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.

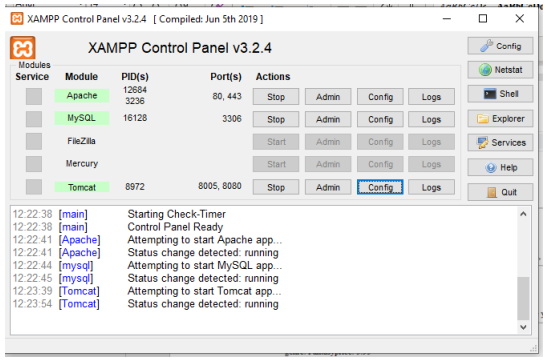
1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

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

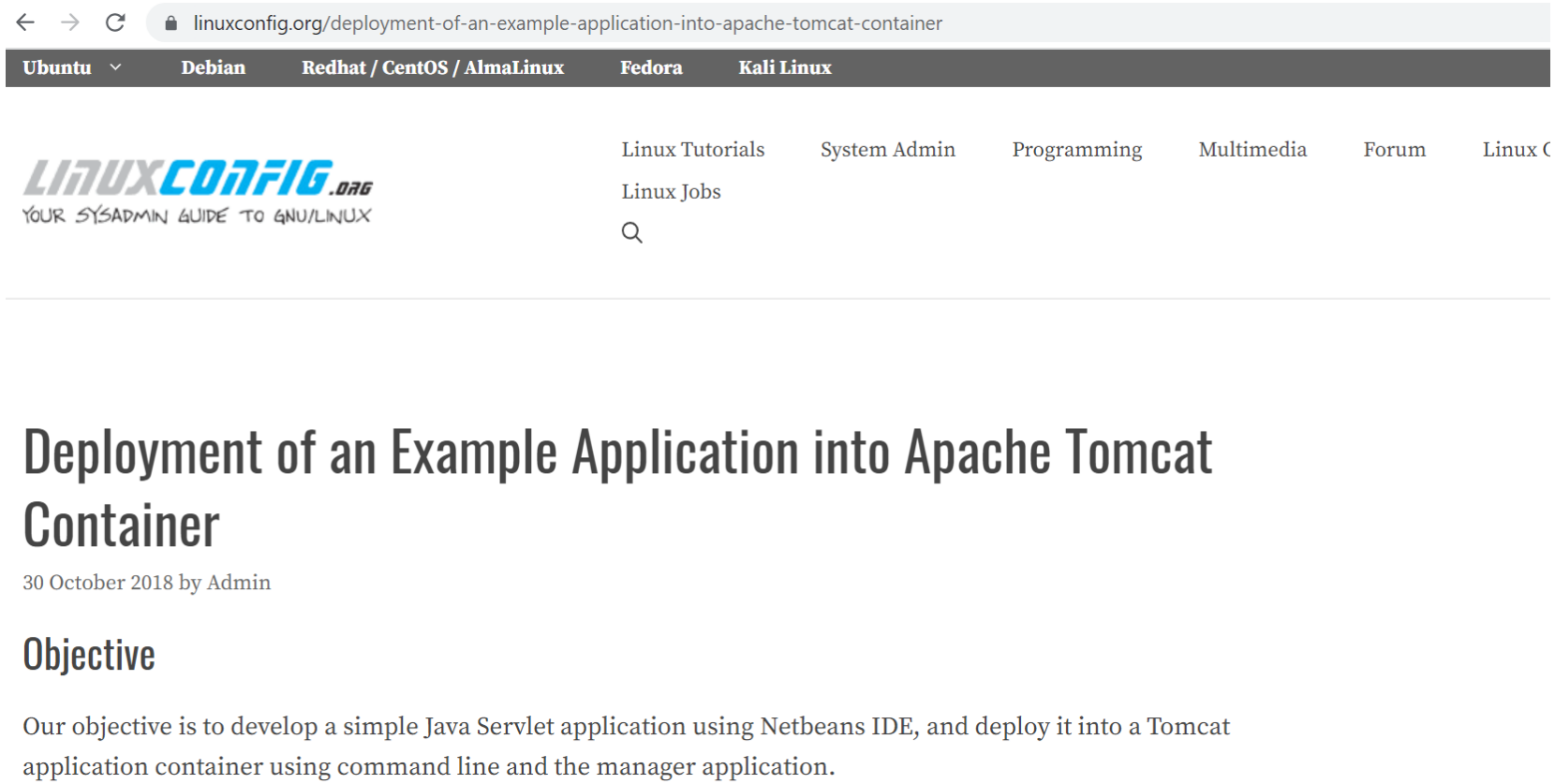
Com a realização desta tarefa passará a ser possível executar os projetos do tutorial 8, em JAVA e em PHP diretamente na plataforma XAMPP (sendo que na particularidade do Java, a parte de desenvolvimento estará sempre protegida a nível do código fonte e no servidor Tomcat apenas as classes):



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:

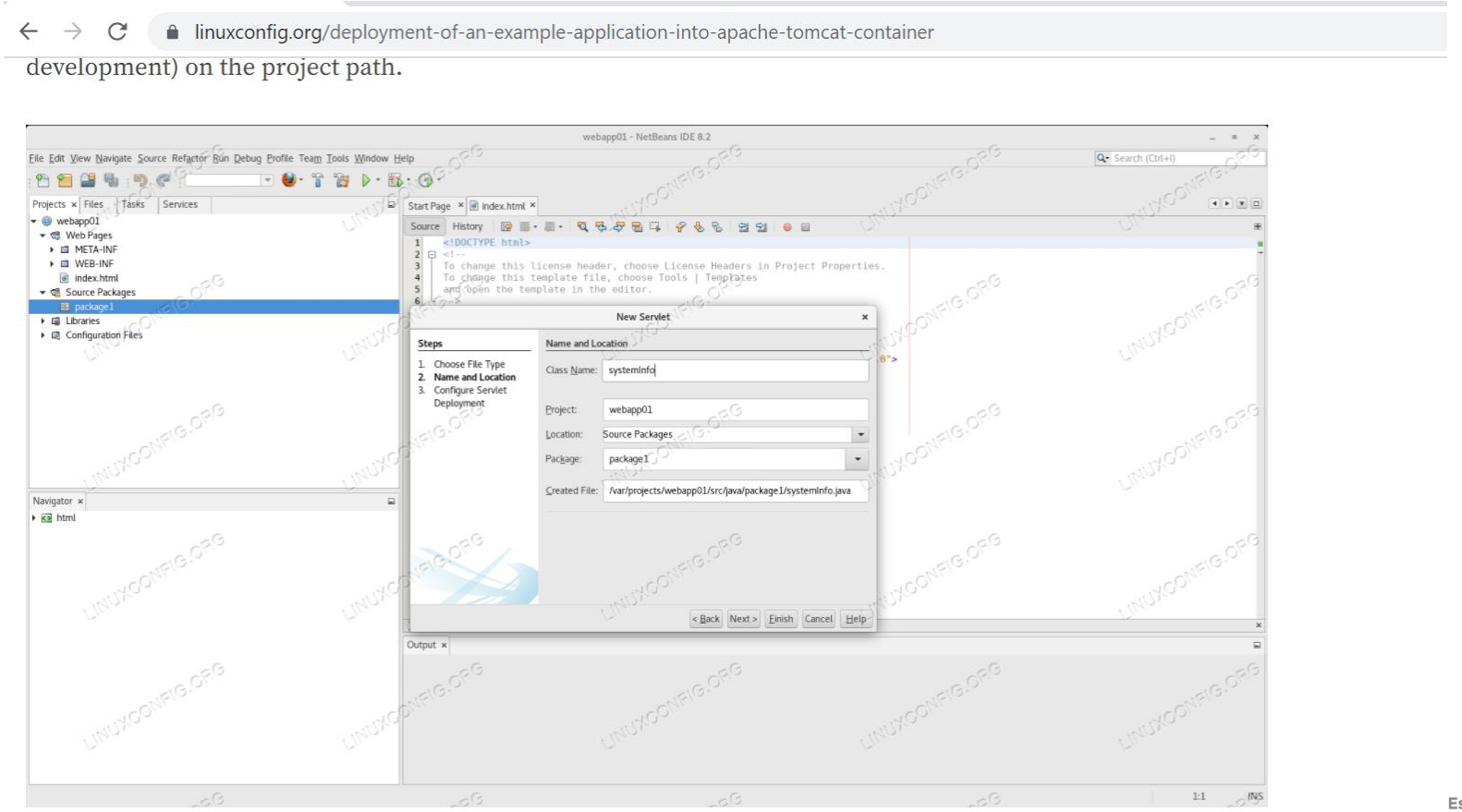
https://linuxconfig.org/deployment-of-an-example-application-into-apache-tomcat-container



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

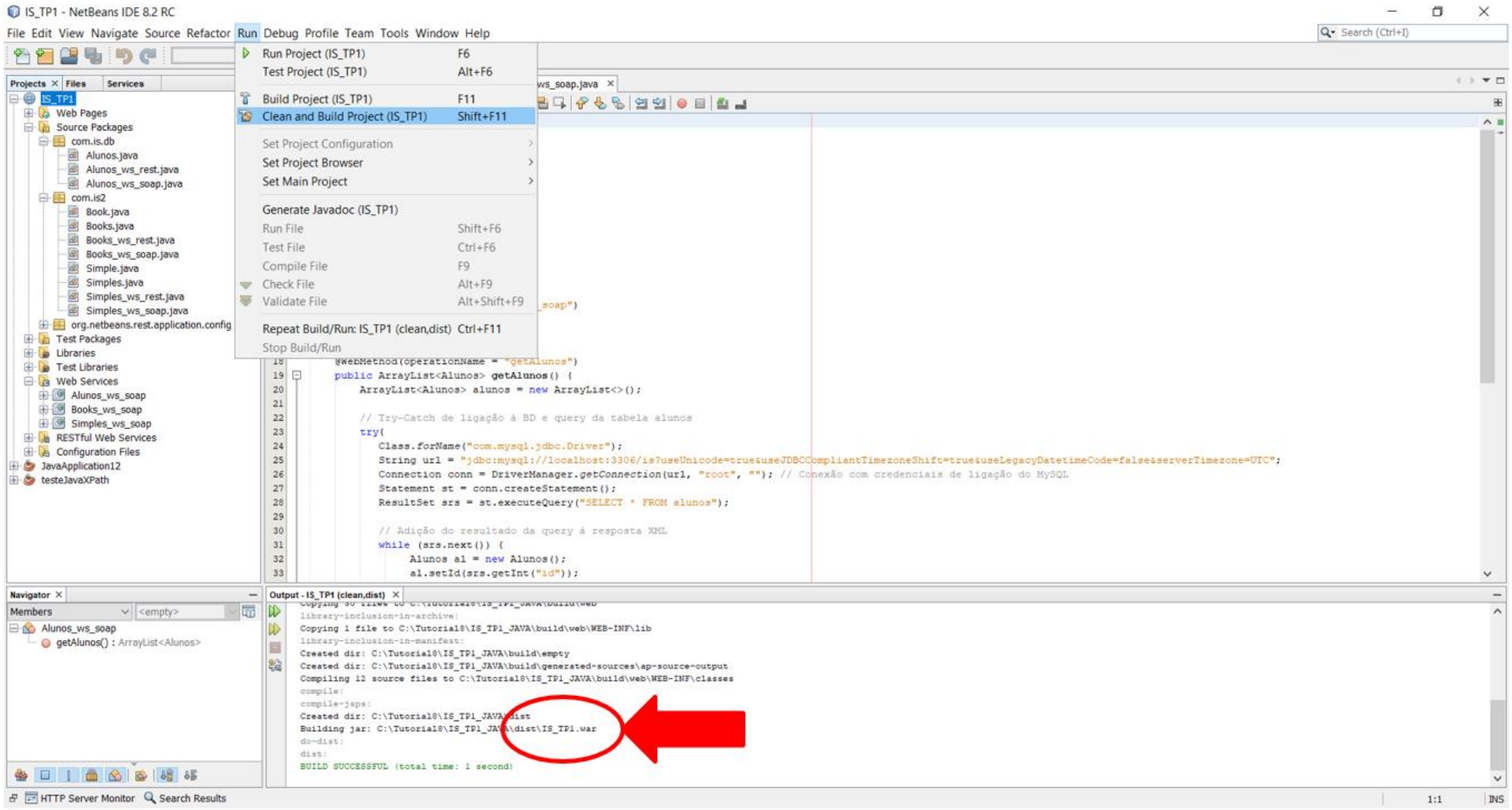


Naming the new servlet in Netbeans

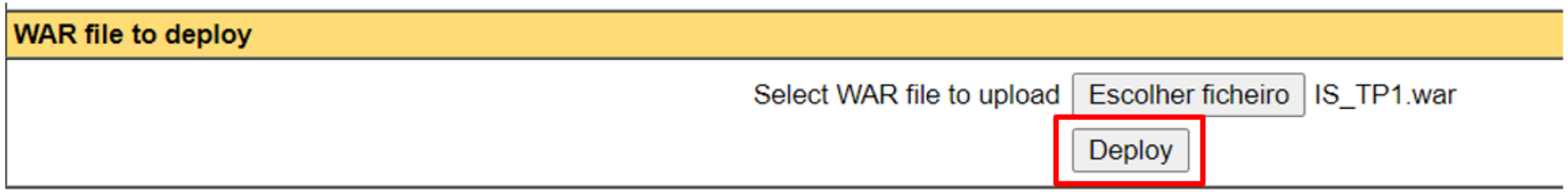
1. Publicação do projeto java (sem código fonte) no servidor Tomcat;

- 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:

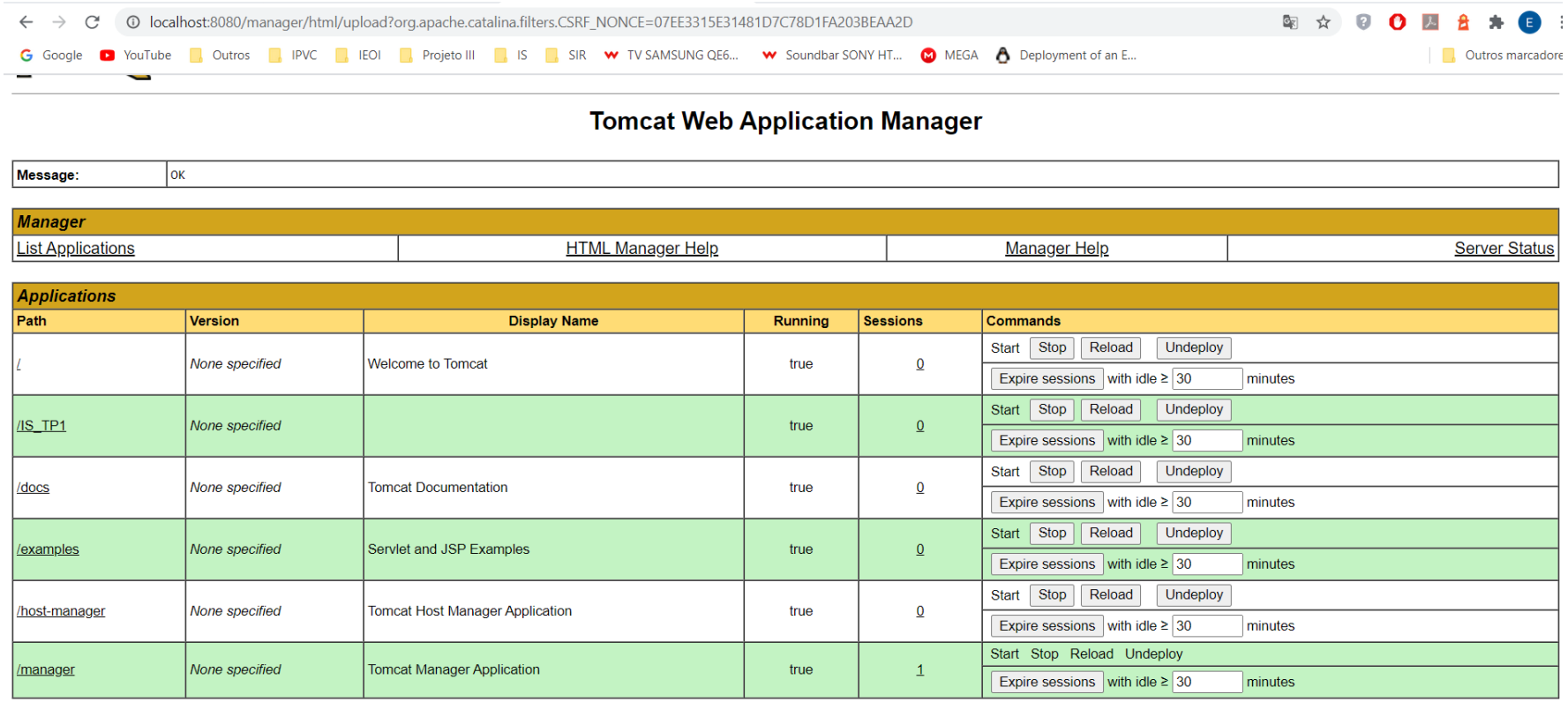
1. Publicação do projeto java (sem código fonte) no servidor Tomcat;



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

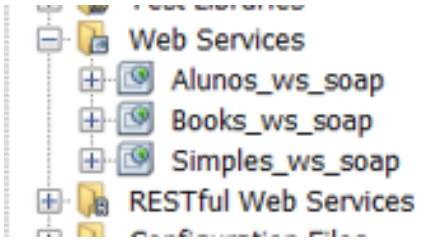


1. Publicação do projeto java (sem código fonte) no servidor Tomcat;



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:

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:

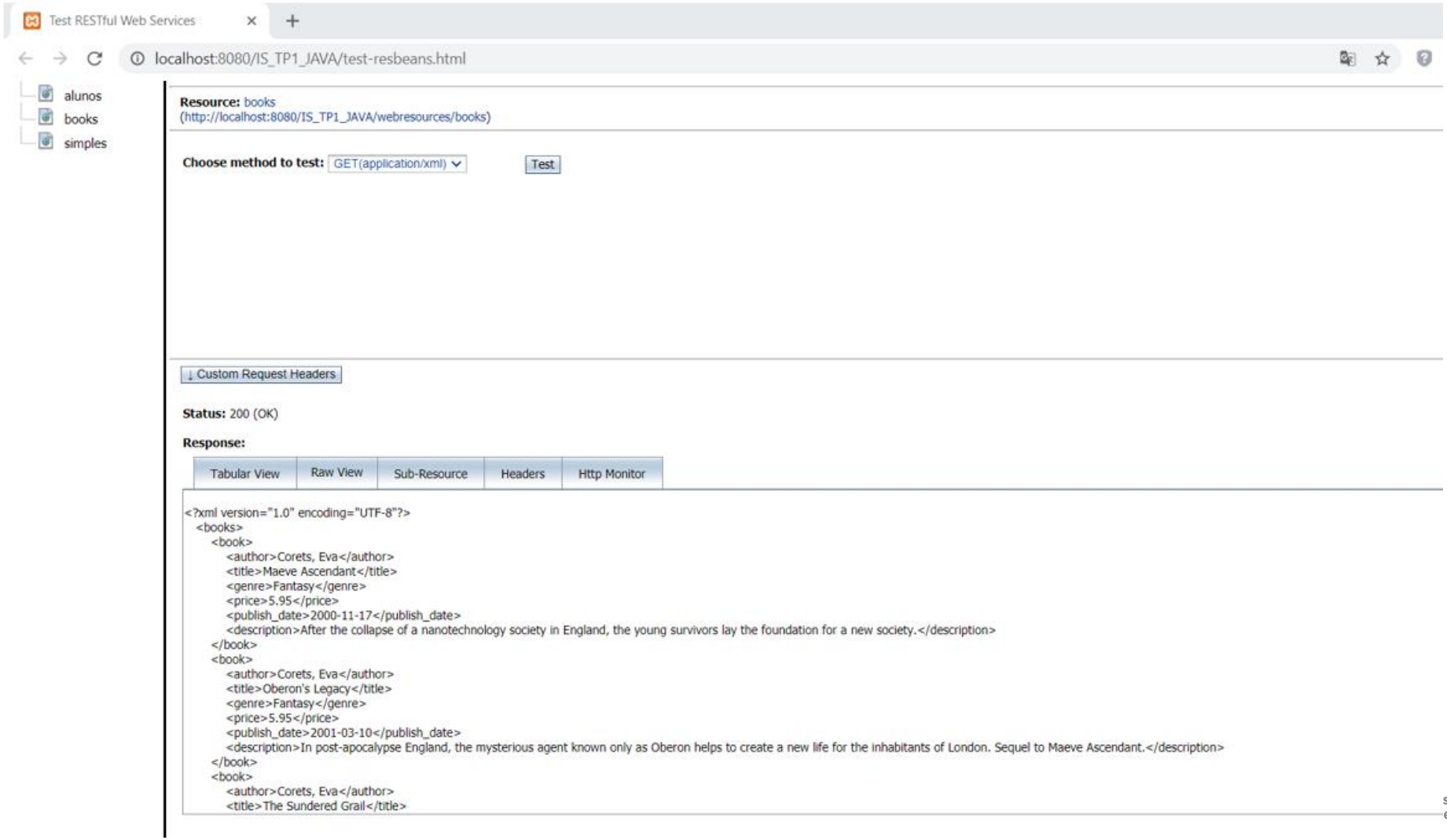
Books_ws_rest.java | Book.java | Alunos_ws_soap.java

Source | History

```
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;

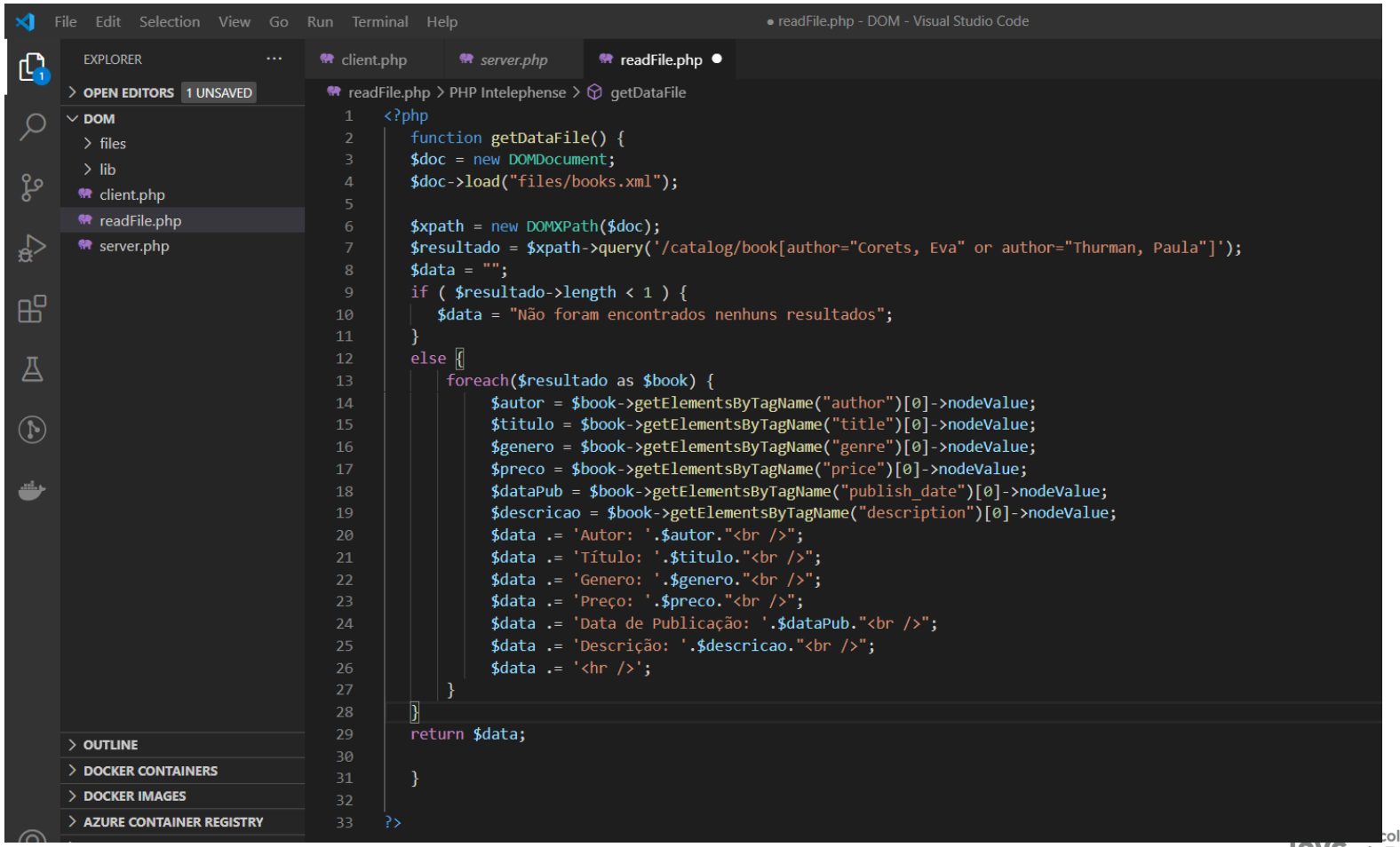
Resultado do webservice



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:



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

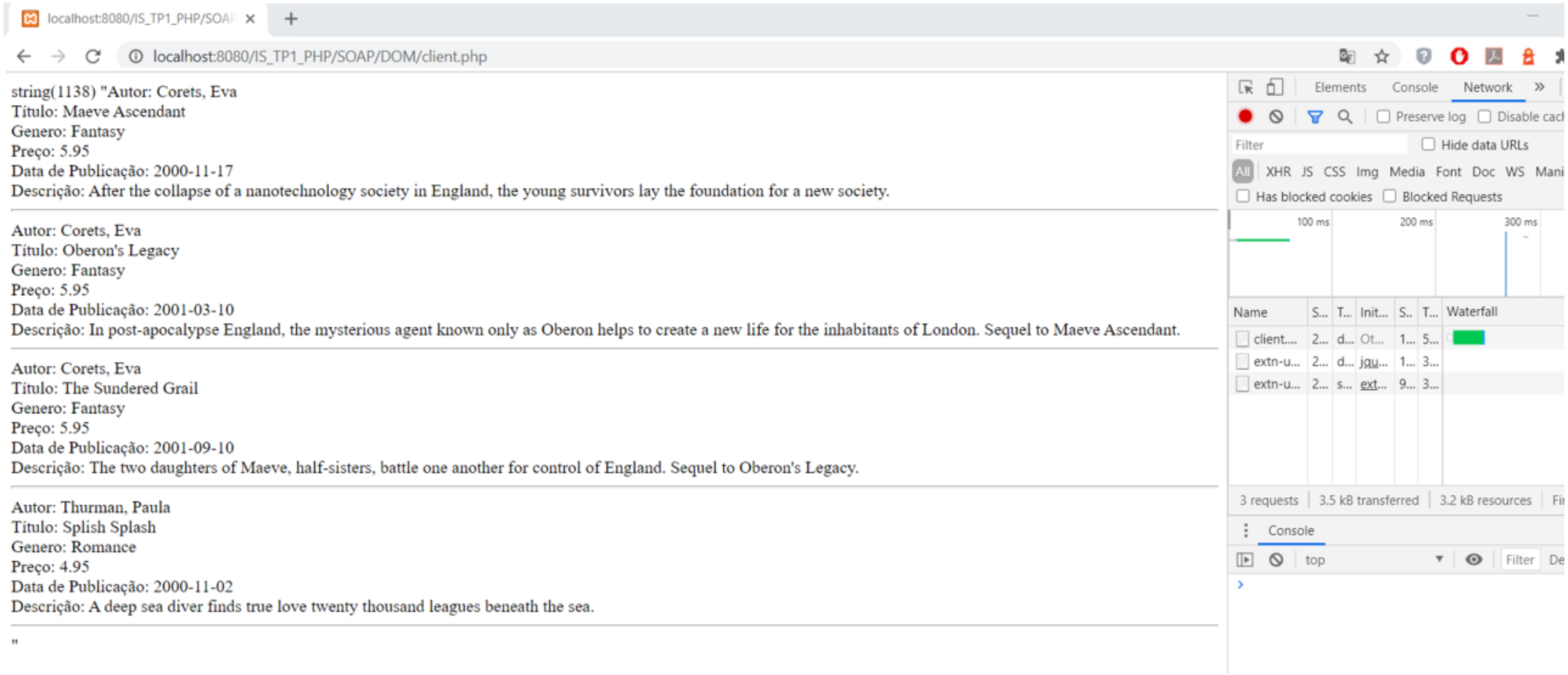
- Em PHP, Implementação de webservice via SOAP para leitura de ficheiro books.xml usando o parser DOM

Retificação do código:

```php
<?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 ) {
       $data = "Não foram encontrados nenhuns resultados";
    }
    else {
        foreach($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 />';
        }
    }

```

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

Resultado da invocação do webservice:

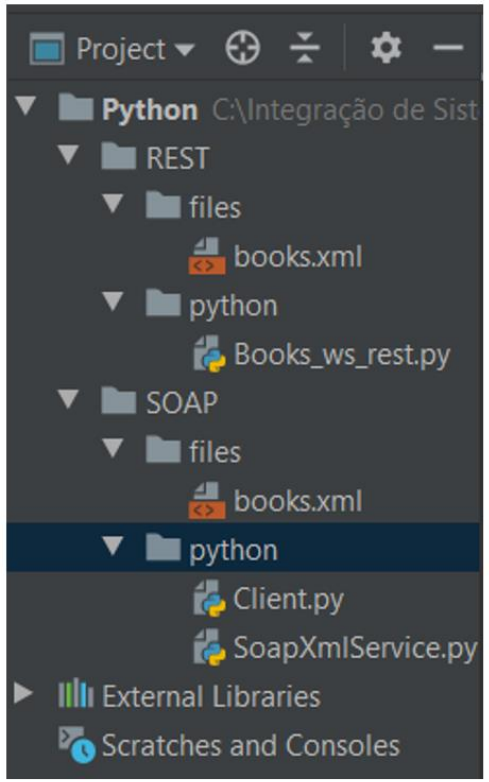


3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação;

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 o aqueles que correspondem à expressão referida. acedendo 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

```python
from xml.dom import minidom
from flask import Flask, request
from lxml import etree
from werkzeug.wrappers import Response
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Esta aplicação contém os seguintes métodos: <h3>"
    html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
    html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/getBooks/books', methods=['GET'])
def getBooks():
    typeRequest = request.args.get('type')
    if typeRequest.upper() == "HTML":
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
            node = itemlist.item(i)
            autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
            titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
            genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
            preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
            data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
            descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
            data = data + "<h5>Autor: " + autor + "<h5>"
            data = data + "<h5>Título: " + 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:

```python
from xml.dom import minidom
from flask import Flask, request
from lxml import etree
from werkzeug.wrappers import Response
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Esta aplicação contém os seguintes métodos: <h3>"
    html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
    html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/getBooks/books', methods=['GET'])
def getBooks():
    typeRequest = request.args.get('type')
    if typeRequest.upper() == "HTML":
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
            node = itemlist.item(i)
            autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
            titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
            genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
            preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
            data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
            descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
            data = data + "<h5>Autor: " + autor + "<h5>"
            data = data + "<h5>Título: " + titulo + "<h5>"
            data = data + "<h5>Genero: " + genero + "<h5>"
            data = data + "<h5>Preço: " + preco + "<h5>"
            data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
            data = data + "<h5>Descrição: " + descricao + "<h5>"
            data = data + "<br>"
32              data = data + "<h5>Título: " + 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:**

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

```
                data = data + "<h5>Genero: " + genero + "<h5>"
                data = data + "<h5>Preco: " + preco + "<h5>"
                data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
                data = data + "<h5>Descrição: " + descricao + "<h5>"
                data = data + "<br>"
        return data
    if typeRequest.upper() == "XML":
        root = etree.parse('../files/books.xml').getroot()
        xml = etree.tostring(root, pretty_print=True)
        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><br>"
    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
        descricao = books[i][5].text
        data = data + "<h5>Autor: " + autor + "<h5>"
        data = data + "<h5>Título: " + titulo + "<h5>"
        data = data + "<h5>Genero: " + genero + "<h5>"
        data = data + "<h5>Preco: " + preco + "<h5>"
        data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
        data = data + "<h5>Descrição: " + descricao + "<h5>"
        data = data + "<br>"
    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)
68
```

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:

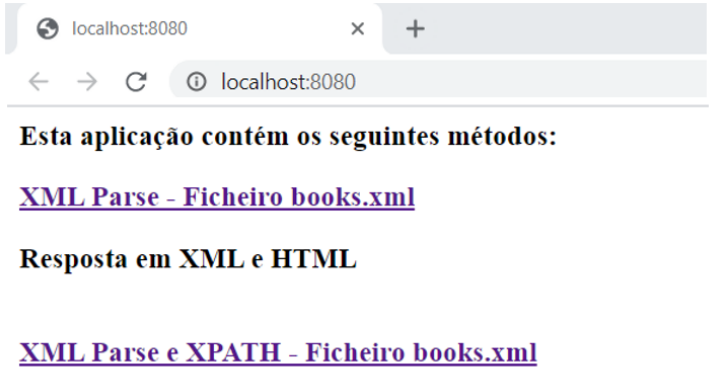
Função principal para iniciar o serviço:

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

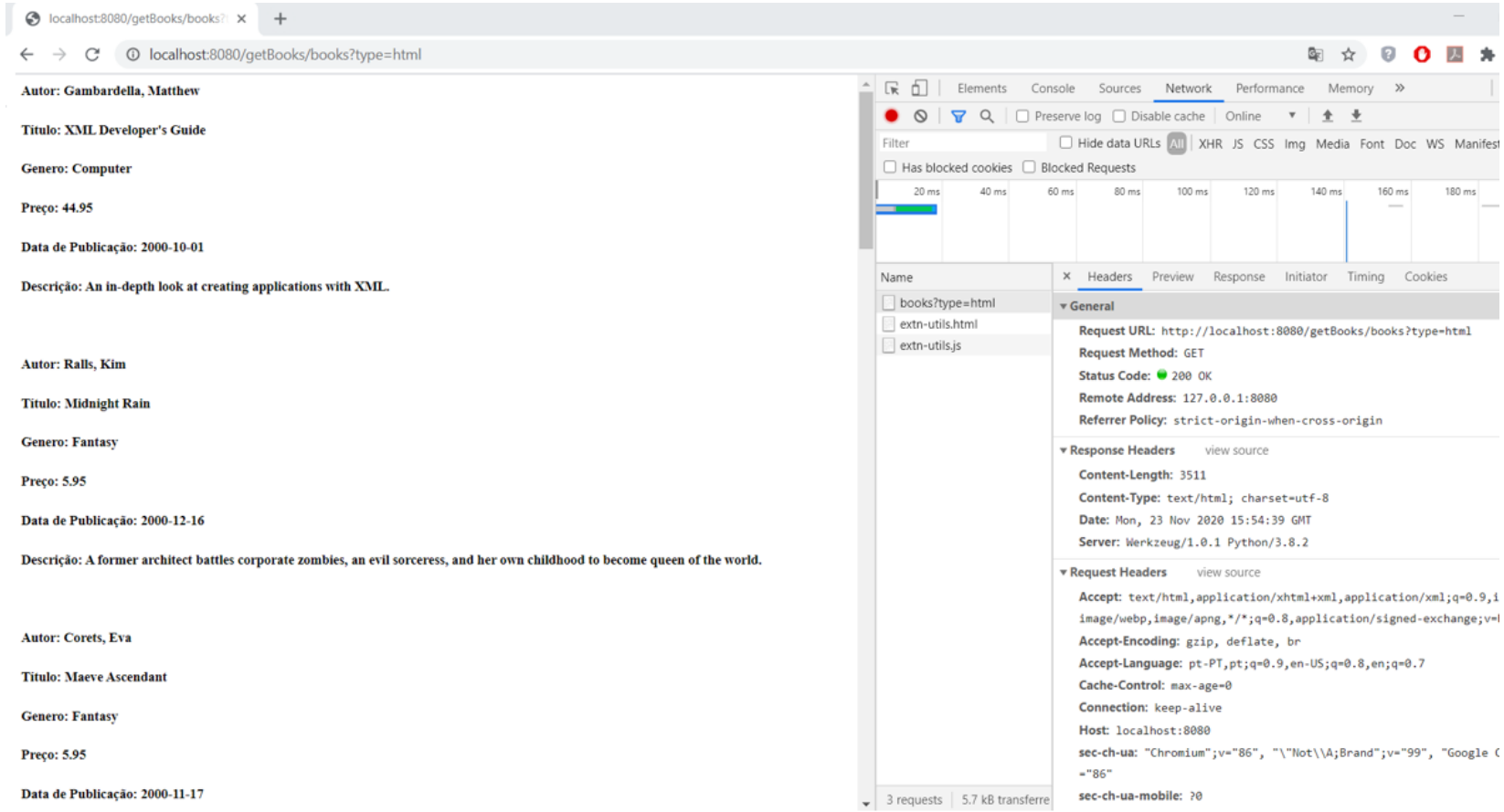
Serviço da Web em execução no localhost, porta 8080 e acessível no ponto de extremidade raiz por meio do navegador

