Šabloni

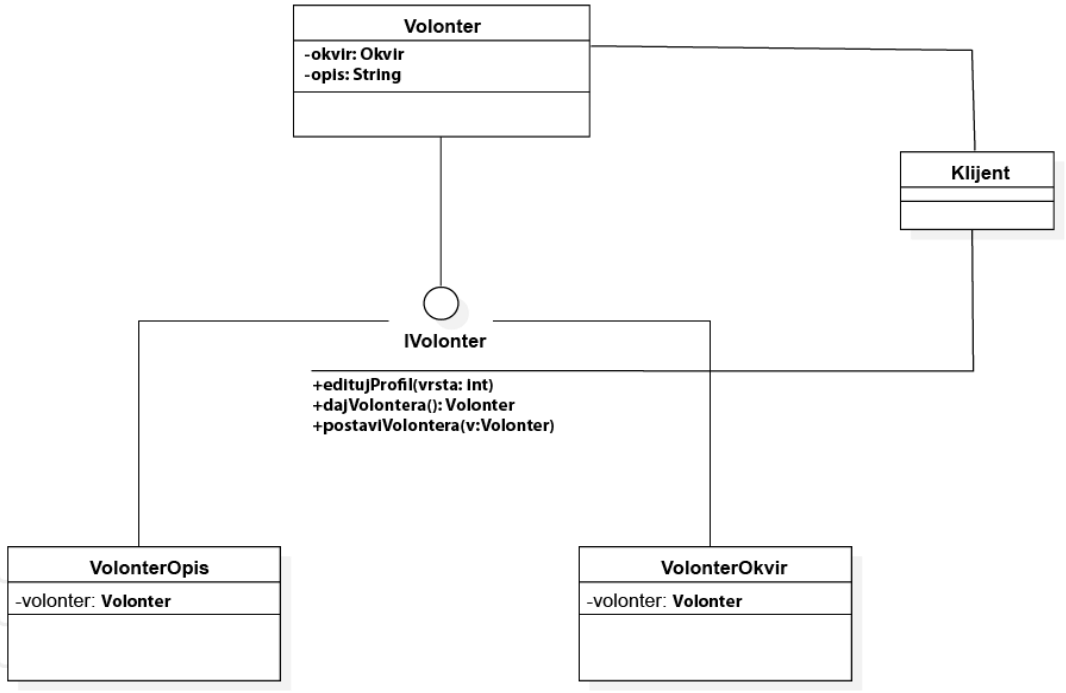
Strukturalni šabloni

1. Facade šablon

Facade šablon bismo mogli koristiti kada korisnik kreira novi zahtjev ili žalbu. Prilikom kreiranja zahtjeva potrebno je izvršiti validaciju podataka, kreirati novi zahtjev ili žalbu, te isti dodati u list zahtjeva i žalbi sistema kojoj administrator može pristupiti.Ovdje je korisniku potrebna samo infromacija o tome da li je njegov zahtjev ili žalba kerirana, a ono što se događa u pozadini se korisnika ne treba da tiče.Zbog ovoga ovdje bi bilo prikladno koristiti Facade šablon.

