KREACIJSKI PATERN

1. ***SINGLETON PATERN:***

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

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

2. ***PROTOTYPE PATERN:***

   Prototype patern se koristi kada zelimo moci kreirati nove instance nekog vec postojeceg objekta bez da se oslanjamo na samu klasu tog objekta.

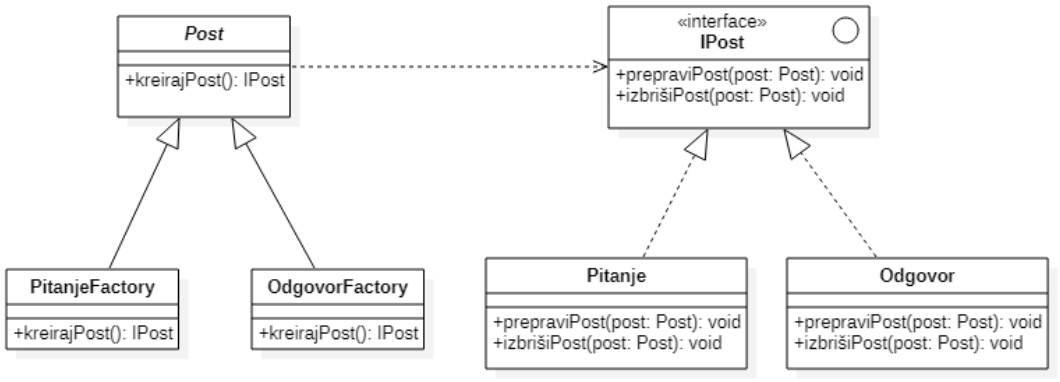
   Ovaj patern bi mogli implementirati kod klase Pitanje. Ovaj pratern i ne bi pravio neku veliku razliku pri kreaciji kada je u pitanju pregled tuđih pitanja, ali kako aplikacija daje mogucnost korisniku da pregleda posebno sva postavljena pitanja a posebno vlasitia, bilo bi povoljno da prilikom pregleda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

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

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

   Factory method patern ce se iskoristiti prilikom kreiranja postova (u nasem slucaju to su trenutno postovi tipa Pitanja i Odgovora). Za realizaciju ovog paterna bio bi potreban interfejs koji se naziva IPost i koji u sebi sadrzi metode izbrisiPost() i prepraviPost(), zatim klasa Post koja ima metodu kreirajPost() cija je povratna vrijednost IPost. Klasu Post nasljeđuju klase Pitanje i Odgovor, a interfejs IPost implementiraju klase PitanjeImpl i OdgovorImpl. Razlog za uvođenje ovog paterna je fleksibilnije kreiranje razlicitih postova u buducnosti koji ne moraju biti nuzno samo tipa Pitanje i Odgovor ali i razlika u implementaciji same metode izbrisiPost. Prilikom brisanja odgovora brisu se svi podaci vezani za taj odgovor i to ne utice na objekte drugih klasa, sem eventualnih

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



4. ***ABSTRACT FACTORY PATERN:***

Abstract method patern se koristi kada zelimo imati mogucnost kreiranja familija povezanih objekata bez potrebe definisanja konkretnih klasa. Ovo je povoljno kada su te klase trenutno nepoznate, ili ako zelimo ostaviti prostora za dalji razvoj i prosirenje funkcionalnosti.
Ovaj patern bi mogli iskoristiti ukoliko bi dodali posebne rubrike za razlicite 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 omoguceno postavljanje pitanja samo u obliku slika a odgovora u obliku glasovnih poruka, u rubriku za nauku ukoliko bi smo omogucili da se tekst pitanja i odgovora formatira putem Latex tekst editora, dok bi rubrika za kulturu imala obicne funkcionalnosti postavljanja pitanja i odgovora. Tako da bi smo imali neki interfejs IRubrikaFactory nju bi implementirale sljedece 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 PATERN:***

Build patern daje mogucnost da objekte, cija inicijalizacija je kompleksa, inicijalizujemo po dijelovima. Ovim paternom mozemo prikazati razlicite vrste jednog objekta cije su samo razlike u inicijalizaciji.

Builder patern ce 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 paterna bice kreiran

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