BO

Here to Slay

Arhitektura softvera

Sardžaj

1. Kontekst i cilj softverskog projekta 2
2. Definisanje zahteva 3
3. Arhitekturni dizajn 4
   - Arhitekturni obrasci 4
   - Generalna arhitektura 6
   - Bihevioralna struktura 7
   - Strukturni pogled 9
4. Implementacija 10

1. Kontekst i cilj softverskog projekta

Here to Slay je izuzetno uzbudljiva igra koja je posebno stvorena za ljubitelje kartaških igara s dodatnim strateškim elementima. Ova igra pruža korisnicima mogućnost da joj pristupe putem veb pregledača, čineći je pristupačnom svima sa stabilnom internet vezom.

Cilj igre je dalje proširen razvojem online multiplejer opcija, fokusirajući se na kreiranje soba za jedan-na-jedan partije. Ova inovativna funkcionalnost omogućava korisnicima da se suoče jedan na jedan, testirajući svoje veštine i strategije direktno protiv drugih igrača.

Proces prijave je jednostavan i brz, što omogućava igračima da se lako povežu i započnu svoje avanture. Nakon prijave, korisnici mogu uživati u realnom vremenu, igrajući partije sa svojim prijateljima ili nepoznatim protivnicima, uz pun pregled i upravljanje svojim kartama.

Svojim jedinstvenim pristupom, *Here to Slay* pruža ljubiteljima strateških igara fantastično iskustvo i uživanje u izazovima dok razvijaju svoje veštine prilikom istraživanja svet ove sjajne online multiplejer igre.

2. Definisanje zahteva

Arhitekturni specifični zahtevi ove izuzetno napredne igre obuhvataju ključne aspekte koji će omogućiti izvanredno korisničko iskustvo.

- Posebno se ističe potreba za realizacijom komunikacije u realnom vremenu, što će omogućiti igračima da istinski uživaju u trenutnim događajima tokom svojih partija.
- Takođe, igra se ističe podrškom za više partija istovremeno, pružajući korisnicima mogućnost da istražuju različite igre i izazove u isto vreme.

Funkcionalni zahtevi ove igre su sveobuhvatni i usmereni ka pružanju bogatog iskustva.

- Proces prijave korisnika postavljen je kao ključan korak, omogućavajući im da brzo i jednostavno pristupe svetu igre.
- Kreiranje partija otvara vrata raznolikosti igara, dok se podela karata odvija sa preciznošću i dinamikom koja održava interesovanje tokom celog iskustva.
- Pregled table igre pruža korisnicima jasnu sliku trenutnog stanja, dok naizmenična igra igrača donosi element iznenađenja i taktičkog razmišljanja.
- Dodatna dimenzija igre dolazi kroz mogućnost korišćenja karata sa posebnim moćima, dajući igračima alate za razvoj jedinstvenih strategija.
- Borba protiv čudovišta dodatno produbljuje iskustvo igre, pružajući izazov i priliku za osvajanje vrednih nagrada.
- Kraj je definisan postizanjem određenog broja poena, čime se postavlja jasan cilj.

Ne-funkcionalni zahtevi su pažljivo definisani kako bi obezbedili da igra postigne vrhunske performanse, nudeći korisnicima brz odziv i neprekidno uzbudljivo iskustvo.

- Ključni aspekt ovih zahteva je optimizacija performansi igre, što znači da igrači mogu uživati u fluidnom i bezbrižnom toku igre bez značajnih kašnjenja.
- Brz odziv je od suštinske važnosti za održavanje angažovanja korisnika.
- Skalabilnost je još jedan ključan ne-funkcionalni zahtev koji se odnosi na sposobnost igre da podrži veliki broj istovremenih korisnika. S obzirom na potrebu za više partija istovremeno i dinamičnim okruženjem multiplejera, igra mora biti u mogućnosti da efikasno rukuje povećanim opterećenjem.
- Sigurnost je neizostavan deo ne-funkcionalnih zahteva, pridajući veliki značaj zaštiti igre od potencijalnih pretnji kao što su neovlašćeni pristupi i prevarantske aktivnosti. Kroz primenu sigurnosnih protokola, enkripciju podataka i sistema zaštite, igra obezbeđuje privatnost i integritet korisničkih informacija, čime stvara poverenje među igračima.

