Univerzitet u Sarajevu
Elektrotehnički fakultet
Sarajevo

Sedmi projektni zadatak

Strukturalni paterni

Objektno orijentisana analiza i dizajn

Naziv grupe: Booking
Članovi: Ajdin Dželo
Nejra Vatrić
Amna Glamoč
Adna Bajramović

Sarajevo, maj 2025

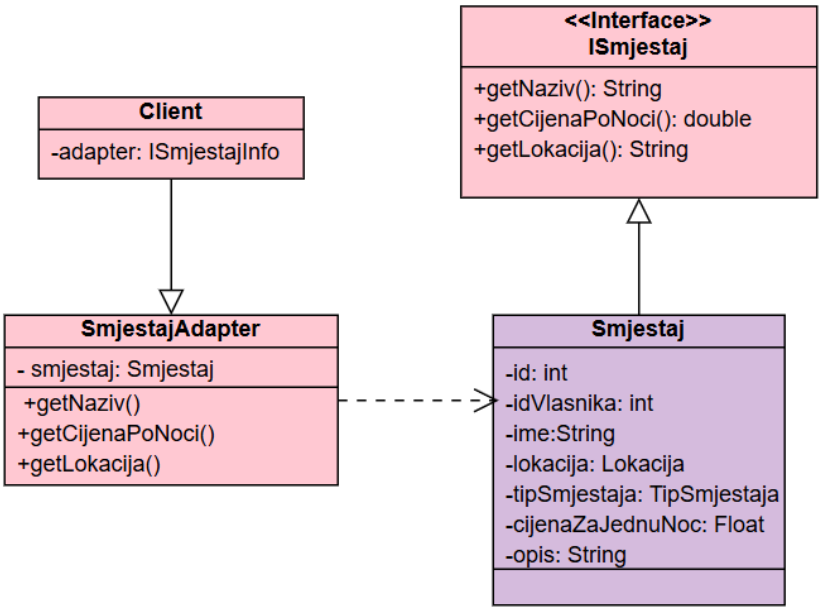
Strukturalni paterni u booking sistemu

Implementacijom strukturalnih paterna proširujemo funkcionalnosti booking sistema i poboljšavamo održivost i sigurnost sistema. Svaki od njih rješava specifičan problem. U nastavku su opisani: Adapter, Facade, Bridge, Proxy, Composite i Decorator.

Adapter

U našem sistemu za rezervaciju smještaja, osnovne klase kao što su Smjestaj, Korisnik, Rezervacija i Recenzija predstavljaju kompleksne entitete povezane međusobnim relacijama. Iako je to optimalno za internu logiku sistema, često se javlja potreba da se ovakvi podaci izlože vanjskim aplikacijama: turističkim agencijama ili mobilnim aplikacijama koji žele prikazati ili analizirati podatke o dostupnom smještaju. Međutim, vanjski sistemi obično očekuju jednostavne, standardizovane formate, sa osnovnim informacijama o smještaju (npr. "naziv", "cijenapoNoci", "lokacija"), koji su jednostavniji od naše unutrašnje strukture.

Da bismo zadovoljili ovu potrebu, koristimo Adapter patern. Tri osnovna elementa u paternu su Client, koji predstavlja bilo koju vanjsku aplikaciju koja će koristiti naš interfejs, Adaptee, odnosno postojeći sistem i Adapter, odnosno klasu npr SmjestajAdapter koja implementira interfejs i koristi instance Smjestaj, te ih transformiše u format koji se može razumjeti. Te naravno imamo ciljani interfejs koji definiše format koji vanjska aplikacija očekuje.



