PRACTICA DE MICROSERVICIOS: API REST

SPS Solutions

Marko Alan Bibiano Cortes
Bcortesmarko@gmail.com

Contenido

Introducción

En este documento se explica la configuración y pasos realizados para realizar una práctica sencilla de microservicios, la programación de una API Rest básica.

Maven

Maven se utiliza en la gestión y construcción de software. Posee la capacidad de realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Es decir, hace posible la creación de software con dependencias incluidas dentro de la estructura del JAR. Es necesario definir todas las dependencias del proyecto (librerías externas utilizadas) en un fichero propio de todo proyecto Maven, el POM (Project Object Model). Este es un archivo en formato XML que contiene todo lo necesario para que a la hora de generar el fichero ejecutable de nuestra aplicación este contenga todo lo que necesita para su ejecución en su interior.

Glass Fish Server

Oracle GlassFish Server ofrece un servidor para poder desarrollar tanto aplicaciones Java Platform Enterprise Edition (Java EE) como webs Java Web Services. Fue creado en base a GlassFish Server Open Source Edition, un servidor de aplicaciones de código abierto de la comunidad de GlassFish. Oracle GlassFish Server ofrece un servidor de aplicaciones Java EE flexible, ligero y listo para la producción.

Configurar Maven de NetBeans

NetBeans viene con una configuración de Maven instalada, para poder usarla tiene que especificar algunas cosas.

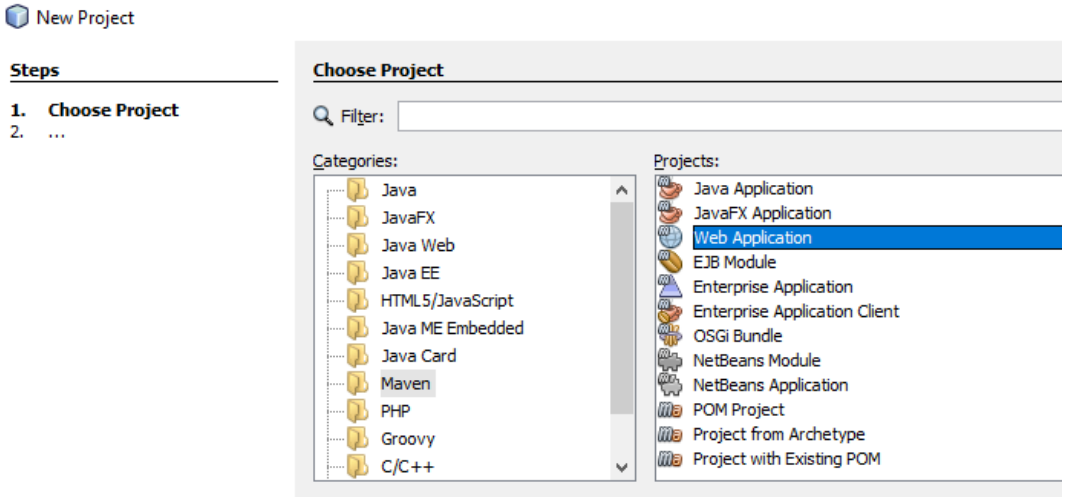
1. Abrimos la siguiente ruta C:\Program Files\NetBeans 8.2\java\maven\conf
2. Abrir el archivo de configuración de Maven settings.xml
3. Agregamos un mirror para el repositorio de Maven

```
<mirror>
  <id>mirror1</id>
  <mirrorOf>central</mirrorOf>
  <name>mirror1</name>
  <url>https://repo.maven.apache.org/maven2/</url>
</mirror>
```

