

Trabalho Prático de Grupo

Processamento Estruturado de Informação

Alunos:

David Santos: 8220651

João Martinez: 8210487

Ricardo Ferreira: 8220132

Professor:

Bruno Oliveira

Pedro Pinto

Índice

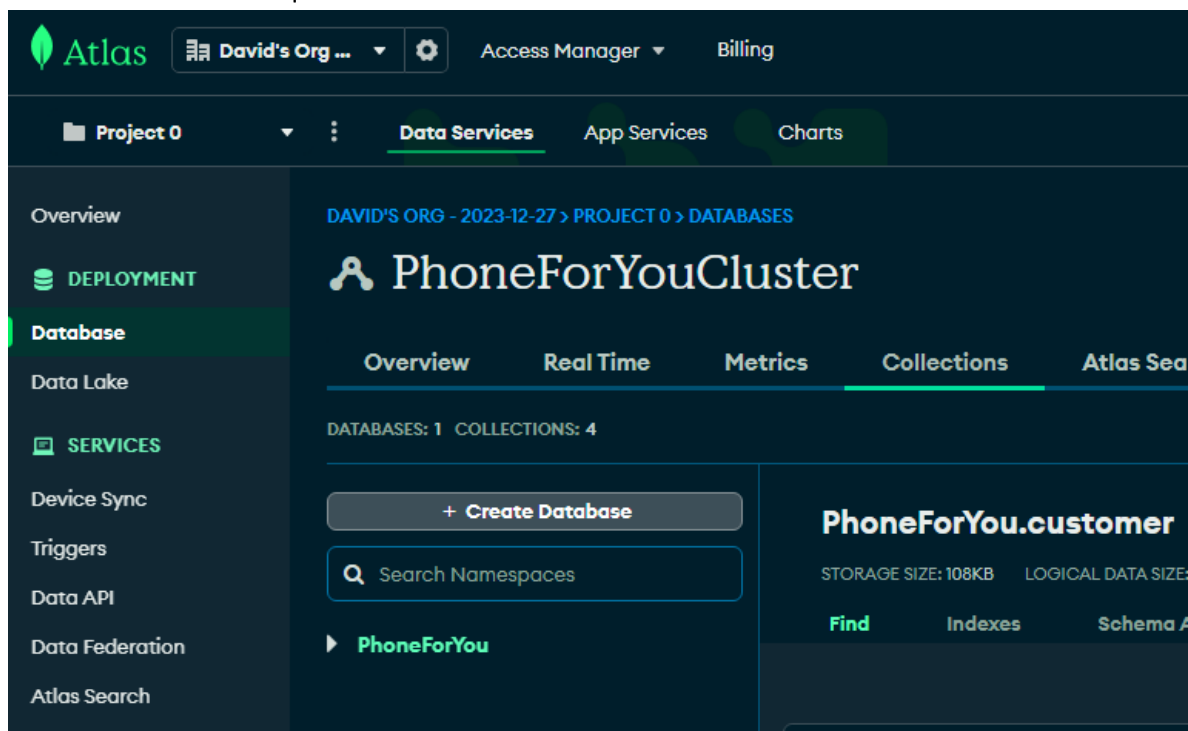
Índice	2
1. Importação de Dados para MongoDB:.....	3
2. Criação do cluster e Ativação da Data API no Mongo Atlas:	3
3. Modelagem e Estruturação de Dados no MongoDB:	4
4. Desenvolvimento de Consultas MongoDB:	8
5. Implementação da API em BaseX:	10
6. Pedido HTTP à API do Mongo Atlas usando BaseX:	10
7. Funções em BaseX para Transformação em XML:	12
8. Postman para efetuar os requests ao BaseX	13
Conclusão	14

1. Importação de Dados para MongoDB:

A importação dos dados, inicialmente em formato CSV, para o MongoDB. Optou-se por essa escolha, devido á flexibilidade do MongoDB, que permite a manipulação de documentos JSON. A estruturação em documentos no MongoDB, comparada às tabelas em bases de dados relacionais, realiza uma análise cuidadosa da carga de trabalho. O alinhamento com as diretrizes do MongoDB, documentadas, é fundamental para garantir a integridade do modelo de dados.

