Instituto Politécnico do Cávado e do Ave

Integração de Sistemas de Informação

***Big Basket* – Processos ETL**

Licenciatura em Engenharia de Sistemas Informáticos

Nathaniel Joshua Armando da Silva Lloyd Jones

Nº 21116

Barcelos, Portugal

15/novembro/2022

**Conteúdo**

[Lista de Figuras 3](#_Toc119430005)

[1. Introdução 4](#_Toc119430006)

[1.1. Contextualização 4](#_Toc119430007)

[1.2. Objetivos 4](#_Toc119430008)

[1.3. Estrutura do documento 5](#_Toc119430009)

[2. Estado de Arte 6](#_Toc119430010)

[2.1. Extract, Transform and Load (ETL) Tools 6](#_Toc119430011)

[*3.* Conteúdo do ficheiro *.csv* 7](#_Toc119430012)

[3.1. *Pentaho Data Integration (PDI)* 7](#_Toc119430013)

[4. Realização do trabalho 8](#_Toc119430014)

[4.1. Transformações 9](#_Toc119430015)

[*4.2.* *Jobs* 19](#_Toc119430016)

[5. Conclusão 25](#_Toc119430017)

[6. Referências 25](#_Toc119430018)

# Lista de Figuras

[Figura 1 - Pequena Amostra do Ficheiro .csv 7](file:///C:\Users\ossie\Desktop\ISI\Relatório.docx#_Toc119430021)

[Figura 2 - Ficheiros .ktr 9](#_Toc119430022)

[Figura 3 – TransformationSortItemCategory 9](#_Toc119430023)

[Figura 4 - CSV file input 9](#_Toc119430024)

[Figura 5 - Operações do CSV file input 10](#_Toc119430025)

[Figura 6 - Sort rows by Category 10](#_Toc119430026)

[Figura 7 - Operações Sort rows by Category 10](#_Toc119430027)

[Figura 8 - Switch/ case 11](#_Toc119430028)

[Figura 9 - Operação Switch/ case 11](#_Toc119430029)

[Figura 10 - Gestão das Categorias 12](#_Toc119430030)

[Figura 11 - Processamento de informação da categoria Baby Care 12](#_Toc119430031)

[Figura 12 - JSON Baby Care 13](#_Toc119430032)

[Figura 13 - Obtenção de campos do Step JSON Baby Care 13](#_Toc119430033)

[Figura 14 - XML Baby Care 14](#_Toc119430034)

[Figura 15 - Obtenção de campos do Step XML Baby Care 14](#_Toc119430035)

[Figura 16 - Conexão à base de dados do MongoDB 14](#_Toc119430036)

[Figura 17 - Ligação do output à coleção Baby Care 15](#_Toc119430037)

[Figura 18 - Conexão a uma implantação do MongoDB 15](#_Toc119430038)

[Figura 19 - Base de dados Big Basket no MongoDB 16](#_Toc119430039)

[Figura 20 - Dados armazenados da categoria Baby Care no MongoDB 16](#_Toc119430040)

[Figura 21 – TransformationCheapestByCategory 17](#_Toc119430041)

[Figura 22 - Sort Baby Care por preço ascendente 17](#_Toc119430042)

[Figura 23 - Sample Row Baby Care mais barato 18](#_Toc119430043)

[Figura 24 - Sample rows do primeiro selecionado 18](#_Toc119430044)

[Figura 25 - XML Cheapest 18](#_Toc119430045)

[Figura 26 – TransformationMostExpensiveByCategory 18](#_Toc119430046)

[Figura 27 - Job Send Email 19](#_Toc119430047)

[Figura 28 – Start 19](#_Toc119430048)

[Figura 29 - Transformação do SortItemCategory 20](#_Toc119430049)

[Figura 30 - Mail validator 20](#_Toc119430050)

[Figura 31 - Email de envio e destino 20](#_Toc119430051)

[Figura 32 - Autenticação do SMTP Server e email em uso 21](#_Toc119430052)

[Figura 33 - Email criado pelo Job 21](#_Toc119430053)

[Figura 34 - Job HTML 22](#_Toc119430054)

[Figura 35 - File exists XML ME (Most Expensive) 22](#_Toc119430055)

[Figura 36 - File exists XML ME função 22](#_Toc119430056)

[Figura 37 - Delete file HTML ME 23](#_Toc119430057)

[Figura 38 - Delete file HTML ME função 23](#_Toc119430058)

[Figura 39 - XSL Most\_Expensive 23](#_Toc119430059)

[Figura 40 - XSL Most\_Expensive função 23](#_Toc119430060)

[Figura 41 - HTML gerado pelo Job 24](#_Toc119430061)

# Introdução

Neste capítulo será elabora os conteúdos deste relatório, dando o leitor a perceber no que consiste este projeto, a contextualização e os objetivos do mesmo, tal como também a estrutura do documento.

