

**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](#_Toc119435223)

[Índice de Figuras 4](#_Toc119435224)

[Introdução 5](#_Toc119435225)

[Objetivos 6](#_Toc119435226)

[Dados utilizados no processo ETL 7](#_Toc119435227)

[Ferramentas utilizadas 9](#_Toc119435228)

[Pentaho Data Integration – Kettle 9](#_Toc119435229)

[Python com framework Flask 10](#_Toc119435230)

[AngularJS 11](#_Toc119435231)

[Git e GitHub 12](#_Toc119435232)

[Solução 13](#_Toc119435233)

[Solução em Pentaho Data Integration (Kettle) 13](#_Toc119435234)

[Solução em Python com Flask (Web API) 20](#_Toc119435235)

[Solução em AngularJS 20](#_Toc119435236)

# Índice de Figuras

[Figura 1 - Exemplo de registos dataset title.basics.tsv.gz 8](#_Toc119422362)

[Figura 2 - Exemplo de registos dataset title.ratings.tsv.gz 9](#_Toc119422363)

[Figura 3 - Alteração memória máxima usada pelo Pentaho Kettle 10](#_Toc119422364)

[Figura 4 - Exemplo de utilização do Pentaho Data Integration (Kettle) no projeto 10](#_Toc119422365)

[Figura 5 - Exemplo de utilização de Python com Flask no projeto 11](#_Toc119422366)

[Figura 6 - Exemplo de utilização de AngularJS no projeto 12](#_Toc119422367)

[Figura 7 - Exemplo de utilização do Git no projeto 13](#_Toc119422368)

[Figura 8 - Exemplo de utilização do GitHub no projeto 13](#_Toc119422369)

[Figura 9 - Step CSV Input no PDI 14](#_Toc119422370)

[Figura 10 - Step Replace in String no PDI 14](#_Toc119422371)

[Figura 11 - Step Split fields no PDI 14](#_Toc119422372)

[Figura 12 - Extração ratings e agrupamento dos dados 15](#_Toc119422373)

[Figura 13 - Step Merge join no PDI 15](#_Toc119422374)

[Figura 14 - Step If field value is null no PDI 16](#_Toc119422375)

[Figura 15 - Ordenação dos dados e switch/case no PDI 17](#_Toc119422376)

[Figura 16 - Step Sort rows no PDI 17](#_Toc119422377)

[Figura 17 - Step Switch / case no PDI 18](#_Toc119422378)

[Figura 18 - Step MongoDB output no PDI 18](#_Toc119422379)

[Figura 19 - Step MongoDB output no PDI 19](#_Toc119422380)

[Figura 20 - Step MongoDB no PDI 19](#_Toc119422381)

[Figura 21 – transformação data\_transformation no PDI 20](#_Toc119422382)

[Figura 22 - job job\_data\_download no PDI 20](#_Toc119422383)

[Figura 23 - job main no PDI 20](#_Toc119422384)

# 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:

*Uma imagem com texto

Descrição gerada automaticamente*

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

Uma imagem com texto

Descrição gerada automaticamente

Figura - 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:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Alteração memória máxima usada pelo Pentaho Kettle

