Sistemas Web Seguros

Atividades – Semana 3

ATIVIDADE FORMATIVA (ON-LINE / PRÉ-AULA)

ATIVIDADE FORMATIVA: SISRH – Construção de *web service* SOAP
ESTUDANTE:

Atividade 1 – Transformando SISRH em um sistema *web service* SOAP – Resolvida

Faça os ajustes necessários para transformar o SISRH, atualmente de um sistema *web* tradicional em um sistema *web service* SOAP.

Entregas:

1. Trechos dos códigos que foram incluídos ou modificados.
2. Testes dos serviços que foram construídos, exibindo as entradas e saídas da ferramenta SoapUI.

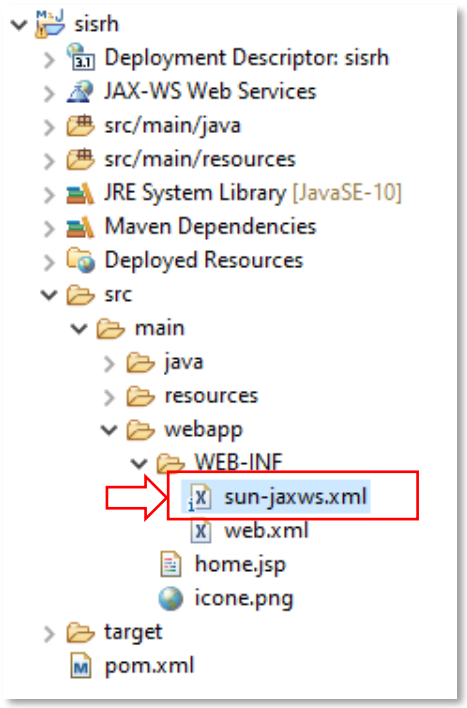
Resolução:

Vamos apresentar o passo a passo dos ajustes necessários.

1. Abra o arquivo **pom.xml** e inclua a dependência da especificação JaxWS (jaxws-rt versão 2.3.2). Segue exemplo, com destaque para o trecho a ser incluído (linhas coloridas).

```
<dependencies>

            <!-- JaxWS -->
            <dependency>
                    <groupId>com.sun.xml.ws</groupId>
                    <artifactId>jaxws-rt</artifactId>
                    <version>2.3.5</version>
            </dependency>

            <!-- Servlet Api -->
            <dependency>
                    <groupId>javax.servlet</groupId>
                    <artifactId>javax.servlet-api</artifactId>
                    <version>3.0.1</version>
                    <scope>provided</scope>
            </dependency>

            <!-- Banco HSQLDB -->
            <dependency>
                    <groupId>org.hsqldb</groupId>
                    <artifactId>hsqldb</artifactId>
                    <version>2.6.0</version>
            </dependency>

    </dependencies>
```

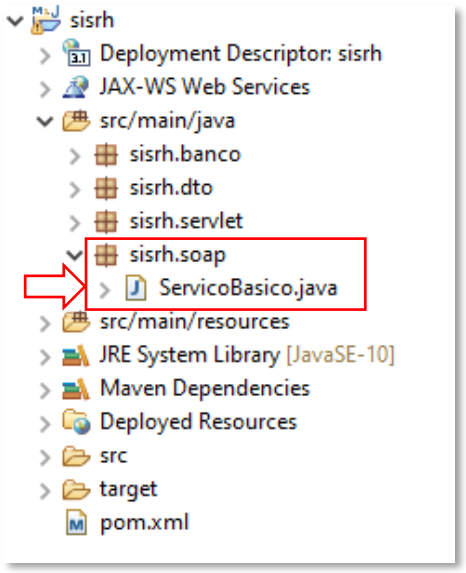
2. Crie o arquivo **sun-jaxws.xml** na pasta **WEB-INF**.



Criaremos um serviço supersimples para testar, chamado **Serviço Básico**. Portanto, informe, no arquivo **sun-jaxws.xml**, o *endpoint* dele.

```
<endpoints
	xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime" version="2.0">

	<endpoint name="SISRHService"
		implementation="sisrh.soap.ServicoBasico" url-pattern="/soap/basico" />

</endpoints>
```

3. Crie a classe **ServicoBasico** dentro do pacote **soap**.



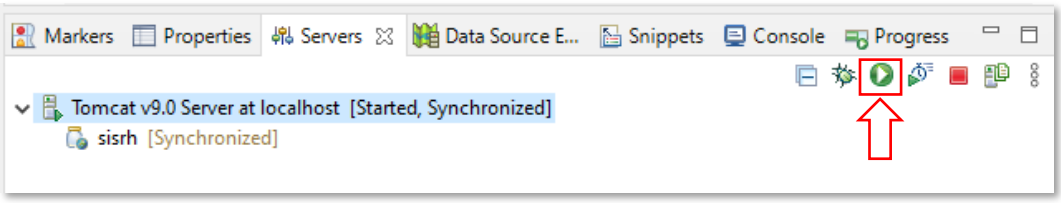
4. Crie uma operação chamada **ping**, que retornará uma string **pong**, mais um identificador único. Isso vai nos ajudar a fazer o primeiro teste. Segue código do serviço e da operação **ping**.

