SOLID PRINCIPI

1. Single Responsibility Principle – Princip pojedinačne odgovornosti

Princip je zadovoljen, kako svaka postojeća klasa posjeduje jednu, pojedinačnu odgovornost. To možemo vidjeti na primjeru klasa Osoba, Korisnik i Admin koje su zadužene za upravljanje informacijama o akterima. Također, imamo i klasu Recept koja samo sadrži link recepta i nema veze sa upravljanjem podataka drugih klasa. Na sličan način su i ostale klase zamišljene kao klase sa pojedinačnim odgovornostima.

2. Open Closed Principle – Otvoreno zatvoren princip

Ovaj princip je također zadovoljen u našem projektu. Klase su kreirane tako da ako ih nadograđujemo, nećemo biti u mogućnosti da ih suštinski mijenjamo, tačnije da mijenjamo njihovu ulogu. Unatoč tome, klase se mogu proširiti određenim funkcionalnostima, ali ne i izmijeniti. Također, koncept projekta je takav da je teško predvidjeti više situacija u kojima će biti potrebe za proširenjem. Uzmimo klasu Recept za primjer, sve informacije koje neki recept posjeduje trenutno, posjedovat će i kasnije

3. Liskov Substitution Principle – Liskov princip zamjene

Princip zamjene ćemo objasniti na primjeru apstraktne klase Osoba, te izvedenih klasa Korisnik, Admin, PremiumKorisnik i RegistrovaniKorisnik. Izvedene klase možemo zamijeniti klasama koje su im predci, a svaku izvedenu klasu možemo zamijeniti klasom Osoba. Klase RegistrovaniKorisnik i PremiumKorisnik možemo zamijeniti klasom Korisnik. Ovo je moguće iz razloga što su RegistrovaniKorisnik i PremiumKorisnik zapravo korisnici samog sistema.

4. Interface Segregation Principle – Princip izoliranja interfejsa

S obzirom na to da nijedna klasa nije zamišljena da bude debela klasa, odnosno da ima ogroman broj metoda , to je već dobar znak da bi ovaj princip trebao biti ispunjen. Klijent sistema će imati pristup samo onim metodama koje su mu zaista potrebne, odnosno klase su zamišljene tako da je svaka od njih cjelina koja predstavlja skup mogućih stvari koje klijent može da uradi. Jasan primjer su klasa Recept i PlanIshrane, jer iako su usko povezane, takve klase treba razdvojiti kao što i već jesu. Iako su PlanIshrane i Recept povezni, u smislu

da PlanIshrane sadrži recepte, klasa PlanIshrane nije dostupna izmjenama od strane klase Recept. Na taj način je to realizirano.

5. Dependency Inversion Principle – Princip inverzije ovisnosti

Za ovaj princip možemo reći da je također ispunjen, jer moduli visokog nivoa ne zavise od modula niskog nivoa. Tačnije, sve izvedene klase zavise od svoje apstraktne klase, ali ne vrijedi i obrnuto. Osoba nikako ne zavisi od izvedenih klasa, jer izmjene u tim klasama (Korisnik, PremiumKorisnik) nemaju veze sa klasom Osoba, s obzirom na to da su u njoj prisutne samo neophodne stvari, koje će uvijek morati biti tu, odnosno neće zavisiti od nekih promjena u izvedenim klasama.