P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Licenciatura em Engenharia Informática

Ano letivo 2018/2019

Processamento estruturado de informação

Trabalho Prático 1

Grupo: 6

Carlos Barros, nº 8160081

João Moreira, nº 8160163

Vitor Barbosa, nº 8160277

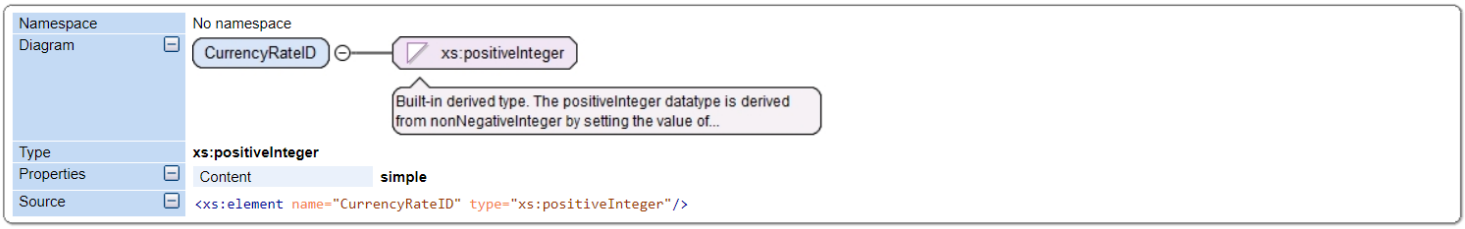
Índice

Requisitos de vocabulário

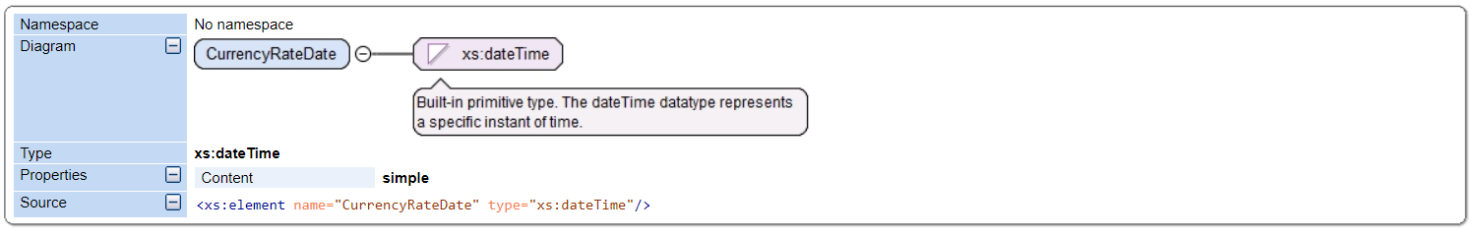
➢ Currency Details

Os requisitos que levantamos para o Currency Details através do csv disponibilizado são os seguintes:

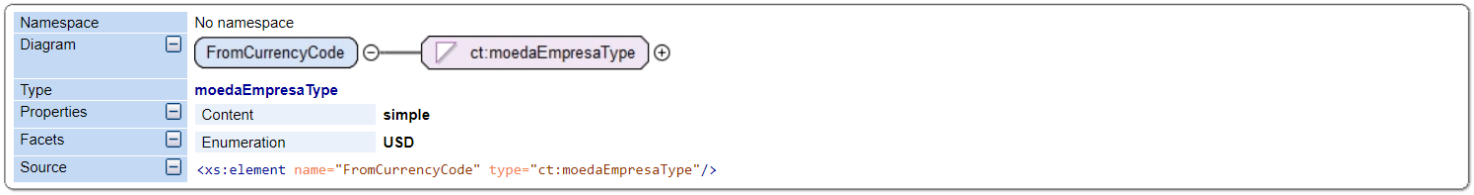
- CurrencyRateID: sendo este a nosso ver do tipo inteiro positivo pois o ID é sempre um número inteiro positivo.



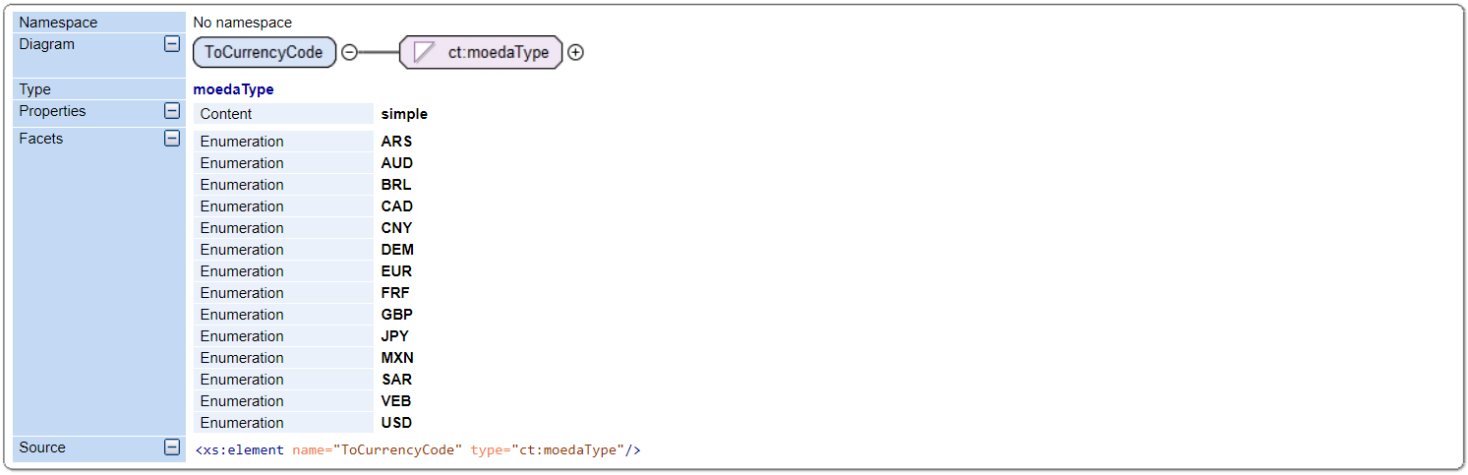
- CurrencyRateDate: colocamos este do tipo dateTime para desta forma conseguir guardar tanto a data como a hora.



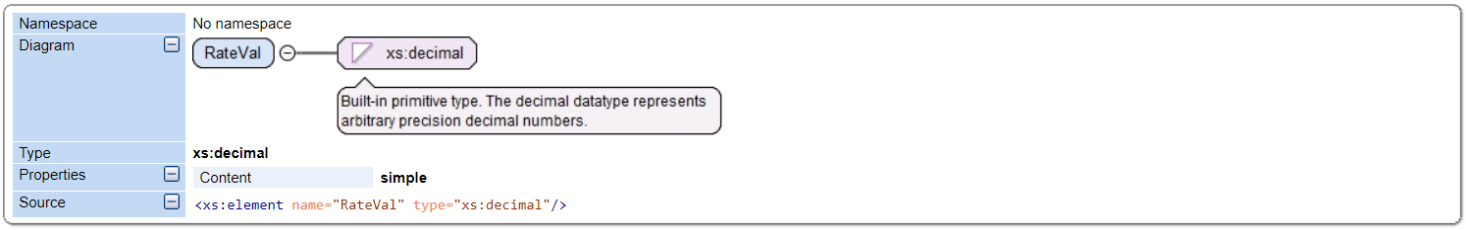
- FromCurrencyCode: criamos para este um tipo novo designado moedaEmpresaType sendo o mesmo uma string que neste momento apenas aceita “USD”. Desta forma se no futuro a empresa decidir começar a utilizar outra moeda diferente ou mais do que uma moeda bastará ser alterado esse tipo. Este novo tipo criado está guardado no ficheiro CommonTypes.xsd.



- ToCurrencyCode: também para este criamos um tipo com todas as moedas utilizadas nas conversões limitando assim as moedas que podem ser convertidas. Este tipo também está guardado no ficheiro CommonTypes.xsd.



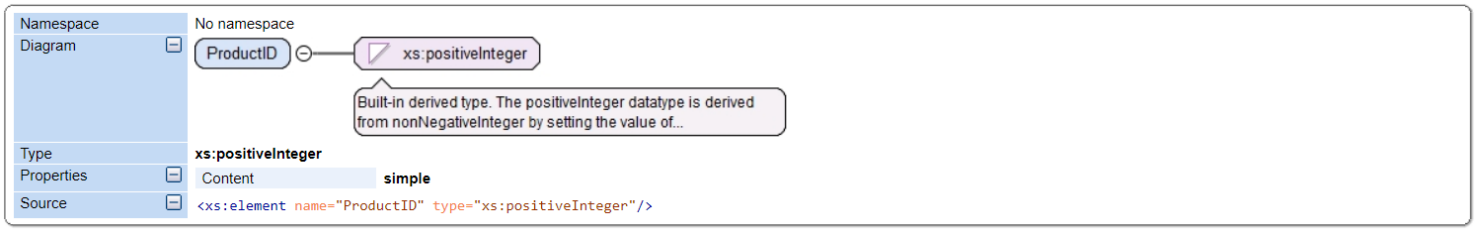
- RateVal: utilizamos para este o tipo decimal pois é o mais acertado para o valor obtido da conversão. Não está limitado pois poderão existir no futuro grandes alterações a nível da taxa de conversões.



➢ Product Details

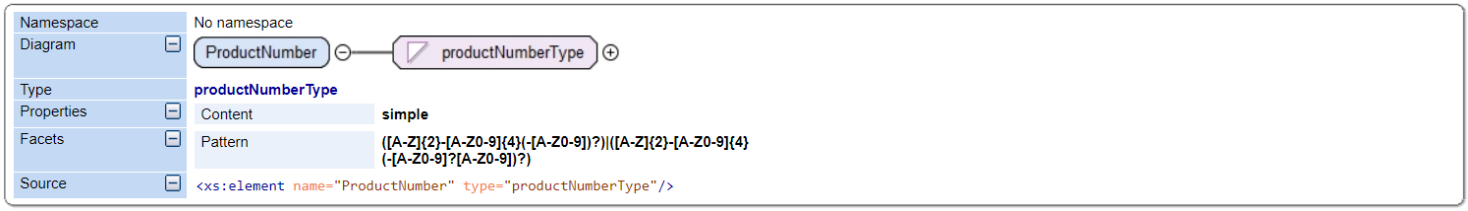
Os requisitos que levantamos para o Product Details através do csv disponibilizado são os seguintes:

- ProductID: sendo este a nosso ver um inteiro positivo pois o id do produto é sempre um número inteiro positivo.



- Name: para este achamos melhor criar um tipo chamado nameType que funciona como uma string que se encontra limitada a quarenta caracteres para desta forma limitar o tamanho do nome.

