



# **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

Resumo .....	2
Introdução.....	5
Resumo .....	5
Objetivos .....	6
Ferramentas .....	6
Implementação .....	7
Estrutura dos Datasets .....	7
Todos os episódios .....	7
Todas as séries .....	8
Solução .....	8
Solução PDI.....	8
Programa de navegação xml .....	19
Conclusão.....	22

1 - TV Time App.....	5
2 - TV Time voto de humor .....	7
3 – Transformation .....	8
4 - Expressão regular.....	9
5 - Ordenar por id.....	9
6 – Merge .....	9
7 - Merge Opções.....	10
8 - Operações com strings .....	10
9 - Verificação de campos nulos .....	11
10 - Caminho de extração direta .....	11
11 - Extração xml.....	12
12 - DB - Configuração de conexão.....	12
13 - Mongo Host .....	12
14 - DB - Opções de output.....	13
15 - 2 caminho.....	13
16 - Switch / Case.....	14
17 - 3 caminho.....	14
18 - Filtro Really bad .....	15
19 – Dummy .....	16
20 – Job .....	16
21 - Set variables.....	17
22 - Verificação de ficheiros input.....	17
23 - Verificação de ficheiros input.....	18
24 - Zip file.....	18
25 - Mail .....	19
26 - Interface Pesquisa XML .....	20
27 - Interface Pesquisa XML - All List.....	20
28 - Interface Pesquisa XML - Pesquisa por título .....	20
29 - Visual Studio - Botão.....	21
30 - Visual Studio - Carregar árvore.....	21
31 - Visual Studio - Adicionar nodo .....	21
32 - Visual Studio - Adicionar nodo com condição.....	22

# Introdução

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

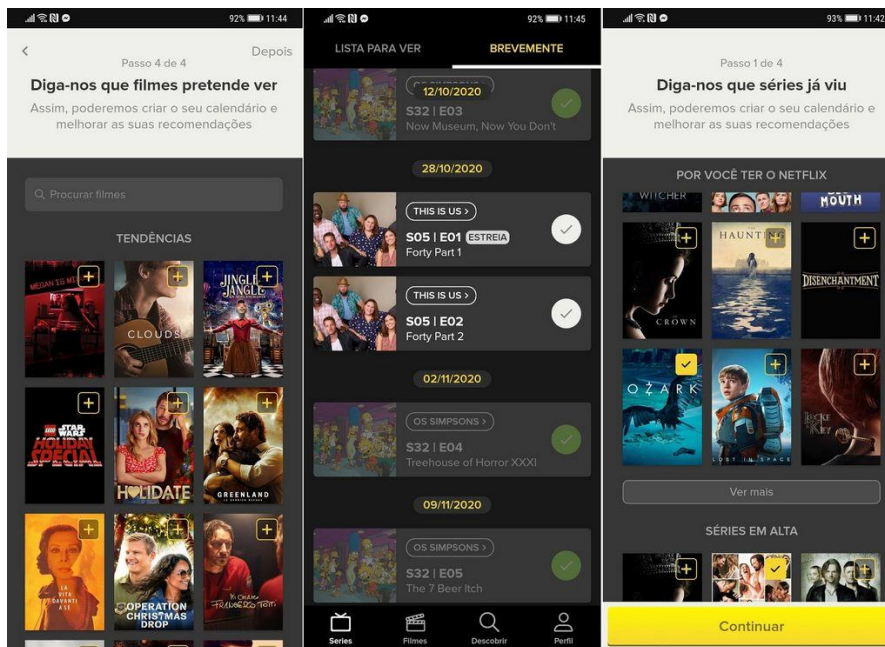
## Resumo



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/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 vírgulas 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.

