INFORMACIONI SISTEM CENTRA ZA TRANSFUZIJU KRVI

Proof of Concept Report

Januar 2023

Sažetak dokumenta

Stavka	Trenurna vrednost
Naslov dokumenta	Proof of Concept Report
Poslednja izmena	26-Jan-2023
Status	Završeno
Opis dokumenta	Ovaj dokument sadrži tehničke detalje vezane za projekat radjen na predmetu "Internet softverske arhitekture"

Učesnici

Ime	Prezime
Darko	Selaković
Rade	Stojanović
Marko	Uljarević
Vojin	Bjelica

SADRŽAJ

Logički dizajn šeme baze podataka

Logički dizajn baze podataka za informacionog sistema centra za transfuziju krvi uključuje normalizaciju podataka kako bi se uklonila redundantnost podataka i poboljšao integritet podataka. To se može postići deljenjem podataka u zasebne tabele i definisanjem relacija među njima.

U slučaju našeg informacionog sistema, glavne celine identifikovane u projektu mogu se razdvojiti u različite tabele. Na primer, tabela "Appointment" sadrži podatke kao što su vreme početka i kraja pregleda, podatak o pacijentu kojem je zakazan termin i slično.

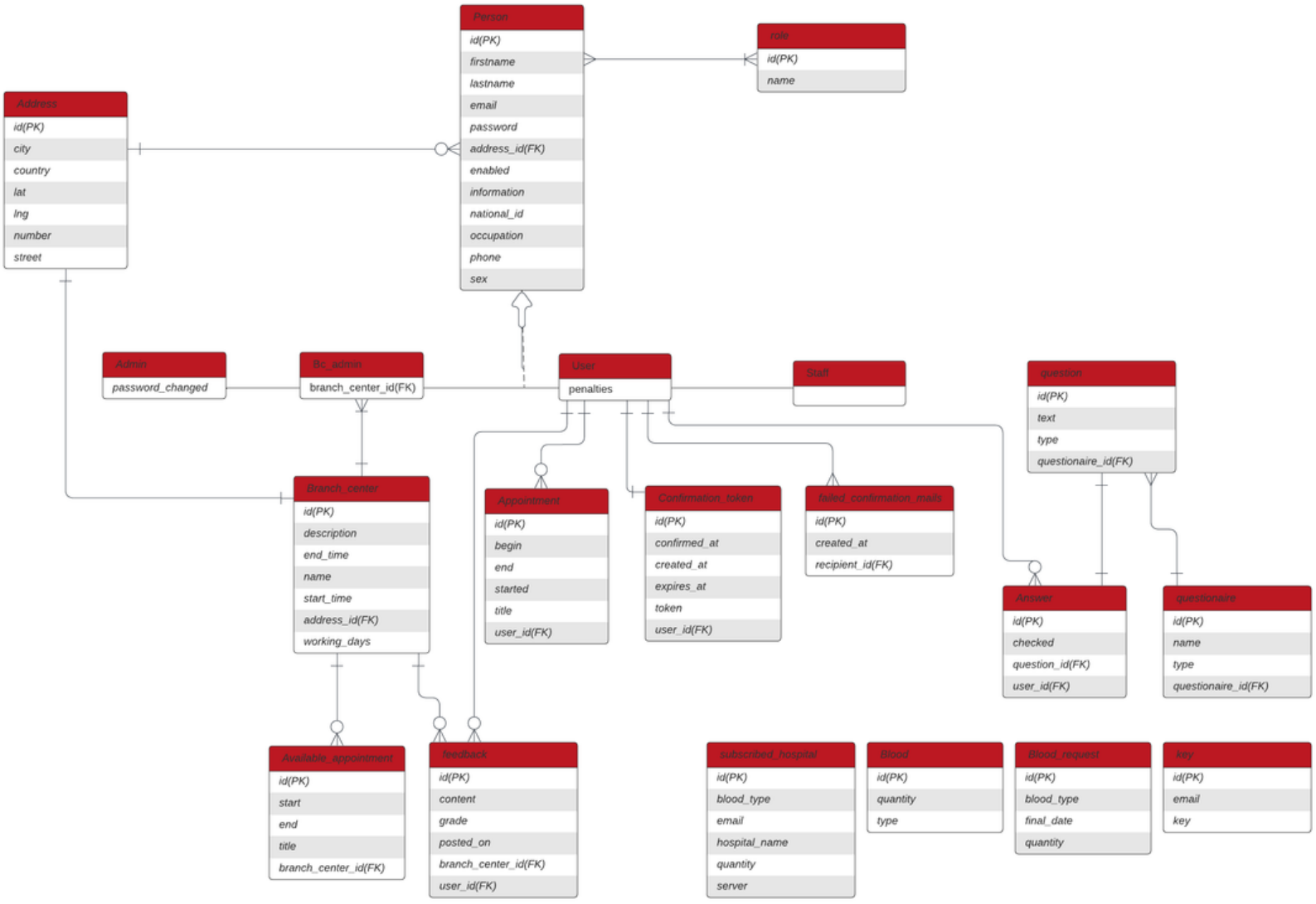
Svaka tabela ima primarni ključ, jedinstveni identifikator za svaki zapis.

