Trabalho Prático 1

Integração de Sistemas de Informação

ESIPL

Aluno:

João Apresentação 21152

Docente:

Óscar Ribeiro

Barcelos, Portugal

15 de Novembro de 2020

Resumo

Este projeto inserido na unidade curricular de Integração de Sistemas de Informação (ISI) pretende a demonstração de aptidões adquiridas e exploração das ferramentas fornecidas com foco em processos ETL (Extract, Transformativo and Load) na manipulação dos dados.

Todo este processo é desenvolvido recorrendo ao uso das ferramentas: Pentaho Kettle, Mongo DB e Visual Studio 22.

Índice

Introdução

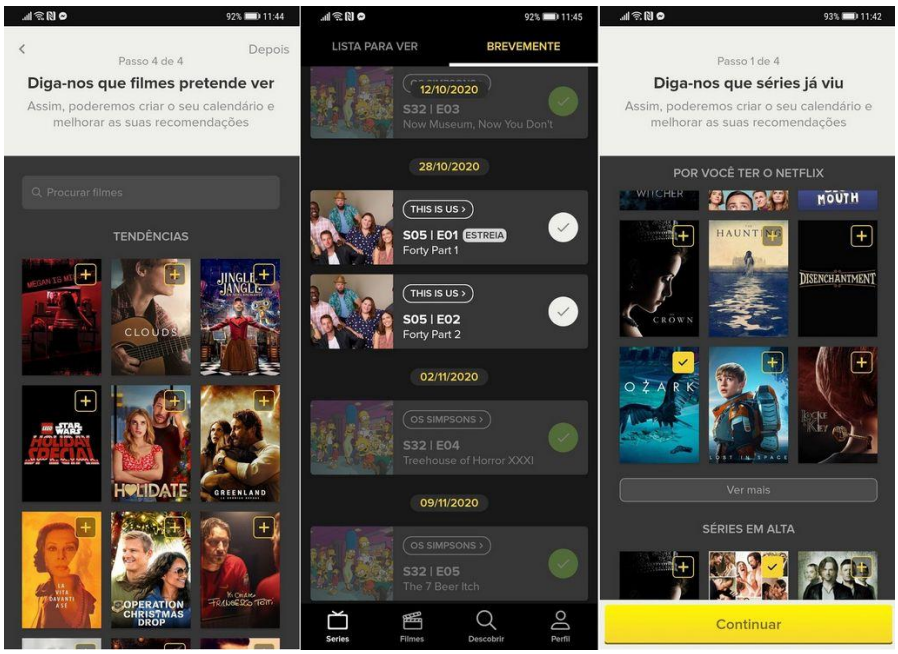
Neste ponto será apresentado um resumo do desenvolvimento, enquadramento e tema do projeto, além dos seus objetivos e ferramentas utilizadas.

Resumo

Para o desenvolvimento deste projeto foi selecionado dois datasets em formato csv do "TV Time" carregado através do site Kaggle contido no seguinte link:

(HTTPS://WWW.KAGGLE.COM/DATASETS/OSCARFRY/TVTIME-SHOWS?SELECT=TVTIMESHOWS.CSV)

TV Time é um serviço online que permite aos seus utilizadores marcar que assistiu uma determinada serie/episódio e gerir o que vai assistindo, avaliar os episódios, entre muitas outras funcionalidades.



1 - **TV Time App**

Os dois datasets baixados representarão:

- all_episodes.csv (contém todos os episódios)
- tvtimeshows.csv (contém todas as séries)

Objetivos

- Manipulação dos dados (Datasets);
    - Inserção
    - Transformação
    - Extração
- Uso de expressões regulares;
- Envio dos dados para uma base de dados;
- Demonstração dos conteúdos através de uma interface.

Ferramentas

- Pentaho Kettle: manipulação dos dados vindos dos datasets (transformation e job);
- Mongo DB: receber os dados extraídos da transformation no Kettle;
- Visual Studio 22: programa de navegação de ficheiros xml (no caso será um dos ficheiros extraídos do kettle).

Implementação

Neste ponto será demonstrado e explicado todos os passos para a construção da solução desenvolvida.

Estrutura dos Datasets

Para este projeto foi utilizado dois datasets com formatos csv, separados por virgulas e apresentando alguns elementos com alguma necessidade de transformação para uma devida extração. Serão agora mostradas as duas estruturas extraídas.

Todos os episódios

O ficheiro all_episodes.csv representa todos os episódios associados a uma determinada série, sendo que este contém todas as séries, mas tratando mais especificamente cada episódio. Este contém:

- show_id – id da série;
- episode_id – id do episódio;
- time – data de lançamento do episódio;
- episode – nº da temporada e nº do episódio;
- title – título do episódio;
- description – descrição do episódio;
- rating – avaliação do episódio;
- times_watched – vezes que o episódio foi assistido;
- mood-good – número de votos para humor bom do episódio;
- mood-fun – número de votos para humor divertido do episódio;
- mood-wow – número de votos para humor “uau” do episódio;
- mood-sad – número de votos para humor triste do episódio;
- mood-so-so – número de votos para humor medíocre do episódio;
- mood-bad – número de votos para humor mau do episódio.

2 - TV Time voto de humor

Todas as séries

O ficheiro tvtimesshows.csv representa todas as séries. Este contém:

- id – id da série;
- name – nome da série;
- followers – seguidores da série (pessoas que marcaram a série);
- nb_rates – número de avaliações feitas á série;
- runtime – número de episódios;
- number_of_seasons – número de temporadas;
- network – serviço de streaming;
- mean_rate – avaliação média da série;
- poster_image – url do poster da série;
- seasons – todas as temporadas e respetivos episódios de cada.

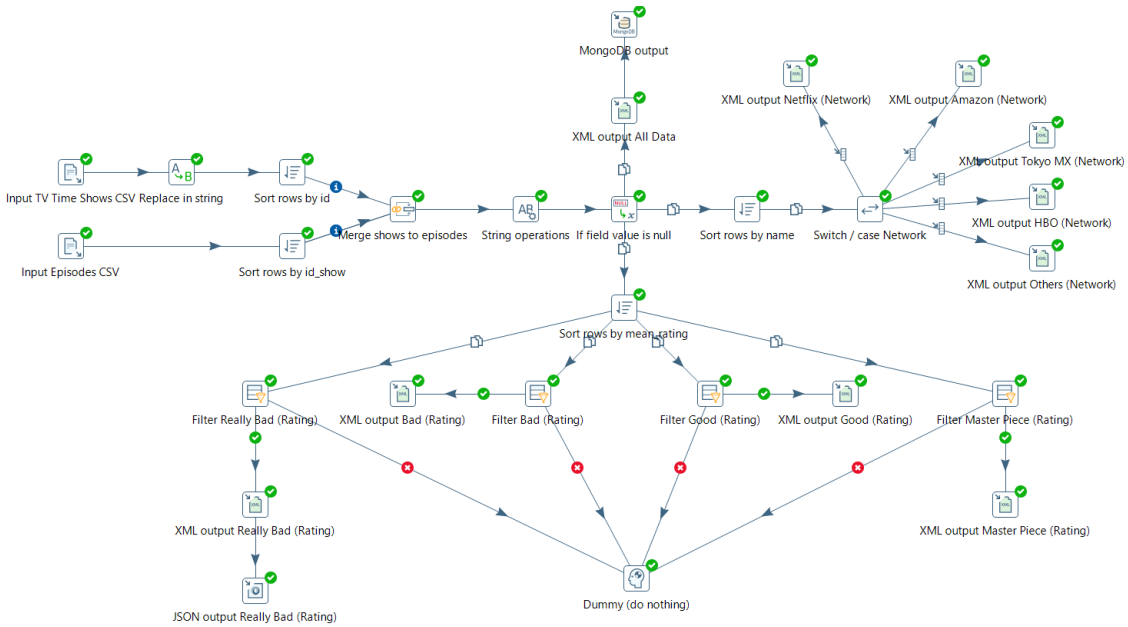
Solução

Solução PDI

Esta solução contém o desenvolvimento da Transformation e do Job.

