Skalabilnost

Povećanje istovremenih korisnika aplikacije znači i veći broj zahteva na koje aplikacija mora odgovoriti. Skalabine aplikacije omogućavaju obradu svih tih zahteva jednako brzo kao i ranije.

Kako poboljšati skalabilnost?

Neophodno je povećati resurse (hardver: brže procesore sa više jezgara, više RAM memorije), ali povećanje resursa povećava i potrebna ulaganja i često nije dodvoljno ukoliko želimo da naša aplikacija opslužuje više hiljada korisnika.

Rešenje koje bi bilo prikaldnije je postojanje više servera koji međusobno sarađuju i distribuiraju zahteve. Komponenta koja ih povezuje je Load Balancer.

Load Balancer redirektuje zahteve ka serveru koji je sposoban (slobodan) da ih obradi. Cilj je da nijedan server ne bude preopterećen, već da svi budu podjednako uposleni. Ukoliko bi došlo do otkaza jednog servera Load Balancer bi automatski preusmerio saobraćaj na drugi server. U slučaju da se se jedan server ugasi, load balanser usmerava saobraćaj ka preostalim aktuelnim serverima, a isto tako kada se novi server doda u grupu, load balanser automatski kreće da redirektuje zahteve ka njemu.

U našoj aplikaciji, korisnik u toku rezervacije leta bira let, zauzima sedišta, izvršava rezervaciju hotela i vozila. U svakoj od ovih akcija korisnik ostavlja podatke koje su neophodne sledećoj. U našem slučaju Load Balancer ne bi morao da vodi računa o ‘session perssistance’ tj da se svi zahtevi sesije ispunjavaju na jednom serveru iz razloga što smo koristili SpringBootSecurity i tokene. Prilikom svakog obraćanja serveru korisnik šalje token sa svojim username-om i password-om.

Sledeća stvar koju bi preduzeli je opttimizacija rada sa bazom. Ne bismo mogli da koristimo MySql zato što se ne skalira dobro sa Spring aplikacijama već bi morali da pređemo na neke druge baze podataka. Ptrebno je poboljšati kvalitet SQL upita tako da maksimalno rasterećuju našu aplikaciju i posao prebacuju u samu bazu. Fokus bi stavili na što je manje moguće obraćanja bazi tako što bi pisali sveobuhvatnije

upite (idealno za pretragu). Ovim bi povećali performanse naše aplikacije budući da je obraćenje bazi najskuplje. Neophodno je i napraviti nekoliko kopija baze. Svesni smo da to usporava celokupan rad aplikacije ali je neophodno zbog reduntantnosti i konzistentnosti budući da se radi o korisničkom novcu i poverenju.

Omogučili bismo i straničenje (paging). Sa povećanjem korisnika aplikacija povećao bi se i broj objekata tako da bi gubili mnogo na performansama ako bi korinsniku davali odjednom sve informacije. Na primer prilikom pretrage učitalo bi se prvih deset rezultata pretrage i prikazalo korisniku, narednih deset bi se učitalo tek kada to korisnik zahteva.

Koristili smo optimistično zaključavanje baze da bi obezbedili što bolju konkurentnost i kako bi naša aplikacija bila bezbedna i pouzana.

Aplikacije koje bismo koristili da nam pomognu u skaliranju su:

- Docker: pomaže da lako pokrenemo aplikaciju i njene kopije ako su potrebne
- Ngnix: koristi se za Load Balancing