Strukturalni paterni

1.Adapter patern se koristi prvenstveno radi proširenja upotrebe već postojeće klase. U našem projektu možemo ga iskoristiti u klasi “Profil” koja koristi podsjetnike. Adapter bismo koristili za sortiranje po “važnosti” tj. po datumu podsjetnika koji dolaze ranije od ostalih tako što ćemo preimenovati klasu Profil u ProfilAdaptee, te kreirati PodsjetnikAdapter koja koristi IPodsjetnik interfejs i ima mogućnost pozivanja njene metode sortByTerms koja bi vraćala sortirane podsjetnike pozivajući postojeću metodu klase ProfilAdaptee-getPodsjetnik(). Ovim bismo postigli širu upotrebu tj upgrade klase Profil bez ikakvih izmjena.

2. Facade patern se koristi kada želimo da klijentima “maskiramo” komplikovani sistem tj. da korisitmo samo mali dio funkcionalnosti velikog i kompleksnog sistema. Ovaj patern nećemo implementirati u naš projekat jer smog a bazirali na vrlo jednostavnim klasama i nemamo potrebu za korištenjem istog, no da smo umjesto klase ZahtjevZaUdomljavanje imali klasu UdomljavanjePsa koja bi vršila potpuni proces udomljavanja psa bilo bi poželjno uvrsitit ovaj patern tako što bi uvrstili i pomoćne interfejse (npr: ILokacija: mjesto gdje bi došli po psa kojeg želimo udomiti, ITermini: termini kada bi mogli doći da udomimo psa) i smjestili ih u class fasadu

3. Decorator patern koristimo kada ne želimo praviti veliki broj podklasa koje predstavljaju kombinacije postojećih klasa već želimo koristiti postojeće klase.

Ovaj patern možemo implementirati u naš sistem tako što bismo omogućili korisnicima koji imaju kreiran profil psa da uređuju fotografije koje dodaju koristeći filtere,rezanjem fotografije, izoštravanjem kvalitete i sl.

Ovaj patern se moze iskoristiti u našem projektu na način da se korisniku omogući dodavanje različitih sadržaja tj. različitih tipova sadržaja na forum, te različitih reakcija na same objave. Što se tiče naše aplikacije i primjene navedenih “dekoracija”, ona će za početak biti u mogućnosti izvršavati samo osnovne zahtjeve korisnika (dodavanje osnovnih sadržaja, standardnih tj. uobičajnih reakcija, te uređivanje fotografije neophodno za izvršavanje funkcionalnosti kao što je npr. rezanje slike na određeni, potrebni format..)

4. Bridge patern nam omogućava da se iste operacije primjenjuju nad različitim podklasama. Na taj način se vrši odvajanje apstrakcije I implementacije tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Najjednostavnija primjena u našem projektu bi bila apstrakcija slanja potvrde prilikom kreiranja profila ili prilikom izmjene lozinke. Da bi se ta radnja finalizirala potrebno je izvrsiti potvrdu u vidu primljenog e-maila ili SMS u kojem se nalazi odredjeni kod.

Također, funkcionalnost koja se veže uz klasu Podsjetnik, moze biti realizirana na više načina. Podsjetnik se treba kreirati automatski na osnovu zakazanih termina, ali i ukoliko ga korisnik sam “manuelno” pokuša kreirati.

5. Proxy patern

Osnovna ideja proxy paterna jeste zaštita pristupa resursima. Proxy je obično jedan mali javni surogat objekat koji predstavlja komplksni objekat cija aktivizacija se postiže na osnovu postavljenih pravila.

Prva primjena ovog paterna jeste prilikom prijave korisnika na profil. Naime da bi korisnik pristupio aplikaciji i svome profile korisnik se

mora prijaviti tj. unijeti password za svoj username tj profil. Nakon toga se vrši autentifikacija korisnika i na taj način je zastićen pristup.

Također, da bi korisnik mogao učestvovati u diskusijama na forumu i komentarisati postove drugih korisnika, on mora biti registrovan. Provjerom tipa tj. klase kojoj korisnik pripada omogućit ce se ogranicen pristup diskusijama na forumu.

6. Composite patern

Composite patern omogucava formiranje strukture stabla pomocu klasa, gdje se listovi stabla I korijeni stable jednako tretiraju.

Cilj je da kroz zajednicki interfejs koristimo na bilo koji nacin objekte nad kojima vrsimo slicne akcije.

Mislimo da bi primjena ovog paterna na nasem projektu bila sljedeca: U nasem forumu postoje razne sekcije (hrana, zabava, trening…). Jedno pitanje se moze naci u vise sekcija, jer moze obuhvatiti vise tema. Korisnik moze postaviti/odgovoriti na pitanje, te izbrisati vlastito pitanje/odgovor.

7. Flyweight patern

Osnovna ideja flyweight paterna je postojanje dijeljenog objekta koji se koristi kao potpora velikom broju drugih objekata. Postojanje dijeljenog objekta, iako dovodi do odredjenog usporavanja izvodjenja programa, primarno smanjuje troškove memorije.

U nasem projektu, ustedu memorije mozemo ostvariti prilikom trazenja veterinarskih stanica. Aplikacija pamti sve stanice koje su se kucale u trazilici, te tako imamo bržu pretragu.

Mislile smo da se usteda memorije moze ostvariti I na forumu. Kada korisnik dodaje sliku, pitanje ili dokument, u slucaju da je to vec dodano od strane drugog korisnika (dakle, postoji u nasem sistemu), iskocit ce mu notifikacija “Pitanje/slika/dokument vec postoji”. Nazalost, ne znamo kako bi se znalo da pitanje/slika/dokument vec

postoji. U slucaju da korisnik A postavi sliku pod nazivom “konj”, a korisnik B istu tu sliku pod nazivom “konj1”, onda bi aplikacija to smatrala razlicitim slikama, iako su one iste. Pitanje “Kojim samponom kupati psa?” I “Kakav sampon je najbolji za pse?” traze iste odgovore, ali bi ih aplikacija prepoznala kao razlicita pitanja.