Processamento Estruturado de Informação

Trabalho Prático

Grupo 4 – 2º Entrega

José Baltar, 8170212

Rodrigo Coelho, 8170282

Rui Silva, 8170283



**Índice**

[1. Introdução 3](#_Toc535006914)

[1.1. Apresentação do Problema 3](#_Toc535006915)

[1.2. Contextualização 3](#_Toc535006916)

[2. Desenvolvimento do Projeto 4](#_Toc535006917)

[2.1. Estruturação e debate 4](#_Toc535006918)

[2.2. Requisitos Pertinentes 4](#_Toc535006919)

[2.3. Execução e explicação 5](#_Toc535006920)

[2.4. *MongoDB* 6](#_Toc535006921)

[2.5. Serviços REST 6](#_Toc535006922)

[2.6. Propriedades dos *Schemas* 7](#_Toc535006923)

[2.7. Detalhes Finais 9](#_Toc535006924)

[3. Conclusão 9](#_Toc535006925)

# Introdução

## Apresentação do Problema

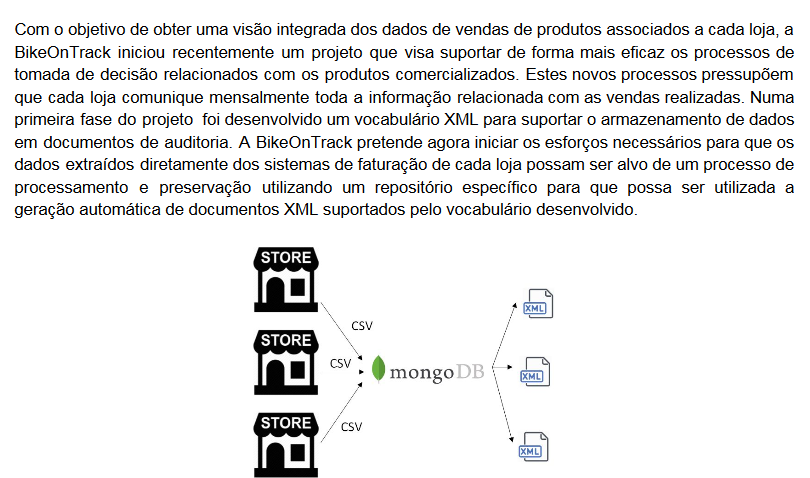
Para a Unidade Curricular de **Processamento Estruturado de Informação**, foi proposto, para Avaliação Contínua, a realização de um Trabalho Prático, divido em várias entregas. Para esta entrega, à semelhança da anterior, foi-nos colocado um enunciado com o problema para resolução, de forma a aplicar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

Uma das maiores diferenças em relação à primeira entrega focou-se na necessidade de alterar a estrutura do *Schema* que iria ser usado para gerar os *XMLs* relativos às pesquisas no *MongoDB*.

Isto é, na entrega anterior a abordagem focava-se na criação de um documento (XML) de auditoria de **uma** loja. Então, o documento gerado teria a informação de uma venda relativa a uma loja, ou vice-versa.

Para esta segunda entrega, tendo em conta os novos tipos a serem adicionados aos *schemas* (nomeadamente, número total de produtos vendidos, valor total das vendas de uma loja em específico, etc.), foi necessário definir um vocabulário que pudesse agregar esses novos tipos sem que a estrutura deixasse de fazer sentido. Dessa forma, o antigo “DocumentoGeral” transformou-se num *schema* de definição de tipos relativo à auditoria de uma loja e foi criado um novo *schema* “geral” que agrega várias auditorias tal como outras informações adicionais, caso haja necessidade.

Um outro requisito, relativo à definição de novos tipos sobre informações a serem processadas pelo MongoDB, consiste na condição entre serem opcionais ou necessários.

No enunciado, no último ponto sobre a adição aos schemas, diz que deve ser possível incluir um novo conjunto de informação no documento XML gerado, sendo que nos pontos anteriores os dados serão somente informações pertinentes relativas a uma só venda ou ao conjunto delas. Consoante esta perspetiva, dissidiu-se que, no primeiro, os tipos seriam necessários (“minInclusive=”1””) e os segundos de introdução opcional (“minInclusive=”2””).

## 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”.

## Alterações nos *Schemas*

Para a segunda fase do projeto, 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 feitas alterações, ou criados de raiz, foram os seguintes (nota: dentro dos próprios *schemas* estão sinalizados, através de um comentário, os novos tipos adicionados):

* ***ExercícioVendas(“ProjetoPEI/Grupo4/Entrega2/ExercicioVendas”):***

Schema criado de raiz (tendo em conta o requisito explicado anteriormente), sendo a partir dele que se criarão os XML.

*É* composto por um elemento principal, “Documento”, que contém os dados relacionados com o Exercício (conjunto de vendas), em forma de lista contendo várias instâncias de auditorias, tal como outros tipos relativos a informações pertinentes, umas obrigatórias e outras opcionais.

Como exemplo, a próxima imagem, referente ao elemento “ValorVendidoPorCliente”, mostra um dos métodos de estrutura para a criação dos elementos relativos aos dados de informações.

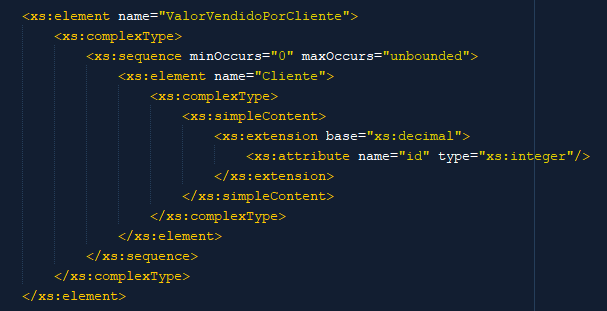


Figura 2 - Elemento "ValorVendidoPorCliente" - SchemaExercícioVendas (Nota: protótipo, o simpleContent foi adicionado ao CommonSchema! Visto na próxima imagem…)

Com o número de ocorrências mínimo 0 e máximo *unbounded,* a *sequence* contém o elemento “Cliente” que possui um *simpleContent*, neste caso uma *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 (utilização do *simpleContent*) acontece com outros dois elementos (não mostrados na imagem), com duas situações distintas.

“TotalProdutosVendidos”, em que a sua base é *unsignedInt* por se tratar de um “número de coisas”.

“ValorTotalVendidoPorMoeda”, onde está presente o atributo “codigo” (não possibilitando o uso do tipo já existente no “CommonSchema”, visto asseguir), que ditará para qual moeda corresponde o valor total da venda.

A abordagem seguida baseou-se na dependência do valor que é pedido, isto é o resultado poderá ser um valor inteiro ou decimal (decidindo a base) e o elemento deverá ser identificado (e dependendo do elemento, o tipo do atributo poderá ser diferente, por exemplo no caso da “Moeda”).

* ***CommonShema ("ProjetoPEI/Grupo/Entrega2/CommonSchema");***

Pequena adição de um *simpleContent* (abordado anteriormente)*,* utilizado pelos novos tipos relativos às informações que serão introduzidas pelo MongoDB, que possibilita, por exemplo, a adição ao elemento “TotalProdutosVendidos”, no *XSD* “ExercicioVendas.xsd”, de outros elementos, “Produtos”, do tipo inteiro juntamente com um atributo relativo ao “id” desse elemento.

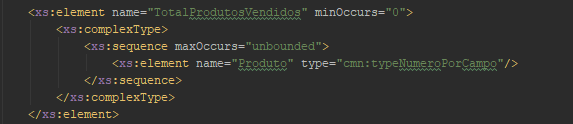


Figura 3 Exemplo no SchemaExercicioVendas

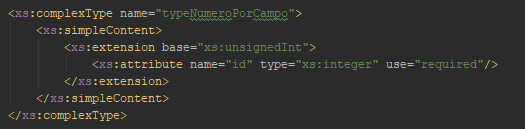


Figura 4 Exemplo no CommonSchema

* ***SchemaInformacaoGeral(“ProjetoPEI/Grupo4/Entrega2/InformacaoAdicional”):***

Define o *complexType “typeInformacaoAdicional”*, que será utilizado no schema abordado anteriormente, que contém uma sequência de elementos relativos aos dados de informação correspondentes à globalidade do exercício (informações sobre o conjunto das várias lojas), sendo utilizado novamente o método já explicado nos pontos anteriores.

Como exemplo é apresentado o caso em específico:

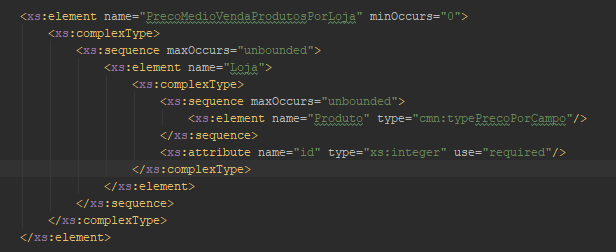


Figura 5 – InformacaoAdicional.xsd

Diferente dos elementos anteriores, o preço médio de venda dos produtos por loja, como o nome indica terá que agregar duas listas (*sequence*), uma para as várias lojas e outras para os vários produtos vendidos por essas lojas.

* ***SchemaAuditoriaLoja ("ProjetoPEI/Grupo4/Entrega2/Auditoria"):***

Este Schema é uma simples adaptação do anterior schema denominado “DocumentoGeral” à nova estrutura, que contém o conjunto dos tipos relacionados a uma auditoria realizada por uma loja. Nomeadamente, a informação da própria loja e os dados sobre a venda. (Detalhes já explicados no relatório anterior).

## 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.