**Arhitekturni dizajn softvera**

**FootballCommunity**

Lazar Stoiljković 16381

Nikola Gocić 16039

Sadržaj

**Kontekst i cilj projekta3**

**Arhitekturni zahtevi3**

Glavni funkcionalni zahtevi3

Atributi kvaliteta5

Tehnička ograničenja5

**Arhitekturni dizajn6**

Arhitekturni obrasci6

Strukturni pogled6

Bihevioralni pogled7

Implementaciona pitanja8

**Kontekst i cilj projekta**

FootballCommunity aplikacija ima za cilj da obezbedi live praćenje fudbalskih rezultata i interakciju i zabavu između korisnika u toku trajanja neke utakmice (*chat*, *live* predikcija i pogađanje rezultata i takmičenje između korisnika). Projekat će biti realizovan kao *client-server web* aplikacija.

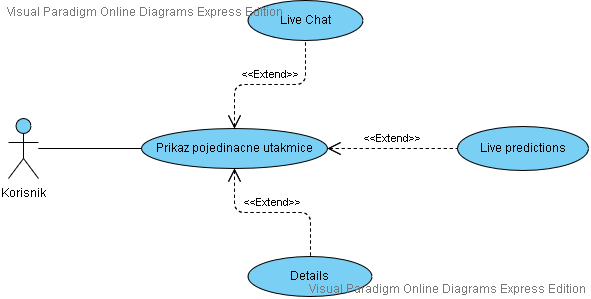
**Arhitekturni zahtevi**

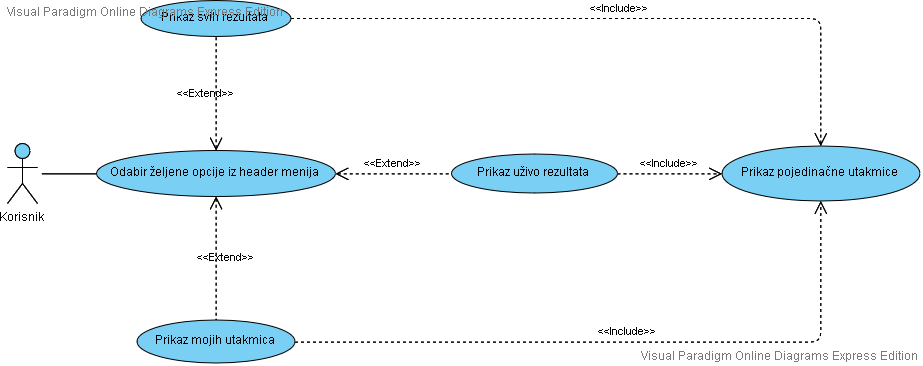
**Glavni funkcionalni zahtevi**

1. **Registracija korisničkog naloga** – kreiranje naloga za korisnika koji želi da koristi aplikaciju je neophodno. Na početnoj strani postoji opcija za kreiranje naloga pri čemu je neophodno odabrati korisničko ime koje nije zauzeto, odabrati lozinku, uneti email (postoji mogućnost registrovanja pomoću *google* naloga).
2. **Prijavljivanje na korisnički nalog** – na poćetnoj strani pored opcije za kreiranje naloga nalaziće se opcija i za logovanje na već postojeći nalog pomoću korisničkog imena i ispravne lozinke.
3. **Odabir željene opcije iz *header* menija** – korisnik može da izabere prikaz određene stavke (*results* , *live results*, *my games*, *profile*).
4. **Prikaz svih rezultata** – prikaz svih utakmica koje se igraju izabranog datuma (podrazumeva utakimice koje su se završile tog dana, utakmice koje se trenutno odigravaju i utakmice koje tek treba da se odigraju). Takođe korisnik će imati mogućnost da izabere datum za koji želi da vidi utakmice, 7 dana pre i 7 dana posle trenutnog datuma. Korisnik ima mogućnost *check-* iranja određene utakmice i na taj način dodaje tu utakmicu u *˝My Games.*
5. **Prikaz uživo rezultata** – deo gde će se prikazivati samo utakmice koje se trenutno igraju *˝Live results˝.*
6. **Prikaz mojih utakmica** – deo gde će se prikazivati samo utakmice za koje je korisnik zainteresovan (*subscribe-*ova se za određenu utakmicu). Postojaće mogućnost da prilikom promene rezultata na određenoj utakmici se obaveste svi korisnici kojima je ta utakmica u ¨*My Games*¨ (vizuelna ili opciono zvučna signalizacija ako je korisnik *online*).
7. **Prikaz profila** – prikaz osnovnih podataka o korisniku. Deo u kojem će korisnik imati mogućnost da izabere neku utakmicu koja još uvek nije počela i da izvrši predikciju za utakmicu. Na profilu će se prikazivati sve predikcije za datog korisnika, kao i statistički parametri u vidu procenta uspešnosti u pogađanju.
8. **Prikaz pojedinačne(konkretne) utakmice** – klikom na određenu utakmicu korisnik se preusmerava na novi prozor za prikaz odabrane utakmice. U ovom prozoru imamo prikaz timova i trenutnog rezultata kao i još jedan meni koji sadrži sledeće opcije: *Live chat, Live predictions, Details.*
9. **Interakcija između korisnika vezana za konkretnu utakmicu** – korisnici će imati mogućnost da u toku utakmice u realnom vremenu ostavljaju komentare, vrše određene predikcije (rezultati će se grafički prikazivati i menjati u realnom vremenu kada neko od korisnika izvrši glasanje odnosno predikciju). Predikcije u okviru podmenija *Live predictions* će se prikazivati kao statistički grafikoni koji predstavljaju procenat korisnika koji su glasali za određenu opciju (npr. Pobednik domaćin, pobednik gost, nerešeno).

