Engenharia Informática 23/24

# Processamento Estruturado de Informação

## Relatório


Realizado por:

Sónia Oliveira 8220114 LEI

João Santos 8220256 LEI

Ian Dinarte 8220005 LEI

## Introdução

Este relatório é realizado no contexto do desenvolvimento do trabalho prático para a cadeira de Processamento Estruturado de Informação, com o intuito de clarificar e justificar as decisões feitas ao longo da realização do projeto.

## Contextualização do domínio

A pesquisa realizada como suporte ao trabalho prático revelou que a apresentação de informação no formato XML é algo altamente flexível e conveniente. XML é um padrão que facilita a normalização de dados, além de facilitar a pesquisa em bases de dados com uma quantidade alargada de informação, através de tags criadas pelos desenvolvedores do vocabulário, com a possibilidade de utilizar a linguagem natural do domínio.

Como exemplo real,  estudámos o uso de ficheiros XML para o armazenamento de dados relacionados com faturas, nomeadamente, SAF-T-PT, *Standard Audit File for Tax Purposes – Portuguese version*. Refere-se a um ficheiro predefinido, em formato XML, que tem o objetivo de reunir e exportar toda a informação fiscal e contabilística de uma empresa, durante um determinado período. Este caso é muito semelhante ao contexto que nos foi apresentado, para o qual tínhamos de desenvolver uma API que desempenhasse a exportação de informação referente a um mês de vendas e devoluções de uma empresa.

Dado o facto que o uso do padrão  XML permite a criação de identificadores (tags) relevantes ao domínio, foi possível desenvolver um vocabulário que faz uso da linguagem natural, facilmente legível.

## Desenvolvimento do vocabulário

O primeiro passo do desenvolvimento deste projeto foi o desenvolvimento do vocabulário e normas a seguir para a exportação de informação em formato XML. Para tal, foram desenvolvidos ficheiros XSDSchema, que detalham quais os dados que serão armazenados, bem como o seu formato, e quaisquer restrições que possam ser aplicadas, no contexto.

## Ficheiros XSDSchema desenvolvidos

### GeneralInfo.xsd

```xml
<!-- GeneralInfo.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Common complex type for address information -->
  <xs:complexType name="Address">
    <xs:sequence>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
      <xs:element name="zipCode" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Complex type for person information -->
  <xs:complexType name="PersonInfo">
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="lastName" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Complex type for partner details -->
  <xs:complexType name="Partner">
    <xs:sequence>
      <xs:element name="partnerName" type="xs:string"/>
      <xs:element name="NIF" type="xs:string"/>
      <xs:element name="address" type="Address"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Complex type for general information -->
  <xs:complexType name="GeneralInfo">
    <xs:sequence>
      <xs:element name="partner" type="Partner"/>
      <xs:element name="fiscalYear" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Neste ficheiro, são definidas as normas para a informação relacionada a parceiros, moradas e pessoas. São informações gerais que serão utilizadas noutros ficheiros.

### CustomerDetails.xsd

```
<!-- CustomerDetails.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:include schemaLocation="GeneralInfo.xsd"/>

  <!-- Complex type for customer details -->
  <xs:complexType name="Customer">
    <xs:complexContent>
      <xs:extension base="PersonInfo">
        <xs:sequence>
          <xs:element name="email" type="xs:string" minOccurs="0" maxOccurs="1" default="Unknown"/>
          <xs:element name="address" type="Address"/>
          <xs:element name="customerType" type="customerType"/>
          <xs:element name="purchaseInfo" type="purchaseInfoType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Complex type for customer type -->
  <xs:simpleType name="customerType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="new"/>
      <xs:enumeration value="regular"/>
      <xs:enumeration value="premium"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Complex type for purchase information -->
  <xs:complexType name="purchaseInfoType">
    <xs:sequence>
      <xs:element name="numPurchases" type="xs:integer"/>
      <xs:element name="totalPurchaseValue" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Este ficheiro representa a informação a ser armazenada acerca de um cliente, o que inclui o primeiro e último nome, email (se o cliente não forneceu o email, este surgirá como “Unknown”), a morada (país, cidade e código postal, como foi detalhado em GeneralInfo.xsd) e ainda o tipo de cliente com base no tempo

que se encontra registado no sistema. Também foi definido um tipo de dados destinado a guardar informação relativa ao histórico de vendas de um cliente, como o número de compras (nos últimos 3 anos, de acordo com o contexto) e o valor total destas.

