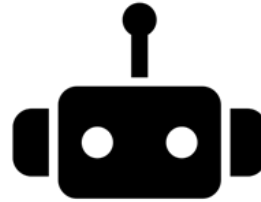


SESIÓN DE LABORATORIO

Soap y Rest



HORARIO 10M1

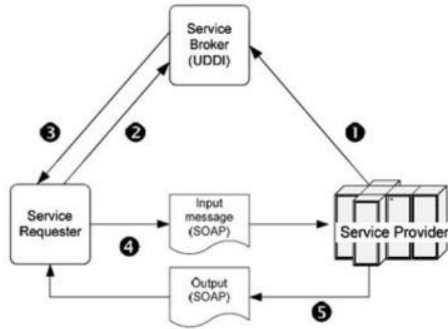
Bienvenidos !

Empezaremos a las 6:10 p.m

Gracias! c:



Que es SOAP?



Steps

- 1 Publish the service (WSDL)
- 2 Consult the directory (UDDI)
- 3 Find the service (WSDL)
- 4 SOAP Request
- 5 SOAP Response

Elements

UDDI	→ Directory of the service
WSDL	→ Service description (XML)
XML	→ Data format
HTTP	→ Transfer protocol of the messages

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



Create

Read

Update

Delete



CRUD Operations

REST VS SOAP



REST

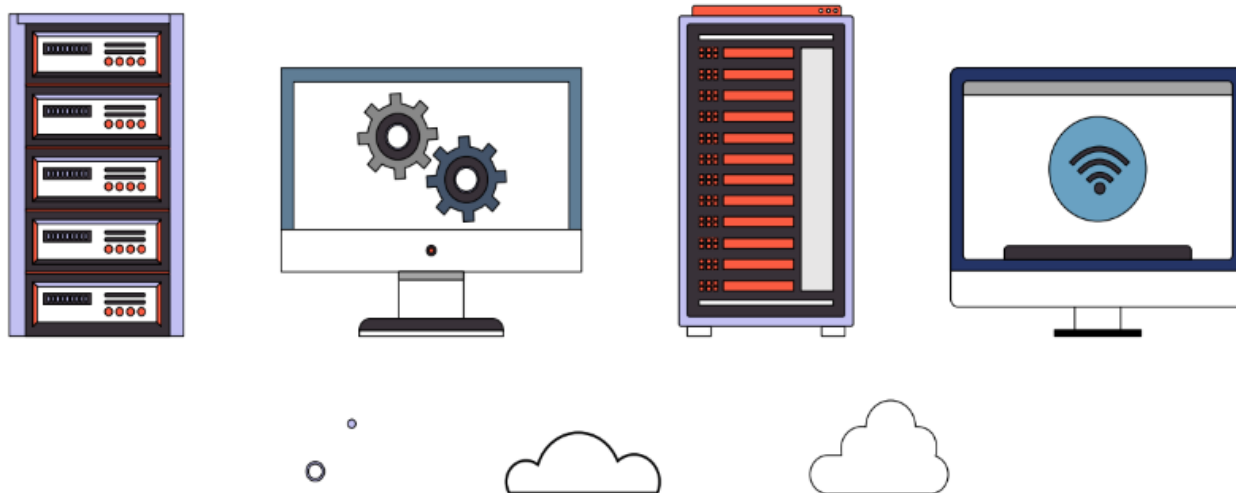
- Social Media
- Web Chat
- Mobile



SOAP

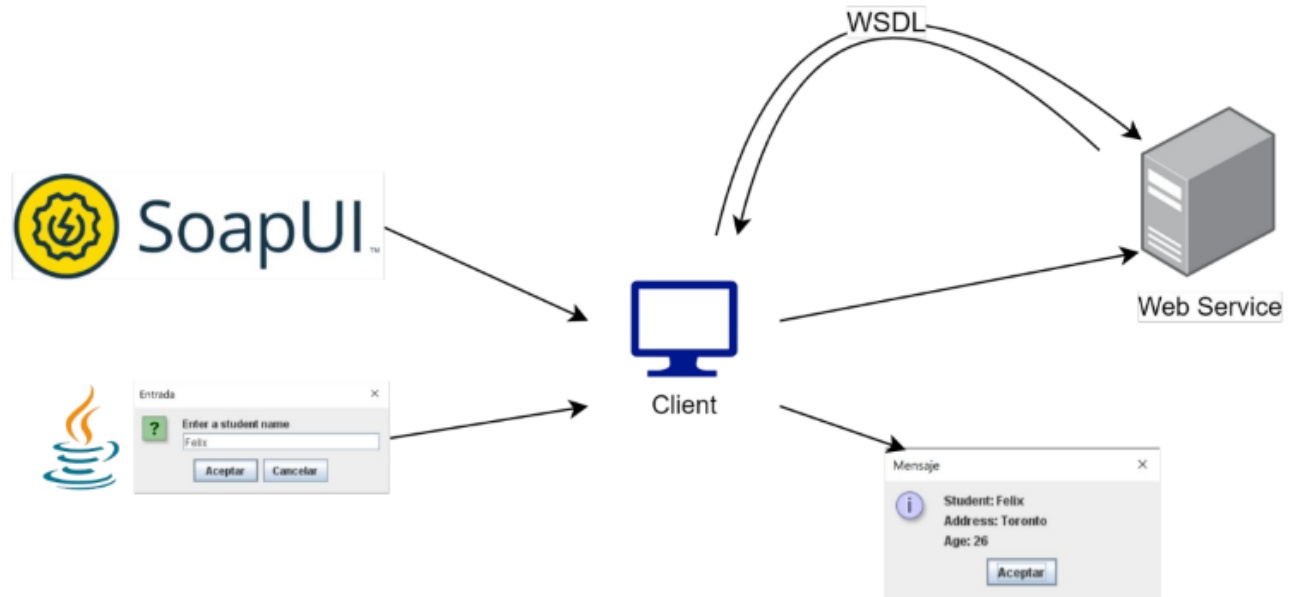
- Financial
- Telecommunication
- Payment Gateways

Desarrollo de nuestra aplicación



Desarrollo de nuestra aplicación SOAP

Sistema de informacion de estudiantes



Desarrollo de nuestra aplicación

Requisitos

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



Instalaciones

Spring Tool Suite Instalación (1/2)

- Deben ir al siguiente url:

<https://spring.io/tools>

- Deben hacer click en “Windows x86_64”
-> Download

Deberan poner este archive en una ruta que recuerden

Finalmente ejecutar el instalador el cual les creara una carpeta.

