**Univerzitet u Nišu**

**Elektronski fakultet**

Firebase - Database as a Service

Predmet: Sistemi za upravljanje bazama podataka

Student: Profesor:

Nevena Čolić 1360 prof. dr. Aleksandar Stanimirović

# Sadržaj

[**Database as a Service 3**](#_a0bicfgi1k13)

[**Firebase 4**](#_idcgwy6cwpzx)

**Rad RTDB 6**

[**Rad Firestore baze podataka**](#_upl6ugliotgz) **15**

[**Realtime Database ili Cloud Firestore?**](#_xfghjukingo3) **18**

# Database as a Service

Baza podataka kao Servis (eng. DBaaS), jeste servis klaud računarstva koji pruža pristup bazi podataka bez zahtevanja fizičkog hardvera, instalacije softvera ili konfigurisanja baze podataka. Veći deo održavanja i administracije izvršava pružaoc servisa, oslobađajući korisnike da brže iskoriste sve prednosti koje im baza nudi.

DBaaS je obuhvaćen pod SaaS (eng. Software as a Service), čija potražnja sve više raste. DBaaS pruža iste funkcionalnosti kao standardni tradicionalni i relacioni modeli, samo bez brige o konfiguraciji, bezbednosti i upravljanju bazom. Baza na ovaj način ubrzava rad organizacija olakšavajući rad i upravljanje bazama podataka.

Relacioni DBaaS koriste SQL bazu i nude tradicionalne sisteme za upravljanje kao što su Oracle, SQL Server i MySQL, ali i klaud rešenja: Amazon RDS i Azure. NoSQL DBaaS obuhvataju više DBMS-ova kao što su graf, dokument i key/value rešenja. Nezavisno od tipa klaud baze, DBaaS nudi elastične service za razvoj aplikacija, test okruženje, produkciono okruženje najčešće sa konzolom lakom za korišćenje i RESTful API.

DBaaS model je idealan za male i srednje biznise koji nemaju dobro opremljen IT departman. Na ovaj način, ovakve organizacije mogu lako implementirati aplikaciije i sisteme koje u suprotnom ne bi bili u mogućnosti.

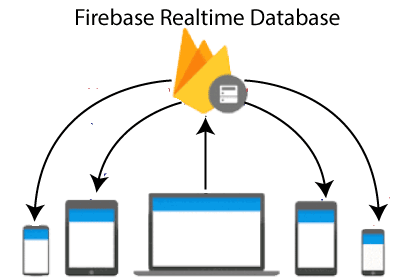
DBaaS model pruža dosta specifičnih prednosti nad tradicionalnom implementacijom:

* Umanjeni zahtevi menadžmenta - pružaoc usluga preuzima na sebe sve odgovornosti upravljanja i administracije.
* Eliminacija fizičke infrastructure - pružaoc DBaaS usluge daje i infrastrukturu potrebnu za upravljanje bazom.
* Smanjeni IT troškovi - korisnici ne moraju planirati i upravljati ažuriranjem hardvera baze.
* Povećana štednja - korisnici više ne moraju ulagati u skup hardver jer fizički hardver više nije potreban. Dodatno, potrebno je manje zaposlenih, manje fizičkog prostora što sa sobom donosi i čuvanje novca.

# Firebase

Firebase baza podataka je baza hostovana na klaudu u kojoj se podaci čuvaju u JSON formatu. Podaci se sinhronizuju u realnom vremenu za svakog povezanog korisnika. Svi korisnici dele jednu instancu baze i automatski primaju obaveštenja o najnovijim podacima.

U pitanju je NoSQL baza podataka u kojoj se mogu čuvati i pratiti podaci u realnom vremenu. Podaci se čuvaju u vidu velikog JSON objekta koji developeri mogu pratiti u realnom vremenu. Korišćenjem jednog API-ja, Firebase baza podataka pruža uvid u trenutnu vrednost podataka i njihovim ažuriranjima ali i lak pristup istim, bilo da je u pitanju web ili mobilni uređaj. Dolazi sa mobilnim i web SDK-ovima, što omogućava korisnicima da prave aplikacije bez potrebe za serverima. Kada su korisnici oflajn, SDK-ovi koriste lokalni keš uređaja za čuvanje promena. Lokalni podaci se automatski sinhronizuju odmah kada se uređaj vrati na mrežu.



**Slika 1. Skica rada Firebase baze podataka**

Sam Firebase obuhvata nekoliko baza podataka od kojih je najstarija i samim tim najstabilnija Firebase Realtime Database. Kao što je već rečeno, to je NoSQL klaud baza podataka koja se može povezati sa aplikacijom i pružiti pristup podacima u realnom vremenu kroz različite platforme.

