Razvoj klijentskih komponenti aplikacije za upravljanje rasporedom

Vučković, Klara

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

KLARA VUČKOVIĆ

RAZVOJ KLIJENTSKIH KOMPONENTI APLIKACIJE ZA UPRAVLJANJE RASPOREDOM

ZAVRŠNI RAD

Pula, rujan, 2019. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

KLARA VUČKOVIĆ

RAZVOJ KLIJENTSKIH KOMPONENTI APLIKACIJE ZA UPRAVLJANJE RASPOREDOM

ZAVRŠNI RAD

JMBAG: 0253040490, redovan student

Studijski smjer: Informatika

Predmet: Programsko inženjerstvo

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: Doc. dr. sc. Tihomir Orehovački

Pula, rujan, 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisana Klara Vučković, kandidat za prvostupnika Informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, Klara Vučković, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom razvoj klijentskih komponenti aplikacije za upravljanje rasporedom koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SADRŽAJ

1. Uvod

Raspored sati jest oblik zapisivanja plana održavanja određenih događaja u točno određenom vremenskom razdoblju. Dakle osnovni sadržaj rasporeda sati jest vremenski periodi u kojemu se namjeravaju izvršiti određeni zadaci, događaji, radnje ili pak slijed događaja kronološkog redoslijeda. Raspored uvelike olakšava organizaciju vremena bilo to u školstvu ili u privatnom životu prilikom organiziranja dana, tjedna ili pak mjeseca. Bilo da pričamo o školskim ustanovama, osobnom organiziranju vremena, rasporeda javnog prijevoza ili dr. kreiranje rasporeda može biti vrlo zahtjevan i dugotrajan proces. Ukoliko su resursi mali razvoj takvoga rasporeda ne predstavlja problem, no u slučaju školskih ustanova ili velikih firmi kojima je potrebna dobra organizacija radnika, kreiranje optimalnog rasporeda predstavlja problem. Cilj rasporeda u ovome radu jest optimalna organizacija svih resursa potrebnih za održavanje nastave unutar obrazovnih ustanova.

Danas postoje mnoga rješenja kojima se pokušalo olakšati kreiranje kompleksnih rasporeda, no sva rješenja su uglavnom ne pregledna te koji puta i beskorisna radi svoje kompleksnosti korištenja i prikaza podataka. Stoga Timetable Maker (TTM) predstavlja rješenje koje olakšava pristup kreiranju kompleksnih rasporeda na način da prikazuje zauzeća određenih dvorana i predavača s mogućnošću uklanjanja predavanja s rasporeda koja imaju manji prioritet. Cilj aplikacije je svakom sljedećom iteracijom zadovoljiti što više ograničenja kao što su primjerice specifična ograničenja dostupnosti profesora iz privatnih razloga, zatim mogućnost rezerviranja dvorana u slučaju da se niti jedno predavanje u željenom vremenskom periodu ne održava, zatim mogućnost uređivanja rasporeda od strane profesora tokom trajanja semestra uz prethodnu konzultaciju s drugim profesorima na čija predavanja bi izmjena imala utjecaja te na posljetku osnovni cilj je težiti automatizaciji kako bi se posao kreiranja rasporeda u potpunosti eliminirao.

Ovaj rad napisan je u svrhu dokumentiranja klijentskih komponenti aplikacije TTM, to podrazumijeva uvid u prototip koji prikazuje osnovne funkcionalnosti koje bi TTM aplikacija trebala imati, zatim razradu funkcionalnost gdje su opisani zahtjevi na sustav i osnovni dijagrami iste koji služe kao gruba slika cijele aplikacije te na posljetku upoznavanje s programskim rješenjem i implementacijom iste.

2. Raspored sati

Raspored sati osnovni je alat namijenjen za upravljanje vremenom. Sastoji se od tablice vremenskog perioda u kojemu se namjeravaju izvršiti određeni zadaci, događaji, radnje ili pak slijed događaja kronološkog redoslijeda. Raspored je koristan bez obzira jeli namijenjen za kratki, kao primjerice dnevni i tjedni, ili dugi period od nekoliko mjeseci ili čak godina. Raspored se često kreira na osnovu kalendara gdje osoba koja sastavlja osobni ili službeni raspored može zabilježiti datume i vrijeme određenih događaja. Postoji nekoliko vrsta rasporeda :

- Javno dostupni rasporedi
    - Radno vrijeme npr. turističke agencije, državne službe i sl.
    - Raspored vožnje autobusa, aviona, vlaka i sl.
    - Raspored koncerta ili sportskih događanja

- Unutarnji rasporedi
    - Plan upravljanja projektom
    - Upravljanje softverskim projektom
- Rasporedi u obrazovanju

3. Motivacija za izradu aplikacije

Kreiranje rasporeda sati zahtjevan je i dugotrajan proces. Kao što je već spomenuto u prethodnom tekstu raspored može biti različite namjene kao i različitog vremenskog perioda djelovanja. Motivacija za izradu aplikacije rasporeda sati u ovome radu potaknuta je problemima koji se javljaju prilikom kreiranja rasporeda unutar obrazovnih ustanova. Svaka obrazovna ustanova prije početka akademske godine mora imati raspored održavanja nastave što zahtjeva dosta vremena i truda. Puno elemenata je potrebno uzeti u obzir prilikom izrade rasporeda. Elementi kao što su studenti, profesori, dvorane, trajanje predavanja, lokacija, kapacitet dvorana i sl. Neki od problema koji se javljaju prilikom kreiranja rasporeda:

- Podjela profesora prema fizičkoj mogućnosti (jedan profesor ne može predavati dva predmeta u isto vrijeme)

- Podjela semestra prema fizičkoj mogućnosti (studenti prvog semestra ne mogu u isto vrijeme biti na dva različita predavanja)
- Podjela dvorana po predmetu (u jednoj dvorani održava se predavanje jednog predmeta u jedinici vremena)
- Podjela dvorana prema kapacitetu (u dvorani kapaciteta 20 osoba ne može se održati predavanje na kojemu treba biti prisutno 80 studenata)

Dakle mnogo faktora je potrebno uzeti u obzir prilikom izrade rasporeda, dok su ovdje nabrojani samo neki od. Svaka ustanova može imati i svoja unutarnja ograničenja što dodatno otežava posao, primjerice mogućnost održavanja nastave na dvije fizičke lokacije i sl. Cilj ove aplikacije je olakšati posao osobama koje su zadužene za kreiranje rasporeda, u smislu obavještavanja ukoliko dođe do narušavanja nekog od pravila ili preklapanja profesora ili dvorana.

4. Istraživanje tržišta

Danas na tržištu postoji nekoliko aplikacija čija je zadaća rješavanje problema rasporeda sati. Osim što postoje komercijalna rješenja koja se naplaćuju na mjesečnoj bazi postoje i unutarnje aplikacije koje su namijenjene za točno određenu ustanovu te u većini slučajeva takve aplikacije izrađuju zaposlenici ili studenti. Primjer takvih aplikacija je:

„Raspored sati“ – aplikacija nastala od strane računskog Sveučilišta Srce u Zagrebu. Aplikacija raspored sati pruža mogućnost odabira akademske godine i semestra za koji se raspored izrađuje. Odabir tjedne nastave za koje se želi izraditi raspored kao i grupe.

„ascTimetables“ – komercijalna aplikacija, pruža funkcionalnosti kao što su kreiranje vlastitih bilješki, automatiziran odabir optimalnog plana, ispisivanje rasporeda prema profesoru, razredu, učionici. Mogućnost ručnog i automatskog kreiranja rasporeda, opciju regionalnih specifičnosti i sl.

„Wise Timetable“ – komercijalna aplikacija s mogućnošću unosa profesora, dvorana, predmetima zatim generiranja rasporeda s obzirom na unesene podatke. Pruža mogućnost korištenja vlastitoga već izrađenog rasporeda uz dodavanje izmjena na

cijelom rasporedu ili zaključavanje određenog dijela rasporeda te automatsko generiranje ostatka. Osim navedenoga posjeduje mogućnost rezervacije dvorana i ispita kao i prilagodbu na različite jezike i sl.

