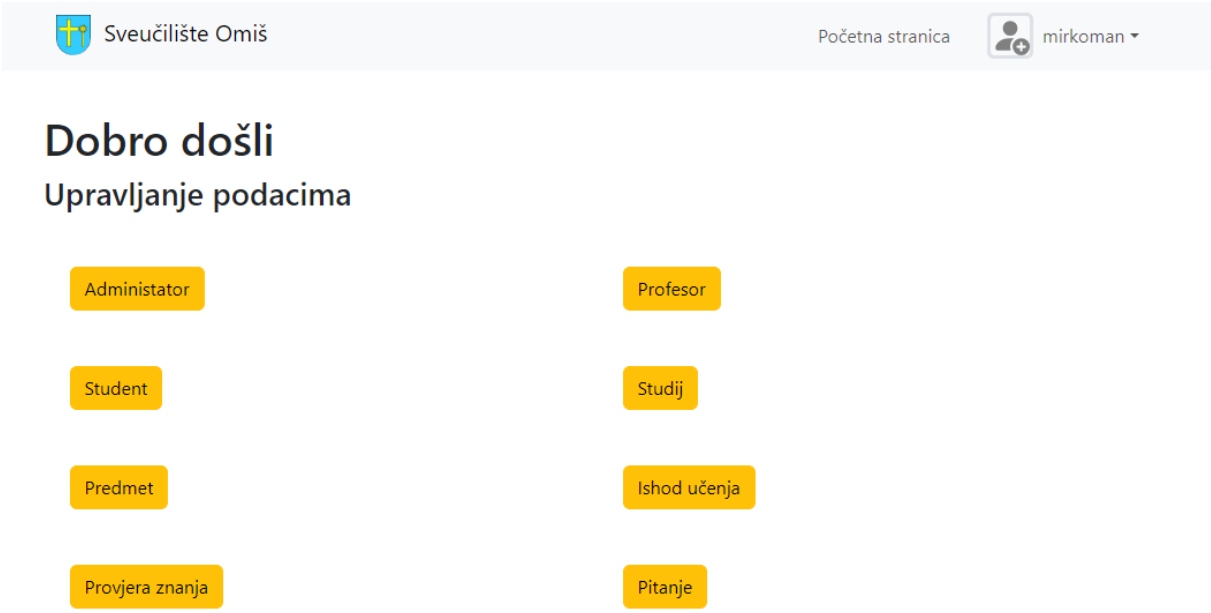
3.2 Stranica za prijavu

Bez obzira na ulogu, svaki korisnik mora se prvo prijaviti u web aplikaciju radi autentifikacije i autorizacije. Na slici 4 prikazan je izgled stranice za prijavu. Korisnik unosi svoje korisničko ime i lozinku povezanu s tim imenom. U slučaju pogrešnog unosa podataka, prikazuje se privremeni *pop-up* s porukom o neuspješnoj prijavi.



Slika 4: Sučelje za prijavu

Kada se korisniku potvrdi uloga i potrebne autorizacije, prosljeđuje se na nadzornu ploču. U ovom slučaju, prijavljuje se kao administrator. Administrator ima pristup svim funkcionalnostima aplikacije, može uređivati, pisati, dodavati i brisati sve entitete u bazi. Na slici 5 prikazana je početna stranica administratora.



Slika 5: Administratorska početna stranica

Na početnoj stranici administratora nalazi se navigacijska komponenta koja prikazuje ime sveučilišta i korisničko ime prijavljene osobe, u ovom slučaju 'mirkoman'. U navigaciji, logo sveučilišta služi kao poveznica za povratak na početnu stranicu aplikacije. Na drugom kraju navigacije nalazi se ime korisnika s *dropdown* komponentom putem koje se korisnik može odjaviti iz aplikacije, čime se brišu svi tokeni i spremljeni podaci u web pregledniku.

Ispod navigacije nalazi se tabela s osam opcija za pregled ili uređivanje podataka odabranog entiteta. Ovi entiteti su: administrator, profesor, student, studij, predmet, ishod učenja, provjera znanja i pitanje. Svaki entitet ima svoju ulogu:

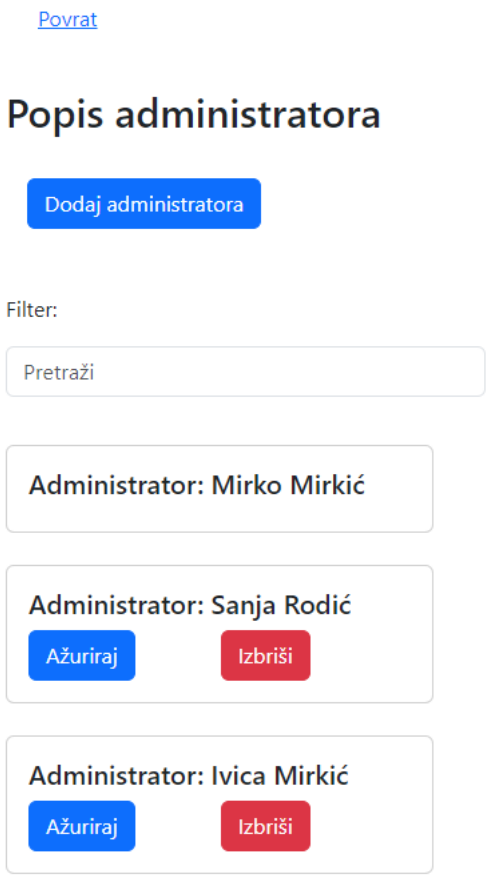
- **Administrator** – osoba s potpunim pristupom web aplikaciji.
- **Profesor** – nositelj ili suradnik predmeta s pristupom svim materijalima, ispitima i studentima upisanim na kolegij.
- **Student** – osoba s najmanjim pravima u aplikaciji, s pristupom svom predmetu i materijalima unutar predmeta (npr. kolokviji, ispiti, seminari).
- **Studij** – smjerovi na fakultetu, pri čemu svaki smjer ima svoje predmete.
- **Predmet** – specifična tema ili oblast studija koja se podučava, s materijalima povezanima s izabranim smjerom.
- **Ishod učenja** – konkretne izjave koje opisuju što studenti trebaju znati, razumjeti i biti sposobni raditi nakon završetka određenog obrazovnog programa.
- **Provjera znanja** – način provjere znanja studenata, koliko su savladali gradivo i kakve su ishode učenja ostvarili.
- **Pitanje** – izjava ili izraz kojim se traži informacija, odgovor ili pojašnjenje.

Svi entiteti i njihove funkcije bit će detaljno razmotreni.

3.2.1 Stranica administratora

Kada se odabere komponenta za administraciju, korisnik će biti preusmjeren na stranicu s popisom administratora. Na slici 6 prikazan je izgled sučelja popisa administratora s različitim

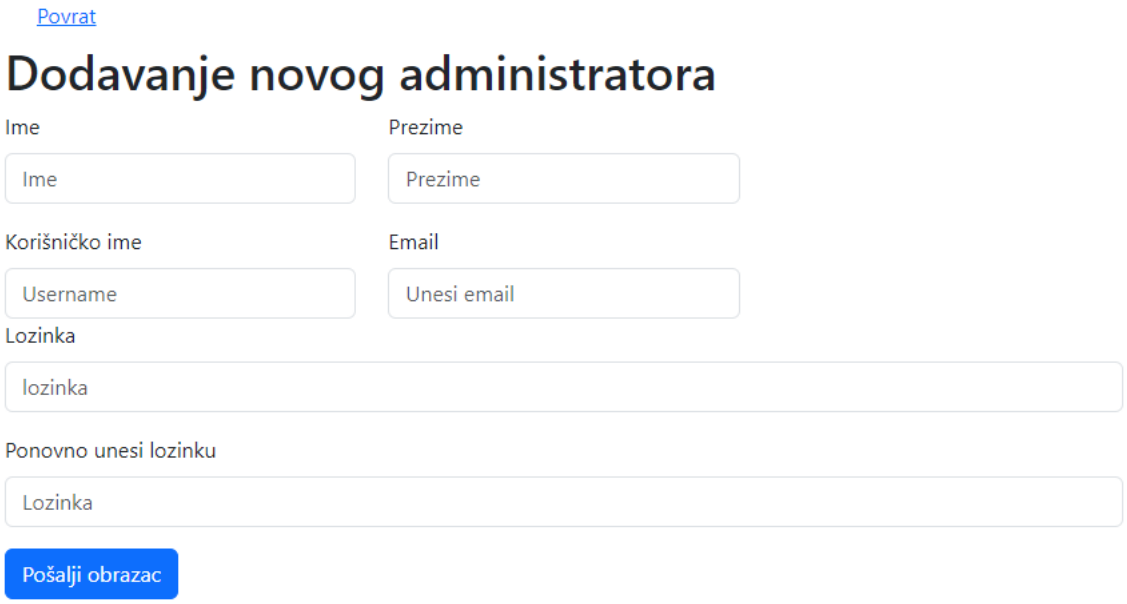
komponentama. Na gornjem dijelu stranice nalazi se poveznica koja vraća korisnika na prethodnu stranicu. Ispod toga nalazi se opcija za dodavanje novog administratora, koja vodi na stranicu za dodavanje administratora. Također, dostupan je filter za pretraživanje administratora po imenu, prezimenu ili ID-u. Ispod filtra prikazana je lista svih administratora na sveučilištu, pri čemu prvi na listi prikazuje trenutno prijavljenog administratora. Zbog sigurnosnih razloga, prijavljeni administrator ne može uređivati vlastite podatke, već samo uređivati ili brisati druge administratore. Klikom na 'ažuriraj' uz odabranog administratora, korisnik će biti preusmjeren na stranicu za ažuriranje administratora, dok će pritiskom na 'izbriši' izabrani administrator biti uklonjen iz baze.



