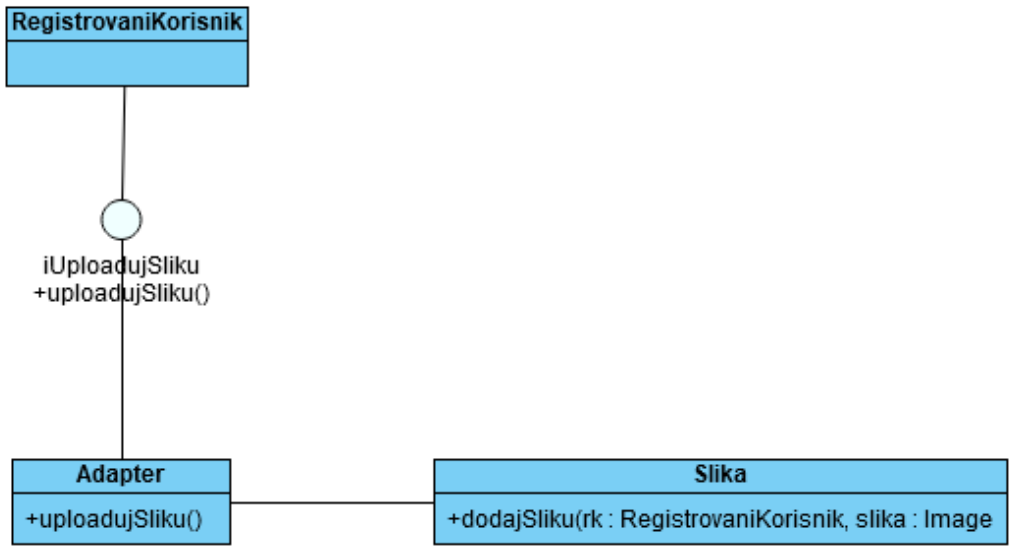
STRUKTURALNI PATERNI

1. ADAPTER

Adapter patern se koristi kada je potreban drugačiji interfejs postojeće klase, a ne želimo mijenjati postojeću klasu.

Na primjer, naš sistem sada predviđa mogućnost ostavljanja recenzija majstorima nakon završenog posla. Recenzije se sastoje od ocjene (enum) i od komentara (string). Ukoliko bismo još dodatno omogućili da korisnik može uploadovati sliku nakon odrađenog posla i ostaviti nju kao recenziju, to bismo mogli izvesti na sljedeći način:



2. FACADE

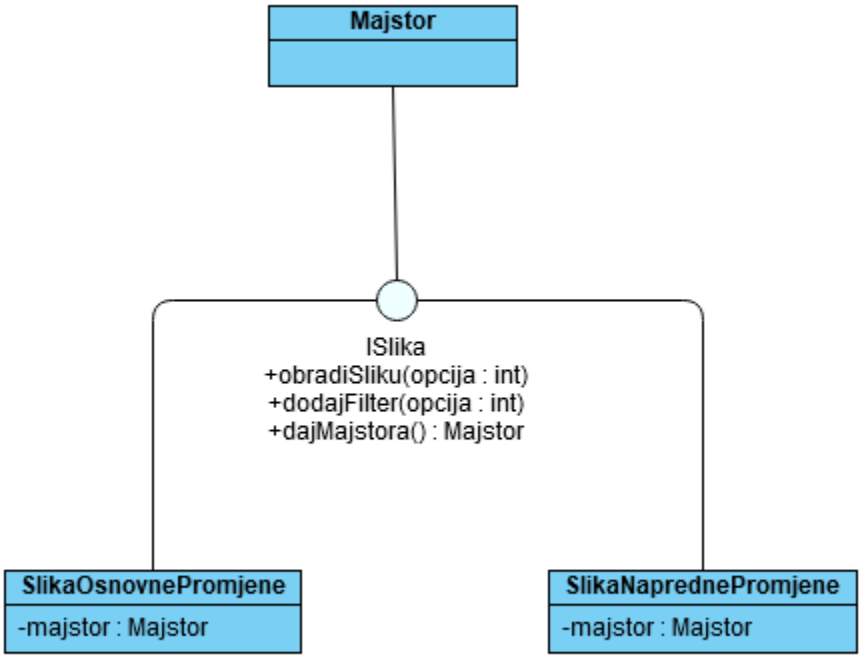
Fasadni patern je strukturalni patern koji krajnjem korisniku dostavlja pojednostavljeni interfejs, koji se odnosi na skup frameworka ili skup kompleksnih klasa, a u cilju pojednostavljenja interakcije između aplikacije i korisnika.
Korisniku nije bitno na koji način se pojedine funkcionalnosti izvršavaju, već mu je bitan samo krajnji proizvod.
U našem sistemu smo to implementirali kod ostavljanja recenzije majstoru. Na primjer,

