



**Universidade do Minho**

Mestrado Integrado em Engenharia Informática

Licenciatura em Ciências da Computação

## **Unidade Curricular de Bases de Dados**

Ano Lectivo de 2017/2018

### **Arpeggio Music – Serviços de Música Digital** *Parte II*

**Inês Sampaio, João Mourão, Pedro Almeida, Rui Vieira**

Janeiro, 2018

# BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

## **Arpeggio Music – Serviços de Música Digital**

### *Parte II*

**Inês Sampaio, João Mourão, Pedro Almeida, Rui Vieira**

Janeiro, 2018

<</opcional Dedicatória>>

## Resumo

O presente relatório é o culminar da elaboração da segunda fase do trabalho prático elaborado no âmbito da disciplina de Base de Dados, no qual foi proposta a criação de uma base de dados, numa primeira fase relacional, em SQL, e agora, na parte que aqui discutimos, a criação de uma base de dados não relacional, em Neo4j (NoSQL).

O caso de estudo tem por base uma plataforma digital de streaming de musica, a Arpeggio Music, que é apresentada no primeiro capítulo deste documento. É nesta primeira fase que, após a contextualização, é esboçada a motivação e os objetivos que sustentam a criação da base de dados não relacional.

Posteriormente, e após apresentado o caso de estudo, é apresentado o esboço do novo Sistema de Base de Dados, seguindo-se a isto os esclarecimentos necessários acerca da migração dos dados, criação de nodos, indexes e relacionamentos. A esta fase, segue-se a representação das queries, através das quais iremos aceder às informações contidas na base de dados e que nos ajudam também a verificar se está construída de acordo com os requisitos já anteriormente definidos.

Por fim, são tecidas as conclusões e apresentados comentários ao trabalho realizado, apontando os seus pontos fortes e fracos, comparativamente com o projeto realizado na primeira fase de implementação do trabalho prático.

**Área de Aplicação:** Desenho e arquitetura de Sistemas de Bases de Dados

**Palavras-Chave:** Base de Dados, não relacional, NoSQL, grafos, nodos, relacionamentos, migração, streaming, música.

# Índice

Resumo	i
Índice	ii
Índice de Figuras	iii
1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objectivos	2
2. Análise do Caso de estudo	3
3. Novo Sistema de Base de Dados	4
4. Migração de Dados	5
4.1. Importação de ficheiros csv para o Neo4j	6
4.2. Criação de Nodos	6
4.3. Criação de Indexes	6
4.4. Criação de Relacionamentos	7
5. Queries	9
6. Versão Final da Base de Dados em Neo4j	10
7. Conclusões e Trabalho Futuro	11
Referências	12
Lista de Siglas e Acrónimos	13

## Índice de Figuras

Figura 1 - Esquema representativo do novo sistema de base de dados.	4
Figura 2 - Script usado para a exportação dos dados presentes na BD em MySQL, para ficheiros .csv.	5
Figura 3 - Script respetivo à criação de nodos.	6
Figura 4 - Criação de indexes.	7
Figura 5 - Criação de relacionamentos.	7
Figura 6 - Query 1.	9
Figura 7 - Query 2.	9
Figura 8 - Query 3.	9
Figura 9 - Modelo final da base de dados implementada em Neo4j.	10

# 1. Introdução

## 1.1. Contextualização

Streaming é uma forma de distribuição digital frequentemente utilizada para partilhar conteúdo multimédia através da Internet, que retira a necessidade de o utilizador armazenar todo este conteúdo no seu computador. O tipo de ficheiros que podem ser distribuídos por este método varia imenso, sendo um dos mais populares os ficheiros de música.

O início do streaming de música deu-se em Janeiro de 1993<sup>1</sup>. O primeiro serviço foi lançado com o nome Internet Underground Music Archive(IUMA), e permitia a músicos que ainda não estivessem ligados a uma certa empresa, partilhar a sua música com os seus fãs<sup>1</sup>.