```
Books_ws_rest
C:\Users\eduve\PycharmProjects\testeXPath\venv\Scripts\python.exe "C:/Integração de Sistemas/TP1/ex5_codigo/Python/REST/python/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)
```

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:

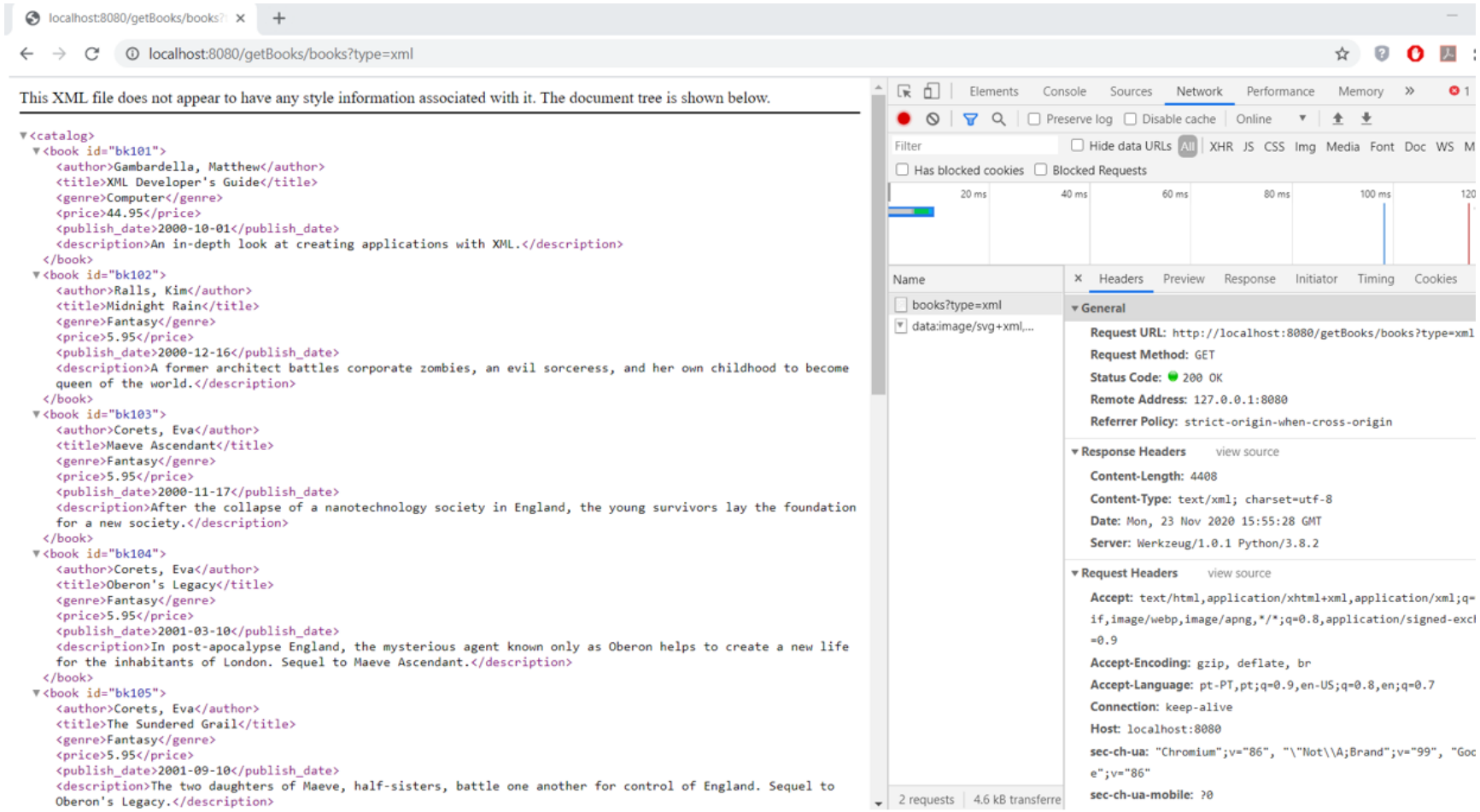


Chamar o endpoint getBooks / books especificando o tipo de saída como html



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



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

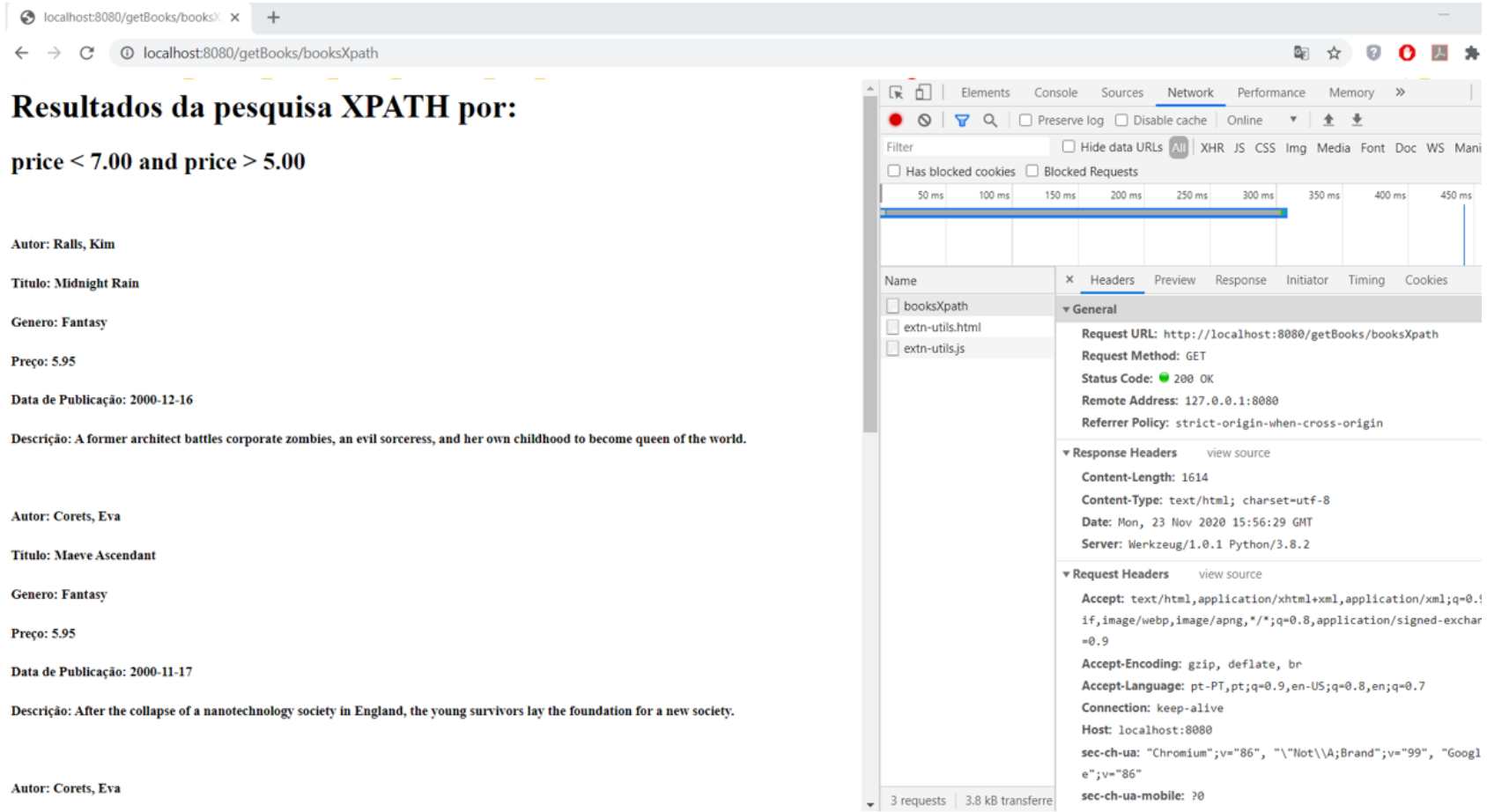
```
        return Response(xml, mimetype='text/xml')

@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 pr
    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
```

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:**

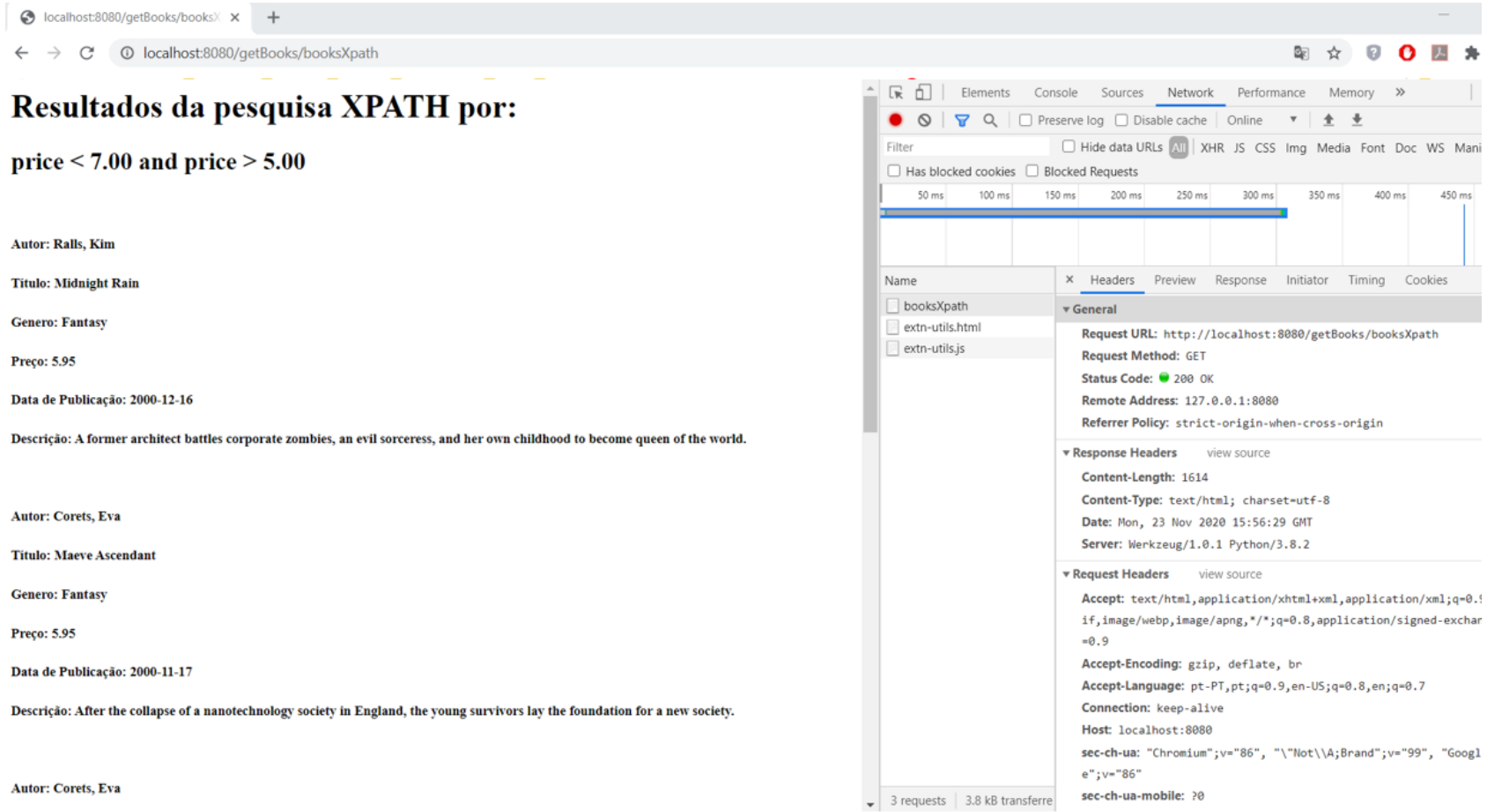
Chamar o endpoint getBooks / booksXpath e o resultado obtido

```
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 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)
68
```

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:



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:



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

```python
from spyne import Application, rpc, ServiceBase, Unicode
from spyne.protocol.soap import Soap11
from spyne.server.wsgi import WsgiApplication
from xml.dom import minidom
from lxml import etree


class SoapXmlService(ServiceBase):
    @rpc(_returns=Unicode)
    def books_parse(ctx):
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
            node = itemlist.item(i)
            autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
            titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
            genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
            preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
            data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
            descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
            data = data + "<h5>Autor: " + autor + "<h5>"
            data = data + "<h5>Título: " + titulo + "<h5>"
            data = data + "<h5>Genero: " + genero + "<h5>"
            data = data + "<h5>Preco: " + preco + "<h5>"
            data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
            data = data + "<h5>Descrição: " + descricao + "<h5>"
            data = data + "<br>"
        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:**

```python
from spyne import Application, rpc, ServiceBase, Unicode
from spyne.protocol.soap import Soap11
from spyne.server.wsgi import WsgiApplication
from xml.dom import minidom
from lxml import etree


class SoapXmlService(ServiceBase):
    @rpc(_returns=Unicode)
    def books_parse(ctx):
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
            node = itemlist.item(i)
            autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
            titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
            genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
            preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
            data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
            descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
            data = data + "<h5>Autor: " + autor + "<h5>"
            data = data + "<h5>Título: " + titulo + "<h5>"
            data = data + "<h5>Genero: " + genero + "<h5>"
            data = data + "<h5>Preço: " + preco + "<h5>"
            data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
            data = data + "<h5>Descrição: " + descricao + "<h5>"
            data = data + "<br>"
        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          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>Preco: " + 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(validator='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(validator='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(validator='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 | Client.py

```python
from zeep import Client
from flask import Flask
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Servidor Flask para teste do Web service em SOAP: <h3>"
    html = html + "<a href='/soap/booksParse'>XML Parse - Ficheiro books.xml</a><br>"
    html = html + "<a href='/soap/Xpath'>XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/soap/booksParse', methods=['GET'])
def booksParseSOAP():
    client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
    data = client.service.books_parse()
    return data

@app.route('/soap/Xpath', methods=['GET'])
def booksXpath():
    client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
    data = client.service.books_Xpath()
    return data

if __name__ == "__main__":
    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:**

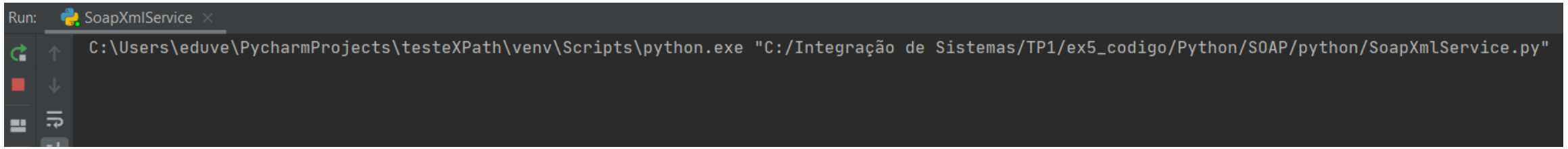
Para o cliente, foi criado um servidor flask com métodos que podem ser chamados a partir do navegador e criar a partir da biblioteca zeep, um cliente, que chama o webservice criado e o respectivo método.

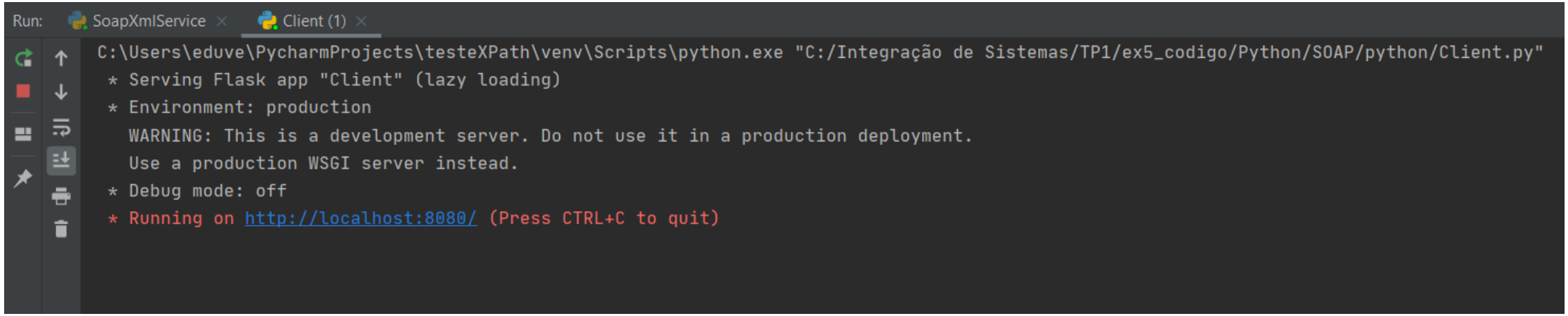
SoapXmlService.py | Client.py

```python
from zeep import Client
from flask import Flask
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Servidor Flask para teste do Web service em SOAP: <h3>"
    html = html + "<a href='/soap/booksParse'>XML Parse - Ficheiro books.xml</a><br>"
    html = html + "<a href='/soap/Xpath'>XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/soap/booksParse', methods=['GET'])
def booksParseSOAP():
    client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
    data = client.service.books_parse()
    return data

@app.route('/soap/Xpath', methods=['GET'])
def booksXpath():
    client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
    data = client.service.books_Xpath()
    return data

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

```

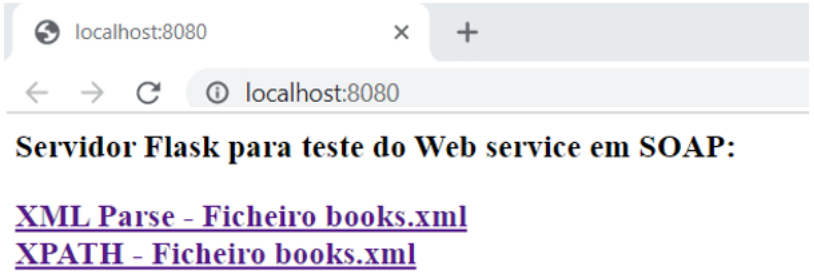
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:**

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.





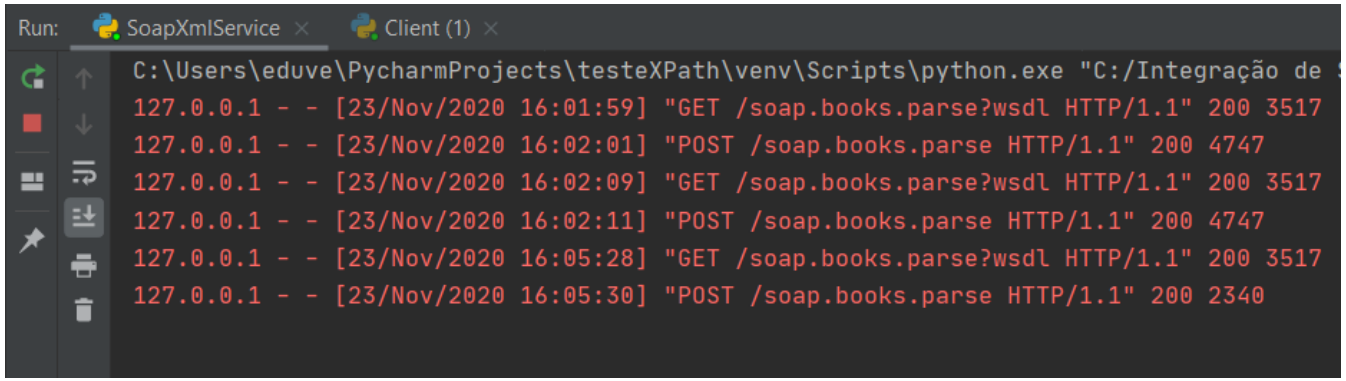
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; IMPLEMENTAÇÃO PYTHON VIA SOAP USANDO XPATH:



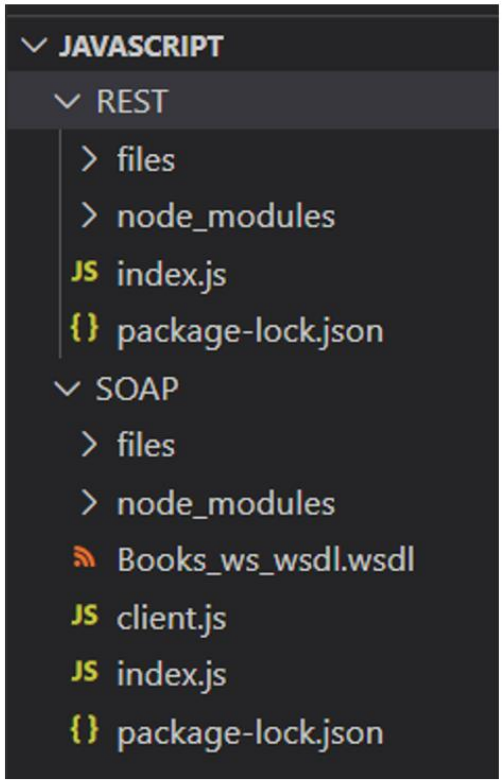
Resultado da chamada do método books_parse no serviço web, com os dados apresentados no lado do cliente.

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

Chamadas registadas para o serviço da web.



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.

index.js

REST > 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    }
32    else {
33       res.setHeader("Content-Type", "text/plain; charset=utf-8");
34       res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
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 necessá
de extremidad
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    }
32    else {
33       res.setHeader("Content-Type", "text/plain; charset=utf-8");
34       res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
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[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     }
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     }
```

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 extrem
expressão XPath para obter
o preço é superior a 30,50 e
Computador. Também co
contendo os dados em forma
ao cliente.

```
app.get('/booksXpath', function (req, res) {
    const queryObject = url.parse(req.url, true).query;
    if (queryObject["file"] == "books") {
        fs.readFile("files/books.xml", 'utf8', function (err, data) {
            var doc = new dom().parseFromString(data)
            var nodes = xpath.select("/catalog/book[price>30.50 and genre='Computer']", doc)
            var ids = xpath.select("/catalog/book/@id", doc)
            var i = 0;
            var dados = "<h2>RESULTADOS FILTRAGEM XPATH POR:</h2><br><p>price > 30.50 e genre = Computer</p><br>";

            for (i = 0; i < nodes.length; i++) {
                var bookId = ids[i].value
                var book = nodes[0];
                var autor = book.getElementsByTagName('author')[0].childNodes[0].nodeValue;
                var titulo = book.getElementsByTagName('title')[0].childNodes[0].nodeValue;
                var genero = book.getElementsByTagName('genre')[0].childNodes[0].nodeValue;
                var preco = book.getElementsByTagName('price')[0].childNodes[0].nodeValue;
                var dataPublicacao = book.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue;
                var descricao = book.getElementsByTagName('description')[0].childNodes[0].nodeValue;
                dados = dados + `<p>Id: ${bookId}</p>`
                dados = dados + `<p>Autor: ${autor}</p>`
                dados = dados + `<p>Título: ${titulo}</p>`
                dados = dados + `<p>Genero: ${genero}</p>`
                dados = dados + `<p>Preço: ${preco}</p>`
                dados = dados + `<p>Data de Publicação: ${dataPublicacao}</p>`
                dados = dados + `<p>Descrição: ${descricao}</p><br>`
            }
            res.setHeader("Content-Type", "text/html; charset=utf-8");
            res.end(dados);
        });
    }
    else {
        res.setHeader("Content-Type", "text/plain; charset=utf-8");
        res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
    }
```

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  })
```

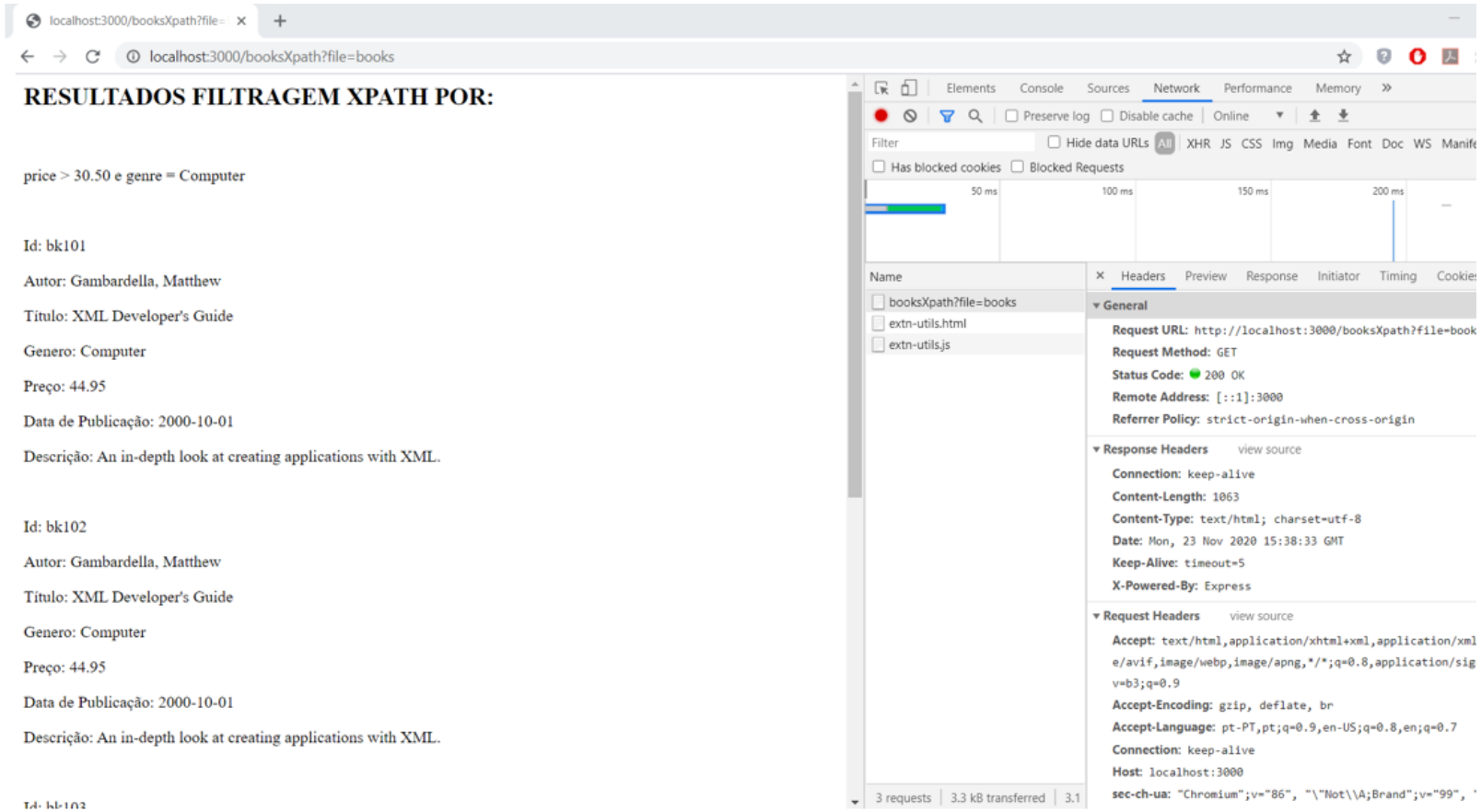
Linha de comandos - node index.js

```
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.



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

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://tempuri.org/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://tempuri.org/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <!-- Definição dos pedidos e respostas para o método parse_books  -->
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="BooksParseRequest">
        <s:complexType>
          <s:sequence>
          <s:element minOccurs="1" maxOccurs="unbounded" name="type" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BooksParseResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="unbounded" name="result" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
```

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:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://tempuri.org/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://tempuri.org/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <!-- Definição dos pedidos e respostas para o método parse_books  -->
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="BooksParseRequest">
        <s:complexType>
          <s:sequence>
          <s:element minOccurs="1" maxOccurs="unbounded" name="type" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BooksParseResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="unbounded" name="result" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <!-- Definição dos dados e do seu tipo entre o cliente e o servidor -->
  <wsdl:message name="BooksParseSoapIn">
    <wsdl:part name="parameters" element="tns:BooksParseRequest"/>
  </wsdl:message>
  <wsdl:message name="BooksParseSoapOut">
    <wsdl:part name="parameters" element="tns:BooksParseResponse"/>
  </wsdl:message>
```

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 ×

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 servic e o cliente, a operação serviço web e a forma como mensagens serão transmitidas

Books_ws_wsdl.wsdl

```
<!-- Definição dos dados e do seu tipo entre o cliente e o servidor -->
<wsdl:message name="BooksParseSoapIn">
  <wsdl:part name="parameters" element="tns:BooksParseRequest"/>
</wsdl:message>
<wsdl:message name="BooksParseSoapOut">
  <wsdl:part name="parameters" element="tns:BooksParseResponse"/>
</wsdl:message>
<wsdl:portType name="BooksParseSoapPort">
  <!-- Definição da operação books_parse, definida no index.js -->
  <wsdl:operation name="BooksParse">
    <wsdl:input message="tns:BooksParseSoapIn"/>
    <wsdl:output message="tns:BooksParseSoapOut"/>
  </wsdl:operation>
</wsdl:portType>
<!-- Definição de como as mensagens serão transmitidas -->
<wsdl:binding name="BooksParseServiceSoapBinding" type="tns:BooksParseSoapPort">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="BooksParse">
    <soap:operation soapAction="BooksParse" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="BooksParseServiceSoap12Binding" type="tns:BooksParseSoapPort">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="BooksParse">
    <soap12:operation soapAction="BooksParse" style="document"/>
    <wsdl:input>
      <soap12:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <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.

index.js

SOAP > 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 necessária web, e o método books_parse, do argumento fornecido na cha análise ou filtro e analisará o arq

index.js

SOAP > index.js > ...

```
var soap = require('soap');
var express = require('express');
var fs = require('fs');
var parser = require('xml2json-light');
var xpath = require('xpath')
var dom = require('xmldom').DOMParser

function books_parse(args) {
  if(args.type == "parse") {
    var data = fs.readFileSync( "files/books.xml", 'utf8');
    var json = parser.xml2json(data);
    var dados = "<h2>PARSE DO FICHEIRO - books.xml</h2><br>";
    var i = 0;
    for(i = 0; i < json.catalog.book.length; i++) {
      var book = json.catalog.book[i]
      dados = dados + `<p>Id: ${book.id}</p>`
      dados = dados + `<p>Autor: ${book.author}</p>`
      dados = dados + `<p>Título: ${book.title}</p>`
      dados = dados + `<p>Genero: ${book.genre}</p>`
      dados = dados + `<p>Preço: ${book.price}</p>`
      dados = dados + `<p>Data de Publicação: ${book.publish_date}</p>`
      dados = dados + `<p>Descrição: ${book.description}</p><br>`
    }
    return {
        result: dados
    }
  }
```

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>,\
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  }
```

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.

index.js

SOAP > 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.

index.js

SOAP > 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.

client.js

SOAP > client.js > app.get('/callBooksParse') callback > soap.createClient() callback

```
var express = require('express');
var app = express();
var soap = require('soap');
var url = 'http://localhost:8000/wsdl?wsdl';

app.get('/callBooksParse', function (req, res) {
    soap.createClient(url, function (err, client) {
      if (err){
        throw err;
      }
      var args = {
        type: 'parse'
      }
      client.BooksParse(args, function (err, result) {
        if (err) {
          throw err;
        }
        else {
          res.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
          res.end(result["result"][0]);
        }
      });
    });
})
```

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.

client.js

SOAP > 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.

client.js

SOAP > 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:3000")
50  })
```

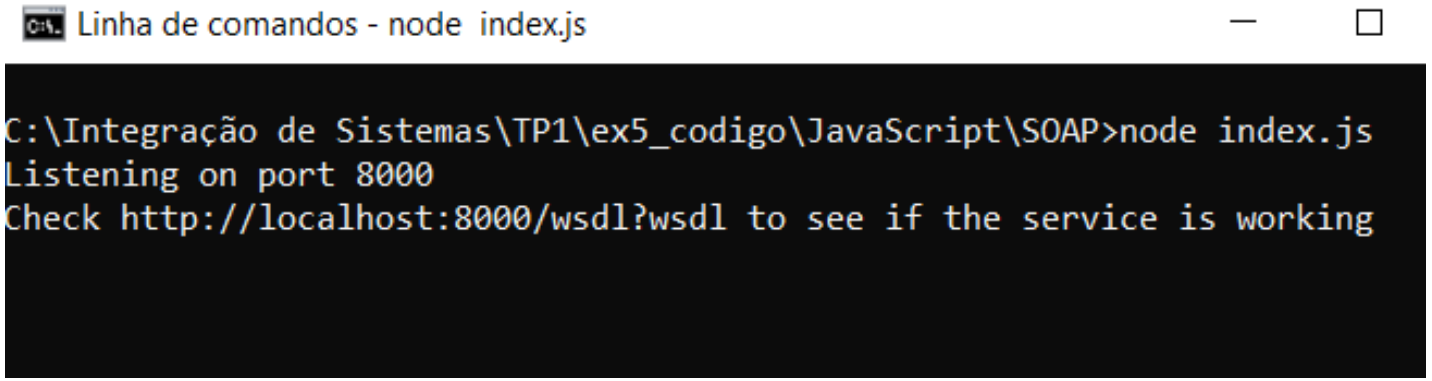
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; **IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:**

