Web aplikacija za vođenje demonstrature

Felding, Emilia

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

WEB APLIKACIJA ZA VOĐENJE DEMONSTRATURE

Diplomski rad

Emilia Felding

Osijek, 2018. godina

SADRŽAJ

1. UVOD

Demonstratori su redoviti studenti koji sudjeluju u nastavi kao pomoć nastavnicima oko pripreme i izvedbe vježbi. Svakog semestra se na stranicama fakulteta objavi poziv studentima da se prijave za demonstraturu. Studenti se mogu prijaviti na predmet kojeg su već položili ili na predmet koji će po prvi put slušati u tekućoj akademskoj godini. Za samu prijavu potrebno je ispuniti obrazac koji se sastoji od dva dijela – osnovni podaci te odabir kolegija. Osnovni podaci su ime i prezime studenta, prosjek ocjena te željeni broj sati demonstrature. Odabir kolegija omogućuje studentu da odabere nekoliko kolegija na kojima želi biti demonstrator te da navede ocjene koje ima iz tih kolegija. Studenti ispunjeni obrazac zatim predaju u studentsku referadu i čekaju objavu na stranicama fakulteta s rezultatima poziva i, u slučaju da su primljeni, brojem sati koje će odraditi. Daljnja procedura oko dogovora termina odrade se temelji na dogovoru između nastavnika i studenta, što često dovodi do komplikacija, što zbog nemogućnosti dogovora kada je riječ o više studenata na istom kolegiju, što zbog poklapanja termina demonstrature i termina predavanja, koja studenti demonstratori moraju i dalje pohađati.

U prvom dijelu ovoga rada obradit će se temeljne tehnologije koje će se koristiti pri izradi web aplikacije koja će cijelu proceduru prijave na demonstraturu te dogovora između studenata i nastavnika o izvedbi iste svesti na jedan brži i jednostavniji sustav. U drugom dijelu rada obrađene tehnologije će se primijeniti te tako dovesti do krajnjeg rješenja navedenog problema, čija će se primjena obraditi u trećem dijelu.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je napraviti web aplikaciju koja će imati mogućnost dodavanja i uređivanja kolegija, nastavnika i demonstratora. Profesori će imati mogućnost otvaranja demonstrature, a studenti prijave ako žele biti demonstratori. Profesor zatim prihvaća željeni broj studenata i obavještava ih putem e-maila. Kroz aplikaciju treba biti moguće dodati raspored demonstratura te odrediti koji student je zadužen za njeno provođenje. Također, potrebno je dati statistiku demonstratura po kolegiju i studentu.

2. PREGLED KORIŠTENIH TEHNOLOGIJA

Web aplikacije su programska rješenja kojima klijent može pristupiti koristeći Internet, odnosno neke od Internet preglednika. Ovakve aplikacije se u većini slučajeva sastoje od dva dijela, onog serverskog i onog klijentskog. Klijentski dio se bavi prezentacijom informacija dok se serverski bavi spremanjem i vraćanjem tih informacija korisniku. Najveća korisnička prednost je ta što su dostupne u bilo koje vrijeme i s bilo kojeg mjesta, uključujući i računala i pametne mobilne telefone. Za njihovo korištenje potrebno je samo osnovno znanje korištenja računalom. Prednost izrade web aplikacija za programere je ta što se one izrađuju nevezano na kojem tipu računala ili operacijskog sustava će se koristiti [1].

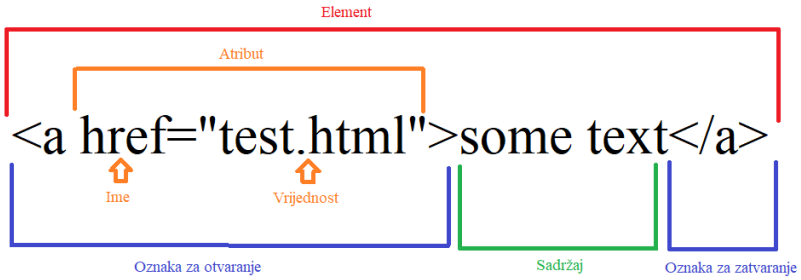
2.1. HTML

HTML je skraćenica od engl. *Hyper Text Markup Language,* jezika nastalog krajem devedesetih prošloga stoljeća, koji se koristi za stvaranje web stranica. Svaka stranica se sastoji od niza veza na druge stranice – hiperlinkova (engl. *hyperlinks*) te je svaka stranica na Internetu napisana jednom od HTML verzija. Prva verzija ovog prezentacijskog jezika izašla je 1993. godine, a od tada su izašle još četiri, uključujući zadnju poznatu kao HTML5.

HTML osigurava pravilno formatiranje teksta i slika tako da preglednici na ispravan način prikažu sadržaj stranice korisniku. HTML daje stranici osnovnu strukturu te se može shvatiti kao kostur (engl. *bones*) stranice dok CSS, koji je objašnjen u sljedećem poglavlju, daje stranici izgled (engl. *skin/appearance*)[2].

HTML se sastoji od oznaka (engl. *tags*), od kojih se svaka može sastojati od nekoliko elemenata (slika 2.1.):

- oznaka za otvaranje (engl. *opening tag*) – sastoji se od znaka manje (″<″), imena oznake te znaka veće (″>″),
- oznaka za zatvaranje (engl. *closing tag*) - sastoji se od znaka manje i kose linije (″/<″), imena oznake te znaka veće (″>″),
- atribut (engl. *attribute*) – koji se sastoji od dva elementa:
  - ime atributa (engl. *attribute name*),
  - vrijednost atributa (engl. *attribute value*),
- sadržaj oznake (engl. *tag content*).



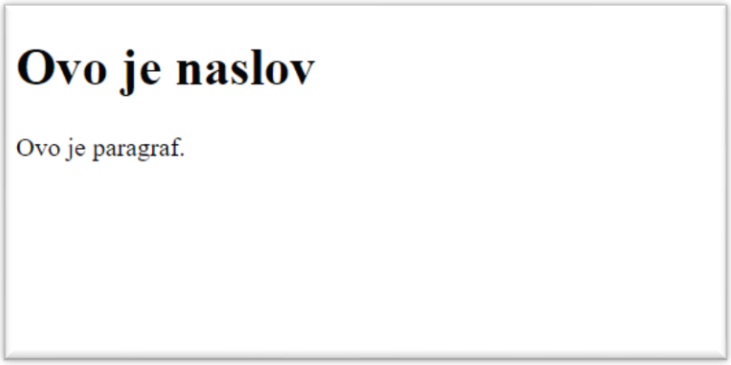
Sl.2.1. *Primjer HTML oznake*

Neke od najčešće korištenih oznaka su:

- za komentare <!-- … -->,
- za hiperlinkove <a>,
- za podebljani tekst <b>,
- za gumbove <button>,
- za sekcije (engl. *divisons*) ,
- za naslove (engl. *headers*) ovisno o veličini, od najvećeg do najmanjeg <h1> - <h6>,
- za slike <img>,
- za paragrafe <p>,
- za tablice <table>.

U nastavku je prikazan jednostavan HTML kod, a na slici 2.2. kako to izgleda u pregledniku.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Primjer HTML koda</title>
</head>
<body>
<h1>Ovo je naslov.</h1>
<p>Ovo je paragraf.</p>
</body>
</html>
```



Sl.2.2. *Prikaz HTML koda u pregledniku*

2.2. CSS

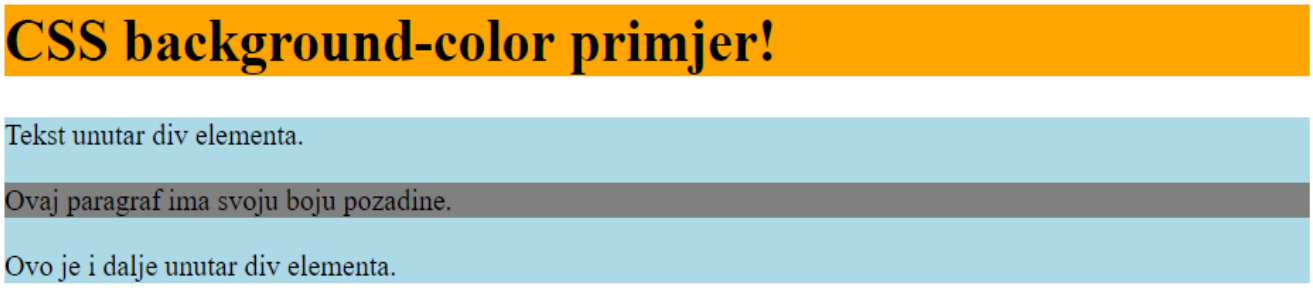
CSS stoji za engl. *Cascading Style Sheets* te predstavlja stilski jezik koji se koristi za opis prezentacije dokumenta napisanog pomoću HTML ili XML jezika. U početku se definicija prezentacije vršila pomoću posebnih elemenata koji su se ubacivali u HTML, no ubrzo je uočena potreba za odvajanjem prezentacije od samog opisa prezentacije te je uveden CSS, kojim se uređuje sam izgled i raspored stranice. CSS se sastoji od tri glavna elementa (sl.2.3.):

- selektor – služi za označavanje dijela *markup*-a na koji se primjenjuje isti stil,
- deklaracijski blok – blok unutar vitičastih zagrada unutar kojih se nalaze deklaracije podijeljene na dva dijela:
  - svojstvo, te
  - oznaka/vrijednost [3].

Sl.2.3. *CSS selektor i deklaracijski blok*

Slika 2.4. prikazuje kako preglednik prikaže dolje navedeni kod za CSS unutar HTML oznake <style>.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
   background-color: orange;
}
div {
   background-color: lightblue;
}
p {
   background-color: grey;
}
</style>
</head>
<body>
<h1>CSS background-color primjer!</h1>
<div>
Tekst unutar div elementa.
<p>Ovaj paragraf ima svoju boju pozadine.</p>
Ovo je i dalje unutar div elementa.
</div>
</body>
</html>
```