Com o passar dos anos estes serviços foram aumentando a sua complexidade permitindo que os seus utilizadores criassem playlists personalizadas, ouvissem os seus artistas favoritos e conhecessem novos tipos de música, tudo isto sem realizar nenhum pagamento (mas ouvindo publicidade), ou pagando por uma conta Premium (sem publicidade e outras funcionalidades).

Em 2017 nasceu um novo serviço de streaming, Arpeggio Music, criada em Portugal com a finalidade de permitir que todo o mundo consiga ouvir música de forma legal, por um preço mais acessível que aquele que o mercado oferecia até então. Para tornar o seu serviço mais atrativo, a Arpeggio Music criou uma subscrição com pagamento mensal de 5€ para até 5 utilizadores da mesma família, enquanto que para um utilizador individual o preço mensal seria de 2,5€.

O objetivo desta nova empresa é chegar a todos os pontos do globo, melhorando a forma como ouvimos música.

## 1.2. Apresentação do Caso de Estudo

Assume-se como principal objetivo desta fase do trabalho a implementação de um Sistema de Base de Dados (SBD) não relacional, mantendo como base para estudo o caso da Arpeggio Music.

Existem vários casos de estudo que assentam em modelos de Bases de Dados (BD) relacionais que têm por base o conceito de transações com propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Contudo, em casos de grandes empresas, devido a problemas de escalabilidade, estas propriedades, propostas pelo modelo relacional, acabaram por ser ignoradas de forma facilitar o crescimento da BD.

Foi com base neste problema que, os principais líderes da Internet como Facebook, Google e outros, passaram a usar a tecnologia NoSQL, uma vez que necessitavam Sistemas de Gestão de Base de Dados (SGBD) capazes de escrever e ler dados em qualquer lugar no mundo, e que garantissem um bom desempenho a grandes conjuntos de dados e milhões de utilizadores.

Desta forma, e com o objetivo de permitir alto desempenho perante grandes quantidades de dados, surgiu o NoSQL, não respeitando as regras impostas pelo modelo relacional, mas permitindo uma maior flexibilidade na construção de uma BD, consoante o objetivo desta.

Para fornecer à Arpeggio Music uma base de dados não relacional, iremos usar Neo4j, ferramenta NoSQL, que se distingue por funcionar à base de grafos - cada nodo e aresta pode possuir um número ilimitado de atributos e rótulos (labels), de forma a restringir a pesquisa.

### **1.3. Motivação e Objectivos**

NoSQL é um modelo de base de dados completamente distinto do tradicional. Como o próprio nome indica, não tem interface SQL e descreve soluções de armazenamento de dados não relacional. Este tipo de BD surgiu efetivamente derivado da necessidade de dar resposta a questões que as bases de dados relacionais não eram capazes de solucionar<sup>2</sup>.

Apesar de não terem uma estrutura de dados relacional (que pode ser preponderante em alguns casos muito específicos) os modelos NoSQL têm a si associados fatores muito vantajosos, tais como:

- custos mais reduzidos
- dados sempre disponíveis
- base de dados orientada a objetos flexíveis
- facilidade de introdução de dados
- facilidade de gestão de grandes quantidades de informação<sup>3</sup>.

Podemos ainda realçar que existem quatro tipos distintos de bases de dados NoSQL: chave-valor, grafos, colunas e documentos, mas o presente trabalho foca-se essencialmente em grafos e na utilização da ferramenta Neo4j.

Tendo a Arpeggio music como ponto de partida, a nossa principal motivação para a criação de um SBD não relacional é facilitar o acesso, a inserção e a modificação de informação, permitindo uma gestão de recursos mais eficiente. A partir da contextualização que apresentamos, é possível deduzir que o sistema deste serviço é extremamente complexo, sendo necessário manter toda a informação completa, atualizada e sempre disponível.

Devido à enormíssima quantidade de informação contida no sistema, é fundamental, para o correto funcionamento da plataforma de streaming de músicas, uma base de dados bem construída.

