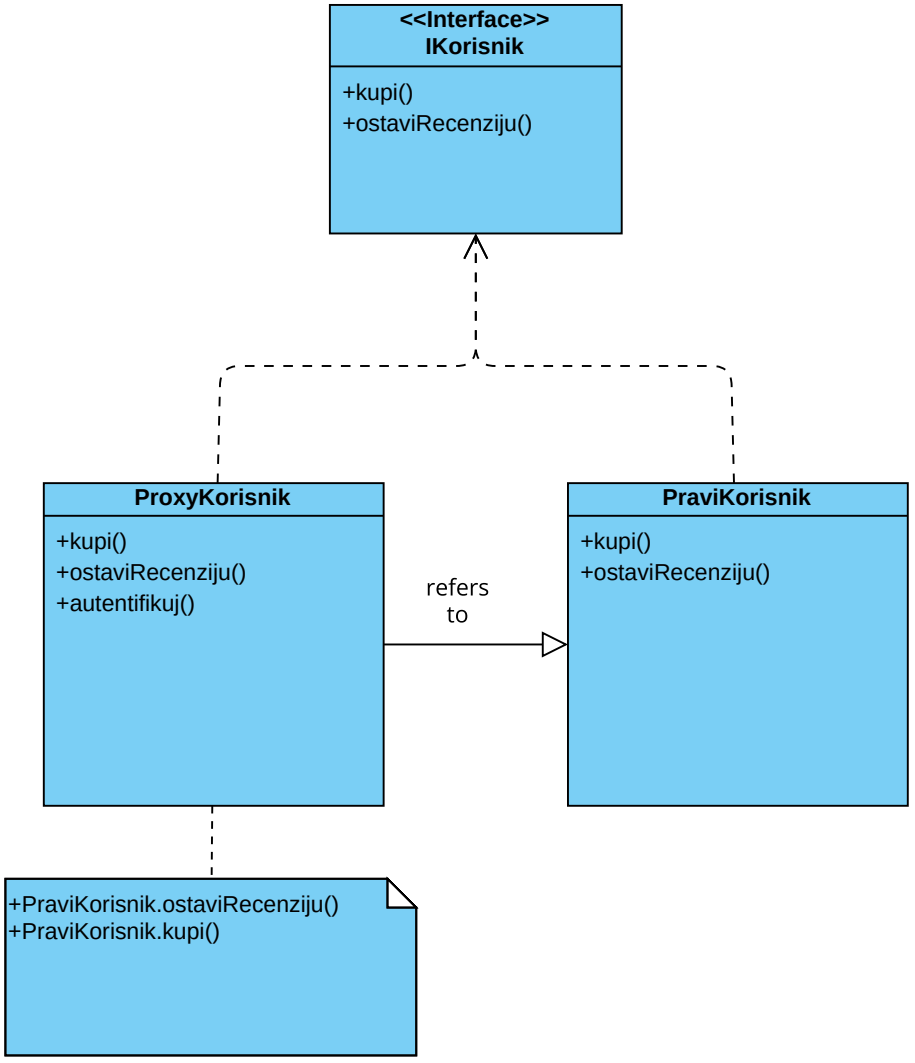
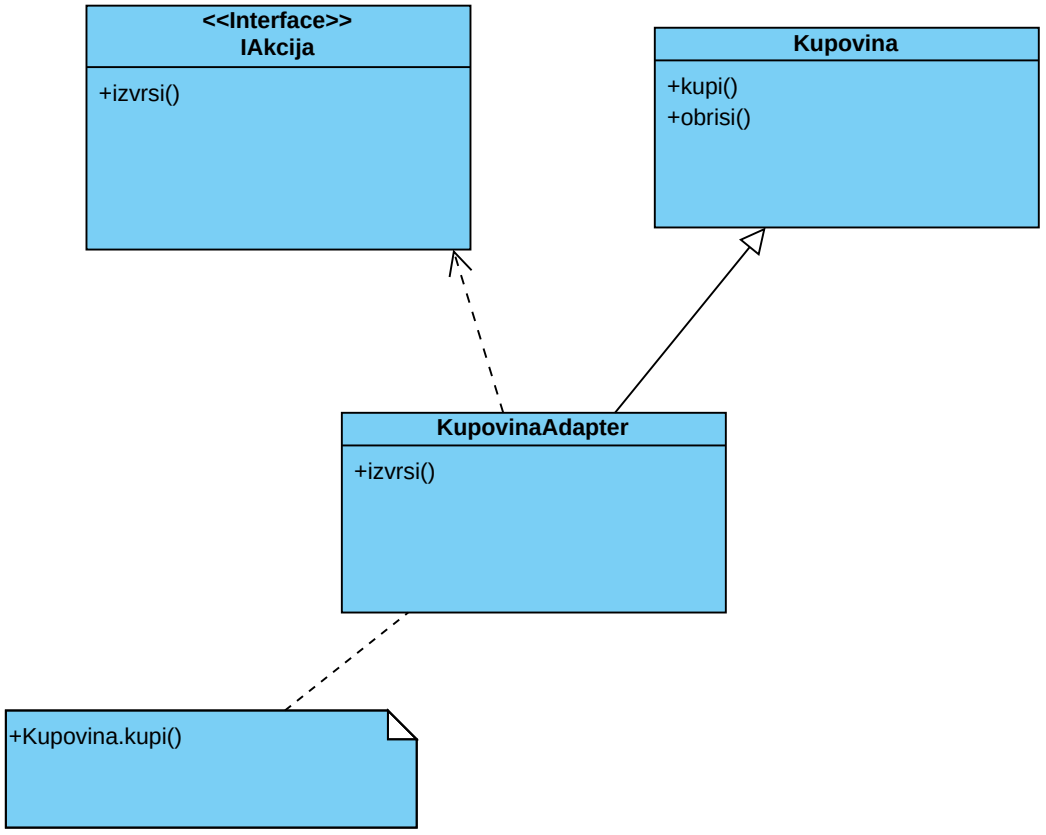
Proxy pattern

