Integração de Dados – Exame da Época de Recurso

Duração: 90 minutos

CTSPs - TPSI - 2º ano/1º semestre 2017/2018

1. [10%] Responda às seguintes questões:

a) Indique duas dificuldades que se podem encontrar na implementação de um sistema de Integração de Dados. Concretize com exemplos simples.

b) Em que consiste um documento XML válido e bem formado?

2. [10%] Analise o ficheiro XML dado no anexo 1 e responda às questões:

a) Escreva a instrução DTD que permita validar o elemento **titulo**

```
<!ELEMENT titulo  ( #PCDATA )>
```

b) Escreva a instrução DTD que permita validar o elemento **livro**

```
<!ELEMENT livro  ( titulo , bonus , lista_autores )>
```

c) Escreva a instrução DTD que permita validar o atributo **cod** como identificador único

```
<!ATTLIST livro cod ID #REQUIRED>
```

d) Escreva a instrução DTD que permita validar o atributo **filial** como uma enumeração de três valores (Lisboa, Porto, Coimbra)

```
<!ATTLIST livraria filial (Lisboa | Porto | Coimbra) #REQUIRED>
```

3. [10%] Analise o XML dado no anexo 1 e responda às questões:

a) Construa um **tipo de dados** XSD que permita validar os elementos **novos** e **usados** (assuma que os restantes elementos e os atributos estão definidos, e use a instrução ref).

```
<xsd:complexType name =”tipoLivro>
   <xsd:sequence>
        <xsd:element ref="descricao" minOccurs=”0” />
        <xsd:element ref="livro" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute ref="data_registo" use="optional"/>
</xsd:complexType>
```

b) Usando o tipo de dados anterior, escreva o XSD que permita validar os elementos **novos** e **usados**

```
<xsd:element name="novos" type="tipoLivro"/>
<xsd:element name="usados" type="tipoLivro"/>
```

c) Escreva o XSD que permita validar o atributo **filial** como uma enumeração de três valores possíveis (Lisboa, Porto, Coimbra)

```
<xsd:attribute name="filial">
      <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="Lisboa"/>
                  <xsd:enumeration value="Porto"/>
                  <xsd:enumeration value="Coimbra"/>
      </xsd:restriction>
      </xsd:simpleType>
</xsd:attribute>
```

d) Escreva o XSD que permita validar o elemento **aut.** Assuma que o atributo já foi definido e use a instrução *ref*.

```
<xsd:element name="aut">
   <xsd:complexType>
         <xsd:attribute ref="id" use="required"/>
   </xsd:complexType>
</xsd:element>
```

4. [10%] Apresente as Expressões Regulares (ER) que permitam executar as seguintes tarefas:

a) Escreva uma ER que permita encontrar todas as palavras de um texto que correspondem a um valor hexadecimal. Um número hexadecimal pode conter dígitos e os caracteres A, B, C, D, E, F, maiúsculos ou minúsculos.

```
\b[0-9a-fA-F]+\b
```

b) Escreva uma ER que procure num texto todas as palavras começadas por **por** ou **Por**. A restante palavra pode conter caracteres minúsculos. Exemplos de palavras aceites por esta ER assinaladas no texto abaixo.
`A Maria foi ao porto cidade de Portugal e por todo o lado viu porcelanas à venda.`

```
\b[Pp]or[a-z]*
```

c) Escreva uma ER que permita verificar se uma cadeia de caracteres corresponde a uma linha XML. Assuma que numa linha XML válida tem de existir uma *tag* de abertura, uma *tag* de fecho e qualquer conteúdo entre as *tags*.
Nota: Para simplificar, **não é necessário** verificar se as *tags* de abertura e de fecho tem o mesmo nome!
Exemplos de linhas XML aceites pela ER:

```
<ana>nome</bela>
<dados>Joana 23 anos</dados>
<idade>23</fim>
<TEXTO>Era uma vez</TEXTO>
```

```
<[a-zA-Z]+>[^<]*</[a-zA-Z]+>
```

d) Escreva uma ER que encontre números inteiros válidos, onde o caracter **.** (ponto) seja usado como separador dos milhares.

- Exemplos de números aceites pela ER: **23.010** **1.500** **120.100.240** **1.100.000.500**
- Exemplos de números NÃO aceites pela ER: **1000** **1.200.2** **5000.000** **5.200500**

```
\b([0-9]{1,3}\.)+[0-9]{3}\b
```

5. [10%] Analise o XML dado em anexo 1. Escreva expressões XPath que permitam responder às seguintes necessidades de informação:

a) Títulos (texto) dos livros com mais de um autor

```
//livro[count(lista_autores/aut)>1]/titulo/text()
```

b) Número de livros novos do ano 2017 presentes no ficheiro

```
count(//novos/livro[@ano="2017"])
```

c) Todos os títulos dos livros do autor **Paulo Coelho**

```
//livro[lista_autores/aut/@id = //autores/autor[nome="Paulo Coelho"]/@id]/titulo
```

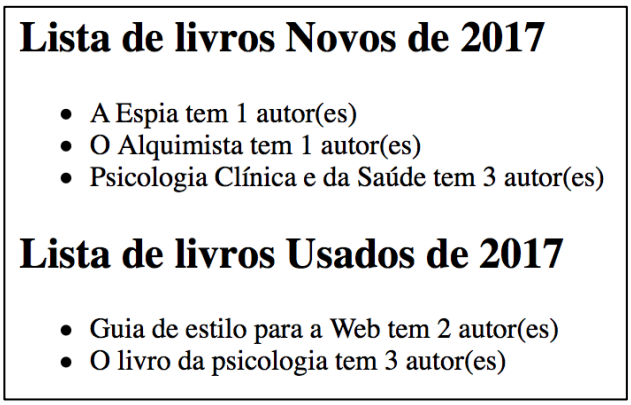
d) Nomes dos autores que recebem mais de 10% (0.1) de direitos de autor

```
//autores/autor[direitos > 0.1]/nome
```

e) Nomes (texto) dos autores do livro cujo titulo é igual a **O livro da psicologia**

```
//autores/autor[@id=//livro[titulo="O livro da ..."]/lista_autores/aut/@id]/nome/text()
```

6. [15%] Analise o ficheiro XML dado no anexo 1.
Escreva o XSLT que permita transformar esse ficheiro num output HTML com a informação mostrada na figura abaixo: título e o número de autores dos livros de 2017
O resultado está ordenado por ordem alfabética do título e apenas é visualizada a informação dos livros de 2017. Deve usar pelo menos uma instrução **for-each** e uma instrução **if**



Lista de livros Novos de 2017

- A Espia tem 1 autor(es)
- O Alquimista tem 1 autor(es)
- Psicologia Clínica e da Saúde tem 3 autor(es)

Lista de livros Usados de 2017

- Guia de estilo para a Web tem 2 autor(es)
- O livro da psicologia tem 3 autor(es)

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
       <xsl:output method="html"/>

     <xsl:template match="livraria">
          <html><body>
          <h2>Lista de livros Novos  de 2017</h2>
          <ul>
            <xsl:apply-templates select="novos/livro">
                   <xsl:sort select="titulo"/>
            </xsl:apply-templates>
          </ul>

           <h2>Lista de livros Usados  de 2017</h2>
           <ul>
             <xsl:apply-templates select="usados/livro">
                   <xsl:sort select="titulo"/>
             </xsl:apply-templates>
           </ul>
         </body></html>
       </xsl:template>

       <xsl:template match="novos/livro">
          <xsl:if test="@ano = 2017">
           <li><xsl:value-of  select="titulo"/>  tem  <xsl:value-of  select="count(lista_autores/aut)"/>
autor(es)</li>
             </xsl:if>
       </xsl:template>

      <xsl:template match="usados/livro">
          <xsl:if test="@ano = 2017">
          <li><xsl:value-of    select="titulo"/>    tem    <xsl:value-of    select="count(lista_autores/aut)"/>
autor(es)</li>
             </xsl:if>
       </xsl:template>
</xsl:stylesheet>
```

7. [30%] Analise o XML dado em anexo e responda às questões:

a) Escreve uma expressão XQuery que aplicada ao ficheiro XML do anexo 1 faça a mesma transformação da pergunta 6.

```
<html><body>
<h2>Lista de livros Novos de 2017</h2>
<ul>
{
	for $x in doc("livraria.xml")//novos/livro
	let $n := count($x/lista_autores/aut)
	where $x/@ano = 2017
	order by $x/titulo
	return <li>{$x/titulo/text()} tem {$n} autor(es)</li>
}
</ul>

<h2>Lista de livros Usados de 2017</h2>
<ul>
{
	for $x in doc("livraria.xml")//usados/livro
	let $n := count($x/lista_autores/aut)
	where $x/@ano = 2017
	order by $x/titulo
	return <li>{$x/titulo/text()} tem {$n} autor(es)</li>
}
</ul>
 </body></html>
```

b) Escreva uma função XQuery de nome **CalculaTotais** que receba o **id** de um livro e o valor do **bónus** e calcule as vendas efectuadas para esse livro. Deve usar as quantidades vendidas e o preço unitário de cada livro (estão em vendas.xml) e adicionar ao valor calculado o respectivo bónus.

```
xquery version "1.0";
declare namespace xsd = "http://www.w3.org/2001/XMLSchema";

declare function local:calculaTotais($id as xsd:string, $bonus as xsd:decimal) as
xsd:double
{
	let $q := doc("vendas.xml")//livro[@id = $id]/@quant
	let $p := doc("vendas.xml")//livro[@id = $id]/@preco
	return $q * $p + $bonus
 };
```

c) Escreva uma expressão XQuery que aplicada ao ficheiro XML **livraria.xml** crie um ficheiro XML com o total de vendas feitas para cada livro de 2017. O valor das vendas de cada livro deve ser calculado pela função da alínea anterior (elemento <**total**> do output). Deve ser criado um elemento <**bonus**> contendo a palavra **Sim** ou **Não** de acordo com o valor do bónus ser superior a zero, ou não, respectivamente. Resultado do output, ordenado por título:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<resultados>
   <vendas>
      <titulo>A Espia</titulo>
      <bonus>SIM</bonus>
      <total>1050</total>
   </vendas>
   <vendas>
      <titulo>Guia de estilo para a Web</titulo>
      <bonus>NÃO</bonus>
      <total>300</total>
   </vendas>
   <vendas>
      <titulo>O Alquimista</titulo>
      <bonus>SIM</bonus>
      <total>150</total>
   </vendas>
   <vendas>
      <titulo>O livro da psicologia</titulo>
      <bonus>SIM</bonus>
      <total>160</total>
   </vendas>
   <vendas>
      <titulo>Psicologia Clínica e da Saúde</titulo>
      <bonus>NÃO</bonus>
      <total>300</total>
   </vendas>
</resultados>
```

```
<resultados>{
    for $liv in doc("livraria.xml")//livro
    let $tot := local:calculaTotais($liv/@cod, $liv/bonus)

    where $liv/@ano = '2017'
    order by $liv/titulo
    return if (number($liv/bonus) > 0) then
                <vendas>
                      <titulo>{$liv/titulo/text()}</titulo>
                      <bonus>SIM</bonus>
                      <total>{$tot}</total>
                 </vendas>
           else
                <vendas>
                      <titulo>{$liv/titulo/text()}</titulo>
                       <bonus>NÃO</bonus>
                       <total>{$tot}</total>
                </vendas>
}
</resultados>
```