

KLJUČ – VRIJEDNOST BAZE PODATAKA

MODELIRANJE KLJUČ – VRIJEDNOST BAZE PODATAKA ZA
PUBLIKACIJU VIDEA I IZGRADNJA WEB-APLIKACIJE NAD NJOM

Juraj Brigljević | Ivona Čižić | Ana Marija Zebić

Prirodoslovno – matematički fakultet u Zagrebu
Napredne baze podataka

12. lipnja 2022.

Sadržaj

1.	UVOD	2
2.	NoSQL MODELI PODATAKA	3
3.	KLJUČ – VRIJEDNOST BAZE PODATAKA	5
4.	REDIS	6
4.1.	Što je Redis?.....	6
4.2.	Tipovi podataka u Redisu.....	6
4.2.1.	String.....	7
4.2.2.	List	7
4.2.3.	Hash	8
4.2.4.	Set.....	8
4.2.5.	Ordered set	9
5.	OBLIKOVANJE MODELA.....	10
5.1.	Analiza korisničkih potreba i zahtjeva	10
5.2.	Oblikovanje modela	11
5.2.1.	Podaci o korisniku & podaci o broju registriranih korisnika	11
5.2.2.	Indeks korisnika	12
5.2.3.	Podaci o videozapisima korisnika s identifikatorom id_korisnika & podatak o indeksu videozapisa	13
5.2.4.	Podaci o videozapisu.....	13
5.2.5.	Podaci o tag-ovima videozapisa id_videozapisa korisnika id_korisnika:	14
5.2.6.	Podaci o tag-ovima.....	14
6.	TEHNIČKI UVJETI	15
7.	UPITI NA BAZU	16
7.1.	Spremanje u bazu.....	16
7.2.	Dohvaćanje podataka iz baze.....	17
7.3.	Pretraživanje	18
8.	APLIKACIJA VideoShare	19
9.	Zaključak i ideje o poboljšanjima..... Pogreška! Knjižna oznaka nije definirana.	
10.	BIBLIOGRAFIJA.....	24

1. UVOD

Ovaj je projekt izrađen u sklopu kolegija **Napredne baze podataka** koji se održava na prvoj godini diplomskog studija Računarstva i matematike na Prirodoslovno – matematičkom fakultetu u Zagrebu.

Glavni je cilj projekta upoznavanje s osnovnim konceptima i svojstvima ključ - vrijednost baza podataka te oblikovanje i izgradnja jedne takve baze. Baza podataka koju ćemo u ovome projektu prezentirati modelirana je tako da pohranjuje podatke o videozapisima i korisnicima koji ih u bazu dodaju posredstvom web-aplikacije. Aplikacija je izgrađena u svrhu jednostavnog, praktičnog i vjerodostojnog prikaza mogućnosti upotrebe ovakvog modela pohrane podataka u svakodnevnom životu. Interesantnom se pokazuje moć ključ - vrijednost baza podataka, pa unatoč tome što koriste zasigurno najjednostavniji *NoSQL* model spremanja informacija, omogućuju uistinu raznovrsnu upotrebu. Postupno razlažući problematiku ovog zadatka objasniti ćemo korake u kojima smo došli do konačnog rješenja, komentirajući zapreke i poteškoće na koje smo pritom naišli.

2. NoSQL MODELI PODATAKA

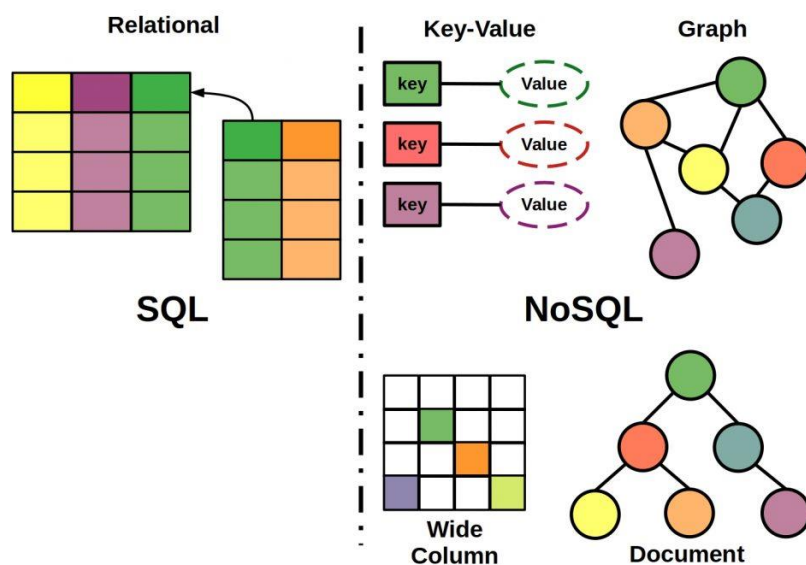
Relacijske su baze podataka odavno utemeljile svoj položaj najraširenijeg načina pohrane podataka. Iako su zbog svoje dugovječnosti zapravo postale sinonim samim bazama podataka, u zadnjih se petnaest godina razvija i na tržištu postupno rasprostranjuje nekolicina alternativnih, takozvanih *NoSQL* baza podataka. Potreba za novim modelom pohrane podataka javila se nekontroliranim i munjevitim širenjem Interneta, koji pred bazu podataka stavlja izazove obrade ogromnih količina podataka i zahtjeva u što kraćem vremenskom intervalu.

Glavna karakteristika *NoSQL* (*Not only SQL*) modela podataka **napuštanje** je **relacijskog modela**. Umjesto njega, koriste se mnogo jednostavniji i slobodniji modeli, što rezultira time da ovakve baze ne zahtijevaju precizno definiranje sheme podataka. Upravo im to omogućuje dinamičnost i prilagodljivost na način da se broj atributa određenog entiteta ne mora definirati prije samog modeliranja baze, već je on promjenjiv pa dodavanjem novih ili brisanjem postojećih atributa nećemo oštetiti ostale podatke u bazi. Ovo svojstvo uvelike pojednostavljuje interakciju modernih objektno-orijentiranih aplikacija i baza podataka.

Dodatna bitna karakteristika *NoSQL* sustava njihova je **distribuiranost**, odnosno mogućnost raspoređivanja podataka iste baze na više računala tako da svako računalo radi sa svojim podskupom podataka. Ovime se uvelike skraćuje vrijeme prihvata i obrade podataka. Ne iznenađuju stoga razmišljanja mnogih stručnjaka koji u ovakvim modelima vide budućnost, navodeći veliku **horizontalnu skalabilnost** i financijsku uštedu do koje ona dovodi kao jedan od najbitnijih faktora za daljnji razvoj ovih tehnologija.

