

Univerzitet u Novom Sadu,
Fakultet tehničkih nauka

SEMINARSKI RAD

Nastavni predmet: Tehnologije i sistemi eUprave

Naziv teme: eFakultet

Student,
Marko Crnčević

Jun, 2024

1. Sažetak	3
2. Ključne reči	3
3. Uvod	4
4. Srodna rešenja i pregled korisničkih tehnologija	5
4.1 Srodna istraživanja	5
4.2 Korišćene tehnologije	6
5. Specifikacija zahteva	7
5.1 Specifikacija funkcionalnih zahteva	7
5.2 Specifikacija nefunkcionalnih zahteva	11
6. Specifikacija dizajna	12
7. Implementacija	14
8. Demonstracija	18
9. Zaključak	25
10. Literatura	26

1. Sažetak

U ovom radu je opisan sistem eFakultet, koji omogućava pregled i upravljanje akademskim procesima za studente i administrativno osoblje. Za realizaciju projekta korišćeni su UML dijagrami klasa i slučajeva korišćenja[1], programski jezik Go[2] za backend, React[3] za frontend, Docker za kontejnerizaciju servisa[4], single sign-on za autentifikaciju korisnika i servisi koji komuniciraju putem HTTP-a[5]. Primenom predloženog rešenja omogućena je efikasna obrada i pregled informacija kao što su aktivni prijemni ispiti za određene studijske programe, rezultati prijemnih ispita, lista departmana i studijskih programa. Ulogovani studenti mogu pregledati dostupne informacije, prijaviti se na aktivne prijemne ispite, pregledati rezultate ispita i podnositi zahteve za kreiranje završne diplome.

2. Ključne reči

- veb aplikacija
- akademska uprava
- prijemi ispit
- autentifikacija korisnika
- upravljanje studentskim programima

3. Uvod

Razvoj savremenog društva donosi sa sobom rastući obim podataka koji se svakodnevno generišu i koriste u različitim sektorima. U ovom kontekstu, izazov je efikasno prikupljanje, obrada i analiza ovih podataka radi dobijanja korisnih informacija za donošenje odluka. Ovaj izazov posebno se ističe u kontekstu obrazovnih institucija poput eFakulteta, gde je sistem kao što je eFakultet osmišljen da unapredi upravljanje akademskim procesima za studente i administrativno osoblje.

Sistem eFakultet rešava kompleksan zadatak prikupljanja, obrade i analize raznovrsnih akademskih podataka iz više izvora. Bez ovakvog sistema, prikupljanje i upravljanje podacima o studentskim prijemnima, ispitima i administraciji bilo bi zahtevno i podložno greškama. Manualno rukovanje ovim podacima može rezultovati nepreciznim informacijama koje mogu negativno uticati na donošenje ključnih odluka unutar obrazovnih institucija.

Motivacija za razvoj sistema poput eFakultet leži u potrebi za tačnim, pravovremenim i bitnim podacima koji su neophodni za efikasno vođenje akademskih procesa i planiranje nastavnih programa. Ovaj sistem omogućava bolje planiranje, predviđanje i reagovanje na različite izazove u obrazovnom sektoru, pružajući kompletno rešenje koje optimizuje administrativne procese i omogućava bržu i precizniju podršku studentima.

Problem prikupljanja i obrade akademskih podataka postoji otkako postoje obrazovni sistemi. Tradicionalni pristupi uključivali su ručno prikupljanje podataka, što je bilo veoma naporno, vremenski zahtevno i sklono greškama. S razvojem tehnologije, pojavljivali su se različiti sistemi i softverska rešenja koja su omogućila automatizaciju ovih procesa, ali su često bila podeljeni na manje delove i nisu nudila sveobuhvatno rešenje. Sistem eFakultet integriše sve neophodne funkcionalnosti u jednu platformu, omogućavajući studentima i administrativnom osoblju efikasno upravljanje akademskim procesima, od prijave na prijemne ispite do podnošenja zahteva za diplome.

Ostatak ovog rada je organizovan kao što je objašnjeno u nastavku. U drugom poglavlju su predstavljena srodna istraživanja koja se bave prikupljanjem i analizom podataka i korišćene tehnologije u izradi projekta. U trećem poglavlju je data specifikacija zahteva za sistem. Četvrto poglavlje obuhvata specifikaciju dizajna sistema. Implementacija rešenja je predstavljena u petom poglavlju. Šesto poglavlje prikazuje demonstraciju funkcionalnosti sistema. Zaključak je dat u sedmom poglavlju, dok osmo poglavlje sadrži literaturu korišćenu u radu.

4. Srodna rešenja i pregled korisničkih tehnologija

U ovom poglavlju je dat pregled postojećih rešenja za upravljanje obrazovnim procesima, kao i tehnologije koje omogućavaju rešavanje problema upravljanja obrazovnih procesa.

4.1 Srodna istraživanja

Ovaj odeljak donosi pregled aplikacija namenjenih upravljanju obrazovnog sektora kroz integrisane tehnologije.

FTN Web Servis

Sistem za Fakultet tehničkih nauka (FTN)[6] koji pruža sveobuhvatno rešenje za efikasno upravljanje administrativnim procesima i unapređenje akademskog iskustva.

Ovaj sistem integriše različite funkcionalnosti:

- **Evidencija studenata:** Omogućava pristup akademskim podacima, evidenciju upisa, pregled ocena i finansijskih informacija.
- **Administracija nastavnih programa:** Nastavnici koriste platformu za upravljanje kurikulumima, postavljanje materijala za nastavu, praćenje prisustva i ocenjivanje studenata.
- **Komunikacija i obaveštenja:** Omogućava objavljivanje obaveštenja, rasporeda predavanja i direktan kontakt putem internih poruka.
- **Administracija infrastrukture:** Sistem podržava upravljanje rasporedom ispita, prostorijama i rezervacijama.

