IZRADA WEB APLIKACIJE ZA PRAĆENJE NATJECATELJSKIH TURNIRA

Šarić, Filip

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:969185

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-12-04**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

FILIP ŠARIĆ

Z A V R Š N I R A D

IZRADA WEB APLIKACIJE ZA PRAĆENJE NATJECATELJSKIH TURNIRA

Split, srpanj 2020.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

Predmet: Programiranje na internetu

Z A V R Š N I R A D

Kandidat: Filip Šarić

Naslov rada: Izrada web aplikacije za praćenje natjecateljskih turnira

Mentor: Marina Rodić, predavač

Split, srpanj 2020.

Sadržaj

Sažetak

Cilj rada je izrada mrežne (engl. *web*) aplikacije za praćenje natjecateljskih turnira u nogometu, koja će omogućiti svim sudionicima jednostavan uvid u stanje pojedinog turnira.

Za izradu rada koristi se Microsoftov uređivač teksta Visual Studio Code. Aplikacija je izrađena u razvojnom okviru Django. Baza podataka je poslužena na Apache poslužitelju koji je podignut na računalu koristeći XAMPP paket. Za stiliranje HTML (HyperText Markup Language) stranica koristi se razvojni okvir Bootstrap.

Trenutno u aplikaciji postoji mogućnost generiranja dva tipa turnira, jedan je eliminacijski tip turnira, a drugi grupna faza i eliminacija. Detaljan opis pojedinog tipa turnira slijedi u nastavku dokumenta.

Ključne riječi: mrežna aplikacija, Visual Studio Code, razvojni okvir Django, MySQL, XAMPP

Summary

Web application development for competitive football tournaments tracking

The purpose of this work is to create a web application for tracking competitive football tournaments which will allow it's participants to monitor activities in tournaments.

The code is written in Microsoft's Visual Studio Code source code editor. The application is developed in Django Framework using MySQL database. The database is hosted on Apache web server which is set up on local computer using XAMPP web server stack package. Bootstrap framework is used for styling HTML (HyperText Markup Language) pages.

Currently, there are two tournament types in the application. One is elimination type and the second is group phase and elimination. Both types are described in detail in this document.

Keywords: web application, Visual Studio Code, Django Framework, MySQL, XAMPP

1. Uvod

Rezultat ovog završnog rada je aplikacija koja ima sve bitne komponente koje jedan nogometni turnir treba imati. Registracija ekipe, registracija igrača u ekipi, prijava ekipe na turnir, iščekivanje dana objave grupa ili suparnika u eliminacijskoj fazi, komentiranje rezultata u turniru i brojne druge funkcionalnosti predstavljaju sve te bitne komponente.

Izrađena aplikacija omogućava detaljan pregled svakog turnira, ekipe i igrača. Bilo koji korisnik u svakom trenutku može vidjeti sve odigrane i buduće utakmice na pojedinom turniru, strijelce golova u utakmicama, najbolje strijelce na pojedinom turniru, statistiku za pojedinu ekipu i statistiku za pojedinog igrača. Korisnici aplikacije mogu kreirati ekipe, dodavati igrače u ekipe i prijavljivati svoje ekipe na turnire. Organizator turnira može kreirati turnir, generirati utakmice, unositi rezultate utakmica i zaključiti turnir. Postoji i uloga upravitelja (engl. *administrator)* koji može upravljati svim turnirima, korisnicima i ekipama. Aplikacija ima i socijalnu komponentu u vidu komentiranja turnira i reputacije korisnika koju si korisnici međusobno dodjeljuju.

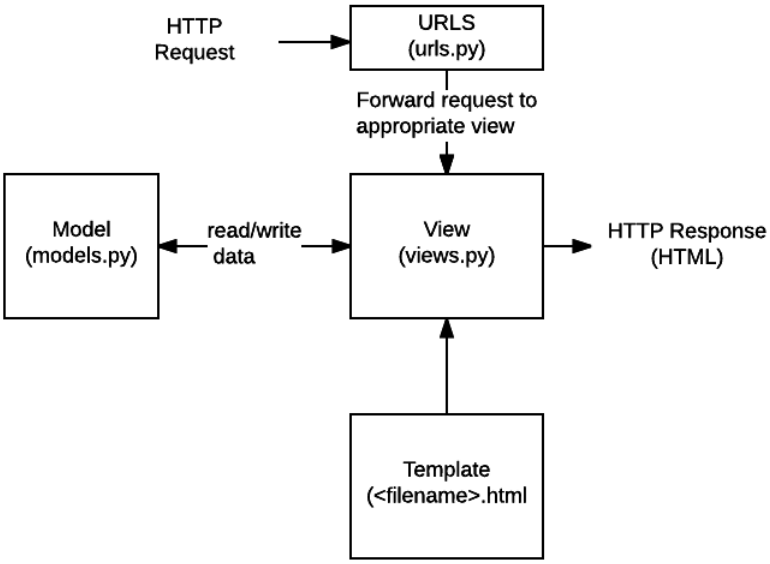
Danas se većina regionalnih, lokalnih i županijskih turnira organizira po principu da se prijava ekipe vrši telefonski, poštom ili dolaskom na određeno mjesto. Izvlačenje papirića ili slične metode, koje ne garantiraju nepristranost, i dalje su sveprisutne. Ova aplikacija zamjenjuje tradicionalne načine upravljanja turnirima i omogućuje sveobuhvatno rješenje za prijavu, izvlačenje parova ili grupa i praćenje turnira.

U sljedećem poglavlju su opisane tehnologije korištene za izradu aplikacije. Treće poglavlje sadrži prikaz E-V (Entiteti-Veze) dijagrama baze podataka aplikacije. Nakon toga, u tri poglavlja opisane su funkcionalnosti aplikacije za sve uloge (engl. *role*). Realizacija socijalne komponente aplikacije i upravljanje korisničkim podacima opisano je u poglavlju 7. Na kraju se nalaze poglavlja zaključak i popis literature.

2. Tehnologije

2.1. Razvojni okvir Django

Django razvojni okvir [1], u daljnjem tekstu samo Django, je razvojni okvir otvorenog koda koji omogućava brz razvoj sigurnih i lako održivih mrežnih aplikacija.. Nastao je 2003. godine, a objavljen je 2005. godine pod BSD (Berkeley Software Distribution) dozvolom (engl. *license*). Koristi MVT (*Model View Template*) arhitekturu, koja je slična MVC (*Model View Controler*) arhitekturi. Po zadanim postavkama Django koristi SQLite bazu podataka. Arhitektura je prikazana na slici 1.



Slika 1. Arhitektura Django projekta

Model označava strukturu podataka koji će se spremiti u bazu podataka, određuje se u obliku klasa, programskog jezika Python, koje se definiraju u datoteci `models.py`. U datoteci `urls.py` definiraju se rute (engl. *path*) koje se koriste za preusmjeravanje HTTP (HyperText Transfer Protocol) zahtjeva na odgovarajući pogled (engl. view). Ono što je u MVC aritekturi *controller*, u MVT arhitekturi je *view*, odnosno on je zadužen za primanje i obradu HTTP zahtjeva. Nakon obrade zahtjeva, *view* vraća HTTP odgovor. *Template* je najčešće HTML stranica na kojoj se prikazuje stvarni odgovor korisniku u pregledniku (engl. *browser*). *Template* može biti bilo koja vrsta datoteke, ne nužno HTML datoteka.

2.2. Razvojni okvir Bootstrap

Razvojni okvir Bootstrap [2] je razvojni okvir otvorenog kôda koji se koristi za stiliziranje HTML stranica. On je najpopularniji CSS (Cascading Style Sheets) razvojni okvir s kojim je moguće razvijati responzivne (engl. *responsive*) mrežne stranice. Najnovija verzija je verzija 4. Sastoji se od velikog broja unaprijed definiranih formi (engl. *form*), dugmadi (engl. *button*), navigacija (engl. *navigation*), kartica, elemenata i sličnih komponenti koje čine jedno korisničko sučelje.

Korištenjem Bootstrap razvojnog okvira moguće je znatno povećati brzinu razvoja mrežnih aplikacija zbog korištenja tisuća već napisanih linija kôda. Na taj način se ostvaruje konzistentnost jer će se kroz aplikaciju koristiti elementi istih ili sličnih karakteristika.

Sve već definirane elemente je na jednostavan način moguće urediti. Vrlo jednostavno se promijeni boja, obrub, veličina ili neko drugo svojstvo.

Jedna od najvećih prednosti ovog razvojnog okvira je prilagodljivost mobilnim uređajima, obzirom da su svi elementi responzivni.

Zbog velike popularnosti ovog razvojnog okvira podrška (engl. *support*) je odlična. Postoji i velika zajednica programera koji rade na njegovom razvoju te je za očekivati je da će s vremenom ta podrška dodatno rasti.

2.3. MySQL

MySQL [3] je besplatni sustav otvorenog kôda za upravljanje bazama podataka. Objavljen je pod GNU General Public License dozvolom. U vlasništvu je švedske tvrtke MySQL AB. Programski kôd MySQL-a je napisan u programskim jezicima C i C++. Zadnja verzija je verzija 8.0 iz 2019. godine. MySQL baza je relacijska baza podataka i ima podršku brojnih programskih jezika kao što su Python, Perl, PHP i Java. MySQL je osvojio veliki dio tržišta jer je besplatan i otvorenog koda.

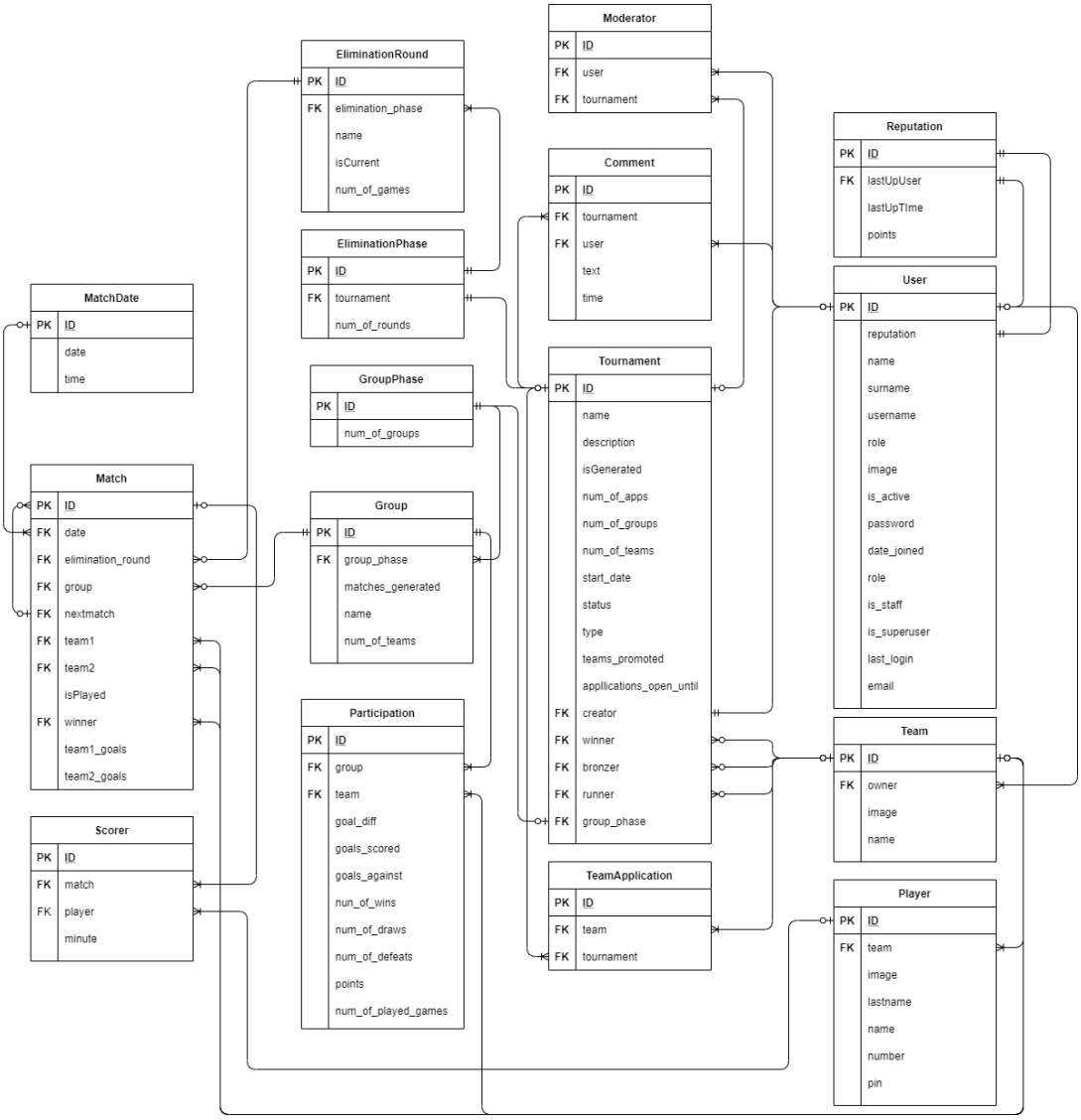
2.4. XAMPP

XAMPP [4] je paket koji se koristi u razvoju mrežnih aplikacija, a sastoji se od Apache HTTP poslužitelja, MariaDB baze podataka i prevoditelja (engl. *interpreter*) za skripte napisane u programskim jezicima PHP i Perl. Razvijen je od strane grupe Apache Friends. Najčešće se koristi za podizanje lokalnog test poslužitelja za razvoj mrežnih aplikacija. MariaDB je softver koji je nastao kao *fork* (softver koji je proizišao kao modifikacija softvera pod GNU GPL dozvolom) MySQL-a.

