# OOAD Projekat - Playoff

**Nastavna grupa:** RI2 – Grupa 9
**Naziv projekta:** Playoff
**Naziv tima:** C--

**Članovi tima:**


- Mujić Sulejman (17873)
- Muminović Amir (17744)
- Salman Haris (17822)


## 1. Opis aplikacije

**Playoff** je aplikacija namijenjena sportistima i rekreativcima, te trenutno podržava pet sportova: fudbal, košarka, odbojka, tenis i trčanje. Prilikom organizovanja prijateljskih susreta postoji mnogo problema koje nemaju sistematično rješenje. Često timovi igraju protiv istih timova i ne mogu naći nove adekvatne protivnike. Nekim timovima fali igrača dok također postoje igrači koji ne mogu naći tim. Koristeći Playoff, korisniku je omogućeno jednostavno pronalaženje/formiranje tima i suigrača na njegovom nivou vještine, kao i organizovanje utakmica sa adekvatnim protivnicima i pregled ostvarenih rezultata u prošlim mečevima na svom profilu.

Algoritam za pronalaženje protivnika koristi kombinaciju informacija unesenih od strane korisnika te ostvarenih rezultata u prijašnjim mečevima da formira relativno preciznu sliku o sportskim sposobnostima svakog korisnika i upari ga sa drugim igračima istog nivoa, eliminirajući standardnu zamisao najboljih i najslabijih igrača.

## 2. GUI, raspodjela i opis formi

Od samog početka rada na aplikaciji, cilj tima je bio da maksimalno pojednostavi građu klasa, a samim tim i korisničkog interfejsa. Tokom rada smo osigurali da svaka forma ima samo jednu specifičnu ulogu. U nastavku slijedi opis postojećih formi.

Pri samom pokretanju aplikacije se prikazuje forma **MainPage**, koja služi za unos korisničkih informacija za pristup njihovom računu, a samim tim i ostatku aplikacije. Alternativno, novi korisnici mogu kreirati novi račun na formi **Registration**, kojoj se može pristupiti sa forme **MainPage**. U bilo kojem trenutku, korisnik može obustaviti kreaciju novog računa i vratiti se na ekran za login koristeći **button** označen kao „Cancel.“

Nakon što se korisnik uspješno prijavi na svoj račun, prikazuje se forma **MainMenu**, koja prikazuje glavni meni aplikacije i omogućava pristup svim ostalim dijelovima aplikacije. Pored standardnog pristupa formama preko klika/dodira, također je omogućen pristup koristeći *voice recognition*. Pošto je planiran prevod aplikacije na Engleski jezik, voice recognition radi kada se nazivi menija izgovore na njemu (profile, teams, tournaments, search, notifications).

Forma **Profile** služi za predstavljanje profila korisnika. Tu korisnik vidi svoje osnovne informacije, svoj ranking i susrete u kojima je učestvovao.

Pregled timova u kojima je korisnik je dat na formi Teams. Ovdje korisnik također može kreirati novi tim tako što ide dalje na formu **CreateTeam**, ili da uredi postojeći team preko forme **ManageTeam**.

Forma **Tournaments** služi za pregled aktuelnih turnira, te kreiranje novih i prijavu na turnire.

Forma **Search** se koristi za pretragu timova ili igrača. Ovo je korisno za pronalaženje novih mogućih suigrača ili eventualnih protivničkih timova za organizovanje susreta.

Na formi **Notifications** korisnik može pregledati notifikacije i razne vrste zahtjeva.

## 3. Klase, njihove strukture i namjene

Klase korištene u kreaciji ovog projekata mogu se podijeliti u tri grupe:

- Komunikacija
- Mečevi
- Ostale klase

### 1) Komunikacija

Prvoj grupi pripadaju klase **Poruka**, **Zahtjev** i **Review**. **Poruka** je osnovno sredstvo komuniciranja dva ili više korisnika. Sastoji se od identifikacionog broja - ID, sadržaja, pošiljaoca, primaoca te **bool** vrijednosti viđenost. Identifikacioni broj je varijabla tipa **int** koja jednoznačno određuje poruku. *Pošiljaoc*, *primaoc* i *sadržaj* su tipa **string** te imaju funkcije očite iz imenovanja varijabli. Varijabla *viđenost* čuva informaciju da li je korisnik već vidio poruku ili ne. Metode za implementaciju opisanih funkcionalnosti su manje-više trivijalni. Svi transferi poruka se zasnivaju na činjenici da svaki korisnik ima svoj ‘mailbox’ te slanjem poruke samo dodamo novi objekat tipa **Poruka** u kolekciju.