4.2 Korišćene tehnologije

Ovaj odeljak donosi pregled aplikacija namenjenih upravljanu obrazovnog sektora.

Golang (Go)

Go[2], poznat i kao Golang, je programski jezik otvorenog koda razvijen od strane inženjera u Google-u[7]. Ovaj jezik se ističe svojom jednostavnošću, brzinom izvršavanja i efikasnošću u radu sa istovremenim zadacima. Upotrebljava se široko u web razvoju, sistemskom programiranju i razvoju softvera za velike sisteme zbog svoje robustnosti i skalabilnosti.

React

React[3] je JavaScript[8] biblioteka otvorenog koda koju je razvio tim inženjera u Facebook-u[9] za izgradnju interaktivnih korisničkih interfejsa. Koristi se za efikasno renderovanje UI komponenti u veb aplikacijama, što je posebno korisno za razvoj modernih single-page aplikacija. React se ističe po podršci za JSX, jezik sličan HTML-u[10], koji olakšava integraciju HTML-a unutar JavaScript koda. Ova biblioteka je popularna zbog svoje jednostavnosti, performansi, čineći je jednog od najčešće korišćenih alata za front-end razvoj.

Docker

Docker[4] je platforma za kontejnerizaciju aplikacija koja omogućava jednostavno pakovanje, distribuciju i pokretanje softverskih aplikacija u izolovanim okruženjima. Ova tehnologija omogućava programerima da "zapakuju" aplikacije zajedno sa svim potrebnim zavisnostima u kontejnere, što olakšava jednako i brzo prenošenje aplikacija između različitih okruženja, kao što su razvoj, testiranje i produkcija.

Single Sign-On (SSO)

Single Sign-On je metod autentifikacije koji omogućava korisnicima da se prijavljuju na više aplikacija sa samo jednim setom kredencijala, kao što su korisničko ime i lozinka. Umesto da korisnici moraju unositi svoje podatke za svaku pojedinačnu aplikaciju, SSO omogućava da se jednom prijavom autentifikuju na sve povezane aplikacije. Ovo ne samo da pojednostavljuje proces prijavljivanja, već i poboljšava sigurnost jer smanjuje potrebu za upravljanjem i pamćenjem više različitih lozinki.

5. Specifikacija zahteva

U ovom poglavlju su objašnjeni funkcionalni i nefunkcionalni zahtevi softverskog rešenja predstavljenog u ovom radu.

5.1 Specifikacija funkcionalnih zahteva

U ovom odeljku su opisani funkcionalni zahtevi koje je potrebno da ispunjava softversko rešenje za eFakultet. Funkcionalni zahtevi ovog softverskog rešenja su predstavljeni UML dijagramom slučajeva korišćenja, kao što je prikazano na slici 1.

Slučajevi korišćenja

Tabela 1 prikazuje opis slučaja korišćenja **“Prijava na prijemi ispit”**

Naziv	Prijava na prijemi ispit
Učesnici	Student
Preduslovi	Student ima pristup sistemu
Koraci	1. Student bira željeni studijski program 2. Studentu se prikazuju detalji studijskog programa 3. Student šalje zahtev za prijavu na željeni prijemni ispit za izabrani studijski program 4. Sistem prima i sačuvava poslatu prijavu 5. Sistem obaveštava studenta o statusu poslate prijave
Rezultat	Student je uspešno poslao prijavu za prijemi ispit
Izuzeci	Sistem nije uspeo da sačuva studentovu prijavu

Tabela 1 - Opis slučajeva korišćenja **“Prijava na prijemi ispit”**

Tabela 2 prikazuje opis slučaja korišćenja **“Zahtev za izdavanje diplome”**

Naziv	Zahtev za izdavanje diplome
Učesnici	Student
Preduslovi	Student ima pristup sistemu
Koraci	1. Student bira opciju za slanje zahteva za kreiranje diplome 2. Student čeka odgovor
Rezultat	Student dobija potvrđan ili odričan odgovor
Izuzeci	Sistem nije uspeo da sačuva zahtev

Tabela 2 - Opis slučajeva korišćenja **“Zahtev za izdavanje diplome”**

Tabela 3 prikazuje opis slučaja korišćenja “**Provera uslova i izdavanje diplome**”

Naziv	Provera uslova i izdavanje diplome
Učesnici	Administrator
Preduslovi	Administrator ima pristup sistemu
Koraci	<ol style="list-style-type: none"> 1. Administrator bira opciju za prikaz svih validnih poslatih zahteva za kreiranje diplome 2. Sistem vraća sve validne zahteve za kreiranje diplome 3. Administrator ima uvid u uslove zahteva 4. Administrator bira između opcija za odobravanje ili odbijanje zahteva 5. Administrator kreira studentovu diplomu 6. Administrator šalje odgovor studentu
Rezultat	Kreirana je diploma i odgovor je poslat
Izuzeci	Sistem nije uspeo da kreira diplomu, zahtev nije uspešno poslat

Tabela 3 - Opis sličajeva korišćenja “**Provera uslova i izdavanje diplome**”

Tabela 4 prikazuje opis slučaja korišćenja “**Kreiranje konkursa**”

Naziv	Kreiranje konkursa
Učesnici	Administrator
Preduslovi	Administrator ima pristup sistemu
Koraci	<ol style="list-style-type: none"> 1. Administrator bira opciju za kreiranje konkursa 2. Administrator popunjava formu za kreiranje konkursa 3. Administrator šalje podatke sistemu za kreiranje
Rezultat	Konkurs je kreiran
Izuzeci	Sistem nije uspeo da kreira konkurs

Tabela 4 - Opis sličajeva korišćenja “**Kreiranje konkursa**”

Tabela 5 prikazuje opis slučaja korišćenja “**Kreiranje studentskog programa**”

