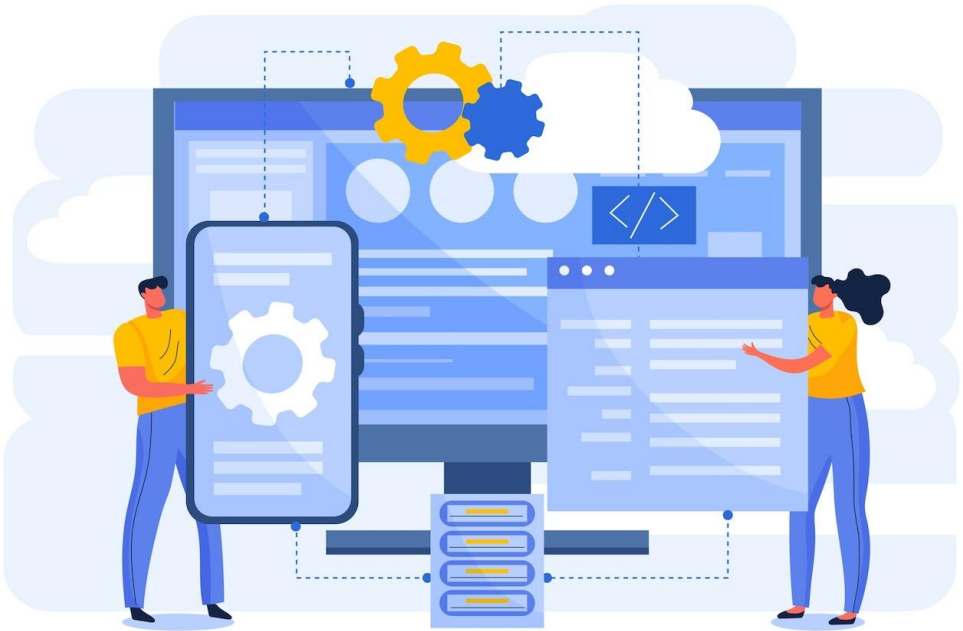
IPCA
INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE

IPCA INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

Licenciatura em Engenharia de Sistemas Informáticos

Integração de Sistemas de Informação

2025/2026



Análise de Dados Musicais

Miguel Silva Areal

Nº 29559 – LESI PL

Docente: Luís Ferreira

Índice

Conteúdo

Índice de Figuras

Glossário

• **ETL (Extract, Transform, Load):** Processo que extrai dados de várias fontes, transforma-os para garantir consistência e qualidade, e carrega-os num destino final para análise ou armazenamento.

• **KNIME:** Ferramenta open-source para criação e automação de processos ETL. Permite extrair, transformar e carregar dados de forma visual e integrada, através de fluxos compostos por nós (nodes).

• **Node-RED:** Ferramenta open-source desenvolvida pela IBM usada para integrar sistemas e fluxos de dados através de uma interface visual baseada em nós (nodes). Utilizada frequentemente em IoT, automação e comunicação entre APIs e dispositivos.

• **Dashboard:** Painel visual que apresenta dados de forma interativa, através de gráficos, tabelas e indicadores, facilitando a análise e a tomada de decisões.

• **API (Application Programming Interface):** Conjunto de regras e definições que permite a comunicação e troca de dados entre diferentes aplicações ou sistemas.

• **CSV (Comma-Separated Values):** Formato de ficheiro de texto simples que armazena dados em formato tabular, separados por vírgulas ou ponto e vírgula.

• **SQL (Structured Query Language):** Linguagem de consulta estruturada usada para gerir e manipular bases de dados relacionais.

• **JSON (JavaScript Object Notation):** Formato leve e estruturado para armazenar e trocar dados entre sistemas, de forma simples e legível por humanos e máquinas.

• **SQLite:** Sistema de gestão de bases de dados relacional leve e embutido, que armazena dados em ficheiros locais. É amplamente usado em aplicações que necessitam de bases de dados simples e portáteis.

• **DB Browser for SQLite:** Aplicação gráfica que permite criar, visualizar, editar e gerir bases de dados SQLite de forma simples e intuitiva.

• **Transformação:** Conjunto de operações aplicadas aos dados dentro de um fluxo (workflow), como limpeza, filtragem, junção e cálculo de novos campos. Representa o núcleo do processo ETL.

• **Expressão Regular (Regex) :** Padrão textual usado para pesquisar, validar ou extrair partes de strings com base em regras de correspondência

• **Dataset :** Coleção de dados relacionados, geralmente organizada de forma estruturada (como num ficheiro JSON ou ficheiro CSV), que pode conter vários tipos de informação como números, texto, imagens ou áudio.

• **Access Token :** Credencial que concede a uma aplicação acesso a recursos específicos em nome de um utilizador. É um código que contém informações utilizadas para autenticar e autorizar um utilizador a um serviço, como uma API.

• **HTTP Requests :** Mensagens enviadas de um cliente para um servidor com o objetivo de pedir algum recurso ou executar uma ação.

Introdução

O presente projeto de avaliação integra a Unidade Curricular de Integração de Sistemas de Informação, lecionada no 1.º semestre do 3.º ano do curso de Engenharia de Sistemas Informáticos do Instituto Politécnico do Cávado e do Ave.

O principal objetivo deste trabalho prático é aplicar e explorar ferramentas associadas a processos *ETL*, no âmbito dos Sistemas de Informação, recorrendo à utilização de scripts próprias.

Um processo *ETL* (*Extract*, *Transform*, *Load*) é responsável pela integração de três etapas fundamentais: Extração, transformação e carregamento de dados, garantindo a aplicação de boas práticas de tratamento, validação e automação. O resultado esperado é a obtenção de informação limpa, estruturada e consistente, adequada para posterior análise.

Na fase final, pretende-se que os resultados obtidos sejam integrados num *dashboard* desenvolvido no *Node-RED*, possibilitando a visualização e análise interativa dos dados processados.

A integração entre *KNIME*, *SQL* e *Node-RED* demonstra a construção de um pipeline de dados completo, no qual o processamento técnico e a análise visual se complementam, permitindo gerar informação estruturada e útil para a tomada de decisão.

Desta forma, o projeto evidencia não só as capacidades técnicas do *KNIME* na integração e automação de dados, mas também a relevância da visualização analítica na fase final do processo, proporcionando uma abordagem prática e aplicada aos conceitos fundamentais dos sistemas de informação empresariais.

Contexto

No atual panorama empresarial, os processos de negócio encontram-se em constante transformação. As organizações são diariamente confrontadas com desafios que exigem capacidade de adaptação, análise e integração de novas soluções informáticas que otimizem o seu funcionamento. A transformação digital, aliada à crescente competitividade do mercado, obriga as empresas a tomarem decisões estratégicas fundamentadas em dados e em ferramentas tecnológicas que potenciem a sua eficiência e rentabilidade.

Neste cenário, torna-se essencial maximizar o retorno do investimento em soluções informáticas, garantindo simultaneamente a coerência e continuidade dos processos de negócio e a integridade dos dados que os sustentam. A escolha de novas tecnologias deve, por isso, ser cuidadosamente avaliada, considerando não só os benefícios diretos que proporciona, mas também os impactos que pode gerar a nível operacional, financeiro e organizacional. Assim, as empresas procuram metodologias e ferramentas que lhes permitam analisar de forma rigorosa as vantagens e desvantagens associadas a cada nova aquisição tecnológica, reduzindo riscos e potenciando oportunidades.

Com este enquadramento em mente, foi concebida uma solução que visa responder a um problema simulado, mas inspirado em situações reais do contexto empresarial:

Problema:
Uma editora discográfica pretende compreender melhor o panorama musical atual e identificar as oportunidades de investimento mais promissoras. O seu objetivo é determinar com quais artistas, géneros ou estilos musicais deve estabelecer parcerias para alcançar maior sucesso económico nos serviços de *streaming*.

Solução:
Recorrendo a dados musicais do mercado, bem como a ferramentas de tratamento, análise e visualização de dados, será desenvolvido um estudo que permita identificar padrões de consumo, tendências emergentes e fatores determinantes para o sucesso no setor musical. Esta análise apoiará a tomada de decisão estratégica da editora, fornecendo-lhe uma base sólida para direcionar os seus investimentos e colaborações futuras de forma informada e sustentada.

