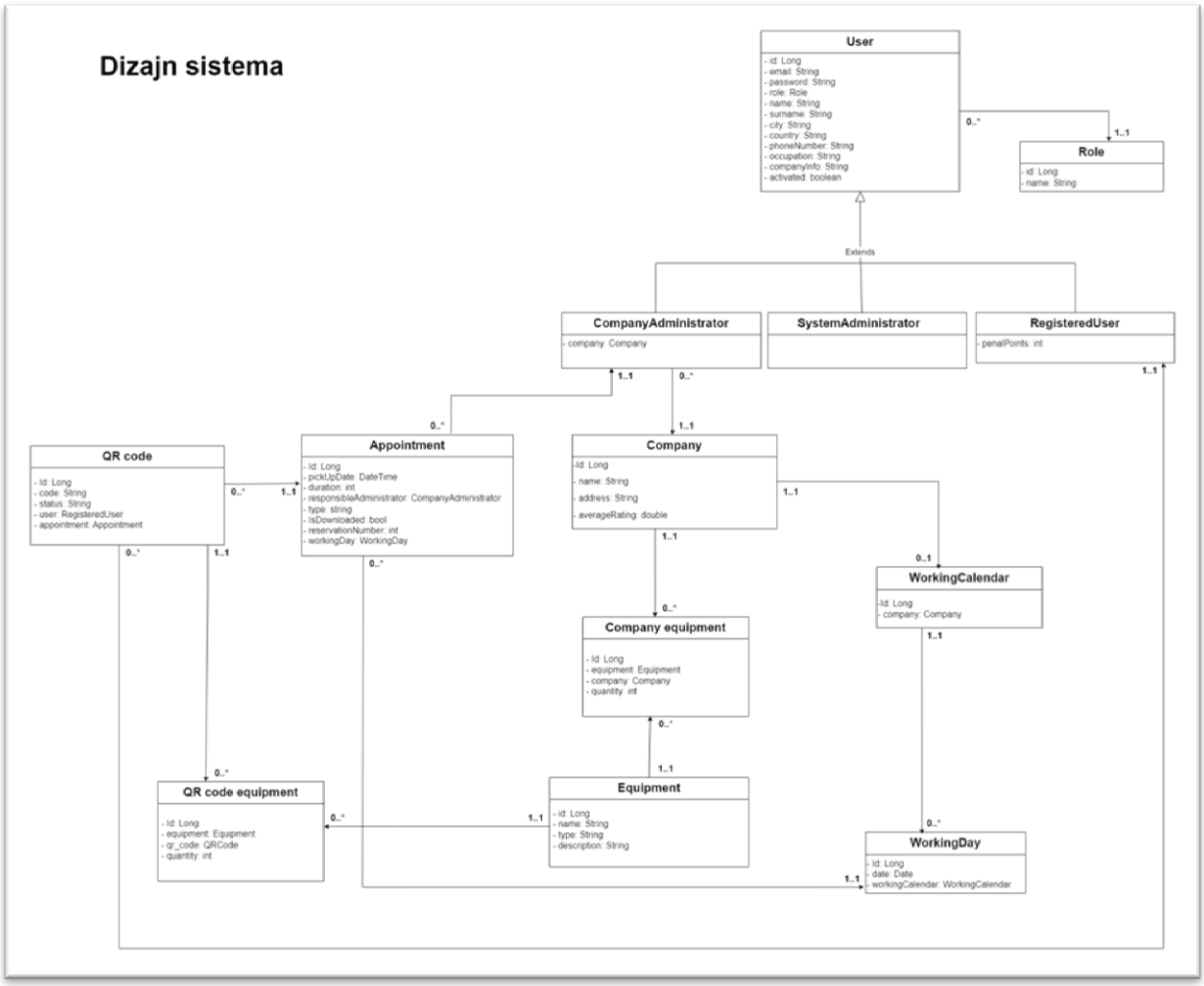
Proof of concept

Tim 25

1. Dizajn šeme baze podataka



2. Predlog strategije za particionisanje podataka

Parcionisanje podataka podrazumeva logičku podelu velikih skupova podataka na manje, upravljive segmente. Kroz strategijsko deljenje podataka na osnovu relevantnih kriterijuma možemo značajno poboljšati celokupne performanse sistema.

