

**Instituto Politécnico do Cávado e do Ave**

Integração de Sistemas de Informação

**Extract, Transform & Load – IMDB Movie Dataset**

Engenharia de Sistemas Informáticos

Duarte Ribeiro de Melo – 21149

Docente Óscar Ribeiro

Barcelos, Portugal

15 novembro de 2022

## Resumo e estrutura do documento

Com o avanço das tecnologias é de extrema importância o cuidado na manipulação, transformação, armazenamento e outras operações realizadas sobre dados, assim como a adaptação dos mesmos a novas tecnologias e, acima de tudo, garantir a flexibilidade e possibilidade de integrar os mesmos em soluções diferentes.

Serve então o presente relatório para demonstrar o trabalho desenvolvido pelo aluno Duarte Ribeiro de Melo, relativo ao Trabalho Prático 01 da Unidade Curricular de Integração de Sistemas de Informação, que toca nos temas mencionados supra.

Numa primeira fase do documento consta uma introdução ao tema, objetivos do projeto, demonstração do *dataset* escolhido e das ferramentas escolhidas pelo aluno.

Numa segunda fase está presente a explicação da solução mais detalhada, tanto a parte realizada no programa *Pentaho Kettle*, em *Python* ou em *AngularJS*.

Por fim, alguns tópicos que visam dar como concluído o projeto e algumas considerações finais.

## Índice

Resumo e estrutura do documento .....	2
Índice de Figuras .....	4
Introdução .....	5
Objetivos .....	6
Dados utilizados no processo ETL .....	7
Ferramentas utilizadas .....	9
Pentaho Data Integration – Kettle .....	9
Python com framework Flask .....	10
AngularJS .....	11
Git e GitHub .....	12
Solução .....	13
Solução em Pentaho Data Integration (Kettle) .....	13
Solução em Python com Flask (Web API) .....	21
Solução em AngularJS .....	22
Conclusão .....	23
Referências .....	24

## Índice de Figuras

Figura 1 - Exemplo de registos dataset title.basics.tsv.gz .....	8
Figura 2 - Exemplo de registos dataset title.ratings.tsv.gz.....	8
Figura 3 - Alteração memória máxima usada pelo Pentaho Kettle .....	9
Figura 4 - Exemplo de utilização do Pentaho Data Integration (Kettle) no projeto .....	9
Figura 5 - Exemplo de utilização de Python com Flask no projeto .....	10
Figura 6 - Exemplo de utilização de AngularJS no projeto .....	11
Figura 7 - Exemplo de utilização do Git no projeto.....	12
Figura 8 - Exemplo de utilização do GitHub no projeto .....	12
Figura 9 - Step CSV Input no PDI .....	13
Figura 10 - Step Replace in String no PDI .....	13
Figura 11 - Step Split fields no PDI .....	14
Figura 12 - Extração ratings e agrupamento dos dados.....	14
Figura 13 - Step Merge join no PDI .....	14
Figura 14 - Step If field value is null no PDI .....	15
Figura 15 - Ordenação dos dados e switch/case no PDI .....	16
Figura 16 - Step Sort rows no PDI .....	17
Figura 17 - Step Switch / case no PDI.....	17
Figura 18 - Step MongoDB output no PDI .....	18
Figura 19 - Step MongoDB output no PDI .....	19
Figura 20 - Step MongoDB no PDI .....	19
Figura 21 – transformação data_transformation no PDI .....	20
Figura 22 - job job_data_download no PDI .....	20
Figura 23 - job main no PDI .....	20
Figura 24 - Web API Python.....	21
Figura 25 - Dashboard AngularJS .....	22
Figura 26 - Dashboard AngularJS Pesquisa.....	22

## Introdução

Nos dias que correm, os nossos dados estão presentes em sistemas informáticos relacionados com as áreas mais sensíveis e importantes da nossa vida, como a saúde, educação, segurança, entre outros. É imprescindível, mas exigente, garantir que estes dados se mantêm coesos, seguros, atualizados e passíveis de serem utilizados em sistemas em constante atualização, que sofrem reestruturações e reformulações com uma rapidez nunca antes vista.

Sistemas informáticos da área da saúde, banca, entre outras que se prendiam por sistemas antigos pela sua segurança e dificuldade de atualização dos mesmos, veem agora a necessidade de se adaptar a um ecossistema totalmente diferente, onde os dados devem estar acessíveis num smartphone, numa caixa de multibanco, num computador, num tablet ou num terminal de pagamento, permitindo que sejam realizadas operações entre estes diferentes sistemas mencionados.

Resumindo, é um grande tópico da área informática a segurança, coesão e flexibilidade dos dados, assim como a grande necessidade de disponibilização dos mesmos em formatos transversais a diversas soluções.

Surge então o tema da integração dos dados, abordado nesta Unidade Curricular, ao qual pertencem os processos de ETL – *Extract, Transform and Load* – extração, transformação e carregamento.

Com base nisto, optou-se pela utilização do *software* Pentaho Kettle, no qual se desenvolveu a abordagem às três letras do ETL, e ainda a utilização de outras ferramentas para a análise e visualização dos resultados obtidos.

## Objetivos

O objetivo principal deste projeto é a extração e transformação de dados provenientes de um *dataset*, nomeadamente o da *Internet Movie Database* (IMDB), e consequente carregamento dos dados noutros sistemas, neste caso, numa base de dados MongoDB – todo este processo é realizado no Pentaho Kettle. Após este carregamento, deve ser possível a visualização dos dados num *dashboard* desenvolvido em *AngularJS*, que consegue obter os dados presentes na base de dados via web API desenvolvida em *Flask*, framework de *Python*.

Foram escolhidos dois ficheiros presentes no *dataset*, nomeadamente o *title.basics.tsv.gz* e o *title.ratings.tsv.gz*, que foram agrupados e tratados na ferramenta de ETL. O resultado do agrupamento e tratamento dos dados destes dois documentos é então exportado para a base de dados MongoDB, para documentos XML e para um ficheiro JSON – separado por tipo de título (filme, curta, episódios, etc.).

## Dados utilizados no processo ETL

Para que a realização deste trabalho fosse possível era necessário escolher um *dataset* disponível na Internet, de preferência com alguma complexidade, de forma a permitir algumas operações sobre o mesmo. Assim sendo, foi escolhido o *dataset* disponibilizado pelo IMDB no seguinte URL - <https://datasets.imdbws.com/> - este tem mais de 10 milhões de registos em alguns ficheiros e diversos campos relativos a produções audiovisuais.

