

UNIVERZITET U BEOGRADU  
ELEKTROTEHNIČKI FAKULTET



DIPLOMSKI RAD

**REALIZACIJA JAVNIH POZIVA  
ZA NAUČNE PROJEKTE**

Mentor:

**Prof. dr Boško Nikolić**

Kandidat:

**Ana Vanzo 2015/0029**

Beograd, Septembar 2021.

*Posvećeno mojoj mami.*

# Sadržaj

UVOD .....	4
ZAHTEVI ZA REALIZACIJOM SISTEMA .....	6
2.1. KORISNIČKI ZAHTEVI .....	6
2.2. OPIS KORIŠĆENIH TEHNOLOGIJA .....	9
2.2.1. ANGULAR .....	9
2.2.2. ANGULAR MATERIAL .....	9
2.2.3. HTML .....	9
2.2.4. CSS .....	9
2.2.5. TYPESCRIPT .....	10
2.2.6. BOOTSTRAP .....	10
2.2.7. NODE JS .....	10
2.2.8. EXPRESS .....	11
2.2.9. MONGOOSE .....	11
2.2.10. MongoDB .....	12
OPIS RADA SISTEMA .....	13
3.1. POČETNA STRANICA .....	13
3.1.1. PRIJAVA .....	13
3.1.2. REGISTRACIJA .....	14
3.2. KORISNIK .....	17
3.2.1. PREGLED JAVNIH POZIVA .....	17
3.2.2. POSLATE PRIJAVE .....	21
3.2.3. NACRTI .....	22
3.2.4. OBAVEŠTENJA .....	23
3.3. ADMINISTRATOR .....	24
3.3.1. OTVARANJE POZIVA .....	24
3.3.2. PREGLED PRIJAVA .....	25
3.3.3. REGISTRACIJA KORISNIKA .....	26
3.4. ZAJEDNIČKE FUNKCIONALNOSTI .....	27
3.4.1. IZVEŠTAJI .....	27
3.4.2. NALOG .....	29
REALIZACIJA SISTEMA .....	31
ZAKLJUČAK .....	43
LITERATURA .....	44

# UVOD

U ovom dokumentu opisana je detaljno internet aplikacija koja realizuje javne pozive za naučne projekte, koja je ujedno bila i tema mog diplomskog rada. Javni poziv za projekte predstavlja način kojim jedna institucija oglašava potrebu za prikupljanjem ponuda za projekte od strane ostalih pravnih lica u cilju realizacije neke ideje, rešenja nekog problema. Javne pozive često vidamo na sajtovima raznih državnih institucija ili u novinama i upravo istraživajući ovu temu i sama sam bila čest posetilac tih sajtova i primetila sam da većina javnih poziva, pored osnovnih informacija o samom pozivu, sadrži i obrazac u vidu Word dokumenta koji treba popuniti kao prijavu za poziv. Prijava na veliki broj javnih poziva izgleda tako što podnosilac preuzme obrazac, često i više njih, sa sajta institucije, zatim ga popuni, odštampa i šalje dalje ili odnosi na ukazanu adresu. Ovakav način prijavljivanja je po mom mišljenju jako neefikasan i verujem iritantan kako za podnosioca prijava, tako i za one koji vrše selekciju prijava, jer se zatrpavaju nepotrebnom papirologijom i gomilom obrazaca i dokumenata koje treba dostaviti ili pregledati, dok istovremeno u vazduhu lebdi mogućnost da ti “zafali jedan papir”. U ovoj aplikaciji javni pozivi za, konkretno, naučne projekte su realizovani upravo tako da se takve situacije izbegnu, da slanje prijava bude efikasnije i za podnosioca prijava i za njihove selektore. Kako je to postignuto? U sistemu postoje dva tipa korisnika administrator i običan korisnik. Administrator je taj koji ima mogućnost da otvori javni poziv i da definiše neophodne podatke i dokumenta koja treba priložiti uz prijavu. Korisnik nakon prijave na sistem ima pregled svih javnih poziva i mogućnost da se na neki od njih prijavi. Prilikom prijave korisniku se prikazuje forma sa traženim podacima. Korisnik je popunjava i prilaže sve zahtevane dokumente direktno u aplikaciji, zatim ima mogućnost da popunjenu prijavu i preuzme u pdf formatu, da je pošalje ili snimi i sačuva za kasnije kada joj može ponovo pristupiti iz *Nacrta*. Korisnik takođe ima pristup svim svojim poslatim prijavama uz uvid u njihov trenutni status (prihvaćeno, odbijeno, na čekanju). Administratoru je omogućen pregled svih prijava po pozivu, nakon pregleda prijava može obaviti selekciju tako što će određene prijave prihvatiti ili odbiti. Ono što je još realizovano u aplikaciji a takođe je značajno jeste klasifikacija poziva po korisničkim grupama. Naime, svaki poziv može imati ciljnu grupu korisnika koju targetira, time se značajno sužava skup selekcije i omogućava da poziv dođe u prave ruke. Pri registraciji korisnik bira kojoj od ponuđenih grupa pripada i njemu će se nadalje

prikazivati isključivo pozivi namenjeni njegovoj grupi i oni pozivi koji su označeni kao vidljivi svima. U prvom delu ovog dokumenta u okviru poglavlja **Zahtevi za realizacijom sistema** opisani su korisnički zahtevi i korišćene tehnologije. **Korisnički zahtevi** sadrže opis funkcionalnosti sistema koje je trebalo realizovati, dok su u okviru **Korišćenih tehnologija** opisane sve tehnologije korišćene pri realizaciji sistema. Sledeće poglavlje **Opis rada sistema** sadrži detaljno korisničko uputstvo koje prezentuje način rada sistema sa propratnim slikama i opisuje sve realizovane funkcionalnosti s ciljem da se olakša korisnicima korišćenje ove aplikacije. U trećem poglavlju pod nazivom **Realizacija sistema** navedeni su najizazovniji problemi sa kojima sam se susretala prilikom realizacije sistema, kao i njihova rešenja uz propratne slike izvornog koda. **Zaključak** predstavlja sažetu rekapitulaciju samog rada, dokumenta i razmatra moguća poboljšanja sistema. U delu **Literatura** navedene su knjige i linkovi ka sajtovima koje sam koristila tokom izrade sistema.

# **ZAHTEVI ZA REALIZACIJOM SISTEMA**

## **2.1. KORISNIČKI ZAHTEVI**

Postoje dva tipa korisnika u sistemu administrator i običan korisnik. Zahtevi za administratora su:

- otvaranje javnog poziva
- registracija korisnika
- pregled prijava po pozivu
- pregled izveštaja

Otvaranje javnog poziva podrazumeva mogućnost definisanja osnovnih informacija o javnom pozivu, kao što su naziv, naučno polje, naziv institucije, datum objave, rok prijave, tekst javnog poziva. Zatim nakon definisanja osnovnih informacija administratoru treba omogućiti definisanje podataka i dokumenata neophodnih za prijavu.

Administratoru treba omogućiti da definiše koji će se podaci tražiti pri registraciji korisnika, korisničke grupe koje postoje u sistemu.

Za svaki otvoreni javni poziv administrator treba da ima uvid u sve poslate prijave, mogućnost da obavi selekciju poslatih prijava.

Izveštaji predstavljaju uvid u statistiku prijava na pozive. Izveštaji treba da prikažu broj prijava po pozivu, po naučnom polju, po instituciji...

Zahtevi za običnog korisnika su:

- pregled javnih poziva
- prijava na odabrani javni poziv
- pregled poslatih prijava
- pristup nacrtima
- pregled izveštaja

Kod pregleda javnih poziva treba obezbediti da se korisniku prikazuju samo oni pozivi koji su namenjeni njegovoj grupi i pozivi koji su namenjeni svim grupama korisnika. Zajedno uz pregled potrebno je omogućiti i prijavu na odabrani poziv.

Prijava na poziv treba da obuhvata formu sa definisanim podacima koje treba popuniti i dokumentima koje treba priložiti. Pre slanja prijave ona se može snimiti i tada će se sačuvati u nacrtima kao nedovršena. Nakon slanja prijave korisniku treba prikazati pregled prijave uz mogućnost preuzimanja iste u pdf formatu, slanja i odustajanja. Ukoliko se odluči da odustane od slanja prijave korisniku treba dati mogućnost da je sačuva ili zaista odustane od slanja. Nakon slanja korisnik se treba obavestiti o uspešnosti slanja.

U okviru opcije *Pregled prijava* korisniku treba prikazati sve poslate prijave i njihov trenutni status. Status podrazumeva da li je prijava prihvaćena, odbijena ili je poslata ali je i dalje na čekanju. Korisnik takođe ima mogućnost da ukloni prijavu tako da mu se u buduću ne prikazuje u okviru ove opcije.

U okviru opcije *Nacrti* se nalaze sve nedovršene prijave korisnika, koje je sačuvao nakon što je odustao od slanja ili odlučio da nastavi popunjavanje kasnije. Svaku prijavu iz nacrta može izmeniti kada se ponovo otvara forma prijave sa već do tada popunjenim podacima, ili može obrisati prijavu iz nacrta bez njenog slanja.

Korisnik treba da ima uvid u izveštaje kao i administrator.

Od dodatnih funkcionalnosti korisniku treba omogućiti pristup opciji *Obaveštenja* gde će korisniku biti prikazana sva obaveštenja o prihvatanju njegovih prijava od strane admina.

Opšti korisnički zahtevi:

- prijava na sistem
- registracija
- opšte funkcionalnosti sa nalogom

Za prijavu na sistem neophodno je uneti korisničko ime i lozinku. Ukoliko korisnik nema nalog treba omogućiti njegovu registraciju.

Opšte funkcionalnosti sa nalogom podrazumevaju:

- odjavu
- pregled osnovnih informacija o nalogu
- promenu lozinke
- deaktiviranje naloga



## 2.2. OPIS KORIŠĆENIH TEHNOLOGIJA

### 2.2.1. ANGULAR

*Angular* je besplatna *open-source* razvojna platforma i *framework* zasnovan na *TypeScript*-u koji omogućava razvoj *single-page* klijentskih aplikacija uz korišćenje *HTML*-a i *TypeScript*-a. Osnovni element strukture Angular aplikacija su komponente. Svaka komponenta sadrži svoj *css*, *html* i *ts* fajl koji definišu njen izgled i logiku. Za implementaciju funkcija koje će se koristiti iz više različitih komponenti koriste se servisi. Servisi se mogu ugraditi (*inject*) u komponente. Veoma značajan modul Angulara je *Routing* modul koji omogućava navigaciju između komponenti. Koreni modul svih Angular aplikacija je *AppModule* i ostali moduli se mogu ugrđivati u njega.

Korišćena verzija: Angular CLI 12.2.1

### 2.2.2. ANGULAR MATERIAL

*Angular Material* je korišćen kao UI framework. Ova biblioteka sadrži puno komponenti koje omogućavaju izgradnju web aplikacija bogatog dizajna. Podržana je od strane svih modernih internet pretraživača i sve komponente su *device independent* tj. podržavaju *responsive* dizajn.

### 2.2.3. HTML

*HyperText Markup Language* iliti *HTML* je opisni jezik namenjen opisu veb stranica. Predstavlja najsonovniju jedinicu građe Web-a. *HTML* dokumenti se sastoje iz dela koji opisuje karakteristike samog dokumenta (*head* tagovi), i dela koji opisuje sam sadržaj dokumenta umetnut između *body* tagova. *HTML* je podržan od strane svih internet pretraživača i trenutna verzija standarda je *HTML5*.

### 2.2.4. CSS

*Cascading Style Sheets* je jezik koji se koristi za stilizovanje HTML dokumenta. Opisuje način na koji svaki HTML element treba biti prikazan. Svaki HTML element ima atribut *style* u okviru kog se može definisati izgled tog elementa, HTML dokument se može formatirati i tako što se uveze neki eksterno definisan *css* fajl i uključi u stranicu u okviru *head* taga.

## 2.2.5. TYPESCRIPT

*TypeScript* je programski jezik koji predstavlja nadskup *JavaScripta*. *TypeScript* je *JavaScript* sa sintaksom za tipove. Omogućava veću povezanost sa editorom. Kod napisan u *TypeScript*-u se konvertuje u *JavaScript* tako da može da se izvršava na svim platformama na kojima i *JavaScript*, a to znači da je podržan od strane svih modernih veb pretraživača bez potrebe za instaliranjem bilo kakvih dodataka.

## 2.2.6. BOOTSTRAP

*Bootstrap* je besplatni *open source framework* koji se koristi za dizajn web aplikacija. Baziran je na HTML-u i CSS-u a takođe podržava *JavaScript* dodatke. *Bootstrap* je responsivnog dizajna što znači da se prilagođava svim veličinama ekrana. Kompatibilan je sa svim modernim pretraživačima. Pruža puno pomoćnih klasa koje olakšavaju i ubrzavaju izradu veb sajtova.

## 2.2.7. NODE JS

*Node.js* je asinhrono, bazirano na događajima, izvršno *JavaScript* okruženje. *Node.js* daje *JavaScriptu* serversku notu, nasuprot klijentske koju je primarno imao. Dizajniran je za izgradnju skalabilnih mrežnih aplikacija. Prilikom svake konekcije sa serverom poziva se *callback* funkcija, ali ukoliko nema posla da izvršava *Node.js* će se uspavati. *Node.js* je *single-threaded* što znači da se izvršava u jednoj niti, ta jedna nit procesira događaj za događajem. Kada npr. stigne neki HTTP zahtev, server (jedna nit) počinje da ga obrađuje u slučaju postojanja blokirajućeg IO poziva server se ne blokira nego nastavlja sa obrađivanjem sledećeg događaja ili zahteva, a po završetku blokirajućeg poziva, što takođe predstavlja događaj, poziva se *callback* funkcija. Ovaj koncept je u suprotnosti sa učestalijim konkurentnim modelom u kom su niti operativnog sistema uposlene. Upravo zbog toga korišćenjem *Node.js*-a nikad ne može doći do *dead-lock*-a. *Node.js* značajno poboljšava performanse aplikacije u poređenju sa ostalim solucijama.