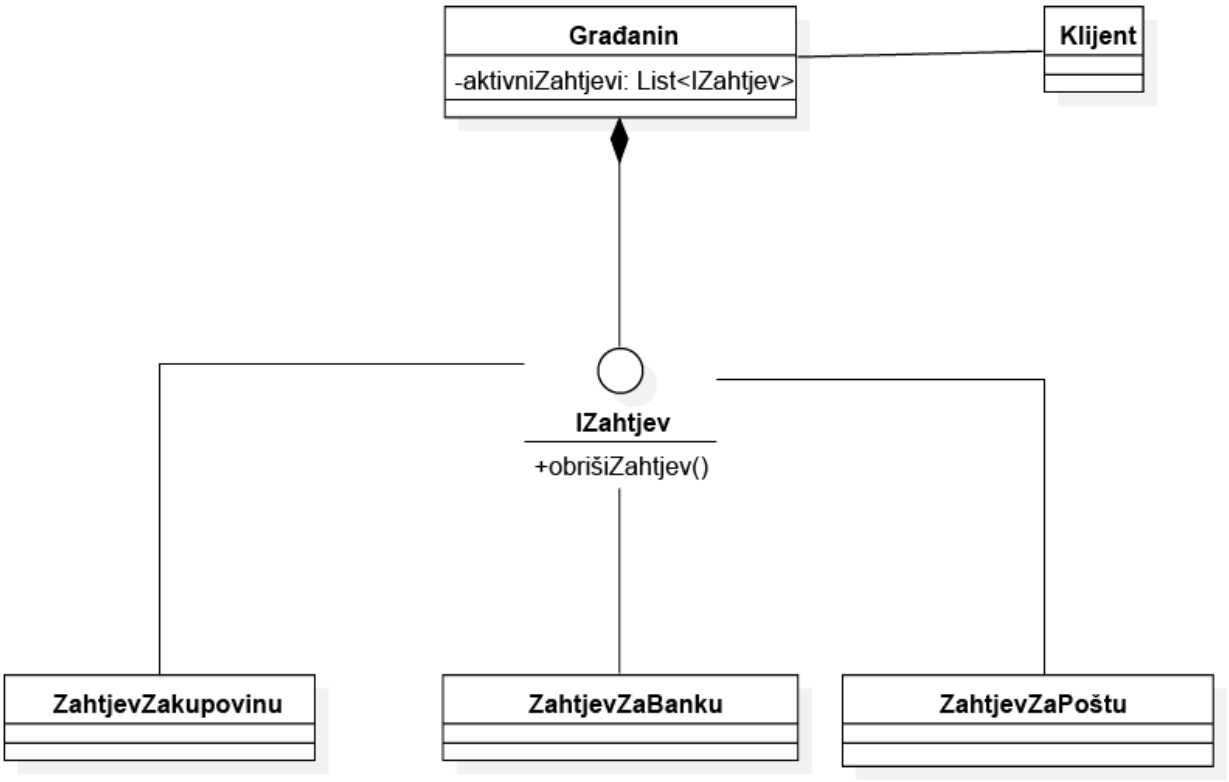
2. Decorator šablon

Decorator šablon bismo mogli koristiti u slučaju kada bismo htjeli korisniku omogućiti da uredi svoj profil, tj. korisnik može da doda opis svom profilu ili okvir oko slike. Ovo postižemo koristeći IVolonter interfejs iz koga su dalje izvedeni VolonterOpis i VolonterOkvir.



3. Composite šablon

Pošto posjedujemo različite vrste zahtjeva mogli bismo implementirati Composite šablon tako što u apstraktnoj klasi Zahtjev imamo metodu dajPodatke() koja bi vraćala neki bitne podatke o zahtjevu.



4. Proxy šablon

Možemo koristi proxy šablon za listu zahtjeva, putem ove liste bismo Volonteru ili Građaninu omogućili da samo vide svoje zahtjeve koje su započeli, dok administratoru bismo omogućili pravo pristupa svim zahtjevima u sistemu.

5. Bridge šablon

Kada bismo za svaki od zahtjeva htjeli dodati važnost - hitna i obična. Umjesto da kreiramo nove klase HitanZahtjevZaKupovinu, ObičanZahtjevZaKupovinu, HitanZahtjevZaBanku i sl. Mi ćemo kreirati podklasu Važnost iz koje su izvedene klase Hitan i Običan i nju ćemo staviti kao atribut klase Zahtjev, tj. klasa Zahtjev sadrži klasu Važnost.

6. Flyweight šablon

Recimo da imamo klasu Okvir koja služi za čuvanje podataka o okviru profilne slike korisnika. Ako bismo za svakog korisnika kreirali novi okvir ovo bi zauzelo mnogo memorije. Bolje rješenje bi bilo da prilikom dodavanja okvira imamo neki pool u kojem čuvamo sve do tad kreirane okvire. Ako okvir sa atributima koje je korisnik odabrao već postoji u poolu onda korisniku dodjeljujemo već postojeći okvir, ako takav okvir ne postoji onda kreiramo novi i dodajemo u pool.