O principal objetivo ao criar esta BD é simplificar e garantir a organização correta de toda a informação relativa ao serviço. Daqui, surge a necessidade de analisar todas as possibilidades à disposição do utilizador e construir a base de dados, mantendo sempre em mente possíveis problemas que surgem com regularidade aos utilizadores.



## 2. Análise do Caso de estudo

Nesta segunda parte do trabalho, apesar de termos alterado o modelo de implementação da BD, assentamos a construção do SBD nos requisitos que havíamos definido na primeira fase de desenvolvimento do projeto.

Devido à mudança de modelo, é notória uma diferença na implementação da base de dados, consequência de passar de uma implementação relacional para uma implementação não relacional e baseada em grafos. Cada implementação tem características muito próprias, com as suas respetivas vantagens e desvantagens.

As entidades definidas no antigo modelo vão ser mantidas, apesar de sofrerem algumas alterações na sua forma. O Neo4j define as entidades como sendo nodos, e são estes nodos que se vão relacionar com a criação de relacionamentos próprios desta ferramenta<sup>4</sup>.

Assumimos também que as questões que o nosso sistema deve responder são as definidas anteriormente, visto que, apesar de alterarmos o modelo de BD, vamos manter os mesmos requisitos. Desta forma, o sistema tem de responder a perguntas como:

- Número de utilizadores, de um determinado país, que seguem um artista específico.
- Procurar o número de faixas de uma playlist.
- Procurar quais são os álbuns de um determinado artista.
- Procura as faixas, de um dado artista, que foram lançadas entre duas datas.

### 3. Novo Sistema de Base de Dados

A figura 1 retrata a implementação da BD da Arpeggio Music em Neo4j, tendo sido identificados os tipos de nodos presentes e os relacionamentos entre eles. Este é o esboço inicial representativo da BD e permite facilitar a sua compreensão.

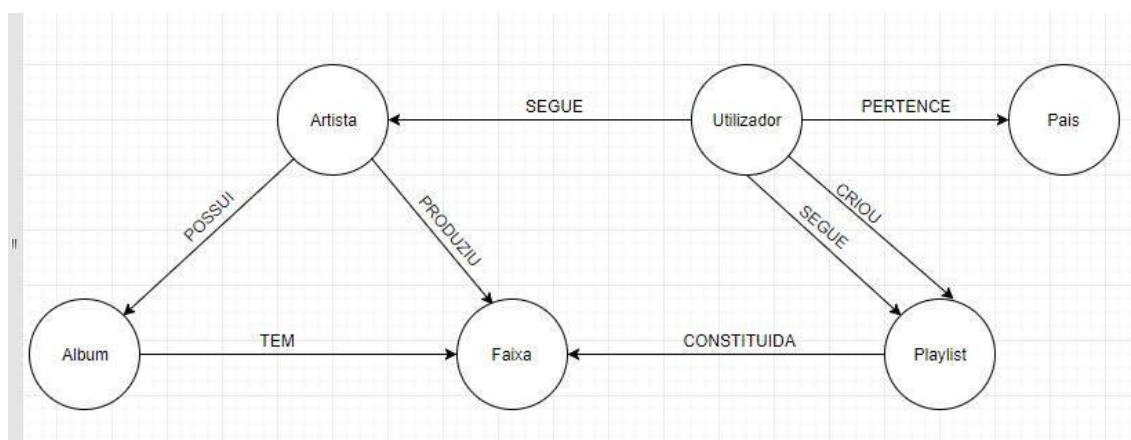


Figura 1 - Esquema representativo do novo sistema de base de dados.

