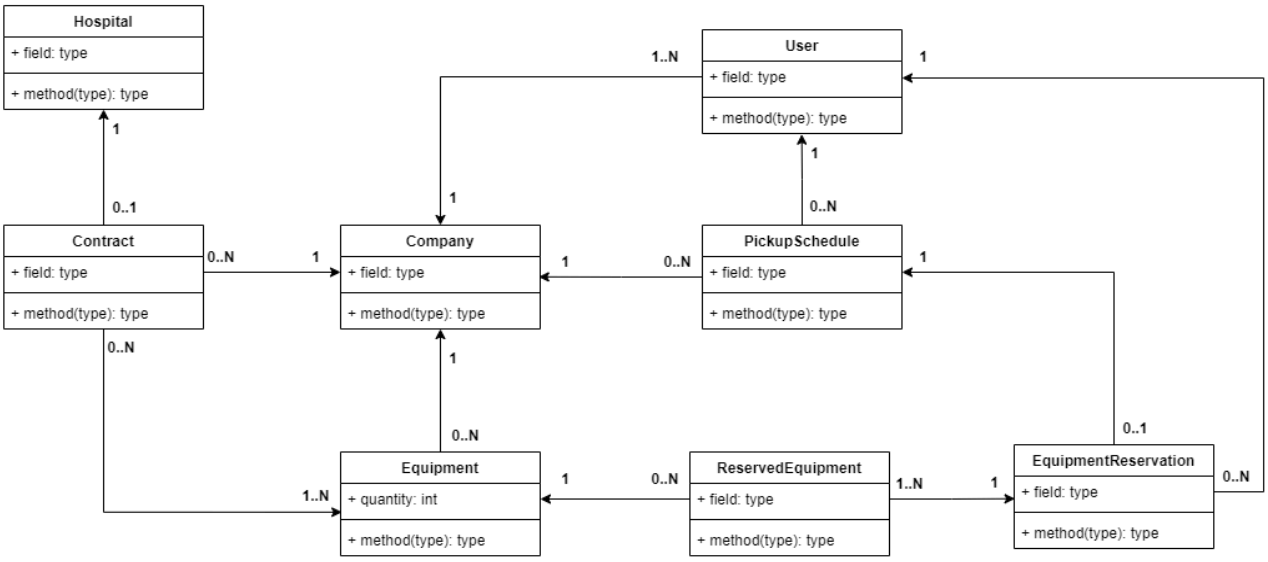
Proof of Concept

1. Dizajn šeme baze podataka



2. Predlog strategije za particionisanje podataka

Sto se particionisanja podataka tice - opredelili smo se za horizontalno skaliranje Vertikalno skaliranje nema puno smisla, jer su sve relacije(tabele) poprilicno jednostavne (3-6 obelezja), i podaci su logicki povezani, tako da je logicka podela na nezavisne celine gotovo nemoguca, sto bi dovodilo do konstantnog spajanja tabela, i znacajnog opadanja performansi Buduci da se ocekuje na stotine miliona korisnika, moze se izvuci zakljucak da se radi o globalnoj aplikaciji Tako da horizontalno skaliranje mozemo primeniti u pogledu podele kompanija i korisnika na osnovu geografskog regiona (imali bi posebne shardove za svaki region), pogotovo zbog prirode aplikacije - korisnik mora da preuzme opremu iz kompanije u kojoj je izvrsio rezervaciju (sto nam govori da su korisnik i kompanija u istom regionu) Rezervacije takodje imaju vise smisla da budu distibuirane nego centralizovane (horizontalno skaliranje).

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

Replikacija baze podataka predstavlja ključnu strategiju koja omogućava optimizaciju performansi, otpornost na pad sistema i povećanu dostupnost baze podataka, pružajući istovremeno dodatni sloj zaštite od potencijalnih gubitaka podataka. U ovom kontekstu, primenjivali bismo Master-Slave strategiju, gde bi glavna baza podataka imala ulogu centralnog entiteta zaduženog za upisne (write) zahteve, dok bi sporedne baze podataka bile specifično namenjene za izvršavanje čitanja (read) zahteva.

