INTERNET SOFTVERSKA ARHITEKTURA

Proof of Concept

Grupa 10

UVOD

Ovaj projekat ima za cilj da implementira web aplikaciju koja ce funkcionisati kao centralizovani informacioni sistem za kompanije specijalizovane u nabavci medicinske opreme.Ova aplikacija ce omoguciti privatnim bolnicama da na najjednostavniji nacin dobijaju isporuke zeljene opreme na njihove lokacije od strane izabrane komanije

.Ideja jeste da se spoje kompanije i bolnice sa najblizim lokacijama radi sto efikasnijeg rada.Pored toga aplikacija ce omoguciti i licni dolazak po opremu na prostorije kompanija u hitnim ili nekim posebnim slucajevima,naravno u dogovoru.
Cilj je napraviti sistem koji moze I da obsluzi I da isprati rad velikog broja korisnika u svakom momentu jer to u ovoj industriji moze puno da znaci

- **Laka I brza komunikacija izmedju kompanija I bolnica**
- **Jednostavan prenos opreme I pracenje porudzbine**
- **Efikasan rad sa placanjem**
- **Azuran I vrlo efikasan sistem**
- **Izuzetna mogucnost za skalabilnost**

Ovaj proof of concept je tu da bi vam prikazao kako nas sistem planira da podrzi rad ogromnog broja korisnika.

SKALABILNOST:

Ovde cemo vam prikazati Proof of Concept nase arhitekture gde cemo govoriti o tome kako cemo implementirati sistem ciji:

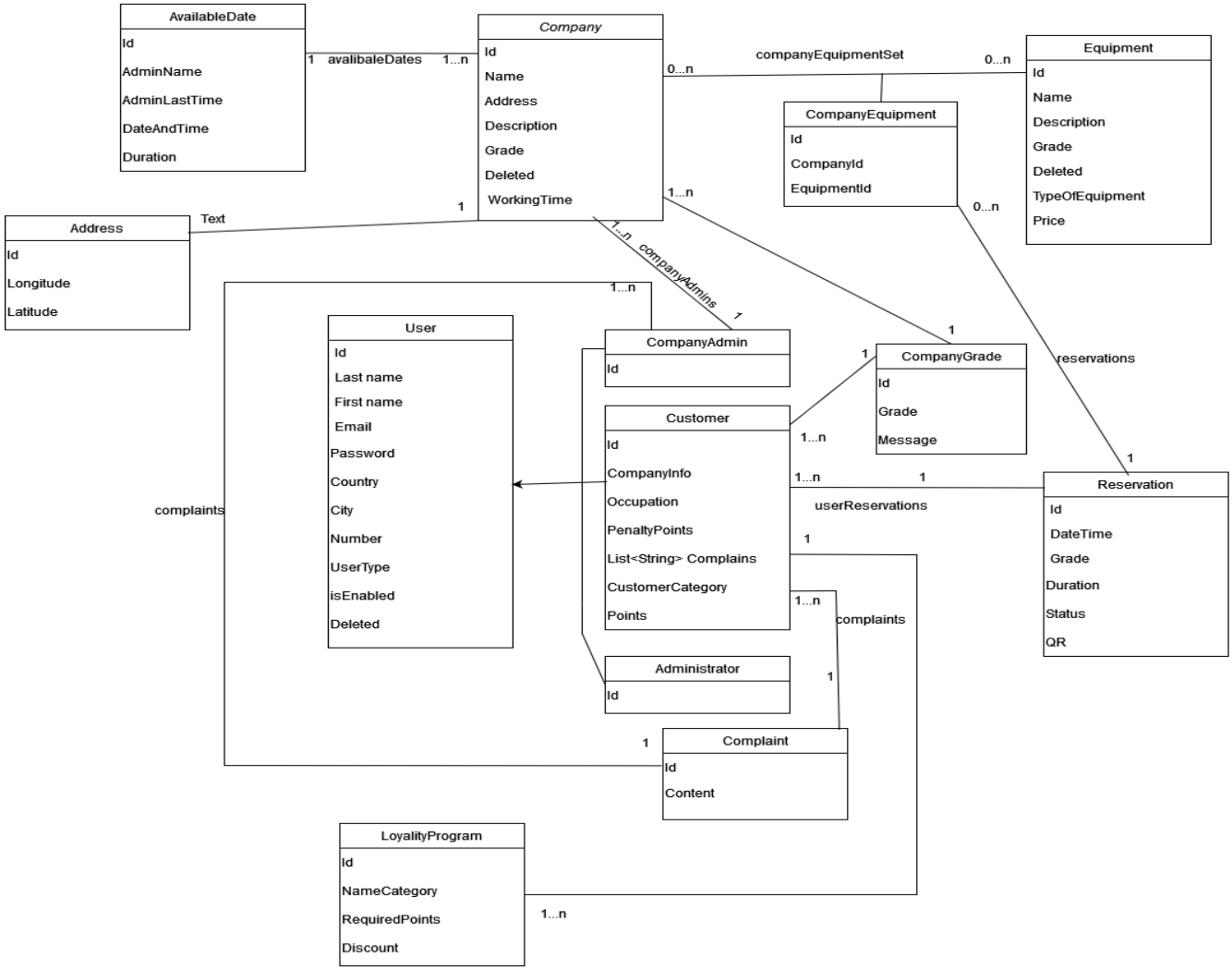
- **Ukupan broj korisnika aplikacije koji iznosi 100 miliona,**
- **Broj rezervacija korisnika koji je na mesecnom nivou 500 000.**
- **Sistem visoke skalabilnosti I dostupnosti**