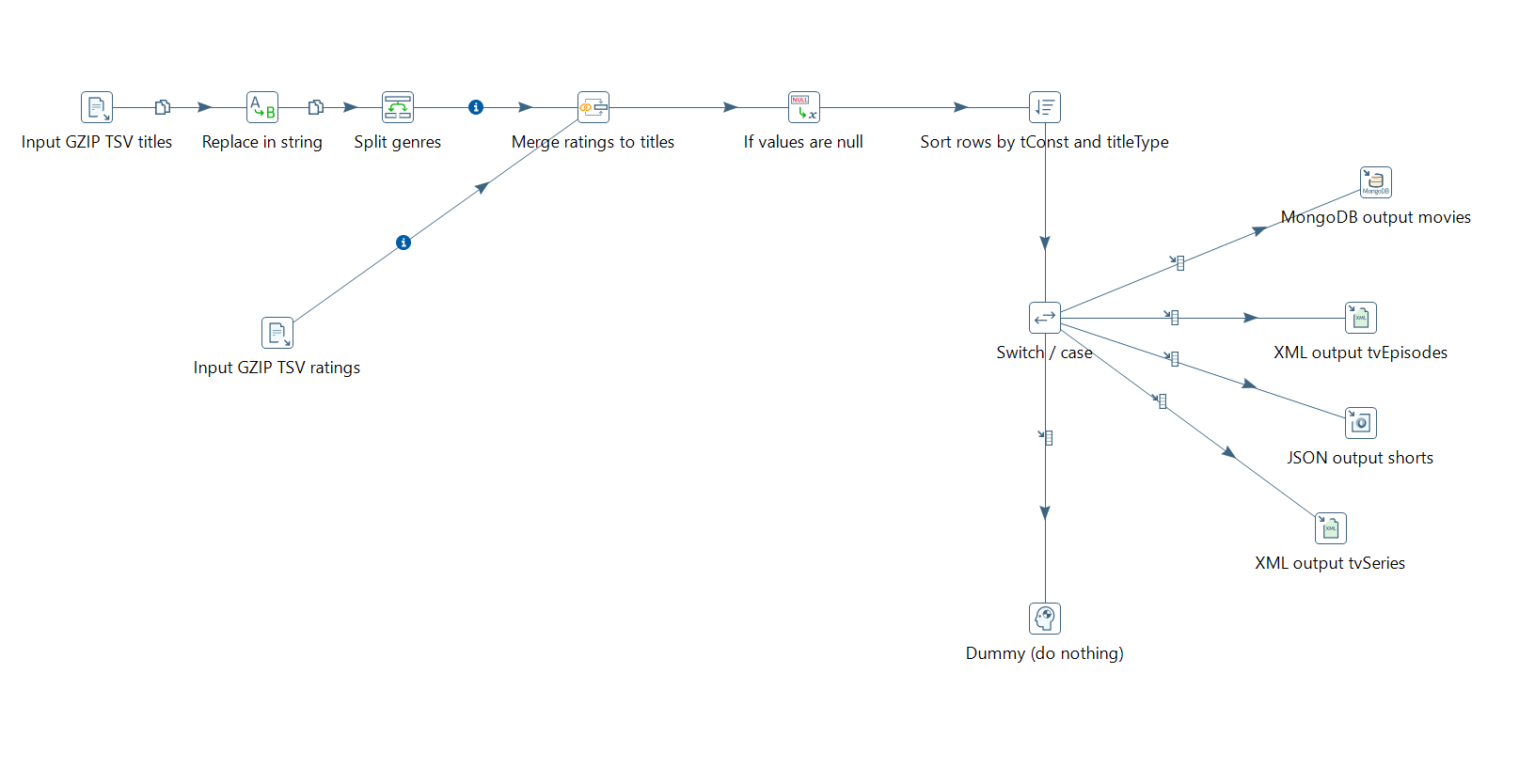


Figura - Exemplo de utilização do Pentaho Data Integration (Kettle) no projeto

## 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

Uma imagem com texto

Descrição gerada automaticamente

Figura - 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.

Uma imagem com texto

Descrição gerada automaticamente

Figura - 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*.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Exemplo de utilização do Git no projeto