Spring Tools 4 for Eclipse

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

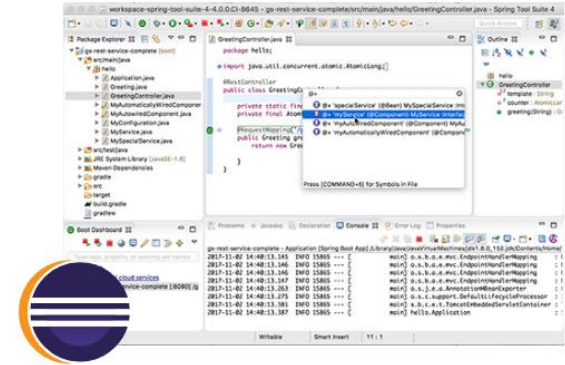
4.19.1 - LINUX X86_64

4.19.1 - LINUX ARM_64

4.19.1 - MACOS X86_64

4.19.1 - MACOS ARM_64

4.19.1 - WINDOWS X86_64



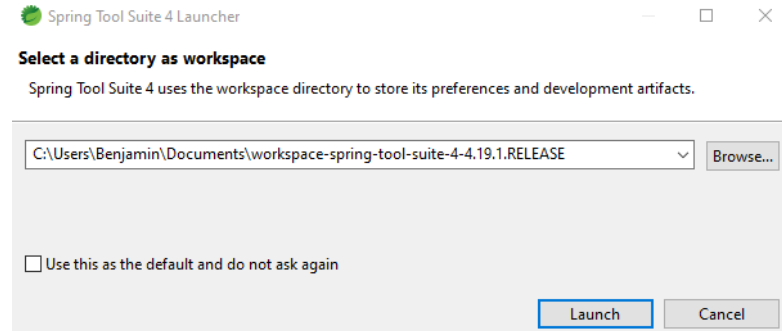
Instalaciones

Spring Tool Suite Instalación (1/2)

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)

Descargar desde

<https://www.soapui.org/downloads/soapui/>



SoapUI Open Source

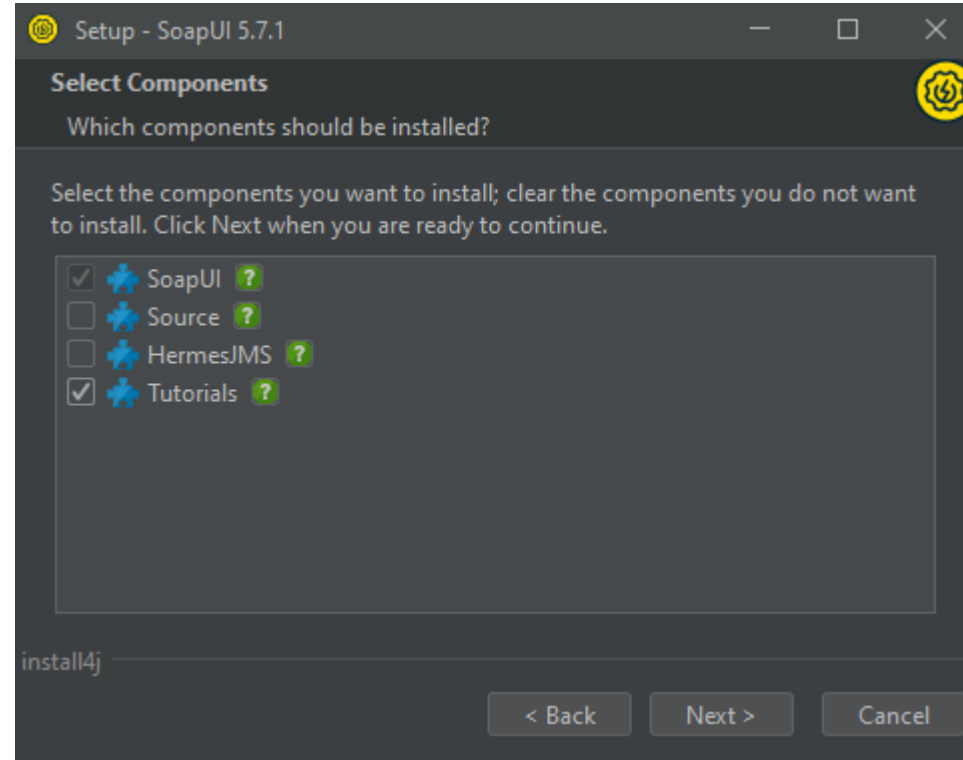
Get the open source version of the most widely
used API testing tool in the world.

