Web za igru "Uno"

Brandejs, Dominik

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

WEB ZA IGRU "UNO"

Završni rad

Dominik Brandejs

Osijek, godina. (Title stil)

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Dominik Brandejs
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	AR4706, 24.09.2018.
JMBAG:	0165080014
Mentor:	Marina Peko, dipl. ing. el.
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Robert Šojo, univ. mag. ing. comp.
Član Povjerenstva 1:	Marina Peko, dipl. ing. el.
Član Povjerenstva 2:	mr. sc. Željko Štanfel
Naslov završnog rada:	Web za igru "Uno"
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Napraviti web stranicu na koju se mogu uključiti do 4 igrača (potrebna je registracija za igranje) te mogu igrati igricu "uno". Svaki korisnik ima podatke o svojim statistikama igranja.

Datum ocjene pismenog dijela završnog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	26.09.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	26.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 26.09.2024.

Ime i prezime Pristupnika:	Dominik Brandejs
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AR4706, 24.09.2018.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Web za igru "Uno"**

izrađen pod vodstvom mentora Marina Peko, dipl. ing. el.

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Pri odabiru teme završnog rada utjecala je ljubav prema društvenim igrama. Prema kompleksnosti postoje jednostavne igre kao „Čovječe ne ljuti se“, ali i jako kompliciranе igre koje sadrže puno kompleksnih dijelova. Igre koje s igraju s kartama odlične su jer karte zauzimaju jako malo mjesta i možemo ih pronaći svugdje, ali kada nemamo karte računala su sljedeća najbolja stvar. Svatko danas pri ruci ima mobitel pa bez problema društvo može odigrati partiju kojoj se može pridružiti i prijatelj koji se nalazi na drugom kraju svijeta.

Igra Uno se istaknula kao očit izbor zbog svoje prepoznatljivosti, jednostavnosti i brzine igre, kako bi korisnici mogli što prije započeti partiju bez dodatnih instalacija, igra unutar web preglednika se nameće kao pravi izbor.

1.1. Zadatak završnog rada

Svrha ovog rada je napraviti web aplikaciju za igranje popularne društvene igre „Uno“. Ova web aplikacija riješiti će problem udaljenosti između igrača ili ako se igrači nalaze u blizini, ali nemaju igrače karte potrebne za igru. Gotovo svaki uređaj ima web preglednik pa će gotovo svatko moći otvoriti preglednik , registrirati se ili s privremenim računom pristupiti igri i zabaviti se s prijateljima. Za kreiranje ove web aplikacije koristio sam razvojni okvir Django koji koristi programski jezik Python. Ovaj razvojni okvir sadrži brojne elemente koji olakšavaju i ubrzavaju razvoj. Zbog prirode igre potrebna web aplikacija ne smije osvježiti stranicu tijekom igre, zbog toga ćemo koristiti proširenje za razvojni okvir Django pod nazivom Channels koje će nam omogućiti asinkronu komunikaciju između klijenta i poslužitelja uz pomoć Web utičnica(*engl. WebSocket*). Klijentska strana je dizajnirana uz pomoć okvira Bootstrap i grafike koje ćemo koristiti su vektorske tako da na zaslonima svih veličina i rezolucija sučelje izgleda dobro. Također ćemo za upravljanje igrom s klijentske strane i za komunikaciju koristiti JavaScript.

2. SLIČNE APLIKACIJE

Uno je jedna od najpopularnijih društvenih igara pa postoji velik broj aplikacija koje omogućavaju njezino igranje. Nativnu aplikaciju za stolna računala i aktualne igrače konzole razvila je tvrtka Ubisoft 2016 godine, dok je za mobilne uređaje aplikaciju razvila tvrtka Mattel koja je kupila licencu za ovu društvenu igru te proizvodi i službene igrače karte. Osim službenih verzija igre postoje web aplikacije Four Colors Runo, DUO With Friends kao i mnoge druge verzije ove društvene igre.

Sl. 2.1. Web aplikacija Four Colors

3. KORIŠTENI ALATI I TEHNOLOGIJE

Za izradu ove web aplikacije korištena je nekolicina alata i tehnologija, ovdje će biti opisane i biti će objašnjeno zašto su one odabrane.

3.1. Django

Django je razvojno okruženje za web aplikacije koje je napisano programskim jezikom Python pa se sukladno tome Python koristi za razvoj web aplikacija unutar Djanga. Temelji se na model-pogled-upravitelj(*engl. model-view-controller*) arhitekturi. Primarni cilj ovog okruženja je pojednostaviti web aplikacije koje se temelje na kompleksnim bazama podataka, također daje naglasak na korištenje i jednostavno „uključivanje“ postojećih komponenti. Django ima integriran velik broj aplikacija i komponenti koje se mogu koristiti navođenjem istih u konfiguracijskoj datoteci ili ako se aplikacija nije integrirana, može se dohvatiti i instalirati pomoću Python upravitelja paketa pip, pomoću kojeg se instalira i Django.

3.1.1. Channels

Zbog načina na koji igra Uno funkcionira potrebna je konstanta komunikacija između klijenta i servere i podatni na klijentima moraju biti konstantni pa se web stranica ne smije osvježiti. Zbog toga se koristi komponenta za Django pod nazivom Channels koja omogućava asinkronu komunikaciju između klijenta i poslužitelja i obratno. Channels koristi osnovne jedinice koda koje obrade dolaznu informaciju, odlučuju kamo je proslijediti i tako ju proslijede. Za poslužitelj i web utore se koriste komponente Daphne i Redis

3.2. HTML

HTML je opisni jezik za izradu web stranica, osnovi je dio svake web stranice. Osmislio i kreirao ga je fizičar Tim Berners-Lee, on je također zaslužan za WWW, URL i HTTP koji su temelj interneta kakvoga danas znamo. Sadržaj dokumenta se nalazi unutar oznaka elementa ovisno o sadržaju dokumenta. Neki od osnovnih elemenata HTML-a su naslov, paragraf, tablica, natuknice. Već pomoću osnovnih elemenata mogu se stvoriti web stranice bogate sadržajem, kvalitetnim svrstavanjem sadržaja unutar elemenata postiže se web stranica koja je jednostavne za konzumaciju. Iako je moguće imati jednu jednostavnu stranicu, hiperveze omogućuju web sjedišta, odnosno skup povezanih web stranica, takav odnos web stranica je jednostavan za shvatiti i koristiti te se na njemu temelji Internet.

3.3. CSS

Stilski jezik kojim se opisuje izgled HTML dokumenata. Zbog nedostataka HTML-a, Hakon Wium Lie 1994. godine predlaže kreiranje CSS-a, pred kraj 1996. godine CSS je bio dovršen i krajem godine postao službenim dijelom Web standarda. Selektorom se odabire element koji treba poprimi izgled te unutar deklaracijskog bloka se opisuje izgled koji želimo postići. Osim grupe elemenata selektorom se može ciljati i klasu elemenata ili jedan element pomoću identifikacijske oznake. Elementu se tada može urediti izgled, ali i pozicija unutar dokumenta, kao i margine i ispunu unutar elementa.