Da bi se održali odnosi između entiteta onako kako su definisani u konceptualnom dizajnu, koriste se strani ključevi. Na primer, u tabeli "Appointment", strani ključ bio bi identifikacioni broj pacijenta, povezujući ga sa tabelom u kojoj se nalaze pacijenti i slično.

Uz tabele i odnose koji su navedeni u prethodnom primeru, postoji nekoliko drugih važnih aspekata koje treba uzeti u obzir prilikom dizajniranja logičke baze podataka za naš informacioni sistem.

- Vrste podataka: u kolonama treba pažljivo odabrati vrste podatka kako bi se osiguralo da su podaci sačuvani u najprikladnijem formatu.
- Ograničenja: koriste se za sprovođenje integriteta podataka. Ograničenja se definišu i na nivou kolona i na nivou same tabele.
- Sigurnost: sigurnost treba uzeti u obzir prilikom dizajniranja baze podataka, kako bi se osiguralo da samo ovlašćeni korisnici mogu pristupiti podacima ili ih mijenjati. Na primer, u tabeli "Person" podatak o šifri korisnika se čuva kao "hash" podatak.

Grafički prikaz dizajna



Opis strategije

Glavni cilj ove implementacije je poboljšanje performansi i skalabilnosti sistema particionisanjem podataka na način koji omogućava brže i učinkovitije pribavljanje podataka.

Jedan od glavnih razloga za implementaciju ove strategije particionisanja je poboljšanje performansi pribavljanja podataka. Particionisanjem podataka možemo lakše pribaviti određene podskupove podataka na osnovu određenih kriterijuma, kao što su datum ili lokacija. Ovo može znatno poboljšati brzinu pribavljanja podataka, posebno za velike skupove podataka.

Drugi razlog za implementaciju ove strategije particionisanja je poboljšanje skalabilnosti. Kako broj korisnika i količina podataka raste, performanse sistema mogu degradirati. Particionisanje podataka može pomoći u ravnomernoj raspodeli opterećenja na više različitih lokacija, što može pomoći u poboljšanju skalabilnosti sistema.

Implementacija ove strategije particionisanja je relativno jednostavna. Prvo bi kreirali novu particionisanu tabelu u *PostgreSQL* bazi podataka, a zatim bi kreirali particije za tabelu na osnovu određenog kriterijuma (na primer datuma). Zatim bi napravili novi *Spring Data JPA* repozitorijum za particionisanu tabelu i servis za izvođenje CRUD operacija na particionisanoj tabeli.

Benefiti

Postoji nekoliko ključnih benefita implementacije particionisanja podataka na našem informacionom sistemu:

- Poboljšane performanse upita**:** particionisanjem podataka možemo lakše dobaviti određene podskupove podataka na osnovu određenih kriterijuma, kao što su datum ili lokacija. Ovo može znatno poboljšati brzinu dobavljanja podataka, posebno za velike skupove podataka. Ovo može biti posebno korisno u našem informacionom sistemu gde se podaci često traže na osnovu specifičnih informacija o pacijentu ili terminu, kao što je datum rođenja ili datum termina.