Destes dados disponibilizados pelo IMDB, foram apenas utilizados dois ficheiros - o *title.basics.tsv.gz* e o *title.ratings.tsv.gz* – o primeiro contém informação geral dos *titles* (*titles* incluem qualquer tipo de produção audiovisual – filmes, séries, episódios, vídeos, etc. – presente no IMDB) e o segundo contém informação sobre as avaliações atribuídas pelos utilizadores da plataforma IMDB aos *titles* visualizados por estes.

É também importante mencionar que a descompactação destes dados é realizada pelo próprio *Kettle*, dado que a extração é feita pelo mesmo, diretamente do URL acima.

Passando à explicação dos campos presentes nos ficheiros obtidos do *dataset*, no *title.basics.tsv.gz*:

- *tconst* – identificador único de cada *title* (produção audiovisual presente no IMDB)
- *titleType* – o tipo de produção (filme, curta metragem, série, etc.)
- *primaryTitle* – o título mais usado/comum para a produção
- *originalTitle* – o título original, na linguagem original, da produção
- *isAdult* – indica se o filme é para adultos (valor 1) ou não (valor 0)
- *startYear* – ano de lançamento, no caso de ser uma série, o ano em que começou
- *endYear* – só se aplica nas séries e é o ano de término, tem o valor de ‘\N’ para outro tipo de produções
- *runtimeMinutes* – duração da produção em minutos
- *géneros* – até três géneros associados à produção (Drama, Biografia, etc.)

No *title.ratings.tsv.gz*:

- *tconst* – identificador único de cada *title* (produção audiovisual) ao qual se refere este *rating*/avaliação
- *averageRating* – valor médio das avaliações
- *numVotes* – número de votos por parte dos utilizadores em relação a esta produção

É importante mencionar que no caso em que não exista informação para algum destes campos, é encontrado um ‘\N’ no mesmo.

Exemplo de registos destas tabelas:

Quarte Miedo > Downloads > title.basics.tsv-1 > data.bv

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance
tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	12	Animation,Short
tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	Comedy,Short
tt0000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	\N	1	Short
tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph	0	1894	\N	1	Short,Sport
tt0000008	short	Edison Kinetoscopic Record of a Sneeze	Edison Kinetoscopic Record of a Sneeze	0	1894	\N	1	Documentary,Short
tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	\N	45	Romance
tt0000010	short	Leaving the Factory	La sortie de l'usine Lumière	0	1895	\N	1	Documentary,Short
tt0000011	short	Akrobatisches Potpourri	Akrobatisches Potpourri	0	1895	\N	1	Documentary,Short
tt0000012	short	The Arrival of a Train	L'arrivée d'un train à La Ciotat	0	1896	\N	1	Documentary,Short
tt0000013	short	The Photographical Congress Arrives in Lyon	Le débarquement du congrès de photographie à Lyon	0	1895	\N	1	Documentary,Short
tt0000014	short	The Waterer Watered	L'arroseur arrosé	0	1895	\N	1	Comedy,Short
tt0000015	short	Autour d'une cabine	Autour d'une cabine	0	1894	\N	2	Animation,Short
tt0000016	short	Boat Leaving the Port	Barque sortant du port	0	1895	\N	1	Documentary,Short
tt0000017	short	Italienischer Bauerntanz	Italienischer Bauerntanz	0	1895	\N	1	Documentary,Short
tt0000018	short	Das boxende Känguruh	Das boxende Känguruh	0	1895	\N	1	Short
tt0000019	short	The Clown Barber	The Clown Barber	0	1898	\N	\N	Comedy,Short
tt0000020	short	The Derby 1895	The Derby 1895	0	1895	\N	1	Documentary,Short,Sport
tt0000022	short	Blacksmith Scene	Les forgerons	0	1895	\N	1	Documentary,Short
tt0000023	short	The Sea Baignade en mer		0	1895	\N	1	Documentary,Short
tt0000024	short	Opening of the Kiel Canal	Opening of the Kiel Canal	0	1895	\N	\N	News,Short
tt0000025	short	The Oxford and Cambridge University Boat Race	The Oxford and Cambridge University Boat Race	0	1895	\N	\N	News,Short,Sport

Figura 1 - Exemplo de registos dataset title.basics.tsv.gz

Quarte Miedo > Downloads > title.ratings.tsv > data.t

tconst	averageRating	numVotes
tt0000001	5.7	1922
tt0000002	5.8	259
tt0000003	6.5	1735
tt0000004	5.6	174
tt0000005	6.2	2548
tt0000006	5.1	175
tt0000007	5.4	797
tt0000008	5.4	2064
tt0000009	5.3	200
tt0000010	6.9	6970
tt0000011	5.3	356
tt0000012	7.4	11958
tt0000013	5.7	1840
tt0000014	7.1	5366
tt0000015	6.2	1031
tt0000016	5.9	1453
tt0000017	4.6	317
tt0000018	5.3	582
tt0000019	5.1	31
tt0000020	4.8	348
tt0000022	5.1	1068
tt0000023	5.7	1397
tt0000024	4.2	101
tt0000025	3.9	45
tt0000026	5.7	1507
tt0000027	5.6	1126
tt0000028	5.1	1043
tt0000029	5.9	3288
tt0000030	5.2	826
tt0000031	5.5	981
tt0000032	5.0	409
tt0000033	5.5	1002

Figura 2 - Exemplo de registos dataset title.ratings.tsv.gz



## Ferramentas utilizadas

Neste tópico serão abordadas as ferramentas utilizadas para o desenvolvimento do projeto, assim como uma breve descrição das mesmas e em que sentido foram necessárias para desenvolver este trabalho.

### Pentaho Data Integration – Kettle

O Pentaho Data Integration (PDI) providenciou ferramentas para a realização dos processos ETL – *Extract, Transform and Load*. Estas facilitaram o processo de extração, correção e armazenamento dos dados utilizando um sistema de simples compreensão, sem necessidade de escrever código e de rápida implementação.

Para que este programa corra, é necessário ter o Java instalado no computador, assim como definir a variável do sistema JAVA\_HOME.