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

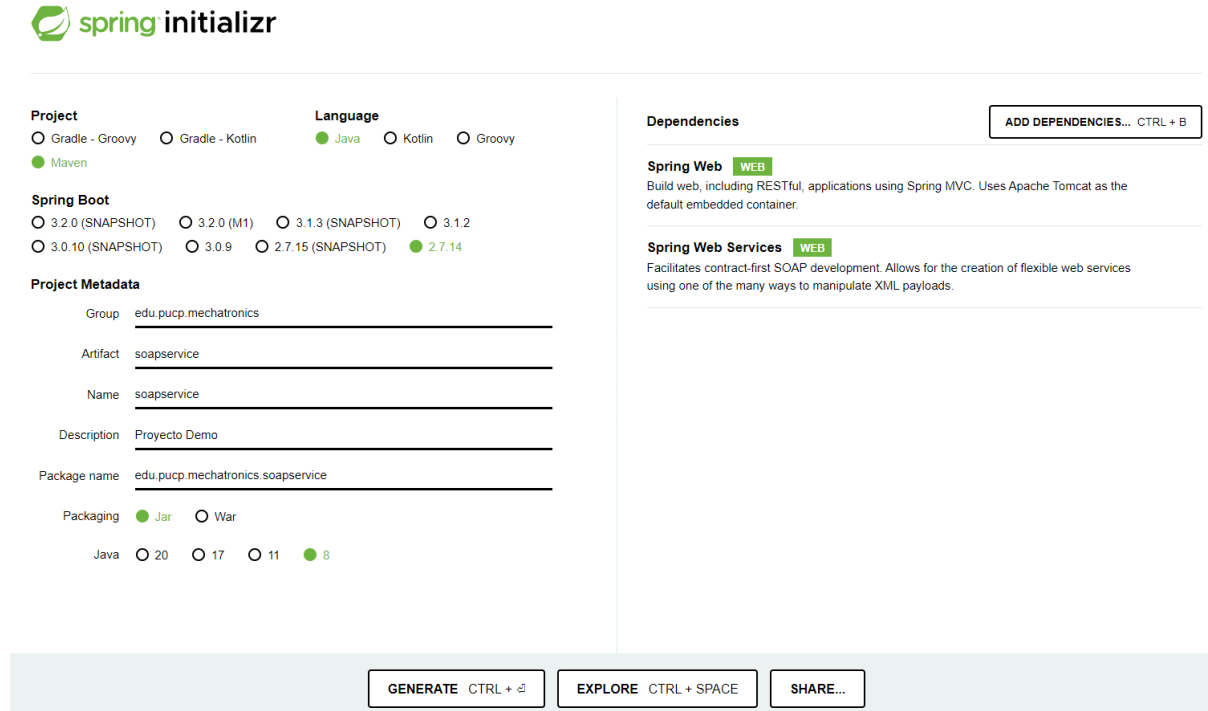


Configuración y creación del proyecto



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

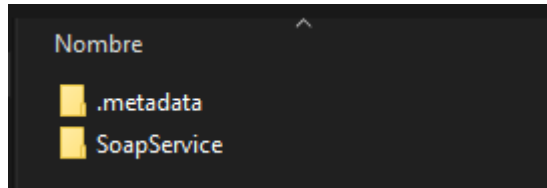


The image shows the Spring Initializr web form for creating a new project. The form is divided into several sections: Project, Language, Spring Boot, Project Metadata, Dependencies, and a bottom bar with action buttons.

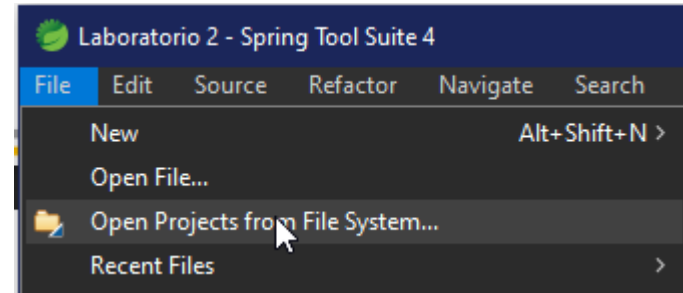
- Project:** Includes radio buttons for `Gradle - Groovy`, `Gradle - Kotlin`, `Java` (selected), `Kotlin`, and `Groovy`. There is also a `Maven` link.
- Language:** Includes radio buttons for `Java` (selected), `Kotlin`, and `Groovy`.
- Spring Boot:** Includes radio buttons for `3.2.0 (SNAPSHOT)`, `3.2.0 (M1)`, `3.1.3 (SNAPSHOT)`, `3.1.2`, `3.0.10 (SNAPSHOT)`, `3.0.9`, `2.7.15 (SNAPSHOT)`, and `2.7.14` (selected).
- Project Metadata:** Includes text input fields for `Group` (filled with `edu.pucp.mechatronics`), `Artifact` (filled with `soapservice`), `Name` (filled with `soapservice`), `Description` (filled with `Proyecto Demo`), and `Package name` (filled with `edu.pucp.mechatronics soapservice`).
- Dependencies:** Includes a button `ADD DEPENDENCIES... CTRL + B`. Below it, there are two dependency cards: `Spring Web` (with a `WEB` tag) and `Spring Web Services` (with a `WEB` tag). The `Spring Web Services` card has a description: "Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the many ways to manipulate XML payloads."
- Bottom Bar:** Includes three buttons: `GENERATE CTRL + G`, `EXPLORE CTRL + SPACE`, and `SHARE...`.

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



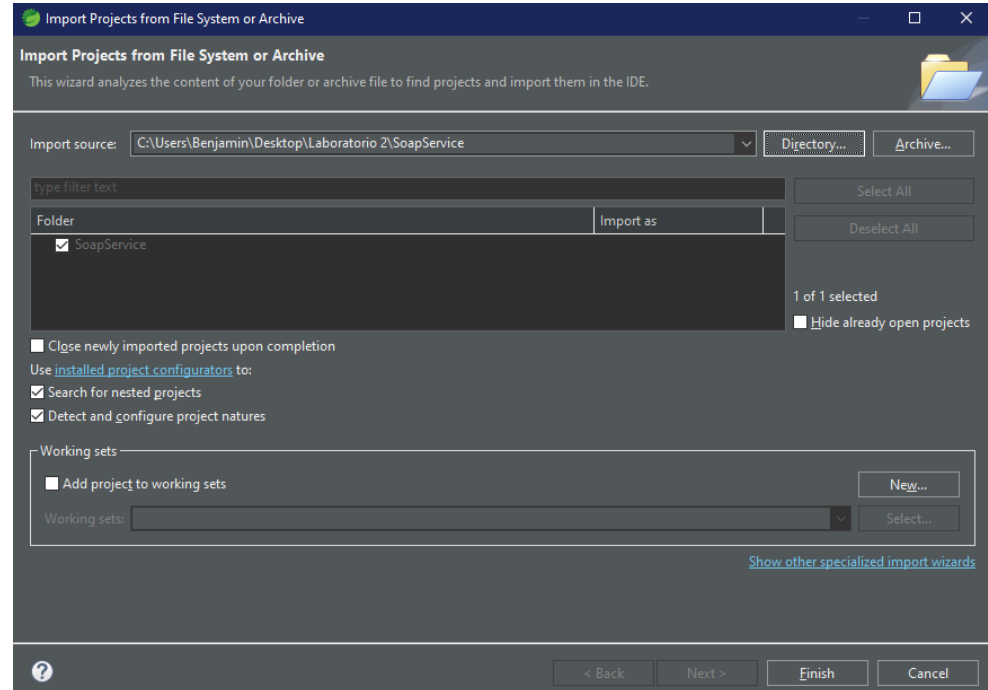
Para poder abrir este nuevo proyecto debemos abrir Spring Tool Suite, dar click en File y luego en Open Projects from File System



Configuración del servidor SOAP (3/6)

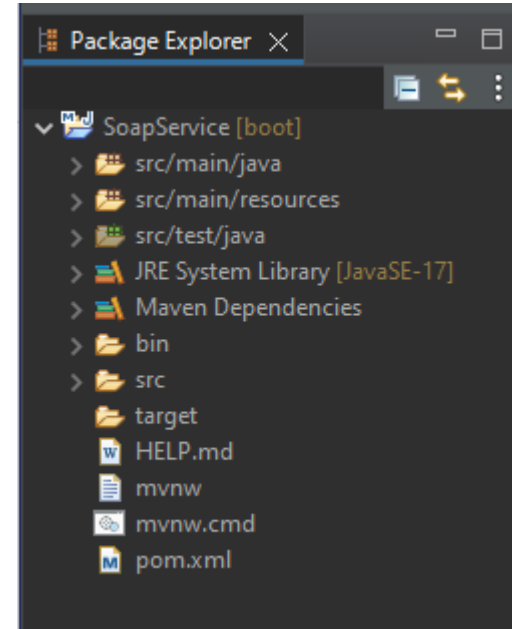
Aquí seleccionaremos el folder que pusimos en nuestro Workspace usando el botón Directory y le damos click en Finish

Aquí deberán esperar a que el proyecto termine de cargar.