- Povećana skalabilnost: kako broj korisnika i količina podataka raste, performanse sistema mogu degradirati. Particionisanje podataka može pomoći u ravnomernijoj raspodeli opterećenja na više servera, što može pomoći u poboljšanju skalabilnosti našeg sistema. Ovo može biti posebno važno u našem informacionom sistemu gde broj pacijenata i količina podataka o pacijentima stalno raste.

- Pojednostavljeno upravljanje podacima: particionisanje podataka može olakšati upravljanje i održavanje podataka. Na primer, ako su podaci podeljeni prema datumu, može biti lakše arhivirati ili izbrisati starije podatke. Osim toga, particionisanje podataka može olakšati izvođenje sigurnosnih kopija i oporavak od kvarova.

- Povećana dostupnost: particionisanje podataka može pomoći u povećanju dostupnosti sistema. Ako je sistem podeljen na više servera, može se osigurati da sistem ostane dostupan čak i ako jedan ili više servera prestanu sa radom. Ovo može biti posebno važno u našem informacionom sistemu gde je dostupnost podataka o pacijentima i terminima pregleda jako bitna.

- Bolja sigurnost podataka: particionisanje podataka može pomoći u povećanju sigurnosti sistema. Ako je sistem podeljen na više servera, napadačima može biti teže pristupiti željenim podacima.

Opis strategije

Glavni cilj ove implementacije je poboljšanje performansi i skalabilnosti sistema particionisanjem podataka na način koji omogućava brže i učinkovitije pribavljanje podataka.

Pre implementacije ove strategije jako je bitno odredite koje su tabele ključne za naš informacioni sistem i zahtevaju visoku dostupnost i redundantnost podataka. To može uključivati informacije o pacijentima, zakazivanje termina i podatke o bolnicama sa kojima se sarađuje.

Jako je bitno i odrediti metodu za replikaciju baze podataka. Replikacija *master-slave* je prikladna metoda za naš informacioni sistem. Ova metoda omogućava primarnoj bazi podataka da rukuje operacijama pisanja, a jednoj ili više sekundarnih baza podataka da rukuju operacijama čitanja i osigurava mogućnost prebacivanja na drugu bazu podataka u slučaju kvara primarne baze podataka.

Ukoliko bi se odlučilo da se primeni ova strategija, bilo bi jako važno izvesti detaljna testiranja kako bi osigurali da se podaci pravilno kopiraju i da prebacivanje na drugu bazu podataka radi prema očekivanjima. To bi trebalo uključivati testiranje scenarija prelaska u kvar kao što su planirani i neplanirani prekidi primarne baze podataka.

Takođe je važno i osigurati da se proces replikacije baze podataka odvija sigurno. Bilo bi korisno implementirati sigurnosne mere kao što su enkripcija i kontrola pristupa kako bi se sprečio neovlašćeni pristup.

Sveukupno, ključ uspešne replikacije i obezbeđivanje otpornosti na greške našeg informacionog sistema centra za tranfuziju krvi je identifikovanje najbitnijih poslovnih potreba, odabir ispravne metode replikacije, implementacija i testiranje replikacije, nadziranje i održavanje sistema, ali i imati plan oporavka od neočekivane greške, sa fokusom na sigurnost.

Benefiti

Postoji nekoliko ključnih benefita implementacije replikacije baze podataka i rada na obezbeđivanju otpornosti na greške na našem informacionom sistemu:

- Visoka dostupnost: replikacijom podataka, sekundarna baza podataka može se koristiti za obradu zahteva za čitanje i pružanje usluga u slučaju kvara primarne baze podataka. To osigurava da su važni podaci, kao što su podaci o pacijentima i njihovim terminima u centru, uvek dostupni osoblju i službenim licima centra za tranfuziju krvi.

- Redundantnost podataka: sa replikacijom se dobijaju višestruke kopije istih podataka, čime se osigurava određeni nivo redundantnosti podataka. To znači da ako se jedna kopija podataka ošteti ili izgubi, može se oporaviti iz druge kopije.