3.4. Bootstrap

Bootstrap je programski okvir za dizajniranje web stranica. Ovaj programski okvir su kreirali djelatnici Twittera Mark Otto i Jacob Thornton kako bi olakšali i ujedinili dizajn unutar tvrtke, te su ga 19. kolovoza 2011 godine objavili kao projekt otvorenog koda. Bootstrap programski okvir omogućuje jednostavno kreiranje responzivnih web stranica, odnosno prilagodljivih prema veličini zaslona uređaja. Bootstrap za responzivnost koristi sustav rešetki, unutar "kontejnera" nalazi se do 12 stupaca koji su na većim zaslonima horizontalno susjedni, dok na manjim zaslonima stupci se prebacuju u sljedeći red. Osim jednostavnog i kvalitetnog rješenja za responzivnost, Bootstrap omogućuje korištenje predefiniranog dizajna HTML elemenata. Kako odredili koji dizajn element koristi potrebno mu je dodijeliti klasu tog elementa.

3.5. JavaScript

Tvrtka Netscape 1995. godine kreira i dodaje u svoj web preglednik programski jezik JavaScript kako bi riješili problem gdje su sve stranice bile statične. Osim što JavaScript omogućuje web stranicama da budu dinamične. 2005 godine Jesse James Garrett kreira i opisuje pojam Ajax. Ajax tehnologije opisuju mogućnost učitavanja podataka u pozadini bez ponovnog učitavanja web stranice. Pojavom Ajax tehnologija popularnost JavaScript značajno raste, iako se popularnost povećala nakon Ajax tehnologija većina web stranica ih ne koristi, ali čak 99 % web stranica koriste JavaScript.

3.6. SQLite

SQLite je najrasprostranjeniji sustav baza podataka na svijetu. Koriste ga mnogi operacijski sustavi kao i web preglednici. Kreirao ga je Dwyne Richard Hipp 2000. godine, kod je otvorene domene što znači da mu se može pristupiti, ali da bi doprinijeli potrebno je biti član SQLite

Konzorcija. SQLite je sustav koji je lagan, brz, samostalan, visoke pouzdanosti, zbog toga se može naći u svakom mobilnom telefonu i skoro svakom računalu.

3.7. Docker

Docker virtualizacijom na razini operacijskog sustava omogućava pokretanje softvera u kontejnerima. Kontejneri su međusobno izolirani ali mogu komunicirati preko definiranih kanala. Docker je dostupan na Linux i Window operacijskim sustavima te će se kontejner jednako ponašati na svim platformama jer se pokreće na Docker sustavu koji sadrži sve potrebne dodatke za pokretanje programa. Docker također koristi manje resursa od virtualnog stroja.

3.8. Redis

Redis je brza baza podataka koja pohranjuje podatke unutar memorije. Zbog toga što se baza nalazi u memoriji Redis omogućava čitanje i pisanje s jako malo latencijom. Zbog čestog pisanja i čitanja podataka pri asinkronom prijenosu podataka Redis je savršena baza podataka za korištenje uz Channels.

3.9. DDNS

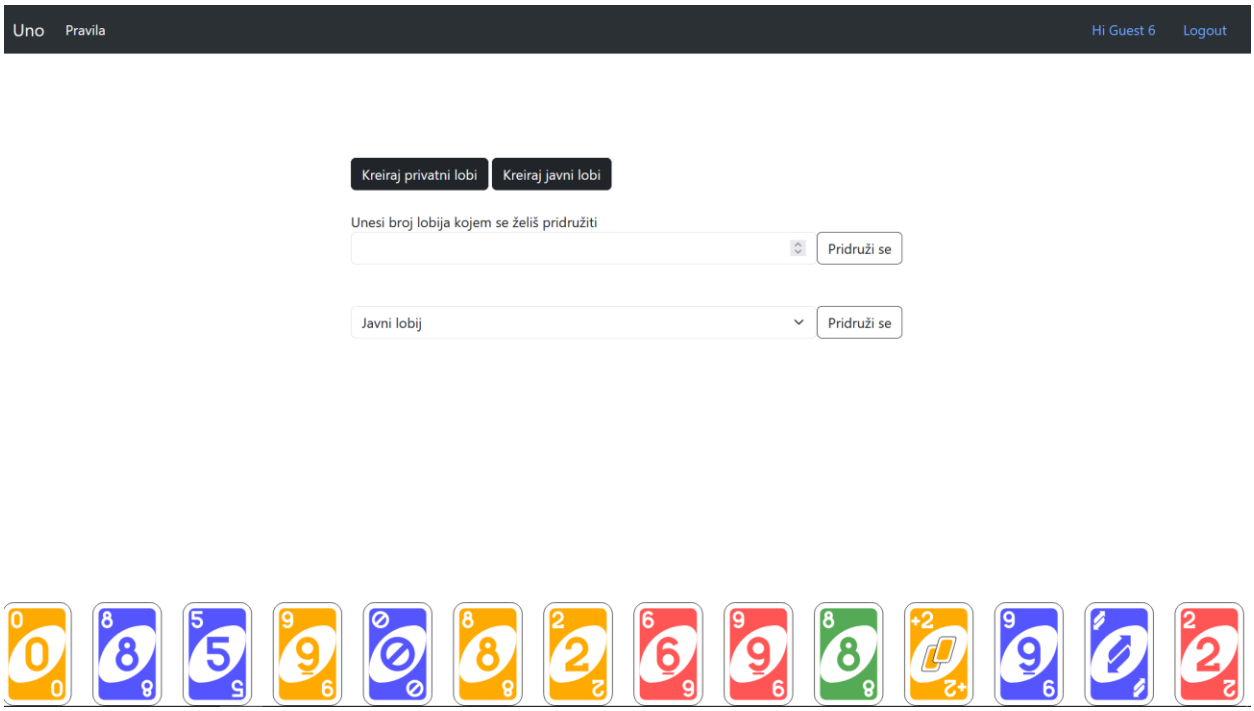
Dinamički DNS omogućava poslužiteljima koji nemaju statičku IP adresu usmjeravanje domene prema trenutnoj IP adresi. Internetski poslužitelj kućnim i malim poslovnim korisnicima često mijenjaju IP adresu, tada korisnici usluga ne mogu usmjeriti svoju domenu prema svom poslužitelju. Taj problem rješava DDNS.