Specijalizaciju klase poruka čini klasa **Zahtjev**. Pored već opisanih atributa, sadrži informaciju o timu. Ova klasa se koristi za modeliranje poziva za ulazak u timove. Metode ove klase omogućavaju korisnicima da prihvate/odbiju//pošalju zahtjev korisniku ili listi korisnika.

Klasa **Review** se sastoji od tri atributa. To su *komentar*, *ocjena* i objekat tipa **Tim**. *Komentar* je tipa **string** i omogućava korisnicima da ostave utisak na prijateljski susret sa suparničkim timom. Također imaju mogućnost da ocijene tim. Ocjena je implementirana kao varijabla tipa **int** ali u .NET implementaciji je dodato **DataAnnotation** ograničenje kako bi prihvatala samo cjelobrojne vrijednosti iz segmenta [1,5]. *Tim* je objekat koji čuva informaciju na koji tim se dati **Review** odnosi. U .NET implementaciji varijabla *tim* je promijenjena u **int** radi pametnije implementacije u cilju ostvarivanja boljeg povezivanja sa bazom podataka (ID tima). Također je dodan atribut *ID* tipa **int** kako bi jednoznačno označavao recenzije.

**2) Mečevi**

Druga grupa sadrži apstraktnu klasu mečevi te sve njene specijalizacije (**Tenis**, **Nogomet**, **Odbojka**, **Trčanje**, **Košarka**). Ova grupa klasa ima veliki značaj jer modelira ključnu problematiku naše teme – sportske mečeve. Apstraktna klasa **Meč** ima parametar tipa **int** koji jednoznačno određuje mečeve. Prilikom analize klase i kreacije odgovarajućih dijagrama klasa uočeni su brojni zajednički elementi za sve klase iz ove grupe. To su: vrijeme održavanja meča – tipa **DateTime** i mjesto održavanja – tipa **string**. Prvenstveno je podržana mogućnost postojanja više timova po meču ali je uočeno da je za odabrane sportove dovoljno uzeti dva tima po meču. Shodno tome imamo dvije varijable tipa **int** koji čuvaju *ID* timova koji igraju. Kako bi meč bio obrađen, neophodno je da kapiteni oba tima ostave recenziju odigranog meča te unesu isti rezultat. Inače nastaje komplikacija. Da bismo imali pregled koji su mečevi usaglašeni – registrovani, a koji nisu, koristimo **bool** varijablu *registrovan*. Implementirane metode obavljaju upravo taj posao te određuju ko je bio pobjednik na osnovu unesenih rezultata. Dodatni atributi naslijeđenim klasama su dodani tako da najbolje modeliraju dati sportski meč. Svaka izvedena klasa mora da implementira metodu ***odrediPobjednika*** – tj. da ima vlastiti mehanizam određivanja pobjednika oviseći od unesenih podataka.

**Trčanje** je najtrivijalnije. Sastoji se od **tuple<double>** i DateTime. Za trčanje važna je samo distanca – koju mjerimo sa **double** i vrijeme trčanja – koje modeliramo sa **DateTime**. Nema pozicija u trčanju pa nije bilo neophodno modelirati pozicije učesnika.

Klasa **Tenis** ima dva **enum**-a koji modeliraju broj setova u tenisu te rezultat. Algoritam za određivanje pobjednika je zasnovan na tim informacijama i u potpunosti podržava unos podataka u toj formi. Za modeliranje rezultata koristi se trodimenzionalni objekat – lista listi objekata tipa **tuple** koji sadrže dvije vrijednosti tipa **enum Gameoutcome**. Broj setova određuje veličinu ovih listi. U tenisu ako igraju četiri osobe – doubles, igrači se naizmjenično mijenjaju za poziciju naprijed-nazad, pa nije potrebno posebno modeliranje.

