DIZAJN PATERNI PONAŠANJA

1. ***Strategy*** – izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Možemo ga iskoristiti prilikom različitih načina pretrage nekretnina (npr. po cijeni, kvadraturi, lokaciji, broju soba). Umjesto da bude jedna velika metoda u klasi, strategije pretrage se mogu izdvojiti u posebne klase i dinamički primjenjivati.

2. ***State*** – mijenja način ponašanja objekata na osnovu trenutnog stanja. U našem sistemu ovaj patern možemo upotrijebiti kod postavljanja različitih stanja oglasa za nekretnine (npr. „Na čekanju“, „Aktivan“, „Odbijen“ i slično). Ponašanje oglasa se može mijenjati u zavisnosti od trenutnog stanja, recimo dopušta ili ne dopušta izmjene.

3. ***TemplateMethod*** – omogućava algoritmima da izdvoje pojedine korake u podklase. Primjena ovog paterna ponašanja unutar našeg sistema može biti kod verifikacije i objave oglasa, gdje postoji definisan redoslijed koraka. To može biti sljedeći: validacija, provjera korisnika, unos slika, itd. Međutim, neki koraci se mogu redefinisati u podklasama za različite tipove korisnika. U našem slučaju pristup je različit za agenta i admina.

4. ***Chain of Responsibility*** – predstavlja listu objekata, ukoliko objekat ne može da odgovori prosljeđuje zahtjev narednom u nizu. Moguće ga je koristiti kod obrada zahtjeva za podršku gdje zahtjev ide kroz lanac zaposlenih. Prvo kreće od vlasnika, pa do agenta, te na samom kraju stoji admin. Zahtjev prolazi lancem sve dok neko ne odgovori.

5. ***Command*** – razdvaja klijenta koji zahtijeva operaciju i omogućava slanje zahtjeva različitim primaocima. Najbolji primjer primjene ovog paterna je za akcije korisnika nad oglasima (dodaj, obriši, izmijeni, aktiviraj...) koje treba da se sačuvaju, izvrše ili poništе na kraju. Koristan je i za undo/redo funkcionalnosti.

6. ***Iterator*** – omogućava pristup elementima kolekcije sekvencijalno bez poznavanja interne strukture kolekcije. Možemo ga koristiti za navigaciju kroz listu nekretnina. Naprimjer, pri prikazu rezultata pretrage ili listanja po kategorijama, bez obzira na implementaciju liste.

7. ***Mediator*** – enkapsulira protokol za komunikaciju među objektima dozvoljavajući da objekti komuniciraju bez međusobnog poznavanja interne strukture objekta. Mogli bi ga efikasno iskoristiti za komunikaciju između više UI komponenti. Idealan primjer za to je situacija kada korisnik promijeni filter da se automatski ažurira prikaz rezultata pretrage. Sve se to odvija na način da komponente međusobno ne znaju jedna za drugu.

8. ***Observer*** – uspostavlja relaciju između objekata takvu da kad se stanje jednog objekta promijeni svi vezani objekti dobiju informaciju. Unutar našeg sistema Observer patern se odlično uklapa u funkcionalnost slanja obavještenja pri pojavi odgovarajuće nekretnine. Kada korisnik postavljanjem određenih kriterija zahtijeva prijem notifikacije za nove oglase, sistem ga obavještava čim se pojavi nova nekretnina koja odgovara njegovim zahtjevima.

9. ***Visitor*** – definira i izvršava nove operacije nad elementima postojeće strukture ne mijenjajući samu strukturu. Možda bismo ga mogli iskoristiti ukoliko želimo nad oglasom izvršiti određene akcije koje ne mijenjaju samu strukturu oglasa. To može biti provjera pravopisnih grešaka.

10. ***Interpreter*** – podržava interpretaciju instrukcija napisanih za određenu upotrebu. Recimo da na osnovu statistike oglasa postavimo *uslove brojPregleda > 1000 AND brojKlikova > 100*, sistem može koristiti interpreter da bi prepoznao kad oglas ispunjava uslove za isticanje – prelazak u „featured“ status.

11. ***Memento*** – omogućava spašavanje internog stanja nekog objekta van sistema i njegovo ponovno vraćanje. U našem sistemu ovaj patern se može koristiti za čuvanje prethodnog stanja oglasa. Kada agent uređuje oglas, sistem može sačuvati prethodno stanje koje se može vratiti ako korisnik klikne na dugme „poništi izmjene“ ili u slučaju greške.