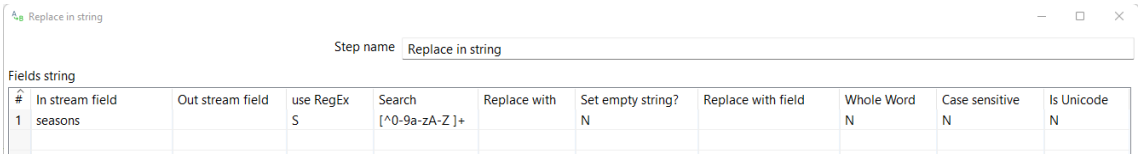
Transformation será usada para receber os dados dos datasets, transformá-los, filtrá-los, ordená-los e extrai-los.

Job é usado para coordenar a transformation além de validar ou reportar problemas.



3 – Transformation

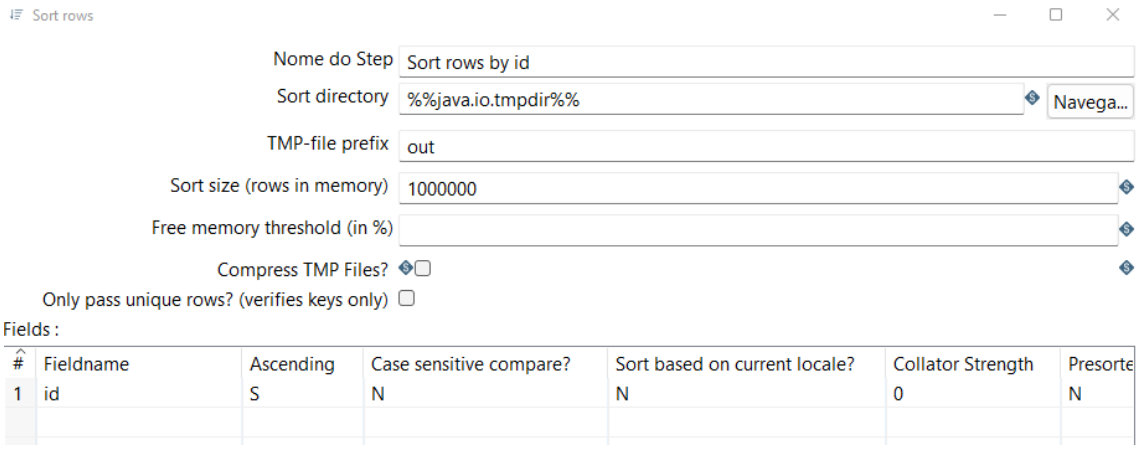
Como é possível se observar os dois data sets são inicialmente inseridos, onde o ficheiro que contém as séries sofre uma transformação recorrendo ao uso de um "Replace in string" e contém a seguinte procura de expressão regular:



4 - Expressão regular

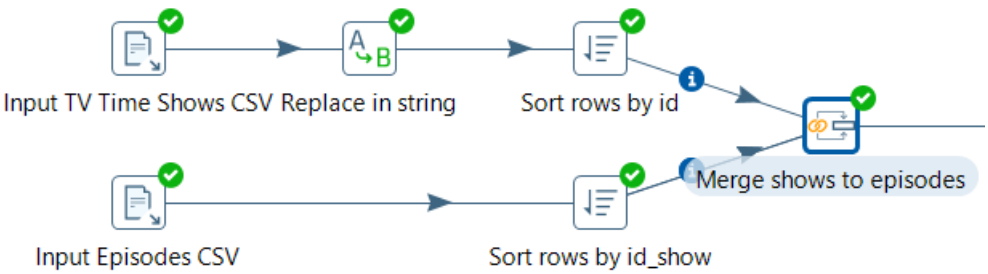
Esta expressão regular procura tudo que não sejam caracteres normais ou espaços e remove-os. É utilizado para remover os caracteres especiais do campo “seasons”.

Depois é feito uma ordenação dos dois datasets em função do id e id_show que representam o mesmo campo, mas em diferentes ficheiros.

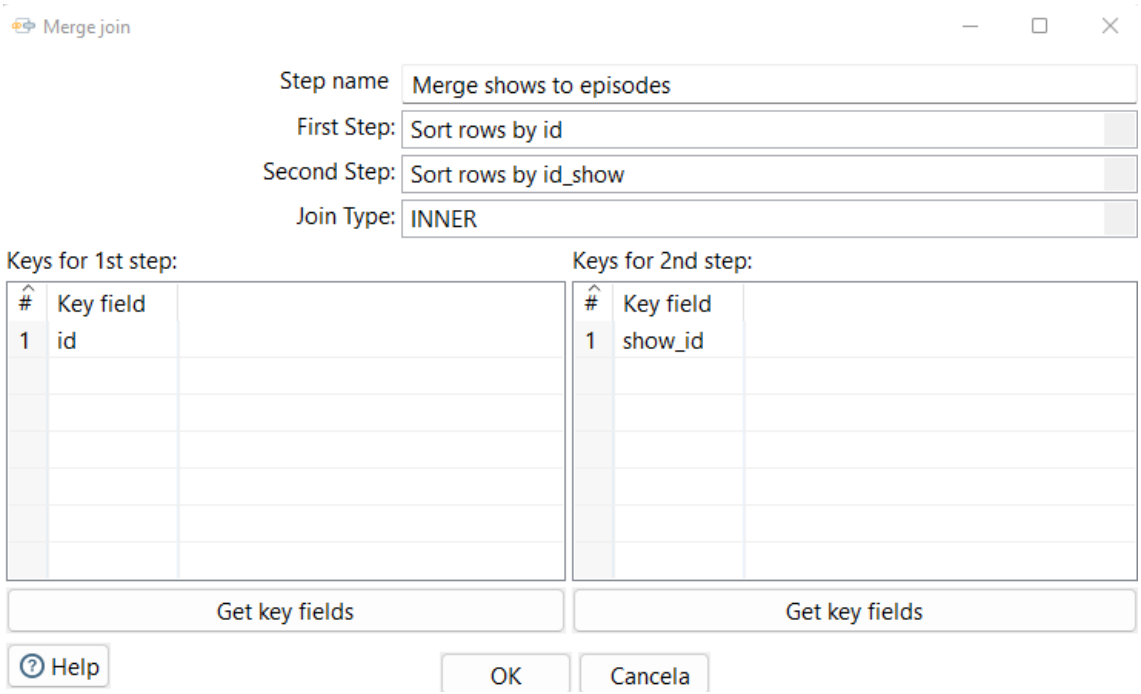


5 - Ordenar por id

Tem de ser ordenado desta forma para que se possa dar merge das duas tabelas. Este processo realiza-se conectando ambas á ferramenta de merge e identificando as chaves comuns às duas tabelas.