## Contextualização

Maior parte dos sistemas de informação exigem uma troca de dados, e com esta exigência podem aparecer vários problemas tais como uma sequência incorreta de informação, inconsistência dos dados ou informação errada ou até um formato de dados errado. Para evitar estes problemas surgiu a área de integração de dados, passando pela análise das regras e processos de negócio e, a partir dai, construir uma solução que permita a comunicação entre dois ou mais sistemas. Um dos processos de integração de sistemas passa pela Extração, Transformação e Carregamento (sigla ETL em inglês) dos dados.

A importância do ETL numa organização é diretamente proporcional quanto à organização do armazenamento de dados. As ferramentas de ETL coletam, leem e migram grandes volumes de dados brutos de várias fontes de dados e entre plataformas diferentes, carregando esses dados num único banco de dados ou um armazenamento de dados para facilitar o acesso. Após isso processam os dados para torná-los significativos com operações como classificação, junção, reformatação, filtragem, incorporação e agregação. Por fim, eles incluem interfaces gráficas para resultados mais rápidos e fáceis do que os métodos tradicionais de movimentação de dados por meio de pipelines de dados codificados manualmente.

As ferramentas modernas de ETL incluem inteligência integrada que deteta e reflete continuamente as alterações nos objetos de origem e destino para garantir a consistência dos dados – a força vital da estratégia e da experiência do cliente.

## Objetivos

O principal propósito deste projeto é a transformação e extração de dados de um ficheiro *.csv*, chamado *Big Basket*, proveniente do website *Kaggle*.

Para a extração destes dados, deve resultar treze ficheiros *Extensible Markup Language (.xml)* e onze ficheiros *JavaScript Object Notation (.json)*.

## Estrutura do documento

O documento encontra-se organizado em 5 capítulos sendo estes a Introdução, Estado de Arte, Conteúdo dos Ficheiros, Realização do Trabalho, Conclusão e Referencias.

## Link Video Youtube

https://youtu.be/uriO6MDWrog

# Estado de Arte

Qualquer supermercado, tanto de grandes dimensões como de pequenas, tem sempre uma vasta lista de artigos e quantidades dos mesmos, entre outras informações. Para a melhor gestão deste estabelecimento é crucial uma base de dados e uma tecnologia que manipule eficazmente o mesmo consoante as necessidades dos utilizadores.

Sem esta organização um funcionário terá de ver artigo a artigo o que tem em stock, sem mencionar os outros dados que tem de ser tomados em conta, tal como o prazo de validade. Com uma base de dados o trabalho deste funcionário será extremamente simplificado e com uma margem de erros no armazenamento dos dados quase inexistente.

Já existem programas que conseguem resolver estes problemas e foi ao estudar alguns deles que eu consegui desenvolver o meu projeto.

## Extract, Transform and Load (ETL) Tools

As ferramentas *ETL* tratam de um processo de três passos de gestão de dados sendo o primeiro a extração dos dados de uma fonte de dados estruturada ou não estruturada, transformando-os num formato que satisfaça os requisitos funcionais e analíticos da empresa e, por fim, carrega os dados no alvo destino.

Um sistema de *ETL* bem desenvolvido extrai os dados, impõem padrões de qualidade dos dados, preenche os dados num formato padronizado para que varias fontes possam ser usadas juntas e entrega os dados prontos para que os desenvolvedores da aplicação a possam desenvolver e os donos do produto possam tomar decisões.

