Curso:	Ciência da Computação / Noite		
Disciplina:	DSD (8º Período)	Valor	10,0
Professor (a):	Flávio Velloso Laper		
Nomes:	Ana Clara Pinho Shreiber Cassiano Medeiros Santana João Paulo Carneiro Aramuni	Nota	
Nº da Atividade/Nome:	Relatório		
Data de Entrega:			
Valor:	10,0 Pts.		

Questão 2 – Projeto

Serviço escolhido: **Validador de CPF**
Utilização do serviço de validação de CPF através de Web Service.
O serviço utiliza o estilo *Document* e foi implementado no servidor *Tomcat*.
Tomcat foi configurado para trabalhar com Transport Layer Security (TLS).
Protocolo: HTTPS e Porta: 443.

Projeto **Cliente**: Acessa o método de validação do CPF através de um Web Service.
Projeto **WebService**: Provê o método de validação do CPF como serviço.

Links de Acesso:

Tomcat : https://localhost/ (Porta: 443 - SSL)
Login e senha: admin / admin

Cliente: https://localhost/Cliente/CPFServlet
Login e senha: convidado / convidado

WebService: https://localhost/WebService/cpf

WSDL: https://localhost/WebService/cpf?wsdl

Configurações de Segurança do Tomcat:

conf/server.xml

```
<!-- Define a blocking Java SSL Coyote HTTP/1.1 Connector on port 443 -->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
port="443" minSpareThreads="5" maxSpareThreads="75"
enableLookups="true" disableUploadTimeout="true"
acceptCount="100" maxThreads="200"
scheme="https" secure="true" SSLEnabled="true"
keystoreFile="C:\senha\key.kdb" keystorePass="facefumec"
clientAuth="false" sslProtocol="TLS"/>
```

Obs.: Todas as demais portas do arquivo são modificadas para **443**.

conf/tomcat-users.xml

```
<tomcat-users>

 <!-- Administrador -->
 <user name="admin" password="admin" roles="manager-gui" />

 <!-- Usuário Convidado -->
 <role rolename="membro"/>
 <user username="convidado" password="convidado" roles="membro"/>

</tomcat-users>
```

Configurações de Segurança do Cliente:

WebContent\WEB-INF\web.xml

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>securedapp</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Restrito</web-resource-name>
        <url-pattern>/CPFServlet</url-pattern>

