Seminarski rad

Tema: Aplikacija za elektronsku evidenciju seminarskih radova studenata

Vojislav Stefanović

Sadržaj

[Verbalni opis 3](#_Toc75174073)

[Opis slučajeva korišćenja 4](#_Toc75174074)

[Opis procesa – dijagrami sekvenci 7](#_Toc75174075)

[Konceptualni model 8](#_Toc75174076)

[Specifikacija REST API – ja 9](#_Toc75174077)

[Opis korišćenih tehnologija 12](#_Toc75174078)

[React 12](#_Toc75174079)

[Express 12](#_Toc75174080)

[Axois 13](#_Toc75174081)

[Mysql 13](#_Toc75174082)

[TypeORM 13](#_Toc75174083)

[Reprezentativni delovi koda 14](#_Toc75174084)

[Link ka udaljenom repozitorijumu 14](#_Toc75174085)

# Verbalni opis

Potrebno je napraviti aplikaciju u kojoj će studenti postavljati seminarske radove, a profesori ih ocenjivati.

Aplikacija treba da prepozna dve vrste korisnika: studenta i profesora.

Student će moći da vidi sve predmete koje sluša kao i obaveze na svakom od njih. Pored toga student može predati seminarski rad, izmeniti ga ukoliko nije ocenjen i pratiti stanje rada (da li je ocenjen).

Profesor može oceniti predati seminarski rad, kreirati novi, izmeniti i obrisati seminarski rad.

Prilikom analize sistema, prepoznato je da jedan student može slušati više predmeta kao i da jedan predmet može slušati više studenata.

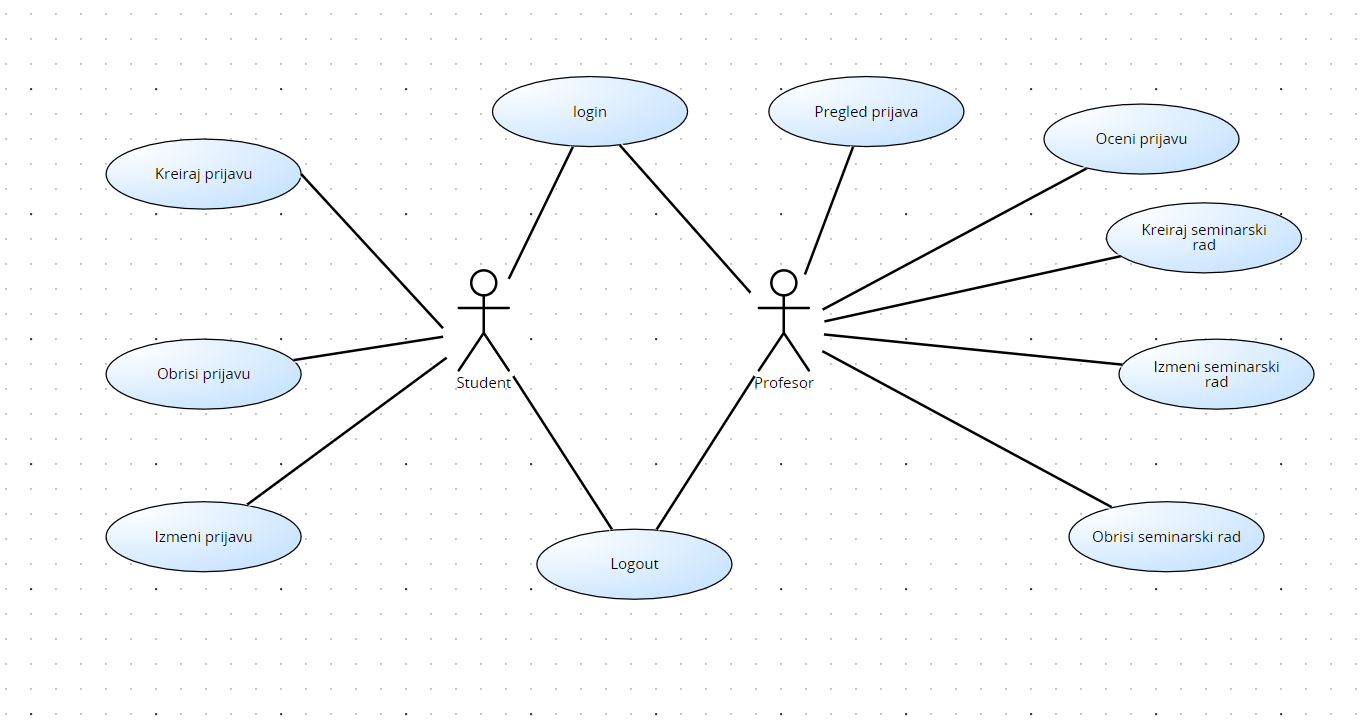
Jedan profesor može predavati na više predmeta, a na jednom predmetu može predavati više profesora.

