Proteção de Dados (Anonimização & Encriptação)

Relatório TP1

Integração de Sistemas de Informação

Aluno: Adelino Daniel da Rocha Vilaça - a16939
Docente: Professor Luis Gonzaga Martins Ferreira

Licenciatura em Engenharia de Sistemas Informáticos (PL)

3º ano

Barcelos | outubro, 2024

Lista de Abreviaturas e Siglas

Índice de Figuras

Índice

1. Enquadramento

Este trabalho foi realizado no âmbito da Unidade Curricular de Integração de Sistemas de Informação, do curso de Engenharia de Sistemas Informáticos (Pós-Laboral). O objetivo deste primeiro trabalho prático é o desenvolvimento de competências em ferramentas ETL e aplicação de técnicas de processamento de dados, com foco na integração, transformação e análise de conjuntos de dados.

Este projeto consistia inicialmente na transformação dos dados utilizados anonimização e encriptação, usando AES como forma mais eficiente para manuseamento dos dados, mas infelizmente não foram encontradas formas de adicionar bibliotecas necessárias para utilização dessa técnica de encriptação de dados no ambiente integrado de Python + Conda, apenas sendo possível a utilização de hashing e codificação com Base64, mas sendo possível surgirem alterações caso uma solução seja encontrada.

Ao longo do projeto foram feitas algumas alterações a nível estrutural para acrescentar a utilização de codificação de dados recebidos através de uma API para uma Base de Dados e eventual exportação/demonstração gráfica dos dados.

2. Problema

O principal objetivo deste trabalho é demonstrar e resolver problemas relacionados com o tratamento, filtragem, transformação e análise final dos dados acerca de duas possíveis escolhas: Noticias relacionadas com Cibersegurança e/ou Dados com informação pessoal (Nomes, Passwords, etc). Para a primeira escolha foi usada a API para input de dados: **Cibersecurity API (NewsAPI)**. Na segunda escolha foram gerados dois Excels, um com dados pessoais sem passord e outro com password, usando a ferramenta online **Mockaroo**.

3. Estratégias utilizadas

Para este projeto, foram necessários operadores/nodes, tanto internos da ferramenta KNIME, como externos, sendo necessário utilizar o hub online da plataforma para investigação de nodes que fossem mais bem aproveitados no contexto do problema.

Nodes/Operadores fundamentais:

MySQL Connector: Para acesso eventual à base de dados para armazenar as notícias recebidas pela API;

Get Request: Para input de dados de notícias a serem tratados eventualmente ao longo do projeto, mas recebidos em JSON;

JSON Path: Para filtragem da informação que apenas necessitamos vindo da API;

Ungroup: Separa os dados filtrados para criação de linhas/colunas tendo em conta os parâmetros utilizados, neste caso name, title, etc;

Column Filter: Para excluir algumas colunas que possam ser inseridas, mas que não sejam precisas, como Status, Content Type, etc;

DB Insert: Para inserir os dados tratados da API para uma tabela na BD;

DB Table Selector: Para confirmação se os dados foram inseridos como pretendido na tabela da base de dados sem ser necessário validar pela Shell do MySql;

Python Script: Para utilização de alguns scripts para codificação dos dados, criptografia das passwords e para anonimização de alguns dados como o primeiro e último nome;

Excel Reader: Para input de dados a tratar e para leitura de dados que foram codificados para voltarem a serem tratados e descodificados;

Excel Writer: Para gerar um output em ficheiro Excel;

Column to XML: Para transformação das colunas de dados em dados XML;

XML Column Combiner: Para unificação/merge das colunas em um só;

XML Writer: Para output em XML.

