



ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

# Processamento Estruturado de Informação

Licenciatura em Engenharia Informática

Licenciatura em Segurança Informática em Redes de Computadores

2020/2021

Trabalho Prático 2

Christopher Meder – 8170022

Hugo Figueiredo – 8170026

Miguel Ribeiro - 8170538

## Índice

Índice de Tabelas .....	3
Índice de Figuras.....	3
Introdução .....	4
Estrutura .....	5
Criação das Coleções .....	9
ProductPrices .....	9
SalesWCP .....	12
Receipt .....	15
Consultas .....	19
Consulta 1 .....	19
Consulta 2 .....	20
Consulta 3 .....	21
Consulta 4 .....	22
Consulta 5 .....	23
Consulta 6 .....	24
Mongo Atlas .....	25
Charts.....	26
Scripts.....	28
Conclusão .....	29

## Índice de Tabelas

Tabela 1 - SalesDetails .....	5
Tabela 2 - CurrencyDetails .....	5
Tabela 3 - ProductDetails.....	6
Tabela 4 - ProductLPH .....	6
Tabela 5 - CustomerDetails .....	7
Tabela 6 - ProductPrices.....	7
Tabela 7 - SalesWCP .....	8
Tabela 8 - Receipt.....	8

## Índice de Figuras

Figura 1 - ProductPrices – Original .....	10
Figura 2 - ProductPrices – lookup ProductLPH .....	10
Figura 3 - ProductPrices - project.....	11
Figura 4 - ProductPrices – unwind .....	11
Figura 5 - SalesWCP - Original .....	13
Figura 6 - SalesWPC - loockup CustomerDetails .....	13
Figura 7 - SalesWCP - lookup ProductPrices .....	13
Figura 8 - SalesWCP - lookup CurrencyDetails.....	14
Figura 9 - SalesWCP - limit.....	14
Figura 10 - Receipt - Original .....	16
Figura 11 - Receipt - unwind Customers .....	16
Figura 12 - Receipt - unwind Currency.....	17
Figura 13 - Receipt - unwind Product.....	17
Figura 14 - Receipt - project.....	17
Figura 15 - Receipt - group .....	18
Figura 16 - Receipt - project.....	18
Figura 17 - Consulta 1 .....	19
Figura 18 - Consulta 2 .....	20
Figura 19 - Consulta3 .....	21
Figura 20 - Consulta 4 .....	22
Figura 21 - Consulta 5 .....	23
Figura 22 - Consulta 6 .....	24
Figura 23 - MongoAtlas.....	25
Figura 24 - MongoAtlas - Users .....	25
Figura 25 - MongoAtlas - Connect .....	25
Figura 26 - Mongo Atlas - Charts .....	26
Figura 27 - Scripts – Coleções.....	28
Figura 28 - Scripts - Consultas.....	28

## Introdução

No âmbito da disciplina de Processamento Estruturado de Informação, foi realizado um trabalho, no caso a “BikeOnTrack”, com o objetivo de obter uma visão integrada dos dados de vendas de produtos iniciou recentemente um projeto que visa suportar de forma mais eficaz os processos de tomada de decisão relacionados com os produtos comercializados. Este processo envolve a recolha e tratamento de dados para suportar a análise e visualização de mesmos de acordo com as necessidades de análise.

Foi efetuada a criação de uma base de dados orientada por documentos utilizando MongoDB de forma a satisfazer as necessidades de estruturação e organização de dados com a utilização de boas práticas de modelação apresentadas ao longo do semestre para a correta estruturação dos dados utilizando documentos e coleções. Todo o processo encontra-se documentado no presente documento.

## Estrutura

Em seguida é apresentada a estrutura das coleções utilizadas e uma breve explicação da sua composição.

### SalesDetails

Coleção fornecida inicialmente.

Nome do Campo	Descrição
<b>ReceiptID</b>	Código da venda. Cada <i>ReceiptID</i> pode surgir várias vezes no mesmo documento. Cada linha representa uma linha de venda.
<b>OrderDate</b>	Data da venda. Apesar de incluir a hora/minuto/segundo, esses valores não contêm informação útil e como tal devem estar descartados
<b>Customer</b>	Código do cliente.
<b>CurrencyRateID</b>	Código da taxa de câmbio utilizada no negócio. Se o valor for NULL, significa que o pagamento foi realizado em dólares.
<b>SubTotal</b>	Total da fatura.
<b>TaxAmt</b>	Valor pago em impostos.
<b>Store</b>	Código da loja.
<b>StoreName</b>	Nome da loja.
<b>ReceiptLineID</b>	Código da linha de venda.
<b>Quantity</b>	Quantidade vendida do produto para a linha de venda.
<b>ProductID</b>	Código do produto.
<b>UnitPrice</b>	Preço unitário do produto.
<b>LineTotal</b>	Total da linha (preço do produto * Preço unitário).

Tabela 1 - SalesDetails

### CurrencyDetails

Coleção fornecida inicialmente.

Nome do Campo	Descrição
<b>CurrencyRateID</b>	Código único da taxa de câmbio.
<b>CurrencyRateDate</b>	Data da atualização da taxa.
<b>FromCurrencyCode</b>	Moeda de origem.
<b>toCurrencyCode</b>	Moeda de destino.
<b>RateVal</b>	Valor da taxa de câmbio.

Tabela 2 - CurrencyDetails

## ProductDetails

Coleção fornecida inicialmente.

Nome do Campo	Descrição
<b>ProductID</b>	Código único do produto
<b>Name</b>	Nome do produto
<b>ProductNumber</b>	Código de negócio do produto
<b>Color</b>	Cor do produto (se aplicável)
<b>ListPrice</b>	Preço de venda
<b>SellStartDate</b>	Data início em que o produto entrou em comercialização
<b>SellEndDate</b>	Data de fim (caso de aplique) em que o produto deixou de ser comercializado

*Tabela 3 - ProductDetails*

## ProductLPH (ProductListPriceHistory)

Coleção fornecida inicialmente.

Nome do Campo	Descrição
<b>ProductID</b>	Código único do produto.
<b>StartDate</b>	Data de início em que o preço do produto se aplicou. Apesar de incluir a hora/minuto/segundo, esses valores não contêm informação útil e como tal devem estar descartados.
<b>EndDate</b>	Data de fim em que o preço do produto se aplicou. Apesar de incluir a hora/minuto/segundo, esses valores não contêm informação útil e como tal devem estar descartados. NULL indica que o preço ainda se encontra ativo.
<b>ListPrice</b>	Preço de venda.
<b>ModifiedDate</b>	Data de modificação. Apesar de incluir a hora/minuto/segundo, esses valores não contêm informação útil e como tal devem estar descartados.

*Tabela 4 - ProductLPH*

## CustomerDetails

Coleção fornecida inicialmente.

Nome do Campo	Descrição
<b>CustomerKey</b>	Código único.
<b>CustomerAlternateKey</b>	Código alternativo.
<b>Title</b>	Título (se existir).
<b>FirstName</b>	Primeiro nome.
<b>MiddleName</b>	Nome do “meio”.
<b>LastName</b>	Último nome.
<b>BirthDate</b>	Data de nascimento.
<b>MaritalStatus</b>	Estado civil.
<b>Gender</b>	Gênero.
<b>EmailAddress</b>	Endereço de email.
<b>Education</b>	Habilitações literárias.
<b>Occupation</b>	Ocupação.
<b>AddressLine1</b>	Morada
<b>AddressLine2</b>	Morada
<b>PhonePhone</b>	Número de telefone
<b>DateFirstPurchase</b>	Data da primeira compra

*Tabela 5 - CustomerDetails*

## ProductPrices

Coleção igual à anterior apresentada “ProductDetails”, a anterior não apresentava valores no campo anteriormente intitulado de “ListPrice”. A seguinte coleção já apresenta valor no campo agora “Price”, esse mesmo valor foi retirado da coleção “ProductLPH” efetuando uma relação através do campo “ProductID”.

Nome do Campo	Descrição
<b>ProductID</b>	Código único do produto
<b>Name</b>	Nome do produto
<b>ProductNumber</b>	Código de negócio do produto
<b>Color</b>	Cor do produto (se aplicável)
<b>Price</b>	Preço de venda
<b>SellStartDate</b>	Data início em que o produto entrou em comercialização
<b>SellEndDate</b>	Data de fim (caso de aplique) em que o produto deixou de ser comercializado

*Tabela 6 - ProductPrices*

## SalesWCP

Combinação das coleções "CustomerDetails", "ProductPrices", "SalesDetails" e "CurrencyDetails". Reunir a informação toda numa coleção para facilitar o tratamento de dados.

Nome do Campo	Descrição
<b>ReceiptID</b>	Código da venda. Cada ReceiptID pode surgir várias vezes no mesmo documento.
<b>OrderDate</b>	Data da venda. Apesar de incluir a hora/minuto/segundo, esses valores não contêm informação útil e como tal devem estar descartados
<b>Customer</b>	Array com a informação do cliente.
<b>SubTotal</b>	Total da fatura
<b>TaxAmt</b>	Valor pago em impostos
<b>Store</b>	Código da loja.
<b>StoreName</b>	Nome da loja.
<b>ReceiptLineID</b>	Código da linha de venda.
<b>Quantity</b>	Quantidade vendida do produto para a linha de venda.
<b>UnitPrice</b>	Preço unitário do produto
<b>LineTotal</b>	Total da linha (preço do produto * Preço unitário)
<b>Product</b>	Array com a informação do produto.
<b>Currency</b>	Array com a informação da taxa de câmbio se aplicável.

Tabela 7 - SalesWCP

## Receipt

Coleção criada para simular uma fatura com toda a informação relevante de forma organizada para facilitar a sua consulta. Todos documentos com o mesmo código de venda foram agrupados num só documento ficando assim toda a informação disposta num só documento.

Nome do Campo	Descrição
<b>ReceiptID</b>	Código da venda. Cada ReceiptID pode surgir várias vezes no mesmo documento.
<b>OrderDate</b>	Data da venda. Apesar de incluir a hora/minuto/segundo, esses valores não contêm informação útil e como tal devem estar descartados.
<b>Store</b>	Código da loja.
<b>StoreName</b>	Nome da loja.
<b>Customer</b>	Array com a informação do cliente.
<b>Products</b>	Array com a informação dos produtos.
<b>Currency</b>	Array com a informação da taxa de câmbio se aplicável.
<b>TaxAmt</b>	Valor pago em impostos.
<b>SubTotal</b>	Total da fatura.

Tabela 8 - Receipt



## Criação das Coleções

Ao carregar as coleções fornecidas toda a informação é recebida como string e é alterada manualmente na aplicação para o respetivo tipo de dado como por exemplo date,int ou decimal. As coleções exportadas em anexo já se encontram no tipo de dados correto.

Apenas o campo “SellEndDate” foi alterado posteriormente o seu tipo de dado de string para date pois apresentava demasiados campos “null” e se fosse selecionado o tipo de campo manualmente iria substituir “null” pela menor dada reconhecida por a aplicação.

## ProductPrices

A pipeline apresentada em seguida foi utilizada para criar a coleção “ProductPrices”. Esta coleção é uma junção da coleção “ProductDetails” e “ProductLPH” visto a coleção “ProductDetails” apresentar o preço a 0.00 e o preço do produto se encontrar na coleção “ProductLPH”.

```
var productprices = db.ProductDetails.aggregate([
  { "$lookup": {from: 'ProductLPH', localField: 'ProductID', foreignField: 'ProductID',
as: 'Prices' }},
  { "$project":{   "ProductID": 1, "Name": 1, "ProductNumber": 1, "Color": 1, "Prices":
"$Prices.Price", "SellStartDate": 1,"SellEndDate": 1}},
  {$unwind:{ path: "$Prices"}}]).toArray()

productprices.forEach(
function(doc){
    db.ProductPrices.save(doc);
})
```

A utilização do operador “\$lookup” efetua a junção da informação de uma coleção com outra, neste caso importa a informação da coleção “ProductLPH” onde fizemos a relação através do campo “ProductID” que é comum em ambas as coleções, a informação fica armazenada num campo chamado de “Prices”.

A utilização do operador “\$project” é utilizado para selecionar os campos a serem apresentados e a sua disposição, pode ser também utilizado para redefinir nomes ou valores de campos ou suprimir os mesmos.

Com o operador “\$unwind” utilizado para decompor o array da informação.

Com a última operação é criada a coleção “ProductPrices” com a informação executada a cima dentro da variável “productprices”.

O comando apresentado de seguida é referente à alteração do tipo de dados do campo “SellEndDate” de string para date.

```
db.ProductPrices.find({"SellEndDate":{"$ne":"NULL"}}).forEach(function(doc) {  
    doc.SellEndDate= ISODate(doc.SellEndDate);  
    db.ProductPrices.save(doc);  
})
```

As imagens que se seguem são amostras dos operadores utilizados passo a passo de forma a exemplificar a sua utilização.

```
_id: ObjectId("600b1635c8ab44a95cb260c8")  
ProductID: 1  
Name: "Adjustable Race"  
ProductNumber: "AR-5381"  
Color: "NULL"  
ListPrice: 0.00  
SellStartDate: 2008-04-29T23:00:00.000+00:00  
SellEndDate: "NULL"
```

Figura 1 - ProductPrices – Original

```
1 /**  
2  * from: The target collection.  
3  * localField: The local join field.  
4  * foreignField: The target join field.  
5  * as: The name for the results.  
6  * pipeline: The pipeline to run on the joined colle  
7  * let: Optional variables to use in the pipeline fi  
8  */  
9 {  
10   from: 'ProductLPH',  
11   localField: 'ProductID',  
12   foreignField: 'ProductID',  
13   as: 'Prices'  
14 }
```

Output after \$lookup stage (Sample of 20 documents)

```
_id: ObjectId("600b1635c8ab44a95cb260c8")  
ProductID: 1  
Name: "Adjustable Race"  
ProductNumber: "AR-5381"  
Color: "NULL"  
ListPrice: 0.00  
SellStartDate: 2008-04-29T23:00:00.000+00:00  
SellEndDate: "NULL"  
Prices: Array
```

Figura 2 - ProductPrices – lookup ProductLPH

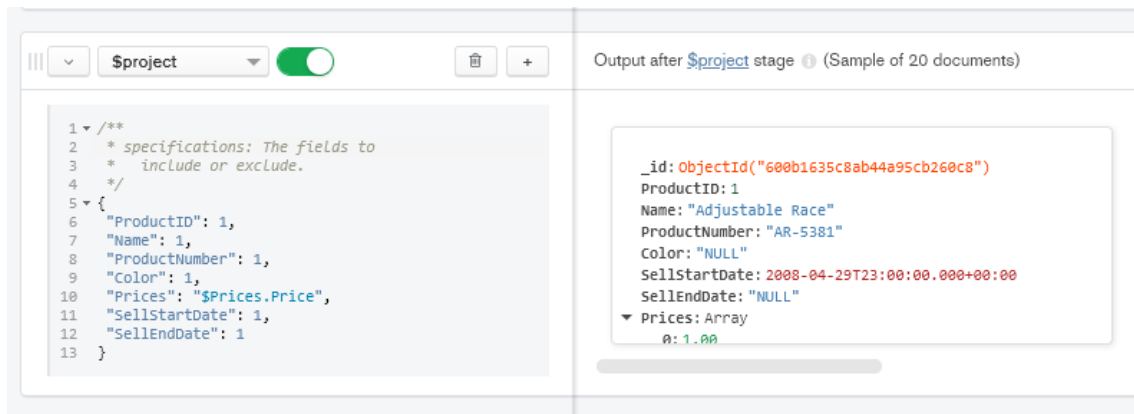


Figura 3 - ProductPrices - project

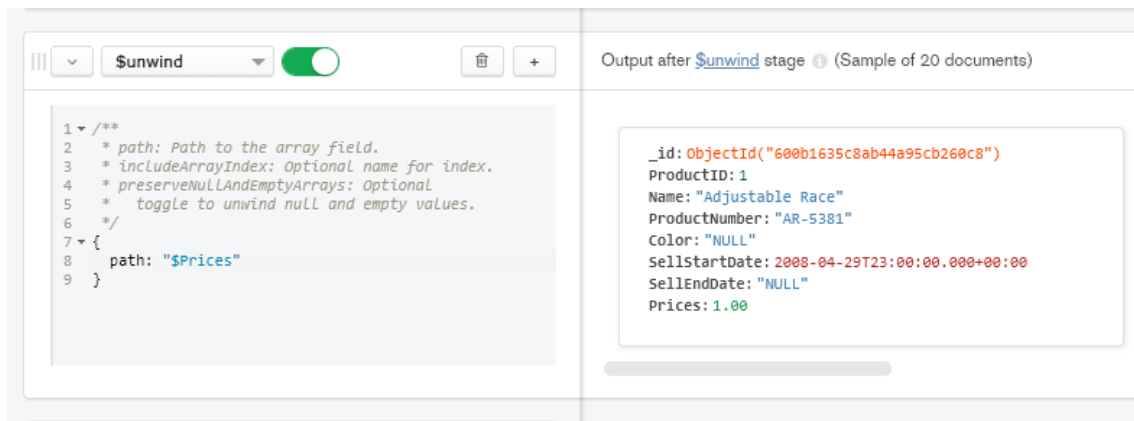


Figura 4 - ProductPrices – unwind

A pipeline apresentada em seguida foi utilizada para criar a coleção “SalesWCP”. Esta coleção é uma junção das coleções “CustomerDetails”, “ProductPrices”, “SalesDetails” e “CurrencyDetails”. A criação da mesma tem o intuito de juntar a informação da venda, dos produtos, do cliente e da moeda num só documento.

O nome “SalesWCP” é referente a Sales With Customers and Prices.

```
var saleswcp = db.SalesDetails.aggregate([
  {$lookup: {from: 'CustomerDetails', localField: 'Customer', foreignField:
'CustomerKey', as: 'Customer' }},
  {$lookup: {from: 'ProductPrices', localField: 'ProductID', foreignField: 'ProductID',
as: 'Product' }},
  {$lookup: {from: 'CurrencyDetails',localField: 'CurrencyRateID',foreignField:
'CurrencyRateID',as: 'Currency'}},
  {$project: {"ReceiptID" : 1,"OrderDate" : 1,"Customer" : 1,"Currency": 1,"SubTotal" :
1,"TaxAmt" : 1,"Store" : 1,"StoreName" : 1,"ReceiptLineID" : 1,"Quantity" :
1,"Product" : 1,"UnitPrice" : 1,"LineTotal" : 1}},
  {$limit: 1000})).toArray()

saleswcp.forEach(
function(doc){
    db.SalesWCP.insertOne(doc);
})
```

A utilização do primeiro operador “\$lookup” importa a informação da coleção “CustomerDetails” onde fizemos a relação através do campo “CustomerKey” que é comum em ambas as coleções, a informação fica armazenada num campo chamado de “Customer”.

A utilização do segundo operador “\$lookup” importa a informação da coleção “ProductPrices” onde fizemos a relação através do campo “ProductID” que é comum em ambas as coleções, a informação fica armazenada num campo chamado de “Product”.

A utilização do último operador “\$lookup” importa a informação da coleção “CurrencyDetails” onde fizemos a relação através do campo “CurrencyRateID” que é comum em ambas as coleções, a informação fica armazenada num campo chamado de “Currency”.

O operador “\$project” é utilizado para selecionar os campos a serem apresentados e a sua disposição.

O “\$limit” foi utilizado para limitar os campos em que estamos a trabalhar pois o servidor não apresenta recursos suficientes para trabalhar com uma extensa quantidade de dados.

Com a última operação é criada a coleção “SalesWCP” com a informação executada a cima dentro da variável “saleswcp”.

As imagens que se seguem são amostras dos operadores utilizados passo a passo de forma a exemplificar a sua utilização.



Figura 5 - SalesWCP - Original

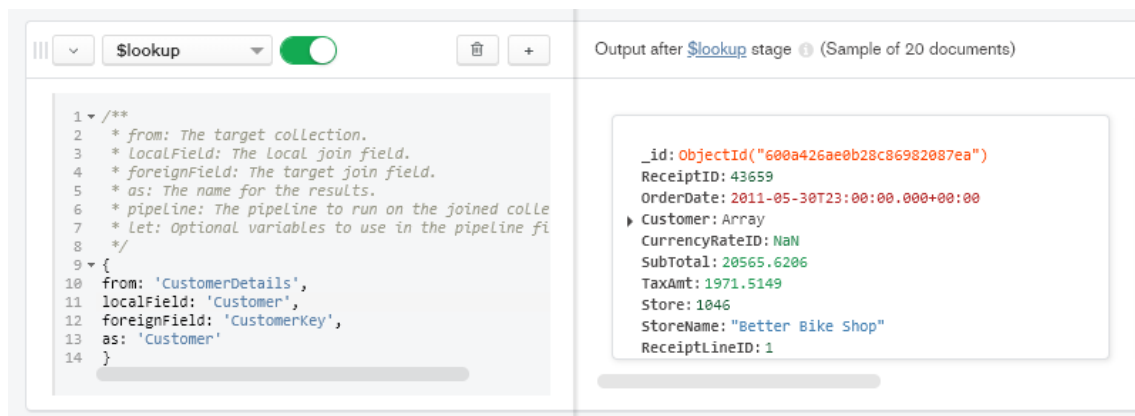


Figura 6 - SalesWPC - loockup CustomerDetails

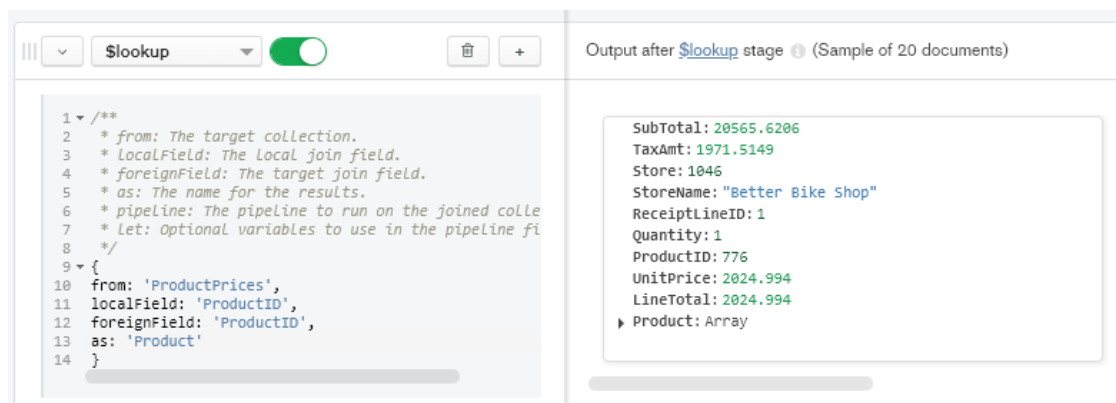


Figura 7 - SalesWCP - lookup ProductPrices



## Receipt

A pipeline apresentada em seguida foi utilizada para criar a coleção “Receipt”. Esta coleção é uma junção das coleções “CustomerDetails”, “ProductPrices”, “SalesDetails” e “CurrencyDetails”. A criação da mesma tem o intuito de juntar a informação da venda, dos produtos, do cliente e da moeda num só documento, a coleção anterior apresentava diversas faturas com o mesmo código separando os produtos por fatura, esta coleção visa juntar toda a informação de uma Receipt num só documento simulando assim uma fatura com toda a informação.

```
var receipt = db.SalesWCP.aggregate([
  {$unwind:{path: "$Customer"}},
  {$unwind:{path: "$Currency"}},
  {$unwind:{path: "$Product"}},
  {$project:{"ReceiptID": "$ReceiptID","OrderDate": "$OrderDate","Store":
"$Store","StoreName": "$StoreName","Customer": {"CustomerKey":
"$Customer.CustomerKey", "FirstName": "$Customer.FirstName","MiddleName":
"$Customer.MiddleName","LastName": "$Customer.LastName", "Gender": "$Customer
.Gender","Email": "$Customer.EmailAddress","AddressLine1": "$Customer.AddressLin
e1","AddressLine2": "$Customer.AddressLine2","Phone": "$Customer.Phone","DateFir
stPurchase": "$Customer.DateFirstPurchase"},"Products": {"ID": "$Product.ProductID", "
Name": "$Product.Name", "Number": "$Product.Name", "Color": "$Product.Color", "Quant
ity": "$Quantity", "Price": "$Product.Prices", "UnitPrice": "$UnitPrice", "LineTotal":
"$LineTotal", "SellStartDate": "$Product.SellStartDate", "SellEndDate": "$Product.SellEn
dDate"},"Currency": "$Currency", "TaxAmt": "$TaxAmt", "SubTotal": "$SubTotal"}},
  {$group: {"_id": {"ReceiptID": "$ReceiptID", "OrderDate": "$OrderDate", "Store": "$Store", "
StoreName": "$StoreName", "Customer": "$Customer", "Currency": "$Currency", "TaxAm
t": "$TaxAmt", "SubTotal": "$SubTotal"}, "Products": {"$push": "$Products"}}},
  {$project: {"_id": 0, "ReceiptID": "$_id.ReceiptID", "OrderDate": "$_id.OrderDate", "Store":
"$_id.Store", "StoreName": "$_id.StoreName", "Customer": "$_id.Customer", "Products":
"$Products", "Currency": "$_id.Currency", "TaxAmt": "$_id.TaxAmt", "SubTotal": "$_id.Su
bTotal"}}}).toArray()
receipt.forEach(
function(doc){
  db.Receipt.save(doc);
})
```

Com o operador “\$unwind” utilizado inicialmente serviu para decompor 3 campos que apresentavam a informação como array e pretendeu-se passar a mesma para object principalmente no campo “Product” devido a apresentar um array com vários arrays dentro, ficando assim a apresentar apenas um array com todos os objects dentro.

O operador "\$project" é utilizado para selecionar os campos a serem apresentados e a sua disposição.

O "\$group" foi utilizado pois existiam diversas receipts com o mesmo id, cada uma com um produto, ficando assim apenas a existir um único documento referente aquele "ReceiptID" com todos os produtos dentro do mesmo.

O operador "\$project" foi novamente utilizado para selecionar os campos a serem apresentados e a sua disposição.

Com a última operação é criada a coleção "Receipt" com a informação executada a cima dentro da variável "receipt".

As imagens que se seguem são amostras dos operadores utilizados passo a passo de forma a exemplificar a sua utilização.

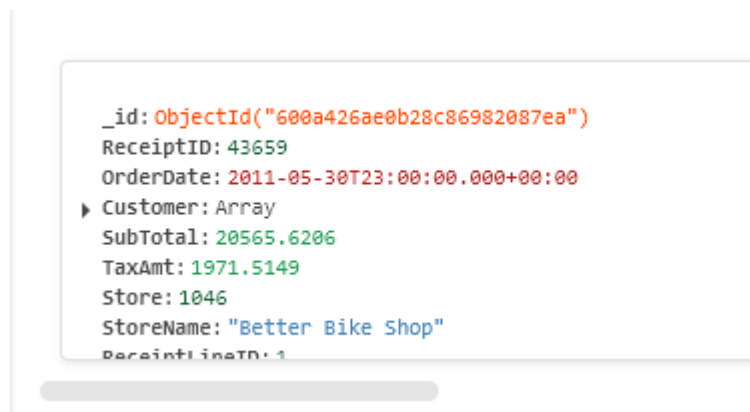


Figura 10 - Receipt - Original

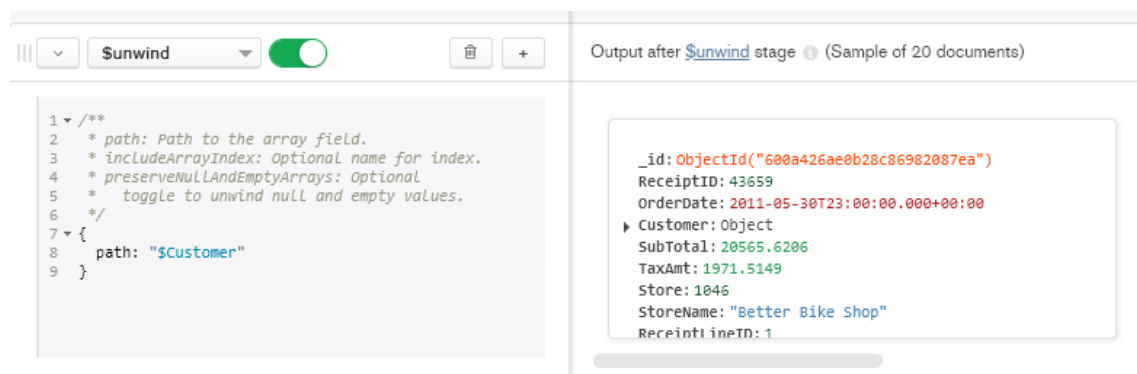


Figura 11 - Receipt - unwind Customers







## Consultas

### Consulta 1

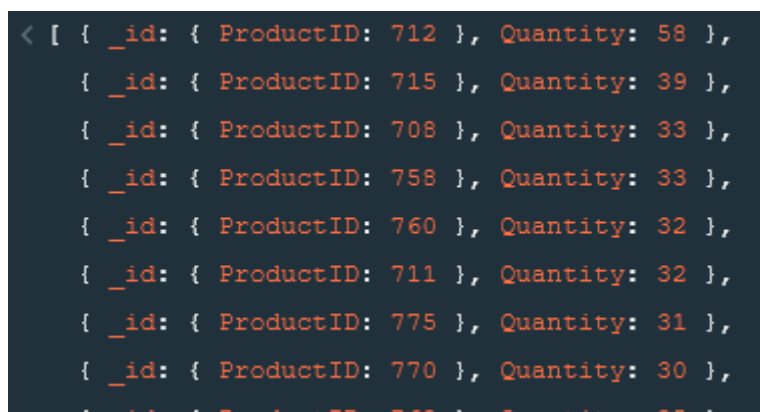
Obter uma listagem do número total de unidade vendidas por produto para um determinado período.

```
db.Receipt.aggregate(  
  [{ $match: { "Products.SellStartDate": { $gte: ISODate("2011-05-30T23:00:00.000+00:00") }, "Products.SellEndDate": { $lte: ISODate("2012-05-29T00:00:00.000+00:00") } } },  
  { "$unwind": { path: "$Products" }, { $group: { "_id": { "ProductID": "$Products.ID" }, "Quantity": { $sum: "$Products.Quantity" } }, { $sort: { "Quantity": -1 } } } ] )
```

Consulta que recolhe produtos com data entre 2011-05-30 a 2012-05-29 e apresenta o número de vezes que esse produto foi vendido nesse período. O operador “\$gte” indica para recolher informação superior ou igual à inserida em seguida e o operador “\$lte” é referente a recolher informação inferior ou igual à inserida em seguida.

Operador “\$unwind” utilizado para decompor o campo “Products” e em seguida agrupar os produtos “\$group” e ordenar os mesmos pela quantidade “\$sort”.

A imagem que se segue é um exemplo do resultado da consulta executada na linha de comandos.



```
< [ { _id: { ProductID: 712 }, Quantity: 58 },  
    { _id: { ProductID: 715 }, Quantity: 39 },  
    { _id: { ProductID: 708 }, Quantity: 33 },  
    { _id: { ProductID: 758 }, Quantity: 33 },  
    { _id: { ProductID: 760 }, Quantity: 32 },  
    { _id: { ProductID: 711 }, Quantity: 32 },  
    { _id: { ProductID: 775 }, Quantity: 31 },  
    { _id: { ProductID: 770 }, Quantity: 30 },  
    { _id: { ProductID: 762 }, Quantity: 25 } ]
```

Figura 17 - Consulta 1

## Consulta 2

Obter uma listagem do número total de unidade vendidas por cliente para um determinado período.

```
db.Receipt.aggregate([
  {$match: {"OrderDate":{$gte:ISODate("2011-05-30T23:00:00.000+00:00"),$lte:ISODate("2011-06-30T23:00:00.000+00:00")}}},{$unwind":{"path: "$Products"}},
  {$group:{"_id":{"CustomerKey":"$Customer.CustomerKey"},
  "Quantity":{"$sum":"$Products.Quantity"}}, {$sort:{"Quantity": -1}}])
```

Consulta que recolhe produtos com data entre 2011-05-30 a 2011-06-30 e apresenta a quantidade de vezes que esse produto foi vendido nesse período. O operador “\$gte” indica para recolher informação superior ou igual à inserida em seguida e o operador “\$lte” é referente a recolher informação inferior ou igual à inserida em seguida.

Operador “\$unwind” utilizado para decompor o campo “Products” e em seguida agrupar os produtos “\$group”, o “\$sum” efetua a soma das quantidades e a ordenação pelo operador “\$sort”.

A imagem que se segue é um exemplo do resultado da consulta executada na linha de comandos.

```
< [ { _id: { CustomerKey: 29705 }, Quantity: 132 },
    { _id: { CustomerKey: 29614 }, Quantity: 93 },
    { _id: { CustomerKey: 29510 }, Quantity: 83 },
    { _id: { CustomerKey: 29698 }, Quantity: 47 },
    { _id: { CustomerKey: 29515 }, Quantity: 46 },
    { _id: { CustomerKey: 29734 }, Quantity: 38 },
    { id: { CustomerKey: 29794 }, Quantitv: 25 },
```

Figura 18 - Consulta 2

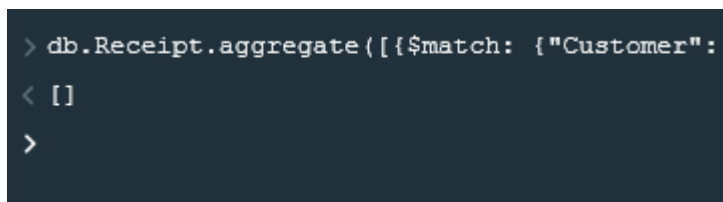
### Consulta 3

Obter uma listagem das vendas em que não existe um cliente válido associado

```
db.Receipt.aggregate([{$match: {"Customer":{"$eq":{}}}},{$project:{"_id":0,
"ReceiptID":1, "OrderDate":1, "Store":1, "StoreName":1}}])
```

Esta consulta verifica se algum cliente se encontra sem informação, se for o caso é considerado inválido.

A imagem que se segue é um exemplo do resultado da consulta, neste caso não devolve nada pois todos os clientes são válidos.



```
> db.Receipt.aggregate([{$match: {"Customer":{"$eq":{}}}},{$project:{"_id":0,
"ReceiptID":1, "OrderDate":1, "Store":1, "StoreName":1}}])
< []
>
```

*Figura 19 - Consulta3*

## Consulta 4

Obter uma listagem dos produtos descontinuados.

```
db.Receipt.aggregate([
  {$match: {"Products.SellEndDate":{"$ne":"NULL"}}},
  {$project:{"_id":0,"Products.ID":1,"Products.Number":1,"Products.Name":1}}])
```

A seguinte consulta verifica se o campo “SellEndDate” é diferente de “NULL”, o operador “\$ne” (not-equal/diferente de) verifica essa condição, caso apresente algum valor o produto é considerado descontinuado.

Esta consulta está a mostrar os produtos descontinuados de cada Receipt, caso seja pretendido ver os produtos descontinuados sem ser por receipt basta alterar a coleção para a “ProductPrices” e correr o mesmo comando.

A imagem que se segue é um exemplo do resultado da consulta.

```
< [ { Products:
  [ { ID: 771,
    Name: 'Mountain-100 Silver. 38',
    Number: 'Mountain-100 Silver. 38' } ] },
  { Products:
    [ { ID: 775,
      Name: 'Mountain-100 Black. 38',
      Number: 'Mountain-100 Black. 38' },
      { ID: 776,
        Name: 'Mountain-100 Black. 42',
        Number: 'Mountain-100 Black. 42' },
      { ID: 777,
        Name: 'Mountain-100 Black. 44',
        Number: 'Mountain-100 Black. 44' },
      { ID: 778,
        Name: 'Mountain-100 Black. 48',
        Number: 'Mountain-100 Black. 48' } ] },
  { Products:
```

Figura 20 - Consulta 4

## Consulta 5

Obter uma listagem dos clientes que compram nenhum produto há mais de um mês.

```
db.Receipt.aggregate([
  {$match: {"OrderDate":{"$lt" : new Date(ISODate()).getTime() - 1000 * 3600 * 24 *
  31}}},{$project:{"_id":0, "Customer.CustomerKey":1}}])
```

A seguinte consulta utiliza o operador “\$lt” para efetuar um comparativo da data, é coletada a data atual e subtraído um mês para efetuar a conta da data a verificar.

A imagem que se segue é um exemplo do resultado da consulta.

```
< [ { Customer: { CustomerKey: 29614 } },
    { Customer: { CustomerKey: 29705 } },
    { Customer: { CustomerKey: 29711 } },
    { Customer: { CustomerKey: 29620 } },
    { Customer: { CustomerKey: 29639 } },
    { Customer: { CustomerKey: 29515 } },
    { Customer: { CustomerKey: 29813 } },
    { Customer: { CustomerKey: 29494 } },
    { Customer: { CustomerKey: 29680 } },
    { Customer: { CustomerKey: 29824 } },
    { Customer: { CustomerKey: 29734 } },
    { Customer: { CustomerKey: 29510 } },
    { Customer: { CustomerKey: 29558 } },
    { Customer: { CustomerKey: 29789 } },
    { Customer: { CustomerKey: 29539 } },
```

Figura 21 - Consulta 5

## Consulta 6

Obter uma listagem dos produtos que não vendidos há mais de uma semana.

```
db.Receipt.aggregate([
  {$match: {"OrderDate":{"$lt" : new Date(ISODate()).getTime() - 1000 * 3600
    * 24 * 7}}}, {"$unwind": {"path": "$Products"}},
  {$project: {"_id": 0, "Products.ID": 1, "Products.Name": 1}}])
```

A seguinte consulta utiliza o operador “\$lt” para efetuar um comparativo da data, é coletada a data atual e subtraído uma semana para efetuar a conta da data a verificar.

A imagem que se segue é um exemplo do resultado da consulta.

```
[ { Products: { ID: 756, Name: 'Road-450 Red. 44' } },
  { Products: { ID: 753, Name: 'Road-150 Red. 56' } },
  { Products: { ID: 760, Name: 'Road-650 Red. 60' } },
  { Products: { ID: 765, Name: 'Road-650 Black. 58' } },
  { Products: { ID: 715, Name: 'Long-Sleeve Logo Jersey. L' } },
  { Products: { ID: 730, Name: 'LL Road Frame - Red. 62' } },
  { Products: { ID: 707, Name: 'Sport-100 Helmet. Red' } },
  { Products: { ID: 711, Name: 'Sport-100 Helmet. Blue' } },
  { Products: { ID: 754, Name: 'Road-450 Red. 58' } },
  { Products: { ID: 712, Name: 'AWC Logo Cap' } },
  { Products: { ID: 729, Name: 'LL Road Frame - Red. 60' } },
  { Products: { ID: 755, Name: 'Road-450 Red. 60' } },
  { Products: { ID: 761, Name: 'Road-650 Red. 62' } },
  { Products: { ID: 770, Name: 'Road-650 Black. 52' } },
  { Products: { ID: 726, Name: 'LL Road Frame - Red. 48' } },
  { Products: { ID: 764, Name: 'Road-650 Red. 52' } },
  { Products: { ID: 766, Name: 'Road-650 Black. 60' } },
  { Products: { ID: 725, Name: 'LL Road Frame - Red. 44' } },
  { Products: { ID: 716, Name: 'Long-Sleeve Logo Jersey. XL' } },
  { Products: { ID: 768, Name: 'Road-650 Black. 44' } } ]
```

Figura 22 - Consulta 6



# Mongo Atlas

Após criar a conta no atlas prosseguimos para a configuração do mesmo.

Efetuando login avançamos diretamente à aba “Network Access” configurar o IP Address. Se quisermos, por questões de segurança, aceder aos dados podemos colocar um IP específico, mas nesta situação não é necessário visto que o seguinte trabalho se destina para fins académicos, então colocamos o respetivo IP da imagem acima para ter total acesso aos dados

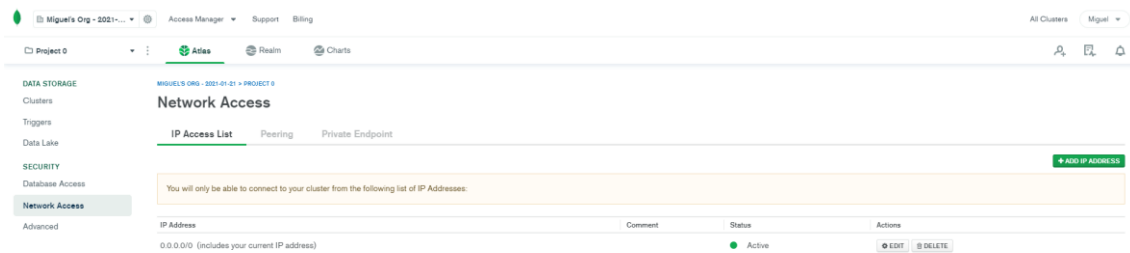


Figura 23 - MongoAtlas

Criação de utilizadores para permitir acesso aos restantes elementos do grupo ao servidor.

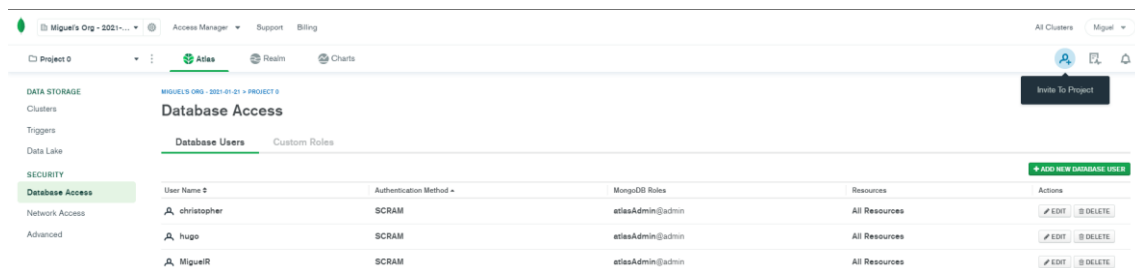


Figura 24 - MongoAtlas - Users

Na figura seguinte podemos observar como obtemos o ip de acesso ao Mongo Atlas utilizando o MongoDBCompass.

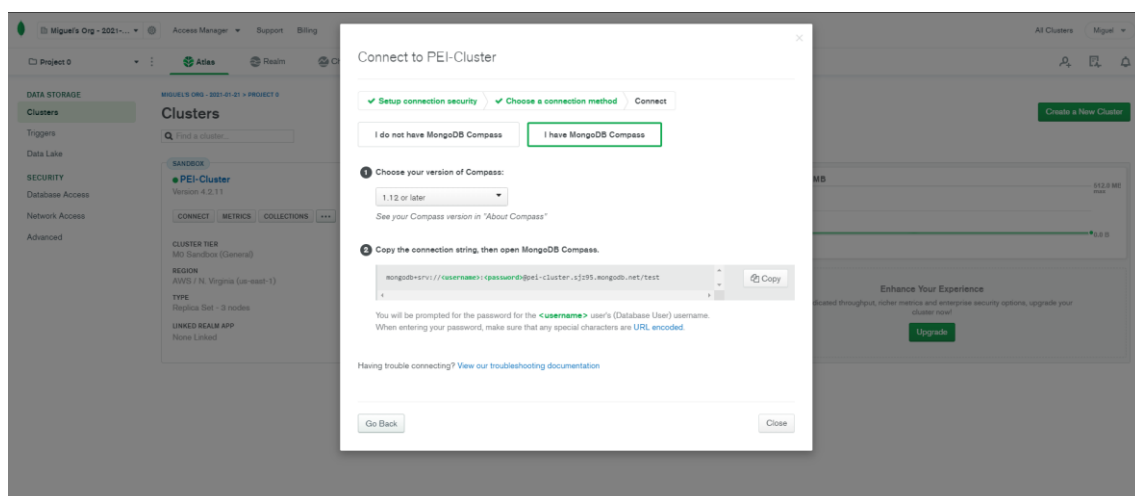


Figura 25 - MongoAtlas - Connect

## Charts

Criação de uma dashboard para facilitar a informação da informação pretendida.

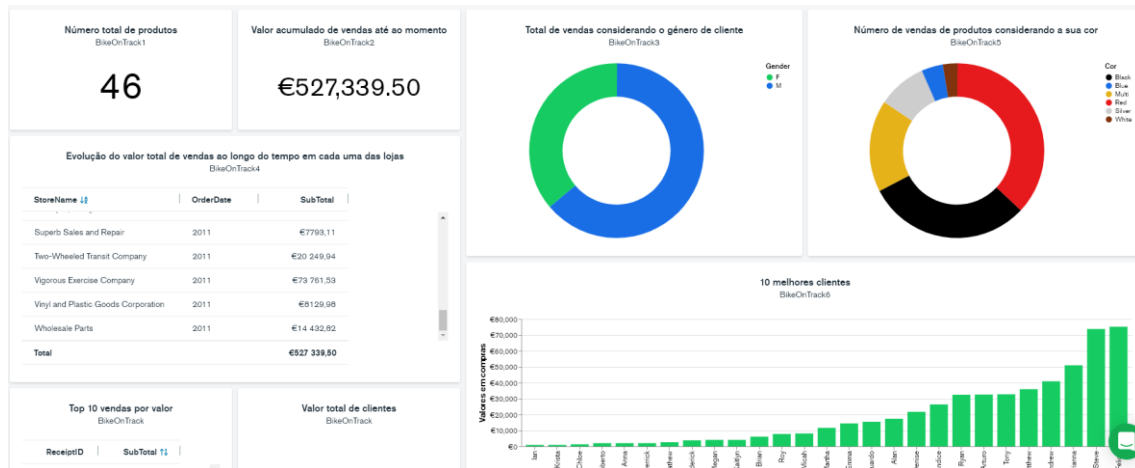


Figura 26 - Mongo Atlas - Charts

Link para consulta dos gráficos:

<https://charts.mongodb.com/charts-project-0-eubco/public/dashboards/600b438b-1423-47a5-8229-7febd8d956e3>

Query:

```
1-[{ "$unwind": {path: "$Products"}}, { "$group": { "_id": { "Products": "$Products.ID" } }}, { "$count": "Products" }]
```

```
2-[{ "$group": { "_id": {}, "Valor de vendas": { "$sum": "$SubTotal" } } }]
```

```
3-[{ "$group": { "_id": { "Customer": "$Customer.Gender", "SubTotal": { "$sum": "$SubTotal" } } }, { "$project": { "_id": 0, "Gender": "$_id.Customer", "SubTotal": 1 } }]
```

```
4-[{ "$group": { "_id": { "StoreName": "$StoreName", "OrderDate": "$OrderDate", "SubTotal": "$SubTotal" } }, { "$project": { "_id": 0, "StoreName": "$_id.StoreName", "OrderDate": "$_id.OrderDate", "SubTotal": "$_id.SubTotal" } }]
```

```
5-[{ "$unwind": { path: "$Products" } }, { "$group": { "_id": { "Color": "$Products.Color", "Quantidade": { "$sum": "$Products.Quantity" } } }, { "$project": { "_id": 0, "Color": "$_id.Color", "Quantidade": 1 } }]
```

```
6-[{ "$group": { "_id": { "FirstName": "$Customer.FirstName", "SubTotal": { "$sum": "$SubTotal" } } }, { "$project": { "_id": 0, "FirstName": "$_id.FirstName", "SubTotal": 1 } }]
```

```
7-[{ "$match": { "OrderDate": { "$gte": ISODate("2011-01-30T23:00:00.000+00:00"), "$lte": ISODate("2021-12-30T23:00:00.000+00:00") } } }, { "$group": { "_id": { "OrderDate": "$OrderDate", "SubTotal": { "$sum": "$SubTotal" } } }, { "$project": { "_id": 0, "OrderDate": "$_id.OrderDate", "SubTotal": 1 } }]
```

--Outras DashBoards----

1.Valor Total de Clientes

```
[{ "$group": { "_id": { "CustomerKey": "$Customer.CustomerKey" } } }, { "$count": "TotalClientes" }]
```

2.Top 10 vendas por valor

```
[{ "$project": { "_id": 0, "SubTotal": "$SubTotal", "ReceiptID": "$ReceiptID" } }, { "$sort": { "SubTotal": -1 } }, { "$limit": 10 }]
```

## Scripts

Criação de um script que após a sua execução cria as coleções extras.

As coleções iniciais já se devem encontrar inseridas no programa para o seu correto funcionamento.

```
//-----  
//Criar a coleção ProductPrices guardada na variável productPrices  
var productPrices = db.ProductDetails.aggregate([  
  { $lookup: { from: 'ProductPH', localField: 'ProductID', foreignField: 'ProductID', as: 'Prices' } },  
  { $project: { "ProductID": 1, "Name": 1, "ProductNumber": 1, "Color": 1, "Prices": "$Prices.Price", "SellStartDate": 1, "SellEndDate": 1 } },  
  { $unwind: { path: "$Prices" } } ]).toArray()  
//Gravar em Ficheiro  
productPrices.forEach(  
  function(doc) {  
    db.ProductPrices.save(doc);  
  }  
)  
//-----  
//Colocar a SellEndDate de string para date nos campos diferentes de null  
db.ProductPrices.find({"SellEndDate":{"$ne":"NULL"}}).forEach(function(doc) {  
  doc.SellEndDate= ISODate(doc.SellEndDate);  
  db.ProductPrices.save(doc);  
})  
//-----  
//Criar a coleção SalesMCP guardada na variável saleswcp. Limitada a 1000 documentos  
var saleswcp = db.SalesDetails.aggregate([  
  { $lookup: { from: 'CustomerDetails', localField: 'Customer', foreignField: 'CustomerKey', as: 'Customer' } },  
  { $lookup: { from: 'ProductPrices', localField: 'ProductID', foreignField: 'ProductID', as: 'Product' } },  
  { $lookup: { from: 'CurrencyDetails', localField: 'CurrencyRateID', foreignField: 'CurrencyRateID', as: 'Currency' } },  
  { $project: { "ReceiptID": 1, "OrderDate": 1, "Customer": 1, "Currency": 1, "SubTotal": 1, "TaxAmt": 1, "Store": 1, "StoreName": 1, "ReceiptLineID": 1, "Quantity": 1, "Product": 1, "UnitPrice": 1, "LineTotal": 1 } },  
  { $limit: 1000 } ]).toArray()  
//Gravar em Ficheiro  
saleswcp.forEach(  
  function(doc) {  
    db.SalesMCP.save(doc);  
  }  
)  
//-----  
//Criar a coleção Receipt guardada na variável receipt  
var receipt = db.SalesMCP.aggregate([  
  { $unwind: { path: "$Customer" } },  
  { $unwind: { path: "$Currency" } },  
  { $unwind: { path: "$Product" } },  
  { $project: { "ReceiptID": "$ReceiptID", "OrderDate": "$OrderDate", "Store": "$Store", "StoreName": "$StoreName", "Customer": { "CustomerKey": "$Customer.CustomerKey", "FirstName": "$Customer.FirstName", "MiddleName": "$Customer.MiddleName", "LastName": "$Customer.LastName", "Gender": "$Customer.Gender", "Email": "$Customer.EmailAddress", "AddressLine1": "$Customer.AddressLine1", "AddressLine2": "$Customer.AddressLine2", "Phone": "$Customer.Phone", "DateFirstPurchase": "$Customer.DateFirstPurchase" }, "Products": { "ID": "$Product.ProductID", "Name": "$Product.Name", "Number": "$Product.Number", "Color": "$Product.Color", "Quantity": "$Quantity", "Price": "$Product.Prices", "UnitPrice": "$UnitPrice", "LineTotal": "$LineTotal", "SellStartDate": "$Product.SellStartDate", "SellEndDate": "$Product.SellEndDate", "Currency": "$Currency", "TaxAmt": "$TaxAmt", "SubTotal": "$SubTotal" }, "TaxAmt": "$TaxAmt", "SubTotal": "$SubTotal" },  
  { $group: { "_id": { "ReceiptID": "$ReceiptID", "OrderDate": "$OrderDate", "Store": "$Store", "StoreName": "$StoreName", "Customer": "$Customer", "Currency": "$Currency", "TaxAmt": "$TaxAmt", "SubTotal": "$SubTotal" }, "Products": { "$push": "$Products" } } },  
  { $project: { "_id": 0, "ReceiptID": "$_id.ReceiptID", "OrderDate": "$_id.OrderDate", "Store": "$_id.Store", "StoreName": "$_id.StoreName", "Customer": "$_id.Customer", "Products": "$Products", "Currency": "$_id.Currency", "TaxAmt": "$_id.TaxAmt", "SubTotal": "$_id.SubTotal" } } ]).toArray()  
//Gravar em Ficheiro
```

Figura 27 - Scripts – Coleções

Script que executa as consultas pretendidas.

```
// 1- Obter uma listagem do número total de unidades vendidas por produto para um determinado período.  
db.Receipt.aggregate([{$match: {"Products.SellStartDate":{"$gte":ISODate("2011-05-30T23:00:00.000+00:00")}, "Products.SellEndDate":{"$lte":ISODate("2012-05-29T00:00:00.000+00:00")}}},  
  { $unwind: { path: "$Products" } },  
  { $group: { "_id": { "ProductID": "$Products.ID" }, "Quantity": { $sum: "$Products.Quantity" } } },  
  { $sort: { "Quantity": -1 } } ])  
  
// 2- Obter uma listagem do número total de unidades vendidas por cliente para um determinado período.  
db.Receipt.aggregate([{$match: {"OrderDate":{"$gte":ISODate("2011-05-30T23:00:00.000+00:00")}, $lte:ISODate("2011-06-30T23:00:00.000+00:00")}}}, {"$unwind": {"path": "$Products"}},  
  { $group: { "_id": { "CustomerKey": "$Customer.CustomerKey" }, "Quantity": { $sum: "$Products.Quantity" } } },  
  { $sort: { "Quantity": -1 } } ])  
  
// 3- Obter uma listagem das vendas em que não existe um cliente válido associado  
db.Receipt.aggregate([{$match: {"Customer":{"$eq":{}}}}, {"$project": {"_id":0, "ReceiptID":1, "OrderDate":1, "Store":1, "StoreName":1}}])  
  
// 4- Obter uma listagem dos produtos descontinuados  
db.Receipt.aggregate([{$match: {"Products.SellEndDate":{"$ne":"NULL"}}}, {"$project": {"_id":0, "Products.ID":1, "Products.Number":1, "Products.Name":1}}])  
  
// 5- Obter uma listagem dos clientes que compram nenhum produto há mais de um mês  
db.Receipt.aggregate([{$match: {"OrderDate":{"$lt": new Date(ISODate().getTime() - 1000 * 3600 * 24 * 31)}}}, {"$project": {"_id":0, "Customer.CustomerKey":1}}])  
  
// 6- Obter uma listagem dos produtos que não vendidos há mais de uma semana  
db.Receipt.aggregate([{$match: {"OrderDate":{"$lt": new Date(ISODate().getTime() - 1000 * 3600 * 24 * 7)}}}, {"$unwind": {"path": "$Products"}},  
  { $project: {"_id":0, "Products.ID":1, "Products.Name":1}}])  
  
//Ex3, outra maneira de fazer  
//db.Receipt.aggregate([{$lookup: { from: 'CustomerDetails', localField: 'Customer.CustomerKey', foreignField: 'CustomerKey', as: 'CustomerKey' } }, {"$match": {"CustomerKey":{"$ne":"Customer.CustomerKey" } } ])
```

Figura 28 - Scripts - Consultas

## Conclusão

Dado por finalizado o trabalho que nos foi proposto, podemos concluir que ficamos a conhecer de forma mais aprofundada toda a informação que nos foi lecionada na aula e disponibilizada na plataforma moodle, aplicando assim os nossos conhecimentos na elaboração de um projeto.

Na nossa opinião esse projeto ajudou-nos a entender um pouco melhor a disciplina, o mesmo aumentou o nosso nível de conhecimento em relação a:

- Conhecer conceitos de armazenamento de documentos e as tecnologias atuais para a estruturação de informação;
- Reconhecer e compreender a semântica e a sintaxe da notação JSON (Javascript Object Notation) bem como mecanismos para a sua preservação e exploração utilizando coleções de documentos em MongoDB;
- Modelação de dados numa perspectiva orientada por documentos.
- Saber explorar dados armazenados numa document-store e apresentar esses dados utilizando um conjunto de visualizações adaptadas ao domínio de negócio.

No decorrer do trabalho foram sentidas diversas dificuldades, inicialmente perceber como íamos estruturar as nossas coleções levou a muito erro o que fez com que tivéssemos que voltar ao início enumeras vezes. Estudando os as operações que podíamos utilizar entramos num ciclo “tentativa-erro” que acabamos por superar e por compreender bem a sua aplicação.

Por último, devemos destacar que, com toda a pesquisa elaborada adquirimos mais competências na área e melhoramos assim a nossa forma de trabalhar como uma equipa num projeto com várias etapas.