NoSQL baze podataka najčešće razvrstavamo prema modelu podataka. Model podataka definira način predstavljanja podataka pohranjenih u bazi. Odabir modela potrebno je prilagoditi potrebama i svojstvima podataka koje njime želimo reprezentirati. Razvijena su i u širokoj upotrebi četiri *NoSQL* modela podataka:

- **Ključ – vrijednost** (*key – value*)
- **Dokument** (*document*)
- **Column family**
- **Graf** (*graph*)



Slika 1: Razlika relacijskog i NoSQL modela podataka

RELATIONAL			NON-RELATIONAL (NO SQL)	
			DOCUMENT	KEY VALUE
 Cloud SQL Managed MySQL, PostgreSQL, SQL Server	 Cloud Spanner Cloud-native with large scale, consistency, 99.999% availability	 Bare Metal Lift and shift Oracle workloads to Google Cloud	 Firestore Cloud Native, serverless, NoSQL document database, backend-as-a-service, global strong consistency, 99.999% SLA	 Cloud Bigtable Cloud-native NoSQL wide-column store for large scale, low-latency workloads
Good For:			Good For:	
General purpose SQL DB	RDBMS+ scale, HA, HTAP	RDBMS+ scale, HA, HTAP	Large scale, complex hierarchical data	Heavy read + write, events
Use Case:			Use Case:	
Web frameworks ERP CRM Ecommerce and web SaaS application	Gaming Global financial ledger Supply chain/inventory management	Legacy applications Data center retirement	Mobile/web/IoT applications Real-time sync Offline sync Personalized apps	Personalization Adtech Recommendation engines Fraud detection

Slika 2: Područja korištenja relacijskih i NoSQL baza podataka

3. KLJUČ – VRIJEDNOST BAZE PODATAKA



Ključ - vrijednost baza podataka nerelacijska je baza zasnovana na najjednostavnijem modelu podataka koji se sastoji od parova (ključ, vrijednost), gdje je svakom tekstualnom ključu pridružena jedna vrijednost proizvoljnog tipa. Vrijednost je zapravo zavisni objekt koji je jedinstveno određen sadržajem svoga ključa.

Ključ je apsolutno krucijalan za snalaženje u ovakvom modelu podataka, budući da se sve osnovne operacije za dohvat i upravljanje podacima u ovakvim bazama izvršavaju upravo posredstvom ključeva. Implementaciju ključ - vrijednost baze podataka dobro je promatrati kao *hash-tablicu* s ključevima i pokazivačima na vrijednosti.

Najveća je prednost ovakvog modela njegova **jednostavnost**. Upravo su zbog toga ključ – vrijednost baze poznate po izvanrednim performansama koje svu svoju moć pokazuju u scenarijima gdje baza dobiva mnogo upita za čitanje i pisanje, a vrlo malo upita za ažuriranje. Također, izuzetno jednostavno particioniranje i velika horizontalna skalabilnost čine ovaj model veoma moćnim, unatoč njegovoj jednostavnosti.

U svom osnovnom obliku, ovakve baze podataka podržavaju tri operacije:

- **GET (ključ)**
 - vraća vrijednost pridruženu zadanom ključu
- **PUT (ključ, vrijednost)**
 - dodaje novi par (ključ, vrijednost) ili pridružuje novu vrijednost postojećem ključu
- **DELETE (ključ)**
 - uklanja par (ključ, vrijednost), ako ključ ne postoji može dojaviti grešku

Pohranjeni identifikator može biti u raznim formatima, a njemu pridružene vrijednosti također mogu biti raznolike, pa tako u ovakvim bazama podataka možemo spremati slike, zvuk, internetske stranice, dokumente i slično.

Mnogo je ovakvih baza na tržištu, ali među najbitnijima i najkorištenijima ističu se **Amazon DynamoDB**, **Riak KV** te **Redis**. Prilikom izrade naše baze podataka koristili smo se upravo *Redisom*.

4. REDIS

4.1. Što je Redis?

Redis je vrlo jednostavna *NoSQL* baza podataka bazirana na ključ - vrijednost modelu podataka koja, za razliku od drugih baza, skup podataka pohranjuje u radnoj memoriji, umjesto na disku te ga povremeno asinkrono upisuje na disk.

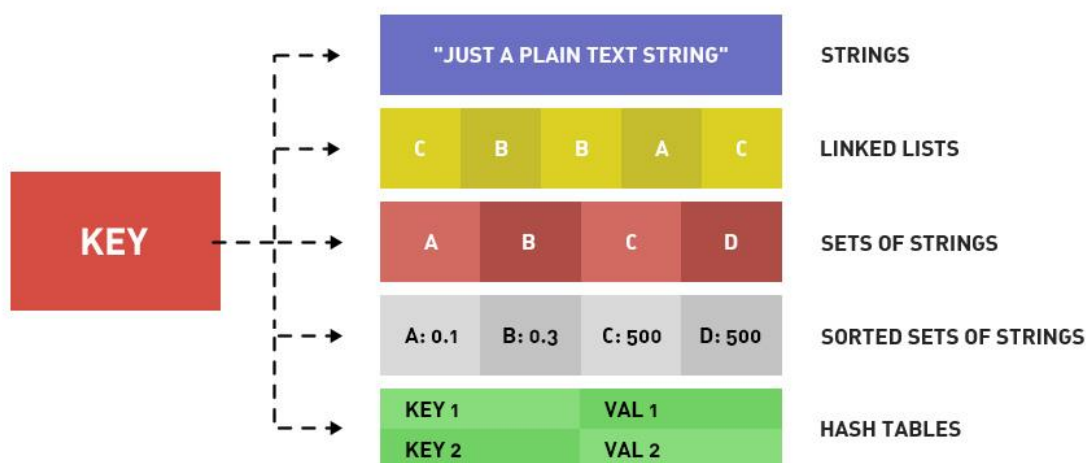
Upravo ovo nesvakidašnje svojstvo omogućuje **nevjerojatno efikasno upravljanje podacima** te čini *Redis* jednom od najbržih baza podataka ovoga tipa. Međutim, asinkrono spremanje podataka izlaže nas opasnosti da u slučaju rušenja sustava trajno izgubimo dio podataka iz baze.