## 2.2.8. EXPRESS

*Express* je *backend* frejmwork web aplikacija u okviru *Node.js* servera dizajniran za razvoj mobilnih i web aplikacija. *Express* je besplatan i *open source* softver pod MIT licencom. Smatra se standardnim serverskim frejmworkom za *Node.js*. Napisan je u *JavaScriptu*. *Express* API pruža mnoštvo HTTP metoda i *middleware*-a (funkcije koje imaju pristup *Request* i *Response* objektima i sledećoj *middleware* funkciji u request-response ciklusu aplikacije, ove funkcije mogu izvršiti bilo kakav kod, pristupiti i izmeniti *Request/Response* objekte i pozvati narednu *middleware* funkciju sa steka). *Express* aplikacije mogu koristiti neke od sledećih tipova *middleware* funkcija:

- application-level middleware
- router-level middleware
- error-handling middleware
- built-in middleware
- third party-middleware

*Express* aplikacija je u suštini serija poziva *middleware* funkcija.

## 2.2.9. MONGOOSE

*Mongoose* je ODM (Object Data Modeling) biblioteka za MongoDB i Node.js. *Mongoose* upravlja odnosima između podataka u bazi, obezbeđuje validaciju šeme i koristi se za preslikavanje objekata u kodu u njihove reprezentacije u bazi MongoDB i obratno.

## **2.2.10. MongoDB**

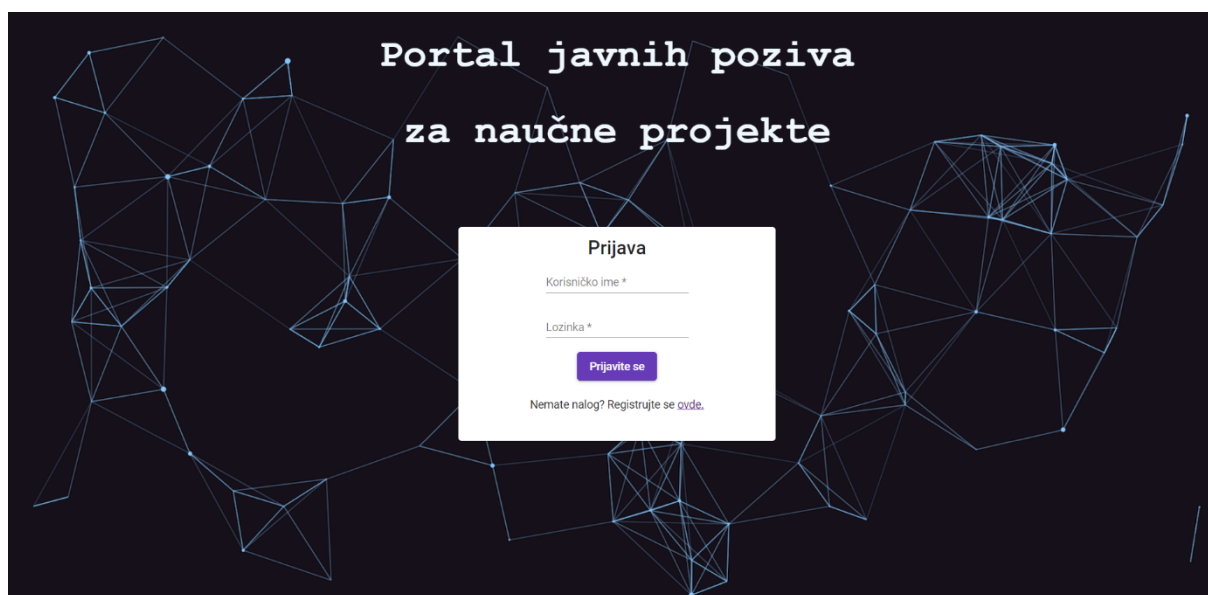
*MongoDB* je neralaciona NoSQL baza podataka. Podaci se u MongoDB bazi skladište u vidu JSON dokumenata. Struktura ovih dokumenata nije uniformna može da varira od dokumenta do dokumenta čak i u okviru iste kolekcije, za razliku od SQL baza gde svi podaci u okviru iste tabele moraju imati istu strukturu. Kolekcija u MongoDB-u je ekvivalentna tabeli u SQL bazi, a dokument je ekvivalentan redu. Polja JSON dokumenta u MongoDB-u su ekvivalentna kolonama u SQL bazi.

# OPIS RADA SISTEMA

## 3.1. POČETNA STRANICA

### 3.1.1. PRIJAVA

Komponenta *LoginComponent*, koja se nalazi u fajlu *frontend/src/app*, opisuje izgled i funkcionalnost početne stranice. Početna stranica sadrži formu za prijavu na sistem, sa poljima za unos korisničkog imena i lozinke i dugmetom za prijavu. Za dizajn je korišćena komponenta Card iz Angular Material paketa kao kontejner za formu. Na početnoj stranici takođe postoji opcija za registraciju na sistem ukoliko korisnik nije registrovan.



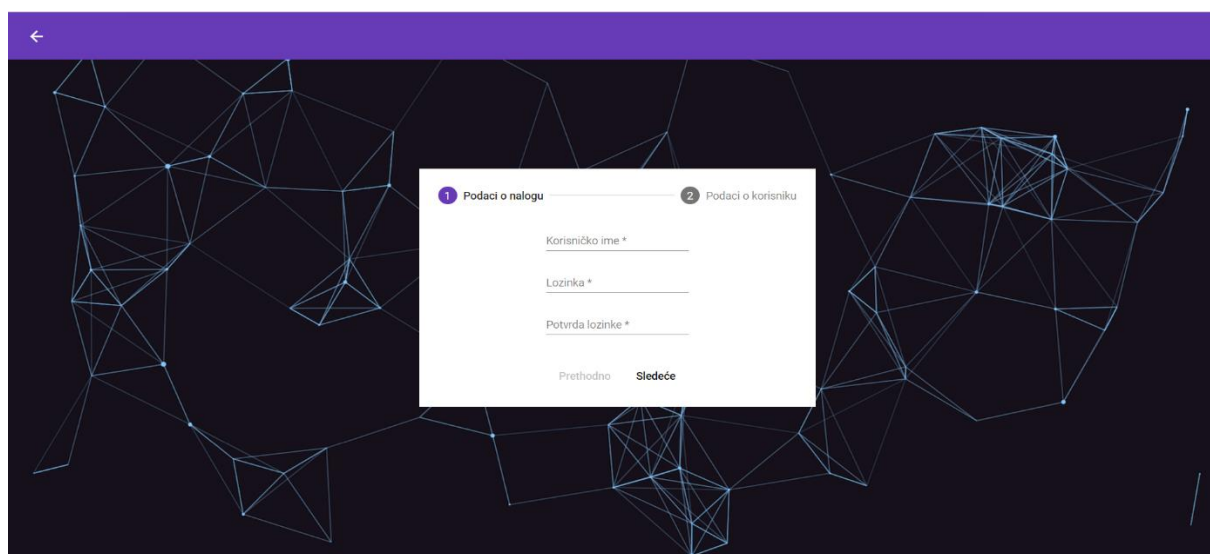
Slika 1. Početna strana - Prijava

Korisnik se prijavljuje na sistem unosom korisničkog imena i lozinke koji su obavezni. Ukoliko korisnik unese pogrešno korisničko ime ili lozinku prikazuje mu se poruka o grešci.

### 3.1.2. REGISTRACIJA

Komponenta RegisterComponent u fajlu frontend/src/app opisuje izgled i funkcionalnost strane za registraciju korisnika. Strana za registraciju sadrži *Toolbar* Material komponentu sa opcijom sa povratak na stranu za prijavljivanje predstavljenu strelicom (*Mat Icon Arrow Back*).

Na sredini stranice za registraciju nalazi se forma predstavljena komponentom *Stepper*. Prvi korak pri registraciji podrazumeva definisanje osnovnih podataka o nalogu kao što su biranje korisničkog imena, lozinke i potvrde lozinke. Ukoliko je izabrano korisničko ime već zauzeto obaveštava se korisnik. Takođe, ukoliko se lozinka i potvrda lozinke razlikuju prikazuje se poruka o grešci. Drugi korak pri registraciji podrazumeva unošenje podataka o samom korisniku koje je prethodno definisao administrator. Administrator sistema ima mogućnost da određuje koji će se podaci zahtevati pri registraciji korisnika. Administrator pri definisanju polja za registraciju bira koja će od zahtevanih polja biti obavezna. Ukoliko obavezna polja nisu popunjena prikazuje se poruka sa obaveštenjem da je polje obavezno popuniti.



Slika 2. – Izgled stranice za registraciju

←

1 Podaci o nalogu 2 Podaci o korisniku

Korisničko ime \*

Lozinka \*

Potvrda lozinke \*

Prethodno Sledeće

*Slika 3. – Podaci o nalogu*

←

1 Podaci o nalogu 2 Podaci o korisniku

Grupa korisnika

Polje "Grupa korisnika" je obavezno!

Ime i prezime

Polje "Ime i prezime" je obavezno!

Email

Naučno zvanje

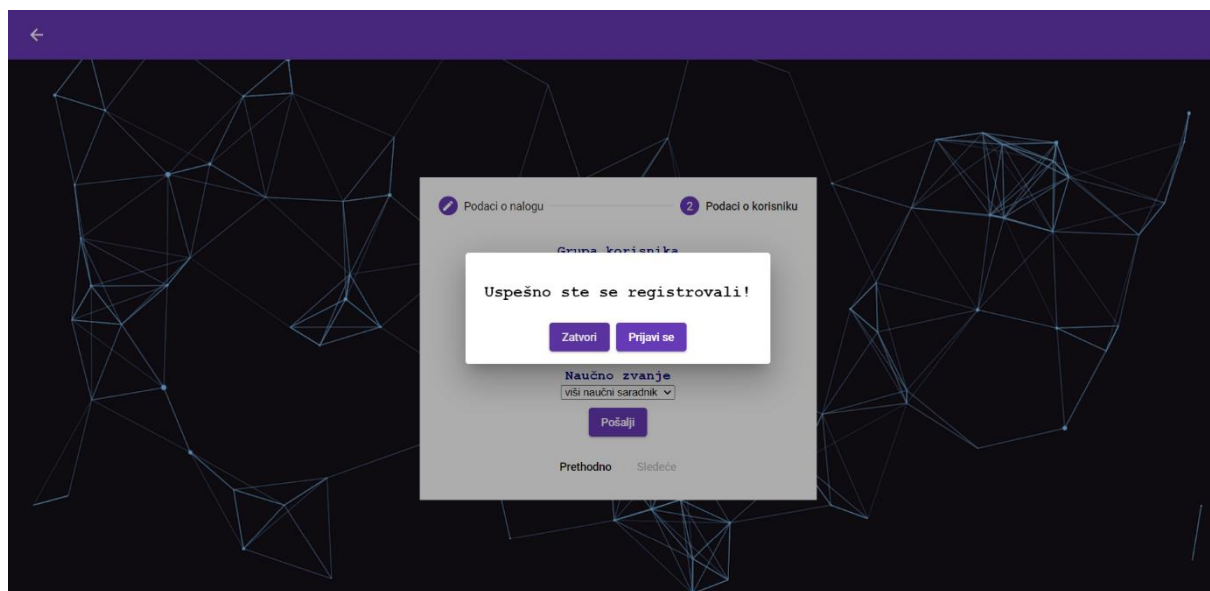
Polje "Naučno zvanje" je obavezno!

Pošalji

Prethodno Sledeće

*Slika 4. – Podaci o korisniku*

Dugme *Pošalji* postaje aktivno tek nakon uspešne validacije forme. Klikom na dugme *Pošalji* prosleđuje se zahtev za registracijom i korisniku se otvara prozor sa obaveštenjem o uspešnoj registraciji i dve opcije, da zatvori prozor ili da se vrati na stranu za prijavu. Prozor je realizovan korišćenjem komponente *Dialog* iz Angular Material paketa.



