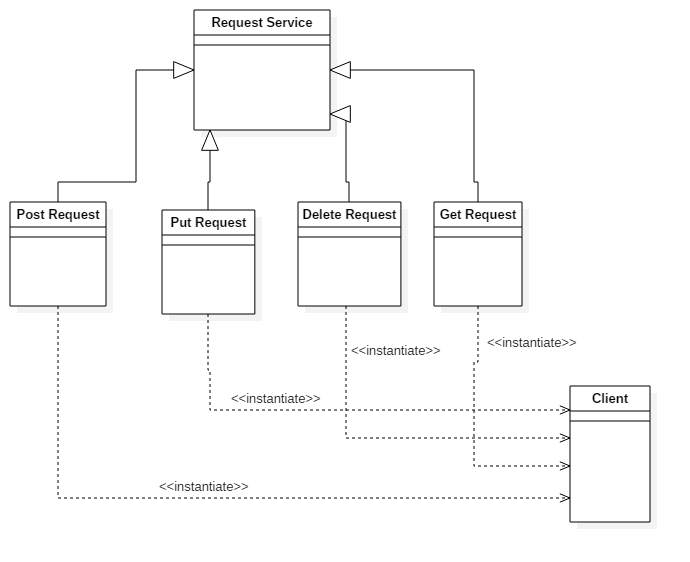
**MACROP - Manageable and Collaborative Resources Of Project**

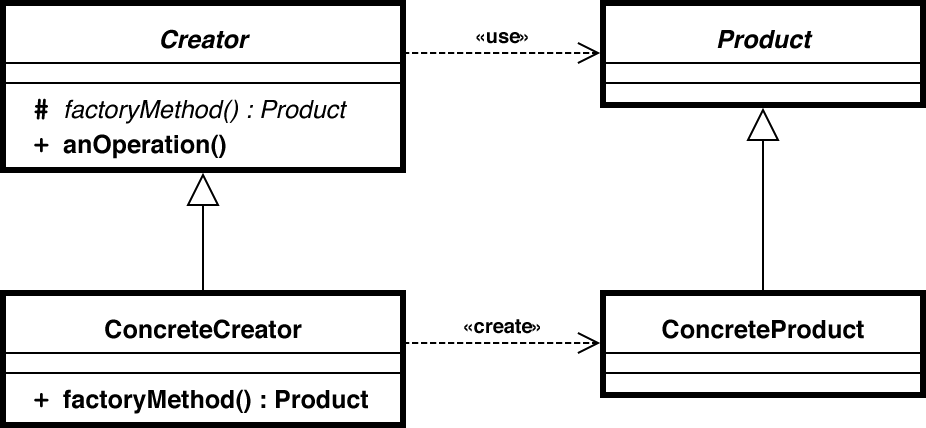
**Project Patterns**

# Factory pattern

Projektni obrazac Factory je iskorišćen prilikom pravljenja request zahteva sa parametrima koji predstavljaju putanju, odnosno URL REST API-ja i podatkom, najčešće u JSON formatu koji se šalje.

