Web aplikacija za učenje i dijeljenje znanja iz STEM područja unutar internetske zajednice studenata

Banov, Marina

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

WEB APLIKACIJA ZA UČENJE I DIJELJENJE ZNANJA IZ STEM PODRUČJA UNUTAR INTERNETSKE ZAJEDNICE STUDENATA

Rijeka, rujan 2020.

Marina Banov

0069083031

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

WEB APLIKACIJA ZA UČENJE I DIJELJENJE ZNANJA IZ STEM PODRUČJA UNUTAR INTERNETSKE ZAJEDNICE STUDENATA

Mentor: Doc. dr. sc. Marko Gulić

Rijeka, rujan 2020.

Marina Banov

0069083031

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 3. ožujka 2020.

Zavod: **Zavod za računarstvo**
Predmet: **Razvoj web aplikacija**
Polje: **2.09 Računarstvo**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Marina Banov (0069083031)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Web aplikacija za učenje i dijeljenje znanja iz STEM područja unutar internetske zajednice studenata / Web application for learning and sharing knowledge from STEM fields within an online community of students**

Opis zadatka:

Razviti web aplikaciju za učenje i dijeljenje znanja između studenata STEM područja. Implementirati funkcionalnost stvaranja zasebnih grupa unutar internetske zajednice od strane studenata koji žele učiti i dijeliti znanje iz sličnog područja. U svakoj grupi implementirati funkcionalnost zadavanja zadataka koje treba riješiti, kao i mogućnost interakcije među studentima s ciljem rješavanja zadanih zadataka. Također, treba razviti i funkcionalnost ocjenjivanja ponuđenih rješenja zadataka. Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Django radni okvir uz proizvoljno odabran sustav za upravljanje bazama podataka. Za razvoj klijentskog dijela aplikacije treba koristiti Bootstrap java script knjižnicu s ciljem učinkovitog renderiranja aplikacije na uređajima s različitim veličinama zaslona. Također, treba opisati cjelokupni razvoj web aplikacije, kao i korištenje pripadajućih tehnologija unutar aplikacije.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Marina Banov

Zadatak uručen pristupniku: 16. ožujka 2020.

Mentor:

Doc. dr. sc. Marko Gulić

Predsjednik povjerenstva za završni ispit:

Izv. prof. dr. sc. Kristijan Lenac

IZJAVA O SAMOSTALNOJ IZVEDBI RADA

Ovime izjavljujem da sam samostalno izradila završni rad pod vodstvom mentora doc. dr. sc. Marka Gulića.

Marina Banov

ZAHVALA

Od srca zahvaljujem mentoru doc. dr. sc. Marku Guliću na korisnim savjetima i stručnom vođenju te svojoj obitelji, kolegama i prijateljima koji su me podržavali tijekom studija i za vrijeme pisanja ovog rada.

SADRŽAJ

1. UVOD

Potreba za online platformama za suradnju je u porastu iz dana u dan i samo je jedna od posljedica napretka u procesu digitalizacije svijeta oko nas. Suradnja među pojedincima može se ostvariti na više načina, pa se tako i alati za suradnju mogu orijentirati na veliki broj zadataka: zajedničko uređivanje datoteka, ostvarivanje učinkovite komunikacije, koordinacija velikih grupa kako bi se svima osigurali potrebni resursi itd. Uzme li se u obzir trend održavanja nastave na daljinu, postaje jasno zašto je i u obrazovanju bitno imati na raspolaganju odgovarajuće alate za online suradnju. Već dugi niz godina u upotrebi su sustavi za upravljanje učenjem (engl. *learning management system* – LMS) koji pružaju učinkovitu podlogu za formalno praćenje nastave i napretka studenata kao npr. Moodle i Google Classroom [1].

Potaknuto neformalnim alatima (poput online zajednice programera Stack Overflow [2]) i željom da se povežu korisnici različitih studijskih grupa, u ovom radu bit će predstavljena web aplikacija za učenje i dijeljenje znanja namijenjena studentima iz područja znanosti, tehnologije, inženjerstva i matematike (engl. *science, technology, engineering, and mathematics* – STEM). Aplikacija svojim korisnicima olakšava dijeljenje znanja, omogućava zadavanje zadataka i potiče ih na međusobno pomaganje.

Za razvoj poslužiteljskog dijela aplikacije korišten je Django radni okvir [3] i SQLite [4] sustav za upravljanje bazama podataka. Za razvoj klijentskog dijela aplikacije korištena je Bootstrap knjižnica s ciljem učinkovitog renderiranja aplikacije na uređajima različitih veličina. Ti alati za razvoj omogućili su kreiranje modularnog rješenja ugodnog dizajna i organiziranog, lako čitljivog koda. Drugo poglavlje će detaljnije opisati te tehnologije.

Glavni ciljevi aplikacije postignuti su na sljedeći način: studenti ovisno o područjima koja ih zanimaju biraju grupe u koje će postavljati pitanja i zadatke i u kojima će moći odgovarati na pitanja drugih korisnika te im tako pomoći pri rješavanju zadataka. Mogućnost kreiranja grupa povećava razinu strukturiranosti sadržaja u aplikaciji, a korisnici znaju točno gdje mogu pronaći ono što ih zanima. Aplikacija je namijenjena studentima različitih, iako srodnih znanstvenih područja, stoga sve grupe osim svog naziva (koji bi mogao odgovarati, primjerice, nazivu kolegija nekog studija) imaju i oznaku kojem od STEM područja pripadaju. Još jedan važan element ove aplikacije je ocjenjivanje ponuđenih rješenja zadataka jer osigurava kvalitetu odgovora, promiče kritičko promišljanje dobivenog odgovora i služi kao motivacija korisnicima na davanje što konkretnijih i boljih rješenja. Naime, ukupan broj odgovora i ukupan broj pozitivnih ocjena na odgovorima vidljivi su na korisničkim profilima i predstavljaju dokaz pouzdanosti dotadašnjih

odgovora korisnika. Svi korisnici mogu neki odgovor ocijeniti pozitivno ili negativno, ali samo onaj korisnik koji je zadao zadatak može jedan odgovor na njega označiti kao prihvaćeni odgovor. Ove i ostale funkcionalnosti aplikacije poput autentifikacije i praćenja grupa od posebnog interesa bit će predstavljene u trećem poglavlju.

U četvrtom će poglavlju biti detaljno objašnjen proces samog razvoja, a pozornost će se skrenuti na njegove glavne korake. S obzirom na to da se razvoj sličnih funkcionalnosti temelji na ponavljanju dobrih praksi, proces će se sažeti i objasniti na primjeru funkcionalnosti postavljanja pitanja koja je dovoljno reprezentativna.

2. KORIŠTENE TEHNOLOGIJE

U narednim potpoglavljima bit će opisane tehnologije koje su bile potrebne za razvoj svih željenih funkcionalnosti u sklopu web aplikacije. Django razvojno okruženje je bilo ključno za poslužiteljski i klijentski dio aplikacije te za neprimjetnu integraciju sa sustavom za upravljanje bazama podataka SQLite. Kao što je uobičajena praksa u razvoju web aplikacija, važnu ulogu za razvoj klijentske strane imali su programski jezici HTML, CSS i JavaScript i knjižnice koje proširuju njihove funkcionalnosti, u prvom redu Bootstrap [5] i jQuery [6]. Kako bi se omogućilo praćenje napretka projekta, korišten je sustav za verzioniranje Git [7].

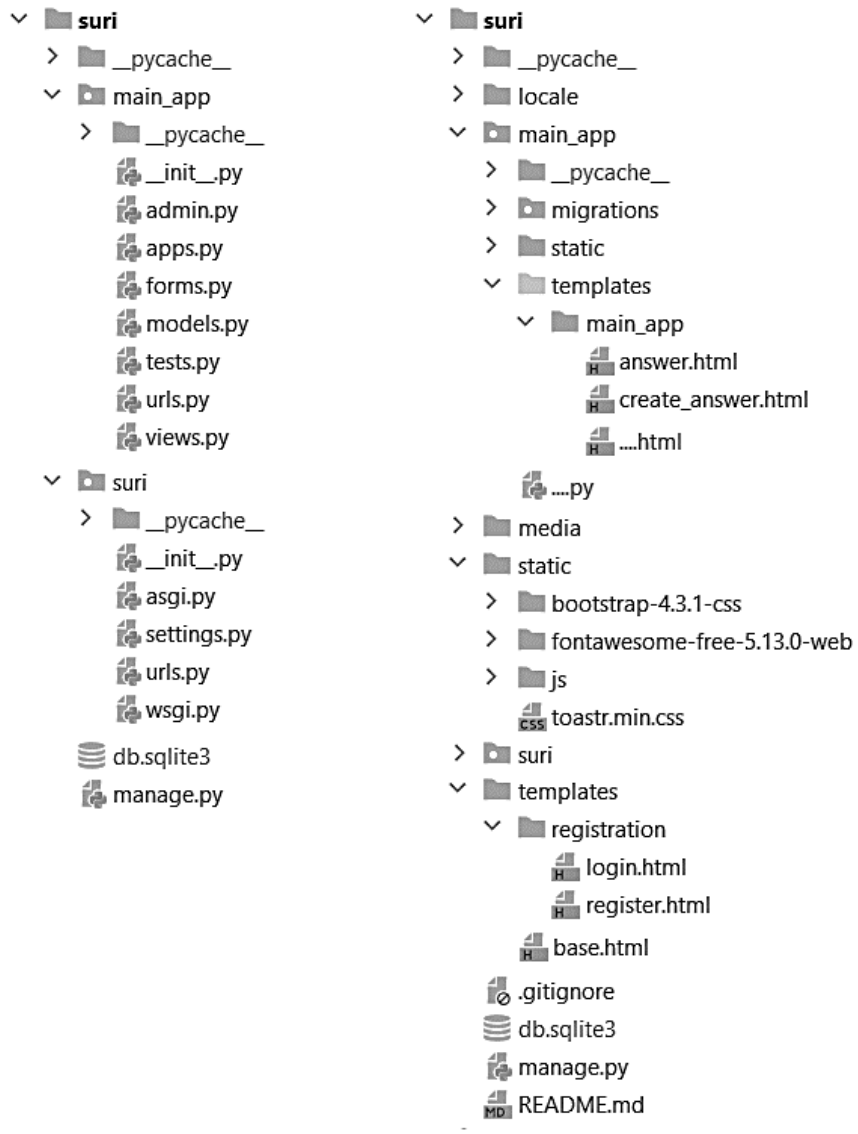
2.1. Django razvojno okruženje

Django je razvojno okruženje pisano u programskom jeziku Python koje programerima nudi širok raspon alata za olakšan razvoj web aplikacija [3]. Unutar Django radnog okvira integrirano je nekoliko rješenja ključnih za pojednostavljeni razvoj pouzdanog autentifikacijskog sustava, o kojima će biti govora u kasnijim poglavljima.

