Universidade de Coimbra

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

Mestrado em Engenharia Informática

Integração de Sistemas

Projeto 1, 2020/2021, 1º Semestre

Docente

Pedro Nuno San-Bento Furtado
(PNF@DEI.UC.PT)

Grupo: 23

David Silva de Paiva – nº 2020178529 - PL2

Ricardo David Da Silva Briceño – nº 2020173503 – PL2

Coimbra, 18 de outubro de 2020

Índice de Figuras

Índice de Tabelas

Integração de Sistemas - Projeto nº1

Índice

Integração de Sistemas - Projeto nº1

1. Enquadramento do projeto

Nesta secção são explicados, de uma forma concisa, os objetivos chave do projeto. Para além disso, é feito um breve enquadramento das funcionalidades da aplicação desenvolvida.

1.1 Objetivo geral

O primeiro projeto, da Unidade Curricular de Integração de Sistemas, consiste na implementação de uma aplicação que manipule um conjunto de dados, presentes num ficheiro no formato Extensible Markup Language (XML), e os apresente numa página no formato Hypertext Markup Language (HTML). Com isto, é espectável que os alunos explorem um conjunto de ferramentas e recursos – XML, XML Schema (XSD) e XSLT – adquirindo assim novas competências relativas às diversas tecnologias XML.

1.2 Enquadramento

Foi desenvolvida uma aplicação, em linguagem Java, que manipula um ficheiro XML que contem a informação de um conjunto de livros. O utilizador pode efetuar um conjunto de pesquisas de modo a filtrar um conjunto de livros através de uma interface texto na consola. Após a pesquisa, o conteúdo resultante da mesma é processado e reorganizado. Por fim, é gerado um ficheiro HTML que apresenta os livros resultantes da pesquisa e do processamento.

O projeto pode ser divido em três blocos fundamentais. Numa primeira parte, a classe *Selector* que começa por transformar a informação presente no ficheiro XML[1] em objetos Java (Unmarshal) seguindo restrições, como o tipo de dados, que são asseguradas por um ficheiro XSD[2]. Neste bloco é onde o utilizador interage com o sistema, efetuando uma pesquisa, de modo a selecionar um conjunto de livros. O *Selector* trata a pesquisa e por fim, escreve o resultado da pesquisa num novo ficheiro XML[3] (Marshal). Na eventualidade de

[1] bookdepository.xml
[2] bookdepositoryXSD.xsd
[3] BookdepositoryAfterSelector.xml

não ser devolvido nenhum livro após a pesquisa, o utilizador recebe uma mensagem de aviso na consola.

Consequentemente, numa segunda parte, a classe *Processor* começa por ler a informação escrita no novo ficheiro XML gerado e reorganiza-a num novo formato, gerando outro ficheiro XML[4]. Este documento estrutura-se seguindo uma lista de autores, onde cada um destes possuí uma lista de livros ordenados pelo ranking de *bestseller*. Para além disso, são adicionadas algumas informações estatísticas como o número de autores processados pela classe *Processor* e o nome de um número N de autores. Este, número N, pode ser manipulado pelo utilizador através da passagem por argumento para o método *main* ou no caso de isto não acontecer, o número é explicitamente solicitado ao utilizador no início da execução da classe *Processor*.

O último e terceiro bloco do projeto, a classe *HTMLViewer*, transforma, o ficheiro XML devolvido pelo *Processor*, no formato HTML recorrendo a um ficheiro XSL. Para que esta transformação ocorra, foi criado um XSL no qual é aplicado um template que permite ver todos os autores e as suas respetivas obras resultantes da pesquisa inicial. Cada livro contém informação, como por exemplo, a data de publicação, o ranking atribuído pelos leitores, a categoria literária, entre outros atributos. Para obter esta informação utilizou-se unicamente XPath, que permite navegar nos elementos e atributos de um ficheiro XML.

