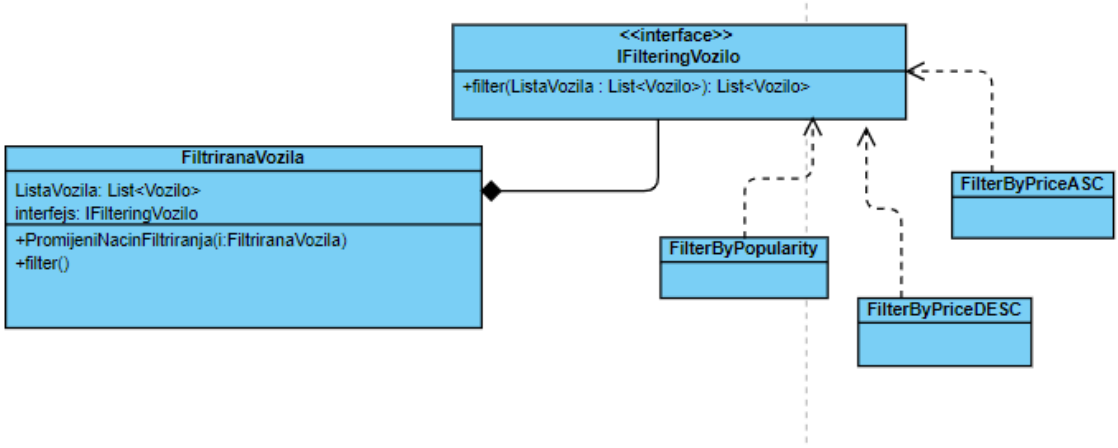
Paterni ponašanja

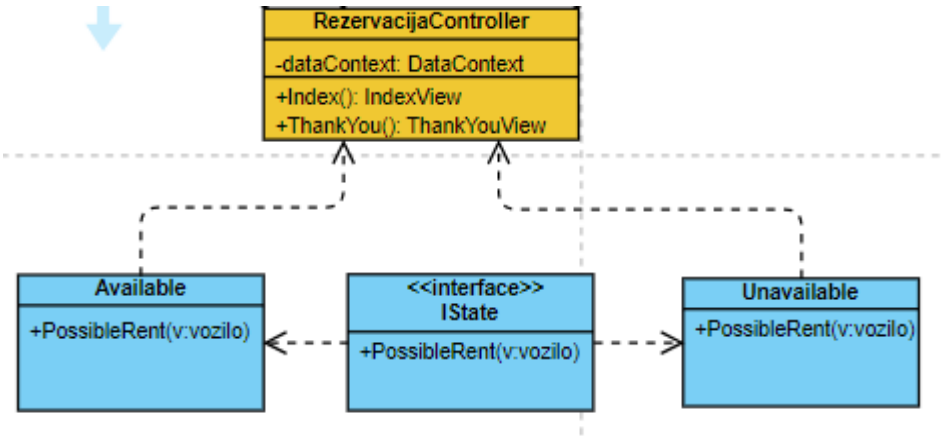
1. Strategy pattern

Strategy pattern izdvaja algoritam iz matične klase i uključuje ga u posebne klase.
Pogodan je za primjenu kada imamo više načina / algoritama koji se mogu iskoristiti u nekom slučaju. Mi smo iskoristili ovaj pattern kod funkcionalnosti sortiranja vozila. S obzirom da korisnici mogu izabrati cijenu i najzastupljenije produkte, uvodimo i pretraživanje po tim stavkama gdje će korisnici pored defaultnog moda moći promijeniti listu datih vozila.



2. State pattern

State patteren mijenja način ponašanja objekta na osnovu trenutnog stanja.
S obzirom da jedno vozilo može biti iznajmljeno samo jednom korisniku u određenom vremenskom periodu, dodali smo dodatni status vozilo koji nam daje informaciju o tome da li je vozilo „dostupno“ ili je vozilo „iznajmljeno“ u tom vremenskom periodu.



3. Template method pattern

Template pattern služi za omogućavanje izmjene ponašanja u jednom ili više dijelova.
Moguća primjena ovog patterna je npr. kod funkcionalnosti registracije. Nakon prve prijave/ registrovanja i kreiranja korisničkog accounta, korisniku možemo poslati e-mail dobrodošlice ili pravila korištenja sistema ili mu prikazati poruku.

4. Observer pattern

Obeserver pattern ima ulogu da poveznicu objekata tako da kada jedan objekat promijeni stanje, drugi zainteresovani objekti se obavještavaju. Ovaj pattern možemo iskoristiti za obavijesti o raspoloživosti vozila. Ako dodamo neko novo vozilo ili vozilo završi svoj period rentanja možemo obavijestiti registrovanog korisnika da je vozilo dodato ili dostupno.
Također možemo ga iskoristiti kod pravljenja rezervacije i odabira dostave. Korisnika možemo obavijestit rezervacija uspjela, poslati mu osnovne rezervacije i vremenski period dostave vozila.
Na taj način bi znatno doprinijeli marketingu firme.

5. Iterator pattern

Iterator pattern služi za efikasno i jednostavno pretraživanje kolekcije podataka, bez da otkrivamo unutarnju strukturu kolekcije. Ako korisnik ima potrebu pretraživati proizvode na osnovu određenih kriterija, ovaj pattern može biti koristan za iteriranje kroz rezultate pretrage. Možemo napraviti iteratorCijene, iteratorAbecedni, iteratorModela koji bi koristili određeni iterator u zavisnosti od odabira filtera pretrage.

6. Chain of responsibility pattern

Chain of responsibility pattern namijenjen je da bi se jedan kompleksni proces obrade razdvojio na način da više objekata na različite načine procesiraju podatke. U našem sistemu smo ga mogli iskoristiti u glavnoj funkcionalnosti pretrage,rentanja i naplaćivanja vozila. Korisnik bi prvo trebao da izabere datum rezervacije, zatim na osnovu slobodnih automobila da izabere vozilo, a na kraju i da odabere način plaćanja. Ukoliko želi može i otkazati rezervaciju.

7. Mediator

Mediator pattern služi kao međuobjekat da bi se izbjeglo direktno povezivanje velikog broja objekata. On je zadužen da komunikaciju između objekata i prosljeđivanje poruka od jednog objekta ka drugom. Ukoliko bi došlo do neke greške u sistemu prilikom rezervacije, plaćanja, refundiranja novca mogli bismo napraviti neku poveznicu i komunikaciju Registrovanog korisnika i administratora gdje možemo provjeriti da li je prenesena poruka automatska (bot) poruka ili neki govor mržnje ili pravi zahtjev za pomoć.