U okviru jednog predmeta mogu se braniti više seminarskih radova, dok se jedan seminarski rad vezuje za jedan predmet.

Jedan seminarski rad može braniti više studenata i jedan student može braniti više seminarskih radova. U ovom slučaju potrebno je pamtiti osvojeni broj poena studenta kao i mentora za njegov seminarski rad.

# Opis slučajeva korišćenja

U sistemu smo prepoznali sledeće slučajeve korišćenja:

* Prijava na sistem
* Odjava sa sistema
* Pregled prijava
* Kreiranje prijave
* Izmena prijave
* Brisanje prijave
* Izmena proizvoda
* Ocenjivanje prijave
* Kreiranje seminarskog rada
* Izmena seminarskog rada
* Brisanje seminarskog rada

Slika 1 Dijagram slučajeva korišćenja

## 

**SK1: Slučaj korišćenja – Kreiranje seminarkog rada**

**Aktori SK**

Profesor

**Učesnici SK**

Profesor i sistem

**Preduslov: Sistem je uključen i profesor je ulogovan pod svojom šifrom. Sistem prikazuje profesoru formu za rad sa seminarskim radom.**

**Osnovni scenario SK**

1. Profesor unosi vrednosti za kreiranje seminarskog rada. (APUSO)
2. Profesor poziva sistem da kreira seminarski rad sa datim podacima. (APSO)
3. Sistem kreira seminarski rad sa datim podacima. (SO)
4. Sistem prikazuje profesoru poruku: „Sistem je usepšno kreirao seminarski rad“. (IA)

Alternativna scenarija

* 1. Ukoliko sistem ne može da kreira seminarski rad on prikazuje profesoru objavu: „Sistem ne može da kreira seminarski rad“.(IA)

**SK2: Slučaj korišćenja – Kreiranje prijave**

**Aktori SK**

Student

**Učesnici SK**

Student i sistem

**Preduslov: Sistem je uključen, student je ulogovan pod svojom šifrom. Sistem prikazuje studentu formu za rad sa prijavom.**

