Razvoj sustava za upravljanje završnim i diplomskim radovima

Bosak, Kris

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

KRIS BOSAK

RAZVOJ SUSTAVA ZA PRIJAVU ZAVRŠNIH I DIPLOMSKIH RADOVA

Završni rad

Pula, rujan, 2019. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

KRIS BOSAK

RAZVOJ SUSTAVA ZA PRIJAVU ZAVRŠNIH I DIPLOMSKIH RADOVA

Završni rad

JMBAG: 0303063314, redoviti student

Studijski smjer: Informatika

Predmet: Programsko inžinjerstvo

Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan, 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

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

Student
Kris Bosak

U Puli, rujan, 2019. godine

IZJAVA
o korištenju autorskog djela

Ja, Kris Bosak dajem odobrenje Sveučilištu Jurja Dobrile
u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Razvoj sustava za prijavu završnih i diplomskih radova“koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 18. rujan 2019. godine

Potpis
Kris Bosak

Sadržaj

1. Uvod

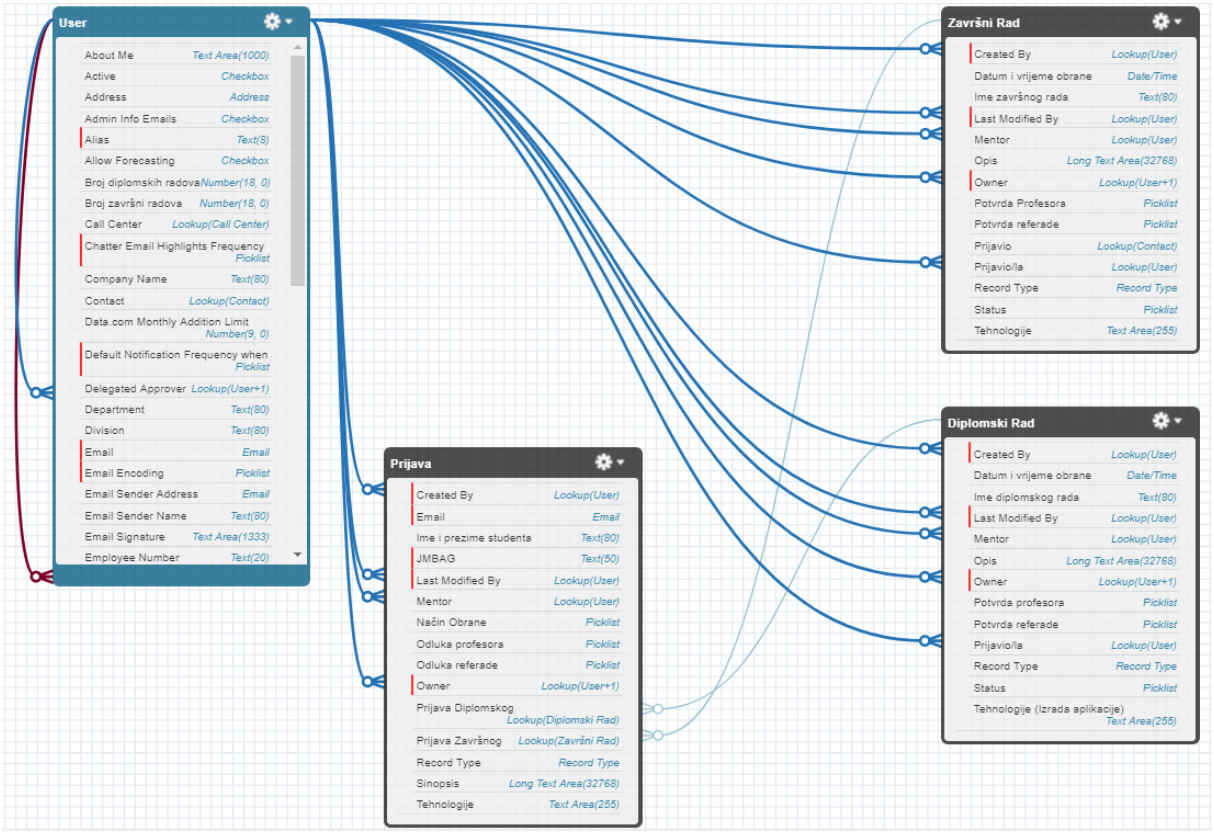
U ovom radu se bavim repliciranjem i u nekim segmentima poboljšanjem trenutnog sustava za prijavu završnih i diplomskih radova. Cilj mi je da u samo nekoliko klikova student može prijaviti svoj rad za razliku od sustava koji je trenutno aktivan u kojem je taj proces iscrpljujuć i za studente, ali i za većinu profesora također. Naime, trenutni proces zahtjeva od studenta da provede gotovi cijeli dan hodajući od vrata do vrata da bi dobio određeni papir te isti onda treba i potvrditi, a kod svakih vrata bi studenta dočekao veliki red ljudi koji rade istu stvar. Profesori s druge strane moraju prihvatiti i poslušati svakog od tih studenata što je također iscrpljujuće na svoj način.

Za izradu ove aplikacije odlučio sam se za „Salesforce“ platformu. Salesforce se bazira na radu u oblaku što je bio jedan od razloga zbog kojeg sam i izabrao baš tu platformu jer nema dodatnog spajanja na eksterni server kako bi se spojila baza podataka koja se napravi. „Salesforce“ radi kao svojevrsni server na kojemu se mogu pohranjivati podaci. Jedan drugi razlog zbog kojeg sam izabrao navedenu platformu je taj što sama platforma nudi konstantan dizajn što dodatno omogućuje korisniku da se lakše snalazi, ali i da se lakše privikne na sustav. „Salesforce Community Cloud“ je socialna platforma na kojoj sam radio prikaz koji će vidjeti krajnji korisnici, a to su profesori i studenti. Community nam omogućava da interakcija između korisnika prođe nesmetano. Korisnici mogu komunicirati jedni sa drugima, ali i sudjelovati u aktivnostima u kojima sudjeluju drugi korisnici. Upravo to omogućuje prikaz i prijavu završnih i/ili diplomskih radova te izradu i prijave na iste. Prednosti community-a su te što ima već napravljene predloške po kojima je lakše početi raditi svoje što se već treba napraviti. Kostumizacija se vrši kroz ugradnju CSS-a u isti community te HTML isječke. Iduća prednost bi bila što je direktno povezana sa Salesforce-ovim cloudom s kojeg odma može povući podatke kada je to potrebno.

Nedostatak je taj što uz puno stvari koje dolaze „Out of the box“ dolaze i stvari koje se ili ne mogu ili se jako teško mogu kostumizirati pa ćak i maknuti iz sučelja. Razvojno okruženje za koje sam se odlučio je „Visual Studio Code“. Za navedeno okruženje sam se odlučio iz razloga što je isti lagan za sustav na kojemu se radi te se jako lagano integrira sa Salesforce-om. Jedan nastavak u („Salesforce Extension Pack“) razvojno okruženje te instalacija naredbenog sučelja („Salesforce CLI“) i sve je spremno. Putem nekoliko klikova se VSC spoji sa Salesforce-om i spremni smo za programiranje. Sada ću u nekoliko idućih poglavlja u kratko pokazati i objasniti aplikaciju.

2. Razrada funkcionalnosti

2.1. Salesforce Schema

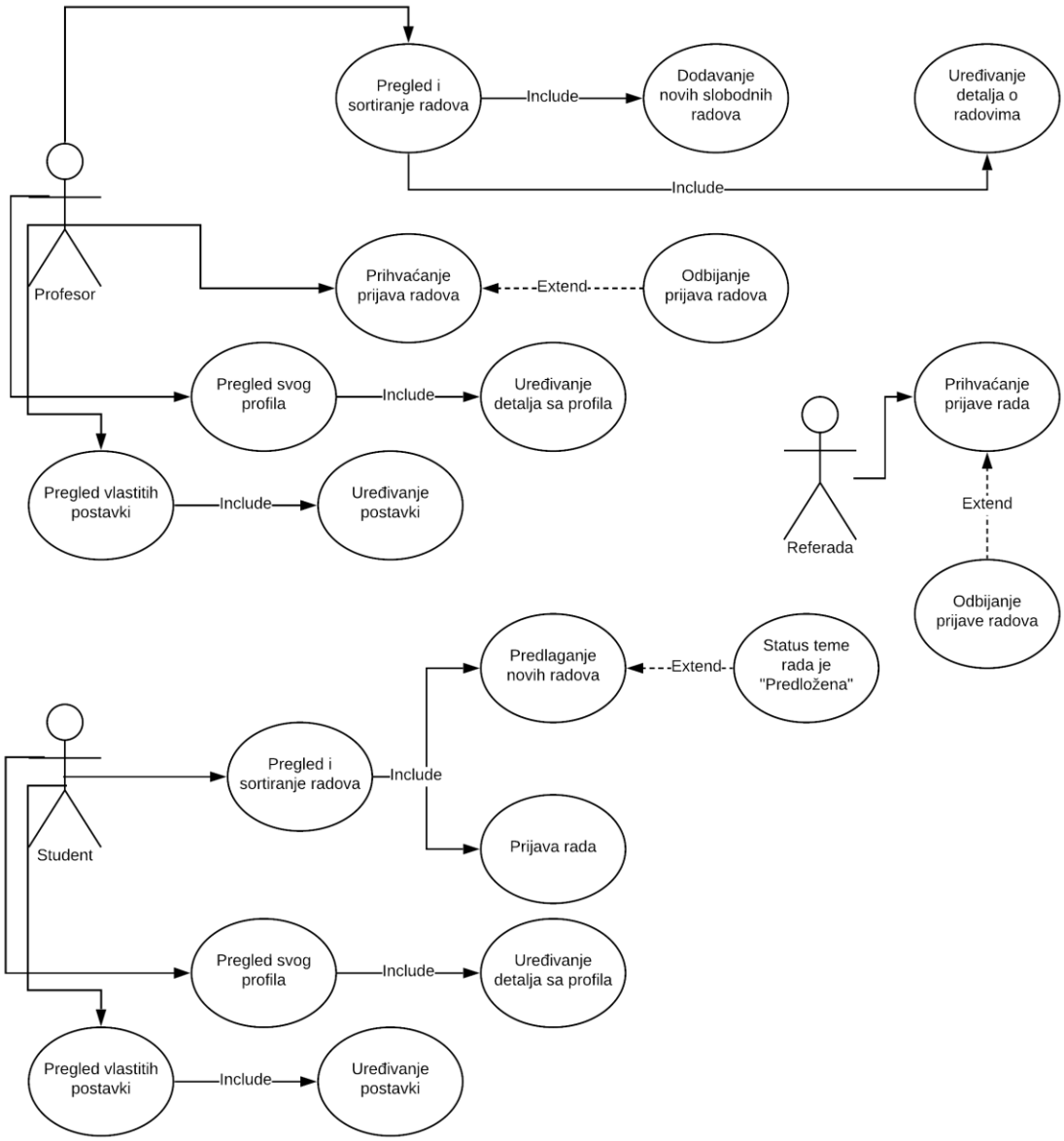


Slika 1 – Salesforce Schema aplikacije

Navedena slika je prikaz autogenerirane Salesforce scheme koja predstavlja svojevrsni klasni dijagram. Klase kao modeli u Salesforce aplikacijama nisu prisutne zbog toga što Salesforce ne koristi standardan pristup razvoju aplikacija. Na navedenoj platformi se koriste standardi objekti koji predstavljaju objekte koji su uvijek na platformi pod istim imenom. Na ovoj slici je to standardni objekt „User“. Svaki taj standardi objekt ima svoja standardna polja, a u ovom primjeru su to sva polja osim „Broj diplomskih radova“ te „Broj zavrsni radova“ kojima ćemo funkciju saznati kasnije. Useri imaju unikatan „Alias“, „Email“ „Id“ i slična polja kako bi se mogli razaznati te kako bi ih bilo moguće lakše pretraživati SOQL(„Salesforce Object Query Language“) upitima. Prilagođeni objekti s druge strane su objekti koje radi administrator ili developer na samoj platformi. Ti objekti u sustavu imaju nastavak „__c“ na svoje ime kako bi se mogli razlikovati od standardnih objekata i kako bi sustav znao o kakvom objektu se radi. Isto tako polja koja su napravljena od strane administratora ili developera imaju oznaku „__c“ iza imena iz istog razloga. Uzmimo primjer polja „Owner“ koje je standardno polje i polja „Mentor“ koje je prilagođeno polje. „Owner“ će u sustavu biti