Uzimajući u obzir njegove izvanredne vremenske performanse, ali i upravo opisanu ranjivost, *Redis* se uglavnom ne koristi kao samostalna baza podataka, već kao potpora nekoj drugoj, tradicionalnoj bazi. Pritom u *Redis* dupliciramo informacije čije bi dohvaćanje ili računanje iz primarne baze iziskivalo značajne vremenske resurse, kao i podatke kojima moramo često pristupati, a rijetko ih mijenjati. Nakon upravo opisane nadogradnje središnje baze *Redisom*, višestruko se smanjuje vrijeme pristupa podacima koji su dodatno pohranjeni u *Redis* bazu što rezultira značajnim poboljšanjem performansi svakog sustava ili aplikacije koji koristi podatke iz naše baze.

4.2. Tipovi podataka u Redisu

Svakome je ključu u *Redis* bazi podataka pridružena vrijednost koja može biti pohranjena u jednom od sljedećih pet osnovnih podržanih tipova podataka:

- **String** (string)
- **Hash** (hash)
- **Lista** (list)
- **Skup** (set)
- **Uređeni skup** (ordered set)



Slika 3: Tipovi podataka u Redisu

4.2.1. String

String je najjednostavniji tip podataka implementiran u *Redisu*, predstavljen kao niz bajtova. Jedino je ograničenje ove strukture veličina koja sadržaj svakoga stringa ograničava na maksimalno 512 MB. Jednostavan primjer postavljanja i dohvaćanja vrijednosti tipa string korištenjem prethodno opisanih naredbi *GET* i *SET* promatramo u sljedećem primjeru:

```
redis 127.0.0.1:6379> SET name "tutorialspoint"
OK
redis 127.0.0.1:6379> GET name
"tutorialspoint"
```

Slika 4: Redis string

4.2.2. List

Redis liste jednostavan su tip podataka koji predstavlja listu stringova sortiranih po vremenu umetanja. Uz mnoge operacije specificirane za upravljanje podacima zapisanima u listi, omogućeno je i umetanje i micanje podataka s početka, kao i sa kraja liste. U jednu je listu moguće pohraniti $2^{32} - 1$ elemenata. Primjer umetanja liste kao vrijednosti uz ključ *tutoriallist* prikazan je na sljedećem primjeru:


```
redis 127.0.0.1:6379> lpush tutoriallist redis
(integer) 1
redis 127.0.0.1:6379> lpush tutoriallist mongodb
(integer) 2
redis 127.0.0.1:6379> lpush tutoriallist rabbitmq
(integer) 3
redis 127.0.0.1:6379> lrange tutoriallist 0 10

1) "rabbitmq"
2) "mongodb"
3) "redis"
```

Slika 5: Redis list

4.2.3. Hash

Redis hash mapiranje je sa liste stringova u skup stringova i upravo je zbog toga idealna struktura za reprezentaciju objekata. U sljedećem primjeru hash pohranjuje objekt o korisniku koji sadrži njegove osnovne informacije poput korisničkog imena, lozinke i broja bodova. Pritom koristimo funkcije *HMSET* i *HGETALL* specifične za upravljanje hashevima u *Redisu*. Svaki hash može pohraniti maksimalno $2^{32} - 1$ prethodno opisanih parova.

```
redis 127.0.0.1:6379> HMSET user:1 username tutorialspoint password
tutorialspoint points 200
OK
redis 127.0.0.1:6379> HGETALL user:1
1) "username"
2) "tutorialspoint"
3) "password"
4) "tutorialspoint"
5) "points"
6) "200"
```

Slika 6: Redis hash

4.2.4. Set

Redis set neuređena je kolekcija stringova, odnosno skup, za koju je omogućeno dodavanje, uklanjanje i provjeravanje pripadnosti elementa skupu s $O(1)$ vremenskom složenosti. Promotrimo primjer funkcionalnosti specifičnih za ovu strukturu podataka, pokušavajući pritom dodati isti element u skup dva puta.

Budući da je svaki element u skupu nužno jedinstven, takva će operacija vratiti 0 informirajući korisnika pritom da operacija nije uspješno izvršena.

```
redis 127.0.0.1:6379> sadd tutoriallist redis
(integer) 1
redis 127.0.0.1:6379> sadd tutoriallist mongodb
(integer) 1
redis 127.0.0.1:6379> sadd tutoriallist rabbitmq
(integer) 1
redis 127.0.0.1:6379> sadd tutoriallist rabbitmq
(integer) 0
redis 127.0.0.1:6379> smembers tutoriallist

1) "rabbitmq"
2) "mongodb"
3) "redis"
```

Slika 7: Redis set

4.2.5. Ordered set

Redis ordered set slični su običnim skupovima, sa istim svojstvom nužne jedinstvenosti svake vrijednosti pohranjene u njima. Razlika je u tome što je svakome elementu skupa pridružena brojana vrijednost koju možemo tumačiti kao neku vrstu mjere odnosno rezultata. Takva nam struktura omogućava sortiranje vrijednosti pohranjenih u kolekciji. Iako su same vrijednosti nužno jedinstvene, rezultati se mogu ponavljati.

```
redis 127.0.0.1:6379> zadd tutoriallist 0 redis
(integer) 1
redis 127.0.0.1:6379> zadd tutoriallist 0 mongodb
(integer) 1
redis 127.0.0.1:6379> zadd tutoriallist 0 rabbitmq
(integer) 1
redis 127.0.0.1:6379> zadd tutoriallist 0 rabbitmq
(integer) 0
redis 127.0.0.1:6379> ZRANGEBYSCORE tutoriallist 0 1000

1) "redis"
2) "mongodb"
3) "rabbitmq"
```

Slika 8: Redis ordered set

5. OBLIKOVANJE MODELA

5.1. Analiza korisničkih potreba i zahtjeva

3. Modelirati ključ-vrijednost bazu podataka za publikaciju videa i izgraditi aplikaciju nad njom

Baza podataka treba omogućavati pohranu podataka o korisnicima i o videima koje korisnici mogu objaviti korištenjem aplikacije. Svaki korisnik može unijeti osobne podatke i sliku te učitati veći broj videozapisa koje može opisati i tagirati. Modelirati način pohrane podataka. Koristiti redis. Napuniti podatke za 10 korisnika (generirati po potrebi) i 20 videa (radi jednostavnosti, videozapisi mogu biti pohranjeni i kao linkovi za embedanje s popularnih servisa kao što su YouTube ili Vimeo). Izgraditi aplikaciju (u programskom jeziku/okruženju po želji) u koju će se korisnik moći jednom registrirati i koja će mu dohvatiti njegove podatke ako je već registriran. Kroz aplikaciju korisnik mora moći pregledati i urediti svoje zapise. Neregistrirani korisnik kroz aplikaciju mora moći pregledati sve zapise i pretraživati ih prema nazivu i tagovima.

