



Instituto Politécnico
de Viana do Castelo

ipvc Instituto Politécnico
de Viana do Castelo



Escola Superior
de Tecnologia e Gestão

Versão 1 – Documento em Atualização

Integração de Sistemas e Tecnologias

Interoperabilidade de Sistemas e Tecnologias

Tutorial 9 – WebServices, XML, SOAP, JavaScript,
Python e com ligação a Base de Dados.

Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Viana do Castelo
Braga, Portugal

Jorge Ribeiro

jribeiro@estg.ipvc.pt



Informação Apresentada na explicação do Tutorial 9:

- Exercício – DOM (simple.xml, books.xml, medsamp.xml)
- Exercício – SAX (simple.xml, books.xml, medsamp.xml, note.xml)
- Exercício – JDOM (books.xml, medsamp.xml, note.xml)
- Exercício – JAXP (simple.xml, books.xml, medsamp.xml)
- Exercício – JAXB (po.xml)
- SOAP – JAVA:
 - Exemplo nº1 – Adicionar dois números,
 - Exemplo nº2 – Devolver o ficheiro Books.xml para leitura
 - Exemplo nº3 – Devolver dados da base de dados
- SOAP – PHP:
 - Exemplo nº1: Exemplo dado pelo sitepoint.com
 - Exemplo nº2: Devolver o ficheiro Books.xml para leitura
 - Exemplo nº3: Devolver registos da base de dados em XML
- REST – JAVA
 - Exemplo nº1: Devolver o ficheiro Books.xml para leitura
 - Exemplo nº2: Acesso à base de dados e devolução de registos em XML
- REST – PHP
 - Exemplo nº1: Acesso à base de dados e devolução de registos em XML
 - Exemplo nº2: Enviar conteúdo do ficheiro Books.xml
- JAVA – SOAP – DOM, SAX, JDOM, JAXP, JAXB
- JAVA – REST – DOM, SAX, JDOM, JAXP, JAXB
- PHP – SOAP – DOM, SAX, JDOM, JAXP, JAXB
- PHP – REST – DOM, SAX, JDOM, JAXP, REST - JAXB
- PHP – SOAP – JDOM com FileUpload



Tarefas acrescidas:

1. Publicação do projeto java (sem código fonte) no servidor Tomcat;
2. Exploração (básica) do XPATH em ficheiros XML;
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação;
4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

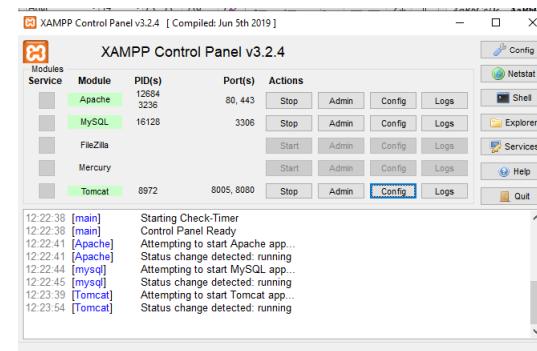
1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

- 1) **Publicação de projeto java (sem código fonte) no servidor Tomcat:** Tendo sido executados os projetos em JAVA, através do modo de desenvolvimento (NetBeans e servidor GlassFish), pretende-se que seja publicado o projeto num servidor web (fora do ambiente de desenvolvimento). Neste caso e uma vez que se está a utilizar o XAMPP, a publicação deverá ser instalada no servidor “Tomcat” de modo a interpretar os ficheiros .jar ou .war exportados a partir do IDE de desenvolvimento.

Algumas orientações:

- Deployment of an Example Application into Apache Tomcat Container: <https://linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container>

Com a realização desta tarefa passará a ser possível executar os projetos do tutorial 8, em JAVA e em PHP diretamente na plataforma XAMPP (sendo que na particularidade do Java, a parte de desenvolvimento estará sempre protegida a nível do código fonte e no servidor Tomcat apenas as classes):





1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

Seguir o exemplo “Deployment of an Example Application into Apache Tomcat Container” no link:

<https://linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container>

The screenshot shows a web browser displaying the LinuxConfig.org website. The URL in the address bar is "linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container". The page content is titled "Deployment of an Example Application into Apache Tomcat Container". Below the title, there is a section titled "Objective" which states: "Our objective is to develop a simple Java Servlet application using Netbeans IDE, and deploy it into a Tomcat application container using command line and the manager application." The LinuxConfig.org logo is visible on the left, and a navigation menu with links like "Ubuntu", "Debian", "Redhat / CentOS / AlmaLinux", "Fedora", "Kali Linux", "Linux Tutorials", "System Admin", "Programming", "Multimedia", "Forum", and "Linux C" is at the top.

Deployment of an Example Application into Apache Tomcat Container

30 October 2018 by Admin

Objective

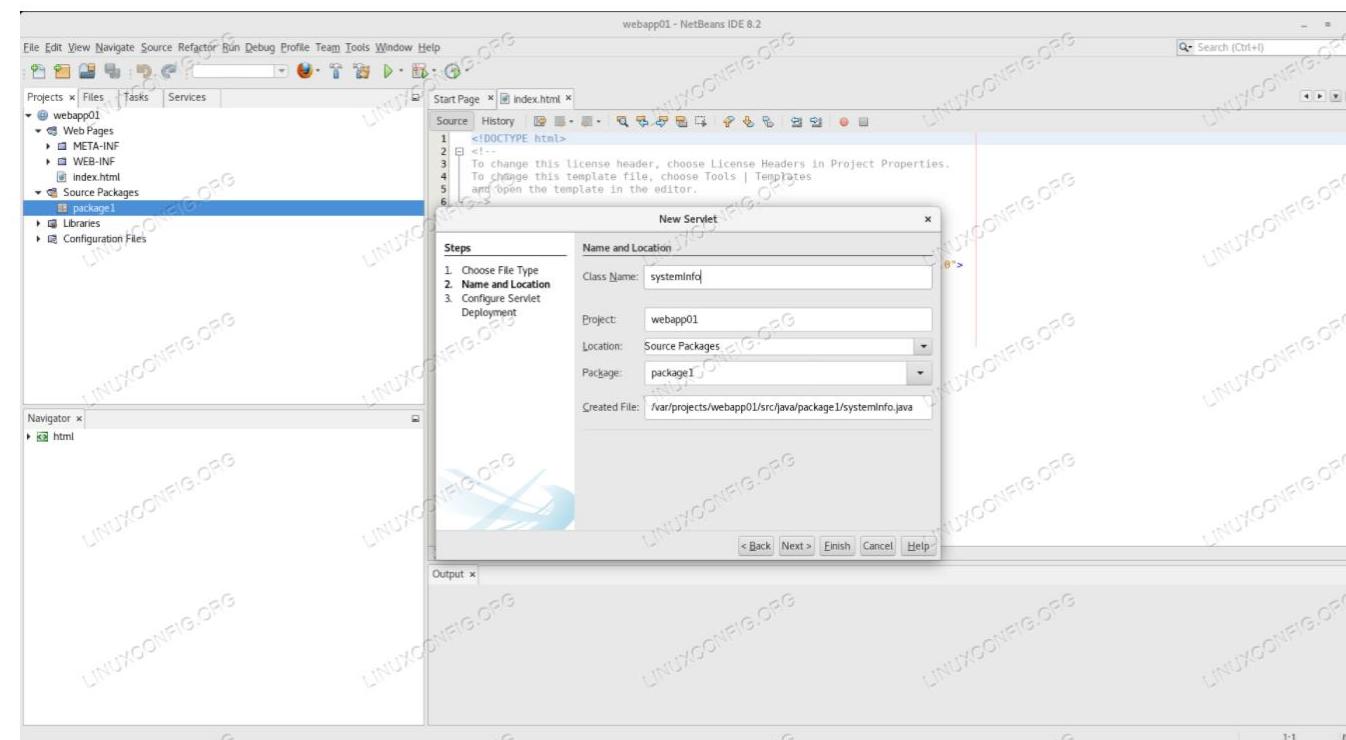
Our objective is to develop a simple Java Servlet application using Netbeans IDE, and deploy it into a Tomcat application container using command line and the manager application.

1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

Seguir o exemplo “Deployment of an Example Application into Apache Tomcat Container” no link:

<https://linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container>

← → C 🔒 linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container
development) on the project path.

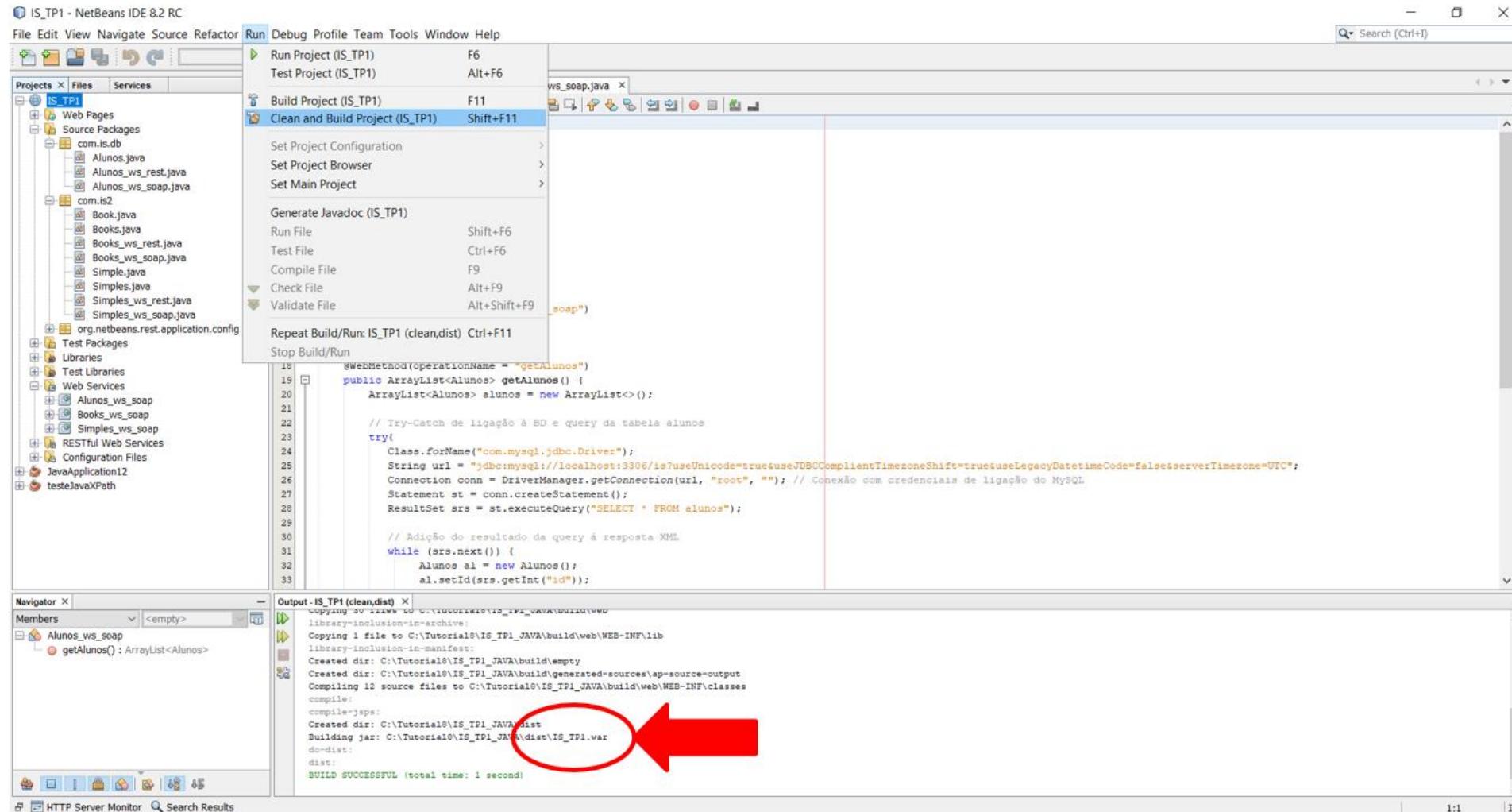




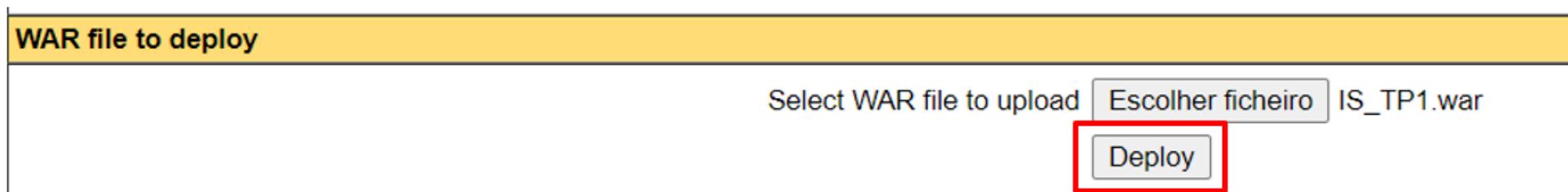
1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

- A primeira etapa deste processo é limpar e construir o projeto, permitindo criar um arquivo **war** que pode ser carregado na interface de gestão de aplicações web do servidor Tomcat.
- A segunda etapa é selecionar o arquivo war e clicar no botão de “deploy”.
- A etapa final é verificar na interface de gestão do Tomcat se o projeto está implantado no servidor. Depois apenas basta abrir um browser no link do serviço:

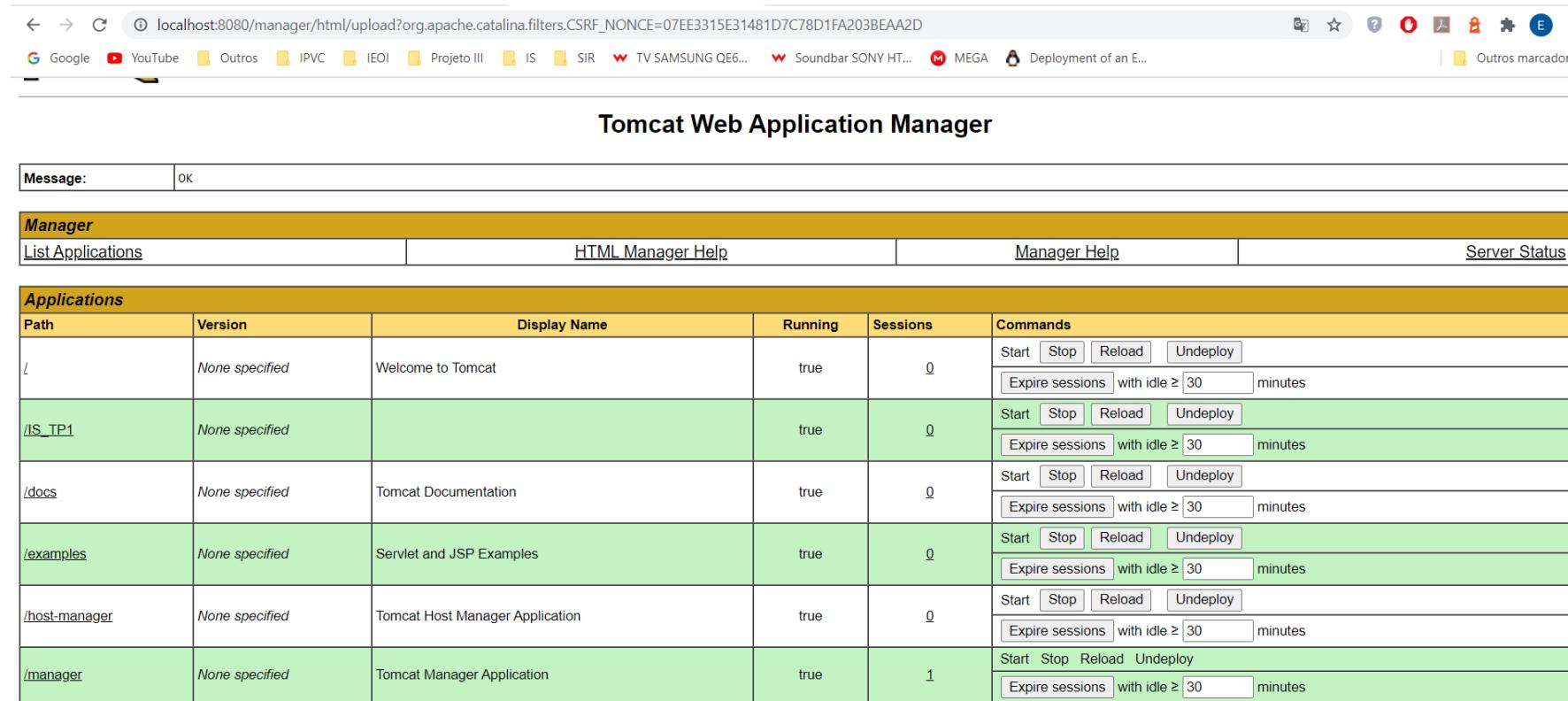
1. Publicação do projeto java (sem código fonte) no servidor Tomcat;



1. Publicação do projeto java (sem código fonte) no servidor Tomcat;



1. Publicação do projeto java (sem código fonte) no servidor Tomcat;



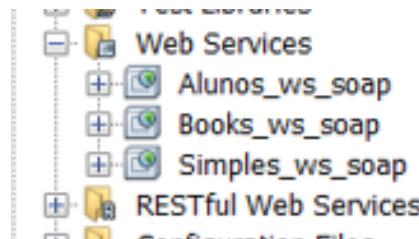
The screenshot shows the Tomcat Web Application Manager interface. The title bar reads "Tomcat Web Application Manager". A message box at the top left says "Message: OK". Below the header, there are tabs: "Manager", "List Applications", "HTML Manager Help", "Manager Help", and "Server Status". The main content area is titled "Applications" and contains a table with the following data:

Path	Version	Display Name	Running	Sessions	Commands
/	<i>None specified</i>	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/IS_TP1	<i>None specified</i>		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	<i>None specified</i>	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	<i>None specified</i>	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	<i>None specified</i>	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	<i>None specified</i>	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes



1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

Abrir o browser no link: http://localhost:8080/IS_TP1/Simples_ws_soap?Tester



Ou seja, na designação do web service do projeto, consoante a implementação de cada aluno:

The screenshot shows a browser window with the URL `localhost:8080/IS_TP1_JAVA/Books_ws_soap?Tester`. The page displays a SOAP Request and its corresponding SOAP Response.

Method parameter(s)

Type	Value
------	-------

Method returned

```
java.util.List<[com.is2.Book@45afaf5e, com.is2.Book@7ede04ed, com.is2.Book@c17e296, com.is2.Book@6892ee35, com.is2.Book@50ff6926, com.is2.Book@4e9ae4b8, com.is2.Book@7dcom.is2.Book@2b61669, com.is2.Book@292be940, com.is2.Book@300d67f9, com.is2.Book@186aea93, com.is2.Book@2a5db2a9]'
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://is2.com/">
    <ns2:getBooks/>
</S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body xmlns:ns2="http://is2.com/">
    <ns2:getBooksResponse>
        <return>
            <author>Gambardella, Matthew</author>
            <title>XML Developer's Guide</title>
        </return>
    </ns2:getBooksResponse>
</S:Body>
</S:Envelope>
```



2. Exploração (básica) do XPATH em ficheiros XML;

- **Exploração (básica) do XPATH em ficheiros XML:** Com base no tutorial 8 e selecionando uma das implementações em Java e em PHP para leitura de ficheiros XML, deverá alterar o código do lado do servidor para proceder a pesquisas/filtros de informação dentro do ficheiro XML através do XPATH. Ex. com base no ficheiro “books.xml”, filtrar só os livros com o “Corets, Eva” e “Thurman, Paula”.
- Terá de se retificar o código tanto em Java como em PHP dos pontos do tutorial anterior:

2. Exploração (básica) do XPATH em ficheiros XML;

Algumas orientações:

- Java [XPath Example](#) – [XPath Tutorial](#): <https://howtodoinjava.com/java/xml/java-xpath-tutorial-example/>
- Java [XPath Parser](#) - [Parse XML Document](#):
https://www.tutorialspoint.com/java_xml/java_xpath_parse_document.htm
- PHP [XPath Parser](#) - <https://www.php.net/manual/en/class.domxpath.php>
- PHP XPath Parser - <https://www.php.net/manual/en/simplexmlelement.xpath.php>
- PHP XPATH Parser - <https://www.ccbtutorials.com/parse-xml-using-xpath-php/>

2. Exploração (básica) do XPATH em ficheiros XML

- Importação do ficheiro books.xml, via REST e Parser: JAXB

Retificação necessária ao código:



```
Books_ws_rest.java x Book.java x Alunos_ws_soap.java x
Source History | Back Forward | Search | Open | Save | Print | Help | Exit

public ArrayList<Book> getBooks() throws JAXBException, InvocationTargetException, XPathExpressionException, ParserConfigurationException, SAXException, IOException {
    // Try-Catch do parsing via JAXB do ficheiro
    try {
        JAXBContext jaxbContext; // Parser usado (JAXB)
        File xmlFile = new File("C:\\Tutorial0\\FicheirosXML\\books.xml"); // Cria variável de ficheiro local indicando o path do ficheiro "books.xml"
        jaxbContext = JAXBContext.newInstance(Books.class); // Cria instance do parse do JAXB na classe da lista de livros
        Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller(); // Cria interface responsável para "des-serializar" o ficheiro XML para objeto Java
        Books books = (Books) jaxbUnmarshaller.unmarshal(xmlFile); // Des-serializa o ficheiro XML para objeto Java

        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder;

        dBuilder = dbFactory.newDocumentBuilder();

        Document doc = dBuilder.parse(xmlFile);
        doc.getDocumentElement().normalize();

        XPath xPath = XPathFactory.newInstance().newXPath();

        String expression = "/catalog/book[author='Corets, Eva' or author='Thurman, Paula']";
        NodeList nodeList = (NodeList) xPath.compile(expression).evaluate(
            doc, XPathConstants.NODESET);

        ArrayList<String> booksToKeep = new ArrayList<>();
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node nNode = nodeList.item(i);

            if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) nNode;
                booksToKeep.add(eElement.getElementsByTagName("title").item(0).getTextContent());
            }
        }

        ArrayList<Book> listaBooksFiltrada = new ArrayList<>();
        for(String title : booksToKeep) {
            for(Book book : books.getBooks()) {
                if(book.getTitle().equals(title)) {
                    listaBooksFiltrada.add(book);
                }
            }
        }
    }
}
```





2. Exploração (básica) do XPATH em ficheiros XML;

Resultado do webservice

The screenshot shows a web-based interface for testing RESTful web services. The title bar says "Test RESTful Web Services". The URL in the address bar is "localhost:8080/IS_TP1_JAVA/test-resbeans.html". On the left, there's a sidebar with a tree view containing "alunos", "books", and "simples". The main area displays the following information:

- Resource:** books (http://localhost:8080/IS_TP1_JAVA/webresources/books)
- Choose method to test:**
- Status:** 200 (OK)
- Response:**

The "Raw View" tab is selected, showing the XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book>
    <author>Corets, Eva</author>
    <title>Maeve Ascendant</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-11-17</publish_date>
    <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
  </book>
  <book>
    <author>Corets, Eva</author>
    <title>Oberon's Legacy</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-03-10</publish_date>
    <description>In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.</description>
  </book>
  <book>
    <author>Corets, Eva</author>
    <title>The Sundered Grail</title>
  </book>
</books>
```



2. Exploração (básica) do XPATH em ficheiros XML;

- Em PHP, Implementação de webservice via SOAP para leitura de ficheiro books.xml usando o parser DOM

Retificação do código:

```
<?php
function getDataFile() {
    $doc = new DOMDocument;
    $doc->load("files/books.xml");

    $xpath = new DOMXPath($doc);
    $resultado = $xpath->query('/catalog/book[author="Corets, Eva" or author="Thurman, Paula"]');
    $data = "";
    if ( $resultado->length < 1 ) {
        $data = "Não foram encontrados nenhum resultados";
    } else {
        foreach($resultado as $book) {
            $autor = $book->getElementsByTagName("author")[0]->nodeValue;
            $titulo = $book->getElementsByTagName("title")[0]->nodeValue;
            $genero = $book->getElementsByTagName("genre")[0]->nodeValue;
            $preco = $book->getElementsByTagName("price")[0]->nodeValue;
            $dataPub = $book->getElementsByTagName("publish_date")[0]->nodeValue;
            $descricao = $book->getElementsByTagName("description")[0]->nodeValue;
            $data .= 'Autor: '.$autor."<br />";
            $data .= 'Título: '.$titulo."<br />";
            $data .= 'Genero: '.$genero."<br />";
            $data .= 'Preço: '.$preco."<br />";
            $data .= 'Data de Publicação: '.$dataPub."<br />";
            $data .= 'Descrição: '.$descricao."<br />";
            $data .= '<hr />';
        }
    }
    return $data;
}
```

2. Exploração (básica) do XPATH em ficheiros XML;

- Em PHP, Implementação de webservice via SOAP para leitura de ficheiro books.xml usando o parser DOM

Retificação do código:

```
1 <?php
2     function getDataFile() {
3         $doc = new DOMDocument;
4         $doc->load("files/books.xml");
5
6         $xpath = new DOMXPath($doc);
7         $resultado = $xpath->query('/catalog/book[author="Corets, Eva" or author="Thurman, Paula"]');
8         $data = "";
9         if ( $resultado->length < 1 ) {
10             $data = "Não foram encontrados nenhuns resultados";
11         }
12         else {
13             foreach($resultado as $book) {
14                 $autor = $book->getElementsByTagName("author")[0]->nodeValue;
15                 $titulo = $book->getElementsByTagName("title")[0]->nodeValue;
16                 $genero = $book->getElementsByTagName("genre")[0]->nodeValue;
17                 $preco = $book->getElementsByTagName("price")[0]->nodeValue;
18                 $dataPub = $book->getElementsByTagName("publish_date")[0]->nodeValue;
19                 $descricao = $book->getElementsByTagName("description")[0]->nodeValue;
20                 $data .= 'Autor: '.$autor."<br />";
21                 $data .= 'Título: '.$titulo."<br />";
22                 $data .= 'Genero: '.$genero."<br />";
23                 $data .= 'Preço: '.$preco."<br />";
24                 $data .= 'Data de Publicação: '.$dataPub."<br />";
25                 $data .= 'Descrição: '.$descricao."<br />";
26                 $data .= '<hr />';
27             }
28         }
29     }
30 }
```



2. Exploração (básica) do XPATH em ficheiros XML;

Resultado da invocação do webservice:

localhost:8080/IS_TP1_PHP/SOAP

localhost:8080/IS_TP1_PHP/SOAP/DOM/client.php

string(1138) "Autor: Corets, Eva
Título: Maeve Ascendant
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2000-11-17
Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva
Título: Oberon's Legacy
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2001-03-10
Descrição: In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.

Autor: Corets, Eva
Título: The Sundered Grail
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2001-09-10
Descrição: The two daughters of Maeve, half-sisters, battle one another for control of England. Sequel to Oberon's Legacy.

Autor: Thurman, Paula
Título: Splish Splash
Genero: Romance
Preço: 4.95
Data de Publicação: 2000-11-02
Descrição: A deep sea diver finds true love twenty thousand leagues beneath the sea.

Network tab of developer tools showing 3 requests, 3.5 kB transferred, and 3.2 kB resources. The waterfall chart shows three requests: client..., extn-u..., and extn-u... with total times of approximately 100 ms, 200 ms, and 300 ms respectively.

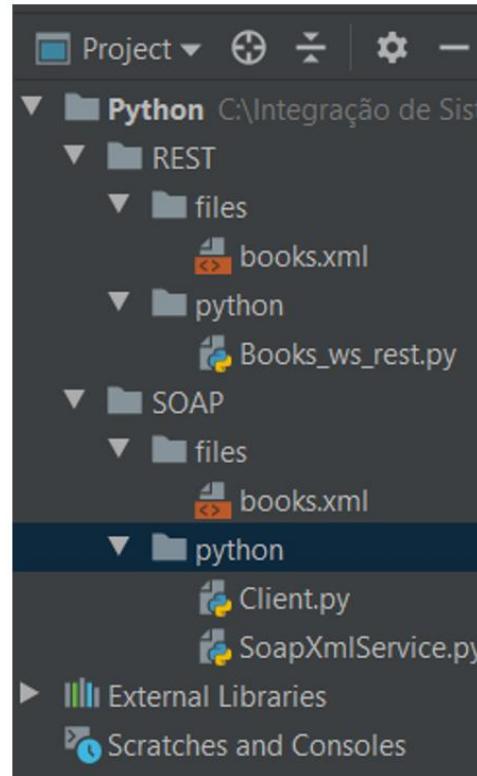


3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação;

IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:

NOTA: Para ilustrar a implementação desta tarefa foi usado o IDE PyCharm para a linguagem de programação Python e o Visual Studio Code para a linguagem de programação Javascript.

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**



Todos os serviços web REST em Python foram criados a partir da biblioteca Flask, que permite a criação de um servidor onde é possível cadastrar vários endpoints para estar pelo cliente, que neste caso é o navegador. Além disso, também foram utilizadas as bibliotecas lxml e xml.dom mais especificamente na análise e leitura do arquivo xml.



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:

- Nesta primeira implementação existem dois endpoints disponíveis, em que um deles executa a análise do arquivo books.xml, enquanto o segundo executa as partes do arquivo e usa expressões XPath para filtrar seu conteúdo, retornando apenas os que correspondem à expressão referida. Acessando ao endpoint raiz, dois links da web são mostrados e, se clicados, chamam o método correspondente no serviço da web.
- Importações, endpoint raiz (em '/'), função getBooks e o endpoint correspondente para analisar o arquivo books.xml, para tipo de argumento html.



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