4. Transformações

a. Transformações relativas à API

i. Filtragem de dados da API

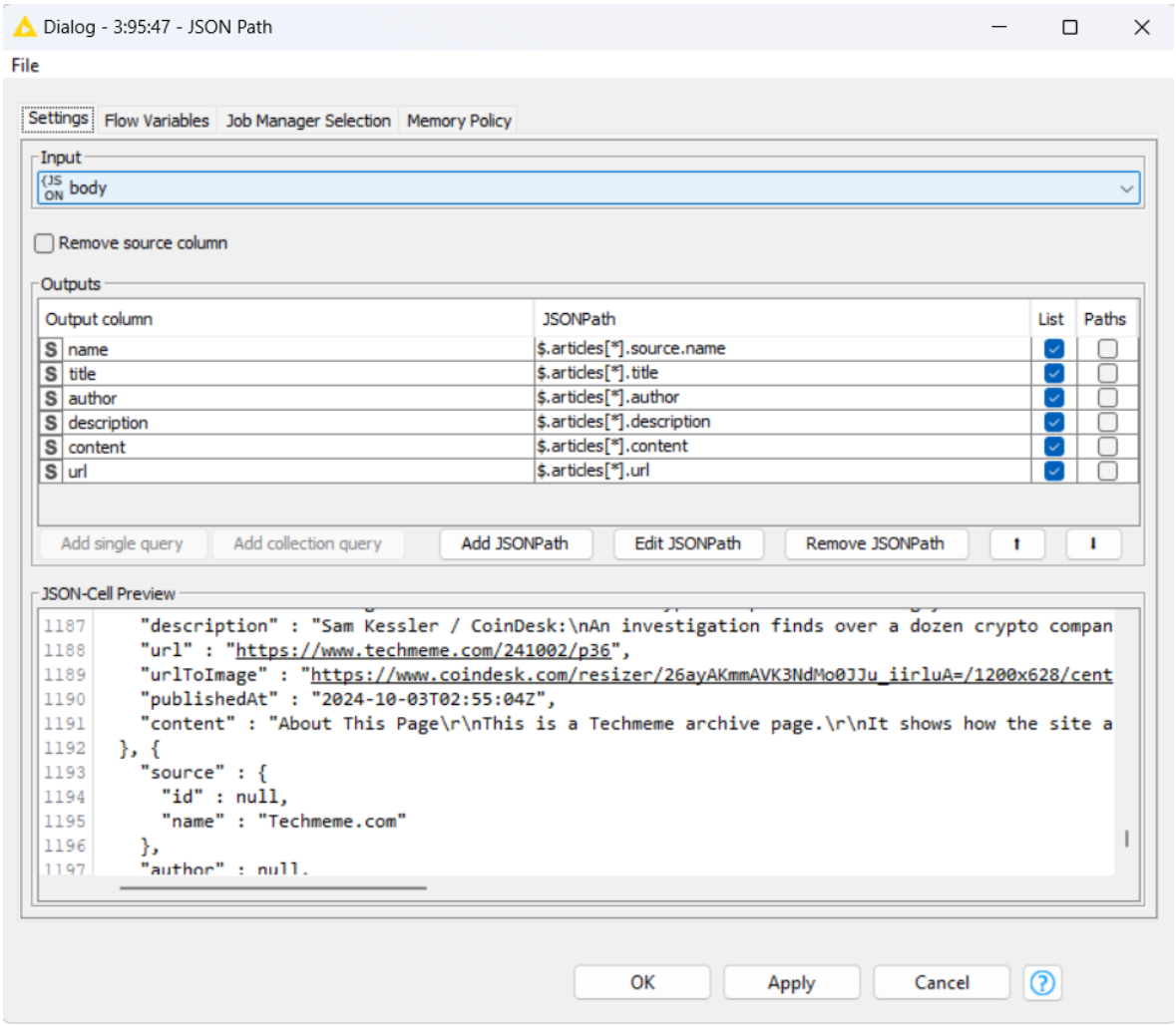

Figura 1 - Visualização da filtragem de dados da API usando o node JSON Path

Após a execução do node GET Request para chamada dos dados da API, é feita uma filtragem ao JSON para apenas serem retirados os dados que necessitamos trabalhar ao longo do projeto, sendo eles o nome, título da noticia, autor, descrição, conteúdo e o link url onde a noticia foi publicada.

ii. Separação dos dados

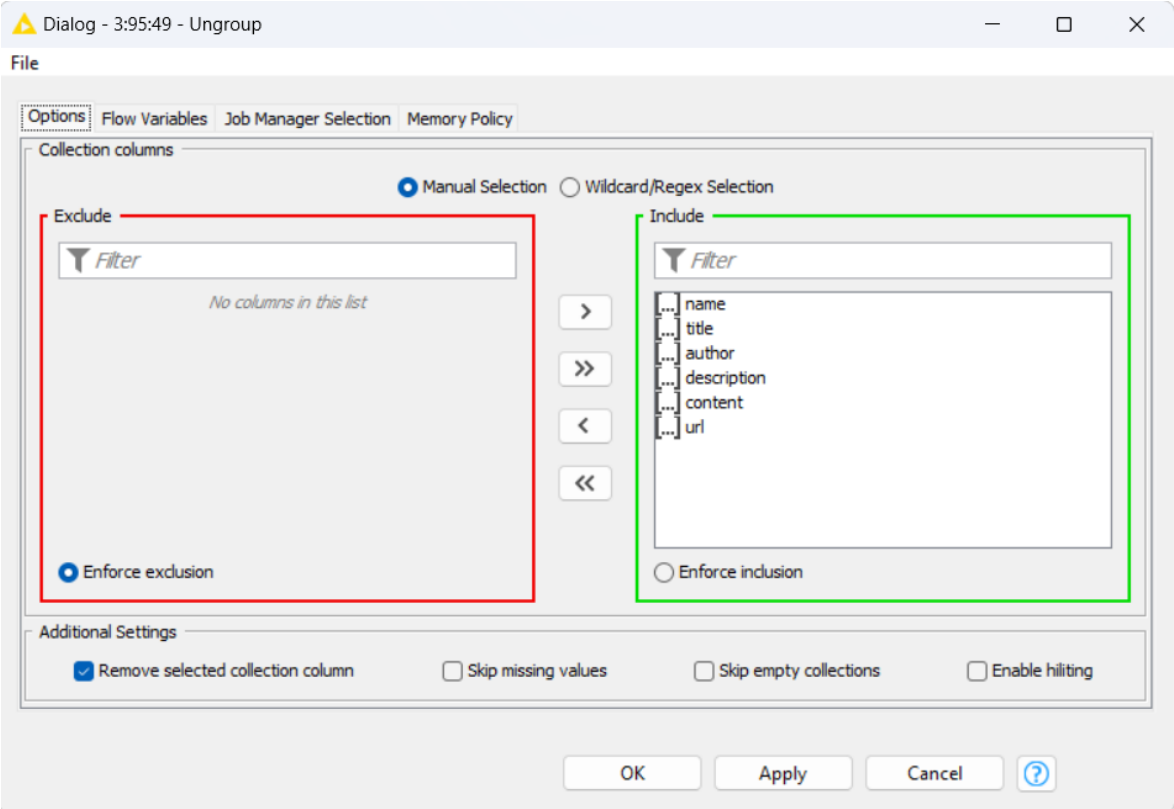

Figura 2 - Separação dos dados recebidos pela filtragem da API usando o node Ungroup

Através do operador Ungroup, fazemos então a separação das linhas de dados tendo em conta a lista de atributos por coluna.

