1. Dizajn šeme baze podataka
2. Predlog strategije za particionisanje podataka

   Particionisanje podataka može pomoći pri smanjenju broja zahteva za pristup resursima.Tabele koje imaju veliki broj podataka možemo podeliti horizontalnim particionisanjem, gde bismo podatke podelili po nekom ključu. Ključ po kom particionišemo podatke na primer u e-Receptima ili rezervaciji može biti ključ određenog broja pacijenta, i tako bi postojalo više grupa e-Recepata u okviru jedne tabele, i omogućio bi se brži pristup podacima. Slično možemo rešiti i problem sa tabelama za savetovanja, i preglede, gde bi ključ bio ključ dermatologa ili farmaceuta. Vertikalno particionisanje možemo primeniti na delove tabela za koje korisnik ređe traži upite. Samim vertikalnim particionisanjem odvajaju se bitnije i manje bitne kolone, što omogućava brži pristup bitnim kolonama.

3. Predlog startegije za replikaciju baze i obezbjeđivanje otpornosti na greške
   Pošto se sistem većinski koristi za rezervacije, očekuje se veliki broj write naredbi, tako da klasterujemo relacione baze podataka u multiple master arhitekturi. Svaki database server obrađuje update i insert upite i iste prosleđuje ostalim database serverima kako bi replikacije bile u konzistentnom stanju. Potrebno je da imamo više od dva database servera da usled otkaza mreže ne bi došlo do skladištenja različitih podataka, kako je svaki master. Jedini događaj koji bi doveo replikacije u nekonzistetno stanje u arhitekturi na slici je otkaz database servera i mreže između druga dva dabase servera, što procenjujemo kao veoma malu verovatnoću. U slučaju da se poveća broj read upita možemo dodati dodatni slave database server sa koga se samo čitaju podaci. Transackije i optimističko zaključavanje
4. Predlog strategije za keširanje podataka

   Keširanje podataka može ubrzati pristup podacima koji se ne menjaju često, kako na primer lični podaci o korisniku, podaci o apoteci, podaci o leku itd. Upiti za podatke koji se često menjaju, kao na primer količina leka na stanju, rezervacije, termini i slični podaci, možemo podržati least recently used cache strategiju gde ćemo keširati rezultate za poslednje upite i to preko EhCache je open-source standardno-baziran keš koji povećava performance, koji rasterećuje bazu i pojednostavljuje skalabilnost. Najčešće se implementira samo dodavanjem anotacija.