4. OPIS WEB APLIKACIJE

Web aplikacija „Uno“ omogućuje korisnicima igranje društvene igre „Uno“ uz brzu registraciju

4.1. Početna stranica

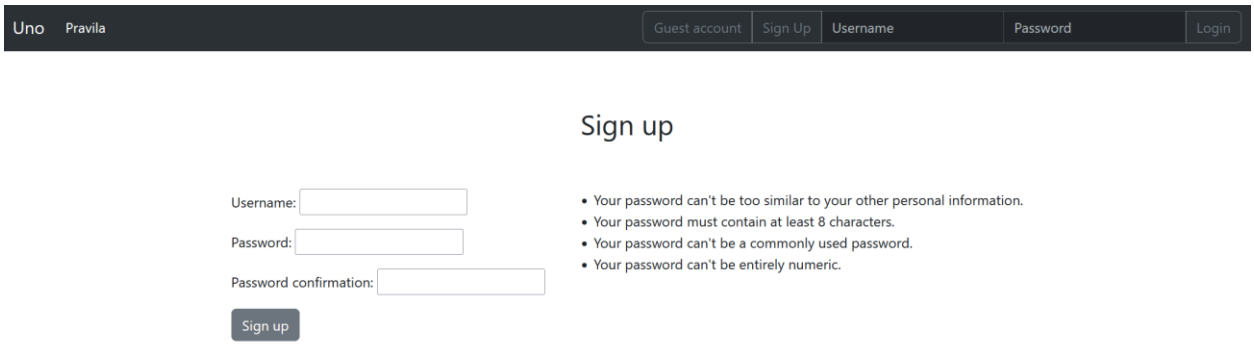
Početna stranica ima napisana pravila koja se koriste za igranje na web aplikaciji. Ukoliko korisnik nije ulogiran nudi mu se izbor korištenja privremenog računa, registracije ili prijave. Ukoliko korisnik ima račun ne treba otvarati stranicu za prijavu nego je forma za prijavu dostupna na početnoj stranici. Nakon što se korisnik prijavi nudi mu se opcija kreiranja virtualnog lobija ili pristup već kreiranom virtualnom lobija.



Sl. 4.1. Početna stranica

4.2. Forma za registraciju

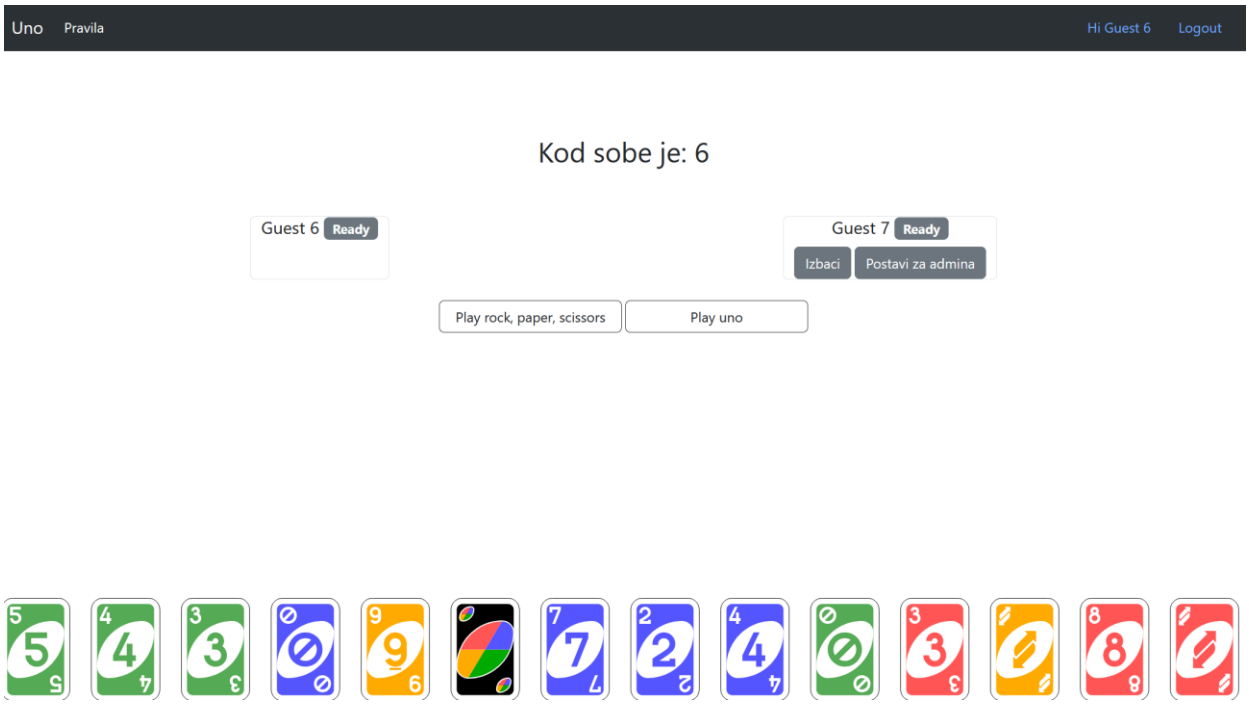
Ukoliko se korisnik odluči za izradu trajnog računa umjesto privremenog to može napraviti na stranici koja sadrži formu za registraciju.



Sl. 4.2. Stranica za registraciju korisnika

4.3. Virtualni lobi

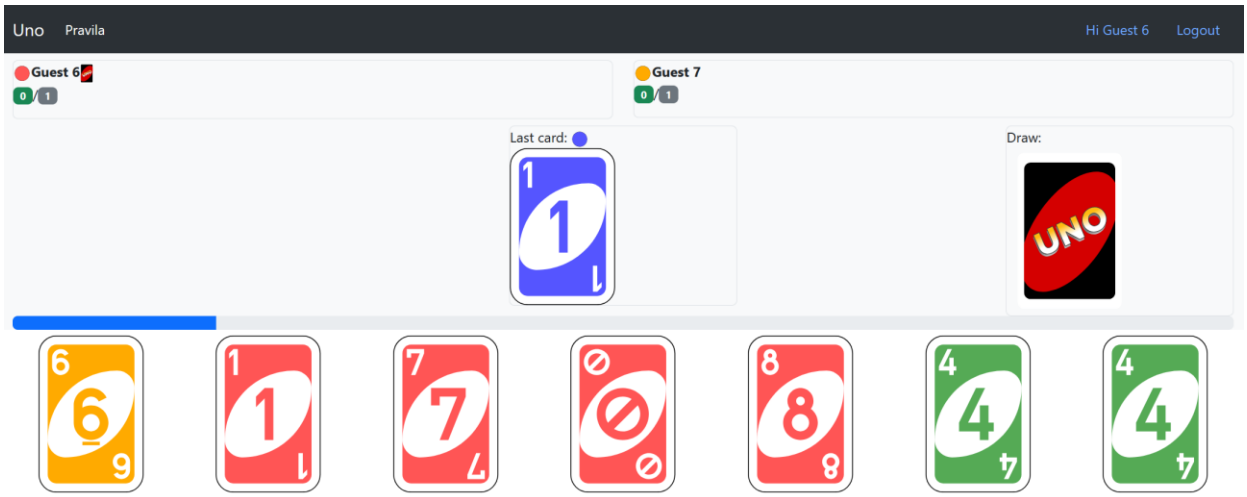
Prijavljeni korisnik prije igre mora pristupiti lobiju, korisnik koji kreira lobi postaje administrator lobija i prikazan mu je kod lobija koji mora podijeliti s drugim korisnicima kako bi mogao započeti igru, igrači koji su pristupili lobiju i tipka kojom počinje igru koja postaje aktivna tek kada se lobiju pridruži drugi igrač.