Rješavanje ovog problema započeli smo promišljanjem o samom modeliranju baze prigodne za pohranu prethodno opisanih podataka. Najbitniji proces u ovome, početnome koraku zasigurno je detaljna analiza potreba koje krajnji korisnik pred nas stavlja. Tako smo naše promišljanje započeli pitanjem koje sve podatke želimo pospremati u bazu, odnosno koju funkciju želimo da strukture, u koje ćemo spremati određene informacije, zadovoljavaju.

Znamo da aplikacija koju razvijamo treba imati korisnike, koji imaju mogućnost **registracije i logiranja**. Registracija omogućuje zapisivanje korisničkog imena, lozinke, imena, prezimena te postavljanje profilne slike korisnika.

Potom želimo da svaki registrirani korisnik nakon uspješno provedene prijave može na stranicu **objaviti jedan ili više videozapisa**. Umjesto pospremanja samih videozapisa u bazu, odlučili smo rješavanje problema olakšati koristeći *embed* funkciju s *YouTube*-a, koja nam daje *html* kod koji omogućava prikazivanje videozapisa u bilo kojem *html* dokumentu. Ovom smo odlukom značajno pojednostavili podatke koje pohranjujemo o korisničkom videozapisu pa se tako spremanje istog svodi na pamćenje njegovog *linka*, naslova i opisa.

Budući da samo pohranjivanje videozapisa ne bi imalo preveliku praktičnu upotrebu bez mogućnosti njihova prikaza, odabir videozapisa za reprodukciju korisniku olakšavamo implementiranjem dodatne funkcionalnosti za **pretraživanje videozapisa** po nekim, korisnim,

parametrima. Očito je rješenje pretraživanje po imenu koje je ovakvim modelom već omogućeno, međutim korisnim uviđamo pamtiti i *tag*-ove, ključne riječi ili skup istih, koje traženi videozapis opisuju na sažet, ali bogat i intuitivan način. Implementacijom ovakve funkcionalnosti omogućeno je brzo pretraživanje videozapisa po njihovoj sličnosti. *Tag*-ove za pojedini zapis upisuju korisnici pa je sličnost uvjetno rečena, na korisnicima je ostavljeno da što preciznije opišu sadržaj svojega videa, što između ostalog i njima olakšava efikasno korištenje aplikacijom.

5.2. Oblikovanje modela

Započnimo sada s oblikovanjem modela pohrane naših podataka u *Redis* bazi podataka. Za početak promatramo način pospremanja informacija o korisniku. Nakon upoznavanja sa sintaksom i mogućnostima *Redisa*, brzo i jednostavno dolazimo do zaključka da je za opisivanje korisničkih informacija najpogodnija i najefikasnija struktura *Redis* hash. Napomena: običan tekst egzaktno je spremljen u bazu, dok je na mjestu [teksta] ubačena pripadna vrijednost.

5.2.1. Podaci o korisniku & podaci o broju registriranih korisnika

HASH

ključ: korisnik:[id]

vrijednost:

- username => [username]
- password => [password]
- ime => [ime]
- prezime => [prezime]
- slika => [slika]

Na ovaj način jednostavno identificiramo korisnika pomoću *id*-a dok čuvamo podatke o njemu u bazi. Preostaje pitanje kako novoregistriranom korisniku dodijeliti vrijednost *id*-a pa ćemo se poslužiti pomoćnom varijablom koja će pamtit broj registriranih korisnika. Kako ćemo kasnije vidjeti, postoji funkcija pomoću koje se vrijednost *Redis* stringa može povećati za određeni iznos pa je dodavanje novih korisnika izuzetno jednostavno.

STRING

ključ: *br_korisnika*

vrijednost: [broj registriranih korisnika]

5.2.2. Indeks korisnika

STRING

ključ: *indeks:korisnik:[username]*

vrijednost: [*id*]

Uočimo da korisniku nismo pridružili nikakvu informaciju o njegovim videozapisima. To je zato što možemo na vrlo jednostavan način odrediti implicitni model naše baze i na temelju njega odrediti pod kojim ključem su spremljeni podaci o videozapisima pojedinog korisnika. S druge strane, zanima nas što će biti struktura u koju ćemo postaviti podatke o videima. Kako je uobičajeno na ostalim platformama za dijeljenje videozapisa, npr. *YouTube*, prilikom prikaza rezultata pretraživanja omogućeno je sortiranje po vremenu objave i sl. Kako bismo ovo omogućili, odlučili smo se pospremati identifikatore svakog videozapisa korisnika u *sorted set* i pridružiti im *score* trenutak objave videozapisa te vrijednost *id* videozapisa. Na ovaj način imamo brzi pristup videozapisu korisnika, a kada pretražujemo videozapise omogućena nam je operacija unije koja nam automatski pruža dosljedan redosljed prikaza videozapisa.

5.2.3. Podaci o videozapisima korisnika s identifikatorom id_korisnika & podatak o indeksu videozapisa

ORDERED SET

ključ: korisnik:[id_korisnika]:video

score: [vrijeme objave videozapisa]

vrijednost: [id_videozapisa]

Identifikator videozapisa za korisnika na sličan način kao ranije zapisujemo pod ključ korisnik:[id]:br_video, gdje je *id* identifikator korisnika. Potreban nam je još indeks s nazivima videozapisa, kako bismo lagano mogli pretraživati po naslovu.

SET

ključ: indeks:video:[naziv_videozapisa]

vrijednost: korisnik:[id_korisnika]:video:[id_videozapisa]

Za kraj potrebno je još pospremiti podatke o pojedinom videozapisu, što je analogno zapisu korisnika.

5.2.4. Podaci o videozapisu

HASH

ključ: korisnik:[id_korisnika]: video:[id_videozapisa]

vrijednost:

- naziv => [naziv videozapisa]
- link => [link videozapisa na YouTube-u]
- opis => [opis videozapisa]

Već postaje jasno kako iskoristiti puni potencijal ovakvog načina indeksiranja ključeva. Ovisno o potrebi moguće je i dalje nastavljati gornji niz, ili dodati druge ključne riječi koje će modelirati neku dodatnu strukturu. U našem slučaju, to su *tag*-ovi.

5.2.5. Podaci o tag-ovima videozapisa *id_videozapisa* korisnika *id_korisnika*:

SET

ključ: korisnik:[id_korisnika]: video:[id_videozapisa]:tags

vrijednost: [nazi tag-a]

Kako želimo pretraživati po nazivu *tag*-ova, potreban nam je još jedan indeks.