2. Criação do cluster e Ativação da Data API no Mongo Atlas:

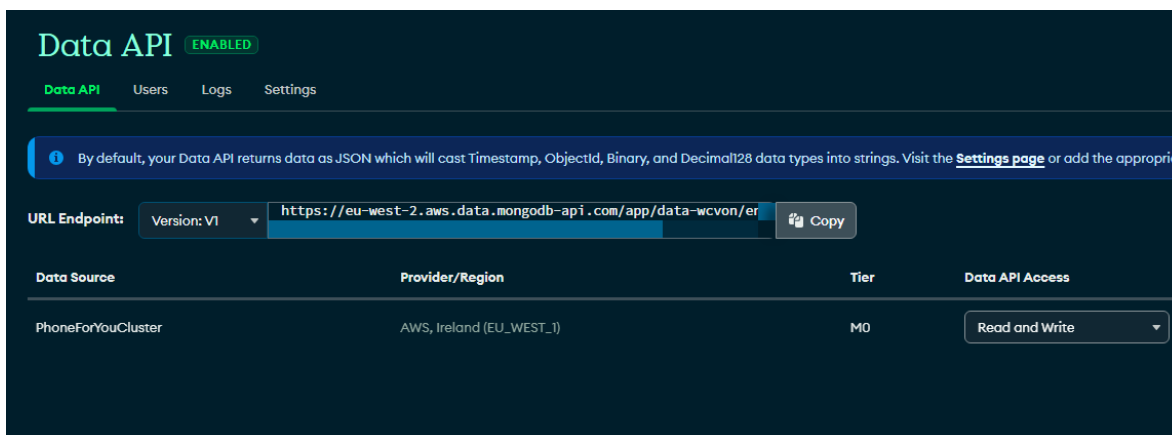
Este projeto foi desenvolvido usando o Mongo Atlas. Primeiramente, foi criado um cluster de nome “PhoneForYouCluster” que é armazena a nossa base de dados “PhoneForYou”.



Com a base de dados criada, passamos para a fase de ativação da Data API do Antlas. A ativação da Data API no Mongo Atlas proporciona um meio eficiente de acesso aos dados armazenados. A

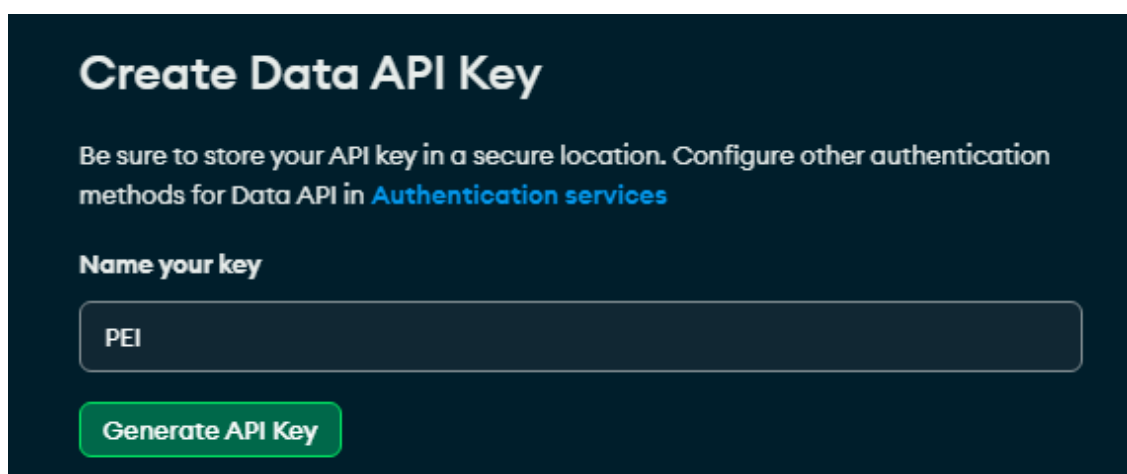
opção por HTTP amplia a acessibilidade e simplifica a integração de sistemas. Essa etapa é fundamental para garantir que a API desenvolvida posteriormente possa recuperar os dados necessários de maneira eficaz, cumprindo as exigências de desempenho e segurança.

