Napredne arhitekture informacionih sistema

Baze podataka tipa ključ-vrednost

Izvođači nastave:
dr Marko Vještica
Elena Akik
Sanja Radić

DAIS
Data Science and Information Systems Group

Sadržaj

Uvod u baze podataka tipa ključ-vrednost

- Baze podataka tipa ključ-vrednost su vrsta *NoSQL* baza podataka koje skladište podatke u obliku **ključ-vrednost**
- Stekle su popularnost zbog **jednostavne strukture** koja omogućava **brzo izvršavanje operacija čitanja i pisanja**
- Svaki zapis u bazi podataka sastoji se od ključa i pridružene vrednosti
  - **Ključ** predstavlja **jedinstveni identifikator** koji ukazuje na vrednost
  - **Vrednosti** mogu biti **različitog tipa**
    - Poput tekstualnih ili numeričkih podataka, ali i kompleksnijih struktura poput lista, JSON dokumenata i slično
    - Različiti zapisi unutar iste baze podataka mogu imati različite tipove podataka kao vrednosti
    - Ovakva fleksibilnost omogućava strukturu podataka koja se prilagođava različitim potrebama softvera

Uvod u baze podataka tipa ključ-vrednost

- **Ne postoji poseban upitni jezik** za baze podataka tipa ključ-vrednost
  - Ali postoji mogućnost dodavanja, brisanja i dobavljanja parova ključ-vrednost
- Vrednosti se ne mogu pretraživati niti se mogu izvršavati kompleksni upiti
  - Podacima se pristupa **isključivo preko ključa**

Ključ	Vrednost
1	"Običan tekst"
2	0010101
3	{"A": 1, "B" : 2}

Različite baze podataka tipa ključ-vrednost

- Postoje **različite baze podataka** tipa ključ-vrednost:
  - Redis
  - Memcasched
  - Amazon DynamoDB (zasnovano na *Cloud*-u)
  - Riak
  - …