### ProductDetails.xsd

```
<!-- ProductDetails.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Complex type for product details -->
  <xs:complexType name="Product">
    <xs:sequence>
      <xs:element name="Code" type="xs:string"/>
      <xs:element name="Brand" type="xs:string"/>
      <xs:element name="Model" type="xs:string"/>
      <xs:element name="Price" type="xs:decimal"/>
      <xs:element name="PriceCategory" type="PriceCategoryType"/>
      <xs:element name="PerformanceCategory" type="PerformanceCategoryType"/>
      <xs:element name="CameraQuality" type="CameraQualityType"/>
      <xs:element name="ScreenSize" type="ScreenSizeType"/>
      <xs:element name="BatteryLife" type="BatteryLifeType"/>
      <xs:element name="StorageSpace" type="StorageSizeType"/>
    </xs:sequence>
  </xs:complexType>
```

Neste ficheiro, é definida a estrutura para a exportação de dados referentes a um produto, bem como diversos tipos de dados para as várias categorias de qualidade de um produto, (Gama de Preços, Desempenho, Qualidade da Câmera Frontal, Tamanho de Ecrã, Capacidade de Bateria e Capacidade de armazenamento).

```
<!-- Complex type for product category -->
<xs:simpleType name="PriceCategoryType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Budget"/>
    <xs:enumeration value="Mid-Range"/>
    <xs:enumeration value="High-End"/>
  </xs:restriction>
</xs:simpleType>

  <!-- Complex type for performance category -->
<xs:simpleType name="PerformanceCategoryType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Basic"/>
    <xs:enumeration value="Standard"/>
    <xs:enumeration value="High"/>
    <xs:enumeration value="Gaming"/>
  </xs:restriction>
</xs:simpleType>

    <!-- Complex type for camera category -->
<xs:simpleType name="CameraQualityType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Basic"/>
    <xs:enumeration value="Good"/>
    <xs:enumeration value="Pro"/>
  </xs:restriction>
</xs:simpleType>
```

```
    <!-- Complex type for screen category -->
<xs:simpleType name="ScreenSizeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Small"/>
    <xs:enumeration value="Medium"/>
    <xs:enumeration value="Large"/>
  </xs:restriction>
</xs:simpleType>

      <!-- Complex type for battery category -->
<xs:simpleType name="BatteryLifeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Short"/>
    <xs:enumeration value="Average"/>
    <xs:enumeration value="Long"/>
  </xs:restriction>
</xs:simpleType>

        <!-- Complex type for storage category -->
<xs:simpleType name="StorageSizeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Low"/>
    <xs:enumeration value="Medium"/>
    <xs:enumeration value="High"/>
  </xs:restriction>
</xs:simpleType>
```

Estes tipos foram criados aderindo às normas do contexto, tendo em conta as subcategorias contidas em cada categoria de qualidade.

### SaleDetails.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:include schemaLocation="ProductDetails.xsd"/>

  <!-- Define a complex type for a sale -->
  <xs:complexType name="Sale">
    <xs:sequence>
      <xs:element name="InvoiceCode" type="xs:string"/>
      <xs:element name="SaleDate" type="xs:date"/>
      <xs:element name="CustomerCode" type="xs:string"/>
      <xs:element name="TotalSaleValue" type="xs:decimal"/>
      <xs:element name="LineItems" type="LineItemList"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Define a complex type for a list of line items -->
  <xs:complexType name="LineItemList">
    <xs:sequence>
      <xs:element name="LineItem" type="LineItem" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Define a complex type for a line item -->
  <xs:complexType name="LineItem">
    <xs:sequence>
      <xs:element name="LineNumber" type="xs:integer"/>
      <xs:element name="ProductCode" type="xs:string"/>
      <xs:element name="Quantity" type="xs:integer"/>
      <xs:element name="TotalLineValue" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
```

Neste ficheiro estão definidas as regras para os detalhes de uma venda, bem como a linha de venda. Além disso, é definida uma lista para o número de vendas por subcategoria de produto. Apesar do contexto ser vago, presumiu-se que o objetivo era apresentar, por categoria, o número de vendas em cada subcategoria que lhe compete.