Da bi se započeo razvoj aplikacije, potrebno je pokrenuti komandu za instalaciju Djanga (`python -m pip install Django`) i započeti novi projekt (`django-admin startproject ime_projekta`). To će stvoriti novu mapu s imenom projekta i unutar nje bazu podataka *db.sqlite3* i nekoliko *.py* skripti. Među njima se ističu *settings.py* koja sadrži postavke poput putanja do statičkih datoteka (JavaScript knjižnice) i *manage.py* koja pokreće lokalni server, mijenja bazu podataka ili kreira nove komponente unutar našeg projekta. Ako se pokrene naredba `python manage.py runserver ip:port` i otvori web preglednik, na adresi `ip:port` bit će vidljiva stvorena web aplikacija, odnosno zadani predložak.

Nakon uspješnog pokretanja lokalnog servera Django projekta mogu se početi dodavati funkcionalnosti, a to će se učiniti uz pomoć komande `python manage.py startapp ime_aplikacije`. To će unutar projekta stvoriti novu mapu i još nekoliko *.py* skripti pa će datotečna struktura projekta biti poput one na Slici 2.1. (lijevo). Mapa `suri` je glavna mapa projekta, a mapa `main_app` je mapa komponente. Za dodavanje novih funkcionalnosti bilo je potrebno kreirati novu komponentu jer se programera potiče na pisanje modularnog koda koji bi se mogao iskoristiti u drugim projektima. Zbog toga se na te komponente može gledati kao na mini-aplikacije: jedan projekt može sadržavati više aplikacija i jedna aplikacija može biti dio više projekata.

S vremenom će količina koda u našem projektu rasti (jer će biti potrebno uključiti JavaScript knjižnice, *.html* datoteke, slike itd.), pa će datotečna struktura biti sličnija onoj na Slici 2.1. (desno).

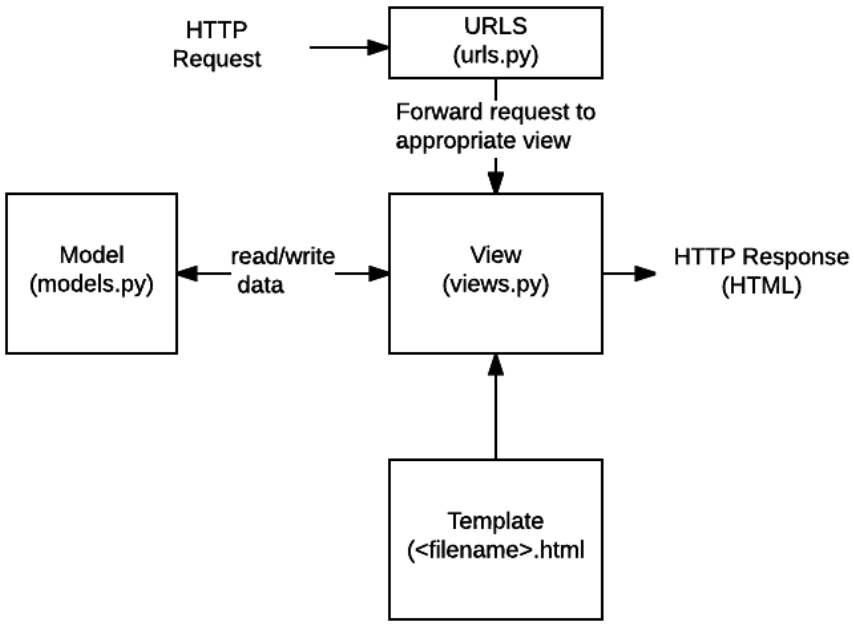


Slika 2.1. Datotečna struktura na početku (lijevo) i na kraju (desno) projekta

Kako bi se objasnila uloga tih datoteka, potrebno je spomenuti neke od glavnih značajki Django radnog okvira.

Django se temelji na arhitekturi model-pogled-predložak (engl. *Model-View-Template* – MVT). Modeli su reprezentacija podataka u bazi i nastaju primjenjujući objektno-relacijsko mapiranje (ORM). Pogledi su funkcije ili klase čija je svrha obrada podataka dobivenih iz interakcije s korisnikom i prikaz sadržaja korisnicima, odnosno primanje HTTP zahtjeva (engl. *request*) i slanje HTTP odgovora (engl. *response*). Predlošci su *.html* datoteke odnosno prilagođeni

sadržaj koji se prikazuje korisniku. Jedna od prednosti MVT arhitekture je slaba uparenost između te tri komponente zbog čega promjene u jednoj komponenti najčešće ne uzrokuju velike promjene u drugim komponentama. Na Slici 2.2. prikazana je shema primjene MVT arhitekture u Djangu.



Slika 2.2. MVT arhitektura u Djangu, preuzeto s: [8]

U četvrtom će poglavlju ovaj mehanizam biti detaljnije pojašnjen na konkretnom primjeru, ali za sada je bitno napomenuti da Django interpretira HTTP zahtjeve koje korisnik šalje kao URL mapiranje. Unutar datoteke *urls.py* traži odgovarajući URL uzorak i poziva poglede iz datoteke *views.py*. Ti pogledi pristupaju bazi i vraćaju *.html* predložak kao HTTP odgovor. Prilikom pristupanja bazi nije potrebno koristiti SQL upite jer ORM paradigma podatke iz baze u kodu predstavlja kao objekte. Njihovim atributima se pristupa direktno, jednostavno je kreirati objekte instanciranjem klasa i vršiti promjene nad njima. Primjerice, kada je potrebno obrisati neku n-torku, to se čini koristeći metodu `delete()` nad konkretnim objektom, a ne pomoću SQL komande `DELETE FROM`. Modeli se definiraju u datoteci *models.py*.

U Kodnom isječku 2.1. (sadržaj preuzet s: [3]) predstavljen je minimalni primjer za sve te datoteke. U datoteci *models.py* definiran je jedan entitet iz baze podataka kao klasa `Question` s dva atributa: datum objave i tekst pitanja. U datoteci *urls.py* definirana je jedna URL putanja koja je povezana s pogledom `index` u datoteci *views.py*. Ta funkcija bez korištenja SQL upita pristupa objektima iz baze kako bi korisniku prikazala pet najnovijih pitanja. Format koji korisnik vidi je prikazan u predlošku *index.html*.

```python
from django.db import models


class Question(models.Model):
  question_text = models.CharField(max_length=200)
  pub_date = models.DateTimeField('date published')
```

```python
from django.urls import path
from . import views


urlpatterns = [ path('', views.index, name='index') ]
```

```python
from .models import Question
from django.shortcuts import render


def index(request):
    context = { 'latest_question_list': Question.objects.order_by('-pub_date')[:5] }
    return render(request, 'main_app/index.html', context)
```

```html
<ul>
{% for question in latest_question_list %}
  <li>{{ question.question_text }}</li>
{% endfor %}
</ul>
```

Kodni isječak 2.1. Minimalni primjer za models.py, urls.py, views.py i predložak index.html

Prilikom kreiranja i mijenjanja modela (tablica) skripta *manage.py* provodi potrebne promjene (migracije) nad strukturom baze podataka. Kako bi se promjene u klasama automatski prevele u SQL upite koji će se potom odraziti u datoteci *db.sqlite3* koriste se naredbe `python manage.py makemigrations` i `python manage.py migrate`. Ako se želi vidjeti ispis SQL upita koje Django automatski kreira, može se koristiti naredba `python manage.py sqlmigrate` (Kodni isječak 2.2.).

```python
from django.db import models


class Example(models.Model):
  string_field =
      models.CharField(max_length=2048,
                       null=True)
  bool_field =
      models.BooleanField(null=True)
  number_field =
      models.IntegerField(null=True)
```

```
BEGIN;
--
CREATE TABLE "main_app_example" (
  "id" integer NOT NULL PRIMARY KEY
     AUTOINCREMENT,
  "string_field" varchar(1024) NULL,
  "bool_field" bool NULL,
  "number_field" integer NULL);
--
COMMIT;
```

Kodni isječak 2.2. Rezultat naredbe „python manage.py sqlmigrate“

Za ovu aplikaciju su bili važni još neki paketi prilagođeni za upotrebu uz Django radni okvir, a to su:

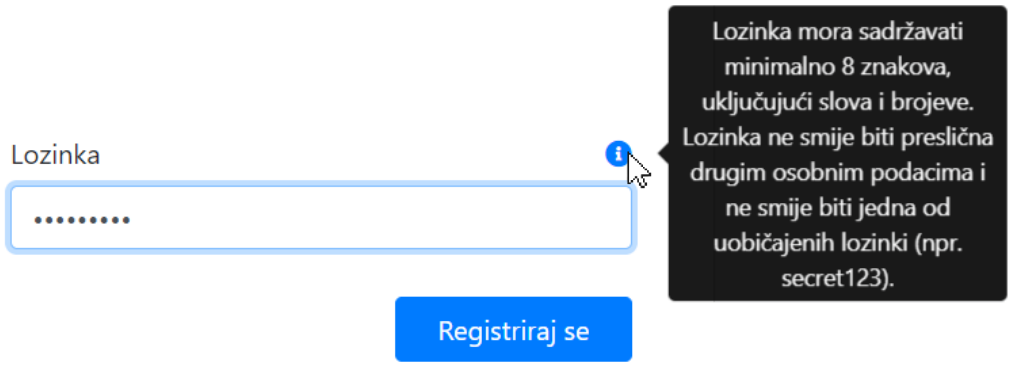
- *django-notifications* koji je omogućio slanje relevantnih obavijesti korisnicima u njihove sandučiće kako bi im mogli pristupiti u bilo kojem trenutku,
- *django-cleanup* koji je vodio računa da se prilikom brisanja pitanja i odgovora obrišu i sve povezane datoteke i
- *django-bootstrap-modal-forms* koji je nadogradio dizajn aplikacije funkcionalnim skočnim prozorima. Tako se, primjerice, kada korisnik želi obrisati neki komentar, pojavi skočni prozor *Sigurno želiš obrisati komentar?* U kasnijim poglavljima bit će objašnjeno kako se u Djangovim pogledima implementiraju CRUD (engl. *create*, *read*, *update*, *delete*) operacije, ali za sada treba spomenuti kako ti skočni prozori pristupaju navigaciji malo drugačije (kako bi korisnik ostao na istoj lokaciji dok je prozor otvoren), a logici pristupaju isto kao i obični pogledi.