Većina produkcijskih mrežnih poslužitelja koristi iste komponente kao XAMPP. Time je prijelaz sa lokalnog, na produkcijski poslužitelj na mreži, olakšan.

3. Model baze podataka

Model je prikazan u obliku E-V (Entiteti-Veze) dijagrama na slici 2.



Slika 2. E-V dijagram

Entitet Tournament može imati nula ili više komentara koje postavljaju korisnici (entitet Comment), nula ili više prijava ekipa (entitet TeamApplication), nula ili jednu eliminacijsku fazu (entitet EliminationPhase) i nula ili jednu grupnu fazu (entitet GroupPhase). Turnir također može imati i moderatore u vidu korisnika koji mogu upravljati turnirom uz samog kreatora turnira, uz ograničene ovlasti

Turnir koji je eliminacijskog tipa ima jednu eliminacijsku fazu koja ima više eliminacijskih rundi (npr. osmina finala, polufinale i slično), a svaka eliminacijska runda ima više utakmica. Grupnu i eliminacijsku fazu ima turnir tipa grupa + eliminacija. Grupna faza sastoji se od više grupa, a u svakoj grupi ima više utakmica. Sudjelovanje ekipe u grupi i ostvareni uspjeh prati se kroz entitet Participation.

Svaka utakmica može i ne mora pripadati grupi, ovisno igra li se u grupi ili eliminacijskoj rundi. Isto vrijedi i za pripadnost utakmice eliminacijskoj rundi. Utakmica ima nula ili više strijelaca, koji pripadaju nekoj ekipi. Utakmica ima i svoj termin odigravanja. Moguće je da utakmica ima i vezu na neku drugu utakmicu, ako je riječ o utakmici koja pripada nekoj eliminacijskoj rundi.

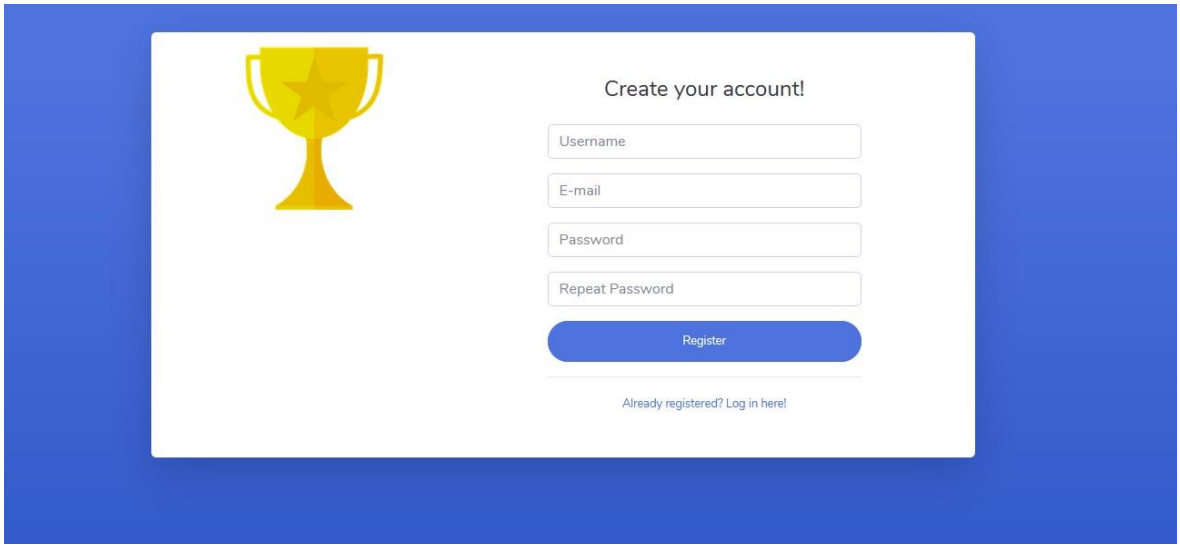
Detaljan opis načina interakcije entiteta se nalazi u nastavku.

4. Funkcionalnosti aplikacije za ulogu sudionika

U ovom poglavlju će biti opisane sve funkcionalnosti aplikacije koje može koristiti korisnik s ulogom sudionika u turniru. Funkcionalnosti koje su zajedničke svim ulogama će se pojasniti samo jednom.

4.1. Registracija

Korisnici pri registraciji moraju popuniti 4 polja: korisničko ime, e-mail adresa, lozinka i ponovljena lozinka. Registracijska forma je vidljiva na slici 3.



Slika 3. Registracija korisnika

Ukoliko su podaci koje je korisnik unio ispravni, na unesenu adresu elektroničke pošte se šalje poruka s aktivacijskom poveznicom (engl. *link*). Korisniku će se prikazati poruka sa slike 4.

Slika 4. Obavijest o slanju aktivacijske poruke

Nakon što korisnik klikne na poveznicu, njegov račun je aktiviran i može se prijaviti. Postupak slanja poruke elektroničke pošte na adresu korisnika, prikazan je u ispisu 1.

```
current_site = get_current_site(request)
email_subject = 'Activate Your Account',
message = render_to_string('auth/activate.html',
            {
                'user': user,
                'domain': current_site.domain,
                'uid': urlsafe_base64_encode
(force_bytes(user.id)),
                'token':
    tokenGenerator.make_token(user)
            })
email_message = EmailMessage(
                email_subject,
                message,
                settings.EMAIL_HOST_USER,
                to=[user.email]
            )
email_message.send()
```

Ispis 1. Slanje aktivacijske poruke

4.2. Prijava

Korisnik se prijavljuje sa korisničkim imenom i lozinkom. Ukoliko se neaktivan korisnik pokuša prijaviti, obavještava ga se da provjeri elektroničku poštu i aktivira svoj račun. U slučaju unosa krivog korisničkog imena ili lozinke, korisnika se obavještava o istome.

Forma koja se koristi za ovjeru (engl. *authentication*) korisnika nasljeđuje klasu `AuthenticationForm` iz paketa `django.contrib.auth.forms`. Klasa `AuthenticationForm` nema mogućnost ispisa poruke o neaktivnom korisniku pa je bilo potrebno napraviti novu

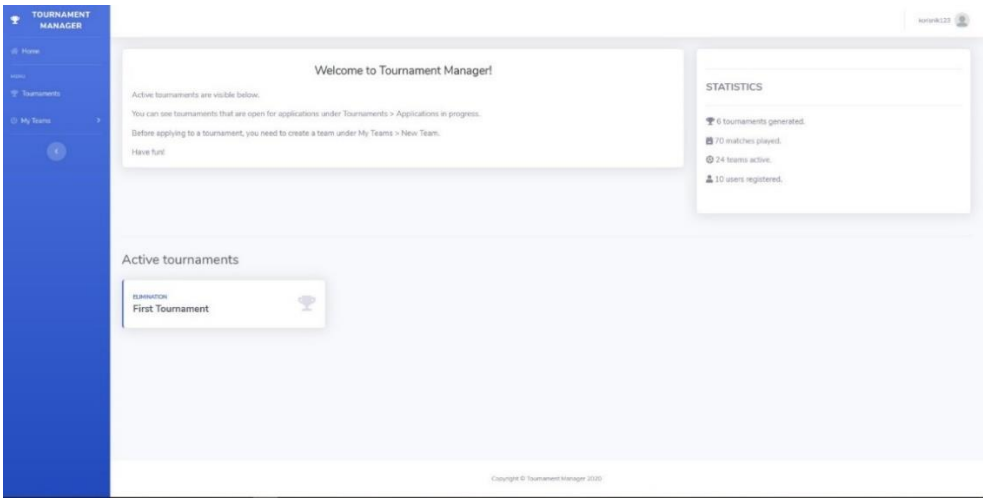
klasu `AuthForm` i metodu `get_invalid_login_error` u kojoj se provjerava je li korisnik aktivirao svoj račun. Postupak je prikazan u ispisu 2.

```
      class AuthForm(AuthenticationForm):
      def get_invalid_login_error(self):
              try:
                  user = User.objects.get(username=self.cleaned_data.get('use
rname'))
              except User.DoesNotExist:
                  return forms.ValidationError(
                      self.error_messages['invalid_login'],
                      code='invalid_login',
                      params={'username': self.username_field.verbose_name},
)
              self.error_messages['inactive'] = 'This account is inactive.
       Check your email for activation link.'
              if not user.is_active and user:
                  raise forms.ValidationError(
                      self.error_messages['inactive'],
                      code='inactive',)
              else:
                  return forms.ValidationError(
                      self.error_messages['invalid_login'],
                      code='invalid_login',
                      params={'username': self.username_field.verbose_name},
)
```

Ispis 2. Klasa `AuthForm`

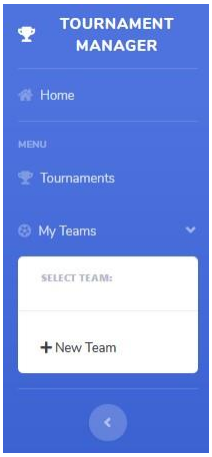
4.3. Navigacija i početna stranica

Nakon uspješne prijave, korisnika se preusmjerava na početnu stranicu koja je prikazana na slici 5. Na početnoj stranici prikazana je poruka dobrodošlice, statistika aplikacije i poveznice na turnire koji su trenutno aktivni.



Slika 5. Početna stranica

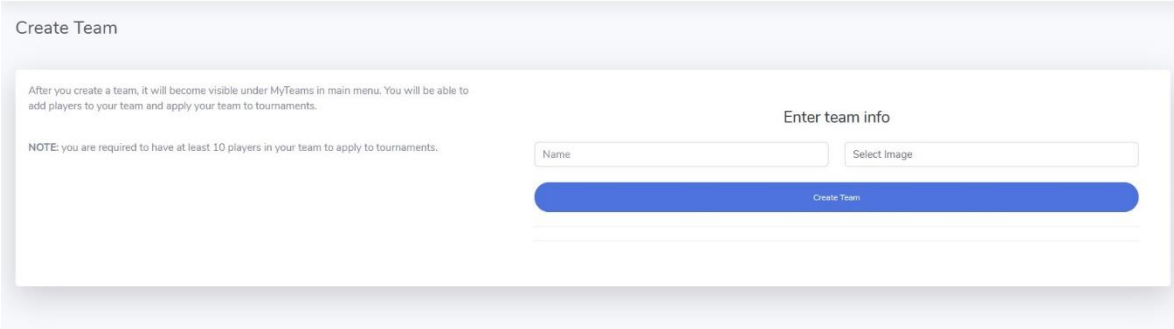
Navigacija se sastoji od poveznice koja vodi na početnu stranicu, poveznice koja vodi na sve turnire i dugmeta koji otvara padajući izbornik u kojemu su prikazane sve ekipe korisnika i poveznica za kreiranje nove ekipe. Navigacija je prikazana na slici 6.



Slika 6. Navigacija

4.4. Kreiranje ekipe

Korisnik može kreirati ekipu klikom na dugme „New Team“. Nakon klika na dugme otvara se forma u koju je potrebno unijeti ime ekipe i sliku. Forma je prikazana na slici 7.



