



Relatório Integração de Sistemas de Informação

No.12747 – João Carvalho

Licenciatura em Engenharia Sistemas Informáticos

3ºano

Barcelos | Outubro, 2024

Índice

1. Introdução	1
2. Desafio.....	2
2.1. Processos utilizados	3
3. Desenvolvimento	5
3.1. Transformação	5
3.2. Job	19
4. Vídeo	23
5. Conclusão	24
6. Anexos	25
7. Bibliografia	26

Índice de Figuras

Figura 1 - Definição das colunas de dados para os pedidos à API REST	5
Figura 2 - Definição dos parâmetros	5
Figura 3 - Pedido à API com o URL e Method definido na Data Grid	6
Figura 4 - Filtragem da resposta do pedido de identificador da API.....	7
Figura 5 - Atribuição do valor da tag à variável API_KEY	7
Figura 6 - Seleção de valores a utilizar nos passos seguintes	8
Figura 7 - Pedido à API para obtenção de dados de refugiados	9
Figura 8 - Parâmetros passados no pedido à API	10
Figura 9 - Filtragem dos dados da variável result	11
Figura 10 - Mapeamento dos dados para as variáveis correspondentes	11
Figura 11 - Seleção dos valores JSON da resposta	12
Figura 12 - Normalização das datas	13
Figura 13 - Transformação dos campos das datas	13
Figura 14 - Ordenação dos dados pela coluna Start	14
Figura 15 - Exportação XML dos dados	14
Figura 16 - Filtragem dos dados	15
Figura 17 - Exportação Excel	16
Figura 18 - Mapeamento das colunas	17
Figura 19 - Visão geral da transformação	18
Figura 20 - Validação existência do ficheiro exportado	19
Figura 21 - Configuração do email	20
Figura 22 - Assunto e Mensagem a enviar no email	21

Figura 23 - Definição de ficheiro a anexar no email.....	22
Figura 24 - Visão geral do Job.....	22

1. Introdução

Este projeto integra-se no âmbito da unidade curricular de Integração de Sistemas de Informação (ISI), lecionada no primeiro semestre do terceiro ano do curso de Licenciatura em Engenharia de Sistemas Informáticos. O trabalho proposto visa aprofundar o conhecimento e a aplicação de técnicas e ferramentas ETL (*Extract, Transformation, and Load*) para integração de sistemas de informação, com foco na manipulação e consolidação de dados provenientes de múltiplas fontes.

Para a realização do projeto, será utilizado o Pentaho Data Integration (Kettle), uma das principais ferramentas ETL *open-source*, que permite a transformação e o fluxo automatizado de dados entre diferentes sistemas. Esta escolha proporciona um ambiente de desenvolvimento visual, facilitando a criação de fluxos de trabalho complexos para transformar, normalizar e carregar dados, o que é fundamental para a obtenção de dados integrados e coerentes num sistema central.

Ao longo do projeto, serão abordados desafios comuns em cenários de integração, como a necessidade de extração de dados de diferentes fontes, transformação conforme regras de negócio específicas e carregamento eficiente em sistemas de destino. Este processo de experimentação e aplicação prática permitirá consolidar o conhecimento teórico em torno dos conceitos de ETL, explorando a interligação de sistemas numa infraestrutura centralizada de dados e simulando condições reais do ambiente profissional de engenharia de sistemas informáticos.

2. Desafio

Neste trabalho, a API *Humanitarian Data Exchange* (HAPI) foi utilizada como uma fonte externa de dados para integrar e enriquecer as informações no processo de ETL realizado no Pentaho Data Integration (Kettle). Através da API de refugiados do *Humanitarian Data Exchange*, obtivemos dados atualizados e detalhados sobre o número de refugiados que entraram em Portugal, juntamente com a distribuição das nacionalidades desses indivíduos.

O fluxo de trabalho foi construído de forma a fazer chamadas à API, utilizando funcionalidades para obter e carregar dados de entrada de refugiados em Portugal, categorizados por nacionalidade. Esta integração permitiu que os dados fossem transformados e preparados no Kettle para gerar métricas.

Além disso, a API trouxe uma camada adicional de automação e consistência ao processo, uma vez que possibilitou a atualização periódica dos dados sem a necessidade de importações manuais. Este método, combinado com as capacidades de transformação e manipulação de dados do Kettle, forneceu uma visão analítica importante para a análise das tendências migratórias de refugiados para Portugal, garantindo que o projeto reflita dados precisos e em tempo real.

Esta integração, portanto, proporcionou uma aplicação prática na recolha e análise de dados de sistemas externos, permitindo um maior entendimento da utilização de APIs no contexto da integração de sistemas de informação e enriquecendo o projeto com dados reais e dinâmicos.