Sve navedene aplikacije u suštini dijele iste funkcionalnosti, neke posjeduju dodatne opcije poput prilagodbe na jezike ili recimo kao „Wise Timetable“ generiranje samo dijela rasporeda i slično. Sve te aplikacije ujedno dijele i zastarjelo grafičko sučelje koje danas nije dobrodošlo, što je ujedno i primarni problem koji aplikacija u ovome radu nastoji riješiti. Danas je jako bitno da su aplikacije kao i web stranice „*User friendly*“.

5. Razrada funkcionalnosti klijentskih komponenti

Kako bismo lakše vizualizirali interakciju korisnika i sustava koristimo razne dijagrame na strani klijentskih, poslužiteljskih komponenti ali i za prikaz logičke sheme baze podataka. U praksi klasni dijagram (engl. *Class diagram*) koristi se za opis strukture sustava prikazom klase s atributima unutar sustava te odnosa između klasa. Sekvencijalni dijagram (engl. *Use case sequence diagram*) prikazuje objekte i klase uključene u scenarij te slijed poruka razmijenjenih potrebnih za izvršavanje funkcionalnosti scenarija. ER dijagram prikazuje odnose između tablica kako to vidimo u bazi. Dijagram slučajeva korištenja (engl*. Use case diagram*) koristimo pri izradi klijentskih komponenti sustava kako bismo prikazali ponašanje sustava u interakciji s korisnikom. Dijagram slučajeva korištenja služi isključivo za opis pogleda na ponašanje sustava od strane korisnika što znači da ne prikazuje funkcionalnosti unutar sustava, za to se koriste druge vrste dijagrama. Prije svega potrebno je specificirati zahtjeve na sustav, zatim izrada dijagrama te na posljetku i prototip.

5.1. Uloga klijenta i poslužitelja

Klijent je zadužen za vizualnu organizaciju podataka dohvaćenih od strane servera, time TTM funkcionalnosti na strani klijenta čine sljedeće (Šturlan, 2019) :

- Isporuka korisničkog sučelja
- Provjera sintakse kod unosa podataka
- Slanje formatiranog upita poslužitelju
- Primanje i obrada odgovora
- Prikaz obrađenih podataka korisniku

Dok poslužitelja odlikuje sljedeće (Šturlan, 2019):

- Prihvaćanje zahtjeva klijenta
- Upravljanje pristupom
- Obrada zahtjeva
- Dohvaćanje podataka iz baze
- Obrada podataka
- Slanje formatiranih podataka klijentu

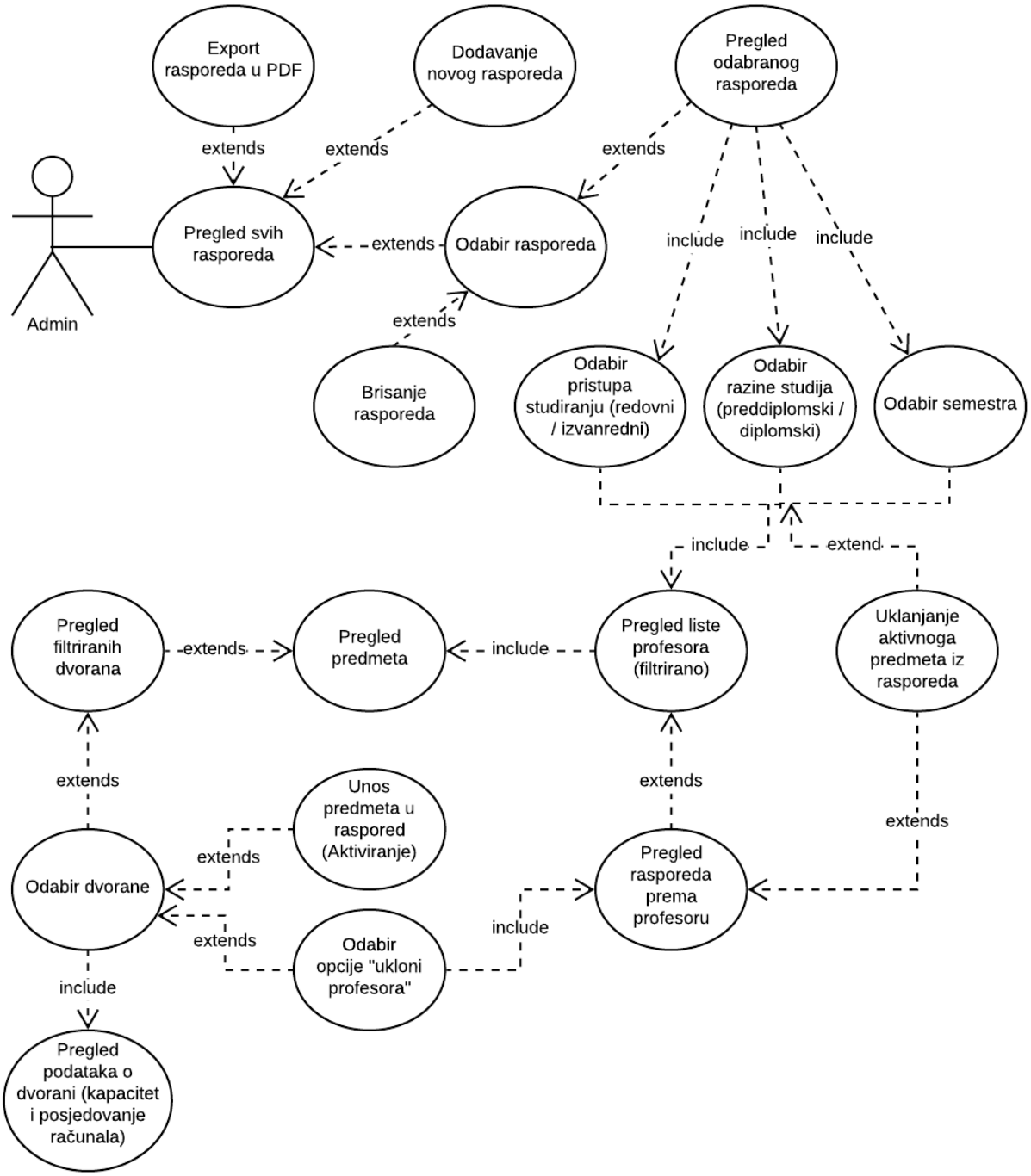
5.2. Zahtjevi na sustav

Osim zahtjeva spomenutih u motivaciji za izradu aplikacije, postoje neki zahtjevi koji su nužni prilikom izrade svake aplikacije koja je namijenjena jednom ili više korisnika, te zahtjevi nužni za nesmetan rad.

- Prijava u aplikaciju
- Pregled postojećih rasporeda
- Dodavanje novog rasporeda
- Brisanje postojećih rasporeda
- Uređivanje odabranog rasporeda
- Preuzimanje rasporeda u pdf obliku
- Obavijesti o ograničenjima