prepoznat kao „Owner“ dok će prilagođeno polje biti prepoznato kao „Mentor__c“. Način na koji su povezani ovi objekti je putem „lookup“ veza koje rade na način da se na jednom objektu mogu pretraživati vrijednosti drugog objekta. Kada pogledamo schemu možemo primjetiti kako je „User“ objekt „roditelj“ objektima „Završni Rad“, „Diplomski Rad“ i „Prijava“. Dok su navedeni objekti „djeca“ koji kroz polja kojima je tip „Lookup“(poput polja „Mentor“ na prilagođenom objektu „Završni rad“) može doći do vrijednosti iz roditelja. Možemo primjetiti kako tih polja sa tipom Lookup ima više. To je tako jer te vrijednosti se rijetko kad koriste za istu stvar pa je tako potrebno više lookup veza. Njih može biti više na prema puno drugih objekata.

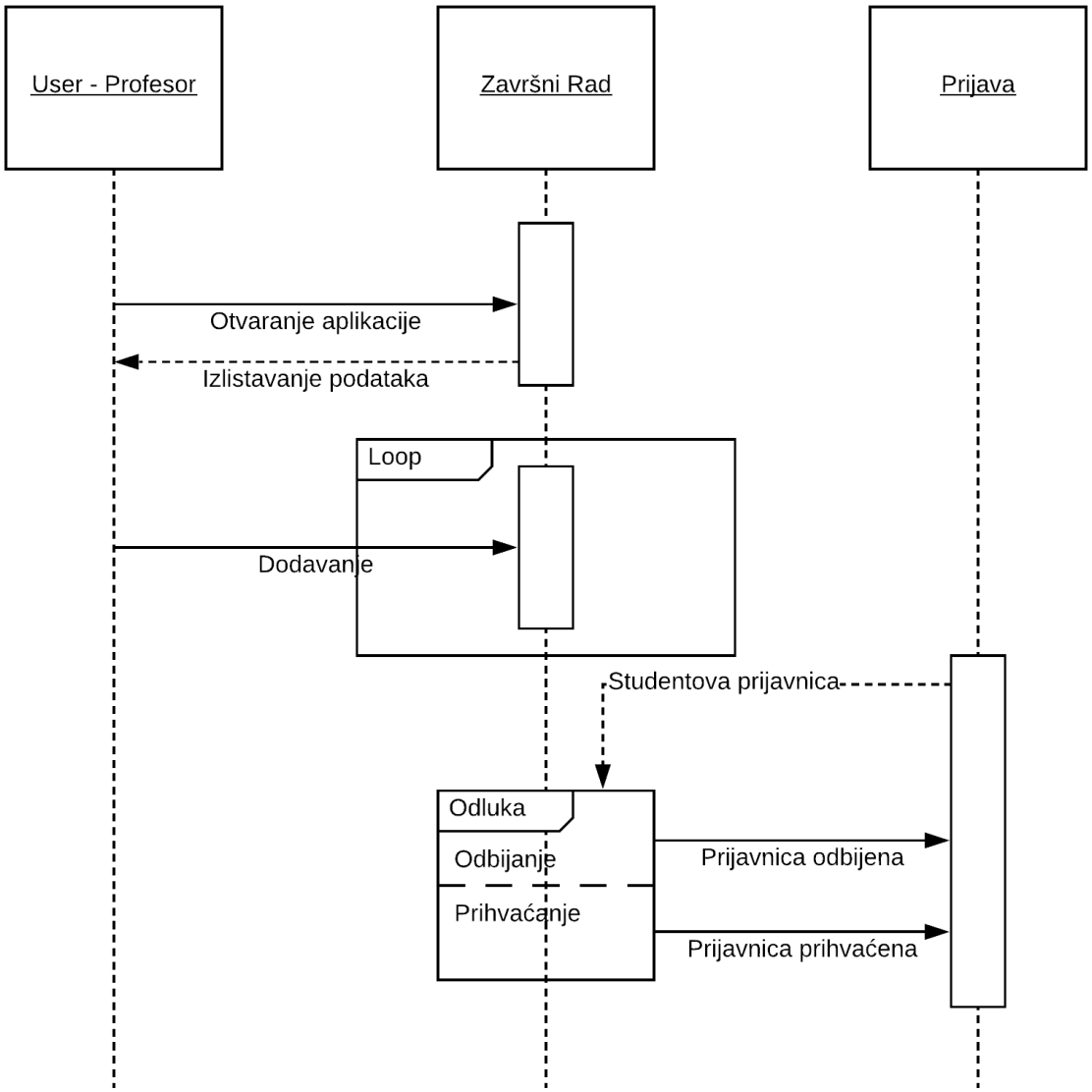
2.2. Use Case dijagram



Slika 2 – Use Case Dijagram

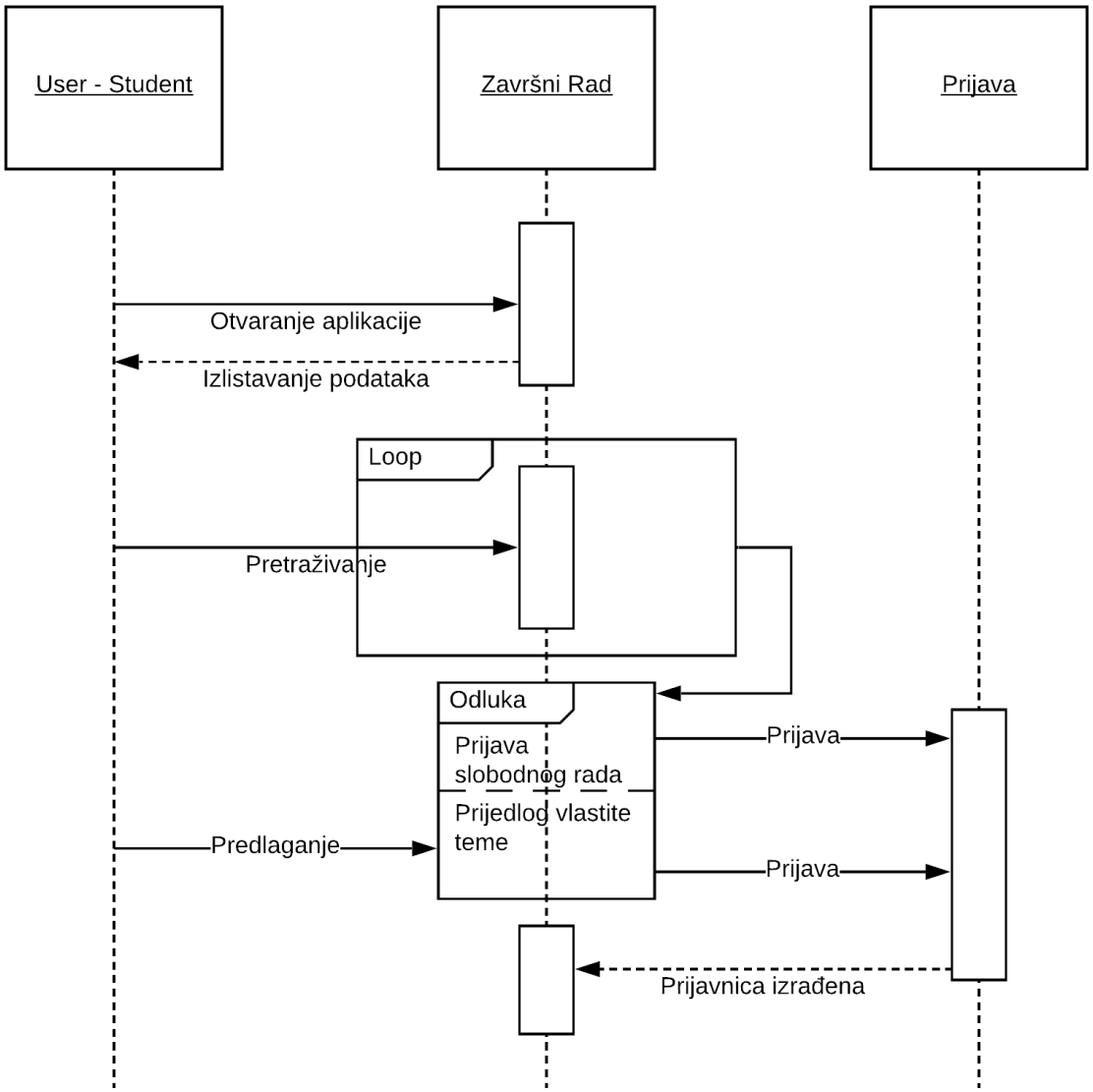
Na prikazanom dijagramu možemo vidjeti kako je aplikacija smišljena da se najčešće koristi. Profesori i studenti imaju pravo mjenjati svoje postavke te izmjenjivati podatke koji su dostupni na njihovom profilu. Oboje također mogu i pregledati te sortirati radove po svojoj volji. No kada je riječ o stvaranju novih radova onda je tu stvar nešto drugačija. Profesori stvaranju slobodne radove dok studenti stvaraju teme koje imaju status predložene teme. Na studentima je da prijave svoju temu, a na profesorima i referadi da istu prihvate ili odbiju.

