Elektronski fakultet u Nišu

Arhitektura i projektovanje softvera
-I faza-

Miljan Dragović 19105
Jakov Jakovljević 19137

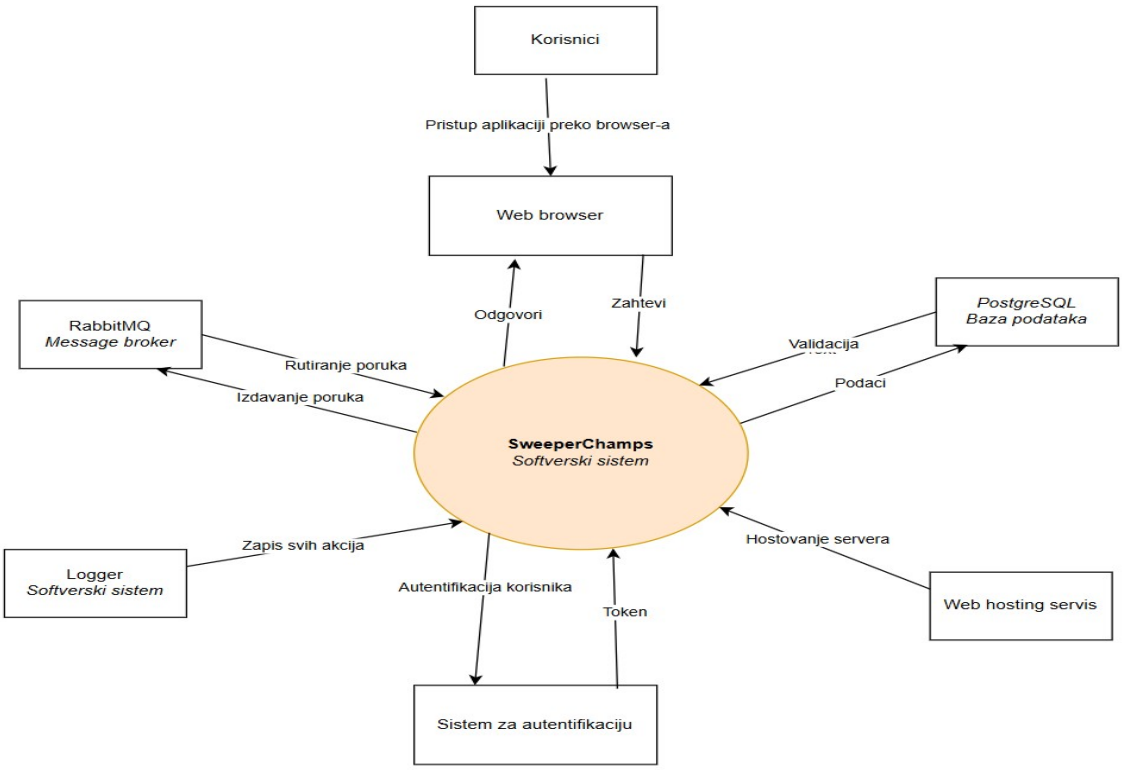
Sadržaj

1. Kontekst i cilj softverskog projekta

SweeperChamps je višekorisnička online web aplikacija inspirisana dobro poznatom igricom *Minesweeper,* uz nekoliko spinova. SweeperChamps mogu igrati više igrača naizmenično jedan protiv drugog, gde je glavni cilj skupljanje poena pod vremenskim pritiskom. Poeni se skupljaju tako što se pogodi tačna lokacije mine. Igrači mogu kompromitovati svoje protivnike postavljajuci lažne mine, zastavice...