A nossa URL endpoint: <https://eu-west-2.aws.data.mongodb-api.com/app/data-wcvon/endpoint/data/v1>



The screenshot shows the MongoDB Data API console. At the top, it says "Data API" with a green "ENABLED" badge. Below this are tabs for "Data API", "Users", "Logs", and "Settings". A blue banner contains a tip: "By default, your Data API returns data as JSON which will cast Timestamp, ObjectId, Binary, and Decimal128 data types into strings. Visit the [Settings page](#) or add the appropriate options to your request." Below the banner, the "URL Endpoint" section shows a dropdown for "Version: V1" and a text input containing the URL "https://eu-west-2.aws.data.mongodb-api.com/app/data-wcvon/er". A "Copy" button is next to the URL. Below this is a table with the following columns: "Data Source", "Provider/Region", "Tier", and "Data API Access". The table has one row: "PhoneForYouCluster", "AWS, Ireland (EU_WEST_1)", "M0", and a dropdown menu set to "Read and Write".

Data Source	Provider/Region	Tier	Data API Access
PhoneForYouCluster	AWS, Ireland (EU_WEST_1)	M0	Read and Write



The screenshot shows the "Create Data API Key" form. It has a title "Create Data API Key" and a subtitle "Be sure to store your API key in a secure location. Configure other authentication methods for Data API in [Authentication services](#)". Below this is a section "Name your key" with a text input field containing "PEI". At the bottom is a green button labeled "Generate API Key".

A nossa API-KEY: nlS5YxG0lkzw7IfmTys39aI7UKv8vB2ROA8kR1g0Qxr33KYkjDAbZSkCRpG7CqMb

3. Modelagem e Estruturação de Dados no MongoDB:

Com a base de dados criada e a Data API ativada, passamos para a fase de modulação das nossas coleções. A nossa estratégia de estruturação dos dados no MongoDB foi baseada em quatro coleções: customer, sales, products e returns. Cada coleção contém junções eficientes entre as coleções dadas para consolidar informações relevantes numa única coleção, proporcionando facilidade na consulta, atualização e compreensão dos dados. Desta maneira tem vantagens

significativas, incluindo a simplificação das operações de manipulação de dados, facilitando a consulta e a atualização de informações correlacionadas em um único documento.

Para a coleção `customer` optou-se por embutir em cada documento `customer` as informações a ele relacionadas. Esse processo baseou-se na junção de `customer`, `address`, `city` e `country` que foram fornecidas permitindo que as informações do cliente sejam facilmente consultadas tendo sempre como foco, o vocabulário exigido.

The screenshot shows the PhoneForYou database interface. On the left, a sidebar lists collections: `customer` (highlighted), `product`, `returns`, `sales`, `admin`, and `local`. The main area displays a document from the `customer` collection. The document structure is as follows:

```
{
  "_id": ObjectId('65a7c58dd3faedccc7ca773f'),
  "id": 1,
  "first_name": "MARY",
  "last_name": "SMITH",
  "email": "MARY.SMITH@sakilacustomer.org",
  "ative": 1,
  "create_date": 2021-02-14T22:04:36.000+00:00,
  "gender": "M",
  "birthDate": 1969-01-29T00:00:00.000+00:00,
  "address": {
    "address": "1913 Hanoi Way",
    "postal_code": 35200,
    "city": "Sasebo",
    "country": "Japan"
  },
  "last3years": {
    "total_purchases": 909,
    "total_spent": 1144169.4328,
    "latest_purchase_date": 2023-12-31T00:00:00.000+00:00,
    "customer_type": "regular"
  }
}
```