Sl. 4.3. Virtualni lobi

4.4. Uno igra

Nakon pokretanja igre, korisnici su proslijeđeni na stranicu na kojoj je igra. Unutar igre su prikazana imena igrača, uz ime igrača je prikazana stražnja strana Uno karte koja prikazuje koji je igrač trenutno na redu. Ispod reda s igračima je prikazana zadnje odigrana karta te je uz nju prikazana boja ukoliko je korisnik odigrao posebnu kartu za promjenu boje. Uz zadnje odigranu kartu je stražnja strana koje predstavlja špil, klikom na špil igrač koji je na redu izvlači kartu i stavlja je u svoju ruku. U trećem redu se nalaze karte koje igrač ima u ruci, na početku igre svaki igrač dobiva 7 slučajnih karata, ukoliko korisnik koristi manji zaslon tada karte prelaze u sljedeći redak, na mobilnom telefonu broj karata u retku su dvije. Kada su karte korisniku prikazane u više redova te ih korisnik mora pomaknuti prema dolje, redak s prikazanom zadnjom kartom i špilom je fiksiran na vrh zaslona. Također nakon što korisnik odigra kartu karta se makne iz retka ruke i prikazuje kao posljednja odigrana karta, te se oznaka pomiče na sljedećeg korisnika. Ukoliko korisnik odigra posebnu kartu promjene boja prikazuje se izbornik u kojemu korisnim može odabrati jednu od četiri moguće boje ili izaći ukoliko je odlučio odigrati drugu kartu.



Sl. 4.4. Sučelje igre

5. PROGRAMSKO RJEŠENJE

5.1. Kreiranje i kofiguriranje projekta

Za rad s programskim okvirom Django potreban je Python, neki operacijski sustavi dolaze s programskim jezikom Python, no na Windows operacijskom sustavu potrebno ga je instalirati, izvršna datoteka za instalaciju Pythona dostupna je za preuzimanje na službenoj stranici. Zbog jednostavnosti pokretanja i verzioniranja koristi se git, koji je također dostupan na službeno web stranici. Zbog lakšeg praćenja verzija paketa koristiti će se Python virtualno okruženje venv.

Linija ***Kod***

```
1:      python -m venv django
2:      Source django/Scripts/activate
```

Sl. 5.1. Naredba za kreiranje i pokretanje virtualnog okruženja venv

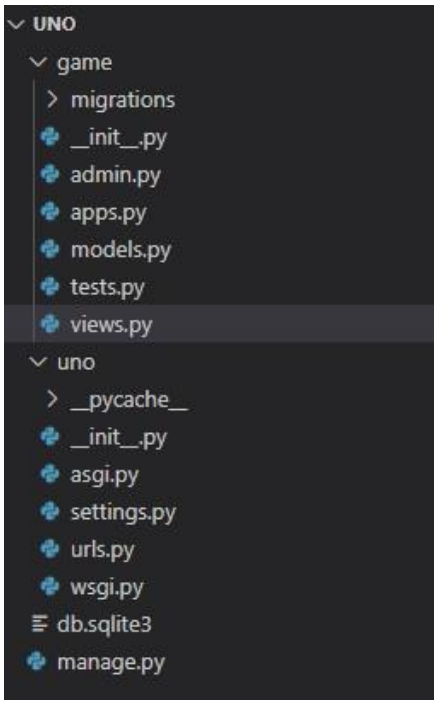
Nakon pokretanja virtualnog okruženja uz pomoć programa za instaliranje paketa za Python pip, instalira se Django te se kreira novi projekt. Također nakon kreiranja projekta, unutar istog se kreira aplikacija game.

Linija ***Kod***

```
1:      python -m pip install Django
2:      django-admin startproject uno
3:      python manage.py startapp polls
```

Sl. 5.1. Naredba za instaliranje Djanga i kreiranje projekta uno

Projekt sada možemo otvoriti u preferiranom editoru koda, uz pomoć git bash naredbenog retka možemo pokrenuti projekt u editoru Visual Studio Code naredbom code.



Sl. 5.3. Datoteke projekta Uno s aplikacijom game

Osim aplikacije „game“ u kojoj je napisana većina koda za pozadinu(*engl. backend*) ovog završnog rada, u ovom projektu koristiti će se aplikacije „Accounts“ za autorizaciju korisnika, Channels za asinkronu komunikaciju između klijenta i poslužitelja i Daphne koja služi kao poslužitelj. Aplikacija Accounts se već nalazu unutar Djanga, dok će se aplikacije Channels i Daphne preuzeti i instalirati pomoću programa za instaliranje paketa pip. Za asinkronu komunikaciju pri korištenjem aplikacije Channels potrebna je i baza podataka koja se nalazi u memoriji Redis. Redis se pokreće unutar programa za virtualizaciju Docker, pri prvom pokretanju Docker će samostalno preuzeti Redis koji se nakon toga trenutno pokreće. Za komunikaciju između aplikacije Channels i Redis služiti će nam sučelje koje se također preuzima programom za instaliranje programa pip.

Linija ***Kod***

```
1:      python manage.py startapp accounts
2:      python -m pip install -U 'channels[daphne]'
3:      python -m pip install channels_redis
4:      docker run --rm -p 6379:6379 redis:7
```

Sl. 5.4. Naredbe za instaliranje i pokretanje preostalih aplikacija

Nakon instalacije potrebnih aplikacija, potrebno je konfigurirati postavke i rute. Postavke projekta se nalaze u datoteci „settings.py“, ondje se nalazi lista instaliranih aplikacija u koju se

dodaju aplikacije Accounts, Channels i Daphne kako bi Django znao da postoje, također se dodaje stavka u kojoj definiramo IP adresu i port baze podataka Redis kako bi znali gdje možemo komunicirati s njom. Nakon postavki još se definiraju rute za asinkronu komunikaciju servera i korisnika kreiranjem datoteke „routuing.py“ unutar aplikacije „game“, ondje se dodaju rute koja povezuje rute s klasama odgovornim za komunikaciju.

```
9   websocket_urlpatterns = [
10      re_path(r"ws/lobby/(?P<lobby_id>\w+)/$", consumers.GameLobbyConsumer.as_asgi()),
11      re_path(r"ws/uno/(?P<lobby_id>\w+)/$", consumers.GameUnoConsumer.as_asgi()),
12  ]
13
14  application = ProtocolTypeRouter({'websocket': AuthMiddlewareStack(URLRouter(websocket_urlpatterns)),
```

Sl. 5.5. Putanje za asinkronu komunikaciju spojene s odgovarajućim klasama

Unutar datoteke „urls.py“ definiraju se rute koje povezuju metode iz projekta „uno“ i aplikacije „game“ za regularnu HTTP komunikaciju.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path("accounts/", include("accounts.urls")),
    path("accounts/", include("django.contrib.auth.urls")),
    path('game', include('game.urls')),
    path('', include(('game.urls', 'game'), namespace='game'))
```

Sl. 5.6. Putanje spojene s odgovarajućim aplikacijama

