

# KREACIJSKI PATERN

## 1. **SINGLETON PATERN:**

Singleton patern se koristi kada je potrebno osigurati da tokom izvršavanja programa postoji samo jedna instanca neke klase.

U ovom projektu singleton patern može biti iskorišten prilikom kreiranja baze podataka, jer je potrebno da sve kontrolerske klase imaju pristup istim podacima te da su svima promjene vidljive u isto vrijeme.

## 2. **PROTOTYPE PATERN:**

Prototype patern se koristi kada želimo moći kreirati nove instance nekog već postojećeg objekta bez da se oslanjamo na samu klasu tog objekta.

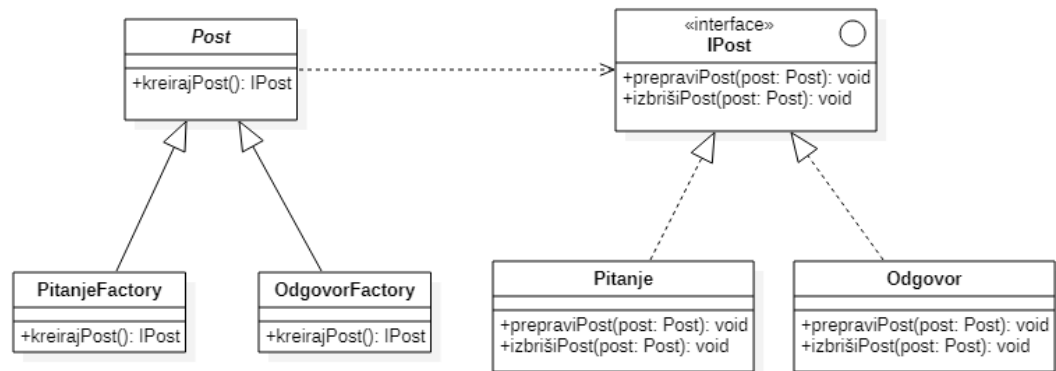
Ovaj patern bi mogli implementirati kod klase Pitanje. Ovaj patern i ne bi pravio neku veliku razliku pri kreaciji kada je u pitanju pregled tuđih pitanja, ali kako aplikacija daje mogućnost korisniku da pregleda posebno sva postavljena pitanja a posebno vlasitja, bilo bi povoljno da prilikom pregleda vlastitih pitanja olašamo kreaciju samih njih, naime ukoliko bi ovdje koristili prototype patern prilikom kreacije svakog novog pitanja bilo bi potrebno samo promijeniti dio pitanja jer postavljač pitanja bi ostao isti, tj korisnik koji gleda svoja postavljena pitanja. U tom slučaju bi imali neki interfejs IPitanje koji posjeduje metodu "clone" te koju bi implementirala klasa Pitanje.

## 3. **FACTORY METHOD PATERN:**

Factory method patern pruža interfejs za kreiranje objekata nekog tipa u jednoj superklasi, ali koji također dozvoljava da podklase mogu mijenjati tip objekta koji će biti kreiran. Ovaj patern je pogodan za korištenje ukoliko unaprijed ne znamo tačan tip objekta s kojim će program raditi.

Factory method patern će se iskoristiti prilikom kreiranja postova (u našem slučaju to su trenutno postovi tipa Pitanja i Odgovora). Za realizaciju ovog patern bio bi potreban interfejs koji se naziva IPost i koji u sebi sadrži metode izbrišiPost() i prepraviPost(), zatim klasa Post koja ima metodu kreirajPost() čija je povratna vrijednost IPost. Klasu Post nasljeđuju klase Pitanje i Odgovor, a interfejs IPost implementiraju klase PitanjeImpl i OdgovorImpl. Razlog za uvođenje ovog patern je fleksibilnije kreiranje različitih postova u budućnosti koji ne moraju biti nužno samo tipa Pitanje i Odgovor ali i razlika u implementaciji same metode izbrišiPost. Prilikom brisanja odgovora brišu se svi podaci vezani za taj odgovor i to ne utiče na objekte drugih klasa, sem eventualnih

tagova koji su u odgovoru, dok brisanje jednog pitanja povlači i brisanje svih odgovora vezanih za to pitanje.



#### 4. **ABSTRACT FACTORY PATTERN:**

Abstract method pattern se koristi kada želimo imati mogućnost kreiranja familija povezanih objekata bez potrebe definisanja konkretnih klasa. Ovo je povoljno kada su te klase trenutno nepoznate, ili ako želimo ostaviti prostora za dalji razvoj i proširenje funkcionalnosti.

Ovaj pattern bi mogli iskoristiti ukoliko bi dodali posebne rubrike za različite teme na koje bi se mogli postavljati postovi, npr. Ukoliko bi imali jednu rubriku za sport jednu za kulturu i jednu za nauku. I ukoliko bi smo dopustili da postovi moraju biti posebnog oblika, npr. U rubriku za sport je omogućeno postavljanje pitanja samo u obliku slika a odgovora u obliku glasovnih poruka, u rubriku za nauku ukoliko bi smo omogućili da se tekst pitanja i odgovora formatira putem Latex tekst editora, dok bi rubrika za kulturu imala obične funkcionalnosti postavljanja pitanja i odgovora. Tako da bi smo imali neki interfejs IRubrikaFactory nju bi implementirale sljedeće klase RubrikaKultureFactory, RubrikaNaukeFactory i RubrikaSportaFactory, koji bi dalje generisali Pitanje/OdgovorSport, Pitanje/OdgovorNauka i Pitanje/OdgovorKultura, koji su implementacije apstraktna klasa Pitanje i Odgovor.

#### 5. **BUILDER PATTERN:**

Build pattern daje mogućnost da objekte, čija inicijalizacija je kompleksa, inicijalizujemo po dijelovima. Ovim patternom možemo prikazati različite vrste jednog objekta čije su samo razlike u inicijalizaciji.

Builder pattern će se iskoristiti prilikom kreiranja Usera tako da je samo polje e-mail, username i password zahtjevano, dok su polja name, i lastname opcionalna polja koja se vremenom mogu promijeniti. Za potrebe realizacije builder patterna biće kreiran

interfejs IUserBuilder sa metodama postaviIme, postaviPrezime, postaviBrojTelefona i postaviSliku. Interfejs će biti implementiran od strane klase UserBuilder koja kao privatni atribut ima objekat tipa User . Pored implementacije metoda, klasa UserBuilder ima i metodu getResult koja vraća instancu Usera nad kojim se vrše promjene.