Cloud Firestore je još jedna NoSQL baza hostovana na klaudu ali za razliku od prethodne, ova baza je dizajnirana za rad u velikim preduzećima koja zahtevaju stabilnost, skalabilnost, kompleksne modele podataka i napredne upite. Za pregled podataka obe baze može se koristiti Firebase konzola. Još jedna zajednička stvarim ovim dvema bazama jeste postojanje SDK-ova za rad sa kodom na serverskoj strani dostupnim u jezicima Python, Node.js, Goland, Ruby, PHP, Java, .NET i C#.

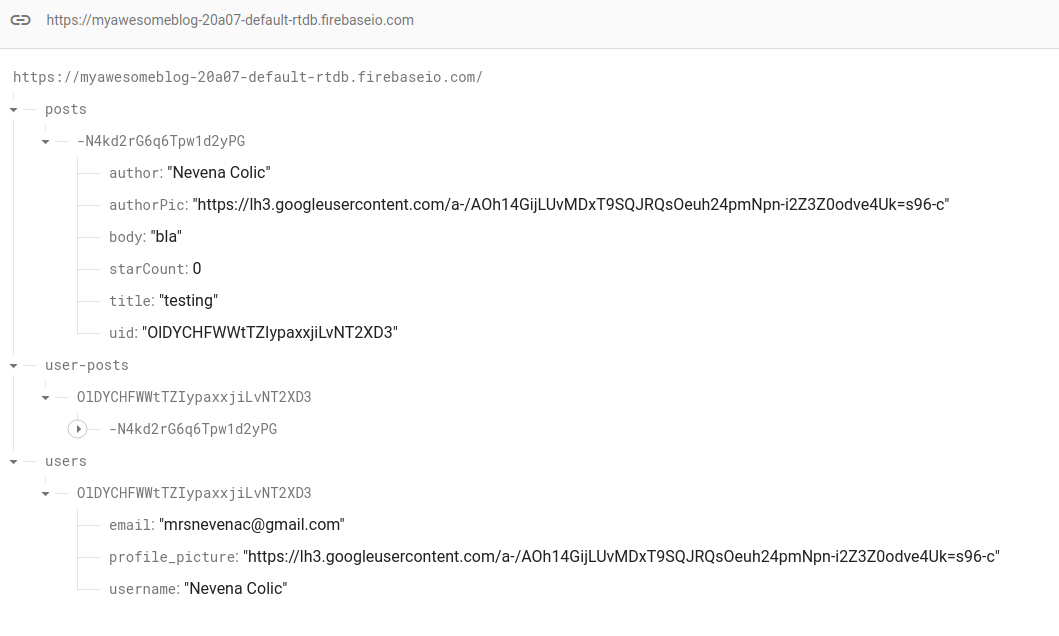
Kada je u pitanju Firebase Realtime Database, neke od mana uključuju i ograničene mogućnosti upita. S obzirom da je baza jedan veliki JSON fajl, filterovanje i izvršavanje kompleksnih upita postaje veoma teško. Još jedna negativna stvar bi bila modelovanje podataka. Upravo zbog strukure “baza kao jedan fajl”, nije moguće implementirati relacije između podataka.

Ključne osobine Realtime baze

1. Realno vreme - Umesto tipičnih HTTP zahteva, Firebase Realtime Database koristi sinhronizaciju podataka, tj. svaki put kada se podaci izmene, svaki povezani uređaj dobija obaveštenje u roku od par milisekundi.
2. Oflajn - Firebase aplikacije ostaju u aktivnom stanju čak i kada nema pristupa Internetu zbog toga što Firebase RTDB SDK čuva podatke na disku. Jednom kada je ponovo na mreži, klijent uređaj dobija sve izmene koje je propustio sinhronizujući se sa trenutnim stanjem servera.
3. Dostupna sa klijent uređaja - Firebase RTDB-u se može pristupiti sa mobilnog ili veb klijenta, nema potrebe za aplikacionim serverom.
4. Skaliranje preko nekoliko baza - Firebase Realtime Database je deo Blaze plana plaćanja i time podržava i skaliranje podataka kroz nekoliko instanci baze u okviru jednog Firebase projekta. Kontrola pristupa svakoj od baza se vrši pomoću korisnički definisanih pravila za svaku od instanci.

