Strukturalni design patterni

Za implementaciju odabrani su Facade i Decorator patterni.

1. Opis strukturalnih dizajn paterna

1.1 Adapter

Adapter patern je design pattern čija je osnovna namjena da interfejs jedne klase pretvori u neki željeni interfejs kako bi se ona mogla koristiti u situaciji u kojoj bi inače problem predstavljali nekompatibilni interfejsi. Primjenom Adapter paterna dobija se željena funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije. U kontekstu OffroadAdventure sistema, mogao bi se koristiti za spajanje sistema plaćanja sa vanjskim sistemima autorizacije kartica koji koriste drugačiji interfejs.

1.2 Bridge

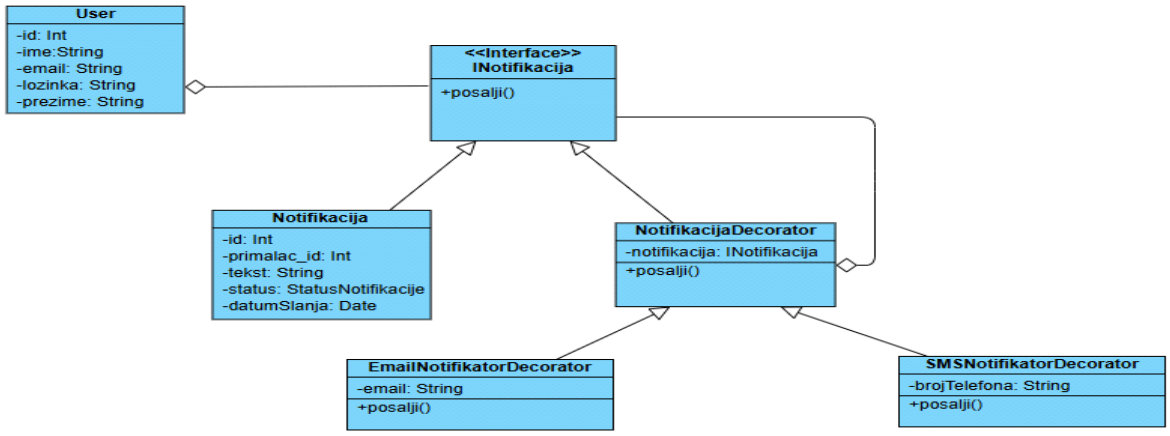
Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. U sistemu se može koristiti za razdvajanje različitih načina isporuke notifikacija. Apstraktna klasa može definisati interfejs, dok se implementacije mogu promijeniti nezavisno od klijentskog koda.

1.3 Composite

Osnovna primjena Composite paterna jeste pravljenje hijerarhije klasa i omogućavanje pozivanja iste metode nad različitim objektima sa različitim implementacijama. U kontekstu komentara u OffroadAdventure sistemu, omogućava tretiranje pojedinačnih komentara i grupa komentara na isti način.

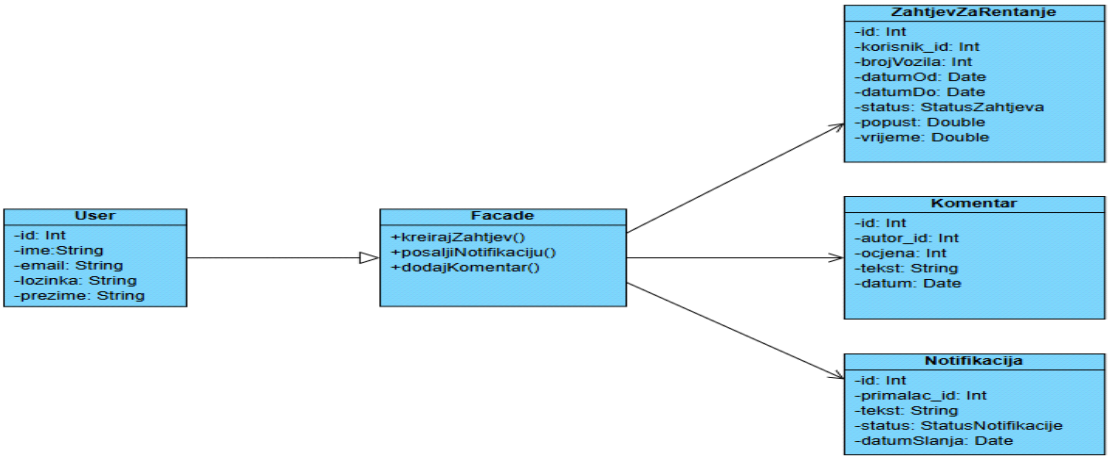
1.4 Decorator

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Decorator pattern je primijenjen na komponentu "Notifikacija", kako bi se omogućilo fleksibilno dodavanje načina slanja notifikacija bez mijenjanja osnovne klase. Korištenjem ovog paterna možemo, na primjer, istu notifikaciju poslati na više načina (email, SMS), bez dupliranja logike. Na dijagramu ispod prikazana je struktura gdje interfejs "INotifikacija" definiše metodu "posalji()". Osnovna klasa "Notifikacija" implementira ovu metodu, dok "NotifikacijaDecorator" služi kao apstraktni dekorator koji dodaje slojeve dodatne funkcionalnosti. Konkretni dekoratori kao što su "EmailNotifikatorDecorator" i " "SMSNotifikatorDecorator" proširuju ponašanje metode "posalji()" i omogućavaju slanje notifikacija na više načina.



1.5 Facade

Facade patern se koristi kada sistem ima više identificiranih podsistema pri čemu su apstrakcije i implementacije podsistema usko povezane. Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme. Ovaj patern sakriva implementaciju podsistema od korisnika i pruža pojednostavljeni interfejs putem kojeg korisnik pristupa sistemu. U ovom projektu koristi se za interakciju sa komponentama za rad sa zahtjevima, plaćanjima i notifikacijama, tako da se složene interakcije kapsuliraju unutar jedne klase. Sledeći dijagram precizno prikazuje kako "Facade" klasa centralizuje i pojednostavljuje komunikaciju sa tri važne poslovne komponente: "ZahtjevZaRentanje", "Komentar" i "Notifikacija". Metode "kreirajZahtjev()", "dodajKomentar()" i "posaljiNotifikaciju()" omogućavaju korisniku da kroz jedinstven interfejs komunicira sa sistemom, bez direktne interakcije sa svakom od komponenti zasebno.



1.6 Proxy

Proxy patern omogućava kontrolisani pristup stvarnim objektima. U sistemu bi mogao biti iskorišten za kontrolu pristupa podacima korisnika, komentara ili zahtjeva, posebno ako postoji mehanizam autentifikacije i autorizacije koji zavisi od prava pristupa.