

Analiza SOLID principa za model sistema (Rentanje offroad vozila)

1. Single Responsibility Principle (SRP)

Svaka klasa ima jednu odgovornost:

- Vozilo: predstavlja entitet vozila sa atributima koji opisuju vozilo (model, tip, dostupnost, cijena...).
- Klijent, Gost, Korisnik: predstavljaju korisničke uloge, hijerarhijski raspoređene.
- Zaposlenik i Vlasnik: specifične vrste osoblja sa vlastitim ulogama.
- ZahtjevZaRentanje i StavkaZahtjeva: modeliraju logiku vezanu za zahtjev i stavke zahtjeva.
- Notifikacija, Komentar, Plaćanje: svaka klasa pokriva isključivo svoju funkcionalnost (obavještenja, komentare, transakcije).

Zadovoljen: svaka klasa ima jasnu i jedinstvenu odgovornost.

2. Open/Closed Principle (OCP)

Klase su otvorene za proširenje, ali zatvorene za izmjene:

- Hijerarhije poput Korisnik → Klijent → Gost omogućavaju dodavanje novih tipova korisnika bez mijenjanja postojećih klasa.
- Enumeracije (StatusZahtjeva, StatusNotifikacije, NacinPlacanja) olakšavaju proširenje bez promjene postojeće logike.

Zadovoljen: sistem može da se proširi novim tipovima korisnika, statusima i načinima plaćanja bez direktnog mijenjanja postojećeg koda.

3. Liskov Substitution Principle (LSP)

Objekti izvedenih klasa mogu zamijeniti objekte baznih klasa bez narušavanja funkcionalnosti:

U prikazanom dijagramu klasa nije implementirana nasljedna hijerarhija između različitih tipova korisnika. Umjesto nasljeđivanja koristi se entitet UserRoles za upravljanje ulogama korisnika preko relacione veze, što je prikladno za domenski model.

Primjena LSP principa u ovom modelu nije relevantna jer se odnosi isključivo realizuju preko relacija, a ne nasljeđivanja.

4. Interface Segregation Principle (ISP)

Klase ne bi trebale biti prisiljene da implementiraju metode koje ne koriste:

- Iako nema eksplicitnih interfejsa, funkcionalnosti su fino razdvojene u različite klase (Notifikacija, Komentar, Plaćanje...), pa se izbjegava zagušenje nepotrebnim metodama.

Zadovoljen: funkcionalnosti su modularno podijeljene, pa bi eventualni interfejsi u implementaciji bili uski i specifični.

5. Dependency Inversion Principle (DIP)

Zavisnosti trebaju biti prema apstrakcijama, ne prema konkretnim klasama:

- Veze u dijagramu su uglavnom prema entitetima i enumeracijama.
- Klase ZahtjevZaRentanje, Plaćanje, Notifikacija zavise od enum tipova (Status, NacinPlacanja), što predstavlja osnovnu apstrakciju u domenskom modelu.

Djelimično zadovoljen: u pravoj implementaciji treba dodatno uvesti interfejse za servise (npr. servis za slanje notifikacija, obradu plaćanja), što nije prikazano na modelu ali ne krši princip.

Zaključak

Prikazani model sistema u velikoj mjeri ispunjava svih pet SOLID principa. Dizajn je modularan, proširiv, stabilan na promjene i podržava dobru praksu objektno orijentisanog programiranja. Za potpunu implementaciju DIP i ISP principa preporučuje se korištenje interfejsa i servisa u aplikacijskom sloju.