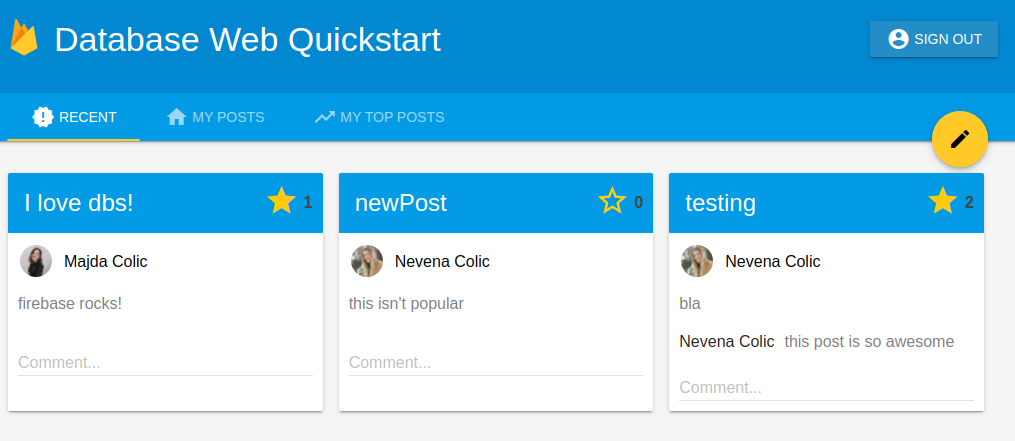
Rad RTDB

Svi RTDB podaci se čuvaju kao JSON objekti i zbog toga se baza i može zamisliti kao JSON stablo hostovano na oblaku. Za razliku od SQL baza, tabele ne postoje. Kada se doda podatak, on postaje čvor u JSON stablu sa odgovarajućim ključem. Te ključeve može obezbediti korisnik, kao što je na primer korisnički ID ili neko semantičko ime, a mogu biti i automatski generisani korišćenjem funkcije push(). Na Slici 2 prikazana je struktura aplikacije MyAwesomeBlog koja je korišćena za demonstraciju rada RTDB. U pitanju je blog sa funkcijama dodavanja posta, komentara, korisnika i rejtinga za dati post.



**Slika 2. MyAwesomeBlog - aplikacija za demonstraciju rada RTDB**

Firebase dozvoljava ugnježdavanje podataka do dubine 32, međutim to nije podrazumevana a samim tim ni preporučena struktura. Kada se podatak uzme iz baze, sa njim dolaze i sva njegova deca čvorovi a sa velikim brojem ugnježdenih podataka, iteracija kroz iste postaje problematična.

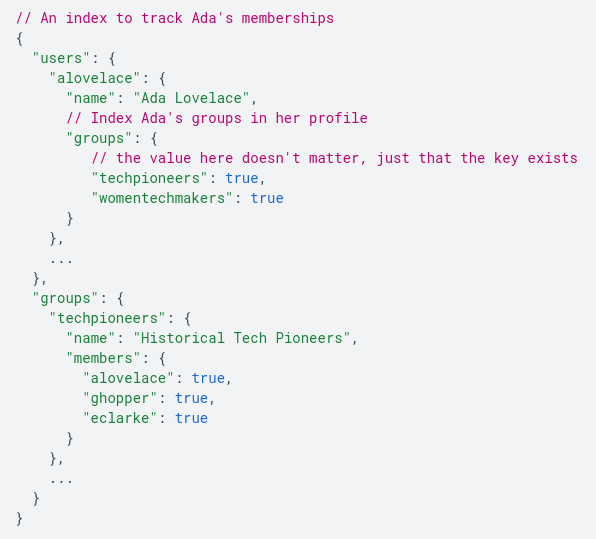


**Slika 3. MyAwesomeBlog korisnički interfejs**

Za bidirekcionalne veze javlja se neizbežna redundantnost. Na primer, ukoliko znamo da korisnici pripadaju određenim grupama ali i da grupe imaju listu korisnika koji su njihovi članovi, potrebno je smisliti elegantno rešenje, recimo za izlistavanje svih grupa kojima neki korisnik pripada. U tom slučaju koristi se indeks grupa kao na Slici 4.

Može se primetiti da se neki podaci na ovaj način dupliraju, čuvajući vezu pod podatkom korisnika Ada i sa grupom. Sada je *lovelace* indeksirana pod grupom i *techpioneers*  su izlistani u Adinom profilu. Tako da, da bi se Ada uklonila iz grupa, mora ista mora biti ažurirana na dva mesta. Ovo se ne može izbeći ali takođe i omogućava brzo i efikasno preuzimanje Adinog članstva čak i kada se lista korisnika skalira na milione ili kada pravila bezbednosti baze ograničavaju pristup nekim od podacima.

Ovaj pristup, invertovanje podataka izlistavanjem ID-eva kao ključeva i postavljenjem vrednosti na true, čini proveru ključeva jednostavnu kao što bi bilo čitanje /users/$uid/groups/$group\_id i provera da li je null. Indeks je brži i dosta efikasniji od upita i pretraživanja podataka.



