P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

David Santos (8170138)
Rui Moreira (8170286)

Ano Letivo 2018/2019

RELATÓRIO

PROCESSAMENTO ESTRUTURADO DE INFORMAÇÃO

Licenciatura em Segurança Informática e Redes de Computadores

ÍNDICE

Índice

Índice de Figuras

INTRODUÇÃO

Introdução

APRESENTAÇÃO DO PROBLEMA

Com o objetivo de obter uma visão integrada dos dados de vendas de produtos associados a cada loja, a BikeOnTrack iniciou recentemente um projeto que visa suportar de forma mais eficaz os processos de tomada de decisão relacionados com os produtos comercializados. Estes novos processos pressupõem que cada loja comunique mensalmente toda a informação relacionada com as vendas realizadas. Numa primeira fase do projeto foi desenvolvido um vocabulário XML para suportar o armazenamento de dados em documentos de auditoria. A BikeOnTrack pretende agora iniciar os esforços necessários para que os dados extraídos diretamente dos sistemas de faturação de cada loja possam ser alvo de um processo de processamento e preservação utilizando um repositório específico para que possa ser utilizada a geração automática de documentos XML suportados pelo vocabulário desenvolvido.

ESTRUTURAÇÃO DO PROBLEMA

Para ser possível planificar o problema de maneira eficaz começamos por perceber os objetivos do enunciado.

Percebemos inicialmente que queríamos que realizar algumas consultas específicas que seriam posteriormente apresentadas num ficheiro XML, todas essas consultas deveriam ser implementadas através da plataforma MongoDB que é uma Base de Dados orientada a documentos e que nos permitia a fácil importação de alguns ficheiro fornecidos para essa mesma base de dados.

Foram-nos fornecidos 4 ficheiros:

1. SalesDetails.csv que representa as vendas de todas as lojas;
2. CurrencyDetails.csv, representa todas as taxas de conversões de moeda entre os diversos países;
3. ProductDetails.csv, apresenta-nos os detalhes de cada produto;
4. ProductListPriceHistory, onde temos um *feed* das listagens dos preços de cada produto.

RESULUÇÃO E DESENVOLVIMENTO DO PROBLEMA / TRABALHO

Resulução e Desenvolvimento do problema / Trabalho

IMPORTAÇÃO DOS FICHEIROS PARA A BASE DE DADOS MONGODB

Inicialmente seria necessário termos o MongoDB instalado nos nossos computadores, mas uma vez que essa tarefa já tinha sido realizada devido a exercícios realizados ao longo da Unidade Curricular, levou-nos diretos à importação dos 4 ficheiros csv. Então para tal, utilizamos o terminal do linux para fazer a importação. É de salientar que ao fazer a importação quando referimos o nome da Base de Dados e da coleção em questão, caso elas não existam, os mesmos são criados.

```
flik@flik-pc:/usr/bin$ mongoimport -d Trabalho2 -c SalesDetails --type csv --file  /media/flik/Novo\ volume/faculdade/PEI/Trabalho2/SalesDetails.csv --headerline
2018-12-26T05:43:14.008+0000    connected to: localhost
2018-12-26T05:43:14.655+0000    imported 33323 documents
flik@flik-pc:/usr/bin$ mongoimport -d Trabalho2 -c CurrencyDetails --type csv --file  /media/flik/Novo\ volume/faculdade/PEI/Trabalho2/CurrencyDetails.csv --headerline
2018-12-26T05:44:49.645+0000    connected to: localhost
2018-12-26T05:44:49.926+0000    imported 13532 documents
flik@flik-pc:/usr/bin$ mongoimport -d Trabalho2 -c ProductDetails --type csv --file  /media/flik/Novo\ volume/faculdade/PEI/Trabalho2/ProductDetails.csv --headerline
2018-12-26T05:45:18.386+0000    connected to: localhost
2018-12-26T05:45:18.539+0000    imported 504 documents
flik@flik-pc:/usr/bin$ mongoimport -d Trabalho2 -c ProductListPriceHistory --type csv --file  /media/flik/Novo\ volume/faculdade/PEI/Trabalho2/ProductListPriceHistory.csv --headerline
2018-12-26T05:45:53.155+0000    connected to: localhost
2018-12-26T05:45:53.324+0000    imported 395 documents
```

Figura 1 – Importação dos 4 ficheiros csv para o MongoDB

Após o processo de importação dos ficheiros podemos ver dentro do terminal do MongoDB que temos todas as coleções pretendidas (Figura 2).

```
> use Trabalho2
switched to db Trabalho2
> show collections
CurrencyDetails
ProductDetails
ProductListPriceHistory
SalesDetails
>
```

Figura 2 - Coleções pertencentes à Base de Dados

MÉTODOS IMPLEMENTADOS PARA AS CONSULTAS / QUERIES

Utilizamos o comando aggregate() para fazermos as consultas precisas, entre as quais nos foram pedidas:

- O vocabulário deverá ser enriquecido com os seguintes dados relacionados com cada venda:
    - Número total de produtos;
    - Número total de produtos diferentes;
    - Valor médio do preço de venda dos produtos.
- O vocabulário deverá ser enriquecido com os seguintes dados relacionados com o exercício (conjunto de vendas):

    - Número total de produtos;
    - Número total de produtos diferentes;
    - Número total de clientes diferentes;
    - Valor vendido por cada cliente. Ordenar o valor por ordem decrescente;
    - Total de unidades vendidas por produto, por ordem decrescente;
    - Valor total da venda por cada moeda utilizada.
- Deve ainda ser possível incluir um novo conjunto de informação que apresente:
    - Total de produtos vendidos por loja;
    - Valor total das vendas por loja;
    - Valor médio do preço de venda dos produtos por loja.

O método aggregate() processa registos de dados e retorna resultados computados. Agrega os valores de um grupo de operações de vários documentos juntos e pode executar uma variedade de operações nos dados agrupados para retornar um único resultado.

Para nos orientarmos realizamos todas as consultas no MongoDB e só após isso avançamos para o serviço REST utilizando a linguagem Java.

Num ficheiro em anexo, enviamos todas as consultas implementadas no MongoDB e os respetivos comandos.

Explicação de alguns dos comandos utilizados:

- $project – esta função pega num documento que pode especificar a inclusão de campos, a supressão do campo _id, a adição de novos campos e a redefinição dos valores dos campos existentes. É possível especificar a exclusão de campos como alternativa.
- $group - agrupa documentos através de uma expressão especificada e gera um valor para cada campo. Os valores de saída contêm um campo _id que contém um campo distinto por chave e também podem conter campos processados que contêm os valores de alguma expressão. O $group não ordena os valores de saída.
- $match - filtra os valores para apresentar somente os que correspondem à(s) condição(ões) especificada(s).
- $sort - classifica todos os documentos de entrada e os retorna-os na ordem pedida (crescente / decrescente).

Para conseguirmos filtrar toda a informação e passar como parâmetro no URL, tivemos que perceber como poderiamos passar o nome da loja e a data de exercício, que neste caso se refere ao ano e mês. Para conseguirmos passar o ano e mês como parâmetro a ser lido da base de dados no MongoDB tivemos que passar todos os valores de campo OrderDate para o formato ISODate. Utilizamos uma função no terminal para aplicar esses comandos. Podemos ver na Figura 3 a função ao pormenor.

RESULUÇÃO E DESENVOLVIMENTO DO PROBLEMA / TRABALHO

```
var cursor = db.SalesDetails.find( { OrderDate: { $exists: true}});
while (cursor.hasNext()) {
   var doc = cursor.next();
   doc.OrderDate= new ISODate(doc.OrderDate);
db.SalesDetails.save(doc) ;
}
```

Figura 3 - Função de conversão do campo OrderDate para o formato ISODate

Conseguindo filtrar a informação de cada consulta a ser mostrada no serviço REST (utilizando uma extensão, no nosso caso recorremos à extensão RESTED do Firefox e ao Postman do Google Chrome), tivemos que alterar alguns parâmetros no programa fornecido e que se encontrava no repositório Git da Unidade Curricular, e conseguimos obter alguns resultados positivos.

```
@RequestMapping("/getstore")
public String getStore(@RequestParam(value="value") String value) {
    MongoConnector mongo = new MongoConnector();
    String res = mongo.getData("PEI2", "SalesDetails", "Store", value);
    return res;
}

@RequestMapping("/aggregateStoreByQueryString")
public String aggregateStoreByQueryString(@RequestParam(value="storeid") String storeID,
        @RequestParam(value="year") String year,
        @RequestParam(value="month") String month){
    //Example: %5B%7B%24group%3A%7B%22_id%22%3Anull%2Ccount%3A%7B%24sum%3A1%7D%7D%7D%5D para [{$group:{"_id":null,count:
    MongoConnector mongo = new MongoConnector();
    String query_t1_1 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},ReceiptID:1,C
    String res_t1_1 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t1_1);
    String query_t1_2 = "[{\"$project\":{year:{$year:'$OrderDate'},Store:1,month:{$month:'$OrderDate'},ReceiptID:1,Recei
    String res_t1_2 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t1_2);
    String query_t1_3= "[{\"$project\":{year:{$year:'$OrderDate'},Store:1,month:{$month:'$OrderDate'},ReceiptID:1,LineTo
    String res_t1_3 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t1_3);


    String query_t2_1 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},Quantity:1}},
    String res_t2_1 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t2_1);
    String query_t2_2 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},ProductID:1}}
    String res_t2_2 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t2_2);
    String query_t2_3 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},Costumer:1}},
    String res_t2_3 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t2_3);
    String query_t2_4 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},LineTotal:1,C
    String res_t2_4 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t2_4);
    String query_t2_5 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},Quantity:1,Pr
    String res_t2_5 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t2_5);
    String query_t2_6 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},CurrencyRateI
    String res_t2_6 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t2_6);


    String query_t3_1 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},StoreName:1,C
    String res_t3_1 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t3_1);
    String query_t3_2 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},StoreName:1,L
    String res_t3_2 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t3_2);
    String query_t3_3 = "[{\"$project\":{year:{$year:\"$OrderDate\"},Store:1,month:{$month:\"$OrderDate\"},StoreName:1,L
    String res_t3_3 = mongo.aggregateDataByQueryString("PEI2", "SalesDetails", query_t3_3);

    return res_t1_1 + "\n\n" + res_t1_2 + "\n\n" + res_t1_3 + "\n\n" + res_t2_1 + "\n\n" + res_t2_2 + "\n\n" + res_t2_3
}
```

Figura 4 - Programa em Java que permite fazer a extração das consultas

RESULUÇÃO E DESENVOLVIMENTO DO PROBLEMA / TRABALHO

Como podemos ver na Figura 4, sendo que é apenas um excerto do programa em si, todas as consultas são feitas, e ao executar são enviadas para o serviço Postman onde podemos ver os resultados das consultas (Figura 5).

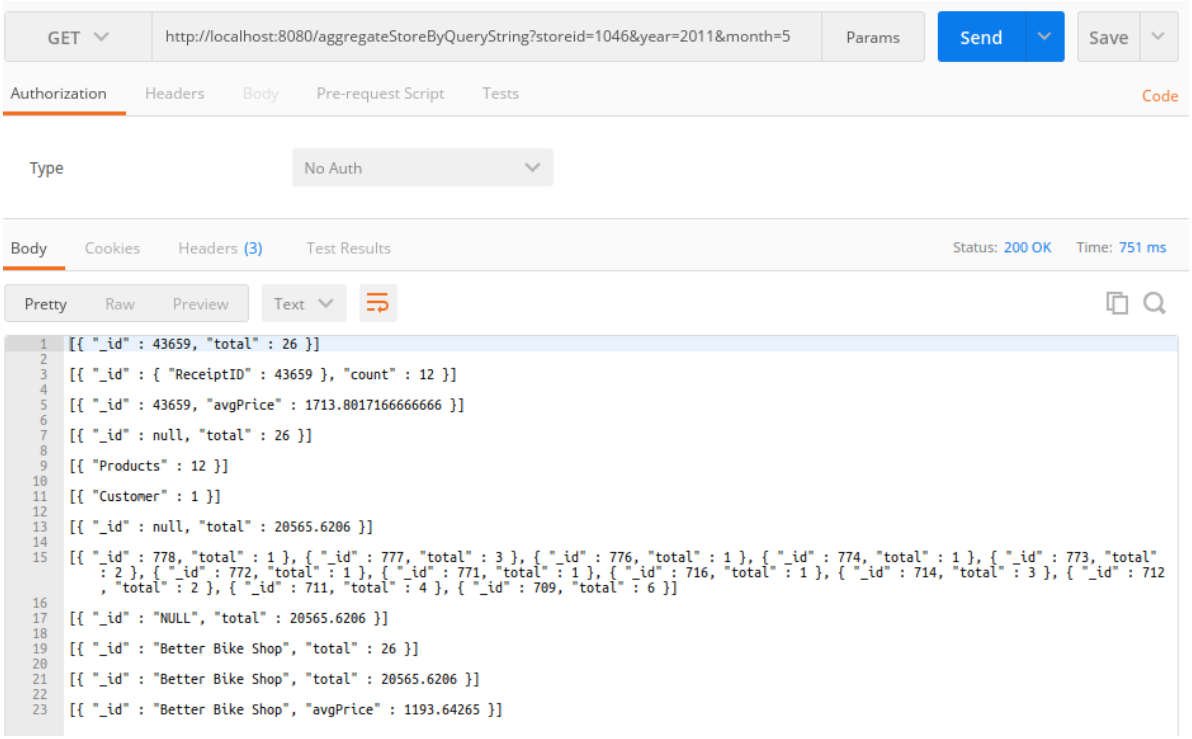

Figura 5 - Todas as consultas apresentadas no serviço Postman

Para validarmos sempre ficheiro a ficheiro se o funcionamento era o correto utilizamos o XMLValidator fornecido no Repositório da Unidade Curricular e fomos aplicando sempre a matéria abordada nas aulas.

Criamos regras específicas para cada um dos ficheiros dos quais iremos falar mais a frente, dependendo do campo e dos valores que recebia, moldamos para que seja aceite somente esse tipo de ficheiro tais como nomes, números ou até mesmo e-mail.

CONCLUSÕES

Conclusões

BIBLIOGRAFIA

- https://stackoverflow.com/
- https://www.tutorialspoint.com/mongodb/
- https://www.mongodb.com/
- https://github.com/PEI-18-19
- Matéria dada nas aulas da Unidade Curricular

CONCLUSÕES

Inicialmente não tínhamos percebido o objetivo desta parte do trabalho, o que seria suposto obter como resultado final, etc, mas após um desenvolvimento e abordagem para tirar dúvidas com os docentes da Unidade Curricular, foi-nos possível perceber o objetivo final.

Após a realização da parte 1 do trabalho, fazia sentido termos a possibilidade de manipular os dados que queriamos e formata-los de maneira organizada, e foi isso mesmo que aconteceu. Foi-nos possível através de diversos métodos organizar toda a informação através da utilização de consultas.

Embora tenhamos conseguido fazer talvez um dos pontos mais importantes do trabalho, é notável que temos alguns pontos em falta, entre os quais a mudança da coluna ListPrice em ProductDetails.csv pela coluna ListPrice do ficheiro ProductListPriceHistory.csv. Consideramos que temos alguns pontos para serem melhorados numa fase futura.

Em suma, consideramos ter feito um bom trabalho, pois fomos capazes de implementar alguns serviços mais essenciais usando a linguagem Java e o MongoDB.