```
<!-- Define a complex type for a list of sales by category -->
<xs:complexType name="SalesByCategoryList">
  <xs:sequence>
    <xs:element name="SalesByPrice" type="SalesByPrice" minOccurs="0" maxOccurs="1"/>
    <xs:element name = "SalesByPerformance" type="SalesByPerformance" minOccurs="0" maxOccurs="1"/>
    <xs:element name = "SalesByCamera" type="SalesByCamera" minOccurs="0" maxOccurs="1"/>
    <xs:element name = "SalesByScreenSize" type="SalesByScreenSize" minOccurs="0" maxOccurs="1"/>
    <xs:element name = "SalesByBattery" type="SalesByBattery" minOccurs="0" maxOccurs="1"/>
    <xs:element name = "SalesByStorage" type="SalesByStorage" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="SalesByPrice">
  <xs:sequence>
    <xs:element name="Budget" type="xs:integer"/>
    <xs:element name="MidRange" type="xs:integer"/>
    <xs:element name="HighEnd" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>
```

```
 <!--  SalesByCategory for PerformanceCategory -->

<xs:complexType name="SalesByPerformance">
    <xs:sequence>
      <xs:element name="Basic" type="xs:integer"/>
      <xs:element name="Sandard" type="xs:integer"/>
      <xs:element name="High" type="xs:integer"/>
      <xs:element name="Gaming" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
```

```
 <!--  SalesByCategory for Camera quality -->

  <xs:complexType name= "SalesByCamera">
    <xs:sequence>
      <xs:element name="Basic" type="xs:integer"/>
      <xs:element name="Good" type="xs:integer"/>
      <xs:element name="Pro" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
```

```
 <!--  SalesByCategory for screen quality -->

 <xs:complexType name="SalesByScreenSize">
   <xs:sequence>
     <xs:element name="Small" type="xs:integer"/>
     <xs:element name="Medium" type="xs:integer"/>
     <xs:element name="Large" type="xs:integer"/>
   </xs:sequence>
 </xs:complexType>
```

```
 <!--  SalesByCategory for battery quality -->

 <xs:complexType name="SalesByBattery">
   <xs:sequence>
     <xs:element name="Short" type="xs:integer"/>
     <xs:element name="Average" type="xs:integer"/>
     <xs:element name="Long" type="xs:integer"/>
   </xs:sequence>
 </xs:complexType>
```

```
<!--  SalesByCategory for storage quality -->

  <xs:complexType name="SalesByStorage" >
    <xs:sequence>
      <xs:element name="Low" type="xs:integer"/>
      <xs:element name="Medium" type="xs:integer"/>
      <xs:element name="High" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
```

## ReturnDetails.xsd

```
<!-- ReturnDetails.xsd -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:include schemaLocation="ProductDetails.xsd"/>
  <xs:include schemaLocation="GeneralInfo.xsd"/>

 <!-- Define a complex type for summary information on returns -->
  <xs:complexType name="ReturnSummary">
    <xs:sequence>
      <xs:element name="NumberOfDistinctProductsReturned" type="xs:integer"/>
      <!-- Add elements for other summary information -->
      <xs:element name="ReturnsByCategory" type="ReturnsByCategoryList"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Define a complex type for a list of sales by category -->
  <xs:complexType name="ReturnsByCategoryList">
    <xs:sequence>
      <xs:element name="ReturnsByPrice" type="ReturnsByPrice" minOccurs="0" maxOccurs="1"/>
      <xs:element name = "ReturnsByPerformance" type="ReturnsByPerformance" minOccurs="0" maxOccurs="1"/>
      <xs:element name = "ReturnsByCamera" type="ReturnsByCamera" minOccurs="0" maxOccurs="1"/>
      <xs:element name = "ReturnsByScreenSize" type="ReturnsByScreenSize" minOccurs="0" maxOccurs="1"/>
      <xs:element name = "ReturnsByBattery" type="ReturnsByBattery" minOccurs="0" maxOccurs="1"/>
      <xs:element name = "ReturnsByStorage" type="ReturnsByStorage" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
```

