SOLID principi

1. Single Responsibility Principle (SRP)

Svaka definirana klasa ima jasnu odgovornost: Korisnik sadrži sve informacije o korisniku, vozilo opisuje karakteristike vozila, recenzija je vezana za ocjenu i tekstualni osvrt i rezervacija predstavlja proces iznajmljivanja vozila i povezuje korisnika, vozilo i recenziju.

2. Liskov Substitution Principle (LSP)

LSP se odnosi na nasljeđivanje, a nema nasljeđivanja među klasama jer su uloge definirane kao varijabla unutar Korisnik klase, a ne kao zasebne podklase.

3. Interface Segregation Principle (ISP)

ISP princip nalaže da korisnici ne bi trebali biti prisiljeni da zavise od interfejsa koje ne koriste, tj. bolje je imati više manjih interfejsa nego jedan "sveobuhvatni interfejs". Zbog toga, implementirali smo više malih interfejsa sa jasno definisanim ulogama: IKorisnickiNalog: prijava/ odjava korisnika, IRezervacijaUpravljanje: upravljanje rezervacijama, IVoziloPretraga: pretraga i upravljanje vozilima, IRecenzijaUpravljanje: upravljanje recenzijama.

4. Dependency Inversion Principle (DIP)

Klase visokog nivoa ne smiju zavisiti od klasa niskog nivoa, već i jedne i druge trebaju da zavise od apstrakcija (interfejsa). Korisnik zavisi od IKorisnickiNalog, Rezervacija implementira IRezervacijaUpravljanje, Vozilo implementira IVoziloPretraga, a Recenzija implementira IRecenzijaUpravljanje. Na taj način, Korisnik se oslanja na apstraktni interfejs, a ne na konkretnu klasu, moguća je izmjena načina upravljanja rezervacijama bez izmjene korisničkog sloja, pretraga vozila zavisi od apstrakcije, a ne od konkretne klase, i klasa Recenzija se može izmijeniti bez utjecaja na ostatak sistema.

5. Open/Closed Principle (OCP)

Klase su otvorene za proširivanje, ali zatvorene za izmjene. Na primjer, ako želimo dodati nove funkcionalnosti u pretrazi vozila (npr. filtriranje po cijeni ili vrsti goriva), to možemo učiniti proširivanjem postojećih interfejsa ili dodavanjem novih klasa bez izmjene već postojećih. Na taj način se osnovna logika sistema ne dira, čime se izbjegavaju greške i omogućava lakše održavanje i nadogradnja koda.