# PATERNI PONAŠANJA

## 1. Strategy pattern

Strategy pattern koristimo onda kada želimo izdvojiti određene algoritme iz njihovih matičnih klasa u posebne klase. Pogodan je za korištenje kada postoje različiti algoritmi koji se mogu primijeniti na isti problem. On omogućava klijentu izbor jednog od algoritama iz familije dostupnih algoritama, dok su implementacije istih neovisne od klijenata koji ih koriste.

U našem slučaju, Strategy pattern bi se mogao iskoristiti kod implementacije pretrage korisnika kojima je potrebno poslati relevantno obavještenje putem dostupnih notification servisa. Ukoliko je to potrebno, moguće je vlasnike ljubimaca filtrirati po adresi stanovanja (geografski lokalizirana selekcija), rasi ljubimca (rasna selekcija), datumu zadnje vakcinacije (selekcija na osnovu recentnosti vakcinisanja) i slično.

## 2. State pattern

State pattern podrazumijeva promjenjiv način ponašanja objekta na osnovu njegovog trenutnog stanja. Pogodan je za korištenje ukoliko objekat može imati više stanja, te ukoliko često dolazi do promjene istih.

U našem sistemu moguće je ovaj pattern iskoristiti kod implementacije klase Recept, te manipulacije s objektima tog tipa. Ljekarski recept može biti u stanju popunjavanja, izdat i poništen. Prema tome, stanja koja bi objekat ove klase mogao imati bi bila „pending“, „processed“ i „discarded“ ili tome slično. U ovisnosti od toga u kojem je stanju, recept može čekati na slanje, biti poslan (kada se može i ručno povući u slučaju greške, ili isprintati u fizičkom obliku), i poništen (kada vlasnik preuzme lijekove od apotekara).

## 3. Template Method pattern

Template method pattern omogućuje izdvajanje određenih koraka nekog algoritma u odvojene podklase. Ova praksa ne utiče na strukturu algoritma, već samo omogućuje implementaciju pojedinih njegovih dijelova na različite načine.

Ovaj pattern bi se u našem sistemu mogao koristiti kod implementacije Pregleda, gdje bi se za različite tipove pregleda tražili različiti podaci. Recimo, ukoliko je u pitanju obični pregled bez intervencije, mogli bi se tražiti samo osnovni podaci o ljubimcu, razlogu posjete, te napomeni ljekara. Ukoliko je pregled tipa operativnog zahvata, vakcinacije ili sličnog tretmana, trebaju se zatražiti podaci o tipu operativnog zahvata, vrsti vakcine koju je ljubimac primio i sl.

## 4. Observer pattern

Ovaj pattern karakteriše uspostava relacije između objekata tako da kada jedan objekat promijeni stanje, drugi povezani objekti se o tome obavještavaju.

Konkretno u našem sistemu je implementiran sistem slanja obavijesti koji funkcionira na način da ciljane grupe korisnika (vlasnika ljubimaca) dobijaju obavijesti o aktuelnim događajima, kako u samom sistemu, tako i aktuelnih veterinarskih događaja koji su relevantni datom korisniku. Recimo, tehničke informacije o dostupnosti servise bi se mogle slati e-mailom, kao i podaci o organiziranoj planskoj vakcinaciji ljubimaca na određenoj lokaciji ili dostupnosti / nedostupnosti lijekova koji su na aktivnom receptu za ljubimca nekog vlasnika.

## 5. Iterator pattern

Princip iterator patterna se ogleda u omogućivanju pristupa elementima bez poznavanja kako je sama kolekcija istih struktuirana. Osnovna ideja iterator patterna je ekstrakcija navigacije kroz kolekciju u odvojene objekte nazvane iteratorima. Iteratori implementiraju različite algoritme navigacije kroz zajednički interface bez znanja o njihovoj temeljnoj implementaciji.

U našem sistemu se iterator pattern može koristiti prilikom pretrage korisnika kojima je potrebno poslati obavijest. Ukoliko se šalje obavijest za geografski lokaliziranu grupu korisnika potrebno je izdvojiti sve korisnike koji su tipa vlasnik, a zatim iteriranjem kroz dobijenu kolekciju izdvojiti one sa relevantnom geografskom lokacijom (recimo kod planske vakcinacije).

## 6. Command pattern

Command pattern se koristi za enkapsulaciju svih informacija potrebnih za odgođeno izvođenje akcije ili pokretanje događaja. Informacije uključuju naziv metode, objekat koji tu metodu posjeduje, kao i vrijednosti parametara metode.

U našem sistemu postoji više različitih načina da se podaci o vlasniku ili njegovom ljubimcu prikažu u sistemu. U ovisnosti od toga koji je način upotrebljen (skeniranje QR koda s kartice, RFID čipa ili unos ID-a ručnim putem), moguće je ovaj pattern iskoristiti za kreiranje zahtjeva za prikaz podataka, dok se informacije potrebne za prikaz korisnika dobavljaju ovisno o odabranom načinu unosa.