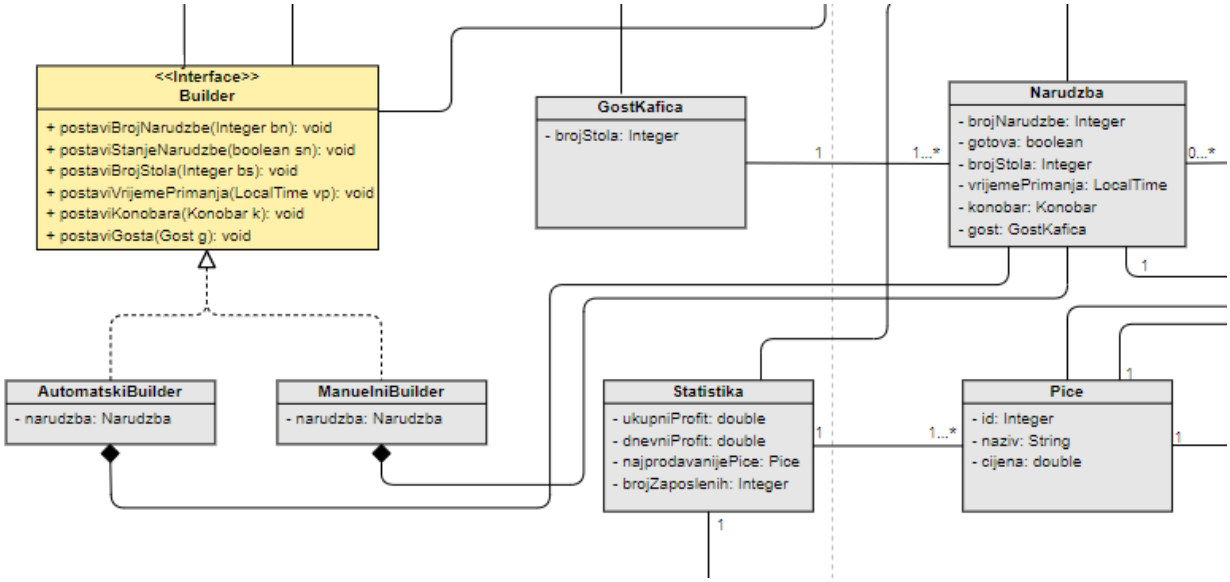
Kreacijski patterni

Kreacijski patterni koje smo odlučile implementirati u našem projektu su: builder i prototype.

Builder pattern – jedan od dseign patterna koji se često koristi u razvoju softvera radi olakšavanja kreiranja kompleksnih objekata. Ovaj pattern je vrlo koristan u implementaciji naše "Order" klase, s obzirom da se istoimena klasa sastoji od različitih atributa kao što su identifikator narudžbe (id), informacija o završetku narudžbe (done), broj stola (tableNumber) i vrijeme narudžbe (orderTime). Također, veza između narudžbe i gosta je ostvarena pomoću foreign key-a (idGuest) i referencirajućeg objekta klase "Guest". Za kreiranje objekata klase "Order" možemo iskoristiti Builder pattern. Korištenje Builder patterna pruža nekoliko prednosti za implementaciju "Order" klase u našem projektu. Prvo, olakšava konstrukciju objekta korak po korak, što je posebno korisno kada imamo kompleksne objekte s mnogo atributa. Builder interface se može povezati s kontrolerima u kojima se klasa "Order" koristi, kao što su "BartenderPanelController", "HomePageController" i "ModifyOrderController". Također, Builder pattern pruža fleksibilnost u postavljanju samo određenih atributa koji su potrebni za kreiranje objekta, ostavljajući ostale atribute sa defaultnim vrijednostima. Ovo je korisno u našem slučaju jer neki atributi, poput identifikatora gosta, mogu biti postavljeni naknadno.



Singleton pattern – pattern koji omogućava kreiranje samo jedne instance klase i osigurava globalan pristup toj instanci. U našem slučaju, gdje postoji samo jedan vlasnik kafića, Singleton pattern je prikladan za implementaciju klase "Owner". To osigurava da se samo jedna instanca vlasnika može kreirati i koristiti unutar cijele aplikacije. To je korisno kada postoji samo jedan vlasnik i kada je potrebno osigurati dosljednost i jedinstvenost podataka o vlasniku. Singleton nam omogućava jednostavno proširenje funkcionalnosti vlasnika kafića. Sve što trebamo učiniti je dodati metode ili svojstva u klasu "Owner", a te promjene bit će vidljive svim dijelovima aplikacije koji koriste Singleton za pristup vlasniku. Mada ovaj pattern se treba pažljivo koristiti, jer globalan pristup jednoj instanci može uticati na fleksibilnost i testiranje koda.

Factory pattern - dizajn pattern koji se koristi za kreiranje objekata bez eksplicitnog specificiranja njihovih klasa. Ovaj pattern pruža centraliziran način kreiranja objekata, što omogućava fleksibilnost i izbjegava neposrednu ovisnost o konkretnim klasama. U našem projektu, Factory pattern bi mogao biti koristan za kreiranje objekata između klasa "Guest" i "Bartender". Ovisno o različitim situacijama, svaka od ovih klasa može zahtijevati instancu objekta druge klase, a za to će nam služiti GuestFactory i BartenderFactory respektivno. GuestFactory bi mogla biti odgovorna za kreiranje objekata klase "Guest" na temelju različitih kriterija, kao što su tip gostiju (redovni gost, VIP gost, novi gost) ili osobne preferencije. Na taj način, umjesto da svaki kontroler ili klasa direktno instancira objekte klase "Guest", koristili biste GuestFactory za dobivanje željenih instanci. Analogno, vrijedi i za BartenderFactory koji bi nam omogućio kreiranje instanci bartendera različitih tipova kao što su: barmeni za koktele, barmeni za cijeđene sokove ili barmene za pića s alkoholom.