A Figura 1, mostra uma ilustração do *workflow* da aplicação desenvolvida neste primeiro projeto.

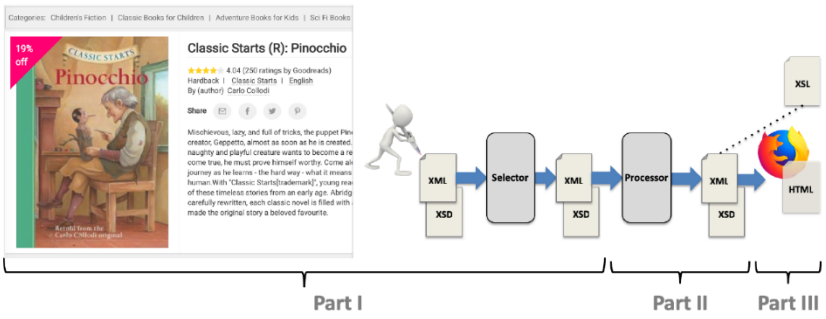

Figura 1 - Ilustração do workflow da aplicação

[4] listofauthors.xml

1.3 Estrutura do documento

Para além deste enquadramento, este documento é composto pelos seguintes capítulos:

- Capítulo 2 – Registo de Trabalhos, onde é apresentado um resumo das funcionalidades implementadas no decorrer do projeto e onde é, também, feita uma autoavaliação global do projeto.
- Capítulo 3 - Estrutura dos ficheiros XML, XSD e XSLT, onde são apresentados os ficheiros fundamentais no processo de integração e manipulação de dados, assim como a sua estrutura interna.
- Capítulo 4 – Aspetos de Implementação, onde são abordadas algumas opções mais significativas, tomadas no desenvolvimento do projeto.
- Capítulo 5 – Considerações Finais, onde é feito um balanço dos objetivos alcançados, assim como as competências adquiridas com a realização do projeto.

2. Registo de Trabalhos

Nesta secção, é apresentado um resumo das funcionalidades implementadas no decorrer do projeto e é, também, feita uma autoavaliação global do projeto.

2.1 Lista de funcionalidades/objetivos implementados

A Tabela 1 contém a lista de funcionalidades/objetivos que foram implementados no primeiro projeto. Em suma, todas as tarefas propostas e descritas no enunciado, foram implementadas com sucesso.

Funcionalidades/Objetivos	Implementado/Não implementado	Responsável	Esforço (horas)
Parte 1 - Selector	**Implementado**	Ambos	8
Ficheiro XML que armazena a informação relativa a um conjunto de livros.	**Implementado**	Ricardo	3
Ficheiro XSD que valida a informação contida no ficheiro XML – referido na linha anterior (trang + adaptações)	**Implementado**	Ricardo	3/4
Geração de classes Java usando o JAXB (jxjc)	**Implementado**	Ricardo	1/4
Geração de objetos Java usando JAXB (Unmarshal)	**Implementado**	Ricardo	1/2
Implementação de regras que permitem ao utilizador realizar uma pesquisa.	**Implementado**	David	2
O utilizador pode conjugar mais do que uma regra para realizar a pesquisa.	**Implementado**	David	1
Pesquisa é realizada e é produzido um novo ficheiro XML com o resultado da pesquisa.	**Implementado**	David	1/4
Informar o utilizador que quando não existem livros que cumpram os objetivos de pesquisa, a pesquisa tem de ser repetida.	**Implementado**	David	1/4
Parte 2 - Processor	**Implementado**	David	6

