SOLID principi u projektu

Ovdje su opisane promjene kod Model klasa. Kontroler i Pogled klase pravljene su u skladu s modelima

S - Single Responsibility Principle

KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU

U originalnom dijagramu neke klase su izvršavale previše zadataka i ovaj princip je time narušen. U novoj verziji dijagrama klasa ovaj princip je ispoštovan (barem do velike mjere). Izmjene koje su napravljene su sljedeće:

1. **Smještajna jedinica** je u starom dijagramu vodila računa o previše drugih stvari. Taj problem je sada riješen tako što su atributi smještajne jedinice sada samo osnovne stvari (poput **id**, **naziv** i **brojZvjezdica**), te **id** tip atributa koji je neophodan da bi se smještajna jedinica mogla povezati s ostalim klasama. Također, sve klase koje ovise o smještajnoj jedinici (**KontaktInfo, Recenzija, Upit, Usluga, Karakteristika, VrstaSobe**) posjeduju kao jedan atribut **idSJ** (id smještajne jedinice za koju su vezani). Te klase čuvaju odgovarajuće informacije o **Smještajnoj jedinici** umjesto da to ona radi sama. Klasa **Smještajna jedinica** je u početnoj verziji bila problematična zbog načina na koji se ima uvid u rezervisane sobe. Klasa **VrstaSobe** povezana je s odgovarajućom smještajnom jedinicom preko klase **SJSobe**. Zadatak da se posmatra broj slobodnih soba, kao i zauzimaju sobe, prepušten je toj klasi.
2. Slična stvar urađena je i za klasu Rezervacija, iz koje je nastala klasa **RezervacijaGost**, koja se brine o čuvanju informacija o gostima rezervacije, čime nestaje potreba klase **Rezervacija** da neposredno ima informacije o svojim gostima. Ova klasa takodjer povezuje gosta s odgovarajucom rezervacijom (u sistemu ce isti gost moci izvrsiti vise rezervacija)
3. Klasa **Uplata** se također brinula o previše stvari. Taj problem je riješen tako što klasa **Uplata** drži samo one attribute koji se neće mijenjati s vremenom (statički podaci). Dinamički podaci su migrirani u novu klasu **UplataStanje**, koja prati stanje uplate (ona se koristi kada korisnik uplati ratu). Preko ove klase, **Uplata** “zna” koliki dio je uplaćen.

Iako sve tri klase imaju po nekoliko metoda koje se pozivaju direktno iz te klase, autori projekta ne smatraju da je SRP narušen, barem ne do nedopustive mjere. Posto ce se na kraju raditi s bazom podataka, autori su nastojali da klase imaju sto slicniju strukturu kao atributi tabela iz BP (odnosno, uvedeni su **id**-evi koji bi bili primarni kljucevi, te **id**-evi za instance drugih klasa koji ce posluziti kao strani kljucevi)

O - Open Closed Principle

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

Jedna važna stvar koju raniji dijagram nije imao, a sada ima jesu interfejsi. Klase se mogu nadograđivati dodavanjem novih metoda interfejsa. Interfejsi koji su uvedeni su **Konvertor** (u biti, ovaj interfejs pretvara **int** u **Map<Integer, Boolean>**, na neki način radi konverziju iz decimalnog u binarne brojeve). Trenutno ova klasa ima dvije metode, koje same po sebi ne bi trebale da se modificiraju. Naravno, ukoliko autori tako procijene, mogu se dodati nove metode, koje samo treba implementirati u odgovarajućim klasama (nadogradnja). Slična stvar urađena je i za interfejs **PregledUpita**, koji je potreban i **Registrovanom Korisniku (samim tim i Administratoru)** i **Smještajnoj Jedinici.** Moguće je da će biti dodano još metoda, a dodane metode neće mijenjati svoju namjenu, niti način implementacije (mada autori još uvijek nisu implementirali metode). Kako se dalje sistem bude prosirivao na rad s bazom podataka, bit ce uvedeno jos interfejsa.

L - Liskov Substitution Principle

PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA

U našem sistemu postoji samo jedno nasljeđivanje, a to je nasljeđivanje klase **Adminstrator** iz klase **RegistrovaniKorisnik** (ovo je isto promjena u odnosu na prošlu verziju). Administrator je korisnik kao i svaki drugi, samo s još većim privilegijama. Administrator može rezervisati smještaj (nepraktično je da pravi obični account da bi to uradio). Dakle, **Administrator** je zamjenjiv klasom **RegistrovaniKorisnik**, čime je ovaj princip ispoštovan. Iz istog razloga (poštivanje ovog principa) klase **RegistrovaniKorisnik** i **SmještajnaJedinica** nisu naslijeđene iz neke općenitije klase (mada obje implementiraju interfejs **PregledUpita**), bez obzira što imaju neke slične atribute. Te klase su logički gledano previše različite i imaju drukčiji pogled na sistem i namjenu (recimo, smještajna jedinica ne može rezervisati smještaj, što korsinik može - jedan od razloga da ove dvije klase nemaju istu baznu klasu)

I - Interface Segregation Principle

KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI

Zbog ovog principa još uvijek vjerovatno nisu ni napisane sve metode, već samo one za koje su autori bili sigurni da su potrebne. Kako se sistem bude prosirivao, dodavat ce se metode u skladu s potrebama.

D - Dependency Inversion Principle

A. MODULI VISOKOG NIVOA NE BI TREBALI OVISITI OD MODULA NISKOG NIVOA.
OBA BI TREBALO DA OVISE OD APSTRAKCIJA.
B. MODULI NE BI TREBALI OVISITI OD DETALJA. DETALJI BI TREBALI BITI OVISNI OD
APSTRAKCIJA.

U našem sistemu nema apstraktnih klasa u pravom smislu te riječi, mada postoje dva interfejsa (a bit ce ih jos kako se sistem bude prosirivao). Medjutim, interfejsi sluze samo za nadogradnju vec postojecih klasa, sto je opisano u **O** dijelu. Interfejsi ne ovise od konkretnih klasa, vec obrnuto, odnosno konkretna klasa mora implementirati sve metode interfejsa ukoliko implementira interfejs.