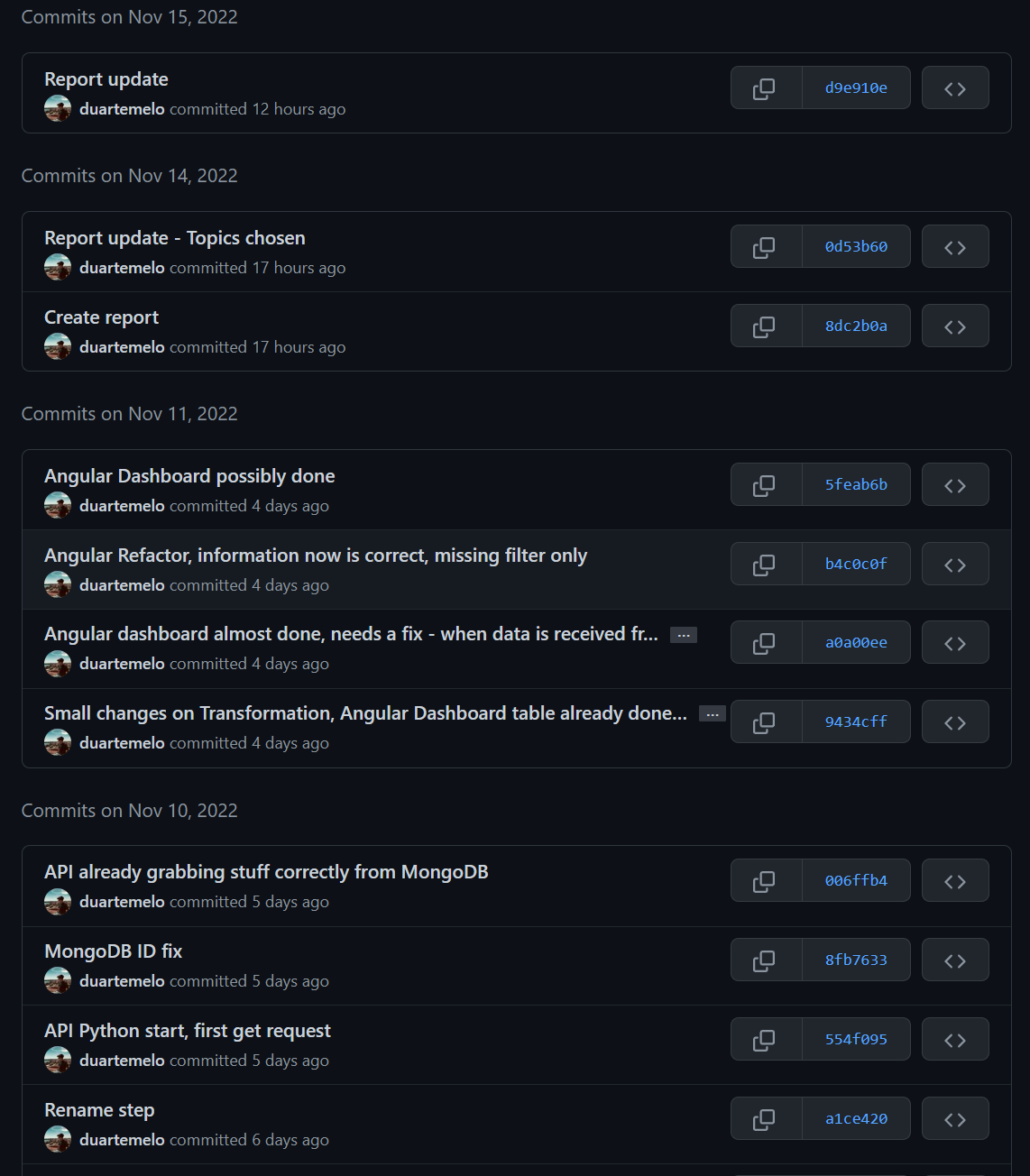


Figura - Exemplo de utilização do GitHub no projeto

# 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.

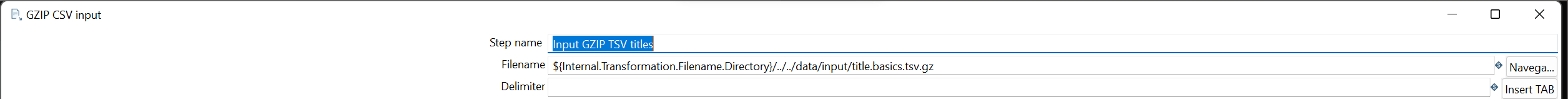


Figura - Step CSV Input no PDI

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

Uma imagem com mesa

Descrição gerada automaticamente

Figura - 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”.*

Uma imagem com texto

Descrição gerada automaticamente

Figura - 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”*.