Tehnička i poslovna ograničenja postavljaju temelje za uspešan razvoj i održavanje igre.

- Ograničen pristup bazi podataka. Ova praksa ne samo da povećava bezbednost podataka već i olakšava buduće promene u strukturi baze podataka bez potrebe za modifikacijama u samoj igri. Ovo pruža veću fleksibilnost u održavanju i poboljšanju sistema.
- Korišćenjem optimizovanih mrežnih protokola i infrastrukture, kao i kontinuirano praćenje performansi mreže.

- Optimizacija za popularne pregledače predstavlja ključni tehnički zahtev. Ovaj pristup obezbeđuje širok doseg korisnika, pružajući im pristup igri bez obzira na to koji pregledač koriste.

U cilju poslovnog uspeha, ova tehnička ograničenja se kombinuju s efikasnim poslovnim modelom i strategijama.

3. Arhitekturni dizajn

Ovaj dizajn ima za cilj da zadovolji funkcionalne i nefunkcionalne zahteve sistema, olakša održavanje, proširivost i skalabilnost, i omogući efikasno upravljanje kompleksnošću projekta. Realizija ove aplikacije je prikazana u obliku arhitekturnih obrazaca, strukturalnog pogleda i bihevioralnog pogleda.

Arhitekturni obrasci

- **Layered Arhitektura**: Arhitekturni obrazac koji forsira podelu strukture aplikacije u slojeve. Svaki sloj koristi usluge sloja neposredno ispod, a pruža usluge sloju neposredno iznad. Kod projektovanja aplikacije **Here to Slay** biće korišćena višeslojna arhitktura:
  - **Klijentski sloj**: Klijentski sloj predstavlja korisnički interfejs aplikacije koji korisnicima omogućava interakciju sa sistemom. Angular se koristi za razvoj ovog sloja. Ovde se definišu komponente, servisi i druge strukture koje omogućavaju korisnicima da pregledaju, unose ili menjaju podatke. Klijentski sloj takođe uključuje logiku za obradu korisničkih događaja i komunikaciju sa serverskim slojem putem HTTP zahteva i WebSocket veza.
  - **Komunikacioni sloj**: Komunikacioni sloj je odgovoran za realizaciju komunikacije između klijentskog i serverskog sloja. U ovom slučaju, koristi se **@nestjs/websockets** biblioteka za implementaciju WebSocket komunikacije. Ovaj sloj omogućava stvaranje realno-vremenskih veza između klijenta i servera, što je posebno korisno za slanje trenutnih ažuriranja ili obaveštenja klijentima bez potrebe za stalnim osvežavanjem stranica.
  - **Serverski sloj:** Serverski sloj obuhvata glavnu logiku i funkcionalnosti aplikacije. NestJS se koristi za implementaciju serverskog sloja. Ovde se definišu kontroleri, servisi, middleware-ovi i druge komponente koje obrađuju zahteve klijenata. Serverski sloj koristi WebSocket modul za prihvatanje WebSocket veza, obrađivanje poruka i slanje odgovora. Takođe, ovde se uspostavlja veza sa bazom podataka radi čitanja i pisanja podataka.
  - **Sloj baze podataka:** Sloj baze podataka je odgovoran za trajno skladištenje i upravljanje podacima. U ovom slučaju, koristi se PostgreSQL kao sistem za upravljanje bazom podataka. Sloj baze podataka sadrži šemu podataka, tabele i upite koji omogućavaju efikasno čuvanje i dohvatanje informacija. Serverski sloj komunicira sa bazom podataka kako bi čitao i upisivao podatke prema zahtevima aplikacije.