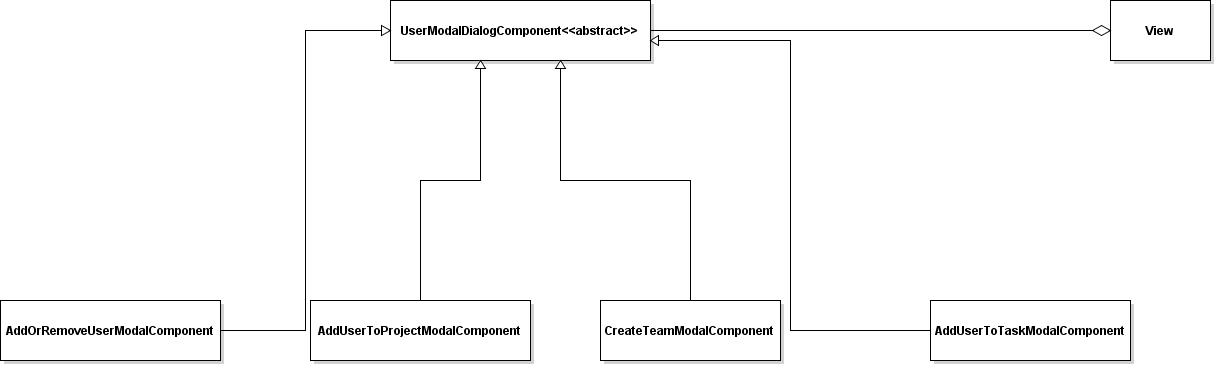


Po GoF-u Factory Method je definisan



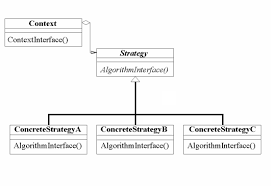
# Strategy pattern

Strategy pattern je pattern koji lako omogucava dodeljivanje novih funkcionalnosti vec postojecem setu funkcionalnosti. Obzirom da je Angular 4 based-compnent framework, arhitekturno gledano MVC, nije bas moguce u potpunosti implementirati strategy, jer komponenta ne moze biti apstraktna, sto I ima smisla. Svaka komponenta se sastoji iz view-a controllera I potrebnih modela za prikazivanje. Kako View koji je zapravo nista drugo do HTML strane , ne moze biti apstraktan, uspeli smo da enkapsuliramo kontroler komponente u apstraktnu klasu UserBaseModalComponent koja zapravo predstavlja skup osnovnih funkcionalnosti zajednickih za sve MODALE. Iz UserBaseModalComponent jer moguce lako izvodjene novih modala, predefinisanje odredjenih funkcija koje omogucuju nacin otvaranja dijaloga, kao I podatke koji ce biti ucitani. UML dijagram ovog patterna mapiranog po nasim potrebama izgleda ovako:



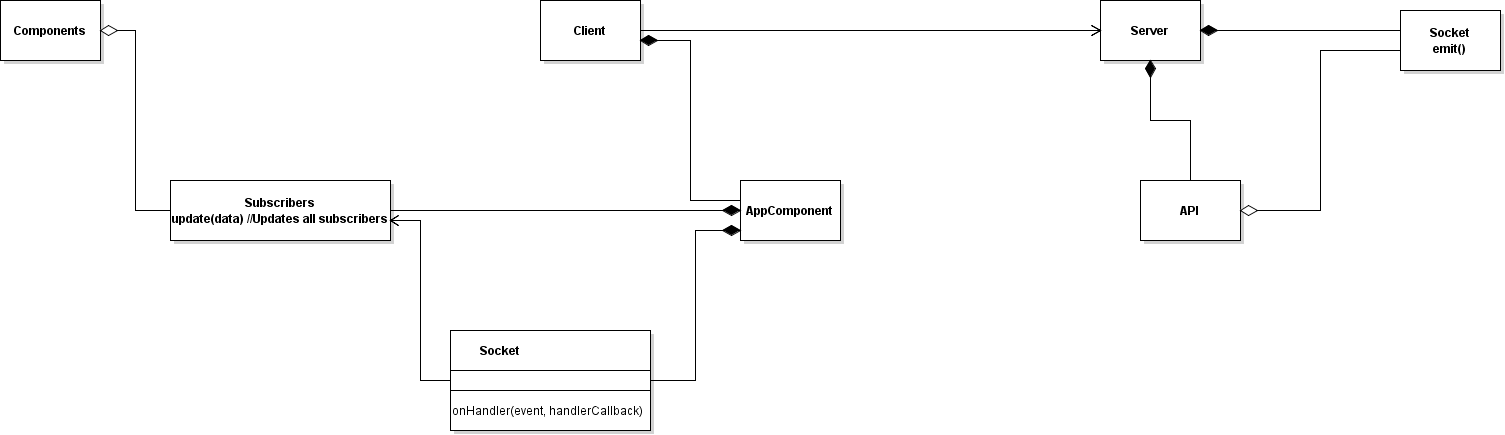
View predstavlja bilo koju komponentu koja implementira ovaj kontroler, I kojim upravlja predefinisani kontroler(naravno misli se na VIEW deo iz MVC-a).

