

# Duel Nexus

Model podataka i perzistencije

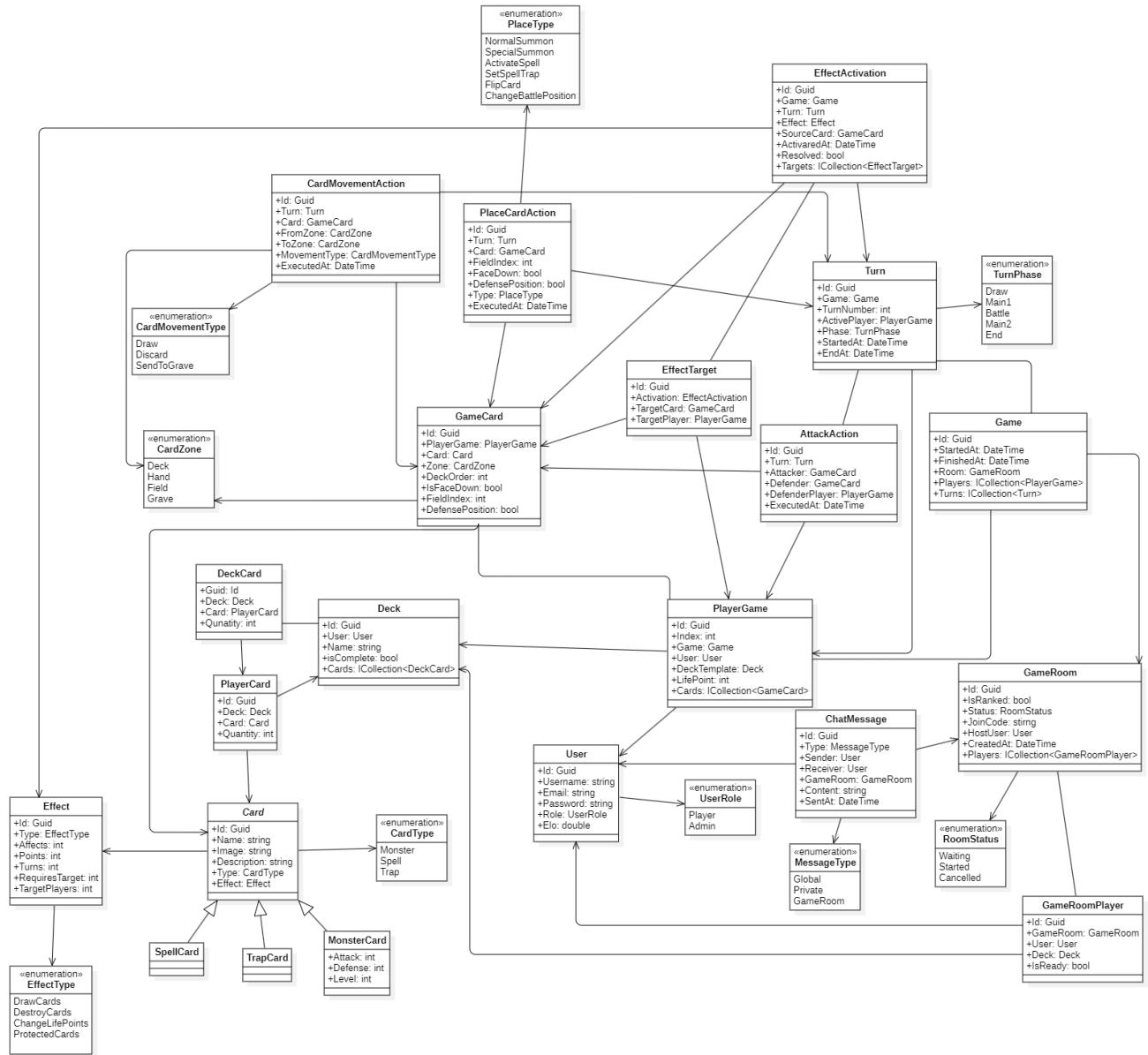
*Lazar Mančić 19206*

*Momčilo Marjanović 19210*

Datum: 11.02.2026.

