

PATERNI PONAŠANJA

1. STRATEGY PATTERN

Koristit ćemo ovaj patern u RezervacijaDogađajaController-u u kojem ćemo implementirati metodu Sortiranje(tipSortiranja: int). Implementirat ćemo interfejs ISortiranje kojeg će nasljeđivati klase SortirajPoOgraničenju(), SortirajPoDatumu(), SortirajPoTipu().

2. STATE PATTERN

Ovaj patern možemo iskoristiti prilikom različitih načina prikazivanja događaja, zavisno od toga da li je korisnik prijavljen na sistem ili nije, tj. neprijavljeni korisnik može pregledati događaje, ali ne rezervirati ih, dok prijavljeni ima mogućnost rezervacije. Implementirat ćemo interfejs IStanje koji će prikazivati odgovarajući pogled zavisno od toga da li je korisnik prijavljen ili nije. Zbog toga implementirali smo još dva pogleda PregledPrijavljenogView() i PregledNeprijavljenogView().

3. TEMPLATE METHOD PATTERN

Ovaj patern služi za omogućavanje izmjene ponašanja u jednom ili više dijelova. Najčešće se primjenjuje kada se za neki kompleksni algoritam uvijek trebaju izvršiti isti koraci, ali pojedinačne korake moguće je izvršiti na različite načine. Naš sistem je relativno jednostavan pa nismo vidjeli konkretnu primjenu ovog paterna.

4. ITERATOR PATTERN

Iterator patern možemo iskoristiti prilikom korisnikovog pregleda događaja. Korisnik može izabrati na koji način će se prikazati događaji, npr. slučajnim rasporedom, prema datumu održavanja itd.

5. OBSERVER PATTERN

Upotreba ovog paterna bi bila da korisnik dobije notifikaciju o nadolazećim događajima iz njegove grupe interesovanja.

Uloge su sljedeće:

- Subject - vlasnik objekta
- IObserver – korisnik
- Update – prijem notifikacije
- Notify – notifikacija koju dobiju svi korisnici iz iste grupe interesovanja
- State – događaj

6. CHAIN OF RESPONSIBILITY PATTERN

Chain of responsibility patern možemo iskoristiti pri rezervaciji događaja kako bismo bili sigurni da rezervaciju vrši registrovani korisnik te da ispunjava sve predulove za

prisustvovanje događaju. U našem primjeru handleri vrše procesiranje podataka i odlučuju da li će zahtjev biti proslijeđen dalje. Pretpostavljajući da će korisnik ispunjavati sve uslove, handleri će uspješno završiti procesiranje te će proces završiti uspješno rezervisanim događajem.

7. *MEDIATOR PATTERN*

Mediator patern možemo iskoristiti pri recenziji događaja pri čemu ćemo implementirati interfejs IMedijator koji će provjeravati da li je korisnik prisustvovao događaju putem metode Provjeri(). Pošto klasa RegistrovaniKorisnik može ostavljati recenzije njoj ćemo dodati atribut tipa IMedijator.