2.2. Bootstrap

Bootstrap je skup alata koji objedinjuje CSS i JavaScript kako bi se postigao moderan i intuitivan dizajn na koji su korisnici naviknuti u svrhu olakšane upotrebe aplikacije. Glavna prednost korištenja Bootstrapovih komponenti je što je garantirana responzivnost dizajna te se

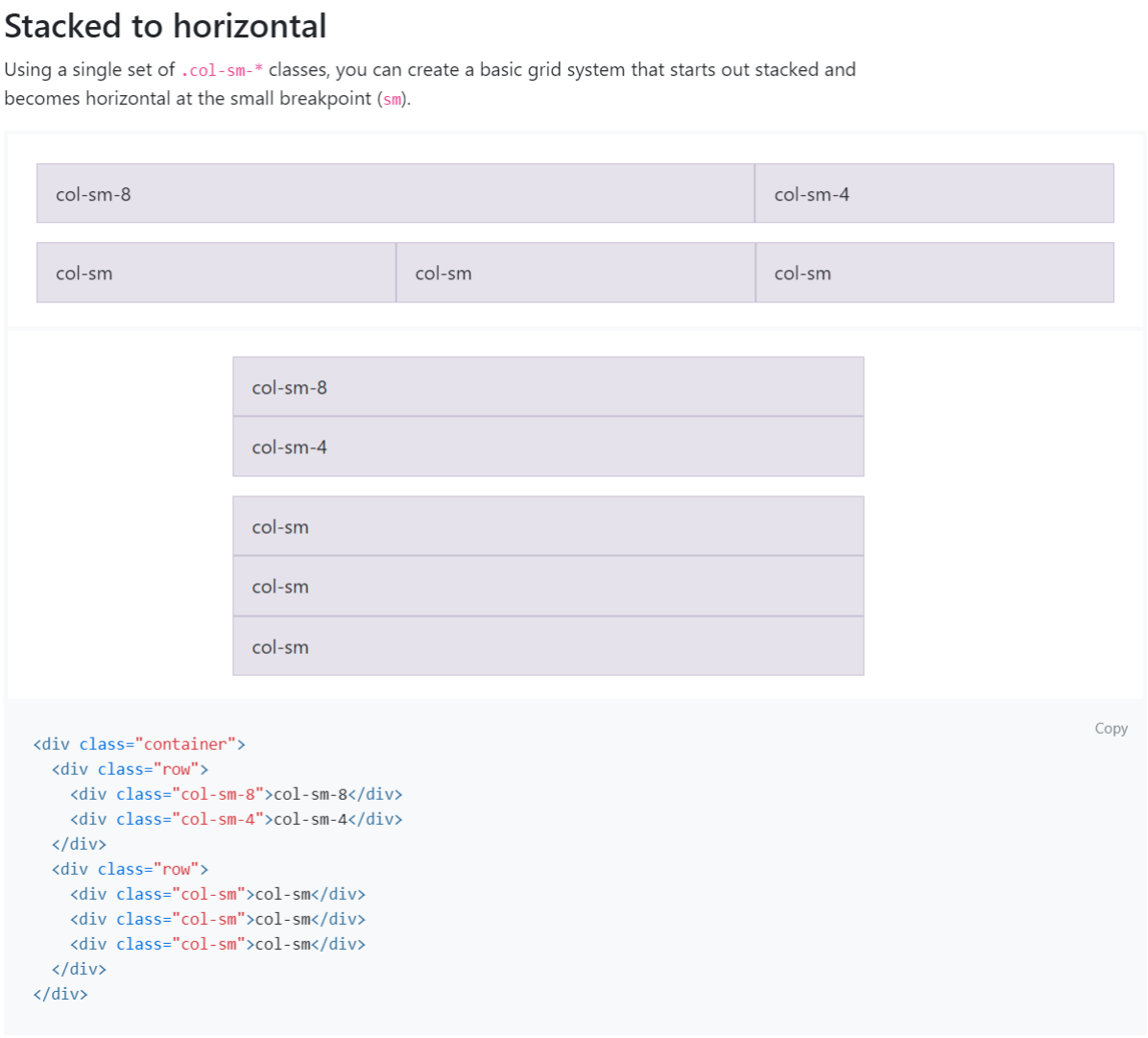
aplikacija učinkovito prilagođava svim širinama zaslona. Također se smanjuje količina CSS-a koju programer mora pisati [5].

U sklopu ove aplikacije, Bootstrap je bio ključan onda kada se htjelo prikazati određene elemente tek nakon interakcije s korisnikom. Primjerice, paket koji je već ranije spomenut, *django-bootstrap-modal-forms*, otvara skočni prozor tek kada korisnik pritisne na određeno dugme. Ta bi funkcionalnost bila zahtjevna za implementaciju isključivo uz Vanilla JavaScript i vlastite CSS klase. Integracijom Bootstrapovog rješenja, omogućeno je fokusiranje na logiku pogleda umjesto na klijentski dio otvaranja prozora i njegovo stiliziranje, a dobiveni rezultat je u potpunosti usklađen s dizajnom ostatka aplikacije. Ostali primjeri korištenja Bootstrapovih komponenata bili bi (Slika 2.3.): polje za unos teksta u obrascima, gumb *Registriraj se* i balončić s dodatnim informacijama koji se pojavljuje kada se mišem pređe preko ikone za dodatne informacije.



Slika 2.3. Korištenje Bootstrapovih komponenti unutar aplikacije

Responzivnost dizajna postignuta je korištenjem definiranih CSS klasa. Poanta je koristiti te CSS klase kako bi se složila rešetkasta struktura čije se ćelije prilagođavaju ekranima različite veličine. Određene HTML elemente treba označiti kao *container,* a ugniježđeni elementi slažu se u redove i stupce. Bootstrap je u svojim datotekama definirao kako bi se te klase trebale ponašati na različitim širinama ekrana (prateći prijelomne točke) i koristi oznake poput „sm“ (*small,* manje od 768px), „md“ (*medium,* manje od 992px), „lg“ (*large,* manje od 1200px) itd. Moguće je odrediti omjer koji neka ćelija zauzima u odnosu na širinu retka uz pomoć koeficijenata, ali važno je imati na umu da zbroj koeficijenata u retku mora biti 12 (npr. ako bi jedna ćelija trebala zauzimati 2/3 retka, a druga 1/3, onda će se prvoj dodati klasu *col-sm-8*, a drugoj *col-sm-4*). Primjer je vidljiv na Slici 2.4 (sadržaj preuzet s: [5]).



Slika 2.4. Izgled Bootstrapovih ćelija na ekranima širine 1532px (gore) i 528px (sredina) uz kod (dolje)

Upotrijebljene su i Fontawesome [9] ikone koje se izgledom dobro slažu s Bootstrapovim komponentama, a također su dobro poznate među korisnicima.

2.3. Programski jezici na klijentskoj strani

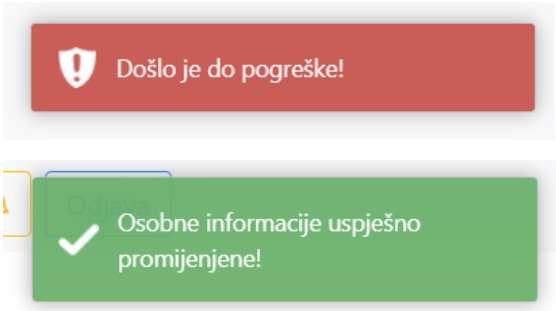
HTML je korišten radi strukturiranja sadržaja koji će biti prikazan na klijentskoj strani. Osim standardnih HTML oznaka, u *.html* datotekama pojavljuje se i posebna Djangova sintaksa kako bi se sadržaj mogao mijenjati ovisno o vrijednostima varijabli, grananjima ili petljama. Dobar primjer je datoteka *group.html* prikazana u Kodnom isječku 2.3. U ovaj predložak će se, prije nego što on bude poslan klijentu, ubaciti kontekst definiran u pogledu.

```
<h1>{{ group.group_name }}</h1>
<span class="badge badge-pill badge-primary">{{ group.field_name }}</span><br>

{% if user.is_authenticated %} ... {% else %} ... {% endif %}

<div class="mt-4">
  {% for question in questions %}
    <p><a href="{% url 'question' question.id %}">{{ question }}</a></p>
  {% endfor %}
</div>
```

Kodni isječak 2.3. groups.html

Većina CSS koda koja je dio ove aplikacije je ustvari dio Bootstrapovih rješenja i samo su u rijetkim slučajevima bile potrebne prilagodbe.

Za kod koji se izvršava na klijentskoj strani korišten je JavaScript i to primarno jQuery knjižnica čija je svrha pojednostavljenje JavaScript koda. Najviše se koristila u kombinaciji s Bootstrapovim komponentama. Također je upotrijebljena knjižnica *toastr.js* [10] za prikaz kratkotrajnih obavijesti sustava (Slika 2.5.).



Slika 2.5. Obavijesti sustava nestaju nakon par sekundi

3. FUNKCIONALNOSTI WEB APLIKACIJE

U ovom poglavlju bit će navedene i ukratko objašnjene funkcionalnosti web aplikacije za učenje i dijeljenje znanja. Aplikacija korisnicima nudi sljedeće mogućnosti:

- autentifikacija (registracija i prijava),
- pregled korisničkih profila,
- uređivanje osobnih informacija,
- promjena lozinke računa,
- kreiranje grupa povezanih s određenim područjem,
- brisanje grupa,
- praćenje grupa od posebnog interesa,
- postavljanje pitanja unutar grupa,
- brisanje pitanja,
- komentiranje na postavljena pitanja,
- brisanje nepoželjnih komentara,
- pozitivno ili negativno ocjenjivanje komentara i
- označivanje komentara kao prihvaćenog odgovora.

Neke od ovih funkcionalnosti ovise o dopuštenjima, odnosno ulogama, koje korisnici imaju. Primjerice, administrator sustava ima ovlasti obrade zahtjeva za kreiranjem grupa i brisanja svog nepoželjnog sadržaja unutar sustava, a samo onaj korisnik koji je postavio neko pitanje može određeni komentar na to pitanje označiti kao prihvaćeni odgovor.