Configuración del servidor SOAP (4/6)

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



Configuración del servidor SOAP (6/6)

En caso de que aparezcan errores :

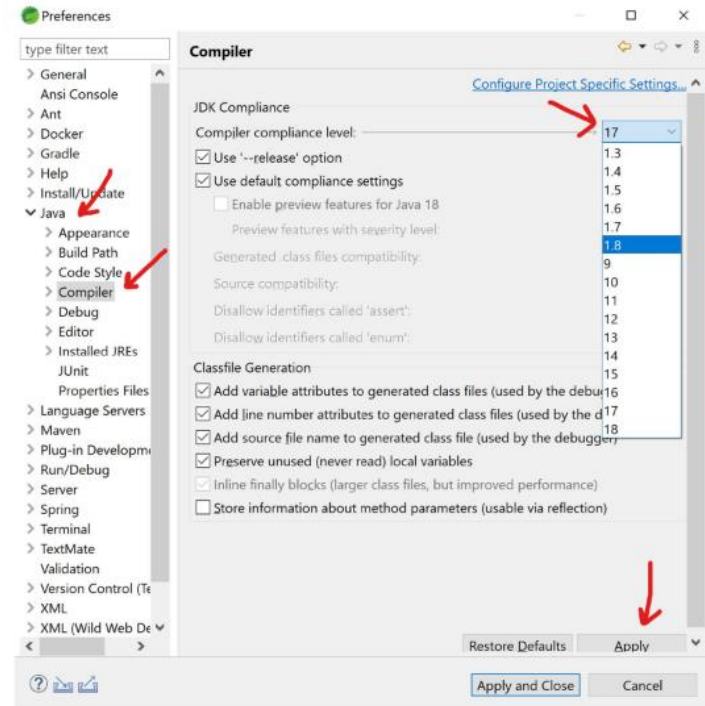
Build path specifies execution environment
JavaSE 1.8

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.

En caso de que aparezcan errores de JRE ,
añadir el archivo previo JRE tal cual hicimos
en el laboratorio anterior.



Configuración del servidor SOAP (5/6)

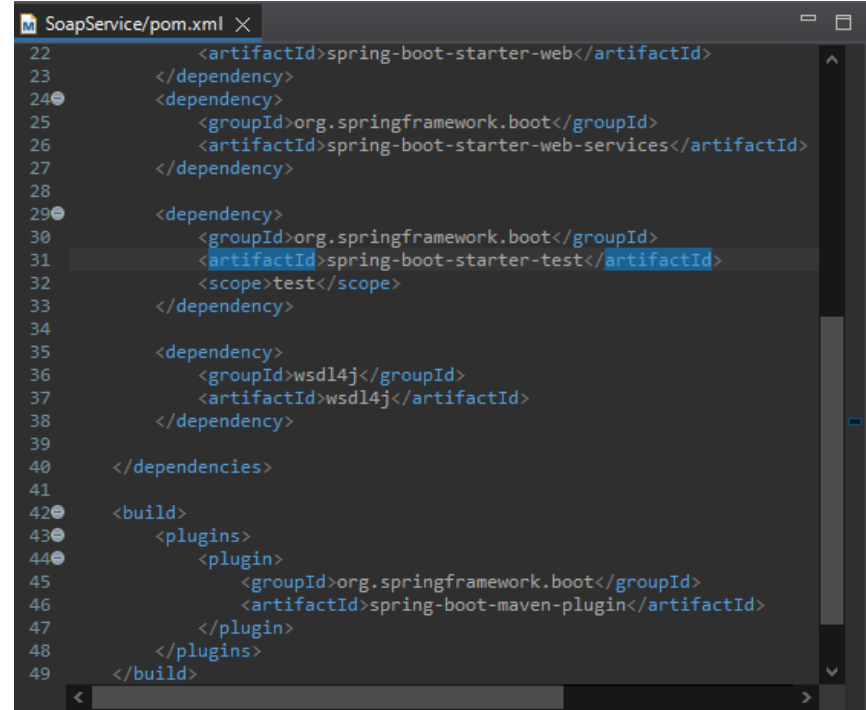
Una vez ya se tenga todo cargado, debemos modificar el archivo POM del proyecto.

Aquí debemos añadir EL Toolikt Web Services Description Language for Java (wsdl4j) :

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

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

Ademas de esto debemos de modificar la version de java a 1.8 ademas del spring boot a la version 2.7.15



```
22 <artifactId>spring-boot-starter-web</artifactId>
23 </dependency>
24 <dependency>
25 <groupId>org.springframework.boot</groupId>
26 <artifactId>spring-boot-starter-web-services</artifactId>
27 </dependency>
28
29 <dependency>
30 <groupId>org.springframework.boot</groupId>
31 <artifactId>spring-boot-starter-test</artifactId>
32 <scope>test</scope>
33 </dependency>
34
35 <dependency>
36 <groupId>wsdl4j</groupId>
37 <artifactId>wsdl4j</artifactId>
38 </dependency>
39
40 </dependencies>
41
42 <build>
43 <plugins>
44 <plugin>
45 <groupId>org.springframework.boot</groupId>
46 <artifactId>spring-boot-maven-plugin</artifactId>
47 </plugin>
48 </plugins>
49 </build>
```

Implementación del Proyecto SOAP



Configuración del proyecto (1/5)

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

Configuración del proyecto (2/5)

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

```
school.xsd X
http://www.w3.org/2001/XMLSchema (with embedded xml.xsd)
10 <?xml:stylesheet type="text/xsl" href="school.xsl" />
2 <!-- Schema for the school application -->
3
4 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://www.pucp.edu.pe/xml/school"
5 targetNamespace="http://www.pucp.edu.pe/xml/school" elementFormDefault="qualified">
6
7   <xs:element name="StudentDetailsRequest">
8     <xs:complexType>
9       <xs:sequence>
10         <xs:element name="name" type="xs:string"/>
11       </xs:sequence>
12     </xs:complexType>
13   </xs:element>
14
15   <xs:element name="StudentDetailsResponse">
16     <xs:complexType>
17       <xs:sequence>
18         <xs:element name="Student" type="tns:Student"/>
19       </xs:sequence>
20     </xs:complexType>
21   </xs:element>
22
23   <xs:complexType name="Student">
24     <xs:sequence>
25       <xs:element name="name" type="xs:string"/>
26       <xs:element name="age" type="xs:int"/>
27       <xs:element name="address" type="xs:string"/>
28     </xs:sequence>
29   </xs:complexType>
30 </xs:schema>
```