Sl.2.4. *Prikaz CSS koda u pregledniku*

2.3. Angular

Angular (odnosi se na Angular verzije 2+) je besplatno okruženje za razvoj web aplikacija razvijeno od strane Google-a 2009. godine s namjenom razvoja jednostraničnih aplikacija (engl. *single-page applications*) koristeći HTML i *TypeScript*. Sam Angular (sl.2.5.) je napisan u *TypeScript*-u te omogućuje implementaciju nekih osnovnih, ali i dodatnih opcionalnih funkcionalnosti u aplikaciju koristeći upravo *TypeScript* biblioteke.

Sl.2.5. *Angular logo*

Osnovni gradivni blokovi ovakve aplikacije su tzv. *NgModules*, koji komponentama (engl. *components*) daju kontekst kompilacije (engl. *compilation context*). Ovi moduli „skupljaju“ povezani kod u smislene cjeline što znači da je Angular aplikacija sastavljena od nekoliko tih modula. Aplikacija uvijek ima korijenski iliti početni modul (engl. *root module*) koji omogućuje njeno početno pokretanje (engl. *bootstrapping*), a uz njega može imati još mnogo dodatnih modula [4].

Komponente definiraju prikaze (engl. *views*), tj. ono što će biti prikazano korisniku, a Angular će ih modificirati prema programskoj logici i podacima. Uz to, komponente koriste servise (engl. *services*), koji aplikaciji pružaju specifične funkcionalnosti koje nisu direktno povezane s korisničkim prikazom. I prikazi i servisi su klase, s dekoratorima koji definiraju njihov tip te metapodatke koji govore Angularu kako ih koristiti.

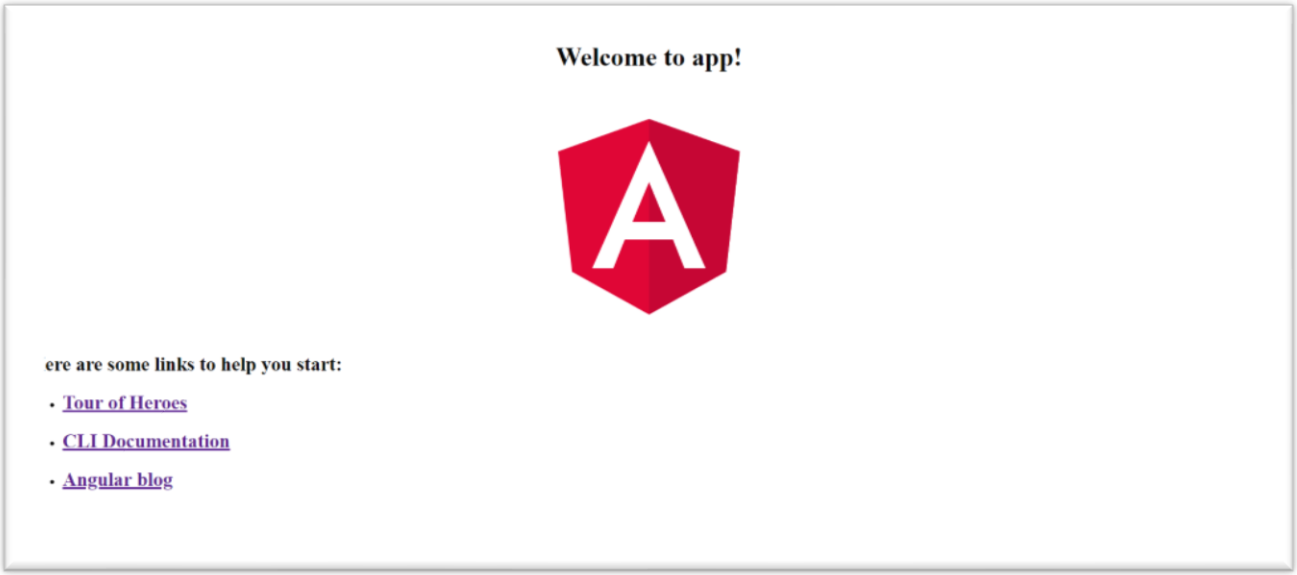
Velika prednost *TypeScript*-a je ta što omogućuje tzv. strogo tipiziranje što se uvelike koristi upravo pri prevođenju jer se vrši provjera nad tipovima podataka kako ne bi došlo do pogreške prilikom samog izvršavanja koda.

Uobičajena je praksa za izgradnju Angular aplikacije koristiti Angular-CLI (engl. *command line interface*). Nakon instalacije *node.js*-a, potrebno je samo nekoliko linija unutar njegovog naredbenog retka da se kreira prva Angular aplikacija:

- npm install -g @angular/cli
- ng new ime-aplikacije

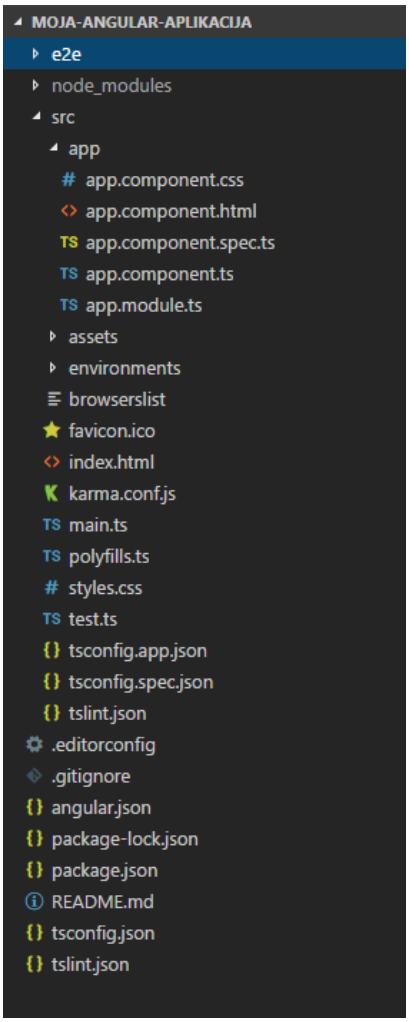
- cd ime-aplikacije
- ng serve

Umjesto ″ime-aplikacije″ potrebno je ubaciti željeno ime aplikacije, a zadnja linija pokreće inicijalnu aplikaciju u pregledniku na adresi *http://localhost:4200/* (slika 2.6.)[5].



Sl.2.6. *Inicijalna Angular aplikacija*

Na slici 2.7. prikazane su sve datoteke koje Angular-CLI instalira – početni *node* moduli, prva komponenta te razne konfiguracijske datoteke.



Sl.2.7. *Struktura početne Angular aplikacije*

2.4. Baze podataka i Firebase

Baza podataka je organizirani skup podataka, tj. zbirka zapisa pohranjenih na sustavan način. Najčešće korištene su relacijske baze podataka – baze podataka čija se organizacija zasniva na relacijskom modelu. To znači da se podaci organiziraju u skup relacija između kojih su jasno definirane veze. Razvojem i popularizacijom Interneta javio se problem količine podataka koji se očituje u poteškoćama tijekom prihvaćanja i obrade velikih količina podataka u kratkom vremenu. Iz toga je izašao novi tip modela podataka koji je puno jednostavniji i „slobodniji“. Riječ je o NoSQL bazama koje ne zahtijevaju definiciju sheme nego su dinamičke – tip i broj atributa nekog entiteta je otvoren, što omogućuje mijenjanje podataka po potrebi bez da se utječe na druge podatke. Najveća prednost ovakvih baza podataka je ta što je vrlo jednostavno preslikati strukturu podataka u određenom programskom jeziku na sličnu strukturu neke takve baze podataka. Uz to, ove baze imaju distribuiran način upotrebe – NoSQL sustavi su raspoređeni na veći broj računala gdje svaki sustav radi sa svojim podacima što ubrzava

dohvaćanje i manipulaciju podacima. Uz to, podaci se mogu modelirati tako da se maksimalno iskoristi efikasnost sustava – velika je skalabilnost.

