Aplikacija za praćenje i analitiku esport rezultata

Ćirić, Dario

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Aplikacija za praćenje i analitiku esport rezultata

Rijeka, rujan 2024.

Dario Ćirić
0069085127

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Aplikacija za praćenje i analitiku esport rezultata

Mentor: prof. dr. sc. Ivan Štajduhar

Rijeka, rujan 2024.

Dario Ćirić
0069085127

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2024.

_____

Dario Ćirić

Sadržaj

Popis slika

Poglavlje 1

Uvod

1.1 Esport

Esport je oblik sporta kojem su primarni aspekti sporta omogućeni elektroničkim sustavima. Rezultat unosa igrača i timova direktna je posljedica posredovanja sučelja između ljudi i računala. Drugi način za definiranje esporta je kao natjecanje u videoigrama emitirano preko interneta.[1]

U ovom radu fokusirati ćemo se na videoigru Dota 2. Dota 2 je jedna od najvećih esport igri današnjice. U 2023. godini dostigla je četvrto mjesto po gledanosti, a po veličinama nagradnog fonda zauzima prvo mjesto, sa preko 320 milijuna američkih dolara nagrađenih najboljim timovima u svojem vijeku.[2]

Iako je Dota 2 jedan od najvećih esport naslova, pregled zakazanih mečeva i provjera prošlih rezultata profesionalnih mečeva prilično su teški, često zahtijevajući korištenje više različitih alata za dobivanje točnih informacija.

Kako bi se pratiteljima esporta omogućilo lakše praćenje, cilj ovog rada je osmisliti i izraditi internetsku aplikaciju Trackerfront, koja će ispuniti zahtjeve korisnika za prikaz budućih i prošlih mečeva, sa dodanom vrijednošću predviđanja rezultata budućih mečeva korištenjem strojnog učenja.

1.2 Dota 2

Dota 2 je videoigra koju je izradio Valve, izdana 2013. godine. Igra je besplatna i može se preuzeti preko Steama, Valveove internetske trgovine videoigrama.[3] U prosjeku Dota 2 istovremeno igra više od 500 tisuća igraća, a ukupna količina igrača koji su preuzeli igru procjenjuje se na preko 200 milijuna.[4]

Igra se sastoji od mečeva unutar kojih dva tima zauzimaju suparničke strane *"Radiant"* i *"Dire"*. Svaki tim se sastoji od pet igrača, koji prije početka meča odabiru svaki po jednog unikatnog heroja od 124 ponuđena, ovisno o njihovim sposobnostima i različitim stilovima igre. Kada svi igrači odaberu svoje heroje, započinje meč. Primjer jednog meča prikazan je na slici 1.1.

Meč se odvija unutar mape na kojoj je svakom timu dodijeljenja svoja bazu s obzirom na stranu koju igraju. Tijekom meča, timovi se međusobno bore za prevlast nad mapom, sakupljajući zlato i iskustvene bodove kako bi mogli poraziti protivnike. Tijekom te borbe, igrači, točnije njihovi heroji, mogu ubiti protivničkog heroja, asistirati svojem suigraču ili mogu umrijeti. Taj odnos predstavlja bodove svakog igrača, pa tako možemo pratiti *"Kills"*, *"Assists"* i *"Deaths"* pojedinog igrača. Zbroj *"kill"*-ova svakog igrača na timu predstavlja ukupni zbroj bodova tog tima. Tim koji prvi uništi protivničku strukturu *"Ancient"* pobjeđuje neovisno o bodovima.

Slika 1.1 Dota 2 profesionalni meč između Team Liquid i Team Spirit

1.3 Profesionalni mečevi

Dota 2 profesionalni mečevi pojavljuju se već u 2011. godini, dvije godine prije objavljivanja same igre. The International 2011, tada poznat kao The International, turnir je koji obilježava početak Dota 2 profesionalnih mečeva. Organizirao ga je Valve, a pozvani natjecatelji dobili su pristup ranoj verziji igre.[5]

Svaki profesionalni meč se odvija unutar turnira, sponzoriranih od strane Valvea ili drugih organizatora. Turniri se odvijaju u dva dijela. U prvom dijelu timovi su raspoređeni u grupe gdje igraju svi protiv svih, formatu poznatom kao "Round Robin", sakupljajući bodove za svaki meč. Najbolji timovi unutar grupa nastavljaju u drugi dio turnira, "Playoff", gdje su raspoređeni prema rezultatima. U drugom djelu turnira timovi koji u međusobnim okršajima pobjeđuju nastavljaju unutar turnira, dok su gubitnici eliminirani. Pobjednik posljednjeg meča je prvak turnira.

Poglavlje 2

Tehnologije

2.1 Python

Za izradu programa Trackerfront, korišten je programski jezik Python. Python je jedan od najraširenijih programskih jezika, a svoju popularnost je razvio na temelju jednostavnosti i robusnosti, te količine knjižnica dostupnih korisnicima Pythona. Za ovaj rad upotrijebljena je verzija Pythona 3.12, a njegova podloga velike količine knjižnica, čitljivosti koda i jednostavnosti uporabe znatno će ubrzati i olakšati izradu aplikacije. Python je također i najbolji programski jezik za strojno učenje zbog svoje pristupačnosti i lakoće upravljanja podatcima.[6][7]

2.1.1 Django

Jedno od najpopularnijih internetskih razvojnih okruženja je Python knjižnica Django, koju koristimo za izradu internetske aplikacije Trackerfront. Djangov konceptualni pristup izradi aplikacija je "baterije uključene", to jest pruža okruženje gdje je veliki dio najčešće korištenih svojstva internetskih aplikacija već stvoren i spreman za uporabu uz male prilagodbe i preinake. Kao jedan od najpopularnijih Python knjižnica, ima aktivnu i veliku zajednicu korisnika koji pružaju razne knjižnice prilagođene Django okruženju, a jednostavno je i naći rješenje bilo kakvih problema s kojima će se svaki programer neizbježno susresti prilikom izrade aplikacije.[8]

Prilikom izbora okruženja za ovaj projekt, ogromnu prednost Djangu nad ostalim okruženjima daje sustav za stvaranje baze podataka na temelju Python objekata.

Ovakav pristup značajno olakšava upravljanje podatcima unutar okvira internetske aplikacije, a zadržava sve odlike pouzdane baze podataka.[9]

2.1.2 SQLite

SQLite je knjižnica otvorenog koda koja implementira brzu, samostalnu i visokopouzdanu bazu podataka s potpunim značajkama *Structured Query Language (SQL)*. Jedna od prednosti SQLite je to da ne zahtjeva svoj serverski proces, nego joj se može pristupiti kroz datoteku na disku.[10]

SQLite je zadani način pohranjivanja Djangovih modela, a knjižnica za upravljanje SQLite bazama podataka sadržana je u standardnim knjižnicama Pythona, što smanjuje ovisnost o drugim paketima naše aplikacije. Iako postoje opcije korištenja drugih sustava za implementaciju baza podataka unutar Django okruženja, uzimajući u obzir nisku interakciju korisnika sa bazom podataka, prednosti u razvoju koje SQLite pruža nadvladavaju prednosti korištenja drugih sustava.

