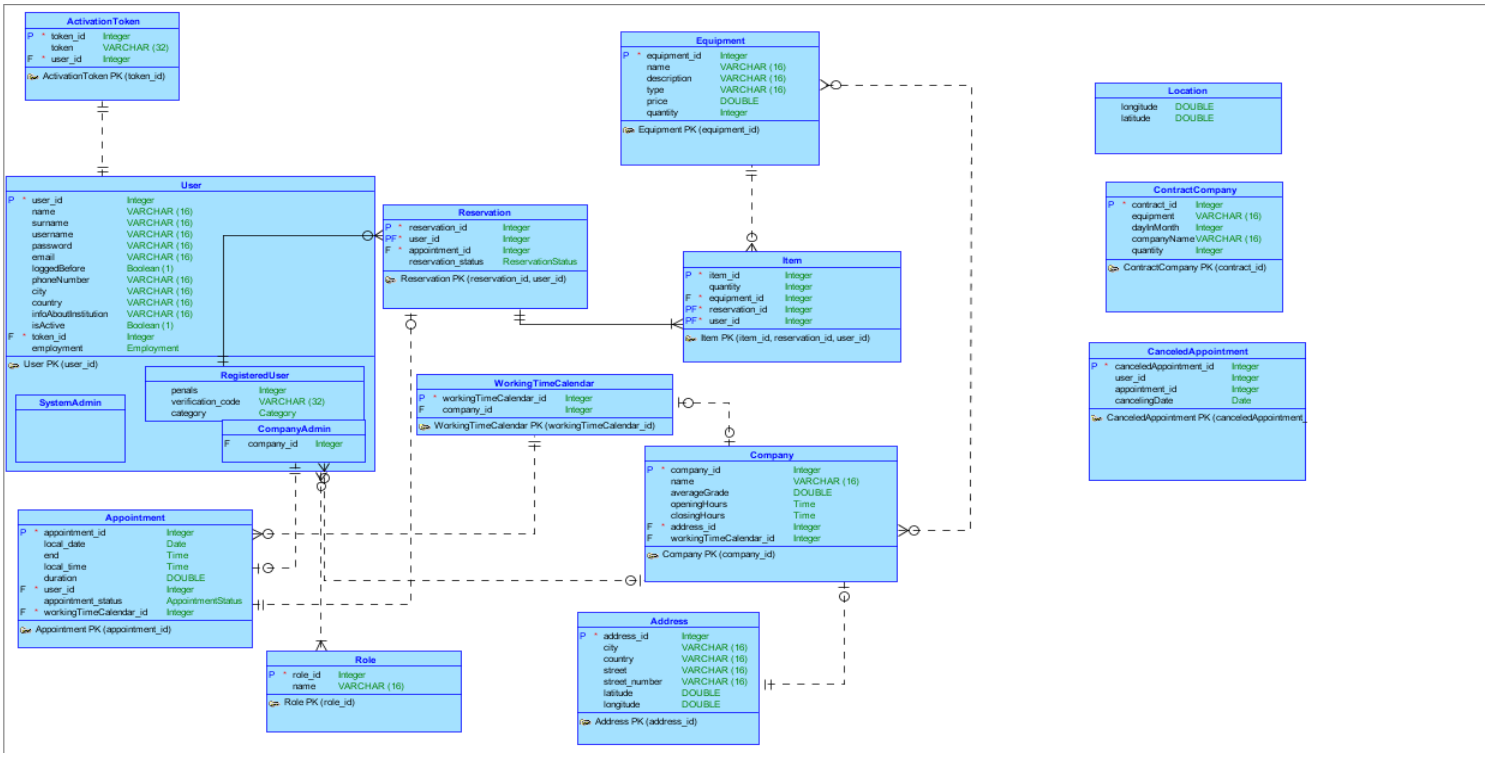
PROOF OF CONCEPT

1. Dizajn šeme baze podataka (konceptualni, logički i fizički)



2. Predlog strategije za particionisanje podataka

Jedan od načina da pristup podacima olakšamo i ubrzamo jeste da razmotrimo da li postoje tabele čiji se podaci ne koriste i ne menjaju istom učestalošću. Kao primer vidimo da se tabeli *user* lični podaci koriste i menjaju dosta ređe od podataka poput korisničkog imena, lozinke i penala. Takve tabele možemo podeliti tako da u jednoj zadržimo podatke koji se koriste često, a u drugoj dodatne, ređe korišćene.

Drugi način bi bio da podatke grupišemo spram funkcionalnosti, recimo da grupišemo naše entitete po modulima gde bi svaki modul bio zadužen za rešavanje grupe funkcionalnosti, a potom da za svaki modul imamo izdvojenu bazu, što bi nam smanjilo dubinu uvezivanja podataka. Recimo jedan modul bi mogao da bude rezervacije, drugi korisnici, treći kompanija.

