Integração Dados – Trabalho Prático 2024/2025

Tema: Países

Licenciaturas em Engenharia
Informática
ISEC
16 de Maio de 2025

Conteúdo

Introdução

O presente relatório documenta o desenvolvimento do Trabalho Prático de Integração de Dados, realizado no âmbito da unidade curricular de Integração de Dados, integrada no 2.º ano do curso de Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra (ISEC), no ano letivo de 2024/2025. Este projeto visa a conceção de uma aplicação em Java que integre dados provenientes de fontes heterogéneas, autónomas e distribuídas, especificamente as plataformas Wikipédia (https://pt.wikipedia.org/wiki/) e DB-City (https://pt.db-city.com/), organizando informação relativa a países num modelo global estruturado em XML.

A aplicação desenvolvida permite extrair, processar e unificar dados como o nome do país, capital, população, área, línguas oficiais, entre outros, proporcionando uma visão consolidada e coerente. Adicionalmente, disponibiliza funcionalidades como consultas XPath/XQuery, transformações XSLT/XQuery, validação de dados através de DTD/XSD e uma interface gráfica amigável para interação com o utilizador. O desenvolvimento recorreu a tecnologias como JDOM2, SAXON e expressões regulares, alinhando-se com os objetivos de aprendizagem da unidade curricular, nomeadamente a capacidade de analisar, modelar e manipular dados em cenários complexos de integração.

Este documento está organizado em secções que detalham o processo de desenvolvimento: análise das fontes de dados, definição do esquema global, implementação dos *wrappers*, manipulação do ficheiro XML, consultas, transformações, interface gráfica, conclusões e referências bibliográficas. Cada secção apresenta as decisões técnicas adotadas, respetivas justificações, exemplos práticos e reflexões sobre os desafios enfrentados, com o intuito de evidenciar a robustez, funcionalidade e adequação da solução proposta aos requisitos estabelecido

Análise das Fontes de Dados

Descrição das Fontes

Wikipédia: A Wikipédia é uma enciclopédia colaborativa que organiza informações em páginas HTML, com dados estruturados em *infoboxes* (tabelas laterais) e secções de texto. Para este trabalho, a Wikipédia foi utilizada exclusivamente para extrair a **densidade populacional**, o **link para a imagem da bandeira** e o **presidente/chefe de Estado**. Os *infoboxes* contêm campos específicos para densidade (ex.: "Densidade: 337 hab/km²" para o Japão) e chefe de Estado (ex.: "Imperador: Naruhito"), enquanto as bandeiras estão disponíveis como imagens com URLs acessíveis (ex.: //upload.wikimedia.org/.../Flag_of_Japan.svg). A escolha da Wikipédia para estes atributos justifica-se pela sua fiabilidade e detalhe, especialmente para dados qualitativos como o chefe de Estado.

DB-City: O DB-City é uma base de dados geográfica com informações estruturadas em tabelas e listas, sendo a fonte primária para os restantes atributos: capital, continente, línguas oficiais, área, população, religiões, cidades importantes, países fronteiriços e casos de COVID-19. A formatação consistente do DB-City facilita a extração de dados numéricos (ex.: "População: 1.395.380.000" para a China) e listas (ex.: cidades importantes), tornando-o ideal para atributos que requerem tratamento padronizado.

Estratégia de Extração

A extração foi realizada através de *wrappers* implementados em Java, utilizando a função HttpRequestFunctions.httpRequest1 para aceder às páginas HTML. Durante os testes, foram usados os *sources* fornecidos no Moodle para o DB-City, evitando bloqueios. Expressões regulares foram aplicadas para capturar os dados, com tratamento posterior para garantir consistência, conforme exigido no enunciado.

- **Atributos da Wikipédia**:
    - **Densidade Populacional**: Extraída do *infobox* da Wikipédia, em campos como "Densidade populacional" (ex.: “337 hab/km²” para o Japão). Usou-se expressões regulares para capturar o valor, padronizando o separador decimal para ponto (ex.: 337.0). No XML, observa-se que a densidade está presente (ex.: <densidade>337</densidade> para o Japão).
    - **Link da Bandeira**: Obtido a partir do atributo src da imagem da bandeira no *infobox* (ex.: //upload.wikimedia.org/.../Flag_of_Japan.svg). O XML reflete esta extração (ex.: <bandeira> para todos os países).
    - **Chefe de Estado**: Extraído do *infobox*, em campos como “Presidente”, “Rei”, “Imperador” ou “Monarca” .
- **Atributos do DB-City**:
    - **Capital**: Obtida da seção “Capital” (ex.: “Capital: Tóquio”)
    - **Continente**: Identificado em listas ou metadados (ex.: “Ásia”), com validação manual para evitar erros.
    - **Línguas Oficiais**: Capturadas em listas (ex.: “Língua: Japonês”).
    - **Área**: Extraída e formatada (ex.: "377.835" → 377835).
    - **População**: Captura habitantes e formatado (ex.: "1.395.380.000" → 1395380000).
    - **Religiões**: Extraídas de listas (ex.: “Budismo, Cristianismo”).
    - **Cidades Importantes**: Obtidas de tabelas extraídas de Cidades Importantes (ex.: “Tóquio, Yokohama”)
    - **Países Fronteiriços**: Extraídos de Fronteiras (ex.: “Mongólia, Rússia”).