iii. Separação dos dados

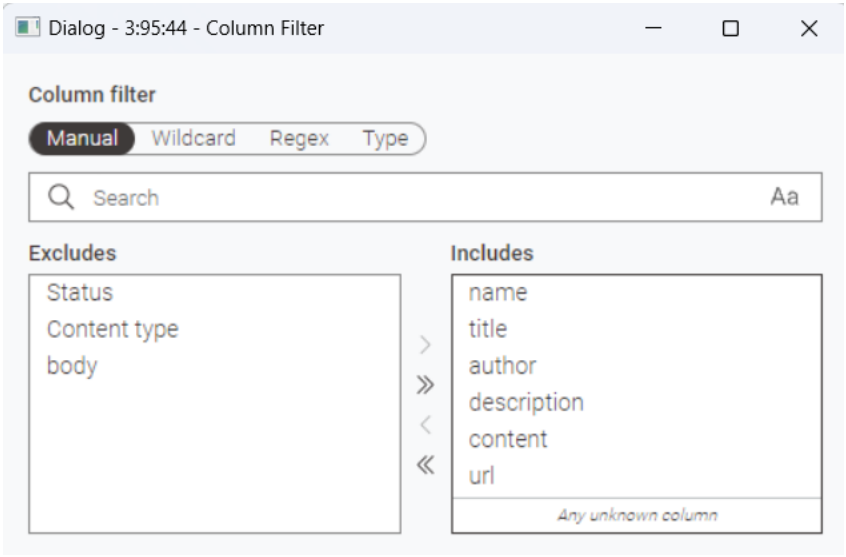

Figura 3 - Filtragem das colunas necessárias utilizando o node Column Filter

Após a separação das linhas de dados pela lista de colunas presente, é feita a filtragem de colunas que são realmente necessárias, excluindo colunas como Status, Content type e body, sendo que não irão ser trabalhadas/tratadas.

b. Anonimização de Dados

i. Filtragem de colunas desnecessárias

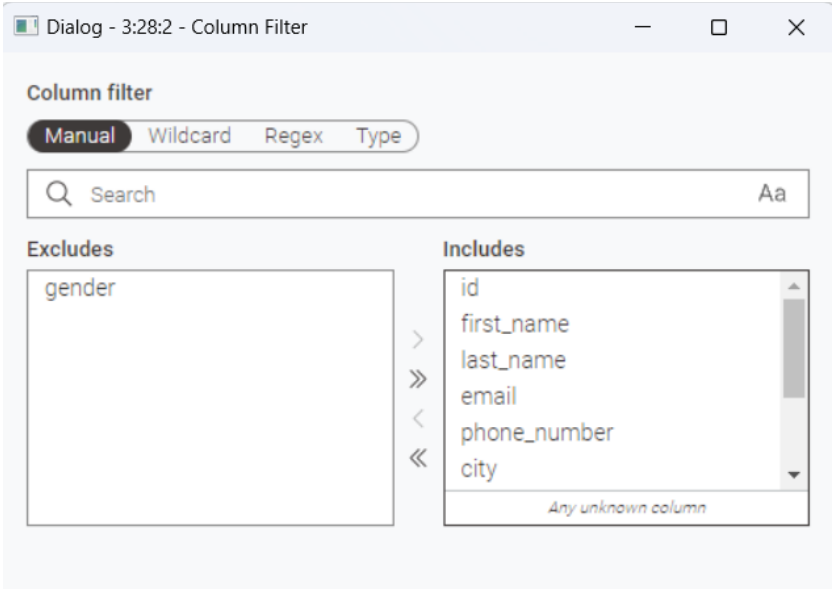

Figura 4 - Filtragem das colunas necessárias utilizando o node Column Filter

Para iniciação da anonimização dos dados, são excluídas as colunas desnecessárias, neste caso a coluna do sexo, e mantidas as colunas que pretendemos transformar.

ii. Criação de uma tabela para contador de ID's

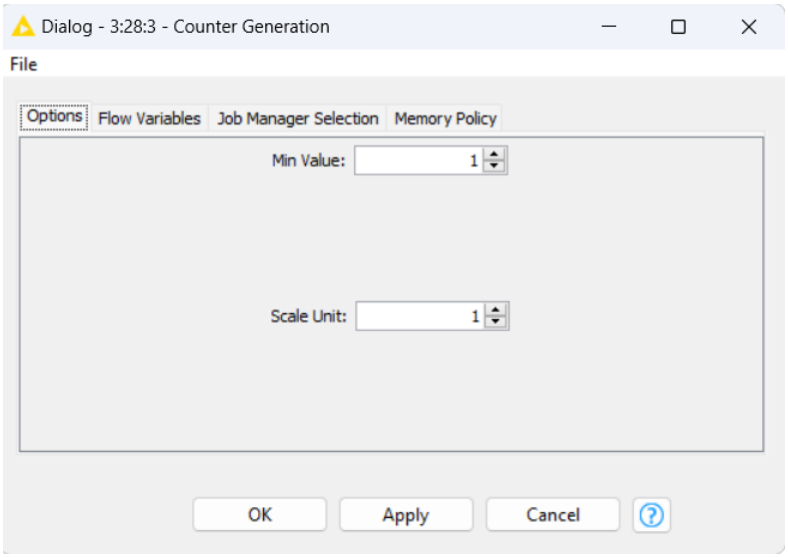

Figura 5 - Contador de ID's utilizando o node Counter Generation

É utilizado o node Counter Generation para contabilizar os ID's numa nova tabela, permitindo uma melhor gestão dos dados.

iii. Criação de uma tabela para contador de ID's

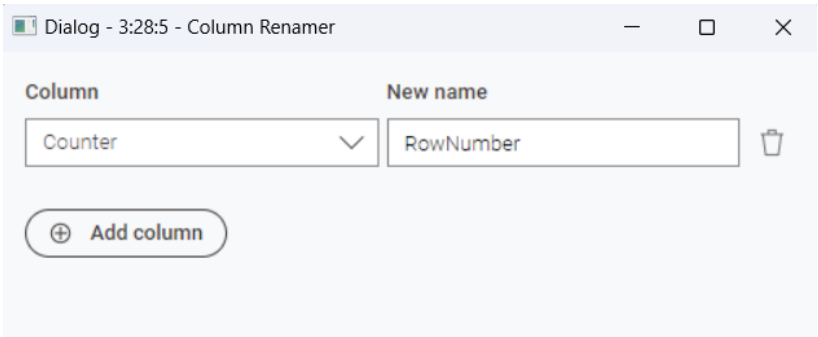

Figura 6 - Renomeação da tabela de contador previamente criada para RowNumber

Após a criação da tabela para contabilizar os ID's, é utilizado o node Column Renamer para alterar a coluna para "RowNumber".