Slika 6: Sučelje s popisom administratora

3.2.2 Dodavanje korisničkog računa administratora

Ako administrator želi dodati novog administratora u sustav, potrebno je ispuniti tražena polja u obrascu. Slika 7 prikazuje stranicu za generiranje novog administratora. Kada se obrazac ispuni, zahtjev se šalje bazi podataka za stvaranje novog administratora. Ako se korisničko ime, e-mail ili lozinka već koriste ili nisu u skladu s pravilima, aplikacija šalje korisniku poruku o pogrešci.



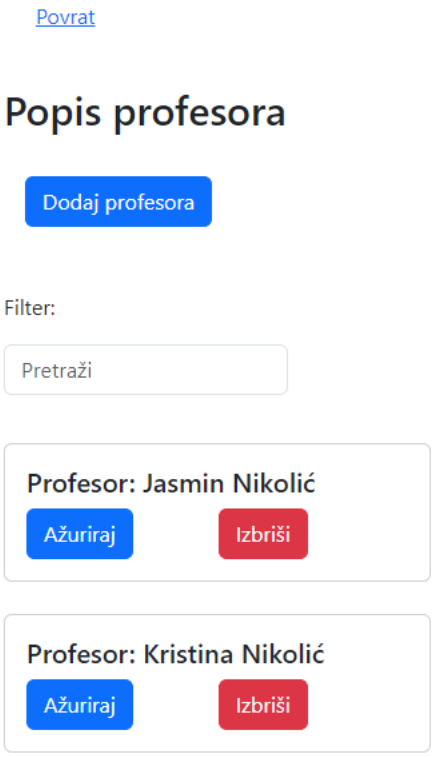
Slika 7: Sučelje za dodavanje novog korisničkog računa administratora

3.2.3 Ažuriranje administratora

Ako korisnik želi ažurirati podatke postojećeg administratora, klikom na gumb 'ažuriraj' odabranog administratora bit će preusmjeren na stranicu za ažuriranje. Stranica je slična stranici za dodavanje novog administratora, osim što su polja automatski popunjena starim podacima.

3.2.4 Stranica profesor

Na stranici s popisom profesora prikazan je popis svih profesora upisanih na sveučilištu. Na vrhu stranice nalazi se poveznica koja vraća korisnika na prethodnu stranicu aplikacije. Tu su i filter za pretraživanje željenog profesora te opcije za brisanje i ažuriranje odabranog profesora. Na slici 8 prikazano je sučelje popisa profesora.

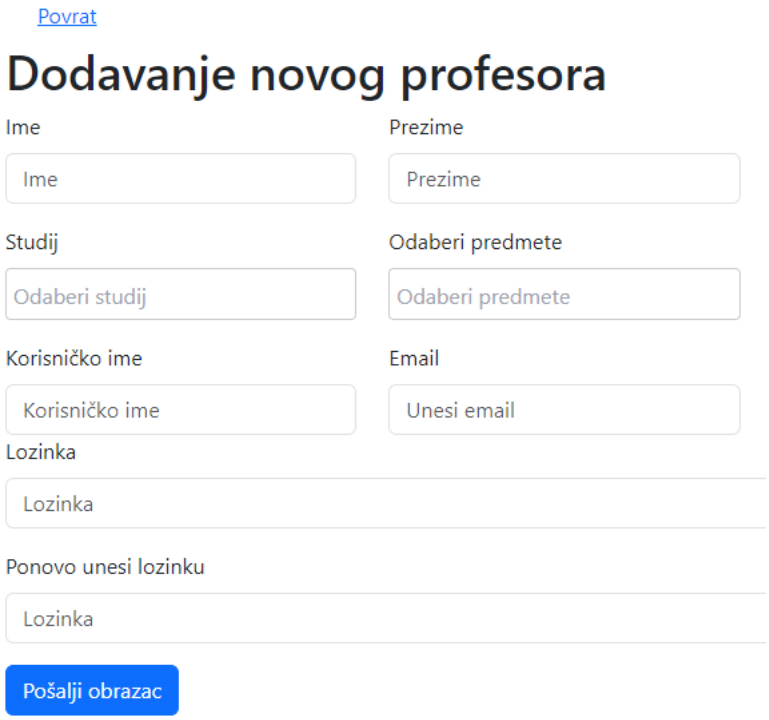


Slika 8: Sučelje s popisom profesora

3.2.5 Dodavanje korisničkog računa profesora

Na stranici za kreiranje novog profesora unose se osnovne informacije potrebne za upis. Na slici 9 prikazan je obrazac za dodavanje novog profesora. Administrator odabire jedan ili više studija koje će profesor predavati. Nakon odabira studija, polje za predmete se popunjava svim predmetima iz tih studija, a korisnik odabire koji će predmet biti nositelj.

Za ažuriranje profesora, sučelje je slično onome na stranici za kreiranje, ali obrazac je popunjen postojećim informacijama koje treba ažurirati.



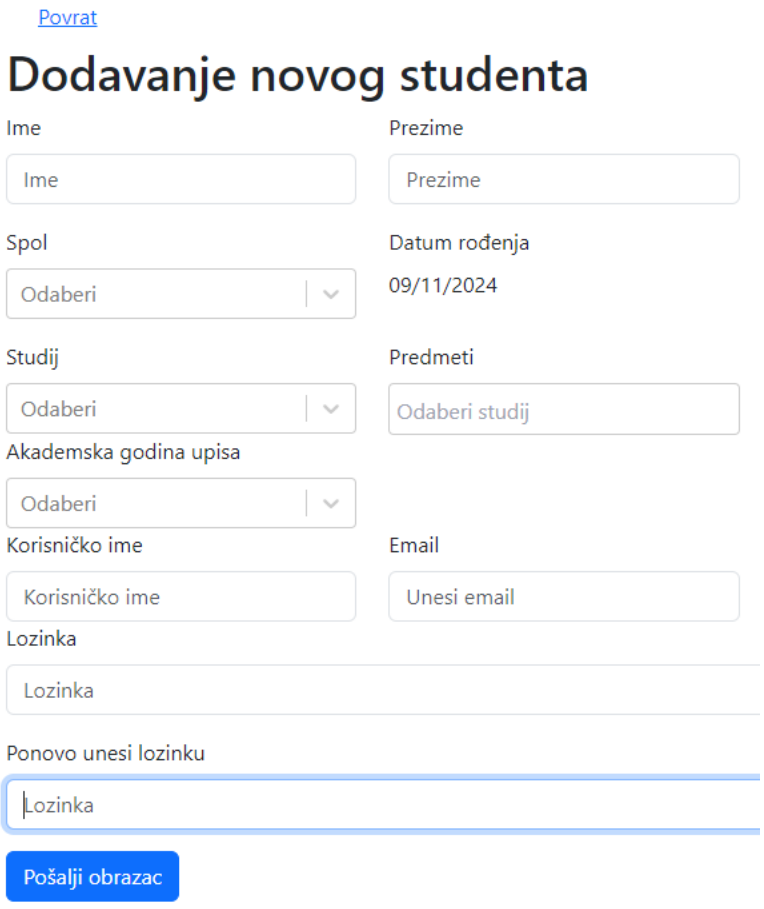
Slika 9: Sučelje za dodavanje novog profesora

3.2.6 Stranica student

Stranica s popisom studenata slična je ostalim stranicama s popisima. Sadrži filter i opcije za dodavanje, ažuriranje i brisanje studenata iz sustava.

