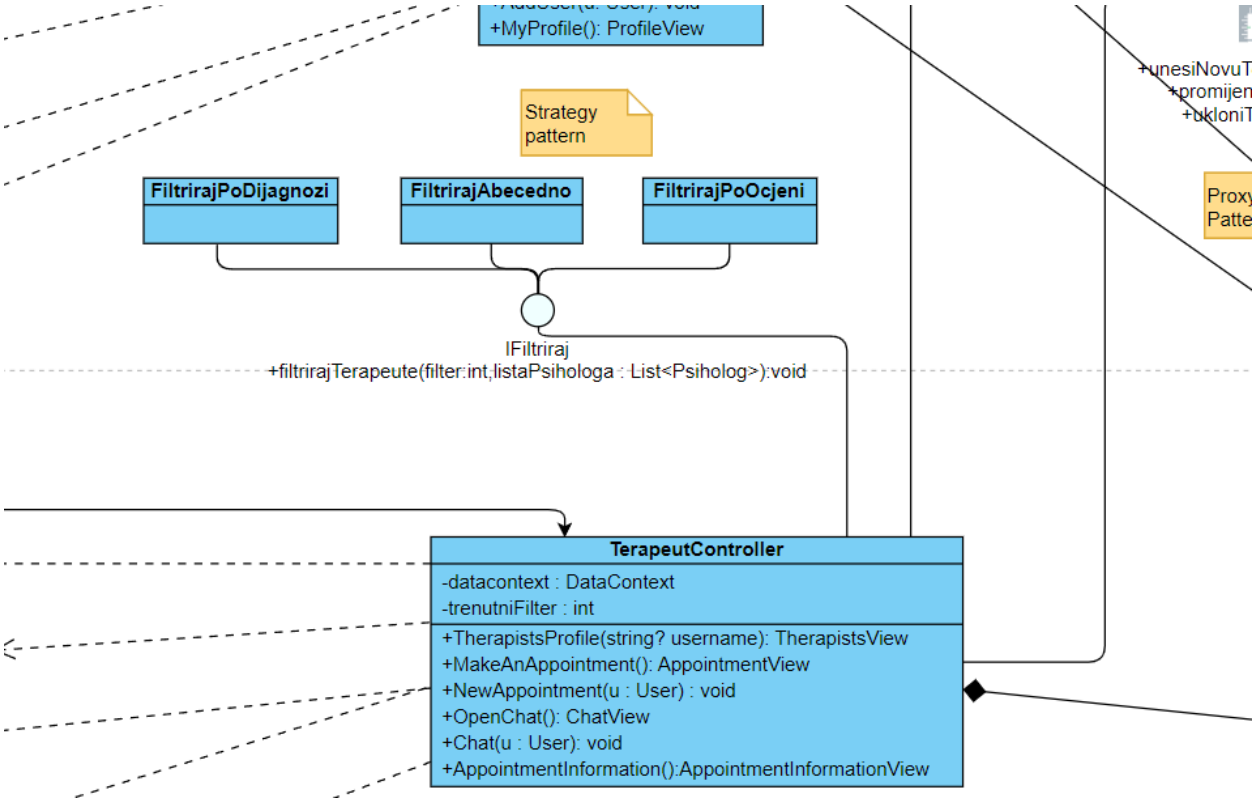
1. **Strategy pattern**

Strategy pattern se koristi za ukidanje velikog broja if-else blokova i uvođenje hijerarhije klasa kada se samo razlikuje način izvršavanja nekog algoritma.

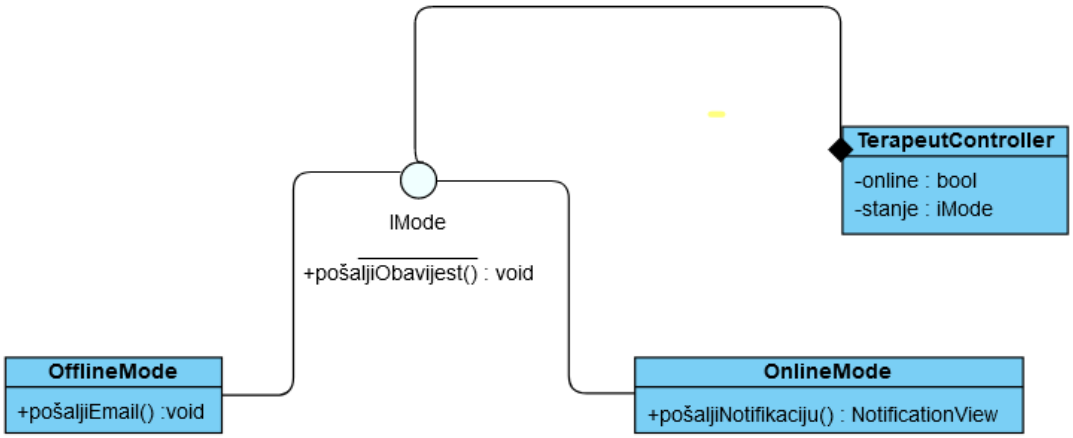
Što se tiče našeg sistema, mi smo implementirali ovaj pattern kod filtriranja psihoterapeuta. Uveli smo tri nove klase i interfejs IFIltriraj, pri čemu TerapeutController implementira pomenuti interfejs na sljedeći način:



2. State pattern

State pattern se, između ostalog, koristi za izdvajanje koda koji je smješten u velike if-else blokove, ovisno o vrijednosti neke varijable.

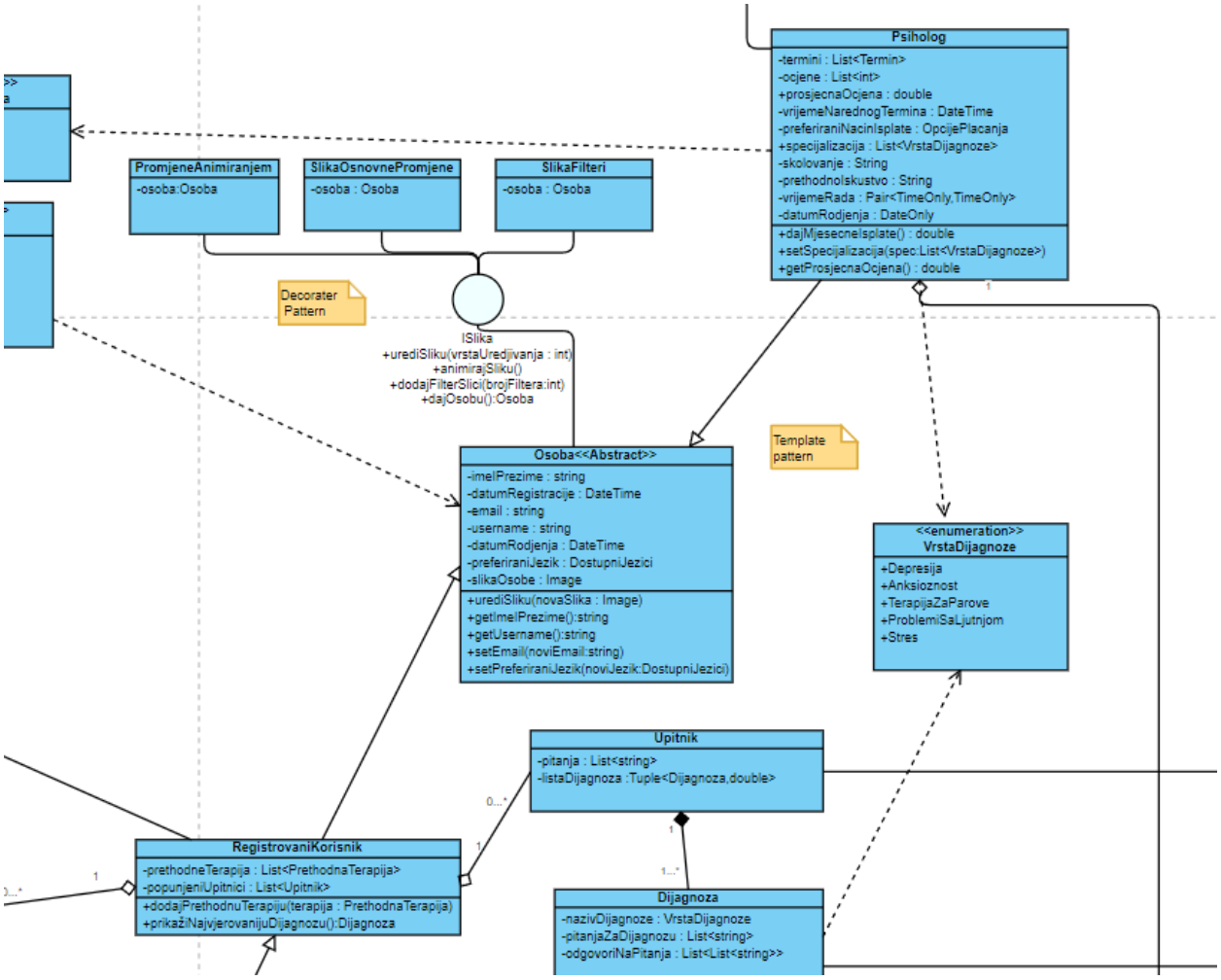
U našem sistemu, ovaj pattern smo iimplementirali kod provjeravanja stanja psihoterapeuta. Ukoliko korisnik pošalje zahtjev za termin, a psihoterapeut je istovremeno online, njemu će stići notifikacija o tome da korisnik želi zakazati termin. Ako je psihoterapeut offline, njemu će stići mail.