Postoji nekoliko tipova NoSQL baza, a najčešće korišteni je sustav ključ – vrijednost. Ovaj sustav se sastoji od parova oblika ključ – vrijednost, tj. jednom tekstualnom ključu pridružuje se jedna vrijednost bilo kojeg tipa. Ove baze nemaju upitni jezik (zato se i zovu NoSQL) već one samo omogućavaju upravljanje podacima na temelju ključa. Za upravljanje podacima se uglavnom koriste tri operacije:

- *GET* – operacija koja vraća vrijednost pridruženu ključu,
- *PUT* – operacija koja ili dodaje novi par ključ-vrijednost ili pridružuje novu vrijednost već postojećem ključu,
- *DELETE* – operacija koja uklanja par ključ – vrijednost [6].

Firebase je Google platforma koja programerima omogućuje jednostavniju izradu aplikacija tako što im pruža veliku količinu alata i servisa. Firebase servisi se mogu podijeliti na dva dijela – dio za razvijanje i testiranje aplikacija te dio za marketing koji se bavi analitikom, proučavanjem korisnika i slično. Za programere je daleko najzanimljivija njihova baza podataka u stvarnom vremenu. Riječ je o NoSQL bazi koja je *cloud-based* te omogućuje spremanje podataka te sinkronizaciju između korisnika u stvarnom vremenu. Ova baza je zapravo jedan veliki JSON objekt iz kojeg se sa samo jednim API upitom može dohvatiti vrijednost željenih podataka ili spremiti objekte u bazu. Uz to, Firebase omogućuje korisnicima korištenje njihovog autentifikacijskog sustava koji omogućuje prijavu utemeljenu na mail/lozinka kombinaciji, broju mobilnog uređaja, korištenju računa raznih društvenih mreža i slično.

2.5. ChartJS

ChartJS je *JavaScript* biblioteka koja korisnicima omogućuje crtanje različitih tipova grafova koristeći HTML5 *canvas* element. Grafovi su responzivni te će se prilagoditi prostoru u koji su implementirani. Postoji osam tipova grafova – *line, bar, radar, dougnut/pie, polar area, bubble, scatter* i *area.* Ova biblioteka se nalazi pod MIT dozvolom te je cijeli projekt *open-source* tipa što ga čini dostupnim javnosti za korištenje i moguću nadogradnju [7].

3. PRIMJENA TEHNOLOGIJA U IZRADI WEB APLIKACIJE

Za izradu Angular aplikacije korišten je Angular CLI prema koracima opisanima u poglavlju 2.3., a aplikacija je nazvana *DemOs*. Pomoću naredbe *ng generate component ImeKomponente* kreirane su sljedeće komponente:

- *About*,
- *Application*,
- *Calendar* ,
    - *AddEvent*,
- *Header*,
- *HoursDone*,
- *Login*,
- *Register*,
- *Statistics*,
- *Students*,
- *Subjects*,
    - *AddSubject*,
    - *EditSubject*,
- *Users*.

Uz komponente, kreirani su i potrebni modeli za prihvaćenog studenta, događaj, kolegij i korisnika. Posebno su još dodani i servisi za prijavu na praksu, za autorizaciju, za kalendar, za kolegije te za korisnike. Na kraju, dodan je u tzv. *guard* koji se koristi pri autorizaciji korisnika.

3.1. Kreiranje korisnika i prijava u sustav

U aplikaciji postoje dva tipa korisnika – autorizirani i anonimni. Uloga autoriziranog korisnika je namijenjena profesorima, a anonimna svima ostalima – što studentima, što bilo kome tko posjećuje stranicu aplikacije. Za prijavu u sustav prvo se potrebno registrirati, tj. dodati korisnika u sustav. To je omogućeno samo unutar aplikacije, odnosno samo već autorizirani korisnici mogu dodavati druge korisnike. Ispod (slika 3.1.) je prikazana implementacija forme za registraciju unutar HTML datoteke:

```
<div class="row">
  <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
    <form (ngSubmit)="onRegister(f)" #f="ngForm" ngNativeValidate>
      <div class="form-group">
        <label for="firstName">Ime</label>
        <input type="firstName" id="firstName" name="firstName" ngModel class="form-control" required>
      </div>
      <div class="form-group">
        <label for="lastName">Prezime</label>
        <input type="lastName" id="lastName" name="lastName" ngModel class="form-control" required>
      </div>
    <div class="form-group">
      <label for="email">Mail</label>
      <input type="email" id="email" name="email" ngModel class="form-control" required>
    </div>
    <div class="form-group">
      <label for="password">Lozinka</label>
      <input type="password" id="password" name="password" ngModel class="form-control" required>
    </div>
    <button class="btn btn-warning" type="submit">Registriraj</button>
  </form>
  <mat-spinner *ngIf="loader" color="accent"></mat-spinner>
  </div>
</div>
```

Sl.3.1. *Forma za registraciju*

Kao što se vidi, za izradu su korištene Angular forme sa validacijom pomoću *ngNativeValidate* na samoj formi te sa *required* na svakom *input* polju. To onemogućuje registraciju korisnika ukoliko nisu sva polja unesena. Prilikom klika na gumb poziva se funkcija *onRegister()* unutar koje se uzimaju podaci iz forme te se pomoću servisa za autorizaciju na Firebase-u kreira korisnik sa navedenim mailom i lozinkom [8]. Na slici 3.2. prikazana je metoda unutar *authorization.service.ts* datoteke koja kreira novog korisnika [9].

```
register(email: string, password: string) {
    firebase.auth().createUserWithEmailAndPassword(email, password)
        .catch(
            error => console.log(error)
        )
}
```

Sl.3.2. *Metoda za registraciju*

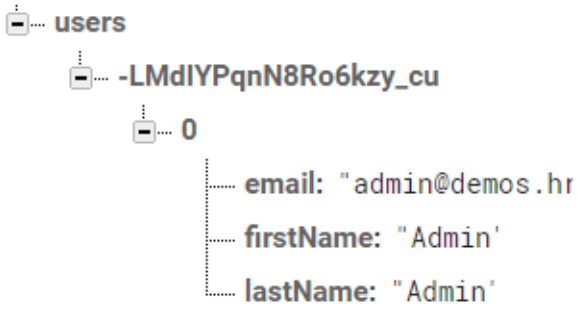
Kao što se vidi na gornjoj slici, metodi je potrebno predati mail tipa *string* te lozinku tipa *string*. Zatim se poziva Firebase metoda *createUserWithEmailAndPassword()* kojoj se predaju mail i lozinka te ih on sprema u bazu. Ukoliko dođe do pogreške, ona će se prikazati unutar konzole, što je omogućeno s *catch()* metodom.

Istovremeno se kreira i JSON objekt unutar baze sa ostalim korisničkim podacima (ime, prezime, mail). Za to se koristi HTTP metoda *POST* unutar *users.service.ts* (slika 3.3.)[10].

```
postUser(user: User[]) {
    const headers = new Headers({'Content-Type' : 'application/json '});
    return this.http.post('https://demos-d8319.firebaseio.com/users.json',
        user,
        { headers : headers });
}
```

Sl.3.3. *Metoda za spremanje u korisnika u bazu*

Za *POST* je potrebno proslijediti vezu do JSON objekta te objekt modela *User* koji želimo spremiti. Unutar baze spremljeni korisnik u JSON-u izgleda kao na slici 3.4. Svaki *POST* zahtjev prema bazi dobije svoj *hash* kao jedinstvenu oznaku pod kojom je unos spremljen.



Sl.3.4. *Korisnik unutar baze*

Nakon što se korisnika registrira, omogućen mu je pristup svim funkcionalnostima aplikacije vezanim za profesore – otvaranje demonstratura, prihvaćanje ili odbijanje studenata, otvaranje termina, pregled statistike i ostalo.

Za prijavu korisnika u sustav se također koristi forma kojoj se predaju mail i lozinka te nakon koje se poziva *login()* metoda unutar *authorization.service.ts* datoteke (slika 3.5.):

```
login(email: string, password: string) {
    firebase.auth().signInWithEmailAndPassword(email, password)
        .then(
            response => {
                firebase.auth().currentUser.getIdToken()
                .then(
                    (token: string) => this.token = token
                )
            }
        )
        .catch(
            error => console.log(error)
        )
}
```

Sl.3.5. *Metoda za prijavljivanje*

Metoda *signInWithEmailAndPassword()* prima unesene podatke o mailu i lozinki te vraća token sa Firebase-a koji sprema u varijablu *token* te koji omogućuje identifikaciju korisnika [11]. On se sprema u lokalno spremište (engl. *local storage*) te se prilikom osvježavanja stranice nije potrebno ponovno prijavljivati u sustav. To je izvedeno pomoću metode *loadUser()* koja provjerava postoji li već neki token unutar spremišta.