No caso específico das transformações e *jobs* desenvolvidos neste trabalho, dado o grande volume da dados, foi necessário expandir a memória RAM máxima utilizada pelo programa para 8GB:

```
REM
if "%PENTAHO_DI_JAVA_OPTIONS%"==" " set PENTAHO_DI_JAVA_OPTIONS="-Xms4096m" "-Xmx8192m"
```

Figura 3 - Alteração memória máxima usada pelo Pentaho Kettle

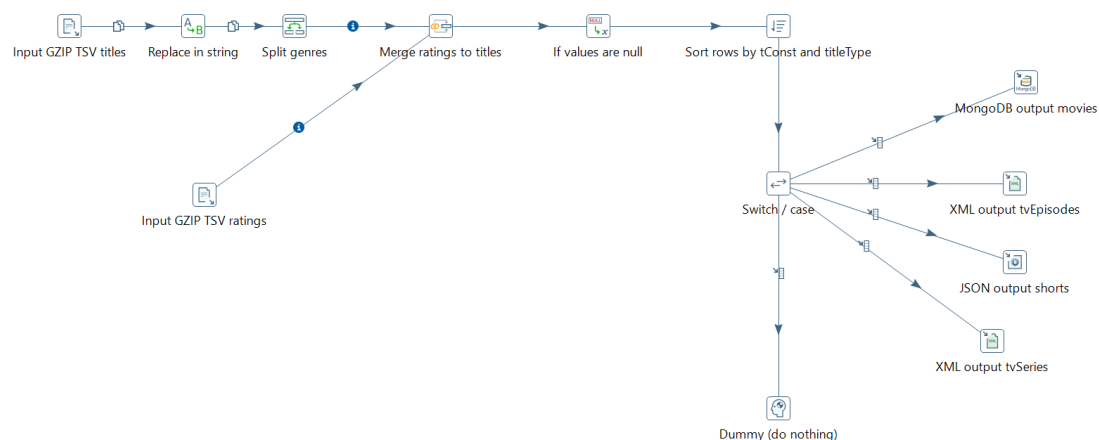


Figura 4 - Exemplo de utilização do Pentaho Data Integration (Kettle) no projeto

## Python com framework Flask

Python é uma linguagem de programação de alto nível, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Prioriza a legibilidade do código sobre a velocidade.

Neste projeto, utilizando uma *framework web* chamada Flask, permitiu a criação de uma pequena *web API* com um só *request GET* que devolve os valores dos *movies* presentes numa base de dados MongoDB.

Outros *packages* Python usados: pymongo, json

```
# CORS

@app.after_request
def after_request(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type, Authorization')
    response.headers.add('Access-Control-Allow-Methods', 'GET, PUT, POST, PATCH, DELETE')
    return response

# GET TITLES
@app.route("/titles", methods=["GET"])
def get_titles():
    try:
        data = list(db.imdb_isi.find())
        # for title in data:
        #     title["_id"] = str(title["_id"])
        return Response(
            response = json.dumps(data),
            status=200,
            mimetype="application/json"
        )
    except Exception as ex:
        print(ex)
        return Response(
            response= json.dumps(
                {"message": "cannot read titles"}
            ),
            status=500,
            mimetype="application/json"
        )

####

if __name__ == "__main__":
    app.run(debug=True)
```

Figura 5 - Exemplo de utilização de Python com Flask no projeto

## AngularJS

AngularJS é um *framework* de JavaScript de código aberto, mantido pela Google, que auxilia na execução de *single-page applications*. A biblioteca distingue-se por permitir declarar *dynamic views* em *web apps*.

Neste caso específico, utilizou-se Angular com o objetivo de criar um *dashboard* com uma tabela que possibilita visualizar os dados obtidos via a *web API* criada em Python.

```
Dashboard_Angular > src > app > view-titles > view-titles.component.html > table.table > thead > tr > th.buttons-header
You, 4 days ago | 1 author (You)
1 <table class="table">
2   <thead>
3     <tr>
4       <th scope="col">#</th>
5       <th scope="col">._id</th>
6       <th scope="col">titleType</th>
7       <th scope="col">primaryTitle</th>
8       <th scope="col">originalTitle</th>
9       <th scope="col">isAdult</th>
10      <th scope="col">startYear</th>
11      <th scope="col">endYear</th>
12      <th scope="col">runtimeMinutes</th>
13      <th scope="col">genre_1</th>
14      <th scope="col">genre_2</th>
15      <th scope="col">genre_3</th>
16      <th scope="col">averageRating</th>
17      <th scope="col">numVotes</th>
18      <th class="buttons-header"> You, 4 days ago • Angular Dashboard possibly done
19      <input type="text" placeholder="Pesquisar" [(ngModel)]="filterText">
20    </th>
21  </tr>
22 </thead>
23 <tbody>
24   <tr *ngFor="let title of titles | filter:filterText; index as i">
25     <th scope="row">{{i+1}}</th>
26     <td>{{title._id}}</td>
27     <td>{{title.titleType}}</td>
28     <td>{{title.primaryTitle}}</td>
29     <td>{{title.originalTitle}}</td>
30     <td>{{title.isAdult}}</td>
31     <td>{{title.startYear}}</td>
32     <td>{{title.endYear}}</td>
33     <td>{{title.runtimeMinutes}}</td>
34     <td>{{title.genre_1}}</td>
35     <td>{{title.genre_2}}</td>
36     <td>{{title.genre_3}}</td>
37     <td>{{title.averageRating}}</td>
38     <td>{{title.numVotes}}</td>
39   </tr>
```

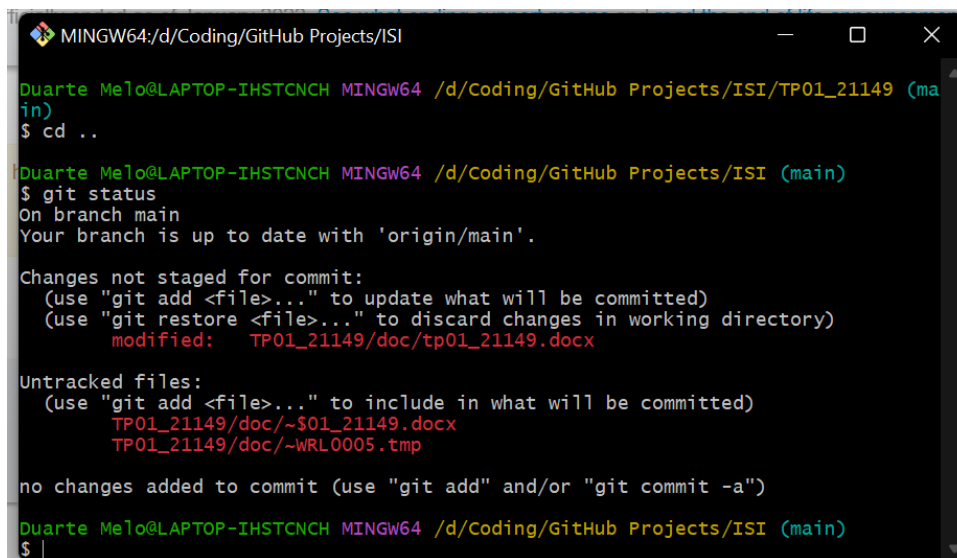
Figura 6 - Exemplo de utilização de AngularJS no projeto