iv. Expressões para modificar os valores de first e last name nos dados

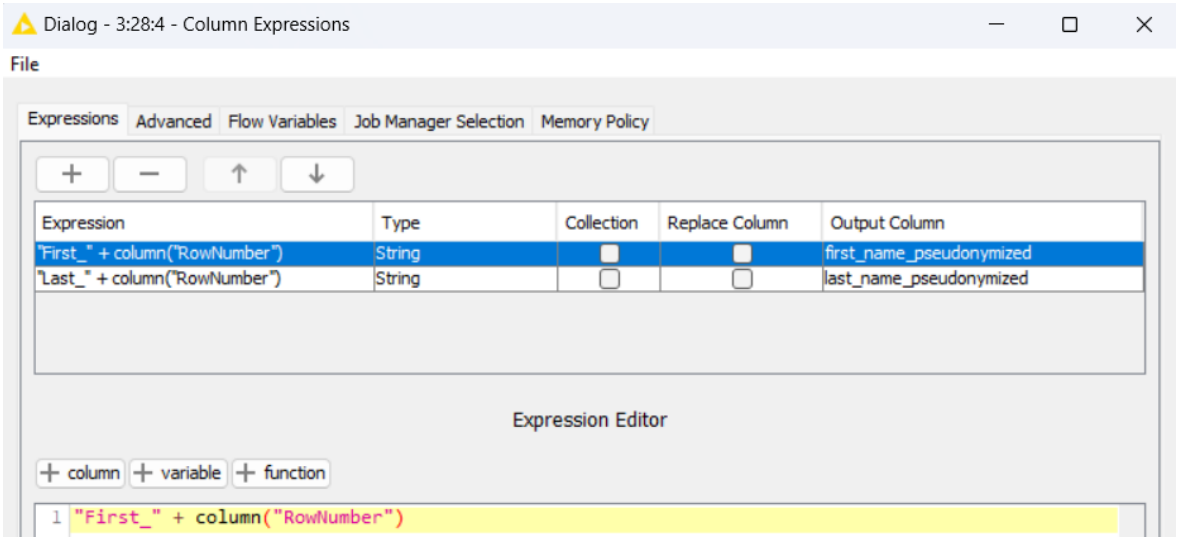

Figura 7 - Utilização de ERs para modificar o 1º e último nome

Através do node Column Expressions, é possível utilizar expressões para alteração de certos dados ao entrarem numa coluna estabelecida, neste caso é utilizado a coluna previamente criada "RowNumber" e é acrescentado um "First_" e um "Last_" antes do ID correspondente à linha de dados, criando, por exemplo, um First_17 com ID 17 subentendido, de forma a proteger dados sensíveis que poderiam estar expostos, não existindo um processo de anonimização.

v. Exclusão das tabelas desnecessárias

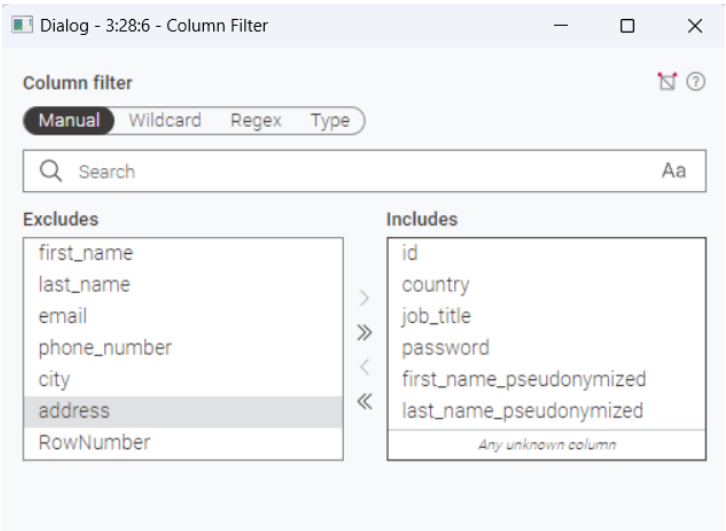

Figura 8 - Exclusão das colunas desnecessárias utilizando o node Column Filter

É utilizado novamente o node de Column Filter para excluir as colunas que não desejamos apresentar na tabela final.

vi. Modificação da ordem das colunas

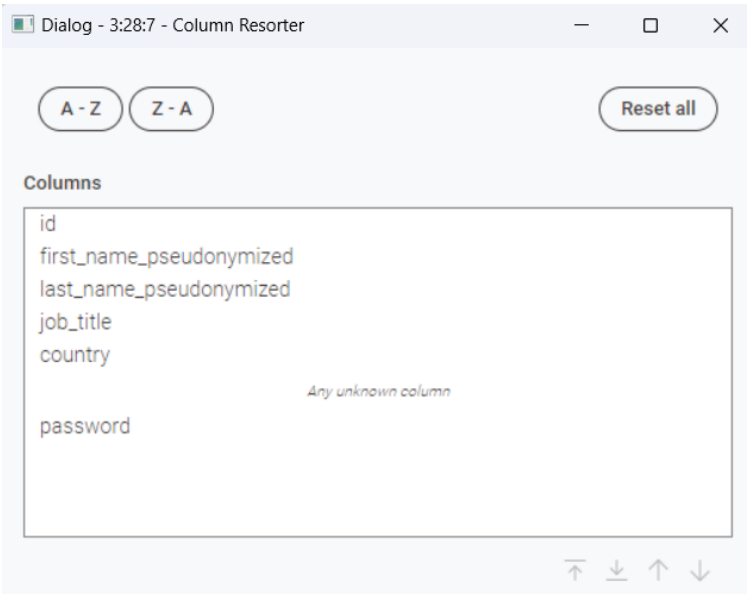

Figura 9 - Modificação da ordem das colunas desejadas

Através do node Column Resorter é possível rearranjar as tabelas pela ordem de preferência, meramente para um melhor aspeto estrutural.