- **Casos de COVID-19**: Capturados casos covid e formatado (ex.: “477.239” → 477239).

Os *wrappers* processaram as páginas linha a linha, aplicando padrões regex específicos. Os valores numéricos foram tratados para remover separadores de milhares e usar o ponto como separador decimal, conforme o exemplo do enunciado: “1.395.380.000” → <habitantes>1395380000</habitantes>.

Exceções e Inconsistências

Durante a extração, foram identificadas situações excecionais que exigiram decisões fundamentadas:

- **Variedade de Tipos de Chefe de Estado**: Na Wikipédia, o chefe de Estado é identificado por diferentes designações, como "Presidente", "Rei", "Imperador" ou "Monarca" (ex.: "Presidente: Xi Jinping" para a China; "Imperador: Naruhito" para o Japão),
- **Valores Numéricos Codificados com # e Números**: No Wikipédia, atributos numéricos como população e área apareciam ocasionalmente codificados com sequências como # seguido de números (ex.: "População: 1.395.380.000 #123456").

Modelo Global (Esquema XML)

Estrutura do Esquema

O esquema XML foi desenhado para representar os dados de múltiplos países, com uma hierarquia clara e flexível que facilita pesquisas XPath/XQuery e transformações XSLT/XQuery. A estrutura baseia-se no ficheiro XML fornecido (paises.xml), mas inclui considerações para a inclusão do atributo **chefe de Estado**, que está ausente no XML, apesar de ser extraído da Wikipédia, conforme indicado.

- **Elemento Raiz**: <Paises> serve como contêiner para todos os países, permitindo a agregação de múltiplas entradas. Esta escolha reflete a necessidade de uma visão global que suporte operações sobre o conjunto de dados.
- **Elemento de País**: Cada país é representado por um elemento <pais>, contendo todos os atributos relevantes como subelementos. Esta abordagem favorece a legibilidade e a compatibilidade com pesquisas XPath/XQuery, em comparação com o uso de atributos XML, que seria menos flexível para listas (ex.: línguas, religiões).
- **Subelementos**:
    - <capital>: Texto simples com o nome da capital (ex.: "Tóquio").
    - <continente>: Texto simples indicando o continente (ex.: "Ásia").
    - <bandeira>: URL da imagem da bandeira (ex.: //upload.wikimedia.org/.../Flag_of_Japan.svg).
    - <linguasOficiais>: Contêiner para uma lista de <lingua>, permitindo múltiplas línguas oficiais (ex.: "Japonês").
    - <area>: Valor numérico em km², sem separadores (ex.: 377835).
    - <populacao>: Valor numérico da população, sem separadores (ex.: 126494000).
    - <densidade>: Valor numérico (inteiro ou decimal) da densidade populacional em hab/km² (ex.: 337).
    - <religioes>: Contêiner para uma lista de <religiao>, suportando múltiplas religiões (ex.: "Budismo", "Cristianismo").

- <cidadesImportantes>: Contêiner para uma lista de <cidade>, representando cidades importantes (ex.: "Tóquio (Capital)").
- <paisesFronteira>: Contêiner para uma lista de <fronteira>, indicando países fronteiriços (ex.: "Mongólia").
- <casosCovid>: Valor numérico dos casos de COVID-19 (ex.: 477239).
- <chefeEstado>: Texto simples com o nome do chefe de Estado (ex.: "Naruhito

Justificativas:

- **Listas**: Elementos como <linguasOficiais>, <religioes>, <cidadesImportantes> e <paisesFronteiras> usam subelementos para representar listas, garantindo flexibilidade e escalabilidade (ex.: países com múltiplas línguas).
- **Valores Numéricos**: Atributos como <area>, <populacao> e <casosCovid> são armazenados sem separadores de milhares, conforme o enunciado, para compatibilidade com operações matemáticas em XPath/XQuery.

Validação do Esquema

O esquema foi validado usando DTD e XSD para garantir a integridade e conformidade dos dados, utilizando a API JDOM2, conforme especificado no enunciado.

- **DTD**:
  - Define a estrutura hierárquica e a cardinalidade dos elementos.
  - Especifica que <paises> contém zero ou mais <pais>, e cada <pais> inclui os subelementos obrigatórios (ex.: <capital>, <area>) e opcionais (ex.: <fronteiras> para países sem fronteiras terrestres, como o Japão).
  - Exemplo simplificado:

```
<!ELEMENT Paises (pais*)>

<!ELEMENT pais (capital, continente, bandeira, chefeEstado ,linguasOficiais, area,
populacao, densidade, religioes, cidadesImportantes, paisesFronteira, casosCovid)>
<!ATTLIST pais nome CDATA #REQUIRED>

<!ELEMENT capital (#PCDATA)>
<!ELEMENT continente (#PCDATA)>
<!ELEMENT bandeira (#PCDATA)>
<!ELEMENT chefeEstado (#PCDATA)>

<!ELEMENT linguasOficiais (lingua+)>
<!ELEMENT lingua (#PCDATA)>

<!ELEMENT area (#PCDATA)>
<!ELEMENT populacao (#PCDATA)>
<!ELEMENT densidade (#PCDATA)>

<!ELEMENT religioes (religiao*)>
<!ELEMENT religiao (#PCDATA)>

<!ELEMENT cidadesImportantes (cidade+)>
<!ELEMENT cidade (#PCDATA)>

<!ELEMENT paisesFronteira (paisFronteira*)>
<!ELEMENT paisFronteira (#PCDATA)>

<!ELEMENT casosCovid (#PCDATA)>

<!ATTLIST Paises xmlns:xsi CDATA #IMPLIED>

<!ATTLIST Paises xsi:noNamespaceSchemaLocation CDATA #IMPLIED>
```