Naziv	Kreiranje studentskog programa
Učesnici	Administrator
Preduslovi	Administrator ima pristup sistemu
Koraci	<ol style="list-style-type: none"> 1. Administrator bira opciju za kreiranje studentskog programa 2. Administrator popunjava formu za kreiranje studentskog programa 3. Administrator šalje podatke sistemu za kreiranje
Rezultat	Studentski program je kreiran
Izuzeci	Sistem nije uspeo da kreira studentski program

Tabela 5 - Opis sličajeva korišćenja “**Kreiranje studentskog programa**”

Tabela 6 prikazuje opis slučaja korišćenja “**Kreiranje departmana**”

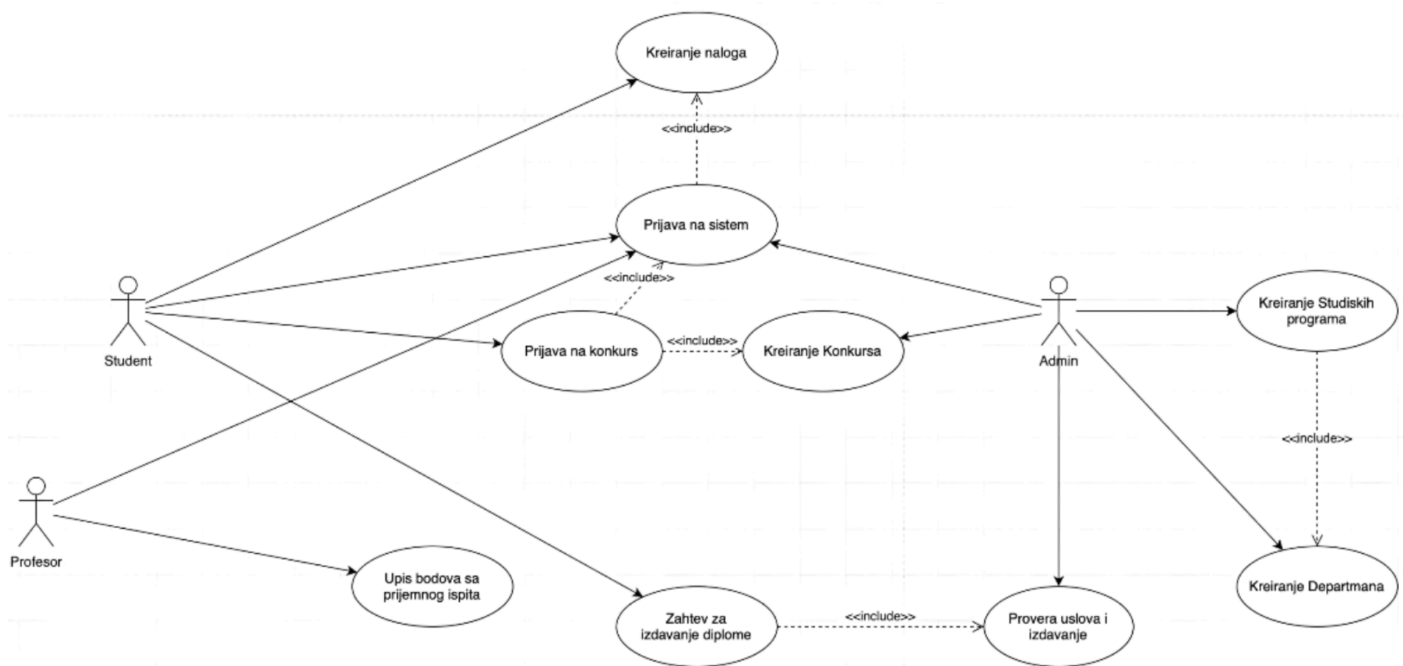
Naziv	Kreiranje departmana
Učesnici	Administrator
Preduslovi	Administrator ima pristup sistemu
Koraci	1. Administrator bira opciju za kreiranje departmana 2. Administrator popunjava formu za kreiranje departmana 3. Administrator šalje podatke sistemu za kreiranje
Rezultat	Departmana je kreiran
Izuzeci	Sistem nije uspeo da kreira departman

Tabela 6 - Opis sličajeva korišćenja “**Kreiranje departmana**”

Tabela 7 prikazuje opis slučaja korišćenja “**Upis bodova sa primenog ispita**”

Naziv	Upis bodova sa prijemnog ispita
Učesnici	Profesor
Preduslovi	Profesor ima pristup sistemu
Koraci	1. Profesor ima opciju za upis bodova sa prijemnog ispita 2. Profesor unosi podatke u formu 3. Profesor šalje podatke iz forme sistemu
Rezultat	Podaci su upisani u listu rezultata za određeni prijemni ispit
Izuzeci	Sistem nije uspeo da sačuva podatke

Tabela 7 - Opis sličajeva korišćenja “**Upis bodova sa primenog ispita**”



Slika 1 – UML dijagram slučaja korišćenja

5.2 Specifikacija nefunkcionalnih zahteva

Pomoću specifikacije nefunkcionalnih zahteva se definišu svojstva softverskog rešenja koja su potrebna da bi se dati problem rešio, ali ne predstavljaju njegove funkcionalnosti.

1. Kontejnerizacija:

Sve servise i baze podataka potrebno je pokrenuti kao Docker [4]kontejnere koristeći Docker Compose alat. Ovo će omogućiti konzistentno i lako upravljanje okruženjem za razvoj, testiranje i produkciju.

2. Single Sign-On (SSO):

Sistem eFakultet mora implementirati Single Sign-On (SSO) autentifikaciju, omogućavajući korisnicima (studentima, profesorima i administraciji) da se prijave u sve povezane aplikacije koristeći jedan skup kredencijala. Ovo će poboljšati korisničko iskustvo i smanjiti potrebu za upravljanjem više lozinki.