- Poboljšane performanse: korišćenjem metode replikacije *master-slave*, operacijama čitanja mogu upravljati sekundarne baze podataka, oslobađajući primarnu bazu od ovog radnog opterećenja i poboljšavajući njene performanse.

- Bolji oporavak od grešaka: sa planom za oporavak od grešaka sistem će biti bolje pripremljen za vraćanje kritičnih baza podataka u slučaju kvara primarne baze podataka, smanjujući uticaj na operacije.

- Bolja sigurnost podataka: Particionisanje podataka može pomoći u povećanju sigurnosti sistema. Ako je sistem podeljen na više podsistema, napadačima može biti teže pristupiti željenim podacima.

- Bolja sigurnost: Implementacijom sigurnosnih mera kao što su enkripcija i kontrola pristupa, osetljivi podaci pacijenata mogu se bolje zaštititi od neovlašćenog pristupa, osiguravajući usklađenost sa propisima kao što je *HIPAA*.

- Bolja skalabilnost: Kako naš sistem bude rastao, strategija replikacije i otpornosti na greške može se skalirati kako bi se zadovoljila povećana potražnja, omogućavajući centru da nastavi pružati visokokvalitetnu uslugu svojim korisnicima.

- Bolje donošenje odluka: sa vrlo dostupnim, tačnim i ažurnim podacima, osoblje može donositi bolje i informisanije odluke, poboljšavajući ukupni kvalitet usluge za korisnike.

V. KEŠIRANJE PODATAKA

Opis strategije

Glavni cilj ove implementacije je poboljšanje performansi sistema čuvanjem podataka kojima se često pristupa na brzo i lako dostupno mesto.

Neophodno je identifikovati kojim skupovima podataka se često pristupa i koji skupovi podataka bi imali koristi od keširanja. To može uključivati podatke o korisnicima, rasporede termina i informacije o bolnicama sa kojima se sarađuje.

Dostupno je nekoliko metoda keširanja podataka, kao što je keširanje u memoriji, keširanje na disku i distribuirano keširanje. Keširanje u memoriji prikladna je metoda za informacioni sistem centra za tranfuziju krvi jer omogućava najbrži pristup podacima. Neophodno je konfigurisati sistem da kešira identifikovane skupove podataka koristeći metodu keširanja u memoriji.

Ukoliko bi se odlučili na ovu strategiju, neophodno je često izvoditi testiranje kako bi se osiguralo da se podaci ispravno skladište u memoriju i da sistem pruža poboljšane performanse. Takođe je neophodno redovno nadziranje procesa keširanja i izvršavanje svih potrebnih prilagođavanja kako bi sistem nesmetano radio, kao i zakazivanje redovnih provera održavanja i ažuriranja.

Kako bi se sprečilo da keš postane prevelik i koristi previše memorije, neophodno je implementirati i proces izbacivanja koji iz keša uklanja podatke koji se najmanje koriste ili istekle podatke.

Kontinualno nadgledanje procesa keširanja i po potrebi implementacija poboljšanja kako bi se osiguralo da sistem ostane brz i učinkovit. Neophodno je pratiti najnovije tehnologije keširanja i najbolje prakse i implementirati ih po potrebi.

Sveukupno, ključ uspešne strategije keširanja podataka za naš sistem je identifikacija ključnih poslovnih potreba, odabir prave metode keširanja, implementacija i testiranje keširanja, nadzor i održavanje sistema, implementacija procesa izbacivanja i fokus na sigurnost. Keširanje može uvelike poboljšati performanse našeg sistema pružanjem bržeg pristupa podacima kojima se često pristupa, što može poboljšati ukupan kvalitet usluge za krajnje korisnike.

Benefiti

Postoji nekoliko ključnih benefita implementacije keširanja podataka na našem informacionom sistemu:

- Poboljšane performanse: Keširanje podataka kojima se često pristupa na brzoj, lako dostupnoj lokaciji može uveliko poboljšati performanse našeg sistema smanjenjem vremena potrebnog za pristup podacima. Ovo može biti posebno važno za vremenski osetljive operacije kao što su hitne tranfuzije i rezultati nekih testova.