- **XSD**:
    - Define tipos de dados específicos (ex.: xs:integer para <area>, xs:string para <capital>) e restrições (ex.: <densidade> como xs:decimal).
    - Inclui validações para URLs em <bandeira> e listas não vazias para <linguas> e <cidades>.
    - Exemplo simplificado:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Paises">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pais" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="capital" type="xs:string"/>
              <xs:element name="continente" type="xs:string"/>
              <xs:element name="bandeira" type="xs:string"/>
             <xs:element name="chefeEstado" type="xs:string"/>
              <xs:element name="linguasOficiais">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="lingua" type="xs:string" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="area" type="xs:integer"/>
              <xs:element name="populacao" type="xs:integer"/>
              <xs:element name="densidade" type="xs:decimal"/>
              <xs:element name="religioes">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="religiao" type="xs:string" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="cidadesImportantes">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="cidade" type="xs:string" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="paisesFronteira">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="paisFronteira" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="casosCovid" type="xs:integer"/>
            </xs:sequence>
            <xs:attribute name="nome" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Implementação dos Wrappers

Descrição dos Wrappers

Os *wrappers* são métodos da classe Wrappers.java que processam ficheiros HTML locais (wiki.html para Wikipédia e dbcity.html para DB-City), extraídos via HttpRequestFunctions.httpRequest1 ou fornecidos no Moodle durante a fase de testes. Cada método é responsável por extrair um atributo específico, utilizando expressões regulares (*regex*) para identificar padrões nas páginas HTML. Abaixo, detalha-se cada *wrapper*, a fonte correspondente e a informação extraída:

- **Wikipédia (wiki.html)**:
    - **procura_bandeira()**:
        - **Atributo**: Link da bandeira.
        - **Descrição**: Extrai a URL da imagem da bandeira do *infobox* da Wikipédia. A *regex* `<div style=\"padding-bottom:3px;\"><span [^t]*typeof=\"mw:File\"><a[^?]+><img alt=\"[^\"]+\" src=\"([a-zA-Z\\/._%0-9-]+)\"` captura o atributo src da tag <img> (ex.: //upload.wikimedia.org/.../Flag_of_Japan.svg).
        - **Exemplo de Saída**: <bandeira>//upload.wikimedia.org/.../Flag_of_Japan.svg</bandeira>.
    - **procura_densidade()**:
        - **Atributo**: Densidade populacional.
        - **Descrição**: Localiza o campo "Densidade" no *infobox* com a *regex* `Densidade<\\/a>`, seguida por `<td[^>]+>([0-9,]+) hab.\\/km²` para capturar o valor (ex.: "337"). A vírgula é substituída por ponto (ex.: "337,0" → 337.0), mas o XML mostra valores inteiros (ex.: <densidade>337</densidade>), indicando arredondamento.
        - **Exemplo de Saída**: <densidade>337</densidade>.
    - **procura_presidente()**:
        - **Atributo**: Chefe de Estado.
        - **Descrição**: Procura designações como "Presidente", "Rei", "Imperador" ou "Monarca" com a *regex*

`(Presidente|Rei|Imperador|Monarca)<\\/a>`, seguida por `<td[^>]+><a[^>]+>([^>]+)<\\/a>` para extrair o nome (ex.: "Naruhito").

- **Exemplo de Saída**: <chefeEstado>Naruhito</chefeEstado>

- **DB-City (dbcity.html)**:
  - **procura_capital()**:
    - **Atributo**: Capital.
    - **Descrição**: Extrai o nome da capital com a *regex* `<a[^>]+>Capital<\\/a>\\s<span[^>]+>[^<]+<\\/span><\\/th><td><a[^>]+>([^<]+)<\\/a>` (ex.: "Tóquio").
    - **Exemplo de Saída**: <capital>Tóquio</capital>.
  - **procura_continente()**:
    - **Atributo**: Continente.
    - **Descrição**: Captura o continente com a *regex* `Continente<\\/th><td><a [^>]+>([^<]+)<\\/a>` (ex.: "Ásia").
    - **Exemplo de Saída**: <continente>Ásia</continente>.
  - **procura_linguas()**:
    - **Atributo**: Línguas oficiais.
    - **Descrição**: Extrai uma lista de línguas com a *regex* `Língua oficial<\\/th><td>([^<]+(?:<br\\/>[^<]+)*)<\\/td>`, dividindo por
 (ex.: "Japonês").
    - **Exemplo de Saída**: <linguas><lingua>Japonês</lingua></linguas>.
  - **procura_area()**:
    - **Atributo**: Área.
    - **Descrição**: Captura a área com a *regex* `Superfície<\\/a><\\/th><td>([0-9.]+) km²<\\/td>`, removendo pontos (ex.: "377.835" → 377835).
    - **Exemplo de Saída**: <area>377835</area>.