Pregled tema

- **1.Dizajn seme baze podataka.**
- **2.Predlog strategije za particionisanje podataka .**
- **3.Predlog strategije za replikaciju baze i obezbedjivanje otpornosti na greske.**
- **4.Predlog strategije za kesiranje podataka.**
- **5.Okvirna procena za hardverske resurse potrebne za skladistenje svih podataka u narednih 5 godina.**
- **6.Predlog strategije za postavljanje load balansera.**
- **7.Predlog koje operacije korisnika treba nadgledati u cilju poboljsanja sistema.**
- **8.Predlog strategije za primenu rate limitera.**
- **9.Crtez dizajna predlozene arhitekture.**

1.Dizajn seme baze podataka

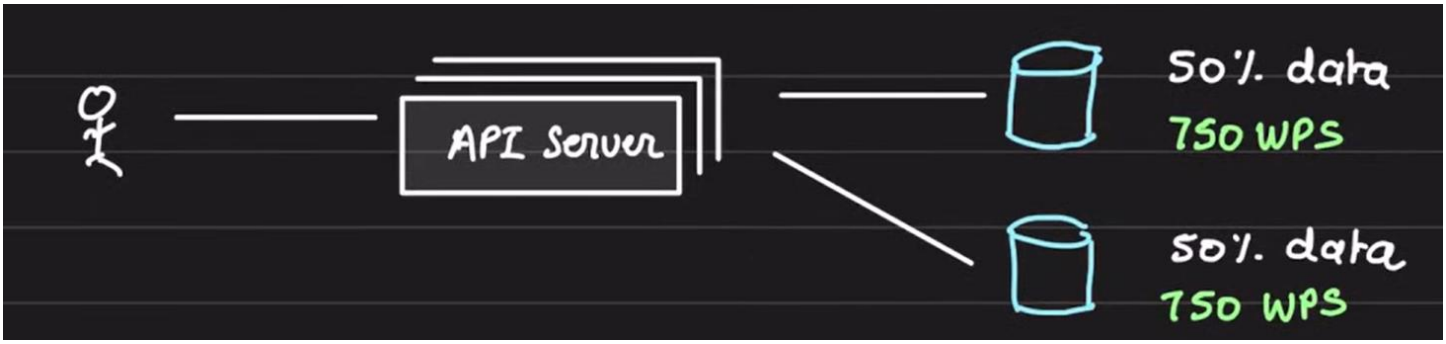
Ovde predstavljamo dizajn seme baze podataka naseg trenutnog protitipa.Da bi u najboljem svetlu prikazali nasu ideju I aplikaciju kreiran je prototip koji je tu da pokaze korisnicima kako bi ta aplikacija izgledala.Ideja prototipa je osnovni rad aplikacije I feedback korisnika a u daljim tackama je objasnjeno kako cemo izmeniti bazu radi skaliaranja da bih mogla da podrzi prethodno pomenuti broj korisnika.