Sadržaj

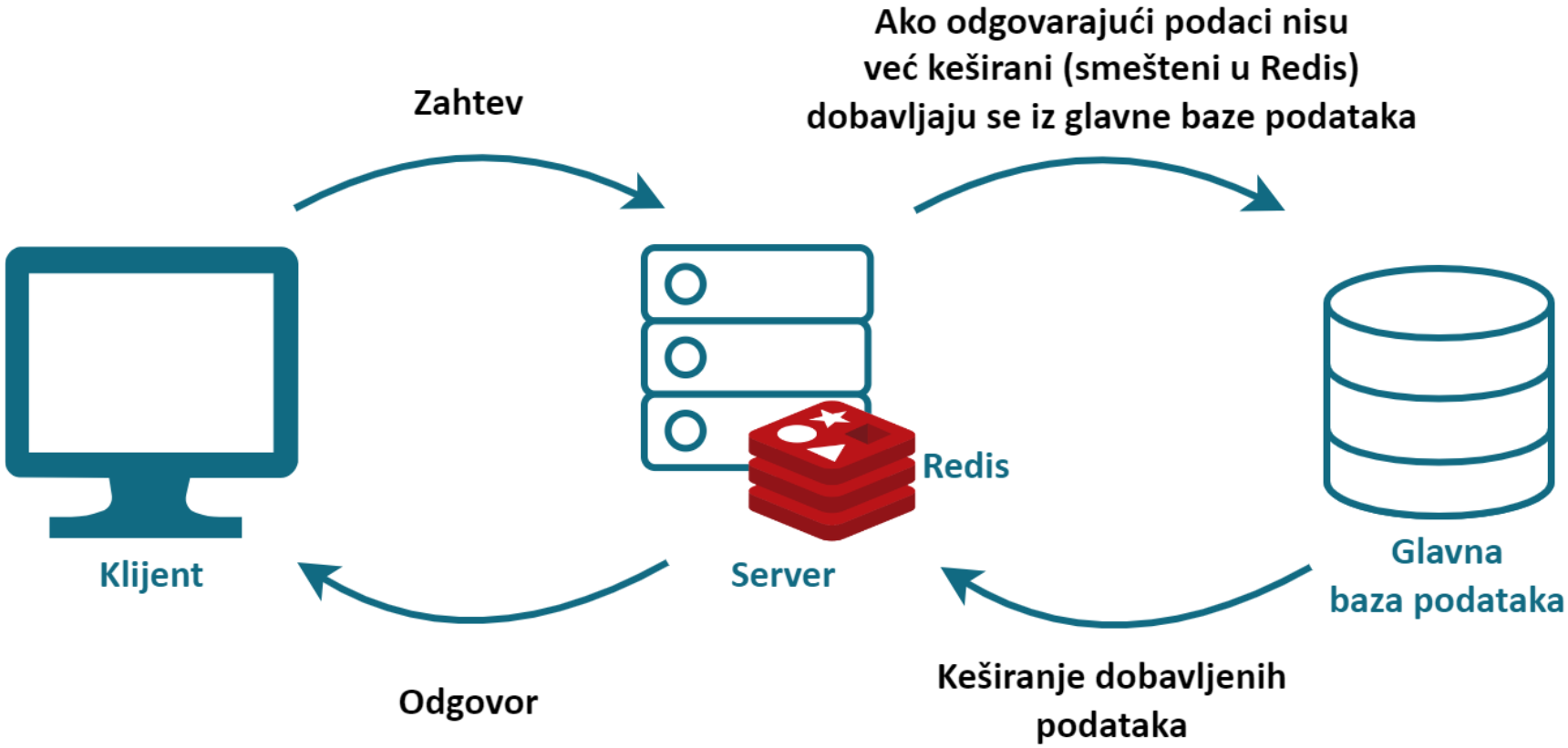
Uvod u *Redis*

- ***Redis*** (engl. ***Re**mote **Di**ctionary **S**erver*) je višenamenska baza podataka (skladište, engl. *Data Store*) koja se, između ostalog, koristi za skladištenje podataka tipa ključ-vrednost, keširanje podataka i kao posrednik u razmeni poruka različitih protokola (engl. *Message Broker*)
- Svi podaci se nalaze u **radnoj memoriji** (engl. *In-Memory Database*) što omogućava brz pristup podacima jer ne zahteva dobavljanje podataka sa diska
  - Rezultat ovakvog pristupa su izuzetne performanse čitanja i pisanja
- Zbog dobrih performansi, mnoge kompanije kao što su *Github* i *Instagram* odlučuju se za ovu bazu podataka

Kada i kako se najčešće koristi *Redis?*

- *Redis se n*ajčešće koristi kao **pomoćna baza podataka** radi **keširanja podataka** iz glavne baze podataka (npr. relacione baze podataka)
  - Na ovaj način brže se pristupa podacima koji se često dobavljaju
- Nakon korisničkog zahteva za određenim podacima, ***Redis* server** vrši proveru postojanja takvih podataka u kešu
- Postoje **dva scenarija**:
  - Zatraženi podaci su dostupni u kešu *(**cashe hit**)*
    1. Dobavljaju se iz pomoćne baze podataka *(Redis)*
  - Zatraženi podaci nisu dostupni u kešu *(**cashe miss**)*
    1. Dobavljaju se iz glavne baze podataka
    2. Upisuju se (keširaju se) u memoriju pomoćne baze podataka kako bi bili dostupni za naredne zahteve
    3. Dobavljaju se iz pomoćne baze podataka *(Redis)*

Kako se najčešće koristi *Redis?*



Tipovi podataka u bazi podataka *Redis*

- **Ključ** je tekstualni podatak
- **Vrednost** može biti sledećeg tipa:
  - Tekstualna vrednost (*String*)
  - Lista *(List)*
  - Kolekcija jedinstvenih tekstualnih vrednosti bez redosleda *(Set)*
  - Uređeni niz *(Ordered set)* – prati redosled na osnovu neke vrednosti koja se može ponavljati, od manje ka većoj
  - Heš *(Hash)* – mapiranje tekstualne vrednosti ključa i tekstualnog podatka
  - tipovi podataka zasnovani na tekstualnoj vrednosti, ali sa sopstvenom semantikom (*Bitmap* i *HyperLogLog*)
  - …

Tipovi podataka u bazi podataka *Redis*

"Zdravo svete!"	*String*
[A -> B -> C -> D]	*List*
{A, B, C, D}	*Set*
{A: 0.1, B: 0.5, C: 20, D: 150}	*Ordered set*
{A : "Pera" , B: "Savo", C: "Milica"}	*Hash*
00110101011001110010101010	*Bitmap*
11001010 11100011 10101010	*HyperLogLog*

