

Relatório de Integração de Sistemas de Informação

Licenciatura em Engenharia de Sistemas Informáticos

# Tratamento de dados e processos ETL

---

Hélder Martins Carvalho – 15310

João Paulo Figueiredo Carvalho – 15314

Barcelos, <Mês> de <Ano>

Afirmo por minha honra que não recebi qualquer apoio não autorizado na realização deste trabalho prático. Afirmo igualmente que não copiei qualquer material de livro, artigo, documento web ou de qualquer outra fonte exceto onde a origem estiver expressamente citada.

Hélder Martins Carvalho – 15310

João Paulo Figueiredo Carvalho – 15314

## Índice

<b>Introdução</b>	<b>5</b>
<i>Problema a resolver</i>	5
<i>Plataformas</i>	5
<i>Objetivos</i>	5
Objetivos do projeto	5
Objetivos do problema a resolver	6
<b>Arquitetura da solução</b>	<b>7</b>
<i>Pentaho Data Integration – Kettle</i>	7
Esquema da solução	7
Funcionamento da solução	9
<i>Knime</i>	17
Esquema da solução	17
Funcionamento da solução	19
<b>Conclusão</b>	<b>31</b>
<b>Bibliografia</b>	<b>33</b>

## Lista de Figuras

Figura 1 – "Job" principal	7
Figura 2 – "Job" de limpeza	7
Figura 3 – Transformação de Leitura/preparação dos dados	8
Figura 4 – Transformação de Exportação total	8
Figura 5 – Transformação de Exportação agrupada por letra inicial	8
Figura 6 – Tabela da base de dados	8
Figura 7 – Passo "XML input"	9
Figura 8 – Previsualização dos dados de saída do passo "XML input"	10
Figura 9 – Passo "Filtro de linhas"	10
Integração de Sistemas de Informação	1

Figura 10 – Previsualização dos dados de saída do passo “Filtro de linhas”	11
Figura 11 – Passo “Definidor de palavra”	11
Figura 12 – Previsualização dos dados de saída do passo “Definidor de palavra”	12
Figura 13 – Passo “Agrupador”	12
Figura 14 – Previsualização dos dados de saída do passo “Agrupador”	12
Figura 15 – Passo “Retirar <i>heads</i> e <i>NULLS</i> ”	13
Figura 16 – Previsualização dos dados de saída do passo “Retirar <i>heads</i> e <i>NULLS</i> ”	13
Figura 17 – Passo “Remover colunas desnecessárias”	13
Figura 18 – Previsualização dos dados de saída do passo “Remover colunas desnecessárias”	14
Figura 19 – Passo “ <i>CSV output</i> ” (total)	14
Figura 20 – Passo “ <i>JSON output</i> ”	15
Figura 21 – Passo “ <i>XML output</i> ”	15
Figura 22 – Passo “ <i>BD output</i> ”	16
Figura 23 – Passo “ <i>CSV output</i> ” (por letra)	16
Figura 24 – “ <i>Workflow</i> ” Principal	17
Figura 25 – “ <i>Workflow</i> ” <i>DicXML</i>	17
Figura 26 – Componente “ <i>JsonAll</i> ”	18
Figura 27 – Componente “ <i>CsvAll</i> ”	18
Figura 28 – Componente “ <i>DbAll</i> ”	18
Figura 29 – Componente “ <i>LetraALetra</i> ”	19
Figura 30 – Configuração do “ <i>XML Reader</i> ”	19
Figura 31 – Dados de saída do “ <i>XML Reader</i> ”	20
Figura 32 – Configuração do “ <i>XPath</i> ” para Filtrar dados	20
Figura 33 – Dados de saída do “ <i>XPath</i> ” para Filtrar dados	20
Figura 34 – Remoção de dados incompletos	21
Figura 35 – Remoção das mudanças de linha	21
Figura 36 – “ <i>Java Snippet</i> ” para obter a letra inicial de uma palavra	22

Figura 37 - Dados saída do “ <i>Java Snippet</i> ”	23
Figura 38 – Carregamento dos dados tratados no “ <i>Workflow</i> ” principal	23
Figura 39 – Saída dos dados do “ <i>Workflow</i> ” de tratamento de dados	23
Figura 40 – Configuração do “ <i>Table to JSON</i> ”	24
Figura 41 – Dados de saída do “ <i>Table to JSON</i> ”	24
Figura 42 – Criação de uma nova coluna com o valor do nome do ficheiro exportado	25
Figura 43 – Alteração do valor do “ <i>RowId</i> ”	25
Figura 44 – Dados de saída do “ <i>RowId</i> ”	25
Figura 45 – Exportação dos dados utilizando “ <i>JSON Writer</i> ”	25
Figura 46 – Configuração do “ <i>CSV Writer</i> ”	26
Figura 47 – Definição do caracter delimitador entre dados em CSV	26
Figura 48 – Configuração do “ <i>MySQL Connector</i> ”	26
Figura 49 – Configuração do “ <i>DB Insert</i> ”	27
Figura 50 – Configuração do “ <i>File Reader</i> ”	27
Figura 51 – Documento de texto que contém quais as letras a serem exportadas	27
Figura 52 – Configuração do “ <i>Row Filter</i> ”	28
Figura 53 – Configuração de valores dinâmicos nos campos	28
Figura 54 – Criação de um <i>Filename</i> dinâmico usando <i>Java</i>	29
Figura 55 – Localização dinâmica da exportação das palavras letra a letra em CSV	29
Figura 56 – Criação do nome dinâmico em <i>JSON</i>	30
Figura 57 – Resultado da exportação letra a letra	30



## Introdução

Este projeto foi concretizado no âmbito da disciplina de Integração de Sistemas de Informação da Licenciatura de Engenharia de Sistemas Informáticos. Nele pretende-se consolidar, melhorar e adquirir conhecimentos relativos aos processos de *ETL*<sup>1</sup> para a análise, processamento e importação/exportação de dados de/para diferentes sistemas.

## Problema a resolver

Tendo em conta que o principal objetivo deste projeto é melhorar os nossos conhecimentos na área de ETL, decidimos enveredar num tema mais lúdico.

Assim sendo, pegamos num ficheiro *XML*<sup>2</sup> que representa um dicionário com mais de 120 mil palavras de teor mais “local/rural” e decidimos trabalhar sobre ele.

## Plataformas

Para a realização deste trabalho, de entre várias disponíveis, foram utilizadas as ferramentas “*Pentaho Data Integration – Kettle*” e “*Knime*” que apresentam uma vasta coleção de processos e funcionalidades aplicáveis ao tema *ETL*.

O elemento Hélder Carvalho encarregou-se de desenvolver em “*Pentaho Data Integration – Kettle*” e o elemento João Carvalho em “*Knime*”.

## Objetivos

### Objetivos do projeto

Relativamente ao projeto, os objetivos são:

- Utilização de Expressões Regulares (ER) em processos de tratamento de dados: normalização, limpeza, etc.;
- Lidar com importação/exportação de dados para *XML* e *JSON*<sup>3</sup>;

---

<sup>1</sup> *Extract, Transform and Load*

<sup>2</sup> *Extensible Markup Language*

<sup>3</sup> *JavaScript Object Notation*

- Operações sobre Bases de Dados;

## Objetivos do problema a resolver

Relativamente ao problema a resolver, os objetivos são:

- Importação de dados em *XML*;
- Tratamento, filtragem e agrupamento de dados;
- Exportação dos dados tratados:
  - Exportação total em *CSV*<sup>4</sup>;
  - Exportação total em *JSON*;
  - Exportação total em *XML*;
  - Exportação total para base de dados;
  - Exportação agrupada por letra inicial da palavra em *CSV*;

A informação desnecessária deve ser descartada.

---

<sup>4</sup> *Comma-separated values*



## Arquitetura da solução

### Pentaho Data Integration – Kettle

#### Esquema da solução

Esta solução apresenta 2 “Jobs” e 3 “Transformations”.

O “Job” mais importante é o “Principal” que quando é executado elimina os ficheiros da execução anterior, cria a pasta onde vão ser criados novos, cria a base de dados se necessário e executa as transformações.

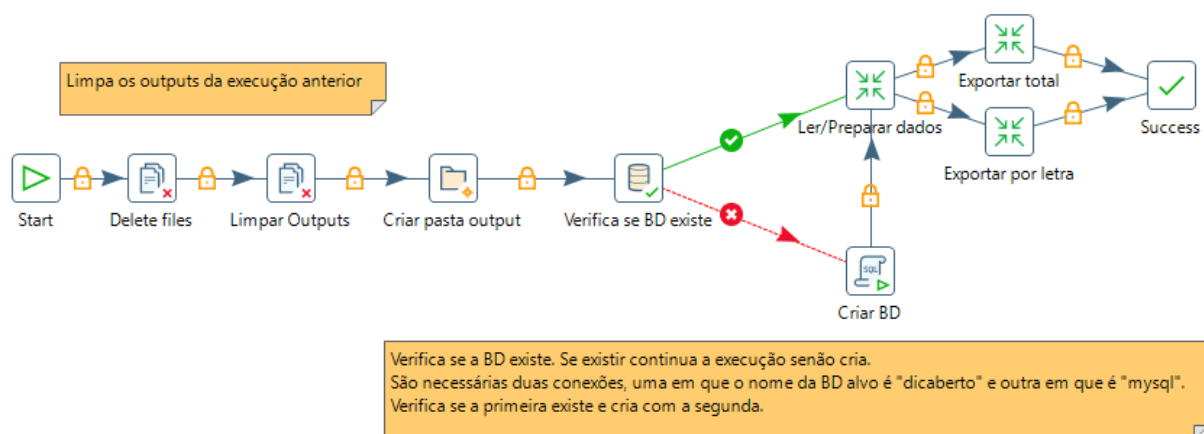


Figura 1 – “Job” principal

O segundo “Job” realiza apenas uma limpeza dos ficheiros e bases de dados utilizados na execução do “Job” anterior (Figura 1).

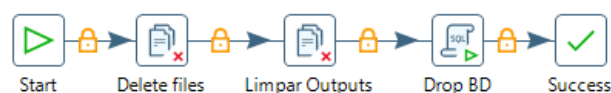


Figura 2 – “Job” de limpeza

Relativamente às transformações, a primeira a ser executada é a de leitura/preparação dos dados sendo que as outras duas são de exportação dos mesmos.

Para a leitura/preparação dos dados foi composta a seguinte transformação:

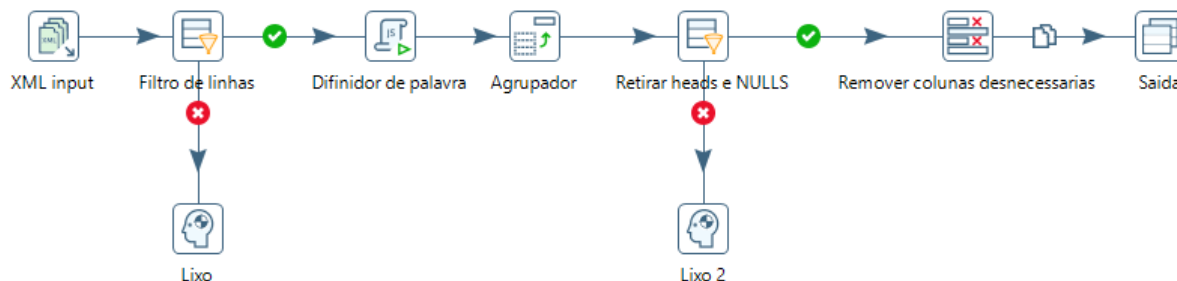


Figura 3 – Transformação de Leitura/preparação dos dados

Para a exportação total dos dados foi composta a seguinte transformação:

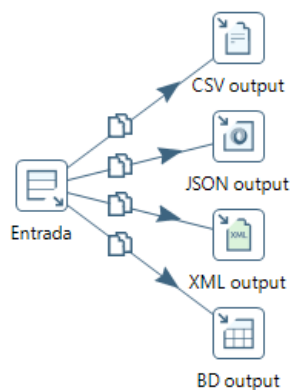


Figura 4 – Transformação de Exportação total

Para a exportação agrupada por letra inicial foi composta a seguinte transformação:



Figura 5 – Transformação de Exportação agrupada por letra inicial

Na a exportação para base de dados, foi estruturada a seguinte tabela:

dicaberto	palavras
id	: int(11)
orth	: varchar(535)
gramGrp	: varchar(30)
def	: text

Figura 6 – Tabela da base de dados

## Funcionamento da solução

No processo de “Leitura/preparação dos dados” (Figura 3) foi utilizado um ficheiro *XML* como entrada de dados.

Este ficheiro é lido no primeiro passo (“*XML input*”) da transformação. A funcionalidade utilizada é a “*XML input stream (StAX)*” que tem como objetivo ler ficheiros *XML* de grandes dimensões rapidamente. Para isso, cada linha do ficheiro é lida de forma individual.

XML input stream (StAX)

Step name: XML input

Filename: \${Internal.Entry.Current.Directory}/dicAberto.xml [Browse...]