```
8   urlpatterns = [
9       path('', views.index, name='index'),
10      path('lobby/', views.createLobby, name='createLobby'),
11      path('lobby/<int:lobby_id>/', views.lobby, name='game_lobby'),
12      path('uno/<int:lobby_id>', views.uno, name='uno'),
13      path('login', views.login, name='login'),
14      path('logout', views.logout_view, name='logout_view'),
15      path('guest', views.guest, name='guest'),
```

Sl. 5.7. Putanje spojene s odgovarajućim metodama aplikacije game

5.2. Autentifikacija

Kako bi korisnik mogao igrati mora imati korisnički račun, web aplikacija „Uno“ je zamišljena da korisnik može što jednostavnije i brzo pristupiti igri. Pri kreiranju računa potrebno je samo

korisničko ime koje nije zauzeto i lozinka koja mora ostvariti uvjete koje Djangov ugrađeni sustav autentifikacije zahtjeva. Ako korisnik ne želi kreirati svoj račun može kreirati privremeni s jednim pritiskom gumba. Kako bi se pojednostavila prijava, forma za istu se nalazi na navigacijskoj traci koja se pojavljuje na svim stranicama kao i na početnoj.

Pri registraciji korisnika koristi se generička metoda koja će i generirati formu i proslijediti ju na sučelje

```
6   class SignUpView(generic.CreateView):
7       form_class = UserCreationForm
8       success_url = reverse_lazy("/")
9       template_name = "game/registration/signup.html"
```

Sl. 5.8. Generička metoda koja generira formu za registraciju

Za prijavu, odjavu i kreiranje privremenog korisnika kreirane su nove i jednostavnije metode te se ne koriste već generirane forme nego će se prilagoditi željenim zahtjevima i dizajnu sučelja.

```
123 def login(request):
124     username = request.POST.get('username')
125     password = request.POST.get('password')
126     user = auth.authenticate(username=username, password=password)
127     if user is not None and user.is_active:
128         auth.login(request, user)
129     else:
130         messages.info(request, 'Invalid username or password')
131         return render(request, "game/rockpaperscissors.html")
132     return redirect('/')
133 def logout_view(request):
134     logout(request)
135     return redirect('/')
136 def guest(request):
137     guests = Guest.objects.create()
138     while User.objects.filter(username='Guest %d' %guests.id).exists():
139         guests = Guest.objects.create()
140     guest=User.objects.create_user('Guest %d' %guests.id,'','')
141     auth.login(request, guest)
142     Guest.objects.filter(pk__in =Guest.objects.order_by("-created_at")[1:]).delete()
143     return redirect('/')
```

Sl. 5.9. Metode prijavu, odjavu i kreiranje privremenog korisnika

5.3. Baza podataka

SQLite je zadana baza podataka Django programskog okruženja, iako se preporučuje za jednostavnije projekte, baza podataka je jednostavna s nekoliko tablica i rijetko se u nju piše te će SQLite baza podataka zadovoljite zahtjeve rada. Zbog toga što je SQLite zadana baza podataka odmah ju možemo koristiti. Django ima specifičan način komuniciranja s bazama podataka, nakon kreiranja aplikacije „game“ unutar mape aplikacije možemo pronaći datoteku „models.py“. U toj datoteci se kreiraju modeli u obliku klase koje će kreirati tablice u bazi podataka, unutar klase definiraju se polja koja će tablica imati kao i njihove atribute. Django će kreirati ekvivalentne tablice s redcima nakon što u naredbenom retku pokrenemo migraciju

Linija ***Kod***

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

Sl. 5.10. Naredbe za kreiranje tablica u bazi podataka prema modelu

```
lass Lobby(models.Model):
   creator = models.ForeignKey(User, on_delete=models.CASCADE)
   numberOfPlayers=models.PositiveSmallIntegerField(default=0)
   winner = models.ForeignKey(User, default=None, related_name="game_winner", on_delete=models.SET_NULL, null=Tru
   def __str__(self):
         return (str(self.id)+ '->' + str(self.creator))
lass Player(models.Model):
    lobby = models.ForeignKey(Lobby, on_delete=models.CASCADE)
    player = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    active = models.BooleanField(default=False)
    move = models.TextField(max_length=9, default='', blank=True)
    def __str__(self):
        return (str(self.lobby.id)+ '->' + str(self.player.username))
lass Guest(models.Model):
   created_at = models.DateTimeField(editable=False, default=timezone.now)
   numberOfGuestAccounts = models.PositiveIntegerField(default=0)

   def addGuestUser(self):
      self.numberOfGuestAccounts+=1

   def __str__(self):
      return (str(self.id)+ '->' + str(self.numberOfGuestAccounts))
lass Wins(models.Model):
   player = models.ForeignKey(User, on_delete=models.CASCADE)
   numberOfWins=models.PositiveSmallIntegerField(default=0)
   numberOfGames=models.PositiveSmallIntegerField(default=0)

   def __str__(self):
      return (str(self.player.username)+ '->' + str(self.numberOfWins)+'/'+str(self.numberOfGames))
```

Sl. 5.21. Modeli potrebni za web aplikaciju „Uno“

Osim kreiranja tablice za bazu podataka potrebno spremati i dohvaćati podatke ih njih. Kod klasičnih Django aplikacija podatcima iz baze podataka se dohvaćaju i spremaju u metodama

unutar datoteke „views.py“, ova aplikacija također dohvaća i sprema podatke prema uputstvima iz datoteke koja je zadužena za asinkronu komunikaciju „consumers.py“

```
def createLobby(request):
    newLobby = Lobby(creator=request.user)
    newLobby.save()
    firstPlayer = Player(lobby=newLobby, player=request.user)
    firstPlayer.save()
    lobby(request, newLobby.id)
    #return redirect('lobby', lobby_id=newLobby.id)
    return HttpResponseRedirect('/lobby/{}/'.format(newLobby.id))
def lobby(request, lobby_id):
    lobby = get_object_or_404(Lobby, pk=lobby_id)
    players= Player.objects.filter(lobby=lobby)
    activePlayersCount = Player.objects.filter(lobby=lobby, active=True).count()
```

Sl. 5.32. Metode iz datoteke „view.py“ koje dohvaćaju i spremaju podatke u bazu podataka

```
@database_sync_to_async
def getActiveUsers(self):
    players = Player.objects.filter(lobby=self.lobby_id, active=True)
    activePlayers=[]
    for player in players:
        activePlayers.append(User.objects.get(pk=player.player.id).username)
    return activePlayers
```

Sl. 5.43. Metode iz datoteke „consumers.py“ koja dohvaća aktivne igrače iz lobija

5.4. Korisničko sučelje

Dokumenti u kojima je potrebno prikazati rezultat nalaze se u mapi „templates“. Kod web aplikacija to su najčešće HTML datoteke. Django koristi standardni HTML, za logiku koriste se Django oznake(*eng. templates*). Već u početnoj stranici pomoću oznaka provjerava se ako je korisnik registriran, ako nije izbornik za kreiranje i pridruživanje lobiju se ne prikazuje, također korisniku se nudi mogućnost kreiranja privremenog računa, prijava ili registracija. Na početnoj stranici prikazane su nasumične igrače karte kako bi stranica privukla igrače umjesto da ih odbije preozbiljnim dizajnom. Prikaz nasumičnih karata postignut je tako da pomoću JavaScript programskog jezika generiraju nasumični brojevi koji predstavljaju karte koje se dodaju u HTML dokument.