2.1. Processos utilizados

Para a realização deste trabalho foram utilizados os seguintes processos:

- **Data Grid**
 - Permite criar e inserir dados manualmente numa transformação, sendo útil para testes, tabelas de referência e dados temporários.
- **Rest Client**
 - Permite fazer pedidos a APIs REST externas para obter, enviar ou atualizar dados, integrando essas informações ao fluxo de transformação para serem processadas ou armazenadas.
- **JSON Input**
 - Usado para ler e processar dados em formato JSON, extraíndo informações específicas para serem manipuladas dentro do fluxo de transformação, como se fosse uma fonte de dados.
- **Select Values**
 - Permite selecionar, renomear, reordenar, remover e alterar o tipo de dados das colunas numa transformação, facilitando a gestão e a organização dos dados antes de etapas posteriores no fluxo.
- **Modified JavaScript Value**
 - Permite manipular e transformar dados através de código JavaScript personalizado, facilitando operações complexas e condicionais durante o processo de ETL.
- **Sort Rows**
 - Utilizado para ordenar dados numa tabela ou fluxo, facilitando o pré-processamento, a análise e a preparação para operações subsequentes.
- **Filter Rows**
 - Usado para selecionar ou descartar linhas de dados com base em condições específicas, permitindo refinar conjuntos de dados antes de outras operações no processo de ETL.

- **Microsoft Excel Writer**
 - Utilizado para exportar dados processados para arquivos no formato Excel (.xls ou .xlsx), permitindo a criação de relatórios e a exportação de dados de forma estruturada e fácil de manipular.
- **XML Output**
 - Utilizado para exportar dados processados para arquivos no formato XML, permitindo a criação de documentos estruturados que podem ser utilizados para integração com outros sistemas ou armazenamento de dados.
- **File Exists**
 - Utilizado para verificar a existência de um ficheiro ou diretório específico no sistema de ficheiros, permitindo controlar o fluxo de execução do processo de ETL com base na presença ou ausência desse ficheiro.
- **Mail**
 - Utilizado para enviar e-mails a partir do fluxo de ETL, permitindo a notificação de eventos, relatórios ou alertas com dados processados diretamente do Kettle.

3. Desenvolvimento

3.1. Transformação

Para este trabalho foi utilizado uma transformação de uma API de dados externa. O primeiro passo foi a criação de uma Data Grid de apenas uma linha com os diferentes parâmetros para executar os pedidos à API.

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Null if	Set empty string?
1	URL	String								N
2	Method	String								N
3	Limit	Integer								N
4	Country	String								N

Figura 1 - Definição das colunas de dados para os pedidos à API REST

#	URL	Method	Limit	Country
1	https://hapi.humdata.org/api/v1/encode_app_identifier?application=Test&email=test%40test.com	GET	10000	PRT

Figura 2 - Definição dos parâmetros

De seguida, foi utilizado o processo Rest Client para fazer o pedido à API para obter o identificador a usar nos diferentes endpoints, no *endpoint* definido nos parâmetros na Figura 2.