Bitniji slučajevi korišćenja:







**Atributi kvaliteta**

1. **Performanse –** Aplikacija bi trebalo da obezbedi što manje vreme odziva. Takođe vreme koje prođe od stvarne promene rezultata na nekoj utkmici do prikaza te promene korisniku bi trebalo da bude manje od 120 sekundi. Aplikacija treba da obezbedi najbolje performanse u zavisnosti od trenutnog broja online korisnika (performanse sistema se odnose na vreme odziva kao i na trenutnu validnost podataka – teži se ka tome što više podataka bude *up-to-date*).
2. **Upotrebljivost –** Klijentska strana aplikacije treba da se izvršava u okviru *Web browsera*-a. Korisnički interfejs treba da bude jednostavan i lak za korišćenje.
3. **Dostupnost –** Aplikacija bi trebalo da bude dostupna korisnicima 24x7x365, sa što je moguće većim procentom dostupnošću.
4. **Skalabilnost –** Aplikacija bi trebalo da podrži veliki broj istovremenih korisnika. Takođe treba obezbediti mehanizam za pribavljanje rezultata sa postojećeg *api*-ja (*rapidApi*) u zavisnosti od trenutnog broja *online* korisnika.

**Tehnička ograničenja –** aplikacija zahteva ograničenja u vidu broja zahteva upućenih rapidApi-ju (maksimalan broj zahteva na dnevnom nivou je 75 000). Potrebno je da serverska strana redukuje broj klijentskih zahteva na maksimalno 75 000 zahteva i slučaju da je broj klijentskih zahteva veći.

**Arhitekturni dizajn**

**Arhitekturni obrasci**

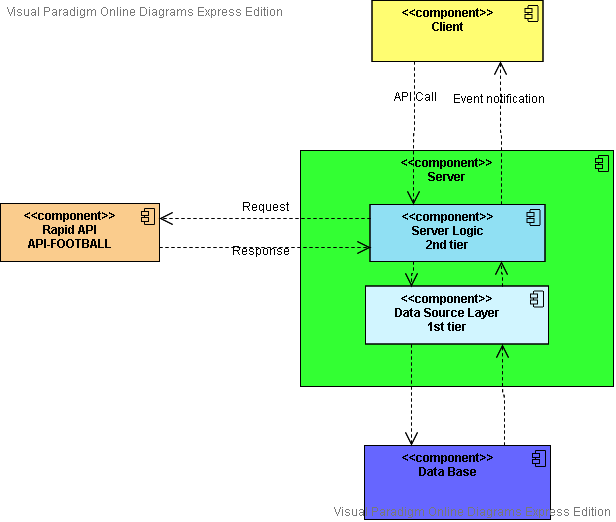
Arhitekturni obrasci koji će biti iskorišćeni prilikom realizacije aplikacije su:

1. **3-slojni *Client-Server* arhitekturni obrazac**

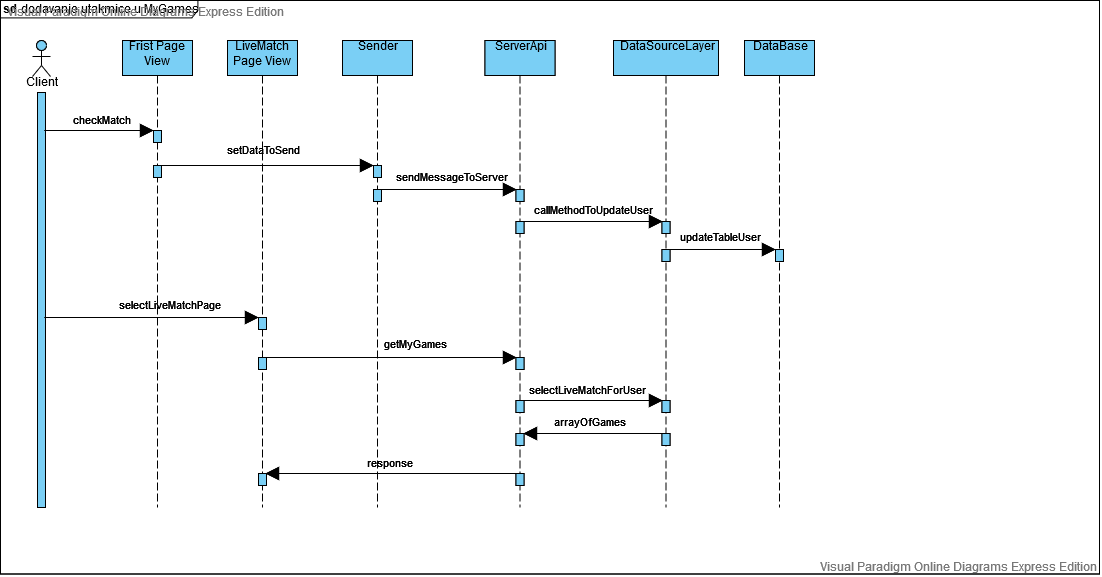
* najviši sloj su klijenti odnosno korisnici koji komuniciraju sa srednjim slojem na serverskom delu putem *RESTful* *Api-*ja.
* srednji sloj treba da obezbedi aplikacionu logiku kao i komunikaciju sa *rapidApi-*jem za preuzimanje *live* rezultata. Takođe potrebno je da obezbedi asinhroni mehanizam slanja notifikacija klijentima, kao i *real-time* komunikaciju između klijenata korišćenjem odgovarajuće biblioteke za to.
* najniži sloj je potrebno da obezbedi logiku za komunikaciju za bazom podataka – implementira *Entity Framework* kao alat za perzistenciju podatak iz relacione baze podataka.

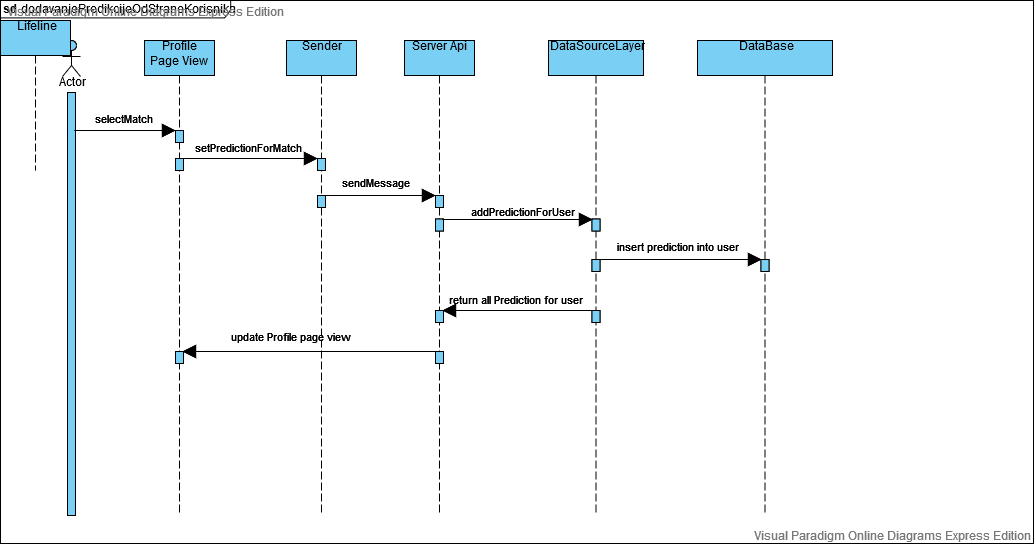
1. **Publish-Subscribe arhitekturni obrazac** – srednji sloj aplikacije treba da implementira ovaj obrazac kako bi se obezbedilo asinhrono obaveštavanje klijenata od strane servera (slanje notifikacija i drugih poruka i podataka). Biće korišćena biblioteka za asinhronu komunikaciju u realnom vremenu.

**Strukturni pogled**



**Bihevioralni pogled**





**Implementaciona pitanja**

Za realizaciju aplikacije koristiće se *.Net Framework.* Klijent i server komuniciraju preko  *REST API-*ja. Za asinhronu *real time* komunikacija između servera i više klijenata koristiće se *SignalR* biblioteka. Za realizaciju klijentske strane odnosno korisničkog interfejsa koristiće se *ReactJS* biblioteka. Za perzistenciju podataka biće iskorišćen *Entity Framework.* Koristiće se *Microsoft SQL server* kao *DBMS* za rad sa bazom podataka.