2023-2024



Projeto de processamento estruturado de infomração

Trabalho Prático – época normal

**Trabalho elaborado por:**

**Grupo 19**

8220169 – César Ricardo Barbosa Castelo

8220337 – Hugo Ricardo Almeida Guimarães

8220307 – Pedro Marcelo Santos Pinho

Índice

[Índice de Figuras 2](#_Toc154605882)

[Chave de Siglas 3](#_Toc154605883)

[Introdução 4](#_Toc154605884)

[1. Mongo DB 5](#_Toc154605885)

[a. Estrutura da base de dados 5](#_Toc154605886)

[a. Clientes 5](#_Toc154605887)

[b. Produtos 5](#_Toc154605888)

[c. Vendas 5](#_Toc154605889)

[d. Devoluções 5](#_Toc154605890)

[b. Importação dos dados fornecidos 6](#_Toc154605891)

[c. Integração do BaseX com o MongoDB 7](#_Toc154605892)

[2. Organização do XML 8](#_Toc154605893)

[a. Estrutura do XML 8](#_Toc154605894)

[b. Regras do XML (XML Schema) 8](#_Toc154605895)

[Conclusão 9](#_Toc154605896)

# Índice de Figuras

# Chave de Siglas

|  |  |
| --- | --- |
| XML | Extensible Markup Language |
| XSD | XML Schema Definition |
| CSV | Comma Separated Values |
| JSON | Java Script Object Notation |

# Introdução

Este trabalho foi realizado para o âmbito da disciplina de *Processamento Estruturado de Informação*, que funcionará como integrador dos conhecimentos adquiridos no decorrer das aulas.

Este trabalho consiste na construção de um vocabulário XML para suportar o envio de relatórios de vendas e devoluções de cada parceiro da empresa *Phone for You*, uma empresa que comercializa smartphones através de várias lojas. Assim cada parceiro terá de implementar nos seus sistemas informáticos, um módulo que suporta a geração de documentos XML de vendas e devoluções de acordo com vocabulário estabelecido.

Para além da construção de um vocabulário XML para suportar o envio de relatórios de vendas e devoluções, também será preciso fazer uma transformação dos dados já existentes de vários ficheiros CSV para uma base de dados orientada a documentos (MongoDB), para além disso, também será necessário fazer uma API para aceder aos dados, e construir os relatórios para depois retornar ao utilizador.

# Mongo DB

## Estrutura da base de dados

Para a realização deste trabalho, foi-nos enviado vários ficheiros, em formato CSV, que representam um subconjunto de informação tipicamente armazenada por um dos parceiros da *Phone for You*, com o objetivo de fazer a importação deles para o Mongo DB. No entanto, como a informação de ficheiros estava muito fragmentada, uma importação direta era inviável, já que as consultas iriam demorar imenso tempo. Então para resolver esse problema, fez-se uma transformação dos dados quando estes foram importados. Ficando os dados divididos segundo as seguintes coleções:

### Clientes



Figura - Estrutura da coleção cliente no mongoDB

Para poder importar informação dos clientes, foi preciso juntar vários ficheiros em uma única coleção, sendo esses ficheiros: “*address.csv”*, “city.csv” e “country.csv”.

Como um cliente tem uma morada registada, e cada morada tem de ter registado uma cidade, e uma cidade está registada a um país, decidiu-se juntar a informação dessas tabelas em uma única coleção.

### Produtos

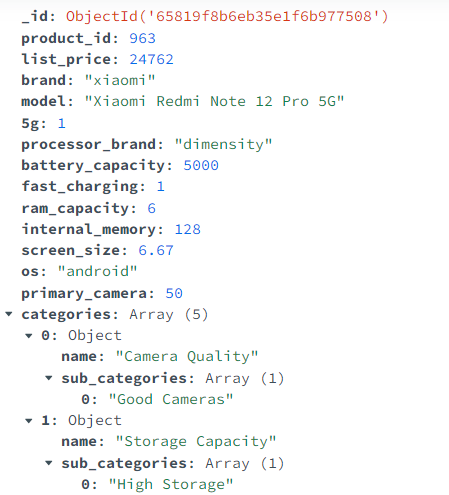


Figura - Estrutura da coleção produtos no mongoDB

Para os produtos aconteceu algo semelhante à coleção dos clientes, no entanto em vez de se juntar a morada, juntou-se o produto com as categorias, visto que um produto tem uma ou mias categorias, e uma categoria pode ter uma ou mais subcategorias. Juntando assim a informação dos ficheiros *“Product.csv”*, *“sub\_category\_product.csv”*, *“sub\_category.csv”* e *“category.csv”*.

### Vendas

[imagem do json]

Para poder criar a coleção das vendas, foi preciso unir os ficheiros: *“sales\_header.csv”*, *“sales\_lines.csv”*, pois, uma “sales\_header” pode ter uma ou mais “sales\_lines”. Para além de unir esses dois ficheiros, também se colocou alguma informação do cliente que comprou o produto, e dos produtos comprados, assim evita-se ter de percorrer várias coleções para quando se for criar relatórios, fazendo com a aplicação tenha um desempenho muito melhor.

### Devoluções

[imagem do json]

Nas devoluções, juntou-se um pouco de toda a informação das outras coleções criadas anteriormente, pois, uma devolução é efetuada por um cliente, sobre um produto em específico, e esse produto está associado a uma venda a partir do código da sua fatura (invoice\_id).

## Importação dos dados fornecidos

Para importar os dados fornecidos para a base de dados, primeiramente, foi preciso criar um script que criasse uma uma base de dados vazia no MongoDB, e depois criasse várias coleções, que irão conter os dados dos ficheiros csv, esse ficheiro chama-se *“setting up the database.txt”*, e ele está na pasta *“.\MongoDB”*

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura - script inicial para criação da base de dados e coleções

Nota: o ficheiro apenas cria coleções vazias, para fazer a importação dos dados dos ficheiros, é preciso fazê-la manualmente a partir do MongoDB Atlas.

Após ter todas as coleções com os dados originais importados, vai ser preciso executar o script de outro ficheiro chamado *“Setting indexes after csv inport.txt”,* ele também esá localizado na pasta *“.\MongoDB”, e tal como o nome indica, ele vai defenir os índices necessários para as migrações que se vai fazer, em alguns casos, as migrações nem são possíveis de realizar sem estes ínidces*



Figura - script para a criiação de índices

Com os dados importados e os índices criados, vai ser preciso fazer uma migração desses dados para novas coleções, que conterão os dados organizados de uma forma que o MongoDB consiga ler de forma eficiente. Para isso, dentro da pasta*”.\MongoDB\pipelines”* contém 4 scripts que contêm todas contêm todas as pipelines de agregação[[1]](#footnote-1) necessárias para a migração dos dados. Elas estão todas numeradas de 1 a 4 para dizer que elas devem ser executadas por ordem crescente, dessa forma, permite que os últimos ficheiros, que contêm mais informação e necessitam de mais alterações, não precisassem de fazer tantas transformações, bastava aceder às coleções que já têm os dados prontos para poder fazer a transferência, poupando assim bastante tempo e esforço.

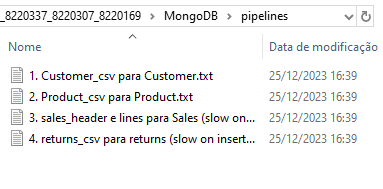


Figura - pipelines para a transformação dos dados

## Integração do BaseX com o MongoDB

Para a integração do baseX com o mongoDB, foram desenvolvidas duas consultas na data Api do mongoDB, para recer relatórios de vendas e devoluções para um determinado mês. Para poder acededr a essas consultas, foi desenvolvida uma API no baseX que comunica com a data API do mongoDB, que devolve os dados e transforma-os para o formato xml. É graças a esta API que é possível especificar o mês e o ano dos dados que queremos receber, pois é nela que eles são definidos como parâmetros.

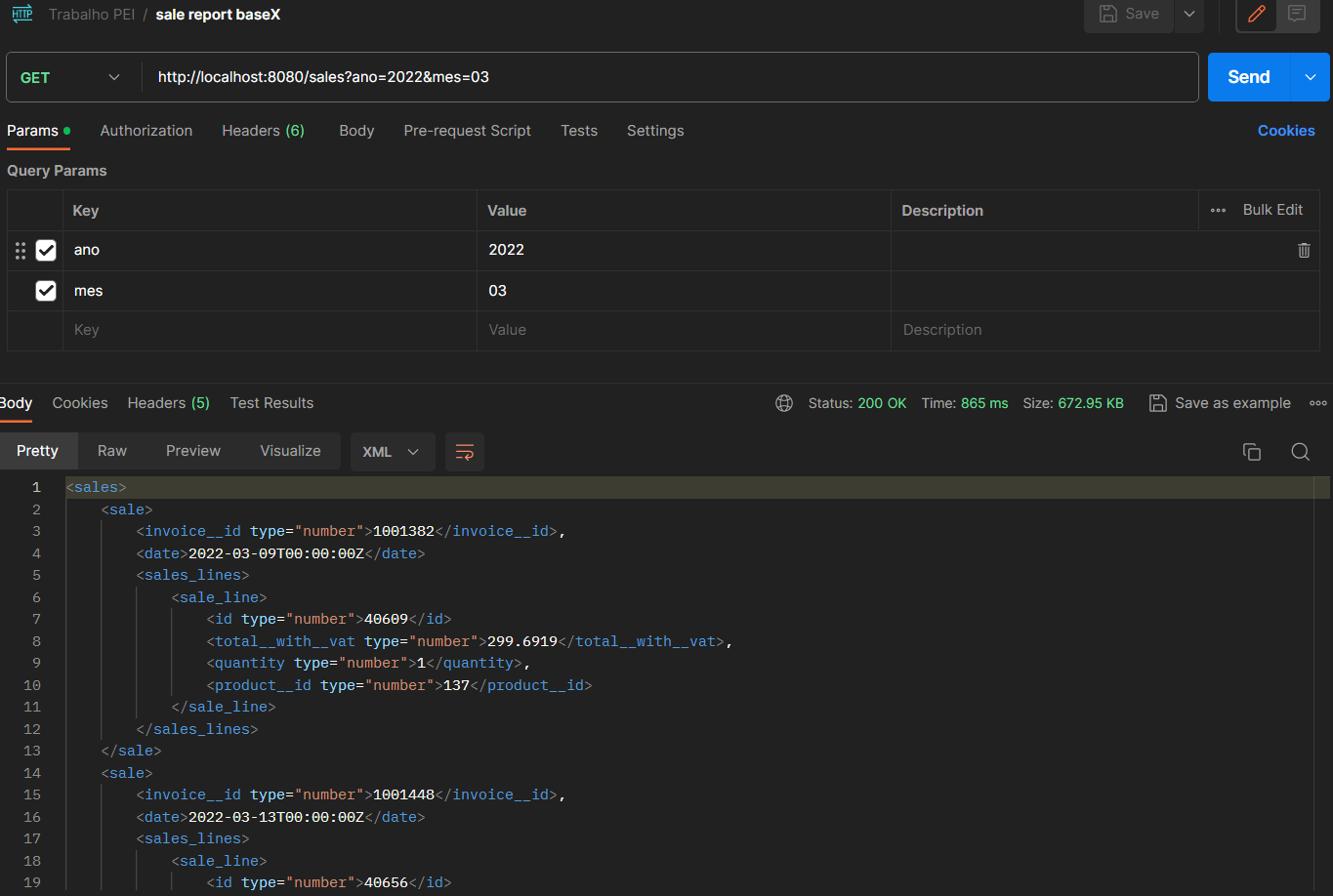


Figura - Relatório de vendas devolvido pelo baseX

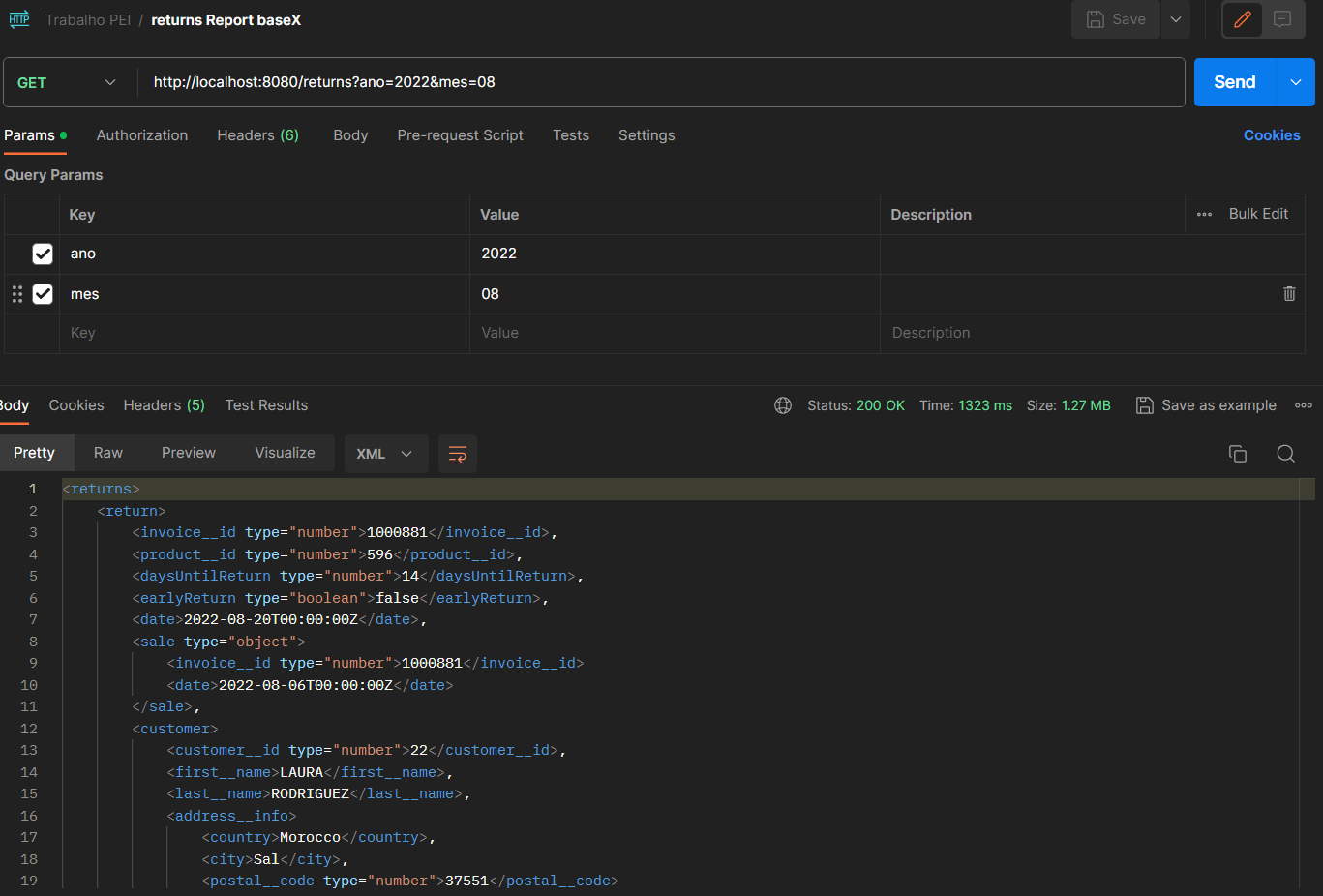


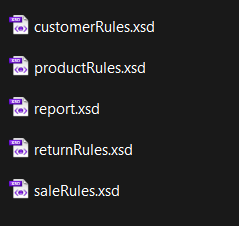
Figura - Relatório de devoluções devolvido pelo baseX

# Organização do XML

## Estrutura do XML

## Regras do XML (XML Schema)

Para as regras do XMl, tentou-se ao máximo com que elas fossem o mais reutilizáveis possíveis, então criou-se vários ficheiros, onde cada um deles define regras específicas para um componente, como pode ser vito na figura a seguir:



Estas regras encontram-se na pasta *“.\BaseX\xsd”*.

Por estarem em ficheiros diferentes, possoiita com que seja possível monstar um relatório costumizado, como por exemplo: um relatório de vendas; devoluções; vendas e devoluções; etc..

# Conclusão

A realização deste trabalho foi bem concebida tendo em conta o que nos foi proposto e também as nossas próprias exigências para o trabalho, entre elas, a constante busca pela estrutura da base de dados, que conseguisse gerar relatórios da forma mais eficiente possível.

Ao longo da realização do projeto deparamo-nos com vários desafios, como por exemplo, a parte da criação da migração dos dados dos ficheiros CSV para o mongoDB, e a passagem dos relatórios de JSON para XML a partir do baseX, que de acordo com todos os elementos do grupo foi o que deu mais trabalho. Mas com isto concluímos que este projeto foi muito importante para o nosso desenvolvimento, pois conseguimos colocar em prática aquilo que foi lecionado durante as aulas, assim, sedimentando os nossos conhecimentos.

1. Um pipeline de agregação consiste em um ou mais estágios que processam documentos, onde cada estágio executa uma operação nos documentos de entrada [↑](#footnote-ref-1)