Source is from a previous step: ☐

Source field name: [dropdown]

Add filename to result?: ☐

Skip (Elements/Attributes): 0

Limit (Elements/Attributes): 0

Default String Length: 1024

Encoding: UTF-8

Add Namespace information?: ☐

Trim strings?: ☒

Include filename in output?: ☐ Fieldname: xml\_filename

Row number in output?: ☐ Fieldname: xml\_row\_number

XML data type (numeric) in output?: ☐ Fieldname: xml\_data\_type\_numeric

XML data type (description) in output?: ☒ Fieldname: xml\_data\_type\_description

XML location line in output?: ☐ Fieldname: xml\_location\_line

XML location column in output?: ☐ Fieldname: xml\_location\_column

XML element ID in output?: ☐ Fieldname: xml\_element\_id

XML parent element ID in output?: ☐ Fieldname: xml\_parent\_element\_id

XML element level in output?: ☐ Fieldname: xml\_element\_level

XML path in output?: ☒ Fieldname: xml\_path

XML parent path in output?: ☐ Fieldname: xml\_parent\_path

XML data name in output?: ☒ Fieldname: xml\_data\_name

XML data value in output?: ☒ Fieldname: xml\_data\_value

[Help] [OK] [Preview] [Cancel]

Figura 7 – Passo “XML input”

Na figura abaixo, podemos ver que este tipo de leitura não devolve objetos representativos de palavras, mas sim toda a informação existente no *XML*, embora o faça de forma muito granular, desde a abertura e fecho de elementos, atributos, texto, etc. Portanto, o problema que esta leitura causa é precisamente ter de ser o utilizador a agrupar os dados, neste caso formando palavras.

#	xml_data_type_description	xml_path	xml_data_name	xml_data_value
1	START_DOCUMENT		<null>	<null>
2	START_ELEMENT	/dic	dic	<null>
3	START_ELEMENT	/dic/head	head	<null>
4	CHARACTERS	/dic/head	head	A
5	END_ELEMENT	/dic/head	head	<null>
6	START_ELEMENT	/dic/entry	entry	<null>
7	ATTRIBUTE	/dic/entry	id	achafundar
8	ATTRIBUTE	/dic/entry	ast	1
9	START_ELEMENT	/dic/entry/form	form	<null>
10	START_ELEMENT	/dic/entry/form/orth	orth	<null>
11	CHARACTERS	/dic/entry/form/orth	orth	Achafundar
12	END_ELEMENT	/dic/entry/form/orth	orth	<null>
13	END_ELEMENT	/dic/entry/form	form	<null>
14	START_ELEMENT	/dic/entry/sense	sense	<null>
15	START_ELEMENT	/dic/entry/sense/gramGrp	gramGrp	<null>
16	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	v. t.
17	END_ELEMENT	/dic/entry/sense/gramGrp	gramGrp	<null>
18	START_ELEMENT	/dic/entry/sense/usg	usg	<null>
19	ATTRIBUTE	/dic/entry/sense/usg	type	style
20	CHARACTERS	/dic/entry/sense/usg	usg	Pop.
21	END_ELEMENT	/dic/entry/sense/usg	usg	<null>
22	START_ELEMENT	/dic/entry/sense/def	def	<null>
23	CHARACTERS	/dic/entry/sense/def	def	Enterrar no lodo; meter no fundo da água.
24	END_ELEMENT	/dic/entry/sense/def	def	<null>
25	END_ELEMENT	/dic/entry/sense	sense	<null>
26	END_ELEMENT	/dic/entry	entry	<null>
27	START_ELEMENT	/dic/entry	entry	<null>
28	ATTRIBUTE	/dic/entry	id	abacamartado
29	ATTRIBUTE	/dic/entry	ast	1
30	START_ELEMENT	/dic/entry/form	form	<null>
31	START_ELEMENT	/dic/entry/form/orth	orth	<null>
32	CHARACTERS	/dic/entry/form/orth	orth	Abacamartado
33	END_ELEMENT	/dic/entry/form/orth	orth	<null>

Figura 8 – Previsualização dos dados de saída do passo “XML input”

O passo seguinte, “Filtro de linhas”, limpa a informação desnecessária e, para isso, foi utilizada a funcionalidade “Filter rows”. Esta funcionalidade permite criar condições sobre os campos recebidos do passo anterior. Neste caso, o filtro descarta todas as ocorrências que não tenham valor e que sejam atributos.

Step name: Filtro de linhas

Send 'true' data to step: Definidor de grupo

Send 'false' data to step: Lixo

The condition:

☐ +

xml\_data\_value IS NOT NULL

AND

xml\_data\_type\_description <> [ATTRIBUTE]

Buttons: Help, OK, Cancel

Figura 9 – Passo “Filtro de linhas”

#	xml_data_type_description	xml_path	xml_data_name	xml_data_value
1	CHARACTERS	/dic/head	head	A
2	CHARACTERS	/dic/entry/form/orth	orth	Achafundar
3	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	v. t.
4	CHARACTERS	/dic/entry/sense/usg	usg	Pop.
5	CHARACTERS	/dic/entry/sense/def	def	Enterrar no lodo; meter no fundo da água.
6	CHARACTERS	/dic/entry/form/orth	orth	Abacamartado
7	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	adj.
8	CHARACTERS	/dic/entry/sense/def	def	Parecido com um bacamarte:
9	CHARACTERS	/dic/entry/sense/def/cit/quote	quote	«_uma cravina abacamartada_»
10	CHARACTERS	/dic/entry/sense/def	def	(De um testamento de 1693)
11	CHARACTERS	/dic/entry/form/orth	orth	Abafo
12	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	m.
13	CHARACTERS	/dic/entry/sense/def	def	Roupa ou pelles, que servem de agasalho.
14	CHARACTERS	/dic/entry/etym	etym	(De _abafar_)

Figura 10 – Previsualização dos dados de saída do passo “Filtro de linhas”

No próximo passo, “Definidor de palavra”, com recurso à funcionalidade “*Modified JavaScript value*” que utiliza código *JavaScript*, as várias linhas da figura acima são enumeradas de acordo com a palavra a que pertencem, é construída a definição da palavra (visto que esta, por vezes, está dividida em várias linhas) e é associada à palavra a sua letra inicial.

Em resumo, a *script* “Item\_0” é a *script* de arranque que define as variáveis e a *script* “Script 1” é a *script* de transformação. Esta última, sempre que deteta um elemento “orth”, que representa o início de uma palavra, incrementa a variável “palavra” (para que os próximos dados sejam a ela associados) e limpa a variável “def”. Depois, sempre que deteta um elemento “def” ou um caminho que pertença a este, o conteúdo desse elemento é concatenado à variável “def” que representa a definição da palavra. Por fim, sempre que se deteta um elemento “head”, que representa a divisória entre a letra inicial das palavras, o seu conteúdo é guardado para futuramente sabermos a letra inicial de cada palavra.

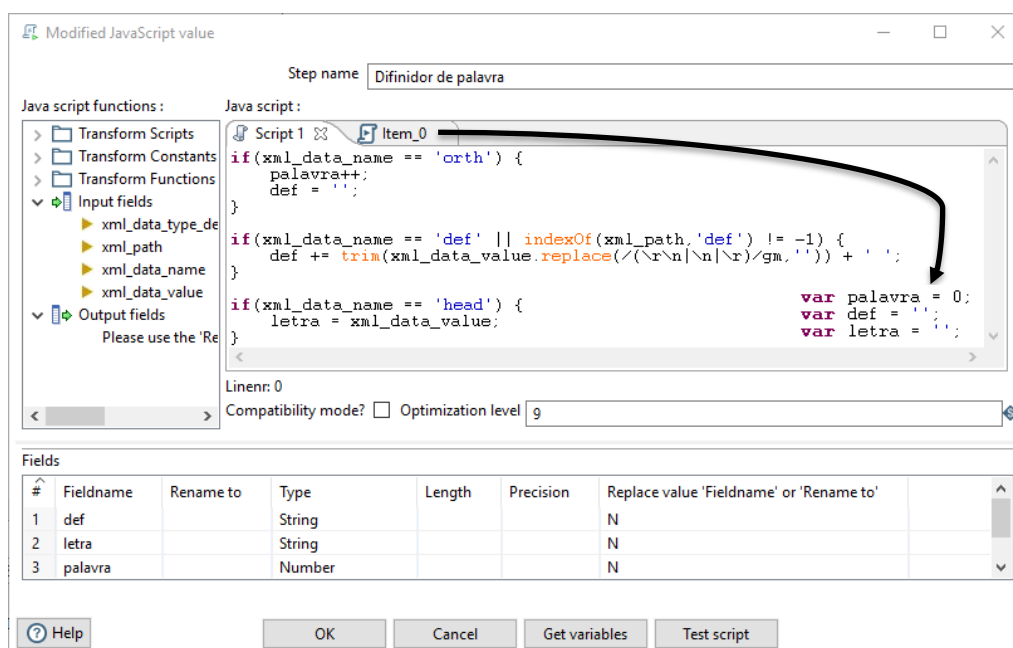


Figura 11 – Passo “Definidor de palavra”

#	xml_data_type_description	xml_path	xml_data_name	xml_data_value	def	letra	palavra
1	CHARACTERS	/dic/head	head	A		A	0,0
2	CHARACTERS	/dic/entry/form/orth	orth	Achafundar		A	1,0
3	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	v. t.		A	1,0
4	CHARACTERS	/dic/entry/sense/usg	usg	Pop.		A	1,0
5	CHARACTERS	/dic/entry/sense/def	def	Enterrar no lodo; meter no fundo da água.	Enterrar no lodo; meter no fundo da...	A	1,0
6	CHARACTERS	/dic/entry/form/orth	orth	Abacamartado		A	2,0
7	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	adj.		A	2,0
8	CHARACTERS	/dic/entry/sense/def	def	Parecido com um bacamarte:	Parecido com um bacamarte:	A	2,0
9	CHARACTERS	/dic/entry/sense/def/cit/quote	quote	«_uma cravina abacamartada_»	Parecido com um bacamarte: «_um...	A	2,0
10	CHARACTERS	/dic/entry/sense/def	def	(De um testamento de 1693)	Parecido com um bacamarte: «_um...	A	2,0
11	CHARACTERS	/dic/entry/form/orth	orth	Abafo		A	3,0
12	CHARACTERS	/dic/entry/sense/gramGrp	gramGrp	m.		A	3,0
13	CHARACTERS	/dic/entry/sense/def	def	Roupa ou pelles, que servem de agasalho.	Roupa ou pelles, que servem de aga...	A	3,0
14	CHARACTERS	/dic/entry/etym	etym	(De _abafar_)	Roupa ou pelles, que servem de aga...	A	3,0

Figura 12 – Previsualização dos dados de saída do passo “Definidor de palavra”

O passo seguinte é onde “a magia acontece”. Utilizando a funcionalidade “Row denormaliser”, as linhas da figura acima são agrupadas pelo campo “palavra”. Desta forma, uma palavra passa a ser apenas representada numa linha com toda a sua informação em colunas.

Step name:

The key field:

The fields that make up the grouping:

#	Group field
1	palavra

Target fields:

#	Target fieldname	Value fieldname	Key value	Type
1	orth	xml_data_value	orth	String
2	gramGrp	xml_data_value	gramGrp	String

Figura 13 – Passo “Agrupador”

#	xml_data_type_description	xml_path	def	letra	palavra	orth	gramGrp
1	CHARACTERS	/dic/head		A	0,0	<null>	<null>
2	CHARACTERS	/dic/entry/sense/def	Enterrar no lodo; meter no fundo da água.	A	1,0	Achafundar	v. t.
3	CHARACTERS	/dic/entry/sense/def	Parecido com um bacamarte: «_uma cravina abacamartada...	A	2,0	Abacamartado	adj.
4	CHARACTERS	/dic/entry/etym	Roupa ou pelles, que servem de agasalho.	A	3,0	Abafo	m.
5	CHARACTERS	/dic/entry/etym	Pouco cómodo, (falando-se do andar do cavalo).	A	4,0	Abaloso	adj.
6	CHARACTERS	/dic/entry/sense/def	Acto ou efeito de abarregar.	A	5,0	Abarregamento	m.
7	CHARACTERS	/dic/entry/etym	Diz-se do reumatismo, que ataca os órgãos, os músculos, ...	A	6,0	Abarticular	adj.
8	CHARACTERS	/dic/entry/sense/def	O mesmo que _besoiro_. (Cast. _abejón_)	A	7,0	Abegão	m.
9	CHARACTERS	/dic/entry/sense/def	Espécie de orquídeas, o mesmo que _abelha-flôr_.	A	8,0	Abelhina	f.
10	CHARACTERS	/dic/entry/sense/def	Apressar-se. Cf. Costa e Sá, 'Diccion.' (Talvez escrita incorre...	A	9,0	Abelhuar-se	v. p.
11	CHARACTERS	/dic/entry/sense/def	O mesmo que _besoiro_.	A	10,0	Abesoiro	m.
12	CHARACTERS	/dic/entry/sense/def	O mesmo que _besouro_.	A	11,0	Abesouro	m.
13	CHARACTERS	/dic/entry/sense/def	O mesmo que _bestunto_.	A	12,0	Abestunto	m.
14	CHARACTERS	/dic/entry/sense/def	O que abetuma.	A	13,0	Abetumador	m.