2.Predlog strategije za particionisanje I sharding podataka

Kada broj zahteva za bazu postane preveliki I kada "vertical scaling" premasi svoje mogucnosti moramo naci alternativne nacine da nastavimo da povecavamo kapacitet nase baze .Sharding predstavlja koncept gde cemo mi uvesti nove baze podataka da nam pomognu("Horizontal scalling"), dok particionisanje predstavlja podelu podataka u te razlicite baze.Particionisanje podataka se moze odraditi I na jednoj bazi ali zbog prevelikog broja korisnika koji mi zelimo,mi cemo uvesti I nove baze shardingom.

Sharding:



Posto je nasa aplikacija radi sa kupcima koji dolaze po posiljke I sa bolnicama kojima mi saljemo nase equipment ideja naseg tima jeste da mi odradimo sharding trenutne baze prototipa na baze 3 baze I da trenutne podatke podelimo u njih particionisanjem.

1.Prva baza ce sadrzati podatke u ugovorima sklopljenim sa bolnicama ,njihove informacije o njima I raditi sa njihovim rezervacijama I slanje opreme njima.

2.Druga baza ce raditi sa obicnim customerima koji dolaze I koji zakazuju licno I preuzimaju licno opremu.

3.Treca baza biti "spona" izmedju dve a to ce biti ona koja ce sadrzati informacije o Kompanijama I njihovim opremama.

Na ovaj nacin mi cemo drasticno smanjiti opterecenje na nas sistem.

3.Predlog strategije za replikaciju baze I obezbedjivanje otpornosti na greske.

Replikacija baze podataka je process u kojem se podaci iz jedne baze kopiraju I distribuiraju u druge baze.
Postoje razliciti tipovi strategija za replikaciju I mi smo se odlucili za koriscenje **"READ REPLICA"**

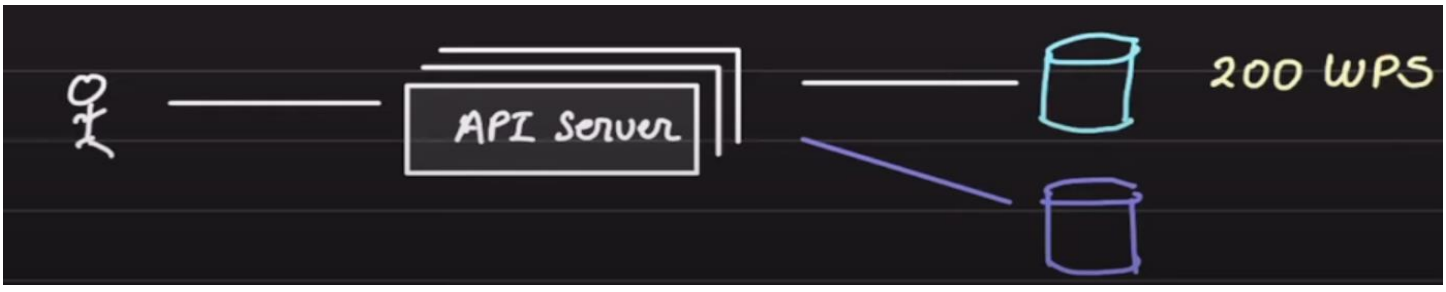
READ REPLICA:

To je tip replike koji se koristi za citanje podataka sto znaci da je moguce citait podatke iz ove replike ,time ce svaki "GET" zahtev biti preusmeren na nju sto ce drasticno smanjiti opterecenje na master bazu.Kao I svaka druga replica ona predstavlja kopiju master baze I pomaze u obezbedjivanju otpornosti na greske prilikom rada.

Prednosti:

- **Distribucija opterecenja :** Koriscenjem read replica moguce je distribuirati opterecenje citanja izmedju glavne baze podataka I replica.Aplikacija rutira citacke zahteve ka njoj I tako ravnomerno rasporedjuje opterecenje izmedju razlicitih servera I poboljsava performance sistema
- **Otpornost na greske :** U slucaju da glavna baza podataka postane nedostupna ili dodje do kvara,ova replica omogucava nastavak citanja podataka cak I ako je glavna baza pala,sto povecava otpornost na greske I smanjuje prekid u pristupu podataka.
- **Tehnicko odrzavanje:** Koriscenjem read replica moguce je izvesti radnje odrzavanja,poput izvodjenja kopija,sigurnosnih kopija ili izvrsavanja analitickih upita,bez uticaja na performance glavne baze podataka ili operativnu funkcionalnost aplikacije.