2.3. Sekvencijalni dijagram



Slika 3 – Sekvencijalni Dijagram za profesora

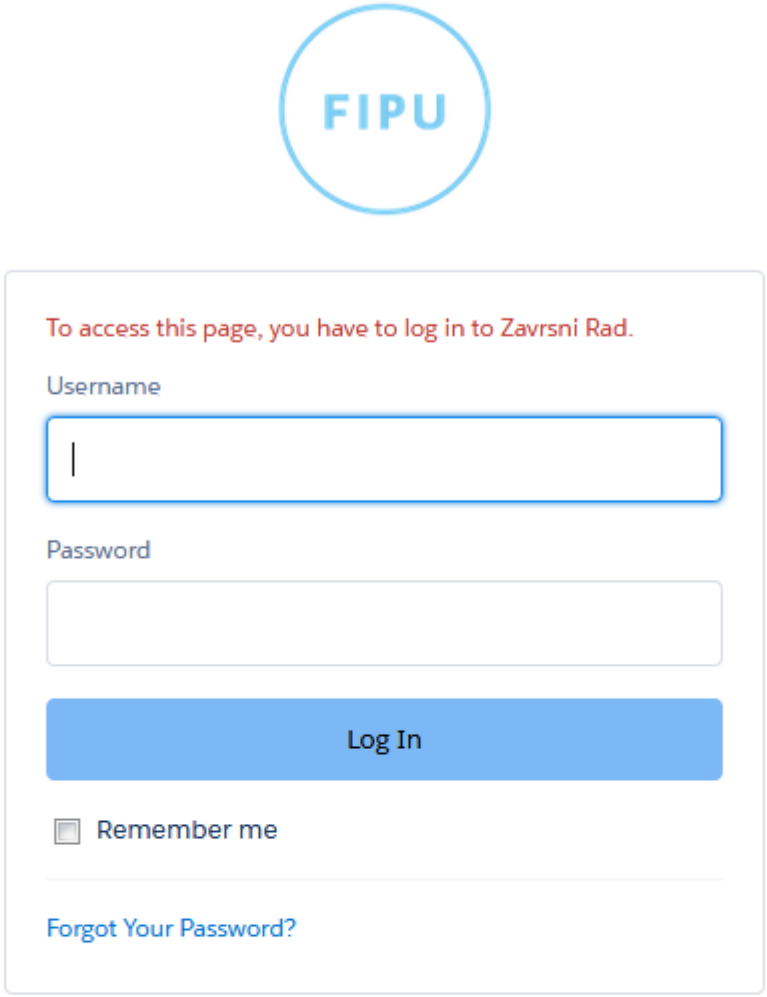
Dijagram na slici nam prikazuje kako bi jedan klasičan tok događaja mogao izgledati kada aplikaciju koristi neki profesor. Prilikom otvaranja aplikacije izlistani su nam svi završni radovi. Uzevši situaciju kada profesor mora dodavati radove za koje smatra da su dostojni završnog ili diplomskog rada onda će ih nakon izlistavanja postojećih, ukoliko ih ima, dodati nove. Njegov posao je tu gotov sve dok ne počnu dolaziti prijavnice od studenata na radove na kojima je on mentor. Tada je odluka prihvatiti tu prijavnicu ili ne.



Slika 4 – Sekvencijalni Dijagram za studenta

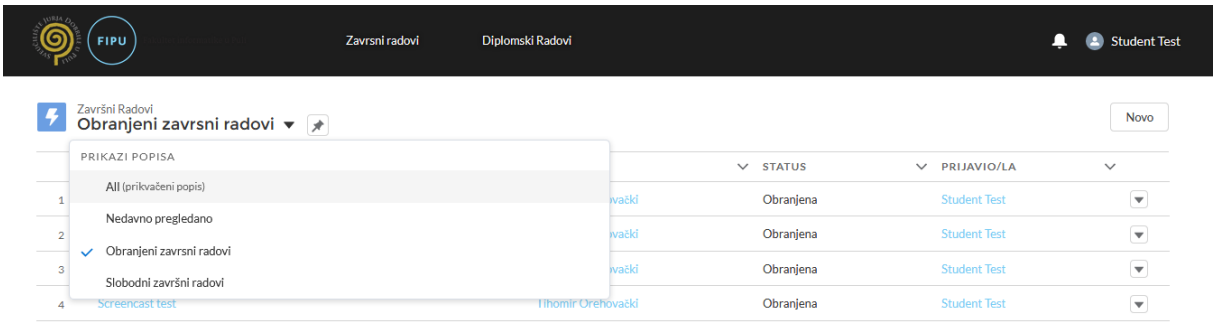
Studentov sekvencijalni dijagram je nešto drugačiji jer student nakon što mu se izlistaju radovi kod otvaranja aplikacije on treba pretraživati radove u nadi da će pronaći nešto što bi mu odgovaralo. Nakon dovoljno pretraživanje suočava se sa odlukom prijave već postojećeg, slobodnog rada ili će se pak odlučiti na predlaganje vlastite teme za završni ili diplomski rad. Nakon što je prijavnica izrađena i poslana na prihvaćanje. Studentu samo preostaje čekati na povratnu informaciju.

3. Prikaz aplikacije



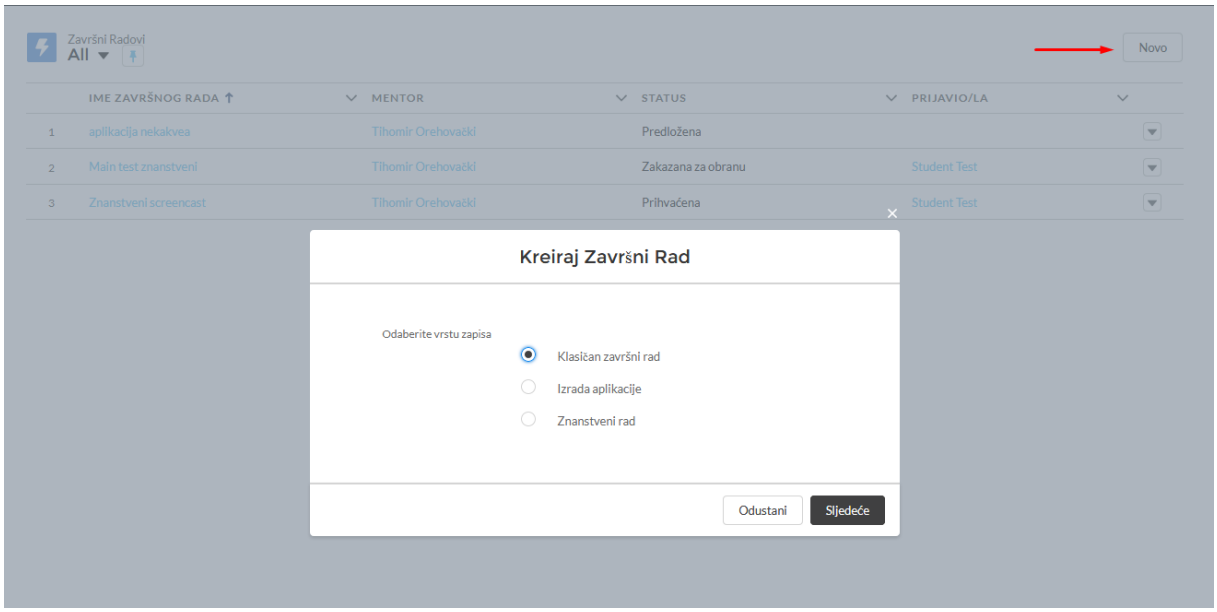
Slika 5 – Prijava u aplikaciju

Na slici možemo vidjeti klasičan ekran za prijavu na kojemu se vrši autentifikacija korisnika. Unosom pravilnog email-a i pripadajuće lozinke korisnik će uči u sustav. Označavanjem kučice „Remember me“ korisnikovi podaci će biti spremljeni kako isti nebi trebao ponovo unositi iste podatke prilikom sljedeće prijave.



Slika 6 – Prikaz Završnih radova

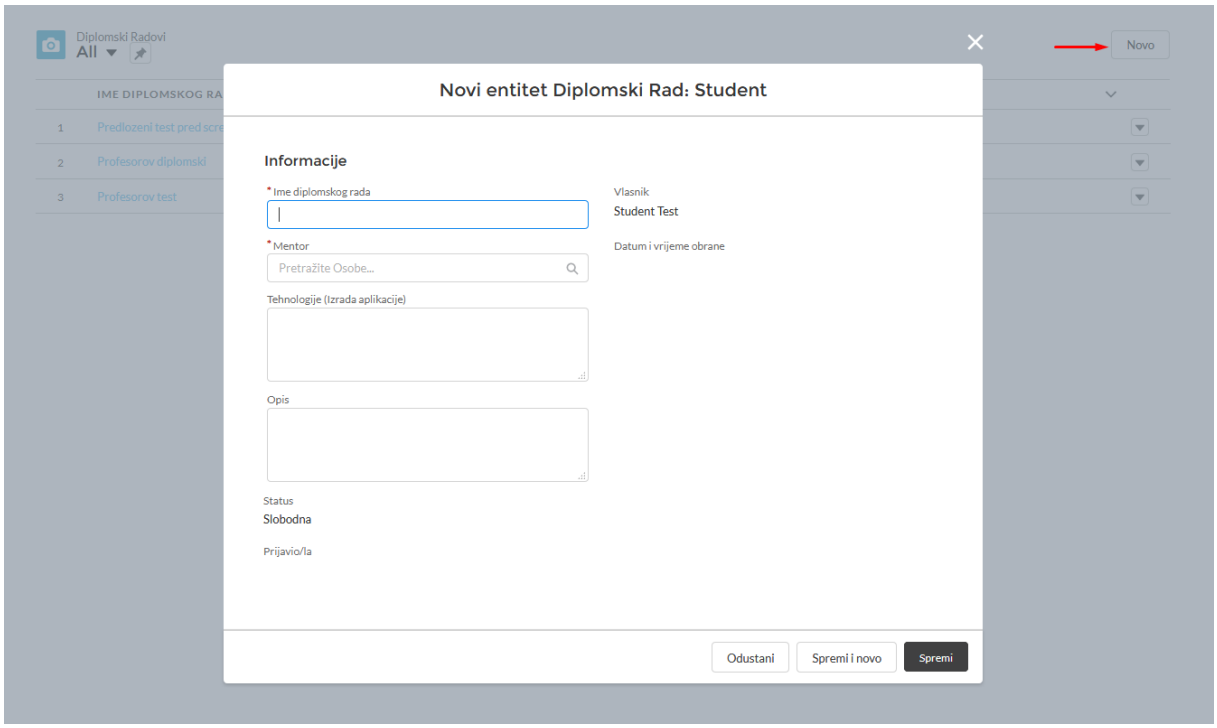
Nakon prijave predstavljeni smo sa ekranom kao na slici 6. Trenutno su na slici prikazani „Obranjeni završni radovi“, ali klikom na strelicu pored tog naziva možemo doći do izbornika koji nam nudi da biramo između više popisa kao što su nedavno pregledani i slobodno radovi. Ista situacija je i kod prikaza diplomskih radova.