3. Upotrebljivost:

Sistem mora biti jednostavan za korišćenje, sa jednostavnim korisničkim interfejsom koji omogućava korisnicima da lako pristupe i koriste sve funkcionalnosti. Ovo uključuje konzistentan dizajn, jasne navigacione putanje i dostupnost svih potrebnih informacija na dohvat ruke.

6. Specifikacija dizajna

Ovo poglavlje objašnjava dizajn softverskog rešenja za sistem eFakultet.

Arhitektura sistema

Sistem eFakultet je veb aplikacija koja omogućava modularnost, skalabilnost i jednostavno održavanje. Sistem se sastoji od više komponenti koje međusobno komuniciraju preko API interfejsa koristeći HTTP[5] protokol.

- **Korisnički interfejs (frontend):** Omogućava korisnicima interakciju sa sistemom kroz web aplikaciju.
- **Backend:** Obradu poslovne logike, upravljanje podacima i komunikaciju sa bazom podataka.
- **Autentifikacioni servis:** Upravljanje autentifikacijom korisnika koristeći Single Sign-On (SSO).
- **Baza podataka:** Čuvanje svih relevantnih podataka za sistem.

Komponente sistema su prikazane na dijagramu klasa, što je prikazano na slici 2. Klasni diagram prikazuje ključne entitete sistema eFakultet i njihove međusobne odnose.

Centralni deo dijagrama čini klasa "User", koja reprezentuje korisnike sistema i sadrži njihove lične podatke i podatke za autentifikaciju na sistem. Svaki korisnik ima tačno jednu ulogu, što je modelovano vezom sa enumeracijom "eRole" koja definiše različite uloge korisnika, poput "Profesor", "Student" i "Administrator".

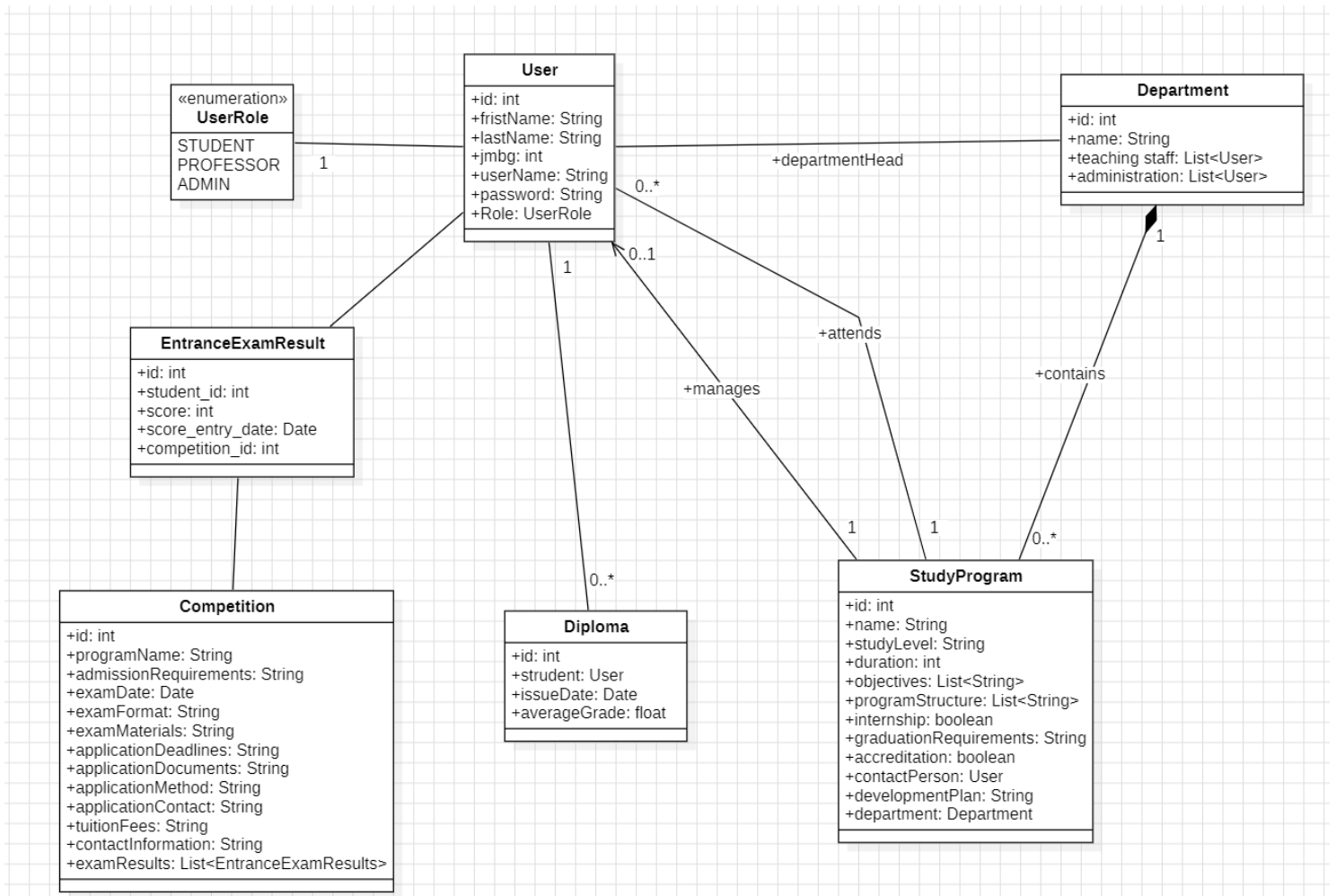
Korisnici mogu polagati prijemne ispite, i njihovi rezultati će biti evidentirani unutar sistema, što je predstavljeno klasom "EntranceExamResults". Svaki rezultat prijemnog ispita sadrži informacije o konkretnom korisniku i ishodu ispita. Ova klasa je povezana sa klasom "Competition", koja poseduje sve bitne podatke o prijemnom ispitu koji studenti mogu polagati.