Read replica:



4.Predlog strategije za kesiranje podataka

Kesiranje je tehnika koja se koristi kako bi se poboljsala performansa I efikasnost sistema cuvanja cesto koriscenih podataka ili resursa na brzem ili lakse dostupnom mestu (lokalnom skladistu).

Mi u nasoj aplikaciji koristimo **MEMORY CACHING** , koji ce nam pomoci da cuvamo rezultate cesto koriscenih upita.Ovo nam daje sposobnost brzeg pristupa podacima jer se citanje iz ove memorije obavlja mnogo brze nego citanje sa diska.

Mi u nasem prototipu imamo mikro primer cachiranja koji je implementiran na stranici za rezervacije ,posto za nju smatramo da ce biti najcesce koriscenja,kada se udje na stranicu prvi put podaci ce se cuvati u cachu odakle svaki sledeco put u jednoj sesiji cemo ih uzimati ako nam trebaju ,time ne pristupamo bazi vise puta sto nam drasticno povecava skalabilnost.

Nas tim se odlucio za Ehcache kao eksterni provider.

```
User@aa2ce5c3
INFO 18244 --- [e [_default_]-6] com.ISA.ISAProject.utils.CacheLogger     : Key: Company2 | EventType: EXPIRED | Old value: com.ISA.ISAPro
INFO 18244 --- [e [_default_]-6] com.ISA.ISAProject.utils.CacheLogger     : Key: Company2 | EventType: CREATED | Old value: null | New val
User@aa2ce5c3
```

5.Okvirna procena za hardverske resurse potrebne za skladistenje svih podataka u narednih 5 godina

- **Baze podataka**:
  Treba odabrati bazu podataka koja moze podrzati veliki broj korisnika I efikasno upravljati podacima.To mogu biti SQL baze podataka poput PostgreSQL ili MySQL ili NoSQL resenje poput MongoDB ili Cassandra.Nas tim se odlucio da protip radi na PostgreSQL

- **Serveri**:
  Potrebno je odabrati servere sa dovoljno procesorkse snage ,memorije I diska za podrsku aplikacije.Najbitnije je da su serveri mogu da se HORIZONTALNO skaliraju kada dostignemo limit vertikalnog skaliranja.Ovi su server su visokih performanis.
  Neki od servera :
  1.Blade Servers
  2.Cloud Servers
  3.Rack Servers
  4.Dedicatet Servers

- **Diskovi**:
  Potrebno je dovoljno prostora na diskovima za skladistenje podataka u narednih 5 godina.To ukljucuje prostor za bazu podataka , rezervisan proctor za sigurnosne kopije podataka I druge resurse sistema.
  1.SolidStateDrives(SSD)
  2.NVMe SSDs(Non-volatile Memory Express)
  3.Cloud Storage
  4.Hybrid Drives

- **Backup**:
  Planirati strategiju backup-a podataka I redudancije kako bi se osiguralo da podaci budu sigurni I dostupni cak i u slucaju kvara sistem ili gubitka podataka
  1.Network attached Storage (NAS)
  2.Cloud backup

- **CPU**: Potrebna brzina I snaga cpu moz varirati ali cesto se koriste server sa vise jezgara(npr 16,32,64 ili vise jezgara) sa visokim brzinama takta(2.5GHz ili vise) kako bi se osigurala dobra obrada zahteva.

