

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

STJEPAN MARKOVČIĆ

**Razvoj web platforme  
za ugovaranje stručne prakse - Frontend**

Završni rad

Pula, Rujan, 2020. godine

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

STJEPAN MARKOVČIĆ

**Razvoj web platforme  
za ugovaranje stručne prakse - Frontend**

Završni rad

**JMBAG: 0303075497, redoviti student**

**Studijski smjer: Informatika**

**Kolegij: Poslovni informacijski sustavi**

**Mentor: doc. dr. sc. Darko Etinger**

**Komentor: doc. dr. sc. Nikola Tanković**

Pula, Rujan, 2020. Godine



## IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani *Stjepan Markovčić*, ovime izjavljujem da je ovaj završni rad rezultat isključivo mogega 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

*Stjepan Markovčić*

U Puli, Rujan, 2020. godine



## IZJAVA O KORIŠTENJU AUTORSKOG DIJELA

Ja, *Stjepan Markovčić* dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom *Razvoj web platforme za ugovaranje stručne prakse – Frontend* 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.

Student

*Stjepan Markovčić*

U Puli, Rujan, 2020. godine

<b>Sadržaj</b>	
<b>Uvod</b>	1
<b>Korištene tehnologije</b>	1
<b>Neregistriran korisnik</b>	2
<b>Poslodavac</b>	2
<b>Administrator</b>	2
<b>Student</b>	3
<b>Pohrana podataka</b>	3
<i>Korisnički podaci</i>	3
<i>Informacije dodane u sustav</i>	3
<b>Paginacija</b>	4
<b>Projekt</b>	5
<i>Dodavanje projekta</i>	5
<i>Uređivanje podataka</i>	6
<i>Brisanje projekta</i>	7
<b>Brisanje poslodavca i studenta</b>	8
<b>Promjena lozinke</b>	10
<b>Pregledi</b>	11
<b>Galerija</b>	12
<i>Header slike</i>	12
<i>Logo</i>	13
<i>Dodavanje slike</i>	14
<b>Odabir projekata</b>	16
<b>Alokacija studenata</b>	17
<b>Popunjavanje prijavnice</b>	19
<b>Predaja dnevnika prakse</b>	20
<b>Dojmovi/recenzije</b>	22
<b>Tablica studenata</b>	23
<b>Postavljanje uputa</b>	24
<b>Sortiranje</b>	25
<b>Pretraživanje</b>	28
<b>Korištene komponente i paketi</b>	28

## Uvod

Vođeni idejom olakšavanja i optimiziranja ugovaranja stručne prakse na FIPU te poboljšanja iskustva kako za studente tako i za poslodavce, kolega Filip Ožbolt i ja smo odlučili napraviti web aplikaciju "Moja-praksa".

Kolega je radio backend dio, a ja pripadajući frontend. Web aplikacija se sastoji od 4 sučelja ovisno o stupnju autorizacije pa tako imamo sučelje za:

- Neregistriran korisnik
- Poslodavac/partner
- Admin
- Student

Github repozitorij projekta: [https://github.com/mstjepan28/moja\\_praksa\\_frontend](https://github.com/mstjepan28/moja_praksa_frontend)

## Korištene tehnologije

Tehnologije korištene u razvoju frontend sučelja za web aplikaciju su:

- HTML i CSS – Za izgled web aplikacije
- JavaScript – Funkcionalnost web aplikacije
- VueJS<sup>1</sup> – Framework
- NPM<sup>2</sup> – Upravljanje paketima
- Axios<sup>3</sup> – Komunikacija sa backend dijelom aplikacije

---

<sup>1</sup> VueJS - <https://vuejs.org/>

<sup>2</sup> NPM - <https://www.npmjs.com/>

<sup>3</sup> Axios - <https://github.com/axios/axios>

**Neregistriran korisnik**

Neregistrirani korisnici imaju najmanje funkcija od svih, to jest sve što je omogućeno takvome korisniku jest generalni pregled projekata i partnera. Kada neregistriran korisnik pokuša otvoriti projekt ili partnera da vidi više informacija o njima, sustav traži da se prijavi sa e-mail adresom i lozinkom.

**Poslodavac**

Poslodavac dodaje projekte koje student zatim odabire. Također dodaje neke informacije o sebi koje bi studentima dale uvijek u njegovo poslovanje te na taj način studenti mogu odabrati ako žele izabrati neki njegov projekt.

Neke od tih informacija su tehnologije koje poslodavac koristi, opis poslodavca te adresa na kojoj se nalazi. Isto tako, poslodavcu se pruža mogućnost da doda poveznice na svoj Facebook i Twitter profil, isto kao i na svoju vlastitu web stranicu gdje bi studenti mogli pronaći više informacija o poslodavcu.

**Administrator**

Administrator ima mogućnost dodavanja partnera i projekata ali njegova glavna funkcija jest odobravanje projekata studentima. Također, nakon što student dogovori detalje prakse i ispuni prijavnicu, administrator vidi tu prijavnicu. Kao i nakon što student preda dnevnik prakse, administrator može preuzeti taj dnevnik prakse.

Administrator ima uvid u tijek prase svakog studenta u obliku tablice ili kao kartice.

Za pomoć pri izvršavanju prakse, administrator može postaviti predložak dnevnika prakse te opisati postupak izvršavanja prakse studentima kroz dodavanje instrukcija za obavljanje prakse.

## **Student**

Student koristi sustav kako bi pronašao 3 projekta koja mu se sviđaju, te će mu se jedan od ta 3 biti dodijeljen. Nakon što mu je projekt dodijeljen, student kontaktira poslodavca i dogovara detalje prakse. Kada je student dogovorio detalje prakse, ispunjava prijavnicu te počinje sa praksom.

Kada student završi sa praksom ispunjava dnevnik prakse te ga predaje u sustav gdje ga administrator može pregledati.

## **Pohrana podataka**

Podatke koje dohvaćamo sa backenda možemo podijeliti u dvije skupine:

- korisnički podaci
- informacije dodane u sustav

### ***Korisnički podaci***

Korisnički podaci su podaci koji se dohvate sa backenda prilikom prijave u sustav, to su podaci koje student i poslodavac unesu u sustav prilikom registracije(npr. ime, prezime, lozinka) i oni koje ubaci naknadno(npr. slike). Iznimka je administrator čiji korisnički podaci se svode na e-mail adresu i lozinku.

Prilikom prijave u sustav, korisnički podaci se dohvaćaju sa backenda zajedno sa tokenom te se pohranjuju u localStorage. Korisnički podaci se pohranjuju u localStorage da bi se sačuvali prilikom osvježavanja stranice, to jest da se korisnik ne mora ponovno prijavljivati u sustav.

### ***Informacije dodane u sustav***

Pod time se misli na podatke poput informacija o poslodavcima, o studentima te o projektima. Kada se takvi podaci dohvaćaju, primjerice, kada se dohvaćaju projekti, dohvaćaju se svi te ih se pohranjuje u lokalnu varijablu *project\_list* unutar *src/store.js*.

Razlog za pohranu u lokalnu varijablu je da bi izbjegli ponovne pozive na backend kada to ne bi bilo potrebno. Isto tako, sustav je osmišljen na način da nije velika mogućnost da će se u njemu naći velik broj takvih podataka koje bi izazivale poteškoće u radu prilikom dohvaćanja.



Ideja iza ovakvog pristupa jest da kada primjerice student uđe u sustav i dohvati sve podatke, ti podaci će se dalje dohvaćati iz te lokalne varijable i biti će obrađivani (npr. sortiranje) u lokalnoj varijabli.

```
async getPartnerList(){
  if(!this.store.partner_list) this.store.partner_list = await Partners.getPartners();
  this.partner_list = this.store.partner_list.slice(0, this.items_per_page)
},
```

*Funkcija 1 src/view/Partners.vue - Dohvaća listu partnera*

Ova funkcija prikazuje provjeru ako postoje neki podaci u lokalnoj varijabli, ako ne postoje, tada ih dohvaća sa backenda i sprema u listu.