        <http-method>DELETE</http-method>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>PUT</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>membro</role-name>
    </auth-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
<security-role>
    <role-name>membro</role-name>
</security-role>
```

Geração de Certificados:

Diretório de certificados: C:\senha
Arquivos: **key.cer, key.csr**, **key.kdb**

Comandos no CMD:

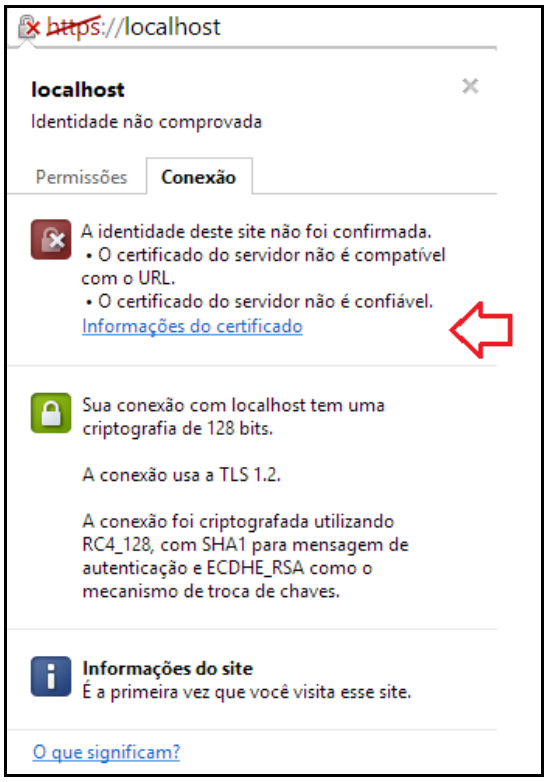
1) Geração do KDB

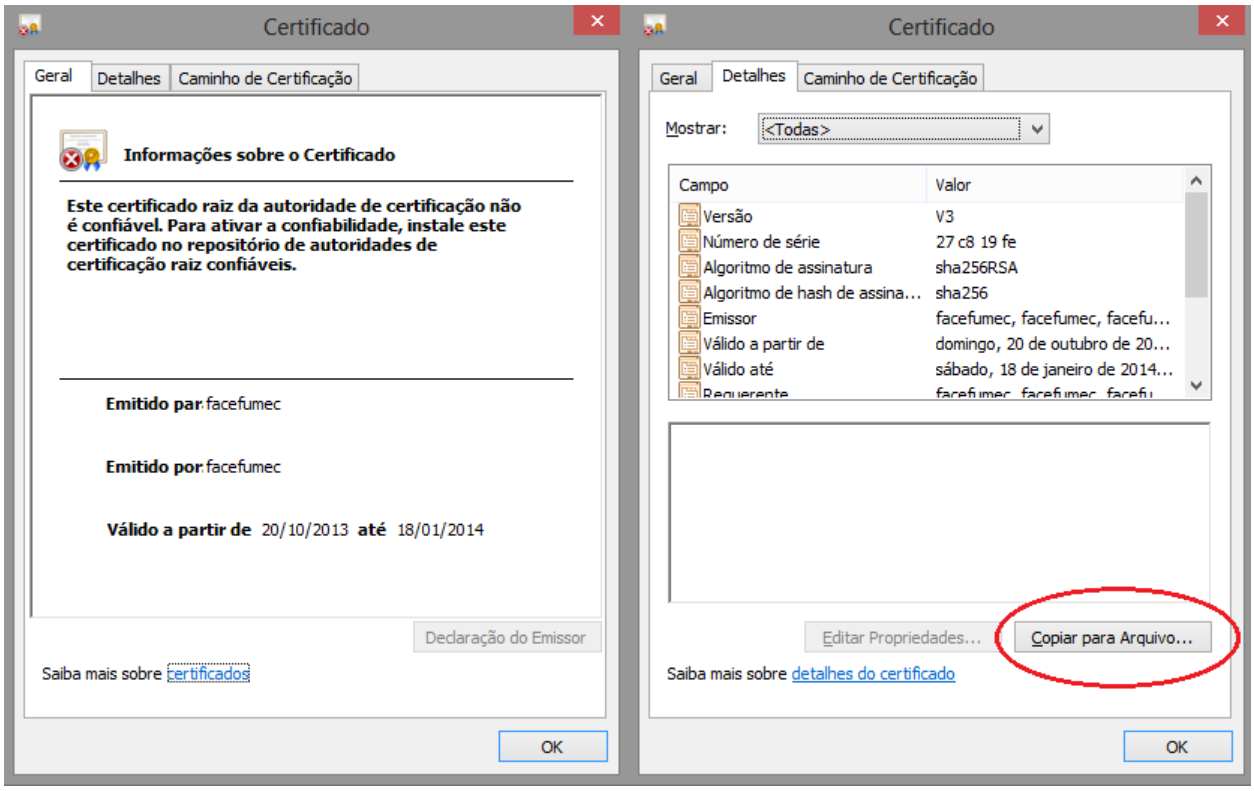
keytool -genkey -alias certificado -keyalg RSA -keystore C:\senha\key.kdb

Preenchimento dos dados:
What is your first and last name?
[Unknown]: facefumec
What is the name of your organizational unit?
[Global Sign]: facefumec
What is the name of your organization?
[Global Sign]: facefumec
What is the name of your City or Locality?
[London]: facefumec
What is the name of your State or Province?
[London]: facefumec
What is the two-letter country code for this unit?
[GB]: FF

2) Geração do CSR
keytool -certreq -keyalg RSA -alias tomcat -file C:\senha\key.csr -keystore C:\senha\key.kdb

3) Geração do CER
3.1 Método de obtenção: Ícone da URL do browser no endereço: https://localhost/





3.2 Aba detalhes > Copiar para Arquivo > C:\senhaz\key.cer

4) Importar as informações do certificado (key.cer) para o arquivo key.kdb.

keytool -import -keystore C:\senha\key.kdb -alias Root -trustcacerts -file C:\senha\key.cer

5) Importar como um certificado confiável.

keytool -import -trustcacerts -keystore C:\senha\key.kdb -alias tomcat -file C:\senha\key.cer

Obs.: Senha requerida é a mesma cadastrada no passo 1: **facefumec**.

6) Visualizar informações do certificado instalado.

keytool -list -keystore C:\senha\key.kdb -v > C:\senha\list.txt

7) Importar certificado para o arquivo cacerts do Java.

keytool -import -noprompt -trustcacerts -alias tomcat -file c:\senha\key.cer -keystore C:\Program Files (x86)\Java\jdk1.7.0_45\jre\lib\security\cacerts -storepass changeit

Senha padrão do Keystore do Java: **changeit**

Obs.: O sétimo passo é **crítico**, caso não retorne sucesso, a aplicação retornará exceção e erro 500 será apontado pelo protocolo HTTP.

Nota: Caso a URL C:\Program Files (x86)\Java\jdk1.7.0_45\jre\lib\security\cacerts não for lida pelo sistema corretamente, basta mover o arquivo cacerts para um diretório mais simples C:\temp e depois de executar o comando substituí-lo pelo arquivo original da pasta. Utilizar %JAVA_HOME%\jre\lib\security é outra opção válida.

WEB SERVICE - Geração de Artefatos com wsgen

Dois casos de uso comuns para a ferramenta wsgen:

- Gerar **artefatos** portáteis JAX-WS (arquivos Java) para a implantação do serviço web.
- Gerar arquivos **WSDL** e **XSD**, para testes ou desenvolvimento de clientes de serviços web.

Para gerar todos os artefatos portáteis JAX-WS para a classe de implementação web service utilizar o comando:

```
wsgen -verbose -keep -cp . nomedopacote.nomedaclasse
```

```
Note:   ap round: 1
[ProcessedMethods Class: nomedopacote.nomedaclasse]
[should process method: nomedometodo hasWebMethods: true ]
[endpointReferencesInterface: false]
[declaring class has WebSevice: true]
[returning: true]
[WrapperGen - method: nomedometodo()]
[method.getDeclaringType():nomedopacote.nomedaclasse]
[requestWrapper: nomedopacote.jaxws.nomedometodo]
[ProcessedMethods Class: java.lang.Object]
nomedopacote\jaxws\nomedometodo.java
nomedopacote\jaxws\nomedometodoResponse.java
Note:   ap round: 2
```

Neste caso serão gerados quatro arquivos:

nomedopacote\jaxws**nomedometodo.java**
nomedopacote\jaxws**nomedometodo.class**
nomedopacote\jaxws**nomedometodoResponse.java**
nomedopacote\jaxws**nomedometodoResponse.class**

Para gerar o arquivo WSDL (Web Services Description Language) e o arquivo xsd (XML Schema) para a classe web service, basta adicionar -wsdl no comando wsgen:

```
wsgen -verbose -keep -cp . nomedopacote.nomedaclasse -wsdl
```

Neste caso serão gerados seis arquivos:

nomedopacote\jaxws**nomedometodo.java**
nomedopacote\jaxws**nomedometodo.class**
nomedopacote\jaxws**nomedometodoResponse.java**
nomedopacote\jaxws**nomedometodoResponse.class**
nomedaclasseService_schema1.xsd
nomedaclasseService.wsdl

Com todos os arquivos prontos, basta publicar com **endpoint publisher**.