- **procura_populacao()**:
  - **Atributo**: População.
  - **Descrição**: Extrai a população com a *regex* `População<\\/a><\\/th><td>([0-9.]+) habitantes`, removendo pontos (ex.: "1.395.380.000" → 1395380000).
  - **Exemplo de Saída**: <habitantes>1395380000</habitantes>.
- **procura_religioes()**:
  - **Atributo**: Religiões.

  Descrição: Captura uma lista de religiões com a *regex* `Religião <span [^>]+>[^<]+<\\/span><\\/h2><div [^<]+><ol [^>]+>(.*?)<\\/ol><\\/div> e <li><strong>([^<]+)<\\/strong>`, extraindo cada item (ex.: "Budismo", "Cristianismo").
  - **Exemplo de Saída**: <religioes><religiao>Budismo</religiao><religiao>Cristianismo</religiao></religioes>.
- **procura_cidadesImportantes()**:
  - **Atributo**: Cidades importantes.
  - **Descrição**: Extrai uma lista com a *regex* `Cidades importantes : ([^.]+)`, dividindo por vírgulas após substituir " e" por "," (ex.: "Tóquio, Yokohama").
  - **Exemplo de Saída**: <cidades><cidade>Tóquio (Capital)</cidade><cidade>Yokohama</cidade></cidades>.
- **procura_fronteira()**:
  - **Atributo**: Países fronteiriços.

  Descrição: Captura uma lista com a *regex* `Fronteira <span [^>]+>[^<]+<\\/span><\\/h2><div [^>]+><ol[^>]+>(.*?)<\\/div>` e `<a [^>]+><img [^>]+> ([^<]+)<\\/a>` (ex.: "Mongólia").
  - **Exemplo de Saída**: <fronteiras><fronteira>Mongólia</fronteira></fronteiras>.

- **procura_covid()**:
  - **Atributo**: Casos de COVID-19.

    Descrição: Extrai o número de casos com a *regex* `<h2 [^>]+>Covid-19 <span [^>]+>[^<]+<\\/span><\\/h2><div [^>]+><table><tr><th>Confirmado<\\/th><td>([0-9.]+)<\\/td><\\/tr>`, removendo pontos (ex.: "477.239" → 477239).

  - **Exemplo de Saída**: <casos_covid>477239</casos_covid>.

- **procura_dados_pais(String pais)**:
  - **Descrição**: Função principal que coordena a extração, verificando se o país já existe no XML via XPath (//pais[@nome='pais']). Se não existe, chama todos os *wrappers* e cria um objeto pais com os dados extraídos.
  - **Exemplo de Saída**: Objeto pais com todos os atributos para inserção no XML.

Tratamento de Dados

O tratamento de dados foi implementado para garantir que os valores extraídos sejam consistentes com os requisitos do enunciado, especialmente para valores numéricos e listas:

- **Valores Numéricos**:
  - **População e Área**: Extraídos do DB-City com pontos como separadores de milhares (ex.: "1.395.380.000"). O método .replace(".", "") remove os pontos, resultando em valores inteiros (ex.: <habitantes>1395380000</habitantes>, <area>9640011</area>).
  - **Casos de COVID-19**: Similarmente, pontos são removidos (ex.: "477.239" → <casos_covid>477239</casos_covid>).
  - **Densidade Populacional**: Extraída da Wikipédia com vírgulas (ex.: "337,0"). O método .replace(",", ".") converte para ponto decimal, mas o XML mostra valores inteiros (ex.: <densidade>337.0</densidade>)
- **Listas**:
  - **Línguas, Religiões, Cidades e Fronteiras**: Extraídas como strings com separadores (ex.:
 para línguas, vírgulas para cidades). Os métodos dividem

as strings em listas usando split(), garantindo elementos individuais no XML (ex.: <linguas><lingua>Japonês</lingua></linguas>).

- Exemplo para cidades: "Tóquio e Yokohama" → .replace(" e", ",") → split(", ") → ["Tóquio", "Yokohama"].

Manipulação do Ficheiro XML

Adição de Dados

A adição de novos países ao ficheiro XML é gerida pelo método **adicionaPais(pais p, Document doc)** da classe ModeloXML.java. Este método permite incorporar um objeto pais, contendo todos os atributos extraídos pelos *wrappers* (nome, capital, continente, bandeira, línguas, área, população, densidade, chefe de estado, religiões, cidades, fronteiras e casos de COVID-19), na estrutura XML.

- **Verificação de Informação já existente**:
  - Antes de adicionar um país, o método **procura_dados_pais** em Wrappers.java verifica a existência do país no XML usando uma expressão XPath (//pais[@nome='pais']). Se o país já existe, a função retorna null, evitando adicionar informação duplicada. Esta verificação é realizada antes da chamada a adicionaPais, garantindo que apenas países novos sejam adicionados.
  - Exemplo: Para adicionar "Japão", o XPath confirma que <pais nome="Japão"> não existe, permitindo a criação do elemento.
- **Criação do Ficheiro XML**:
  - Se o documento XML é nulo (doc == null), o método cria um novo documento com o elemento raiz <Paises> (notar que a capitalização difere de <paises> no XML fornecido, sugerindo possível inconsistência a corrigir). O novo documento é inicializado com **new Document(raiz)**.
  - Caso o documento já exista, o método obtém a raiz existente com **doc.getRootElement()** e adiciona o novo país como filho.