Korišćenje baze podataka *Redis*

- Najjednostavniji način za korišćenje *Redis* servera je preko ***Docker* slike**:

```
> docker run -d --name <naziv_kontejnera> -p <lokalni_port_bp>:6379
  redis/redis-stack-server
```

- ***RedisInsight*** je klijentska aplikacija koju je moguće pokrenuti uz *Redis* server:

```
> docker run -d --name <naziv_kontejnera> -p <lokalni_port_bp>:6379 –p
  <port_za_RedisInsight>:8001 redis/redis-stack
```

Korišćenje baze podataka *Redis*

- Pristupanje *Redis* klijentu preko terminala:

```
> docker exec -it <naziv_kontejnera> redis-cli
```

- Klijentskoj aplikaciji moguće je pristupiti na:
  - http://localhost:<port_za_RedisInsight>

- Dodatne informacije nalaze se na sledećem linku:
  - https://redis.io/docs/install/install-stack/docker/

Sadržaj

Provera rada servera *Redis*

- Nakon pristupa klijentu *redis-cli*, moguće je pisati *Redis* naredbe iz terminala
- Provera rada servera *Redis* moguća je pomoću naredbe ***PING***

```
> PING
PONG
```

- Naredba ***QUIT*** koristi se za prekid rada sa klijentom *Redis*

```
> QUIT
```

- Naredbe koje se navode nisu osetljive na velika i mala slova

Kreiranje novog zapisa

- Pomoću naredbe ***SET*** moguće je kreirati novi zapis (slog)

```
> SET <ključ> <vrednost> [NX | XX] [GET] [EX sekunde | PX milisekunde | EXAT
  unix-time-seconds | PXAT unix-time-milliseconds] [KEEPTTL]
```

  - **NX** – kreiranje zapisa samo ukoliko ne postoji zapis sa istom vrednošću ključa
  - **XX** – kreiranje zapisa samo ukoliko postoji zapis sa istom vrednošću ključa (stara vrednost će biti zamenjena novom)
  - **GET** – stara vrednost će biti prikazana, ili će biti prikazano *(nil)* ukoliko ključ nije postojao
  - **EX, PX, EXAT, PXAT** – postavljanje specifičnog vremena nakon kojeg zapis više ne postoji
  - **KEEPTTL** – pri promeni vrednosti, zadržava se vreme života zapisa

- **Napomena:** ukoliko se ne navede NX ili XX i ukoliko postoji zapis sa istim ključem, stara vrednost će biti zamenjena novom

Kreiranje novog zapisa – primeri

- Kreiranje zapisa sa ključem „k1“ i vrednošću „Zdravo!“

```
> SET k1 "Zdravo!"
OK
```

- Kreiranje zapisa sa ključem „k1“ i vrednošću „Dobro jutro!“, ali samo ukoliko ne postoji zapis sa istim ključem

```
> SET k1 "Dobro jutro!" NX
(nil)
```

Kreiranje novog zapisa – primeri

- Kreiranje zapisa sa ključem „k1“ i vrednošću „Dobro jutro!“, ali samo ukoliko postoji zapis sa istim ključem

```
> SET k1 "Dobro jutro!" XX
OK
```

- Kreiranje zapisa sa ključem „k2“ i vrednošću „istice za minut“ koji će postojati 60 sekundi

```
> SET k2 "istice za minut" EX 60
OK
```

Kreiranje novog zapisa – zadaci

- Kreirati zapis sa ključem „k4“ i vrednošću „Dobro vece!“

```
> SET k4 "Dobro vece!"
OK
```

- Kreirati zapis sa ključem „k4“ i vrednošću „Laku noc!“. Ukoliko već postoji vrednost sa istim ključem, zameniti je novom vrednošću i ispisati staru vrednost

```
> SET k4 "Laku noc!" GET
"Dobro vece!"
```

Kreiranje novog zapisa sa životnim vekom

- Kreiranje zapisa sa životnim vekom izraženim u sekundama moguće je i pomoću naredbe ***SETEX***

```
> SETEX <ključ> <sekunde> <vrednost>
```

- Provera za koliko sekundi će zapis biti obrisan moguće je pomoću naredbe ***TTL***

```
> TTL <ključ>
```