# 1 Model podataka

Model podataka u aplikaciji Duel Nexus predstavljen je sledećim klasnim dijagramom:



**User** - Predstavlja korisnika sistema sa osnovnim podacima (username, email, lozinka) i ELO ratingom koji se koristi za rangiranu igru. Svaki korisnik može da ima kolekciju karata, decks i da učestvuje u igrama.

**Card** - Apstraktna bazna klasa koja definiše zajedničke osobine sve tri vrste karata (naziv, slika, opis, tip i opcioni efekat). Ovo je roditeljska klasa za MonsterCard, SpellCard i TrapCard.

**Monster Card** - Konkretna implementacija kartice sa dodatnim svojstvima: napad, odbrana i nivo. Koristi se u borbi sa protivničkim čudovištima.

**Spell Card** - Konkretna implementacija kartice koja predstavlja magiju sa specifičnim efektima koji se mogu aktivirati tokom igre.

**Trap Card** - Konkretna implementacija kartice koja predstavlja zamku i obično se postavlja licem prema dole da bi se aktivirala kao odgovor na protivničke akcije.

**Effect** - Predstavlja efekat koji kartica može imati, čuvajući informacije o tipu efekta, broju karata koje je pogađa, poena štete/lečenja i broju poteza tokom kojih traje.

**Deck** - Kolekcija karata koju kreira korisnik i koristi kao svoj igrački deck. Svaki deck ima naziv i status kompletnosti te pripada jednom korisniku.

**Deck Card** - Veza između decka i kartice sa količinom te kartice u decku. Ista kartica može da se koristi više puta

**Player Card** - Predstavlja kolekciju karata koju korisnik poseduje - to je njegov inventar od kog može da pravi dekove.

**Game Room** – Predstavlja lobi koji kreira host korisnik. Može biti rangirani ili casual, sa opcionalnim join kodom za pristup. Služi kao prostor gde se igrači okupljaju, biraju svoj deck, označavaju da su spremni.

**Game Room Player** - Veza između korisnika i game room-a sa prikazom da li je korisnik spreman i koji je deck odabrao

**Game** - Predstavlja samu igru sa vremenom početka i završetka, zajedno sa referencom na game room u koje se igra odigrava i listom svih poteza.

**Player Game** - Veza između korisnika i konkretne igre sa life points-ima (8000 na početku) i listom svih karata koje taj igrač koristi u toj igri.

**Game Card** - Kartica tokom konkretne igre sa pozicijom, zonom (ruka, polje, graveyard itd), orientacijom i statusima (licem prema dole, u odbrani).

**Turn** - Predstavlja jedan potez u igri sa rednim brojem, aktivnim igračem, fazom poteza i vremenima početka i kraja.

**Attack Action** - Beleži akciju napada jedne kartice na drugu karticu ili direktno na igrača tokom određenog poteza.

**Card Movement Action** - Beleži akciju premeštanja kartice između različitih zona (iz ruke na polje, iz polja u graveyard itd) tokom poteza.

**Place Card Action** - Beleži akciju postavljanja kartice sa ruke na polje sa informacijama o poziciji, orijentaciji i tipu postavljanja.