Kontekst SweeperChamps je sačinjen od raznih entiteta/sistema koji interaguju sa njim. Kontekst je sačinjen od:

1. Korisnika - njima je namenjena aplikacija i oni interaguju sa njom, preko ***web browser*** - a na računaru ili telefonu.
2. Baze podataka - njena uloga je da čuva sve podatke softverskog sistema, kao što su: potezi, stanje i učesnici u meču, sve odigrane mečeve itd.
3. Web hosting servis - mesto na kome će da se nalazi serverski deo aplikacije.
4. Sistem za autentifikaciju - svaki korisnik mora da se registruje/uloguje i da prođe autentifikaciju. Time se omogućava verifikacija svakog korisnika koji koristi aplikaciju.
5. Logger - logger zapisuje sva dešavanja koja se dešavaju unutar softverskog sistema.
6. Message broker - omogućava asinhrono rutiranje poruka između različitih delova softverskog sistema



Slika 1. Konteksni dijagram

Cilj je web aplikacija kojoj korisnici mogu da pristupe bilo kada, bile gde, sa računara ili sa mobilnog uređaja. Korisnici će moći brzo i lako, kroz par klikova da kreiraju i zatim odigraju meč SweeperChamps-a. Uspešna aplikacija bi bila ona koja ima server koji je dostupan 24/7, brzo uparivanje udaljenih igrača i real-time mečeve sa minimalnom latencijom. Uspešna aplikacija je istovremeno održiva i skalabilna.

2. Arhitekturno specifični zahtevi

Konkretni funkcionalni zahtevi su:

- Registracija, logovanje i autentifikacija korisnika
- Jedinstvena identifikacija svakog korisnika
- Izmena korisničkog imena/šifre/email adrese
- Brisanje naloga
- Biranje jednog od predefinisanih tipova meča
- Pronalaženje adekvatnog protivnika
- Spajanje dva ili više kompatibilna igrača
- Napuštanje/odustajanje od meča
- Igranje poteza i njegova validacija
- Sinhronizacija stanja igre između svih igrača
- Provera završetka meča
- Prekid meča nakon gubitka konekcije
- Prikaz statistike nakon završetka meča
- Čuvanje i prikaz istorije svih mečeva svih igrača
- Dopisivanje/čet između korisnika

3. Atributi kvaliteta

Konkretni nefunkcionalni zahtevi su:

- Dostupnost servera 24/7
- Lep i intuitivan korisnički interfejs
- Minimalna latencija tokom meča
- Prevencije i provere protiv varanja igrača (očuvanje integriteta meča)
- Bezbednost protiv osnovnih malicioznih napada
- Lako dodavanje novih funkcionalnosti

- Skalabilno u pogledu broja igrača/mečeva
- Održiv i lako modifikovan/promenljiv kod

4. Tehnička i poslovna ograničenja

Tehnička ograničenja

1. **Web tehnologije (browser-based)**
   - Aplikacija mora raditi isključivo u web browser-u, bez instalacije dodatnog softvera.
   - Mora biti kompatibilna sa modernim browserima (Chrome, Firefox, Edge).
   - Ograničenja performansi zavise od klijentskog uređaja i browser-a.

2. **Real-time komunikacija**
   - Sinhronizacija igre zahteva real-time komunikaciju.
   - Latencija mreže direktno utiče na kvalitet igre i korisničko iskustvo.
   - Potrebno je rukovanje gubitkom konekcije i reconnect mehanizmima.

3. **Baza podataka**
   - Baza mora podržati istovremeni pristup više korisnika.
   - Potrebno je čuvanje istorije mečeva, poteza i statistike.
   - Performanse baze mogu postati usko grlo pri velikom broju aktivnih igrača.

4. **Bezbednosna ograničenja**
   - Autentifikacija i autorizacija moraju biti implementirane.
   - Zaštita od osnovnih napada (SQL Injection, XSS, CSRF).
   - Nemoguće je garantovati potpunu zaštitu od svih oblika varanja.

5. **Skalabilnost**
   - Horizontalna skalabilnost zahteva dodatnu infrastrukturu (load balancer, message broker).
   - U akademskom okruženju skalabilnost je ograničena dostupnim resursima.

Poslovna ograničenja

1. **Akademski karakter projekta**

- Projekat je namenjen u obrazovne svrhe, a ne komercijalnoj upotrebi.
- Fokus je na arhitekturi, dizajnu i pravilnoj primeni obrazaca.
- Ne zahteva potpunu produkcionu stabilnost.