5.2.6. Podaci o tag-ovima

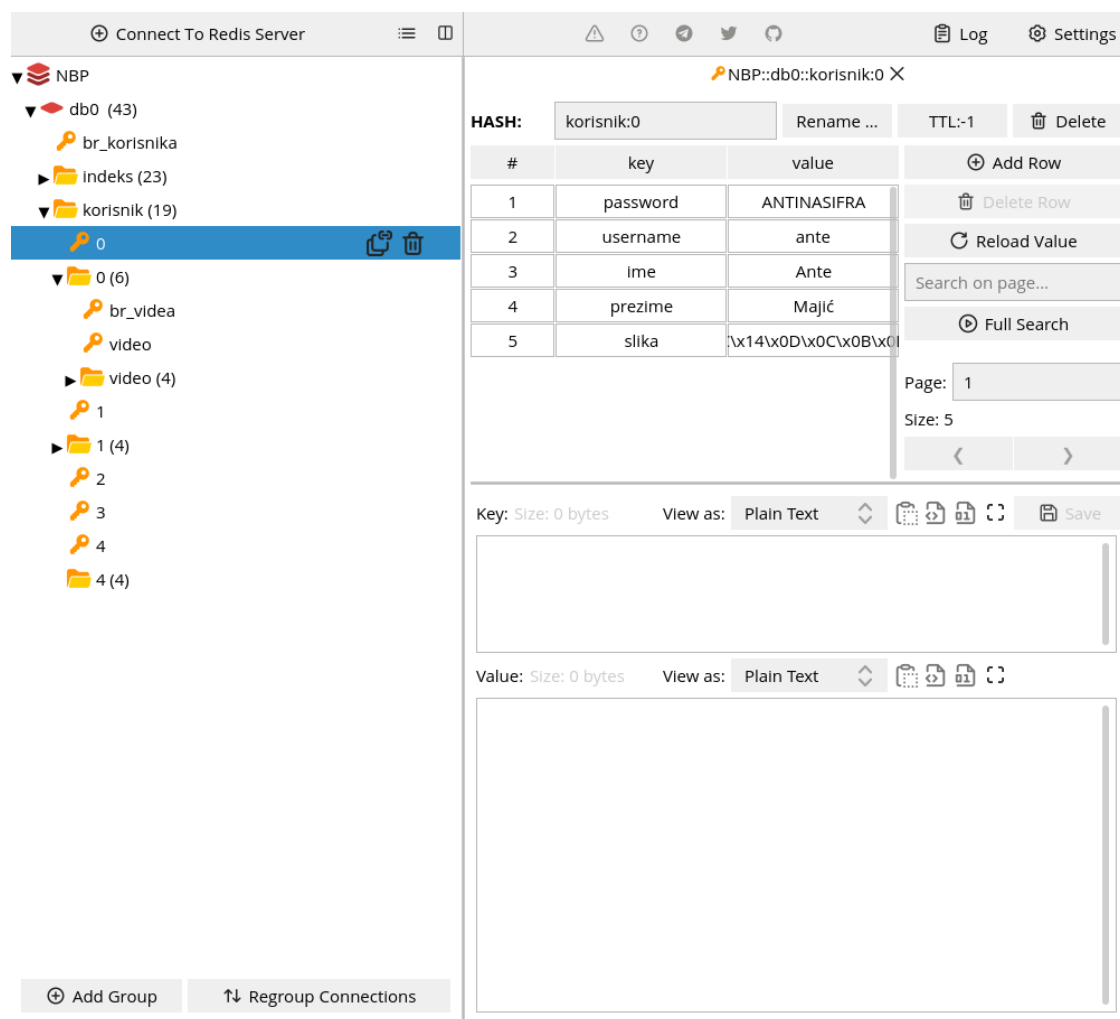
SET

ključ: indeks:tag:[naziv tag-a]

vrijednost: korisnik:[id_korisnika]:video:[id_videozapisa]

6. TEHNIČKI UVJETI

Bazu smo napravili na *Redis cloud* servisu [1] što nam je olakšalo ikakve moguće probleme sa samom bazom. Registracija je besplatna te se može dobiti baza s relativno malim spremištem, ali dovoljno za realizaciju ovakvog projekta. Znali smo da ćemo aplikaciju razvijati u PHP-u pa smo morali potražiti ekstenziju za upravljanje *Redis*-om iz njega. Pronašli smo API *PhpRedis* [3] koji vrlo jednostavno i smisleno implementira sve funkcije koje *Redis* omogućava. Pokretanje same aplikacije smo proveli na jednostavnom lokalnom *apache* serveru. Za jednostavan pregled sadržaja baze koristili smo se aplikacijom *resp.app* koja omogućava jednostavno manipuliranje sadržajem ključeva, vrijednosti, dodavanje istih i sl. Povezivanje na bazu vrlo je jednostavan postupak s navedenim API-jem i zahtjeva par linija koda s autentifikacijom.



Slika 9: resp.app

7. UPITI NA BAZU

7.1. Spremanje u bazu

Promotrimo sada koliko se jednostavno pozivaju funkcije za pospremanje podataka u bazu iz php-a. Metode kojima se koristimo u potpunosti ovise o vrsti strukture na koju ključ pokazuje. Postoje i neke generalne metode, kao npr. `exists`, koje olakšavaju neke stvari.

Promotrimo na primjeru registracije korisnika neke generalne metode i one vezane uz strukturu *Hash*. Metodom `exists` provjeravamo postojanje korisnika u bazi (tj. korisničkog imena). Registracija je omogućena samo ako ono već ne postoji, i tada se metodom `get` dohvaća broj registriranih korisnika koji se automatski povećava pozivanjem metode `incr`. Metodom `set` zatm postavljamo indeks za korisnika, a metodom `hMSet` postavljamo više vrijednosti unutar strukture *Hash* za korisnika. U našem slučaju to su naravno, `username`, `password`, `ime`, `prezime` i `slika`. Zanimljivo je to da sliku možemo pospremiti u *Redis* bazu u obliku stringa. Naime, ovo je moguće jer su *Redis*-ovi stringovi binarno sigurni.

```
if(!($r->exists("korisnik:".$this->username))){
    // odredi id korisnika
    $this->id = $r->get("br_korisnika");
    $r->incr("br_korisnika");

    // spremi ga u indeks
    $r->Set("indeks:korisnik:".$this->username, $this->id);

    // spremi podatke o korisniku
    $r->hMSet("korisnik:".$this->id, ["username" => $this->username,
    "password" => $this->password,
    "ime" => $this->ime, "prezime" => $this->prezime,
    "slika" => $this->slika]);
    return true;
}

return false;
```