**Slika 4. Bidirekciona veza između podataka**

Što se tiče upisa i čitanja podataka, najpre je potrebno uzeti referencu baze podataka komandom sa Slike 5:



**Slika 5. Referenca baze podataka**

Na referencu je zakačen asinhroni listener koji se okida kada se prvi put inicijalizuje baza i zatim svaki put kada se podaci promene. Za osnovne upise, može se koristiti funkcija set() koja prepisuje podatke sa određene lokacije, uključujući i decu čvorove, ukoliko oni postoje.



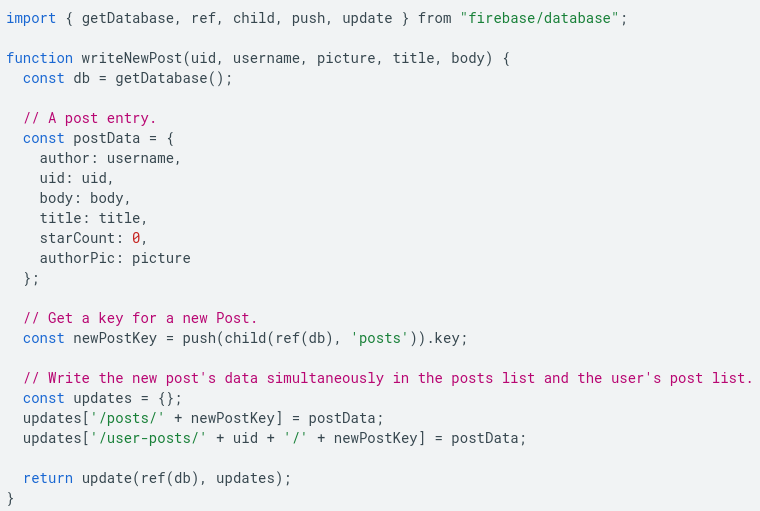
**Slika 6. Write operacija**

Za čitanje podataka sa lokacije i osluškivanje promena, koristi se onValue() listener. Ovaj metod se pokreće jednom na početku inicijalizacije i zatim svaki put kada se bilo šta od podataka, uključujući i decu, promeni. Callback funkciji se prosleđuje snapshot koji sadrži sve podatke sa lokacije. U slučaju da takvih nema, vratiće false vrednost kada se nad njim pozove funkcija exists() i null kada se pozove val(). Preporuka je da se ovaj listener zakači sto dublje u bazu, tj. na najniži mogući nivo gde je željeno pratiti promene jer se poziva na svaku promenu čvora i sve njegove dece, samim tim, nije dobra praksa zakačiti ga na koren baze.

