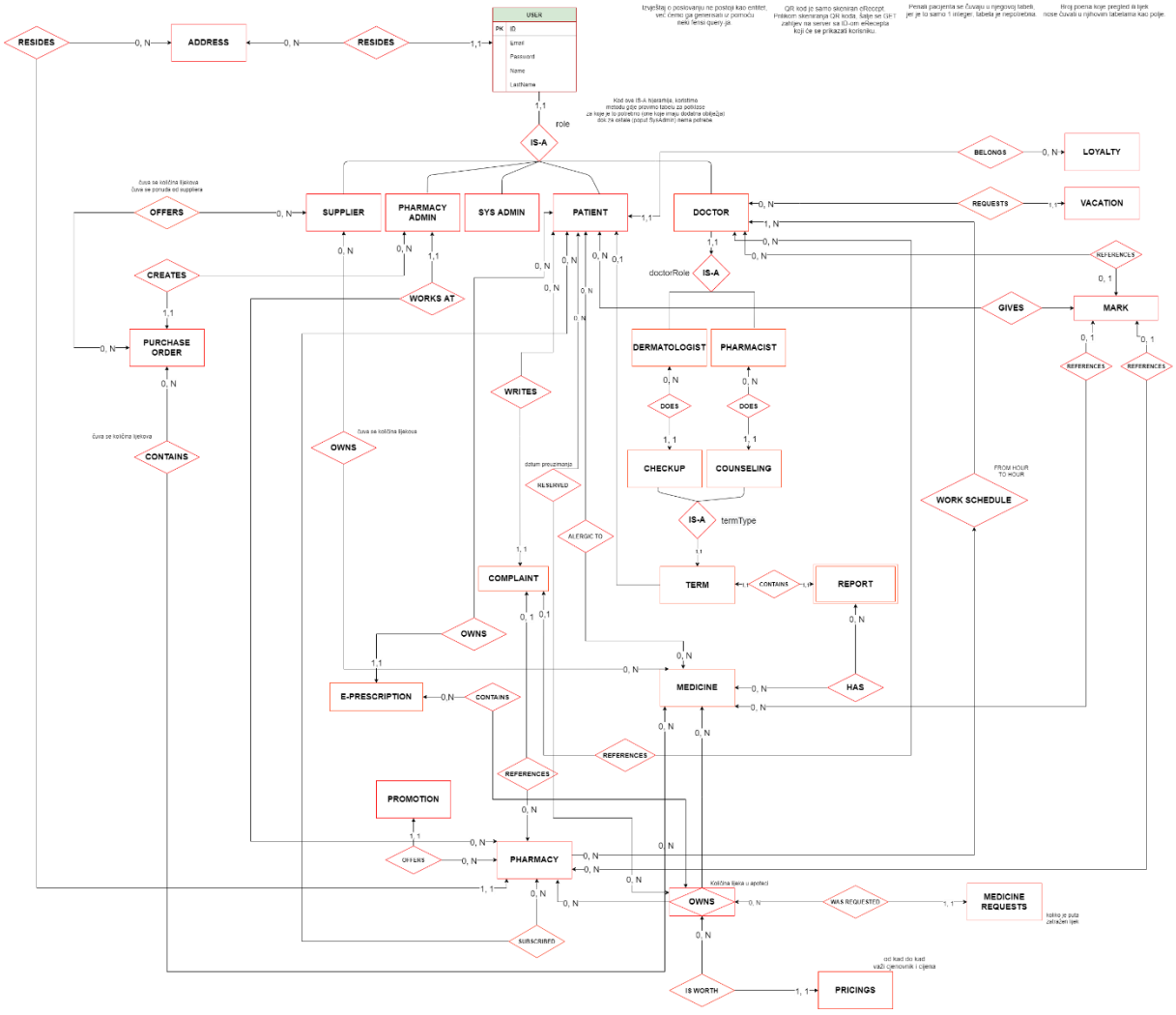
Skalabilnost

Zbog velike količine korisnika koja bi u budućnosti koristila nas softver najbolje i najisplativije rešenje je korišćenje Cloud-a. Kako vecina cloud provajdera(*Google Cloud,Digital Ocean..*) kao se servis u sebi ima Kubernetes nas predlog je koriscenje istih. Kupovina hardvera za tako veliki broj korsnika je veoma skupa, a pored toga zahteva i razne druge stvari kako bi se ti serveri redovno i ispravno održavali(*mora postajati Data centar*).