## 4. Migração de Dados

A migração de dados é um processo que consiste na exportação de dados presentes na base de dados em MySQL, já povoada durante a execução da primeira parte do projeto, para ficheiros com a extensão .csv. Cada tabela foi exportada para um ficheiro diferente, com o respetivo nome, como se pode verificar na figura 2. Devido às permissões do MySQL Server, neste processo de migração foi necessário importar os documentos para uma diretoria específica<sup>5</sup>.

```
use arpeggio;

SELECT * FROM album
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/album.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM artista
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/artista.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM faixa
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/faixa.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM pais
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/pais.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM playlist
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/playlist.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM utilizador
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/utilizador.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM playlist_has_faixa
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/playlist_has_faixa.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM utilizador_follows_artista
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/utilizador_follows_artista.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

SELECT * FROM utilizador_follows_playlist
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/utilizador_follows_playlist.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';
```

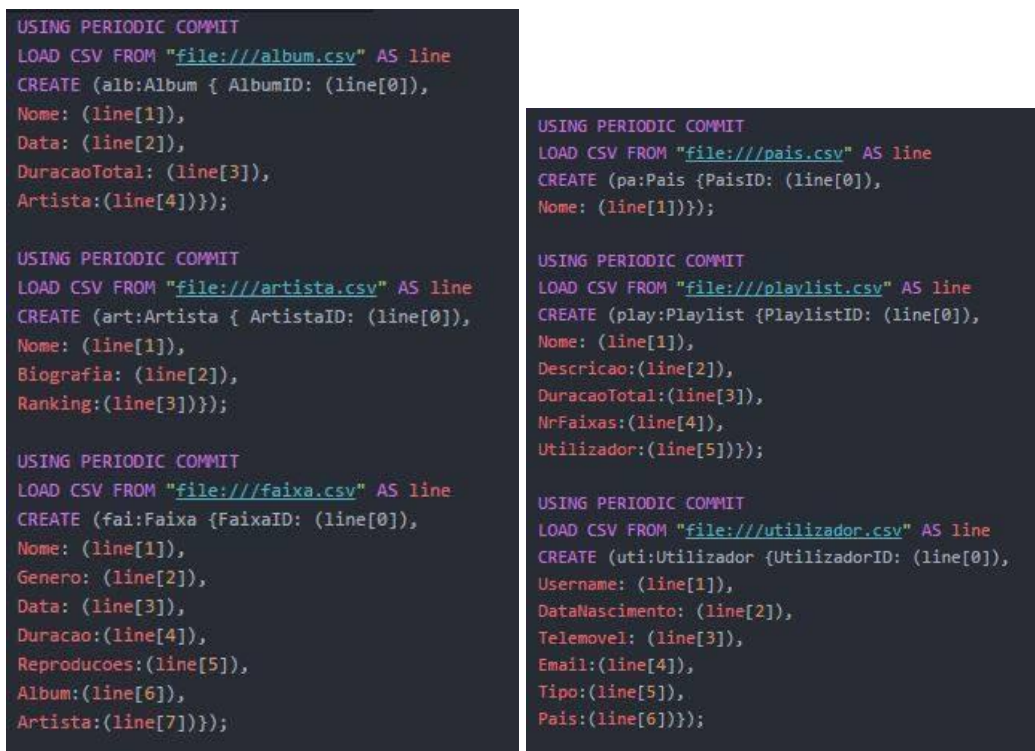
Figura 2 - Script usado para a exportação dos dados presentes na BD em MySQL, para ficheiros .csv.

## 4.1. Importação de ficheiros csv para o Neo4j

Depois de criados os ficheiros .csv, foram então feitos vários scripts na linguagem Cypher para criar os nodos e os seus respetivos relacionamentos. Foi necessário fazer vários scripts devido à incompatibilidade da versão utilizada do Neo4j, uma vez que esta não permite a criação dos nodos todos no mesmo script. Para fazer o carregamento do ficheiro .csv com as informações da tabela para o Neo4j utilizou-se a funcionalidade “LOAD” e os respetivos nodos foram criados recorrendo à funcionalidade ‘CREATE’. Por fim, para criar os relacionamentos presentes entre os nodos utilizou-se a funcionalidade ‘MERGE’.