Facade

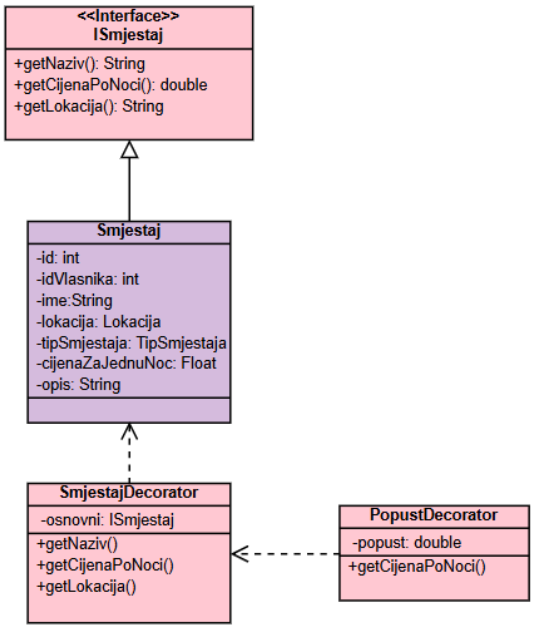
Kako naš sistem sadrži više podsistema, potreban nam je jednostavniji interfejs koji omogućuje Facade pattern jer naprimjer ako se želi rezervisati smještaj mora se validirati korisnik, provjeriti dostupnost, zatim rezervirati itd, što predstavlja prevelik broj poziva različitim klasama.

Kreirat ćemo klasu koja objedinjuje sve te pozive u jednostavan interfejs koji sadrži metode kao pretraziSmjestaj ili rezervisiSmjestaj, iza kojih se kriju složeni pozivi prema više klasa (pretraga smještaja, filtriranje, validacija korisnika, provjera dostupnosti), što smanjuje zavisnost i odvaja nepotrebne kompleksnosti sistema.

Decorator

U sistemima kao što je Booking, često postoji potreba da se funkcionalnost objekata proširuje bez mijenjanja postojeće strukture klasa. Na primjer, oznake kao što su Top Rated ili Last Minute, ili dodavanje popusta, podataka o popularnosti...

Kreiramo apstraktnu klasu SmjestajDecorator koja implementira interfejs ISmjestaj i sadrži referencu na osnovni Smjestaj,a zatim pravimo konkretne dekoratore, npr: PopustDecorator koji dodaje informaciju o cijeni s popustom ili PopularnostDecorator , dodaje broj pregleda i oznaku popularnosti. Ovo omogućava fleksibilno proširenje funkcionalnosti bez uticaja na postojeće klase.



Bridge

Bridge pattern omogućava da se iste operacije primjenjuju nad različitim podklasama, te ono što je karakteristično jeste da te operacije imaju jedan zajednički korak, nakon čega je sve različito. Ako naš sistem ima različite vrste smještaja i različite metode plaćanja, ovaj pattern omogućava rezervacija ima referencu na nacinPlacanje, da rezervacija zna za Smjestaj i da korisnik pravi rezervaciju za određeni smještaj sa određenim načinom plaćanja. Na ovaj način odvaja rezervaciju (apstrakciju) i nacinPlacanja (implementaciju).

Proxy

Ovaj pattern služi za dodatnu zaštitu objekata od pogrešne upotrebe, odnosno onemogućava manipulaciju objektima ukoliko određeni uslov nije ispunjen. Kako u našem sistemu recenziju

mogu ostavljati samo prijavljeni korisnici, odličan način da osiguramo taj proces je upravo upotreba Proxy paterna. Ukoliko je korisnik registrovan, moći će izvršiti akciju komentarisanja i ocjenjivanja, a ako nije onda bi mu se ponudila opcija da se ponovo registruje. Korištenjem Proxy paterna kreira se kontrolni sloj koji provjerava da li korisnik ima prava pristupa metodi dodajRecenziju() prije nego što se pozove stvarna implementacija.

Composite

Composite pattern je pogodan u slučaju kada je potrebno obrisati određenog korisnika iz sistema. Kako prijavljeni korisnik ima mogućnost ostavljanja recenzija na objekte, kada njegov profil bude obrisan, trebale bi biti obrisane i sve recenzije koje je on napisao. Klasa koja je direktno povezana sa ovim patterrnom je Korisnik klasa, s obzirom na to da je administrator jedini akter koji ima mogućnost brisanja korisničkih računa.

//opcionalno

Također, u sistemu postoji potreba za hijerarhijskom organizacijom smještaja. Na primjer, jedan objekat može sadržavati više jedinica koje se zasebno iznajmljuju. Implementacija Composite paterna omogućava da se pojedinačni i grupni smještaji tretiraju isto.