**Grupo 2**

**Relatório do Trabalho Prático Bike on Track**

**Processamento Estruturado de Informação**

**Docentes:**

Bruno Oliveira

Marco Gomes

**Realizado por:**

Abílio Castro - 8170054

Ricardo Cardoso - 8170278

Vitor Santos - 8170312

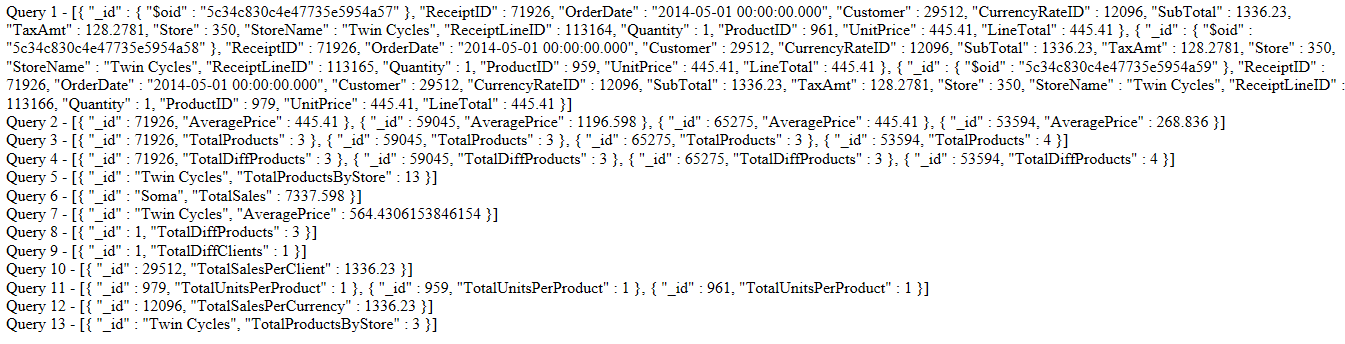
1. **Manual de utilização**

O primeiro passo para abrir a aplicação é abrir o projeto em IntelIiJ, executar o Main e esperar que execute.



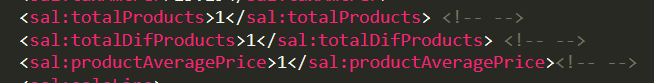
Para obter os dados, utilizar o link acima onde se modifica a store (utilizar %20 em vez de espaços), o month e o year.

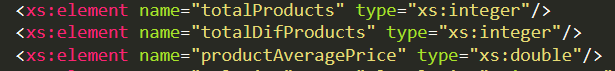
Em caso de utilização do link referido, será apresentado algo semelhante ao exemplo seguinte:

****

1. **Alterações no vocabulário**

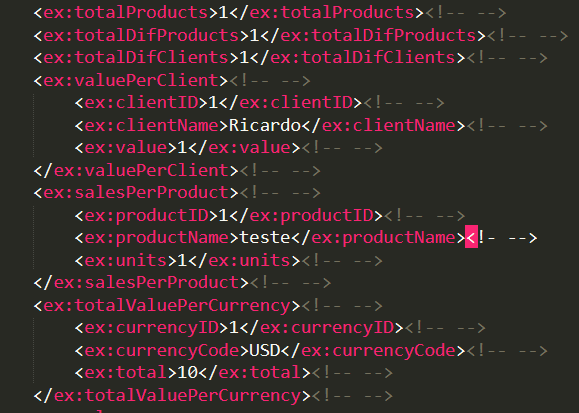
Para continuar a desenvolver o trabalho precisamos de fazer algumas alterações no nosso vocabulário nomeadamente no nosso xml final (Store) e em alguns schemas (Store, Exercise e Sale)





Em sale adicionamos campos para guardar informação do total de produtos, o total de produtos diferentes e a média do preço do produto por venda.

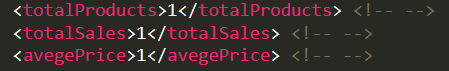
Sendo o total de produtos um inteiro, o total de produtos diferentes um inteiro e a média do preço do produto um double (número com casas decimais)

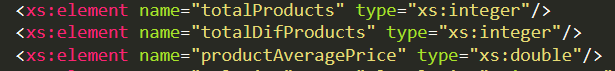




Em exercise adicionamos o total de produtos, o total de produtos diferentes, o total de clientes diferentes, o valor vendido por cliente ( onde temos o ID do cliente, o nome do cliente e o valor vendido), vendas por produto (onde temos o ID do produto, o nome do produto e o número de unidades vendidas), e o valor total por moeda ( onde temos o id da moeda, o código da moeda e o valor total de vendas com aquela moeda) durante um exercício.

Note que pode não existir limite máximo para no número de campos relativos ao valor vendido por cliente, vendas por produto e o valor total de vendas por moeda.

****

****

Em store adicionamos o total de produtos, o total de vendas e o valor médio do preço de venda dos produtos por loja.

1. **Consultas**

Antecipadamente as querys contêm a variável ‘store’ e data que provêm do link inicial.

Para casos de teste foi utilizado a store: ‘Twin Cycles’, month:5, year:2014.

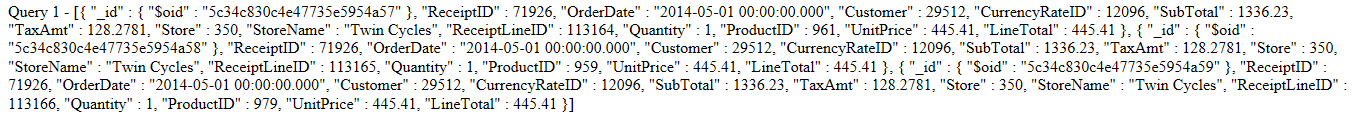
A variável data é a junção de year + “-” + month, se o month for menor que 10 é adicionado um 0 para a data ficar correta.

Estas foram as querys usadas para os desafios que está no enunciado.

**Query 1:**



Esta pesquisa retorna-nos todas as informações sobre as vendas da loja (store) num determinado exercício (data), fazemos o match da StoreName com a store e OrderDate com a data, nós decidimos estruturar com $gte e $lte, para que tenhamos um intervalo de tempo no mês e ano selecionado, isto porque as vendas têm dias diferentes e isto serve para aceitar todos os dias, este método é utilizado em todas as querys que utilizem a data.



**Query 2:**

Esta pesquisa retorna-nos a média do preço de produtos vendidos por venda, foi utilizado o método $avg que nos calcula a média e o método $group para agrupar por venda ($ReceiptID).



**Query 3:**



Esta pesquisa retorna-nos o total de produtos por venda, foi utilizado o método $sum para somar a quantidade ($Quantity) e o método $group para agrupar por venda($ReceiptID).



**Query 4:**



Esta pesquisa retorna-nos o total de produtos diferentes por venda, foram utilizados os métodos, $sum para somar a quantidade de linhas e $group para agrupar por loja ($ReceiptID).



**Query 5:**



Esta pesquisa retorna-nos o total de produtos vendidos por loja, foram utilizados os métodos, $sum para somar a quantidade ($Quantity) e $group para agrupar por loja ($StoreName).



**Query 6:**



Esta pesquisa retorna-nos o total de vendas por loja, foram utilizados os métodos, $group para agruparmos venda ($ReceiptID), $first para obter o total por venda($SubTotal), $group para juntar tudo e $sum para somar todos os totais ($first).



**Query 7:**

Esta pesquisa retorna-nos o valor médio do preço de venda dos produtos por loja, foi utilizado o método $avg que nos calcula a média de preços unitários ($UnitPrice) e o método $group para agrupar por loja ($StoreName).



**Query 8:**



Esta pesquisa retorna-nos o número de produtos diferentes no exercício, foram utilizados os métodos, $group para agrupar por produtos individuais, e depois $sum para somar todos os produtos individuais.



**Query 9:**



Esta pesquisa retorna-nos o número total de clientes por exercício, foram utilizados os métodos, $group para agrupar por cliente ($Customer), $group para juntar tudo e $sum para somar todos os clientes diferentes.



**Query 10:**

Esta pesquisa retorna-nos o valor vendido a cada cliente, foram utilizados os métodos $sum para somar o total por linha ($LineTotal), o método $group para agrupar por cliente ($Customer) e o $sort para ordenar o total de vendas a cada cliente (TotalSalesPerClient).

**Query 11:**



Esta pesquisa retorna-nos, por produto, o total de unidades vendidas, foram utilizados os métodos $sum para somar a quantidade ($Quantity), o método $group para agrupar por produto ($ProductID) e o $sort para ordenar o total de unidades vendidas por produto (TotalUnitsPerProduct).

**Query 12:**

Esta pesquisa retorna-nos o total da venda por cada moeda utilizada, foram utilizados os métodos $sum para somar o total por linha ($LineTotal), o método $group para agrupar por taxa de conversão de moeda ($CurrencyRateID) e o $sort para ordenar as vendas totais por moeda (TotalSalesPerCurrency).



**Query 13:**

Esta pesquisa retorna-nos o total de produtos por exercício, foi utilizado os métodos $sum para somar a quantidade ($Quantity) e o método $group para agrupar por loja ($StoreName).



1. **Exportação para XML**

Apesar de ainda ser um conceito vago, pretendemos importar toda a informação do mongoDB com a nossa rest API e, de seguida, concatenar numa String de modo a que seja possível recorrer à classe XSLTransformer disponibilizada no git da UC.

Para isso, reconhecemos a necessidade de criar um ficheiro xsl com as “transformationRules” que melhor se adequa ao nosso ficheiro XML.

Um dos principais desafios que prevemos enfrentar é o facto de ser impossível implementar agregações entre coleções, com isto em mente, teremos de arranjar uma maneira de tratar os JsonObjects de forma a gerar querys que relacionam toda a informação (por exemplo um ID de um produto encontrado em SalesDetails com toda a sua informação em ProductDetails).

**5.Conclusão**

Concluímos assim a segunda entrega deste trabalho prático, tendo melhorado a abordagem inicial e concluindo com sucesso o problema a resolver.

Deparámo-nos com algumas dificuldades, como por exemplo, a compreensão do enunciado.

**6.Repositório GitHub**

Repositório GitHub utilizado no auxilio ao desenvolvimento do projeto: [**https://github.com/Vmvs007/PEI**](https://github.com/Vmvs007/PEI)

**7.Bibliografia**

<https://www.mongodb.com/>

<https://stackoverflow.com/>

<https://www.google.com/docs/>