Configuración del proyecto (3/5)

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

Asi mismo es necesario añadir dos nueva dependencia para permitir el correcto funcionamiento :

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

```
<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/schools.xsd</source>  
</sources>  
</configuration>  
</plugin>
```

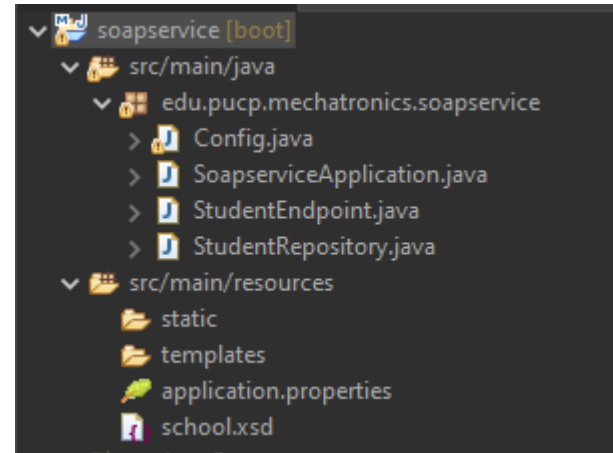
Configuración del proyecto (4/5)

Configuración del paquete edu.pucp.mechatronics.soapservice package

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

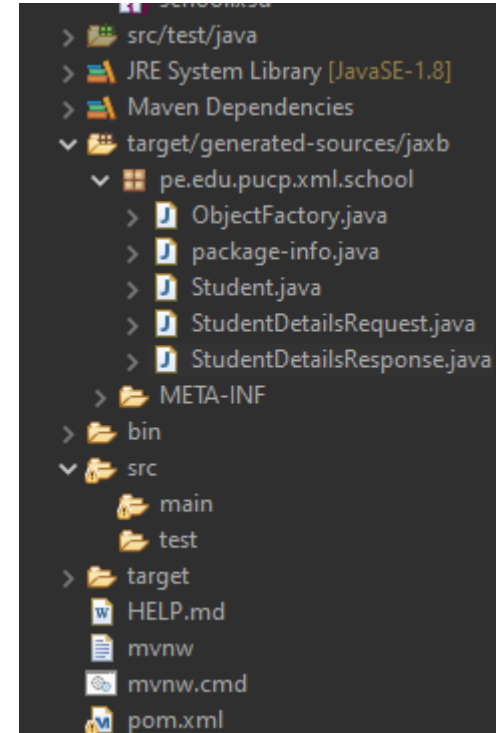
Este tendrá las clases :

- Config
- SoapServiceApplication
- StudentRepository
- StudentEndpoint



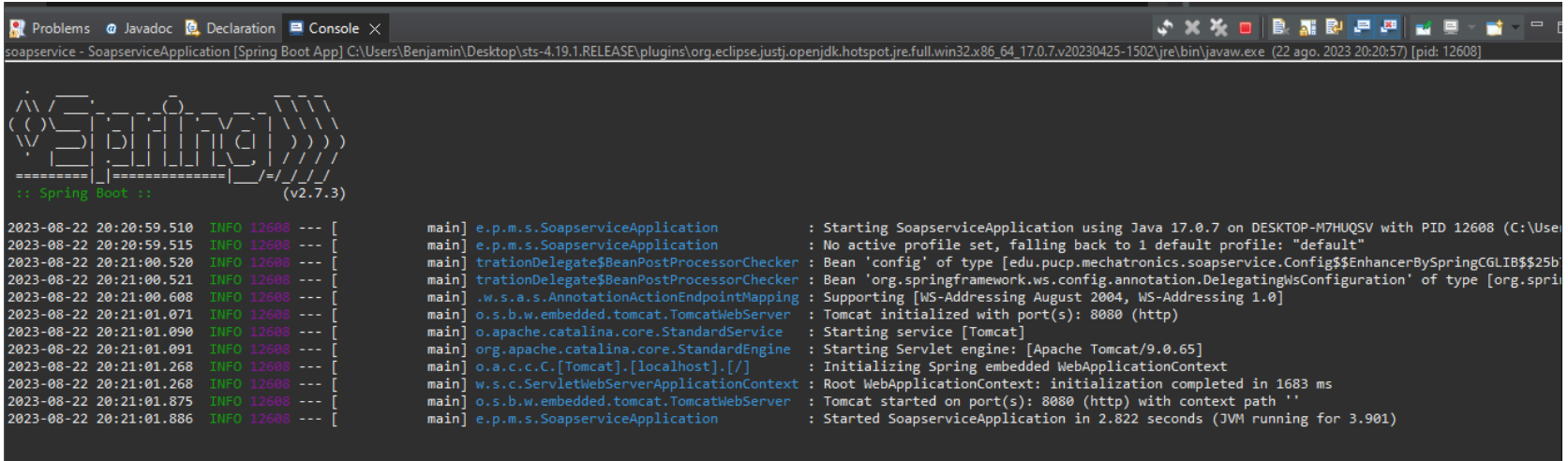
Configuración del proyecto (5/5)

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)

