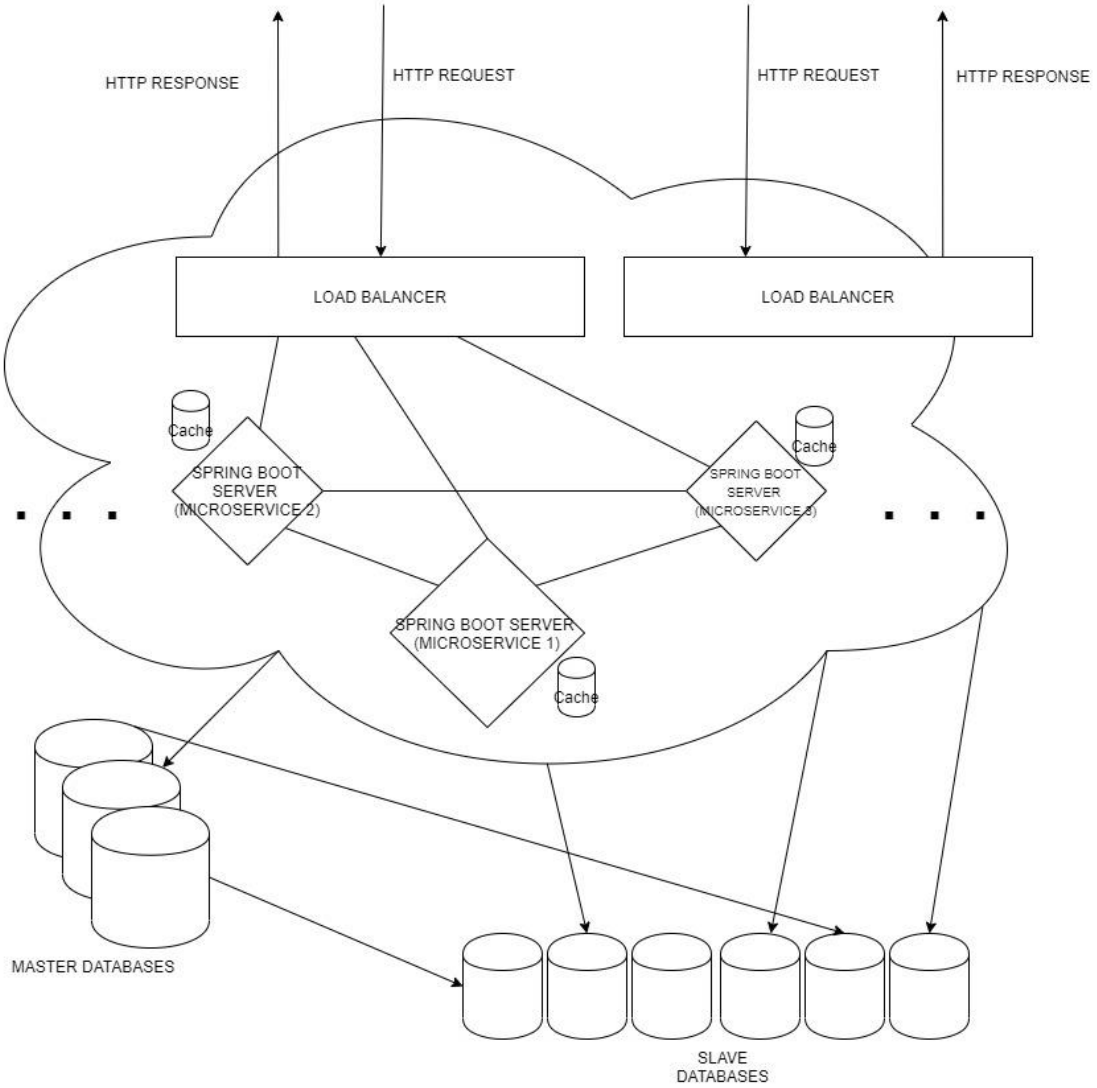
Proof Of Concept za skaliranje veb aplikacije “Klinički Sistem”



Prvi korak u skaliranju aplikacije bio bi podela na funkcionalnosti koje bi bile implementirane u okviru zasebnih spring boot aplikacija, i pokrenute na odvojenim serverima kako bi se postigla mikroserverska arhitektura. S obzirom da potrebe aplikacije prevazilaze mogućnosti jednog servera, svaki mikroservis, a pogotovo oni krucijalni za funkcionisanje klinika, bili bi podignuti na više različitih servera. Problem koji nastaje implementiranjem mikroserverske arhitekture jeste redirekcija dolaznih HTTP zahteva na odgovarajuće servere u zavisnosti od funkcionalnosti koje se traže, ali i od preopterećenosti pojedinih servera. Ovaj posao obavlja *load balancer.* Programi poput NGINX-a koriste se za implementaciju load balancera, kontrolišu saobraćaj na ulaznom port-u i preusmeravaju ga dalje. Pošto u ovom scenariju load balancer postaje *single point of failure,* oni se često implementiraju u parovima.

Što se tiče baze podataka, veličina aplikacije prevazilazi mogućnosti bilo koje baze, tako da je potrebno izvršiti particiju podataka. Prva ćemo podeliti bazu na *MASTER* i *SLAVE.* Master baze će se koristiti za upis podataka, a slave za čitanje. Konzistentnost između ovih baza će predstavljati problem, ali ćemo se osloniti na koncept "*eventual consistency*". Periodično će se vršiti prepisivanje podataka iz master u slave baze, pogotovo kod podataka koji nisu od krucijalnog značaja, poput ocena klinika i doktora. Ovom podelom smo rešili problem sporih baza koje su preopterećene zahtevima, ali ne i prevelike količine podataka. To ćemo učiniti pomoću Hashed Sharding postupka, gde se podaci na osnovu nekog kriterujuma čuvaju u različite baze, a u hash tabeli pamtimo koji torke su na kojim fizičkim lokacijama. Pored ovoga, svaki server će imati i svoju keš memoriju, kako bi se upiti ka bazama sveli na minimum. U ovoj memoriji ćemo čuvati podatke o korisnicima koji su u bliskoj prošlosti koristili usluge datog servisa, i šansa je da su i dalje online, i da će uskoro ponovo tražiti neke podatke. Cache-iranje podataka se može implementirati pomoću softvera poput *Memcached*-a Takođe treba implementirati i database connection pooling. Otvorene konekcije ka bazi će biti dostupne korisnicima, čak i kada ih niko ne koristi, jer se time eliminiše potreba za konstantnim otvaranjem i zatvaranjem novih konekcija. Broj konekcija će se menjati na osnovu doba da,a ili dana u nedelji, tačnije očekivanog saobraćaja na veb aplikaciji.