SOLID principi

1) *Single Responsibility Principle* – po ovom principu, svaka klasa treba imati samo jedno zaduženje, samo jedan razlog za promjenu. Ako pogledamo dijagram, vidjet ćemo da svaka klasa ima samo ono što joj je zaista neophodno i da sadrži samo one informacije koje su bitne za tematiku kojom se „bavi“. Recimo, klasa RegistrovaniKorisnik sadrži samo osnovne informacije o korisniku kao što su njegova lozinka, username, e-mail adresa, te njegov status (je li administrator sistema ili ne), te konstruktor, gettere i settere bez kojih bi bilo nemoguće pristupiti ovim informacijama. Klasa Film također samo nosi informacije o filmu, kao i metode za pristup istima, dok su za ostale operacije kao što su pregled, brisanje, dodavanje novih ili modifikacija postojećih filmova zadužene druge klase.
2) *Open Closed Principle* – ovaj princip zahtijeva da svaka klasa bude otvorena za nadogradnju, ali zatvorena za promjene, odnosno da dodavanje novih atributa i funkcionalnosti jednoj klasi ne izaziva potrebu za izmjenama na nekim drugim klasama s kojima je prva klasa u vezi. Na dijagramu vidimo da među klasama između ostalog postoje veze agregacije i kompozicije koje označavaju da jedna klasa sadrži atribut koji je tipa druge klase, ili čitavu kolekciju takvih objekata. Uzmimo za primjer klasu Film koja kao atribut ima listu objekata koji su tipa TerminPrikazivanja. Ukoliko bismo u klasu TerminPrikazivanja dodali bilo šta, to se ne bi odrazilo na klasu Film, s obzirom na to da se klasa Film koristi samo već postojećim operacijama i atributima ove klase.
3) *Liskov Substitution Principle* - ovaj princip zahtijeva da podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima, odnosno da se na svim mjestima gdje se koriste objekti tipa izvedene klase

mogu koristiti objekti tipa bazne klase, a kako u projektu nema nasljeđivanja svakako je zadovoljen.

4) *Interface Segregation Principle* – princip izoliranja interfejsa zahtijeva da korisnik ne ovisi o metodama koje neće koristiti. Poštivanje ovog principa često krše „debele“ klase odnosno klase koje imaju previše metoda. Kako nismo imali niti jednu klasu s toliko metoda, nismo vidjeli potrebu ni za izdvajanjem interfejsa, samim time smatramo da je ovaj princip zadovoljen.
5) *Dependency Inversion Principle* - da je ovaj princip ispoštovan lako se vidi iz dijagrama klasa. Kako nije korišteno nasljeđivanje, nema ni strelica koje iz jedne konkretne klase vode prema nekoj apstraktnoj, ni prema interfejsu, a ni prema nekoj drugoj konkretnoj klasi pa tako niti jedna klasa sigurno ne ovisi o nekoj drugoj koja bi se trebala često mijenjati.