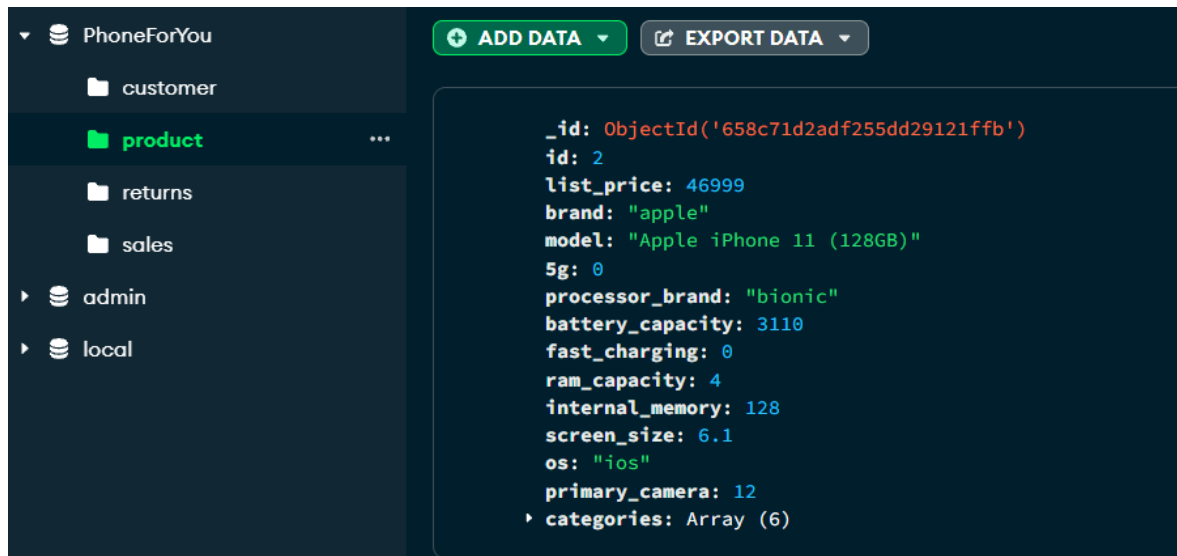
O mesmo acontece para a coleção `sales` e a coleção `products`. Em ambos optou-se também por embutir os seus dados relacionados.

Começando por `sales` que optou-se por embutir a `sales_lines` na coleção `sales_header`, de forma a facilitar as consultas posteriormente desenvolvidas para satisfazer o vocabulário.

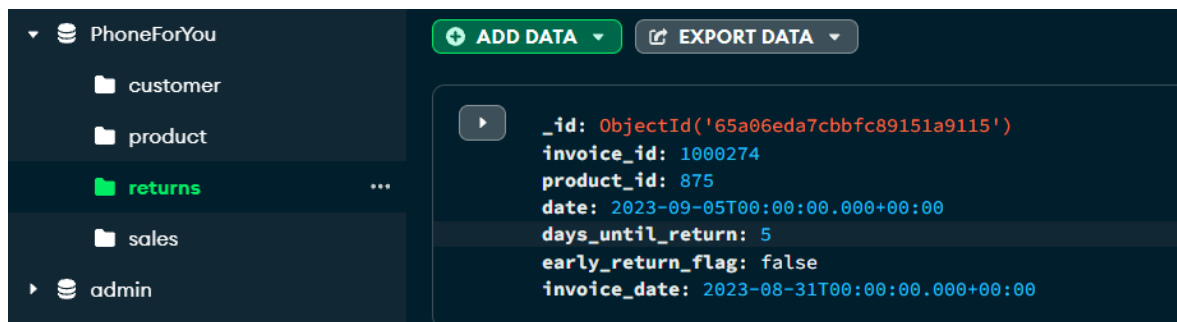
The screenshot shows the PhoneForYou database interface. On the left, a sidebar lists collections: `customer`, `product`, `returns`, `sales` (highlighted), `admin`, and `local`. The main area displays a document from the `sales` collection. The document structure is as follows:

```
{
  "_id": ObjectId('65a7f730d3faedccc7ca92a4'),
  "invoice_id": 11,
  "date": 2022-11-21T00:00:00.000+00:00,
  "customer_id": 33,
  "total": 902.5798,
  "lines": [
    {
      "id": 331237,
      "product_id": 420,
      "quantity": 1,
      "total_with_vat": 259.99
    },
    {
      "id": 331238,
      "product_id": 12,
      "quantity": 1,
      "total_with_vat": 642.5898
    }
  ]
}
```

Já a coleção products contém a junção de products com sub_category_product, sub_category e category, permitindo fácil acesso as suas informações relacionadas.



Para a coleção returns optou-se por usar uma extended reference de sales que contém o invoice_id relativo ao código da venda e o invoice_date relativo à data da venda. Isso tudo tendo em conta facilitar as consultas e o vocabulário exigido.



Os aggregate usados para todo o processo de modulação encontram-se no ficheiro “**Modulação.js**” enviado junto com o relatório.