Figura 14 – Previsualização dos dados de saída do passo “Agrupador”

No próximo passo, “Retirar *heads* e *NULLS*”, utilizando de novo a funcionalidade “*Filter rows*”, as linhas da figura acima que representem “*heads*” ou palavras que tenham “*orth*”, “*gramGrp*” ou “*def*” a “*NULL*”, por exemplo a primeira, são retiradas, ficando este passo a devolver apenas palavras.

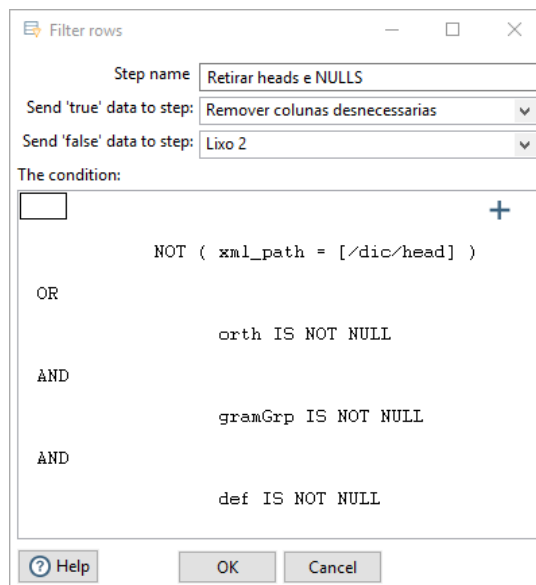


Figura 15 – Passo “Retirar *heads* e *NULLS*”

Examine preview data

Rows of step: Saída 2 (1000 rows)

#	xml_data_type_description	xml_path	def	letra	palavra	orth	gramGrp
1	CHARACTERS	/dic/entry/sense/def	Enterrar no lodo; meter no fundo da água.	A	1,0	Achafundar	v. t.
2	CHARACTERS	/dic/entry/sense/def	Parecido com um bacamarte: «_uma cravina abacamartad...	A	2,0	Abacamartado	adj.
3	CHARACTERS	/dic/entry/etym	Roupa ou pelles, que servem de agasalho.	A	3,0	Abafó	m.
4	CHARACTERS	/dic/entry/etym	Pouco cômodo, (falando-se do andar do cavalo).	A	4,0	Abaloso	adj.
5	CHARACTERS	/dic/entry/sense/def	Acto ou effeito de abarregar.	A	5,0	Abarregamento	m.
6	CHARACTERS	/dic/entry/etym	Diz-se do rheumatismo, que ataca os órgãos, os músculos,...	A	6,0	Abarticular	adj.
7	CHARACTERS	/dic/entry/sense/def	O mesmo que _besoiro_.(Cast._abejón_)	A	7,0	Abegão	m.
8	CHARACTERS	/dic/entry/sense/def	Espécie de orquídeas, o mesmo que _abelha-flôr_.	A	8,0	Abelhina	f.

Close Stop Get more rows

Figura 16 – Previsualização dos dados de saída do passo “Retirar *heads* e *NULLS*”

Por fim, no passo “Remover colunas desnecessárias”, utilizando a funcionalidade “*Select values*”, apenas as colunas que serão utilizadas na continuação da execução do projeto são passadas para a saída da transformação.

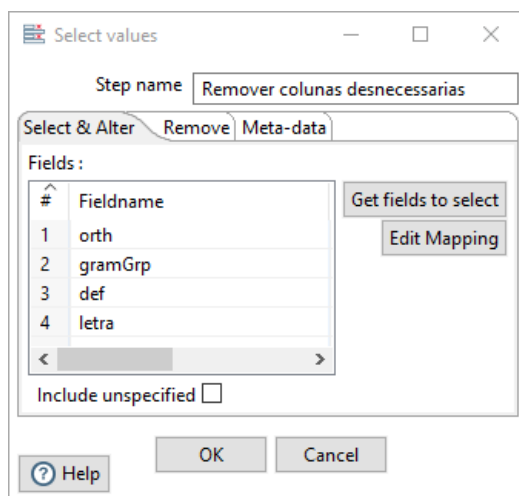
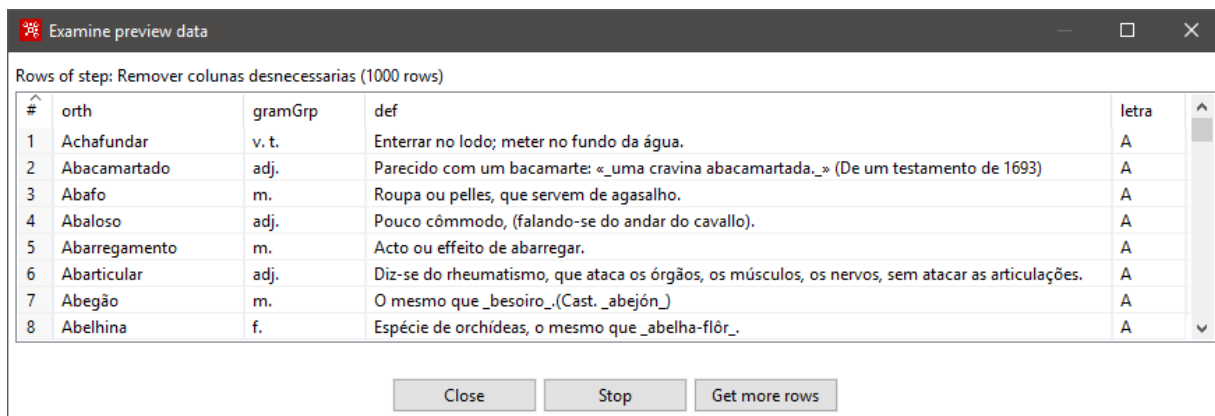


Figura 17 – Passo “Remover colunas desnecessárias”

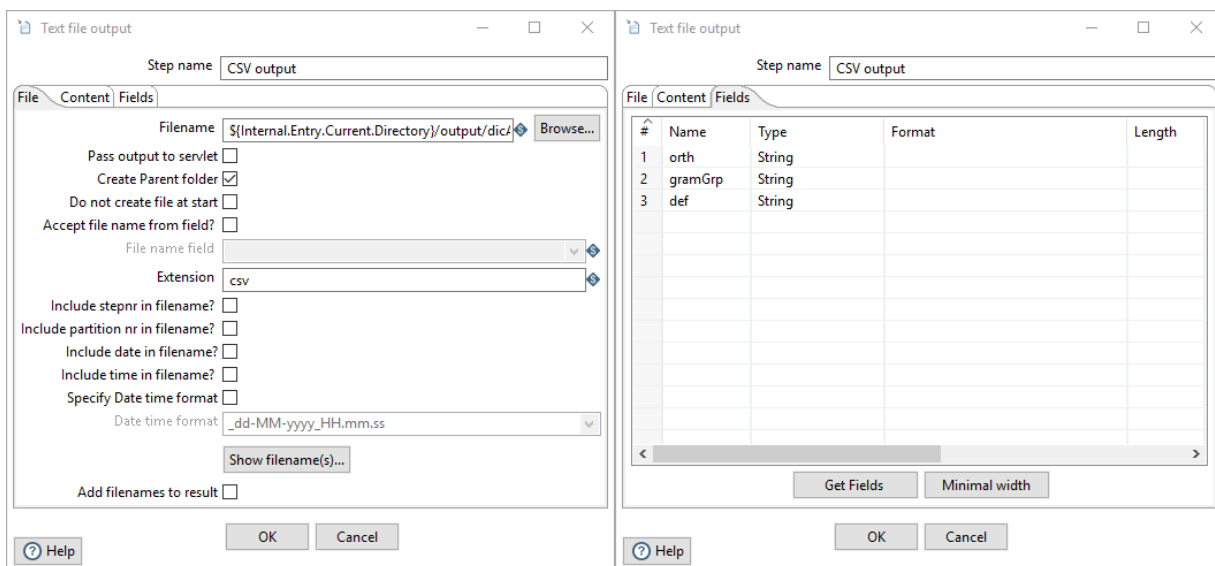


#	orth	gramGrp	def	letra
1	Achafundar	v. t.	Enterrar no lodo; meter no fundo da água.	A
2	Abacamartado	adj.	Parecido com um bacamarte: « uma cravina abacamartada.» (De um testamento de 1693)	A
3	Abafó	m.	Roupa ou pelles, que servem de agasalho.	A
4	Abaloso	adj.	Pouco cômodo, (falando-se do andar do cavalo).	A
5	Abarregamento	m.	Acto ou efeito de abarregar.	A
6	Abarticular	adj.	Diz-se do reumatismo, que ataca os órgãos, os músculos, os nervos, sem atacar as articulações.	A
7	Abegão	m.	O mesmo que _besoiro_. (Cast. _abejón_)	A
8	Abelhina	f.	Espécie de orchídeas, o mesmo que _abelha-flôr_.	A

Figura 18 – Previsualização dos dados de saída do passo “Remover colunas desnecessárias”

No processo de “Exportação total de dados” (Figura 4) foram utilizados como dados de entrada os da Figura 18. Neste processo são efetuadas 3 exportações, em *CSV*, *JSON* e *XML*.

No passo “*CSV output*” é necessário indicar o diretório onde o ficheiro vai ser guardado, o seu nome, extensão e que campos vão nele ser escritos.



Step name: CSV output

File Content Fields

Filename: \${Internal.Entry.Current.Directory}/output/dic/ Browse...

Pass output to servlet: ☐

Create Parent folder: ☒

Do not create file at start: ☐

Accept file name from field: ☐

File name field: [dropdown]

Extension: csv

Include stepnr in filename: ☐

Include partition nr in filename: ☐

Include date in filename: ☐

Include time in filename: ☐

Specify Date time format: ☐

Date time format: \_dd-MM-yyyy\_HH.mm.ss

Show filename(s)...

Add filenames to result: ☐

OK Cancel

Step name: CSV output

File Content Fields

#	Name	Type	Format	Length
1	orth	String		
2	gramGrp	String		
3	def	String		

Get Fields Minimal width

OK Cancel

Figura 19 – Passo “*CSV output*” (total)



No passo, “*JSON output*” é necessário indicar o mesmo que no “*CSV output*” com a adição do nome do bloco *JSON* e quantas linhas queremos por bloco (0 significa todas as linhas no mesmo bloco).

The screenshot shows the 'JSON output' configuration window. The 'Step name' is 'JSON output'. The 'General' tab is selected. The 'Operation' is set to 'Write to file'. Under 'Settings', 'Json bloc name' is 'data', 'Nr rows in a bloc' is '0', and 'Output Value' is 'outputValue'. 'Compatibility mode' is unchecked. Under 'Output File', 'Filename' is '\$\${Internal.Entry.Current.Directory}/o', 'Append' is unchecked, 'Create Parent folder' is checked, 'Do not open create at' is unchecked, 'Extension' is 'js', and 'Encoding' is 'UTF-8'. There are checkboxes for 'Pass output to servlet', 'Include date in', and 'Include time in', all of which are unchecked. A 'Show filename(s)...' button is present. At the bottom, there is an 'Add File to result' checkbox (unchecked), a 'Help' button, and 'OK' and 'Cancel' buttons.

Figura 20 – Passo “*JSON output*”

No passo “*XML output*” é apenas necessário o mesmo que no “*CSV output*”.

The screenshot shows the 'XML output' configuration window. The 'Step name' is 'XML output'. The 'File' tab is selected. The 'Filename' is '\$\${Internal.Entry.Current.Directory}/o', with a 'Browse...' button. There are checkboxes for 'Do not create file at start', 'Pass output to servlet', 'Include stepnr in filename?', 'Include date in filename?', 'Include time in filename?', and 'Specify Date time format', all of which are unchecked. The 'Extension' is 'xml'. There is a 'Date time format' dropdown menu. A 'Show filename(s)...' button is present. At the bottom, the 'Add filenames to result' checkbox is checked, and there is a 'Help' button, 'OK' button, and 'Cancel' button.

Figura 21 – Passo “*XML output*”

No passo “*BD output*” é necessário criar uma conexão com o nosso servidor de base de dados, neste caso *MySQL*, indicando qual esquema queremos aceder e em que tabela queremos inserir os dados. Podemos também indicar quantos registos serão inseridos de cada vez, por exemplo 10 000, para que seja necessário aceder menos vezes à base de dados e por consequência melhorar a velocidade de execução. Por fim, indicamos que campos recebidos do passo anterior correspondem a que colunas da tabela na base de dados.

