

Trabalho Prático I

Legendas de filmes em XML

Integração de
Sistemas de
Informação

Docente: Luis Gonzaga Martins Ferreira

Autor: Nuno Rodrigo Rebelo Sá Silva

LESI PL Nº 28005

Data: 18/10/2025

Contents

Introdução.....	4
Ferramentas utilizadas e como instalar.....	5
Knime Workflow.....	6
Caminhos Relativos	6
Leitura de ficheiros + Validação DTD	8
Validação DTD	8
Filtragem de XML+DTD	8
Transformação de dados	9
Transformação de dados: Números Inteiros.....	10
Transformação de dados: Strings.....	11
Transformação de dados: Listas.....	12
Separação de listas	12
Legendas.....	13
Hórrarios (Timestamps).....	13
Agregação de dados.....	14
Retorno de JSON para API Endpoint	16
Escrita de Logs	17
XML inválido de acordo com DTD	18
Legendas / Hórrarios (Timestamps) inválidos.....	19
Falta de informação.....	21
ASP.NET.....	23
Web APP-React.....	23
Estutura de ficheiros	24
Bibliografia.....	25
Video da Aplicação	25

Figura 1 - Caminhos Relativos - Input Location.....	6
Figura 2 - Caminhos Relativos - Flow Variables	7
Figura 3 - Caminhos Relativos - Variable to String	7
Figura 4 - Caminhos Relativos - Workflow	7
Figura 5 –XML Reader + DTD - Workflow.....	8
Figura 6 –Transformação de dados - Workflow.....	9
Figura 7 – Transformação de Inteiros	10
Figura 8 - Transformação de Strings (Regex)	11
Figura 9 – Listas - Workflow.....	12
Figura 10 - Tranformação de Timestamps.....	13
Figura 11 –Agregação de dados - Workflow.....	14
Figura 12 - Configuração de agregação.....	15
Figura 13 - Configuração POST Request.....	16
Figura 14 - Exemplo Log - Workflow	17
Figura 15 - DTD Log - Workflow.....	18
Figura 16 - Rule Engine para Legendas e Timestamps.....	19
Figura 17 - Log para legendas e timestamps - Workflow.....	20
Figura 18 - Rule Engine para falta de dados	21
Figura 19 - Log para dados em falta - Workflow.....	22
Figura 20 - Estrutura do Projeto	24

Introdução

Com este trabalho da Disciplina de Integração de Sistemas de Informação (ISI) pretende-se focar a aplicação e experimentação de ferramentas em processos de ETL (Extract, Transformation and Load), inerentes a processos de Integração de Sistemas de informação ao nível dos dados.

Após a leitura dos dados é nos atribuída os seguintes objetivos:

- 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;

Esta aplicação desenvolvida para este projeto chama-se SubVaulte e foi desenvolvida para ser uma aplicação web que gere ficheiros XML relacionados com legendas de filmes. A plataforma permite carregar e descarregar ficheiros XML, processar e tratar automaticamente os dados contidos neles, e apresentar de forma organizada as legendas armazenadas numa base de dados centralizada.

Ferramentas utilizadas e como instalar

Este projeto consiste em uma aplicação web feita em React, ASP.NET e Knime Workflows.

Para correr o projeto é necessário:

- React
 - Node.js instalado
 - Na pasta ./React/ no terminal (cmd) correr
 - npm install
 - npm start
 - O serviço deve começar a correr em localhost:3000
- ASP.NET
 - Visual Studio 2022
 - Para a aplicação funcionar os Endpoints deve estar configurados para utilizar a porta 5110
 - Caso o SQL Server não seja (localhost)\SQLServer, é preciso alterar no ficheiro appsettings.json o caminho da base de dados correto
- Knime
 - É necessário ter Oracle Java instalado
 - É necessário ter o Knime Analytics Platform
 - É necessário que o workflow esteja na mesma diretoria que a aplicação ASP.NET (já vai entregue desta forma)
- SQL Server
 - É necessário criar ou restaurar a base de dados
 - Conteúdo disponível em ./SQL/
 - Backup SQL Server versão 16.0.1000.6
 - Se SQL Server Management Studio for de versão inferior
 - Crie as tabelas com a query presente na pasta
 - É necessário que a base de dados premita Server Authentication
 - SQL Server and Windows Authentication mode
 - Sobre a base de dados
 - Database: ISI
 - User: admin
 - Pass: 1234
 - User role: db_owner

Knime Workflow

O workflow de Knime faz:

- Leitura de ficheiros XML
- Validação DTD (Java Parser)
- Extração de dados Xpath
- Tratamento de dados
- Criação de logs de erro
- Conversão para JSON
- Envio de JSON para API Endpoint (Request Body)

Caminhos Relativos

O workflow usa caminhos relativos de forma a ser mais fácil para o utilizador partilhar o workflow, para leitura e criação de ficheiro a configuração é simples:

Read from: Mountpoint | LOCAL

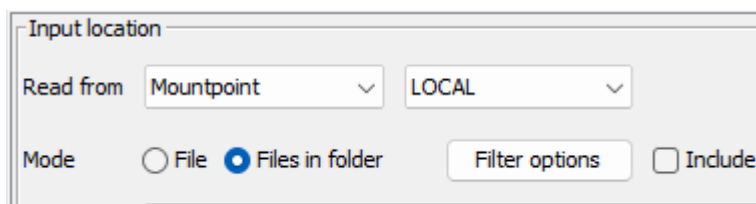



Figura 1 - Caminhos Relativos - Input Location

Mas nem para a validação do XML com DTD é diferente. Para conseguir usar o caminho do DTD em código é preciso um caminho local, não relativo.

Para obter esse caminho utilizamos a unica variável que é sempre criada em um workflow de knime, knime.workspace, este é o caminho fisico onde se encontra o workflow.

Para usar este caminho em código Java é necessário passar a variável para string e com algum tratamento de dados é possível para de:


► 1: Variable table  Flow Variables

Count: 1

Owner ID	Data Type	Variable Name	Value
	StringType	knime.workspace	C:\Users\epere\Documents\GitHub\ISI-WebApp\Knime

Figura 2 - Caminhos Relativos - Flow Variables

Para:

► 1: Output Table  Flow Variables

Rows: 1 | Columns: 1

#	RowID	LOCALPath <small>String</small>
1	Row0	C:\Users\epere\Documents\GitHub\ISI-WebApp\Knime\Upload\movies.dtd

Figura 3 - Caminhos Relativos - Variable to String

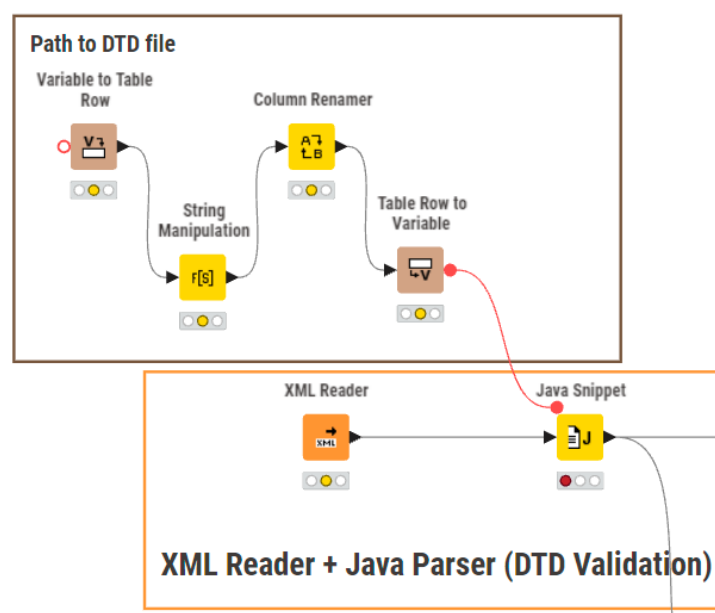


Figura 4 - Caminhos Relativos - Workflow

Leitura de ficheiros + Validação DTD

A leitura de ficheiros XML é uma leitura direta dos conteúdos dentro do caminho relativo ./Upload/. Este node está configurado para ler todos os ficheiros com extensão XML dentro da pasta.

Validação DTD

Como Knime não tem nenhuma ferramenta que faça a verificação DTD diretamente existem três opções:

1. Fazer a verificação externamente antes de começar o workflow
2. Utilizar um Parser em Python externamente
3. Utilizar um Parser em Java

Foi optado por se utilizar um Parser em Java pois Knime permite Java Code Snippets.

Esta validação utiliza o ficheiro movies.dtd, que se encontra dentro da pasta ./Upload, mencionado na secção “Caminhos Relativos”, para comparar com cada XML lido pelo XML Reader node.

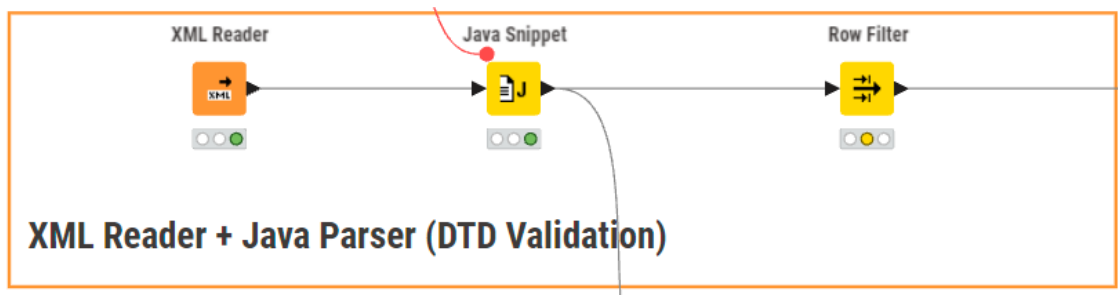


Figura 5 –XML Reader + DTD - Workflow

Filtração de XML+DTD

Este parser retorna uma das seguintes opções:

- Erro de validação DTD: The content of element type "movie" must match "(Id,title,year,language,movieLength,subtitles)".
- XML válido segundo o DTD.

Apenas dados com “XML válido segundo o DTD.” passam para a secção “Transformação de dados”

Transformação de dados

Neste workflow a transformação de dados pode ser repartida entre 3 partes:

1. Números inteiros
2. Strings
3. Listas
 - a. Legendas
 - b. Horário (Timestamp)

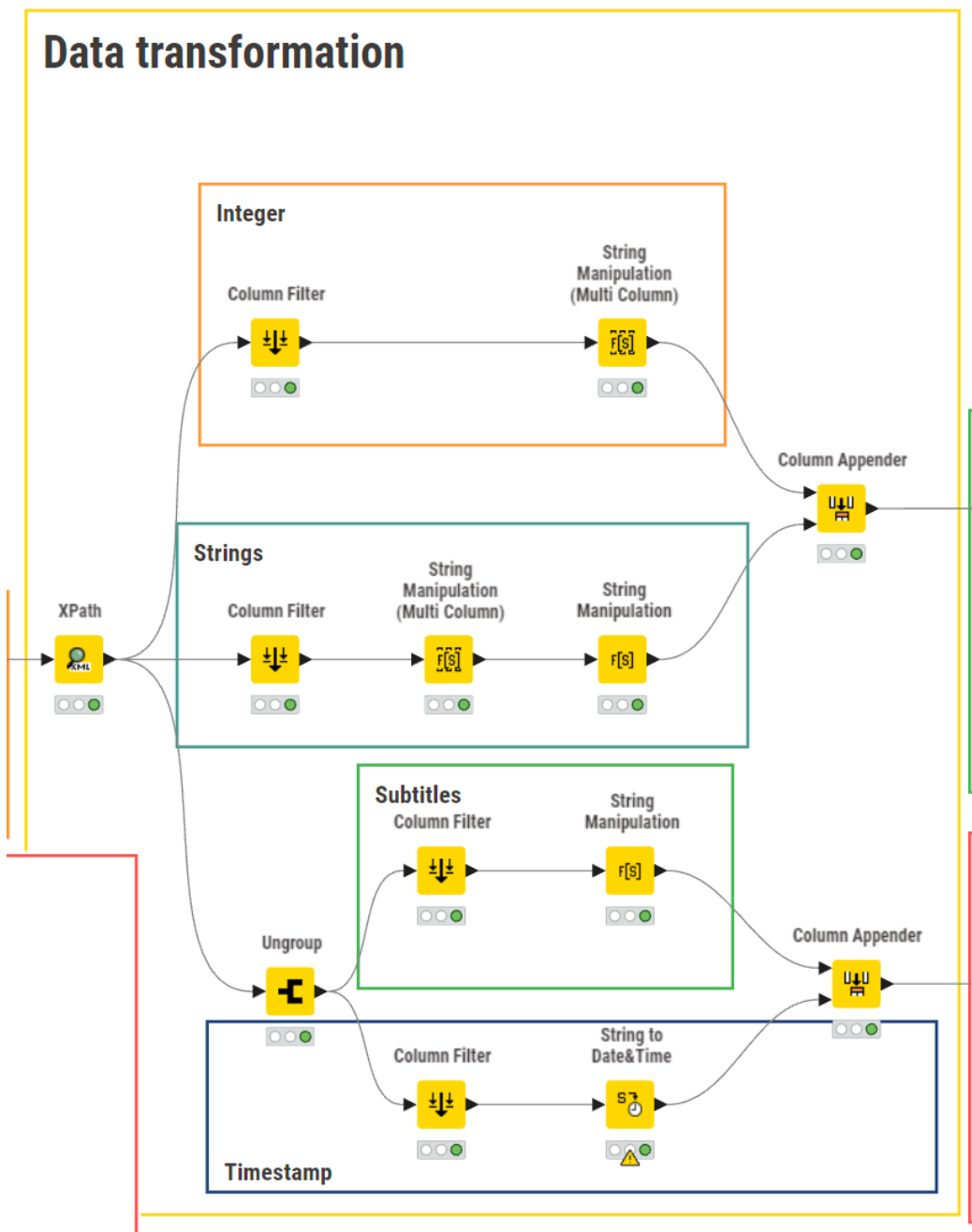


Figura 6 – Transformação de dados - Workflow

Transformação de dados: Números Inteiros

No processo de tratamento de dados de números começos por ler os mesmo com XPath. Os dados são todos lidos como string para evitar erros no XPath sem escrever log.