A seguir encontra-se um exemplo de aggregate usado para o processo de modulação dos dados para as vendas:

```
db.sales_header.aggregate([
  {
    $lookup: {
      from: "sales_lines",
      localField: "invoice_id",
      foreignField: "invoice_id",
```

```

        as: "lines"
    }
},
{
    $unwind: "$lines"
},
{
    $group: {
        _id: "$invoice_id", //agrupar pelo invoice_id
        date: { $first: "$date" },
        customer_id: { $first: "$customer_id" },
        total: { $sum: "$lines.total_with_vat" }, //soma do total das
linhas de venda
        lines: { $push: "$lines" } //meter em array
    }
},
{
    $project: {
        _id: 0,
        "invoice_id": "$_id",
        "date": "$date",
        customer_id: "$customer_id",
        total: "$total",
        lines: {
            $map: {
                input: "$lines",
                as: "line",
                in: {
                    id: "$$line.id",
                    product_id: "$$line.product_id",

```

```

        quantity: "$$line.quantity",
        total_with_vat: "$$line.total_with_vat"
    }
}
}
}
},
{
    $out: "sales" // a nova coleção
}
])

```

4. Desenvolvimento de Consultas MongoDB:

A fase de desenvolvimento de consultas utilizando as funções `find()` e `aggregate()` é fundamental para a extração da informação necessária. Cada consulta é elaborada, considerando as características do modelo de dados no MongoDB desenvolvido e os requisitos específicos para a gerar os relatórios. A análise dos relacionamentos entre objetos nas coleções é crucial, assim como a aplicação de padrões para garantir a consistência e a performance do sistema.

Todas as consultas usadas encontram-se no ficheiro “**Consultas.js**” enviado junto com o relatório.

Exemplo de consulta usada para a lista dos clientes com compras em um janeiro de 2023:

```

db.sales.aggregate([
    {
        $match: {
            "date": {
                "$gte": ISODate("2023-01-01T00:00:00.000Z"),
                "$lt": ISODate("2023-02-01T00:00:00.000Z")
            }
        }
    }
])

```



```

    }
  },
  {
    $lookup: {
      from: "customer",
      localField: "customer_id",
      foreignField: "id",
      as: "customer"
    }
  },
  {
    $unwind: "$customer"
  },
  {
    $group: {
      _id: "$customer.id",
      id: { $first: "$customer.id" },
      first_name: { $first: "$customer.first_name" },
      last_name: { $first: "$customer.last_name" },
      email: { $first: { $ifNull: ["$customer.email",
"desconhecido"] } },
      address: {
        $first: {
          address: "$customer.address.address",
          city: "$customer.address.city",
          country: "$customer.address.country",
          postal_code: "$customer.address.postal_code"
        }
      },
      customer_type: { $first: "$customer.customer_type" },

```

```

        last3years: { $first: "$customer.last3years" }
    }
},
{
    $project: {
        _id: 0,
        "last3years.latest_purchase_date": 0
    }
},
{
    $sort: {
        id: 1
    }
}
]])

```

5. Implementação da API em BaseX:

A implementação da API em BaseX envolve a criação de dois recursos distintos: `sales_report` e `returns_report`. Ambos os recursos visam a obtenção de relatórios em XML para um mês e parceiro específicos, utilizando os pacotes XQuery e XMLDB do BaseX. Os recursos são projetados para atender aos requisitos específicos de relatórios de vendas e devoluções, incorporando a lógica do negócio necessária para fornecer informações relevantes.

A API em baseX refere-se ao documento “API.xq” enviado junto com o relatório.

6. Pedido HTTP à API do Mongo Atlas usando BaseX:

A última etapa do processo envolve a realização de pedidos HTTP à API do Mongo Atlas utilizando BaseX para transformar os dados conforme o vocabulário XML desenvolvido. Essa integração finaliza o ciclo de geração de relatórios, garantindo a entrega dos resultados desejados de maneira eficiente e compatível com os padrões estabelecidos.

