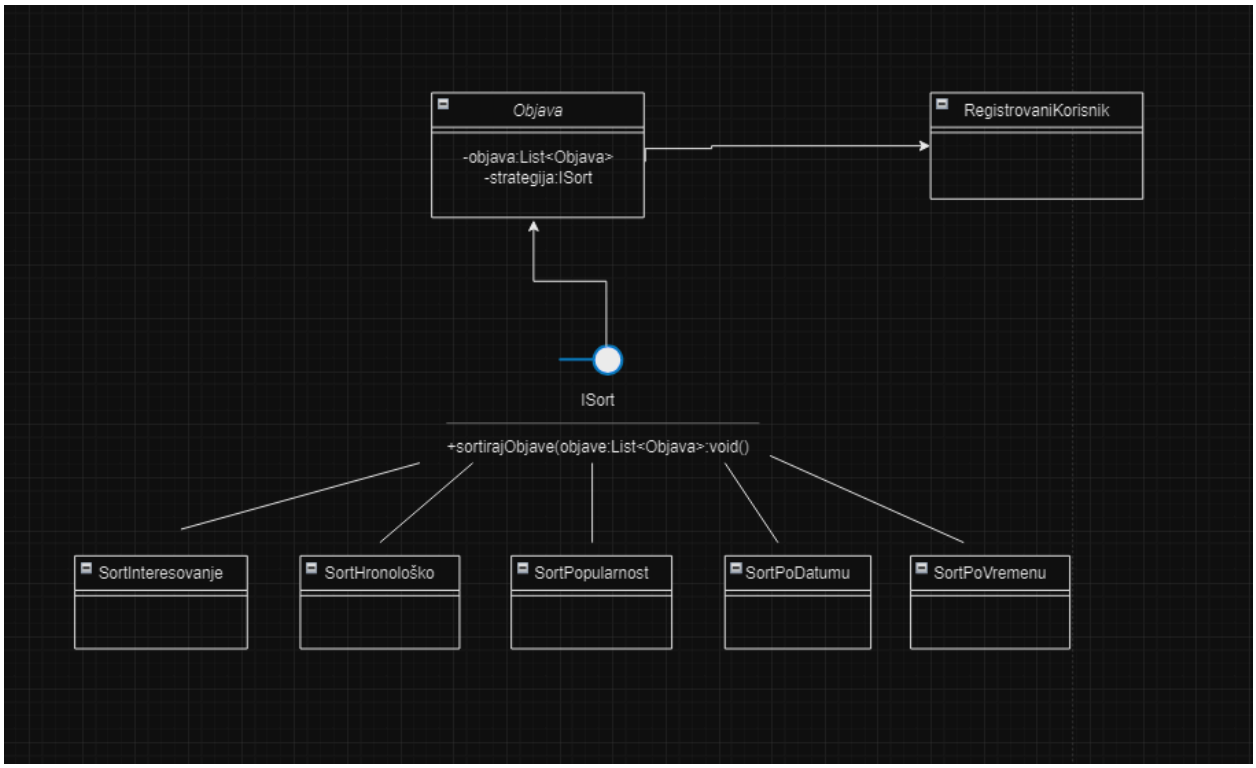
Paterni ponašanja

Strategy patern

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Pogodan je kada postoje različiti primjenjivi algoritmi (strategije) za neki problem. Strategy patern omogućava klijentu izbor jednog od algoritma iz familije algoritama za korištenje.

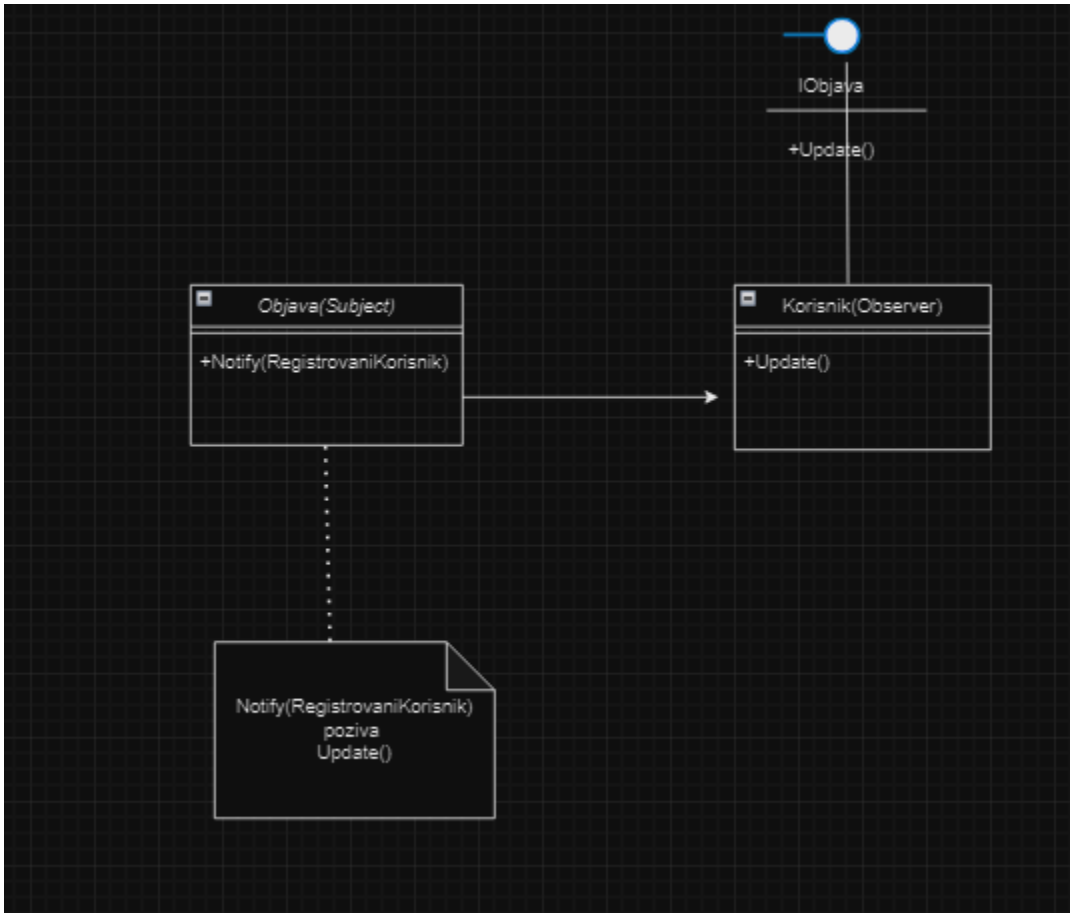
U našem sistemu bismo Strategy patern mogli iskoristiti tako da prethodno RegistrovanomKorisniku omogućimo da sortira objave. Objave bi se mogle sortirati na više načina (hronološko sortiranje, sortiranje prema popularnosti ili prilagođeno sortiranje bazirano na interesovanjima korisnika). Conext klasa bi za atribut imala listu objava i strategiju sortiranja.