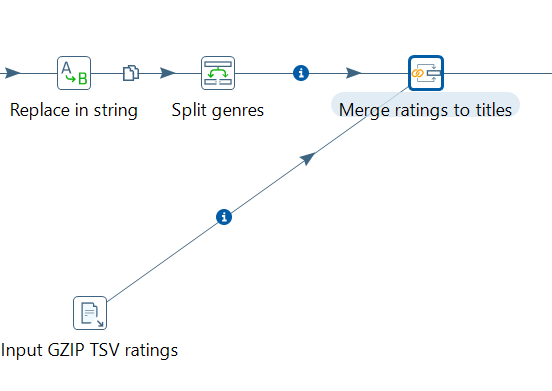


Figura - Extração ratings e agrupamento dos dados

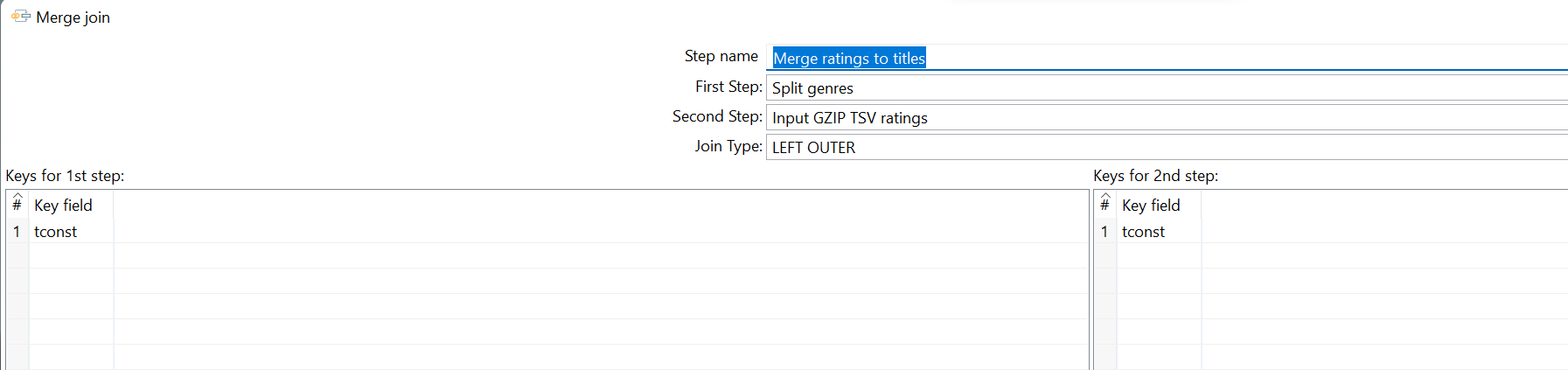


Figura - Step Merge join no PDI

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.

Uma imagem com mesa

Descrição gerada automaticamente

Figura - 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.