Slika 5. – Uspešna registracija



## 3.2 KORISNIK

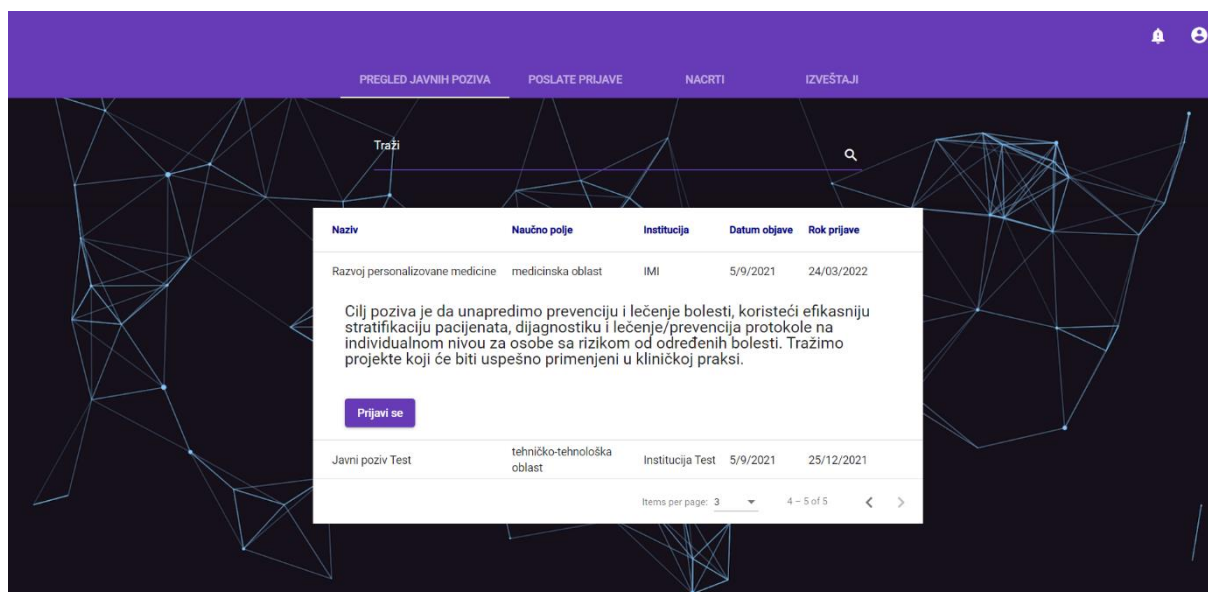
### 3.2.1. PREGLED JAVNIH POZIVA

Kada se korisnik uspešno prijavi na sistem prikazuje mu se stranica sa određenim funkcionalnostima namenjenih korisniku. U zaglavlju stranice nalazi se *Toolbar* sa dve ikonice u desnom uglu: *Account* (sadrži osnovne funkcionalnosti koje omogućavaju rad sa nalogom) i *Notifications* (sadrži obaveštenja o prijavama na javne pozive).

Ispod *Toolbar*-a nalazi se horizontalni meni sa tabovima (korišćena material komponenta *Tab*). U meniju su korisniku omogućene sledeće funkcionalnosti:

- *Pregled javnih poziva*
- *Poslate prijave*
- *Nacrti*
- *Izveštaji*

Odmah nakon prijave na sistem korisniku se prikazuju svi javni pozivi koji targetiraju grupu kojoj korisnik pripada.



Slika 6. – Pregled javnih poziva

Opcija **Pregled javnih poziva** sadrži tabelarni prikaz poziva sa informacijama o nazivu poziva, naučnom polju, instituciji, datumu objave i roku prijave. Klikom na neki red tabele ona se ekspanduje i u ekspanдованом delu se prikazuje tekst javnog poziva sa dugmetom za prijavu na poziv. Tabela ima paginator i mogućnost da se pretraži po bilo kom polju.

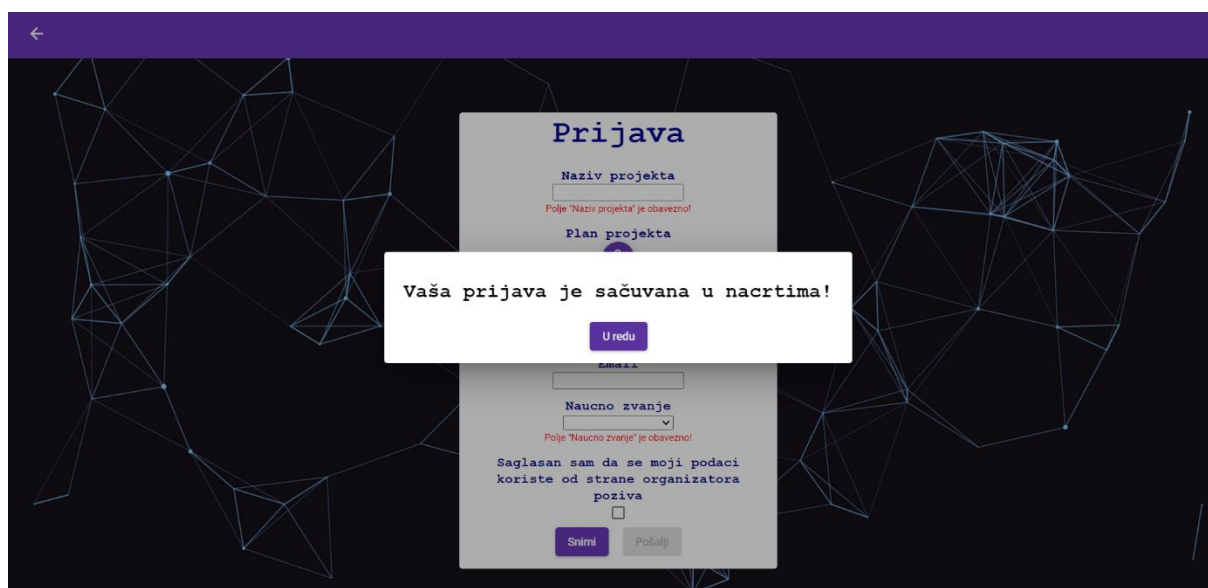
Korišćena Material komponenta je *Table* i *Paginator*. Izgled i funkcionalnost ove stranice opisuje komponenta *PlainUserComponent*.

Klikom na dugme *Prijavi se* korisniku se prikazuje stranica sa formom za prijavu.



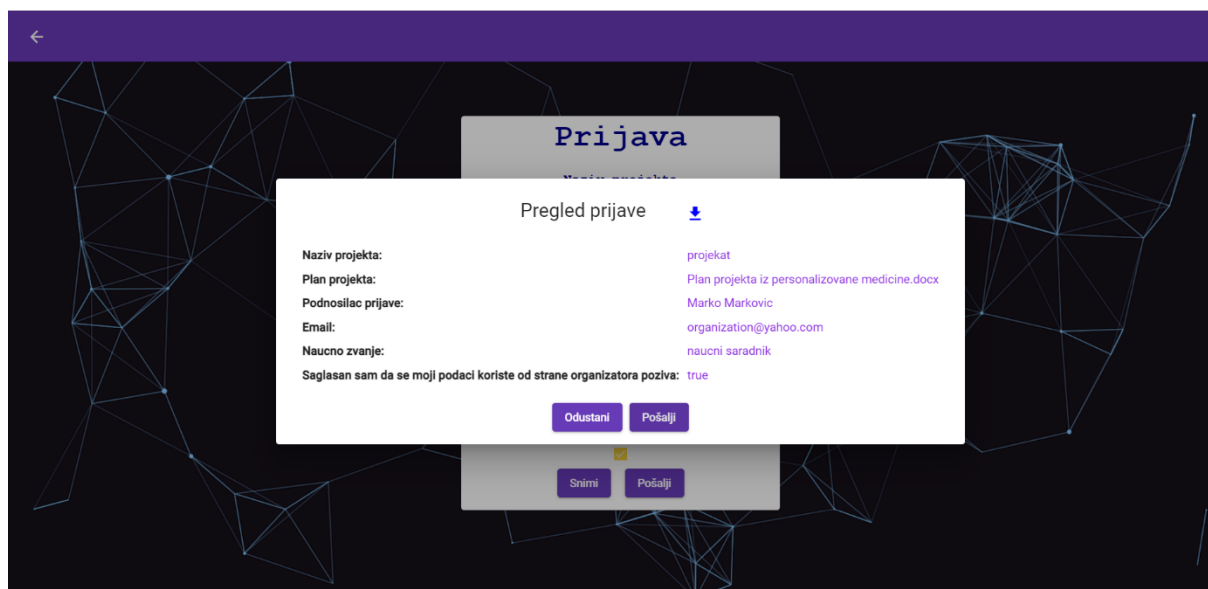
Slika 7. – *Prijava na poziv*

Prijava sadrži podatke koje je prethodno definisao administrator pri otvaranju poziva. Korisnik mora da popuni sve obavezne podatke i priloži sva obavezna dokumenta. Ukoliko u nekom trenutku korisnik odustane od prijave ima mogućnost da istu snimi klikom na dugme *Snimi* kada se prijava sa do tada popunjenim podacima čuva u Nacrtima. I korisniku se prikazuje prozor sa obaveštenjem da je njegova prijava sačuvana u Nacrtima.



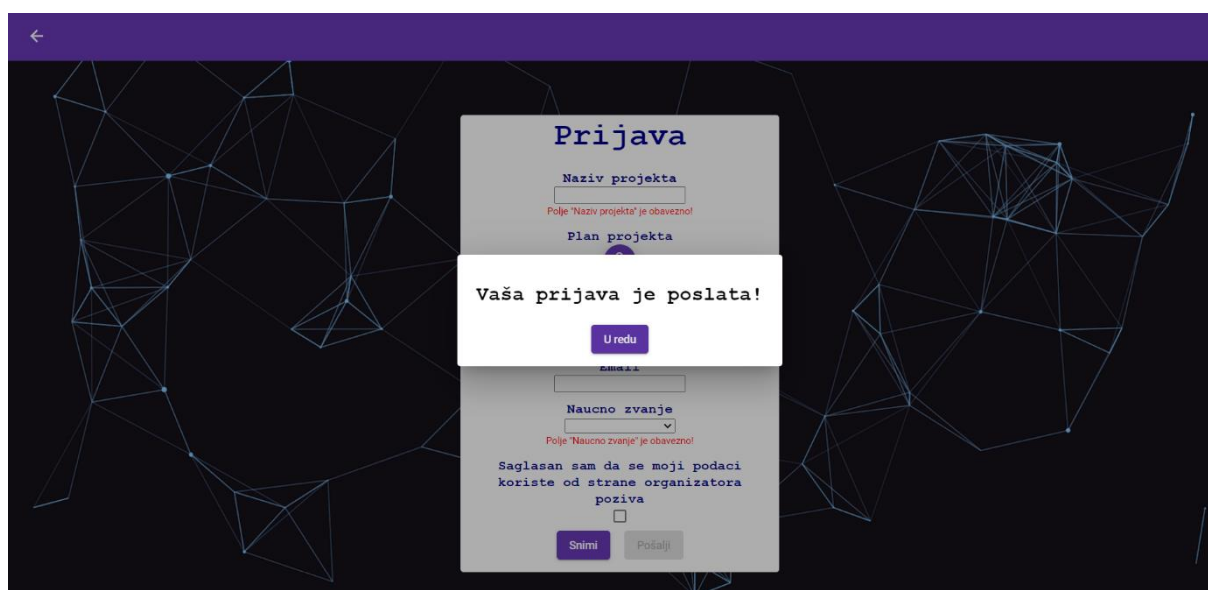
Slika 8. – *Snimanje prijave*

Kada korisnik popuni prijavu, unese sva obavezna polja dugme *Pošalji* će postati aktivno. Klikom na dugme *Pošalji* korisniku se otvara prozor sa pregledom prijave. Postoji mogućnost da se prijava preuzme kao pdf dokument, da se potvrdi slanje ili odustane od prijave.



Slika 8. – Pregled prijave

Kada korisnik klikne *Pošalji* iz prozora *Pregled prijave* prikazuje mu se obaveštenje o uspešnom slanju prijave.

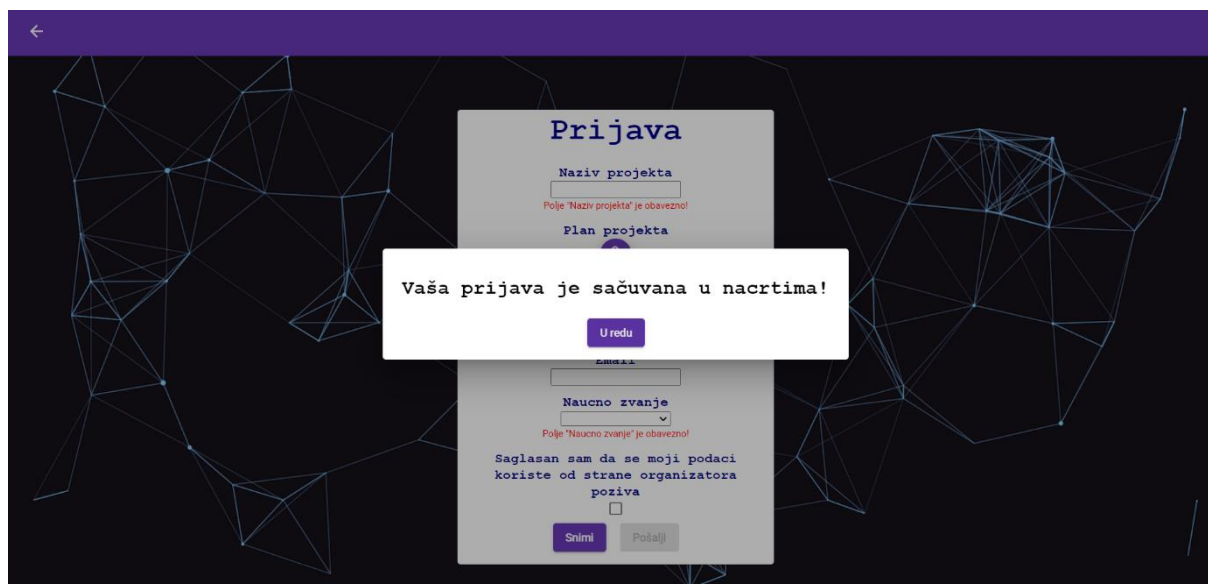


Slika 9. – Uspešno slanje prijave

Ukoliko nakon pregleda prijave korisnik odluči da ipak odustane od slanje prijave to će moći da uradi klikom na dugme *Odustani*. Nakon toga otvara se novi prozor sa pitanjem da li želi da sačuva prijavu u Nacrtima.