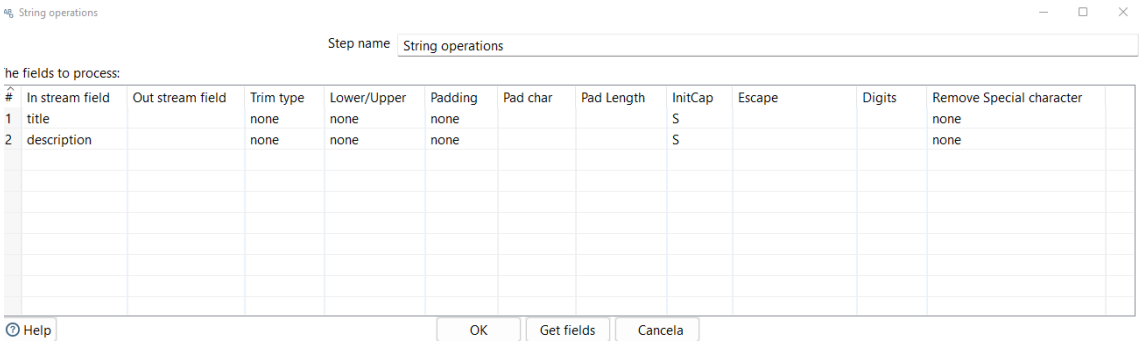


6 – Merge



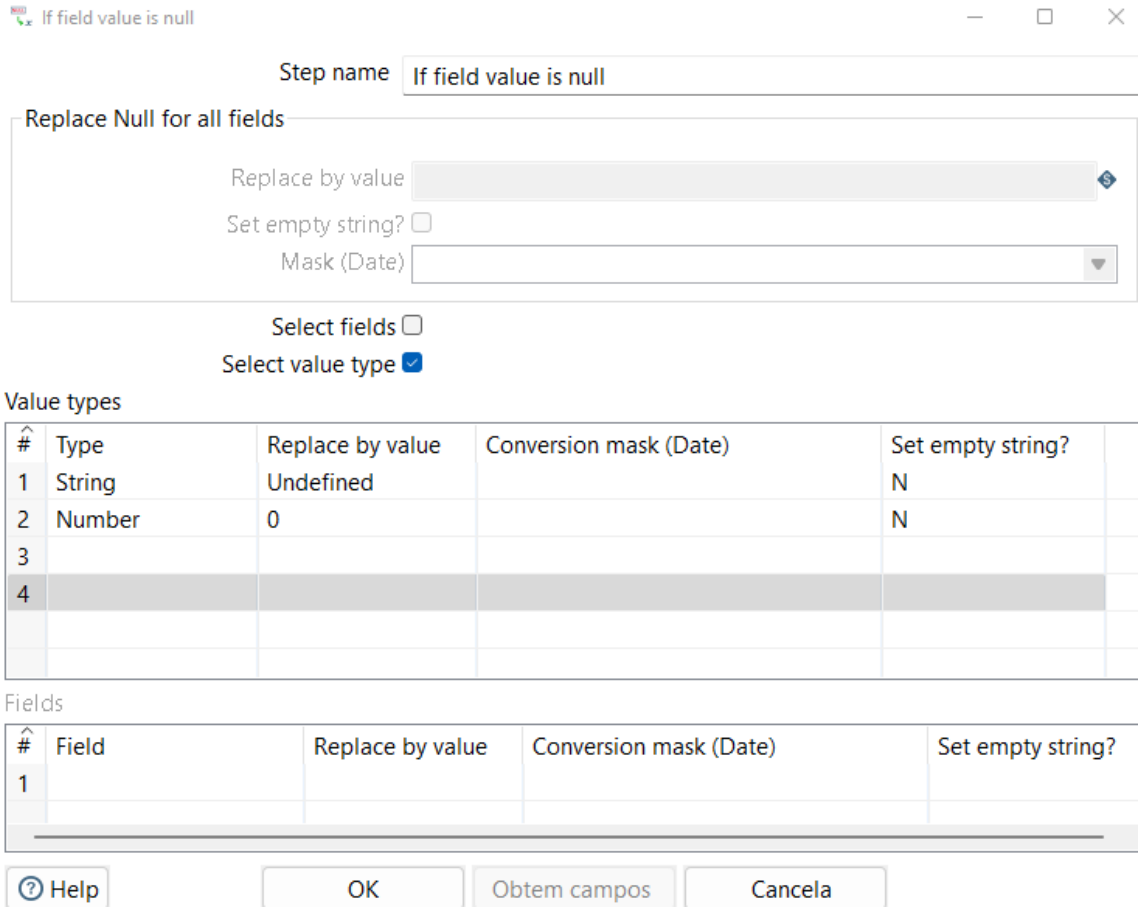
7 - Merge Opções

Depois é utilizado um “String operations” para remover caps locks desnecessários no meio de palavras nos campos “title” e “description”.



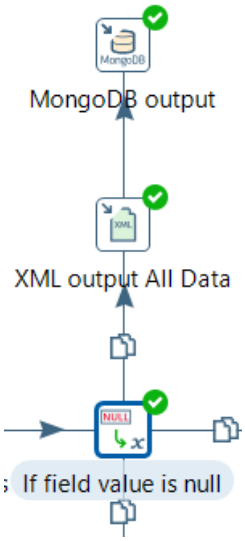
8 - Operações com strings

Ainda são verificados com com valor nulo e substituídos por 0 ou “undefined” dependendo do tipo de variável.

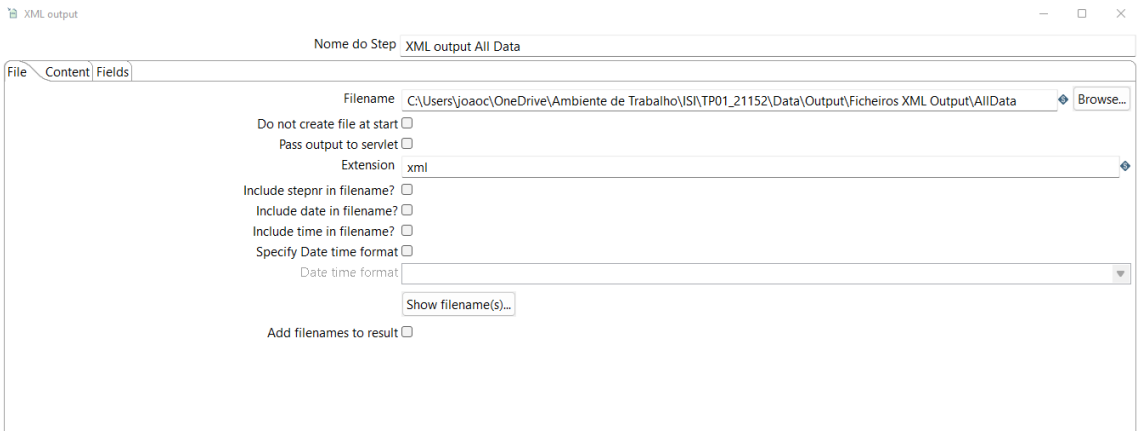


9 - Verificação de campos nulos

É realizada algumas separações a partir deste ponto. Um desses caminhos segue para uma extração em formato xml de todos os dados e depois para uma base de dados local do MongoDB.