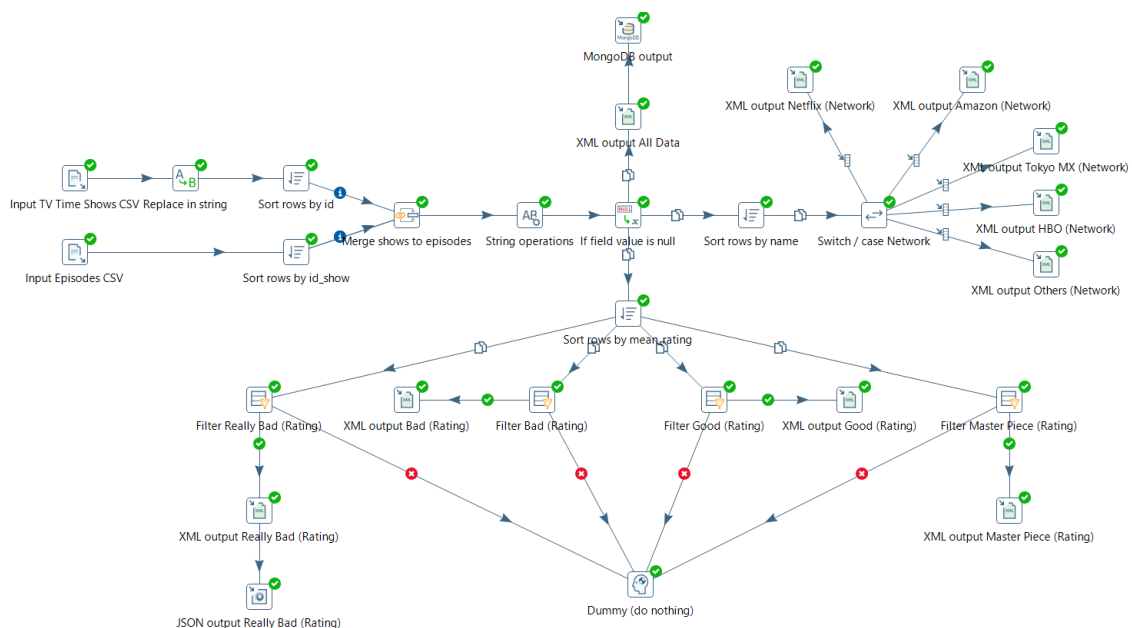
## Solução

### 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 transformação 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:



Replace in string

Step name: Replace in string

#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?	Replace with field	Whole Word	Case sensitive	Is Unicode
1	seasons		S	[^0-9a-zA-Z ]+		N		N	N	N

#### 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.

Sort rows

Nome do Step: Sort rows by id

Sort directory: %%java.io.tmpdir%% [Navega...](#)

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☐

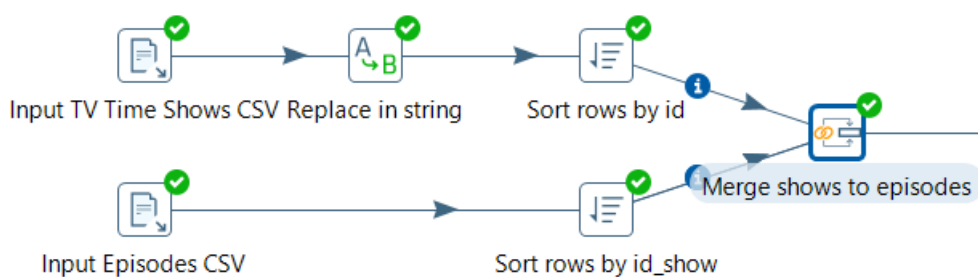
Only pass unique rows? (verifies keys only) ☐

Fields:

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presort
1	id	S	N	N	0	N

#### 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

Merge join

Step name: Merge shows to episodes

First Step: Sort rows by id

Second Step: Sort rows by id\_show

Join Type: INNER

Keys for 1st step:

#	Key field
1	id

Get key fields

Keys for 2nd step:

#	Key field
1	show_id

Get key fields

Help OK Cancela

## 7 - Merge Opções

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

String operations

Step name: String operations

he fields to process:

#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap	Escape	Digits	Remove Special character
1	title		none	none	none			S			none
2	description		none	none	none			S			none

Help OK Get fields Cancela

## 8 - Operações com strings

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

If field value is null

Step name If field value is null

Replace Null for all fields

Replace by value

Set empty string? ☐

Mask (Date)