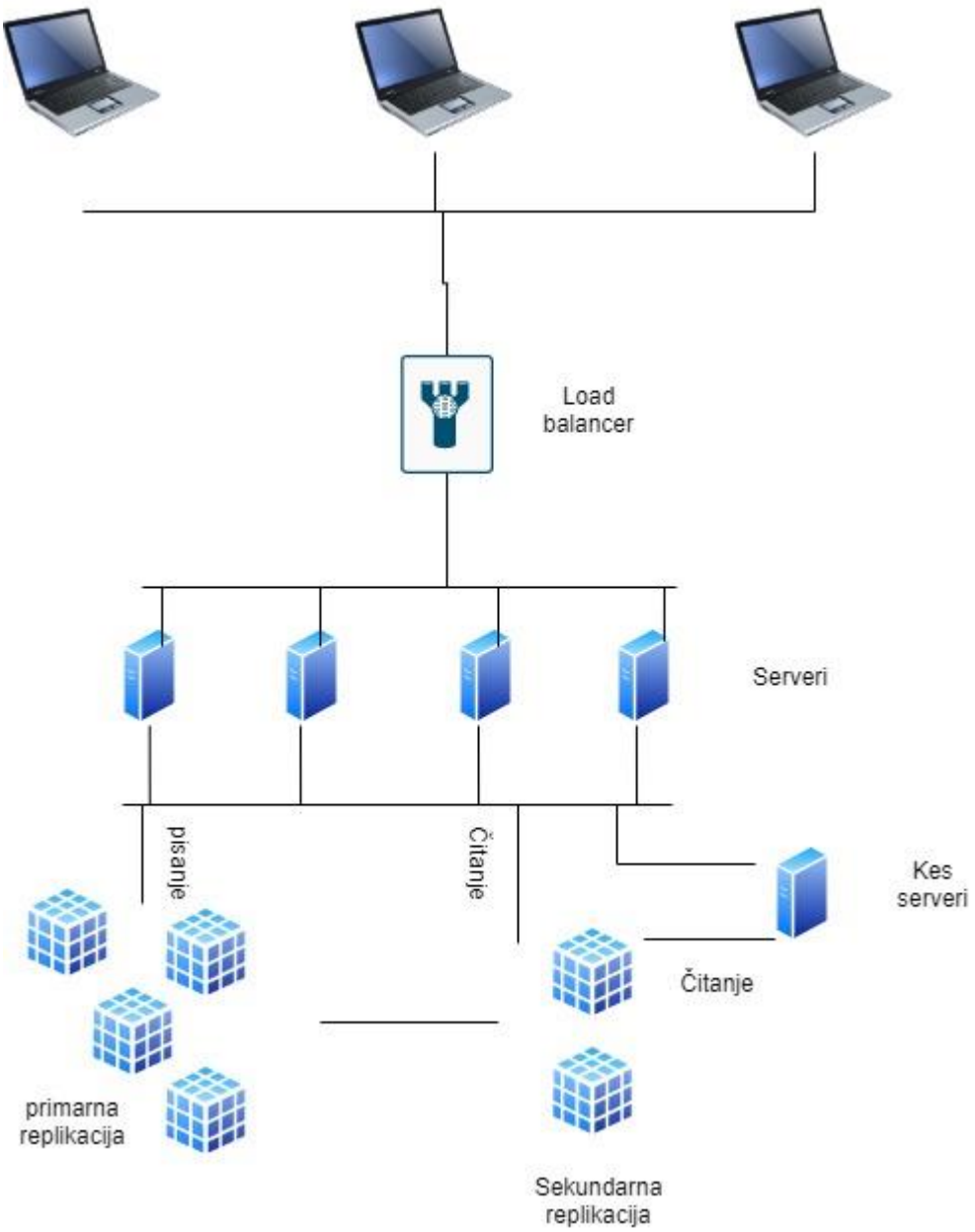
5. Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina
   Uočili smo entitete koji se registruju samo jednom u okviru sistema i čiji broj nije dovoljno velik tako da utiče na potrebno skladište u dužem periodu na ovako veliki broj korisnika. Entiteti uzeti u obzir su nalozi korisnika, pregledi, izveštaj sa pregleda, rezervacije lekova, i ocene i žalbe. Sa ovim uzetim u obzir ako je ukupan broj korisnika 100 miliona i perzistencija korisnika u bazi zauzima 0.5KB po korisniku što čini približno 47.8GB. Od procenjenih 100000 pregleda i rezervacija leka, uzećemo da na svaku desetu rezervaciju leka će biti jedan pregled. Za perzistenciju jedne rezervacije leka u proseku je potrebno 0.15kb što u narednih pet godina čini 7.72gb. Za perzistenciju jednog pregleda je u proseku potrebno 0.2kb. Ukoliko na mesečnom

nivou imamo 100.000 pregleda, nakon pet godina za preglede će biti potrebno 1.14gb. Procenjujemo da će 5% korisnika otkazati pregled ili se neće pojaviti na istom. Procenjuje se da će mesečno biti 95.000 izveštaja što nakon pet godina čini približno 8.15gb. U proseku ocena ili komentar zauzimaju 0.4kb što u slučaju da 10% korisnika ostavi ukpno će zauzeti 250mb. Perzistencija svih ostalih entiteta će biti sigurno manja od ocena i komentara, računaćemo da će u proseku uzimati 30mb što za ostalih 40 entiteta čini 1.2gb. Tako da ukupno procenjujemo da će u narednih pet godina biti potrebno 114gb za skladištenje svih podataka.

6. Predlog startegije za postavljanje load balansera

   Radi smanjenja preopterećenja zadataka, koristili bismo LoadBalanser . Pod pretpostavkom da nemamo servere podjednako snažnih konfiguracija, iskoristili bismo weighted round robin algoritam. Zahtevi bi se raspoređivali ciklično po čvorovima, uzimajući u obzir I njihove težine. Jače konfiguracije bi imale veću težinu i primale bi vise zahteva. U nšem slučaju aplikacija je stateless.

7. Predlog koje operacije korisnik treba nadgledati u cijlju poboljšanja sistema
   Implementacijom event sourcing mehanizma imaćemo sve potrebne podatke za implementaciju bilo koje statistike koju će poslovanje tražiti od nas. Što tehničkog dela rešenja, njegove upotrebljivosti tako i ponašanja korisnika i njihovih interesovanja. Na ovaj način možemo pratiti potražnju za lekove i na taj način obezbediti da apoteke imaju uvek tačnu informaciju za nove nabavke. Kao i broj poseta farmaceutu ili dermatologu na koji način se može pratiti kvalitet njihove usluge.
8. Kompetan crtež dizajna predložene arhitekture (aplikativni serveri, serveri baza, serveri za keširanje)


Load
balancer
Serveri
pisanje
Čitanje
Kes
serveri
Čitanje
primarna
replikacija
Sekundarna
replikacija