```
{% if user.is_authenticated %}
    <a class="btn btn-dark" href="/lobby">Kreiraj lobi</a>
    <br><br><br><br>
    Unesi broj lobija kojem se želiš pridružiti<br>
    <input id="lobby-name-input" type="number" >
    <input class="btn btn-outline-dark" id="lobby-name-submit" type="button" value="Pridruži se"><br>
{% endif %}
```

Sl. 5.54. Prikaz HTML sadržaja samo prijavljenim korisnicim uz pomoć Django oznaka

```
for (let i = 0; i < 50; i++) {
    let x=Math.floor(Math.random() * 54);
    document.getElementById('cards').innerHTML+=document.getElementById(x).outerHTML;
}
```

Sl. 5.65. Metoda za odabir i prikaz nasumičnih karata na početnoj stranici

HTML dokument na kojemu korisnici igraju igru „Uno“ ima najviše koda vezanog za korisničko sučelje jer mora prikazati sve igrače i karte te konstantno ažurirati stanje istih za sve igrače. Nakon što se igrači prebace iz lobija u igru, svaki igrač će dobiti podatke o igračima i kartama. Te podatke uz pomoć JavaScripta se obrađuju i kreiraju HTML oznake s podatcima koje se ubacuju u HTML dokument.

```
function playerHTMLString(playerIndex) {
  let path="";
  let first="<div id='"+gameData.players[playerIndex]+"' class='col'>"+
            "<div class='border rounded' style='width: auto;'>"+
            "<div class=''>"+
            "<span class=''>"+
            "<img src='";
  let second = "' style='width: 1rem;' class='card-img-top' alt='...'><b>";
  let playerHTMLStringEnd = "</b><img id='img"+gameData.players[playerIndex]
  +"' hidden src='{% static 'game/back.svg' %}' style='width: 1rem;'></img>"
  +"<b hidden id='winner"+gameData.players[playerIndex]
  +"'>WINNER</b><p>Pobjede<span class='badge text-bg-success'>5</span>"
  +" Odigrano<span class='badge text-bg-secondary'>10</span></p></div></div></div>";
  if (playerIndex==0){
    path="{% static 'game/red.svg' %}"
  }
  else if (playerIndex==1){
    path="{% static 'game/yellow.svg' %}"
  }
  else if (playerIndex==2){
    path="{% static 'game/green.svg' %}"
  }
  else if (playerIndex==3){
    path="{% static 'game/blue.svg' %}"
  }
  else{
    path="{% static 'game/back.svg' %}"
  }
  return first+path+second+gameData.players[playerIndex]+playerHTMLStringEnd;
}
```

Sl. 5.76. Metoda kreiranje HTML oznaka s podatcima igrača

```
for (let i = 0; i < player.hand.length; i++) {
  document.getElementById('hand').innerHTML+='<div class="col"><button id='+i
  +' onclick="playCard(this.id)" class="btn">'
  +document.getElementById(player.hand[i].category+player.hand[i].number).outerHTM
  +'</button></div>';
}
```

Sl. 5.87. Metoda kreiranje HTML oznaka karata koje igrač ima u ruci

Pri kreiranju HTML oznaka igrača, pokraj imena svakog igrača se nalazi slika pozadine „Uno“ karte koje su skrivene, slika pokraj imena igrača koji je na potezu se otkriva. Osim informacije koji je korisnik trenutno na potezu, jako bitna informacija za igrače je koja je boja trenutno aktivna, pogotovo kada je zadnje odigrana posebna karta za promjenu boje. Ta se informacija prikazuje uz posljednje odigranu kartu.

```
function updateCurrentColor(newColor){
  document.getElementById('currentRed').hidden=true;
  document.getElementById('currentYellow').hidden=true;
  document.getElementById('currentGreen').hidden=true;
  document.getElementById('currentBlue').hidden=true;
  if(newColor=="R"){
    document.getElementById('currentRed').hidden=false;
  }
  else if(newColor=="Y"){
    document.getElementById('currentYellow').hidden=false;
  }
  else if(newColor=="G"){
    document.getElementById('currentGreen').hidden=false;
  }
  else if(newColor=="B"){
    document.getElementById('currentBlue').hidden=false;
  }
}
```

Sl. 5.98. Metoda kreiranje HTML oznaka karata koje igrač ima u ruci

U pravilima igre je definirano koju kartu možemo odigrati na posljednje odigranu kartu, kako se resursi ne bi trošili bespotrebno, pritisnuta karta će se prvo provjeriti metodom koja uspoređuje

kartu sa zadnjom odigranom te će dozvoliti slanje podataka o karti poslužitelju ako se karta može odigrati.

```
canPlay(card) {
    if(me !== currentGame.getCurrentPlayer()) {
        return false;
    }
    let category = card.category, number = card.number;
    let topCategory = this.topCard.category, topNumber = this.topCard.number;
    return category === this.topColor || number === topNumber || category === "W"||category === "WF";
}
```

Sl. 5.109. Metoda za provjeru može li se karta odigrati

5.5. Implementacija lobija

Kako bi meč mogao biti pokrenut potrebno je prethodno spojiti igrače, to se ostvaruje kreiranjem lobija kojem korisnici mogu pristupiti unosom koda, ili kreiranjem javnog lobija. Nakon što korisnik kreira lobi dobiti će kod koji će dati drugim korisnicima kako bi pristupili lobiju. Kreiranje lobija je komunikacija u jednom smjeru, ali da bi ostali korisnici pristupili i vidjeli ostale igrače koji su prisutni i da administrator lobija može pokrenuti meč potrebna je asinkrona komunikacija. Metoda za kreiranje lobija nalazi se u datoteci „views.py“ i već je prikazana slikom 5.112, ta metoda omogućava korisniku obradu zahtjeva i prikaz HTML dokumenta. Unutar dokumenta se nalazi JavaScript kod koji otvara komunikaciju prema poslužitelju uz pomoć web utora, poslužitelj prihvaća komunikaciju pomoću metode „connect“ koja se nalazi u datoteci „consumers.py“, unutar te datoteke se također nalaze metode za postupanje za u slučaju primitka poruke i napuštanja komunikacije.