A informação do ficheiro XML, resultante da pesquisa feita no Selector, é reorganizada.	**Implementado**	David	2
A informação é reorganizada numa lista de autores, onde cada um contém uma lista de livros ordenada pelo bestsellers rank.	**Implementado**	David	2
Estatísticas (número de autores processados e nome de N autores) são adicionadas no fim da reorganização.	**Implementado**	David	1
O utilizador pode facilmente alterar o valor de N.	**Implementado**	David	1/4
É produzido um ficheiro XML com o resultado da reorganização.	**Implementado**	David	1/4
Ficheiro XSD que valida a informação contida neste ficheiro XML.	**Implementado**	David	1/2
Parte 3 – HTML Viewer	**Implementado**	Ricardo	5
Escrita do ficheiro XSL	**Implementado**	Ricardo	1
Usar XPath para obter dados do ficheiro XML	**Implementado**	Ricardo	2
Integração de CSS	**Implementado**	Ricardo	1/4
Integração de JavaScript	**Implementado**	Ricardo	1/2
Transformação da informação do ficheiro XML - produzido pelo Processor - num ficheiro HTML.	**Implementado**	Ricardo	1
Apresentação num browser da página HTML produzida.	**Implementado**	Ricardo	1/4
Relatório	**Implementado**	Ambos	8

Tabela 1 - Lista de funcionalidades/objetivos implementados

2.2 Autoavaliação individual e global do projeto

O aluno, David Paiva, autoavalia o desempenho do grupo com uma nota de dezanove valores - numa escala de zero a vinte. Relativamente a uma autoavaliação individual, avalia o seu desempenho e o do colega, Ricardo Briceño, com dezanove valores para ambos.

O aluno, Ricardo Briceño, autoavalia o desempenho do grupo com uma nota de dezanove valores - numa escala de zero a vinte. Relativamente a uma

autoavaliação individual, avalia o seu desempenho e o do colega, David Paiva, com dezanove valores para ambos.

3. Estrutura dos ficheiros XML, XSD e XSLT

Nesta secção, são apresentados os ficheiros fundamentais no processo de integração e manipulação de dados, assim como a sua estrutura interna.

3.1 Ficheiro XML bookdepository

O ficheiro *XML, bookdepository,* armazena a informação de cerca de cinquenta e seis livros, toda ela retirada do *website: https://www.bookdepository.com/*. O ficheiro seguiu uma estrutura simples como mostra a Figura 2.

```
<bookdepository>
    <book>
        <bookTitle>Fast Exercise</bookTitle>
        <bookInitials>FE</bookInitials>
        <author>
            <authorName>Dr Michael Mosley</authorName>
            <authorDescription>Dr Michael Mosley trained as a doctor before becoming a
            journalist and television presenter. He is the author of The Fast Diet,
            The 8-Week Blood Sugar Diet, The Clever Guts Diet and The Fast 800. He
            is married with four children. Dr Michael Mosley trained as a doctor before
            oming a journalist and television presenter. He is the author of The Fast
            Diet, The 8-Week Blood Sugar Diet, The Clever Guts Diet and The Fast 800.
            He is married with four children.</authorDescription>
        </author>
        <categorie>Fitness &amp;amp; Diet</categorie>
        <bookSummary> The simple secret of high intensity training: get fitter,
        stronger and better toned in just a few minutes a day</bookSummary>
        <rating>3.79/5</rating>
        <pagesNumber>208</pagesNumber>
        <publicationDate>2020-12-19</publicationDate>
        <publisher>Short Books Ltd</publisher>
        <locationCity>London</locationCity>
        <locationCountry>United Kingdom</locationCountry>
        <language>English</language>
        <ISBN10>1780721986</ISBN10>
        <ISBN13>781780721989</ISBN13>
        <sellerRank>22679</sellerRank>
        <linkImg>https://d1w7fb2mkkr3kw.cloudfront.net/assets/images/book/lrg/9781/7807/9781780721989.jpg
    </book>
```

Figura 2 - Estrutura do ficheiro bookdepository.xml

O Root element é o elemento bookdepository, que como o próprio nome indica, representa o repositório de livros. Por sua vez, o elemento book, que representa um livro, é composto por outros elementos, como é o caso dos seguintes elementos:

- elementos *bookTitle* – título do livro;
- *author* – autor do livro;
- *categorie* – categoria do livro;

- *bookSummary* – resumo do livro, *rating* – avaliação dos comprados*;*
- *pagesNumber* - número de páginas do livro;
- publicationDate – data de publicação do livro;
- publisher – Editor;
- locationCity – Cidade de lançamento;
- locationCoutry – País de lançamento;
- language – Língua em que o livro está escrito;
- ISBN10 – Código de identificação de dez dígitos;
- ISBN13 – Código de identificação de treze dígitos;
- sellerRank – Posição no *bestseller rank*;
- linkImg – *link* para uma imagem que representa a capa do livro.

Alguns destes elementos, são também elementos complexos, ou seja, são eles também compostos por outros elementos, como é o caso do elemento *author*.

Foi ainda, adicionada informação que não é visualizada pelo utilizador de forma direta, como o caso da informação contida na tag *<bookInitials>FE</bookInitials>*. Esta informação serviu de ajuda no bloco *HTMLViewer*, mais concretamente no ficheiro *XSL,* para manipulação de variáveis e atributos que ajudaram na parte visual do sistema.

O grupo decidiu não colocar informação comercial, como o preço ou descontos ativos para um determinado livro, de modo a não copiar na integra a *BookDepository's online Store,* optando por apresentar apenas informação sobre os livros e os seus autores.

Este é, o ficheiro, usado como ponto de partida para gerar objetos Java (processo de *Unmarshal*) e, que, por sua vez, serve como base de pesquisa para a filtragem de dados que é realizada no *Selector*.

3.2 Ficheiro XML BookdepositoryAfterSelector

O ficheiro XML *BookdepositoryAfterSelector* é um ficheiro gerado automaticamente após o bloco *Selector* ser executado e o processo de *Marshal* ter terminado. Contém a informação dos livros filtrados, com base na pesquisa

realizada pelo utilizador no bloco *Selector*. É, resumidamente, um ficheiro com a mesma estrutura que o ficheiro XML *bookdepository* mas com a informação filtrada /reduzida.

3.3 Ficheiro XML listofauthors

O ficheiro XML *listofauthors* é um ficheiro gerado a partir do ficheiro XML *BookdepositoryAfterSelector. Contém,* exatamente a mesma informação, mas está estruturado/organizado de maneira diferente. Ou seja, enquanto que o ficheiro *BookdepositoryAfterSelector* armazena todos os livros e a informação a si associada, o ficheiro *listofauthors* armazena a informação dos autores e os livros a si associados. Para deixar claro esta distinção, a **Error! Reference source not found.** mostra um exemplo do ficheiro listofauthors.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<listofauthors>
    <authorp>
        <name>George R. R. Martin</name>
        <authorDescription>George R.R. Martin is the author of fourteen novels,
        including five volumes of A SONG OF ICE AND FIRE, several collections
        of short stories and numerous screen plays for television drama and
        feature films. He lives in Santa Fe, New Mexico</authorDescription>
        <bookp>
            <bookTitle>A Game of Thrones : Book 1 of A Song of Ice and Fire</boo
            <bookInitials>GOTBAIF</bookInitials>
            <categorie>Fiction</categorie>
            <bookSummary>The first volume of A Song of Ice and Fire, the
            greatest fantasy epic of the modern age. GAME OF THRONES is now
            a major TV series from HBO, starring Sean Bean. Summers span
            decades. Winter can last a lifetime. And the struggle for the
            Iron Throne has begun. As Warden of the north, Lord Eddard Stark
            counts it a curse when King Robert bestows on him the office of
            the Hand. His honour weighs him down at court where a true man
            does what he will, not what he must ...and a dead enemy is a thing
            of beauty. The old gods have no power in the south, Stark's family
            is split and there is treachery at court. Worse, the vengeance-mad
            heir of the deposed Dragon King has grown to maturity in exile in
            the Free Cities. He claims the Iron Throne.</bookSummary>
```

Figura 3 - Estrutura do ficheiro listofauthors.xml

3.4 Ficheiros XSD bookdepositoryXSD e listofauthorsXSD

Estes ficheiros foram criados com o objetivo de definir de forma robusta os respetivos esquemas dos documentos XML, ou seja, a forma como os elementos aparecem no documento, os tipos de dados e restrições de aridade.

No exemplo, apresentado na Figura 4, é possível observar uma *tag: complexType*. Esta *tag*, declara elementos que são compostos por outros elementos. Neste caso, em específico, o elemento *statistics* é composto por outro com a referência *totalauthorsprocessed* e *authorName* ambos com restrições que de certa forma os tornam elementos de presença tornam obrigatórias no ficheiro XML. De seguida, defininem-se os tipos de dados que estes elementos iram tomar no esquema XML.