**Osnovni scenario SK**

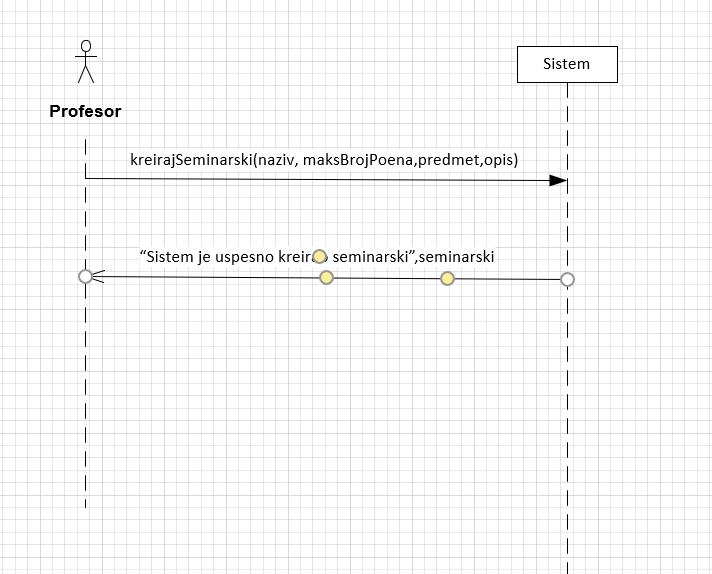
1. Student bira seminarski rad koji želi da prijavi. (APUSO)
2. Student unosi naziv teme i mentora. (APUSO)
3. Student unosi fajl sa radom. (APUSO)
4. Student poziva sistem da kreira prijavu sa datim podacima. (APSO)
5. Sistem čuva fajl koji je poslat uz prijavu.(SO)
6. Sistem kreira prijavu sa datim podacima. (SO)
7. Sistem prikazuje korsiniku poruku: „Sistem je usepšno sačuvao prijavu“. (IA)

Alternativna scenarija

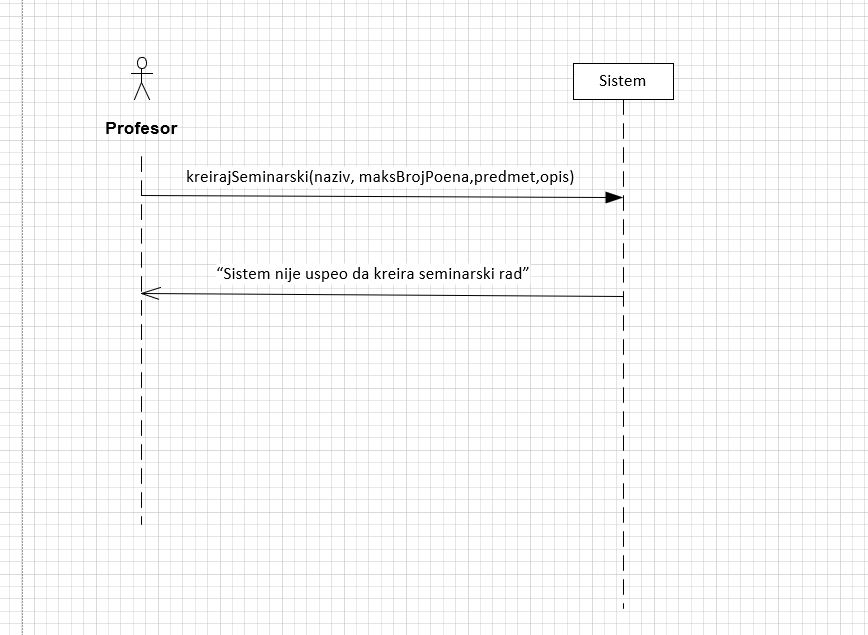
* 1. Ukoliko sistem ne može da sačuva fajl on prikazuje studentu poruku: „Sistem ne može da sačuva fajl“.(IA) prekida se izvršenje SK
  2. Ukoliko sistem ne može da sačuva prijavu on prikazuje studentu poruku: „Sistem ne može da sačuva prijavu“.