Objetivos

Os objetivos do desenvolvimento deste trabalho são os seguintes:

- Consolidar conceitos associados à *Integração de Sistemas de Informação* com foco em dados, compreendendo a importância da extração, transformação e carga (*ETL*) na gestão de informação.
- Explorar e dominar ferramentas de suporte a processos de *ETL*, como *KNIME*, *SQL* e *Node-RED*, entendendo as suas funcionalidades, limitações e potenciais integrações.
- Experimentar novas tecnologias, *frameworks* e paradigmas que possam otimizar processos de integração e transformação de dados, como APIs, JSON, CSV, *dashboards* interativos e automação de fluxos.
- Desenvolver competências em documentação e comunicação técnica, contribuindo para a qualidade da escrita de relatórios e apresentação de resultados.
- Integrar visualização analítica e interpretação de dados, permitindo a criação de *dashboards* que traduzam informações complexas em *insights* claros e acionáveis.
- Promover boas práticas de gestão de dados, incluindo organização de bases de dados, normalização, validação e transformação de informação proveniente de múltiplas fontes.

Estratégia

A estratégia adotada no desenvolvimento deste trabalho baseou-se numa abordagem prática e iterativa permitindo construir progressivamente todas as fases de um processo *ETL* e garantir a sua integração no final, dando resposta a todos os pontos pedidos no enunciado.

Para o planeamento foram definidos os objetivos principais. Criar uma transformação capaz de importar dados, validar a sua consistência, aplicar operações de transformação e exportar resultados em diferentes formatos.

Para a importação foi carregado um *dataset* de dados de músicas do serviço de streaming de músicas *Spotify* em formato CSV como a base dos dados desta análise.

A partir da informação do *dataset* foi recolhida e agregada informação adicional proveniente da API do *Spotify*, como dados extra de artistas e músicas.

Toda a informação disponível sofreu transformações e separada de forma a ser inserida numa base de dados *SQLite*.

Para demonstrar alterações nos dados foi criada uma de análise no *Node-Red*, alimentada pela base de dados criada.

Esta abordagem permite uma integração completa entre a extração técnica e a interpretação visual dos resultados, proporcionando uma visão global sobre o ciclo de vida dos dados.

Assim, a estratégia seguida privilegiou a organização, modularidade e validação contínua, garantindo um desenvolvimento controlado.

Dados

Os dados utilizados no projeto estão divididos em diferentes pastas.

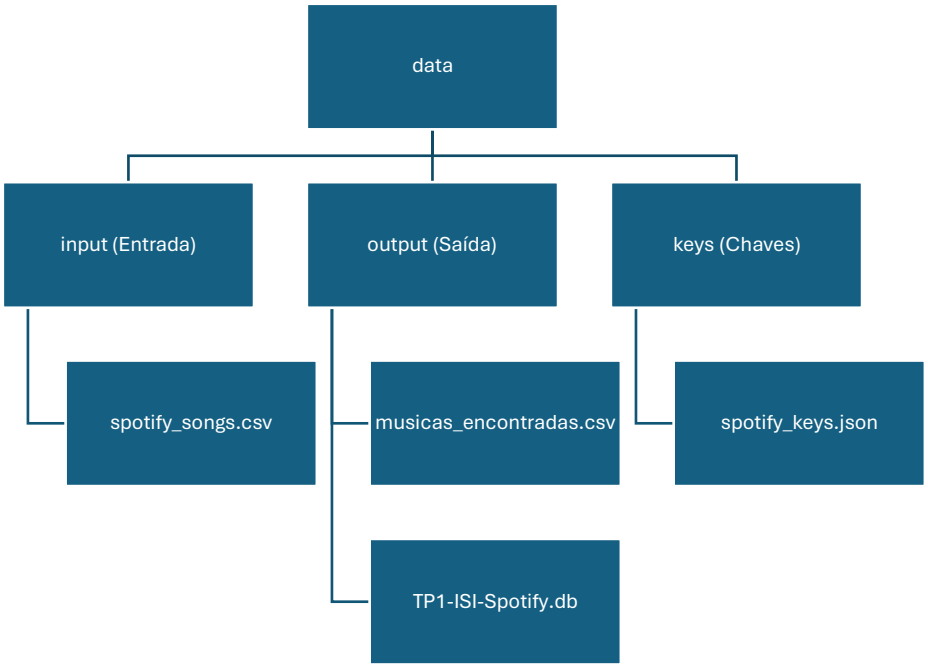


Figura 1 - Estrutura hierárquica dos dados utilizados.

Legenda:

- Data/Input/spotify_songs.csv
  - *Dataset* de entrada que contém cerca de 30000 músicas, no formato *CSV*.
- Data/Output/musicas_encontradas.csv
  - Ficheiro CSV que contém todas as músicas encontradas durante o processo *ETL*, é enviado como anexo no e-mail de notificação final.
- Data/Output/TP1-ISI-Spotify.db
  - Ficheiro de base de dados *SQL* que contém todos os dados da transformação *ETL*, separado em diferentes tabelas.
- Data/Keys/spotify_keys.json
  - Ficheiro em formato JSON que contém as credenciais de acesso à API do *Spotify*, estas sendo o client_id e client_secret.

KNIME - Processo ETL

1. Autenticação na API do Spotify

Antes de dar início à transformação dos dados, são criadas as condições necessárias para a autenticação na API do Spotify.

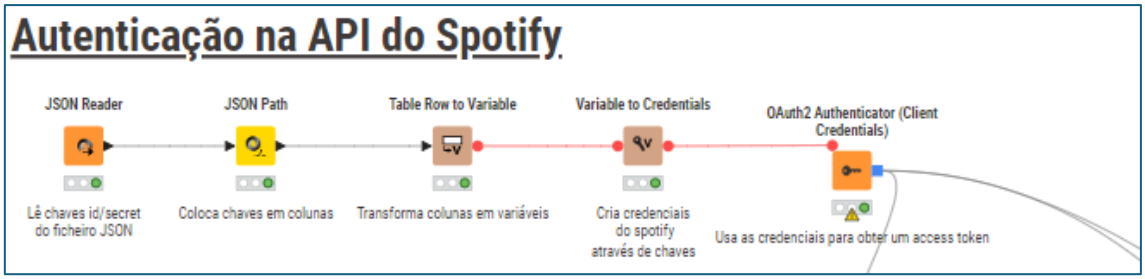


Figura 2 – Workflow de autenticação no serviço do Spotify

Neste processo, o objetivo é obter um *Access Token*. Este é responsável por nos permitir executar todas as *GET Requests* necessárias para podermos completar os dados do dataset original.

Para isto, precisamos de primeiramente ter uma conta Spotify, e criar uma app através do painel de desenvolvedor (https://developer.spotify.com/). Esta *app* permite-nos requisitar as credenciais *client_id* e *client_secret*, que neste contexto ficam gravadas em "data/keys/spotify_keys.json" no formato:

```
{
    "client_id":"12345",
    "client_secret":"12345"
}
```

Figura 3 - Formato exemplo do ficheiro de dados de credenciais.

Este ficheiro é importado e lido pelo *JSON Reader*, e os seus dados são extraídos para colunas utilizando o *JSON Path*. De seguida, estas novas colunas são transformadas em variáveis do *workflow* para que possam ser convertidas para o tipo de dados *Credential*. Esta *credential* pode então ser utilizada facilmente pelo node de autenticação, que faz um pedido de requisição do *Access Token* pelo *Endpoint* https://accounts.spotify.com/api/token. Se aceite, podemos agora utilizar este node para todos os próximos pedidos á API do *Spotify.*

2. Leitura de Dataset

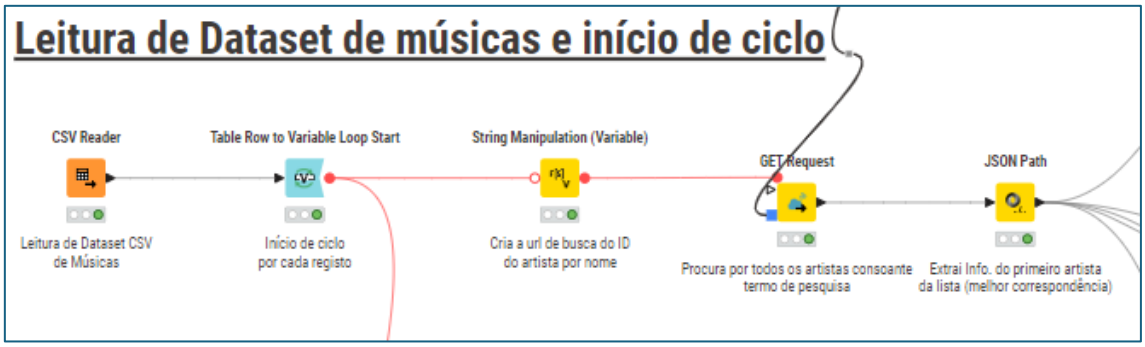


Figura 4 - Workflow de Leitura de dataset e início de ciclo.

