KREACIJSKI PATERNI

SINGLETON PATTERN

Kreacijski Singleton patern osigurava da klasa ima samo jednu instancu i pruža globalnu tačku pristupa toj instanci. Glavna svrha ovog paterna je kontrola pristupa resursima koji se dijele, kao što su na primjer konfiguracijski objekti ili veze s bazom podataka.

U našem sistemu, ovaj patern možemo primjeniti na sistem notifikacija. To se se odnosi na korištenje Singleton patterna za NotifikacijaController. Ova klasa upravlja kreiranjem, dohvatom i brisanjem notifikacija u sistemu. Pošto su notifikacije centralna funkcionalnost koja se koristi širom aplikacije (npr. za obavještavanje korisnika o novim aukcijama ili promjenama statusa umjetnina), logično je imati jednu jedinu instancu NotifikacijaController-a koja koordinira sve notifikacije. Time se:

- izbjegava višestruko kreiranje kontrolera koje bi moglo dovesti do dupliranja notifikacija,
- omogućava centralizovano upravljanje i lakša kontrola logike slanja obavijesti,
- olakšava pristup toj funkcionalnosti iz različitih dijelova aplikacije.

PROTOTYPE PATTERN

Prototype dizajn patern dozvoljava da se kreiraju prilagođeni objekti bez poznavanja njihove klase ili detalja kako je objekat kreiran. Prototype obezbjeđuje interfejs za konstrukciju objekata kloniranjem jedne od postojećih prototip instanci i modifikacijom te kopije. Prototype patern se koristi ako je kreiranje novog objekta resursno zahtjevno. U tom slučaju rezonski je vršiti kloniranje već postojećeg objekata.

Konkretno u našem sistemu možemo implementirati ovaj patern pri unosu umjetnina. Naime, u tom slučaju često može doći do dupliranja umjetnina prilikom kreiranja sličnih unosa, jer se može desiti scenario gdje administrator unosi više umjetnina istog autora, tehnike i perioda (jer možda pripadaju istoj kolekciji) ali treba promjeniti samo ime ili datum. Umjesto da svaki put unosi sve podatke ispočetka, može klonirati (kopirati) postojeći objekat klase Umjetnina i izmijeniti samo potrebne atribute (naziv, datum).

FACTORY PATTERN

Uloga Factory Method paterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Factory patern odvaja (decoupling) zahtijevanje objekata od kreiranja objekata. Klijent ne treba da zna koji tip objekta je kreiran.

U našem sistemu ovaj patern se može primjeniti pri kreiranju korisnika prema ulozi. S obzirom da se pri kreiranju korisnika u našem sistemu njemu automatski dodjeljuje i određena uloga (korisnik, kupac, administrator i kritičar), ovaj patern se može iskoristiti za kreiranje tipova korisnika tako da klijent ne zna koji tip objekta je kreiran.

ABSTRACT FACTORY PATTERN

Abstract Factory pattern je kreacijski patern koji omogućava kreiranje porodica međusobno povezanih objekata bez navođenja njihovih konkretnih klasa. Koristi se kada sistem treba biti nezavisan o načinu kreiranja, sastavljanja i predstavljanja objekata iz iste porodice.

Konkretno u našem sistemu ovaj patern se može primjeniti pri kreiranju notifikacija. Sistem podržava obavještenja putem e-maila i aplikacije (vidljivo iz enum klase Obavijest).
Abstract Factory bi mogao kreirati kompletnu „notifikacionu porodicu“ (npr. INotifikacija, ISablonPoruke, IKanalSlanja) za različite tipove obavještenja.

BUILDER PATTERN

Builder pattern je kreacijski patern koji omogućava postepeno konstruisanje kompleksnih objekata, odvajanjem procesa izgradnje od same reprezentacije objekta. Koristi se kada objekat ima mnogo opcionalnih ili međusobno zavisnih dijelova, kako bi se izbjegla komplikacija kroz veliki broj konstruktora.

U našem sistemu, ovaj patern se može koristiti pri kreiranju objekta aukcije. Objekat Aukcija ima više parametara: umjetnina, početna cijena, trenutna cijena, datum početka i završetka, status, kupac. Koristeći Builder pattern, možemo fleksibilno graditi aukciju korak po korak (npr. setUmjetnina(), setPocetnaCijena()...), bez pretrpavanja konstruktora.