# Opis procesa – dijagrami sekvenci

## SK1: Kreiranje seminarskog rada

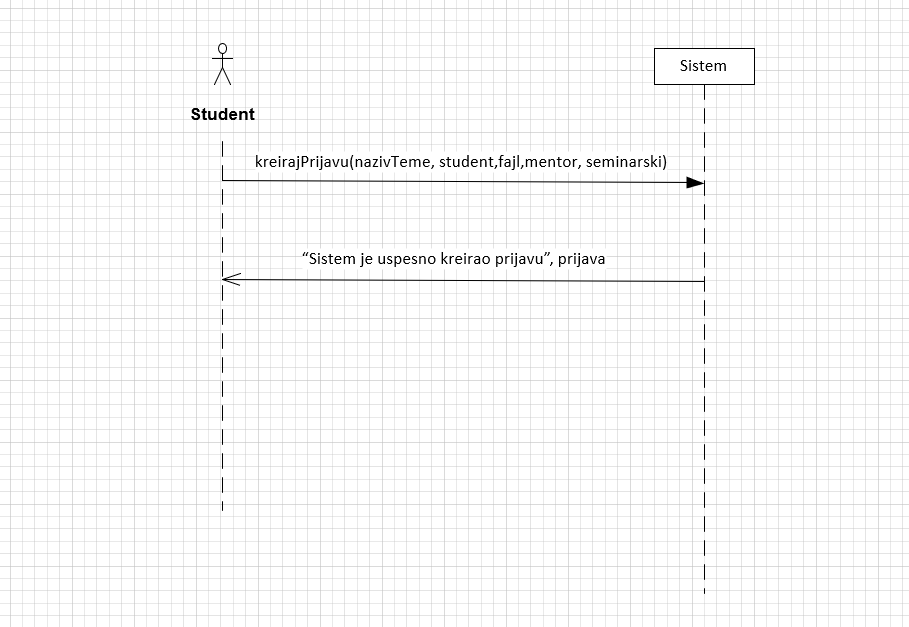


Slika SK1: Osnovni scenario

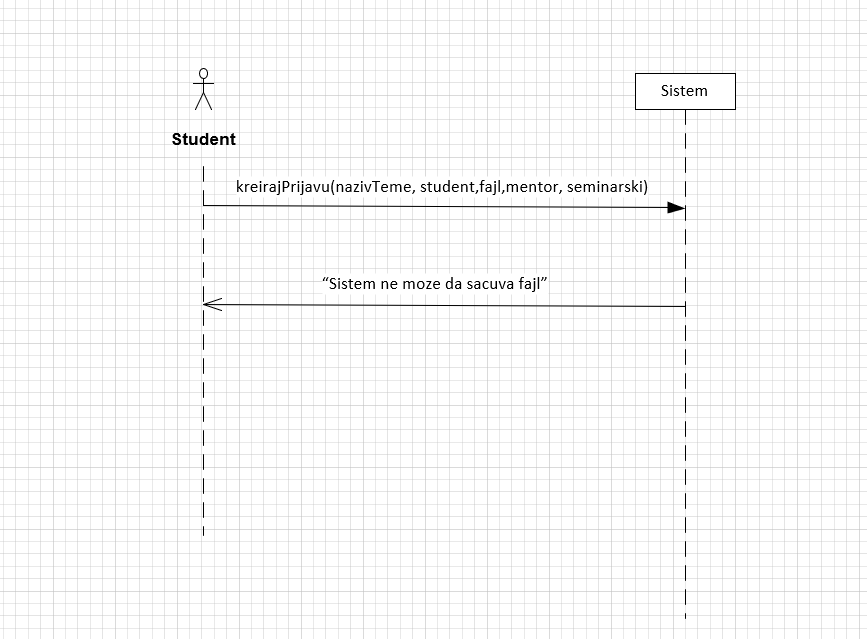


Slika SK1: Alternativni scenario

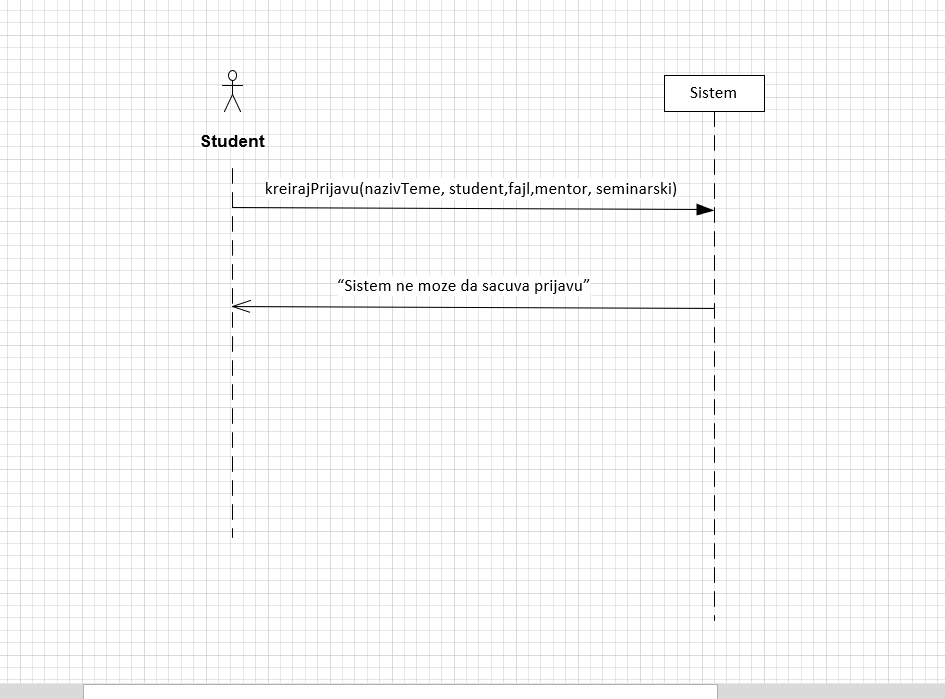
## SK2: Kreiranje prijave



Slika SK2: Osnovni scenario

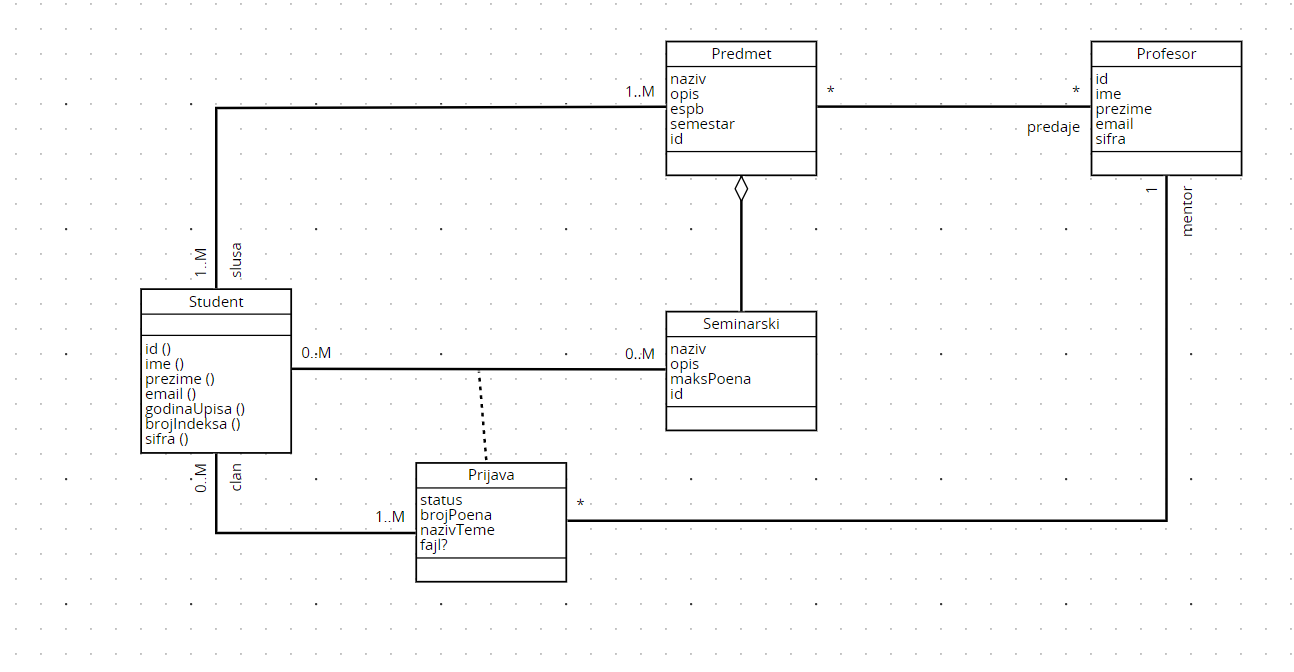


Slika SK2: Alternativni scenario – sistem ne moze da sacuva fajl



Slika SK2: Alternativno scenario – sistem ne moze da sacuva prijavu

# Konceptualni model



Slika 7 Konceptualni model

# Specifikacija REST API – ja

|  |  |
| --- | --- |
| Opis funkcije | Vrati sve seminarske |
| HTTP metoda | GET |
| URL | /seminarski |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  [      {          "id": 1,          "maksBrojPoena": 10,          "naziv": "php",          "opis": "afdg"      },      {          "id": 2,          "maksBrojPoena": 10,          "naziv": "laravel",          "opis": "aefdsg"      }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Vrati sve profesore |
| HTTP metoda | GET |
| URL | /profesor |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  [      {          "id": 1,          "ime": "dusan",          "email": "db@fon.rs",          "prezime": "barac",          "predaje": [              {                  "id": 