Neste processo, é utilizado o node *CSV Reader* para efetuar a leitura do dataset que contém cerca de 30000 dados de músicas. Alguns dos dados que este possui são:

- track_id
  - Identificador de música.
- track_name
  - Nome da música.
- track_artist
  - Nome do artista.
- track_album_id
  - Identificador do álbum à qual a música pertence.
- track_album_name,
  - Nome do álbum à qual a música pertence.
- track_album_release_date,
  - Data de lançamento do álbum à qual a música pertence.
- danceability,
  - Atributo de 'dançabilidade' da música (escala de 0 a 1)
- energy
  - Atributo de 'energia da música (escala de 0 a 1)
- Etc.

Como cada linha do dataset representa uma música, a solução encontrada para a análise de dados é utilizar o node *Table Row to Variable Loop Start* para dar início a um ciclo que percorrerá todas as linhas do *dataset*, aplicando todos os próximos tratamentos de dados.

Apesar do *dataset* ter imensas informações úteis, reparei que para obtermos mais informações sobre o artista de uma música, seria necessário fazer um *GET Request* ao *Spotify*, utilizando o nome do artista, para obtermos o ID do artista. Para isto, foi utilizado o node *'String Manipulation (Variable)'* para gerar um URL dinâmico:

```
join("https://api.spotify.com/v1/search?q=",regexReplace($${Str
ack_artist}$$," ","%20"),"&type=artist&limit=1")
```

Figura 5 - Manipulação de string para URL de busca de ID de artista.

Esta manipulação permite-nos utilizar o conteúdo da coluna '*Track_Artist*' para gerar um link customizado que buscará o resultado do artista que melhor se adequa a esse nome, utilizando r*egex*.

Utilizamos o resultado desta transformação para utilizar o node *GET Request* para então executar o pedido. O resultado do pedido, em JSON, é de seguida utilizado pelo node *JSON Path* para podermos extrair as informações para diferentes colunas, que serão utilizadas nos próximos processos.

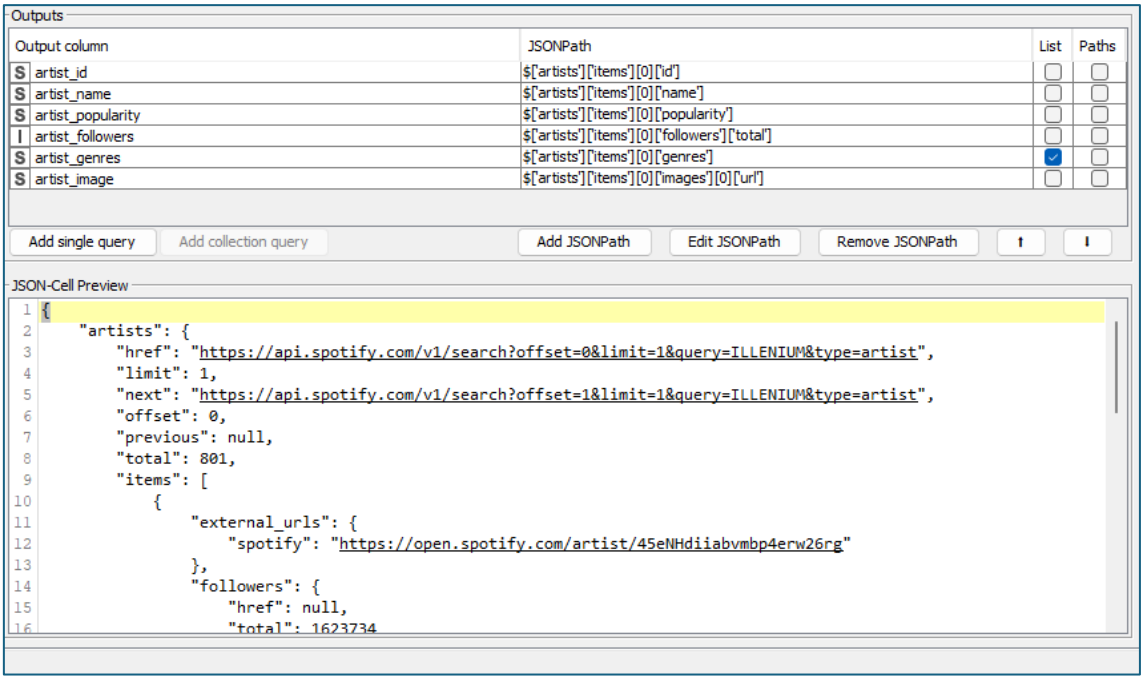


Figura 6 – Exemplo de processo de extração de dados de JSON para colunas.

artist_id String	artist_name String	artist_popularity String	artist_followers Number (Integer)	artist_genres Collection (List)	artist_image String
45eNHdiiabvmbp4erw26	ILLENIUM	71	1623734	[["melodic bass","future bass","edm"]]	https://i.scdn.co/image/a

Figura 7 - Exemplo de resultado de extração de dados de artista.

3. Tratamento de dados de géneros musicais de um artista

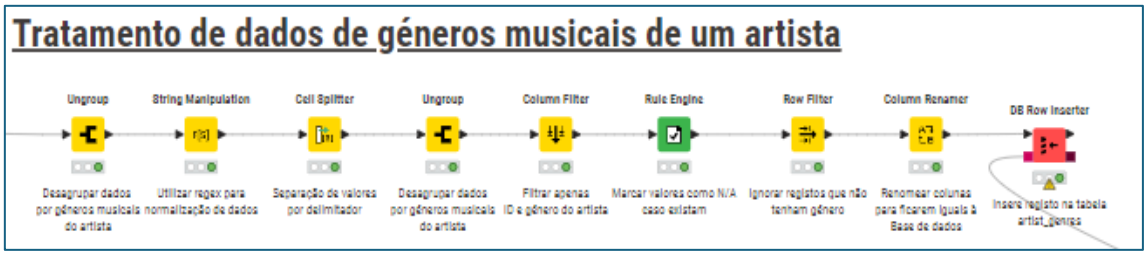


Figura 8 - Processo de tratamento de dados de géneros musicais.

Neste subprocesso, tratamos os dados do artista extraídos no processo anterior para podermos classificar facilmente quais são os géneros musicais que um artista produz.

Utilizamos o node *Ungroup* para desagrupar os géneros musicais de um artista (pois estes vêm como *Collection*(*list*)):

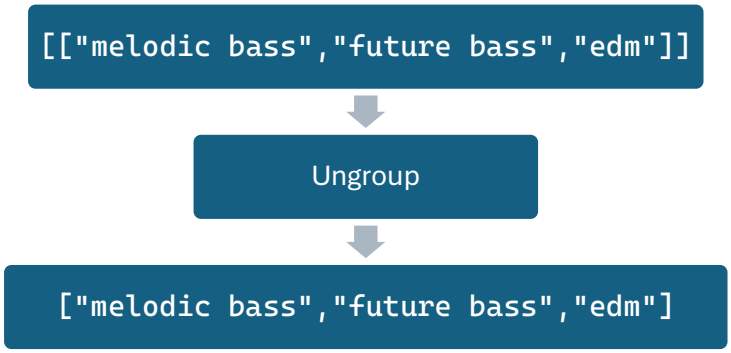


Figura 9 - Exemplo de tranformação Ungroup na lista artist_genres.

De seguida, normalizamos os dados com recurso ao node *String Manipulation* e uso de *regex*, para remover parêntesis retos. Utilizamos o *Cell Splitter* para tornar o valor novamente numa *Colletion(list),* delimitando no caractere ‘,’ e removendo as aspas:

```
[melodic bass, future bass, edm]
```

Figura 10 - Exemplo de lista transformada após Cell Splitter.

Utilizamos novamente o *Ungroup* para desagrupar o resultado, fazendo assim com que sejam geradas uma linha para cada género musical do artista:

artist_id String	artist_name String	artist_popularity String	artist_followers Number (Integer)	artist_image String	artist_genres_SplitR... String
45eNHdiiabvmbp4erw26rg	ILLENIUM	71	1623734	https://i.scdn.co/image/ab	melodic bass
45eNHdiiabvmbp4erw26rg	ILLENIUM	71	1623734	https://i.scdn.co/image/ab	future bass
45eNHdiiabvmbp4erw26rg	ILLENIUM	71	1623734	https://i.scdn.co/image/ab	edm

Figura 11 - Exemplo de resultado de Ungroup final de géneros musicais.

Para extrairmos apenas as informações essenciais da relação entre o artista e os géneros musicais, utilizamos o *Column Filter* para manter apenas o ID do artista e o género musical, e utilizamos o node *Rule Engine* para termos a certeza da validade dos dados, para não registar dados vazios.

```
MISSING
$artist_genres_SplitResultList$
=> "N/A"

$artist_genres_SplitResultList$ =
"[]" => "N/A"

TRUE =>
$artist_genres_SplitResultList$
```

Figura 12 - Rule Engine de géneros musicais.