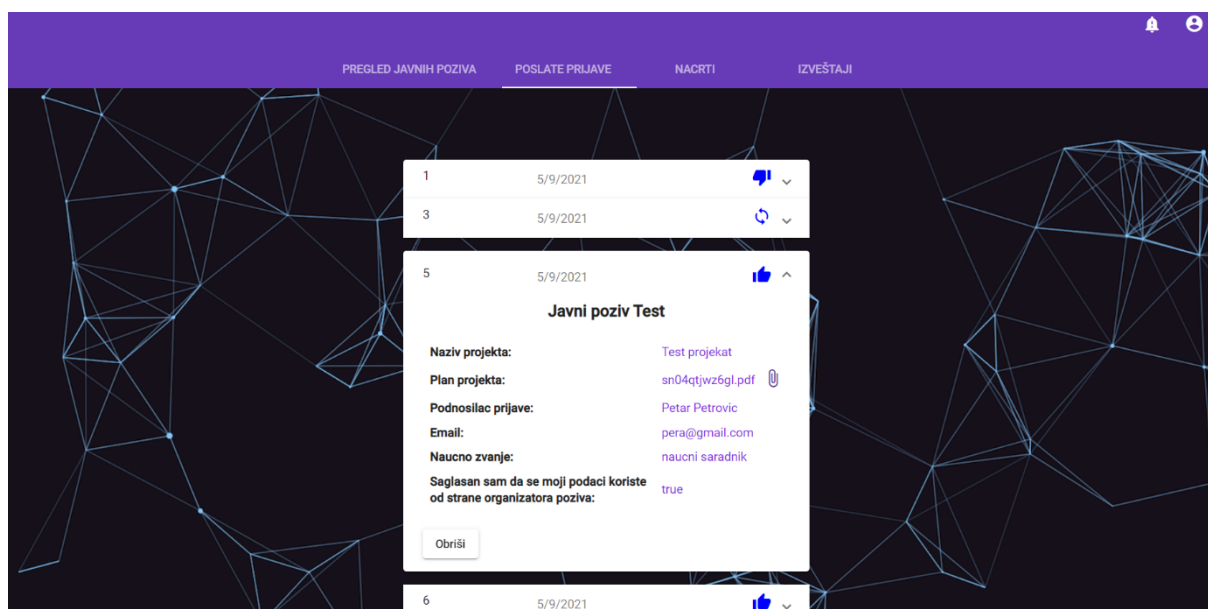
*Slika 10. – Odustajanje od prijave i čuvanje u Nacrtima*



*Slika 11. – Čuvanje prijave u Nacrtima*

### 3.2.2. POSLATE PRIJAVE

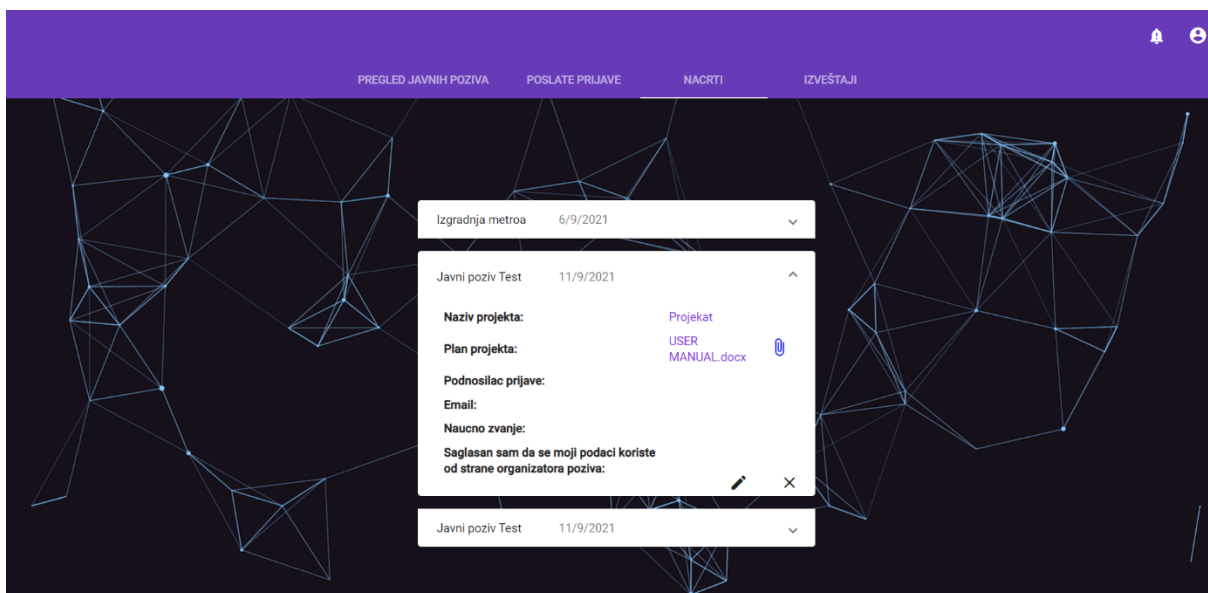
U okviru meni opcije *Poslate prijave* korisniku se prikazuju sve poslate prijave. Za prikaz svake prijave korišćena je komponenta *Expansion Panel* dok je za grupisanje svih prijava korišćena komponenta *Accordion*. U neekspandovanom obliku prijave prikazuju se ID javnog poziva na koji se prijava odnosi, datum slanja prijave i status prijave. Status prijave je predstavljen jednom od tri ikonice iz Angular Material-a: *loop* (ukoliko je prijava poslata, ali nije pregledana), *thumb up* (ukoliko je prijava prihvaćena od strane administratora) i *thumb down* (ukoliko je prijava odbijena od strane administratora). Klikom na prijavu ona se ekspanduje, u ekspanovanom delu se prikazuju podaci prijave, koje je korisnik popunio i dugme *Obriši* kojim se korisniku daje mogućnost da obriše prijavu tako da mu se ubuduće ne prikazuje u okviru poslatih prijava.



Slika 12. – Prikaz poslatih prijava

### 3.2.3. NACRTI

U okviru opcije *Nacrti* korisniku se prikazuju sve prethodno nedovršene i sačuvane prijave. Za implementaciju nacrtu su takođe korišćene komponente *Expansion panel* i *Accordion*. U zaglavlju panela su prikazani naziv javnog poziva na koji se prijava odnosi i datum kreiranja nacrtu. U ekspanzovanom delu panela prikazuju se podaci prijave koje je korisnik popunio do trenutka čuvanja prijave u nacrtima. Takođe se u ekspanzovanom delu prikazuju i dve opcije *Izmena nacrtu* i *Brisanje nacrtu*, predstavljeni ikonicama *Create* i *Clear* respektivno.



Slika 13. – Nacrti

Klikom na ikonu olovke korisniku se ponovo prikazuje stranica sa formom prijave sa popunjenim podacima iz nacrtu prijave. Korisnik ima mogućnost da izmeni te podatke, da popuni prijavu do kraja i da je pošalje.

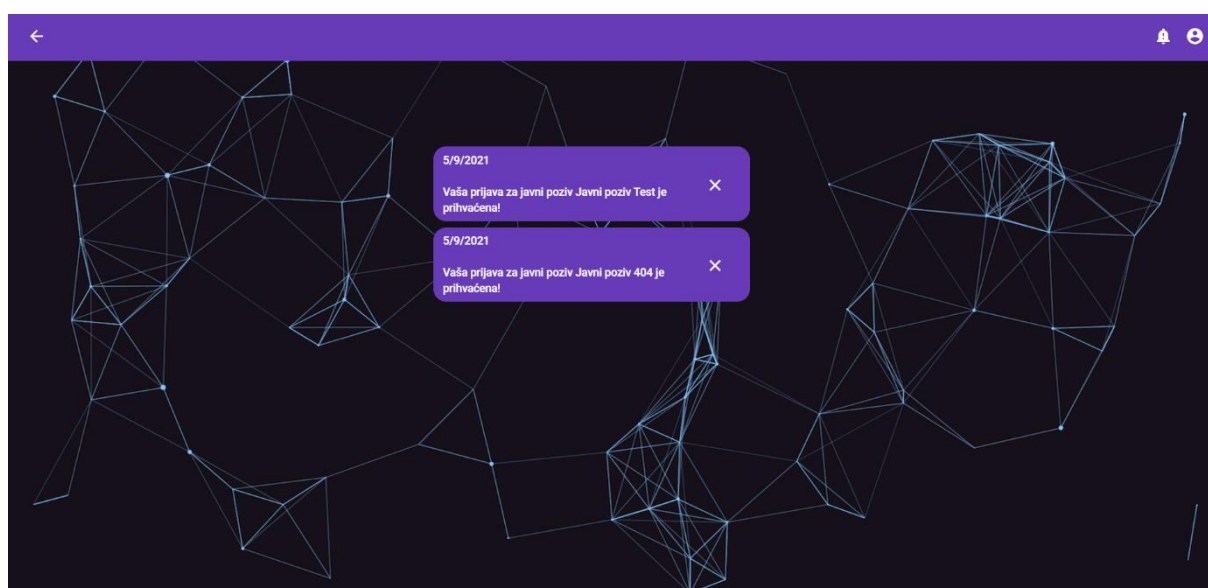


Slika 14. – Izmena nacrtu prijave



### 3.2.4. OBAVEŠTENJA

Korisnik može da pristupi *Obaveštenjima* klikom na ikonicu zvona koje se nalazi u desnom uglu *Toolbar-a*. U obaveštenjima se nalaze poruke koje korisnik dobija za svaku prijavu koju administrator prihvati. Poruke sadrže datum slanja, tekst poruke i dve dodatne opcije. Ukoliko je poruka nepročitana prikazuje se ikonica poruke, klikom na ikonicu poruka se označava kao pročitana. Druga dostupna opcija je ikonica *Clear* (x) kojom se poruka briše iz obaveštenja. Poruke su implementirane korišćenjem komponente *Chips* iz Angular Material-a.

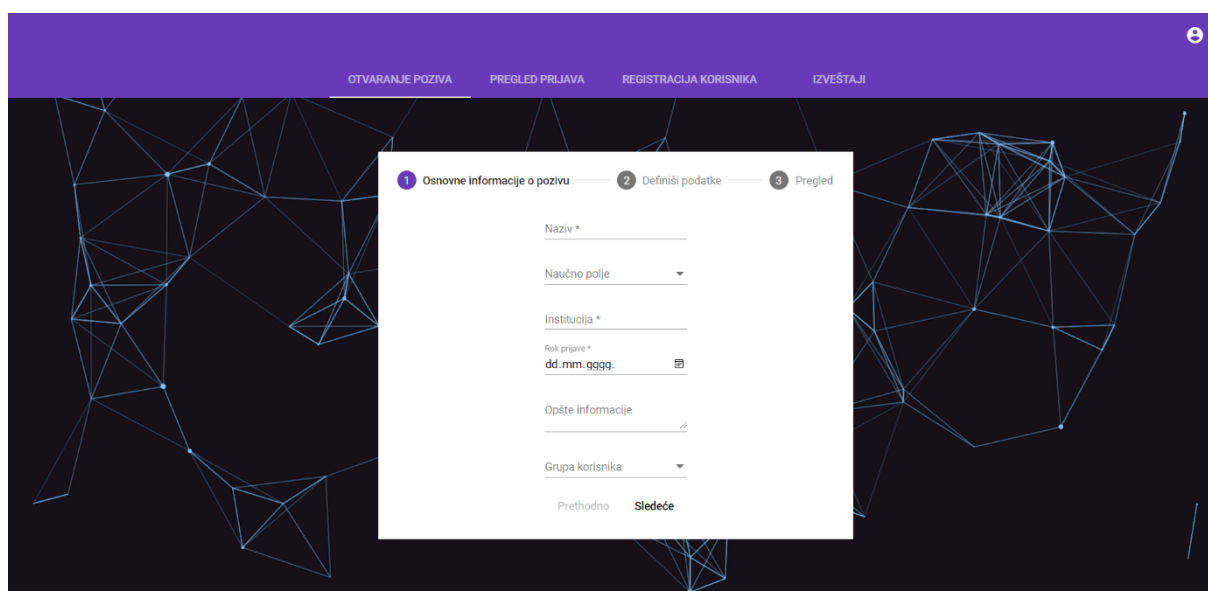


Slika 15. - Obaveštenja

## 3.3. ADMINISTRATOR

### 3.3.1. OTVARANJE POZIVA

Nakon uspešne prijave administratora na sistem, prikazuje mu se stranica sa formom za otvaranje novog javnog poziva. Opcija menija *Otvaranje poziva* sadrži komponentu *Stepper*. Pri otvaranju javnog poziva prolazi se kroz sledeća tri koraka: definisanje osnovnih informacija o pozivu (ovaj korak podrazumeva definisanje naziva javnog poziva, naučnog polja, institucije, roka prijave, teksta poziva i ciljne grupe korisnika), definisanje podataka neophodnih za prijavu na poziv (ovaj korak podrazumeva definisanje naziva podatka, tipa podatka i određivanje da li je podatak obavezan) i poslednji korak pri otvaranju poziva sadrži pregled upravo otvorenog javnog poziva.