Slika 7. Forma za kreiranje ekipe

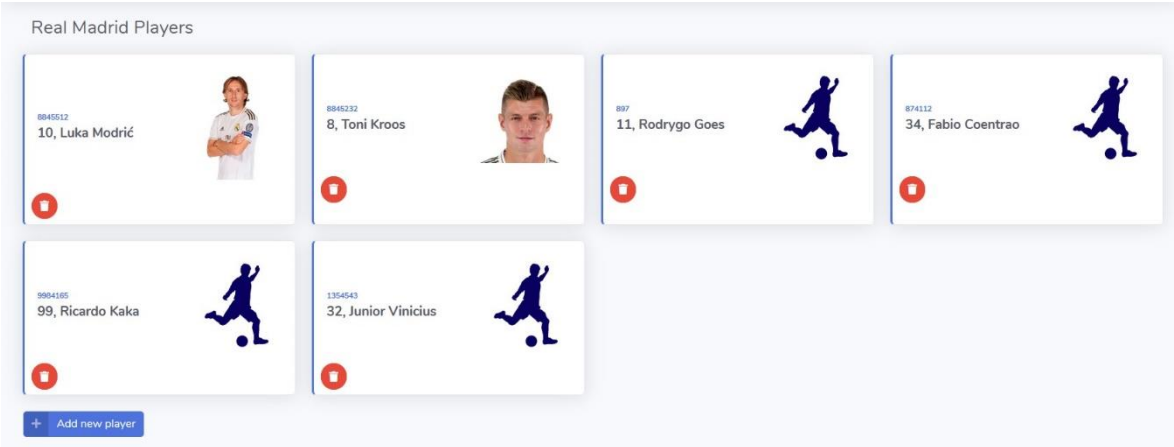
Ekipu je moguće kreirati i bez unosa slike, u tom slučaju se postavlja unaprijed zadana slika. U ispisu 3 prikazana je Django forma koja se koristi za kreiranje ekipe.

```
class TeamForm(ModelForm):
image = ImageField(label=(''),required=False, widget=FileInput)
class Meta:
model = Team
      fields = ['name', 'image']
```

Ispis 3. Django forma za kreiranje ekipe

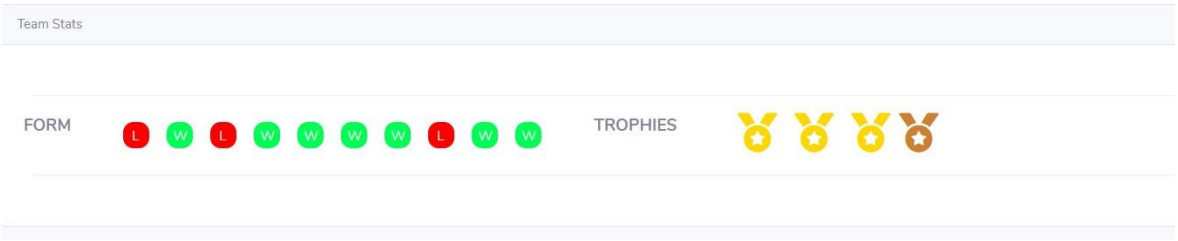
4.5. Kartica ekipe

Na kartici *pojedine* ekipe prvo su prikazani svi igrači registrirani u toj ekipi. Pojedinog igrača je moguće obrisati klikom na ikonu kante. Igrači u ekipi su prikazani na slici 8.



Slika 8. Igrači na kartici ekipe

Zatim slijedi prikaz statistike ekipe. Statistika se sastoji od forme i trofeja. Forma je prikazana za posljednjih 10 utakmica ekipe na slici 9.

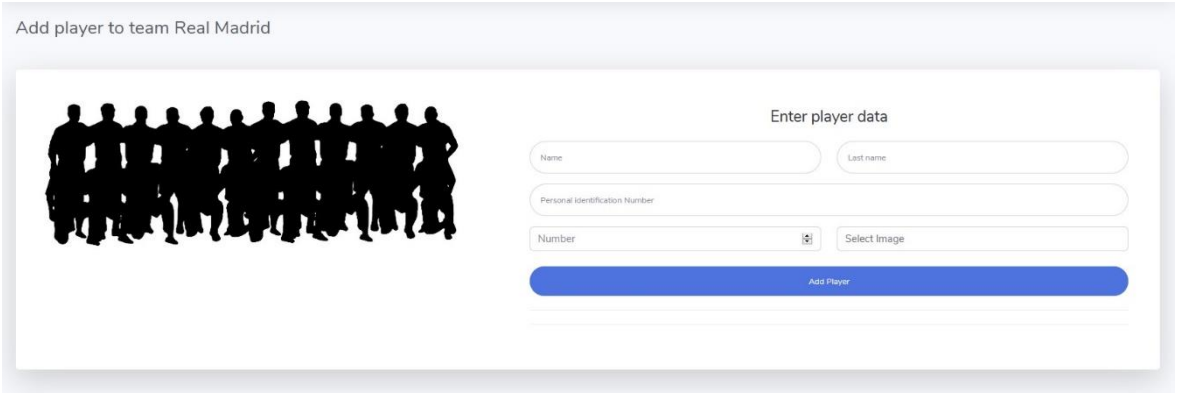


Slika 9. Statistika ekipe

Nakon prikaza statistike slijedi prikaz posljednjih 10 odigranih utakmica. Na kraju su prikazana dugmeta kojima je moguće izbrisati ekipu i mijenjati podatke o ekipi.

4.6. Dodavanje igrača

Igrača je moguće dodati u ekipu na kartici *ekipe* (prethodno poglavlje) klikom na dugme „Add New Player“. Nakon klika na dugme otvara se forma za unos podataka o igraču, vidljiva na slici 10.



Slika 10. Forma za dodavanje igrača

Potrebno je unijeti sljedeće podatke: ime, prezime, osobni identifikacijski broj, broj na dresu i sliku. Osobni identifikacijski broj mora biti jedinstven na razini aplikacije. To je

riješeno u definiranju klase `Player`, postavljanjem svojstva „`unique=True`“, vidljivo u ispisu 4.

```
      class Player(models.Model):
          name = models.CharField(max_length=30, default='')
          lastname = models.CharField(max_length=30, default='')
          team = models.ForeignKey(Team, on_delete=models.CASCADE, null=T
rue, blank=True)
          image = models.ImageField(upload_to='players/', default='player
s/player.png', blank=True, null=True)
          pin = models.CharField(unique=True, max_length=15, default='')
          number = models.IntegerField(range(1,99))
```

Ispis 4. Klasa Player

Broj na dresu igrača je ograničen na raspon brojeva od 1 do 99. Na razini ekipe svaki broj mora biti jedinstven. Provjera jedinstvenosti broja je prikazana u ispisu 5.

```
       for _player in players:
                       if _player.number == player.number:
                           return render(request, 'error/playerRegisterError.
html')
```

Ispis 5. Provjera jedinstvenosti broja igrača u ekipi

Varijabla naziva `_player` će kroz petlju poprimiti vrijednosti svih ostalih igrača u timu, a varijabla naziva `player` označava novog igrača.

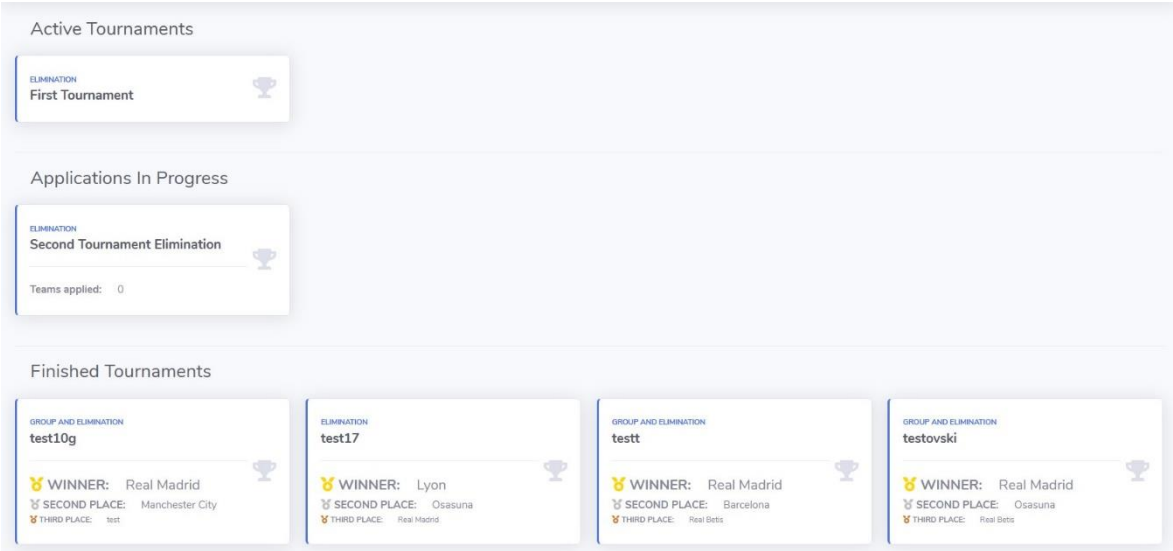
4.7. Kartica igrača

Na kartici pojedinog igrača prikazani su sljedeći podaci: ime, prezime, osobni identifikacijski broj, broj na dresu i naziv tima za kojeg igrač nastupa. Klikom na poveznicu „update info“ otvara se prozor u kojemu je moguće mijenjati podatke o igraču. S desne strane prikazana je statistika igrača u vidu broja odigranih utakmica i postignutih pogodaka. Kartica igrača vidljiva je na slici 11.

Slika 11. Kartica igrača

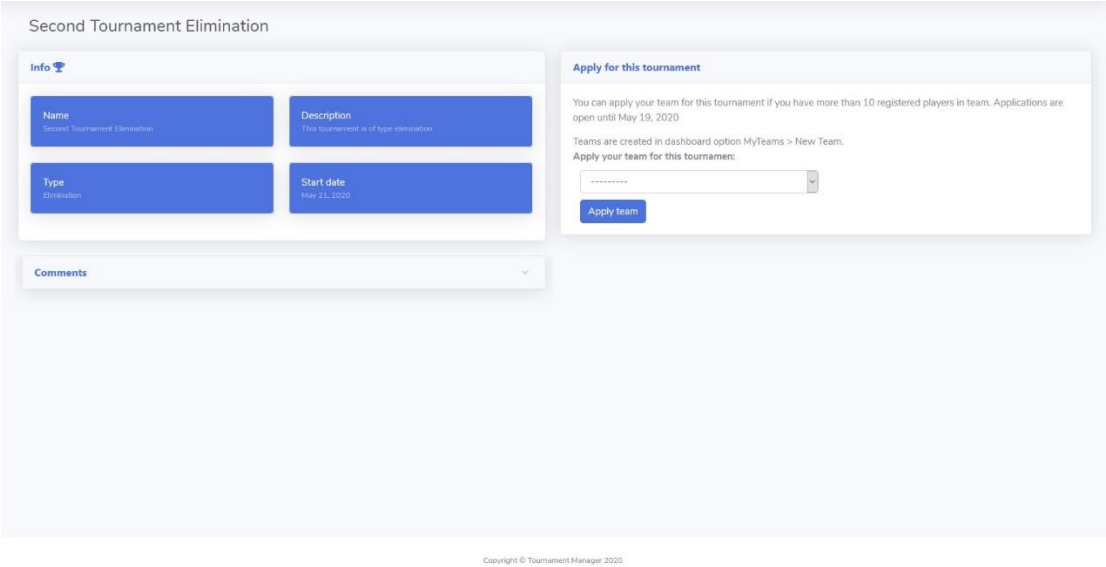
4.8. Prijava ekipe na turnir

Turniri na koje korisnici mogu prijavljivati ekipe su vidljivi nakon klika na navigacijsku stavku „Tournaments“, pod rubrikom „Applications in progress“. Prikaz turnira vidljiv je na slici 12.



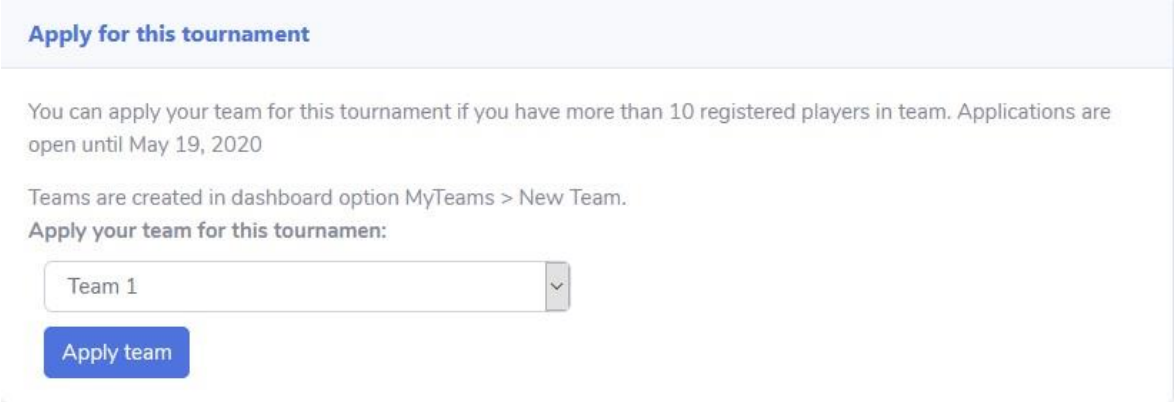
Slika 12. Prikaz turnira

Klikom na turnir otvorit će se kartica *turnir*. Za turnir kojemu su prijave u tijeku moguće je vidjeti informacije o turniru, dodavati komentare i prijaviti ekipu za turnir. Prikaz spomenute kartice za turnir kojem su prijave u tijeku prikazan je na slici 13.