- **Estruturação do País**:
    - O método cria um elemento <pais> com um atributo nome (ex.: <pais nome="Japão">) e adiciona subelementos para cada atributo:
        - <capital>, <continente>, <bandeira>, <area>, <populacao>, <densidade>, <chefeEstado>, <casosCovid>: Elementos com texto simples (ex.: <capital>Tóquio</capital>).
        - <linguasOficiais>, <religioes>, <cidadesImportantes>, <paisesFronteira>: Elementos que contêm listas de subelementos (<lingua>, <religiao>, <cidade>, <paisFronteira>), geradas iterativamente a partir das listas do objeto pais.
    - Exemplo para línguas: List<String> listalinguas é percorrida, criando <lingua>Japonês</lingua> dentro de <linguasOficiais>.

Edição e Eliminação

A edição e eliminação de dados são suportadas por métodos específicos em ModeloXML.java, que permitem alterar atributos de países existentes e remover países do ficheiro XML, com validações para assegurar a integridade dos dados.

- **Edição de Atributos**:
    - Métodos como **alteraCapital**, **alteraPopulacao**, **alteraArea**, **alteraContinente**, **alteraDensidade** e **alteraCovid** permitem atualizar atributos específicos (capital, população, área, continente, densidade e casos de COVID-19) de um país identificado pelo atributo nome.
    - **Processo**:
        - O método verifica se o documento existe (doc != null). Se nulo, retorna erro.
        - Obtém a raiz (<Paises>) e itera sobre os elementos <pais> usando raiz.getChildren("pais").
        - Verifica se o atributo nome corresponde ao país desejado com p.getAttributeValue("nome").contains(nome).
        - Atualiza o subelemento correspondente com setText(novaValor) (ex.: p.getChild("capital").setText(novaCap)).
        - Retorna o documento atualizado ou null se o país não for encontrado.

- **Eliminação de Países**:
  - O método **removePais(String nome, Document doc)** remove um país do XML com base no atributo nome.
  - **Processo**:
    - Verifica se o documento existe. Se nulo, retorna erro.
    - Itera sobre os elementos <pais> e remove o elemento correspondente com p.getParent().removeContent(p) se p.getAttributeValue("nome").contains(nome).
    - Exibe mensagem de sucesso ou erro se o país não for encontrado.

Pesquisas XPath

Tipos de Pesquisas Implementadas

A interface gráfica, implementada em Interface.java, oferece dez pesquisas XPath acessíveis através do menu **XPATH**, cada uma associada a uma funcionalidade específica. As consultas são geridas no método btnXpathActionPerformed, que constrói expressões XPath dinamicamente com base nos inputs do utilizador (via txtXapth e txtXapth1) e no título da janela de diálogo (jdXpath). Abaixo, listam-se as pesquisas implementadas, com as respetivas expressões XPath e exemplos.

- **Pesquisa 1: Pesquisar pelo nome**
  - **Descrição**: Retorna todas as informações de um país com base no atributo nome.
  - **Expressão XPath**: //pais[@nome = 'pesquisa']
  - **Exemplo**: Para "Japão", a expressão //pais[@nome = 'Japão'] retorna:

```
<pais nome="Japão">
    <capital>Tóquio</capital>
    <continente>Ásia</continente>
    <!-- Outros elementos -->
</pais>
```

- **Pesquisa 2: Pesquisar cidades importantes de um dado país**
  - **Descrição**: Lista as cidades importantes de um país específico.
  - **Expressão XPath**: //pais[@nome='pesquisa']/cidadesImportantes/cidade
  - **Exemplo**: Para "Japão", a expressão //pais[@nome='Japão']/cidadesImportantes/cidade retorna:

```
<cidade>Tóquio (Capital)</cidade>
<cidade>Yokohama</cidade>
```

- **Pesquisa 3: Pesquisar países com número de habitantes superior a um valor introduzido**
  - **Descrição**: Lista os nomes dos países com população superior a um valor.
  - **Expressão XPath**: //pais[number(populacao) > valor] /@nome
  - **Exemplo**: Para população > 100 milhões, a expressão //pais[number(populacao) > 100000000] /@nome retorna:

```
China
Japão
```

- **Pesquisa 4: Pesquisar países com uma determinada religião**
  - **Descrição**: Lista os nomes dos países com uma religião específica.
  - **Expressão XPath**: //pais[//religioes/religiao ='pesquisa']/@nome
  - **Exemplo**: Para "Budismo", a expressão //pais[//religioes/religiao ='Budismo']/@nome retorna:

```
Japão
```

- **Pesquisa 5: Pesquisar todas as capitais existentes no ficheiro**
  - **Descrição**: Lista todas as capitais presentes no XML.
  - **Expressão XPath**: //pais/capital
  - **Exemplo**: A expressão //pais/capital retorna:

```
<capital>Tóquio</capital>
<capital>Pequim</capital>
<capital>Roma</capital>
<capital>Antananarivo</capital>
```