3.2.7 Dodavanje korisničkog računa student

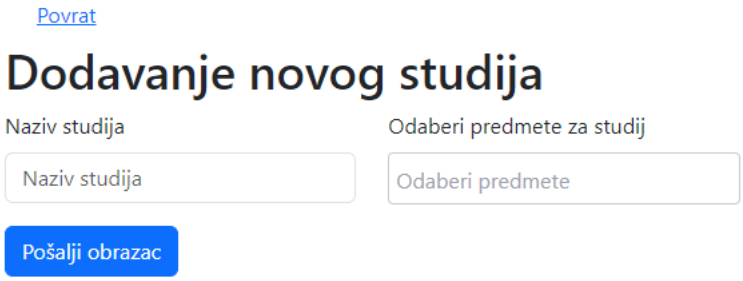
Za dodavanje novog studenta potrebno je ispuniti obrazac s traženim informacijama. Administrator odabire studij za koji se student prijavljuje i predmete povezane s tim studijem.



Slika 10: Sučelje za dodavanje novog studenta

3.2.8 Dodavanje novog studija

Za kreiranje novog studija prvo je potrebno stvoriti nove predmete povezane s tim studijem. Nakon toga može se kreirati novi studij i dodati predmeti koji se nude na sveučilištu. Na slici 11 prikazan je obrazac za kreiranje studija.



Slika 11: Sučelje za dodavanje novog studija

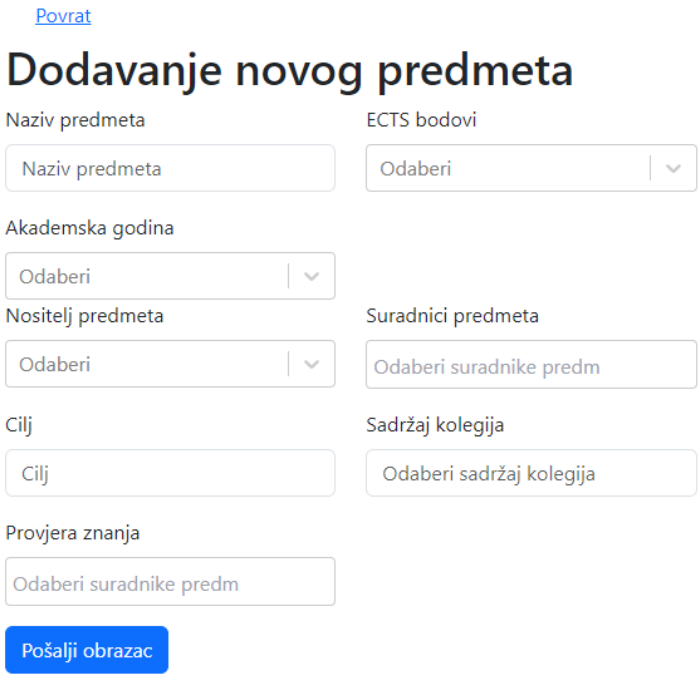
3.2.9 Stranica predmet

Sučelje s popisom predmeta slično je ostalim sučeljima u aplikaciji, ali s dodatnim opisnim informacijama. Na slici 12 prikazan je popis predmeta s brojem ECTS bodova i akademskom godinom u kojoj se predmet odvija.

Slika 12: Sučelje s popisom predmeta

3.2.10 Dodavanje novog predmeta

Za kreiranje novog predmeta potrebno je ispuniti obrazac s potrebnim informacijama. Treba odabrati profesora koji će biti nositelj predmeta, a po potrebi dodati suradnike predmeta. Nositelj predmeta ne može istovremeno biti suradnik istog predmeta.



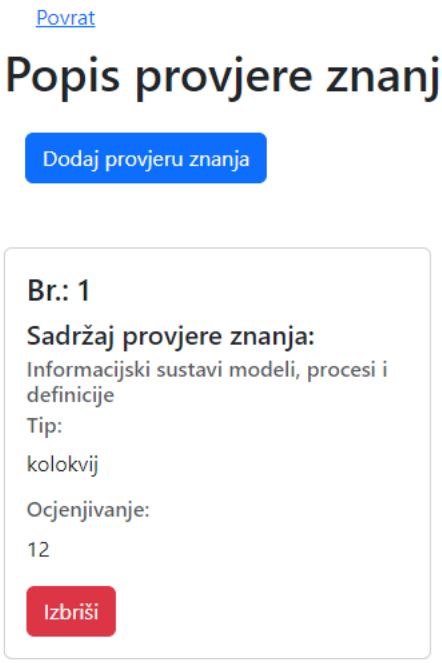
Slika 13: Sučelje za dodavanje novog predmeta

3.2.11 Stranica ishod učenja

Sučelje za ishod učenja slično je ostalim entitetima u aplikaciji. Za kreiranje novog ishoda učenja potrebno je opisati ishod i odabrati predmet kojem pripada. Definiranje predmeta je važno jer će se, kada se kreira kolokvij ili provjera znanja za taj predmet, prikazivati ishodi učenja vezani za taj predmet.

3.2.12 Stranica provjera znanja

Na sučelju za popis provjere znanja dostupne su opcije za kreiranje i brisanje provjera znanja. Na popisu su prikazani detalji svake provjere znanja: sadržaj, tip i broj bodova.

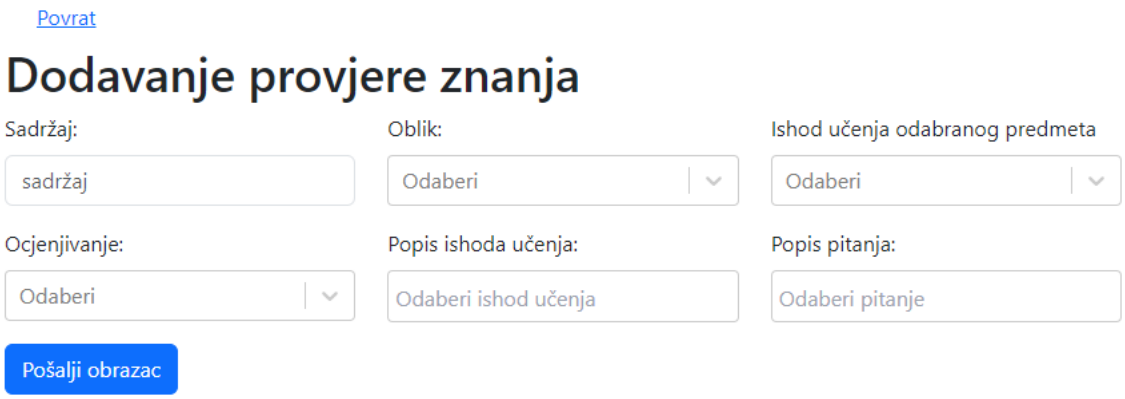


Slika 14: Sučelje s popisom provjere znanja

3.2.13 Dodavanje provjere znanja

Kada se kreira provjera znanja, na stranici se koristi obrazac za popunjavanje potrebnih informacija. Na slici 15 prikazan je obrazac za kreiranje nove provjere znanja. Pri specificiranju sadržaja treba odabrati tip provjere znanja: kolokvij, ispit ili laboratorijske vježbe. Definiranje osnovnih podataka uključuje odabir predmeta kojem će pripadati provjera znanja. Za ocjenjivanje je predviđen raspon od 1 do 15 bodova. U izborniku za ishode učenja prikazane su sve opcije koje pripadaju odabranom predmetu, čime se povlače svi ishodi učenja tog predmeta. Na kraju, mogu se odabrati višestruka pitanja koja će se koristiti za provjeru znanja i ocjenjivanje.

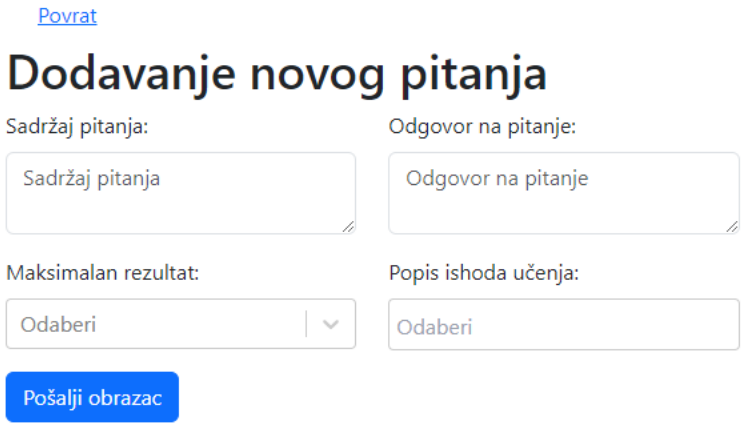
Provjere znanja ne mogu se ažurirati zbog sigurnosti i preciznosti; umjesto ažuriranja, kreira se nova provjera znanja.