```
Books_ws_rest.py x
1  from xml.dom import minidom
2  from flask import Flask, request
3  from lxml import etree
4  from werkzeug.wrappers import Response
5  app = Flask(__name__)
6
7
8  @app.route('/')
9  def index():
10     html = "<h3>Esta aplicacão contém os seguintes métodos: <h3>"
11     html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
12     html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
13     return html
14
15  @app.route('/getBooks/books', methods=['GET'])
16  def getBooks():
17      typeRequest = request.args.get('type')
18      if typeRequest.upper() == "HTML":
19          xmldoc = minidom.parse('../files/books.xml')
20          itemlist = xmldoc.getElementsByTagName('book')
21          data = ""
22          i = 0
23          for i in range(0, len(itemlist)):
24              node = itemlist.item(i)
25              autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
26              titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
27              genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
28              preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
29              data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
30              descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
31              data = data + "<h5>Autor: " + autor + "<h5>"
32              data = data + "<h5>Titulo: " + titulo + "<h5>"
```

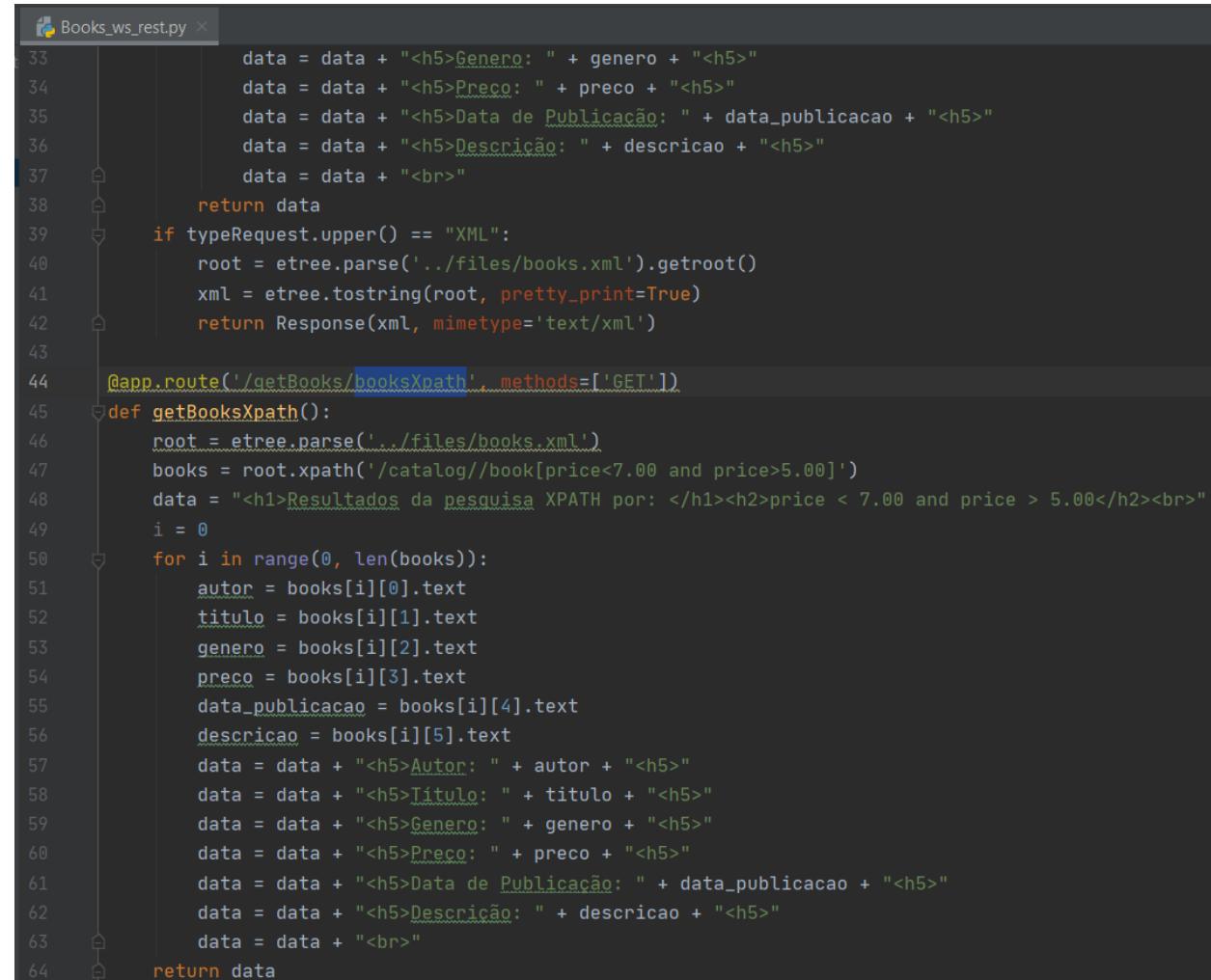
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

```
1  from xml.dom import minidom
2  from flask import Flask, request
3  from lxml import etree
4  from werkzeug.wrappers import Response
5  app = Flask(__name__)
6
7
8  @app.route('/')
9  def index():
10     html = "<h3>Esta aplicação contém os seguintes métodos: </h3>"
11     html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
12     html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
13     return html
14
15  @app.route('/getBooks/books', methods=['GET'])
16  def getBooks():
17      typeRequest = request.args.get('type')
18      if typeRequest.upper() == "HTML":
19          xmldoc = minidom.parse('../files/books.xml')
20          itemlist = xmldoc.getElementsByTagName('book')
21          data = ""
22          i = 0
23          for i in range(0, len(itemlist)):
24              node = itemlist.item(i)
25              autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
26              titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
27              genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
28              preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
29              data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
30              descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
31              data = data + "<h5>Autor: " + autor + "</h5>"
32              data = data + "<h5>Título: " + titulo + "</h5>"
33              data = data + "<h5>Genero: " + genero + "</h5>"
34              data = data + "<h5>Preço: " + preco + "</h5>"
35              data = data + "<h5>Data de Publicação: " + data_publicacao + "</h5>"
36              data = data + "<h5>Descrição: " + descricao + "</h5>"
37              data = data + "<br>"
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Retorno da string construída com html, solicitação do tipo xml e a função getBooksXpath para filtrar os resultados usando uma expressão Xpath.



```
Books_ws_rest.py
33     data = data + "<h5>Genero: " + genero + "<h5>"
34     data = data + "<h5>Preco: " + preco + "<h5>"
35     data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
36     data = data + "<h5>Descrição: " + descricao + "<h5>"
37     data = data + "<br>"
38     return data
39 if typeRequest.upper() == "XML":
40     root = etree.parse('../files/books.xml').getroot()
41     xml = etree.tostring(root, pretty_print=True)
42     return Response(xml, mimetype='text/xml')
43
44 @app.route('/getBooks/booksXpath', methods=['GET'])
45 def getBooksXpath():
46     root = etree.parse('../files/books.xml')
47     books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
48     data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"
49     i = 0
50     for i in range(0, len(books)):
51         autor = books[i][0].text
52         titulo = books[i][1].text
53         genero = books[i][2].text
54         preco = books[i][3].text
55         data_publicacao = books[i][4].text
56         descricao = books[i][5].text
57         data = data + "<h5>Autor: " + autor + "<h5>"
58         data = data + "<h5>Título: " + titulo + "<h5>"
59         data = data + "<h5>Genero: " + genero + "<h5>"
60         data = data + "<h5>Preco: " + preco + "<h5>"
61         data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
62         data = data + "<h5>Descrição: " + descricao + "<h5>"
63         data = data + "<br>"
64     return data
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Retorno da string construída com html, solicitação do tipo xml e a função getBooksXpath para filtrar os resultados usando uma expressão Xpath.

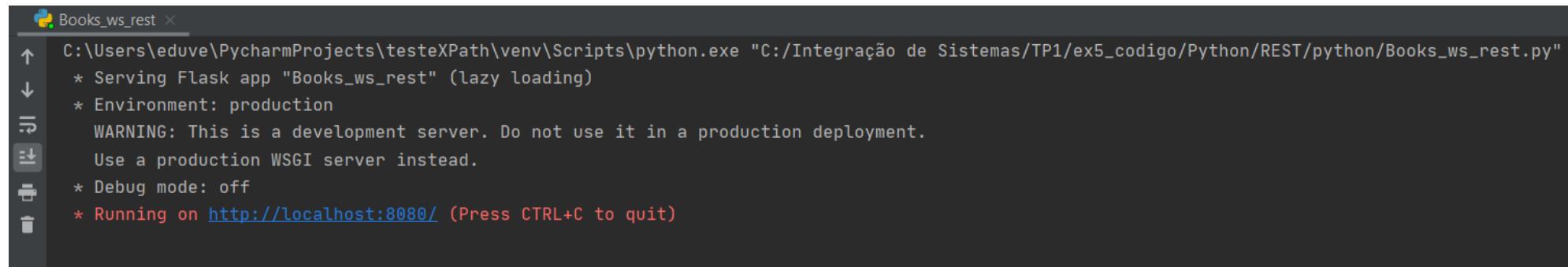
```
39     if typeRequest.upper() == "XML":
40         root = etree.parse('../files/books.xml').getroot()
41         xml = etree.tostring(root, pretty_print=True)
42         return Response(xml, mimetype='text/xml')
43
44 @app.route('/getBooks/booksXpath', methods=['GET'])
45 def getBooksXpath():
46     root = etree.parse('../files/books.xml')
47     books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
48     data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"
49     i = 0
50     for i in range(0, len(books)):
51         autor = books[i][0].text
52         titulo = books[i][1].text
53         genero = books[i][2].text
54         preco = books[i][3].text
55         data_publicacao = books[i][4].text
56         descricao = books[i][5].text
57         data = data + "<h5>Autor: " + autor + "<h5>"
58         data = data + "<h5>Título: " + titulo + "<h5>"
59         data = data + "<h5>Genero: " + genero + "<h5>"
60         data = data + "<h5>Preço: " + preco + "<h5>"
61         data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
62         data = data + "<h5>Descrição: " + descricao + "<h5>"
63         data = data + "<br>"
64     return data
65
66 if __name__ == "__main__":
67     app.run(host='localhost', port=8080)
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Função principal para iniciar o serviço:

```
65  
66 ► if __name__ == "__main__":  
67     app.run(host='localhost', port=8080)  
68
```

Serviço da Web em execução no localhost, porta 8080 e acessível no ponto de extremidade raiz por meio do navegador

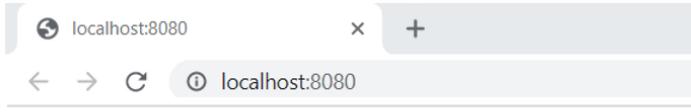


The screenshot shows the PyCharm terminal window with the title 'Books_ws_rest'. It displays the command run in the terminal: 'C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/TP1/ex5_codigo/Python/REST/python/Books_ws_rest.py"'. The output shows the Flask application starting up with the message: '* Serving Flask app "Books_ws_rest" (lazy loading)', '* Environment: production', 'WARNING: This is a development server. Do not use it in a production deployment.', 'Use a production WSGI server instead.', '* Debug mode: off', and '* Running on http://localhost:8080/ (Press CTRL+C to quit)'.



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Chamar o endpoint getBooks / books especificando o tipo de saída como html



Esta aplicação contém os seguintes métodos:

[XML Parse - Ficheiro books.xml](#)

Resposta em XML e HTML

[XML Parse e XPATH - Ficheiro books.xml](#)

Autor: Gambardella, Matthew
Titulo: XML Developer's Guide
Genero: Computer
Preço: 44.95
Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

Autor: Ralls, Kim
Titulo: Midnight Rain
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva
Titulo: Maeve Ascendant
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2000-11-17

Network tab in developer tools showing a request to http://localhost:8080/getBooks/books?type=html. Request Method: GET, Status Code: 200 OK, Response Headers include Content-Type: text/html; charset=utf-8, and Request Headers include Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3.



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Chamar o endpoint getBooks / books especificando o tipo de saída como xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
    <description>A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.</description>
  </book>
  <book id="bk103">
    <author>Corets, Eva</author>
    <title>Maeve Ascendant</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-11-17</publish_date>
    <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
  </book>
  <book id="bk104">
    <author>Corets, Eva</author>
    <title>Oberon's Legacy</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-03-10</publish_date>
    <description>In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.</description>
  </book>
  <book id="bk105">
    <author>Corets, Eva</author>
    <title>The Sundered Grail</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-09-10</publish_date>
    <description>The two daughters of Maeve, half-sisters, battle one another for control of England. Sequel to Oberon's Legacy.</description>
  </book>
```

Network tab details:

- Request URL: http://localhost:8080/getBooks/books?type=xml
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 127.0.0.1:8080
- Referrer Policy: strict-origin-when-cross-origin
- Content-Length: 4408
- Content-Type: text/xml; charset=utf-8
- Date: Mon, 23 Nov 2020 15:55:28 GMT
- Server: Werkzeug/1.0.1 Python/3.8.2
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exct=0.9
- Accept-Encoding: gzip, deflate, br
- Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
- Connection: keep-alive
- Host: localhost:8080
- sec-ch-ua: "Chromium";v="86", "Not\A;Brand";v="99", "Google";v="86"
- sec-ch-ua-mobile: ?0

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Chamar o endpoint getBooks / booksXpath e o resultado obtido

```
        return Response(xml, mimetype='text/xml')

@app.route('/getBooks/booksXpath', methods=['GET'])
def getBooksXpath():
    root = etree.parse('../files/books.xml')
    books = root.xpath('/catalog/book[price<7.00 and price>5.00]')
    data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><table border='1'><thead><tr><th>Autor</th><th>Título</th><th>Género</th><th>Preço</th><th>Data Publicação</th></tr></thead><tbody>"
    i = 0
    for i in range(0, len(books)):
        autor = books[i][0].text
        titulo = books[i][1].text
        genero = books[i][2].text
        preco = books[i][3].text
        data_publicacao = books[i][4].text
        data += "<tr><td>" + autor + "</td><td>" + titulo + "</td><td>" + genero + "</td><td>" + preco + "</td><td>" + data_publicacao + "</td></tr>
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

Chamar o endpoint getBooks / booksXpath e o resultado obtido

```
44 @app.route('/getBooks/booksXpath', methods=['GET'])
45 def getBooksXpath():
46     root = etree.parse('../files/books.xml')
47     books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
48     data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 na price > 5.00</h2><br>"
49     i = 0
50     for i in range(0, len(books)):
51         autor = books[i][0].text
52         titulo = books[i][1].text
53         genero = books[i][2].text
54         preco = books[i][3].text
55         data_publicacao = books[i][4].text
56         descricao = books[i][5].text
57         data = data + "<h5>Autor: " + autor + "<h5>"
58         data = data + "<h5>Título: " + titulo + "<h5>"
59         data = data + "<h5>Genero: " + genero + "<h5>"
60         data = data + "<h5>Preço: " + preco + "<h5>"
61         data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
62         data = data + "<h5>Descrição: " + descricao + "<h5>"
63         data = data + "<br>"
64     return data
65
66 if __name__ == "__main__":
67     app.run(host='localhost', port=8080)
```





3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

localhost:8080/getBooks/booksXpath

localhost:8080/getBooks/booksXpath

Resultados da pesquisa XPATH por:

price < 7.00 and price > 5.00

Autor: Ralls, Kim

Título: Midnight Rain

Gênero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva

Título: Maeve Ascendant

Gênero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva

Network

Request URL: http://localhost:8080/getBooks/booksXpath

Request Method: GET

Status Code: 200 OK

Remote Address: 127.0.0.1:8080

Referrer Policy: strict-origin-when-cross-origin

Content-Length: 1614

Content-Type: text/html; charset=utf-8

Date: Mon, 23 Nov 2020 15:56:29 GMT

Server: Werkzeug/1.0.1 Python/3.8.2

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7

Connection: keep-alive

Host: localhost:8080

sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google";v="86"

sec-ch-ua-mobile: ?0

3 requests | 3.8 kB transferred



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA REST USANDO XPATH:**

localhost:8080/getBooks/booksXpath

localhost:8080/getBooks/booksXpath

Resultados da pesquisa XPATH por:

price < 7.00 and price > 5.00

Autor: Ralls, Kim

Titulo: Midnight Rain

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva

Titulo: Maeve Ascendant

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva

Network

Request URL: http://localhost:8080/getBooks/booksXpath

Request Method: GET

Status Code: 200 OK

Remote Address: 127.0.0.1:8080

Referrer Policy: strict-origin-when-cross-origin

Content-Length: 1614

Content-Type: text/html; charset=utf-8

Date: Mon, 23 Nov 2020 15:56:29 GMT

Server: Werkzeug/1.0.1 Python/3.8.2

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7

Connection: keep-alive

Host: localhost:8080

sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google";v="86"

sec-ch-ua-mobile: ?0

3 requests | 3.8 kB transferred



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:

- Nota: Para implementar via SOAP usando Python foi usada a Framework Spyne, que permite criar um servidor, registrar os métodos criados e mapear os tipos de saída e entrada do serviço web, que neste caso são ambos SOAP11.
- Importações necessárias, função books_parse, o código necessário usando minidom para analisar o arquivo e obter uma lista de elementos e a construção da string html para enviar de volta ao cliente.



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

```
SoapXmlService.py x
1  from spyne import Application, rpc, ServiceBase, Unicode
2  from spyne.protocol.soap import Soap11
3  from spyne.server.wsgi import WsgiApplication
4  from xml.dom import minidom
5  from lxml import etree
6
7
8  class SoapXmlService(ServiceBase):
9      @rpc(_returns=Unicode)
10     def books_parse(ctx):
11         xmldoc = minidom.parse('../files/books.xml')
12         itemlist = xmldoc.getElementsByTagName('book')
13         data = ""
14         i = 0
15         for i in range(0, len(itemlist)):
16             node = itemlist.item(i)
17             autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
18             titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
19             genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
20             preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
21             data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
22             descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
23             data = data + "<h5>Autor: " + autor + "<h5>"
24             data = data + "<h5>Título: " + titulo + "<h5>"
25             data = data + "<h5>Genero: " + genero + "<h5>"
26             data = data + "<h5>Preco: " + preco + "<h5>"
27             data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
28             data = data + "<h5>Descrição: " + descricao + "<h5>"
29             data = data + "<br>"
30
31     return data
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:

```
1  from spyne import Application, rpc, ServiceBase, Unicode
2  from spyne.protocol.soap import Soap11
3  from spyne.server.wsgi import WsgiApplication
4  from xml.dom import minidom
5  from lxml import etree
6
7
8  class SoapXmlService(ServiceBase):
9      @rpc(_returns=Unicode)
10     def books_parse(ctx):
11         xmldoc = minidom.parse('../files/books.xml')
12         itemlist = xmldoc.getElementsByTagName('book')
13         data = ""
14         i = 0
15         for i in range(0, len(itemlist)):
16             node = itemlist.item(i)
17             autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
18             titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
19             genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
20             preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
21             data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
22             descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
23             data = data + "<h5>Autor: " + autor + "<h5>"
24             data = data + "<h5>Título: " + titulo + "<h5>"
25             data = data + "<h5>Genero: " + genero + "<h5>"
26             data = data + "<h5>Preço: " + preco + "<h5>"
27             data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
28             data = data + "<h5>Descrição: " + descricao + "<h5>"
29             data = data + "<br>"
30
31     return data
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

Segundo método, books_Xpath, usando etree da biblioteca lxml para filtrar todos os livros no arquivo books.xml e retornar apenas a lista de elementos que verificam a expressão, construindo o html para enviar para o cliente.

```
SoapXmlService.py ×
31
32     @rpc(_returns=Unicode)
33     def books_Xpath(ctx):
34         root = etree.parse('../files/books.xml')
35         books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
36         data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"
37
38         i = 0
39         for i in range(0, len(books)):
40             autor = books[i][0].text
41             titulo = books[i][1].text
42             genero = books[i][2].text
43             preco = books[i][3].text
44             data_publicacao = books[i][4].text
45             descricao = books[i][5].text
46             data = data + "<h5>Autor: " + autor + "<h5>"
47             data = data + "<h5>Título: " + titulo + "<h5>"
48             data = data + "<h5>Genero: " + genero + "<h5>"
49             data = data + "<h5>Preco: " + preco + "<h5>"
50             data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
51             data = data + "<h5>Descrição: " + descricao + "<h5>"
52             data = data + "<br>"
53
54
55         application = Application([SoapXmlService],
56                                 tns='soap.books.parse',
57                                 in_protocol=Soap11(validation='lxml'),
58                                 out_protocol=Soap11()
59                                 )
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

Segundo método, books_Xpath, usando etree da biblioteca lxml para filtrar todos os livros no arquivo books.xml e retornar apenas a lista de elementos que verificam a expressão, construindo o html para enviar para o cliente.

```
32 @rpc(_returns=Unicode)
33 def books_Xpath(ctx):
34     root = etree.parse('../files/books.xml')
35     books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
36     data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"
37     i = 0
38     for i in range(0, len(books)):
39         autor = books[i][0].text
40         titulo = books[i][1].text
41         genero = books[i][2].text
42         preco = books[i][3].text
43         data_publicacao = books[i][4].text
44         descricao = books[i][5].text
45         data = data + "<h5>Autor: " + autor + "<h5>"
46         data = data + "<h5>Título: " + titulo + "<h5>"
47         data = data + "<h5>Genero: " + genero + "<h5>"
48         data = data + "<h5>Preço: " + preco + "<h5>"
49         data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
50         data = data + "<h5>Descrição: " + descricao + "<h5>"
51         data = data + "<br>"
52     return data
53
54
55 application = Application([SoapXmlService],
56                             tns='soap.books.parse',
57                             in_protocol=Soap11(validation='lxml'),
58                             out_protocol=Soap11()
59                             )
60 if __name__ == '__main__':
61     from wsgiref.simple_server import make_server
62
63     wsgi_app = WsgiApplication(application)
64     server = make_server('localhost', 8000, wsgi_app)
65     server.serve_forever()
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

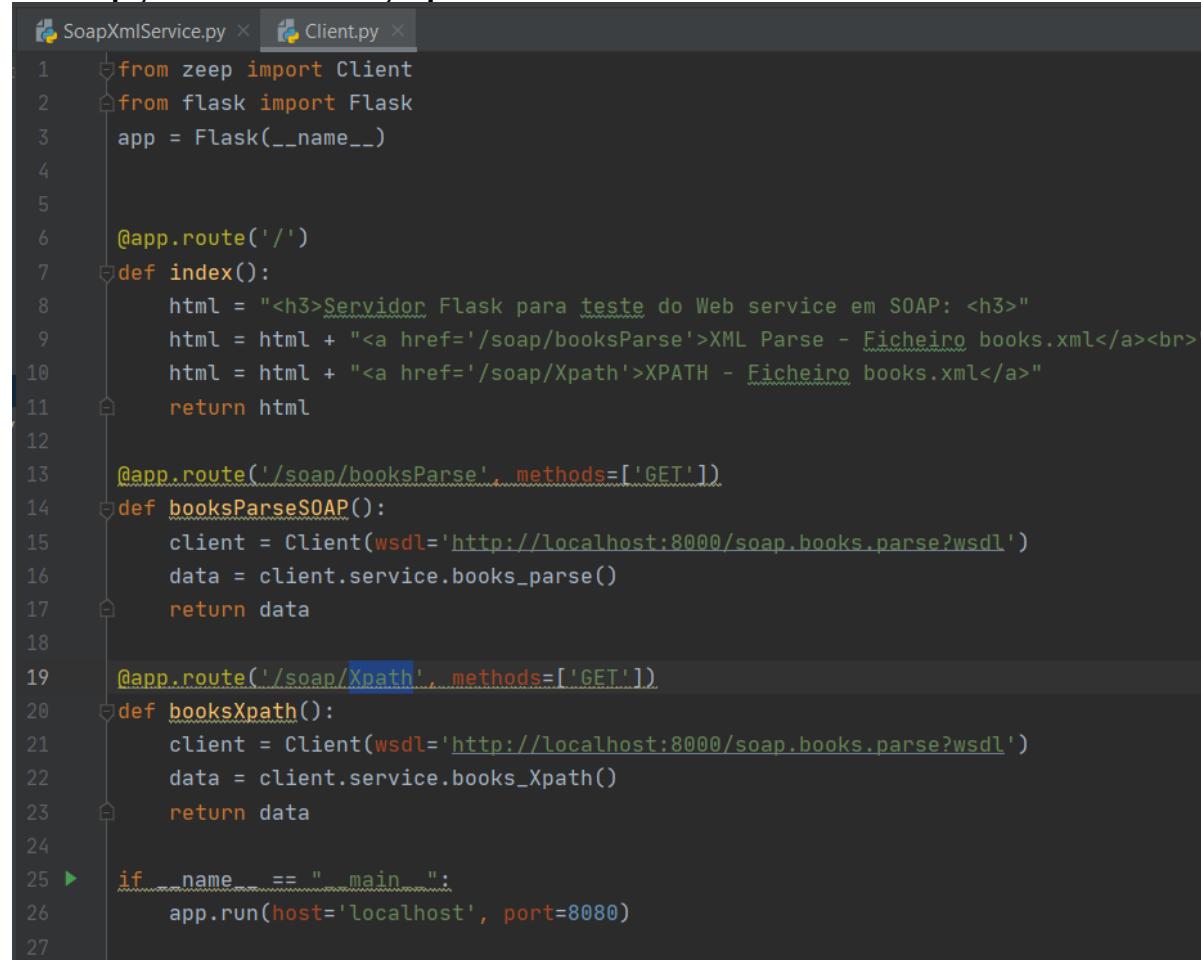
- Adicionalmente o serviço web é registado na aplicação, referindo-se a tns, o in_protocol e o out_protocol.
- Função principal onde o servidor é criado e inicializado em localhost na porta 8000, neste caso.

```
60 ► if __name__ == '__main__':
61     from wsgiref.simple_server import make_server
62
63     wsgi_app = WsgiApplication(application)
64     server = make_server('localhost', 8000, wsgi_app)
65     server.serve_forever()
66
```

```
54
55 application = Application([SoapXmlService],
56                             tns='soap.books.parse',
57                             in_protocol=Soap11(validation='lxml'),
58                             out_protocol=Soap11())
59
60 if __name__ == '__main__':
61     from wsgiref.simple_server import make_server
62
63     wsgi_app = WsgiApplication(application)
64     server = make_server('localhost', 8000, wsgi_app)
65     server.serve_forever()
66
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

Para o cliente, foi criado um servidor flask com métodos que podem ser chamados a partir do navegador e criar a partir da biblioteca zeep, um cliente, que chama o webservice criado e o respectivo método.



```
SoapXmlService.py x Client.py x
1  from zeep import Client
2  from flask import Flask
3  app = Flask(__name__)
4
5
6  @app.route('/')
7  def index():
8      html = "<h3>Servidor Flask para teste do Web service em SOAP: </h3>"
9      html = html + "<a href='/soap/booksParse'>XML Parse - Ficheiro books.xml</a><br>"
10     html = html + "<a href='/soap/Xpath'>XPATH - Ficheiro books.xml</a>"
11     return html
12
13
14  @app.route('/soap/booksParse', methods=['GET'])
15  def booksParseSOAP():
16      client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
17      data = client.service.books_parse()
18      return data
19
20
21  @app.route('/soap/Xpath', methods=['GET'])
22  def booksXpath():
23      client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
24      data = client.service.books_Xpath()
25      return data
26
27
28 if __name__ == "__main__":
29     app.run(host='localhost', port=8080)
```

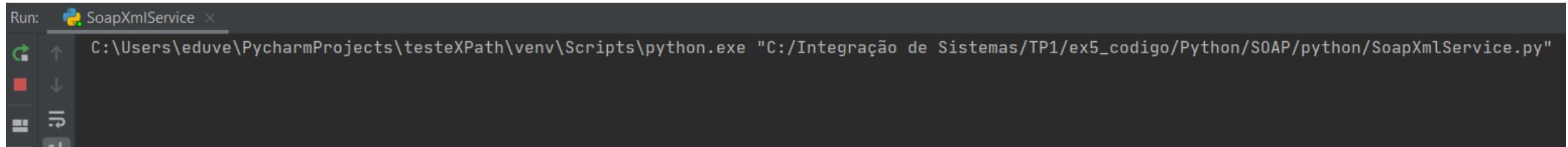
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

Para o cliente, foi criado um servidor flask com métodos que podem ser chamados a partir do navegador e criar a partir da biblioteca zeep, um cliente, que chama o webservice criado e o respectivo método.

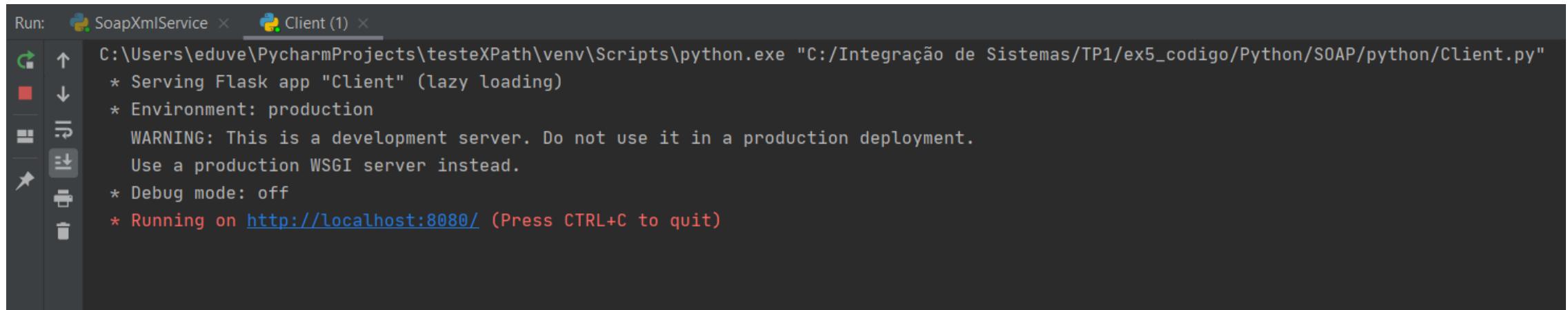
```
SoapXmlService.py x Client.py x
1  from zeep import Client
2  from flask import Flask
3  app = Flask(__name__)
4
5
6  @app.route('/')
7  def index():
8      html = "<h3>Servidor Flask para teste do Web service em SOAP: </h3>"
9      html = html + "<a href='/soap/booksParse'>XML Parse - Ficheiro books.xml</a><br>"
10     html = html + "<a href='/soap/Xpath'>XPATH - Ficheiro books.xml</a>"
11     return html
12
13 @app.route('/soap/booksParse', methods=['GET'])
14 def booksParseSOAP():
15     client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
16     data = client.service.books_parse()
17     return data
18
19 @app.route('/soap/Xpath', methods=['GET'])
20 def booksXpath():
21     client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
22     data = client.service.books_Xpath()
23     return data
24
25 if __name__ == "__main__":
26     app.run(host='localhost', port=8080)
27
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

Na primeira imagem seguinte é o serviço da web em execução, a segunda demonstra o cliente em execução e a terceira apresenta o endpoint raiz no servidor cliente, utilizado para apresentar os dados.



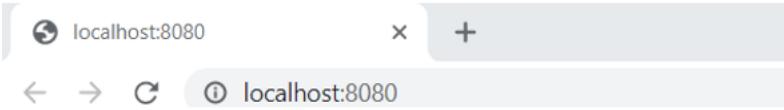
Run: SoapXmlService x
C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/TP1/ex5_codigo/Python/SOAP/python/SoapXmlService.py"



Run: SoapXmlService x Client (1) x
C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/TP1/ex5_codigo/Python/SOAP/python/Client.py"
* Serving Flask app "Client" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on <http://localhost:8080/> (Press CTRL+C to quit)



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**



Servidor Flask para teste do Web service em SOAP:

[XML Parse - Ficheiro books.xml](#)
[XPATH - Ficheiro books.xml](#)

Resultado da chamada do método books_parse no serviço web, com os dados apresentados no lado do cliente.