Kreiranje novog zapisa sa životnim vekom – primeri

- Kreiranje zapisa sa ključem „k3“ i vrednošću „istice za 30 sekundi“ koji će postojati 30 sekundi

```
> SETEX k3 30 "istice za 30 sekundi"
OK
```

- Proveriti za koliko sekundi se zapis sa ključem „k3“ briše

```
> TTL k3
(integer) 23
```

Dobavljanje vrednosti za ključ

- Dobavljanje zapisa sa određenim ključem moguće je pomoću ***GET*** naredbe

```
> GET <ključ>
```

- Dobaviti vrednost zapisa sa ključem „k1“

```
> GET k1
  "Dobro jutro!"
```

Dobavljanje ključeva na osnovu obrasca

- Vrednosti ključeva koji odgovaraju određenom obrascu moguće je dobaviti pomoću naredbe ***KEYS***

```
> KEYS <obrazac>
```

- Specijalni karakteri:
  - ? – tačno jedan karakter
  - * – 0 ili više karaktera
  - [xy] – x ili y
  - [^x] – sve sem x
  - [x-y] – karakteri od x do y (po abecedi)
- **Napomena:** ovu naredbu koristiti sa oprezom, jer može uticati na performanse izvršavanja upita u bazama podataka sa velikom količinom podataka
  - Retko se koristi u aplikativnim rešenjima

Dobavljanje ključeva na osnovu obrasca – primeri

- Izlistati sve ključeve

```
> KEYS *
  1) "k1"
  2) "k4"
```

- Izlistati ključeve koji počinju na slovo „k“

```
> KEYS [k]*
  1) "k1"
  2) "k4"
```

Dobavljanje ključeva na osnovu obrasca – zadaci

- Izlistati ključeve koji imaju tačno 2 karaktera

```
> KEYS ??
  1) "k1"
  2) "k4"
```

- Izlistati ključeve koji se ne završavaju karakterom „4“

```
> KEYS *[^4]
  1) "k1"
```

Brisanje zapisa

- Brisanje zapisa na osnovu njihovih ključeva moguće je izvršiti pomoću naredbe ***DEL***
  - Rezultat izvršavanja naredbe je broj obrisanih zapisa

```
> DEL <ključ1> [<ključ2> … <ključN>]
```

- Brisanje zapisa čiji su ključevi „k1“ i „k2“

```
> DEL k1 k2
(integer) 1
```

Napomena: ključ „k2“ ne postoji, stoga će biti obrisan samo zapis čiji je ključ „k1“

Kreiranje zapisa tipa *Hash*

- Za jednu vrednost ključa moguće je dodati više vrednosti koje simuliraju vrednosti objekta korišćenjem heš struktura
- Za kreiranje ovakvog zapisa koristi se naredba ***HSET***

```
> HSET <ključ> <polje1> <vrednost1> [<polje2> <vrednost2> ...]
```

Kreiranje zapisa tipa *Hash* – primer

- Kreirati zapis sa ključem "osoba_id:1" i postaviti vrednost "Milica" za polje *ime,* vrednost "Savic" za polje *prezime* i vrednost "Subotica" za polje *mesto_rodjenja*

```
> HSET osoba_id:1 ime "Milica" prezime "Savic" mesto_rodjenja "Subotica"
(integer) 3
```

- Zapisu sa ključem "osoba_id:1" dodati polje *godine* i postaviti vrednost na 25

```
> HSET osoba_id:1 godine 25
(integer) 1
```

Kreiranje zapisa tipa *Hash* – zadatak

- Kreirati zapis sa ključem "osoba_id:2" i postaviti vrednosti za polja *ime, prezime, mesto_rodjenja, godine* i *datum_rodjenja* tako da odgovaraju podacima studenta.

