SOLID principi

1) Single Responsibility Principle

Princip glasi: “Klasa bi trebala imati samo jedan razlog za promjenu”. U našem dijagramu klasa, ovaj princip je zadovoljen jer svaka klasa upravlja svojim atributima, koji su privatni. Funkcionalnosti sistema su razdvojene tako da svaka klasa ima pojedinačnu odgovornost. Npr. klasa Rezervacija služi za upravljanje rezervacijom, te je razdvojena od Korisnika i Karte. Također, iako su Korisnici ti koji postavljaju i odgovaraju na pitanja, ove funkcionalnosti smo odvojili u zasebnu klasu Pitanje, budući da to nije primarna uloga Korisnika.

2) Open Closed Principle

Princip glasi: “Entiteti softvera (klase, moduli, metode) trebali bi biti otvoreni za nadogradnju, ali zatvoreni za modifikacije”. Obzirom da su klase povezane vezama asocijacije i agregacije, te da kao atribute često imamo objekte drugih klasa, promjena u jednoj klasi neće značiti promjenu u ostalim klasama. Npr., promjenom objekata klasa Korisnik i Karta, korištenih unutar klase Rezervacija, ne dolazi do stvarne promjene nad klasom Rezervacija, koja nastavlja vršiti svoje funkcionalnosti.

3) Liskov Substitution Principle

Princip glasi: “Podtipovi moraju zamjenjivi njihovim osnovnim tipovima”. Unutar našeg dijagrama klasa nemamo apstraktnih klasa, te samim tim je princip zadovoljen. Razlikovanje vrsta korisnika sistema je implementirano preko enuma Uloga (Korisnik, Zaposlenik i Administrator).

4) Interface Segregation Principle

Princip glasi: “Klijenti ne treba da ovise o metodama koje neće upotrebljavati”. U našem sistemu je ovaj princip zadovoljen, budući da se funkcionalnosti razdvajaju u zavisnosti od uloge korisnika, definisane kao atribut klase Korisnik.

5) Dependency Inversion Principle

Princip glasi: “Moduli visokog nivoa ne bi trebali ovisiti od modula niskog nivoa” / “Moduli ne bi trebali ovisiti od detalja”. Kao i na primjeru principa 3, kako nemamo apstraktnih klasa, ovaj princip je time zadovoljen.