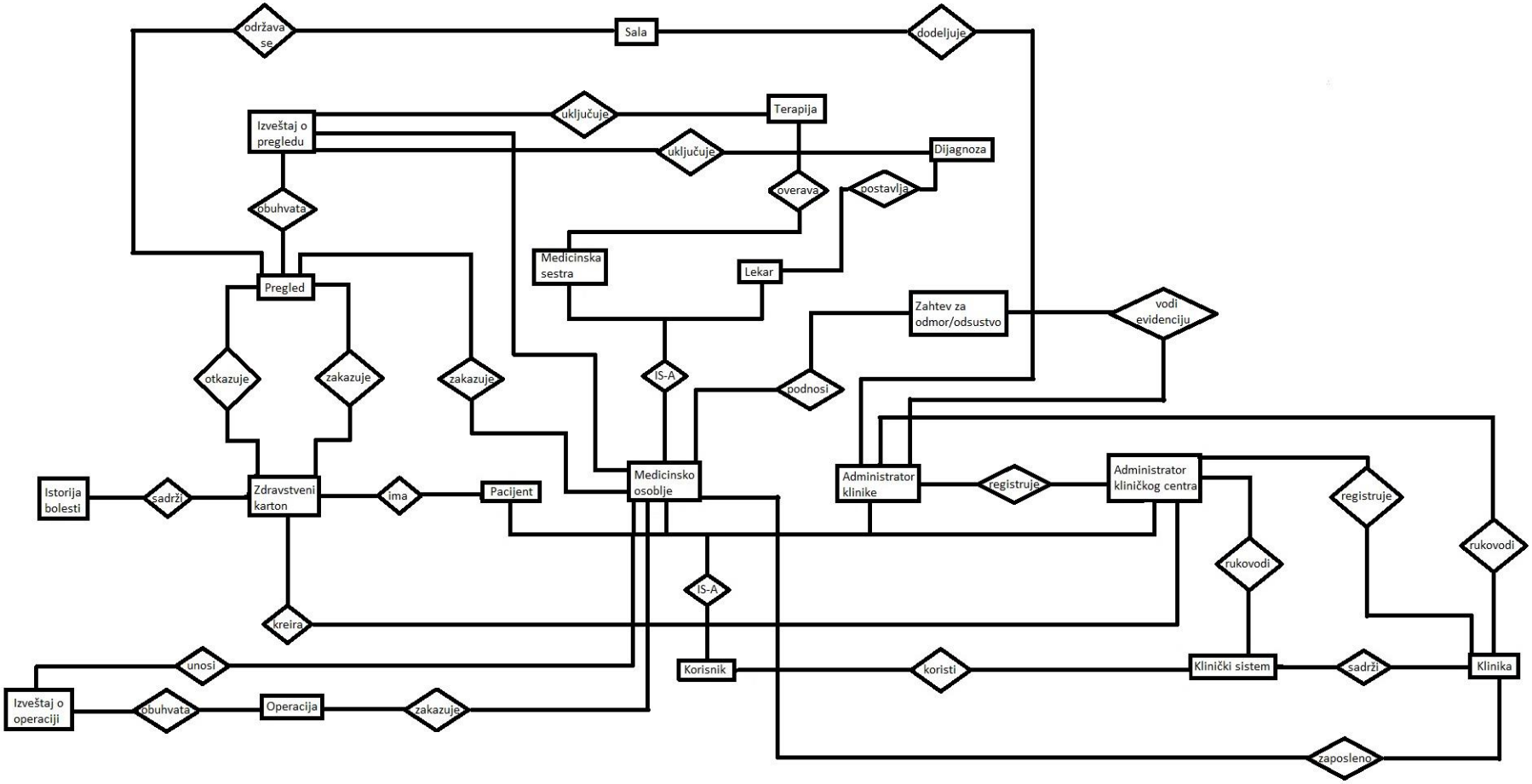
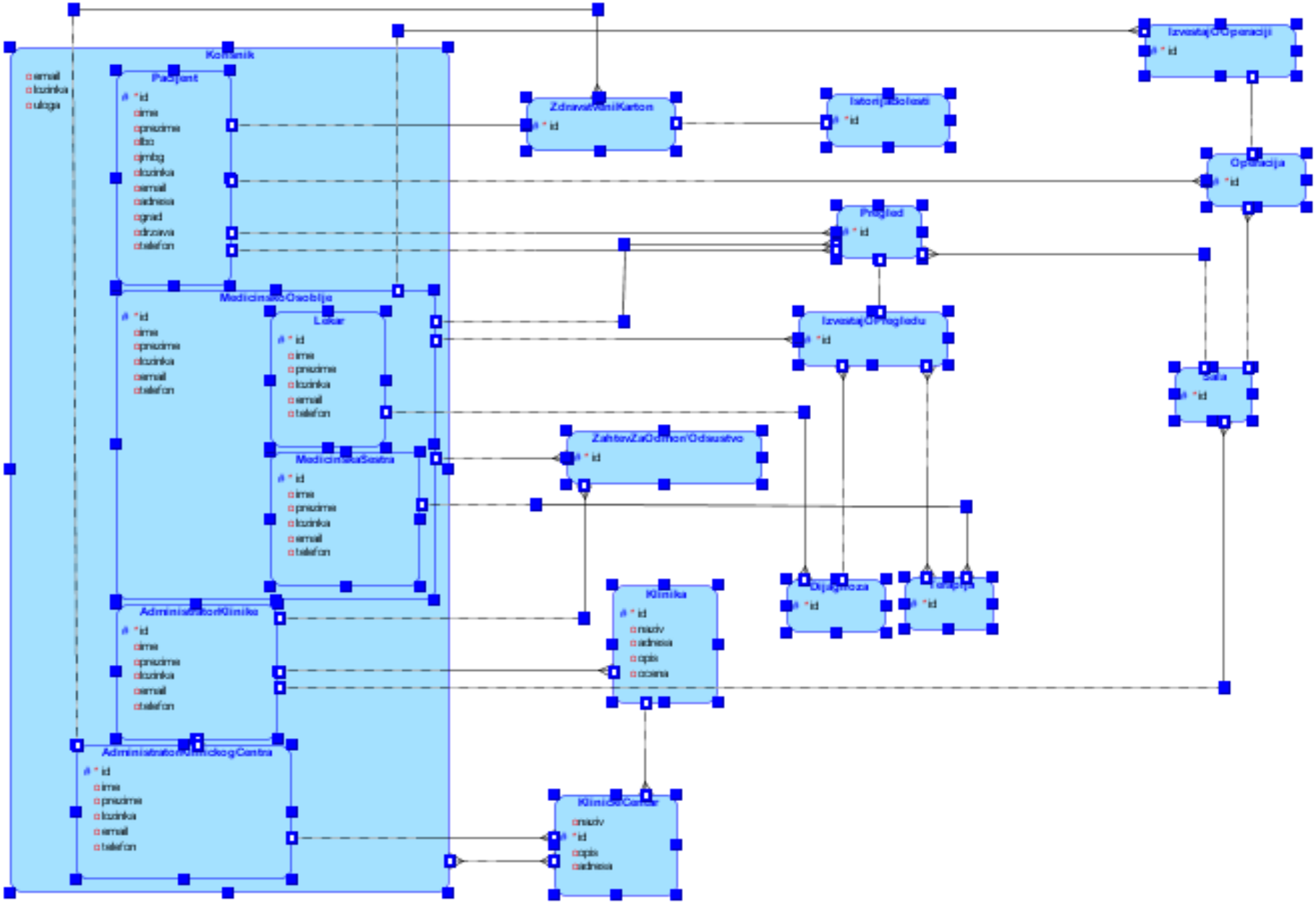
