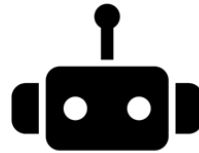




**PUCP**

## SESIÓN DE LABORATORIO 2

### Soap y Rest



*HORARIO 10M1*

Empezaremos a las 8:10 p.m

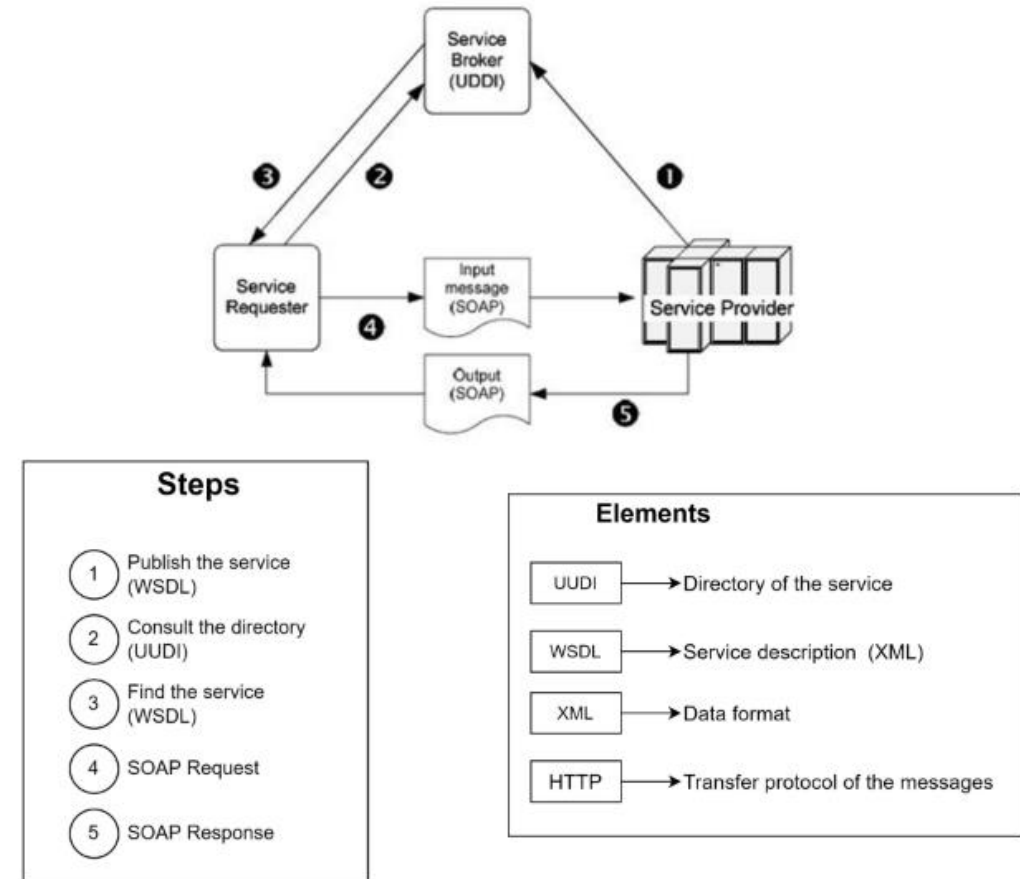
Gracias!



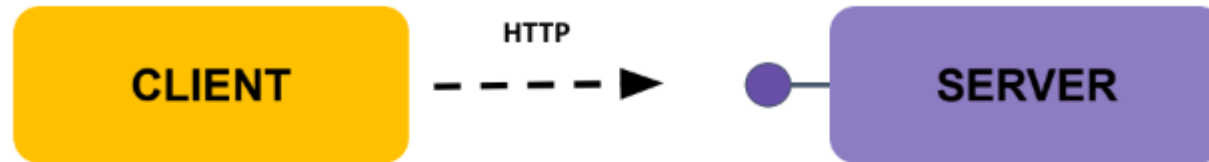
# Que es SOAP?

Es un protocolo estándar que incluye reglas para intercambio de mensajes usando HTTP.

Los mensajes SOAP necesitan ser formateados como documentos XML

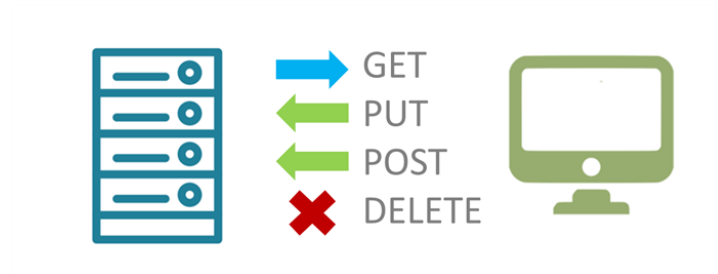


# Que es REST?



**Representational State Transfer**

# Operaciones CRUD



## CRUD Operations

# REST VS SOAP



REST

- Social Media
- Web Chat
- Mobile

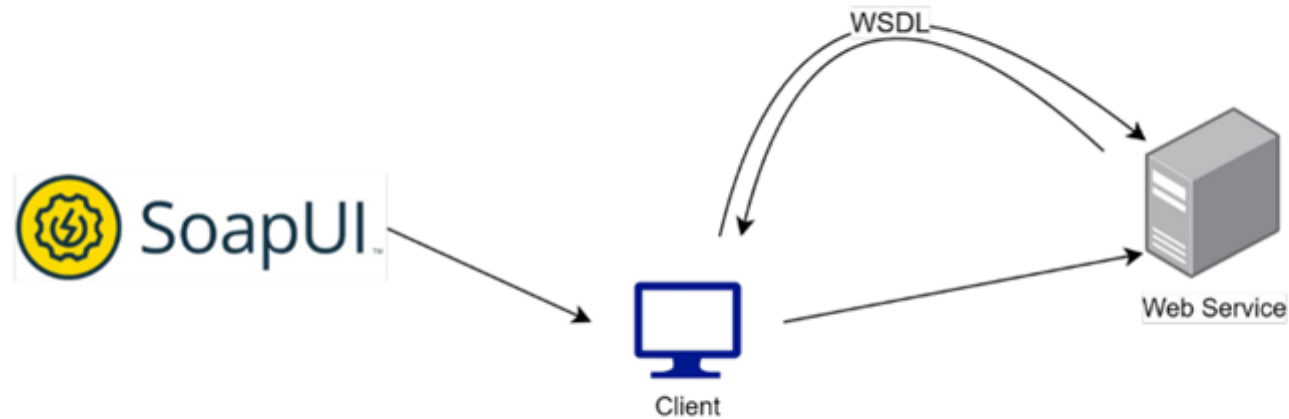


SOAP

- Financial
- Telecommunication
- Payment Gateways

# Desarrollo de nuestra aplicación SOAP

## Sistema de información de estudiantes



# Requisitos

- Java JDK 8
- Eclipse IDE
- Spring Tools 4
- SoapUI





# Instalaciones

## Spring Tool Suite Instalación (1/2)

El link de instalación se encuentra en el [github](#)

Una vez en la pagina deben hacer click en “4.24.0 Windows x86\_64” para descargarlo

Deberán poner este archivo en una ruta que recuerden

Finalmente , al ejecutar el instalador el cual les creara una carpeta del nombre : sts-4.24.0.RELEASE

## Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4. Free.  
Open source.