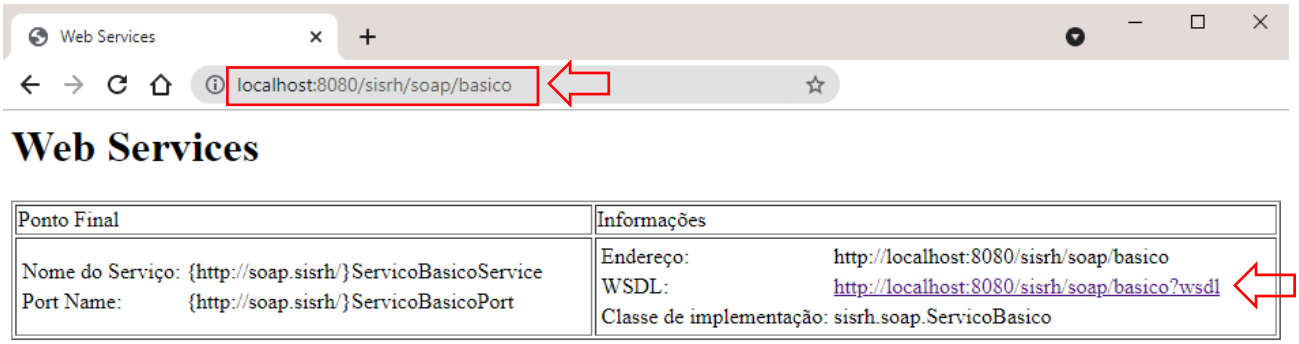
```
package sisrh.soap;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.UUID;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

@WebService
@SOAPBinding(style = Style.RPC)
public class ServicoBasico {

	@WebMethod(action = "ping")
	public String ping() {
		UUID uuid = UUID.randomUUID();
		return "pong: " + uuid;
	}
}
```

5. Outra operação será o retorno da hora do sistema. Dessa forma, poderemos saber o horário atual do SISRH. Na classe **ServicoBasico**, inclua o método **dataHoraServidor**.

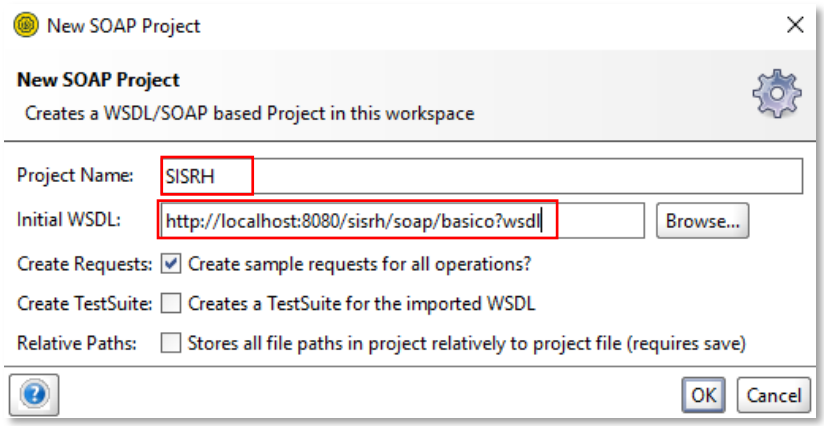
```
@WebMethod(action = "dataHoraServidor")
public String dataHoraServidor() {
	String pattern = "dd/MM/YYYY - HH:mm:ss";
	SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
	return simpleDateFormat.format(new Date());

}
```

6. Tudo pronto! O SISRH agora é um sistema *web service* SOAP com duas operações. Inicie o Apache Tomcat e acesse a URL do serviço.