2. **Budžetska ograničenja**
   - Nema finansijskih sredstava za plaćene cloud servise.
   - Hosting, baza i alati moraju biti besplatni ili open-source.
   - Ograničeni resursi za testiranje pod velikim opterećenjem.

3. **Ograničen tim**
   - Mali razvojni tim (2 člana).
   - Jedna osoba često ima više uloga (arhitekta, backend, frontend, dizajn).
   - Ograničen kapacitet za paralelni razvoj i detaljno testiranje.

4. **Održavanje**
   - Dugoročno održavanje aplikacije nije garantovano.
   - Nema stalne tehničke podrške za krajnje korisnike.

5. **Ciljna grupa**
   - Aplikacija je namenjena manjem broju korisnika.
   - Nije optimizovana za masovno tržište.

6. **Pravni i regulativni aspekt**
   - Ne prikupljaju se osetljivi lični podaci korisnika.
   - Nema monetizacije (reklame, oglasi, kupovine u aplikaciji).

5. Arhitekturni obrasci

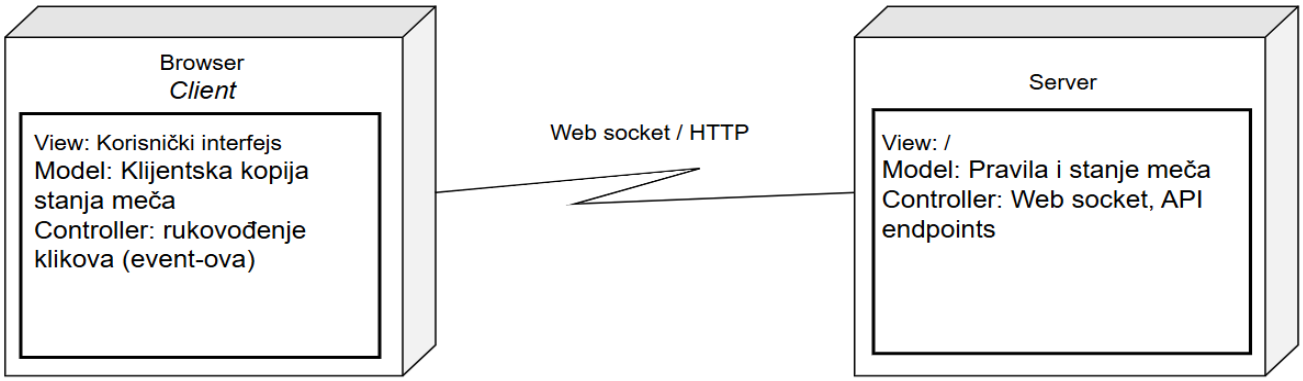
Na arhitekturnom nivou koristiće se Client - server obrazac. On daje jasnu fizičku podelu: klijent/korisnik sa neke udaljene tačke preko browser-a komunicira sa serverom preko web socketa; sva biznis logika kao što je validacija poteza, uparivanje igrača, čuvanje mečeva itd. će se obavljati na serveru.

Na slici 2. je dat pogled na arhitekturu sistema.

Na nivou dizajna/strukture koda koristiće se MVC (Model-View-Controller) obrazac. On definiše slojeve softvera i koje će funkcije svaki sloj obavljati. MVC se nalazi i na klijentskom i na serverskom delu, sa time što server nema "view" deo.

Kod klijenta, view je korisnički interfejs (UI). On pribavlja podatke od modela i prikazuje ih. Model u sebi sadrži **kopiju** podataka koji su prethodno validirani od strane servera. Kontroler rukovodi raznim ulazima od strane korisnika i samo on može da šalje zahteve serveru.