# Conteúdo do ficheiro *.csv*

O ficheiro *.csv* tem os seguintes campos:

* *ProductName* do tipo *String* que guarda o nome do produto;
* *Brand* do tipo *String* que guarda a marca do produto;
* *Price* do tipo *Number* com formato “#.#” que guarda o preço do produto;
* *DiscountPrice* do tipo *Number* com formato “#.#” que guarda o preço do produto em desconto;
* *Quantity* do tipo *String* que guarda a quantidade do artigo em peças, litros ou kilos;
* *Category* do tipo *String* que guarda a categoria do produto;
* *SubCategory* do tipo *String* que guarda a subcategoria do produto;



Figura - Pequena Amostra do Ficheiro .csv

## *Pentaho Data Integration (PDI)*

O *Pentaho Data Integration (PDI)* fornece os recursos *Extract, Transform e Load (ETL)* que facilitam o processo de captura, limpeza e armazenamento de dados usando um formato uniforme e consistente que é acessível e relevante para usuários finais e tecnologias *IoT* (*Internet of* *Things*).

Os usos comuns do *Pentaho Data Integration* incluem:

* Migração de dados entre diferentes bases de dados e aplicações;
* Importação em massa de dados em bases de dados, aproveitando ao máximo os ambientes de processamento em *Cloud*, agrupamentos e ambientes de processamento paralelo maciços;
* Limpeza de dados com etapas que variam de transformações muito simples a muito complexas;
* Integração de dados, incluindo a capacidade de alavancar ETL em tempo real como fonte de dados para *Pentaho Reporting*;
* População de *data* *warehouse* com suporte integrado para dimensões de alteração lenta e criação de chave substituta.

# Realização do trabalho

***.CSV***

Para iniciar o trabalho foi preciso obter um ficheiro *.csv* (*Comma-seperated values*)adequado ao tema que é *stock* de um supermercado.

Um ficheiro *.csv* é caracterizado pela separação de informação por virgulas, maior parte das vezes usado em bases de dados e folhas de cálculo. Estes tipos de ficheiros são usados para mover dados entre programas que não tem as capacidades básicas de trocas de informação.

***.XML***

Um ficheiro *.xml* (*Extensible Markup Language*) é usado para estruturar dados para armazenamento e transporte.

Este ficheiro tem de ter compreensão fácil pois utilizam a língua do ser humano em vez de linguagem da máquina. A informação contida neles está etiquetada por categoria para que esta seja organizada e facilmente identificada.

***.XSL***

Um ficheiro *.xsl* (*Extensible Stylesheet Language*) tem nele contigo código capaz de criar uma folha de dados que descreve os dados que serão apresentados na página *WEB.*

***.HTML***

*HTML* (*Hyper Text Markup Language*) é a linguagem principal para o desenvolvimento de páginas *WEB.*

## Transformações

As transformações estão guardadas num tipo de ficheiro *.ktr* (*Kettle Transformation File*). O meu projeto tem os seguintes ficheiros *.ktr*:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Ficheiros .ktr

***TransformationSortItemCategory***

A primeira transformação é a “*TransformationSortItemCategory*” que filtra as categorias dos artigos da base de dados e exporta os dados em *.json*, *.xml* e também para a base de dados *MongoDB*.