Takođe, podatke možemo particionisati po horizontalnoj strategiji. Rezervacije i termine bismo mogli grupisati po terminima, što bi dosta olakšalo upite koji se koriste za prikaz istih na kalendaru kompanije.

3. **Predlog strategije za replikaciju baze i obezbeđivanje otpornosti na greške**

   Na projektu smo koristili relacionu PostgreSQL bazu koja pruža nekoliko strategija za replikaciju. Jedna od njih je *HotStandby*, koja koristi *Streaming* replikaciju, odnosno automatsko kopiranje transakcija između glavne (master) i replicirane baze. Master baza šalje WAL (Write-Ahead-Logging) promene svojim replikama, a sama je zadužena za očuvanje konzistentnosti u sistemu. Replike se mogu koristi za čitanje podataka, ali i za preuzimanje podataka ako master baza otkaže.

   Ukoliko bismo pratili master-slave konfiguraciju mogli bismo sve read-only zahteve usmeriti na replike, a write zahteve isključivo na master, čime bismo dosta rasteretili master bazu.

4. **Predlog strategije za keširanje podataka**
   First level cache je podrazumevano podržan u Hibernate-u, pa i u našoj aplikaciji. Za *Second level cache* koristili smo spoljnji provajder *EHCache.*

   Primenili smo strategiju *Transactional* kako bismo osigurali da se u bazi ne nađu zastareli podaci koji ne oslikavaju realno stanje sistema. Keširanjem podataka koji se ređe menjaju optimizovali smo pristup istima.

   U našoj aplikaciji keširali smo podatke o kompanijama i opremi.

5. **Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina**
   Trenutno u našoj aplikaciji, ako uzmemo da dužina polja koja imaju tip string neće prelaziti 16 karaktera, potreban memorijski prostor za skladištenje podataka o jednom registrovanom aktivnom korisniku iznosi oko 268 B. Ako bi se kroz 5 godina broj korisnika povećao na 100 miliona, okvirna je da bi nam bilo neophodno oko 24 GB za podatke o korisnicima.

Ukoliko uzmemo da je za podatke o kompaniji koja ima 5 admina kompanije potrebno oko 180B (nisu uračunati termini, jer su uračunati u rezervacije), a za podatke o jednoj medicinskoj opremi koju ta kompaniju nudi oko 550B i da kompanija ima u proseku 10 oprema u ponudi za podatke o jednoj kompaniji bi bilo potrebo oko 5,5KB. Ako okvirno uzmemo da bi broj kompanija mogao da poraste na 1000, gde bi svaka nudila oko 100 različitih oprema, procena je da bi nam bilo neophodno oko 75MB.

Podaci o jednoj rezervaciji koja ima 2 stavke rezervacije iznose oko 120B. Ako bismo za 5 godina ukupno imali oko 200 miliona rezervacija bilo bi nam neophodno oko 20GB memorije za skladištenje podataka o rezervacijama.

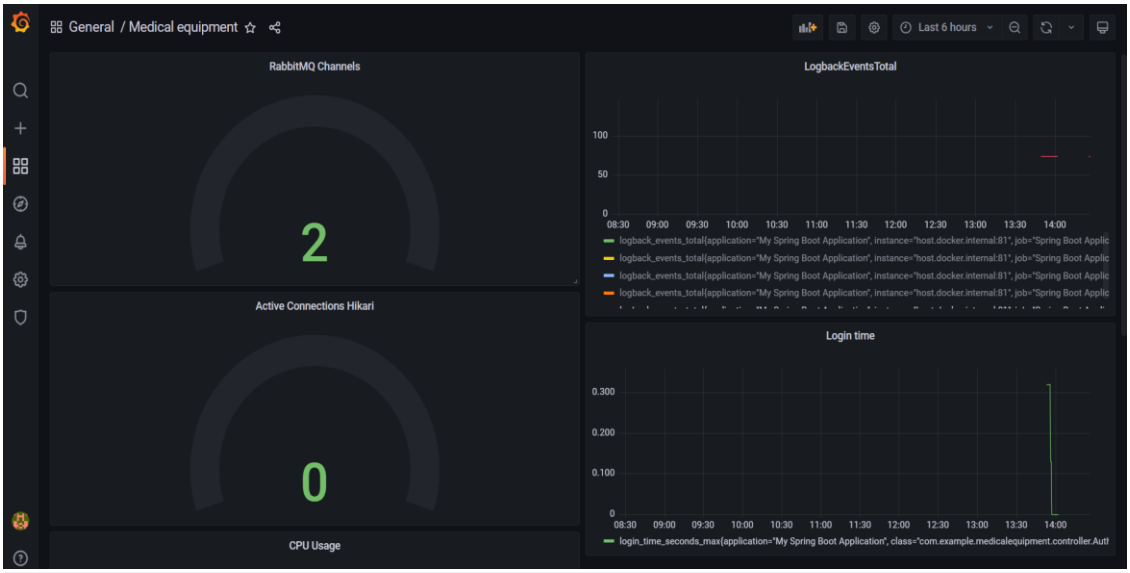
Ako na za jedan ugovor za jednu opremu trenutno treba oko 50B, i ako bi smo za 5 godina imali oko 200 takvih ugovora, trebalo bi nam oko 20KB za ugovore.

