****

**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**

[**Glossário** 4](#_Toc211251811)

[**Introdução** 5](#_Toc211251812)

[**Contexto** 6](#_Toc211251813)

[**Objetivos** 7](#_Toc211251814)

[**Estratégia** 8](#_Toc211251815)

[**Dados** 9](#_Toc211251816)

[**Processo ETL** 10](#_Toc211251817)

[1. Autenticação na API do Spotify 10](#_Toc211251818)

[2. Leitura de Dataset 11](#_Toc211251819)

[**Transformação 1** 11](#_Toc211251820)

[**Jobs** 14](#_Toc211251821)

[**Transformação 2** 17](#_Toc211251822)

[**Power BI** 19](#_Toc211251823)

[**Conclusão** 21](#_Toc211251824)

[**Trabalhos Futuros** 22](#_Toc211251825)

[**Bibliografia** 23](#_Toc211251826)

# **Índice de Figuras**

[Figura 1 - Variável que guarda os dados no ficheiro CSV 9](#_Toc210858039)

[Figura 2 - Ficheiros carregados no Pentaho 9](#_Toc210858040)

[Figura 3 - Dados do ficheiro csv 10](#_Toc210858041)

[Figura 4 - Transformação 11](#_Toc210858042)

[Figura 5 - Expressão Regular 11](#_Toc210858043)

[Figura 6 - Novo campo e distinção da data 12](#_Toc210858044)

[Figura 7 - Linha que foram para os ficheiros 12](#_Toc210858045)

[Figura 8 - Validação 12](#_Toc210858046)

[Figura 9 - Linhas apenas datas válidas 13](#_Toc210858047)

[Figura 10 - Exemplo output do ficheiro xml 13](#_Toc210858048)

[Figura 11 - job para gerar texto 14](#_Toc210858049)

[Figura 12 - Texto gerado a partir do job 14](#_Toc210858050)

[Figura 13 - Job completo 14](#_Toc210858051)

[Figura 14 - Confirmação 15](#_Toc210858052)

[Figura 15 - Caixa de entrada do email 15](#_Toc210858053)

[Figura 16 - Texto do email 15](#_Toc210858054)

[Figura 17 - Configuração 15](#_Toc210858055)

[Figura 18 - Configuração 16](#_Toc210858056)

[Figura 19 - Configuração 16](#_Toc210858057)

[Figura 20 - Transformação 17](#_Toc210858058)

[Figura 21 - Dados do ficheiro Clientes 17](#_Toc210858059)

[Figura 22 - Dados do ficheiro Vendas 17](#_Toc210858060)

[Figura 23 - Fórmulas aplicadas 18](#_Toc210858061)

[Figura 24 - Ficheiro gerado em xml 18](#_Toc210858062)

[Figura 25 - Dados retirados do xml alterado 19](#_Toc210858063)

[Figura 26 - Gráfico com as temperaturas registadas 20](#_Toc210858064)

[Figura 27 - Cidades e temperaturas 20](#_Toc210858065)

# **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:**
* **GET Requests:**
* **Endpoint:**
* **URL:**

# **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 usando Dados;

- Analisar e especificar cenários de aplicação de processos de ETL;

- Explorar ferramentas de suporte a processos de ETL;

- Explorar novas Tecnologias, Frameworks ou Paradigmas;

- Potenciar a experiência no desenvolvimento de software;

- Facilitar a assimilação do conteúdo da Unidade Curricular.

- Contribuir para qualidade da escrita de relatórios

# **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.
  + Extraído a partir de <https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs>
* 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**

## 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:

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

{

    "client\_id":"12345",

    "client\_secret":"12345"

}

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.*

## 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:

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

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

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.