## 4.2. Criação de Nodos

Na figura 3 é possível observar os scripts que permitiram a criação dos nodos seguindo a estrutura que se encontra no ficheiro .csv. Para cada nodo específico é inserida toda a informação que o ficheiro .csv continha sobre este.



```

USING PERIODIC COMMIT
LOAD CSV FROM "file:///album.csv" AS line
CREATE (alb:Album { AlbumID: (line[0]),
Nome: (line[1]),
Data: (line[2]),
DuracaoTotal: (line[3]),
Artista:(line[4])});

USING PERIODIC COMMIT
LOAD CSV FROM "file:///artista.csv" AS line
CREATE (art:Artista { ArtistaID: (line[0]),
Nome: (line[1]),
Biografia: (line[2]),
Ranking:(line[3])});

USING PERIODIC COMMIT
LOAD CSV FROM "file:///faixa.csv" AS line
CREATE (fai:Faixa {FaixaID: (line[0]),
Nome: (line[1]),
Genero: (line[2]),
Data: (line[3]),
Duracao:(line[4]),
Reproducoes:(line[5]),
Album:(line[6]),
Artista:(line[7])});

USING PERIODIC COMMIT
LOAD CSV FROM "file:///pais.csv" AS line
CREATE (pa:Pais {PaisID: (line[0]),
Nome: (line[1])});

USING PERIODIC COMMIT
LOAD CSV FROM "file:///playlist.csv" AS line
CREATE (play:Playlist {PlaylistID: (line[0]),
Nome: (line[1]),
Descricao:(line[2]),
DuracaoTotal:(line[3]),
NrFaixas:(line[4]),
Utilizador:(line[5])});

USING PERIODIC COMMIT
LOAD CSV FROM "file:///utilizador.csv" AS line
CREATE (uti:Utilizador {UtilizadorID: (line[0]),
Username: (line[1]),
DataNascimento: (line[2]),
Telemovel: (line[3]),
Email:(line[4]),
Tipo:(line[5]),
Pais:(line[6])});

```

Figura 3 - Script respetivo à criação de nodos.

## 4.3. Criação de Indexes

Os Indexes (Índices) são uma cópia redundante das informações da base de dados e têm como objetivo tornar a pesquisa de dados mais eficiente. Isto requer um espaço extra de armazenamento e gravações mais lentas, pelo que este processo desempenha

um papel importante e muitas das vezes não trivial. A linguagem Cypher permite a criação de índices sobre uma dada propriedade para todos os nodos que têm um rótulo em comum. Depois da criação do índice, este é automaticamente gerado e atualizado pela base de dados, sempre que o gráfico for alterado.

Na figura 4 mostramos como foi feita a criação de índices relativamente à nossa base de dados.

```
CREATE INDEX ON :Album(AlbumID);
CREATE INDEX ON :Artista(ArtistaID);
CREATE INDEX ON :Faixa(FaixaID);
CREATE INDEX ON :Pais(PaisID);
CREATE INDEX ON :Playlist(PlaylistID);
CREATE INDEX ON :Utilizador(UtilizadorID);
```

Figura 4 - Criação de indexes.

