

Strukturalni Paterni

Mi ćemo za naš projekat koristiti samo Facade i Decorator pattern-e.

Facade Pattern

Facade patern služi kao prednji interfejs koji maskira složeniji kod tj. skriva složenost većeg sistema i klijentu pruža jednostavniji intefejs. Ovaj patern se koristi kada imamo neki kompleksan sistem koji ima mnogo međuzavisnih klasa ili kojem je nepoznat izvorni kod.

Facade Pattern ćemo primijeniti u Revalb sistemu kako bismo pojednostavili složene operacije pretrage albuma. Trenutno, korisnici mogu pretraživati albume po različitim kriterijima (žanr, ocjena, naziv, artist), što zahtijeva interakciju s više klasa i logike. Ovo može dovesti do neorganiziranog koda i otežati održavanje.

- **Klase:**
 - PretragaFacade – Centralizuje operacije pretrage (po žanru, ocjeni, artistu).
 - PretragaPoZanru, PretragaPoOcjeni – Klase koje implementiraju logiku.

Decorator Pattern

Decorator patern služi da omogući da se dinamičko ponašanje dodaje za pojedinačne objekte i da isto ne utiče na ponašanje drugih objekata iz iste klase. Također omogućava i da se funkcionalnost podijeli između klasa u zavisnosti od područja interesa. Ovaj patern naslijeđuje originalnu klasu, ali se ne oslanja na nasljeđivanje prilikom dodavanja novih atributa i objekata, već kreira nove.

Decorator Pattern ćemo koristiti u našem sistemu Revalb kako bismo omogućili dinamičko proširenje funkcionalnosti recenzija bez mijenjanja postojeće klase Recenzija. Trenutno, recenzije podržavaju samo tekstualne komentare i numeričke ocjene, ali želimo dodati podršku za različite formate poput audio zapisa, video komentara ili čak linkova na eksterne platforme (npr. Spotify, YouTube).

Preostali paterni koje nećemo koristiti su:

Adapter Pattern

Adapter Pattern je strukturni dizajn obrazac (design pattern) koji omogućuje suradnju između **nekompatibilnih interfejsa** tako što "prevodi" zahtjeve jednog sučelja u format koji drugi objekt može razumjeti. Funkcionira kao **most između dvije klase koje inače ne bi mogle komunicirati**, bez mijenjanja njihovog izvornog koda.

Bridge Pattern

Bridge Pattern je strukturni dizajn obrazac (design pattern) koji **odvaja apstrakciju od njezine implementacije**, omogućavajući im da se mijenjaju nezavisno jedna od druge. Glavna svrha je **smanjenje krutih hijerarhija nasljeđivanja** i povećanje fleksibilnosti koda.

Composite Pattern

Composite Pattern je strukturni dizajn obrazac (design pattern) koji omogućuje **tretiranje pojedinačnih objekata i njihovih kompozicija (grupa) na isti način**. Koristi se za reprezentaciju hijerarhijskih struktura gdje se pojedinačni elementi i cijele grupe elemenata mogu koristiti identično.

Proxy Pattern

Proxy Pattern je strukturni dizajn obrazac koji djeluje kao zamjenski objekt između klijenta i stvarnog objekta. Njegova glavna svrha je kontrola pristupa i dodavanje dodatne funkcionalnosti bez mijenjanja originalnog koda. Proxy može optimizirati performanse kroz keširanje, omogućiti odgođeno učitavanje resursa ili osigurati zaštitne mehanizme poput provjere ovlasti. Na primjer, u web aplikacijama proxy može filtrirati zahtjeve prije nego što im dozvoli pristup osjetljivim podacima. Ovaj obrazac održava čist kod dok pruža fleksibilnost u upravljanju pristupom objektima.