Instituto Superior de Engenharia

Politécnico de Coimbra

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

Integração de Dados com XML
Integração de Dados

Relatório de Licenciatura

Autores

Ana Rita Conceição Pessoa – 2023112690

David José Gomes de Matos – 2023143725

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Coimbra, abril de 2025

1 ÍNDICE

1.1 Índice de texto

1.2 Índice de figuras

1.3 Índice de tabelas

2 INTRODUÇÃO

O objetivo deste trabalho consiste na criação de uma aplicação integradora desenvolvida em linguagem Java, composta por diversos **Wrappers**, que apresenta uma vista unificada de informação relativa a diferentes países.

A informação é recolhida de duas fontes de dados distintas: **Wikipedia** e **DB-City**, ambas heterogéneas, distribuídas e autónomas. Após a recolha, os dados são tratados com expressões regulares e armazenados num ficheiro XML designado por **paises.xml**.

A aplicação permite ao utilizador visualizar, pesquisar, adicionar, editar e eliminar países do ficheiro XML, garantindo sempre a validação do mesmo através de esquemas DTD e XSD. Também permite executar consultas XPath e XQuery sobre os dados armazenados.

Além disso, é possível gerar ficheiros de output em diversos formatos (HTML, TXT e XML) recorrendo a transformações XSLT e XQuery, com o objetivo de apresentar a informação de forma personalizada e acessível. A aplicação inclui ainda uma interface gráfica intuitiva, que permite realizar todas estas operações de forma simples e funcional.

3 ANÁLISE DAS FONTES DE DADOS

Neste trabalho, foram utilizadas duas fontes principais de informação: o site da **Wikipedia em português** (https://pt.wikipedia.org/wiki/) e o site **DB-City** (https://pt.db-city.com/).

3.1 Wikipedia

A Wikipedia foi utilizada para recolher:

- **Nome do país**
- **Link para a imagem da bandeira do país**

Embora a Wikipedia contenha muitos outros dados úteis, optou-se por usá-la apenas para estes dois atributos devido à sua estrutura HTML ser bastante variável de país para país, o que dificultaria a automatização da recolha de dados com expressões regulares.

3.2 DB-City

A maior parte dos dados foram extraídos do site DB-City, uma vez que apresenta uma estrutura de HTML mais estável e organizada, permitindo assim a criação de wrappers eficazes. Esta fonte forneceu os seguintes dados:

- **Capital do país**
- **Continente**
- **Língua(s) oficial(ais)**
- **Área do país (em km²)**
- **População**
- **Densidade populacional**
- **Presidente/Chefe de Estado**
- **Religiões**
- **Cidades importantes**
- **Países de fronteira**
- **Nº de casos COVID registados**

4 DEFINIÇÃO DO ESQUEMA GLOBAL (G)

Para garantir a integração coerente da informação proveniente das duas fontes de dados (Wikipedia e DB-City), foi definido um **modelo global** que representa os principais atributos associados a cada país.

4.1 Estrutura Hierárquica

A raíz do ficheiro XML é o elemento <paises>, que contém múltiplos elementos <pais>, um por cada país registado. Cada <pais> inclui elementos simples e estruturas aninhadas para listas de dados.

```xml
<paises>
  <pais>
    <nome>Portugal</nome>
    <capital>Lisboa</capital>
    <continente>Europa</continente>
    <bandeira>https://upload.wikimedia.org/flag_of_Portugal.svg</bandeira>
    <area>92090</area>
    <populacao>10310000</populacao>
    <densidade>111.3</densidade>
    <chefeEstado>Marcelo Rebelo de Sousa</chefeEstado>
    <covid>1234567</covid>

    <linguas>
      <lingua>Português</lingua>
    </linguas>

    <religioes>
      <religiao>Catolicismo</religiao>
      <religiao>Protestantismo</religiao>
    </religioes>

    <cidades>
      <cidade>Lisboa</cidade>
      <cidade>Porto</cidade>
    </cidades>

    <fronteiras>
      <fronteira>Espanha</fronteira>
    </fronteiras>
  </pais>
</paises>
```

Figura 1: Estrutura XML de exemplo para o país Portugal no ficheiro paises.xml

4.2 Justificação da Estrutura

A estrutura foi feita para representar corretamente os dados obtidos das duas fontes. Usámos **elementos simples** para dados únicos, como a capital ou a área, e **elementos compostos** para listas, como línguas, religiões, cidades e fronteiras.