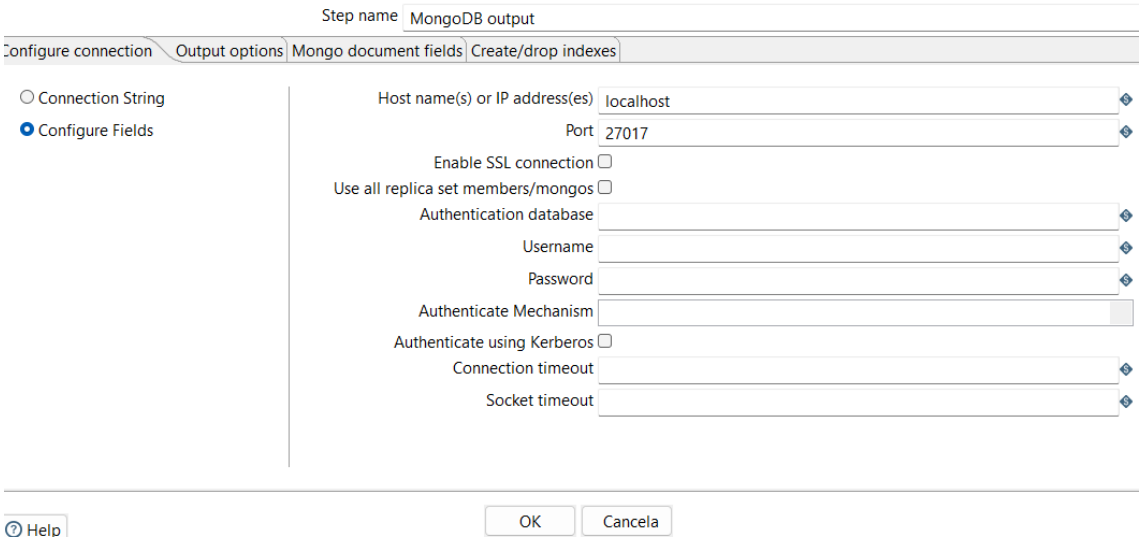


10 - Caminho de extração direta

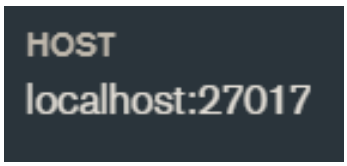


11 - Extração xml

A base de dados terá uma configuração com host name “localhost” e porta “27017” visíveis no MongoDB.

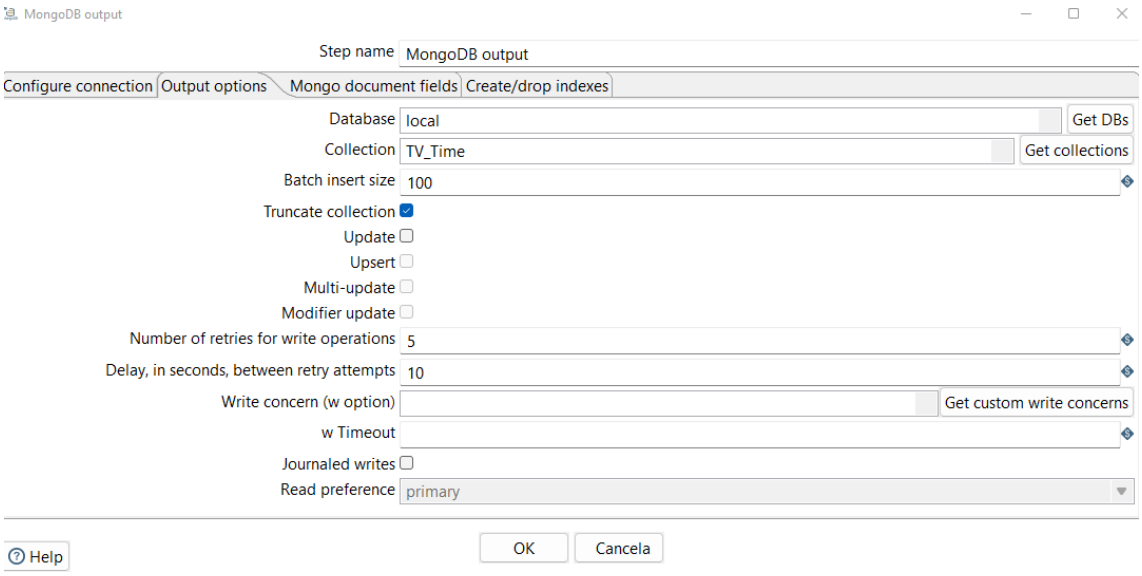


12 - DB - Configuração de conexão



13 - Mongo Host

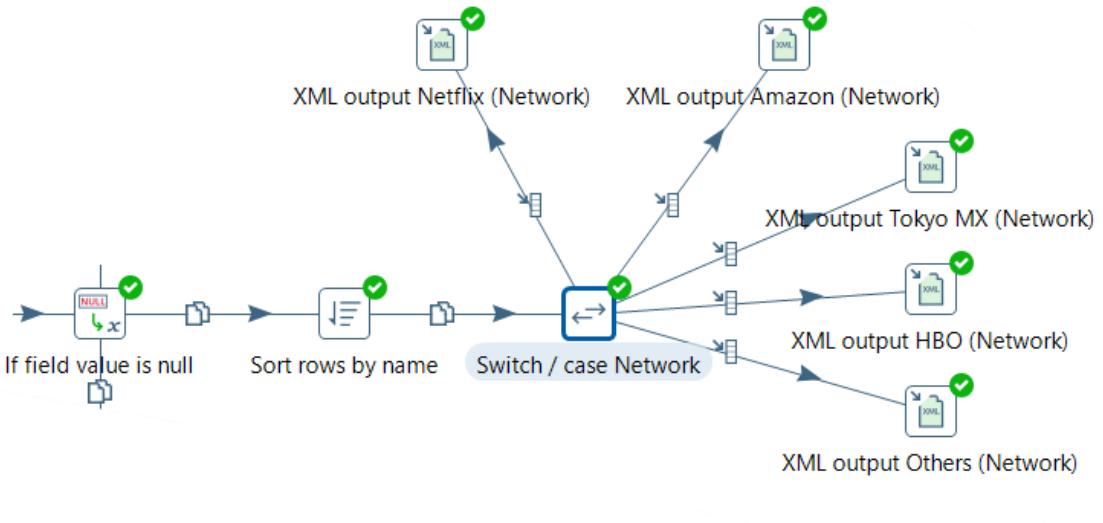
Nas opções de output selecionamos a base de dados que no caso está nomeada como “local” e respetiva coleção “TV_Time” criada. Além disso é ativada a opção de truncar coleção.



