Paterni Ponašanja

Strategy Pattern

Strategy patern omogućava korištenje različitih algoritama bez mijenjanja glavne logike programa. U mojoj aplikaciji *OffroadAdventure*, ovaj patern je primijenjen prilikom obračuna popusta pri kreiranju zahtjeva za rentanje vozila. U zavisnosti od broja vozila i trajanja rentanja, koristi se drugačija logika za izračun popusta, čime se postiže fleksibilnost i jednostavnije održavanje koda.

State Pattern

State patern omogućava da objekat promijeni svoje ponašanje u zavisnosti od trenutnog stanja u kojem se nalazi. U aplikaciji *OffroadAdventure*, ovaj patern je primijenjen kroz model ZahtjevZaRentanje, koji koristi enumeraciju StatusZahtjeva sa stanjima poput NA_CEKANJU, ODOBREN i ODBIJEN. Na osnovu tog stanja određuje se kako će se zahtjev dalje obrađivati i prikazivati korisnicima u interfejsu.

Template Method Pattern

Template Method patern se koristi za definisanje osnovne strukture algoritma, dok se pojedini koraci mogu mijenjati prema potrebi. U mojoj aplikaciji *OffroadAdventure*, ovaj patern je primijenjen kod prikaza komentara, gdje je omogućeno sortiranje komentara po ocjeni rastuće ili opadajuće, bez mijenjanja glavne logike dohvaćanja i prikaza komentara.

Chain of Responsibility Pattern

Chain of Responsibility patern omogućava da se zahtjev prosljeđuje kroz niz objekata dok neki od njih ne preuzme odgovornost za njegovu obradu. U aplikaciji *OffroadAdventure*, ovaj patern je primijenjen kod upravljanja zahtjevima za rentanje, gdje zaposleni prvo obrađuju pristigle zahtjeve, a vlasnik ima mogućnost da nadgleda i upravlja svim zahtjevima. Na taj način omogućena je fleksibilna i hijerarhijska obrada zahtjeva bez direktnog povezivanja svih aktera.

Command Pattern

Command patern omogućava da se zahtjevi enkapsuliraju kao objekti, čime se olakšava njihovo čuvanje, upravljanje i kasnije izvršavanje. U aplikaciji *OffroadAdventure*, ovaj patern se ogleda u načinu na koji korisnik kreira zahtjev za rentanje vozila, koji se zatim čuva i obrađuje od strane zaposlenika ili vlasnika. Time se omogućava fleksibilno upravljanje zahtjevima, kao i kasnija reakcija sistema kroz notifikacije.

Iterator Pattern

Iterator patern omogućava prolazak kroz kolekciju objekata bez izlaganja detalja o njenoj strukturi. U aplikaciji *OffroadAdventure*, ovaj patern se koristi prilikom prikaza listi vozila, komentara i zahtjeva, gdje se elementi kolekcije prolaze i prikazuju korisniku pomoću standardnih foreach petlji. Na taj način se omogućava jednostavan i fleksibilan prikaz podataka iz baze.

Mediator pattern

Mediator patern omogućava centralizovanu komunikaciju između različitih objekata bez njihovog međusobnog direktnog povezivanja. U aplikaciji *OffroadAdventure*, ovaj obrazac se djelimično ogleda u NotifikacijaController-u, koji služi kao posrednik između kontrolera koji iniciraju određene događaje (npr. obrada zahtjeva, plaćanje) i korisnika koji trebaju biti obaviješteni. Na taj način se smanjuje direktna povezanost između komponenti i postiže veća modularnost sistema. Iako ovaj patern nije eksplicitno implementiran kao posebna klasa u aplikaciji, njegova logička struktura je prisutna kroz način na koji se notifikacije posreduju između različitih dijelova sistema.

Observer Pattern

Observer patern omogućava automatsko obavještavanje svih zainteresovanih objekata kada dođe do promjene stanja u sistemu. U aplikaciji *OffroadAdventure*, ovaj obrazac se logički koristi kod sistema notifikacija, gdje se korisnici obavještavaju kada dođe do promjene statusa njihovog zahtjeva za rentanje ili nakon uspješnog plaćanja. Iako nije eksplicitno implementiran kao zaseban patern ili klasa, princip posmatranja i reagovanja na promjene stanja jasno je prisutan kroz mehanizam notifikacija koje se šalju iz različitih dijelova sistema.

Visitor Pattern

Visitor patern omogućava dodavanje novih operacija nad objektima bez izmjene njihovih klasa, što je korisno u sistemima koji često zahtijevaju proširivanje funkcionalnosti nad postojećim strukturama podataka. U aplikaciji *OffroadAdventure*, ovaj patern nije implementiran, jer sistem ne zahtijeva izvođenje dodatnih operacija nad objektima kao što su User, Vozilo ili Komentar, van onih koje su već direktno definisane u kontrolerima.

Interpreter pattern

Interpreter patern koristi se za interpretaciju i evaluaciju izraza definisanih u nekom domenskom jeziku. U aplikaciji *OffroadAdventure*, ovaj patern **nije implementiran**, jer ne postoji potreba za analizom i izvršavanjem posebnih izraza, pravila ili sintakse. Sve funkcionalnosti aplikacije se realizuju direktno kroz programski kod i standardne kontrole, bez upotrebe sopstvenog jezika ili parsera.

Memento pattern

Memento patern koristi se za čuvanje prethodnog stanja objekta, kako bi se po potrebi moglo vratiti u to stanje, bez narušavanja enkapsulacije. U aplikaciji *OffroadAdventure*, ovaj patern nije implementiran, jer sistem ne predviđa funkcionalnosti poput undo/redo mehanizama ili vraćanja prethodnih verzija zahtjeva, komentara ili drugih podataka. Sve akcije u sistemu su konačne i direktno utiču na trenutno stanje objekata.