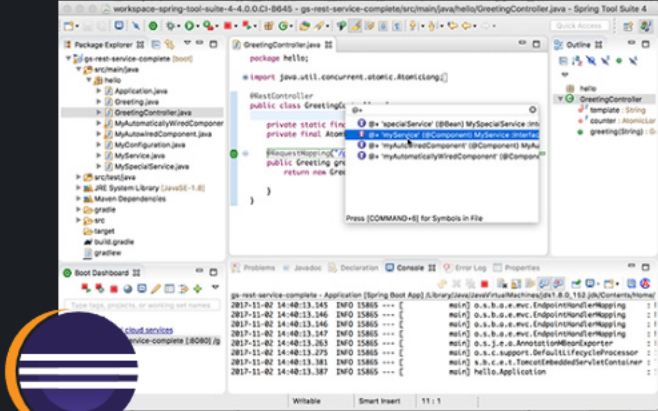
4.24.0 - LINUX X86\_64

4.24.0 - LINUX ARM\_64

4.24.0 - MACOS X86\_64

4.24.0 - MACOS ARM\_64

4.24.0 - WINDOWS X86\_64



# Instalaciones

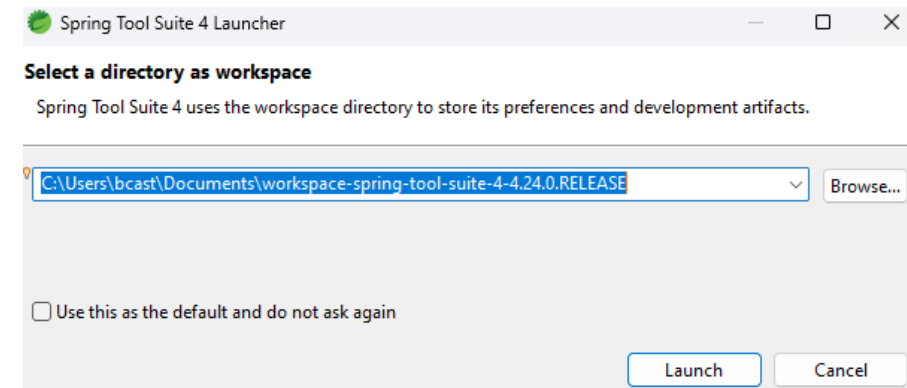
## Spring Tool Suite Instalación (2/2)

Esta carpeta contendrá el programa y para poder abrirlo

Deberán buscar el ejecutable SringToolSuite4

Nombre	Fecha de modificación	Tipo	Tamaño
configuration	22/08/2023 14:56	Carpeta de archivos	
dropins	22/08/2023 14:56	Carpeta de archivos	
features	22/08/2023 14:56	Carpeta de archivos	
p2	22/08/2023 14:56	Carpeta de archivos	
plugins	22/08/2023 14:56	Carpeta de archivos	
readme	22/08/2023 14:56	Carpeta de archivos	
.eclipseproduct	22/08/2023 14:56	Archivo ECLIPSEP...	1 KB
artifacts	22/08/2023 14:56	Documento XML	151 KB
license	22/08/2023 14:56	Archivo TXT	12 KB
open-source-licenses	22/08/2023 14:56	Archivo TXT	751 KB
SpringToolSuite4	22/08/2023 14:56	Aplicación	521 KB
SpringToolSuite4	22/08/2023 14:56	Opciones de confi...	1 KB
SpringToolSuite4c	22/08/2023 14:56	Aplicación	233 KB

Al ejecutar este archivo solo deberán seleccionar una carpeta de su preferencia para que sirva como workspace



# Instalaciones

## SOAP UI Instalación (1/2)

El archivo de instalación se encuentra en el GitHub y solamente tendrá que descargarlo desde el enlace



## SoapUI Open Source

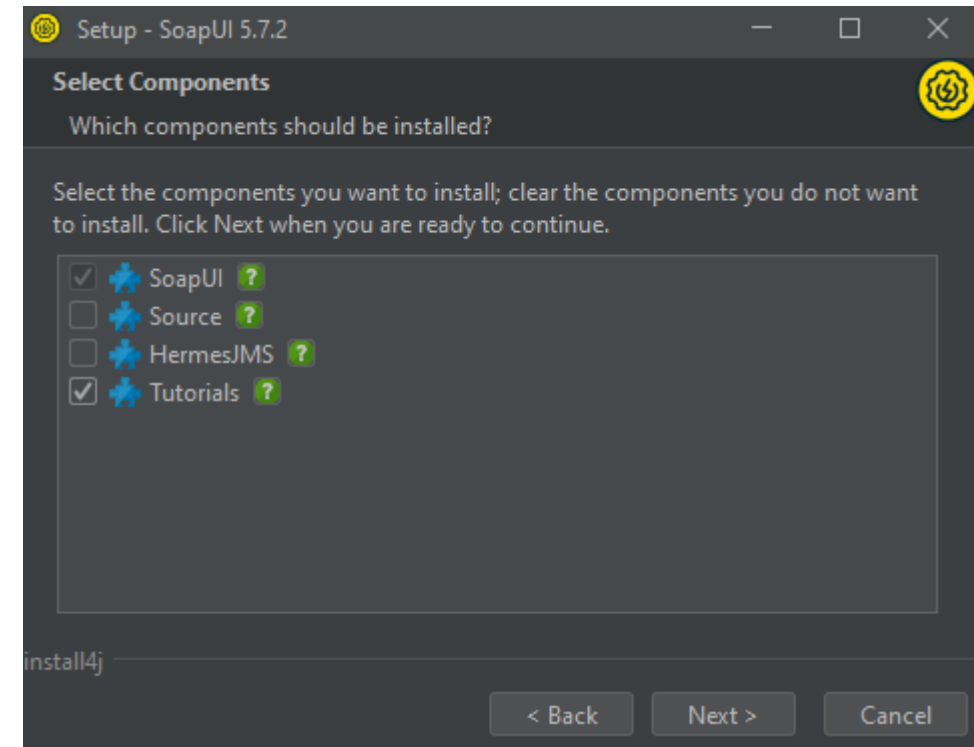
Get the open source version of the most widely used API testing tool in the world.

[Download SoapUI Open Source](#)

# Instalaciones

## SOAP UI Instalación (2/2)

Una vez ejecuten el archivo de instalación solo deberán seleccionar en next en cada pantalla y aceptar los valores por defecto