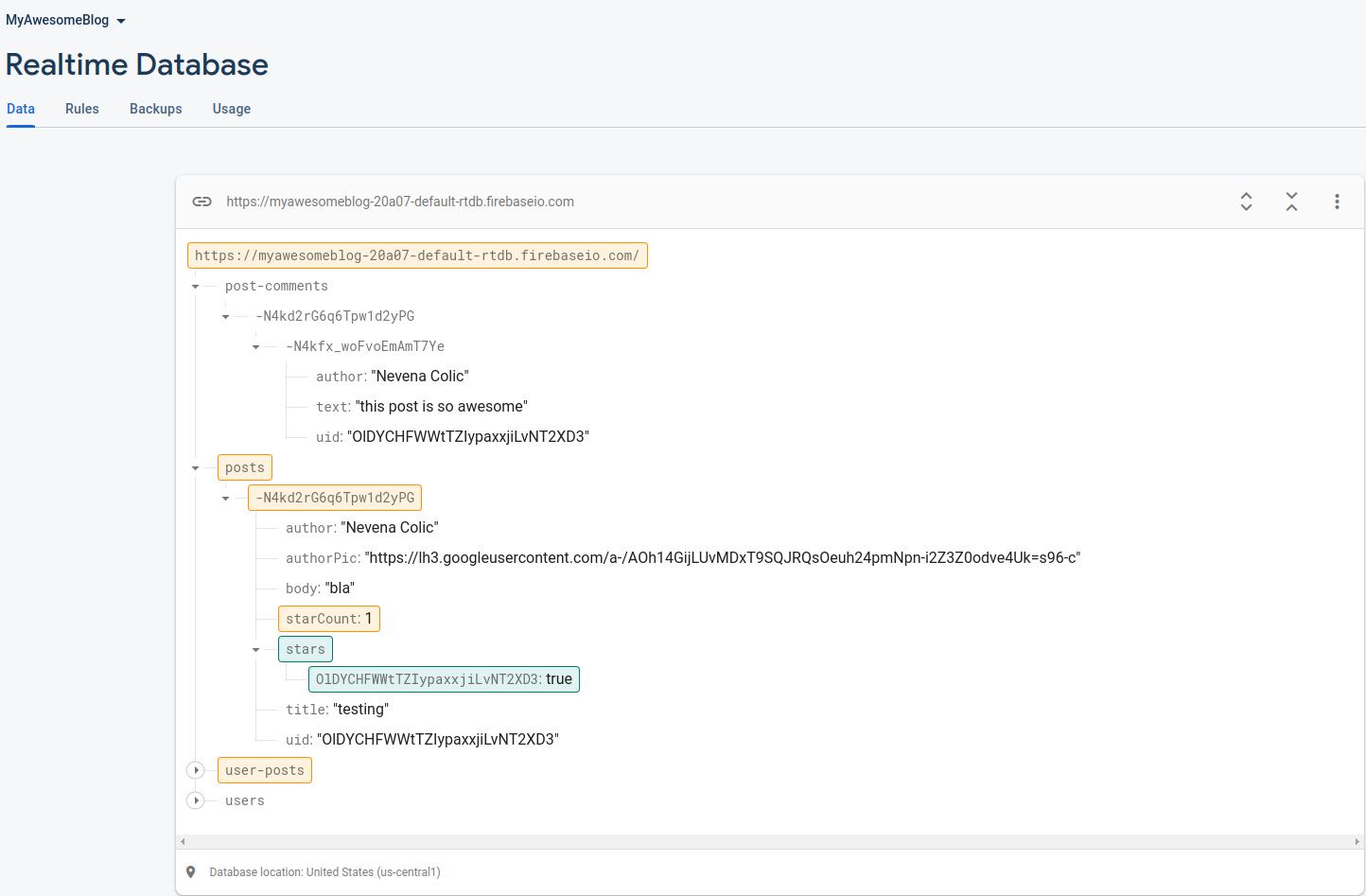


**Slika 7. Read operacija**

Takođe, za upis i ažuriranje određenih čvorova bez uticaja na druge, može se koristiti update() metoda. Na ovaj način se mogu ažurirati članovi nižih nivoa u bazi zadavajući putanju do ključa. U slučaju se podaci čuvaju na više lokacija radi boljeg skaliranja, mogu se ažurirati sve instance koristeći data fan-out.



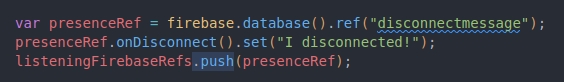
**Slika 8. Funkcija update()**



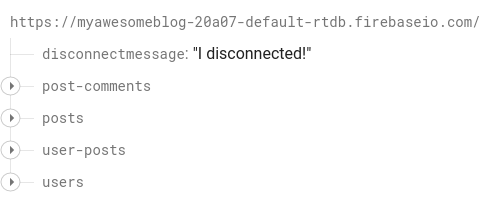
**Slika 9. Ažuriranje podataka**

Kada su u pitanju aplikacije koje se izvršavaju u realnom vremenu, često je veoma korisno pratiti stanje konekcije korisnika. U slučaju gubitka mreže, korisnik se može označiti kao oflajn ili se može preduzeti neka druga akcija.

RTDB pruža jednostavne primitive kojima se može vršiti upis u bazu i kada se klijent diskonektovao sa Firebase servera. Ova ažuriranja obavljaju nezavisno od načina i razloga gubitka konekcije, tako da su od poverenja što se tiče čišćenja podataka u slučaju kraha klijenta ili pada mreže. Sve operacije upisa se mogu izvršiti nakon prekida veze.



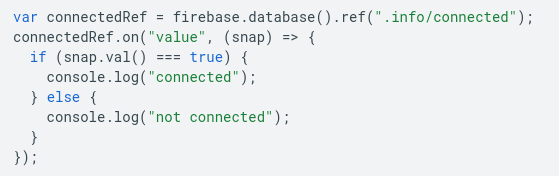
**Slika 10. Disconnect funkcija**



**Slika 11. Ažuriranje baze po gubitku veze**

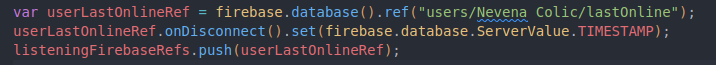
Kada se postavi onDisconnect() operacija, ona živi na RTDB serveru. Server proverava stanje i mogućnost izvršavanja operacija upisa i javlja aplikaciji ukoliko je došlo do nemogućnosti istog. Server zatim proverava konekciju i ukoliko ona istekne ili bude zatvorena od strane klijenta, server ponovo proverava validnost write operacije i zatim poziva operaciju.

Za mnoge operacije, korisno je da aplikacija zna kada je onlajn a kada je bez mreže. Firebase Realtime Database pruža specijalnu lokaciju */.info/connected* koja se ažurira svaki put kada se promeni stanje konekcije. U pitanju je boolean vrednost koja se ne sinhronizuje između klijenta jer zavisi od stanja konkretnog klijenta.



