Strukturalni paterni

Vrste strukturalnih paterna:

1. **Adapter** - omogućava širu upotrebu već postojećih klasa, u situacijama kada nam je potreban drugačiji interfejs već postojeće klase a ne želimo mijenjati klasu koristimo Adapter.

   Zbog potrebe za verifikacijom korisničkog računa pomoću ličnih dokumenata, slanjem CV-a ili drugih vrsta datoteka, korisnicima će biti omogućeno da postavljaju dokumente sa različitih izvora – računara (desktopa), cloud-a, pa čak i drugih vanjskih sistema.

   S obzirom na to da ovi izvori koriste različite mehanizme za pristup i učitavanje fajlova, nameće se potreba za implementacijom logike koja će se baviti obradom takvih datoteka. Ovdje se može primijeniti Adapter (strukturalni) patern, koji omogućava da se svi različiti izvori fajlova "zamaskiraju" iza jedinstvenog interfejsa. Na taj način aplikacija obrađivati datoteke, bez potrebe da zna sa kojeg izvora dolaze.

2. **Facade** – osigurava više pogleda visokog nivoa na podsisteme čija je implementacija skrivena od korisnika.

   Da ne bismo pretrpavali kontrolere raznim metodama za oglase(objava, prijava, pregled, itd.), plaćanje(razni načini plaćanja), recenzije i drugih funkcionalnosti. Možemo koristiti Facade strukturalni pattern, koji će dosta ovih metoda objediniti iza jedne jedinstvene metode koju samo dodamo u kontroler.

3. **Decorator** - omogućava dinamičko dodavanje novih elemenata i funkcionalnosti postojećim objektima.

   Standardni oglas bismo mogli proširiti tako da bude promovisani, hitan, ima rok za prijavu, itd. Za ovakve nadogradnje klase najbolje je koristiti Decorator strukturalni patern.

Svaka dodatna funkcionalnost se implementira kao dekorator koji koristi isti interfejs kao i osnovni oglas, čime se postiže fleksibilnost u proširenju ponašanja klase.

4. **Bridge** - omogućava odvajanje apstrakcije i implementacije neke klase, tkd. Klasa može posjedovati više različitih apstrakcija i više različitih implementacija.

   U našem slučaju apstrakcije su načini plaćanja(pretplata, polog, itd.), a implementacije su platforme na kojima će se vršiti transakcije(PayPal, bankovni račun, kreditna kartica, itd.). Pomoću Bridge strukturalnog paterna, svaki način plaćanja i platforma na kojoj se vrši plaćanje se mogu kombinirati, a da nema potrebe za novim klasama što omogućava lakše održavanje koda.

5. **Composite** – koristi se za upravljanje grupe objekata.

   U našem projektu ovaj patern konkretno možemo primjeniti tako da omogućimo da jedan oglas sadrži više podoglasa. Na taj način, korisnik može kreirati složeni oglas koji se sastoji od više pojedinačnih poslova, a sistem ih tretira jednako prilikom prikaza i obračuna cijene.

6. **Proxy** - koristimo ga kada želimo kontrolisati pristup nekom objektu.

   Ovaj patern bismo mogli iskoristiti za sljedeće funkcionalnosti:

   Keširanje - stavlja u neku keš memoriju rezultat skupih operacija poput pregleda statistike

   Logiranje - bilježi sve radnje korisnika nad oglasima, što bi se moglo iskoristit za statistiku koju će administrator pregledati

   Kontrola pristupa - provjerava korisničke dozvole prije nego što se pošaje zahtjev za neku radnju(npr. izmjena oglasa, pristup privatnim podacima itd.)