3. Template pattern

Template pattern se koristi kod omogućavanja promjena samo dijela algoritma i korištenje preostalog dijela koda.

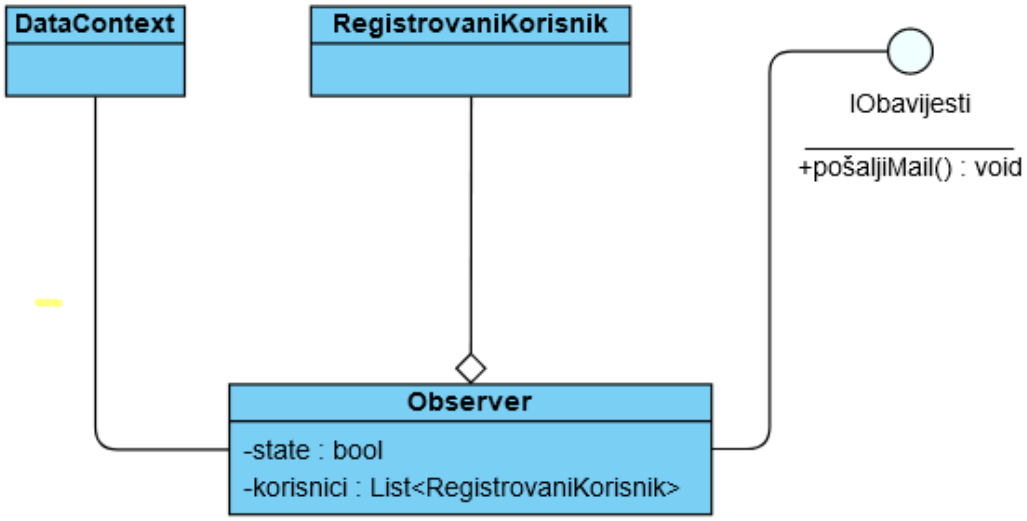
Ovaj pattern smo mogli implementirati kod apstraktne klase Osoba i njene izvedene klase Psiholog na sljedeći način:



4. Observer pattern

Ovaj pattern omogućava propagiranje promjena sa jednog objekta na druge objekte. On također omogućava automatsko pozivanje metoda drugih klasa pri pojavi nekog događaja.

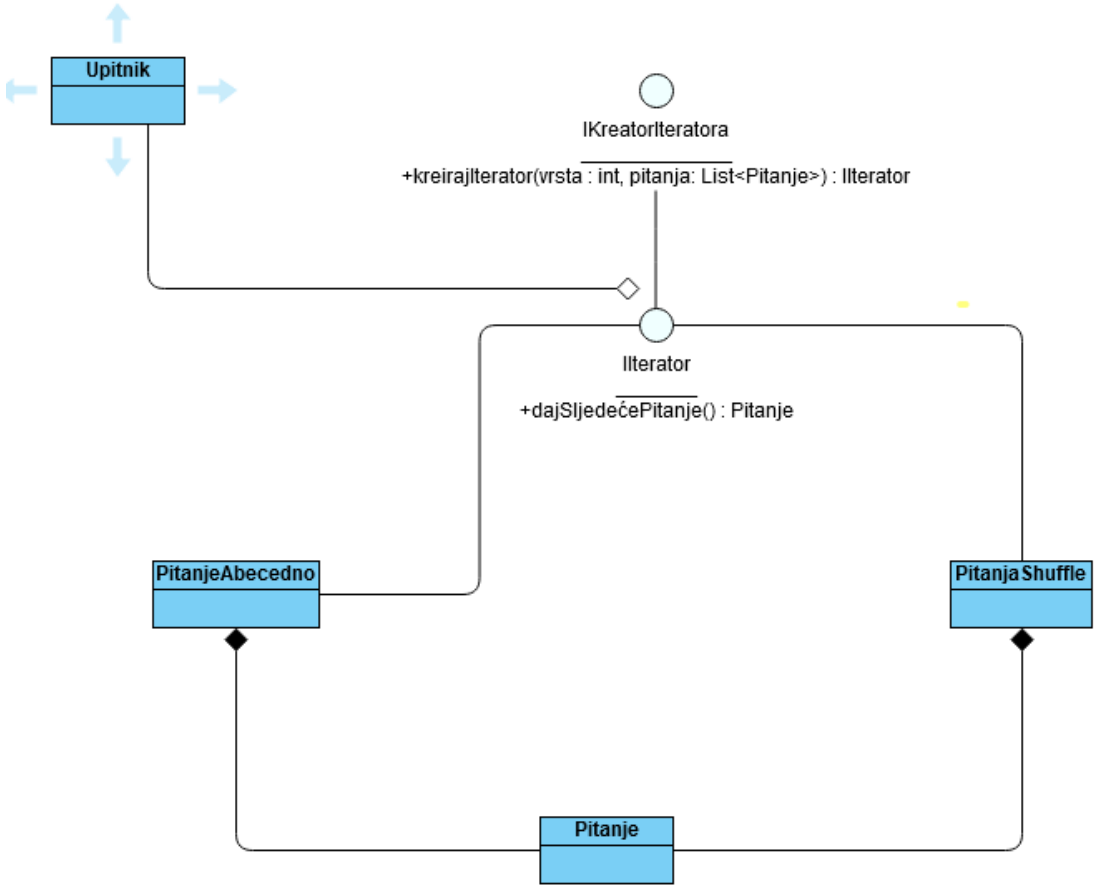
Što se tiče potencijalne implementacije kod našeg sistema, mi smo predvidjeli mogućnost da kada se registruje psiholog koji je specijaliziran za jednu vrstu dijagnoze, da se korisnicima kojima je to najvjerovatnija dijagnoza pošalje obavještenje da postoji novi registrovani psiholog koji je specijaliziran za njihov problem.



