INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia Informática e de Computadores

Relatório da CMDB

Chelas Movies Data Base

Trabalho realizado por:

Nome: André Silva	Nº 47224
Nome: Danilo Vieira	Nº 49988
Nome: Diogo Santos	Nº 48459

Docente: Filipe Bastos de Freitas

14 de janeiro de 2023

1 Introdução

Neste relatório irá encontrar informação sobre o manuseamento de dados e organização da estrutura da nossa aplicação web, desde organização do armazeno de dados no ElasticSearch e em memória, organização da aplicação e relação entre módulos e rotas do Web-site.

2 Desenvolvimento do Trabalho

2.1 Estrutura da aplicação

A nossa aplicação é composta por 2 estruturas o servidor e o cliente, o servidor gere os pedidos feitos pelo cliente e responde com a informação pedida.

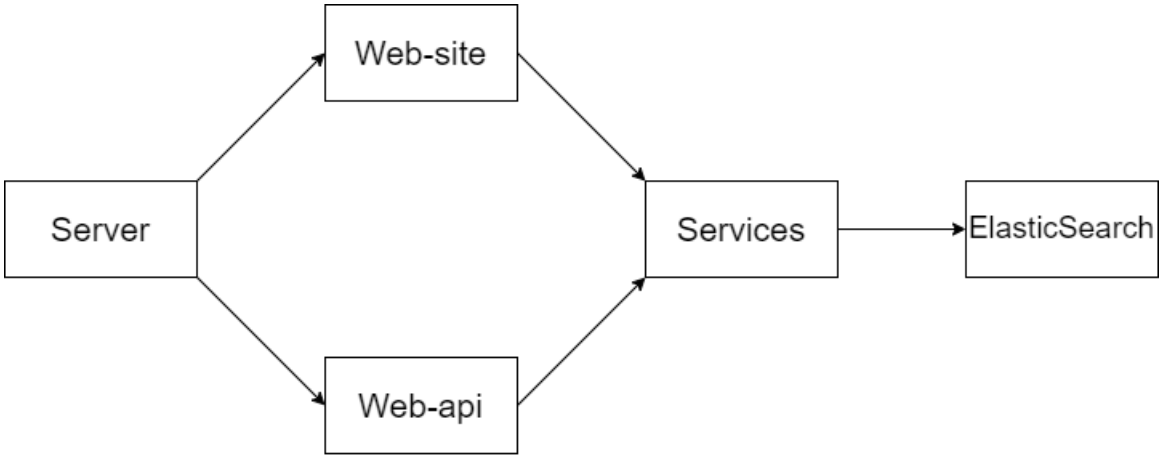

Figure 1 - Estrutura do Servidor

Na figura 1 podemos observar que o *entry point* da nossa aplicação é o Server, neste módulo temos defenido todos os URI's para cada pedido da nossa aplicação, tanto a nível de browser como da API. O módulo Web-api tem a responsabilidade de interpretar os pedidos HTTP e retirar a informação desses pedidos, tais como parâmetros da request e informação que se encontra no Body, para passar como parâmetro das funções dos Services.

O módulo Web-site está a correr em paralelo com o módulo Web-api, pois este módulo manda a representação do pedido para o cliente em HTML/CSS, para tal usa os métodos suportados pelo browser que são o GET e o POST, para métodos como DELETE e PUT é executado código em JavaScript do lado do cliente.

O módulo Services está encarregue de processar a informação pedida e verificar se esta foi corretamente enviada e executar toda a lógica da nossa aplicação, desde a criação de usuário a apagar um filme, este módulo não tem contacto com HTTP logo a responsabilidade de passar os parâmetros às funções do deste módulo é do módulo Web-api.

O módulo ElasticSearch é o módulo de armazenamento de dados, este guarda a informação dos grupos, usuários e filmes.