Slika 15: Sučelje za dodavanje provjere znanja

3.2.14 Dodavanje novog pitanja

Kada se kreira novo pitanje koje će se kasnije uključiti u provjeru znanja, potrebno je definirati pitanje sa sadržajem i točnim odgovorom. Dodaje se broj bodova povezan s pitanjem i odabiru se višestruki ishodi učenja povezani s tim pitanjem. Na slici 16 prikazan je obrazac za popunjavanje potrebnih podataka za kreiranje novog pitanja.



Slika 16: Sučelje za dodavanje novog pitanja

3.3 Sučelje profesora

Prolaskom kroz stranice administratora dobiva se detaljan uvid u sve entitete povezane s ovom aplikacijom i njihove relacije. Glavna uloga administratora je kreiranje, ažuriranje i brisanje entiteta u bazi podataka. Time se postavljaju uvjeti za sljedeće korisnike (nastavnike/profesore), čije će se dopuštenje i slučajevi korištenja detaljno razmotriti.

Kada se profesor prijavi u web aplikaciju, sustav za provjeru baze podataka i autentifikacije provjerava ima li korisnik valjanu autorizaciju i ulogu za pristup početnoj stranici sučelja za profesore. Na slici 17 prikazan je primjer početne stranice profesora koji predaje dva predmeta. U ovom slučaju, profesor predaje predmete iz sustava za skladištenje podataka i statistike. To znači da profesor predaje predmete na dva različita studija na fakultetu, što je uobičajeno u stvarnom svijetu.

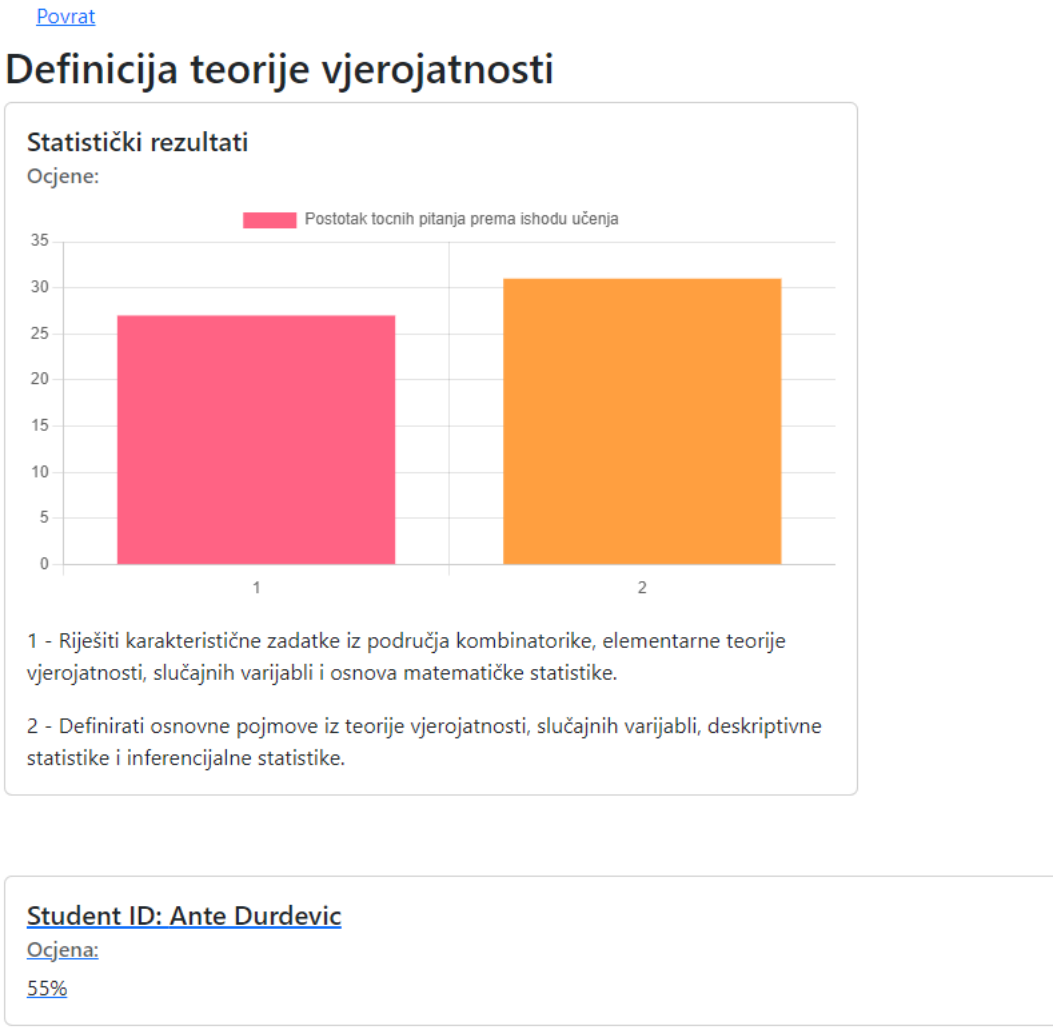
Slika 17: Sučelje početne stranice profesora

Profesor ima pristup svim predmetima za koje je nositelj ili suradnik. Kad profesor odabere predmet kojem želi pristupiti, u ovom slučaju statistiku, bit će preusmjeren na stranicu

s detaljima predmeta. Na slici 18 prikazana je detaljna stranica predmeta statistike s opisom predmeta, brojem ECTS bodova, ciljem i sadržajem predmeta.

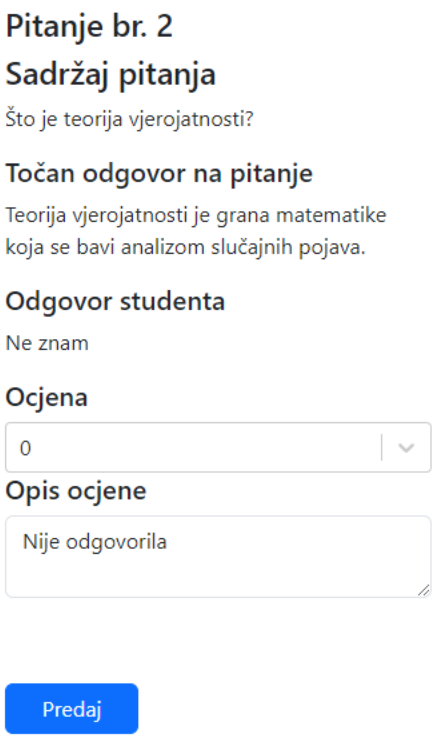
Slika 18: Sučelje s opis predmeta

Na slici je također prikazan popis provjera znanja koje testiraju znanje studenata i utvrđuju ispunjavaju li sve ishode učenja. Profesor odabire provjeru znanja za ocjenjivanje studenata i dodavanje povratnih informacija na njihove odgovore. Na slici 19 vidimo statističku analizu ispita iz provjere znanja o definicijama teorija vjerojatnosti. Grafikon prikazuje prosječnu ocjenu svih odgovora studenata i pruža povratne informacije profesoru. U ovom primjeru, studenti su u prosjeku ispunili 31% ishoda učenja u vezi s definiranjem osnovnih pojmova teorije vjerojatnosti, dok je 27% studenata točno odgovorilo na pitanja vezana za rješavanje karakterističnih problema iz područja kombinatorike. Ispod grafa prikazan je popis svih studenata upisanih na kolegij i njihove ocjene na provjeri znanja.



Slika 19: Sučelje s rezultatima ocjena iz provjere znanja o definicija teorije vjerojatnosti

Klikom na jednog od studenata koji su polagali ispit, korisnik će biti preusmjeren na stranicu koja prikazuje odgovore studenata na pitanja, dodijeljene ocjene i razlog zašto je nastavnik dodijelio određenu ocjenu. Ove funkcionalnosti pomažu profesorima da dobiju jasne informacije o ispitima i znanju studenata iz određenog predmeta. Glavni cilj projekta je pružiti nastavnicima korisne povratne informacije kako bi unaprijedili i implementirali potrebne promjene u predmetu i time omogućili studentima bolje razumijevanje. Ovaj princip poboljšava znanje studenata i povećava postotak prolaznosti predmeta. Na slici 20 vidimo sučelje za ispit studenta. Sučelje za ispit prikazuje sva pitanja u ispitu, studentov odgovor, zasluženu ocjenu i razlog dodjeljivanja ocjene, koji je opisao profesor.