Resultado da chamada do método books_Xpath e o resultado apresentado no navegador:

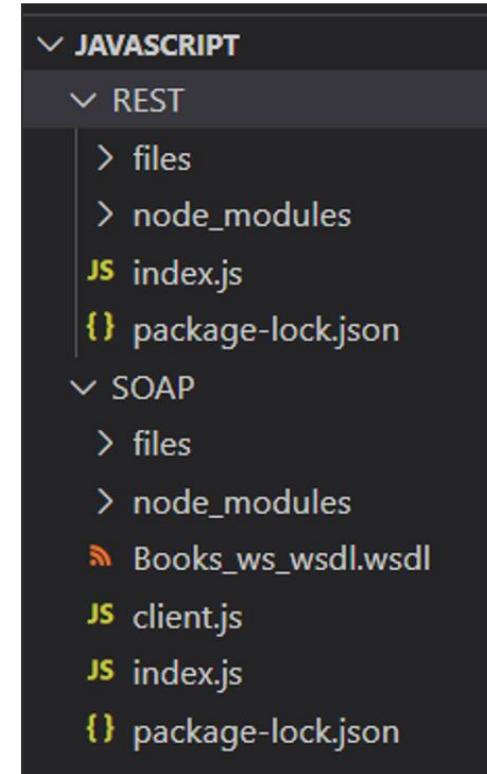
Chamadas registadas para o serviço da web.

```
C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/SoapXmlService.py"
127.0.0.1 - - [23/Nov/2020 16:01:59] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [23/Nov/2020 16:02:01] "POST /soap.books.parse HTTP/1.1" 200 4747
127.0.0.1 - - [23/Nov/2020 16:02:09] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [23/Nov/2020 16:02:11] "POST /soap.books.parse HTTP/1.1" 200 4747
127.0.0.1 - - [23/Nov/2020 16:05:28] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [23/Nov/2020 16:05:30] "POST /soap.books.parse HTTP/1.1" 200 2340
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:**





3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:

Imports necessários, primeiro ponto de extremidade registrado para executar a análise do arquivo books.xml, construindo a string para enviar os dados de volta ao cliente.

```
JS index.js  X
REST > JS index.js > app.get('/booksXpath') callback
1  var express = require('express');
2  var app = express();
3  var fs = require("fs");
4  var parser = require('xml2json-light');
5  var xpath = require('xpath')
6  var dom = require('xmldom').DOMParser
7  const http = require('http');
8  const url = require('url');

9
10 app.get('/booksParse', function (req, res) {
11   const queryObject = url.parse(req.url, true).query;
12   if (queryObject["file"] == "books") {
13     fs.readFile("files/books.xml", 'utf8', function (err, data) {
14       var json = parser.xml2json(data);
15       var i = 0;
16       var dados = "";
17       for (i = 0; i < json.catalog.book.length; i++) {
18         var book = json.catalog.book[i]
19         dados = dados + `Id: ${book.id}\n`;
20         dados = dados + `Autor: ${book.author}\n`;
21         dados = dados + `Título: ${book.title}\n`;
22         dados = dados + `Genero: ${book.genre}\n`;
23         dados = dados + `Preço: ${book.price}\n`;
24         dados = dados + `Data de Publicação: ${book.publish_date}\n`;
25         dados = dados + `Descrição: ${book.description}\n\n`;
26       }
27
28       res.setHeader("Content-Type", "text/plain; charset=utf-8");
29       res.end(dados);
30     });
31   } else {
32     res.setHeader("Content-Type", "text/plain; charset=utf-8");
33     res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
34   }
35 }
36
37 })
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:

Imports necessários
de extremidade
executar a a
books.xml, cons
enviar os dados

```
1 var express = require('express');
2 var app = express();
3 var fs = require("fs");
4 var parser = require('xml2json-light');
5 var xpath = require('xpath')
6 var dom = require('xmldom').DOMParser
7 const http = require('http');
8 const url = require('url');

9
10 app.get('/booksParse', function (req, res) {
11   const queryObject = url.parse(req.url, true).query;
12   if (queryObject["file"] == "books") {
13     fs.readFile("files/books.xml", 'utf8', function (err, data) {
14       var json = parser.xml2json(data);
15       var i = 0;
16       var dados = "";
17       for (i = 0; i < json.catalog.book.length; i++) {
18         var book = json.catalog.book[i]
19         dados = dados + `Id: ${book.id}\n`
20         dados = dados + `Autor: ${book.author}\n`
21         dados = dados + `Título: ${book.title}\n`
22         dados = dados + `Genero: ${book.genre}\n`
23         dados = dados + `Preço: ${book.price}\n`
24         dados = dados + `Data de Publicação: ${book.publish_date}\n`
25         dados = dados + `Descrição: ${book.description}\n\n`
26       }
27
28       res.setHeader("Content-Type", "text/plain; charset=utf-8");
29       res.end(dados);
30     });
31   } else {
32     res.setHeader("Content-Type", "text/plain; charset=utf-8");
33     res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
34   }
35 }
36 }
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:

Segundo ponto de extremidade, que usa a expressão XPath para obter todos os livros, onde o preço é superior a 30,50 e o gênero do livro é Computador. Também constrói uma string contendo os dados em formato html e os retorna ao cliente.

```
38
39 app.get('/booksxpath', function (req, res) {
40   const queryObject = url.parse(req.url, true).query;
41   if (queryObject["file"] == "books") {
42     fs.readFile("files/books.xml", 'utf8', function (err, data) {
43       var doc = new dom().parseFromString(data)
44       var nodes = xpath.select("/catalog/book[price>30.50 and genre='Computer']", doc)
45       var ids = xpath.select("/catalog/book/@id", doc)
46       var i = 0;
47       var dados = "<h2>RESULTADOS FILTRAGEM XPATH POR:</h2><br><p>price > 30.50 e genre = Computer</p><br>";
48
49       for (i = 0; i < nodes.length; i++) {
50         var bookId = ids[i].value
51         var book = nodes[i];
52         var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;
53         var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;
54         var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;
55         var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;
56         var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;
57         var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;
58         dados = dados + `<p>Id: ${bookId}</p>`;
59         dados = dados + `<p>Autor: ${autor}</p>`;
60         dados = dados + `<p>Título: ${titulo}</p>`;
61         dados = dados + `<p>Genero: ${genero}</p>`;
62         dados = dados + `<p>Preço: ${preco}</p>`;
63         dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`;
64         dados = dados + `<p>Descrição: ${descricao}</p><br>`;
65       }
66       res.setHeader("Content-Type", "text/html; charset=utf-8");
67       res.end(dados);
68     });
69   } else {
70     res.setHeader("Content-Type", "text/plain; charset=utf-8");
71     res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
72   }
73 }
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:

Segundo ponto de extremidade REST. Neste caso é feita uma requisição para o endpoint '/booksXpath'. No corpo da requisição é passada uma expressão XPath para obter os livros que têm o preço superior a 30,50 e o género é Computador. Também conseguimos obter os dados de cada livro, contendo os dados em formato JSON e enviando os resultados para o cliente.

```
39  app.get('/booksXpath', function (req, res) {
40    const queryObject = url.parse(req.url, true).query;
41    if (queryObject["file"] == "books") {
42      fs.readFile("files/books.xml", 'utf8', function (err, data) {
43        var doc = new dom().parseFromString(data)
44        var nodes = xpath.select("/catalog/book[price>30.50 and genre='Computer']", doc)
45        var ids = xpath.select("/catalog/book/@id", doc)
46        var i = 0;
47        var dados = "<h2>RESULTADOS FILTRAGEM XPATH POR:</h2><br><p>price > 30.50 e genre = Computer</p><br>";
48
49        for (i = 0; i < nodes.length; i++) {
50          var bookId = ids[i].value
51          var book = nodes[0];
52          var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;
53          var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;
54          var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;
55          var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;
56          var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;
57          var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;
58          dados = dados + `<p>Id: ${bookId}</p>`;
59          dados = dados + `<p>Autor: ${autor}</p>`;
60          dados = dados + `<p>Título: ${titulo}</p>`;
61          dados = dados + `<p>Genero: ${genero}</p>`;
62          dados = dados + `<p>Preço: ${preco}</p>`;
63          dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`;
64          dados = dados + `<p>Descrição: ${descricao}</p><br>`;
65        }
66        res.setHeader("Content-Type", "text/html; charset=utf-8");
67        res.end(dados);
68      });
69    } else {
70      res.setHeader("Content-Type", "text/plain; charset=utf-8");
71      res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
72    }
73  }
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:**

A primeira imagem, apresenta o código de criação e inicialização do servidor, configurado para escutar na porta 3000 do localhost. O segundo é a execução do arquivo criado, através da linha de comando, e o serviço web rodando.

```
77  var server = app.listen(3000, function () {  
78    var host = server.address().address  
79    var port = server.address().port  
80    console.log("Books REST Web Service listening at http://localhost:3000")  
81  })
```

```
77  var server = app.listen(3000, function () {  
78    var host = server.address().address  
79    var port = server.address().port  
80    console.log("Books REST Web Service listening at http://localhost:3000")  
81  })
```

Linha de comandos - node index.js

```
C:\Integração de Sistemas\TP1\ex5_codigo\JavaScript\REST>node index.js  
Books REST Web Service listening at http://localhost:3000
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA REST USANDO XPATH:

Resultado obtido chamando o primeiro nó de extremidade para analisar o arquivo books.xml.

Resultado obtido chamando o segundo nó de extremidade para analisar e filtrar o arquivo books.xml com XPath.

localhost:3000/booksXpath?file=books

RESULTADOS FILTRAGEM XPATH POR:

price > 30.50 e genre = Computer

Id: bk101

Autor: Gambardella, Matthew

Título: XML Developer's Guide

Genero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

Id: bk102

Autor: Gambardella, Matthew

Título: XML Developer's Guide

Genero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

Network Tab (Chrome DevTools):

- Request URL: http://localhost:3000/booksXpath?file=book
- Request Method: GET
- Status Code: 200 OK
- Remote Address: [::1]:3000
- Referrer Policy: strict-origin-when-cross-origin
- Response Headers:
 - Connection: keep-alive
 - Content-Length: 1063
 - Content-Type: text/html; charset=utf-8
 - Date: Mon, 23 Nov 2020 15:38:33 GMT
 - Keep-Alive: timeout=5
 - X-Powered-By: Express
- Request Headers:
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
 - Accept-Encoding: gzip, deflate, br
 - Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
 - Connection: keep-alive
 - Host: localhost:3000
 - Sec-Ch-Ua: "Chromium";v="86", "\Not\A;Brand";v="99", "

3 requests | 3.3 kB transferred | 3.1



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

- Para criar um serviço web SOAP, foi necessário criar um arquivo wsdl que especifica o serviço web, definindo as operações que ele inclui, os argumentos e o tipo de chamada, que neste caso é sabão, e os nomes e porta onde o serviço pode ser chamado.
- O código para o serviço da web foi criado usando o servidor expresso e pacotes de sabão para Node Js, e um cliente SOAP para chamar o serviço da web criado.
- Primeira parte do arquivo Books_ws_wsdl, que define as solicitações e os elementos que elas incluem.



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

```
Books_ws_wsdl.wsdl ×

SOAP > Books_ws_wsdl.wsdl
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsdl:definitions targetNamespace="http://tempuri.org/"
3     xmlns:s="http://www.w3.org/2001/XMLSchema"
4     xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
5     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
6     xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
7     xmlns:tns="http://tempuri.org/"
8     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
9     xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
10    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
11    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
12    <!-- Definição dos pedidos e respostas para o método parse_books --&gt;
13    &lt;wsdl:types&gt;
14      &lt;s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/"&gt;
15        &lt;s:element name="BooksParseRequest"&gt;
16          &lt;s:complexType&gt;
17            &lt;s:sequence&gt;
18              &lt;s:element minOccurs="1" maxOccurs="unbounded" name="type" type="s:string"/&gt;
19            &lt;/s:sequence&gt;
20          &lt;/s:complexType&gt;
21        &lt;/s:element&gt;
22        &lt;s:element name="BooksParseResponse"&gt;
23          &lt;s:complexType&gt;
24            &lt;s:sequence&gt;
25              &lt;s:element minOccurs="1" maxOccurs="unbounded" name="result" type="s:string"/&gt;
26            &lt;/s:sequence&gt;
27          &lt;/s:complexType&gt;
28        &lt;/s:element&gt;
29      &lt;/s:schema&gt;
30    &lt;/wsdl:types&gt;</pre>
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://tempuri.org/" 
3      xmlns:s="http://www.w3.org/2001/XMLSchema" 
4      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" 
5      xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" 
6      xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" 
7      xmlns:tns="http://tempuri.org" 
8      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" 
9      xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" 
10     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" 
11     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
12     <!-- Definição dos pedidos e respostas para o método parse_books --&gt;
13     &lt;wsdl:types&gt;
14         &lt;s:schema elementFormDefault="qualified" targetNamespace="<a href="http://tempuri.org/">http://tempuri.org/">
15             <s:element name="BooksParseRequest">
16                 <s:complexType>
17                     <s:sequence>
18                         <s:element minOccurs="1" maxOccurs="unbounded" name="type" type="s:string"/>
19                     </s:sequence>
20                 </s:complexType>
21             </s:element>
22             <s:element name="BooksParseResponse">
23                 <s:complexType>
24                     <s:sequence>
25                         <s:element minOccurs="1" maxOccurs="unbounded" name="result" type="s:string"/>
26                     </s:sequence>
27                 </s:complexType>
28             </s:element>
29         </s:schema>
30     </wsdl:types>
31     <!-- Definição dos dados e do seu tipo entre o cliente e o servidor --&gt;
32     &lt;wsdl:message name="BooksParseSoapIn"&gt;
33         &lt;wsdl:part name="parameters" element="tns:BooksParseRequest"/&gt;
34     &lt;/wsdl:message&gt;
35     &lt;wsdl:message name="BooksParseSoapOut"&gt;
36         &lt;wsdl:part name="parameters" element="tns:BooksParseResponse"/&gt;
37     &lt;/wsdl:message&gt;</pre>
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

Esta segunda parte define o tipo de comunicação entre o servidor e o cliente, a operação no serviço web e a forma como as mensagens serão transmitidas.

```
Books_ws_wsdl.wsdl X
SOAP > Books_ws_wsdl.wsdl
31  <!-- Definição dos dados e do seu tipo entre o cliente e o servidor -->
32  <wsdl:message name="BooksParseSoapIn">
33  | <wsdl:part name="parameters" element="tns:BooksParseRequest"/>
34  </wsdl:message>
35  <wsdl:message name="BooksParseSoapOut">
36  | <wsdl:part name="parameters" element="tns:BooksParseResponse"/>
37  </wsdl:message>
38  <wsdl:portType name="BooksParseSoapPort">
39  | <!-- Definição da operação books_parse, definida no index.js -->
40  <wsdl:operation name="BooksParse">
41  | <wsdl:input message="tns:BooksParseSoapIn"/>
42  | <wsdl:output message="tns:BooksParseSoapOut"/>
43  </wsdl:operation>
44  </wsdl:portType>
45  <!-- Definição de como as mensagens serão transmitidas -->
46  <wsdl:binding name="BooksParseServiceSoapBinding" type="tns:BooksParseSoapPort">
47  | <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
48  <wsdl:operation name="BooksParse">
49  | <soap:operation soapAction="BooksParse" style="document"/>
50  | <wsdl:input>
51  | | <soap:body use="literal"/>
52  | </wsdl:input>
53  | <wsdl:output>
54  | | <soap:body use="literal"/>
55  | </wsdl:output>
56  </wsdl:operation>
57  </wsdl:binding>
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Esta segunda parte define o tipo de comunicação entre o serviço web e o cliente, a operação

serviço web e a forma como mensagens serão transmitidas

```
Books_ws_wsdl.wsdl X
31  <!-- Definição dos dados e do seu tipo entre o cliente e o servidor -->
32  <wsdl:message name="BooksParseSoapIn">
33      <wsdl:part name="parameters" element="tns:BooksParseRequest"/>
34  </wsdl:message>
35  <wsdl:message name="BooksParseSoapOut">
36      <wsdl:part name="parameters" element="tns:BooksParseResponse"/>
37  </wsdl:message>
38  <wsdl:portType name="BooksParseSoapPort">
39      <!-- Definição da operação books_parse, definida no index.js -->
40      <wsdl:operation name="BooksParse">
41          <wsdl:input message="tns:BooksParseSoapIn"/>
42          <wsdl:output message="tns:BooksParseSoapOut"/>
43      </wsdl:operation>
44  </wsdl:portType>
45  <!-- Definição de como as mensagens serão transmitidas -->
46  <wsdl:binding name="BooksParseServiceSoapBinding" type="tns:BooksParseSoapPort">
47      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
48      <wsdl:operation name="BooksParse">
49          <soap:operation soapAction="BooksParse" style="document"/>
50          <wsdl:input>
51              <soap:body use="literal"/>
52          </wsdl:input>
53          <wsdl:output>
54              <soap:body use="literal"/>
55          </wsdl:output>
56      </wsdl:operation>
57  </wsdl:binding>
58  <wsdl:binding name="BooksParseServiceSoap12Binding" type="tns:BooksParseSoapPort">
59      <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
60      <wsdl:operation name="BooksParse">
61          <soap12:operation soapAction="BooksParse" style="document"/>
62          <wsdl:input>
63              <soap12:body use="literal"/>
64          </wsdl:input>
65          <wsdl:output>
66              <soap12:body use="literal"/>
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

A parte final cria a definição do serviço ao qual o método será vinculado.

```
Books_ws_wsdl.wsdl ×
SOAP > Books_ws_wsdl.wsdl
58  <wsdl:binding name="BooksParseServiceSoap12Binding" type="tns:BooksParseSoapPort">
59    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
60    <wsdl:operation name="BooksParse">
61      <soap12:operation soapAction="BooksParse" style="document"/>
62      <wsdl:input>
63        <soap12:body use="literal"/>
64      </wsdl:input>
65      <wsdl:output>
66        <soap12:body use="literal"/>
67      </wsdl:output>
68    </wsdl:operation>
69  </wsdl:binding>
70  <!-- Definição do serviço -->
71  <wsdl:service name="BooksParseService">
72    <wsdl:port name="BooksParseServiceSoapPort" binding="tns:BooksParseServiceSoapBinding">
73      <soap:address location="http://localhost:8000/wsdl"/>
74    </wsdl:port>
75    <wsdl:port name="BooksParseServiceSoap12Port" binding="tns:BooksParseServiceSoap12Binding">
76      <soap12:address location="http://localhost:8000/wsdl"/>
77    </wsdl:port>
78  </wsdl:service>
79 </wsdl:definitions>
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

A parte final cria a definição do serviço ao qual o método será vinculado.

```
Books_ws_wsdl.wsdl ×  
58 <wsdl:binding name="BooksParseServiceSoap12Binding" type="tns:BooksParseSoapPort">  
59   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>  
60   <wsdl:operation name="BooksParse">  
61     <soap12:operation soapAction="BooksParse" style="document"/>  
62     <wsdl:input>  
63       <soap12:body use="literal"/>  
64     </wsdl:input>  
65     <wsdl:output>  
66       <soap12:body use="literal"/>  
67     </wsdl:output>  
68   </wsdl:operation>  
69 </wsdl:binding>  
70 <!-- Definição do serviço -->  
71 <wsdl:service name="BooksParseService">  
72   <wsdl:port name="BooksParseServiceSoapPort" binding="tns:BooksParseServiceSoapBinding">  
73     <soap:address location="http://localhost:8000/wsdl"/>  
74   </wsdl:port>  
75   <wsdl:port name="BooksParseServiceSoap12Port" binding="tns:BooksParseServiceSoap12Binding">  
76     <soap12:address location="http://localhost:8000/wsdl"/>  
77   </wsdl:port>  
78 </wsdl:service>  
79 </wsdl:definitions>
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Esta primeira parte do arquivo apresenta as importações iniciais necessárias para o serviço web, e o método books_parse, que dependendo do argumento fornecido na chamada, executará a análise ou filtro e analisará o arquivo books.xml.

```
JS index.js ●  
SOAP > JS index.js > ...  
1 var soap = require('soap');  
2 var express = require('express');  
3 var fs = require('fs');  
4 var parser = require('xml2json-light');  
5 var xpath = require('xpath')  
6 var dom = require('xmldom').DOMParser  
7  
8 function books_parse(args) {  
9   if(args.type == "parse") {  
10     var data = fs.readFileSync( "files/books.xml", 'utf8');  
11     var json = parser.xml2json(data);  
12     var dados = "<h2>PARSE DO FICHEIRO - books.xml</h2><br>";  
13     var i = 0;  
14     for(i = 0; i < json.catalog.book.length; i++) {  
15       var book = json.catalog.book[i]  
16       dados = dados + `<p>Id: ${book.id}</p>`  
17       dados = dados + `<p>Autor: ${book.author}</p>`  
18       dados = dados + `<p>Título: ${book.title}</p>`  
19       dados = dados + `<p>Genero: ${book.genre}</p>`  
20       dados = dados + `<p>Preço: ${book.price}</p>`  
21       dados = dados + `<p>Data de Publicação: ${book.publish_date}</p>`  
22       dados = dados + `<p>Descrição: ${book.description}</p><br>`  
23     }  
24     return {  
25       result: dados  
26     }  
27 }
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Esta primeira parte do arquivo apresenta as importações iniciais necessárias para o web, e o método books_parse(). O argumento fornecido na chamada à função fará a análise ou filtro e analisará o arq

```
JS index.js ●  
SOAP > JS index.js > ...  
  
1 var soap = require('soap');  
2 var express = require('express');  
3 var fs = require('fs');  
4 var parser = require('xml2json-light');  
5 var xpath = require('xpath')  
6 var dom = require('xmldom').DOMParser  
7  
8 function books_parse(args) {  
9     if(args.type == "parse") {  
10        var data = fs.readFileSync( "files/books.xml", 'utf8' );  
11        var json = parser.xml2json(data);  
12        var dados = "<h2>PARSE DO FICHEIRO - books.xml</h2><br>";  
13        var i = 0;  
14        for(i = 0; i < json.catalog.book.length; i++) {  
15            var book = json.catalog.book[i]  
16            dados = dados + `<p>Id: ${book.id}</p>`  
17            dados = dados + `<p>Autor: ${book.author}</p>`  
18            dados = dados + `<p>Título: ${book.title}</p>`  
19            dados = dados + `<p>Genero: ${book.genre}</p>`  
20            dados = dados + `<p>Preço: ${book.price}</p>`  
21            dados = dados + `<p>Data de Publicação: ${book.publish_date}</p>`  
22            dados = dados + `<p>Descrição: ${book.description}</p><br>`  
23        }  
24        return {  
25            result: dados  
26        }  
27    }  
}
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Esta primeira parte do arquivo apresenta as importações iniciais necessárias para o serviço web, e o método books_parse, que dependendo do argumento fornecido na chamada, executará a análise ou filtro e analisará o arquivo books.xml.

```
29 | if(args.type == "xpath") {  
30 |     var data = fs.readFileSync( "files/books.xml", 'utf8');  
31 |     var doc = new dom().parseFromString(data)  
32 |     var nodes = xpath.select("/catalog/book[genre='Fantasy' and contains(author, 'Eva')]", doc)  
33 |     var ids = xpath.select("/catalog/book/@id", doc)  
34 |     var i = 0;  
35 |     var dados = "<h2>XPATH - books.xml</h2><p>Filtragem por livros do género <i>Fantasy</i>,\\  
36 |     e onde o nome do autor contém a string, <i>Eva</i></p><br>";  
37 |     for(i = 0; i < nodes.length; i++) {  
38 |         var bookId = ids[i].value  
39 |         var book = nodes[0];  
40 |         var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;  
41 |         var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;  
42 |         var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;  
43 |         var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;  
44 |         var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;  
45 |         var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;  
46 |         dados = dados + `<p>Id: ${bookId}</p>`  
47 |         dados = dados + `<p>Autor: ${autor}</p>`  
48 |         dados = dados + `<p>Título: ${titulo}</p>`  
49 |         dados = dados + `<p>Genero: ${genero}</p>`  
50 |         dados = dados + `<p>Preço: ${preco}</p>`  
51 |         dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`  
52 |         dados = dados + `<p>Descrição: ${descricao}</p><br>`  
53 |     }  
54 |     return {  
55 |         result: dados  
56 |     }  
57 | }  
58 | }  
59 | }
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Esta primeira parte do arquivo apresenta as importações iniciais necessárias para o serviço web, e o método books_parse, que dependendo do argumento fornecido na chamada, executará a análise ou filtro e analisará o arquivo books.xml.

```
28 if(args.type == "xpath") {  
29     var data = fs.readFileSync( "files/books.xml", 'utf8');  
30     var doc = new dom().parseFromString(data)  
31     var nodes = xpath.select("/catalog/book[genre='Fantasy' and contains(author, 'Eva')]", doc)  
32     var ids = xpath.select("/catalog/book/@id", doc)  
33     var i = 0;  
34     var dados = "<h2>XPATH - books.xml</h2><p>Filtragem por livros do género <i>Fantasy</i>,\\  
e onde o nome do autor contém a string, <i>Eva</i></p><br>";  
35     for(i = 0; i < nodes.length; i++) {  
36         var bookId = ids[i].value  
37         var book = nodes[0];  
38         var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;  
39         var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;  
40         var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;  
41         var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;  
42         var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;  
43         var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;  
44         dados = dados + `<p>Id: ${bookId}</p>`  
45         dados = dados + `<p>Autor: ${autor}</p>`  
46         dados = dados + `<p>Título: ${titulo}</p>`  
47         dados = dados + `<p>Genero: ${genero}</p>`  
48         dados = dados + `<p>Preço: ${preco}</p>`  
49         dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`  
50         dados = dados + `<p>Descrição: ${descricao}</p><br>`  
51     }  
52     return {  
53         result: dados  
54     }  
55 }  
56 }  
57 }
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Objeto de serviço criado e arquivo wsdl lido, configurando o método book_parse para ser executado quando o servidor for acessado na porta 8000 no localhost.

```
JS index.js ●
SOAP > JS index.js > ...
59
60  var serviceObject = {
61    BooksParseService: {
62      BooksParseServiceSoapPort: {
63        BooksParse: books_parse
64      },
65      BooksParseServiceSoap12Port: {
66        BooksParse: books_parse
67      }
68    };
69
70
71  var xml = fs.readFileSync('Books_ws_wsdl.wsdl', 'utf8');
72  var app = express();
73
74  app.get('/', function (req, res) {
75    res.send('SOAP webservice - Parse de ficheiros XML e XPATH<br />');
76  })
77
78  var port = 8000;
79  app.listen(port, function () {
80    console.log('Listening on port ' + port);
81    var wsdl_path = "/wsdl";
82    soap.listen(app, wsdl_path, serviceObject, xml);
83    console.log("Check http://localhost:" + port + wsdl\_path + "?wsdl to see if the service is working");
84  });
}
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

Objeto de serviço criado e arquivo wsdl lido, configurando o método book_parse para ser executado quando o servidor for acessado na porta 8000 no localhost.

```
JS index.js ●
SOAP > JS index.js > ...
59 var serviceObject = {
60   BooksParseService: {
61     BooksParseServiceSoapPort: {
62       BooksParse: books_parse
63     },
64     BooksParseServiceSoap12Port: {
65       BooksParse: books_parse
66     }
67   };
68 };
69
70 var xml = fs.readFileSync('Books_ws_wsdl.wsdl', 'utf8');
71 var app = express();
72
73 app.get('/', function (req, res) {
74   res.send('SOAP webservice - Parse de ficheiros XML e XPATH<br />');
75 })
76
77 var port = 8000;
78 app.listen(port, function () {
79   console.log('Listening on port ' + port);
80   var wsdl_path = "/wsdl";
81   soap.listen(app, wsdl_path, serviceObject, xml);
82   console.log("Check http://localhost:" + port + wsdl\_path + "?wsdl to see if the service is working");
83 })
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Código do cliente, usando as bibliotecas soap e express para chamar o serviço da web e apresentar os resultados, o primeiro ponto de extremidade para a análise do arquivo books.xml.

```
JS client.js  ×  
SOAP > JS client.js > app.get('/callBooksParse') callback > soap.createClient() callback  
1 var express = require('express');  
2 var app = express();  
3 var soap = require('soap');  
4 var url = 'http://localhost:8000/wsdl?wsdl';  
5  
6 app.get('/callBooksParse', function (req, res) {  
7     soap.createClient(url, function (err, client) {  
8         if (err){  
9             throw err;  
10        }  
11        var args = {  
12            type: 'parse'  
13        }  
14        client.BooksParse(args, function (err, result) {  
15            if (err) {  
16                throw err;  
17            }  
18            else {  
19                res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});  
20                res.end(result["result"][0]);  
21            }  
22        });  
23    });  
24})  
25
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Código do cliente, usando as bibliotecas soap e express para chamar o serviço da web e apresentar os resultados, o primeiro ponto de extremidade para a análise do arquivo books.xml.

```
JS client.js  ×  
SOAP > JS client.js > app.get('/callBooksParse') callback > soap.createClient() callback  
1  var express = require('express');  
2  var app = express();  
3  var soap = require('soap');  
4  var url = 'http://localhost:8000/wsdl?wsdl';  
5  
6  app.get('/callBooksParse', function (req, res) {  
7    soap.createClient(url, function (err, client) {  
8      if (err){  
9        throw err;  
10     }  
11     var args = {  
12       type: 'parse'  
13     }  
14     client.BooksParse(args, function (err, result) {  
15       if (err) {  
16         throw err;  
17       }  
18       else {  
19         res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});  
20         res.end(result["result"][0]);  
21       }  
22     });  
23   });  
24 }  
25 }
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Segundo endpoint onde o serviço web é chamado especificando o tipo de ação como XPath, apresentando os resultados como html.

```
js client.js  x
SOAP > JS client.js > app.get('/callBooksParse') callback > soap.createClient() callback
25
26 app.get('/callBooksXpath', function (req, res) {
27   soap.createClient(url, function (err, client) {
28     if (err){
29       throw err;
30     }
31     var args = {
32       type: 'xpath'
33     }
34     client.BooksParse(args, function (err, result) {
35       if (err) {
36         throw err;
37       }
38       else {
39         res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
40         res.end(result["result"][0]);
41       }
42     });
43   });
44 }
45
46 var server = app.listen(3000, function () {
47   var host = server.address().address
48   var port = server.address().port
49   console.log("Example app listening at http://localhost:" + port)
50 })
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Segundo endpoint onde o serviço web é chamado especificando o tipo de ação como XPath, apresentando os resultados como html.

```
26 app.get('/callBooksXpath', function (req, res) {
27   soap.createClient(url, function (err, client) {
28     if (err) {
29       throw err;
30     }
31     var args = {
32       type: 'xpath'
33     }
34     client.BooksParse(args, function (err, result) {
35       if (err) {
36         throw err;
37       }
38       else {
39         res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
40         res.end(result["result"][0]);
41       }
42     });
43   });
44 })
45
46 var server = app.listen(3000, function () {
47   var host = server.address().address
48   var port = server.address().port
49   console.log("Example app listening at http://localhost:3000")
50 })
```

3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

Executando o servidor (imagem na parte superior), rodando o cliente (imagem na parte inferior).

Linha de comandos - node index.js

```
C:\Integração de Sistemas\TP1\ex5_codigo\JavaScript\SOAP>node index.js
Listening on port 8000
Check http://localhost:8000/wsdl?wsdl to see if the service is working
```

Linha de comandos - node client.js

```
C:\Integração de Sistemas\TP1\ex5_codigo\JavaScript\SOAP>node client.js
Example app listening at http://localhost:3000
```



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

Resultados obtidos chamando o serviço da web para analisar o arquivo books.xml.

PARSE DO FICHEIRO - books.xml

Id: bk101
Autor: Gambardella, Matthew
Título: XML Developer's Guide
Género: Computer
Preço: 44.95
Data de Publicação: 2000-10-01
Descrição: An in-depth look at creating applications with XML.