4,                  "naziv": "iteh",                  "opis": "agsr",                  "espb": 6,                  "semestar": 7,                  "seminarski": [                      {                          "id": 1,                          "maksBrojPoena": 10,                          "naziv": "php",                          "opis": "afdg"                      }                  ]              }          ]      }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Kreiraj prijavu |
| HTTP metoda | POST |
| URL | /prijava |
| URL parametri | (nema ) |
| HTTP body parametri | Primer:      {          "studentId": 1,          "nazivTeme": "teagr",          "file": "itehL.pdf",          "seminarski": {              "id": 1          },          "mentor": {              "id": 1          }      } |
| Format HTTP body parametara | application/json |
| Izlazni parametri | Primer:      {          "studentId": 1,          "nazivTeme": "teagr",          "file": "itehL.pdf",          "status": "kreirana",          "seminarski": {              "id": 1          },          "mentor": {              "id": 1          }      } |
| Format izlaznih parametara | application/json |

# Opis korišćenih tehnologija

Prilikom izrade aplikacije korišćene su sledeće tehnologije:

* React – kreiranje poslovne logike na strani klijenta
* Express – kreiranje servera
* Axios – biblioteka za pravljenje AJAX zahteva
* Typeorm – objektno – relacioni maper
* Mysql – baza podatakaReact

## React

React predstavlja biblioteka za kreiranje korisničkog interfejsa i poslovne logike na klijentskoj strani. React služi za kreiranje SPA web aplikacija odnosno aplikacija koje imaju samo jednu html stranu i kojima glavnu ulogu ima javascript.

Glavni koncept u React – je komponenta.

Komponenta predstavlja nešto što može da se iscrta kao jedna celina.

Celokupni prikaz aplikacije dobija se kompozicijom raznih elemenata u okviru jedne glavne komponente koja se, najčešće, naziva App.

Komponente mogu definisane kao funkcija i kao klasa.

Svaka komponenta ima svoje atribute koje je dobila komponente roditelja i svoje stanje koje je ona sama kreirala. Komponenta može da menja svoje stanje ali ne može da menja svoje atribute. Prilikom promene stanja poziva se posebna funkcija koja to radi nakon čega se prikaz na ekranu automatski menja na svim mestima koja su se iscrtavala na osnovu tog stanja.

## Express

Predstavlja „framework“ za izradu serverske strane u Node.js okruženju. Za razliku od većine ostalih servera, Express ne podržava višenitno programiranje. Problem konkurentnosti on rešava pomoću asinhronog programiranja i „event loop“ – a.