Exemplo de pedido HTTP à DATA API usando a API do BaseX:

```
let $getSales := concat('{
  "dataSource": "PhoneForYouCluster",
  "database": "PhoneForYou",
  "collection": "sales",
  "pipeline": [
    {
      "$match": {
        "date": {
          "$gte": {"$date": "'", $startDate, '" },
          "$lt": {"$date": "'", $endDate, '" }
        }
      }
    },
    {
      "$project": {
        "_id": 0
      }
    },
    {
      "$sort": {
        "date": 1
      }
    }
  ]
}')

let $sales-response:= http:send-request(
  <http:request method='post'>
    <http:header name="api-key" value='{$api-key}'/>
    <http:body media-type='application/json'>{$getSales}</http:body>
```

```
</http:request>,  
$api-url      )
```

7. Funções em BaseX para Transformação em XML:

As funções implementadas em BaseX desempenham um papel fundamental na transformação dos dados em XML, seguindo o vocabulário XML desenvolvido para os relatórios. A função `create-date` é utilizada para criar datas no formato `xs:dateTime`, enquanto a função `validateXML` verifica a conformidade do XML com um esquema XSD. Além disso, as funções `transformCustomer`, `transformSale` e `transformProducts` são projetadas especificamente para transformar os dados do cliente, vendas e produtos em XML, garantindo a uniformidade nos dados obtidos.

Exemplo de função de transformação de dados do `product` em XML:

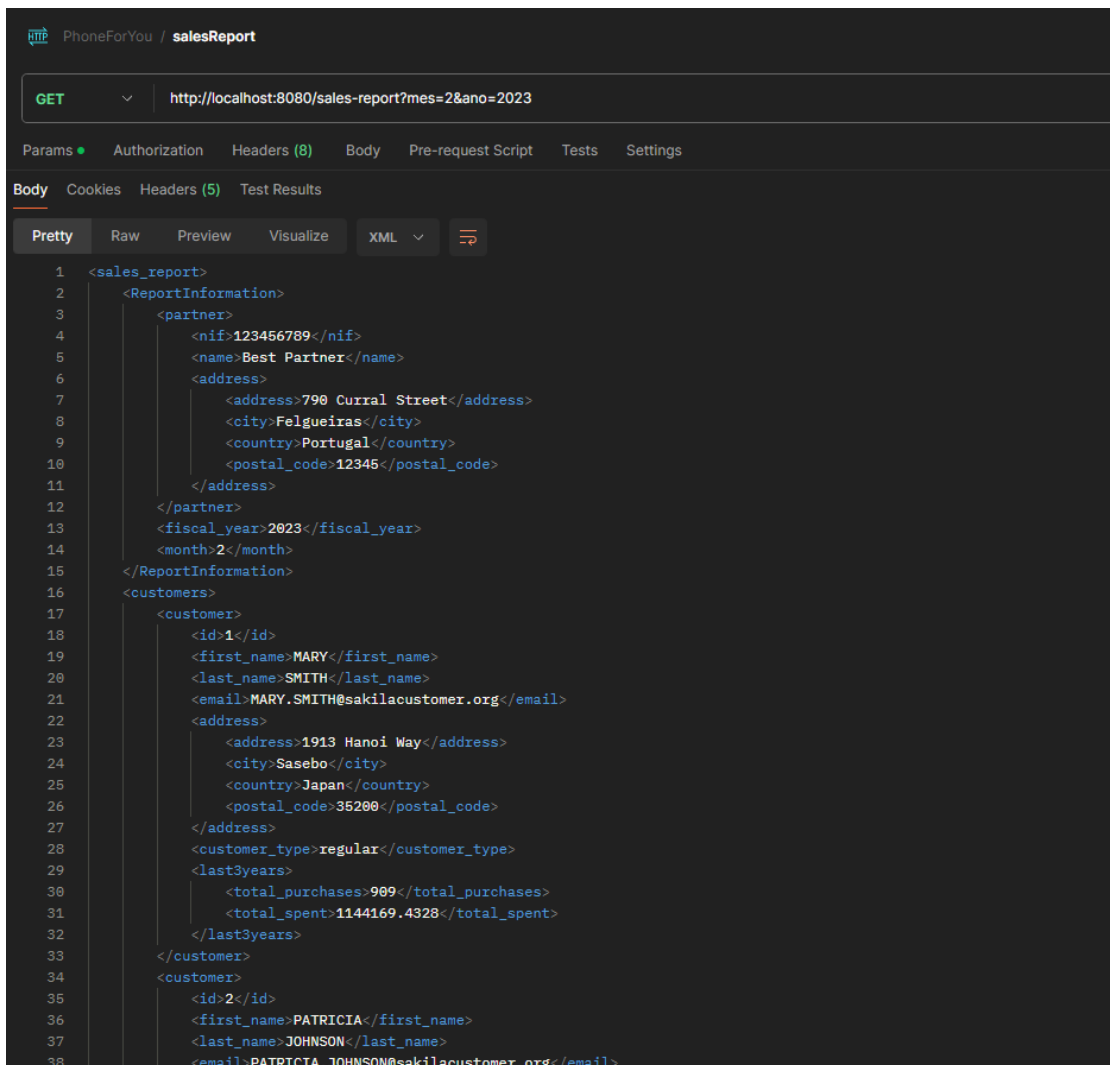
```
declare function page:transformProduct($product) {  
  element product {  
    element id { data($product/id) },  
    element brand { data($product/brand) },  
    element model { data($product/model) },  
    element price { data($product/price) },  
    element categories {  
      for $category in $product/categories/_  
      return  
        element { xs:QName(replace($category/category__name, ' ', '_')) }  
        {  
          data($category/sub__category__name)  
        }  
    }  
  }  
};
```