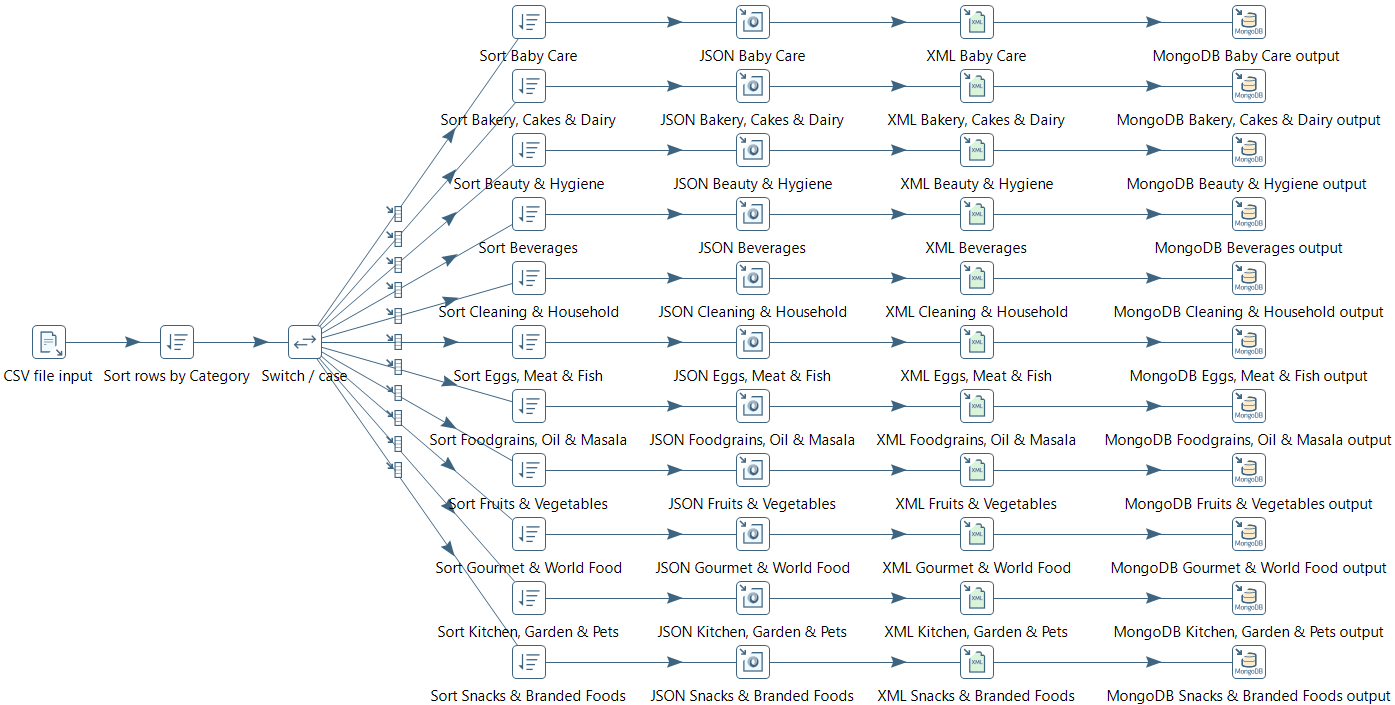


Figura – TransformationSortItemCategory

Os dados são inseridos no *input* “*CSV file input*” que identifica as tabelas, nome, tipo de dados, formato e comprimento do ficheiro *.csv*.

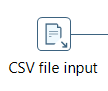


Figura - CSV file input

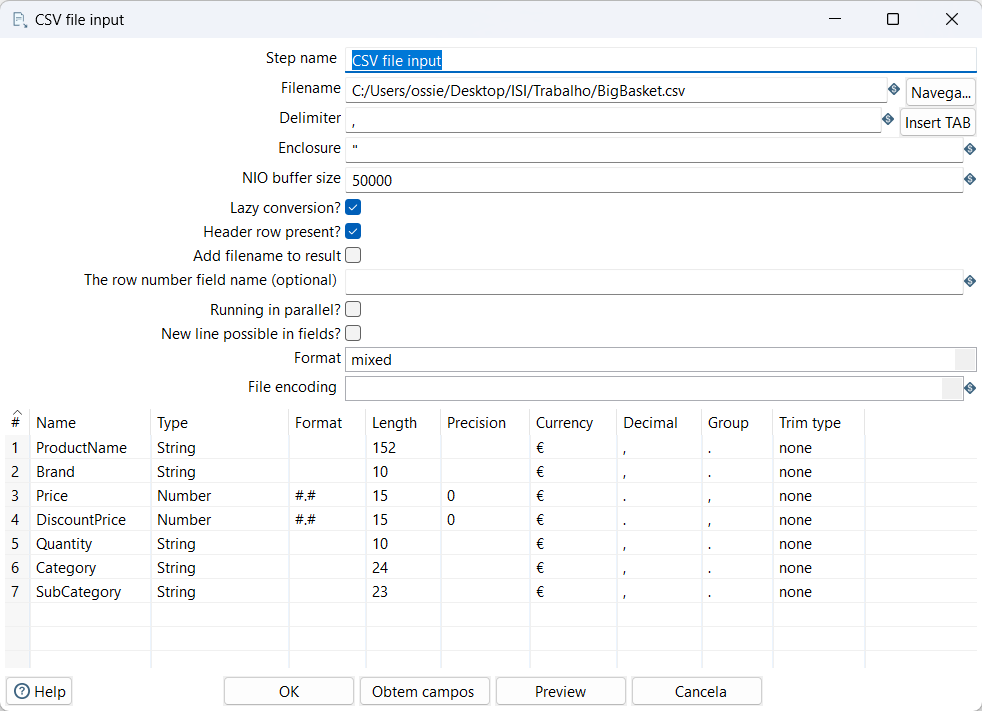


Figura - Operações do CSV file input

Após a inserção dos dados, o campo com nome “*Category*” é filtrado por ordem alfabética no passo “*Sort rows by Category*”.

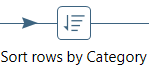


Figura - Sort rows by Category

Uma imagem com texto

Descrição gerada automaticamente

Figura - Operações Sort rows by Category

Depois para organizar e separar as categorias recorri a um “Switch / case” que verifica o que está escrito dentro da coluna “Category” e compara o texto do formato *String* com o valor inserido no Switch/ case.



Figura - Switch/ case

Uma imagem com texto

Descrição gerada automaticamente

Figura - Operação Switch/ case

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Gestão das Categorias

**Gestão do *Baby Care***



Figura - Processamento de informação da categoria Baby Care

No início do processo é ligado ao *Switch/ case* o *Sort Baby Care* que separa a categoria por *Baby Care* das restantes.

De seguida a categoria *Baby Care* é convertida para o formato *.json*, *.xml* e exportado para a base de dados no *MongoDB*.

***Step JSON Baby Care***

Uma imagem com texto

Descrição gerada automaticamente

Figura - JSON Baby Care

Aqui ocorre a conversão do formato *.csv* para *.json* e terá como diretório “${Internal.Entry.Current.Directory}/Ficheiros de Data/Baby Care” sendo que o “${Internal.Entry.Current.Directory}” servirá para encontrar sempre o diretório em que se encontra o ficheiro *.csv*.

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Obtenção de campos do Step JSON Baby Care

***Step XML Baby Care***

Uma imagem com texto

Descrição gerada automaticamente

Figura - XML Baby Care

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Obtenção de campos do Step XML Baby Care

Aqui ocorre a conversão do ficheiro *.csv* para *.xml* onde é guardado no ficheiro referido acima na figura 14 e a obtenção dos campos e respetivos tipos e formatos.

***Step MongoDB Baby Care output***

Uma imagem com texto

Descrição gerada automaticamente

Figura - Conexão à base de dados do MongoDB

Uma imagem com texto

Descrição gerada automaticamente

Figura - Ligação do output à coleção Baby Care

Neste passo é necessário selecionar a base de dados a que queremos ligar o output do MongoDB do nosso *PDI* e também a coleção onde queremos armazenar os campos e dados do *Baby Care*.

**Ligação à base de dados *MongoDB***

Uma imagem com texto

Descrição gerada automaticamente

Figura - Conexão a uma implantação do MongoDB

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Base de dados Big Basket no MongoDB

Uma imagem com texto

Descrição gerada automaticamente

Figura - Dados armazenados da categoria Baby Care no MongoDB

Nas três imagens acima temos a ligação da base de dados do *MongoDB* (figura 18) que permite que a base de dados esteja ativa e que o *output* do *PDI* tenha onde armazenar os dados, a segunda imagem (figura 19) tem aberto a base de dados *Big Basket* e também as categorias lá inseridas, tais como a *Baby Care* e a *Beverages*…, e por fim a última imagem (figura 20) tem aberto a categoria *Baby Care* e apresenta os primeiros 3 artigos e respetivos atributos desta categoria.

***TransformationCheapestByCategory***

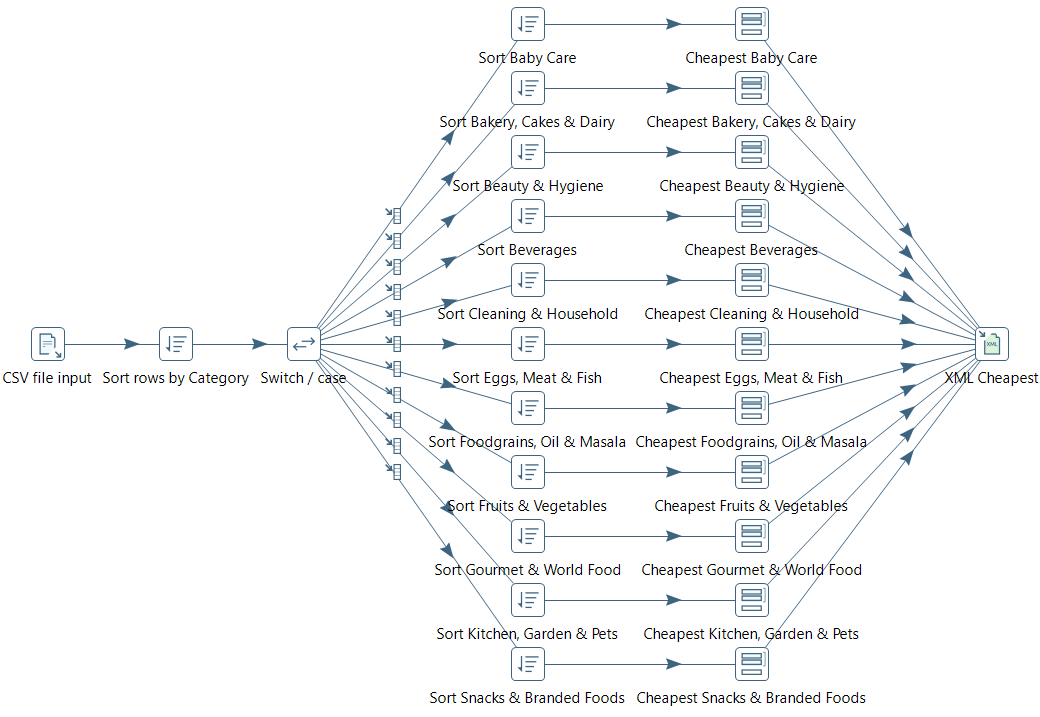


Figura – TransformationCheapestByCategory

Nesta transformação, até ao passo do *Sort* após o *Switch/ case* faz todas as operações da transformação explicada no ponto acima porem, o *Sort* faz uma filtração para obter o artigo mais barato por categoria, depois exportando para um *.xml*.

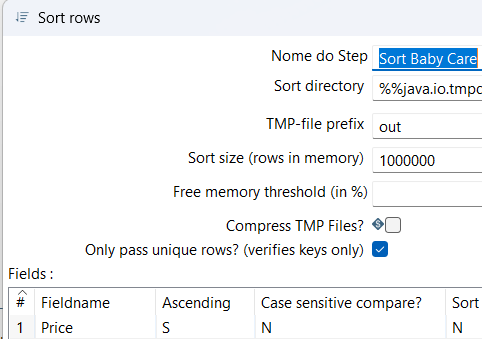


Figura - Sort Baby Care por preço ascendente



Figura - Sample Row Baby Care mais barato

Uma imagem com texto

Descrição gerada automaticamente

Figura - Sample rows do primeiro selecionado

Neste *Sample rows* é selecionado o que está inserido na primeira linha da coluna do *Sort Row* anterior, sendo este o artigo mais barato.



Figura - XML Cheapest

Este *XML output*, tal como os anteriores, guarda um ficheiro *.xml* na pasta de Ficheiros de Data.

***TransformationMostExpensiveByCategory***

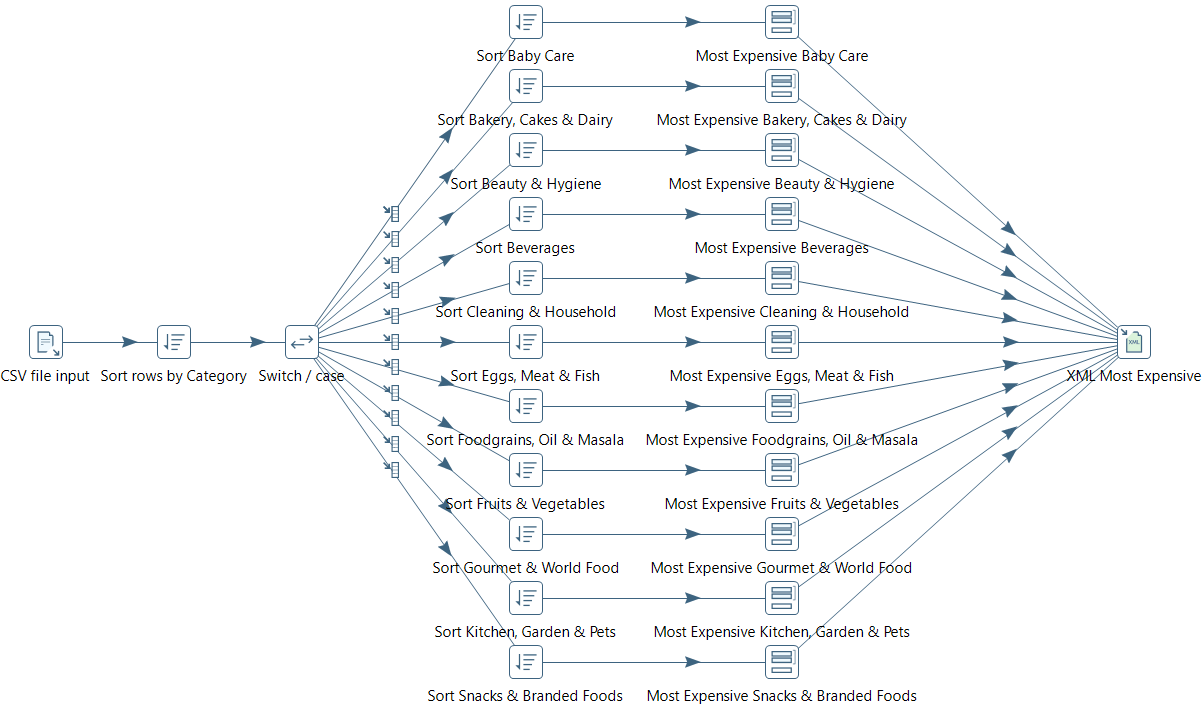


Figura – TransformationMostExpensiveByCategory

Esta transformação faz exatamente o que a anterior faz, mas em vez do Sort estar por ordem ascendente, esta por ordem decrescente, ficam com a primeira linha com o artigo mais caro.

## *Jobs*

O *Job* é o que orquestra as atividades *ETL* do *PDI*.

Este projeto tem dois *Jobs*. O primeiro que cria páginas HTML com os dados das tabelas e o segundo que envia um email com os *logs*.

***Job Send Email***

Uma imagem com texto, relógio

Descrição gerada automaticamente

Figura - Job Send Email

Uma imagem com texto

Descrição gerada automaticamente

Figura – Start

Com isto inicia o *Job*.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Transformação do SortItemCategory

Isto corre a transformação que separa os artigos por categoria.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Mail validator

Este passo valida o email utilizado no processo seguinte.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Email de envio e destino

Email de envio e destino.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Autenticação do SMTP Server e email em uso

Dados do *SMTP* *Server* do *Outlook* e autenticação do meu email.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Email criado pelo Job

***Job HTML***

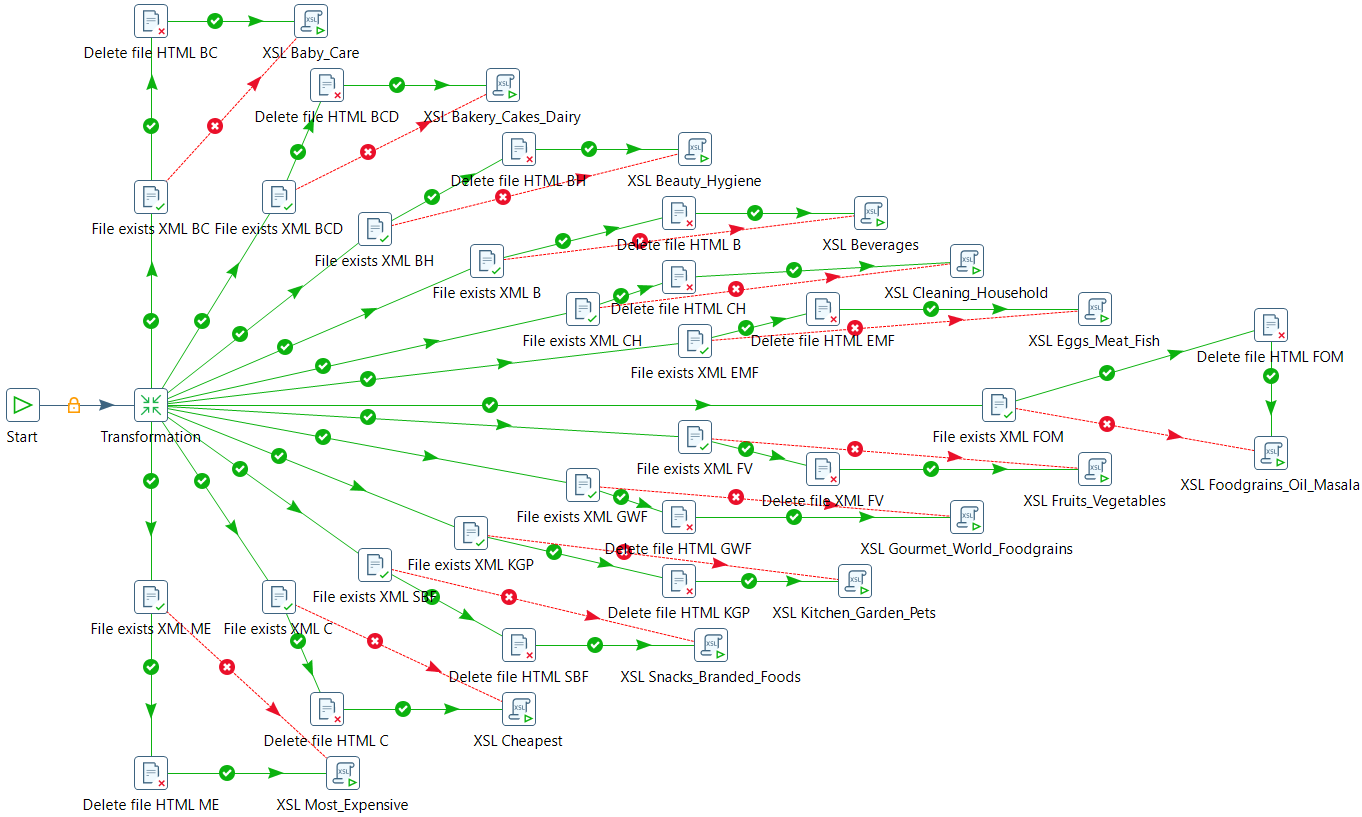


Figura - Job HTML

Neste Job é utilizado a transformação “*SortItemCategory*”e a partir dos ficheiros *.xml* é criado páginas *HTML*.

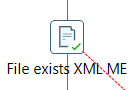


Figura - File exists XML ME (Most Expensive)

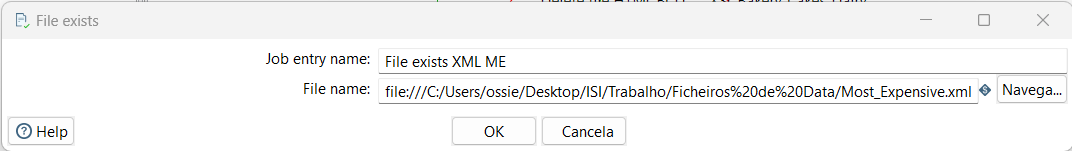


Figura - File exists XML ME função

No File Exists o Job procura se o ficheiro *.xml* existe. Caso ele exista ele apaga o ficheiro HTML. Se não existir cria o ficheiro HTML.



Figura - Delete file HTML ME

Uma imagem com texto

Descrição gerada automaticamente

Figura - Delete file HTML ME função

Aqui o ficheiro *.html* é eliminado.



Figura - XSL Most\_Expensive

Uma imagem com texto

Descrição gerada automaticamente

Figura - XSL Most\_Expensive função

Caso o ficheiro de HTML não existe, esta função procura o ficheiro *.xml* e *.xsl* e com esses dados consegue criar o ficheiro *.html*.

Uma imagem com mesa

Descrição gerada automaticamente

Figura - HTML gerado pelo Job

# Conclusão

A integração de dados forneceu uma solução prática para um problema complexo.

Atualmente, conseguimos utilizar uma aplicação, e efetuar o envio de dados para outra aplicação completamente diferente, mediante um conjunto de padrões associados. No entanto, e dado o elevado risco de segurança, trabalhar com dados online é um trabalho bastante difícil e complexo, uma vez que potencia a exploração de falhas de segurança, isto é injeção de código.... Nesse seguimento, as ferramentas ETL tornam-se uma necessidade, não só pela redução no tempo despendido a criar a solução, mas também a extrair informação e transformar a mesma.

# Referências

Como referências foi utilizado o material fornecido pelo professor e também:

https://www.kaggle.com/datasets/chinmayshanbhag/big-basket-products