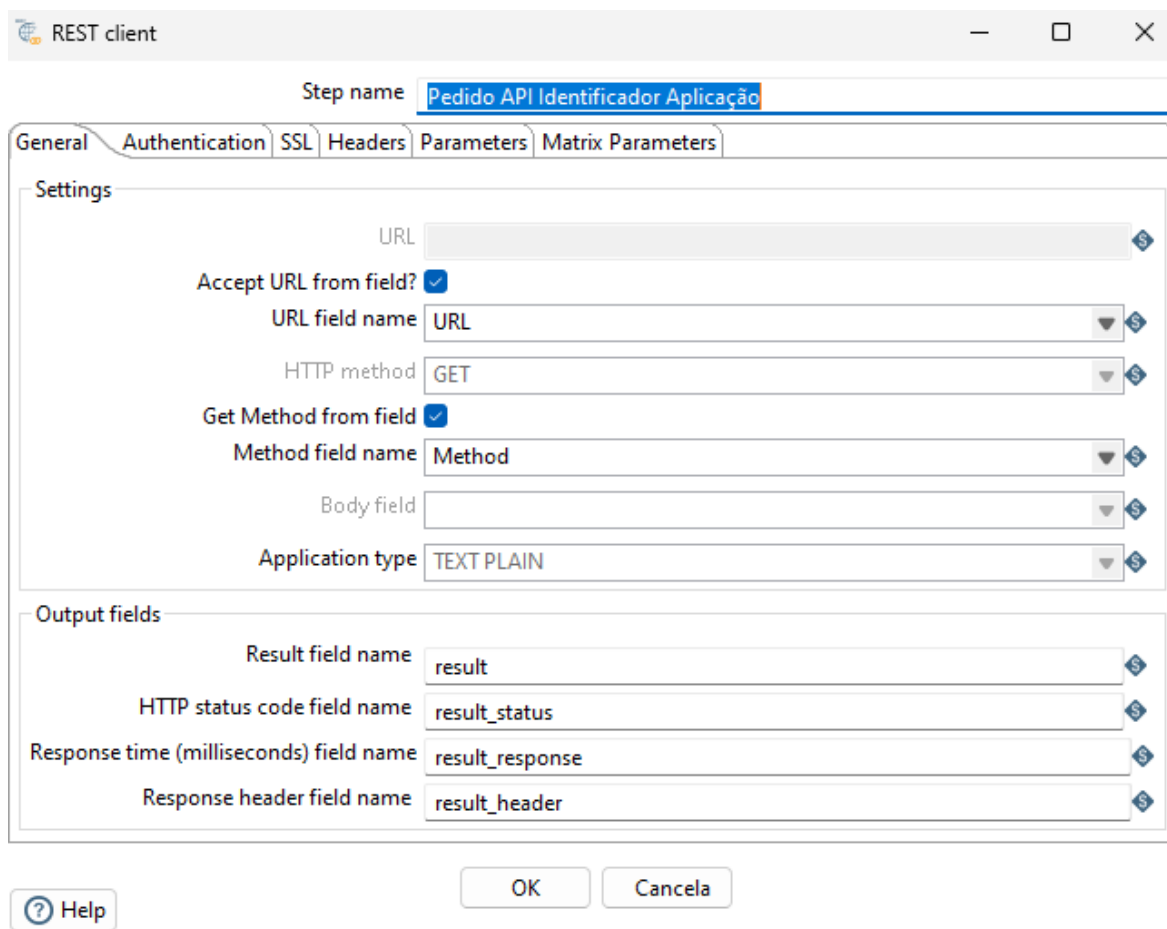


Figura 3 - Pedido à API com o URL e Method definido na Data Grid

O URL e o método do pedido à API, bem como os consequentes parâmetros usados nos pedidos à API tiveram de ser inseridos na Data Grid devido a uma limitação de *software* do Kettle que não permite passar parâmetros na aba 'Parameters' se o tipo de pedido for *GET*.

Após o pedido, foi utilizado o JSON Input para filtrar a resposta devolvida pela API ao pedido, utilizando para isso a variável *result* definida no passo anterior como nome de variável para o resultado, sendo depois esse valor da resposta atribuída à variável *API_KEY*.

Nome do Step: **Filtrar resposta API**

File | Content | Fields | Additional output fields

Source from field

Source is from a previous step: ☒

Select field: **result**

Use field as file names: ☐

Read source as URL: ☐

Do not pass field downstream: ☐

File or directory:

Regular Expression:

Exclude Regular Expression:

Selected files:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders
1				N	N

Show filename(s)...

Buttons: Help, OK, Preview rows, Cancela

Figura 4 - Filtragem da resposta do pedido de identificador da API

Nome do Step: **Filtrar resposta API**

File | Content | Fields | Additional output fields

#	Name	Path	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	API_KEY	encoded_app_identifier	String							none

Select fields

Buttons: Help, OK, Preview rows, Cancela

Figura 5 - Atribuição do valor da tag à variável API_KEY

The screenshot shows the 'REST client' window with the 'Step name' set to 'Pedido API Dados Refugiados'. The 'General' tab is selected, displaying the 'Settings' section. The 'URL' is 'https://hapi.humdata.org/api/v1/affected-people/refugees'. The 'HTTP method' is 'POST'. The 'Get Method from field' checkbox is checked, and the 'Method field name' is 'Method'. The 'Body field' is empty. The 'Application type' is 'TEXT PLAIN'. The 'Output fields' section shows 'Result field name' as 'result', 'HTTP status code field name' as 'result_status', 'Response time (milliseconds) field name' as 'result_response', and 'Response header field name' as 'result_header'. At the bottom, there are 'OK' and 'Cancela' buttons, and a 'Help' button with a question mark icon.

REST client

Step name Pedido API Dados Refugiados

General Authentication SSL Headers Parameters Matrix Parameters

Settings

URL https://hapi.humdata.org/api/v1/affected-people/refugees

Accept URL from field? ☐

URL field name URL

HTTP method POST

Get Method from field ☒

Method field name Method

Body field

Application type TEXT PLAIN

Output fields

Result field name result

HTTP status code field name result_status

Response time (milliseconds) field name result_response

Response header field name result_header

Help OK Cancela

Figura 7 - Pedido à API para obtenção de dados de refugiados

Mais uma vez, foi necessário definir o método do pedido através da variável Method, para permitir que a aba 'Parameters' estivesse ativa para passar os restantes parâmetros no pedido à API.

Tal como no passo seguinte do pedido para obter a `API_KEY`, utilizei novamente o `JSON Input` para filtrar o resultado do pedido à API, indicando novas colunas e quais os paths para a obtenção dos dados no `JSON` da resposta da API, novamente obtendo os dados da variável `result`, definida no pedido à API.