U postojeći sistem možemo dodati proxy strukturalni pattern, kako bismo zaštitili
metode koje mogu koristiti samo prijavljeni korisnici (Protected Proxy) i centralizovali
provjeru autentifikacije i autorizacije
Korisnik nakon login-a dobija pristup
nekim operacijama — recenzijama, kupovinama,
bookmarkovima itd.



Adapter pattern

Adapter
U postojeći sistem možemo dodati adapterstrukturalni pattern, kako bismo napravili razliku između kupovine ulaznica za događaj u kojem potrebna i rezervacija mjesta i
događaj na kojem se samo kupuje ulaznica.



3. Bridge

Bridge pattern odvaja apstrakciju od njene implementacije tako da obje mogu nezavisno evoluirati. U ovom primjeru koristi se za slanje obavijesti. Klasa *Obavijest* ne mora znati tačno kako se obavijest šalje – ona samo koristi interfejs *NotifikacijaSender*. Implementacije tog interfejsa mogu uključivati slanje putem emaila, SMS-a ili push notifikacija. Ovo omogućava lako dodavanje novih načina obavještavanja bez izmjena u osnovnoj logici slanja poruka.

4. Decorator

Decorator pattern omogućava dinamičko dodavanje novih funkcionalnosti objektima bez izmjene njihove strukture. U primjeru se koristi za obogaćivanje recenzija dodatnim oznakama, poput “verifikovana” ili “anonimna”. Ove oznake predstavljaju dodatne informacije koje se mogu fleksibilno dodavati, a da se osnovna klasa recenzije ne mora mijenjati.

5. Facade

Facade pattern nudi pojednostavljen interfejs ka kompleksnom podsistemu. U kontekstu kupovine ulaznica, klasa *KupovinaFacade* objedinjuje sve korake kupovine – provjeru podataka, plaćanje, izdavanje ulaznice i slanje notifikacije. Umjesto da klijent direktno komunicira s više različitih komponenti, koristi samo jednu facadu koja upravlja cijelim procesom.