Os dados filtrados com “Column Filter” node são: MovieID e Year.

String Manipulation é muito simples neste caso, como a única informação que queremos retirar destes dados é saber se o valor é realmente um número, então só precisamos de converter de string para inteiro.

Expressão:

“toInt(\$\$CURRENTCOLUMN\$\$)”

Após a conversão temos:

- Ou o valor era um número inteiro ou não
 - Se valor era int, então o valor é convertido para int
 - Se não, o node está configurado para não falhar mas sim passar valor “MISSING” ou vais fácil de perceber “NULL”, este error é filtrado depois no Rule Engine

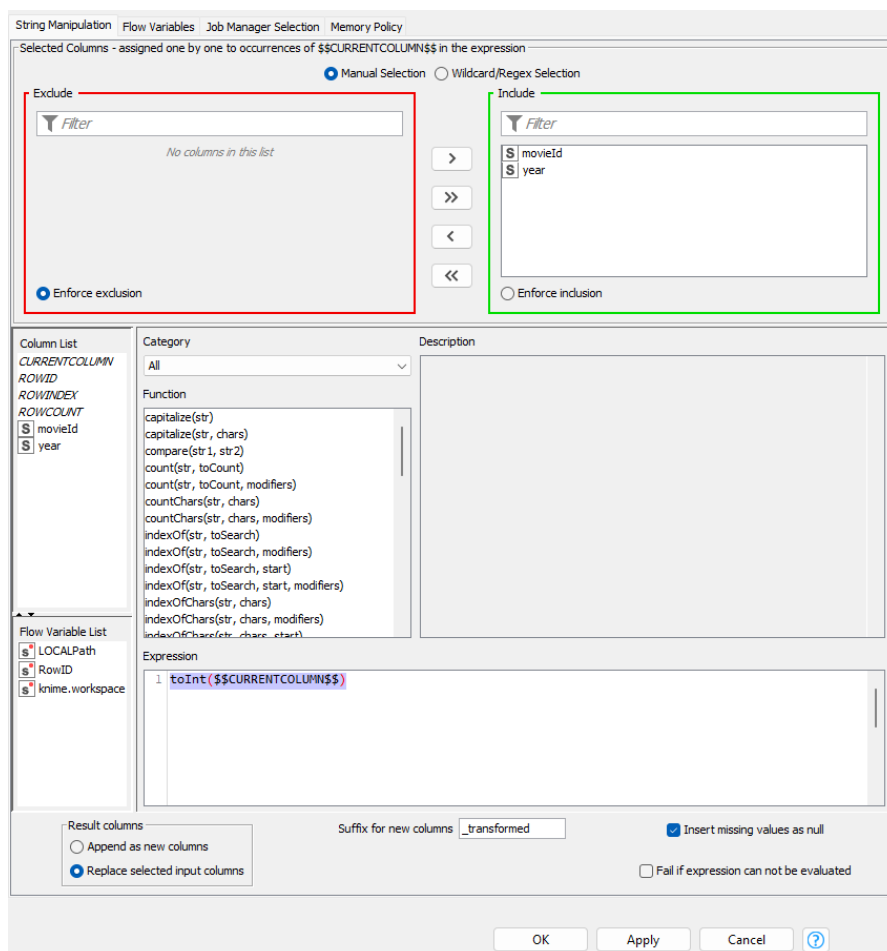


Figura 7 – Transformação de Inteiros

Transformação de dados: Strings

O processo inicial é igual para todos os dados, leitura em XPath e separação de colunas.

As colunas filtradas são: Title e Language

A primeira String Manipulation é para ambas colunas e tem como finalidade remover:

- Espaços, “\n” (Enters) e “\t”(Tabs), do início e fim dos mesmos
- Se a própria string em si for apenas um espaço, “\n” ou “\t” é substituída por “” (string vazia)

Expressão:

regexReplace(\$\$CURRENTCOLUMN\$\$, "^[\s\\n\\t]+|[\s\\n\\t]+\$", "")

A segunda String Manipulation afeta apenas a coluna Language e tem como finalidade:

- Remover números da string, nenhuma língua é representada numericamente
- Passar para letras maiúsculas, pt -> PT

Expressão:

upperCase(regexReplace(\$language\$, "[0-9]", ""))

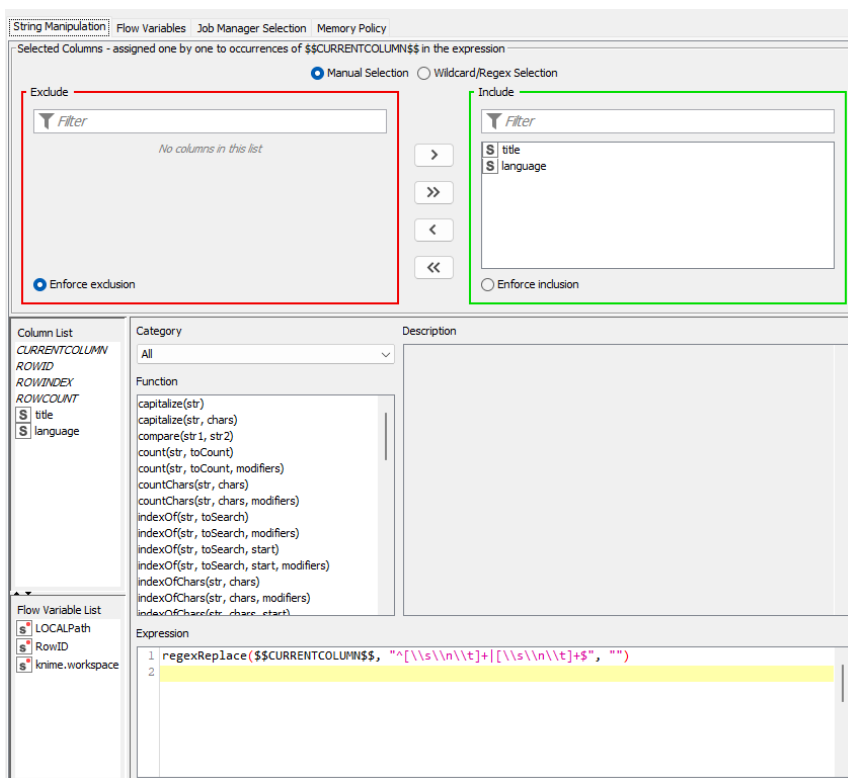


Figura 8 - Transformação de Strings (Regex)

Transformação de dados: Listas

O processo inicial é diferente de todos os dados anteriores, leitura em XPath, separação de colunas e separação da lista.