```
<!--  SalesByCategory for PriceCategory -->

<xs:complexType name="ReturnsByPrice">
  <xs:sequence>
    <xs:element name="Budget" type="xs:integer"/>
    <xs:element name="MidRange" type="xs:integer"/>
    <xs:element name="HighEnd" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>


<!--  SalesByCategory for PerformanceCategory -->

<xs:complexType name="ReturnsByPerformance">
  <xs:sequence>
    <xs:element name="Basic" type="xs:integer"/>
    <xs:element name="Sandard" type="xs:integer"/>
    <xs:element name="High" type="xs:integer"/>
    <xs:element name="Gaming" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

<!--  SalesByCategory for Camera quality -->

<xs:complexType name= "ReturnsByCamera">
  <xs:sequence>
    <xs:element name="Basic" type="xs:integer"/>
    <xs:element name="Good" type="xs:integer"/>
    <xs:element name="Pro" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>
```

```
<!--  SalesByCategory for screen quality -->

<xs:complexType name="ReturnsByScreenSize">
  <xs:sequence>
    <xs:element name="Small" type="xs:integer"/>
    <xs:element name="Medium" type="xs:integer"/>
    <xs:element name="Large" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

<!--  SalesByCategory for battery quality -->

<xs:complexType name="ReturnsByBattery">
  <xs:sequence>
    <xs:element name="Short" type="xs:integer"/>
    <xs:element name="Average" type="xs:integer"/>
    <xs:element name="Long" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

<!--  SalesByCategory for storage quality -->

<xs:complexType name="ReturnsByStorage" >
  <xs:sequence>
    <xs:element name="Low" type="xs:integer"/>
    <xs:element name="Medium" type="xs:integer"/>
    <xs:element name="High" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```

Neste ficheiro é definida a estrutura para armazenar informação relativa às devoluções, nomeadamente, o número de produtos devolvidos, e as devoluções por categoria de produto.

### MonthlyReturnsReport.xsd

```xml
<!-- MonthlyReturnsReport.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Include necessary schemas -->
  <xs:include schemaLocation="GeneralInfo.xsd"/>
  <xs:include schemaLocation="ProductDetails.xsd"/>
  <xs:include schemaLocation="ReturnDetails.xsd"/>

  <!-- Define the main element for Monthly Returns Report -->
  <xs:element name="MonthlyReturnsReport">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="generalInfo" type="GeneralInfo"/>
        <xs:element name="returnsInfo" type="ReturnList"/>
        <xs:element name="returnsSummary" type="ReturnSummary"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>


  <!-- Define a complex type for a return -->
  <xs:complexType name="Return">
    <xs:sequence>
      <xs:element name="ReturnInvoiceCode" type="xs:string"/>
      <xs:element name="ReturnDate" type="xs:date"/>
      <xs:element name="PartnerInfo" type="Partner"/>
      <xs:element name="ProductDetails" type="ProductList"/>
      <!-- Additional elements for return-specific information -->
      <xs:element name="DaysToReturn" type="xs:integer"/>
      <xs:element name="EarlyReturn" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
```

Aqui são definidas as regras para a informação de uma devolução, bem como a informação de todas as devoluções necessárias (em contexto, de um certo mês), além da lista de produtos devolvidos.

```
  <xs:complexType name="ProductList">
    <xs:sequence>
      <xs:element name="Product" type="Product" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>


  <!-- Define a complex type for a list of returns -->
  <xs:complexType name="ReturnList">
    <xs:sequence>
      <xs:element name="Return" type="Return" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

## MonthlySalesReport.xsd

```
<!--MonthlySalesReport.xsd: main file -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:include schemaLocation="GeneralInfo.xsd"/>
  <xs:include schemaLocation="CustomerDetails.xsd"/>
  <xs:include schemaLocation="ProductDetails.xsd"/>
  <xs:include schemaLocation="SaleDetails.xsd"/>
  <xs:include schemaLocation="ReturnDetails.xsd"/>