Desta forma, temos a certeza que os dados que não tiverem valor são convertidos para "N/A", e os valores válidos mantêm-se.

Renomeamos então as colunas utilizando o *Column Renamer* para ficarem iguais ao nome das colunas na tabela '*artist_genres*' da base de dados, e finalmente inserimos os registos.

4. Tratamento de dados de géneros musicais de um artista

Figura 13 - Processo de tratamento de dados de álbuns.

Neste subprocesso, tratamos e extraímos os dados do álbum ao qual a música pertence.

Para obter informação extra à cerca do álbum, que não vem no *dataset*, utilizamos o *String Manipulation* para gerar um URL dinâmico de busca de dados, tendo em conta o identificador do mesmo:

```
join("https://api.spotify.com/v1/albums/",$${Strack_album_id}$$)
```

Figura 14 - Manipulação de string para URL de busca de dados extra de álbum.

Utilizamos a URL gerada no node *GET Request* junto com as credenciais para executar um pedido ao *Spotify* dos dados. Passando o corpo da resposta em formato JSON, utilizamos novamente o *JSON Path* para extraír a informação útil:

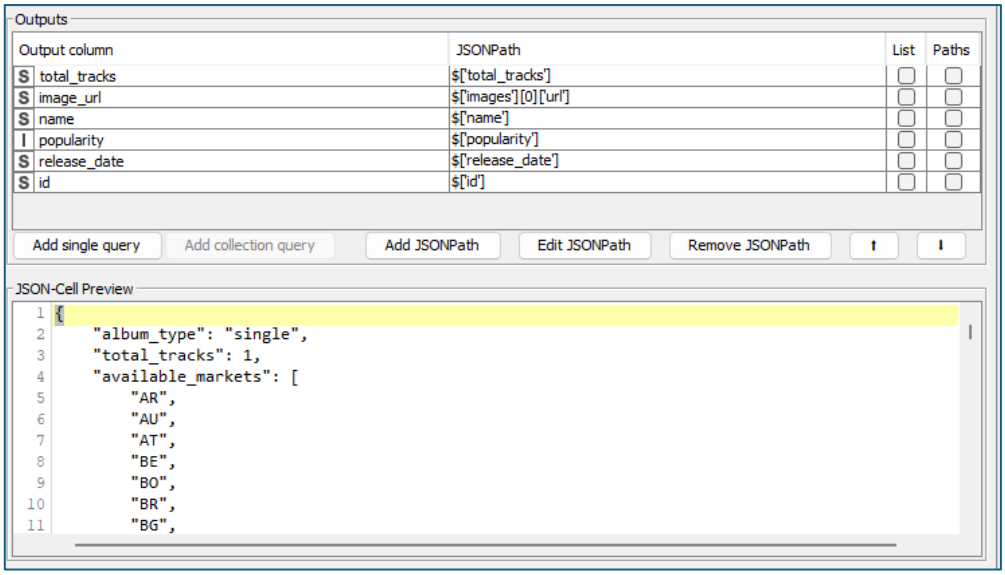


Figura 15 - Exemplo de extração de dados de resposta para colunas.

Filtramos então apenas a informação necessária relacionada com o álbum e inserimos os dados na base de dados, na tabela 'albums' utilizando o node *DB Row Inserter*.

Renomeamos então as colunas utilizando o *Column Renamer* para ficarem iguais ao nome das colunas na tabela *'artist_genres'* da base de dados, e finalmente inserimos os registos.

5. Tratamento de dados de artista

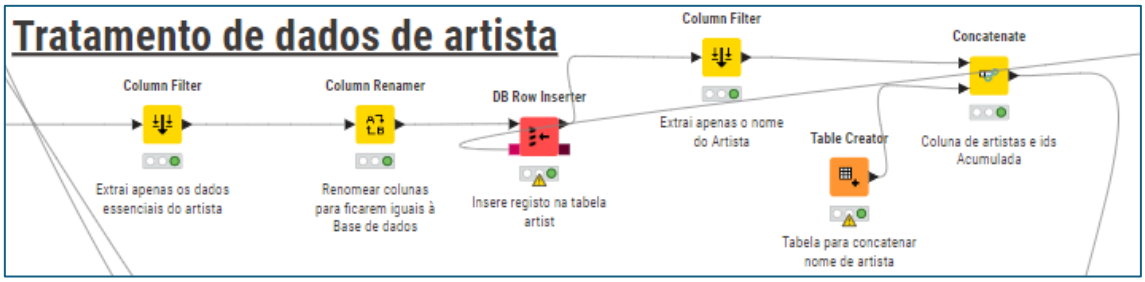


Figura 16 - Processo de tratamento de dados de artista.

Como anteriormente extraímos todos os dados necessários acerca do artista, neste subprocesso filtramos apenas as colunas relacionadas com o mesmo e renomeamo-las para ficarem com o nome idêntico à tabela '*artist*' na base de dados.

Após a inserção dos dados, extraímos apenas o identificador e o nome do artista para uma tabela interna acumulada, que utilizamos no fim no processo de envio de e-mail.

6. Tratamento de dados de músicas

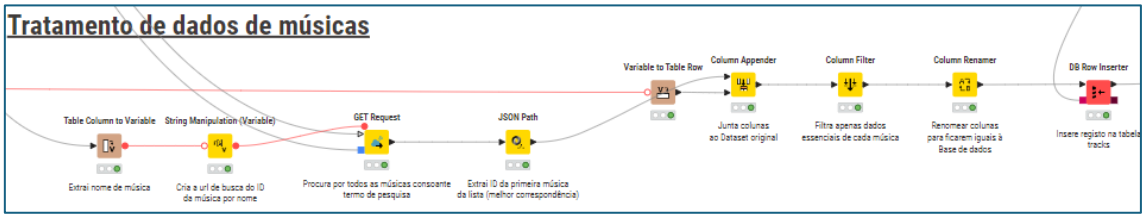


Figura 17 - Processo de tratamento de dados de músicas.

Neste subprocesso, utilizamos o nome da música fornecida pelo *dataset* para gerar um URL dinâmico para pesquisa de detalhes da música ao serviço do *Spotify*, semelhante ao processo do artista:

```
join("https://api.spotify.com/v1/search?q=",regexReplace($${Str
ack_name}$$," ","%20"),"&type=track&limit=1")
```

Figura 18 - Manipulação de string para URL de busca de detalhes da música.

Utilizamos o *JSON Path* para extraír o ID da música do resultado do corpo *do GET Request* efetuado:

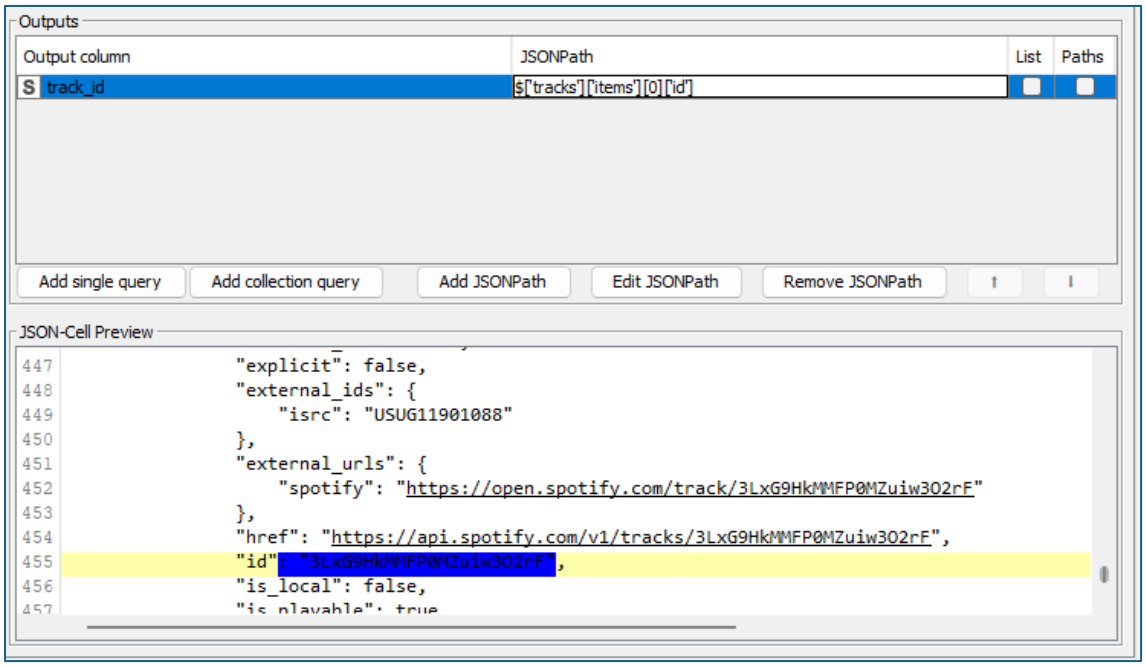


Figura 19 - Exemplo de extração de ID da música para coluna.