2.1.3 Requests

Python knjižnica requests omogućuje slanje *Hypertext Transfer Protocol (HTTP)* zahtjeva servisima *Representational State Transfer (REST)* arhitekture na jednostavan i brz način, a za Trackerfront je korišten za preuzimanje podataka sa internetskih servisa.[11]

REST je stil arhitekture korišten u razvoju internetskih servisa i aplikacija koji omogućava razmjenu podataka između različitih sistema preko interneta. REST definira način razvoja internetskih aplikacija i interakcije sa njima, primarno sa programskim sučeljima (eng. *Application Programming Interface (API)*) koja su prilagođena za korištenje HTTP metoda.[12]

2.1.4 Scikit-learn

Moderni algoritmi strojnog učenja za srednje velike nadzirane i nenadzirane zadatke integrirani su u Scikit-learn Python knjižnicu. Ova knjižnica pomaže s uvođenjem

strojnog učenja korisnicima s manje stručnog znanja. Fokus ove knjižice je na dokumentaciju, dosljednost APIja, performanse i jednostavnost korištenja. Njegova distribucija pod pojednostavljenom licencom i minimalne ovisnosti potiču njegovu upotrebu u komercijalnim i akademskim kontekstima.[13]

2.2 Resursi

S obzirom da naša aplikacija zahtjeva veliku količinu podataka visoke točnosti, potrebno je pronaći pouzdan izvor. Valve za Dota 2 pruža svim korisnicima pristup direktnom Steam WebAPI sa podatcima o svim mečevima koji su se odvijali ili se odvijaju.[14] Za našu aplikaciju to rješenje nije idealno jer su nam potrebni podatci samo o profesionalnim mečevima. Valve ne radi distinkciju između profesionalnih i neprofesionalnih Dota 2 mečeva, a podatci koji nam ovaj servis pruža šturi su i neobrađeni, što ih čini neprikladnima za razvoj. Iz tog razloga podatke o profesionalnim mečevima dohvaćamo putem drugih, prikladnijih servisa.

2.2.1 OpenDota

OpenDota je internet servis otvorenog koda koji pruža pristup ogromnoj količini podataka o Dota 2 mečevima. OpenDota koristi Valveov Steam WebAPI za preuzimanje informacija o mečevima, ali i za preuzimanje reprize samih mečeva.[15]

Dota 2 nakon svakog odigranog meča sve podatke o njemu pohranjuje kao reprizu (eng. *replay*), datoteku .dem formata koja se na Valveovim serverima čuva dva tjedna. Te datoteke OpenDota preuzima odmah nakon završetka meča i zatim obrađuje, pružajući nam puno više informacija nego Steam WebAPI.

Za obradu preuzetih mečeva, OpenDota koristi Clarity, program otvorenog koda za izvlačenje poželjnih podataka iz .dem datoteka. Clarity daje detaljan zapis događaja koji su se dogodili u meču, pružajući informaciju o gotovo svakom detalju koji se dogodio za vrijeme meča, ali i pregled meča. Pregled meča sadrži informacije o turnirima, timovima, igračima, pobjedniku i dužini trajanja meča.[16] Ti podatci omogućuju da OpenDota prepoznaje profesionalne mečeve, te nam time olakšava pronalazak podataka koji nam trebaju za procese Trackerfront aplikacije.

OpenDota uz svoju internet stranicu za pregled podataka ima i robustan API,

koji će naša aplikacija koristiti za preuzimanje podataka. OpenDota API nam besplatno pruža podatke o herojima, ligama, timovima i mečevima u *JavaScript Object Notation (JSON)* formatu, uz ograničenja pristupu na 2000 API poziva dnevno i 60 poziva u minuti.[17]

2.2.2 Liquipedia

Liquipedia je internetska stranica koja pruža *"wiki"* platformu za esport raznih igri, uključujući Dotu 2. Liquipediju održava zajednica volontera i ljubitelja esporta, a u esport zajednici je poznat i pouzdan izvor zbog svojeg dugogodišnjeg posvećivanja prema točnosti i ažurnosti podataka o turnirima, timovima, igrama i vijestima o događajima vezanih uz esport.

Za našu aplikaciju pristupati ćemo Liquipedia APIju, koji će nam pružati podatke o budućim mečevima i turnirima. Također, koristiti ćemo podatke s Liquipedie kako bi definirali razliku između profesionalnih, poluprofesionalnih i amaterskih mečeva, što će smanjiti količinu mečeva koje moramo preuzeti i pregledati, a povećati točnost i vjerodostojnost podataka.

2.2.3 Figma Design

Figma Design je alat za izradu prototipa internetskih stranica, s fokusom na dizajn stranice. Alat je jednostavan za korištenje i znatno ubrzava proces osmišljanja dizajna internetskih stranica.[18] Za ovaj rad u Figma Designu je sastavljen dizajn svih stranica unutar aplikacije koje će korisnik posjećivati. Dizajn se postigao kombinacijom ubacivanja pruženih i izradom novih grafičkih elemenata, sa naglaskom na minimalistički dizajn i jednostavnost korištenja. Nakon što je dizajn izrađen, pomoću alata unutar Figma Designa izvezeni su stilovi elemenata u *Cascading Style Sheets (CSS)* formatu, a slike koje će biti korištene u *Scalable Vector Graphics (SVG)* formatu.

Poglavlje 3

Aplikacija

Aplikacija Trackerfront razvijana je u dva dijela. Prvi dio aplikacije je Django aplikacija, to jest aplikacija za prikaz podataka korisnicima putem internetskog preglednika. Drugi dio aplikacije služi za preuzimanje i pohranjivanje podataka potrebnih za funkcionalnost aplikacije, te za sortiranje i vježbanje našeg modela strojnog učenja.

3.1 Izrada baze podataka

Kako bi Trackerfront aplikacija bila funkcionalna, potrebno je osmisliti način dobavljanja podataka o profesionalnim Dota 2 mečevima. Preuzimanje podataka podrazumijeva da znamo koje podatke preuzeti te da iste imamo gdje i pohraniti, pa je osmišljanje logike baze podataka prvi korak za izradu naše aplikacije.

Za stvaranje naše baze poslužiti ćemo se Django modelima, kako bi kasnije imali jednostavnu povezanost naše baze i prikaza kroz Django. Osmišljena struktura baze programira se unutar Django modela, te zatim Django stvara SQLite bazu podataka prema zadanim parametrima.[19]

3.1.1 Model Matchroom

Najbitniji model unutar naše aplikacije je model u koji će se pohranjivati podatci o samom meču, tako da je to model koji prvi izrađujemo. Model Matchroom (kodni isječak 3.1) sadrži polja za unikatan primarni ključ meča `match_id` zadan od Valvea, `match_date` koji predstavlja datum održavanja meča u Unix formatu, polje

`tourName` sadrži ime turnira unutar kojeg se meč održao. Model zatim sadrži i unikatne ključeve oba tima koji su sudjelovali u meču u poljima `team1_id` i `team2_id`, gdje prvi predstavlja *"Radiant"* a drugi *"Dire"*, zatim logičku varijablu `radiantWin` koja označava je li pobjednik *"Radiant"* tim, dužinu samog meča u sekundama zapisanog u polje `duration`. Nadalje, `radiantNwDelta` koji prikazuje razliku između osvojenog zlata između dvaju timova kao skup brojeva odvojen zarezom gdje se za svaku minutu meča zapisuje brojčana razlika u korist prvog tima. Matchroom sadrži i ukupni broj bodova koji su oba tima ostvarili pod `radiantScore` i `direScore`, te za kraj podatke o svakom igraču zasebno, počevši inkremetno od polja `p0`, pa sve do polja `p9`, koja sadrže JSON podatke o igraču uključujući njihovo ime, bodove i heroja kojega su odabrali.

Isječak koda 3.1 Model Matchroom

```
33 class Matchroom(models.Model):
34     match_id = models.IntegerField(primary_key=True)
35     match_date = models.IntegerField(null=True, blank=True)
36     tourName = models.JSONField(null=True, blank=True)
37     team1_id = models.IntegerField(null=True, blank=True)
38     team2_id = models.IntegerField(null=True, blank=True)
39     radiantWin = models.BooleanField(null=True, blank=True)
40     duration = models.IntegerField(null=True, blank=True)
41     radiantNwDelta = models.TextField(null=True, blank=True)
42     radiantScore = models.IntegerField(null=True, blank=True)
43     direScore = models.IntegerField(null=True, blank=True)
44     p0 = models.JSONField(null=True, blank=True)
45     p1 = models.JSONField(null=True, blank=True)
46     p2 = models.JSONField(null=True, blank=True)
47     p3 = models.JSONField(null=True, blank=True)
48     p4 = models.JSONField(null=True, blank=True)
49     p5 = models.JSONField(null=True, blank=True)
50     p6 = models.JSONField(null=True, blank=True)
51     p7 = models.JSONField(null=True, blank=True)
52     p8 = models.JSONField(null=True, blank=True)
53     p9 = models.JSONField(null=True, blank=True)
54
55
```

```
56      def __str__(self):
57          return f"Match {self.match_id}"
```

3.1.2 Model Team

Uzimajući u obzir model Matchroom, idući model koji je bilo potrebno kreirati je model Team (kodni isječak 3.2), koji će povezati unikatne ključeve `team1_id` i `team2_id` sa imenima, logotipovima i skraćenicama timova.