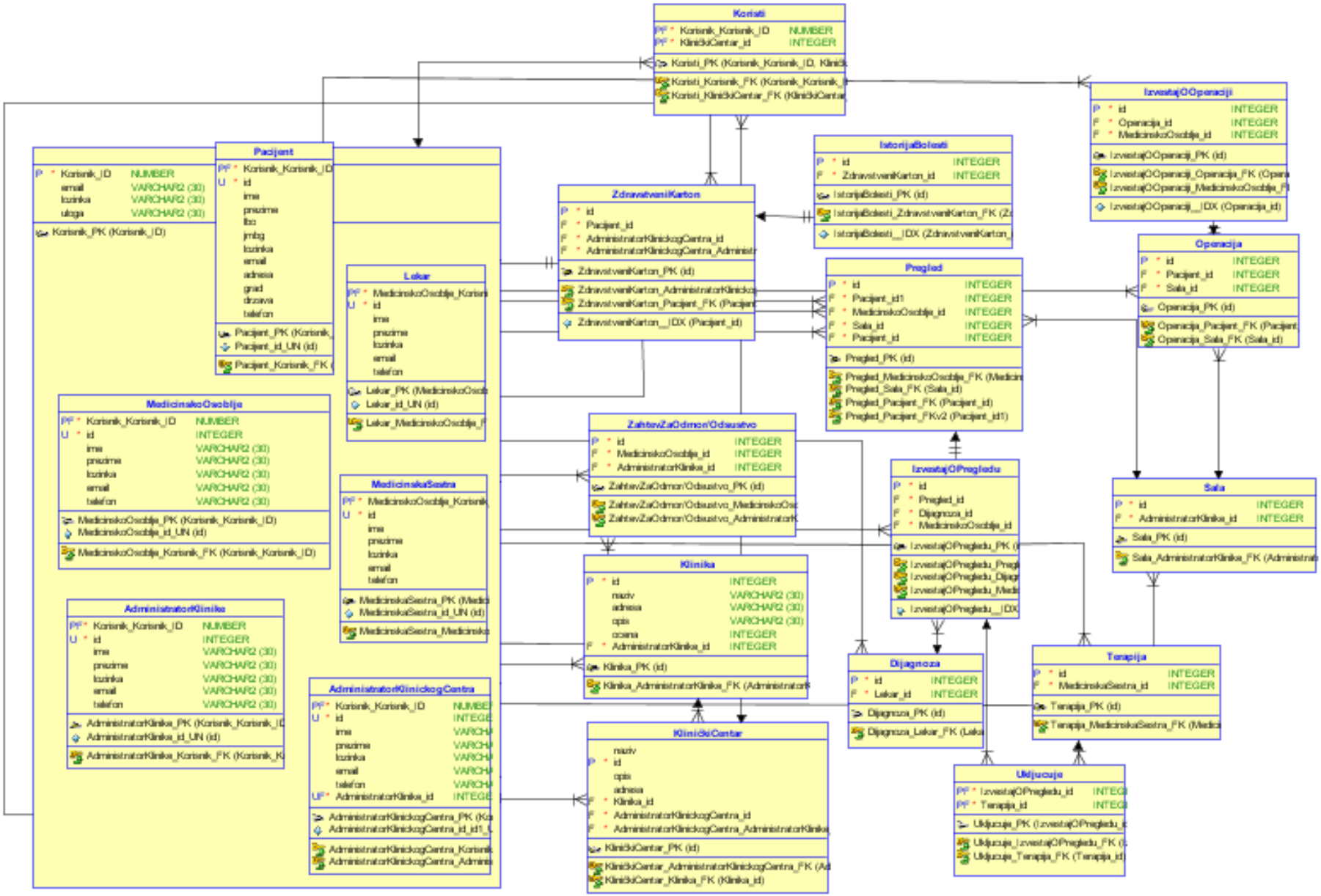
PROOF OF CONCEPT

