SOLID PRINCIPI

Ovaj dokument služi kao pojašnjenje kako su svi SOLID principi zadovoljeni u sistemu DMovies.

1. Single Responsibility Principle – Princip pojedinačne odgovornosti

Princip je zadovoljen, kako svaka postojeća klasa posjeduje jednu, pojedinačnu odgovornost. Najjednostavnije je pojasniti preko klasa što se tiču filmova i povezanih informacija. Tu je klasa Movie koja sadrži određene linkove za stream i download, ali ostale detaljnije informacije su smještene u drugim klasama, odnosno klasa MovieInfo čija je odgovornost upravo ta da dostavi određene informacije o samom filmu, kao što su glumci, kratki opis i slično. Te pored njih, tu je i CommentRating klasa koja je opet povezana sa filmom, ali njena jedina uloga jeste prikaz recenzija na filmove. Na sličan način su i ostale klase zamišljene kao klase sa pojedinačnim odgovornostima.

1. Open Closed Principle – Otvoreno zatvoren princip

Ovaj princip je također zadovoljen. Svaka klasa je zamišljena tako da iako vršimo promjene nad njom, odnosno ako je nadograđujemo, nećemo biti u mogućnosti da je mijenjamo, to jest da mijenjamo svrhu njenog postojanja. Klase su spremne da budu proširene određenim funkcionalnostima, ali ne i na izmjenu. Ono što je olakšavajuća okolnost jeste da je tema projekta poprilično statična i trenutno se ne može predvidjeti previše situacija u kojima će uopće doći do potrebe proširenja. To jest uzimajući za primjer filma, sve informacije koje neki film posjeduje trenutno, posjedovat će i kasnije, nema nekih scenarija u kojima bi se slične izmjene mogle očekivati.

1. Liskov Substitution Principle – Liskov princip zamjene

Princip zamjene se može prikazati preko najočiglednijeg primjera, a to je apstraktna klasa User, te izvedene klase Critic, Editor, Administrator i RegisteredUser. Svaka izvedena klasa može da bude zamijenjena svojom klasom koja joj je predak. Odnosno svaku navedenu izvedenu klasu koja je navedena u prethodnoj rečenici, možemo zamijeniti sa User. Ovo je moguće jer i kritičar, editor, admin i registrovani korisnik su zapravo svi korisnici samog sistema, odnosno user-i.

1. Interface Segregation Principle – Princip izoliranja interfejsa

Kako nijedna klasa nije zamišljena da bude debela klasa, odnosno da ima ogroman broj metoda ili atributa, tako da je to već dobar znak da bi ovaj princip bio ispunjen. Klijent sistema će imati pristup samo onim metodama koje su mu zaista potrebne, odnosno interfejsi su tako razdvojeni da je svaki od njih cjelina koja predstavlja skup mogućih stvari koje klijent može da uradi. Jasan primjer su klasa Movie i CommentRating, jer iako usko povezane, takve interfejse treba razdvojiti i jesu razdvojeni. Obzirom da CommentRating iako se odnosi na film, ne treba da bude dostupan izmjeni i klijentima koji su povezani specifično sa klasom Movie. Na taj način je to realizirano.

1. Dependency Inversion Principle – Princip inverzije ovisnosti

Zaista moduli visokog nivoa ne zavise od modula niskog nivoa, odnosno sve klase koje su izvedene zavise od svoje apstraktne klase, ali ne i obrnuto. User nikako ne zavisi od izvedenih klasa, jer izmjene u takvim klasama ne interesuje klasu User, obzirom da su u njoj su prisutne samo nužno potrebne stvari, koje će uvijek morati biti tu, odnosno neće zavisiti od nekih promjena u izvedenim klasama.