- **Pesquisa 6: Mostrar países com casos COVID-19 entre um valor mínimo e um valor máximo**
  - **Descrição**: Lista os nomes dos países com casos de COVID-19 num intervalo.
  - **Expressão XPath**: //pais[number(casosCovid) >= min and number(casosCovid) <= max]/@nome
  - **Exemplo**: Para casos entre 400000 e 500000, a expressão //pais[number(casosCovid) >= 400000 and number(casosCovid) <= 500000]/@nome retorna:

```
Japão
```

- **Pesquisa Adicional 1: Pesquisar países que partilham fronteira com um determinado país**
  - **Descrição**: Lista os nomes dos países com fronteira com o país especificado.
  - **Expressão XPath**: //pais[paisesFronteira/paisFronteira[contains(., '"+pesquisa+"')]]/@nome
  - **Exemplo**: Para "Mongólia", a expressão //pais[paisesFronteira/paisFronteira='Mongólia']/@nome retorna:

```
China
```

- **Pesquisa Adicional 2: Pesquisar países que tenham uma determinada língua oficial**
  - **Descrição**: Lista os nomes dos países com uma língua oficial específica.
  - **Expressão XPath**: //pais[linguasOficiais/lingua[contains(., '"+pesquisa+"')]]/@nome
  - **Exemplo**: Para "Japonês", a expressão //pais[linguasOficiais/lingua='Japonês']/@nome retorna:

    ```
    Japão
    ```

- **Pesquisa Adicional 3: Pesquisar países num determinado continente**
  - **Descrição**: Lista os nomes dos países pertencentes a um continente específico.
  - **Expressão XPath**: //pais[continente='pesquisa']/@nome
  - **Exemplo**: Para "Ásia", a expressão //pais[continente='Ásia']/@nome retorna:

    ```
    Japão
    China
    ```

- **Pesquisa Adicional 4: Pesquisar países com área maior que um valor introduzido**
  - **Descrição**: Lista os nomes dos países com área superior a um valor definido.
  - **Expressão XPath**: //pais[area > valor]/@nome
  - **Exemplo**: Para área superior a 1000000 km², a expressão //pais[area > 1000000]/@nome retorna:

    ```
    China
    ```

Transformações XSLT/XQuery

As transformações obrigatórias abrangem quatro processos que manipulam os dados do ficheiro XML (paises.xml) para gerar saídas em diferentes formatos, utilizando XSLT ou XQuery. Estas transformações estão implementadas na interface gráfica da aplicação, acessíveis através dos menus **XSLT** e **XQUERY** da classe Interface.java. Cada transformação é descrita abaixo, incluindo a sua funcionalidade, o ficheiro XSLT/XQuery utilizado, o método de execução e a justificação para a escolha da tecnologia.

- **Transformação 1: Gerar HTML com as bandeiras dos países**
    - **Descrição**: Produz um ficheiro HTML (bandeiras.html) que apresenta uma tabela com os nomes dos países e as respetivas bandeiras, extraídas do elemento <bandeira> do XML. É acionada pelo item de menu **Gerar HTML com as bandeiras dos países** (jMenuItem1).
    - **Ficheiro XSLT**: xsl_tabelaPaises.xsl

Transformação 2: Gerar ficheiro XML com os cinco países mais populosos

- **Descrição**: Gera um ficheiro XML (Top5PaisesPopulosos.xml) com os cinco países de maior população, ordenados por <populacao> em ordem decrescente. É acionada pelo item de menu **Gerar ficheiro XML com os 5 países mais populosos** (jMenuItem3).
- **Ficheiro XSLT**: xsl_paisesPopulosos.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<Top5>
    <pais nome="China">
        <populacao>1395380000</populacao>
    </pais>
    <pais nome="Canadá">
        <populacao>36994000</populacao>
    </pais>
</Top5>
```

Transformação 3 (Adicional): Gerar TXT com as línguas e religiões de cada país

- **Descrição**: Produz um ficheiro TXT (Linguas_Religioes.txt) que lista, para cada país, as línguas oficiais e religiões, extraídas dos elementos <linguasOficiais> e <religioes>. É acionada pelo item de menu **Gerar TXT com as línguas e religiões de cada país** (jMenuItem7).
- **Ficheiro XSLT**: xsl_linguasReligioes.xsl

```
China:
- Línguas: Língua chinesa
- Religiões: Budismo, Cristianismo, Islão, Outros

