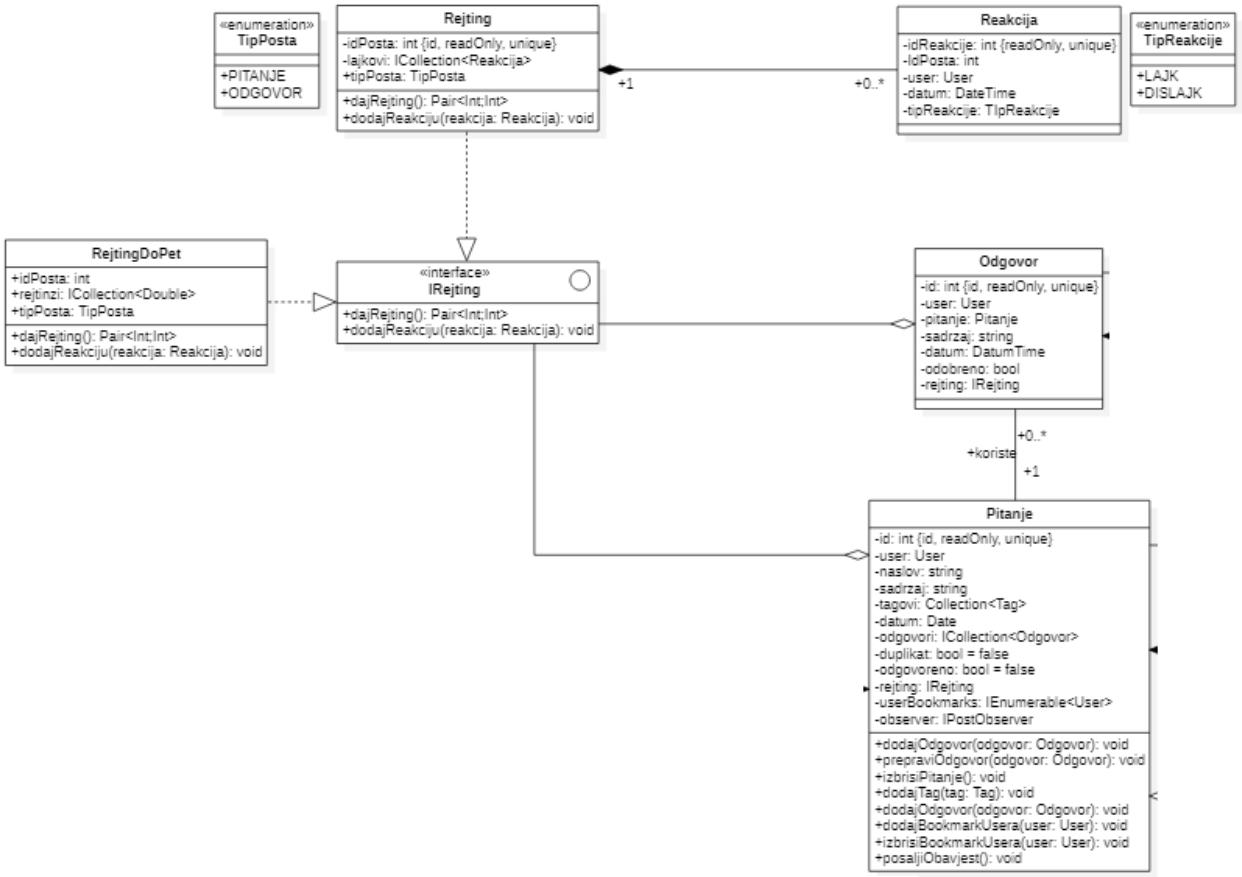
PATERNI PONASANJA

1. ***STRATEGY PATERN:***

Strategy pattern se zasniva na ideji da mozemo imati familiju algoritama koji su međusobno zamjenjivi, te u zavisnosti od potrebe sistema mozemo lagano izmjenuti jedan algoritam sa drugim.
Ovaj pattern smo iskoristili kod dijela programa zaduzenog za racunanje rejtinga nekog posta. Nama kontekst klasu predstavljaju klase Pitanje i Odgovor, strategiju nam predstavlja interfejs IRejting, a implementacija je Rejting. Ovdje je povoljno iskoristiti ovaj patern iz razloga ako budemo htjeli prosiriti aplikaciju na nacin da imamo na razlicite nacine racunanja rejtinga, npr ako bi htjeli da rejting nije samo lajk i dislajk nego i dodatno od nula do 5 i slicno.



2. ***STATE PATERN:***

State pattern se koristi kada imamo objekat koji se ponasa razlicito u zavisnosti od njegovog trenutnog stanja. Ovaj patern je najkorisniji ukoliko objekat ima veliki broj stanja.