Unutar modela Team stvoriti ćemo primarni ključ `team_id`, unikatan ključ svakog tima zadan od Valvea prilikom registracije tima. Svakom timu pridružujemo ime i skraćenicu pod `team_name` i `team_tag`, te internetsku lokaciju na Valveovim serverima koja sadrži logotip tima pod `team_logo`.

Za potrebe strojnog učenja dodano je još tri polja, `team_wins` i `team_losses` koji predstavljaju ukupne pobjede i poraze svakog tima, te polje `rating`, brojčanu vrijednost Elo evaluacije svakog tima. Elo sustav rangiranja je metoda izračuna relativne razine vještina sudionika u igrama koje uključuju dvije suparničke strane, u kojima je rezultat pobjeda jedne strane i gubitak druge. Ime je dobio po mađarsko-američkom profesoru fizike, Arpadu Elu, koji je 60-tih godina osmislio sustav kao poboljšan model ocjenjivanja vještine igrača u šahu. Elo evaluacija Dota 2 timova koristi istu podlogu kao rangiranje igrača šaha, te se relativna evaluacija računa na temelju ishoda odigranih mečeva između timova.[20]

U model Team dodano je i polje `wiki_link`, koje sadrži internetsku adresu tog tima na stranicama Liquipedie, čij će nam sadržaj biti potreban za preuzimanje podataka o budućim mečevima.

Isječak koda 3.2 Model Team

```
3  class Team(models.Model):
4      team_id = models.IntegerField(primary_key=True)
5      team_name = models.CharField(max_length=100, null=True,
       blank=True)
6      team_tag = models.CharField(max_length=50, null=True, blank
       =True)
7      team_logo = models.CharField(max_length=100, null=True,
       blank=True)
```

```
8     wiki_link = models.CharField(max_length=100, null=True,
    blank=True)
9     team_wins = models.IntegerField(null=True, blank=True)
10    team_losses = models.IntegerField(null=True, blank=True)
11    rating = models.FloatField(null=True, blank=True)
12
13    def __str__(self):
14        return self.team_name
```

3.1.3 Model Hero

Model Hero (kodni isječak 3.3) stvoren je isključivo za prikaz odabranih heroja korisnicima aplikacije, pa mu je iz tog razloga struktura vrlo jednostavna. Ključ `hero_id` povezuje sa poljima `hero_name` koje sadrži ime heroja i `hero_image`, internetsku poveznicu na izvor slike heroja.

Isječak koda 3.3 Model Hero

```
25 class Hero(models.Model):
26     hero_id = models.IntegerField(primary_key=True)
27     hero_name = models.CharField(max_length=100)
28     hero_image = models.CharField(max_length=50)
29
30     def __str__(self):
31         return self.hero_name
```

3.1.4 Model League

Model League (kodni isječak 3.4) stvoren je za pohranjivanje podataka o turnirima. Polje `league_id` ponovno sadrži unikatan ključ koji predstavlja taj turnir, također zadan od Valvea, a polje `league_name` sadrži ime turnira.

Za prepoznavanje turnira prilikom preuzimanja podataka o budućim mečevima dodano je polje `league_wiki`, u kojem se nalazi internet adresa tog turnira na stranicama Liquipedie. Također je dodano i polje `do_fetch`, logička varijabla koja će

naznačiti je li se turnir već dogodio pa je potrebno preuzeti mečeve tog turnira za popunjavanje baze.

Isječak koda 3.4 Model League

```
16 class League(models.Model):
17     league_id = models.IntegerField(primary_key=True)
18     league_name = models.CharField(max_length=100, null=True,
   blank=True)
19     league_wiki = models.CharField(max_length=100, null=True,
   blank=True)
20     do_fetch = models.BooleanField(null=True, blank=True)
21
22     def __str__(self):
23         return self.league_name
```

3.1.5 Model Upcoming

Zadnji model, model Upcoming (kodni isječak 3.5), sadrži podatke o nadolazećim mečevima. Polje `upcoming_id` je primarni ključ i način raspoznavanja budućeg meča. Zatim polje `start_time` koje sadrži datum kada bi se meč trebao igrati u ISO formatu. Polja `team1_id` i `team2_id` sadrže unikatne ključeve svakog tima, a `league_link` poveznicu tog turnira na stranicama Liquipedie. Polje `last_cached` služi za provjeru ažurnosti podataka, a logičko polje `predictedRadiant` sadrži izlaz našeg modela strojnog učenja za taj meč.

Isječak koda 3.5 Model Upcoming

```
60 class Upcoming(models.Model):
61     upcoming_id = models.CharField(primary_key=True, max_length
   =10)
62     start_time = models.IntegerField(null=True, blank=True)
63     league_link = models.CharField(max_length=100, null=True,
   blank=True)
64     team1_id = models.IntegerField(null=True, blank=True)
65     team2_id = models.IntegerField(null=True, blank=True)
66     last_cached = models.FloatField(null=True, blank=True)
```

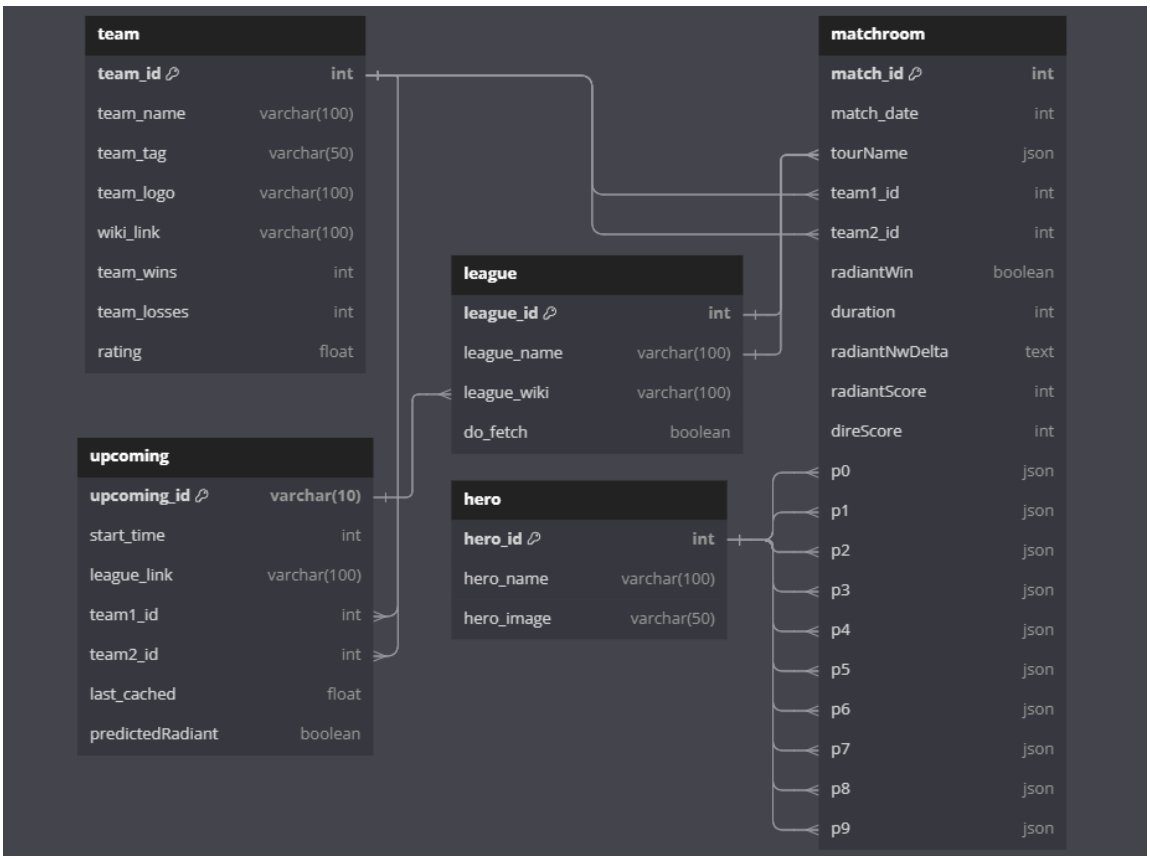
```
67      predictedRadiant = models.BooleanField(null=True, blank=
        True)
68
69      def __str__(self):
70          return self.upcoming_id
```

3.1.6 Stvaranje baze

SQLite bazu podataka stvaramo korištenjem Django naredbi za stvaranje migracija i migriranje (kodni isječak 3.6). Kada smo to izvršili Django stvara bazu podataka prema zadanim modelima. Shema baze koju smo dobili i njene relacije prikazane su na slici 3.1.

