Development of a WebService (Spring Boot & SOAP)

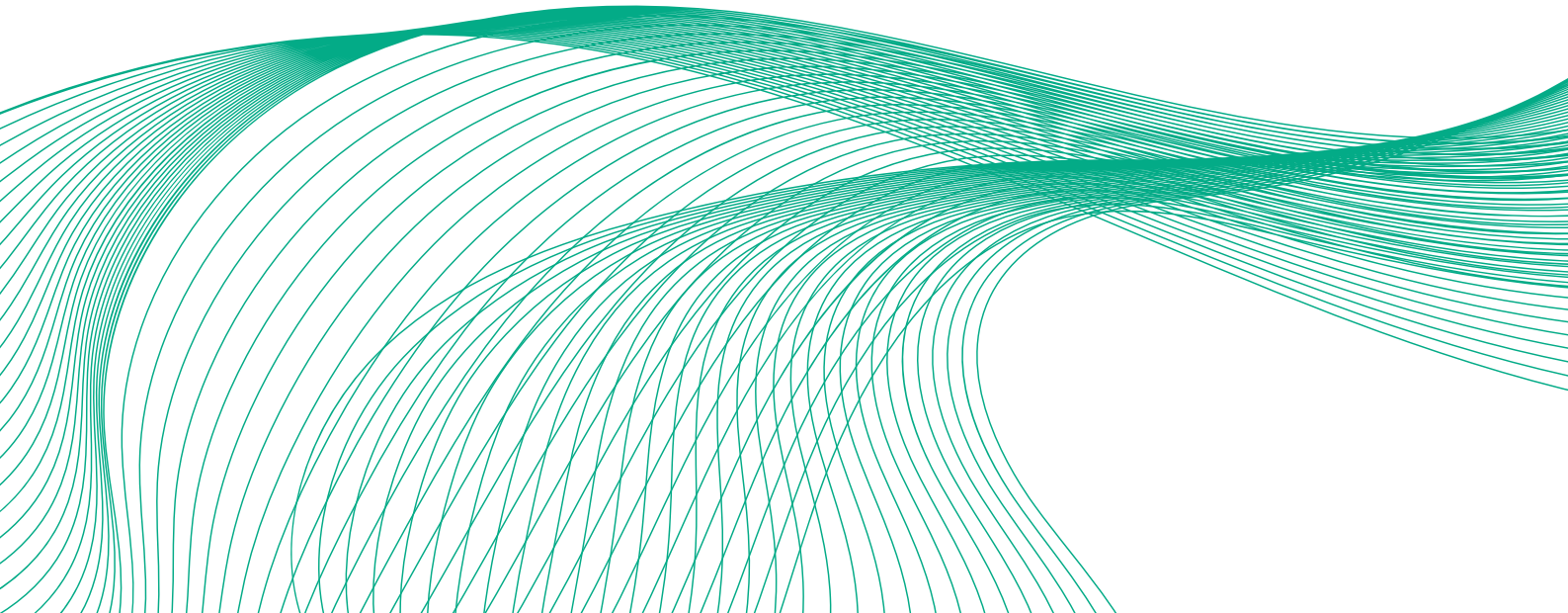
MELAKU GARCIA
GARY VACA
JUREK LOPEZ

INDEX

INTRODUCITION

In this practice, we will explore the development of a web service using two key technologies: Spring Boot and SOAP (Simple Object Access Protocol). Spring Boot is a Java application development framework that simplifies the process of creating Spring-based applications, providing minimal configuration and a quick way to start projects. On the other hand, SOAP is a standard communication protocol based on XML that allows interoperability between different systems over the web.

The purpose of this practice is to understand the basic concepts of web service creation, as well as to become familiar with development using Spring Boot and the implementation of SOAP-based services. Completing this practice, we will learn to configure a Spring Boot project, define and expose web services using SOAP, and perform tests to ensure their correct operation.



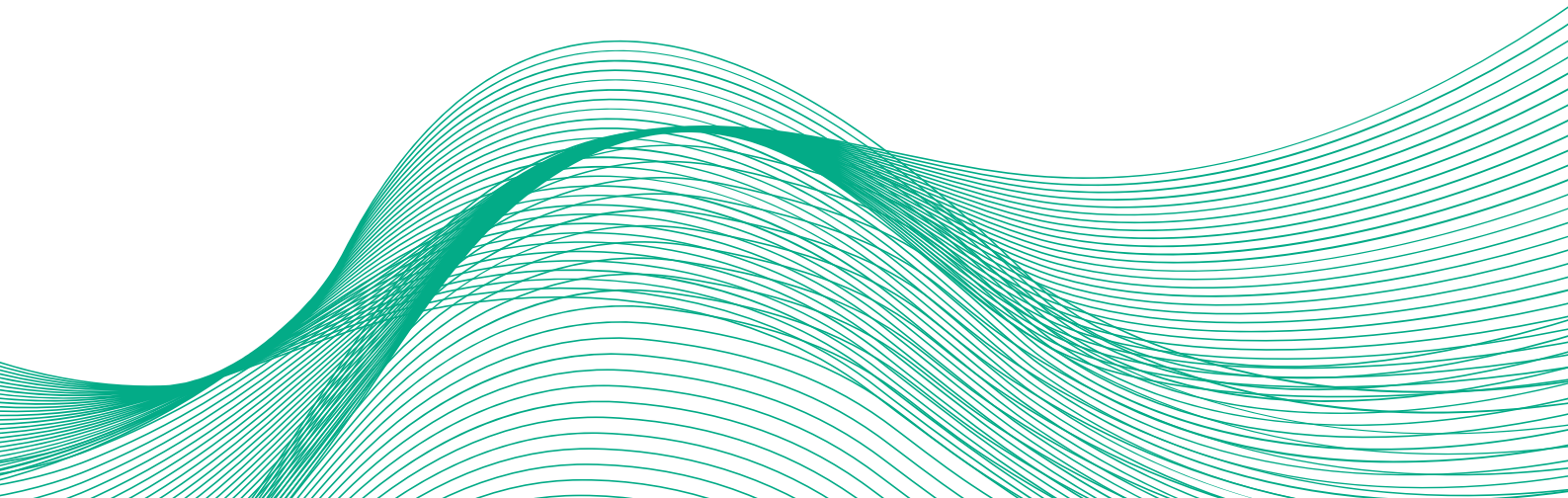
SPRING WEB SERVICES

DEFINITION

Spring Web Services (Spring-WS) is a product of the Spring community and is focused on creating document-driven web services. Spring Web Services aims to facilitate contract-first SOAP service development, allowing for the creation of flexible web services by using one of the many ways to manipulate XML.