c. Codificação e Hashing dos Dados

i. Codificação em Base 64

```python
import knime.scripting.io as knio
import pandas as pd
from base64 import b64encode

# Load your input table into a DataFrame
input_table = knio.input_tables[0].to_pandas()

# Define a function to encode a cell using base64
def encode_cell(cell_value):
    # Convert the cell to a string (if it's not already) and encode
    if pd.isna(cell_value):  # Handle NaN values
        return cell_value
    else:
        return b64encode(str(cell_value).encode()).decode()

# Apply the encoding function to every cell in the DataFrame
encoded_table = input_table.applymap(encode_cell)

# Output the encoded DataFrame to the output table
knio.output_tables[0] = knio.Table.from_pandas(encoded_table)
```

Figura 10 - Código Python da Codificação dos dados em Base64

Este código em Python aplica codificação base64 a cada célula de uma tabela de dados recebida de uma API de cibernotícias ou de um ficheiro Excel com dados pessoais. Ao converter as células para texto e codificá-las em base64, o script preserva os valores originais (incluindo os NaN /

Nulos para valores em branco). Depois, a tabela codificada é exportada para ser inserida numa base de dados.

ii. Descodificação de Base 64

```python
import knime.scripting.io as knio
import pandas as pd
from base64 import b64decode

# Load your input table into a DataFrame
input_table = knio.input_tables[0].to_pandas()

# Define a function to decode a cell using base64
def decode_cell(cell_value):
    # Convert the cell to a string (if it's not already) and decode
    if pd.isna(cell_value):  # Handle NaN values
        return cell_value
    else:
        try:
            return b64decode(str(cell_value).encode()).decode()
        except Exception:
            return cell_value  # In case the value wasn't encoded

# Apply the decoding function to every cell in the DataFrame
decoded_table = input_table.applymap(decode_cell)

# Output the decoded DataFrame to the output table
knio.output_tables[0] = knio.Table.from_pandas(decoded_table)
```

Figura 11 - Código Python da Descodificação dos dados em Base64

Este código em Python recebe uma tabela de entrada no KNIME, converte-a para um DataFrame e aplica uma função de descodificação base64 a cada célula. Para cada valor na célula, verifica se está vazio ou não codificado em base64, e tenta descodificá-lo; caso contrário, mantém o valor original. A tabela descodificada resultante é então exportada para uma tabela de saída, pronta para uso em bases de dados ou outras análises.

iii. Hashing SHA256

```python
import knime.scripting.io as knio
import pandas as pd
import hashlib

# Load your input table into a DataFrame
input_table = knio.input_tables[0].to_pandas()

# Define a function to hash a cell using SHA-256
def hash_password(cell_value):
    # Convert the cell to a string (if it's not already) and hash
    if pd.isna(cell_value):  # Handle NaN values
        return cell_value
    else:
        # Hash the cell value using SHA-256
        hashed_value = hashlib.sha256(str(cell_value).encode()).hexdigest()
        return hashed_value

# Apply the hashing function only to the 'password' column
input_table['password'] = input_table['password'].apply(hash_password)

# Output the DataFrame with the hashed password to the output table
knio.output_tables[0] = knio.Table.from_pandas(input_table)
```

Figura 12 - Código Python da utilização de hashing nas passwords

Este código utiliza Python no KNIME para aplicar hashing SHA-256 a dados sensíveis na coluna "password" de uma tabela de entrada. Primeiro, a tabela é convertida para um DataFrame, onde uma função personalizada converte cada valor de célula em uma string e gera um hash SHA-256, deixando valores nulos inalterados. A coluna "password" é então substituída pela versão hashed e a tabela processada é exportada para uma tabela de saída, protegendo os dados sensíveis de forma segura.