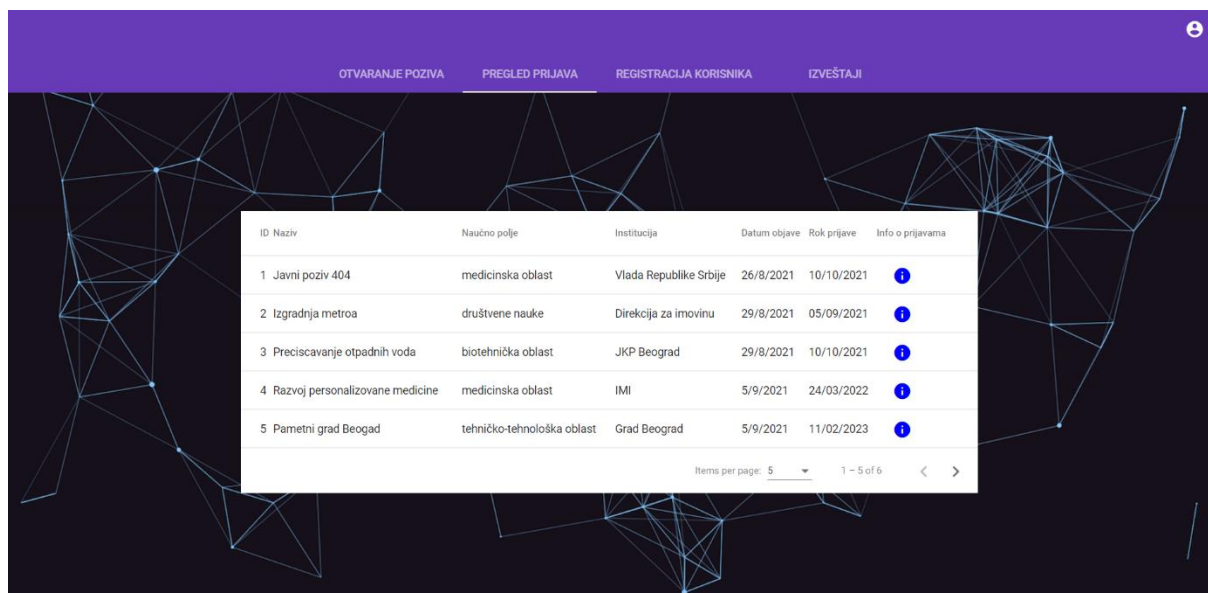


Slika 16. – Otvaranje javnog poziva



### 3.3.2. PREGLED PRIJAVA

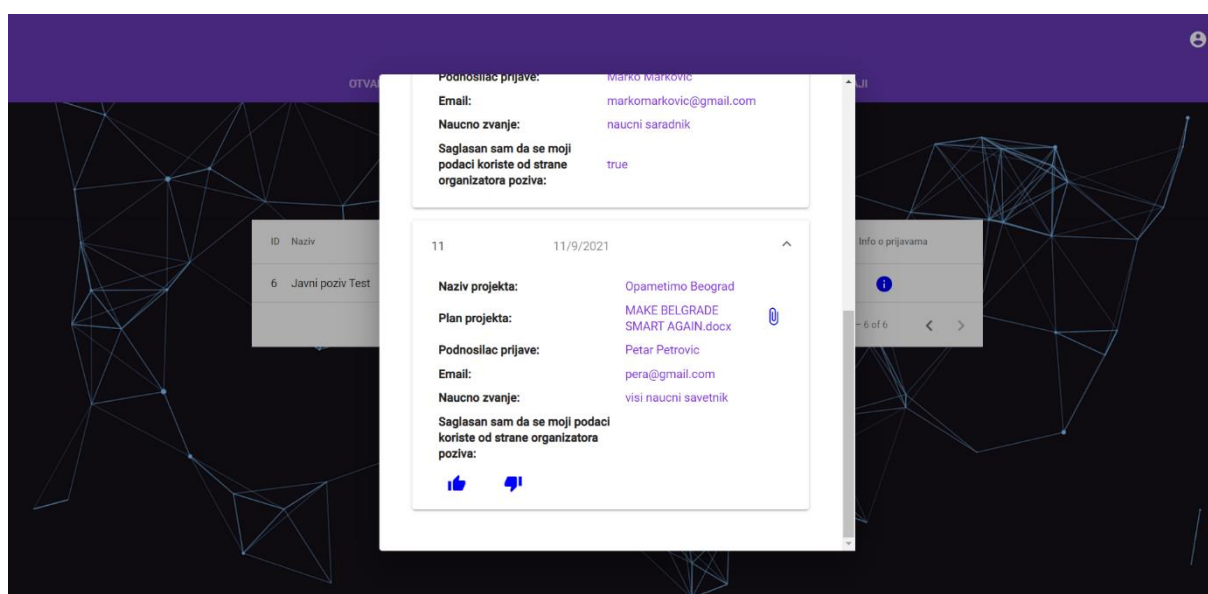
U okviru opcije *Pregled prijava* administratoru se prikazuju tabelarno svi otvoreni javni pozivi. Tabela sadrži informacije o ID-ju, nazivu, naučnom polju, instituciji, datumu objave, roku prijave i u dodatnoj koloni *Info o prijavama* nalazi se ikonica info, klikom na nju administratoru se prikazuje prozor sa svim dospelim prijavama za odabrani javni poziv.



ID	Naziv	Naučno polje	Institucija	Datum objave	Rok prijave	Info o prijavama
1	Javni poziv 404	medicinska oblast	Vlada Republike Srbije	26/8/2021	10/10/2021	
2	Izgradnja metroa	društvene nauke	Direkcija za imovinu	29/8/2021	05/09/2021	
3	Preciscavanje otpadnih voda	biotehnička oblast	JKP Beograd	29/8/2021	10/10/2021	
4	Razvoj personalizovane medicine	medicinska oblast	IMI	5/9/2021	24/03/2022	
5	Pametni grad Beograd	tehničko-tehnološka oblast	Grad Beograd	5/9/2021	11/02/2023	

Slika 17. – Pregled prijava

U novootvorenom prozoru sa prijavama administratoru se prikazuju sve prijave za taj poziv sa svim popunjenim podacima i priloženim dokumentima, takođe se prikazuju i datum slanja prijave i ID prijave. Administratoru su za svaku prijavu dostupne dve opcije da prihvati prijavu (*thumb\_up*) ili da odbije prijavu (*thumb\_down*). Nakon što izvrši jednu od ove dve akcije, administratoru te akcije više neće biti vidljive, što znači da je prijava pregledana i evidentirana.



Podnosilac prijave: Marko Markovic  
Email: markomarkovic@gmail.com  
Naučno zvanje: naucni saradnik  
Saglasan sam da se moji podaci koriste od strane organizatora poziva: true

11 11/9/2021

Naziv projekta: Opametimo Beograd  
Plan projekta: MAKE BELGRADE SMART AGAIN.docx  
Podnosilac prijave: Petar Petrovic  
Email: pera@gmail.com  
Naučno zvanje: visi naucni savetnik  
Saglasan sam da se moji podaci koriste od strane organizatora poziva:

Slika 18. – Prozor sa poslatim prijavama

### 3.3.3. REGISTRACIJA KORISNIKA

U ovoj opciji administratoru je omogućeno da definiše podatke koji će se zahtevati pri registraciji korisnika. Forma je implementirana u okviru komponente *Stepper* i proces registracije korisnika se odvija u tri koraka. U prvom koraku administrator definiše sve moguće kategorije (grupe) korisnika u sistemu, npr. medicinska grupa, biotehnička grupa, prirodno-matematička grupa, grupa društvenih nauka itd. U drugom koraku se definišu sami podaci registracije, upisuje se naziv traženog podatka, bira se tip podataka (tekstualni podatak, email, link, dokument, datum, lozinka, radio dugme...) i definiše se da li je obavezan ili ne. Treći korak sadrži pregled upravo definisanih podataka za registraciju.

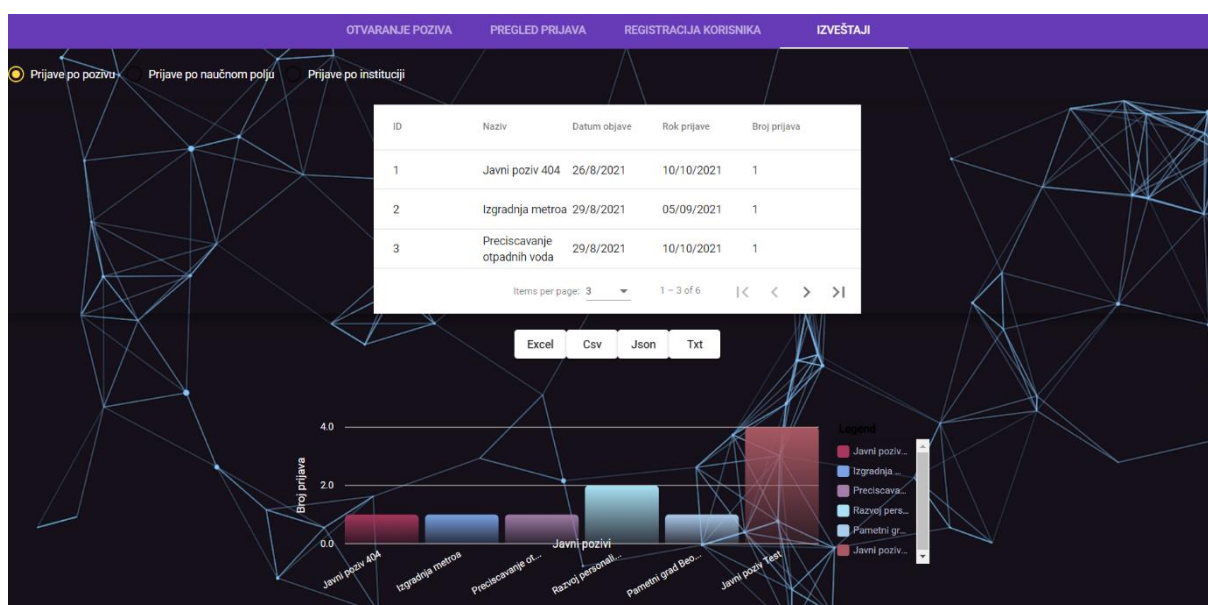
The screenshot shows a web application interface with a purple header. The header contains four navigation links: 'OTVARANJE POZIVA', 'PREGLED PRIJAVA', 'REGISTRACIJA KORISNIKA' (which is highlighted), and 'IZVEŠTAJI'. Below the header, the main content area has a dark background with a white geometric pattern. The title 'Definisanje podataka za registraciju korisnika' is displayed in white. A white form is centered on the screen, featuring a stepper with three steps: '1 Korisničke grupe', '2 Ostali podaci' (which is active), and '3 Pregled'. The form includes a text input field for 'Traženi podatak', a dropdown menu for 'Tip podatka', and a checkbox for 'Obavezan podatak?'. A purple button labeled 'Dodaj podatak' is positioned below the checkbox. At the bottom of the form, there are two links: 'Prethodno' and 'Sledeće'.