Diplome su predstavljene klasom "Diploma", koja je povezana sa klasom "User" i sadrži sve bitne informacije o diplomi korisnika, uključujući kome diploma pripada, vreme kada je izdata i prosečnu ocenu korisnika.

Klasa "StudyProgram" je povezana sa klasom "User" i "Department". Ova klasa sadrži sve bitne informacije o studijskom programu, uključujući ko je upravnik programa i listu studenata koji pohađaju taj program. Veza sa korisnikom omogućava praćenje studenata, dok veza sa departmanom definiše kojem departmanu program pripada.

Departmani su predstavljeni klasom "Department", koja je povezana sa klasama "User" i "StudyProgram". Veza sa korisnikom omogućava evidentiranje liste profesorskog osoblja koje radi u tom departmanu, dok veza sa studijskim programima omogućava organizaciju i raspodelu studijskih programa unutar departmana.

Ovaj dijagram klasa daje jasan uvid u strukturu sistema eFakultet i načine na koje su podaci povezani i obrađeni, čime se postiže efikasna organizacija i upravljanje obrazovnim procesima i evidencijama.



Slika 2 - UML Klasni dijagram

7. Implementacija

Ovo poglavlje prikazuje način na koji su implementirane neke od funkcija sistema eFakultet. Objašnjena je implementacija ključnih komponenti sistema, uključujući implementaciju autentifikacije korisnika, kreiranje rezultata prijemnog ispita studenta i kreiranje prijemnog ispita.

Implementacija autentifikacije korisnika

U ovom odeljku je predstavljena implementacija funkcije za autentifikaciju korisnika putem izdvojenog servisa za autentifikaciju, koja se koristi za autentifikaciju korisnika kroz poslani JWT token, kao što je prikazano na listingu 1.

Ova funkcija Auth služi za autentifikaciju korisnika u sistemu eFakultet putem JWT tokena. Nakon što se primi zahtev, dekodira se JSON telo zahteva i pokušava se verifikovati pristiglu JWT token autentifikacijom preko `jwtMaker.VerifyToken`. U slučaju neuspele verifikacije tokena, vraća se odgovarajuća greška sa statusom HTTP Unauthorized. U suprotnom, vraća se potvrda da je token validan sa statusom OK.

```
func (uh *UserHandler) Auth(w http.ResponseWriter, r *http.Request) {
    log.Println("req received")

    dec := json.NewDecoder(r.Body)

    var rt model.RegToken
    err := dec.Decode(&rt)
    if err != nil {
        log.Println(err)
        log.Println("Request decode error")
    }

    log.Println(rt.Token)

    _, err = uh.jwtMaker.VerifyToken(rt.Token)
    if err != nil {
        // If the token verification fails, return an error
        log.Println("error in token verification")
        sendErrorMessage(w, err.Error(), http.StatusUnauthorized)
        return
    }

    sendErrorMessage(w, "Valid", http.StatusOK)
}
```

Listing 1 - Implementacija autentifikacije korisnika

Implementacija kreiranja rezultata prijemog ispita studenta:

Ovaj odeljak opisuje implementaciju funkcije za kreiranje rezultata prijemnog ispita studenta u sistemu eFakultet. Funkcija detaljno objašnjava proces validacije, dekodiranja JSON formata, generisanja jedinstvenog identifikatora za rezultat ispita, te prosleđivanju podataka u repozitorijum sistema koji komunicira sa bazom podataka kao što je prikazano na listingu 2.

U ovoj funkciji CreateUserExamResult obrađuju se zahtevi za kreiranje rezultata ispita za korisnika. Na početku se proverava tip formata koji je poslat u zaglavlju zahteva kako bi se osiguralo da se koristi JSON format. Nakon uspešnog dekodiranja tela zahteva, rezultat ispita dobija novi identifikator objekta (ID) i datum unosa rezultata. Zatim se rezultat ispita čuva u repozitorijumu, a ako dođe do greške pri čuvanju, vraća se odgovarajuća greška sa odgovarajućim HTTP statusom. Na kraju, korisniku se šalje potvrda da je rezultat ispita uspešno unet.

```
func (nh *newHandler) CreateUserExamResult(w http.ResponseWriter, req *http.Request) {
    log.Println("Usli u CreateUserExamResult")

    contentType := req.Header.Get("Content-Type")
    mediatype, _, err := mime.ParseMediaType(contentType)
    if err != nil {
        log.Println("Error cant mimi.ParseMediaType")
        sendErrorMessage(w, err.Error(), http.StatusBadRequest)
        return
    }

    if mediatype != "application/json" {
        err := errors.New("expect application/json Content-Type")
        sendErrorMessage(w, err.Error(), http.StatusUnsupportedMediaType)
        return
    }

    log.Println("Pre decodeBody")
    rt, err := decodeExamResultBody(req.Body)
    if err != nil {
        log.Println("Decode: ", err)
        sendErrorMessage(w, "Error when decoding data", http.StatusBadRequest)
        return
    }

    rt.ID = primitive.NewObjectID()
    rt.ScoreEntryDate = time.Now()

    err = nh.repo.InsertUserExamResult(rt)
    if err != nil {
        log.Println(err)
        sendErrorMessage(w, "Cant insert user exam result", http.StatusInternalServerError)
        return
    }

    sendErrorMessage(w, "User exam result inserted", http.StatusCreated)
}
```

Listing 2 - Kreiranja rezultata prijemog ispita studenta

Kreiranje prijemnog ispita

U ovom odeljku opisuje se implementacija funkcije za kreiranje prijemnog ispita u sistemu eFakultet, kao što je prikazano na listingu 3.