Canadá:
- Línguas: Inglês, Francesa
- Religiões: Cristianismo, Animismo, Islão, Outros, Hinduísmo, Budismo, Judaísmo
```

Transformação 4: Gerar ficheiro TXT com cidades importantes de um país

- **Descrição**: Gera um ficheiro TXT (CidadesImportantes.txt) com as cidades importantes de um país especificado pelo utilizador, extraídas de <cidadesImportantes/cidade>. É acionada pelo item de menu **Gerar ficheiro TXT com cidades importantes de um país** (jMenuItem2).
- **Ficheiro XQuery**: cidadesImp.xql

Transformação 5: Gerar HTML com os países fronteiriços de um país escolhido

- **Descrição**: Produz um ficheiro HTML (PaisesFronteiricos.html) que lista os países que partilham fronteira com um país especificado pelo utilizador, extraídos do elemento <paisesFronteira/paisFronteira>. É acionada pelo item de menu **Gerar ficheiro HTML com os países fronteiriços de um país escolhido** (jMenuItem4).
- **Ficheiro XQuery**: paisesFront.xql

Transformação 6 (Adiconal): Gerar HTML com países por continente

- **Descrição:** Produz um ficheiro HTML (PaisesPorContinente.html) que agrupa os países por continente, apresentando uma tabela com os nomes e capitais dos países de um continente especificado pelo utilizador. Extende a transformação obrigatória XQuery (jMenuItem5).
- **Ficheiro XQuery:** infoPorCont.xql

Transformação 7 (Adicional): Gerar XML com a informação de cada continente existente

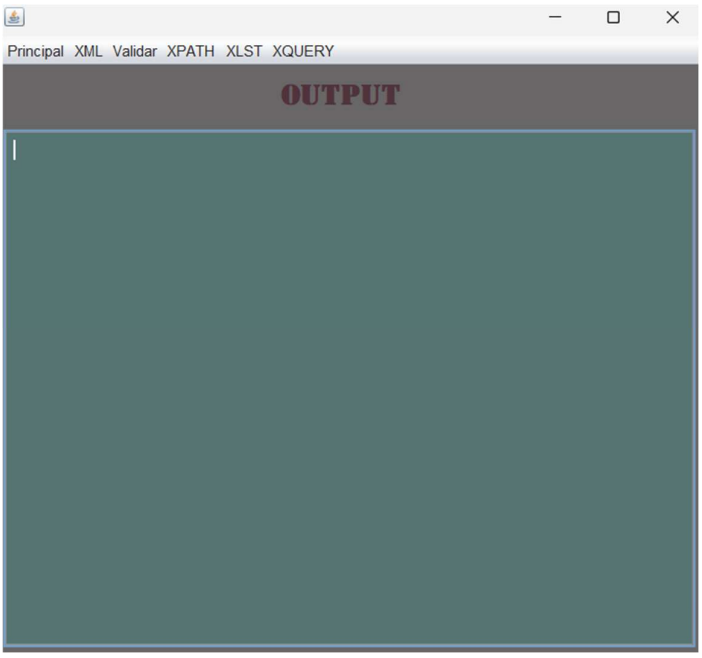
- **Descrição**: Gera um ficheiro XML (InfoContinentes.xml) que lista todos os continentes presentes no ficheiro paises.xml, incluindo, para cada continente, os países associados e informações como nome e capital. É acionada pelo item de menu Gerar XML com a informação de cada continente existente (jMenuItem6).
- **Ficheiro XQuery:** infoContinentes.xql

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Continentes>
    <continente nome="Ásia">
        <numeroDePaises>1</numeroDePaises>
        <Paises>
            <Pais>China</Pais>
        </Paises>
    </continente>
    <continente nome="América">
        <numeroDePaises>1</numeroDePaises>
        <Paises>
            <Pais>Canadá</Pais>
        </Paises>
    </continente>
</Continentes>
```

Interface Gráfica

A interface apresenta um design centrado numa barra de menus (JMenuBar) com várias opções e uma área de saída (JTextArea) para exibir conteúdos processados, como o XML original, resultados de consultas XPath/XQuery ou outputs de transformações XSLT.

- **Interface Gráfica**



Conteúdos Gráficos:

A interface apresenta uma barra de menus, localizada na parte superior da janela. Esta barra conta com 6 menus, estes são:

- Principal
- XML
- Validar
- XPATH
- XLST
- XQUERY

Onde ao selecionarmos o menu pretendido irá aparecer várias opções sobre o mesmo.

- **Barra de Menus:**

- **Menu Principal:**

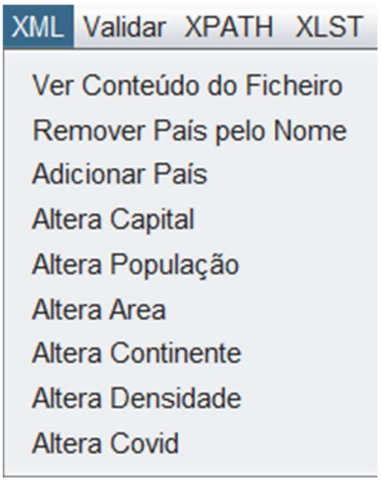
Inclui a opção "Sair" para encerrar a aplicação.

Menu Principal

- Menu XML:

Oferece funcionalidades para visualizar o conteúdo do ficheiro XML, remover países, adicionar novos países e alterar atributos como capital, população, área, continente, densidade ou casos de COVID-19.

Menu XML



- Menu Validar:

Permite validar o documento XML através de DTD (paises.dtd) ou XSD (paises.xsd).