DIZAJN ŠEME BAZE PODATAKA




Korisnik
email
lozinka
uloga
Pacijent
id
ime
prezime
dlbo
mbg
lozinka
email
adresa
spol
drzava
telefon
ZdravstveniKarton
id
IstorijaBolesti
id
IzvestajOOperaciji
id
Operacija
id
Pregled
id
MedicinskoOsoblje
id
ime
prezime
lozinka
email
telefon
Lekar
id
ime
prezime
lozinka
email
telefon
IzvestajOPregledu
id
Soba
id
MedicinskaSestra
id
ime
prezime
lozinka
email
telefon
ZahtevZaOdmorIOdsustvo
id
AdministratorKlinike
id
ime
prezime
lozinka
email
telefon
Klinika
id
naziv
adresa
opis
ocena
Dijagnoza
id
Terapija
id
AdministratorKlinickogCentra
id
ime
prezime
lozinka
email
telefon
KlinickiCentar
naziv
id
opis
adresa


Koristi
Pacijent
IstorijaBolesti
ZdravstveniKarton
IzvestajOOperaciji
Operacija
Pregled
Lekar
MedicinskaSestra
MedicinskoOsoblje
ZahtevZaOdmon/Odsustvo
IzvestajOPregledu
Sala
Klinika
AdministratorKlinike
AdministratorKlinickogCentra
KlinickiCentar
Dijagnoza
Terapija
Uldjucuje