Za horizontalno particionisanje bih napravio podelu po geografskoj lokaciji, vremenskim zonama i vremenskim periodima (meseci ili periodi dana) koji bi imali velike varijacije računajući da naša aplikacija treba da podrži 100 miliona korisnika i 500.000 rezervacija na mesečnom nivou. Na osnovu rasprostaranjenosti i angažovanosti naših korisnika bi mogli izabrati optimalan odnos između ovih strategija.

Što se tiče vertikalnog particionisanja izvršio bih podelu po poslovnoj logici i odvojio podatke koji se koriste manje frekventno unutar klasa, kao što su npr. detalji o kompaniji, adresi i zaposlenju korisnika.

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

Računajući da naša aplikacija trenutno funkcioniše samo sa jednom bazom podataka, da bi skalirali i obezbedili otpornost na greške moramo uvesti nekoliko instanci baze podataka.

Međusobni odnos ovih baza podataka bi bila master-slave, gde bi se sve izmene tj. write komande izvršavale na master bazi podataka. Te izmene bi onda onda bile replicirane na ostale slave baze podataka, sa kojih bi se izvršavale read komande.

Ovim pristupom bi bili otporniji na scenario otkazivanja jedne od baza podataka, jer bi imali njene replike i na taj način mogli da sačuvamo veliku većinu podataka.

Druga strategija koju bih takođe pomenuo je da napravimo multi-master sistem gde bi sve baze međusobno razmenjivale podatke i dozvoljavale i read i write. Na ovaj način bi mogli rasporediti baze podataka po različitim kontinentima i pobojšati brzinu komunikacije sa različitih geografskih lokacija. Morali bi i koristiti softver koji bi koordinisao baze podataka i rešavao konflikte u komunikaciji više mastera kao npr. SymmetricDS ili Bucardo.

4. Predlog strategije za keširanje podataka

Obzirom da koristimo Hibernate, to podarazumijeva da je L1 keš već uključen u našu aplikaciju. Samim tim, poboljšane su perfomanse pristupa podacima u okviru jedne sesije. Sa druge strane, kada je u pitanju drugi nivo keširanja, L2, i on je podržan od strane Hibernate-a. Njegovom uključenošću, možemo riješiti problem keširanja na

nivou cjelokupne aplikacije. Međutim, potreban je jedan eksterni provajder, poput EhCache, kako bismo do kraja konfigurisali L2 keš. Upravo je EhCache dobro podržan od strane Hibernate-a, te bi on sam bio naš izbor za tu ulogu, kao i kod najvećeg broja Spring aplikacija. EhCache može znatno ubrzati pristup podacima i smanjiti opterećenje na bazi podataka. Osnovna svrha i njegov zadatak bi bilo keširanje podataka koji se samo ponekad modifikuju i vrlo često čitaju iz baze. U našem slučaju, to bi bili podaci o kompanijama i raspoloživoj opremi u sistemu. Za strategiju keširanja bismo izabrali Transactional Read-Write. Ona bi omogućila da se svi objekti u kešu osvježavaju nakon završetka transakcije, čime se održava konzistentnost sa bazom podataka. Dakle, Transactional Read-Write donosi efikasno ažuriranje i čitanje keširanih objekata, te oni odgovaraju stvarnom stanju u bazi. Samim tim, korišćenjem adekvatnih keširanih podataka, broj upita ka bazi u okviru jedne transakcije bi bio smanjen. Kada je riječ o front-end keširanju, tu bismo se opredijelili za CDN keširanje, koje bi ubrzalo učitavanje sadržaja i, ujedno, rasteretilo server, čime bismo unaprijedili skalabilnost i pouzdanost naše aplikacije. Kako se CDN sastoji iz više servera ili čvorova širom svijeta, svaki od njih bi sadržao par keširanih kopija određenih resursa, poput HTML stranica i CSS stilova, slika i slično.