Isječak koda 3.6 Stvaranje baze podataka

```
1 py manage.py makemigrations
2 py manage.py migrate
```



Slika 3.1 Shema baze podataka

3.2 Preuzimanje i obrada podataka

Jedan od dva dijela Trackerfront aplikacije zadužen je za preuzimanje i obradu podataka koji će biti prezentirani korisniku kroz internetsku aplikaciju. Podatci se preuzimaju sa već navedenih APIja internet servisa OpenDota i Liquipedia. Za preuzimanje podataka je zadužen niz samostalnih Python skripti koje vrše inicijalizaciju podataka u našu bazu podataka.

Korištenjem zasebnih skripti koje pristupaju bazi direktno i odvajanjem od Django okruženja omogućujemo laku izmjenu i fleksibilnost, s obzirom da su internet servisi koje koristimo podložni promjenama. Također, ne narušavamo funkcionalnost aplikacije prikazanu prema krajnjem korisniku u slučaju problema s dostupnošću internet servisa s kojih preuzimamo podatke.

Za sve podatke osim nadolazećih mečeva koristimo OpenDota API naveden ranije u tekstu. OpenDota API ima plaćen pristup, ali i besplatan pristup uz ograničenja na 2000 poziva dnevno i 60 poziva u minuti. S obzirom na naše potrebe i činjenicu da je inicijalizaciju dovoljno napraviti jednom, Python skripte za preuzimanje podataka s OpenDota APIja prilagođene su granicama besplatnog pristupa.

Za preuzimanje i pohranu podataka koristiti ćemo Python knjižnice `requests` i `sqlite3`, od kojih nam prva daje jednostavan i pregledan pristup HTTP zahtjevima[21], a druga mogućnost izvršavanja SQL koda nad SQLite bazama podataka.[11]

3.2.1 Profesionalni Turniri

Prije nego što počnemo preuzimati podatke o profesionalnim mečevima, odrediti ćemo što za našu aplikaciju predstavlja profesionalni Dota 2 meč. S obzirom na visoku promjenjivost unutar Dota 2 esport scene, kako bi za naš model strojnog učenja imali precizne podatke i mogli napraviti predviđanja, nećemo uključiti amaterske i poluprofesionalne Dota 2 esport turnire. Iz tog razloga, fokus ove aplikacije biti će na *"Tier 1"* profesionalnim turnirima. Takvi turniri nude izvanredan nagradni fond i okupljaju najbolje profesionalne timove na svijetu, a vode ih dobro poznati organizatori te se smatraju prestižnima unutar Dota 2 esport zajednice.[22]

Te turnire potrebno je unesti u našu bazu podataka, a s obzirom na ograničenja Liquipedia APIja i zabranu bilo kojeg drugog programskog prikupljanja podataka sa stranice, taj unos odrađujemo ručno. Za to nam je potreban program koji može pristupiti i urediti SQLite bazu podataka, u ovom slučaju program otvorenog koda DB Browser for SQLite.[23] Izvor podataka je Liquipedia, a podatke zapisujemo u tablicu `trackerfront_league`. S obzirom da je Dota 2 igra s kontinuiranom uslugom (eng. *live service game*), podložna je velikim promjenama kroz životni vijek, pa tako relevantni podatci brzo zastarijevaju. Za izradu ovog rada uneseni turniri su ograničeni na turnire unutar 2023. i 2024. godine kako bi se izbjeglo narušavanje kakvoće podataka. Naravno, s obzirom da se unosi ručno, moguće je ovaj opseg povećati ili smanjiti po potrebi. Nakon ovog koraka započinje inicijalizacija podataka u bazu.

3.2.2 Mečevi unutar turnira

Nakon što su turniri uneseni, potrebno je preuzeti unikatne ključeve svakog meča u svim zadanim turnirima. To vrši skripta `get_match_ids_by_league`, koja čita podatke o turnirima iz tablice, te zatim koristeći OpenDota API dohvaća JSON podatke koji u sebi sadrže unikatne ključeve `match_id`. U kodnom isječku 3.7 prikazana je glavna funkcija ove skripte.

Isječak koda 3.7 Funkcija get_pro_matches za preuzimanje popisa mečeva

```
13 def get_pro_matches(league_ids):
14     matches = []
15     for league_id in league_ids:
16         url = f'https://api.opendota.com/api/leagues/{league_id
   }/matches'
17         response = requests.get(url)
18         if response.status_code == 200:
19             matchlist = response.json()
20         else:
21             print(f'Failed to fetch matches', response.
   status_code)
22         for match in matchlist:
23             matches.append(match['match_id'])
24     return matches
```

3.2.3 Preuzimanje prošlih mečeva

Kada program izvrši pronalaženje svih unikatnih mečeva u zadanim turnirima, potrebno je te mečeve i preuzeti. Tome služi skripta `get_match_data_by_id`, koja ima dvije funkcije. Prvo, prema pronađenim unikatnim ključevima mečeva pomoću OpenDota APIja dohvaća JSON odgovor koji sadrži podatke potrebne za popunjavanje tablice Matchroom. Zatim, za preuzeti meč stvara novi red u tablici, te ispunjava sva polja sa podatcima iz odgovora. Kada stvori taj zapis, potpuni odgovor se dodatno sprema kao JSON datoteka u slučaju da se program ili format tablice želi izmijeniti, kako bi se izbjeglo ponovno preuzimanje svih podataka.

Količina mečeva u svim turnirima prelazi ograničenja besplatnog pristupa OpenDota APIju, pa su u skripti postavljeni zastoji koji će pauzirati rad skripte dok nova kvota poziva ne bude dostupna. U slučaju korištenja platnog modela, moguće je ukloniti ova ograničenja.

3.2.4 Podatci timova

Nakon što smo uspješno popunili tablicu svih profesionalnih mečeva, poznati su nam svi timovi koji su u tim mečevima sudjelovali. Pronalaskom svakog unikatnog ključa iz polja `team1_id` i `team2_id` unutar Matchroom tablice, možemo saznati sve timove koji su ikada sudjelovali u profesionalnim mečevima (kodni isječak 3.8).

Pomoću unikatnih ključeva tih timova sa OpenDota APIja preuzimamo podatke o tim timovima te ih zatim sve pohranjujemo u tablicu Team.

Isječak koda 3.8 Funkcija get_unique_team_ids za pronalaženje ključeva timova

```
6  def get_unique_team_ids(db_path):
7      try:
8          conn = sqlite3.connect(db_path)
9          cursor = conn.cursor()
10         cursor.execute('SELECT DISTINCT team1_id FROM
       trackerfront_matchroom')
11         team1_ids = [row[0] for row in cursor.fetchall()]
12         cursor.execute('SELECT DISTINCT team2_id FROM
       trackerfront_matchroom')
13         team2_ids = [row[0] for row in cursor.fetchall()]
14         unique_team_ids = list(set(team1_ids + team2_ids))
15         return unique_team_ids
16     except sqlite3.Error as e:
17         print(f"Database error: {e}")
18         return []
19     finally:
20         if conn:
21             conn.close()
```

3.2.5 Podatci heroja

Podatci o herojima koje možemo uskladiti sa podatcima mečeva preuzetih s OpenDota APIja nalaze se na GitHub repozitoriju *dotaconstants*. Sa repozitorija preuzimamo datoteku `heroes.json`, te zatim pomoću skripte `heroes_into_base` unesemo u bazu potrebne podatke o herojima za prikaz krajnjem korisniku. Od ponuđenih podataka najbitniji su nam identifikacijski ključ heroja, kojeg možemo upariti sa ključevima svakog igrača unutar meča, ime i poveznica na sliku heroja.