JSON input

Nome do Step: Filtrar Pedido API Dados Refugiados

File Content Fields Additional output fields

Source from field

Source is from a previous step: ☒

Select field: result

Use field as file names: ☐

Read source as URL: ☐

Do not pass field downstream: ☐

File or directory: Add Browse

Regular Expression: Add

Exclude Regular Expression: Add

Selected files:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders
1				N	N

Show filename(s)...

Help OK Preview rows Cancela

Figura 9 - Filtragem dos dados da variável result

JSON input

Nome do Step: Filtrar Pedido API Dados Refugiados

File Content Fields Additional output fields

#	Name	Path	Type	Format	Length	Precision	Currency	Decimal	Group
1	ID	\$.data[*].resource_hdx_id	String						
2	ORIGIN_LOC_REF	\$.data[*].origin_location_ref	Integer						
3	ASY_LOC_REF	\$.data[*].asylum_location_ref	Integer						
4	POPU_GRP	\$.data[*].population_group	String						
5	GENDER	\$.data[*].gender	String						
6	AGE_RANGE	\$.data[*].age_range	String						
7	MIN_AGE	\$.data[*].min_age	Integer						
8	MAX_AGE	\$.data[*].max_age	Integer						
9	NUM_POP	\$.data[*].population	Integer						
10	START	\$.data[*].reference_period_start	String						
11	END	\$.data[*].reference_period_end	String						
12	ORIGIN_LOC_ID	\$.data[*].origin_location_code	String						
13	ORIGIN_LOC_NAME	\$.data[*].origin_location_name	String						
14	ASY_LOC_CODE	\$.data[*].asylum_location_code	String						
15	ASY_LOC_NAME	\$.data[*].asylum_location_name	String						

Select fields

Help OK Preview rows Cancela

Figura 10 - Mapeamento dos dados para as variáveis correspondentes

No passo seguinte utilizei novamente o Select Values para obter as variáveis dos dados resultantes do JSON.

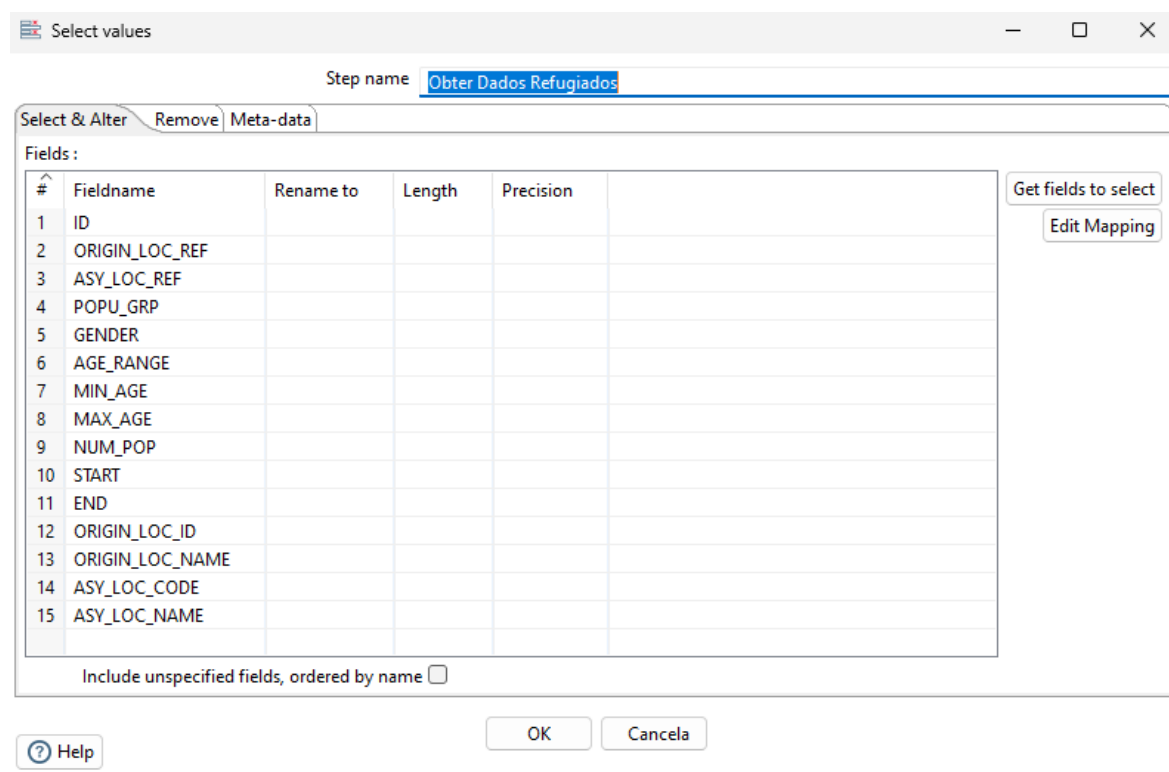


Figura 11 - Seleção dos valores JSON da resposta

Depois de obtidos os dados, procedi à normalização dos campos das datas, utilizando para isso o processo Modified Javascript Value, para substituir parte da *string* das datas.

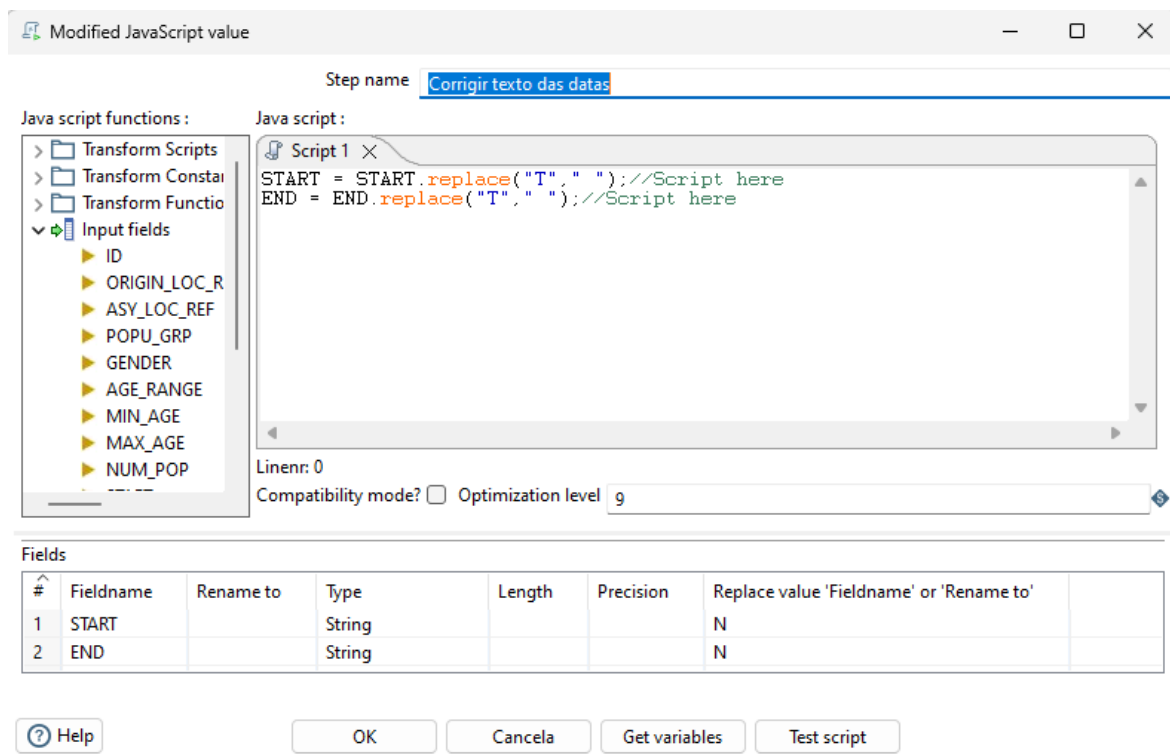


Figura 12 - Normalização das datas

Utilizei novamente o Select Value após este passo, para transformar os campos de datas para o tipo DATE na aba 'Meta-Data' com o respetivo formato.

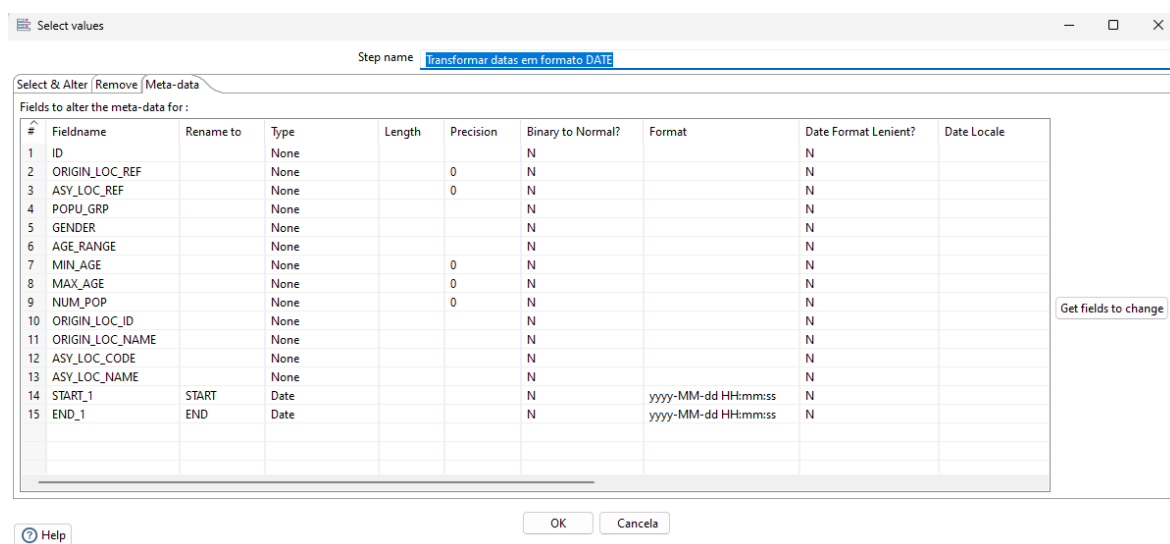


Figura 13 - Transformação dos campos das datas

Após a normalização dos campos das datas, procedi à ordenação dos dados pela coluna START.

Nome do Step: Ordenação por Data de Início

Sort directory: %%java.io.tmpdir%%

TMP-file prefix: out

Sort size (rows in memory): 1000000

Free memory threshold (in %):

Compress TMP Files? ☒

Only pass unique rows? (verifies keys only) ☐

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

Buttons: Help, OK, Cancela, Obtem campos

Figura 14 - Ordenação dos dados pela coluna Start

Após este passo, é gerado um ficheiro XML com todos os dados presentes e ordenados, passível de ser usado por um outro sistema caso seja necessário.

Nome do Step: Exportação XML

Filename: \$(Internal.Entry.Current.Directory)/Refugiados

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:

Show filename(s)...

Add filenames to result ☐

Buttons: Help, OK, Cancela

Figura 15 - Exportação XML dos dados

Apliquei de seguida uma filtragem aos dados presentes para demonstrar que é possível aprimorar os dados para obtermos apenas os dados necessários nas condições que desejámos.

Filter rows

Step name:

Send 'true' data to step:

Send 'false' data to step:

The condition:

☐ +

NOT (GENDER = [f])

AND

POPU_GRP = [REF]

AND

AGE_RANGE = [all]

AND

NUM_POP <> [0]

Figura 16 - Filtragem dos dados

Após a filtragem, seguiu-se a exportação dos dados para um ficheiro de formato Excel, para uma melhor compreensão e gestão por parte dos utilizadores que desejem ter estes dados.

Microsoft Excel writer

Nome do Step: Exportação Excel

File & Sheet Content

File

Filename: Navega...

Extensão:

Create parent folder ☐

Stream XLSX data ☐

Split every ... data rows:

Include stepnr in filename? ☐

Include date in filename? ☐

Include time in filename? ☐

Specify Date time format ☐

Date time format:

Show filename(s)...

If output file exists:

Wait for first row before creating file ☐

Add filenames to result ☒

Sheet

Sheet name (max. 31 characters):

Make this the active sheet ☒

If sheet exists in output file:

Protect sheet? (XLS format only) ☐

Protected by user:

Password:

Template

Use template when creating new files ☐

Template file: Navega...

Use template when creating new sheets ☐

Template sheet:

Hide Template Sheet ☐

Help OK Cancela

Figura 17 - Exportação Excel

Microsoft Excel writer

Nome do Step: Exportação Excel

File & Sheet Content

Content options

Start writing at cell: A1

When writing rows: overwrite existing cells

Write Header ☒

Write Footer ☐

Auto size columns ☐

Retain NULL values ☒

Force formula recalculation ☐

Leave styles of existing cells unchanged ☐

Extend data validation range ☐

When writing to existing sheet

Start writing at end of sheet (appending lines) ☐

Offset by ... rows: 0

Begin by writing ... empty lines: 0

Omit header ☐

Fields

#	Name	Type	Format	Style from cell	Fiel
1	ORIGIN_LOC_REF	Integer			ORI
2	ASY_LOC_REF	Integer			ASY
3	POPU_GRP	String			POF
4	GENDER	String			GEN
5	AGE_RANGE	String			AGE
6	MIN_AGE	Integer			MIN
7	MAX_AGE	Integer			MA
8	NUM_POP	Integer			NUI
9	ORIGIN_LOC_ID	String			ORI
1.	ORIGIN_LOC_NAME	String			ORI
1.	ASY_LOC_CODE	String			ASY
1.	ASY_LOC_NAME	String			ASY
1.	START	Date	m/d/yy h:mm		STA
1.	END	Date	m/d/yy h:mm		ENE

Obtem campos Minimal width

Help OK Cancela

Figura 18 - Mapeamento das colunas

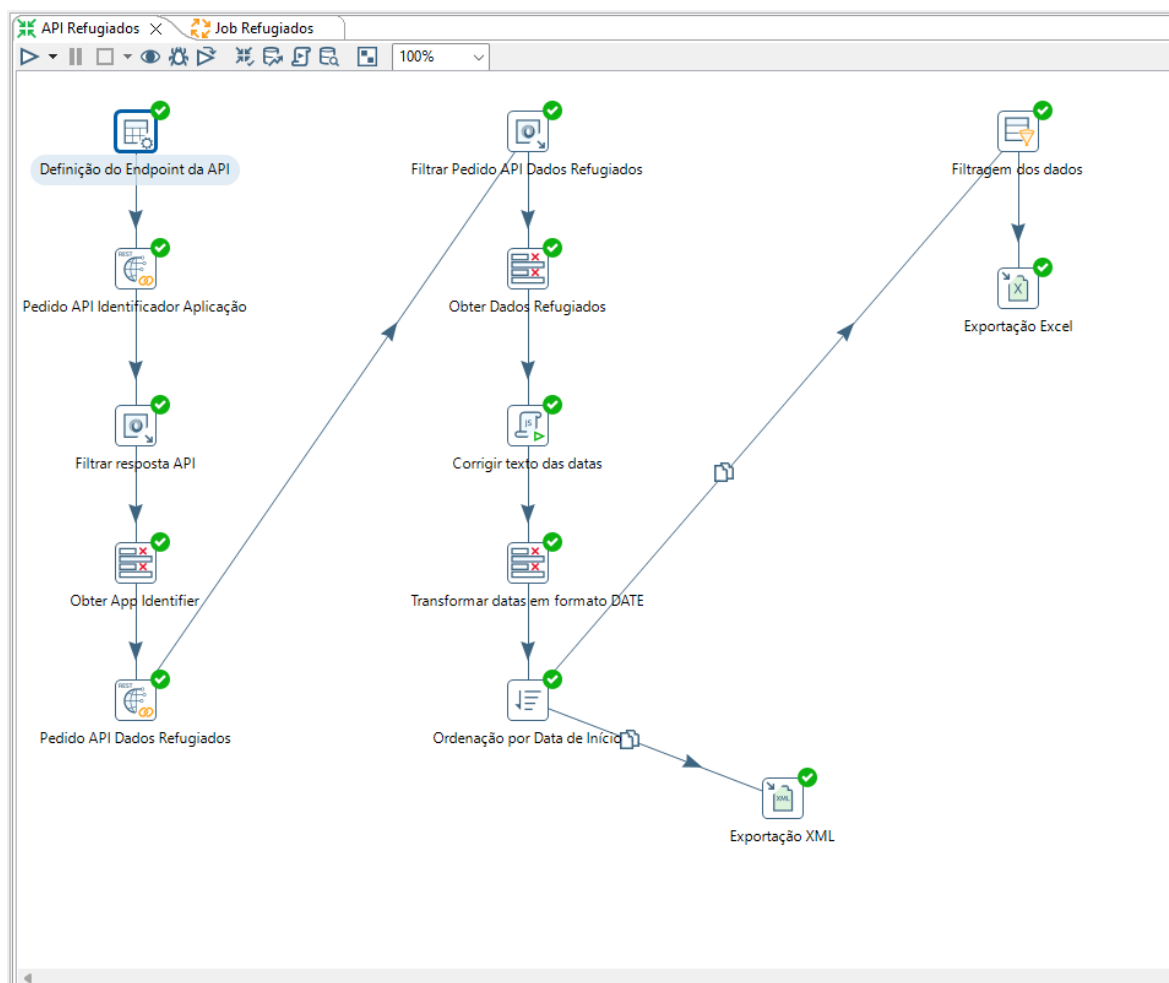


Figura 19 - Visão geral da transformação

3.2. Job

Foi criado um Job para permitir o envio do ficheiro Excel exportado na transformação para um email definido. Para isso foi incluída a execução da transformação dentro do Job, seguido de uma validação da existência do ficheiro exportado.

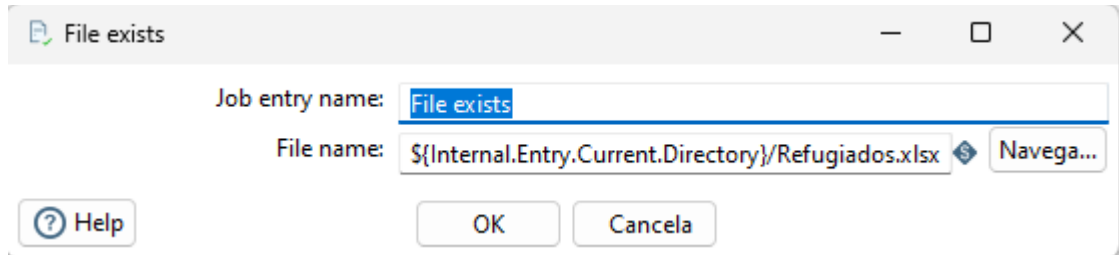


Figura 20 - Validação existência do ficheiro exportado

Caso esta validação seja positiva, ou seja, o ficheiro existe, é enviado então o email com as configurações definidas.