## Git e GitHub

Git é um sistema de controlo de versões, usado principalmente no desenvolvimento de *software*, mas que pode ser utilizado para registar o histórico de edições de qualquer tipo de ficheiro. Foi projetado por Linus Torvalds, mas entretanto adotado mundialmente.

GitHub é uma plataforma de hosting de código-fonte e arquivos que faz uso do Git para gerir as versões dos mesmos. Permite a criação de repositórios públicos e privados e a colaboração entre programadores nos mesmos.

Neste projeto, foram utilizadas ambas as ferramentas, o Git para a realização de *commits* com alterações e novas versões do projeto e o GitHub para a hospedagem do mesmo na *web*.



```
MINGW64:/d/Coding/GitHub Projects/ISI
Duarte Melo@LAPTOP-IHSTCNCH MINGW64 /d/Coding/GitHub Projects/ISI/TP01_21149 (main)
$ cd ..
Duarte Melo@LAPTOP-IHSTCNCH MINGW64 /d/Coding/GitHub Projects/ISI (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   TP01_21149/doc/tp01_21149.docx

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        TP01_21149/doc/~$01_21149.docx
        TP01_21149/doc/~WRL0005.tmp

no changes added to commit (use "git add" and/or "git commit -a")
Duarte Melo@LAPTOP-IHSTCNCH MINGW64 /d/Coding/GitHub Projects/ISI (main)
$
```

Figura 7 - Exemplo de utilização do Git no projeto

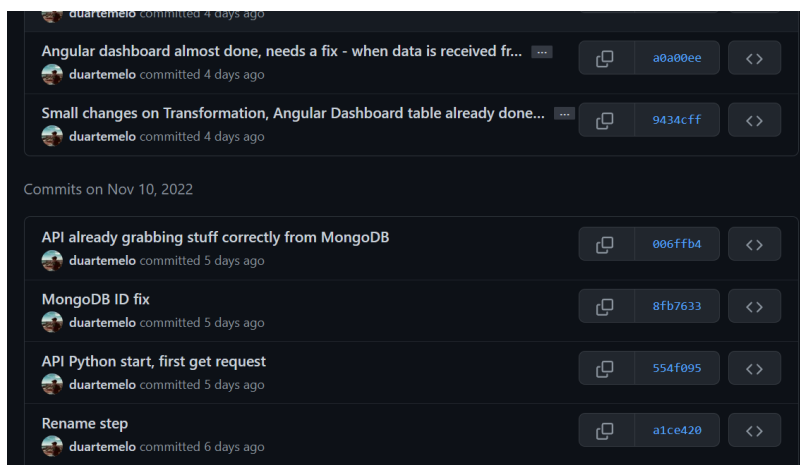


Figura 8 - Exemplo de utilização do GitHub no projeto

## Solução

No presente tópico será explicado o desenvolvimento da solução, as opções tomadas e as diferentes fases da criação do projeto.

É importante salientar que os ficheiros criados pelo aluno no PDI e o código desenvolvido em Python e AngularJS podem ser encontrados no repositório GitHub ou na entrega realizada pelo Moodle.

### Solução em Pentaho Data Integration (Kettle)

Após escolha do *dataset*, o aluno procedeu à visualização de alguns exercícios realizados nas aulas em Kettle de forma a relembrar alguns tópicos abordados e poder iniciar o desenvolvimento da transformação.

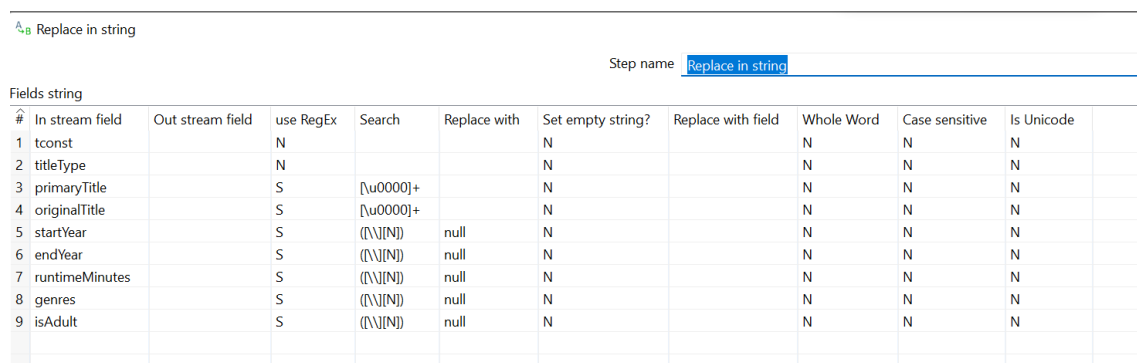
Numa segunda fase, o aluno procedeu à realização da transferência dos ficheiros do *dataset* para a sua máquina e começou a trabalhar com eles localmente (sem utilizar um *step* HTTP para obtenção dos dados). Dado que os ficheiros são TSV (*tab separated values*) e não CSV (*comma separated values*), foi necessário alterar o valor do *delimiter* no Pentaho Data Integration para TAB, apesar de se usar na mesma o *step* do input CSV.

Dado que o ficheiro vem do *dataset* comprimido em .gz, foi também necessário usar o *step* “*gzip CSV Input*”, em vez do normal “*CSV Input*”. Assim sendo, a descompactação é feita pelo próprio PDI.



Figura 9 - Step CSV Input no PDI

Posteriormente, procedeu-se à substituição de valores ‘\N’ por ‘null’ e remoção de caracteres inválidos utilizando o *step* “*Replace in String*”.