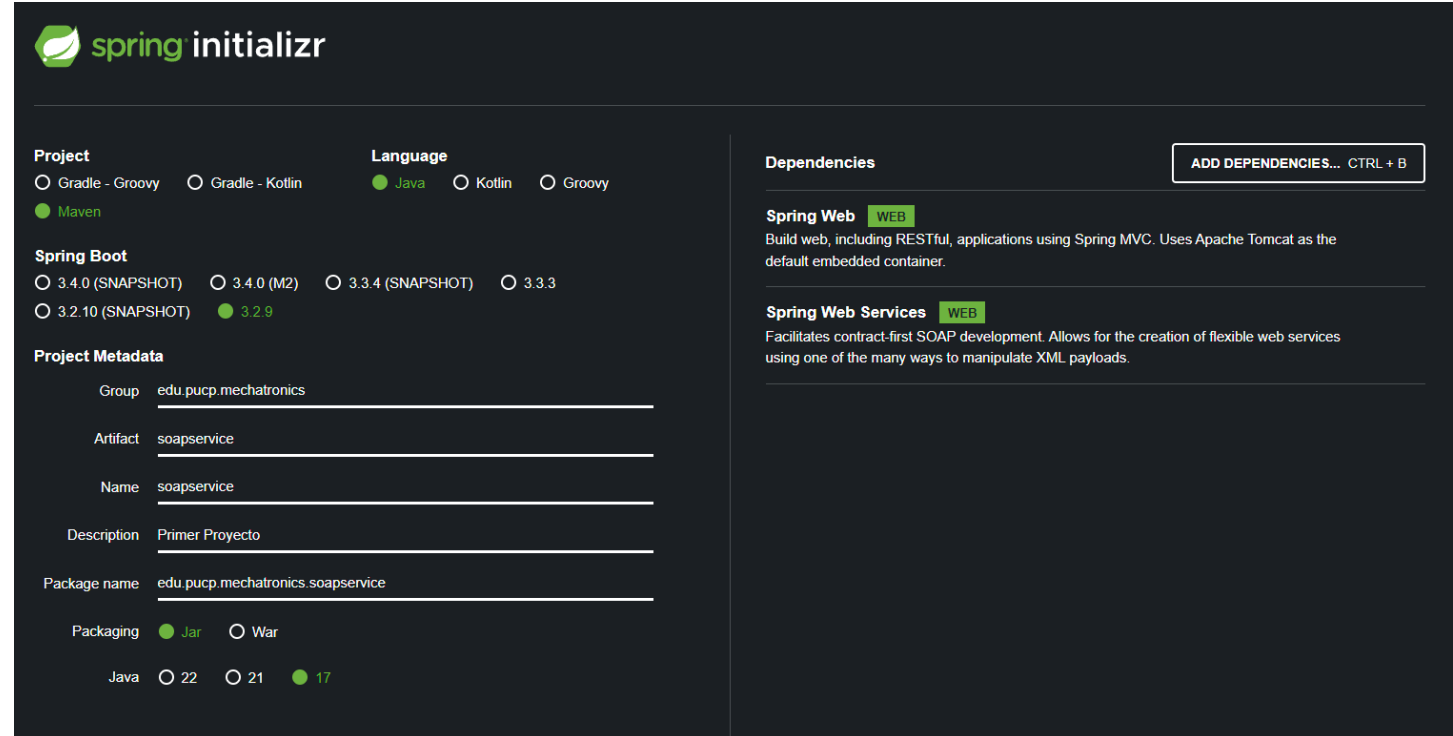
# Proyecto

## Configuración del servidor SOAP (1/6)

Para esto vamos a crear un nuevo proyecto utilizando Spring Initializr, por lo cual debemos ir a : <https://start.spring.io/>

Aquí llenaremos la data para el Project Metadata (soapservice) y añadiremos Spring Web y Spring Web Services como dependencias y le damos en Generate

Además el proyecto debe estar en Maven con lenguaje Java y el Spring boot por defecto en 3.2.9 y además llamaremos al Group como edu.pucp.mechatronics



The screenshot shows the Spring Initializr web interface. The 'Project' section has 'Maven' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '3.2.9' selected. The 'Project Metadata' section has the following fields filled: Group (edu.pucp.mechatronics), Artifact (soapservice), Name (soapservice), Description (Primer Proyecto), and Package name (edu.pucp.mechatronics.soapservice). The 'Packaging' section has 'Jar' selected. The 'Dependencies' section has 'Spring Web' and 'Spring Web Services' selected. A button 'ADD DEPENDENCIES... CTRL + B' is visible in the top right of the dependencies section.

**Project**

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Maven

**Language**

☒ Java ☐ Kotlin ☐ Groovy

**Spring Boot**

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M2) ☐ 3.3.4 (SNAPSHOT) ☐ 3.3.3 ☒ 3.2.10 (SNAPSHOT) ☒ 3.2.9

**Project Metadata**

Group

Artifact

Name

Description

Package name

**Packaging**

☒ Jar ☐ War

Java ☐ 22 ☐ 21 ☒ 17

**Dependencies** ADD DEPENDENCIES... CTRL + B

**Spring Web** WEB  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

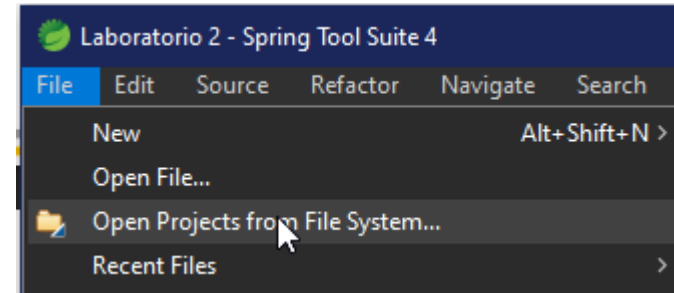
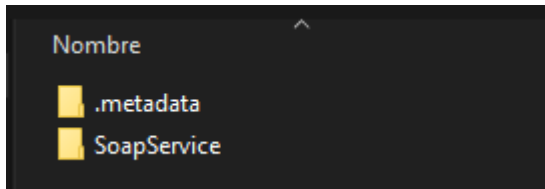
**Spring Web Services** WEB  
Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the many ways to manipulate XML payloads.

# Instalaciones

## Configuración del servidor SOAP (2/6)

Una vez que se haya generado, descargará el proyecto como un archivo Zip el cual extraeremos en nuestro Workspace

Para poder abrir este nuevo proyecto debemos abrir Spring Tool Suite, dar click en File y luego en Open Projects from File System

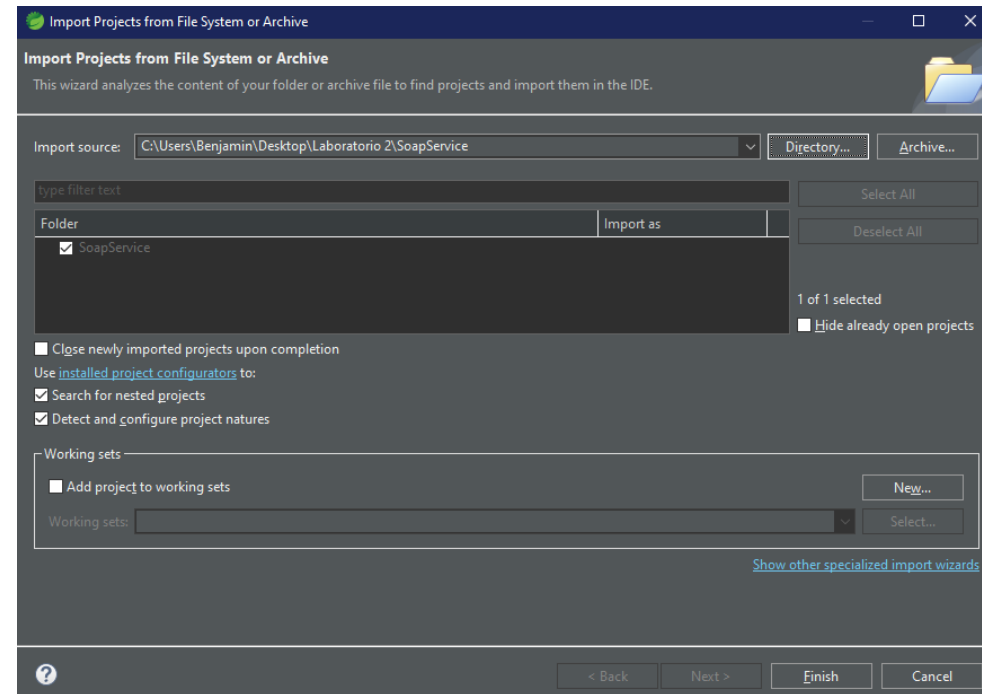


# Instalaciones

## Configuración del servidor SOAP (3/6)

Aquí seleccionaremos el folder que pusimos en nuestro Workspace usando el botón Directory y le damos click en Finish

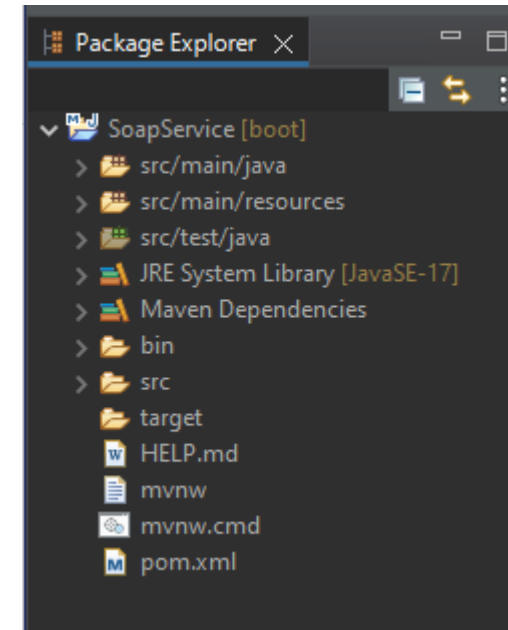
Aquí deberán esperar a que el proyecto termine de cargar.