nakon što korisnik ostavi neku recenziju, majstoru se ta ocjena mora evidentirati i mora se adekvatno izračunati srednja ocjena. Korisnik ne mora znati te detalje.

3. DECORATER

Dekorater patern se koristi kada ne želimo praviti veliki broj podklasa koje predstavljaju kombinacije već postojećih klasa, već želimo iskoristiti postojeće klase u svrhu omogućavanja novih funkcionalnosti.

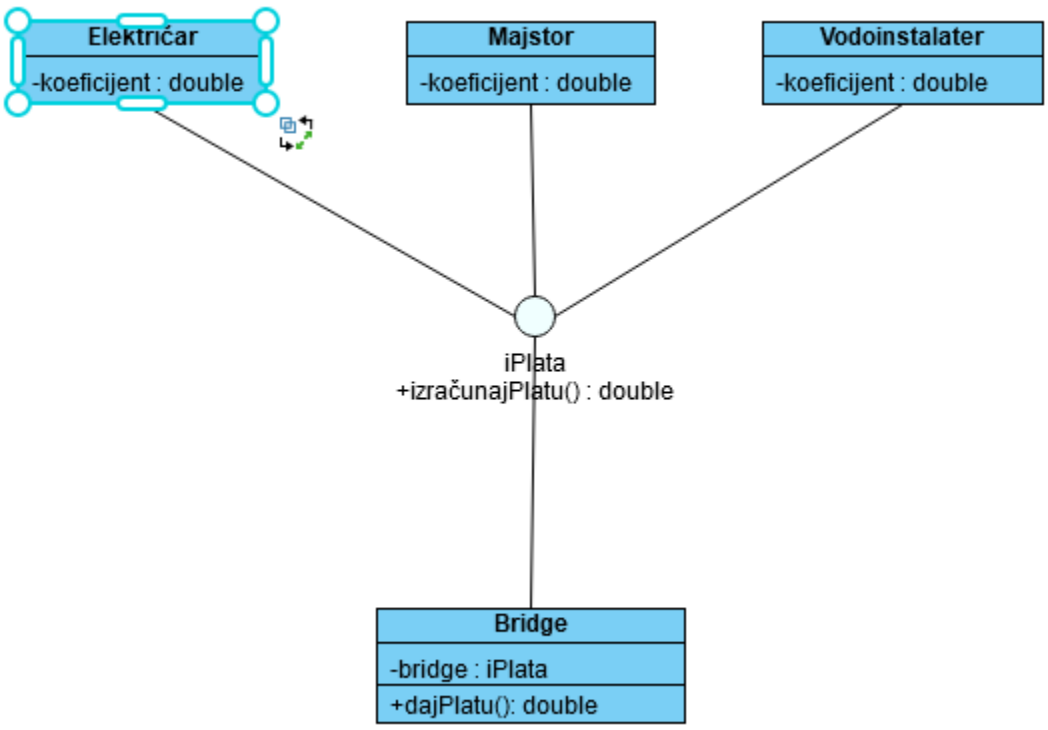
U našem sistemu bi se on mogao izvesti na način da majstori mogu uploadovati profilne slike, ili slike svojih prijašnjih radova kao dokaz o poslovanju. Te slike bi se dalje mogle uređivati u okviru same aplikacije, na sljedeći način:



4. BRIDGE

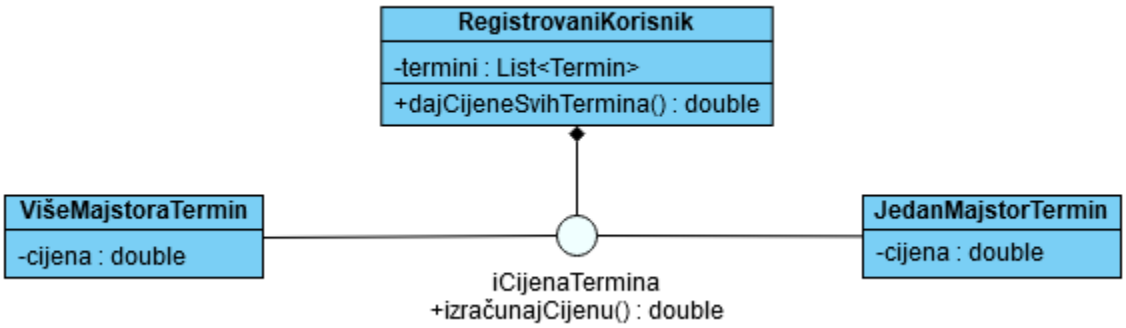
Bridge patern omogućava da se iste operacije primjenjuju nad različitim podklasama. Također se koristi kod izbjegavanja kreiranja novih metoda za postojeće funkcionalnosti. Ono što je karakteristično za ovaj patern je to da te operacije imaju jedan zajednički korak, nakon čega su različite.

Ako bi naš sistem zahtijevao uvođenje dodatnih zaposlenika sistema, njihovu platu bismo mogli računati na sljedeći način:



5. COMPOSITE

Kompozitni patern je jako sličan bridge paternu, ali je jedina razlika što te operacije nemaju nikakvih zajedničkih dijelova. Na primjer, ukoliko bi naš sistem podržavao mogućnost da, osim jednog majstora, na jedan zakazan termin može doći više majstora (u cilju ubrzavanja radova), cijena bi se računala znatno drugačije.



6. PROXY

Proxy patern se koristi kod zaštite pristupa resursima u sistemu. On također ubrzava pristup korištenjem cache objekata.

Ovaj patern je implementiran u naš sistem, da bi osigurali da korisnik može ocjenjivati samo majstora sa čijim je radom zaista upoznat. Da bi to postigli dodali smo interfejs IRecenzijaProxy , kao i klasu RecenzijaProxy koja odobrava korisniku ostavljanje recenzije u slučaju da je koristio usluge datog majstora.

7. FLYWEIGHT

Ovaj patern se koristi kako bi se objekti mogli “reciklirati”, tj. kako bi bili ponovo iskoristljivi. Ovo se radi u cilju smanjenja potrošnje RAM-a.

Što se tiče našeg sistema, to bi se moglo implementirati kod klase TehničkiProblem. Na primjer, ako bi naš sistem omogućavao korisnicima da, pri unošenju problema koje imaju sa aplikacijom, odaberu neku od opcija (npr. 1 - problemi sa ostavljanjem recenzije, 2 - problemi sa zakazivanjem termina, 3 - problemi sa plaćanjem), sajt bi automatski mogao korisniku prikazati spisak potencijalnih rješenja za njegov problem, na osnovu iskustava prethodnih korisnika koji su se izjasnili istom opcijom (tj. da imaju isti problem). To bi značilo da bi sistem u bazi čuvao spisak potencijalnih rješenja tog problema i u zavisnosti od odabrane opcije prikazivao ta rješenja korisnicima. Na taj

način bi se izbjeglo kreiranje novog spiska potencijalnih rješenja tog problema za korisnike koji imaju isti problem.