SOLID PRINCIPI

1. **S**ingle responsibility principle (princip pojedinačne odgovornosti)

SRP: "KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU!"

Klase treba da znaju samo o jednoj stvari. One trebaju imati pojedinačnu odgovornost. Treba postojati samo jedan razlog za promjenu klase.
Princip pojedinačne odgovornosti je zadovoljen jer svaka klasa ima samo po jednu odgovornost ili skup odgovornosti vezanih za samo jedan posao.
Sve dodatne odgovornosti povezane za te klase su izdvojene u interfejs Profil. Primjer je klasa RegistrovaniKorisnik koja koristi interface za uredivanje profila.

2. **O**pen closed principle (otvoreno-zatvoreni princip)

OCP: "ENTITETI SOFTVERA (KLASE, MODULI, FUNKCIJE) TREBALI BI BITI OTVORENI ZA NADOGRADNJU, ALI ZATVORENI ZA MODIFIKACIJE!"

Ovaj princip podrazumijeva da klase trebaju biti otvorene za nadogradnje i proširivanja, bez da to mijenja druge, već postojeće klase tj. trebamo biti sposobni mijenjati okruženje oko modula bez promjene samog modula.
Ovaj princip je također zadovoljen s obzirom na činjenicu da je najveći broj veza u sistemu agregacija, što znači da klase sadrže druge klase i izmjena neke od njih ne povlači za sobom izmjene drugih klasa.
Primjer je klasa Pitanja koja podrazumijeva samo skupinu pitanja sa već ponuđenim odgovorima, a možemo je lako nadograditi sa novom vrstom pitanja koja recimo kao odgovor podrazumijevaju skalu.
RegistrovaniKorisnik (samim tim i VIPKorisnik) agregiraju klasu Pitanja, što znači da sve ostaje nepromijenjeno uz nove mogućnosti.

3. **L**iskov substitution principle (Liskov princip zamjene)

LSP: "PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA!"

Ovaj princip traži da podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima tj. da upotrebom bazne umjesto izvedene klase dobijamo isti rezultat.
U našem sistemu je ovo zadovoljeno s obzirom na to da imamo samo jedno nasljeđivanje. Očigledno je klasa VIPKorisnik zamjenjiva klasom RegistrovaniKorisnik jer oboje podrazumijevaju korisnika aplikacije i moraju imati iste ishode korištenjem svih metoda koje možemo koristiti za klasu RegistrovaniKorisnik.

4. **I**nterface segregation principle (princip izoliranja interfejsa)

ISP: “KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI!”

Ovaj princip za cilj ima zaštitu klijenta od promjena metoda koje ga se ne tiču, te zaštitu od od poznavanja imlementacije objekta kojeg koristi.
S obzirom na činjenicu da naš sistem posjeduje samo jedan interfejs, Profil, koji ograničava klasu da podrži samo potrebne metode, ovaj princip smatramo zadovoljenim. Nije potrebno grananje interfejsa, jer sve klase (tj. samo RegistrovaniKorisnik) koriste sve njegove metode.

5. **D**ependency inversion principle (princip inverzije ovisnosti)

DIP1: “MODULI VISOKOG NIVOA NE BI TREBALI OVISITI OD MODULA NISKOG NIVOA. OBA BI TREBALO DA OVISE OD APSTRAKCIJA!”

DIP2: “MODULI NE BI TREBALI OVISITI OD DETALJA. DETALJI BI TREBALI BITI OVISNI OD APSTRAKCIJA!”

Ne treba ovisiti od konkretnih klasa. Prilikom nasljeđivanja treba razmotriti slučaj da je osnovna klasa apstraktna, jer se apstraktne klase i interfejsi znatno manje mijenjaju od njihovih konkretnih izvedenica. Na ovaj način štitimo sistem od utjecaja mogućih promjena.
Naš sistem slijedi i ovaj princip jer svaka strelica završava u apstraktnoj klasi ili interfejsu. Primjer je jedino VIPKorisnik koji nasljeđuje klasu RegistrovaniKorisnik, a koja implementira interfejs Profil.
Važno je ipak napomenuti da postoji ovisnost o klasama String i List, koje su konkretne, te se, pretpostavlja se, neće znatno mijenjati u narednoj dekadi, što znači da je oslanjanje na njih sigurno. Zahvaljujući ovome sistem nije osjetljiv na promjene.