Express aplikacije su zamišljene da rade kao REST servisi, a obrada zahteva klijenta vrši se na sledeći način:

* Pristiže zahtev klijenta
* Izvršavaju se funkcije koje su definisane da se izvrše nad svakim zahtevom („middleware“)
* Zahtev se rutira na osnovu svog URL – a
* Izvršava se funkcija pridodata za dati URL
* Šalje se odgovor korisniku

## Axois

Axios je biblioteka koja pojednostavljuje rad sa AJAX zahtevima. Axios omogućava kreiranje svih vrsta http zahteva kao i kreiranje zaglavlja. Kao odgovor sa servera, axios vraća korisniku originalni odgovor sa pridruženim metapodacima koje je axios izgenerisao.

## Mysql

Relaciona baza podataka koja je besplatna za korišćenje.

## TypeORM

Objektno – relacion maper u javascript – u. Pravila prevođenja vrše se anotacijom klasa u programskom kodu. Sva konfiguracija nalazi se fajlu ormconfig.json. TypeORM omogućava rad sa bazom na 2 načina:

* Preko repozitorijuma (Repository pattern)
* Preko menadžera

U prvom slučaju olakšan je rad sa bazom ali su mogućnost kontrole toka i prilagođavanje rada ograničeni dok je u drugom omogućena mnogo veća kontrola i kastomizacija na račun komplikovanijeg koda.

# Reprezentativni delovi koda

Putanje REST api – ja:

import { getFile } from "./actions/FajlActions";

import { vratiProfesore } from "./actions/ProfesorActions";

const upload = multer({ dest: path.resolve('file/') })

export interface Route {

    method: 'get' | 'post' | 'patch' | 'delete',

    route: string,

    action: any[],

}

export const Routes: Route[] = [{

    method: 'get',

    route: '/prijava',

    action: [vratiSvePrijave]

}, {

    method: 'post',

    route: '/prijava',

    action: [upload.single('file'), handleUpload, kreirajPrijavu]

}, {

    method: 'patch',

    route: '/prijava/oceni',

    action: [oceniPrijavu]

}, {

    method: 'patch',

    route: '/prijava/:seminarski',

    action: [nadjiPrijavu, upload.single('file'), handleUpload, izmeniPrijavu]

}, {

    method: 'delete',

    route: '/prijava/:seminarski',

    action: [nadjiPrijavu, obrisiPrijavu]

}, {

    method: 'get',

    route: '/seminarski',

    action: [vratiSveSeminarske]

}, {

    method: 'post',

    route: '/seminarski',

    action: [kreirajSeminarski]

}, {

    method: 'patch',

    route: '/seminarski/:id',

    action: [nadjiSeminarski, izmeniSeminarski]

}, {

    method: 'delete',

    route: '/seminarski/:id',

    action: [nadjiSeminarski, obrisiSeminarski]

}, {

    method: 'get',

    route: '/fajl/:filename',

    action: [getFile]

}, {

    method: 'get',

    route: '/profesor',

    action: [vratiProfesore]

}

];

Domenska klasa Prijava:

import { Column, Entity, JoinColumn, ManyToOne, PrimaryColumn } from "typeorm";

import { Profesor } from "./Profesor";

import { Seminarski } from "./Seminarski";

import { Student } from "./Student";

@Entity()

export class Prijava {

    @PrimaryColumn()

    studentId: number;

    @Column({ nullable: true })

    brojPoena: number;

    @ManyToOne(type => Student, { primary: true })

    @JoinColumn({ name: 'studentId' })

    student: Student;

    @ManyToOne(type => Seminarski, { eager: true, primary: true })

    seminarski: Seminarski

    @Column()

    nazivTeme: string

    @Column()

    file: string

    @ManyToOne(type => Profesor, { eager: true })

    mentor: Profesor;

    @Column()

    status: 'kreirana' | 'ocenjena'