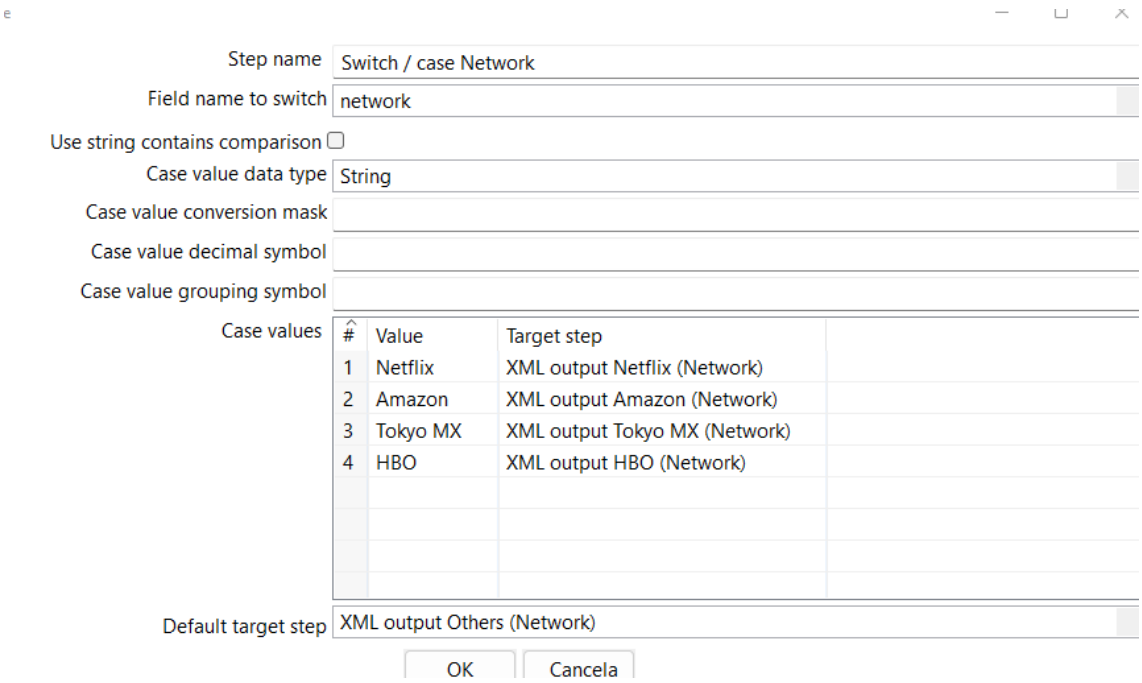
14 - DB - Opções de output

Além disto ainda em “Mongo document fields” busca-se todos os campos para ser enviado para a base de dados.

Seguindo o 2º caminho é ordenada a lista por nome e acontece uma estrutura de decisão “Switch / Case” para exportar em formato xml separadamente as Networks “Netflix”, “Amazon”, “Tokyo MX”, “HBO” e default todas as restantes.

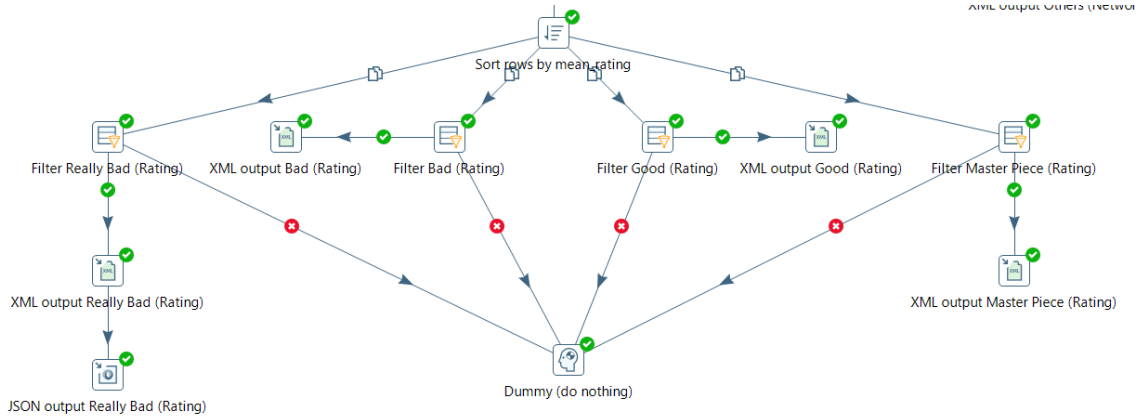


15 - 2 caminho



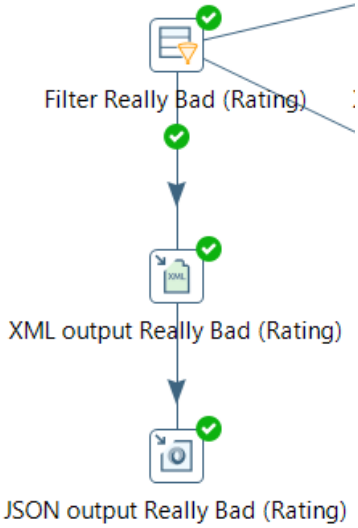
16 - Switch / Case

O terceiro caminho segue para uma ordenação por avaliação média das séries e por sua vez alguns filtros para avaliação média.