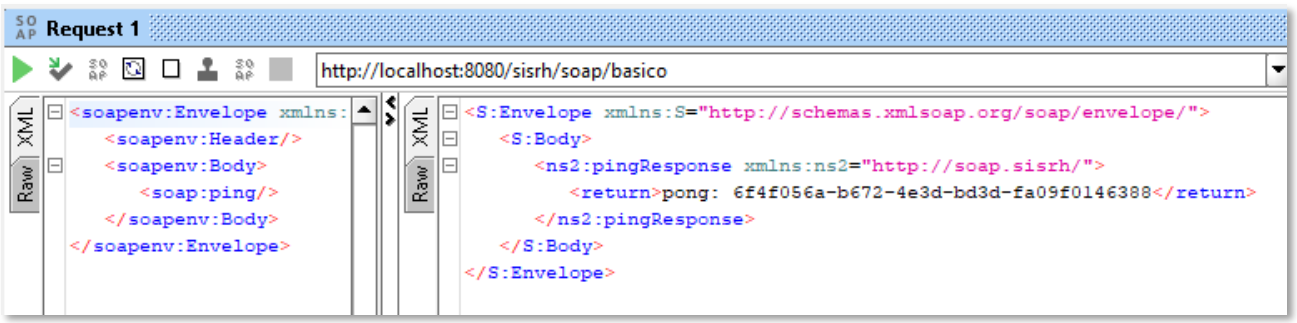


7. Copie a URL do documento WSDL (http://localhost:8080/sisrh/soap/basico?wsdl) e crie um projeto SoapUI chamado **SISRH**. Teste o serviço com as duas operações criadas.

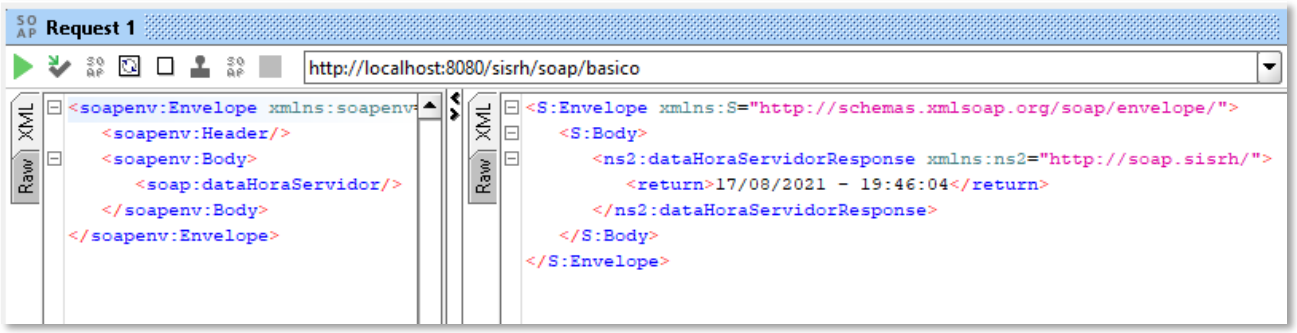


8. Segue o teste das operações.

Teste do serviço **ping**



Teste do serviço **dataHoraServidor**



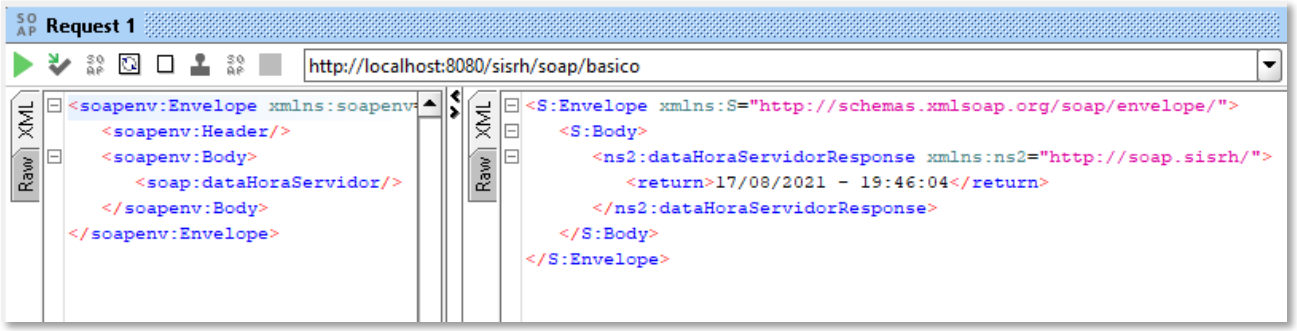
Este exercício foi resolvido, mas você deve fazer a entrega para compreender o processo.

Resposta:

1. Testes dos serviços que foram construídos, exibindo as entradas e saídas da ferramenta SoapUI.

Teste do serviço **ping**

Teste do serviço **dataHoraServidor**



Atividade 2 – Serviço para listar empregados com *login* e senha – Resolvida

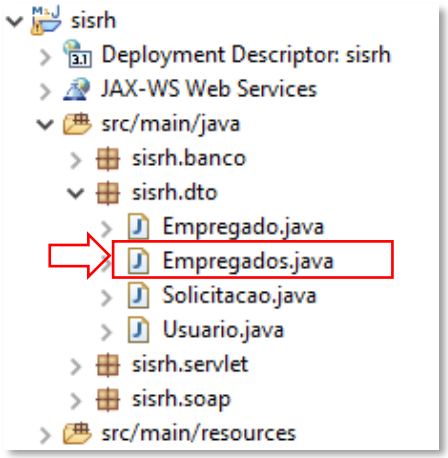
Crie uma operação para listar todos os empregados do SISRH. A operação deve ser autenticada de tal forma que apenas usuários autenticados poderão acessar a listagem.