U funkciji CreateCompetition koja se koristi za kreiranje takmičenja u sistemu eFakultet, prvo se proverava tip medija u zaglavlju zahteva kako bi se osiguralo da se koristi JSON format. Nakon uspešnog dekodiranja tela zahteva, takmičenje dobija novi identifikator objekta (ID). Zatim se takmičenje čuva u repozitorijumu sistema, a ako dođe do greške pri čuvanju, vraća se odgovarajuća greška sa statusom HTTP Internal Server Error. Na kraju, korisniku se šalje potvrda da je takmičenje uspešno kreirano sa odgovarajućim statusom HTTP Created.

```
func (nh *newHandler) CreateCompetition(w http.ResponseWriter, req *http.Request) {
    log.Println("Usli u Create")
    contentType := req.Header.Get("Content-Type")
    mediatype, _, err := mime.ParseMediaType(contentType)
    if err != nil {
        log.Println("Error cant mimi.ParseMediaType")
        sendErrorMessage(w, err.Error(), http.StatusBadRequest)
        return
    }

    if mediatype != "application/json" {
        err := errors.New("expect application/json Content-Type")
        sendErrorMessage(w, err.Error(), http.StatusUnsupportedMediaType)
        return
    }

    rt, err := decodeCompetitionBody(req.Body)
    if err != nil {
        log.Println("Decode: ", err)
        sendErrorMessage(w, "Error when decoding data", http.StatusBadRequest)
        return
    }

    rt.ID = primitive.NewObjectID()

    log.Println("Competition: ", rt)
    var err error
    err = nh.repo.InsertCompetition(rt)
    if err != nil {
        log.Println(err)
        sendErrorMessage(w, err.Error(), http.StatusInternalServerError)
        return
    }

    sendErrorMessage(w, "Competition Created", http.StatusCreated)
}
```

Listing 3 - Kreiranje prijemnog ispita

Kriranje liste rezultata prijemog ispita

U ovom pasusu opisuje se funkcija `GetAllExamResultsByCompetitionId` koja se koristi za kreiranje liste rezultata prijemnog ispita na osnovu datog identifikatora prijemnog ispita, kao što je prikazano u listingu 4.

Ova funkcija `GetAllExamResultsByCompetitionId` koristi se za dobijanje svih rezultata ispita na osnovu identifikatora prijemnog ispita u sistemu eFakultet. Koristi kontekst sa vremenskim ograničenjem od 5 sekundi kako bi se obezbedilo pravilno upravljanje resursima. Pristupa se kolekciji rezultata ispita, filtriraju se rezultati na osnovu datog `competitionId`-a, iterira se kroz rezultat koristeći kursor, dekodira se svaki rezultat i agregiraju se u strukturu `ExamResults`. Na kraju, vraća se lista rezultata ili greška ako dođe do problema prilikom pristupa ili obrade podataka.

```
func (nr *NewRepository) GetAllExamResultsByCompetitionId(competitionId string) (*model.ExamResults, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    // competitionId = "6658d76eed49f71587b7c4b1" // Note: This line is for testing purposes and can be removed

    resultatCollection, err := nr.getCollection(5)
    if err != nil {
        log.Println("Error getting collection: ", err)
        return nil, err
    }

    var examResults model.ExamResults

    cursor, err := resultatCollection.Find(ctx, bson.M{"competitionId": competitionId})
    if err != nil {
        log.Println("Error finding exam results: ", err)
        return nil, err
    }
    defer cursor.Close(ctx)

    log.Println("Pre petlje")

    for cursor.Next(ctx) {
        log.Println("Petlja")
        var result model.ExamResult
        if err := cursor.Decode(&result); err != nil {
            log.Println("Error decoding exam result:", err)
            return nil, err
        }
        log.Println("Decoded result:", result)
        examResults = append(examResults, &result)
    }

    log.Println("Posle petlje")
    if err := cursor.Err(); err != nil {
        log.Println("Cursor error:", err)
        return nil, err
    }

    log.Println("Results: ", examResults)

    return &examResults, nil
}
```

Listing 4 - Kriranje liste rezultata prijemog ispita

8. Demonstracija

Ovo poglavlje ilustruje praktičnu primenu aplikacije eFakultet. Prikazani su scenariji koji obuhvataju upotrebu softvera za administraciju, analizu i pregled podataka o studijskim programima, prijemnim ispitima i akademskim rezultatima studenata.

Prikaz svih aktivnih prijemnih ispita

Kada korisnik pristupi sistemu prikazuje se stranica na slici 2. Ova stranica prikazuje listu svih aktivnih prijemnih ispita koji sadrže neke osnovne podatke kao:

Admission Requirements(uslovi prijave),
Exam Date(datum ispita),
Application Method(način prijave) i
Tuition Fees(troškovi školarine).

Na ovoj stranici administrator ima opciju kreiranja novog prijemnog ispita za odabrani studijski program. Student ima mogućnost uvida u ostale detalje aktivnog prijemnog ispita prilikom pritiska na naziv prijemnog ispita.

Competitions List		
Create Competition		
Computer Science Admission Requirements: High school diploma, minimum GPA of 3.0 Exam Date: 2024-06-15 Application Method: Online application Tuition Fees: \$5000 per semester	Electrical Engineering Admission Requirements: High school diploma, minimum GPA of 3.5 Exam Date: 2024-07-20 Application Method: Online application Tuition Fees: \$6000 per semester	Business Administration Admission Requirements: Bachelor's degree, minimum GPA of 3.0, work experience preferred Exam Date: 2024-08-10 Application Method: Online application Tuition Fees: \$7000 per semester
Nursing Admission Requirements: Associate degree in Nursing, minimum GPA of 3.2, clinical experience required Exam Date: 2024-09-05 Application Method: Online application Tuition Fees: \$8000 per semester		

Slika 2 - Prikaz svih aktivnih prijemnih ispita

Prikaz detalja odabraniog prijemnog ispita

Kada korisnik odabere pregled detalja prijemnog ispita prikazuje se stranica na slici 3. Na ovoj stranici korisnik ima uvid u sve detalje odabranog ispita kao i opciju za uvid u listu rezultata za isti prijemi ispit. Ako lista ne postoji prikazaće se prazna stranica bez popunjene liste rezultata.

