STRUKTURALNI PATERNI

**Adapter pattern**

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. U situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu koristi se Adapter patern. Adapter patern kreira novu adapter klasu koja služi kao posrednik između orginalne klase i željenog interfejsa.  
Adapter patern bi mogli iskoristiti kada bi trenutni tip atributa Slika koji je sada string zamijenili tipom JPG ili PNG. Tada bi uz pomoć metoda konverzija Adapter klase omogućili prihvatanje raznih formata slike.

**Facade pattern**

Kako mu i sam naziv govori, ovaj patern se korisiti s ciljem da osigura više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika).   
Facade patern bi mogli iskoristiti kada bi kreirali novu klasu PutovanjeFacade koja bi sadržavala samo atribute id Mačke, ime, vjerovatnoću odabira i id putovanja kako bi smanjili kompleksnost rada sistema pri odabiru mačaka za putovanje.

**Decorator pattern**

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.   
U našoj aplikaciji bismo dodati mogućnost editovanja slika kao što su rotacija, rezanje i dodavanje filtera koji postavlja astronautsku kacigu na profilnu sliku mačke.

**Bridge pattern**

Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije.   
Za svaku mačku trebamo izračunati vjerovatnoću odabira. Ta vjerovatnoća je različita ukoliko je mačka free ili premium korisnik. I Mačka i Premium imaju različitu implementaciju iste metode izracunajVjerovatnocu.

**Composite pattern**

Composite patern služi za kreiranje hijerarhije objekata. Koristi se kada svi objekti imaju različite implementacije nekih metoda, no potrebno im je svima pristupati na isti način, te se na taj način pojednostavljuje njihova implementacija.   
Ovaj patern možemo iskoristiti ako želimo administratoru dozvoliti pregled svih profila mačaka koje su registrovane u aplikaciji uz podjelu na free i premium. Ova lista bi se dobila pozivom nove metode dajSveMacke.

**Proxy pattern**

Proxy patern služi za dodatno osiguravanje objekata od pogrešne ili zlonamjerne upotrebe. Primjenom ovog paterna omogućava se kontrola pristupa objektima, te se onemogućava manipulacija objektima ukoliko neki uslov nije ispunjen, odnosno ukoliko korisnik nema prava pristupa traženom objektu.   
U našoj aplikaciji ovaj patern koristimo kod unosa lozinke prilikom   
login-a. Autentifikacijski proxy prije proslijeđivanja pristupa profilu prvo pita za password, te u slučaju dobro unešenog passworda odobrava pristup mogućnostima registrovanog korisnika, dok ga u suprotnom onemogućuje.

**Flyweight pattern**

Flyweight patern koristi se kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji svi u suštini predstavljaju jedan objekat. Samo ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama (tzv. specifično stanje), vrši se njegova instantacija, dok se u svim ostalim slučajevima koristi postojeća opća instance objekta (tzv. bezlično stanje).   
Ovaj patern možemo iskoristiti u komentarima kada bi željeli omogućiti korisniku korištenje emoji-a ili fotografija. Dodali bismo klase Emoji i Slika te metodu dajDodatak koja se poziva pri unosu u oba slučaja.