## Paginacija

Paginacija funkcionira preko komponente *vuejs-paginate*<sup>4</sup>.

Paginacija funkcionira na način da se sa backenda dohvaća sveukupan broj dokumenata koji se nalaze u određenoj kolekciji te se prema tome određuje ukupan broj stranica.

```
async getTotalPages(){
  const total_items = await App.getDocAmount();
  this.total_pages = Math.ceil(total_items.partnersCounter / this.items_per_page);
},
```

*Funkcija 2 src/view/Projects.vue - Izračunaj ukupan broj stranica*

Funkcija za određivanje broja stranica dobiva ukupan broj dokumenata te taj broj dijeli sa brojem dokumenata koje želimo prikazati na stranici, te rezultat tog dijeljenja zaokružimo na veći broj.

```
async clickCallback(pageNum){
  const first_item = pageNum * this.items_per_page - this.items_per_page + 1
  const last_item = pageNum * this.items_per_page

  const saved_partners = this.store.partner_list
  if(saved_partners.length < last_item && saved_partners.length >= first_item){
    this.partner_list = this.store.partner_list.slice(first_item-1, saved_partners.length+1)
  }
  else{
    this.partner_list = this.store.partner_list.slice(first_item-1, last_item)
  }
}
```

*Funkcija 3 src/view/Projects.vue - Promjeni sadržaj stranice*

---

<sup>4</sup> VueJS-Paginate izvor: <https://www.npmjs.com/package/vuejs-paginate>

Kada korisnik odabere stranicu iz paginacije, pokreće se funkcija koja određuje raspon elemenata u listi koje treba prikazati. Taj raspon se određuje tako da se pronađe indeks prvog i zadnjeg dokumenta kojeg treba prikazati te se zatim dohvate svi dokumenti koji su između tih indeksa pomoću funkcije `.slice()`.

Ukoliko nema dovoljno elemenata više u listi, dohvaća ih se onoliko koliko ih je preostalo, zbog tog razloga se stranice zaokružuju na veći broj.

Indeks prvog dokumenta se izračunava po na način:

$$\text{brojStranica} * \text{brojDokumenataPoStranici} - \text{brojDokumenataPoStranici} + 1$$

Indeks posljednjeg dokumenta se izračunava kao:

$$\text{brojStranica} * \text{brojDokumenataPoStranici}$$

## **Projekt**

Projekti se prikazuju studentima koji ih zatim pregledavaju i odabiru 3 projekta koja žele odraditi u sklopu prakse. Nakon što studenti odaberu projekte, administrator odobrava jedan od tih projekata studentu koji zatim stupa u kontakt sa poslodavcem.

### ***Dodavanje projekta***

Poslodavac dodaje projekt na način da popunjava formu sa svim traženim informacijama. Te informacije su: naziv poslodavca(popunjava se automatski), opis projekta, broj studenata potrebnih za izvršavanje projekta, kontakt(npr. email ili broj), tehnologije koje će se koristiti, preferanse, potrebna oprema, trajanje prakse, lokacija i napomena.

U pozadini se dodaju `partnerID` koji govori kojem poslodavcu je projekt pridružen i `userID` koji govori koji korisnik je vlasnik projekta. Ako administrator stvara projekt, dodaje se također i `created_by_admin`.

Pri dodavanju projekta od strane administratora, administrator mora odabrati kojemu će poslodavcu dodijeliti projekt. Poslodavac mora biti također kreiran od strane administratora da bi mu mogao dodijeliti taj projekt.

```

async uploadProject(){
  this.project_info.allocated_to = new Array(parseInt(this.list_size)).fill(false);
  return await Projects.AddProject(this.project_info);
},

```

*Funkcija 4 src/view/addProject.vue - Stvara listu slobodnih mjesta u projektu*

Prije samog dodavanja projekta, stvaramo polje veličine koliko je potrebno studenata za obavljanje tog projekta. Stvaramo polje te veličine i popunjavamo ga sa vrijednostima *false*. Kada se studentu odobri projekt, taj *false* će se zamijeniti sa id studenta kojega želimo alocirati na taj projekt. Koliko postoji *false* vrijednosti u polju toliko je ostalo slobodnih mjesta na projektu.

### **Uređivanje podataka**

Podaci poslodavca, studenta i projekta se većinom uređuju i ažuriraju na isti način.

Sve na što se svodi uređivanje podataka jest da vlasnik klikom na gumb 'Uredi' otvara način za uređivanje, promjeni neke vrijednosti te ih pohrani. Tada se cijeli izmijenjeni objekt šalje na backend.

```

async update_project(){
  await Projects.UpdateProject(this.project_info, this.id, true);

  this.store.project_list = await Projects.getProjects();
  this.getProjectInfo();

  this.edit_enabled = false;
},

```

*Funkcija 5 src/view/projectInfo.vue - Ažuriranje projekta*

Razlika u uređivanju podataka nastupa pri ažuriranju lokalnih podataka. Partner i projekt se ažuriraju na isti način, jedina razlika je funkcije koje se pozivaju.

```

async updateUser(){
  const token = this.user_data.token

  delete this.user_data.token
  const result = await App.updateUser(this.user_data, true);

  this.user_data.token = token

  if(!result){
    this.modal_error = "Prilikom pokušaja izmjene korisničkih podataka došlo je do greške";
    $('#error_modal').modal('show')
  }
  else
    localStorage.setItem('user', JSON.stringify(this.user_data));

  this.edit_enabled = false;
},

```

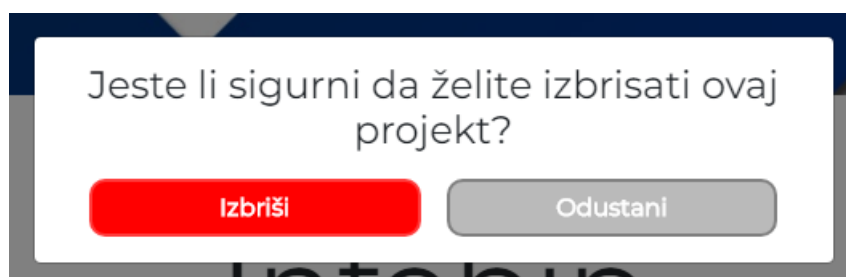
*Funkcija 6 src/views/UserInfo.vue - Ažuriranje podataka studenta*

Kod uređivanja korisničkih podataka studenta, token pohranjujemo u zasebnu varijablu i brišemo iz objekta kojeg šaljemo na backend da se ne bi pohranio na bazi. Ako dođe do greške iskače modal koji obavještava korisnika o grešci ali ako je sve uredu, podaci se pohranjuju u localStorage i zatvara se način uređivanja.

### **Brisanje projekta**

Korisnik briše svoj projekt klikom na 'Izbriši' gumb na stranici tog projekta. Pritiskom na taj gumb pojavljuje se modal za potvrdu brisanja projekta.

Ako se projekt obriše a student je alociran na njega, tada taj student gubi dodijeljeni projekt.



