Clinical System PoC

Članovi:

Petaković Aleksandar, Perendija Mihajlo, Bašić Petar

Zahtevi:

Implementirati funkcionalni informacioni sistem kliničkog centra.

Trenutno stanje:

Uspešno izgrađena aplikacija za opsluživanje manjeg broja korisnika. Korišćene tehnologije:

- Java
- Spring Boot
- Angular

Trenutni ciljevi:

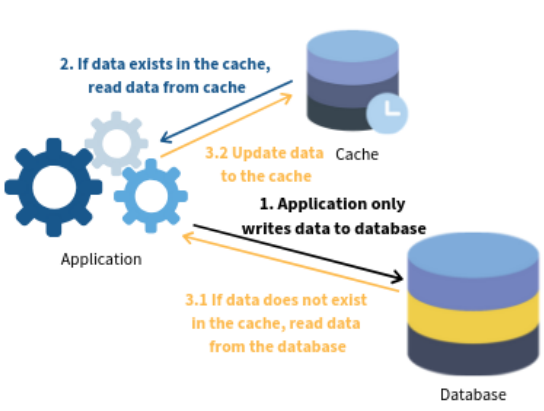
Nakon stečenog iskustva sa izradom web aplikacije za mali broj korisnika, i istraživanja korišćenih tehnologija, uvidjeni su načini na koje bi se aplikacija mogla unaprediti kako bi opsluživala veći broj korisnika:

1. Keširanje

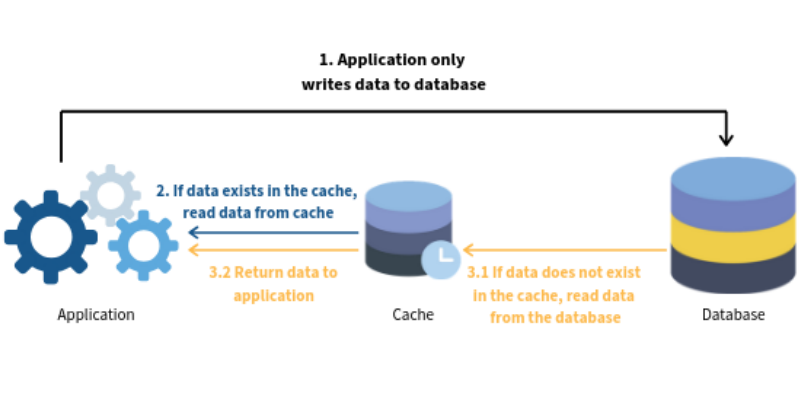
Potreba za skladištenjem podataka u cache memoriji javlja se kod aplikacija sa velikim brojem podataka i velikim protokom tih podataka. Za aplikaciju kao što je klinički sistem, bilo bi pogodno keširati podatke koji se redje menjaju, i čuvati ih do njihove sledeće promene. Tu spadaju lični podaci korisnika, podaci o klinikama, lekovima, dijagnozama, ordinacijama i tipovima pregleda. Podatke koji se često menjaju, kao što su pregledi, operacije, zauzeće doktora, nije pogodno čuvati u cache memoriji.

Korišćenje eksternih usluga, kao što su lokacijski servisi, može dovesti do zakrčenja tokom perioda visokih skokova u opterećenju sistema. Iz ovog razloga, pogodno bi bilo keširati i dobijene podatke ovih servisa. U slučaju naše aplikacije, ovo su zahtevi za lokacijama i prikazom klinika na mapi.

Pogodna strategija za keširanje bi bila “write-around”:



Write around + Cache aside

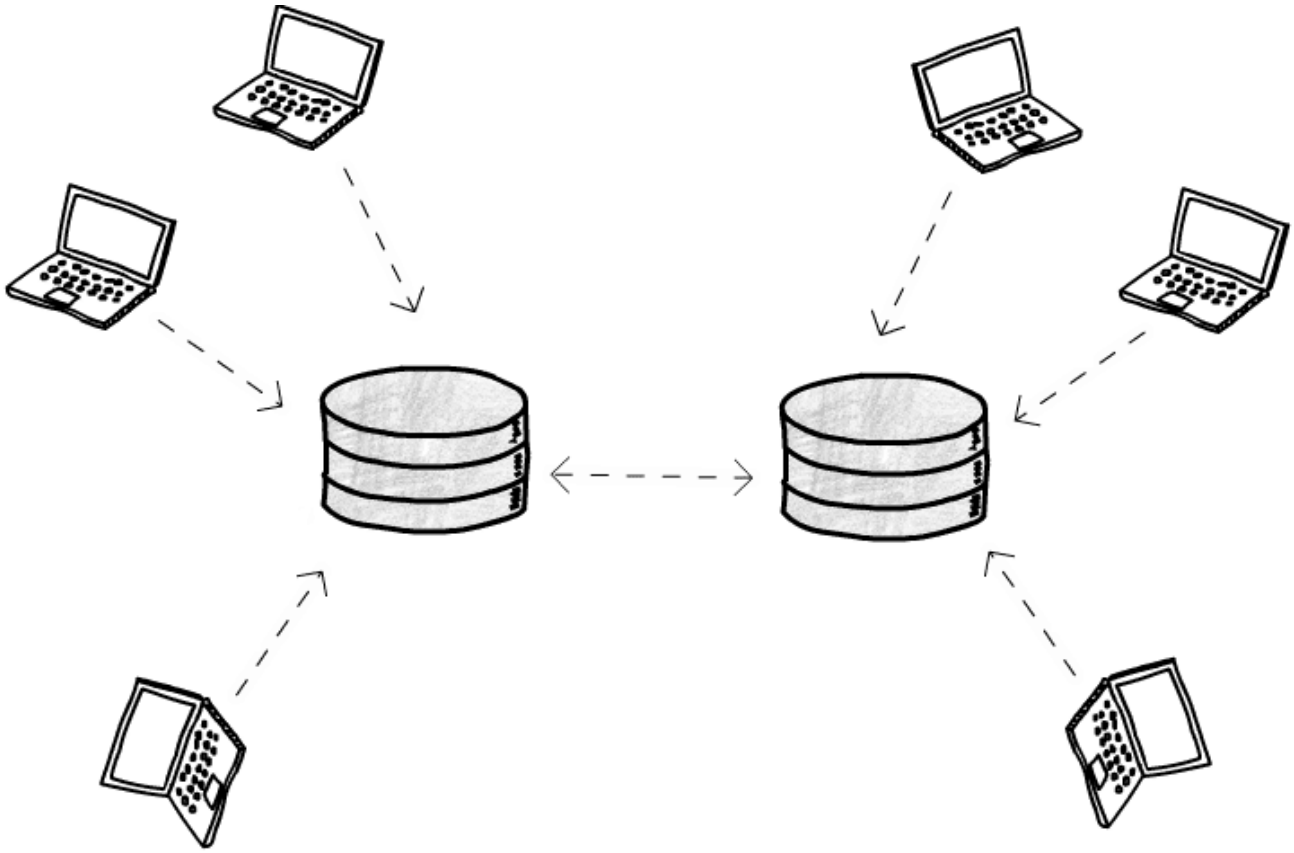


Write around + Read through

Write around se kombinuje najčešće sa “cache-aside” i “read-through” strategijama. Čitanje podataka se vrši kao u navedenim strategijama, medjutim aplikacija ima mogućnost da direktno zapisuje podatke u bazu.

2. Replikacija baze podataka

Usled velikog broja korisnika sa različitih lokacija i velike količine podataka kojima oni pristupaju, pogodno je primeniti replikaciju baze. Najpogodnije rešenje jeste rasporediti baze podataka po gustini korisnika čime bi svaki korisnik pristupao podacima geografski najbliže baze podataka. Strategija koja bi bila pogodna jeste kreirati po jednu bazu podataka za svaki region koji sadrži nekoliko desetina klinika(broj zavisi od potencijalnog broja korisnika i medjusobne geografske udaljenosti klinika):



Ovakav način replikacije bi podrazumevao čuvanje podataka na mnoštvu geografski balansirano postavljenih centara koji bi medjusobno po potrebi komunicirali. Ovi centri bi mogli biti potpuno ekvivalentni, ali bi mogla biti izvršena i fragmentacija relacija, kao na primer horizontalna fragmentacija relacije koja povezuje klinike i pacijente. Na ovaj način bi se podaci o klinikama i pacijentima jednog regiona mogli čuvati samo u tom regionu.