```
async def connect(self):
    self.lobby_id = self.scope["url_route"]["kwargs"]["lobby_id"]
    self.room_group_name = f"lobby_{self.lobby_id}"
    user = self.scope['user']
    self.lobby = GameLobby.create_new_lobby(lobby_id=self.lobby_id, player=user)
    await self.channel_layer.group_add(self.room_group_name, self.channel_name)
    await self.accept()
```

Sl. 5.20. Asinkrona metoda za kreiranje lobija i spajanje korisnika ukoliko lobi postoji

Korisnik nakon uspostave komunikacije odmah šalje poslužitelju svoje korisničko ime, metoda za primanje poruka tada svim članovima te grupe šalje poruku u „JSON“ obliku s podatcima o lobiju kao i svim članovima grupe.

Linija ***Kod***

```
        {'type': 'lobby.message', 'gameLobby': {'lobbyId': '692', 'players':
1:      ['Guest 71', 'Guest 72'], 'adminUsername': 'Guest 71'}}
```

Sl. 5.121. Podatci o grupi u JSON obliku nakon pridruženja korisnika „Guest 72“

HTML dokument se osvježava nakon svake poruke o novom korisniku, ili u slučaju prekida veze uklanja korisnika koji je napustio grupu. Kada je u grupi dovoljno korisnika za pokretanje igre, administrator lobija unutar svog HTML dokumenta dobiva opciju pokretanja igre tako da tipka za pokretanje igre postane aktivira.

5.6. Implementacija igre „Uno“

Kada je igra pokrenuta korisnici se preusmjeravaju na novu stranicu na kojoj se uspostavlja nova asinkrona komunikacija uz pomoć web utora. Kako bi igra sa svim korisnicima iz lobija započela u bazu podataka se sprema broj korisnika u lobiju, iako se stranica učitala igra nije, kada posljednji korisnik ostvari konekciju, poslužitelju se šalje poruka da može pokrenuti igru.

```
if (data.type == 'game.join'){
  console.log(e);
  numberOfPlayers++;
  if (numberOfPlayers == numberOfPlayersFromLobby){
    let data = {"status": "start_game", "message": "Game is starting.",
    "data": {"username": '{{user.username}}' }};
    let response = {"type": "start.game", "text": data};
    webSocket.send(JSON.stringify(response));
  }
}
```

Sl. 5.22. Kod koji pokreće igru kada su svi korisnici prisutni

Kada poslužitelj dobije poruku za početak počinje priprema za igru, prvi korak je miješanje karata, drugi korak je dijeljenje karata korisnicima, treći korak odabir prve karte sa špila koja će služiti za prvi kartu, nakon ta tri koraka poslužitelj šalje podatke korisnicima.

Linija Kod

```
1:      "type": "websocket.send",
2:      "text": {
3:          "status": "start_game",
4:          "message": "Game is starting.",
5:          "data": {
6:              "username": "Guest 70"
7:          },
8:          "gameData": "{\"lobbyId\": \"687\",
9:                \"players\": [\"Guest 70\", \"Guest 62\"],
10:               \"topCard\": {\"category\": \"R\", \"number\": 8},
11:               \"topColor\": \"R\",
12:               \"direction\": \"+\",
13:               \"currentPlayerIndex\": 0,
14:               \"adminUsername\": \"Guest 70\"}",
15:         "serializedPlayer": "{\"username\": \"Guest 62\",
16:                   \"hand\": [{\"category\": \"Y\", \"number\": 6},
17:                     {\"category\": \"Y\", \"number\": 4},
18:                     {\"category\": \"Y\", \"number\": 10},
19:                     {\"category\": \"WF\", \"number\": 13},
20:                     {\"category\": \"R\", \"number\": 12},
21:                     {\"category\": \"R\", \"number\": 3},
22:                     {\"category\": \"G\", \"number\": 4}]}"
```

Sl. 5.133. Podatci koje je primio korisnik Guest 62 za početak igre

Kada korisnik primi podatke korisnicima se prikazuje prva karta, označen je korisnik je na redu i svaki igrač ima prikazane svoje karte. Pri prvom potezu izgledno je da će korisnik imati kartu koju može odigrati pa slijedi opis postupak igranja karte. Kada korisnik klikne kartu koja se može odigrati šalje se poruka poslužitelju da će se odigrati karta, tko igra kartu, koju kartu i ukoliko je posebna karta promjene boje, koja će biti sljedeća boja. Poslužitelj će tada još jednom provjeriti je li taj korisnik na redu i može li odigrati tu kartu, ukoliko poruka prođe provjeru, odigrana karta se postavlja kao posljednja odigrana karta, te slijedeći korisnik dolazi na red. Poslužitelj će također provjeriti ako korisnik više nema karata u ruci te će poslati poruku da je pobijedio.

```
def is_valid_play_move(self, client_data, server_data):
    client_card, client_card_index = client_data['card'], client_data['index']
    client_username, client_next_top_color = client_data['username'], client_data['next_top_color']

    server_username = server_data['username']

    client_card_obj = Card(client_card['category'], client_card['number'])

    server_current_player = self.get_current_player()
    if server_current_player.username != client_username:
        return False

    if client_username != server_username:
        return False

    if not server_current_player.is_card_in_hand(card=client_card_obj, card_index=client_card_index):
        return False

    can_play = self.can_play_over_top(player=server_current_player, card=client_card_obj)
    if can_play is False:
        return False
    return True
```

Sl. 5.24. Metoda za provjeru ako je potez moguć

Ako korisnik nema kartu koju može odigrati povući će kartu tako da pritisne špil. Nakon pritiska špila poslati će se poruka u kojoj se nalazi ime korisnika i da dobrovoljno izvlači kartu. Poslužitelj će provjeriti ako je korisnik koji izvlači kartu na redu, provjeriti može li se karta odigrati i ako može dopustiti korisniku igranje te karte te će poslati korisnicima poruku o izvlačenju karte no samo će korisnik koji izvlači moći vidjeti kartu.

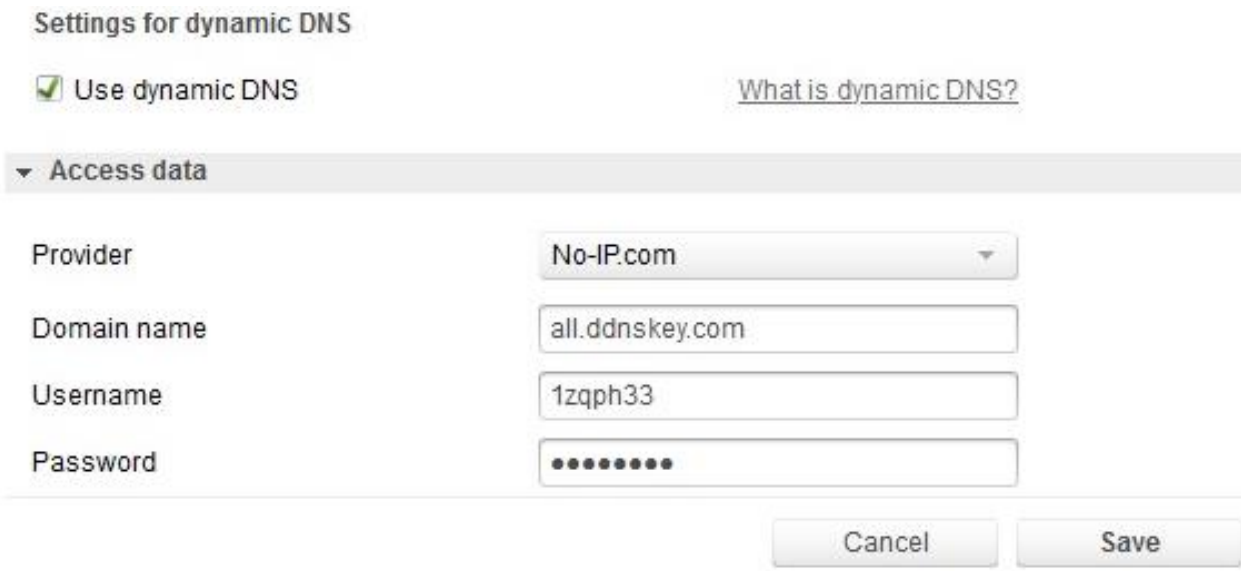
```
def draw_card(self):
    server_current_player = self.get_current_player()
    drawn_card = server_current_player.draw(self.deck)