KEY FEATURES:

- Powerful mapping
- XML API support
- Flexible XML Marshalling
- Reusing your Spring expertise
- Support for WS-Security
- Integration with Spring Security
- Apache license



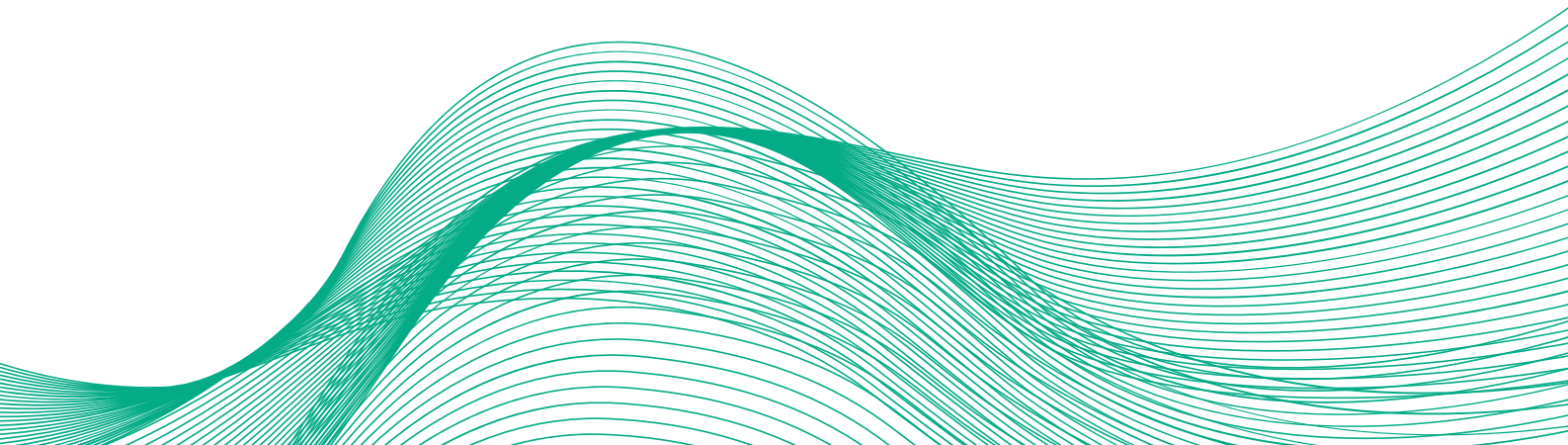
SOAP

DEFINITION

SOAP (Simple Object Access Protocol) is a lightweight protocol for exchanging information in decentralized and distributed environments. SOAP messages are the transmissions of information from senders to receivers. SOAP messages can be combined to create request/response patterns.

SOAP is an XML-based protocol that defines three parts in all messages:

- **Envelope** (sobre) . The header defines an infrastructure for describing what is in a message and how to process it.
- **Encoding Rules**. The set of encoding rules expresses instances of application-defined data types. Encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types.
- **Communication Styles**. Communications can follow either the RPC (Remote Procedure Call) or message-oriented (Document) format.
    - **Remote Procedure Call (RPC)**: Invocation of an operation that returns a result. It's typically used with SOAP encoding, which is not WS-I compliant.
    - **Document Style (Estilo de documento)**: Also known as document-oriented or message-oriented style. This style provides a lower layer of abstraction and requires more programming work.



SOAP

Example 1 SOAP Message Embedded in HTTP Request

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Following is the response message containing the HTTP message with the SOAP message as the payload:

Example 2 SOAP Message Embedded in HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