Select fields ☐

Select value type ☒

Value types

#	Type	Replace by value	Conversion mask (Date)	Set empty string?
1	String	Undefined		N
2	Number	0		N
3				
4				

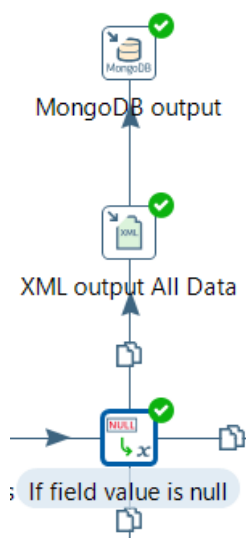
Fields

#	Field	Replace by value	Conversion mask (Date)	Set empty string?
1				

Help OK Obtem campos Cancela

#### 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

Nome do Step: XML output All Data

File | Content | Fields

Filename: C:\Users\joao\OneDrive\Ambiente de Trabalho\JS\TP01\_21152\Data\Output\Ficheiros XML Output\AllData [Browse...]

Do not create file at start: ☐

Pass output to servlet: ☐

Extension: xml

Include stepnr in filename?: ☐

Include date in filename?: ☐

Include time in filename?: ☐

Specify Date time format: ☐

Date time format: [dropdown]

Show filename(s)...

Add filenames to result: ☐

#### 11 - Extração xml

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

Step name: MongoDB output

Configure connection | Output options | Mongo document fields | Create/drop indexes

Connection String: ☐

Configure Fields: ☒

Host name(s) or IP address(es): localhost

Port: 27017

Enable SSL connection: ☐

Use all replica set members/mongos: ☐

Authentication database: [empty]

Username: [empty]

Password: [empty]

Authenticate Mechanism: [empty]

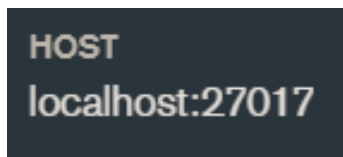
Authenticate using Kerberos: ☐

Connection timeout: [empty]

Socket timeout: [empty]

Help | OK | Cancela

#### 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**.

MongoDB output

Step name: MongoDB output

Configure connection | Output options | **Mongo document fields** | Create/drop indexes

Database: local Get DBs

Collection: TV\_Time Get collections

Batch insert size: 100

Truncate collection: ☒

Update: ☐

Upsert: ☐

Multi-update: ☐

Modifier update: ☐

Number of retries for write operations: 5

Delay, in seconds, between retry attempts: 10

Write concern (w option): Get custom write concerns

w Timeout:

Journalled writes: ☐

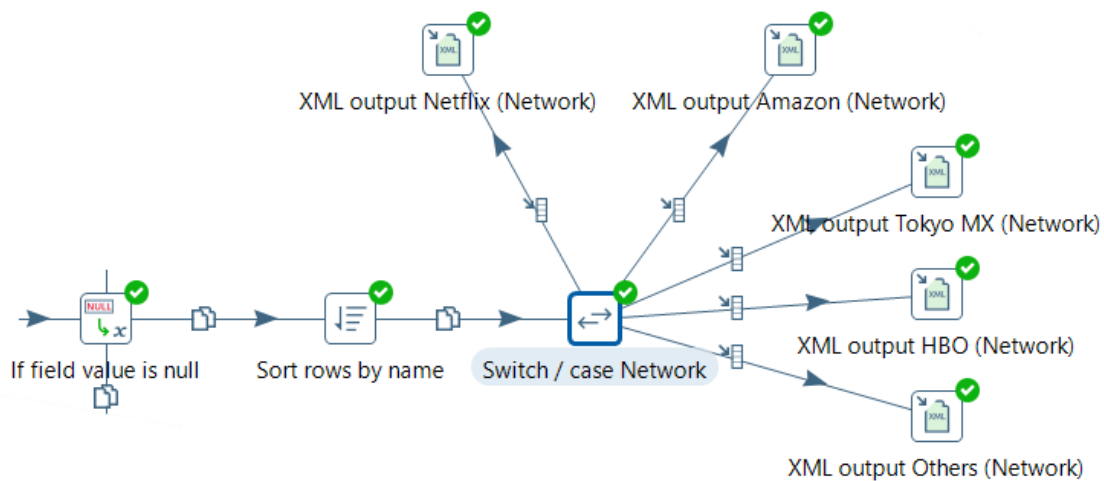
Read preference: primary

Help OK Cancela

#### 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

Step name Switch / case Network

Field name to switch network

Use string contains comparison ☐

Case value data type String

Case value conversion mask

Case value decimal symbol

Case value grouping symbol

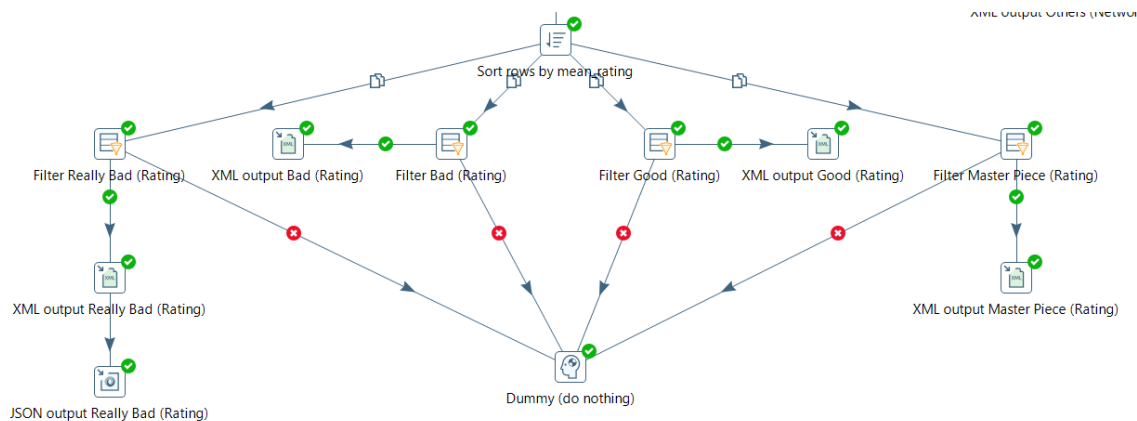
Case values	#	Value	Target step
	1	Netflix	XML output Netflix (Network)
	2	Amazon	XML output Amazon (Network)
	3	Tokyo MX	XML output Tokyo MX (Network)
	4	HBO	XML output HBO (Network)

Default target step XML output Others (Network)

OK Cancela

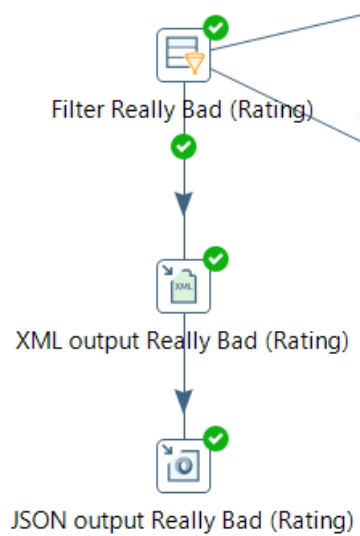
#### 16 - Switch / Case

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.

Filter rows
—
□
×

Step name	Filter Really Bad (Rating)
Send 'true' data to step:	XML output Really Bad (Rating)
Send 'false' data to step:	Dummy (do nothing)

The condition:

+

`mean_rate >= [0,00]`

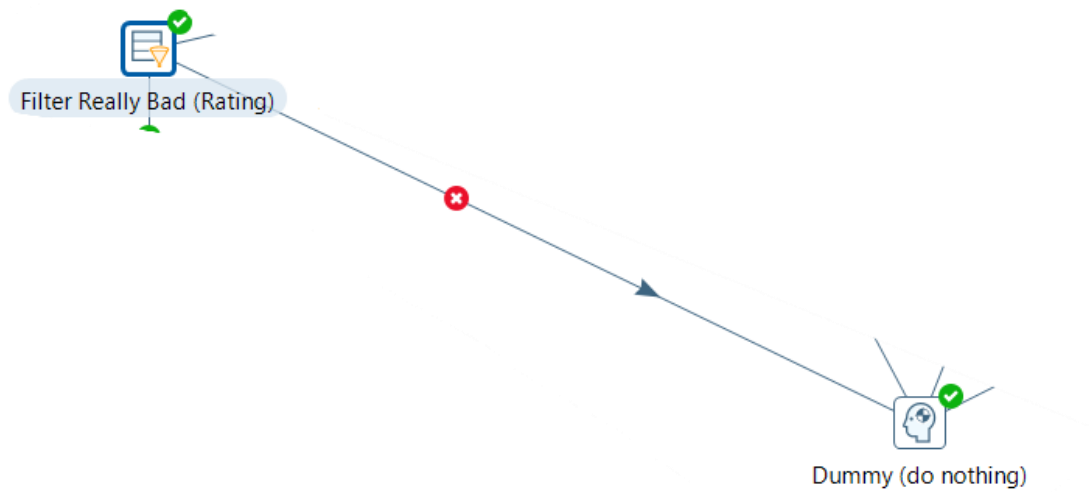
AND

`mean_rate <= [2,00]`

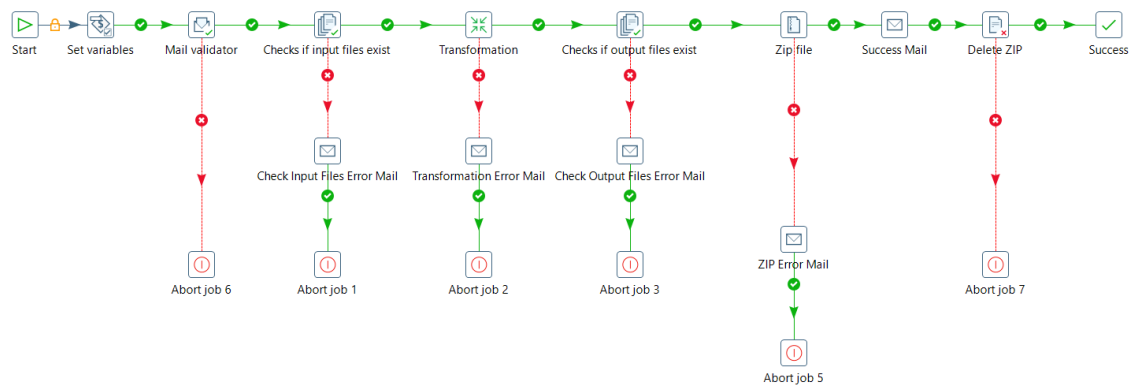
Help
OK
Cancela

#### 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.



Set variables

Job entry name: Set variables

Properties file

Name of properties file

Variable scope: Valid in the Java Virtual Machine

Settings

Variable substitution? ☒

Variables :

#	Variable name	Value	Variable scope type
1	PATH_INPUT	\${Internal.Job.FileName.Directory}/../Data/Input/	Valid in the Java Virtual Machine
2	EPISODES_FILE_NAME	all_episodes.csv	Valid in the Java Virtual Machine
3	SHOWS_FILE_NAME	tvtimeshows.csv	Valid in the Java Virtual Machine
4	PATH_OUTPUT	\${Internal.Job.FileName.Directory}/../Data/Output/	Valid in the Java Virtual Machine
5	PATH_DATA	\${Internal.Job.FileName.Directory}/../Data/	Valid in the Java Virtual Machine
6	TRANSFORMATION_FILE_NAME	transformation.ktr	Valid in the Java Virtual Machine
7	PATH_SOURCE	\${Internal.Job.FileName.Directory}/	Valid in the Java Virtual Machine
8	ZIP_FILE_NAME	OutputZIP.zip	Valid in the Java Virtual Machine

Help OK Cancela

## 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.

Checks if files exist

Job entry name: Checks if input files exist

File/folder name:

Add File... Folder...