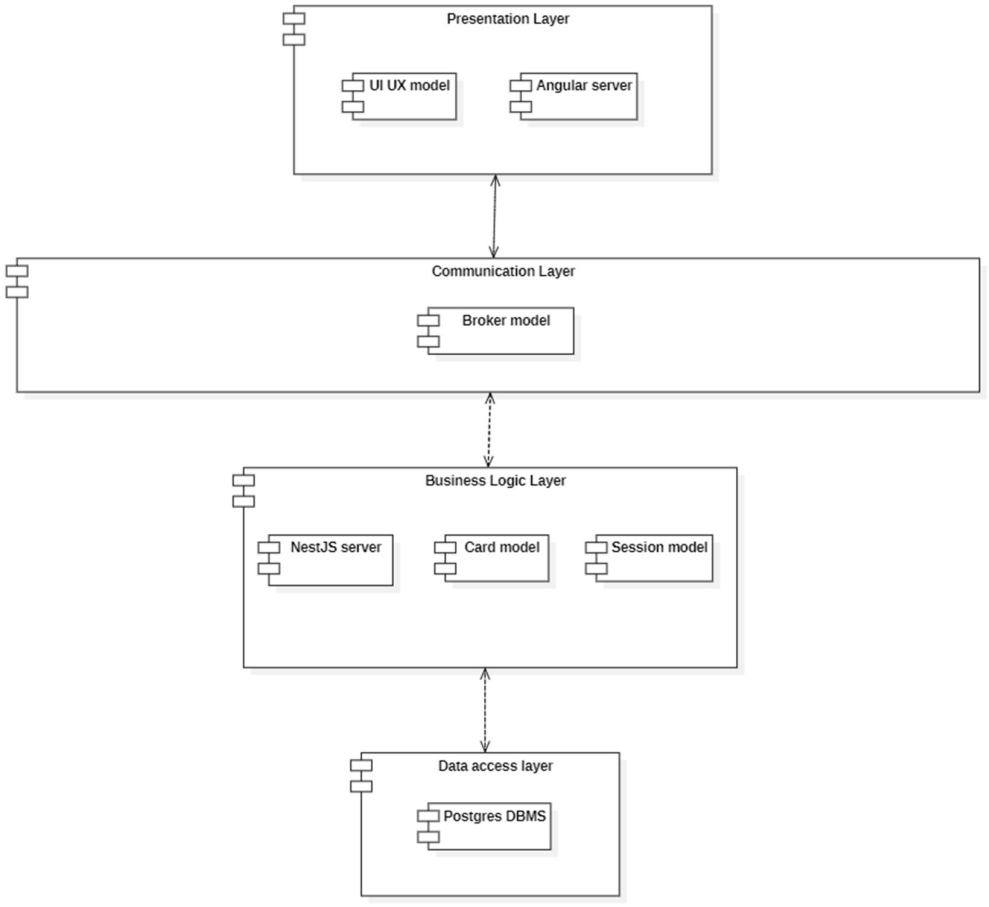
- **Broker obrazac:** Websocket u ovoj aplikacije deluje kao broker, odnosno posrednik u komunikaciji između klijenta i servera. Kada igrač izvrši neku akciju, front-end putem

Websocket-a šalje poruku back-end-u. Nakon obrade akcije, ažurira se stanje igre a zatim šalje poruku nazad sa ažuriranim stanjem igre klijentu.

- **Event-based arhitektura:** komponente u ovoj arhitekturi komuniciraju putem razmene događaja (events). Događaji predstavljaju bilo kakve akcije korisnika kao potezi ili napadi na monster karte. Korišćenjem ovog obrazca, moguće je konstanto vršiti upite I reagovati na njih u realnom vremenu kao npr za modifier karte. Pomoću brokera se izvršava event-based arhitektura i dolazi do tečnog odliva i priliva događaja.
- **Model-View-Controller (MVC):** je arhitekturni obrazac u programiranju koji organizuje logiku aplikacije u jasno definisane slojeve: Model, Pogled (View) i Kontroler (Controller). Svaki sloj ima specifične zadatke, a interakcija između slojeva omogućava koordiniranu i podeljenu isporuku funkcionalnosti aplikacije. Ovaj obrazac obuhvata celu aplikaciju, od korisničkog interfejsa (UI) do osnovnog modela podataka.