Mail

Name of mail job entry: Mail

Addresses Server EMail Message Attached Files

SMTP Server

SMTP Server: smtp.office365.com

Port: 587

Authentication

Use authentication? ☒

Authentication user: a12747@alunos.ipca.pt

Authentication password:

Use secure authentication? ☒

Secure connection type: TLS

Help OK Cancela

Figura 21 - Configuração do email

Mail

Name of mail job entry: Mail

Addresses Server **EMail Message** Attached Files

Message Settings

Include date in message? ☒

Only send comment in mail body? ☒

Use HTML format in mail body? ☒

Encoding UTF-8

Manage priority ☐

Priority Normal

Importance Normal

Sensitivity Normal

Message

Subject: Exportação Dados Refugiados

Comment: Segue em anexo o ficheiro exportado.

Help OK Cancela

Figura 22 - Assunto e Mensagem a enviar no email

Mail

Name of mail job entry: 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:

Embedded images

Filename Add Browse files ...

Content ID

Embedded images

#	Image	Content ID
1	C:\Users\JP\Desktop\Refugiados.xlsx	4p6aq6r8eecoo

Delete Edit

Help OK Cancela

Figura 23 - Definição de ficheiro a anexar no email

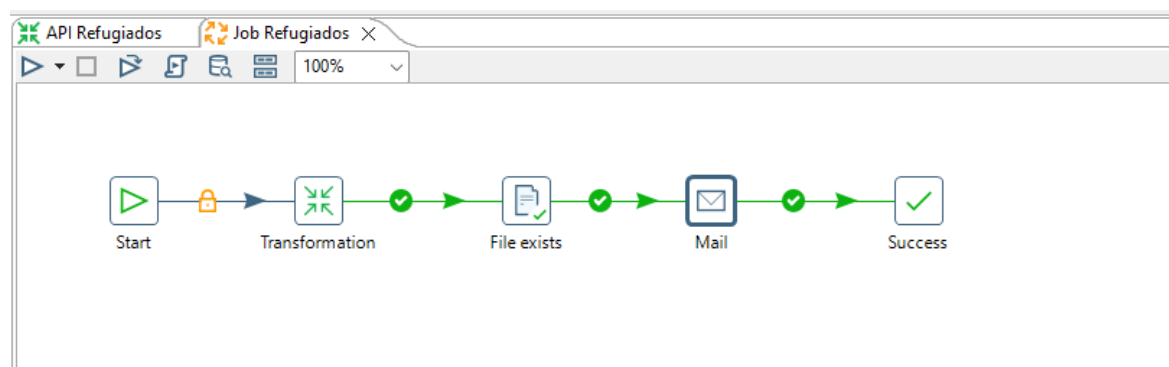


Figura 24 - Visão geral do Job

4. Vídeo

Deixo abaixo o código QR gerado com o link para o vídeo da execução do processo de transformação dos dados.



5. Conclusão

Este trabalho demonstrou que as ferramentas de ETL (Extract, Transform, Load) oferecem uma vasta gama de recursos para a integração, transformação e análise de dados. A experiência adquirida ao explorar essas ferramentas foi bastante enriquecedora, permitindo compreender diversas funcionalidades que se destacam pela simplicidade e versatilidade.

Além disso, a capacidade de se integrar com outras ferramentas, como bases de dados e APIs, e de extrair dados de múltiplas fontes, transformando-os conforme as necessidades específicas e carregando-os nos destinos desejados, representa um grande diferencial.

Os conhecimentos adquiridos nesta disciplina não apenas ampliam a compreensão dos processos ETL, mas também podem ser aplicados diretamente no contexto profissional, facilitando a resolução de desafios relacionados ao tratamento de dados.

Em resumo, as ferramentas de ETL são uma excelente escolha para análise e tratamento de dados, oferecendo vantagens significativas para seus usuários e contribuindo para o desenvolvimento de habilidades práticas no mercado de trabalho.

6. Anexos

- [Exemplo Exportação Excel](#)
- [Exemplo Exportação XML](#)

7. Bibliografia

- **API**
 - https://hdx-hapi.readthedocs.io/en/latest/data_usage_guides/affected_people/#refugees-persons-of-concern
 - <https://hapi.humdata.org/docs#/>
- **Kettle**
 - <https://docs.hitachivantara.com/r/en-us/pentaho-data-integration-and-analytics/10.1.x/mk-95pdia000/getting-started-with-pdi/pentaho-data-integration-pdi-tutorial/step-1-extract-and-load-data>