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

Appointment:

- id [long] - 8 bytes
- pickUpDate [LocalDateTime] - 8 bytes
- duration [int] - 4 bytes
- type [int] - 4 bytes
- isDownloaded [boolean] - 1 bit
- reservationNumber [int] - 4 bytes
- winnerId [long] - 8 bytes
- downloadedAt [LocalDateTime] - 8 bytes
- companyAdministratorId [long] - 8 bytes

- workingDayId [long] - 8 bytes

  TOTAL: 61 bytes

QRcode:

- id [long] - 8 bytes
- code [String] - 6* bytes
- status [int] - 4 bytes
- registeredUserId [long] - 8 bytes
- appointmentId [long] - 8 bytes

  TOTAL: 34 bytes

QREquipment:

- id [long] - 8 bytes
- qrCodeId [long] - 8 bytes
- equipmentId [long] - 8 bytes
- quantity [int] - 4 bytes

  TOTAL: 28 bytes

WorkingCalendar:

- id [long] - 8 bytes
- companyId [long] - 8 bytes

  TOTAL: 16 bytes

WorkingDay:

- id [long] - 8 bytes

- date [Date] - 8 bytes
- workingCalendarId [long] - 8 bytes

  TOTAL: 24 bytes

Company:

- id [long] - 8 bytes
- name [String] - 10* bytes
- address [String] - 40 bytes
- averageRating [double] - 8 bytes
- workingCalendarId [long] - 8 bytes

  TOTAL: 74 bytes

CompanyEquipment:

- id [long] - 8 bytes
- companyId [long] - 8 bytes
- equipmentId [long] - 8 bytes
- quantity [int] - 4 bytes

  TOTAL: 28 bytes

Equipment:

- id [long] - 8 bytes
- name [String] - 20* bytes
- type [String] - 10* bytes
- description [String] - 50* bytes
- price [double] - 8 bytes

TOTAL: 96 bytes

CompanyAdministrator:

- userId [long] - 8 bytes
- companyId [long] - 8 bytes

TOTAL: 16 bytes

RegisteredUser:

- userId [long] - 8 bytes
- penalPoints [int] - 4 bytes

TOTAL: 12 bytes

Role:

- id [long] - 8 bytes
- name [String] - 15* bytes

TOTAL: 23 bytes

SystemAdministrator:

- passwordChanged [boolean] - 1 bit
- userId [long] - 8 bytes

TOTAL: 9 bytes

User:

- id [long] - 8 bytes
- username [String] - 30* bytes

- password [String] - 8 bytes
- name [String] - 10* bytes
- surname [String] - 15* bit
- city [String] - 10* bytes
- country [String] - 15* bytes
- phoneNumber [String] - 13* bytes
- occupation [String] - 15* bytes
- companyInfo [String] - 10* bytes
- enabled [boolean] - 1 bit
- lastPasswordResetDate [Timestamp] - 8 bytes

  TOTAL: 143 bytes

ESTIMACIJA:

- Appointment 500 hiljada na mjesečnom nivou, godišnje 6 miliona
- QRcode rezervaciju pravi 50% registrovanih korisnika, 25 miliona
- QREquipment svaki od njih prosječno rezerviše dvije vrste opreme, 50 miliona
- WorkingCalendar obzirom na broj kompanija, sveukupno 15 miliona
- WorkingDay 22 prosječno u mjesecu, oko 250 godišnje, 3 750 000 000
- Company obzirom na broj korisnika, sveukupno 15 miliona
- CompanyEquipment svaka kompanija ima ponudu, sveukupno 200 miliona
- Equipment obzirom na broj korisnika, 100 miliona
- CompanyAdministrator 30% korisnika, 30 miliona
- RegisteredUser 50% korisnika, 50 miliona
- Role 3 moguće uloge
- SystemAdministrator 20% korisnika, 20 miliona

- User vrlo korišćeno, ukupan broj korisnika je 100 miliona

KONAČNO:

- Appointment 6 000 000 * 61 bytes * 5 = 1 800 000 000 = 1.8 GB
- QRcode 25 000 000 * 34 bytes = 850 000 000 = 0.85 GB
- QREquipment 50 000 000 * 28 bytes = 1 400 000 000 = 1.4 GB
- WorkingCalendar 15 000 000 * 16 bytes = 240 000 000 = 0.24 GB
- WorkingDay 3 750 000 000 * 24 bytes * 5= 450 000 000 000= 450 GB
- Company 15 000 000 * 74 bytes = 1 110 000 000 = 1.11 GB
- CompanyEquipment 200 000 000 * 28 bytes = 5 600 000 000 = 5.6 GB
- Equipment 100 000 000 * 96 bytes * 5 = 48 000 000 000= 48 GB
- CompanyAdministrator 30 000 000 * 16 = 480 000 000 = 0.48 GB
- RegisteredUser 50 000 000 * 12 bytes = 600 000 000 = 0.6 GB
- Role 3 * 23 bytes = 69 bytes
- SystemAdministrator 20 000 000 * 9 bytes = 180 000 000 = 0.18 GB
- User 100 000 000 * 143 bytes = 14 300 000 000 = 14.3 GB

TOTAL: približno 524 GB

6. Predlog strategije za postavljanje load balansera

Imajući u vidu cilj da naša aplikacija bude visoko dostupna i skalabilna, neophodno je da ispravno postavimo i strategiju load balansera. Time bismo poboljšali i horizontalno proširili našu aplikaciju. Vertikalno širenje je uvijek izvodljivo adekvatnom nadogradnjom hardvera, memorije odnosno jačeg procesora. Horizontalno širenje, koje ne bi narušilo stabilnost naše aplikacije, bi podrazumijevalo postepeno uvođenje dodatnih servera. Shodno tome, važnost izabranog algoritma load balansera dolazi do izražaja. Mi bismo se opredijelili za round-robin algoritam, koji bi ravnomjerno raspoređivao zahtjeve ka svim serverima, te nijednom ne bi došlo do momenta da je neki server nedovoljno korišćen, dok je jedan preopterećen. Takođe, round-robin algoritam bi ubrzao rad naše aplikacije, ali bi i pored toga riješio još jedan potencijalni problem, a to je otkazivanje servera. Naime, load balanser sa round-robin algoritmom bi omogućio nesmetano i neprekidno funkcionisanje naše aplikacije u tom slučaju, što bi se odrazilo na opšti utisak korisnika.

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

Da bismo dijagnostifikovali sposobnosti naše aplikacije koje imaju prostora za unapređivanje, trebamo sprovesti detaljno praćenje i analizu korisničkih zahtjeva. U tome nam može pomoći koncept monotoringa, gdje bismo, računajući različite vrste metrika, poput aktivnosti sistema ili brzine odgovora na zahtjeve, detektovali nedostatke koji su se pojavili u međuvremenu. To bi bio preduslov za kontinuirano i kvalitetno funkcionisanje naše aplikacije. Uzimajući u obzir težnju ka proširenju upotrebe naše aplikacije od strane korisnika, računanje takvih metrika bi bio siguran korak ka poboljšanju korisničkih usluga. U našem primjeru, ideja da se u aplikaciju implementira pametni sistem koji korisniku nudi opremu koja je najčešće pretraživana, ili kompanije koje su najčešće rezultat pretrage, apsolutno ima utemeljenje. Takođe, imalo bi smisla upustiti se u analizu statistike termina, zarad informacije o tome koji termini su najposjećeniji i koji stvaraju najveću gužvu i zadržavanje pri realizaciji. Bilo bi interesantno razmotriti mogućnost prikazivanja kvaliteta opreme u nekim kompanijama na osnovu ocjena registrovanih korisnika. Na kraju, naš sistem ima ugrađenu Prometheus datoteku, koja prikuplja osnovne informacije o aplikaciji, poput zauzeća memorije ili iskorišćenosti procesora; dok je za grafički prikaz zadužena Grafana.

8. Kompletan crtež dizajna predložene arhitekture