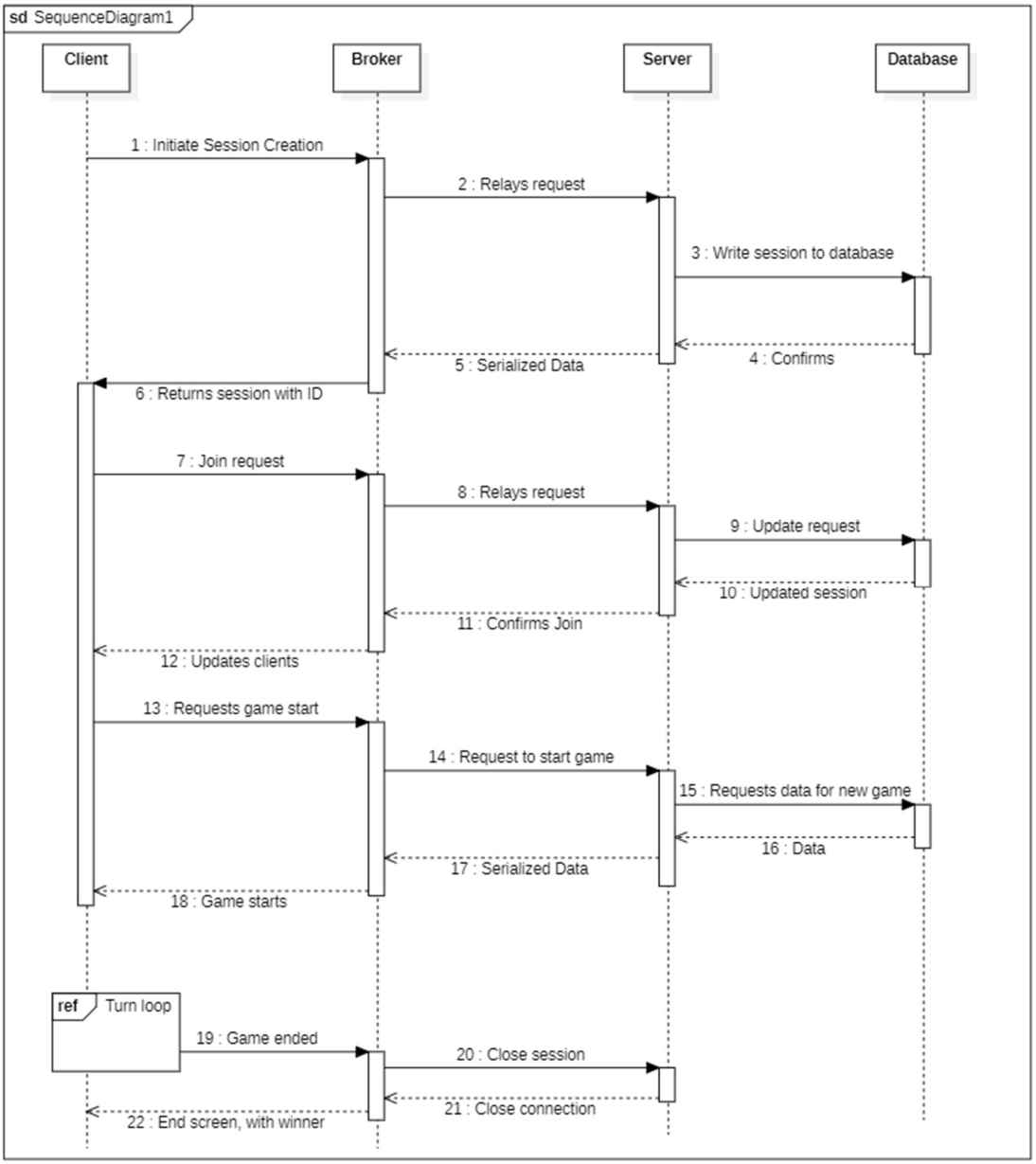
Generalna arhitektura

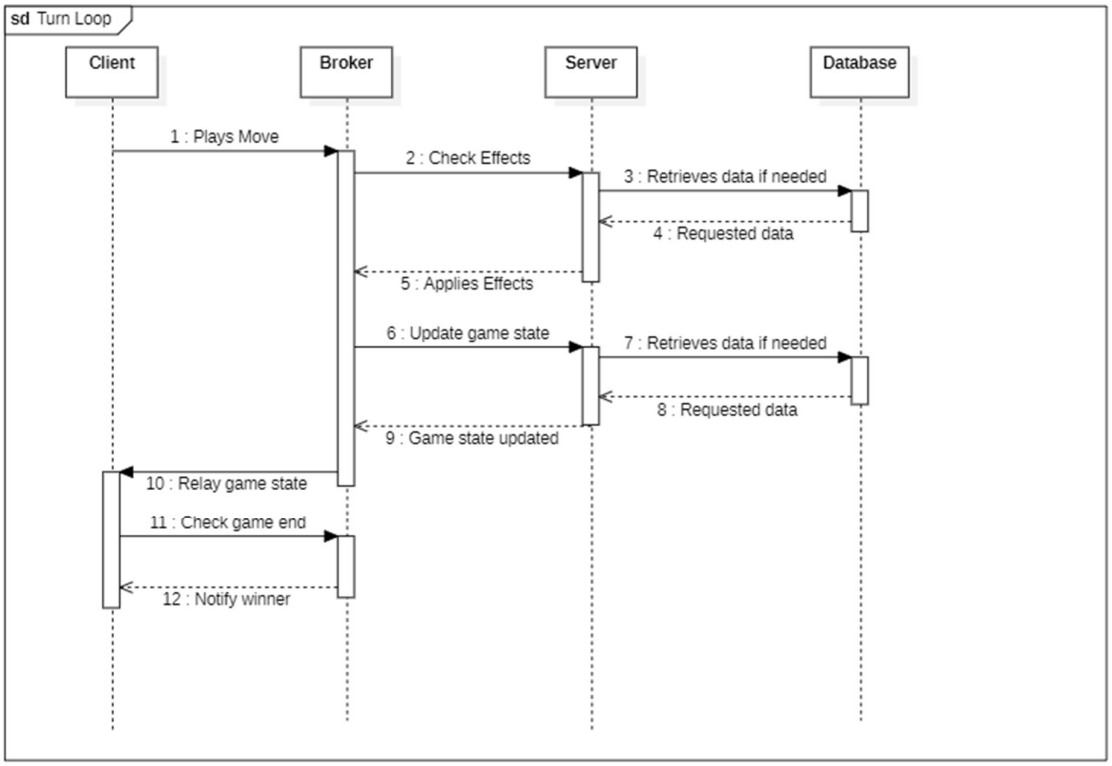
- Generalna arhitektura se najbolje prikazuje slojevitom strukturom u kome imamo uvid o svakom podsistemu I kako se komunicira između njih.