Id: bk102
Autor: Ralls, Kim
Título: Midnight Rain
Género: Fantasy
Preço: 5.95
Data de Publicação: 2000-12-16
Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Id: bk103
Autor: Corets, Eva

Network Tab (Developer Tools):

- Request URL:** http://localhost:3000/callBooksParse
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** [::1]:3000
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
 - Connection: keep-alive
 - Content-Type: text/html; charset=utf-8
 - Date: Mon, 23 Nov 2020 15:48:36 GMT
 - Keep-Alive: timeout=5
 - Transfer-Encoding: chunked
 - X-Powered-By: Express
- Request Headers:**
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
 - Accept-Encoding: gzip, deflate, br
 - Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
 - Connection: keep-alive
 - Host: localhost:3000
 - sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google Ch



3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em

duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

Resultados obtidos chamando o webservice para analisar e filtrar o arquivo books.xml com Xpath.

The screenshot shows a browser window with the URL `localhost:3000/callBooksXpath`. The left side of the screen displays the XML response from the webservice, which includes two book entries. The right side shows the Network tab of the Chrome DevTools, providing detailed information about the request and response.

Left Side (Browser Content):

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <book id="bk101">
        <author>Corets, Eva</author>
        <title>Maeve Ascendant</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <published>2000-11-17</published>
        <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
    </book>
    <book id="bk102">
        <author>Corets, Eva</author>
        <title>Maeve Ascendant</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <published>2000-11-17</published>
        <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
    </book>
</books>
```

Right Side (DevTools Network Tab):

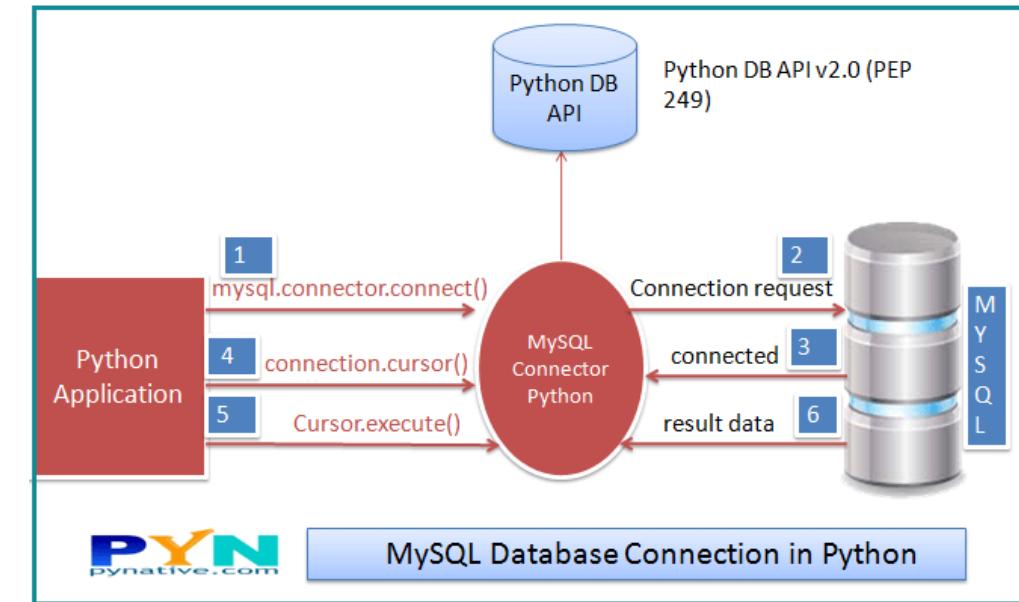
- Request URL:** `http://localhost:3000/callBooksXpath`
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** `[::1]:3000`
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
 - Connection: keep-alive
 - Content-Type: text/html; charset=utf-8
 - Date: Mon, 23 Nov 2020 15:49:12 GMT
 - Keep-Alive: timeout=5
 - Transfer-Encoding: chunked
 - X-Powered-By: Express
- Request Headers:**
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
 - Accept-Encoding: gzip, deflate, br
 - Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
 - Connection: keep-alive
 - Host: localhost:3000
 - sec-ch-ua: "Chromium";v="86", "\Not\\A;Brand";v="99", "Google Chrome"

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

1) Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação: Proceda a criação de um webservice em Python para leitura dos dados da tabela alunos (na base de dados mySQL implementada em mySQL) e devolução da informação no formato XML.

Algumas Orientações:

- Como conectar Python com mySQL:
<https://pynative.com/python-mysql-database-connection/>



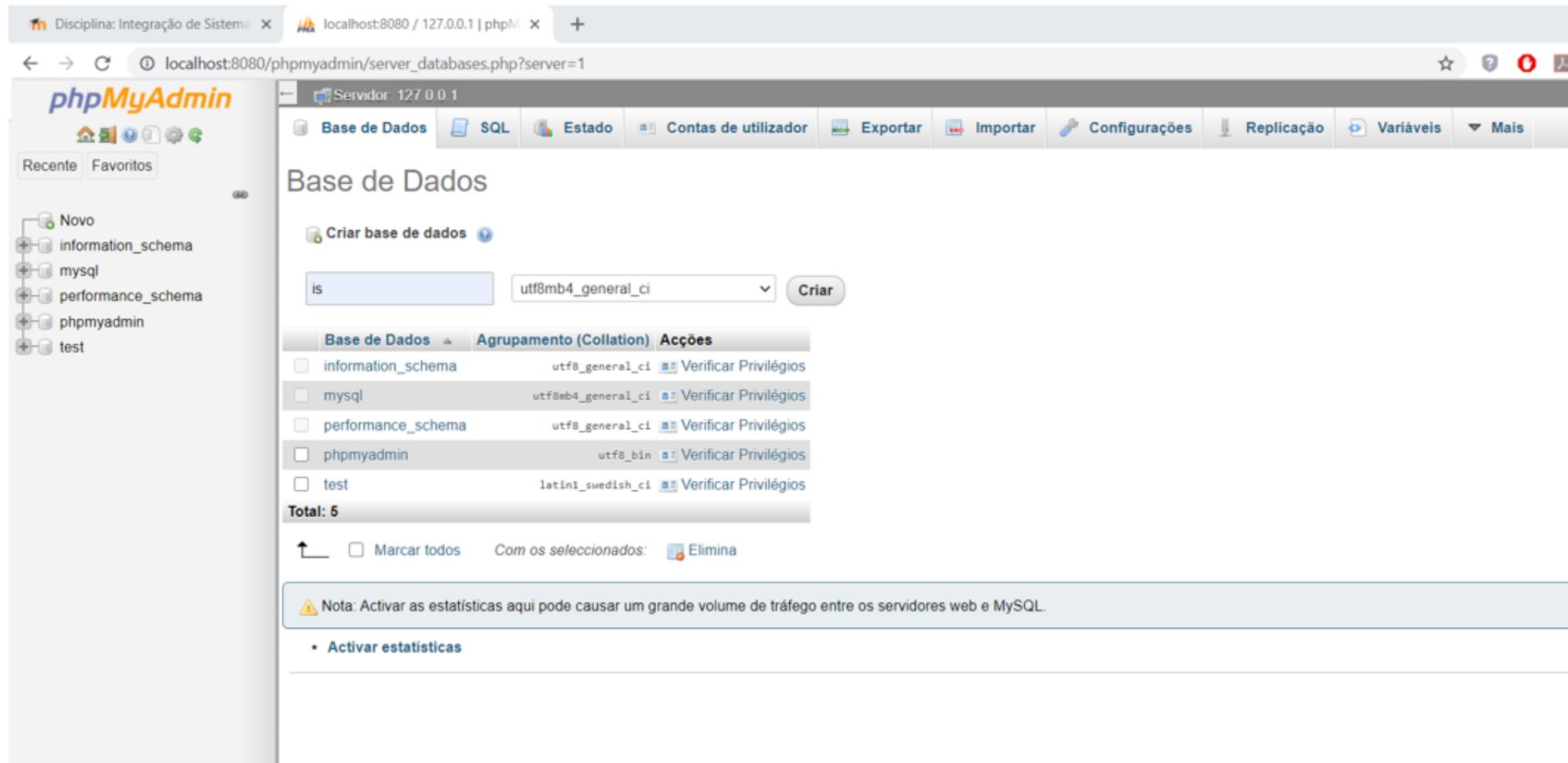


4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

- Para os web services REST e SOAP que envolvem o acesso a uma base de dados MySQL, foram utilizadas as mesmas bibliotecas, mas adicionalmente foi importado o conector MySQL para Python, permitindo a criação de uma conexão com o banco de dados rodando no localhost na porta 3306.

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Criação da tabela na base de dados, usando Mysql e o phpMyAdmin, criando a tabela alunos:



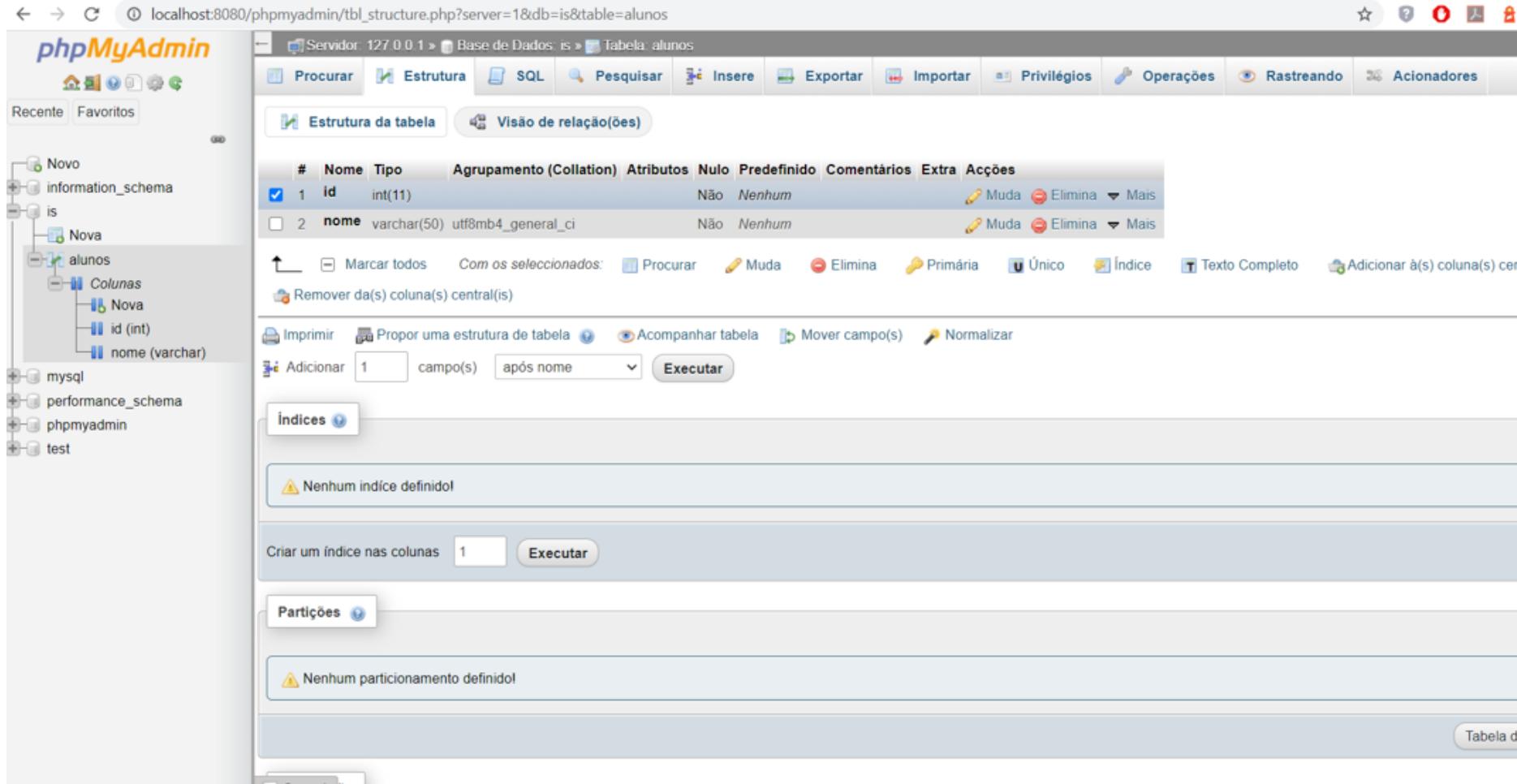
The screenshot shows the phpMyAdmin interface on a web browser. The title bar indicates the window is for 'Disciplina: Integração de Sistemas' and the address is 'localhost:8080 / 127.0.0.1 | phpMyAdmin'. The main menu bar includes 'Base de Dados', 'SQL', 'Estado', 'Contas de utilizador', 'Exportar', 'Importar', 'Configurações', 'Replicação', 'Variáveis', and 'Mais'. On the left, a sidebar lists existing databases: 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', and 'test'. A 'Novo' button is available for creating a new database. The central panel is titled 'Base de Dados' and shows a table of existing databases. A new database named 'is' is being created, with 'utf8mb4_general_ci' selected as the character set. The table lists the following databases:

Base de Dados	Agrupamento (Collation)	Acções
information_schema	utf8_general_ci	Verificar Privilégios
mysql	utf8mb4_general_ci	Verificar Privilégios
performance_schema	utf8_general_ci	Verificar Privilégios
phpmyadmin	utf8_bin	Verificar Privilégios
test	latin1_swedish_ci	Verificar Privilégios

Total: 5

At the bottom, there are buttons for 'Marcar todos', 'Com os selecionados:', and 'Eliminar'. A note at the bottom states: 'Nota: Activar as estatísticas aqui pode causar um grande volume de tráfego entre os servidores web e MySQL.' with a link to 'Activar estatísticas'.

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.



The screenshot shows the phpMyAdmin interface for the 'is' database. The left sidebar lists databases: information_schema, is, Nova, mysql, performance_schema, phpmyadmin, and test. The 'alunos' table is selected under the 'is' database. The main panel displays the table structure:

#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra	Acções
1	<input checked="" type="checkbox"/> id	int(11)			Não	Nenhum		 Muda  Elimina  Mais	
2	<input type="checkbox"/> nome	varchar(50)	utf8mb4_general_ci		Não	Nenhum		 Muda  Elimina  Mais	

Below the table structure, there are buttons for: Marcar todos, Com os seleccionados:, Procurar, Muda, Elimina, Primária, Único, Índice, Texto Completo, Adicionar à(s) coluna(s), Remover da(s) coluna(s) central(is), Imprimir, Propor uma estrutura de tabela, Acompanhar tabela, Mover campo(s), Normalizar, Adicionar, campo(s), após nome, Executar, Índices, Criar um índice nas colunas, Executar, Partições, and Nenhum particionamento definido!.



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

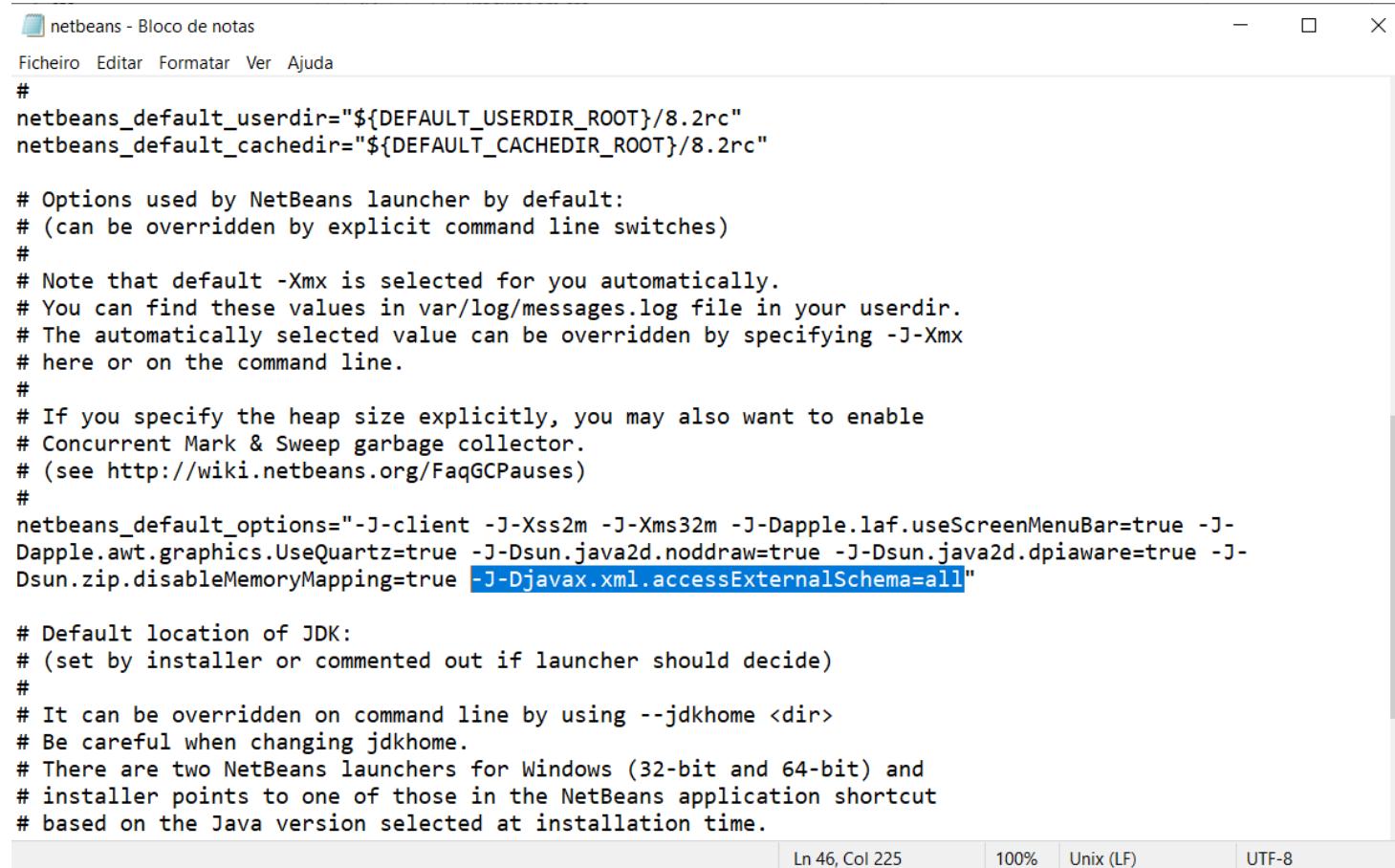
The screenshot shows the phpMyAdmin interface for a MySQL database named 'is'. The left sidebar shows the database structure with 'alunos' as the selected table. The main panel displays the 'Estrutura da tabela' (Table Structure) for the 'alunos' table. The table has two columns: 'id' (int(11)) and 'nome' (varchar(50)). The 'id' column is defined as the primary key (PRIMARY). Below the table structure, there is an 'Índices' (Indexes) section showing a single index named 'PRIMARY' using the BTREE algorithm. A warning message at the bottom states 'Nenhum particionamento definido!' (No partitioning defined!).

#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra	Acções
1	id	int(11)			Não	Nenhum			Muda Elimina Mais
2	nome	varchar(50)	utf8mb4_general_ci		Não	Nenhum			Muda Elimina Mais

Acções	Nome da chave	Tipo	Único	Pacote	Coluna	Quantidade	Agrupamento (Collation)	Nulo	Comentário
Edita Elimina	PRIMARY	BTREE	Sim	Não	id	0	A	Não	

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Opção adicionada à configuração padrão do NetBeans



The screenshot shows a Windows-style Notepad window titled "netbeans - Bloco de notas". The window contains configuration code for the NetBeans launcher. A specific line of code, `-J-Djax.xml.accessExternalSchema=all`, is highlighted with a blue selection bar. The code includes comments explaining various launcher options and their default values.

```
# netbeans_default_userdir="${DEFAULT_USERDIR_ROOT}/8.2rc"
# netbeans_default_cachedir="${DEFAULT_CACHEDIR_ROOT}/8.2rc"

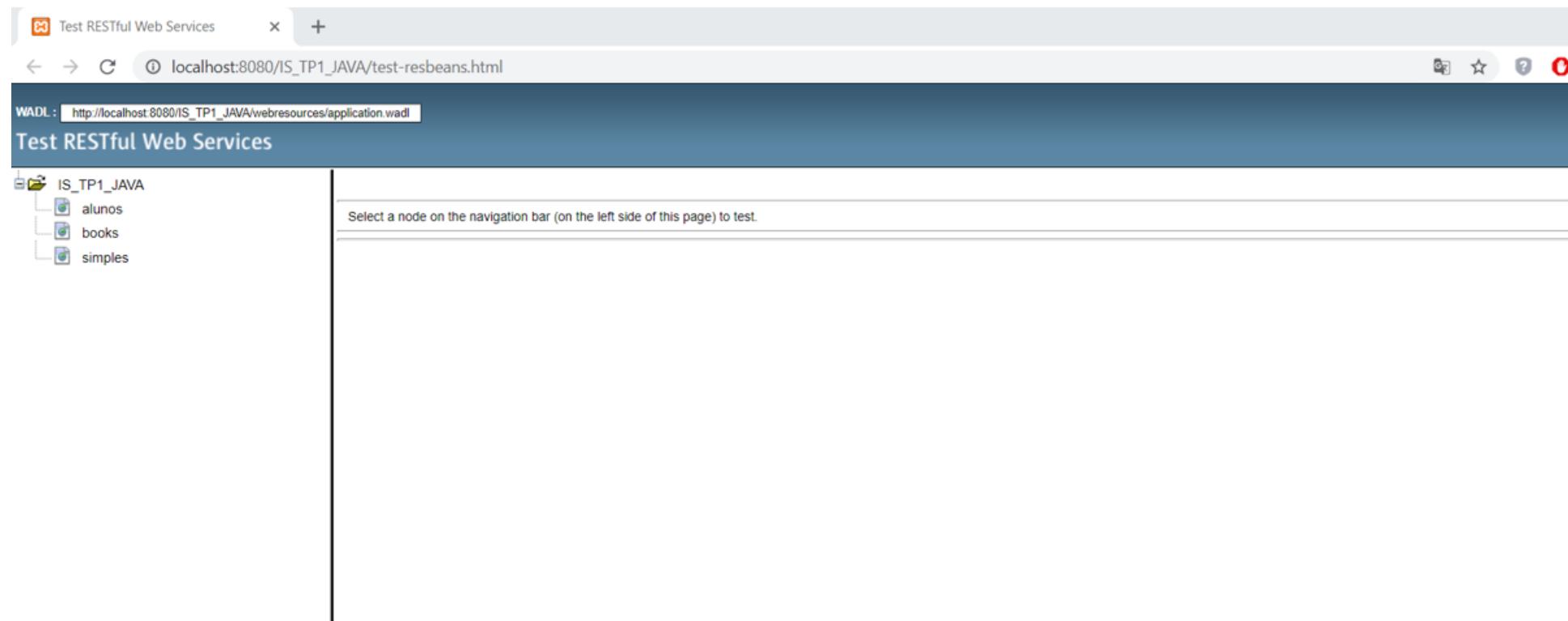
# Options used by NetBeans launcher by default:
# (can be overridden by explicit command line switches)
#
# Note that default -Xmx is selected for you automatically.
# You can find these values in var/log/messages.log file in your userdir.
# The automatically selected value can be overridden by specifying -J-Xmx
# here or on the command line.
#
# If you specify the heap size explicitly, you may also want to enable
# Concurrent Mark & Sweep garbage collector.
# (see http://wiki.netbeans.org/FaqGCPauses)
#
netbeans_default_options="-J-client -J-Xss2m -J-Xms32m -J-Dapple.laf.useScreenMenuBar=true -J-
Dapple.awt.graphics.UseQuartz=true -J-Dsun.java2d.noddraw=true -J-Dsun.java2d.dpiaware=true -J-
Dsun.zip.disableMemoryMapping=true -J-Djax.xml.accessExternalSchema=all

# Default location of JDK:
# (set by installer or commented out if launcher should decide)
#
# It can be overridden on command line by using --jdkhome <dir>
# Be careful when changing jdkhome.
# There are two NetBeans launchers for Windows (32-bit and 64-bit) and
# installer points to one of those in the NetBeans application shortcut
# based on the Java version selected at installation time.
```

Ln 46, Col 225 | 100% | Unix (LF) | UTF-8

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Serviços da web REST disponíveis



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Serviços Web em Java, via SOAP:

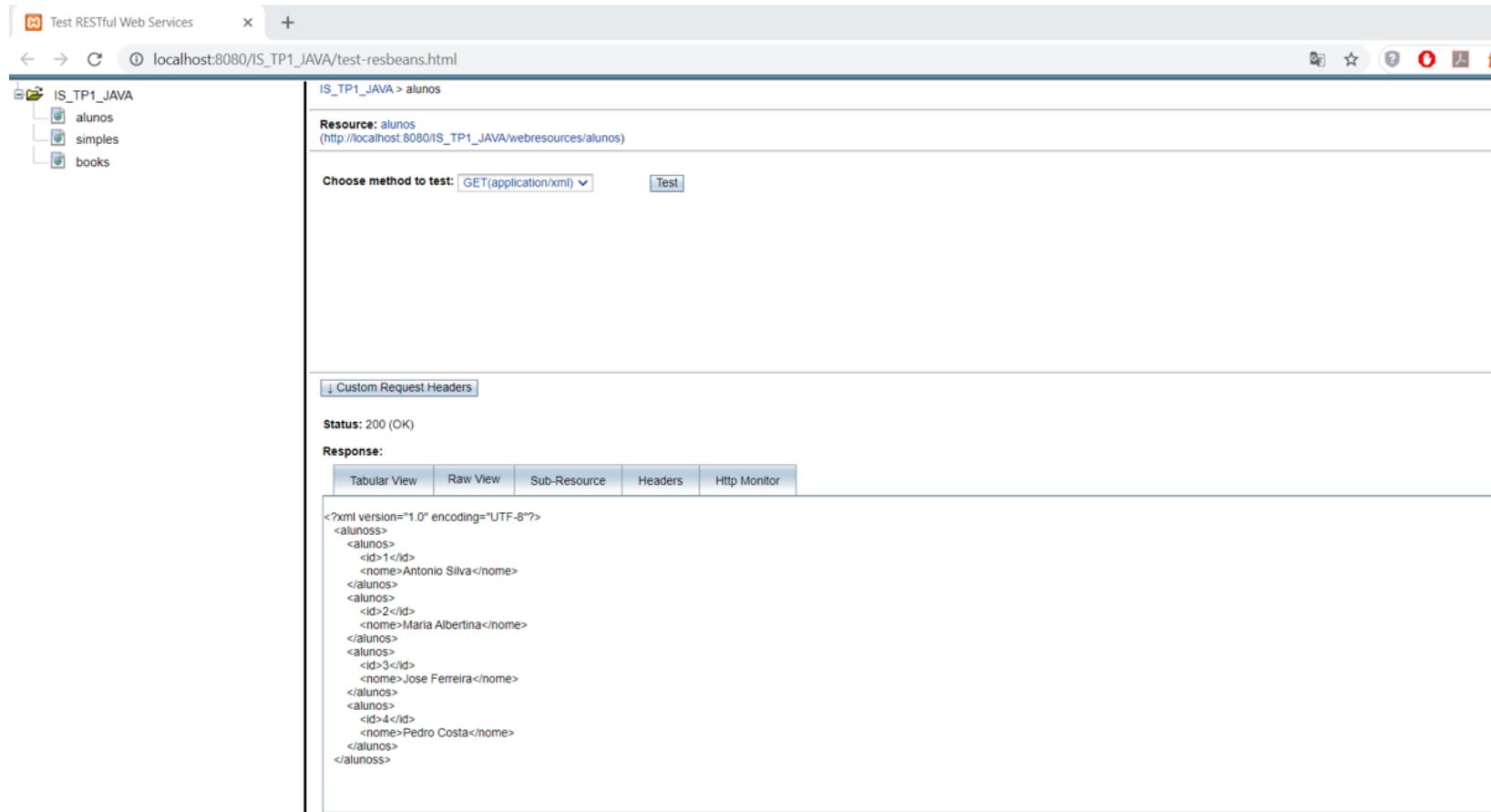


Endpoint	Information
Service Name: {http://is2.com/}Books_ws_soap Port Name: {http://is2.com/}Books_ws_soapPort	Address: http://localhost:8080/IS_TP1_JAVA/Books_ws_soap WSDL: http://localhost:8080/IS_TP1_JAVA/Books_ws_soap?wsdl Implementation class: com.is2.Books_ws_soap
Service Name: {http://is2.com/}Simples_ws_soap Port Name: {http://is2.com/}Simples_ws_soapPort	Address: http://localhost:8080/IS_TP1_JAVA/Simples_ws_soap WSDL: http://localhost:8080/IS_TP1_JAVA/Simples_ws_soap?wsdl Implementation class: com.is2.Simples_ws_soap
Service Name: {http://db.is.com/}Alunos_ws_soap Port Name: {http://db.is.com/}Alunos_ws_soapPort	Address: http://localhost:8080/IS_TP1_JAVA/Alunos_ws_soap WSDL: http://localhost:8080/IS_TP1_JAVA/Alunos_ws_soap?wsdl Implementation class: com.is.db.Alunos_ws_soap



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Serviço web REST em java para o ficheiro alunos.xml

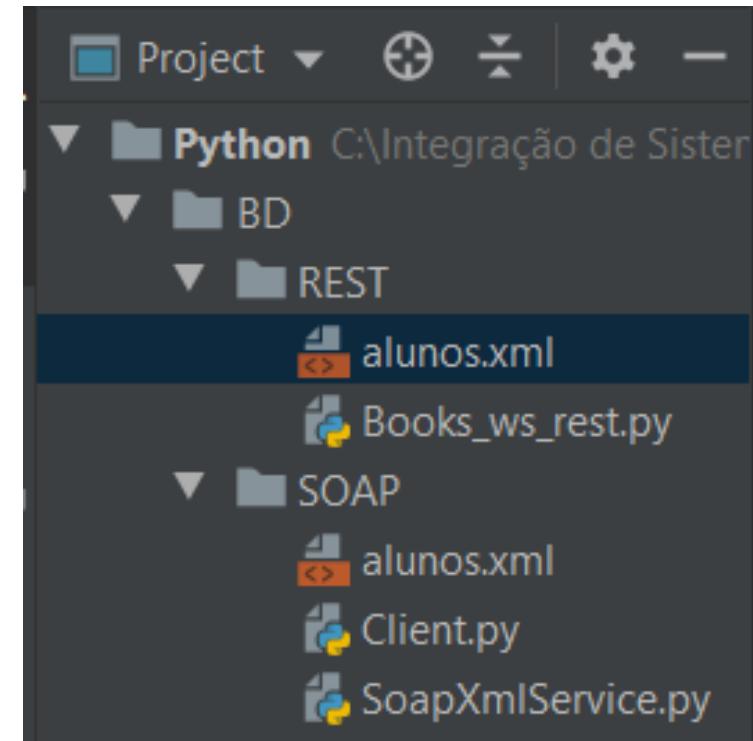


The screenshot shows a web-based interface for testing RESTful services. The URL is localhost:8080/IS_TP1_JAVA/test-resbeans.html. On the left, there's a tree view of the service structure under 'IS_TP1_JAVA': 'alunos' is expanded, showing 'simples' and 'books' as children. The main panel displays the 'alunos' resource details. A dropdown menu 'Choose method to test:' is set to 'GET(application/xml)'. Below it, a 'Test' button is visible. The status is 'Status: 200 (OK)'. Under the 'Response' tab, the XML content is shown:

```
<?xml version="1.0" encoding="UTF-8"?>
<alunos>
  <aluno>
    <id>1</id>
    <nome>Antonio Silva</nome>
  </aluno>
  <aluno>
    <id>2</id>
    <nome>Maria Albertina</nome>
  </aluno>
  <aluno>
    <id>3</id>
    <nome>Jose Ferreira</nome>
  </aluno>
  <aluno>
    <id>4</id>
    <nome>Pedro Costa</nome>
  </aluno>
</alunos>
```

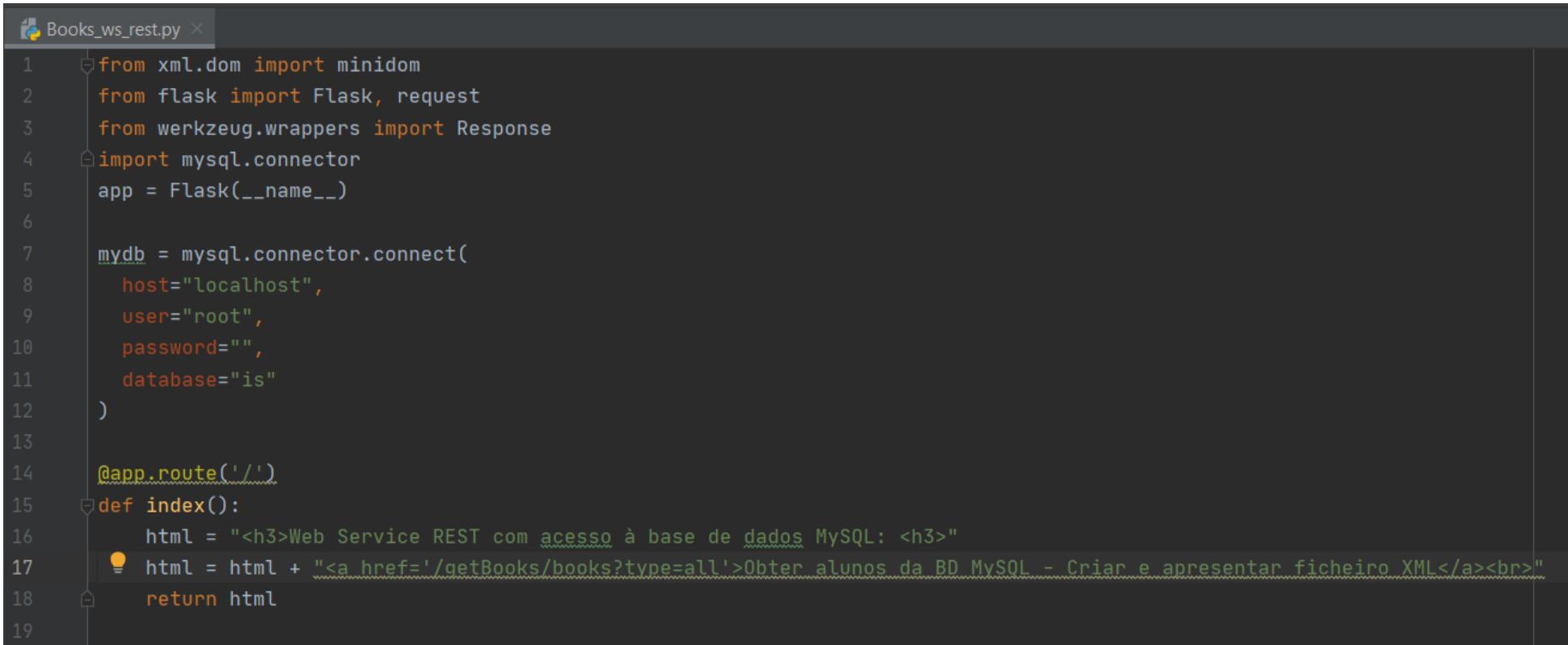
4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

A estrutura do projeto criada com os serviços da web SOAP e REST, em python:



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Importações necessárias utilizadas no serviço web, definição da conexão ao banco de dados MySQL e o primeiro endpoint do serviço web.