*Slika 1 Modal za potvrdu brisanja projekta*

```

async delete_project(){
  $('#deleteProject').modal('hide');
  $('#DeleteConfirmation').modal({
    backdrop: 'static',
    keyboard: false,
  })

  await Projects.DeleteProject(this.id, false);

  this.store.project_list = await Projects.getProjects();
},

```

*Funkcija 7 Src/views/ProjectInfo.vue - Brisanje projekta*

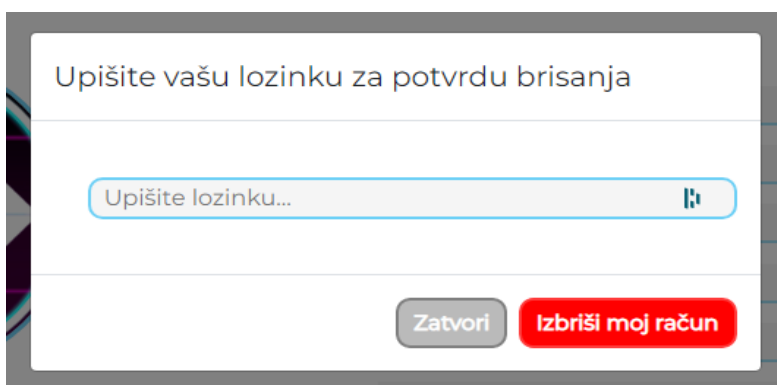
Pritiskom na gumb 'Izbriši' korisnik potvrđuje brisanje projekta te se pokreće poziva se funkcija za brisanje. Funkcija zatvara trenutni modal te otvara novi koji govori korisniku da je projekt izbrisan. Na backend šaljemo id projekta za brisanje te nakon što se projekt izbriše, dohvaća ponovno se dohvaća lista svih projekata.

### **Brisanje poslodavca i studenta**

Za brisanje korisničkog računa, korisnik odlazi na svoj profil te na uređivanje svog profila gdje odabire opciju za brisanje računa.

Prije nego što student ili poslodavac može izbrisati svoj korisnički račun, moraju unijeti svoju lozinku da bi potvrdili brisanje. Također, kada administrator želi izbrisati nekog poslodavca mora unijeti svoju lozinku da bi to potvrdio.

Brisanje poslodavca uzrokuje brisanje svih njegovih projekata.



*Slika 2 Modal za brisanje korisničkog računa studenta*

```

async delete_partner(){
  $('#deletePartner').modal('hide');
  if(this.user_data.account_type == 'Admin') this.partners_info.account_type = 'Admin'
  this.partners_info.password = this.current_password

  const response = await Partners.UpdatePartner(this.partners_info, this.id, false);

  if(!response){
    this.modal_error = "Došlo je do greške prilikom brisanja";
    $('#error_modal').modal('show');

    return;
  }

  this.store.partner_list = await Partners.getPartners();
  this.store.project_list = await Projects.getProjects();

  if(this.user_data.account_type == "Poslodavac"){
    Auth.logout();
    this.$router.push({ name: 'Login' });
  }
  else this.$router.push({ name: 'Partners' });
},

```

*Funkcija 8 Src/views/partnerInfo.vue - Brisanje partnera*

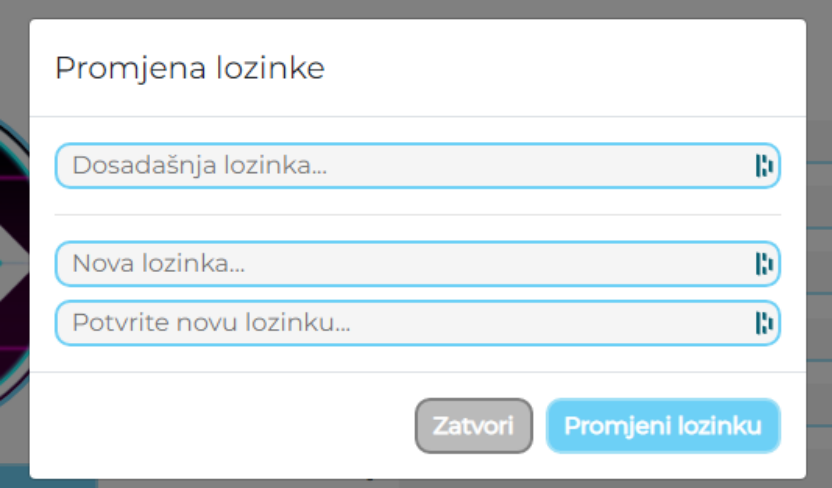
Nakon unosa lozinke i pritiska na gumb 'Izbriši' briše se određeni korisnik ili poslodavac. Ukoliko je lozinka pogrešna, dolazi do greške i pojavljuje se modal koji ispisuje tu grešku.

Razlika u brisanju poslodavca i studenta je u tome što kod brisanja poslodavca ažuriramo lokalne podatke o projektima i poslodavcima te ako administrator briše poslodavca, tada odlazi na rutu *Partners* a ne na login.

## Promjena lozinke

Kada korisnik želi promijeniti lozinku, mora otići na svoj profil te omogućiti uređivanje, tada odabire opciju za promjenu lozinke.

Nakon što korisnik odabere tu opciju, pojavljuje se modal za promjenu lozinke u koji unosi dotadašnju lozinku te dvaput novu.



Slika 3 Modal za promjenu lozinke

Korisnik mora unijeti novu lozinku koja se razlikuje od stare te se nova lozinka mora podudarati sa onom ponovno unesenom.

```
async change_password(){
  if(!this.passwordCheck()) return;
  $('#change_password_modal').modal('hide')

  const result = await Auth.changePassword({'oldPassword': this.current_password, 'newPassword': this.new_password});

  if(!result){
    this.modal_error = "Prilikom pokušaja promjene lozinke došlo je do greške";
    $('#error_modal').modal('show')
  }

  this.current_password = this.new_password = this.confirm_password = undefined;
  this.edit_enabled = false;
},
```

Funkcija 9 Src/views/PartnerInfo.vue - Promjena lozinke

Nakon što se provjeri ako je nova šifra uredi, šalje se na backend, ako backend vrati grešku pojaviti će se modal koji obavještava korisnika o toj grešci. U suprotnome, zatvara se način uređivanja i lozinka je uspješno promijenjena.

## Pregledi

Poslodavac i projekt zbrajaju posjete na stranicama sa njihovim informacijama. Ideja toga je da student vidi što ostali studenti smatraju zanimljivim, to jest koje projekte i poslodavce najviše posjećuju. Ako je trenutni korisnik vlasnik tog projekta ili partnera, tada se pregledi ne povećavaju.

```
checkIfOwner(){
  if(this.user_data._id == this.id)
    this.isOwner = true;

  else if(this.user_data.account_type == "Admin" && this.partners_info.created_by_admin)
    this.isOwner = true;
},

async addView(){
  if(this.isOwner) return;

  this.partners_info.views++;

  await Partners.addPartnerView({
    '_id': this.id,
    'views': this.partners_info.views,
    'collectionName' : 'partners'
  });
},
```

*Funkcija 10 Src/views/PartnerInfo.vue - Projera vlasnika i dodavanje pregleda*

Funkcija checkIfOwner() provjerava ako je trenutni korisnik vlasnik projekta, to jest poslodavca na čijoj stranici se trenutno nalazi. Provjera se razlikuje za projekt i za poslodavca, dok provjera za administratora ostaje ista.

- Poslodavac – ako je id trenutnog korisnika isti kao id ovog poslodavca, tada je trenutni korisnika vlasnik
- Projekt – ako je id trenutnog korisnika isti kao onaj u kome pripada ovaj projekt(varijabla partnerID), tada je trenutni korisnik vlasnik
- Administrator – ako je trenutni korisnik administrator i projekt/poslodavac je stvoren od strane administrator(varijabla created\_by\_admin), tada je trenutni korisnik vlasnik

Funkcija addView() provjerava ako je trenutni korisnik vlasnik, ako je tada se pregled ne dodaje, u suprotnome, na trenutne preglede se dodaje +1 te se šalje na backend.