7. Adapter šablon

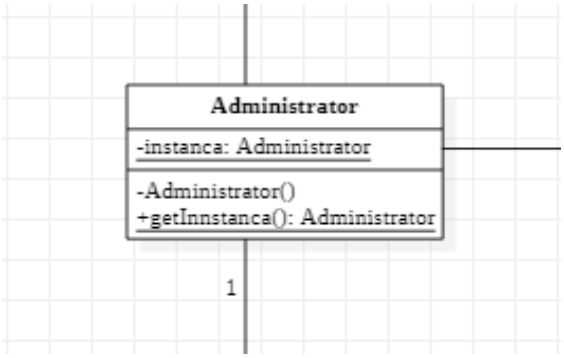
Možemo napraviti klasu Izvještaj. Ova klasa bi nam služila za kreiranje izvještaja o korisniku ili o nekom zahtjevu. Kako ne bismo morali pravit dvije različite klase izvještaja možemo implementirat adapter koji omogućava da ova klasa može kreirati i izvještaj o korisniku i o zahtjevu.

Kreacijski šabloni

1. Singleton šablon

Singleton šablon osigurava a da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Postoje objekti čija je samo jedna instanca potrebna i nad kojima je potrebna jedinstvena kontrola pristupa. Instanciranje više nego jednom može prouzrokovati probleme kao što su nekorektno ponašanje programa, neadekvantno korištenje resursa ili nekonzistentan rezultat.

U našem projektu klasa Administrator će biti realizovana kao singleton klasa, konstruktor klase proglašavamo privatnim, dodajemo statički atribut tipa Administrator kao i statičku metoda getInstanca(): Administrator.



2. Prototype šablon

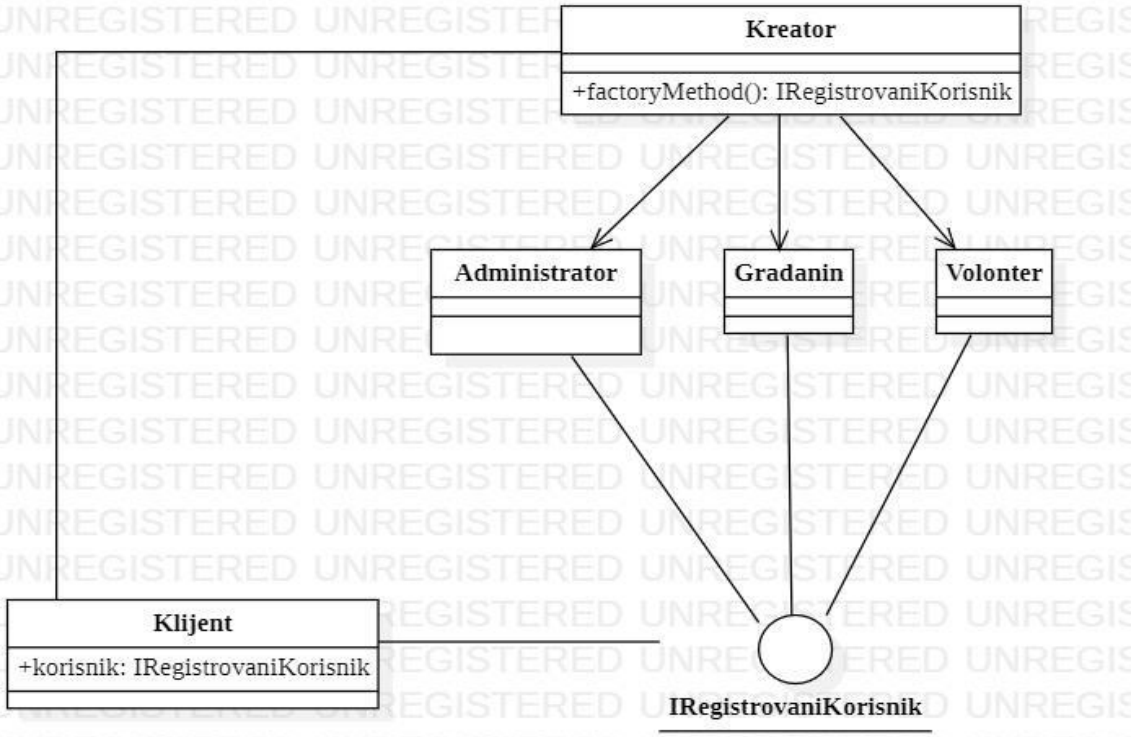
Prototype šablon kreira nove objekte klonirajući jednu od postojećih prototip instanci. Dakle, ovaj šablon omogućava jednostavnije kreiranje novih instanci klasa kod kojih je veliki broj atributa identičan za većinu instanci.