Student ima opciju za prijavu na prijemni ispit. Dok Profesor ima opciju da unese konačne rezultate prijemnog ispita.

Competition Details

Competition ID: 66620923bdf6221a0d6cee6
Program Name: Computer Science
Admission Requirements: High school diploma, minimum GPA of 3.0
Exam Date: 2024-06-15
Exam Format: Written exam and interview
Exam Materials: Mathematics, Programming
Application Deadlines: 2024-05-01
Application Documents: Application form, transcripts, recommendation letters
Application Method: Online application
Application Contact: admissions@university.edu
Tuition Fees: \$5000 per semester
Contact Information: University Admissions Office, 123 University St, City, Country

[Register](#)[Add Results](#)[Results](#)

Slika 3 - Prikaz detalja odabraniog prijemnog ispita

Prikaz liste rezultata prijemnog ispita

Kada korisnik izabere opciju za uvid u listu rezultata prijemnog ispita prikaže se stranica kao na slici 4, koja sadrži tabelu sa imenima, bodovima i datumima unosa podataka za svakog studenta.

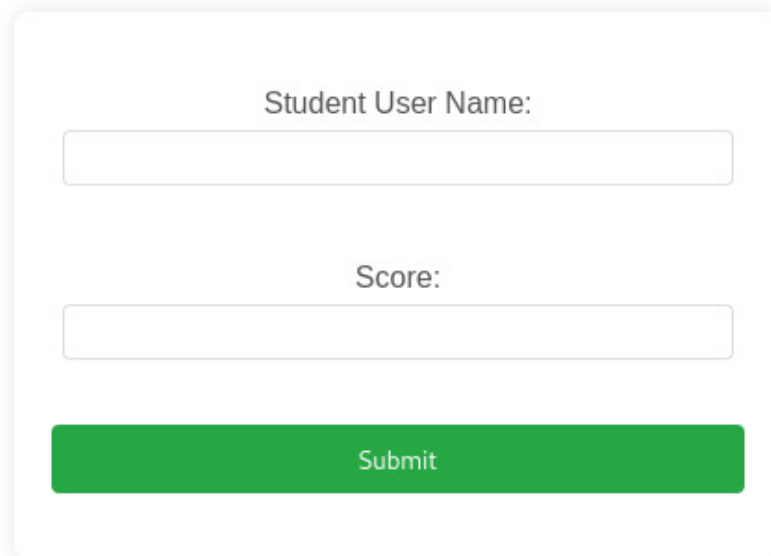
Exam Results		
User Name	Score	Score Entry Date
example_user	85	6/14/2024, 12:21:35 AM
blabla	91	6/14/2024, 12:21:59 AM
blabla1	91	6/14/2024, 12:22:06 AM
blabla2	95	6/14/2024, 12:22:11 AM
blabla3	100	6/14/2024, 12:22:19 AM
blabla4	90	6/14/2024, 12:22:27 AM
blabla5	100	6/14/2024, 12:22:46 AM
zika	100	6/15/2024, 12:14:21 AM
pera1	95	6/15/2024, 12:32:30 AM
pera2	100	6/15/2024, 12:32:55 AM
pera3	99	6/15/2024, 12:33:07 AM

Slika 4 - Prikaz liste rezultata prijemnog ispita

Unos konačnih rezultata prijemnog ispita

Kada profesor odabere opciju za unos konačnih bodova prijemnog ispita prikaže se stranica na slici 5, koja sadrži polje za studentovo korisničko ime koje je jedinstveno za svakog korisnika i broj bodova koje je taj student osvojio na prijemnom ispitu.

Create Exam Result



Student User Name:

Score:

Submit

Slika 5 - Unos konačnih rezultata prijemnog ispita

Prikaz liste studiskih programa

Korisnik ima opciju uvida u listu svih studiskih programa, prikaz na slici 6. Prikazani su osnovni podaci o studiskom programu kao:

- Study Level(nivo studiranja),
- Duration(trajanje studiranja),
- Internship(da li studiski program podržava opciju prakse),
- Accreditation(da li studiski program podržava akreditaciju),
- Contact Person ID(Identifikacioni podatak odgovorne osoba za studiski program),
- Development Plan(opis plana razvijanja)

Na ovoj stranici administrator ima opciju kreiranja novog studiskog programa u sklopu odabranog departmana. Korisnik imaju mogućnost uvida u ostale detalje odabranog studiskog programa prilikom pritiska na naziv studiskog programa.

Study Programs

Create Study Program

Computer Science

Study Level: Bachelor
Duration: 4 years
Internship: Yes
Accreditation: Yes
Contact Person ID: 60d5ec4f5b3b0f6a4dbe6ac9
Development Plan: Continuous improvement through industry feedback and academic research.

Electrical Engineering

Study Level: Bachelor
Duration: 4 years
Internship: Yes
Accreditation: Yes
Contact Person ID: 60d5ec4f5b3b0f6a4dbe6acb
Development Plan: Continuous improvement through practical workshops and industry collaboration.

Business Administration

Study Level: Master
Duration: 2 years
Internship: No
Accreditation: Yes
Contact Person ID: 60d5ec4f5b3b0f6a4dbe6acc
Development Plan: Integration of emerging business trends and technologies into the curriculum.

Nursing

Study Level: Bachelor
Duration: 4 years
Internship: Yes
Accreditation: Yes
Contact Person ID: 60d5ec4f5b3b0f6a4dbe6acd
Development Plan: Enhancement of clinical simulation facilities and partnerships with healthcare providers.

Slika 6 - Prikaz liste studiskih programa

Prikaz svih detalja odabranog studiskog programa

Prilikom pritiska na naziv studiskog programa korisniku se prikazuje stranica na slici 7, koja sadrži tabelu sa svim detaljima odabranog studiskog programa.