Slika 7 – Prikaz prozora koji se otvara okidanjem gumba „Novo“

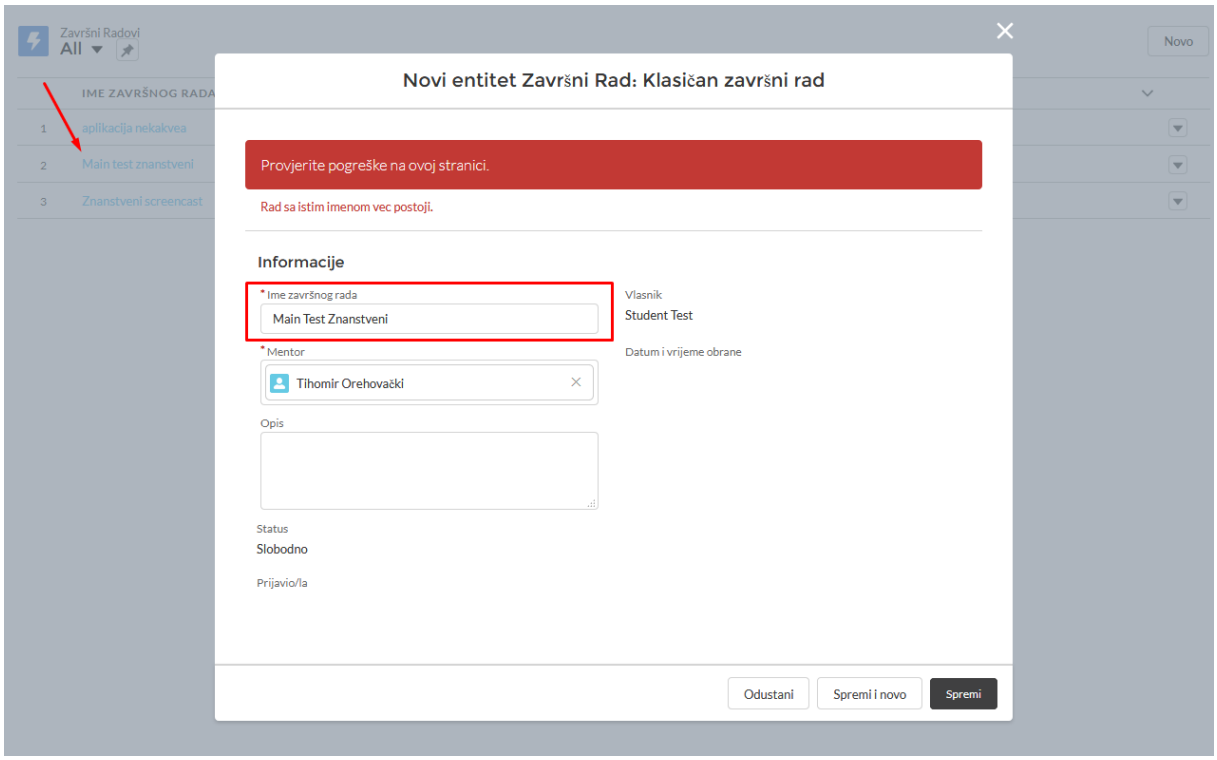
Pritiskom na gumb „Novo“ prikazuje nam se prozor u kojemu možemo izabrati jedno od tri načina kreiranja teme završnog rada.

Kada isti kreira profesor onda je rad napravljen sa statusom „Slobodan“ dok kad istu funkciju izvrši student onda će ta tema biti napravljena sa statusom „Predložena“. Polja za unos koja se prikažu nakon odabira izrade aplikacije nisu ista kao kad bi izabrali klasičan završni rad ili znanstveni rad. Izrada aplikacije ima ponuđene stvari koji su specifične za takav način izrade završnog rada, a to je polje poput „Tehnologije“ u kojima će se taj rad odnosno aplikacija izrađivati. Situacija kod izrade novih diplomskih radova nije ista jer diplomski radovi u ovom trenutku još nemaju mogućnost biranja između više načina izrade pa u tom slučaju se otvara sljedeći prozor:



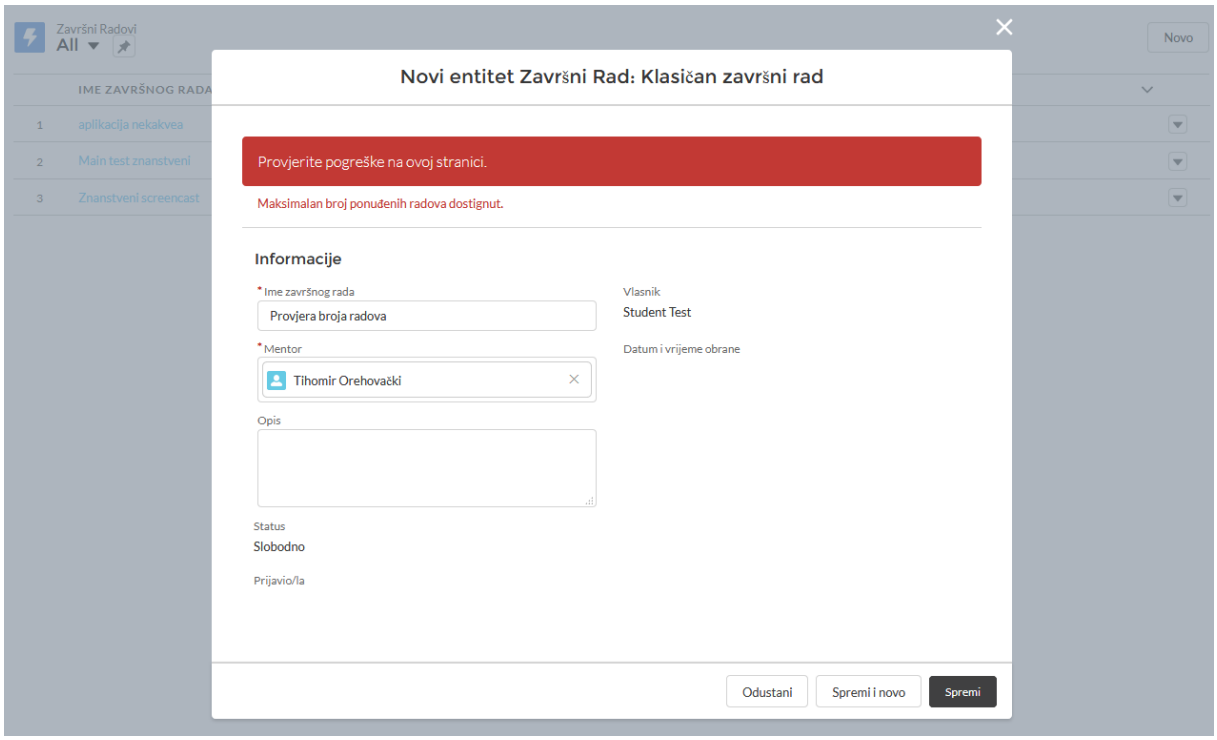
Slika 8 – Prozor koji se otvara okidanjem gumba „Novo“ na objektu „Diplomski Radovi“

Već ovdje možemo primjetiti polje „Mentor“ o kojemu smo pričali na početku. Bilo je rečeno kako je to „Lookup“ polje koje zapravo spaja standardni objekt „User“ i prilagođeni objekt „Diplomski Rad“ u ovom slučaju. Lookup polja je najlakše prepoznati po znaku povećala u pripadajućem polju jer to označuje pretraživanje, a u ovom slučaju je to pretraživanje po podacima iz drugog objekta. Kada unesemo sve potrebne podatke preostaje nam samo stisnuti spremi i rad će biti prikazan u pripadajućem popisu.



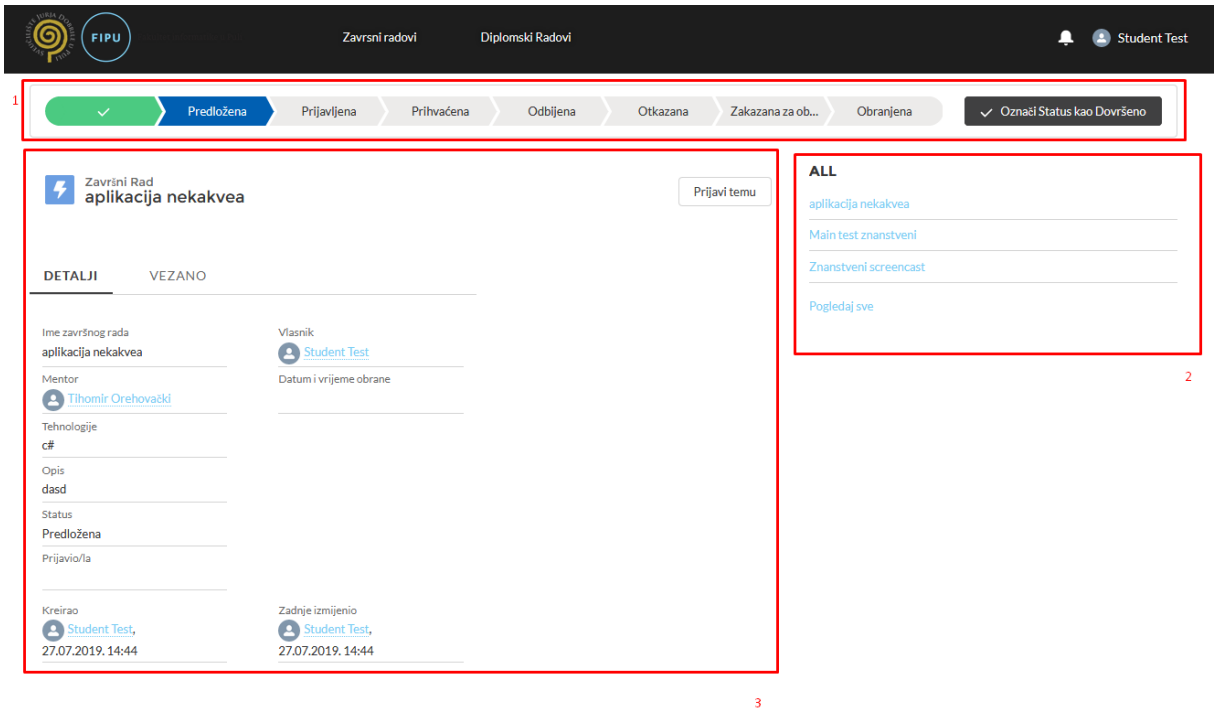
Slika 9 – Pogreška (Rad sa istim imenom već postoji)

Ukoliko kod spremanja podataka koje smo unijeli u formu za izradu završnog rada postoji već rad sa istim imenom. Onda nam sustav vraća pogrešku na tom polju koje trebamo izmjeniti.



Slika 10 – Pogreška (Maksimalan broj ponuđenih radova dostignut)

Zbog ograničenja koje mentori imaju na broj radova koje mogu imati u jednoj akademskoj godini bilo je potrebno postaviti ograničenje koje će sprječavati da jedan mentor ima 90% studenata, a neki drugi profesori dijele ostalih 10%. U mojoj aplikaciji to ograničenje je postavljeno na 10 završnih radova te 10 diplomskih radova po profesoru.