**Slika 12. Provera konekcije**

Još jedna od mogućnosti RTDB-a jeste provera kada je klijent poslednji put bio na mreži. Ovo, u kombinaciji sa onDisconnect funkcijom pruža lak način pouzdanog praćenja vremena u kojem je klijent tačno izgubio konekciju:



**Slika 13. Praćenje lastOnline timestamp-a**



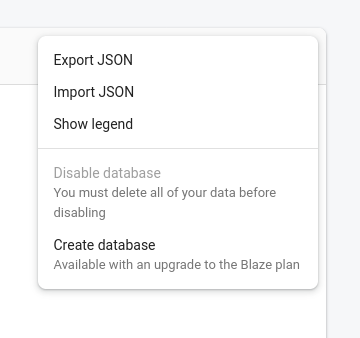
**Slika 14. lastOnline i disconnectMessage flegovi baze**

Najbolji način za optimizaciju perfomansi i skaliranja podataka u RTDB jeste podela podataka na nekoliko instanci baze, takođe poznato i kao database sharding. To pruža potrebnu fleksibilnost za skaliranje preko granica koje ima individualna instanca baze ali i dodatno balansiranje opterećenja i optimizaciju izvršenja. Neki od scenarija kada je sharding potreban uključuju:

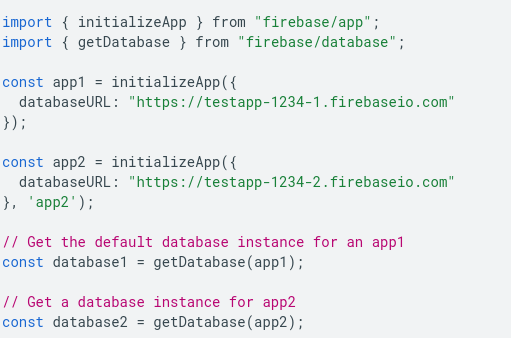
* Potrebno je skaliranje preko ograničena od 200,000 istovremenih konekcija, 1,000 operacija upisa po sekundi ili bilo kog drugog definisanog limita.
* Postoji nekoliko diskretnih skupova podataka i potrebna je optimizacija perfomansi (npr. chat aplikacija koja ima različite, nezavisne grupe korisnika).
* Potreban je balans opterećenja preko nekoliko baza radi poboljšanja aktivnog vremena i smanjenja rizika od preopterećenja jedne instance.

Kada se radi mapiranje podataka da bi se omogućio sharding, potrebno je zadovoljiti sledeće uslove:

* Svaki upit radi na jednoj instanci. RTDB ne podržava upite koji se izvršavaju na više instanci istovremeno.
* Minimalno ili nepostojeće dupliranje podataka.
* Svaka instanca aplikacije se povezuje na samo jednu instancu u jednom trenutku.



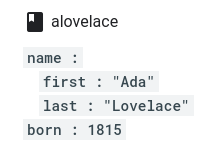
**Slika 15. Kreiranje nove instance**



**Slika 16. Povezivanje aplikacije sa bazama**

# Rad Firestore baze podataka

Firestore je takođe NoSQL baza hostovana na oblaku sa kojom Apple, Android i veb aplikacije komuniciraju direktno koristeći native SDK. Firestore je takođe dostupna i native Node.js, Java, Python, Unity, C++ i Go SDK-ovima. Firestore je document-oriented baza podataka tako da za razliku od SQL baza, nema tabela i redova. Naprotiv, podaci se čuvaju u dokumentima koji su dalje organizovani u kolekcije. Dokument je lightweight podatak koji se sastoji iz polja koja se mapiraju u vrednosti. Svaki dokument ima svoje ime. Dokument koji predstavlja korisnika *alovelave* izgleda kao na Slici 17. Kompleksni, ugnježdeni objekti u okviru dokumenta se nazivaju mape i oni su prikazani na Slici 18.