Slika 13. Kartica *turnir* kojemu su prijave u tijeku

Za prijavu ekipe potrebno je imati barem 10 registriranih igrača u ekipi. Iz padajućeg izbornika potrebno je izabrati ekipu i klikom na dugme „Apply“ vrši se prijava. Padajući izbornik vidljiv je na slici 14.



Slika 14. Prijava ekipe na turnir

Prijava ekipe na turnir realizirana je klasom naziva `TeamApplication`, vidljivo u ispisu 6.

```
      class TeamApplication(models.Model):
          team = models.ForeignKey(Team, on_delete=models.CASCADE)
          tournament = models.ForeignKey(Tournament, on_delete=mode
ls.CASCADE, null=True, blank=True)
```

Ispis 6. Klasa `TeamApplication`

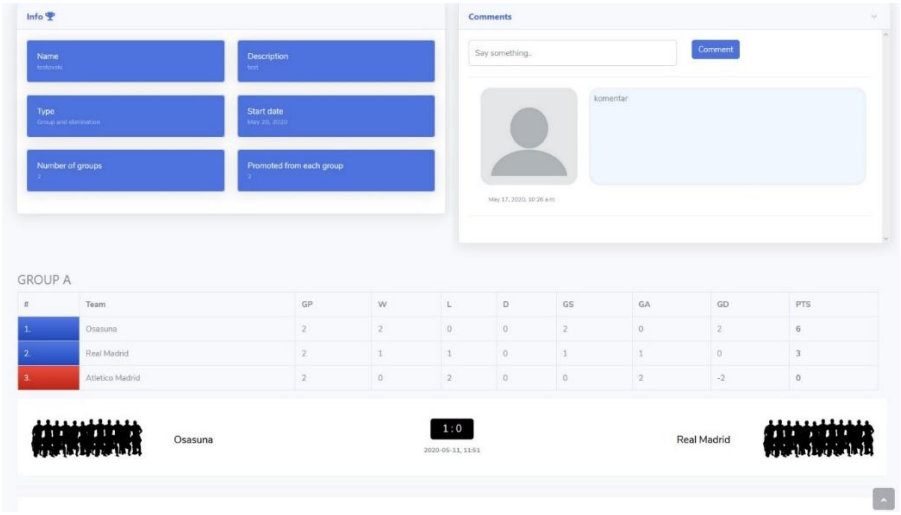
Korisnik ne može prijavljivati više ekipa na isti turnir. Jednom kad je korisnik prijavio ekipu, padajući izbornik mu više nije vidljiv.

4.9. Kartica aktivnog turnira

Turnir može imati 3 statusa: aktivan, završen i prijave u tijeku. Ovisno o statusu, korisniku je prikazan različit sadržaj. Turnir može imati 2 tipa: grupa + eliminacija i samo eliminacija. U ovom poglavlju će se prikazati izgled kartice turnira za oba tipa turnira. Detaljan opis načina generiranja pojedinog tipa turnira opisan je u poglavlju 4.

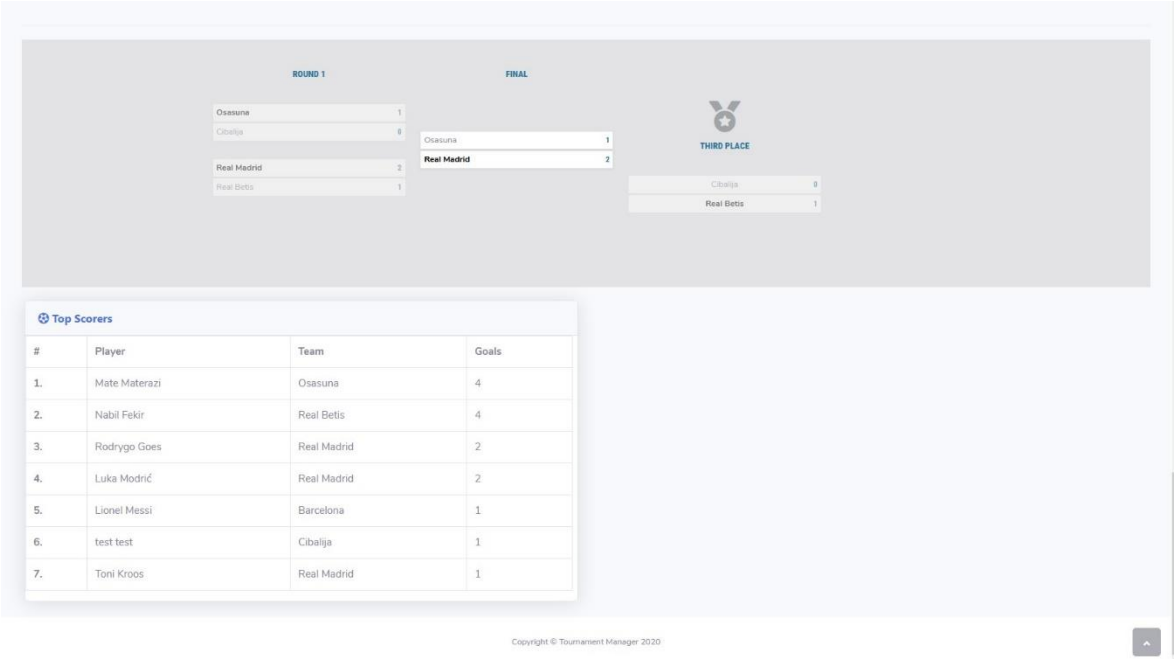
4.9.1. Tip turnira grupa + eliminacija

Na kartici ovog tipa turnira prvo su prikazane osnovne informacije o turniru. Zatim slijedi prozor u kojemu su komentari i u kojemu je moguće komentirati turnir. Nakon toga ide element u kojemu su prikazane grupe i utakmice. Prvo je prikazana tablica grupe pa sve utakmice u grupi, vidljivo na slici 15.



Slika 15. Grupa + eliminacija

Ukoliko je grupna faza završena, eliminacijska faza može započeti. U tom slučaju ispod prikaza grupa i utakmica u grupama je prikazano stablo eliminacijske faze. Ispod stabla eliminacijske faze prikazani su najbolji strijelci na turniru, vidljivo na slici 16.



Slika 16. Stablo eliminacije i prikaz strijelaca

Klikom na pojedinu utakmicu u stablu turnira ili u grupi otvara se kartica *utakmice*. Na kartici *utakmice* vidljiv je rezultat i strijelci golova, prikazano na slici 17.

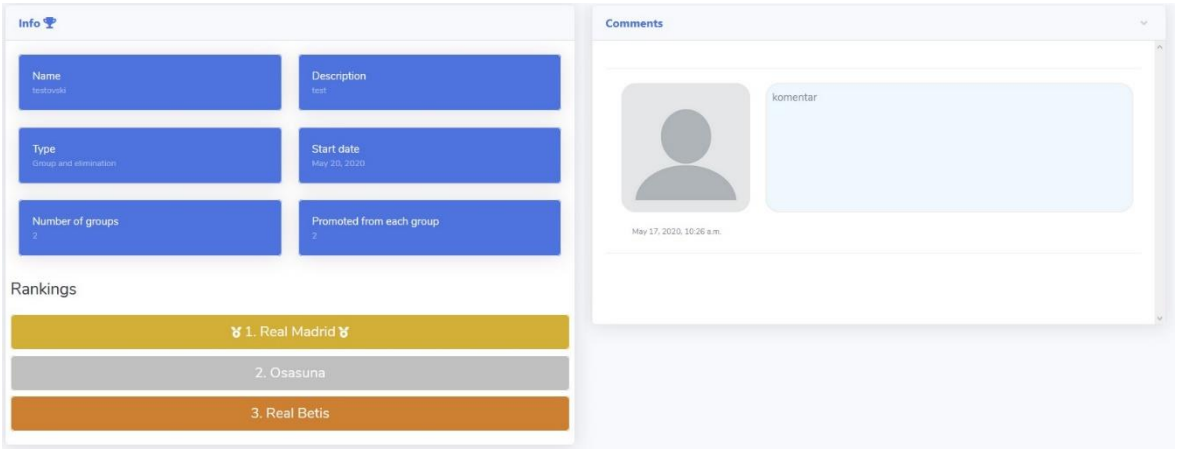
Slika 17. Kartica utakmice

4.9.2. Tip turnira eliminacija

Za ovaj tip turnira generira se samo stablo eliminacije, ovisno o broju prijavljenih ekipa. Stablo eliminacije prikazano je u prethodnom poglavlju.

4.10. Kartica završenog turnira

Nakon što turnir završi korisnici više ne mogu komentirati turnir. Prikazan je i element s ekipama koje su osvojile prva 3 mjesta na turniru. Ovo su jedine 2 razlike u prikazu završenog i aktivnog turnira. Element s ekipama koje su osvojile medalju prikazan je na slici 18.

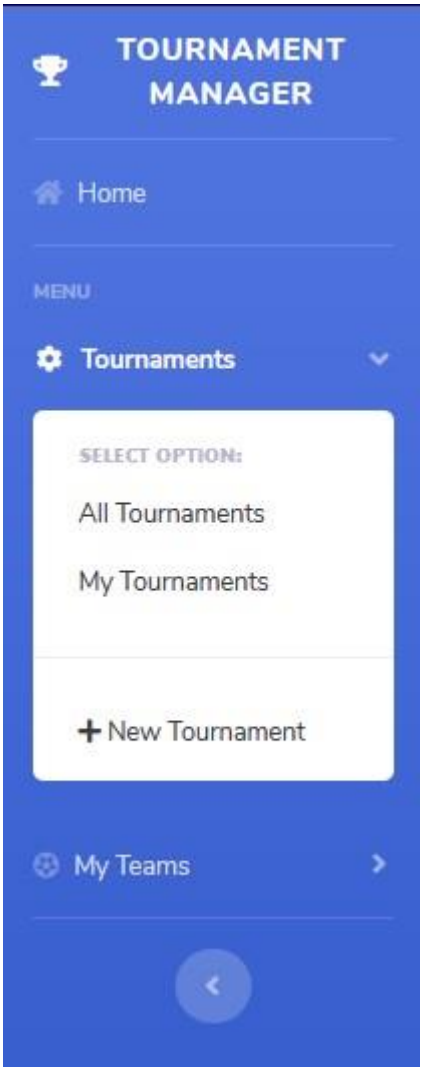


Slika 18. Komentari i informacije završenog turnira

5. Funkcionalnosti aplikacije za ulogu kreatora turnira

5.1. Navigacija

Navigacija za korisnika s ulogom kreatora turnira je nešto drugačija nego za korisnika s ulogom sudionika u turniru. Vidljivo na slici 19.

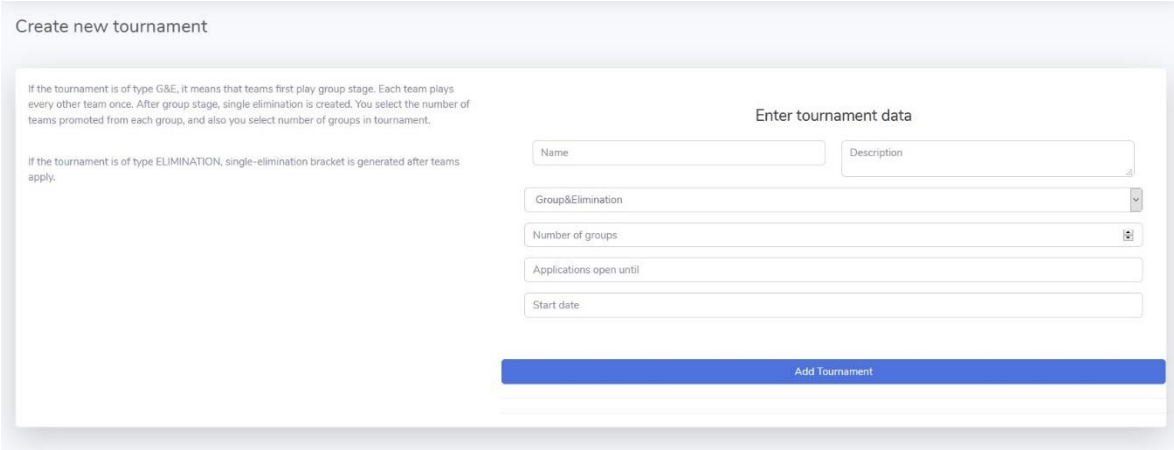


Slika 19. Navigacija korisnika s ulogom kreatora turnira

Klikom na poveznicu „All Tournaments“ korisnik će biti preusmjeren na stranicu sa svim turnirima u aplikaciji, a klikom na poveznicu „My Tournaments“ će biti preusmjeren na stranicu sa prikazom vlastitih turnira. Korisnik s ulogom kreatora turnira također može stvarati ekipe i prijavljivati ih na turnire, osim na turnire koje on sam organizira, pa mu je dostupna i poveznica „My Teams“. Način kreiranja ekipe je opisan u poglavlju 4.4.