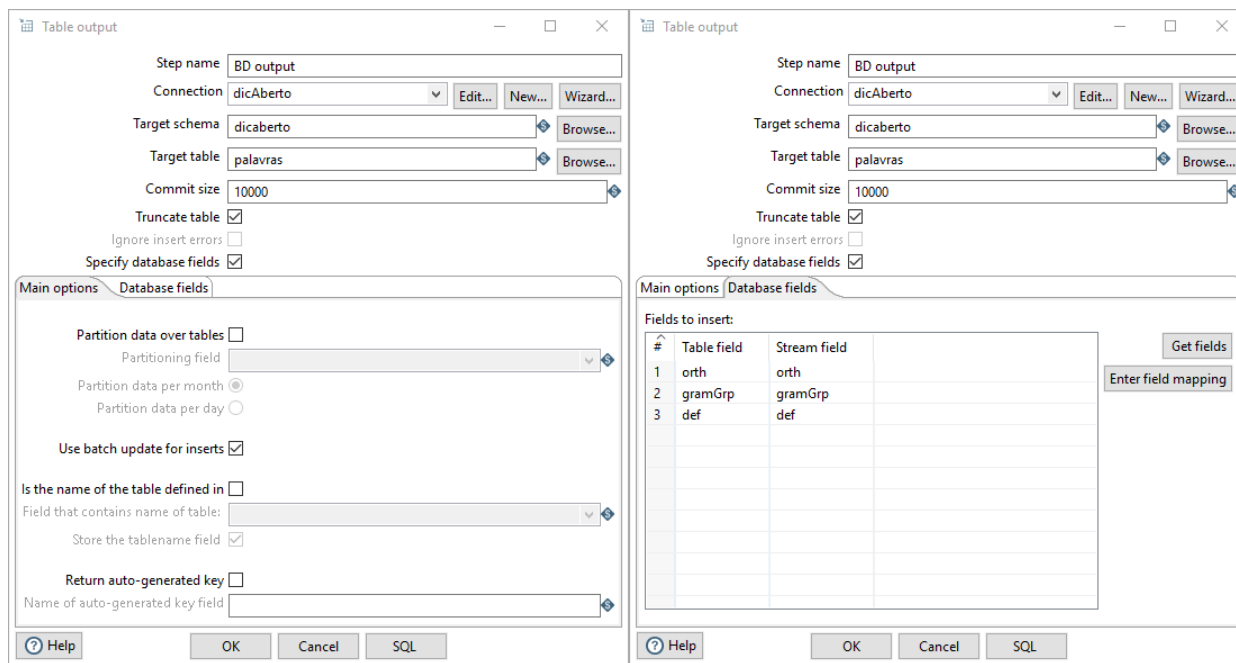


Figura 22 – Passo “*BD output*”

No processo de “Exportação agrupada por letra inicial” (Figura 5) foram utilizados como dados de entrada os da Figura 18. Neste processo é efetuada uma exportação em CSV para cada letra do alfabeto. Para isso, é utilizado o campo “letra” para dar o nome ao ficheiro, porém, ao utilizar esta funcionalidade, já não podemos definir onde guardar estes ficheiros, que vão acabar por ser guardados na pasta de instalação do programa.

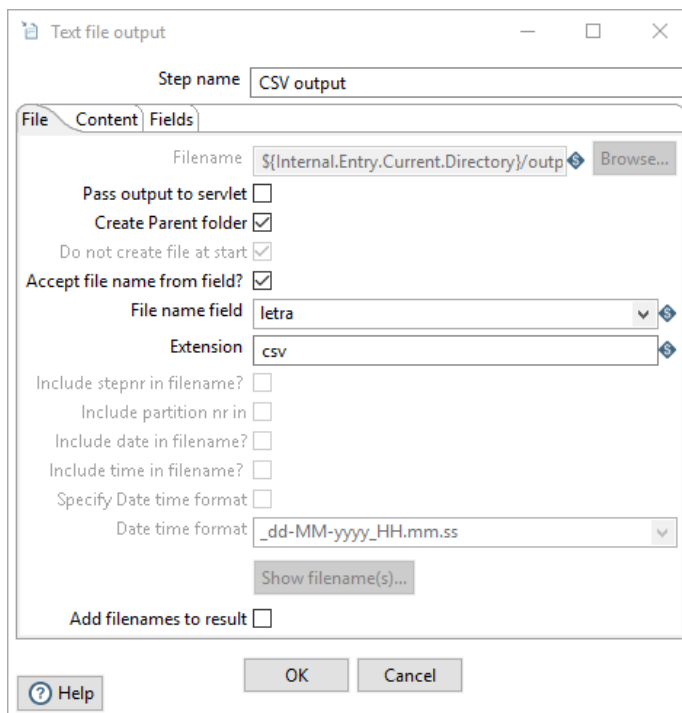


Figura 23 – Passo “*CSV output*” (por letra)

## Knime

### Esquema da solução

Esta solução apresenta 2 “Workflows” e 4 Componentes.

O primeiro “Workflow” agrupa todas as funcionalidades do projeto. É neste onde recebemos os dados de um segundo “Workflow” e os distribuimos pelos 4 componentes que por sua vez exportam os dados. Possibilita a visualização dos resultados da exportação realizada pelos componentes com também permite a eliminação desses mesmos resultados.

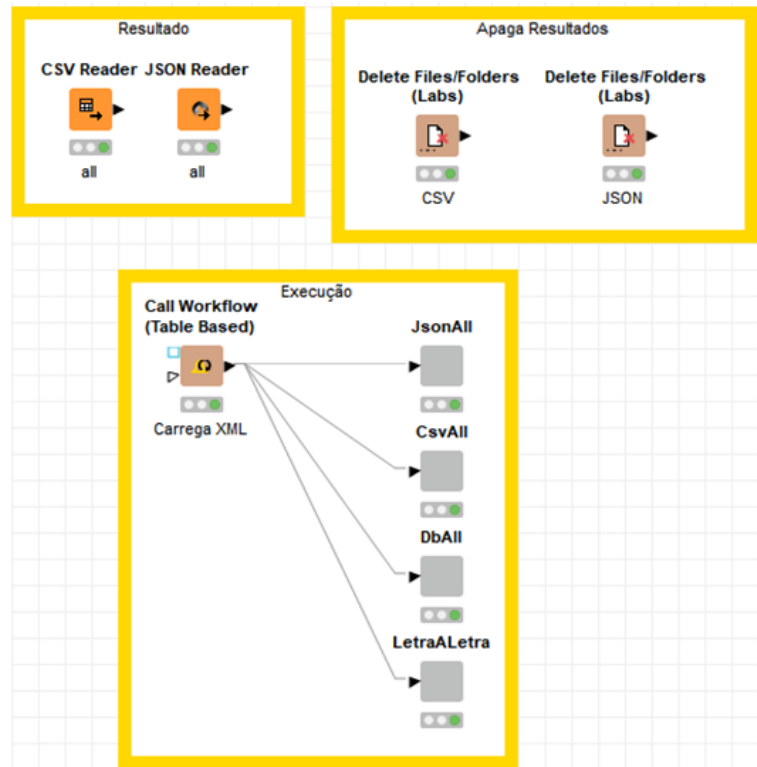


Figura 24 – “Workflow” Principal

O segundo “Workflow” é responsável por carregar os dados do dicionário de palavras em formato XML. Utilizando filtros e expressões regulares cada registo é filtrado/alterado de maneira a remover a informação incompleta e remover os caracteres especiais de mudança de linha.

Cada um dos 4 componentes exporta os dados de diferente forma:

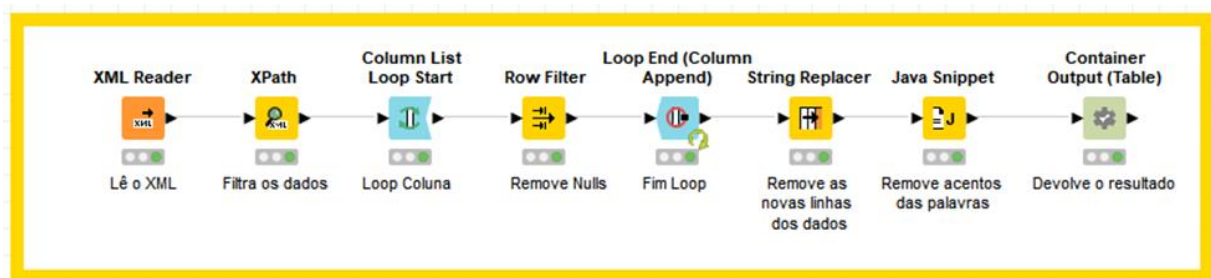


Figura 25 – “Workflow” DicXML

- Todas as palavras para *JSON*;

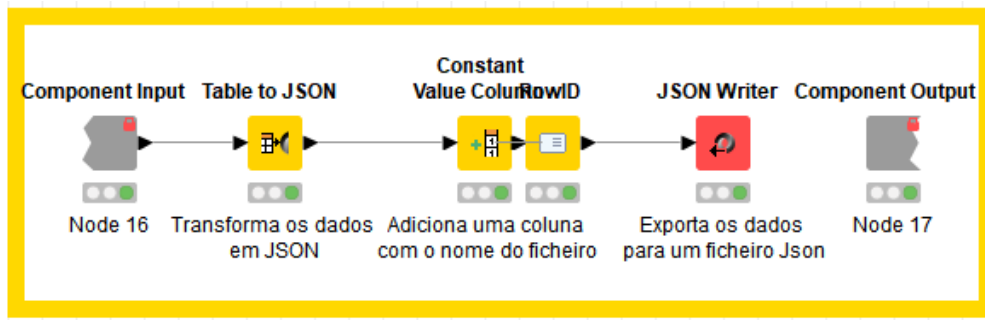


Figura 26 – Componente "JsonAll"

- Todas as palavras para *CSV*;

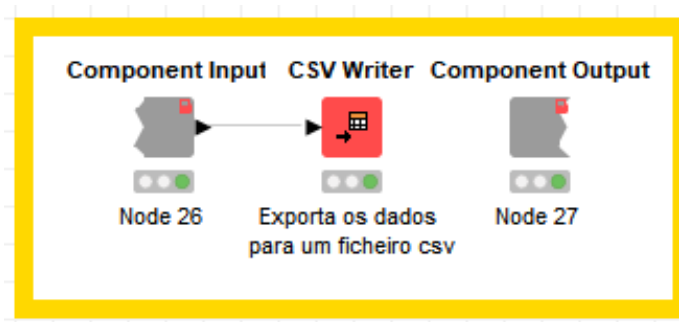


Figura 27 – Componente "CsvAll"

- Todas as palavras para uma base de dados;

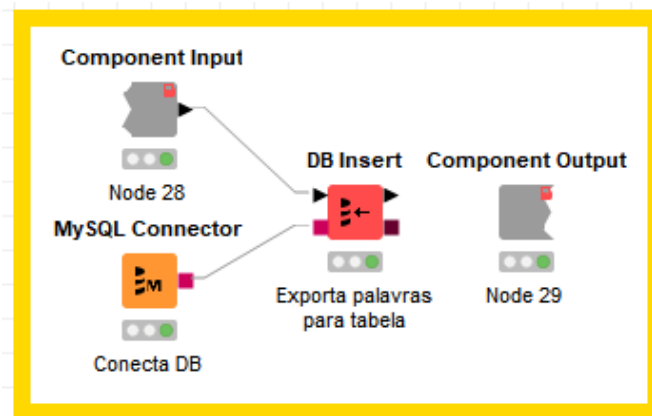


Figura 28 – Componente "DbAll"

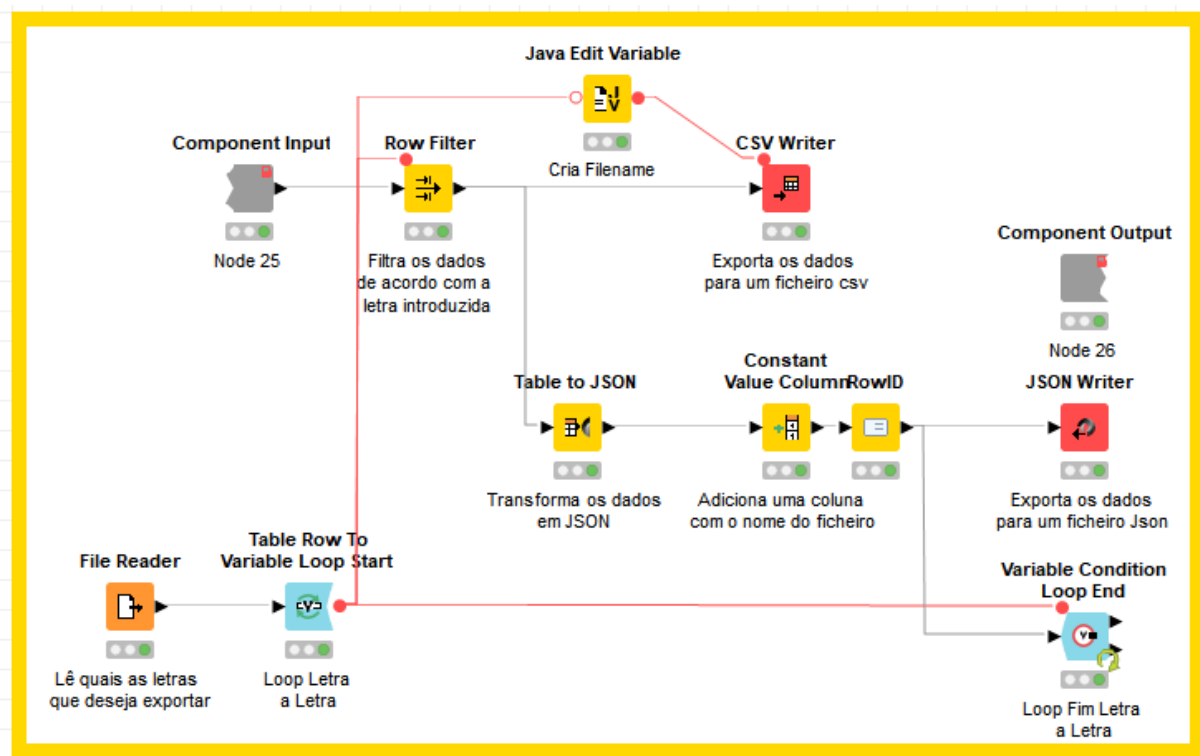


Figura 29 – Componente “LetraALetra”

## Funcionamento da solução

A execução do “Workflow” principal, está dividida em duas tarefas principais.