}

Osnovna konfiguracija i pokretanje HTTPS servera:

import "reflect-metadata";

import { createConnection, getRepository } from "typeorm";

import \* as express from "express";

import \* as session from 'express-session'

import { Request, Response } from "express";

import { Routes } from "./routes";

import \* as cors from 'cors'

import \* as https from 'https'

import \* as fs from 'fs'

import { Student } from "./entity/Student";

import { Profesor } from "./entity/Profesor";

createConnection().then(async connection => {

    const key = fs.readFileSync('./key.pem', 'utf8');

    const cert = fs.readFileSync('./cert.pem', 'utf8');

    // create express app

    const app = express();

    app.use(cors({

        credentials: true,//protiv xss napada

        methods: ['GET', 'POST', 'PATCH', 'DELETE'],

        origin: 'http://localhost:3000'

    }));

    app.use(express.json());

    app.use(session({

        secret: 'adsfgdhtydadsfsafsjtiuyi',

        resave: false,

        saveUninitialized: false,

        cookie: {

            sameSite: 'none',

            secure: true,

            maxAge: 1000 \* 60 \* 10,//10min

            httpOnly: true,

        }

    }))

    app.post('/login', async (req, res) => {

        const { email, password } = req.body;

        const studenti = await getUser(Student, email, password);

        if (studenti.length !== 1) {

            const profesori = await getUser(Profesor, email, password);

            if (profesori.length !== 1) {

                res.sendStatus(404);

            } else {

                (req.session as any).user = profesori[0];

                req.session.save();

                res.json(profesori[0]);

            }

        } else {

            (req.session as any).user = studenti[0];

            req.session.save();

            console.log(studenti[0]);

            res.json(studenti[0]);

        }

    })

    app.post('/logout', async (request: Request, response: Response) => {

        request.session.destroy((err) => {

            if (err)

                response.sendStatus(500);

        })

        response.sendStatus(204);

    })

    app.get('/check', async (req, res) => {

        const user = (req.session as any).user;

        if (!user) {

            res.sendStatus(401);

            return;

        }

        if (user.brojIndeksa) {

            (req.session as any).user = await getRepository(Student).findOne(user.id)

        } else {

            (req.session as any).user = await getRepository(Profesor).findOne(user.id)

        }

        req.session.save();

        res.json((req.session as any).user);

    })

    /\*     app.use((req, res, next) => {

            const user = (req.session as any).user;

            if (!user) {

                res.sendStatus(403);

            } else {

                next();

            }

        }) \*/

    Routes.forEach(route => {

        app[route.method](route.route, ...route.action);

    });

    // setup express app here

    // ...

    // start express server

    const server = https.createServer({

        key: key,

        cert: cert,

    }, app)

    server.listen(process.env.PORT || 4000, () => console.log('app is listening'))

}).catch(error => console.log(error));

async function getUser(table: typeof Student | typeof Profesor, email: string, password: string) {

    const repository = getRepository(table);

    return await repository.find({

        where: {

            email: email,

            sifra: password,

        }

    })

}

Ajax poziv klijenta:

 const kreirajSeminarski = async (seminarski: Partial<Seminarski>, predmet: Predmet) => {

    const data = (await axios.post(SERVER\_URL + '/seminarski', {

      ...seminarski,

      predmet: {

        id: predmet.id

      }

    })).data;

    setUser(prev => {

      if (!prev || isStudent(prev)) {

        return prev;

      }

      return {

        ...prev, predaje: prev.predaje.map(element => {

          if (element.id === predmet.id) {

            return {

              ...element, seminarski: [...element.seminarski, data]

            }

          } else {

            return element;

          }

        })

      }

    })

  }

Funkcija za setovanje stanja polja forme:

export function setFormState<T>(setState: (val: T) => void) {

    return function (e: any) {

        const value = e.currentTarget.value;

        setState(value);

    }

}

# Link ka udaljenom repozitorijumu

https://github.com/Voja/seminarski