APP

Adding the Spring-WS dependency

XML Schema

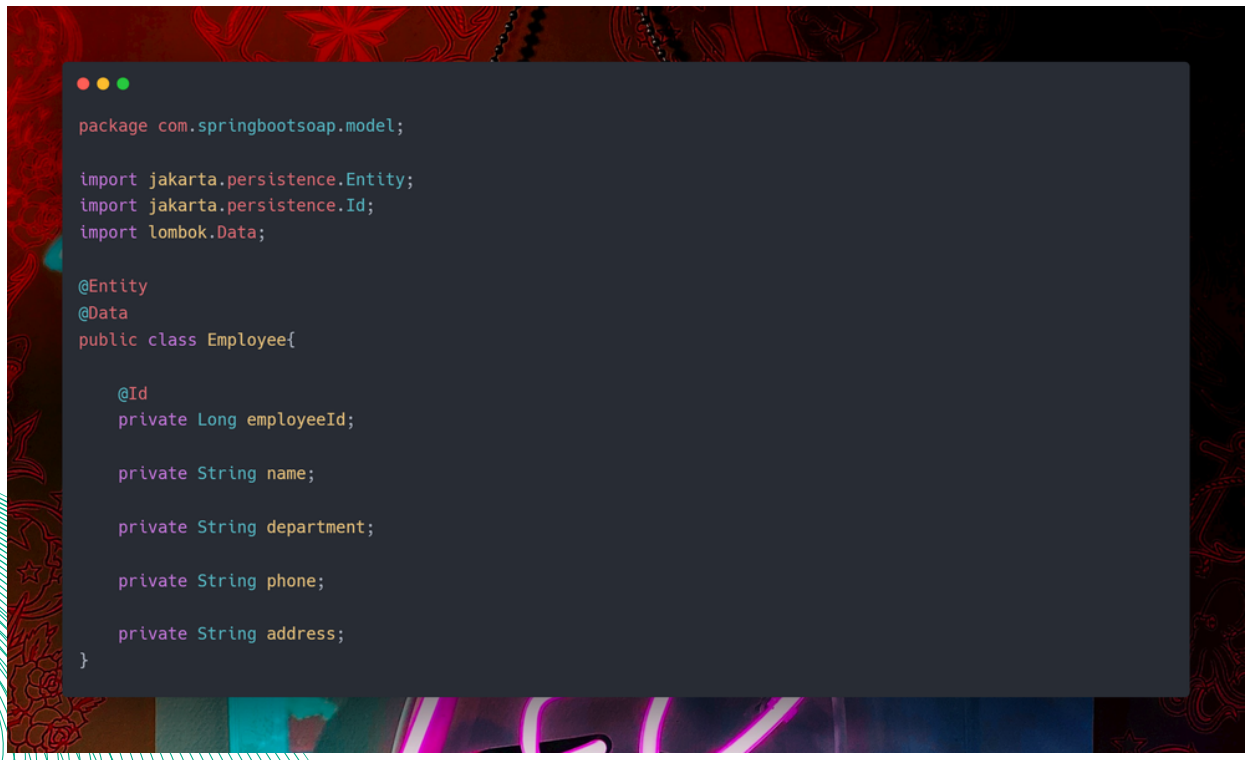
```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tns="http://com.springbootsoap.allapis"
           targetNamespace="http://com.springbootsoap.allapis"
           elementFormDefault="qualified">


    <xs:complexType name="employeeInfo">
        <xs:sequence>
            <xs:element name="employeeId" type="xs:long"/>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="department" type="xs:string"/>
            <xs:element name="phone" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="serviceStatus">
        <xs:sequence>
            <xs:element name="status" type="xs:string"/>
            <xs:element name="message" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="addEmployeeRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="employeeInfo"
type="tns:employeeInfo"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>


    <xs:element name="addEmployeeResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="serviceStatus"
type="tns:serviceStatus"/>
                <xs:element name="employeeInfo"
type="tns:employeeInfo"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="getEmployeeByIdRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="employeeId" type="xs:long"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>


    <xs:element name="getEmployeeByIdResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="employeeInfo"
type="tns:employeeInfo"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="updateEmployeeRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="employeeInfo"
type="tns:employeeInfo"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="updateEmployeeResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="serviceStatus"
type="tns:serviceStatus"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="deleteEmployeeRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="employeeId" type="xs:long"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="deleteEmployeeResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="serviceStatus"
type="tns:serviceStatus"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

APP

Main Method

```java
package com.springbootsoap;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SoapApplication {

    public static void main(String[] args) {
        SpringApplication.run(SoapApplication.class, args);
    }
}
```

MODEL



```java
package com.springbootsoap.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.Data;

@Entity
@Data
public class Employee{

    @Id
    private Long employeeId;

    private String name;

    private String department;

    private String phone;

    private String address;
}
```

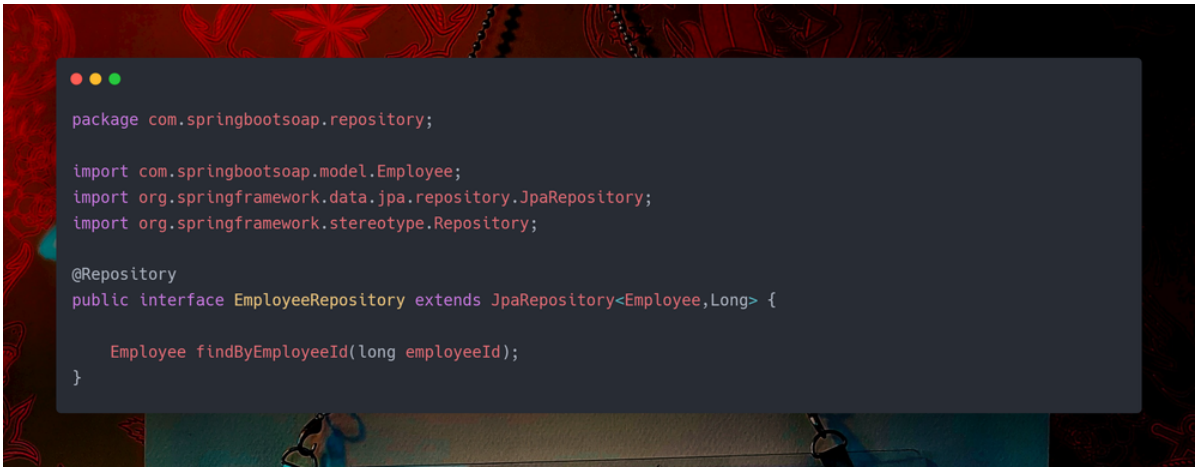
APP

ENDPOINT METHODS

```java
@Endpoint
public class EmployeeEndpoint {

    private static final String NAMESPACE_URI="http://com.springbootsoap.allapis";

    @Autowired
    private EmployeeService employeeService;

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "addEmployeeRequest")
    @ResponsePayload
    public AddEmployeeResponse addEmployee(@RequestPayload AddEmployeeRequest request){
        AddEmployeeResponse response=new AddEmployeeResponse();
        ServiceStatus serviceStatus=new ServiceStatus();

        Employee employee=new Employee();
        BeanUtils.copyProperties(request.getEmployeeInfo(),employee);
        employeeService.AddEmployee(employee);
        serviceStatus.setStatus("SUCCESS");
        serviceStatus.setMessage("Content Added Seccessfully");
        response.setServiceStatus(serviceStatus);
        response.setEmployeeInfo(request.getEmployeeInfo());
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getEmployeeByIdRequest")
    @ResponsePayload
    public GetEmployeeByIdResponse getEmployee(@RequestPayload GetEmployeeByIdRequest request){
        GetEmployeeByIdResponse response=new GetEmployeeByIdResponse();
        EmployeeInfo employeeInfo=new EmployeeInfo();
        BeanUtils.copyProperties(employeeService.getEmployById(request.getEmployeeId()),employeeInfo);
        response.setEmployeeInfo(employeeInfo);
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "updateEmployeeRequest")
    @ResponsePayload
    public UpdateEmployeeResponse updateEmployee(@RequestPayload UpdateEmployeeRequest request){
        Employee employee=new Employee();
        BeanUtils.copyProperties(request.getEmployeeInfo(),employee);
        employeeService.updateEmployee(employee);
        ServiceStatus serviceStatus=new ServiceStatus();
        UpdateEmployeeResponse response=new UpdateEmployeeResponse();
        serviceStatus.setStatus("SUCCESS");
        serviceStatus.setMessage("Content Updated Seccessfully");
        response.setServiceStatus(serviceStatus);
        return response;
    }