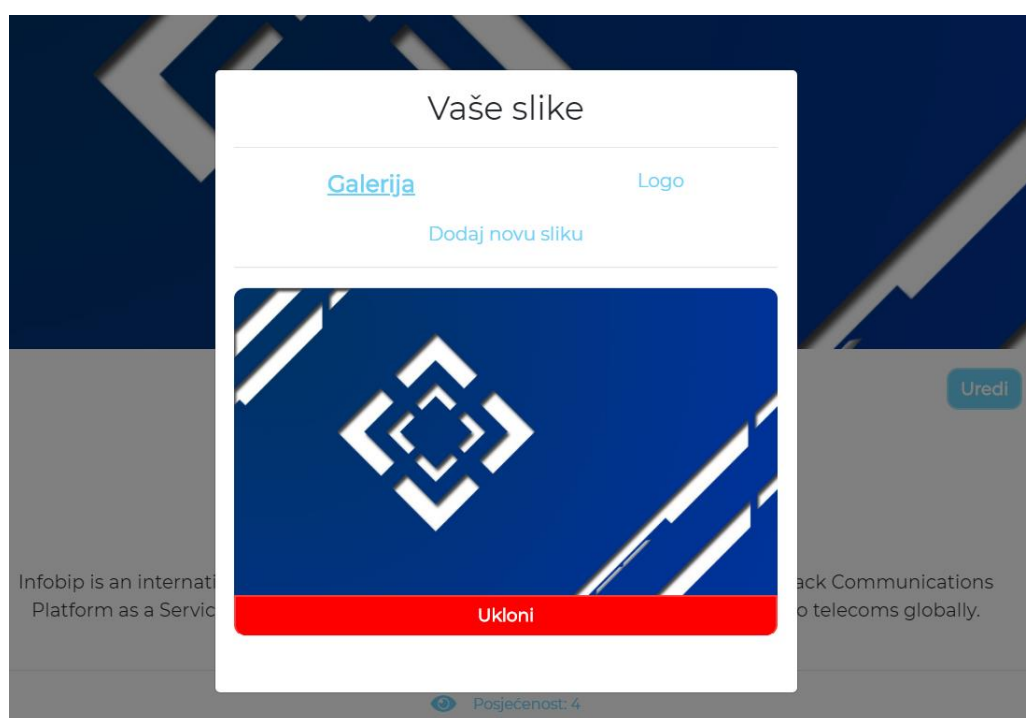
## Galerija

Student može dodati samo svoj logo, to jest profilnu sliku dok poslodavac ima mogućnost dodavanja i header slike. Kada poslodavac doda nove header slike ili novi logo, te slike se automatski primijeniti i na projekt. Ukoliko poslodavac ne postavi logo ili header slike, prikazivat će se zadane slike.

Galerija se otvara klikom na gumb 'Galerija' na stranici poslodavca, gumb se neće prikazati ako trenutni korisnik nije vlasnik tog poslodavca. Klik na taj gumb otvara modal sa komponentom galerije. Poslodavac tada ima izbor ako želi dodati/ukloniti header sliku ili želi dodati novi logo.

## Header slike

Header slika može biti mnogo te one zahtijevaju poslodavca da ih izbriše. Header slike se prikazuju na vrhu profila partnera i projekata tog partnera. Slike se izmjenjuju jedna za drugom te je tako u nekom trenutku vidljiva samo jedna.

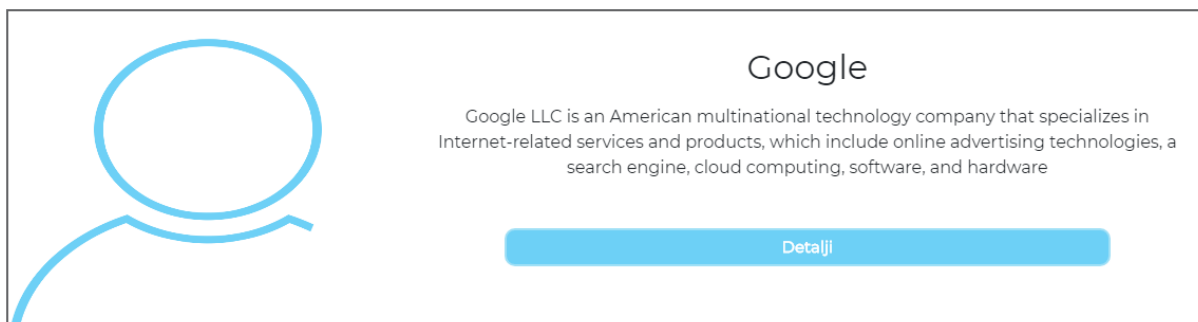


Slika 4 Prikaz header slike te gumb za uklanjanje

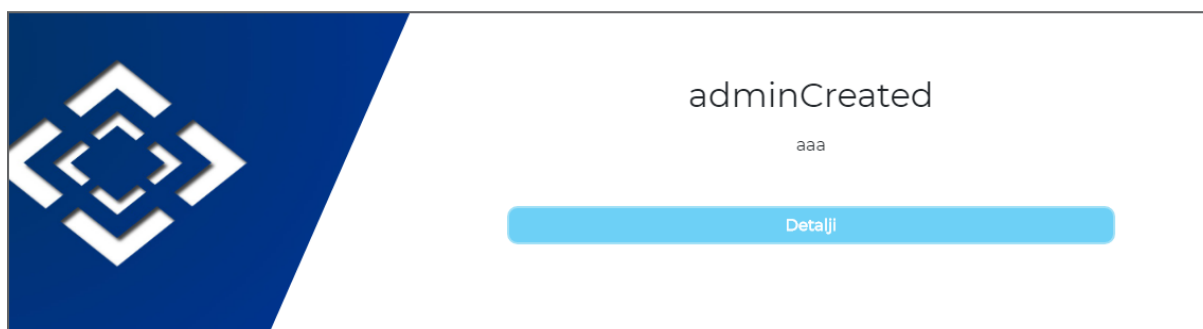
## Logo

Logo može biti samo jedan tako pri svakom novom dodavanju se onaj stari izbriše.

Logo poslodavca se prikazuje na karticama poslodavca, karticama projekata tog poslodavca te gumbu poslodavca koji se nalazi na početnoj stranici.



Slika 5 Kartica poslodavca - zadana slika



Slika 6 Kartica poslodavca - logo



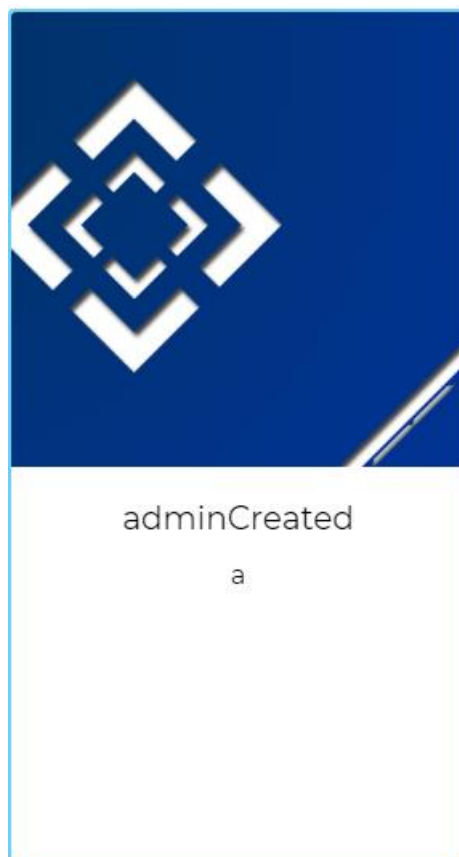
Slika 7 Gumb poslodavca - zadana slika



Slika 8 Gumb poslodavca - logo



Slika 9 Kartica projekta - zadana slika



Slika 10 Kartica projekta - logo

### **Dodavanje slike**

Za dodavanje slike, to jest za *drag and drop* služi Filepond<sup>5</sup>komponenta te nekoliko plugin-a za Filepond. Najvažniji plugin ovdje je FileEncode pomoću kojega se slika pretvara u base64url.