5.3. Dijagram slučajeva korištenja

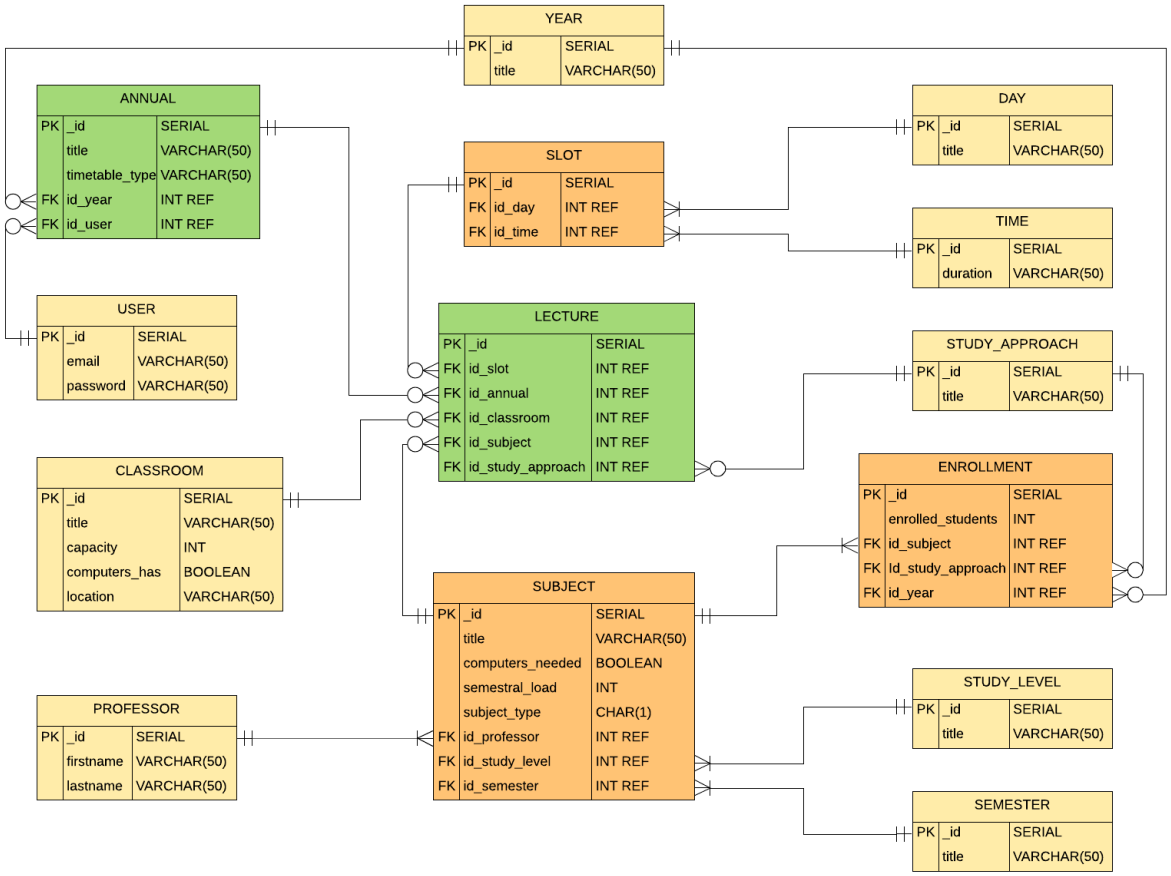
Dijagram slučajeva korištenja sa slike 1 prikazuje interakciju između korisnika i klijentskih komponenti sustava za kreiranje rasporeda sati. Dijagram detaljno opisuje sve korisnikove mogućnosti prilikom korištenja aplikacije. Prilikom pristupa aplikaciji prvo je potrebno prijaviti se u sustav s podacima poput korisničkog imena i lozinke. Nakon prijave administrator može dodati novi ili obrisati postojeći raspored koji se odnosi na određenu akademsku godinu. Osim dodavanja i brisanja raspored je moguće i izvući u pdf, te na posljetku ukoliko želimo uređivati raspored potrebno je odabrati jedan od istih ako oni postoje. Nakon odabira ili kreiranja rasporeda otvara se mogućnost uređivanja istog na način da je potrebno odabrati razinu studija (diplomski/preddiplomski), pristup studiju (redovni/izvanredni) te semestar za koji želimo urediti raspored, ukoliko korisnik ne odabere ništa od nabrojanoga, aplikacija koristi svoje zadane postavke nakon čega nam se filtriraju profesori s svojim predmetima koje je potrebno unijeti u raspored. Svaki predmet posjeduje i popis dvorana u kojima je moguće izvršiti predavanje s obzirom na kapacitet dvorane. Nakon odabira dvorane određenom predmetu otvara se mogućnost ubacivanja predmeta u raspored nakon čega aplikacija javlja ukoliko je dvorana nedostupna ili profesor zauzet u određeno. Ukoliko je profesor zauzet, slot će se prikazati crvenom bojom s mogućnošću uklanjanja predavanja odabranome profesoru, ako je dvirana zauzeta radi drugih predavanja slot će biti nedostupan bez mogućnosti brisanja i mijenjanja, no ukoliko je slot zauzet radi odabrane dvorane i profesora, otvara se mogućnost brisanja predavanja sa slota iz razloga što nam se pod tim uvjetom oslobađaju sva ograničenja koja sprječavaju održavanje izabranoga predmeta.



Slika 1.Dijagram slučajeva korištenja za TTM (Use Case diagram)

5.4. ER dijagram

ER dijagram opisuje veze između tablica na način na koji su spremljene u bazi, kako bismo jednostavnije vizualizirali način na koji planiramo upravljati podacima. ER dijagram se pokazao izričito koristan u praksi, osobito prilikom planiranja većih projekata, te olakšava danji rast aplikacije, time i baze podataka.



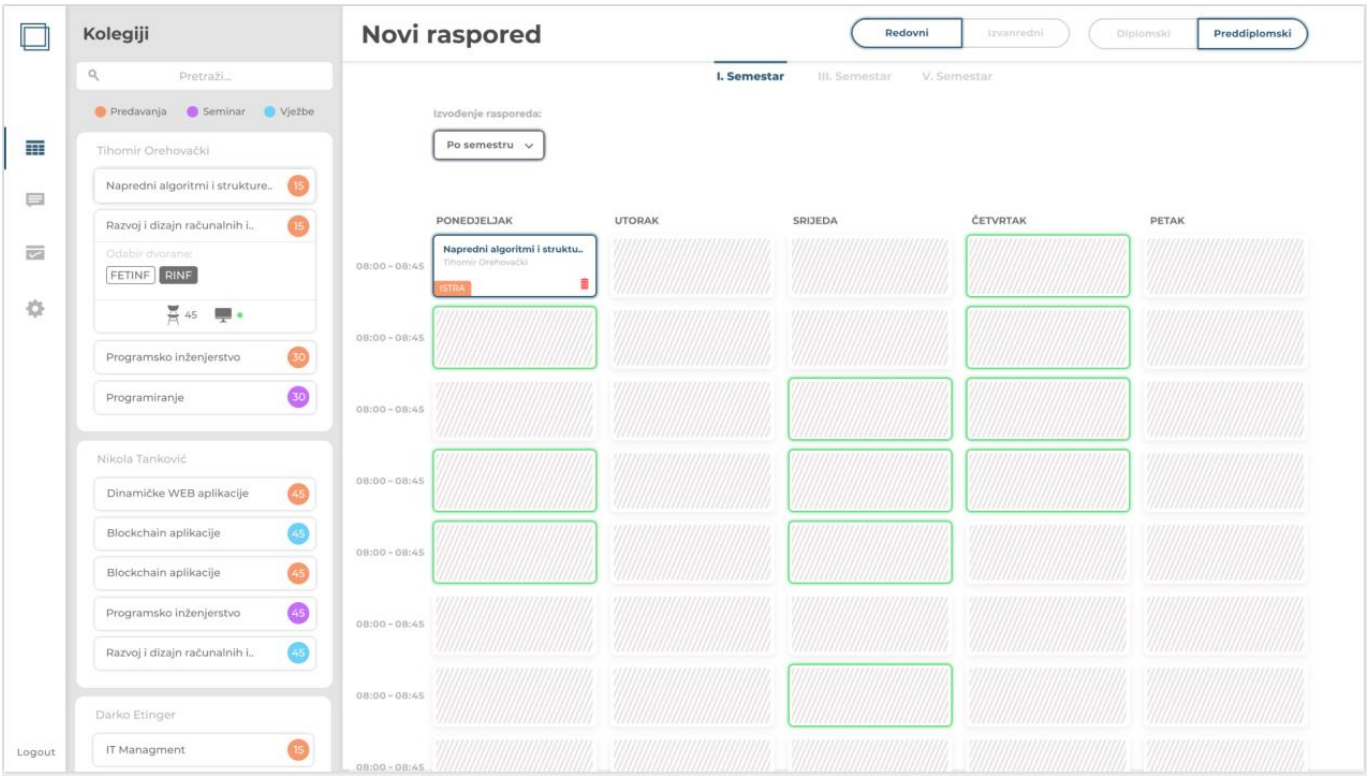
Slika 2.ER dijagram TTM aplikacije (Šturlan, 2019)