PREDLOG STRATEGIJE ZA PARTICIONISANJE PODATAKA

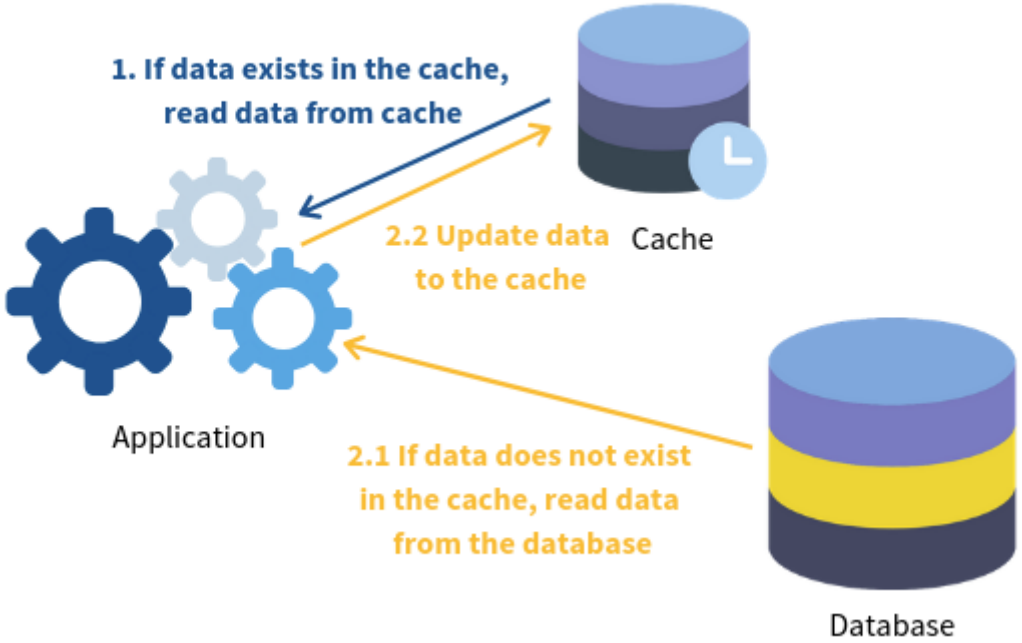
Kako zadati informacioni sistem zahterad sa velikim brojem podataka na dnevnom nivou, neophodno je particionisati podatke na više mašina. Na ovaj način možemo pojednostaviti, a samim tim i ubrzati operacije čitanja i pisanja. S obzirom da svaka od strategija ima svoje prednosti i mane, uobičajeno je da se izabere ona strategija koja donosi najmanje štete, a pritom podržava one stavke za koje smo procenili da su nam bitnije u odnosu na ostale. Zbog takvih i razloga sličnih njima, u ovom slučaju je izabrana strategija particionisanja podataka kombinacijom particionisanja na osnovu identifikatora pregleda i vremena zakazivanja istog. Ovom kombinacijom možemo da iskoristimo prednost particionisanja podataka na osnovu pregleda, za koji postoji funkcija koja će mapirati jedinstveni identifikator pregleda na proizvoljan server gde će se pregled čuvati. Pacticionisanje na osnovu vremena kreiranja pregleda ima za benefit brže preuzimanje pregleda koji su zakazani ili izvršeni u skorije vreme. U tom slučaju dovoljno je znati na kom se serveru (serverima) čuvaju takvi pregledi. Manu ovog pristupa (sveži pregledi idu na jedan server koji postaje najviše opterećen prilikom čitanja) rešava prethodno navedena pozitivna osobina particionisanja na osnovu identifikatora pregleda, dok pretraživanje svega nekoliko servera koji čuvaju najnovije preglede smanjuje vreme koje bi bilo potrebno za pretragu svih particija da bi se našao određeni pregled. Pored vrednosti koja predstavlja identifikator pregleda čuvaće se i vreme izraženo u sekundama.