# Instalaciones

## Configuración del servidor SOAP (4/6)

Una vez el proyecto haya terminado de cargar debería poderse ver de la siguiente forma:





# Instalaciones

## Configuración del servidor SOAP (5/6)

Una vez ya se tenga todo cargado, debemos modificar el archivo POM del proyecto.

Aquí debemos añadir EL Toolikt Web Services Description Language for Java (wsdl4j) y el jaxb-runtime

Además de esto debemos modificar la versión de Java a 1.8 y la versión de Spring Boot a 2.7.15 para que todo pueda funcionar correctamente

Luego para actualizar el Proyecto, grabamos con `ctrl + s`, luego, click derecho, maven, update project, OK

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.7.15</version>
<relativePath/> <!-- lookup parent from repository -->

<properties>
<java.version>1.8</java.version>
</properties>

<dependency>
<groupId>wsdl4j</groupId>
<artifactId>wsdl4j</artifactId>
</dependency>

<dependency>
<groupId>org.glassfish.jaxb</groupId>
<artifactId>jaxb-runtime</artifactId>
</dependency>
```

# Instalaciones

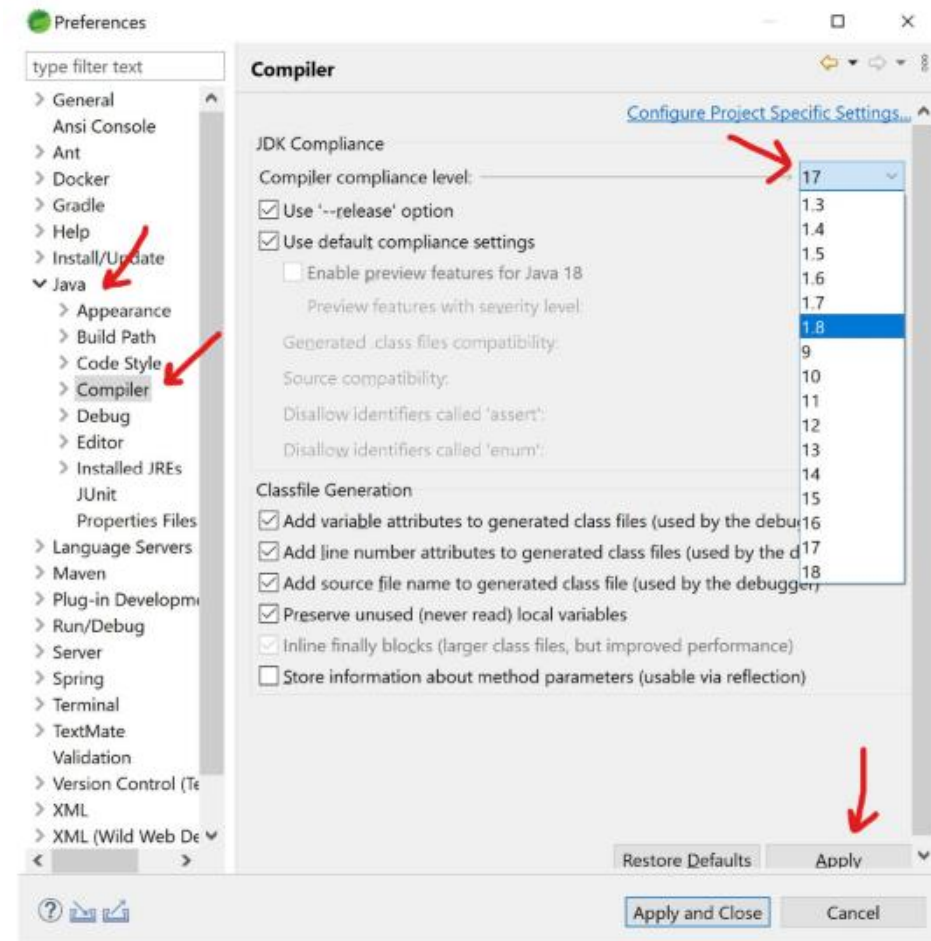
## Configuración del servidor SOAP (6/6)

Para evitar errores también vamos a modificar el build path para esto:

Dar clic en Windows , Preferencias.

En el panel izquierdo expandir Java y dar clic en Compilador.

Aquí seleccionar Compiler Compliance level de 1.8 y aplicar



# Implementación del Proyecto SOAP

Se crearán los siguientes paquetes y archivos :

School.xsd : Define las clases y métodos del servicio

edu.pucp.mechatronics.soapservices :

Este contiene clases para publicar el wsdl

Contiene la data para el repositorio del servicio

Contiene la clase principal para que pueda correr la aplicación

pe.edu.pucp.xml.school :

Contiene las clases autogeneradas para el archivo xsd

Contiene la clase student

Contiene las clases para enviar y recibir request del servicio

# Implementación del Proyecto SOAP

Implementacion del archivo  
school.xsd :  
Este va como recurso en el  
apartado src/main/resources

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.pucp.edu.pe/xml/school"
targetNamespace="http://www.pucp.edu.pe/xml/school"
elementFormDefault="qualified">
<xs:element name="StudentDetailsRequest">
<xs:complexType>
<xs:sequence>
<xs:element name="name" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="StudentDetailsResponse">
<xs:complexType>
<xs:sequence>
<xs:element name="Student" type="tns:Student"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="Student">
<xs:sequence>
<xs:element name="name" type="xs:string"/>
<xs:element name="age" type="xs:int"/>
<xs:element name="address" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

# Implementación del Proyecto SOAP

Añadiremos el plugin jaxb2 a nuestro archivo pom.xml para el generado automatico de clases desde schools.xsd

```
<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>jaxb2-maven-plugin</artifactId>
<version>2.5.0</version>
<executions>
<execution>
<id>xjc</id>
<goals>
<goal>xjc</goal>
</goals>
</execution>
</executions>
<configuration>
<sources>
<source>${project.basedir}/src/main/resources/school.xsd</source>
</sources>
</configuration>
</plugin>
```

# Implementación del Proyecto SOAP

Configuración del paquete edu.pucp.mechatronics.soapservice package  
Este contiene clases para publicar el wsdl y contiene la data para el servicio del repositorio

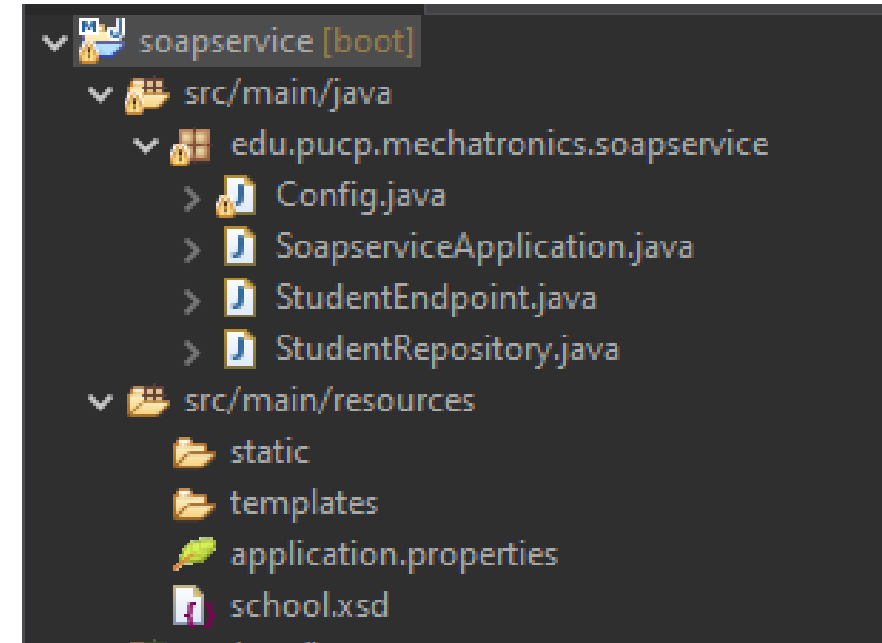
Este tendrá las clases :

Config

SoapServiceApplication

StudentRepository

StudentEndpoint



# Config