5.2. Kreiranje turnira

Turnir se može kreirati klikom na poveznicu „New Tournament“. Nakon klika, korisniku se otvara stranica s formom za unos podataka o turniru, prikazano na slici 20.



Slika 20. Forma za unos podataka o turniru

Potrebno je unijeti ime turnira, opis, tip, datum isteka roka za prijave i datum početka.

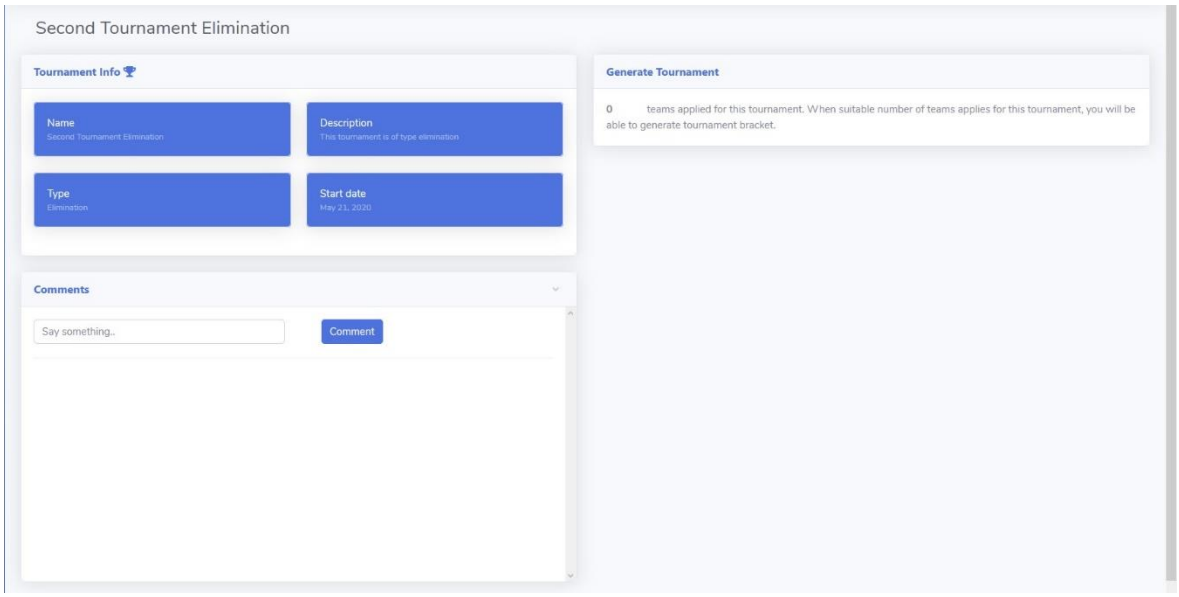
5.3. Kartica turnira otvorenog za prijave

Nakon klika na dugme „Add Tournament“ kreira se turnir i on je otvoren za prijave. Korisnici se mogu prijaviti na turnir do datuma do kojeg su prijave otvorene. U datoteci `timeDateValidators.py` nalaze se funkcije koje služe za validaciju datuma. Funkcijom `isInFuture` (Ispis 7), koja prima datum i provjerava je li datum u budućnosti u odnosu na sadašnji datum, se provjerava može li se korisnik prijaviti na turnir.

```
def isInFuture(date):
    today = datetime.now().date()
    return True if date > today else False
```

Ispis 7. Funkcija `isInFuture`

S desne strane kartice turnira nalazi se element s naslovom „Generate Tournament“ u kojem se vidi broj prijavljenih ekipa na turnir. Turnir je moguće generirati na datum koji je naznačen kao datum početka turnira, ne prije. Kartica turnira prikazana je na slici 21.

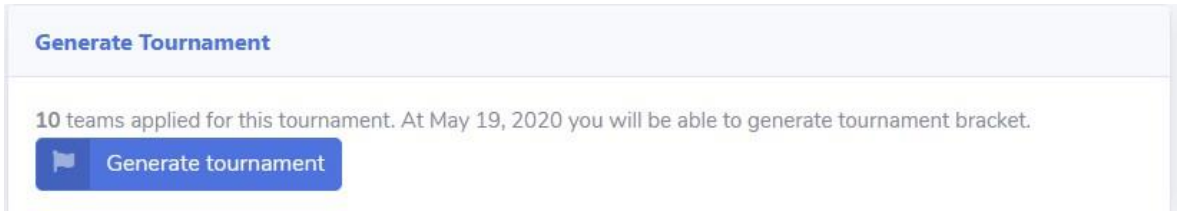


Slika 21. Kartica turnira otvorenog za prijave

Detaljan opis načina generiranja pojedinog tipa turnira slijedi u sljedećem poglavlju.

5.4. Generiranje turnira

Na datum koji je naznačen kao datum početka turnira prikazat će se dugme „Generate Tournament“ u istoimenom elementu na kartici turnira otvorenog za prijave, prikazano na slici 22.



Slika 22. Dugme za generiranje turnira

Klikom na dugme pokreće se algoritam generiranja turnira.

5.4.1. Eliminacijski tip turnira

Svaki turnir eliminacijskog tipa ima jednu eliminacijsku fazu. Klasa `EliminationPhase` prikazana je u ispisu 8.

```
      class EliminationPhase(models.Model):
          num_of_rounds = models.IntegerField()
          tournament = models.ForeignKey(Tournament, on_delet
e=models.CASCADE, null=True, blank=True)
```

Ispis 8. Klasa `EliminationPhase`

Svaka eliminacijska faza se sastoji od više rundi kao npr. finale, polufinale, četvrtfinale i slično. Klasa `EliminationRound` je prikazana u ispisu 9.

```
      class EliminationRound(models.Model):
          num_of_games = models.IntegerField(null=True, blank=True, d
efault=0)
          name = models.CharField(max_length=30)
          elimination_phase = models.ForeignKey(EliminationPhase, on_
delete=models.CASCADE, null=True, blank=True)
          isCurrent = models.BooleanField(default = False)
```

Ispis 9. Klasa **EliminationRound**

U jednom trenutku u turniru može se igrati samo jedna runda i taj podatak čuva podatkovni član `isCurrent`. Svaka runda se sastoji od više utakmica.

Algoritam generiranja turnira se razlikuje ovisno o broju prijavljenih ekipa na turnir.

Za broj prijavljenih ekipa koji je potencija broja 2 algoritam je jednostavniji. Prvo je potrebno uzeti drugi logaritam od broja prijavljenih ekipa, kako bi se izračunao broj rundi u eliminacijskoj fazi. Zatim se kreira eliminacijska faza. Postupak je prikazan u ispisu 10.

```
      roundsNum = math.log(teams.__len__(), 2)
          eliminationPhase = EliminationPhase(num_of_rounds = rou
ndsNum, tournament = tournament)
          eliminationPhase.save()
```

Ispis 10. Stvaranje objekta eliminacijske faze

Nakon toga je potrebno generirati runde i izračunati broj utakmica u svakoj rundi. U prvoj rundi će broj utakmica biti broj prijavljenih ekipa podijeljen s 2. U svakoj idućoj rundi broj utakmica će biti upola manji. Postupak generiranja rundi prikazan je u ispisu 11.

```
      matchesInRound = teamsCount / 2
      while i < roundsNum:
              round = EliminationRound(num_of_games = matchesInR
ound, elimination_phase = eliminationPhase)
              round.save()
              i+=1
              matchesInRound /= 2
```

Ispis 11. Generiranje rundi

Kad je unaprijed poznat broj utakmica u rundi, lako je za svaku rundu generirati utakmice. Problem se javlja kod određivanja sljedećih utakmica, odnosno određivanja sljedećeg suparnika za pobjednika svake utakmice. Svaka utakmica ima vezu sama sa sobom, odnosno nosi informaciju o idućoj utakmici. Iz tog razloga je najlakši način za odrediti iduće utakmice prolazak kroz listu rundi u obrnutom redoslijedu, od finala prema prvoj rundi. Prva runda koja dolazi na red je finale i ono neće imati iduću utakmicu jer je to zadnja utakmica turnira, ali će se to finale ubaciti u listu idućih utakmica (varijabla `nextRoundMatches`). Nakon toga će se za polufinale, četvrtfinale i sve ostale runde generirati utakmice i za svaki par utakmica postavljati jedna sljedeća utakmica iz liste `nextRoundMatches` . Svaka utakmica se ubacuje u listu idućih utakmica za iduću rundu koja je na redu za generiranje. Algoritam je prikazan u ispisu 12.

```
      for round in rounds[::-1]:
              i=0
              if round.num_of_games == 1: #final
                  match = Match(elimination_round = round)
                  match.save()
                  nextRoundMatches.append(match)
              else:
                  nextMatchIndex = 0
                  tmp = nextRoundMatches
                  nextRoundMatches = []
                  while i < round.num_of_games:
                      match = Match(elimination_round = round,
nextMatch = tmp[nextMatchIndex])
                      match.save()
                      nextRoundMatches.append(match)
                      if i % 2 == 1 and i != 0:
                          nextMatchIndex+=1
                      i+=1
```

Ispis 12. Generiranje utakmica u rundama

Za broj prijavljenih ekipa koji nije potencija broja 2 prvo je potrebno izračunati najmanji broj koji je potencija broja 2 i ujedno veći od broja prijavljenih ekipa. Npr. za 12 prijavljenih ekipa to je broj 16. Tada je potrebno broj prijavljenih ekipa oduzeti od tog broja da bi se izračunao broj ekipa koje direktno idu u drugi krug. Za 12 prijavljenih ekipa to su 4 ekipe. 4 ekipe idu direktno u drugi krug, a 8 ekipa igra prvu rundu. Iz prve runde u drugu će proći 4 ekipe i sa 4 ekipe koje su direktno išle u drugu rundu se dobije 8 ekipa. Broj 8 je potencija broja 2. Postupak generiranja eliminacijskog tipa turnira s brojem prijavljenih ekipa koji je potencija broja 2 je prethodno objašnjen. U ispisu 13 je prikazan izračun broja ekipa koje idu direktno u drugu rundu.

```
      i = teamsCount + 1
              while True:
                  if math.log(i, 2).is_integer():
                      toNextRoundCount = i - teamsCount
                      break
                  i+=1
```

Ispis 13. Izračun broja ekipa koje idu direktno u drugu rundu

Pri postavljanju idućih utakmica za utakmice prvog kruga, potrebno je provjeriti ima li iduća utakmica već postavljene ekipe (one ekipe koje idu direktno u drugi krug).

5.4.2. Tip turnira grupa + eliminacija

Korisnik pri kreiranju turnira određuje broj grupa, datum do kojeg traju prijave i datum početka turnira. Za kreiranje ovog tipa turnira minimalan broj prijavljenih ekipa je 4. Turnir je moguće generirati nakon isteka roka za prijave, ne prije. Dokle god je broj prijavljenih ekipa veći od 4 moguće je generirati turnir, neovisno o tome je li broj prijavljenih ekipa djeljiv s brojem grupa. Sve grupe u turniru ne moraju imati isti broj ekipa.