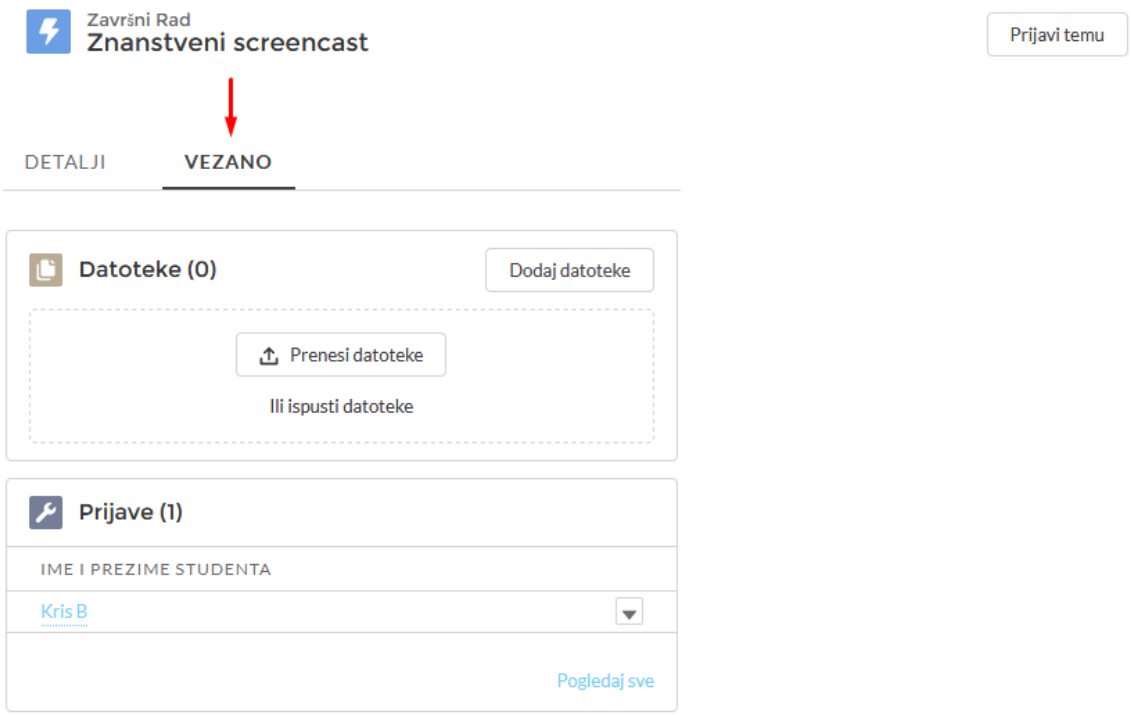


Slika 11 – Prikaz detalja Završnog rada

Kada iz popisa završnih ili diplomskih radova odaberemo neki od ponuđenih radova doći ćemo na ekran poput ovoga na slici gore.

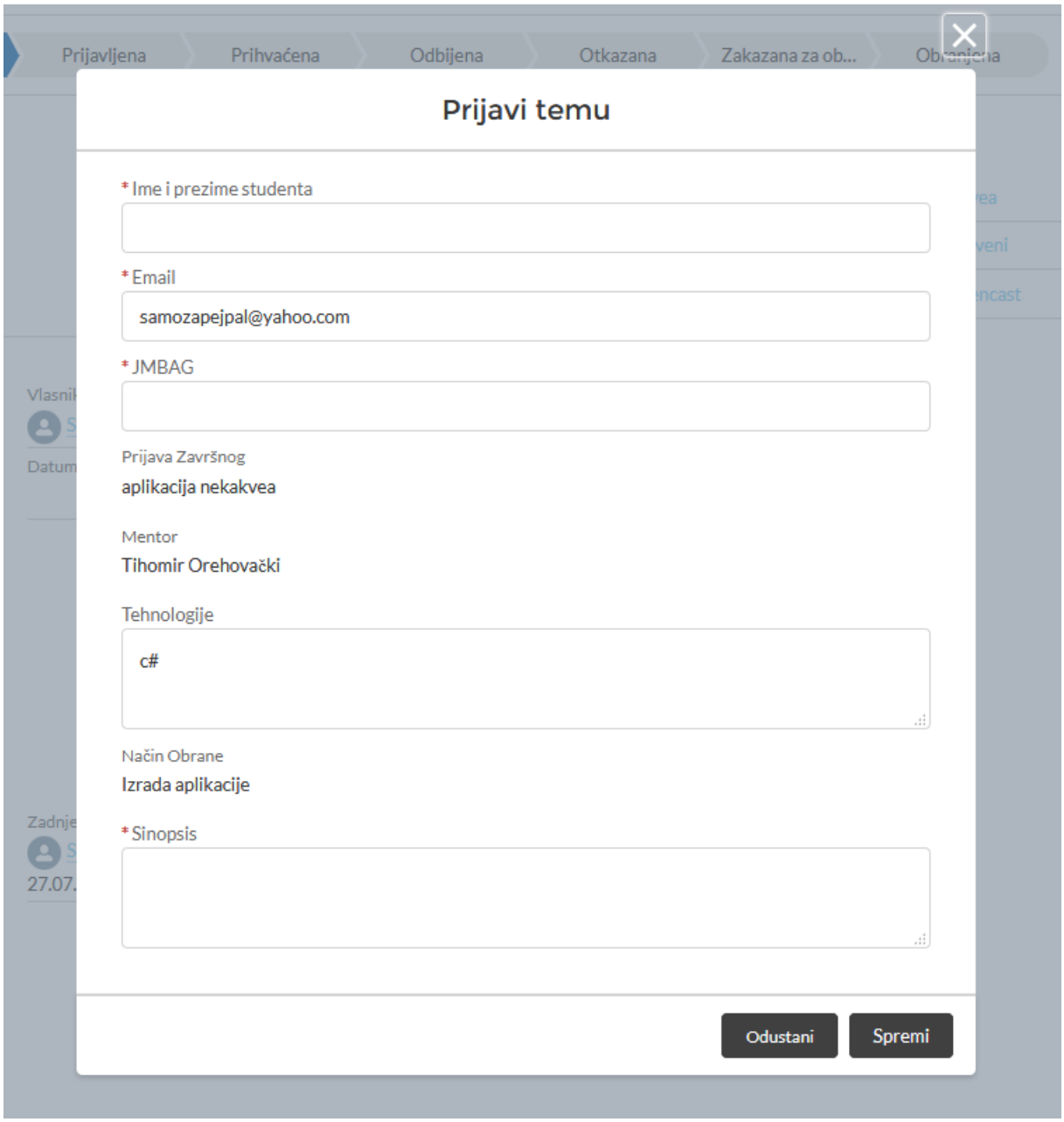
Pod brojem 1 na vrhu vidimo „Put“ koji služi kao ljepši pogled na status s kojeg ujedno možemo i mijenjati taj status ukoliko imamo dopuštenje da to možemo napraviti. U Salesforce-u se nerijetko vidi ovakav „Put“, a isto tako se većinom nalazi zajedno sa normalnim poljem „Status“ kako bi korisnik mogao birati što mu je bolje i intuitivnije. Broj 2 nam prikazuje sve radove koji nisu obranjeni kako bi mogli bez vračanja na popis radova „skočiti“ na drugi rad koji nas zanima.

Zadnja stvar na ovom ekranu, ali i ona najbitnija su sami detalji o tom radu kojeg gledamo. Studenti tu imaju samo pravo gledanja dok će profesori moći uređivati polja ukoliko je to potrebno.



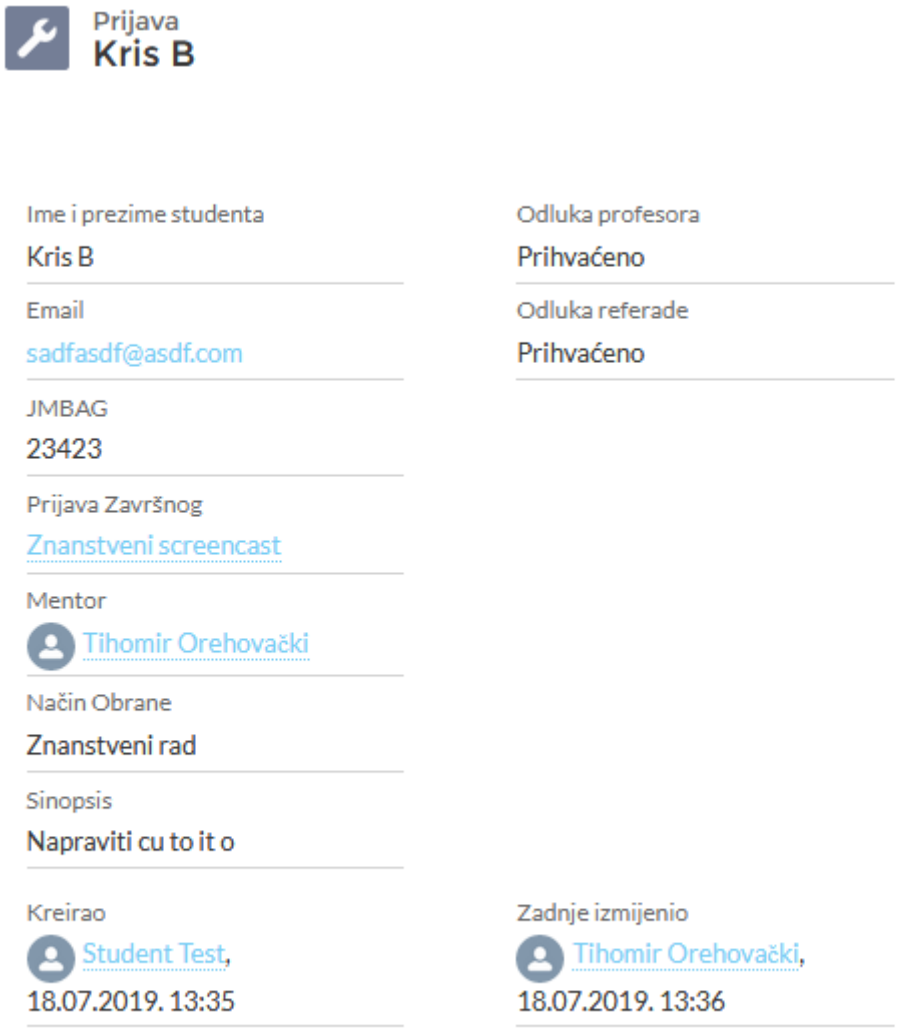
Slika 12 – Vezane stvari uz odabrani rad

Klikom na karticu „Vezano“ dolazimo do mjesta gdje možemo vidjeti prijave na taj rad te klikom na ime i prezime studenta koji je prijavio rad možemo doći do njegove ili njezine prijavnice. Pod stavkom datoteke možemo dodati svoju verziju rada kako bi taj rad bio dostupan ljudima koji će htjeti pogledati isti.



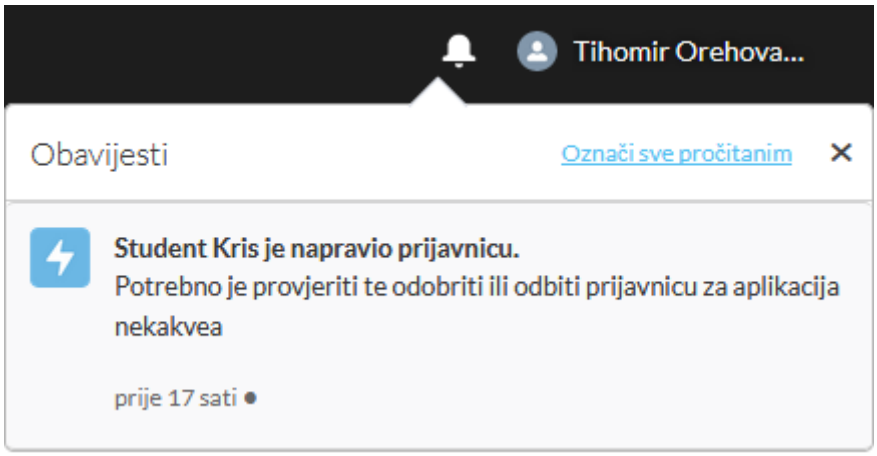
Slika 13 – Pritisak na gumb „Prijavi temu“

Pritiskom na gumb „Prijavi temu“ koji se nalazi u desnom gornjem kutu kod detalja o radu prikaže nam se ovaj prozor. U ovom prozoru unosimo sve podatke koji su potrebni referadi i mentoru.