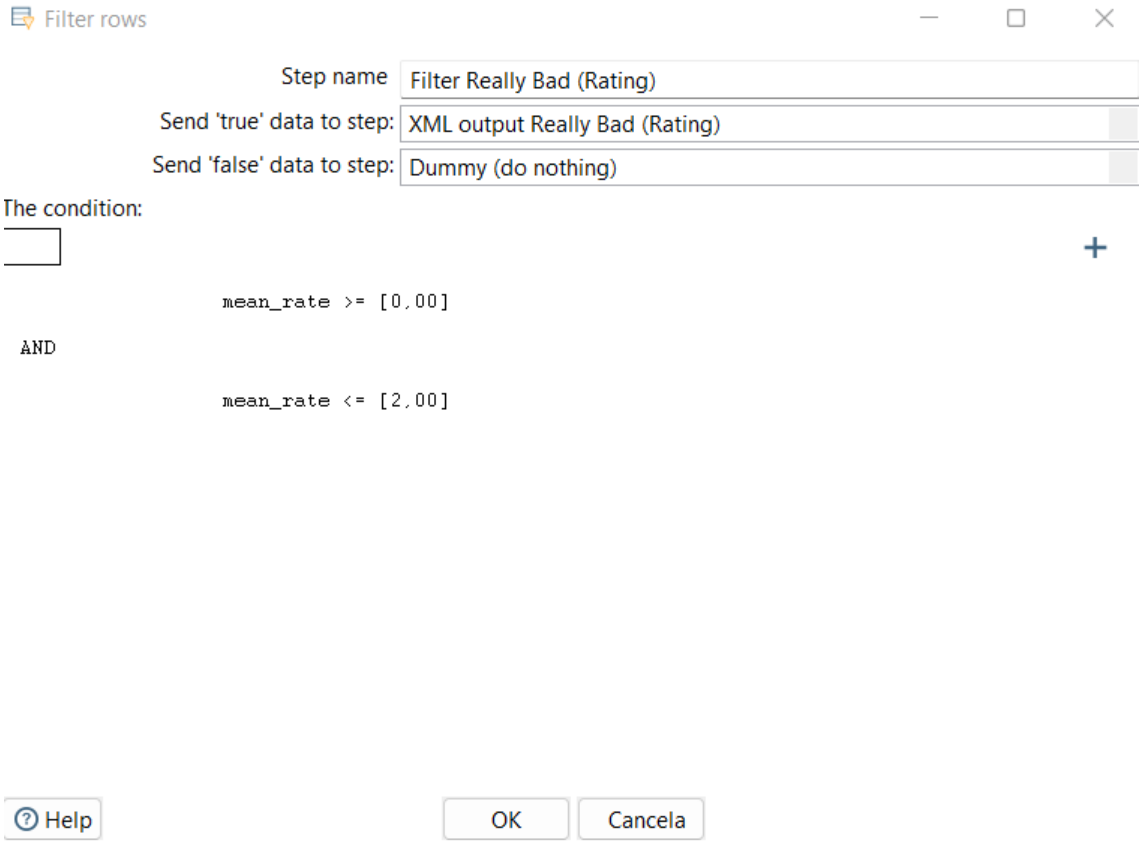


17 - 3 caminho

Num desses filtros temos por exemplo avaliações muito más.

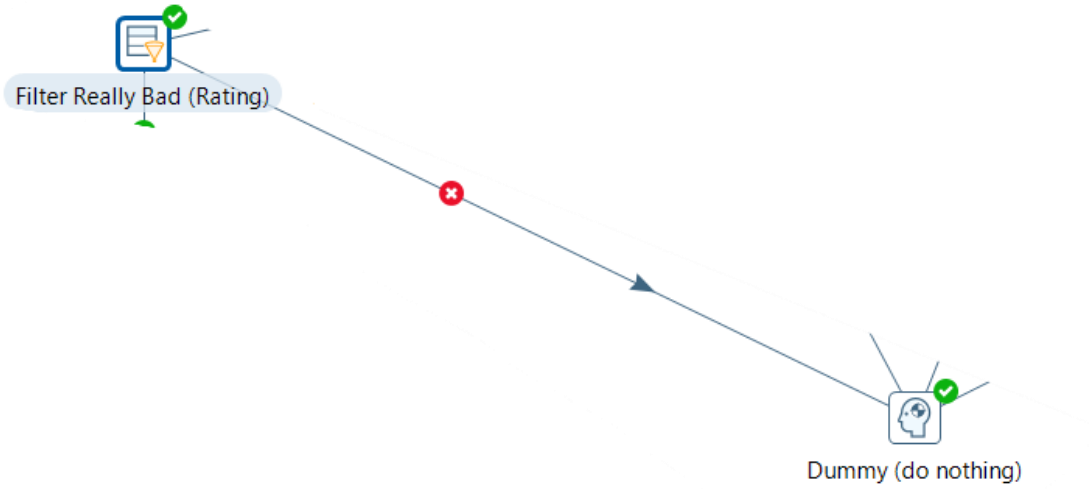


Em que se corresponder ao filtro exigido este será enviado para um xml a ser exportado e por sua vez um ficheiro tipo JSON.

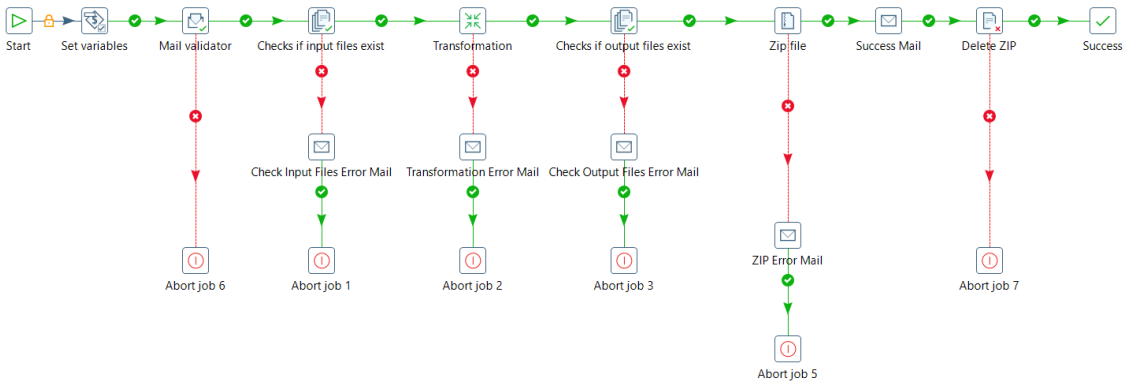


18 - Filtro Really bad

Caso não seja obedecida a condição do filtro este será enviado para um Dummy pois não pretendo fazer nada com estes dados.