    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "deleteEmployeeRequest")
    @ResponsePayload
    public DeleteEmployeeResponse deleteEmployee(@RequestPayload DeleteEmployeeRequest request){
        employeeService.deleteEmployee(request.getEmployeeId());
        DeleteEmployeeResponse response=new DeleteEmployeeResponse();
        ServiceStatus serviceStatus=new ServiceStatus();
        serviceStatus.setStatus("SUCCESS");
        serviceStatus.setMessage("Content Deleted Seccessfully");
        response.setServiceStatus(serviceStatus);
        return response;
    }
}


// Photo by Jon Tyson on Unsplash
```

APP

REPOSITORY



```java
package com.springbootsoap.repository;

import com.springbootsoap.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee,Long> {

    Employee findByEmployeeId(long employeeId);
}
```

APP

SERVICES

```java
package com.springbootsoap.services;

import com.springbootsoap.model.Employee;

public interface EmployeeService {

    void AddEmployee(Employee employee);

    Employee getEmployById(long employeeId);

    void updateEmployee(Employee employee);

    void deleteEmployee(long employeeId);
}
```

```java
@Service
public class EmployeeServiceImpl implements EmployeeService{

    @Autowired
    EmployeeRepository employeeRepository;
    @Override
    public void AddEmployee(Employee employee) {
        employeeRepository.save(employee);
    }

    @Override
    public Employee getEmployById(long employeeId) {
        Employee employee=employeeRepository.findByEmployeeId(employeeId);
        return employee;
    }

    @Override
    public void updateEmployee(Employee employee) {
        employeeRepository.save(employee);
    }

    @Override
    public void deleteEmployee(long employeeId) {
        employeeRepository.deleteById(employeeId);
    }
}
```

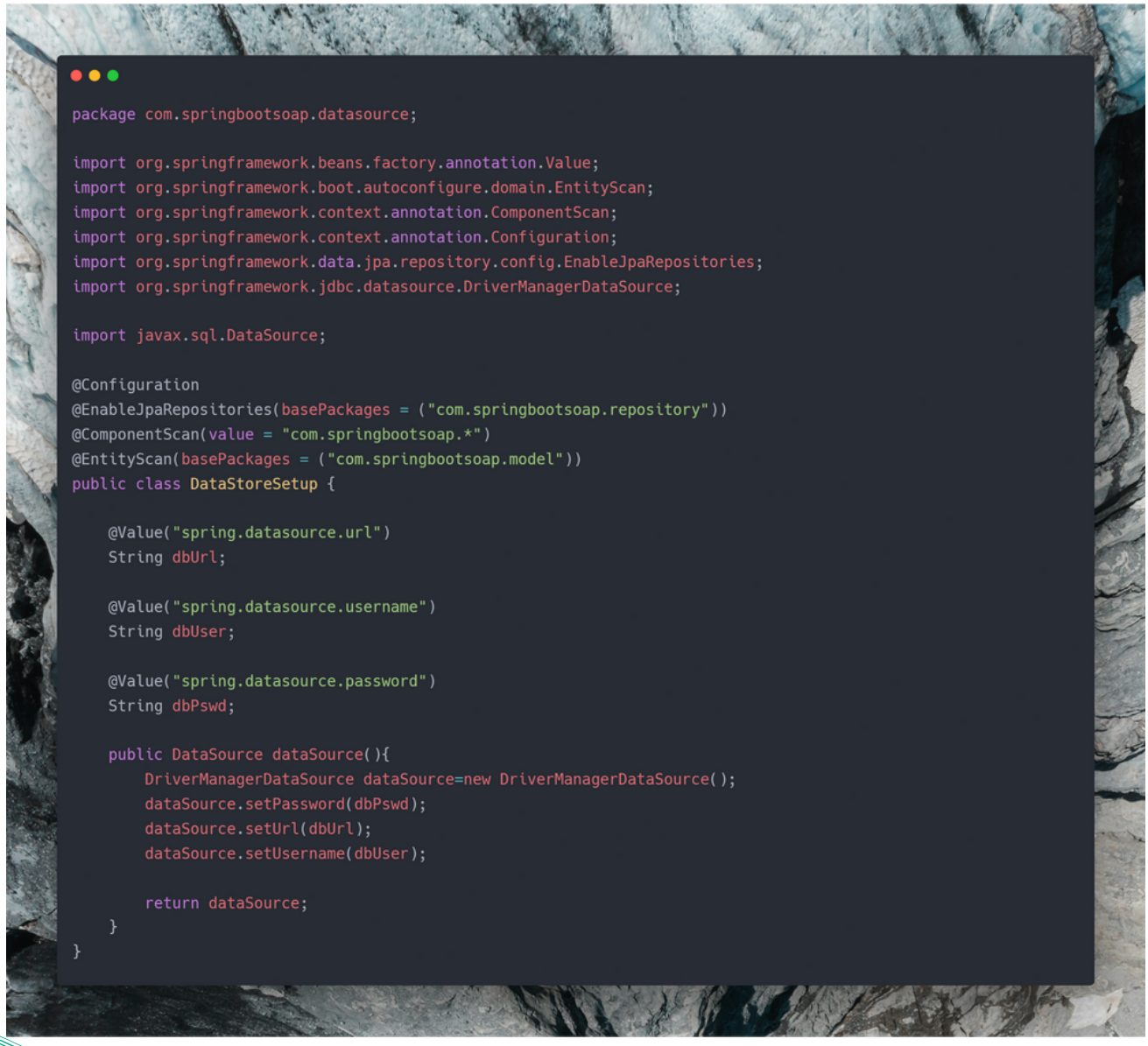
APP

CONFIG

```java
package com.springbootsoap.config;

import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.config.annotation.WsConfigurerAdapter;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;