Po GoF-u Strategy pattern izgleda ovako:

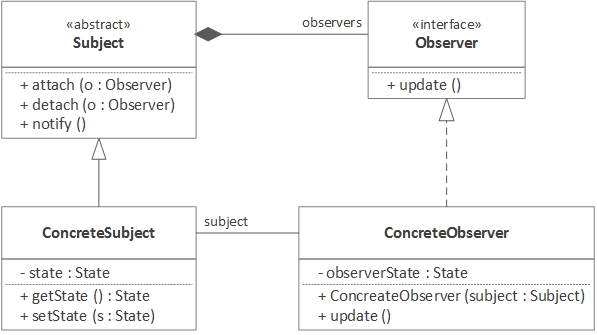


# Observer pattern

Observer pattern se koristi kada zelimo da promene inicirane od jednog korisnika Sistema, budu vidljive od strane svih korisnika sistema. Mnogi framework-ovi imaju implementiran observer mehanizam, medjutim da bi aplikacija radila u real-time rezimu, potrebno je implementirati odredjene funkcionalnosti. Kako se na MACROP aplikacijii koriste servisi za komunikaciju klijenata I centralnog servera deploy-ovanog na HEROKU, potrebno je otvoriti komunikacioni kanal izmedju klijenata aplikacije I servera. U tu svrhu je iskoriscena biblioteka Socket.io koja otvara virtuelni kanal ka server (socket), attachuje se na server, I za svakog klijenata otvara poseban socket. Proces komunikacije je sledeci: Klijent salje request server (upis nekog podatka, ili neka promena, u MACROP- takve promene zovemo poruke), server obradjue dati request, vraca podatke klijentu koji ih je inicirao, I emituje event (io.emit(data)). Na klijentskoj strain u root aplikacionom delu (AppComponent) izvrsena je pretplata (subscribe) na Socket na datoj lokaciji (new Socket (url)) , I za svaki event se definise handler, (io.on(event, handlerCallback)). HandlerCallback zatim na klijentskoj strani publishuje promene svim subscriberima pretplacenim na dati servis (npr. ChatSubscriber service). Na taj nacin poruke stizu samo do ljudi kojima su potrebne. UML dijagram koji najprostije opisuje ovu komunikaciju prikazan je na slici:



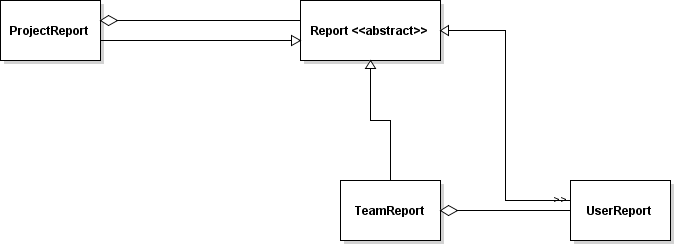
Po GoF-u Observer pattern je definisan na sledeci nacin:



# Composite pattern

Composite pattern se koristi kada je potrebno organizovati objekte (modele) u strukturu stabla. U MACROP aplikaciji postoji Report system koji poseduje hijerarhiju datih reportova od strane korisnika sistema, kao I privilegije nad njima. Da bi hijerarhiju predstavili strukturom stable, uvedeni su sledeci modeli: Report- apstraktna klasa , ProjectReport – klasa koja nasledjuje Report, ujedno je I Composite klasa jer ima listu Report-ova, sto znaci da ProjectReport za childove moze imati sve sto ima Report po sebi. Razlog za uvodjenje ovog patterna je taj, sto zelimo da Project owner moze da vidi project report , team report I user report(hijerahiju reportova),odnosno vidi Team reportove za svaki tim na projektu, dok Team leader moze da vidi samo Team report(koji sacinjavaju User reportovi njegovo tima). Na taj nacin smo uveli hijerarhiju, privilegije (nivoe), za svakog korisnika sistema, jer svaki korisnik moze biti user, leader ili project owner. UML dijagram MACROP report Composite patterna izgelda ovako:

\*NAPOMENA\* : U agregaciji izmedju ProjectReporta I Report<<abstract>> , ProjectReport sadrzi listu Reportova, ali UML editor nije podrzavao labelu za oznacavanje 1 to \* relacija.



Prema GoF-u Composite pattern je definisan:

