Uma imagem com texto

Descrição gerada automaticamente Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e Sistemas

Licenciatura em Engenharia Informática

Unidade Curricular: Integração de Dados

2020/2021

Relatório de Trabalho Prático

*Integração de dados com XML*

­

Isabel Ramos Castro 2018013160

Índice

[Estrutura do projeto 3](#_Toc73469756)

[Interface Gráfica 4](#_Toc73469757)

[HttpRequestFunctions 4](#_Toc73469758)

[XMLJDomFunctions 5](#_Toc73469759)

[XPathFunctions 5](#_Toc73469760)

[JDOMFunctions\_Validar 6](#_Toc73469761)

[JDOMFunctions\_XSLT 7](#_Toc73469762)

[SaxonFunctions\_XQuery 7](#_Toc73469763)

[Jogador 8](#_Toc73469764)

[Wrappers 9](#_Toc73469765)

[Modelo XML 13](#_Toc73469800)

[Pesquisas XPATH 14](#_Toc73469801)

[XSLT 15](#_Toc73469808)

[Ficheiro HTML com fotos dos jogadores 15](#_Toc73469809)

[Ficheiro com Top 5 dos jogadores mais valiosos 16](#_Toc73469810)

[Ficheiro com jogadores ordenados por ordem de idade 16](#_Toc73469811)

[Ficheiro com clubes atuais e situação dos jogadores 17](#_Toc73469812)

[XQUERY 17](#_Toc73469813)

[Ficheiro XML com lista de jogadores de um clube 17](#_Toc73469814)

[Ficheiro com jogadores de uma nacionalidade 18](#_Toc73469815)

[Ficheiro com alguns dados mais importantes de jogadores 18](#_Toc73469816)

[Ficheiro com lista de trofeus de jogadores 19](#_Toc73469817)

# Estrutura do projeto

O projeto criado no Netbeans divide-se em vários ficheiros:

− HttpRequestFunctions

− XMLJDomFunctions

− XPathFunctions

− JDOMFunctions\_Validar

− JDOMFunctions\_XSLT

− SaxonFunctions\_XQuery

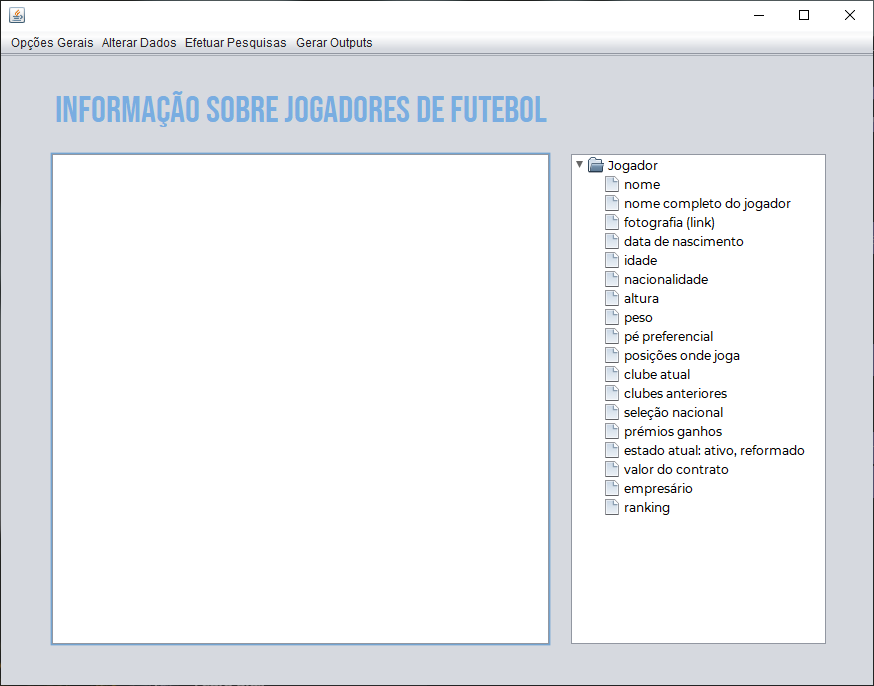
− Jogador

− Wrappers

− ModeloXML

− Interface

− main

Interface Gráfica  
A figura seguinte ilustra a interface gráfica implementado para o projeto. A  
área de texto maior é o espaço onde serão apresentados a maioria dos resultados das diversas opções do menu, do lado direito há uma lista dos dados que são apresentados.

HttpRequestFunctions  
*Ficheiro fornecido nas aulas práticas*  
public static int httpRequest1(String link, String pesquisa, String outFile)  
public static void httpRequest2(String link, String pesquisa, String outFile)

Ambas recebem um link de uma página web e uma string (pesquisa) com o que se pretende pesquisar, estas funções juntam o link e a pesquisa formando o URL a que vão aceder, e de seguida guardam a informação da página obtida num ficheiro html, cujo o nome é passado por argumento para a função (variável outFile).  
A única diferença entre estas duas funções é a codificação usada para os caracteres, pois pode variar de site para site, neste projeto utilizei as duas funções pois um dos websites (ZeroZero) apresentava por vezes dificuldade no acesso.

# XMLJDomFunctions

*Ficheiro fornecido nas aulas práticas*

public static Document lerDocumentoXML(String caminhoFicheiro)

public static void escreverDocumentoParaFicheiro(Document doc, String caminhoFicheiro)

public static String escreverDocumentoString(Document doc)

Tal como o nome indica a função lerDocumentoXML lê o documento XML, para que posteriormente se possa pesquisar, alterar ou transformar o seu conteúdo, o ficheiro XML tem um nome que é passado como argumento da função através da variável caminhoFicheiro, a função devolve um documento com o conteúdo do ficheiro XML lido.

A função escreverDocumentoParaFicheiro cria em disco um documento, caso ele não exista, caso exista altera-o, nesse documento fica o conteúdo de um documento XML que está previamente em memória. Na variável doc está guardado o conteúdo XML, na variável caminhoFicheiro o nome do ficheiro que vai ser criado/alterado.

Com a função escreverDocumentoString é possível aceder a um documento com o conteúdo de um ficheiro XML e guardar todo o seu conteúdo numa variável do tipo String, a função recebe como parâmetro um documento (doc), previamente guardado na memória.

Para que estas funções funcionem corretamente é necessário adicionar ao projeto a API JDOM.

XPathFunctions  
*Ficheiro fornecido nas aulas práticas*  
static XdmValue executaXpath(String xp, String xmlFile)  
static String listaResultado(XdmValue lista)

A função executaXpath é responsável pela pesquisa XPath num ficheiro XML, a  
variável xp é uma String com a pesquisa XPath e a variável xmlFile tem o nome do  
ficheiro XML onde a pesquisa vai ser efetuada, esta função devolve a lista de resultados que encontrou.  
A função listaResultados recebe o resultado da função anterior (executaXpath) e  
devolve uma String com o resultado da pesquisa efetuada anteriormente.

# JDOMFunctions\_Validar

*Ficheiro fornecido nas aulas práticas*

public static Document validarDTD(String caminhoFicheiro)

public static Document validarXSD(String caminhoFicheiro)

public static int validarDocumentoDTD(String xmlFile, String DTDFile)

public static int validarDocumentoXSD(String xmlFile, String XSDFile)

A função validarDTD abre o ficheiro XML indicado na variável caminhoFicheiro e introduz o cabeçalho necessário para a validação posterior, a função validarXSD faz exatamente o mesmo, desta vez, introduz o cabeçalho correspondente à validação por XSD, caso o ficheiro seja válido por XSD a função validarXSD devolve um documento com conteúdo do ficheiro XML validado, caso o ficheiro seja válido por DTD a função validarDTD devolve um documento com conteúdo do ficheiro XML validado.

A função validarDocumentoDTD, recebe como parâmetros o nome do ficheiro DTD com a estrutura do ficheiro XML pretendido (DTDFile) e o nome do ficheiro XML que se pretende verificar (xmlFile) esta função devolve -1 caso o ficheiro seja inválido e 1 caso seja válido.

A função validarDocumentoXSD recebe como parâmetros o nome do ficheiro XSD com a estrutura do ficheiro XML pretendido (XSDFile) e o nome do ficheiro XML que se pretende verificar (xmlFile), caso o ficheiro seja inválido a função devolve -1 caso seja válido devolve 1.

Para o correto funcionamento destas funções é necessário o recurso à API JDOM e SAXON, neste projeto adicionei duas bibliotecas da API SAXON, saxon9.jar e saxon-s9api.jar.

# JDOMFunctions\_XSLT

*Ficheiro fornecido nas aulas práticas*

public static Document transformaDocumento(Document XMLdoc, String xmlFile, String xslFile)

public static void transformaDocumento2(String xmlFile,String xslFile, String sOutFile)

A função transformaDocumento recebe um documento com o conteúdo XML a transformar, o nome do ficheiro XML a transformar e o nome do ficheiro XSL com o template de transformação e devolve um documento com o conteúdo XML/HTML resultante.

A função transformaDocumento2 recebe o nome do ficheiro XML a transformar, o nome do ficheiro XSL com o template de transformação e o nome do ficheiro TXT onde ficará guardado o conteúdo resultante da operação.

SaxonFunctions\_XQuery  
*Ficheiro fornecido nas aulas práticas*  
public static void xQueryToText(String outputFile, String queryFile)  
public static void xQueryToHtml(String outputFile, String queryFile)  
public static void xQueryToXml(String outputFile, String queryFile)

A função xQueryToText recebe como parâmetros o nome do ficheiro TXT onde  
será guardado o resultado outputFile e o nome do ficheiro onde se encontra a query queryFile.  
A função xQueryToHtml recebe como parâmetros o nome do ficheiro HTML onde  
será guardado o resultado outputFile e o nome do ficheiro onde se encontra a query  
queryFile.  
A função xQueryToXml recebe como parâmetros o nome do ficheiro XML onde  
será guardado o resultado outputFile e o nome do ficheiro onde se encontra a query  
queryFile.

# Jogador

O ficheiro jogador.java tem armazenada a classe Jogador com o seu construtor e respetivos getters e setters. Esta é composta por vários atributos:

* String altura
* String clubeAtual
* ArrayList<String> clubesAnteriores
* String dataNasc
* String empresario
* String estadoAtual
* String fotografia
* Int idade
* String nacionalidade
* String nomeCompleto
* String alcunha
* String peso
* String posicao
* ArrayList<String> trofeus
* String pé preferencial
* String ranking
* String selecao
* String contrato

Depois da análise das fontes de dados foram privilegiados, sempre que possível,

os dados presentes na página TransferMarket, pois o website ZeroZero apesar de ter o source mais linear e mais fácil de extrair dados tinha uma limitação de acessos. Assim sendo, os dados foram extraídos dos websites segundo a tabela do lado esquerdo.

# Wrappers

String Obtem\_Alcunha(String pesquisa)

Devolve “não disponível” caso não haja um nome mais reduzido para o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa a seguinte ER:

### String er = "<title>([A-Za-zÀ-ÿ]+)(\\s[A-Za-zÀ-ÿ]+)";

String Obtem\_Altura(String pesquisa)

Devolve “não disponível” caso não haja um nome mais reduzido para o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas.

### String er = "<th[^>]\*>Altura";

### String er1 = "</th>";

### String er2 = "<td[^>]\*>([0-9\\s\\,m]+)";

ArrayList<String> Obtem\_ClubeAnterior

Devolve um array com os 4 mais recentes clubes anteriores do jogador.

### String er = "<td class=\"hauptlink no-border-links vereinsname\"><a class=\"vereinprofil\_tooltip\" id=\"([0-9])+\" href=\"([^\"]+)\">([^<]+)</a></td>";

String Obtem\_ClubeAtual(String pesquisa)

Devolve “não disponível” caso não haja um clube atual para o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er = "Clube atual:";

### String er1 = "\\s+</th>";

### String er2 = "\\s+<td>";

### String er3 = "\\s+<a title=\"[a-zA-Z\\s\\-]+\" class=\"+[a-zA-Z0-9\\s\"=/><:\\.\\-]+\\?[a-z=0-9\"\\s]+([a-zA-Z-\\s]+)\"";

### String er4 = "<a class=\"[a-zA-Z\\s\\-\\\_]+\" id=\"+[a-zA-Z0-9\\s\"=/><:\\.\\-]+\\?[a-z=0-9\"\\s]+[a-zA-Z \\s=\"\\&]+\\;\"\\salt=\"([À-ÿA-Za-z\\s\\-]+)\"";

String Obtem\_DataNascimento(String pesquisa)

Devolve “não disponível” caso não haja uma data concreta na página do jogador ou então retorna uma variável do tipo string com a data encontrada.

### String er = "<td[^>]\*><a[^N]+Nascimentos\"[^\"]+\"([^\"]+)\"[^<]+</a>([\\sde]+)[^>]+>([0-9]+)</a>";

String Obtem\_Empresario(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o empresário do jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er = "<th>Empresários:</th>";

### String er1 = "\\s+<td>";

### String er2 = "\\s+<a href=\"[^>]\*>([À-ÿ0-9A-Za-z\\.\\s\\']+)</a>";

String Obtem\_EstadoAtual(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado.

### String er = "<span>Situação</span>([^<0-9-]+)[^>]+>";

String Obtem\_Fotografia(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado.

### String er = "<meta property=\"og:image\" content=\"([^\"]+)[^>]+>";

int Obtem\_Idade(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo inteiro com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

String er = "<meta name=\"description\" content=\"[a-zA-Z\\s,]+([0-9]+)[a-zA-Z\\s,]+\\b";

String er1 = "<th>Idade:</th>";

### String er2 = "<td>([0-9]+)</td>";

### String er3 = "<span class=\"dataItem\">Falecido:</span>";

### String er4 = "<span itemprop=\"deathDate\" class=\"dataValue\">[0-9\\s\\.]+\\s\\(([0-9]+)\\)</span>";

String Obtem\_Nacionalidade(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er = "<meta name=\"description\" content=\"[a-zA-Z\\s,0-9]+:\\s([a-zA-ZÀ-ÿ]+)\\b";

### String er1 = "&nbsp;&nbsp;([^<]\*)</td>";

String Obtem\_NomeCompleto(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er = "<th[^>]\*>Nome completo";

### String er1 = "</th>";

### String er2 = "<td[^>]\*>([a-zA-ZÀ-ÿ\\s]+)";

String Obtem\_PePreferido(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er = "\\s+<th>Pé:</th>";

### String er1 = "<td>([a-z]\*)</td>";

String Obtem\_Peso(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado.

String er = "<span>Peso</span>([0-9\\skg]+)</div>";

String Obtem\_Posicao(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er = "<span class=\"dataItem\">Posição:</span>";

### String er1 = "[\\s]\*<span class=\"dataValue\">[\\s]\*";

### String er2 = "\\s+([À-ÿA-Z-a-z\\s]+)\\b\\s+</span>";

String Obtem\_Ranking(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado.

### String er = "<span class=\"dataRN\">([^<]+)</span>";

String Obtem\_Selecao(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado. Usa as seguintes ERS, pois no website estava separado por linhas e/ou havia certas páginas com formatação diferente.

### String er0 = "<th colspan=\"[0-9]+\" >Seleção nacional</th>";

### String er = "<a class=[^>]\*>([^<]+)</a>";

ArrayList<String> Obtem\_Trofeus(String pesquisa)

Devolve um array com os 4 mais recentes troféus que o jogador tem.

String er = "<img src=\"[^\"]+\"\\stitle=\"([^\"]+)\"\\salt=\"[^\"]+\"\\sclass=\"dataErfolgImage\" />"

String Obtem\_ValorContrato(String pesquisa)

Devolve “não disponível” caso não haja informação sobre o jogador ou então retorna uma variável do tipo string com o nome encontrado.

### String er = "<meta name=\"description\" content=\"[^V]+[V]\*[^V]+ Valor de Mercado:\\s([0-9\\s\\,a-zA-Z€]+)";

# Modelo XML

Em relação ao modelo XML foram criadas 6 funcões:

Document adicionaJogador(Jogador j, Document doc)

Esta função é responsável pela criação do ficheiro XML, caso não exista. Se já existir o ficheiro então adiciona mais um elemento “jogador” à árvore armazenada no ficheiro.

Document removeJogadorNome(String nome, Document doc)

Esta função recebe como argumentos uma string que corresponde ao que o

utilizador introduziu, e um documento com ficheiro XML, procura o jogador na árvore e caso o encontre eliminar esse nó filho avisando o utilizador. No fim da execução

devolve um documento com o ficheiro XML atualizado.

Document alteraIdade(String nome, int novaIdade, Document doc)

A função recebe uma string com o nome do jogador introduzido pelo utilizador,  
um inteiro com o nova idade, também introduzido pelo utilizador, e um documento com o ficheiro XML. Procura na árvore o nó correspondente ao jogador indicado e atualiza o atributo “idade” com o nova idade , no fim da execução a função devolve o  
documento com o ficheiro XML atualizado.

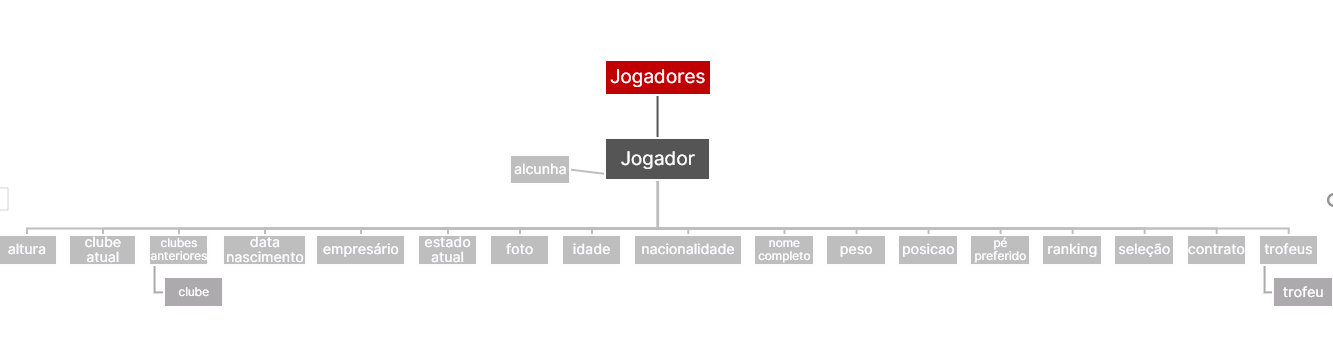
Document alteraNacionalidade(String nome, String novaNacionalidade, Document doc)

A função recebe uma string com o nome do jogador introduzido pelo utilizador,  
um inteiro com o nova idade, também introduzido pelo utilizador, e um documento com o ficheiro XML. Procura na árvore o nó correspondente ao jogador indicado e atualiza o atributo “idade” com a nova idade , no fim da execução a função devolve o  
documento com o ficheiro XML atualizado.

Document alteraClubeAtual(String nome, String novoClube, Document doc)

A função recebe uma string com o nome do jogador introduzido pelo utilizador,  
uma string com o novo clube, também introduzido pelo utilizador, e um documento com o ficheiro XML. Procura na árvore o nó correspondente ao jogador indicado e atualiza o atributo “clubeAtual” com o novo clube, no fim da execução a função devolve o documento com o ficheiro XML atualizado.

Document alteraEstado(String nome, String novoEstado, Document doc)

A função recebe uma string com o nome do jogador introduzido pelo utilizador,  
uma string com o novo estado, também introduzido pelo utilizador, e um documento com o ficheiro XML. Procura na árvore o nó correspondente ao jogador indicado e atualiza o atributo “estadoAtual” com o novo estado, no fim da execução a função devolve o documento com o ficheiro XML atualizado.

# Pesquisas XPATH

Para que quando se procurasse informação sobre um jogador conseguisse visualizar sobre o que era, adaptei uma função dada nas aulas práticas. Assim, guardei num *array* a identificação de todos os nós do XML e à medida que fazia um *append* de uma nova linha ao que era encontrado, era adicionado previamente a identificação.

Pesquisa por nome

### String xp = "//jogador[@nome ='" + jTextField11.getText() + "']//text()";

Pesquisa por clube

### String xp = "//jogador[contains(clubeAtual,'" + jTextField12.getText() + "')]//@nome";

Pesquisa por nacionalidade

### String xp = "//jogador[contains(nacionalidade,'" + jTextField13.getText() + "')]//@nome";

Pesquisa por posição

### String xp = "//jogador[contains(posicao,'" + jTextField14.getText() + "')]//@nome";

Pesquisa por seleção

### String xp = "//jogador[contains(selecao,'" + jTextField16.getText() + "')]//@nome";

Pesquisa por intervalo de idade

### String xp = "//jogador[(idade >= " + jTextField17.getText() + " and idade <= " + jTextField18.getText() + ")]//@nome";

Pesquisa por situação atual

String xp = "//jogador[not(contains(estadoAtual,'No ativo'))]//@nome";