3.2.6 Stvaranje modela strojnog učenja

Nakon uspješnog popunjavanja baze, imamo podatke potrebne za treniranje modela strojnog učenja. Kako bi započeli sa strojnim učenjem, potrebno je istražiti koji model je povoljan za naše podatke. S obzirom na manjak znanstvenih radova i članaka o modelima korištenim za predviđanje esport rezultata, primjere nalazimo iz tradicionalnih sportova. Kod predviđanja rezultata za nogomet[24] ili košarku[25], model koji ima najveću prosječnu točnost je *Random Forest*, pa taj model primjenjujemo unutar aplikacije.

U knjižnici scikit-learn koju koristimo za ovaj projekt nalazi se model RandomForestClassifier. Ovaj model strojnog učenja prilagođava brojna stabla odlučivanja na različite uzorke skupa podataka, te zatim bira stabla koja su imala najbolje rezultate.[26] Za potrebe naše aplikacije skupovi uzoraka ograničeni su na 100 uzoraka, a za testiranje ostaviti ćemo 20 posto podataka (kodni isječak 3.9).

Isječak koda 3.9 Parametri modela strojnog učenja

```
181     X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
182     model = RandomForestClassifier(n_estimators=100,
    random_state=42)
183     model.fit(X_train, y_train)
```

Podatci koje koristimo za treniranje modela strojnog učenja sastavljeni su od podataka svih mečeva u našoj bazi. Za svaki meč zapisujemo ključeve oba tima i pobjednika, zatim pronalazimo Elo rang svakog tima, koliko pobjeda svaki tim ima

u prethodnih deset mečeva, za koliko zlata je tim vodio u tim prethodnim mečevima, te koliko timovi imaju pobjeda u zadnjih 5 međusobnih okršaja (kodni isječak 3.10).

Isječak koda 3.10 Priprema podataka za treniranje modela

```
144     for match in matches:
145         match_team1_id, match_team2_id, radiantWin = match
146         match_team1_elo = get_elo_rating(match_team1_id, conn)
147         match_team2_elo = get_elo_rating(match_team2_id, conn)
148 
149         match_team1_recent_matches = get_last_n_matches(
    match_team1_id, 10, conn)
150         match_team2_recent_matches = get_last_n_matches(
    match_team2_id, 10, conn)
151 
152         match_team1_recent_wins = count_wins(
    match_team1_recent_matches, match_team1_id)
153         match_team2_recent_wins = count_wins(
    match_team2_recent_matches, match_team2_id)
154 
155         match_team1_avg_gold_lead = compute_average_gold_lead(
    match_team1_recent_matches, match_team1_id)
156         match_team2_avg_gold_lead = compute_average_gold_lead(
    match_team2_recent_matches, match_team2_id)
157 
158         match_head_to_head_matches = get_head_to_head_matches(
    match_team1_id, match_team2_id, 5, conn)
159         match_team1_h2h_wins = count_wins(
    match_head_to_head_matches, match_team1_id)
160         match_team2_h2h_wins = len(match_head_to_head_matches)
    - match_team1_h2h_wins
161 
162         match_features = [
163             match_team1_elo,
164             match_team2_elo,
165             match_team1_recent_wins,
166             match_team2_recent_wins,
```

```
167             match_team1_h2h_wins,
168             match_team2_h2h_wins,
169             match_team1_avg_gold_lead,
170             match_team2_avg_gold_lead
171         ]
172         match_target = 1 if radiantWin else 0
173         data.append((match_features, match_target))
```

Nakon obrađivanja svakog meča dobivene podatke pretvaramo u DataFrame klasu koju nam pruža knjižnica pandas, te iste raspoređujemo u podatke o mečevima i ciljeve istih (kodni isječak 3.11). Nakon tog koraka spremni smo započeti treniranje modela.

Isječak koda 3.11 Prilagodba podataka za treniranje modela

```
176     df_train = pd.DataFrame(data, columns=['features', 'target'
    ])
177     X = list(df_train['features'])
178     y = df_train['target']
```

Nakon uspješnog treniranja model koji smo odabrali ima točnost predviđanja od 58.91%, što je konzistentno sa točnošću takvih modela primijenjenih na tradicionalne sportove.[24][25] Taj model zatim spremamo kako bi ga mogli koristiti u sustavima aplikacije bez potrebe za ponovnim treniranjem modela.

3.2.7 Nadolazeći mečevi

Zadnji potrebni podatci su podatci o nadolazećim mečevima. Za razliku od ostalih skripti, skripta `get_upcoming` se odvija unutar Django procesa, s obzirom da je te podatke potrebno često ažurirati. Prilikom otvaranja početne stranice se poziva Django naredba `updateMatches`, koju smo prije toga stvorili i registrirali u Django okruženje.[27] Ta naredba kao prvi korak provjerava kada je zadnje ažurirana tablica nadolazećih mečeva. Ako je tablica ažurirana prije više od tri sata, poziva se funkcija skripte `get_upcoming` koje preuzimaju podatke o nadolazećim mečevima i ažuriraju tablicu (kodni isječak 3.12). Također, ta funkcija će se pozvati na naš izrađeni model strojnog učenja te predvidjeti rezultat meča pomoću `predictWin` funkcije.

Podatci o nadolazećim mečevima preuzimaju se sa stranica Liquipedie, a te podatke nam pruža API sa GitHub repozitorija *beeequeue/dota-matches-api*.[28] S obzirom da nas zanimaju podatci samo sa turnira sadržanih u našoj tablici turnira, potrebno je dobiveni JSON odgovor filtrirati po `leagueUrl` stavki, uspoređujući sa poljima `wiki_link` naše tablice turnira.

Isto tako, JSON odgovor sadrži samo imena i Liquipedia adrese timova koji sudjeluju u meču, pa ih je potrebno usporediti sa `wiki_link` poljima tablice timova, te zatim iz nje prema tim podatcima vratiti vrijednost polja `team_id` u odgovarajućem redu.

Isječak koda 3.12 Funkcija process_matches za dohvaćanje nadolazećih mečeva

```
6  def process_matches(api_url, db_path):
7      matches = fetch_matches(api_url)
8      clear_cached_table(db_path)
9      if matches:
10         wiki_links = get_wiki_links(db_path)
11         for match in matches:
12             league_url = match.get('leagueUrl')
13             if league_url and league_url in wiki_links:
14                 team1 = match.get('teams')[0]
15                 team2 = match.get('teams')[1]
16                 if team1.get('url') and team2.get('url'):
17                     match_data = {
18                         'hash': match.get('hash'),
19                         'startsAt': match.get('startsAt'),
20                         'leagueUrl': match.get('leagueUrl'),
21                         'team1_id': team1.get('url'),
22                         'team2_id': team2.get('url'),
23                         'last_cached': int(time.time())
24                     }
25                     match_data['team1_id'], match_data['
    team2_id'] = check_team_ids(team1.get('url'), team2.get('url
    '), db_path)
26                     insert_match_data(db_path, match_data)
```

3.3 Prikaz podataka

Drugi dio Trackerfront aplikacije zadužen je za prikaz podataka prema krajnjem korisniku. Za ovaj dio implementirani su Django pogledi (eng. *views*)[29] i predlošci (eng. *templates*)[30], koji će na modularan način moći krajnjem korisniku prikazati relevantne podatke. Dizajn prikaza prati planirani dizajn stvoren u alatu Figma.

3.3.1 Temeljni predložak

Predlošci su ključni dio izrade internetskih stranica u Djangu, te nam omogućuju dinamičnu promjenu podataka i korištenje Django sintakse unutar samih HTML datoteka, to jest predložaka (eng. *Django template language*).[31] Ta sintaksa nam dozvoljava korištenje mnogih logičkih operacija te pridruživanja (eng. *extend*) predložaka drugim predlošcima.