```
Books_ws_rest.py x
1  from xml.dom import minidom
2  from flask import Flask, request
3  from werkzeug.wrappers import Response
4  import mysql.connector
5  app = Flask(__name__)
6
7  mydb = mysql.connector.connect(
8      host="localhost",
9      user="root",
10     password="",
11     database="is"
12 )
13
14 @app.route('/')
15 def index():
16     html = "<h3>Web Service REST com acesso à base de dados MySQL: </h3>"
17     html = html + "<a href='/getBooks/books?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
18     return html
19
```



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Importações necessárias utilizadas no serviço web, definição da conexão ao banco de dados MySQL e o primeiro endpoint do serviço web.

```
Books_ws_rest.py x
1 from xml.dom import minidom
2 from flask import Flask, request
3 from werkzeug.wrappers import Response
4 import mysql.connector
5 app = Flask(__name__)
6
7 mydb = mysql.connector.connect(
8     host="localhost",
9     user="root",
10    password="",
11    database="is"
12 )
13
14 @app.route('/')
15 def index():
16     html = "<h3>Web Service REST com acesso à base de dados MySQL: <h3>"
17     html = html + "<a href='/getAlunos/alunos?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
18     return html
19
```



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Segundo endpoint com a execução de uma consulta SQL para obter todos os alunos da tabela alunos, bem como a criação dos tags para construção do arquivo XML, que é criado na mesma pasta onde está definido o books_ws_rest.py.

No final, os dados são devolvidos ao cliente e impressos como XML.

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

```
Books_ws_rest.py x
20 @app.route('/getAlunos/alunos', methods=['GET'])
21 def getAlunos():
22     selectType = request.args.get('type')
23     if selectType.upper() == "ALL":
24         mycursor = mydb.cursor()
25         mycursor.execute("SELECT * FROM alunos")
26         myresult = mycursor.fetchall()
27         doc = minidom.Document()
28         tag_alunos = doc.createElement('alunos')
29         for aluno in myresult:
30             id_aluno = str(aluno[0])
31             nome = aluno[1]
32             tag_aluno = doc.createElement('aluno')
33
34             tag_id = doc.createElement('id')
35             content_id = doc.createTextNode(id_aluno)
36             tag_id.appendChild(content_id)
37
38             tag_nome = doc.createElement('nome')
39             content_nome = doc.createTextNode(nome)
40             tag_nome.appendChild(content_nome)
41
42             tag_aluno.appendChild(tag_id)
43             tag_aluno.appendChild(tag_nome)
44
45             tag_alunos.appendChild(tag_aluno)
46
47             doc.appendChild(tag_alunos)
48             string_xml = doc.toprettyxml(indent="\t")
49             caminho_ficheiro = "alunos.xml"
50             with open(caminho_ficheiro, "w") as f:
51                 f.write(string_xml)
```

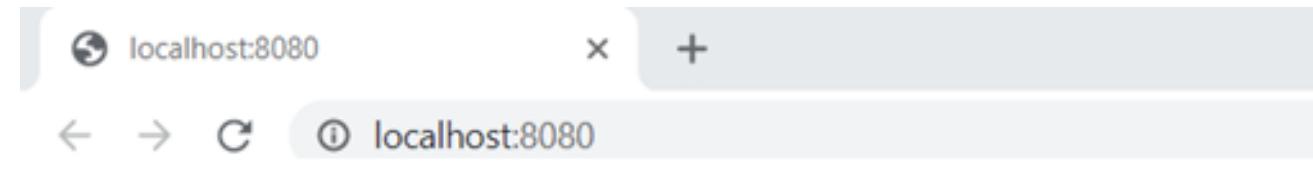
```
20 @app.route('/getAlunos/alunos', methods=['GET'])
21 def getBooks():
22     selectType = request.args.get('type')
23     if selectType.upper() == "ALL":
24         mycursor = mydb.cursor()
25         mycursor.execute("SELECT * FROM alunos")
26         myresult = mycursor.fetchall()
27         doc = minidom.Document()
28         tag_alunos = doc.createElement('alunos')
29         for aluno in myresult:
30             id_aluno = str(aluno[0])
31             nome = aluno[1]
32             tag_aluno = doc.createElement('aluno')
33
34             tag_id = doc.createElement('id')
35             content_id = doc.createTextNode(id_aluno)
36             tag_id.appendChild(content_id)
37
38             tag_nome = doc.createElement('nome')
39             content_nome = doc.createTextNode(nome)
40             tag_nome.appendChild(content_nome)
41
42             tag_aluno.appendChild(tag_id)
43             tag_aluno.appendChild(tag_nome)
44
45             tag_alunos.appendChild(tag_aluno)
46
47             doc.appendChild(tag_alunos)
48             string_xml = doc.toprettyxml(indent="\t")
49             caminho_ficheiro = "alunos.xml"
50             with open(caminho_ficheiro, "w") as f:
51                 f.write(string_xml)
52             return Response(string_xml, mimetype='text/xml')
```

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

A resposta retornada ao cliente e o link para chamar o serviço web e a imagem final a chamada registrada no serviço web quando é chamada são apresentados nas figuras seguintes:

```
52     return Response(string_xml, mimetype='text/xml')
53
54 ► if __name__ == "__main__":
55     app.run(host='localhost', port=8080)
56
```

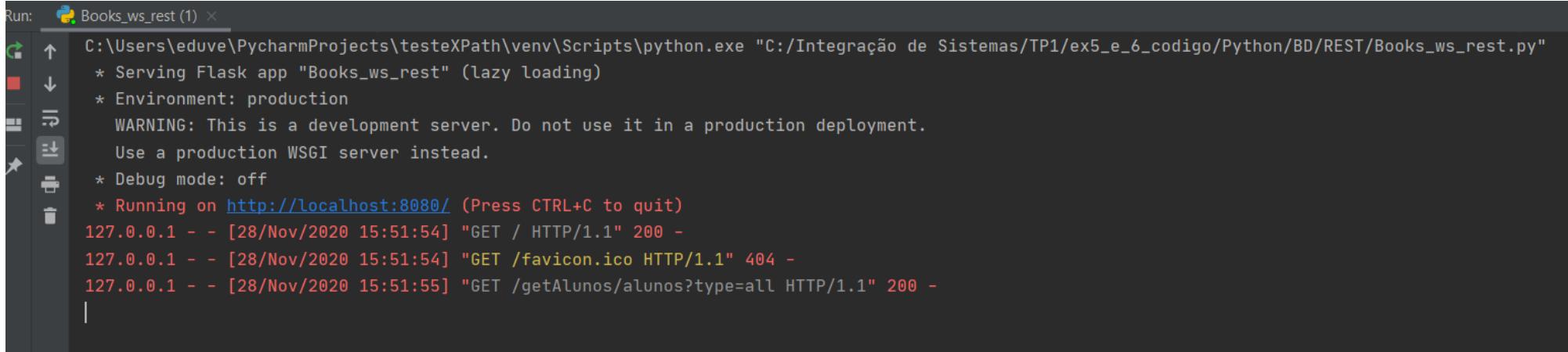
```
54 if __name__ == "__main__":
55     app.run(host='localhost', port=8080)
56
```



Web Service REST com acesso à base de dados MySQL:

Obter alunos da BD MySQL - Criar e apresentar ficheiro XML

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.



The screenshot shows the PyCharm IDE's Run console window. The title bar says "Run: Books_ws_rest (1) ×". The main area displays the following log output:

```
C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/TP1/ex5_e_6_codigo/Python/BD/REST/Books_ws_rest.py"
* Serving Flask app "Books_ws_rest" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://localhost:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [28/Nov/2020 15:51:54] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/Nov/2020 15:51:54] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [28/Nov/2020 15:51:55] "GET /getAlunos/alunos?type=all HTTP/1.1" 200 -
```



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Resultado obtido chamando o serviço web:

The screenshot shows a browser window with the URL `localhost:8080/getAlunos/alunos` and a sub-request `localhost:8080/getAlunos/alunos?type=all`. The Network tab of the developer tools is active, displaying a list of requests. One request is selected, showing the following details:

- Name:** `alunos?type=all`
- Request URL:** `http://localhost:8080/getAlunos/alunos?type=all`
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** 127.0.0.1:8080
- Referrer Policy:** strict-origin-when-cross-origin
- Response Headers:**
 - Content-Length: 286
 - Content-Type: text/xml; charset=utf-8
 - Date: Tue, 24 Nov 2020 11:15:56 GMT
 - Server: Werkzeug/1.0.1 Python/3.8.2
- Request Headers:**
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
 - Accept-Encoding: gzip, deflate, br
 - Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
 - Connection: keep-alive
 - Host: localhost:8080
 - Referer: http://localhost:8080/
 - sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google Chrome"
 - sec-ch-ua-mobile: ?0
 - Sec-Fetch-Dest: document

4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Para o serviço web SOAP o código é semelhante ao serviço web REST, sendo o uso de spyne, criando um servidor e registrando o serviço web com um tns, a única diferença em relação ao serviço web REST, como

demonstrado pelas seguintes imagens.

```
57 ► if __name__ == '__main__':
58     from wsgiref.simple_server import make_server
59
60     wsgi_app = WsgiApplication(application)
61     server = make_server('localhost', 8000, wsgi_app)
62     server.serve_forever()
63
```

```
57 ┌ if __name__ == '__main__':
58   └ from wsgiref.simple_server import make_server
59
60   wsgi_app = WsgiApplication(application)
61   server = make_server('localhost', 8000, wsgi_app)
62   server.serve_forever()
63
```



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

```
SoapXmlService.py x Client.py x
30
31     tag_id = doc.createElement('id')
32     content_id = doc.createTextNode(id_aluno)
33     tag_id.appendChild(content_id)
34
35     tag_nome = doc.createElement('nome')
36     content_nome = doc.createTextNode(nome)
37     tag_nome.appendChild(content_nome)
38
39     tag_aluno.appendChild(tag_id)
40     tag_aluno.appendChild(tag_nome)
41
42     tag_alunos.appendChild(tag_aluno)
43
44     doc.appendChild(tag_alunos)
45     string_xml = doc.toprettyxml(indent="\t")
46     caminho_ficheiro = "alunos.xml"
47     with open(caminho_ficheiro, "w") as f:
48         f.write(string_xml)
49
50
51     return string_xml
52
53 application = Application([SoapXmlService],
54                           tns='soap.alunos.db',
55                           in_protocol=Soap11(validation='lxml'),
56                           out_protocol=Soap11()
57 )
```

```
25
26     for aluno in myresult:
27         id_aluno = str(aluno[0])
28         nome = aluno[1]
29         tag_aluno = doc.createElement('aluno')
30
31         tag_id = doc.createElement('id')
32         content_id = doc.createTextNode(id_aluno)
33         tag_id.appendChild(content_id)
34
35         tag_nome = doc.createElement('nome')
36         content_nome = doc.createTextNode(nome)
37         tag_nome.appendChild(content_nome)
38
39         tag_aluno.appendChild(tag_id)
40         tag_aluno.appendChild(tag_nome)
41
42         tag_alunos.appendChild(tag_aluno)
43
44         doc.appendChild(tag_alunos)
45         string_xml = doc.toprettyxml(indent="\t")
46         caminho_ficheiro = "alunos.xml"
47         with open(caminho_ficheiro, "w") as f:
48             f.write(string_xml)
49
50
51     return string_xml
52
53 application = Application([SoapXmlService],
54                           tns='soap.alunos.db',
55                           in_protocol=Soap11(validation='lxml'),
56                           out_protocol=Soap11()
57 )
```



4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Resultado obtido ao chamar o serviço web com o cliente.

The screenshot shows a browser window displaying the XML response from a web service. The URL is `localhost:8080/getAlunos/alunos`. The XML content is as follows:

```
<?xml version="1.0"?>
<alunos>
  <aluno>
    <id>1</id>
    <nome>Antonio Silva</nome>
  </aluno>
  <aluno>
    <id>2</id>
    <nome>Maria Albertina</nome>
  </aluno>
  <aluno>
    <id>3</id>
    <nome>Jose Ferreira</nome>
  </aluno>
  <aluno>
    <id>4</id>
    <nome>Pedro Costa</nome>
  </aluno>
</alunos>
```

To the right of the browser, a screenshot of a developer tools browser extension (Werkzeug) is shown. The Network tab is selected, showing a single request to `localhost:8080/getAlunos/alunos?type=all`. The request method is GET, status code is 200 OK, and the response content length is 286. The response headers include Content-Type: text/xml; charset=utf-8, Date: Tue, 24 Nov 2020 11:15:56 GMT, and Server: Werkzeug/1.0.1 Python/3.8.2. The request headers include Accept, Accept-Encoding, Accept-Language, Connection, Host, Referer, Sec-Ch-Ua, and Sec-Fetch-Dest.



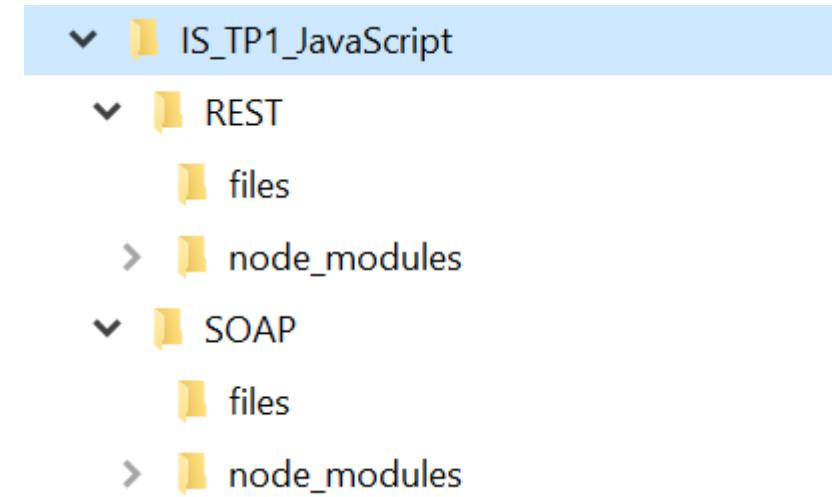
4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

```
* Running on http://localhost:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [24/Nov/2020 11:21:41] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Nov/2020 11:22:11] "GET /soap/alunosXmlBd HTTP/1.1" 200 -
127.0.0.1 - - [24/Nov/2020 11:22:19] "GET /soap/alunosXmlBd HTTP/1.1" 200 -
```

```
C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/TP1/ex5_e_6_codigo/Python/BD/SOAP/SoapXmlService.py"
127.0.0.1 - - [24/Nov/2020 11:22:09] "GET /soap.alunos.db?wsdl HTTP/1.1" 200 2402
127.0.0.1 - - [24/Nov/2020 11:22:11] "POST /soap.alunos.db HTTP/1.1" 200 738
127.0.0.1 - - [24/Nov/2020 11:22:17] "GET /soap.alunos.db?wsdl HTTP/1.1" 200 2402
127.0.0.1 - - [24/Nov/2020 11:22:19] "POST /soap.alunos.db HTTP/1.1" 200 738
```



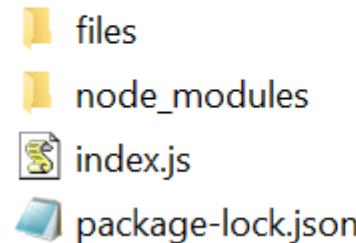
IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:



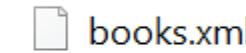


IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

IS_TP1_JavaScript > REST



REST > files





IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
index.js x
1 var express = require('express');
2 var app = express();
3 var fs = require("fs");
4 var parser = require('xml2json-light');
5 var xpath = require('xpath')
6 var dom = require('xmldom').DOMParser
7 const http = require('http');
8 const url = require('url');

9
10 app.get('/booksParse', function (req, res) {
11   const queryObject = url.parse(req.url, true).query;
12   if (queryObject["file"] == "books") {
13     fs.readFile("files/books.xml", 'utf8', function (err, data) {
14       var json = parser.xml2json(data);
15       var i = 0;
16       var dados = "";
17       for (i = 0; i < json.catalog.book.length; i++) {
18         var book = json.catalog.book[i]
19         dados = dados + `Id: ${book.id}\n`
20         dados = dados + `Autor: ${book.author}\n`
21         dados = dados + `Título: ${book.title}\n`
22         dados = dados + `Genero: ${book.genre}\n`
23         dados = dados + `Preço: ${book.price}\n`
24         dados = dados + `Data de Publicação: ${book.publish_date}\n`
25         dados = dados + `Descrição: ${book.description}\n\n`
26       }
27
28       res.setHeader("Content-Type", "text/plain; charset=utf-8");
29       res.end(dados);
30     });
31   } else {
32     res.setHeader("Content-Type", "text/plain; charset=utf-8");
33     res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
34   }
35 }
36
37 })
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
index.js x
37  })
38
39  app.get('/booksXpath', function (req, res) {
40    const queryObject = url.parse(req.url, true).query;
41    if (queryObject["file"] == "books") {
42      fs.readFile("files/books.xml", 'utf8', function (err, data) {
43        var doc = new dom().parseFromString(data)
44        var nodes = xpath.select("/catalog/book[price>30.50 and genre='Computer']", doc)
45        var ids = xpath.select("/catalog/book/@id", doc)
46        var i = 0;
47        var dados = "<h2>RESULTADOS FILTRAGEM XPATH POR:</h2><br><p>price > 30.50 e genre = Computer</p><br>";
48
49        for (i = 0; i < nodes.length; i++) {
50          var bookId = ids[i].value
51          var book = nodes[0];
52          var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;
53          var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;
54          var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;
55          var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;
56          var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;
57          var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;
58          dados = dados + `<p>Id: ${bookId}</p>`;
59          dados = dados + `<p>Autor: ${autor}</p>`;
60          dados = dados + `<p>Título: ${titulo}</p>`;
61          dados = dados + `<p>Genero: ${genero}</p>`;
62          dados = dados + `<p>Preço: ${preco}</p>`;
63          dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`;
64          dados = dados + `<p>Descrição: ${descricao}</p><br>`;
65        }
66        res.setHeader("Content-Type", "text/html; charset=utf-8");
67        res.end(dados);
68      });
69    } else {
70      res.setHeader("Content-Type", "text/plain; charset=utf-8");
71      res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
72    }
73  }
```



index.js

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
55     var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;
56     var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;
57     var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;
58     dados = dados + `<p>Id: ${bookId}</p>`;
59     dados = dados + `<p>Autor: ${autor}</p>`;
60     dados = dados + `<p>Título: ${titulo}</p>`;
61     dados = dados + `<p>Genero: ${genero}</p>`;
62     dados = dados + `<p>Preço: ${preco}</p>`;
63     dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`;
64     dados = dados + `<p>Descrição: ${descricao}</p><br>`;
65   }
66   res.setHeader("Content-Type", "text/html; charset=utf-8");
67   res.end(dados);
68 });
69 }
70 else {
71   res.setHeader("Content-Type", "text/plain; charset=utf-8");
72   res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
73 }
74 }
75 }
76
77 var server = app.listen(3000, function () {
78   var host = server.address().address
79   var port = server.address().port
80   console.log("Books REST Web Service listening at http://localhost:3000")
81 })
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
package-lock.json x
359     "integrity": "sha1-sr906FFKrmFltIF4KdIbLvSZB0w="
360   },
361   "utf8": {
362     "version": "3.0.0",
363     "resolved": "https://registry.npmjs.org/utf8/-/utf8-3.0.0.tgz",
364     "integrity": "sha1-2E8VjFIQ/TyQgp+TZfS618yp/xWppSAHzidGiRrqe4bK4XP9pTRyKFgGJpO3SN7zdX4DeomTrwaseCHovfpFcqQ=="
365   },
366   "utils-merge": {
367     "version": "1.0.1",
368     "resolved": "https://registry.npmjs.org/utils-merge/-/utils-merge-1.0.1.tgz",
369     "integrity": "sha1-n5VxD1CiZ5R7LMwSR0HBAoQn5xM="
370   },
371   "vary": {
372     "version": "1.1.2",
373     "resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",
374     "integrity": "sha1-IpnwLG3tMNSllhsLn3RSShj2NPw="
375   },
376   "xml2json-light": {
377     "version": "1.0.6",
378     "resolved": "https://registry.npmjs.org/xml2json-light/-/xml2json-light-1.0.6.tgz",
379     "integrity": "sha1-ERaHcQorAfVS/hj0GK31Ed5UfzQ="
380   },
381   "xmldom": {
382     "version": "0.4.0",
383     "resolved": "https://registry.npmjs.org/xmldom/-/xmldom-0.4.0.tgz",
384     "integrity": "sha1-2E93k08T30Ugs+34HBSTQLVtpi6mCddaY8u0+pMNk1pqSjV5vElzn4mmh6KLxN3hki8rNcHSYzILoh3TEWORvA=="
385   },
386   "xpath": {
387     "version": "0.0.32",
388     "resolved": "https://registry.npmjs.org>xpath/-/xpath-0.0.32.tgz",
389     "integrity": "sha1-rxMJhSIoiO8vXcWvSiFkqhV96GjiD5wYb8/QHdoRyQvraTpp4IEv944nhGausZZ3u7dhQXteZuZbaqfpB7uYw=="
390 }
```



```
Linha de comandos - node index.js

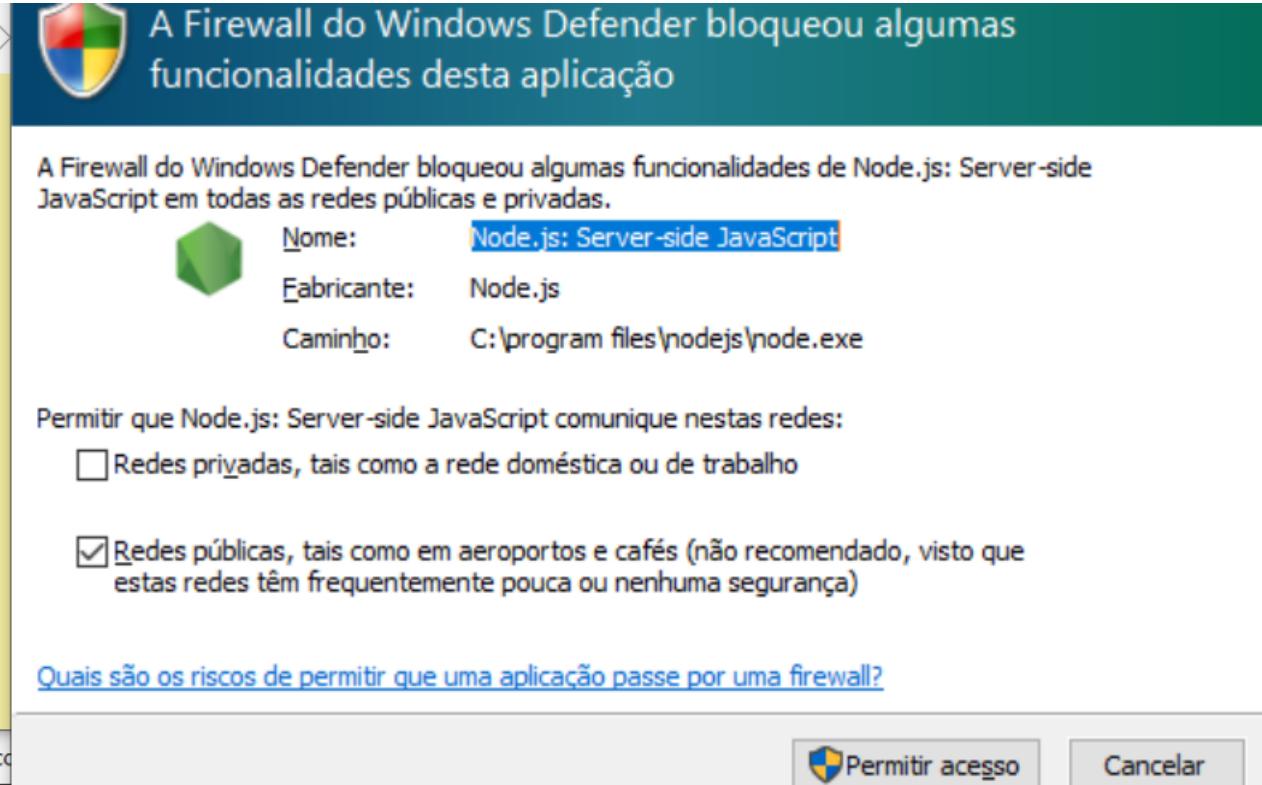
Directory of C:\testes

15/11/2021 16:24 <DIR> .
15/11/2021 16:24 <DIR> ..
15/11/2021 16:24 <DIR> IS_TP1_JavaScript
15/11/2021 16:24 <DIR> IS_TP1_Python
    0 File(s)          0 bytes
    4 Dir(s)  656 517 922 816 bytes free

C:\testes>cd IS_TP1_JavaScript

C:\testes\IS_TP1_JavaScript>cd REST

C:\testes\IS_TP1_JavaScript\REST>node index.js
Books REST Web Service listening at http://localhost:3000
```





IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

localhost:3000/booksParse?file=books

Id: bk101
Autor: Gambardella, Matthew
Título: XML Developer's Guide
Genero: Computer
Preço: 44.95
Data de Publicação: 2000-10-01
Descrição: An in-depth look at creating applications with XML.

Id: bk102
Autor: Ralls, Kim
Título: Midnight Rain
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2000-12-16
Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Id: bk103
Autor: Corets, Eva
Título: Maeve Ascendant
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2000-11-17
Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Id: bk104
Autor: Corets, Eva
Título: Oberon's Legacy
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2001-03-10
Descrição: In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.

Id: bk105
Autor: Corets, Eva
Título: The Sundered Grail
Genero: Fantasy
Preço: 5.95
Data de Publicação: 2001-09-10
Descrição: The two daughters of Maeve, half-sisters, battle one another for control of England. Sequel to Oberon's Legacy.

Id: bk106
Autor: Randall, Cynthia
Título: Lover Birds
Genero: Romance
Preço: 4.95
Data de Publicação: 2000-09-02

<http://localhost:3000/booksParse?file=books>

Network tab details:
Request URL: http://localhost:3000/booksParse?file=books
Request Headers:
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: pt-PT,pt;q=0.9,pt-BR;q=0.8,en-US;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Host: localhost:3000
Sec-Ch-Ua: "Chromium";v="86", "\N
Google Chrome";v="86"
Sec-Ch-Ua-Mobile: ?0
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows !
AppleWebKit/537.36 (KHTML, like C
240.198 Safari/537.36
Query String Parameters: file: books



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

localhost:3000/booksXpath?file=books

RESULTADOS FILTRAGEM XPATH POR:

price > 30.50 e genre = Computer

Id: bk101

Autor: Gambardella, Matthew

Titulo: XML Developer's Guide

Genero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

Id: bk102

Autor: Gambardella, Matthew

Titulo: XML Developer's Guide

Genero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

...

Network

Filter: booksXpath?file=books

Name: booksXpath?file=books, extn-utils.html, extn-utils.js

Headers

Request URL: http://localhost:3000/booksXpath?file=books

Request Method: GET

Status Code: 200 OK

Remote Address: [::1]:3000

Referrer Policy: strict-origin-when-cross-origin

Response Headers

Connection: keep-alive

Content-Length: 1063

Content-Type: text/html; charset=utf-8

Date: Mon, 23 Nov 2020 15:38:33 GMT

Keep-Alive: timeout=5

X-Powered-By: Express

Request Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7

Connection: keep-alive

Host: localhost:3000

sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google"

3 requests | 3.3 kB transferred | 3.1 ms



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

IS_TP1_JavaScript > SOAP

- files
- node_modules
- Books_ws_wsdl.wsdl
- client.js
- index.js
- package-lock.json

SOAP > files

- books.xml



index.js

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
1 var soap = require('soap');
2 var express = require('express');
3 var fs = require('fs');
4 var parser = require('xml2json-light');
5 var xpath = require('xpath')
6 var dom = require('xmldom').DOMParser
7
8 function books_parse(args) {
9   if(args.type == "parse") {
10     var data = fs.readFileSync( "files/books.xml", 'utf8' );
11     var json = parser.xml2json(data);
12     var dados = "<h2>PARSE DO FICHEIRO - books.xml</h2><br>";
13     var i = 0;
14     for(i = 0; i < json.catalog.book.length; i++) {
15       var book = json.catalog.book[i]
16       dados = dados + `<p>Id: ${book.id}</p>`
17       dados = dados + `<p>Autor: ${book.author}</p>`
18       dados = dados + `<p>Título: ${book.title}</p>`
19       dados = dados + `<p>Genero: ${book.genre}</p>`
20       dados = dados + `<p>Preço: ${book.price}</p>`
21       dados = dados + `<p>Data de Publicação: ${book.publish_date}</p>`
22       dados = dados + `<p>Descrição: ${book.description}</p><br>`
23     }
24     return {
25       result: dados
26     }
27 }
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
index.js x
27 }
28 if(args.type == "xpath") {
29     var data = fs.readFileSync( "files/books.xml", 'utf8' );
30     var doc = new dom().parseFromString(data)
31     var nodes = xpath.select("/catalog/book[genre='Fantasy' and contains(author, 'Eva')]", doc)
32     var ids = xpath.select("/catalog/book/@id", doc)
33     var i = 0;
34     var dados = "<h2>XPATH - books.xml</h2><p>Filtragem por livros do género <i>Fantasy</i>, \
35     e onde o nome do autor contém a string, <i>Eva</i></p><br>";
36     for(i = 0; i < nodes.length; i++) {
37         var bookId = ids[i].value
38         var book = nodes[0];
39         var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;
40         var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;
41         var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;
42         var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;
43         var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;
44         var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;
45         dados = dados + `<p>Id: ${bookId}</p>`;
46         dados = dados + `<p>Autor: ${autor}</p>`;
47         dados = dados + `<p>Título: ${titulo}</p>`;
48         dados = dados + `<p>Genero: ${genero}</p>`;
49         dados = dados + `<p>Preço: ${preco}</p>`;
50         dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`;
51         dados = dados + `<p>Descrição: ${descricao}</p><br>`;
52     }
53     return {
54         result: dados
55     }
56 }
57 }
```



index.js

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
57 }  
58  
59 var serviceObject = {  
60   BooksParseService: {  
61     BooksParseServiceSoapPort: {  
62       BooksParse: books_parse  
63     },  
64     BooksParseServiceSoap12Port: {  
65       BooksParse: books_parse  
66     }  
67   }  
68 };  
69  
70 var xml = fs.readFileSync('Books_ws_wsdl.wsdl', 'utf8');  
71 var app = express();  
72  
73 app.get('/', function (req, res) {  
74   res.send('SOAP webservice - Parse de ficheiros XML e XPATH<br />');  
75 })  
76  
77 var port = 8000;  
78 app.listen(port, function () {  
79   console.log('Listening on port ' + port);  
80   var wsdl_path = "/wsdl";  
81   soap.listen(app, wsdl_path, serviceObject, xml);  
82   console.log("Check http://localhost:" + port + wsdl\_path + "?wsdl to see if the service is working");  
83 })
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

client.js

```
1 var express = require('express');
2 var app = express();
3 var soap = require('soap');
4 var url = 'http://localhost:8000/wsdl?wsdl';
5
6 app.get('/callBooksParse', function (req, res) {
7   soap.createClient(url, function (err, client) {
8     if (err){
9       throw err;
10    }
11    var args = {
12      type: 'parse'
13    }
14    client.BooksParse(args, function (err, result) {
15      if (err) {
16        throw err;
17      }
18      else {
19        res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
20        res.end(result["result"][0]);
21      }
22    });
23  });
24 });
25 })
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

client.js

```
25
26 app.get('/callBooksXpath', function (req, res) {
27   soap.createClient(url, function (err, client) {
28     if (err){
29       throw err;
30     }
31     var args = {
32       type: 'xpath'
33     }
34     client.BooksParse(args, function (err, result) {
35       if (err) {
36         throw err;
37       }
38       else {
39         res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
40         res.end(result["result"][0]);
41       }
42     });
43   });
44 })
45
46 var server = app.listen(3000, function () {
47   var host = server.address().address
48   var port = server.address().port
49   console.log("Example app listening at http://localhost:3000")
50 })
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions targetNamespace="http://tempuri.org/"
3      xmlns:s="http://www.w3.org/2001/XMLSchema"
4      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
5      xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
6      xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
7      xmlns:tns="http://tempuri.org/"
8      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
9      xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
10     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
11     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
12  <!-- Definição dos pedidos e respostas para o método parse_books -->
13  <wsdl:types>
14      <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
15          <s:element name="BooksParseRequest">
16              <s:complexType>
17                  <s:sequence>
18                      <s:element minOccurs="1" maxOccurs="unbounded" name="type" type="s:string"/>
19                  </s:sequence>
20              </s:complexType>
21          </s:element>
22          <s:element name="BooksParseResponse">
23              <s:complexType>
24                  <s:sequence>
25                      <s:element minOccurs="1" maxOccurs="unbounded" name="result" type="s:string"/>
26                  </s:sequence>
27              </s:complexType>
28          </s:element>
29      </s:schema>
30  </wsdl:types>
31  <!-- Definição dos dados e do seu tipo entre o cliente e o servidor -->
32  <wsdl:message name="BooksParseSoapIn">
33      <wsdl:part name="parameters" element="tns:BooksParseRequest"/>
34  </wsdl:message>
35  <wsdl:message name="BooksParseSoapOut">
36      <wsdl:part name="parameters" element="tns:BooksParseResponse"/>
37  </wsdl:message>
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
37 </wsdl:message>
38 <wsdl:portType name="BooksParseSoapPort">
39   <!-- Definição da operação books_parse, definida no index.js -->
40   <wsdl:operation name="BooksParse">
41     <wsdl:input message="tns:BooksParseSoapIn"/>
42     <wsdl:output message="tns:BooksParseSoapOut"/>
43   </wsdl:operation>
44 </wsdl:portType>
45   <!-- Definição de como as mensagens serão transmitidas -->
46 <wsdl:binding name="BooksParseServiceSoapBinding" type="tns:BooksParseSoapPort">
47   <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
48   <wsdl:operation name="BooksParse">
49     <soap:operation soapAction="BooksParse" style="document"/>
50   <wsdl:input>
51     <soap:body use="literal"/>
52   </wsdl:input>
53   <wsdl:output>
54     <soap:body use="literal"/>
55   </wsdl:output>
56 </wsdl:operation>
57 </wsdl:binding>
58 <wsdl:binding name="BooksParseServiceSoap12Binding" type="tns:BooksParseSoapPort">
59   <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
60   <wsdl:operation name="BooksParse">
61     <soap12:operation soapAction="BooksParse" style="document"/>
62   <wsdl:input>
63     <soap12:body use="literal"/>
64   </wsdl:input>
65   <wsdl:output>
66     <soap12:body use="literal"/>
67   </wsdl:output>
68 </wsdl:operation>
69 </wsdl:binding>
70   <!-- Definição do serviço -->
71 <wsdl:service name="BooksParseService">
72   <wsdl:port name="BooksParseServiceSoapPort" binding="tns:BooksParseServiceSoapBinding">
73     <soap:address location="http://localhost:8000/wsdl"/>
```



Books_ws_wsdl.wsdl X

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

```
52      </wsdl:input>
53    <wsdl:output>
54      <soap:body use="literal"/>
55    </wsdl:output>
56  </wsdl:operation>
57</wsdl:binding>
58<wsdl:binding name="BooksParseServiceSoap12Binding" type="tns:BooksParseSoapPort">
59  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
60  <wsdl:operation name="BooksParse">
61    <soap12:operation soapAction="BooksParse" style="document"/>
62    <wsdl:input>
63      <soap12:body use="literal"/>
64    </wsdl:input>
65    <wsdl:output>
66      <soap12:body use="literal"/>
67    </wsdl:output>
68  </wsdl:operation>
69</wsdl:binding>
70<!-- Definição do serviço -->
71<wsdl:service name="BooksParseService">
72  <wsdl:port name="BooksParseServiceSoapPort" binding="tns:BooksParseServiceSoapBinding">
73    <soap:address location="http://localhost:8000/wsdl"/>
74  </wsdl:port>
75  <wsdl:port name="BooksParseServiceSoap12Port" binding="tns:BooksParseServiceSoap12Binding">
76    <soap12:address location="http://localhost:8000/wsdl"/>
77  </wsdl:port>
78</wsdl:service>
79</wsdl:definitions>
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

Linha de comandos - node index.js

```
C:\Integração de Sistemas\TP1\ex5_codigo\JavaScript\SOAP>node index.js
Listening on port 8000
Check http://localhost:8000/wsdl?wsdl to see if the service is working
```

Linha de comandos - node client.js

```
C:\Integração de Sistemas\TP1\ex5_codigo\JavaScript\SOAP>node client.js
Example app listening at http://localhost:3000
```



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

localhost:3000/callBooksParse +
localhost:3000/callBooksParse

PARSE DO FICHEIRO - books.xml

Id: bk101

Autor: Gambardella, Matthew

Titulo: XML Developer's Guide

Genero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

Id: bk102

Autor: Ralls, Kim

Titulo: Midnight Rain

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Id: bk103

Autor: Corets, Eva

Network

Preserve log Disable cache Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest O

Has blocked cookies Blocked Requests

Name	Headers	Preview	Response	Initiator	Timing	Cookies
callBooksParse						
extrn-utils.html						
extrn-utilsjs						

General

Request URL: http://localhost:3000/callBooksParse
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:3000
Referrer Policy: strict-origin-when-cross-origin

Response Headers

Connection: keep-alive
Content-Type: text/html; charset=utf-8
Date: Mon, 23 Nov 2020 15:48:36 GMT
Keep-Alive: timeout=5
Transfer-Encoding: chunked
X-Powered-By: Express

Request Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7
Connection: keep-alive
Host: localhost:3000
sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google Chro

3 requests | 5.9 kB transferred



IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

localhost:3000/callBooksXpath

localhost:3000/callBooksXpath

XPATH - books.xml

Filtragem por livros do género *Fantasy*, e onde o nome do autor contém a string, *Eva*

Id: bk101

Autor: Corets, Eva

Título: Maeve Ascendant

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Id: bk102

Autor: Corets, Eva

Título: Maeve Ascendant

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Network

Filter: Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Has blocked cookies Blocked Requests

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms 180 ms 200 ms

Name	Headers	Preview	Response	Initiator	Timing	Cookies
callBooksXpath	General	Request URL: http://localhost:3000/callBooksXpath	Request Method: GET	Status Code: 200 OK	Remote Address: [::1]:3000	Referrer Policy: strict-origin-when-cross-origin
extrn-utils.html	Response Headers	Connection: keep-alive	Content-Type: text/html; charset=utf-8	Date: Mon, 23 Nov 2020 15:49:12 GMT	Keep-Alive: timeout=5	Transfer-Encoding: chunked
extrn-utils.js		X-Powered-By: Express				

General

Request URL: http://localhost:3000/callBooksXpath

Request Method: GET

Status Code: 200 OK

Remote Address: [::1]:3000

Referrer Policy: strict-origin-when-cross-origin

Response Headers

Connection: keep-alive

Content-Type: text/html; charset=utf-8

Date: Mon, 23 Nov 2020 15:49:12 GMT

Keep-Alive: timeout=5

Transfer-Encoding: chunked

X-Powered-By: Express

Request Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

Accept-Encoding: gzip, deflate, br

Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7

Connection: keep-alive

Host: localhost:3000

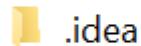
sec-ch-ua: "Chromium";v="86", "\Not\A;Brand";v="99", "Google Chrom"



IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

IS_TP1_Python >

Nome



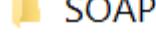
IS_TP1_Python > REST



Nome



files



python

IS_TP1_Python > REST > python

Nome



Books_ws_rest.py

IS_TP1_Python > REST > files

Nome



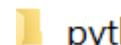
books.xml

> IS_TP1_Python > SOAP

Nome



files



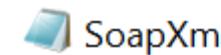
python

IS_TP1_Python > SOAP > python

Nome



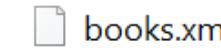
Client.py



SoapXmlService.py

IS_TP1_Python > SOAP > files

Nome



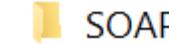
books.xml

> IS_TP1_Python > BD >

] Nome



REST



SOAP

> IS_TP1_Python > BD > REST

Nome



alunos.xml



Books_ws_rest.py

IS_TP1_Python > BD > SOAP

Nome



alunos.xml



Client.py



SoapXmlService.py



IS_TP1_Python >

Nome

.idea

BD

REST

SOAP

IS_TP1_Python > REST

Nome

files

python

IS_TP1_Python > REST > python

Nome

Books_ws_rest.py

IS_TP1_Python > REST > files

Nome

books.xml

Books_ws_rest.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
1  from xml.dom import minidom
2  from flask import Flask, request
3  from lxml import etree
4  from werkzeug.wrappers import Response
5  app = Flask(__name__)

6
7
8  @app.route('/')
9  def index():
10     html = "<h3>Esta aplicação contém os seguintes métodos: </h3>"
11     html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
12     html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
13     return html
14
15  @app.route('/getBooks/books', methods=['GET'])
16  def getBooks():
17      typeRequest = request.args.get('type')
18      if typeRequest.upper() == "HTML":
19          xmldoc = minidom.parse('../files/books.xml')
20          itemlist = xmldoc.getElementsByTagName('book')
21          data = ""
22          i = 0
23          for i in range(0, len(itemlist)):
24              node = itemlist.item(i)
25              autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
26              titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
27              genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
28              preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
29              data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
30              descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
31              data = data + "<h5>Autor: " + autor + "</h5>"
32              data = data + "<h5>Título: " + titulo + "</h5>"
33              data = data + "<h5>Genero: " + genero + "</h5>"
34              data = data + "<h5>Preço: " + preco + "</h5>"
35              data = data + "<h5>Data de Publicação: " + data_publicacao + "</h5>"
36              data = data + "<h5>Descrição: " + descricao + "</h5>"
37              data = data + "<br>"
```



Books_ws_rest.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
37         data = data + "<br>"  
38     return data  
39     if typeRequest.upper() == "XML":  
40         root = etree.parse('../files/books.xml').getroot()  
41         xml = etree.tostring(root, pretty_print=True)  
42         return Response(xml, mimetype='text/xml')  
43  
44     @app.route('/getBooks/booksXpath', methods=['GET'])  
45     def getBooksXpath():  
46         root = etree.parse('../files/books.xml')  
47         books = root.xpath('/catalog//book[price<7.00 and price>5.00]')  
48         data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"  
49         i = 0  
50         for i in range(0, len(books)):  
51             autor = books[i][0].text  
52             titulo = books[i][1].text  
53             genero = books[i][2].text  
54             preco = books[i][3].text  
55             data_publicacao = books[i][4].text  
56             descricao = books[i][5].text  
57             data = data + "<h5>Autor: " + autor + "<h5>"  
58             data = data + "<h5>Título: " + titulo + "<h5>"  
59             data = data + "<h5>Genero: " + genero + "<h5>"  
60             data = data + "<h5>Preço: " + preco + "<h5>"  
61             data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"  
62             data = data + "<h5>Descrição: " + descricao + "<h5>"  
63             data = data + "<br>"  
64     return data  
65  
66     if __name__ == "__main__":  
67         app.run(host='localhost', port=8080)
```



Add Python Interpreter

- Virtualenv Environment
- Conda Environment
- System Interpreter
- Pipenv Environment
- SSH Interpreter
- Vagrant
- WSL
- Docker
- Docker Compose

New environment

Location:

C:\testes\IS_TP1_Python\venv



Base interpreter:

C:\Users\jribe\AppData\Local\Programs\Python\Python39\python.exe



Inherit global site-packages

Make available to all projects

Existing environment

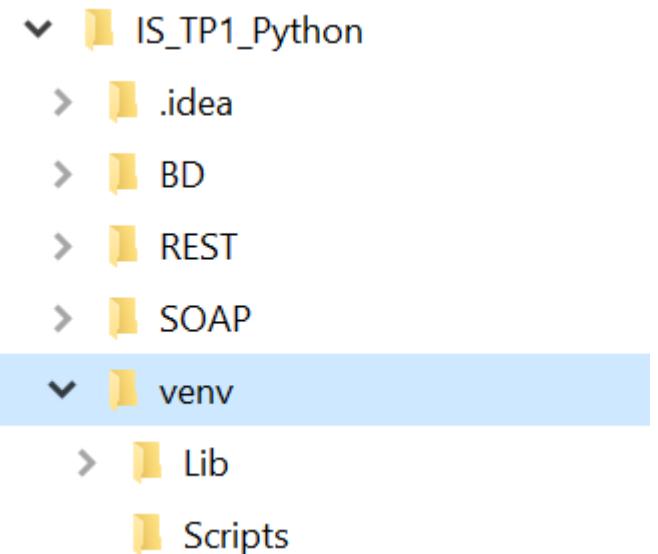
Interpreter: <No interpreter>



Make available to all projects

OK

Cancel





Linha de comandos

— □ ×

```
C:\testes\IS_TP1_Python\REST\python>c:\testes\IS_TP1_Python\venv\Scripts\python.exe Books_ws_rest.py
```

```
Traceback (most recent call last):
```

```
  File "C:\testes\IS_TP1_Python\REST\python\Books_ws_rest.py", line 2, in <module>
    from flask import Flask, request
```

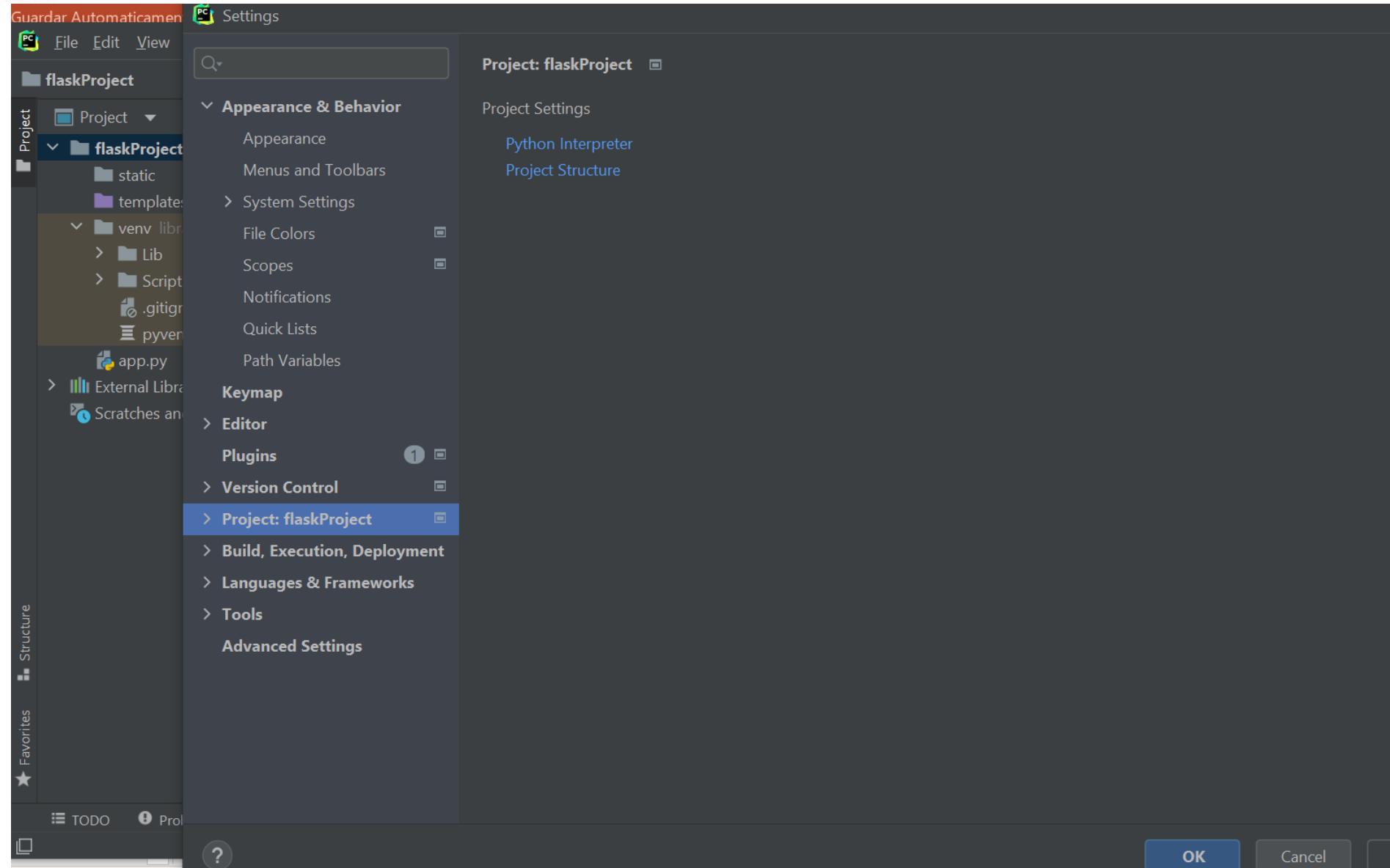
```
ModuleNotFoundError: No module named 'flask'
```

```
C:\testes\IS_TP1_Python\REST\python>
```



<https://www.jetbrains.com/help/pycharm/creating-web-application-with-flask.html>

The screenshot shows a web browser window displaying a PyCharm documentation page. The URL in the address bar is [jetbrains.com/help/pycharm/creating-web-application-with-flask.html](https://www.jetbrains.com/help/pycharm/creating-web-application-with-flask.html). The page header includes the PyCharm logo (PC), version 2021.2, and links for "Upcoming webinar" and "Shortcuts: Windows". A prominent blue button on the right says "Get PyCharm". The main content area has a teal header with the text "10 Reasons You'll Love PyCharm Even More in 2021" and the date "Tuesday, November 23, 2021, 17:00-18:00 GMT". Below this, a large section title "Creating Web Applications with Flask" is displayed in bold black font. A sub-section title "Web Development / Flask / Creating Web Applications with Flask" is also present. On the left sidebar, there is a navigation tree under "Web Development" with "Flask" expanded, showing "Creating a Flask Project" and "Creating Web Applications with Flask" (which is highlighted in blue). Other items in the tree include "Installation guide", "Get started", "Configure PyCharm", "Configure projects in PyCharm", "Work with source code", "Run, debug, test, and deploy", "Analyze applications", "Python", and "Creating a Login Form". A callout box in the center says "Professional feature: download PyCharm Professional ↗ to try.". The main text area describes the tutorial as explaining how to develop a web-based application using the Flask web framework, mentioning a MeteoMaster application that processes meteo data stored in a database and presents it in the form of charts. A small chart titled "Scatter chart" is shown at the bottom.





Settings

Project: flaskProject > Python Interpreter

Python Interpreter: Python 3.9 (IS_TP1_Python) C:\testes\IS_TP1_Python\venv\Scripts\python.exe

Appearance & Behavior

- Appearance
- Menus and Toolbars
- System Settings
- File Colors
- Scopes
- Notifications
- Quick Lists
- Path Variables

Keymap

Editor

Plugins 1

Version Control

Project: flaskProject

Python Interpreter

Project Structure

Build, Execution, Deployment

Package	Version	Latest version
pip	21.1.2	▲ 21.3.1
setuptools	57.0.0	▲ 59.1.0
wheel	0.36.2	▲ 0.37.0



Available Packages

Q lxml X

lxml	Description
lxml	Powerful and Pythonic XML processing library combining libxml2/libxslt with the ElementTree API.
lxml-asserts	
lxml-element-maker	
lxml-mate	
lxml-odidev	
lxml-stubs	
lxml-to-dict	
lxml-wrapper	
lxml-xpath-ipaddress	
lxml-xpath2-functions	

Package 'lxml' installed successfully

Install Package Manage Repositories



The screenshot shows the PyCharm settings interface. The left sidebar has a tree view with the following expanded sections: Appearance & Behavior, System Settings, Keymap, Editor, Plugins (with one update available), Version Control, Project: flaskProject (selected), Build, Execution, Deployment, Languages & Frameworks, Tools, and Advanced Settings. Under Project: flaskProject, Python Interpreter is selected. The main pane displays the Python Interpreter configuration for the flaskProject. It shows the current Python interpreter as "Python 3.9 (IS_TP1_Python) C:\testes\IS_TP1_Python\venv\Scripts\python.exe". Below this is a table of installed packages:

Package	Version	Latest version
lxml	4.6.4	4.6.4
pip	21.1.2	▲ 21.3.1
setuptools	57.0.0	▲ 59.1.0
wheel	0.36.2	▲ 0.37.0

A green status bar at the bottom says "Package 'lxml' installed successfully". At the bottom right are OK, Cancel, and Apply buttons.



The screenshot shows the 'New Project' dialog in PyCharm. On the left, a sidebar lists various project templates: New Project, Pure Python, Django, Flask (which is selected), Google App Engine, Pyramid, Web2Py, Scientific, Angular CLI, AngularJS, Bootstrap, HTML5 Boilerplate, React, and React Native. The main area is titled 'Python Interpreter: New Virtualenv environment'. It shows a radio button selected for 'New environment using Virtualenv' (with a dropdown menu icon). The 'Location:' field contains 'C:\testes\IS_TP1_Python\REST\venv'. Below it, 'Base interpreter:' is set to 'C:\Users\jribe\AppData\Local\Programs\Python\Python39\python.exe'. There are two unchecked checkboxes: 'Inherit global site-packages' and 'Make available to all projects'. Another radio button is available for 'Previously configured interpreter' with a dropdown menu. The 'Interpreter:' dropdown shows '<No interpreter>'. A 'More Settings' section is at the bottom.



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The current file is Books_ws_rest.py - Books_ws_rest.py. The left sidebar has tabs for Project, Favorites, Structure, and Terminal. The Terminal window shows a Windows PowerShell session with the command PS C:\testes\IS_TP1_Python>. A file browser dialog is open in the center, titled 'Open File or Project'. It shows a tree view of the 'C:' drive, with the path 'C:\testes\IS_TP1_Python' selected. The dialog also includes a search bar and buttons for OK and Cancel.

File Edit View Navigate Code Refactor Run Tools VCS Window Help Books_ws_rest.py - Books_ws_rest.py

IS_TP1_Python

Project

IS_TP1_Python [Python] C:\testes\IS_TP1_Python

- > BD
- > REST
- > SOAP
- External Libraries
- Scratches and Consoles

Books_ws_rest.py

No Python in this project

Open File or Project

C:\testes\IS_TP1_Python

Hide path

C:

- > Dev_IPVC
- > Documents and Settings
- > Drivers
- > Program Files
- > Program Files (x86)
- > temp
- > testes
 - > IS_TP1_JavaScript
 - > IS_TP1_Python
- > Users
- > Windows
- > xampp

Todos: <h3>"

- Ficheiro books.xml<

Terminal: Local +

Windows PowerShell

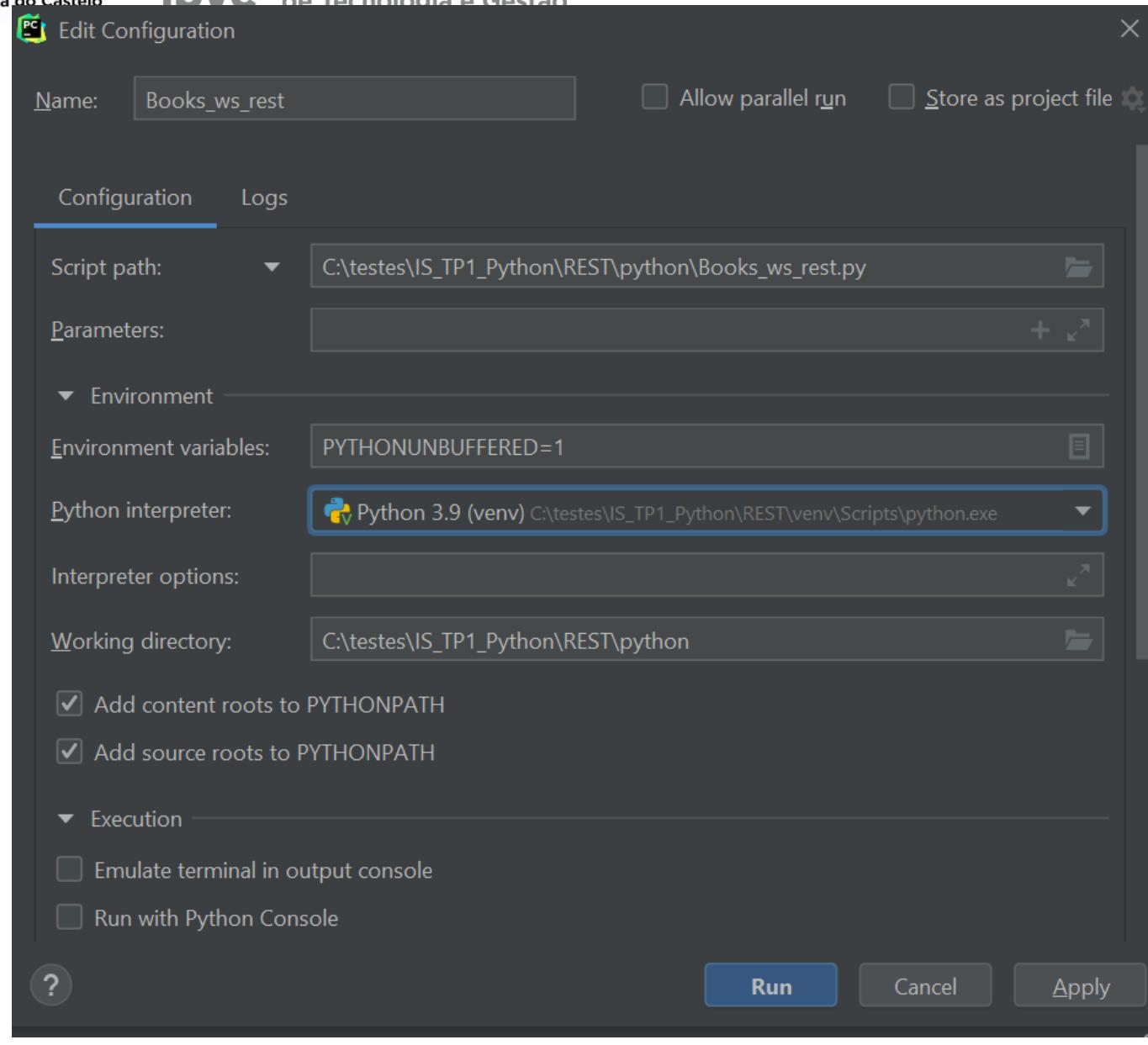
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/powershell>

PS C:\testes\IS_TP1_Python> []

?