Svaki turnir ovog tipa ima pripadajući grupnu fazu. Klasa `GroupPhase` je prikazana u ispisu 14.

```
class GroupPhase(models.Model):
    num_of_groups = models.IntegerField()
```

Ispis 14. Klasa `GroupPhase`

Svaka grupna faza se sastoji od više grupa. Klasa Group je prikazana u ispisu 15.

```
class Group(models.Model):
    name = models.CharField(max_length=20)
    group_phase = models.ForeignKey(GroupPhase, on_de
lete=models.CASCADE)
    num_of_teams = models.IntegerField(default=0)
    matches_generated = models.BooleanField(default =
 False)
```

Ispis 15. Klasa Group

U jednoj grupi se natječe više timova. Poredak timova u grupi određen je brojem bodova. Pobjeda u grupi nosi 3 boda, neriješen rezultat 1 bod, a poraz 0 bodova. Ukoliko timovi imaju isti broj bodova, prednost ima tim koji ima bolju gol razliku. Gol razlika je razlika broja zabijenih golova i broja primljenih golova. Ukoliko je i gol razlika ista, tada prednost ima tim koji je zabio više golova. Klasa `Participation` koja prikazuje odnos ekipe i grupe je prikazana u ispisu 16.

```
      class Participation(models.Model):
          group = models.ForeignKey(Group, on_delete=model
s.CASCADE)
          team = models.ForeignKey(Team, on_delete=models.
CASCADE)
          num_of_played_games = models.IntegerField(defaul
t=0)
          num_of_victories = models.IntegerField(default=0
)
          num_of_defeats = models.IntegerField(default=0)
          num_of_draws = models.IntegerField(default=0)
          points = models.IntegerField(default=0)
          goals_against = models.IntegerField(default=0)
          goals_scored = models.IntegerField(default=0)
          goal_diff = models.IntegerField(null=True, blank
=True, default=0)
```

Ispis 16. Klasa `Participation`

Prvi korak generiranja turnira tipa grupa + eliminacija je stvaranje grupne faze. Nakon stvaranja grupne faze stvaraju se grupe i razvrstavaju ekipe u grupe. Postupak stvaranja grupa i razvrstavanja ekipa prikazan je u ispisu 17.

```
      random.shuffle(teams)
              groups = []
              i = 0
              while i < tournament.num_of_groups:
                  group = Group(name = letters[i] ,num_of_
teams = tournament.num_of_apps / tournament.num_of_groups,
 group_phase = groupPhase)
                  group.save()
                  groups.append(group)
                  j = 0
                  while j < group.num_of_teams:
                      team = teams.pop()
                      part = Participation(group = group,
team = team, num_of_played_games = 0, num_of_victories = 0
, num_of_defeats = 0, points = 0)
                      part.save()
                      j+=1
                  i+=1
```

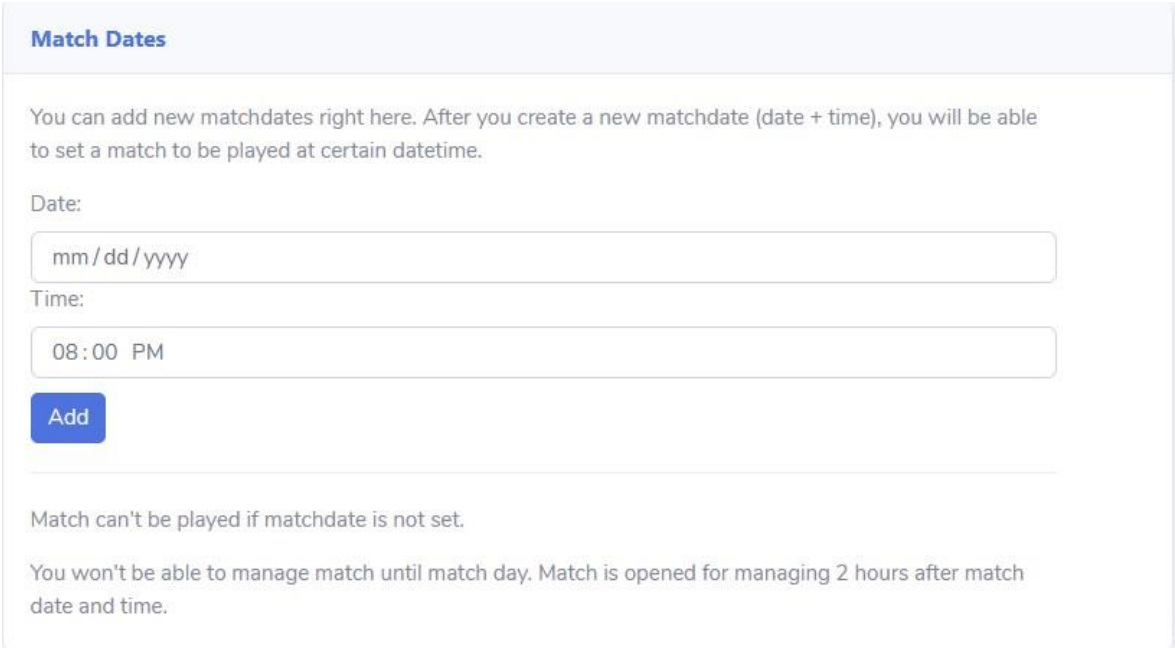
Ispis 17. Stvaranje grupne faze i grupa

Ukoliko broj prijavljenih ekipa nije djeljiv s brojem grupa, potrebno je naći najveći broj koji je djeljiv s brojem grupa, a manji je od broja prijavljenih ekipa. Tada će se taj broj ekipa razvrstati u grupe, a ostale ekipe će se na kraju jedna po jedna dodavati u svaku grupu. Npr. za 15 ekipa i 4 grupe prvo će se napraviti 4 grupe po 3 ekipe, a onda će se u prve 3 grupe dodati preostale ekipe, u svaku grupu po jedna ekipa.

5.5. Kartica turnir

5.5.1. Termini odigravanja utakmica

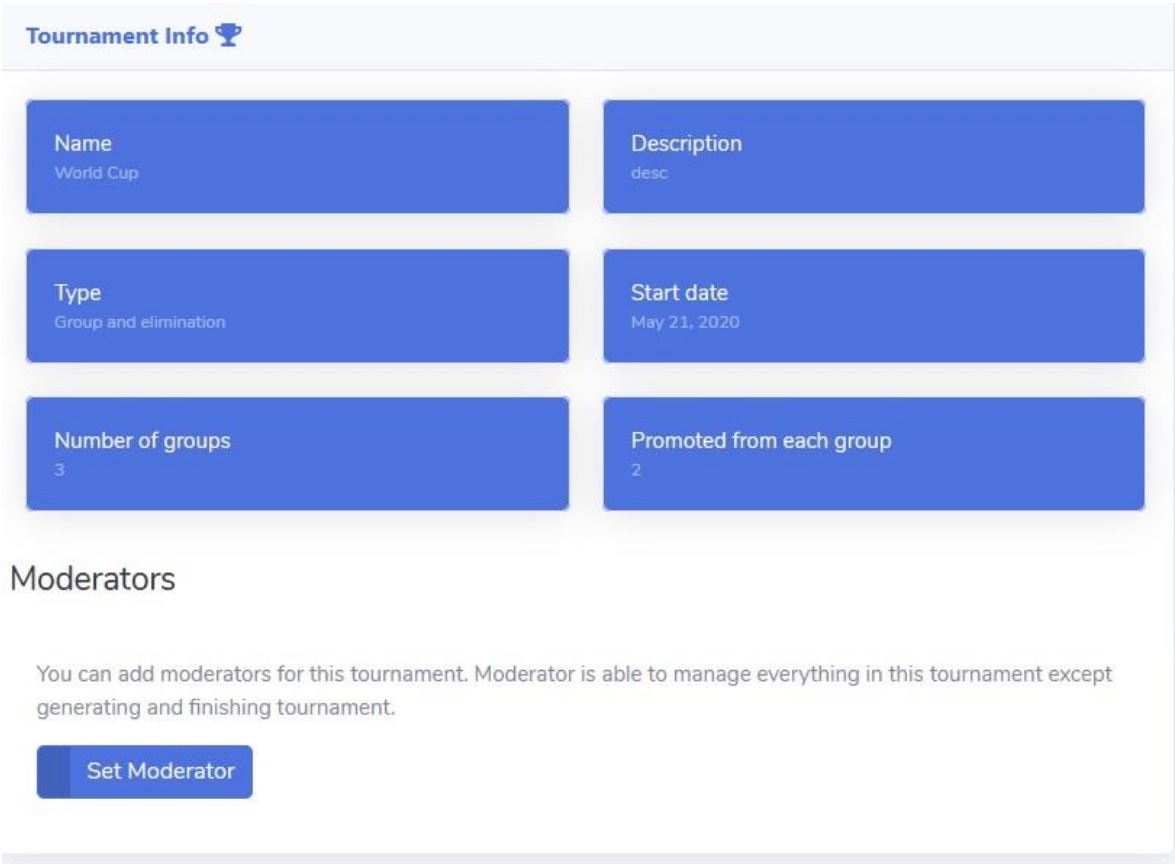
Na desnoj strani kartice *turnir* nalazi se element naziva Match Dates, vidljivo na slici 23. U navedenom elementu se unose termini. U jednom terminu se može igrati više utakmica.



Slika 23. Unos termina

5.5.2. Moderatori

Kreator turnira za svoj turnir može postaviti moderatora, odnosno osobu koja uz samog kreatora turnira može unositi rezultate utakmica. Moderator također mora imati ulogu kreatora turnira u aplikaciji. Moderator ne može generirati turnir niti ga završiti, ali može unositi rezultate utakmica i termine odigravanja utakmica. Moderatori se postavljaju u elementu u kojemu se nalaze informacije o turniru. Klikom na dugme „Set Moderator“ otvara se skočni prozor u kojemu je iz padajućeg izbornika moguće odabrati moderatora, vidljivo na slici 24.



Slika 24. Postavljanje moderatora

Moderator je u aplikaciji realiziran kroz klasu `Moderator` prikazanu u ispisu 18.

```
      class Moderator(models.Model):
          user = models.ForeignKey(User, on_delete=models.CA
SCADE)
          tournament = models.ForeignKey(Tournament, on_dele
te=models.CASCADE, null=True, blank=True)
```

Ispis 18. Klasa Moderator

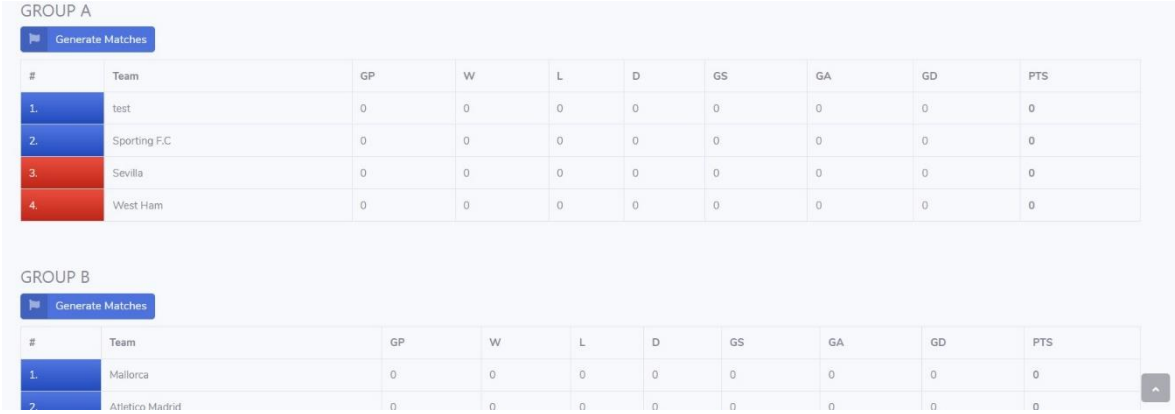
Prava moderatora nad turnirom utvrđuju se provjeravanjem postoji li odgovarajući objekt Moderator, prikazano u ispisu 19.

```
Moderator.objects.filter(user = request.user, tournament = tournament)
```

Ispis 19. Provjera postoji li objekt moderator

5.5.3. Grupna faza

Nakon generiranja turnira, prikazat će se tablica za svaku grupu u turniru. Iznad tablice će biti dostupno dugme za generiranje utakmica u grupi, vidljivo na slici 25.