Slika 10: Isječak koda registracije korisnika

Ubacivanje podataka o videozapisima obavlja se na vrlo sličan način, ali još nismo ni na koji način povezali korisnike s njihovim videozapisima. U modelu smo spomenuli da ćemo se koristiti *ordered set*-om, a metoda koju pozivamo kako bismo ubacili podatke u tu strukturu je `zAdd`.

```
$r->zAdd("korisnik:" . $id . ":video", $video->vrijeme, $brojvideo);
```

Slika 11: Dodavanje novog videozapisa

Ostali podaci o videozapisu koji se pospremaju u strukturu *hash* ubacuju se na analogan način kao i kod korisnika pa promotrimo još kako smo ubacili podatke o *tag*-ovima koji se, kao što znamo iz modela, ne nalaze unutar samog *hasha*. Znamo da svakom videozapisu pridružujemo skup *tag*-ova te još generiramo indeks radi lakšeg pretraživanja pomoću metode `sAdd`.

```
$r->sAdd("korisnik:" . $id . ":video:" . $brojvideo . ":tags", $tag);  
$r->sAdd("indeks:tag:" . $tag, "korisnik:" . $id . ":video:" . $brojvideo);
```

Slika 12: Dodavanje podataka o tagovima

Još nekolicina podataka spremjena je u obliku stringova, a njih se vrlo jednostavno definira pomoću metode `set`.

7.2. Dohvaćanje podataka iz baze

Vidjeli smo metode za postavljanje podataka u bazu, ali potrebe naše aplikacije su između ostaloga i prikazivanje tih podataka pa proučimo neke primjere dohvaćanja već spremljenih podataka.

Na slici 12. možemo vidjeti isječak iz funkcije koja provjerava *login* korisnika. Jasno, kao i ranije prvo provjeravamo postojanje navedenog ključa i zatim dohvaćamo sve podatke koji su spremljeni u bazi metodom `hMGet` (uočimo: svaka metoda *setanja* ima svoj par metodu *getanja*, u ovom slučaju to je postavljanje, odnosno dobavljanje više podataka iz *hasha*). Jednostavnom provjerom možemo utvrditi je li korisnik unio pravilan par podataka te ih spremamo u objekt koji ćemo postaviti u *session* varijablu koju nam omogućava PHP.

```

$this->id = $r->Get("indeks:korisnik:".$this->username);
if($r->exists("korisnik:".$this->id)){
    $podaci = $r->hMGet("korisnik:".$this->id,
        ["password", "ime", "prezime", "slika"]);

    if($this->password === $podaci["password"]){
        $this->ime = $podaci["ime"];
        $this->prezime = $podaci["prezime"];
        $this->slika = $podaci["slika"];
        return true;
    }
}
}

```

Slika 13: Isječak prijave korisnika

Dohvaćanje videozapisa je zanimljiviji postupak. Potrebno je iz *ordered set*-a dohvatiti sve indekse na kojima su pospremljeni videozapisi korisnika. Zatim laganom iteracijom po tim indeksima lagano pristupamo podacima o svakom videozapisu korisnika.

7.3. Pretraživanje

```

public function dohvati_name_vidoe($videoName)
{
    $r = DB::getConnection();
    $listaIndeksa = [];
    $listaVidea = [];

    $smece = array_pop($listaIndeksa);
    $smece = array_pop($listaVidea);

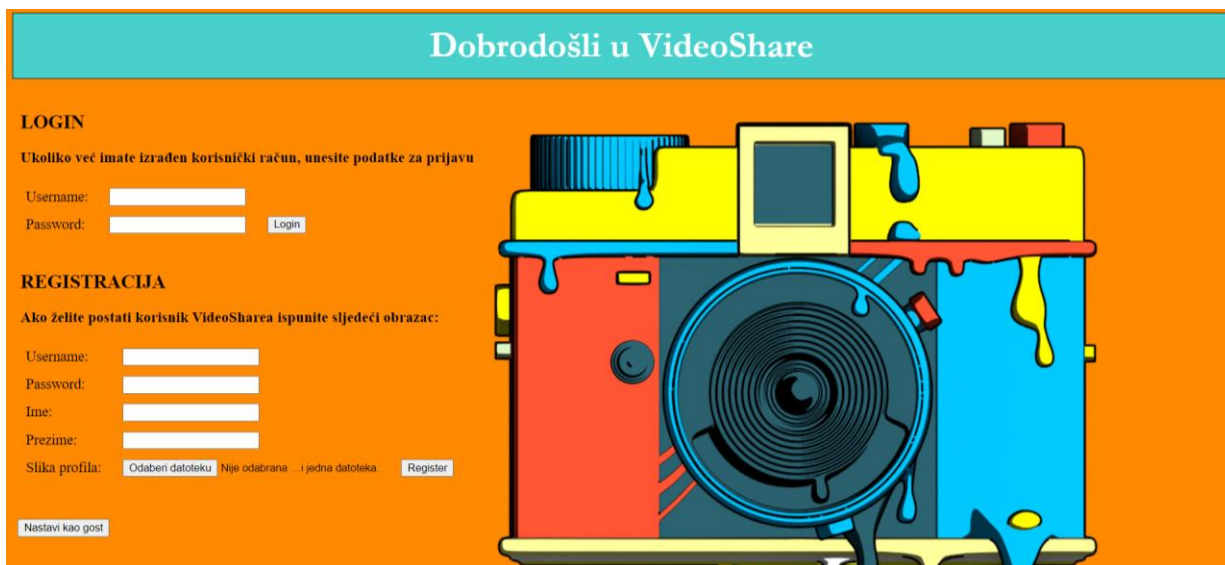
    $indeks = $r->smembers("indeks:video:" . $videoName);
    foreach($indeks as $in)
    {
        $video = $r->hvals($in);
        $listaVidea[] = $video;
    }
    return $listaVidea;
}