Kada se korisnik želi odjaviti iz sustava, potrebno je samo ukloniti token iz lokalnog spremišta (postaviti mu vrijednost na *null*), što je implementirano prilikom klika na gumb *Odjava* [11].

3.2. Zaštita od neovlaštenog pristupa

Da bi se onemogućilo neovlaštenim osobama da pristupe nečemu unutar aplikacije čemu ne bi trebali imati pristup, korišten je *AuthGuard*. *AuthGuard* je servis koji pri svakoj promjeni rute provjerava ima li osoba potrebne dozvole za pristup – u našem slučaju to je token. Iz *AuthGuard*-a se poziva metoda *isAuthenticated()* koja vraća *boolean* vrijednost na temelju koje će se korisnika propustiti u sustav ili mu zabraniti pristup. Unutar *app.module.ts* datoteke definirane su rute kao na slici 3.5. [12]:

```
const appRoutes: Routes = [
  { path: '', component: AboutComponent },
  { path: 'addEvent', component: AddEventComponent, canActivate: [AuthGuard] },
  { path: 'application', component: ApplicationComponent },
  { path: 'editHours', component: HoursDoneComponent, canActivate: [AuthGuard] },
  { path: 'calendar', component: CalendarComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent, canActivate: [AuthGuard] },
  { path: 'subjects', component: SubjectsComponent },
  { path: 'subjects/add', component: AddSubjectComponent, canActivate: [AuthGuard] },
  { path: 'subjects/edit/:id', component: EditSubjectComponent, canActivate: [AuthGuard] },
  { path: 'students', component: StudentsComponent, canActivate: [AuthGuard] },
  { path: 'statistics', component: StatisticsComponent, canActivate: [AuthGuard] },
  { path: 'users', component: UsersComponent, canActivate: [AuthGuard] }
];
```

Sl.3.5. *Rute*

Na rute na koje se treba primijeniti provjera iz *AuthGuard*-a dodano je *canActivate* polje u kojem se definiraju koje sve provjere se moraju izvršiti prije nego se korisnika preusmjeri na tu stranicu [13]. Kao što se vidi na slici, anonimni korisnik može pristupiti samo komponentama *About*, *Application*, *Calendar*, *Login* i *Subjects*. Također, na slici se vide veze do određenih komponenti. Ako korisnik unese vezu do komponente u za to predviđeno mjesto unutar pretraživača, on će biti preusmjeren samo ako ima dozvolu za to, tj. samo ako mu *AuthGuard* to dozvoli.

Uz onemogućavanje pristupa nekim stranicama, ovaj servis se koristi i za onemogućavanje prikaza nekih opcija neregistriranim korisnicima. Najbolji primjer je na *Header* komponenti koja će ovisno o prisustvu ili odsustvu tokena prikazati veze na druge stranice (slika 3.6.).

```
<div class="navbar-default">
  <ul class="nav navbar-nav">
    <li><a routerLink="/subjects">Otvorene prijave</a></li>
    <li><a routerLink="/calendar">Termini</a></li>
  <ng-template [ngIf]="authService.isAuthenticated()">
    <li><a routerLink="/statistics">Statistika</a></li>
    <li><a routerLink="/students">Studenti</a></li>
    <li><a routerLink="/users">Korisnici</a></li>
  </ng-template>
  </ul>
  <ul class="nav navbar-nav navbar-right">
    <ng-template [ngIf]="!authService.isAuthenticated()">
      <li><a routerLink="/login">Prijava<span class="glyphicon glyphicon-user"></span></a></li>
    </ng-template>
    <ng-template [ngIf]="authService.isAuthenticated()">
      <li><a routerLink="/register">Dodaj korisnika <span class="glyphicon glyphicon-plus"></span></a></li>
    <li><a routerLink="/about" (click)="onLogOut()">Odjava<span class="glyphicon glyphicon-off"></span></a></li>
  </ng-template>
  </ul>
</div>
```

Sl.3.6. *Komponenta zaglavlja*

Veze su enkapsulirane unutar *ng-template* oznaka na koje je stavljen *ngIf* uvjet koji će odlučiti hoće li se veze prikazati ili ne, ovisno o *boolean* vrijednosti koju mu metoda *isAuthenticated()* vrati. Na primjer, ukoliko mu metoda iz servisa vrati vrijednost *true*, veza na zaglavlju će s prikazati. Da bi se iz HTML-a mogao pozivati određeni servis te metode unutar njega, potrebno ga je samo navesti u konstruktoru odgovarajuće *TypeScript* datoteke te ga navesti unutar *import*-a te iste datoteke.

3.3. Otvaranje, prijava i brisanje demonstratura

Demonstrature otvaraju profesori, odnosno autorizirani korisnici. Podaci uneseni unutar forme spremaju se u bazu podataka koristeći *POST* metodu kojoj se šalje model *Subject* koji ima sljedeća polja (slika 3.7.):

```
export class Subject {
    public name: string;
    public year: number;
    public hours: number;
    public author: string;

    constructor(name: string, year: number, hours: number, author: string) {
        this.name = name;
        this.year = year;
        this.hours = hours;
        this.author = author;
    }
}
```

Sl.3.7. *Model Subject*

Kolegij ima četiri polja – naziv kolegija, godinu na kojoj se kolegij održava, broj potrebnih sati demonstrature te autora – profesora koji je otvorio demonstraturu.

Ti objekti se spremaju u bazu pod novim JSON poljem imena *subjects*. Unutar istoimene komponente podaci se prikazuju korisniku tako što se čitaju iz baze pomoću HTTP *GET* metode.

```
getSubjects() {
    return this.http.get('https://demos-d8319.firebaseio.com/subjects.json')
    .map(
        (response: Response) => {
            const data = response.json();
            return data;
        }
    );
}

postSubject(subjects: Subject[]) {
    const token = this.authService.getToken();
    const headers = new Headers({'Content-Type' : 'application/json '});
    return this.http.post('https://demos-d8319.firebaseio.com/subjects.json?auth=' + token,
        subjects,
        { headers : headers });
}
```

Sl.3.8. *GET i POST metode*

Na slici 3.8. prikazane se metode za *GET* i *POST* kolegija [10]. Za *GET* metodu nije potreban token, dok je za POST metodu potreban, što je dodatna zaštita za neovlaštenu manipulaciju podacima. Token se dodaje na kraj veze kojom se pristupa podacima unutar baze. On je dohvaćen pomoću autorizacijskog servisa, tj. pomoću njegove metode *getToken()*. Podaci se iz baze čitaju tako što se spremaju u *subjects* polje te se unutar HTML datoteke dinamički kreira lista kolegija, ovisno o broju podataka u navedenom polju, tj. o broju kolegija za koje je otvorena demonstratura. To je omogućeno korištenjem *ngFor* Angular opcije za iteraciju kroz polje unutar HTML datoteka. Ovisno o ulozi korisnika, bit će mu prikazan gumb za prijavu na demonstraturu (ako je anoniman) ili gumbi za brisanje prijave za demonstraturu te dodavanje nove demonstrature (ako je prijavljen u sustav).

Ako je korisnik anoniman, klikom na gumb *Prijavi se* preusmjerava ga se na komponentu za prijave na demonstraturu. Korisnik treba unijeti osobne podatke unutar forme koja se zatim sprema na bazu pod *appliedStudents* JSON polje. Podaci koji se traže su ime, prezime, mail adresa, prosjek ocjena te ocjena iz kolegija na koji se student želi prijaviti. Prilikom spremanja dohvaćaju se i podaci o predmetu za koji se student prijavljuje te se u *appliedStudent* objekt spremaju i oni – ime kolegija te broj predviđenih sati za odrađivanje. Navedeni objekt ima i polje *hours* koji se postavlja na nulu te koji će se koristiti prilikom povećanja broja odrađenih sati. Prilikom unosa valja navesti ispravne osobne podatke, posebice mail adresu na koju treba stići potvrda ukoliko je student prihvaćen, te prosjek ocjena na temelju kojega profesor može donijeti odluku o prihvaćanju na demonstraturu. Da bi se spriječio netočan unos ocjena iz kolegija i prosjeka, postavljene se minimalne i maksimalne vrijednosti za ta dva polja te je tako moguće

unijeti samo ocjene između 2 i 5, uz dodatak unosa na tri decimale za polje prosjek ocjena. Da bi se spriječio netočan unos za mail adresu, dodana je validacija unosa tipa *email* koja će obavijestiti korisnika ukoliko nije unio mail adresu ili ukoliko ona nije u ispravnom formatu.

Registrirani korisnik vidi prijave unutar komponente *students*. Komponenta je podijeljena na dva dijela, jedan za prijavljene studente, a drugi za prihvaćene studente. Metoda *getUsers()* se poziva unutar Angular metode *ngOnInit()* koja se uvijek prva poziva prilikom inicijalizacije komponente [14]. Pošto je *getUsers()* metoda pozvana unutar nje, i ona će se pozvati odmah pri inicijalizaciji te će se tako odmah prikazati podaci o prijavljenim studentima dobiveni iz baze na Firebase platformi. Podaci se čitaju pomoću HTTP metode *GET* unutar *applicationService* servisa te se spremaju u polje *applied* koristeći JavaScript metodu *push()* (slika 3.9.).

```
async getUsers() {
  await this.applicationService.getUser()
  .subscribe(
    (data: appliedStudent []) => {
      this.data = data;
      let keys = Object.keys(data);
      for ( var key in keys) {
        let uid = keys[key];
        for (let i of data[uid]){
          this.applied.push(i);
        }
      }
    },
    (error) => console.log(error)
  );
}
```

Sl.3.9. *Dohvaćanje i spremanje korisnika*

Kada se podaci spreme u polje *applied*, *ngFor* petlja unutar HTML datoteke dinamički kreira popis, ovisno o broju elemenata unutar polja *applied*, tj. ovisno o broju prijavljenih studenata (slika 3.10.).

```
<h3>Prijave:</h3>
<div class="list-group">
  <ul *ngFor="let apply of applied">
    <a class="list-group-item">
      <h4 class="list-group-item-heading">Kolegij: {{apply.subject}}</h4>
      <p class="list-group-item-text">Ime i prezime: {{apply.firstName}} {{apply.lastName}} </p>
      <p class="list-group-item-text">Email: {{apply.email}} </p>
      <p class="list-group-item-text">Prosjek ocjena: {{apply.averageGrade}} </p>
      <p class="list-group-item-text">Ocjena iz kolegija: {{apply.subjectGrade}} </p>
      <br>
      <button class="btn btn-warning" (click)="acceptStudent(apply)">Prihvati studenta i obavijesti ga mailom</button>
      <button class="btn btn-warning" (click)="deleteApplication(apply)">Odbij studenta</button>
    </a>
  </ul>
  </div>
```

Sl.3.10. *Popis prijavljenih studenata*

Za svakog prijavljenog studenta postoje i dva gumba – jedan za prihvaćanje studenta, a drugi za odbijanje. Ukoliko profesor želi prihvatiti studenta kao demonstratora, potrebno je kliknuti na gumb koji će u pozadini pokrenuti funkciju s dva zadatka: prebacivanje unutar baze iz *appliedStudents* u *acceptedStudents* JSON polja te slanje maila na adresu koju je student prilikom prijave na demonstraturu unio. Za izvršavanje oba zadatka potrebno je metodi kao argument proslijediti studenta na kojeg se odnosi prijava. Proslijeđeni objekt se zatim šalje u tri metode: prva metoda je *acceptStudent()* metoda unutar servisa za prijavu na demonstraturu koja objekt sprema u novo JSON polje, druga metoda je metoda za brisanje prijave iz baze prijavljenih studenata, a treća metoda je za slanje maila prihvaćenom studentu. Slanje maila se vrši preko *smtp2go* servisa. Za korištenje ovog servisa potrebno je prijaviti se na njihovu stranicu da bi se dobio odgovarajući API ključ. Slanje se vrši preko *fetch()* metode kojoj je potrebno proslijediti nekoliko argumenata: *smtp2go* link na koji se šalje određeni HTTP zahtjev, tip zahtjeva – u našem slučaju *POST*, odgovarajuća zaglavlja te tijelo samog maila. Unutar tijela potrebno je proslijediti API ključ koji smo dobili prilikom prijave na servis *smtp2go*, mail adresu pošiljatelja, mail adresu primatelja te sam sadržaj poruke. Na slici 3.11. je prikazana cijela metoda za slanje mail poruke [15].

```
sendEmail(appliant: appliedStudent) {
  let to = "Client <" + appliant.email + ">";
  fetch('https://api.smtp2go.com/v3/email/send', {
    method: 'POST',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      "api_key": "api-7DA80028B9F911E89616F23C91BBF4A0",
      "to": [to],
      "sender": "DemOs <noreply@demos.hr>",
      "subject": "DemOs",
      "text_body": "Rezultati natječaja",
      "html_body": "<h3>Čestitamo " + appliant.firstName + " !</h3> <br><p>Prihvaćeni ste na demonstraturu iz kolegija " + appliant.subject +" !</p>" +
      "<br> <p><i>DemOs</i></p>"
      ,
      })
      }).then(function(response) {
      return response.json();
      });
}
```

Sl.3.11. *Metoda za slanje poruke*

Podaci dobiveni iz proslijeđenog objekta *appliant* su iskorišteni pri definiranju adrese primatelja te samog tijela poruke. Ukoliko je student pri prijavi na demonstraturu unio netočnu mail adresu, neće mu stići mail poruka o potvrdi jer će ona otići na netočnu adresu što može dovesti do nepotrebnih komplikacija.

Za odbijanje studenta koristi se ista metoda koja i prilikom prihvaćanja studenta – metoda koja ga briše sa popisa prijavljenih studenata. Odbijanje se temelji na provjeri maila i predmeta što istovremeno znači dvije stvari: isti student se može prijaviti na više kolegija jer se brisanje iz baze temelji i na predmetu (ne samo na mailu), te ukoliko je student poslao više prijavnica sa istim podacima, sa samo jednim klikom obrisat će se sve iste prijave (čime se sustav može zaštiti od tzv. *spam*-anja prijavama). Unutar metode za brisanje poziva se metoda sa servisa. Ona prima samo *string* koji predstavlja *hash* koji Firebase stvara za svaki unos u bazu te će se tako obrisati svi podaci o studentu (slika 3.12.) [10].

```
deleteFromApplied(subjectToDelete: string) {
    const token = this.authService.getToken();
    return this.http.delete('https://demos-d8319.firebaseio.com/appliedStudents/'+subjectToDelete+'/0.json?auth=' + token)
    .toPromise();
}
```

Sl.3.12. *Brisanje studenata iz baze*

3.4. Broj odrađenih sati i statistika

Ukoliko je student prihvaćen na demonstraturu, njegovi podaci će biti prikazani unutar popisa na desnoj strani komponente *Students*. Za svakog studenta, uz osobne podatke, postoji i podatak o odrađenom broju sati. Ispod svakog studenta su dva gumba, jedan za brisanje studenta pomoću HTTP *DELETE* metode unutar *applicationService* servisa, te drugi koji vodi do komponente unutar koje je moguće povećati broj odrađenih sati za odabranog studenta. Nakon

što unese novi broj sati, podatak se sprema u bazu te se korisnika vraća na komponentu *Students* na kojoj je sada prikazan novi broj odrađenih sati. Prije samog preusmjeravanja korisnika nazad na komponentu *Students* valja napomenuti da se mijenja stanje varijable *loader*. Ova varijabla je pri dolasku na stranicu sakrivena (*loader = false*), a nakon klika na gumb ona postaje *true* te se prikazuje *spinner* koji je vezan za tu varijablu pomoću *ngIf*. Nakon *timeout*-a od dvije sekunde, korisnik se preusmjerava (slika 3.13.). Ovo je implementirano da bi korisniku bilo jasnije da se u pozadini nešto izvršava, tj. da se spremaju podaci na bazu.

```
setTimeout(() => {
  this.router.navigate(['/students']);
  },2000);
```

Sl.3.13. *Preusmjeravanje nakon dvije sekunde*

Prilikom svake promjene satnice, mijenja se i graf na komponenti *Statistics*. Za implementaciju grafa bilo je potrebno instalirati dodatan *node module – ChartJS*. To je izvršeno pomoću naredbe unutar konzole *npm install chart.js --save*. Pri inicijalizaciji ove komponente poziva se metoda iz servisa koja dohvaća sve prihvaćene studente. Ovoj metodi su potrebni podaci o prezimenu studenta, broju predviđenih sati te o broju odrađenih sati. Svaki od podataka se sprema u zasebno polje te će se u nastavku iskoristiti za samo crtanje grafa. Nakon što su svi podaci dohvaćeni, poziva se metoda *drawStudentChart()* (slika 3.14.)[7].

```
async drawStudentChart() {
  this.studentChart = new Chart('studentCanvas', {
    type: 'bar',
    data: {
      labels: this.chartDataLastName,
      datasets: [{
        data: this.chartDataSubjectHours,
        borderColor: "#3cba9f",
        backgroundColor: "#ffbb33",
        label: "Predviđeno sati"
      },
      {
        data: this.chartDataHours,
        borderColor: "#3cba9f",
        backgroundColor: "#33b5e5",
        label: "Odrađeno sati"
      },
      ]
    },
    options: {
      title: {
        display: true,
        text: "Statistika odrađenih sati",
        fontSize: 20
      },
      legend: {
        display: true
        }
      },
      scales: {
        xAxes: [{
          display: true
        }],
        yAxes: [{
          display: true
        }],
      }
    }
  )
}
```

Sl.3.14. *Metoda za crtanje grafa*

Ova metoda stvara novi graf tipa *bar*, na x os stavlja imena studenata od kojih svaki ima dva stupca u grafu – prvi koji prikazuje broj predviđenih sati koji su dodijeljeni studentu i drugi koji prikazuje trenutno stanje sati koje je student odradio, a na y os stavlja vrijednosti očitane iz polja vezanih za satnicu. Iznad grafa definiran je tekst koji će se prikazati („Statistika odrađenih sati“), zajedno sa legendom prikazanih podataka („Predviđeno sati“ i „Odrađeno sati“). Ukoliko korisnik prijeđe mišem preko podataka na grafu, dobit će također i objašnjenje koje sadrži što je prikazano i koliki je iznos prikazanog.

3.5. Unos termina odrade demonstrature

Termini odrade demonstrature su definirani unutar komponente *Calendar*. Ova komponenta ima dva moguća prikaza, ovisno o tome tko je korisnik. Neautorizirani korisnici

mogu samo pregledavati ovu komponentu, dok autorizirani mogu dodavati nove termine te brisati postojeće. Za unos novih komponenata je korištena forma, u kojoj su, uz četiri polja za unos tipa *text* (za ime kolegija, grupu za laboratorijske vježbe, ime i prezime studenta), i dodatna dva polja – jedan tipa *date* i drugi tipa *time* – za unos datuma i vremena održavanja termina. Na sva polja je dodana validacija koja onemogućuje neispravan ili prazan unos. Nakon spremanja termina, novi unos se može vidjeti u tablici za koju je također korištena opcija dinamičkog popunjavanja polja pomoću *ngFor*. Zamisao je da ove termine unosi profesor, po mogućnosti nakon nekakve konzultacije sa studentima, a ukoliko student nije u mogućnosti odraditi demonstraturu u navedenom terminu, potrebno je ili naći zamjenu ili obavijestiti profesora koji će zatim odrediti daljnje korake. Kada termin više nije aktualan (promijenio se ili je vježba održana), registrirani ga korisnik može obrisati jednostavnim klikom na gumb *Obriši termin.*

4. FUNKCIONALNOSTI WEB APLIKACIJE

U ovome poglavlju ovoga rada bit će opisane same funkcionalnosti web aplikacije potkrijepljene slikama i objašnjenima.

4.1. Prijava u sustav i registracija novih korisnika

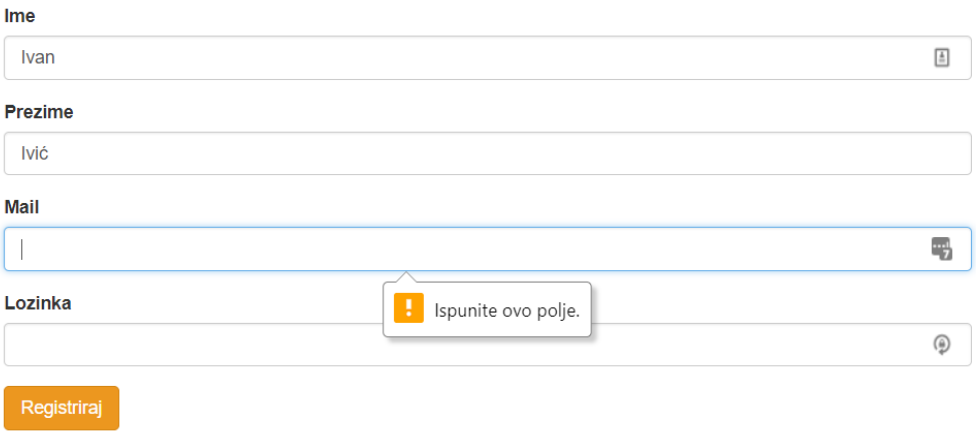
Prilikom pokretanja aplikacije korisnika se dovodi na početnu stranicu (slika 4.1.).

Sl.4.1. *Početna stranica*

Na vrhu stranice nalazi se zaglavlje sa vezama kojima može pristupiti korisnik dok nije prijavljen. Na samoj početnoj stranici nalazi se logo aplikacije te poruka dobrodošlice. Uz to, kreirana su i dva linka unutar teksta koja vode na neke od funkcionalnosti ove aplikacije. Gore desno se nalazi gumb imena *Prijava* koji ga vodi na prozor sa formom za prijavu (slika 4.2.).

Sl.4.2. *Stranica za prijavu*

Prijaviti se mogu samo korisnici koji su već u sustavu, koje je već netko unutar sustava prethodno dodao. Nakon prijave, ukoliko se korisnika autorizira, Firebase vraća token koji se sprema u lokalno spremište te se korisniku omogućuje pristup aplikaciji. Istovremeno se učitava novo zaglavlje, sada sa dodatnim vezama za komponente za koje sada ima pristup te koje sada smije vidjeti, te se korisnika preusmjerava na komponentu sa popisom studenata. Klikom na gumb *Dodaj korisnika* otvara se prozor sa formom za registraciju. Ovdje već registrirani korisnici mogu registrirati, tj. dodati nove korisnike te tako im omogućiti pristup funkcionalnostima aplikacije. Potrebno je unijeti ime i prezime korisnika te željenu mail adresu i lozinku. Ukoliko se korisnika pokuša registrirati sa nepotpunim podacima, izbacit će mu se poruka upozorenja (slika 4.3.). Isto vrijedi i za netočan format unosa mail adrese.

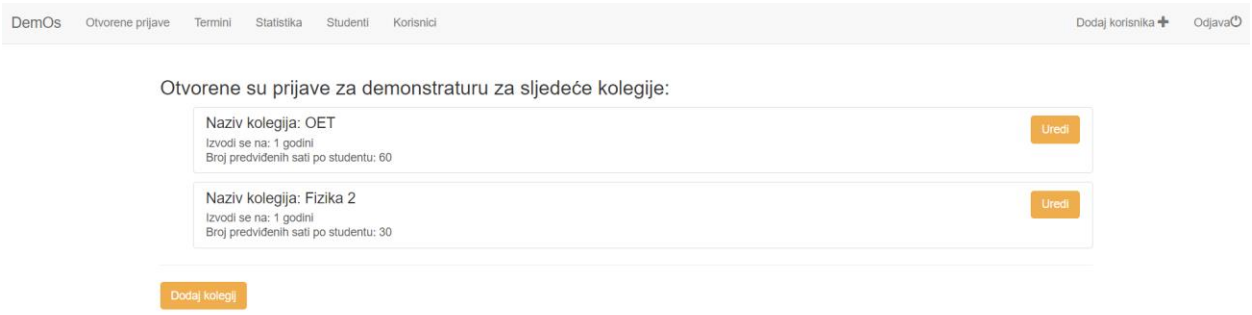


Sl.4.3. *Stranica za registraciju*

Novi korisnik se zatim može vidjeti unutar popisa ispod stavke *Korisnici* ponuđene u zaglavlju. Nakon ovoga korisnik se nalazi i u Firebase bazi unutar JSON objekta *users* te mu je sada omogućen pristup cjelokupnoj aplikaciji. Valja napomenuti da se u JSON objekt *users* ne sprema korisnička lozinka.

4.2. Otvaranje demonstratura te njihovo uređivanje

Otvorene prijave za demonstraturu se mogu pronaći pod stavkom *Otvorene prijave* na zaglavlju. Nakon učitavanja, stranica prijavljenom korisniku izgleda poput one na slici 4.4.

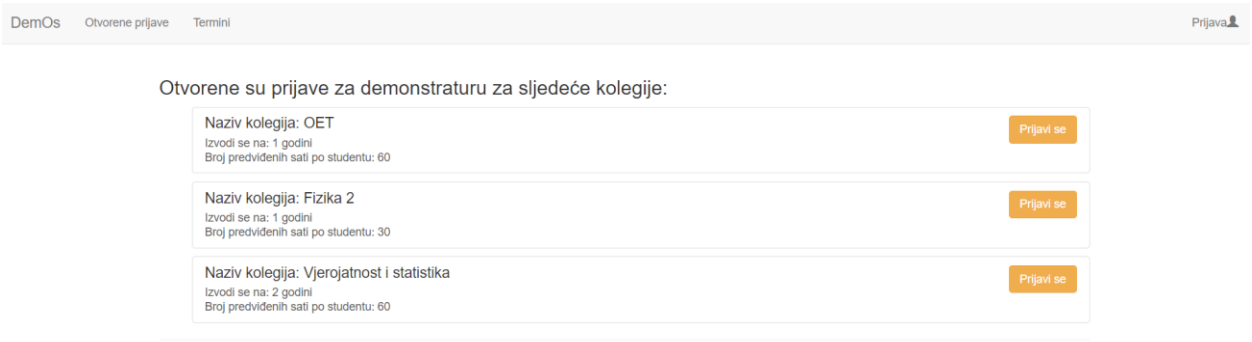


Sl.4.4. *Stranica sa popisom otvorenih demonstratura – prijavljeni korisnik*

Klikom na gumb *Dodaj kolegij* otvara se forma za dodavanje novog kolegija unutar koje je potrebno definirati naziv kolegija, godinu na kojoj se kolegij održava te broj sati po studentu. Kao i na svim ostalim formama, dodana je validacija unosa. Nakon klika na gumb *Spremi* pokreće se *spinner* te se novi kolegij sprema u bazu. Slijedi preusmjeravanje na prethodnu komponentu te se novi unos prikazuje na popisu. Ukoliko korisnik želi urediti neku od otvorenih prijava ili vidjeti detalje o prijavi, to mu je omogućeno klikom na gumb *Uredi* koja korisnika preusmjerava na komponentu na kojoj je to omogućeno.

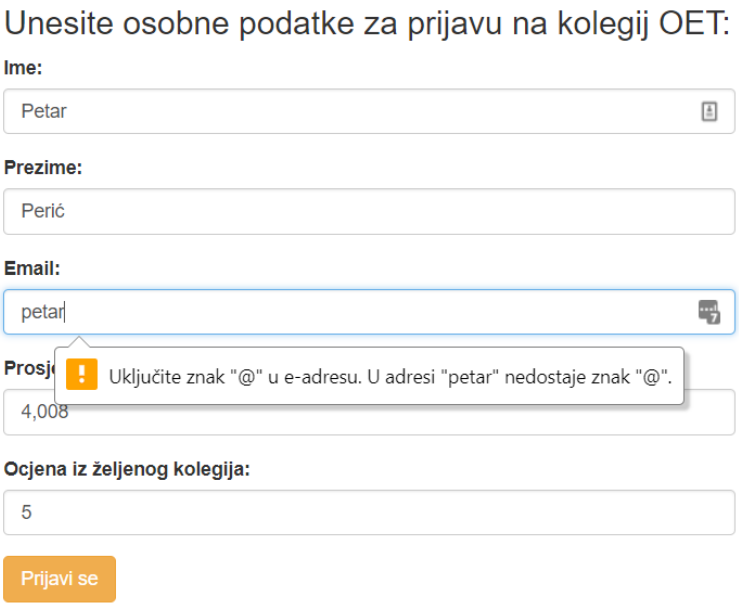
4.3. Prijava na demonstraturu i prihvaćanje studenata

Neautoriziranim korisnicima stranica sa prijavama izgleda poput one na slici 4.5.:



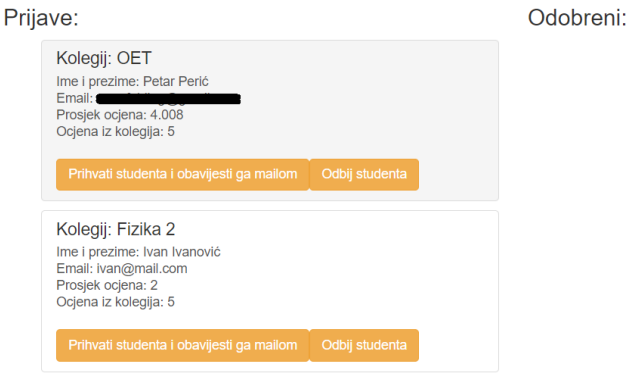
Sl.4.5. *Stranica sa popisom demonstratura – anonimni korisnik*

Sakrivene su opcije za dodavanje i brisanje demonstratura, a prikazan je novi gumb. Klikom na gumb *Prijavi se* korisnika se vodi na prozor sa formom za prijavu. Ukoliko korisnik pokuša poslati prijavnicu sa mail adresom u netočnom formatu, bit će o tome obaviješten prikladnom porukom (slika 4.6.). Isto vrijedi i za ostala polja za unos.



Sl.4.6. *Stranica sa prijavom za demonstraturu*

Prijavljeni korisnik vidi zaprimljene prijave u obliku kao na slici 4.7.:



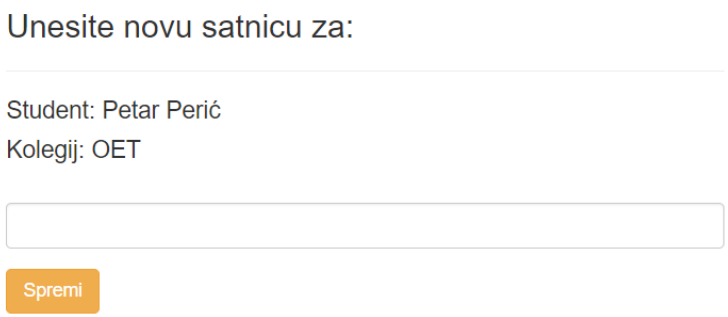
Sl.4.7. *Stranica sa popisom prijavljenih studenata na demonstraturu*

S lijeve strane prozora nalazi se popis prijavljenih studenata, a sa desne popis studenata kojima je demonstratura odobrena. Kada se studenta prihvati, pošalje mu se mail na navedenu adresu, koji, kada mu stigne, izgleda poput onog na slici 4.8.:

Sl.4.8. *Primjer primljenog maila*

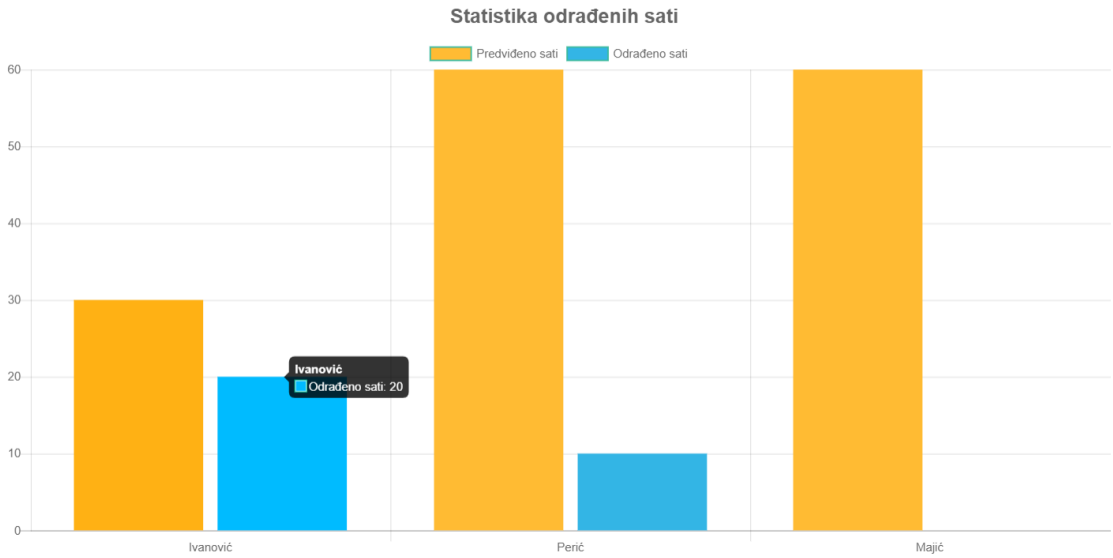
4.4. Promjena odrađenih sati i prikaz statistike

Kada se student odobri, prebacuje se iz *appliedStudents* u *acceptedStudents* JSON objekta te mu je tada moguće mijenjati broj odrađenih sati unutar forme kao što je ona prikazana na slici 4.9.



