



Relatório do 2º Trabalho Prático

Integração de Sistemas de Informação

Aluno:

18836 – Carlos Martins

Professor:

Luís Ferreira

Licenciatura em Engenharia de Sistemas Informáticos

Braga, Dezembro, 2021

Conteúdo

1. Introdução.....	4
1.1. Contextualização	4
1.2. Motivação e objetivos	4
1.3. Documentos de Entrega	4
2. Implementação	5
2.1. Base de dados.....	5
2.2. Número de vendas por categoria.....	7
2.2.1. Transformation	7
2.2.2. Job	8
2.3. Migração de dados para MongoDB	9
2.3.1. Transformations.....	9
2.3.2. Job	10
2.4. Dividir stock em categorias	11
2.4.1. Transformations.....	11
2.4.2. Job.....	13
2.5. Filtrar Produtos por Categoria.....	14
2.5.1. Transformation	14
2.6. Listar produtos de 16-Inches	16
2.6.1. Transformation	16
2.7. NodeJS REST WebServices com MongoDB	16
2.7.1. Método GET – Retornar todos os produtos	17
2.7.2. Método GET –Retornar um produto com um certo nome	17
2.7.3. Método GET – Retornar um produto com um certo ID	18
2.7.4. Método POST – Criar um produto	19
2.7.5. Método DELETE – Apagar um produto pelo ID	20
2.7.6 Método PATCH – Atualizar o preço de um produto	21
2.8. Obter os produtos via Web Service	22
2.8.1. Transformation	22
2.8.2. Job	24
3. Conclusão.....	25
4. Bibliografia.....	26

Figura 1- Base de dados BikeStores.....	5
Figura 2- Transformation Load_Customers.....	5
Figura 3 - Transformation Load FTOrdes.....	6
Figura 4 - Job Load All Tables	6
Figura 5 - Transformação N_Vendas/Categoria	7
Figura 6 - Número de vendas/categoria.....	7
Figura 7 - XML com as categorias	7
Figura 8 - Job N_Vendas/Categoria.....	8
Figura 9 - XSL Tranformation N_Vendas/Categoria	8
Figura 10 - PieChart N_Vendas/Categoria.....	8
Figura 11 – Transformation ClientesToMongoDB	9
Figura 12 - Filtrar nulls tabela Customers	9
Figura 13 - Configurações do servidor MongoDB	10
Figura 14 - Definições da base de dados.....	10
Figura 15 - Definição dos campos do documento	10
Figura 16 - Job MigrarDadosMongoDB.....	10
Figura 17 - Filter Good Stock Products	11
Figura 18 - Configurações ExcelWriter	11
Figura 19 – Filter Medium Stock Products	12
Figura 20 - Filter Low Stock Products	12
Figura 21 - Excel resultante	12
Figura 22 - Transformação para obter o nome do ficheiro	13
Figura 23 - Enviar email.....	13
Figura 24 - Dados para o Email	13
Figura 25 - Job Enviar CSV por Email	13
Figura 26 - Resultado JOB EnviarCSVMail.....	14
Figura 27 - MongoDBProdutos Fields	14
Figura 28 - SwitchCase configurations.....	15
Figura 29 - CategoryXML Fields	15
Figura 30 - Transformation FilterBy16Inch	16
Figura 31 – Get All Products	17
Figura 32 - GetProduct By Name	18
Figura 33 - Get Product By Id	18
Figura 34 - CreateProduct PostMan.....	19
Figura 35 - Produto criado com sucesso.....	20
Figura 36 - GetProductByID – Created product.....	20
Figura 37 - DeleteProduct Method	21
Figura 38 - UpdatePrice - PATCH Method	22
Figura 39 - Transformation GetProducts WS	22
Figura 40 - GenerateRows GetProductsWS	22
Figura 41 - Configurações REST Client	23
Figura 42 - JSON Input configuration GetProductWS	23
Figura 43 - Job GerarHTML WS	24
Figura 44 - Resultado Job GerarHTML WS	24

1. Introdução

Este trabalho prático foi ligeiramente diferente dos habituais, porque não tenha um tema definido e cabia a cada grupo escolher o seu tema, mas abordá-lo em diferentes plataformas de ETL.

O nosso tema para este trabalho foi trabalhar uma base de dados de uma loja de bicicletas. Vale a pena notar que esta base de dados também foi utilizada na disciplina de Sistemas de Apoio à Decisão para fazer o processo dimensional. Como se trata de uma base de dados de uma loja de bicicletas, a mesma possui tabelas com informações acerca dos clientes, das lojas existentes, dos produtos, das vendas e outras informações.

1.1. Contextualização

O presente relatório foi realizado devido à necessidade de documentar todo o trabalho realizado.

1.2. Motivação e objetivos

Tive uma motivação extra na realização deste trabalho porque era de tema aberto e isso forçou-me a investigar tecnologias que eu nunca tinha utilizado e que não conhecia. Além disso, ajudou a cimentar os conhecimentos obtidos nas cadeiras de Integração de Sistemas Informáticos e Sistemas de Apoio à Decisão.

Os meus objetivos após a realização deste trabalho passam por cimentar os conhecimentos obtidos na realização do mesmo em cadeiras futuras e no meu futuro profissional.

1.3. Documentos de Entrega

- Na pasta ETL encontram-se todos as transformations e jobs.
- Na pasta WebServices encontram-se os webservices desenvolvidos em NodeJS.

2. Implementação

2.1. Base de dados

A base de dados deste tema é a mesma base de dados utilizada num exercício da cadeira de Sistemas de Apoio à Decisão. Utilizei a minha resolução, que foi feita em aula, com ligeiras alterações para este trabalho. Apresento abaixo o esquema de base de dados BikeStores.

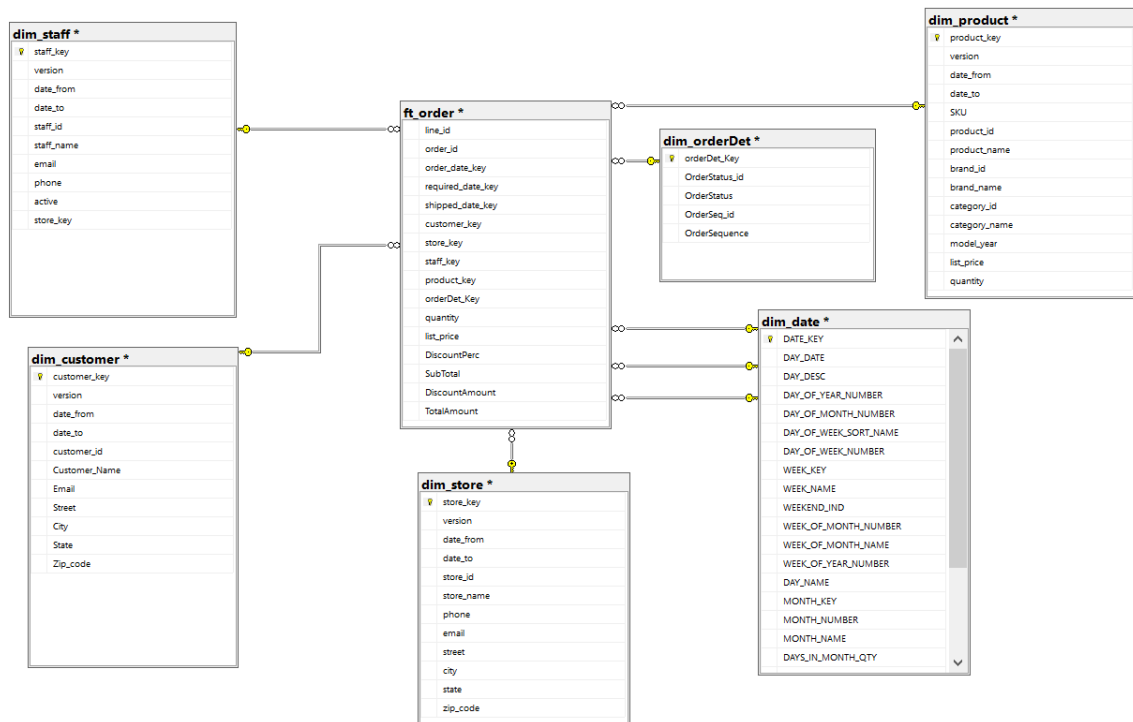


Figura 1- Base de dados BikeStores

Como já referi em cima o carregamento destas tabelas para uma base de dados em SQL Server foi um exercício realizado na disciplina de Sistemas de Apoio à Decisão e não foi algo que realizei completamente para este trabalho, porque fiz apenas algumas alterações então não me vou alongar muito sobre este ponto.

Na figura é possível ver uma transformação que recebe os dados da tabela Customers da base de dados original (Customers input) depois é feita a inserção dos mesmos dados na tabela do modelo dimensional (Customers Dimension lookup/update).



Figura 2- Transformation Load_Customers

É feito praticamente o mesmo processo para as outras dimensões, com exceção da tabela da dimensão data, que é baseada num template do Matt Casters e está devidamente referenciado na própria transformação, e da tabela de factos.

A tabela de factos funciona basicamente como um recibo. Esta tabela contém todas as informações respetivas a uma encomenda. Como é possível ver, começa por receber os dados da tabela Orders e associa as respetivas datas, cliente, loja, staff, produto e estado da encomenda à respetiva encomenda.

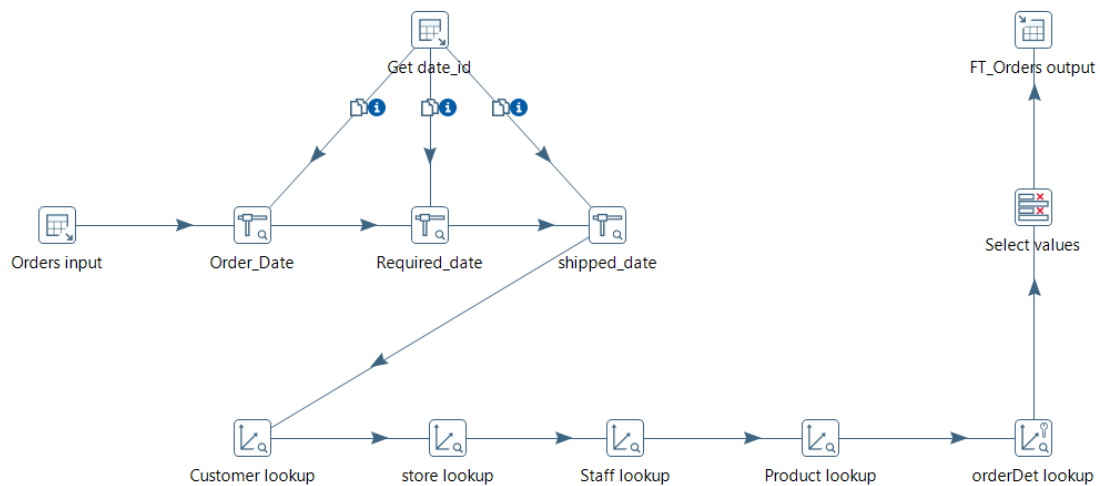


Figura 3 - Transformation Load FTOrdes

Para finalizar criei um job simples que carrega todas as transformações de carregamento de tabelas para a base de dados.



Figura 4 - Job Load All Tables

2.2. Número de vendas por categoria

2.2.1. Transformation

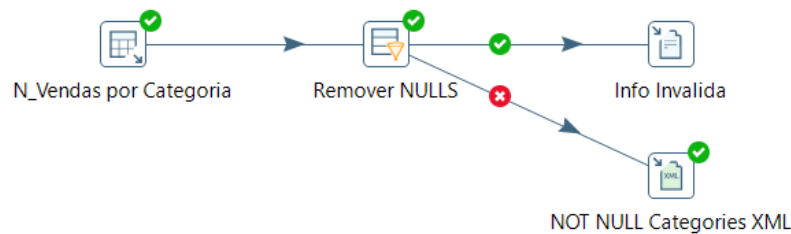


Figura 5 - Transformação N_Vendas/Categoria

Comecei por criar uma query que me devolvesse o número de vendas respetivas a cada categoria. A query é a seguinte:

```
SELECT pr.category_name AS Categoria, COUNT(pr.category_name) AS Quantidade
FROM dim_product pr INNER JOIN ft_order ft ON pr.product_key = ft.product_key
GROUP BY category_name
```

O resultado obtido a partir da query foi o seguinte:

#	Nome	Quantidade
1	Comfort Bicycles	537
2	Electric Bikes	212
3	Road Bikes	374
4	Cruisers Bicycles	1378
5	Mountain Bikes	1183
6	Children Bicycles	782
7	Cyclocross Bicycles	256

Figura 6 - Número de vendas/categoria

No Spoon utilizei um Table Input, que recebia os dados obtidos pela query indicada em cima, seguido de um Filter Rows com o objetivo de remover categorias com o nome ou quantidade definida a null (o que neste caso não existia, mas foi uma forma de prevenir). Se fosse encontrada alguma linha com um valor null o resultado do filtro era True e enviava-os para um ficheiro de texto com o nome InfoInválida, se não enviava-os para um ficheiro XML.

```
<?xml version='1.0' encoding='UTF-8'?>
<Categorias>
<Categoria><Nome>Comfort Bicycles</Nome> <Quantidade>537</Quantidade></Categoria>
<Categoria><Nome>Electric Bikes</Nome> <Quantidade>212</Quantidade></Categoria>
<Categoria><Nome>Road Bikes</Nome> <Quantidade>374</Quantidade></Categoria>
<Categoria><Nome>Cruisers Bicycles</Nome> <Quantidade>1378</Quantidade></Categoria>
<Categoria><Nome>Mountain Bikes</Nome> <Quantidade>1183</Quantidade></Categoria>
<Categoria><Nome>Children Bicycles</Nome> <Quantidade>782</Quantidade></Categoria>
<Categoria><Nome>Cyclocross Bicycles</Nome> <Quantidade>256</Quantidade></Categoria>
</Categorias>
```

Figura 7 - XML com as categorias

2.2.2. Job

Decidi fazer um job para esta transformação porque decidi investigar a tecnologia XSL referida pelo professor Luís Ferreira. Esta tecnologia consiste em receber um ficheiro XML e combiná-lo com um ficheiro XSL e gerar um ficheiro HTML a partir disso.

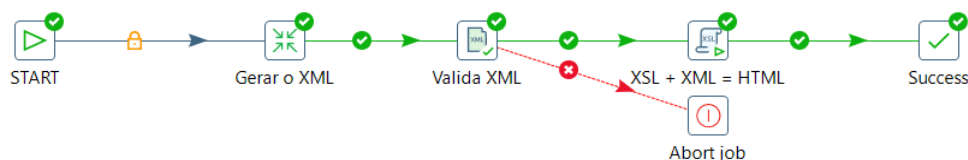


Figura 8 - Job N_Vendas/Categoria

Começamos por carregar a transformação criada no tópico anterior, validámos a mesma e depois executamos o processo XSL Transformation. Neste processo recebemos o ficheiro XML gerado pela transformação anterior e um ficheiro XSL e se os ficheiros forem válidos é gerado um ficheiro HTML.

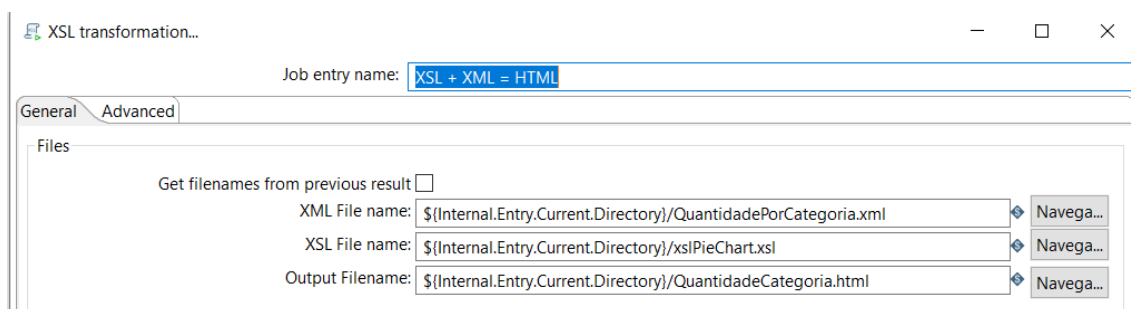


Figura 9 - XSL Transformation N_Vendas/Categoria

Decidi usar “`${Internal.Entry.Current.Directory}`” em vez dos diretórios comuns, porque se não o fizer quando for executar a transformação noutra computador ela não vai executar porque não vai ter o mesmo caminho, assim o Spoon procura os ficheiros na pasta onde está localizada a transformação.

O resultado deste Job é uma página HTML com o seguinte gráfico:

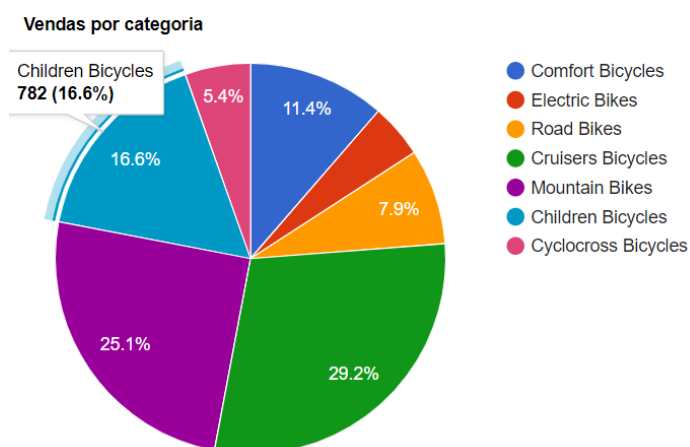


Figura 10 - PieChart N_Vendas/Categoria

2.3. Migração de dados para MongoDB

2.3.1. Transformations

O MongoDB é um tipo de base de dados NoSQL e que é utilizado em alternativa a base de dados relacionais. As bases de dados NoSQL são utilizadas para trabalhar com grandes dimensões de dados, o que não é o caso, mas fiquei curioso e decidi investigar e trabalhar sobre a mesma. No MongoDB os dados são guardados num formato muito semelhante ao JSON e não há tabelas nem linhas, mas sim coleções e documentos (respetivamente).

Como o MongoDB é uma base de dados não relacional, não carreguei a tabela de factos para a mesma, até porque não faz sentido. Além disso, devo notar, que ao fazer esta migração de dados, ou escolhemos manter os mesmos Ids das respetivas linhas e os mesmos ficam guardados dentro de um objeto no MongoDB ou deixamos o MongoDB gerar novos Ids e os outros são apagados, ou geramos novos Ids e mantemos os outros, mas os antigos perdem o “estatuto” de Id. Eu escolhi que o MongoDB gerasse novos Ids e também que mantesse os antigos.

Esta migração é um processo simples e fiz o mesmo para todas as tabelas, por isso vou apenas apresentar uma delas e explicar todo o seu processo.

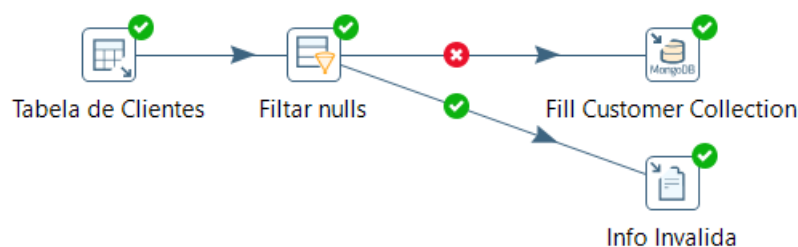


Figura 11 – Transformation ClientesToMongoDB

Comecei por receber todos os dados da tabela de clientes e trato os mesmos dados com um FilterRows que verifica se qualquer um dos valores é null. Eu escolhi não ter valores null nenhuns, por isso se existir pelo menos um valor null numa linha, a linha é completamente descartada e enviada para o ficheiro InfoInvalida. Se tudo correr bem e não for encontrado nenhum null, a informação é enviada para a coleção respetiva no MongoDB.



Figura 12 - Filtrar nulls tabela Customers

No processo “Fill Customer Collection” apenas tive que definir as seguintes configurações:

- Definir o endereço do servidor e a respectiva porta:

Step name

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

Host name(s) or IP address(es)

Port

Figura 13 - Configurações do servidor MongoDB

- Definir para que base de dados e coleção devem ser enviados os dados.

Step name

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

Database

Collection

Batch insert size

Truncate collection ☒

Figura 14 - Definições da base de dados

- Definir os respetivos campos do documento.

MongoDB Output

Step name

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	customer_key		Y	Ignore	N	N	N/A	Insert&Update
2	Customer_Name		Y	Ignore	N	N	N/A	Insert&Update
3	Email		Y	Ignore	N	N	N/A	Insert&Update
4	Street		Y	Ignore	N	N	N/A	Insert&Update
5	City		Y	Ignore	N	N	N/A	Insert&Update
6	State		Y	Ignore	N	N	N/A	Insert&Update
7	Zip_code		Y	Ignore	N	N	N/A	Insert&Update

Figura 15 - Definição dos campos do documento

2.3.2. Job

Realizei um Job simples que pega em todas as transformações de migração para MongoDB.



Figura 16 - Job MigrarDadosMongoDB

2.4. Dividir stock em categorias

2.4.1. Transformations

Para este pronto decidi criar uma transformação que dividisse os produtos consoante o seu stock atual. Estas categorias estão guardadas em diferentes páginas no Excel cada uma para a respetiva categoria (Good, Medium e Low Stock). Por fim, este ficheiro é enviado por email usando o SMTP da Google.

Comecei por receber por input do MongoDB a coleção de produtos inteira, depois no Filter Good stock assumi que todos os produtos cuja quantidade (stock) fosse superior a 15 vão para a página Good Stock do ficheiro Excel e os outros são guardados num ficheiro JSON (Porque ocupa menos espaço) para serem tratados noutra transformação.

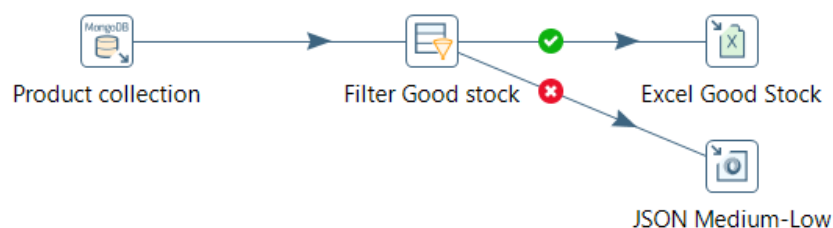


Figura 17 - Filter Good Stock Products

O processo Excel Good Stock é um Microsoft Excel Writer onde tive de definir certas configurações. Acertar nestas configurações até obter o resultado pretendido foi um bocado tentativa-erro. Tive de escolher a extensão .xls para conseguir utilizar um template para o output dos dados.

File					
Filename	\${Internal.Entry.Current.Directory}/BikeStock				Navega...
Extens	xls [Excel 97 and above]				
If output file exists	replace with new output file				
Sheet					
Sheet name (max. 31 characters)	Good Stock				
Make this the active sheet	<input checked="" type="checkbox"/>				
If sheet exists in output file	replace with new sheet				
Template					
Use template when creating new files	<input checked="" type="checkbox"/>				
Template file	\${Internal.Entry.Current.Directory}/Template.xls				Navega...
Use template when creating new sheets	<input checked="" type="checkbox"/>				
Template sheet	Template				
Hide Template Sheet	<input checked="" type="checkbox"/>				
Fields					
#	Name	Type	Format	Style from cell	Field title
1	product_key	Integer			product_key
2	product_name	String			product_name
3	brand_name	String			brand_name
4	category_name	String			category_name

Figura 18 - Configurações ExcelWriter

De seguida, tive de filtrar os produtos com stock médio. Para tal recebi o JSON gerado na transformação anterior e assumi que todos os produtos com stock superior a 5 unidades são considerados produtos com stock medio e os restantes com stock baixo e são enviados para o JSON Low Stock.

É um processo idêntico ao anterior, sendo que a única diferença está no input dos dados que em vez de ser pelo MongoDB é feito pelo JSON gerado na transformação anterior. O Excel Writer é exatamente igual ao anterior com a diferença do Sheet Name que neste caso é Medium Stock.

Tentei utilizar outro Excel Writer no lugar do JSON Low Stock, mas estava-me a dar um erro cujo não consegui resolver.

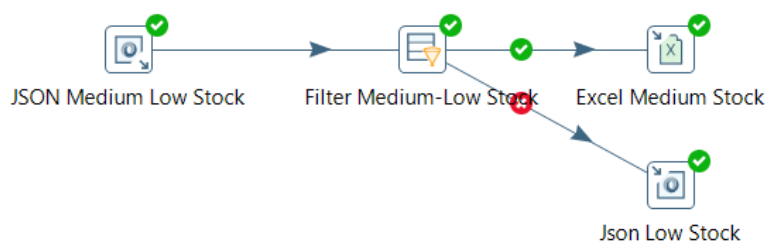


Figura 19 – Filter Medium Stock Products

Para finalizar os filtros, recebi o JSON gerado pela transformação anterior e enviei o seu conteúdo para um Excel Writer idêntico aos anteriores.



Figura 20 - Filter Low Stock Products

Com isto o processo de gerar o Excel com diferentes páginas (categorias) está finalizado e um excerto do resultado é o seguinte:

	Product_key	Product_r
	1	Trek 820 - 2016
	4	Trek Fuel EX 8 29 - 2016
	5	Heller Shagamaw Frame - 201
	10	Surly Straggler - 2016
	12	Electra Townie Original 21D -
	18	Pure Cycles Western 3-Speed
	20	Electra Townie Original 7D EQ
	Good Stock	Medium Stock Low Stock

Figura 21 - Excel resultante

Agora falta apenas a parte de enviar o ficheiro por Email. Para isso utilizei duas transformações. Uma para obter o nome do ficheiro e outra para enviá-lo por email.

Vou começar por apresentar a de obter o nome do ficheiro.

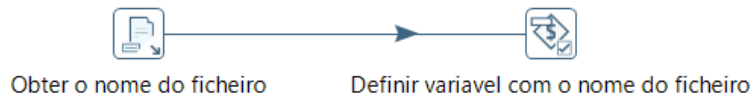


Figura 22 - Transformação para obter o nome do ficheiro

É uma transformação relativamente simples que recebe o nome do ficheiro e guarda-o numa variável.

Para enviar o email defini esta transformação:



Figura 23 - Enviar email

No processo “Dados para o Email” estão todas as informações necessárias para o próximo campo. Tive que as definir num processo Generate Rows com limite de 1 (Para enviar apenas um email) porque se as definisse diretamente em “Enviar Email” dava erro.

Os campos presentes no “Dados para o Email” são os seguintes:

Fields :

#	Nome	Tipo	Formato	Tamanho	Precis	Moe...	Decimal	Grupo	Valor	Set empty string
1	Destination Address	String							carlosmatrins@gmail.com	N
2	Sender Name	String							TP2 ISI	N
3	Sender Address	String							tp2isi2021@gmail.com	N
4	SMTP Server	String							smtp.gmail.com	N
5	PORT	String							465	N
6	Authentication User	String							tp2isi2021@gmail.com	N
7	Authentication Password	String							ipca2021	N
8	SUBJECT	String							Resultados do Stock	N
9	Comment	String							Envio em anexo o resultado dos stocks.	N

Figura 24 - Dados para o Email

Utilizei muitos recursos da Google para a realização deste trabalho prático e neste exemplo utilizei o seu serviço de SMTP.

Com isto todas as transformações estão prontas e basta agora criar um Job que as junte todas.

2.4.2. Job

Para este Job basta apenas juntar todas as transformações (começando pelas que tratam do Excel e a seguir a sequência Good -> Medium -> Low) e obter a variável com o nome do ficheiro antes de enviar o Email.

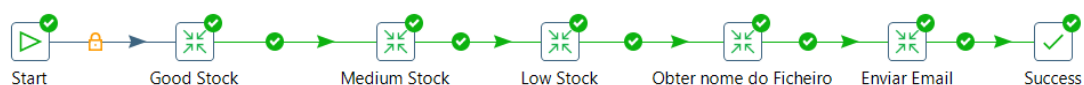


Figura 25 - Job Enviar CSV por Email

O resultado é o seguinte:

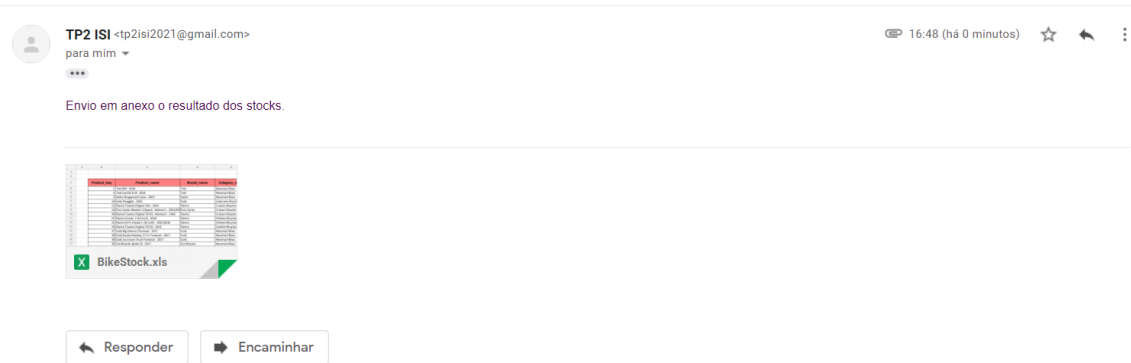
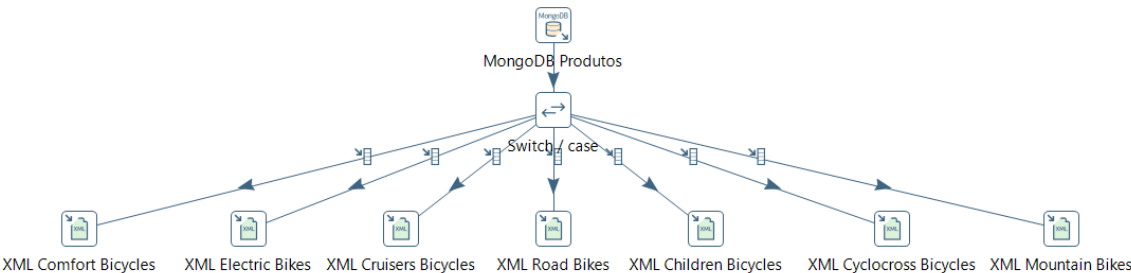


Figura 26 - Resultado JOB EnviarCSVMail

2.5. Filtrar Produtos por Categoria

2.5.1. Transformation

Para este ponto criei uma transformação que recebesse a coleção de produtos do MongoDB e que a dividisse em diferentes ficheiros XML de acordo com a sua respetiva categoria.



No processo MongoDB Produtos eu estabeleço ligação com a base de dados BikeStores e a coleção produtos. Desativo a opção de receber o input numa única variável (Para ser mais fácil de trabalhar e não ter de estar a aceder a cada elemento individualmente, ou seja, por exemplo, `[$*].category_name` para obter a categoria. Depois disso utilizo o botão “Get Fields” para obter os campos que eu quero.

MongoDB input

Step name: MongoDB Produtos

Configure connection | Input options | Query | Fields

Output single JSON field: ☐
Name of JSON output field: json

#	Name	Path	Type	Indexed values	Sample: z
1	_id	\$.id	String		
2	product_key	\$.product_key	Integer		
3	category_name	\$.category_name	String		
4	quantity	\$.quantity	Integer		
5	category_id	\$.category_id	Integer		
6	model_year	\$.model_year	Integer		
7	brand_name	\$.brand_name	String		
8	list_price	\$.list_price	Number		
9	product_name	\$.product_name	String		
10	brand_id	\$.brand_id	Integer		

Figura 27 - MongoDBProdutos Fields

De seguida configurei um processo switch case com as seguintes opções:

Switch / case

Step name: Switch / case

Field name to switch: category_name

Use string contains comparison: ☒

Case value data type: String

Case value conversion mask:

Case value decimal symbol:

Case value grouping symbol:

#	Value	Target step
1	Comfort Bicycles	XML Comfort Bicycles
2	Electric Bikes	XML Electric Bikes
3	Road Bikes	XML Road Bikes
4	Cruisers Bicycles	XML Cruisers Bicycles
5	Mountain Bikes	XML Mountain Bikes
6	Children Bicycles	XML Children Bicycles
7	Cyclocross Bicycles	XML Cyclocross Bicycles

Default target step:

Help OK Cancela

Figura 28 - SwitchCase configurations

Escolhi tratar o campo `category_name`, porque quero filtrar os produtos por categoria, depois defini cada um dos valores a obter e o respetivo ficheiro para qual devem ser enviados.

Para finalizar realizei as configurações para cada um dos ficheiros XML. A configuração para todos eles é igual com exceção do caminho. O caminho utilizado foi “`${Internal.Entry.Current.Directory}\NomeDaCategoria`” e por fim foram definidos os campos que devem estar presentes no ficheiro na aba Fields.

File Content Fields				
#	Fieldname	Element name	Content type	Type
1	_id		Attribute	String
2	category_name		Element	String
3	quantity		Element	Integer
4	model_year		Element	Integer
5	brand_name		Element	String
6	list_price		Element	Number
7	product_name		Element	String

Figura 29 - CategoryXML Fields

2.6. Listar produtos de 16-Inches

2.6.1. Transformation

O objetivo desta transformação é obter apenas os produtos que possuam “16-Inch” no nome. Esta RegEX é idêntica a fazer um SELECT dos produtos no SQL com a adição de:

```
WHERE [product_name] LIKE '%16-Inch%';
```

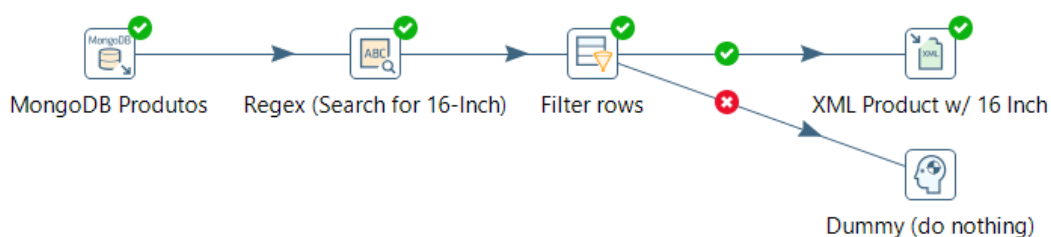


Figura 30 - Transformation FilterBy16Inch

Comecei por receber a coleção Produtos guardada no MongoDB e escolhi todos os campos presentes em cada documento (linha).

Depois utilizei uma RegEX para filtrar os resultados. A RegEX utilizada foi: `((.)+(16-Inch)(. *))`. Esta RegEX diz encontra pelo menos um caractere antes de “16-Inch” e depois disso pode ou não encontrar mais que um caractere.

De seguida filtro os resultados obtidos da seguinte maneira: Se a RegEX encontrar um resultado válido o mesmo é enviado para o ficheiro XML, se não, não é feito nada com o resultado. Decidi não fazer nada com o resultado, porque a maioria dos produtos não possuem “16-Inch” no nome e se fosse guardar os registos era armazenamento utilizado sem necessidade nenhuma.

Por fim, configurei o ficheiro XML com os respetivos campos que achei necessário estarem presentes no ficheiro tal como foi feito em alíneas anteriores.

2.7. NodeJS REST WebServices com MongoDB

Como o título diz, para esta alínea desenvolvi um conjunto de serviços REST em NodeJS.

Nunca tinha explorado NodeJS mas sempre tive uma certa curiosidade para tal e quando investiguei MongoDB descobrir que os dois possuem uma grande interatividade. Posto isto, desenvolvi 5 serviços não muito complexos e que correspondem aos principais métodos HTTP.

A pasta “models” guarda o ficheiro product.js. Este ficheiro funciona basicamente como uma classe em C# e serve para definir um objeto Product, a pasta “rotas” contém os serviços web criados e por fim o ficheiro App.js atua como uma main e é lá onde se inicia a execução do programa. Creio que tenho todos os ficheiros comentados devidamente, por isso não vou entrar em muito detalhe no relatório.

2.7.1. Método GET – Retornar todos os produtos

Comecei por desenvolver um serviço que utiliza um método GET e retorna todos os produtos presentes na coleção de produtos do MongoDB. (Para iniciar o servidor é preciso executar “npm init” no terminal)

```
router.get('/GetProducts', async (request, response) => {
  try{
    const products = await Product.find(); // Encontra os produtos
    response.json(products); // Resposta JSON com os produtos
  } catch(err)
  {
    response.json({message:err}); // Retorna erro
  }
});
```

Depois de ter o servidor a correr é só utilizar o seguinte URL:

<http://localhost:3000/Product/GetProducts>

Se tudo correr bem, o resultado (excerto) é o seguinte:

```
[{"_id":"61cdccccc799d23a80d12df9","product_key":1,"product_name":"Trek 820 - 2016","brand_name":"Trek","category_name":"Mountain Bikes","model_year":2016,"list_price":379.99,"quantity":27}, {"_id":"61cdccccc799d23a80d12dfa","product_key":2,"product_name":"Ritchey Timberwolf Frameset - 2016","brand_name":"Ritchey","category_name":"Mountain Bikes","model_year":2016,"list_price":749.99,"quantity":5}, {"_id":"61cdccccc799d23a80d12dfb","product_key":3,"product_name":"Surly Wednesday Frameset - 2016","brand_name":"Surly","category_name":"Mountain Bikes","model_year":2016,"list_price":999.99,"quantity":6},
```

Figura 31 – Get All Products

2.7.2. Método GET –Retornar um produto com um certo nome

Como o título diz este produto método retorna um certo produto dado um certo nome. Este nome é recebido no URL e o método find (Já está presente na biblioteca do NodeJS) encontra o produto.

```
// Retorna um produto com um nome especifico
router.get('/GetProducts/Name/:product_name', async(request, response) =>
{
  try {
    const product = await Product.find({product_name:
request.params.product_name}).exec(); // Encontra o produto com o nome
recebido no url
    response.json(product); // Retorna o produto encontrado
  } catch(err) {
    response.json({message: err}); // Retorna erro
  }
});
```

Para executar o método basta executar o seguinte URL:

<http://localhost:3000/Product/GetProducts/Name/NomedoProduto>

(Por exemplo:

<http://localhost:3000/Product/GetProducts/Name/Surly%20Wednesday%20Frameset%20-%20202016>)

O resultado do método indicado no exemplo é o seguinte:

```
[{"_id":"61cdccecc799d23a80d12dfb","product_key":3,"product_name":"Surly Wednesday Frameset - 2016","brand_name":"Surly","category_name":"Mountain Bikes","model_year":2016,"list_price":999.99,"quantity":6}]
```

Figura 32 - GetProduct By Name

2.7.3. Método GET – Retornar um produto com um certo ID

Este método é semelhante ao anterior, também recebe um valor por parâmetro que neste caso é o Id, mas é mais de mais simples compreensão, porque o NodeJS já possui um método que procura pelo Id.

```
router.get('/GetProducts/Id/:productId', async(request, response) => {  
  try {  
    const product = await Product.findById(request.params.productId);  
  
    response.json(product);  
  } catch(err) {  
    response.json({message: err});  
  }  
});
```

Tive um pequeno entrave neste método, porque tentei muito tempo por o método a procurar pelo Id que veio do SQL Server em vez do ID gerado automaticamente pelo MongoDB. Acabei por não conseguir pô-lo a procurar pelo ID do SQL Server, mas ficou a funcionar a procurar pelo do MongoDB.

Para executar o método basta usar o URL:

<http://localhost:3000/Product/GetProducts/Id/IdaProcurar>

(Por exemplo:

<http://localhost:3000/Product/GetProducts/Id/61cdccecc799d23a80d12df9>)

O resultado do exemplo é o seguinte:

```
{"_id":"61cdccecc799d23a80d12df9","product_key":1,"product_name":"Trek 820 - 2016","brand_name":"Trek","category_name":"Mountain Bikes","model_year":2016,"list_price":379.99,"quantity":27}
```

Figura 33 - Get Product By Id

2.7.4. Método POST – Criar um produto

O objetivo deste método é criar um produto. Como é um método post temos que utilizar uma ferramenta como o Postman.

```
router.post('/CreateProduct', async (request, response) => {

    const product = new Product({ // Criar um objeto para enviar
        product_name: request.body.product_name,
        brand_name: request.body.brand_name,
        category_name: request.body.category_name,
        model_year: request.body.model_year,
        list_price: request.body.list_price,
        quantity: request.body.quantity
    });

    try {
        const savedProduct = await product.save(); // Guardar na BD
        response.json(savedProduct);
    } catch (err) {
        response.json({message: err});
    }

});
```

É um método relativamente simples, que recebe (no request) os campos necessários para a criação do produto.

Como referi em cima, para executar este método é preciso utilizar uma ferramenta como o postman. Para executar o método pode ser algo como:

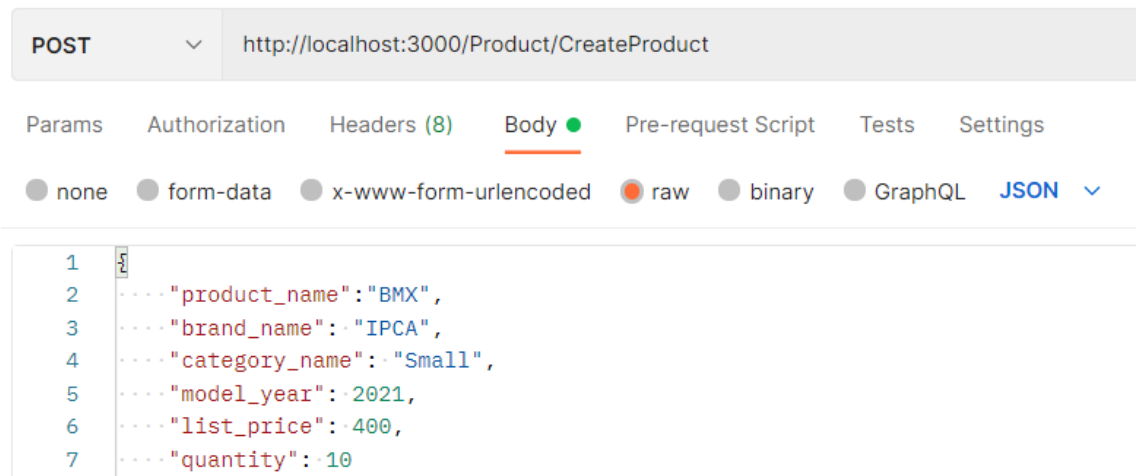


Figura 34 - CreateProduct PostMan

Se tudo correr bem, o postman deve responder com o novo produto criado.

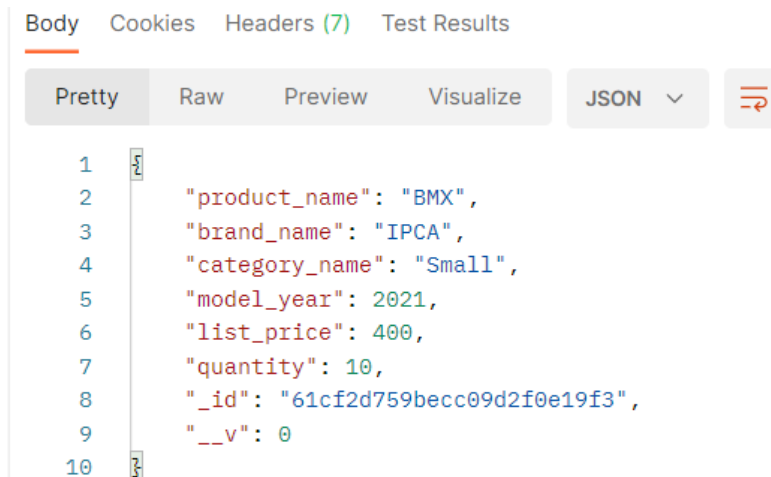


Figura 35 - Produto criado com sucesso

Agora para verificar se realmente o produto existe podemos utilizar o método de procurar por Id, ou seja:

<http://localhost:3000/Product/GetProducts/Id/61cf2d759becc09d2f0e19f3>

```
{"_id":"61cf2d759becc09d2f0e19f3","product_name":"BMX","brand_name":"IPCA","category_name":"Small","model_year":2021,"list_price":400,"quantity":10,"__v":0}
```

Figura 36 - GetProductById – Created product

2.7.5. Método DELETE – Apagar um produto pelo ID

Este método também é relativamente simples e recebe o ID de um produto por parâmetro. Se o Id for válido, o próprio apagado e é enviada uma mensagem de sucesso, se não for válido responde com uma mensagem de erro.

```
router.delete('/DeleteProduct/Id/:productId', async(request, response) =>
{
  try {
    await Product.deleteOne({ _id: request.params.productId }); // Se
    existir apaga o produto com o ID recebido
    response.json('Deleted successfully!')
  } catch(err) {
    response.json({message: err});
  }
});
```

Para executar este método também temos que utilizar o postman e para tal fazemos o seguinte:

URL: <http://localhost:3000/Product/DeleteProduct/Id/IdDoProduto>

(Por exemplo:

<http://localhost:3000/Product/DeleteProduct/Id/61cf2d759becc09d2f0e19f3>)

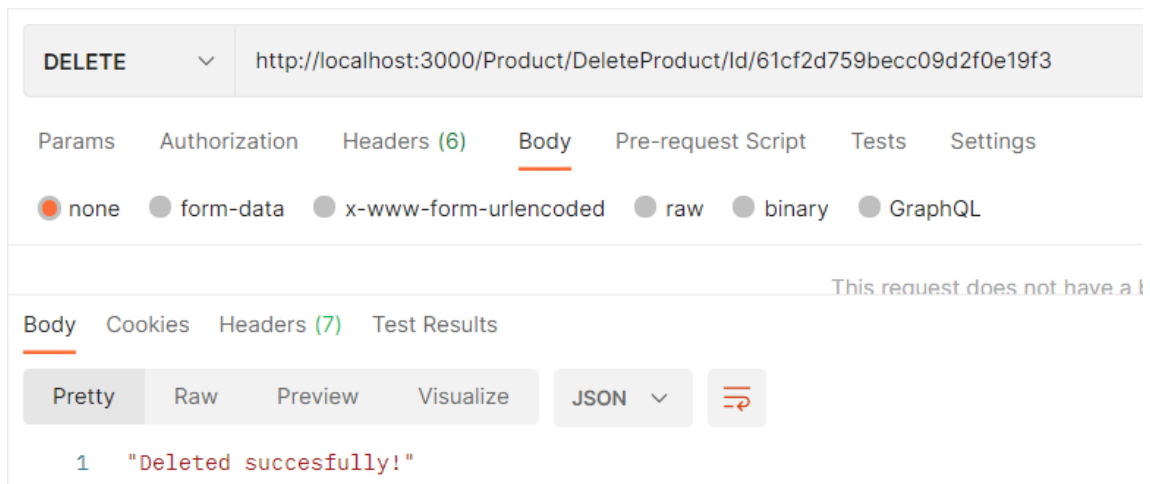


Figura 37 - DeleteProduct Method

Como é possível ver o produto foi apagado com sucesso.

2.7.6 Método PATCH – Atualizar o preço de um produto

Para fazer uma atualização utilizei um método PATCH. A razão para não ter utilizado PUT é simplesmente porque o método PATCH é o método definido para atualizações parciais e neste caso eu quero só atualizar o preço.

```
router.patch('/UpdatePrice/Id/:productId', async(request, response) => {
  try {
    const updatedPost = await Product.updateOne(
      { _id: request.params.productId }, // Encontra o produto
      { $set: { list_price: request.body.list_price } }
    ); //Atualiza-o

    response.json(updatedPost); // Retorna o produto atualizado

  } catch(err) {
    response.json({message: err}); // Erro
  }
});
```

Este método recebe o ID do produto a atualizar e depois nos headers do Postman deve ser indicado o novo preço, ou seja:

URL: <http://localhost:3000/Product/UpdatePrice/Id/IdDoProduto>

(Por exemplo: <http://localhost:3000/Product/UpdatePrice/Id/61cf30cb9becc09d2f0e19f7>)

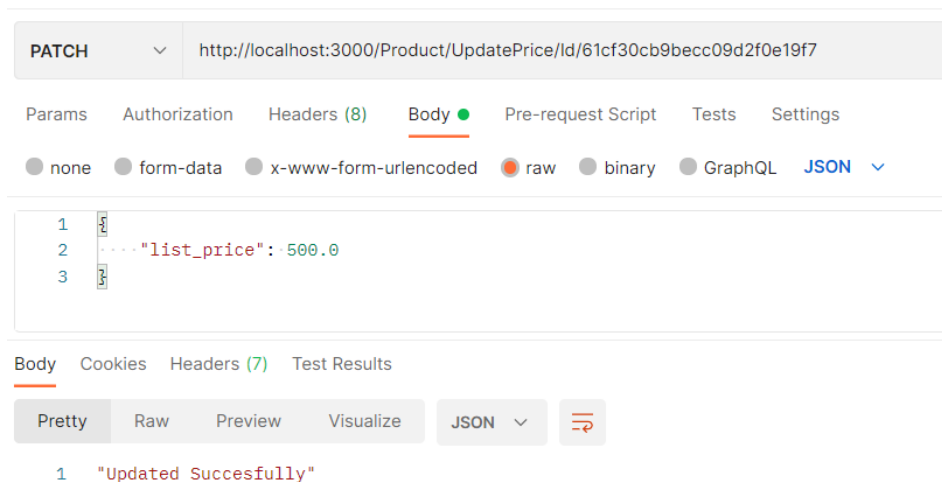


Figura 38 - UpdatePrice - PATCH Method

Com isto concluo a apresentação dos serviços Web Desenvolvidos.

2.8. Obter os produtos via Web Service

2.8.1. Transformation

Como está explícito no título, para esta transformação decidi utilizar um serviço Web que devolvesse os produtos todos e, além disso, como o formato de resposta é JSON eu converti-o em XML para no Job poder utilizar XSL mais uma vez. Para obter os produtos utilizei o primeiro serviço web indicado no tópico anterior: GetProducts.

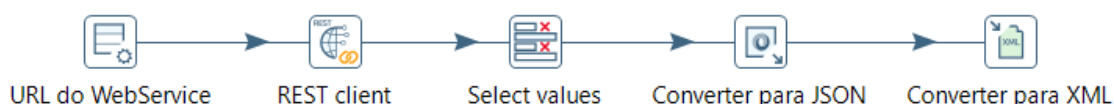


Figura 39 - Transformation GetProducts WS

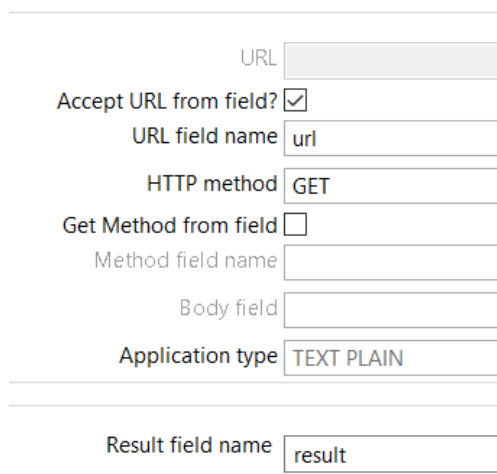
Comecei por utilizar o processo Generate Rows para gerar apenas uma linha com o URL do serviço. Este URL vai ser guardado na variável URL criada nos fields.



Figura 40 - GenerateRows GetProductsWS

No processo REST Client defini que o URL era definido por um campo anterior e que a variável era o URL e que o método ia ser um GET.

Além disso, foi definida uma variável com o resultado da execução do serviço (result).



URL

Accept URL from field? ☒

URL field name

HTTP method

Get Method from field? ☐

Method field name

Body field

Application type

Result field name

Figura 41 - Configurações REST Client

Depois, por uma questão de organização, alterei o nome de result para JSON e descartei a variável URL em Select Values. De seguida, enviei o resultado para um JSON Input e defini o path e o tipo das variáveis, ou seja:

Nome do Step Converter para JSON			
File	Content	Fields	Additional output fields
#	Name	Path	Type
1	product_key	[*].product_key	Integer
2	product_name	[*].product_name	String
3	brand_name	[*].brand_name	String
4	category_name	[*].category_name	String
5	model_year	[*].model_year	Integer
6	list_price	[*].list_price	Number
7	quantity	[*].quantity	Integer

Figura 42 - JSON Input configuration GetProductWS

Sem esta definição de path, eu não consigo gerar um XML legível, porque os valores vão estar todos dentro de uma tag Json e não separados de acordo com o que representam.

Por fim, enviei o JSON Input para um XML Output fazendo a típica configuração apresentada múltiplas vezes ao longo deste relatório.

2.8.2. Job

Para finalizar este trabalho, criei um Job cujo objetivo é transformar o XML obtido na transformação anterior num ficheiro HTML com os produtos numa tabela.



Figura 43 - Job GerarHTML WS

É um processo idêntico ao do ponto 2.2, sendo que a única diferença é o ficheiro XSL recebido pelo XSL transformation. Desta vez adaptei um template Bootstrap, que gera uma tabela, para XSL e de modo a satisfazer as necessidades do problema. O resultado (excerto) deste job é o seguinte:

Produtos						
Id	Nome	Marca	Categoria	Ano	Preço	Quantidade
1	Trek 820 - 2016	Trek	Mountain Bikes	2016	379,99	27
2	Ritchey Timberwolf Frameset - 2016	Ritchey	Mountain Bikes	2016	749,99	5
3	Surly Wednesday Frameset - 2016	Surly	Mountain Bikes	2016	999,99	6

Figura 44 - Resultado Job GerarHTML WS

3. Conclusão

Chego ao fim do semestre a fazendo uma retrospectiva da disciplina creio que foi uma disciplina trabalhosa, mas que trabalhei com todo o gosto. Foi uma disciplina em que passei algumas dificuldades mas consegui ultrapassar a maioria delas e creio que este trabalho foi uma reflexão disso mesmo.

Este trabalho foi diferente de todos os outros desenvolvidos ao longo do curso, porque foi um trabalho que não tinha um tema nem um conjunto de alíneas que eram necessárias apresentar, o que deu asas à criatividade e à investigação de novas tecnologias referidas pelo professor Luís Ferreira como, por exemplo, MongoDB e XSL e também tecnologias que descobri a investigar sobre as referidas anteriormente como, por exemplo, NodeJS.

Além disso, foi um trabalho que serviu para desenvolver várias áreas da informática e que me lembrou que informática não é só programação mas sim uma área muito abrangente com várias frentes.

Para finalizar concluo que o trabalho foi uma mais-valia muito grande porque consegui desenvolver o meu conhecimento em ferramentas como o spoon e também ao nível da programação e ainda ganhei muito conhecimento sobre novas tecnologias.

4. Bibliografia

- Powerpoints e exemplos de código disponibilizados pelo professor Luís Ferreira.
- Sebenta da disciplina.
- <https://stackoverflow.com/questions/67160064/how-can-i-get-xsl-3-0-to-work-with-pentaho-xsl-step>
- <https://anotherreeshu.wordpress.com/2015/03/05/inserting-xml-node-into-a-xml-source-data-using-pentaho-data-integration/>
- <https://www.youtube.com/watch?v=JYQliiAt2tM&t=259s>
- [https://help.hitachivantara.com/Documentation/Pentaho/8.1/Products/Data Integration/Transformation Step Reference/MongoDB Input](https://help.hitachivantara.com/Documentation/Pentaho/8.1/Products/Data_Integration/Transformation_Step_Reference/MongoDB_Input)
- <https://stackoverflow.com/questions/35833564/cant-post-param-id-in-nodejs-server-on-express>
- <https://nodejs.dev/learn/make-an-http-post-request-using-nodejs>
- https://www.w3schools.com/nodejs/nodejs_mongodb_find.asp
- <https://www.youtube.com/watch?v=vjf774RKrLc>
- <https://stackoverflow.com/questions/16739327/export-data-into-different-sheets-in-an-excel-spreadsheet-using-pentaho-kettle>
- <https://stackoverflow.com/questions/29961421/how-to-remove-duplicate-row-from-output-table-using-pentaho-di>
- <https://forums.pentaho.com/threads/230636-Converting-HTML-Content-From-JSON-to-Text-Field/>
- <https://www.youtube.com/watch?v=gs-VtcAIDU>
- <https://docs.mongodb.com/manual/release-notes/3.6-compatibility/#http-interface-and-rest-api>
- https://www.youtube.com/watch?v=wcVJsgF_pPo
-