5. Iterator pattern

Iterator pattern se koristi kod sakrivanja kompleksnog algoritma za prelazak na naredni element od korisnika. On također omogućava prolazak kroz kolekciju objekata koja nije niz niti lista prema željenoj logici.

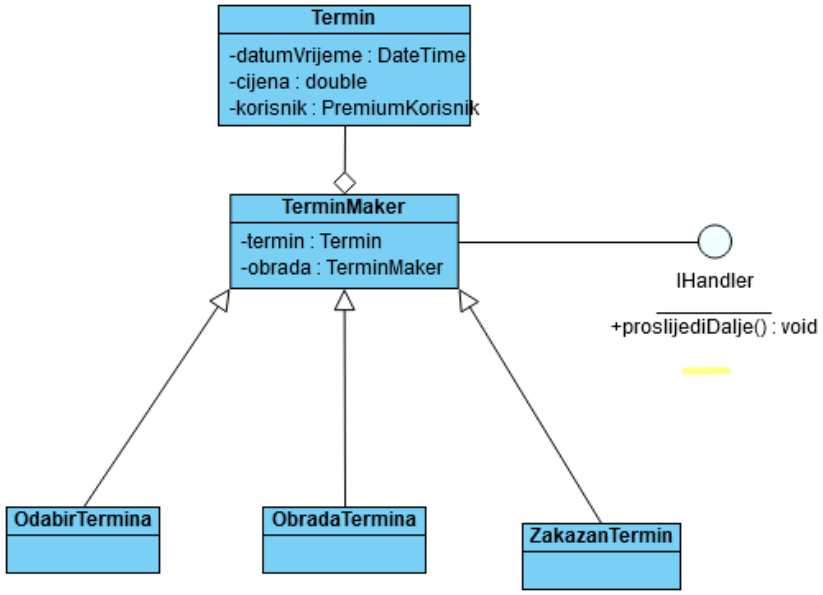
Što se tiče našeg sistema, ovaj pattern je moguće implementirati kod upitnika sa listom pitanja. Na taj način bismo mogli implementirati različit prikaz pitanja korisnicima.



6. Chain of responsibility

Ovaj pattern se koristi kod primjena procesa u kojima je važno definisati korake i njihov redoslijed. On također služi za razdvajanje dijelova procesa tako da se mogu neovisno mijenjati.

Sljedeći isječak prikazuje potencijalnu implementaciju ovog patterna. Prije nego što finalizira zakazivanje termina, korisnik mora odabrati termin, slijedi odobravanje termina od strane psihoterapeuta nakon čega korisnik mora uplatiti termin i nakon toga dobija potvrdu o zakazanom terminu.



7. Mediator pattern

Mediator pattern se koristi kod smanjivanja direktnih veza između klasa te za kreiranje centralizovanog sistema.

Ovaj pattern bismo mogli implementirati kod provjeravanja sadržaja poruka u cilju otkrivanja malicioznih poruka. Na taj način, administrator će putem interfejsa IMedijator provjeravati poruke između psihologa i registrovanog korisnika.