Resolução:

1. Abra a classe **Empregado** e insira as anotações nos campos que serão apresentados na listagem de empregados.

```
import javax.xml.bind.annotation.*;

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
public class Empregado {

        @XmlElement(name = "matricula")
        private String matricula;

        @XmlElement(name = "nome")
        private String nome;

        @XmlElement(name = "admissao")
        private Date admissao;

        @XmlElement(name = "desligamento")
        private Date desligamento;

        @XmlElement(name = "salario")
        private Double salario;

        ...
```

2. No pacote **dto**, criaremos uma classe chamada **Empregados**, que representará a coleção de empregados a ser retornados pelo serviço.

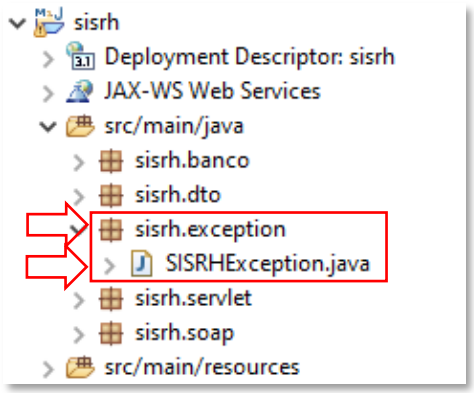


3. A classe **Empregados** terá apenas uma lista (*list*) de objetos do tipo **Empregado**.

```
package sisrh.dto;

import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
```

```java
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
public class Empregados {

    @XmlElement(name = "empregado")
    private List<Empregado> empregados = new ArrayList<Empregado>();

    public List<Empregado> getEmpregados() {
            return empregados;
    }
}
```

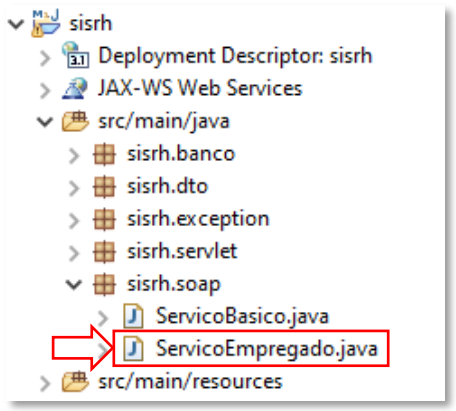
4. Você lembra que falamos da forma de tratamento de falhas em *web services* SOAP? Como teremos um serviço autenticador, vamos tratar da falha caso o serviço seja chamado sem um usuário e senha válidos. Criaremos um pacote chamado **exception** para organizar nossas mensagens de erro, como também a primeira classe, chamada **SISRHException**, que enviará mensagens gerais do sistema.



5. A classe terá a seguinte estrutura:

```java
package sisrh.exception;

import javax.xml.ws.WebFault;

@WebFault(name = "SISRH")
public class SISRHException extends Exception {

    private static final long serialVersionUID = 1L;

    public SISRHException(String mensagem) {
            super(mensagem);
    }
}
```

6. Crie a classe **ServicoEmpregado** no pacote **soap**.




```
package sisrh.soap;

import java.util.List;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

import sisrh.banco.Banco;
import sisrh.dto.Empregado;
import sisrh.dto.Empregados;


@WebService
@SOAPBinding(style = Style.RPC)
public class ServicoEmpregado {

	@WebMethod(action = "listar")
	public Empregados listar() throws Exception {

		Empregados empregados = new Empregados();

		List<Empregado> lista = Banco.listarEmpregados();
		for(Empregado emp: lista) {
			empregados.getEmpregados().add(emp);
		}
		return empregados;
	}

}
```

