Documento XML grande: a bela ou o monstro?

Marta H. Jacinto

ITIJ — Instituto das Tecnologias de Informação na Justiça
Ministério da Justiça
1049-068 Lisboa
marta.jacinto@itij.mj.pt

Resumo O ITIJ foi encarregado de enviar os dados da DGRN, que tem inúmeros serviços e vários funcionários em cada um deles, para a BDAP — base de dados que deve conter os dados referentes a todos os funcionários públicos. Para isso foi necessário trabalhar com documentos da ordem dos 22MB e fazer várias transformações sobre eles, algumas resultando em documentos com tamanho na ordem dos 14MB. O facto de as primeiras transformações, feitas em ambiente não académico e para um caso real, demorarem tempos relativamente curtos, motivaram a escrita deste artigo e levaram à avaliação de todas as outras, numa tentativa de responder à questão "Documento XML grande: a bela ou o monstro?". Conclui-se que, apesar de algumas transformações serem feitas mais rapidamente que outras, os documentos XML desta ordem de grandeza são tratáveis.

1 Introdução

O XML tem vindo a ganhar cada vez mais adeptos, conquistando, a pouco e pouco, todas as áreas de informação. O projecto BDAP (Base de Dados de Recursos Humanos da Administração Pública) — descrito na secção 2 — que pretende ser um banco de informação sobre todos os funcionários públicos do país, não é excepção. Para o transporte de informação entre aplicações foi escolhido o XML. Essa escolha adequa-se perfeitamente à diversidade dos vários organismos da Administração Pública, permitindo estabelecer uma plataforma única para o intercâmbio da informação.

O ITIJ (Instituto das Tecnologias de Informação na Justiça), "centro de informática" do Ministério da Justiça, presta serviços a vários serviços daquele Ministério, nomeadamente a DGRN (Direcção-Geral dos Registos e Notariado). Esta última tem vários serviços (Conservatórias do Registo Predial, Conservatórias do Registo Comercial, etc) e dispunha já de uma base de dados com dados sobre os funcionários quando surgiu a iniciativa da BDAP, pelo que incumbiu o ITIJ de fazer as alterações necessárias para recolher os dados adicionais requeridos para a BDAP e posteriormente gerar o XML necessário para o envio. Na secção 3 apresenta-se o formato utilizado para recolher os dados adicionais,

Microsoft Access 2003, e o modo como se gerou automaticamente um dialecto XML standard usando esse programa.

Depois de se dispor dos dados num dialecto XML do Microsoft Access foi necessário transformar esse XML no dialecto XML definido para a BDAP. A complexidade de desenho desta transformação não é grande e o seu processamento é rápido, como se pode ver na secção 4 em que se apresenta o método utilizado para gerar tal XML.

Visto que esta transformação, feita sobre um documento XML de tamanho não académico, demorou pouco tempo, surgiu a ideia de avaliar todas as outras transformações numa tentativa de encontrar uma resposta à questão "Documento XML grande: a bela ou o monstro?". Foi precisamente essa avaliação que motivou a escrita deste artigo.

Uma vez que a introdução de dados no Microsoft Access não é objecto de qualquer validação, foi necessário fazer uma folha de estilos XSL para validar os dados inseridos e gerar um relatório com os erros resultantes da validação, como se mostra na secção 5, permitindo assim a alteração dos dados por parte da DGRN. Nesta mesma secção apresenta-se a forma como se lidou com os erros reportados pelo Instituto de Informática ao receber os ficheiros XML.

Na figura 1 apresenta-se o esquema da arquitectura de todo o sistema que, como enunciado acima, se descreve pormenorizadamente nas secções 2, 3 e 4.