Slika 19. – Definisanje podataka za registraciju korisnika

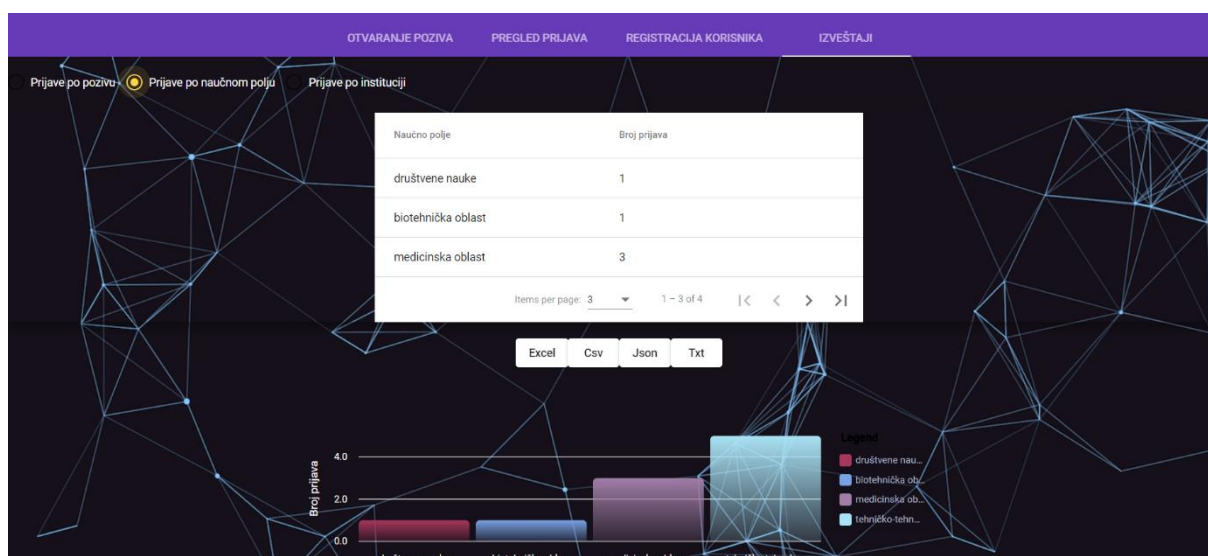
## 3.4. ZAJEDNIČKE FUNKCIONALNOSTI

### 3.4.1. IZVEŠTAJI

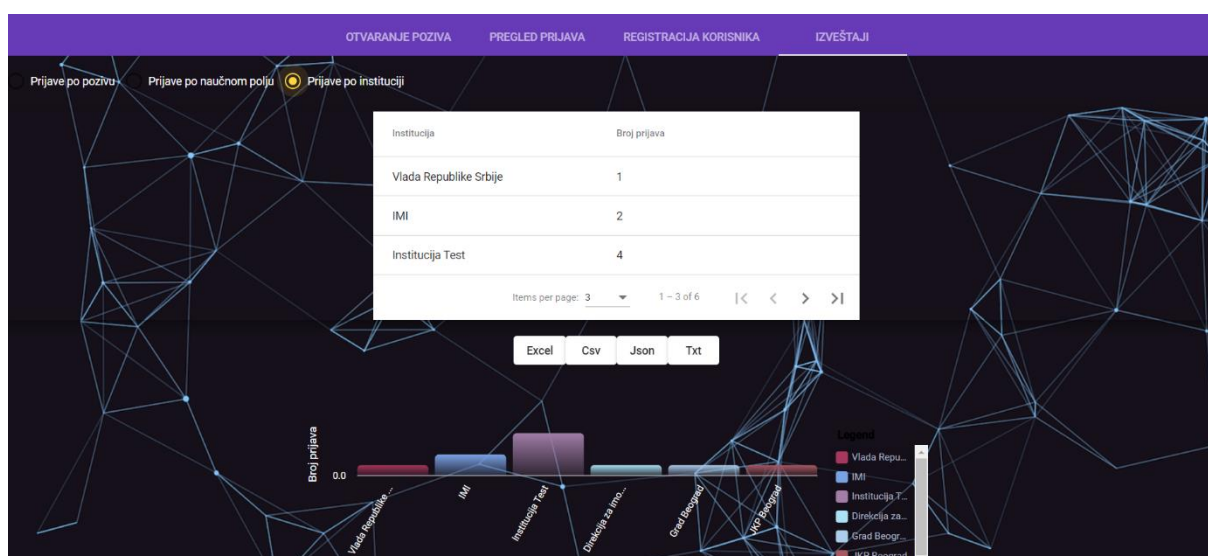
I korisnik i administrator u okviru svojih menija imaju opciju *Izveštaji*. Ova opcija služi za prikaz statistike samih prijava na javne pozive. Postoje tri tipa izveštaja u sistemu: izveštaji za prijave po pozivu, izveštaji za prijave po naučnom polju i izveštaje za prijave po instituciji. Izveštaji su prikazani u dva oblika, u vidu tabele, sa mogućnošću eksportovanja iste kao excel, csv, text ili json fajl, i u vidu grafikona (grafikon je implementiran korišćenjem komponente *horizontal bar chart* iz *ngx-chart* paketa).



Slika 20. – Izveštaj za prijave po javnom pozivu



Slika 21. – Izveštaj za prijave po naučnom polju

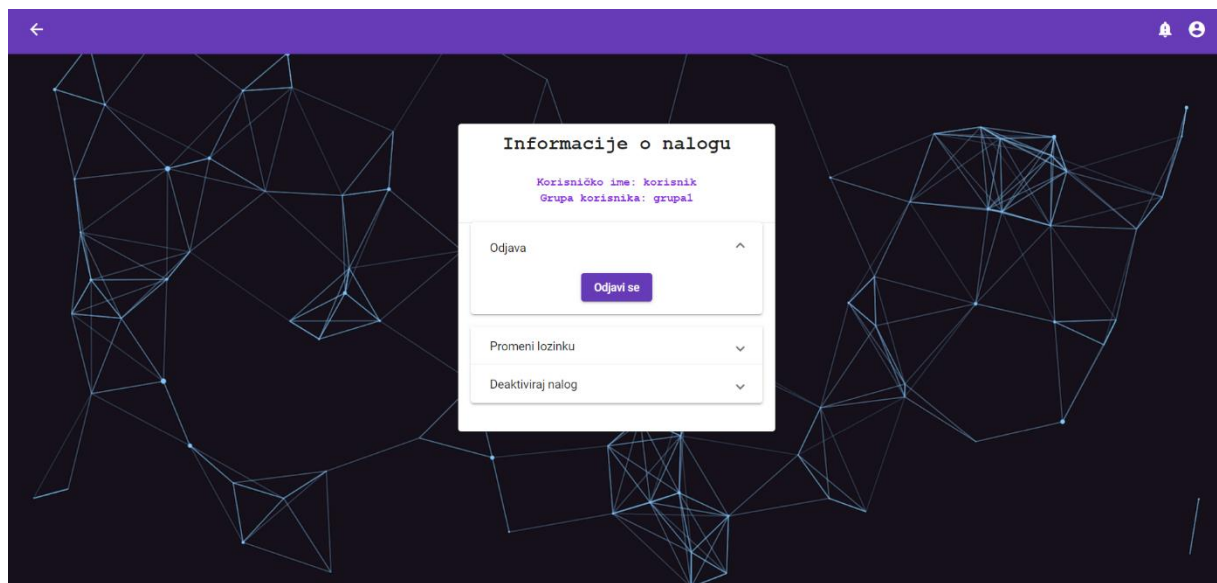


Slika 22. – Izveštaj za prijave po instituciji

### 3.4.2. NALOG

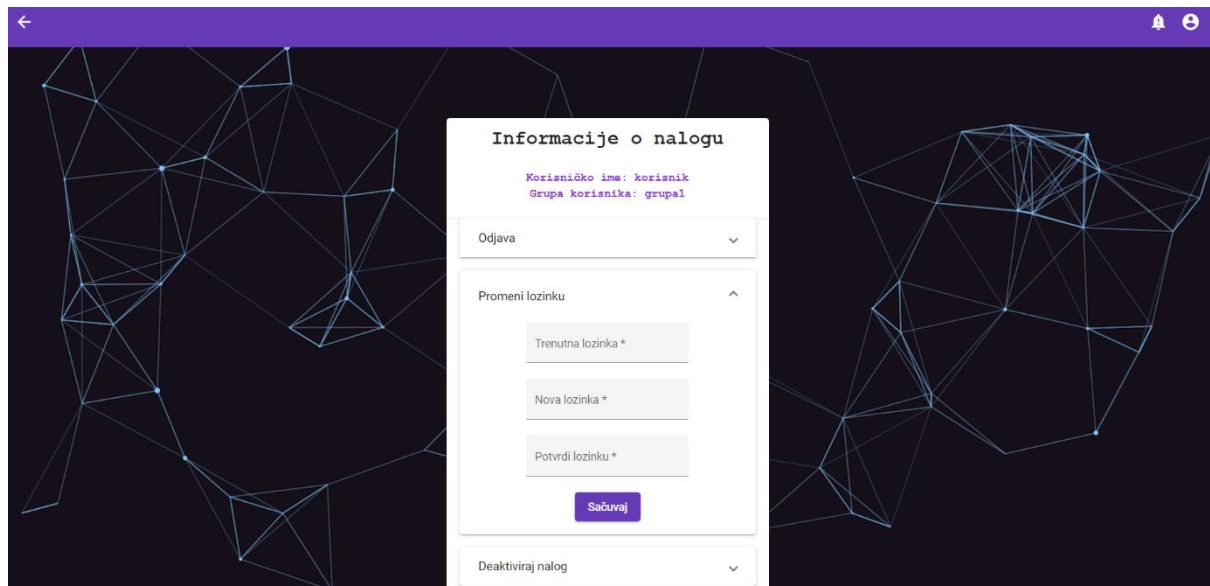
U opciji *Nalog* kojoj se pristupa klikom na ikonicu *Account circle* u desnom uglu *Toolbar*-a, dostupne se osnovne funkcionalnosti sa nalogom. Korisnici (admin i obični korisnici) imaju uvid u neke osnovne podatke o nalogu i tri opcije, odjava, promena lozinke i deaktiviranje naloga.

Klikom na odjavu korisnik se odjavljuje sa sistema i navigira na početnu stranicu za prijavu.



Slika 23. – Odjava

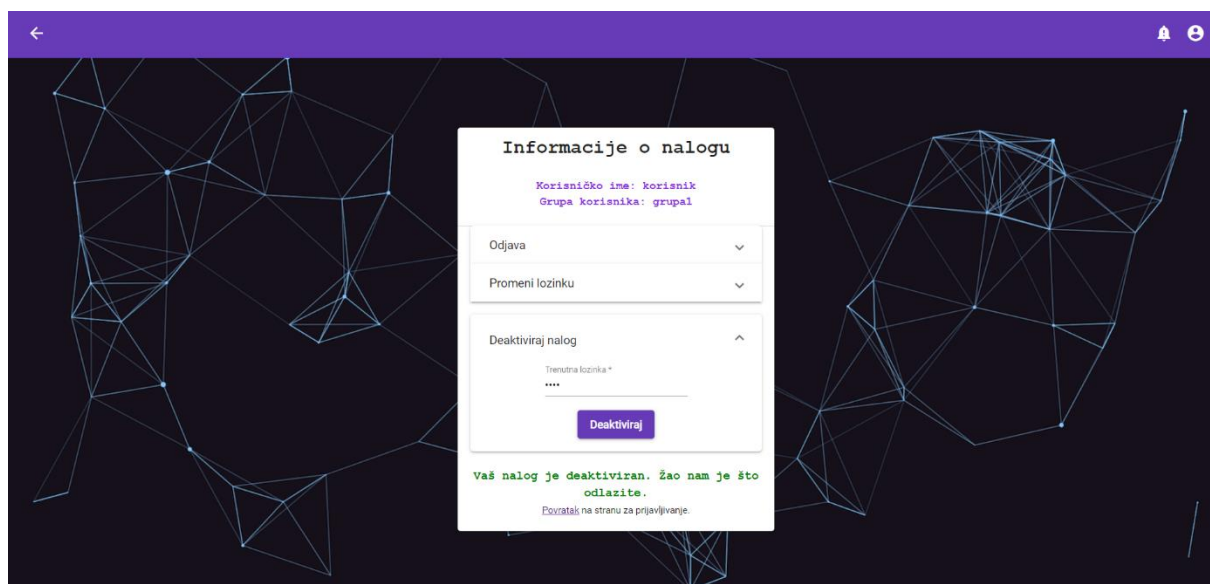
U okviru opcije *Promena lozinke* korisnik mora prvo da unese trenutnu lozinku zatim da izabere novu lozinku i da je potvrdi. Nakon uspešne promene lozinke korisnik se obaveštava o tome i pruža mu se mogućnost da se vrati na stranu za prijavu.



The screenshot shows a mobile application interface with a dark purple header bar containing a back arrow, a bell icon, and a user profile icon. The background is a dark blue with a white geometric wireframe pattern. A white modal box titled 'Informacije o nalogu' is centered. It displays user information: 'Korisničko ime: korisnik' and 'Grupa korisnika: grupal'. Below this are three expandable sections: 'Odjava' (Logout), 'Promeni lozinku' (Change Password), and 'Deaktiviraj nalog' (Deactivate Account). The 'Promeni lozinku' section is expanded, showing three input fields: 'Trenutna lozinka \*', 'Nova lozinka \*', and 'Potvrdi lozinku \*'. A purple 'Sačuvaj' (Save) button is at the bottom of this section.

Slika 24. – Promena lozinke

Kao poslednja opcija korisnicima se pruža mogućnost da deaktiviraju nalog, nakon koje se njihov nalog označava kao neaktivan i onemogućena je prijava na sistem pomoću tog naloga. Nakon uspešne deaktivacije naloga korisnik se automatski odjavljuje sa sistema i navigira na početnu stranu za prijavu.



This screenshot shows the same 'Informacije o nalogu' modal box, but now the 'Deaktiviraj nalog' section is expanded. It contains a single input field for 'Trenutna lozinka \*' with masked characters (\*\*\*\*). A purple 'Deaktiviraj' button is below the field. At the bottom of the modal, a green success message reads: 'Vaš nalog je deaktiviran. Žao nam je što odlazite.' followed by a link: 'Povratni na stranu za prijavljivanje.'

Slika 25. – Deaktiviranje naloga



# REALIZACIJA SISTEMA

**Prvi problem** sa kojim sam se susrela prilikom izrade aplikacije bio je **modelovanje prijava u bazi**. Naime, podaci prijave mogu da se razlikuju od javnog poziva do javnog poziva. Podaci prijave nisu statički niti predefinisani, nego administrator sistema ima mogućnost da za svaki poziv definiše podatke prijave. Samim tim zbog nehomogenosti prijava korišćenje neke SQL baze mi nije delovalo najlogičnije, već se korišćenje MongoDB-a nametnulo kao prirodno rešenje jer ne forsira homogenost i uniformnu struktruiranost podataka u okviru iste grupe (kolekcije/tabele). U MongoDB-u srodni podaci koji se grupišu u okviru iste kolekcije ne moraju imati i istu strukturu, samim tim bilo je moguće napraviti kolekciju prijava koja će sadržati prijave koje obuhvataju različite skupove podataka.

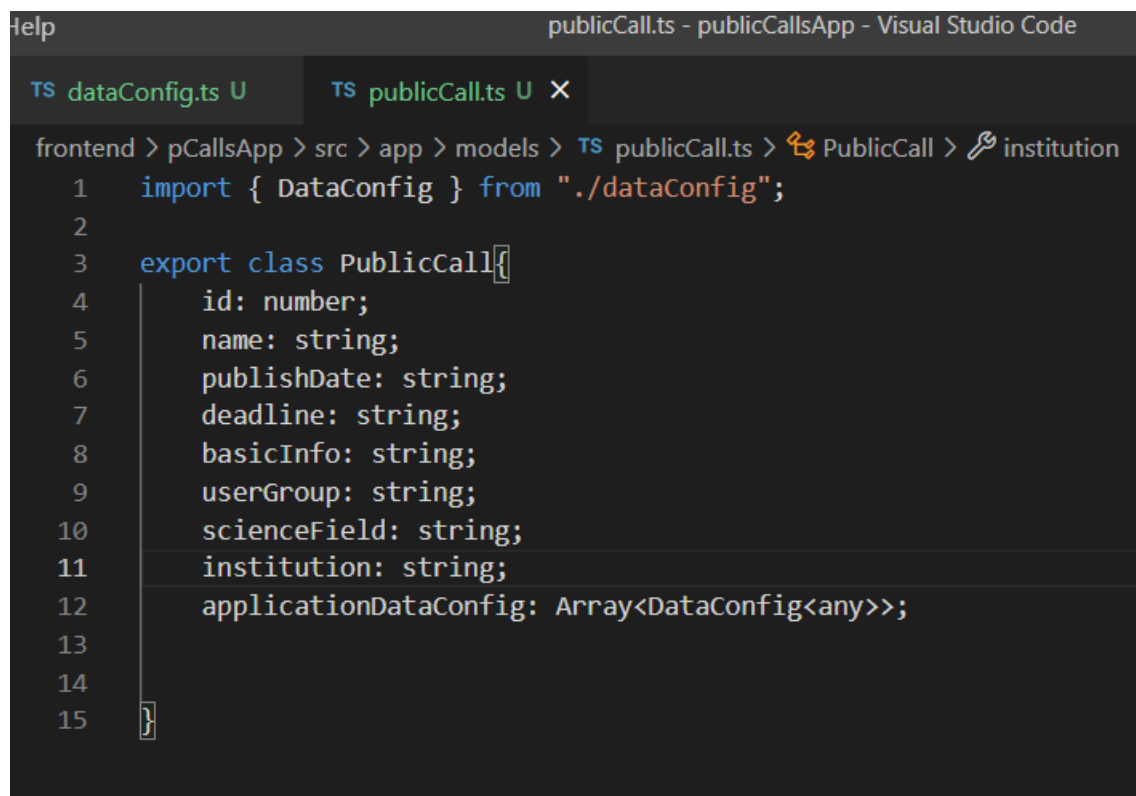
**Drugi problem i treći problem** su bili **modelovanje javnog poziva i realizacija dinamičkih formi**. U okviru zahteva sistema nalazi se mogućnost da administrator definiše podatke prijave za svaki javni poziv koje će kasnije korisnik popunjavati u okviru neke forme. Ovo je rešeno tako što sam modelovala javni poziv tako da, pored osnovnih informacija o pozivu koja postoje kao polja modela (klase) *PublicCall*, sadrži i polje tipa niza konfiguracionih objekta. Napravila sam klasu *DataConfig* koja predstavlja u suštini jedno polje forme i sve attribute koji opisuju to polje, kao što su: naziv polja (name atribut), labela polja (label atribut), da li je polje obavezno (required), tip polja (type), zatim tip input polja (inputType koje može biti text, url, file, password, radio, date...), vrednost polja (value atribut) i opcije u slučaju da je polje tipa *select* ili *radio button*. Svaki objekat klase *PublicCall* sadrži kao jedan od atributa niz ovih *DataConfig* objekata, koji u suštini opisuje samu formu prijave.

```
TS dataConfig.ts U X
frontend > pCallsApp > src > app > models > TS dataConfig.ts > DataConfig
1  export class DataConfig<T>{
2      value: T | undefined;
3      name: string;
4      label: string;
5      inputType: string;
6      type: string;
7      required: boolean;
8      options: {key: string, value: string}[];
9
10
11
12  constructor( dataConfigParameter: {
13      value?: T;
14      name?: string;
15      label?: string;
16      inputType?: string;
17      type?: string;
18      required?: boolean;
19      options?: {key: string, value: string}[];
20
21  }={}){
22      this.value=dataConfigParameter.value;
23      this.name=dataConfigParameter.name || '';
24      this.label=dataConfigParameter.label || '';
25      this.inputType=dataConfigParameter.inputType || '';
26      this.type=dataConfigParameter.type || '';
27      this.required= !!dataConfigParameter.required;
28      this.options=dataConfigParameter.options || [];
29
30  }
31
32  }
```

Slika 26. – Izgled klase DataConfig



Administratoru je u okviru opcije *Otvaranje javnog poziva* obezbeđena funkcija definisanja podataka prijave, tako što za svaki podatak unese naziv tog podatka, to je u stvari label atribut polja forme, zatim u okviru padajuće liste bira tip podatka i ima opciju da čekira da li podatak treba da bude obavezan (što se preslikava u required atribut polja forme). Na osnovu ovako definisanog izgleda jednog podatka prijave formirala sam na odgovarajući način *DataConfig* objekat i dodavala ga u niz *applicationDataConfig* koji predstavlja polje objekta *PublicCall*.



```
Help publicCall.ts - publicCallsApp - Visual Studio Code
TS dataConfig.ts U TS publicCall.ts U X
frontend > pCallsApp > src > app > models > TS publicCall.ts > PublicCall > institution
1 import { DataConfig } from "../dataConfig";
2
3 export class PublicCall {
4     id: number;
5     name: string;
6     publishDate: string;
7     deadline: string;
8     basicInfo: string;
9     userGroup: string;
10    scienceField: string;
11    institution: string;
12    applicationDataConfig: Array<DataConfig<any>>;
13
14 }
15 }
```

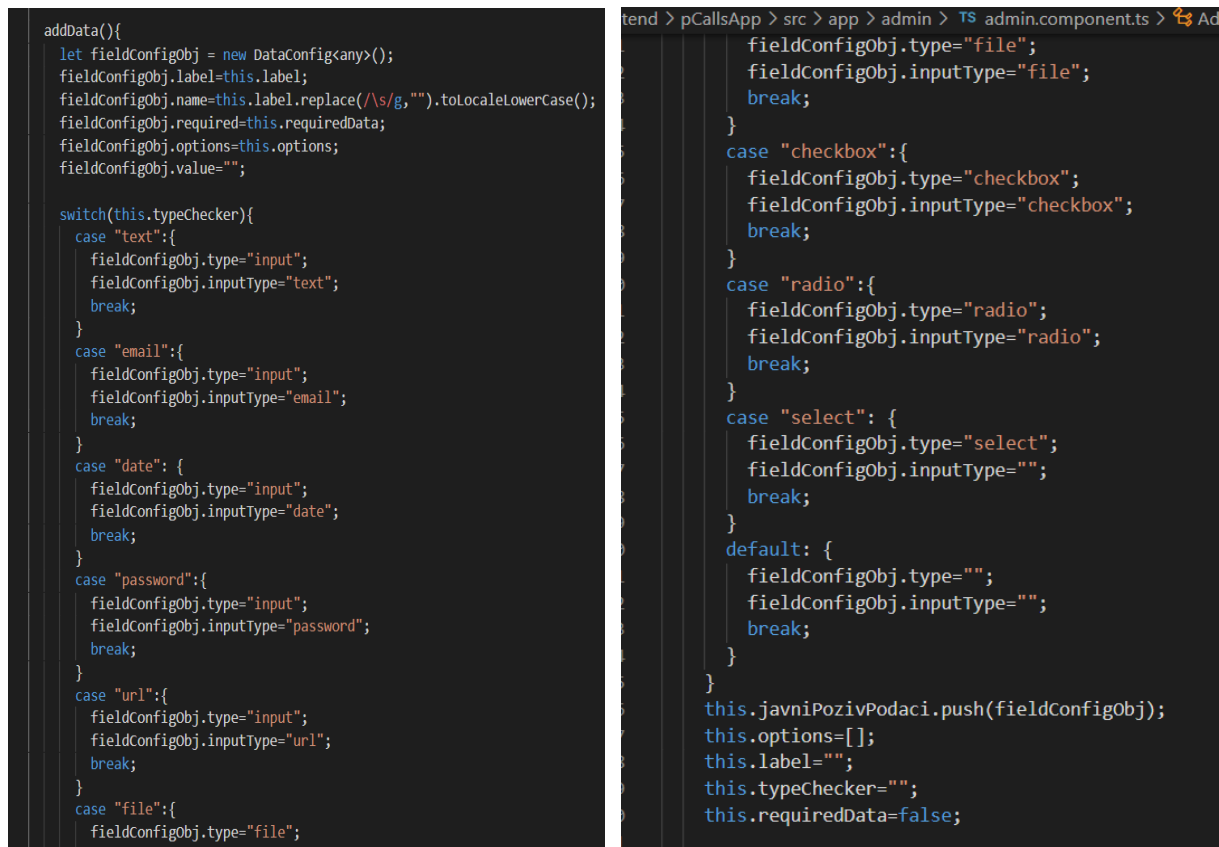
Slika 27. – Izgled klase Javni poziv (*PublicCall*)

```
config.ts U    TS publicCall.ts U    admin.component.html U X
> pCallsApp > src > app > admin > admin.component.html > mat-tab-group > mat-tab > mat-stepper > mat-step > form
<form>
  <ng-template matStepLabel>Definiši podatke</ng-template>

  <mat-form-field>
    <mat-label>Traženi podatak</mat-label>
    <input matInput type="text" name="fieldName" [(ngModel)]='label'>
  </mat-form-field><br>
  <mat-form-field>
    <mat-label>Tip podatka</mat-label>
    <mat-select name="tip" [(ngModel)]='typeChecker'>
      <mat-option value="text">Tekstualni podatak</mat-option>
      <mat-option value="email">Email adresa</mat-option>
      <mat-option value="date">Datum</mat-option>
      <mat-option value="password">Lozinka</mat-option>
      <mat-option value="url">Link</mat-option>
      <mat-option value="file">Dokument</mat-option>
      <mat-option value="checkbox">Štikliranje</mat-option>
      <mat-option value="select">Padajući meni</mat-option>
      <mat-option value="radio">Radio dugme</mat-option>
    </mat-select>
  </mat-form-field><br>

  <mat-checkbox name="required" [(ngModel)]='requiredData'>Obavezan podatak?</mat-checkbox>
  <br><br><br>
  <div *ngIf='typeChecker=="select" || typeChecker=="radio"' style="justify-content: center;">
    <h3>Dodavanje opcija</h3>
    <mat-form-field>
      <mat-label>Naziv</mat-label>
      <input matInput type="text" name="opcija" [(ngModel)]='optionValue'>
    </mat-form-field>
    <br><br><br>
    <button mat-raised-button color="primary" (click)="addOption()">Dodaj opciju</button>
  </div>
</form>
```

Slika 28. – Definisane podataka prijave od strane admina (html fajl)



Slika 29. – Definisanje podataka prijave od strane admina (ts fajl)

U ts fajlu koji se vidi na prethodnoj slici realizuje se kreiranje objekta tipa *DataConfig* na osnovu unesenih podataka u formi prikazanoj na slici 28. i dodavanje istog u niz.

Na slici 30. prikazana je implementacija funkcije otvaranje javnog poziva (openCall) u okviru koje se kreira objekat tipa *PublicCall*, setuju se sva polja tog objekta, kao što su naziv, tekst javnog poziva, datum objave, rok prijave, naučno polje, institucija, korisnička grupa, ali se takođe setuje i polje *applicationDataConfig* koje predstavlja konfiguracioni niz prijave za javni poziv.

```

this.service.openPublicCall([this.javniPoziv.name,
    publishDateFormatted, deadlineFormatted,
    this.javniPoziv.basicInfo,
    this.javniPoziv.userGroup,
    this.javniPoziv.scienceField,
    this.javniPoziv.institution,
    this.javniPozivPodaci]).subscribe(res=>{
    if(res['message']=="added call"){
        this.message = "Uspešno ste otvorili javni poziv!";
        const dialogRef = this.dialog.open(DialogOverviewExampleDialog,{
            width: '300px'
        });

        dialogRef.afterClosed().subscribe(result=>{
            this.message="";
        })
    }else{
        console.log(res);
    }
});
this.javniPoziv={} as PublicCall;
this.javniPozivPodaci=[];
this.publishDatePretty="";
this.deadlinePretty="";
stepper.selectedIndex=0;

```

*Slika 30. – Implementacija funkcije openCall*

Na slici 30. vidimo poziv metode servisa *openPublicCall* koja na osnovu prosleđenih parametara šalje http zahtev ka backend delu gde se kreira novi javni poziv u bazi.

**Četvrti problem** je bio **implementacija dinamičkih formi**. Nakon što smo otvorili javni poziv i definisali sve podatke prijave za taj javni poziv (tj. konfigurisali svako polje forme za prijavu na javni poziv), trebalo je realizovati prikaz te forme korisniku. Cilj je bio iskoristiti nekako onaj konfiguracioni niz iz objekta kojim je modelovan javni poziv i na osnovu njega kreirati formu i prikazati je korisniku. Ovo je rešeno tako što nakon što korisnik klikne na dugme *Prijavi se*, kojim se prijavljuje na izabran poziv, iz baze se dovlači taj javni poziv, a zatim se u frontend delu aplikacije konkretno u html fajlu prijave za poziv, iterira kroz konfiguracioni niz izabranog javnog poziva i uz pomoć ngSwitcha-a prikazuju odgovarajuća polje forme.

```
}
this.currentUser=JSON.parse(localStorage.getItem("loggedUser"));
this.service.getQuestionsFromBackend(parseInt(id)).subscribe((publicCall: PublicCall)=>{
    this.pCall=publicCall;
    this.questions=publicCall.applicationDataConfig;
    this.form=this.toFormGroup();|
```

*Slika 31. – Dohvatanje poziva iz baze*

Na slici 31. prikazano je dohvaćanje izabranog javnog poziva iz baze setovanje polja *questions* konfiguracionim nizom objekta javni poziv i formiranje forme.

```
isValid(question) {
    return this.form.controls[question.name].valid;
}

toFormGroup() {
    const group: any = {};

    this.questions.forEach(question => {
        group[question.name] = question.required ? new FormControl(question.value || '', Validators.required)
        : new FormControl(question.value || '');
    });
    return new FormGroup(group);
}
```

*Slika 32. – Implementacija metoda isValid i toFormGroup*

Metoda *toFormGroup* za svaki objekat niza *questions*, koji predstavlja jedno polje forme, kreira grupu i dodeljuje joj odgovarajuću *FormControl* klasu. Ova metoda služi da bismo dinamički validirali sva polja forme koja su označena kao obavezna.

