Strukturalni design patterni

Odabrani strukturalni design patterni za upotrebu su: composite i proxy.

Proxy design pattern - koristiti ćemo proxy pattern za implementaciju kontrole pristupa i sigurnosnih mjera za određene funkcionalnosti aplikacije ili osjetljive operacije. Koristiti ćemo proxy objekt za autentifikaciju i autorizaciju zaposlenika za obavljanje upravljačkih zadataka poput upravljanja cijenama, pregled financijskih izvještaja, prihvatanje ili odbijanje narudžbe te uvid u statistiku. Ovim ćemo postići da će proxy objekt omogućiti različitim korisnicima različit nivo pristupa i funkcionalnosti, štiteći sigurnost aplikacije. Proxy će posredovati između klijenta (korisnika) i stvarne implementacije objekta, pružajući dodatnu kontrolu pristupa i osiguravajući da se samo ovlašteni korisnici mogu baviti određenim operacijama.

Flyweight design pattern - u slučaju kada imamo meni s mnogo različitih pića, Flyweight pattern može nam pomoći optimizirati memoriju koja se koristi za pohranu informacija o tim pićima. Dijeljenjem zajedničkih podataka možemo smanjiti potrebnu memoriju za pohranu istih informacija za svaku instancu. Obzirom da broj pića u našem meniju nije toliko velik, odlučili smo da ne implementiramo ovaj pattern, međutim ukoliko bi se to promijenilo i došlo do naglog skoka broja pića ovaj pattern bi bio vrlo koristan.

Façade design pattern - facade pattern olakšava klijentima (gostima, konobarima, vlasnicima) da koriste funkcionalnosti aplikacije bez potrebe za poznavanjem unutrašnjeg složenog sistema. Ako bi naša aplikacija ima složene backend sisteme, kao što su upravljanje inventarom, obrada narudžbi, dostava i druge operacije, Facade pattern može ograničiti izlaganje tih kompleksnosti korisnicima. Kroz Facade, korisnici mogu pristupiti samo pojednostavljenim metodama koje su im potrebne, a kompleksne interne operacije se rješavaju unutar fasade. Također, pruža strukturu koja pomaže organizirati kod i povezane funkcionalnosti. Fasada djeluje kao jedinstvena tačka ulaza za korisnike, što olakšava razumijevanje i održavanje aplikacije. Ovaj pattern nije odabran za implementaciju jer ipak naš backend sistem nije toliko složen da bi potreba za njim bila opravdana, međutim u daljnjim nadogradnjama svakako da bi koristio i da bi se mogao implementirati.

Bridge design pattern – kada bi u našoj aplikaciji dodali funkcionalnosti da postoji loyality program za odabrane korisnike, tu bi nam koristio i dosta pomogao bridge pattern. Korištenjem bridge patterna nam omogućuje fleksibilnost u upravljanju loyality programa kafića. Tada bi smo mogli mijenjati ili zamijeniti loyality program bez uticaja na ostale funkcionalnosti aplikacije. Jedan od primjera je da želimo ažurirati loyality program ili dodati nove vrste nagrada bez mijenjanja korisničkih klasa i načina na koji korisnici pristupaju programu sa posebnim beneficijama.