Os tipos de dados foram tratados e convertidos apropriadamente:

- **Área, população, densidade e casos COVID** são guardados como valores numéricos (double ou long).
- Os **nomes e listas** são tratados como texto limpo, com remoção de tags HTML e caracteres especiais.

4.3 Validação do Esquema

Para garantir que o ficheiro XML gerado respeita a estrutura definida, foram criadas duas definições de validação:

- **DTD (Document Type Definition)**: valida a existência e ordem dos elementos.
- **XSD (XML Schema Definition)**: valida também os tipos de dados (por exemplo, garantir que a população é um número inteiro positivo).

Ambos os ficheiros foram usados para validar automaticamente os documentos XML produzidos pela aplicação. A validação é realizada durante a execução, recorrendo às funções implementadas em Java nos ficheiros `**ValidarXML.java**` (para XSD) e `**JDOMFunctions_Validar.java**` (para DTD).

Cada vez que o utilizador adiciona, edita ou elimina um país, a aplicação invoca a validação dos dados com base nos esquemas definidos, assegurando a conformidade com o modelo global.

5 IMPLEMENTAÇÃO DOS WRAPPERS (M)

A extração dos dados foi realizada através de **wrappers desenvolvidos em Java**, que analisam ficheiros HTML descarregados previamente através da classe HttpRequestFunctions. A leitura é feita linha a linha, usando expressões regulares para localizar e extrair a informação necessária.

Cada wrapper foi criado como uma função autónoma, responsável por extrair um campo específico do modelo. As expressões regulares foram ajustadas à estrutura HTML de cada fonte.

5.1 Lógica de Aquisição de Dados

Antes da aplicação das expressões regulares, o HTML da página de cada país é obtido automaticamente e guardado num ficheiro local:

```
HttpRequestFunctions.httpRequest1("https://pt.wikipedia.org/wiki/", pais, pais + "_wiki.html");
```

Posteriormente, o wrapper lê o ficheiro linha a linha e aplica um padrão:

```
Pattern p = Pattern.compile(er);
...

while (ler.hasNextLine()) {
    linha = ler.nextLine();
    m = p.matcher(linha);
    while (m.find()) {
        ...
        }
    }
```

Este método é usado sempre que é preciso apanhar várias ocorrências numa só linha, como acontece com as cidades importantes ou as fronteiras. Assim, evita-se voltar a compilar o padrão em cada ciclo, o que torna o código mais limpo e eficiente.

5.2 Wrappers Implementados

A tabela seguinte apresenta os campos extraídos, as funções wrapper implementadas para cada um e as respetivas fontes de dados utilizadas.

Tabela 1: Mapeamento dos campos recolhidos com as respetivas funções wrapper e fontes de dados utilizadas.

Campo	Função	Fonte
Nome do país	nomePais()	Wikipedia
Capital	capital()	DB-City
Continente	continente()	DB-City
Link da bandeira	bandeira()	Wikipedia
Línguas oficiais	linguas()	DB-City
Área (km²)	area()	DB-City
População	populacao()	DB-City
Densidade populacional	densidade()	DB-City
Chefe de Estado	chefeEstado()	DB-City
Religiões	religioes()	DB-City
Cidades importantes	cidadesImportantes()	DB-City
Países de fronteira	fronteiras()	DB-City
Casos COVID confirmados	covidConfirmados()	DB-City

5.3 Tratamento e Normalização

Foram implementadas várias medidas de normalização dos dados:

- **Remoção de pontos nos milhares**
    - Muitos números vinham formatados com pontos, como por exemplo "1.234.567". Para possibilitar cálculos e comparações numéricas, os pontos foram removidos, transformando o valor em "1234567". Esta substituição foi feita com **.replace(".", "").**
- **Conversão de vírgula para ponto decimal**
    - Números com casas decimais, como densidade populacional ("111,3"), foram convertidos para o formato com ponto ("111.3"), usando **.replace(",", ".")**, que é o formato reconhecido por parsers XML/XQuery.
- **Remoção de espaços codificados HTML como **
    - Alguns dados continham espaços não visíveis ou codificados (ex:). Estes foram removidos com **.replaceAll(" ", "")** e

.replaceAll("\\s+", ""), assegurando que o conteúdo extraído é limpo e sem ruído.