```
package nomedopacote.endpoint;

import javax.xml.ws.Endpoint;
import nomedopacote.nomedaclasse;

//Endpoint publisher
public class WsPublisher{

	public static void main(String[] args) {
	Endpoint.publish("https://localhost/WebService/cpf", new nomedaclasse());

	System.out.println("Service is published!");
}

}
```

A interface do terminal de serviço ou service endpoint interface (SEI) é uma classe de interface Java que define os métodos a serem expostos como um serviço web.

CLIENTE - Geração de Artefatos com wsimport

A ferramenta **wsimport** é usada para analisar um arquivo WSDL (Web Services Description Language) e gerar os arquivos necessários (JAX-WS artefatos portáteis) para o **cliente** de web service acessar os serviços da web publicados. Esta ferramenta wsimport está disponível na pasta JDK / bin.

O arquivo WSDL citado está disponível em: https://localhost/WebService/cpf?wsdl

Comando para a ferramenta wsimport analisar o arquivo WSDL e gerar os artefatos (arquivos Java) **cliente** para acessar o serviço disponibilizado pelo web service:

```
wsimport -keep -verbose https://localhost/WebService/cpf?wsdl

parsing WSDL...

generating code...
nomedopacote\nomedaclasse.java
nomedopacote\nomedaclasseImplService.java
```

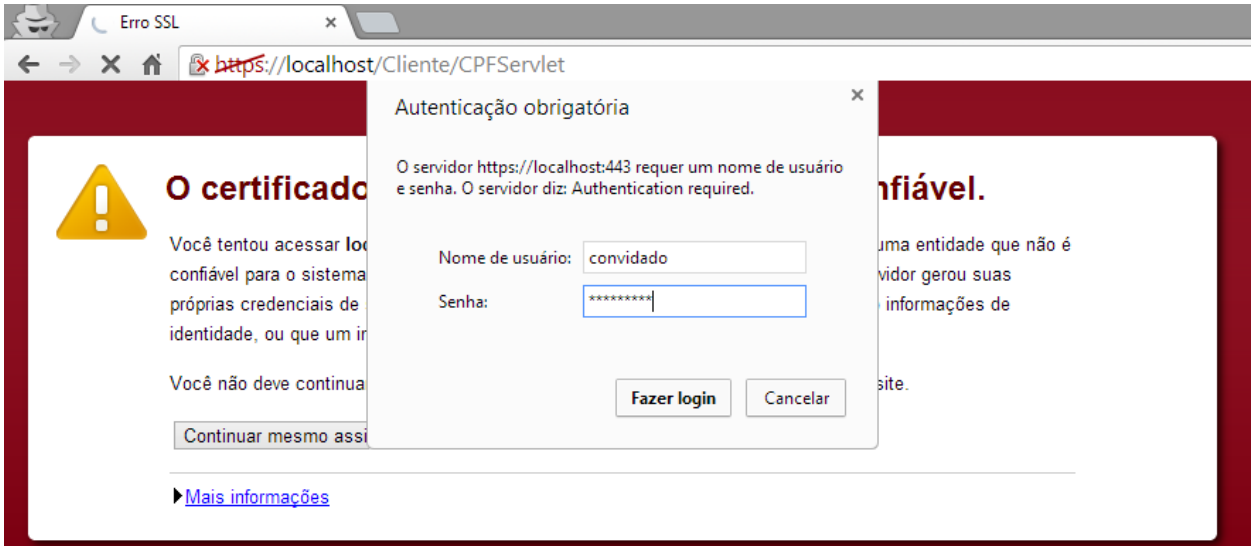
Neste caso, dois arquivos serão gerados pela ferramenta wsimport.

Através dessas classes Java, o cliente poderá acessar os serviços/métodos disponibilizados pelo web service.

Telas do Sistema – **CLIENTE**