Dobavljanje naziva i vrednosti polja tipa *Hash*

- Za dobavljanje svih naziva i vrednosti polja zapisa koristi se naredba ***HGETALL***

```
> HGETALL <ključ>
```

- Za dobavljanje vrednosti određenog polja zapisa koristi se naredba ***HGET***

```
> HGET <ključ> <polje>
```

- Za dobavljanje vrednosti više polja zapisa koristi se naredba ***HMGET***

```
> HMGET <ključ> <polje1> [<polje2> ...]
```

Dobavljanje naziva i vrednosti polja – primer

- Dobavljanje svih naziva i vrednosti polja zapisa čiji je ključ „osoba_id:1“

```
> HGETALL osoba_id:1
  1) "ime"
  2) "Milica"
  3) "prezime"
  4) "Savic"
  5) "mesto_rodjenja"
  6) "Subotica"
  7) "godine"
  8) "25"
```

- Dobavljanje vrednosti polja *ime* za zapis čiji je ključ „osoba_id:1“

```
> HGET osoba_id:1 ime
  "Milica"
```

Dobavljanje naziva i vrednosti polja – primer

- Dobavljanje vrednosti polja *ime* i *prezime* za zapis čiji je ključ „osoba_id:1“

```
> HMGET osoba_id:1 ime prezime
  1) "Milica"
  2) "Savic"
```

Dobavljanje naziva i vrednosti polja tipa *Hash*

- Za dobavljanje naziva svih polja zapisa koristi se naredba ***HKEYS***

```
> HKEYS <ključ>
```

- Za dobavljanje vrednosti svih polja zapisa koristi se naredba ***HVALS***

```
> HVALS <ključ>
```

Dobavljanje naziva i vrednosti polja – primer

- Dobavljanje naziva svih polja zapisa čiji je ključ „osoba_id:1“

```
> HKEYS osoba_id:1
  1) "ime"
  2) "prezime"
  3) "mesto_rodjenja"
  4) "godine"
```

- Dobavljanje vrednosti svih polja zapisa čiji je ključ „osoba_id:1“

```
> HVALS osoba_id:1
  1) "Milica"
  2) "Savic"
  3) "Subotica"
  4) "25"
```

Brisanje polja tipa *Hash*

- Za brisanje polja zapisa koristi se naredba ***HDEL***

```
> HDEL <ključ> <polje1> [<polje2> ...]
```

- Brisanje polja *godine* za zapis čiji je ključ "osoba_id:1"

```
> HDEL osoba_id:1 godine
(integer) 1
> HGET osoba_id:1 godine
(nil)
```

