Relatório

Processamento Estruturado de Informação

2º Ano – 1º Semestre

Afonso Santos 8150025 - LSIRC

Nuno Josefino 8150189 - LEI

Hugo Silva 8130142 - LEI



# Índice

[Índice 2](#_Toc530936363)

[Introdução 3](#_Toc530936364)

[User details **Erro! Marcador não definido.**](#_Toc530936365)

[Currency details 4](#_Toc530936366)

[Product Details 4](#_Toc530936367)

[Sales Details 4](#_Toc530936368)

[Store Details 5](#_Toc530936369)

[ProductListPriceHistory 6](#_Toc530936370)

[Criação da base de dados e collections 7](#_Toc530936371)

[Importação dos dados csv para as respetivas coleções 7](#_Toc530936371)

[Querys de pesquisa 8](#_Toc530936371), 9, 10

[Conclusão 11](#_Toc530936371)

Introdução

Nesta segunda entrega, podemos observar algumas alterações nos documentos .xsd que por sua vez vão mudar ligeiramente a estrutura dos ficheiros.

Temos também o desenvolvimento inicial das querys de pesquisa que vão ser necessárias para este projeto

UserDetails

Não foram necessárias alterações neste ficheiro para a segunda parte do projeto

CurrencyDetails

Não foram necessárias alterações neste ficheiro para a segunda parte do projeto

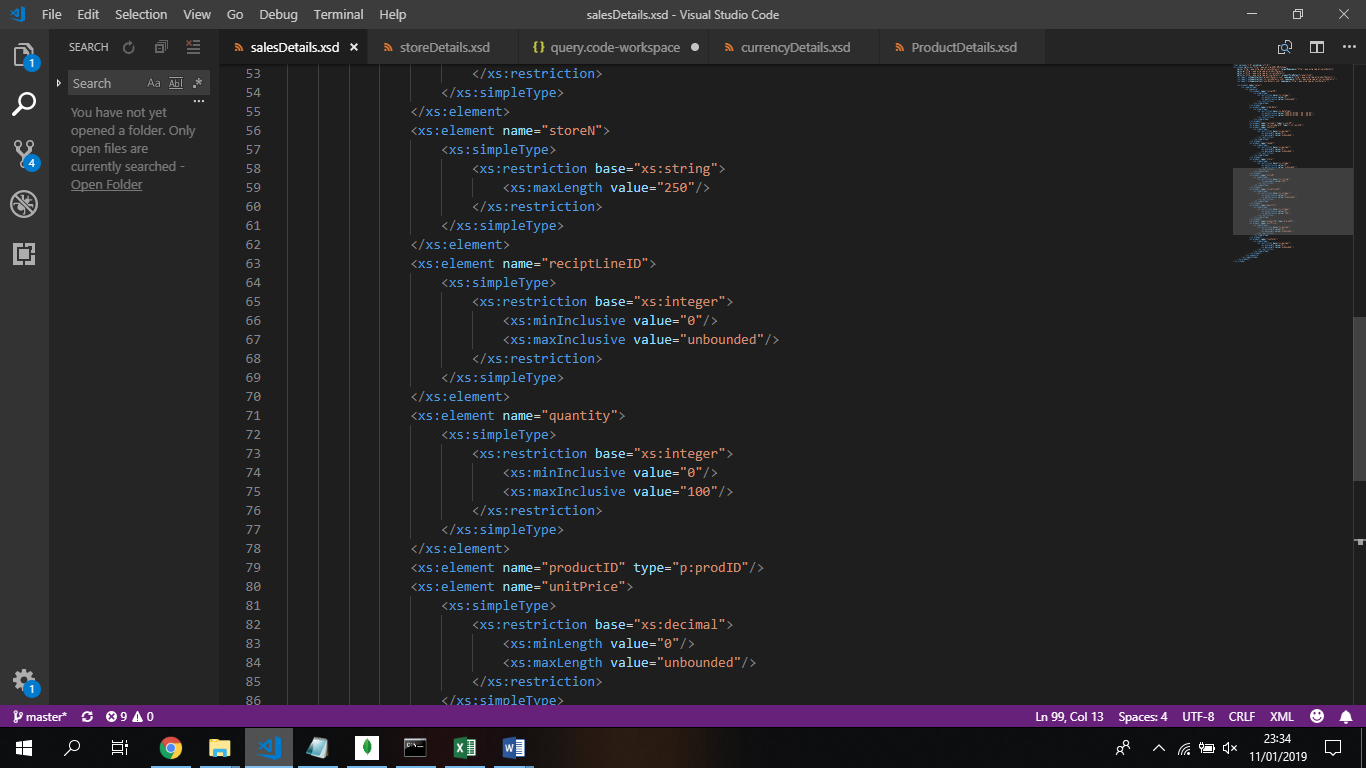
ProductDetails

Não foram necessárias alterações neste ficheiro para a segunda parte do projeto

SalesDetails

Foi alterado o documento salesDetails.xsd de modo a suportar os novos requisitos

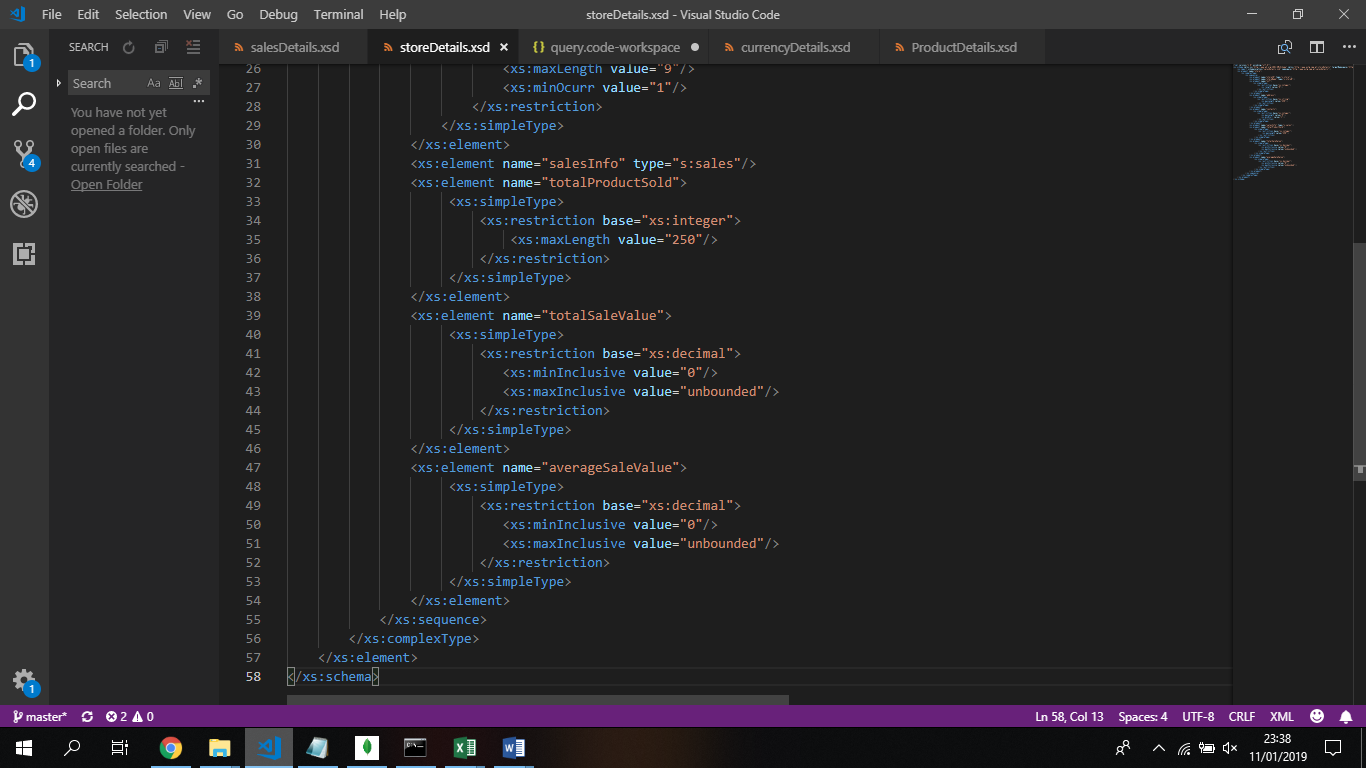
O valor “order” foi substituído pelo valor “quantity” que apresenta as mesmas características e validações