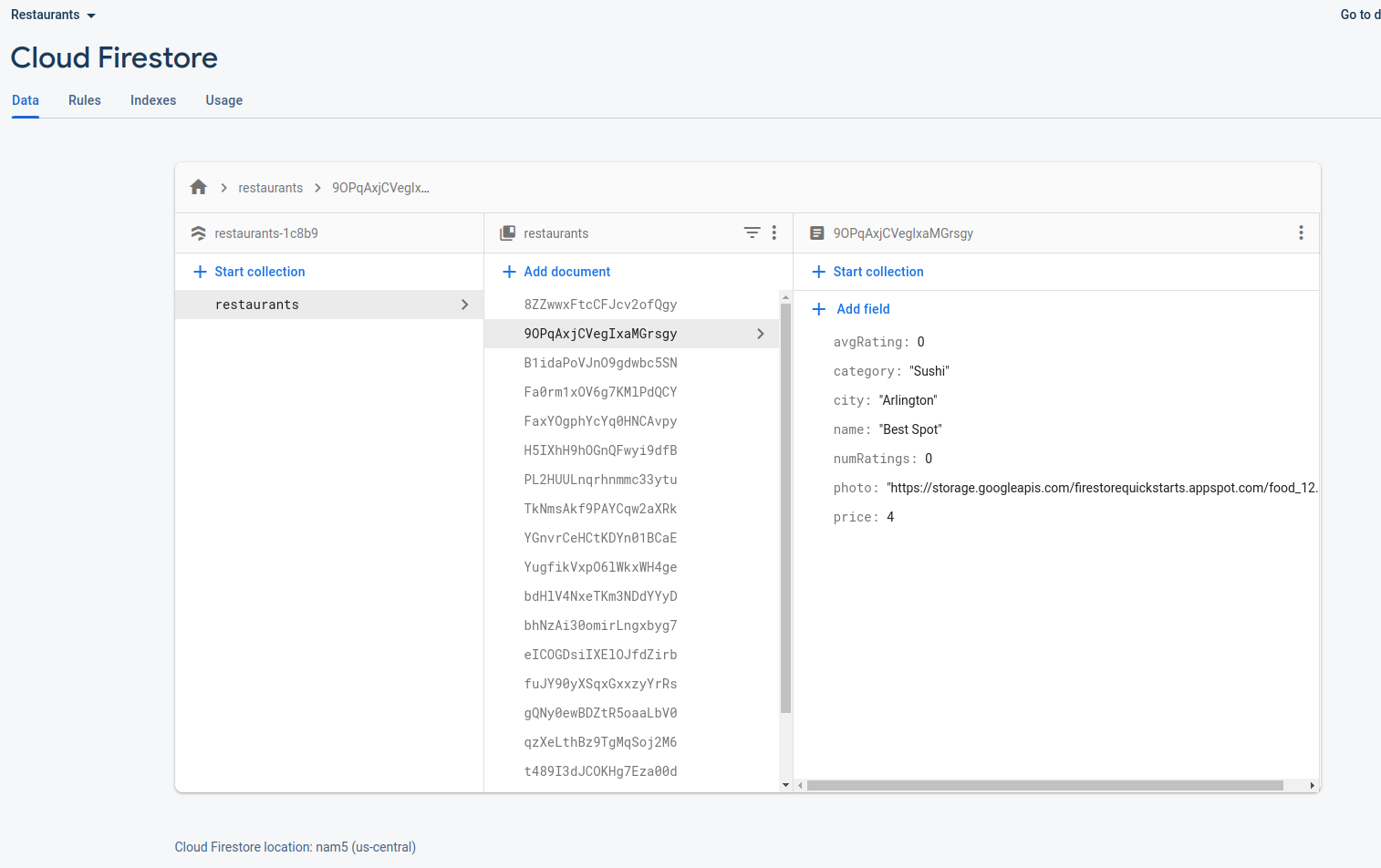
 

**Slika 17. Korisnički dokument** **Slika 18. Dokument sa ugnježdenim podacima**

Dokumenti žive u kolekcijama. Cloud Firestore je bez šeme tako da pruža kompletnu slobodu oko toga koja polja ima pojedinačni dokument i kog su tipa ti podaci. Dokumenti u okviru iste kolekcije mogu imati različita polja ili podatke različitog tipa. Preporuka je međutim, da kolekcije budu što konzistentnije radi boljeg pretraživanja.



**Slika 19. Firestore inicijalizacija**



**Slika 20. Struktura podataka**

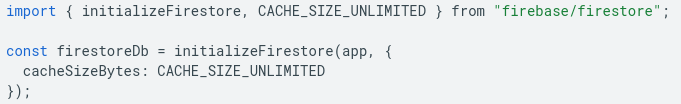


**Slika 21. Write operacija Firestore baze podataka**



**Slika 22. Firestore upiti i filtriranje - sintaksa**

Da bi postojala konzistentnost i kada aplikacija nije na mreži, prilikom inicijalizacije aplikacije potrebno je postaviti vrednost održavanje perzistencije na true. Kada su u pitanju mobilne aplikacije, to je podrazumevano podešavanje dok je za veb potrebno to eksplicitno postaviti. Razlog za to je to što se keš ne čisti između sesija, samim tim, ukoliko aplikacija radi sa osetljivim podacima, potrebno je pitati korisnika da li su na sigurnom uređaju pre nego što se dozvoli oflajn perzistencija.



**Slika 23. Podešavanje veličine keša**

Jednom kada uređaj više nije na mreži a oflajn perzistencija je omogućena, listeneri će biti obavešteni kada se podaci iz keša promene. U slučaju get poziva, tj. read operacije, ako uređaj nije na mreži, podaci će biti dobavljeni iz keša. Moguće je takođe i izvršavanje različitih upita i filtriranja no sve će se izvršavati nad keširanim podacima.