A este ponto, utilizamos o node *Variable to Table Row*, para efetuar uma 'cópia' da linha original desta iteração do ciclo, e acrescentar a nova coluna '*track_id*' com recurso ao *Column Appender*.

Feito isto, usamos o *Column Filter* para ficar apenas com as colunas relativas às músicas e renomeamos as colunas para ficarem de acordo com as colunas da tabela '*tracks*' da base de dados. Procedemos à inserção dos valores com o *DB Row Inserter*.

7. Fim de ciclo e processo final

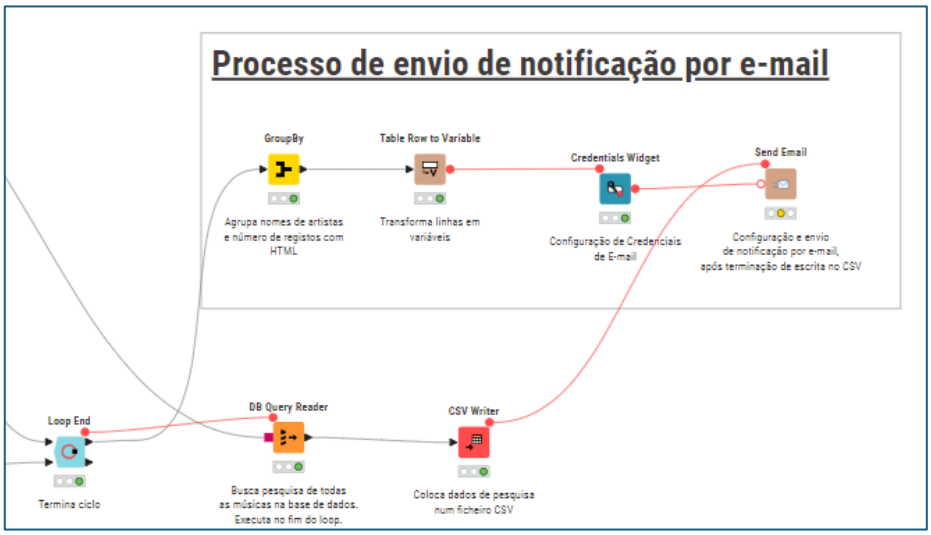


Figura 20 - Fim de ciclo e processo de notificação.

Após todos os registos do *dataset* (ou número desejado de linhas) passarem pela transformação e inserção dos dados, o ciclo termina e é iniciado o processo de envio de notificação via e-mail.

Em anexo do e-mail, é enviado um ficheiro CSV com todos os dados de músicas. Para isso, utilizamos o *DB Query Reader*, que efetua a seguinte consulta em *SQL*:

```
select a.name as 'Artista', b.name as 'Nome da música',
b.popularity as 'Popularidade'

from artists a, tracks b

where a.id = b.artist_id

order by b.popularity desc
```

Figura 21 - Consulta SQL de dados da base de dados.

Esta consulta devolve uma tabela com todos os dados do artista, nome da música e popularidade da música presentes na base de dados (não só os encontrados no momento).

O resultado é utilizado para gerar o anexo CSV 'musicas_encontradas.csv', utilizando o *CSV Writer*:

	A	B	C
1	Artista	Nome da música	Popularidade
2	Kygo	Higher Love	87
3	Billie Eilish	bad guy (with Justin Bieber)	83
4	Zara Larsson	All the Time - Don Diablo Remi	70
5	Avicii	Heaven - David Guetta & MOR	70
6	AJ Mitchell	Slow Dance (feat. Ava Max) - S	70
7	Lewis Capaldi	Someone You Loved - Future H	69
8	Sam Feldt	Post Malone (feat. RANI) - GAT	69
9	Sam Smith	Dancing With A Stranger (With	69
10	Avicii	Tough Love - Tiësto Remix / Ra	68
11	Avicii	SOS - Laidback Luke Tribute Re	68

Figura 22 - Exemplo de dados gerados pelo CSV Writer.

Ao mesmo tempo que esta função é realizada, utilizamos a função *Group By* para agregarmos os dados id e nome de artista com *Unique Concatenate* e *Unique Count* respetivamente para podermos ter uma lista de todos os artistas sem estarem repetidos (separados pelo elemento html <*br*>, para quebra de linha no e-mail) e o número de artistas.

id String	name String
6eUKZXaKkcviH0Ku9w2n3V	Ed Sheeran
04gDigrS5kc9YWfZHwBETP	Maroon 5
1Xylc3o4UrD53lo9CvFvVg	Zara Larsson
69GGBxA162lTqCwzJG5jLp	The Chainsmokers
4GNC7GD6oZMSxPGyXy4MNB	Lewis Capaldi
6eUKZXaKkcviH0Ku9w2n3V	Ed Sheeran
6jJ0s89eD6GaHleKKya26X	Katy Perry
20gsENnposVs2I4rQ5kvrf	Sam Feldt
1vCWHaC5f2uS3yhpwWbIA6	Avicii

Figura 23 - Exemplo de dados antes do Group By

Unique concatenate(name) String	Unique count(id) Number (Integer)
Ed Sheeran Maroon 5 Zara Larsson The Chainsmokers Lewis Capaldi Katy Perry Sam Feldt	55

Figura 24 - Exemplo de dados após Group By

Após a conversão, utilizamos o node *Table Row to Variable* para converter estes valores em variáveis reutilizáveis dentro do workflow, e utilizamos o *Credentials Widget* para que o utilizador consiga introduzir as suas credenciais de e-mail do IPCA (e-mail e password) para que estas sejam utilizadas no próximo node.

Para terminar, o node *Send E-mail* foi customizado para enviar uma notificação para o utilizador a avisar que o workflow foi terminado com sucesso, com o anexo de músicas da base de dados:

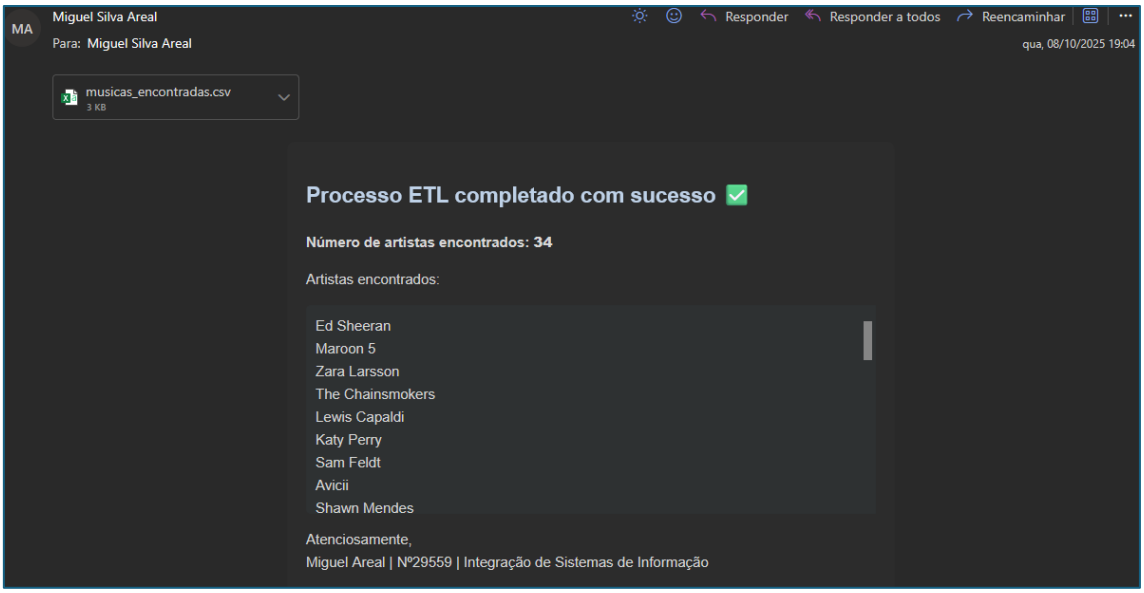


Figura 25 - Notificação via e-mail.

SQLite – Base de dados

Foi utilizado o *software* **DB Browser for SQLite**, uma ferramenta *open source* desenvolvida para criar, visualizar e gerir bases de dados em **SQLite**, uma versão leve, portátil e eficiente de sistemas de gestão de bases de dados relacionais.

Esta ferramenta permitiu criar tabelas, executar instruções *SQL* e verificar a integridade dos dados processados pelo *KNIME*, facilitando a análise e validação da informação resultante das transformações aplicadas no *workflow*.