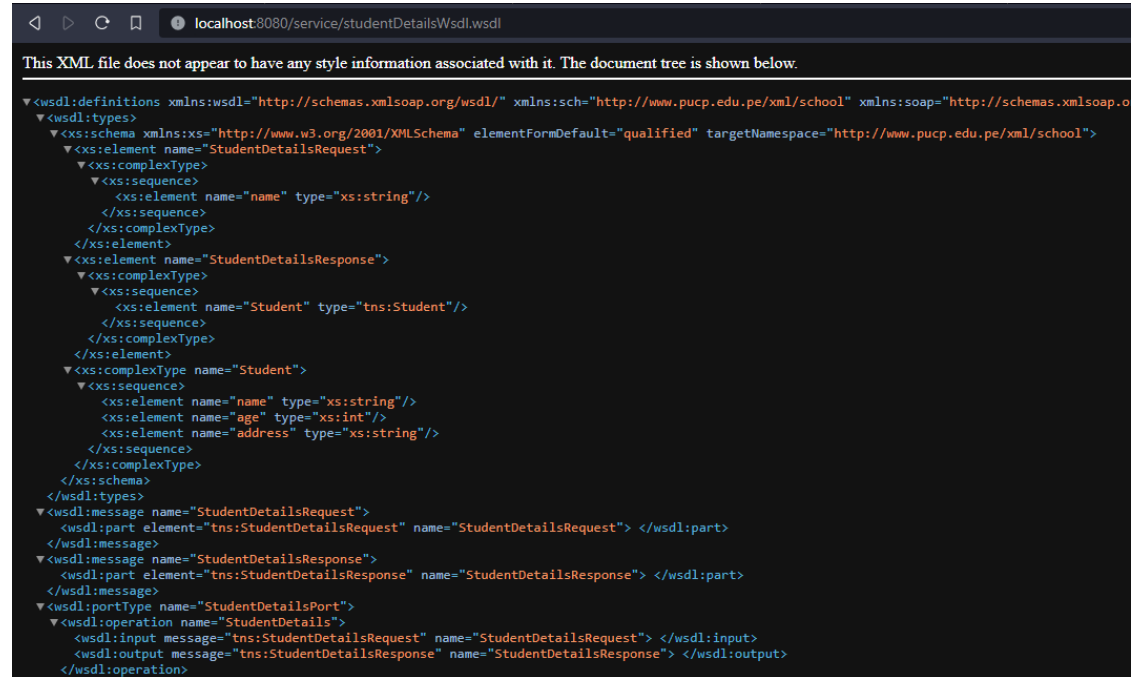
2023-08-22 20:20:59.510 INFO 12608 --- [main] e.p.m.s.SoapserviceApplication : Starting SoapserviceApplication using Java 17.0.7 on DESKTOP-M7HUQSV with PID 12608 (C:\Users\Benjamin\Desktop\sts-4.19.1.RELEASE\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502\jre\bin\javaw.exe)
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 bean validation: [org.springframework.beans.factory.config.ConfigurableBeanFactory$1]
2023-08-22 20:21:00.521 INFO 12608 --- [main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.ws.config.annotation.DelegatingWsConfiguration' of type [org.springframework.ws.config.annotation.DelegatingWsConfiguration] is not eligible for bean validation: [org.springframework.beans.factory.config.ConfigurableBeanFactory$1]
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 3.901s)
  
```

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



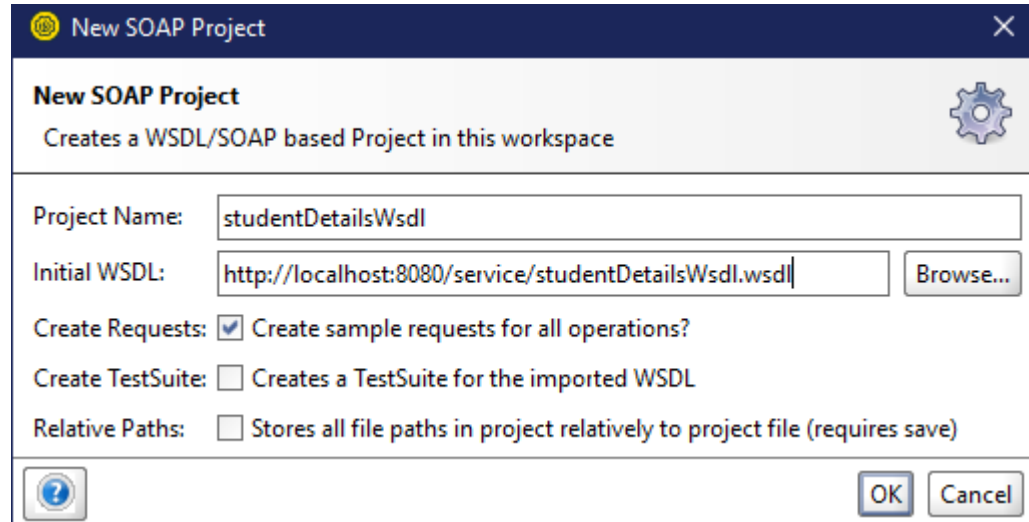
```
<?xml version='1.0' encoding='UTF-8'>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:sch="http://www.pucp.edu.pe/xml/school" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <wsdl:types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" 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>

```

Probando el proyecto (1/2)

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.



New SOAP Project

Creates a WSDL/SOAP based Project in this workspace

Project Name: studentDetailsWsdI

Initial WSDL: http://localhost:8080/service/studentDetailsWsdI.wsdl Browse...

Create Requests: ☒ Create sample requests for all operations?

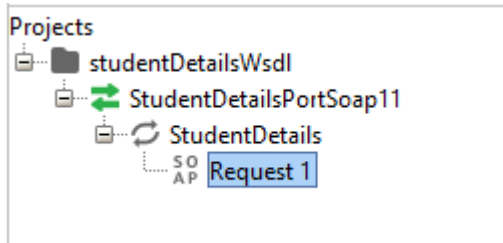
Create TestSuite: ☐ Creates a TestSuite for the imported WSDL

Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)

OK Cancel

Probando el proyecto (2/3)

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

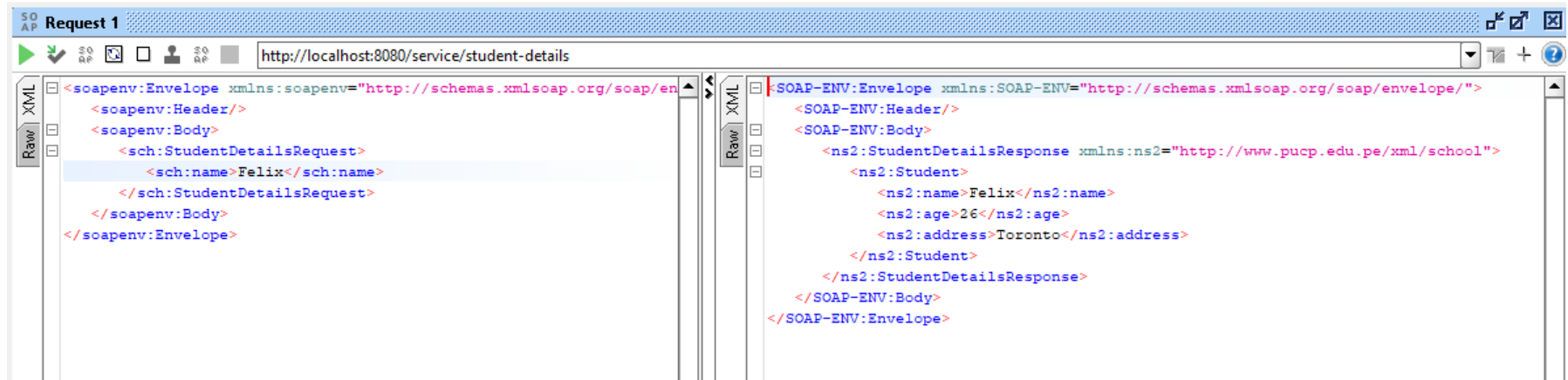


Esto abrirá un nuevo request , que nos permitirá añadir un estudiante, para eso ingresaremos un nombre que este en StudentRepository entre <sch:name> y enviaremos el request



Probando el proyecto (3/3)

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



The screenshot displays a SOAP client interface with two panels. The left panel shows the outgoing request, and the right panel shows the incoming response. Both are in XML format.

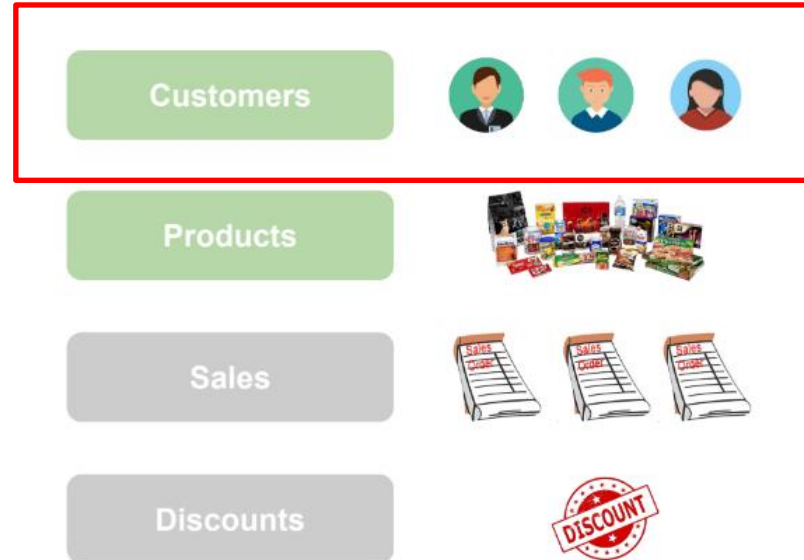
Request (Left Panel):