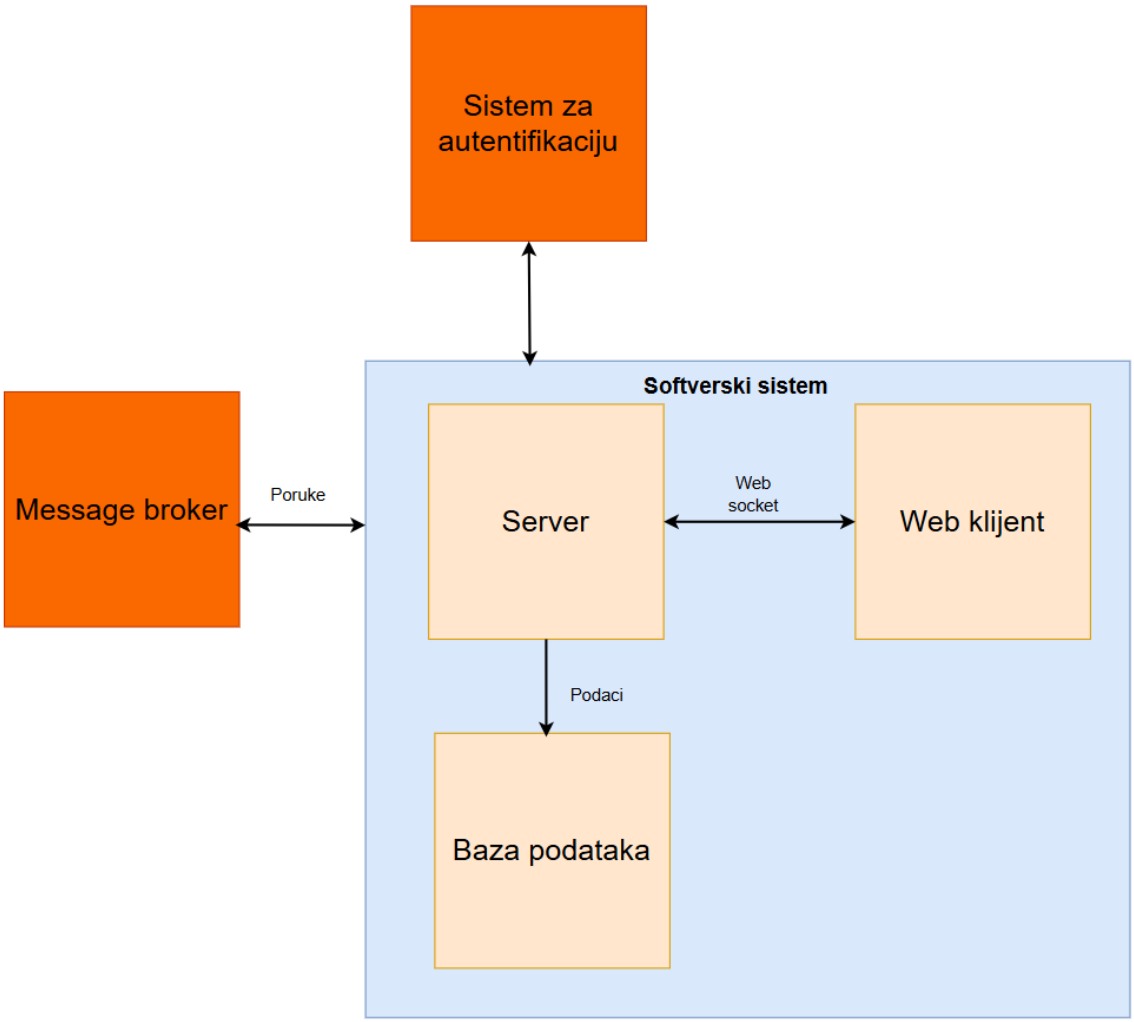
Server ne sadrži view. Model su svi podaci u toku jednog meča. Server prima zahtev od strane klijenta, obrađuje ga i ako je zahtev validan, upisuje ga u bazu podataka i šalje odgovor klijentu. Kontroler je jedan od interfejsa backend API-a.



Slika 2. Arhitektura sistema

6. Generalna arhitektura

Na slici 3. je dat box dijagram. Dijagram prikazuje "high level" arhitekturu celog softverskog sistema.