8. Postman para efetuar os requests ao BaseX

As endpoints criadas:

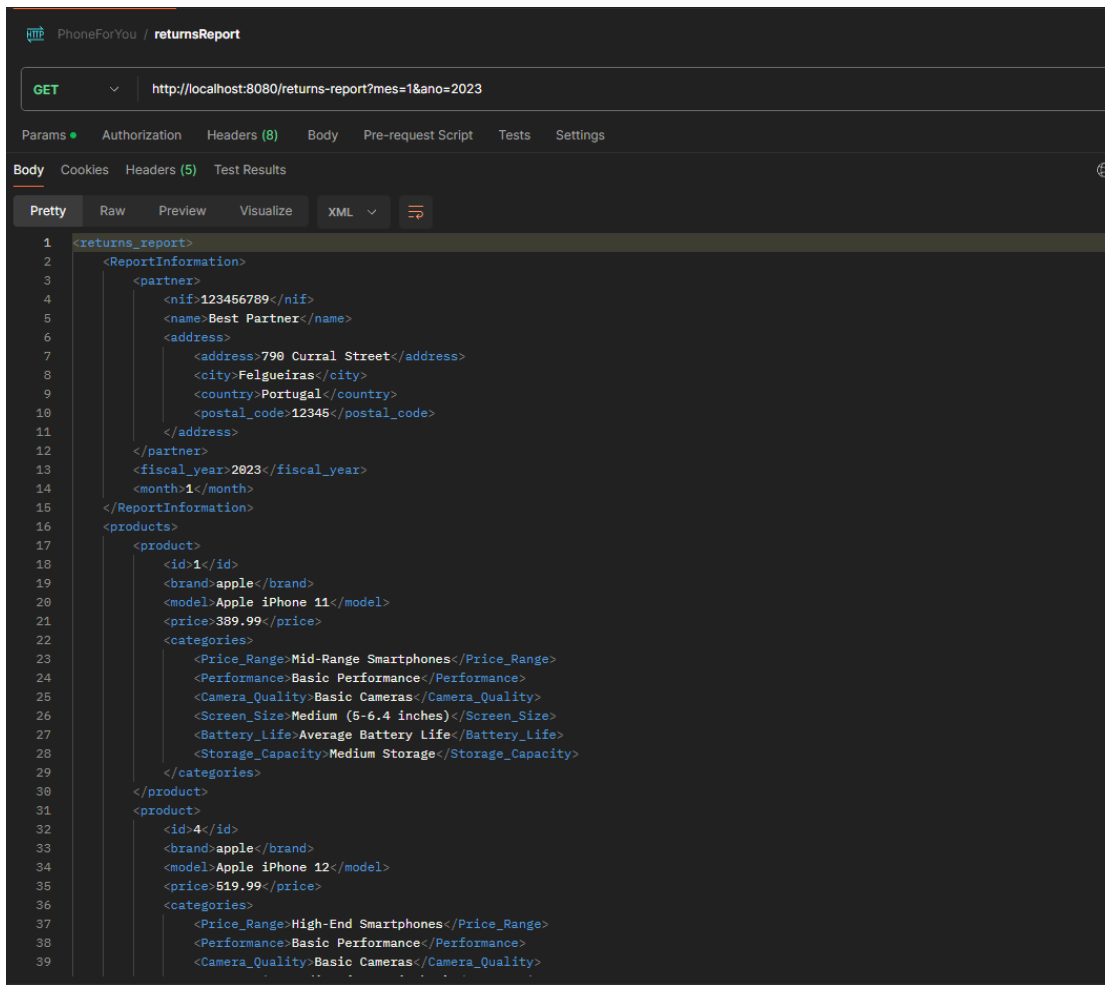
<http://localhost:8080/sales-report?mes=2&ano=2023>