As colunas filtradas são: Subtitles, MovieLength, StartSub, EndSub, MovieID

Apesar de MovieLength (duração do filme) não ser uma lista, é necessário que ela acompanhe estas outras colunas para verificação de erros e ser tratada como uma variável Date&Time em conjunto com StartSub (início de legendas) e EndSub (fim de legenda).

MovieID também não é uma lista mas é enviada para listas para no fim do processo, quando voltarem a ser agregados todos os valores, as legendas serem inseridas na linha correta correspondendo os MovieIDs.

Separação de listas

Separação da lista é pegar em cada valor da lista e passar para uma linha, e como o XPath lê os valores todos do XML na mesma ordem, ao desenvolver a lista para linha ficamos com, subtitle + startSub + endSub da mesma ordem que estavam no XML.

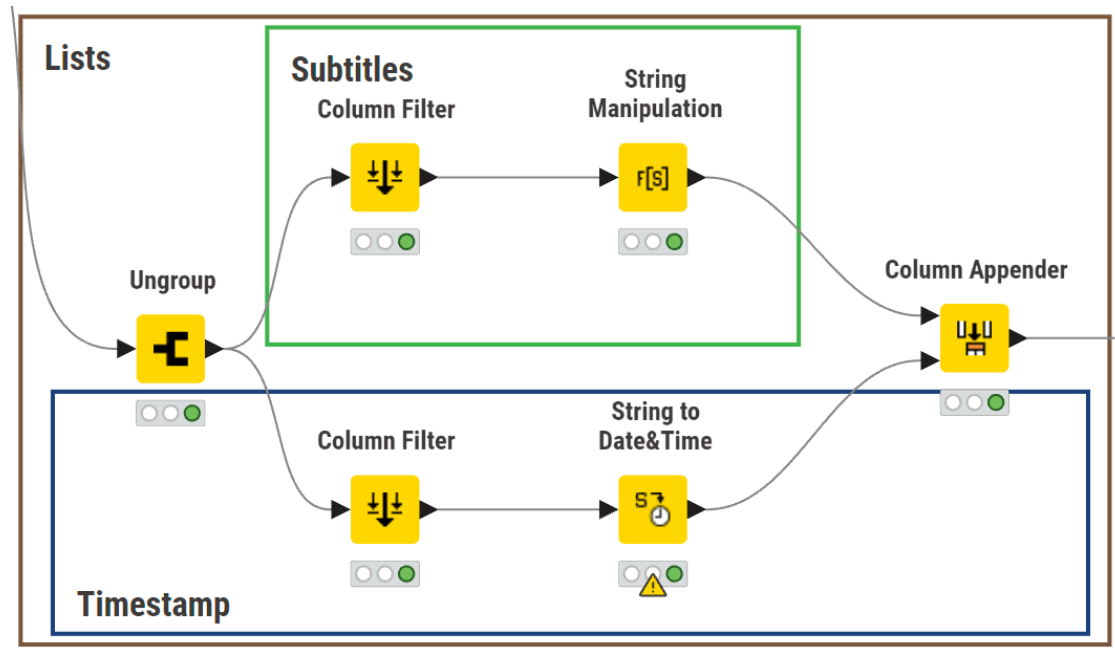


Figura 9 – Listas - Workflow

Legendas

A String Manipulation tem como finalidade remover:

- Espaços, “\n” (Enters) e “\t” (Tabs), do início e fim dos mesmos
- Se a própria string em si for apenas um espaço, “\n” ou “\t” é substituída por “” (string vazia)

Expressão:

regexReplace(\$\$CURRENTCOLUMN\$\$, "^[\\s\\n\\t]+|[\\s\\n\\t]+\$", "")

A expressão é a mesma aplicada em para as restantes strings mas devido a ser uma lista inicialmente foi necessário fazer em um caminho diferente e pois foi necessário separar as listas para que elas fiquem na ordem correta.

Hórrarios (Timestamps)

O node String to Date&Time tem como finalidade:

- Converter string para Date&Time para formato HH:MM:SS
- Se falhar a conversão converter para “MISSING”

Configuração:

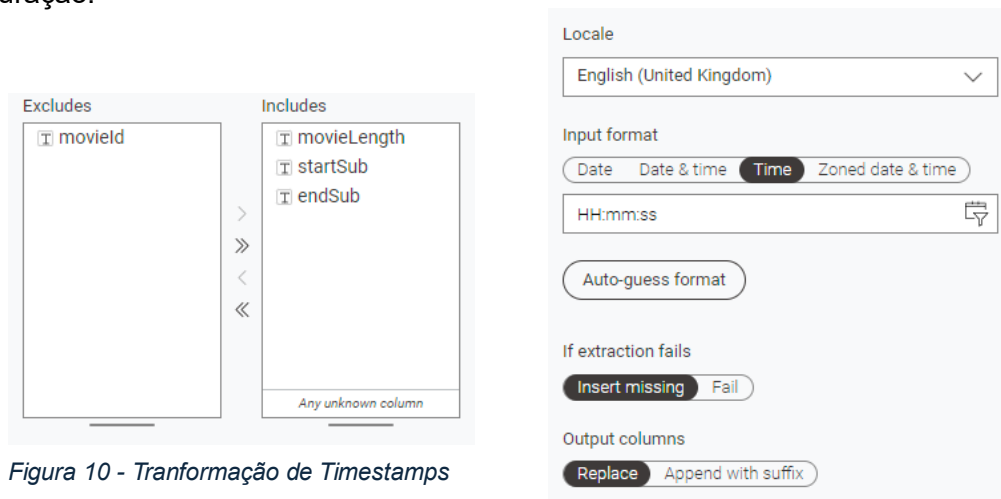


Figura 10 - Transformação de Timestamps

A expressão é a mesma aplicada em para as restantes strings mas devido a ser uma lista inicialmente foi necessário fazer em um caminho diferente e pois foi necessário separar as listas das restas colunas para que elas fiquem na ordem correta.

Agregação de dados

Após o tratamento/transformação de dados, é necessário juntar os dados de forma/ordem correta novamente, para que isto seja possível, MovieID é enviado juntamente com as listas para a secção “Listas”.

Para fazer esta agregação as Listas que previamente tinham sido repartidas, voltam a ser unidas para associar uma lista com um filme.

E para fazer a ligação correta todas as legendas mantêm o MovieID a que pertencem e quando são reduziadas a lista fica associadas por linha ao filme a que pertence, logo só é preciso juntar os dados onde os MovieID são iguais.