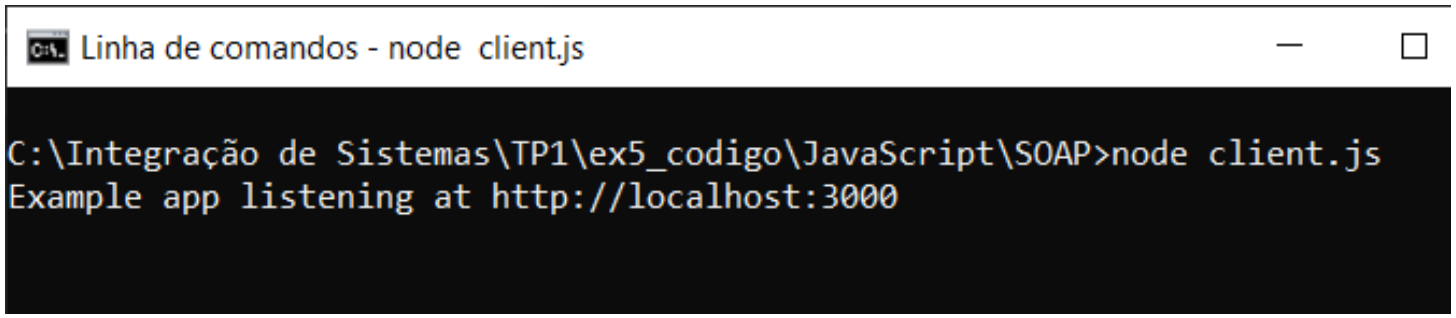
Segundo endpoint onde o serviço web é chamado especificando o tipo de ação como XPath, apresentando os resultados como html.

```
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:**

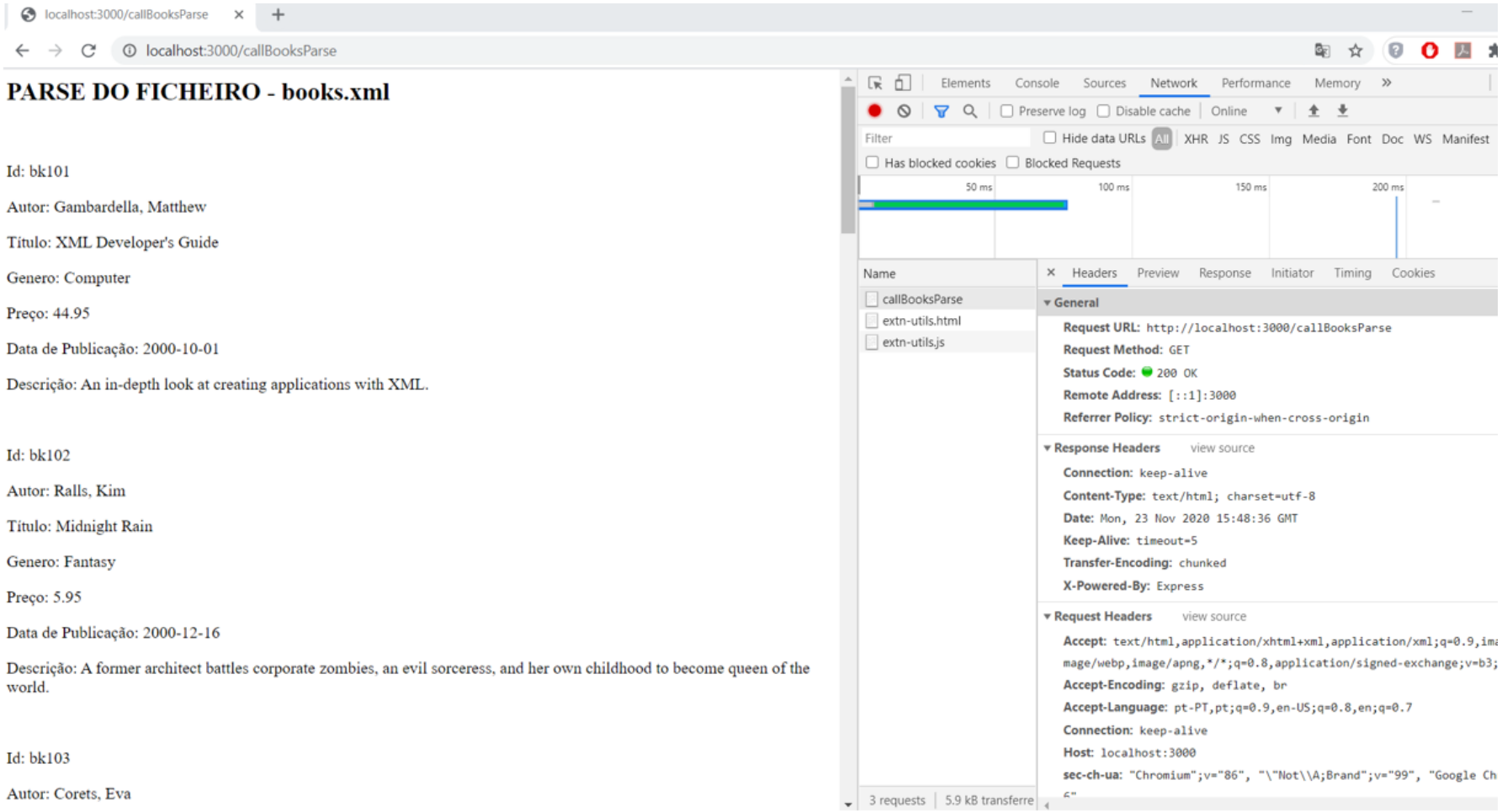
Executando o servidor (imagem na parte superior), rodando o cliente (imagem na parte inferior).





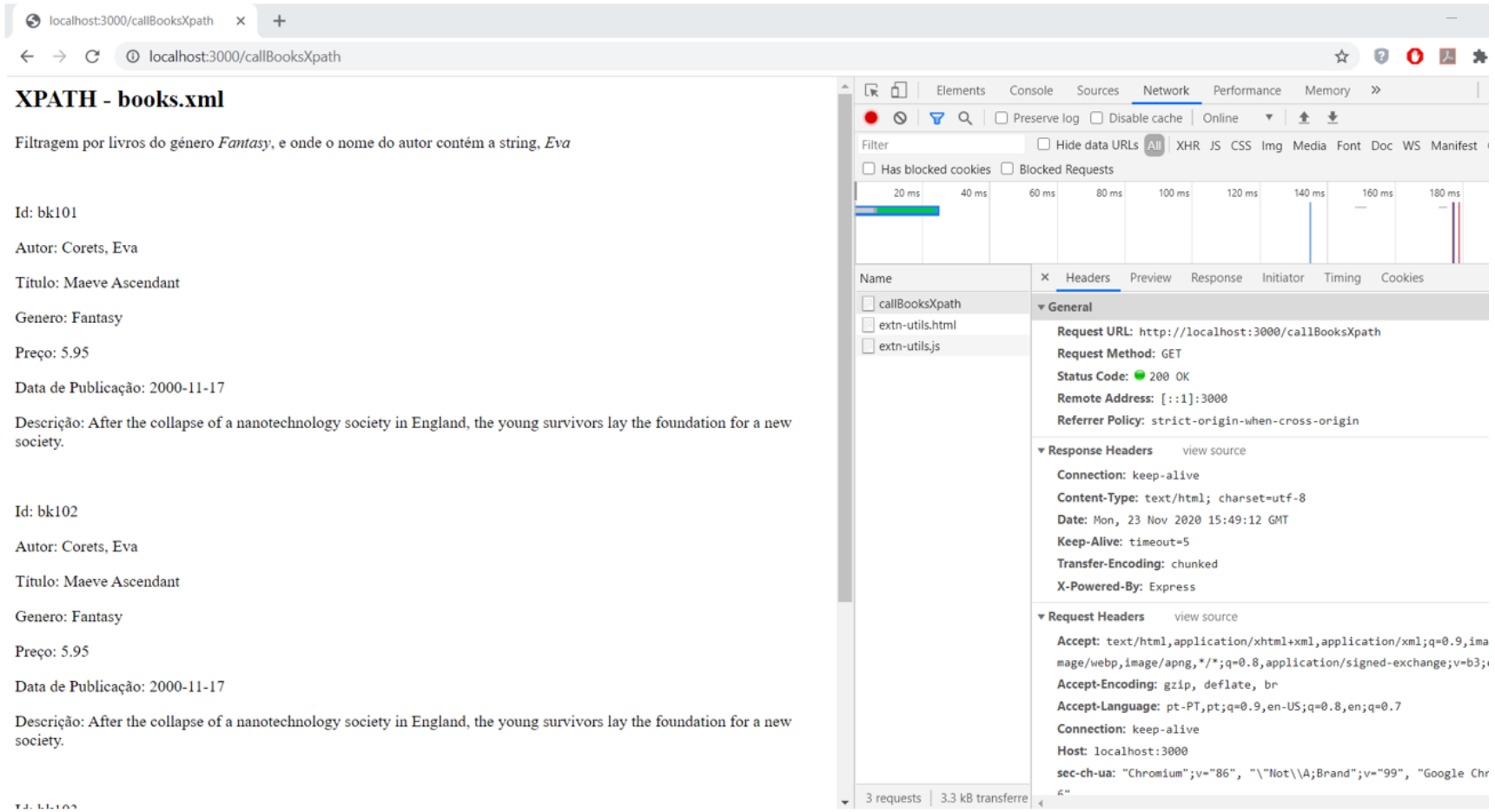
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

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



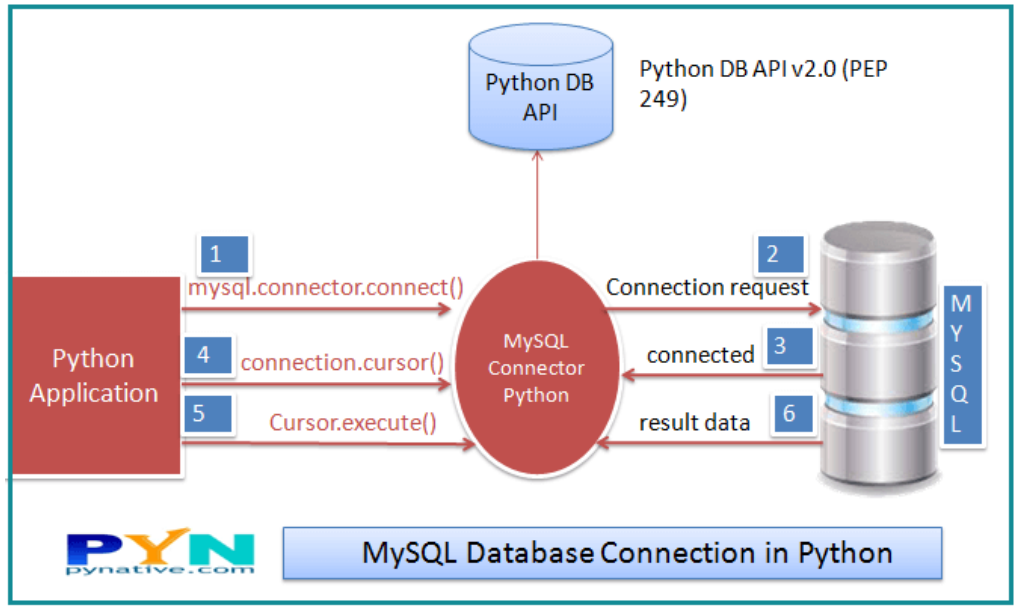
3. Implementação de Webservices (SOAP e REST) com parsing de ficheiros XML e filtros via XPATH em duas linguagens de programação; IMPLEMENTAÇÃO JAVASCRIPT VIA SOAP USANDO XPATH:

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



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:

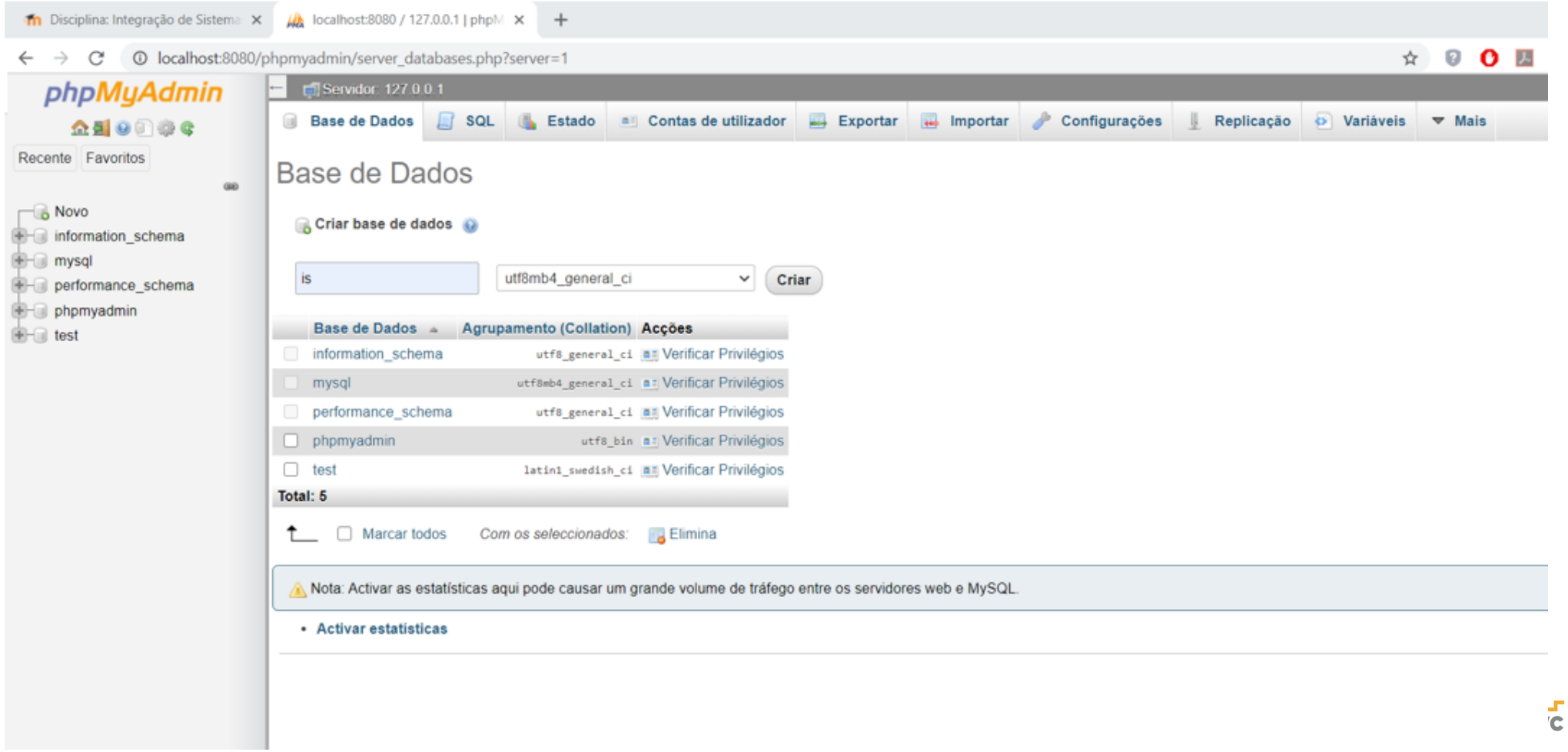
- Como conectar Python com mySQL: https://pynative.com/python-mysql-database-connection/

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

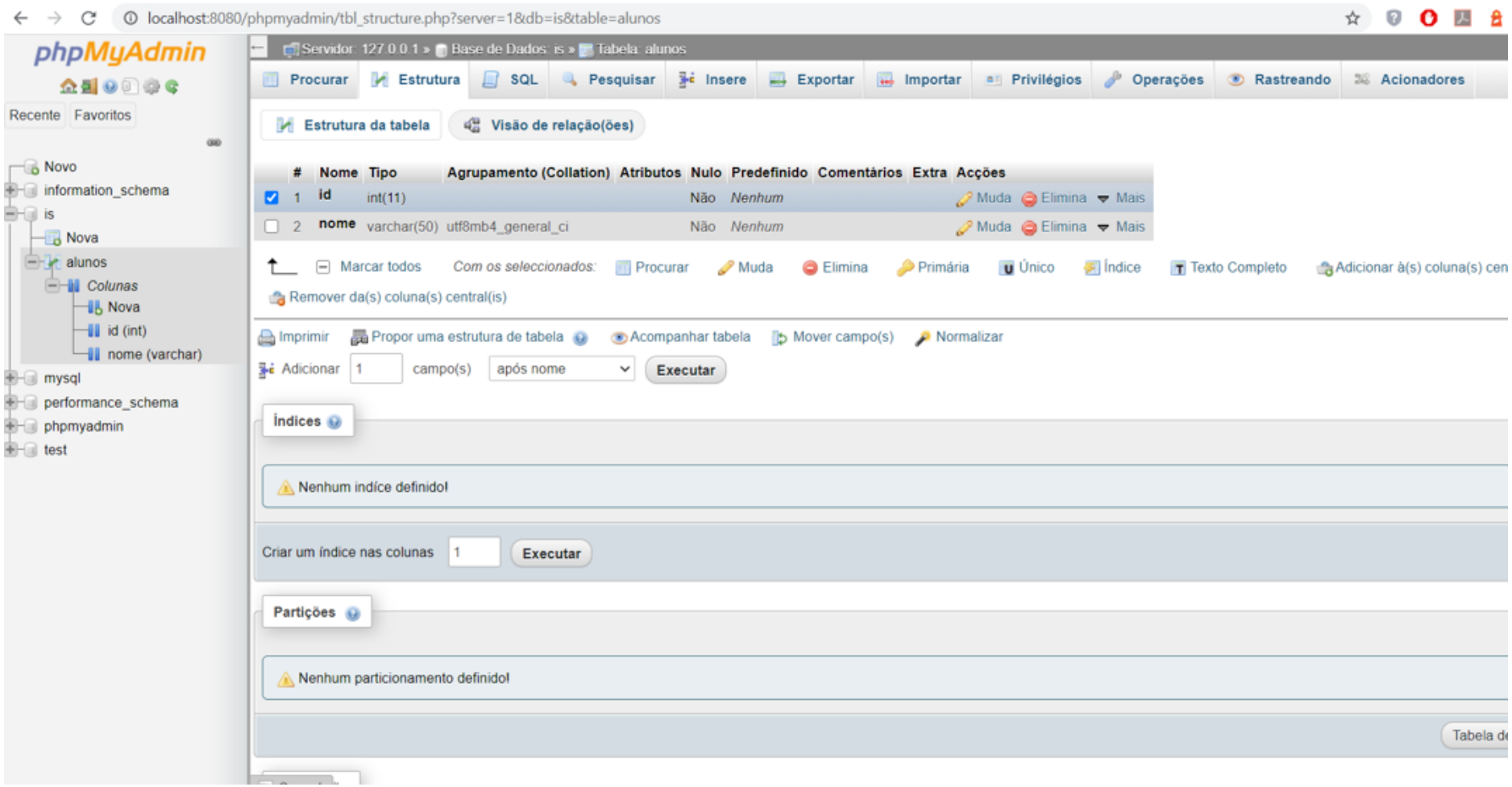
- Para os web services REST e SOAP que envolvem o acesso a uma base de dados MySQL, foram utilizadas as mesmas bibliotecas, mas adicionalmente foi importado o conector MySQL para Python, permitindo a criação de uma conexão com o banco de dados rodando no localhost na porta 3306.

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

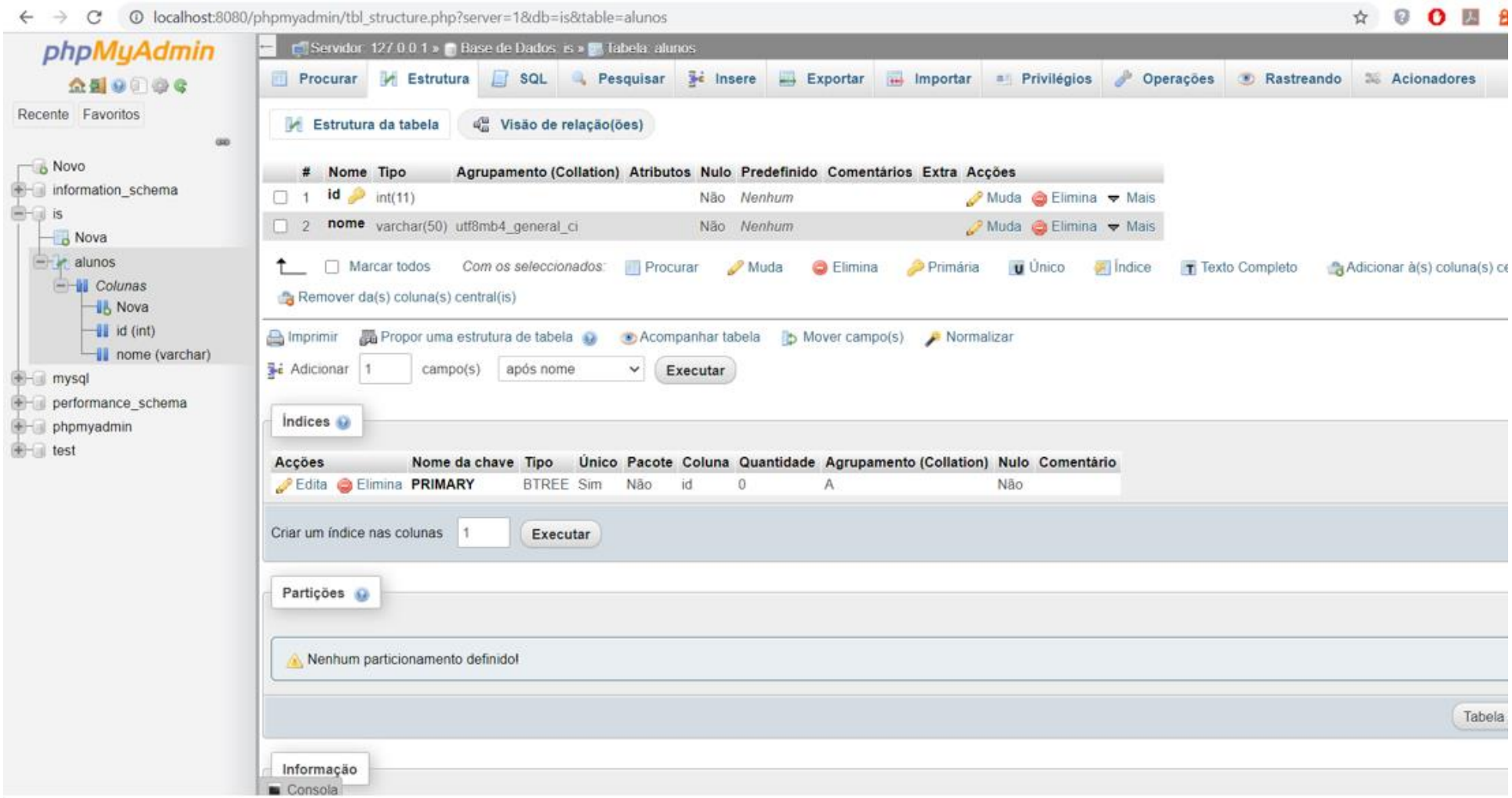
Criação da tabela na base de dados, usando Mysql e o phpMyAdmin, criando a tabela alunos:



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

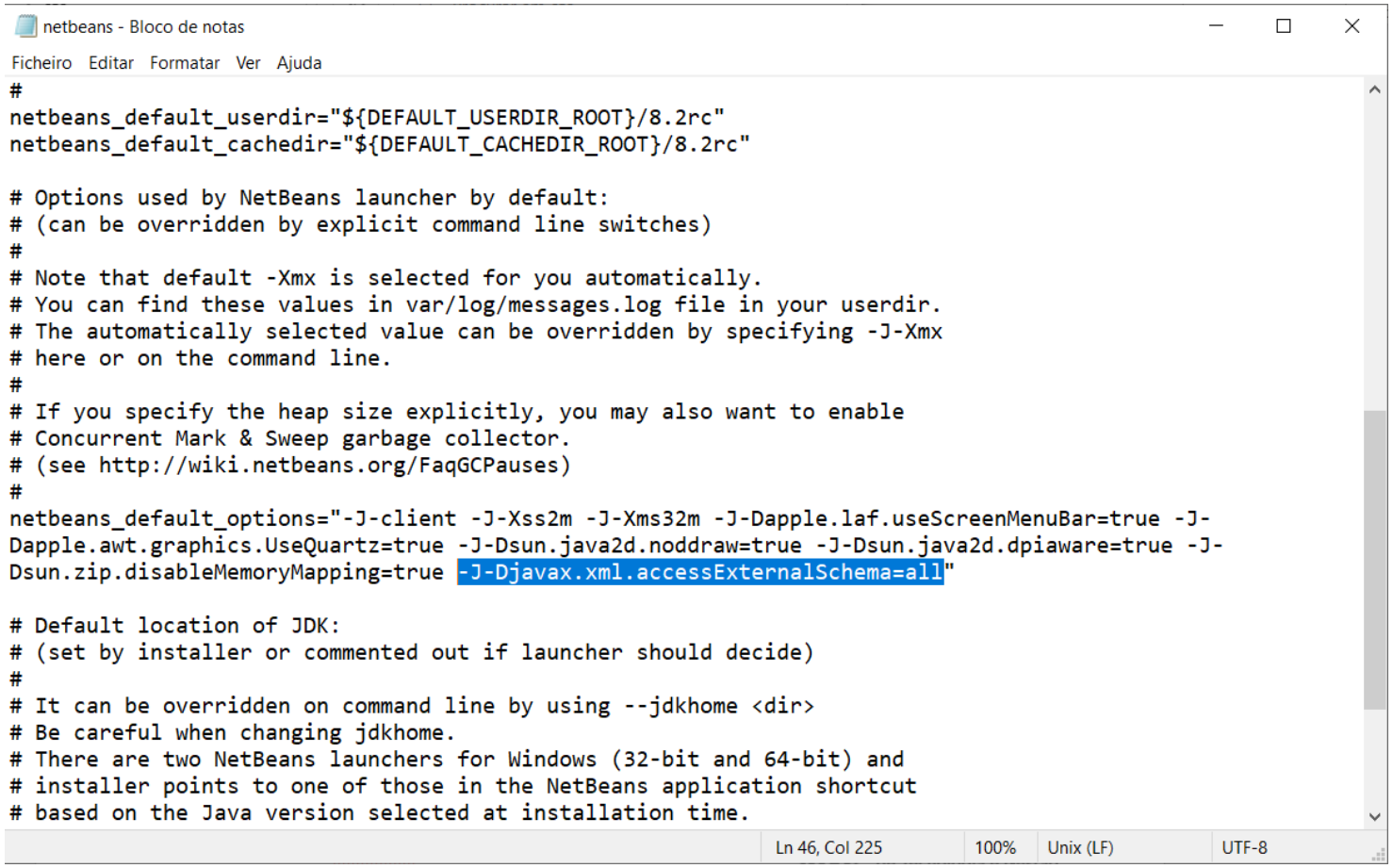


4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.



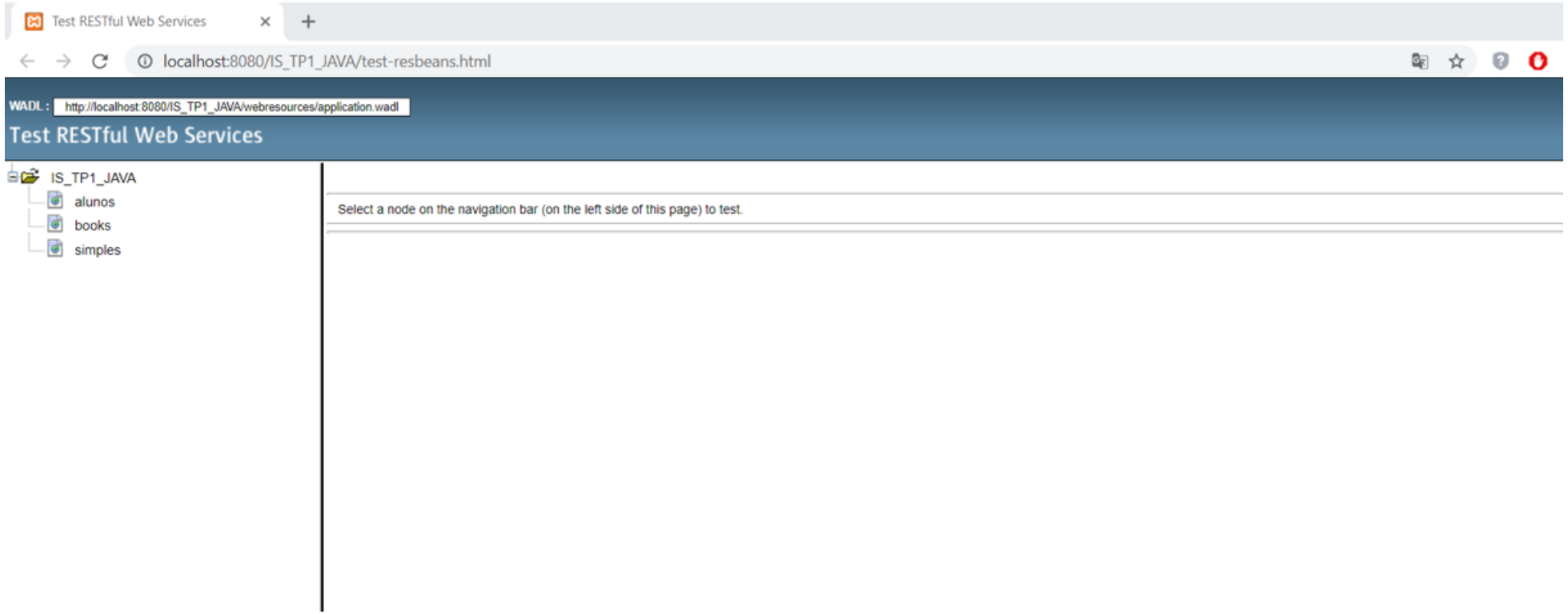
4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Opção adicionada à configuração padrão do NetBeans



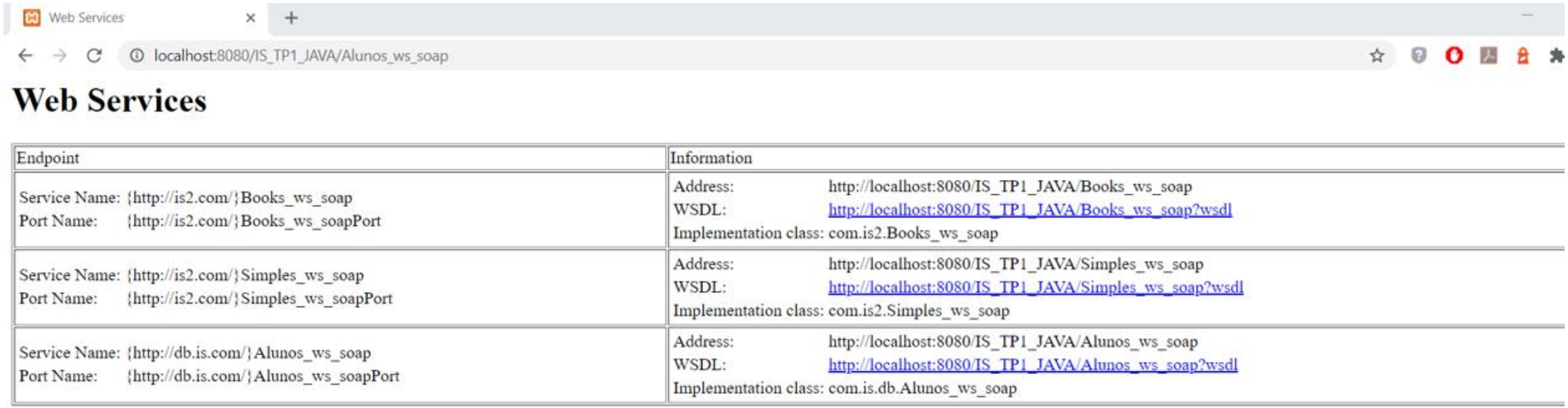
4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

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:

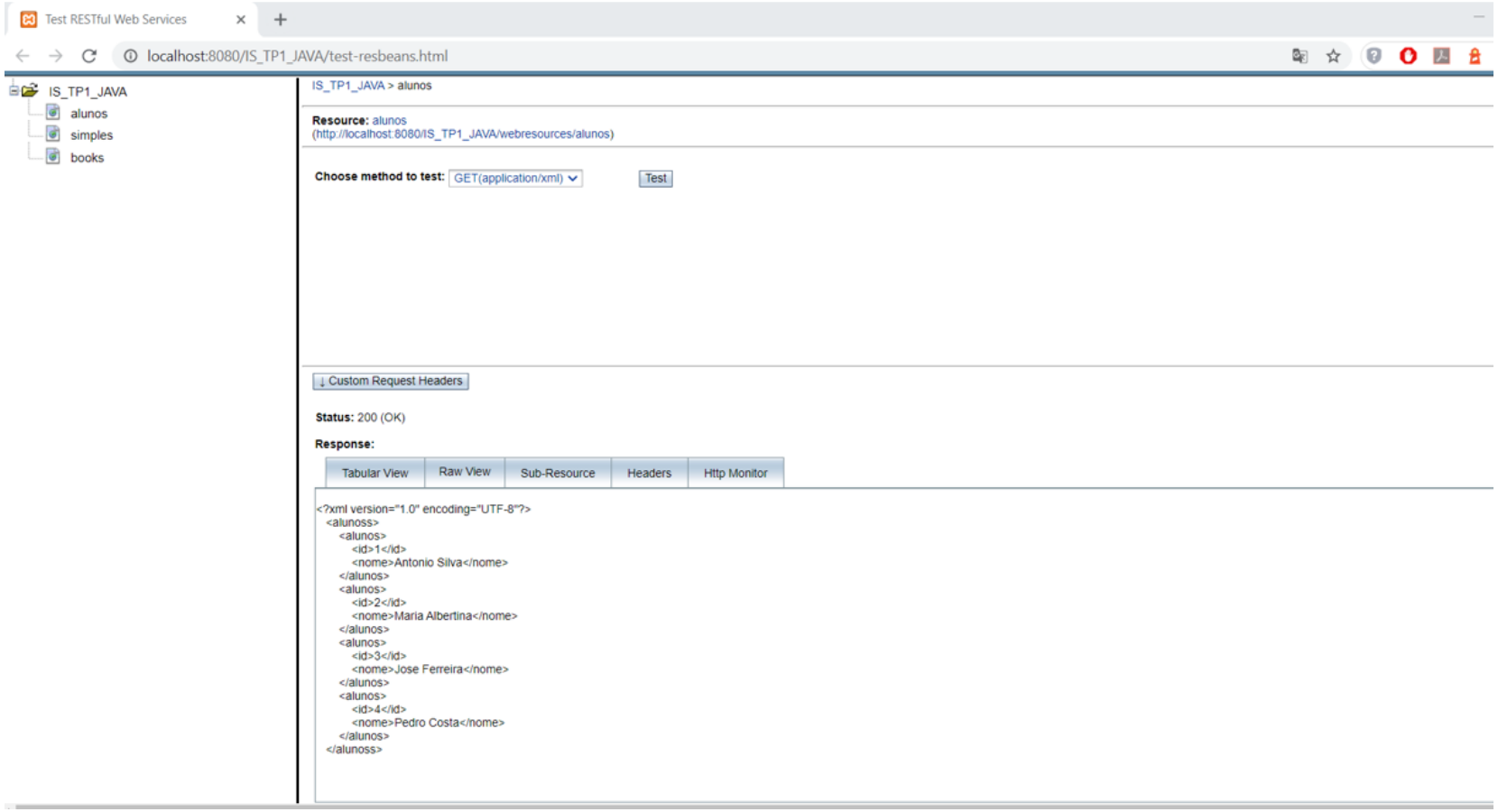


Web Services

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

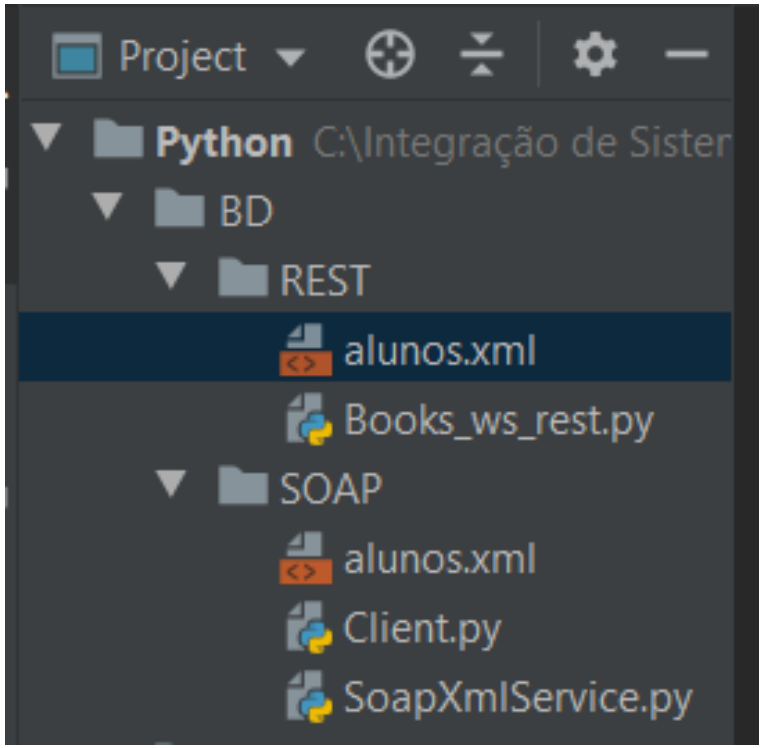
4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

Serviço web REST em java para o ficheiro alunos.xml



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

```python
from xml.dom import minidom
from flask import Flask, request
from werkzeug.wrappers import Response
import mysql.connector
app = Flask(__name__)

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="is"
)

@app.route('/')
def index():
    html = "<h3>Web Service REST com acesso à base de dados MySQL: <h3>"
    html = html + "<a href='/getBooks/books?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
    return html

```

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

```python
from xml.dom import minidom
from flask import Flask, request
from werkzeug.wrappers import Response
import mysql.connector
app = Flask(__name__)

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="is"
)

@app.route('/')
def index():
    html = "<h3>Web Service REST com acesso à base de dados MySQL: <h3>"
    html = html + "<a href='/getAlunos/alunos?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
    return html

```

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