Naš sistem se sadrži od klasa kod kojih je većina atributa za većinu instanci različita, čak i svi različiti, jer su u pitanju klase koje modeliraju jedinstvenu osobu, zahtjev ili žalbu pa niti ima potrebe za kloniranje, niti ima smisla isto činiti.

3. Factory Method šablon

Factory Method šablon omogućava kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite podklase mogu na različite načine implementirati interfejs.

U Solidarity sistemu ovaj šablon može biti iskorišten kod klasa koje su izvedene iz klase RegistrovaniKorisnik. Instanciranje klasa može se prebaciti na jedno mjesto u programu, što će omogućiti lakše dodavanje novih vrsta korisnika ukoliko bude potrebe za time prilikom budućeg razvoja aplikacije.



4. Abstract Factory šablon

Abstract Factory šablon se koristi pri radu sa familijama različitih objekata. Na osnovu apstraktne familije objekata kreiraju se konkretne fabrike produkata različitih tipova i različitih kombinacija.

U našem sistemu ovaj šablon bi se mogao primjeniti ukoliko bismo imali više različitih mode-ova korisničkog interfejsa.

5. Builder šablon

Builder šablon odvaja specifikaciju kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

Za potrebe naše aplikacije nije potrebno koristiti builder šablon prilikom kreiranja objekata pošto su klase u sistemu relativno proste, odnosno nije potrebno odvojeno kreiranje dijelova objekta. Međutim, za slučaj da je klasa RegistrovaniKorisnik ili podklase izvedene iz nje kompleksnije, tada bi učinkovitije bilo atribute grupisati u dijelove i onda iz tih dijelova sklopiti objekat.

Šabloni ponašanja

1. Strategy šablon

   Unutar volontera imamo listu Zahtjeva koje je preuzeo. Možemo implementirati strategy šablon tako da možemo odabrati načine sortiranja (po datumu, imenu ili prezimenu podnosioca zahtjeva i sl.)

2. State šablon

   Možemo imati dva state-a ulogovan i neulogovan. Ako korisnik nije ulogovan aktivan je state neulogovan. Dok je ovaj state aktivan korisnik ima samo osnovne mogućnosti npr. samo pregled podnesenih zahtjeva. Kada je korisnik ulogovan aktivan je state ulogovan. Sada korisnik dobija dodatne mogućnosti u zavisnosti da li je Građanin, Administrator ili Volonter.

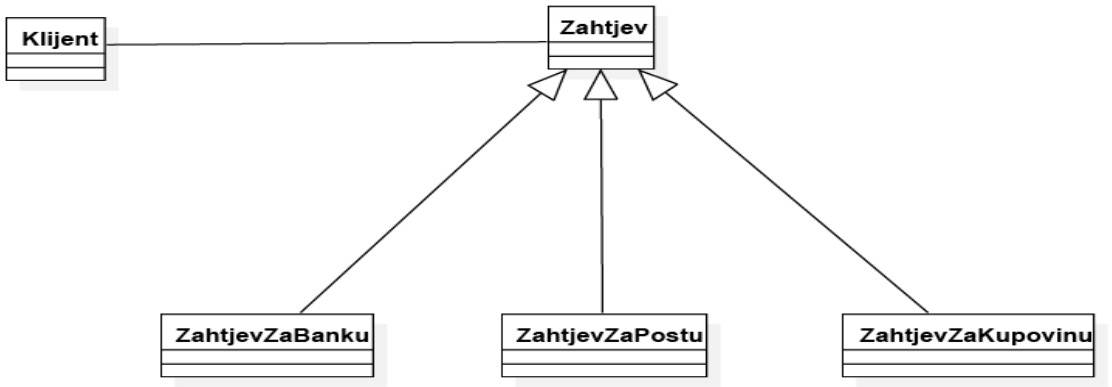
   Volonter - može preuzimati i nadgledati preuzete zahtjeve
   Građanin - može kreirati nove zahtjeve i pregledati zahtjeve koje je do tad kreirao
   Administrator - dobija pristup administratorskom panelu