Bihevioralna struktura

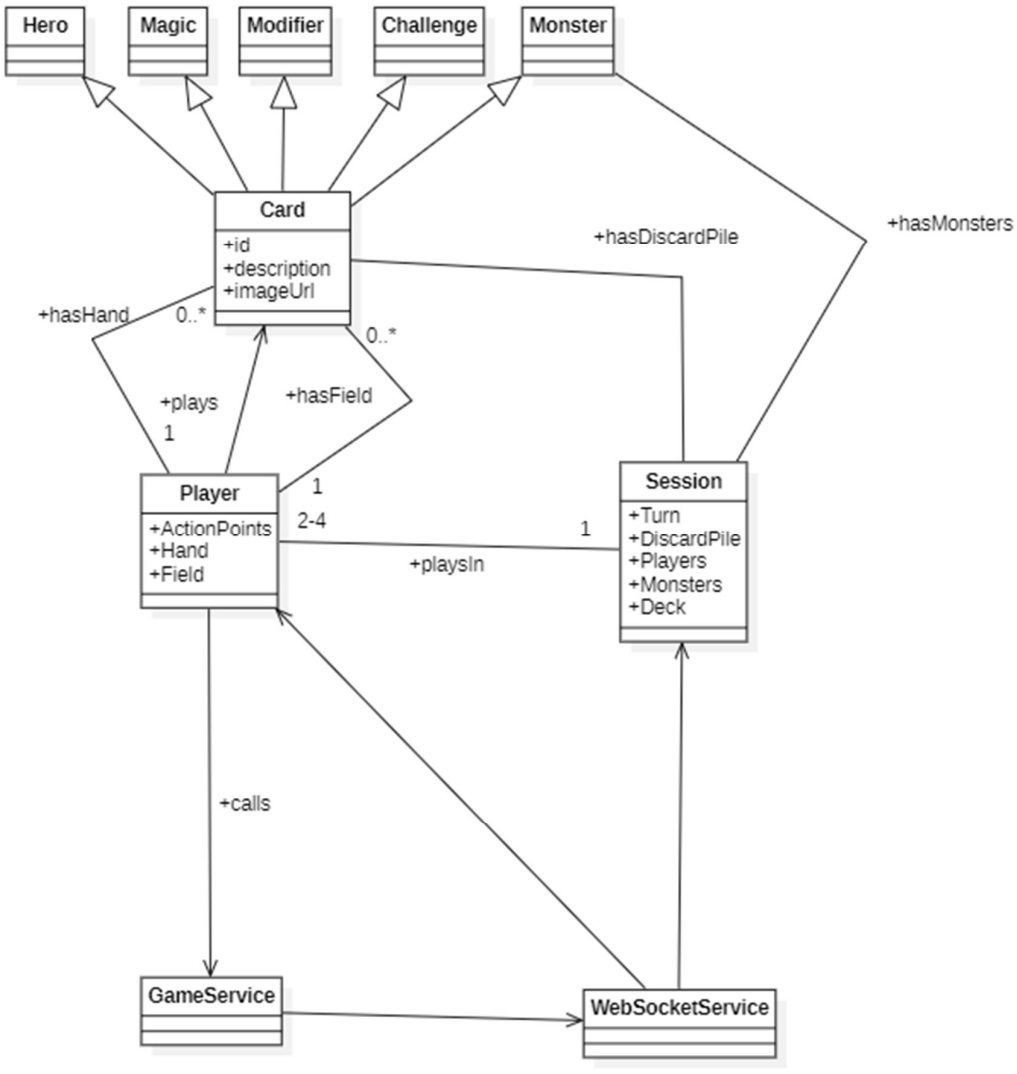
- Opšta bihevioralna struktura se manifestuje kroz formiranje sesija i odigravanje same sesije.U spešno odigravanje između dva ili više igrača se prikazuje sledećim sekvencijalnim dijagramima.




sd Turn Loop
Client
Broker
Server
Database
1 : Plays Move
2 : Check Effects
3 : Retrieves data if needed
4 : Requested data
5 : Applies Effects
6 : Update game state
7 : Retrieves data if needed
8 : Requested data
9 : Game state updated
10 : Relay game state
11 : Check game end
12 : Notify winner

Strukturni pogled

- Predstavlja se kroz ovaj UML klasni dijagram koji ujedno povezuje klijentsku stranu preciznije kako komunicira preko brokera (koristeći servise) i kako elementi zavise jedno u druge u toku realne sesije igre.



4. Implementacija

- **Angular** – front-end
- Angular je popularni open-source framework za izgradnju modernih veb-aplikacija. Razvijen od strane Google-a, Angular koristi TypeScript i omogućava lako kreiranje dinamičnih i skalabilnih korisničkih interfejsa. Sa snažnim sistemom komponenata, Angular olakšava organizaciju i održavanje koda, uz podršku za kompleksne funkcionalnosti poput rutiranja, HTTP komunikacije i upravljanja stanjem aplikacije.
- **Nest.JS** – back-end
- NestJS je moderni, open-source Node.js framework za izradu skalabilnih server-side aplikacija. Baziran na TypeScriptu i inspirisan Angularom, NestJS koristi modularnu arhitekturu zasnovanu na modulima, što olakšava organizaciju i održavanje koda. Sa ugrađenom podrškom za asinhrono programiranje i integraciju sa popularnim alatima, NestJS je idealan za izgradnju efikasnih i robustnih back-end sistema..
- **PostgreSQL** – objetno-relacioni sistem za upravljanje podacima
- **WebSockets** - biblioteka koja omogućava dinamičko ažuriranje podataka u stvarnom vremenu
- **TypeORM** – TypeScript kompatibilan alat za objektno-relaciono mapiranje