Files/Folders:

#	File/Folder
1	\${PATH_INPUT}\${EPISODES_FILE_NAME}
2	\${PATH_INPUT}\${SHOWS_FILE_NAME}

Delete Edit

Help OK Cancela

## 22 - Verificação de ficheiros input

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

Checks if files exist

Job entry name: Checks if output files exist

File/folder name:  Add File... Folder...

Files/Folders:

#	File/Folder	
1	\${PATH_OUTPUT}Ficheiros XML Output\AllData.xml	
2	\${PATH_OUTPUT}Ficheiros XML Output\Ratings\Bad_Ratings.xml	
3	\${PATH_OUTPUT}Ficheiros XML Output\Ratings\Really_Bad_Ratings.xml	
4	\${PATH_OUTPUT}Ficheiros XML Output\Ratings\Good_Ratings.xml	
5	\${PATH_OUTPUT}Ficheiros XML Output\Ratings\Master_Piece_Ratings.xml	
6	\${PATH_OUTPUT}Ficheiros XML Output\Networks\Amazon_Shows.xml	
7	\${PATH_OUTPUT}Ficheiros XML Output\Networks\HBO_Shows.xml	
8	\${PATH_OUTPUT}Ficheiros XML Output\Networks\Netflix_Shows.xml	
9	\${PATH_OUTPUT}Ficheiros XML Output\Networks\Others_Shows.xml	
1..	\${PATH_OUTPUT}Ficheiros XML Output\Networks\TokyoMX_Shows.xml	
1..	\${PATH_OUTPUT}Ficheiros JSON Output\Really_Bad_Ratings_0.js	

Help OK Cancela

### 23 - Verificação de ficheiros input

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

Zip file

Job entry name: Zip file

General Advanced

Source files

Get arguments from previous ☐

Source Directory/file: \${PATH\_OUTPUT}Ficheiros XML Output\AllData.xml File... Folder...

Include Wildcard (RegExp):

Exclude Wildcard (RegExp):

Include sub-folders? ☒

Zip file

Zip File name: \${PATH\_DATA}\${ZIP\_FILE\_NAME} Navega...

Create Parent folder ☐

Include date in filename ☐

Include time in filename ☐

Specify Date time format ☐

Date time format:

Show filename

Help OK Cancela

### 24 - Zip file

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

Mail

Name of mail job entry: Success Mail

Addresses | Server | EMail Message | Attached Files

Files added in result filename

Attach file(s) to message? ☒

Select file type: General  
Log  
Error line  
Error  
Warning

Zip files to single archive? ☒

Name of zip archive: \${ZIP\_FILE\_NAME}

Embedded images

Filename  Add Browse files ...

Content ID

Embedded images

#	Image	Content ID
1		

Delete

Edit

Help OK Cancela

## 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.

Form1

Show Search:



**TV Time (Amazon Shows)**

## 26 - Interface Pesquisa XML

Form1

Show Search:



**TV Time (Amazon Shows)**

Amazon Shows

- [-] Deutsch-les-Landes : S01E01
  - [-] id
    - 348848
  - [-] name
    - Deutsch-les-Landes
  - [-] followers
    - 910
  - [-] nb\_rates
    - 11
  - [-] runtime
    - 25
  - [-] number\_of\_seasons
    - 1
  - [-] network
    - Amazon
  - [-] mean\_rate
    - ...

## 27 - Interface Pesquisa XML - All List

Form1

Show Search:



**TV Time (Amazon Shows)**

Amazon Shows

- [-] The Boys: S01E01
- [-] The Boys: S01E02
- [-] The Boys: S01E03
- [-] id
  - 355567
- [-] name
  - The Boys
- [-] followers
  - 271129
- [-] nb\_rates
  - 218
- [-] runtime
  - 60
- [-] number\_of\_seasons
  - 1
- [-] network
  - ...

## 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();
    XmlDocument.Load(Environment.CurrentDirectory + @"..\..\..\..\Data\Output\Ficheiros XML Output\Networks\Amazon_Shows.xml");
    load_tree(XmlDoc, treeView1);
    treeView1.ExpandAll();
}
```

29 - Visual Studio - Botão

A função de Load da árvore 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 árvore. 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.

```

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.