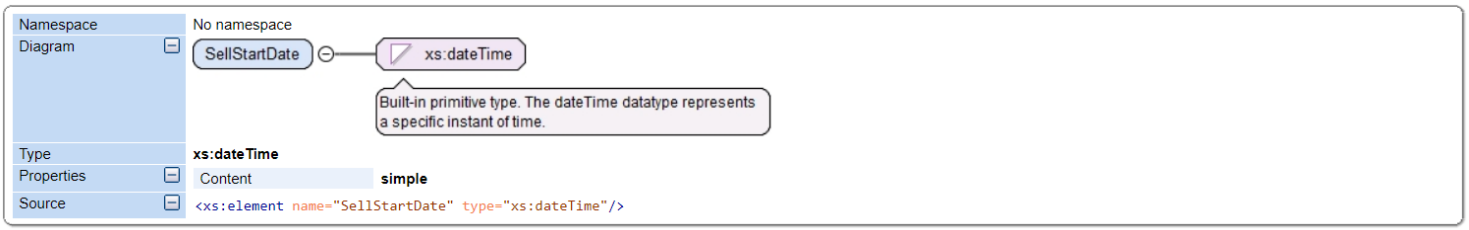
- ProductNumberType: criamos para este um tipo de seu nome productNumberType sendo por base uma string mas alterada de forma a que terá de ter dois caracteres maiúsculos no início seguido de um traço (“-“) tendo em seguida quatro caracteres que pode ser de A-Z ou de 0-9 sendo esta parte obrigatória. Acrescentamos ainda a seguir a esta parte obrigatória poderá ser seguida por uma parte opcional constituída por um traço (“-“) e um ou dois caracteres.



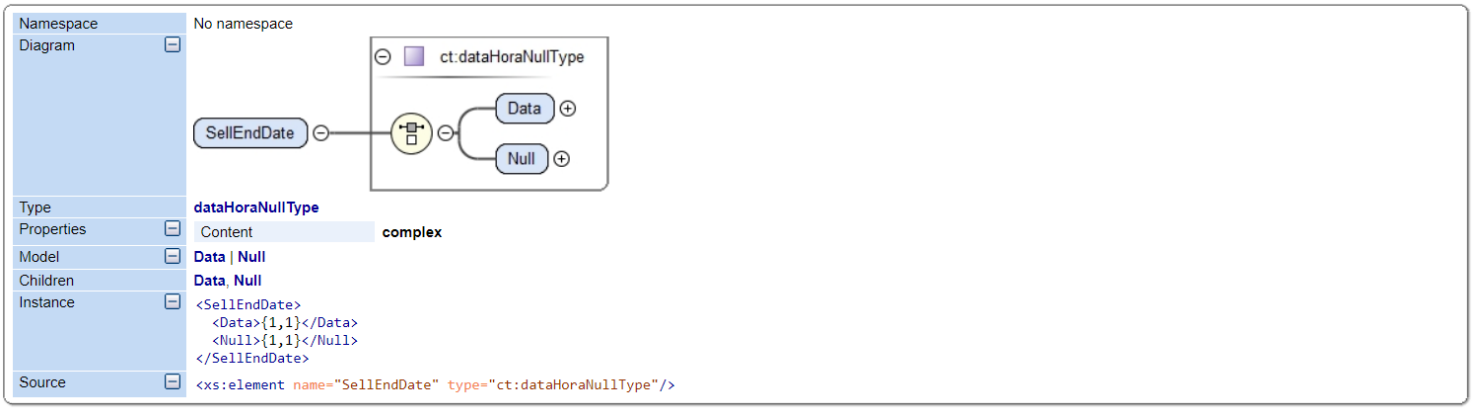
- Color: decidimos criar para este um tipo colorType que tem por base uma string limitada com diversas cores disponíveis onde caso não tenha assume NULL e se tiver mais do que uma assume Multi.

- ListPrice: criamos para este um tipo designado listPriceType que é do tipo decimal, mas restringimos o número de casas decimas a dois.

- SellStartDate: colocamos este do tipo dateTime para desta forma conseguir guardar tanto a data como a hora.



- SellEndDate: decidimos para este criar um tipo chamado dataHoraNullType que se encontra no ficheiro commonTypes que restringe a data a dateTime ou a NULL.

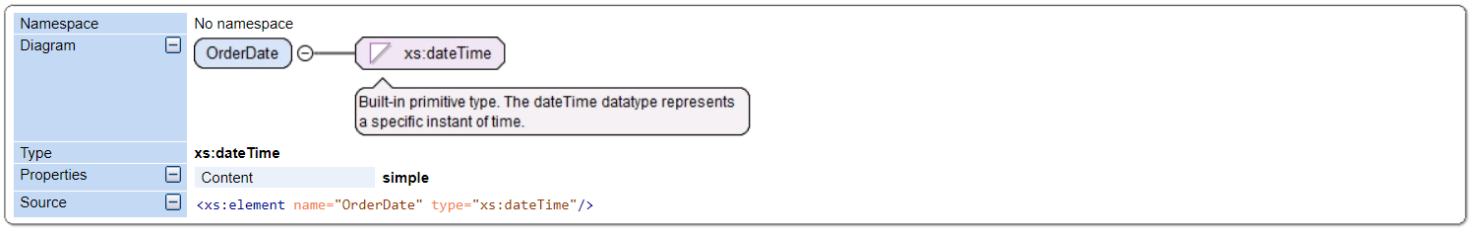


➢ Sales Details

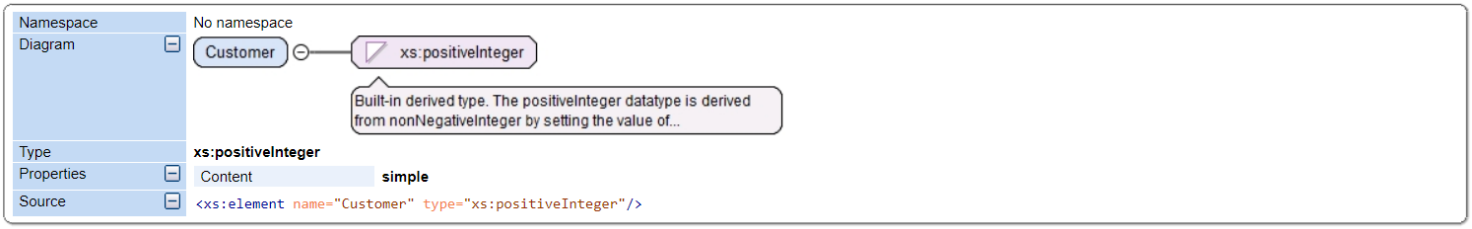
Os requisitos que levantamos para o Sales Details através do csv disponibilizado são os seguintes:

- ReceiptID: tivemos a ideia de criar para este, um tipo com o nome receiptIDType que é um número inteiro positivo com no máximo cinco dígitos para desta forma podermos limitar o mesmo.

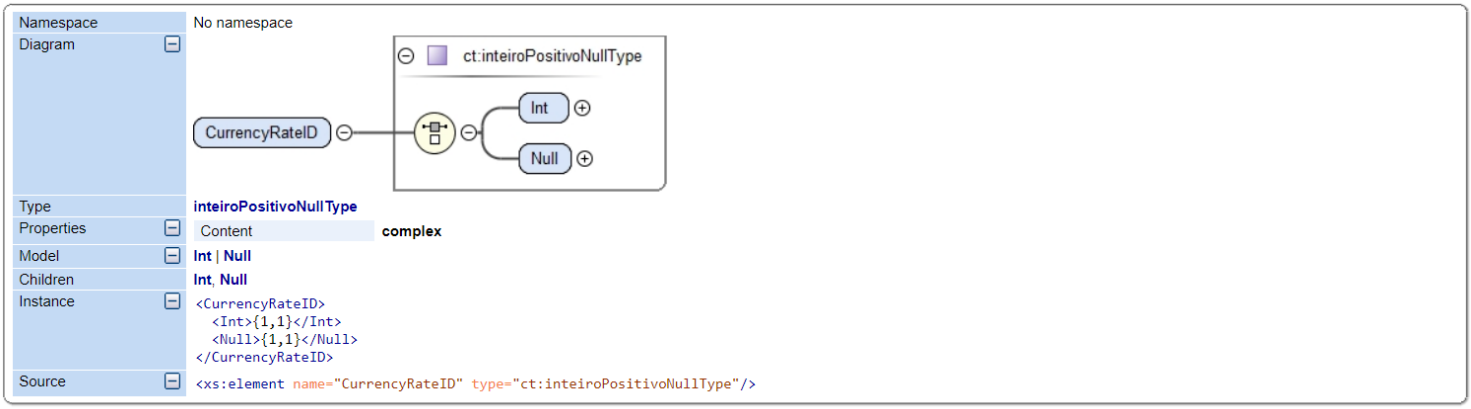
- OrderDate: colocamos este do tipo dateTime para desta forma conseguir guardar tanto a data como a hora.



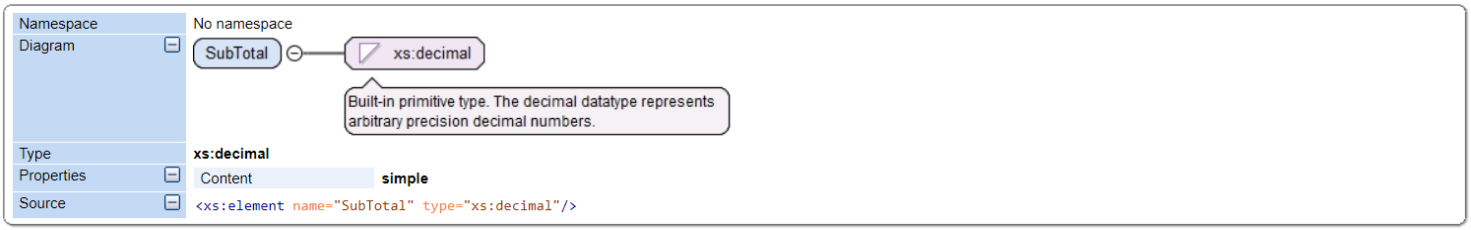
- Customer: utilizamos para este o tipo positiveInteger pois como o número do cliente é um número inteiro positivo, este tipo é o que mais se adequa nesta situação. Não limitamos o número de casa decimais pois a empresa pode aumentar consideravelmente o seu número de clientes.



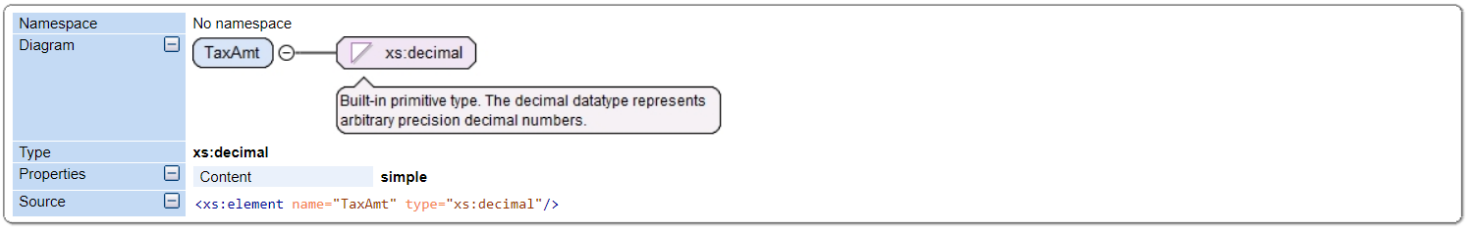
- CurrencyRateID: decidimos criar para este o tipo inteiroPositivoNullType podendo este ser um inteiro positivo ou NULL.