#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?	Replace with field	Whole Word	Case sensitive	Is Unicode
1	tconst		N			N		N	N	N
2	titleType		N			N		N	N	N
3	primaryTitle		S	[\\u0000]+		N		N	N	N
4	originalTitle		S	[\\u0000]+		N		N	N	N
5	startYear		S	(\\N [N])	null	N		N	N	N
6	endYear		S	(\\N [N])	null	N		N	N	N
7	runtimeMinutes		S	(\\N [N])	null	N		N	N	N
8	genres		S	(\\N [N])	null	N		N	N	N
9	isAdult		S	(\\N [N])	null	N		N	N	N

Figura 10 - Step Replace in String no PDI

Após isto, o aluno procedeu à separação da coluna com o *array* de géneros [gen\_1, gen\_2, gen\_3] em três diferentes colunas (genre\_1, genre\_2, genre\_3) utilizando o step “Split fields”.

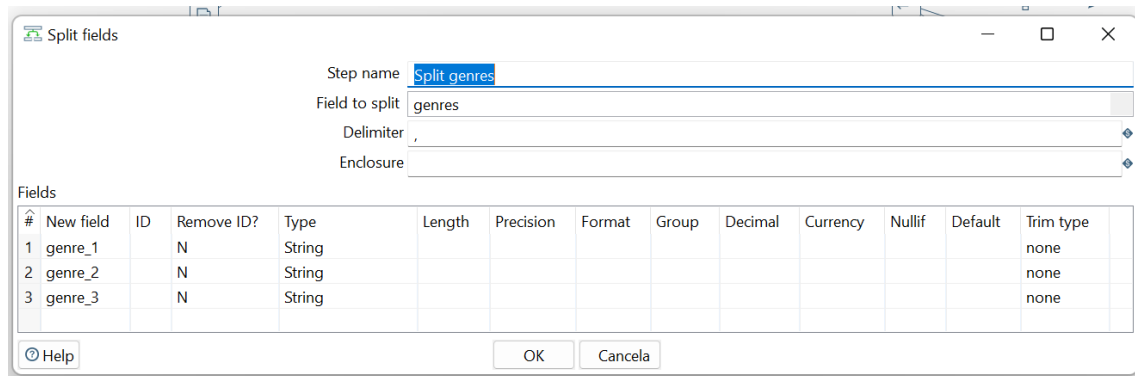


Figura 11 - Step Split fields no PDI

De seguida, e após extração dos dados presentes no *title.ratings.tsv.gz*, estes mesmos dados foram agrupados à anteriormente extraída tabela *title.basics.tsv.gz*, de forma a conseguir juntar os *ratings* dos filmes aos mesmos. Para tal, foi utilizado o step “Merge join”.

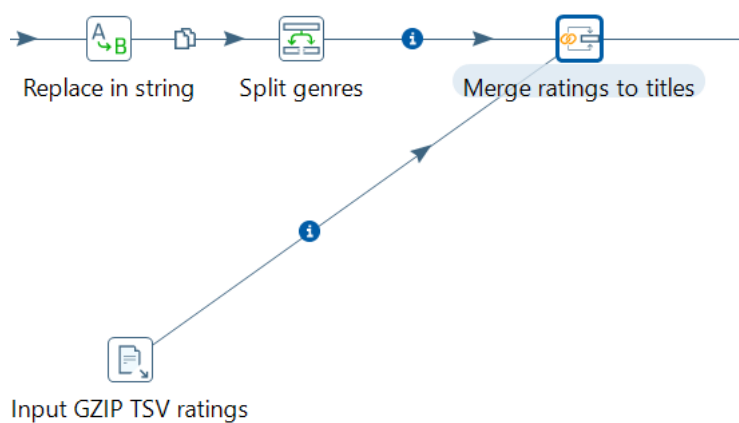


Figura 12 - Extração ratings e agrupamento dos dados

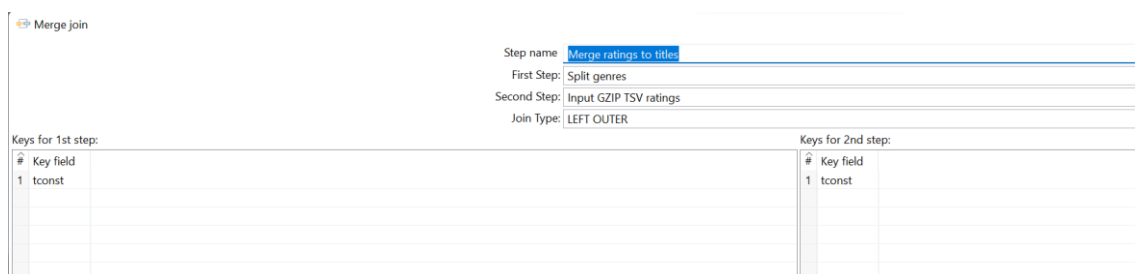


Figura 13 - Step Merge join no PDI

Sucessivamente, foi utilizado um *step* com objetivo de reforçar que qualquer campo que tivesse o valor *null*, fosse substituído pela *string* “*null*”, de forma a manter a coesão dos dados nos diferentes carregamentos finais.

[illegible]

Figura 14 - Step If field value is null no PDI

Mais tarde, os dados foram ordenados pelo seu identificador e tipo de produção. Depois, foi utilizado um *switch/case* para que consoante o *titleType* (tipo de produção) os dados fossem distribuídos para diferentes carregamentos.

