Processamento Estruturado de Informação

Trabalho Prático

Grupo 4 – 2º Entrega

José Baltar, 8170212

Rodrigo Coelho, 8170282

Rui Silva, 8170283



**Índice**

[1. Introdução 3](#_Toc530841496)

[1.1. Apresentação do Problema 3](#_Toc530841497)

[1.2. Contextualização 3](#_Toc530841498)

[2. Desenvolvimento do Projeto 4](#_Toc530841499)

[2.1. Estruturação e debate 4](#_Toc530841500)

[2.2. Requisitos 4](#_Toc530841501)

[2.3. Execução e explicação 4](#_Toc530841502)

[2.4. Propriedades dos *Schemas* 6](#_Toc530841503)

[2.5. Detalhes Finais 8](#_Toc530841504)

[3. Conclusão 8](#_Toc530841505)

# Introdução

## Apresentação do Problema

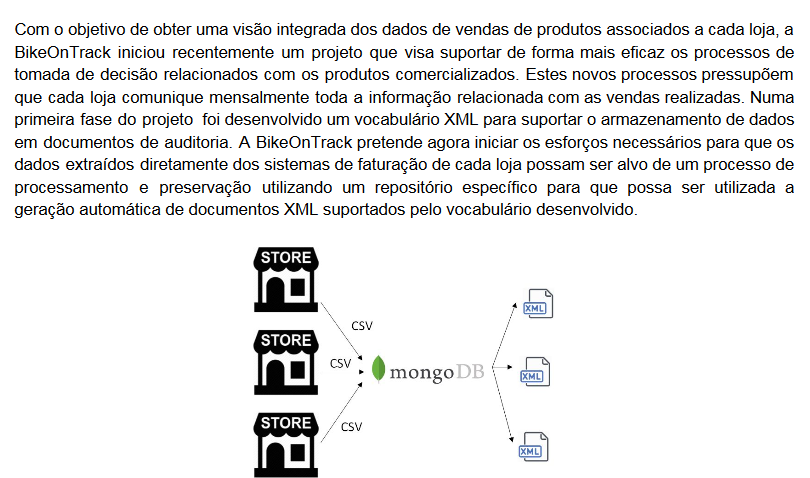
Para a Unidade Curricular de **Processamento Estruturado de Informação**, foi proposto, para Avaliação Contínua, a realização de um Trabalho Prático, divido em várias entregas. Para esta entrega, à semelhança da anterior, foi-nos colocado um enunciado com o problema para resolução, de forma a aplicar as diversas técnicas lecionadas.

Figura 1 Enunciado básico do problema

## Contextualização

Para desenvolver este projeto, foi necessário utilizar conhecimento adquirido nas aulas da U.C., para que se consiga adaptar o vocabulário aos novos problemas que são levantados, e tendo por base as resoluções e requisitos já anteriormente levantadas, e continuando com o cuidado de planear sempre para entregas futuras, de forma a ser possível um melhoramento contínuo e incremental do Trabalho.

Assim sendo, o projeto foi já pensado e planeado para permitir a melhor transformação possível dos documentos na base de dados em *MongoDB* para o vocabulário XML por nós criado na entrega anterior.

Este relatório servirá, por tantom como apoio à interpretação do projeto, de forma a elucidar em relação à estratégia e raciocínio seguidos, segundo a interpretação do enunciado apresentado, dúvidas e problemas encontrados.

# Desenvolvimento do Projeto

## Estruturação e debate

Inicialmente, tal como na anterior entrega, foi necessária a realização de um debate entre os membros do grupo face à abordagem a realizar, pois as interpretações do enunciado variavam de membro para membro. No entanto, sendo este enunciado mais explícito em relação ao anterior, foi mais fácil chegar a um consenso do que seria necessário desenvolver, e como o fazer.

Assim sendo, chegamos a três conclusões sobre o que seria necessário desenvolver:

* Adaptação dos Schemas e do nosso vocabulário aos novos requisitos no enunciado
* Criação de queries para MongoDB que permitissem moldar os dados para uma futura transformação em XML.
* Criação de um serviço REST, em Java, com o objetivo de implementar as queries desenvolvidas neste serviço, para apoiar na transformação para XML numa futura entrega.

## Requisitos

Com base nos requisitos anteriores,

## Execução e explicação

Com o projeto já planeado, foram divididas tarefas entre os membros para facilitar a sua execução, mas, uma vez que a comunicação e entreajuda são extremamente importantes para um projeto destes, os membros do grupo prontamente entraram e colaboraram no desenvolvimento das restantes componentes. Foi também utilizado o Git para ajudar na organização, desenvolvimento e comunicação.

As modificações feitas ao vocabulário procuram refletir as adições que o enunciado requere, e as queries executadas no MongoDB procuram criar os dados necessários para colmatar essas lacunas, que não se encontram nos documentos originais, em formato “csv”. O serviço REST criado procura ao máximo cobrir, com diversas funções, estas lacunas e executá-las, para mais tarde utilizar esses dados para a exportação para os ficheiros XML, através do Transformer.

Foi também necessário proceder à alteração da informação da coleção “ProductDetails”, que continha um problema com os ListPrices, em que algum dos documentos se encontravam com esse valor a 0, sendo necessário ir corrigir esses dados com base na collection “ProductListPriceHistory”, que continha os preços atuais dos produtos e as suas datas de utilização.

Assim sendo, as etapas de forma ordenada da elaboração do projeto foram:

* Interpretação do problema
* Debate sobre a abordagem a seguir e decisão final
* Atribuição e escalonamento de tarefas
* Execução das diversas tarefas atribuídas
* Revisão e correção de erros
* Elaboração do Relatório
* Entrega do projeto

## MongoDB

Seguindo os requisitos levantados do enunciado, foram desenvolvidas várias Queries para manipular os dados do *MongoDB,* como estas pretendiam.

Em primeiro lugar, foi necessário importar as collections (ou seja, os documentos csv) para o Mongo, através do comando “mongoimport”. Após termos as collections em base de dados, foram desenvolvidas queries básicas para perceber e relacionar corretamente os dados. Com isto, foram sendo estruturadas queries complexas, através da função “aggregate”. Adicionalmente, foi necessário recorrer a JavaScript para poder corrigir dados errados nas collections, nomeadamente o *ListPrice* da collection “ProductList”, em que alguns produtos se encontravam a 0, sendo necessário substituir pelo preço que se encontrava na “ProductListPriceHistory”. Nos requisitos, levantava-se o problema de não ser possível mudar o preço para diferente de 0, mas tal foi possível e nenhum dos produtos se encontra com o preço a zero.

As queries construídas refletem os requisitos levantados no enunciado da Entrega. Estas encontram-se no ficheiro “QueriesExecutadas.txt”, que é entregue com o projeto. As queries executadas

## Serviços REST

Para poder implementar os serviços REST, foi necessário recorrer à API Java “SpringRest”, distribuída através do Gradle, para podermos executar, localmente, as queries necessárias.

Como isto, foi necessário implementar diversos métodos, com inspiração na lógica das queries efetuadas em Mongo, de forma a manipular os dados como é pretendido, para a sua posterior exportação para XML.

De forma a facilitar a compreensão do código, este encontra-se devidamente documentado, explicando o que cada método deverá executar, parâmetros e retornos, para que, efetivamente, se perceba o código e o que deverá ser suposto executar.

O nosso Gradle também se encontra presente, de maneira a que, se necessário, se possa perceber as condições de execução que tivemos, de forma a prevenir problemas de compatibilidade.

Executamos no browser, algumas procuras de teste, com sucesso, nomeadamente as seguintes, que encontraram em anexo, no ficheiro “links-do-Rest.txt”.

## Propriedades dos *Schemas (Alterar)*

Para a segunda fase do projeto proposto, foram necessárias alterações a schemas anteriormente desenvolvidos, sendo que os seguintes não foram modificados:

* ***SchemaLoja ("ProjetoPEI/Grupo/Entrega1/Loja");***
* ***SchemaCliente ("ProjetoPEI/Grupo/Entrega1/Cliente");***
* ***SchemaProduto ("ProjetoPEI/Grupo/Entrega1/Produto");***

Sendo que a informação relativa aos *Schemas* acima referidos se encontra exposta na primeira fase de entrega do projeto.

Com isto, os únicos *schemas* anteriormente criados onde foram necessárias alterações foram os seguintes:

* ***SchemaVendas ("ProjetoPEI/Grupo/Entrega1/*** ***ExercicioVendas");***
* ***CommonShema ("ProjetoPEI/Grupo/Entrega1/CommonSchema");***

Em resposta aos dois primeiros campos do enunciado, “Número total de produtos” e “Número total de produtos diferentes”, foram adicionados no “CommonSchema” dois *simpleTypes* distintos sendo que um deles corresponde ao número total de produtos e o segundo corresponde ao número total de produtos diferentes, ambos possuem um *restriction element* seguidos de *minInclusive* e *maxInclusive* restringindo o valor de cada um deles para que não possa ser menor que 0 e que o valor máximo não tenha limite, sendo o “*maxInclusive*” definido como *unbounded*. Estes *simpleTypes* foram inseridos no *“*CommonSchema” pois são usados no “SchemaVendas” e “ExercícioVendas.xsd” - *schema* referente ao exercício (conjunto de vendas). Por sua vez foi adicionado ao “SchemaVendas” um novo elemento “InfProdutos” que contém os dados acima referidos. Também no “SchemaVendas” houve a necessidade de criar um novo elemento referente ao preço médio de venda dos produtos, tendo-se atribuído assim o tipo decimal.

* ***ExercícioVendas(“ProjetoPEI/Grupo4/Entrega1/ExercicioVendas”):*** *Schema* composto por um elemento principal “Documento” que possui todos os dados relacionados com o Exercício (conjunto de vendas). Este *schema* contém um *simpleType* referente ao Númerototal de clientes diferentes, sendo este do tipo inteiro, restringido ao valor mínimo de zero e não limitado quanto ao valor máximo. O elemento principal deste *schema,* “Documento”, é um *complexType* seguido de *sequence* que contém os elementos alusivos ao número de produtos diferentes, número total de produtos e ao número total de clientes diferentes, sendo que cada um destes possui o seu *type* previamente definido.

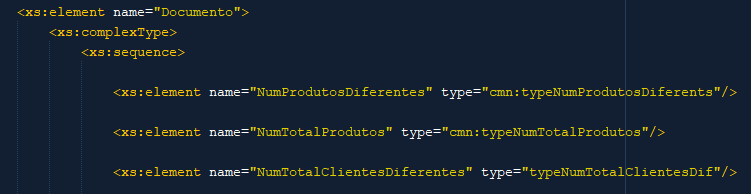


Figura 2 SchemaExercícioVendas

A próxima imagem correspondente ao elemento “ValorVendidoPorCliente”, inicializada como *complexType* e contendo uma *sequence* que poderá acontecer diversas vezes, sendo então atribuído um *minOccurs* e *maxOccurs*, com valor mínimo 0 e máximo *unbounded,* a *sequence* contém o elemento “Cliente” que possui um *simpleContent*, neste caso *extension* com a base decimal pois trata-se de um valor (preço), este elemento possui um atributo “id” do tipo inteiro que se refere ao id do cliente em questão. O mesmo acontece com os dois seguintes elementos, “ProdutosIguaisVendidos” - em que o seu tipo é *unsignedInt* por se tratar de um valor inteiro positivo e “ValorTotalVenda” - neste está presente o atributo “codigoMoeda” que ditará para qual moeda corresponde o valor total da venda. A sintaxe escolhida foi a demonstrada pois o valor é pedido por cada cliente/produto/moeda, ou seja, o elemento “ProdutosIguaisVendidos” varia conforme o produto em questão, o mesmo acontece com o cliente e a moeda.

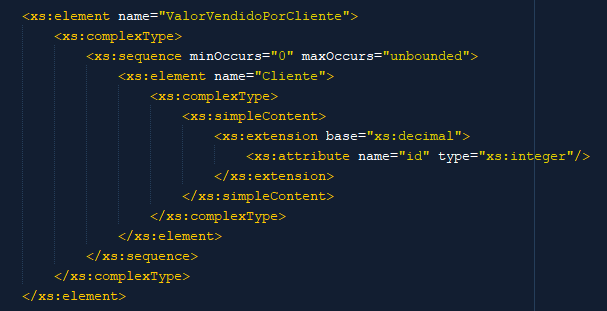


Figura 3 Elemento "ValorVendidoPorCliente" -SchemaExercícioVendas

* ***SchemaInformacaoGeral(“ProjetoPEI/Grupo4/Entrega1/InformacaoGeral”):*** Schema composto por um *complexType “typeInformacaoAdicional” que contém uma sequência de elementos semelhantes ao “ValorVendidoCliente”- Figura 3*, desta vez correspondente ao total de produtos vendidos por loja, valor total das vendas por loja e ao valor médio do preço de venda dos produtos, também por loja, sendo necessário para todos eles ter em consideração o id da loja em questão.



* ***SchemaAuditoriaLoja ("ProjetoPEI/Grupo4/Entrega1/Auditoria"):*** Exclusivamente para a segunda parte do projeto, desenvolveu-se o *schema* acima designado, neste é contido um *complexType* do tipo“typeAuditoria”, em uso no “SchemaExercícioVendas”, este contém um elemento “loja” com o tipo específico da loja, que possui elementos como “Nome”, “Morada”, “Contactos” e “NIF”. Dentro do mesmo *sequence* desenvolveu-se um outro elemento complexo que contém os elementos “Moeda” e “DadosVenda”, que possuem como atributo a data de início e a data de fim da venda. O elemento “typeAuditoria” inclui também ele um atributo designado por “dataCriacao” que corresponde à data de criação do documento.

## Detalhes Finais

Diversas modificações ao projeto encontram num repositório *Git* que poderá ser encontrado em: “<https://github.com/JoseBaltar/ProjetoPEI>”. Os *commits* realizados ao repositório procuram representar aquilo que foi sendo feito por parte dos membros do grupo, mas não são réplicas fiéis do trabalho realizado por cada um. O esforço, sobretudo na procura de erros foi geral e discutida entre os membros do grupo, embora normalmente tenha sido apenas levado para o *git* por um membro. As *queries*, devido à sua natureza, foram sendo construídas separadamente do *git*, mas sempre em contacto constante entre os diversos membros do grupo.

A execução deste relatório, igualmente, recaiu na ponderação de todos os membros do grupo.

# Conclusão

Neste projeto abordamos o processo de adaptação e reutilização de um vocabulário XML, através da modificação de documentos XSD e *namespaces* associados, criação e estruturação de queries na plataforma *MongoDB*, e a utilização da API Java *Spring Rest*, de forma a planear a implementação das pesquisas para a transformação dos documentos em formato BSON, guardados na base de dados do *MongoDB,* para ficheiros XML, seguindo um conjunto de regras e requisitos.

Adicionalmente, uma importante componente deste projeto foi a nossa capacidade, enquanto estudantes e membros de um grupo de trabalho, na interpretação e discussão relativas a um problema pré-estabelecido, decidindo a abordagem a seguir.

Acreditamos que este projeto foi extremamente positivo para nós, pois permitiu-nos reforçar as nossas técnicas e conhecimentos relativos aos conteúdos usados, incluindo na criação deste relatório. Não apenas isto, mas também foi essencial para a formação de um maior espírito crítico em relação à análise e abordagens de problemas práticos.