Klasa **Odbojka** je slična klasi **Tenis**. Također ima poseban broj setova – ali samo dvije mogućnosti, 3 ili 5 a ne 1, 3, 5 kao sa tenisom. Rezultat modeliramo u listi listu tipa **tuple** od dva **int**-a. Oviseći od broja setova kreiramo listu odgovarajuće veličine. Pozicije u odbojci se rotiraju pa nije potrebno unositi pozicije.

Klase **Nogomet** i **Košarka** zahtijevaju modeliranje pozicija. Zbog toga su pored svake klase dodani enumi koji ih modeliraju. Za nogomet: *Golman, Odbrana, Veza i Napad*, a za košarku: *Center, Shooting_Guard, Power_Forward, Point_Guard, Small_Forward*. Rezultate u oba slučaja čuvamo u listama parova dva **int**-a – jedan za modeliranje igrača koji je dao gol, a drugi za broj bodova koje je dodao timu. U slučaju nogometa to je uvijek 1 dok u košarci može biti trica ili dvica. Pored toga imamo listu **tuple**-ova **Korisnik** i pozicije koje modeliraju pozicije na kojima korisnici igraju. Odgovarajuće metode omogućavaju unos rezultata te određivanje pobjednika.

### 3) Ostale klase

Treću grupu čine klase koje se nisu mogle svrstati u prve dvije. Ovoj grupi pripadaju klase **Tim**, **Sport**, **Korisnik**, **Baza**. Klasa **Korisnik** modelira korisnika sistema te sadrži sve neophodne informacije za njega. To podrazumjeva *ID* tipa **int** za već opisanu funkcionalnost, informacije o imenu, prezimenu, korisničkom imenu, lozinci, državi, gradu, datuma rođenja (ime ovih atributa opisuje njihovu funkciju). Dalje imamo Boolean atribut koji prikazuje dostupnost, odnosno da li korisnik želi da se pojavljuje u rezultatima pretrage kapitena kada traže igrače. Također imamo liste timova, poruka, prošlih i budućih mečeva koji prestavljaju kolekcije tih objekata. Posebno imamo i listu para vrijednosti Tim i DateTime, koji čuvaju informacije o prošlim timovima i datima odlaska. Lozinka je tipa MD5 Hash u svrhu kreiranja sigurnosnog okruženja za korisnika. Metode ove klase su bazirane na osnovnu funkcionalnost kao što je otkazivanje mečeva.

Klasa sport je napravljena kako bi čuvala najvažnije informacije o sportovima – njihovo ime, minimalan i maksimalan broj igrača. Njenom primjenom olakšan je rad sa mečevima jer je lakše modelirati ograničenja koja broji sportovi imaju. Klasa tim služi za modeliranje najvažnije kolekcije u aplikaciji. Svaki tip ima ime – za koji koristimo string, vlastiti indentifikacioni broj – cjelobrojni ID, broj pobjeda, gubitaka, nerješenih mečeva, specifični sport, listu prošlih i budućih susreta, članova te recenzija koji određuju rating tima. Također svaki tim ima kapitena koji je odgovoran za regulisanje tima, ugovaranje mečeva... Finalno imamo klasu Baza koja se koristi za rad sa bazom podataka te koja sadrži osnovne metode za rad.

## 4. Baza podataka i njena građa

Baza Podataka je kreirana po principu Database first, jer je SQL (u našem slučaju Transact SQL) jezik namjenjen za komuniciranje sa bazom podataka pa je samim time i efikasnije, u .sql fajlovima se nalazi kreiranje tabela, procedura , funkcija i sekvenci.

Da bi se komuniciralo sa bazom podataka kreiran je korisnik koji ima samo permisije da poziva funkcije (kako bi dobio podatke) i procedure (kako bi popunjavao tabele). ERD je dostupan svakom da vidi izgleda tabela(i veza između tabela) na Bazi podataka.
Također su dodana ograničenja kao što je primary key, foreign key, unique itd. Dalje su dodani određeni indexi na pojedine kolone kako bi se ubrzao pristup (predviđeno je da se tim kolonama često pristupa tokom aplikacije, kao što je npr. password korisnika)
Da bi aplikacija komunicirala sa bazom podataka napisane su funkcije u C# koje koriste procedure/funkcije na bazi podataka da upisuju/dobavljaju podatke sa baze podataka.
Također je omogućena komunikacija sa bazom podataka preko Web API-ija koja je nešto sporija nego direktna.

## 5. Uloga .NET i njegova implementacija