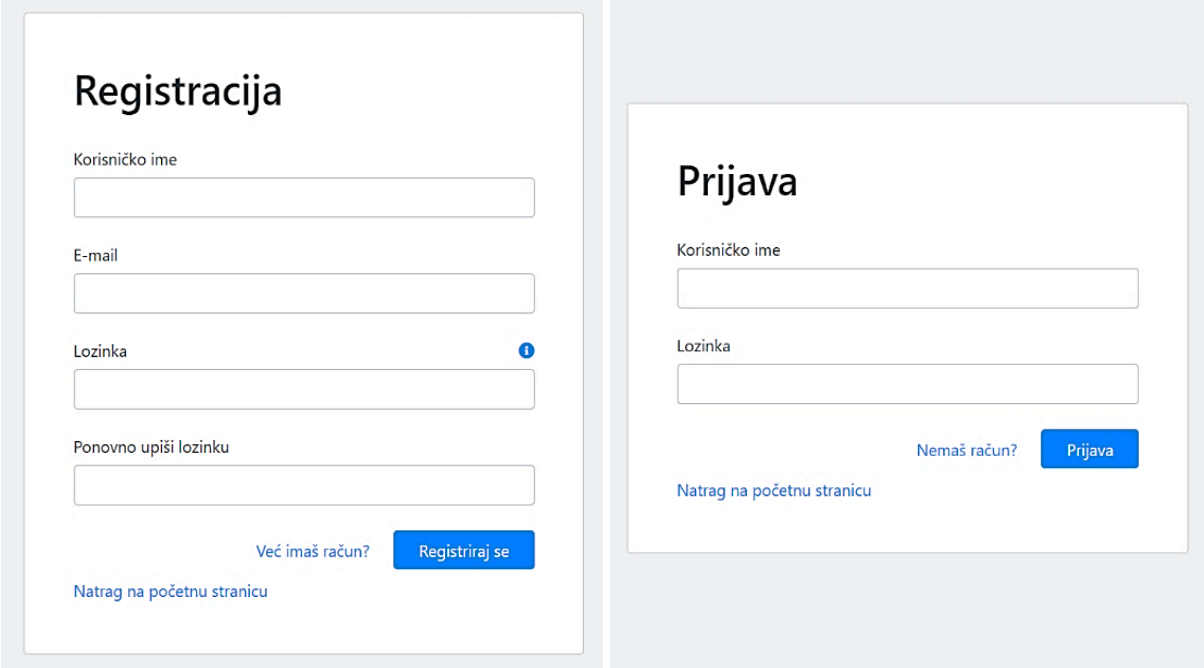
Većina funkcionalnosti nije dostupna korisnicima koji nisu registrirani odnosno prijavljeni u sustav. Oni imaju pristup pregledu postavljenih pitanja i danih komentara kao i pristup pregledu korisničkih profila. Neprijavljeni korisnici su u nekim situacijama potaknuti na registraciju/prijavu u sustav primjerenim porukama, npr. *Da biste odgovorili na ovo pitanje, molimo prijavite se ili registrirajte*. U slučaju pokušaja pristupa ograničenim akcijama oni su jednostavno preusmjereni na stranicu za prijavu.

Bitna značajka aplikacije jest njena sposobnost davanja povratnih informacija. Razlikuju se kratkotrajne i sugestivno obojene obavijesti sustava o uspješnoj ili neuspješnoj provedbi neke korisničke akcije od obavijesti koje korisnik prima u svom sandučiću kada drugi korisnici odrade neku akciju koja je za njega relevantna npr. pozitivno ocjenjivanje komentara.

3.1. Autentifikacija (registracija i prijava)

Autentifikacija je ključna funkcionalnost jer je posjedovanje korisničkog računa preduvjet za interaktivno iskustvo s aplikacijom. Korištena su rješenja integrirana u Django radni okvir koja olakšavaju kreiranje korisničkih računa i postižu željenu razinu sigurnosti prilikom autentifikacije. Implementirani su ugrađeni pogled za prijavu, sustav za *hashiranje* lozinki i sustav za provjeru snage lozinki (korisničke lozinke moraju sadržavati minimalno 8 znakova, uključujući slova i brojeve, ne smiju biti preslične drugim osobnim podacima i ne smiju biti jedna od uobičajenih lozinki npr. *password*).

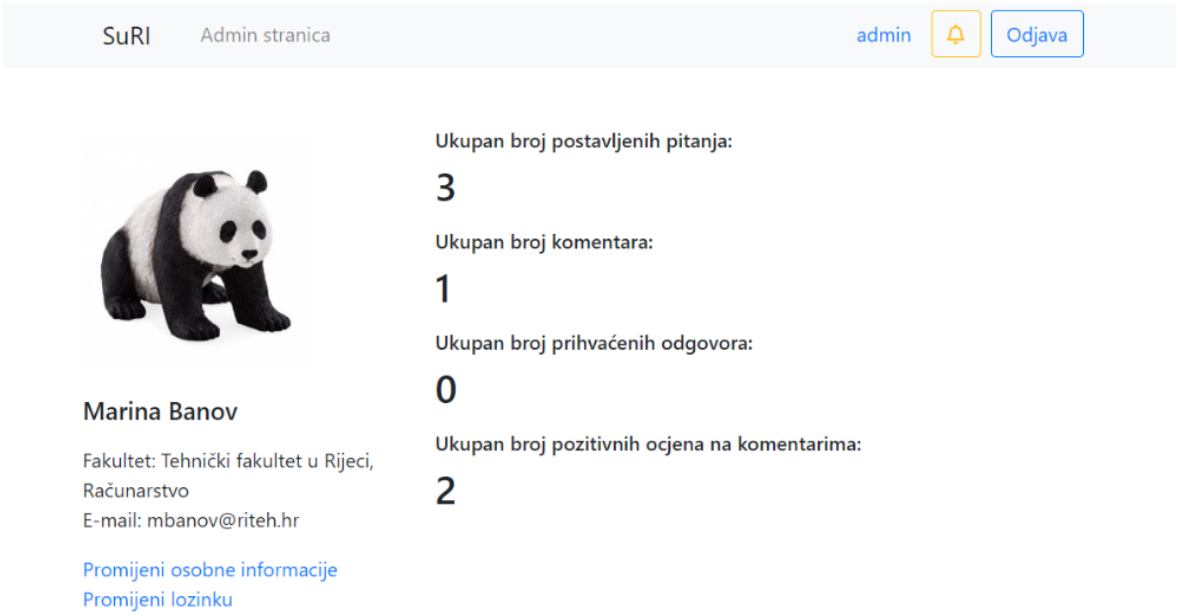
Django nudi i predloške za registraciju i prijavu, ali oni su u ovom slučaju zasebno kreirani zbog potrebe usklađivanja sa sveukupnim dizajnom aplikacije (Slika 3.1.).



Slika 3.1. Obrasci za registraciju i prijavu

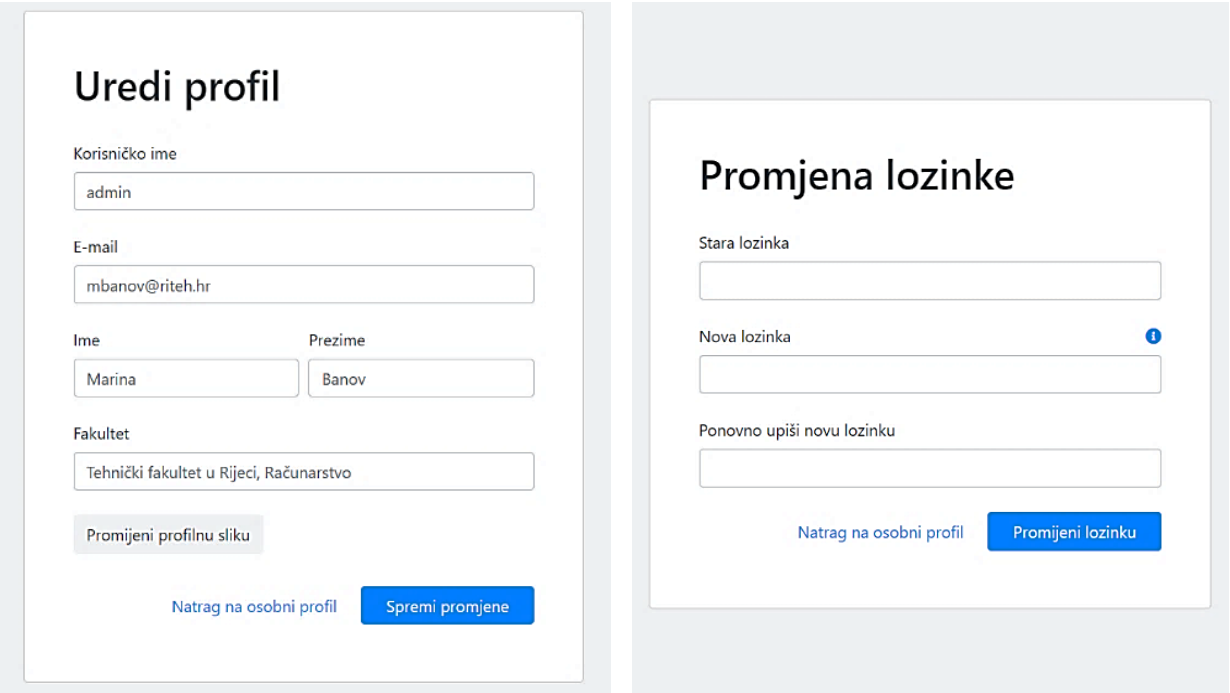
3.2. Pregled korisničkih profila

Na profilnoj stranici (Slika 3.2.) vidljiva je profilna slika, ime, prezime, fakultet na kojem korisnik studira, e-mail adresa kao kontakt te neke statistike koje govore o aktivnosti korisnika i pozitivnom utjecaju koje je on imao na druge. Brojke poput ukupnog broja postavljenih pitanja i ukupnog broja komentara nisu dio same baze podataka već se po potrebi izračunavaju, a u sljedećem poglavlju bit će objašnjeno zašto.



Slika 3.2. Prikaz korisničkog profila

Vlasnici profila mogu promijeniti osobne informacije ili lozinku, što je također jedna od funkcionalnosti koje Django nudi kao dio autentifikacijskog paketa. Slično kao i kod predložaka za registraciju i prijavu, ponuđeni obrasci su prilagođeni kako bi odgovarali cjelokupnom dizajnu aplikacije (Slika 3.3.).



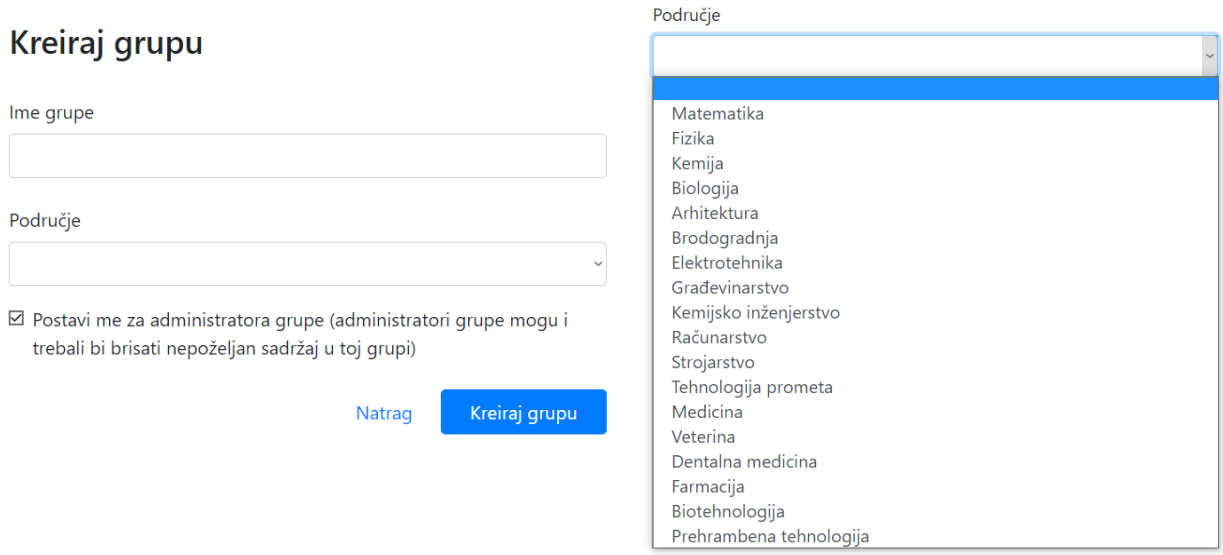
Slika 3.3. Obrasci za uređivanje profila i promjenu lozinke

3.3. Kreiranje grupa povezanih s određenim područjem

Kako bi se olakšalo studentima raznih područja da u mnoštvu sadržaja pronađu upravo ono što im treba, potiče se sistematično kreiranje grupa i postavljanje pitanja i zadataka u odgovarajućim grupama. Svaka grupa ima svoje ime i područje kojem pripada. Primjeri dobro kreiranih grupa bili bi: *Baze podataka – Računarstvo*, *Energetski sustavi – Strojarstvo*, itd.