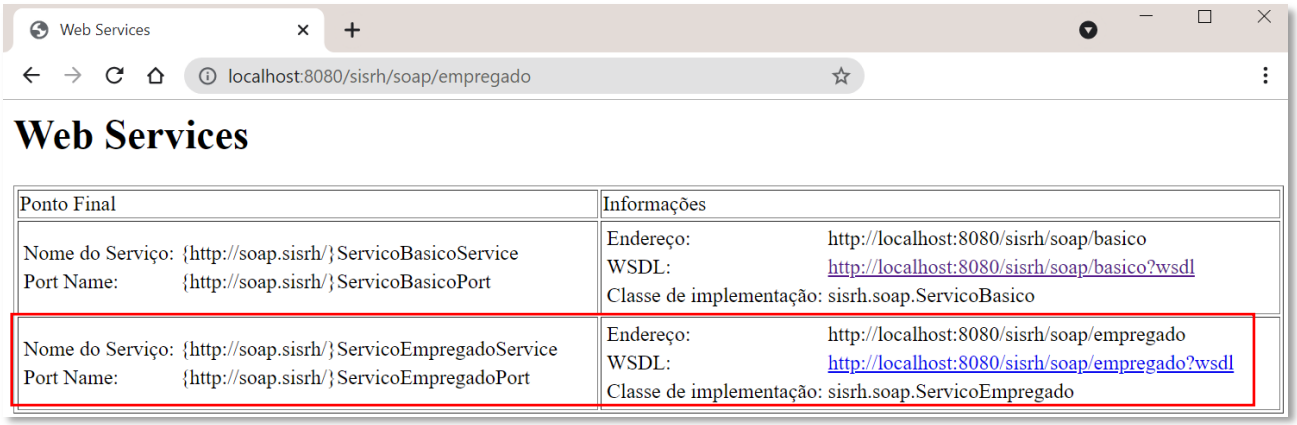
Observe que precisamos converter a lista de empregados (`List<Empregado>`) no objeto do tipo **Empregados**. Pode não ser a solução mais elegantes, mas será necessário para que a nossa classe **Banco** não dependa das classes **Soap**, pois no futuro vamos utilizá-la nos *web services* REST. Por isso, sugerimos que sempre faça essa conversão.

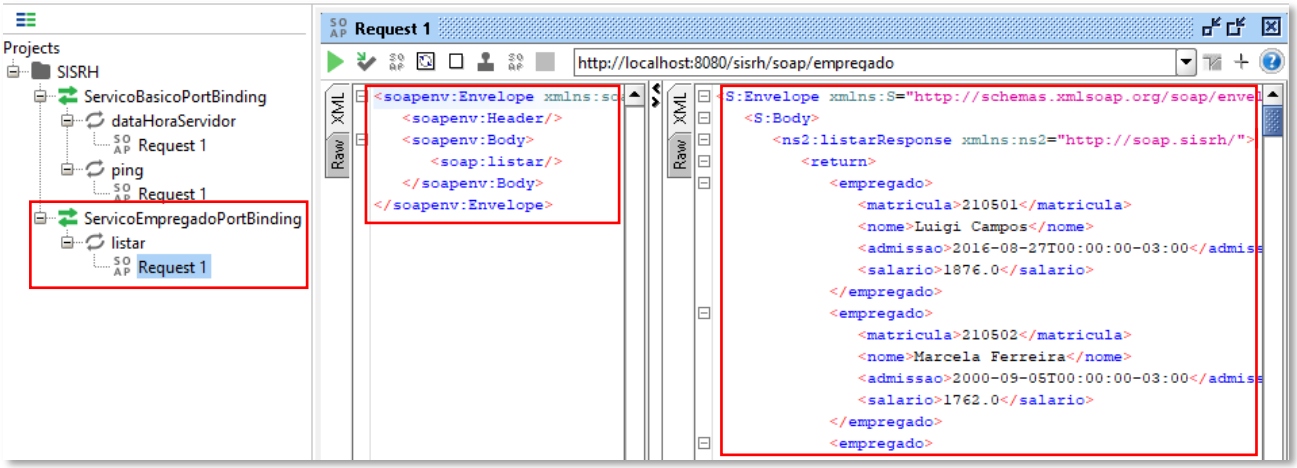
7. Inclua o **ServicoEmpregado** no arquivo **sun-jaxws.xml**.

```
<endpoints
	xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime" version="2.0">

	<endpoint name="SISRHService"
		implementation="sisrh.soap.ServicoBasico" url-pattern="/soap/basico" />
```

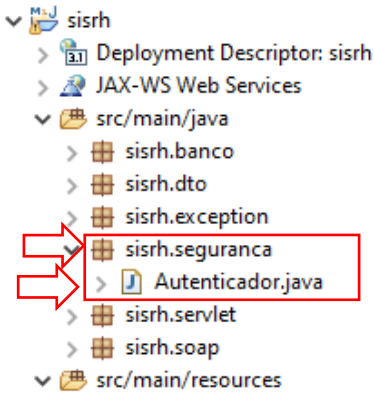
```
    <endpoint name="EmpregadoService"
            implementation="sisrh.soap.ServicoEmpregado"
            url-pattern="/soap/empregado">
    </endpoint>

</endpoints>
```