Temeljni predložak naše aplikacije nalazi se u datoteci `main.html`, te sadrži sve zajedničke elemente svih pogleda unutar aplikacije. Unutar temeljnog predloška definirani su glavni elementi poput stilske i skriptne datoteke. Također, pridružen je i predložak `navbar.html`, s obzirom da je poželjno navigacijsku traku imati u svakom od pogleda (kodni isječak 3.13). Sintaksa `{% block content %}` označava početak, a `{% endblock content %}` kraj opsega unutar kojega će se pridruživati ostatak predložaka.

Isječak koda 3.13 Temeljni predložak main.html

```
<!DOCTYPE html>
{% load static %}
<html>
<head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <title>Trackerfront</title>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <link rel='stylesheet' type='text/css' media='screen' href="{% static 'styles/globals.css' %}">
```

```
10      <link rel='stylesheet' type='text/css' media='screen' href=
    "{% static 'styles/style.css' %}">
11      <script src="{%static 'scripts/main.js' %}"></script>
12  </head>
13  <body>
14      <div class="main">
15          {% include 'navbar.html' %}
16
17          {% block content %}
18
19          {% endblock content %}
20      </div>
21  </body>
22  </html>
```

3.3.2 Pogled openMain

Prva stranica sa kojom će se susresti krajnji korisnik kada pristupi našoj aplikaciji definirana je funkcijom `openMain` (kodni isječak 3.14). Ovaj pogled će iz naše baze izvući po pet zadnjih odigranih i nadolazećih mečeva, te podatke svih timova. Te informacije pohranjuju se u Python rječnik koji zatim prosljeđujemo u naš predložak `home.html`

Isječak koda 3.14 Pogled openMain

```
43  def openMain(request):
44      call_command('updateMatches')
45      displayMatches = Matchroom.objects.all().order_by('-
    match_date')[:5]
46      displayUpcoming = Upcoming.objects.all().order_by('
    start_time')[:5]
47      displayLeagues = League.objects.all()
48      teams = Team.objects.all()
49      context = {'pastMatches':displayMatches, 'upcomingMatches':
    displayUpcoming, 'team':teams, 'league':displayLeagues}
50      return render(request, 'trackerfront/home.html', context)
```

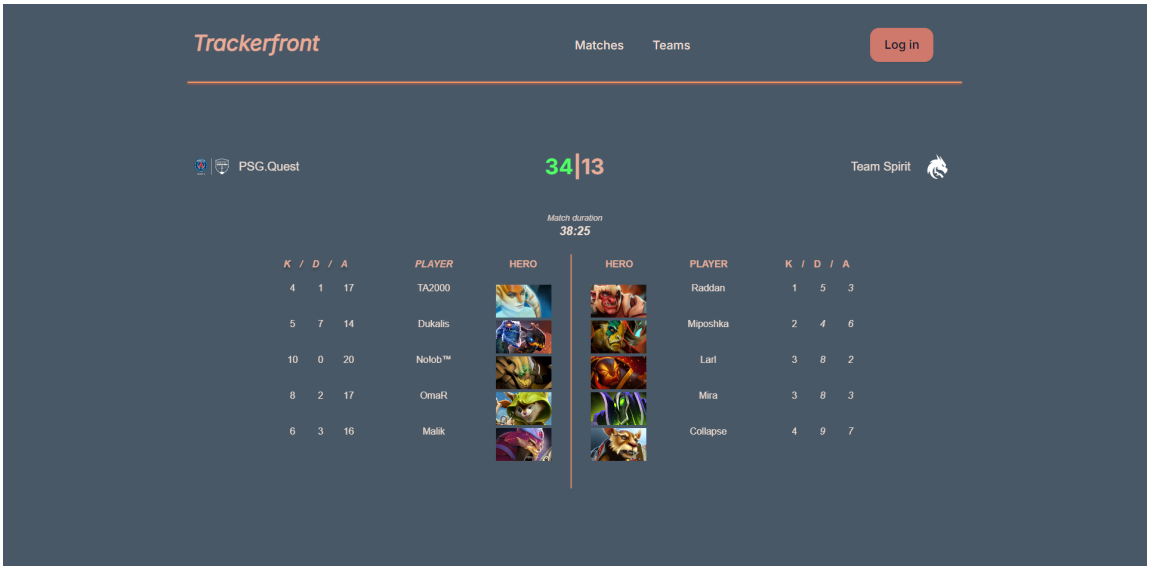
Predložak `home.html` koristi naredbe pružane putem Django sintakse za dinamički prikaz timova koji su sudjelovali u mečevima, podatke o pobjedniku ili predviđenom pobjedniku meča, bodove za odigrane mečeve i vrijeme održavanja nadolazećih mečeva. Također, naredbe pomažu pri stiliziranju elemenata kako bi se na efektivan način korisniku prikazale informacije. Ovaj predložak koristi i javascript za sakrivanje elementa `spoiler-button`, koji je prilikom prikaza podataka stvoren da korisnik ne vidi bilo koji rezultat koji za njega nije poželjno vidjeti. Pritiskom na element on se sakriva, te je moguće vidjeti rezultat meča i pristupiti stranici s detaljima istog. U tablici nadolazećih mečeva timovi čija je pobjeda predviđena označeni su zlatnom bojom, a datum i vrijeme kada će se meč igrati automatski se prilagođava vremenskoj zoni krajnjeg korisnika. Rezultat koji korisnik vidi prikazan je na slici 3.2.

Slika 3.2 Početni prikaz

3.3.3 Pogled openMatches

Korisnicima aplikacije svakako treba omogućiti pregled detalja pojedinog meča, a tome služi pogled kojeg definira funkcija `openMatches`. Ovaj pogled prema primarnom ključu proslijeđenim kroz internet adresu dohvaća odgovarajući red u modelu

Matchroom. Zatim se dohvaćaju podatci o heroju svakog igrača te podatci o oba tima. Ti podatci se dalje prezentiraju kroz predložak `matchroom.html`. Prikazuju se imena i logotipi timova, bodovi timova za taj meč, te su bodovi pobjednika označeni zelenom bojom. Isto tako prikazana je statistika za svakog igrača, uključujući sliku heroja kojeg je svaki igrač igrao u tom meču (slika 3.3).