## 4.4. Criação de Relacionamentos

Como podemos observar na figura 5 existem diferentes tipos de relacionamentos entre os nodos.

```
USING PERIODIC COMMIT
LOAD CSV FROM "file:///album.csv" AS row
MATCH (alb:Album {AlbumID: row[0]})
MATCH (art:Artista {ArtistaID: row[4]})
MERGE (art)-[:POSSUI]->(alb);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///faixa.csv" AS row
MATCH (fai:Faixa {FaixaID: row[0]})
MATCH (alb:Album {AlbumID: row[6]})
MERGE (alb)-[:TEM]->(fai);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///faixa.csv" AS row
MATCH (fai:Faixa {FaixaID: row[0]})
MATCH (art:Artista {ArtistaID: row[7]})
MERGE (art)-[:PRODUZIU]->(fai);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///playlist.csv" AS row
MATCH (play:Playlist {PlaylistID: row[0]})
MATCH (uti:Utilizador {UtilizadorID: row[5]})
MERGE (uti)-[:CRIOU]->(play);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///utilizador.csv" AS row
MATCH (uti:Utilizador {UtilizadorID: row[0]})
MATCH (pai:Pais {PaisID: row[6]})
MERGE (uti)-[:PERTENCE]->(pai);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///utilizador follows artista.csv" AS row
MATCH (uti:Utilizador {UtilizadorID: row[0]})
MATCH (art:Artista {ArtistaID: row[1]})
MERGE (uti)-[:SEGUE]->(art);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///utilizador follows playlist.csv" AS row
MATCH (uti:Utilizador {UtilizadorID: row[0]})
MATCH (play:Playlist {PlaylistID: row[1]})
MERGE (uti)-[:SEGUE]->(play);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///playlist has faixa.csv" AS row
MATCH (play:Playlist {PlaylistID: row[0]})
MATCH (fai:Faixa {FaixaID: row[1]})
MERGE (play)-[:CONSTITUIDA]->(fai);
```

Figura 5 - Criação de relacionamentos.

Os tipos de relacionamentos são os seguintes:

- **(ARTISTA)-[:POSSUI]->(ALBUM)**  
É o relacionamento entre cada artista e cada álbum que ele possui
- **(ALBUM)-[:TEM]->(FAIXA)**  
É o relacionamento entre cada álbum e cada faixa. Representa as faixas contidas dentro de um dado álbum.
- **(ARTISTA)-[:PRODUZIU]->(FAIXA)**  
É o relacionamento entre cada artista e cada faixa que ele produziu.
- **(UTILIZADOR)-[:CRIOU]->(PLAYLIST)**  
É o relacionamento entre cada utilizador e cada playlist que ele criou.
- **(UTILIZADOR)-[:PERTENCE]->(PAIS)**  
É o relacionamento entre cada utilizador e o país onde reside.
- **(UTILIZADOR)-[:SEGUE]->(ARTISTA)**  
É o relacionamento entre o utilizador e o artista. Este relacionamento representa os seguidores de um determinado artista.
- **(UTILIZADOR)-[:SEGUE]->(PLAYLIST)**  
É o relacionamento entre o utilizador e cada playlist que ele segue.
- **(PLAYLIST)-[:CONSTITUIDA]->(FAIXA)**  
É o relacionamento entre a playlist e cada faixa. Representa as faixas que constituem uma determinada playlist

Estes relacionamentos vão desempenhar um papel importantíssimo, uma vez que possibilitam a navegação no grafo com o objetivo de obter a informação desejada na elaboração de queries.

## 5. Queries

A elaboração de queries em qualquer modelo de Base de Dados é um processo importante para obter os dados pretendidos e concluir, através desses resultados, se a elaboração da base de dados cumpre todos os requisitos pedidos. As queries no modelo Neo4j baseiam-se na linguagem Cypher.

De seguida apresentamos algumas queries e a sua respetiva explicação que efetuamos na nossa base de dados criada nesta segunda fase do projeto.

A query presente na figura 6 permite selecionar todos os utilizadores da base de dados, retornando os seus usernames e os respetivos emails.

```
MATCH(uti:Utilizador)
RETURN uti.Username, uti.Email;
```

Figura 6 - Query 1.

A query presente na figura 7 permite selecionar o utilizador onde o UtilizadorID é 1 e ver quais os artistas que ele segue retornando o username do utilizador e o nome do artista.

```
MATCH (uti{UtilizadorID:'1'})-[:SEGUE]->(art:Artista)
RETURN uti.Username, art.Nome;
```

Figura 7 - Query 2.

Na figura 8 está representada a query que permite contar quantas faixas tem o álbum cujo nome é Reputation, retornando o nome do álbum e essa contagem.

```
MATCH (a:Album) -[:TEM]-> (f:Faixa)
WHERE (a.Nome = "Reputation")
RETURN a.Nome, count(f);
```

Figura 8 - Query 3.