Para realizar esta agregação então é feito:

- Remover legendas com MovieID e Timestamps iguais
- Passagem de Date&Time das timestamps para String
- Redução de linhas com Listas (GroupBy node)
- Junção por MovieID (Joiner node), como os MovieIDs são diferentes, um é tipo Int e o outro tipo String, a verificação é por valor String

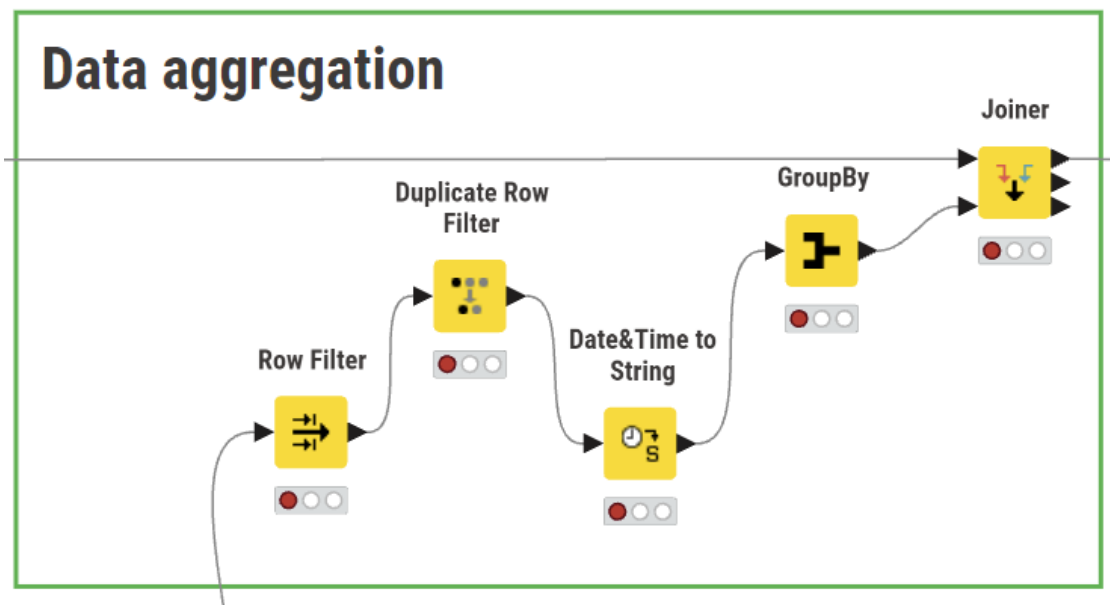


Figura 11 –Agregação de dados - Workflow

Configuração (Joiner):

Matching Criteria

Match

All of the following Any of the following

Criterion 1

Top input ('left' table)

movieId

Bottom input ('right' table)

movieId

+ Add matching criterion

Compare values in join columns by

String representation

Output Columns

Top input ('left' table)

Manual Wildcard Regex Type

Search Aa

Excludes

No columns in this list.

Includes

movieId

year

title

language

Any unknown column

Bottom input ('right' table)

Manual Wildcard Regex Type

Search Aa

Excludes

movieId

Includes

subtitles

startSub

endSub

movieLength

Any unknown column

☐ Merge join columns

Include in Output

☒ Matching rows

☐ Left unmatched rows

☐ Right unmatched rows




Figura 12 - Configuração de agregação

Retorno de JSON para API Endpoint

No fim do workflow, quando os dados já foram lidos, tratados e verificados os mesmos são transformados em JSON para enviar como Request Body para um endpoint em ASP.NET.

Este endpoint recebe o JSON como parametro, e com Microsoft.EntityFrameworkCore Scaffold, que utiliza LINQ para consultar a base de dados, a informação é toda guardada, filme e legendas.

Configuração (POST Request):

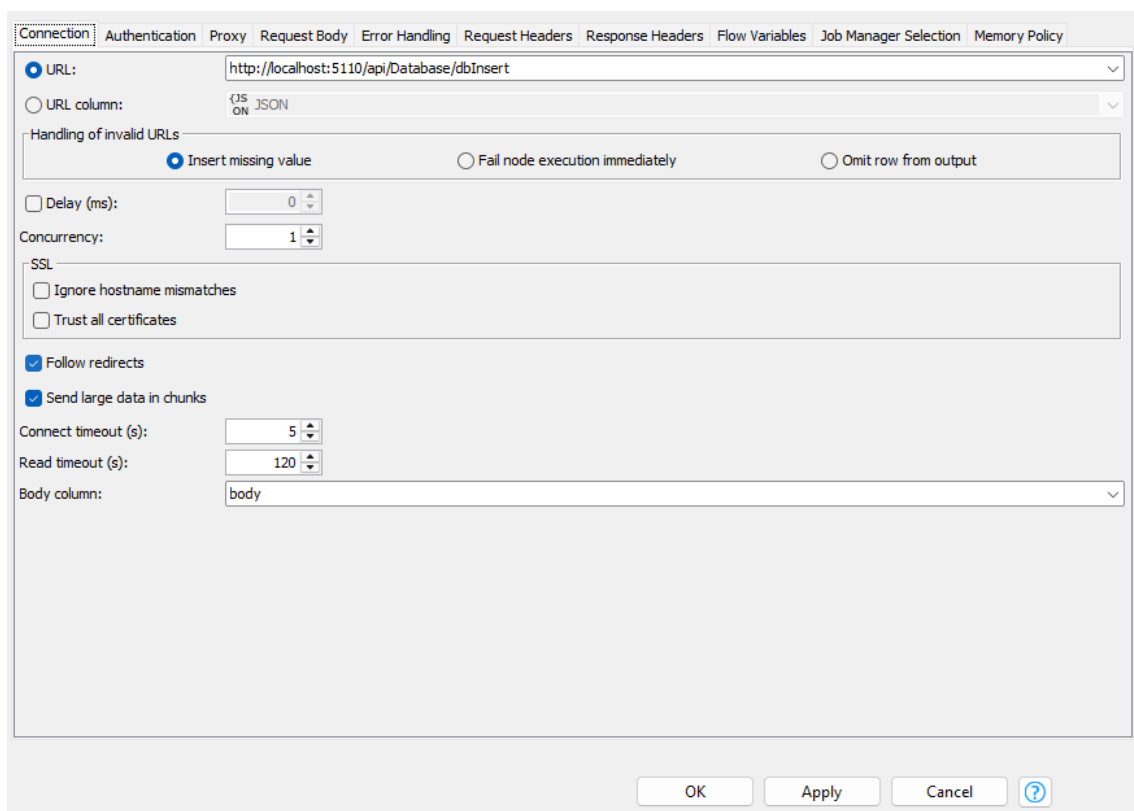
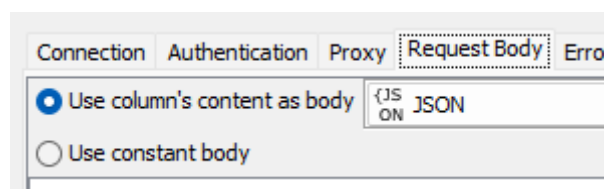


Figura 13 - Configuração POST Request



Escrita de Logs

A escrita de Logs deriva de um Rule Engine ou do Java Snippet (validação de XML + DTD).