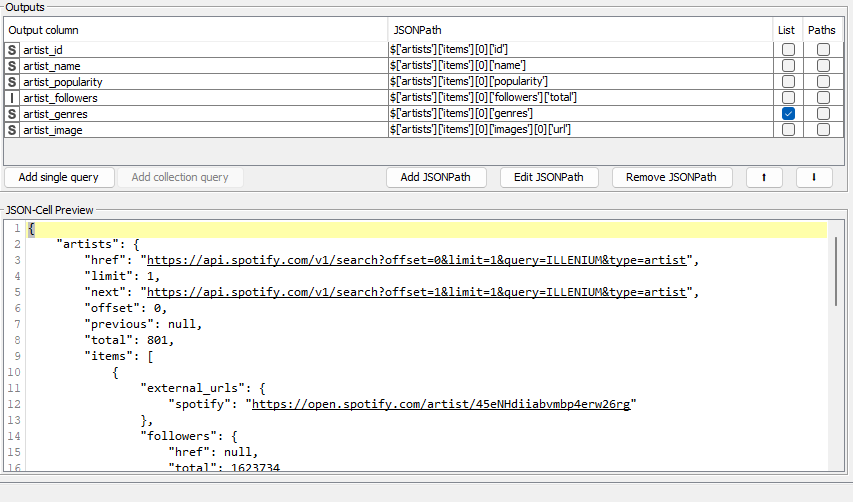


Figura 6 – Exemplo de processo de extração de dados de JSON para colunas.



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

## 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:

Figura 10 - Exemplo de lista transformada após Cell Splitter.

[melodic bass, future bass, edm]

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

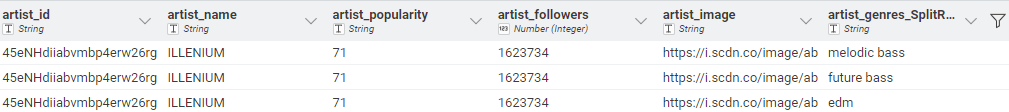


Figura 11 – Exemplo de resultado de Ungroup final de géneros musicais.

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.

Figura 12 - Rule Engine de géneros musicais.

MISSING $artist\_genres\_SplitResultList$ => "N/A"

$artist\_genres\_SplitResultList$ = "[]" => "N/A"

TRUE => $artist\_genres\_SplitResultList$

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.

## 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:

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

join("https://api.spotify.com/v1/albums/",$${Strack\_album\_id}$$)

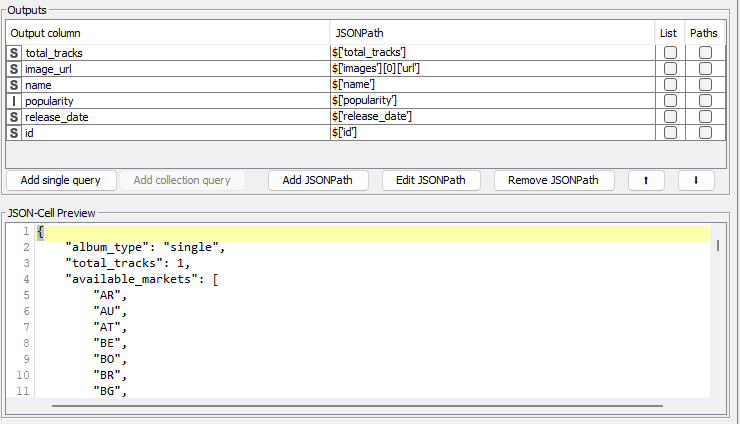
Utilizamos a URL gerada no node GET Request junto com as credenciais para executar um pedido ao Spotify dos dados. Passando o corpo da resposta em formato JSON, utilizamos novamente o JSON Path para extraír a informação útil:

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.

## Tratamento de dados de artista

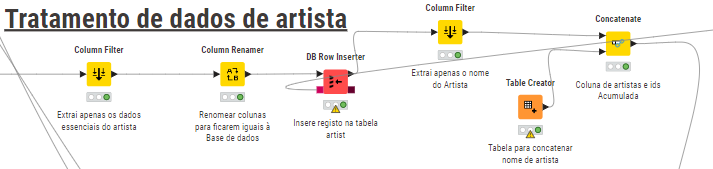


Figura 16 - Processo de tratamento de dados de artista.

Como no 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.

## Tratamento de dados de músicas

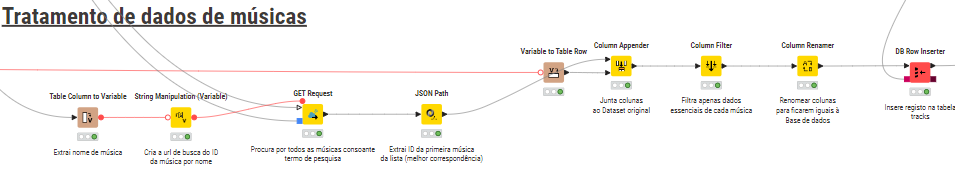


Figura 17 - Processo de tratamento de dados de músicas.