Nakon što korisnik odabere i ubaci željenu sliku u Filepond, klikom na 'Dodaj sliku', ta slika se obrađuje i pohranjuje na Firestorage.

---

<sup>5</sup> Filepond i Filepond plugins izvor: <https://pqina.nl/filepond/docs/>

```

async uploadFile(){
  const file = this.$refs.pond.getFiles()[0]
  if(!file) return;

  let fileName = '';
  if(this.isActive == 'logo'){
    if(this.user_data.id) fileName = this.isActive + '_' + this.user_data.id + '.png';
    else fileName = this.isActive + '_' + this.user_data._id + '.png';
  }
  else fileName = this.isActive + '_' + Date.now() + '.png';

  console.log(fileName)

  const result = await firebase.storage().ref(fileName).putString(file.getFileEncodeDataURL(), 'data_url');
  const imgUrl = await result.ref.getDownloadURL();

  this.updateUser({ name: fileName, imgUrl: imgUrl});
},

```

*Funkcija 11 Src/components/gallery\_editor.vue - Dodavanje nove slike u Firestorage*

Prva stvar koja se dešava u ovoj funkciji jest da se provjerava ako je korisnik predao sliku, ako nije funkcija se prekida jer nema razloga za nastavak.

Ovisno o tome ako je korisnik šalje logo ili header naziv slike je drugačiji, ako je to logo, tada se sprema id korisnika kao naziv slike a ako je header sprema se trenutni unix time. Razlog tome jest to da ako dodamo sliku s istim nazivom kao ona već pohranjena, nova slika će prebrisati staru. Dok header slika može biti više i zbog toga se ona prva ne smije prebrisati.

Nakon što se slika pohrani na Firestorage, dohvaća se poveznica na tu sliku i pohranjuje u objekt zajedno sa imenom slike. Taj objekt se tada šalje na backend i pohranjuje u odgovarajući atribut korisnika. Nova header slika se dodaje u polje *headers* dok se novi logo pohranjuje u atribut *logo*

## Odabir projekata

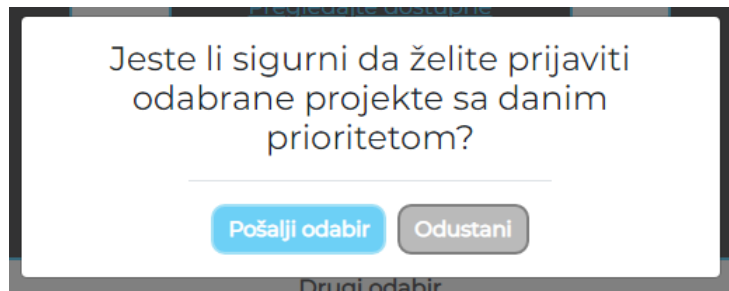
Student odabire 3 projekta koja zatim šalje na odobravanje. Da bi se projekt odabrao, nekoliko uvijete mora biti ispunjeno:

- Korisnik mora biti student
- Mora imati izabrano manje od 3 projekta
- Projekt mora imati slobodnih mjesta

Kada korisnik odabere projekt taj projekt se sprema u listu projekata u localStorage. Razlog za korištenje localStorage-a je da prilikom osvježavanja stranice, projekti ostaju pohranjeni te se ne izbrišu kao ostale informacije koje se spremaju u varijablu *project\_list* u *src/store.js* gdje se izbrišu svaki put kada se stranica osvježi.

Kada korisnik odabere sva 3 projekta, njihov prioritet će biti određen na način kako ih je korisnik odabirao, ali student može mijenjati prioritete. Nakon što student postavi sve projekte po željenim prioritetima šalje ih administratoru i te tada čega alokaciju na jedan od tih projekata.

Student klikom na gumb 'Pošalji svoj odabir' potvrđuje odabir svojih projekata ali mu se još jednom otvara modal i traži ga da ponovno potvrdi svoj odabir.



Slika 11 Modal za potvrdu slanja odabranih projekata

```
async send_project_selection(){
  await Projects.submit_projects([this.first_choice.id, this.second_choice.id, this.third_choice.id]);
  this.updateCurrentUser();
},
updateCurrentUser(){
  let user_data = this.auth.user_data;
  user_data.chosenProjects = [this.first_choice.id, this.second_choice.id, this.third_choice.id];

  localStorage.setItem('user', JSON.stringify(user_data));
  this.selectionConfirmed = true;
},
```

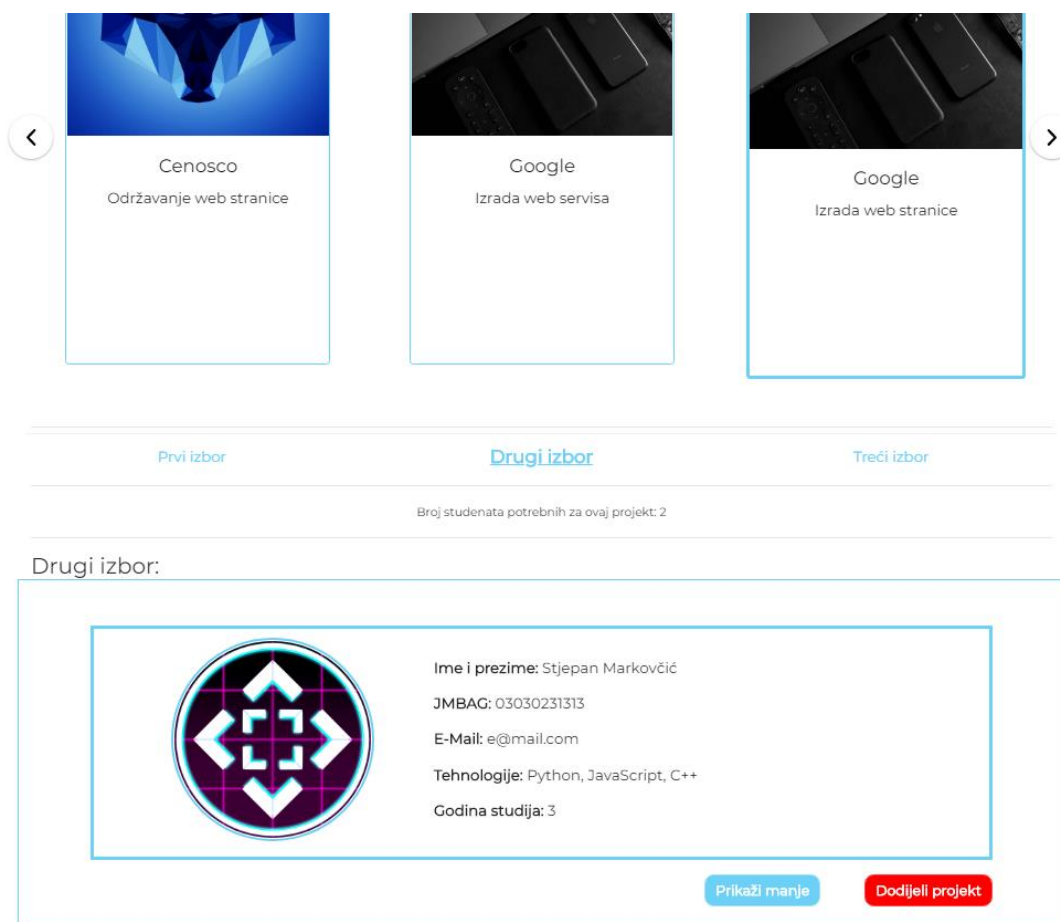
Funkcija 12 *src/views/SelectedProjects.vue* - Slanje odabranih projekta

Klikom na 'Pošalji odabir', pokreće se funkcija koja šalje id svakog od tih projekata u listu te se nakon toga ta lista pohranjuje i lokalno u localStorage, to jest u korisničke podatke studenta.