## 6. Versão Final da Base de Dados em Neo4j

O Neo4j na parte da visualização gráfica da base de dados, fornece-nos uma identificação colorida dos diferentes tipos de nodos presentes na base dados, facilitando a interpretação dos relacionamentos e dos tipos de nodos nela presentes.

Como podemos verificar na figura 9 cada tipo de nodo tem uma cor específica que permite a sua identificação.

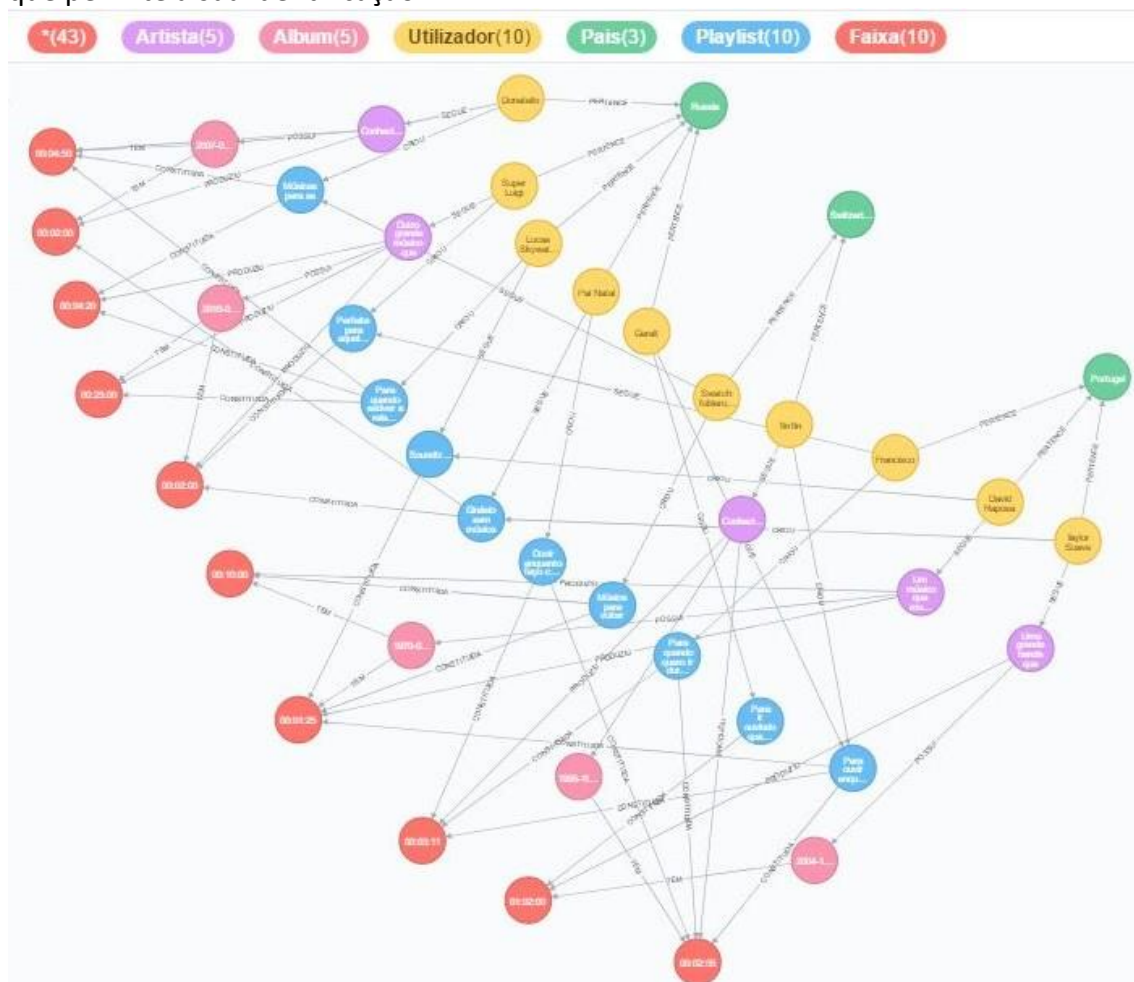


Figura 9 - Modelo final da base de dados implementada em Neo4j.