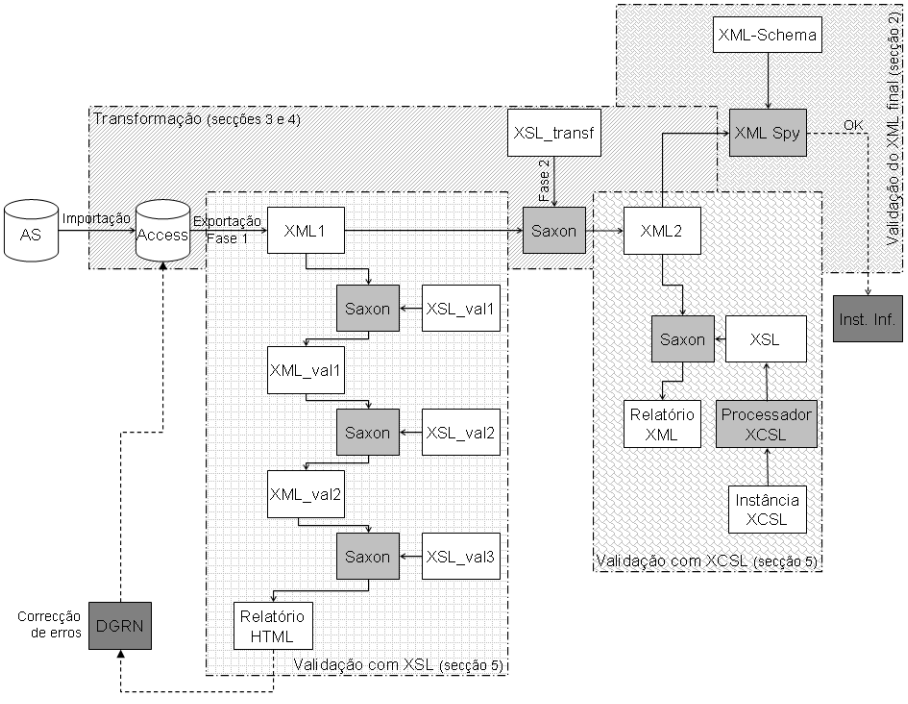

Figura 1. Arquitectura de todo o sistema.

Na secção 6 conclui-se o artigo, apresentando a comparação dos tempos de processamento das várias transformações enumeradas.

Todas as transformações foram feitas usando um PC com processador Pentium 4 com 2,40GHz e 504MB de RAM.

2 BDAP

A BDAP[1] (Base de Dados de Recursos Humanos da Função Pública) criada pelo Decreto-Lei nº 47/98, de 7 de Março obriga a um processo de permanente actualização capaz de fornecer, a todo o momento, a informação necessária à produção de indicadores de gestão e planeamento dos recursos humanos da Administração [1]. É, assim, um sistema de informação que pretende agregar todos os dados relativos aos funcionários públicos, independentemente da natureza do seu vínculo à função pública. Deve conter dados de identificação como o nome, o número de contribuinte, o sexo e a data de nascimento; dados sobre a remuneração e os suplementos de remuneração; as acções de formação; as habilitações literárias; os organismos de funções, vínculo e pagador; etc.

Esta base de dados é da responsabilidade conjunta da DGAP (Direcção-Geral da Administração Pública) e do Instituto de Informática do Ministério das Finanças.

Para o carregamento dos dados há dois processos disponíveis: interactivo — para organismos até 50 trabalhadores e sem sistema de gestão de recursos humanos; envio de um ficheiro XML — para organismos com mais de 50 trabalhadores.

Tendo vários milhares de trabalhadores, a DGRN faz parte deste segundo grupo de organismos.

