Paterni ponašanja - "Upravljanje univerzitetom"

Strategy patern

Ovaj patern je primijenjen kako bi se omogućilo generisanje različitih vrsta **izvještaja** na fleksibilan način. Iako je osnovni mehanizam pokretanja generisanja izvještaja isti, konkretan algoritam (strategija) zavisi od vrste izvještaja koju student odabere. Na primjer, jedna strategija je odgovorna za kreiranje izvještaja o uspjehu, dok druga može generisati potvrdu o prijavljenim ispitima. Ovakav pristup omogućava lako dodavanje novih vrsta izvještaja u budućnosti, bez izmjene postojećeg koda u kontroleru.

State patern

State patern se koristi za upravljanje stanjem objekta notifikacija. Svaka notifikacija može postojati u dva osnovna stanja: nepročitana i pročitana. U zavisnosti od trenutnog stanja, objekat može promijeniti svoje ponašanje ili način na koji se prikazuje u korisničkom interfejsu (npr. nepročitana notifikacija može biti boldirana). Prelazak iz jednog stanja u drugo (npr. kada korisnik klikne na notifikaciju) je jasno definisan, što čini kod organizovanim i predvidljivim.

Observer patern

Observer patern je ključan za sistem obavještavanja. Kada profesor (subjekt) unese novi termin ispita ili objavi rezultate, svi studenti (observeri) koji su prijavljeni na taj predmet automatski dobijaju notifikaciju. Subjekt (profesor) ne mora da zna koji su sve studenti prijavljeni; on samo objavljuje promjenu, a sistem se brine da svi zainteresovani observeri budu obaviješteni. Ovo efikasno razdvaja logiku profesora od logike studenata.

Iterator patern

Ovaj patern se implicitno koristi u cijeloj aplikaciji za prikazivanje kolekcija podataka. Kada kontroler proslijedi listu profesora, predmeta ili studenata ka view-u, Iterator omogućava jednostavno i uniformno prolazak kroz te kolekcije (npr. unutar foreach petlje u Razor pogledu). Prednost je u tome što je kod za prikazivanje podataka potpuno odvojen od interne strukture kolekcije.

Chain of Responsibility patern

Ovaj patern je fundamentalan za sistem autorizacije u aplikaciji, implementiran kroz ASP.NET Core middleware. Kada korisnik pokuša pristupiti određenoj funkcionalnosti (npr. Student/PrijaviIspit), njegov zahtjev prolazi kroz lanac odgovornosti:

1. Prvi handler provjerava da li je korisnik uopšte prijavljen (autentifikovan).
2. Ako jeste, sljedeći handler u lancu ([Authorize(Roles = "Student")]) provjerava da li korisnik ima odgovarajuću ulogu.
   Zahtjev se uspješno izvršava samo ako prođe sve provjere u lancu, što osigurava siguran sistem kontrole pristupa.

Mediator patern

U MVC arhitekturi ove aplikacije, kontroleri (StudentController, ProfesorController, itd.) preuzimaju ulogu mediatora. Oni stoje između korisničkog interfejsa (view) i poslovne logike/podataka (model). Korisnik ne interaguje direktno sa modelima. Umjesto toga, sve akcije idu preko kontrolera, koji prima zahtjev, obrađuje ga pozivajući odgovarajuće servise ili modele, i na kraju vraća odgovarajući view. Ovo smanjuje direktnu zavisnost između view-a i modela, čineći sistem organizovanijim.

Interpreter patern

Interpreter patern pruža mehanizam za definisanje i tumačenje jednostavnog jezika ili formata. U ovoj aplikaciji, njegova primjena se ogleda u validaciji formata broja indeksa prilikom kreiranja novog studenta. Definisan je jasan format: broj indeksa mora sadržavati tačno šest cifara. Interpreter je komponenta koja provjerava da li uneseni string odgovara ovom pravilu. Na ovaj način se osigurava integritet podataka prije upisa u bazu, a patern pruža strukturu za eventualno dodavanje kompleksnijih pravila validacije u budućnosti.