```
package edu.pucp.mechatronics.soapservice;
```

```
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 Config extends WsConfigurerAdapter{
```

```
@Bean
```

```
public ServletRegistrationBean messageDispatcherServlet(ApplicationContext
applicationContext)
```

```
{
```

```
MessageDispatcherServlet servlet = new MessageDispatcherServlet();
```

```
servlet.setApplicationContext(applicationContext);
```

```
servlet.setTransformWsdlLocations(true);
```

```
return new ServletRegistrationBean(servlet, "/service/*");
```

```
}
```

```
@Bean(name = "studentDetailsWsdl")
```

```
public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema
countriesSchema)
```

```
{
```

```
DefaultWsdl11Definition wsdl11Definition = new
```

```
DefaultWsdl11Definition();
```

```
wsdl11Definition.setPortTypeName("StudentDetailsPort");
```

```
wsdl11Definition.setLocationUri("/service/student-details");
```

```
wsdl11Definition.setTargetNamespace("http://www.pucp.edu.pe/xml/school
");
```

```
wsdl11Definition.setSchema(countriesSchema);
```

```
return wsdl11Definition;
```

```
}
```

```
@Bean
```

```
public XsdSchema countriesSchema()
```

```
{
```

```
return new SimpleXsdSchema(new ClassPathResource("school.xsd"));
```

```
}
```

```
}
```

# StudentRepository

```
package edu.pucp.mechatronics.soapservice;
```

```
import org.springframework.ws.server.endpoint.annotation.Endpoint;  
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;  
import org.springframework.ws.server.endpoint.annotation.RequestPayload;  
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;
```

```
import pe.edu.pucp.xml.school.StudentDetailsRequest;  
import pe.edu.pucp.xml.school.StudentDetailsResponse;
```

```
@Endpoint
```

```
public class StudentEndpoint {  
    private static final String NAMESPACE_URI = "http://www.pucp.edu.pe/xml/school";  
    private StudentRepository StudentRepository;  
    public StudentEndpoint(StudentRepository StudentRepository) {  
        this.StudentRepository = StudentRepository;  
    }  
    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "StudentDetailsRequest")  
    @ResponsePayload
```

```
    public StudentDetailsResponse getStudent(@RequestPayload StudentDetailsRequest request) {  
        StudentDetailsResponse response = new StudentDetailsResponse();  
        response.setStudent(StudentRepository.findStudent(request.getName()));  
        return response;  
    }  
}
```



# StudentEndpoint

```
package edu.pucp.mechatronics.soapservice;

import java.util.HashMap;
import java.util.Map;

import javax.annotation.PostConstruct;

import org.springframework.stereotype.Component;
import org.springframework.util.Assert;

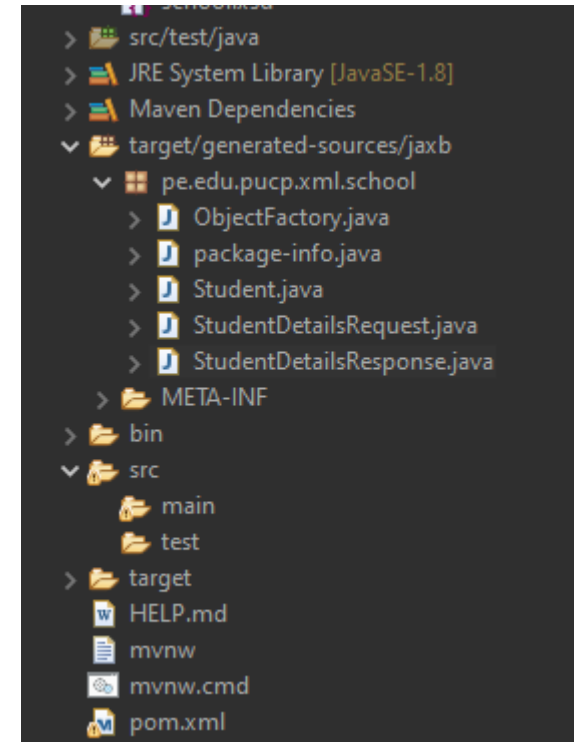
import pe.edu.pucp.xml.school.Student;

@Component
public class StudentRepository {
    private static final Map<String, Student> students = new HashMap<>();
    @PostConstruct
    public void initData() {
        Student student = new Student();
        student.setName("Hugo");
        student.setAge(27);
        student.setAddress("Lima");
        students.put(student.getName(), student);
        student = new Student();
        student.setName("Victor");
        student.setAge(28);
        student.setAddress("Nara");
        students.put(student.getName(), student);
    }
}
```

# Implementación del Proyecto SOAP

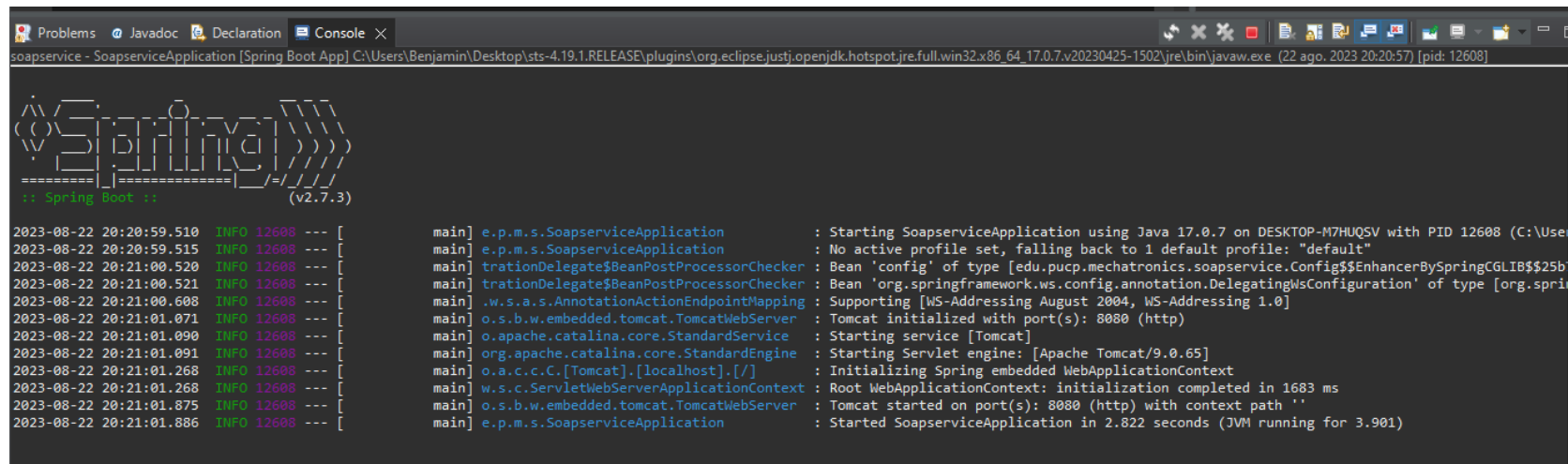
Configuración del paquete `pe.edu.pucp.xml.school`  
package

Este contiene clases para publicar el wsdl y  
contiene la data para el servicio del repositorio



# Corriendo el Servidor (1/2)

Para poder correr el servidor es necesario dar click derecho en el proyecto, run as , spring boot app