- **Dizajn šeme baze podataka**



- **Strategija za particionisanje podataka**

Relacione baze podataka je jako tesko horizontalno particionisati.Pošto nas sistem koristi PostgreSql strategija koju bismo predložili je vertikalno particionisanje.Podaci koji su “slow-moving” mogu se odvojiti od dinamičkih podataka i oni su dobar kandidat za keširanje u memoriji aplikacije. Time

omogućujemo bolju sigurnost podataka jer se osetljivi podaci mogu čuvati na posebnim particijama, smanjujemo kolicinu istovremenog potrebnog pristupa itd.

- **Predlog strategije za replikaciju baze i obezbedjivanje otportnosti na greške**

Pošto bi se koristio Kubernetes možemo postaviti vise replikacija baza podataka („slave“) . Ukoliko dođe do padanja baze Kubernetes servisi ce podici neku od replikacija i time postižemo “persistency” naših podataka.

- **Predlog strategije za kesiranje podataka**

Strategija koja bi bila najpogodnija za keširanje podataka u našem slučaju je Write-Around. Podaci se direkno upisuju u bazu, dok se u keš upisuju prilikom read-a. Veliki deo naših podataka neće biti korišten odmah nakon upisa, termini se definišu unapred , takođe nabavke lekova se isto rade unapred (možda čak i sezonski) tako da bi drugi način keširanja dovodio do bespotrebne potrošnje resursa.

- **Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina**

Procena za skladištenje podataka ovako velike aplikacije može biti veoma nezahvalna, pogotovo nekome ko nema iskustva sa radom tako velikih sistema. Istraživanjem došli smo do podatka da za 1 bilion redova u PostgreSql bazi podataka „košta“ 100gb hardverskog prostora.Naša procena je da nam je za godinu dana dovoljno 4 biliona redova.Za period od 5 godina to nam iznosi 2TB prostora.Međutim posle 2 godine zastarele podatke bimo skladištili u zipovane backup-e, racunajući da okvirno zip format smanjuje veličinu za 30% . Tako dobijamo 2 godine po punih 400gb i 3 godine u zipovanim fajlovima po 280gb. To ukupno iznosi 1.64TB od kojih je aktivno u bazi podataka 800gb dok je ostatak skladišten kao zip.

- **Predlog strategije za postavljanje load balansera**

Posto je naša aplikacija globalno rasporstranjena sa jako puno korisnika dok su svi zahtevi slične “ težine” verovatno najbolja strategija za postavljanje *load balansera* je *Cross-region*. Svaki od servera opluživao bi svoj region

time bismo dobili veoma malo kašnjenje odgovora. Pored toga mogli bismo da menjamo regionalne *endpoint-e* bez uticaja na korisnike. Ukoliko se desi da regionalni server padne zahtevi se prosleđuju na drugi koji mu je najbliži.

- **Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema**
  - Zakazivanje termina kod određenog lekara, koji je lekar najviše tražen
  - Zakazivanje termina u određenom vremenu, koje je vreme najčešće traženo
  - Zakazivanje termina u određenoj apoteci, koja apoteka je najtraženija
  - Rezervacija leka, koji lek je najtraženiji
  - Izdavanje leka, koji procenat rezervacija je na kraju izdato
  - Davanje ocena, pacijenti sa koliko loyalty points-a daju koje ocene
  - Davanje ocena,da li je doktor sa najboljim ocenama najviše tražen

- **Crtež dizajna predložene arhitekture**