O Instituto de Informática e a DGAP estabeleceram algumas regras para a escrita destes documentos XML, que constam de [2], para além de outros documentos com regras sobre a recolha dos dados e os vários carregamentos. Foi disponibilizado ainda um esqueleto do documento XML que, apesar de nada dizer quanto à cardinalidade de cada elemento, dá uma ideia dos elementos a utilizar e suas dependências [3]. A seguir apresenta-se um extracto desse esqueleto:

```
<entidaderemetente cod_local_trabalho="" cod_servico="" cod_organismo="">
  <organismo cod_organismo="" num_fiscal="">
     <dataref/>
     <nenvio/>
     <tipo_operacao/>
     <trabalhador num_fiscal="">
        <cenario/>
        <nome/>
        <sexo/>
        ...
        <data_nasc/>
        <nacionalidade/>
        <antiguidade_fp>
```

[1] http://www.bdap.min-financas.pt

```
            ...
         </antiguidade_fp>
         <emprego_organismo>
            ...
            <situacao_profissional>
               ...
            </situacao_profissional>
            ...
            <remuneracao>
               ...
            </remuneracao>
            ...
         </emprego_organismo>
         <habilitacoes_literarias data_obtencao="">
            ...
         </habilitacoes_literarias>
         <formacao_mais_18h data="">
            ...
         </formacao_mais_18h>
      </trabalhador>
   </organismo>
</entidaderemetente>
```

Uma vez que o Instituto de Informática não disponibilizou qualquer XML-Schema, foi necessário escrever um XML-Schema ([4,5]) que agregasse tanto quanto possível a informação constante nos vários documentos já referidos e permitisse depois validar os documentos XML gerados para envio — neste caso não se põe o problema de ajudar o utilizador a escrever documentos mas sim de ter um XML-Schema para ajudar a validar o documento XML final. Nele foram incluídas todas as indicações dos documentos [2] e [6]. Abaixo encontra-se um extracto desse XML-Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="entidaderemetente">
  <xs:complexType>
   <xs:sequence maxOccurs="unbounded">
    <xs:element name="organismo" type="torganismo"/>
   </xs:sequence>
   <xs:attribute name="cod_local_trabalho" type="tnum6" use="required"/>
   <xs:attribute name="cod_servico" type="tnum6" use="required"/>
   <xs:attribute name="cod_organismo" type="tnum6" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:complexType name="tremuneracao_mensal">
  <xs:sequence>
```