a. Validação de Emails

i. Expressão regular para email

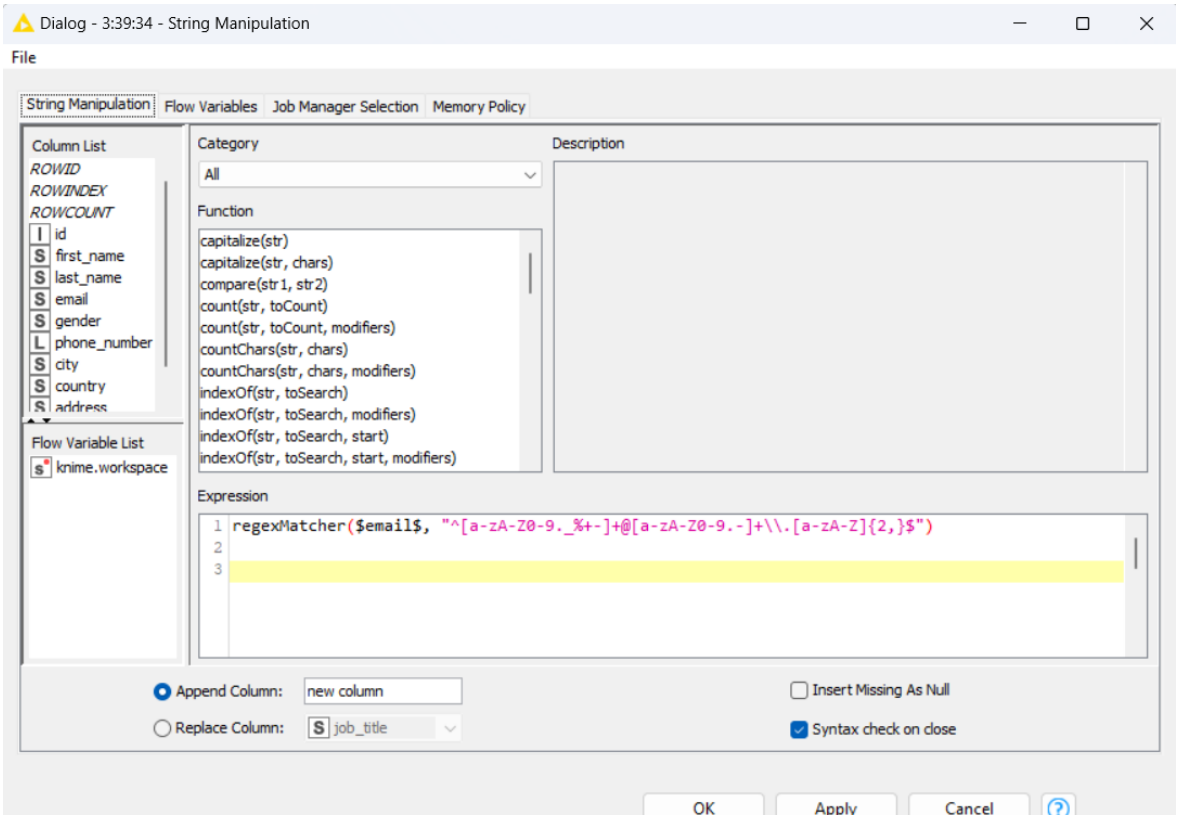

Figura 13 - Utilização de uma expressão regular para validar um email

Através do node String Manipulation é possível utilizar uma expressão regular para identificarmos o que é um email nos dados recebidos, para uma eventual validação desse mesmo campo na tabela.

ii. Regra de validação de Emails usando a ER

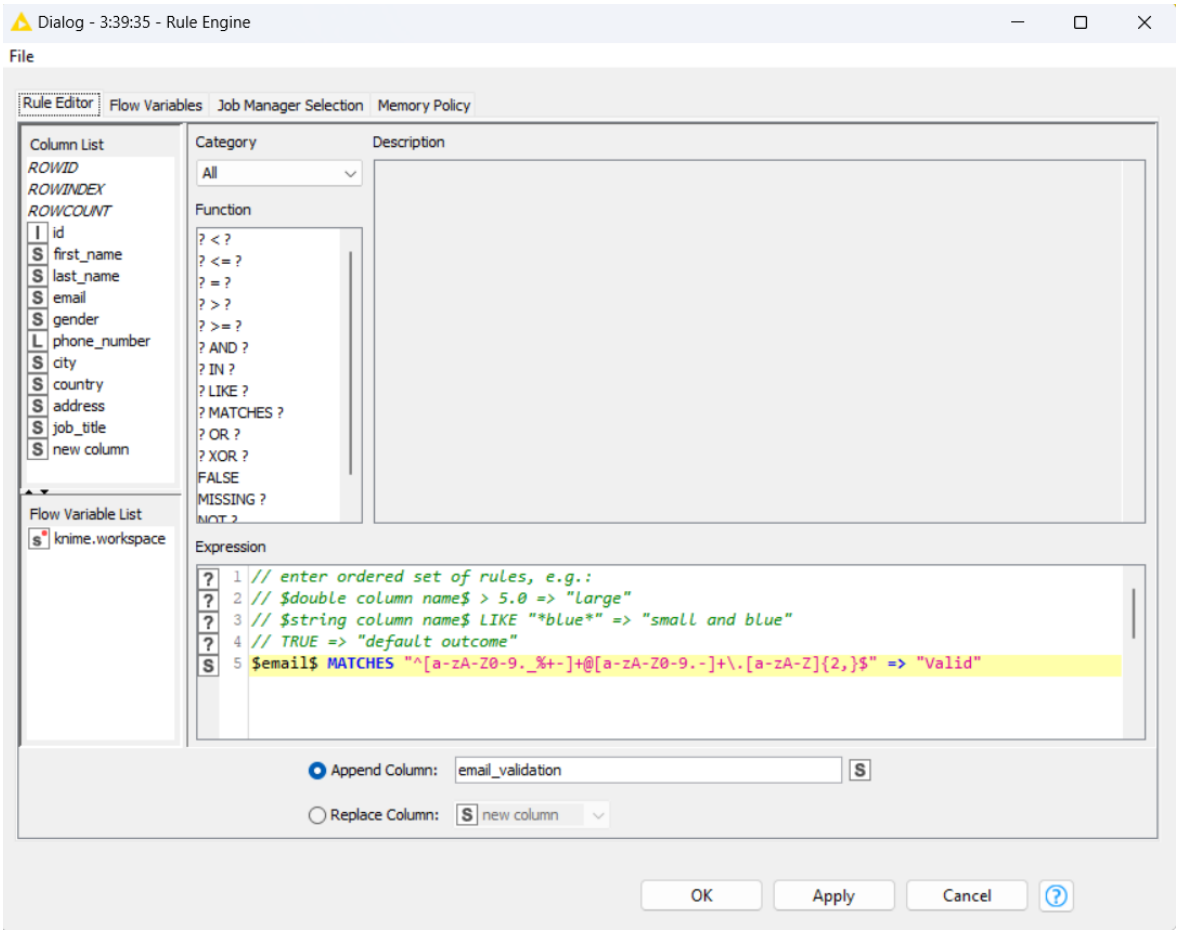

Figura 14 - Regra de validação de email nos dados

Criação da regra de validação caso um email esteja dentro da expressão regular preparada, caso seja válida é mantida nos dados a serem tratados.