    if not self.can_play_over_top(player=server_current_player, card=drawn_card):
        self.previous_player_index = self.current_player_index
        self.current_player_index = self.increment_or_decrement_current_player_index(amount=1
    response = {
        "username": server_current_player.username,
        "drawnCard": json.dumps(drawn_card, cls=CustomEncoder),
    }
    return response
```

Sl. 5.25. Metoda izvlačenje karte

5.7. Postavljanje DDNS-a

Za korištenje DDNS potrebno je odlučiti se za pružatelja usluge i kreirati račun. Također modem za pristup internetu treba imati podršku za DDNS, te se provjerava koji DDNS pružatelji usluga su podržani na tom modemu. Jedan od DDNS pružatelja koji je podržan i koji se koristi za ovaj

rad je „No-IP.com“. Nakon kreiranja računa i domene dobiveni podatci se unose u modem. Osim podataka za DDNS potrebno je postaviti pravilo za usmjeravanje portova kako bi modem mogao usmjeriti promet prema poslužitelju.



Sl. 5.27. Podatci za DDNS uneseni u modem.

6. Zaključak

Početna ideja ovog završnog rada je bila omogućiti prijateljima koji su udaljeni ili nemaju karte da mogu zaigrati igru „Uno“ uz pomoć web preglednika i pristupu internetu. Proučavanjem dostupnih rješenja htio sam da se moja web aplikacija prepoznatljiva po jednostavnom pristupu, samo jednim klikom osoba može kreirati račun i pridružiti se igranju s prijateljima. Za tehnologije koje sam odabrao odlučio sam se jer sam oh već koristio pa samo mogao primijeniti, ali i proširiti postojeće znanje. „Django“ programski okvir je uvelike pojednostavnio rad s autentifikacijom i omogućio brzu implementaciju kreiranja računa jednim klikom. Zbog načina na koji igra funkcionira morao sam se i upoznati asinkronom komunikacijom preko web utora koja je omogućena aplikacijom za „Django“ od nazivom „Channels“. Korištenje nove tehnologije teklo je od kreiranja jednostavne komunikacije gdje se korisnici pridružuju virtualnom lobiju te igraju igru kamen, papir, škare do kompliciranije komunikacije koja omogućava igranje zabavne igre kao „Uno“.

Premda postoje mnoge alternative za igranje igre „Uno“ pri izradio ovog rada iskoristio sam mnoga znanja, ali isto tako sam stekao nova, od rada s korisničkim sučeljem, bazom podataka, poslužiteljem, pa na kraju omogućiti pristup web aplikaciji s interneta. To su znanja koja ću cijeniti, ali zacijelo i ponovno iskoristiti.

LITERATURA

[1] „Django documentation“, dostupno na: https://docs.djangoproject.com/en/5.0/ [30.6.2024]

[2] „Django Channels”, dostupno na : https://channels.readthedocs.io/en/latest/index.html [30.6.2024]

[3] „W3Schools”, dostupno na : https://www.w3schools.com/ [30.6.2024]

[4] „Four Colors”, dostupno na : https://www.cbc.ca/kids/games/play/four-colours [30.6.2024]

[5] „UNO | Ubisoft (US)”, dostupno na : https://www.ubisoft.com/en-us/game/uno/uno [30.6.2024]

[6] „MDN Web Docs - Mozilla”, dostupno na : https://developer.mozilla.org/en-US/ [30.6.2024]

[7] „venv — Creation of virtual environments”, dostupno na : https://docs.python.org/3/library/venv.html [30.6.2024]

[8] „Bootstrap · The most popular HTML, CSS, and JS library in the world.”, dostupno na : https://getbootstrap.com/ [30.6.2024]

[9] „What Is SQLite?”, dostupno na : https://www.sqlite.org/index.html [30.6.2024]

[10] „Docker: Accelerated Container Application Development”, dostupno na : https://www.docker.com/ [30.6.2024]

[11] „Redis - The Real-time Data Platform”, dostupno na : https://redis.io/ [30.6.2024]

[12] „Dynamic DNS | Cloudflare”, dostupno na : https://www.cloudflare.com/en-gb/learning/dns/glossary/dynamic-dns/ [30.6.2024]

[13] „A complete Uno cards deck”, dostupno na : https://commons.wikimedia.org/wiki/File:UNO_cards_deck.svg [30.6.2024]

SAŽETAK

Naslov: Web za igru "Uno"

Web aplikacija „Uno“ omogućuje korisnicima igranje popularne kartaške igre, jednim klikom korisnik kreira privremeni račun te u tren oka može pokrenuti partiju s prijateljima. Nakon prijave korisnik odmah može kreirati virtualni lobi i pozvati prijatelje dijeljenjem numeričkog koda kojim oni pridružuju igri. Korisničko sučelje aplikacije realizirano je pomoću JavaScript biblioteke Bootstrap koja osim što je jednostavna za korištenje, omogućava kvalitetnu izvedbu responzivnosti. Za poslužiteljsku stranu zadužen Django uz ugrađene aplikacije omogućava brzu i kvalitetnu izvedbu dijela aplikacije kao što je autentifikacija. Asinkrona komunikacija web utorima između klijenta i poslužitelja omogućena je korištenjem Channels-a i Redis.

Ključne riječi: Asinkrona komunikacija, Igra, Privremeni račun, Responzivno, Web aplikacija

ABSTRACT

Title: Web application „Uno“

The web application "Uno" allows users to play the popular card game. With a single click, a user creates a temporary account and can start a game with friends in no time. After logging in, the user can immediately create a virtual lobby and invite friends by sharing a numerical code that allows them to join the game. The application's user interface is implemented using the JavaScript library Bootstrap, which, besides being easy to use, allows for high-quality responsive design. The server-side is handled by Django, which, along with built-in applications, enables fast and quality implementation of parts of the application such as authentication. Asynchronous communication between the client and server through web sockets is enabled using Channels and Redis.

Keywords:

Ključne riječi: Asynchronous communication, Game, Responsive, Temporary account, Web application