```
  <xs:element name="statistics">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" ref="totalauthorsprocessed"/>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="authorName"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="totalauthorsprocessed" type="xs:integer"/>
  <xs:element name="authorName" type="xs:string"/>
</xs:schema>
```

Figura 4 - Estrutura do ficheiro listofauthorsXSD.xsd

Todos os ficheiros XSD produzidos seguiram a estrutura apresentada. Primeiro, foram identificados os elementos e os seus *complexType*, com restrições de aridade, e de seguida foram definidos os tipos de dados. Depois de construídos estes ficheiros são aplicados na validação dos seus respetivos ficheiros XML.

3.5 Ficheiro XSL HTMLViewer e HTML finalOutputHTML

O ficheiro XSL, *HTMLViewer,* irá transformar o ficheiro XML *listofauthors* num ficheiro HTML denominado *finalOutputHTML.* No decorrer desta transformação, são usadas bibliotecas do Bootstrap de CSS e JavaScript de como a tornar a visualização dos dados mais agradável e apelativa. Toda a seleção de dados, realizada para apresentar no ficheiro HTML, é feita com recurso à tecnologia XPath. Esta, permite percorrer os nós pretendidos e obter assim os dados de interesse.

No exemplo da Figura 5, é possível observar de que forma é que os resultados foram obtidos. Neste exemplo, em concreto, são procurados todos os livros no XML, e de seguida, são obtidos todos os valores dos seus respetivos nós, como é o caso de bookInitials, linkImg, categorie, entre outros. É importante referir a definição de atributos nas diferentes tags HTML, como é o caso da tag <img></img> que precisa de uma fonte - "src" - para fazer o display de uma imagem. Esta imagem corresponde à capa do livro.