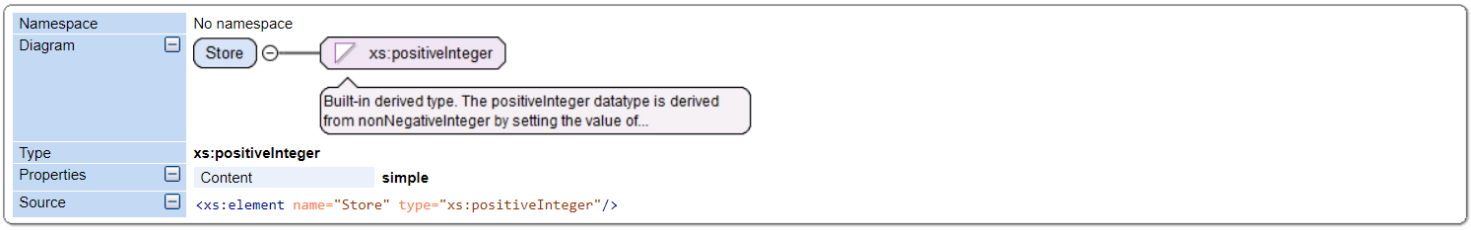
- SubTotal: achamos que o tipo mais adequado para a informação nos dada será o decimal.



- TaxAmt: utilizamos para este o tipo decimal pois era o que se enquadrava melhor com a situação.

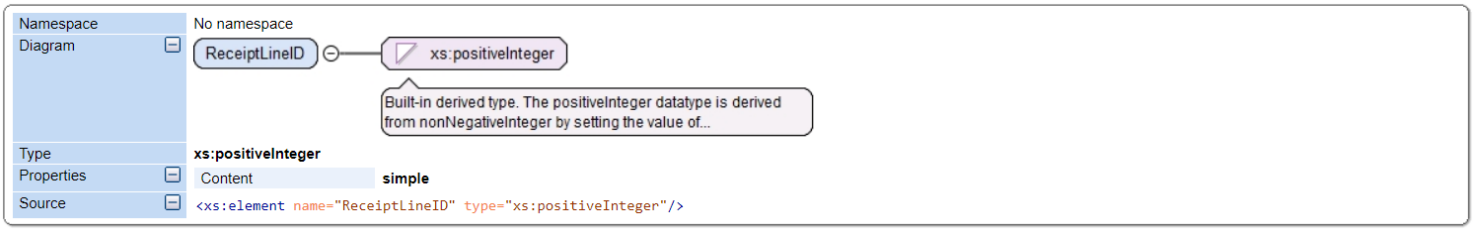


- Store: decidimos utilizar para este o tipo positiveInteger pois são sempre números inteiros positivos não podendo limitar.

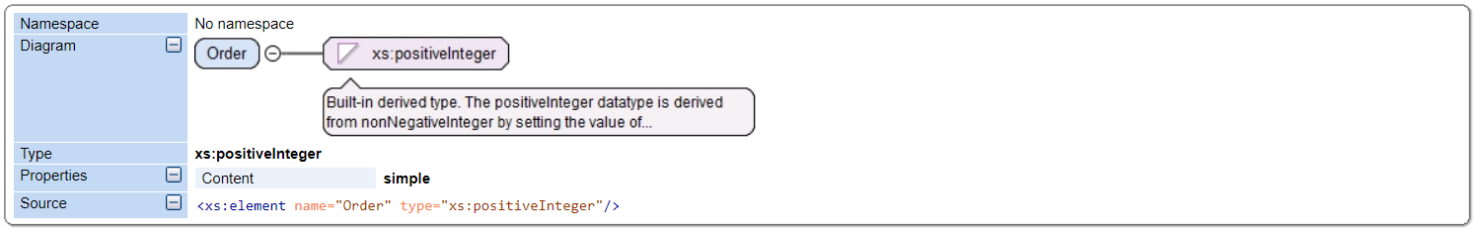


- StoreName: para este achamos melhor criar um tipo chamado nameType que funciona como uma string que se encontra limitada a quarenta caracteres para desta forma limitar o tamanho do nome.

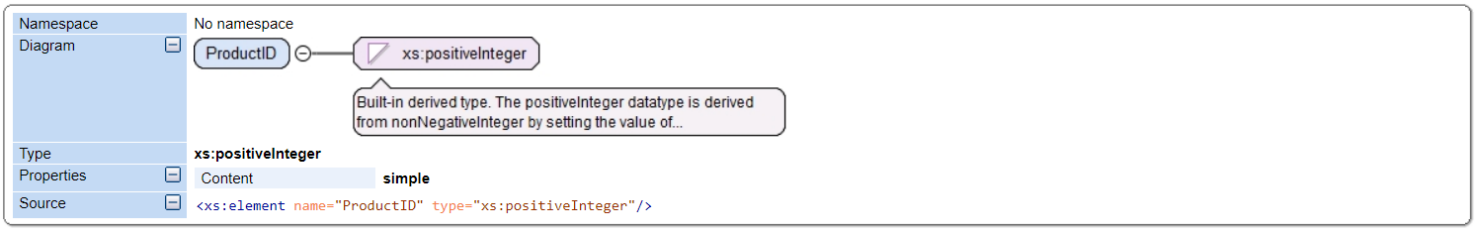
- ReceiptLineID: utilizamos para este o tipo positiveInteger pois são sempre números inteiros positivos não podendo limitar.



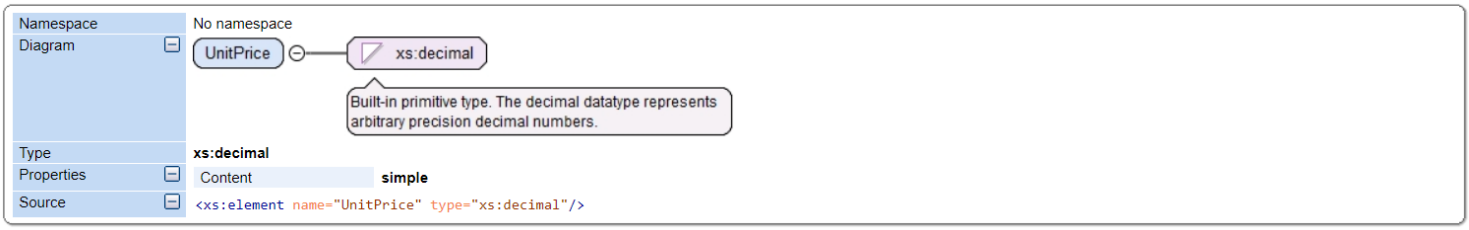
- Order: achamos melhor utilizar para este o tipo positiveInteger pois são sempre números inteiros positivos não podendo limitar porque não sabemos a proporção máxima da quantidade.



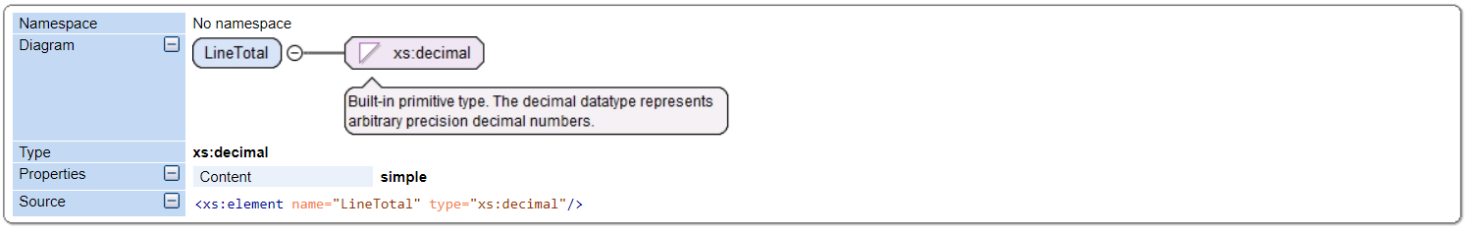
- ProductID: utilizamos para este o tipo positiveInteger pois são sempre números inteiros positivos não podendo limitar.



- UnitPrice: para este achamos que o tipo mais adequado para a informação nos dada será o decimal.



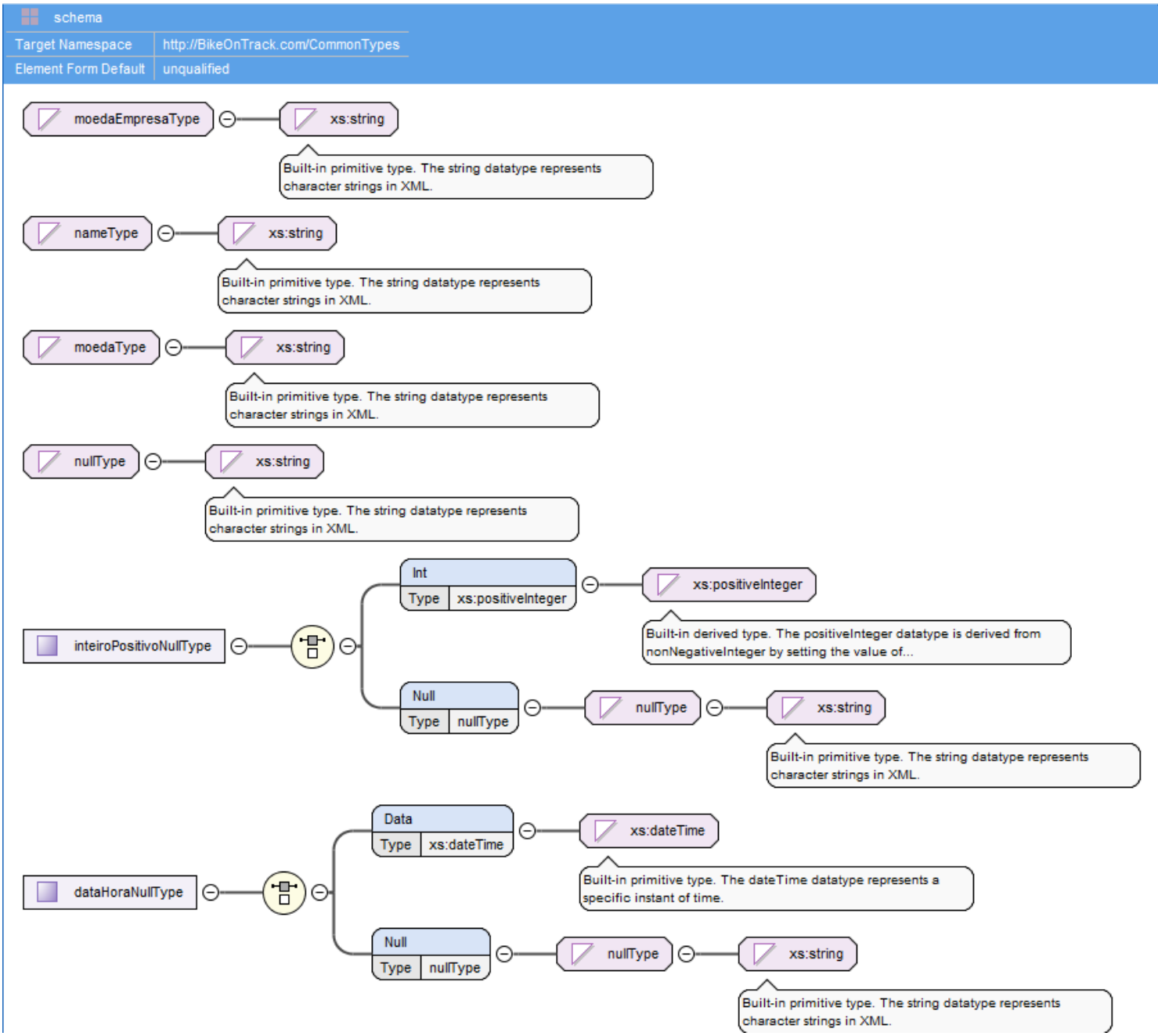
- LineTotal: achamos por bem utilizar o tipo decimal.



Identificação de schemas

✓ CommonTypes

- NameSpace: http://BikeOnTrack.com/CommonTypes.
- Propriedades: elementFormDefault="unqualified".
- Localização: CommonTypes.xsd
- Descrição: Elemento contendo os tipos comuns entre todos os restantes schemas.
- Descrição visual:



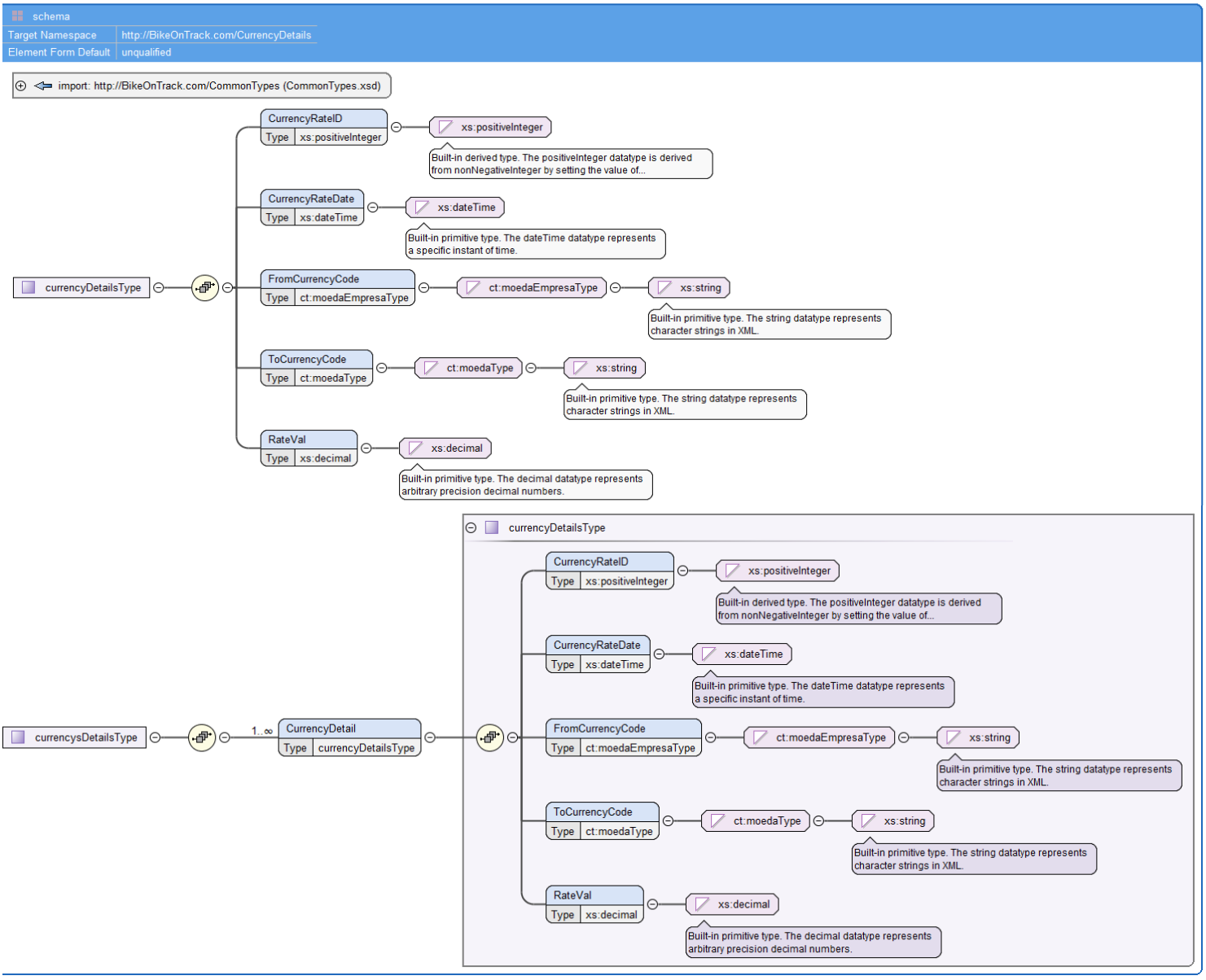
Estrutura XSD:

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://BikeOnTrack.com/CommonTypes"
           targetNamespace="http://BikeOnTrack.com/CommonTypes"
           elementFormDefault="unqualified">

    <!-- Definição do tipo de moeda da empresa  -->
    <xs:simpleType name="moedaEmpresaType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="USD"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição de um tipo string com no maximo  -->
    <xs:simpleType name="nameType">
        <xs:restriction base="xs:string">
            <xs:maxLength value="40"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição do tipo de moeda da empresa  -->
    <xs:simpleType name="moedaType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ARS"/>
            <xs:enumeration value="AUD"/>
            <xs:enumeration value="BRL"/>
            <xs:enumeration value="CAD"/>
            <xs:enumeration value="CNY"/>
            <xs:enumeration value="DEM"/>
            <xs:enumeration value="EUR"/>
            <xs:enumeration value="FRF"/>
            <xs:enumeration value="GBP"/>
            <xs:enumeration value="JPY"/>
            <xs:enumeration value="MXN"/>
            <xs:enumeration value="SAR"/>
            <xs:enumeration value="VEB"/>
            <xs:enumeration value="USD"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <!-- Definição do tipo NULL que e umua string  -->
    <xs:simpleType name="nullType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NULL"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição do tipo inteiro positivo ou Null  -->
    <xs:complexType name="inteiroPositivoNullType">
        <xs:choice>
            <xs:element name="Int" type="xs:positiveInteger"/>
            <xs:element name="Null" type="nullType"/>
        </xs:choice>
    </xs:complexType>

    <!-- Definição do tipo data ou Null  -->
    <xs:complexType name="dataHoraNullType">
        <xs:choice>
            <xs:element name="Data" type="xs:dateTime"/>
            <xs:element name="Null" type="nullType"/>
        </xs:choice>
    </xs:complexType>

</xs:schema>
```

✓ CurrencyDetails

- NameSpace: http://BikeOnTrack.com/CurrencyDetails.
- Propriedades: elementFormDefault="unqualified".
- Localização: CurrencyDetails.xsd
- Descrição: Elemento contendo as informações relativas aos detalhes da moeda usada.
- Descrição visual:



Estrutura XSD:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ct="http://BikeOnTrack.com/CommonTypes"
           xmlns="http://BikeOnTrack.com/CurrencyDetails"
           targetNamespace="http://BikeOnTrack.com/CurrencyDetails"
           elementFormDefault="unqualified">


    <!-- Importar XSD secundário -->
    <xs:import schemaLocation="CommonTypes.xsd" namespace="http://BikeOnTrack.com/CommonTypes"/>

    <!-- Definição de currencyDetailsType -->
    <xs:complexType name="currencyDetailsType">
        <xs:sequence>
            <xs:element name="CurrencyRateID" type="xs:positiveInteger"/>
            <xs:element name="CurrencyRateDate" type="xs:dateTime"/>
            <xs:element name="FromCurrencyCode" type="ct:moedaEmpresaType"/>
            <xs:element name="ToCurrencyCode" type="ct:moedaType"/>
            <xs:element name="RateVal" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Unidade tem de ter no mínimo um caso de currencyDetails -->
     <xs:complexType name="currencysDetailsType">
        <xs:sequence>
            <xs:element name="CurrencyDetail" type="currencyDetailsType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Exemplo XML:

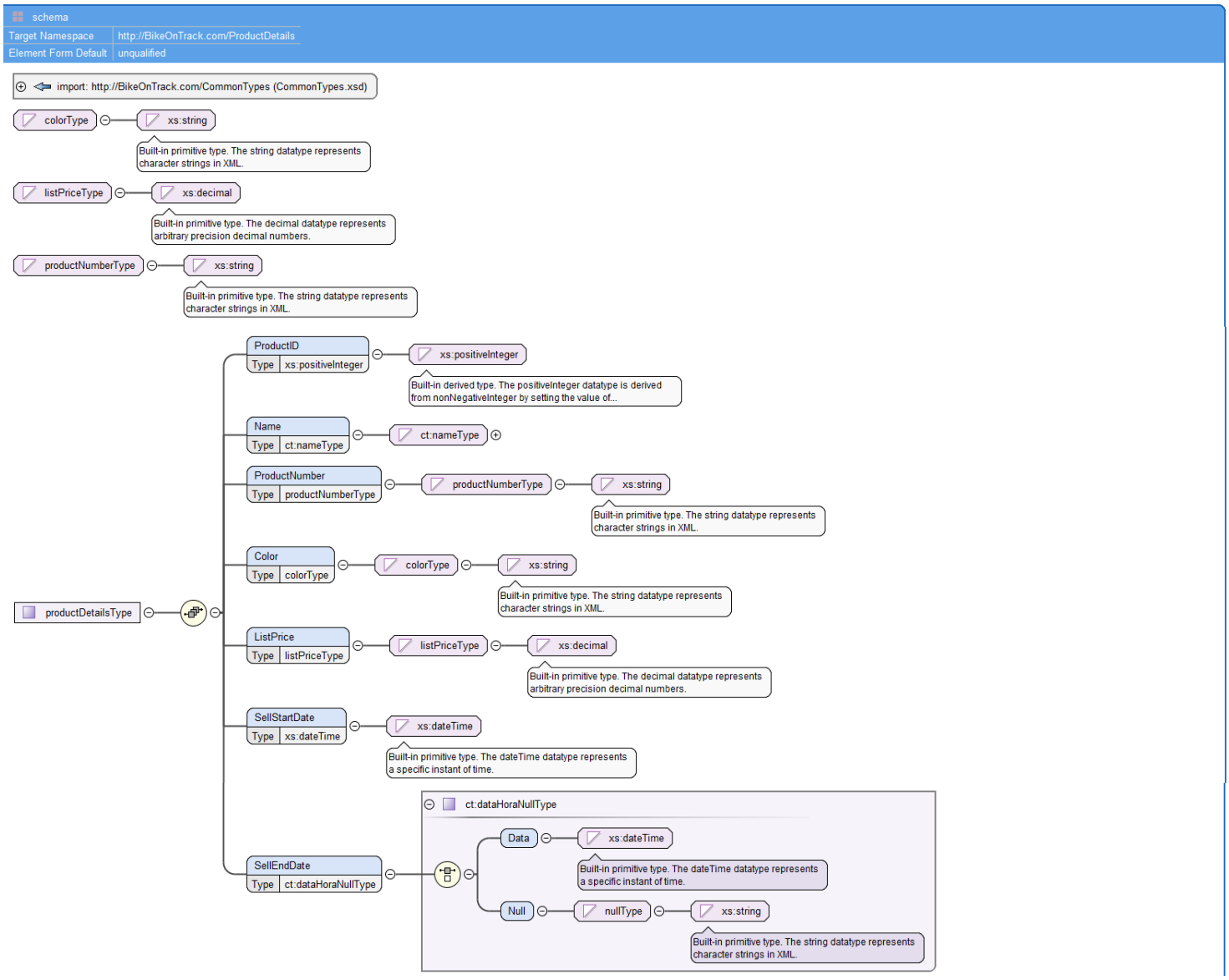
```xml
<!-- CurrencysDetails -->
<CurrencysDetails>
 <!-- Primeiro exemplo de currencyDetail -->
 <CurrencyDetail>
     <CurrencyRateID>1</CurrencyRateID>
     <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
     <FromCurrencyCode>USD</FromCurrencyCode>
     <ToCurrencyCode>ARS</ToCurrencyCode>
     <RateVal>1.0002</RateVal>
 </CurrencyDetail>
 </CurrencysDetails>
```

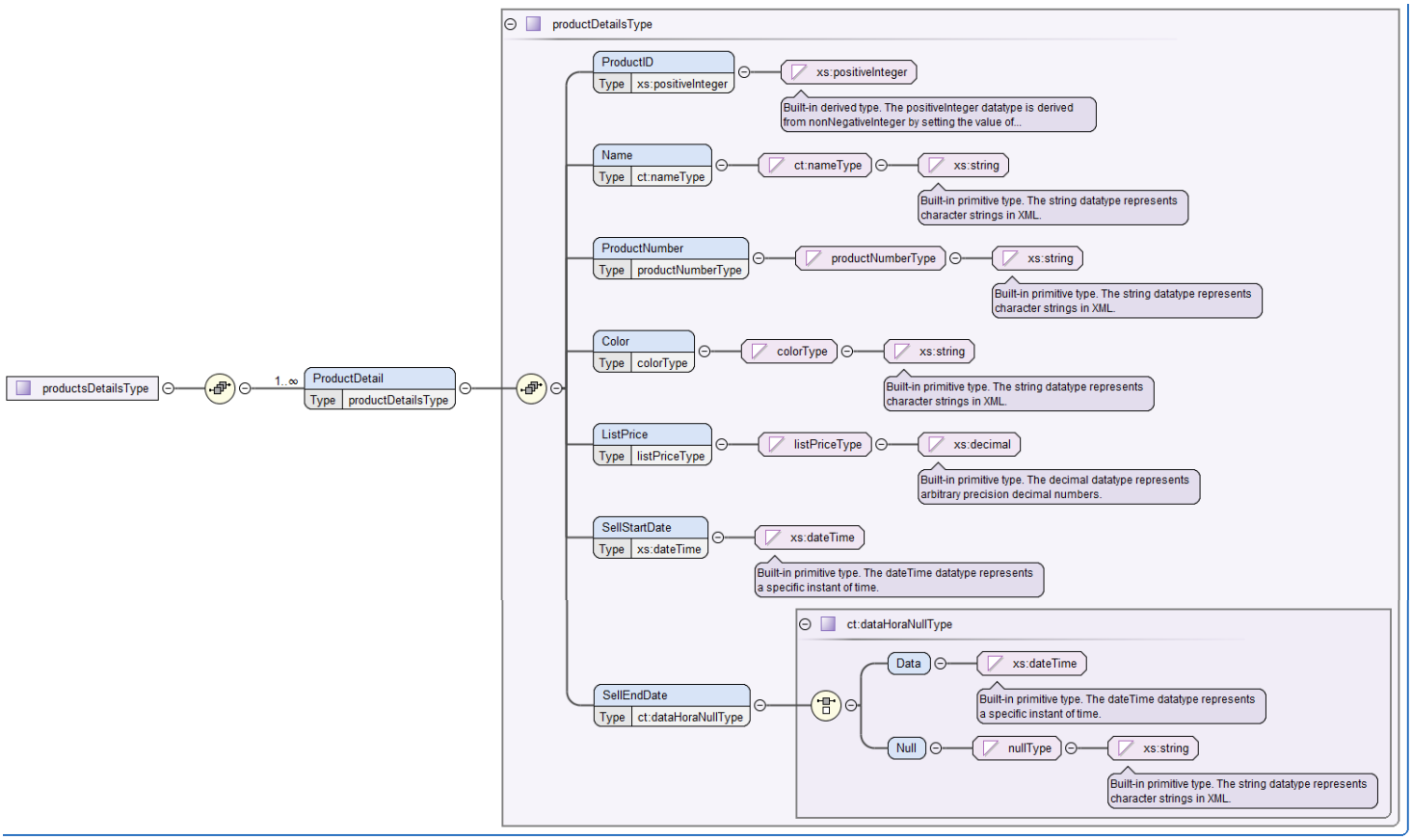
Dependências:

- Tipo: ct:CommonTypes
- NameSpace: http://BikeOnTrack.com/CommonTypes.
- Localização: CommomTypes.xsd

✓ ProductsDetails

- NameSpace: http://BikeOnTrack.com/ProductDetails.
- Propriedades: elementFormDefault="unqualified".
- Localização: ProductsDetails.xsd
- Descrição: Elemento contendo as informações relativas aos detalhes do produto.
- Descrição visual:




productsDetailsType
1..∞
ProductDetail
Type productDetailsType
productDetailsType
ProductID
Type xs:positiveInteger
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
Name
Type ct:nameType
ct:nameType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
ProductNumber
Type productNumberType
productNumberType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
Color
Type colorType
colorType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
ListPrice
Type listPriceType
listPriceType
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
SellStartDate
Type xs:dateTime
xs:dateTime
Built-in primitive type. The dateTime datatype represents a specific instant of time.
SellEndDate
Type ct:dataHoraNullType
ct:dataHoraNullType
Data
xs:dateTime
Built-in primitive type. The dateTime datatype represents a specific instant of time.
Null
nullType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.

Estrutura XSD:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ct="http://BikeOnTrack.com/CommonTypes"
           xmlns="http://BikeOnTrack.com/ProductDetails"
           targetNamespace="http://BikeOnTrack.com/ProductDetails"
           elementFormDefault="unqualified">

    <!-- Importar XSD secundário -->
    <xs:import schemaLocation="CommonTypes.xsd" namespace="http://BikeOnTrack.com/CommonTypes"/>

    <!-- Tipos de cores disponíveis -->
    <xs:simpleType name="colorType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Black"/>
            <xs:enumeration value="Silver"/>
            <xs:enumeration value="Red"/>
            <xs:enumeration value="White"/>
            <xs:enumeration value="Multi"/>
            <xs:enumeration value="Blue"/>
            <xs:enumeration value="Yellow"/>
            <xs:enumeration value="Green"/>
            <xs:enumeration value="Silver/Black"/>
            <xs:enumeration value="Grey"/>
            <xs:enumeration value="NULL"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição de listPriceType com 2 casas decimais -->
    <xs:simpleType name="listPriceType">
        <xs:restriction base="xs:decimal">
            <xs:fractionDigits value="2"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição de productNumberType -->
    <xs:simpleType name="productNumberType">
        <xs:restriction base="xs:string">
            <xs:pattern value ="[A-Z]{2}-[A-Z0-9]{4}(-[A-Z0-9])?"/>
            <xs:pattern value ="[A-Z]{2}-[A-Z0-9]{4}(-[A-Z0-9]?[A-Z0-9])?"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição de productDetailsType -->
    <xs:complexType name="productDetailsType">
        <xs:sequence>
            <xs:element name="ProductID" type="xs:positiveInteger"/>
            <xs:element name="Name" type="ct:nameType"/>
            <xs:element name="ProductNumber" type="productNumberType"/>
            <xs:element name="Color" type="colorType"/>
            <xs:element name="ListPrice" type="listPriceType"/>
            <xs:element name="SellStartDate" type="xs:dateTime"/>
            <xs:element name="SellEndDate" type="ct:dataHoraNullType"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Unidade tem de ter no mínimo um caso de currencyDetails -->
    <xs:complexType name="productsDetailsType">
        <xs:sequence>
            <xs:element name="ProductDetail" type="productDetailsType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Exemplo XML:

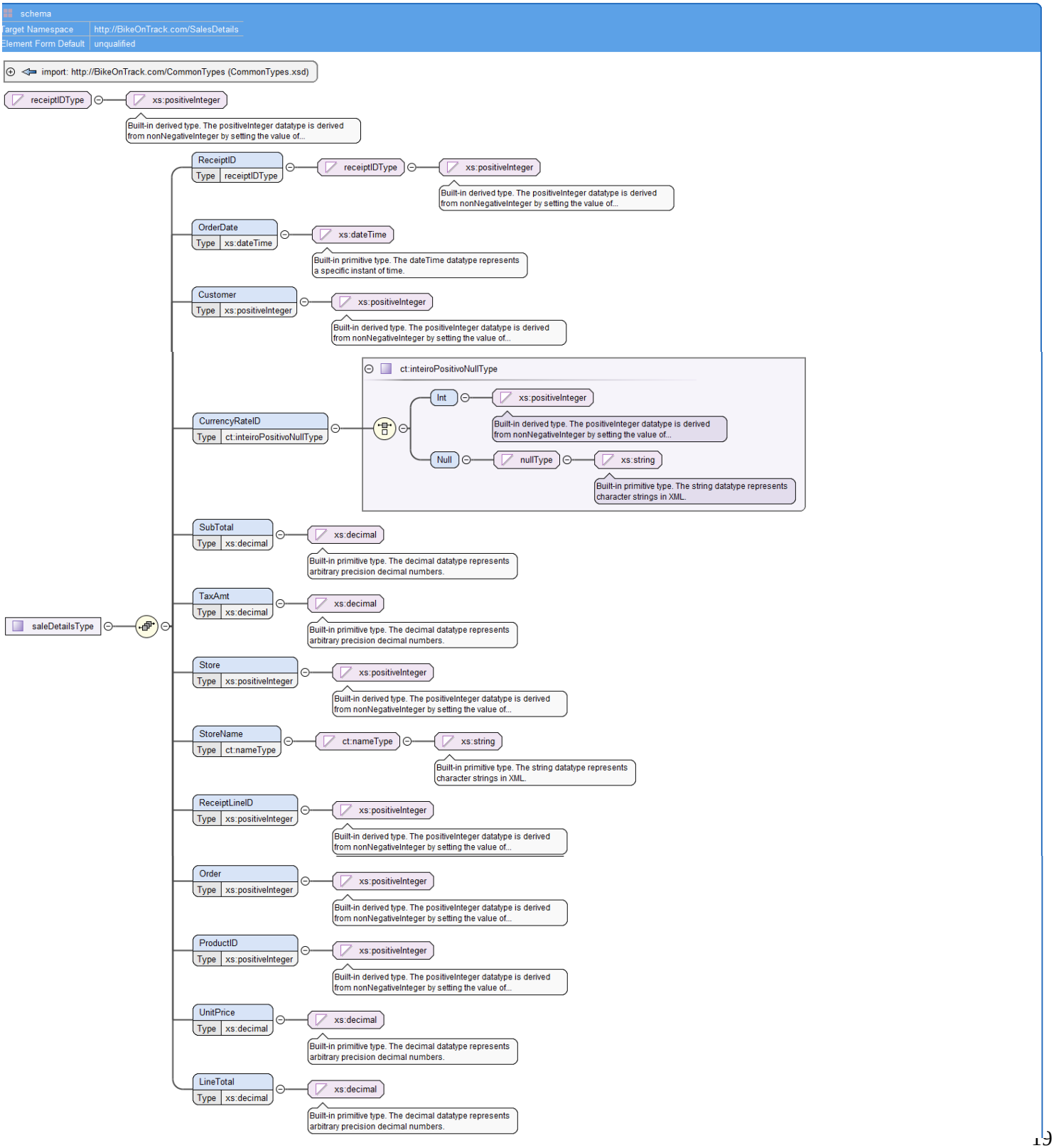
```
<!-- ProductsDetails -->
<ProductsDetails>
  <!-- Primeiro exemplo de ProductDetail -->
  <ProductDetail>
      <ProductID>1</ProductID>
      <Name>Adjustable Race</Name>
      <ProductNumber>AR-5381</ProductNumber>
      <Color>NULL</Color>
      <ListPrice>0</ListPrice>
      <SellStartDate>2008-04-30T00:00:00</SellStartDate>
      <SellEndDate>
          <Null>NULL</Null>
      </SellEndDate>
  </ProductDetail>
  </ProductsDetails>
```

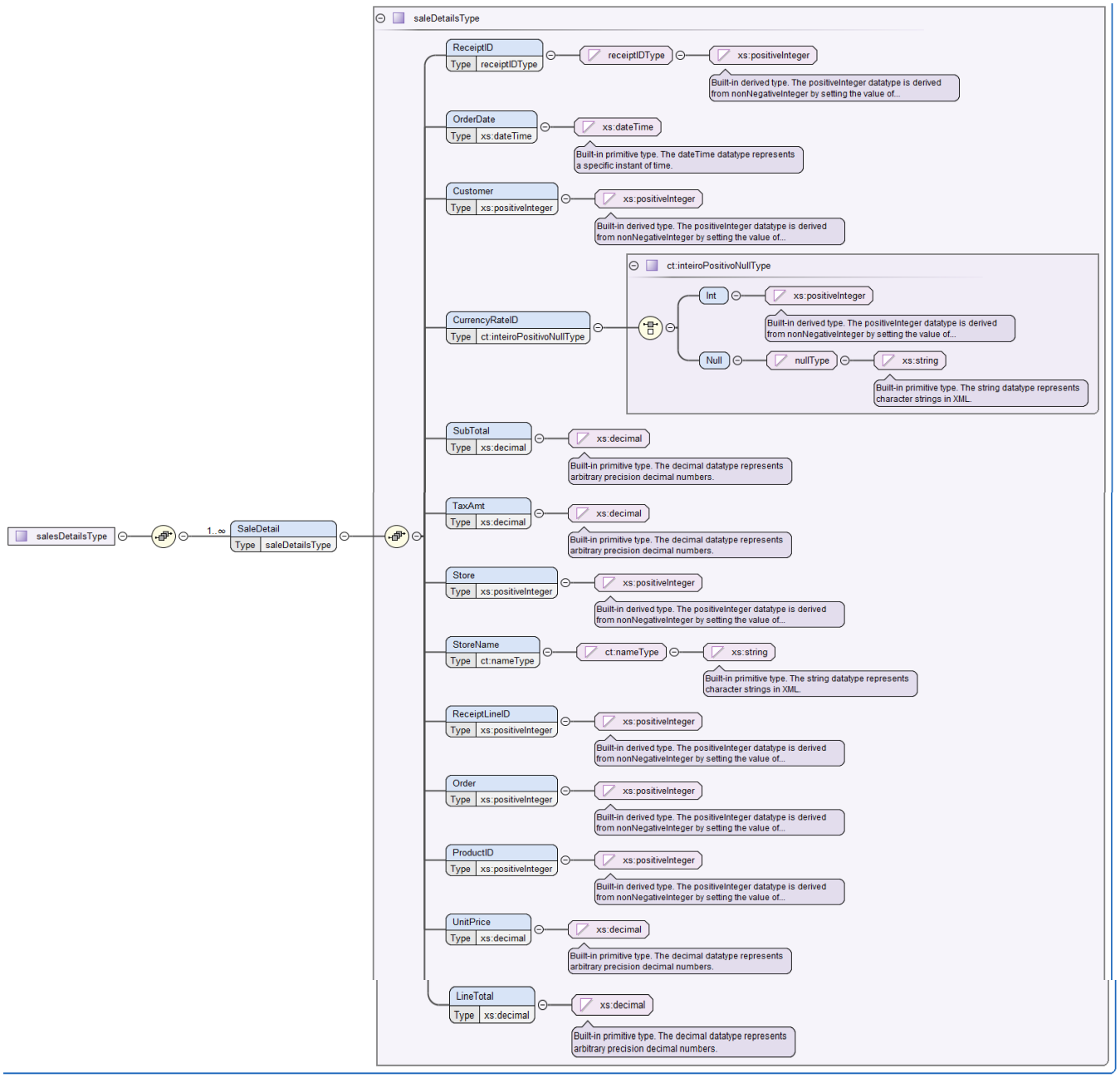
Dependências:

- Tipo: ct:CommonTypes
- NameSpace: http://BikeOnTrack.com/CommonTypes.
- Localização: CommomTypes.xsd

✓ SalesDetails

- NameSpace: http://BikeOnTrack.com/SalesDetais.
- Propriedades: elementFormDefault="unqualified".
- Localização: SalesDetais.xsd
- Descrição: Elemento contendo as informações relativas aos detalhes das vendas.
- Descrição visual:




salesDetailsType
1..∞
SaleDetail
Type saleDetailsType
saleDetailsType
ReceiptID
Type receiptIDType
receiptIDType
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
OrderDate
Type xs:dateTime
xs:dateTime
Built-in primitive type. The dateTime datatype represents a specific instant of time.
Customer
Type xs:positiveInteger
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
CurrencyRateID
Type ct:inteiroPositivoNullType
ct:inteiroPositivoNullType
Int
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
Null
nullType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
SubTotal
Type xs:decimal
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
TaxAmt
Type xs:decimal
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
Store
Type xs:positiveInteger
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
StoreName
Type ct:nameType
ct:nameType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
ReceiptLineID
Type xs:positiveInteger
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
Order
Type xs:positiveInteger
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
ProductID
Type xs:positiveInteger
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
UnitPrice
Type xs:decimal
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
LineTotal
Type xs:decimal
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.

Estrutura XSD:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ct="http://BikeOnTrack.com/CommonTypes"
           xmlns="http://BikeOnTrack.com/SalesDetails"
           targetNamespace="http://BikeOnTrack.com/SalesDetails"
           elementFormDefault="unqualified">

    <!-- Importar XSD secundário -->
    <xs:import schemaLocation="CommonTypes.xsd" namespace="http://BikeOnTrack.com/CommonTypes"/>

    <!-- Definição do tipo de receiptIDType  -->
    <xs:simpleType name="receiptIDType">
        <xs:restriction base="xs:positiveInteger">
            <xs:totalDigits value="5" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Definição de saleDetails -->
    <xs:complexType name="saleDetailsType">
        <xs:sequence>
            <xs:element name="ReceiptID" type="receiptIDType"/>
            <xs:element name="OrderDate" type="xs:dateTime"/>
            <xs:element name="Customer" type="xs:positiveInteger"/>
            <xs:element name="CurrencyRateID" type="ct:inteiroPositivoNullType"/>
            <xs:element name="SubTotal" type="xs:decimal"/>
            <xs:element name="TaxAmt" type="xs:decimal"/>
            <xs:element name="Store" type="xs:positiveInteger"/>
            <xs:element name="StoreName" type="ct:nameType"/>
            <xs:element name="ReceiptLineID" type="xs:positiveInteger"/>
            <xs:element name="Order" type="xs:positiveInteger"/>
            <xs:element name="ProductID" type="xs:positiveInteger"/>
            <xs:element name="UnitPrice" type="xs:decimal"/>
            <xs:element name="LineTotal" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Unidades tem de ter, no mínimo, 1 elemento de saleDetailsType-->
    <xs:complexType name="salesDetailsType">
        <xs:sequence>
            <xs:element name="SaleDetail" type="saleDetailsType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

Exemplo XML:

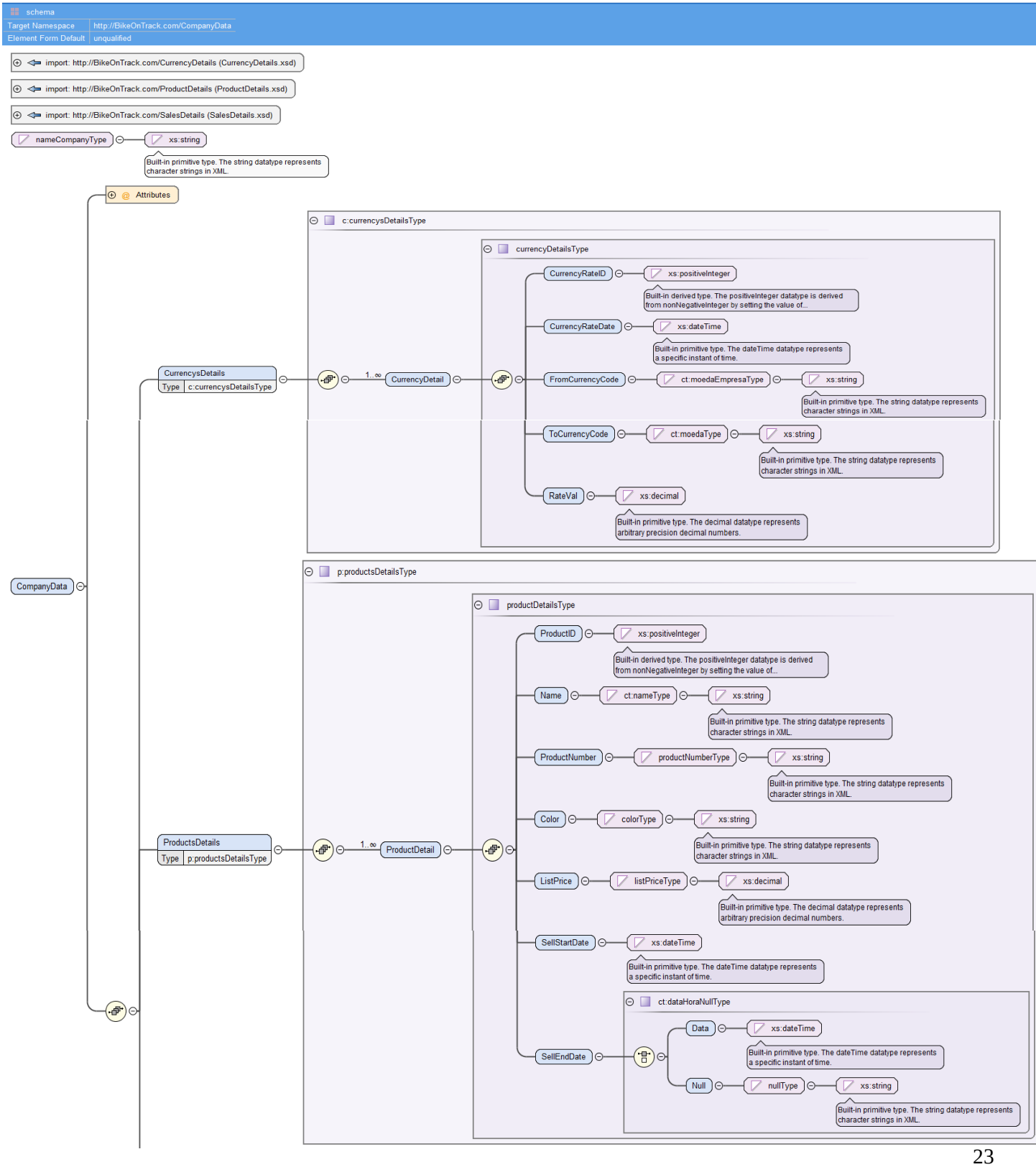
```xml
<!-- SalesDetails -->
<SalesDetails>

  <!-- Primeiro exemplo de SaleDetail -->
  <SaleDetail>
      <ReceiptID>43659</ReceiptID>
      <OrderDate>2011-05-31T00:00:00</OrderDate>
      <Customer>29825</Customer>
      <CurrencyRateID>
          <Null>NULL</Null>
      </CurrencyRateID>
      <SubTotal>20565.6206</SubTotal>
      <TaxAmt>1971.5149</TaxAmt>
      <Store>1046</Store>
      <StoreName>Better Bike Shop</StoreName>
      <ReceiptLineID>1</ReceiptLineID>
      <Order>1</Order>
      <ProductID>776</ProductID>
      <UnitPrice>2024.994</UnitPrice>
      <LineTotal>2024.994000</LineTotal>
  </SaleDetail>
</SalesDetails>
```

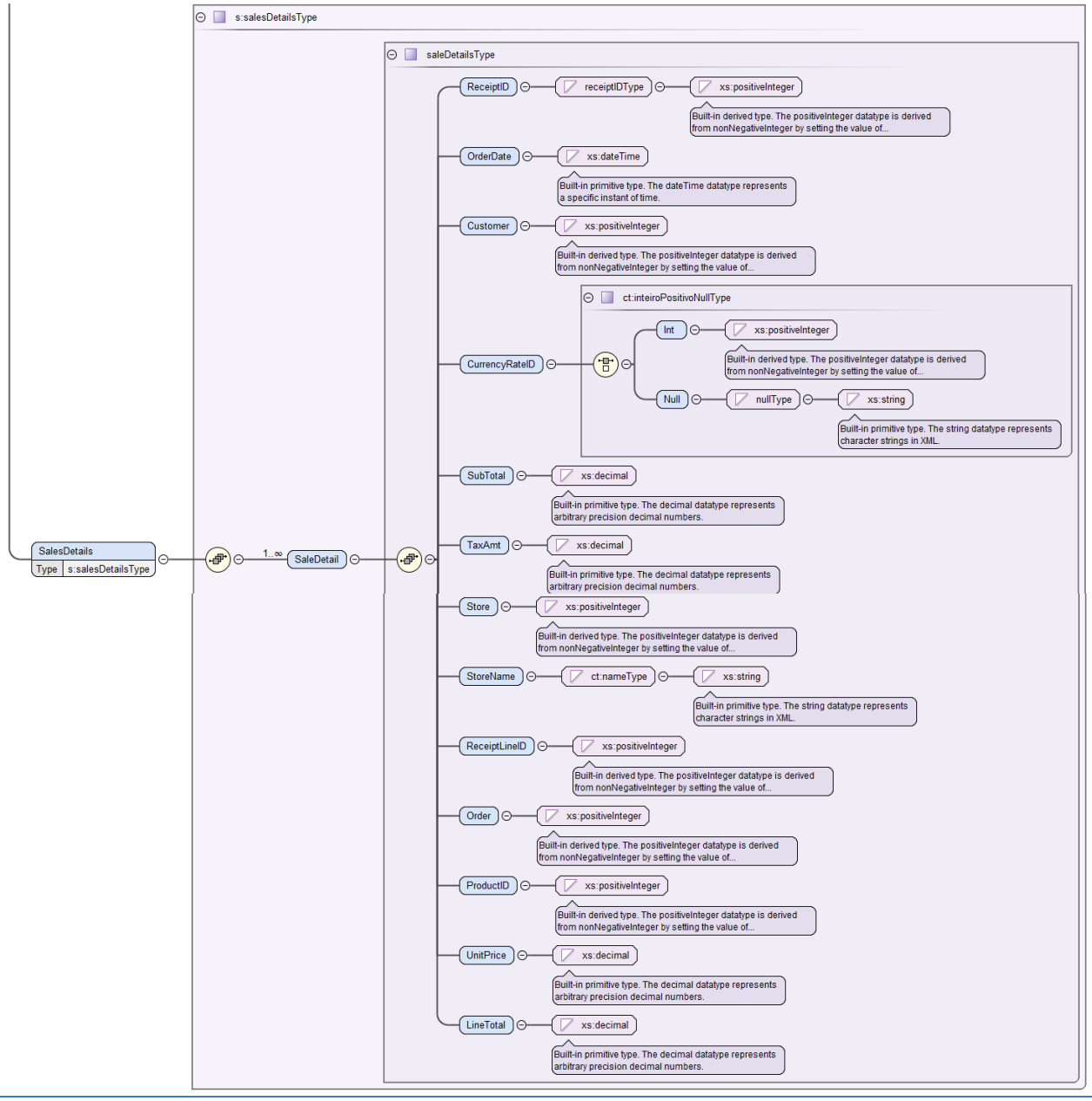
Dependências:

- Tipo: ct:CommonTypes
- NameSpace: http://BikeOnTrack.com/CommonTypes.
- Localização: CommomTypes.xsd

✓ CompanyData

- NameSpace: http://BikeOnTrack.com/CompanyData.
- Propriedades: elementFormDefault="unqualified".
- Localização: CompanyData.xsd
- Descrição: Elemento contendo as informações relativas aos detalhes da empresa.
- Descrição visual:




s:salesDetailsType
saleDetailsType
SalesDetails
Type
s:salesDetailsType
1..∞
SaleDetail
ReceiptID
receiptIDType
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
OrderDate
xs:dateTime
Built-in primitive type. The dateTime datatype represents a specific instant of time.
Customer
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
CurrencyRateID
ct:inteiroPositivoNullType
Int
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
Null
nullType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
SubTotal
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
TaxAmt
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
Store
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
StoreName
ct:nameType
xs:string
Built-in primitive type. The string datatype represents character strings in XML.
ReceiptLineID
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
Order
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
ProductID
xs:positiveInteger
Built-in derived type. The positiveInteger datatype is derived from nonNegativeInteger by setting the value of...
UnitPrice
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.
LineTotal
xs:decimal
Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.

Estrutura XSD:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:c="http://BikeOnTrack.com/CurrencyDetails"
           xmlns:p="http://BikeOnTrack.com/ProductDetails"
           xmlns:s="http://BikeOnTrack.com/SalesDetails"
           xmlns:="http://BikeOnTrack.com/CompanyData"
           targetNamespace="http://BikeOnTrack.com/CompanyData"
           elementFormDefault="unqualified">

    <!-- Importar XSD's secundários -->
    <xs:import schemaLocation="CurrencyDetails.xsd" namespace="http://BikeOnTrack.com/CurrencyDetails"/>
    <xs:import schemaLocation="ProductDetails.xsd" namespace="http://BikeOnTrack.com/ProductDetails"/>
    <xs:import schemaLocation="SalesDetails.xsd" namespace="http://BikeOnTrack.com/SalesDetails"/>

    <!-- Definição do tipo nameCompanyType -->
    <xs:simpleType name="nameCompanyType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="BikeOnTrack"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Estrutura básica do XML -->
    <xs:element name="CompanyData">
        <xs:complexType>
            <xs:sequence>

                <!-- Estrutura do CurrencysDetails dentro de CurrencyDetails.xsd -->
                <xs:element name="CurrencysDetails" type="c:currencysDetailsType"/>

                <!-- Estrutura do ProductsDetails dentro de ProductDetails.xsd -->
                <xs:element name="ProductsDetails" type="p:productsDetailsType"/>

                <!-- Estrutura do SalesDetails dentro de SalesDetails.xsd -->
                <xs:element name="SalesDetails" type="s:salesDetailsType"/>
            </xs:sequence>
            <xs:attribute name="Company" type="xs:string" use="required" fixed="BikeOnTrack"/>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

Exemplo XML:

```xml
<!-- CurrencysDetails -->
<CurrencysDetails>
 <!-- Primeiro exemplo de currencyDetail -->
 <CurrencyDetail>
     <CurrencyRateID>1</CurrencyRateID>
     <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
     <FromCurrencyCode>USD</FromCurrencyCode>
     <ToCurrencyCode>ARS</ToCurrencyCode>
     <RateVal>1.0002</RateVal>
 </CurrencyDetail>


 <!-- Segundo exemplo de currencyDetail -->
 <CurrencyDetail>
     <CurrencyRateID>2</CurrencyRateID>
     <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
     <FromCurrencyCode>USD</FromCurrencyCode>
     <ToCurrencyCode>AUD</ToCurrencyCode>
     <RateVal>1.55</RateVal>
 </CurrencyDetail>

 <!-- Terceiro exemplo de currencyDetail -->
 <CurrencyDetail>
     <CurrencyRateID>3</CurrencyRateID>
     <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
     <FromCurrencyCode>USD</FromCurrencyCode>
     <ToCurrencyCode>BRL</ToCurrencyCode>
     <RateVal>1.9419</RateVal>
 </CurrencyDetail>
</CurrencysDetails>

<!-- ProductsDetails -->
<ProductsDetails>
 <!-- Primeiro exemplo de ProductDetail -->
 <ProductDetail>
     <ProductID>1</ProductID>
     <Name>Adjustable Race</Name>
     <ProductNumber>AR-5381</ProductNumber>
     <Color>NULL</Color>
     <ListPrice>0</ListPrice>
     <SellStartDate>2008-04-30T00:00:00</SellStartDate>
     <SellEndDate>
         <Null>NULL</Null>
     </SellEndDate>
 </ProductDetail>
```

```
    <!-- Segundo exemplo de ProductDetail -->
    <ProductDetail>
        <ProductID>713</ProductID>
        <Name>Long-Sleeve Logo Jersey, S</Name>
        <ProductNumber>LJ-0192-S</ProductNumber>
        <Color>Multi</Color>
        <ListPrice>49.99</ListPrice>
        <SellStartDate>2011-05-31T00:00:00</SellStartDate>
        <SellEndDate>
            <Null>NULL</Null>
        </SellEndDate>
    </ProductDetail>

    <!-- Terceiro exemplo de ProductDetail -->
    <ProductDetail>
        <ProductID>717</ProductID>
        <Name>HL Road Frame - Red, 62</Name>
        <ProductNumber>FR-R92R-62</ProductNumber>
        <Color>Red</Color>
        <ListPrice>1431.50</ListPrice>
        <SellStartDate>2011-05-31T00:00:00</SellStartDate>
        <SellEndDate>
            <Null>NULL</Null>
        </SellEndDate>
    </ProductDetail>
</ProductsDetails>

<!-- SalesDetails -->
<SalesDetails>

    <!-- Primeiro exemplo de SaleDetail -->
    <SaleDetail>
        <ReceiptID>43659</ReceiptID>
        <OrderDate>2011-05-31T00:00:00</OrderDate>
        <Customer>29825</Customer>
        <CurrencyRateID>
            <Null>NULL</Null>
        </CurrencyRateID>
        <SubTotal>20565.6206</SubTotal>
        <TaxAmt>1971.5149</TaxAmt>
        <Store>1046</Store>
        <StoreName>Better Bike Shop</StoreName>
        <ReceiptLineID>1</ReceiptLineID>
        <Order>1</Order>
        <ProductID>776</ProductID>
        <UnitPrice>2024.994</UnitPrice>
        <LineTotal>2024.994000</LineTotal>
    </SaleDetail>
```

```
    <!-- Segundo exemplo de SaleDetail -->
    <SaleDetail>
        <ReceiptID>44100</ReceiptID>
        <OrderDate>2011-08-01T00:00:00</OrderDate>
        <Customer>29690</Customer>
        <CurrencyRateID>
            <Null>NULL</Null>
        </CurrencyRateID>
        <SubTotal>81164.2329</SubTotal>
        <TaxAmt>7884.6612</TaxAmt>
        <Store>764</Store>
        <StoreName>Fashionable Bikes and Accessories</StoreName>
        <ReceiptLineID>1652</ReceiptLineID>
        <Order>6</Order>
        <ProductID>709</ProductID>
        <UnitPrice>5.7</UnitPrice>
        <LineTotal>34.200000</LineTotal>
    </SaleDetail>

    <!-- Terceiro exemplo de SaleDetail -->
    <SaleDetail>
        <ReceiptID>43668</ReceiptID>
        <OrderDate>2011-05-31T00:00:00</OrderDate>
        <Customer>29614</Customer>
        <CurrencyRateID>
            <Int>4</Int>
        </CurrencyRateID>
        <SubTotal>35944.1562</SubTotal>
        <TaxAmt>3461.7654</TaxAmt>
        <Store>592</Store>
        <StoreName>Retail Mall</StoreName>
        <ReceiptLineID>81</ReceiptLineID>
        <Order>3</Order>
        <ProductID>756</ProductID>
        <UnitPrice>874.794</UnitPrice>
        <LineTotal>2624.382000</LineTotal>
    </SaleDetail>
  </SalesDetails>

</c:CompanyData>
```

Dependências:

- Tipo: c:CurrencyDetails
- NameSpace: http://BikeOnTrack.com/CurrencyDetails.
- Localização: CurrencyDetails.xsd

Tipo: p:ProductDetails

- NameSpace: http://BikeOnTrack.com/ProductDetails.
- Localização: ProductDetails.xsd

Tipo: s:SalesDetails

- NameSpace: http://BikeOnTrack.com/SalesDetails.
- Localização: SalesDetails.xsd

Exemplo da aplicação do Schema

Na criação do schema tentamos que as restrições impostas não tornassem o XML menos flexível, impedindo, ao mesmo tempo, que o utilizador cometesse erros críticos para o funcionamento de alguma aplicação que eventualmente venha a usar o nosso XML.

Por exemplo, restringindo os ID´s a serem números inteiros positivos para que desta forma o utilizador não posso colocar números negativos ou não inteiros, restringimos os nomes a quarenta caracteres para que desta forma o utilizador não coloque uma string com conteúdo simples e direto para uma mais fácil compreensão e não limitamos o tamanho do "UnitPrice" mas sim o número de casas decimais que apesar de no ficheiro disponibilizado notarmos que nunca ultrapassava um certo número, sabemos que no futuro poderá ser preciso colocar um número maior nesse campo.

```
<?xml version="1.0" encoding="UTF-8"?>

<c:CompanyData Company="BikeOnTrack"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:c="http://BikeOnTrack.com/CompanyData"
        xsi:schemaLocation="http://BikeOnTrack.com/CompanyData CompanyData.xsd"> <!-- Importação do Schema XSD -->

  <!-- CurrencysDetails -->
  <CurrencysDetails>
   <!-- Primeiro exemplo de currencyDetail -->
   <CurrencyDetail>
      <CurrencyRateID>1</CurrencyRateID>
      <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
      <FromCurrencyCode>USD</FromCurrencyCode>
      <ToCurrencyCode>ARS</ToCurrencyCode>
      <RateVal>1.0002</RateVal>
   </CurrencyDetail>

   <!-- Segundo exemplo de currencyDetail -->
   <CurrencyDetail>
      <CurrencyRateID>2</CurrencyRateID>
      <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
      <FromCurrencyCode>USD</FromCurrencyCode>
      <ToCurrencyCode>AUD</ToCurrencyCode>
      <RateVal>1.55</RateVal>
   </CurrencyDetail>

   <!-- Terceiro exemplo de currencyDetail -->
   <CurrencyDetail>
      <CurrencyRateID>3</CurrencyRateID>
      <CurrencyRateDate>2011-05-31T00:00:00</CurrencyRateDate>
      <FromCurrencyCode>USD</FromCurrencyCode>
      <ToCurrencyCode>BRL</ToCurrencyCode>
      <RateVal>1.9419</RateVal>
   </CurrencyDetail>
  </CurrencysDetails>

  <!-- ProductsDetails -->
  <ProductsDetails>
   <!-- Primeiro exemplo de ProductDetail -->
   <ProductDetail>
      <ProductID>1</ProductID>
      <Name>Adjustable Race</Name>
      <ProductNumber>AR-5381</ProductNumber>
      <Color>NULL</Color>
      <ListPrice>0</ListPrice>
      <SellStartDate>2008-04-30T00:00:00</SellStartDate>
```

```
    <SellEndDate>
      <Null>NULL</Null>
    </SellEndDate>
  </ProductDetail>

  <!-- Segundo exemplo de ProductDetail -->
  <ProductDetail>
    <ProductID>713</ProductID>
    <Name>Long-Sleeve Logo Jersey, S</Name>
    <ProductNumber>LJ-0192-S</ProductNumber>
    <Color>Multi</Color>
    <ListPrice>49.99</ListPrice>
    <SellStartDate>2011-05-31T00:00:00</SellStartDate>
    <SellEndDate>
      <Null>NULL</Null>
    </SellEndDate>
  </ProductDetail>

  <!-- Terceiro exemplo de ProductDetail -->
  <ProductDetail>
    <ProductID>717</ProductID>
    <Name>HL Road Frame - Red, 62</Name>
    <ProductNumber>FR-R92R-62</ProductNumber>
    <Color>Red</Color>
    <ListPrice>1431.50</ListPrice>
    <SellStartDate>2011-05-31T00:00:00</SellStartDate>
    <SellEndDate>
      <Null>NULL</Null>
    </SellEndDate>
  </ProductDetail>
</ProductsDetails>


<!-- SalesDetails -->
<SalesDetails>

  <!-- Primeiro exemplo de SaleDetail -->
  <SaleDetail>
    <ReceiptID>43659</ReceiptID>
    <OrderDate>2011-05-31T00:00:00</OrderDate>
    <Customer>29825</Customer>
    <CurrencyRateID>
      <Null>NULL</Null>
    </CurrencyRateID>
    <SubTotal>20565.6206</SubTotal>
    <TaxAmt>1971.5149</TaxAmt>
    <Store>1046</Store>
    <StoreName>Better Bike Shop</StoreName>
    <ReceiptLineID>1</ReceiptLineID>
    <Order>1</Order>
    <ProductID>776</ProductID>
    <UnitPrice>2024.994</UnitPrice>
    <LineTotal>2024.994000</LineTotal>
  </SaleDetail>

  <!-- Segundo exemplo de SaleDetail -->
  <SaleDetail>
    <ReceiptID>44100</ReceiptID>
    <OrderDate>2011-08-01T00:00:00</OrderDate>
    <Customer>29690</Customer>
```

```
      <CurrencyRateID>
         <Null>NULL</Null>
      </CurrencyRateID>
      <SubTotal>81164.2329</SubTotal>
      <TaxAmt>7884.6612</TaxAmt>
      <Store>764</Store>
      <StoreName>Fashionable Bikes and Accessories</StoreName>
      <ReceiptLineID>1652</ReceiptLineID>
      <Order>6</Order>
      <ProductID>709</ProductID>
      <UnitPrice>5.7</UnitPrice>
      <LineTotal>34.200000</LineTotal>
</SaleDetail>

   <!-- Terceiro exemplo de SaleDetail -->
   <SaleDetail>
      <ReceiptID>43668</ReceiptID>
      <OrderDate>2011-05-31T00:00:00</OrderDate>
      <Customer>29614</Customer>
      <CurrencyRateID>
         <Int>4</Int>
      </CurrencyRateID>
      <SubTotal>35944.1562</SubTotal>
      <TaxAmt>3461.7654</TaxAmt>
      <Store>592</Store>
      <StoreName>Retail Mall</StoreName>
      <ReceiptLineID>81</ReceiptLineID>
      <Order>3</Order>
      <ProductID>756</ProductID>
      <UnitPrice>874.794</UnitPrice>
      <LineTotal>2624.382000</LineTotal>
   </SaleDetail>
  </SalesDetails>

</c:CompanyData>
```

Justificação da abordagem utilizada

A abordagem que utilizamos obriga a que no XML o utilizador ou o sistema tenha de colocar no mínimo um parâmetro de cada “CurrencyDetails”, “ProductDetails” e “SalesDetails” por esta mesma ordem.
Apesar de estas restrições tornarem o nosso XML muito menos flexível obriga o utilizador ou o sistema a manter o ficheiro organizado para que desta forma seja mais simples e mais rapidamente seja a visualização do conteúdo que procuramos.

Obrigamos também a ter um exemplo em cada um para que o documento fique sempre estruturado sabendo sempre como e onde colocar cada informação.

Poderíamos ter optado por deixar ficar a informação solta no meio do XML tornando assim mais fácil a sua escrita, porém, isso iria tornar a sua consulta deveras mais complicada.
Tínhamos também a opção de não obrigar a possuir no mínimo um exemplo de cada, mas isso tornaria um pouco mais confuso pois não teríamos um exemplo por onde nos guiar que por vezes nos deixaria sem saber onde colocar a informação.

Conclusão e principais dificuldades

As principais dificuldades que encontramos na realização do projeto foi descobrir o meio termo entre restringir e ser flexível na criação do XML, pois não queríamos que o seu conteúdo ficasse perdido no meio do XML mas também não queríamos que fosse muito complexo a sua criação. Achando esta forma a melhor maneira para conseguir ficar entre o meio termo.

A principal conclusão que retiramos do projeto relaciona-se com a utilidade do XML e as suas tecnologias adjacentes para a organização e obtenção de informação.

Gostávamos também de informar que utilizamos “git ” para a realização do trabalho, não para controlar as partes dos vários elementos do grupo, pois o trabalho foi sempre realizado em conjunto, mas sim para controlo de algumas versões e principalmente como backup do trabalho. Segue o link do repositório (https://github.com/xPromate/TrabalhoPEIParte1) .