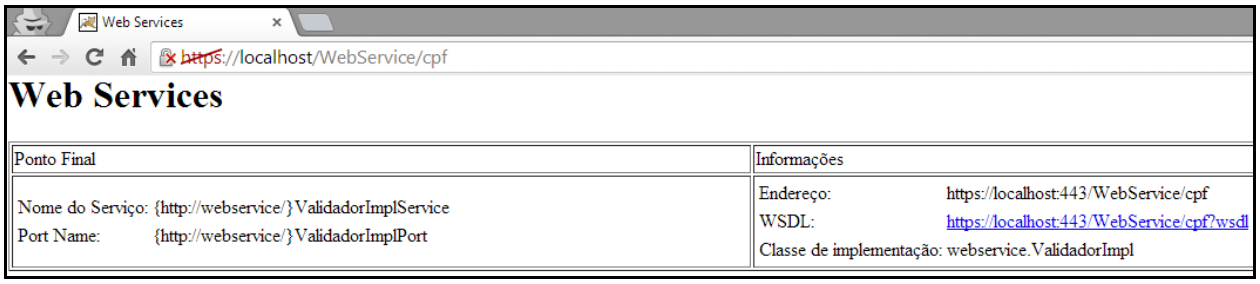
1) HTTPS – Certificado

2) Autenticação Obrigatória provida pelo próprio container (Tomcat).
Login e Senha padrão para usuários: convidado / convidado

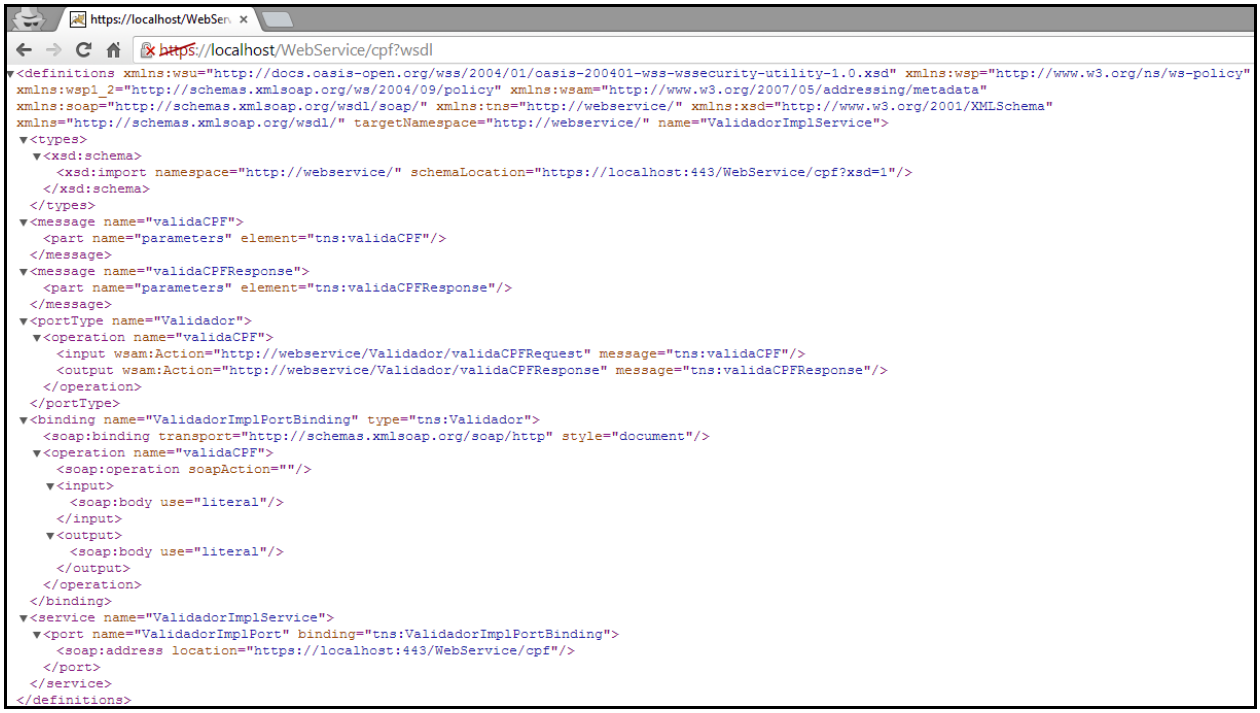


3) Tela principal do sistema:

Telas do Sistema – **WebService**



Telas do Sistema – **WSDL**



Libs extras necessárias do Projeto:

Cliente - \Cliente\WebContent\WEB-INF\lib

jsp-api.jar
servlet-api.jar

WebService - \WebService\WebContent\WEB-INF\lib

FastInfoset.jar
gmbal-api-only.jar
ha-api.jar
javax.annotation-api.jar
javax.xml.soap-api.jar
jaxb-api.jar
jaxb-core.jar
jaxb-impl.jar
jaxb-jxc.jar
jaxb-xjc.jar
jaxws-api.jar
jaxws-rt.jar
jaxws-tools.jar
jsr181-api.jar
management-api.jar
mimepull.jar
policy.jar
resolver.jar
saaj-impl.jar
stax2-api.jar
stax-ex.jar
streambuffer.jar
woodstox-core-asl.jar

Handler – Cliente

O handler do Cliente irá adicionar o IP Cliente no cabeçalho SOAP da mensagem para que o handler do web service possa recuperar o IP do Cliente e compará-lo aos IPs da lista de IPs válidos do seu web.xml.

Cliente/src/cliente/handler/handles.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<javaee:handler-chains xmlns:javaee="http://java.sun.com/xml/ns/javaee"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <javaee:handler-chain>
        <javaee:handler>
            <javaee:handler-class>cliente.handler.IpInjector
            </javaee:handler-class>
        </javaee:handler>
    </javaee:handler-chain>
</javaee:handler-chains>
```