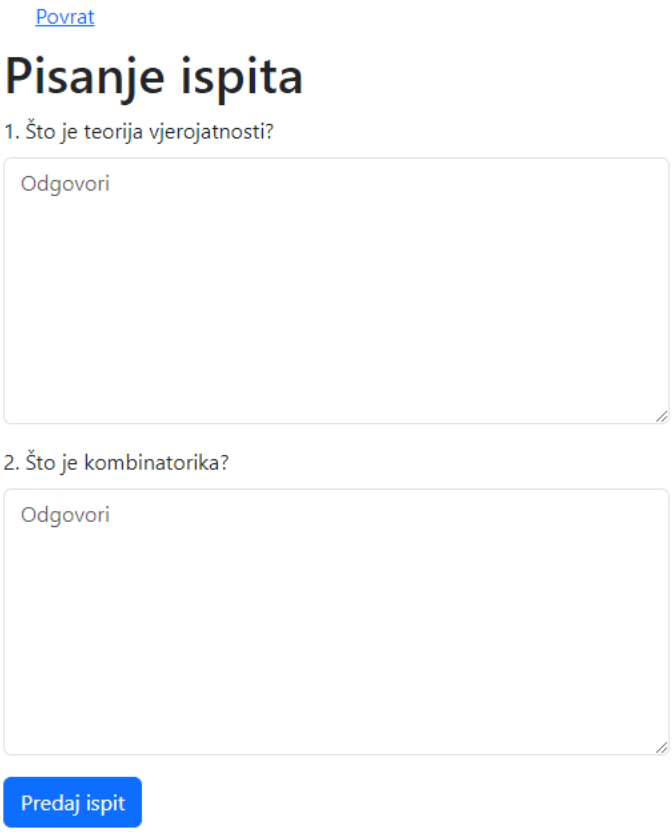
Slika 20: Sučelje Ispit - Ocjenjivanje

3.4 Sučelje studenta

Slijedi konačna korisnička uloga s najmanjim pravima u projektu – studenta. Kad se student prijavi u web aplikaciju, sustav za autorizaciju i autentifikaciju provjerava ima li korisnik valjanu autorizaciju i ulogu za pristup početnoj stranici sučelja za studente. Nakon uspješne prijave, korisnik će biti preusmjeren na početnu stranicu za studenta, prikazanu na slici 21, koja sadrži popis upisanih predmeta. U ovom slučaju, student je upisan na kolegij statistika za akademsku godinu 2020./2021.

Slika 21: Sučelje početne stranice studenta

Klikom na predmet, student će biti preusmjeren na stranicu predmeta, prikazanu na slici 22, gdje mora ispuniti otvoreni ispit za provjeru znanja. Aplikacija obuhvaća sva pitanja dodijeljena u provjeri znanja, na koja student mora odgovoriti i predati ih. Nakon završetka ispita i predaje, aplikacija će automatski završiti ispit i proslijediti unesene podatke na arhiviranje.



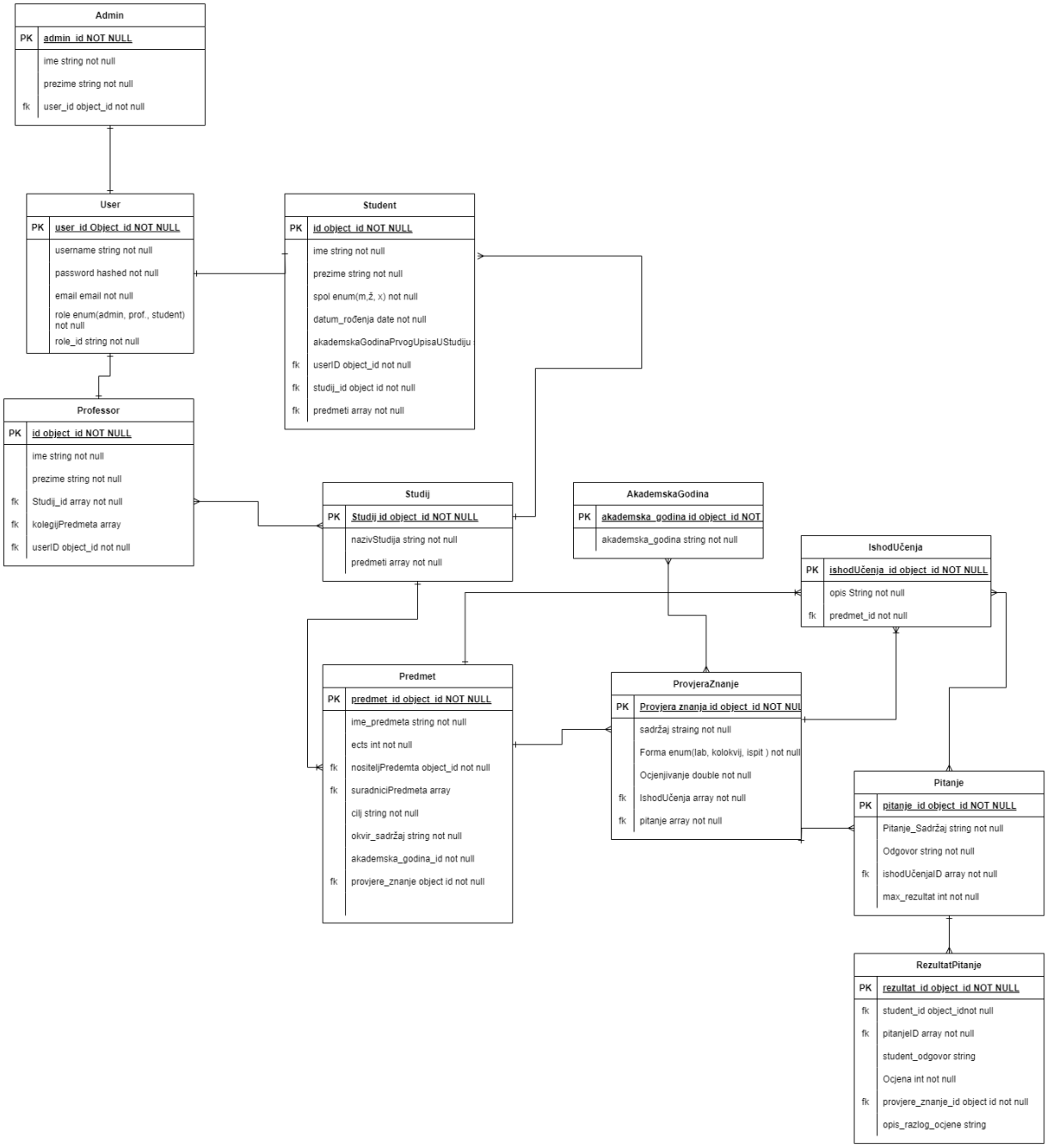
Slika 22: Sučelje studentskog ispita

3 Implementacija

Opisani su svi aspekti funkcionalnosti u projektu. Sljedeći korak je pregled tehničke realizacije. Fokus će biti na temeljim tehnologijama, arhitekturi i strategijama kodiranja. U ovom poglavlju detaljno će se raspraviti cijela arhitektura projekta, uključujući bazu podataka, *backend* i *frontend* implementaciju.

4.1 Implementacija baze podataka

Prije bilo kakvih radnji na bazi podataka, potrebno je prvo modelirati sve entitete i veze između objekata koji su potrebni u projektu. Na slici 23 prikazan je ERD (entitet-relacijski dijagram) izrađen u draw.io, besplatnom online softveru za kreiranje dijagrama.