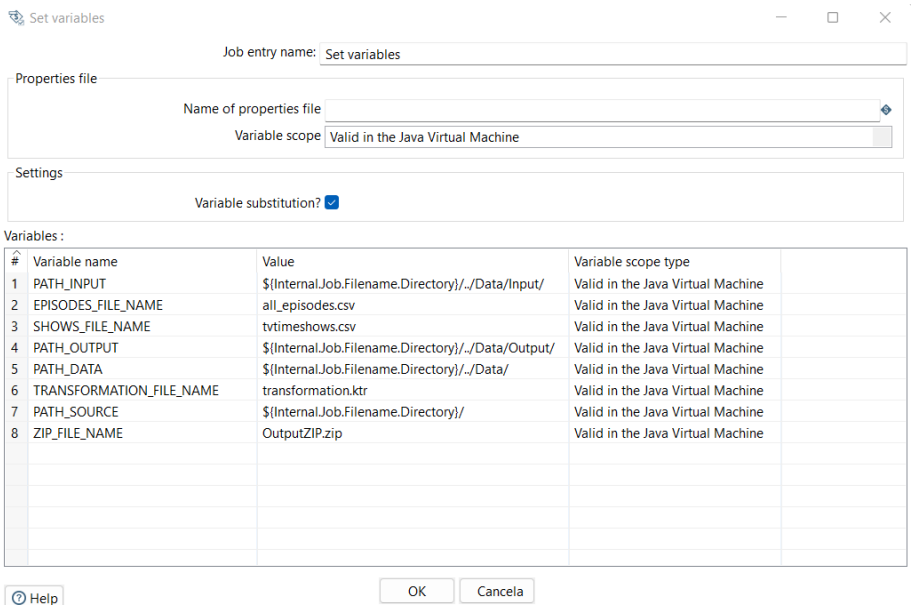


19 – Dummy



20 – Job

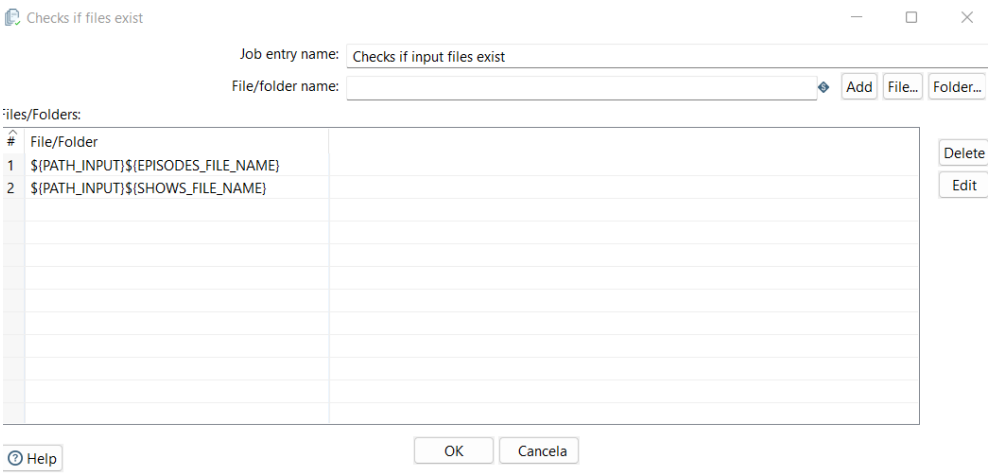
No job é criado variáveis no set variables a serem utilizados na inserção de caminhos ou para nomes de ficheiros.



21 - Set variables

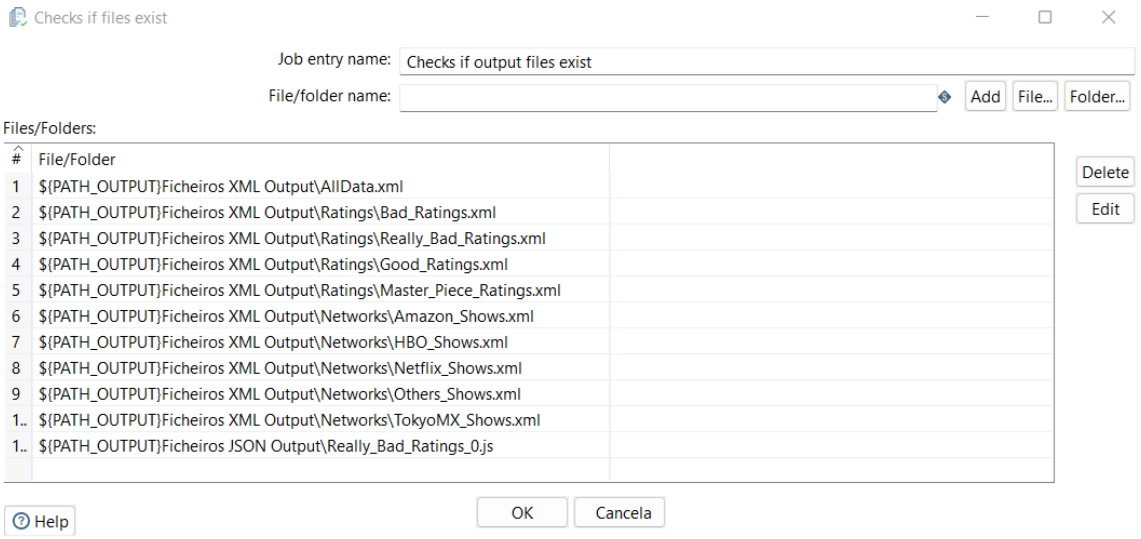
É validado o mail para qual será reportado todos os erros ou sucessos.

Verifica se os os ficheiros input da transformation existem.



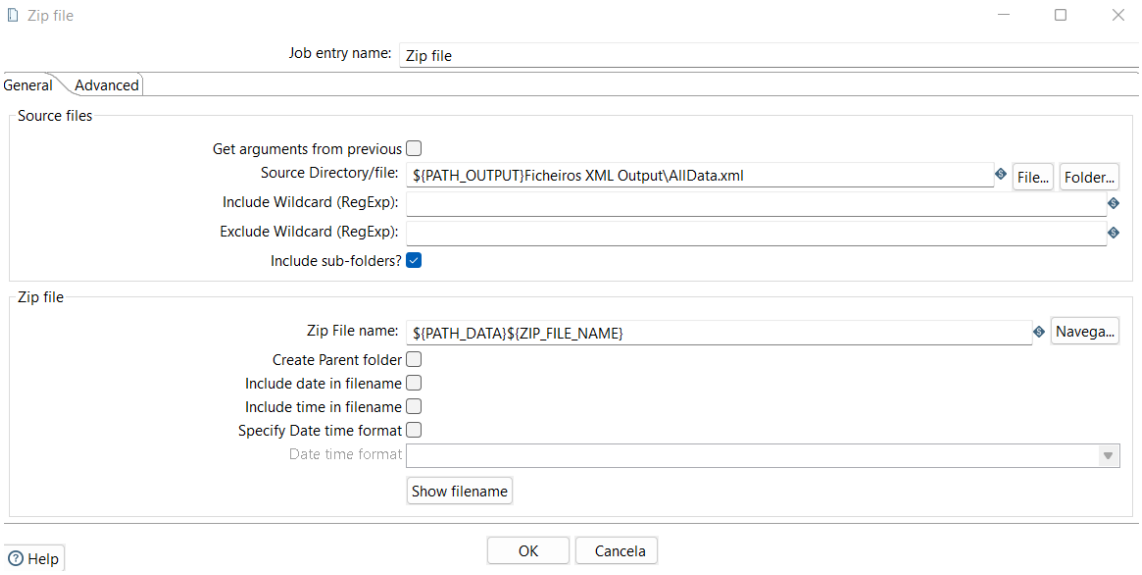
22 - Verificação de ficheiros input

Ocorre a transformação anteriormente explicada e depois é verificado se os ficheiros output foram criados com sucesso.