Slika 25. Tablica grupa s dugmetom za generiranje utakmica

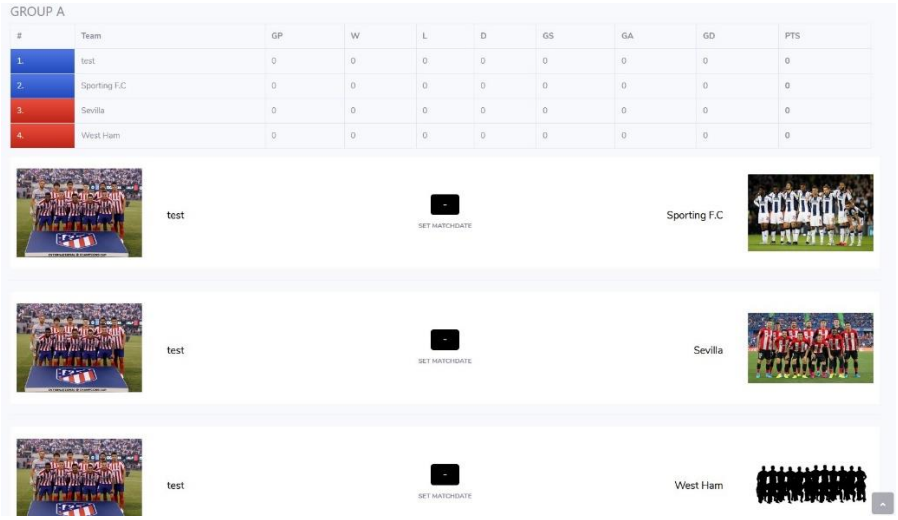
Algoritam generiranja utakmica u grupi prikazan je u ispisu 20.

```
                i = 0
                j = 0
                while i < teams.__len__():
                    j = i + 1
                    while j < teams.__len__():
                        match = Match(team1 = teams[i], team2
= teams[j], group = group)
                        match.save()
                        j+=1
                    i+=1
                group.matches_generated = True
                group.save()
```

Ispis 20. Generiranje utakmica u grupi

U varijabli `teams` su pohranjene sve ekipe iz grupe.

Nakon generiranja utakmica ispod tablice grupe prikazat će se utakmice u grupi, vidljivo na slici 26.



Slika 26. Tablica grupe s utakmicama

Kako bi se u utakmici mogao mijenjati rezultat, potrebno je postaviti datum i vrijeme odigravanja utakmice. Klikom na „SET MATCHDATE“ otvorit će se skočni prozor u kojemu će biti dostupni prethodno uneseni datumi. Rezultat utakmice nije moguće unositi prije naznačenog termina odigravanja utakmice.

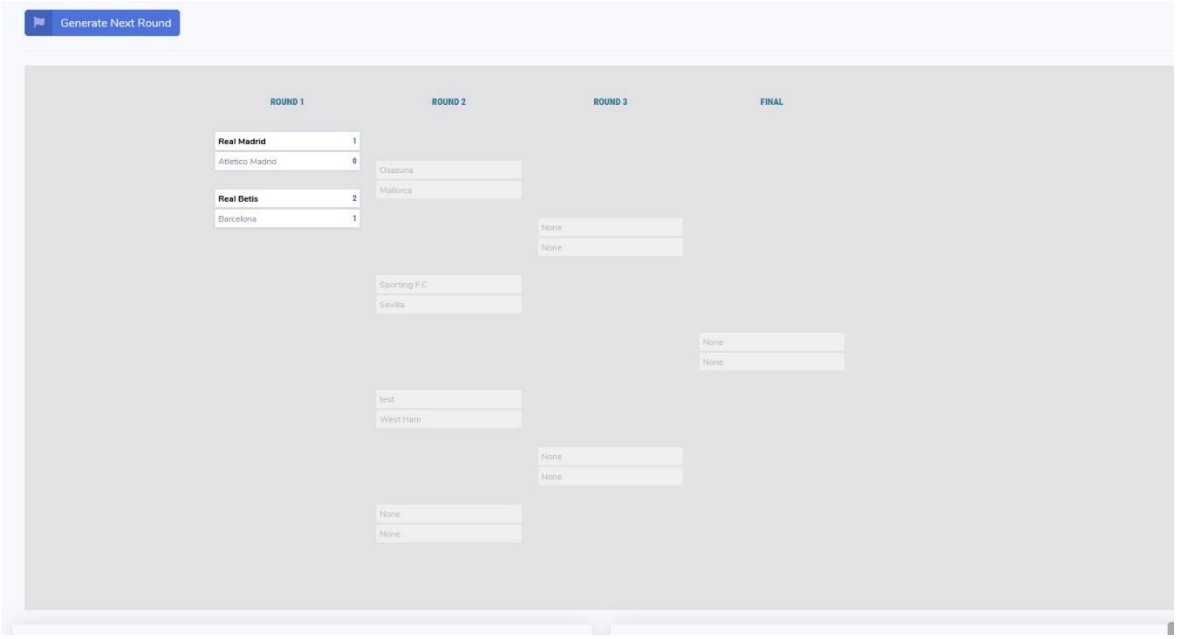
Nakon odigravanja svih utakmica u svim grupama, moguće je generirati eliminacijsku fazu s ekipama pobjednicima grupa. Dugme za generiranje eliminacijske faze nalazi se na dnu kartice *turnir* i postaje dostupno nakon odigravanja svih utakmica u grupi. Postupak generiranja eliminacijske faze prikazan je u poglavlju 5.4.1.

5.5.4. Eliminacijska faza

Generiranjem eliminacijske faze napravljeno je samo stablo turnira. Za svaku rundu je potrebno postaviti ekipe u utakmice. Nakon generiranja turnira dostupno je dugme „Generate Next Round“ za ispunjavanje utakmica ekipama. Za generiranje svake iduće runde potrebno je završiti sve utakmice trenutne runde.

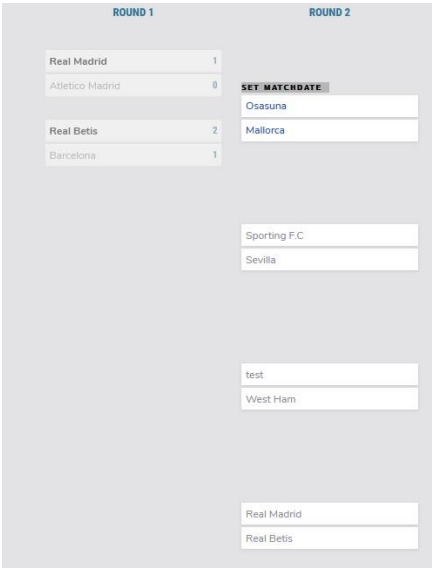
U trenutku kada je miš iznad utakmice pojavit će se natpis „SET MATCHDATE“ iznad utakmice. Klikom na natpis otvara se skočni prozor u kojemu se postavlja termin odigravanja utakmice.

Na slici 27 prikazano je stablo eliminacijske faze. U trenutku na slici završena je prva runda. Dostupno je dugme pomoću kojeg će se generirati čitava druga runda, obzirom da je dio ekipa već raspoređen zbog broja prijavljenih ekipa koji nije potencija broja 2.



Slika 27. Eliminacijska faza

Ekipe Real Madrid i Real Betis će nakon generiranja iduće runde igrati međusobno u toj rundi. Tek u trenutku kad je druga runda ujedno i trenutna runda eliminacijske faze, moguće je na utakmice druge runde postavljati termine odigravanja.



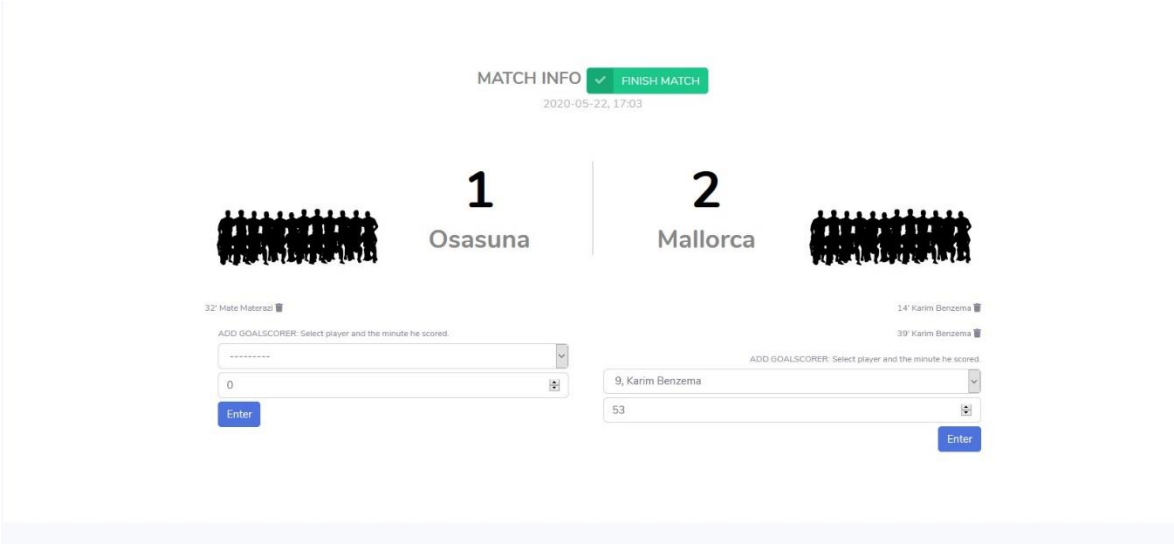
Slika 28. Eliminacijska faza nakon generiranja druge runde

Na slici 28 je prikazano stanje eliminacijske faze nakon generiranja druge runde.

Nakon odigravanja finala i utakmice za treće mjesto dugme za završavanje turnira postaje dostupno.

5.5.5. Unos rezultata

Klikom na utakmicu otvara se kartica *utakmice* (slika 29) . U terminu odigravanja utakmice moguće je unositi golove u utakmicu. S lijeve strane prikazana je prva ekipa utakmice, a s desne druga. U padajućim izbornicima dostupni su svi igrači pojedine ekipe. Gol se unosi na način da se odabere igrač iz padajućeg izbornika i unese minuta gola. Unos rezultata je dostupan 2 sata od trenutka naznačenog kao termin odigravanja utakmice. Utakmice grupne faze je moguće završiti neriješenim rezultatom, dok utakmice eliminacijske faze nije moguće završiti neriješenim rezultatom. Klikom na dugme „Finish Match“ završava se utakmica.

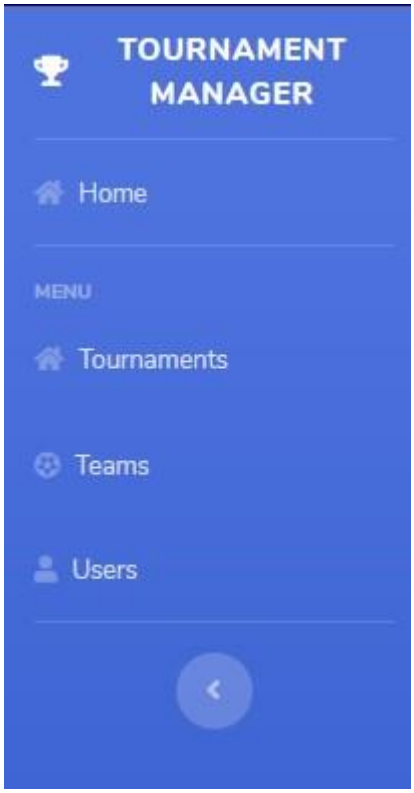


Slika 29. Kartica utakmice

6. Funkcionalnost aplikacije za ulogu upravitelja

6.1. Navigacija

Upravitelj može raditi promjene po svim turnirima, timovima i korisnicima. U skladu s tim je izrađena i navigacija (slika 30).



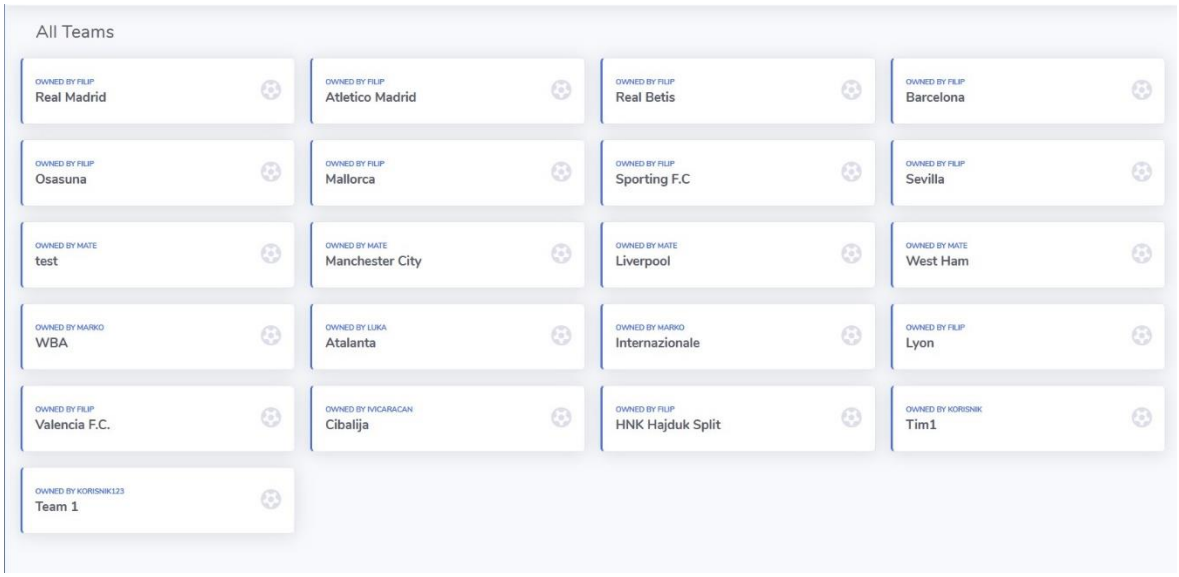
Slika 30. Navigacija korisnika s ulogom upravitelja

6.2. Turniri

Klikom na poveznicu „Tournaments“ otvara se kartica s prikazom svih turnira razvrstanih po statusu. Upravitelj ima ista prava nad svim turnirima kao i kreatori turnira. Kartica sa prikazom svih turnira je vidljiva na slici 12.