A primeira tarefa desempenhada é a invocação do “Workflow” de leitura dos dados de um ficheiro XML (Figura 30).

Este ficheiro é lido utilizando um “XML Reader” que abre um ficheiro XML (“Input Location”) e utilizando um filtro “XPath” com “/dic/entry” envia apenas esses elementos (palavras) para o componente seguinte.

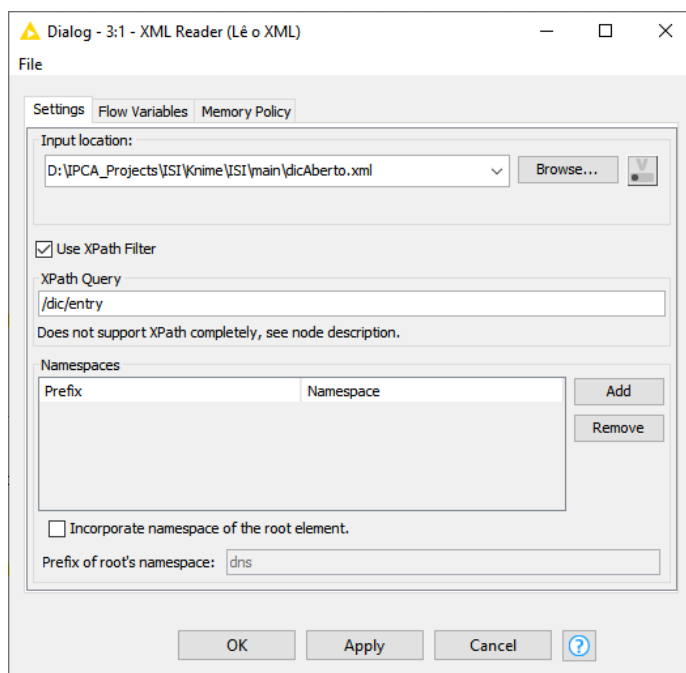


Figura 30 – Configuração do “XML Reader”

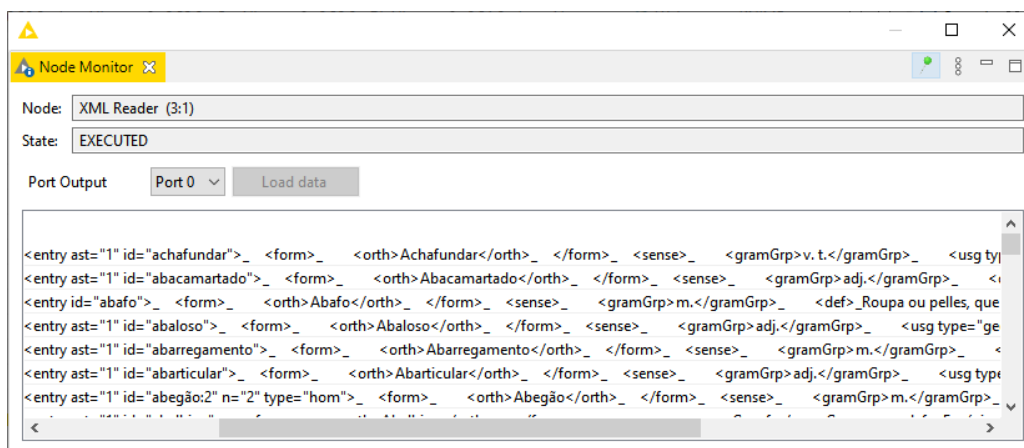


Figura 31 – Dados de saída do “XML Reader”

O passo seguinte é transformar este XML, que contém a informação de uma palavra, num registo (linha) de uma tabela. A informação relevante que irá ser extraída será:

- “Orth”: palavra;
- “Def”: definição da palavra;
- “GramGrp”: classe gramatical;

Para essa finalidade, utiliza-se o “XPath” para extrair a informação da palavra contida entre as “tags” de XML.

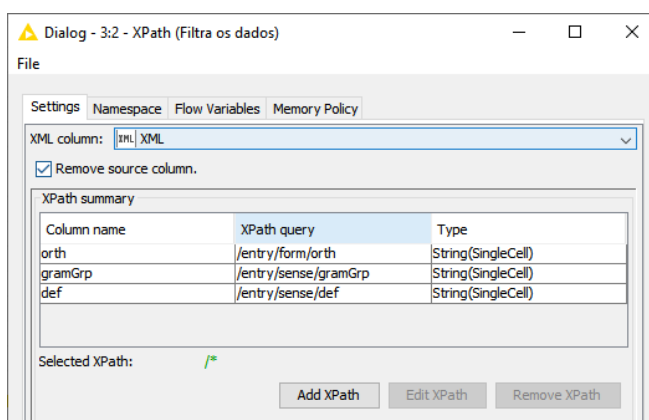


Figura 32 – Configuração do “XPath” para Filtrar dados

Output Table - 3:2 - XPath (Filtra os dados)			
File Edit Hilite Navigation View			
Table "default" - Rows: 151457 Spec - Columns: 3 Properties Flow Variables			
Row ID	orth	g...	def
Row0_1	Achafundar	v. t.	Enterrar no lodo; meter no fundo da água.
Row1_1	Abacamartado	adj.	Parecido com um bacamarte: «_uma cravina ...
Row2_1	Abafo	m.	Roupa ou pelles, que servem de agasalho.
Row3_1	Abaloso	adj.	Pouco cômodo, (falando-se do andar do ca...
Row4_1	Abarregamento	m.	Acto ou efeito de abarregar.
Row5_1	Abarticular	adj.	Diz-se do reumatismo, que ataca os órgãos...
Row6_1	Abegão	m.	O mesmo que _besoiro_.
Row7_1	Abelhina	f.	Espécie de orchídeas, o mesmo que _abelha...
Row8_1	Abelhuar-se	v. p.	Apressar-se. Cf. Costa e Sá, _Diccion. (Tal...
Row9_1	Abesoiro	m.	O mesmo que _besoiro_.
Row10_1	Abesouro	m.	O mesmo que _besouro_.
Row11_1	Abestunto	m.	O mesmo que _bestunto_. (Colhido em Villa...
Row12_1	Abetumador	m.	.

Figura 33 – Dados de saída do “XPath” para Filtrar dados

De seguida, cada registo é filtrado/alterado de maneira a remover a informação incompleta e remover os caracteres especiais de mudança de linha.

Relativamente á remoção da informação incompleta, é utilizado um “*Column List Loop*” que percorre cada coluna e por cada iteração, irá enviar para o “*Row Filter*” a coluna atual.

Na Figura 34, podemos observar a configuração do “*Row Filter*” que remove todas as linhas que não possuam dados da coluna atual da iteração. O nome da coluna é fornecido e alterado a cada iteração do *loop*.

Por fim, o “*Loop End (Column Append)*” é responsável por ir juntando os resultados intermédios de cada iteração para que no final do *loop* voltemos a possuir a estrutura da tabela inicial.

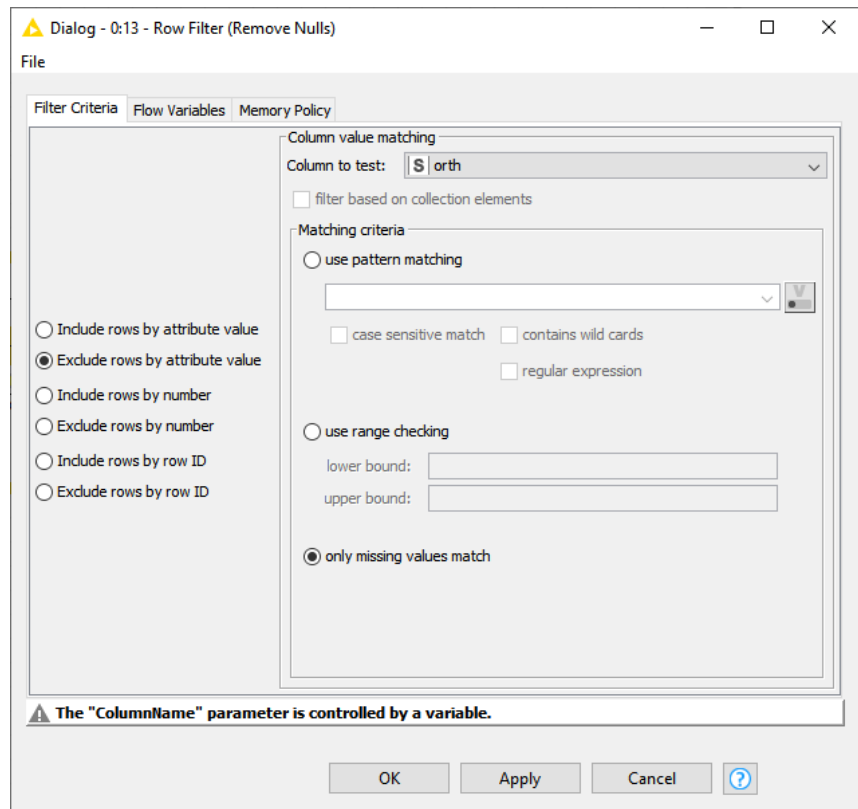


Figura 34 – Remoção de dados incompletos

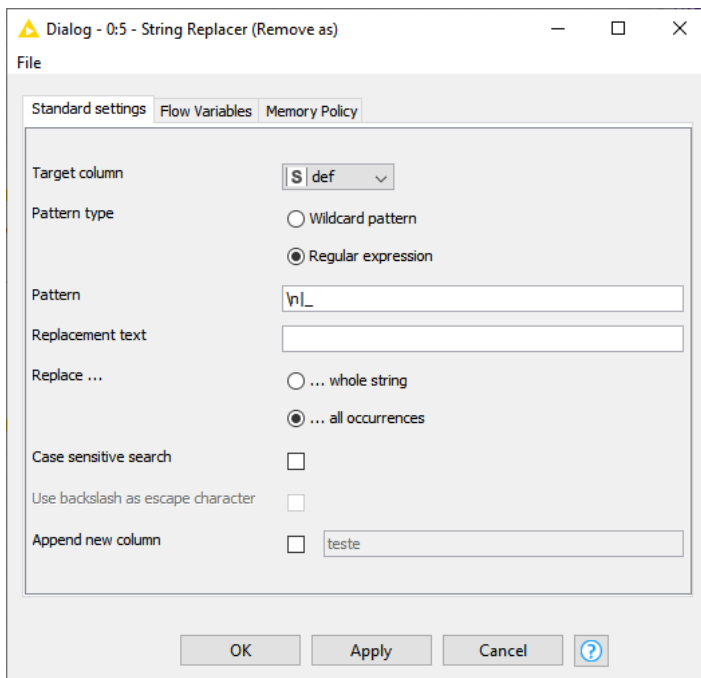


Figura 35 – Remoção das mudanças de linha

As definições das palavras possuem, na sua maioria, um *line-feed* (nova linha, mais conhecido por “\n”) no início e no fim da frase.

Para o remover, juntamente com todos os “\_” (underscore), é utilizada um “*String Replacer*”, que por sua vez usa expressões regulares para identificar esses caracteres e remover.

De forma a facilitar a distinção de qual a letra inicial de uma palavra, para mais tarde ser mais simples obter todas as palavras iniciadas por essa letra e tendo em conta que algumas letras podem começar com acentos, foi desenvolvida uma função utilizando o “*Java Snippet*” que transforma esses caracteres em caracteres equivalentes sem os acentos.

Essa função usa a classe “*Normalizer*” que possui um método chamado “*normalize*” que é utilizado para normalizar caracteres, ou seja, de acordo com o formato escolhido transforma uma “*string*” num formato comum de sequências de códigos que representam os mesmos caracteres. O método recebe dois parâmetros, a “*string*” a ser alterada, e o formato da normalização que pode ser:

- NFC – Formato de Normalização Canônico de Composição;
- NFD – Formato de Normalização Canônico de Decomposição;
- NFKC – Formato de Normalização de Compatibilidade de Composição;
- NFKD – Formato de Normalização de Compatibilidade de Decomposição;

A “*string*” enviada para o método a primeira letra da palavra dentro da coluna “*orth*”, e o formato da normalização usado é o NFD. O resultado desta operação é por exemplo:

- Antes da decomposição: “Á”
- Depois da decomposição: “A’”

Depois da decomposição, utilizando expressões regulares, removem-se todos os caracteres exceto as letras que vão de A até Z, ficando assim apenas a letra inicial.

Essa informação da letra inicial é guardada numa nova coluna chamada letra.