  <xs:element name="MonthlySalesReport">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="generalInfo" type="GeneralInfo"/>
        <xs:element name="customerInfo" type="Customer" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="ListOfProducts" type="ProductList"/>
        <xs:element name="ListOfSales" type="SalesList"/>
        <xs:element name="salesInfo" type="SalesList"/>
        <!-- Add summarized information elements here -->
        <xs:element name="NumberOfDistinctProducts" type="xs:integer"/>
        <xs:element name="TotalSales" type="xs:decimal"/>
        <xs:element name="NumberOfDistinctCustomers" type="xs:integer"/>
        <xs:element name="SalesByCategory" type="SalesByCategoryList"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Neste ficheiro está definida a estrutura de um relatório de vendas mensal, bem como a informação sumarizada, a lista de vendas  e produtos.

```
<!-- Define a complex type for a list of products -->
<xs:complexType name="ProductList">
  <xs:sequence>
    <xs:element name="Product" type="Product" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<!-- Define a complex type for a list of sales -->
<xs:complexType name="SalesList">
  <xs:sequence>
    <xs:element name="Sale" type="Sale" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

**Organização de ficheiros CSV na plataforma MongoDB**

Os ficheiros disponibilizados para a realização do trabalho possuem referências entre si, sendo desse modo interligados. Por isso, foi necessário adotar uma estratégia de organização que permitisse o acesso à informação da forma mais eficaz possível.

Assim, foram criadas várias coleções, uma por cada ficheiro, inicialmente, e de seguida, realizadas pesquisas e agregações de modo a fazer o melhor uso das referências presentes.

Começou-se por tratar da informação referente a clientes, primeiro guardando a morada inteira (cidade e país, que antes estavam separados) dentro da coleção que continha dados de moradas, e de seguida embutindo estes dados dentro da coleção de clientes.  Foram também realizados os cálculos necessários para classificar um cliente como regular, novo ou premium.

- Para isto, foi criado um índice para o id da morada

Quanto aos produtos, cada subcategoria passou a conter a categoria correspondente, e depois foram associadas aos respectivos produtos, através da informação no ficheiro que continha a informação da relação produto - subcategoria.

- Criaram-se índices para os id 's das categorias e subcategorias, bem como dos produtos.

Em relação às vendas,  a informação foi armazenada numa coleção à parte, que contém cada cabeçalho com a respectiva linha de venda. Além disso, contém os produtos e clientes associados a cada venda.

- Foram criados índices para os id's de clientes, produtos, e ainda para as faturas (invoice_id)

Quanto às devoluções, a estratégia foi semelhante às vendas, com a exceção de não ter sido necessário utilizar uma coleção à parte, tendo ficado tudo armazenado na coleção inicial de devoluções. Aqui,  também foi adicionado um campo de modo a classificar a devolução como precoce ou não.

- Os índices criados correspondem ao código de faturas, e produtos.

**Desenvolvimento da API**

A API desenvolvida define um esqueleto para que a informação possa ser recuperada usando o Postman como interlocutor. Desta forma, apenas é necessário indicar o mês e ano, e a API realizará as restantes operações para retornar informação, sem que seja necessário indicar qualquer corpo no Postman.

Esta abordagem permitiu maior flexibilidade nos resultados, e também faz mais sentido, para que seja possível a qualquer um retirar informação sem necessariamente ter conhecimento de como desenvolver vocabulário em XML.

Apesar de a tarefa ter sido de um modo geral bem desenvolvida, houve informação que o grupo não conseguiu incluir nos relatórios: o número de vendas e devoluções por subcategoria de produto. No entanto, toda a informação relevante foi, de resto, esquematizada e recuperada com sucesso.

**Conclusão**

A realização do trabalho permitiu sem dúvida um alto nível de familiarização com as ferramentas fornecidas. No entanto, de um modo geral, o grupo não se sentia preparado para a quantidade de dados a processar e informação a ter em conta, além de termos sentido falta de conhecimento em muitas alturas. Para o BaseX, especificamente, foi muito difícil encontrar recursos e exemplos que ajudassem no desenvolvimento do trabalho, levando a várias incertezas.

Apesar das várias dificuldades encontradas, o grupo foi capaz de desenvolver o projeto o mais completo possível, dentro das nossas capacidades, e a opinião geral é de que foi obtido um razoável nível de conhecimento quanto aos conteúdos lecionados na unidade curricular.