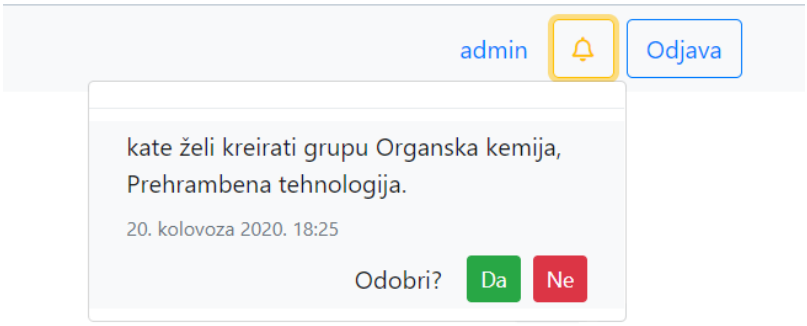
Prilikom kreiranja grupe (Slika 3.4.) potrebno je odabrati jedno od 18 ponuđenih područja definiranih uz pomoć *Pravilnika o znanstvenim i umjetničkim područjima, poljima i granama* [11] imajući na umu da je ova aplikacija namijenjena studentima STEM područja.

Ako je korisnik koji kreira grupu administrator stranice, on automatski postaje i administrator te grupe i njegova je dužnost brisanje nepoželjnog sadržaja. U suprotnom, korisnik se mora izjasniti pristaje li na tu odgovornost. Ako ne pristaje, administrator stranice će opet automatski postati i administrator grupe.



Slika 3.4. Obrazac za kreiranje grupe

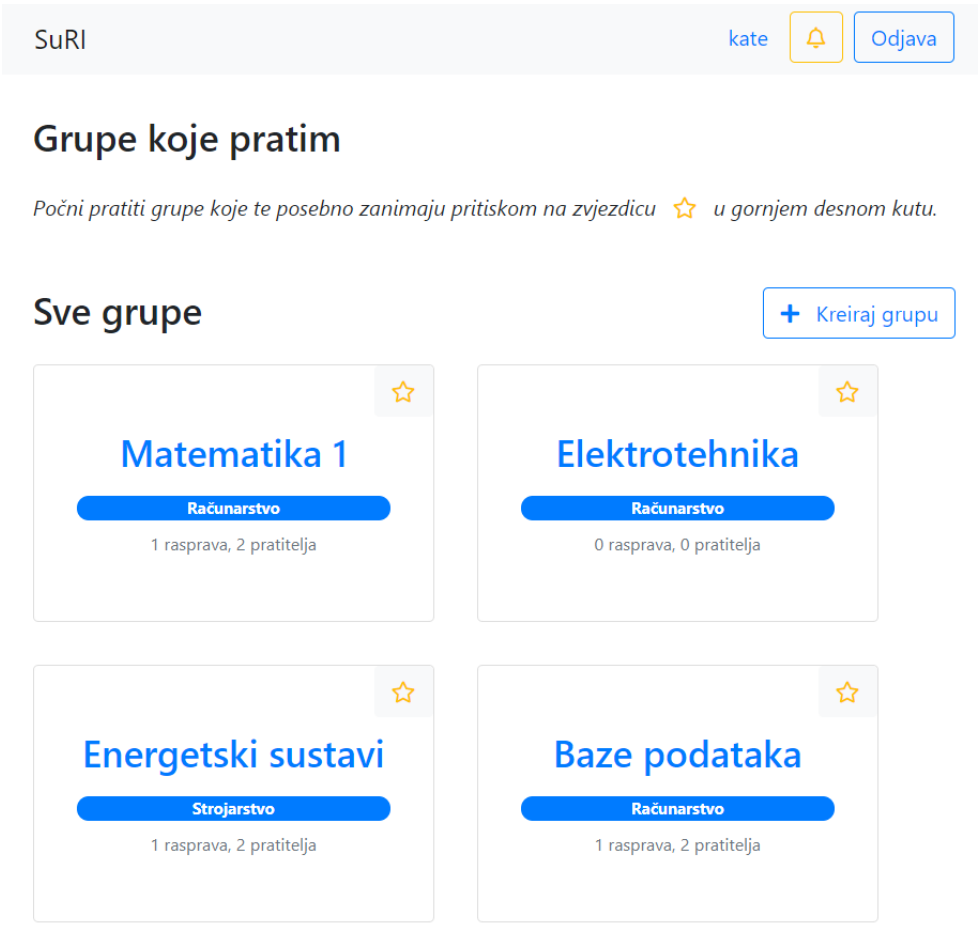
Grupe koje kreiraju obični korisnici (oni koji nisu administratori stranice), ne pojavljuju se odmah u aplikaciji. Prije nego one postanu vidljive, administrator stranice ih mora odobriti putem zahtjeva koji se pojavljuju u njegovom sandučiću (Slika 3.5.).



Slika 3.5. Zahtjev za odobrenje nove grupe

3.4. Praćenje grupa od posebnog interesa

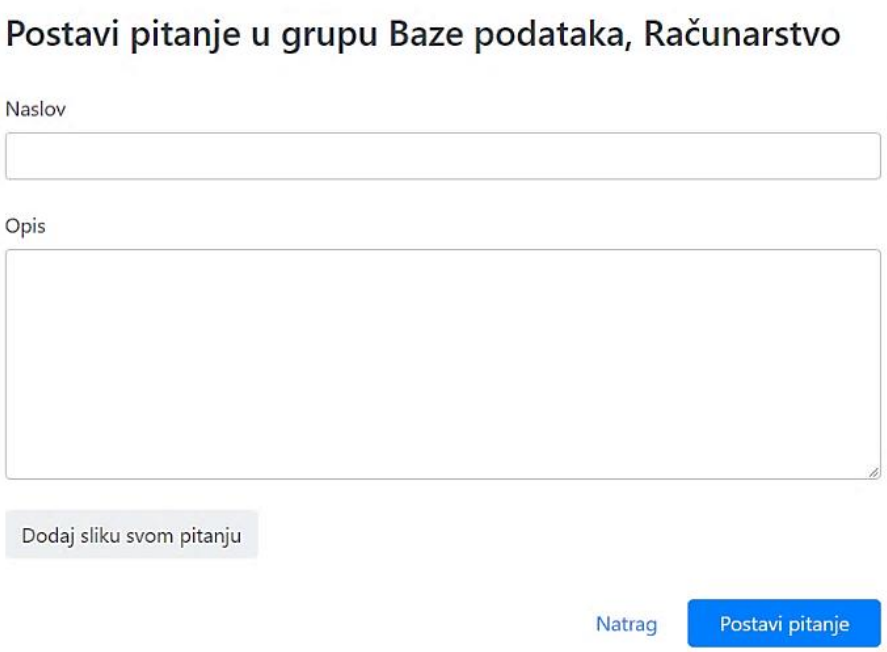
Prijavljeni korisnici mogu na naslovnoj stranici odabrati one grupe koje ih posebno zanimaju i početi ih pratiti klikom na zvjezdicu (Slika 3.6.). Praćenje grupa za korisnika znači dvije stvari: grupe koje prati posebno su istaknute svojim položajem na vrhu naslovne stranice i korisnik će u svom sandučiću dobiti obavijesti kada se u tim grupama postave nova pitanja.



Slika 3.6. Glavna stranica

3.5. Postavljanje pitanja unutar grupa

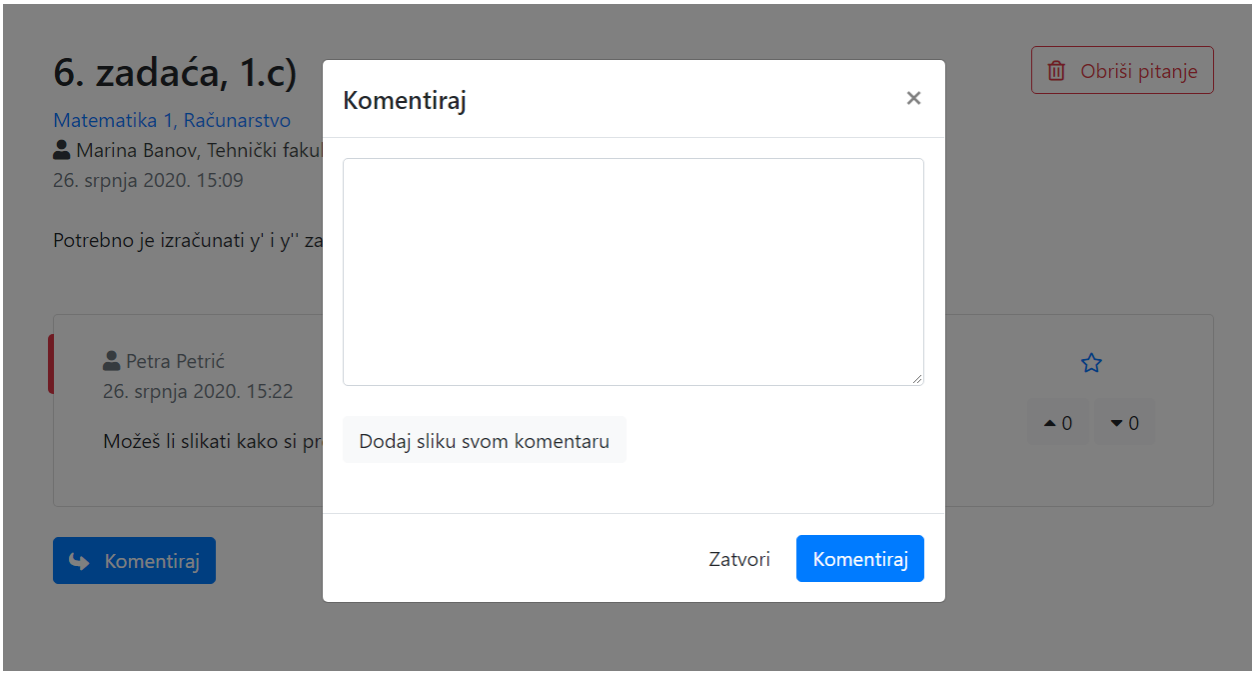
Postavljanje pitanja i zadataka unutar grupa je jednostavan proces: potrebno je posjetiti grupu u koju se želi postaviti pitanje te odrediti kratki naslov i detaljni opis pitanja. Ako je student u nekoj zbirci pronašao zadatak oko kojeg mu treba pomoći, može mobitelom uslikati svoj pokušaj rješavanja i tu sliku pridružiti pitanju postavljenom u web aplikaciji (Slika 3.7.).