3. Template šablon

   Template šablon možemo implementirati tako što Građanin može da kreira zahtjev pri čemu ima mogućnost da kreira jednu od tri vrste zahtjeva:

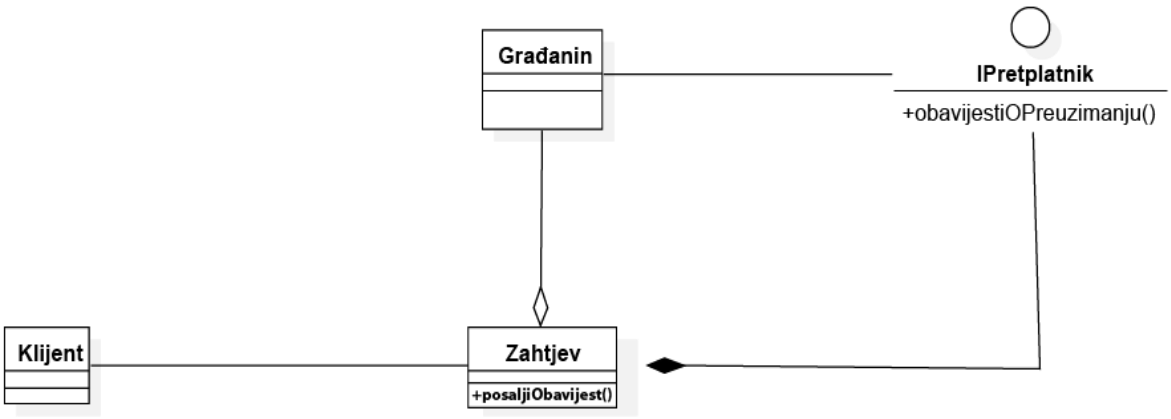
   a. ZahtjevZaKupovinu
   b. ZahtjevZaBanku
   c. ZahtjevZaPostu

   Dakle imamo Zahtjev klasu koja na osnovu onoga što je korisnik odabrao omogućava da s e kreira jedna od tri klase koje su navedene iznad.



4. Observer šablon

Kada volonter preuzme zahtjev, građaninu koji je kreirao zahtjev se šalje notifikacija da je njegov zahtjev preuzet. Ako bi se nastavio dalji razvoj aplikacije mogla bi se dodati mogućnost da korisnik ima pravo da odabere da mu se pošalje notifikacija, npr. e-mailom, kada se nakupi određeni broj novokreiranih zahtjeva.



5. Iterator šablon

Svaki volonter može vidjeti listu zahtjeva koje je preuzeo i svaki građanin može vidjeti listu zahtjeva koje je kreirao. Nad ovim listama se može implementirat iterator šablon. Ovako možemo prolaziti kroz listu na različite načine.

6. Chain of responsibility šablon

Ovaj šablon možemo koristiti prilikom autentifikacije i autorizacije različitih vrsta korisnika i njihovom pravu pristupa zahtjevima. Možemo kreirati handler koji provjerava da li korisnik ima prava čitanja, kreiranja ili brisanja zahtjeva iz liste. Kasnije ako bi se razvoj ove aplikacije nastavio može doći do potrebe za dodavanjem novih handlera koji će vršiti potrebne provjere.

7. Mediator šablon

U daljem razvoju aplikacije ukoliko bismo željeli omogućiti interakciju između korisnika pute chat-a možemo implementirati mediator šablon koji bi vršio kontroluinterakcije između korisnika.