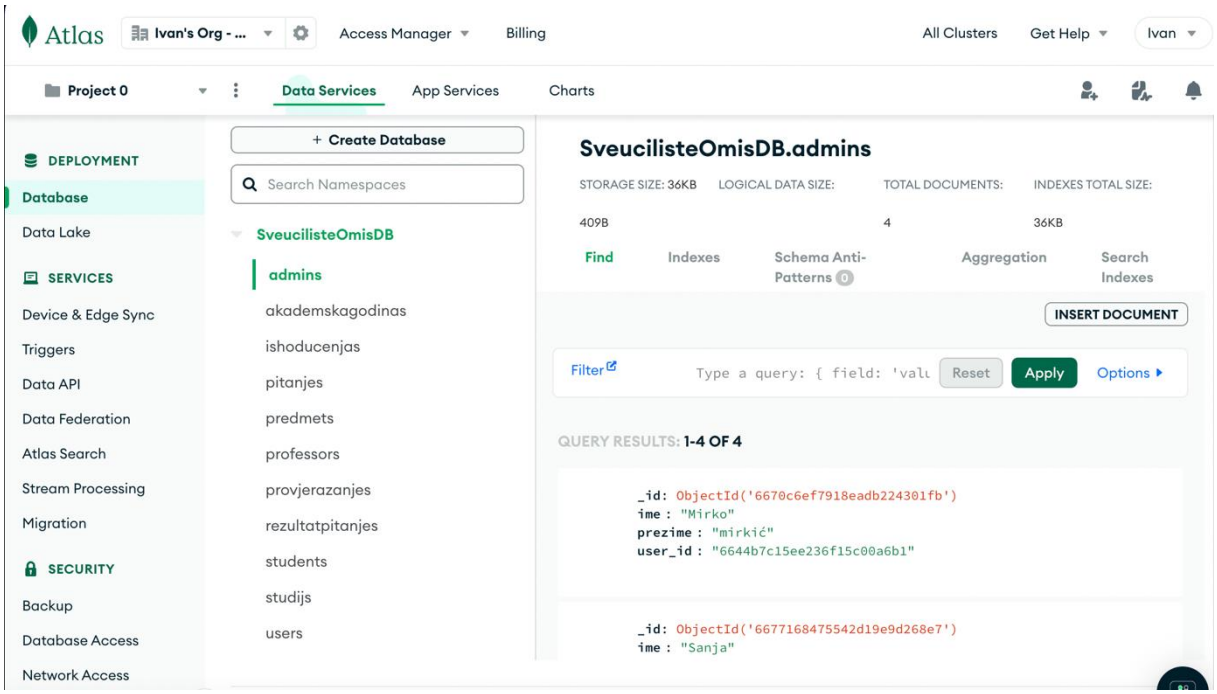


Slika 23: Relacijski dijagram entiteta i veza modela projekta

Ovaj dijagram prikazuje relacije i veze između svih entiteta u projektu, što omogućava bolje razumijevanje sistemskih zahtjeva i identifikaciju entiteta i njihovih veza. Pomaže u otkrivanju problema u ranijoj fazi razvoja projekta, omogućujući identificiranje potencijalnih

problema u dizajnu baze podataka, kao što su dupliciranje podataka, loša normalizacija ili neefikasne veze između entiteta. Ova shema pomaže u izbjegavanju skupih i dugotrajnih promjena tijekom kasnijih faza razvoja projekta.

Nakon definiranja sheme entiteta, može se započeti s kreiranjem MongoDB baze podataka i kolekcijama tih entiteta. Prvo se kreira nova baza podataka u MongoDB-u, a zatim se kreiraju kolekcije za svaki entitet u shemi. Slika 24 prikazuje popis kolekcija u MongoDB bazi podataka "Sveučilište Omiš".



Slika 24: Popis kolekcija u MongoDB bazi podataka

Korištenje MongoDB-a omogućava veću fleksibilnost u shemi, brži pristup podacima i dodatnu prilagodljivost. MongoDB JSON dokumenti podržavaju složene i ugniježđene strukture podataka, smanjujući potrebu za razdvajanjem informacija u više tablica i pojednostavljujući dizajn baze podataka. Kompatibilnost s modernim web i mobilnim

aplikacijama, koje često koriste JSON za razmjenu podataka, omogućuje direktnu integraciju i poboljšava performanse. Ažuriranja kolekcije u bazi i struktura podataka su trenutni, čime se eliminira potreba za uređivanjem relacija i tablica kao u relacijskim bazama podataka.

Nakon kreiranja baze podataka s potrebnim kolekcijama za skladištenje podataka entiteta, sljedeći korak je izgradnja *backend* dijela aplikacije koji povezuje *frontend* zahtjeve s MongoDB bazom podataka.

4.2 Implementacija backenda

Kod kreiranja bilo koje aplikacije ili programa, vrlo je bitno imati dobro dizajniranu bazu podataka i strukturiranu mapu s čistim kodom. Imati organiziranu strukturu mapa za izgradnju programa ključno je za učinkovitost i organizaciju tijekom razvoja. Jasno definirana struktura omogućava brži pristup svim komponentama projekta, što povećava produktivnost i smanjuje vrijeme potrebno za pronalaženje i izmjenu datoteka. Organizirani pristup također olakšava održavanje i nadogradnju programa jer pomaže u razumijevanju međusobne povezanosti različitih dijelova.

Prije instalacija ili programiranja, potrebno je kreirati mape koje će sadržavati cijeli projekt. Na slici 27 prikazana je mapa koja sadrži cijeli projekt i naziva se „`ZAVRSNI-RAD`“. Unutar te mape nalaze se dvije podmape:

- `sveuciliste-omis` – mapa koja sadrži *frontend* dio
- `sveuciliste-omis-api` - mapa koja sadrži *backend* dio

U detaljima će biti razmotreni koraci za kreiranje projekta i instaliranje potrebnih alata za rad na *backendu*.

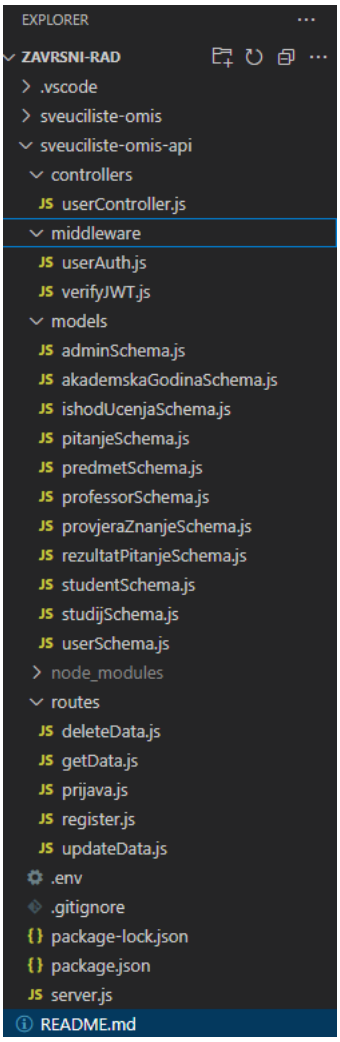
4.2.1 Inicijalizacija i instalacija

Kada se kreira novi direktorij za projekt, potrebno je ući u taj direktorij putem terminala i inicijalizirati Node.js poslužitelj. Ispis 1 prikazuje postupak instalacije Node.js-a i potrebnih alata za projekt.

```
      npm init -y
      # kreira package.json datoteku.
      npm install express mongoose cors b-crypt dotenv   jsonwebtoken
nodemon
      # instalira sve potrebne alate i ekstenzije za projekt.
      # Nakon što su svi alati potrebni za projekt instalirani u projektu,
može se krenuti s radom na backendu
```

Ispis 1: Skripta za inicijalizacije Node.js-a

Slika 25 prikazuje strukturiranu mapu završnog rada, uključujući sve podmape i datoteke. Udaljit će se najvažniji elementi i objasniti njihove funkcionalnosti i namjene.



Slika 25: Strukturirana mapa datoteka *backend*

Važne mape i datoteke su:

- `Package.json` – Datoteka koja sadrži metapodatke i ovisnosti za projekt. Omogućuje jednostavno upravljanje i dijeljenje projekta s drugima.
- `Package-lock.json` – Datoteka za upravljanje verzijama koja sadrži popis biblioteka koje Node.js projekt treba za izvođenje.

- `Server.js` – Datoteka koja se inicijalizira i pokreće prva. Ovdje se konfigurira Express poslužitelja.
- `.gitignore` – Datoteka koja definira koje se datoteke ignoriraju i nisu važne za projekt.
- `.env` – Datoteka koja pohranjuje konfiguracijske postavke, varijable okoline i osjetljive informacije.
- `Node_modeules`– Mapa koja pohranjuje vanjske ovisnosti projekta.
- `Models` – Mapa koja sadrži strukture svih entiteta iz baze (npr. administrator, predmet, pitanje). Predstavlja podatke i njihove odnose unutar aplikacije te pomaže u organizaciji, manipulaciji i interakciji s podacima unutar baze.
- `Controllers` – Komponente koje upravljaju protokom podataka između modela i *frontenda*. Rukovode korisničkim unosima, obrađuju zahtjeve i vraćaju odgovore.
- `Routes` – Mapa koja sadrži datoteke s odgovarajućim rutama za funkcije.
- `Middleware` – Mapa s datotekama koja obrađuje HTTP zahtjeve i odgovore između klijenta i poslužitelja. Može obavljati različite zadatke kao što su autentifikacija, logiranje itd.