```

Slika 14: Funkcija dohvati_name_vidoe

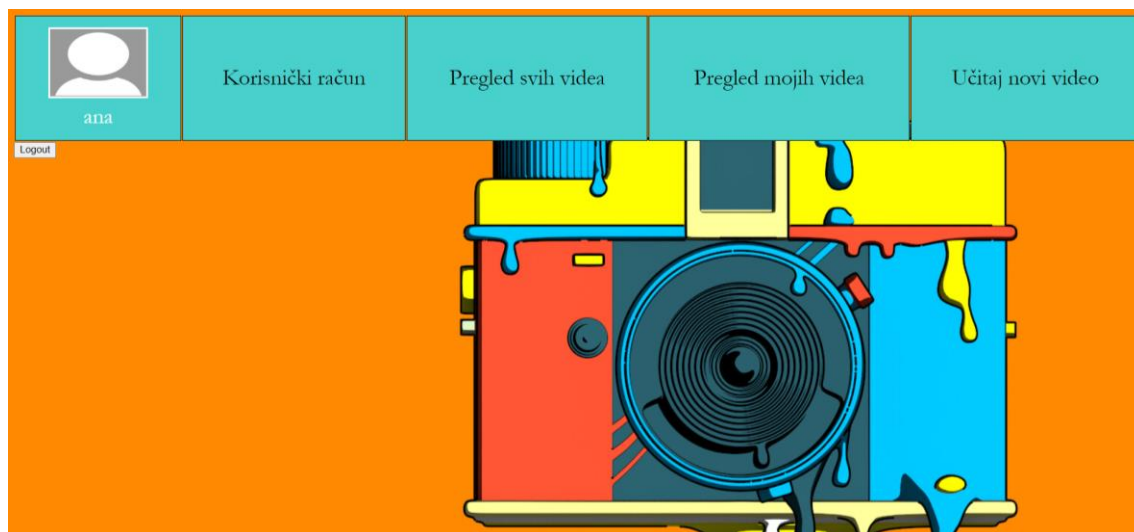
8. APLIKACIJA VideoShare

Promotrimo sada krajnji rezultat našeg istraživanja, aplikaciju *VideoShare* oblikovanu da komunicira s već objašnjenom bazom podataka, te korisnicima omogućuje jednostavno unošenje i dohvaćanje podataka. Za početak donosimo prikaz početne stranice koja sadrži formu za prijavu i registraciju. Korisniku je omogućeno korištenje aplikacije i u guest načinu rada, no pritom naravno ima smanjeni broj funkcionalnosti za korištenje u odnosu na registrirane korisnike. Konkretno, takvi korisnici imaju mogućnost pretraživati sve videozapise iz baze, ali nije im dozvoljeno unositi podatke u bazu, odnosno objavljivati svoje videozapise.



Slika 15: Početna stranica aplikacije VideoShare

Nakon uspješne registracije i logina, korisniku se prikazuje navigacijski izbornik pa promotrimo ponuđene opcije.



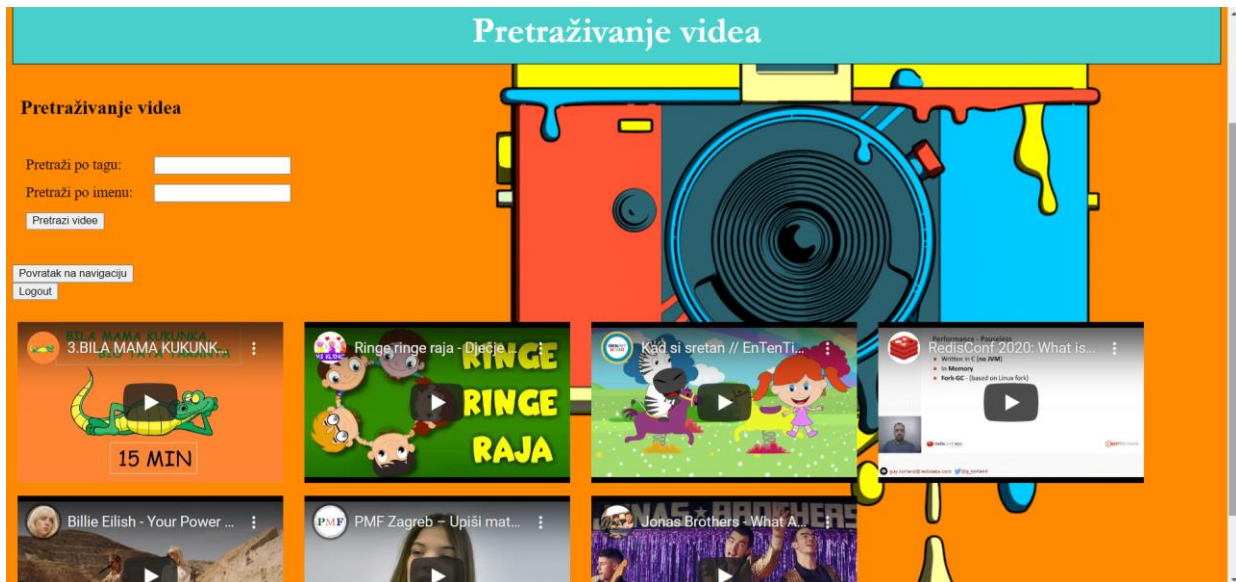
Slika 16: Navigacijski izbornik

Klikom na opciju korisnički račun, korisnik dobiva pregled svojih podataka pohranjenih u bazi podataka te mogućnost njihove izmjene. Također, vidimo da ukoliko korisnik pri registraciji ne postavi sliku po svome izboru, dobiva defaultnu sliku profila.

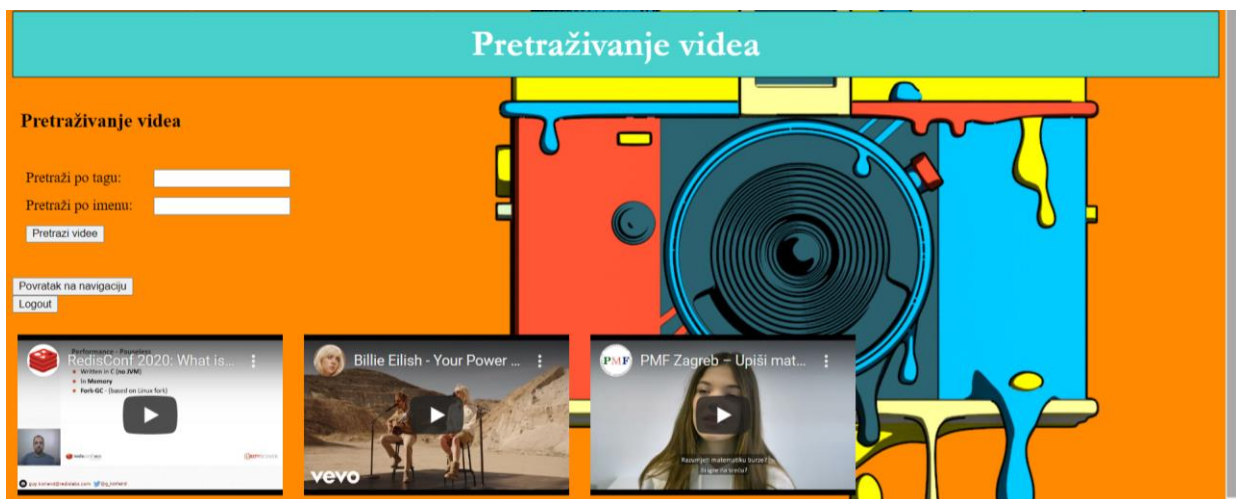


Slika 17: Otvaranje opcije 'Korisnički račun'

