Refactoring koda (GoFly)

Obzirom da mi nismo uradili backend projekta, tj. nismo završili implementaciju to ćemo ovdje samo spomenuti prijedloge gdje bi u našem kodu mogli koristiti design pattern-e za refactoring koda.

Proxy pattern
Možemo koristiti kod prijave na našu aplikaciju. Nije koristen u projektu, ali mozemo proxy design pattern koristiti pri prijavi na nasu aplikaciju. Mogli bi da kreiramo objekat koji ce primati zahtjeve za pristup aplikaciji, zajedno sa podacima o mailu i sifri korisnika, a zatim u zavisnosti od proslijedenih podataka otvaramo neku formu aplikacije, ili javljamo gresku.

Decorator pattern
U nasem projektu mozemo koristiti ovaj patern prilikom dodavanja novih ili brisanja starih letova, gdje se lista svih letova dinamicki prikazuje. Takoder mozemo koristiti ovaj patern prilikom rezervacije ili ponistenja rezervacije karata, gdje treba da dinamicki prikazujemo broj slobodnih mjesta u odredenom letu.

Iterator pattern
Pogodno mjesto za implementaciju ovog patterna bi svakako bila npr. klasa PonudaLetova u kojoj čuvamo listu letova. U ovoj klasi bi definitivno trebalo iskoristiti ovaj pattern i tako omogućiti pristup svakom letu pojedinačno bez da otkrivamo internu strukturu naše klase Let.

Strategy pattern
Naime u folderu Helper gdje smo sadržavali nekoliko pomoćnih klasa, imali smo i klasu Validation. U njoj kreiramo MD5 hash za šifre pomoću funkcije CreateMD5. Ukoliko bismo iskoristili Strategy pattern u našem projektu, mogli bismo podržati puno više načina za hash-iranje šifri kao i dati podršku promjeni hash funkcije ukoliko bi za to bilo potrebe u budućnosti.

Observe pattern
Moguća implementacija i upotreba ovog patterna bi se ogledala kroz sistem za notifikacije, tj. obavještavanje korisnika o dostupnim akcijama/popustima za karte određenih letova, odgođene letove i sl.