Nakon upoznavanja sa strukturom mape projekta, potrebno je proći kroz inicijalizaciju Express poslužitelja i povezivanje s MongoDB bazom podataka.

U datoteci `server.js` konfigurira se Express poslužitelj. Na ispisu 2 prikazan je kôd za kreiranje Express poslužitelja. Kôd prikazuje URL na kojem se web aplikacija prikazuje i popis HTTP metoda koje su dopuštene.

```
//express app
const app = express()

app.use(express.json())


app.use(cors({
    origin: ['http://localhost:3000' /** other domains if any */ ],
    Methods: ["GET", "POST" , "PUT", "DELETE", "HEAD"],
    credentials: true,//access-control-allow-credentials:true
    optionSuccessStatus:200
}))


app.use(cookieParser())
```

Ispis 2: Skripta za stvaranje i pokretanje Express poslužitelja

Kad je kreiran Express poslužitelj, potrebno ga je povezati s MongoDB bazom podataka. Povezivanje s MongoDB-om vrši se pomoću URI-ja (engl. *Uniform Resource Identifier)*, koji aplikaciji omogućava direktan pristup podacima i skladištu.

```
//connect to mongo db
const dbURI = process.env.MONGO_URI //ovde stavljamo MongoDB URI koje
//povlacivamo od baze i stavljamo u .env datoteku kao parameter MONGO_URI


mongoose.connect(dbURI, {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(()=>{ //ako je URI validan, povez sa bazom podataka je uspješan
    app.listen(process.env.PORT, () => {
        console.log('listening on port:', process.env.PORT)
```

```
        console.log('Server connected to MongoDB')
    })
}).catch((error)=>{ //u slucaju neuspjesnog povezivanja ispiši razlog
    console.log('Unable to connect to Server/MongoDB')
    console.log(error)
})
```

Ispis 3: Skripta za povezivanje Express poslužitelja s MongoDB bazom podataka

Nakon što Express poslužitelj bude konfiguriran i MongoDB povezan, *backend* je spreman primiti zahtjeve s korisničkog sučelja i povezati ih s odgovarajućim funkcijama.

Kao i kod svake aplikacije, postoji autentifikacija i autorizacija različitih korisnika i uloga kako bi se spriječile zlonamjerne aktivnosti i sigurnosni propusti u sustavu na strani korisnika. U direktoriju `routes`, u datoteci `prijava.js`, prikazan je kod u ispisu 4, gdje validiramo i autentificiramo prijavljenog korisnika.

Unutar skripte koristi se biblioteka `bcrypt`, koja se koristi za sigurno heširanje lozinki. Ona pretvara unesenu lozinku u sučelju za prijave u niz karaktera (heš) koji nije lako vratiti u izvorni oblik. Bcrypt je dizajniran da bude sporiji kako bi otežao napade poput brute-force napada i koristi kriptografski algoritam za kreiranje heša lozinke. Također automatski generira „salt“, koji je slučajni niz karaktera dodan lozinci prije haširanja. Može se podešavati složenost heširanja pomoću broja iteracija (radnih ciklusa), što kontrolira koliko vremena i resursa je potrebno za heširanje lozinke. Veći broj iteracija pruža jaču zaštitu, ali uzrokuje sporije heširanje.

Uz bcrypt, koristi se i JWT (engl. *JSON Web Token)* za autentifikaciju, gdje se korisnički identitet potvrđuje i zatim koristi za verifikaciju pristupa resursima ili uslugama.

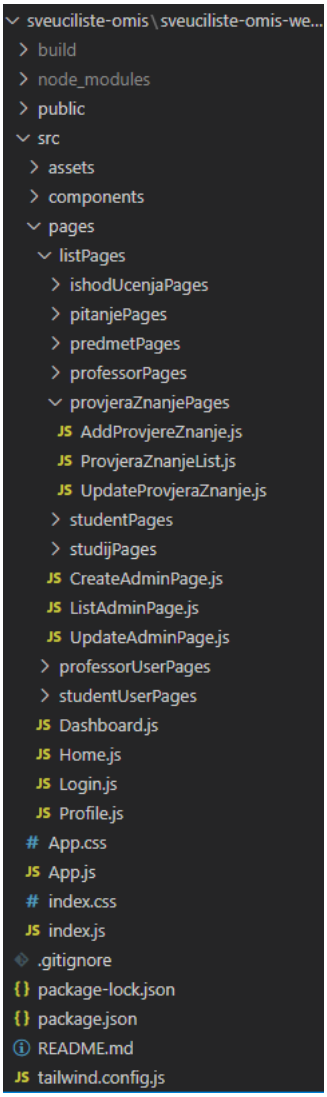
```
const bcrypt = require('bcrypt')
```

```
const router = express.Router()
const jwt = require('jsonwebtoken')
require('dotenv').config()

//dohvati registrirani user
router.post("/prijava", async(req, res) => {
    try{
        const {username, password} = req.body
        const user = await User.findOne({username})

        if(!user){
            return res.status(401).json({error: "Invalidni
podatci"})
        }

        const isPasswordValid = await bcrypt.compare(password,
user.password)
        if(!isPasswordValid){
            return res.status(401).json({error: "Invalidni
podatci"})
        }else{
            const token = jwt.sign({userId: user._id, role:
user.role, role_id: user.role_id}, process.env.SECRET_KEY,
{expiresIn: '1hr'})
            res.cookie('token', token)
            res.json({Status: "Uspješna prijava", role: user.role,
loginSuccessfull: true, user: user})
        }
    }catch(error){
        res.status(500).json({error: "Neuspješna prijava!"})
    }
})
```

Ispis 4: Skripta za autentifikaciju i autorizaciju korisnika

4.3 Implementacija frontenda

Sad kad je *backend* online i spreman za primanje zahtjeva od korisnika, može se pregledati struktura i funkcionalnosti korisničkog sučelja. Na slici 25 prikazana je struktura datoteka na *frontend* aplikacije.



Slika 25: Struktura mape u *frontendu*

Na *frontendu* se nalaze slične datoteke kao kod *backenda*, uz dodatke:

- `Indeks.js` – osnovna datoteka koja služi kao ulaz u aplikaciju.
- `App.js` – sadrži web rute i povezuje ih s ostalim stranicama u aplikaciji.
- `Login.js` – stranica za prijavu korisnika u aplikaciju putem web preglednika.
- `Pages` – datoteka koja sadrži sve stranice projekta.
- `Components` – datoteka sa skupom gradivnih blokova, odnosno ponovo upotrebljivih dijelova korisničkog sučelja.
- `Assets` – datoteka koja sadrži digitalne resurse aplikacije koji se koriste unutar projekta.

Sve ove datoteke organizirane su prema svojim karakteristikama i klasifikacijama. U React aplikaciji, organizacija datoteka u mape poput `pages` i `components` ključna je za održavanje čistoće i skalabilnosti koda. Mapa `pages` sadrži datoteke koje predstavljaju pojedinačne stranice ili rute aplikacije. To omogućava jasno razdvajanje različitih pogleda, olakšavajući upravljanje navigacijom i održavanje aplikacije. S druge strane, mapa `components` sadrži ponovno upotrebljive UI komponente, poput dugmadi, obrazaca ili navigacijskih traka, koje se mogu koristiti na više stranica. Ova struktura omogućava bolje organiziran kod, lakšu ponovnu upotrebu komponenata i olakšava rad u timovima, čineći razvoj aplikacije učinkovitijim i preglednijim.