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:

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

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

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

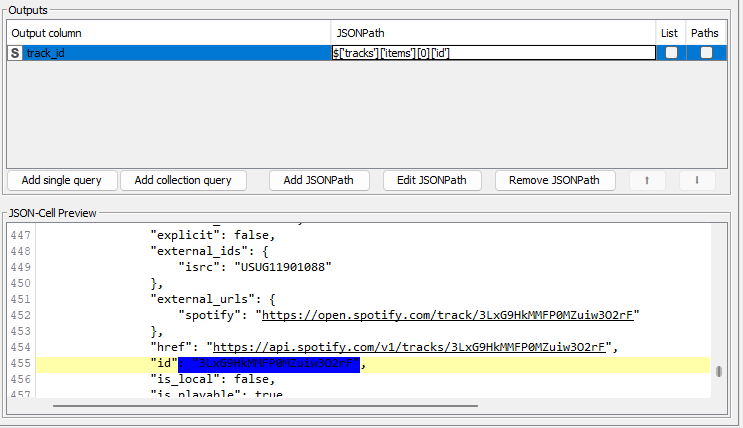
 A este ponto, utilizamos o node Variable to Table Row, para efetuar uma ‘cópia’ da linha original desta iteração do ciclo, e acrescentar a nova coluna ‘track\_id’ com recurso ao Column Appender.

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

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.

## 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:

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

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

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).

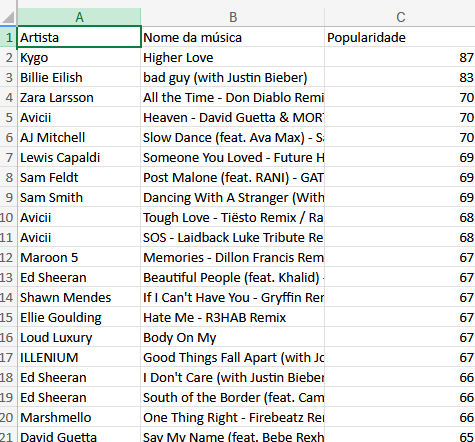
O resultado é utilizado para gerar o anexo CSV ‘musicas\_encontradas.csv’, utilizando o CSV Writer:

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.

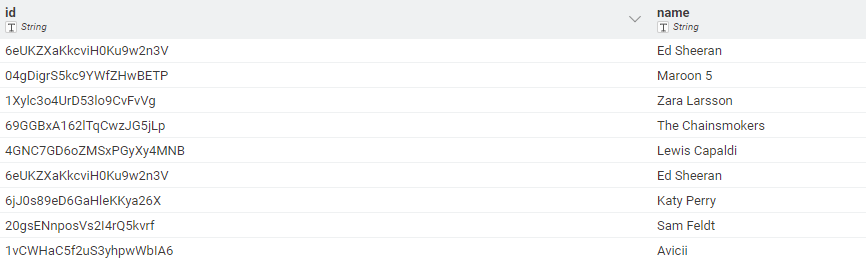


Figura 23 – Exemplo de dados antes do Group By



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:

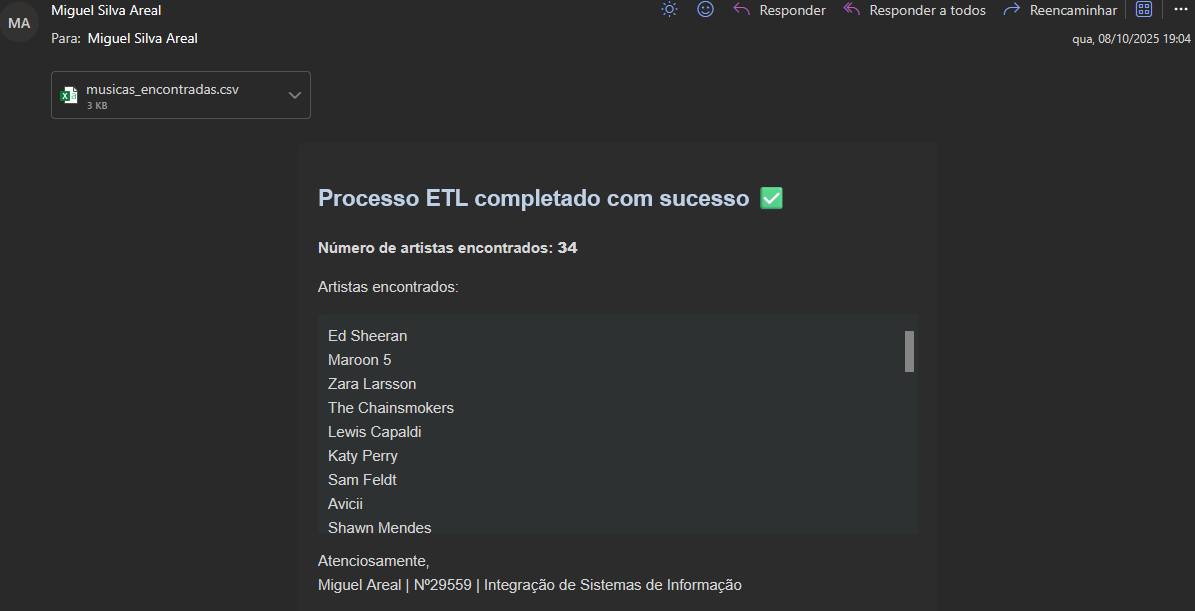


Figura 25 - Notificação via e-mail.

# **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](https://flows.nodered.org/node/@flowfuse/node-red-dashboard)
* [node-red-node-sqlite](https://flows.nodered.org/node/node-red-node-sqlite)

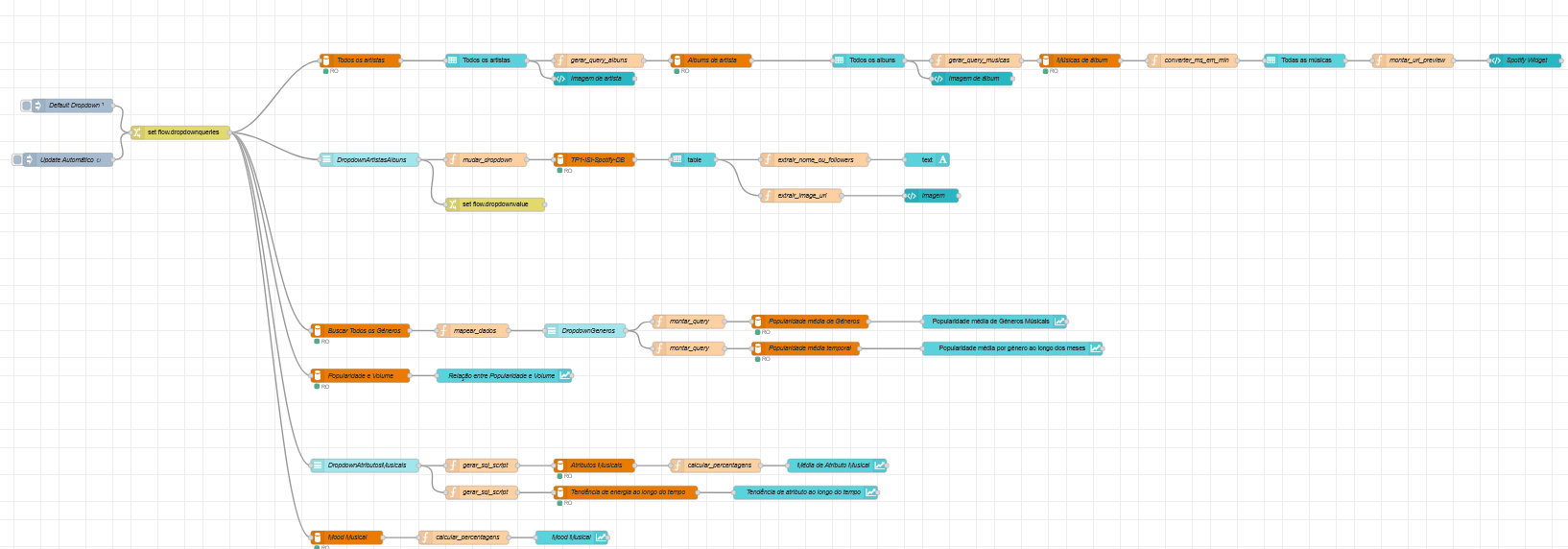


Figura 26 - Flow completo de todas as páginas e processos do Node-Red.