```python
@app.route('/getAlunos/alunos', methods=['GET'])
def getAlunos():
    selectType = request.args.get('type')
    if selectType.upper() == "ALL":
        mycursor = mydb.cursor()
        mycursor.execute("SELECT * FROM alunos")
        myresult = mycursor.fetchall()
        doc = minidom.Document()
        tag_alunos = doc.createElement('alunos')
        for aluno in myresult:
            id_aluno = str(aluno[0])
            nome = aluno[1]
            tag_aluno = doc.createElement('aluno')

            tag_id = doc.createElement('id')
            content_id = doc.createTextNode(id_aluno)
            tag_id.appendChild(content_id)

            tag_nome = doc.createElement('nome')
            content_nome = doc.createTextNode(nome)
            tag_nome.appendChild(content_nome)

            tag_aluno.appendChild(tag_id)
            tag_aluno.appendChild(tag_nome)

            tag_alunos.appendChild(tag_aluno)

        doc.appendChild(tag_alunos)
        string_xml = doc.toprettyxml(indent="\t")
        caminho_ficheiro = "alunos.xml"
        with open(caminho_ficheiro, "w") as f:
            f.write(string_xml)
        return Response(string_xml, mimetype='text/xml')
```

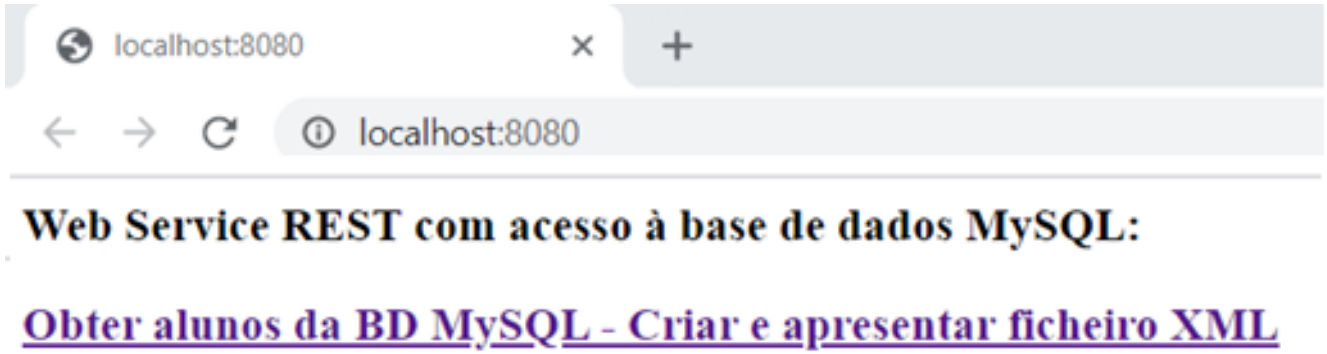
```python
@app.route('/getAlunos/alunos', methods=['GET'])
def getBooks():
    selectType = request.args.get('type')
    if selectType.upper() == "ALL":
        mycursor = mydb.cursor()
        mycursor.execute("SELECT * FROM alunos")
        myresult = mycursor.fetchall()
        doc = minidom.Document()
        tag_alunos = doc.createElement('alunos')
        for aluno in myresult:
            id_aluno = str(aluno[0])
            nome = aluno[1]
            tag_aluno = doc.createElement('aluno')

            tag_id = doc.createElement('id')
            content_id = doc.createTextNode(id_aluno)
            tag_id.appendChild(content_id)

            tag_nome = doc.createElement('nome')
            content_nome = doc.createTextNode(nome)
            tag_nome.appendChild(content_nome)

            tag_aluno.appendChild(tag_id)
            tag_aluno.appendChild(tag_nome)

            tag_alunos.appendChild(tag_aluno)

        doc.appendChild(tag_alunos)
        string_xml = doc.toprettyxml(indent="\t")
        caminho_ficheiro = "alunos.xml"
        with open(caminho_ficheiro, "w") as f:
            f.write(string_xml)
        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
```

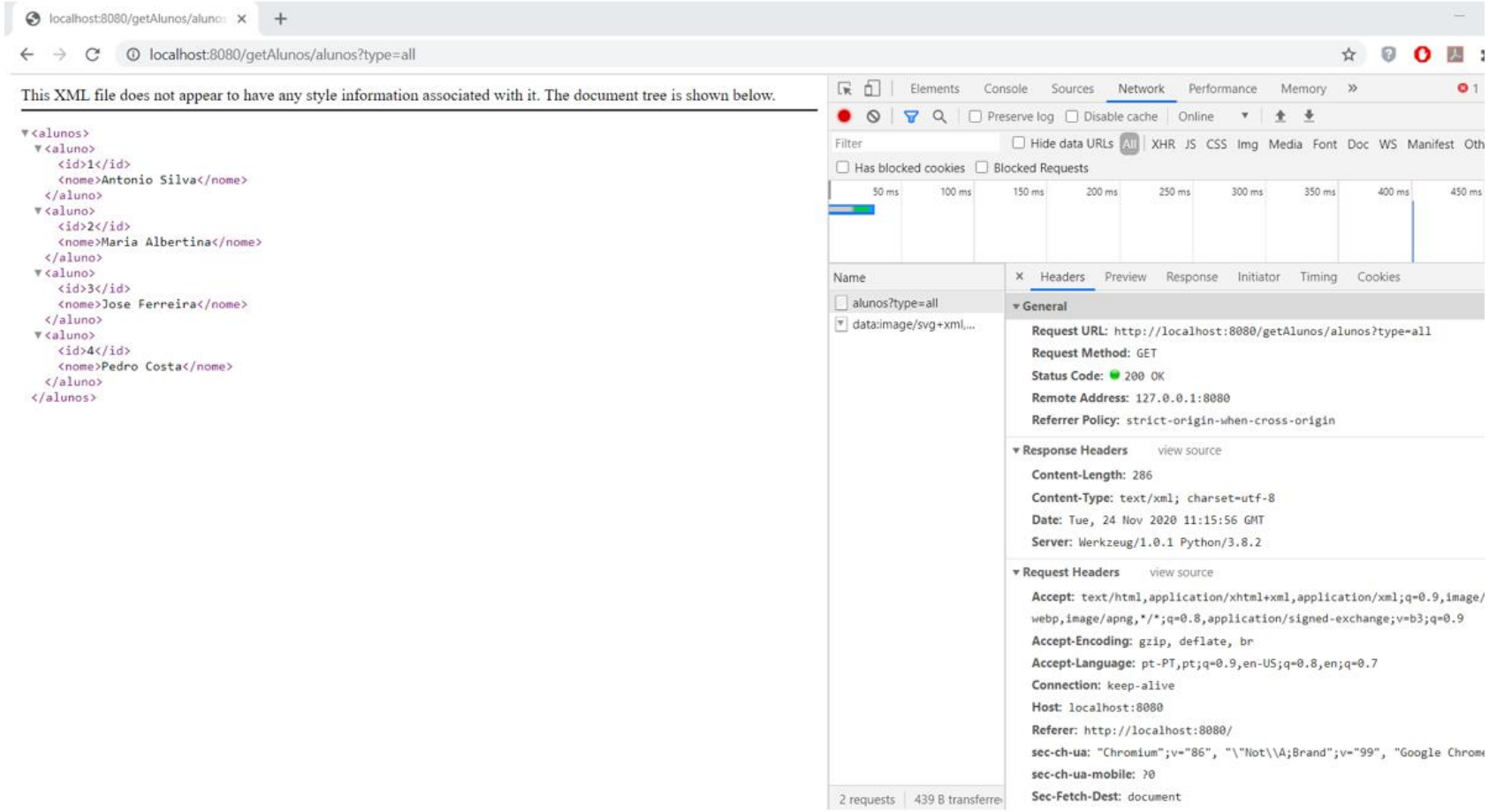
```
54  if __name__ == "__main__":
55      app.run(host='localhost', port=8080)
56
```



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

Resultado obtido chamando o serviço web:



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.

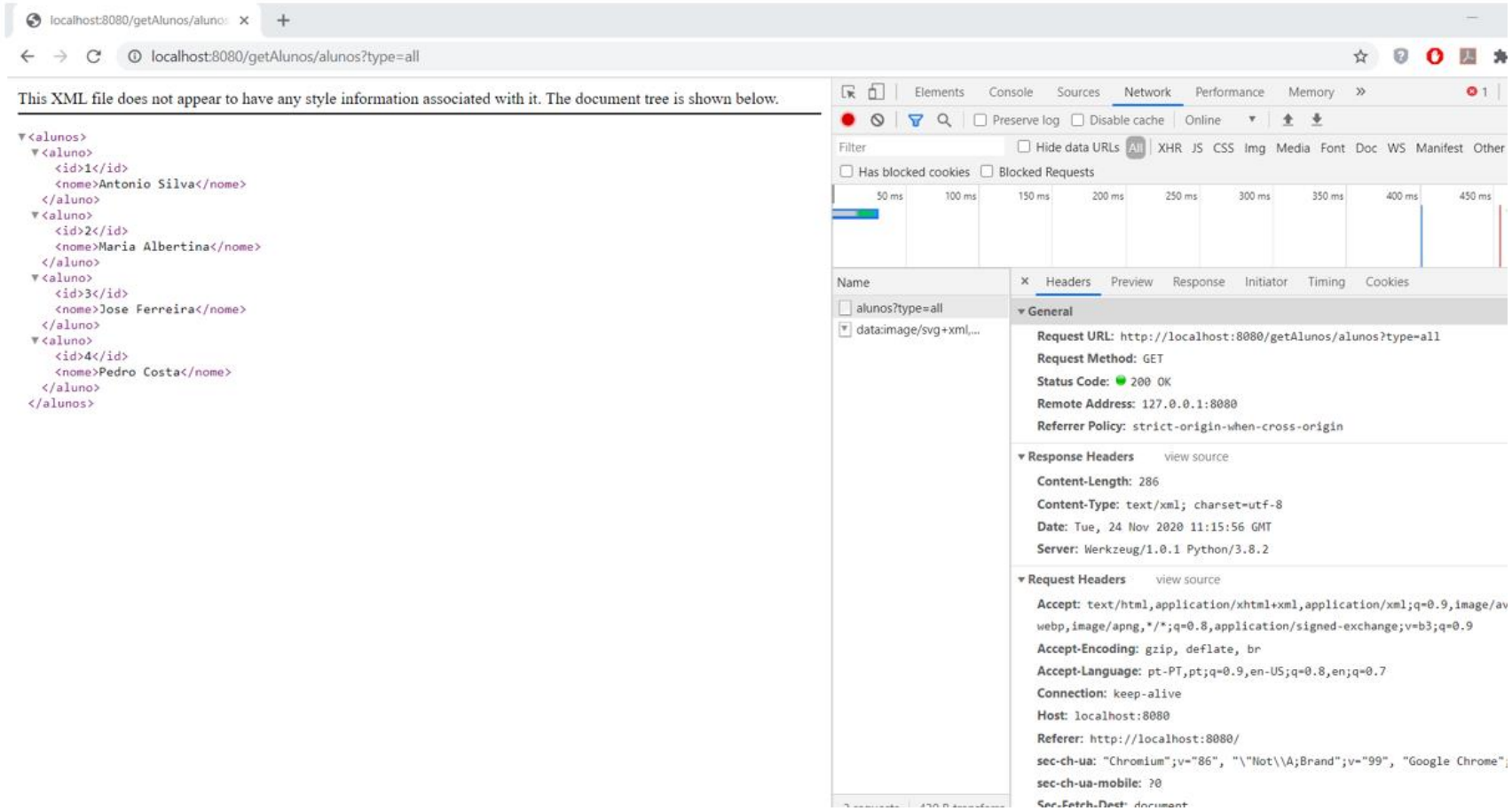
SoapXmlService.py | Client.py

```
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            return string_xml
50
51
52    application = Application([SoapXmlService],
53                              tns='soap.alunos.db',
54                              in_protocol=Soap11(validator='lxml'),
55                              out_protocol=Soap11()
56                              )
```

```
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            return string_xml
50
51
52    application = Application([SoapXmlService],
53                              tns='soap.alunos.db',
54                              in_protocol=Soap11(validator='lxml'),
55                              out_protocol=Soap11()
56                              )
```

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.

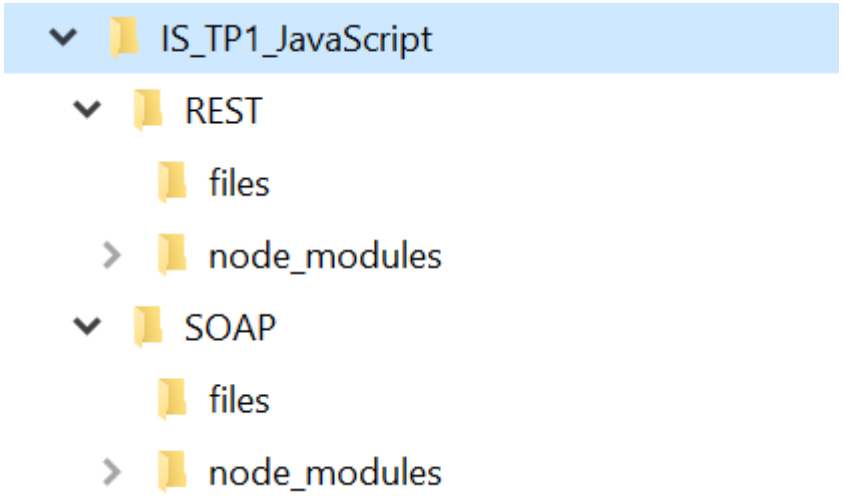


4. Interligação via WebServices e output em XML de leitura de Tabela de uma base de dados implementado noutras linguagens de programação.

```
 * 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

- files
- node_modules
- index.js
- package-lock.json

REST › files

- books.xml

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

index.js

```
var express = require('express');
var app = express();
var fs = require("fs");
var parser = require('xml2json-light');
var xpath = require('xpath')
var dom = require('xmldom').DOMParser
const http = require('http');
const url = require('url');

app.get('/booksParse', function (req, res) {
   const queryObject = url.parse(req.url, true).query;
   if (queryObject["file"] == "books") {
      fs.readFile("files/books.xml", 'utf8', function (err, data) {
         var json = parser.xml2json(data);
         var i = 0;
         var dados = "";
         for (i = 0; i < json.catalog.book.length; i++) {
            var book = json.catalog.book[i]
            dados = dados + `Id: ${book.id}\n`
            dados = dados + `Autor: ${book.author}\n`
            dados = dados + `Título: ${book.title}\n`
            dados = dados + `Genero: ${book.genre}\n`
            dados = dados + `Preço: ${book.price}\n`
            dados = dados + `Data de Publicação: ${book.publish_date}\n`
            dados = dados + `Descrição: ${book.description}\n\n`
         }

         res.setHeader("Content-Type", "text/plain; charset=utf-8");
         res.end(dados);
      });
   }
   else {
      res.setHeader("Content-Type", "text/plain; charset=utf-8");
      res.end("O URL não contém o nome do ficheiro XML para a realização do parse!\n\n Pedido Inválido!");
   }

})
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

index.js

```
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      }
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      }
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

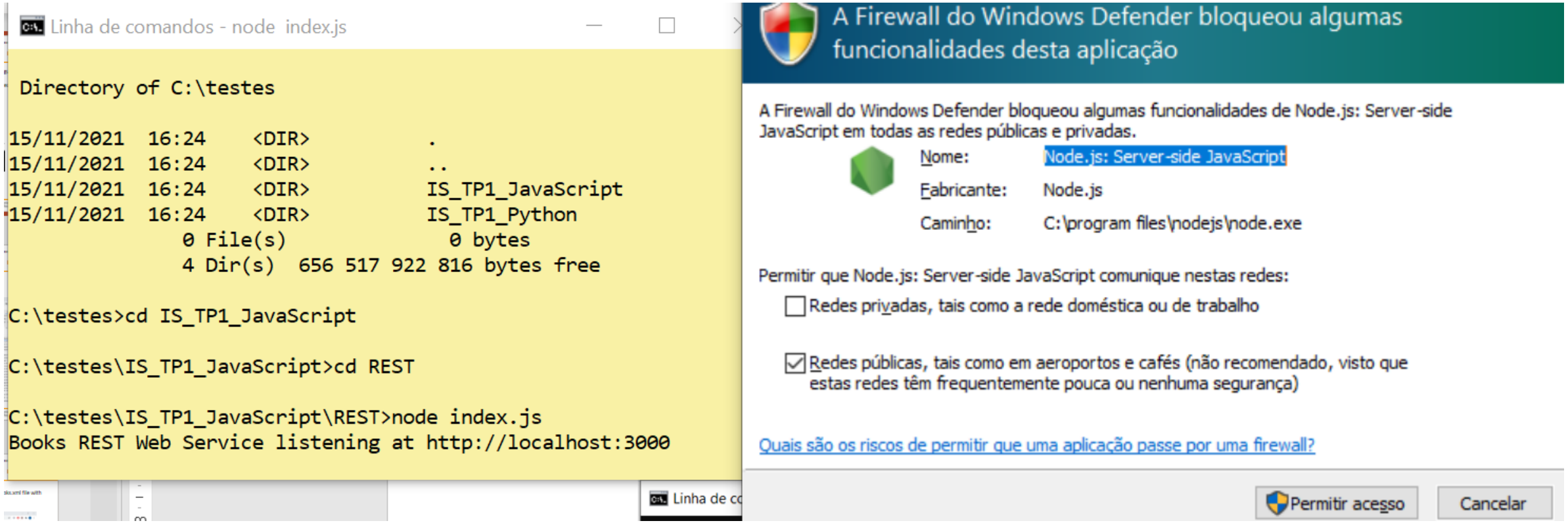
index.js

```
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  })
```

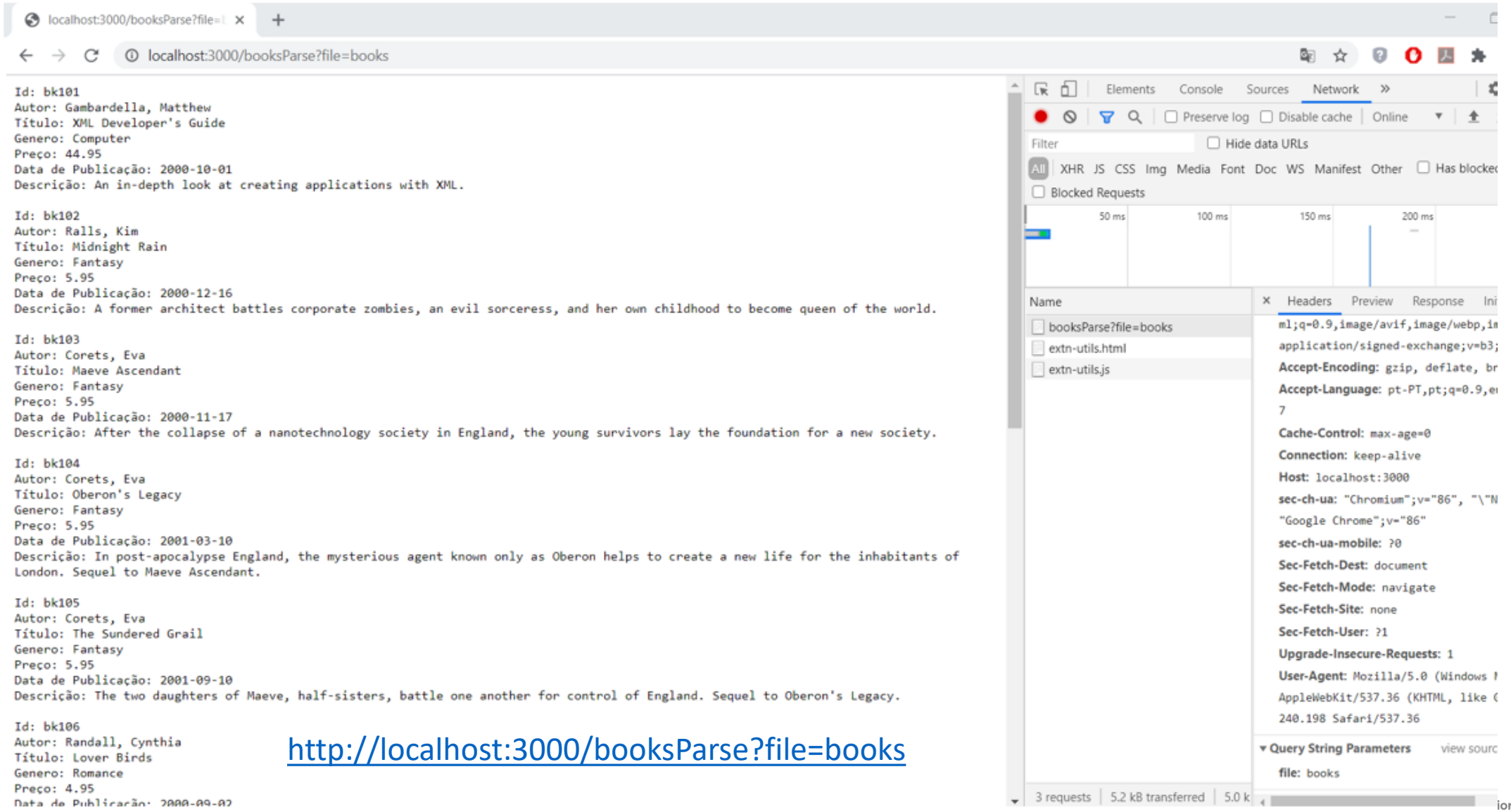
IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

package-lock.json


```
    "integrity": "sha1-sr9O6FFKrmFltIF4KdIbLvSZBOw="
  },
  "utf8": {
    "version": "3.0.0",
    "resolved": "https://registry.npmjs.org/utf8/-/utf8-3.0.0.tgz",
    "integrity": "sha512-E8VjFIQ/TyQgp+TZfS6l8yp/xWppSAHzidGiRrqe4bK4XP9pTRyKFgGJpO3SN7zdX4DeomTrwaseCHovfpFcqQ=="
  },
  "utils-merge": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/utils-merge/-/utils-merge-1.0.1.tgz",
    "integrity": "sha1-n5VxD1CiZ5R7LMwSR0HBAoQn5xM="
  },
  "vary": {
    "version": "1.1.2",
    "resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",
    "integrity": "sha1-IpnwLG3tMNSllhsLn3RSShj2NPw="
  },
  "xml2json-light": {
    "version": "1.0.6",
    "resolved": "https://registry.npmjs.org/xml2json-light/-/xml2json-light-1.0.6.tgz",
    "integrity": "sha1-ERaHcQorAfvS/hj0GK31Ed5UfzQ="
  },
  "xmldom": {
    "version": "0.4.0",
    "resolved": "https://registry.npmjs.org/xmldom/-/xmldom-0.4.0.tgz",
    "integrity": "sha512-2E93k08T30Ugs+34HBSTQLVtpi6mCddaY8uO+pMNk1pqSjV5vElzn4mmh6KLxN3hki8rNcHSYzILoh3TEWORvA=="
  },
  "xpath": {
    "version": "0.0.32",
    "resolved": "https://registry.npmjs.org/xpath/-/xpath-0.0.32.tgz",
    "integrity": "sha512-rxMJhSIoiO8vXcWvSifKqhvV96GjiD5wYb8/QHdoRyQvraTpp4IEv944nhGausZZ3u7dhQXteZuZbaqfpB7uYw=="
  }
```

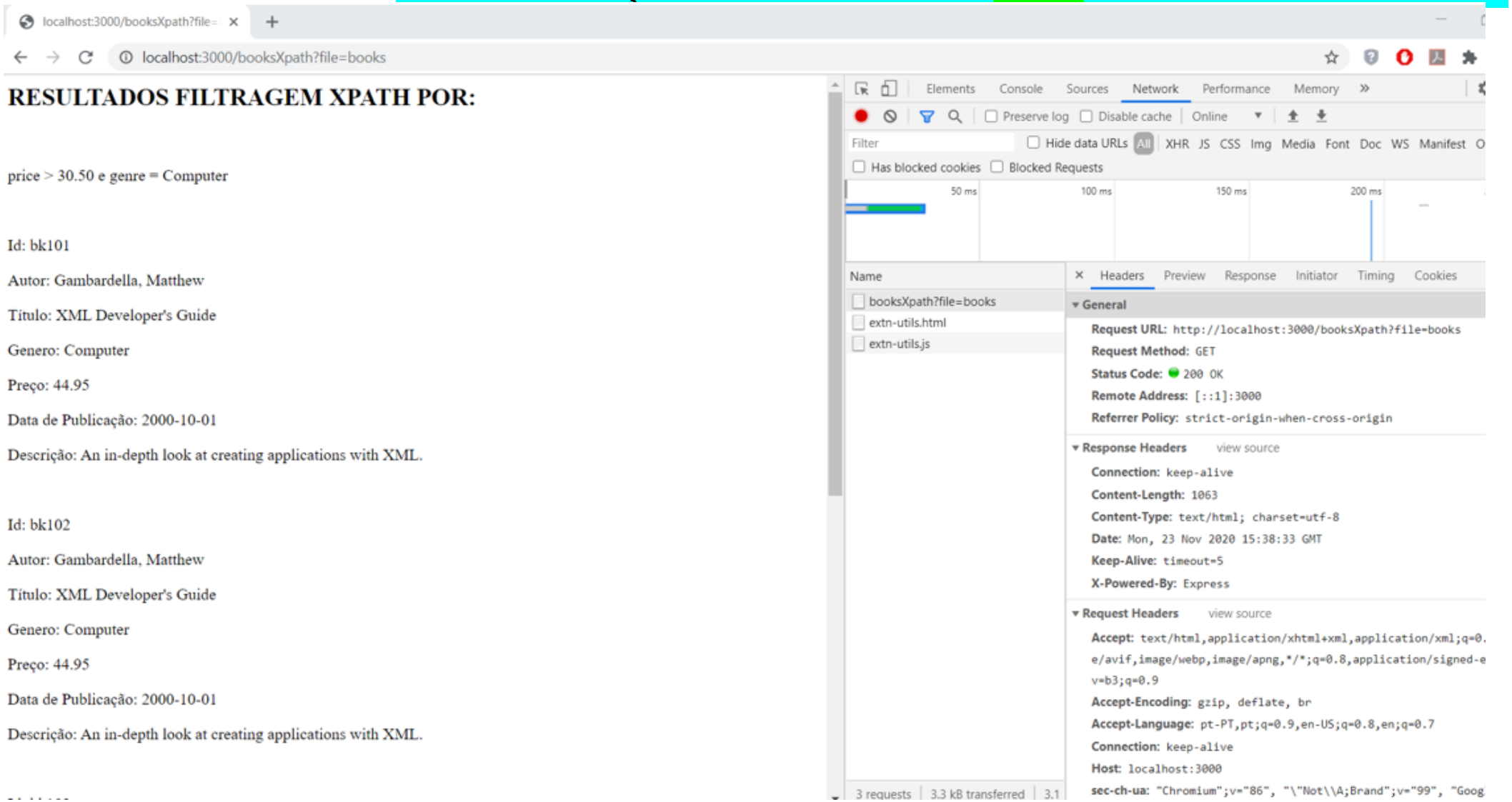

Linha de comandos - node index.js
Directory of C:\testes
15/11/2021 16:24 <DIR> .
15/11/2021 16:24 <DIR> ..
15/11/2021 16:24 <DIR> IS_TP1_JavaScript
15/11/2021 16:24 <DIR> IS_TP1_Python
0 File(s) 0 bytes
4 Dir(s) 656 517 922 816 bytes free
C:\testes>cd IS_TP1_JavaScript
C:\testes\IS_TP1_JavaScript>cd REST
C:\testes\IS_TP1_JavaScript\REST>node index.js
Books REST Web Service listening at http://localhost:3000
A Firewall do Windows Defender bloqueou algumas funcionalidades desta aplicação
A Firewall do Windows Defender bloqueou algumas funcionalidades de Node.js: Server-side JavaScript em todas as redes públicas e privadas.
Nome: Node.js: Server-side JavaScript
Fabricante: Node.js
Caminho: C:\program files\nodejs\node.exe
Permitir que Node.js: Server-side JavaScript comunique nestas redes:
Redes privadas, tais como a rede doméstica ou de trabalho
Redes públicas, tais como em aeroportos e cafés (não recomendado, visto que estas redes têm frequentemente pouca ou nenhuma segurança)
Quais são os riscos de permitir que uma aplicação passe por uma firewall?
Permitir acesso
Cancelar

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:



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

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:



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

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

index.js

```
var soap = require('soap');
var express = require('express');
var fs = require('fs');
var parser = require('xml2json-light');
var xpath = require('xpath')
var dom = require('xmldom').DOMParser

function books_parse(args) {
  if(args.type == "parse") {
    var data = fs.readFileSync( "files/books.xml", 'utf8');
    var json = parser.xml2json(data);
    var dados = "<h2>PARSE DO FICHEIRO - books.xml</h2><br>";
    var i = 0;
    for(i = 0; i < json.catalog.book.length; i++) {
      var book = json.catalog.book[i]
      dados = dados + `<p>Id: ${book.id}</p>`
      dados = dados + `<p>Autor: ${book.author}</p>`
      dados = dados + `<p>Título: ${book.title}</p>`
      dados = dados + `<p>Genero: ${book.genre}</p>`
      dados = dados + `<p>Preço: ${book.price}</p>`
      dados = dados + `<p>Data de Publicação: ${book.publish_date}</p>`
      dados = dados + `<p>Descrição: ${book.description}</p><br>`
    }
    return {
        result: dados
    }
  }
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

index.js

```
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  }
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

index.js

```
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:

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  -->
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:

Books_ws_wsdl.wsdl

```
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"/>
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:

Books_ws_wsdl.wsdl

```
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:

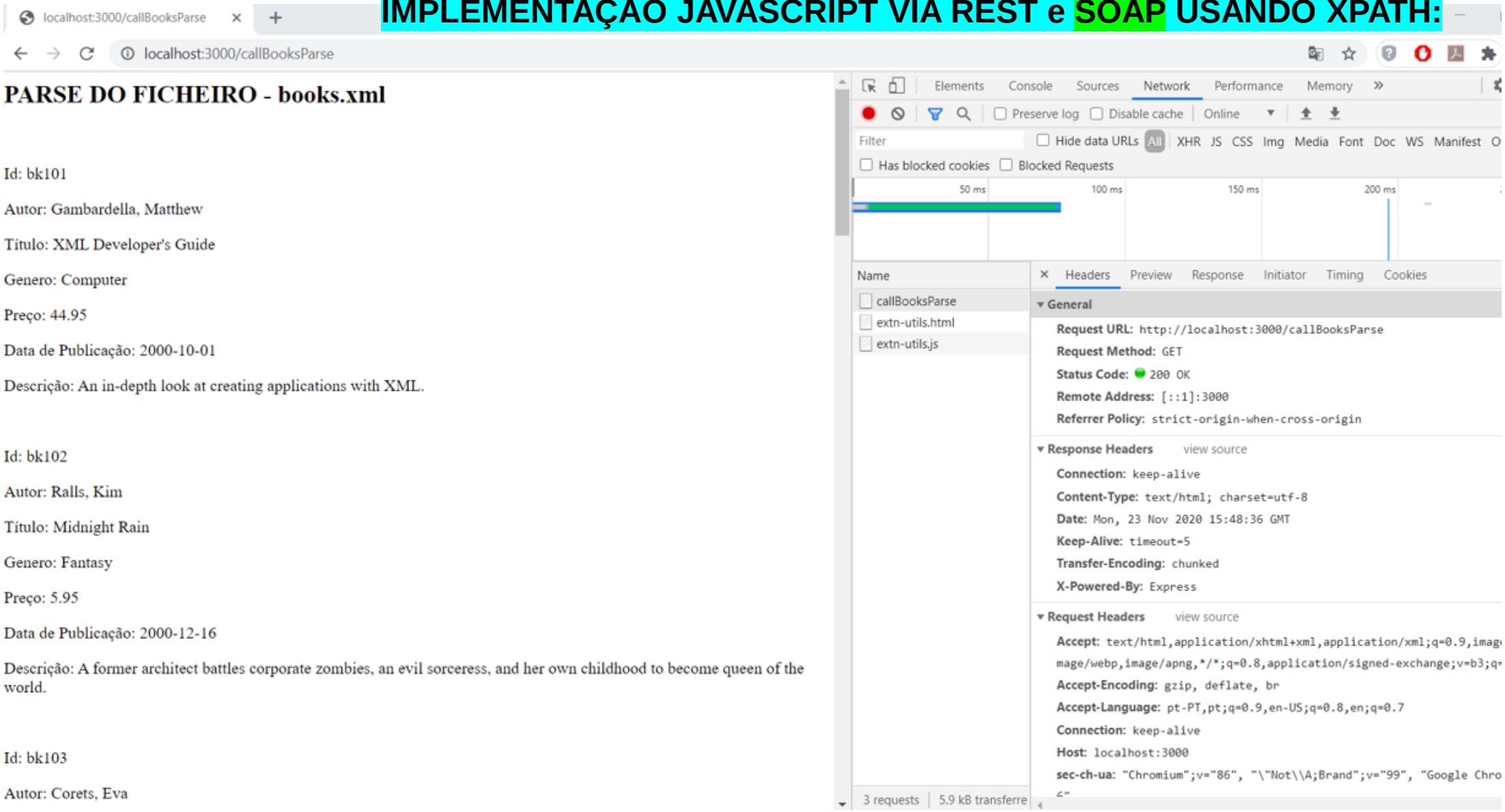
Linha de comandos - node index.js

```
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

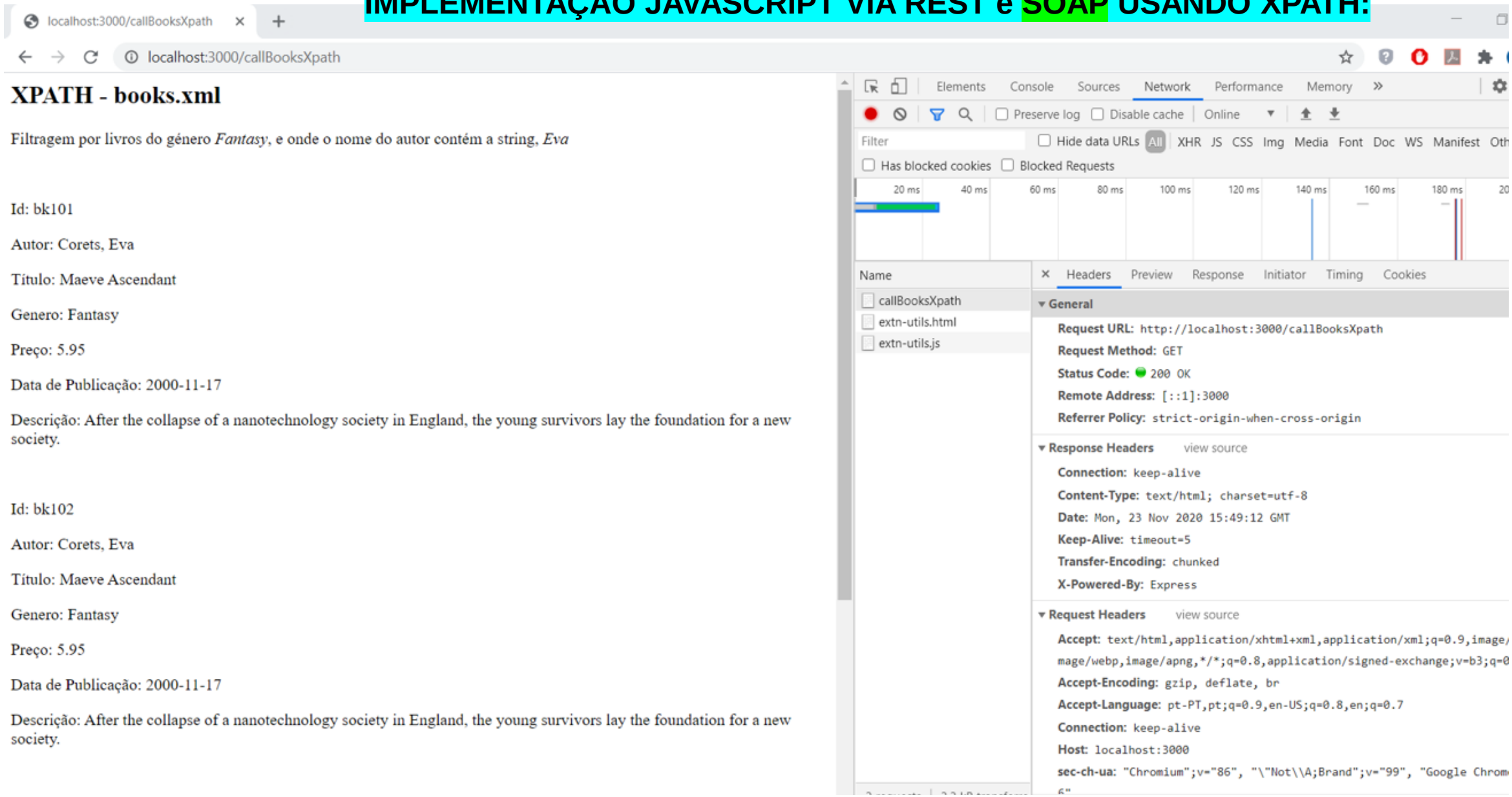
```
C:\Integração de Sistemas\TP1\ex5_codigo\JavaScript\SOAP>node client.js
Example app listening at http://localhost:3000
```

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:



Versão 1 – Documento em Atualização

IMPLEMENTAÇÃO JAVASCRIPT VIA REST e SOAP USANDO XPATH:



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

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

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

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

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

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

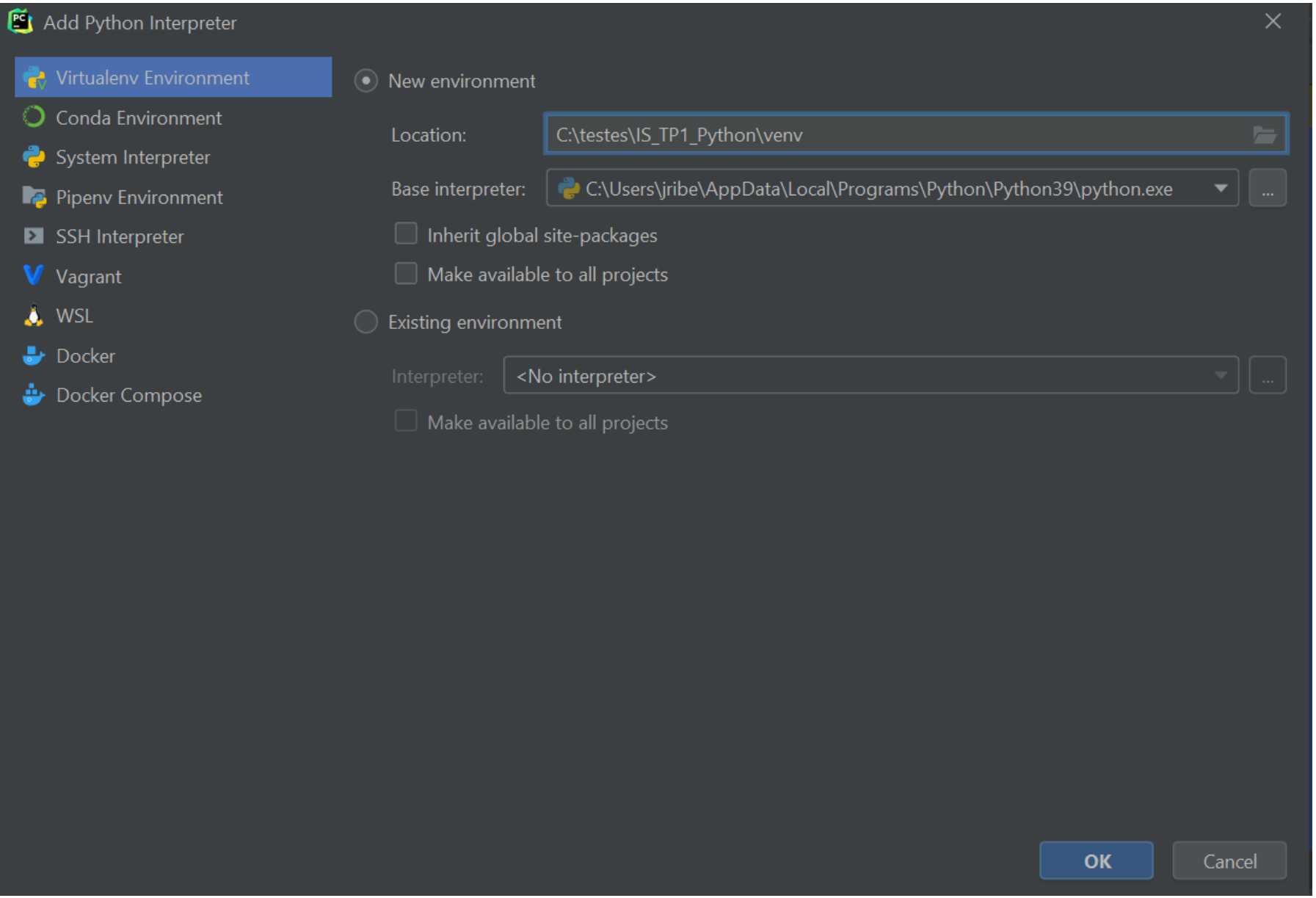
Books_ws_rest.py

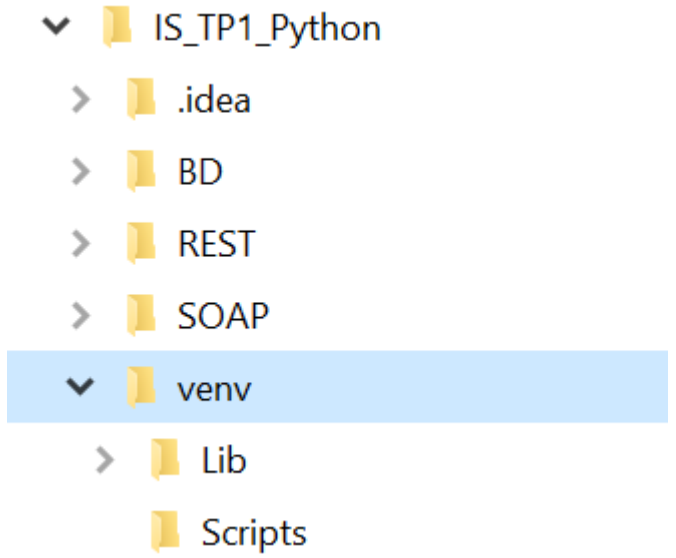
```python
from xml.dom import minidom
from flask import Flask, request
from lxml import etree
from werkzeug.wrappers import Response
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Esta aplicação contém os seguintes métodos: <h3>"
    html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
    html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/getBooks/books', methods=['GET'])
def getBooks():
    typeRequest = request.args.get('type')
    if typeRequest.upper() == "HTML":
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
            node = itemlist.item(i)
            autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
            titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
            genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
            preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
            data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
            descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
            data = data + "<h5>Autor: " + autor + "<h5>"
            data = data + "<h5>Título: " + titulo + "<h5>"
            data = data + "<h5>Genero: " + genero + "<h5>"
            data = data + "<h5>Preço: " + preco + "<h5>"
            data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
            data = data + "<h5>Descrição: " + descricao + "<h5>"
            data = data + "<br>"
```

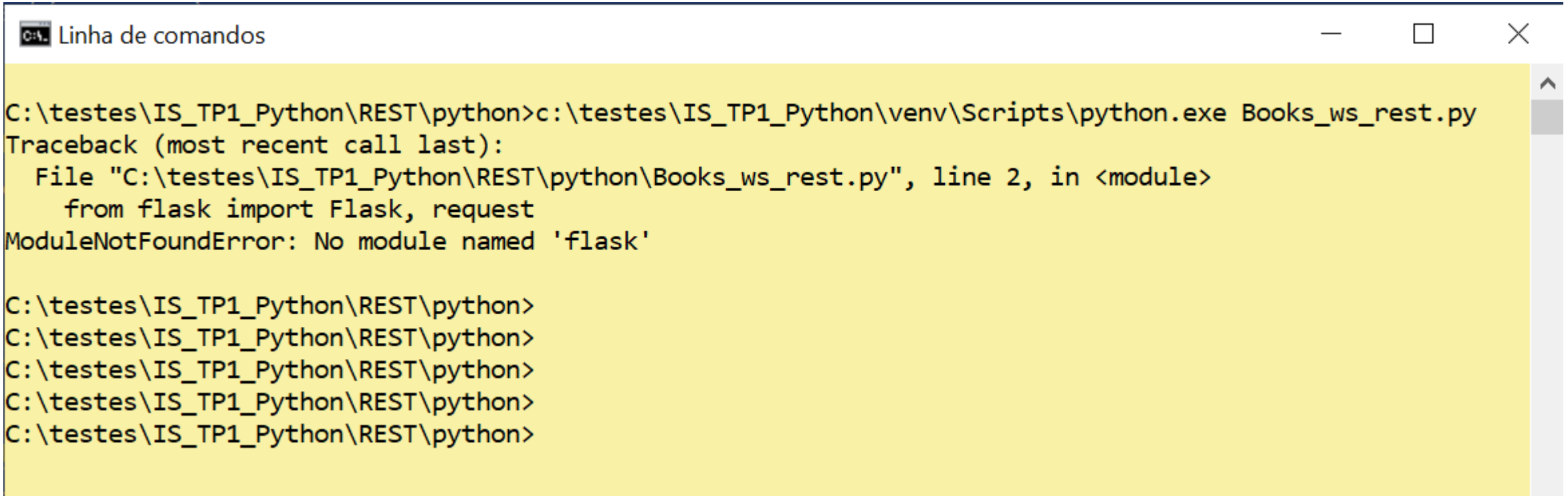
IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

Books_ws_rest.py

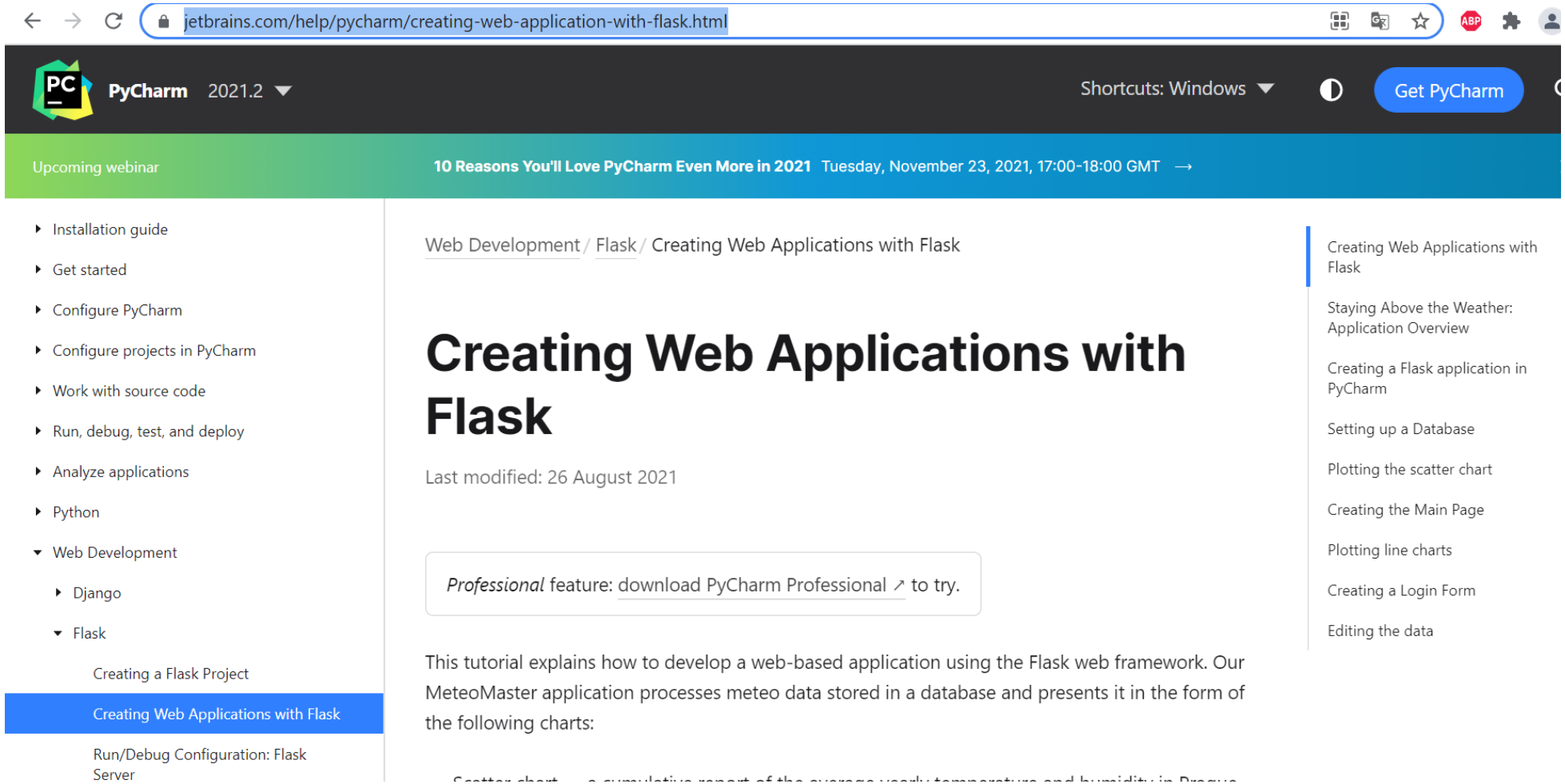
```
37              data = data + "<br>"
38          return data
39      if typeRequest.upper() == "XML":
40          root = etree.parse('../files/books.xml').getroot()
41          xml = etree.tostring(root, pretty_print=True)
42          return Response(xml, mimetype='text/xml')
43
44  @app.route('/getBooks/booksXpath', 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)
68
```

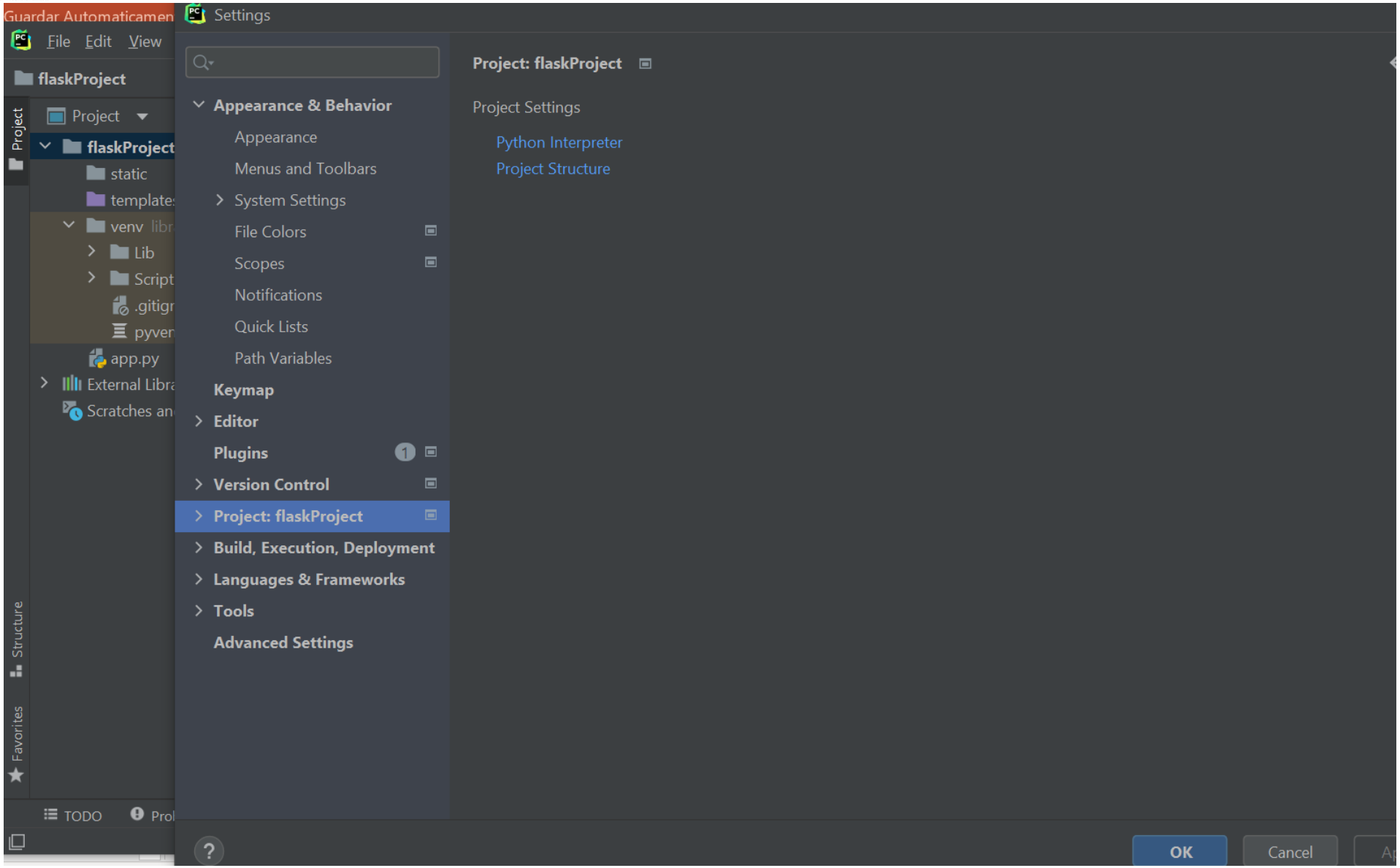

Add Python Interpreter
Virtualenv Environment
Conda Environment
System Interpreter
Pipenv Environment
SSH Interpreter
Vagrant
WSL
Docker
Docker Compose
New environment
Location:
C:\testes\IS_TP1_Python\venv
Base interpreter:
C:\Users\jribe\AppData\Local\Programs\Python\Python39\python.exe
Inherit global site-packages
Make available to all projects
Existing environment
Interpreter:
<No interpreter>
Make available to all projects
OK
Cancel

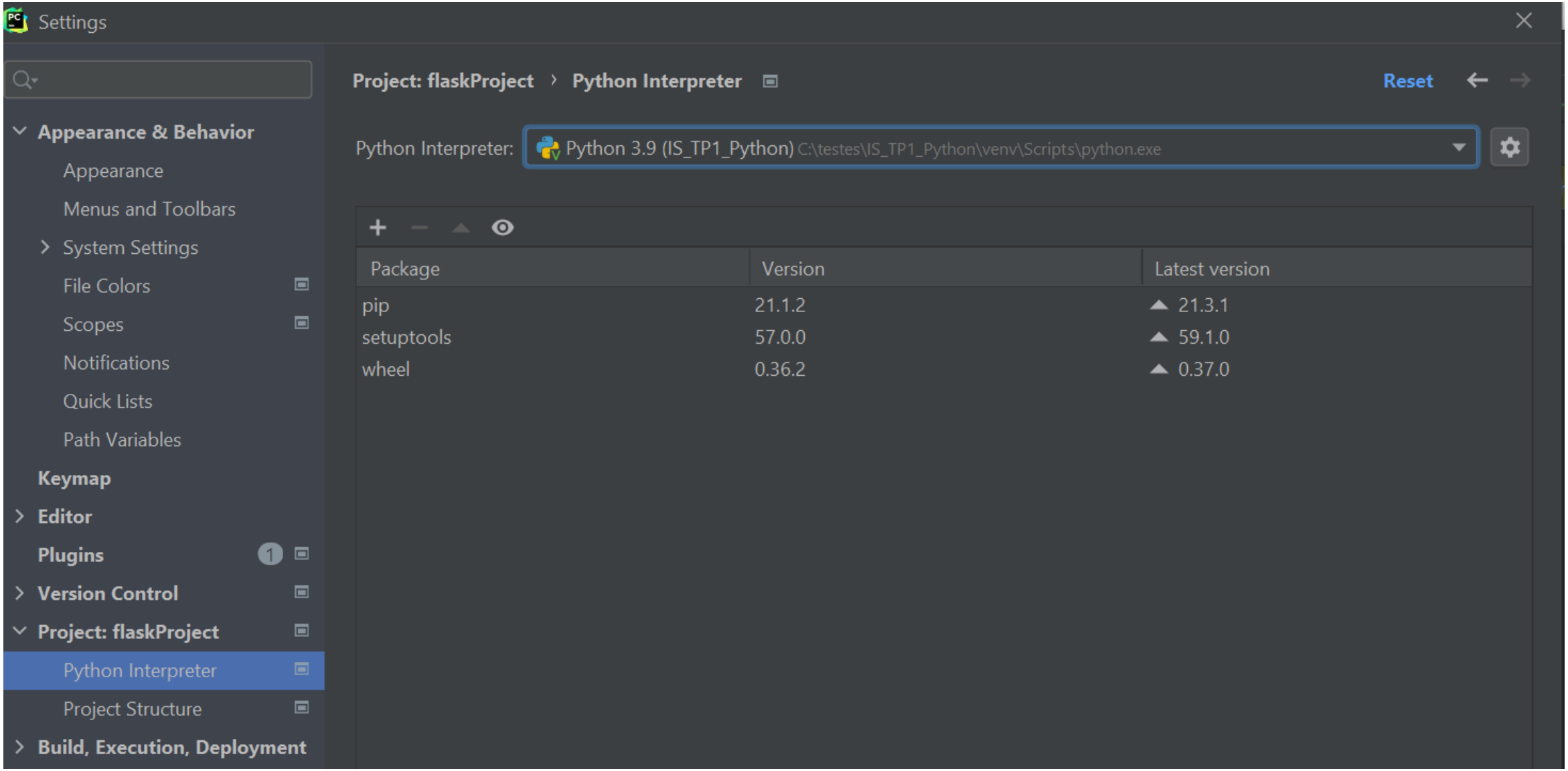

IS_TP1_Python
.idea
BD
REST
SOAP
venv
Lib
Scripts

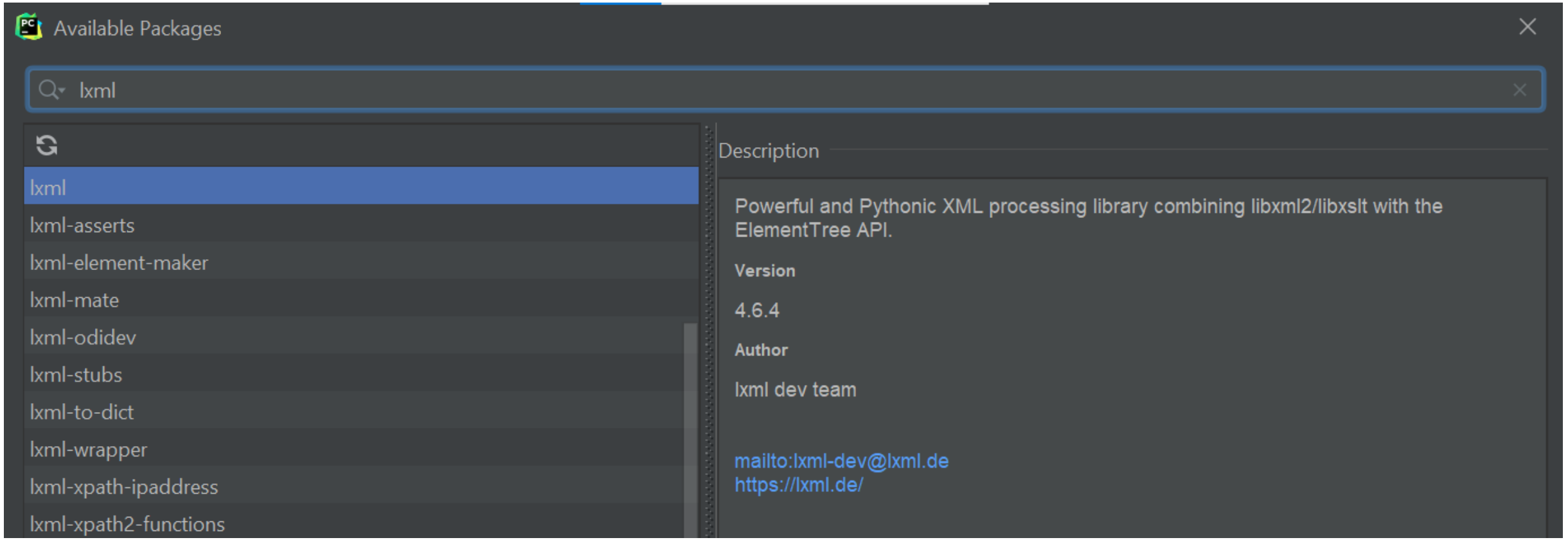

Linha de comandos
C:\testes\IS_TP1_Python\REST\python>c:\testes\IS_TP1_Python\venv\Scripts\python.exe Books_ws_rest.py
Traceback (most recent call last):
File "C:\testes\IS_TP1_Python\REST\python\Books_ws_rest.py", line 2, in <module>
from flask import Flask, request
ModuleNotFoundError: No module named 'flask'
C:\testes\IS_TP1_Python\REST\python>
C:\testes\IS_TP1_Python\REST\python>
C:\testes\IS_TP1_Python\REST\python>
C:\testes\IS_TP1_Python\REST\python>
C:\testes\IS_TP1_Python\REST\python>

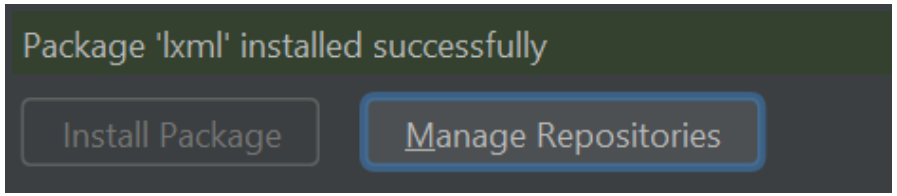
https://www.jetbrains.com/help/pycharm/creating-web-application-with-flask.html

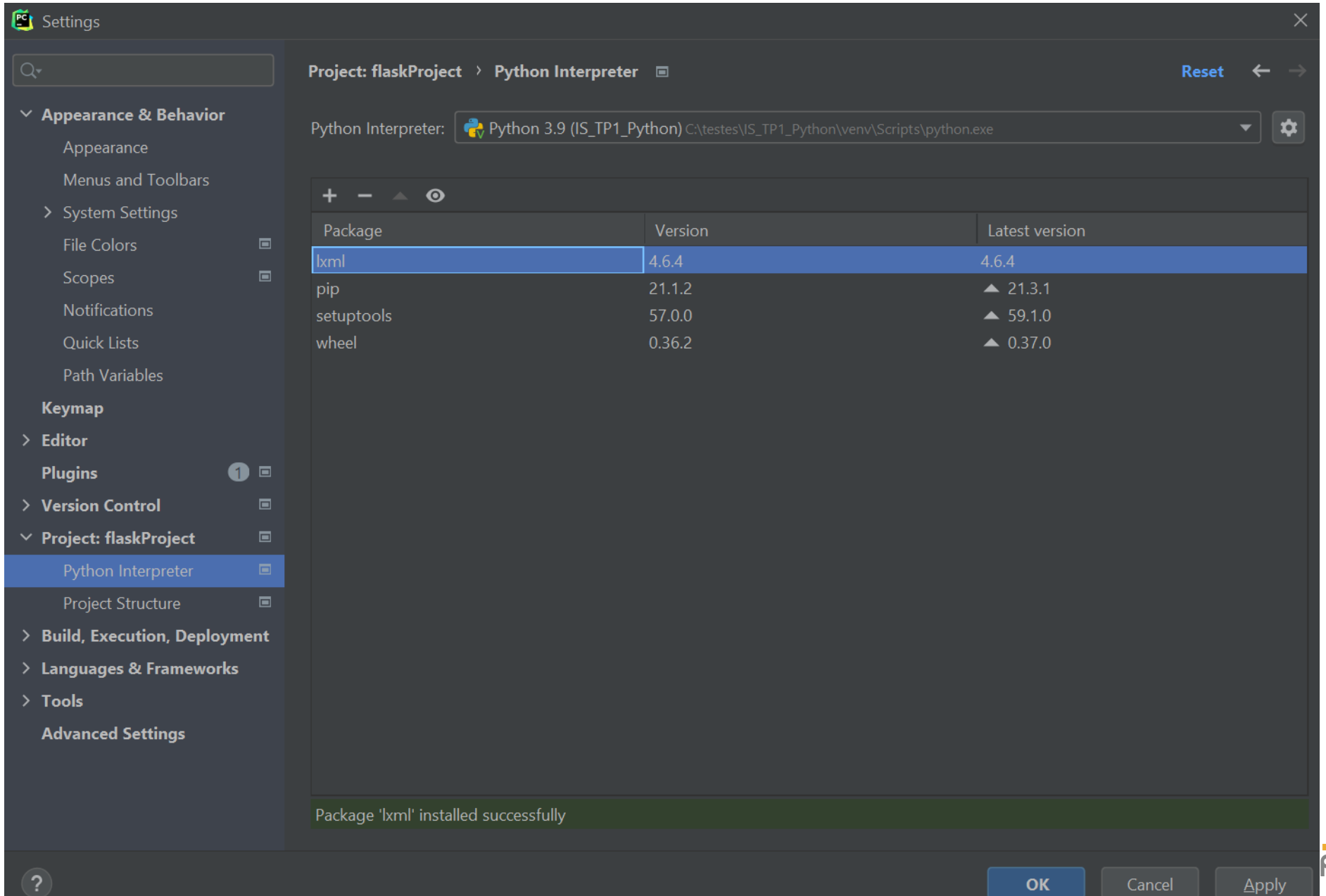


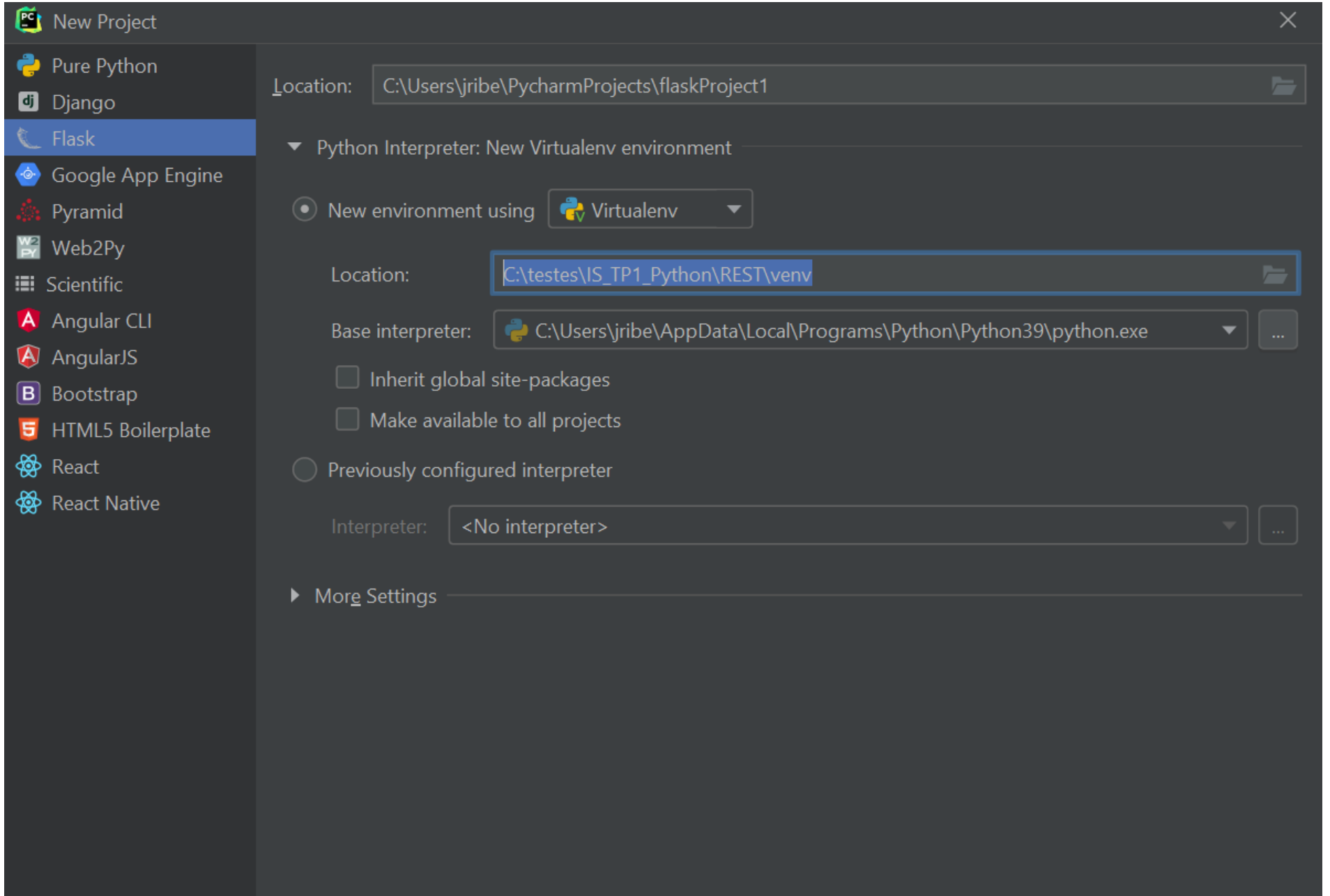

Guardar Automaticamen
File Edit View
flaskProject
Project
flaskProject
static
templates
venv
Lib
Script
.gitign
pyven
app.py
External Libra
Scratches an
Structure
Favorites
TODO
Settings
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Project: flaskProject
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Project: flaskProject
Project Settings
Python Interpreter
Project Structure
OK
Cancel

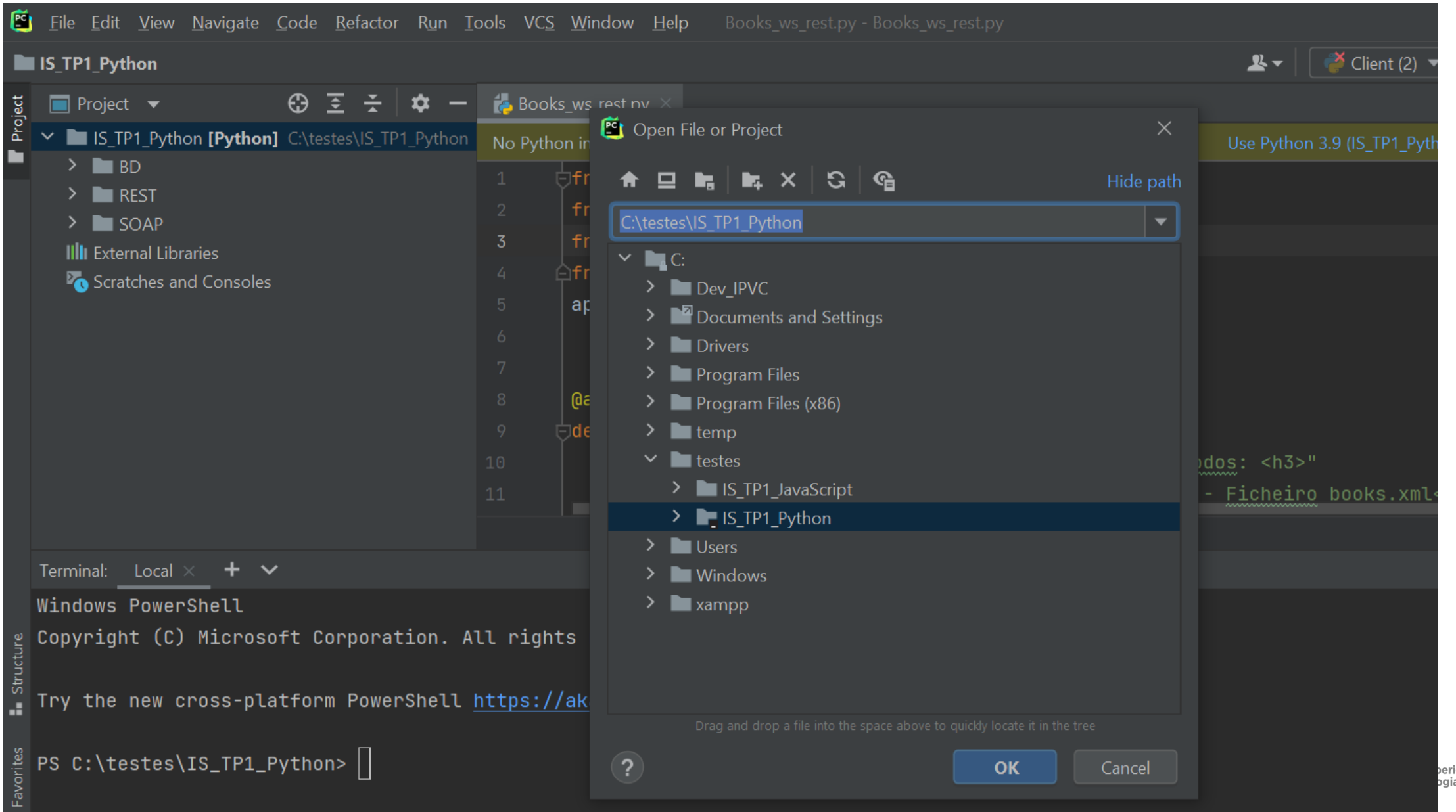

Settings
Project: flaskProject › Python Interpreter
Reset
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Project: flaskProject
Python Interpreter
Project Structure
Build, Execution, Deployment
Python Interpreter: Python 3.9 (IS_TP1_Python) C:\testes\IS_TP1_Python\venv\Scripts\python.exe
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

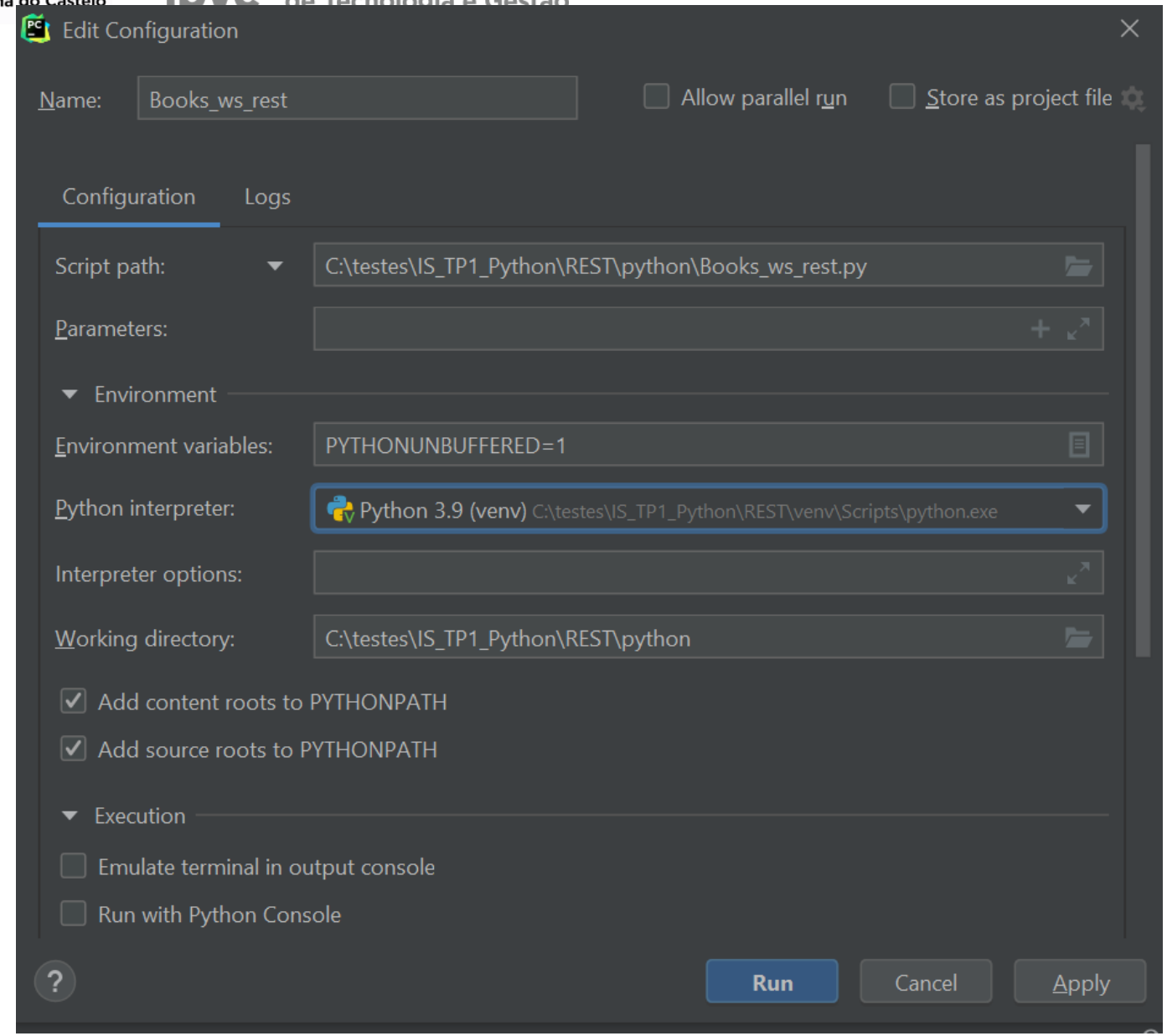

Available Packages
lxml
lxml
lxml-asserts
lxml-element-maker
lxml-mate
lxml-odidev
lxml-stubs
lxml-to-dict
lxml-wrapper
lxml-xpath-ipaddress
lxml-xpath2-functions
Description
Powerful and Pythonic XML processing library combining libxml2/libxslt with the ElementTree API.
Version
4.6.4
Author
lxml dev team
mailto:lxml-dev@lxml.de
https://lxml.de/