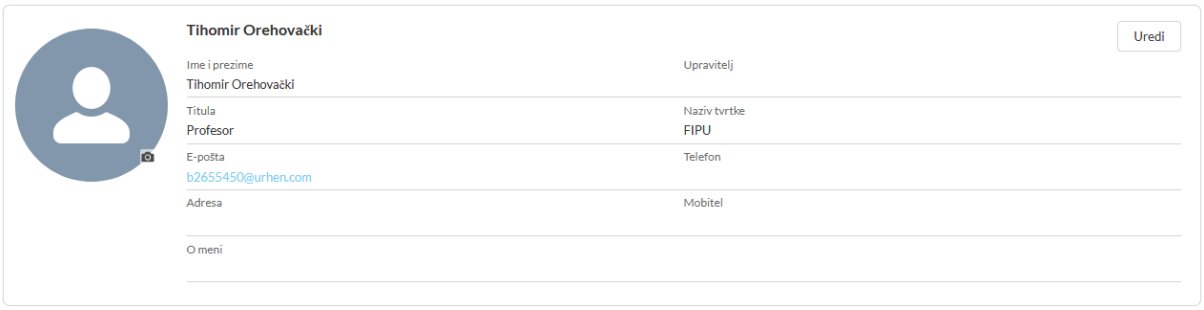
Slika 14 – Detalji prijavnice

Izradom prijavnice referada dobiva obavijest da je student prijavio neki rad. Referada u trenutnom procesu samo ima ulogu pregledavanja prijavnice i svih ostalih detalja vezanih uz studenta te donosi odluku o prihvaćanju ili odbijanju prijavnice. U slučaju prihvaćanja prijavnice profesor odnosno mentor dobiva obavijest kako je na njemu red da odluči jel prijavnica zadovoljava njegove ili njezine kriterije. Ukoliko i profesor prihvati prijavnicu. Tema se smatra prijavljena od strane studenta čija je prijavnica prihvaćena te isti može krenuti sa izradom svog rada.



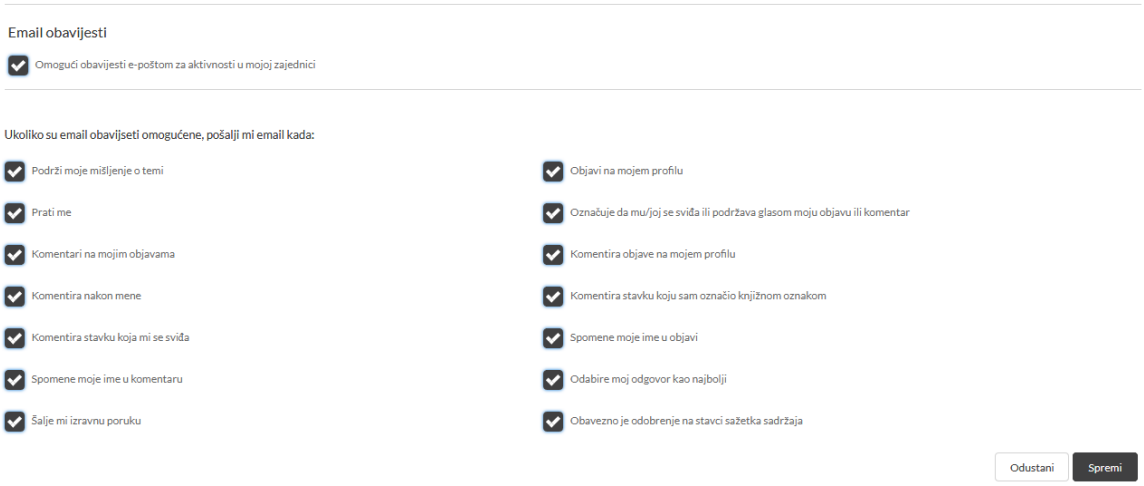
Slika 15 – Obavijest

Na slici možemo vidjeti primjer obavijesti koja se nalazi na vrhu aplikacije u desnom kutu.



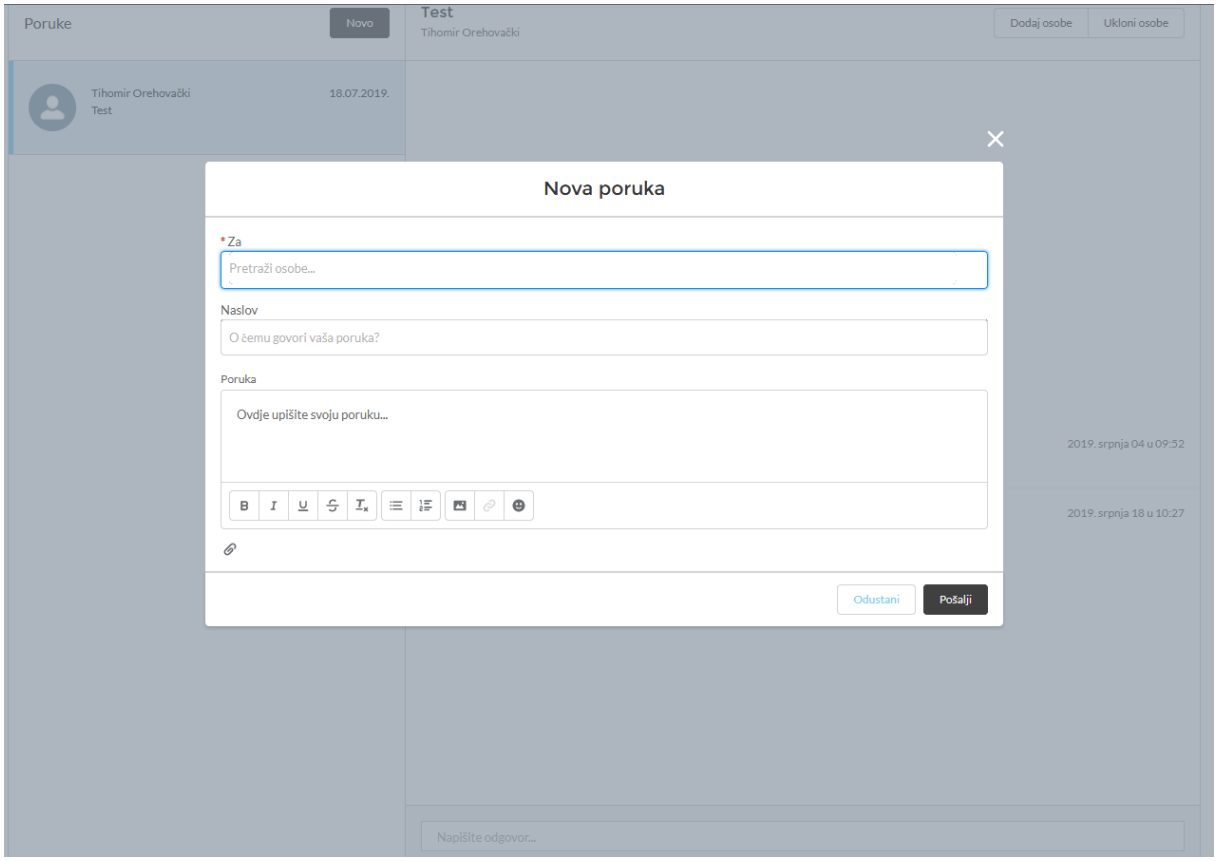
Slika 16 - Profil

Klikom na svoje ime na vrhu aplikacije prikazuje se mali izbornik u kojem je jedan od izbora „Profil“ na kojemu se mogu vidjeti neki detalji o trenutnom korisniku. Trenutni korisnik isto tako može i uređivati svoje podatke.



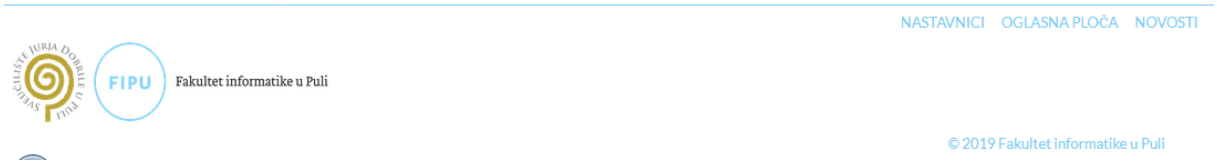
Slika 17 - Postavke

Slično kao i profil. Na postavkama imamo nešto više opcija za uređivanje i prilagođavanje, a ovo su neke od njih.



Slika 18 - Poruke

Na istom tom izborniku se nalaze i poruke koje možemo koristiti kao rješenje za komunikaciju između profesora i studenta.



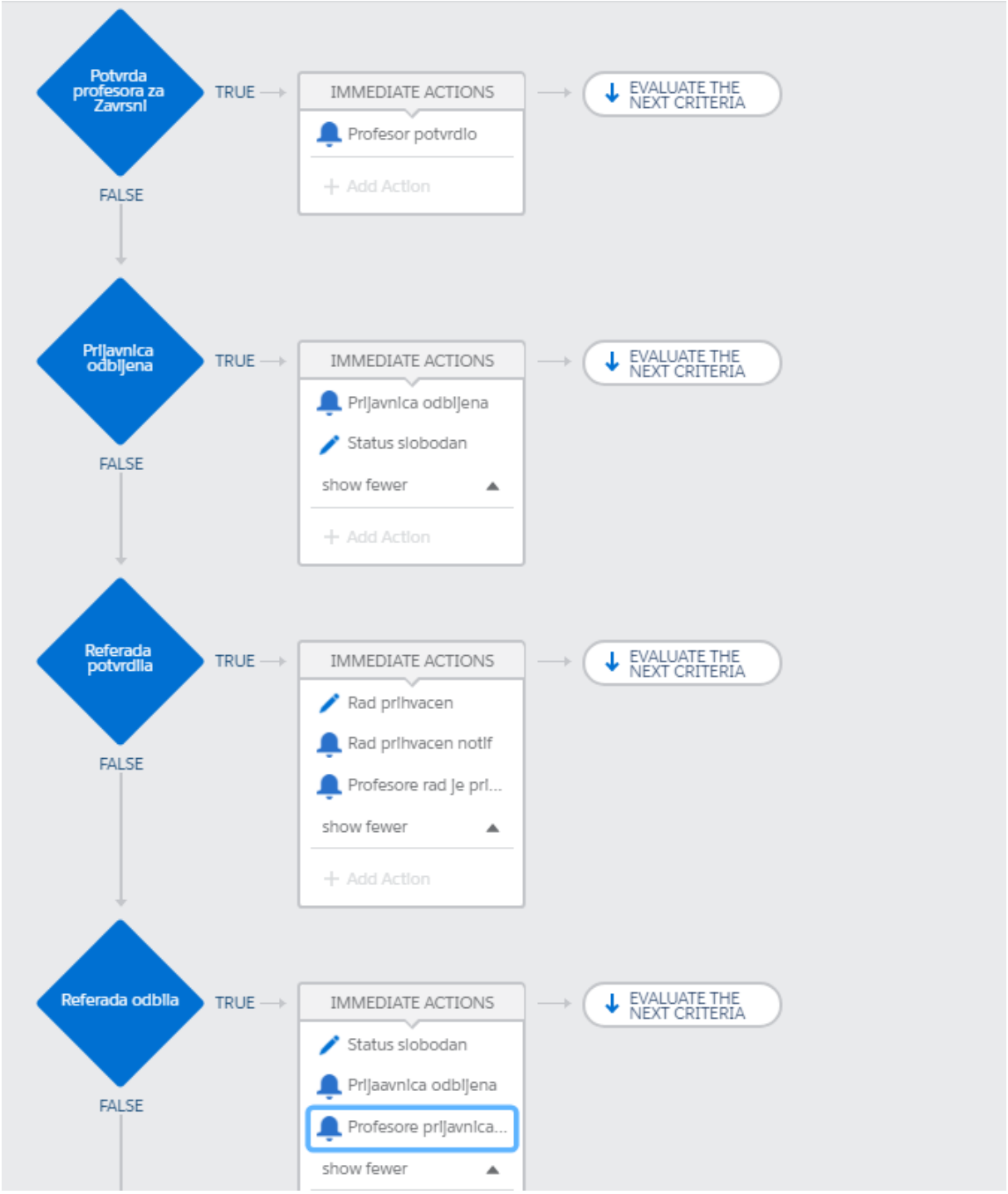
Slika 19 – Podnožje

U podnožju kao što možemo vidjeti se nalazi logo sveučilišta i fakulteta zbog prepoznatljivosti. Linkovi koji se nalaze u gornjem desnom kutu vode na stranicu fakulteta na predviđene lokacije te je sve u finoj i prepoznatljivoj svijetlo plavoj boji fakulteta.

