PATERNI PONAŠANJA

1. STRATEGY

Ovaj patern možemo dodati u naš sistem na sljedeći način. Poznato nam je da unutar našeg sistema imamo 3 vrste korisnika: admin-a, zaposlenog (nutricionistu) te samog klijenta. Stoga prilikom samog prijavljivanja korisnika, imamo sigurno 3 opcije prilikom registracije. Detekcija o kojem od tri nabrojana korisnika se radi, vrši se na osnovu ulaznih podataka. Na osnovu datog tipa klase tj. korisnika mogli bismo realizovati 'set-up' nakon registracije u kojem bi za svaki tip korisnika postojao drugačiji prikaz (znamo da ova tri krosnika mogu pristupiti i vidjeti različite stvari), samim tim možemo i drugačije podatke dobiti pomoću api request-a prema našoj bazi. Još jedan primjer gdje bi se mogao dodati ovaj patern odnosi se na različite planove ishrane. U zavisnosti od ulaznih podataka klijenta, nutricionista treba da izabere “strategiju” tj. tip prehrane koji odgovara ulaznim podacima kijenta.

2. STATE

Obzirom da trenutno ne vidimo neki koristan način da se ovaj patern doda u naš sistem, osmislit ćemo jednu situaciju nadovezujući se na dokument o kreacijskim paternima. U tom dokumentu je opisano postojanje takozvanog baznog obroka, čiji je cilj olakšati nutricionisti da osmisli plan ishrane. Zamislimo situaciju u kojoj postoji klijent koji ima intoleranciju na gluten i alergičan je na orašaste plodove, te klijent koji je alergičan na orašaste plodove i jagode. Mogli bismo osmisliti još nebrojeno mnogo ovakvih kombinacija. To je mjesto gdje se potencijalno javlja potreba za dodavanjem state paterna. Mogli bismo napraviti interfejs IStanjaKlijenta koji komunicira sa različitim stanjima – alergijama, preferencijama, tipovima ishrane (ovo bi sve bile klase), a klasa Meal (obrok) bi ustvari predstavljala context klasu.

3. TEMPLATE METHOD

Ovaj patern možemo dodati u naš sistem na sljedeći način. Obzirom da API za nas radi samo kreiranje plana ishrane, te sam način na koji to radi nije nam od velikog interesa bio dosad, zamislit ćemo ponovo hipotetičku situaciju i nadovezati ćemo se na prethodni patern i postojanje baznog obroka. Za dodavanje ovog paterna bit će nam potrebna apstraktna klasa BazniObrok sa metodama templateMethod, filtrirajNamirnice. U ovisnosti od toga kakav je tip ishrane određenog klijenta, ove metode će biti različito implementirane. Za konkretan tip ishrane bit će primijenjena konkretna klasa od 9 mogućih: Gluten Free, Ketogenic... U svakoj od navedenih klasa, metoda

filtrirajOpcenito bi izdvajala one namirnice koje nisu podržane u toj dijeti. Na taj način bi se omogućilo olakšano kreiranje unikatnog plana ishrane za svakog klijenta.

4. OBSERVER

Ovaj patern možemo dodati u naš sistem uz implementaciju nove funkcionalnosti sa svrhom da olakšamo posao nutricionisti. Uprkos jako velikom broju mogućih kombinacija specifičnih zahtjeva svakog klijenta, počevši od tjelesnih karakteristika, pa sve do alergija, intolerancija i tipova ishrane, svjesni smo da su neka stanja mnogo češća od drugih. Tako se može desiti da se pojavi neki klijent čiji zahtjevi odgovaraju zahtjevima nekog od prethodno registriranih klijenata. Nutricionisti bi bilo korisno "da ga se obavijesti" u slučaju da su zahtjevi tog klijenta već viđeni i da je kreiran plan ishrane koji prati takve zahtjeve. U takvoj situaciji ovaj patern bi bio koristan, s tim da bi se morala vršiti "analiza" prethodno registriranih klijenata i njihovih specifikacija s ciljem uporedbe s novim klijentom (efikasnost takve radnje je upitna).

5. ITERATOR

Ovaj patern možemo dodati u naš sistem na sljedeći način. Recimo da nutricionista želi da izvrši pregled klijenata. Default-ni način bi bio po abecednom redu. No u slučaju da nutricionista želi da pregleda klijente sortirane po BMI – u (npr. od manjeg ka većem) ili soritrane po tipu ishrane (počevši npr. od Gluten Free). U opisanoj situaciji, ovaj patern bi moga biti od koristi.

6. CHAIN OF RESPONSIBILITY

Ovaj patern možemo dodati u naš sistem uzevši opet u obzir postojanje tzv. Baznog obroka. Ukoliko nutricionista kreira plan ishrane sačinjen od 3 obroka prema potrebama svakog klijenta, desit će se da se neki obroci razlikuju u možda svega nekoliko namirnica. Tako bi uz uvažavanje klijentovih potreba, na već postojeći bazni obrok bile dodane neke namirnice, ili oduzete u slučaju da ih klijent ne podnosi. Prošlo bi se kroz tip ishrane koji je klijent označio kao željeni i izvršila detekcija onih namirnica koje nisu dopuštene u tom tipu ishrane, potom bi se provjerilo da li obrok zadovoljava nutritivne vrijednosti koje su potrebne za konkretnog klijenta, a na samom kraju bi se izdvojile one namirnice na koje klijent posjeduje alergiju ili intoleranciju.

7. MEDIATOR

Obzirom da mediator i facade patern rade sličan posao, a mi smo facade patern dodali na način da smo ApiWrapper klasu proglasi fasadnom klasom obzirom da ona vrši komunikaciju sa svim drugim klasama i obavlja za nas posao kreiranja plana ishrane, što ujedno predstavlja srž našeg sistema, i ovaj patern ćemo dodati u naš sistem proglašavanjem ApiWrapper mediator klasom. Naime čitav proces kreiranja planova ishrane prilično je haotičan. Postoji mnogo zahtjeva koje klijent može napraviti, od tipa ishrane, BMI-a, alergija, intolerancija itd. Usaglašavanje svih tih zahtjeva bilo bi poput dodjeljivanja prednosti vozačima na raskrsnici, a policajac u našem slučaju je upravo ApiWrapper klasa koja taj haotični proces uobliči i da nam finalni product – plan ishrane za svakog klijenta.