Metodu *isValid* koristimo da bismo ispitali da li je odgovarajuće polje forme, ovde na slici 32. označeno kao *question*, validno, tj. da li je popunjeno ukoliko je označeno kao obavezno ukoliko ovaj uslov nije ispunjen korisniku se prikazuje greška.

```
<div>
  <form (ngSubmit)="onSubmit()" [formGroup]="form" *ngIf="form" enctype='multipart/form-data'>
    <div *ngFor="let question of questions" class="form-row">
      <div [formGroup]="form">
        <label [attr.for]="question.name"
          style="color: darkblue; font-family: 'Courier New', Courier, monospace; font-weight: bold; font-size: 17px;">{{question.label}}</lab

        <div [ngSwitch]="question.type">
          <input *ngSwitchCase="'input'" [formControlName]="question.name" [id]="question.name"
            [type]="question.inputType">
          <div class="form-group" *ngSwitchCase="'file'">
            <input type="file" [id]="question.name" (change)="onFileChange($event, question.name)"
              [formControlName]="question.name" #fileUpload class="file-input">
            <div class="file-upload">
              <button mat-mini-fab color="primary" class="upload-btn" type="button" (click)="fileUpload.click()">
                <mat-icon>attach_file</mat-icon>
              </button>
              {{fileNames[question.name]}}
            </div>
          </div>
          <select [id]="question.name" *ngSwitchCase="'select'" [formControlName]="question.name">
            <option *ngFor="let opt of question.options" [value]="opt.value">{{opt.value}}</option>
          </select>
          <mat-radio-group *ngSwitchCase="'radio'" [formControlName]="question.name">
            <mat-radio-button *ngFor="let item of question.options" [value]="item.value">
              {{item.value}}&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</mat-radio-button>
            </mat-radio-group>
          <section *ngSwitchCase="'checkbox'">
            <mat-checkbox [formControlName]="question.name"></mat-checkbox>
          </section>
        </div>
        <div class="errorMessage" *ngIf="!isValid(question)" style="font-size: 12px;">Polje "{{question.label}}" je
          obavezno!</div>
      </div>
    </div>
  </form>
</div>
```

Slika 33. – Implementacija dinamičke forme (html fajl)

**Peti problem** sa kojim sam se susrela je **kako pokupiti i skladištiti u bazi unete podatke forme za prijavu na javni poziv**. Prijava na javni poziv je modelovana klasom *Application*, ta klasa pored polja koja su zajednička za svaku prijavu, sadrži i objekat koji sadrži specifične podatke prijave koji mogu da se razlikuju po strukturi od poziva do poziva. Implementacija klase *Application* prikazana je na slici 34.

```
export class Application{  
  id: number;  
  idCall: number;  
  callName: string;  
  user: string;  
  postingDate: string;  
  status: string;  
  labels: string[];  
  toShowUser: number;  
  field: string;  
  institution: string;  
  data: Object;  
}
```

Slika 34. – Implementacija klase *Application*

```
onSubmit() {  
  this.payload = this.form.getRawValue();  
  this.currentApplication=new Application();  
  this.currentApplication.data=this.payload;  
}
```

Slika 35. – Čuvanje podataka forme u objektu prijave

Na slici 35. prikazana je inicijalizacija polja data objekta tipa klase *Application*, podacima iz forme. Na taj način su podaci koje je korisnik uneo u prijavi za javni poziv sačuvani u objektu prijave.

**Šesti problem** sa kojim sam se susrela tokom realizacije ove aplikacije bio je **uploadovanje dokumenata i čuvanje istih u bazi**. Korisnik je u okviru prijave na javni poziv mogao da priloži sve tražene dokumente. Te dokumente je potrebno čuvati negde u backend delu a zatim ih i prikazati administratoru u okviru pregleda prijava. Ovaj problem je rešen korišćenjem *Multer* middleware-a za Node.js, konkretno njegovog *DiskStorage*-a.

```
import multer from 'multer';

const diskStorage = multer.diskStorage({
  destination: 'D:/publicCallsApp/backend/uploads',
  filename: (req, file, cb) => {
    const mimeType = file.mimetype.split('/');
    const fileType = mimeType[1];
    const fileName = file.originalname;
    //console.log(fileName);
    cb(null, fileName);
  },
});

const storage = multer({ storage: diskStorage });
```

Slika 36. – Uploadovanje fajlova pomoću *multer storage*-a

U backend delu sam napravila folder *uploads* koji sam koristila kao destinacioni folder za sve fajlove, *Multer* middleware presreće svaki Request objekat koji u svom telu sadrži neki fajl i skladišti ga u *uploads* folderu, dok se sam naziv fajla čuva u bazi.

```
1 import express from 'express';
2 import storage from '../helper/storage';
3
4 import { FilesController } from '../controllers/files.controller';
5 const fileRouter = express.Router();
6
7 fileRouter.post('/upload', storage.array(['file']),
8 (req, res) => new FilesController().uploadFile(req, res))
9
```

Slika 37. – Uključivanje *multer middleware*-a u *file/uploads* rutu



```

</mat-expansion-panel-header>
<table style="justify-content: center;" cellspacing="8">
  <tr *ngFor='let item of app.labels;let k=index'>
    <td style="font-weight: bolder; font-size:15px">{{item}}:</td>
    <td style="color: blueviolet; font-size:15px">{{values[i].displayArr[k]]}</td>
    <td *ngIf="filePropertyFlags[i*app.labels.length + k]"><a
      href="{{uploadsURI}}{{values[i].displayArr[k]]}" target="_blank">
        <mat-icon>attach_file</mat-icon>
    </a></td>
  </tr>
</table>

```

Slika 38. - Prikazivanje dokumenata administratoru

Dokumente sam prikazivala administratoru tako što sam href atribut linka postavila na vrednost `http://localhost:4000/uploads/ naziv dokumenta iz baze` i omogućila otvaranje linka u novom prozoru. Podržane ekstenzije fajlova u sistemu su: jpg, png, pdf, doc, docx, xlsx, txt.

**Sedmi problem** tokom realizacije aplikacije bio je **korišćenje grafikona i mogućnost eksportovanja tabela izveštaja**. Grafikoni za prikaz statistike su korišćeni iz paketa `ngx-charts`, međutim naišla sam na problem tokom stilizovanja istih, pošto je pozadina aplikacije tamna trebalo je podesiti da oznake pojedinačnih grafikona na x osi budu vidljive kao i legenda. Zatim podešavanje layouta i veličine grafikona.

```

<div class="chart-container">
  <div class="spectre-bar-chart">
<ngx-charts-bar-vertical

  [view]="[800,200]"
  [results]="data"

  [xAxisLabel]="Naucno polje"

  [yAxisLabel]="Broj prijava"
  [legend]="true"
  [showXAxisLabel]="true"
  [showYAxisLabel]="true"
  [xAxis]="true"
  [yAxis]="true"
  [gradient]="true"
  style="fill: white;"
</ngx-charts-bar-vertical>
</div>
</div>

:host {
  height: 100%;
  width: 100%;
}

.chart-container {
  padding-left: 10px;
  padding-right: 10px;
}

.spectre-bar-chart {
  height: 200px;
  width: 800px;
  margin: 5% auto;
}

```

Slika 39. – Podešavanje stila grafikona

**Osmi problem** je bio problem oko **eksportovanja tabele u xlsx, json, cvs i txt fajlove**. Za eksportovanje tabela korišćena je komponenta *MatTableExporter* međutim postojao je problem kada sam za prikazivanje 3 izveštaja koristila ngIf da bi u zavisnosti od odabranog radio dugmeta prikazala određeni izveštaj. Prvobitno sam 3 izveštaja prikazala u 3 različita div taga i u svakom div tagu koristila sam ngIf da bih kontrolisala koji mi se div prikazuje. Međutim problem je bio taj što MatTableExporter ne radi lepo, tj. ne eksportuje tabele ukoliko se nalazi u nekom div-u koji je uslovljen. Problem sam rešila tako što sam za svaki izveštaj napravila posebnu komponentu i te komponente ugradila u jednu roditeljsku komponentu. Nakon toga je MatTableExporter normalno radio. Implementacija je prikazana na slici 40.

```
frontend > pCallsApp > src > app > reports > <> reports.component.html > app-report1
1  <br>
2  <mat-radio-group [(ngModel)]="selected">
3    <mat-radio-button *ngFor="let report of reports" [value]="report">
4      <span style="color: ■white;"> {{report}} </span> &nbsp; &nbsp; &nbsp;
5    </mat-radio-button>
6  </mat-radio-group>
7  <br>
8  <app-report1 *ngIf="selected=='Prijava po pozivu'">
9
10  </app-report1>
11
12  <app-report2 *ngIf="selected=='Prijava po naučnom polju'">
13
14  </app-report2>
15
16  <app-report3 *ngIf="selected=='Prijava po instituciji'">
17
18  </app-report3>
```

*Slika 40. – Realizacija roditeljske komponente izveštaja*

```
<div style="justify-content: center; align-items: center; text-align: center;">
  <button mat-raised-button
    (click)="exporter.exportTable('xlsx', {fileName:'test', sheet: 'sheet_name', Props: {Author: 'Talha'}})">Excel</button>
  <button mat-raised-button (click)="exporter.exportTable('csv')">Csv</button>
  <button mat-raised-button (click)="exporter.exportTable('json')">Json</button>
  <button mat-raised-button (click)="exporter.exportTable('txt')">Txt</button>
</div>
```

*Slika 41. – Realizacija eksportovanja tabele u različite tipove fajlova*

# ZAKLJUČAK

Ova aplikacija je napravljena s ciljem da se realizuju javni pozivi za naučne projekte na efikasan i pregledan način, nasuprot realizaciji koju imamo prilike da često vidamo u praksi. Većina prijava na javne pozive na sajtovima institucija koje ih objavljuju je realizovana tako što se korisnicima ostavlja gomila word obrazaca koja treba da se preuzme popuni i pošalje na adresu, takođe je uglavnom priložena gomila dokumenata o samoj proceduri slanja koju podnosilac prijave treba da pročita. Takav način realizacije javnih poziva meni deluje vrlo haotično i neorganizovano i loše i po aplikante i po ljude koji iste prijave pregledaju. Ova aplikacija pruža nekakav red i organizovanost prijavljivanja na javne pozive. Značajno je to što korisnik ne treba da preuzima nikakve obrasce već popunjava sve unutar aplikacije, pozivi targetiraju određene korisničke grupe, tako da ne postoji mogućnost da se javni pozivi za naučne projekte iz oblasti tehničkih nauka prikazuju korisniku koji se bavi medicinom. Jednostavno je postignuto da poziv stiže uvek u prave ruke. Ovim dokumentom opisan je detaljno proces izrade takve aplikacije, od korisničkih zahteva, korišćenih tehnologija, poput *Angulara*, *Node.js-a*, *Express-a* i *MongoDB-a*, zatim je opisan i deo implementacije najizazovnijih problema i detaljno opisan rad celog sistema. Međutim ni ova aplikacija nije savršena i postoji bezbroj poboljšanja koja bi bilo poželjno realizovati. Jedno od osnovnih poboljšanja bi bilo rad na dizajnu, dizajnirati ovu aplikaciju tako da zadovoljava sve moderne standarde veb dizajniranja, korišćenje nekih bogatijih UI frameworka i komponenti, izbeći tzv. školski dizajn. Takođe mogle bi se realizovati još neke funkcionalnosti koje bi dodatno olakšale rad sa javnim pozivima svim korisnicima. Neke funkcionalnosti su se mogle realizovati i na jednostavniji način bez mnogo komplikovanja. Od funkcionalnosti bih možda dodala i da aplikantima stižu obaveštenja na mejl koji su ostavili pri registraciji, o statusu njihove prijave. Takođe da se i javni pozivi reklamiraju odgovarajućoj target grupi putem mejla, kao neki vid promocije. Kod notifikacija bih realizovala još i bedževe sa brojem nepročitanih poruka. Izmenila bih sam dizajn aplikacije pogotovo za pregled prijava, javnih poziva...

# LITERATURA

- [1] <https://angular.io/>
- [2] <https://material.angular.io/>
- [3] <https://nodejs.org/en/>
- [4] <https://www.mongodb.com/>
- [5] <https://mongoosejs.com/>
- [6] <http://stackoverflow.com/>
- [7] <https://www.htmlelements.com/docs/angular-dynamic-forms/>
- [8] <https://developer.mozilla.org/en-US/>
- [9] <https://youtube.com/playlist?list=PLlilGF-RfqbZMNtaOXJQiDebNXjVapWPZ>
- [10] <https://www.npmjs.com/>
- [11] Mike Cantelon, Marc Harter, T.J. Holowaychuk, Nathan Rajlich (2013): Node.js in Action
- [12] Elad Elrom (2016): Pro MEAN Stack Development
- [13] McCalay Nicholas (2017): MEAN Cookbook: The meanest set of MEAN stack solutions around
- [14] David Flanagan (1996): Speaking Java Script
- [15] Steve Souders (2007): High Performance Web Sites