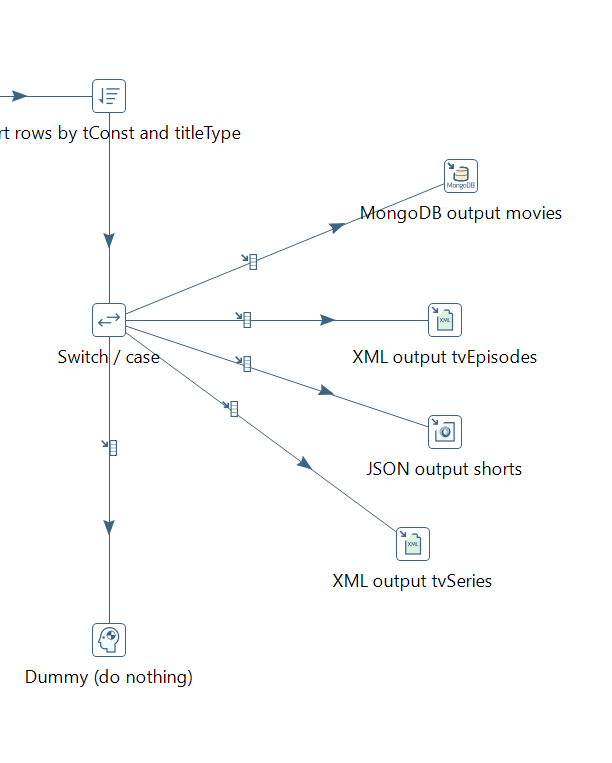


Figura - Ordenação dos dados e switch/case no PDI

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Step Sort rows no PDI

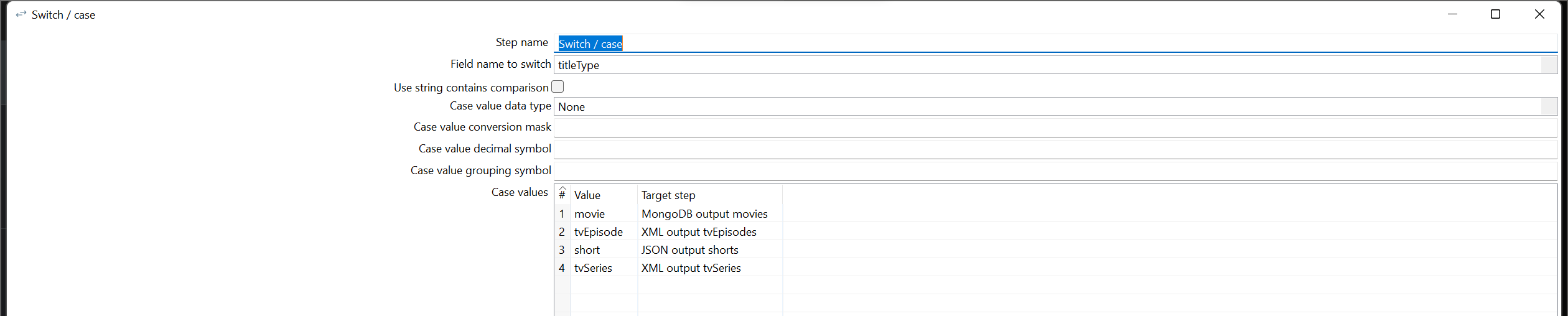


Figura - Step Switch / case no PDI

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.