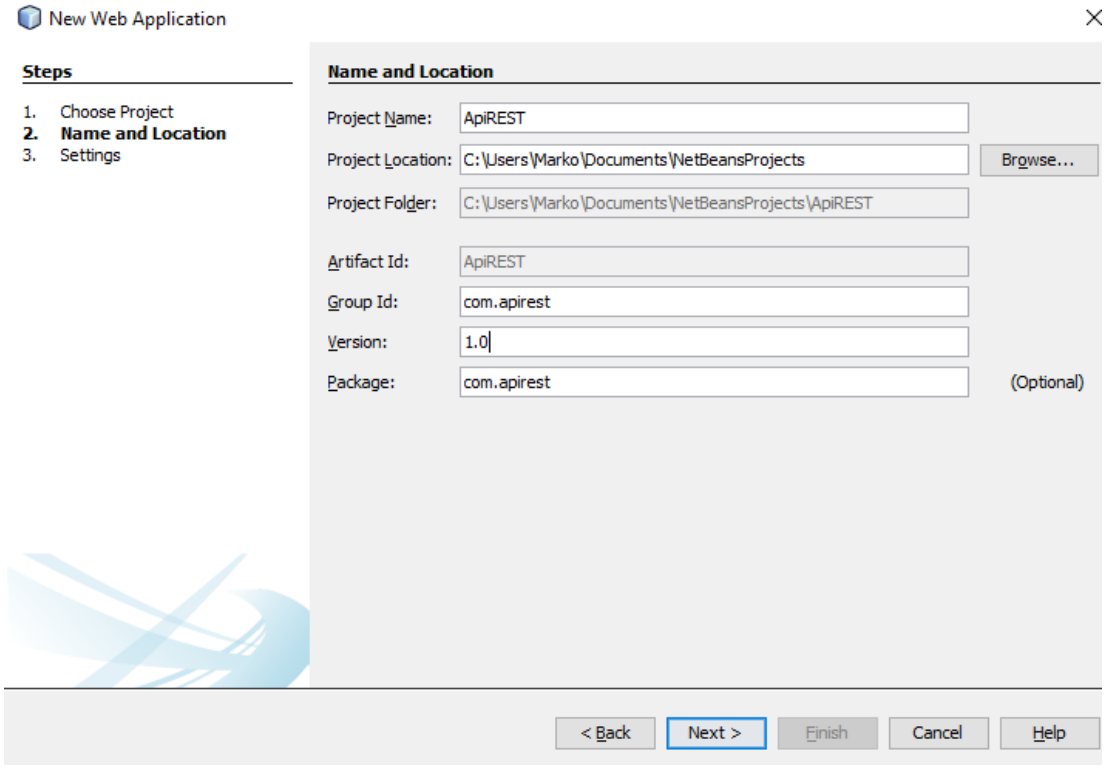
4. Guardar los cambios como administrador
5. Reiniciar NetBeans

Crear un proyecto Maven

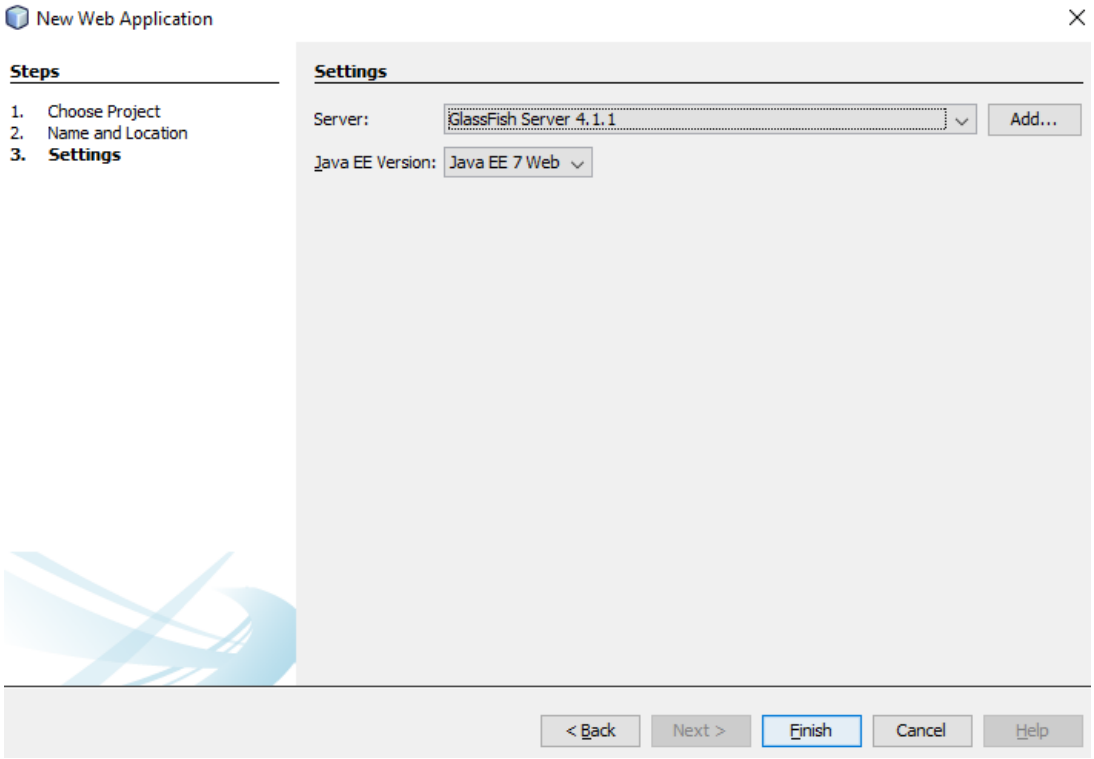
1. En NetBeans seleccionamos crear un proyecto
2. Seleccionamos Maven > Web Aplicación