```
    ...
    <xs:element name="suplementos_regulares" type="tsuplementos_regulares"
    minOccurs="0" maxOccurs="unbounded"/>
    ...
   </xs:sequence>
  </xs:complexType>
  ...
  <xs:complexType name="tsuplementos_regulares">
   <xs:sequence>
     <xs:element name="codigo" type="tnum3"/>
     <xs:element name="valor_euros" type="treal42"/>
     ...
   </xs:sequence>
  </xs:complexType>
  ...
  <xs:simpleType name="treal42">
   <xs:restriction base="xs:string">
     <xs:pattern value="([0-4]{0,1}[0-9]{0,3}|[0-4]{0,1}[0-9]{0,3},
     [0-9]{0,2})"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:schema>
```

3 Origem dos dados

Quando surgiu a iniciativa da BDAP já havia um conjunto de dados relativos aos funcionários da DGRN em AS. Esses dados foram transferidos para uma base de dados em Microsoft Access, utilizando-se posteriormente o próprio Microsoft Access para recolher os dados para a BDAP.

Depois da recolha colocou-se a questão da transformação dos dados recolhidos para o XML pedido pela BDAP. Este problema foi resolvido em duas fases: 1^a — exportação automática dos dados para um dialecto XML standard do Microsoft Access; 2^a — transformação do dialecto XML gerado automaticamente no dialecto XML da BDAP.

Para a 1^a fase utilizou-se uma opção disponível a partir da versão 2003 do Microsoft Access que permite exportar os dados para XML, ou seja, gerar automaticamente XML a partir de uma base de dados, colocando um elemento raiz com o nome *dataroot*; tantos elementos filho com o nome da tabela (neste caso `T_Bdap`) quantos os registos; e cada um destes com tantos elementos filho quantos os campos da tabela preenchidos no caso desse registo. Os caracteres especiais são traduzidos automaticamente para conjuntos de caracteres, por exemplo "(" é substituído por "_x0028_" e " " é substituído por "_x0020_".

A geração automática de XML a partir da base de dados em Microsoft Access com 38MB demora 7 minutos, gerando um documento da seguinte forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="bdap_access_20041022.xsd"
generated="2004-10-22T15:28:35">
 <T_Bdap>
  <Cod_Serviço_x0028_interno_x0029_>0106401
       </Cod_Serviço_x0028_interno_x0029_>
  <ID>375</ID>
  <Serviço>Conservatória Registo Civil, Predial Castelo Paiva</Serviço>
  ...
  <Dataref>2003-06-30T00:00:00</Dataref>
  <Nenvio>1</Nenvio>
  <Tipo_operacao>A</Tipo_operacao>
  <Data_x0020_Ingresso>1981-01-27T00:00:00</Data_x0020_Ingresso>
  <Cod_Ingresso>1</Cod_Ingresso>
  <N_x00BA__x0020_Fiscal_x0020_Contribuinte_Trab>112529597
       </N_x00BA__x0020_Fiscal_x0020_Contribuinte_Trab>
  <Cenario>1.1</Cenario>
  <Nome_x0020_Completo>Guilhermina Cunha Santos</Nome_x0020_Completo>
  <Cod_categoria_x0028_interno_x0029_>24
       </Cod_categoria_x0028_interno_x0029_>
  <Desc_categoria_x0028_interno_x0029_>ESCRITURARIO SUPERIOR
       </Desc_categoria_x0028_interno_x0029_>
  <Especie_Tipo>CP</Especie_Tipo>
  <Sexo>F</Sexo>
  <Cod_Nacionalidade>201</Cod_Nacionalidade>
  <Cod_Distrito>01</Cod_Distrito>
  <Cod_Concelho>0601</Cod_Concelho>
  <Cod_Freguesia>090601</Cod_Freguesia>
  <Data_x0020_Nascimento>1957-06-18T00:00:00</Data_x0020_Nascimento>
  <Cod_x0020_Relação_x0020_Jurídica>12</Cod_x0020_Relação_x0020_Jurídica>
  ...
 </T_Bdap>
 ...
 <T_Bdap>...</T_Bdap>
</dataroot>
```

Para os cerca de 6200 funcionários inseridos até ao momento, este ficheiro XML ficou com 22MB, o que representa um tamanho considerável para um documento XML. A manipulação deste ficheiro com o XMLSpy[2] é bastante morosa devido à identificação com cores que o programa faz dos elementos e dos atributos, mas é possível.

Repare-se que o facto de os vários campos de um registo serem colocados todos como irmãos e filhos de um mesmo elemento simplifica significativamente as transformações a efectuar sobre este documento para gerar o definitivo, i.é., a 2^a fase que se apresenta na secção seguinte.

[2] http://www.xmlspy.com

4 Transformação

No caso da 2^a fase da transformação dos dados recolhidos para o XML pedido para a BDAP (o definido na secção 2), foi necessário fazer uma folha de estilos XSL [7,8]. Esta folha de estilos permite gerar o XML de destino a partir do obtido na 1^a fase da transformação.

Na construção da folha de estilos houve necessidade de fazer duas abordagens diferentes, consoante os elementos fossem ou não obrigatórios. Assim, a transformação dos elementos obrigatórios seguiu um esquema como exemplificado abaixo:

```
<sub_grupo><xsl:value-of select="Profissao_Sub_Grupo"/></sub_grupo>

<data_inicio>
   <xsl:value-of select="substring(Prof_Data_Inicio_Carreira,1,10)"/>
</data_inicio>

<remuneracao_base_euros>
  <xsl:value-of select="translate(Remuneração_x0020_Base,'.',',')"/>
</remuneracao_base_euros>
```

ou seja, limitámos-nos a escrever para cada um dos elementos de destino o valor original correspondente. No entanto no caso das datas foi necessário usar a função *substring* para retirar apenas os primeiros dez caracteres (uma vez que o campo no Microsoft Access é do tipo data e hora) e no caso dos valores monetários foi necessário usar a função *translate* para traduzir o separador da parte decimal de "." para ",".