StoreDetails

Foi alterado o documento storeDetails.xsd de modo a suportar os novos requisitos

Foram adicionados os valores “totalProductSold”, “totalSaleValue” e “averageSaleValue” que vão servir para representar valores de vendas das lojas

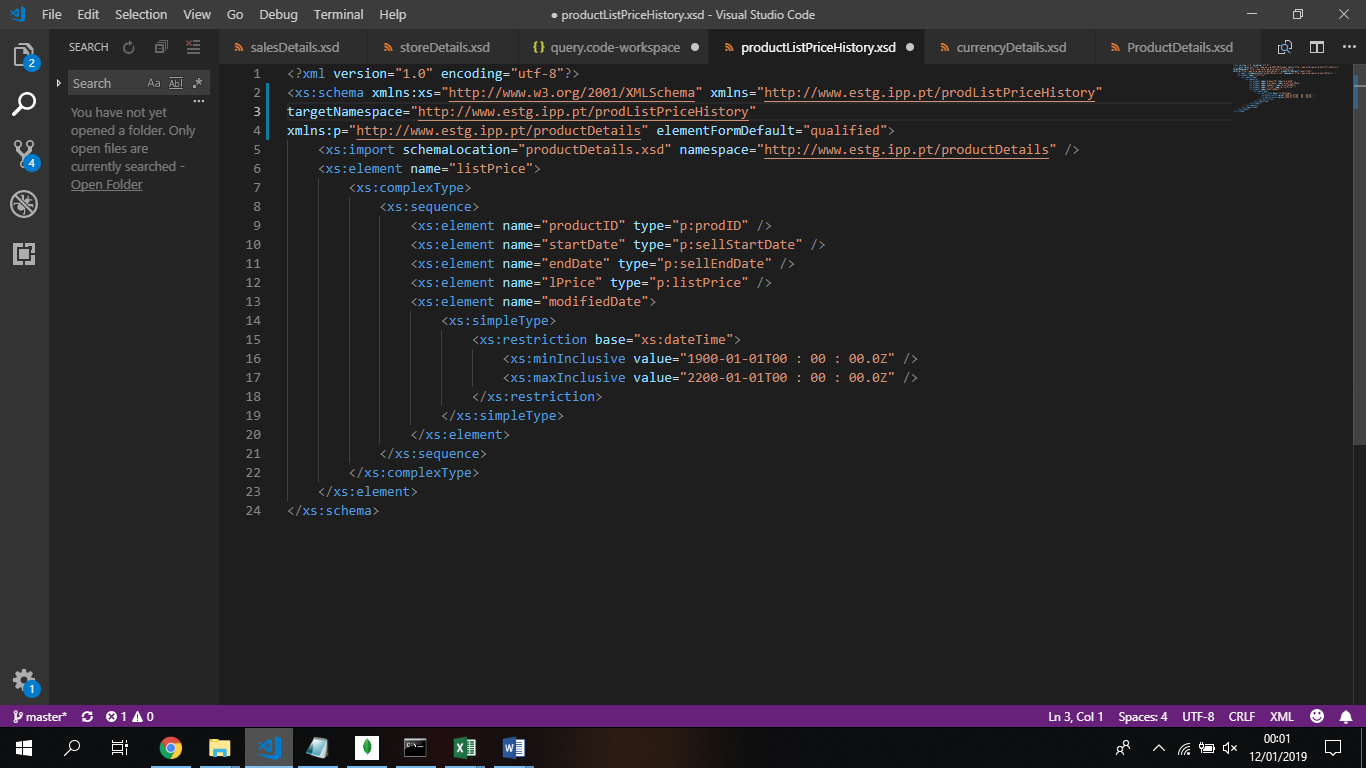


productListPriceHistory

Foi adicionado o ficheiro productListPriceHistory.xsd que trata e valida a informação do documento ProductListPriceHistory.csv

Uma vez que este ficheiro utiliza informações utilizadas e definidas em ficheiros anteriores, recorremos á ultização de imports para os reutilizar e a utilização de namespaces para que não haja conflito entre os elementos.

Utilizamos o “:p” para caracterizar elementos do schema productDetails.



CRIAÇÃO DA BASE DE DADOS E COLLECTIONS

use db BikeOnTrack \\cria a base de dados BikeOnTrack no MongoDB

db.createCollection("CurrencyDetails") \\cria a colleção CurrencyDetails(mongoDB shell)

db.createCollection("ProductDetails") \\cria a colleção ProductDetails(mongoDB shell)

db.createCollection("ProductListPriceHistory") \\cria a colleção ProductListPriceHistory(mongoDB shell)

db.createCollection("SalesDetails") \\cria a colleção SalesDetails(mongoDB shell)

IMPORTAÇÃO DOS DADOS CSV PARA AS RESPETIVAS COLEÇÕES

mongoimport -d BikeOnTrack -c CurrencyDetails ---type csv --file ~/Desktop/moodle/CurrencyDetails.csv --headerline

\\importa os dados csv do ficheiro CurrencyDetails para a coleção CurrencyDetails da base de dados BikeOnTrack(cmd shell)

mongoimport -d BikeOnTrack -c ProductDetails ---type csv --file ~/Desktop/moodle/ProductDetails.csv --headerline

\\importa os dados csv do ficheiro ProductDetails para a coleção ProductDetails da base de dados BikeOnTrack(cmd shell)

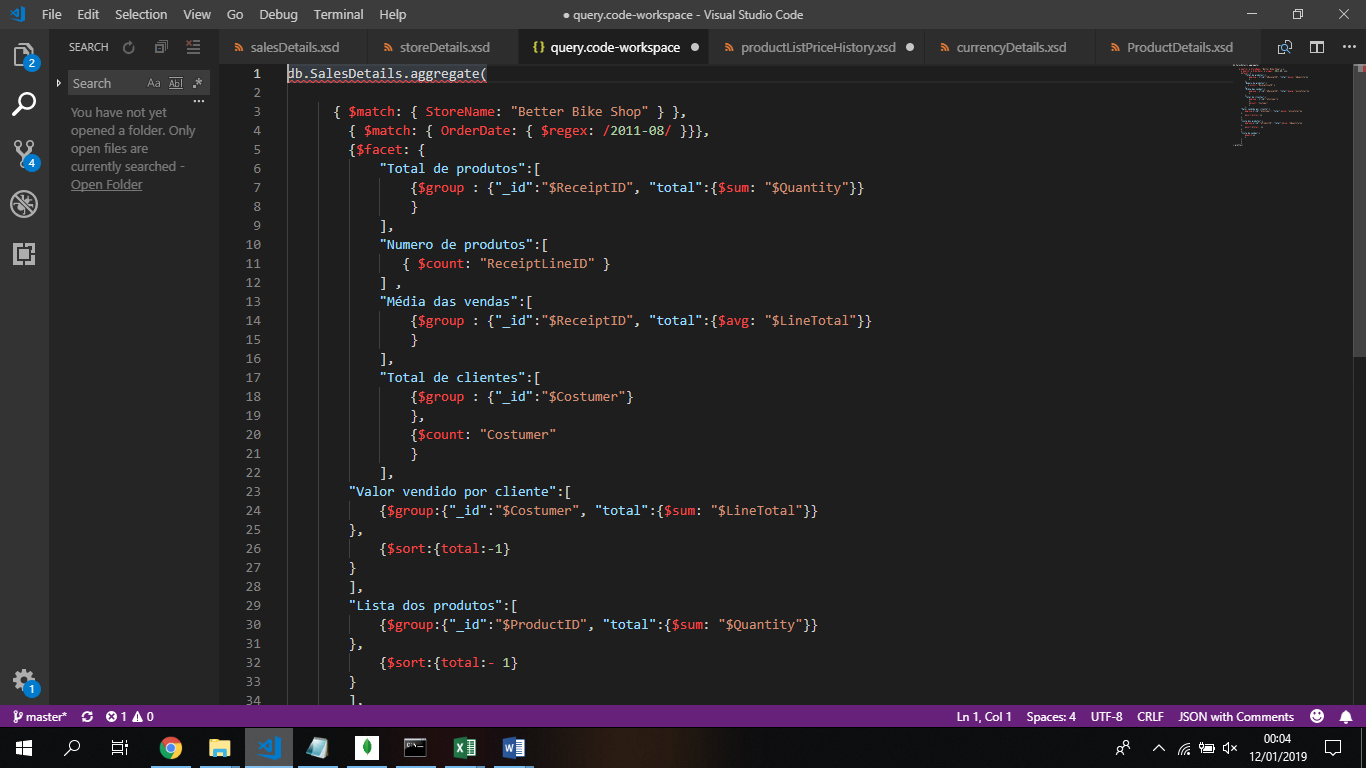
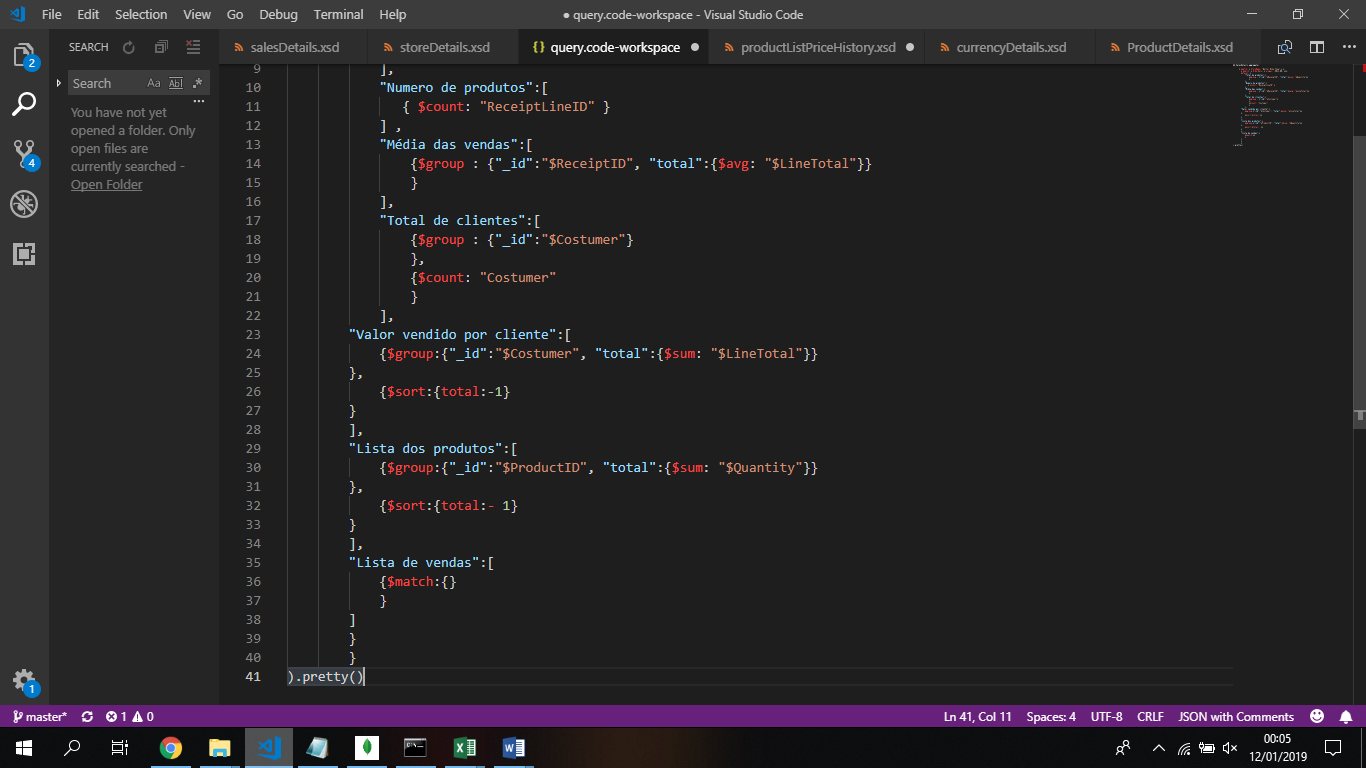
mongoimport -d BikeOnTrack -c ProductListPriceHistory --type csv --file ~/Desktop/moodle/ProductListPriceHistory.csv --headerline

\\importa os dados csv do ficheiro ProductListPriceHistory para a coleção ProductListPriceHistory da base de dados BikeOnTrack(cmd shell)

mongoimport -d BikeOnTrack -c SalesDetails ---type csv --file ~/Desktop/moodle/SalesDetails.csv --headerline

\\importa os dados csv do ficheiro SalesDetails para a coleção SalesDetails da base de dados BikeOnTrack(cmd shell)

Querys de pesquisa



db.SalesDetails.aggregate([{$match:{ StoreName: "Better Bike Shop"}},

{$match:{OrderDate:{$regex:/2011-08/}}}]).pretty()

Mostra os dados do SalesDetails para uma loja("Better Bike Shop" neste exemplo) num certo mês (2011-08 neste exemplo)

db.SalesDetails.aggregate([{$match:{StoreName:"Better Bike Shop"}},

{$match:{OrderDate:{$regex:/2011-08/}}},

{$group:{"\_id":"$ReceiptID", "total":{$sum: "$Quantity"}}}])

Para uma loja especifica("Better Bike Shop" neste exemplo), num mes especifico(2011-08 neste exemplo) mostra a quantidade total de produtos vendidos

{$facet: {"Total de produtos":[{$group:{"\_id":"$ReceiptID", "total":{$sum: "$Quantity"}}}],

Mostra o total de produtos vendidos

"Numero de produtos":[{$count: "ReceiptLineID" }],

Mostra o numero de produtos diferentes vendidos

"Média das vendas":[{$group:{"\_id":"$ReceiptID", "total":{$avg: "$LineTotal"}}}],

Mostra a média das vendas

"Lista de vendas":[{$match:{}}]}}).pretty()

Mostra todas as vendas feitas

"Total de clientes":[{$group : {"\_id":"$Costumer"}},{$count: "Costumer"}],

Mostra o total de clientes a quem a loja vendeu produtos

"Valor vendido por cliente":[{$group:{"\_id":"$Costumer", "total":{$sum: "$LineTotal"}}},

{$sort:{total:-1}}],

Mostra o valor vendido a cada cliente por ordem decrescente

"Lista dos produtos":[{$group:{"\_id":"$ProductID", "total":{$sum: "$Quantity"}}},

{$sort:{total:- 1}}],

Mostra a lista com a quantidade de produtos vendidos por ordem decrescente

"Lista de vendas":[{$match:{} }]

Mostra a lista com todas as informações das vendas para aquela loja naquele mês especifico

Para obter os resultados em relação a todas as lojas, apenas temos que remover o match que especifica a loja.

Conclusão

Para a próxima entrega, a entrega final, vai ser elaborado um método que vai converter os resultados dos querys para ficheiros XML tendo por base as regras definidas nos ficheiros .xsd