iii. **Regra de Email usando a ER**

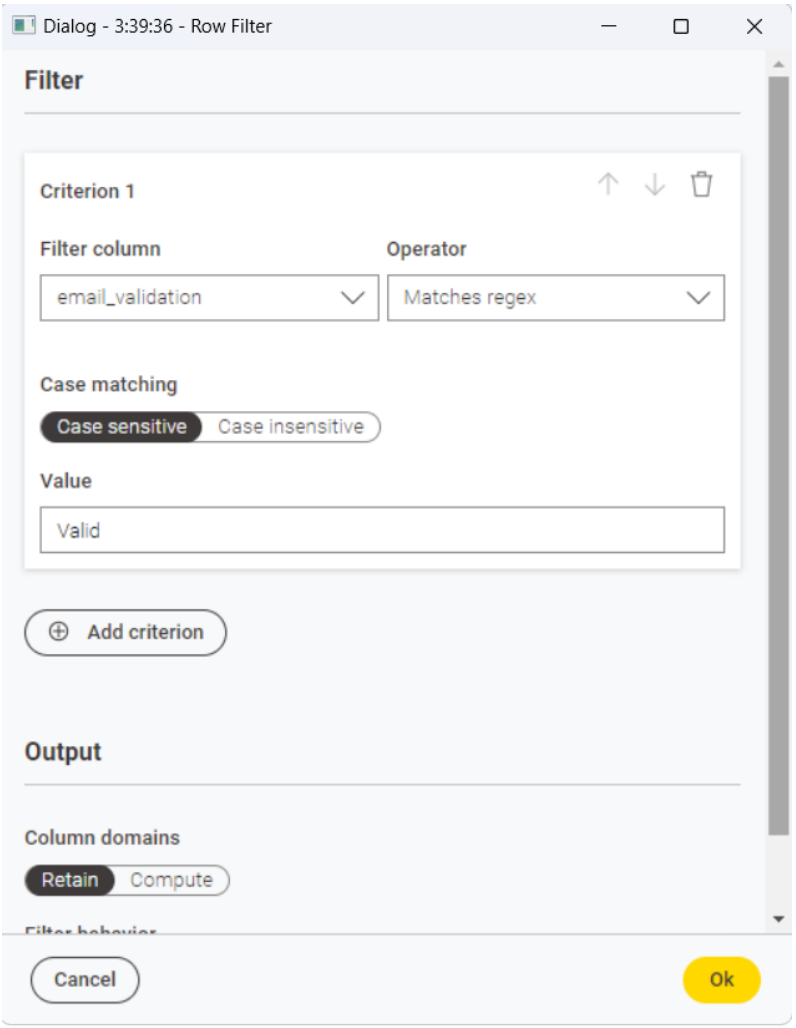

Figura 15 - Filtragem de emails válidos na tabela de dados

Através do node Row Filter podemos, após a validação dos emails pela regra no Rule Engine, acrescentar uma tabela de Emails que se encontram validados pela mesma.

b. Jobs

i. Base de Dados (API)

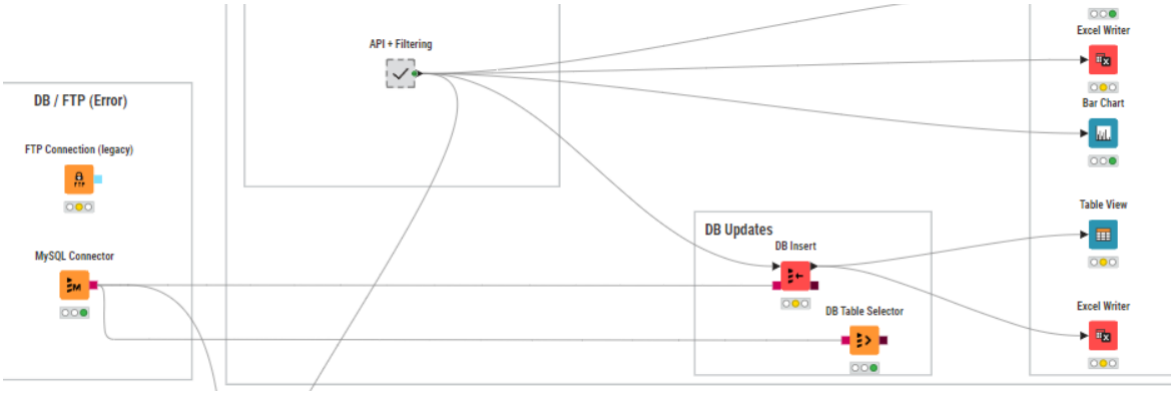

Figura 16 - Fluxo de BD + API + Exportação

ii. Base de Dados (API + Dados Codificados)

