Strukturalni paterni

- BRIDGE PATTERN

  Ovaj patern, kao jedan od ključnih paterna u našoj aplikaciji, bi mogao biti iskorišten u različitom prikazivanju potencijalnih partnera za registrovanog korisnika i VIP korisnika. Kako je zamišljeno aplikacijom, registrovani korisnik bi imao prikaz jednog idealnog partnera, dok bi VIP korisniku bio dozvoljen prikaz liste partnera sa najvećim procentom kompatibilnosti izračunatim od strane sistema.

- COMPOSITE PATTERN

  Composite pattern služi za omogućavanje kreiranja hijerarhije objekata čime bi postigli objekte kojima se pristupa na isti način, a koji različito implementiraju određene metode.
  Ovo omogućava formiranje strukture stabla u kojoj se kompozicije individualnih objekata (korijen) i individualni objekti (list) tretiraju ravnopravno, što znači da je moguće pozivati zajedničku metodu nad svim klasama. U našoj aplikaciji ovaj patern bi mogao biti iskorišten za omogućavanje korisniku da postoje dvije različite vrste klase Pitanje namijenjene za popunjavanje atributa Obrazac klase RegistrovaniKorisnik.
  Jedna vrsta pitanja imala bi enum odgovore isključivo DA i NE (odnosno binarni odgovori), a na drugu vrstu pitanja bi se moglo odgovoriti skalom od 1 do 5. U tom slučaju došlo bi do potrebe za stvaranjem klase Pitanje u vidu korijena stabla, kao i klasa BinarnoPitanje i SkalaPitanje u vidu listova stabla. Na ovaj način lista Obrazac bi mogla primati obje ove vrste pitanja.

- ADAPTER PATTERN

  Kako svaki registrovani korisnik ima pravo da ostavi svoj komentar i ocjenu za rad aplikacije, te ima uvid u sve ostale recenzije, adapter patternom se može omogućiti da korisnik odabere sortirani prikaz svih recenzija sa proizvoljno odabranom ocjenom. Također, moguće bi bilo odabrati prikaz recenzija počevši od najstarije,ili u određenom vremenskom periodu.

- FLYWEIGHT PATTERN

  U aplikaciji je omogućena chat interakcija između dva registrovana korisnika. Flyweight patern bismo iskoristili kako bi korisnik, pored pisane poruke, mogao poslati i druge vrste implementiranih poruka, kao što su fotografije, videi ili emotikoni.Implementacija unutar naše aplikacije bi se izvršila tako što bi se na klasu Poruka povezao interfejs IPrepoznajPoruku za raspoznavanje različitih poruka, na koji bi bile povezane sve klase za različite vrste poruka, kao recimo klasa GIF. Svaka od tih klasa bi bila određena sa specifičnim atributom koji bi bio zatražen od strane interfejsa.

- PROXY PATTERN

  Proxy pattern služi za onemogućavanje neispravne i/ili zloupotrebe objekata. Sistem je zaštićen od neželjenih ponašanja zahvaljujući kontrolisanju pristupa i manipulacije nad objektima ukoliko nisu svi uslovi za to ispunjeni. U našoj aplikaciji ovaj patern bi mogao biti iskorišten za osiguravanje tačnosti informacija vezanih za statističke podatke o procentu sklopljenih brkova putem aplikacije love algorithm.
  Ovo bi bilo postignuto objektom koji vrši verifikaciju o obostranom podnošenje zahtjeva za ažuriranje statusa braka korisnika jednog sa drugim. Ovako iskorištenim proxy paternom postigli bismo provjerene i sigurne podatke, što bi značilo vjerodostojnost statističkog opisa efikasnosti same aplikacije.

- DECORATOR PATTERN

  Decorator pattern služi za omogućavanje dinamičkog dodavanja novih ekemenata i funkcionalnosti već postojećim objektima. Na ovaj način nadograđuje se jedna vrsta objekta što bi inače podrazumijevalo veliki broj izvedenih klasa. "Dekorirani" objekat ne zna za svoje nadograđivanje, čime je omogućena njegova naknadna upotreba. Ovaj patern se koristi kada za cilj imamo dodavanje novih ponašanja objektu za vrijeme izvođenja programa, a da ne moramo vršiti izmjene u kodu koji će biti u daljnoj interakciji sa tim objektom.U našoj aplikaciji ovaj patern bi mogao biti iskorišten za omogućavanje korisniku da vrši

adaptacije nad upload-ovanom slikom profila u vidu rezanja i rotacije.

- FACADE PATTERN

  Kako unutar naše aplikacije imamo mnogo zahtjeva koje jedan registrovani korisnik može poslati(zahtjev za promjenu slike, zahtjev za promjenu lozinke, zahtjev za pristup chatu itd.), možemo napraviti određenu klasu ZahtjevFacade koja će objediniti sve moguće zahtjeve koje korisnik može uputiti. ZahtjevFacade bi u dijagramu bio povezan sa klasom RegistrovaniKorisnik, interfejsom IProfil, te klasom Chat.