8. O serviço ainda não possui a autenticação, mas já está funcionando. Inicie o Tomcat e teste-o no SoapUI.





9. Inclua a autenticação, recebendo o usuário e senha por meio dos *headers* da requisição Soap, conforme vimos na Unidade de Estudo.
10. Para organizar o código, crie uma classe chamada **Autenticador** em um pacote que criaremos chamado **segurança**.




```java
package sisrh.seguranca;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.sql.*;
import java.util.*;


import javax.xml.ws.WebServiceContext;
import javax.xml.ws.handler.MessageContext;

import sisrh.banco.Banco;
import sisrh.exception.SISRHException;

public class Autenticador {

	@SuppressWarnings("rawtypes")
	public static boolean autenticarUsuarioSenha(WebServiceContext context) throws Exception {
		MessageContext messageContext = context.getMessageContext();
		Map httpHeaders = (Map) messageContext.get(MessageContext.HTTP_REQUEST_HEADERS);

		if (!httpHeaders.containsKey("usuario")) {
			throw new SISRHException("Informe o usuario");
		}
		if (!httpHeaders.containsKey("senha")) {
			throw new SISRHException("Informe a senha");
		}
		String usuario = ((List) httpHeaders.get("usuario")).get(0).toString();
		String senha = ((List) httpHeaders.get("senha")).get(0).toString();
		if (!autenticarUsuarioSenha(usuario, senha)) {
			throw new SISRHException("Usuario e senha invalidos");
		}
		return true;
	}


	@SuppressWarnings("rawtypes")
	public static String getUsuario(WebServiceContext context) throws Exception {
		MessageContext messageContext = context.getMessageContext();
		Map httpHeaders = (Map) messageContext.get(MessageContext.HTTP_REQUEST_HEADERS);
		return ((List) httpHeaders.get("usuario")).get(0).toString();
	}

	private static boolean autenticarUsuarioSenha(String usuario, String senha) throws Exception {
		try {
			String senhaMd5 = md5(senha);
			Connection conn = Banco.getConexao();
			String sql = "SELECT nome, senha FROM Usuario WHERE nome = ? and senha = ?";

			PreparedStatement prepStmt = conn.prepareStatement(sql);
			prepStmt.setString(1, usuario);
			prepStmt.setString(2, senhaMd5);
```

```
                        ResultSet rs = prepStmt.executeQuery();
                        while (rs.next()) {
                                return true;
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return false;
        }

        public static String md5(String valor) throws Exception {
                String s = valor;
                MessageDigest m = MessageDigest.getInstance("MD5");
                m.update(s.getBytes(), 0, s.length());
                return "" + new BigInteger(1, m.digest()).toString(16);
        }

}
```

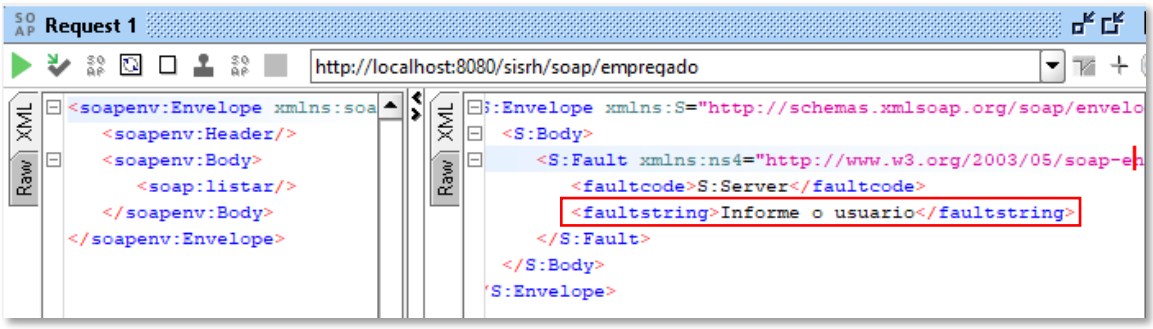
Observe que a senha do usuário não é salva na base de dados, apenas o respectivo *hash* MD5. Dessa forma, garantimos que ninguém tenha acesso direto às senhas. Quando o usuário informa sua senha, nós a convertemos no *hash* MD5 e verificamos na consulta SQL se o MD5 informado está na base para aquele usuário.

