Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za računarstvo i informatiku

Objektno-orjentirana analiza i dizajn

Projekat: GameHub

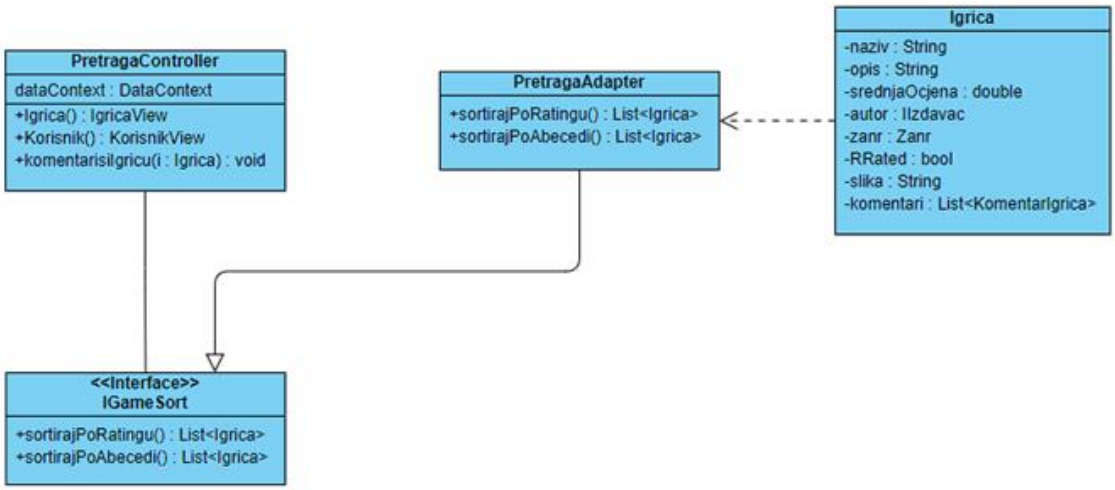
Strukturalni paterni

Grupa 4 – Oštri Momci (#momci):
Elvedin Smajić
Vedad Grbo
Tarik Beganović

1.) Adapter pattern

Za naš sistem je uobičajeno da se igrice dodaju putem sistema direktno u bazu. Dakle, Developer ili Gaming Kompanija postavljaju podatke o igri i one se tako dodaju u bazu. Ovaj pattern bismo mogli iskoristiti kako bismo putem nekog WebAPI-ja uspjeli prikupiti informacije o svim igrama i tako ih dodavati u bazu, a ne samo putem sistema. Recimo, prikupi se neki JSON od igrice koji bi sadržavao podatke o njoj, a mi bismo uzeli iz toga one podatke koje su potrebne za naš sistem. Međutim, najvjerovatnije radi jednostavnosti koristićemo samo nekoliko igara koje ćemo mi dodati u bazu, jer bi onda imali previše igara i korisnika sistema.

Ovaj pattern ćemo iskoristiti, ipak, na drugi način. Adapter pattern ćemo iskoristiti za sortiranje igrica po različitim kriterijima. Za sad smo planirali dvije vrste sortiranja: po abecedi i ratingu, ali uvijek postoje mogućnosti da se u budućnosti nešto doda.



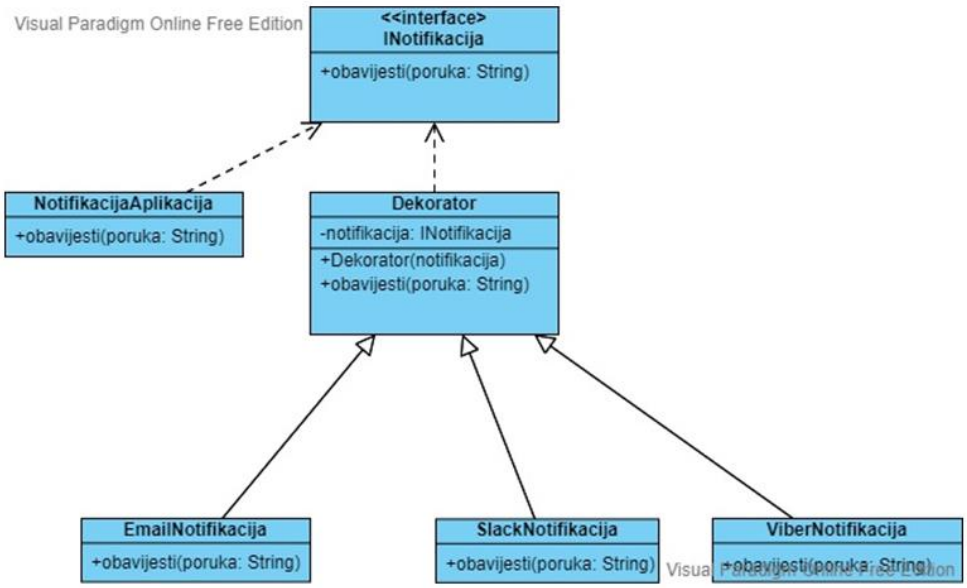
2.) Facade pattern

Kolo sreće je jedan primjer gdje bismo mogli iskoristiti ovaj pattern. Korisnik sve što treba da uradi jeste da ima dovoljan broj kredita i da pokrene kolo na dugme. Međutim, u pozadini implementacija bi mogla biti dosta komplikovana. Neke od stvari koje bi se trebale implementirati su:

- provjera količine kredita
- iscrtavanje kola sa nagradama
- animiranje kola tako da se počne vrtiti kada korisnik klikne na dugme i da se zaustavi nakon određenog vremena
- učitavanje rezultata okretanja
- generisanje nagradnog koda
- slanje nagrade korisniku u obliku notifikacija itd.

3.) Decorator pattern

Za decorator pattern smo odabrali neku sigurnu rezervu u slučaju da ne uspijemo implementirati neke od dva odabrana patterna. Pomoću decorator patterna možemo implementirati više vrsta notifikacija: e-Mail, Viber, Slack... Ukoliko budemo u mogućnosti, ovo ćemo ubaciti u našu aplikaciju.



4) Bridge pattern

U prethodnim taskovima smo spomenuli način objavljivanje igrice putem interfejsa IIzdavac koji se odnosi na Gaming Kompaniju i na Developera. Tu bismo mogli primijeniti ovaj pattern kako bismo omogućili objavljivanje igre za obje vrste korisnika.
Implementacija objavljivanja igre za obje klase je slična, ali se razlikuje u nekim sitnicama (npr. Developer treba potvrdu Administratora kako bi se objavila igrica).
Isto postoji mogućnost ubacivanja bridge patterna za različit način računanja nagrada pri ispunjavanju taskova i slično (npr. da aktivniji korisnici dobiju više MouseCredits).

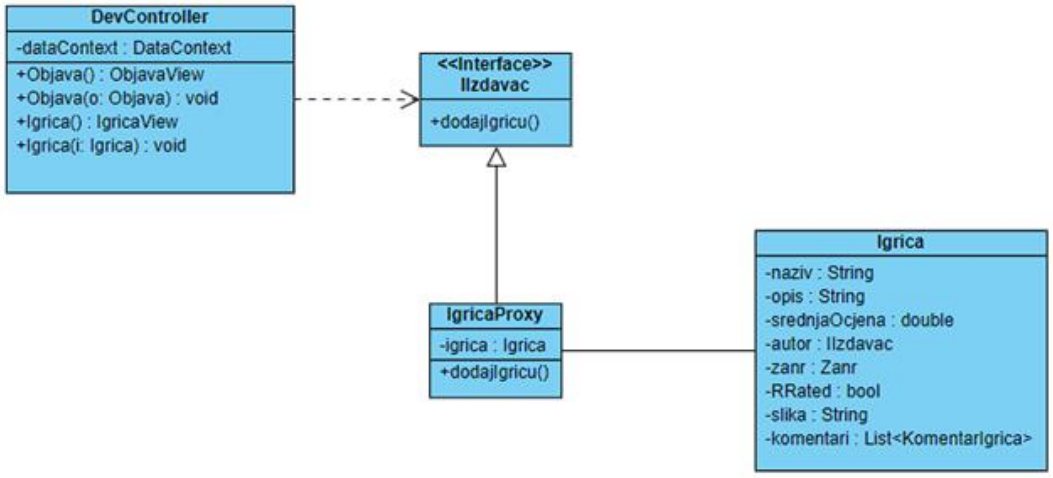
5) Composite pattern

Composite pattern bismo mogli iskoristiti kod klase Objava gdje imamo kao komponente više različitih klasa:
Korisnik koji je objavio objavu

- Igrica koja je na slici
- slika
- lista Komentara za tu objavu
- lista Korisnika koji su lajkovali objavu.

6) Proxy pattern

Proxy pattern ćemo najvjerovatnije koristiti pri implementaciji objave igrice za Developera. Developer treba tražiti zahtjev od Administratora za kreiranje nove igrice u bazi sistema. Isto tako, registracija korisnika bi mogla biti implementirana koristeći proxy pattern.



7) Flyweight pattern

Flyweight pattern bismo mogli iskoristiti kod implementiranja različitog izgleda korisničkog profila. Svaki korisnik će imati neki defaultni izgled svog profila, a imao bi i pravo promjene na neki alternativni izgled profila koji bi promijenio boju elemenata na korisničkom profilu.