- **Tratamento de listas com marcação HTML**
  - Quando listas vinham separadas por
 (como no caso das línguas oficiais), estas tags foram substituídas por um separador customizado (|) e, posteriormente, os valores foram divididos com **.split("\\|")**. Isto permitiu gerar listas estruturadas no XML (com elementos como <lingua>Português</lingua>).
- **Remoção de tags HTML em texto**
  - Em conteúdos onde poderiam existir tags dentro de células HTML (por exemplo Texto), foi feita a remoção dessas tags com expressões como **.replaceAll("<[^>]+>", "")**.
- **Trim e limpeza final**
  - Todos os valores extraídos foram submetidos a **.trim()** para remover espaços iniciais e finais e garantir que os dados inseridos no XML não contêm espaços indesejados.

5.4 Gestão de Exceções

Durante a recolha e tratamento da informação das diferentes fontes (Wikipedia e DB-City), foi necessário implementar mecanismos que assegurassem o funcionamento contínuo da aplicação, mesmo quando certos dados estavam ausentes ou não respeitavam o formato esperado.

- **Dados em falta ou com formato inesperado:** Quando certos dados não estão presentes ou surgem com formatação inconsistente, as funções retornam null ou listas vazias, evitando erros durante o tratamento da informação.
- **Comportamento na criação de objetos Pais:** Durante a criação do objeto Pais, valores ausentes são substituídos por valores por defeito (como strings vazias, zero ou listas vazias). Isto assegura que o ficheiro XML gerado tem sempre todos os campos esperados, mesmo que alguns estejam vazios.
- **Continuidade da execução da aplicação:** Mesmo com dados incompletos, a aplicação continua a funcionar normalmente, registando os países com a informação disponível e evitando falhas na execução.

6 CONSTRUÇÃO DO FICHEIRO XML

Após a recolha dos dados através dos wrappers, o passo seguinte consiste na **construção do ficheiro XML final**, que representa todos os países recolhidos segundo o modelo definido.

6.1 Criação do Objeto País

A função **criarObjetoPais(String nome)**, definida no ficheiro **Trabalho_De_ID.java**, centraliza toda a lógica de integração. Esta função atua como ponte entre os dados extraídos e a estrutura interna do programa.

A sua responsabilidade é:

- Receber o nome do país como parâmetro;
- Invocar todos os métodos de extração (**Wrappers.java**) para recolher os diferentes atributos;
- Criar uma instância da classe **Pais** (definida em **Pais.java**) e preenchê-la com os dados recolhidos;
- Validar os dados e aplicar verificações de **null**, conversões seguras e valores por defeito para garantir que a aplicação continua funcional mesmo com dados em falta.

A seguir, o objeto **Pais** criado pode ser usado diretamente para gerar o conteúdo XML do ficheiro **paises.xml.**

Exemplo de chamada:

```
Pais portugal = criarObjetoPais("Portugal");
```

6.2 Inserção no Documento XML

Depois de criado o objeto **Pais**, este é inserido no documento XML através da função **ModeloPaisesXML.adicionaPais(...)**, localizada no ficheiro **ModeloPaisesXML.java**.

Esta função é responsável por:

- **Verificar se o ficheiro XML já existe:**
  - Caso não exista, cria um novo documento com a raiz <paises>.
  - Caso exista, carrega o ficheiro atual e atualiza-o.

- **Garantir a não duplicação de dados:**
  - Verifica se já existe um elemento <pais> com o mesmo nome.
  - Se o país já existir, a função ignora a adição.
- **Criar a estrutura XML para o país:**
  - Cada campo do objeto **Pais** é convertido num subelemento (<nome>, <capital>, <area>, etc.).
  - As **listas** (como <linguas>, <religioes>, <cidades_importantes> e <fronteiras>) são representadas com elementos aninhados, assegurando a coerência com o modelo global.

Adicionar o novo país ao elemento raiz <paises> do documento.

```
doc = ModeloPaisesXML.adicionaPais(portugal, doc);
```

6.3 Escrita para Ficheiro

Após a inserção do país no documento XML, o conteúdo é gravado no disco através da função **XMLJDomFunctions.escreverDocumentoParaFicheiro(...)**, definida no ficheiro **XMLJDomFunctions.java**.