O Rule Engine verifica condições e cria uma coluna chamada “prediction” com o resultado.

Neste workflow existem 3 tipos de logs:

1. XML inválido de acordo com DTD
2. Legendas / Hórrarios (Timestamps) inválidos
3. Falta de informação

Todos os Logs são escritos da mesma forma:

- Variável de tempo, Timestamp é criada
- Passar variável para coluna
- Escrever Timestamp + conteúdo com erro + Mensagem de erro

Exemplo de Log:

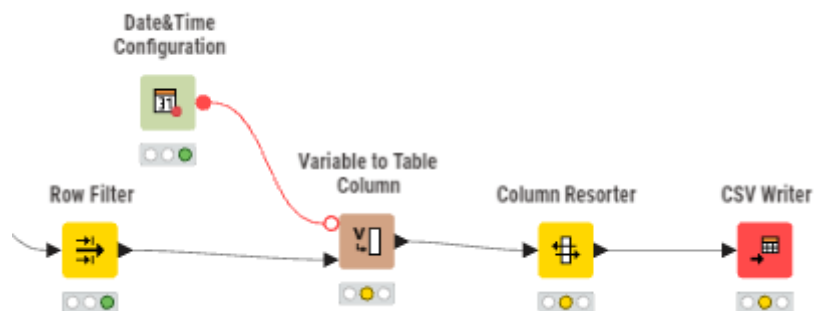


Figura 14 - Exemplo Log - Workflow

Legendas / Hórrarios (Timestamps) inválidos

Após um Rule Engine é criada uma coluna “prediction”, com o resultado da validação, tudo o que não for válido é escrito em ./Logs/subtitle_error.log.

Regras do Rule Engine verificam se:

- falta informação
- se informação está errada
 - se endSub, fim de legendas, não pode ser menor, antes, que startSub, início da legenda
 - se os startSub e endSub passam dos limites do filmes, MovieLength
 - StartSub < 0, legenda começa antes do filme
 - EndSub > MovieLength, legenda acaba depois do filme

Rule Engine:

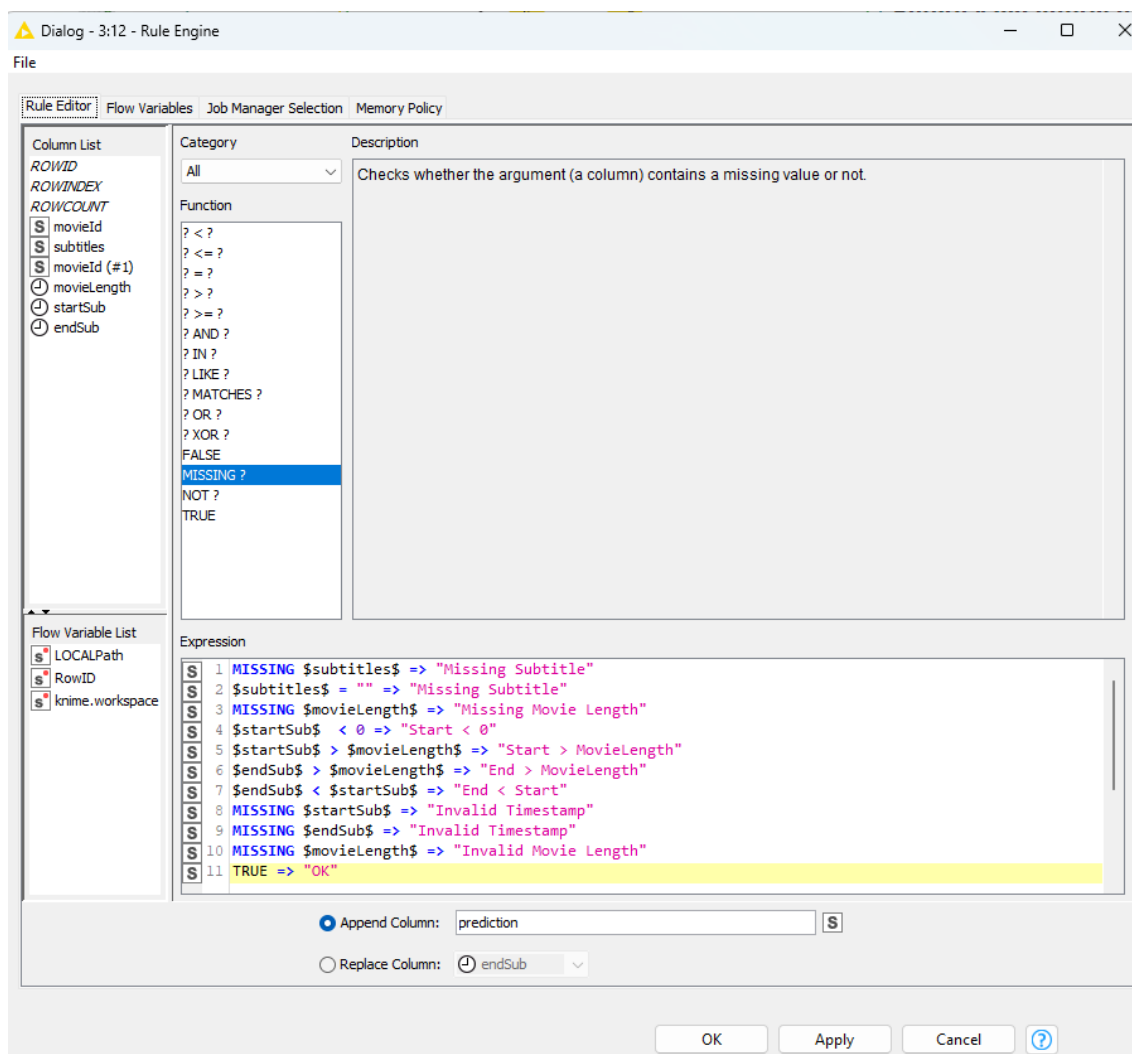


Figura 16 - Rule Engine para Legendas e Timestamps

Workflow:

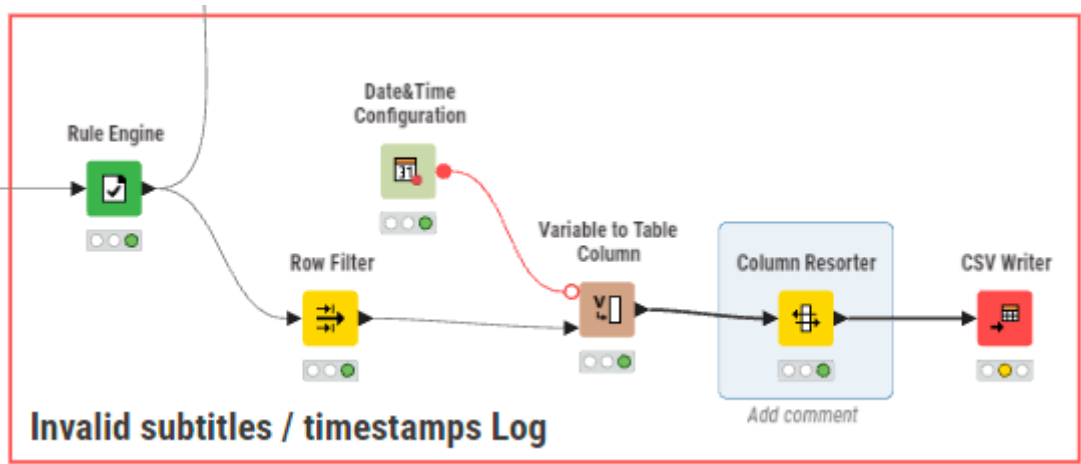


Figura 17 - Log para legendas e timestamps - Workflow

Exemplo:

#	RowID	date-input	movieId	movieLength	subtitles
1	Row1_	2025-10-18T03:08:09.997	1234	00:05	This is an exemple 3
2	Row1_	2025-10-18T03:08:09.997	1234	00:05	This is an exemple 4
3	Row1_	2025-10-18T03:08:09.997	1234	00:05	This is an exemple 5
4	Row1_	2025-10-18T03:08:09.997	1234	00:05	
5	Row3_	2025-10-18T03:08:09.997	1534	00:05	This is an exemple 3
6	Row3_	2025-10-18T03:08:09.997	1534	00:05	This is an exemple 4
7	Row3_	2025-10-18T03:08:09.997	1534	00:05	This is an exemple 5
8	Row3_	2025-10-18T03:08:09.997	1534	00:05	TimeStamp error
9	Row3_	2025-10-18T03:08:09.997	1534	00:05	

startSub	endSub	prediction
00:04:59	00:05:01	End > MovieLength
00:05:01	00:04:01	Start > MovieLength
00:04:01	00:03:01	End < Start
00:03:01	00:04:01	Missing Subtitle
00:04:59	00:05:01	End > MovieLength
00:05:01	00:04:01	Start > MovieLength
00:04:01	00:03:01	End < Start
00:03:01	00:03:01	Invalid Timestamp
00:03:01	00:04:01	Missing Subtitle

Falta de informação

Após um Rule Engine é criada uma coluna “prediction”, com o resultado da validação, tudo o que não for válido é escrito em ./Logs/missing_data.log.

Regras do Rule Engine verificam se:

- falta informação
 - Válida novamente legendas, pois apenas legendas inválidas são removidas no processo anterior
 - Ex: Filme X tem 5 legendas, 3 inválida, as 2 válidas ainda são submetidas
 - Este filtro é para não passar o filme para o JSON caso não tenha nenhuma legendas válida

Rule Engine:

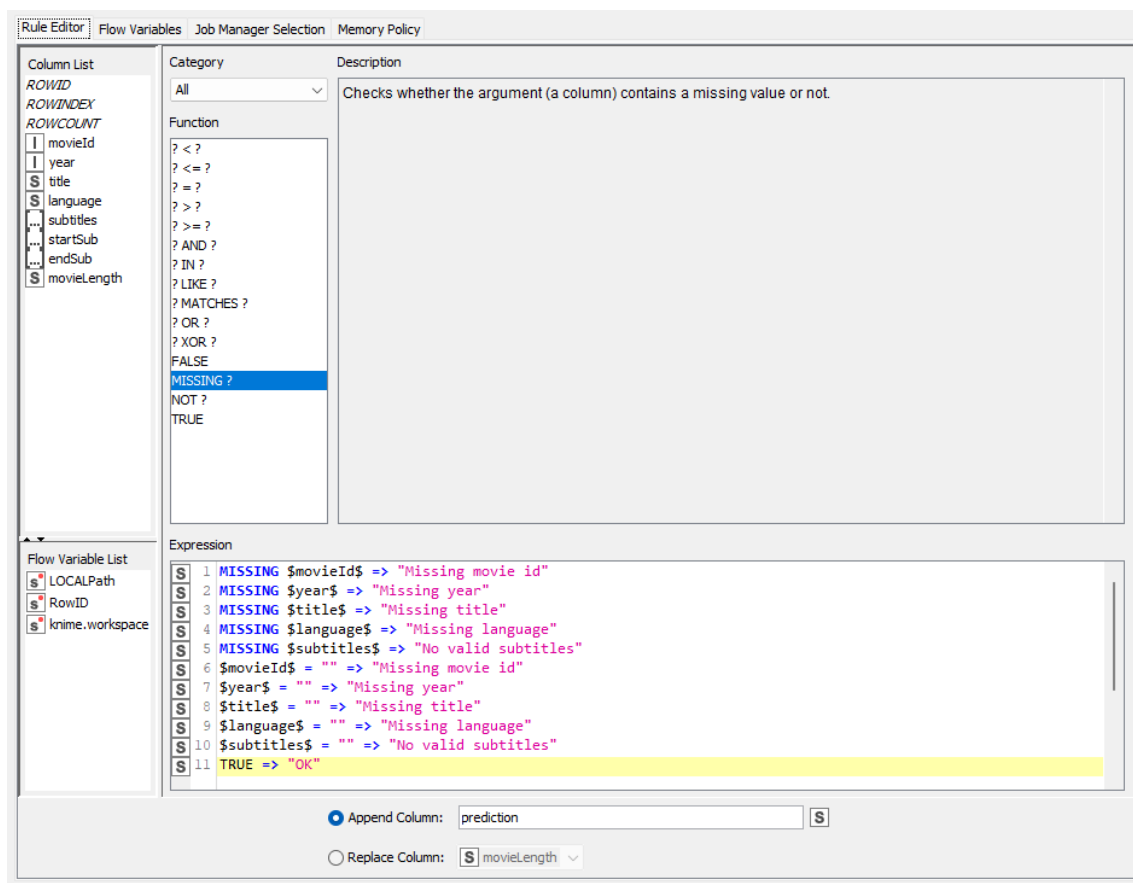


Figura 18 - Rule Engine para falta de dados

Workflow:

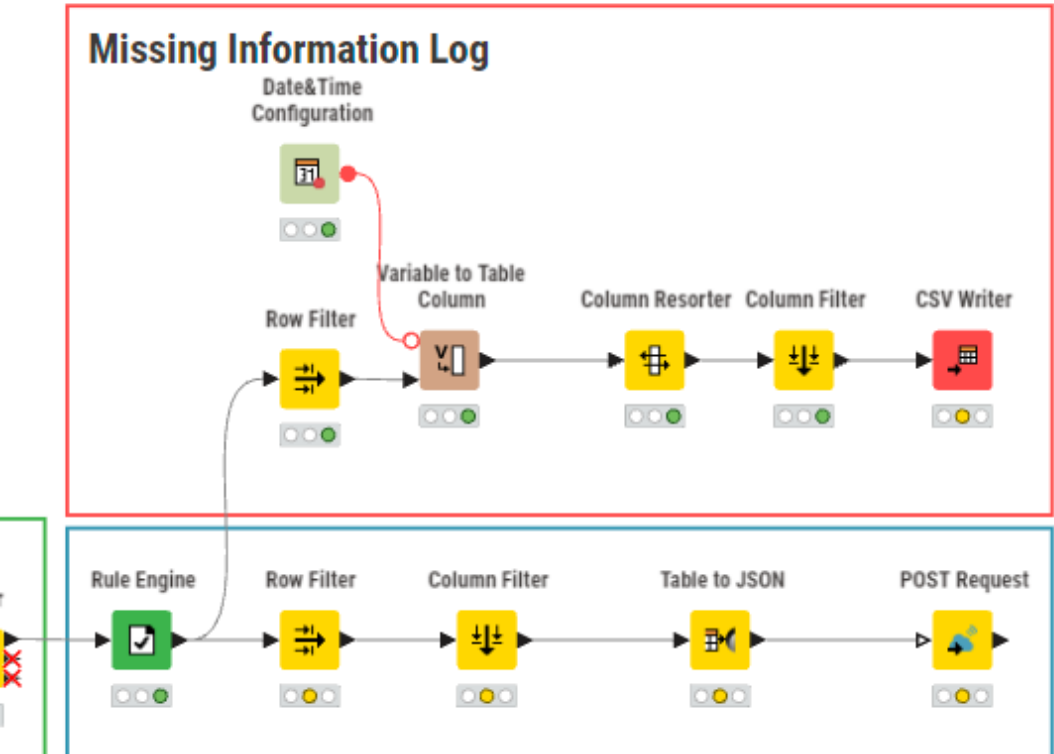


Figura 19 - Log para dados em falta - Workflow

Exemplo:

Rows: 4 Columns: 7				Table Statistics		
#	RowID	date-input	movieid	title	year	
1	Row4_	2025-10-18T05:45:54.171	1038392	The Conjuring: Last Rites		
2	Row1_	2025-10-18T05:45:54.171	1234	Error Movie		
3	Row3_	2025-10-18T05:45:54.171	1534	Filme Teste	2025	
4	Row2_	2025-10-18T05:45:54.171	755898		2025	

language	movieLength	prediction
EN	00:05:00	Missing year
PT	00:05:00	Missing year
	00:05:00	Missing language
EN	01:31:00	Missing title

ASP.NET

Criado para:

- Passar as legendas na base de dados para a aplicação web
- Fazer upload de ficheiros XML para a pasta do workflow do Knime
- Inserir dados em JSON na base de dados

Detalhes:

- Arquitetura N-Tier, MVC
- Microsoft.EntityFrameworkCore
 - Core
 - Design
 - SqlServer
 - Tools
 - Utiliza LINQ para consulta

Web APP-React

Criado para:

- Mostrar catálogo de filmes
- Mostrar página de cada filme
 - Se existirem legendas para este filme na base de dados
 - Mostrar legendas no formato XML válido
 - Se não existirem
 - Mostrar exemplo
- Permitir que o utilizador
 - Descarregue o XML Exemplo
 - XML com legendas da base de dados
 - Faça upload de um XML

Detalhes:

- Catálogo de filmes adquirido em:
 - The Movie Database
 - <https://developer.themoviedb.org>

Estutura de ficheiros

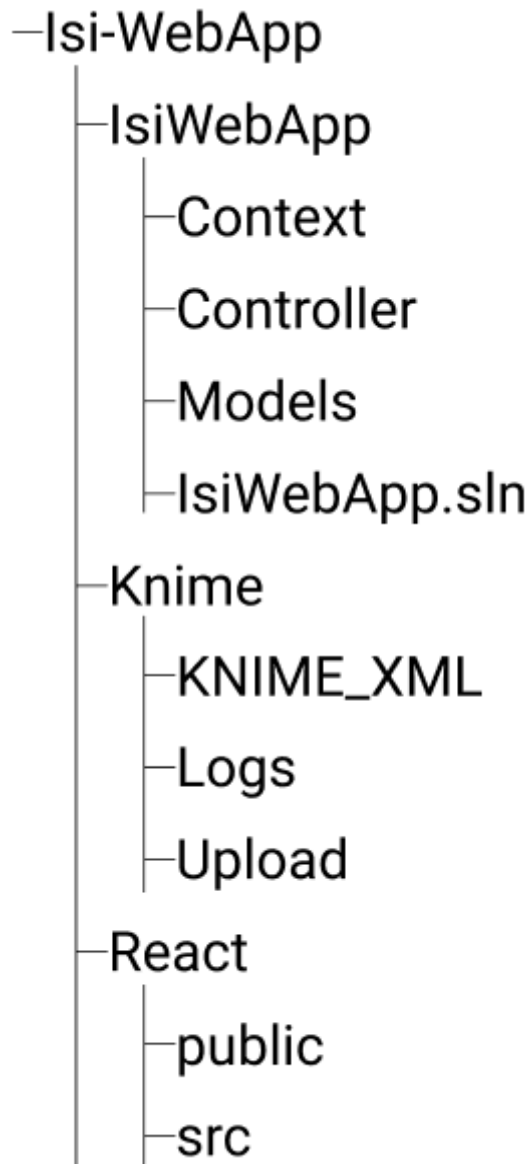


Figura 20 - Estrutura do Projeto

Bibliografia

- KNIME Documentation. KNIME Analytics Platform User Guide — KNIME AG, 2025.
 - <https://docs.knime.com/>
- Mozilla Developer Network (MDN). React Documentation: Components, Props and State — MDN Web Docs, 2025.
 - <https://developer.mozilla.org/>
- React Team. React: A JavaScript library for building user interfaces — Meta Platforms, Inc., 2025.
 - <https://react.dev/>
- The Movie Database (TMDb). API Documentation — The Movie Database, 2025.
 - <https://developer.themoviedb.org/>
- Microsoft. ASP.NET Core Web API Documentation — Microsoft Learn, 2025.
 - <https://learn.microsoft.com/aspnet/core>
- Microsoft. Entity Framework Core Documentation — Microsoft Learn, 2025.
 - <https://learn.microsoft.com/ef/core>
- Oracle. Java Platform, Standard Edition – API Specification — Oracle Corporation, 2024.
 - <https://docs.oracle.com/javase/>

Video da Aplicação