No caso dos elementos opcionais foi necessário avaliar, para cada um deles, se o elemento de origem estava ou não preenchido e, caso afirmativo, escrever o elemento de destino com o valor desse elemento. Assim, a transformação seguiu um esquema semelhante ao seguinte:

```
<xsl:if test="string-length(Data_x0020_Saída)&gt;=1">
  <data_cessacao>
     <xsl:value-of select="substring(Data_x0020_Saída,1,10)"/>
  </data_cessacao>
</xsl:if>

<xsl:if test="string-length(Cod_x0020_Suplemento_1)&gt;=1">
   <suplementos_regulares>
     <codigo><xsl:value-of select="Cod_x0020_Suplemento_1"/></codigo>
     <valor_euros>
        <xsl:value-of select="translate(Valor_Suplemento_1,'.',',')"/>
     </valor_euros>
     <ano><xsl:value-of select="Ano_Supl_1"/></ano>
     <mes><xsl:value-of select="Mês_Supl_1"/></mes>
   </suplementos_regulares>
</xsl:if>
```

Nesta 2^a fase, a transformação, na linha de comandos e usando o saxon[3], é feita em 12 segundos, gerando um ficheiro com 14MB. O facto de a transformação exigir poucos cálculos pode justificar esta rapidez.

Para verificar a validade do ficheiro XML final contra o XML-Schema apresentado na secção 2 usou-se o XMLSpy. Esta validação demorou cerca de 1 minuto e meio, evidenciando mais uma vez que os documentos XML grandes são tratáveis.

5 Validação

Como se disse anteriormente, a introdução de dados no Microsoft Access não é objecto de qualquer validação em tempo real, tornando necessário efectuar validações sobre o documento XML.

Houve duas razões para procurar uma alternativa ao XML-Schema para a validação: a validação dos dados contra o XML-Schema pára a cada erro, e era necessário gerar um relatório amigável em HTML com os vários erros para que a DGRN os pudesse corrigir por uma questão de celeridade; não é possível fazer certas validações com um XML-Schema [9].

A solução inicialmente encontrada foi utilizar o XCSL [10,11], uma linguagem de especificação de restrições sobre documentos XML que permite validar todos os tipos de condições conhecidas [12]. Escreveu-se então o documento XCSL para algumas validações, por exemplo:

```
<constraint>
  <selector selexp="//trabalhador"/>
  <let name="chavetrab" value="@num_fiscal"></let>
  <cc>count(//trabalhador[@num_fiscal=$chavetrab])=1</cc>
  <action><message>
       <value selexp="@num_fiscal"/>
      (<value selexp="count(//trabalhador[@num_fiscal=$chavetrab])"/>
      iguais).
  </message></action>
</constraint>
```

que permite avaliar se há algum número de contribuinte fiscal utilizado para mais do que um funcionário, ou seja, encontrar os trabalhadores repetidos. Conforme se pode ver na caixa inferior direita da figura 1, este documento de restrições é transformado por um processador específico resultando num documento XSL. Este último é depois aplicado ao documento XML final (resultante da 2^a fase de transformação) para obter o relatório de erros em XML.

No entanto a utilização desta ferramenta não permitia separar os erros encontrados por serviço e, para cada serviço, por trabalhador, separação esta que facilitaria substancialmente a correcção dos erros. Este facto fez com que posteriormente se optasse por fazer um documento XSL de raiz para as validações sobre

[3] http://users.iclway.co.uk/mhkay/saxon

o documento XML original (obtido na 1^a fase de transformação), deixando em XCSL apenas a validação apresentada como exemplo. A validação do documento XML final contra o documento XSL gerado pelo processador XCSL apenas com aquela restrição demorou 33 minutos.