Esta função utiliza a API JDOM2 para:

- Serializar o objeto **Document** contendo todos os países;
- Aplicar uma formatação legível ao XML utilizando **Format.getPrettyFormat()**, com indentação e quebras de linha apropriadas;
- Escrever o conteúdo formatado num ficheiro chamado **paises.xml**.

Este passo é fundamental para persistir os dados e garantir que todas as alterações feitas à estrutura em memória ficam refletidas no sistema de ficheiros.

```
XMLJDomFunctions.escreverDocumentoParaFicheiro(doc, "paises.xml");
```

6.4 Estrutura Gerada

O ficheiro **paises.xml** gerado pela aplicação segue integralmente o modelo definido na **secção 4**.

A estrutura inclui:

- **Elementos simples**: utilizados para representar dados unitários como <nome>, <capital>, <area>, <habitantes>, <densidade> ou <chefe_estado>, armazenando valores textuais ou numéricos diretamente.
- **Elementos compostos com listas aninhadas**: utilizados para representar coleções de dados, como:
    - <linguas> contendo múltiplos elementos <lingua>;
    - <religioes> com vários <religiao>;
    - <cidades_importantes> com elementos <cidade>;
    - <fronteiras> com múltiplos <vizinho>.
- **Suporte para múltiplos países**: todos os elementos <pais> são agrupados dentro da raiz <paises>, permitindo o armazenamento de informação sobre diversos países num único ficheiro XML.

Esta estrutura é flexível e pode ser facilmente expandida com novos países ou atributos. O ficheiro gerado suporta consultas XPath, transformações XSLT/XQuery e validação com DTD ou XSD.

7 VALIDAÇÃO DOS FICHEIROS XML

Para garantir que o ficheiro XML gerado segue rigorosamente a estrutura definida no modelo global, foi implementado um sistema de validação baseado em **DTD (Document Type Definition)**.

A validação serve para:

- Garantir que a hierarquia e ordem dos elementos são corretas;
- Detetar erros de estrutura ou elementos inválidos;
- Assegurar a compatibilidade com outras ferramentas que requerem XML bem formado e validado.

7.1 Validação com DTD

A validação estrutural do ficheiro **paises.xml** com recurso a **DTD** é realizada em duas fases distintas:

1. Associação do DTD ao XML

A função **validarDocumentoDTD(...)**, localizada no ficheiro **ValidarXML.java**, é responsável por associar o ficheiro **.dtd** ao documento XML. Esta função:

- Lê o documento XML existente com **XMLJDomFunctions.lerDocumentoXML(...)**;
- Cria um objeto **DocType** com o nome da raiz (<paises>) e o caminho do ficheiro **.dtd**;
- Define esse **DocType** no documento;
- Reescreve o XML no disco com a declaração <!DOCTYPE> no topo, através de **XMLJDomFunctions.escreverDocumentoParaFicheiro(...)**.

2. Validação estrutural do documento

Após a associação do DTD, o ficheiro é validado estruturalmente através da função **validarDTD(...)**, definida em **JDOMFunctions_Validar.java**. Esta função:

- Utiliza um **SAXBuilder** com validação ativada (new SAXBuilder(true)), que ativa a verificação DTD;
- Tenta construir o documento a partir do ficheiro XML;
- Em caso de erro de estrutura, apresenta uma mensagem de erro detalhada no terminal;
- Caso o ficheiro esteja conforme o DTD, devolve um objeto Document válido.

7.2 Resultados do Processo

A função **validarDocumentoDTD(...)**, presente na classe **ValidarXML**, devolve um valor booleano indicando o sucesso ou falha da validação:

- **true** – se a validação for bem-sucedida e o ficheiro XML estiver conforme o DTD;
- **false** – se ocorrer algum dos seguintes casos:
    - O ficheiro XML ou DTD não existe;
    - A estrutura do XML não cumpre as regras definidas no DTD.

7.3 Validação com XSD

Para além da validação estrutural com DTD, foi implementado um segundo mecanismo de validação utilizando **XSD** (XML Schema Definition), com o objetivo de verificar a conformidade dos tipos de dados definidos no modelo XML.

A validação com XSD é realizada através da função **validarDocumentoXSD(...)**, localizada no ficheiro **ValidarXML.java**. Esta função:

- Cria um objeto **Schema** a partir do ficheiro **.xsd** indicado;
- Utiliza a classe **Validator** da API javax.xml.validation para validar o ficheiro XML;
- Apresenta no terminal o resultado da verificação, indicando se o ficheiro é válido ou se ocorreram erros.