Komunikacija izmedju centara bi bila transakciona, odnosno pri svakom pisanju u lokalnu repliku, vršila bi se sinhronizacija podataka sa ostalim.

Ovakav sistem bi zadovoljavao sledeće uslove:

- svaka lokalna baza podataka je autonomna
- svi čvorovi ovog sistema su ravnopravni
- korisnik nema uvid o tome kako je distribucija podataka fizički izvedena

Prednosti replikacije baza podataka obuhvataju:

- veću dostupnost podataka
- veću sigurnost podataka
- bolje performanse

3. Particionisanje podataka

U slučaju naše aplikacije, smatramo da bi horizontalno particionisanje najviše doprinelo performansama. Konkretno ovakvo particionisanje bi bilo moguće primeniti na relacijama koje vode evidenciju o pregledima i operacijama(na osnovu klinike), pacijentima(na osnovu pridodate hash vrednosti), vezama pacijenata i klinike(na osnovu klinike).

U slučaju proširenja modela dodatnim poljima, moglo bi se razmatrati vertikalno particionisanje po poljima koja se ređe koriste. Tačnije, u slučaju dodavanja nekog multimedijalnog sadržaja, on bi se čuvao odvojeno od ostatka entiteta na koji se odnosi.

4. Load balancing

Kao “network load balancing” strategiju najbolje bi bilo primeniti “Agent-Based Adaptive Load Balancing”. Svaki server iz klastera bi slao informacije o trenutnom opterećenju na osnovu kojih bi load balancer odlučio kojem serveru će potom poslati zahtev. Uz ovu tehniku bilo bi požejno primeniti metodu “Performance Traffic Routing” koja bi dodatno distribuirala zahteve prema geografski najbližem čvoru. Ovakva kombinacija bi uzela u obzir ujedno i geografsku udaljenost i opterećenost servera.

5. Unapredjenje na osnovu nadgledanja korišćenja

U cilju poboljšanja i prilagodjavanja sistema potrebama korisnika, potrebno je nadgledati odredjene komponente sistema sa kojima korisnik najčešće radi. U slučaju naše aplikacije, pretpostavka je da najviše zahteva upućuju pacijenti, lekari i administratori klinika.

Ako posmatramo aktivnosti pacijenta, treba obratiti pažnju na način zakazivanja pregleda, odnosno kojom putanjom najčešće dolazi do tog cilja. Moguće putanje obuhvataju pretragu klinika, koja nije obavezna, a kojoj zatim sledi obavezna pretraga doktora koja se u slučaju izvršene pretrage klinika

obavlja automatski, u suprotnom manuelno. U skladu sa rezultatima posmatranja, potrebno je poboljšati iskustvo korisnika: unapredjenjem algoritama pretraga klinika i doktora, kao i njihovih dostupnosti i izmenom korisničkog interfejsa.

Potrebno je pratiti trajanje svakog pojedinačnog tipa pregleda i na osnovu toga davati preporuku za zadavanje trajanja prilikom izbora dužine trajanja konkretnog pregleda.