Slika 3.7. Obrazac za postavljanje pitanja

3.6. Komentiranje na postavljena pitanja

Obrazac za komentiranje na postavljena pitanja vrlo je sličan obrascu za postavljanje pitanja. Glavna razlika je u tome što se pojavljuje u obliku skočnog prozora (Slika 3.8.), za što je bilo potrebno integrirati paket *django-bootstrap-modal-forms 2.0.0*. Također je moguće dodati sliku komentaru.

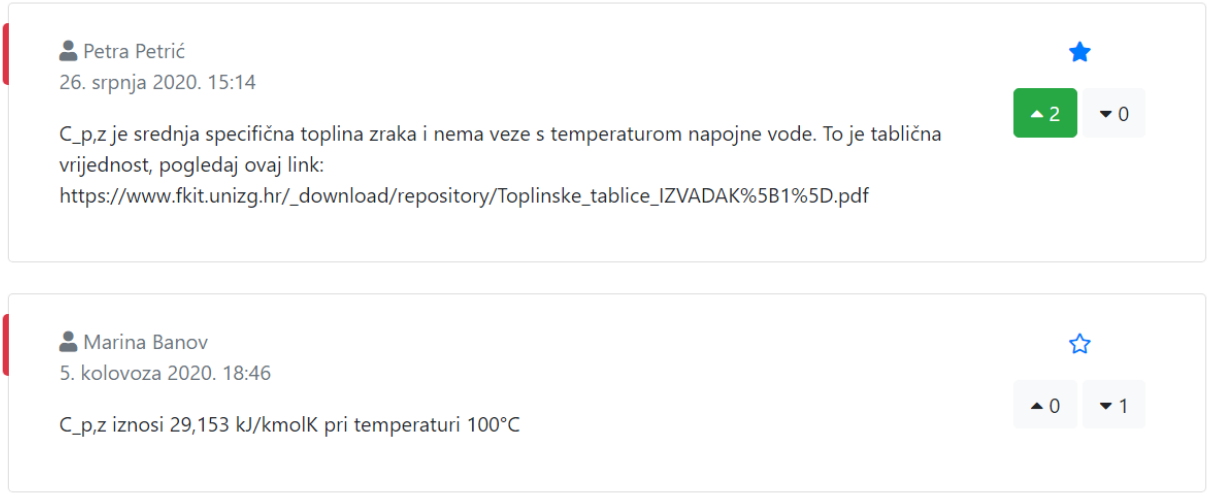


Slika 3.8. Obrazac za ostavljanje komentara

3.7. Pozitivno ili negativno ocjenjivanje komentara

U aplikaciju je integrirana opcija ocjenjivanja komentara, korisna u scenarijima poput ovoga: nakon posljednjih vježbi iz kolegija Energetski sustavi studentu nije bilo posve jasno što je oznaka $C_{p,z}$. U iskušenju je postaviti pitanje u grupu *Energetski sustavi – Strojarstvo*, no primijetio je da je netko drugi prije njega već imao istu poteškoću. Pronašao je pitanje koje opisuje upravo njegovu dvojbu i na svu sreću, dvije kolegice su već pokušale pomoći i dale korisne komentare (Slika 3.9.). Međutim, ta dva komentara nisu usklađena, a on je posve izgubljen u gradivu i ne može dobro procijeniti – koji od njih uzeti u obzir?

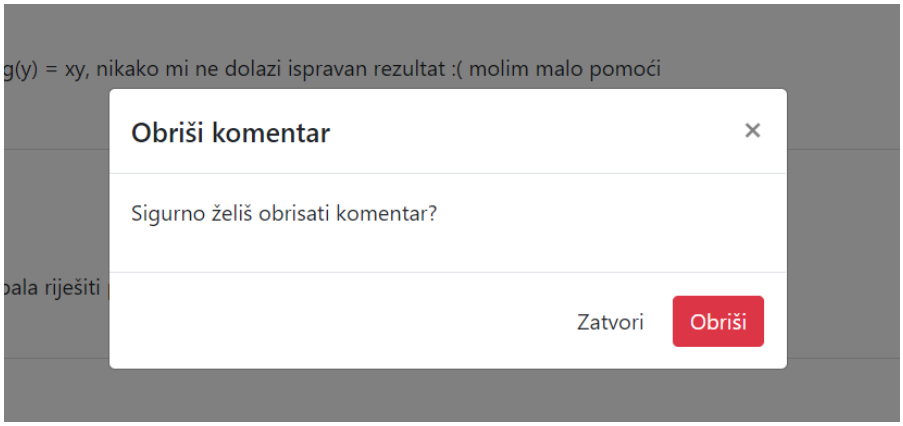
U tom slučaju, na desnoj strani komentara mogu se pronaći neke korisne informacije: koliko je ljudi pozitivno ili negativno ocijenilo komentare te koji od njih ima ispunjenu zvjezdicu – pokazatelj da je taj odgovor prihvaćen od strane onog korisnika koji je postavio pitanje, odnosno da je njemu bio od velike pomoći.



Slika 3.9. Ocjenjivanje komentara

3.8. Brisanje nepoželjnog sadržaja

Brisanje nepoželjnog sadržaja važno je radi održavanja kolegijalne atmosfere, očuvanja razine kvalitete sadržaja i smanjenja preopterećenja baze podataka. Kada se želi obrisati nepoželjna grupa, pitanje ili komentar pojavljuje se skočni prozor poput onog na Slici 3.10. Treba napomenuti da ograničen broj korisnika ima mogućnost brisanja sadržaja: osim administratora stranice, to mogu učiniti samo autori tog nepoželjnog sadržaja i administratori grupa u kojima se taj sadržaj nalazi.



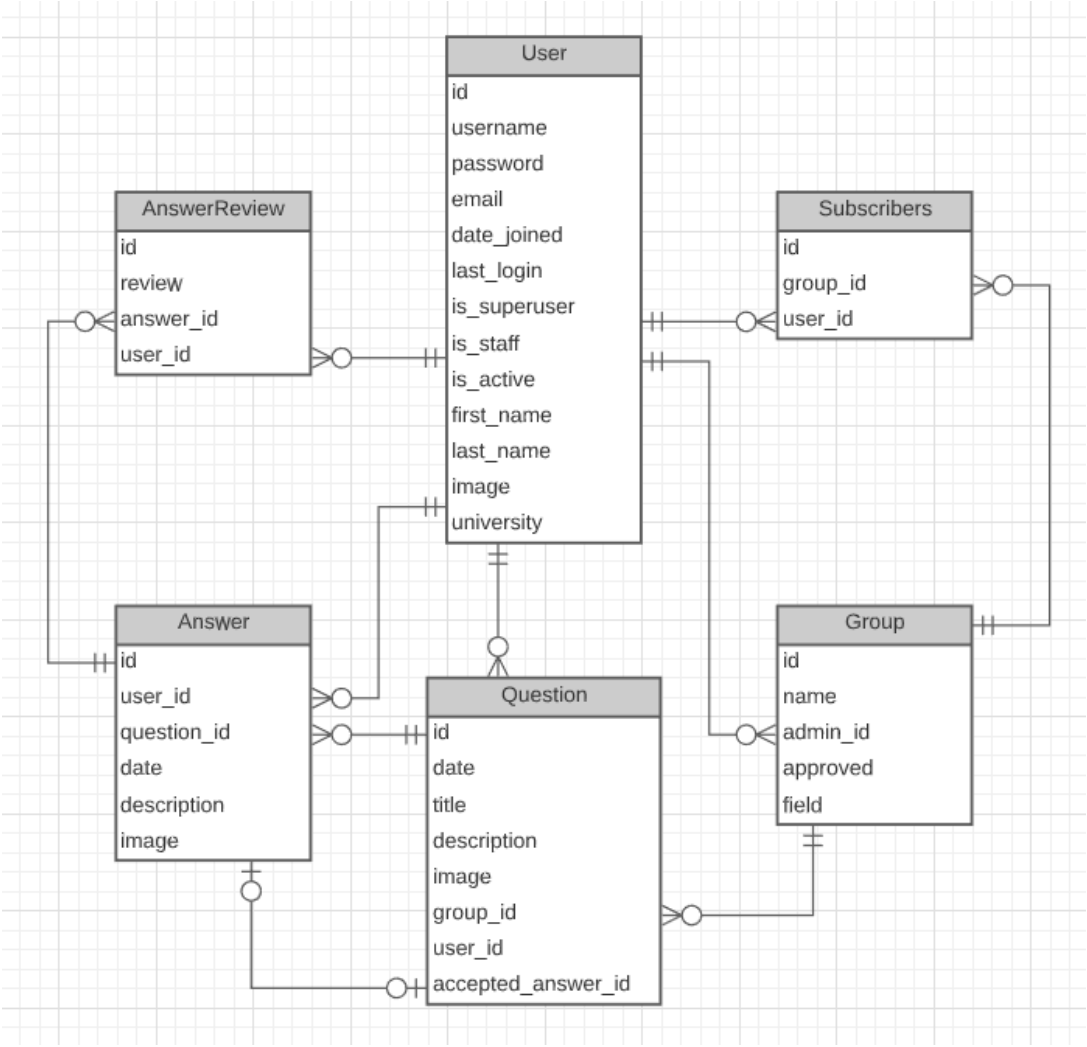
Slika 3.10. Skočni prozor koji traži potvrdu od korisnika

4. DETALJNI PREGLED NA PRIMJERU POSTAVLJANJA PITANJA

U ovom poglavlju bit će istaknuti neki od glavnih koraka razvoja web aplikacije: oblikovanje baze podataka, navigacija i logika obrade obrazaca u pogledima. Za primjer je uzeta funkcionalnost postavljanja pitanja.

4.1. Oblikovanje baze podataka

Razvoj web aplikacije započeo je definiranjem entiteta koji će biti dio baze podataka i relacija među njima. Na Slici 4.1. prikazana je struktura baze podataka koristeći dijagram entiteta i relacija (ER).