```
soapservice - SoapserviceApplication [Spring Boot App] C:\Users\Benjamin\Desktop\sts-4.19.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502\jre\bin\javaw.exe (22 ago, 2023 20:20:57) [pid: 12608]

:: Spring Boot :: (v2.7.3)

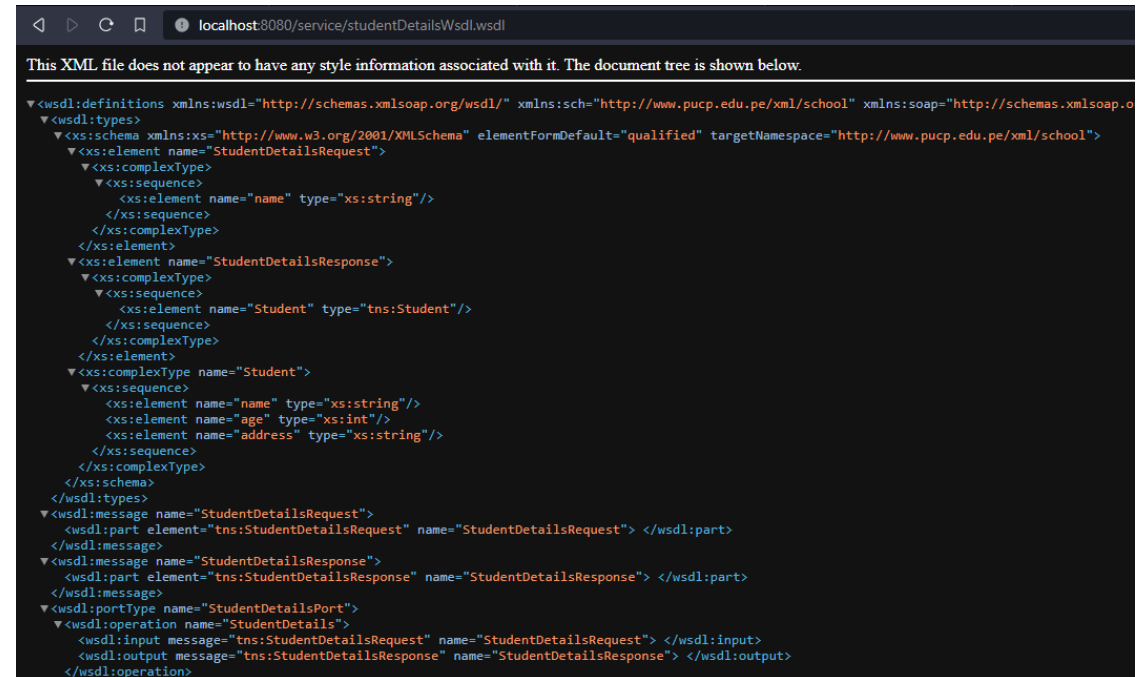
2023-08-22 20:20:59.510 INFO 12608 --- [main] e.p.m.s.SoapserviceApplication : Starting SoapserviceApplication using Java 17.0.7 on DESKTOP-M7HUQSV with PID 12608 (C:\Users\Benjamin\Desktop\sts-4.19.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502\jre\bin\javaw.exe)
2023-08-22 20:20:59.515 INFO 12608 --- [main] e.p.m.s.SoapserviceApplication : No active profile set, falling back to 1 default profile: "default"
2023-08-22 20:21:00.520 INFO 12608 --- [main] trationDelegate$BeanPostProcessorChecker : Bean 'config' of type [edu.pucp.mechatronics.soapservice.Config$$EnhancerBySpringCGLIB$$25b1a1e1] is not eligible for getting resolved by 'org.springframework.context.annotation.AnnotationMethodProcessing'
2023-08-22 20:21:00.521 INFO 12608 --- [main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.ws.config.annotation.DelegatingWsConfiguration' of type [org.springframework.ws.config.annotation.DelegatingWsConfiguration] is not eligible for getting resolved by 'org.springframework.context.annotation.AnnotationMethodProcessing'
2023-08-22 20:21:00.608 INFO 12608 --- [main] .w.s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2023-08-22 20:21:01.071 INFO 12608 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-08-22 20:21:01.090 INFO 12608 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-08-22 20:21:01.091 INFO 12608 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2023-08-22 20:21:01.268 INFO 12608 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-08-22 20:21:01.268 INFO 12608 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1683 ms
2023-08-22 20:21:01.875 INFO 12608 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-08-22 20:21:01.886 INFO 12608 --- [main] e.p.m.s.SoapserviceApplication : Started SoapserviceApplication in 2.822 seconds (JVM running for 3.901)
```

# Corriendo el Servidor (2/2)

Se puede ver que el servidor esta corriendo correctamente en este link :

<http://localhost:8080/service/studentDetailsWsdL.wsdl>

Aquí se podrá ver el contenido del archivo .wsdl

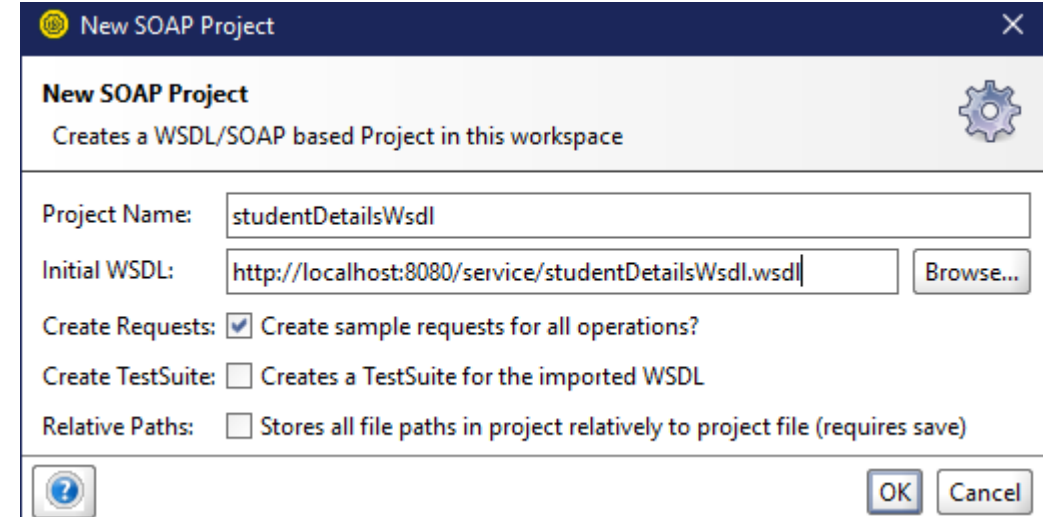


```
<?xml version='1.0' encoding='UTF-8'>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://www.pucp.edu.pe/xml/school" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <wsdl:types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://www.pucp.edu.pe/xml/school">
      <xs:element name="StudentDetailsRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="StudentDetailsResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Student" type="tns:Student"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:complexType name="Student">
        <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="age" type="xs:int"/>
          <xs:element name="address" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="StudentDetailsRequest">
    <wsdl:part element="tns:StudentDetailsRequest" name="StudentDetailsRequest"/>
  </wsdl:message>
  <wsdl:message name="StudentDetailsResponse">
    <wsdl:part element="tns:StudentDetailsResponse" name="StudentDetailsResponse"/>
  </wsdl:message>
  <wsdl:portType name="StudentDetailsPort">
    <wsdl:operation name="StudentDetails">
      <wsdl:input message="tns:StudentDetailsRequest" name="StudentDetailsRequest"/>
      <wsdl:output message="tns:StudentDetailsResponse" name="StudentDetailsResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

# Probando el proyecto (1/3)

Para hacer las pruebas usaremos SOAP UI , Para esto daremos click en File, new soap Project , luego en Initial WSDL , añadiremos el url.