Este tipo de validação permite, por exemplo:

- Confirmar que elementos como <habitantes>, <area> ou <densidade> contêm valores numéricos válidos;
- Garantir que listas seguem o tipo correto de repetição;
- Reforçar a interoperabilidade do XML com outras aplicações ou ferramentas que dependem de tipos de dados bem definidos.

Com esta abordagem complementar à DTD, é possível garantir não só a **estrutura** correta do XML, mas também a **validade** e **coerência** dos dados inseridos.

8 INTERFACE DA APLICAÇÃO

A aplicação desenvolvida inclui uma interface gráfica intuitiva, que permite ao utilizador interagir facilmente com todas as funcionalidades do sistema. A interface está organizada em menus e botões, oferecendo um conjunto completo de operações sobre o ficheiro **paises.xml**.

A janela principal está dividida em quatro menus principais: **Geral**, **Gestão de Países**, **Pesquisar** e **Transformações**, conforme ilustrado nas imagens seguintes.

- **Menu "Geral"**

  Permite visualizar o conteúdo atual do ficheiro paises.xml e realizar a validação do mesmo (com DTD e XSD). Inclui ainda a opção para sair da aplicação.

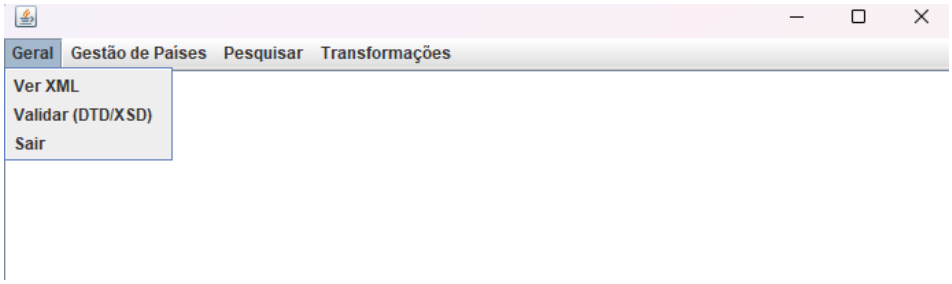

Figura 2: Menu "Geral" aberto

- **Menu "Gestão de Países"**

  Contém as operações fundamentais sobre o XML:

  - Adicionar um novo país (através dos wrappers);
  - Eliminar um país existente;
  - Editar atributos específicos de um país (como população, área ou chefe de estado).

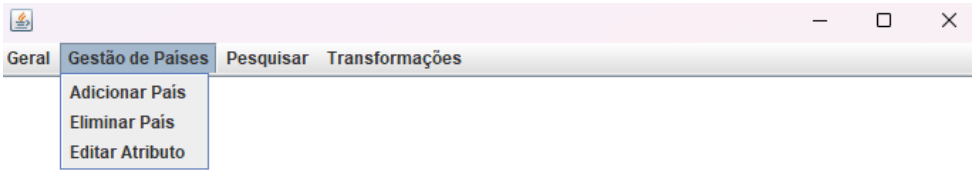

Figura 3: Menu "Gestão de Países" aberto

- **Menu "Pesquisar"**

  Permite realizar diversas pesquisas com base no conteúdo XML, utilizando expressões XPath:

  - Por nome de país;
  - Por população;
  - Por religião;
  - Por capital;
  - Por número de casos de COVID;
  - Por número de cidades importantes.

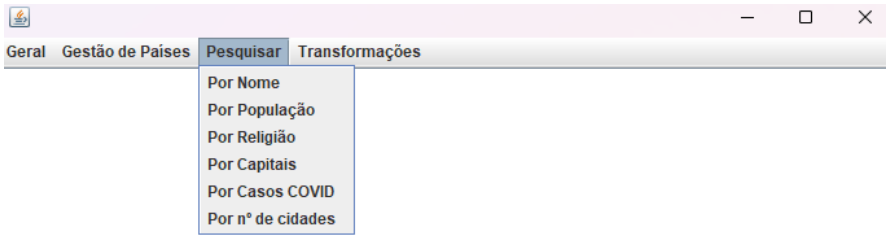

Figura 4: Menu "Pesquisar" aberto

- **Menu "Transformações"**

  Dá acesso às funcionalidades de exportação dos dados, usando XSLT e XQuery:

  - **HTML Bandeiras:** gera uma página HTML com as bandeiras e nomes dos países;
  - **TXT Cidades:** cria um ficheiro de texto com as cidades importantes de um país selecionado;
  - **XML Mais Populosos:** produz um ficheiro XML com os cinco países mais populosos;
  - **HTML Países Fronteira:** apresenta uma lista dos países que fazem fronteira com um país indicado;
  - **HTML Continente:** lista os países agrupados por continente;
  - **HTML Ranking Densidade Populacional:** ordena e apresenta os países por densidade populacional;
  - **TXT Línguas:** gera um ficheiro de texto com as línguas oficiais de cada país.

Figura 5: Menu "Transformações" aberto

A interface permite, assim, realizar todo o ciclo de integração de dados de forma assistida: desde a recolha e estruturação, até à consulta, edição, validação e exportação. A estrutura por menus facilita a navegação e torna a aplicação intuitiva para qualquer utilizador.

9 TRANSFORMAÇÕES E CONSULTAS

9.1 Consultas XPath

A aplicação permite realizar diversas consultas ao ficheiro **paises.xml** utilizando **XPath**. Estas pesquisas são executadas através de métodos implementados na classe **XPathFunctions.java**.

Tabela 2: Consultas XPath implementadas na aplicação

Tipo de Pesquisa	Expressão XPath	Explicação
País por nome	//pais[nome='Portugal']	Seleciona o elemento <pais> cujo filho <nome> é "Portugal". Retorna toda a informação desse país.
Capitais	//capital	Retorna todos os elementos <capital> presentes no documento, independentemente da hierarquia.
Cidades de um país	//pais[nome='Espanha']/cidades/cidade	Acede ao país "Espanha" e retorna a lista de cidades importantes listadas dentro do elemento <cidades>.
População superior a X	//pais[populacao > 10000000]/nome	Retorna os nomes dos países cuja população é superior a 10 milhões.
Países por religião	//pais[religioes/religiao='Cristianismo']/nome	Retorna os nomes dos países que incluem "Cristianismo" na lista de religiões.
Casos de COVID entre dois valores	//pais[covid >= 50000 and covid <= 500000]/nome	Retorna países com número de casos de COVID entre 50.000 e 500.000.
Países por continente	//pais[continente='África']/nome	Retorna os nomes dos países cujo elemento <continente> tem o valor "África".
Países com mais de X cidades importantes	//pais[count(cidades/cidade) > 5]/nome	Retorna os nomes dos países com mais de 5 cidades listadas dentro de <cidades>.

10 TRANSFORMAÇÕES COM XSLT E XQUERY

Para além da visualização e pesquisa direta dos dados no ficheiro XML, a aplicação permite gerar automaticamente ficheiros de saída em diversos formatos — **HTML, TXT** e **XML** — com base no conteúdo do modelo global (paises.xml). Estas transformações são realizadas através das linguagens **XSLT** e **XQuery**.

Todas as transformações estão acessíveis através da interface gráfica, no menu **Transformações**.

10.1 Bandeiras dos Países (XSLT → HTML)

Uma das transformações implementadas gera um ficheiro **bandeiras.html** com as bandeiras e nomes dos países do ficheiro **paises.xml**, utilizando XSLT. Através do ficheiro **bandeiras.xslt**, são percorridos os elementos <pais> e extraídas as bandeiras, garantindo que não há repetições com o uso de uma chave (<xsl:key>).

Figura 6: Ficheiro HTML gerado com as bandeiras dos países (bandeiras.html)

10.2 Cidades por País (XQuery → TXT)

A transformação **cidades_pais.xq** gera um ficheiro de texto com as cidades importantes de um país escolhido pelo utilizador. Usando **XQuery**, a aplicação localiza o elemento <pais>, extrai os elementos <cidade> e grava-os em **cidades.txt**, uma por linha.

```
<?xml version="1.0" encoding="UTF-8"?>Lisboa
Sintra
Vila Nova de Gaia
Porto
Cascais
Loures
Braga
Amadora
Oeiras
Matosinhos
Almada
Seixal
Gondomar
Odivelas
Guimarães
Vila Franca de Xira
Santa Maria Da Feira
Maia
Coimbra
Vila Nova de Famalicão
```

Figura 7: Exemplo de ficheiro TXT gerado com as cidades importantes de Portugal

10.3 Países por Continente (XSLT → HTML)

Esta transformação gera um ficheiro **HTML** com a listagem dos países organizados por continente. Utiliza a linguagem XSLT aplicada ao ficheiro **paises.xml**, com base no ficheiro **continente_paises.xslt**, para estruturar a informação de forma visual e agrupada.