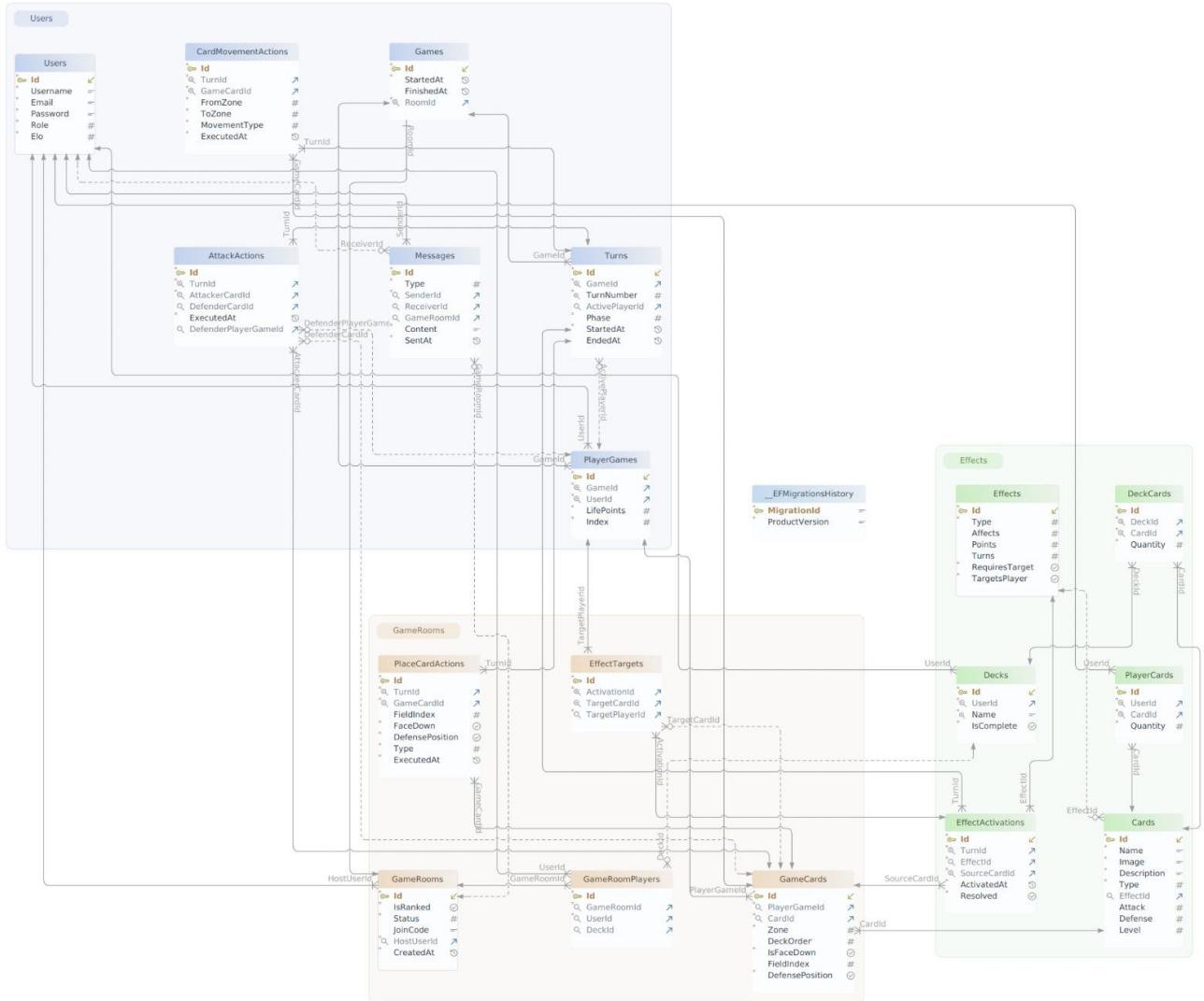
**Effect Activation** - Predstavlja aktivaciju efekta tokom poteza sa informacijom o kartici koja je izazvala efekat, da li je razrešen i listom ciljeva efekta.

**Effect Target** - Čuva informaciju o cilju efekta - može biti određena kartica ili igrač. Svaki efekat može imati više ciljeva.

**Chat Message** - Predstavlja poruku u čatu između igrača sa informacijom o tipu (privatna ili u sobi), pošiljaocu, primaocu i vremenu slanja.

## 2 Model Perzistencije

Predstavljeni modeli podataka se perzistiraju na odgovarajući način u bazi podataka, predstavljen u vidu modela entiteta



## 3 Mehanizmi mapiranja

Za perzistenciju i rad sa bazom podataka je korišćen Postgres SQL, dok je za mapiranje između objekata klase iz modela podataka i entiteta baze podataka korišćen objektno-relacioni mape Entity Framework Core. Princip koji se koristi za mapiranje je code-first, gde se na osnovu Entity klase, kojima je prestavljen model podataka, kreiraju tabele relacione baze podataka. Prilikom mapiranja su primenjeni i DataLayer obrasci Repository i UnitOfWork. Takođe, pored modela podataka, iskorišćene su i DTO klase za prenos podataka između slojeva aplikacije.