<http://localhost:8080/returns-report?mes=2&ano=2023>



```
1 <sales_report>
2   <ReportInformation>
3     <partner>
4       <nif>123456789</nif>
5       <name>Best Partner</name>
6       <address>
7         <address>790 Curral Street</address>
8         <city>Felgueiras</city>
9         <country>Portugal</country>
10        <postal_code>12345</postal_code>
11      </address>
12    </partner>
13    <fiscal_year>2023</fiscal_year>
14    <month>2</month>
15  </ReportInformation>
16  <customers>
17    <customer>
18      <id>1</id>
19      <first_name>MARY</first_name>
20      <last_name>SMITH</last_name>
21      <email>MARY.SMITH@sakilacustomer.org</email>
22      <address>
23        <address>1913 Hanoi Way</address>
24        <city>Sasebo</city>
25        <country>Japan</country>
26        <postal_code>35200</postal_code>
27      </address>
28      <customer_type>regular</customer_type>
29      <last3years>
30        <total_purchases>909</total_purchases>
31        <total_spent>1144169.4328</total_spent>
32      </last3years>
33    </customer>
34    <customer>
35      <id>2</id>
36      <first_name>PATRICIA</first_name>
37      <last_name>JOHNSON</last_name>
38      <email>PATRICIA.JOHNSON@sakilacustomer.org</email>
```

Exemplo de pedido de relatório de vendas tendo em conta o vocabulário.



The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8080/returns-report?mes=1&ano=2023`
- Method: `GET`
- Body tab selected, showing XML response in "Pretty" format.

```
1 <returns_report>
2   <ReportInformation>
3     <partner>
4       <nif>123456789</nif>
5       <name>Best Partner</name>
6       <address>
7         <address>790 Curral Street</address>
8         <city>Felgueiras</city>
9         <country>Portugal</country>
10        <postal_code>12345</postal_code>
11      </address>
12    </partner>
13    <fiscal_year>2023</fiscal_year>
14    <month>1</month>
15  </ReportInformation>
16  <products>
17    <product>
18      <id>1</id>
19      <brand>apple</brand>
20      <model>Apple iPhone 11</model>
21      <price>389.99</price>
22      <categories>
23        <Price_Range>Mid-Range Smartphones</Price_Range>
24        <Performance>Basic Performance</Performance>
25        <Camera_Quality>Basic Cameras</Camera_Quality>
26        <Screen_Size>Medium (5-6.4 inches)</Screen_Size>
27        <Battery_life>Average Battery Life</Battery_life>
28        <Storage_Capacity>Medium Storage</Storage_Capacity>
29      </categories>
30    </product>
31    <product>
32      <id>4</id>
33      <brand>apple</brand>
34      <model>Apple iPhone 12</model>
35      <price>519.99</price>
36      <categories>
37        <Price_Range>High-End Smartphones</Price_Range>
38        <Performance>Basic Performance</Performance>
39        <Camera_Quality>Basic Cameras</Camera_Quality>
```

Exemplo de pedido de relatório de devoluções tendo em conta o vocabulário.

Conclusão

Em síntese, a elaboração de relatórios XML para registros de vendas e devoluções em um mês específico abrange várias fases, desde a importação inicial dos dados até sua conversão para o formato XML. Destaca-se a importância de selecionar tecnologias apropriadas e realizar a implementação eficiente de consultas e APIs. Cada etapa do procedimento contribui para uma solução sólida e escalável adaptada às necessidades específicas do projeto.