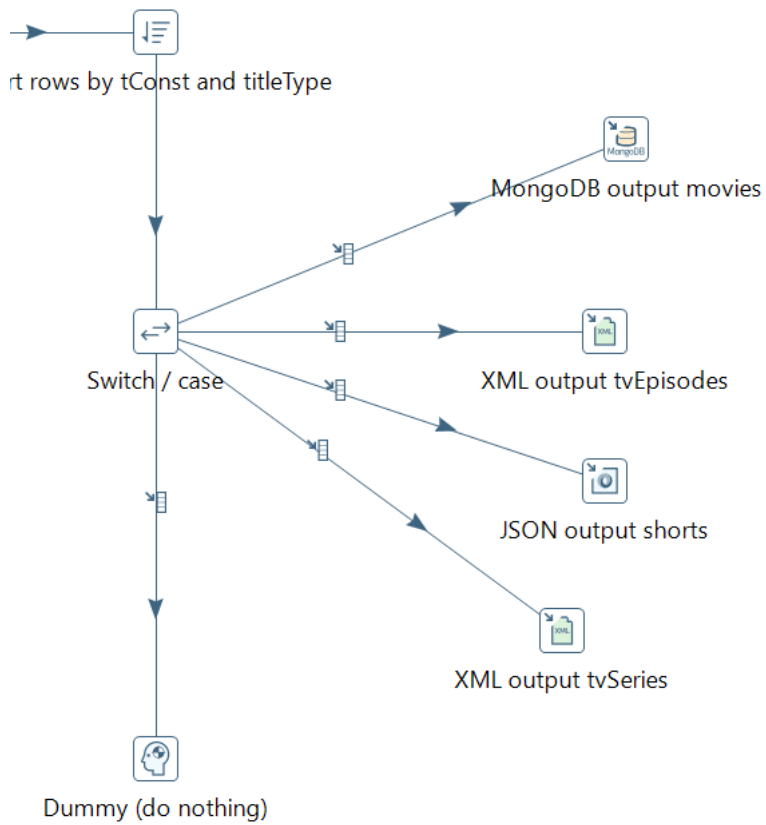


Figura 15 - Ordenação dos dados e switch/case no PDI



Sort rows

Nome do Step: Sort rows by tConst and titleType

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) ☐

Fields :

#	Fieldname	Ascending	Case sensitive compare?	Sort based on current locale?	Collator Strength	Presorted?
1	tconst	S	N	N	0	N
2	titleType	S	N	N	0	N
3	primaryTitle	N	N	N	0	N
4	originalTitle	N	N	N	0	N
5	isAdult	N	N	N	0	N
6	startYear	N	N	N	0	N
7	endYear	N	N	N	0	N
8	runtimeMinutes	N	N	N	0	N
9	genre_1	N	N	N	0	N
1.	genre_2	N	N	N	0	N
1.	genre_3	N	N	N	0	N
1.	tconst_1	N	N	N	0	N
1.	averageRating	N	N	N	0	N
1.	numVotes	N	N	N	0	N

Figura 16 - Step Sort rows no PDI

Switch / case

Step name: Switch / case

Field name to switch: titleType

Use string contains comparison: ☐

Case value data type: None

Case value conversion mask:

Case value decimal symbol:

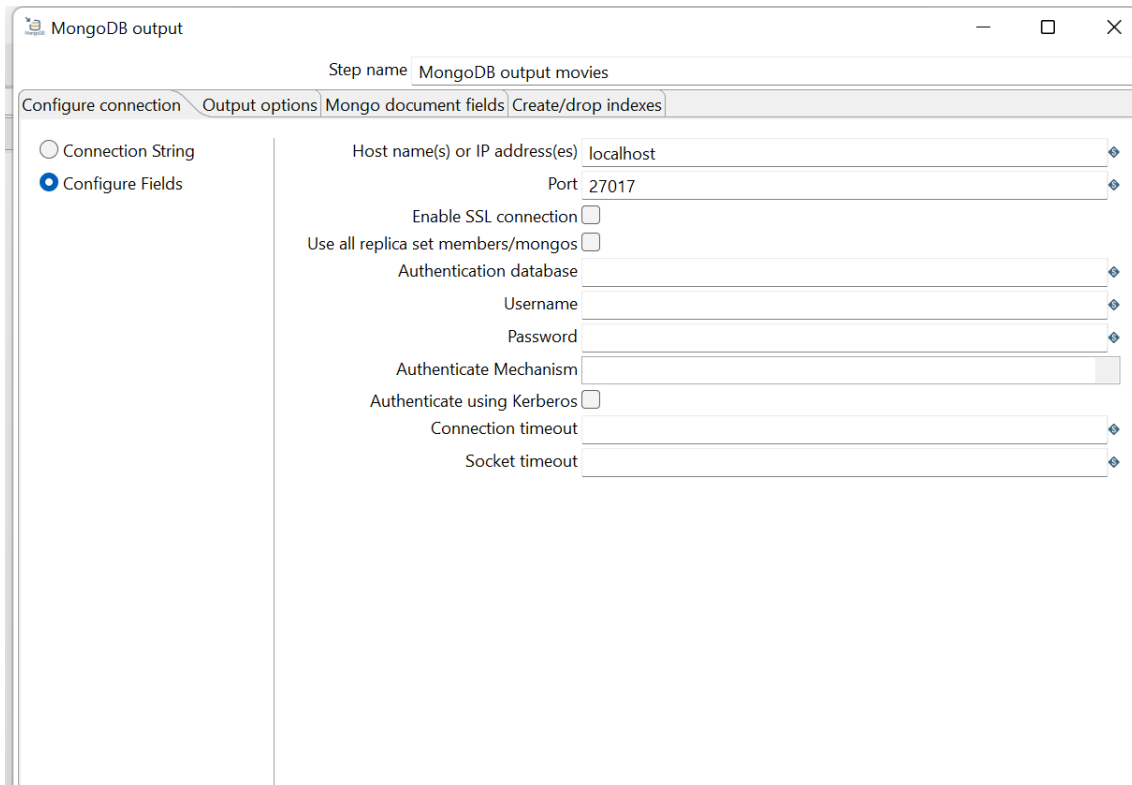
Case value grouping symbol:

#	Value	Target step
1	movie	MongoDB output movies
2	tvEpisode	XML output tvEpisodes
3	short	JSON output shorts
4	tvSeries	XML output tvSeries

Figura 17 - Step Switch / case no PDI

Por fim, em cada tipo de carregamento diferente, foram escolhidas as colunas que eram pertinentes carregar para o destino final e selecionadas algumas opções para que este carregamento fosse feito da forma pretendida.

No caso do carregamento para a base de dados MongoDB, foi ainda necessário configurar a conexão à mesma.



