STRUKTURALNI PATTERNI ZA ULAZNICE.COM

1) ADAPTER PATTERN

Adapter pattern se primarno koristi da bi proširili funkcionalnosti i upotrebu već postojeće klase. Ovaj pattern možemo iskoristiti prilikom nagrađivanja lojalnosti korisnika, tj. kod praćenja koliko svaki korisnik ima ostvarenih kupovina u klasi “NagradnaIgra”. Adapter bismo korsitili za soritanje po vrijednosti kupovine u BAM, u našem slučaju bismo gledali koji korisnik ima najviše ostvarenih kupovina i tako bismo ih sortirali po opadajućem poretku.

2) FACADE PATTERN

Ovaj pattern se koristi kada želimo “sakrijemo” komplikovani sistem krajnjim korisnika. To jest, da korisitmo samo mali dio funkcionalnosti kompleksnog sistema. Konkretno, u našem projektu to je riječ o generisanju QR koda na kojem smo i zasnovali čitav sistem. Za generisanje, i općenito za upravljanje QR kodovima će biti potrebna i zasebna klasa.

3) DECORATOR PATTERN

Ovaj pattern se koristi kada ne želimo praviti ogroman broj novih klasa koje su varijacija postojećih klasa koje već imamo. Ovim patternom želimo koristiti klase koje već postoje, ovaj pattern se može iskoristiti u našem sistemu tako što bismo omogućili korisnicima koji imaju svoj profil da promijene svoje korisničke podatke (email, password, i što je najvažnije dodati ili ukloniti bankovni račun). U našem slučaju za vrstu plaćanja ne možemo primijeniti ovaj pattern jer različiti načini plaćanja zahtijevaju različite algoritme.

4) BRIDGE PATTERN

Bridge pattern nam omogućava da se iste operacije primjenjuju nad različitim podklasama. Na taj način se vrši odvajanje apstrakcije i implementacije tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Bridge pattern bi iskoristili prilikom odabira načina plaćanja. Npr, svaki od različitih načina plaćanja (gotovina, vaučer, kartica) zahtjeva različite algoritme obrade tih podataka. Za ovaj princip bismo implementirali 3 različite klase (Vaucer, Gotovina, Kartica) koje bi naslijedile apstraktnu baznu klasu Placanje.

5) PROXY PATTERN

Uloga proxy paterna je osiguranje pristupa resursima. Primjena ovog paterna jeste prilikom prijave korisnika na profil. Prilikom logiranja korisnika i/ili administratora na sistem vrsi se autentifikacija podataka i na taj način je zastićen pristup.

6) COMPOSITE PATTERN

Composite patern opisuje grupu objekata koji se tretiraju na isti način kao pojedinačna instanca istog tipa objekta. Namjera kompozita je da "komponira" objekte u strukture stabla koje predstavljaju hijerarhiju dio-cjelina. Zadovoljili smo principe ovog patterna jer naš sistem zasnovan na ulaznicama, koje se sastoje iz više različitih aspekata. To su: kupac/naručilac (u našem slučaju korisnik), tip manifestacije, njena lokacija, lokacija na tribini/parteru, cijena (koja će varirati ukoliko se korisnik odluči na kupovinu više njih) odgovarajuće QR kodove, te jedan od 3 načina kojim se mogla obaviti kupovina (gotovina, vaučer, kartica). Svaki od ovih aspekata će „staviti u kutiju“ (u našem slučaju kutija=karta) njihov glavni atribut, i tek nakon svih tih operacija ulaznice će biti izgenerisane na način da će na ulaznici biti navedeni svi ti podaci.

7) FLYWEIGHT PATTERN

Flyweight pattern omogućava programima da podrže ogromne količine objekata održavajući njihovu potrošnju memorije niskom. Uzorak to postiže dijeljenjem dijelova stanja objekta između više objekata. Ovaj pattern bi se u našem sistemu manifestirao kroz pretragu kupljenih ulaznica. Prilikom pretrage ulaznica korisniku bi se samo prikazao broj ulaznica koje je kupio za određeni događaj. Npr za kulturne manifestacije je kupljeno 17, a za sportske 9 ulaznica. Tek klikom na njih bi se prikazale kupljene ulaznice, međutim bez imena kupca (što se podrazumijeva jer korisnik pretražuje svoje kupljene ulaznice, ime kupca bi služilo samo za informaciju koju bi administrator imao na raspolaganju), QR koda, i načina na koji je ulaznica kupljena. Na taj način bi postigli uštedu podataka. Ili drugi način da se prilikom pretrage korisniku prikaže samo pdf ili jpeg fajl ulaznice koji se ranije generisao u trenutku kada je kupovina iste bila validirana. Tj. u ovom slučaju podaci se u tom trenutku ne bi „izvlačili“ iz baze, već bi korisniku bio proslijeđen samo odgovarajući fajl.