# Realtime Database ili Cloud Firestore?

Kao što je prethodno i objašnjeno, Firebase nudi dve baze podataka, a izbor prave zavisi od više faktora vezanih za potrebe konkretne aplikacije.

|  | **RTDB** | **Firestore** |
| --- | --- | --- |
| Uloga baze | Primarna sinhronizacija podataka sa osnovnim mogućnostima upita | Napredniji upiti, sortiranje i transakcije |
| Operacije nad podacima | Nekoliko GB podataka ili manje se često menja | Stotine GB do TB podataka koji se čitaju češće nego što se menjaju |
| Model podataka | Jednostavno JSON stablo | Dokumenti organizovani u kolekcije |
| Dostupnost | Garancija dostupnosti najmanje 99.95% vremena | Ekstremno visoka garancija dostupnosti od 99.999% |
| Oflajn upiti | Retko | Sofisticirani oflajn upiti nad lokalnim podacima |
| Broj instanci baze | Više baza, na primer po jedna za svakog velikog klijenta | Jedna instanca |

Još neke od razlika koje treba uzeti u obzir:

1. **Model podataka**

RTDB - podatke čuva u vidu JSON stabla. Jednostavni podaci su laki za čuvanje dok kompleksni, hijerarski podaci postaju komplikovaniji za skaliranje.

Cloud Firestore - čuva podatke u kolekcije dokumenata koji su strukture slične JSON-u. Kompleksne podatke je lako snimiti kao podkolekcije sa dokumentima. Zahteva manje denormalizacije i ispravljanja podataka.

1. **Prisustvo**

Korisno je znati kada je klijent onlajn ili oflajn. RTDB može čuvati konekciju klijenta, njen status i pružiti odgovarajuća obaveštenja svaki put kada se to stanje promeni.

Cloud Firestoru nema takvu podršku već ukoliko je potrebno, sinhronizuje se sa RTDB pomoću Cloud Funkcija.

1. **Upiti**

RTDB - Duboki upiti sa ograničenim sortiranjem i filtriranjem podataka. Upiti mogu sortirati ili filtrirati ali ne oba. Upit uvek vraća kompletno stablo i mogu pristupiti i poslednjem listu. Upitima nije potreban indeks međutim optimalnost opada kako skup podataka raste.

Cloud Firestore - Indeksovani upiti sa naprednim sortiranjem i filtriranjem podataka. Funkcije se mogu ulančati u okviru istog upita. Upiti ne idu u dubinu, uvek vrate samo traženi dokument a ne i celu podkolekciju. Po defoltu su indeksovani. Performanse upita zavise od veličine rezultata, ne svih podataka.

1. **Pouzdanost i perfomanse**

RTDB je dostupna u regionalnim konfiguracijama. Baze su ograničene na zonsku dostupnost u okviru regiona. Ekstremno mala latencija, ovo je idealna opcija za učestalu sinhronizaciju stanja.

Cloud Firestore je regionalno i multi regionalno rešenje koje se skalira automatski. Ona čuva podatke u nekoliko centara u različitim regionima postižući globalnu skalabilnost i jaku pouzdanost podataka.

1. **Skalabilnost**

RTDB - Skaliranje zahteva podelu podaka u nekoliko baza. Moguće je skalirati 200,000 konkurentnih konekcija i 1,000 upisa po sekundi. Sve preko toga zahteva podelu podataka u više baza.

Cloud Firestore - Skaliranje se vrši automatski. Trenutno, ograničenja skaliranja su na oko 1 million konkurentnih konekcija i 10,000 upisa po sekundi međutim trenutno se radi na povećanju tih ograničenja.

1. **Cena**

RTDB - naplaćuje protok podataka i skladište ali po većoj ceni.

Cloud Firestore - naplaćuje po operacijama koje se vrše nad bazom (read, write, delete) i po nižoj ceni skladište i protok.

# Literatura

<https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>

<https://www.educative.io/answers/what-is-firebase>

<https://firebase.google.com/docs/database/web/start>

<https://firebase.google.com/docs/database/web/read-and-write#web-version-9_1>

<https://firebase.google.com/docs/database/web/offline-capabilities>

<https://firebase.google.com/docs/database/usage/sharding>

<https://firebase.google.com/docs/database/web/structure-data>

<https://blog.back4app.com/what-is-firestore/>

<https://firebase.google.com/docs?gclid=Cj0KCQjwkruVBhCHARIsACVIiOzvtXCt9B-sVM__RulLJpKXW8WbeMSDYf8Mer7-nqMNlDzD76p53oUaAtVNEALw_wcB&gclsrc=aw.ds>

<https://firebase.google.com/docs/firestore/manage-data/enable-offline>

<https://firebase.google.com/docs/firestore/rtdb-vs-firestore>