6.3. Ekipe

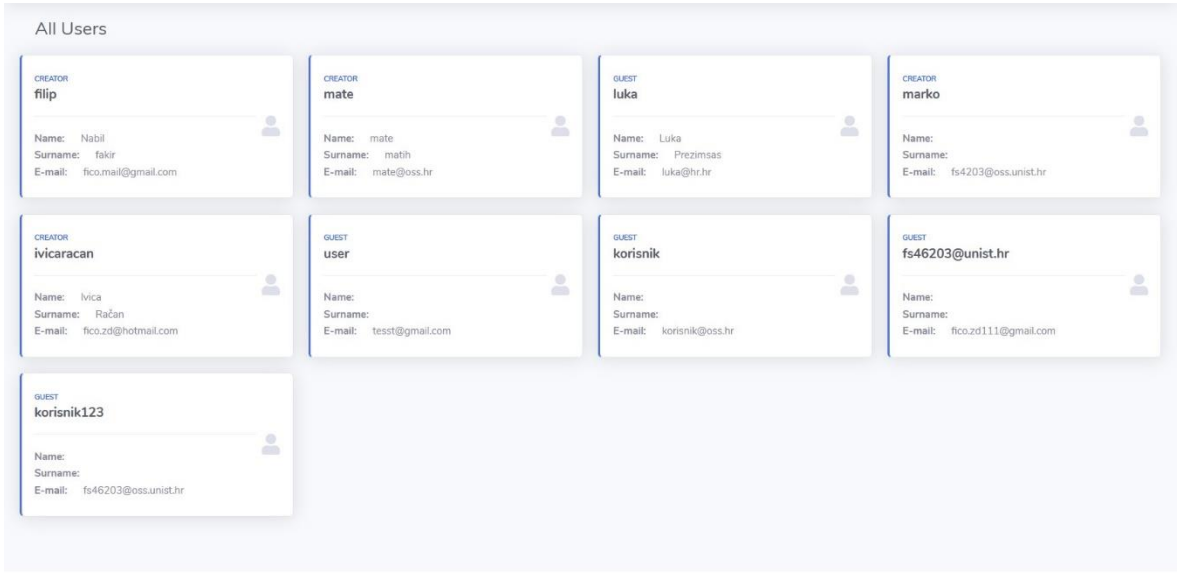
Klikom na poveznicu „Teams“ otvara se kartica sa prikazom svih ekipa. Upravitelj ima ista prava nad ekipama kao i vlasnici ekipa. Upravljanje ekipama opisano je u poglavlju 4. Kartica sa prikazom svih ekipa vidljiva je na slici 31.



Slika 31. Prikaz svih ekipa

6.4. Korisnici

Klikom na poveznicu „Users“ otvara se kartica s prikazom svih korisnika. Upravitelj može mijenjati ime, prezime i ulogu svakog korisnika. Kartica sa prikazom svih korisnika vidljiva je na slici 32.

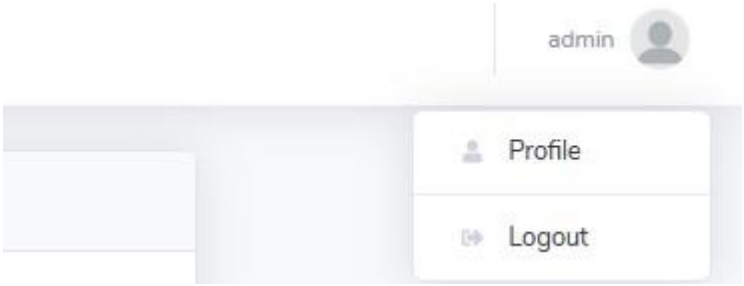


Slika 32. Kartica sa prikazom svih korisnika

Klikom na pojedinog korisnika otvara se kartica *korisnik.* Uređivanje korisničkih podataka bit će opisano u sljedećem poglavlju.

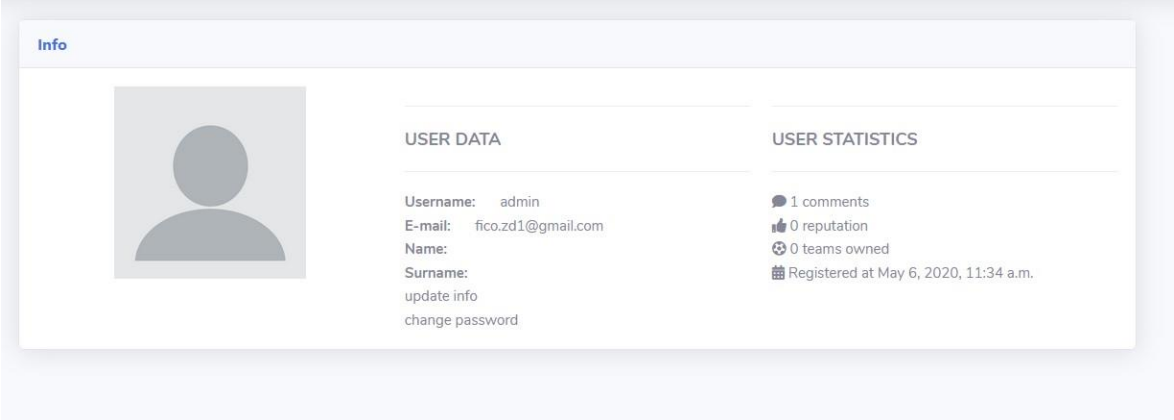
7. Korisnički podaci

U gornjem desnom uglu prikazano je korisničko ime svakog korisnika uz popratnu sliku. Klikom na isto otvara se padajući izbornik u kojemu je moguće odjaviti se iz aplikacije i otvoriti karticu korisničkog računa, vidljivo na slici 33.



Slika 33. Padajući izbornik sa stavkama odjave i otvaranja kartice korisničkog računa

Klikom na poveznicu „Profile“ otvara se kartica korisničkog računa (slika 34).



Slika 34. Kartica korisničkog računa

S lijeve strane je prikazana slika, u sredini korisnički podaci i s desne strane statistika korisnika. Klikom na poveznicu „update info“ otvara se skočni prozor u kojemu je moguće mijenjati sliku, ime i prezime. Klikom na poveznicu „change password“ otvara se prozor u kojemu je moguće mijenjati lozinku.

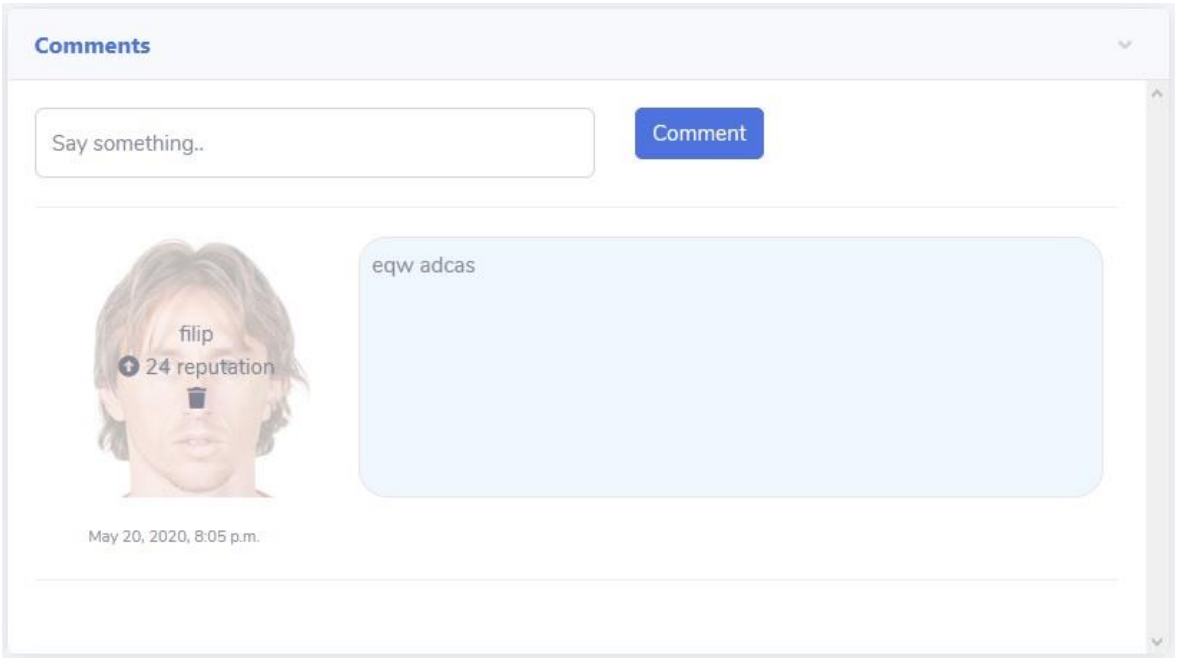
7.1. Reputacija

Svaki korisnik ima reputaciju koja kreće od broja 0. Klasa `Reputation` prikazana je u ispisu 21.

```
      class Reputation (models.Model):
          points = models.IntegerField(default = 0)
          lastUpTime = models.DateTimeField(auto_now=True)
          lastUpUser = models.ForeignKey('User', on_delete=models.CAS
CADE, null=True, blank=True, related_name="lastupuser")
```

Ispis 21. Klasa `Reputation`

Reputaciju drugom korisniku je moguće podignuti kroz komentare. Komentar je sastavljen od teksta komentara i korisnika. Postavljanjem miša na sliku korisnika prikazuje se ime korisnika, iznos reputacije i strelica za podizanje reputacije. Komentar je prikazan na slici 35.



Slika 35. Komentar

Jedan korisniku drugomu može podignuti reputaciju za jedan, svako sat vremena. Način podizanja reputacije prikazan je u ispisu 22.

```
            now = datetime.datetime.now()
            tz_info = reputation.lastUpTime.tzinfo
            diff = datetime.datetime.now(tz_info) - reputation.lastU
pTime
            diff_s =  diff.total_seconds()
            diff = divmod(diff_s, 3600)[0]
            if diff > 0 or request.user != reputation.lastUpUser:
                reputation.lastUpTime = now
                reputation.points += 1
                reputation.points
                reputation.lastUpUser = request.user
                reputation.save()
```

Ispis 22. Podizanje reputacije korisniku

8. Zaključak

Django se pokazao kao izuzetno moćan razvojni okvir. Posebno je impresivna iscrpna dokumentacija kojom se programeru početniku lako voditi. Dobro modeliran je i način ovjere (engl. *authentication*) korisnika, koji omogućuje vrlo jednostavno i lako razdvajanje korisnika po ulogama i dodjeljivanje prava pristupa. Sustav za mapiranje relacijskih entiteta o objekte (ORM) je vrlo intuitivan, te posjeduje mogućnost automigracija, što je izuzetno korisno. Microsoftov uređivač teksta Visual Studio Code je pokazao neke nedostatke. Najveća od njih je to što nema podršku za Djangov *templating engine* pa u nekim situacijama postavlja nepotrebne razmake, što rezultira nemogućnošću učitavanja (engl. *load*) statičkih datoteka.

U budućnosti bi bilo dobro osmisliti i implementirati nove tipove turnira, poput eliminacijskog tipa sa 2 utakmice u rundi i ligaškog tipa turnira. Također bi bilo korisno proširiti aplikaciju dodavajući klijentski radni okvir. Time bi se doprinijelo bogatijem korisničkom iskustvu i manjim brojem pristupa bazi podataka.

9. Literatura

[1] Django Introduction, https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (posjećeno 25.5.2020.)

[2] Learn About Bootstrap: Get Started Developing Responsive Websites, https://www.whoishostingthis.com/resources/bootstrap/ (posjećeno 25.5.2020.)

[3] History of MySQL, https://www.datasciencecentral.com/profiles/blogs/history-of-mysql (posjećeno 25.5.2020.)

[4] What is XAMPP, https://www.wpblogx.com/what-is-xampp/ (posjećeno 25.5.2020.)