Kreacijski paterni

U ovom sistemu dva paterna koje je pogodno implementiratu su:

- Singleton,
- Factory Method

U nastavku je dato objašnjenje gdje i zašto bi se primjenili ovi paterni.

Singleton

Singleton patern je koristan za klasu Korisnik u situacijama kada u aplikaciji u svakom trenutku postoji samo jedan trenutno aktivan (ulogovan) korisnik. Korištenjem Singletona možemo omogućiti globalni pristup toj jednoj instanci korisnika, bez potrebe da se objekat stalno prosleđuje kroz funkcije ili klase. Samim tim možemo omogućiti samo jednu konekciju sa bazom u pokrenutoj aplikaciji na uređaju. Ovim bi postigli očuvanje resursa, umjesto da se svaki put otvara nova konekcija, koristi se jedna zajednička instanca.
Još jedan primjer upotrebe bi bio korištenjem za logiku slanja Newslettera da bi izbjegli kreiranje više instanci odjednom. Ovako samo jedan objekt šalje newsletter svima i spriječava se da više modula paralelno šalje isti sadržaj.

FactoryMethod

Factory Method je dizajnerski obrazac koji omogućava kreiranje objekata bez navođenja njihove konkretne klase, već preko metode koja odlučuje koji tip objekta da instancira na osnovu proslijeđenih parametara. Može se primijeniti na više načina i na različitim klasama. Kao prvo može se primjeniti na klasama Korisnik, Recept i pri kreiranju instanci i izmjena već postojećih instanci. Bez ovog paterna za kreiranje svake insance ili modifikacije iste morali bi pozvati svaku pojedinačnu klasu, dok ovako na osnovu proslijeđenih parametara pozivamo samo metodu. Time izbjegavamo ponavljanje unutar koda i “čišći” kod.

Abstract Factory

Kako svaki tip korisnih ima razlika u pogledu pomoću ovog paterna kreiramo povezane objekte za svakog tipa. Te razlike mogu uključivati sve od dugmadi do dostupnosti pogleda. Samim time vidimo otkud dolazi naziv Abstract factory, uvodimo nivo abstrakcije u aplikaciju za razlikovanje pojedinačnih u našem slučaju interfejsa u API sloju.

Builder

Builder pattern koristimo za sklapanje objekata složenih klasa. Unutar naše aplikacije jedine klase za koje se može reći da su složenije su korisnik i recept. Složenost ovih klasa i nije toliko ozbiljna da bi zahtijevalo primjenu ovog paterna mada bi mogla doprinjeti dodatnoj kontroli pri kreiranju instanci ovih klasa.

Prototype

Prototype pattern jednostavnim riječima služi za kopiranje postojećih objekata. Možda u nekim specifičnim slučajevima mogla bi se pronaći primjena za ovaj pattern, kao npr. da imamo dva identična recepta gdje se razlikuju začini i neki sastojci ali generalno nema velike primjene.