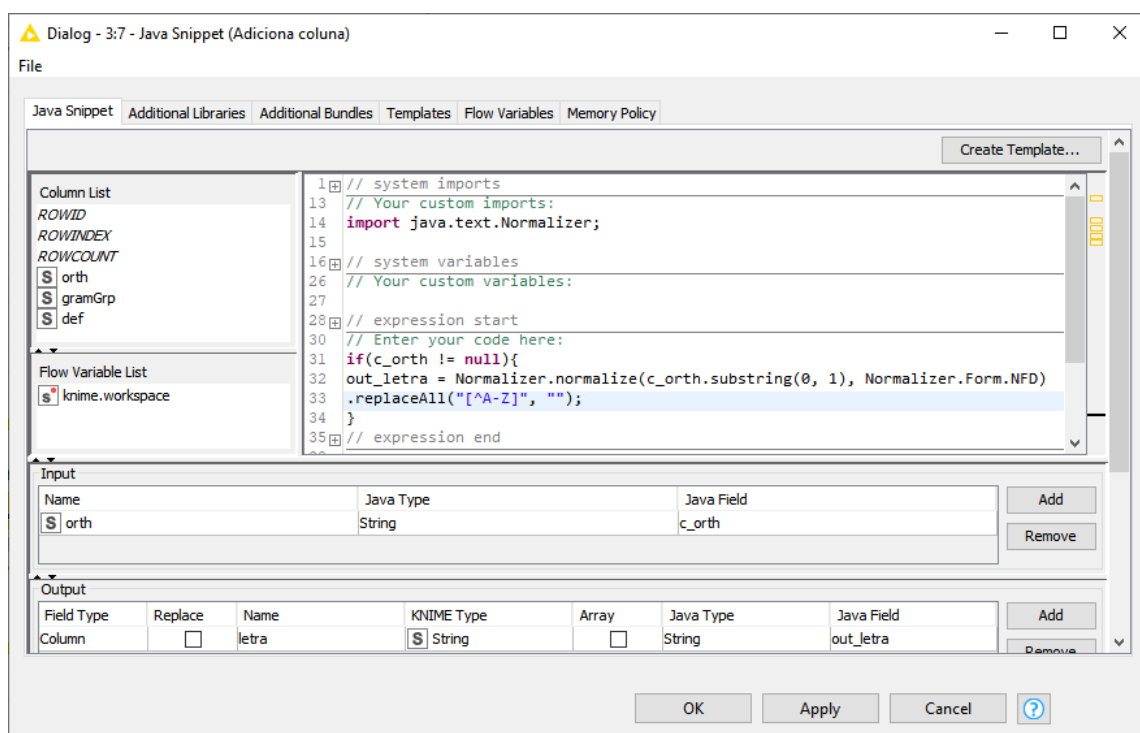


Figura 36 – “*Java Snippet*” para obter a letra inicial de uma palavra



Appended table - 3:7 - Java Snippet (Adiciona coluna)

File Edit Hilite Navigation View

Table "default" - Rows: 127583 Spec - Columns: 4 Properties Flow Variables

Row ID	S orth	S gramGrp	S def	S letra
Row0	Achafundar	v. t.	Enterrar no lodo; meter no fundo da água.	A
Row1	Abacarnatado	adj.	Parecido com um bacarnarte: «uma cravi...	A
Row2	Abafar	m.	Roupa ou pelles, que servem de agasalho.	A
Row3	Abaloso	adj.	Pouco cómodo, (falando-se do andar d...	A
Row4	Abarregar...	m.	Acto ou effeito de abarregar.	A
Row5	Abarticular	adj.	Diz-se do reumatismo, que ataca os órg...	A
Row6	Abegão	m.	O mesmo que besoiro. (Cast. abejón)	A
Row7	Abelhina	f.	Espécie de orchídeas, o mesmo que abel...	A
Row8	Abelhuar-se	v. p.	Apressar-se. Cf. Costa e Sá, Dicion. (T...	A
Row9	Abesoiro	m.	O mesmo que besoiro.	A
Row10	Abesouro	m.	O mesmo que besouro.	A
Row11	Abestunto	m.	O mesmo que bestunto. (Colhido em Villa...	A
Row12	Abetumador	m.	O que abetuma.	A
Row13	Abodegação	f.	Acto ou effeito de abodegar.	A
Row14	Aboiadinho	adj.	Diz-se do peixe morto, que os vapores d...	A
Row15	Aboiado	adj.	Que anda á tona da água.	A
Row16	Abomba	f.	O mesmo que bomba^1. Cf. Rev. Lus., I...	A
Row17	Aboucar	v. t.	O mesmo que esmocar.Matar. (Colhido e...	A
Row18	Abrachiocep...	f.	Estado de abrachiocéphalo.	A
Row19	Abrachiocep...	m.	Monstro, sem braços nem cabeça.	A
Row20	Abraquiocef...	f.	Estado de abraquiocéfalo.	A
Row21	Abraquiocéfalo	m.	Monstro, sem braços nem cabeça.	A
Row22	Abrasoar	v. t.	O mesmo que abrasonar.	A
Row23	Abronceiro	m.	O mesmo que espinheiro.	A
Row24	Abrótiga	f.	O mesmo que abrótega. Cf. Rev. Lus., X...	A
Row25	Abrunhar	v. t.	O mesmo que acabrunhar.	A
Row26	Abundidade	f.	O mesmo que abundeza. Cf. Camillo, Cur...	A
Row27	Abúlico	adj.	Relativo á abulia. Que padece abulia.	A
Row28	Abusamento	m.	O mesmo que abuso.	A
Row29	Acadar	v. t.	Receber nas mãos ou no regaço. (Contr. ...	A

Figura 37 - Dados saída do “Java Snippet”

O último passo do “Workflow” é enviar os dados para um “Container Output (Table)”. Isto significa que o “Workflow” principal ao invocar este “Workflow” irá receber todos os seus dados.

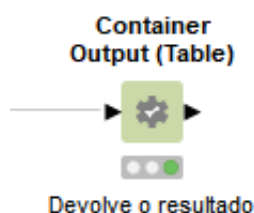


Figura 39 – Saída dos dados do “Workflow” de tratamento de dados



Figura 38 – Carregamento dos dados tratados no “Workflow” principal

A segunda tarefa principal é a exportação dos dados para os formatos referidos anteriormente neste documento.

O componente “JsonAll” transforma todas as palavras da tabela num ficheiro de JSON. Com essa finalidade em conta, o componente ao receber os dados, utilizando o “Table to JSON” são transformados todos os dados que até agora estavam numa tabela num formato JSON.

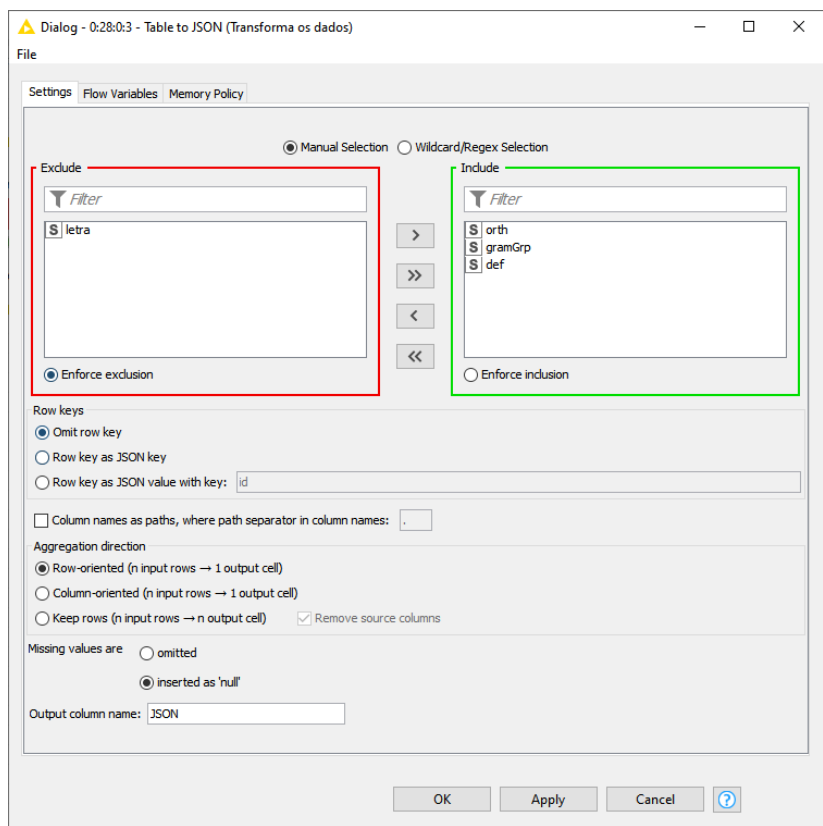


Figura 40 – Configuração do “Table to JSON”

O resultado deste passo, é uma tabela com apenas uma linha e duas colunas. A primeira coluna contém o “RowID” com o valor que representa o nome do ficheiro a ser criado. A segunda coluna contém todas as palavras em formato *JSON*.

Table "default" - Rows: 1	
Row ID	JSON export
Row1	[{"orth": "Achafundar", "gramGrp": "v. t.", "def": "Achafundar"}]

Figura 41 – Dados de saída do “Table to JSON”

De forma a alterar o nome do ficheiro exportado, foi usado um “Constant Value Column” para criar uma nova coluna com o nome pretendido para o ficheiro exportado e utilizado um “RowId” para substituí-lo pelo valor antigo na tabela.

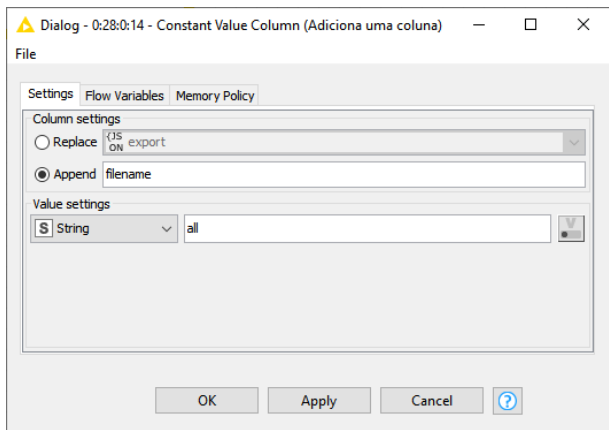


Figura 42 – Criação de uma nova coluna com o valor do nome do ficheiro exportado

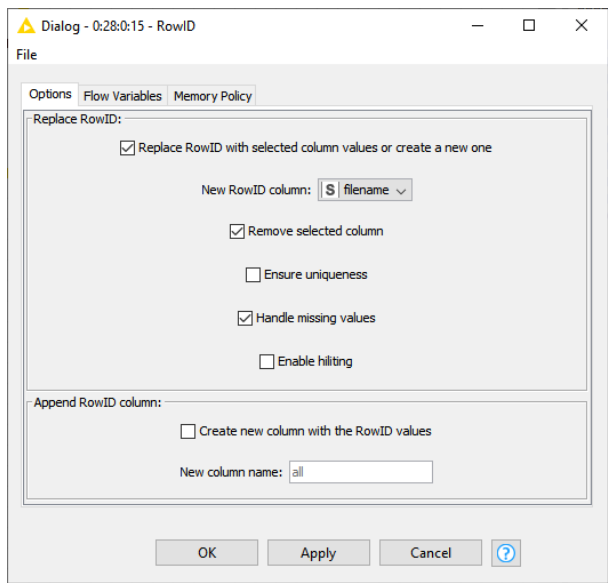


Figura 43 – Alteração do valor do “RowId”

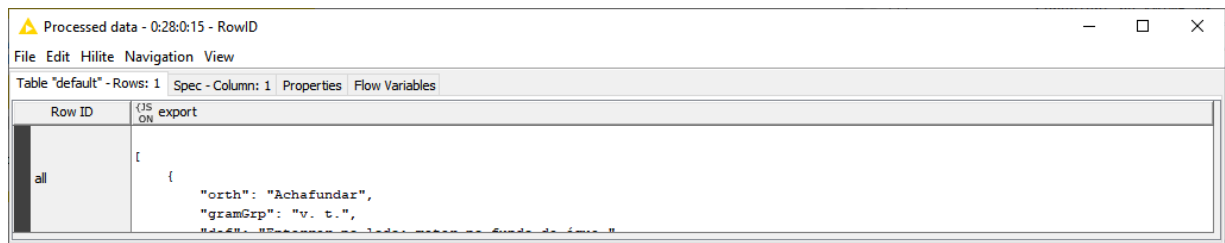


Figura 44 – Dados de saída do “RowID”

O último passo é exportar os dados para um ficheiro com extensão “.json”, para isso, é utilizado o “JSON Writer”.

Neste componente, definimos apenas a diretoria onde queremos guardar o ficheiro e a sua extensão.

A inexistência de uma forma de atribuição do nome do ficheiro exportado, por exemplo através da seleção da localização e não diretoria levou-me a implementar os passos anteriores para o conseguir definir.