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

1. SetDefaultDropdown
   1. Tem como objetivo no início do flow definir o valor pré-definido de ‘DropdownArtistasAlbuns’, que pertence à segunda página.
2. UpdateAutomático
   1. 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:

Figura 27 - Exemplo de um dos scripts inseridos no JSON dinâmico.

{

    "artistas": "SELECT id, ROW\_NUMBER() OVER (ORDER BY popularity DESC) as rank, name, popularity, image\_url, followers FROM artists ORDER BY popularity DESC"

}

## 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:

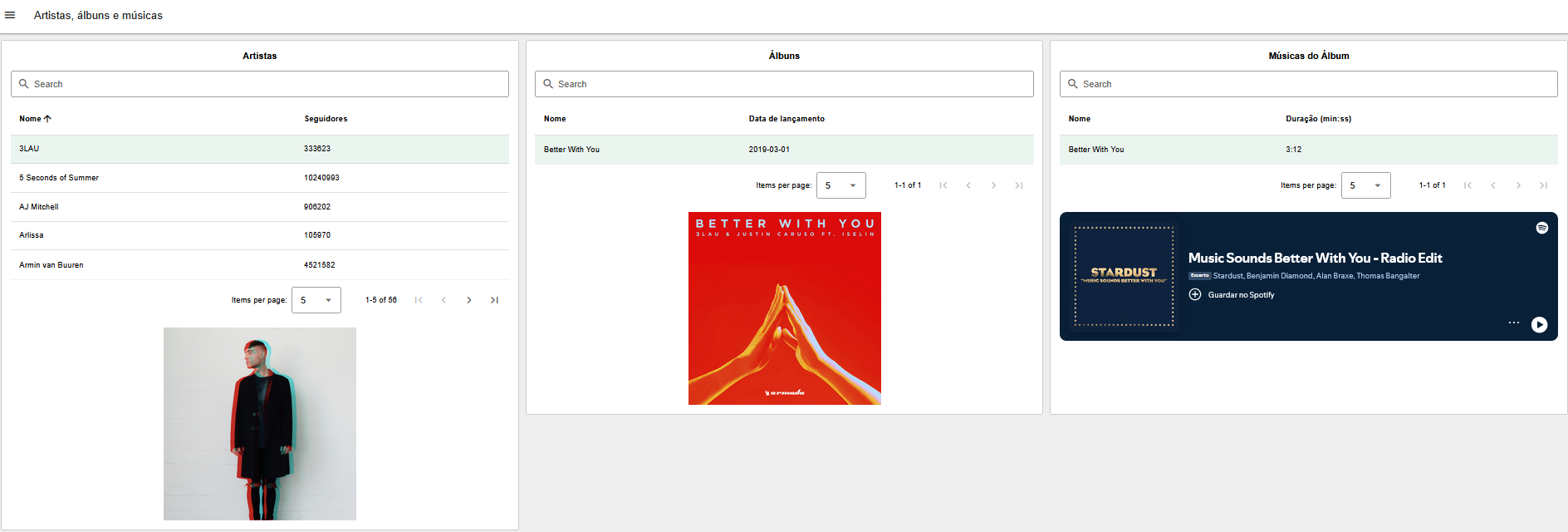


Figura 28 - Página de artistas, álbuns e músicas.

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.

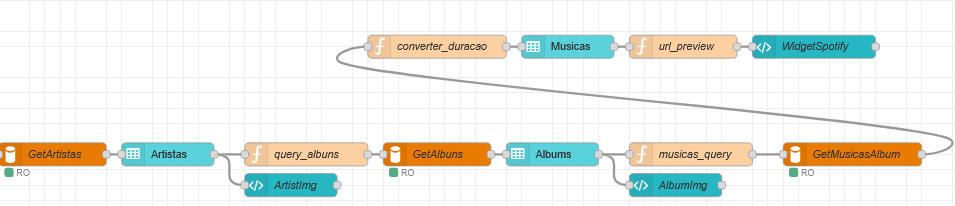


Figura 29 - Flow da página artistas, álbuns e músicas.