Por forma a tornar o relatório HTML o mais pequeno possível e apenas com a informação necessária (não interessa que sejam listados os serviços nem os trabalhadores para os quais não sejam detectados erros), a geração do ficheiro HTML foi feita em três fases (de acordo com a caixa inferior esquerda da figura 1):

1. Documento XSL para gerar documento XML intermédio com todos os elementos (mesmo os sem conteúdo) e por ordem de serviço. A seguir apresenta-se um extracto dessa folha de estilos:

```
...
<xsl:template match="dataroot">
 ...
 <xsl:variable name="servico2" select="//Cod_Serviço_BDAP">
 </xsl:variable>
 <xsl:for-each select="T_Bdap[Cod_Serviço_BDAP=$servico2]">
  <xsl:sort order="ascending"/>
  <xsl:apply-templates select="."/>
 </xsl:for-each>
 <xsl:text disable-output-escaping="yes">&lt;/dataroot></xsl:text>
</xsl:template>
...
<xsl:template match="T_Bdap">
 <T_Bdap>
  <Cod_Serviço_x0028_interno_x0029_>
   <xsl:apply-templates select="Cod_Serviço_x0028_interno_x0029_"/>
  </Cod_Serviço_x0028_interno_x0029_>
 ...
```

   Esta fase do processamento é muito rápida, demorando apenas 43 segundos e gerando um ficheiro com 33MB. É de notar que, ao contrário do que acontecia na 2^a fase da geração do XML para envio, não são usadas quaisquer funções nem nenhum elemento *xsl:if*.

2. Documento XSL para gerar um ficheiro XML com elemento raiz *tudo* e um elemento *servico* para cada servico. *servico* vai conter os sub-elementos: *nome* — o nome do serviço; *cod* — o código do serviço; *trabalhador* — elemento para cada trabalhador daquele serviço, independentemente de haver ou não erros de preenchimento relacionados com esse trabalhador. *trabalhador* tem por sua vez um sub-elemento *nome* e tantos sub-elementos *li* quantos os erros encontrados para esse trabalhador. A seguir apresenta-se um extracto dessa folha de estilos:

```
<xsl:template match="dataroot">
 <tudo><xsl:apply-templates select="T_Bdap" mode="m1"/></tudo>
</xsl:template>
<xsl:template match="T_Bdap" mode="m1">
 <xsl:if test="position()=1">
  <xsl:text disable-output-escaping="yes">&lt;servico></xsl:text>
  <nome><xsl:value-of select="Serviço"/></nome>
  <cod>
   <xsl:value-of select="Cod_Serviço_x0028_interno_x0029_"/>
  </cod>
 </xsl:if>
 ...
 <xsl:if test="not(Indicador_Nao_Envio='s' or
       Indicador_Nao_Envio='S')">
  <xsl:apply-templates select="." mode="m2"/>
 </xsl:if>
 ...
</xsl:template>
<xsl:template match="T_Bdap" mode="m2">
 <trabalhador>
  <nome> ... </nome>
  <xsl:if test="string-length(Cod_Local_trabalho)&lt;1">
   <li>Falta preencher o campo "Cod_Local_trabalho"</li>
  </xsl:if>
  ...
  <xsl:if test="(HabiIitações_x0028_1_x0020_a_x0020_9_x0029_=7 or
   HabiIitações_x0028_1_x0020_a_x0020_9_x0029_=8 or
   ...">
   <li>O campo "Área_Temática_(1_a_9)" tem que ser preenchido
   já que o campo "HabiIitações_(1_a_9)" vale
   <xsl:value-of select="
       HabiIitações_x0028_1_x0020_a_x0020_9_x0029_"/>.</li>
  </xsl:if>
  ...
  <xsl:variable name="dreft" select="concat(substring($dref,1,4),
  substring($dref,6,2),substring($dref,9,2))"/>
  <xsl:variable name="mesg" select="concat('deve valer entre ',
       $dnmd,' e ',$dref,'.')"/>
  <xsl:if test="not(
   (concat(substring(Prof_Data_Inicio_Carreira,1,4),
   substring(Prof_Data_Inicio_Carreira,6,2),
   substring(Prof_Data_Inicio_Carreira,9,2)) &gt;= $dnasct + $ia) and
   ...">
   <li>Prof_Data_Inicio_Carreira vale
    <xsl:value-of select="substring(Prof_Data_Inicio_Carreira,1,10)"/>
    e <xsl:value-of select="$mesg"/>
   </li>
  </xsl:if>
  ...
```