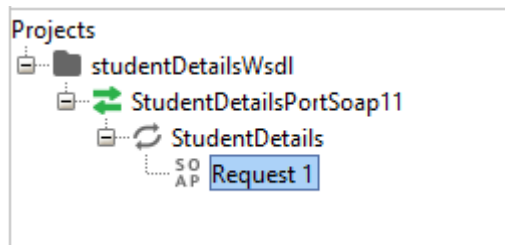
Aquí el nombre del proyecto se generará automáticamente.



# Probando el proyecto (2/3)

Esto nos generará nuestro nuevo proyecto, y una vez este hecho debemos hacer click en Request 1

Esto abrirá un nuevo request , que nos permitirá añadir un estudiante, para eso ingresaremos un nombre que este en StudentRepository entre <sch:name> y enviaremos el request

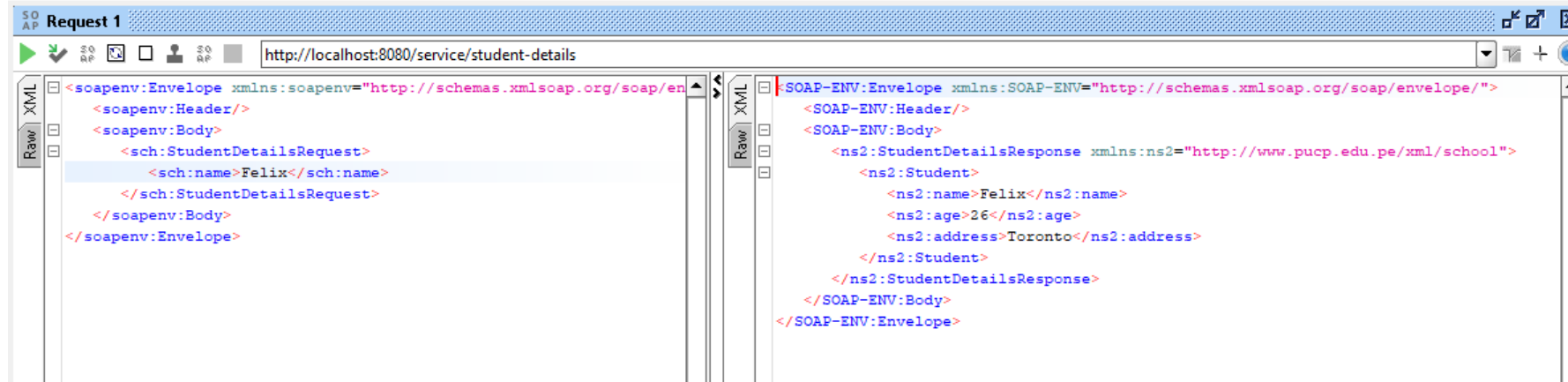


Vista de XML de un request en SOAPUI. El título es 'Request 1' y la URL es 'http://localhost:8080/service/student-details'. El XML es un mensaje SOAP con un cuerpo que contiene un request de detalles de un estudiante.

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sch="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <sch:StudentDetailsRequest>
      <sch:name>Benjamin</sch:name>
    </sch:StudentDetailsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

# Probando el proyecto (3/3)

Esto nos dará como resultado la información de uno de los estudiantes registrados dentro de StudentRepository.java



The screenshot shows a SOAP client interface with two panels. The left panel, titled 'Request 1', shows the XML request for the 'http://localhost:8080/service/student-details' endpoint. The right panel shows the XML response.

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <sch:StudentDetailsRequest>
      <sch:name>Felix</sch:name>
    </sch:StudentDetailsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:StudentDetailsResponse xmlns:ns2="http://www.pucp.edu.pe/xml/school">
      <ns2:Student>
        <ns2:name>Felix</ns2:name>
        <ns2:age>26</ns2:age>
        <ns2:address>Toronto</ns2:address>
      </ns2:Student>
    </ns2:StudentDetailsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# Desarrollo de nuestra aplicación REST

## Sistema de Clientes para un negocio





# Metodos a utilizar

## HTTP METHODS

GET

POST

PUT

Delete

Request

## Get Customers

**GET** /api/customers

Response

```
[  
  { id: 1, name: 'Hugo'},  
  { id: 2, name: 'Kim'},  
  ...  
]
```

# Metodos a utilizar

## HTTP METHODS

GET

POST

PUT

Delete

## Get a Customer

Request

GET /api/customers/1

Response

{ id: 1, name: 'Hugo' }

# Metodos a utilizar

## HTTP METHODS

GET

POST

PUT

Delete

## Create a Customer

Request

**POST** /api/customers

{ name: 'Kim' }

Response

{ id: 1, name: 'Kim' }

# Metodos a utilizar

## HTTP METHODS

GET

POST

PUT

Delete

## Update a Customer

Request

**PUT** /api/customers/1

{ name: 'Fernando' }

Response

{ id: 1, name: 'Fernando' }

# Metodos a utilizar

## HTTP METHODS

GET

POST

PUT

DELETE

## Delete a Customer

Request **DELETE** `/api/customers/1`

Response

`{ id: 1, name: 'Fernando' }`

# Requisitos

- Node JS
- Visual Studio Code
- npm
- Express JS  
y Postman



# Node JS

Para su instalación solo deben descargar la versión LTS más reciente desde el link provisto en GitHub

En caso de que quieran verificar la instalación pueden usar el comando CMD : `node --version`

## Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

[Download Node.js \(LTS\)](#) 

Downloads Node.js **v20.17.0**<sup>1</sup> with long-term support.  
Node.js can also be installed via [package managers](#).

Want new features sooner? Get **Node.js v22.8.0**<sup>1</sup> instead.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.3324]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Benjamin>node --version
v18.17.1
```

# Visual Studio Code

Para su instalación solo deben descargar la versión más reciente desde el link provisto en Github



↓ Windows

Windows 10, 11

User Installer	x64	x86	Arm64
System Installer	x64	x86	Arm64
.zip	x64	x86	Arm64
CLI	x64	x86	Arm64



# Configuración del proyecto

Para la configuración del proyecto utilizaremos un framework, en este caso Express JS , para su instalación en nuestro proyecto solo debemos escribir el comando npm install express

Una vez instalado haremos un nuevo archivo index.js con el siguiente código y luego ejecutaremos el comando node index.js para ejecutarlo

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\Benjamin\Desktop\Lab 2 Rest> npm install express

added 58 packages in 5s

8 packages are looking for funding
  run `npm fund` for details
PS C:\Users\Benjamin\Desktop\Lab 2 Rest> |
```

```
const express = require ('express')
const app = express()
```

```
app.use(express.json())
```

```
const customers = [
  {id :1 , name: "Hugo"},
  {id: 2 , name: "Kim"}
]
```

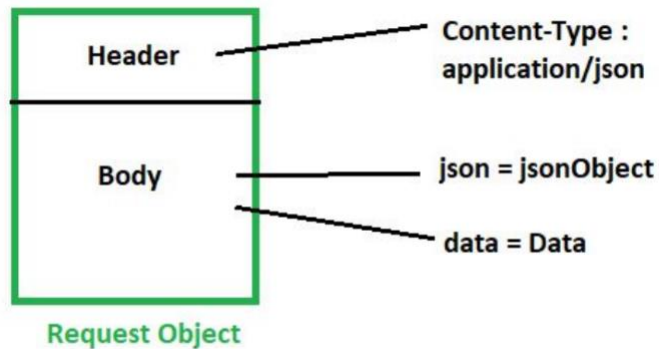
```
app.get('/',(req,res)=>{
  res.send('Hola mundo');
})
```

```
app.get('/api/customers', (req,res)=> {
  res.send(customers);
})
```

```
app.listen(3000,
  ()=> console.log('Escuchando en el puerto 3000....'));
```

# Configuración del proyecto

Continuando, también implementaremos el método post , el cual consta de un Header y un body para solicitar el objeto a trabajar



```
app.get('/api/customers/:id', (req,res)=> {  
  const customer = customers.find(c => c.id ===parseInt(req.params.id));  
  if(!customer){  
    res.status(404).send('El usuario con este ID no se encuentra')  
  }  
  res.send(customer)  
})
```

```
app.post ('/api/customers', (req,res)=> {  
  const customer = {  
    id: customers.length + 1,  
    name : req.body.name,  
  }  
  customers.push(customer);  
  res.send(customer);  
}  
)
```

# Probando el proyecto

Con el fin de poder probar lo que hemos implementado hasta ahora, haremos uso de Postman

Para su instalación solo deben descargar la versión mas reciente desde el link provisionado en GitHub

## The Postman app

Download the app to get started with the Postman API Platform.