Kada se student ponovno prijavi u sustav, poziva se ruta koja pronalazi i dohvaća tu odabrane projekte.

### Alokacija studenata

Nakon što su studenti odabrali projekte, administrator može te projekte dodijeliti jednom od tih studenata. Alokacija se vrši tako da administrator na rutu *AllocateStudent* odabere jedan projekt za koji će mu se zatim prikazati gumb za prvi, drugi i treći prioritet. Klikom na jedan od ta 3 gumba, administrator vidi studente koji su projekte izabrali pod tim prioritetom. Te ima tada opcije za dodjeljivanje tog projekta jednom od studentu, ili može otvoriti više informacija o tom studentu.



Slika 12 Alokacija studenta na projekt



Slika 13 Modal za potvrdu dodjele projekta

Nakon što administrator potvrdi dodjele projekta pokreće se funkcija `assignProject()` koja najprije vrši lokalno ažuriranje projekta te se taj projekt šalje na backend za ažuriranje.

```
// Nakon potvrđivanja alokacije studenta, vrši se update projekta
assignProject(){
  const project = this.updateLocalProjects();
  Projects.UpdateProject(project, project.id, true);
  this.getProjects();

  this.selectedProject = this.selectedStudent = false;
},

// Dodajemo id alociranog studenta na prvi 'false' u atributu 'allocated_to' odabranog projekta
// Ponovno pozivamo funkciju za dohvaćanje liste projekata, ako smo popuili sva slobodna mjesta na projektu, neće se više pojavljivati
// Vracamo projekt te se on updatea na bazi
updateLocalProjects(){
  const project_index = this.store.project_list.findIndex(project => project.id == this.selectedProject);
  const project = this.store.project_list[project_index]

  project.allocated_to[project.allocated_to.indexOf(false)] = this.selectedStudent;

  this.store.project_list[project_index] = project;

  return project
},
```

Funkcija 13 `src/views/allocateStudents.vue` - Ažuriranje projekta tj. alokacija studenta na projekte

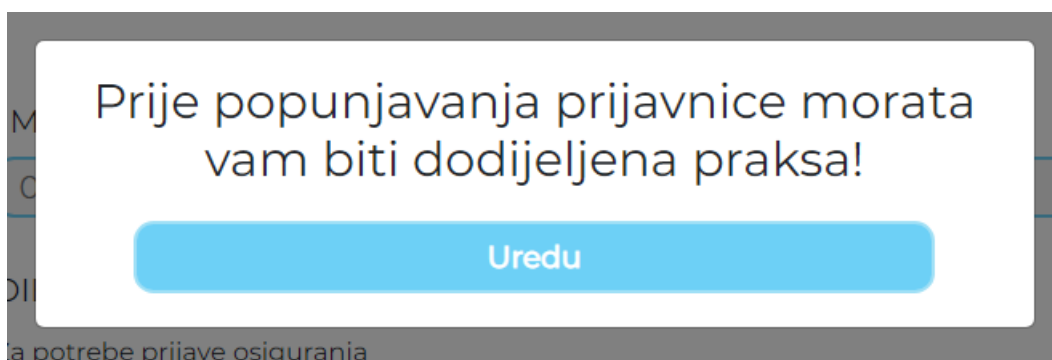
Lokalno ažuriranje funkcionira na način da se pronađe indeks odabranog projekta te se dohvati objekt sa tim indeksom. Dohvaćamo atribut tog projekta `allocated_to` koji je lista `false` elemenata. Prvi `false` element u toj listi se zamjenjuje sa id odabranog studenta te se taj ažuriran objekt pohranjuje nazad u listu na njegov indeks.

Nakon što se projekt lokalno ažurira, šaljemo ga na backend putem rute za ažuriranje projekata. Nakon toga dohvaćamo sve projekte koji još uvijek imaju slobodnih mjesta za studente.

## Popunjavanje prijavnice

Kada je studentu dodijeljena praksa, on tada treba stupiti u kontakt sa poslodavcem pomoću jednog od navedenih kontakata u projekte ili na profilu poslodavca.

Nakon što student dogovori detalje prakse sa poslodavcem. Ako student pokuša popuniti prijavnicu prije nego što mu je odobrena praksa, tada se pojavljuje modal koji ga obavještava da mora pričekati na alokaciju projekta od strane administratora. Ista stvar će se dogoditi i kada student pokuša predati dnevnik prakse a nije predao prijavnicu.



Slika 14 Modal za obavještavanje studenta da ne može ispuniti prijavnicu

Prilikom popunjavanja prijavnice, neki podaci će se sami popuniti na temelju onih koji su pohranjeni u projektu te u podacima studenta(npr. ime i prezime studenta).

```
// Popunjena prijavnica se pohranjuje u bazu
async send_application(){
  this.convert_date();
  const result = await App.upload_application_form(this.application_form, this.user_data._id);

  this.user_data.application = this.application_form;
  localStorage.setItem('user', JSON.stringify(this.user_data));

  this.$router.push({ name: 'Home' })
},

// Datume pretvaramo u unix time
convert_date(){
  this.application_form.start_date = Date.parse(this.start_date);
  this.application_form.end_date = Date.parse(this.end_date)
},
```

Funkcija 14 src/views/FillApplicationForm.vue - Slanje prijavnice na backend i lokalno ažuriranje



Prije pohranjivanja prijavnice, odabrani datum se pretvara u unix time i pohranjuje u prijavnici. Prijavnica se zatim šalje na backend te se vrši ažuriranje lokalnih podataka studenta.

### **Predaja dnevnika prakse**

Kada je student gotov sa izvršavanje prakse, popunjeni dnevnik prakse predaje u sustav gdje ga zatim administrator može preuzeti.

Predložak dnevnika prakse, administrator može postaviti u sustav tako da ga studenti mogu preuzeti i iskoristiti za izradu svog dnevnika prakse. Ukoliko predložak dnevnika ne postoji a student ga pokuša preuzeti, ispisati će mu se odgovarajuća poruka da dnevnik prakse nije dostupan u tom trenutku.

Isto kao i kod popunjavanja prijavnice, ako student pokuša predati dnevnik prije nego što je popunio prijavnicu iskače mu modal koji ga obavještava da predaja dnevnika trenutno nije moguća.

Student nije ograničen na jednu predaju dnevnika, već kada jednom preda dnevnik prakse, ima mogućnost predati ga ponovno u slučaju greške.