11. Crie a referência do WebServiceContext e chame o autenticador dentro do método **lista** de **ServiçoEmpregado**.

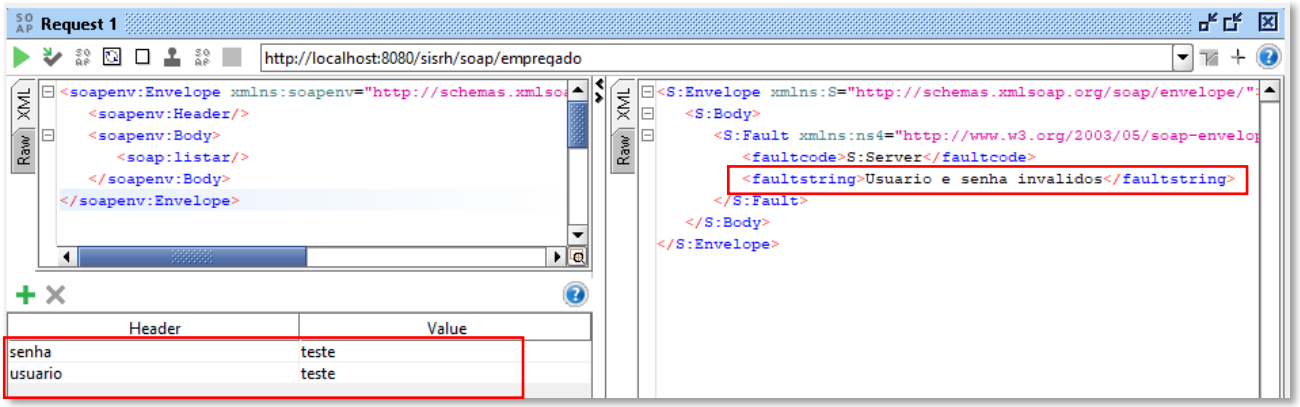
```
@WebService
@SOAPBinding(style = Style.RPC)
public class ServicoEmpregado {

        @Resource
        WebServiceContext context;

        @WebMethod(action = "listar")
        public Empregados listar() throws Exception {

                Autenticador.autenticarUsuarioSenha(context);

                Empregados empregados = new Empregados();

                List<Empregado> lista = Banco.listarEmpregados();
                for(Empregado emp: lista) {
                        empregados.getEmpregados().add(emp);
                }
                return empregados;
        }

}
```

12. Faça os testes simulando as diversas possibilidades no SoapUI.

Teste sem passar usuário



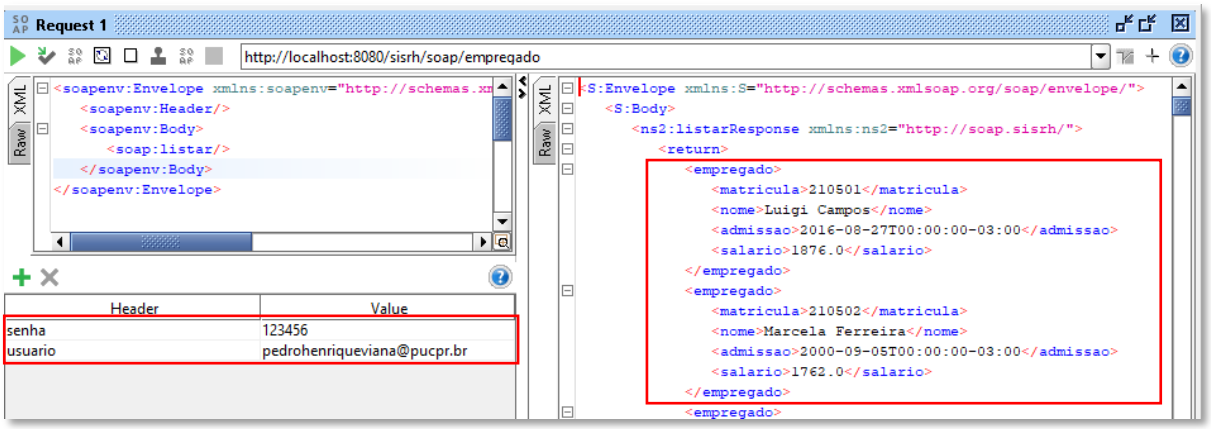
Teste informando usuário e senha inválidos