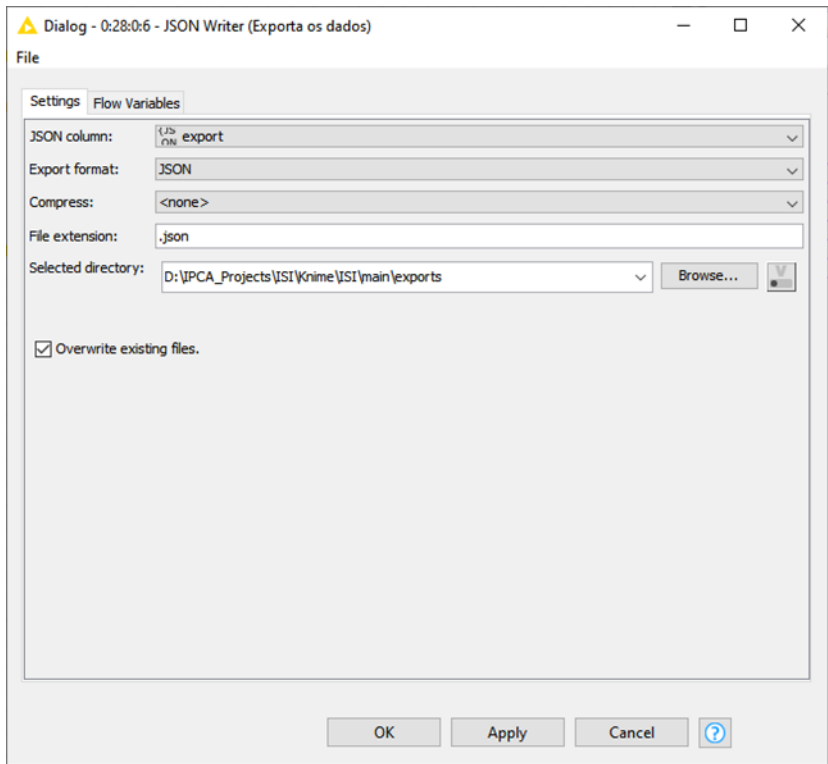


Figura 45 – Exportação dos dados utilizando “JSON Writer”

O componente “*CsvAll*”, responsável por exportar todas as palavras para CSV, é o componente “*CSV Writer*”.

É necessário configurar o “*Output Location*” que se trata da localização do ficheiro. Foi ativada a opção de “*Write Column Header*” que gera o ficheiro com os nomes das colunas (*orth*, *gramGrp* e *def*) na primeira linha do documento. O delimitador dos dados escolhido foi o caracter “;”.

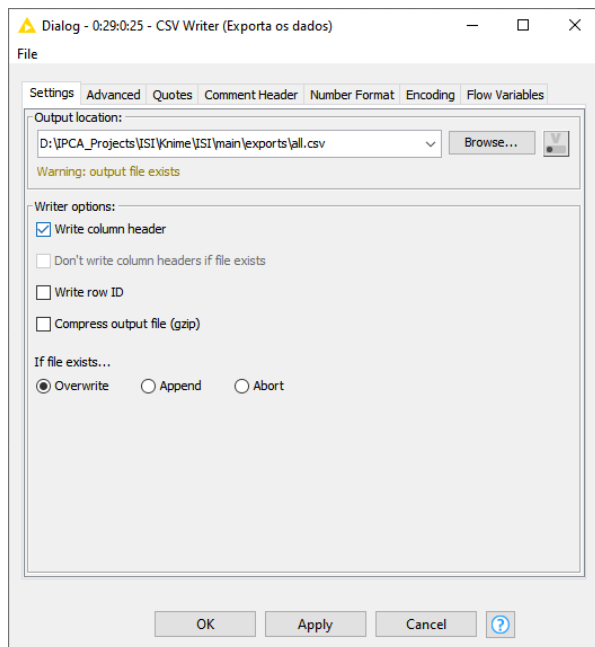


Figura 46 – Configuração do “*CSV Writer*”

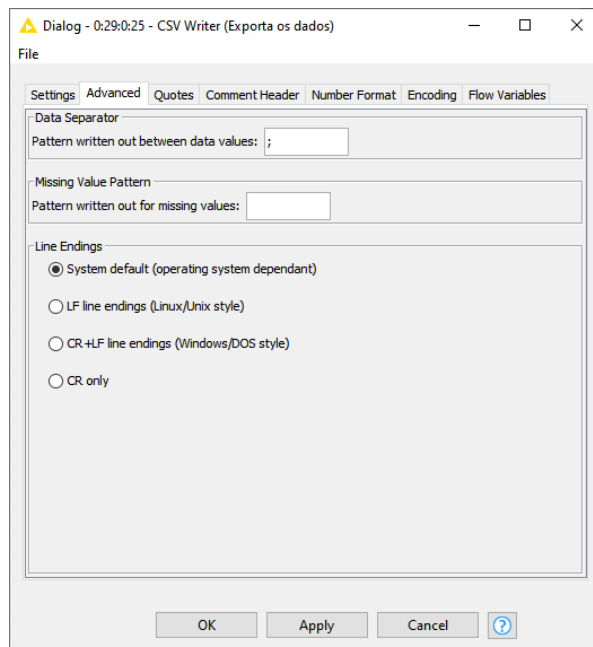


Figura 47 – Definição do caracter delimitador entre dados em CSV

O componente “*DbAll*” é constituído por um “*MySQL Connector*” que tem como função estabelecer uma ligação com o servidor de base de dados.

É necessário definir qual o gestor de base dados utilizado no campo “*Database Dialect*”, que neste caso foi o “*MySQL*”.

No campo “*Hostname*”, que se trata do local onde a base de dados se encontra, que neste caso, é a minha própria máquina, o valor deste campo é “*localhost*”.

A porta por predefinição é a 3306.

O campo “*Database name*” representa o nome da base de dados.

Para efetuar a autenticação utilizei o modo “*Username & password*”.

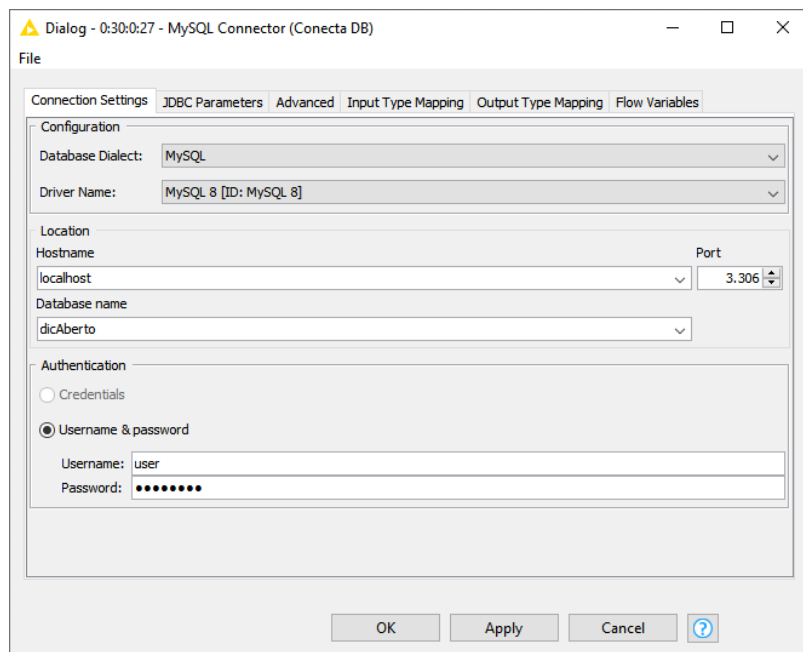


Figura 48 – Configuração do “*MySQL Connector*”

Depois de a conexão estar estabelecida, foi usado o “DB Insert” para inserir todas as palavras numa tabela na base de dados.

Para a sua configuração, foi necessário definir o nome da tabela e os campos a ser exportados.

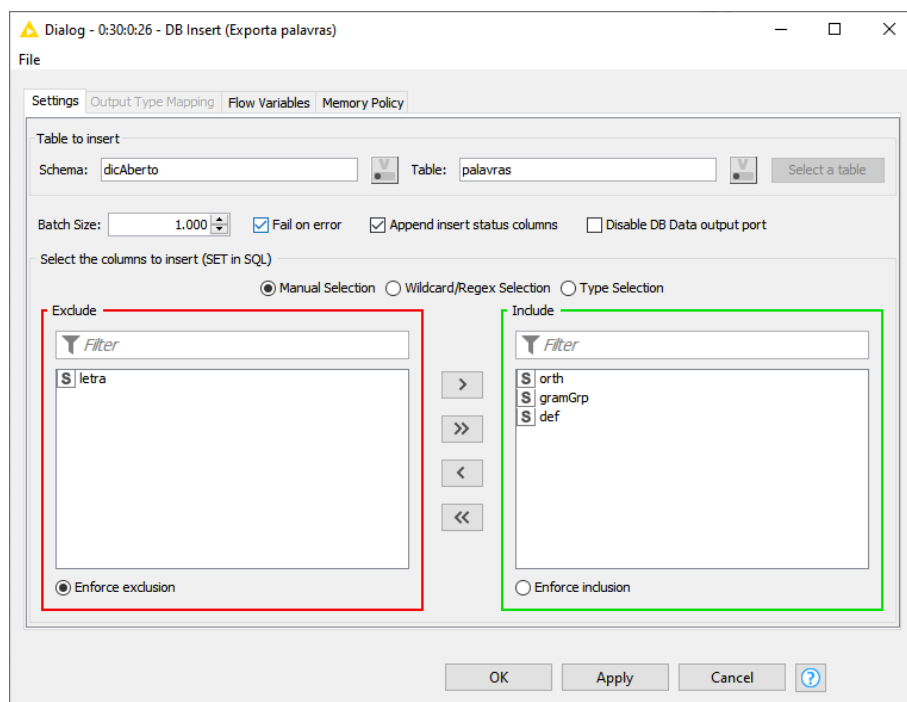


Figura 49 – Configuração do “DB Insert”

O componente “LetraALetra”, é responsável por separar para um ficheiro, todas as palavras iniciadas por uma determinada letra.

Para o componente saber quais as letras a serem exportadas, foi criado um ficheiro de texto onde o utilizador as pode escrever.

Este ficheiro é carregado para memória utilizando um “File Reader”.

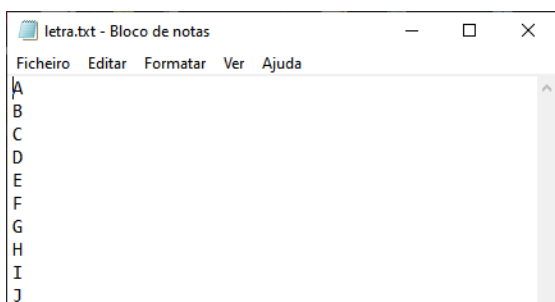


Figura 51 – Documento de texto que contém quais as letras a serem exportadas

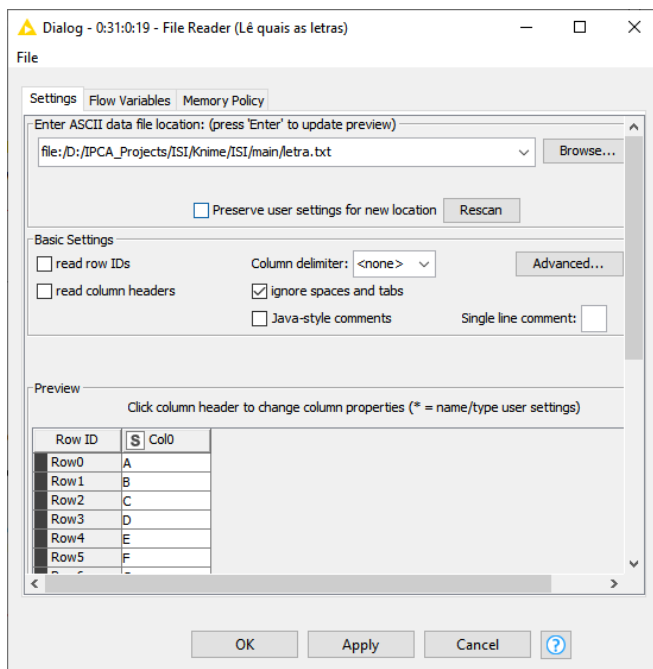


Figura 50 – Configuração do “File Reader”

Para percorrer cada letra a ser exportada, foi utilizado um “*Table Row To Variable Loop Start*” juntamente com um “*Variable Loop End*” que a cada iteração devolve uma letra através de uma variável.

Foi usado um “*Row Filter*”, que recebendo a letra, a compara ao valor da coluna que contem a letra inicial. Se for igual adiciona essa palavra ao resultado.

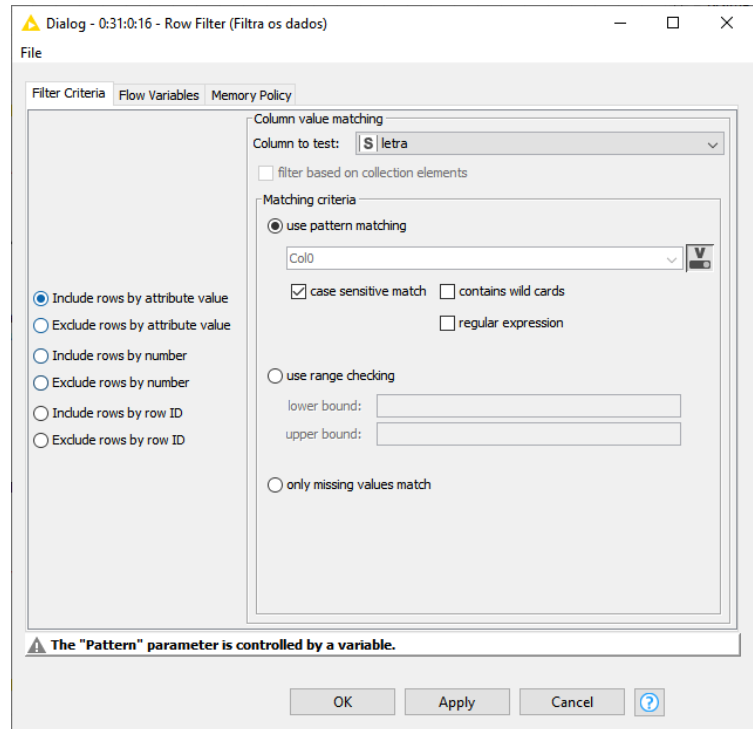


Figura 52 – Configuração do “*Row Filter*”

Para utilizar a variável recebida do *loop*, é necessário configurar, no separador das “*Flow Variable*” o campo onde esta variável irá ser preenchida.