Isto kao i za slike, za predaju dnevnika koristi se Filepond. Nakon što student priloži dnevnik te pritisne na gumb za predaju dnevnik pokreće se funkcija za obradu i pohranu dnevnika. Isti proces vrijedi i za administratora prilikom dodavanja predloška dnevnika.

```

createFile(){
  const file = this.$refs.pond.getFile();
  if(!file) return;
  if(this.user_type == "Student"){
    this.checkAccess();
    return;
  }

  this.UploadFile({
    fileName: file.filename,
    fileData: file.getFileEncodeDataURL()
  })
},

async UploadFile(newFile){
  let uploadFile = {
    get Admin(){ return App.upload_template(newFile)},
    get Student(){ return App.upload_journal(newFile)}
  }
  const result = await uploadFile[this.user_type];

  if(result){
    this.response_message = "Dnevnik Prakse uspješno predan";
    $('#response_message').modal({
      backdrop: 'static',
      keyboard: false,
    })
  }
},

```

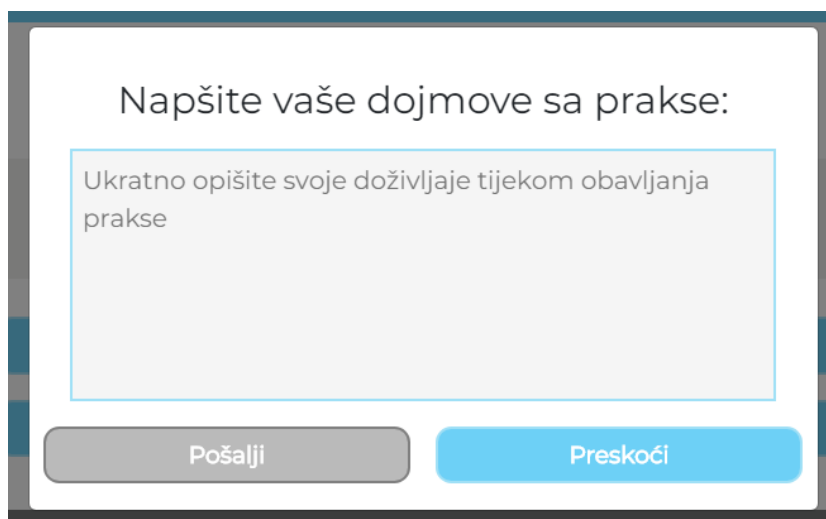
*Funkcija 15 src/views/journal.vue - Stvaranje dokumenta te slanje na backend*

Nakon što se pokrene funkcija za slanje dnevnika na bazu, provjerava se ako je korisnik predao datoteku, ako nije funkcija se ne izvršava dalje. Također provjerava se ako je student predao prijavnicu, ukoliko nije pojavljuje mu se modal sa odgovarajućom porukom.

Dokument se sastoji od imena dnevnika prakse te predanog dnevnika u base64URL obliku. Dalje se poziva funkcija za slanje dnevnika na backend. Ovisno o tome ako je trenutni korisnik student ili administrator poziva se druga funkcija za slanje na backend. Ako je dnevnik uspješno dodan, korisniku se prikazuje modal sa porukom te korisnik ima mogućnost ostaviti svoje dojmove o praski.

## Dojmovi/recenzije

Nakon što student preda dnevnik prakse, pojavljuje se modal koji studentu daje mu se mogućnost da napiše svoje dojmove o praksi koje će zatim administrator i ostali studenti moći vidjeti i pročitati. Student ima također mogućnost da ne napiše dnevnik prakse već da samo to preskoči.



Slika 15 Modal za pisanje dojмова studenta o praksi

```
async sendReview(){
  let user = Auth.state.user_data;
  user.review = this.review;

  const result = await App.updateUser(user, true);
  console.log(result);

  this.state = 'sent';
}
```

Funkcija 16 src/components/review\_form.vue - pohrana dojмова studenta

Kada student napiše svoje dojmove i preda ih, poziva se funkcija `sendReview()`. Funkcija dohvaća podatke trenutnog korisnika, to jest studenta te mu dodaje atribut `review` u koji se pohranjuje napisani dojam studenta. Ti podaci se zatim šalju na backend i ažuriraju.

Administrator i studenti mogu pročitati dojmove u tablici studenata.

## Tablica studenata

Tablici studenata mogu pristupati administrator i studenti. Ideja je da administratoru posluži kao lakši pregled studenata dok studentima služi za transparentnost. Po transparentnošću se misli da studenti mogu vidjeti da je neki projekt dodijeljen drugom studentu a ne da su oni samo zakinuti njime. Također služi da bi studente potaknulo na izvršavanje svojih obaveza kao npr. odabir projekata, dogovaranje sa poslodavcem i slično.

U tablici studenata također studenti i administrator mogu pročitati dojmove studenata nakon završetka njihove studentske prakse. Ti dojmovi mogu ostalim studentima dati uvid u to poduzeće a administratoru mogu dati uvid kako poboljšati studentsku praksu.

Tablica studenata

JMBAG	Godina studija	Praksa	Status	Prijavnica	Dnevnik Prakse	Dojmovi
03030231313	3	-	Nedefinirana	Nije predano	Nije predano	Pročitaj
34252	3	-	Nedefinirana	Nije predano	Nije predano	Pročitaj
21413	2	-	Nedefinirana	Nije predano	Nije predano	Pročitaj
2131233232324	2	-	Nedefinirana	Nije predano	Nije predano	Pročitaj
3243	2	-	Nedefinirana	Nije predano	Nije predano	Pročitaj
123	3	-	Nedefinirana	Nije predano	Nije predano	Pročitaj

Slika 16 Tablica studenata - Kako student vidi

Tablica studenata

Ime	Prezime	JMBAG	Godina studija	Praksa	Status	Prijavnica
Stjepan	Marković	03030231313	3	-	Nedefinirana	Pregled
Miro	Gavran	34252	3	-	Nedefinirana	Pregled
Probn	student	21413	2	-	Nedefinirana	Pregled
f	o	2131233232324	2	-	Nedefinirana	Pregled
student	student	3243	2	-	Nedefinirana	Pregled
test	test	123	3	-	Nedefinirana	Pregled

Slika 17 Tablica studenata - Kako administrator vidi

Tablica studenta sadrži polja za:

- Ime – Samo administrator
- Prezime – Samo administrator
- JMBAG
- Godina studija
- Praksa – Naziv poslodavca
- Status – Obavljena/ Dogovorena/ Dodijeljen projekt/ Odabrani projekt/ Nedefinirana
- Prijavnica – Studenti vide ako je predano, administrator ju može pregledati
- Dnevnik prakse – Studenti vide ako je predano, administrator ga može preuzeti
- Dojmovi

### **Postavljanje uputa**

Upute služe studentima da bi mogli znati koje korake treba slijediti za izvršavanje studentske prakse.

Administrator ima mogućnost dodavanja, brisanja te uređivanja uputa. To radi na način da omogući način za uređivanje klikom na gumb 'Uredi'.

Svaka uputa ima svoj redoslijed koji joj je dodijeljen kada je ona stvorena. Kada je jedna instrukcija obriše, redoslijed ostalih se ažurira.

### Upute za izvršavanje studentske prakse:

- 1 Pogledati zanimljive slobodne projekte, ukoliko želite specifično poduzeće koje nije na listi, javite im se da popune zadatak
- 2 Prijaviti se na 3 najdraža projekta
- 3 Pričekati da vam se dodijeli tvrtka
- 4 Ako čekate predugo na alokaciju, javite se (npr. više od 4-5 dana)...
- 5 Kontaktirati mentora sa zadatka na koji ste alocirani. Predstaviti se i reći da ste dobili zadatak, eventualno obaviti intervju ako to traže.
- 6 Dogovoriti početak izvršavanja prakse.
- 7 Kada sve finalno dogovorite s mentorom popunite Prijavnicu
- 8 Na mail ćete dobiti (i mentor) praznu Potvrdu o obavljenoj praksi, mora ju ispuniti mentor.
- 9 Popuniti i predati dnevnik prakse
- 10 Prijaviti ispit, ocjena se unosi automatski, ne morate dolaziti.

*Slika 18 Pregled popisa uputa*

## Upute za izvršavanje studentske prakse:

Pohrani promjene

Odustani

Upišite novu uputu!