2.2 Estrutura do cliente

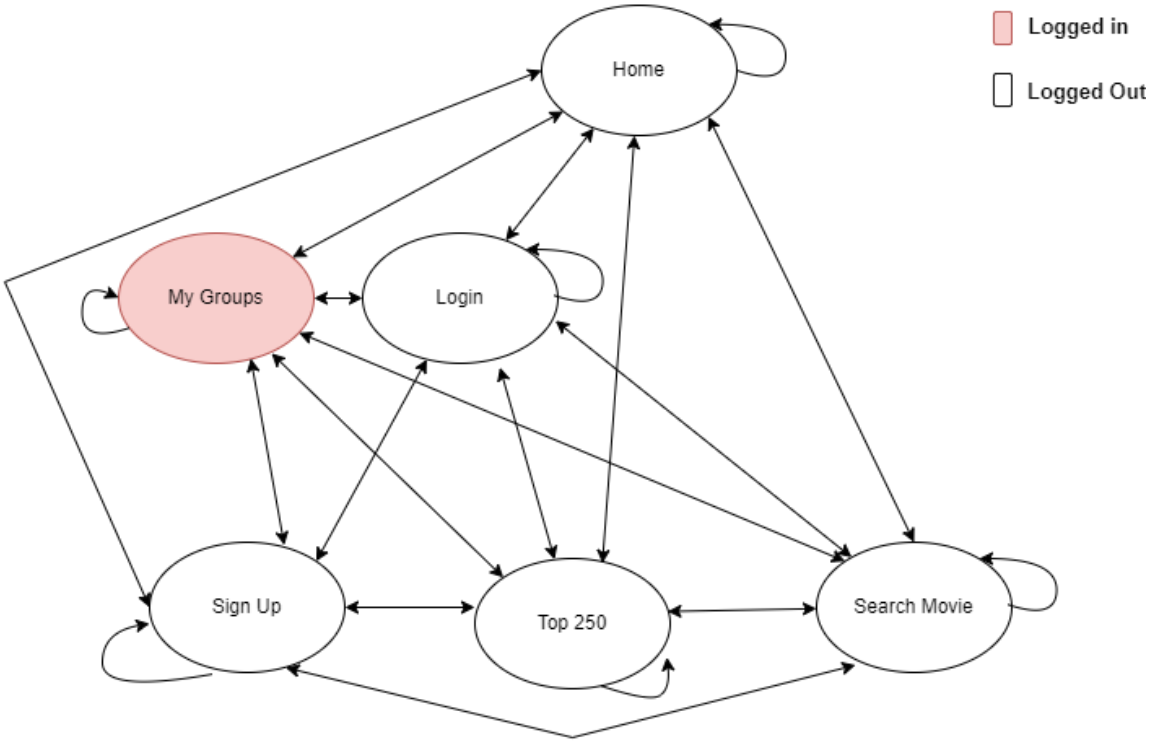

Figure 2 – mapa de acessos do cliente

Na página Home o usuário pode visualizar os vários sitios para onde pode ir dentro do website.

Na página Login o usuário pode iniciar sessão com os seus dados pessoais que estão guardados em memória ou no ElasticSearch.

Na página MyGroups, o usuário(**caso esteja logado**) pode ver os seus grupos.

Na página Sign Up podemos fazer criação dum novo usuário com nome e password.

No Top 250 é possivel ir buscar **até** 250 filmes mais bem cotados no IMDB recorrendo à API deste.

Na Search Movie apresentamos os vários filmes com nome semelhante ao que foi introduzido pelo user.

2.3 Design do data storage

O dados dos usuários da nossa aplicação web, informação do usuário e grupos, são armazenados numa base de dados ElasticSearch. Para armazenar esta informação é necessário defenir índices e documentos, onde índices contém os documentos, por exemplo:

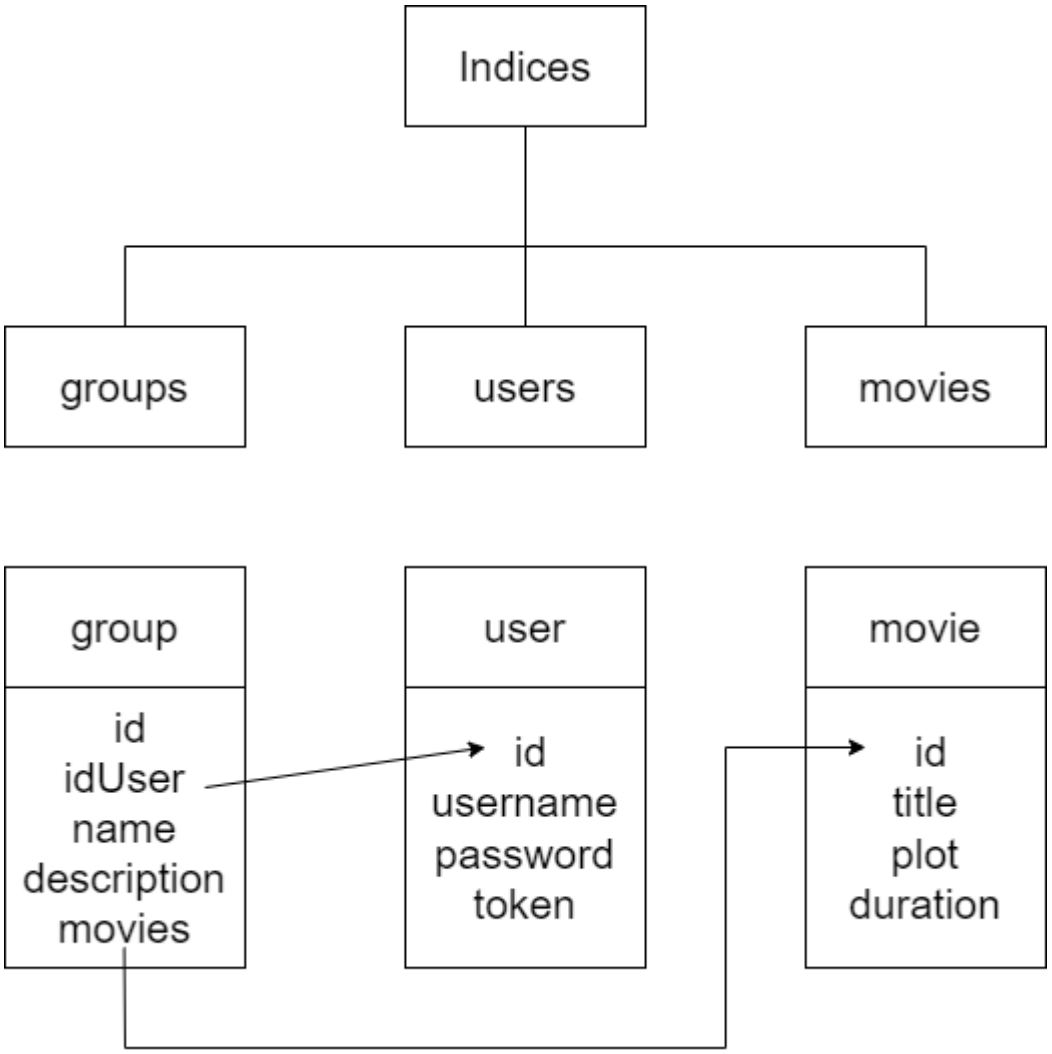

Figure 3 - Data Storage in ElasticSearch

Na figura em cima está representado o nosso modelo de armazenamento de dados onde existe 3 índices ***groups***, onde é armazenado a informação de cada grupo, ***users***, onde é armazenado a informação de cada usuário da nossa aplicação quando este se regista e ***movies*** onde é armazenado os filmes que estão presentes em ***groups***.

O nosso modelo faz uso de chaves estrangeiras para não haver informação repetida em cada índice, em cada documento ***group*** existe uma chave estrangeira **idUser** que faz referência a **id** de ***user*** para guardarmos a referência de qual usuário criou o grupo.

O mesmo se aplica em **movies** de *group*, nós temos um array de **id**'s de *movies* que fazem referência a **id** de *movie*, esta estratégia faz com que não estejamos a guardar o mesmo filme mais do que uma vez.

2.4 Descrição do mapeamento

O modo de armazenamento de dados no ElasticSearch, apesar de semelhante ao modo de armazenamento de dados em memória, existe uma divergência no formato do armazenamento de dados no ElasticSearch.