Study Program Details

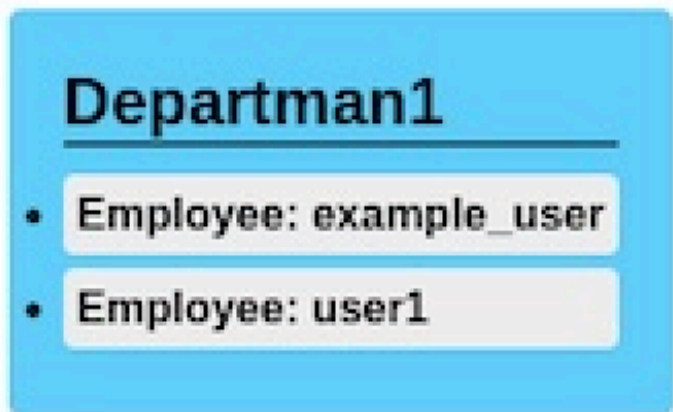
Program ID:	665e165a7819bcef4a231ba6
Name:	Computer Science
Study Level:	Bachelor
Duration:	4 years
Objectives:	To provide students with a solid foundation in computer science and practical experience.
Program Structure:	Year 1: Basics, Year 2: Intermediate, Year 3: Advanced, Year 4: Capstone Project
Internship:	Yes
Graduation Requirements:	Completion of all coursework, a capstone project, and an internship.
Accreditation:	Yes
Contact Person ID:	60d5ec4f5b3b0f6a4dbe6ac9
Development Plan:	Continuous improvement through industry feedback and academic research.
Department ID:	665cef9baf82e83a86824acd

Slika 7 - Prikaz svih detalja odabranog studiskog programa

Prikaz liste departmana

Korisniku odabirom opcije za pregled svih departmana se prikazuje stranica na slici 8, koja prikazuje listu svih departmana, gde svaki departman sadrži listu zaposlenog osoblja.

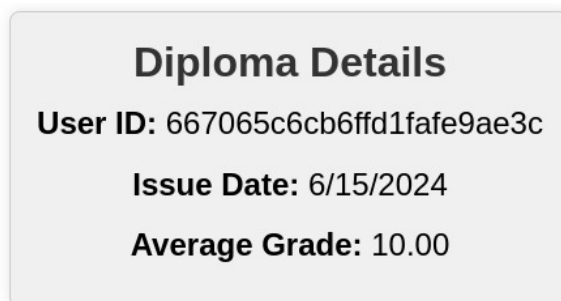
Departments



Slika 8 - Prikaz liste departmana

Prikaz detalja diplome studenta

Kada student pošalje zahtev za kreiranje diplome i zahtev je odobren, diploma je kreirana. Student ima opciju uvida u detalje svoje diplome, kao što je prikazno na slici 9. Diploma sadrži identifikacioni broj korisnika, datum izdavanja, i prosečnu ocenu studenta.



Slika 9 - Prikaz detalja diplome studenta

Ova demonstracija prikazuje kako korisnici i zaposleni mogu koristiti aplikaciju eFakultet za kreiranje i pregled prijemnih ispita, uvid u konačne liste rezultata sa prijemnih ispita, izdavanje diploma studentima, pregled studijskih programa i prijavu na prijemne ispite. Aplikacija omogućava efikasno prikupljanje, analizu i pregled podataka, kao i komunikaciju i razmenu informacija. Sistem je dizajniran da olakša upravljanje akademskim procesima na visokoškolskim ustanovama.

Zaključak

U ovom radu je predstavljen projekat eFakultet, razvijen korišćenjem tehnologija kao što su Go[2] za backend i React za frontend [3], uz Docker [4] kontejnerizaciju. Sistem omogućava efikasno upravljanje akademskim procesima, uključujući objavu konkursa prijemnih ispita, prijavu na njih, kreiranje rezultata prijemnih ispita i izdavanje diploma studentima.

Kroz implementaciju različitih funkcionalnosti, kao što su pregled studijskih programa, uvid u konačne rezultate prijemnih ispita i automatizacija izdavanja diploma, demonstrirano je kako softversko rešenje može značajno unaprediti administrativne i operativne procese u visokoškolskim ustanovama. Osim toga, korišćenjem SSO (Single Sign-On) sistema, aplikacija omogućava korisnicima jednostavan i siguran pristup, što dodatno olakšava upotrebu sistema.

Ovaj projekat pokazuje kako tehnologija može biti iskorišćena za unapređenje akademskih procesa, pružajući korisnicima pouzdane i lako dostupne informacije, dok zaposlenima olakšava administrativne zadatke i poboljšava efikasnost rada.

10. Literatura

- [1] UML. 2024. UML Documentation, <https://www.uml.org/>.
- [2] Go. 2024. Go Programming Language, <https://go.dev/>.
- [3] React. 2024. React Documentation, <https://react.dev/>.
- [4] Docker. 2024. Docker Documentation, <https://www.docker.com/>.
- [5] HTTP. 2024. HTTP Documentation, <https://www.w3.org/Protocols/>.
- [6] FTN. 2024. FTN Web Servis, <https://ssluzba.ftn.uns.ac.rs/touchmain/>.
- [7] Google. 2024. <https://www.google.com/>.
- [8] JavaScript. 2024. JavaScript Documentation, [https://developer.mozilla.org/en-US/docs/](https://developer.mozilla.org/en-US/docs/Web/JavaScript)
[Web/JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript).
- [9] Facebook. 2024. <https://www.facebook.com/>.
- [10] HTML. 2024. HTML Documentation, [https://developer.mozilla.org/en-US/docs/](https://developer.mozilla.org/en-US/docs/Web/HTML)
[Web/HTML](https://developer.mozilla.org/en-US/docs/Web/HTML).