Espanha Bélgica Portugal

Figura 8: Listagem de países organizados por continente (Europa)

10.4 Países por Continente (XSLT → HTML)

Esta transformação gera um ficheiro **HTML** com a lista dos países que fazem fronteira com um país selecionado pelo utilizador. Baseia-se no ficheiro **fronteiras_pais.xslt** e aplica-se ao documento **paises.xml**, apresentando o resultado de forma simples e clara.

Figura 9: Países que fazem fronteira com Portugal

10.5 Densidade populacional (XSLT → HTML)

Esta transformação gera uma página **HTML** com um ranking dos países por densidade populacional, ordenando os resultados do mais denso para o menos denso. Baseia-se no ficheiro **densidade_ranking.xslt.**

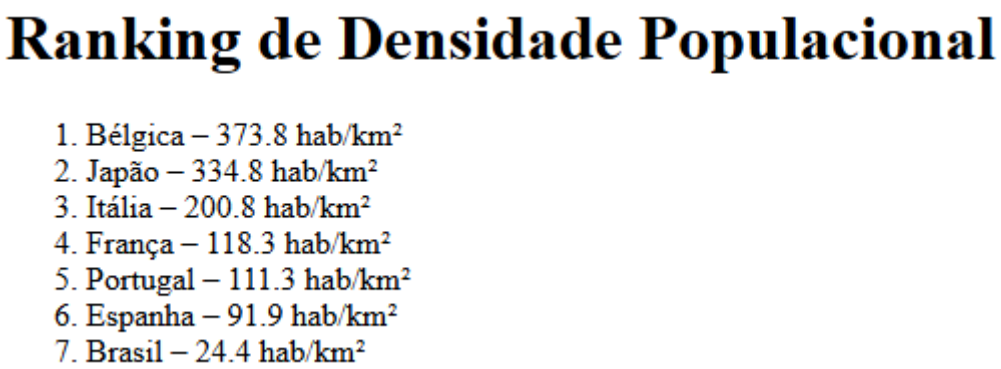

Figura 10: Ranking de países por densidade populacional

10.6 Línguas por país (XQuery → TXT/XML)

Esta transformação gera uma página **HTML** com um ranking dos países por densidade populacional, ordenando os resultados do mais denso para o menos denso. Baseia-se no ficheiro **densidade_ranking.xslt.**

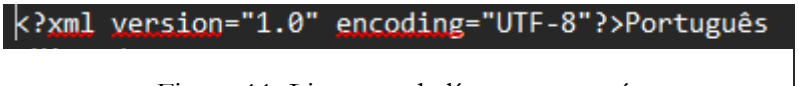

Figura 11: Listagem de línguas por país

11 CONCLUSÃO

Este trabalho permitiu aplicar, de forma prática, os conceitos fundamentais de integração de dados, focando-se na recolha, normalização, armazenamento e consulta de informação proveniente de fontes heterogéneas.

Através do uso de Java, expressões regulares, JDOM2, XSLT e XQuery, foi possível construir uma aplicação completa que:

- Extrai dados de páginas HTML da Wikipedia e DB-City;
- Organiza os dados num ficheiro XML bem estruturado e validado por DTD e XSD;
- Permite adicionar, editar, eliminar e pesquisar países através de uma interface gráfica intuitiva;
- Gera ficheiros de saída (HTML, TXT, XML) com base no conteúdo XML, usando transformações definidas.

A organização das funcionalidades por menus tornou a aplicação mais clara e fácil de usar. A validação automática dos dados ajuda a manter tudo consistente e sem erros. Além disso, com XPath e XQuery, é possível fazer pesquisas e extrair informação do ficheiro XML de forma bastante flexível.

O projeto foi também expandido com transformações extra, o que mostra a versatilidade da aplicação na forma como apresenta e exporta os dados.

Este trabalho ajudou a consolidar os conhecimentos sobre tecnologias XML e mostrou como podem ser usadas numa aplicação prática.

Instituto Superior de Engenharia
Politécnico de Coimbra