@EnableWs
@Configuration
public class WebServiceConfiguration extends WsConfigurerAdapter
{
    @SuppressWarnings({"rawtypes","unchecked"})
    @Bean
    public ServletRegistrationBean messageDispatcherServlet(ApplicationContext applicationContext)
    {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(applicationContext);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean(servlet, "/allService/*");
    }

    @Bean(name = "employees")
    public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema employeeSchema)
    {
        DefaultWsdl11Definition wsdl11Definition = new DefaultWsdl11Definition();
        wsdl11Definition.setPortTypeName("allServiceSoapHttp");
        wsdl11Definition.setLocationUri("/allService");
        wsdl11Definition.setTargetNamespace("com.springbootsoap.allapis");
        wsdl11Definition.setSchema(employeeSchema);
        return wsdl11Definition;
    }

    @Bean
    public XsdSchema employeeSchema()
    {
        return new SimpleXsdSchema(new ClassPathResource("employee.xsd"));
    }
}

// Photo by Pierre Jeanneret on Unsplash
```

APP

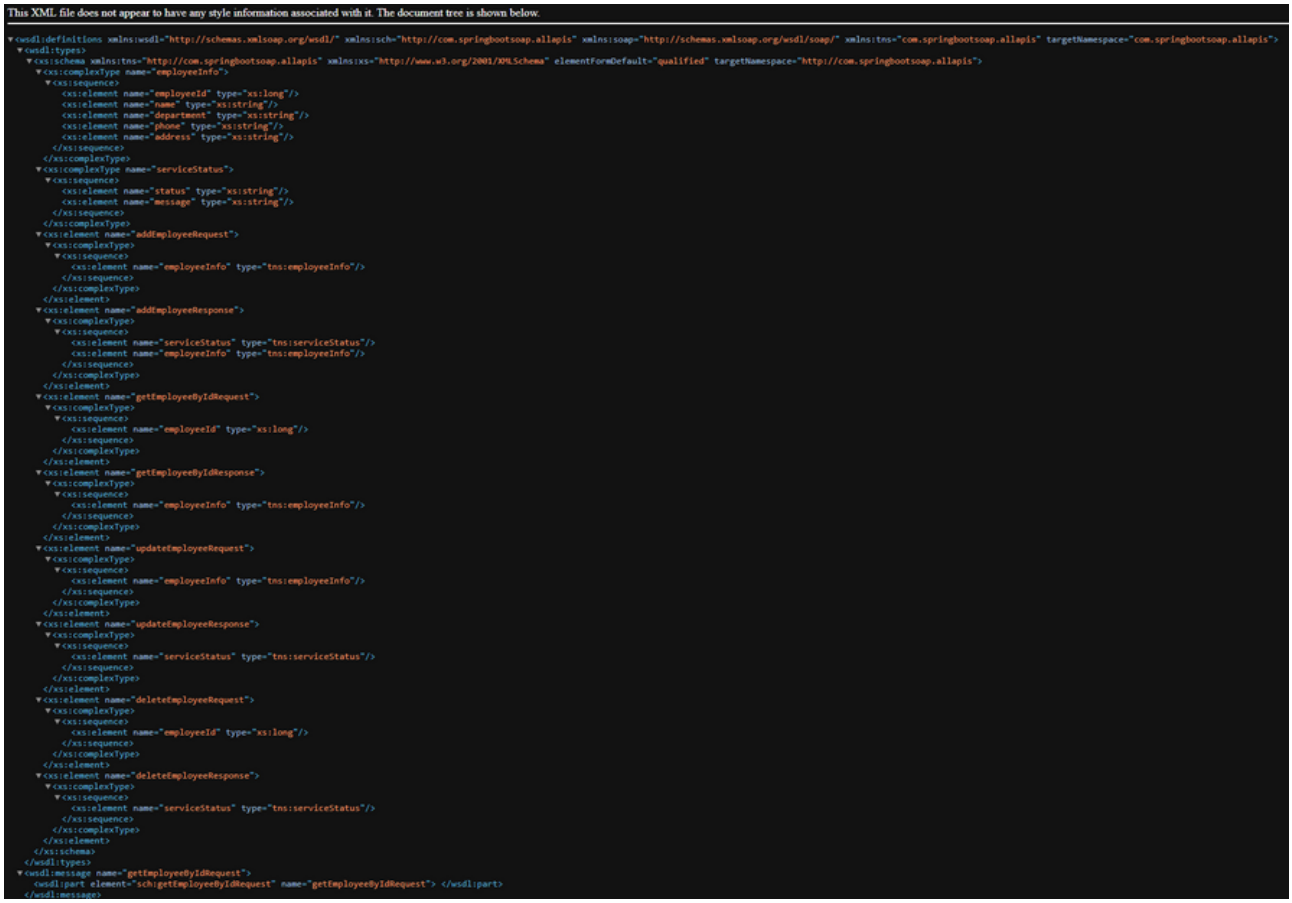
DATASTORE



```java
package com.springbootsoap.datasource;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

import javax.sql.DataSource;

@Configuration
@EnableJpaRepositories(basePackages = ("com.springbootsoap.repository"))
@ComponentScan(value = "com.springbootsoap.*")
@EntityScan(basePackages = ("com.springbootsoap.model"))
public class DataStoreSetup {

    @Value("spring.datasource.url")
    String dbUrl;

    @Value("spring.datasource.username")
    String dbUser;