PREDLOG STRATEGIJE ZA REPLIKACIJU BAZE I OBEZBEĐIVANJE OTPORNOSTI NA GREŠKE

Glavni razlog zbog kog se odlučuje za replikaciju baze je ubrzanje transakcija i upita. Takođe, ovaj korak može biti od koristi i prilikom testiranja programa. Ukoliko se odlučimo za replikaciju baze, moramo osigurati da sve promene izvršene nad jednom lokacijom budu automatski reflektovane na svim ostalim lokacijama. Iako obezbeđivanje konzistentnosti zahteva ponekad kompleksne metode, ona je neophodna. Mogućnost kompletnog povraćaja baze podataka se obezbeđuje u slučaju da primarni server nije dostupan. Rezultat je distribuirana baza podataka kojoj korisnici mogu da pristupaju shodno njihovim zahtevima, a da pritom ne ometaju rad ostalih. Kako je program najčešće namenjen čitanju podataka, može se i samo ovaj fragmenat replicirati. Broj kopija može da varira u zavisnosti od bitnosti podataka. Prethodno bi povećalo dostupnost baze podataka i njenu zaštitu. Kada bi se izvršila potpuna replikacija baze, teže bi se moglo rukovati konkurentnošću, a i sam proces ažuriranja podatka bi bio sporiji. U slučaju da primarni server nije dostupan, može se izabrati neki od sekundarnih, koji bi postao primaran.

PREDLOG STRATEGIJE ZA KEŠIRANJE PODATAKA

Mehanizam keširanja podataka se koristi u svrhu postizanja visokog nivoa skalabilnosti i performansi. Kešitanje može unaprediti dostupnost podataka, čak i kad je *backend* nedostupan. U troslojnoj arhitekturi, koja se koristi u svrhu obezbeđivanja fleksibilnih i sistema za višekratnu upotrebu, može se desiti da su slojevi na različitim hostovima. U tom slučaju, brzina je ograničena performansama same mreže, a vreme odziva svakako igra ključnu ulogu pri oformljavanju korisničkog iskustva. Samo keširanje podatak podrazumeva skladištenje često korišćenih podataka u privremenu memoriju. Od nekoliko strategija koje postoje za keširanje podataka, *Cache Aside* bi bila strategija koja bi se najbolje uklopila u projekat. Uprkos tome što ponekad može da se javi nekonzistentnost između keša i baze podataka, ova strategija je prilično otporna na greške koje se javljaju prilikom pada baze podataka. Uspešno pokriva i slučaj kada je model u kešu i u bazi različit, i uprkos tome što se javlja kao rezultat *cache miss* prilikom preuzimanja podataka po prvi put, jedna je od najboljih strategija za čitanje podataka generalno. Korišćenjem ovog postupka možemo da obezbedimo keširanje podataka najčešće korišćenih podataka od strane najaktivnijih korisnika. Dodavanjem prethodno nave



PREDLOG STRATEGIJE ZA POSTAVLJANJE LOAD BALANCERA

Load balance-eri služe za preusmeravanje zahteva na servere. Oni vrše rutiranje zahteva od strane korisnika ka određenim serverima. Služe za distribuciju saobraćaja ka mrežama, efikasno usmeravajući zahteve ka serverima. Visoka dostupnost se obezbeđuje time što se zahtevi upućuju onim serverima koji su dostupni i aktivni. Takođe, jedna od većih prednosti korišćenja *Load balancera* je jednostavnost u dodavanju i uklanjanju servera u mreži zasnovanih na ličnim zahtevima kompanije na primer. Kada se govori o load balanserima, najčešće su korišćeni algoritmi kao što je *Round Robin.* Korišćenje ove strategije, zahtevi se dodeljuju sekvencijalno, tačnije, svaki server dobija novi zahtev svaki put. To znači da ako je *server1* prihvatio zahtev prvi, sledeći zahtev će biti redirektovan na *server2*. Pored ove, postoji i strategija koja zahtev redirektuje na onaj server koji ima najmanje zahteva. U ovom slučaju *load balancer* mora da poseduje informacije o zauzetosti servera, pa bi primenjivanje ove strategije zahtevalo dodatnu kalkulaciju, a samim tim i više vremena. U slučaju aplikacije *Kliničko centra*, bilo bi korisno da se uvedu *load balanceri* u strukturu, što bi dodatno poboljšalo performanse aplikacije.

PREDLOG KOJE OPERACIJE KORISNIKA TREBA NADGLEDATI U CILJU POBOLJŠANJA SISTEMA

Situacije koje bi bilo poželjno nadgledati su svakako one koje su potencijalne konfliktne situacije. Na ovaj način sistem bi obezbedio svoju konzistentnost. Kako se u specifikaciji ovog projekta može uočiti nekoliko mogućih nepoželjnih situacija na primeru zakazivanja pregleda i sale za taj pregled, to su operacije koje zahtevaju dodatni oprez i nadgledanje.

CRTEŽ DIZAJNA PREDLOŽENE ARHITEKTURE