Brisanje svih zapisa

- Brisanje svih zapisa moguće je izvršiti pomoću naredbe ***FLUSHALL***

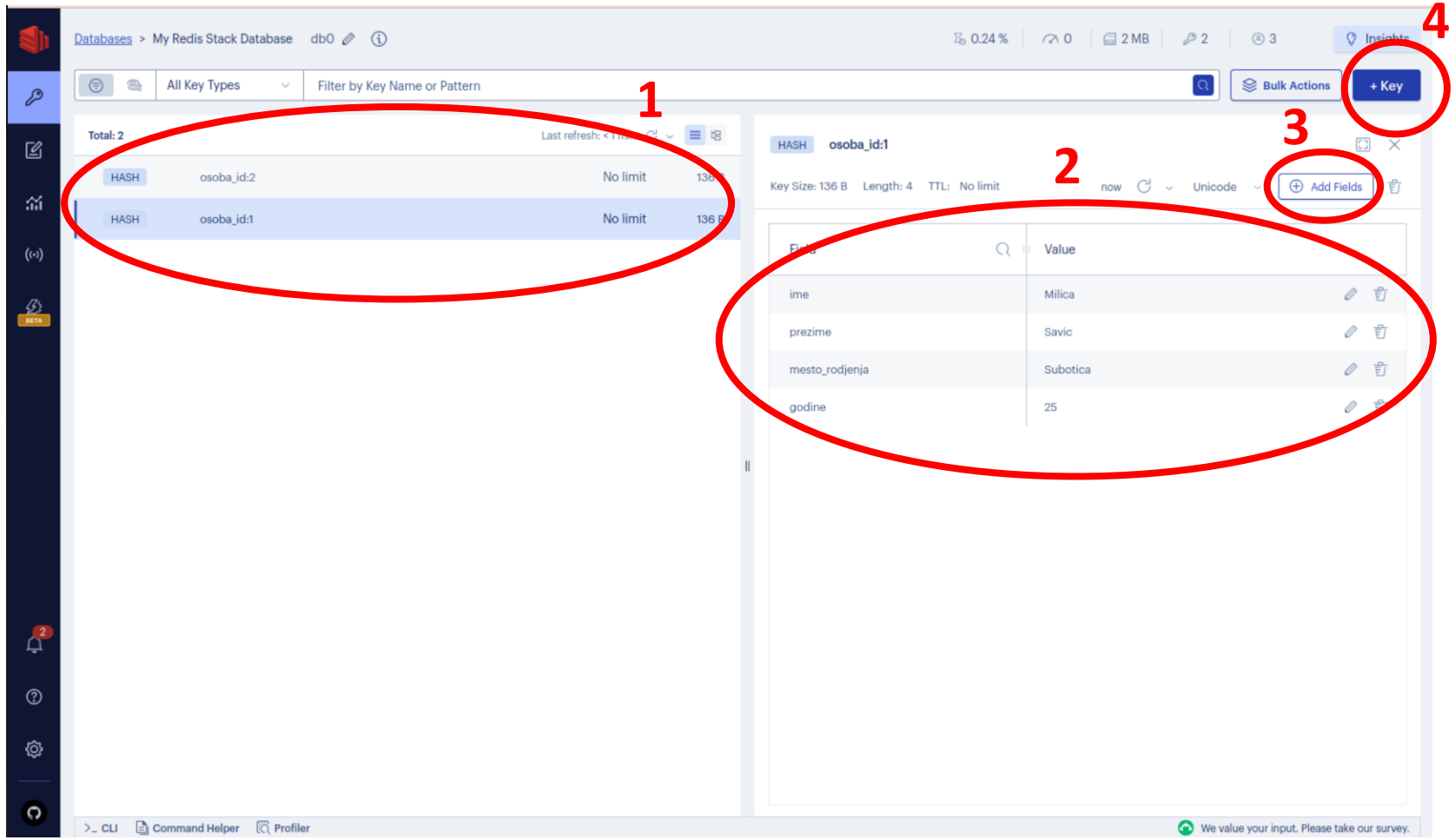
```
> FLUSHALL
```

- Brisanje svih zapisa

```
> FLUSHALL
OK
> KEYS *
(empty array)
```

Sadržaj

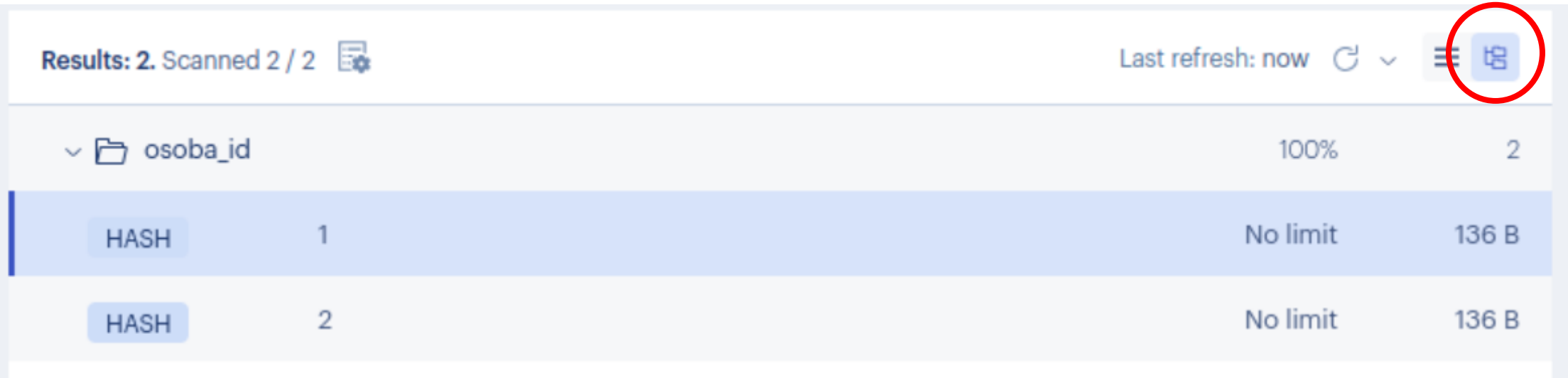
Rad sa klijentskom aplikacijom *RedisInsight*



Legenda:

1. Pregled zapisa u bazi podataka
2. Pregled vrednosti zapisa
3. Dodavanje novog polja
4. Dodavanje novog zapisa

Rad sa klijentskom aplikacijom *RedisInsight*



- Ukoliko su ključevi zapisa u formatu *naziv:broj* i promeni se prikaz zapisa u bazi podataka, biće klasifikovani po nazivu ključa