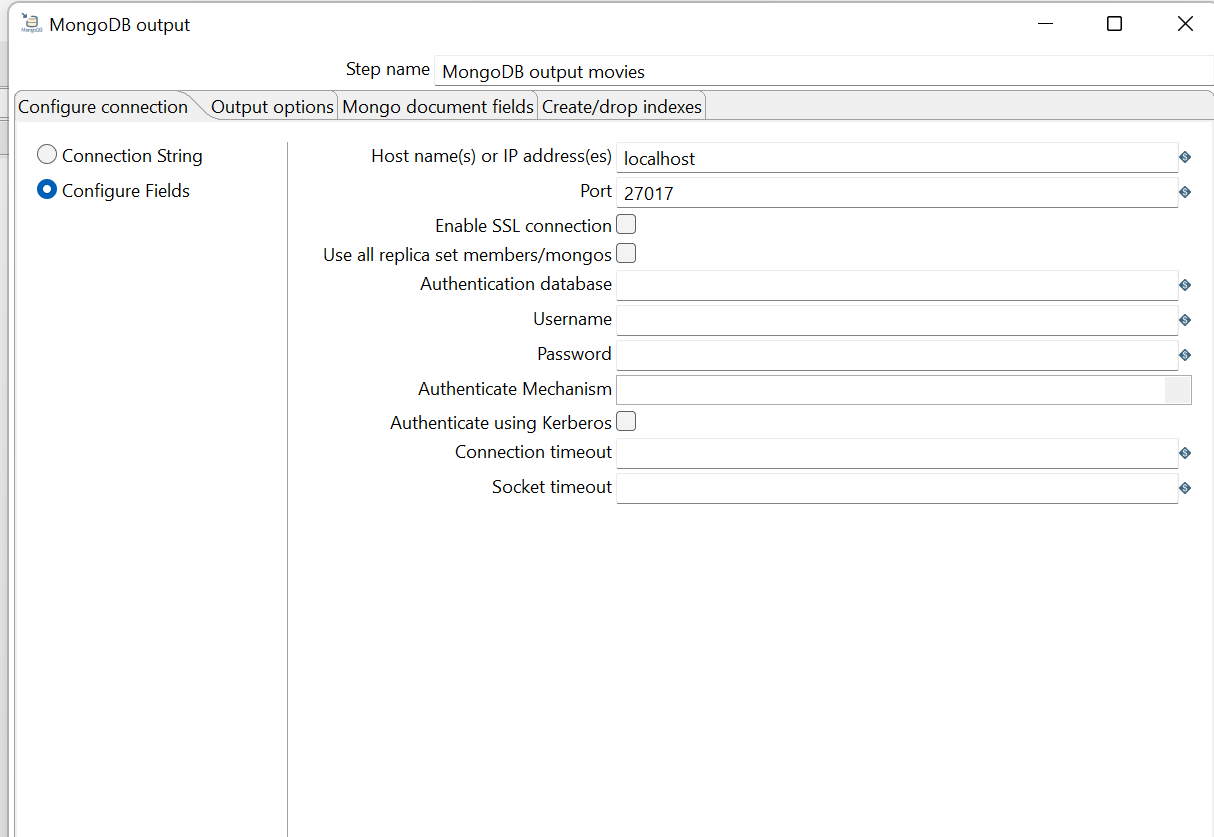


Figura - Step MongoDB output no PDI

Uma imagem com texto

Descrição gerada automaticamente

Figura - Step MongoDB output no PDI

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Step MongoDB no PDI

A transformação final:

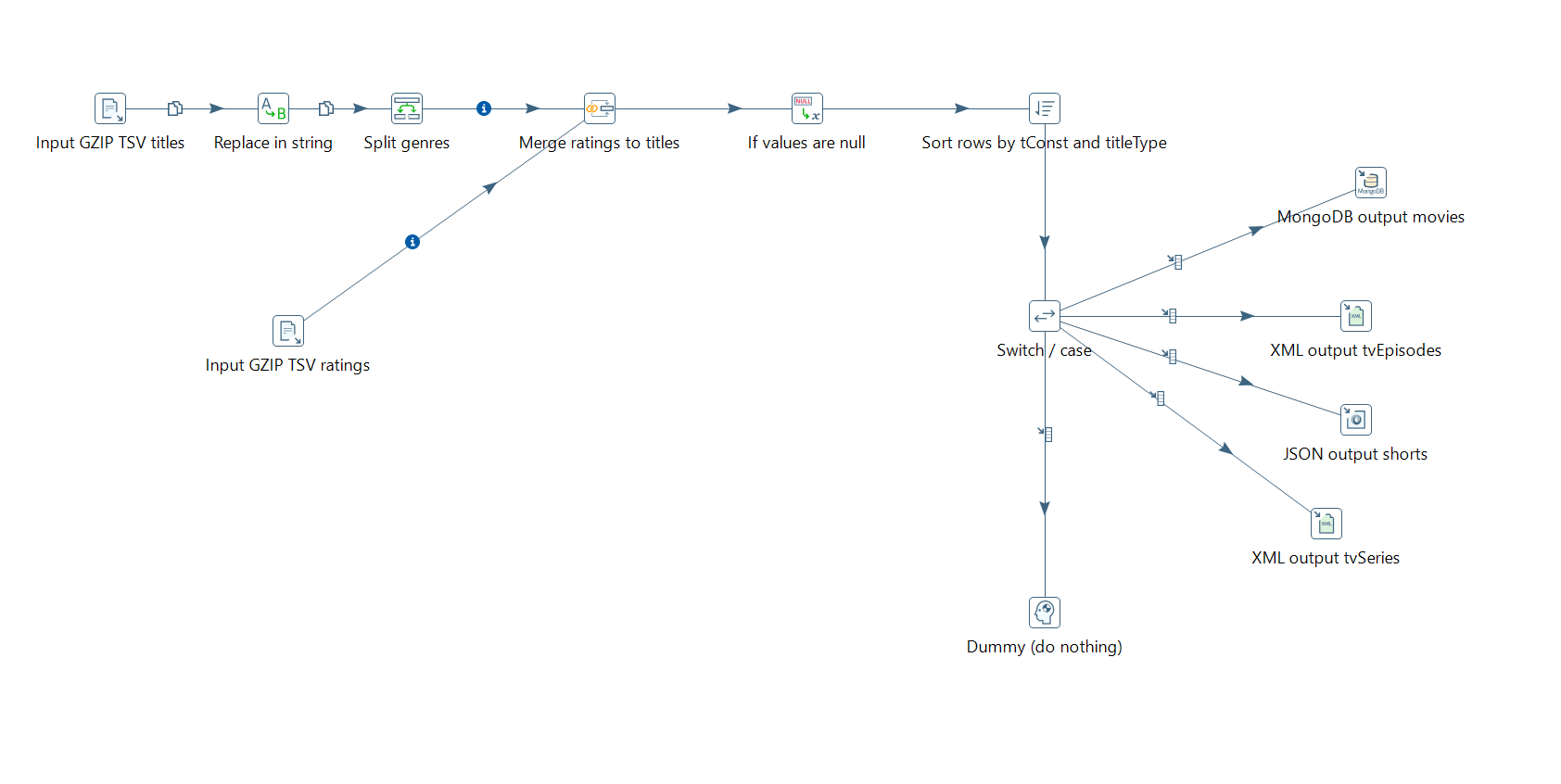


Figura – transformação data\_transformation no PDI

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.

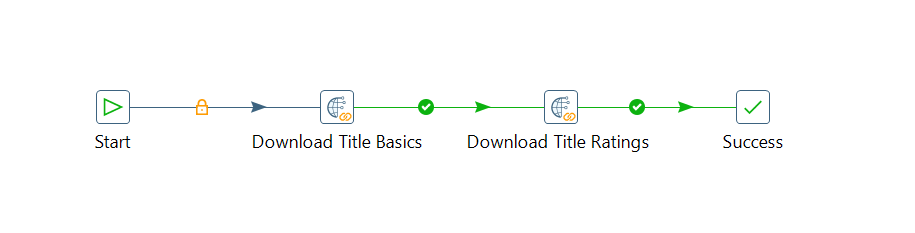


Figura - job job\_data\_download no PDI

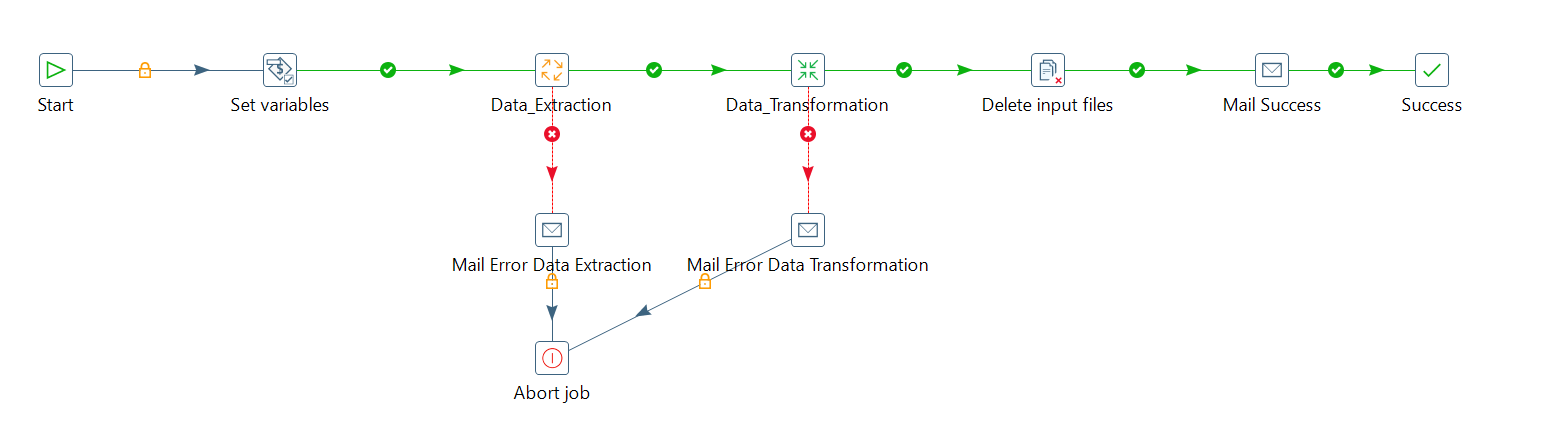


Figura - job main no PDI

## 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.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Web API Python

## Solução em AngularJS

Solução em Python Flask (API)

Solução em AngularJS

Conclusão

Lições aprendidas

Trabalho futuro

Apreciação final

Webgrafia