OK Cancel





```
Run: Books_ws_rest ×
▶ C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/REST/python/Books_ws_rest.py
↑
↓
Traceback (most recent call last):
File "C:\testes\IS_TP1_Python\REST\python\Books_ws_rest.py", line 3, in <module>
    from lxml import etree
ModuleNotFoundError: No module named 'lxml'

Process finished with exit code 1
```



3. If it doesn't work, try "pip3 install lxml" or "python -m pip install lxml".
4. Wait for the installation to terminate successfully.

The package is now installed on your macOS.

How to Install lxml in PyCharm?

Given a [PyCharm](#) project. How to install the lxml library in your project within a [virtual environment](#) or globally? Here's a [solution](#) that always works:

- Open `File > Settings > Project` from the PyCharm menu.
- Select your current project.
- Click the `Python Interpreter` tab within your project tab.
- Click the small + symbol to add a new library to the project.
- Now type in the library to be installed, in your example "lxml" without quotes, and click `Install Package`.
- Wait for the installation to terminate and close all pop-ups.

Here's the general package installation process as a short animated video—it works analogously for lxml if you type in "lxml" in the search field instead:

[←](#) [→](#) [C](#)[codetd.com/en/article/12931460](#)[Home](#) [Mobile](#) [Internet](#) [Server](#) [Language](#) [Enterprise](#) [Database](#) [News](#) [Others](#)[Search](#)

Install lxml in PyCharm

[Others](#)

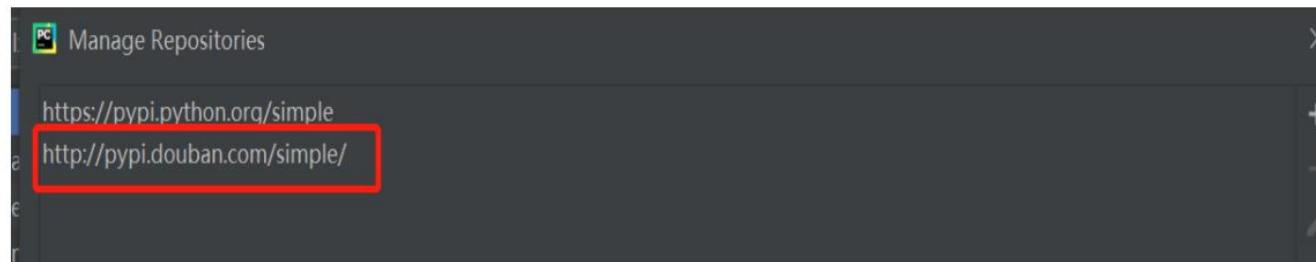
2021-03-28 03:11:13

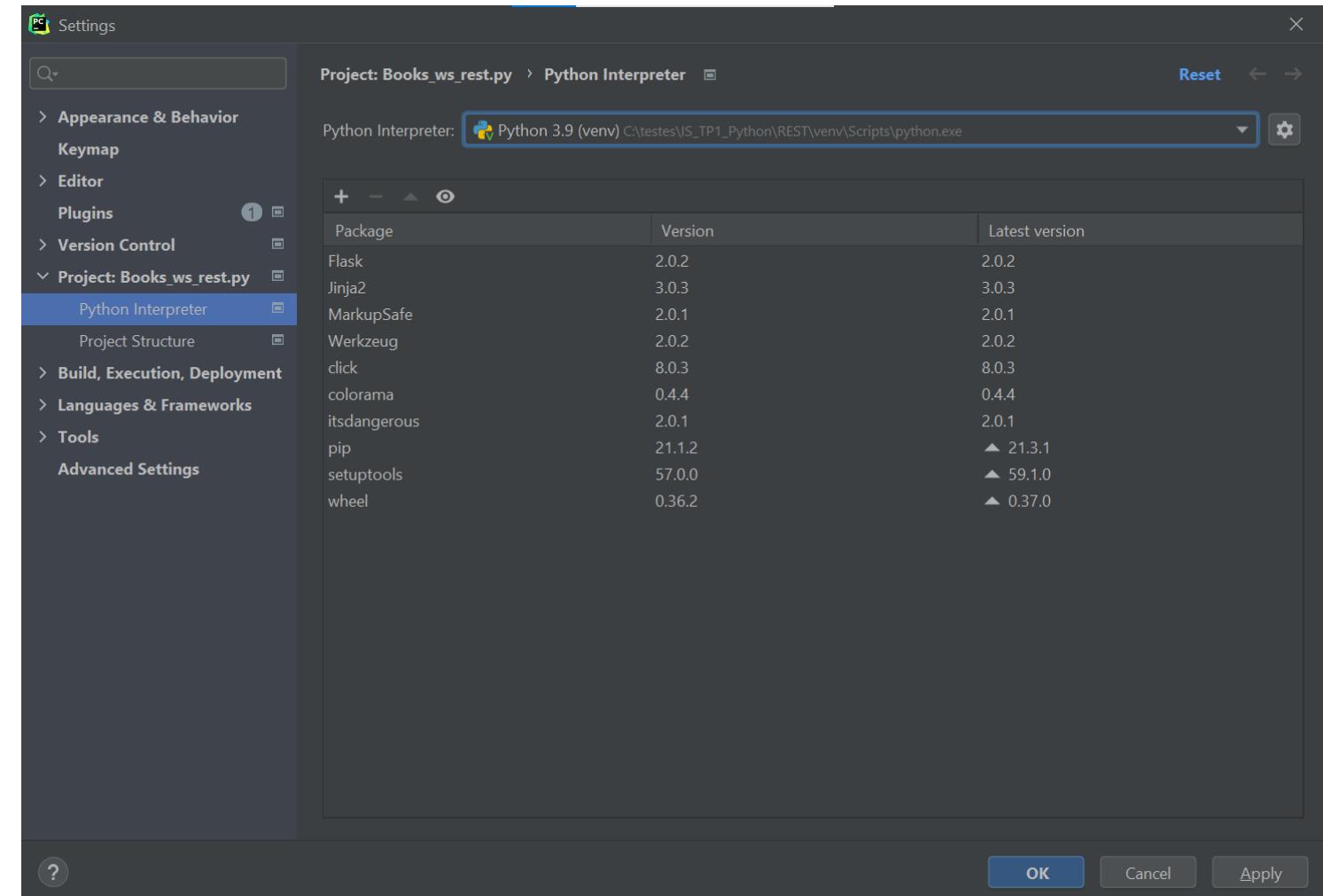
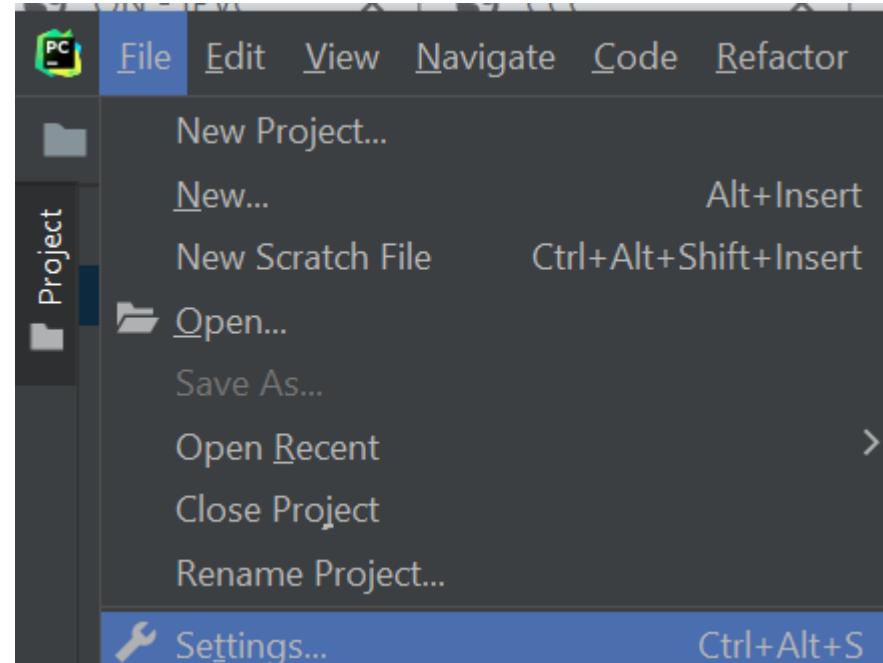
views: null

The point of doubt is that lxml is successfully installed in git cmd, but it cannot be used in PyCharm, and an error is reported.

ModuleNotFoundError: No module named 'lxml'

In git cmd, it was successfully installed using pip install lxml -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com, so choose new website resources,







Available Packages

Q lxml

- lxml-stubs
- lxml-to-dict
- lxml-wrapper
- lxml-xpath-ipaddress
- lxml-xpath2-functions
- lxml2dict
- lxml2json
- lxmlHtml
- lxmlbind
- lxmlmechanize
- lxxmlmeld
- lxxmlmiddleware
- lxmlproc
- lxmlrpc_monkey
- lxmlx
- lxmlextree
- medialog.lxml
- plone.recipe.lxml

Description

Powerful and Pythonic XML processing library combining libxml2/libxslt with the ElementTree API.

Version

4.6.4

Author

lxml dev team

<mailto:lxml-dev@lxml.de>
<https://lxml.de/>

Specify version 4.6.4 ▾

Options

Package 'lxml' installed successfully

Install Package Manage Repositories



Settings

Project: Books_ws_rest.py > Python Interpreter

Python Interpreter: Python 3.9 (venv) C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe

Appearance & Behavior

Keymap

Editor

Plugins 1

Version Control

Project: Books_ws_rest.py

Python Interpreter

Project Structure

Build, Execution, Deployment

Languages & Frameworks

Tools

Advanced Settings

+ - ▲ ○

Package	Version	Latest version
Flask	2.0.2	2.0.2
Jinja2	3.0.3	3.0.3
MarkupSafe	2.0.1	2.0.1
Werkzeug	2.0.2	2.0.2
click	8.0.3	8.0.3
colorama	0.4.4	0.4.4
itsdangerous	2.0.1	2.0.1
lxml	4.6.4	4.6.4
pip	21.1.2	▲ 21.3.1
setuptools	57.0.0	▲ 59.1.0
wheel	0.36.2	▲ 0.37.0

Package 'lxml' installed successfully

?

OK Cancel Apply



The screenshot shows a terminal window within a development environment. The title bar says "Run: Books_ws_rest". The command run was "C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/REST/python/Books_ws_rest.py". The output shows:

```
* Serving Flask app 'Books_ws_rest' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://localhost:8080/ (Press CTRL+C to quit)
```

The bottom navigation bar includes "Run", "Debug", "TODO", "Problems", "Terminal", "Python Packages", and "Python Console". A status message at the bottom left says "Packages installed successfully. Installed packages: 'bxml' (a minute ago)".

← → ⏪ ⓘ localhost:8080

Esta aplicação contém os seguintes métodos:

[XML Parse - Ficheiro books.xml](#)

Resposta em XML e HTML

[XML Parse e XPATH - Ficheiro books.xml](#)

← → C ⓘ localhost:8080/getBooks/books

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either

Run: Books_ws_rest ×

```
rv = self.dispatch_request()
File "C:\testes\IS_TP1_Python\REST\venv\lib\site-packages\flask\app.py", line 1502, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
File "C:\testes\IS_TP1_Python\REST\python\Books_ws_rest.py", line 22, in getBooks
    if typeRequest.upper() == "HTML":
AttributeError: 'NoneType' object has no attribute 'upper'
127.0.0.1 - - [15/Nov/2021 18:45:08] "GET /getBooks/books HTTP/1.1" 500 -
teste1
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Books_ws_rest.py - Books_ws_rest.py

IS_TP1_Python > REST > python > Books_ws_rest.py

Project

- IS_TP1_Python [Python] C:\testes\IS_TP1_Python
 - BD
 - REST
 - files
 - books.xml
 - python
 - Books_ws_rest.py
 - venv library root
 - SOAP
- External Libraries
- Scratches and Consoles

Books_ws_rest.py books.xml

```
def getBooks():
    print("teste1")
    typeRequest = request.args.get('type')
    print(typeRequest)
    print("teste2")

    if typeRequest.upper() == "HTML":
        xmldoc = minidom.parse('C:\testes\books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
```

getBooks()

Run: Books_ws_rest ×

```
* Debug mode: off
127.0.0.1 - - [15/Nov/2021 18:43:27] "GET / HTTP/1.1" 200 -
teste1
None
teste2
[2021-11-15 18:43:28,284] ERROR in app: Exception on /getBooks/books [GET]
Traceback (most recent call last):
```

Structure

Favorites



localhost:8080/getBooks/books?type=html

Autor: Gambardella, Matthew

Título: XML Developer's Guide

Gênero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

<http://localhost:8080/getBooks/books?type=html>

Autor: Ralls, Kim

Título: Midnight Rain

Gênero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva

Título: Maeve Ascendant

Gênero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva

Título: Oberon's Legacy

Gênero: Fantasy



← → ⌂ ⓘ localhost:8080/getBooks/books?type=xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
    <description>A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.</description>
  </book>
  <book id="bk103">
    <author>Corets, Eva</author>
    <title>Maeve Ascendant</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-11-17</publish_date>
    <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
  </book>
  <book id="bk104">
    <author>Corets, Eva</author>
    <title>Oberon's Legacy</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-03-10</publish_date>
    <description>In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.</description>
  </book>
  <book id="bk105">
    <author>Corets, Eva</author>
    <title>The Sundered Grail</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-09-10</publish_date>
    <description>The two daughters of Maeve, half-sisters, battle one another for control of England. Sequel to Oberon's Legacy.</description>
  </book>
  <book id="bk106">
    <author>Randall, Cynthia</author>
    <title>Lover Birds</title>
    <genre>Romance</genre>
    <price>4.95</price>
    <publish_date>2000-09-02</publish_date>
    <description>When Carla meets Paul at an ornithology conference, tempers fly as feathers get ruffled.</description>
  </book>
  <book id="bk107">
    <author>Thurman, Paula</author>
    <title>Splish Splash</title>
    <genre>Romance</genre>
```

<http://localhost:8080/getBooks/books?type=xml>



← → ⌂

localhost:8080/getBooks/booksXpath

Resultados da pesquisa XPATH por:

price < 7.00 and price > 5.00

Autor: Ralls, Kim

Título: Midnight Rain

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva

Título: Maeve Ascendant

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva

Título: Oberon's Legacy

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2001-03-10

Descrição: In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.



IS_TP1_Python >

Nome

- .idea
- BD
- REST
- SOAP

> IS_TP1_Python > SOAP

Nome

- files
- python

IS_TP1_Python > SOAP > python

Nome

- Client.py
- SoapXmlService.py

IS_TP1_Python > SOAP > files

Nome

- books.xml



SoapXmlService.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
1  from spyne import Application, rpc, ServiceBase, Unicode
2  from spyne.protocol.soap import Soap11
3  from spyne.server.wsgi import WsgiApplication
4  from xml.dom import minidom
5  from lxml import etree
6
7
8  class SoapXmlService(ServiceBase):
9      @rpc(_returns=Unicode)
10     def books_parse(ctx):
11         xmldoc = minidom.parse('../files/books.xml')
12         itemlist = xmldoc.getElementsByTagName('book')
13         data = ""
14         i = 0
15         for i in range(0, len(itemlist)):
16             node = itemlist.item(i)
17             autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
18             titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
19             genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
20             preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
21             data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
22             descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
23             data = data + "<h5>Autor: " + autor + "</h5>"
24             data = data + "<h5>Título: " + titulo + "</h5>"
25             data = data + "<h5>Genero: " + genero + "</h5>"
26             data = data + "<h5>Preço: " + preco + "</h5>"
27             data = data + "<h5>Data de Publicação: " + data_publicacao + "</h5>"
28             data = data + "<h5>Descrição: " + descricao + "</h5>"
29             data = data + "<br>"
30
31     return data
```

SoapXmlService.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
31
32     @rpc(_returns=Unicode)
33     def books_Xpath(ctx):
34         root = etree.parse('../files/books.xml')
35         books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
36         data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"
37         i = 0
38         for i in range(0, len(books)):
39             autor = books[i][0].text
40             titulo = books[i][1].text
41             genero = books[i][2].text
42             preco = books[i][3].text
43             data_publicacao = books[i][4].text
44             descricao = books[i][5].text
45             data = data + "<h5>Autor: " + autor + "<h5>"
46             data = data + "<h5>Título: " + titulo + "<h5>"
47             data = data + "<h5>Genero: " + genero + "<h5>"
48             data = data + "<h5>Preço: " + preco + "<h5>"
49             data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
50             data = data + "<h5>Descrição: " + descricao + "<h5>"
51             data = data + "<br>"
52         return data
53
54
55     application = Application([SoapXmlService],
56                               tns='soap.books.parse',
57                               in_protocol=Soap11(validation='lxml'),
58                               out_protocol=Soap11()
59                               )
60
61     if __name__ == '__main__':
62         from wsgiref.simple_server import make_server
63
64         wsgi_app = WsgiApplication(application)
65         server = make_server('localhost', 8000, wsgi_app)
66         server.serve_forever()
```



Client.py X

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
1  from zeep import Client
2  from flask import Flask
3  app = Flask(__name__)
4
5
6  @app.route('/')
7  def index():
8      html = "<h3>Servidor Flask para teste do Web service em SOAP: </h3>"
9      html = html + "<a href='/soap/booksParse'>XML Parse - Ficheiro books.xml</a><br>"
10     html = html + "<a href='/soap/Xpath'>XPATH - Ficheiro books.xml</a>"
11     return html
12
13 @app.route('/soap/booksParse', methods=['GET'])
14 def booksParseSOAP():
15     client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
16     data = client.service.books_parse()
17     return data
18
19 @app.route('/soap/Xpath', methods=['GET'])
20 def booksXpath():
21     client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
22     data = client.service.books_Xpath()
23     return data
24
25 if __name__ == "__main__":
26     app.run(host='localhost', port=8080)
27
```



IS_TP1_Python > SOAP > python > SoapXmlService.py

Project

- IS_TP1_Python [Python] C:\testes\IS_TP1_Python
 - BD
 - REST
 - files books.xml
 - python
 - Books_ws_rest.py
 - venv library root
 - SOAP
 - files
 - python
 - Client.py
 - SoapXmlService.py
 - External Libraries
 - Scratches and Console

Run: Books_ws_rest x

Structure

Favorites

File "C:\\"
from sp
ModuleNotFo
Process fin

1 from spyne import Application, rpc, ServiceBase, Unicode
2 from spyne.protocol.soap import Soap11
3 from spyne.server.wsgi import WsgiApplication
4 from xml.dom import minidom
5 from xml import etree

New

Cut Ctrl+X

Copy Ctrl+C

Copy Path/Reference...

Paste Ctrl+V

Find Usages Alt+F7

Inspect Code...

Refactor

Add to Favorites

Reformat Code Ctrl+Alt+L

Optimize Imports Ctrl+Alt+O

Delete...

Delete

Override File Type

Run 'SoapXmlService' Ctrl+Shift+F10

Debug 'SoapXmlService'

More Run/Debug

Open in Right Split Shift+Enter

Open In >

```
SoapXmlService(ServiceBase):  
    rpc(_returns=Unicode)  
    def books_parse(ctx):  
        xmldoc = minidom.parse('../files/books.xml')  
        itemlist = xmldoc.getElementsByTagName('book')  
        data = ""
```

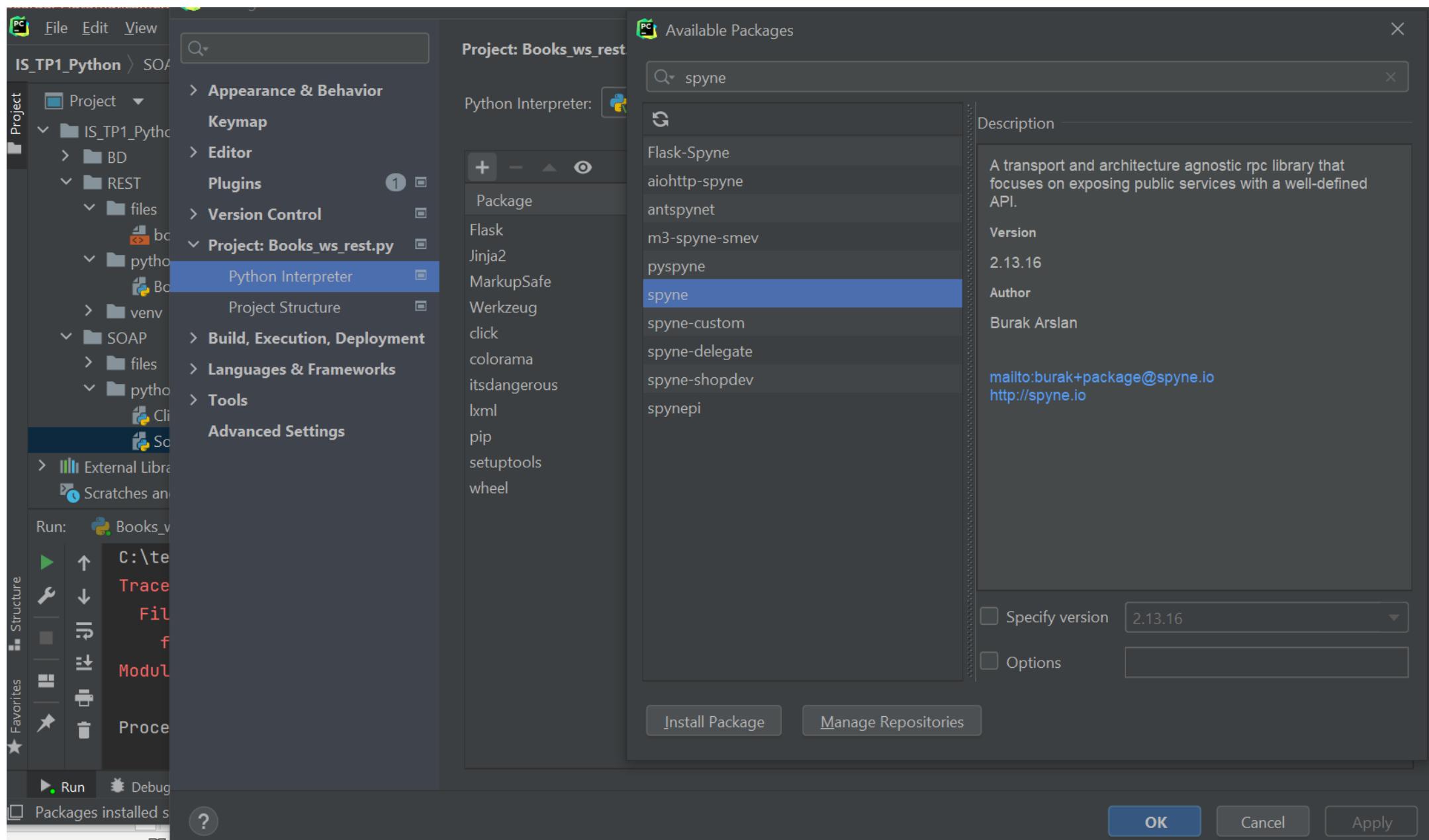
thon.exe C:/testes/IS_TP1_Python/SOAP/python/SoapXmlService.py", line 1, in <module>
iceBase, Unicode



The screenshot shows a Python development environment with the following details:

- Run:** Books_ws_rest × SoapXmlService ×
- Path:** C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/SOAP/python/SoapXmlService.py
- Traceback (most recent call last):**
- File:** C:\testes\IS_TP1_Python\SOAP\python\SoapXmlService.py, line 1, in <module>
- Code:** from spyne import Application, rpc, ServiceBase, Unicode
- Error:** ModuleNotFoundError: No module named 'spyne'
- Message:** Process finished with exit code 1

The interface includes a sidebar with icons for Run, Structure, Favorites, and a toolbar with Run, Debug, TODO, Problems, Terminal, Python Packages, and Python Console.





The screenshot shows a code editor interface with two files listed in the sidebar: Client.py and SoapXmlService.py. The Client.py file is currently open and contains the following code:

```
Client.py
SoapXmlService.py
External Libraries
Scratches and Consoles

Run: Books_ws_rest × SoapXmlService × Client ×

c:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe c:/testes/IS_TP1_Python/SOAP/python/Client.py

Traceback (most recent call last):
  File "C:\testes\IS_TP1_Python\SOAP\python\Client.py", line 1, in <module>
    from zeep import Client
ModuleNotFoundError: No module named 'zeep'

Process finished with exit code 1
```

The code editor has a status bar at the bottom showing the command: `c:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe c:/testes/IS_TP1_Python/SOAP/python/Client.py`. The output pane shows a traceback indicating a `ModuleNotFoundError` for the `zeep` module.



Available Packages

Search: zeep

- fuzeepass
- netas-zeep
- onvif-zeep
- onvif-zeep-async
- onvif-zeep-roboticia
- onvif2-zeep
- python-zeep
- robotframework-zeep
- robotframework-zeelibrary
- zeep**
- zeep-adv
- zeep-bold
- zeep-roboticia
- zeep.sms
- zeeppoptesting

Description

A modern/fast Python SOAP client based on lxml / requests

Version

4.1.0

Author

Michael van Tellingen

<mailto:michaelvantellingen@gmail.com>
<http://docs.python-zeep.org>

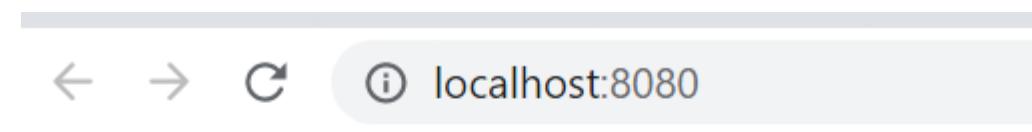
Specify version 4.1.0

Options

Install Package Manage Repositories



```
Run: Books_ws_rest × SoapXmlService × Client ×
C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/SOAP/python/Client.py
* Serving Flask app 'Client' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://localhost:8080/ (Press CTRL+C to quit)
```



Servidor Flask para teste do Web service em SOAP:

[XML Parse - Ficheiro books.xml](#)
[XPATH - Ficheiro books.xml](#)



← → ⌂ i localhost:8080/soap/booksParse

Autor: Gambardella, Matthew

Titulo: XML Developer's Guide

Genero: Computer

Preço: 44.95

Data de Publicação: 2000-10-01

Descrição: An in-depth look at creating applications with XML.

Autor: Ralls, Kim

Titulo: Midnight Rain

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva

Titulo: Maeve Ascendant

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva

Titulo: Oberon's Legacy

Genero: Fantasy

i DevTools is now available in Portuguese!

Always match Chrome's language Switch DevTools to Portuguese Don't show again

The screenshot shows the Network tab of the Chrome DevTools interface. A single request named 'booksParse' is listed. The request URL is `http://localhost:8080/soap/booksParse`, the method is GET, and the status code is 200 OK. The response content length is 3511 bytes, and the content type is text/html; charset=utf-8. The Date header indicates the request was made on Monday, November 15, 2021, at 19:23:19 GMT. The Server header shows Werkzeug/2.0.2 Python/3.9.5. The request headers include Accept: text/html, application/xhtml+xml, application/xml.

Name	Headers	Preview	Response	Initiator	Timing
booksParse					
▼ General					
Request URL: <code>http://localhost:8080/soap/booksParse</code>					
Request Method: GET					
Status Code: 200 OK					
Remote Address: 127.0.0.1:8080					
Referrer Policy: strict-origin-when-cross-origin					
▼ Response Headers View source					
Content-Length: 3511					
Content-Type: text/html; charset=utf-8					
Date: Mon, 15 Nov 2021 19:23:19 GMT					
Server: Werkzeug/2.0.2 Python/3.9.5					
▼ Request Headers View source					
Accept: text/html, application/xhtml+xml, application/xml					



Resultados da pesquisa XPATH por:

price < 7.00 and price > 5.00

Autor: Ralls, Kim

Título: Midnight Rain

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-12-16

Descrição: A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.

Autor: Corets, Eva

Título: Maeve Ascendant

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2000-11-17

Descrição: After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.

Autor: Corets, Eva

Título: Oberon's Legacy

Genero: Fantasy

Preço: 5.95

Data de Publicação: 2001-03-10

Descrição: In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.



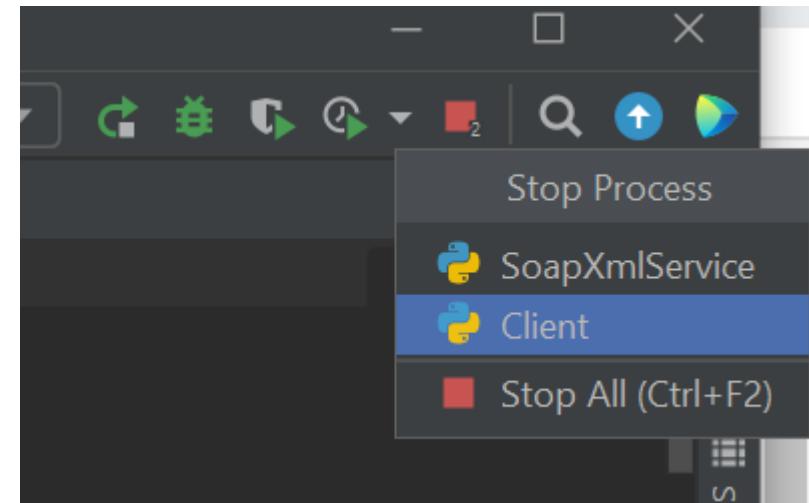
Instituto Politécnico
de Viana do Castelo

ipvc Instituto Politécnico
de Viana do Castelo



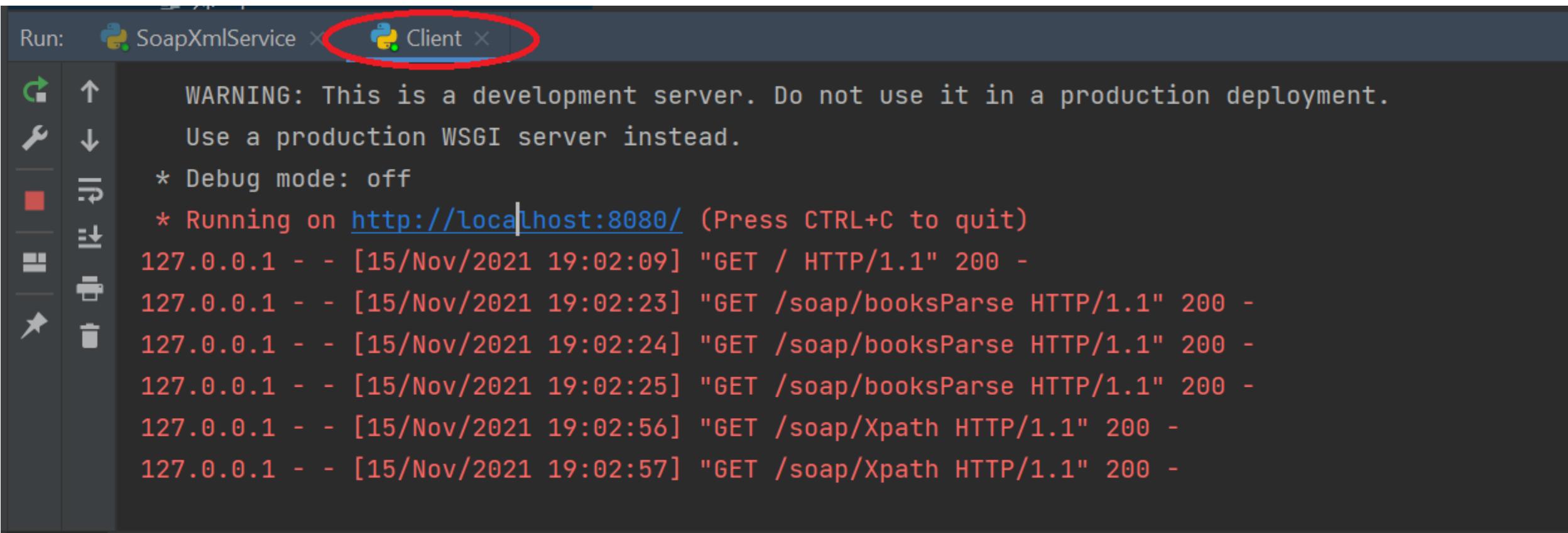
Escola Superior
de Tecnologia e Gestão

Versão 1 – Documento em Atualização



Run: SoapXmlService Client