4. Implementacija

Slika 20 – „Salesforce Process Builder“ za proces diplomskih radova

Salesforce-ov graditelj procesa je većim dijelom „Point and Click“ alat, ali kao i svugdje potrebno je malo konfiguriranja i/ili programiranja. Ovo je proces po kojem radi objekt „Diplomski Rad“, ali u ovom slučaju po istom principu je napravljen i proces za objekt završnih radova. Navedeni proces se okida kod unosa ili izmjene nad radom. Plavi rombovi označavaju kriterije po kojima se ovaj proces evaluira dok stvari koje su navedene u bijelim pravokutnicima označuju akcije koje će se desiti odma nakon što proces utvrdi da je kriterij zadovoljen. Ukoliko kriterij nije zadovoljen onda proces ide na drugi kriterij u nizu sve dok ne dođe do kraja ili dok ne dođe do onog kriterija koji zadovoljava. Kada dođe do onog koji zadovoljava onda izvršava zadane akcije te onda ukoliko je postavljena vrijednost da nakon izvršenih akcija ode na drugi kriterij onda proces ide na drugi kriterij. Isto tako ta vrijednost može biti i „STOP“, a to znači da prekida s procesom i izlazi iz njega. Akcija STOP se uvijek mora nalaziti na zadnjem kriteriju. Klikom na kriterij ili akciju desno nam se otvara jedno sučelje u kojemu možemo postaviti/konfigurirati/isprogramirati ono što želimo da se desi za tu akciju, a u ovom trenutnom sučelju koje je otvoreno možemo primjetiti kako izgleda slanje obavijesti kroz te akcije. Konkretno ovaj proces postavlja status teme na predloženu ukoliko ju je student napravio te šalje obavijesti mentoru da je student predložio temu te da ju je potrebno pregledati. Također šalje obavijesti kada je rad zakazan za obranu te kad je obranjen.



Slika 21 – „Salesforce Process Builder“ za proces prijava

Ovo je jedan dio procesa prijave završnih i diplomskih radova jer bi postavljanje cijelog procesa bilo jako nezgrapno. Princip izrade procesa je uglavnom isti kao i kod procesa za diplomske odnosno završne radove samo što ovaj proces radi druge stvari i na drugom objektu. Ovaj proces se odvija na prilagođenom objektu „Prijave“, a konkretno postavlja primjerene statuse za prijavnice te šalje odgovarajuće obavijesti korisnicima.

```
trigger ZavrsniRadTrigger on Zavrsni_Rad__c (after insert, after delete, before insert) {
    Set<Id> newUsersId = new Set<Id>();
    Set<Id> oldUsersId = new Set<Id>();
    List<User> userListZaUpdate = new List<User>();
    List<User> oldUserListZaUpdate = new List<User>();
    List<String> oldZavrsniNameList = new List<String>();

    if(Trigger.isBefore){
        if(Trigger.isInsert){
            for(Zavrsni_Rad__c zr : [SELECT Id, Name FROM Zavrsni_Rad__c]){
                String uMalaSlova = zr.Name.toLowerCase();
                oldZavrsniNameList.add(uMalaSlova);
            }

            for(Zavrsni_Rad__c zr : Trigger.new){
                newUsersId.add(zr.Mentor__c);

                String provjeraMalihSlova = zr.Name.toLowerCase();
                if(oldZavrsniNameList.contains(provjeraMalihSlova)){
                    zr.addError('Rad sa istim imenom vec postoji.');
                }
            }

            for(User u : [SELECT Broj_zavrsni_radova__c, Name FROM User WHERE Id IN: newUsersId]){
                u.Broj_zavrsni_radova__c += 1;
                userListZaUpdate.add(u);
            }

            update userListZaUpdate;
        }
    }

    if(Trigger.isAfter){
        if(Trigger.isDelete){
            for(Zavrsni_Rad__c zr : Trigger.old){
                oldUsersId.add(zr.Mentor__c);
            }

            for(User u : [SELECT Broj_zavrsni_radova__c FROM User WHERE Id IN: oldUsersId]){
                u.Broj_zavrsni_radova__c -= 1;
                oldUserListZaUpdate.add(u);
            }

            update oldUserListZaUpdate;
        }
    }
}
```

Slika 22 – Apex trigger

Salesforce-ov glavni „Back-End“ programski jezik je „Apex“. Apex je objektno orijentiran jezik koji se bazira na programskim jezicima kao što su Java i C#. Kako bi se sama upotrebljivost jezika u svrhu upravljanja sa pozadiskim događanjima aplikacije povećala potreban je i jezik s kojim bi Apex mogao dolaziti do podataka u Salesforce-u, a to je SOQL. SOQL ili „Salesforce Object Query Language“ perfektno surađuje sa Apex-om kako bi mu dohvatio potrebne podatke.

Nadalje, u našem primjeru primječujemo da je napravljen okidač na prilagođenom objektu „Završni Rad“, a vidimo da je prilagođen jer je napisan u stilu „Zavrsni_Rad__c“, a znamo što znači taj nastavak „__c“.

Poslije ključne riječi „trigger“ dolazi ime tog okidača te na kojem je objektu taj okidač napravljen. U zagradama dolaze tipovi tog okidača koje možemo koristiti unutar tog okidača. U ovom slučaju je to „After insert“, „After delete“ i „Before insert“. Osim njih tu su još i druge varijante, ali svaka varijanta može biti samo „before“ ili „after“. Te druge varijante su „Update“ i „Undelete“.

Na vrhu su definirani Set-ovi i liste koji se koriste u kodu. Apex putem IF provjere koja slijedi ustvrđuje s kojim tipom okidača će se baviti sad te ukoliko je riječ o „Before“ okidaču onda će znati kako će se tu baviti poslovima koji se odvijaju prije spremanja u Salesforce-ovu bazu. Iduće provjerava je li se bavi sa umetanjem podataka. U „for each“ petlji koja slijedi vidimo točno kako Apex može iskoristiti SOQL. U ovom primjeru ga iskorištava na način da pomoću varijable koja je istog tipa kao i podaci koje SOQL prolazi kroz sve podatke koje taj upit može dohvatiti te u svakoj iteraciji uzima ime završnog rada te ga pretvara u sva mala slova te dodaje u listu koju smo gore definirali. Ime se mora pretvoriti u sva mala slova kako se sustav nebi zbuino kod provjere je li rad s takvim imenom već postoji. U suprotnom bi jednostavna promjena jednog slova u malo ili veliko rješilo problem, ali ovako sprječavamo to. Dalje možemo vidjeti jedan novi koncept for each petlje jer sada postoji varijabla tipa Zavrsni Rad, ali odi po svim novim podacima koji su u tom okidanju bili uneseni. Tako da će u ovom setu završiti samo mentori koji su navedeni u novim unesenim radovima. Isto tako dohvaćamo imena novih radova te gledamo je li se rad s tim imenom već nalazi u onoj listi koju smo prethodno popunili sa postojećim imenima. U idućoj for each petlji dohvaćamo sve user-e koji su u setu kojeg smo prethodno popunili sa mentorima koji se nalaze u novim radovima. Sada dolaze na red one varijeble koje sam spominjao skroz gore u Schema odlomku, a to su „Broj zavrsni radova“ odnosno „Broj diplomskih radova“ ako je riječ o diplomskim radovima. Kada upit vrati mentore koji odgovaraju postavljenom onda se uzima polje „Broj zavrsni radova“ ili „Broj diplomskih radova“ za diplomske radove i uvećava se za 1 za svaki novi rad. Te se onda nakon toga vrši „update“ nad tim user-om. Sustav nakon toga provjerava putem „Validation rule“ svojstva.

Rule Name	MaximalanBrojZavrsnihRadova
Error Condition Formula	Mentor__r.Broj_zavrsni_radova__c > 10
Error Message	Maksimalan broj ponuđenih radova dostignut.

Slika 23 – „Salesforce Validation Rule“ pravilo za maksimalan broj završnih radova

Ovo svojstvo omogućuje provjeru nakon ažuriranja tog mentora u okidaču da baci navedenu pogrešku prije unosa u bazu.

Drugi dio okidača je „after“ dio, a after koristimo iz razloga što nam je potrebam identifikator onoga što već je u bazi kako bi došli do tog podatka i radili promjene nad time. U ovom

slučaju želi se ažurirati podatak na user-u nakon brisanja rada. U for each petlji varijabla tipa završni rad ovoga puta uzima sve podatke koji su bili prisutni prije unosa novih podataka. U idućoj pelji opet prolazimo kroz user-e, ali ovaj puta filtriramo po drugom set-u koji je uzeo samo stare mentore odnosno one koji su već bili pohranjeni te ukoliko se izbrisao rad na kojemu su oni bili mentor onda im se smanjuje broj završnih radova za jedan te se ažurira taj user.

```
<div style = "background-color: white; clear: both; position: relative; margin-top: 500px; margin-bottom: -15px; border-top: 1px solid #78CDF5;">
    <div>
        <div>
            <div>
                <div >

                </div>
            </div>
            <div style = "text-align: center; margin-top: 5px">
                <p>
                    <a href="https://fipu.unipu.hr/fipu/o_fakultetu/nastavnici" class="text-black">NASTAVNICI</a>     
                    <a href="https://fipu.unipu.hr/fipu/za_studente/oglasna_ploca" class="text-black">OGLASNA PLOČA</a>     
                    <a href="https://fipu.unipu.hr/fipu/novosti" class="text-black">NOVOSTI</a>     
                </p>
            </div>
        </div>
        <div>
            <div>
                <div>
                    <a href="https://fipu.unipu.hr/">
                       <img style = "width: 300px; height: 90px; align: center" src="https://fipu.unipu.hr/_pub/themes_static/oikt/default/icons/fipu_hr.png" alt=""/>
                    </a>
                    <p style = "text-align: center; padding-bottom: 5px; color: #78CDF5">
                        © 2019 Fakultet informatike u Puli
                    </p>
                </div>
            </div>
        </div>
    </div>
</div>
```

[1]

Slika 24 – Podnožje HTML

Jednostavan kod koji sam koristio kako bi podnožje napravio jednostavnim, ali prepoznatljivim.