Također, ovaj patern možemo dodati u naš sistem na sljedeći način. Poznato nam je da unutar našeg sistema imamo 3 vrste korisnika: admin-a, modeatora, te samog klijenta. Stoga prilikom samog prijavljivanja korisnika, imamo sigurno 3 opcije prilikom registracije. Detekcija o kojem od tri nabrojana korisnika se radi, vrši se na osnovu ulaznih podataka. Na osnovu datog tipa klase tj. korisnika mogli bismo realizovati 'setup' nakon registracije u kojem bi za svaki tip korisnika postojao drugačiji prikaz (znamo da ova tri krosnika mogu pristupiti i vidjeti različite stvari), samim tim možemo i drugačije podatke dobiti pomoću api request-a prema našoj bazi.

Observer patern
Uloga Observer paterna je da uspostavi relaciju između objekata tako kada jedan objekat promijeni stanje drugi zainteresirani objekti se obavještavaju.

Ovaj patern smo primjenili u našem sistemu tako što korisnik dobije notifikaciju (obavještenje) kada drugi korisnik ima neku interakciju sa njegovim profilom.
Također, ovaj patern možemo dodati u naš sistem na sljedeći način. Poznato je da sistem posjeduje funckionalnost koja omogućava klijentu da ažurira svoj ptofil. Kada korisnik objavi novi post ili komentar, svi korisnici bivaju obavešteni i njihov feed se ažurira.
Također, obzirom da ćemo observer patern dodati na način da ćemo omogućiti administratoru ili zaposleniku da svakoga puta kada bilo koji klijent ažurira svoj korisnički profil, da on dobije obavijest o tome. Tako će biti u stanju da vidi ažurirane informacije.



State patern
State patern je dinamička verzija Strategy paterna i omogućava promjenu načina ponašanja objekta u zavisnosti od stanja u kom se nalazi. Za razliku od Strategy paterna, objekat sam mijenja svoje stanje, ono nije izabrano od strane klijenta.
Recimo da želimo omogućiti korisnicima da žele svoj profil učiniti privatnim. Tada bi klasa Korisnik imala dva mode-a: privatni i javni, odnosno ovo bi bile klase koje implementiraju

interface IMode. U zavisnosti od klase, odnosno mode-a bi omogućili drugim korisnicima prikaz profila tog korisnika, odnosno poziv metode prikaziProfil klase PrijavljeniKorisnikController.

Iterator patern
Iterator patern omogućava sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija struktuirana. U našem sistemu ovaj patern možemo primijeniti tako što ćemo omogućiti korisnicima da biraju redoslijed prikaza korisnika koji su im lajkali fotografiju ili koji su komentarisali njihovu objavu.

Medijator patern
Medijator patern enkapsulira protokol za komunikaciju među objektima dozvoljavajući da objekti komuniciraju bez međusobnog poznavanja interne strukture objekta.
Ovaj patern možemo koristiti u našoj aplikaciji, ali možemo proširiti sistem da imamo više vrsta registrovanih korisnika. Mogli bi dodati Premium i VIP korisnike u naš sistem. Korisnici da bi stupili u kontakt mogu koristiti neki vid chat platforme. U našem slučaju bi bez ograničenja komunicirati Premium i VIP korisnici dok standardni korisnici ne bi imali sve mogućnosti.

Chain of responsibility patern
Chain of responsibility omogućava razdvajanje jednog kompleksnog procesa obrade na više objekata koji na različite načine procesiraju primljene podatke. U našem sistemu bismo mogli primijeniti ovaj patern kada bismo omogućili korisnicima da Chain of responsibility da biraju da li žele prihvatiti zahtjev za praćenje.

Template method patern
Omogućava izdvajanje određenih koraka algoritma u odvojene podklase. Struktura algoritma se ne mijenja - mali dijelovi operacija se izdvajaju i ti se dijelovi mogu implementirati različito.

U našem sistemu ovaj patern možemo primijeniti na način da omogućimo primjenu više algoritama pri pretraživanju korisnika. Tako bismo omogućili da se pretraženi korisnici sortiraju na različite načine (dob, broj followera…)