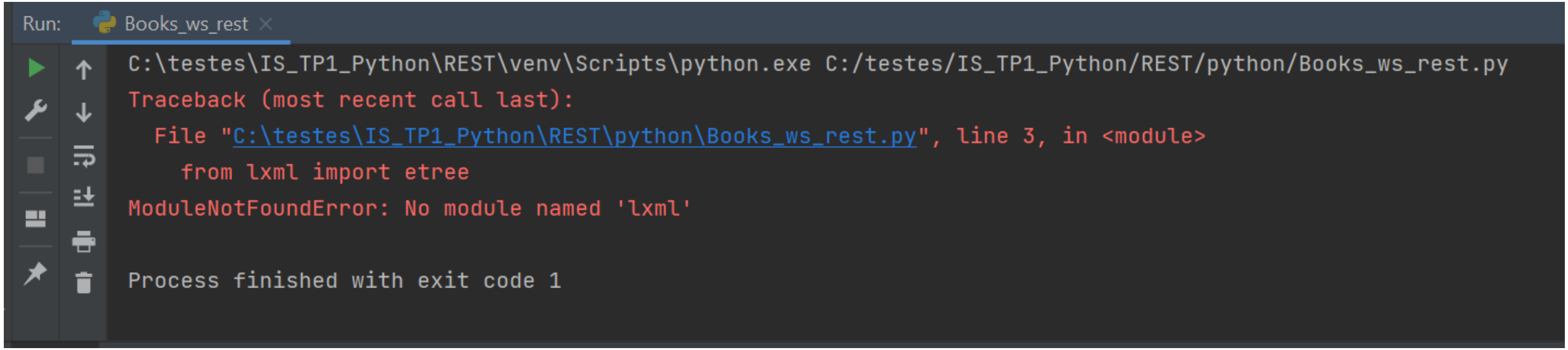

Package 'lxml' installed successfully
Install Package
Manage Repositories


Settings
Project: flaskProject › Python Interpreter
Reset
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
Version Control
Project: flaskProject
Python Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
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
Package 'lxml' installed successfully
OK
Cancel
Apply


New Project
Pure Python
Django
Flask
Google App Engine
Pyramid
Web2Py
Scientific
Angular CLI
AngularJS
Bootstrap
HTML5 Boilerplate
React
React Native
Location: C:\Users\jribe\PycharmProjects\flaskProject1
Python Interpreter: New Virtualenv environment
New environment using Virtualenv
Location: C:\testes\IS_TP1_Python\REST\venv
Base interpreter: C:\Users\jribe\AppData\Local\Programs\Python\Python39\python.exe
Inherit global site-packages
Make available to all projects
Previously configured interpreter
Interpreter: <No interpreter>
More Settings


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Books_ws_rest.py - Books_ws_rest.py
IS_TP1_Python
Client (2)
Project
IS_TP1_Python [Python] C:\testes\IS_TP1_Python
BD
REST
SOAP
External Libraries
Scratches and Consoles
Books_ws_rest.py
Use Python 3.9 (IS_TP1_Pyth
Open File or Project
Hide path
C:\testes\IS_TP1_Python
C:
Dev_IPVC
Documents and Settings
Drivers
Program Files
Program Files (x86)
temp
testes
IS_TP1_JavaScript
IS_TP1_Python
Users
Windows
xampp
Drag and drop a file into the space above to quickly locate it in the tree
OK
Cancel
Terminal: Local
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights
Try the new cross-platform PowerShell https://ak
PS C:\testes\IS_TP1_Python>
Structure
Favorites


Edit Configuration
Name: Books_ws_rest
Allow parallel run
Store as project file
Configuration
Logs
Script path: C:\testes\IS_TP1_Python\REST\python\Books_ws_rest.py
Parameters:
Environment
Environment variables: PYTHONUNBUFFERED=1
Python interpreter: Python 3.9 (venv) C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe
Interpreter options:
Working directory: C:\testes\IS_TP1_Python\REST\python
Add content roots to PYTHONPATH
Add source roots to PYTHONPATH
Execution
Emulate terminal in output console
Run with Python Console
Run
Cancel
Apply


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

https://blog.finxter.com/how-to-install-lxml-in-python/

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:

codetd.com/en/article/12931460

CodeTD Home Mobile Internet Server Language Enterprise Database News Others Search

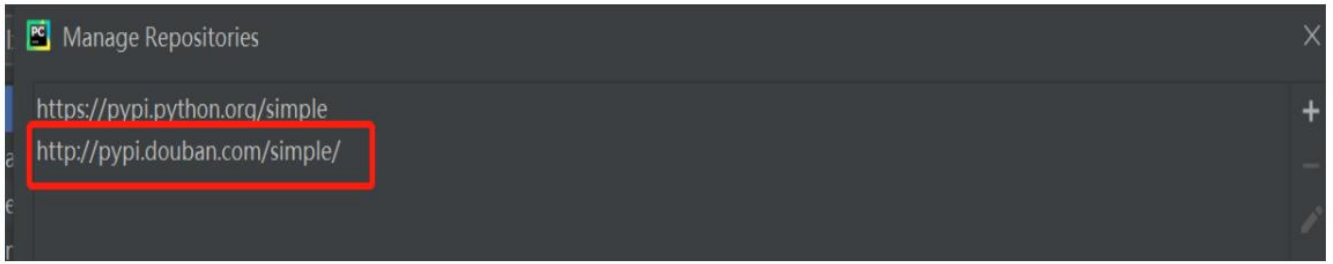
Install lxml in PyCharm

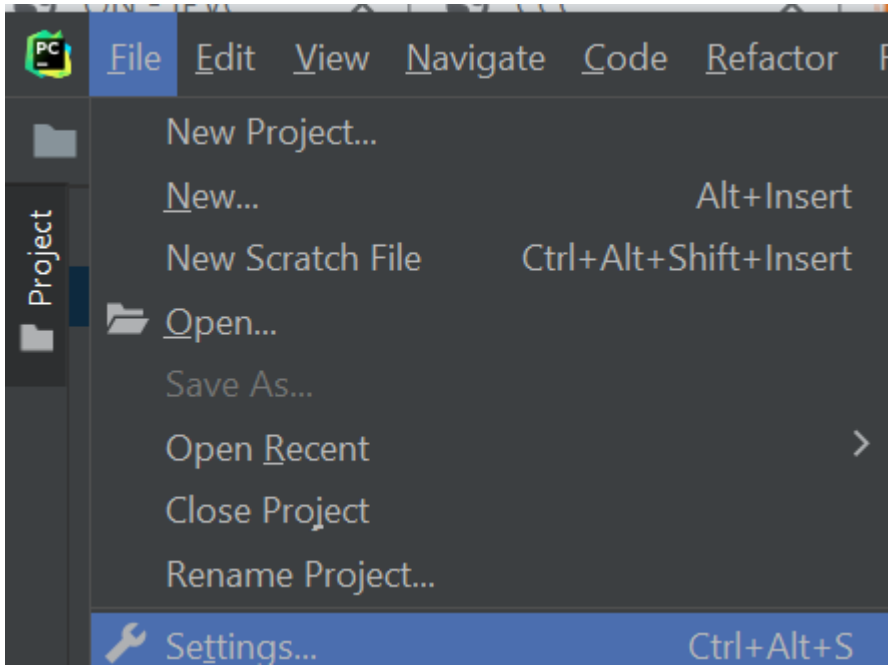
Others 2021-03-28 03:11:13 views: null

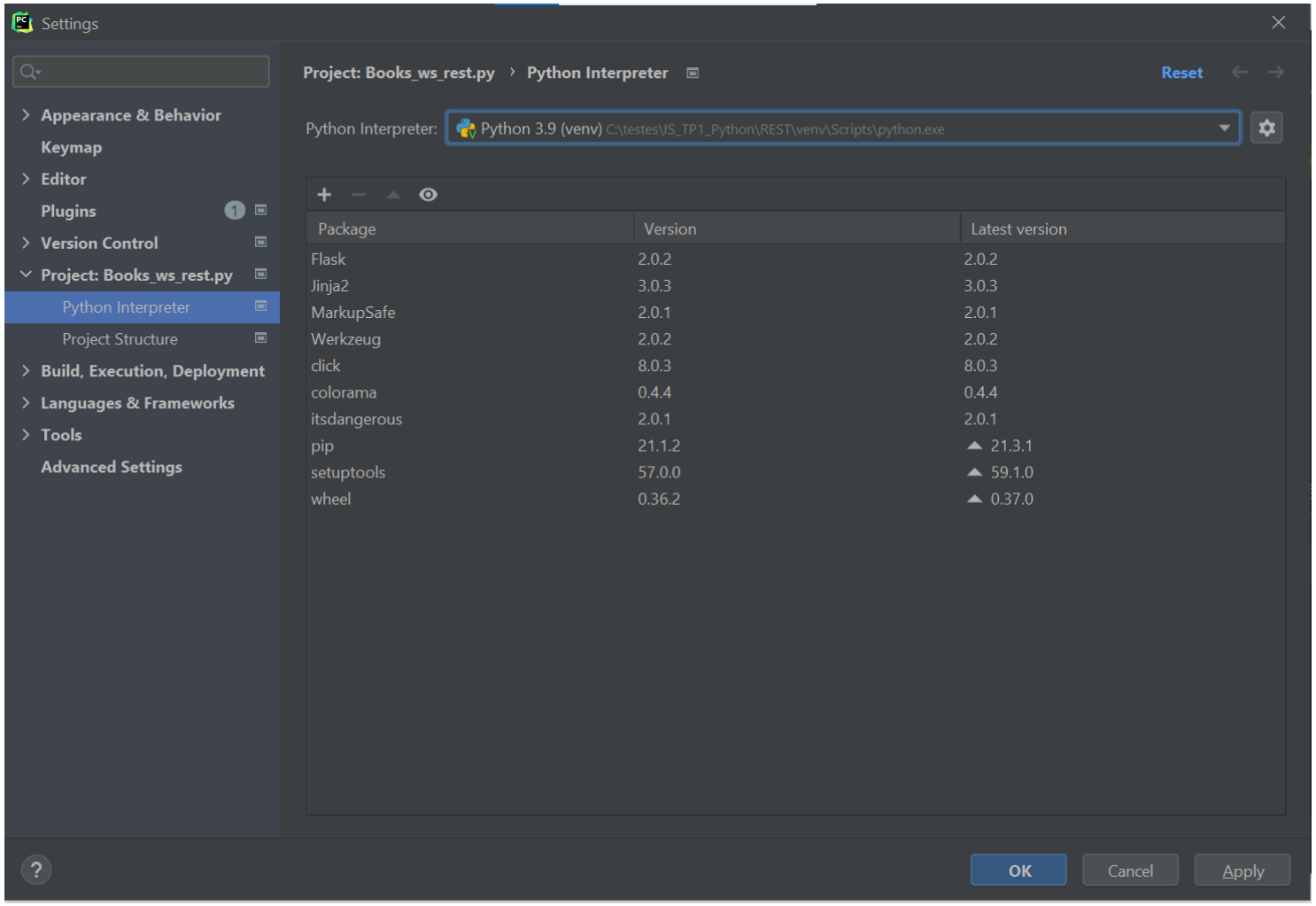
The point of doubt is that lxml is successfully installed in git cmd, but it cannot be used in PyCram, and an error is reported.

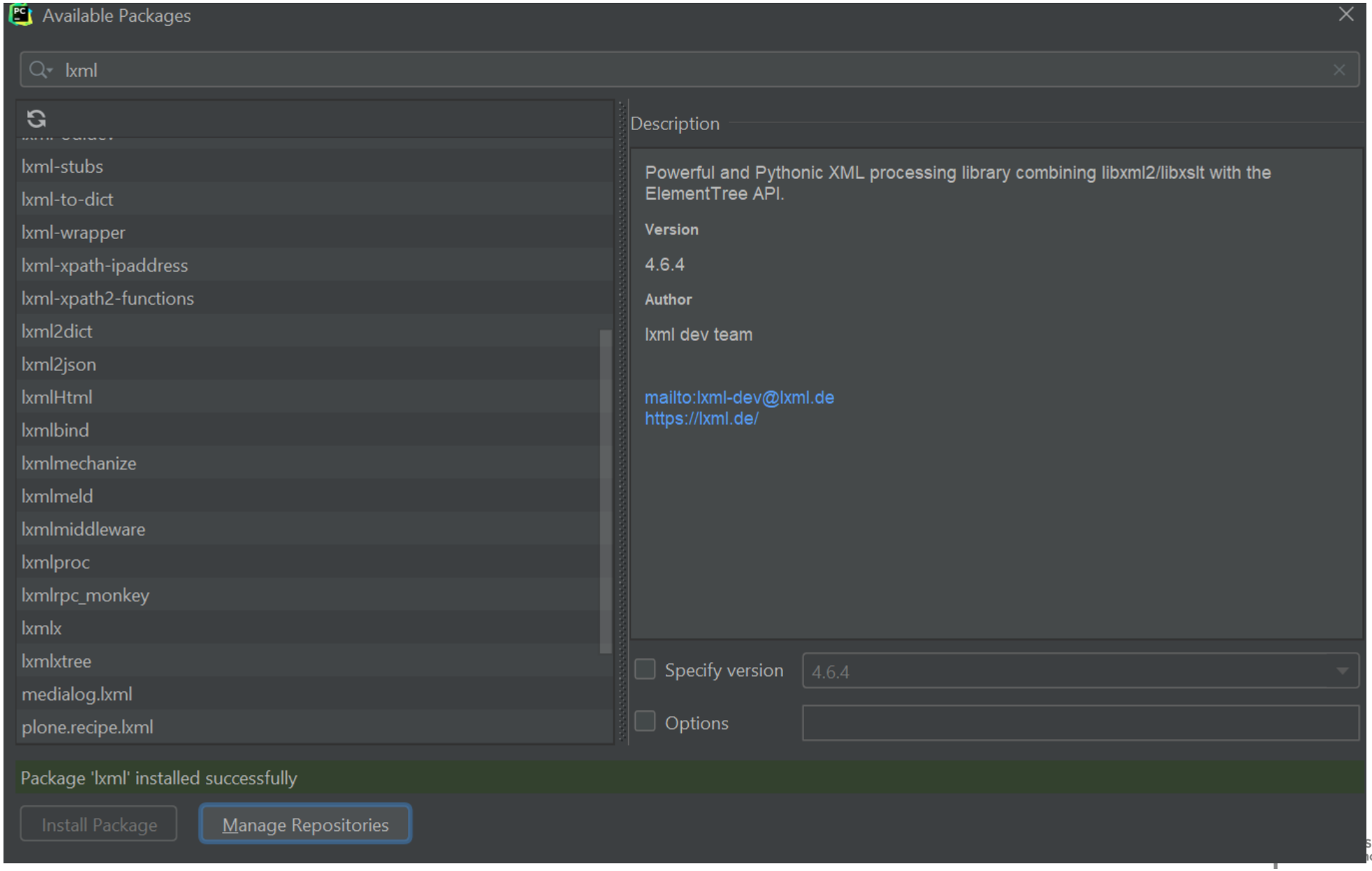
ModuleNotFoundError: No module named 'lxml'

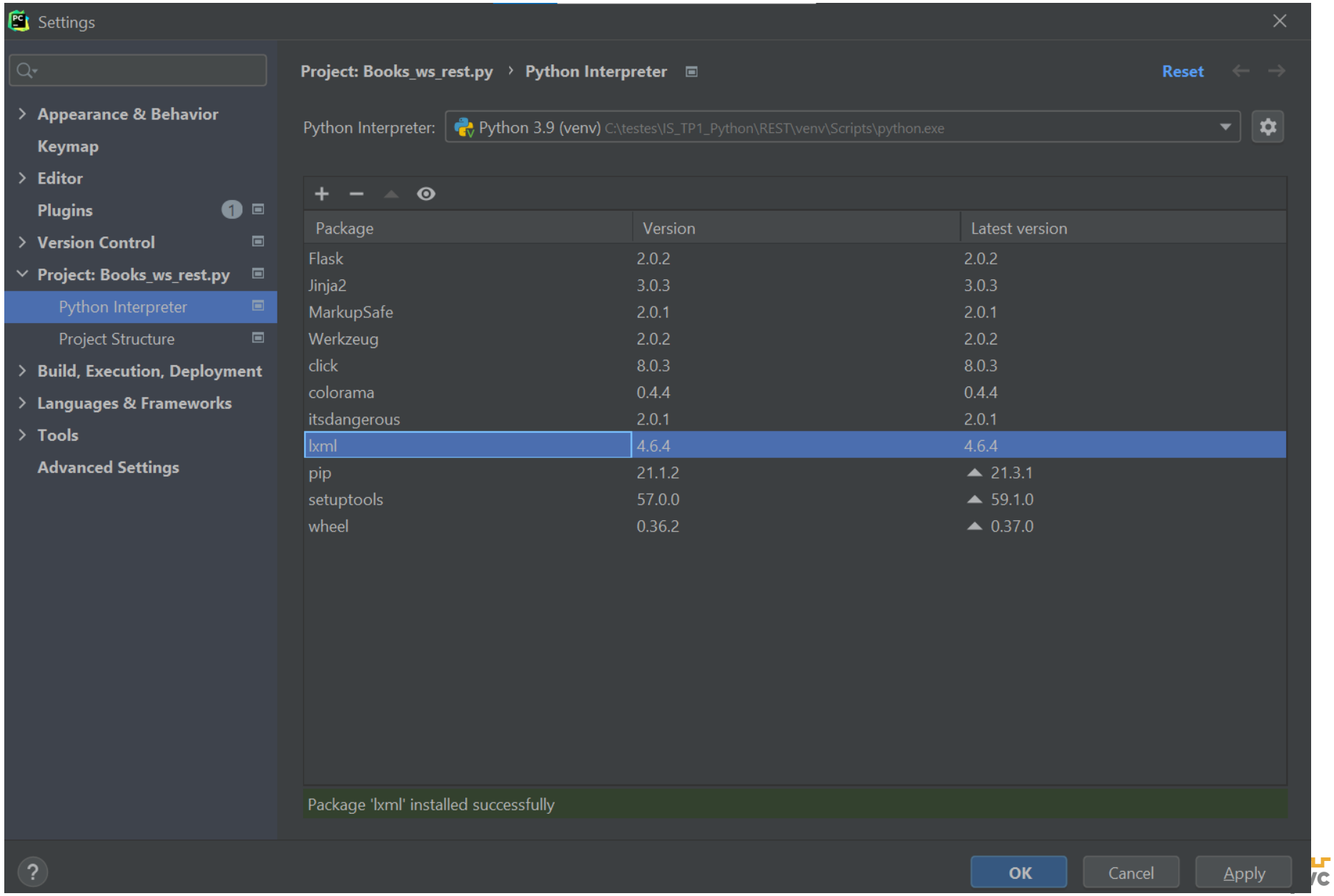
In git cmd, it was successfully installed using pip install lxml -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com, so choose new website resources,

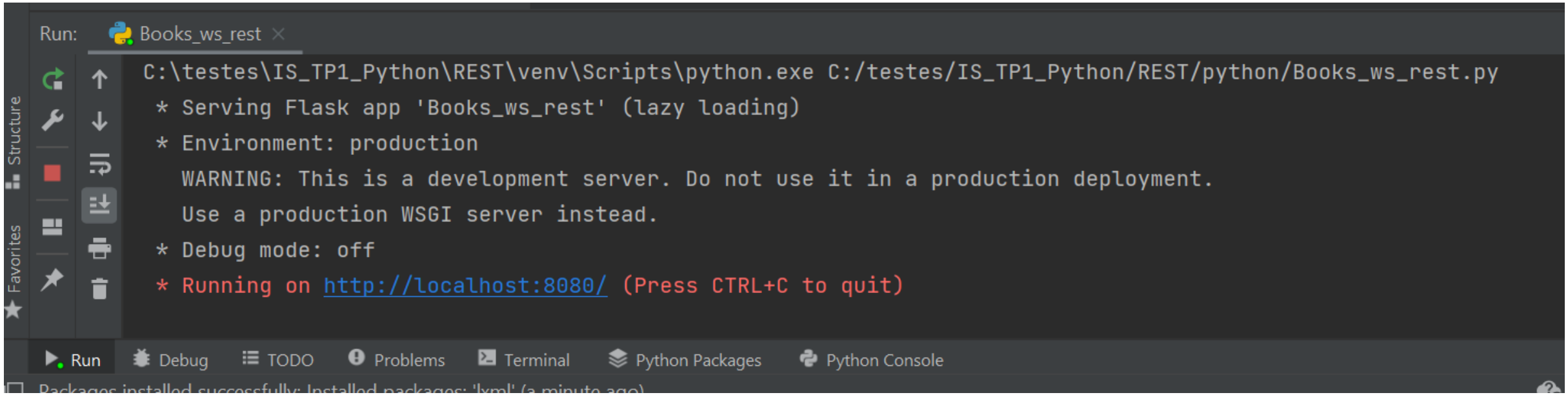


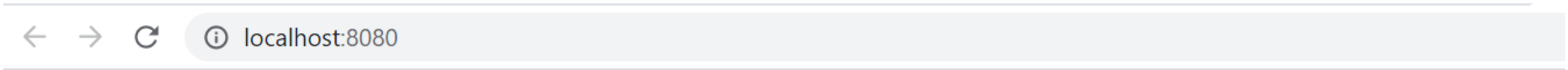

File
Edit
View
Navigate
Code
Refactor
Project
New Project...
New...
Alt+Insert
New Scratch File
Ctrl+Alt+Shift+Insert
Open...
Save As...
Open Recent
Close Project
Rename Project...
Settings...
Ctrl+Alt+S


Settings
Project: Books_ws_rest.py
Python Interpreter
Reset
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Project: Books_ws_rest.py
Python Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Python Interpreter:
Python 3.9 (venv) C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe
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
pip 21.1.2 21.3.1
setuptools 57.0.0 59.1.0
wheel 0.36.2 0.37.0
OK
Cancel
Apply


Available Packages
lxml
lxml-stubs
lxml-to-dict
lxml-wrapper
lxml-xpath-ipaddress
lxml-xpath2-functions
lxml2dict
lxml2json
lxmlHtml
lxmlbind
lxmlmechanize
lxmlmeld
lxmlmiddleware
lxmlproc
lxmlrpc_monkey
lxmlx
lxmlxtree
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
Reset
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Project: Books_ws_rest.py
Python Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Python Interpreter: Python 3.9 (venv) C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe
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





Esta aplicação contém os seguintes métodos:

XML Parse - Ficheiro books.xml

Resposta em XML e HTML

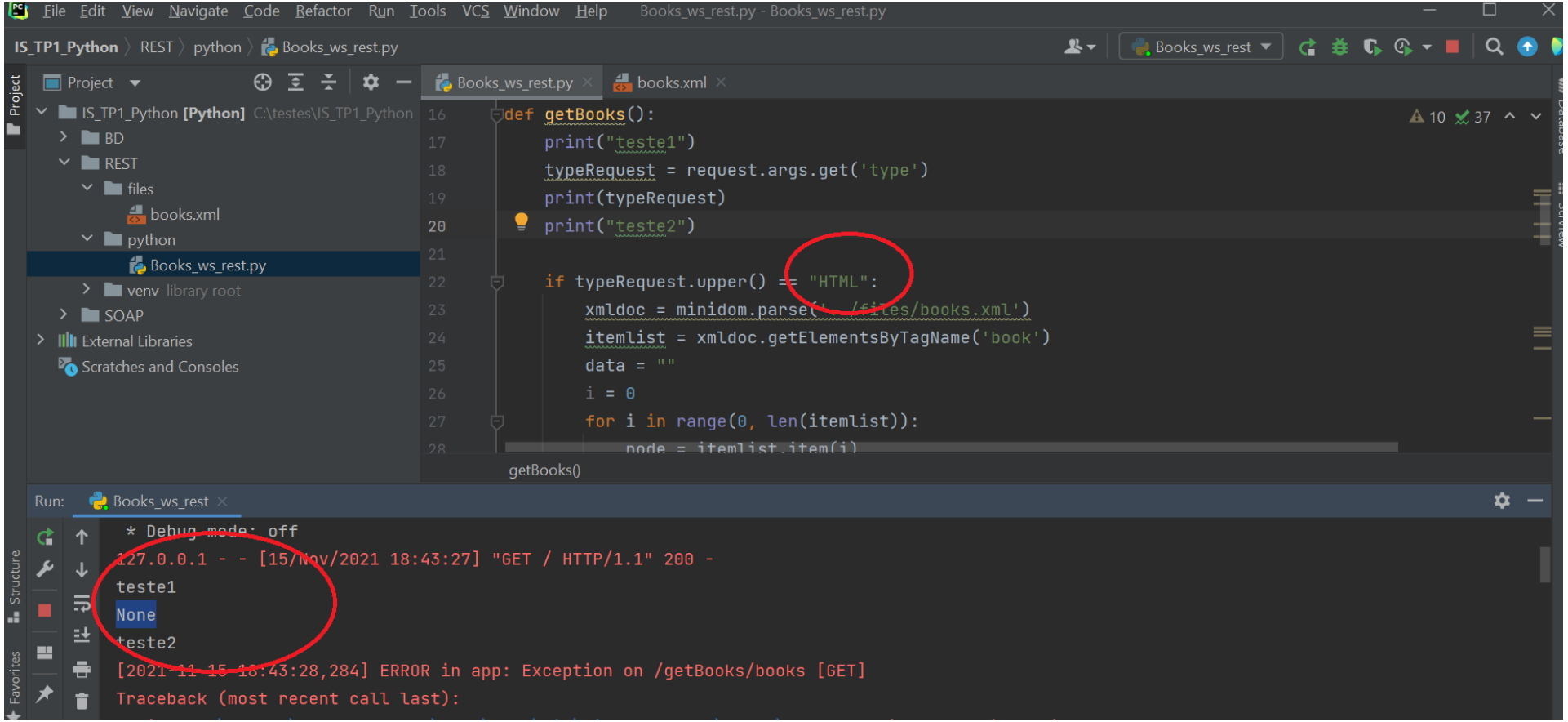
XML Parse e XPATH - Ficheiro books.xml

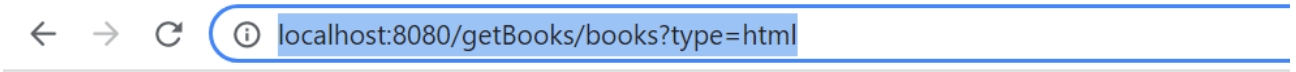
localhost:8080/getBooks/books

Internal Server Error

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

```
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
```





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.

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

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

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.

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


```
▼<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>
```

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

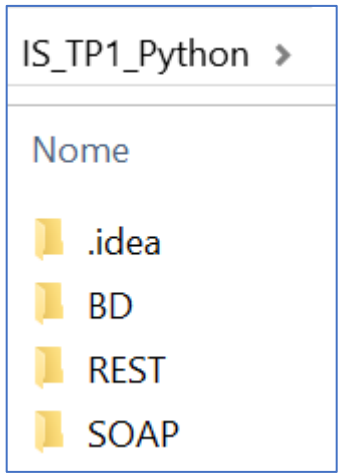
Genero: Fantasy

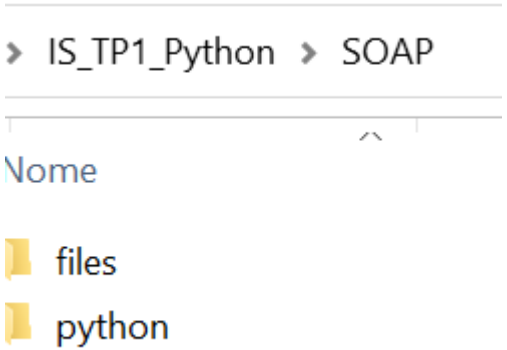
Preço: 5.95

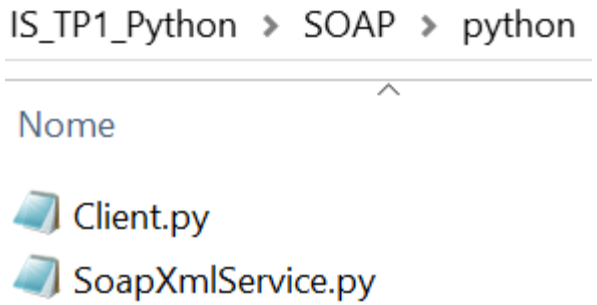
Data de Publicação: 2001-03-10

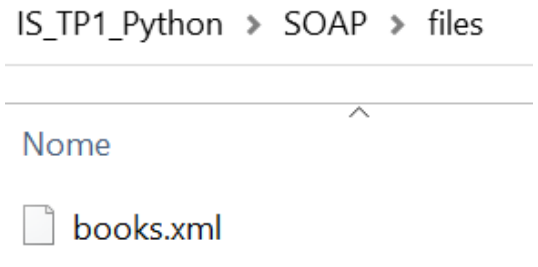
Descrição: In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.

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









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

SoapXmlService.py

```python
from spyne import Application, rpc, ServiceBase, Unicode
from spyne.protocol.soap import Soap11
from spyne.server.wsgi import WsgiApplication
from xml.dom import minidom
from lxml import etree


class SoapXmlService(ServiceBase):
    @rpc(_returns=Unicode)
    def books_parse(ctx):
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
        for i in range(0, len(itemlist)):
            node = itemlist.item(i)
            autor = node.getElementsByTagName('author')[0].childNodes[0].nodeValue
            titulo = node.getElementsByTagName('title')[0].childNodes[0].nodeValue
            genero = node.getElementsByTagName('genre')[0].childNodes[0].nodeValue
            preco = node.getElementsByTagName('price')[0].childNodes[0].nodeValue
            data_publicacao = node.getElementsByTagName('publish_date')[0].childNodes[0].nodeValue
            descricao = node.getElementsByTagName('description')[0].childNodes[0].nodeValue
            data = data + "<h5>Autor: " + autor + "<h5>"
            data = data + "<h5>Título: " + titulo + "<h5>"
            data = data + "<h5>Genero: " + genero + "<h5>"
            data = data + "<h5>Preço: " + preco + "<h5>"
            data = data + "<h5>Data de Publicação: " + data_publicacao + "<h5>"
            data = data + "<h5>Descrição: " + descricao + "<h5>"
            data = data + "<br>"
        return data

```

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

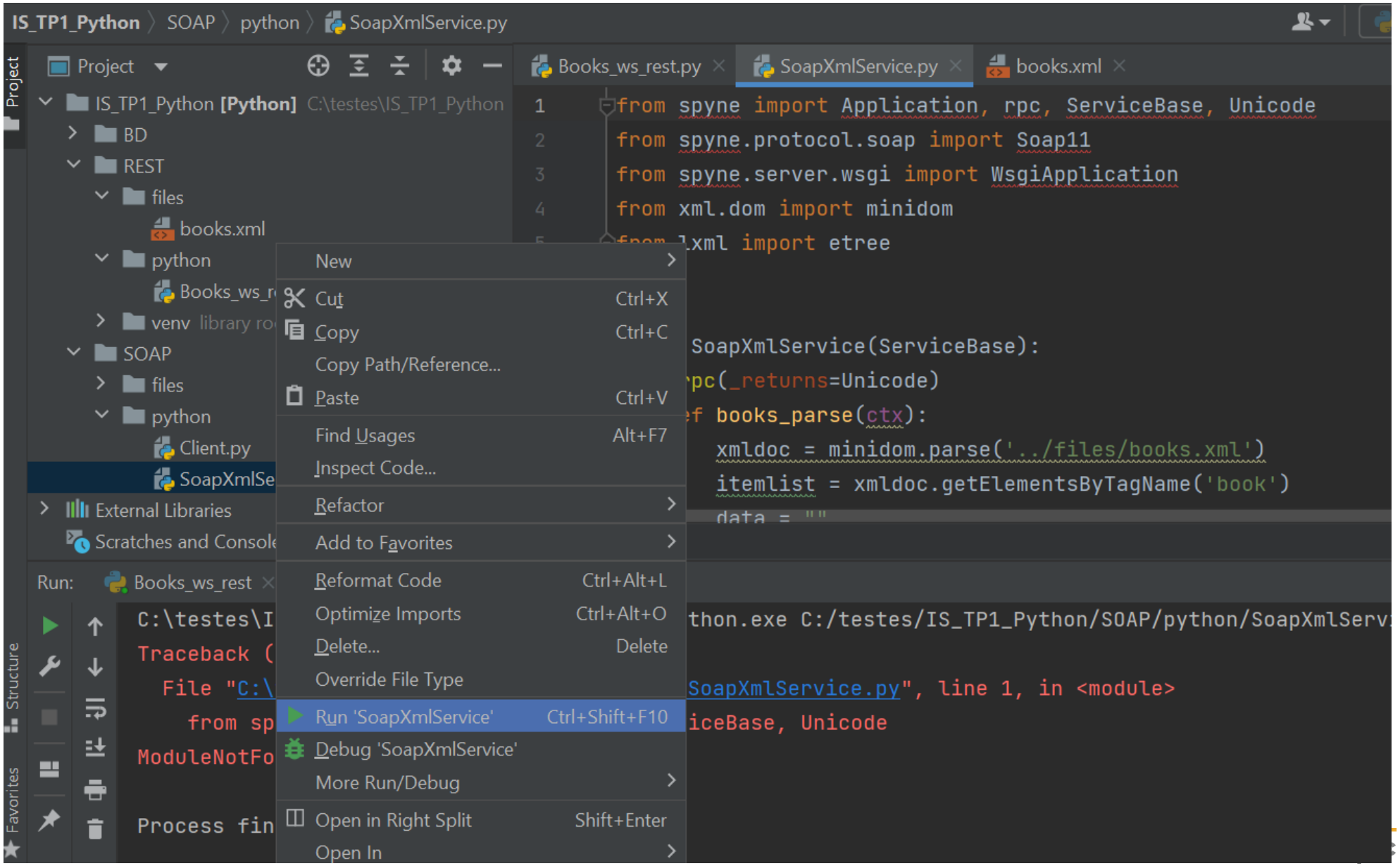
SoapXmlService.py

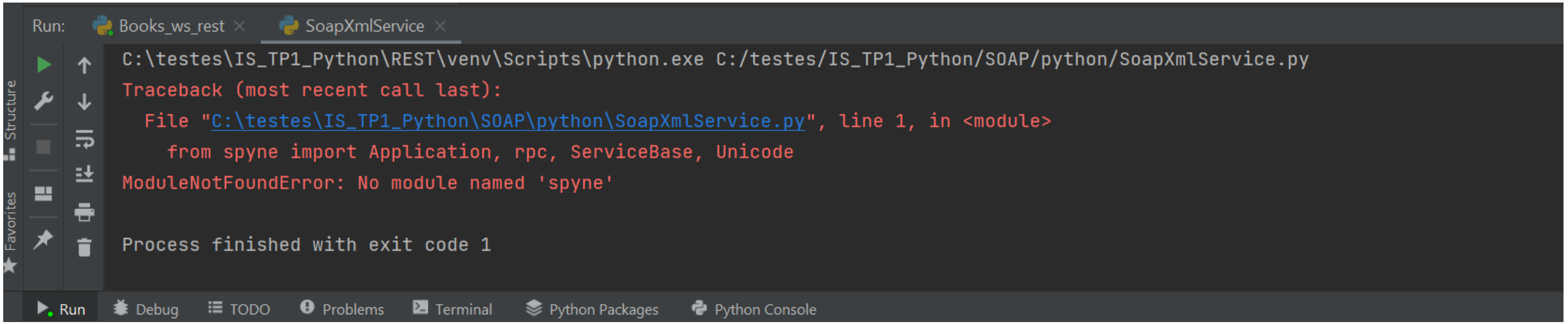
```
31
32      @rpc(_returns=Unicode)
33      def books_Xpath(ctx):
34          root = etree.parse('../files/books.xml')
35          books = root.xpath('/catalog//book[price<7.00 and price>5.00]')
36          data = "<h1>Resultados da pesquisa XPATH por: </h1><h2>price < 7.00 and price > 5.00</h2><br>"
37          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(validator='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
```

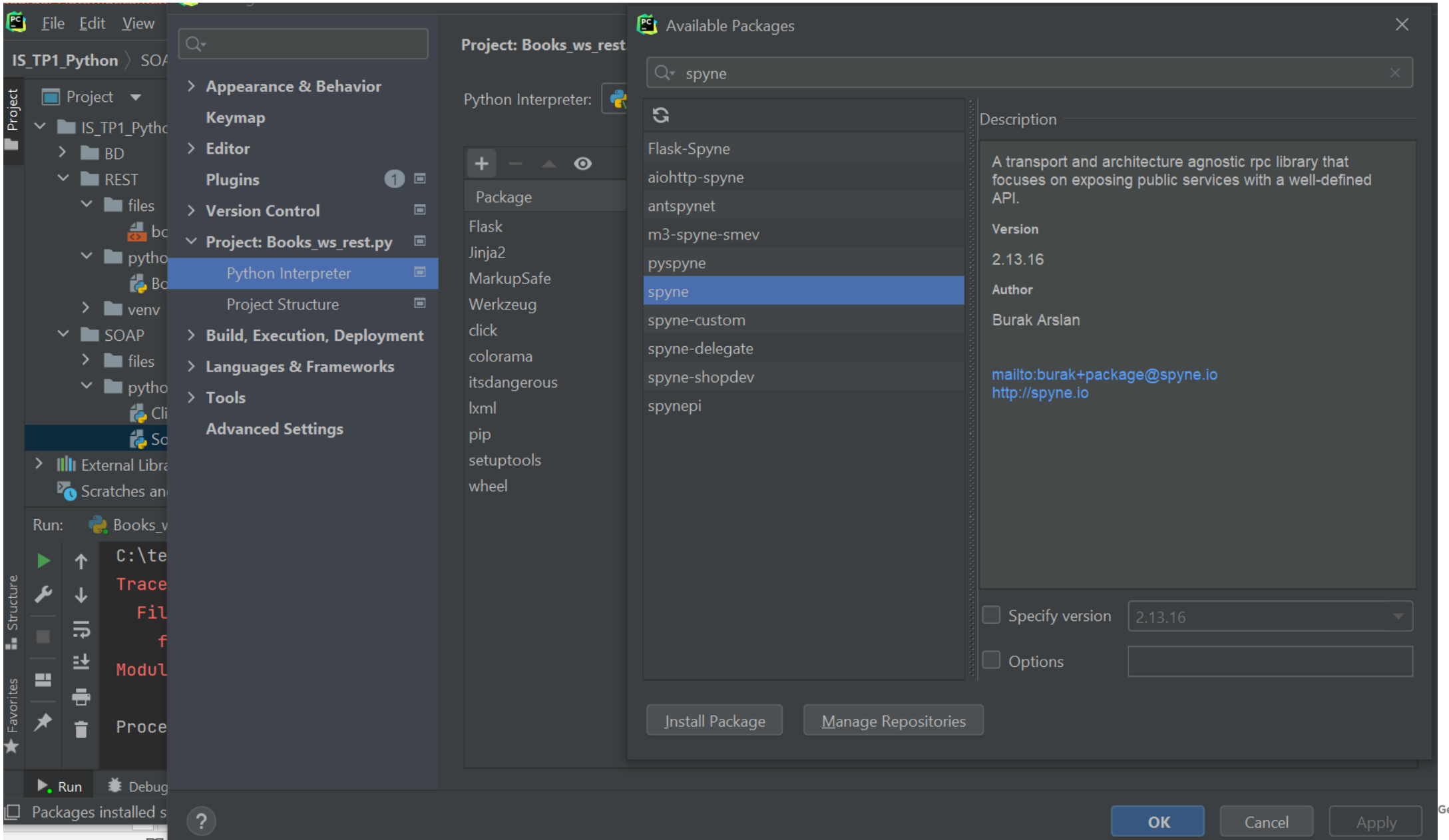
IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

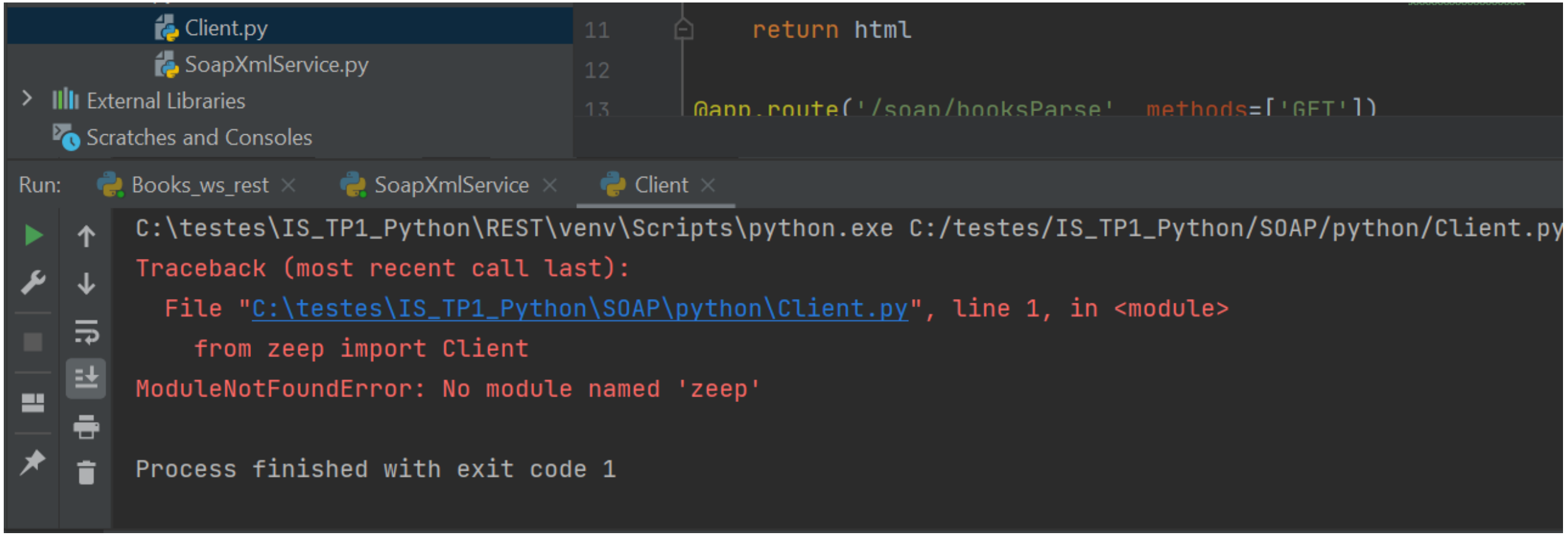
Client.py

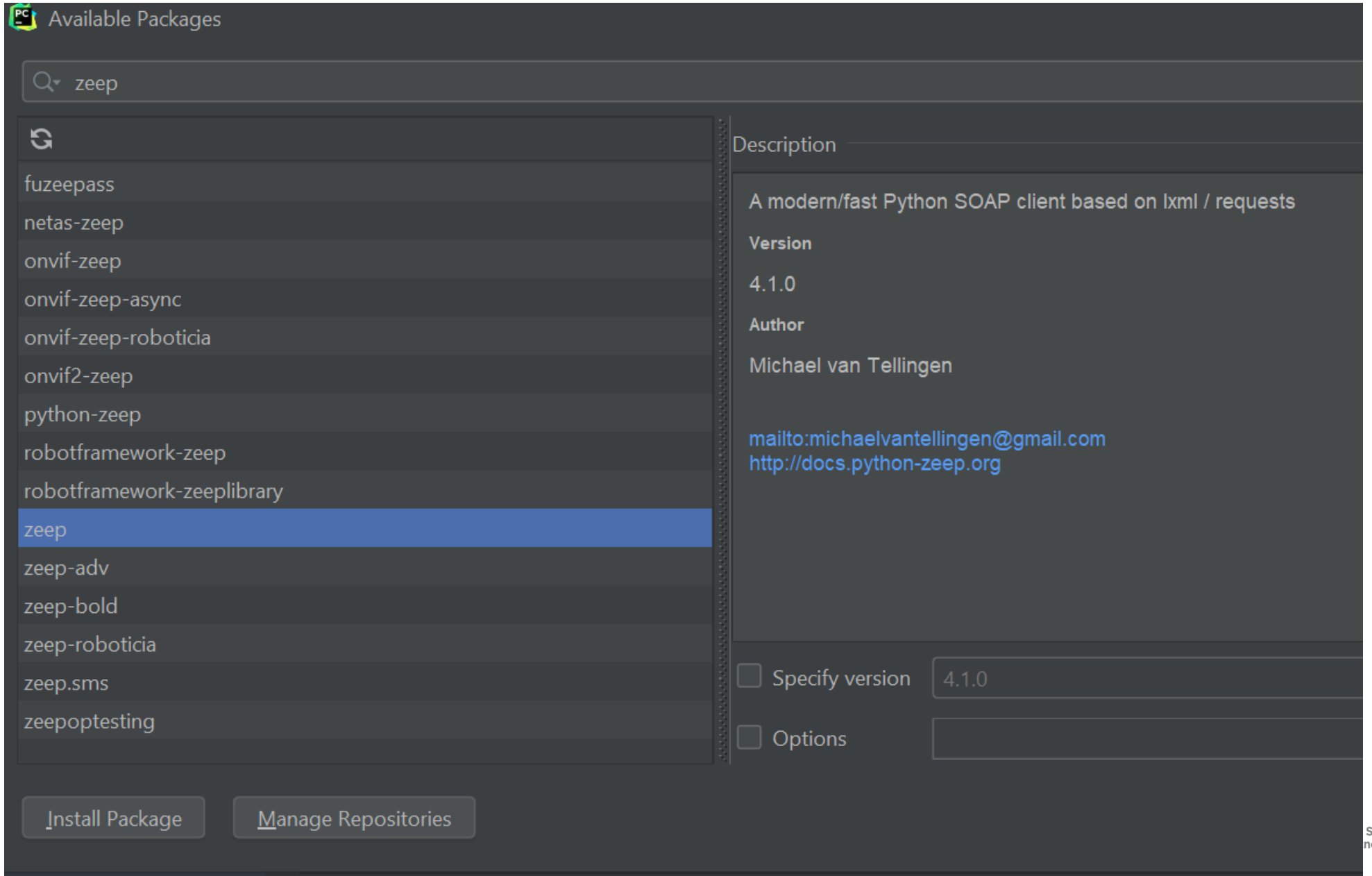
```python
from zeep import Client
from flask import Flask
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Servidor Flask para teste do Web service em SOAP: <h3>"
    html = html + "<a href='/soap/booksParse'>XML Parse - Ficheiro books.xml</a><br>"
    html = html + "<a href='/soap/Xpath'>XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/soap/booksParse', methods=['GET'])
def booksParseSOAP():
    client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
    data = client.service.books_parse()
    return data

@app.route('/soap/Xpath', methods=['GET'])
def booksXpath():
    client = Client(wsdl='http://localhost:8000/soap.books.parse?wsdl')
    data = client.service.books_Xpath()
    return data

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

```

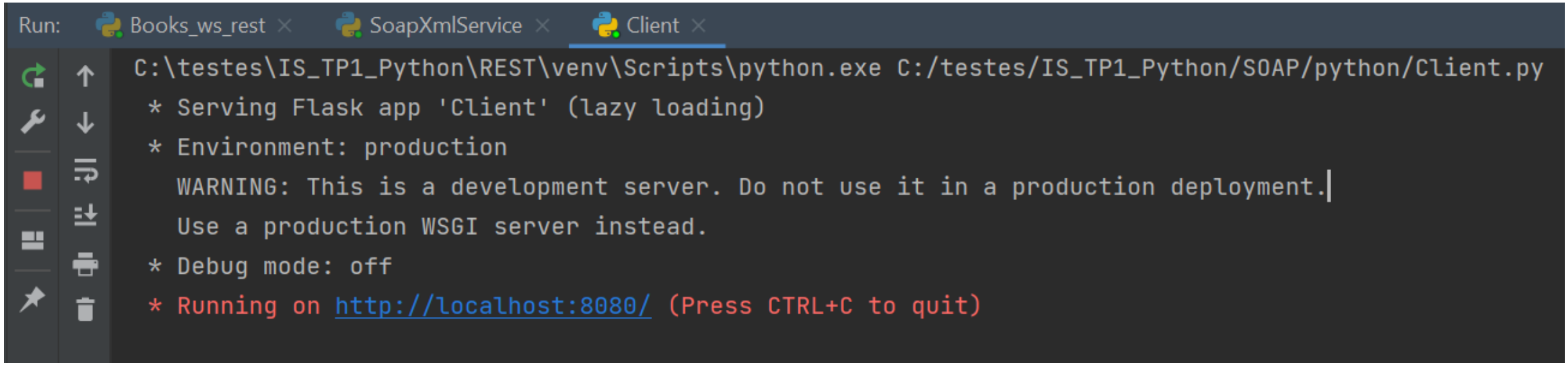

IS_TP1_Python 〉 SOAP 〉 python 〉 SoapXmlService.py
Project
IS_TP1_Python [Python] C:\testes\IS_TP1_Python
BD
REST
files
books.xml
python
venv library ro
SOAP
files
python
Client.py
External Libraries
Scratches and Consol
Books_ws_rest.py
SoapXmlService.py
books.xml
from spyne import Application, rpc, ServiceBase, Unicode
from spyne.protocol.soap import Soap11
from spyne.server.wsgi import WsgiApplication
from xml.dom import minidom
xml import etree
SoapXmlService(ServiceBase):
pc(_returns=Unicode)
f books_parse(ctx):
xmldoc = minidom.parse('../files/books.xml')
itemlist = xmldoc.getElementsByTagName('book')
data = ""
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
Run: Books_ws_rest
C:\testes\I
Traceback (
File "C:\
from sp
ModuleNotFo
Process fin
thon.exe C:/testes/IS_TP1_Python/SOAP/python/SoapXmlServ
SoapXmlService.py", line 1, in <module>
iceBase, Unicode
Structure
Favorites

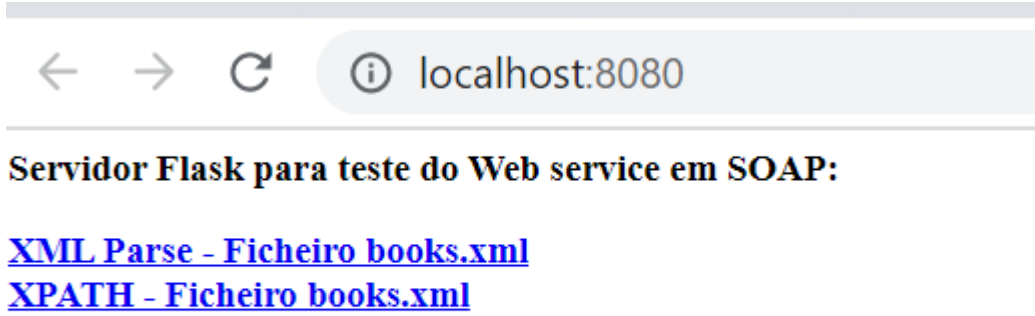

Run:
Books_ws_rest
SoapXmlService
C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/SOAP/python/SoapXmlService.py
Traceback (most recent call last):
File "C:\testes\IS_TP1_Python\SOAP\python\SoapXmlService.py", line 1, in <module>
from spyne import Application, rpc, ServiceBase, Unicode
ModuleNotFoundError: No module named 'spyne'
Process finished with exit code 1
Structure
Favorites
Run
Debug
TODO
Problems
Terminal
Python Packages
Python Console

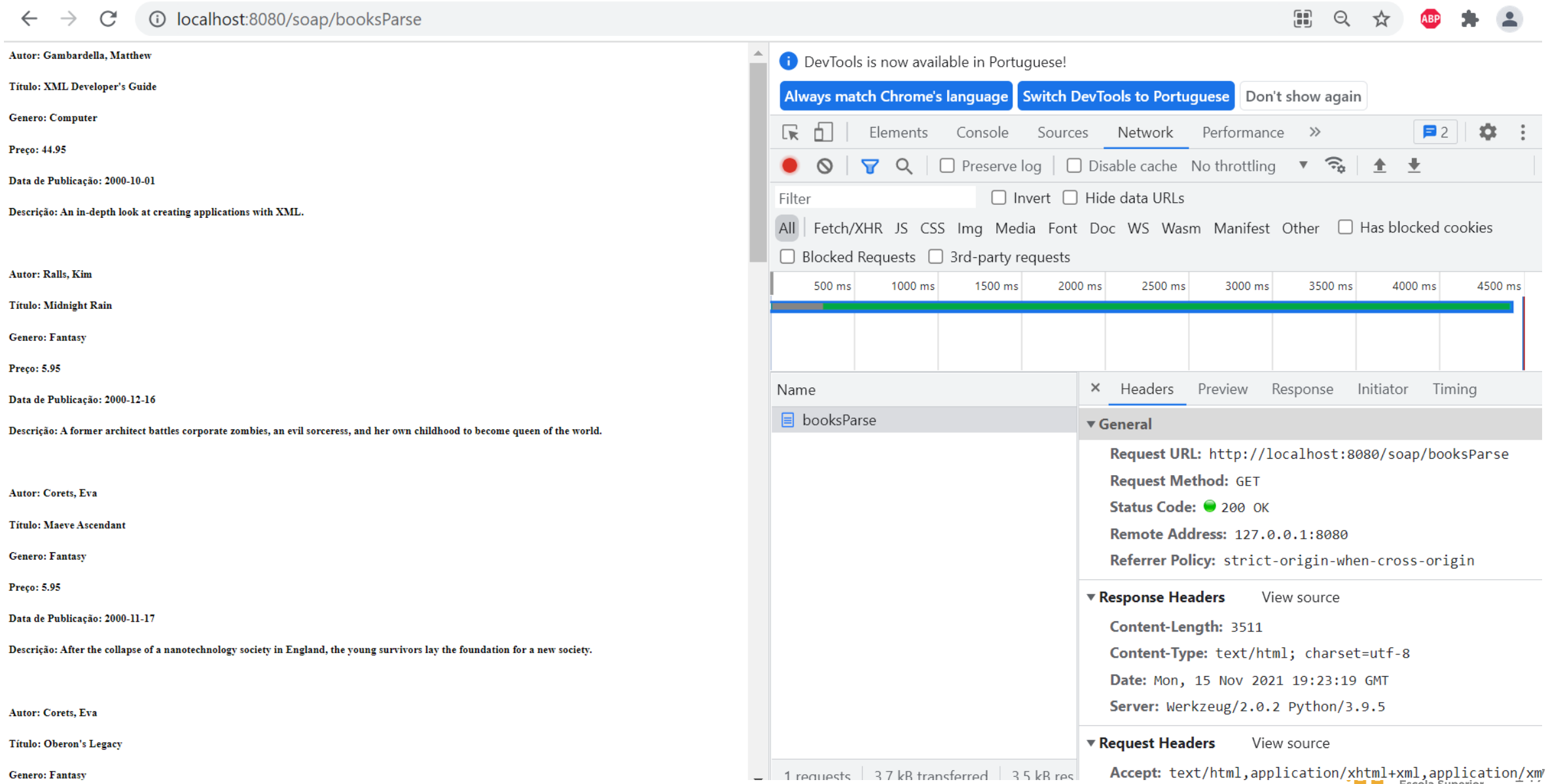

File Edit View
IS_TP1_Python
Project
IS_TP1_Pytho
BD
REST
files
venv
SOAP
files
External Libra
Scratches an
Run:
Books_v
C:\te
Trace
Fil
f
Modul
Proce
Run
Debug
Packages installed s
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Project: Books_ws_rest.py
Python Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Project: Books_ws_rest
Python Interpreter:
Package
Flask
Jinja2
MarkupSafe
Werkzeug
click
colorama
itsdangerous
lxml
pip
setuptools
wheel
Available Packages
spyne
Flask-Spyne
aiohttp-spyne
antspynet
m3-spyne-smev
pyspyne
spyne
spyne-custom
spyne-delegate
spyne-shopdev
spynepi
Description
A transport and architecture agnostic rpc library that focuses on exposing public services with a well-defined API.
Version
2.13.16
Author
Burak Arslan
mailto:burak+package@spyne.io
http://spyne.io
Specify version
2.13.16
Options
Install Package
Manage Repositories
OK
Cancel
Apply

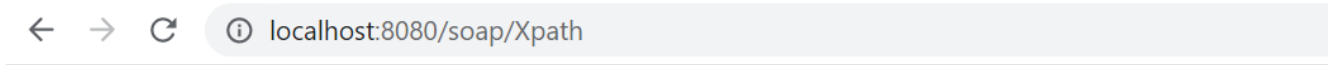

Client.py
SoapXmlService.py
External Libraries
Scratches and Consoles
11 return html
12
13 @app.route('/soap/booksParse', methods=['GET'])
Run: Books_ws_rest SoapXmlService Client
C:\testes\IS_TP1_Python\REST\venv\Scripts\python.exe C:/testes/IS_TP1_Python/SOAP/python/Client.py
Traceback (most recent 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


Available Packages
zeep
fuzeepass
netas-zeep
onvif-zeep
onvif-zeep-async
onvif-zeep-roboticia
onvif2-zeep
python-zeep
robotframework-zeep
robotframework-zeeplibrary
zeep
zeep-adv
zeep-bold
zeep-roboticia
zeep.sms
zeepoptesting
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)


localhost:8080
Servidor Flask para teste do Web service em SOAP:
XML Parse - Ficheiro books.xml
XPATH - Ficheiro books.xml


localhost:8080/soap/booksParse
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.
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
DevTools is now available in Portuguese!
Always match Chrome's language
Switch DevTools to Portuguese
Don't show again
Elements
Console
Sources
Network
Performance
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
4500 ms
Name
booksParse
Headers
Preview
Response
Initiator
Timing
General
Request URL: http://localhost:8080/soap/booksParse
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/xm



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

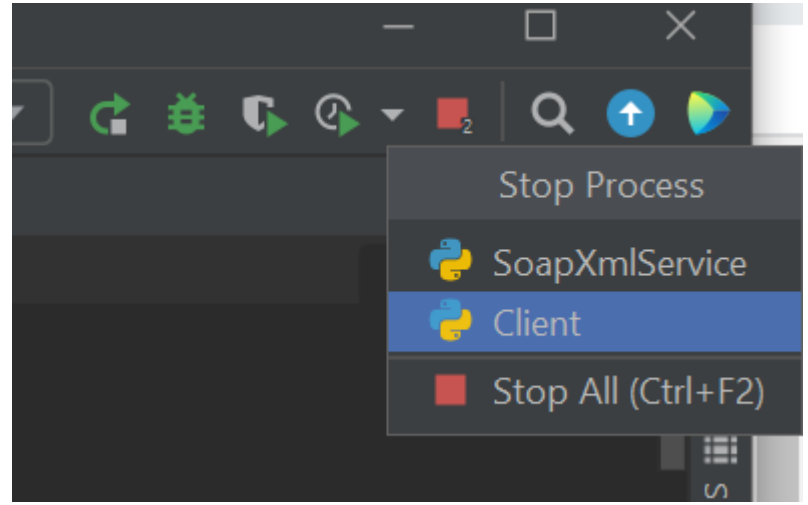
Título: Oberon's Legacy

Genero: Fantasy

Preço: 5.95

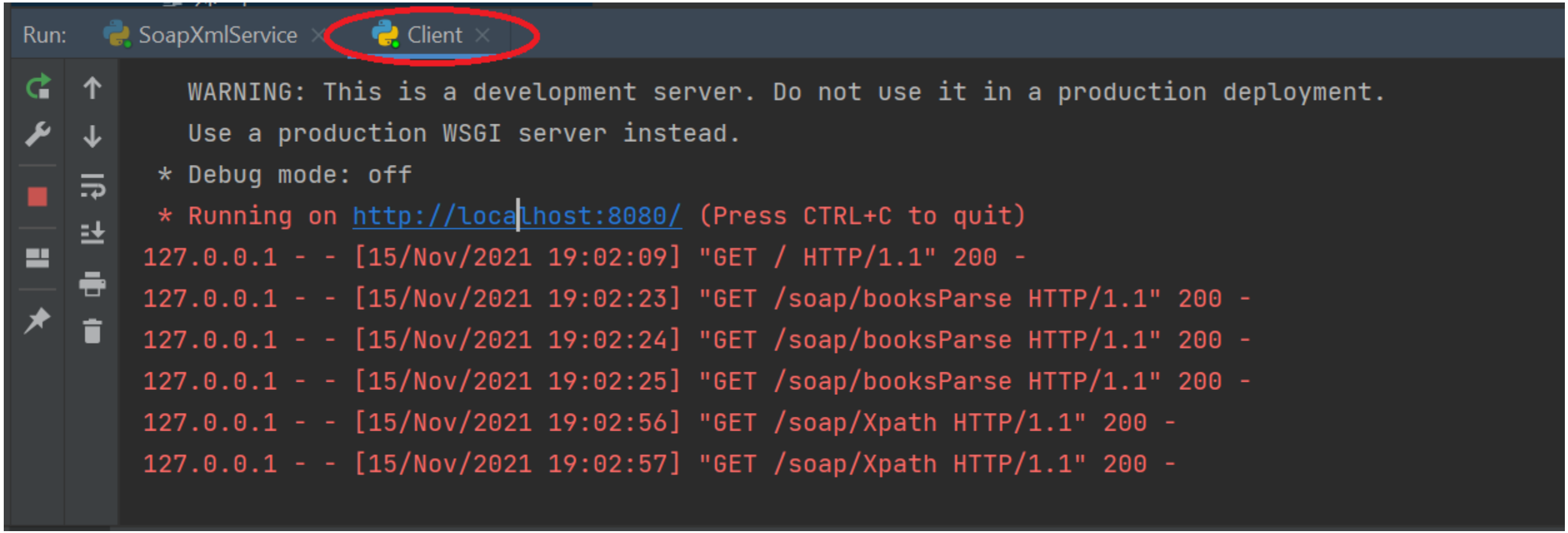
Data de Publicação: 2001-03-10

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.



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/pkolclcpcbkghocbmacdmcheaioncncl

chrome web store

Iniciar sessão

Página Inicial > Extensões > SOAP Analyzer

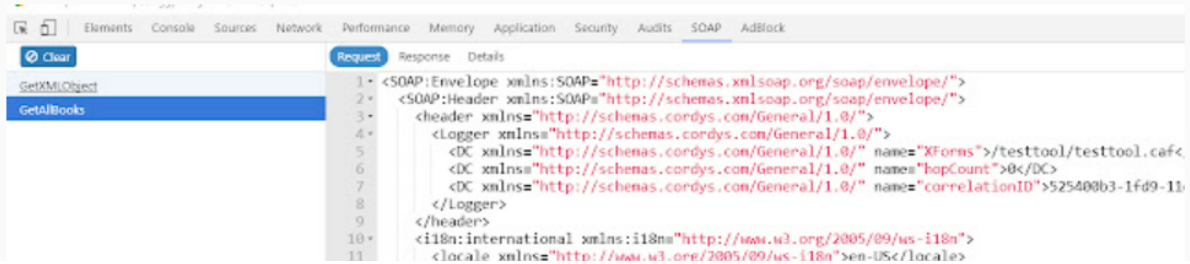
SOAP Analyzer

Instalar

Disponibilizado por: DJ Collier

21 | Ferramentas do Programador | 4 000+ utilizadores

Vista geral | Práticas de Privacidade | Críticas | Suporte | Relacionados



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

soapui.org/docs/soap-and-wsdl/working-with-wsdls/

SoapUI Supported by SMARTBEAR — Tools — Learn — Docs — Resources — Try ReadyAPI Free

How can we help? Clear

- SoapUI Projects
- REST API Testing
- SOAP and WSDL
  - Working with WSDL Coverage
  - HTTP Headers
  - Working with WSDLs
  - Attachments and Files
  - Operations and Requests
  - Tips & Tricks

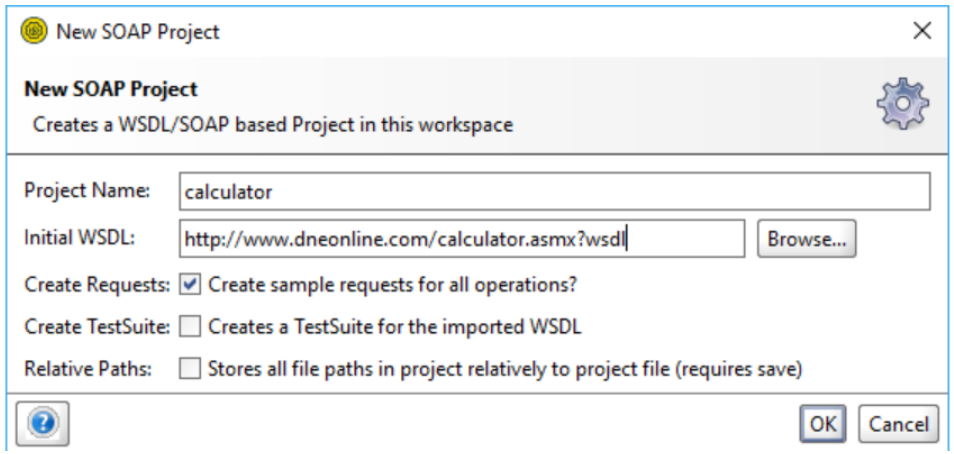
Explore WSDL
Validate the WSDL against the WS-I Basic Profile
Generating Code for your WSDL

Work with WSDLs in SoapUI

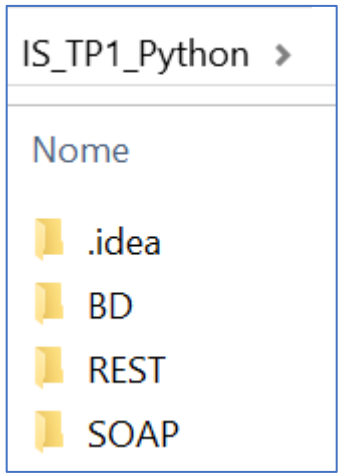
Create Project From WSDL

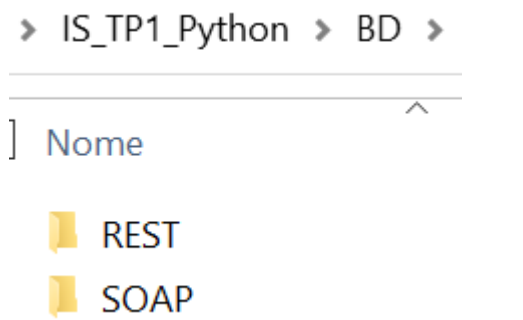
To take a closer look at a WSDL file, create a new project and import a sample WSDL file:

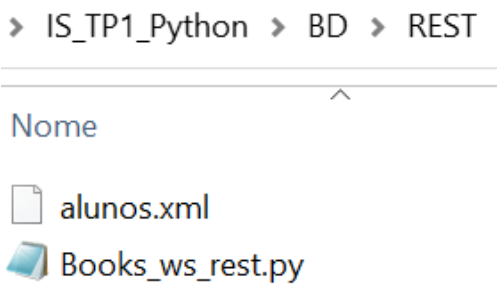
1. In SoapUI, click or select **File > New SOAP Project**

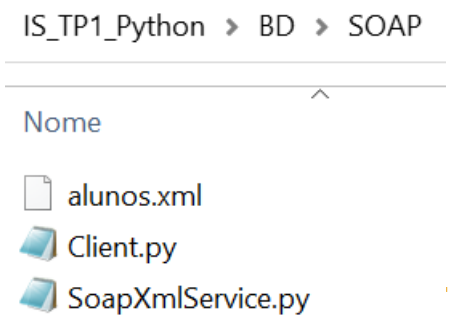


IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

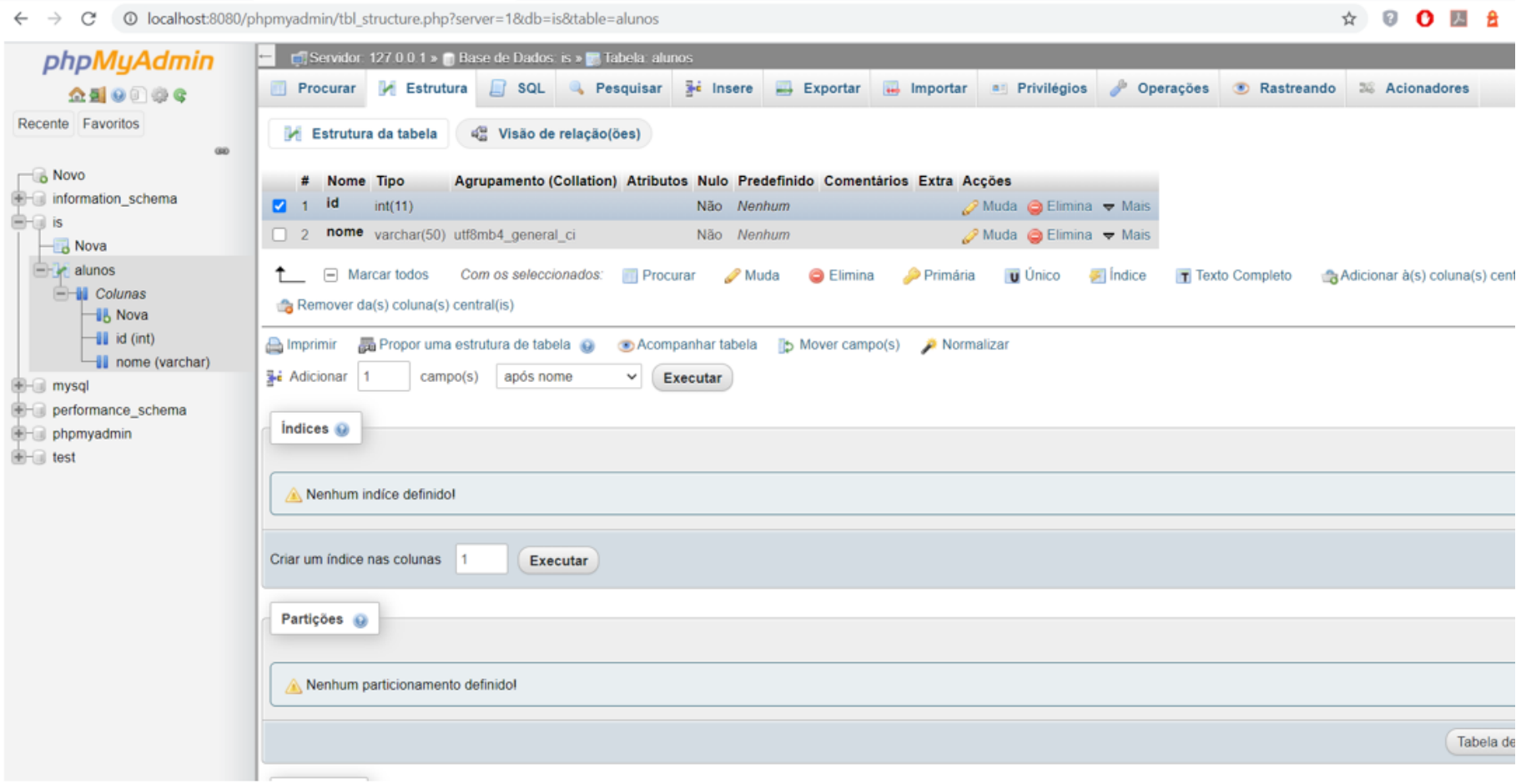




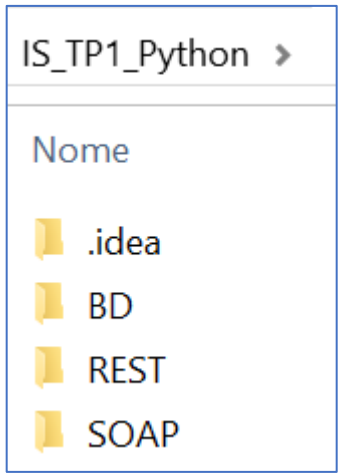


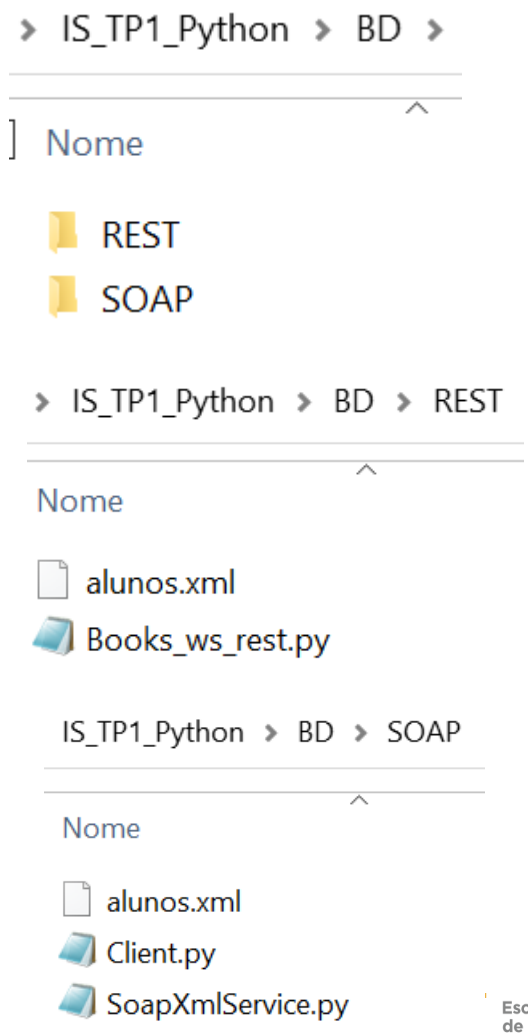


IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS



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





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

Books_ws_rest.py

```python
from xml.dom import minidom
from flask import Flask, request
from werkzeug.wrappers import Response
import mysql.connector
app = Flask(__name__)

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="is"
)

@app.route('/')
def index():
    html = "<h3>Web Service REST com acesso à base de dados MySQL: <h3>"
    html = html + "<a href='/getAlunos/alunos?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
    return html

@app.route('/getAlunos/alunos', methods=['GET'])
def getBooks():
    selectType = request.args.get('type')
    if selectType.upper() == "ALL":
        mycursor = mydb.cursor()
        mycursor.execute("SELECT * FROM alunos")
        myresult = mycursor.fetchall()
        doc = minidom.Document()
        tag_alunos = doc.createElement('alunos')
        for aluno in myresult:
            id_aluno = str(aluno[0])
            nome = aluno[1]
            tag_aluno = doc.createElement('aluno')

            tag_id = doc.createElement('id')
            content_id = doc.createTextNode(id_aluno)
            tag_id.appendChild(content_id)
```