    @Value("spring.datasource.password")
    String dbPswd;

    public DataSource dataSource(){
        DriverManagerDataSource dataSource=new DriverManagerDataSource();
        dataSource.setPassword(dbPswd);
        dataSource.setUrl(dbUrl);
        dataSource.setUsername(dbUser);

        return dataSource;
    }
}
```

Pruebas

```
http://localhost:8080/allService/employees.wsdl
```



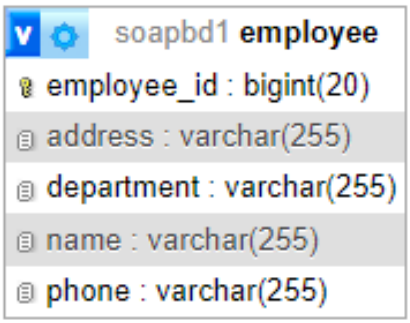
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://com.springbootsoap.allapis" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="com.springbootsoap.allapis" targetNamespace="com.springbootsoap.allapis">
  ▼<wsdl:types>
    ▼<xs:schema xmlns:tns="http://com.springbootsoap.allapis" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://com.springbootsoap.allapis">
      ▼<xs:complexType name="employeeInfo">
        ▼<xs:sequence>
            <xs:element name="employeeId" type="xs:long"/>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="department" type="xs:string"/>
            <xs:element name="phone" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      ▼<xs:complexType name="serviceStatus">
        ▼<xs:sequence>
            <xs:element name="status" type="xs:string"/>
            <xs:element name="message" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      ▼<xs:element name="addEmployeeRequest">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="employeeInfo" type="tns:employeeInfo"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="addEmployeeResponse">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="serviceStatus" type="tns:serviceStatus"/>
              <xs:element name="employeeInfo" type="tns:employeeInfo"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="getEmployeeByIdRequest">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="employeeId" type="xs:long"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="getEmployeeByIdResponse">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="employeeInfo" type="tns:employeeInfo"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="updateEmployeeRequest">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="employeeInfo" type="tns:employeeInfo"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="updateEmployeeResponse">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="serviceStatus" type="tns:serviceStatus"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="deleteEmployeeRequest">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="employeeId" type="xs:long"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      ▼<xs:element name="deleteEmployeeResponse">
        ▼<xs:complexType>
          ▼<xs:sequence>
              <xs:element name="serviceStatus" type="tns:serviceStatus"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:schema>
    </wsdl:types>
  ▼<wsdl:message name="getEmployeeByIdRequest">
      <wsdl:part element="sch:getEmployeeByIdRequest" name="getEmployeeByIdRequest"> </wsdl:part>
    </wsdl:message>
```

Probar la API

El primer requisito que debemos cumplir es que tengamos una bd con el nombre que se indica en application.properties y un usuario con privilegios que debe aparecer en dicho archivo. En nuestro caso sera la bd: soapbd1 el usuario ecom_user y sin contraseña. Si tu pruebas la app tienes 2 opciones, crear esta bd y este usuario en tu sgdb o sino cambiar application properties para que cuadre con usuarios que tienes en tu servidor.
Esta es la única tabla de nuestra bd por ahora, adjunto en este archivo zip / repositorio tienes un archivo .sql para poder reconstruirla con unos cuantos registros ya.

```
spring.datasource.name=soap        You, yesterday • soap API
spring.datasource.username=ecom_user
spring.datasource.password=
spring.datasource.url=jdbc:mysql://localhost:3306/soapbd1
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
```

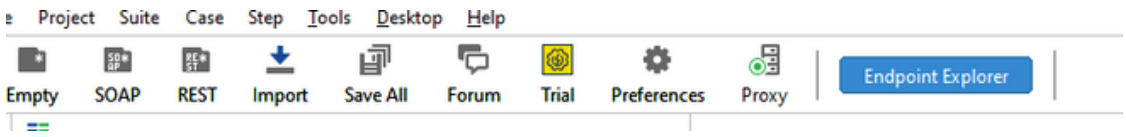


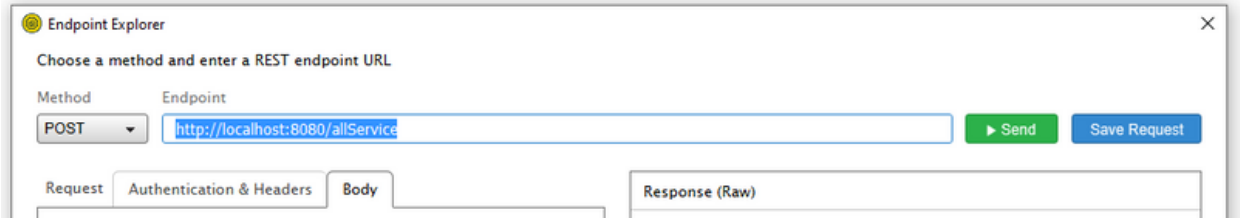
Probar la API

Para probar la API tenemos que usar un cliente SOAP como puede ser soapui. Desde este mandaremos XML con peticiones a los endpoints si todo funciona veremos como los registros se actualizan automaticamente.

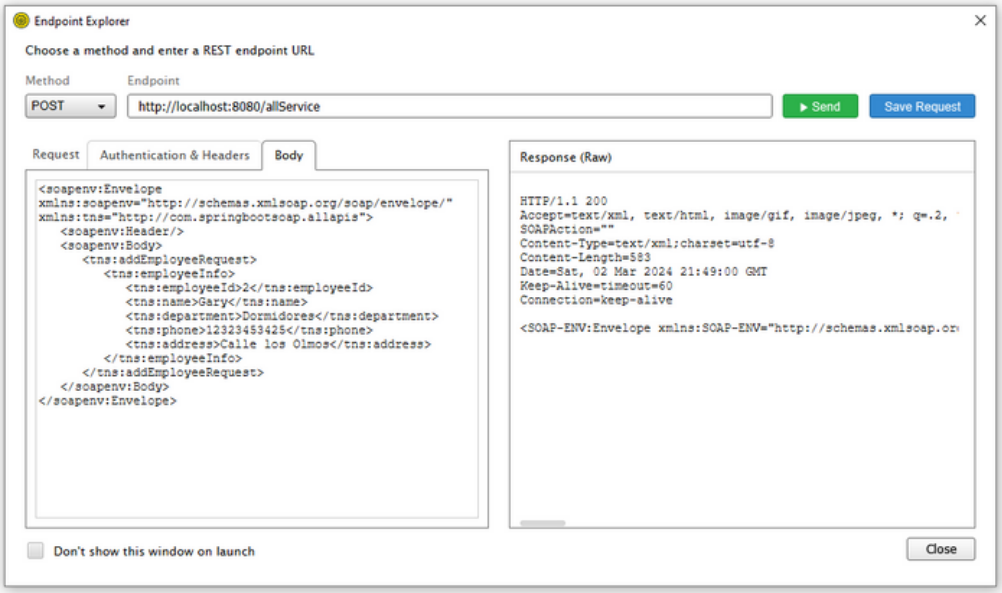
Le daremos a endpoint explorer, en el metodo pondremos post y en el body iremos poniendo los diferentes xml que dejamos a continuación. La ruta a la que mandamos las peticiones es donde está el web service corriendo en nuestro caso deberemos usar:

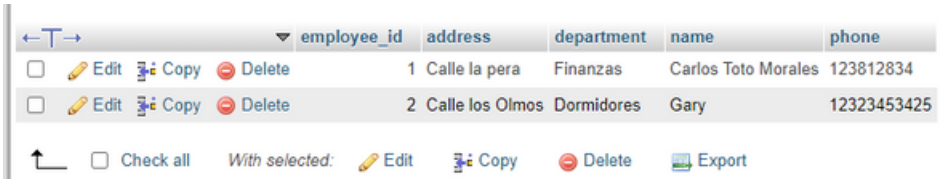
http://localhost:8080/allService





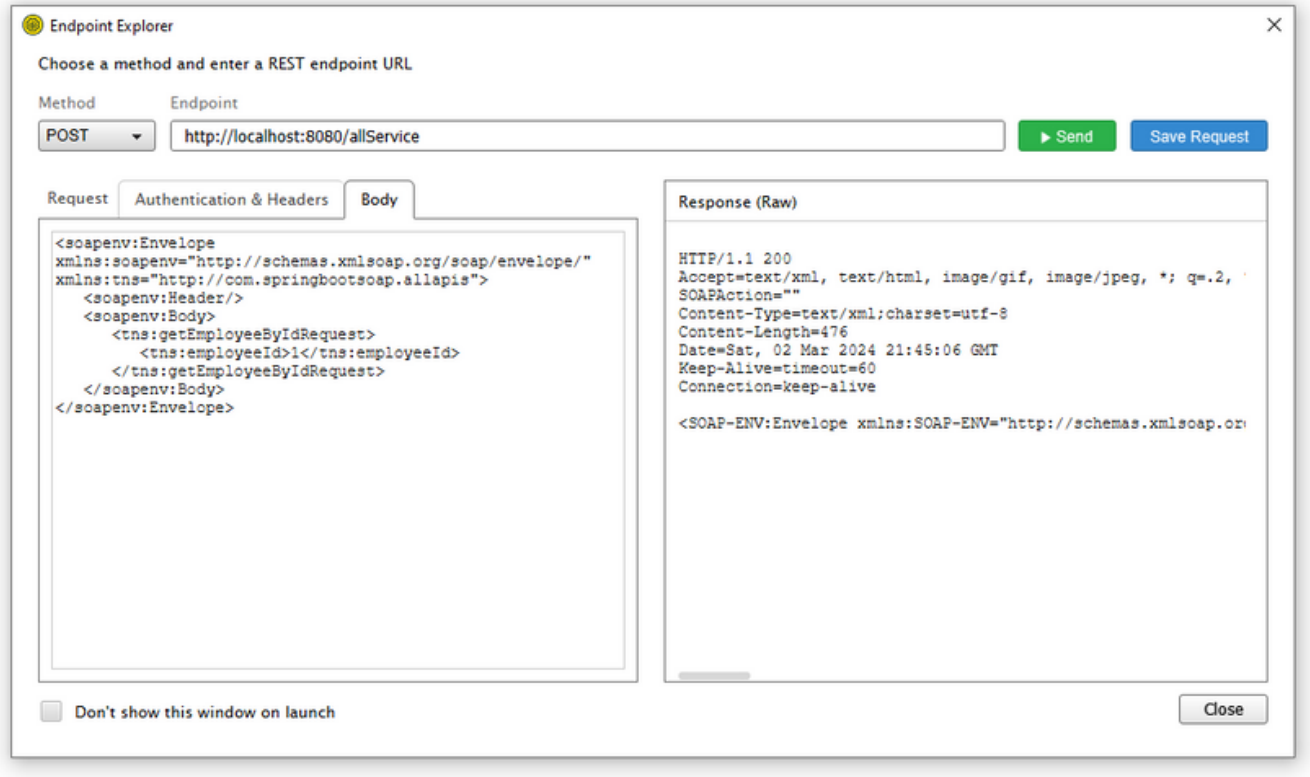
Crear un nuevo empleado





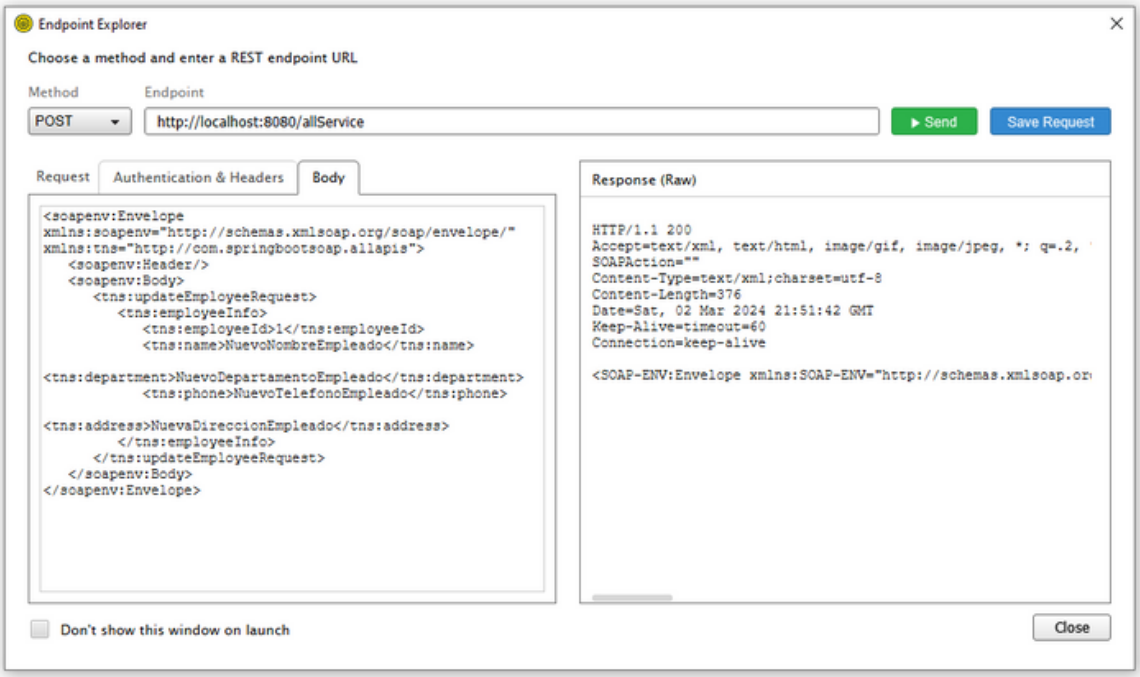

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://com.springbootsoap.allapis">
  <soapenv:Header/>
  <soapenv:Body>
    <tns:addEmployeeRequest>
      <tns:employeeInfo>
        <tns:employeeId>1</tns:employeeId>
        <tns:name>NombreEmpleado</tns:name>
        <tns:department>DepartamentoEmpleado</tns:department>
        <tns:phone>TelefonoEmpleado</tns:phone>
        <tns:address>DireccionEmpleado</tns:address>
      </tns:employeeInfo>
    </tns:addEmployeeRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

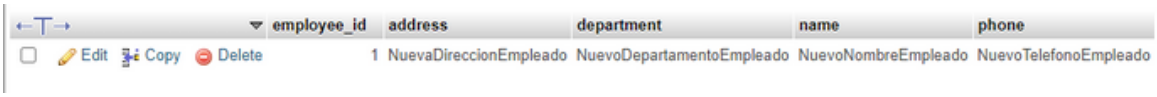
Recuperar datos de un empleado



<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://com.springbootsoap.allapis">
<soapenv:Header/> <soapenv:Body>
<tns:getEmployeeByIdRequest>
<tns:employeeId>1</tns:employeeId>
</tns:getEmployeeByIdRequest> </soapenv:Body>
</soapenv:Envelope>

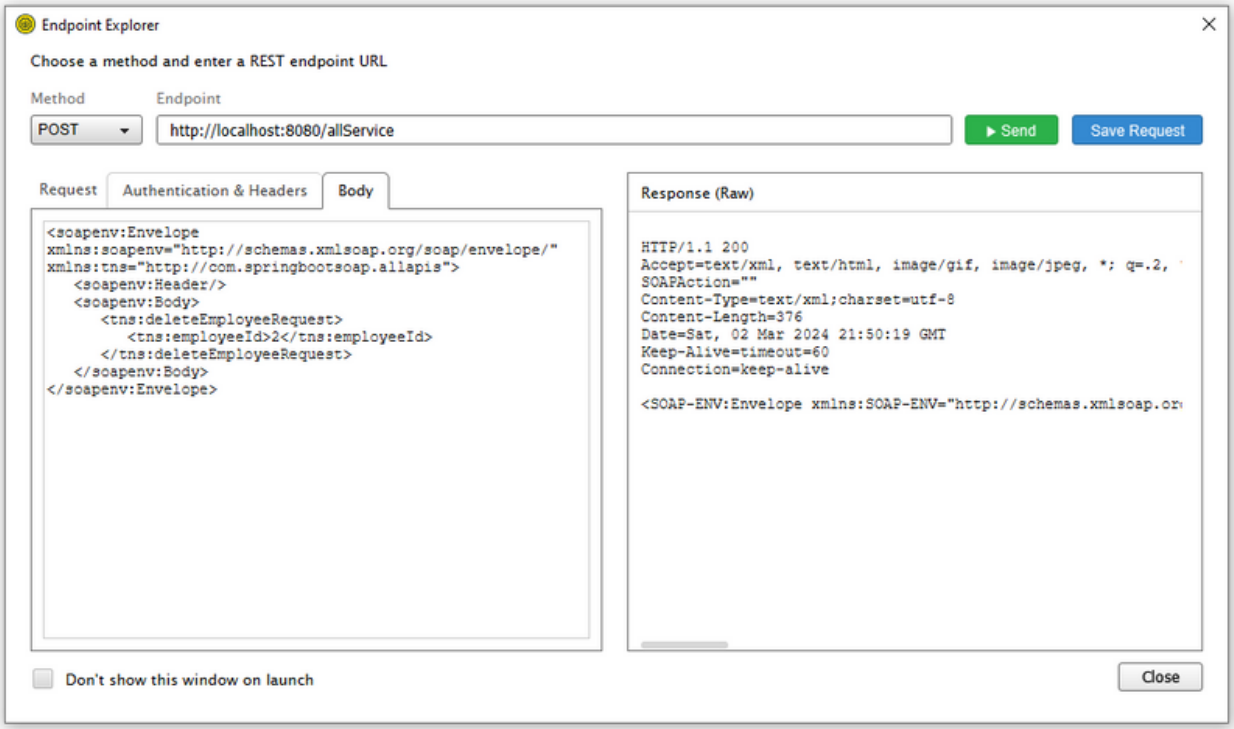
Modificar datos de un empleado






```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://com.springbootsoap.allapis">
  <soapenv:Header/>
  <soapenv:Body>
    <tns:updateEmployeeRequest>
      <tns:employeeInfo>
        <tns:employeeId>1</tns:employeeId>
        <tns:name>NuevoNombreEmpleado</tns:name>

<tns:department>NuevoDepartamentoEmpleado</tns:department>
        <tns:phone>NuevoTelefonoEmpleado</tns:phone>
        <tns:address>NuevaDireccionEmpleado</tns:address>
      </tns:employeeInfo>
    </tns:updateEmployeeRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Eliminar un empleado




```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://com.springbootsoap.allapis">
  <soapenv:Header/>
  <soapenv:Body>
    <tns:deleteEmployeeRequest>
      <tns:employeeId>1</tns:employeeId>
    </tns:deleteEmployeeRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

BIBLIOGRAPHY

- HTTPS://WWW.IBM.COM/DOCS/ES/RSAS/7.5.0?TOPIC=STANDARDS-SOAP
- HTTPS://SPRING.IO/GUIDES/GS/PRODUCING-WEB-SERVICE
- HTTPS://DOCS.SPRING.IO/SPRING-WS/DOCS/CURRENT/REFERENCE/HTML/