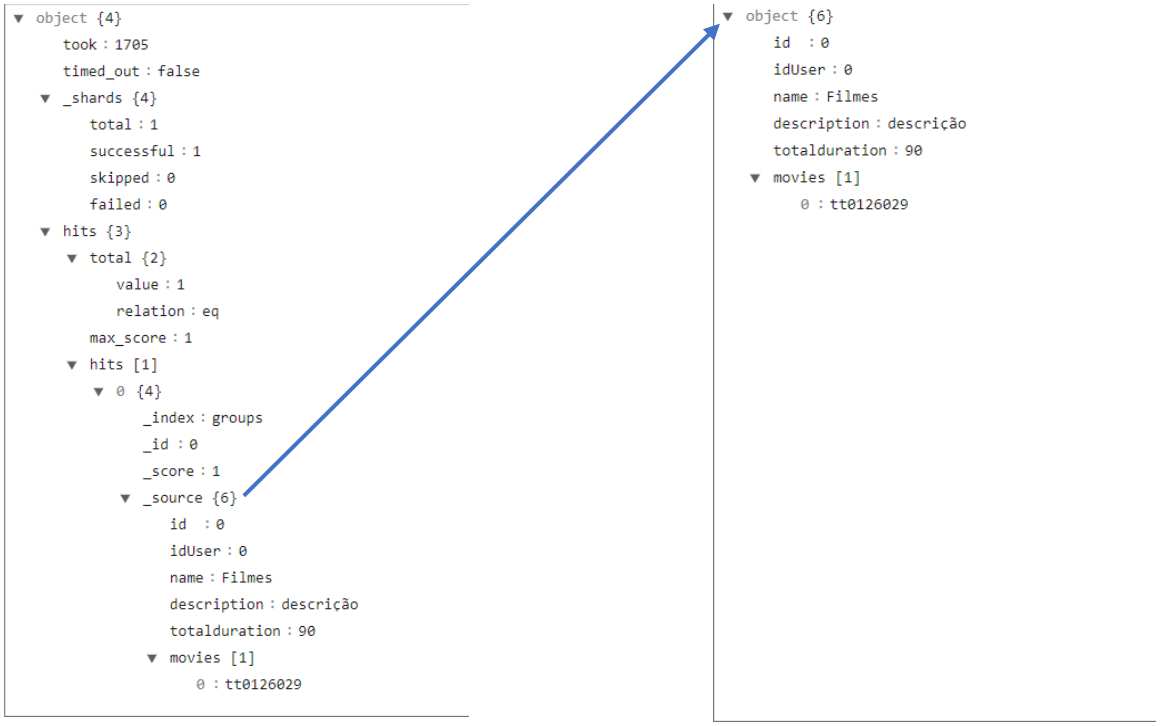

Figure 5 - Armazenamento em ElasticSearch

Figure 4 - Armazenamento em memória

Como podemos observar na figura 3 e 4, a informação em ElasticSearch fica guarda numa propriedade "_source" que pertence a uma propriedade array "hits" de uma propriedade "hits" do objeto armazenado (documento) no ElasticSearch. Dentro desta "_source" está a informação guardada em formato igual ao de memória.

2.5 Executar a aplicação

Para executar a nossa aplicação web em qualquer máquina, foi criada uma pasta na diretoria do Github com o nome de **setup**, essa pasta contém um ficheiro do tipo MJS com as funções necessárias a criar os índices *groups*, *users* e *movies* no ElasticSearch, para executar este ficheiro basta abrir uma consola no ficheiro de topo e executar o comado “node .\setup\setup.mjs”.

Para a execução dos testes abra uma consola na pasta **test** e executar o comando “node run test” .

```
6   import * as dataMem from './data/cmdb-data-mem.mjs'
7   import * as moviesdata from './data/cmdb-movies-data.mjs'
8   import * as dataElastic from './data/cmdb-data-elastic.mjs'
9   import webApiFunction from './web/api/cmdb-web-api.mjs'
10  import webSiteFunction from './web/site/cmdb-web-site.mjs'
11  import servicesFunction from './services/cmdb-services.mjs'
12  import authRouter from './cmdb-auth-web-site.mjs'
13
14  const PORT = 8080
15
16  const __dirname = url.fileURLToPath(new URL('.', import.meta.url));
17
18  const services = servicesFunction(dataElastic,moviesdata)
19  const webapi = webApiFunction(services)
20  const website = webSiteFunction(services)
```

Figure 6 - Setup de testes.

Certifique-se que o seu ficheiro “cmdb-server.mjs“ na linha 18 no primeiro parâmetro está o módulo de armazenamento de dados onde deseja testar.

3 Conclusões

Neste trabalho implementámos um servidor NodeJs usando o módulo Express para atender pedidos dos clientes de forma assíncrona, registando os seus dados ou em memória ou na base de dados ElasticSearch. Usamos também linguagem HTML usando também a framework bootstrap para embelezar o nosso website. Este servidor tem uma API e um website, sendo que ambos podem ser acessados quando o servidor está ativo, construímos o código em diferentes módulos para que alguns destes não dependessem doutros e que a informação na memória pudesse percorrer estes.