- Smanjeno opterećenje servera: keširanjem podataka, naš sistem može smanjiti opterećenje delova sistema kao što su baze podataka, što može pomoći u poboljšanju ukupne brzine sistema.

- Bolja skalabilnost: Kako naš sistem stalno raste, strategija keširanja podataka može se skalirati kako bi se zadovoljila povećana potražnja, omogućavajući centru da nastavi pružati visokokvalitetnu uslugu korisnicima.

- Bolja sigurnost: Implementacijom sigurnosnih mera kao što su enkripcija i kontrola pristupa, osetljivi podaci korisnika mogu se bolje zaštititi od neovlašćenog pristupa, osiguravajući usklađenost sa propisima kao što je *HIPAA*.

- Veća brzina odziva: Keširanje omogućava skladištenje podataka bliže korisniku, smanjujući vreme čekanja na zahtev i poboljšanje ukupnog odziva sistema.

- Ušteda troškova: Smanjenjem opterećenja pozadinskih sistema, keširanje može pomoći u smanjenju troškova povezanih sa održavanjem i skaliranjem tih sistema.

- Povećana propusnost: keširanje može povećati stopu dobavljanja podataka iz sistema, omogućavajući više pacijenata da budu usluženi u isto vrieme, što može pomoći u poboljšanju ukupne učinkovitosti našeg centra.

Opis potreba i komponenata

Koristeći ranije navedene strategije procenjujemo da bi hardver dovoljan za normalno funkcionisanje sistema u narednih 5 godina, morao da ispuni uslove koji slede.

Particionisanje podataka u manje skupove zahtevalo bi veliku količinu memorijskog prostora za čuvanje podataka, u rasponu od nekoliko desetina terabajta pa do par stotina terabajta.

Replikacija baze podataka master-slave strategijom zahtevala bi veliku količinu memorije. Primarni sistem trebao bi biti u stanju da se izbori sa velikim brojem korisnika i velikom stopom promena podataka koja bi iznosila i do 500.000 rezervacija na mesečnom nivou, što bi zahtevalo sistem visokih performansi sa velikom količinom memorije za čuvanje podataka. Sekundarni sistem trebao bi imati slične resurse, ali sa manje prostora za čuvanje podataka. Neka procena je da bi nam trebali serveri sa oko 32GB RAM memorije i nekoliko stotina terabajta prostora za skladištenje podataka u bazu.

Kako bi se osigurala visoka dostupnost i redundantnost podataka, bilo bi potrebno imati redundantne hardverske resurse kao što su dodatni serveri, memorija za skladištenje podataka i sigurnosnih kopija. To bi verojatno uključivalo više slojeva redundantnosti, kao što su redundantni izvori napajanja i višestruke kopije podataka čuvane na različitim lokacijama.

Primer komponenti

Naziv	Komponenta
Procesor	Intel Xeon Gold 6248
RAM	Nekoliko DDR4 sa 32GB
Memorija	Nekoliko SSD diskova sa 20TB memorije
Napajanje	Nekoliko napajanja od 500w snage

Opis strategije

Imajući u vidu da će naš sistem imati veći broj servera, neophodno je izabrati load balanser koji će moći da proširi mrežu i sam saobraćaj na sve dostupne servere.

Znajući da će naš sistem koristiti više od 10 mililona korisnika, došli smo do zaključka da bi za naš sistem odgovarao *HAProxy* load balanser.

HAProxy load balanser nudi mogućnost upotrebe *Round-robin* algoritma, koji bi nam omogućio da podjednako raspodelimo zahteve na sve odgovarajuće i slobodne servere u tom trenutku, pa bi bio veoma dobar izbor. *HAProxy* nam takođe nudi mogućnost korišćenja različitih algoritama, u zavisnosti od potreba u datoj situaciji.

Kako bi se osigurala visoka dostupnost, važno je imati redundantne instance load balansera. To se može postići korištenjem više instanci load balansera u aktivno-pasivnoj konfiguraciji ili čak korišćenjem load balanser klastera.