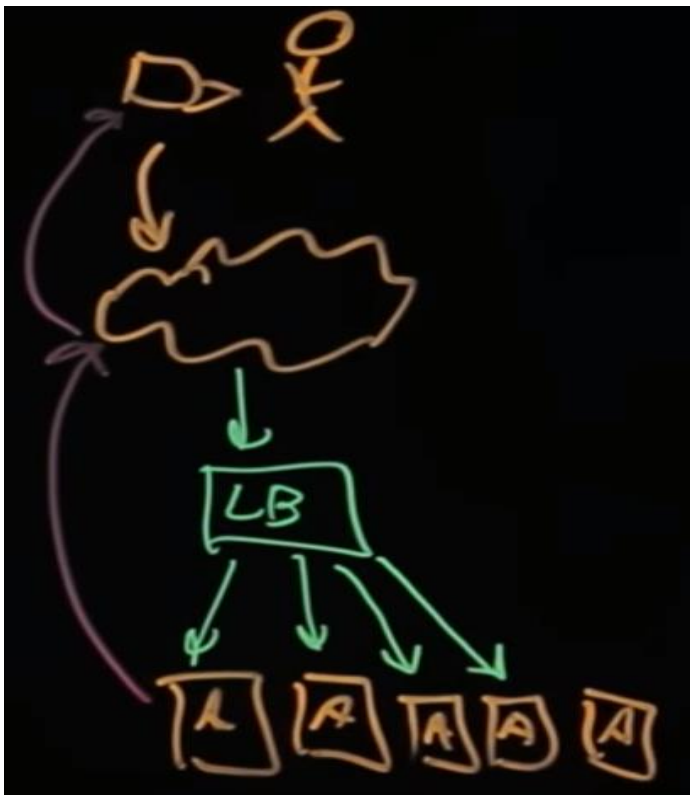
- **RAM**: Potrebna ram memorija zavisi od mnogo faktora:Od tipa aplikacije,Tip opterecenja,Cach memorija,Efikasnost koda…
  Za ovakve potrebe koriste se server sa velikom kolicinom RAM memorije(64GB ,128GB ili vise RAM-a)

6.Predlog strategije za postavljanje load balancera

Kada pravimo veliku aplikaciju neophodno je skaliranje servera.Serveri se skaliraju vertikalno do jednog momenta nakon cega je potrebno uvesti horizontalno skaliranje , sto znaci povecanje broja aplikativnih servera kojima cemo sada podeliti posao.Onaj koji razvrstava koji posao ide kome serveru jeste **LOAD BALANCER**.

Postoje razlicite strategije za implementiranje load balancera ,mi smo se odlucili za **SMART LOAD** Balancer.
On iako jeste skup I najtezi za konfiguraciju mi smatramo da je neophodan za jedan ovakav sistem.

Ovaj tip load balancera ne bira sam gde ce koji zahtev sa fronta ici vec ce se direktno "dogovarati" sa nasim aplikativnim serverima sto pospesuje rad I protok informacije dok istovremeno smanjuje opterecenje na servere.



7.Predlog koje operacije korisnika treba nadgledati u cilju poboljsanja sistema

Kada govorimo o skaliranju projekta nadgledanje operacija korisnika moze biti kljucno za identifikaciju potencijalnih tacaka optimizacije I poboljsanja performansi sistema.Evo nekoliko operacija korisnika koje cemo pratiti radi poboljsanja sistema :

- **1.Kreiranje rezervacije** – Kao glavna komponenta na kojoj se zasniva rad nase aplikacije I sa kojom korisnici najvise komuniciraju potrebno je pratiti njen rad I traziti nacin poboljsanja
- **2.Datumi** – Hocemo da pored dodavanja skalabilnosti nase aplikacije I dalje ocuva precizan rad sa datumima kako ne bi doslo preklapanja termina korisnika
- **3.Position simulator** – Kao jedna od najatraktivnijih operacija korisnicima kao I zbog bezbednosti potrebno je konstantno pratiti I unapredjivati simulator
- **4.Kesiranje** – za nas kao developere je bitno da obratimo paznju u buducnosti na operacije koje cemo stavljati u kes da ne dodje I do njegovog neoptimalnog koriscenja kada se poveca broj korisnika.