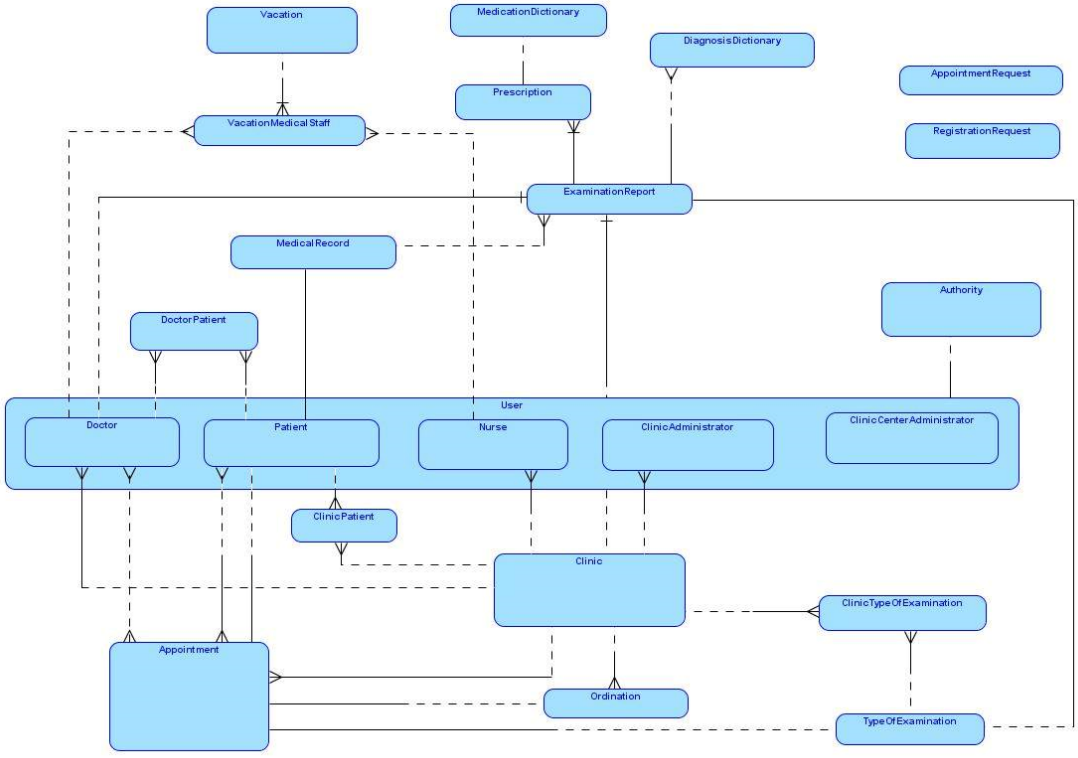
Radi moguće reorganizacije i izmene korisničkog interfjesa prilikom rada lekara poželjno je pratiti njegov način rada sa kalendarom.

Ukoliko se ustanovi da pretraga dostupnosti lekara i sala predstavlja usko grlo, potrebno je optimizovati i te algoritme.

U cilju smanjenja protoka nepotrebnih podataka, potrebno je pratiti posećenost odredjenih komponenti i iskorišćenost podataka koji su tim putem dobavljeni i u skladu sa tim reorganizovati korisničke stranice.

6. Dizajn šeme baze podataka

Na narednoj slici je primer konceptualnog dizajna šeme baze podataka na kojem je zasnovana trenutna verzija aplikacije:



7. Procena hardverskih resursa

Table	Bytes/row	No. of rows in 5yrs	Total size [GB]	
Authority	~25	5	~0	
Clinic	~360	~700	~0.0000001	
TypeOfExamination	~75	~500	~0.0000375	
ClinicTypeOfExamination	~30	~210000	~0.0063	
Medical_record	~137	~200,000,000	~27.5	
Clinic_administrator	~386	~25000	~0.0095	1 per 8.5K users
Clinic_center_administrator	~386	~2500	~00097	
Doctor	~450	~600,000	~0.27	Based on 3 doctors per 1000 patients
Nurse	~430	~1,300,000	~0.6	~X2
Patient	~460	~198,000,000	~91.08	
User_authority	~16	~200,000,000	~3.2	
Medication_dictionary	~48	~15000	~0.00072	
Diagnosis dictionary	~170	~15000	~0.00255	Diagnosis register of R. Serbia
Examination_report	~700	~60,000,000	~42	Based on ~326 letters per examination desc.
Ordination	~56	~140,000	~0.00784	
Appointment	~86	~60,000,000	~5.16	
Appointment_request	~73	~50,000	~0.00365	
Patients_pending_appointments	~16	~500,000	~0.008	
Patients_finished_appointments	~16	~60,000,000	~0.96	
Appointment_doctors	~16	~120,000,000	~1.92	
Clinic_patient	~21	~60,000,000	~1.26	
Doctor_patient	~21	~60,000,000	~1.26	
Prescription	~41	~120,000,000	~4.92	
Vacation	~41	~10,000,000	~0.41	
Registration_request	~451	~100,000	~0.045	When approved becomes patient
Total			**~181.973**	

8. Dizajn predložene arhitekture