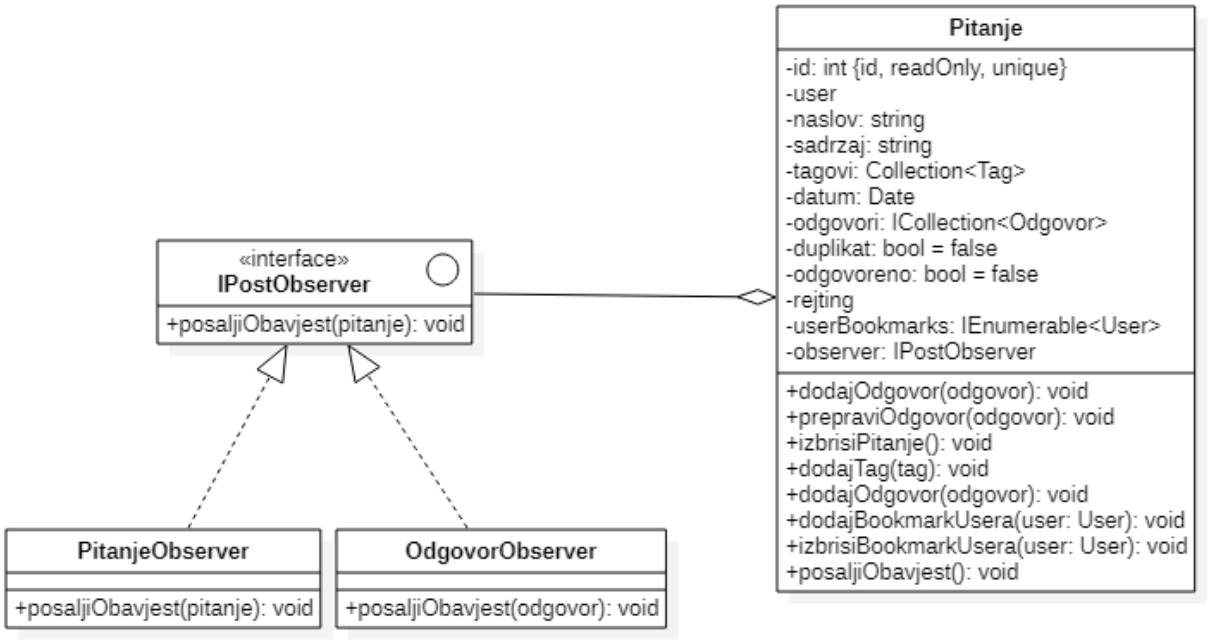
Ovaj patern bi se mogao iskoristiti u dijelu programa zasluzenog za prikaz odgovora na pitanje. Naime, kada je pitanje u stanju neodgovorenog nacin na koji se prikazuju odgovorio bi bio da odgovori koji imaju najvise lajkova budu pri vrhu, a kada pitanje pređe u stanje odgovorenog, samim tim ako je pitanje odgovoreno mora postojati barem jedan prihvacen odgovor, sada bi se prikaz odgovora radio na nacin da se prihvaceni odgovori, u zavisnosti od vremena postavljanja, stavljaju pri vrhu dok ostali opet prema broju reakcija.

3. ***TEMPLATE PATERN:***

Template patern se koristi ukoliko imamo vise algoritama koji rade slicne stvari i koji se međusobno razlikuju bilo po redoslijedu obavljanja koraka ili po broju. Ovaj patern govori da se svi koraci koji se ponavljaju između tih algoritama izdvoje u jednu apstraktnu klasu koja sadrzi zajednicke korake u tim algoritmima, te pojedinacni algoritmi mogu proizvoljno implementirati metode te klase prema vlastitim potrebama.
Ovaj patern bi mogli koristiti u slucaju ako bi smo imali neki oblik sorting algoritma, te na osnovu njega pravili algoritme za sortiranje liste objekata Pitanje i Odgovor, s tim da bi se koritstili ti pojedini koraci iz glavne klase za sortiranje.

4. ***OBSERVER PATERN:***

Observer patern se koristi kada vrijedi da promjena stanja jednog objekta povlaci za sobom da se trebaju promenuti i drugi objekti, a broj tih drugih objekata je u pocetku nepoznat.
Ovaj patern cemo iskoristiti u okviru sistema za slanje notifikacija korisnicima, tacnije u slucaju postavljanja odgovora na pitanje. Kako ce sem korisnika i drugi korisnici imati pravo spasiti neka pitanja kao bookmarks, te prilikom dodavanja novog odgovora na pitanje ce se slati notifikacija svim korisnicima koji imaju to pitanje u bookmarks. Subjekt klasa u nasem slucaju bi bila klasa Pitanje, imali bi interfejs IPostObserver koji ima metodu Update koja ce se pozvati prilikom svakog dodavanja novog pitanja, dok korektna implementacija ovog interfejsa bila PitanjeObserver, a slican slucaj bi imali kod OdgovorObserver, tj korisnika bi notifikovali kada se desi neka promjena.



5. ***ITERATOR PATERN:***

Iterator patern se koristi kada je potrebno napraviti neki nacin prolaska kroz neku kolekciju objekata s tim da je u potpunosti nepoznata građa takve kolekcije.
Iterator patern bi mogli koristiti u kontekstu pitanja. Ako bi smo napravili poseban konternjer za skup odgovora na nekom pitanju koji radi po principu heap-a ciji se elementi redaju na osnovu broja reakcija i prihvacenosti odgovora. Potom ako bi dodali iterator u tu kolekciju koji bi kretajuci samo radio breadth-first search.

6. ***MEMENTO PATERN:***

Memento patern omogucava da se vrsi stasavanje i povratak trenutnog stanja nekog objekta bez da pokazemo nacin na koji se taj objekat implementira.
Ovaj patern bi se mogao iskoristiti kod postavljanje pitanja, tj.ukoliko bi omogucili da prilikom postavljanja pitanja korisnik izađe iz datog obrasca da se promjene koje je vec napisao ne izbrisu nego spase pa kada ponovo bude ulazio u taj obrazac da budu uneseni ti podaci.

7. ***VISITOR PATERN:***

Visitor patern se koristi kada zelimo odvojiti algoritam od objekata na kojima se on vrsi.

Ovaj patern bi iskoristili u algoritmu za preporuku, te tada bi bilo svejedno da li se algoritam koristi nad tagovima ili pitanjima.