É a partir desta fonte que os dashboards no *Node-Red* obtém os dados.

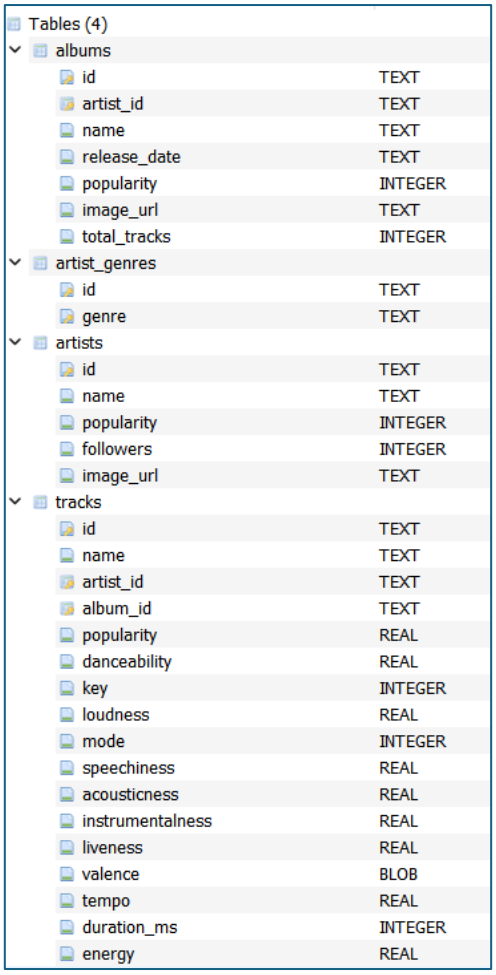


Figura 26 - Tabelas, atributos, e tipos de dados da base de dados TP1-ISI-Spotify

Node-Red - Dashboards

Para a visualização e análise dos dados transformados no KNIME, foram elaborados diferentes *dashboards*, com o intuito de simplificar a amostragem dos dados.

1. Artistas, álbuns e músicas
2. Popularidade de artistas e álbuns
3. Estatísticas de géneros musicais
4. Estatísticas de músicas

Para a elaboração das *dashboards*, foram utilizadas as bibliotecas externas:

- @flowfuse/node-red-dashboard
- node-red-node-sqlite

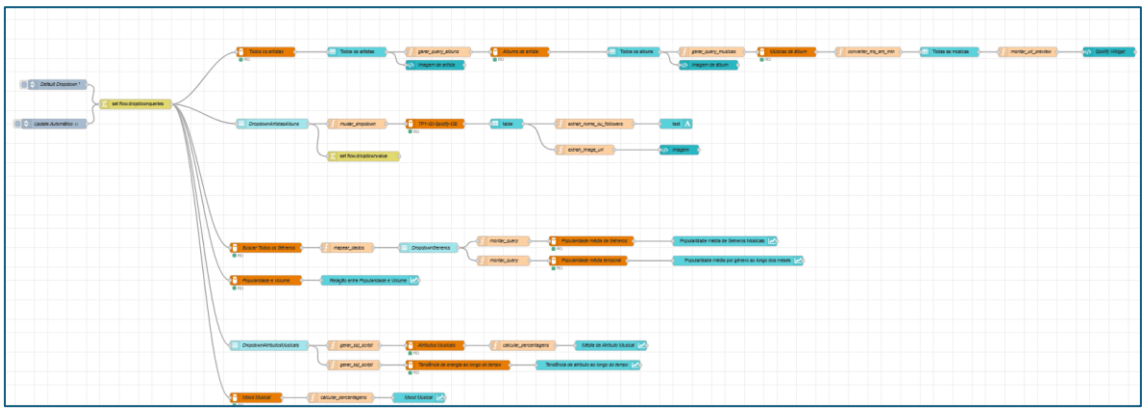


Figura 27 - Flow completo de todas as páginas e processos do Node-Red.

Para os dados serem utilizados, o *flow* é iniciado por dois *Inject nodes*:

1. *SetDefaultDropdown*
    a. Tem como objetivo no início do *flow* definir o valor pré-definido de '*DropdownArtistasAlbuns*', que pertence à segunda página.
2. *UpdateAutomático*
    a. Atualiza os dados da *dashboard* todos os minutos para garantir fiabilidade.

De seguida, existe um node *SetFlowQueries* que se responsabiliza por definir todos os scripts SQL utilizados de forma dinâmica noutros componentes, no formato JSON:

```
{
      "artistas": "SELECT id, ROW_NUMBER() OVER (ORDER BY
popularity DESC) as rank, name, popularity, image_url, followers
FROM artists ORDER BY popularity DESC"
}
```

Figura 28 - Exemplo de um dos scripts inseridos no JSON dinâmico.

1. Artistas, álbuns e músicas

Esta página demonstra todos os artistas, álbuns e músicas que constam na base de dados, ordenados de forma a fácil pesquisa e demonstração, de forma relacional:

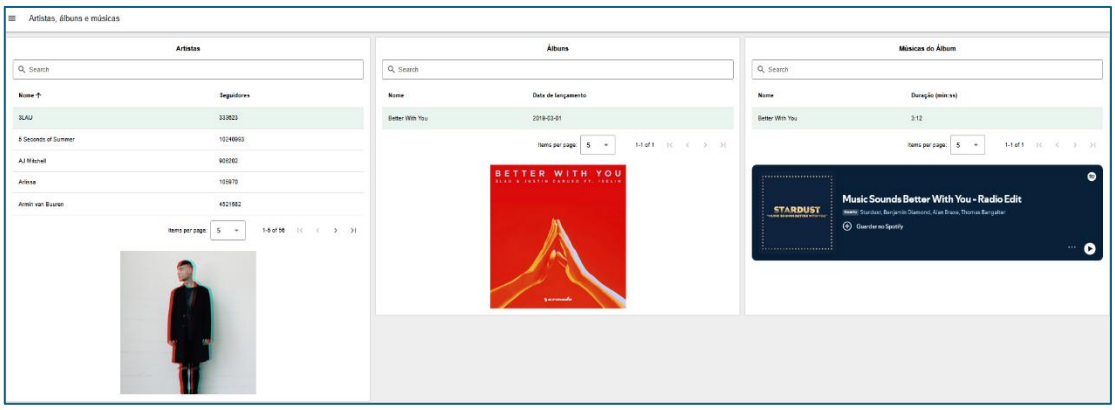


Figura 29 - Página de artistas, álbuns e músicas.

Composta por 3 grupos, aqui o utilizador pode pesquisar por um artista, verificar os álbuns que o mesmo produziu, e as músicas que constam no álbum. Pode também visualizar a capa de perfil do artista e álbum e também ouvir um excerto da música selecionada.

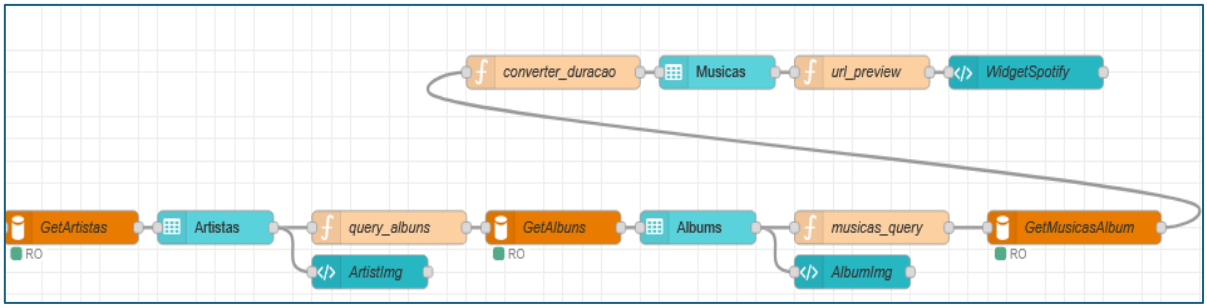


Figura 30 - Flow da página artistas, álbuns e músicas.

O *flow* é iniciado por um node *SQLite* (*GetArtistas*), responsável por executar uma consulta em busca de todos os artistas disponíveis na base de dados, seguido de um node *Table* (*Artistas*) que demonstra o nome do artista e número de seguidores no *Spotify*.

Ao clicar numa linha da tabela, é desencadeado um processo onde é gerada a *query* dinâmica (*query_albuns*) de busca dos álbuns pertencentes ao artista selecionado com um *Function* node, e um *Widget* (*ArtistaImg*) que mostra a imagem do artista, utilizando o URL da imagem do artista.

Após a geração da *query*, esta é executada pelo node *SQLite* (*GetAlbuns*), é demonstrada a segunda *table* (*Albuns*). O processo é semelhante, o utilizador seleciona um álbum, e é gerado um *Widget* com a imagem do álbum e um script SQL para busca de músicas pertencentes ao álbum.