Escolhi o campo “*pattern*” pois este é responsável por comparar o seu valor com a coluna selecionada (letra).

Coloquei como valor “*Col0*” porque é a variável que a cada iteração do *loop* contém a letra atual.

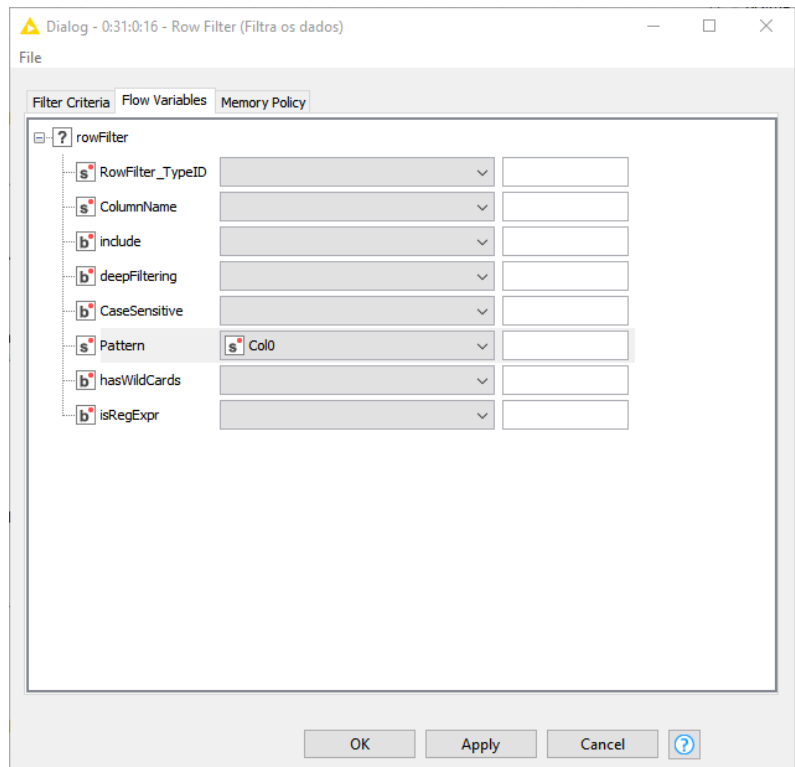


Figura 53 – Configuração de valores dinâmicos nos campos

Sabendo agora que por cada *loop* teremos cada uma das letras do ficheiro, e que por cada uma delas temos todas as palavras começadas por essa letra, é necessário exportar essas palavras para um ficheiro *JSON* e para um *CSV*. O nome dos ficheiros é a letra inicial mais a extensão corresponde (".json" e ".csv").

Na a exportação para *CSV*, utilizei como anteriormente, um "*CSV Writer*", mas como agora irão ser criados múltiplos ficheiros, foi utilizado um "*Java Edit Variable*" que, usando *java*, cria uma variável com a junção do caminho da pasta mais a letra atual do *loop* mais a extensão do ficheiro.

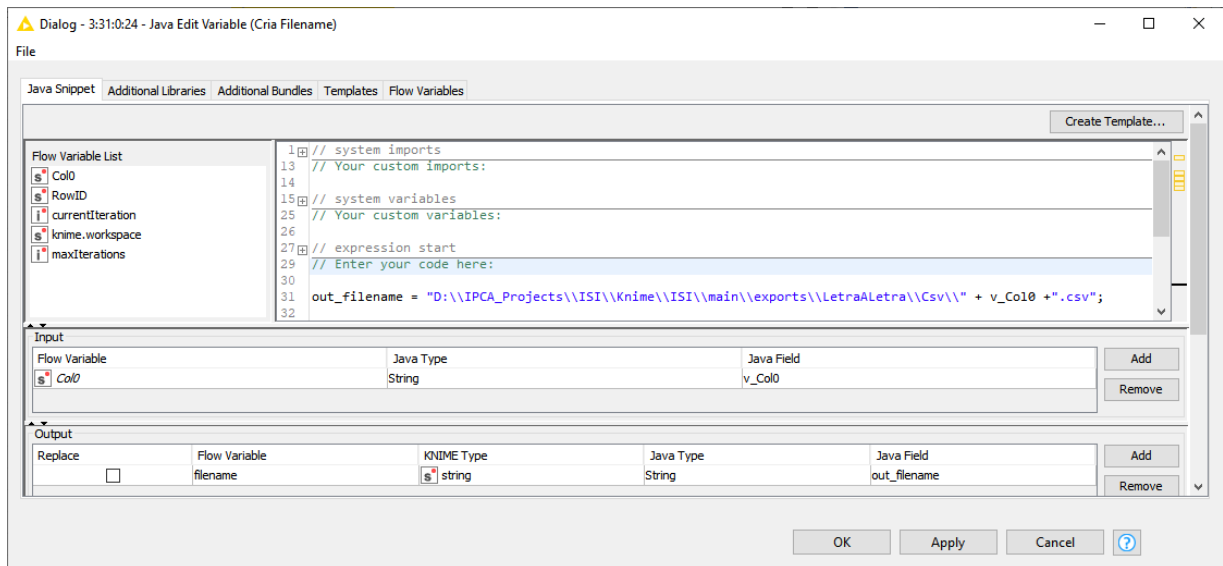


Figura 54 – Criação de um *Filename* dinâmico usando *Java*

Para usar esta variável com o caminho do ficheiro (da letra atual) a ser criado no "*CSV Writer*", esta foi colocada no campo "*filename*" do separador "*Flow Variables*".

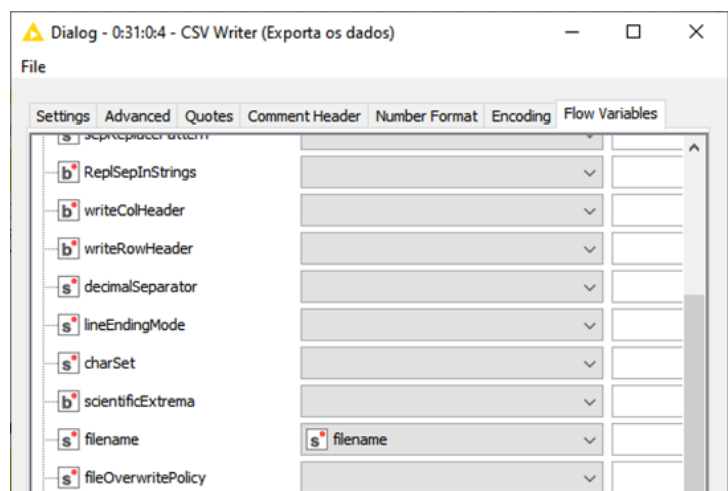


Figura 55 – Localização dinâmica da exportação das palavras letra a letra em *CSV*

Na a exportação para *JSON*, utilizei como anteriormente, um “*JSON Writer*”, mas como agora irão ser criados múltiplos ficheiros foi utilizada a combinação do “*Constant Value Column*” com o “*RowID*” que usando a variável com a letra atual do *loop* defino o nome do ficheiro.

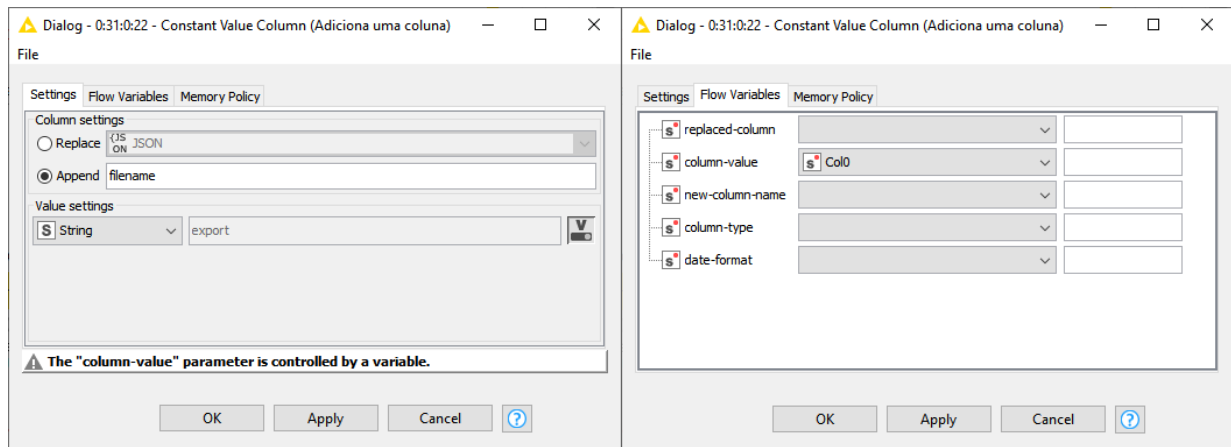


Figura 56 – Criação do nome dinâmico em *JSON*

O resultado da exportação letra a letra é a criação de duas pastas (“*Csv*” e “*Json*”) onde cada uma delas contém um ficheiro para cada letra, ficheiro este que está preenchido com as palavras que começam por aquela letra.

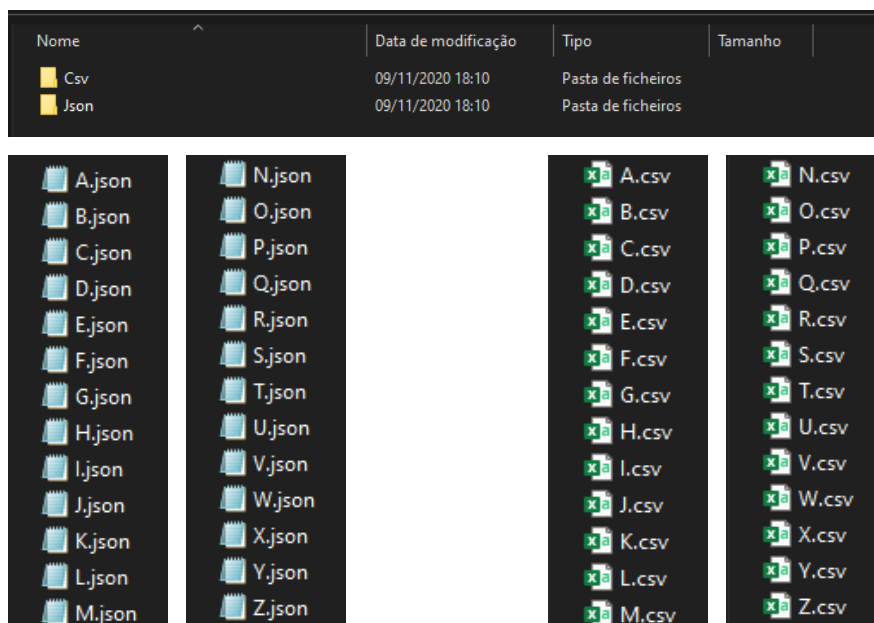


Figura 57 – Resultado da exportação letra a letra



## Conclusão

Durante este trabalho prático, como já mencionado anteriormente, foi-nos proposto consolidar, melhorar e adquirir conhecimentos relativos aos processos de *ETL* para a análise, processamento e importação/exportação de dados de/para diferentes sistemas. Assim sendo, foi através do desenvolvimento deste que tivemos a oportunidade de aplicar os conteúdos e matérias lecionadas durante as aulas. Com base nas mesmas, na pesquisa em grupo e no conhecimento já adquirido foi possível cumprir os objetivos.

Foi-nos muito fácil perceber que *ETL* aliadas com expressões regulares e linguagens de programação têm um grande potencial de processamento e são muito uteis para todo o tipo de trabalhos que envolvam manipulação de dados.

Para terminar, este trabalho prático foi uma peça essencial não só para o nosso desenvolvimento técnico e profissional, mas também para o nosso sucesso na Unidade Curricular.



## Bibliografia

Hitachi Vantara LLC. (s.d.). *9.1 - Pentaho Documentation*. Obtido de Hitachi Vantara - Data Storage and Analytics, DataOps, IoT, Cloud, Consulting: <https://help.pentaho.com/Documentation/9.1>

Nachum, M. (5 de Agosto de 2016). *Parsing Huge XML Files in Pentaho Kettle*. Obtido de Moran Nachum | Technical Tips: <https://morannachum.wordpress.com/2016/08/05/parsing-huge-xml-files-in-pentaho-kettle/>