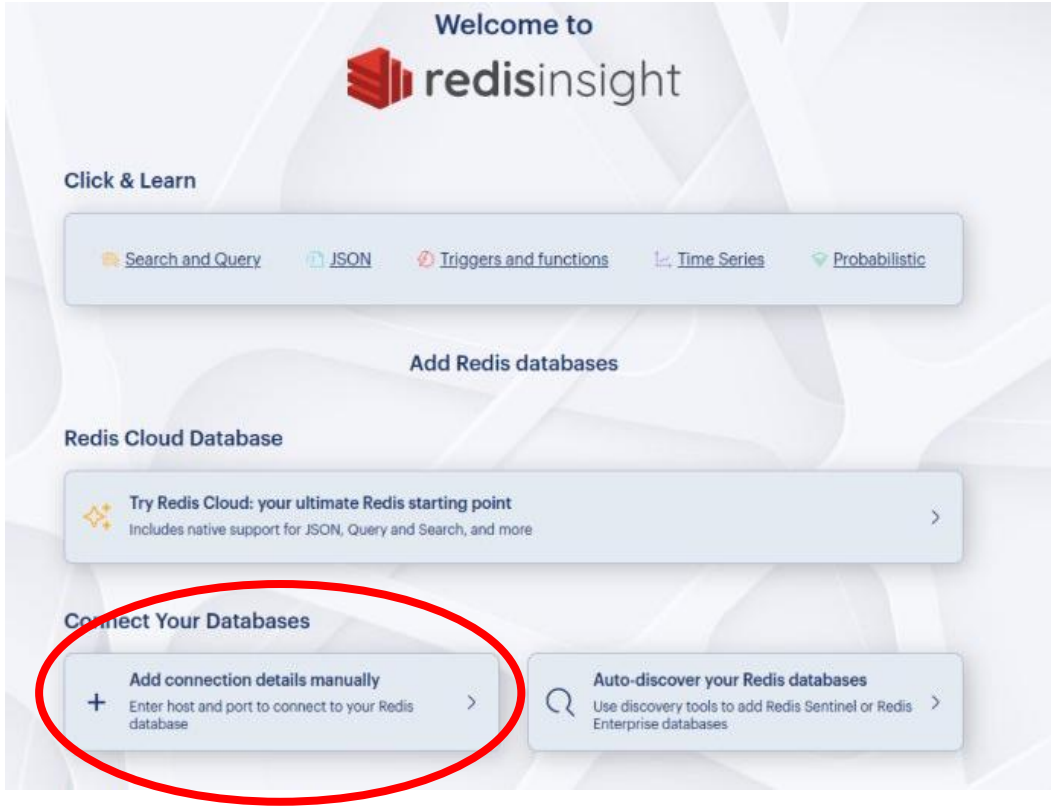
Sadržaj

Primer aplikacije *Spring* sa keširanjem pomoću *Redis*-a

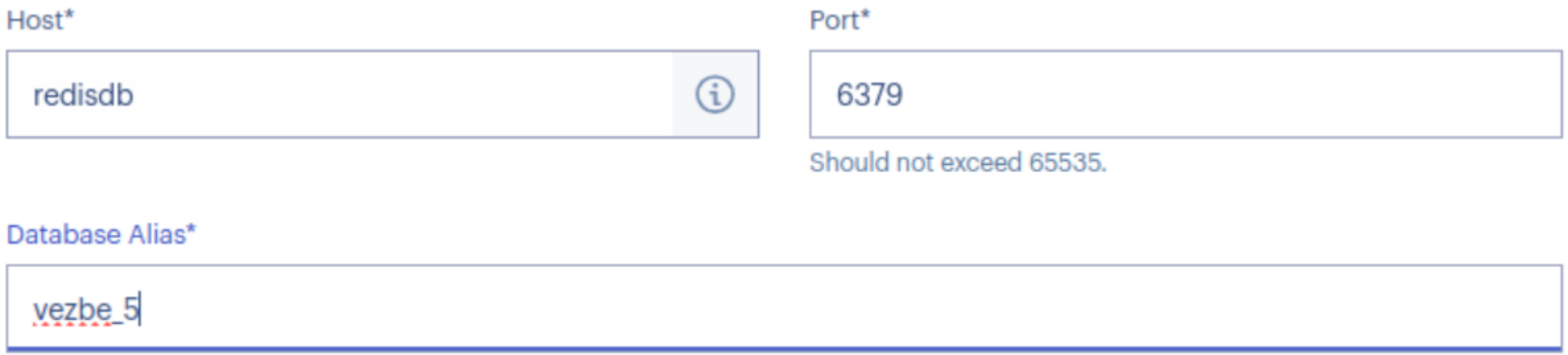
- Projekat se nalazi na acs-u u okviru repozitorijuma
- U okviru aplikacije nalazi se datoteka *Docker compose*
- Nakon pokretanja aplikacije pomoću datoteke *Docker compose*, klijentskoj aplikaciji je moguće pristupiti na linku:
  - http://localhost:5540/