Esta fase do processamento demora 1 minuto e 5 segundos, gerando um documento cujo tamanho depende do número de erros encontrados. A diferença de tempo face à fase anterior tem a ver com o facto de neste caso se usarem várias funções e elementos *xsl:if*. Segue-se um elemento *trabalhador* resultado:

```
<trabalhador>
  <nome>MARIA CELESTE BARROSO DA GRAÇA BOSQUET - 142614785</nome>
  <li>Valor_Suplemento_1 vale 44038 e tem que valer entre 0 e 4999.</li>
  <li>Cod_Relação_Jurídica não pode valer 12 para o cenário 1.1.</li>
</trabalhador>
```

3. Documento XSL para gerar um ficheiro HTML com a enumeração dos vários serviços para os quais foram detectados erros, descrevendo os trabalhadores com erros de preenchimento e quais são esses erros. A seguir apresenta-se um extracto dessa folha de estilos:

```
<xsl:template match="/">
  <html>
    ...
    <body><xsl:apply-templates select="tudo"/></body>
  </html>
</xsl:template>
<xsl:template match="tudo">
  <xsl:choose>
    <xsl:when test="count(//li)=0">
      <h3 align="center">
        <font color="green">
           Validação efectuada sem encontrar erros!
        </font>
      </h3>
    </xsl:when>
    <xsl:otherwise>
      <h3 align="center">
        <font color="green">
           Erros no preenchimento da Base de Dados
        </font>
      </h3>
      <xsl:apply-templates select="servico" mode="m1"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="servico" mode="m1">
 <xsl:if test="count(trabalhador[count(li)!=0])!=0">
   <xsl:if test="position()=1">
     <hr color="#800000"/>
     <font color="blue" size="4">
        <xsl:value-of select="nome"/> (<xsl:value-of select="cod"/>)
     </font>
     <br/><br/>
```

```
    </xsl:if>
            ...
    <xsl:apply-templates select="trabalhador" mode="m2"/>
  </xsl:if>
 </xsl:template>
 <xsl:template match="trabalhador" mode="m2">
    ...
    - <xsl:value-of select="."/>
 </xsl:template>
```

Nesta fase, o tempo do processamento depende obviamente do tamanho do ficheiro obtido na fase anterior (do número de erros). Para o caso em que aquele tem aproximadamente 1MB demora menos de dois segundos. A figura 2 apresenta um ficheiro HTML resultado.