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 (Musicas), e é gerado o último URL, que serve para a pré-visualização da música no Widget (WidgetSpotify)

## 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.

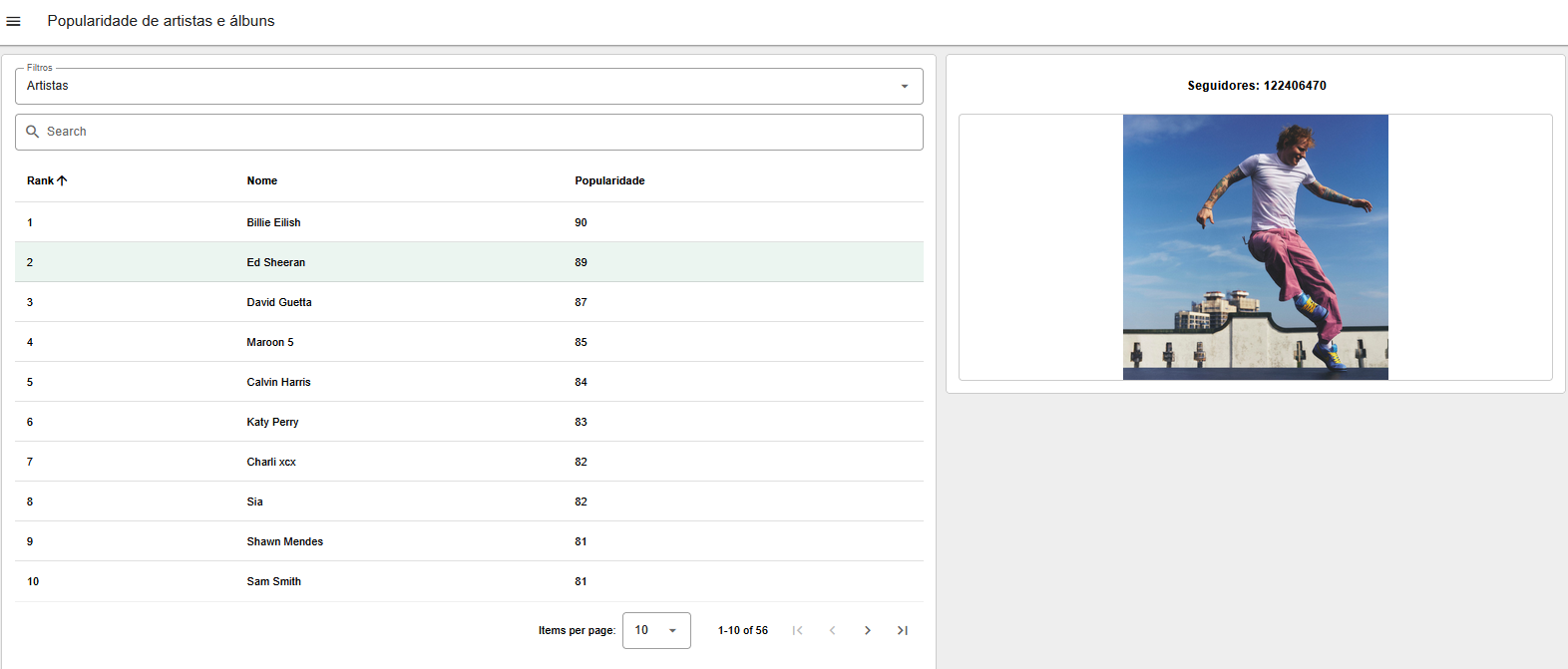


Figura 30 - Visualização de artistas.