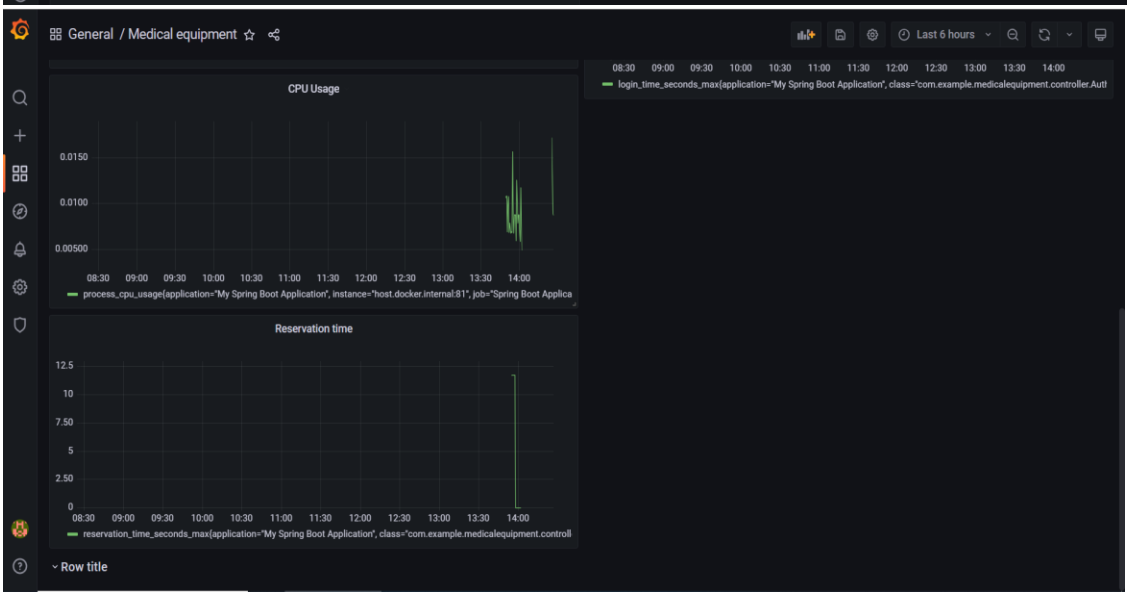
**Dati podaci su okvirne procene i nisu obuhvaćeni svi podaci jer nisu urađene funkcionalnosti kojih smo prethodno oslobođeni, poput loyalti programa, ocenjivanja kompanija, izveštaja, tako da bi se sigurno potrebni hardverski resursi povećali kada bi se i ti podaci uzeli u obzir.

6. **Predlog strategije za postavljenja load balansera**
   Postavljanjem jednostavnog *nginx*-ovog, *round robin* load balansera omogućili bismo da se zahtevi ravnomerno raspoređuju na servere, što bi poboljšalo dostupnost naše aplikacije. Loša strana ovog rešenja bi možda bila ako nisu svi serveri istih performansi i opterećenja, ali pretpostavićemo da jesu.

7. **Predlog koje operacije treba nadgledati u cilju poboljšanja sistema**
   Praćenjem događaja, odnosno aktivnosti korisnika sistema možemo unaprediti tehničku i biznis stranu naše aplikacije. Sa tehničke strane značajni bi nam bili podaci o tome u kom periodu dana, godine, na koje datume je aktivnost najveća, kako bismo mogli obezbediti bolje resurse.
   Sa biznis strane bile bi nam interesante informacije o tome u koju kategoriju spadaju našu korisnici, koje su najčešće aktivnosti, pretrage i greške istih, kako bismo videli šta treba unaprediti.

Trenutno u našoj aplikaciji pratimo opšte informacije poput broja message queue koji su trenutno aktivni, broja otvorenih konekcija ka bazi, vreme maksimalnog odgovora servera za česte operacije kao što su logovanje i rezervacija, iskorišćenja procesora korišćenjem open source aplikacije *Prometheus*, a za grafički prikaz istih iskorišćen je *Grafana*.





8. Kompletan crtež dizajna predložene arhitekture