Jedna od glavnih funkcionalnosti i ciljeva projekta je vizualno prikazivanje i grafičko izračunavanje rezultata studenata te njihovo povezivanje s ishodima učenja. Na ispisu 5 prikazana je skripta koja generira grafikon rezultata i potrebne parametre za računanje rezultata. Ispis također prikazuje komponentu navigacije „`NavigationBar`“, koja renderira navigacijsku traku. Za prikaz „`StatisticsCard`“ grafikonske kartice potrebno je imati validan objekt provjere znanja koji je profesor odabrao za prikaz statističkih rezultata studenata. Ako postoji provjera znanja, tada se u „`StatisticsCard`“ unose sljedeći parametri:

- `provjeraZnanjeId` – ID provjere znanja ispita.
- `studentsInCourse` – studenti upisani u predmetu u kojem se polaže provjera znanja.

```
<div>
        <Navigationbar getUser={getUser}></Navigationbar>
        <div className='ml-10'>
            <BackButton destination={"/dashboard"}></BackButton>
            {provjeraZnanje ? (
              <h2>{provjeraZnanje.sadrzaj}</h2>
            ):(<div></div>)}
             {provjeraZnanje &&
            <StatisticsCard provjeraZnanjeId={provjeraZnanje._id}
students={studentsInCourse}></StatisticsCard>
             }
            <h4 className='mb-4'>Studenti upisani u predmet</h4>
            {loading ? (
    <Spinner/>
```

Ispis 5: Skripta za prikazivanje grafikona

U ispisu 6 prikazan je kod za prikaz grafikona i potrebni parametri. Ovdje su obuhvaćeni svi ishodi učenja povezani s odabranom provjerom znanja.

```
return (
    <div>
    <Card style={{ width: '40rem'}} className="mb-4">
      <Card.Body>
        <Card.Title>Statistika Rezultati</Card.Title>
        <Card.Subtitle className="mb-2 text-
muted">Ocjena:</Card.Subtitle>
        <Card.Text>
            <Bar data={data} />
```

```
            {
        ishodUcenja && ishodUcenja.map((ishod, index)=>
            <div className="mt-3">
             {(index+1) + " - "+ ishodUcenjaData[index]}
            </div>
        )}
        </Card.Text>
      </Card.Body>
    </Card>
    </div>
  )
```

Ispis 6: Grafikon komponent

Ova komponenta dohvaća sva pitanja iz provjere znanja i prikuplja rezultate studenata da bi se prikazivali na grafikonu. Funkcionira na način da prvo dohvaća sva pitanja iz provjere znanja, a zatim prolazi kroz sve studente koji sudjeluju u tom kolegiju i koji su odgovorili na ta pitanja. Kada se dobiju rezultati ocjenjivanja, dohvaća se ishod učenja vezanog za to pitanje i ažurira s rezultatom.

4.4 Pokretanje aplikacije

Za pokretanje aplikacije potrebni su detalji o postupnom pokretanju koji su navedeni u `README.md` datoteci. `README.md` je tekstualna datoteka koja pruža osnovne informacije o projektu. Služi kao prvi dokument s kojim korisnici ili programeri dolaze u kontakt kada pregledaju projekt. Svrha datoteke uključuje pregled projekta, upute za instalaciju i korištenje i druge relevantne informacije.

Kako bi se pokrenuo završni rad, projekt treba otvoriti u okruženju za kodiranje, npr. (Visual Studio Code, IntelliJ, itd.). Nakon otvaranja projekta, potrebno je otvoriti dva odvojena tekstualna terminala, jedan za *frontend* i jedan za *backend*.

Prije pokretanja *backenda*, potrebno je instalirati sve pakete navedene u `package.json-u`. Nakon instalacije, poslužitelj se može pokrenuti. Ispis 7 prikazuje kôd i upute za pokretanje *backenda*. Kad poslužitelj bude uspješno pokrenut, može se nastaviti s pokretanjem *frontenda*.

```
cd sveuciliste-omis-api // datoteka koja sadržava back-end
npm install // instaliranje node datoteka, biblioteka i modula
npm run server // pokreće back-end server
//uspjesno povezivanje isprinta poruku 'Server connected to MongoDB'
```

Ispis 7: Skripta za pokretanje *backenda*.

Za pokretanje *frontenda*, treba preći u `sveuciliste-omis` direktorij i pokrenuti kôd prikazan u ispisu 8.

```
cd sveuciliste-omis
cd svuciliste-omis-web-app
npm start // pokretanje front-end-a
```

Ispis 8: Skripta za pokretanje *frontenda*.

Pokretanje ove posljednje linije koda automatski otvara preglednik u kojem se prikazuje početna stranica za prijavu. Time je završni rad uspješno pokrenut.

5 Zaključak

U ovom završnom radu opisane su sve funkcionalnosti i način izrade projekta za praćenje postizanja ishoda učenja zadanih u nastavnom procesu. Rad prikazuje kako se sve četiri tehnologije u MERN stogu koriste u potpunosti i kako olakšavaju kreiranje *fullstack* aplikacija. Sve četiri tehnologije koriste JSON dokument format za prijenos podataka i JavaScript programski jezik, koji je jedan od najpopularnijih jezika u svijetu.

Projekt omogućuje profesorima i sveučilištu da dobiju rezultate o postignuću studenata tijekom provjera znanja iz određenog predmeta u akademskoj godini. Profesori imaju vizualni prikaz uspjeha studenata u određenom predmetu i ishodima učenja koje moraju zadovoljiti kako bi uspješno položili ispit. Projekt pruža važne povratne informacije profesorima, koje im pomažu da prilagode način provjere znanja, ažuriraju materijale i metode predavanja. Studentima se vraćaju povratne informacije o provjerama znanja, uključujući objašnjenje dodijeljenih ocjena i komentare ocjenjivača.

Osim dobivanja ocjena i rezultata provjera znanja, ovaj projekt grafički prikazuje rezultate na način koji je čitljiv i razumljiv predavačima, omogućujući im da dobiju što bolje povratne informacije. Korištenjem statističkih podataka i analiza, nastavnici i sveučilišta mogu bolje reagirati i učinkovito primjenjivati promjene u načinu predavanja. Time sveučilište može poboljšati svoju učinkovitost i privući buduće studente.

Tijekom rada na projektu bio sam potaknut na drugačiji način razmišljanja i razumijevanja u vezi s pretraživanjem i implementacijom tehnologija u projektu. Stekao sam duboko razumijevanje o tome kako *backend* i *frontend* komuniciraju, kao i o organizaciji koda radi bolje skalabilnosti i održavanja. Osim tehničkih vještina, poboljšao sam svoje sposobnosti rješavanja problema i logičkog razmišljanja, posebno u pronalaženju i ispravljanju grešaka. Suočio sam se s izazovima, kao što su optimizacija performansi aplikacije, poboljšavanje korisničkog sučelja u Reactu te osiguravanje sigurne autentifikacije i autorizacije korisnika. Ti su izazovi bili vrijedne prilike za učenje, koje su mi omogućile da unaprijedim svoje vještine i bolje razumijem cijeli proces razvoja web-aplikacija. Također, u tehnološkom sektoru postoje

različiti alati i aplikacije koje mogu pojednostaviti rad i uštedjeti vrijeme programerima, omogućujući im da se posvete važnijim zadacima.

Ovaj web projekt može se proširiti i poboljšati na nekoliko ključnih načina. Prvo, dodavanje novih funkcionalnosti može značajno unaprijediti korisničko iskustvo i proširiti opseg aplikacije. Na primjer, integracija s vanjskim API-jem za dodatne podatke ili usluge može pružiti korisnicima dodatnu vrijednost. Također, poboljšanje performansi kroz optimizaciju koda i upotrebu tehnika poput *server-side renderinga* može učiniti aplikaciju bržom. Implementacija naprednih sigurnosnih mjera, kao što su poboljšana autentifikacija i autorizacija, može osigurati veću zaštitu korisničkih podataka. Upotreba modernih alata i tehnologija, poput TypeScripta za statičku provjeru tipova ili GraphQL-a za efikasnije upravljanje podacima, može dodatno unaprijediti razvoj i održavanje aplikacije. Razmatranje korisničkih povratnih informacija za iterativno poboljšanje sučelja i funkcionalnosti također može pomoći u stvaranju aplikacije koja više zadovoljava potrebe korisnika. Uvođenjem automatiziranih testova i CI/CD procesa može se poboljšati kvaliteta kôda i brzina isporuke novih značajki.

Smatra se da je korištenje ovih tehnologija vrlo zanimljivo i lako razumljivo. Ovaj rad pojednostavljuje proces kreiranja *fullstack* web aplikacije, što je znatno kvalitetnije i zanimljivije u odnosu na prethodne tehnologije koje sam koristio tijekom studija. Projekt također pokazuje da se tehnologije mijenjaju brže nego što industrije reagiraju. Na nama programerima i budućim IT (engl. *Information Technology*) stručnjacima je da implementiramo nove ideje i rješenja u našem tehnološkom sektoru.

Literatura

[1] Heap**,** „What is a Tech Stack: Examples, Components, and Diagrams**,**“ https://www.heap.io/topics/what-is-a-tech-stack (posjećeno 7.08.2024.).

[2] MongoDB, „MERN stack explained,“ https://www.mongodb.com/resources/languages/mern-stack (posjećeno 8.08.2024.).

[3] mCloud, „Visual Studio Code – vodič za početnike,“ https://www.mcloud.rs/blog/visual-studio-code-vodic-za-pocetnike/ (posjećeno 10.08.2024.).

[4] React, „React“ https://legacy.reactjs.org (posjećeno 9.08.2024.).

[5] TechTarget, „MongoDB,“ https://www.techtarget.com/searchdatamanagement/definition/MongoDB (posjećeno 14.08.2024.).