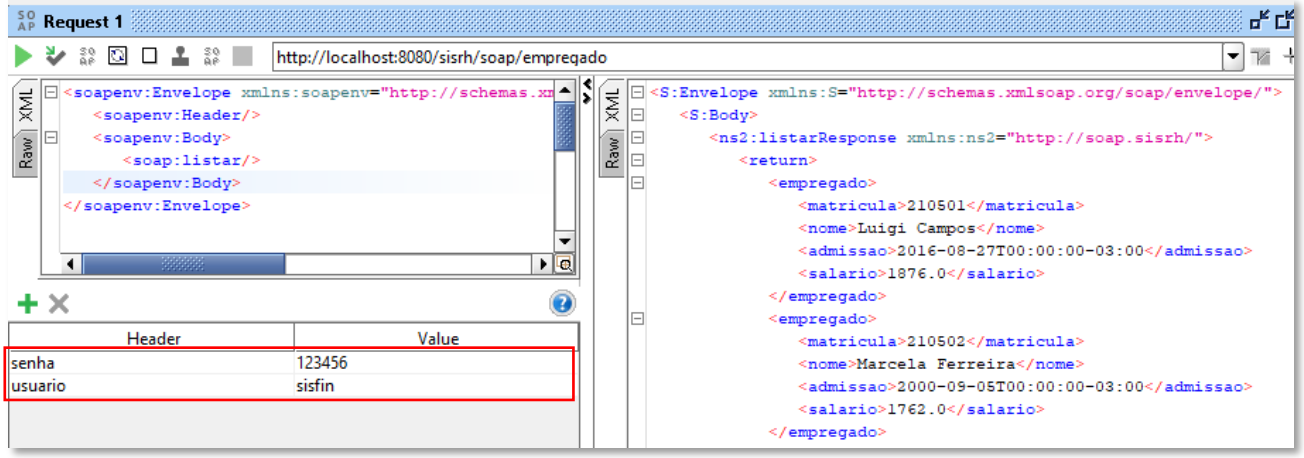
Teste informando usuário e senha válidos

Importante: criamos a base de dados com todas as senhas definidas com o valor “123456”.

Logo:



Teste do usuário de serviço **sisfin** (representa o sistema financeiro, que a partir de agora poderá acessar a lista de empregados)



Atividade 3 – Listar empregados ativos e inativos

No Serviço de Empregados, crie duas operações de listagem. A primeira listará os empregados ativos e a segunda, os empregados inativos. Ambas as operações retornarão a lista de empregados em ordem alfabética para o campo **nome**.

Algumas dicas:

- O empregado ativo é aquele que possui o campo **desligamento** igual a nulo.
- Crie na classe **Banco** um método para listar os empregados. Ele pode receber como parâmetro um valor *boolean* (*true* para ativos e *false* para inativos), daí você poderá ajustar a consulta SQL conforme a valor do parâmetro.
- Lembre-se de que as operações de empregado devem estar na classe **ServicoEmpregado**, cada uma sendo um método. Veja como seria a estrutura básica da classe:

```
@WebService
@SOAPBinding(style = Style.RPC)
public class ServicoEmpregado {

        @WebMethod(action = "listar")
        public Empregados listar() throws Exception {
        }
        @WebMethod(action = "listarAtivos")
        public Empregados listarAtivos() throws Exception {
        }
        @WebMethod(action = "listarInativos")
        public Empregados listarInativos() throws Exception {
        }
}
```

- Mantenha a mesma solução de autenticação da operação **listar**.

Atividade 4 – Novo serviço para listar as solicitações dos empregados

Construa um serviço para listar as solicitações de cada empregado. Esse serviço listará apenas as solicitações do empregado vinculado ao usuário logado. Por exemplo, se o empregado "Bianca Alves" tentar acessar o serviço com o seu usuário "biancaalves@pucpr.br", ela receberá apenas as solicitações vinculadas a ela.

Mais algumas dicas e lembretes:

- No pacote **soap**, sugerimos criar a classe **ServicoSolicitacao**, usando como modelo **ServicoEmpregado**.
- No pacote **dto**, você precisará criar a classe **Solicitacoes** para representar o conjunto de solicitações, usando como modelo a classe **Empregados**.
- Lembre-se de que a classe **Solicitacao** precisará das anotações para descrever seus campos; use como modelo a classe **Empregado**.
- A classe **Banco** terá um novo método para listar solicitações a partir do nome do usuário. Veja uma proposta:

```
public static List<Solicitacao> listarSolicitacoes(String usuario) throws SQLException {
```

Vamos passar o exemplo do SQL, que fará o *join* entre as tabelas Solicitacao, Empregado e Usuário.

```
SELECT * FROM Solicitacao as s
INNER JOIN Empregado as e ON s.matricula = e.matricula
INNER JOIN Usuario as u ON e.matricula = u.matricula
WHERE u.nome = ?
```

- Lembre-se de registrar o serviço no arquivo **sun-jaxws.xml**. Lá você informará um novo *endpoint* para acessar o serviço **ServicoSolicitacao**.
- Lembre-se de que precisa testar o serviço de modo autenticado. A senha de todos os usuários é "123456" e você consegue ver a lista de usuários por meio do arquivo **db_sisrh_dados.sql**, procurando pela *string* **@pucpr.br**.

PUCPR
GRUPO MARISTA