The screenshot shows the 'MongoDB output' configuration window. The 'Step name' is 'MongoDB output movies'. The 'Configure Fields' tab is selected. On the left, 'Configure Fields' is selected with a radio button. The main area contains the following fields and options:

- Host name(s) or IP address(es): localhost
- Port: 27017
- Enable SSL connection: ☐
- Use all replica set members/mongos: ☐
- Authentication database:
- Username:
- Password:
- Authenticate Mechanism:
- Authenticate using Kerberos: ☐
- Connection timeout:
- Socket timeout:

Figura 18 - Step MongoDB output no PDI

MongoDB output

Step name: MongoDB output movies

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

Database: local Get DBs

Collection: imdb\_isi Get collections

Batch insert size: 100

Truncate collection: ☒

Update: ☐

Upsert: ☐

Multi-update: ☐

Modifier update: ☐

Number of retries for write operations: 5

Delay, in seconds, between retry attempts: 10

Write concern (w option): Get custom write concerns

w Timeout:

Journalized writes: ☐

Read preference: primary

Figura 19 - Step MongoDB output no PDI

MongoDB output

Step name: MongoDB output movies

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

#	Name	Mongo document path	Use field name	NULL values	JSON	Match field for update	Modifier operation	Modifier policy
1	tconst	_id	N	Ignore	N	N	N/A	Insert&Update
2	titleType		Y	Ignore	N	N	N/A	Insert&Update
3	primaryTitle		Y	Ignore	N	N	N/A	Insert&Update
4	originalTitle		Y	Ignore	N	N	N/A	Insert&Update
5	isAdult		Y	Ignore	N	N	N/A	Insert&Update
6	startYear		Y	Ignore	N	N	N/A	Insert&Update
7	endYear		Y	Ignore	N	N	N/A	Insert&Update
8	runtimeMinutes		Y	Ignore	N	N	N/A	Insert&Update
9	genre_1		Y	Ignore	N	N	N/A	Insert&Update
1.	genre_2		Y	Ignore	N	N	N/A	Insert&Update
1.	genre_3		Y	Ignore	N	N	N/A	Insert&Update
1.	averageRating		Y	Ignore	N	N	N/A	Insert&Update
1.	numVotes		Y	Ignore	N	N	N/A	Insert&Update

Figura 20 - Step MongoDB no PDI

A transformação final:

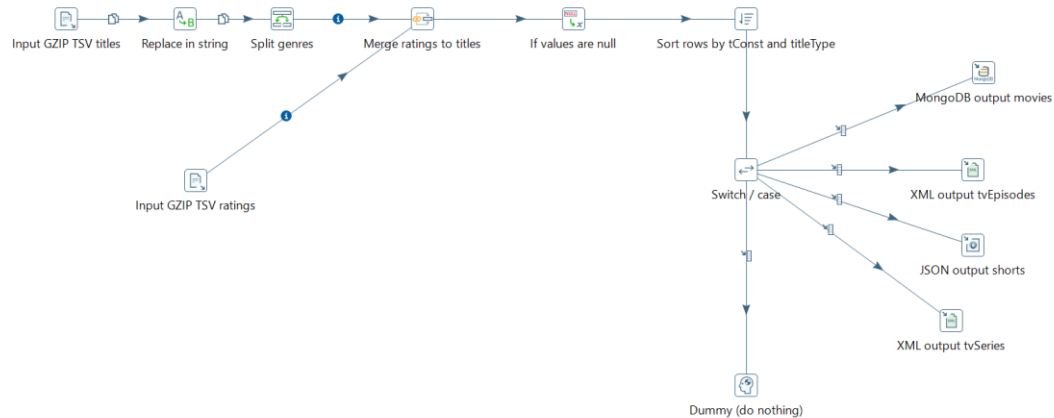


Figura 21 – transformação data\_transformation no PDI

Após ter terminado a transformação no PDI, o aluno procedeu à criação de dois *jobs*: o primeiro – *job\_data\_download* – visa extrair os ficheiros do *dataset* utilizando um *web service*, com um *step* HTTP; o segundo – *main* – visa criar toda a sequência do processo ETL, desde a definição de variáveis para a transferência dos ficheiros até ao email de sucesso. No caso de alguma transformação ou *job* falhar, são enviados emails de insucesso ao utilizador, e antes de terminar toda esta sequência, os ficheiros de *input* são apagados.



Figura 22 - job job\_data\_download no PDI

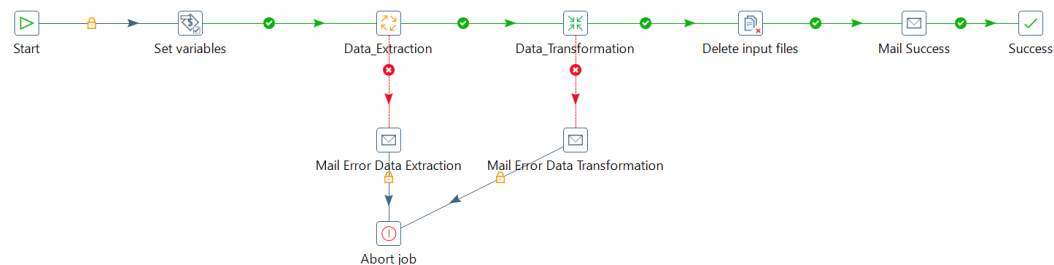


Figura 23 - job main no PDI

## Solução em Python com Flask (Web API)

Após o término do desenvolvimento dos *jobs* e *transformations* no Pentaho Data Integration (Kettle), o aluno quis desenvolver uma *dashboard* na qual fosse possível visualizar os dados carregados pelo Kettle para o MongoDB de uma forma mais agradável ao utilizador. Para tal, desenvolveu uma *web API* simples em Python, utilizando uma *framework* chamada Flask, que com apenas um *request GET*, consegue fazer chegar os dados a qualquer tipo de aplicação que consiga fazer *requests* a uma API, no caso, uma *web app* em AngularJS.

```
try:
    mongo = pymongo.MongoClient(
        host="localhost",
        port=27017,
        serverSelectionTimeoutMS = 1000
    )
    db = mongo.local
    mongo.server_info() # trigger exception if cannot connect to db
except:
    print("ERRORR - Cannot connect to db")

# CORS

@app.after_request
def after_request(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type, Authorization')
    response.headers.add('Access-Control-Allow-Methods', 'GET, PUT, POST, PATCH, DELETE')
    return response

# GET TITLES
@app.route("/titles", methods=["GET"])
def get_titles():
    try:
        data = list(db.imdb_isi.find())
        # for title in data:
        #     title["_id"] = str(title["_id"])
        return Response(
            response = json.dumps(data),
            status=200,
            mimetype="application/json"
        )
    except Exception as ex:
        print(ex)
        return Response(
            response= json.dumps(
                {"message": "cannot read titles"}
            ),
            status=500,
            mimetype="application/json"
        )
```

Figura 24 - Web API Python

## Solução em AngularJS

Por fim, e como mencionado anteriormente, o aluno desenvolveu um *dashboard* simples, com apenas uma tabela, onde é possível visualizar a informação de 100 registos obtidos via *request* e pesquisar pelos campos dos mesmos.