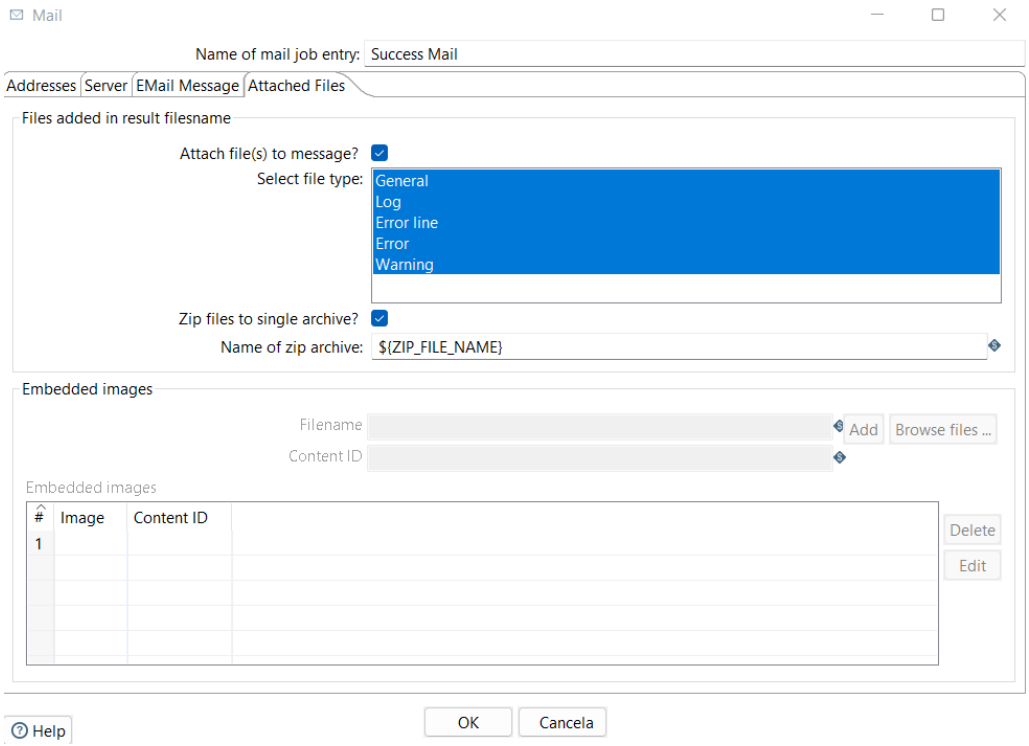
23 - Verificação de ficheiros input

É zipado um ficheiro com todos os dados que será enviado por mail caso de sucesso e depois removido.



24 - Zip file

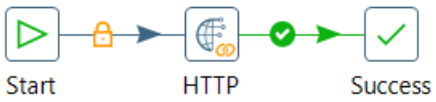
No mail enviado como sucesso é selecionado todo o tipo de ficheiros e indicado o nome do ficheiro zip.



25 - Mail

Depois de removido o zip, o job termina e dá por terminado o programa.

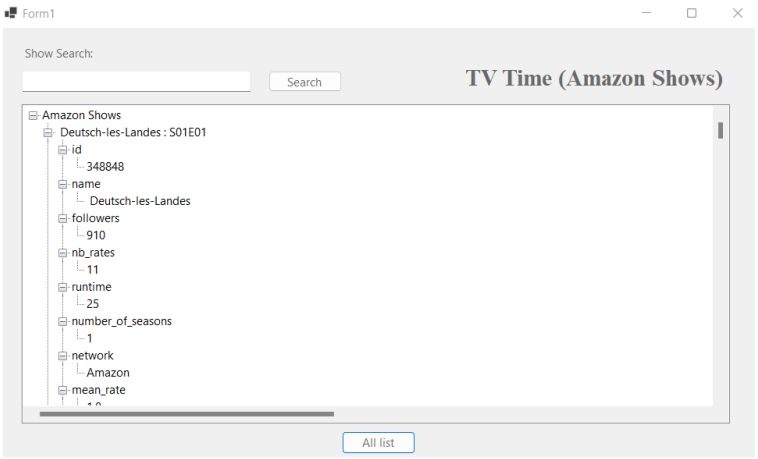
Nota: É de mencionar que gostaria de ter criado outro job para utilizador o HTTP para realizar download dos ficheiros exportados do site de origem dos datasets, mas não foi possível devido ao site apresentar uma política de autenticação para dowload dos ficheiros. Após isso seria inserido um job dentro do job que eu criei para enviar os dois datasets para o transformation. Teria este formato:



Programa de navegação xml

Após o desenvolvimento ao nível de tratamento dos dados é corrido um programa realizado em Visual Studio 22, linguagem C# Windows forms, que lê o xml e dispões este numa árvore. Este ainda dispões de uma pesquisa por título de série.

26 - Interface Pesquisa XML



27 - Interface Pesquisa XML - All List

28 - Interface Pesquisa XML - Pesquisa por título

Para a realização deste código foi feito no Form1.cs

No botão “All list” onde é apresentado toda a lista carregada do ficheiro é realizada uma limpeza inicial dos nodos da tree view que é o display que apresenta os dados em formato árvore.

É carregado o ficheiro Amazon_Shows.xml e enviado para a função que irá carregar os dados para a tree view. E por fim é expandida toda a tree view para visualização dos dados.

```
private void button2_Click_1(object sender, EventArgs e)
{
    treeView1.Nodes.Clear();
    XmlDoc.Load(Environment.CurrentDirectory + @"\..\..\..\..\..\..\..\Data\Output\Ficheiros XML Output\Networks\Amazon_Shows.xml");
    load_tree(XmlDoc, treeView1);
    treeView1.ExpandAll();
}
```

29 - Visual Studio - Butão

A função de Load da arvore recebe o documento xml importado e a tree view que vai ser apresentada. É criado o nodo mais alto da hierarquia da tree view com o nome de “Amazon Show”. É enviado o nodo e os elementos do xml para uma outra função.

```
private void load_tree(XmlDocument xml_doc, TreeView tree_view)
{
    TreeNode root = new TreeNode("Amazon Shows");
    tree_view.Nodes.Clear();
    add_tree_view_model(root,xml_doc.DocumentElement);
}
```

30 - Visual Studio - Carregar árvore

Nesta função são adicionados todos os nodos á arvore. Percorre todos os nodos filhos e cria um nodo com hierarquia inferior á do root no qual é apresentado nome da serie e número de episódio.

Depois percorre dentro desse episódio todos os seus elementos e criado outra hierarquia, que no caso a mais inferior e nesta é adicionado o texto de que cada elemento é composto.

Por fim tudo adicionado á tree view.

```
private void add_tree_view_model(TreeNode parent_nodes, XmlNode xml_node)
{
    foreach (XmlNode rows in xml_node.ChildNodes)
    {
        TreeNode new_node = parent_nodes.Nodes.Add(rows.ChildNodes[1].InnerText + ": " + rows.ChildNodes[13].InnerText);
        foreach (XmlNode row in rows)
        {
            TreeNode new_node2 = new_node.Nodes.Add(row.Name);
            new_node2.Nodes.Add(row.InnerText);
        }
    }
    treeView1.Nodes.Add(parent_nodes);
}
```

31 - Visual Studio - Adicionar nodo

Da mesma forma é feito os passos de cima para o botão de pesquisa, mas com uma condição de pesquisa ser igual ao escrito na texto box.

```
1 reference
private void add_search_tree_view_model(TreeNode parent_nodes, XmlNode xml_node)
{
    foreach (XmlNode rows in xml_node.ChildNodes)
    {
        if (rows.ChildNodes[1].InnerText.ToLower().Contains(search_box.Text.ToLower()))
        {
            TreeNode new_node = parent_nodes.Nodes.Add(rows.ChildNodes[1].InnerText + ": " + rows.ChildNodes[13].InnerText);
            foreach (XmlNode row in rows)
            {
                TreeNode new_node2 = new_node.Nodes.Add(row.Name);
                new_node2.Nodes.Add(row.InnerText);
            }
        }
    }
    treeView1.Nodes.Add(parent_nodes);
}
```

32 - Visual Studio - Adicionar nodo com condição

Conclusão

Com o desenvolvimento deste projeto foi possível adquirir as competências base para entender e realizar processos de transformação de dados e manipulação dos mesmos para apresentação e pesquisa.