# 4. Predlog strategije za keširanje podataka

Keširanje podataka je još jedan koncept koji može doprineti skalabilnosti naše aplikacije. Ovo nam dodatno pomaže da smanjimo opterećenje nad samom bazom tako što ćemo imati učitan podskup podataka. Keširanje se može implementirati na više različitih nivoa, od kojih su dva najčešća L1 (Level 1) i L2 (Level 2). Hibernate podržava i jedan, i drugi nivo keširanja, s tim što se uz L2 mora koristiti i eksterni provajder, kao što su: EhCache, Infinispan, Redis...

L1 keširanje predstavlja keširanje na nivou sesije, tj. ono nam omogućava izbegavanje konfliktnih podataka i učitavanja duplikata na taj način što će zahtev za objektom iz baze uvek vratiti istu instancu. Takođe, memorija se ne opterećuje, jer životni vek keširanog objekta traje koliko i Hibernate sesija.

L2 keširanje operiše na nivou „fabrika sesije“, i za njega postoje strategije adaptirane za različite situacije rada sa aplikacijom: Read Only, Read Write, Nonrestricted Read Write i Transactional. Read Only služi za podatke koji se samo čitaju i nikada se ne ažuriraju, pa je samim tim i najjednostavnija strategija sa optimalnim performansama. Read Write, pored čitanja, može vršiti i ažuriranje podataka, a Nonrestricted Read Write funkcioniše po istom principu, s tim što se podaci ažuriraju veoma retko. Transactional strategija se koristi kod konkuretnih sistema, gde je od velike važnosti konzistentnost podataka. Konkretno u našem sistemu, Read Only bi se moglo primeti nad raznim prikazima podataka, gde nisu moguće CRUD operacije. Nad većinom ostalih operacija i podataka bi bilo najbolje koristiti Read Write strategiju, kako bi se izbegli problemi do kojih može dovesti Nonrestricted Read Write. Takođe, Transactional se može upotrebiti za onemogućavanje konfliktnih situacija operisanja nad podacima, što je i implementirano u pojedinim delovima naše aplikacije.

Kako bi keš memorija bila u toku sa izmenama u sistemu, a da pri tome ne utiče na opterećenje baze, poželjno bi bilo podešavati njen životni vek (TTL - Time To Live). Takođe, ukoliko dođe do popunjenja memorije, poželjno bi bilo koristiti Evicted, koji po principu LRU (Least Recently Used) izbacuje najmanje nedavno korišćeni objekat.