#	_id	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genre_1	genre_2	genre_3	averageRating	numVotes	Pesquisar
1	tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	null	45	Romance	null	null	5.2	200	
2	tt0000502	movie	Bohemios	Bohemios	0	1905	null	100	null	null	null	4.2	14	
3	tt0000574	movie	The Story of the Kelly Gang	The Story of the Kelly Gang	0	1906	null	70	Action	Adventure	Biography	6.0	794	
4	tt0000591	movie	The Prodigal Son	L'enfant prodigue	0	1907	null	90	Drama	null	null	5.1	20	
5	tt0000615	movie	Robbery Under Arms	Robbery Under Arms	0	1907	null	null	Drama	null	null	4.3	23	
6	tt0000630	movie	Hamlet	Amleto	0	1908	null	null	Drama	null	null	2.9	25	
7	tt0000675	movie	Don Quijote	Don Quijote	0	1908	null	null	Drama	null	null	4.2	19	
8	tt0000679	movie	The Fairylogue and Radio-Plays	The Fairylogue and Radio-Plays	0	1908	null	120	Adventure	Fantasy	null	5.2	66	
9	tt0000838	movie	A Cultura do Cacau	A Cultura do Cacau	0	1909	null	null	null	null	null	null	null	
10	tt0000842	movie	De Garraf a Barcelona	De Garraf a Barcelona	0	1909	null	null	null	null	null	null	null	
11	tt0000846	movie	Un día en Xochimilco	Un día en Xochimilco	0	1909	null	null	null	null	null	null	null	
12	tt0000850	movie	Los dos hermanos	Los dos hermanos	0	1909	null	null	null	null	null	null	null	
13	tt0000859	movie	Fabricación del corcho en Sant Feliu de Guixols	Fabricación del corcho en Sant Feliu de Guixols	0	1909	null	null	null	null	null	null	null	
14	tt0000862	movie	Faldgruben	Faldgruben	0	1909	null	null	null	null	null	4.5	16	
15	tt0000867	movie	Fiesta de toros	Fiesta de toros	0	1909	null	null	null	null	null	null	null	
16	tt0000868	movie	Fiestas de Santa Lucía - Belenes	Fiestas de Santa Lucía - Belenes	0	1909	null	null	null	null	null	null	null	
17	tt0000869	movie	Fiestas en La Garriga	Fiestas en La Garriga	0	1909	null	null	null	null	null	null	null	
18	tt0000879	movie	Gira política de Madero y Pino Suárez	Gira política de Madero y Pino Suárez	0	1909	null	null	null	null	null	null	null	
19	tt0000886	movie	Hamlet, Prince of Denmark	Hamlet	0	1910	null	null	Drama	null	null	4.7	38	

Figura 25 - Dashboard AngularJS

#	_id	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genre_1	genre_2	genre_3	averageRating	numVotes	Miss
1	tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	null	45	Romance	null	null	5.2	200	

Figura 26 - Dashboard AngularJS Pesquisa

## Conclusão

Após todo o desenvolvimento do projeto, o aluno conseguiu interiorizar alguns conceitos da UC de Integração de Sistemas de Informação e em que é que este tópico é tão importante no dia-a-dia.

A integração de dados deve ser sempre considerada uma prioridade no desenvolvimento de novas aplicações ou na atualização de existentes, dado que é impossível prever aquilo que será feito no futuro – e na área IT, as coisas movem-se muito rápido, tornando assim este tema mais pertinente – e, como tal, os dados devem ser sempre tratados como algo que deve ser flexível (fácil de integrar), mas sem nunca perder algumas características que são quase sempre esperadas dos mesmos – a sua coesão, segurança e manutenção.

Posto isto, as ferramentas de ETL, como a que foi aqui utilizada, são bastante facilitadoras no processo de agilizar estes processos e adequar os dados à solução ou soluções que pretendemos implementar.

Para além da utilização da ferramenta de ETL foram utilizadas também outras ferramentas, nomeadamente linguagens de programação, para demonstrar que os resultados obtidos a partir da ferramenta *Pentaho Data Integration* têm aplicabilidade em soluções reais – no caso deste projeto, optou-se por utilizar MongoDB, Python e AngularJS, ferramentas e linguagens muito utilizadas no mercado à data de hoje.

Também o desenvolvimento de uma *web API* se pode considerar um processo de integração de sistemas de informação, dado que a mesma pode ser consumida tanto por uma *dashboard* feita em AngularJS, como por uma aplicação móvel em Kotlin, ou outro tipo de soluções...

Por fim, é importante mencionar que o aluno considera que cumpriu com todos os objetivos propostos no enunciado do Trabalho Prático e crê que conseguiu demonstrar a aplicabilidade da Integração de Dados no mundo real e “sequencialmente” – começando pela extração dos dados, transformação e carregamento via Kettle; *web API* que consegue devolver os dados carregados pelo Kettle para uma base de dados; por fim uma *dashboard* que consegue visualizar estes mesmos dados, utilizando a *web API*.

## Referências

*AngularJS*. (s.d.). Obtido de <https://angularjs.org/>

*Flask*. (s.d.). Obtido de [https://pt.wikipedia.org/wiki/Flask\\_\(framework\\_web\)](https://pt.wikipedia.org/wiki/Flask_(framework_web))

*Flask*. (s.d.). Obtido de <https://flask.palletsprojects.com/en/2.2.x/>

*GitHub*. (s.d.). Obtido de <https://github.com/>

Hitachi. (s.d.). *Pentaho Data Integration*. Obtido de [https://help.hitachivantara.com/Documentation/Pentaho/8.3/Products/Pentaho\\_Data\\_Integration](https://help.hitachivantara.com/Documentation/Pentaho/8.3/Products/Pentaho_Data_Integration)

Python. (s.d.). *Python*. Obtido de <https://www.python.org/>

Wikipedia. (s.d.). *AngularJS*. Obtido de <https://pt.wikipedia.org/wiki/AngularJS>

Wikipedia. (s.d.). *Git*. Obtido de <https://pt.wikipedia.org/wiki/Git>

Wikipedia. (s.d.). *GitHub*. Obtido de <https://pt.wikipedia.org/wiki/GitHub>

Wikipedia. (s.d.). *Python*. Obtido de <https://pt.wikipedia.org/wiki/Python>