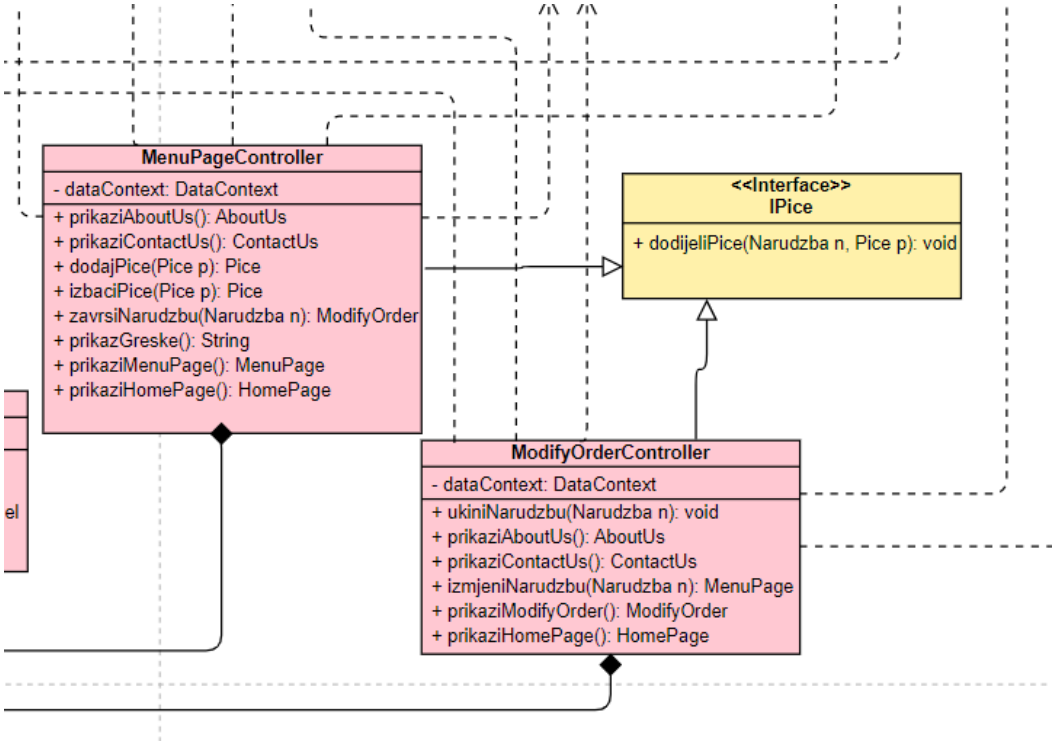
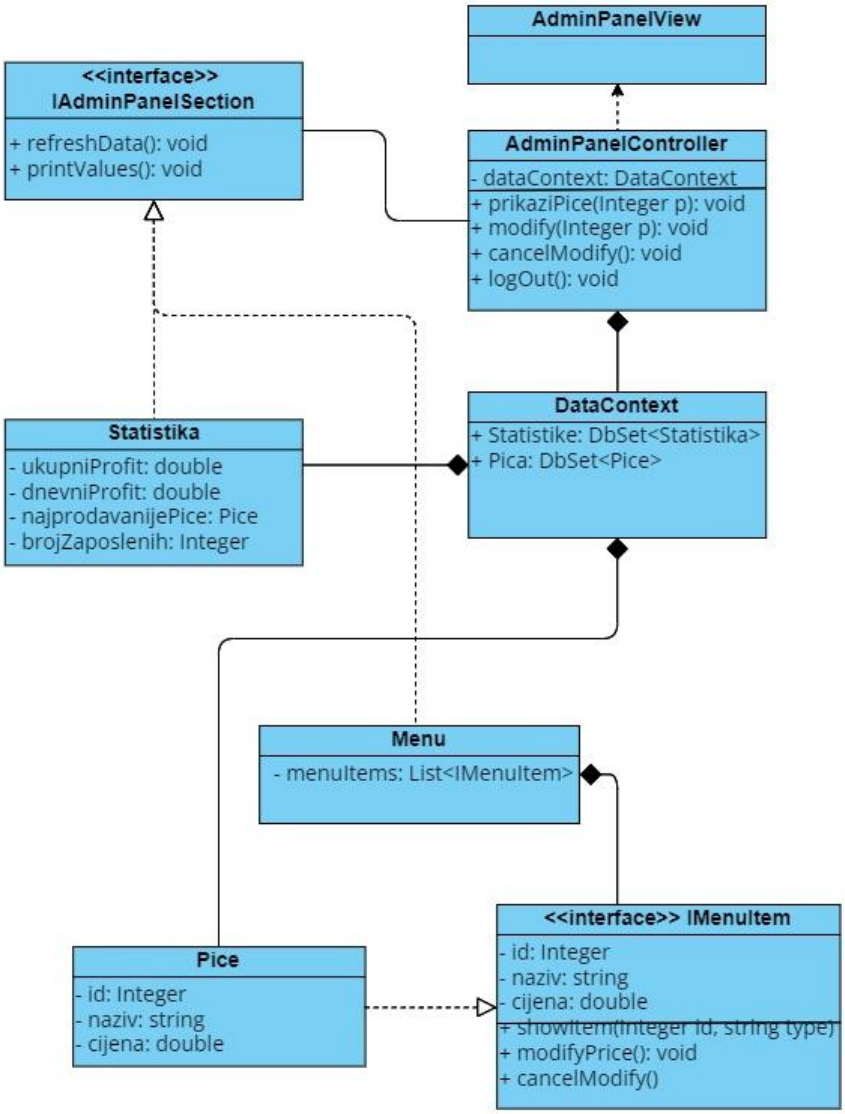
Strukturalni design patterni

Odabrani strukturalni design patterni za implementaciju su: composite i flyweight.