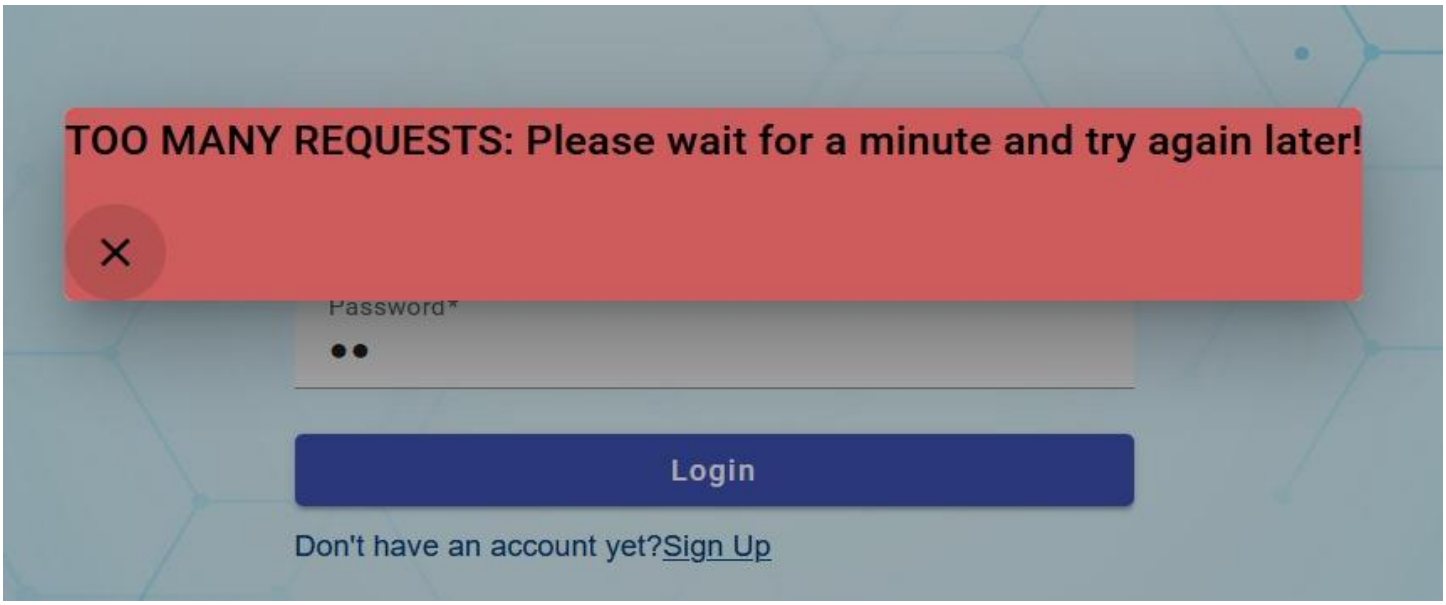
8.Predlog primene strategije za primenu rate limitera

Sa porastom broja korisnika aplikacije, raste i broj zahteva koje server treba da opsluži. U takvim slučajevima želimo da izbegnemo situacije u kojima server može biti preopterećen. RateLimiting je jedan od načina ograničavanja opterećenja servera. Bezbednost aplikacija je bitan aspekt svakog razvoja. Jedan od čestih napada sa kojima se srećemo je i DoS napad. Jedan od načina zaštite od ovakvih napada jeste ograničavanje broja poziva upućenih na naš API.