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

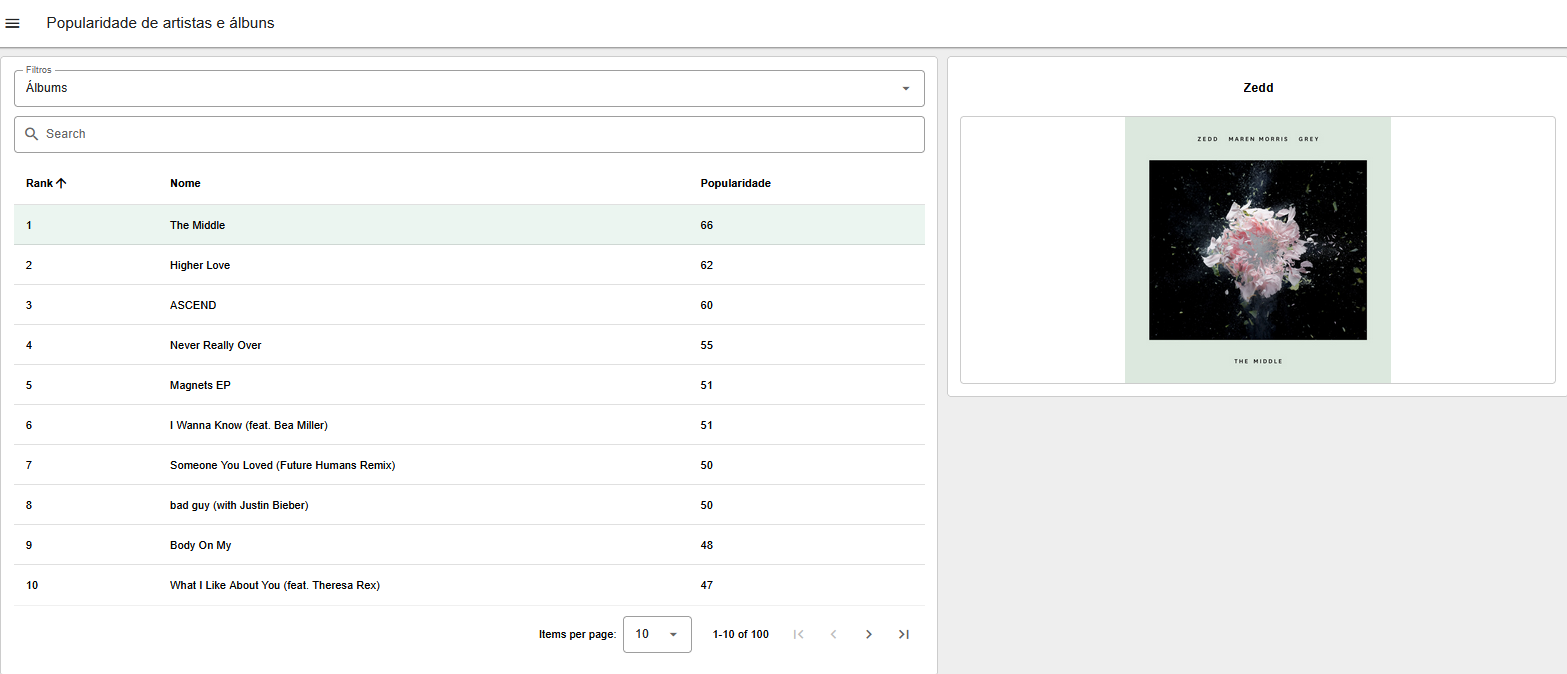


Figura 31 - Visualização de álbuns.

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.

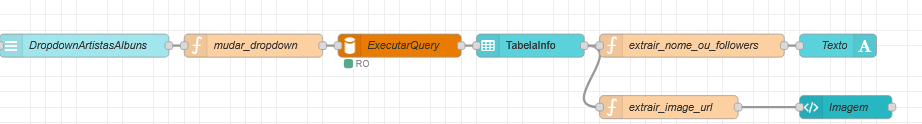


Figura 32 - Flow de página de Popularidade de artistas e álbuns.

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.

# **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 uma transformação completa que lê dados, valida e trata a informação, e exporta os resultados em diferentes formatos (CSV e Base de dados).

A criação do workflow automatizado permitiu controlar todo o processo, gerar ficheiros e enviar notificações por e-mail, tornando o sistema mais fiável útil.

A integração de bases de dados exemplificou a formatação e validação de dados úteis, redundantes, e de fácil acesso e consulta via SQL.

Por fim, a ligação ao Node-Red mostrou a importância da análise visual dos dados, ajudando a transformar os resultados do ETL em informação útil e dedutiva.

No geral, o trabalho foi bastante enriquecedor, pois juntou a parte técnica da integração de dados com a parte analítica, permitindo ter uma visão completa do ciclo de vida da informação.

# **Trabalhos Futuros**

Tendo em conta o pouco tempo que tive para realizar este trabalho, pretendo melhorar o mesmo e implementar todas as funcionalidades que faltam relativamente ao que foi pedido.

De resto, existe sempre espaço para melhorar e para evoluir.

# **Bibliografia**

- https://elearning.ipca.pt/2526/course/view.php?id=38561

- https://pentaho.com/pentaho-ee-onprem/

- <https://www.microsoft.com/pt-pt/power-platform/products/power-bi?market=pt>

- https://support.microsoft.com/pt-pt/office/xml-para-principiantes-a87d234d-4c2e-4409-9cbc-45e4eb857d44