Decorator

Decorator patern može se koristiti za dinamičko dodavanje novih odgovornosti objektima. Na primjer, PlaceOrderView se može omotati dekoratorom koji bi omogucio dodavanje funkcije koja bi vrsila validaciju recepta za narudzbe sa receptom bez mijenjanja izvorne klase. Mogao bi se takodjer implementirati dekorater koji moze dodati različite funkcije oblikovanje same forme za narucivanje. Ovaj pristup omogućuje fleksibilno poboljšanje komponenti prikaza kombiniranjem više dekoratora prema potrebi.

Bridge

Bridge patern pomaže odvojiti apstrakciju od njezine implementacije, cineci ih neovisnima. Konkretno u nasem slucaju različite korisničke uloge poput Administratora, Farmaceuta i Kupca mogu imati različite poglede, ali logika interakcije ostaje ista. Odvajanjem logike korisničkog interfejsa od pozadinskih interakcija, možete se prebacivati između različitih implementacija bez mijenjanja apstrakcije. Ovaj obrazac povećava fleksibilnost u upravljanju različitim korisničkim interfejsima sto je veoma korisno za prelazak izmedju HTML i mobile pogleda.

Composite

Composite patern koristan je za upravljanje hijerarhijskim strukturama kao što je na primjer katalog proizvoda. Svaki proizvod, u nasem slucaju lijek, može biti kombinacija vise lijekova ili imati vise komponenti, što omogućuje da složeni lijekovi budu sastavljeni od jednostavnijih. Ova je struktura posebno korisna za prikaz ugniježđenih informacija o proizvodu u ProductDetailsViewu, gdje se i pojedinačni proizvodi i grupe proizvoda (paketi) tretiraju jednoobrazno. Ovaj patern pojednostavljuje kod na strani klijenta dopuštajući jedinstveni tretman pojedinačnih i kompleksnih objekata.

Proxy

Proxy patern kontrolira pristup objektu djelujući kao neka vrsta surogata ili rezervisanog mjesta. U nasem slucaju npr. mogao bi se implementirati neki PlacanjeProxy patern koji može potvrditi podatke o plaćanju prije prosljeđivanja zahtjeva stvarnom servisu placanja (npr. Paypal). Time se prije svega osigurava sigurnost, a i može se nositi s dodatnim odgovornostima kao sto su predmemorisanja ili zapisivanja. Korištenjem proxyja mogu se zaštititi osjetljive operacije i optimizirati njihova upotreba.

Flyweight

Flyweight patern ima za cilj minimizirati korištenje memorije dijeljenjem zajedničkih dijelova objekata. U nasem slucaju ovaj patern se može primijeniti za upravljanje često korištenim objektima kao što su npr detalji proizvoda u katalogu proizvoda. Dijeljenjem zajedničkih podataka među višestrukim instancama smanjuje se opterećenje memorije. Ovaj pristup je koristan za aplikacije s velikim brojem sličnih objekata, poput nase apoteke s mnogo identičnih proizvoda. Ovim paternom se povećava učinkovitost i same performanse.