Dodaj ✓

Pogledati zanimljive slobodne projekte, ukoliko želite specifično poduzeće koje nije na listi, javite im se da popune zadatak

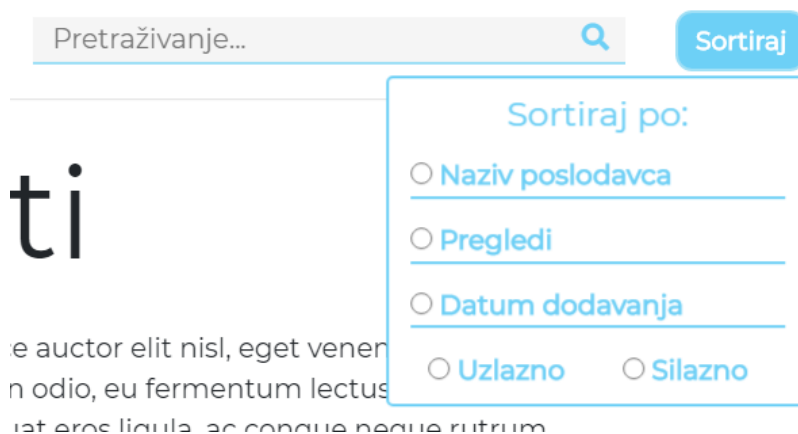
Ukloni ✕

Slika 19 Način za uređivanje uputa

### Sortiranje

Svaki korisnik može sortirati projekte i poslodavce dok administrator ima mogućnost još sortirati i studente.

Sortiranje se vrši na listi podataka koja se pohranjuje u `src/store.js` te dokumenti unutar tih lista sortiraju po jednom od njegovih atributa. Također, lista se može sortirati uzlazno i silazno po danom atributu.



Slika 20 Sortiranje projekata po danim atributima

Atributi se odabiru kroz radio gumbе tako da u jednom trenutku može samo jedan atribut biti odabran. Isto to vrijedi za opcije uzlaznog i silaznog sortiranja.

```
sortItems(sort_order){  
  if(!this.sort_values) return;  
  this.store.sort_items(this.sort_values, sort_order, "project_list");  
  this.getProjects();  
},
```

*Funkcija 17 src/views/Projects.vue - Pozivanje funkcije za sortiranje sa parametrima za sortiranje*

Vrijednosti koje su potrebne da bi sortiranje pravilno funkcioniralo jesu:

- *sort\_values* – objekt koji određuje atribut za sortiranje i tip podatka za sortiranje(npr. string)
  - *sort\_values* = {atr: 'company', type: 'string'}
- *sort\_order* – ako se sortira uzlazno ili silazno
  - *sort\_order* = 'asc'
- naziv liste za sortiranje – eksplicitno zadana, ovisi o kojoj ruti se radi
  - 'project\_list'

Odabirom vrste sortiranja, to jest uzlazno ili silazno ova funkcija se poziva, ako nije odabran atribut za sortiranje tada se funkcija neće izvršiti. U suprotnome poziva se funkcija *sort\_items()* iz *src/store.js* gdje se nalazi da se izbjegne redundancija koda, pošto se ista funkcija poziva sa tri rute.

```

sort_items(sort_values, sort_order, list){
  let sorter = {
    items: false, // Polje elemenata koje sortiramo
    atr: false, // Atribut po kojem sortiramo polje

    get asc_number(){
      return this.number_sort()
    },
    get desc_number(){
      return this.number_sort().reverse();
    },
    get asc_string(){
      return this.string_sort()
    },
    get desc_string(){
      return this.string_sort().reverse()
    },
    get asc_bool(){
      return this.bool_sort()
    },
    get desc_bool(){
      return this.bool_sort().reverse()
    },
    // Funkcije za sortiranje su izdvojene da se ne ponavljaju
    string_sort(){
      return this.items.sort((a, b) => {
        const first_item = a[this.atr].toUpperCase();
        const second_item = b[this.atr].toUpperCase();

        if (first_item < second_item) return -1;
        if (first_item > second_item) return 1;

        return 0;
      })
    },
    number_sort(){
      return this.items.sort((a, b) => { return a[this.atr] - b[this.atr] })
    },
    bool_sort(){
      return this.items.sort((a, b) => { return (a[this.atr] === b[this.atr])? 0 : a[this.atr]? -1 : 1 })
    }
  }

  sorter.atr = sort_values.atr;
  sorter.items = this[list];

  this[list] = sorter[sort_order + "_" + sort_values.type]
},

```

*Funkcija 18 src/store.js*

Funkcija sortira na način da atribut za sortiranje te listu koja će se sortirati pohrani u objekt *sorter*. Nakon što se te vrijednosti pohrane u objekt, dohvaća se sortirana lista pozivom *get* funkcije unutar objekta.

Funkcija koju pozivamo ovisi o parametrima koje smo proslijedili, točnije o *sort\_order* te tipu podatka unutar *sort\_values*. Tip podatka je bitan jer za različiti tipovi podataka sortiraju na različite načine.



## Pretraživanje

Sortiranje se vrši pomoću funkcije *debounce()* iz paketa *lodash*<sup>6</sup>. Debounce služi za odgodu izvršavanja funkcije za pretraživanje koja šalje upit na backend.

```
async searchProjects(search){
  const result = await Projects.getProjects(search);

  this.total_pages = Math.ceil(result.length / this.items_per_page);

  this.store.project_list = result;
  this.project_list = this.store.project_list.slice(0, this.items_per_page);
},
```

Funkcija 19 src/views/Projects.vue

Funkcija dohvaća sa backenda projekte koji po nekom atributu odgovaraju vrijednosti u parametru *search*. Iz tih rezultata dobivamo broj stranica te prikazujemo onaj broj dokumenata koliko je definirano u *items\_per\_page*.

## Korištene komponente i paketi

- Bootstrap
  - Responzivnost i izgled web aplikacije
  - Izvor: <https://getbootstrap.com/>
- FontAwesome
  - Ikone na web aplikaciji
  - Izvor: <https://fontawesome.com/>
- Vue-Filepond
  - *Drag and drop* funkcija za dodavanje datoteka u sustav
  - Izvor: <https://github.com/fuxingloh/vue-horizontal-list>
- Filepond-Encode
  - Plugin za Filepond, omogućuje šifriranje datoteke u base64Url
  - Izvor: <https://github.com/pqina/filepond-plugin-file-encode>
- Filepond-Validate-size
  - Plugin za Filepond, omogućuje provjeru veličine predane datoteke
  - Izvor: <https://github.com/pqina/filepond-plugin-file-validate-size>

---

<sup>6</sup> Lodash izvor: <https://www.npmjs.com/package/lodash>

- Filepond-Image-crop
  - Plugin za Filepond, omogućuje obrezivanje slika u željenoj rezoluciji
  - Izvor: <https://github.com/pqina/filepond-plugin-image-crop>
- Filepond-Image-preview
  - Plugin za Filepond, omogućuje prikaz slika nakon dodavanja
  - Izvor: <https://github.com/pqina/filepond-plugin-image-preview>
- Filepond-Image-transform
  - Plugin za Filepond omogućuje primjenu obrezivanja Image crop plugina
  - Izvor: <https://github.com/pqina/filepond-plugin-image-transform>
- Vue-horizontal-list
  - Prikaz projekata u horizontalnoj listi
  - Izvor: <https://github.com/fuxingloh/vue-horizontal-list>
- Lodash
  - *debounce()* funkcija za odgodu izvršavanja funkcija
  - Izvor: <https://lodash.com/>
- Vue-flux
  - Prikaz slika kao slideshow
  - Izvor: <https://github.com/deulos/vue-flux>
- Vuejs-paginate
  - Komponenta za paginaciju
  - Izvor: <https://github.com/lokyoung/vuejs-paginate>
- Firestore/Firestorage
  - Pohrana slika
  - Izvor: <https://firebase.google.com/products/firestore>