Prilikom odabira opcije 'Pregled svih videa' korisniku se prikažu svi videi spremljeni u bazi te dobiva dodatnu mogućnost pretraživanja pohranjenih videozapisa po tagu ili naslovu. Ukoliko odabere opciju 'Pregled mojih videa', dobiva podatke o svim videozapisima koje je sam uploadao, ponovno s mogućnosti da te podatke pretraži po naslovu ili tagovima.

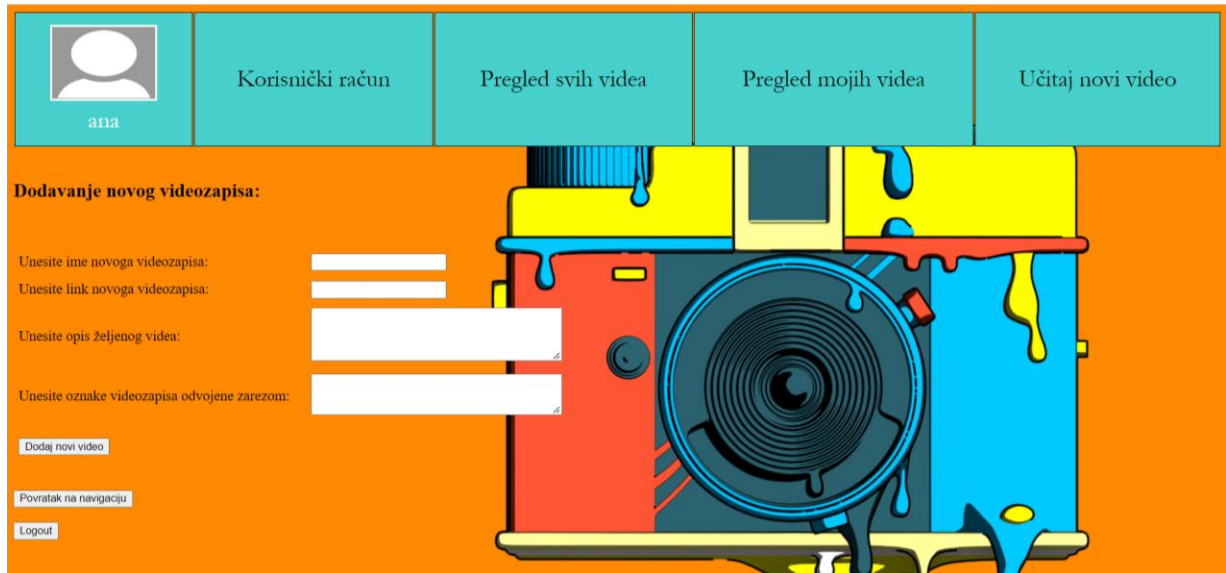


Slika 18: Pregled svih videa



Slika 19: Pregled mojih videa

Posljednja opcija navigacijskog izbornika korisniku omogućuje uploadanje novog videozapisa, odnosno njegovo spremanje u bazu podataka. Pritom se ispunjavaju podaci potrebi za kasnije efikasno pronalaženje videozapisa u bazi podataka.



Slika 20: Učitaj novi video

9. ZAKLJUČAK I IDEJE O POBOLJŠANJIMA

Baza Redis omogućava jednostavnu manipulaciju podacima, bilo to pospremanje ili dohvaćanje, te omogućava brojne operacije koje olakšavaju implementaciju ostatka aplikacije. Koristeći se mnogim strukturama, definirali smo model koji je bio prilagođen potrebama aplikacije koju smo razvili. Ako je bilo pitanje spremanja podataka koji sliče objektima, struktura hash nam je omogućila brzo ubacivanje takvih podataka, dok s druge strane, ako smo imali potrebu ubaciti dodatne strukture kako bismo olakšali određeno pretraživanje, jednostavnim manipulacijama ključeva uspjeli smo i taj problem riješiti. Razumijevanje mogućnosti pojedinih struktura, njihovih svojstava odnosno nedostataka, ključno je za razvoj baze koja će moći efikasno pristupiti podacima koji su potrebni. Dokumentacija je dovoljno kvalitetna da smo se uspjeli brzo prilagoditi jeziku baze i modelirati zadovoljavajuću bazu za naše potrebe. Pritom smo razmišljali o skalabilnosti našega rješenja pa smo morali dodati neke indekse kako bismo olakšali pretraživanje te ne bismo morali svakim pozivom povući sve podatke iz baze. Aplikacija koju smo izgradili dozvoljava jednostavno definiranje korisničkih podataka odnosno dodavanje videozapisa te pretraživanje i prikazivanje videozapisa ostalih korisnika.

Nismo se pretjerano brinuli o sigurnosti već smo prednost dali modeliranju same baze odnosno funkcionalnosti aplikacije. Svakako se lagano može dodati hashiranje lozinke kako bi se povećala sigurnost korisničkih podataka. Što se tiče funkcionalnosti, ovaj model lagano se može proširiti na raznim područjima. Struktura hash vrlo jednostavno dozvoljava dodefiniranje novih elemenata pa bismo mogli dodati neke dodatne informacije o korisnicima odnosno videozapisima, kao što je npr. *lajkanje* na Youtube-u. Također, moguće bi bilo dohvaćati podatke uživo tijekom pretraživanja, pa bi se rezultati prikazivali odmah na temelju samo početka naziva. Naravno kod većih aplikacija ovo bi možda bilo previše pa su svakako moguća dodefiniranja nekih struktura koje bi se mogle brinuti o predlaganju rezultata pretraživanja ili sl.

10.BIBLIOGRAFIJA

- [1] <https://app.redislabs.com/>
- [2] <https://redis.io/docs/>
- [3] <https://github.com/phpredis/phpredis>
- [4] <https://resp.app/>