

Universidade de Aveiro, DETI

Complementos de Bases de Dados

Guião das aulas práticas

LEI – Licenciatura em Engenharia Informática

Ano: 2024/2025

# Lab 1 Bases de Dados Chave-Valor

## Objetivos

Os objetivos deste trabalho são:

- Compreender os fundamentos das bases de dados de chave-valor.
- Instalar e utilizar uma solução de código aberto.
- Desenvolver soluções para diversos casos de uso

## Nota prévia

Este módulo deverá ser preferencialmente desenvolvido em Linux. Caso pretenda usar Windows verifique as notas sobre compatibilidade do software que irá usar.

Submeta o código/resultados/relatórios no elearning. Utilize uma pasta (1, 2, ..) para cada exercício, compactadas num único ficheiro.

Bom trabalho!

## 1.1 REDIS

Redis (*REmote DIctionary Service*) foi desenvolvido em 2009 e tem sofrido atualizações sucessivas, sendo um dos mais populares repositórios do tipo chave-valor em memória. Pode ser utilizado como base de dados, como *cache* de dados, ou como sistema de mensagens (*message broker*).

- a) Instale o Redis no seu computador pessoal (<https://redis.io/>). Execute o servidor (*ou passe à alínea b caso fique instalado como serviço do SO*):

```
$ redis-server
```

- b) Estude o funcionamento do sistema testando os comandos mais usados, através de linha de comandos.

```
$ redis-cli
```

- c) Consulte os slides disponibilizados para a disciplina (*CBS\_05\_KeyValue*) e sítios web com documentação sobre o Redis. Alguns exemplos:

- *Redis web site*, <https://redis.io>
- *Redis Java*, [https://www.tutorialspoint.com/redis/redis\\_java.htm](https://www.tutorialspoint.com/redis/redis_java.htm)
- *Redis Tutorial - w3resource*, <http://www.w3resource.com/redis/index.php>

Alguns conceitos para os quais deve ter particular atenção:

- Escrita e leitura, persistência, TTL, tipos
- Estruturas (Hash, List, Set, Sorted Set, Streams)
- Operações sobre estruturas (ranges, unions, intersections, subtractions)
- ACLs, Transações e Namespace

No final deste exercício, no seu diretório raiz encontrará um ficheiro com o nome ".rediscli\_history". Copie-o para outro ficheiro com o nome "CBD-11-<NMEC>.txt" (onde <NMEC> deve ser substituído pelo seu nº mecanográfico).

Deverá entregar os ficheiros:

CBD-11-<NMEC>.txt – com os comandos usados no redis-cli.

## 1.2 Redis – Inserção massiva

O Redis permite inserir dados a partir de um ficheiro, através de linha de comandos. Para testar esta funcionalidade, escreva no ficheiro CBD-12-batch.txt um conjunto de comandos de redis (entre 10 a 20). De seguida, utilize-o para submeter estes comandos ao servidor redis.

Deverá entregar os ficheiros:

CBD-12-batch.txt

CBD-12.txt – comando(s) usados para carregar o ficheiro no Redis

## 1.3 Redis – Acesso programático

- a) Instale um driver de Redis para Java (e.g. [Jedis](https://redis.io/clients), disponível em <https://redis.io/clients>) e crie um pequeno programa para ligação ao servidor Redis, repetindo algumas das operações anteriores. Note que terá de usar o ficheiro *jar* diretamente ou através de um projeto *maven*.