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

Books_ws_rest.py

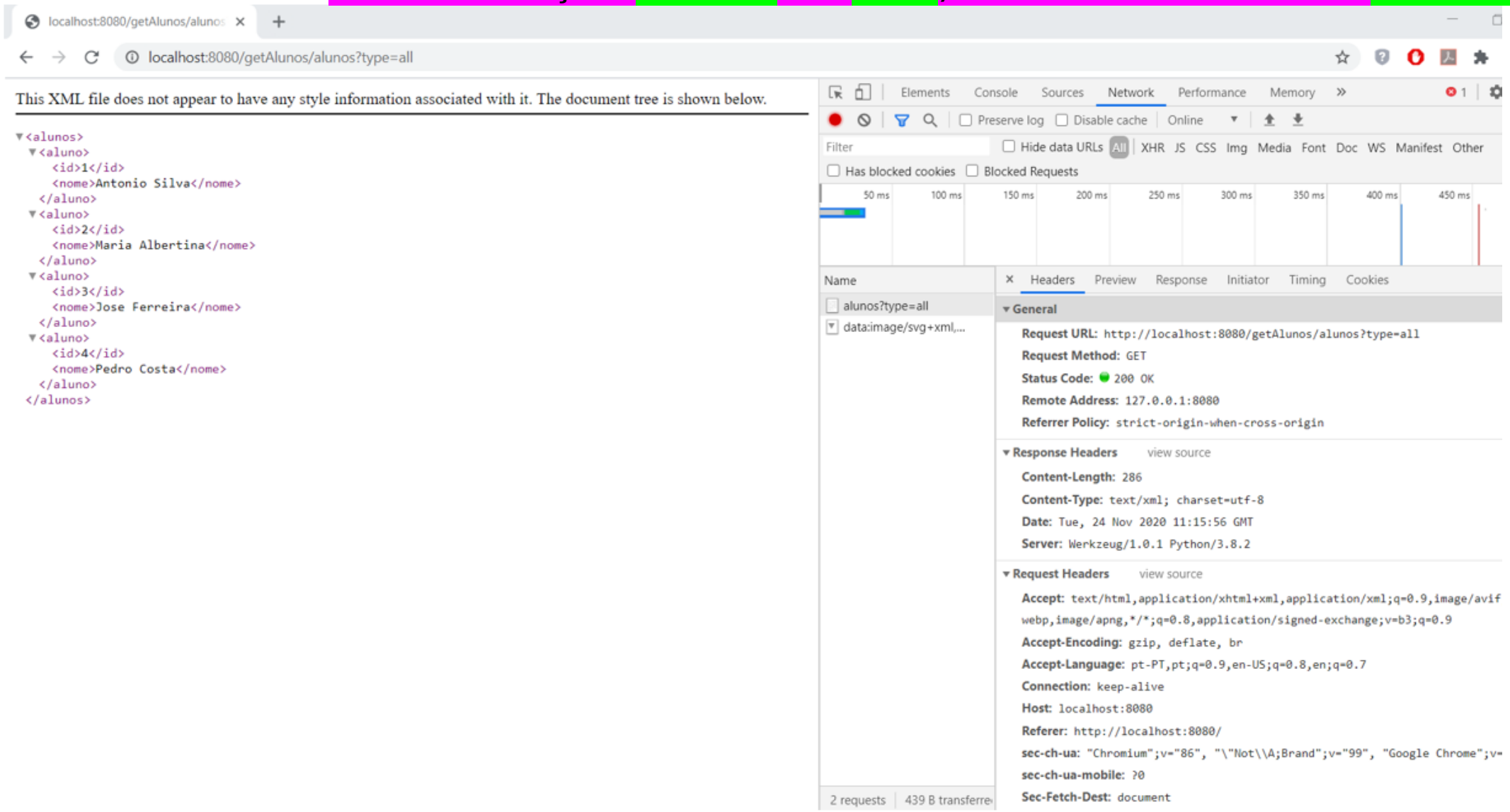
```python
from xml.dom import minidom
from flask import Flask, request
from werkzeug.wrappers import Response
import mysql.connector
app = Flask(__name__)

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="is"
)

@app.route('/')
def index():
    html = "<h3>Web Service REST com acesso à base de dados MySQL: <h3>"
    html = html + "<a href='/getAlunos/alunos?type=all'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
    return html

@app.route('/getAlunos/alunos', methods=['GET'])
def getBooks():
    selectType = request.args.get('type')
    if selectType.upper() == "ALL":
        mycursor = mydb.cursor()
        mycursor.execute("SELECT * FROM alunos")
        myresult = mycursor.fetchall()
        doc = minidom.Document()
        tag_alunos = doc.createElement('alunos')
        for aluno in myresult:
            id_aluno = str(aluno[0])
            nome = aluno[1]
            tag_aluno = doc.createElement('aluno')

            tag_id = doc.createElement('id')
            content_id = doc.createTextNode(id_aluno)
            tag_id.appendChild(content_id)
```

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

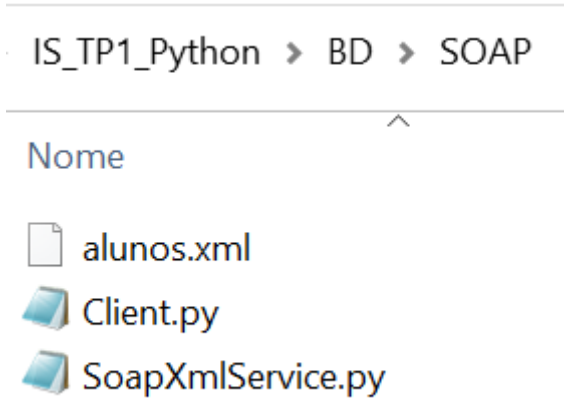
Books_ws_rest.py

```
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)
56
```

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



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



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

SoapXmlService.py

```python
from spyne import Application, rpc, ServiceBase, Unicode
from spyne.protocol.soap import Soap11
from spyne.server.wsgi import WsgiApplication
from xml.dom import minidom
import mysql.connector



mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="is"
)

class SoapXmlService(ServiceBase):

    @rpc(_returns=Unicode)
    def alunos_bd(ctx):
        mycursor = mydb.cursor()
        mycursor.execute("SELECT * FROM alunos")
        myresult = mycursor.fetchall()
        doc = minidom.Document()
        tag_alunos = doc.createElement('alunos')
        for aluno in myresult:
            id_aluno = str(aluno[0])
            nome = aluno[1]
            tag_aluno = doc.createElement('aluno')

            tag_id = doc.createElement('id')
            content_id = doc.createTextNode(id_aluno)
            tag_id.appendChild(content_id)

            tag_nome = doc.createElement('nome')
            content_nome = doc.createTextNode(nome)
            tag_nome.appendChild(content_nome)
```

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

SoapXmlService.py

```
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          return string_xml
50
```

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

SoapXmlService.py

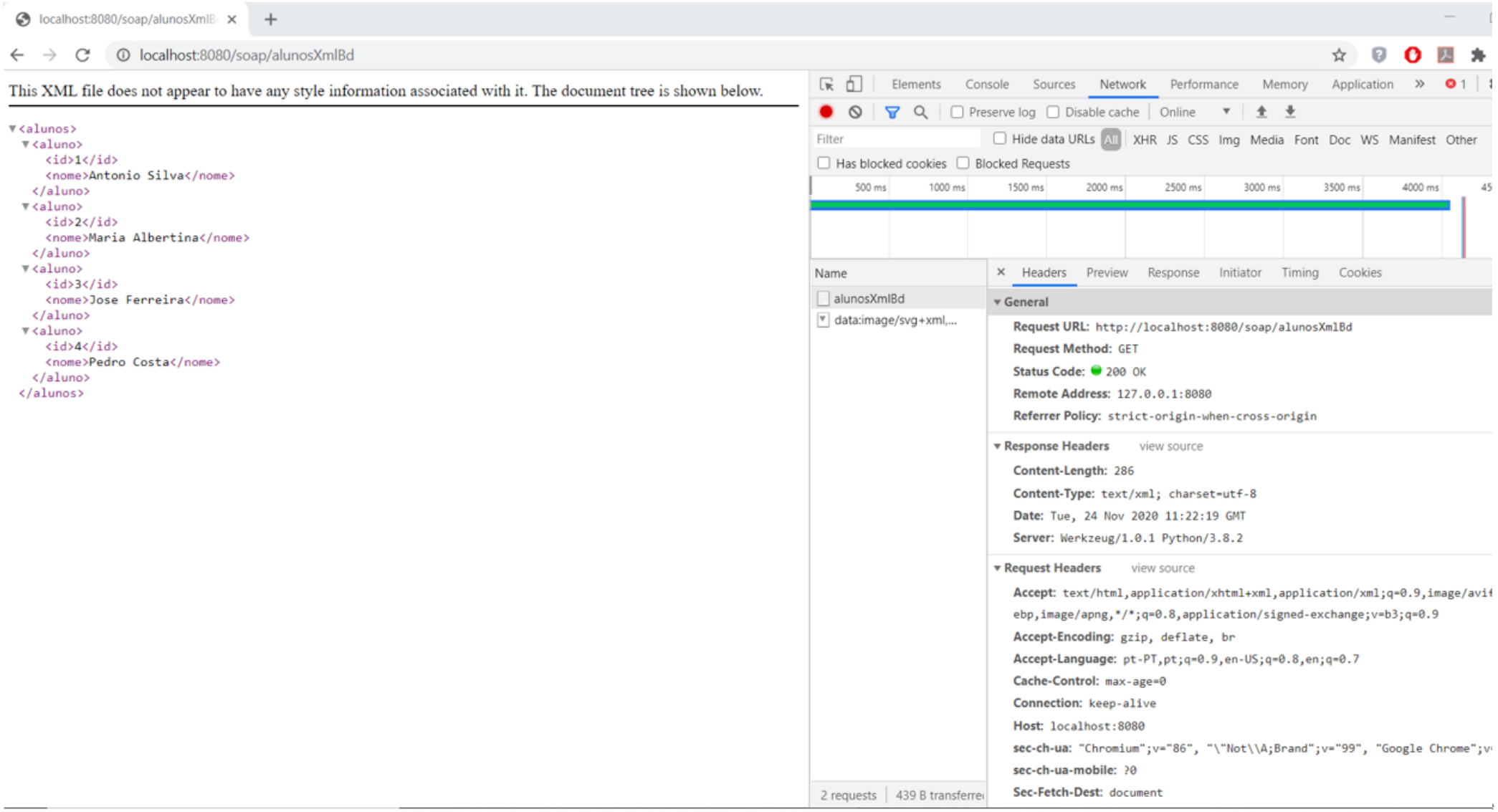
```
43              doc.appendChild(tag_alunos)
44              string_xml = doc.toprettyxml(indent="\t")
45              caminho_ficheiro = "alunos.xml"
46              with open(caminho_ficheiro, "w") as f:
47                  f.write(string_xml)
48
49              return string_xml
50
51
52  application = Application([SoapXmlService],
53                            tns='soap.alunos.db',
54                            in_protocol=Soap11(validator='lxml'),
55                            out_protocol=Soap11()
56                            )
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
```

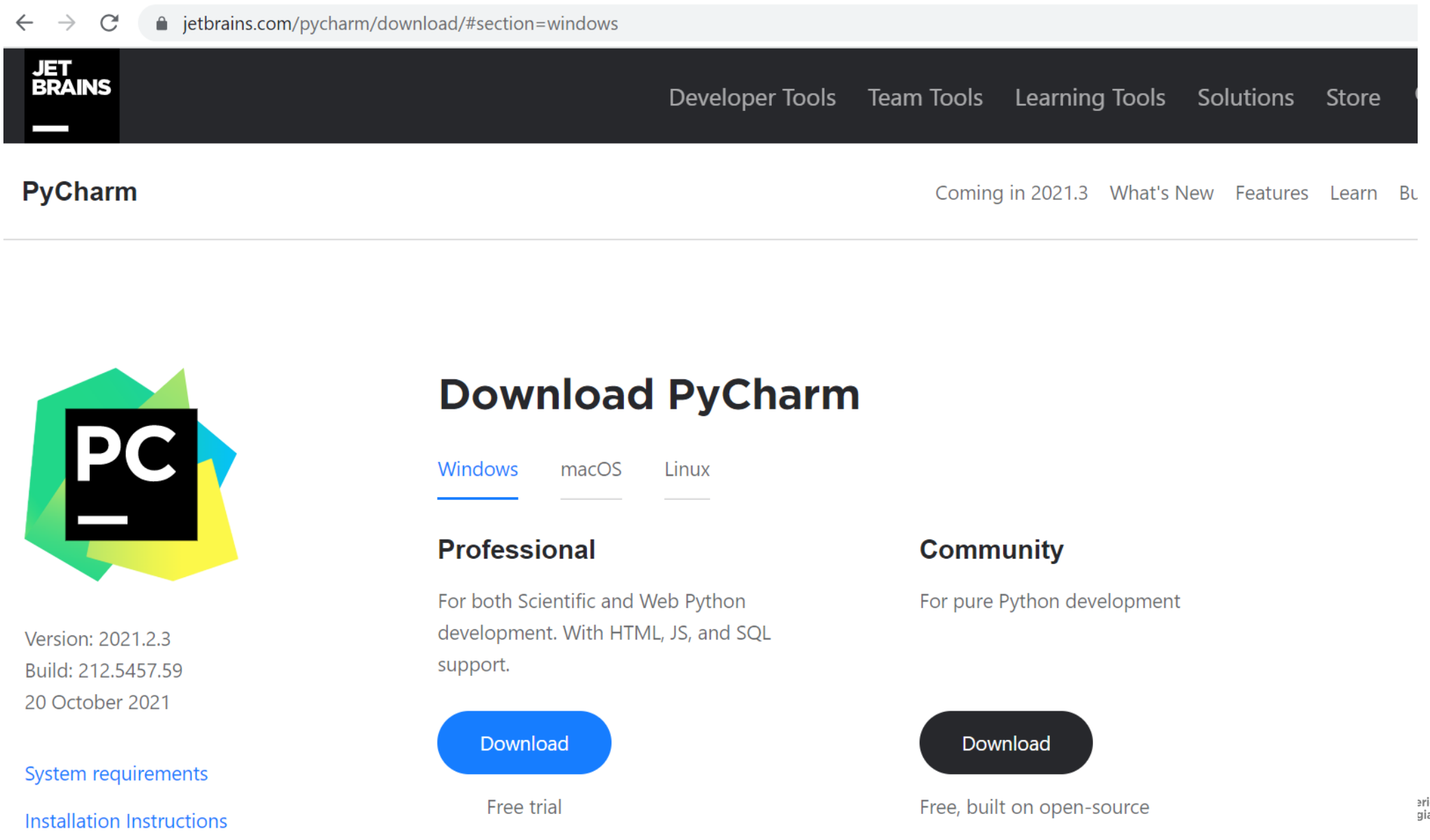
IMPLEMENTAÇÃO PYTHON VIA REST, SOAP USANDO XPATH E BASE DE DADOS

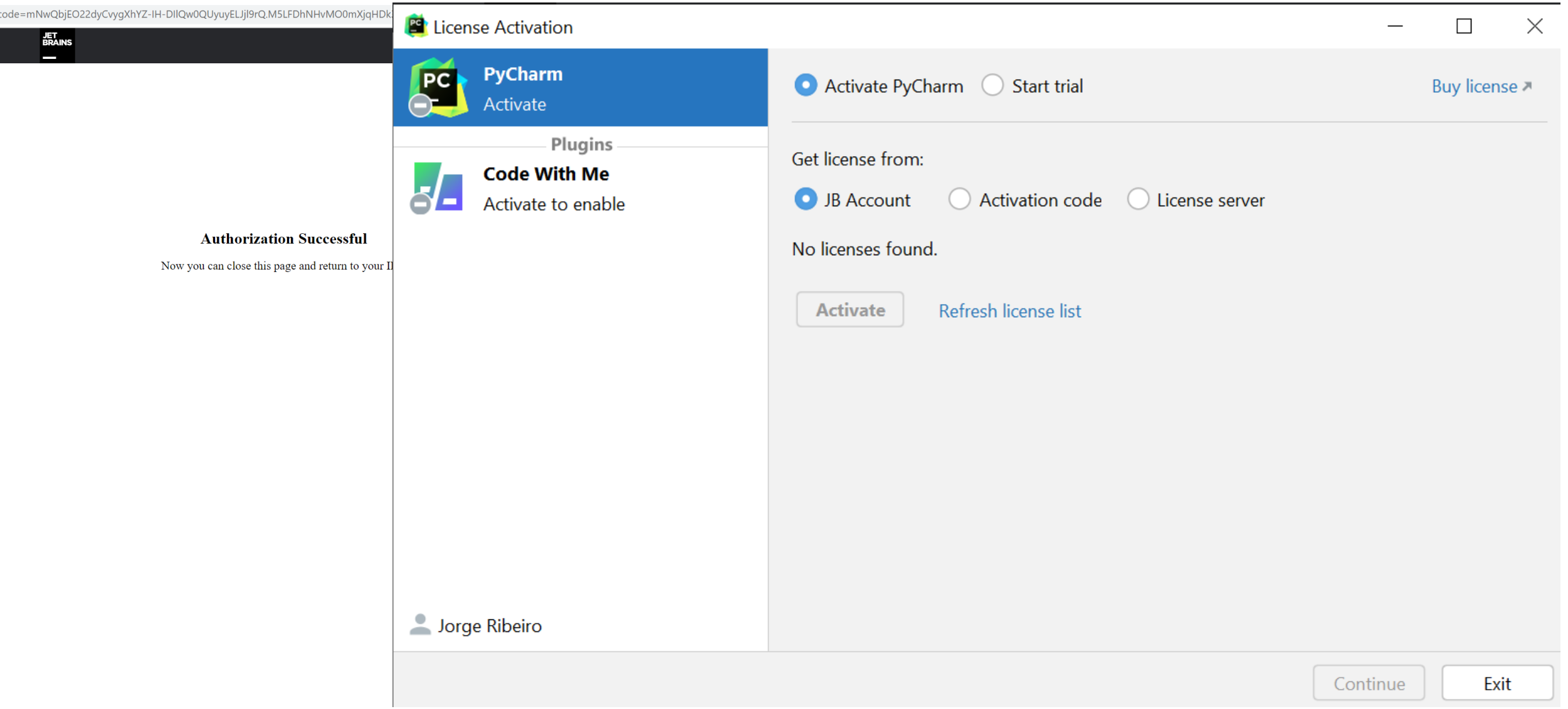
SoapXmlService.py | Client.py

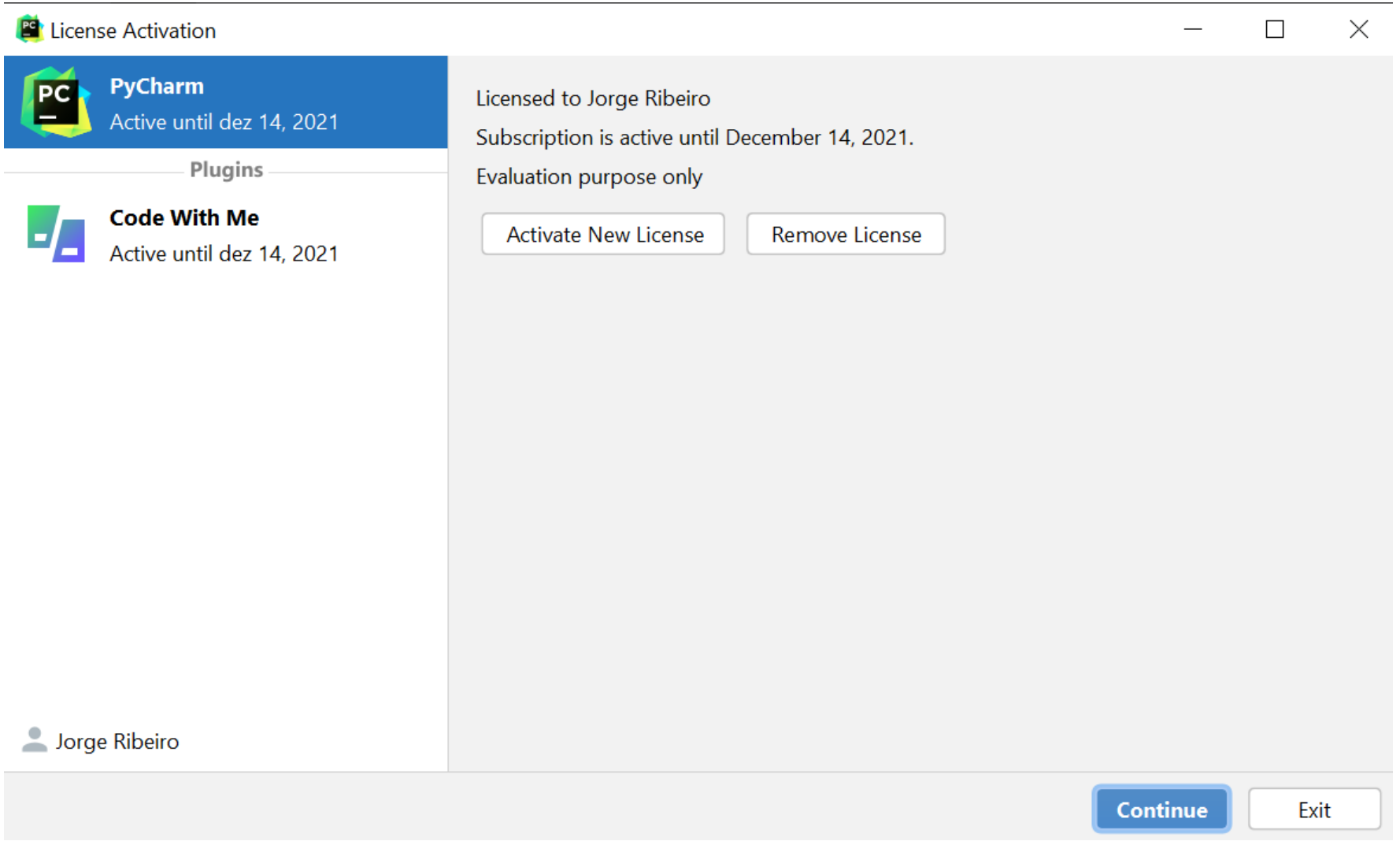
```python
from zeep import Client
from flask import Flask
from werkzeug.wrappers import Response
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Métodos disponíveis: <h3>"
    html = html + "<a href='/soap/alunosXmlBd'>Obter alunos da BD MySQL - Criar e apresentar ficheiro XML</a><br>"
    return html

@app.route('/soap/alunosXmlBd', methods=['GET'])
def alunosXmlBd():
    client = Client(wsdl='http://localhost:8000/soap.alunos.db?wsdl')
    data = client.service.alunos_bd()
    return Response(data, mimetype='text/xml')

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

```

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




jetbrains.com/pycharm/download/#section=windows
JET BRAINS
Developer Tools
Team Tools
Learning Tools
Solutions
Store
PyCharm
Coming in 2021.3
What's New
Features
Learn
Bu
PC
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
Version: 2021.2.3
Build: 212.5457.59
20 October 2021
System requirements
Installation Instructions


code=mNwQbjEO22dyCvygXhYZ-IH-DIlQw0QUyuyELJjl9rQ.M5LFDhNHvMO0mXjqHDk
JET BRAINS
Authorization Successful
Now you can close this page and return to your I
License Activation
PyCharm
Activate
Plugins
Code With Me
Activate to enable
Jorge Ribeiro
Activate PyCharm
Start trial
Buy license
Get license from:
JB Account
Activation code
License server
No licenses found.
Activate
Refresh license list
Continue
Exit


License Activation
PyCharm
Active until dez 14, 2021
Plugins
Code With Me
Active until dez 14, 2021
Jorge Ribeiro
Licensed to Jorge Ribeiro
Subscription is active until December 14, 2021.
Evaluation purpose only
Activate New License
Remove License
Continue
Exit

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 Client (2)

Books_ws_rest.py

```python
from xml.dom import minidom
from flask import Flask, request
from lxml import etree
from werkzeug.wrappers import Response
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Esta aplicação contém os seguintes métodos: <h3>"
    html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
    html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/getBooks/books', methods=['GET'])
def getBooks():
    typeRequest = request.args.get('type')
    if typeRequest.upper() == "HTML":
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
        i = 0
```

3 9

Project Structure Favorites Database SciView

TODO Problems Terminal Python Packages Python Console Event Log

1:1 CRLF UTF-8 4 spaces <No interpreter>

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

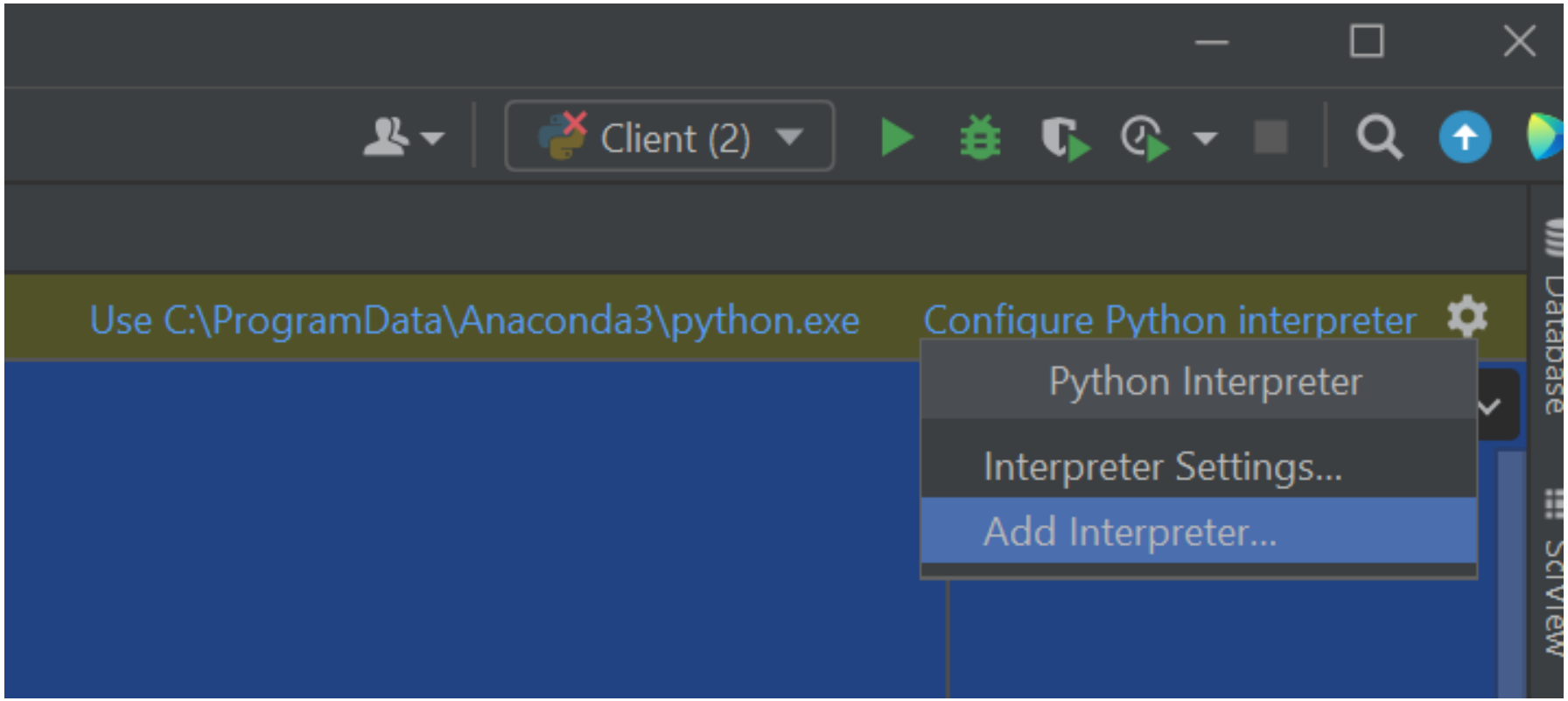
IS_TP1_Python 〉 REST 〉 python 〉 Books_ws_rest.py

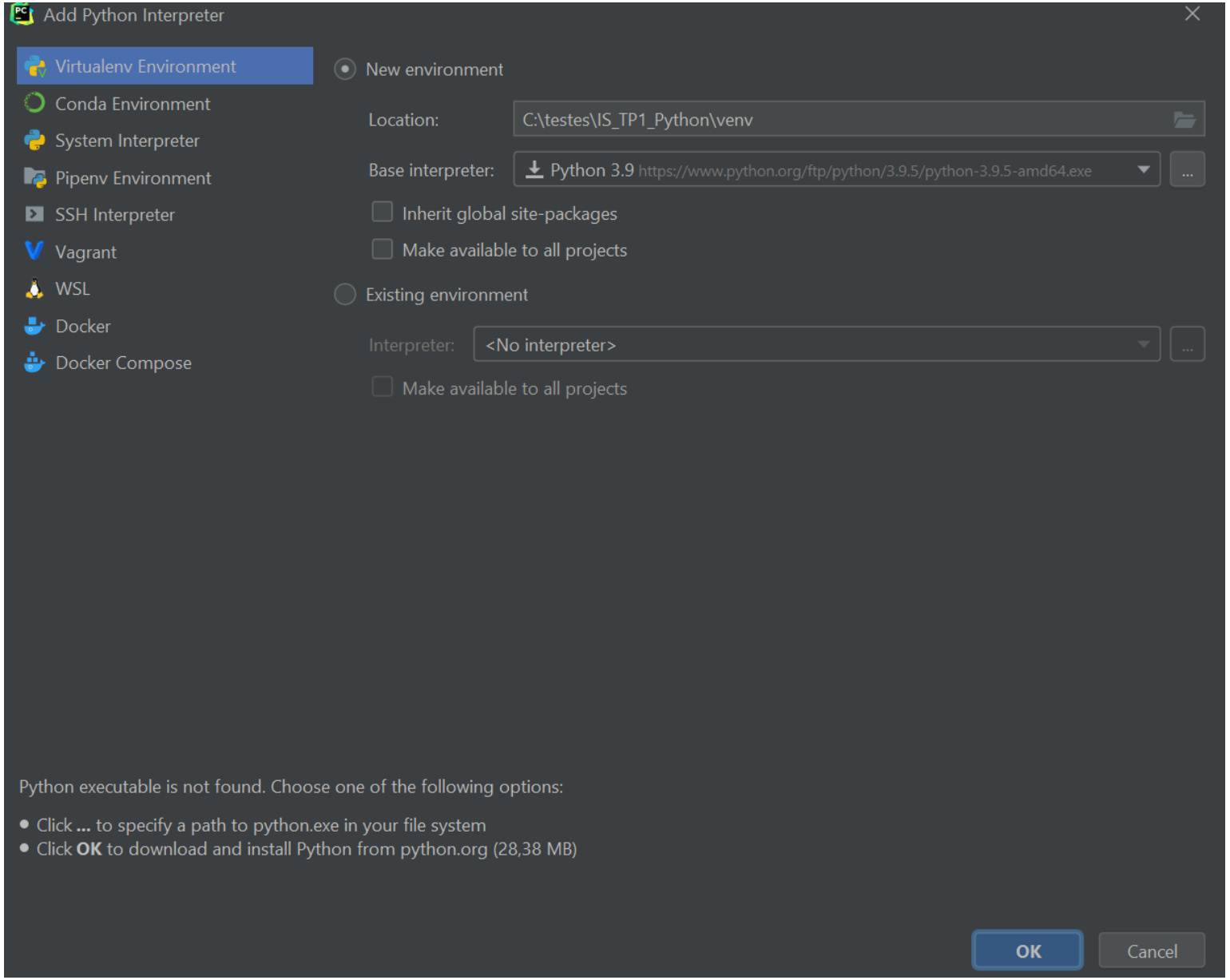
Client (2)

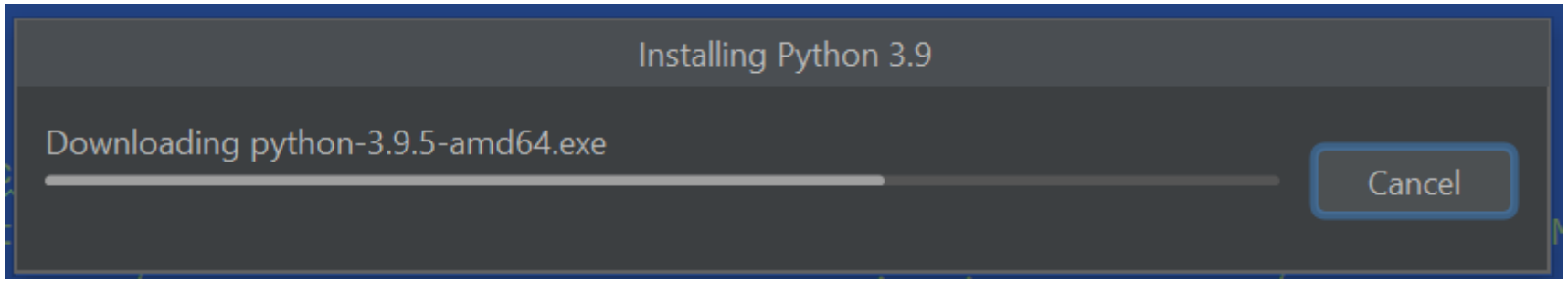
Books_ws_rest.py

No Python interpreter configured for the project — Use C:\ProgramData\Anaconda3\python.exe — Configure Python interpr

```python
from xml.dom import minidom
from flask import Flask, request
from lxml import etree
from werkzeug.wrappers import Response
app = Flask(__name__)


@app.route('/')
def index():
    html = "<h3>Esta aplicação contém os seguintes métodos: <h3>"
    html = html + "<a href='/getBooks/books'>XML Parse - Ficheiro books.xml</a><p>Resposta em XML e HTML</p><br>"
    html = html + "<a href='/getBooks/booksXpath'>XML Parse e XPATH - Ficheiro books.xml</a>"
    return html

@app.route('/getBooks/books', methods=['GET'])
def getBooks():
    typeRequest = request.args.get('type')
    if typeRequest.upper() == "HTML":
        xmldoc = minidom.parse('../files/books.xml')
        itemlist = xmldoc.getElementsByTagName('book')
        data = ""
```


Client (2)
Use C:\ProgramData\Anaconda3\python.exe
Configure Python interpreter
Python Interpreter
Interpreter Settings...
Add Interpreter...
Database
SciView


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:
Python 3.9 https://www.python.org/ftp/python/3.9.5/python-3.9.5-amd64.exe
Inherit global site-packages
Make available to all projects
Existing environment
Interpreter:
<No interpreter>
Make available to all projects
Python executable is not found. Choose one of the following options:
Click ... to specify a path to python.exe in your file system
Click OK to download and install Python from python.org (28,38 MB)
OK
Cancel


Installing Python 3.9
Downloading python-3.9.5-amd64.exe
Cancel

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;