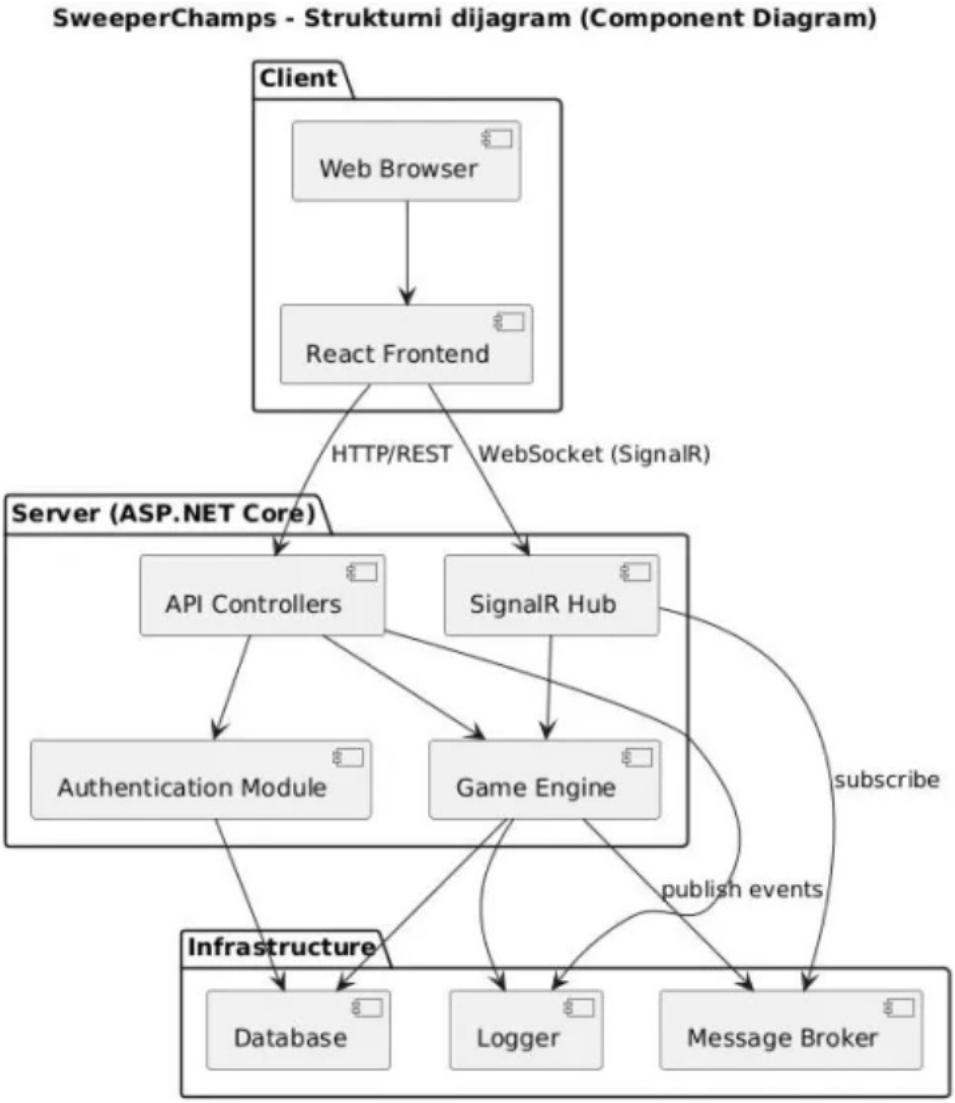


Slika 3. Box dijagram

7. Strukturni i bihevioralni dijagrami

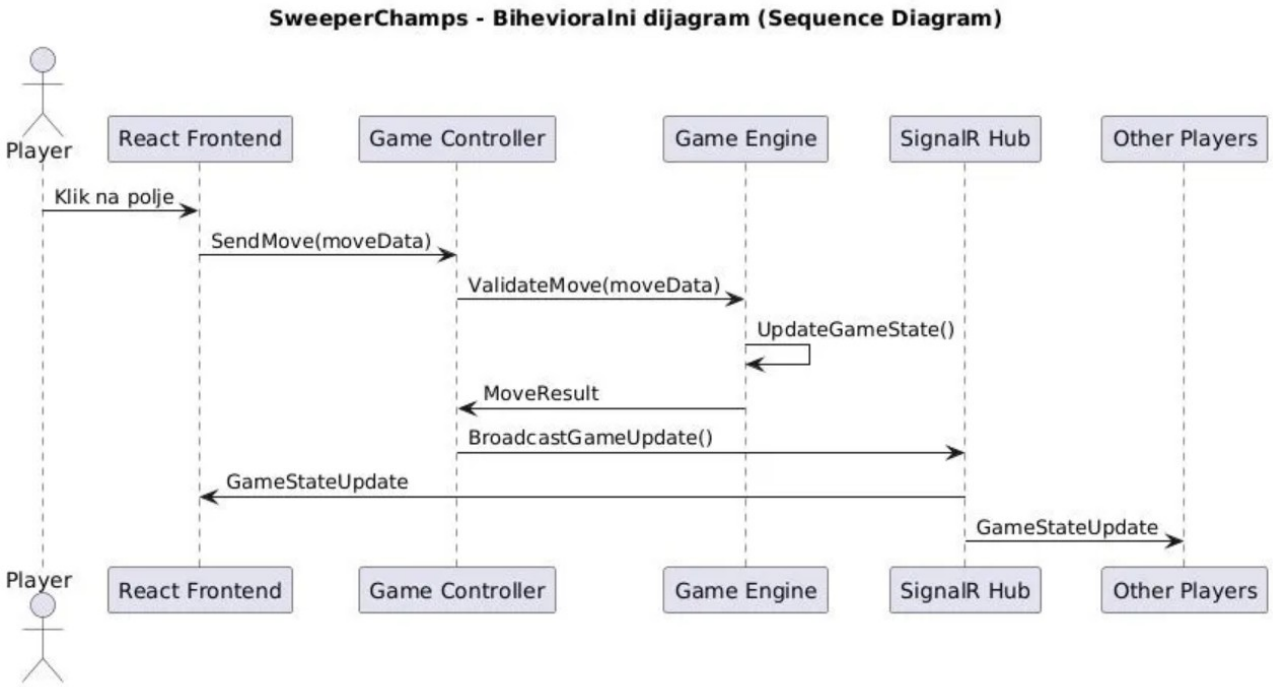
Strukturni pogled sistema prikazuje statičku organizaciju softverskih komponenti i njihove međusobne veze. Sistem je organizovan u više slojeva: klijentski sloj, API sloj, domen sloj, sloj perzistencije i infrastrukturni sloj. Klijentska aplikacija razvijena u React framework-u komunicira sa serverskim API kontrolerima i SignalR hub-om. Poslovna logika igre centralizovana je u Game Engine modulu, koji je logički odvojen od API sloja. Infrastrukturne komponente obezbeđuju real-time komunikaciju, logovanje i asinhronu razmenu poruka.

Na slici 4. prikazan je strukturni dijagram.