O Neo4j oferece uma API simples para acesso aos dados, por exemplo, ao observar o resultado gráfico da BD podemos de imediato identificar todos os tipos de dados nela presentes de uma forma fácil. Para além disso, trata-se de uma base de dados de custo mais reduzido em comparação com uma base de dados relacional e também facilita a introdução de novos dados.



## 7. Conclusões e Trabalho Futuro

A realização deste trabalho permitiu compreender as principais diferenças entre bases de dados relacionais e não relacionais, além de, como é óbvio, contribuir para a aquisição de competências no que toca a bases de dados NoSQL.

Comparativamente com a primeira fase do trabalho, revelou-se mais prática, por não ser necessário redefinir contextos e requisitos que haviam sido explicitados anteriormente. No entanto, estes foram tidos como ponto de partida na implementação de um novo sistema, passando esta segunda parte do trabalho, essencialmente por adaptar a base de dados da Arpeggio Music de relacional para não relacional.

Por ser baseada em grafos e nodos, o Neo4j trouxe-nos uma perspetiva completamente diferente daquilo que pode ser uma BD. Uma das grandes vantagens de uma base de dados NoSQL é possuir um elevado grau de distribuição de dados, o que possibilita mais solicitações de dados e permite que o sistema fique indisponível durante menos tempo. Por outro lado, consideramos o modelo relacional mais consistente e rigoroso, muito devido às regras que é necessário cumprir para a sua implementação.

Para terminar, concluímos que a realização global deste trabalho (partes 1 e 2) é fundamental para adquirir noções claras do que é uma base de dados, das diferenças entre relacional e não relacional e de que forma ambas se encaixam em diferentes situações do mercado de trabalho. Admitimos que este conhecimento será uma mais valia num futuro em que tenhamos que decidir que tipo de SGBD deve ser aplicado em determinado projeto e como implementá-lo.

## Referências

- [1] Sutori.com. (2017). Sutori. [online] Disponível em: <https://www.sutori.com/story/history-of-music-streaming> [Acedido em: 28 Dec. 2017].
- [2] DevMedia, p. (2018). Repositório de Dados Relacional ou NoSQL?. [online] DevMedia. Disponível em: <https://www.devmedia.com.br/repositorio-de-dados-relacional-ou-nosql/27500> [Acedido em: 11 Jan. 2018].
- Hadoop360. (2018). Advantages and Disadvantages of NoSQL databases – what you should know. [online] Disponível em: <https://www.hadoop360.datasciencecentral.com/blog/advantages-and-disadvantages-of-nosql-databases-what-you-should-k> [Acedido em 13 Jan. 2018].
- [3] Neo4j Graph Database Platform. (2018). The Neo4j Graph Platform – The #1 Platform for Connected Data. [online] Disponível em: <https://neo4j.com/> [Acedido em 13 Jan. 2018]
- [4] Tech-Recipes: A Cookbook Full of Tech Tutorials. (2018). Save MySQL query results into a text or CSV file. [online] Disponível em: <http://www.tech-recipes.com/rx/1475/save-mysql-query-results-into-a-text-or-csv-file/> [Acedido em 13 Jan. 2018].

## Lista de Siglas e Acrónimos

<b>API</b>	Application Programming Interface
<b>ACID</b>	Atomicidade, Consistência, Isolamento e Durabilidade
<b>BD</b>	Base de Dados
<b>NoSQL</b>	Not Only SQL
<b>SQL</b>	Structured Query Language
<b>SBD</b>	Sistema de Base de Dados
<b>SGBD</b>	Sistema de Gestão de Base de Dados