„ER dijagram sastoji se od 14 međusobno povezanih relacija od kojih svaka kao primarni ključ sadrži jedinstveni id te ostale atribute koji odgovaraju fizičkoj implementaciji u bazi. Primarni ključ svake relacije u bazi se generira sekvencijalno pa 12 je stoga označen sa PostgreSQL tipom podatka SERIAL. Također su i ostali tipovi podataka usklađeni sa Postgres specifikacijom. Ovaj model prikazuje i strane ključeve relacija te veze i kardinalnosti između relacija Martinovom notacijom. Iz dijagrama se mogu izdvojiti relacije označene zelenom bojom (Lecture i Annual) jer se CRUD operacije na bazu izvršavaju isključivo nad njima. Te relacije, građene po uzoru na

model pahuljice, sadrže strane ključeve okolnih relacija(dimenzija), pa se tako primjerice relacija Lecture sastoji od stranih ključeva svih relacija koje sadrže detalje o predavanjima, a s obzirom da je korisnik taj koji kreira polja s predavanjima uređivanjem rasporeda, redaka u toj tablici može biti relativno mnogo(ovisno o broju godišnjih rasporeda i svih rasporeda unutar pojedinog godišnjeg rasporeda). Iz tog razloga, relacija Lecture kreirana je u normaliziranom obliku sa minimalnom redundancijom podataka. S druge strane, žuta boja relacije označava relacije koje ne sadrže strane ključeve već isključivo podatke koji se koriste u ostalim relacija putem stranih ključeva poput narančasto označenih relacija(na kojima se, u suprotnosti zeleno obojanim relacijama, ne izvršava operacija INSERT). Svaka relacija na ovom dijagramu na strani poslužitelja reprezentirana je Sequelize modelom. Naredna poglavlja opisuju tehnologije korištene na poslužitelju te na stvarnim primjerima detaljno prikazuju povezanost baze podataka i poslužitelja.“(Šturlan, 2019).

5.5. Prototip sučelja

Prototip sučelja prikazuje samo glavni prozor namijenjen za prikaz i uređivanje određenog rasporeda što možemo vidjeti na slici 3.



Slika 3. Prototip glavnog prozora sučelja

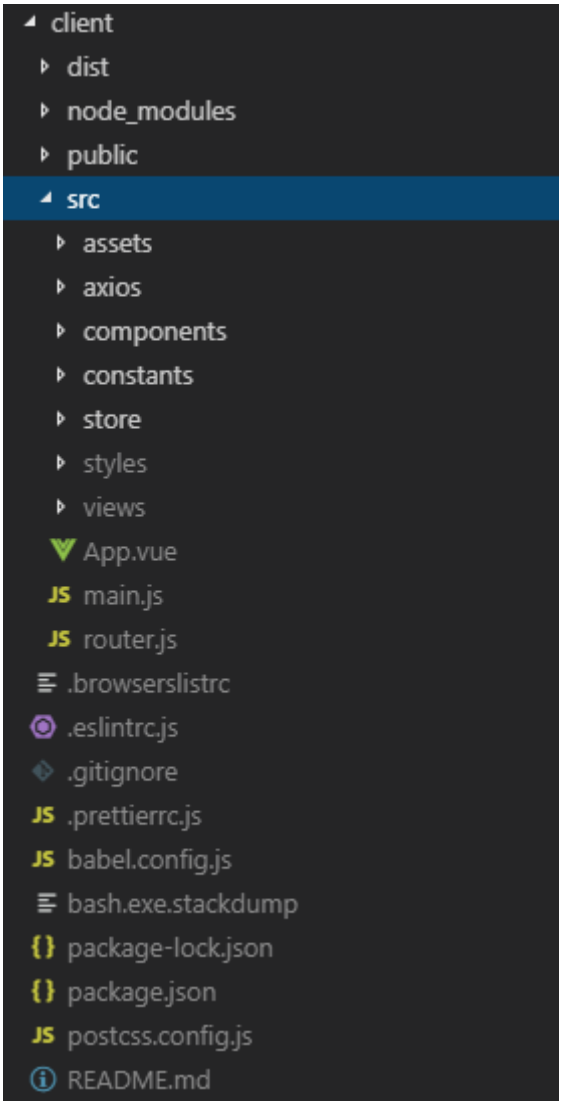
Slika 3 prikazuje način na koji se korisnika obavještava koji termini su slobodni za traženi predmet, dakle zelena polja znače da se tada u toj dvorani ništa ne održava, plavo polje nam prikazuje samo da postoji predmet koji je već u rasporedu, dok siva polja označavaju zauzeća na koja se ne može utjecati. Lijevo u listi profesora, nakon klika na određeni predmet prikazuju se dostupne dvorane s obzirom na kapacitet, klikom na dvoranu prikazuju se detalji iste.

6. Programsko rješenje i implementacija

TTM aplikacija izgrađena je u Vue.js okruženju na strani klijenta, NODE.JS okruženju na strani servera te koristi bazu PostreSQL. Osim navedenih tehnologija koje su nužne za bilo kakav rad aplikacije, koristi se i Docker koji omogućava nesmetan rad aplikacija na bilo kojem računalu, bez obzira dali su specifični paketi, poput primjerice PostgreSQL 5.7 baza podataka, instalirani ili ne. Ovaj rad temelji se na izgradnji klijentskih komponenti aplikacije stoga je ostatak rada strogo orijentiran na Vue.js te način na koji upravljamo podacima kako bi jedan ovakav sustav mogao nesmetano raditi. Vue.js je JavaScript aplikacijski okvir nastao po uzoru na Angular.js i React.js u svrhu lakšeg kodiranja klijentskih komponenti aplikacije. Detaljan opis načina rada Vue.js-a, izgled kostura komponente, Vuex i sl. moguće je pronaći na službenim stranicama https://vuejs.org/ , kao i instalaciju istog.

6.1. Arhitektura aplikacije

Kako bismo kreirali sofisticiranu aplikaciju, organizacija samog projekta vrlo je bitna, pogotovo prilikom izgradnje aplikacije velikog opsega gdje znamo da će aplikacija još rasti. Na slici 4. možemo vidjeti arhitekturu klijentskih komponenti aplikacije TTM.



Slika 4. Arhitektura klijentskih komponenti TTM aplikacije

Osnovna mapa unutar koje se odvija sav sadržaj aplikacije nalazi se u „src“, dok ostali dokumenti i mape služe za određene ovisnosti potrebne za nesmetan rad u Vue.js okruženju. U „src“ mapi prvi file koji vidimo je „assets“ namijenjen za sve potrebne slike, video zapise, mp3 zapise i sl.“Axsios“ sadrži dokument koji služi kao konekcija na backend. „Components“ mapa sadrži sve sitne elemente koji u određenoj kombinaciji čine jednu stranicu u aplikaciji. Mapa „constants“ sastoji se od dokumenta koji sadrži sve konstante pomoću kojih se izvršavaju upiti na backend, ukratko možemo reći da taj dokument sadrži api-e. „Store“ ima ulogu centralnog stanja aplikacije. Naime kako bismo izbjegli slanje podataka s komponente na komponentu, kreirali smo jedan centralni store koji sadrži točno, trenutno stanje aplikacije i svaka komponenta može jednostavno pristupiti svim potrebnim podacima, te na taj način i manipuliramo podacima kako bi se oni reflektirali na sve ostale komponente. Osim što

je Vue.js konstruiran na način da svaka komponenta ima svoj neovisni css ili scss te na taj način različite komponente mogu koristiti iste nazive klasa, svejedno postoje stilovi koji su globalni i nema potrebe za gomilanjem koda u svim komponentama stoga se javila potreba za mapom pod nazivom „styles“ namijenjena isključivo stilovima koji se protežu kroz cijelu aplikaciju kao na primjer stilovi za buttone, naslove, animacije, varijable boja i sl. Posljednja mapa je „views“ koja sadrži sve prozore u aplikaciji, a svaki taj prozor se sastoji od određenih komponenti. „App.vue“ dokument je dokument čija inicijalna struktura sadrži sve ostale komponente odnosno view-ove.