```
<?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>
```

Response (Right Panel):

```
<?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



Desarrollo de nuestra aplicación REST Sistema de Clientes para un negocio



Desarrollo de nuestra aplicación REST

Metodos a utilizar

HTTP METHODS

GET

POST

PUT

Delete

Request

Response

Get Customers

GET /api/customers

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

Desarrollo de nuestra aplicación REST

Metodos a utilizar

HTTP METHODS

GET

POST

PUT

Delete

Get a Customer

Request

GET /api/customers/1

Response

{ id: 1, name: 'Hugo' }

Desarrollo de nuestra aplicación REST

Metodos a utilizar

HTTP METHODS

GET

POST

PUT

Delete

Request

Create a Customer

POST /api/customers

{ name: 'Kim' }

Response

{ id: 1, name: 'Kim' }

Desarrollo de nuestra aplicación REST

Metodos a utilizar

HTTP METHODS

GET

POST

PUT

Delete

Request

PUT /api/customers/**1**

{ name: 'Fernando' }

Response

{ id: 1, name: 'Fernando' }

Update a Customer

Desarrollo de nuestra aplicación REST

Metodos a utilizar

HTTP METHODS

GET

POST

PUT

DELETE

Delete a Customer

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

Response

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

Desarrollo de nuestra aplicación

Requisitos

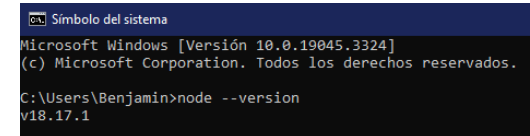
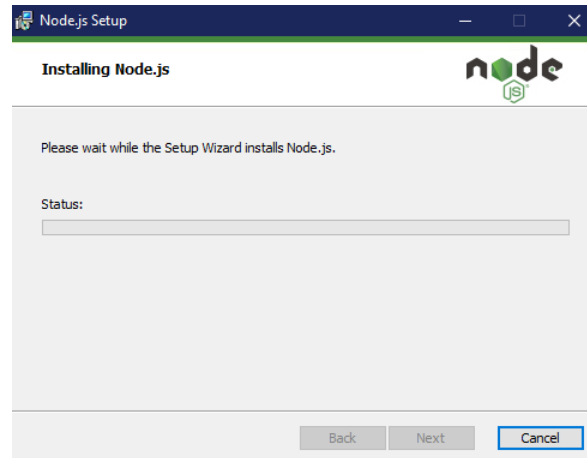
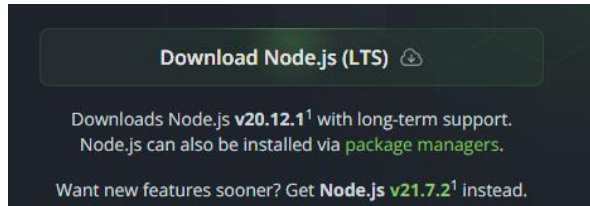
- Node JS
- Visual Studio Code
- npm
- Express JS
y Postman



Node JS

Para su instalación solo deben descargar la versión LTS mas reciente desde <https://nodejs.org/en/> e instalarlo

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



Visual Studio Code

Para su instalación solo deben descargar la versión LTS mas reciente desde

<https://code.visualstudio.com/download> e instalarlo



↓ **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 app.js` para ejecutarlo y para probarlo podemos usar : `http://localhost:3000/api/customers`

```
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> |
```

```
JS app.js
JS app.js > ...
1  const express = require('express');
2  const app = express();
3
4  app.use(express.json())
5
6  const customers = [
7    {id: 1, name : "Hugo"},
8    {id: 2 , name: "Kim"}
9  ]
10
11 app.get('/', (req, res) =>{
12   res.send('Hola mundo');
13 })
14
15 app.get(['/api/customers', (req, res)=>{
16   res.send(customers);
17 }]
18
19 app.listen(3000,
20   ()=> console.log('Escuchando en el puerto 3000...'));
```

Configuración del proyecto

Continuando con la implementación de nuestro proyecto , implementaremos la obtención de usuarios mediante el id