Slika 4.1. Dijagram entiteta i relacija

- Entitet *Korisnik* sadrži podatke vezane za sam korisnički račun (ime, prezime, korisničko ime, lozinka, e-mail adresa, datum registracije i posljednje prijave, je li korisnik aktivan, administrator stranice ili dio osoblja) i podatke koje pobliže opisuju korisnika u kontekstu ove aplikacije (profilna slika i fakultet kojem pripada).
- Entitet *Grupa* sadrži ime grupe, područje kojem pripada (koje je element iz skupa unaprijed određenih područja), je li odobrena od strane administratora i time vidljiva na naslovnoj stranici i strani ključ na administratora grupe. Veza *Grupa-KorisnikAdministrator* je vrste n:1 jer svaka grupa ima samo jednog administratora, a pojedini korisnik može biti administrator više grupa.
- Entitet *Pretplate* nastao je preslikavanjem veze tipa m:n koja opisuje kako jedan korisnik može biti pretplaćen na više grupa, a na jednu grupu se može pretplatiti više korisnika. Ovaj entitet nije bilo potrebno definirati u kodu jer ga Djangov mehanizam ORM preslikavanja automatski kreira kada se npr. u entitet *Grupa* dodaje atribut `subscribers` koji je instanca klase `ManyToManyField`.
- Entitet *Pitanje* sadrži datum kada je pitanje postavljeno te naslov, opis i sliku pitanja. Ostali atributi rezultat su preslikavanja veza:
    - *Pitanje-Korisnik* n:1 (pitanje uvijek ima samo jednog autora, ali jedan korisnik može postaviti mnogo pitanja),
    - *Pitanje-Grupa* n:1 (pitanje je uvijek dio samo jedne grupe, ali svaka grupa može sadržavati mnogo pitanja) i
    - *Pitanje-PrihvaćeniOdgovor* 1:1 (autor pitanja može samo jedan odgovor istaknuti kao od posebne pomoći, a svaki odgovor uvijek pripada samo jednom pitanju).
- Entitet *Odgovor* sadrži slične podatke kao i entitet *Pitanje*: datum, opis, sliku, preslikavanje veze *Odgovor-Korisnik* tipa n:1 (jer odgovor uvijek ima samo jednog autora, ali jedan korisnik može ponuditi mnogo odgovora na različita pitanja) i preslikavanje veze *Odgovor-Pitanje* tipa n:1 (jer se neki odgovor uvijek odnosi samo na jedno pitanje, ali neko pitanje može imati mnogo ponuđenih odgovora).
- Entitet *Ocjena* također je nastao preslikavanjem veze m:n (jedan korisnik može ocijeniti više odgovora i jedan odgovor može imati više ocjena), ali sadrži i još jedan podatak koji označava vrijednost ocjene. Važno je razlikovati pozitivne od negativnih ocjena (vrijednost +1 ili -1). Također, treba voditi računa o tome da je zapis korisnik-odgovor uvijek

jedinstven (jedan korisnik ne može na neki odgovor dati više pozitivnih ili više negativnih ocjena).

Svi entiteti sadrže identifikacijski broj.

Kako je već prije spomenuto, Django olakšava razvoj web aplikacija jer se zasniva na objektno-relacijskom mapiranju. Zbog toga nije bilo potrebno pisati SQL naredbe za CRUD operacije niti za inicijalizaciju tablica u bazi, ali je prije početka rada na funkcionalnostima bilo potrebno preslikati definirane entitete u klase, što je napravljeno u datoteci *models.py* koristeći paket *django.db.models*.

Kodni isječak 4.1. pokazuje kako se entitet *Pitanje* preslikava u model. Definira se klasa `Question` koja nasljeđuje Djangovu klasu `Model` i njene atribute koji odgovaraju atributima entiteta.

```
from django.db import models
from django.contrib.auth import get_user_model

class Question(models.Model):
  group = models.ForeignKey(Group, on_delete=models.CASCADE)
  user = models.ForeignKey(get_user_model(), on_delete=models.SET_NULL, null=True)
  date = models.DateTimeField(auto_now_add=True)
  title = models.CharField(max_length=200)
  description = models.CharField(max_length=2048, null=True)
  image = models.FileField(upload_to=update_filename, null=True, blank=True,
                           verbose_name='')
  accepted_answer = models.ForeignKey('Answer', on_delete=models.SET_NULL, null=True,
                                      related_name='accepted_answer')
```

Kodni isječak 4.1. Model Pitanje

Posebnu pozornost treba obratiti na način na koji se definiraju veze između tablica, odnosno na liniju `group = models.ForeignKey(Group, on_delete=models.CASCADE)`. Gledajući ER dijagram, može se vidjeti da će se veza *Pitanje-Grupa* (n:1) u bazi odraziti tako što će se primarni ključ entiteta *Grupa* dodati kao strani ključ entitetu *Pitanje*. Zato je unutar klase `Question`, kod dodavanja atributa `group`, instancirana klasa `ForeignKey`, koja se povezala s klasom `Group` i definirano je da će se prilikom brisanja neke grupe obrisati i sva pitanja koja su bila povezana s njom (`CASCADE`). Na sličan način je oblikovana i veza između pitanja i korisnika koji je

postavio pitanje. S obzirom na to da od svih odgovora na neko pitanje samo jedan može biti označen kao prihvaćeni, radi se o vezi 1:1 koja je opcionalna s obje strane i može se formulirati isto tako.

Drugi atributi (`date`, `title`, `description`, `image`) koji predstavljaju konkretne informacije o objektu su instance odgovarajućih polja – `DateTimeField`, `CharField` i `FileField`. Za datum je dodatno određeno da se automatski sprema ono vrijeme u kojem je pitanje postavljeno. Za sliku koju korisnik može priložiti svom pitanju određena je metoda `update_filename` koja će ju prilikom učitavanja preimenovati kako bi se izbjeglo slučajno prebrisavanje istoimenih slika. Jedinstveno ime datoteka osigurano je funkcijom `uuid4` (Kodni isječak 4.2.).

```
import os
from uuid import uuid4

def update_filename(instance, filename):
  prefix = 'u'
  if isinstance(instance, Question):
    prefix = 'q'
  elif isinstance(instance, Answer):
    prefix = 'a'
  return os.path.join('images/',
                      '{}_{}.{}'.format(prefix, uuid4().hex, filename.split('.')[-1]))
```

Kodni isječak 4.2. Automatsko preimenovanje datoteka

Model Pitanje i odgovarajuća tablica u bazi sadrže sve atribute koji su krajnjem korisniku vidljivi. Međutim, to nije uvijek slučaj što se može vidjeti na primjeru statistike koja je vidljiva na korisničkim profilima i koja ukazuje na pouzdanost odgovora (Slika 3.2.). Ranije je spomenuto od kojih se sve atributa sastoji entitet Korisnik, a pritom nije naveden ukupan broj postavljenih pitanja, ukupan broj komentara ni druge. Ti se podaci izračunavaju po potrebi, a u Kodnom isječku 4.3. prikazano je kako. Klasa `User` osim atributa sadrži svojstva, odnosno funkcije koje izračunavaju one podatke koje nije potrebno držati u bazi jer su izvedeni iz nekih drugih podataka. Takav pristup garantira točan prikaz statistike, smanjuje rizik od pogreške prilikom brisanja pitanja i odgovora te smanjuje redundanciju u bazi.

```
from django.db import models
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
  ...
  @property
  def questions_asked(self):
    return Question.objects.filter(user=self).count()

  @property
  def answers_count(self):
    return Answer.objects.filter(user=self).count()
  ...
```

Kodni isječak 4.3. Svojstva koja se izračunavaju po potrebi

4.2. Navigacija

Kada korisnik želi postaviti pitanje, u MVT arhitekturi to znači da želi pristupiti pogledu koji će mu prikazati obrazac za postavljanje pitanja i koji će obraditi podatke primljene putem tog obrasca. Pogledi su jednoznačno povezani sa svojom URL adresom i pristupa im se upisivanjem željene putanje u adresnu traku ili klikom na poveznice. Putanje do svih pogleda nalaze se u datoteci *urls.py*, čiji je manji dio prikazan u Kodnom isječku 4.4.

Putanje se označavaju u obliku `path(route, view, name)`. `route` je uzorak URL-a. Pri obradi zahtjeva, Django unutar popisa `urlpatterns` traži prvi uzorak koji se podudara s traženim URL-om. URL uzorci mogu sadržavati promjenjive vrijednosti kao primjerice `'group/<int:group_id>/'`. To znači da su zahtjevi `'group/1/'` i `'group/6532/'` jednako validni i da će se korisniku prikazati stranica željene grupe ako postoji grupa s identifikacijskim brojem 1 ili 6532, jer će Django pronaći odgovarajući URL uzorak i pozvati funkciju pogleda `view`. Svi pogledi su definirani u datoteci *views.py* bilo kao funkcija ili kao klasa.

```
from django.contrib.auth.decorators import login_required
from django.urls import path
from . import views

urlpatterns = [
  path('', views.index, name='index'),
  path('group/<int:group_id>/', views.group, name='group'),
  path('create_question/<int:group_id>/', login_required(views.CreateQuestion.as_view()),
       name='create_question'),
  path('question/<int:question_id>/', views.question, name='question'),
  ...
]
```

Kodni isječak 4.4. urls.py

Pogledi definirani kao funkcije najčešće su jednostavniji i logiku svode na preusmjeravanje korisnika i prikazivanje sadržaja. Primjer za to je pogled `group` (Kodni isječak 4.5.). Kao što je već rečeno, ruta koja poziva ovaj pogled mora sadržavati identifikacijski broj grupe kojoj se želi pristupiti. Prvi argument funkcije je objekt tipa `HttpRequest` a drugi je identifikacijski broj grupe. U pogledu se dohvaćaju potrebni podatci (na ORM način, bez pisanja SQL upita) i korisniku se vraća `HttpResponse` objekt odnosno prikaz predloška u koji je ukomponiran kontekst.

```
from .models import Group, Question
from django.shortcuts import render

def group(request, group_id):
    context = {
        'group': Group.objects.get(id=group_id),
        'questions': Question.objects.filter(group=group_id)
    }
    return render(request, 'main_app/group.html', context)
```

Kodni isječak 4.5. Pogled kao funkcija

Pogledi definirani kao klase imaju složeniju logiku, očekuju više interakcije s korisnikom i najčešće moraju obrađivati obrasce. U Kodnom isječku 4.6. vidljiv je samo onaj dio pogleda `CreateQuestion` koji se odnosi na prikaz predloška. Kasnije će biti objašnjeno na koji se način obrađuju podaci primljeni iz obrasca.

```
from django.views.generic import CreateView
from .forms import QuestionForm

class CreateQuestion(CreateView):
  form_class = QuestionForm
  template_name = 'main_app/create_question.html'
  ...
```