Sl.4.9. *Stranica za unos satnice*

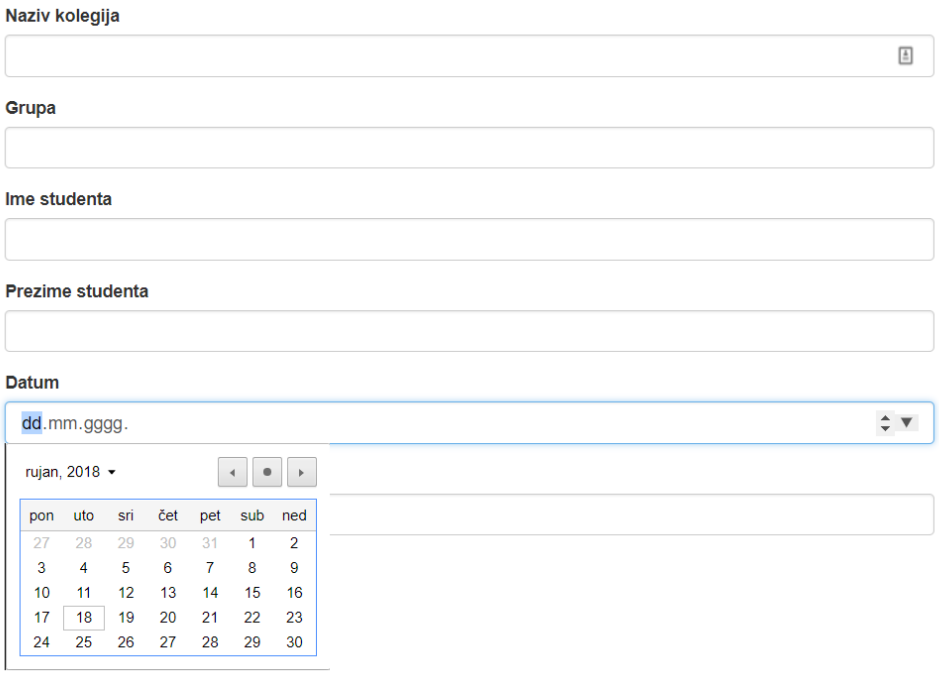
Statistika odrađenih sati po studentu može se vidjeti na grafu dostupnom pod *Statistika* unutar zaglavlja, a primjer je prikazan na slici 4.10. Na slici se također vidi i prikaz podataka prelaskom miša preko grafa – ime studenta, koji su podaci prikazani i koji im je iznos. Iznad grafa nalazi se naslov sa legendom po bojama koje su definirane unutar koda.



Sl.4.10. *Stranica sa statistikom*

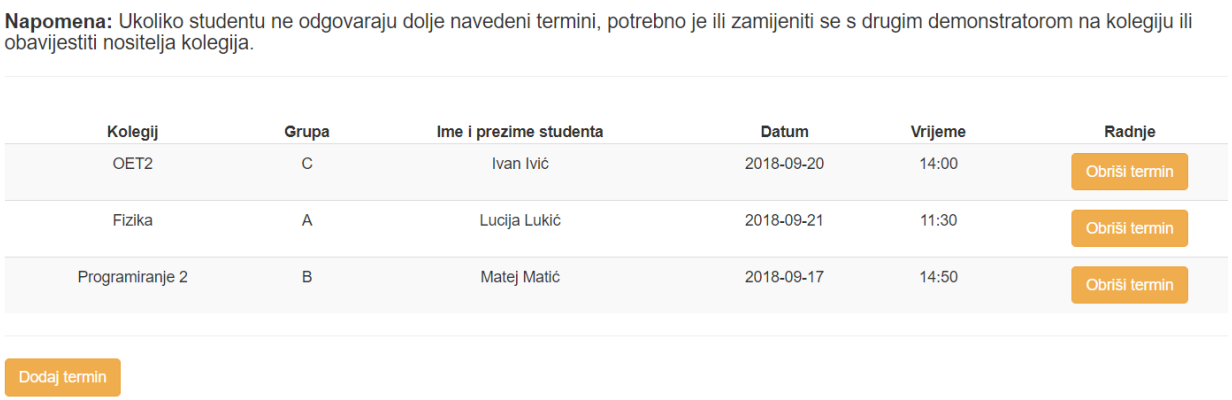
4.5. Unos termina demonstrature

Za unos termina se koristi forma dostupna preko gumba *Dodaj termin* ispod opcije zaglavlja *Termini*. Na slici 4.11. prikazana je forma s otvorenim izbornikom za datum. Potrebno je unijeti naziv kolegija, grupu za laboratorijske vježbe, ime i prezime studenta te termin održavanja.



Sl.4.11. *Stranica za unos termina demonstrature*

Dodani termini prikazani u tablici sličnog su izgleda kao i na slici 4.12. (prikaz za prijavljene korisnike),



Sl.4.12. *Stranica sa terminima odrade demonstrature – prijavljeni korisnik*

dok neautorizirani korisnici vide nešto poput prikaza na slici 4.13. Za njih su sakriveni gumbi za brisanje i dodavanje termina.

Sl.4.13. *Stranica sa terminima odrade demonstrature – anonimni korisnik*

5. ZAKLJUČAK

Web aplikacija za vođenje demonstrature omogućuje nastavnicima lakše otvaranje prijava za demonstrature na njihovim kolegijima, definiranje sati odrade te prihvaćanje zainteresiranih studenata. Nadalje, studenti se umjesto u referadu prijavljuju preko stranica aplikacije te im se, ukoliko budu prihvaćeni, šalje mail sa detaljima demonstrature. Najveća prednost ove aplikacije je raspored termina odrade u kojemu je jasno definirano kada koji student mora odraditi demonstraturu. Uz to, dana je i statistika za svakog studenta na svakom kolegiju.

LITERATURA

[1] D.,Nations: What Exactly Is A Web Application?, 22.02.2018., https://www.lifewire.com/what-is-a-web-application-3486637 (25.06.2018.)

[2] HTML, 20.12.2017., https://www.computerhope.com/jargon/h/html.htm (25.06.2018.)

[3] Introduction to CSS, 22.06.2018., https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Syntax (26.06.2018.)

[4] Architecture overview, https://angular.io/guide/architecture (26.06.2018.)

[5] https://github.com/angular/angular-cli/wiki (26.06.2018.)

[6] A.Stojanović: Osvrt na NoSQL baze podataka – četiri osnovne tehnologije, Polytechnic and Design, Vol. 4, No.1, 2016.

[7] Creating a Chart - http://www.chartjs.org/docs/latest/ (16.09.2018.)

[8] Forms - https://angular.io/guide/forms (14.09.2018.)

[9] Get started with Firebase authentication on websites - https://firebase.google.com/docs/auth/web/start (14.09.2018.)

[10] Read and write dana on the web - https://firebase.google.com/docs/database/web/read-and-write (14.09.2018.)

[11] Authenticate with Firebase using password – based accounts using Javascript https://firebase.google.com/docs/auth/web/password-auth (14.09.2018.)

[12] Rounting and navigation - https://angular.io/guide/router (15.09.2018.)

[13] CanActivate - https://angular.io/api/router/CanActivate (15.09.2018.)

[14] OnInit - https://angular.io/api/core/OnInit (14.09.2018.)

[15] POST /email/send - https://apidoc.smtp2go.com/documentation/#/POST%20/email/send (16.09.2018.)

SAŽETAK

Web aplikacija za vođenje demonstrature je aplikacija koja omogućuje jednostavniju komunikaciju između studenata demonstratora i nastavnika. Nastavnici imaju mogućnost otvaranja određenog broja demonstratura s određenim brojem sati. Studenti koji se prijave na željene kolegije su ili odbijeni ili prihvaćeni, u kojem slučaju dobivaju mail potvrde. Što se same odrade demonstrature tiče, aplikacija nudi nastavnicima mogućnost unosa termina demonstrature. Uz to, nastavnici mogu vidjeti statistiku za svakog studenta o tome koliko su sati odradili na određenom kolegiju od predviđenog broja. Za izradu se koristio Angular.

Ključne riječi: Angular, Firebase, ChartJS, demonstratura, web aplikacija

ABSTRACT

Web application for conducting demonstration

Web application for conducting demonstration is an application which allows easier communication between students who are demonstrators and teachers. Teachers can open a number of places for a demonstration with a certain number of hours. Applied students can be denied or accepted, in which case they will receive an e-mail with all the details. As far as the demonstration scheduling, the application offers the possibility of entering the times when the demonstration will take place. Additionally, teachers can see statistics for each student – how many hours they have worked on a specific course. Angular was used for the development of the application.

Keywords: Angular, Firebase, ChartJS, demonstration, web application

ŽIVOTOPIS

Emilia Felding rođena je 1.lipnja 1994. u Slavonskom Brodu. U Oriovcu je završila osnovnu školu te je nakon toga upisala Klasičnu gimnaziju fra Marijana Lanosovića u Slavonskom Brodu. Završetkom gimnazije seli se u Osijek gdje upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija. Tamo završava preddiplomski sveučilišni studij elektrotehnike, smjer Komunikacije i informatika te potom upisuje diplomski studij na istom fakultetu, smjer Mrežne tehnologije. Od stranih jezika služi se engleskim i njemačkim.

Emilia Felding

PRILOZI

[1] Link za Github repozitorij - https://github.com/gregormali/DemOs/tree/develop