„Main.js“ je dokument koji zapravo pokreće cijelu aplikaciju time što svim važnim dokumentima čini zajedničku točku gdje se spaja „App.vue“, „store“ i „router.js“. Na posljetku imamo „router.js“ , prikazan na slici 5, koji služi za kretanje po aplikaciji pomoću specifičnih ruta, gdje je svaka ruta vezana za određeni view.

```
Vue.use(Router)

const router = new Router({
  mode: "history",
  base: process.env.BASE_URL,
  routes: [
    {
      path: "/login",
      name: "login",
      component: Login
    },
    {
      path: "/overview",
      name: "overview",
      component: Overview
    },
    {
      path: "/dashboard",
      name: "dashboard",
      component: Dashboard
    },
    {
      path: "",
      redirect: "/login"
    },
    {
      path: "**",
      redirect: "/"
    }
  ]
})
```

Slika 5. Router.js u TTM aplikaciji

6.2. Korisničko sučelje i funkcionalnosti

Nakon upoznavanja s rasporedom klijentskih komponenti i ruouter-om koji omogućuje pristup svakom od view-ova, prelazimo na analizu komponenti, kako one izgledaju korisniku te način na koji su kreirane.

6.2.1. Prozor za prijavu u TTM

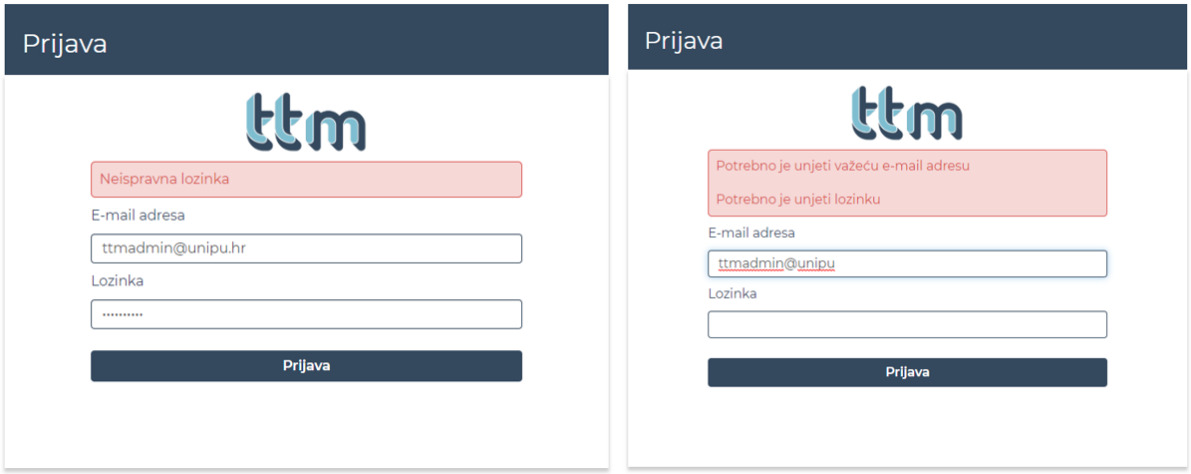
Prilikom korištenja aplikacije korisnik prvo mora proći prijavu u sustav. Kako bismo izbjegli ne ovlašteni pristup aplikaciji, prijava u sustav je moguća samo putem administracijskog email-a i lozinke. Ne ovlašten pristup na strani klijenta znači da korisnik ne smije imati mogućnost korištenja bilo koje rute dok uspješno ne prođe autentifikaciju na strani servera. Slika 6 prikazuje nam jedan od načina za implementaciju takvoga oblika autentifikacije.

```
export default {
    data(){
        return{
            User:{
                email:'',
                password:''
            },
            errors: []
        }
    },
    methods:{
        login(){
            axios.post(api.API_USER,this.User)
            .then(res =>{
                this.errors = []
                this.$store.dispatch('LOGIN', {token:res.data.token, id:res.data.user._id})
                this.$router.push('overview')
            })
            .catch(e =>{
                if(e.response){
                    if(e.response.status === 422 || e.response.status === 403){
                        console.log(e.response.data)
                        this.errors=e.response.data.errors
                        this.showErrors=true;
                    }
                }
            })
        }
    }
}
```

Slika 6. Zaštićene rute TTM aplikacije

Dakle nakon unosa email-a i lozinke te prilikom klika na gumb „Prijava“, klijent prvo mora provjeriti dali navedeni korisnik postoji, a to činimo s naredbom „axios.post“ koja

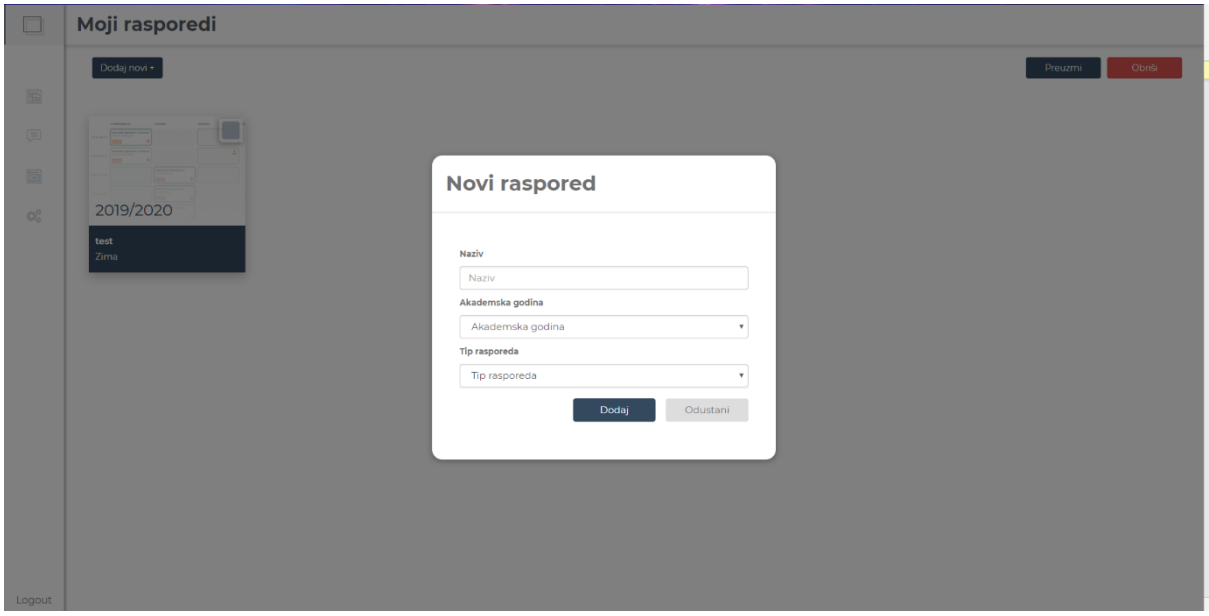
se nalazi u prvom redu funkcije. Nakon što se odgovor vrati sa servera moramo provjeriti jeli korisnik uspješno prijavljen ili smo za odgovor dobili status o grešci te u skladu s odgovorom spremamo korisnikov token ili prikazujemo poruku o grešci kao primjerice na slici 7. Ukoliko je korisnik uspješno prijavljen omogućava mu se pristup ostalim rutama u aplikaciji.



Slika 7. Poruka o grešci prilikom neuspješne prijave

6.2.2. Prozor za upravljanje rasporedima

Nakon uspješne prijave u sustav omogućen nam je pristup rasporedima, dodavanje novog rasporeda, brisanje više rasporeda od jednom te odabir aktivnog rasporeda što možemo vidjeti na slici 8.