Classe Java do Cliente, responsável por adicionar o IP Cliente no cabeçalho SOAP da mensagem: **Cliente/src/cliente/handler/IpInjector.java**

Handler – Web Service

O handler do Web Service irá recuperar o IP do cliente através do cabeçalho SOAP da mensagem e comparar o IP com todos os outros IPs cadastrados previamente no arquivo web.xml.

WebService/src/webservice/handler/handlers.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<javaee:handler-chains xmlns:javaee="http://java.sun.com/xml/ns/javaee"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <javaee:handler-chain>
        <javaee:handler>
            <javaee:handler-class>webservice.handler.IpValidator
            </javaee:handler-class>
        </javaee:handler>
    </javaee:handler-chain>
</javaee:handler-chains>
```

Classe Java do Web Service, responsável por recuperar o IP Cliente no cabeçalho SOAP da mensagem e validar o IP através da lista de IPs válidos do web.xml: **WebService/src/webservice/handler/IpValidator.java**

Caso o IP Cliente **não** esteja cadastrado no web.xml, a aplicação retornará ERRO 500 e o IpValidator retornará a mensagem:

Endereço IP inválido, acesso negado.

Caso o IP Cliente esteja cadastrado no web.xml, a aplicação irá acessar o serviço normalmente.

O acesso à aplicação cliente, uma vez que o usuário possua o login e senha (convidado/convidado), é livre. Nenhum handler irá impedir o usuário de visualizar a página principal da aplicação cliente. Porém, a utilização do serviço provido pelo web service é de restrito acesso à apenas os IPs cadastrados no web.xml.

IPs cadastrados no web.xml:
WebService\WebContent\WEB-INF\web.xml