Load balanser je neophodno i konfigurisati da distribuira promet na odgovarajuće servere na osnovu potreba našeg sistema. To može uključivati konfigurisanje provera stanja sistema, kao i postavljanje pravila za distribuciju prometa saobraćaja. Takođe je neophodno i testirati load balanser u uslovima različitog nivoa saobraćaja na sistemu da bi se uverili da radi kako je očekivano.

Kako se broj korisnika našeg sistema bude povećavao, očekivano je da će doći do situacije kada će biti neophodno uključiti u rad i dodatne instance load balansera.

Važno je i održavati load balansere i servere koji distribuiraju promet sigurnima, stoga je neophodno implementirati sigurnosne mere kao što su *firewall* i otkrivanje/sprečavanje upada.

Benefiti

Postoji nekoliko ključnih benefita implementacije load balansera na našem informacionom sistemu:

- Poboljšane performanse i skalabilnost: load balanseri distribuiraju dolazni promet na više servera, što može poboljšati ukupne performanse i skalabilnost sistema.

- Visoka dostupnost: load balanseri mogu se konfigurisati za automatsko prebacivanje na rezervni server u slučaju kvara, osiguravajući visoku dostupnost sistema.

- Bolja sigurnost: load balanseri mogu se konfigurisati za implementaciju sigurnosnih mera kao što su *SSL termination* i liste kontrole pristupa (*ACL*) za ograničavanje pristupa određenim resursima.

- Napredno usmeravanje: load balanseri nude napredne mogućnosti preusmeravanja kao što je usmeravanje zasnovano na sadržaju, što nam omogućava usmeravanje prometa na temelju sadržaja samog zahteva.

- Provera ispravnosti: load balanseri mogu se konfigurisati za proveru ispravnosti servera na koje distribuišu saobraćaj, osiguravajući da samo zdravi serveri primaju zahteve od korisnika.

- Bolje praćenje: load balanseri pružaju detaljnu statistiku i zapise koji se mogu koristiti za praćenje performansi i statusa sistema.

- Učinkovitije korišćenje resursa: load balanseri mogu ravnomernije rasporediti opterećenje na servere, što može pomoći u sprečavanju preopterećenja bilo kojeg pojedinačnog servera i usporavanje sistema.

- Poboljšana pouzdanost: load balanseri mogu pomoći u održavanju rada sistema čak i ako jedan ili više servera zakaže, automatskim preusmeravanjem saobraćaja na druge servere.

- Bolje korisničko iskustvo: load balanseri mogu osigurati da su korisnici usmereni na server koji najbolje može obraditi njihov zahtev, što može dovesti do kraćeg vremena odziva i boljeg ukupnog korisničkog iskustva.

Opis strategije

Kako bi poboljšali naš sistem, važno je pratiti određene korisničke operacije kako bi lakše identifikovali i rešili sve probleme koji se mogu pojaviti.

Neophodno je pratiti izvršavanje i vreme odziva upita baze podataka kako bi identifikovali spore ili problematične upite. To može pomoći u prepoznavanju i rešavanju bilo kakvih problema povezanih sa performansama ili skalabilnošću baze podataka.

Potrebno je nadzirati izvršavanje i vreme odgovora *API* poziva kako bi identifikovali probleme povezane sa *API*-jem ili servisnim uslugama koje on poziva.

Takođe je jako bitna i provera autentifikacije i autorizacije korisnika. Važno je nadzirati izvršavanje i stopu uspešnosti operacija autentifikacije i autorizacije korisnika kako bi lakše identifikovali probleme povezane sa prijavom korisnika ili sigurnošću.

Neophodno je pratiti i sam angažman korisnika. Treba pratite izmerene podatke o angažmanu korisnika kao što su stopa klikanja, trajanje sesije i stopa napuštanja početne stranice kako bi identifikovali probleme povezane sa korisničkim iskustvom ili angažmanom.

Nadziranje izvršavanja i stopa uspešnosti operacija rezervacije i zakazivanja termina kako bi lakše identifikovali probleme povezane sa sistemom rezervacija ili korisničkim iskustvom.