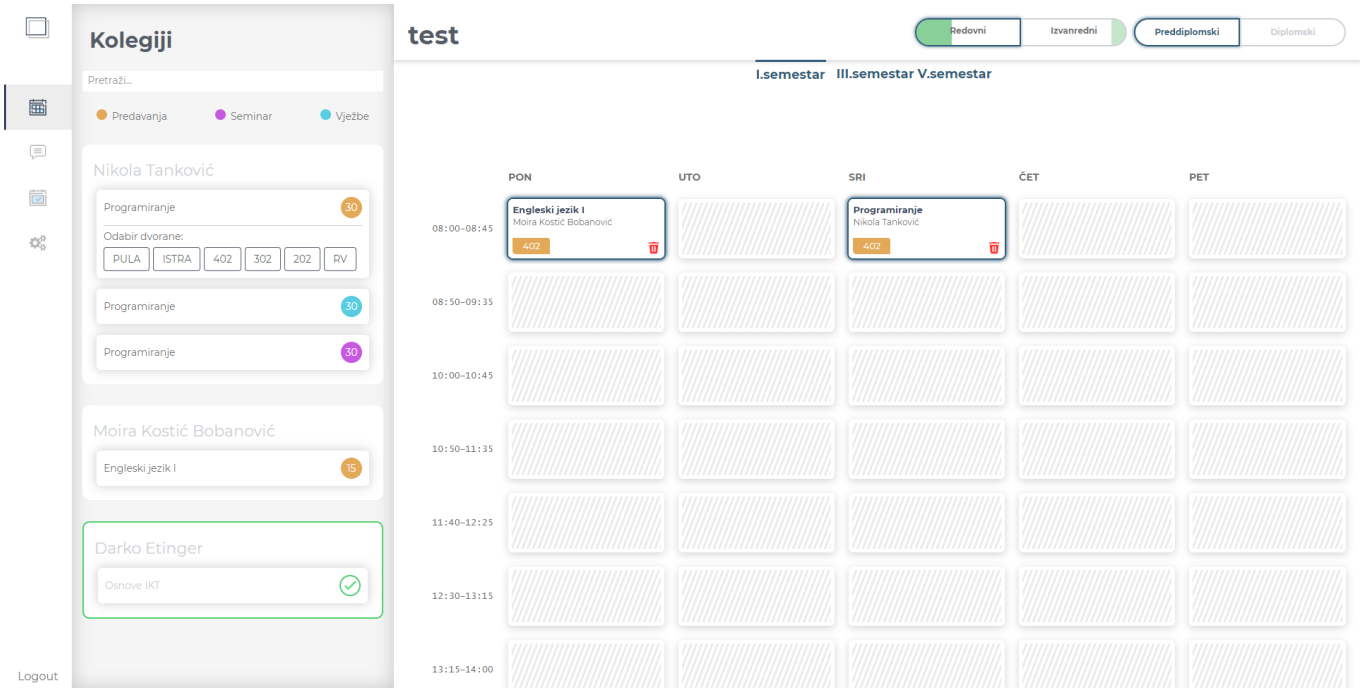


Slika 8. Dodavanje novog rasporeda

Jedan raspored namijenjen je za jedno semestralno razdoblje, a rasporeda može biti više, stoga je potrebno odabrati raspored koji želimo uređivati, te nakon odabira rasporeda imamo mogućnost posjetiti rutu koja nam to i omogućuje.

6.2.3. Prozor za uređivanje rasporeda

Kada je odabrani raspored aktivan pruža nam se mogućnost uređivanja istoga.



Slika 9.Uređivanje aktivnog rasporeda

Slika 9 prikazuje prozor za uređivanje rasporeda koji posjeduje opcije poput smjera studija, pristupa studiju te semestra koji su prema default-u odabrani prema prvom id-u, zatim se s obzirom na selektirane opcije sortiraju profesori zajedno s potrebnim predmetima. Klikom na predmet prikažu nam se sve dvorane koje imaju dovoljan kapacitet za broj studenata na tom predavanju, vidljivo na slici 10. Nakon što smo profesoru pronašli njegove predmete te proslijedili taj predmet komponenti sa slike 10. prvo provjeravamo ima li taj predmet još uvijek predavanja koja treba rasporediti, ako ima otvara se dijalog s mogućim dvoranama, u suprotnome ne događa se ništa. Otvaranjem dijaloga za dvorane potrebno je postaviti neke vrijednosti koje su potrebne za buduće funkcionalnosti u store-u kao što su odabrani profesor, id aktivnog predmeta, promjena aktivne dvorane i ponovno renderiranje rasporeda. Zatim tražimo predmet kojemu odgovara aktivni predmet i aktivni pristup studiju kako bismo provjerili

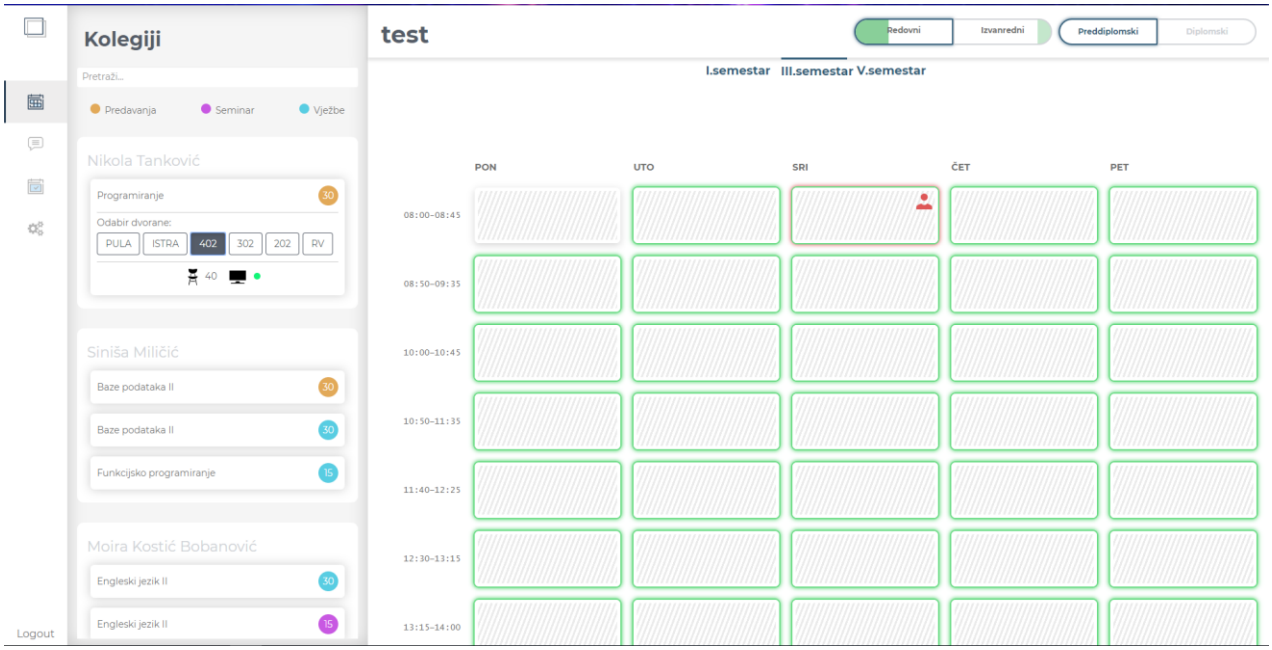
broj studenata na predavanju te time filtrirali dvorane u kojima je moguće izvršiti predavanje.

```
<template>
  <div class="subjectDash">
    <div @click="subject.semestral_load === 0
                 ? null
                 : openDialog(subject._id)" class="subjectDash-header" :class="[subject.semestral_load === 0 ? 'done' : null]">
      {{ subject.title }}
      <span v-if="subject.subject_type === 'P'" :class="[subject.semestral_load === 0 ? 'dot' : 'dot dot-P']">{{
        subject.semestral_load === 0 ? '' : subject.semestral_load
      }}</span>
      <span v-if="subject.subject_type === 'V'" :class="[subject.semestral_load === 0 ? 'dot' : 'dot dot-V']">{{
        subject.semestral_load === 0 ? '' : subject.semestral_load
      }}</span>
      <span v-if="subject.subject_type === 'S'" :class="[subject.semestral_load === 0 ? 'dot' : 'dot dot-S']">{{
        subject.semestral_load === 0 ? '' : subject.semestral_load
      }}</span>
    </div>
    <ClassroomDash :dropdownActive="subject.isActive" :subject="subject" />
  </div>
</template>
```

```
methods: {
  openDialog(id) {
    this.$store.dispatch("setSelectedProfessor", this.professor),
    this.$store.dispatch("setActiveSubjectStatus", id),
    this.$store.dispatch("setActiveClassStatus", -1),
    this.$store.dispatch("setRowData"),
      axios
        .get(
          api.API_ENROLLMENT +
            "/" +
            this.subject._id +
            "/" +
            this.ACTIVE_APPROACH
        )
        .then(r => r.data)
        .then(enrollment => {
          if (enrollment[0]) {
            this.$store.dispatch(
              "setNumberOfStudents",
              enrollment[0].enrolled_students
            )
            this.$store.dispatch(
              "getActiveClassrooms",
              enrollment[0].enrolled_students
            )
          } else {
            this.$store.dispatch("setNumberOfStudents", 0)
            this.$store.dispatch("deleteActiveClassrooms")
          }
        })
        .catch(error => {
          console.log(error)
        })
  }
}
```