5. Testovi

```
public class TestDataFactory {
    public static List<User> createProfesor(Integer kolicina){
        List<User> u = new List<User>();
        Profile p = [SELECT Id FROM Profile WHERE Name='Profesor'];
        For(Integer i = 1; i <= kolicina; i++){
            u.add(new User(Alias = 'prof' + i, Email='profesoruser' + i + '@profesor.com',
            EmailEncodingKey='UTF-8', LastName='Testing' + i, LanguageLocaleKey='en_US',
            LocaleSidKey='en_US', ProfileId = p.Id,
            TimeZoneSidKey='America/Los_Angeles', UserName='profesorusers' + i + '@profesor.com'));
        }

        return u;
    }
}
```

Slika 25 – Pomoćna Apex klasa za generiranje user-a

[1] Korišteni linkovi za preusmjeravanje su dostupni na: https://fipu.unipu.hr/fipu [20. srpnja 2019.]

```apex
@isTest
public class ZavrsniRadTriggerTest {
    @testSetup
    static void setupData(){
        Integer i = 1;
        List<User> listaProfesora = TestDataFactory.createProfesor(i);

        insert listaProfesora;
    }

    @isTest
    static void zavrsniInsertAndDeleteTest(){
        User profesor = [SELECT Id FROM User WHERE Alias =: 'prof1' LIMIT 1];

        Test.startTest();

        System.runAs(profesor){
            for(Integer i = 1; i <= 10; i++){
                Zavrsni_Rad__c zavrsni = new Zavrsni_Rad__c(
                    Name = 'zavrsni' + i,
                    Mentor__c = profesor.Id
                );
                insert zavrsni;
            }

            Integer count = database.countQuery('SELECT count() FROM Zavrsni_Rad__c');
            System.assertEquals(10, count);
            List<User> brojZavrsnih = [SELECT Id, Broj_zavrsni_radova__c FROM User WHERE Id =: profesor.Id LIMIT 1];
            System.assertEquals(10, brojZavrsnih[0].Broj_zavrsni_radova__c);

            Zavrsni_Rad__c zavrsniIstoIme = new Zavrsni_Rad__c(
                    Name = 'zavrsni1',
                    Mentor__c = profesor.Id
            );
            try{
                insert zavrsniIstoIme;
            }
            catch(Exception e){
                Boolean expectedExceptionThrown =  e.getMessage().contains('Rad sa istim imenom vec postoji') ? true : false;

                System.assertEquals(expectedExceptionThrown, true);
                Integer count3 = database.countQuery('SELECT count() FROM Zavrsni_Rad__c');
                System.assertEquals(10, count3);
                List<User> brojZavrsnihIstoIme = [SELECT Id, Broj_zavrsni_radova__c FROM User WHERE Id =: profesor.Id LIMIT 1];
                System.assertEquals(10, brojZavrsnihIstoIme[0].Broj_zavrsni_radova__c);
            }

            Zavrsni_Rad__c maksimum = new Zavrsni_Rad__c(
                Name = 'zavrsni',
                Mentor__c = profesor.Id
            );
            try{
                insert maksimum;
            }
            catch(Exception e){
                Boolean expectedExceptionThrown =  e.getMessage().contains('Maksimalan broj ponuđenih radova dostignut.') ? true : false;

                System.assertEquals(expectedExceptionThrown, true);
                Integer count4 = database.countQuery('SELECT count() FROM Zavrsni_Rad__c');
                System.assertEquals(10, count4);
                List<User> brojZavrsnihMaksimum = [SELECT Id, Broj_zavrsni_radova__c FROM User WHERE Id =: profesor.Id LIMIT 1];
                System.assertEquals(10, brojZavrsnihMaksimum[0].Broj_zavrsni_radova__c);
            }

            Zavrsni_Rad__c zavrsniBrisanje = [SELECT Id, Name FROM Zavrsni_Rad__c WHERE Name =: 'zavrsni1' LIMIT 1];
            system.debug('brisanje: ' + zavrsniBrisanje);
            delete zavrsniBrisanje;
            Integer count5 = database.countQuery('SELECT count() FROM Zavrsni_Rad__c');
            System.assertEquals(9, count5);
        }
        Test.stopTest();
    }
}
```

Slika 26 – Test za gore navedeni okidač

Počevši sa gornjom slikom na kojoj je Apex klasa koja generira user-e. „Best practice“ Salesforce-a je takav da oni podaci koji će se koristiti više puta ili koji će se u više namjena koristiti stavi u klasu pod imenom „TestDataFactory“ te se putem nje dolazi do tog podatka s kojim želimo upravljati. Metoda koja se zove iz te klase prima parametar „kolicina“ koji

definira koliko tih user-a želimo da metoda vrati.

Na slici 26 vidimo klasu koja je definirana sa „@isTest“ što označava da se ta klasa koristi u svrhu testiranja drugih funkcionalnosti u aplikaciji te da sustav prilikom pokretanja testova zna koje klase gledati, ali i metode jer se i one moraju označavati sa istom oznakom. Osim @isTest oznake primječujemo is oznaku „testSetup“, a ta oznaka označuje metodu koja se poziva prije pokretanja svake druge metode u svrhu pružanja novih testnih podataka. U našem slučaju radi jednog profesora. Glavna metoda pri pokretanju prima podatke iz @testSetup metode. Varijabla profesor koje je tipa User poprima vrijednosti tog novog testnog profesora jer je postavljem upit nad točno tip profesorom.

„Test.startTest()“ je funkcija koja alocira novi Salesforce limite kako bi test mogao proći nesmetano, ali se mora imati na umu da tu funkciju se može pozvati samo jednom u metodi te se uvijek poslije nje negdje u kodu mora nalaziti „Test.stoptTest()“ kako nebi došlo do pada testiranja. U for petlji koja slijedi je definirano da se napravi 10 novih radova te da se isti umetnu. Nakon toga radimo upit nad bazom te utvrđujemo je li ih uistinu ima 10 te ako je onda nam taj dio testa prolazi. Isto tako moramo i utvrditi je li broj radova porastao za 10 kod imaginarnog profesora te ako je onda i taj test prolazi. Ukoliko dodamo još jedan rad sustav sa imenom koji bi već trebao postojati onda bi nam sustav trebao vratiti pogrešku te nakon što uhvatimo „Exception“ ako se ima što za uvatiti mora se utvrditi je li ta poruka zaista ona očekivana poruka. Opet, ako ustvrdimo da je poruka uistinu bila onakva kakvom smo ju očekivali onda nam je i taj test uspješan. Naravno, moramo utvrditi i da se broj radova nije povećao. Ista takva provjera se mora izvršiti i sa unosom koji bi prekoračio dozvoljeni broj radova po mentoru. Zadnja provjera je provjera brisanja u kojoj radimo upit nad jednim imaginarnim radom te ga prišemo iz baze. Nakon brisanja utvrđujemo je li se broj radova na user-u uistinu smanjio. Na kraju uistinu primječujemo „Test.stopTest()“ koji označava kraj testiranja.

Salesforce zahtjeva minimalni prolaznost testova od 75%, ali najbolja praksa je naravno da se ide na što više moguće, a ovo testiranje je imalo prolaznost 100% na testiranju okidača i za završne radove i za diplomske.

6. Zaključak

U ovo moderno doba kada je sve više-manje digitalno trebali bi težiti ka tome da pratimo tu modernizaciju. Problemi koje nam donose papiri i sporost cijelog sustava dovoljno su nam zadavali glavobolje. Samom implementacijom digitalnih sustava sve više i više ubrzavamo protok informacija i samu efikasnost nas kao ljudskih bića.

Ova aplikacija predstavlja potencijalno rješenje za sporost i nezgrapnost trenutnog sustava u kojem su glavne karakteristike sporost i neorganiziranost. Implementacija ove aplikacije je izvršena na Salesforce platformi koristeći neke od mnogih svojstava navede platforme. Neka od svojstva koja se koriste u ovom radu su „Validation Rules“, „Process Builder“, Apex programski jezik, SOQL te sama baza Salesforce-a koja radi na Cloud tehnologiji. Osim navedenih stvari koje nudi sam Salesforce korišten je i HTML te CSS. Nadalje, Visual Studio Code se pokazao kao dostojno i ugodno okruženje za rad.

Nastavno na cilj i zadatak ovog rada koji je bio repliciranje i u nekim dijelovima poboljšanje samog sustava rekao bih da sam zadovoljan sa konačnim ishodom jer sam sustav pruža još puno mjesta za napredak iako je ispunjeno sve što sam htio implementirati u ovaj sustav. Najveći problem u cijeloj implementaciji su predstavljale licence koje se moraju kupiti od Salesforce-a kako bi više korisnika moglo koristiti aplikaciju, ali i kako bi se ista mogla bolje istestirati.

Osim toga sama funkcionalnost aplikacije kakva je zadovoljava onu brzinu koja se htjela postići, a to je da u nekoliko klikova i bez frustracija od nepotrebnog trošenjem vremena u čekaonici ne radeći ništa.

Ovaj sustav i aplikacija služe kao primjer što se može napraviti na navedenoj platformi te koje su njene prednosti, ali i nedostaci. Kada bi se ovakav sustav išao profesionalno implementirati onda je za takav pothvat potreban tip iskusnih administratora, arhitekata i developera koji bi razvili ovaj sustav do perfekcija. Naravno imajući u obzir da klijent pruži sve potrebne licence.

7. Literatura

1. Službena stranica fakulteta:
   https://fipu.unipu.hr/fipu [20. srpnja 2019.]

8. Sažetak

Ova aplikacija je izrađena u sklopu završnog rada kojemu je tema „Razvoj sustava za prijavu završnih i diplomskih radova“ te je kao takva i izrađena.
Sustav se bavi poboljšanjem trenutnog sustava koji nije digitaliziran na način da se proces replicira u digitalnom obliku. Za taj pothvat korištena je Salesforce platforma koja pruža lijep izgled, cloud tehnologiju, lakoću implementiranja i još mnogo drugih pogodnosti, ali sa sobom nosi i mane kao što su licence koje klijent treba kupiti od Salesforce-a kako bi njegovi korisnici zapravo mogli i koristiti aplikaciju.
Razvijanjem ove aplikacije dolazi se do dijela u kojemu je jako limitirano daljnje razvijanje aplikacije baš zbog problema tih licenci zbog kojih je otežano testiranje, ali i daljnja implementacija drugih funkcionalnosti. Trenutna verzija aplikacije je izvrstan pokazatelj što aplikacija može te bi profesionalna implementacija ovakvog sustava mogla biti jako uspješna

Ključne riječi: Salesforce, Apex, SOQL, aplikacija, Završni rad, Diplomski rad, Sustav

9. Summary

This application is built as a part of a final thesis which dealt with „Development of the system for the registration of final and diploma papers“ and it was built with that in mind. Systems main focus is on bettering current system which is not digitalized by replicating it digitaly. For that undertaking Salesforce platform was used because it gives us a fresh new look, cloud technology, ease of implementing and many other things but there are also some downsides such as licensing problems. Licenses have to be bought by the client from Salesforce for it's clients to be able to use this new system.
Development of this application came to the part where the whole development is now limited by those licences because withut those licences the system can not be tested properly with many users and so further implementation of more functionalities is limited. Current version of the application is more than enough to demonstrate what this platform can do and what it could be if it were profesionaly implemented.

Keywords: Salesforce, Apex, SOQL, application, Final thesis, Diploma thesis, System