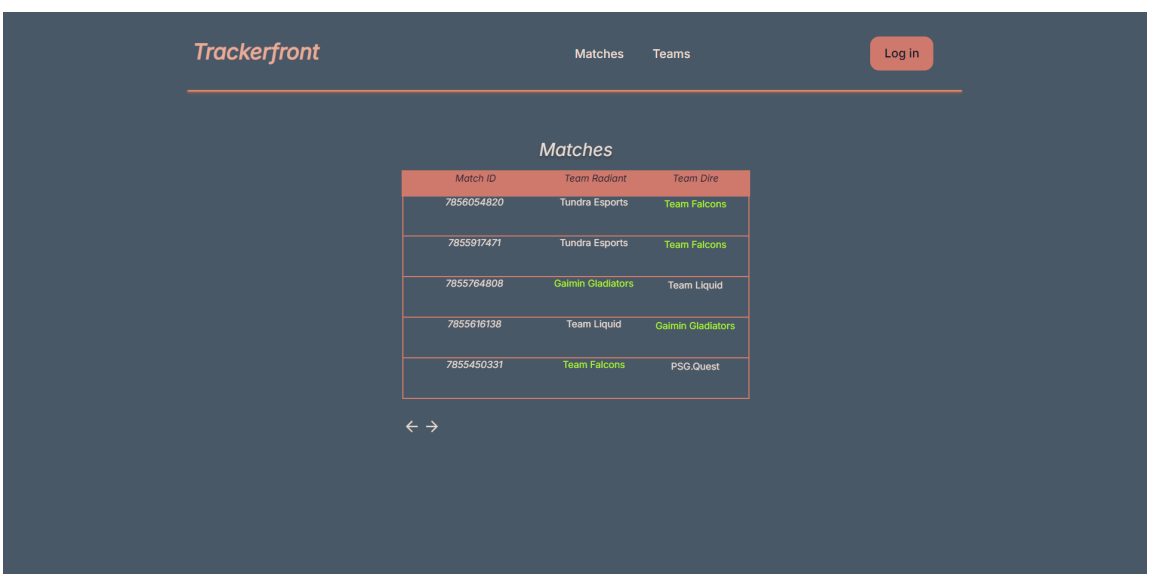
Slika 3.3 Prikaz detalja o meču

3.3.4 Pogled openMatchlist

U slučaju da krajnji korisnik želi vidjeti više od posljednjih pet mečeva koji su mu pruženi na početnoj stranici, to može učiniti putem tipke *Matches*. Tada se otvara pogled definiran funkcijom `openMatchlist` (kodni isječak 3.15), koji preuzima pet redova iz modela Matchroom prema datumu odvijanja meča. Korisniku je omogućeno pomicati ovaj opseg unaprijed i unatrag za vrijednost iteracijske varijable unutar internet adrese. U funkciji je implementirana zaštita od prekoračenja iteracijskog broja, kako korisnici ne bi mogli izazvati grešku ručnim unošenjem podataka. Ovaj pogled vraća predložak `matchlist.html`, koji prikazuje tablicu sa unikatnim ključevima mečeva i imenima timova, gdje je pobjednički tim označen zelenom bojom (slika 3.4).

Isječak koda 3.15 Pogled openMatchlist

```
94 def openMatchlist(request, iterate):
95     teams = Team.objects.all()
96     matchList = Matchroom.objects.all().order_by('-match_date')
97     maxcount = matchList.count()
98     cursor = (iterate - 1) * 5
99     if cursor < (maxcount - 5):
100         matchList = matchList[cursor:cursor+5]
101     else:
102         matchList = matchList[cursor:maxcount]
103     context = {'mlist': matchList, 'team':teams, 'maxmatches':
    (maxcount // 5 + 1), 'iterate':iterate}
104     return render(request, 'trackerfront/matchlist.html',
    context)
```



Slika 3.4 Prikaz popisa svih mečeva

3.3.5 Pogledi openTeams i openTeamroom

Pogledu `openTeams` pristupa se putem tipke *Teams* unutar navigacijske trake. Funkcija `openTeams` koju taj pogled poziva ponaša se jednako kao i funkcija `openMatchlist`,

uz razliku da umjesto popisu mečeva pristupa redovima u modelu Team. Predložak `teams.html` koji nam ova funkcija vraća prikazuje popis timova sa njihovim imenima, skraćenicama i unikatnim ključevima.

Pritiskom na bilo koji tim s popisa, funkcija `openTeamroom`, implementirana na veoma sličan način, vraća predložak `teamroom.html`. Korisniku se prikazuju logo i ime tima, uz tablicu njihovih mečeva sa detaljima o turniru, vremenu događanja i primarnom ključu meča.

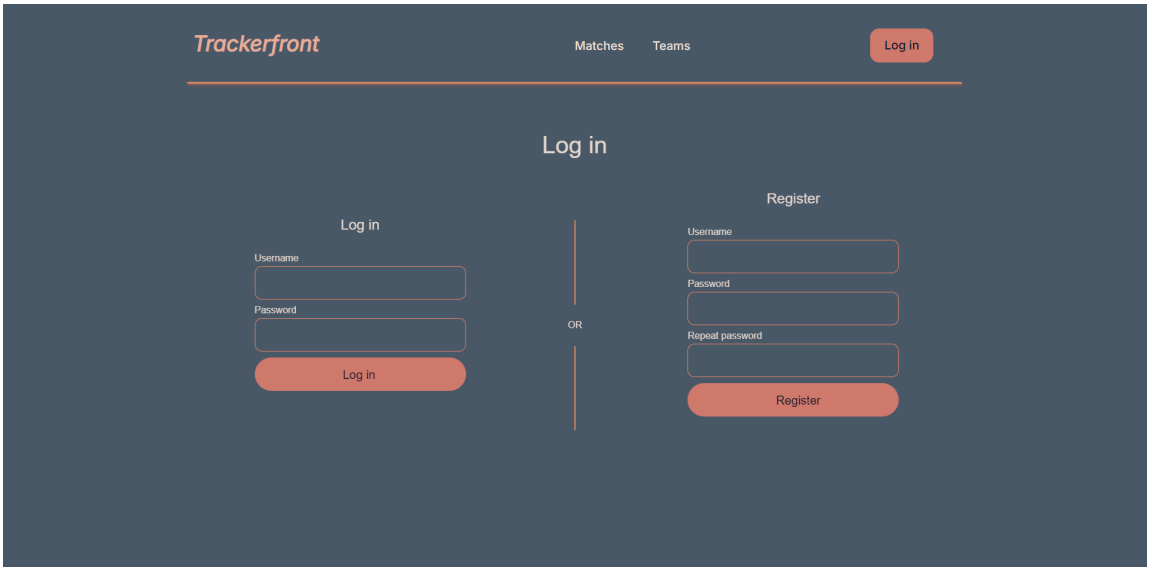
3.3.6 Pogled openLogin

Aplikacija Trackerfront pruža korisnicima mogućnost prijave. Za implementaciju ove mogućnosti korišten je sustav autentifikacije ugrađen u Django.[32] Pristup prijavi korisnicima je pružen pritiskom na tipku *Log in* koja poziva funkciju `openLogin` (kodni isječak 3.16). Korisniku se zatim otvara stranica sa obrascima za prijavu ili stvaranje računa putem predloška `login.html` (slika 3.5). Nakon što korisnik upiše potrebne podatke, funkcija na temelju pritisnute tipke *Log in* ili *Register* vrši odabranu akciju pomoću unesenih podataka i CSRF tokena, kojega Django automatski generira. Ako je neki od podataka netočno unesen, korisniku će se prikazati poruka s detaljima pogreške. U slučaju uspjeha, korisnik se vraća na početnu stranicu.

Isječak koda 3.16 Pogled openLogin

```
8  def openLogin(request):
9      if request.method == 'POST':
10         action = request.POST.get('action')
11
12         if action == 'registerpress':
13             username = request.POST.get('r_username')
14             password = request.POST.get('r_password')
15             repeat_password = request.POST.get('
       r_repeat_password')
16
17             if password != repeat_password:
18                 messages.error(request, "Passwords do not match
       .")
```

```
19              elif User.objects.filter(username=username).exists
    ():
20                  messages.error(request, "Username already
    exists.")
21              else:
22                  user = User.objects.create_user(username=
    username, password=password)
23                  user.save()
24                  login(request, user)
25                  return redirect('home')
26
27          elif action == 'loginpress':
28              username = request.POST.get('l_username')
29              password = request.POST.get('l_password')
30              user = authenticate(request, username=username,
    password=password)
31
32              if user is not None:
33                  login(request, user)
34                  return redirect('home')
35              else:
36                  messages.error(request, "Invalid username or
    password.")
37           return render(request, 'trackerfront/login.html')
```



Slika 3.5 Prikaz forme za prijavu

Poglavlje 4

Zaključak

U ovom radu izrađena je aplikacija za praćenje i analitiku esport rezultata pod nazivom Trackerfront. Korisnicima internetske aplikacije pružene su informacije o rezultatima profesionalnih Dota 2 mečeva, te je primjenom strojnog učenja razvijen algoritam predviđanja rezultata nadolazećih mečeva.

Aplikacija uz dobar balans pristupačnog dizajna i prikaza važnih podataka o natjecanjima omogućuje korisnicima brzo i interaktivno informiranje o Dota 2 esportu, te pruža servis u kojem korisnici mogu na jednom mjestu pronaći informacije o proteklim i nadolazećim mečevima, koje drugi internet servisi ne pružaju. Upotrebom alata Figma Design na pristupačan i intuitivan način osmišljen je grafički dizajn stranice bez oslanjanja na vanjske izvore.

Kroz razvoj aplikacije predstavljene su mogućnosti Django razvojnog okruženja za operacije nad bazama podataka, te predstavljanje istih u okruženju internet aplikacije. Nadalje, prikazane su neke od prednosti i jednostavnosti načina uporabe Pythona i njegovih knjižnica poput requests, sqlite3 i scikit-learn. Također, pokazali smo kako je uporabom APIja raznih internet servisa moguće lako i besplatno doći do velike količine potrebnih podataka koji su inače teško dostupni.