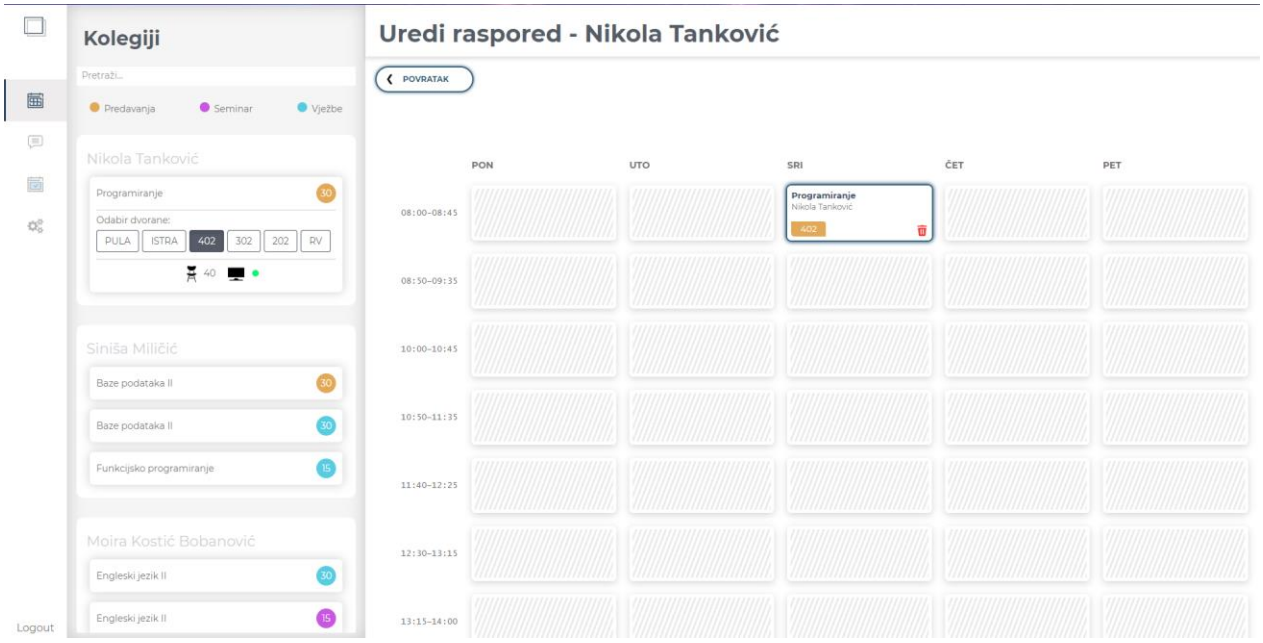
Slika 10.Prikaz predmeta s pripadajućim dvoranama

Osim sortiranih profesora, sadržaj rasporeda se popunjava sa predavanjima koji su već aktivni u rasporedu uz mogućnost uklanjanja predavanja. Klikom na dvoranu, kao što možemo vidjeti na slici 11, prikazuju se detalji o dvorani poput kapaciteta dvorane i posjedovanja računala, te raspored prikazuje dostupnost dvorane i profesora s obzirom na odabranu dvoranu željenom predmetu.



Slika 11.Dostupnost dvorane

Zelena polja označavaju slobodnu dvoranu i profesora, siva polja označavaju zauzeće dvorane na koje se ne može utjecati, ukoliko je polje popunjeno podacima ali ima zeleni okvir to znači da je ta dvorana zauzeta radi prikazanog predavanja, a profesor slobodan te ukoliko se predavanje ukloni oslobađa se dvorana, crveno polje s ikonom profesora znači da je dvorana slobodna u traženom terminu ali je profesor zauzet te klikom na crvenu ikonu otvara nam se raspored od profesora te mogućnost uklanjanja predmeta kao što možemo vidjeti na slici 12.



Slika 12.Uređivanje profesorovog rasporeda

Kako bismo ovo postigli, nakon svakog klika na dvoranu, pristup, semestar ili razinu studija potrebno je ponovno generirati raspored kako bismo vidjeli aktivno sranje odabranih polja, a to činimo metodom „setRowData()“. Metoda „setRowData“ prvo kreira objekt „data“ koji predstavlja jedan red u rasporedu. Objekt se sastoji od id-a vremena i od liste slotova, dok se slot sastoji od podataka kao što su dostupnost dvorane, dostupnost profesora, jeli slot popunjen te ukoliko je koristimo „filledData“ objekt koji sadrži informacije o profesoru i predmetu. Nakon kreiranja objekta „data“ dohvaćamo sva vremena kako bismo ih podijelili na sve „data“ objekte. Zatim svakom redu u rasporedu prolazimo kroz sve slotove gdje je id vremena u našem data objektu, te time imamo sve slotove pokrivene. Zatim je potrebno pronaći sva predavanja koja se već događaju u svakom od slotova, nakon što pronađemo sva predavanja za slot provjeravamo jeli odabrana dvorana ili prikazujemo samo raspored, ako je samo raspored prikazuju se predmeti u plavome okviru za taj semestar, a ukoliko je odabrana dvorana prvo tražimo predmete koji se poklapaju sa selektiranim vrijednostima, zatim provjeravamo pripada si označenoj dvorani i jeli profesor zauzet u tom slotu, ako se sve poklapa prikazuje se predavanje u zelenom okviru što znači da je profesor slobodan i uoliko se ukloni predavanje oslobađa se i dvorana. Ako predmet nije u označenoj dvorani prikazujemo ga u plavome okviru što ne znači nužno da će se osloboditi dvorana ili profesor ako se ukloni predavanje, uz mogućnost brisanja. Nakon što smo provjerili sve lekcije koje su aktivne prema označenim

vrijednostima koje su prioritet, provjeravamo ostale slotove kako bismo definirali zauzeće slota. Prolazimo kroz sva predavanja koja se nalaze u slotu ali nisu aktivna, ukoliko pronađemo predavanje kojemu se id dvorane poklapa sa selektiranom dvoranom, postavljamo taj slot kao zauzet, zatim provjeravamo dali to isto predavanje posjeduje i istog profesora, te ukoliko je to istina popunjavamo slot na način da je profesoru moguće ukloniti predavanje te time potraga u tom slotu staje, ukoliko nismo pronašli navedeno predavanje provjeravamo jeli slobodna dvorana i profesor za isto predavanje, ako je postavljamo slot kao dostupan, a ako nije dalje provjeravamo jeli profesor zauzet u nekom od predavanja te prikazujemo slot s mogućnošću uklanjanja profesora. Osim navedenoga, za svaki slot je potrebno brojati posjeduje li barem jednu lekciju, te ako ne posjeduje postavljamo taj slot dostupnim.

Nakon što aplikacija prikaže trenutno stanje raspoloživih dvorana, kao i profesora, klikom na zeleno polje predmet pod odabranom dvoranom sprema se u raspored te broj potrebnih predavanja iz lijevog izbornika se smanjuje za onoliko sati koliko ima tjedana odabrani semestar, primjer koda odgovornog za ispravan rad ovakve funkcionalnosti moguće je vidjeti na slici 13.

```
async addSelectedLecture(slotId) {
  const subject = this.ACTIVE_SUBJECTS.find(element => {return element.isActive === true})
  const classroom = this.SELECTED_CLASSROOM._id;
  const annual = this.GET_ACTIVE_TIMETABLE_ID;
  const approach = this.ACTIVE_APPROACH;
  const lecture = {
    id_subject: subject._id,
    id_classroom: classroom,
    id_annual: annual,
    id_study_approach: approach,
    id_slot: slotId
  }
  await this.$store.dispatch("updateNumberOfLectures", [subject._id, 'remove']);
  await this.$store.dispatch("addLecture",lecture);
```

Slika 13. Dodavanje predmeta u raspored

Dakle pronalazimo predmet koji je aktivan, dvoranu, aktivni raspored i pristup studiju dok id slota prosljeđujemo klikom na isti, te time kreiramo objekt koji se sprema u bazu. Nakon što je predmet uspješno kreiran unutar slota potrebno je ažurirati preostali broj lekcija za označeni predmet što činimo funkcijom „updateNumberOfLectures“. Istoj je potrebno proslijediti id premeta kao i naziv akcije koju želimo izvršiti. Ukoliko dodajemo predmet u raspored potrebno je smanjiti preostali broj potrebnih sati predavanja, dok ukoliko brišemo predmet iz rasporeda moramo uvećati broj sati, odnosno vratiti prethodno stanje predavanja.

```
UPDATE_SUBJECT(state, [subject, action]) {
  state.activeSubjects.forEach(element => {
    if (element._id === subject && element.semestral_load !== 0 && action === 'remove') {
      element.semestral_load -= 15
    }
    else if (element._id === subject && action === 'revert') {
      element.semestral_load += 15
    }
  });
},
```

Slika 14. Ažuriranje broja sati preostalih predavanja

Slika 14 prikazuje funkciju koja prvo prolazi kroz sve predmete koji su aktivni s obzirom na odabrane opcije zatim s obzirom na proslijeđenu akciju dodaje ili uklanja određeni broj predavanja s obzirom na broj tjedana u semestru.