```
<xsl:for-each select="bookp">
<div class="collapse">
    <xsl:attribute name="id">
        <xsl:value-of select="bookInitials"/>
    </xsl:attribute>
    <hr></hr>
        <div>
            <img>
                <xsl:attribute name="src">
                    <xsl:value-of select="linkImg"/>
                </xsl:attribute>
            </img>
        </div>
        <small class="card-text">
            Categorie:
            <xsl:value-of select="categorie"/><br/>
            Pages:
            <xsl:value-of select="pagesNumber"/><br/>
            Language:
            <xsl:value-of select="language"/><br/>
            Rating:
            <xsl:value-of select="rating"/> <i class='far fa-star'></i><br/>
        </small>
```

Figura 5 - Estrutura do ficheiro HTMLViewer.xsl

4. Aspetos de Implementação

Nesta secção, são abordadas algumas das opções mais significativas, tomadas no desenvolvimento do projeto.

4.1 Estrutura do projeto

A implementação do projeto foi dividida em duas *packages*. A *package generatedclass,* que contém as classes geradas automaticamente pela biblioteca *jaxb*. Este conjunto de classes representa o modelo de dados, dentro do código, ou seja, são estas classes que guardam a informação lida dos ficheiros XML e permitem que, a mesma seja manipulada. Por sua vez, a *package mainpackage* contém as classes que implementam os três blocos do projeto: *Selector, Processor* e *HTMLViewer.* Estas, são as classes responsáveis, por manipular o modelo de dados, como já foi explicado anteriormente.

Cada uma das classes, que implementam os três blocos do projeto, possui um método *main,* para que possam ser executadas de forma independente.

4.2 Pesquisa de livros - Selector

Foram implementadas seis regras de pesquisa. De uma forma sucinta, o utilizador tem a possibilidade de pesquisar livros pelo nome do autor, pelo ano mínimo de publicação, pelo título, pelo mínimo de rating, pelo um valor máximo do *bestseller rank* e, também, pela categoria. Para além disso, o utilizador tem a possibilidade de combinar mais do que uma regra, no decorrer uma pesquisa, de modo a restringir o conjunto de livros que pretende consultar.

A pesquisa é feita numa interface de texto, na consola, onde o utilizador tem a possibilidade de escolher as diferentes regras de pesquisa e inserir os valores de pesquisa pretendidos. As escolhas de pesquisa e os valores inseridos são armazenados em dois objetos do tipo *ArrayList*, respetivamente. Após o utilizador dar a pesquisa por terminada, as opções de pesquisa e os respetivos valores são processados. Isto, é feito no método *processUserPreferrences – classe Selector* - que, no fim do processamento, devolve um novo objeto, do tipo *bookdepository*, com o conjunto de livros encontrado.

No caso em que, no fim da pesquisa não é encontrado nenhum resultado, é gerada uma exceção do tipo *NoSearchResultException*. Esta exceção é propagada para o método *main, onde é apanhada através do bloco try catch*, com o objetivo de informar o utilizador que a pesquisa não devolveu nenhum resultado e que, portanto, deve ser realizada uma nova pesquisa.

4.3 Reorganização da informação – Processor

Nesta fase, os objetos, *book* e *author* são convertidos em objetos, *bookp* e *auhtorp*, respetivamente. Isto acontece, para que a informação reorganizada seja escrita no ficheiro XML seguindo um novo formato – como explicado na secção 3.3.

Posto isto, a reorganização da informação é feita no método *reorganizeInformation – classe Processor* – com ajuda de objetos *ArrayList* auxiliares. No entanto, para ordenar os livros de cada autor pelo *bestseller rank,* foi necessário implementar um *Comparator*, que implementa uma ordenação ascendente, que é utilizado pelo método *sort* de um objeto ArrayList.

No fim da reorganização da informação, são também adicionadas informações estatísticas, como já foi referido anteriormente.

4.4 Número N de nomes de autores a adicionar nas estatísticas

Uma das funcionalidades pedidas no enunciado do projeto, era que, deviam ser adicionados, às informações estatísticas, um número N de nomes de autores com livros com o *bestseller rank* mais elevado. No entanto, era imposto que este número fosse facilmente especificado pelo utilizador. De forma a tornar isto possível, foram implementadas duas alternativas para que, o utilizador especifique este número. Este valor pode ser passado por argumento para o método *main* da classe *Processor,* ou então, caso isto não aconteça, no início da execução é solicitado que o utilizador insira um possível valor.

4.5 HTML Viewer

Com o objetivo de mostrar o resultado da pesquisa ao utilizador, depois de feita a transformação da informação para uma página HTML, é aberto o *browser default* da máquina onde o programa é executado.

5. Considerações finais

Dado por terminado o projeto é necessário fazer um balanço dos objetivos alcançados, assim como as competências adquiridas.

De uma forma geral, todas as funcionalidades solicitadas no enunciado foram implementadas com sucesso. Com isto em mente, é importante realçar que os objetivos de aprendizagem foram alcançados. Nomeadamente, foram apreendidas um conjunto de técnicas e ferramentas XML, como por exemplo XML Schema, XSLT, entre outras.