```
C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/SOAP/python/SoapXmlService.py
127.0.0.1 - - [15/Nov/2021 19:02:21] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [15/Nov/2021 19:02:22] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [15/Nov/2021 19:02:23] "POST /soap.books.parse HTTP/1.1" 200 4747
127.0.0.1 - - [15/Nov/2021 19:02:23] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [15/Nov/2021 19:02:24] "POST /soap.books.parse HTTP/1.1" 200 4747
127.0.0.1 - - [15/Nov/2021 19:02:25] "POST /soap.books.parse HTTP/1.1" 200 4747
127.0.0.1 - - [15/Nov/2021 19:02:54] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [15/Nov/2021 19:02:55] "GET /soap.books.parse?wsdl HTTP/1.1" 200 3517
127.0.0.1 - - [15/Nov/2021 19:02:56] "POST /soap.books.parse HTTP/1.1" 200 2340
127.0.0.1 - - [15/Nov/2021 19:02:57] "POST /soap.books.parse HTTP/1.1" 200 2340
```



Run:  SoapXmlService ×  Client ×

WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.

* Debug mode: off

* Running on <http://localhost:8080/> (Press CTRL+C to quit)

```
127.0.0.1 - - [15/Nov/2021 19:02:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2021 19:02:23] "GET /soap/booksParse HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2021 19:02:24] "GET /soap/booksParse HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2021 19:02:25] "GET /soap/booksParse HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2021 19:02:56] "GET /soap/Xpath HTTP/1.1" 200 -
127.0.0.1 - - [15/Nov/2021 19:02:57] "GET /soap/Xpath HTTP/1.1" 200 -
```



← → ⌂ chrome.google.com/webstore/detail/soap-analyzer/pkolclcpcbkghocbmacdmcheaioncnl



chrome web store



Iniciar sessão

Página Inicial > Extensões > SOAP Analyzer



SOAP Analyzer

Disponibilizado por: DJ Collier

Instalar



21 | Ferramentas do Programador



4 000+ utilizadores

Vista geral

Práticas de Privacidade

Críticas

Suporte

Relacionados

```
1. <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
2.   <SOAP:Header xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
3.     <header xmlns="http://schemas.cordys.com/General/1.0/">
4.       <Logger xmlns="http://schemas.cordys.com/General/1.0/">
5.         <DC xmlns="http://schemas.cordys.com/General/1.0/" name="XForms">/testtool/testtool.caf</DC>
6.         <DC xmlns="http://schemas.cordys.com/General/1.0/" name="hopCount">0</DC>
7.         <DC xmlns="http://schemas.cordys.com/General/1.0/" name="correlationID">525400b3-1fd9-11</DC>
8.       </Logger>
9.     </header>
10.    <i18n:international xmlns:i18n="http://www.w3.org/2005/09/ws-i18n">
11.      <locale xmlns="http://www.w3.org/2005/09/ws-i18n">en-US</locale>
```

<https://www.soapui.org/docs/soap-and-wsdl/working-with-wsdl/>

The screenshot shows a web browser displaying the SoapUI documentation at <https://www.soapui.org/docs/soap-and-wsdl/working-with-wsdl/>. The page title is "Work with WSDLs in SoapUI" under the heading "Create Project From WSDL". A sidebar on the left lists various topics related to SoapUI, including "Working with WSDL Coverage", "HTTP Headers", "Working with WSDLs", "Attachments and Files", "Operations and Requests", and "Tips & Tricks". The main content area contains instructions for creating a project from a WSDL file, followed by a screenshot of the "New SOAP Project" dialog box. The dialog box shows the "calculator" project name, the initial WSDL URL (<http://www.dneonline.com/calculator.asmx?wsdl>), and several configuration options like "Create Requests" (checked) and "Create TestSuite".

How can we help? Clear

Tools ▼ Learn Docs ▼ Resources

Validate the WSDL against the WS-I Basic Profile

Generating Code for your WSDL

Work with WSDLs in SoapUI

Create Project From WSDL

To take a closer look at a WSDL file, create a new project and import a sample WSDL file:

1. In SoapUI, click or select **File > New SOAP Project**

New SOAP Project

New SOAP Project
Creates a WSDL/SOAP based Project in this workspace

Project Name:

Initial WSDL:

Create Requests: Create sample requests for all operations?

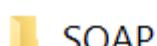
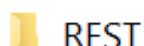
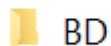
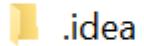
Create TestSuite: Creates a TestSuite for the imported WSDL

Relative Paths: Stores all file paths in project relatively to project file (requires save)



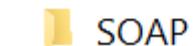
IS_TP1_Python >

Nome



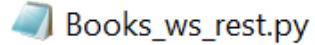
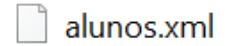
> IS_TP1_Python > BD >

] Nome



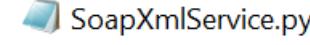
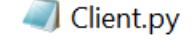
> IS_TP1_Python > BD > REST

Nome



IS_TP1_Python > BD > SOAP

Nome





IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

The screenshot shows the phpMyAdmin interface for the 'is' database. The left sidebar shows the database structure with 'alunos' as the selected table. The main panel displays the table structure:

#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra	Acções
1	id	int(11)			Não	Nenhum			Muda Elimina Mais
2	nome	varchar(50)	utf8mb4_general_ci		Não	Nenhum			Muda Elimina Mais

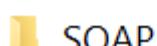
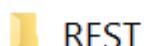
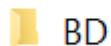
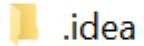
Below the table structure, there are several buttons and input fields for managing the table:

- Procurar, Estrutura, SQL, Pesquisar, Insere, Exportar, Importar, Privilégios, Operações, Rastreando, Açãoadores
- Novo, information_schema, Nova, MySQL, performance_schema, phpmyadmin, test
- Novo, id (int), nome (varchar)
- Procurar, Muda, Elimina, Primária, Único, Índice, Texto Completo, Adicionar à(s) coluna(s) centrais, Remover da(s) coluna(s) central(is)
- Imprimir, Propor uma estrutura de tabela, Acompanhar tabela, Mover campo(s), Normalizar
- Adicionar, 1 campo(s), após nome, Executar
- Índices: Nenhum índice definido!
- Criar um índice nas colunas, 1 Executar
- Partições: Nenhum particionamento definido!



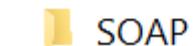
IS_TP1_Python >

Nome



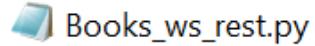
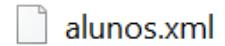
> IS_TP1_Python > BD >

] Nome



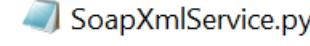
> IS_TP1_Python > BD > REST

Nome



IS_TP1_Python > BD > SOAP

Nome





Books_ws_rest.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
1  from xml.dom import minidom
2  from flask import Flask, request
3  from werkzeug.wrappers import Response
4  import mysql.connector
5  app = Flask(__name__)
6
7  mydb = mysql.connector.connect(
8      host="localhost",
9      user="root",
10     password="",
11     database="is"
12 )
13
14 @app.route('/')
15 def index():
16     html = "<h3>Web Service REST com acesso à base de dados MySQL: </h3>"
17     html = html + "<a href='/getAlunos/alunos?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
18     return html
19
20 @app.route('/getAlunos/alunos', methods=['GET'])
21 def getBooks():
22     selectType = request.args.get('type')
23     if selectType.upper() == "ALL":
24         mycursor = mydb.cursor()
25         mycursor.execute("SELECT * FROM alunos")
26         myresult = mycursor.fetchall()
27         doc = minidom.Document()
28         tag_alunos = doc.createElement('alunos')
29         for aluno in myresult:
30             id_aluno = str(aluno[0])
31             nome = aluno[1]
32             tag_aluno = doc.createElement('aluno')
33
34             tag_id = doc.createElement('id')
35             content_id = doc.createTextNode(id_aluno)
36             tag_id.appendChild(content_id)
37
```



Books_ws_rest.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
1  from xml.dom import minidom
2  from flask import Flask, request
3  from werkzeug.wrappers import Response
4  import mysql.connector
5  app = Flask(__name__)
6
7  mydb = mysql.connector.connect(
8      host="localhost",
9      user="root",
10     password="",
11     database="is"
12 )
13
14 @app.route('/')
15 def index():
16     html = "<h3>Web Service REST com acesso à base de dados MySQL: </h3>"
17     html = html + "<a href='/getAlunos/alunos?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
18     return html
19
20 @app.route('/getAlunos/alunos', methods=['GET'])
21 def getBooks():
22     selectType = request.args.get('type')
23     if selectType.upper() == "ALL":
24         mycursor = mydb.cursor()
25         mycursor.execute("SELECT * FROM alunos")
26         myresult = mycursor.fetchall()
27         doc = minidom.Document()
28         tag_alunos = doc.createElement('alunos')
29         for aluno in myresult:
30             id_aluno = str(aluno[0])
31             nome = aluno[1]
32             tag_aluno = doc.createElement('aluno')
33
34             tag_id = doc.createElement('id')
35             content_id = doc.createTextNode(id_aluno)
36             tag_id.appendChild(content_id)
37
```



IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
22     selectType = request.args.get('type')
23     if selectType.upper() == "ALL":
24         mycursor = mydb.cursor()
25         mycursor.execute("SELECT * FROM alunos")
26         myresult = mycursor.fetchall()
27         doc = minidom.Document()
28         tag_alunos = doc.createElement('alunos')
29         for aluno in myresult:
30             id_aluno = str(aluno[0])
31             nome = aluno[1]
32             tag_aluno = doc.createElement('aluno')
33
34             tag_id = doc.createElement('id')
35             content_id = doc.createTextNode(id_aluno)
36             tag_id.appendChild(content_id)
37
38             tag_nome = doc.createElement('nome')
39             content_nome = doc.createTextNode(nome)
40             tag_nome.appendChild(content_nome)
41
42             tag_aluno.appendChild(tag_id)
43             tag_aluno.appendChild(tag_nome)
44
45             tag_alunos.appendChild(tag_aluno)
46
47             doc.appendChild(tag_alunos)
48             string_xml = doc.toprettyxml(indent="\t")
49             caminho_ficheiro = "alunos.xml"
50             with open(caminho_ficheiro, "w") as f:
51                 f.write(string_xml)
52             return Response(string_xml, mimetype='text/xml')
53
54     if __name__ == "__main__":
55         app.run(host='localhost', port=8080)
```



IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<alunos>
    <aluno>
        <id>1</id>
        <nome>Antonio Silva</nome>
    </aluno>
    <aluno>
        <id>2</id>
        <nome>Maria Albertina</nome>
    </aluno>
    <aluno>
        <id>3</id>
        <nome>Jose Ferreira</nome>
    </aluno>
    <aluno>
        <id>4</id>
        <nome>Pedro Costa</nome>
    </aluno>
</alunos>
```

The screenshot shows a browser window displaying an XML response from the URL `localhost:8080/getAlunos/alunos?type=all`. The XML structure lists four students with their IDs and names. Below the browser is a screenshot of the Network tab in a developer tools interface, specifically the Network tab of Chrome DevTools. It shows a single request to the same URL, with a status of 200 OK. The Headers section shows standard HTTP headers like Content-Type and Date. The Response section shows the raw XML content of the response. The bottom of the developer tools shows a summary with 2 requests and 439 B transferred.



IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

IS_TP1_Python > BD > SOAP

Nome

 alunos.xml

 Client.py

 SoapXmlService.py

SoapXmlService.py

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
1  from spyne import Application, rpc, ServiceBase, Unicode
2  from spyne.protocol.soap import Soap11
3  from spyne.server.wsgi import WsgiApplication
4  from xml.dom import minidom
5  import mysql.connector
6
7
8
9  mydb = mysql.connector.connect(
10    host="localhost",
11    user="root",
12    password="",
13    database="is"
14  )
15
16 class SoapXmlService(ServiceBase):
17
18     @rpc(_returns=Unicode)
19     def alunos_bd(ctx):
20         mycursor = mydb.cursor()
21         mycursor.execute("SELECT * FROM alunos")
22         myresult = mycursor.fetchall()
23         doc = minidom.Document()
24         tag_alunos = doc.createElement('alunos')
25         for aluno in myresult:
26             id_aluno = str(aluno[0])
27             nome = aluno[1]
28             tag_aluno = doc.createElement('aluno')
29
30             tag_id = doc.createElement('id')
31             content_id = doc.createTextNode(id_aluno)
32             tag_id.appendChild(content_id)
33
34             tag_nome = doc.createElement('nome')
35             content_nome = doc.createTextNode(nome)
36             tag_nome.appendChild(content_nome)
37
```

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
16 class SoapXmlService(ServiceBase):
17
18     @rpc(_returns=Unicode)
19     def alunos_bd(ctx):
20         mycursor = mydb.cursor()
21         mycursor.execute("SELECT * FROM alunos")
22         myresult = mycursor.fetchall()
23         doc = minidom.Document()
24         tag_alunos = doc.createElement('alunos')
25         for aluno in myresult:
26             id_aluno = str(aluno[0])
27             nome = aluno[1]
28             tag_aluno = doc.createElement('aluno')
29
30             tag_id = doc.createElement('id')
31             content_id = doc.createTextNode(id_aluno)
32             tag_id.appendChild(content_id)
33
34             tag_nome = doc.createElement('nome')
35             content_nome = doc.createTextNode(nome)
36             tag_nome.appendChild(content_nome)
37
38             tag_aluno.appendChild(tag_id)
39             tag_aluno.appendChild(tag_nome)
40
41             tag_alunos.appendChild(tag_aluno)
42
43             doc.appendChild(tag_alunos)
44             string_xml = doc.toprettyxml(indent="\t")
45             caminho_ficheiro = "alunos.xml"
46             with open(caminho_ficheiro, "w") as f:
47                 f.write(string_xml)
48
49
50             return string_xml
```

IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
SoapXmlService.py x
43     doc.appendChild(tag_alunos)
44     string_xml = doc.toprettyxml(indent="\t")
45     caminho_ficheiro = "alunos.xml"
46     with open(caminho_ficheiro, "w") as f:
47         f.write(string_xml)
48
49     return string_xml
50
51
52     application = Application([SoapXmlService],
53                               tns='soap.alunos.db',
54                               in_protocol=Soap11(validation='lxml'),
55                               out_protocol=Soap11()
56                               )
57
58 if __name__ == '__main__':
59     from wsgiref.simple_server import make_server
60
61     wsgi_app = WsgiApplication(application)
62     server = make_server('localhost', 8000, wsgi_app)
63     server.serve_forever()
```



IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

```
SoapXmlService.py Client.py
1  from zeep import Client
2  from flask import Flask
3  from werkzeug.wrappers import Response
4  app = Flask(__name__)
5
6
7  @app.route('/')
8  def index():
9      html = "<h3>Métodos disponíveis: </h3>"
10     html = html + "<a href='/soap/alunosXmlBd'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
11     return html
12
13 @app.route('/soap/alunosXmlBd', methods=['GET'])
14 def alunosXmlBd():
15     client = Client(wsdl='http://localhost:8000/soap.alunos.db?wsdl')
16     data = client.service.alunos_bd()
17     return Response(data, mimetype='text/xml')
18
19 if __name__ == "__main__":
20     app.run(host='localhost', port=8080)
21
```



IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<alunos>
    <aluno>
        <id>1</id>
        <nome>Antonio Silva</nome>
    </aluno>
    <aluno>
        <id>2</id>
        <nome>Maria Albertina</nome>
    </aluno>
    <aluno>
        <id>3</id>
        <nome>Jose Ferreira</nome>
    </aluno>
    <aluno>
        <id>4</id>
        <nome>Pedro Costa</nome>
    </aluno>
</alunos>
```

The screenshot shows a browser window with the URL `localhost:8080/soap/alunosXmlBd`. The page content displays an XML document structure representing student data. To the right of the browser, the developer tools Network tab is open, showing a single request for the same URL. The request details show a successful `GET` request with status code `200 OK`, a content length of 286 bytes, and a content type of `text/xml; charset=utf-8`. The response body is identical to the XML shown in the browser. The developer tools also show the request headers and a list of 2 requests transferred.



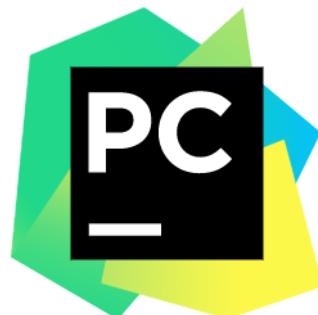
jetbrains.com/pycharm/download/#section=windows



Developer Tools Team Tools Learning Tools Solutions Store

PyCharm

Coming in 2021.3 What's New Features Learn Bu



Version: 2021.2.3
Build: 212.5457.59
20 October 2021

[System requirements](#)

[Installation Instructions](#)

Download PyCharm

[Windows](#) [macOS](#) [Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free trial

Community

For pure Python development

[Download](#)

Free, built on open-source



code=mNwQbjEO22dyCvygXhYZ-IH-DlIQw0QUuyyELjij9rQ.M5LFDhNHvMO0mXjqHDk

JET
BRAINS

License Activation

PyCharm Activate

Plugins

Code With Me Activate to enable

Authorization Successful

Now you can close this page and return to your IDE.

Get license from:

JB Account Activation code License server

No licenses found.

[Activate](#) [Refresh license list](#)

Jorge Ribeiro

[Continue](#) [Exit](#)



License Activation

— X

PyCharm
Active until dez 14, 2021

Plugins

Code With Me
Active until dez 14, 2021

Licensed to Jorge Ribeiro
Subscription is active until December 14, 2021.
Evaluation purpose only

Activate New License Remove License

Jorge Ribeiro

Continue Exit



The screenshot shows a Python code editor interface with the following details:

- Project:** Books_ws_rest.py
- Code Content:**

```
1  from xml.dom import minidom
2  from flask import Flask, request
3  from lxml import etree
4  from werkzeug.wrappers import Response
5  app = Flask(__name__)
6
7
8  @app.route('/')
9  def index():
10     html = "<h3>Esta aplicacão contém os seguintes métodos: </h3>"
11     html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
12     html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
13     return html
14
15  @app.route('/getBooks/books', methods=['GET'])
16  def getBooks():
17      typeRequest = request.args.get('type')
18      if typeRequest.upper() == "HTML":
19          xmldoc = minidom.parse('../files/books.xml')
20          itemlist = xmldoc.getElementsByTagName('book')
21          data = ""
22          i = 0
```
- Editor Features:** The code is syntax-highlighted in orange, green, and blue. The interface includes a Project sidebar, a Structure sidebar, and a Favorites sidebar. The bottom navigation bar includes links for TODO, Problems, Terminal, Python Packages, and Python Console.
- Toolbars and Status:** The top toolbar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and a status message Books_ws_rest.py - Books_ws_rest.py. The bottom status bar shows file encoding (CRLF), character set (UTF-8), spaces (4 spaces), and a note about no interpreter.



Instituto Politécnico
de Viana do Castelo

ipvc Instituto Politécnico
de Viana do Castelo



Escola Superior
de Tecnologia e Gestão

Versão 1 – Documento em Atualização



IS_TP1_Python > REST > python > Books_ws_rest.py

Client (2)

Books_ws_rest.py

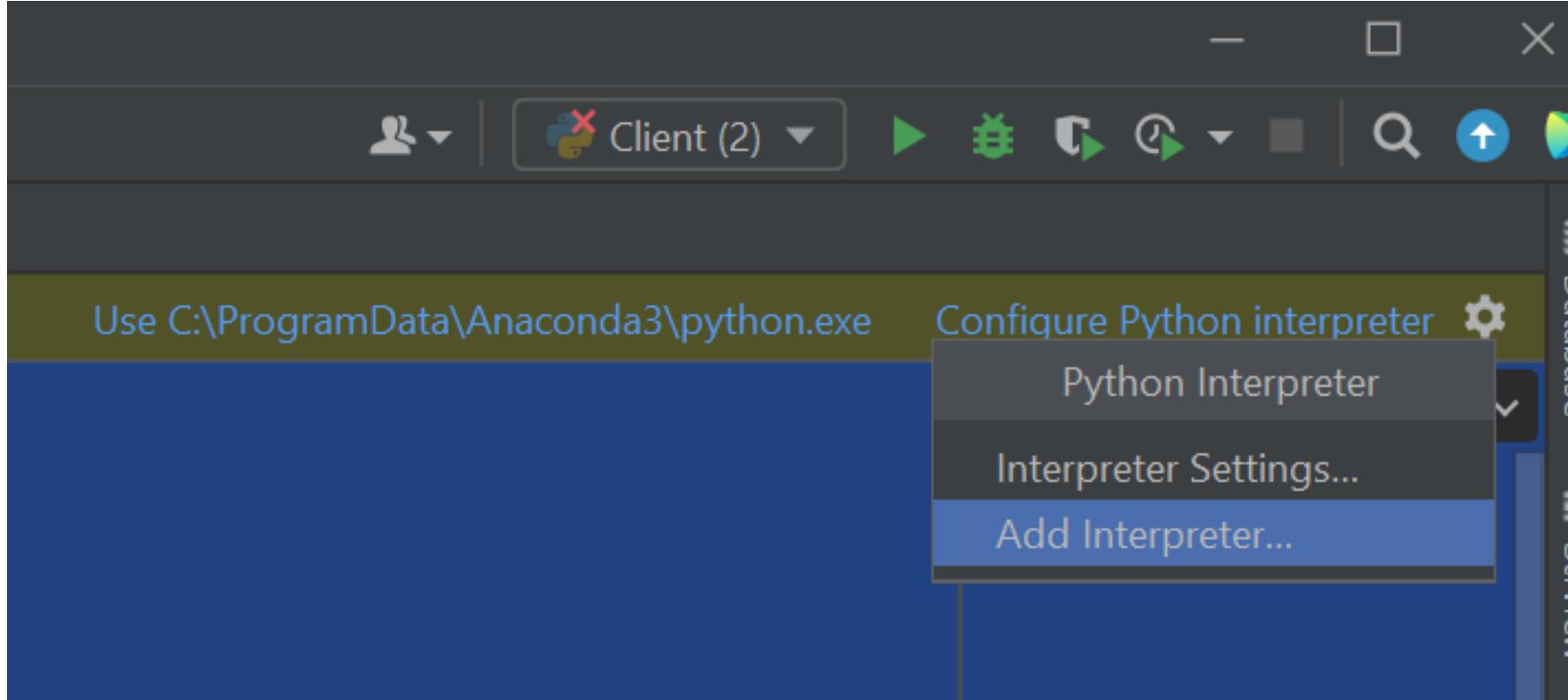
No Python interpreter configured for the project

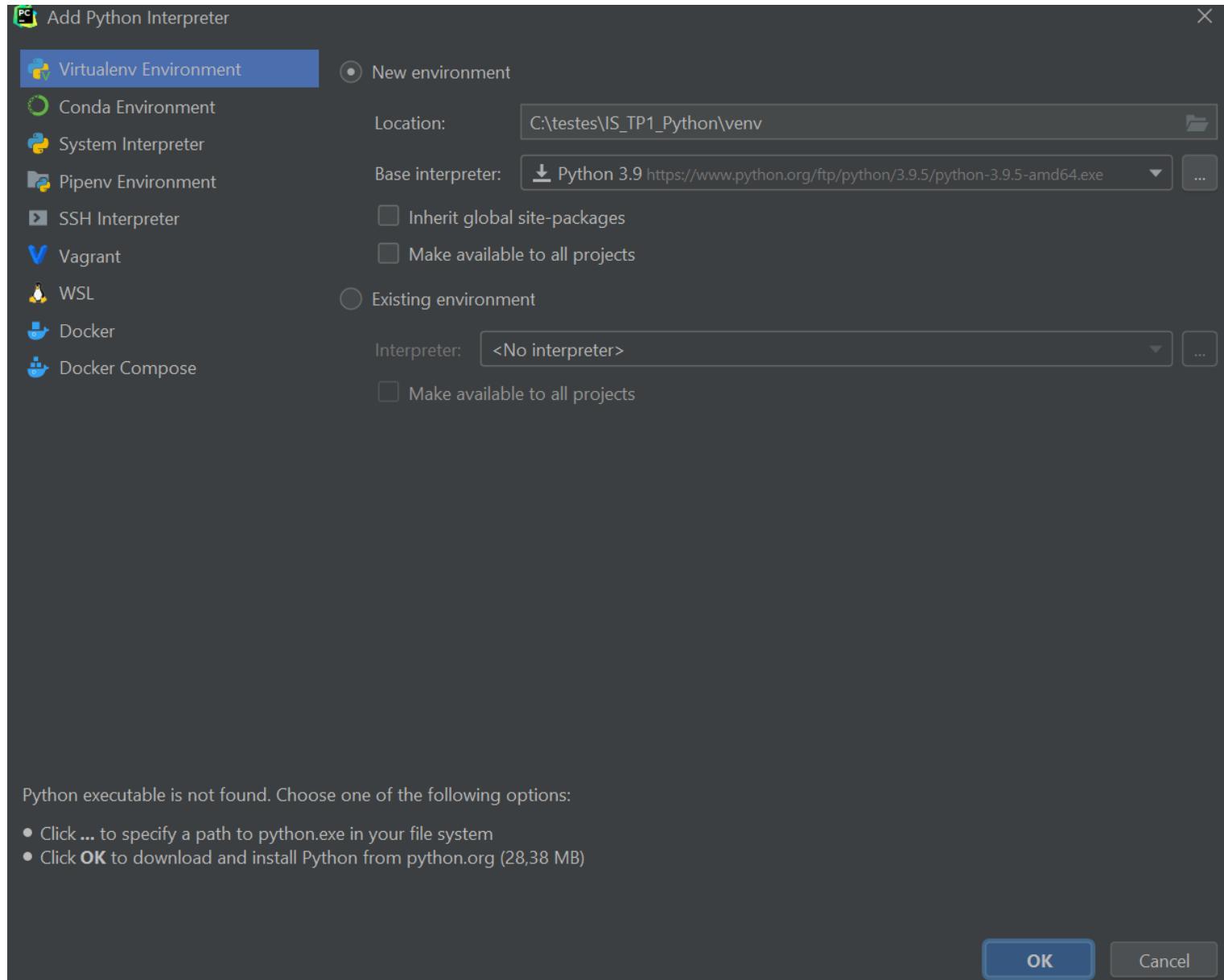
Use C:\ProgramData\Anaconda3\python.exe

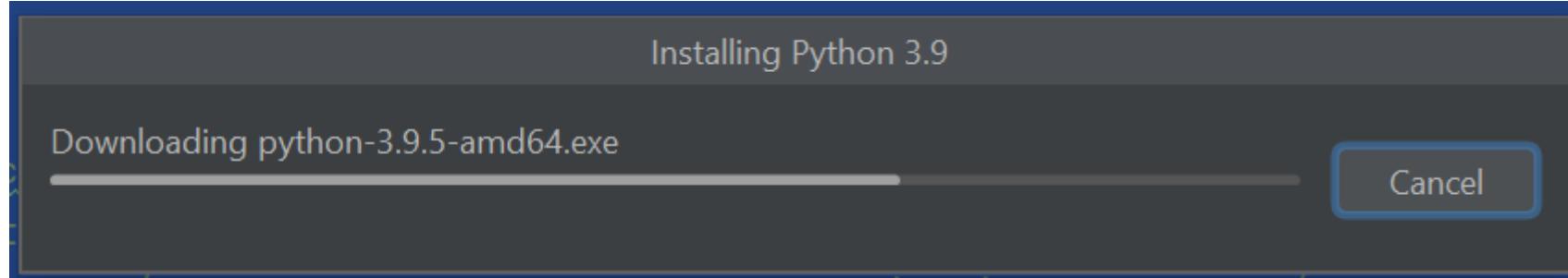
Configure Python interpre

A 7 ✓ 3

```
1  from xml.dom import minidom
2  from flask import Flask, request
3  from lxml import etree
4  from werkzeug.wrappers import Response
5  app = Flask(__name__)
6
7
8  @app.route('/')
9  def index():
10     html = "<h3>Esta aplicacão contém os seguintes métodos: </h3>"
11     html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
12     html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
13     return html
14
15  @app.route('/getBooks/books', methods=['GET'])
16  def getBooks():
17      typeRequest = request.args.get('type')
18      if typeRequest.upper() == "HTML":
19          xmldoc = minidom.parse('../files/books.xml')
20          itemlist = xmldoc.getElementsByTagName('book')
21          data = ""
```









Bibliografia e Referências:

[1]. TutorialsPoint . 2020. Java XML - Parsers. Available at:

https://www.tutorialspoint.com/java_xml/java_xml_parsers.htm;

[2]. PhpMyAdmin Contributors. 2020. Documentation. Available at:

<https://www.phpmyadmin.net/docs/>;

[3]. W3Schools . 2020. XML Validator. Available at:

https://www.w3schools.com/xml/xml_validator.asp;

[4]. Arskom Ltd. 2020. What is Spyne?. Available at:

<http://spyne.io/#inprot=Soap11&outprot= XmlDocument&s=rpc&tpt=WsgiApplication&validator=true>;

[5]. LinuxConfig.org . 2020. Deployment of an Example Application into Apache Tomcat Container. Available at:

<https://linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container>;



Bibliografia e Referências:

[6]. Python Software Foundation. 2020. Flask-Spyne 0.3.1. Available at:

<https://pypi.org/project/Flask-Spyne/>;

[7]. Onejohi. 2020. Building a simple REST API with NodeJS and Express. Available at:

<https://medium.com/@onejohi/building-a-simple-rest-api-with-nodejs-and-express-da6273ed7ca9>;

[8]. Filestack. 2020. Common problems with large file uploads. Available at:

<https://blog.filestack.com/thoughts-and-knowledge/common-problems-with-large-file-uploads/>;

[9]. Filestack. 2019. How To Upload Large Files. Available at:

<https://blog.filestack.com/thoughts-and-knowledge/how-to-upload-large-files/>;

[10]. James Kleeh. 2019. Code sample: Async parsing. Available at:

<https://github.com/FasterXML/aalto-xml/wiki/Code-sample:-Async-parsing>;

[11]. The PHP Group. 2020. XMLReader. Available at:

<https://www.php.net/manual/en/book.xmlreader.php>;