

Gestão eficiente de Vendas e Devoluções para a Phone For You: Relatório de projeto do grupo 7

Licenciatura em Engenharia Informática

Licenciatura em Segurança Informática em Redes de Computadores

2023/2024

Grupo 7

André Eduardo Araújo Faria– [8220787](mailto:%38%32%32%30%3787@e%73tg.%69%70%70.%70%74)

Paulo Sérgio Marques Coelho – 8220195

Tiago Manuel Nunes Coelho - 8220196

Índice

[1. Introdução 3](#_Toc156721183)

[2. Abordagem seguida e apreciação crítica do trabalho 4](#_Toc156721186)

[3. Estruturação de dados no MongoDB e API desenvolvida 5](#_Toc156721187)

* MongoDB
* BaseX

[4. Conclusão 11](#_Toc156721188)

## Introdução

A Phone for You é uma empresa que vende smartphones através de várias lojas. A empresa tem parcerias em todo o país para realizar as vendas, e não só entrega os equipamentos nas lojas, mas também trata do processo de devolução de produtos.

Devido ao crescimento nos últimos anos, a Phone for You quer que cada parceiro apresente mensalmente um relatório de vendas dos smartphones, incluindo as devoluções. Para facilitar esse processo, cada parceiro terá de adicionar um módulo nos seus sistemas informáticos para exportar as informações necessárias sobre vendas e devoluções.

Para ajudar no envio desses relatórios, foi decidido disponibilizar um vocabulário XML. Cada parceiro terá que implementar nos seus sistemas um módulo que consiga gerar documentos XML de vendas e devoluções, seguindo o vocabulário definido. Este passo visa tornar a integração dos dados dos parceiros mais fácil.

Esta abordagem não só fortalece a colaboração entre a Phone for You e os seus parceiros, mas também estabelece um padrão comum para a comunicação de dados. Com isso, espera-se aumentar a eficiência e transparência em toda a cadeia de distribuição, contribuindo para o crescimento sustentável e a excelência operacional da Phone for You.

## Abordagem seguida e apreciação crítica do trabalho desenvolvido

A nossa abordagem consistiu, numa primeira fase, na elaboração dos documentos XSD com os quais verificaríamos os XML que seriam retornados através dos pedidos. Depois, passamos à parte da exportação dos dados e tratamento dos mesmos. Assim, começamos por importar os dados que constavam nos CSV´s para coleções dentro da base de dados PhoneForYouDB. De seguida criamos coleções através do método “aggregate” de forma a garantir que, no futuro, aquando da criação das funções para a obtenção do relatório não fosse necessário usar este tipo de consultas que levariam a uma má performance no processo de requisição dos dados. Depois ativamos a Data API no Mongo Atlas para conseguirmos comunicar com a base de dados através de softwares externos, nomeadamente o BaseX e o Postman. O passo seguinte foi implementar uma API no BaseX que possuía os recursos: obter o relatório de vendas em XML para um determinado mês e obter o relatório de devoluções em XML para um determinado mês. Por fim, restou realizar pedidos HTTP à API do Mongo Atlas através do BaseX.

De forma geral achamos que procedemos quase sempre da melhor maneira e conseguimos atingir os objetivos que nos foram propostos. O único ponto critico do trabalho é aquando do retorno do XML e a verificação do mesmo, tendo em conta que, apesar do documento apresentar a estrutura correta, é necessário colocar os namespaces manualmente para que o nosso XSD verifique o documento.

## Estruturação de dados no MongoDB e API desenvolvida

### MongoDB

Em termos de base de dados do MongoDB, começamos por criar a base de dados PhoneForYouDb que serviu de suporte a todas as etapas do trabalho. Em primeiro lugar, começamos por adicionar as coleções consoante os CSV’s que nos foram disponibilizados, ou seja, temos uma coleção por cada CSV disponibilizado. Para conseguirmos fazer uma distinção entre as coleções por nós criadas e essas coleções importadas, optamos por deixar os nomes das coleções da nossa autoria em português, tendo em conta que nos CSV´s os nomes estavam em inglês.

Decidimos então organizar os dados de uma forma que nos fosse mais conveniente para a realização dos futuros relatórios, para o efeito, foram criadas as coleções: Clientes, Produtos, NovaDevolucoes, Devoluções, Valor\_Total\_Linhas e Vendas.

1. **Coleção Clientes:**

Nesta agregação, a coleção "Clients" é enriquecida com informações de outras coleções. O operador $lookup é utilizado para integrar dados da coleção "Address", "City" e "Country" com base nos seus campos estrangeiros. Em seguida, o $unwind é aplicado para desdobrar os documentos criados pelo $lookup. O $project é utilizado para remodelar a estrutura do documento, incluindo a criação de subcampos aninhados para representar as informações do endereço. O resultado final é armazenado na nova coleção "Clientes".

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

Descrição gerada automaticamente

Figura : Organização dos dados na coleção Clientes

1. **Coleção Devoluções:**

Aqui, foi construída uma pipeline para a coleção "Returns". Em primeiro lugar, é realizado um agrupamento pelo campo "invoice\_id" utilizando o $group, onde os produtos associados a cada fatura são agrupados num array chamado "produtos", e a data da fatura é preservada. Em seguida, o $project é usado para renomear e selecionar os campos desejados. O resultado é armazenado na nova coleção "Devoluções".

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

Descrição gerada automaticamente

Figura 2: Organização dos dados na coleção Devoluções

1. **Coleção Produtos:**

A coleção "Product" é enriquecida com informações das coleções "Sub\_Category\_Product" e "Sub\_Category". Através do $lookup, os produtos são conectados às subcategorias, e o $addFields cria um campo chamado "sub\_category\_name" com base em condições específicas. O $project é utilizado para selecionar e renomear os campos desejados, e o resultado é armazenado na coleção "Produtos".

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

Descrição gerada automaticamente

Figura 3: Organização dos dados na coleção Produtos

1. **Coleção Vendas:**

Agregação na coleção "Sales\_Header" envolve $lookup com as coleções "Sales\_Lines" e "Valor\_Total\_Linhas". O $project é utilizado para selecionar e renomear campos específicos, e, por fim, o resultado é armazenado na coleção "Vendas".

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

Descrição gerada automaticamente

Figura 4: Organização dos dados na coleção Vendas

1. **Coleção NovaDevolucoes:**

Esta coleção serviu de apoio para a formação da coleção Devoluções, apesar de já não estar presente na base de dados, achamos pertinente explicá-la porque fez parte do processo de estruturação dos dados. Para a coleção "Devoluções", é feito um $lookup com a coleção "Vendas". O $unwind é aplicado para desdobrar os documentos resultantes do $lookup. A fase $project é então usada para selecionar e calcular a diferença em dias entre a venda e a devolução. O resultado é armazenado na nova coleção "NovaDevolucoes".

### BaseX

Em termos de BaseX, criamos uma API com as funções necessárias à implementação dos relatórios, assim como as funções que retornam os próprios relatórios. Sendo assim, a nossa API é composta por 8 funções, as quais vamos enumerar e resumir agora.

**Função getClients()**

**Objetivo:** Recuperar informações dos clientes de uma fonte de dados MongoDB.

**Operações:**

Definição das variáveis $url e $api-key para a URL e chave da API do MongoDB.

Construção da string JSON $json-data para especificar a coleção e filtros desejados.

Utilização do comando http:send-request para enviar uma solicitação POST ao MongoDB.

Iteração sobre os resultados obtidos para criar elementos XML representando os clientes.

A função retorna informações detalhadas sobre cada cliente, incluindo ID, nome, email, endereço, tipo de cliente, compras nos últimos 3 anos e valor total de compras.

**Função getProdutos()**

**Objetivo:** Recuperar informações dos produtos de uma fonte de dados MongoDB.

**Operações:**

Definição das variáveis $url e $api-key para a URL e chave da API do MongoDB.

Construção da string JSON $json-data para especificar a coleção e filtros desejados.

Utilização do comando http:send-request para enviar uma solicitação POST ao MongoDB.

Iteração sobre os resultados obtidos para criar elementos XML representando os produtos, com lógica para categorizar informações adicionais.

A função retorna informações detalhadas sobre cada produto, incluindo código, marca, modelo, preço e categorias específicas.

**Função getVendas()**

**Objetivo:** Recuperar informações sobre vendas de uma fonte de dados MongoDB.

**Operações:**

Definição das variáveis $url e $api-key para a URL e chave da API do MongoDB.

Construção da string JSON $json-data para especificar a coleção e filtros desejados.

Utilização do comando http:send-request para enviar uma solicitação POST ao MongoDB.

Iteração sobre os resultados obtidos para criar elementos XML representando as vendas e suas linhas.

A função retorna informações sobre cada venda, incluindo código da fatura, data, código do cliente, valor total e detalhes das linhas de venda.

**Função getRelatorioVendas($ano as xs:integer, $mes as xs:integer)**

**Objetivo:** Gerar um relatório de vendas para um ano e mês específicos, combinando dados de clientes, produtos e vendas.

**Operações:**

Chamada às funções page:getClients(), page:getProdutos(), e page:getVendas() para obter informações necessárias.

Manipulação e filtragem dos dados para gerar um relatório estruturado em XML.

O relatório inclui informações gerais, dados dos clientes ativos, produtos vendidos, vendas no mês e um resumo estatístico.

**Função postRelatorioVendas($ano as xs:integer, $mes as xs:integer)**

**Objetivo:** Gerar e salvar um relatório de vendas para um ano e mês específicos em um arquivo XML.

**Operações:**

Chamada à função page:getRelatorioVendas($ano, $mes) para gerar o relatório.

Definição da variável $caminhoDoArquivo para especificar o local onde o arquivo XML será salvo.

Utilização do comando file:write para escrever o conteúdo do relatório no arquivo.

**Função getDevolucoes()**

**Objetivo:** Recuperar informações sobre devoluções de uma fonte de dados MongoDB.

**Operações:**

Definição das variáveis $url e $api-key para a URL e chave da API do MongoDB.

Construção da string JSON $json-data para especificar a coleção e filtros desejados.

Utilização do comando http:send-request para enviar uma solicitação POST ao MongoDB.

Iteração sobre os resultados obtidos para criar elementos XML representando as devoluções.

A função retorna informações sobre cada devolução, incluindo ID da fatura, data, dias entre venda e devolução, status de devolução precoce e lista de produtos devolvidos.

**Função getRelatorioDevolucoes($ano as xs:integer, $mes as xs:integer)**

**Objetivo:** Gerar um relatório de devoluções para um ano e mês específicos, combinando dados de produtos e devoluções.

**Operações:**

Chamada às funções page:getProdutos() e page:getDevolucoes() para obter informações necessárias.

Manipulação e filtragem dos dados para gerar um relatório estruturado em XML.

O relatório inclui informações gerais, produtos devolvidos, devoluções no mês e um resumo estatístico.

**Função postRelatorioDevolucoes($ano as xs:integer, $mes as xs:integer)**

**Objetivo:** Gerar e salvar um relatório de devoluções para um ano e mês específicos em um arquivo XML.

**Operações:**

Chamada à função page:getRelatorioDevolucoes($ano, $mes) para gerar o relatório.

Definição da variável $caminhoDoArquivo para especificar o local onde o arquivo XML será salvo.

Utilização do comando file:write para escrever o conteúdo do relatório no arquivo.

## Conclusão

Em jeito de conclusão, a realização deste trabalho fez-nos aprofundar os nossos conhecimentos em termos de estruturação de dados e o processamento dos mesmos. Em termos de constrangimentos, podemos apenas sublinhar algumas dificuldades em utilizar os softwares que foram, eventualmente, ultrapassadas com ajuda dos professores e pesquisas na WEB. Como já referido acima, os objetivos não foram 100% concluídos tendo em conta uma adversidade no momento de verificar os XSD’s.