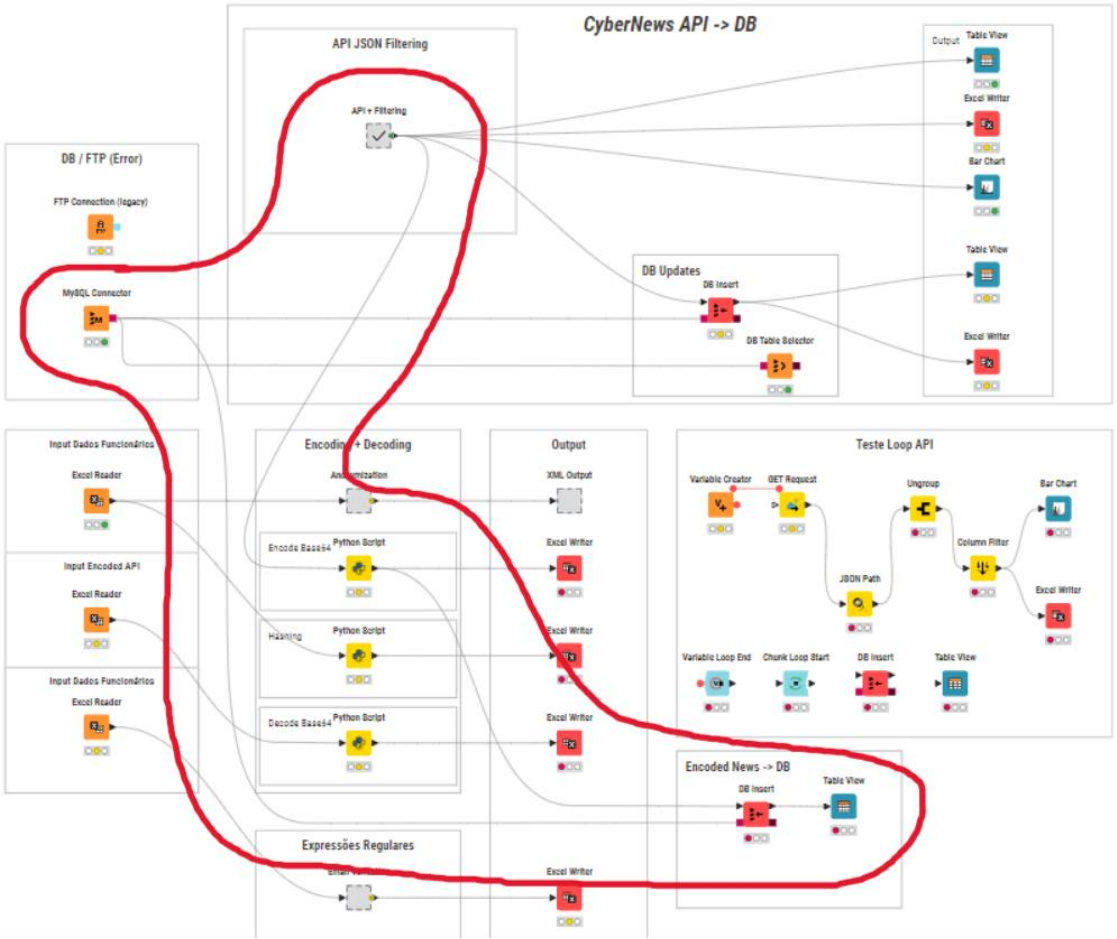

Figura 17 - Fluxo de API + BD + Codificação + Exportação/Visualização

iii. **Teste de Loop API**

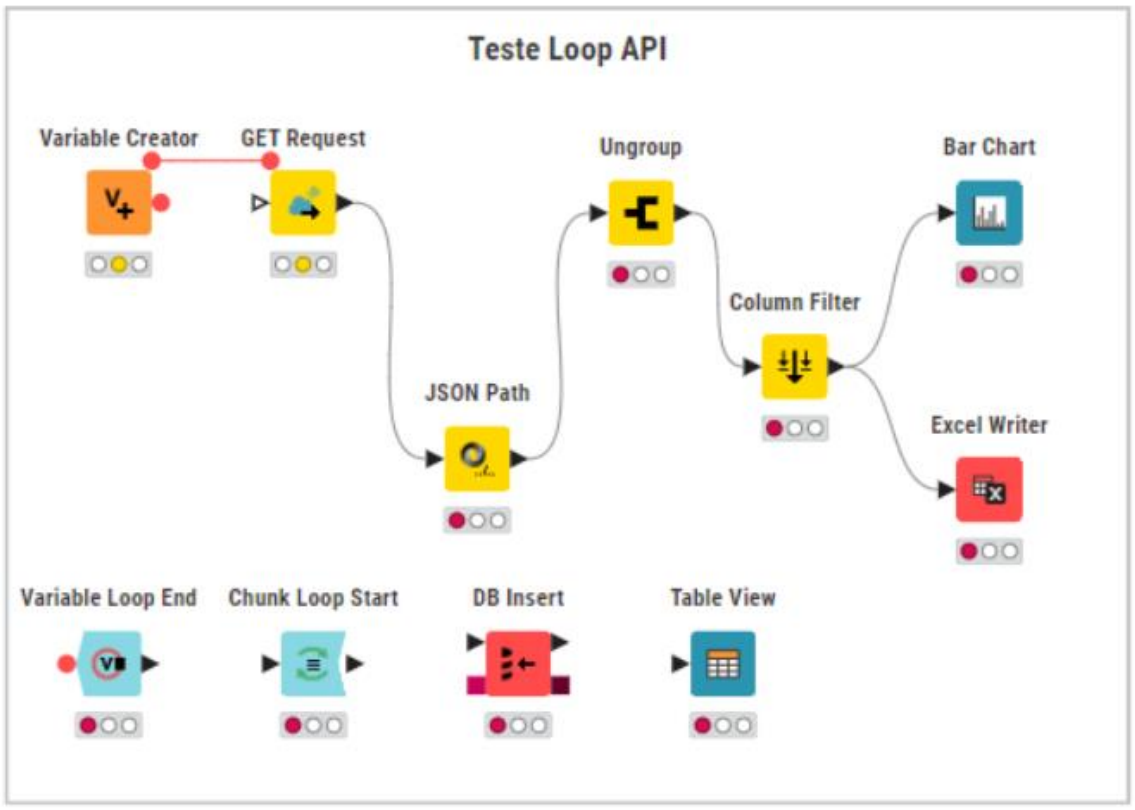

Figura 18 - Inicialização de Teste para uma constante procura de novas notícias nos websites através da API

4.1. Vídeo

4.2. Conclusão e Trabalhos Futuros

Este trabalho, apesar da cronologia um pouco apertada, tem grande valor para mim pois mesmo que eventualmente não trabalhemos com ferramentas ETL, sabemos que temos sempre algo que nos pode facilitar a automatização de certos processos em tratamento de dados.

Gostaria de ter tido mais tempo para tentar arranjar solução tanto para a encriptação em AES, talvez forçando algum ambiente externo para que o KNIME aceitasse as bibliotecas necessárias em Conda/Python, e também gostaria de ter finalizado o loop de gathering de dados através da API e dar append na tabela apenas a noticias não repetidas na mesma. Mas no geral foi um trabalho que nos dá bastantes valências, pela compreensão de como os processos podem ser tratados/analisados.

5. Bibliografia

Não existem origens no documento atual.