Flyweight design pattern – odabrali smo ovaj pattern, jer u slučaju kada imamo meni s mnogo različitih pića, Flyweight pattern može nam pomoći optimizirati memoriju koja se koristi za pohranu informacija o tim pićima. Dijeljenjem zajedničkih podataka možemo smanjiti potrebnu memoriju za pohranu istih informacija za svaku instancu. Možemo uzeti za primjer koktele koji se nalaze na našem meniju. U bazi ćemo imati samo po jednu instancu koktela koju ćemo dalje koristiti za kreiranje narudžbi. Time ćemo ostvariti značajnu uštedu memorije i izbjeći kreiranje novih instanci koktela.



Composite design pattern – kao design pattern koji omogućuje grupiranje objekata u hijerarhijske strukture i tretira istoimene kao pojedinačni objekat, odlično se može uklopiti u imlementaciji Admin Panela naše web aplikacije. Hijerarhija će se sastojati od dvije glavne komponente: Statistika i Meni. Uz definisanje Leaf klasa (komponenti) olakšat ćemo dodavanje novih funkcionalnosti jer se nove komponente Admin Panela mogu dodavati bez izmjena prethodnog koda. Ovakva prednost patterna nam daje mogućnost, prilikom nadogradnje sistema i prilagođavanja potreba klijenta, da se i Meni može proširiti sa drugim Leaf klasama (trenutno ima samo Pice) kao što su jela, topla pića i sl. Labele koje će se koristiti za prikaz podataka unutar Statistike će sada biti grupisane i time smanjiti složenost koda. Sa Component klasom možemo definirati metode koje će biti zajedničke za Leaf i Composite objekte formirajući time stablastu strukturu.



Adapter design pattern – Integracija vanjskog QR skenera u aplikaciju. Prilikom skeniranja QR koda za identifikaciju stola, možemo koristiti Adapter da bismo prilagodili različite modele i protokole različitih QR skenera koje bi korisnici koristili. Adapter bi se pobrinuo za pretvaranje izlaza skenera u jedinstven oblik kojeg naša aplikacija može koristiti za identifikaciju broja stola. Osim ovog pristupa, primjena Adapter patterna se može pronaći u integraciji sistema za naplatu sa aplikacijom. Prilikom razvoja sistema bilo bi dobro dati mogućnost korisniku da plati svoju narudžbu putem aplikacije i svog bankovnog računa. Time dajemo aplikaciji dodatnu komunikaciju između ta dva sistema tako da prilagodi i prevodi zahtjeve i odgovore između njih.

Decorator design pattern – pattern koji omogućava dodavanje novih funkcionalnosti objektima dinamički, bez mjenjanja njihove osnovne strukture ili postojećeg koda. Jedno od područja u kojem bi se mogao primijeniti Decorator design pattern je za proširenje funkcionalnosti narudžbe pića. Osnovni objekat može predstavljati narudžbu pića, ali uz ovaj pattern dobijamo mogućnost dodavanja opcija kao što su unos promotivnog koda za smanjenje cijene pića, odabira veličine pića ili dodatnih sastojaka. Svaki dodatak (Decorator) ima svoju cijenu koja se dodaje osnovnoj cijeni narudžbe. Uz ovaj pattern, imamo mogućnost građi različitih kombinacija dekoratora da prilagodimo narudžbu pića prema specifičnim zahtjevima korisnika.

Proxy design pattern - koristili bi proxy objekt za autentifikaciju i autorizaciju zaposlenika za obavljanje upravljačkih zadataka poput upravljanja cijenama, pregled financijskih izvještaja, prihvatanje ili odbijanje narudžbe te uvid u statistiku. Ovim ćemo postići da će proxy objekt omogućiti različitim korisnicima različit nivo pristupa i funkcionalnosti, štiteći sigurnost aplikacije. Proxy će posredovati između klijenta (korisnika) i stvarne implementacije objekta, pružajući dodatnu kontrolu pristupa i osiguravajući da se samo ovlašteni korisnici mogu baviti određenim operacijama.

Façade design pattern - facade pattern olakšava klijentima (gostima, konobarima, vlasnicima) da koriste funkcionalnosti aplikacije bez potrebe za poznavanjem unutrašnjeg složenog sistema. Ako bi naša aplikacija ima složene backend sisteme, kao što su upravljanje inventarom, obrada narudžbi, dostava i druge operacije, Facade pattern može ograničiti izlaganje tih kompleksnosti korisnicima. Kroz Facade, korisnici mogu pristupiti samo pojednostavljenim metodama koje su im potrebne, a kompleksne interne operacije se rješavaju unutar fasade. Također, pruža strukturu koja pomaže organizirati kod i povezane funkcionalnosti. Fasada djeluje kao jedinstvena tačka ulaza za korisnike, što olakšava razumijevanje i održavanje aplikacije. Ovaj pattern nije odabran za implementaciju jer ipak naš backend sistem nije toliko složen da bi potreba za njim bila opravdana, međutim u daljnjim nadogradnjama svakako da bi koristio i da bi se mogao implementirati.

Bridge design pattern – kada bi u našoj aplikaciji dodali funkcionalnosti da postoji loyality program za odabrane korisnike, tu bi nam koristio i dosta pomogao bridge pattern. Korištenjem bridge patterna nam omogućuje fleksibilnost u upravljanju loyality programa kafića. Tada bi smo mogli mijenjati ili zamijeniti loyality program bez uticaja na ostale funkcionalnosti aplikacije. Jedan od primjera je da želimo ažurirati loyality program ili dodati nove vrste nagrada bez mijenjanja korisničkih klasa i načina na koji korisnici pristupaju programu sa posebnim beneficijama.