```
<env-entry>
    <env-entry-name>IP_Casa_1</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>192.168.0.1</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>IP_Casa_2</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>192.168.0.2</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>IP_Fumec</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>172.16.0.1</env-entry-value>
</env-entry>
```

Após o usuário inserir o CPF a ser validado e clicar em **Validar**, o handler do cliente será acionado para adicionar o IP no cabeçalho SOAP da mensagem e posteriormente o handler do web service será acionado para recuperar e validar o IP Cliente.

Abaixo, página de erro demonstrando exceção para IP que não está cadastrado no web.xml.
Página de erro da aplicação Cliente, todos os erros do tipo ERRO 500 são mostrados aqui:

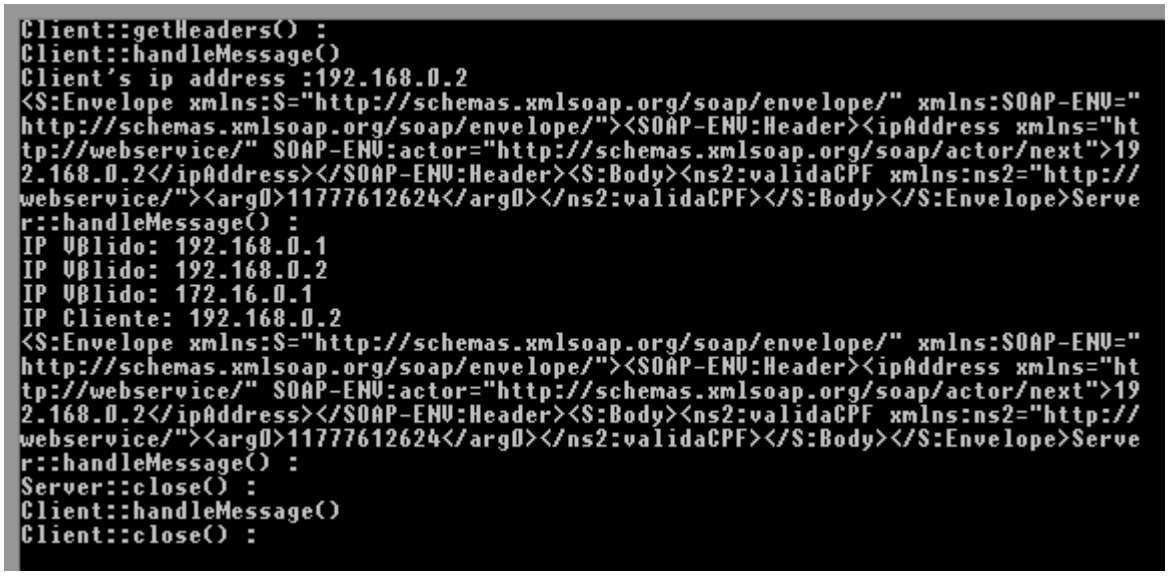
Cliente\WebContent\WEB-INF\web.xml

```
<error-page>
    <error-code>500</error-code>
    <location>/erro.jsp</location>
</error-page>
```

Fluxo de funcionamento dos handlers:

Diversos comandos “System.out.println” foram colocadas nos handlers do Cliente e do WebService para melhor visualizar o fluxo de acontecimentos.

Console do Tomcat:



Log do Tomcat:

Client::getHeaders() :
Client::handleMessage()
Client's ip address :192.168.0.2

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header><ipAddress xmlns="http://webservice/" SOAP-ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">192.168.0.2</ipAddress></SOAP-ENV:Header><S:Body><ns2:validaCPF xmlns:ns2="http://webservice/"><arg0>11777612624</arg0></ns2:validaCPF></S:Body></S:Envelope>

Server::handleMessage() :
IP Válido: 192.168.0.1
IP Válido: 192.168.0.2
IP Válido: 172.16.0.1
IP Cliente: 192.168.0.2

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header><ipAddress xmlns="http://webservice/" SOAP-ENV:actor="http://schemas.xmlsoap.org/soap/actor/next">192.168.0.2</ipAddress></SOAP-ENV:Header><S:Body><ns2:validaCPF xmlns:ns2="http://webservice/"><arg0>11777612624</arg0></ns2:validaCPF></S:Body></S:Envelope>

Server::handleMessage() :
Server::close() :
Client::handleMessage()
Client::close() :