Use como base o exemplo seguinte (*baseado em Jedis*):

```
package redis.ex3;
import redis.clients.jedis.Jedis;

public class Forum {
    public static void main(String[] args) {
        // Ensure you have redis-server running
        Jedis jedis = new Jedis();
        System.out.println(jedis.ping());
        System.out.println(jedis.info());
        jedis.close();
    }
}
```

- b) Crie um programa que escreva e leia usando i) uma lista e ii) um hashmap. Tome como base o exemplo seguinte que escreve e lê sobre um Set.  
(Nota: quando mudar entre tipos, garanta que usa nomes diferentes para as para evitar colisões entre tipos).

```
package redis.ex3;
import redis.clients.jedis.Jedis;

public class SimplePost {
    public static String USERS_KEY = "users"; // Key set for users' name

    public static void main(String[] args) {
        Jedis jedis = new Jedis();
        // some users
        String[] users = { "Ana", "Pedro", "Maria", "Luis" };
    }
}
```

```

        // jedis.del(USERS_KEY); // remove if exists to avoid wrong type
        for (String user : users)
            jedis.sadd(USERS_KEY, user);
        jedis.smembers(USERS_KEY).forEach(System.out::println);
        jedis.close();
    }
}

```

## 1.4 Autocomplete

- a) Usando o servidor Redis e o driver cliente (Jedis) crie um programa para fornecer uma lista de termos, ordenados por ordem alfabética, para uma função de *autocomplete*. Use o ficheiro "names.txt" como base para os termos a procurar. A interação final com o utilizador pode ser a seguinte:

```

Search for ('Enter' for quit): susann
susann
susanna
susannah
susanne

Search for ('Enter' for quit): zora
zora
zorah
zorana

```

- b) O ficheiro "nomes-pt-2021.csv" contém uma lista dos nomes pessoais e o total de registos que foram registados em 2021, Desenvolva uma variante da alínea anterior onde o resultado da pesquisa para *autocomplete* é ordenado por popularidade decrescente do nome.

Deverá entregar os ficheiros:

- \*.java – dos dois exercícios
- CBD-14a-out.txt – exemplos de interação e de output do programa a)
- CBD-14b-out.txt – exemplos de interação e de output do programa b)

## 1.5 Sistema de Atendimento

- a) Construa um sistema de atendimento em Java usando Redis e o driver cliente (Jedis). Este deverá permitir que determinado utilizador (*username*) peça um número máximo (*limit*) de produtos (*product*) por unidade de tempo (*timeslot*). O sistema deve ir registando os pedidos (*username*, *product*), não atendendo pedidos de novos produtos se já tiver excedido o limite máximo definido para a janela temporal. Caso exceda o limite, o sistema deve produzir uma mensagem de erro. O *limit* e *timeslot* são valores definidos para todo o sistema.

Exemplo: O sistema atende, para cada utilizador, um máximo de 30 produtos diferentes a cada 60 minutos.

- b) Altere o programa para que cada pedido possa ter uma quantidade (*quantity*) associada, i.e. as unidades de produto solicitados no pedido. Nesta versão, o limite passa a ser o número total de unidades de produto requisitadas, por janela temporal.

Exemplo: O sistema atende, para cada utilizador, um máximo de 30 unidades de produtos a cada 60 minutos.

Deverá entregar os ficheiros:

\*.java – dos dois exercícios

CBD-15a-out.txt – exemplos de interação e de output do programa a)

CBD-15b-out.txt – exemplos de interação e de output do programa b)

## 1.6 Sistema de mensagens (Opcional)

a) Construa um sistema de mensagens usando Redis e o driver cliente (Jedis). Deve prever as seguintes funcionalidades:

- Adição de utilizadores (identificados pelo nome).
- Associação entre utilizadores (e.g. se *userA* segue *userB*, então *userA* deve ter informação sobre todas as mensagens enviadas por *userB* para o sistema).
- Envio de mensagens, por exemplo:  
`storeMsg(userA, "Isto vai ser fácil!")`.
- Leitura de mensagens (por utilizador, etc.).

Elabore e desenvolva algumas funcionalidades adicionais.

Descreva as estruturas que criou/usou (e.g. num ficheiro *readme.txt*).

Deverá entregar os ficheiros:

\*.java – implementação

CBD-16.txt – documentação das estruturas de dados/chaves usadas

CBD-16a-out.txt – exemplos de interação e output do programa