Após a execução do *script*, utilizamos um *Function* node (converter_duracao) para converter o valor de '*duration_ms*' no formato 'min:ss' para melhor visualização. Estes dados são passados para a última tabela (Músicas), e é gerado o último URL, que serve para a pré-visualização da música no *Widget* (*WidgetSpotify*) através de HTML embebido.

2. Popularidade de artistas e álbuns

Esta página demonstra todos os artistas ou álbuns ordenados por ordem de popularidade, que permite pesquisar, ordenar resultados e visualizar imagens.

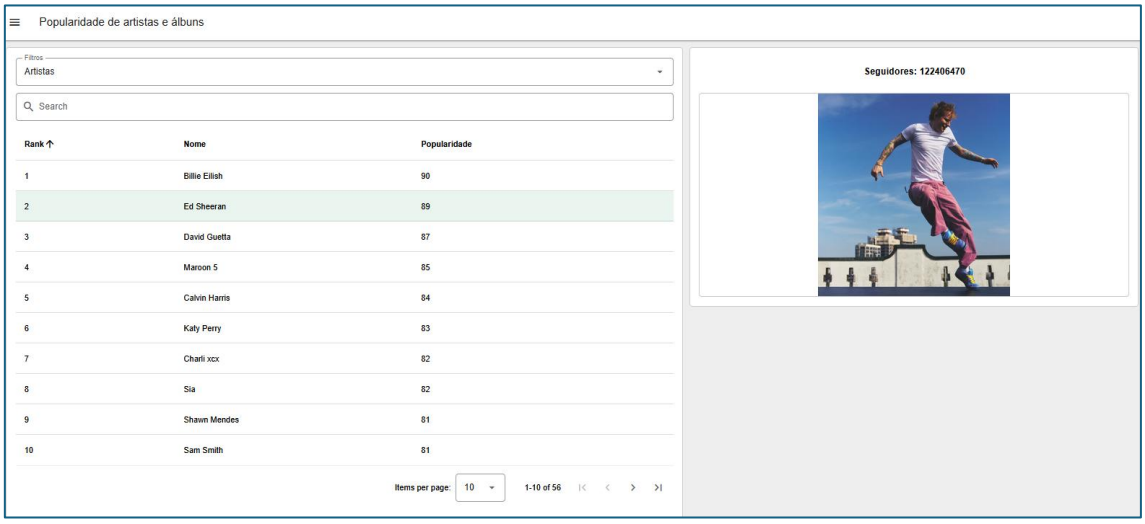


Figura 31 - Visualização de artistas.

Na consulta de artistas, ao selecionar um artista é disposta a imagem do artista do lado direito, com o número de seguidores em cima.

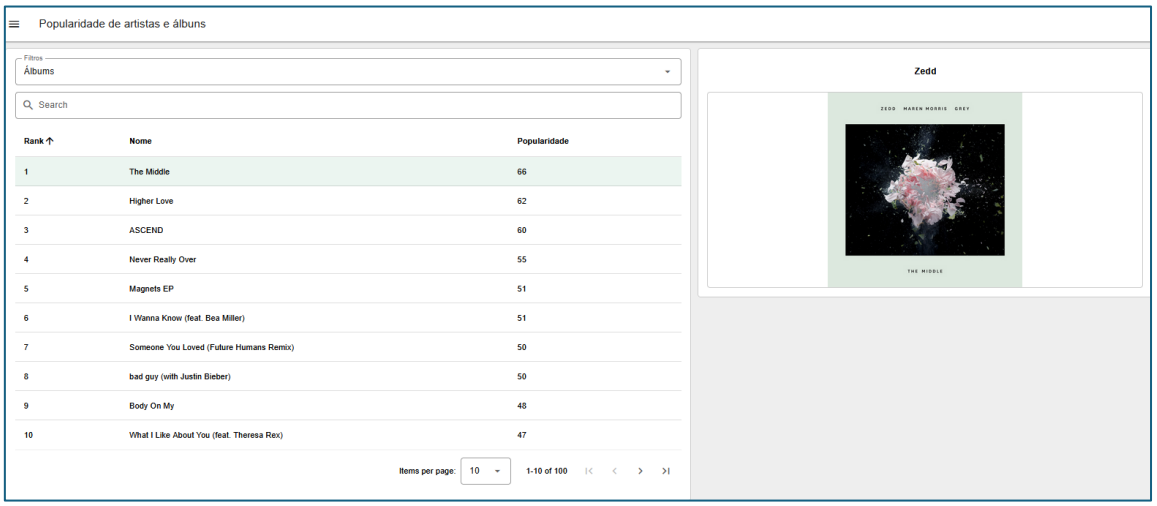


Figura 32 - Visualização de álbuns.

Na consulta de álbuns, ao selecionar um álbum é disposta a imagem da capa do álbum do lado direito, e o nome do artista criador do álbum em cima.

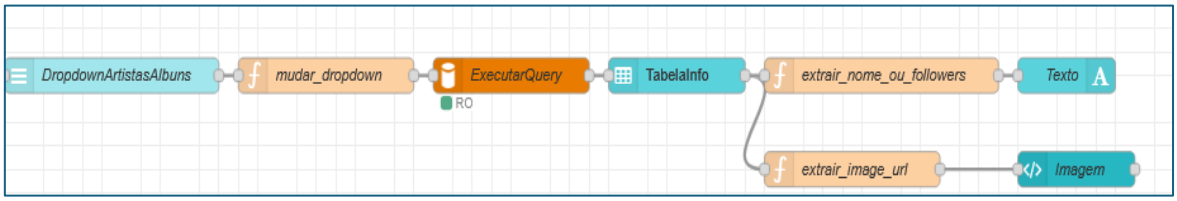


Figura 33 - Flow de página de Popularidade de artistas e álbuns.

Neste *flow*, é primeiro configurado uma *dropdown*, responsável por disponibilizar a lista fixa de filtros de pesquisa. É executada a função mudar_dropdown, que altera o script SQL a utilizar, dependendo do valor escolhido na lista anterior.

O script é executado no node ExecutarQuery e os dados são demonstrados no node TabelaInfo (*rank*, nome e popularidade). Ao selecionar uma linha, são executadas as funções extrair_nome_ou_followers e extrair_image_url.

A primeira extrai o nome do criador do álbum (caso a opção álbum esteja selecionada) ou seguidores (caso a opção artista esteja selecionada) para mostrar no node Texto.

A segunda função é responsável por extraír o atributo *image_url,* de modo a mostrar a imagem correspondente no *widget* Imagem.

3. Estatísticas de géneros musicais

Esta página demonstra algumas estatísticas pertinentes sobre a popularidade de géneros musicais.

Figura 34 - Página de estatísticas de géneros musicais.

Aqui é apresentada uma lista de valores de géneros musicais, que permite selecionar um ou mais géneros para comparação (afeta os primeiros dois gráficos). Se não for escolhido nenhum valor, todos os géneros são tomados em conta para apresentação de dados. Existem três gráficos, cada um deles representando uma estatística diferente:

1. Popularidade média de géneros musicais
    a. Permite visualizar de forma geral e intemporal quais são os géneros musicais predominantes do *dataset* gerado.
2. Popularidade média por género ao longo dos meses
    a. Permite verificar dados de forma temporal, comparando a relação entre a popularidade média de um certo dado, dado o mês e ano.
3. Relação entre popularidade e volume
    a. Permite responder à questão: Existe relação entre volume e popularidade? De forma a entender se mais quantidade significa maior sucesso ou vice-versa.

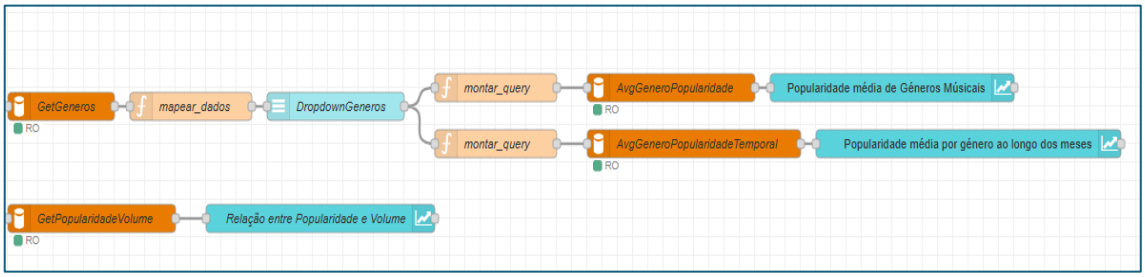


Figura 35 - Flow da página de estatísticas de géneros musicais.