Jedan od predloga za nadgledanje operacija korisnika u cilju poboljšanja sistema je i praćenje korišćenja resursa. To bi uključivalo praćenje količine memorije, CPU-a i diska koje korisnici zauzimaju, kao i broj zahteva koje korisnici šalju sistemu. Ova informacija može se koristiti za identifikaciju problema sa performansama i optimizaciju sistema.

Mi smo se odlučili da za nadgledanje stanja sistema i operacija korisnika koristimo alat pod nazivom *Prometheus*.

Benefiti

Nadgledanje korisničkih operacija kao deo strategije za poboljšanje našeg sistema može doneti nekoliko prednosti, uključujući:

- Poboljšanja performansi i skalabilnost sistema: praćenjem upita baze podataka, API poziva, korišćenja resursa i drugih operacija, možemo identifikovati i rešiti sve probleme koji mogu uticati na performanse ili skalabilnost sistema.

- Povećana stabilnost i pouzdanost sistema: praćenjem zapisa grešaka i korišćenja resursa, možemo identifikovati i rešiti sve probleme koji mogu uticati na stabilnost i pouzdanost sistema.

- Poboljšana sigurnost: praćenjem postupaka provere autentifikacije korisnika i autorizacije, možemo identifikovati i rešiti sve probleme povezane sa sigurnošću.

- Bolje korisničko iskustvo: praćenjem metrike korisničkog angažmana možemo identifikovati i rešiti sve probleme povezane sa korisničkim iskustvom ili angažmanom.

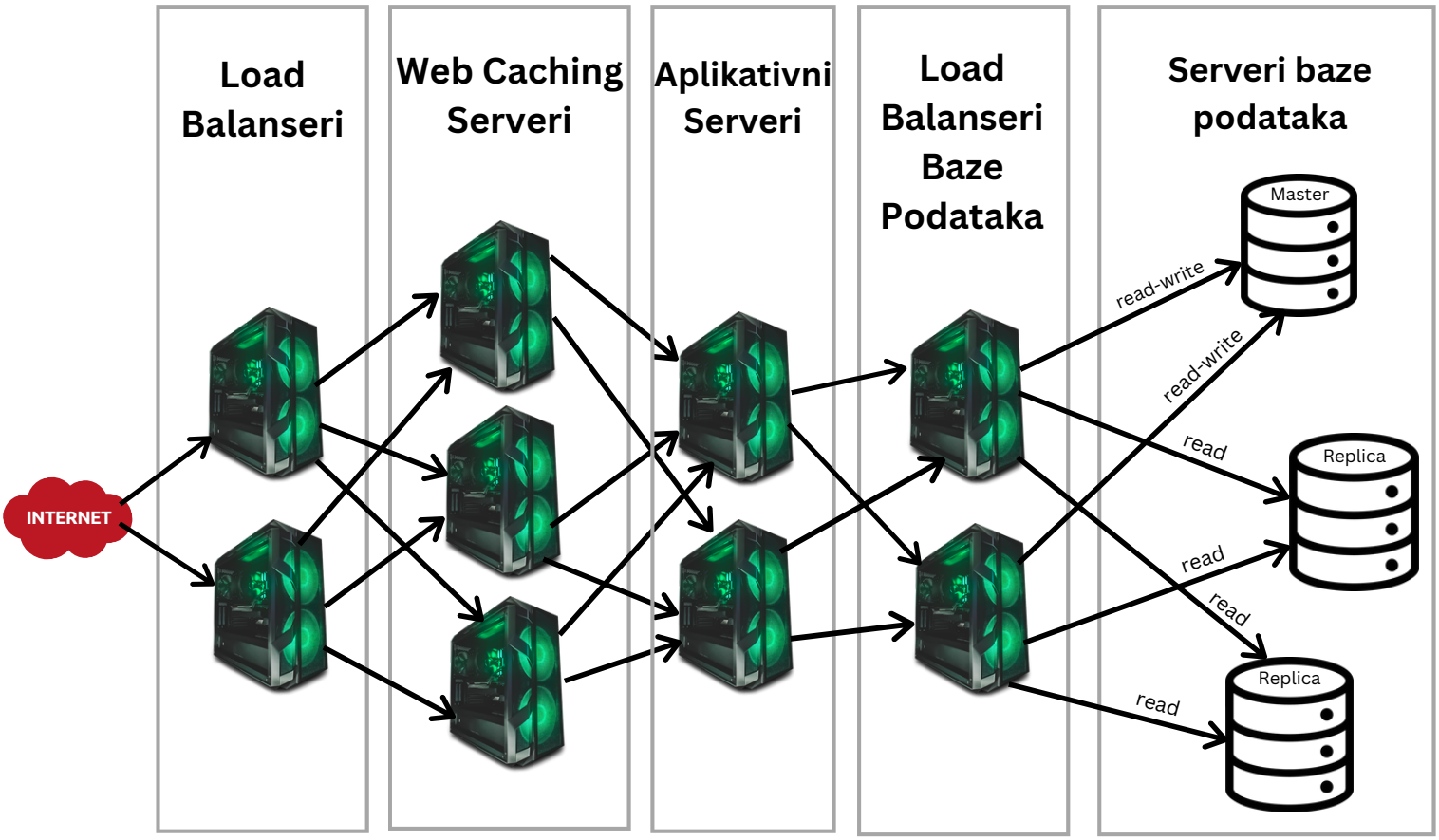
- Bolje praćenje sistema rezervacija i termina: praćenjem performansi i stope uspešnosti operacija rezervacija termina, možemo identifikovati i rešiti sve probleme povezane sa sistemom rezervacija ili korisničkim iskustvom.