U aplikaciji Trackerfront uključen je i element predviđanja pobjednika u nadolazećim Dota 2 profesionalnim mečevima, za potrebu čega se scikit-learn pokazao kao odličan izvor informacija o strojnom učenju i jednostavnosti njegove implementacije. Mogućnost predviđanja pobjednika pruža korisniku dodanu vrijednost korištenja aplikacije i produljuje njegov angažman s aplikacijom.

Uz nastavak razvoja i unaprjeđenja, aplikacija Trackerfront može korisnicima pružati još više mogućnosti. Primjeri područja u kojima bi se aplikacija mogla poboljšati su povezana uz interakciju korisnika i stranice, poput mogućnosti objavljivanja komentara o mečevima. Kontinuirano ažuriranje aplikacije bi omogućilo da administratori stranice dodaju informacije o timovima i turnirima koje ne dolaze iz podataka o mečevima.

Literatura

[1] J. Hamari i M. Sjöblom, "What is eSports and why do people watch it?" *Internet research*, sv. 27, br. 2, str. 211–232, 2017.

[2] Esports Charts: "Dota 2 Esports Viewership and Statistics", s Interneta, `https://escharts.com/games/dota2`, kolovoz 2024.

[3] Steam: "Dota 2 on Steam", s Interneta, `https://store.steampowered.com/app/570/Dota_2/`, kolovoz 2024.

[4] SteamDB: "Dota 2 SteamDB charts", s Interneta, `https://steamdb.info/app/570/charts/`, kolovoz 2024.

[5] Liquipedia: "The International 2011", s Interneta, `https://liquipedia.net/dota2/The_International/2011`, kolovoz 2024.

[6] M. Butwall, P. Ranka i S. Shah, "Python in Field of Data Science: A Review," *International Journal of Computer Applications*, sv. 178, br. 49, str. 20–24, rujan 2019., ISSN: 0975-8887. DOI: `10.5120/ijca2019919404`.

[7] H. Singh i V. Dhir, "Effectiveness Of Python Libraries In Machine Learning: A Review," *Webology (ISSN: 1735-188X)*, sv. 16, br. 1, 2019.

[8] Django: "The web framework for perfectionists with deadlines", s Interneta, `https://www.djangoproject.com/`, kolovoz 2024.

[9] Django: "Models", s Interneta, `https://docs.djangoproject.com/en/5.1/topics/db/models/`, kolovoz 2024.

[10] K. P. Gaffney, M. Prammer, L. Brasfield, D. R. Hipp, D. Kennedy i J. M. Patel, "Sqlite: past, present, and future," *Proceedings of the VLDB Endowment*, sv. 15, br. 12, 2022.

[11] Requests: "HTTP for Humans™", s Interneta, `https://requests.readthedocs.io/en/latest/`, kolovoz 2024.

[12] Codecademy: "What is REST?" s Interneta, `https://www.codecademy.com/article/what-is-rest`, kolovoz 2024.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort i dr., "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, sv. 12, str. 2825–2830, 2011.

[14] Steam Community: "Steam Web API Documentation", s Interneta, `https://steamcommunity.com/dev`, kolovoz 2024.

[15] The OpenDota Blog: "FAQ", s Interneta, `https://blog.opendota.com/2014/08/01/faq/`, kolovoz 2014.

[16] GitHub: "skadistats/clarity", s Interneta, `https://github.com/skadistats/clarity`, kolovoz 2024.

[17] OpenDota: "OpenDota API", s Interneta, `https://docs.opendota.com/`, kolovoz 2024.

[18] N. Ibrahim, A. Y. Chandra, E. M. Saari, P. T. Prasetyaningrum i I. Pratama, "The Effectiveness of Web 2.0 Tools Training Workshop Using Canva and Figma in Developing Creative Visual Content," *Asian Journal of Assessment in Teaching and Learning*, sv. 13, br. 2, str. 35–45, prosinac 2023., ISSN: 2821-2916. DOI: `10.37134/ajatel.vol13.2.4.2023`.

[19] Django: "Databases", s Interneta, `https://docs.djangoproject.com/en/5.1/ref/databases/`, kolovoz 2024.

[20] A. Ebtekar i P. Liu, "Elo-mmr: A rating system for massive multiplayer competitions," *Proceedings of the Web Conference 2021*, 2021., str. 1772–1784.

[21] Python Documentation: "sqlite3 — DB-API 2.0 interface for SQLite databases", s Interneta, `https://docs.python.org/3/library/sqlite3.html`, kolovoz 2024.

[22] Liquipedia Dota 2 Wiki: "Tier 1 Tournaments", s Interneta, `https://liquipedia.net/dota2/Tier_1_Tournaments`, kolovoz 2024.

[23] DB Browser for SQLite: "About", s Interneta, `https://sqlitebrowser.org/about/`, kolovoz 2024.

[24] F. Rodrigues i Â. Pinto, "Prediction of football match results with Machine Learning," *Procedia Computer Science*, sv. 204, str. 463–470, 2022., ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.08.057.

[25] T. Horvat, J. Job, R. Logozar i Č. Livada, "A Data-Driven Machine Learning Algorithm for Predicting the Outcomes of NBA Games," *Symmetry*, 2023. DOI: 10.3390/sym15040798.

[26] scikit-learn documentation: "RandomForestClassifier", s Interneta, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, kolovoz 2024.

[27] Django: "How to create custom django-admin commands", s Interneta, https://docs.djangoproject.com/en/5.1/howto/custom-management-commands/, kolovoz 2024.

[28] GitHub: "beeequeue/dota-matches-api", s Interneta, https://github.com/BeeeQueue/dota-matches-api, kolovoz 2024.

[29] Django: "Writing views", s Interneta, https://docs.djangoproject.com/en/5.1/topics/http/views/, kolovoz 2024.

[30] Django: "Templates", s Interneta, https://docs.djangoproject.com/en/5.1/topics/templates/, kolovoz 2024.

[31] Django: "The Django template language", s Interneta, https://docs.djangoproject.com/en/5.1/ref/templates/language/, kolovoz 2024.

[32] Django: "Using the Django authentication system", s Interneta, https://docs.djangoproject.com/en/5.1/topics/auth/default/, kolovoz 2024.

Pojmovnik

API Application Programming Interface. 5–7, 14–18, 21, 30

CSS Cascading Style Sheets. 7

HTTP Hypertext Transfer Protocol. 5, 15

JSON JavaScript Object Notation. 7, 9, 16, 21

REST Representational State Transfer. 5

SQL Structured Query Language. 5, 15

SVG Scalable Vector Graphics. 7

Sažetak

U ovom radu izrađena je aplikacija Trackerfront, koja je osmišljena za praćenje i analizu esport rezultata, točnije za Dota 2 mečeve. Aplikacija korisnicima pruža sveobuhvatne informacije o prošlim i nadolazećim mečevima, uz dodatak algoritma strojnog učenja za predviđanje ishoda mečeva. Trackerfront kombinira pristupačan dizajn s učinkovitim prikazom ključnih podataka, nudeći jedinstvenu uslugu koja se ne može pronaći na drugim platformama. Proces razvoja naglašava mogućnosti Django okvira, jednostavnost korištenja Python knjižnica kao što su requests, sqlite3 i scikit-learn, te potencijal integracije APIja za prikupljanje podataka.

Ključne riječi — **Dota 2, esport, Django, scikit-learn**

Abstract

This thesis presents the development of the Trackerfront application, designed for tracking and analyzing esports results, specifically for Dota 2 matches. The application provides users with comprehensive information about past and upcoming matches, enhanced by a machine learning algorithm for predicting match outcomes. Trackerfront combines an accessible design with the effective presentation of key data, offering a unique service not found in other platforms. The development process highlights the capabilities of the Django framework, the ease of using Python libraries like requests, sqlite3, and scikit-learn, and the potential of API integration for data acquisition.

Keywords — **Dota 2, esport, Django, scikit-learn**