# XSLT

XML para HTML

## Ficheiro HTML com fotos dos jogadores

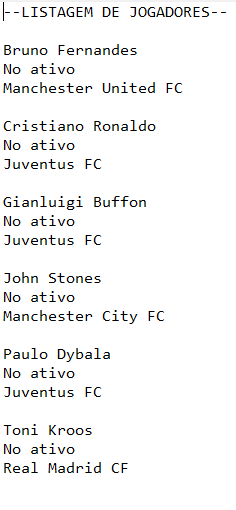
XML para XML

## Ficheiro com Top 5 dos jogadores mais valiosos

Devolve um ficheiro XML com a seguinte estrutura:

XML para XML

## Ficheiro com jogadores ordenados por ordem de idade

XML para TXT

## Ficheiro com clubes atuais e situação dos jogadores

# XQUERY

XML para XML

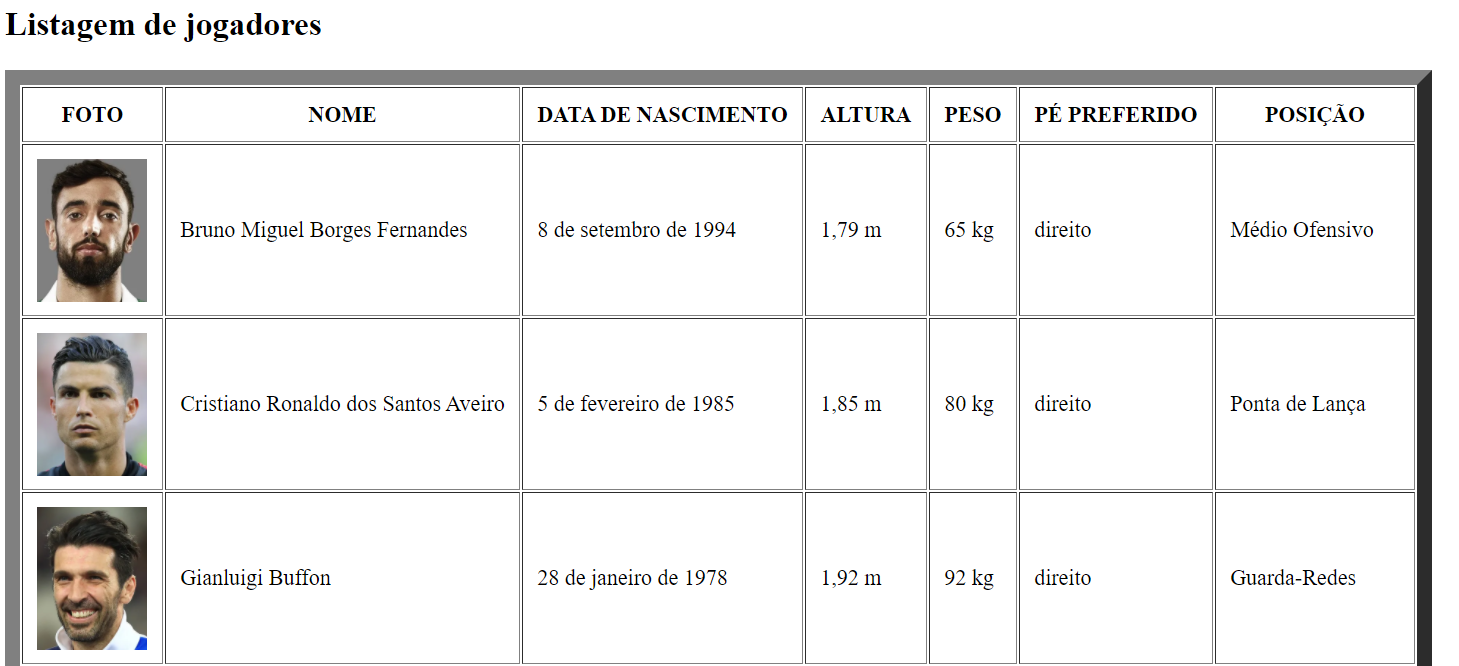
## Ficheiro XML com lista de jogadores de um clube

XML para TXT

## Ficheiro com jogadores de uma nacionalidade

XML para HTML

## Ficheiro com alguns dados mais importantes de jogadores





XML para HTML

## Ficheiro com lista de trofeus de jogadores