Menu Validar

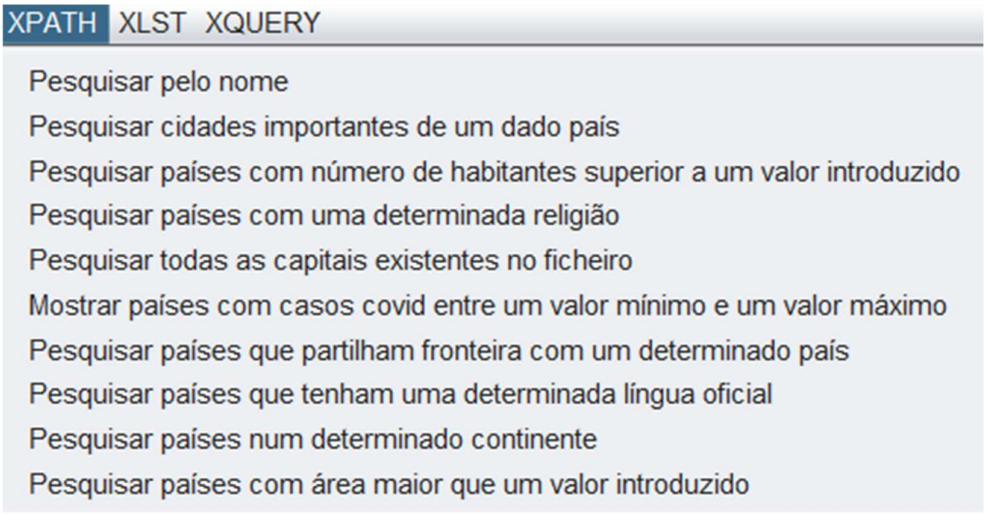
- Menu XPATH:

Disponibiliza opções para realizar consultas XPath, sendo estas:

- pesquisar países por nome
- cidades importantes,
- População
- Religião

- Capitais
- Casos de COVID-19
- Fronteiras
- Língua oficial
- Continente
- Área.

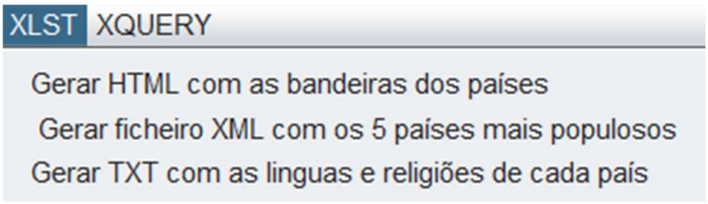
Menu XPATH:



- Menu XSLT:

Inclui transformações para gerar HTML com bandeiras dos países, XML com os cinco países mais populosos e TXT com línguas e religiões.

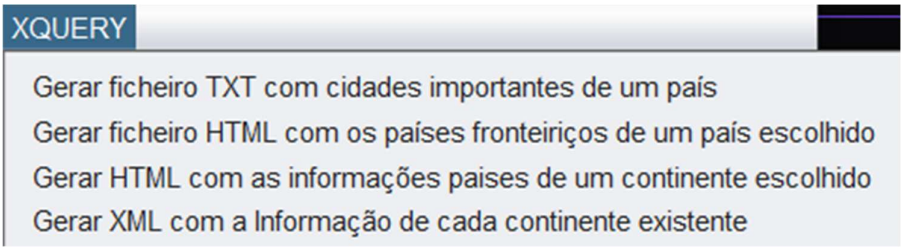
Menu XSLT



- Menu XQUERY:

Permite gerar ficheiros TXT com cidades importantes, HTML com países fronteiriços, HTML com informações de países por continente e XML com dados de continentes.

Menu XQUERY


XQUERY
Gerar ficheiro TXT com cidades importantes de um país
Gerar ficheiro HTML com os países fronteiriços de um país escolhido
Gerar HTML com as informações paises de um continente escolhido
Gerar XML com a Informação de cada continente existente

Referências Bibliográficas

Esta secção apresenta as fontes bibliográficas e recursos digitais consultados durante a realização do Trabalho Prático de Integração de Dados, no âmbito da unidade curricular de Integração de Dados do Instituto Superior de Engenharia de Coimbra (ISEC) para o ano letivo de 2024/2025. As referências incluem documentação oficial, plataformas de desenvolvimento, bases de dados online e ferramentas de inteligência artificial, que foram essenciais para a implementação das transformações XSLT/XQuery, pesquisas XPath, programação em Java e resolução de problemas técnicos. As citações seguem um formato adaptado para clareza e conformidade com padrões académicos, com links diretos para recursos online, conforme solicitado.

1. **Wikipédia**
    - **Descrição**: A Wikipédia foi utilizada como fonte de informação geral sobre países, que serviram de base para a estruturação do ficheiro XML (paises.xml).
    - **Link**: https://www.wikipedia.org
2. **DB-City**
    - **Descrição**: O portal DB-City forneceu informações detalhadas sobre países complementando os dados do ficheiro XML.
    - **Link**: https://www.db-city.com
3. **ChatGPT**
    - **Descrição**: A ferramenta de inteligência artificial ChatGPT, desenvolvida pela OpenAI, foi utilizada para criar estilos visuais (CSS) para as saídas HTML geradas pelas transformações XSLT e XQuery, garantindo uma apresentação mais apelativa. Além disso, o ChatGPT auxiliou na identificação e correção de erros durante a execução do trabalho, fornecendo explicações sobre exceções Java (ex.: XPathException, IOException) e sugestões para otimizar o código XSLT/XQuery.