Brisanje podataka iz rasporeda vrši se na način da klikom na ikonu za brisanje slota proslijedimo id lekcije i id predmeta, te na taj način uklanjamo lekciju iz rasporeda, vidljivo na slici 15.

```
async removeLecture(lecture_id, subject_id) {
  await this.$store.dispatch("deleteLecture", lecture_id);
  await this.$store.dispatch("updateNumberOfLectures", [subject_id, 'revert'])
  await this.$store.dispatch("setRowData");
  if(this.ACTIVE_APPROACH === 1) {
      await this.$store.dispatch("updateRegularLeft", 15)
      this.$store.dispatch("setProgresBar")
  } else {
    await this.$store.dispatch("updatePartTimeLeft", 15)
    this.$store.dispatch("setProgresBar")
  }
},
```

Slika 15.Uklanjanje predmeta iz rasporeda

Dakle nakon što znamo id lekcije i premeta, funkcijom „deleteLecture“ uklanjamo lekciju, zatim ažuriramo preostali broj sati te ponovno postavljamo raspored s ažuriranim podacima. Osim broj sati potrebno je izračunati i postotak širine progres bara a to vršimo na način da zbrojimo sve potrebne lekcije za redovne i izvanredne studente, zatim zbrajamo koliko je lekcija još preostalo, odnosno koliko predavanja je još potrebno unijeti u raspored. Nakon što znamo te dvije informacije potrebno je 100 podijeliti s ukupnim brojem predavanja zatim preostali broj predavanja množimo s tim brojem te oduzimamo od 100 kako bismo dobili postotak riješenih predmeta za oba pristupa studiju, primjer koda prikazan na slici 16.

```
SET_PROGRESS_BAR(state, [regularLoad, partTimeLoad, regularLeft, partTimeLeft]) {
  let reg = 100 / regularLoad;
  let part = 100 / partTimeLoad;
  state.regularProgress = 100 - (reg * regularLeft);
  state.partTimeProgress = 100 - (part * partTimeLeft);
}
```

Slika 16. Funkcija za postavljanje postotka unesenih predmeta u raspored

7. Zaključak

Aplikacija „TTM“ pruža mogućnost administratorima pregledan pristup svakoj vrsti studija te uvelike olakšava generiranje rasporeda na način da nije potrebno paziti hoće li se predmeti, profesori, dvorane ili semestri preklapati u terminima iz razloga što aplikacija to čini za njih obavještavajući korisnika da određeni entitet ne može biti izvršen u određeno vrijeme radi ograničenja. Aplikacija na ta način olakšava zahtjevan i dugotrajan proces sklon greškama. Tehnologije korištene prilikom izgradnje klijentskih komponenti „TTM“ aplikacije su HTML, SCSS i JavaScript u okruženju JavaScript aplikacijskog okvira Vue.js koji koristi sve tri tehnologije u gotovo savršenoj kombinaciji. Osim tehnologija za izgradnju korisničkog sučelja koristili smo i tehnologije poput Git-a koji olakšava upravljanje verzijama koda te Docker koji rješava problem instalacija raznih tehnologija kako bi aplikaciju uopće bilo moguće pokrenuti.

Aplikacija je za početak namijenjena samo olakšavanju kreiranja rasporeda, dok buduća poboljšanja dolaze u kasnijim verzijama. Spomenuta poboljšanja odnose se na funkcionalnost poput razmjene poruka, mogućnost kreiranja više verzija jednog segmenta rasporeda, primjerice rasporeda za redovne studente prvoga semestra, kako bismo kasnije mogli kombinirati segmente te na taj način od više rješenja pronaći ono optimalno za profesore i studente. Isto tako jedno od poboljšanja bi bilo uvođenje osobnih ograničenja po profesoru ili čak mogućnost ažuriranja rasporeda od strane profesora ili ček studenata. Krajnje poboljšanje bi bila sama automatizacija kreiranja rasporeda, to bi bilo izvedivo na način da algoritam sam uzme u obzir sve inpute te na taj način kreira više vrsta rasporeda te predloži optimalni raspored.

Aplikacija je podijeljena na tri repozitorija. Prvi repozitorij sastoji se od preostala dva za klijenta i server te od docker-compose datoteke koja je namijenjena za kreiranje baze podataka. Glavni repozitorij moguće je preuzeti na adresi https://github.com/kvuckov/Raspored, klijentske komponente nalaze se na https://github.com/kvuckov/client, dok serverske komponente je moguće preuzeti na adresi https://github.com/asturlan/server. Korisničke podatke za prijavu u sustav moguće je pronaći u README.md datoteci kao i upute za pokretanje aplikacije.

Literatura

Internet izvori:

1. Wise Timetable, Dostupno na: http://www.wisetimetable.com/PDF/Wise%20Timetable%20cro.pdf [12.07.2019]
2. Applied Software Consultants, 2016. aSc Timetables, Dostupno na: https://www.asctimetables.com/ [2.10.2019]

Završni radovi:

1. Šturlan Adriana (2019) Razvoj poslužiteljskih komponenti aplikacije za upravljanje rasporedom.
2. Doroteo Macan (2016) Aplikacija za upravljanje problema rasporeda

Popis slika

Sažetak

Unatoč činjenici da programsko inženjerstvo ima prilično kratku povijest postojanja, u zadnjih je nekoliko godina sve zastupljenija znanstvena grana. Web stranice su se razvile do te mjere da je danas gotovo ne moguće pronaći razliku između dinamičke web stranice i web aplikacije s obzirom da dijele relativno jednake osobine. Kako su se aplikacije razvijale tako je postajala sve veća zainteresiranost od strane korisnika za integracijom sustava u određene ustanove kako bi se repetativni poslovi mogli izbjeći te teški poslovi mogli olakšati. Softvere danas možemo pronaći svagdje oko nas od perilice za rublje do aplikacije na zaslonu našega ekrana. Ovaj rad usmjeren je izgradnji klijentskih komponenti web aplikacije koja bi olakšala posao administratorima unutar sveučilišnih ustanova prilikom kreiranja rasporeda sati za narednu akademsku godinu. Detaljno su razrađuje funkcionalnosti na strani klijenta kako bi jedna takva aplikacija mogla nesmetano funkcionirati.

Ključne riječi: web aplikacija, klijentske komponente, HTML, CSS, SCSS, JavaScript, Vue.js, Raspored sati, Axios, Docker

Abstract

Despite the fact that programming engineering has a fairly short history of existence, over the last few years it is increasingly represented in science. Websites have evolved to such an extent that today it is virtually impossible to find the difference between a dynamic web site and web application since they share relatively similar characteristics. As the applications developed, the user's have become more and more interested in integration of software into a certain institution in order to avoid repetitive jobs, and alleviate difficult jobs. The software today can be found everywhere around us from the washing machine to the application on the screen of our monitor. This thesis focuses on building frontend of web application that would facilitate business administrators within university institutions while creating schedule for the next academic year. It elaborates on client side functionalities so that one such application can run flawless.

Keywords: web application, frontend, HTML, CSS, SCSS, JavaScript, Vue.js, Schedule, Axios, Docker