O *flow* para esta página é dividido em duas partes, a primeira para os dados dos dois gráficos no topo e o segundo *flow* para o terceiro gráfico.

Para o primeiro *flow*, primeiro é utilizado um node *SQLite* (*GetGeneros*) para popular a lista de valores de géneros para o utilizador manipular os resultados. Depois são formulados os scripts SQL dinâmicos utilizando node de *Function*. Um node para cada gráfico. Estes scripts SQL utilizam as opções selecionadas para ir buscar os dados necessários à base de dados automaticamente, e demonstrar os resultados no gráfico de barras e no gráfico de linha.

O segundo *flow* é mais simples, é utilizado diretamente um node *SQLite* (*GetPopularidadeVolume*) para buscar os dados: *genre*, *track_count* e *avg_popularity* através de uma consulta SQL. Os dados são então visualizáveis pelo gráfico de dispersão.

4. Estatísticas de músicas

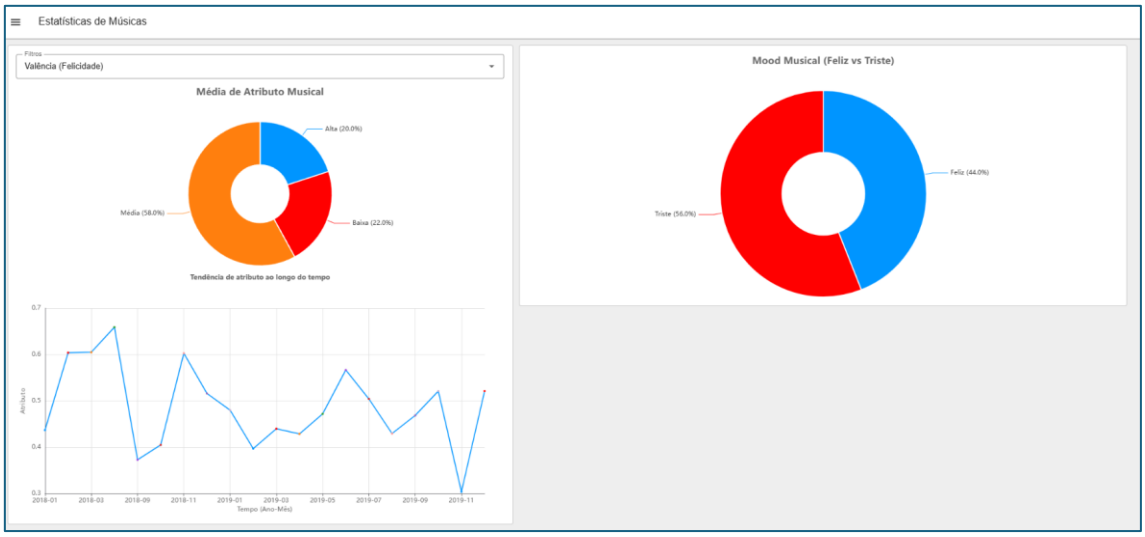


Figura 36 - Página de estatísticas de géneros musicais.

Esta página dispõe de estatísticas de músicas, mais propriamente em relação aos atributos que estas possuem, e que podem ser valores entre zero e um. Estes são:

Atributo	Descrição
Volume	Nível geral de som da música, medido em decibéis (dB).
Danceability	Mede quão apropriada a música é para dançar. Baseia-se em ritmo, estabilidade, batida e regularidade.
Loudness	Média da intensidade sonora da música. Também medido em decibéis (dB).
Speechiness	Mede quanto da música é composta por fala ao invés de música.
Acousticness	Indica quão acústica a música é.
Instrumentalness	Mede quanto da música é instrumental (sem vocais).
Liveness	Indica a probabilidade de a música ter sido gravada ao vivo ou com público.
Valence	Mede a “emoção” da música, ou quão positiva/feliz ela soa.
Energy	Mede intensidade e atividade da música. Leva em conta volume, ritmo, timbre e perceção de movimento.

Figura 37 - Legenda de atributos de músicas.

Estes atributos podem ser filtrados utilizando a lista de valores presente na página. Esta afeta o primeiro gráfico circular e o gráfico de linha.

O primeiro gráfico circular representa média desse atributo perante a dimensão de dados recolhidos. Ao passar o rato por cima, é possível verificar a percentagem dos dados e a quantidade de músicas que foram contabilizadas para tal.

O segundo gráfico circular "*Mood* musical" é uma estatística relativa à média do atributo valência. Inferior a 0.5, a música é considerada "triste", e acima é considerada "Feliz".

O terceiro gráfico, de linha, representa a média da tendência da presença do atributo ao longo do tempo, para o utilizador ter uma perceção da alteração de modas perante o tempo.

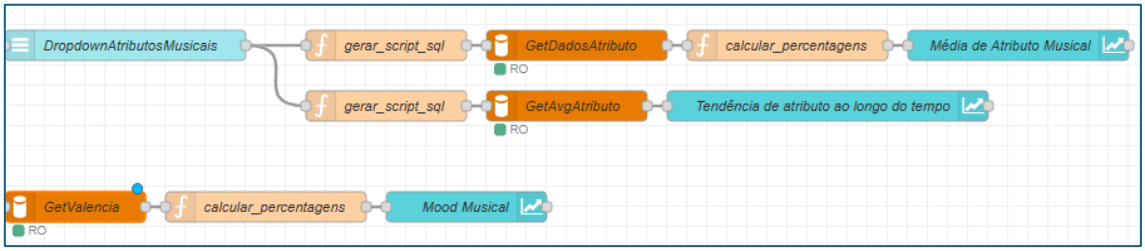


Figura 38 - Flow da página de estatísticas de músicas.

Novamente esta página utiliza dois pequenos *flows*, o primeiro para os gráficos que utilizam a lista de valores, e o segundo para o gráfico de "*Mood* musical".

Os atributos presentes na lista de valores são colocados manualmente, de forma a existir controlo sobre quais são os dados a utilizar. São então utilizadas dois *Function* nodes para gerar os scripts dinâmicos em SQL que serão utilizados para ir buscar os dados necessários à base de dados, um para cada gráfico.

Para o gráfico circular permitir visualizar a percentagem de registos que influenciam a análise, é utilizado um *function* node (calcular_percentagens)

Para o gráfico de "*Mood* musical", é utilizado o *SQLite* node (*GetValencia*) para obter os dados agrupados pelo atributo, e é calculado posteriormente a percentagem afetada para visualização na *dashboard*.

Vídeo de demonstração

Figura 39 - Ligação para vídeo de demonstração.

Link direto

Conclusão

Este trabalho permitiu compreender de forma prática como funciona um processo ETL e como as várias etapas: extração, transformação e carga, se interligam dentro do *KNIME*. Foi possível criar um workflow completo que lê dados, valida e trata a informação, e exporta os resultados em diferentes formatos (CSV e Base de dados).

A automatização do processo com *KNIME* permitiu gerar ficheiros, enviar notificações por e-mail e garantir consistência nos dados, tornando o sistema mais fiável. A integração com *SQLite* demonstrou como estruturar e gerir dados de forma organizada, facilitando consultas SQL e posterior análise.

Adicionalmente, a integração com *dashboards* no *Node-RED* permitiu visualizar e analisar os dados de forma interativa, facilitando a interpretação dos resultados e mostrando tendências relevantes no *dataset* musical.

Em relação ao problema simulado, acredito que a minha solução seja válida e bastante útil, sendo uma ferramenta que permite analisar os dados de forma a visar pelos interesses da empresa simulada.

No geral, o trabalho consolidou conhecimentos técnicos em integração de dados, manipulação de APIs, transformação de dados e visualização analítica, oferecendo uma visão completa do ciclo de vida da informação.

Trabalhos Futuros

Para evoluir este projeto, algumas melhorias e extensões podem ser implementadas:

1. Otimização do *workflow*
    - Melhorar a performance do *KNIME* para processar *datasets* ainda maiores ou mais complexos.
2. Dashboards mais avançados
    - Criar filtros dinâmicos, gráficos interativos e análises mais detalhadas no *Node-RED*.
3. Automatização adicional
    - Implementar alertas automáticos ou notificações baseadas em condições específicas nos dados processados.
4. *Streaming* de dados
    - Implementar sistemas com diferentes tecnologias para demonstrar conhecimentos de *streaming* de dados em tempo real com integração no *KNIME* ou *Node-Red*.

Bibliografia

- KNIME Analytics Platform - https://www.knime.com
- Node-RED - https://nodered.org/
- SQLite - https://www.sqlite.org/index.html
- DBBrowser for SQLite - https://sqlitebrowser.org/
- Spotify for Developers (API) – https://developer.spotify.com/
- Joe Beach Capital (2021). *30,000 Spotify Songs Dataset*. Kaggle. https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs