ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

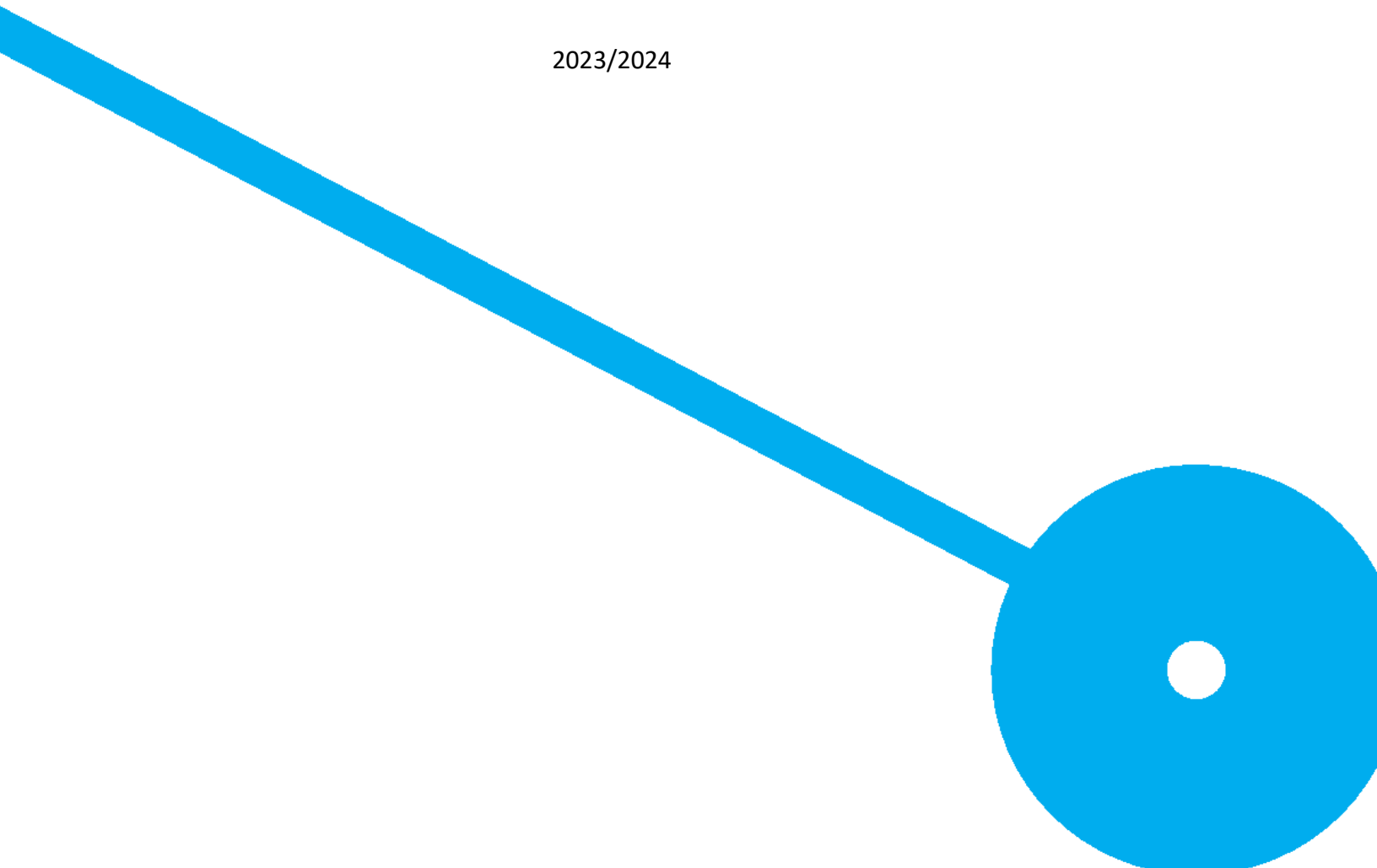
LICENCIATURA DE ENGENHARIA INFORMÁTICA / SEGURANÇA INFORMÁTICA EM REDES DE COMPUTADORES

Processamento Estruturado de Informação

Trabalho Prático

João Rafael Da Cunha Guerra- 8200098

2023/2024



P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Conteúdo

1. Introdução

1.1. Contextualização do problema

A Phone for You é uma empresa que vende smartphones através de várias lojas em parceria a nível nacional. A empresa conta com diversos parceiros responsáveis pelas vendas, e não só fornece os equipamentos para as lojas, mas também gere o processo de devolução de produtos.

Devido ao considerável crescimento nos últimos anos, a Phone for You deseja que cada um de seus parceiros apresente, mensalmente, um relatório de vendas dos smartphones, incluindo informações sobre as devoluções realizadas. Para facilitar esse processo, a empresa decidiu disponibilizar um vocabulário XML, que deve ser utilizado por cada parceiro na implementação de um módulo em seus sistemas informáticos. Esse módulo terá a função de gerar documentos XML contendo dados essenciais sobre vendas e devoluções, conforme especificado no vocabulário fornecido.

1.2. Ferramentas Utilizadas

Para o desenvolvimento do trabalho foram utilizadas as seguintes ferramentas.

Oxgynen

Foi utilizado a ferramenta Oxygen para criar os relatórios de vendas e devoluções em formato XML, desenvolvendo também os seus esquemas XML (XSD) correspondentes.

MongoDB

MongoDB foi utilizado para desenvolver as consultas necessárias a fim de extrair as informações essenciais e apoiar o processo de exportação. Além disso, foi utilizado o MongoDB Atlas para criar a base de dados no cluster.

BaseX

Foi utilizado a funcionalidade de fazer pedidos HTTP ao URL endpoint da data API e posteriormente retornar os resultados desse pedido.

2. Criação do vocabulário XML para Vendas e para Devoluções

2.1.Relatório Vendas

Inicialmente, foi criado um arquivo XSD denominado "Relatorio_vendas.xsd", que contém os elementos essenciais para representar um relatório de vendas. Os elementos incluídos neste contexto são "Parceiro", "Clientes", "Produtos", "TipoVenda" e "Resumo".

```
<xs:element name="relatorio_vendas">
    <xs:complexType>
       <xs:sequence>
           <xs:element name="parceiro" maxOccurs="unbounded" type="pa:TipoParceiro"/>
           <xs:element name="Clientes" maxOccurs="unbounded" type="c:TipoClientes"/>
           <xs:element name="Produtos" maxOccurs="unbounded" type="p:TipoProdutos"></xs:element>
           <xs:element name="TipoVenda" maxOccurs="unbounded" type="v:TipoVenda"></xs:element>
           <xs:element name ="Resumo" maxOccurs="unbounded" type="r:resumoVendas"></xs:element>
       </xs:sequence>
```

Em seguida, cada elemento foi associado a um tipo específico, e para cada tipo, foi desenvolvido um arquivo XSD correspondente.

Para o "TipoParceiro", foram adicionados os seguintes atributos: NIF, nome, morada, Ano fiscal e Mês. Uma restrição foi aplicada ao atributo NIF para aceitar apenas 9 dígitos, enquanto para o atributo Ano fiscal, uma restrição foi estabelecida para aceitar apenas 4 dígitos.

```
<xs:simpleType name="TipoNif">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{9}"></xs:pattern>
    </xs:restriction>
</xs:simpleType>

    <xs:simpleType name="TipoAnoFiscal">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{4}"></xs:pattern>
        </xs:restriction>
    </xs:simpleType>
```

No tipo de cliente, foram acrescentados sete elementos: primeiro nome, último nome, e-mail, morada, tipo de cliente, compras nos últimos 3 anos e valor total. Para o elemento "email", foi estabelecido um valor padrão de "desconhecido", assegurando que, caso o cliente não tenha um endereço de e-mail, seja atribuído automaticamente o valor default. Quanto ao elemento "morada", foi designado um tipo "TipoMorada", que inclui três elementos: país, cidade e código postal, sendo que este último possui uma restrição de aceitar apenas 5 dígitos.

Além disso, para o tipo de cliente, foi aplicada uma restrição do tipo enumeração, garantindo que apenas os valores "novo", "premium" e "regular" sejam aceitos.

```
<xs:complexType name="TipoClientes">
    <xs:sequence>
        <xs:element name="Primeiro_nome" type="xs:string"/>
        <xs:element name="Ultimo_nome" type="xs:string"/>
        <xs:element name="Email" type="xs:string" default="Desconh
        <xs:element name="Morada" type="TipoMorada"/>
        <xs:element name="Tipo_cliente" type="TipoCliente"/>
        <xs:element name="compras_3anos" type="xs:integer"/>
        <xs:element name="ValorTotal" type="xs:decimal"/>
    </xs:sequence>
</xs:complexType>


<xs:complexType name="TipoMorada">
    <xs:sequence>
        <xs:element name="País" type="xs:string"/>
        <xs:element name="Cidade" type="xs:string"/>
        <xs:element name="Codigo_Postal">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern value="\d{5}"></xs:pattern> <!-- I
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>


<xs:simpleType name="TipoCliente">
    <xs:restriction base="xs:string">
        <xs:enumeration value="novo"/>
        <xs:enumeration value="regular"/>
        <xs:enumeration value="premium"/>
    </xs:restriction>
</xs:simpleType>
```

Para o "TipoProduto", foram definidos os elementos "Código do Produto", "Marca", "Modelo", "Preço Atual" e "Categoria".

Para a "Categoria", foi criado um tipo chamado "TipoCategoria", que inclui os elementos "Nome", "Gama de Preços", "Desempenho", "Qualidade da Câmera", "Tamanho do Ecrã", "Capacidade da Bateria" e "Capacidade de Armazenamento". Cada um desses elementos em "TipoCategoria" possui uma restrição do tipo enumeração, permitindo apenas valores específicos de acordo com a enumeração correspondente.

```
<xs:complexType name="TipoProdutos">
    <xs:sequence>
        <xs:element name="codigo" type="xs:string"/>
        <xs:element name="marca" type="xs:string"/>
        <xs:element name="modelo" type="xs:string"/>
        <xs:element name="PrecoAtual" type="xs:decimal"/>
        <xs:element name="Categoria" type="TipoCategoria"></xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="TipoCategoria">
    <xs:sequence>
        <xs:element name="gama_precos" type="GamaPrecos"/>
        <xs:element name="desempenho" type="Desempenho"/>
        <xs:element name="qualidade_camera" type="QualidadeCamera"/>
        <xs:element name="tamanho_ecra" type="TamanhoEcras"/>
        <xs:element name="capacidade_bateria" type="CapacidadeBateria"/>
        <xs:element name="capacidade_armazenamento" type="CapacidadeArmazenamento"/>
    </xs:sequence>
</xs:complexType>
```

Exemplo do “GamaPreços”.

```
<xs:simpleType name="GamaPrecos">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Budget"/>
        <xs:enumeration value="Mid-Range"/>
        <xs:enumeration value="High-End"/>
    </xs:restriction>
</xs:simpleType>
```

Para o "TipoVenda", foram definidos os elementos "Código da Fatura", "Data da Venda", "Código do Cliente", "Valor Total da Venda" e "Linhas de Venda". Cada linha de venda possui os elementos "Número da Linha", "Código do Produto", "Quantidade" e "Valor Total da Linha de Venda".

No caso do "Resumo", foram definidos os elementos "Número de Produtos", "Total de Vendas", "Número de Clientes" e "Vendas das Categorias".

2.2. Relatório de devoluções

No relatório de devoluções, são incluídos quatro elementos: "Parceiro", "Produtos", "Devolução" e "Resumo". O “Parceiro” e “Produtos” foram reutilizados pois o esquema era o mesmo.

Para a "Devolução", é introduzido o tipo "TipoDevolução", que contém cinco elementos: "Código da Fatura", "Data da Fatura", "Código do Produto", "Dias de Devolução" e "Devolução Precoce". A restrição de enumeração aplicada ao elemento "Devolução Precoce" permite apenas os valores "sim" ou "não".

O "Resumo" é composto por dois elementos: "Número de Produtos" e "Número de Devoluções por Categoria". Esses elementos resumem as informações relacionadas aos produtos e ao número de devoluções, categorizadas por categoria.

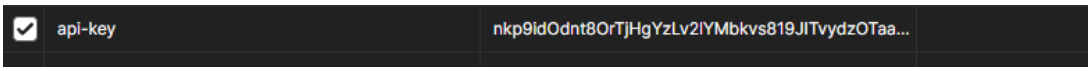
3. MongoDB

3.1. MongoDB atlas Data API

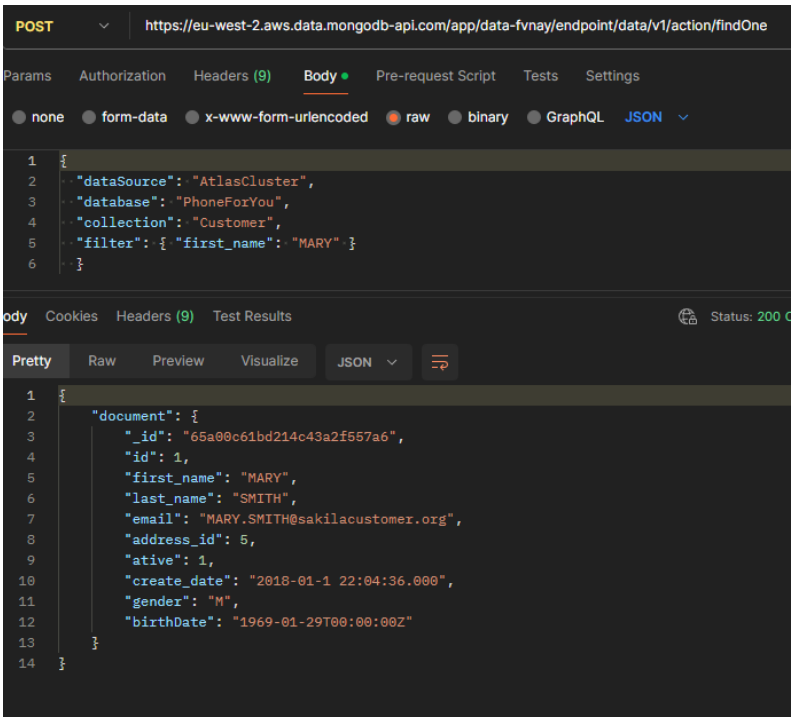
Uma base de dados chamada “PhoneForYou” foi criada no cluster no MongoDB Atlas.

Em seguida, foram criadas 11 coleções, cada uma correspondendo a um arquivo CSV fornecido.

Para testar a data API, no Postman, a API Key foi adicionada aos cabeçalhos. Isso ajuda a autenticar as solicitações e a garantir que apenas utilizadores autorizados tenham acesso aos dados da base de dados.



Após a configuração da API key, foi utilizado o método POST no Postman, incluindo o URL do endpoint do cluster do MongoDB. A rota “/action/findOne” foi adicionada á url para realizar uma solicitação específica de encontrar um único documento na coleção.



3.2. Modelação de dados

Para evitar múltiplos lookups, os produtos foram misturados com suas categorias. Isso envolveu a execução de lookups nas coleções "sub_category_product", "sub_category" e "Category", associando IDs e realizando unwind para evitar arrays. Um estágio de group foi usado para agrupar dados e projetar campos específicos. E nesse group foi criada uma array de categorias usando $addToSet, e $cond foi usado para adicionar condições específicas à array, por exemplo, caso o nome da categoria fosse “Price Range” então ele projetava que Price Range era o nome da subcategoria, caso não fosse avançava para próxima condição.

Por fim, $set foi utilizado para misturar a array em um único objeto, seguido pelo merge para criar uma nova coleção.


```
_id: ObjectId('65a00c67bd214c43a2f55a1c')
▾ Categories: Object
    Screen Size: "Medium (5-6.4 inches)"
    Battery Life: "Average Battery Life"
    Performance: "Standard Performance"
    Storage Capacity: "High Storage"
    Camera Quality: "Good Cameras"
    Price Range: "High-End Smartphones"
  brand: "apple"
  id: 31
  list_price: 129990
  model: "Apple iPhone 14 Pro (256GB)"
```

```
_id: ObjectId('65a00c67bd214c43a2f55abe')
▾ Categories: Object
    Price Range: "Budget Smartphones"
    Screen Size: "Large (6.5 inches and above)"
    Battery Life: "Average Battery Life"
    Camera Quality: "Basic Cameras"
    Performance: "Basic Performance"
    Storage Capacity: "Low Storage"
  brand: "letv"
  id: 193
  list_price: 5499
  model: "Letv Y1 Pro"
```

Para a coleção "address", adotou-se uma abordagem semelhante, considerando que um endereço sempre terá o mesmo país e cidade. Vários lookups foram realizados para associar a cidade e o país ao endereço. Ao final do processo, a operação $out foi utilizada para atualizar a coleção existente com os novos campos, como cidade e país.


```
_id: ObjectId('65a00bc5b9548d88c83a07e8')
address_id: 1
address: "47 MySakila Drive"
address2: null
district: " "
city_id: 300
postal_code: null
Cidade: "Lethbridge"
País: "Canada"
```

```
_id: ObjectId('65a00bc5b9548d88c83a07fe')
address_id: 23
address: "1417 Lancaster Avenue"
address2: null
district: " "
city_id: 267
postal_code: 72192
Cidade: "Kimberley"
País: "South Africa"
```


Na coleção "sales_header", dois lookups foram realizados para associar o cliente a "sales_header". Em seguida, na coleção "sales_line", as linhas de vendas foram associadas aos clientes. Posteriormente, foi utilizado um estágio de group para agrupar os documentos pelo ID de "sales_header". Durante esse processo, alguns campos específicos foram projetados e um novo campo chamado "Valor Total" foi criado. O cálculo desse valor total envolveu a multiplicação da quantidade pelo “total_with_vat" para cada linha de venda, seguida pela soma dos valores resultantes.

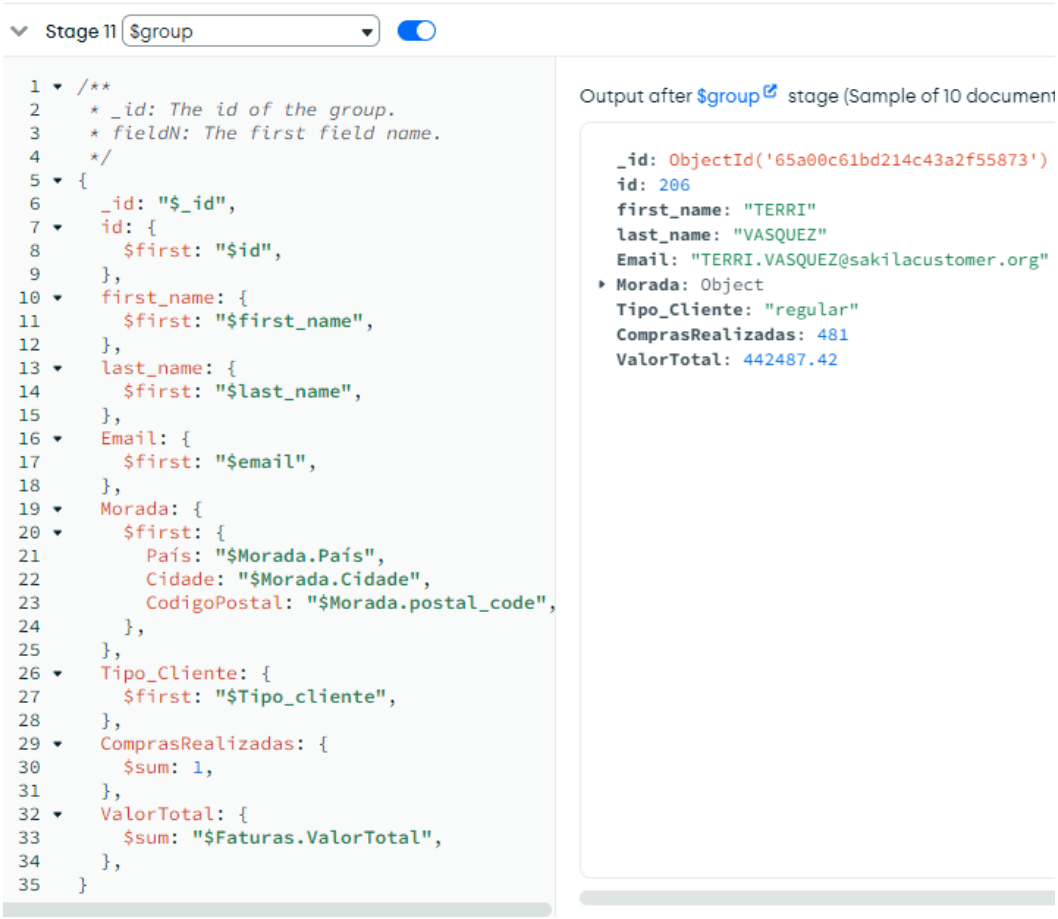
Por fim, foi utilizado o estágio "$out" para atualizar a coleção existente, garantindo que ela agora incluísse o parâmetro de "Valor Total" para cada fatura.

Na coleção "Customer", foi realizado um lookup para associar a morada ao cliente. Em seguida, utilizando o estágio "$addFields", foi criado um parâmetro chamado "TipoCliente", que calcula se o cliente é classificado como novo, premium ou regular, para isso foi usada um switch case, onde fazia a subtração da data atual e da criação do cliente e divida por 31536000000(3 anos em milissegundos) e caso fosse menor que 1 ele classificava como “novo”, menor que 5 anos classificava como “regular”, e caso não fosse nenhum destes 2 casos, classificava como “premium”.

```
Tipo_cliente: {
  $switch: {
    branches: [
      {
        case: {
          $lt: [
            {
              $divide: [
                {
                  $subtract: [
                    "$$NOW",
                    {
                      $dateFromString: {
                        dateString:
                          "$create_date",
                      },
                    },
                  ],
                },
                31536000000,
              ],
            },
            1,
          ],
        },
        then: "novo",
      },
      {
        case: {
          $lte: [
            {
              $divide: [
                {
                  $subtract: [
                    "$$NOW",
                    {
```

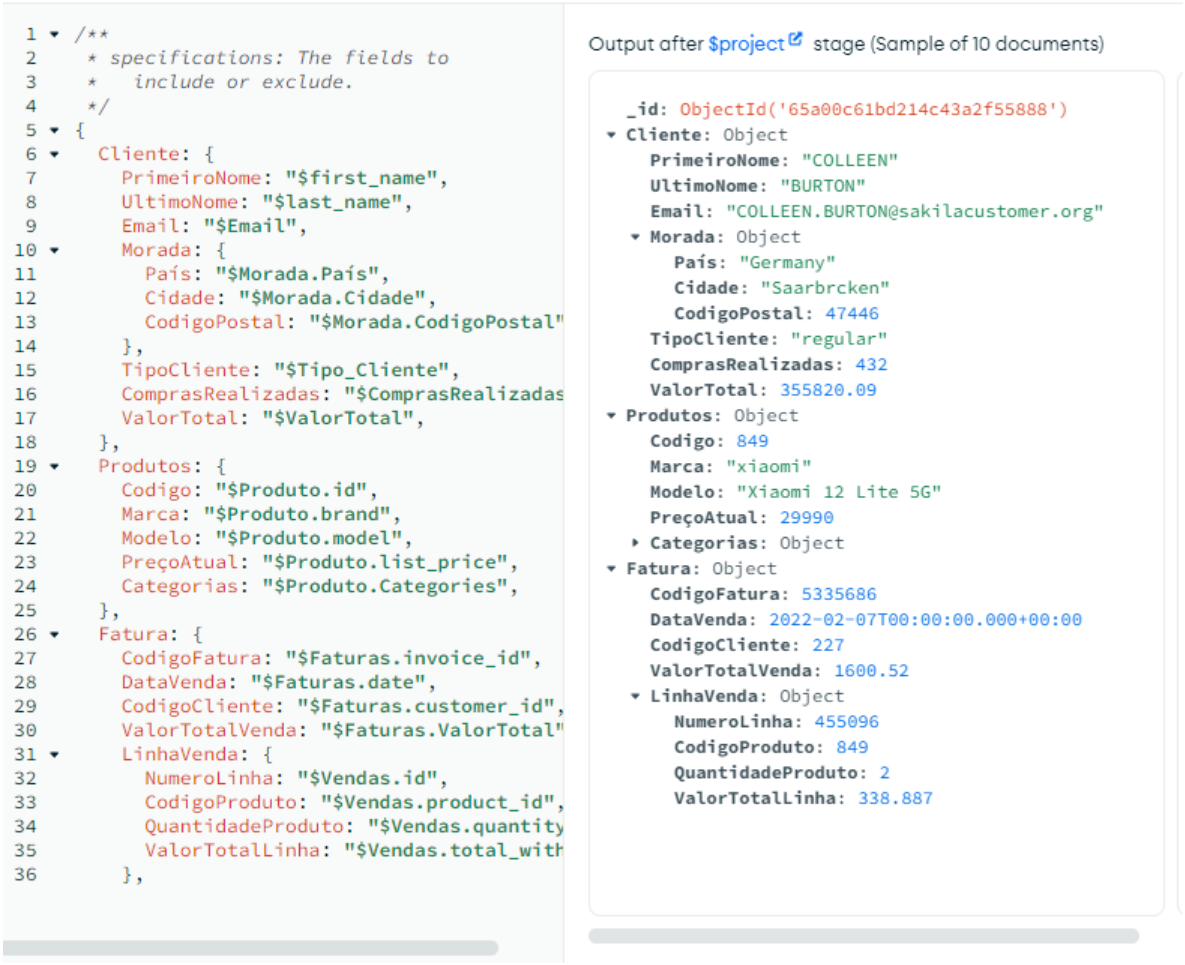
Após essa etapa, foi realizado um novo lookup para associar todas as faturas a um cliente.

Posteriormente, foi utilizado o estágio "$match" para filtrar apenas as faturas com menos de 3 anos. Após essa filtragem, foi utilizado o estágio "$group" para agrupar os documentos pelo ID e projetar alguns campos específicos e foram realizadas somas para calcular o número de compras realizadas ("ComprasRealizadas") e o valor total das faturas nos últimos 3 anos.



Finalmente, foi realizado um merge para criar uma nova coleção, incorporando todas as transformações e agregações efetuadas nas etapas anteriores.

Na nova coleção, foram realizados lookups para associar as coleções "sales_header" aos "customer", "sales_lines" aos "sales_header", e "produtos" às "sales_lines". Em seguida, foi utilizado um estágio de projeção para incluir todas as informações necessárias na coleção resultante.



```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  Cliente: {
    PrimeiroNome: "$first_name",
    UltimoNome: "$last_name",
    Email: "$Email",
    Morada: {
      País: "$Morada.País",
      Cidade: "$Morada.Cidade",
      CodigoPostal: "$Morada.CodigoPostal"
    },
    TipoCliente: "$Tipo_Cliente",
    ComprasRealizadas: "$ComprasRealizadas
    ValorTotal: "$ValorTotal",
  },
  Produtos: {
    Codigo: "$Produto.id",
    Marca: "$Produto.brand",
    Modelo: "$Produto.model",
    PreçoAtual: "$Produto.list_price",
    Categorias: "$Produto.Categories",
  },
  Fatura: {
    CodigoFatura: "$Faturas.invoice_id",
    DataVenda: "$Faturas.date",
    CodigoCliente: "$Faturas.customer_id",
    ValorTotalVenda: "$Faturas.ValorTotal"
    LinhaVenda: {
      NumeroLinha: "$Vendas.id",
      CodigoProduto: "$Vendas.product_id",
      QuantidadeProduto: "$Vendas.quantity
      ValorTotalLinha: "$Vendas.total_with
    },
```

Output after $project stage (Sample of 10 documents)

```
_id: ObjectId('65a00c61bd214c43a2f55888')
Cliente: Object
  PrimeiroNome: "COLLEEN"
  UltimoNome: "BURTON"
  Email: "COLLEEN.BURTON@sakilacustomer.org"
  Morada: Object
    País: "Germany"
    Cidade: "Saarbrcken"
    CodigoPostal: 47446
  TipoCliente: "regular"
  ComprasRealizadas: 432
  ValorTotal: 355820.09
Produtos: Object
  Codigo: 849
  Marca: "xiaomi"
  Modelo: "Xiaomi 12 Lite 5G"
  PreçoAtual: 29990
  Categorias: Object
Fatura: Object
  CodigoFatura: 5335686
  DataVenda: 2022-02-07T00:00:00.000+00:00
  CodigoCliente: 227
  ValorTotalVenda: 1600.52
  LinhaVenda: Object
    NumeroLinha: 455096
    CodigoProduto: 849
    QuantidadeProduto: 2
    ValorTotalLinha: 338.887
```

Na coleção "returns", foram realizados lookups associando as "sales_header" ao "invoice" de "returns". Em seguida, foram criados dois parâmetros: "DiasAteDevolucao" e "DevolucaoPrecoce".

Para o parâmetro "DiasAteDevolucao", foi realizado um cálculo onde a diferença de dias entre a data de "returns" e a data de "sales_header" foi calculada, dividindo-se pelo valor 86400000 (correspondente a um dia em milissegundos).

Para o parâmetro "DevolucaoPrecoce", foi estabelecida uma condição: se a diferença entre as datas de "returns" e "sales_header" fosse menor que 259200000 (correspondente a 3 dias em milissegundos), o valor era definido como "sim"; caso contrário, era definido como "não".

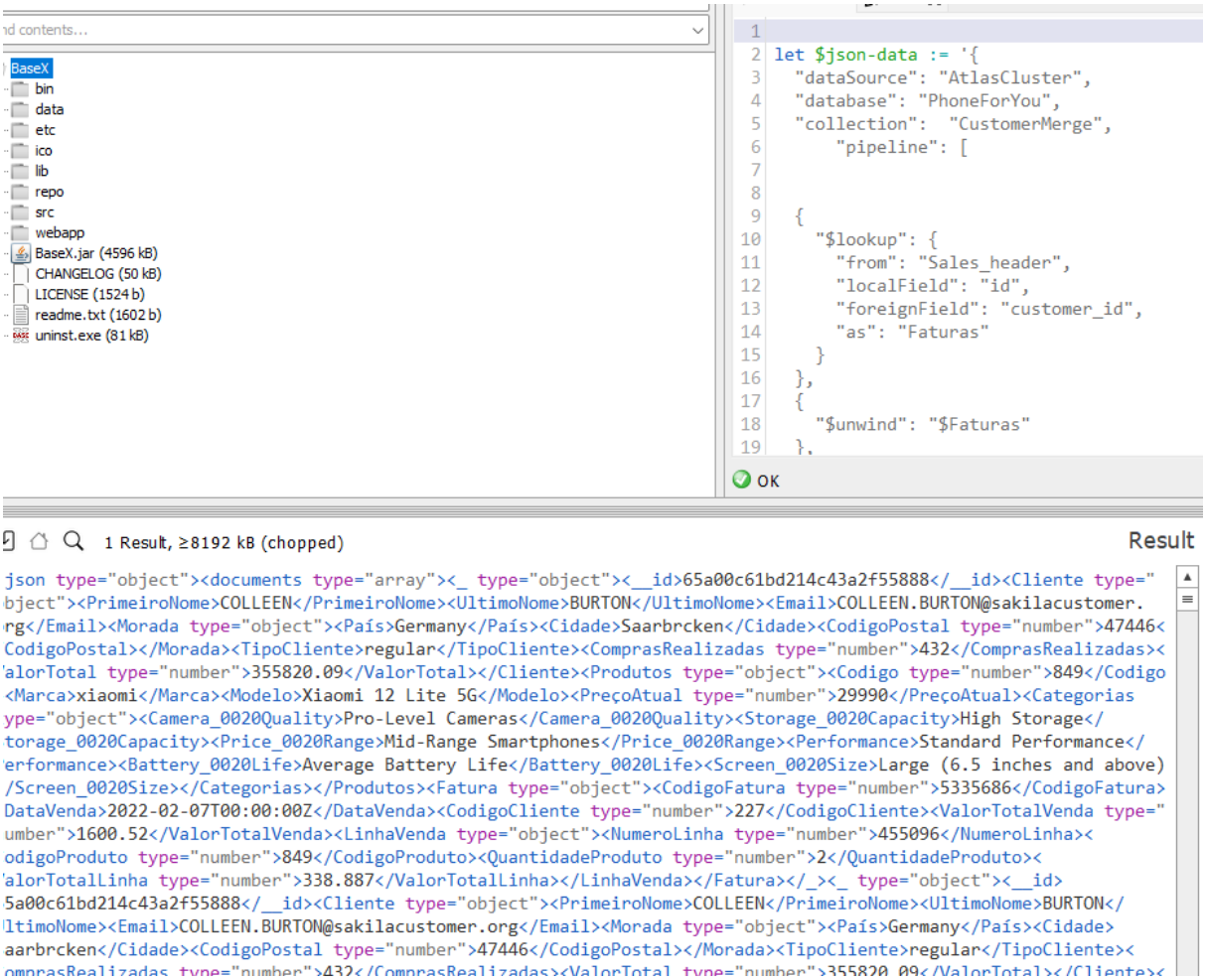
```
{
  DiasAteDevolucao: {
    $divide: [
      {
        $subtract: [
          "$date",
          "$Faturas.date",
          // Substitua por seu campo real
        ],
      },
      86400000, // 1 dia em milissegundos
    ],
  },
  DevolucaoPrecoce: {
    $cond: {
      if: {
        $lte: [
          {
            $subtract: [
              "$date",
              // Substitua por seu campo r
              "$Faturas.date",
            ],
          },
          259200000, // 3 dias em milisseg
        ],
      },
      then: "Sim",
      else: "Não",
    },
  },
}
```

Output after $addFields stage (Sample of 10 documents)

```
_id: ObjectId('65a00d2dab05e15386492feb')
invoice_id: 1000021
product_id: 664
date: 2022-05-27T00:00:00.000+00:00
Faturas: Object
  _id: ObjectId('65a00c71bd214c43a2f55dd4')
  invoice_id: 1000021
  date: 2022-05-27T00:00:00.000+00:00
  customer_id: 213
  ValorTotal: 1483.47
DiasAteDevolucao: 0
DevolucaoPrecoce: "Sim"
```

3.3.BaseX

No BaseX, foi elaborada uma consulta XQuery que efetua um pedido HTTP à sua pipeline na base de dados do cluster do MongoDB e, em seguida, retorna a resposta desse pedido.



4. Conclusão

Este projeto foi uma oportunidade valiosa de aprendizado, durante a qual foi possível adquirir conhecimentos práticos em modelagem de dados, integração eficiente de coleções no MongoDB e desenvolvimento de consultas. A experiência também abrangeu a criação de XML e XSD, fornecendo uma compreensão mais aprofundada sobre a estruturação e validação de dados.

4.1. Limitações

A criação da API enfrentou obstáculos na implementação de dois recursos cruciais: a obtenção de relatórios de vendas e devoluções em XML para meses específicos. A incapacidade de concluir com êxito essa tarefa deve-se principalmente à má gestão do tempo e à falta de conhecimento em áreas específicas, dificultando a configuração adequada da API. Além disso, a realização de pedidos HTTP por meio do Postman também foi comprometida por essas limitações.

Adicionalmente, destaca-se que não foi possível implementar o resumo de ambos os relatórios na construção do vocabulário XML.