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

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)
})
```

HTTP response status codes

Informational responses	100 - 199
Successful responses	200 - 299
Redirection messages	300 - 399
Client error responses	400 - 499
Server error responses	500 - 599

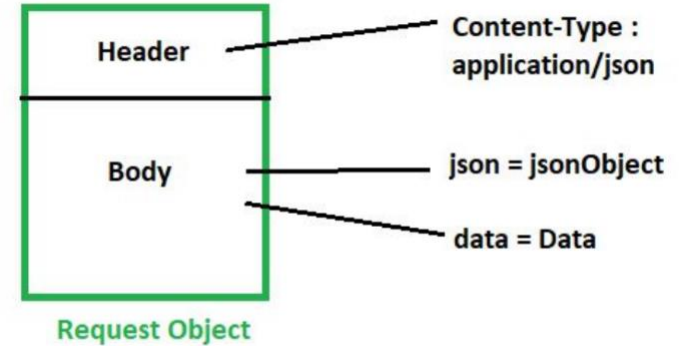
Configuración del proyecto

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 <https://www.postman.com/downloads/> e instalarlo

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](#).

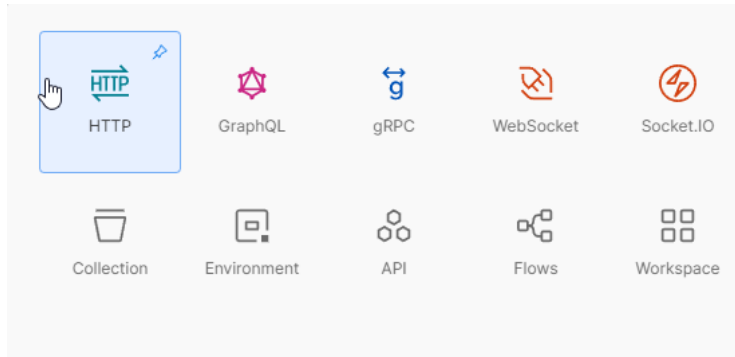
[Release Notes](#) · [Product Roadmap](#)

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

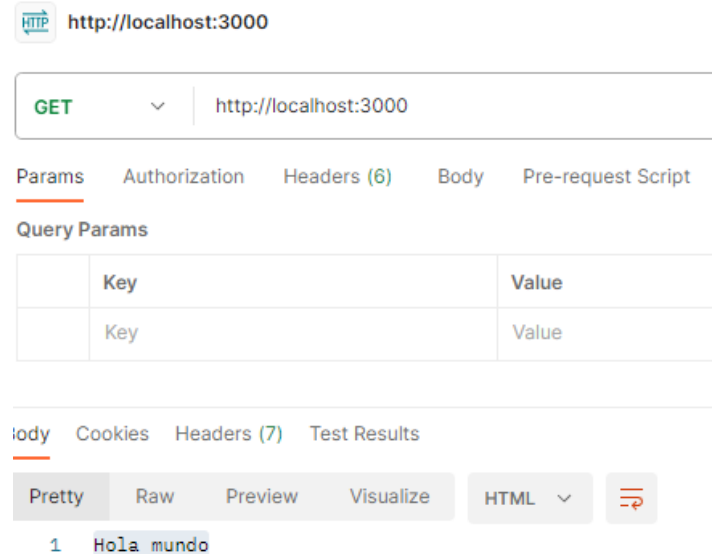


Probando el proyecto

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

Aquí podremos probar tanto el método get, como el método post que ya se han creado

 `http://localhost:3000/api/customers/1`


GET `http://localhost:3000/api/customers/1`

Params Authorization Headers (6) Body Pre-request Script

Query Params

Key	Value
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON 

```

1 [
2   {
3     "id": 1,
4     "name": "Hugo"
5   }
6 ]
  
```


GET `http://localhost:3000/api/customers`

Params Authorization Headers (6) Body Pre-request Script

Query Params


Key	Value
Key	Value

Body Cookies Headers (7) Test Results


Pretty Raw Preview Visualize JSON 


```

1 [
2   {
3     "id": 1,
4     "name": "Hugo"
5   },
6   {
7     "id": 2,
8     "name": "Kim"
9   }
10 ]
  
```

 `http://localhost:3000/api/customers`

POST `http://localhost:3000/api/customers`

Params Authorization Headers (8) Body  Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON 

```

1 [
2   {
3     "name": "Benjamin"
4   }
5 ]
  
```

Body Cookies Headers (7) Test Results  200 OK 9 ms

Pretty Raw Preview Visualize JSON 

```

1 [
2   {
3     "id": 4,
4     "name": "Benjamin"
5   }
6 ]
  
```

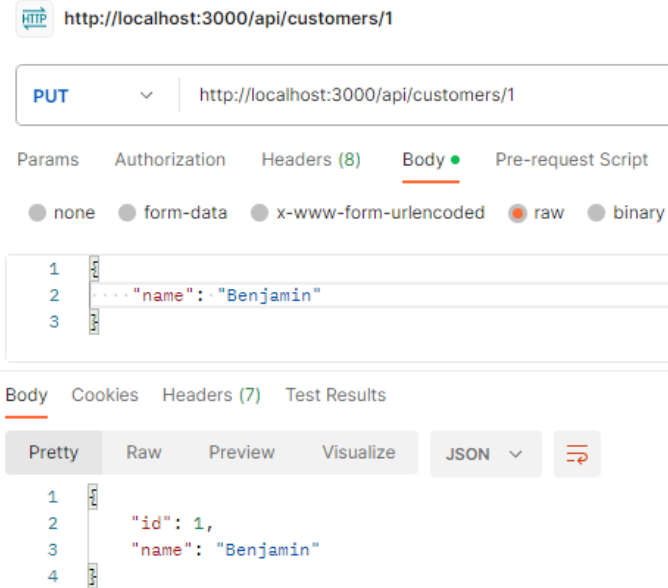
Configuración del proyecto

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(400).send("El nombre debe tener al menos 3 caracteres")
    return;
  }

  customer.name = req.body.name;
  res.send(customer);
})

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

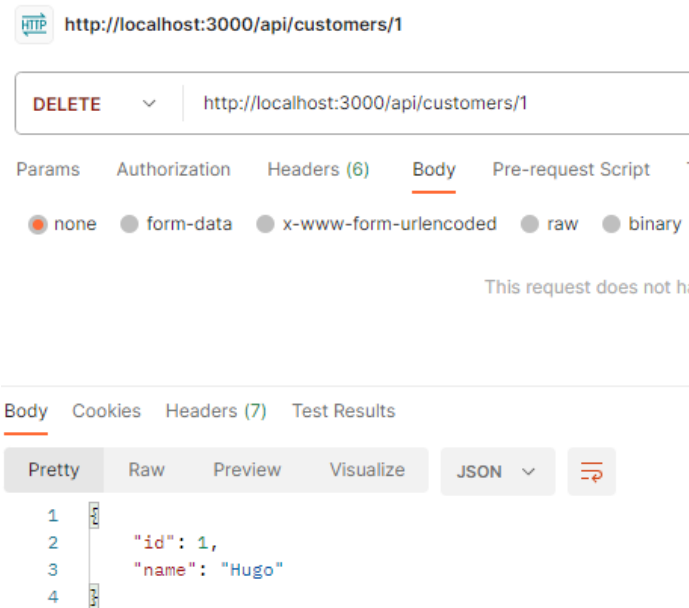


Configuración del proyecto

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);
})

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



Información Importante

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

pet Everything about your Pets		Find out more: http://swagger.io	^
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	▼	🔒