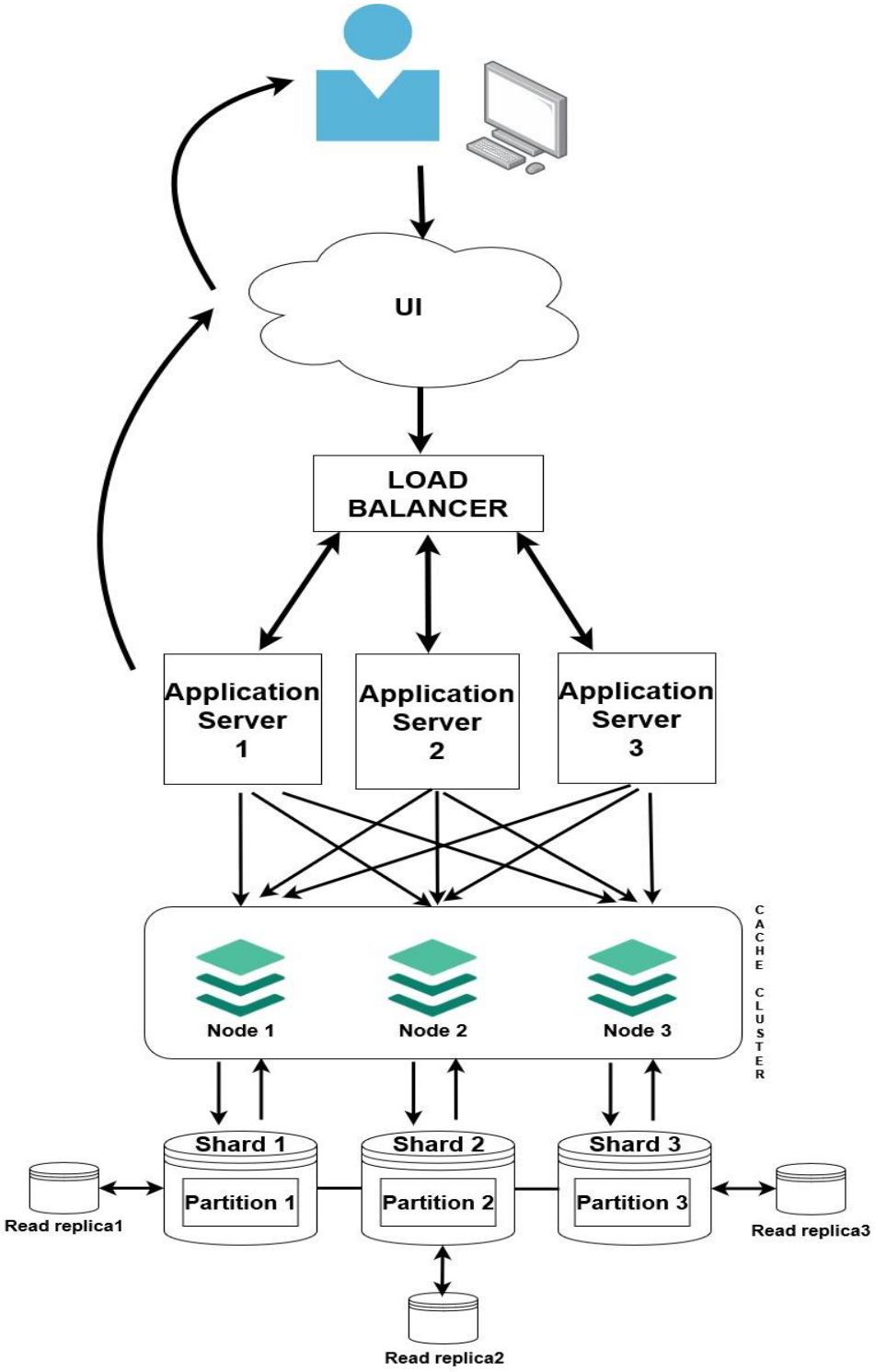
Mi smo u nasoj prototip aplikaciji implementirali rate Limiter na logovanju,koji dozvoljava korisniku da proba da se uloguje 5 puta prilikom jedne sesije, ako se to ne ostvari morace da saceka neki odredjen period vremena pre nego sto bude mogao opet da proba da se uloguje.

U zavisnosti od razvijanja nase aplikacije mozda bude neophodno I druge stvari staviti pod restrikciju limitera kako bi odrzali brz I efikasan rad nasih servera.

Mi u nasem prototipu koristimo Resiliance4j biblioteku.



9.Crtez dizajna predlozene arhitekture



Zakljucak

U zakljucku,nas proof of concept pruza uvid u planiranu arhitekturu naseg sistema I kako nameravamo da podrzimo veliki broj korisnika u svakom trenutku.Sistem koji smo osmislili ima za cilj da bude pouzdan efikasan I skalabilan , omogucavajuci nasoj web aplikaciji da bude kljucni alat za kompanije I bolnice u oblasti nabavke medicinske opreme.

Literatura:

- https://www.netguru.com/blog/proof-of-concept-in-software-development - PoC
- https://www.youtube.com/watch?v=sCR3SAVdyCc&ab_channel=IBMTechnology – Load balancer
- https://bp.powerschool-docs.com/bp-documentation/latest/23-5-x-hardware-and-software-requirements - hardverski zahtevi
- https://www.youtube.com/watch?v=wXvljefXyEo&ab_channel=ArpitBhayani – Database sharding partitioning and replication
- https://www.youtube.com/watch?v=bP4BeUjNkXc&t=221s&ab_channel=SoftwareDeveloperDiaries + ftn vezbe – caching
- rateLimiter – ftn vezbe

Napomene:

- **1.Crtala konacnu arhitekturu – Nina Knezevic**
- **2.Kreirala logo – Katarina Medic**
- **3.Licencirao ime kompanije – Spasoje Brboric**
- **4.Sat vremena trazio kako se brise stranica u wordu -Aleksandar Srajer**

I dalje nisam skontao kako…