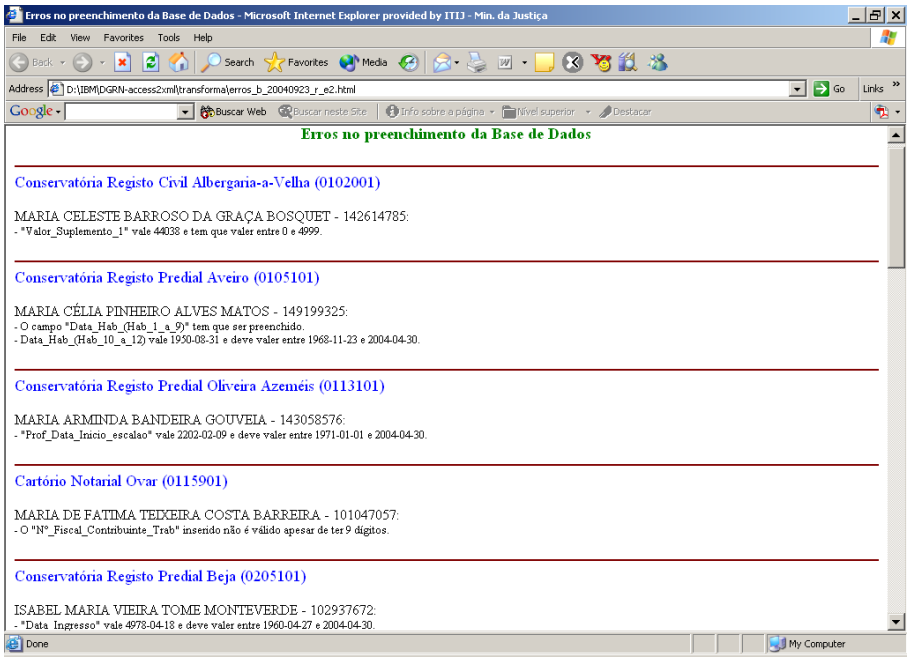

Figura 2. Documento HTML com os erros de validação

Quando os ficheiros XML submetidos a todas aquelas validações foram enviados para o Instituto de Informática, este organismo devolveu uma lista com erros que não constavam dos vários documentos disponibilizados. As novas validações foram incluídas no XML-Schema quando possível e no documento XSL da segunda fase do processo explicado anteriormente.

6 Conclusão

A necessidade de efectuar várias transformações sobre documentos XML de tamanho não académico trouxe a oportunidade de fazer uma avaliação cuidada do tempo desse processamento numa situação real. Verificou-se que a transformação sobre documentos XML grandes (22 MB) é praticamente imediata quando é simples, ou seja, não envolve muitas condições. Quando se aumenta a complexidade da transformação (no caso para analisar a existência de erros), o tempo de processamento aumenta substancialmente, não sendo contudo proibitivo. A única situação em que de facto o processamento de um documento XML é consideravelmente demorado é aquela em que se analisa os elementos repetidos.

Conclui-se então que os documentos XML grandes não são monstros e são processáveis em tempo real.

Referências

1. Instituto de Informática e DGAP: Glossário BDAP (2003)
2. Instituto de Informática: Instruções sobre o Ficheiro XML da BDAP (2003)
3. Instituto de Informática: Código fonte do ficheiro XML. (2003)
4. Duckett, J., Griffin, O., Mohr, S., Norton, F., Ozu, N., Stokes-Rees, I., Tennison, J., Williams, K., Cagle, K.: Professional XML Schemas. Wrox Press (2001)
5. Eric van der Vlist: XML Schema. O'Reilly & Associates, Inc. (2002)
6. Instituto de Informática e DGAP: Regras e definições para preenchimento dos dados (2003)
7. Bradley, Neil: The XSL companion. Addison-Wesley (2000)
8. K. Cagle and M. Corning and J. Diamond and T. Duynstee and O. Gudmundsson and J. Jirat and M. Mason and J. Pinnock and P. Spencer and J. Tang and P. Tchistopolskii and J. Tennison and A. Watt: Professional XSL. Wrox Press (2001)
9. Marta Henriques Jacinto, Giovani Rubert Librelotto, José Carlos Ramalho, Pedro Rangel Henriques: Constraint Specification Languages: comparing XCSL, Schematron and XML-Schema. Actas do XML Europe'02, Barcelona, Espanha (2002)
10. José Carlos Leite Ramalho:Anotação Estrutural de Documentos e sua Semântica. Universidade do Minho (2000)
11. Marta H. Jacinto, Giovani R. Librelotto, José C. Ramalho, Pedro R. Henriques: XCSL: XML Constraint Specification Language. Actas da conferência CLEI02 - XXVIII Conferencia Latino Americana de Informática, Montevideo, Uruguai (2002)
12. Marta Henriques Jacinto:Validação Semântica em Documentos XML. Universidade do Minho (2002)