Odluka da se izbegne primena Master-Master strategije proizlazi iz potrebe za preciznim rukovanjem terminologijom i ograničenjima dostupne opreme. Master-Master strategija, iako pruža određeni stepen horizontalne skalabilnosti, ne garantuje totalnu konzistentnost podataka, što bi moglo izazvati nepredvidive posledice. Stoga, kako bismo održali visok nivo konzistentnosti, planiramo korišćenje Synchronous update pristupa.

U scenariju otkaza glavne baze podataka, planiramo implementaciju strategije automatske promene uloge, gde bi jedna od sporednih baza podataka bila unapređena u glavnu. Ova tranzicija biće realizovana kroz upotrebu alata poput PgBouncer ili HAProxy, čime će se obezbediti besprekorna kontinuitet rada sistema. Unapređena baza podataka preuzeće odgovornost za upisne (write) zahteve, čime će osigurati neprekidan rad sistema u slučaju otkaza glavne baze.

Kao ključni tehnološki sastojci, koristićemo psycopg2 za podršku PostgreSQL bazi podataka, dok će django-db-readonly biti angažovan za efikasno usmeravanje read/write operacija. Uz to, definisaćemo specifične rutere kako bismo precizno usmeravali operacije čitanja i pisanja prema odgovarajućim bazama podataka, pridržavajući se postavljene strategije i obezbeđujući visok nivo efikasnosti i pouzdanosti sistema.

4. Predlog strategije za keširanje podataka

U cilju optimizacije performansi i efikasnog rukovođenja povećanim brojem istovremenih korisnika, predlažemo implementaciju strategije keširanja za prikaz kompanija i opreme, naročito fokusirajući se na keširanje prvih nekoliko stranica koje se najčešće traže. Takođe, planiramo keširanje prvih stranica najčešće primenjenih filtera kako bismo dodatno ubrzali odziv aplikacije. Trenutna implementacija keširanja koristi memoriju servera, međutim, kako bismo obezbedili skalabilnost i efikasnije upravljanje keširanim podacima, planiramo prelazak na korišćenje NoSQL baze podataka Redis. Prelazak na Redis: Redis će biti korišćen kao backend za keširanje

umesto memorije servera. Redis je brza, in-memory NoSQL baza podataka koja omogućava efikasno keširanje podataka. Ova promena omogućiće nam bolje rukovanje velikim brojem istovremenih korisnika i povećati brzinu pristupa podacima.

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

Pretpostavke:
- ukupan broj korisnika aplikacije je 100 miliona,
- broj rezervacija svih entiteta na mesečnom nivou je 500.000,
- sistem mora biti skalabilan i visoko dostupan.

User: 184 bajta * 100 000 000 = 1 840 000 000 ~ 1.84GB
Company: Ukupno: (50 + 50 + 240 + 50 + 50 + 2 + 3 + 3) = 448 * c ~
Equipment: Ukupno: (4 + 50 + 240 + 4 + 4 + 4 + 1) = 307 * e + 4 * e

c - broj kompanija
e - broj oprememe

PickupSchedule: Ukupno: (4 + 4 + 4 + 3 + 3) = 500000 * 18 * 12 * 5 * + 500000 * 12 * 5 * (4 + 3*4) = 1.02GB*
EquipmentReservation: Ukupno: 15 ~ 15 500000 * 12 * 5 * + 500000 * 12 * 5 * (4 + 4) = 0.69GB
ReservedEquipment: Ukupno: 4 ~ 12 * X * 500000 * 12 * 5 = 0.36 * X gb X - broj rezervisane obreme Ukupno (1.84 + 448 * c + 311 * e + 1.02 + 0.69 + 0.36 * X)GB

gdje je:
c: broj registrovanih kompanija u sistemu
e: broj registrovane opreme u sistemu
X: broj rezervisane opreme

6. Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

U trenutnoj implementaciji aplikacije, najveci deo saobracaja cine rezervacije. Tako da bi to bila prva i najbitnija stavka za monitoring

- koliko se prosecno opreme nalazi u jednoj rezervaciji
- takodje s' obzirom na planirani prelazak na mikroservisnu arhitekturu, bitno je ustanoviti u kojim se geografskim regionima odvija najveci saobracaj, kako bi obezbedili potrebnu infrastrukturu koja isti moze podrzati

- neophodno je vrsiti

7. Kompletan crtež dizajna predložene arhitekture