Kodni isječak 4.6. Pogled kao klasa

Svaka putanja ima i svoje ime. Imenovanjem putanje omogućeno je jednostavnije preusmjeravanje s bilo kojeg mjesta u projektu, a najčešće se pojavljuje u predlošcima. U Kodnom isječku 4.7., koji je dio predloška za prikaz jedne grupe, može se vidjeti kako funkcionira preusmjeravanje pomoću Django putanja.

```
{% if user.is_authenticated %}
<a href="{% url 'create_question' group.id %}" class="btn btn-outline-primary">
  <i class="fas fa-question"></i>
  <span class="pl-2">Postavi pitanje</span>
</a>
{% else %}
<p class="mt-2"><i class="text-secondary">
  Da biste postavili pitanje u ovu grupu, molimo
  <a href="{% url 'login' %}?next={% url 'create_question' group.id %}">prijavite se</a> ili
  <a href="{% url 'register' %}?next={% url 'create_question' group.id %}">registrirajte</a>.
</i></p>
{% endif %}
```

Kodni isječak 4.7. Poveznica do obrasca za postavljanje pitanja

U slučaju da je korisnik prijavljen prikazuje se dugme (poveznica je oblikovana koristeći Bootstrap klase i Fontawesome ikone) koji će ga preusmjeriti na obrazac za postavljanje pitanja zbog ovog dijela koda: `href="{% url 'create_question' group.id %}"`. Ako korisnik nije prijavljen prikazuju mu se poveznice na obrasce za prijavu ili registraciju. Nakon uspješne autentifikacije bit će automatski preusmjeren na obrazac za postavljanje pitanja: `href="{% url 'login' %}?next={% url 'create_question' group.id %}"`.

4.3. Logika obrade obrazaca u pogledima

Jednom kada je korisnik došao na željenu stranicu, prikazuje mu se obrazac za postavljanje pitanja. Obrazac se temelji na klasi `QuestionForm` vidljivoj u Kodnom isječku 4.8. Ona od svih atributa modela `Question` odabire one koje će korisnik morati upisati dok kreira pitanje, a to su naslov, opis i slika. Web aplikacija može sama pohraniti datum i vrijeme nastajanja pitanja i strane ključeve na grupu u kojoj se pitanje objavljuje i korisnika. Strani ključ na prihvaćeni odgovor ne može se definirati prije nego što je pitanje uopće postavljeno tako da nema smisla da taj atribut bude vidljiv prilikom postavljanja pitanja.

```
from django.forms import ModelForm
from .models import Question

class QuestionForm(ModelForm):
  class Meta:
    model = Question
    fields = ('title', 'description', 'image')
```

Kodni isječak 4.8. forms.py

Kada korisnik unese naslov, opis i sliku i pritisne na dugme *Postavi pitanje* pogled `CreateQuestion` će pozvati funkciju `form_valid` (Kodni isječak 4.9.). Ta funkcija kreira novu instancu klase `Question`, sprema ju u bazu i preusmjerava korisnika na stranicu na kojoj je prikazano novopostavljeno pitanje.

```
def form_valid(self, form):
  g = Group.objects.get(id=self.kwargs['group_id'])
  q = Question(
    group=g,
    user=self.request.user,
    title=form.cleaned_data['title'],
    description=form.cleaned_data['description'],
    image=self.request.FILES.get('image')
  )
  q.save()
  return HttpResponseRedirect(reverse('question', kwargs={'question_id': q.id}))
```

Kodni isječak 4.9. Obrada obrasca i pohranjivanje pitanja

Strani ključ za grupu u kojoj se objavljuje dobiva se preko URL putanje koja sadrži identifikacijski broj grupe (`g = Group.objects.get(id=self.kwargs['group_id'])`). Korisnik koji je postavio pitanje je Djangu poznat i dohvatljiv preko izraza `self.request.user`. Naslov i opis pitanja su podaci koji se dohvaćaju direktno iz obrasca, a slika (s obzirom na to da se radi o binarnoj datoteci, a ne čistom tekstu) se pohranjuje u `self.request.FILES`.

Još jedan dio ove funkcije je istaknut u Kodnom isječku 4.10. Radi se o logici za slanje obavijesti svim korisnicima koji su pretplaćeni na tu grupu. Za razvoj te funkcionalnosti korišten je paket *django-notifications*, a ključna je bila klasa `Notification` i funkcija `notify.send`.

```
for s in g.subscribers.exclude(id=self.request.user.id):
  if s.notifications.filter(verb='Ima novih pitanja u grupi',
                            target_object_id=g.id).count() == 0:
    notify.send(sender=self.request.user,
                recipient=s,
                verb='Ima novih pitanja u grupi',
                target=g)
```

Kodni isječak 4.10. Logika slanja obavijesti

Svaki put kada neki pogled pozove funkciju `notify.send()` s definiranim pošiljateljem, primateljem i sadržajem obavijesti, u sandučiću primatelja dodaje se nova instanca klase

`Notification`. U ovom slučaju želi se obavijestiti sve korisnike koji su se pretplatili na tu grupu osim, naravno, onog korisnika koji je objavio pitanje da su objavljena nova pitanja (`for s in g.subscribers.exclude(id=self.request.user.id)`). Kako se sandučić korisnika ne bi previše natrpao, ta obavijest će se poslati samo ako se u njegovom sandučiću još ne nalazi takva obavijest.

5. ZAKLJUČAK

U sklopu ovog rada izrađena je i opisana web aplikacija za učenje i dijeljenje znanja iz STEM područja unutar internetske zajednice studenata. Za razliku od LMS-ova koji su već u upotrebi, ova aplikacija osmišljena je tako da bude atraktivna studentima koji žele neformalno dijeliti znanje i istraživati interdisciplinarne teme STEM područja. Primarno je orijentirana na interakciju putem postavljanja pitanja i davanja odgovora, ali nudi i druge funkcionalnosti poput grupiranja srodnih pitanja, praćenja grupa od posebnog interesa, ocjenjivanja tuđih odgovora, prikaza pouzdanosti dosadašnjih odgovora na korisničkim profilima i primanja obavijesti u sandučiću.

Od korištenih tehnologija posebno se ističe Django radno okruženje, s obzirom na to da se razvoj aplikacije temeljio na njegovoj MVT arhitekturi. Django potiče programera na pisanje modularnog koda i aplikacija koje je lako nadograditi. Također, olakšava dio komunikacije s bazom podataka jer nije potrebno pisati SQL komande da bi se čitalo ili mijenjalo podatke. Oni su uz korištenje ORM paradigme predstavljeni kao objekti s atributima i metodama.

Korištenje Bootstrapovih komponenti omogućilo je razvoj potpuno responzivne aplikacije koja se uspješno renderira na uređajima svih veličina zaslona, a njen dizajn nadopunio je dinamičnim elementima ugodnog dizajna.

Predstavljeni alati pojednostavljuju razvoj funkcionalnosti i odvajaju poslužiteljsku logiku od klijentskih zahtjeva. Struktura programskog koda podijeljena na datoteke namijenjene za definiciju modela, pogleda i predložaka omogućuje jednostavnu implementaciju ekspanzija u slučaju da se pojavi želja za proširenjem funkcionalnosti. To bi moglo uključivati primjerice mogućnost čavrljanja (engl. *live chat*) i grupnog čavrljanja.

LITERATURA

[1] Grgić, I. H.: „Otvorenost u znanosti i visokom obrazovanju“, Školska knjiga, Zagreb, 2018.

[2] Stack Exchange Inc: „Stack Overflow – Where Developers Learn, Share, & Build Careers“, s Interneta, https://stackoverflow.com, 8. rujna 2020.

[3] Django Software Foundation: „The Web framework for perfectionists with deadlines | Django“, s Interneta, https://www.djangoproject.com, 21. srpnja 2020.

[4] The SQLite Consortium: „SQLite Home Page“, s Interneta, https://sqlite.org/index.html, 8. rujna 2020.

[5] Otto, M.; Thornton, J. i Bootstrap suradnici: „Bootstrap · The most popular HTML, CSS, and JS library in the world“, s Interneta, https://getbootstrap.com, 21. srpnja 2020.

[6] The jQuery Foundation: „jQuery“, s Interneta, https://jquery.com, 8. rujna 2020.

[7] Software Freedom Conservancy: „Git“, s Interneta, https://git-scm.com, 8. rujna 2020.

[8] Marsilio, A.: „Django“, s Interneta, http://alessandromarsilio.com/django/, 8. rujna 2020.

[9] Fonticons, Inc.: „Font Awesome“, s Interneta, https://fontawesome.com, 8. rujna 2020.

[10] CodeSeven: „Toastr by CodeSeven“, s Interneta, https://codeseven.github.io/toastr/, 8. rujna 2020.

[11] Rektorski zbor: „Pravilnik o znanstvenim i umjetničkim područjima, poljima i granama“, neslužbeni pročišćeni tekst, Narodne novine, br. 118/09, 82/12 i 32/13, 34/16, Zagreb, 2016.

POPIS SLIKA

POPIS KODNIH ISJEČAKA

SAŽETAK

U ovom radu opisana je web aplikacija koja potiče suradnju i dijeljenje znanja među studentima STEM područja. Korisnici mogu formirati i pratiti grupe u koje će objavljivati pitanja i zadatke vezane za određenu temu, nuditi vlastita rješenja i ocjenjivati tuđa rješenja, a o svim relevantnim događajima obaviješteni su putem kratkih poruka u sandučiću. Ovakav alat koristan je u kontekstu održavanja nastave na daljinu i predstavlja rješenje za neformalnu komunikaciju usmjerenu na učenje. U radu je također opisano korištenje Django radnog okvira u svrhu izrade poslužiteljskog dijela web aplikacije te prednosti njegovog autentifikacijskog paketa, MVT arhitekture i ORM paradigme. Od ostalih tehnologija bitnih za razvoj klijentskog dijela aplikacije ističu se Bootstrap i JavaScript knjižnice jQuery i *toastr.js* koje su doprinijele stvaranju intuitivnog i responzivnog dizajna.

Ključne riječi: Django, pitanja i odgovori, suradnja, učenje na daljinu, web aplikacija.

ABSTRACT

This paper describes a web application that encourages collaboration and knowledge sharing among STEM students. Users can create and follow groups in which they will post questions and tasks related to a particular topic, offer their own solutions and evaluate other people's solutions, and all relevant events are notified via short messages in their mailbox. This tool is useful in the context of distance learning and it is a solution for informal communication focused on learning. The paper also describes the use of the Django framework for the purpose of creating the server part of the web application and the advantages of its authentication package, MVT architecture and ORM paradigm. Other technologies important for the development of the client part of the application include Bootstrap and JavaScript libraries jQuery and *toastr.js*, which contributed to the creation of an intuitive and responsive design.

Keywords: Django, questions and answers, cooperation, online learning, web application.