- Učinkovitije korišćenje resursa: praćenjem korišćenja resursa, možemo identifikovati i rešiti sve probleme koji se odnose na performanse ili skalabilnost sistema i učinkovitije korišćenje resursa.

- Bolja identifikacija i rešavanje problema: praćenjem različitih operacija, možemo ostvariti više podataka za prepoznavanje problema, kao i za razumijevanje uzroka problema i implementaciju odgovarajućih rešenja.

- Bolje donošenje odluka: praćenjem različitih operacija, možemo ostvariti više podataka za donošenje boljih odluka o sistemu, kao što je dodavanje ili uklanjanje resursa ili unošenje promena u arhitekturu sistema.

Grafički prikaz dizajna



Lista referenci

- C.J. Date(2003). *An Introduction to Database Systems*. London: Pearson
- A. Silberschatz(2010). *Database System Concepts*. New York City: McGraw Hill
- C.J. Date(2012). *Database Design and Relational Theory: Normal Forms and All That Jazz*. California: O'Reilly Media
- M. Tamer Özsu & P. Valduriez(2019). *Principles of Distributed Database Systems*. SpringerLink. Sa https://link.springer.com/book/10.1007/978-3-030-26253-2
- A. Buchmann(2009). *Database Replication Techniques*. dbis.ipd.kit.edu. Sa https://dbis.ipd.kit.edu/download/06a-replication.pdf
- B. Lutkevich(2022). *Database Replication*. TechTarget. Sa https://www.techtarget.com/searchdatamanagement/definition/database-replication
- J. Jenkov(2014). *Caching Techniques*. Jenkov. Sa https://jenkov.com/tutorials/software-architecture/caching-techniques.html
- *Shivang*(2019). *Distributed Cache*. Scaleyourapp. Sa https://scaleyourapp.com/distributed-cache-101-the-only-guide-youll-ever-need/
- K. Canchi(2020). *Healthcare Distributed Cache*. LinkedIn. Sa https://www.linkedin.com/pulse/healthcare-distributed-cache-kiran-canchi
- I. Ivanisenko & T. Radivilova(2015). *Survey of major load balancing algorithms in distributed system. Kharkiv: IEEE*
- N. Ramirez(2022). *What Is Load Balancing*. HAProxy. Sa https://www.haproxy.com/blog/what-is-load-balancing/
- A.Prabhu(2022). *Monitoring Spring Boot Application with Prometheus and Grafana*. RefactorFirst. Sa https://refactorfirst.com/spring-boot-prometheus-grafana
- Rajat, S. (2018, January 15). Web Application Architecture - Load Balancing and Caching [Video fajl]. Sa https://www.youtube.com/watch?v=U9P96XQ2688