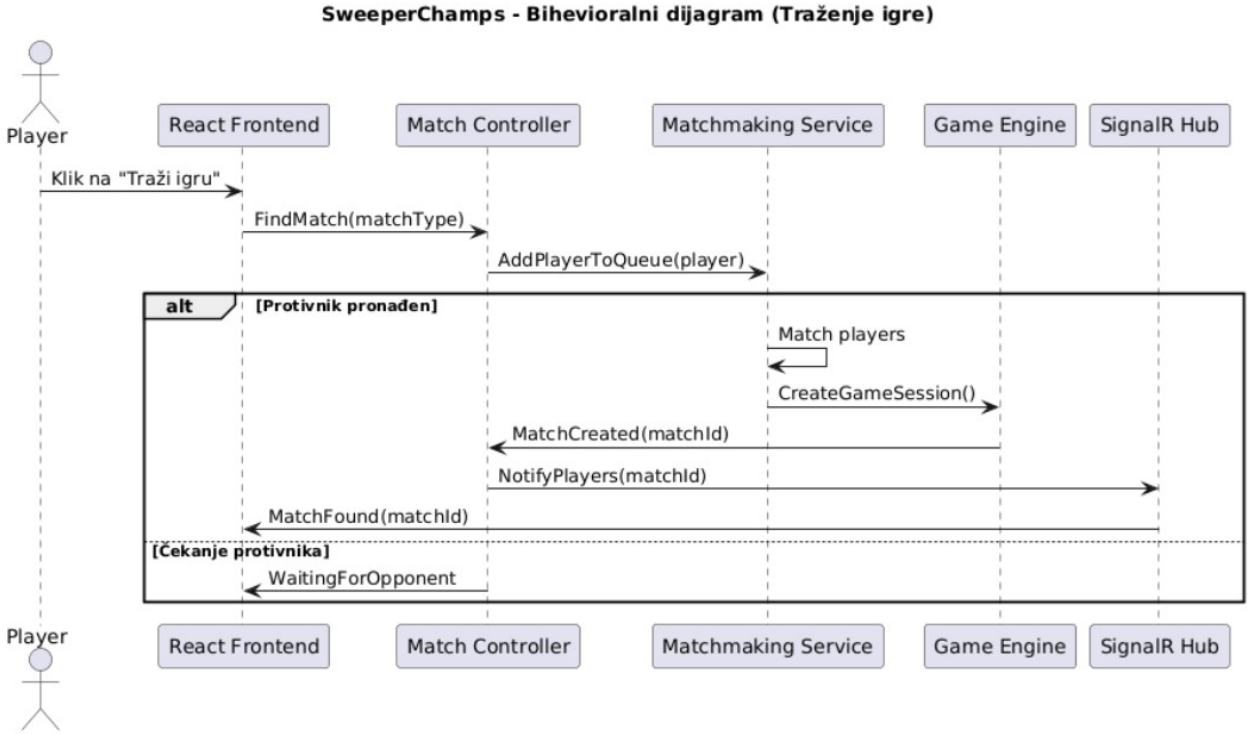
Slika 4. Strukturni dijagram

Bihevioralni dijagram prikazuje tok odigravanja jednog poteza u igri. Nakon korisničke akcije, klijentska aplikacija prosleđuje potez serverskom delu sistema, gde se vrši validacija i ažuriranje stanja igre u okviru Game Engine modula. Nakon uspešne obrade, ažurirano stanje se putem SignalR mehanizma distribuira svim učesnicima meča u realnom vremenu.



Slika 5. Bihevioralni dijagram za odigravanje poteza

Bihevioralni dijagram prikazuje proces traženja i uparivanja igrača za novi meč. Nakon zahteva klijentske aplikacije, serverski deo dodaje igrača u red za uparivanje. Ukoliko se pronađe kompatibilan protivnik, kreira se nova sesija igre i svi učesnici se obaveštavaju putem real-time SignalR komunikacije. U suprotnom, igrač ostaje u stanju čekanja dok se ne pojavi odgovarajući protivnik.



Slika 6. Bihevioralni dijagram za traženje igre

8. Aplikacioni okviri i biblioteke

Za implementaciju serverske komponente sistema biće korišćen aplikacioni okvir **ASP.NET Core**, koji primenjuje obrasce **MVC** i slojevite arhitekture za razdvajanje prezentacione logike, poslovne logike i pristupa podacima. Klijentska komponenta biće realizovana korišćenjem biblioteke **React**, koja primenjuje principe **MVVM/MV*** arhitekture kroz komponentno zasnovan pristup, deklarativni prikaz korisničkog interfejsa i upravljanje stanjem aplikacije. Za perzistenciju podataka koristiće se **PostgreSQL** relaciona baza podataka, dok će **RabbitMQ** biti korišćen kao posrednik za razmenu poruka radi asinhrone komunikacije i obrade događaja unutar sistema.