 Windows 64-bit

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#) · [Product Roadmap](#)

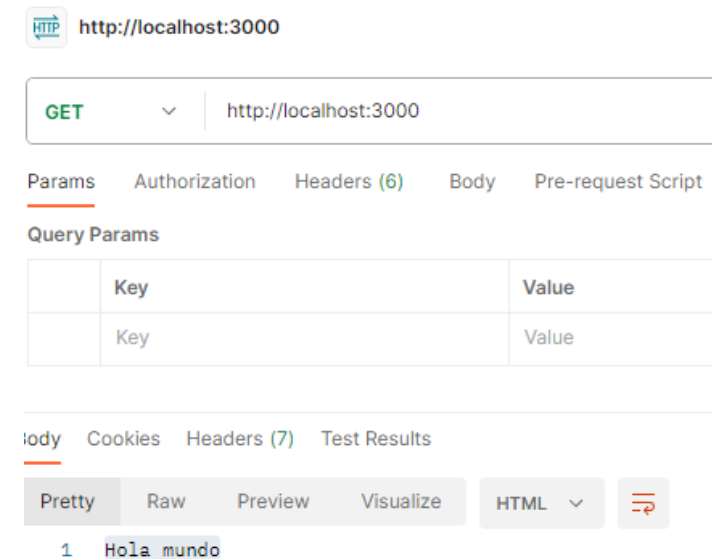
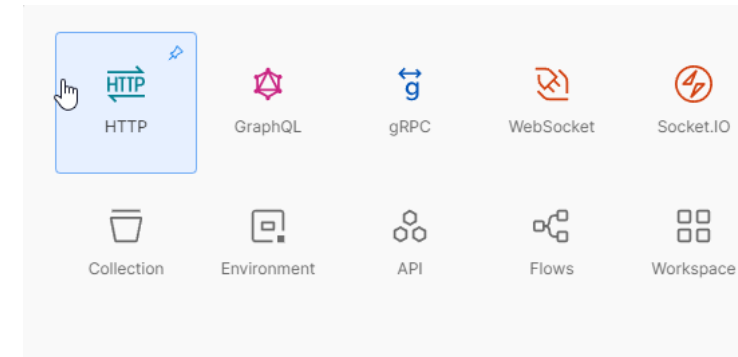
Not your OS? Download for Mac ([Intel Chip](#), [Apple Chip](#)) or Linux ([x64](#), [arm64](#))



# Probando el proyecto

Una vez lo tengamos abierto, nos dirigiremos al apartado My Workspace y daremos click en New y seleccionaremos donde dice HTTP

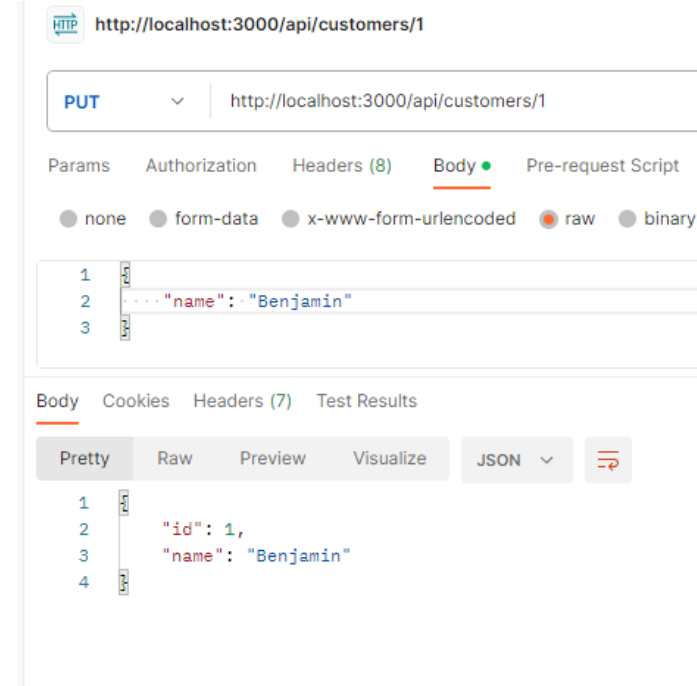
Una vez tengamos esto , haremos la prueba con los apis creados, para esto debemos inicializar el proyecto con node index.js



# Probando el proyecto

Continuando, tambien implementaremos el método put

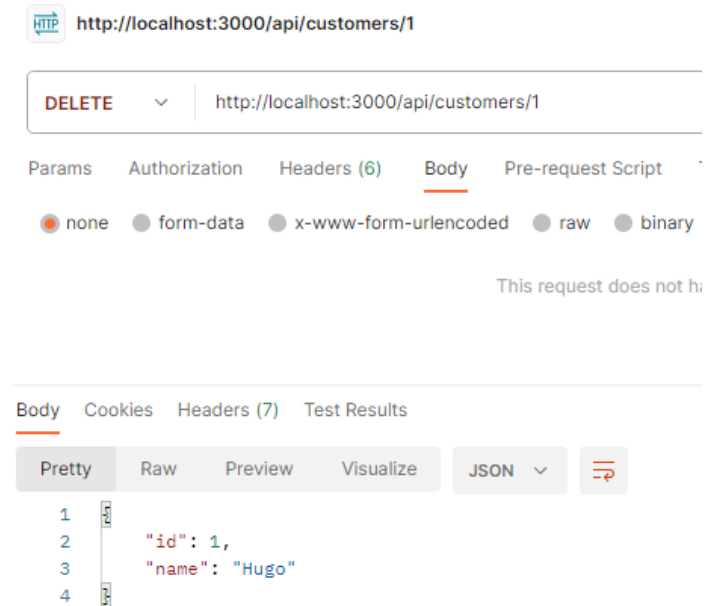
```
app.put('/api/customers/:id', (req,res)=> {  
  const customer = customers.find(c => c.id ===parseInt(req.params.id));  
  if(!customer){  
    res.status(404).send('El usuario con este ID no se encuentra')  
  }  
  if(!req.body.name || req.body.name.length<3){  
    res.status(403).send("El nombre debe tener al menos 3 caracteres")  
    return  
  }  
  
  customer.name =req.body.name;  
  res.send(customer);  
})
```



# Probando el proyecto

Finalmente implementaremos el método delete

```
app.delete('/api/customers/:id', (req,res)=> {  
  const customer = customers.find(c => c.id ===parseInt(req.params.id));  
  if(!customer){  
    res.status(404).send('El usuario con este ID no se encuentra')  
  }  
  const index = customers.indexOf(customer);  
  customers.splice(index,1);  
  res.send(customer);  
})
```



# Información importante

Swagger – Open Api <https://petstore.swagger.io/>

<b>pet</b> Everything about your Pets		Find out more: <a href="http://swagger.io">http://swagger.io</a>	^
GET	/pet/{petId} Find pet by ID	✓	🔒
POST	/pet/{petId} Updates a pet in the store with form data	✓	🔒
DELETE	/pet/{petId} Deletes a pet	✓	🔒
POST	/pet/{petId}/uploadImage uploads an image	✓	🔒
POST	/pet Add a new pet to the store	✓	🔒
PUT	/pet Update an existing pet	✓	🔒
GET	/pet/findByStatus Finds Pets by status	✓	🔒