Rad sa klijentskom aplikacijom *RedisInsight*

- Neophodno je uneti konekcione parametre ručno



Rad sa klijentskom aplikacijom *RedisInsight*



Postavljanje projekta

- Za potrebe korišćenja baze podataka *Redis* neophodno je dodati zavisnost u datoteku *pom.xml*:

```
1  <dependency>
2    <groupId>org.springframework.boot</groupId>
3    <artifactId>spring-boot-starter-data-redis</artifactId>
4  </dependency>
```

- Navođenje parametara za povezivanje na server *Redis* u okviru datoteke *application.properties*:

```
1  redis.host = localhost
2  redis.port = 6379
```

Konfiguracija baze podataka *Redis*

- Server *Redis* moguće je konfigurisati u skladu sa potrebama aplikacije
- Primer konfiguracije nalazi se u datoteci *RedisConfiguration.java*

@EnableCaching

- Anotacija ***@EnableCaching*** omogućava keširanje u okviru servisa
- Moguće je obezbediti keširanje i na nivou aplikacije navođenjem anotacije *@EnableCaching* iznad klase *main*

```
1 @Service
2 @EnableCaching
3 public class UserService {
4
5 }
```

@Cacheable

- Anotacija ***@Cacheable*** označava da će se povratne vrednosti metode dobavljati iz keša ukoliko postoje ili iz repozitorijuma ukoliko ne postoje u kešu
  - U slučaju dobavljanja vrednosti iz repozitorijuma, dodaje se odgovarajući zapis u keš i pri sledećem pozivu se dobavlja iz keša
- Parametri:
  - **Value** – naziv keša u koji se zapis smešta
  - **Key** – vrednost ključa – pristupa se parametrima funkcije navođenjem '#'
    - Moguće je pristupanje obeležjima parametra (npr. #user.id) i poziv metoda (npr. #user.getName())
  - **Condition** – dodatni uslov koji mora biti ispunjen da bi se metoda pozvala iz keša (preduslov)
  - **Unless** – uslov koji sprečava keširanje rezultata metode ukoliko je ispunjen

```
1 @Cacheable(value = "users", key = "#userId", condition = "#userId != null")
2 public String greetUser(Long userId) {
3
4 }
```

@CachePut

- Anotacija ***@CachePut*** omogućava smeštanje povratnih vrednosti u keš
  - Metoda se uvek izvrši, bez obzira na postojanje keša za navedeni ključ
  - Ažuriraju se vrednosti i u glavnoj bazi (repozitorijumu) i u kešu

```
1 @CachePut(value = "users", key = "#userId + '-' + #newFirstName")
2 public User updateUser(Long userId, String newFirstName) throws Exception {
3
4 }
```

@CacheEvict

- Anotacija ***@CacheEvict*** briše zapise iz keša za prosleđeni ključ

```
1 @CacheEvict(value = "users", key = "#userId")
2 public void deleteUser(Long userId) {
3
4 }
```

Sadržaj

Korisni linkovi

- Zvanična dokumentacija *Redis*
  - *https://redis.io/docs/*
- Naredbe u klijentu *Redis*
  - https://redis.io/commands/

Napredne arhitekture informacionih sistema

Baze podataka tipa ključ-vrednost

Pitanja?

Izvođači nastave:
dr Marko Vještica
Elena Akik
Sanja Radić

DAIS
Data Science and Information Systems Group