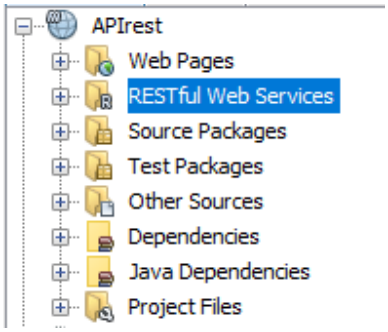
3. Escribimos el nombre del proyecto



4. Seleccionamos el servidor GlassFish Server 4.1.1, viene por defecto en la configuración de NetBeans:

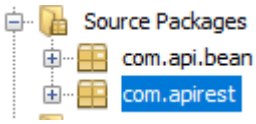


5. Se creará un proyecto con la siguiente estructura:

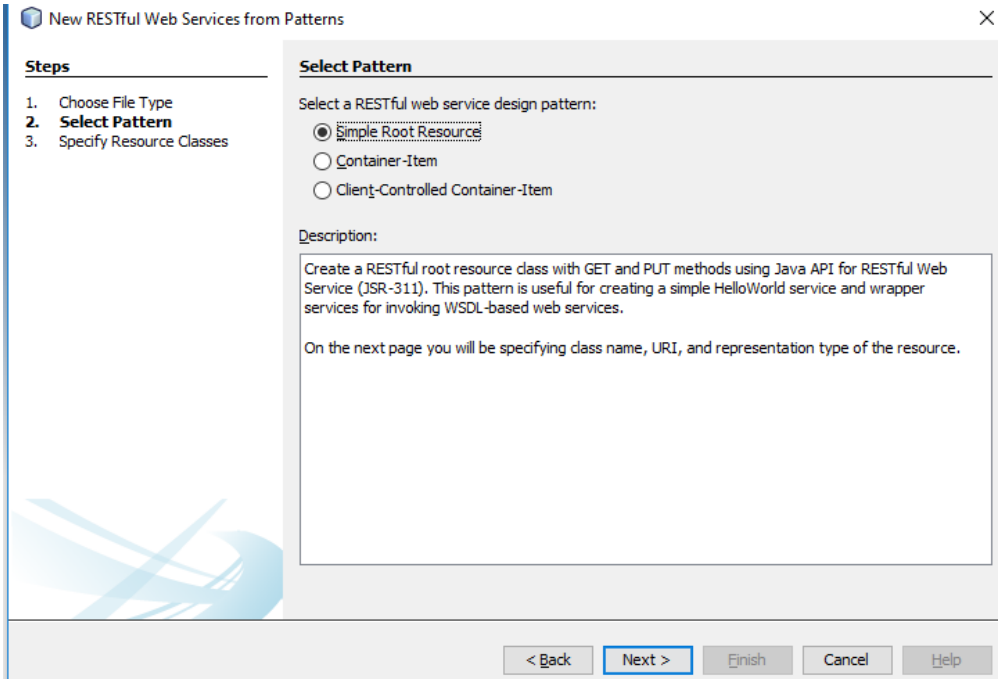


Crear Paquetes y clases

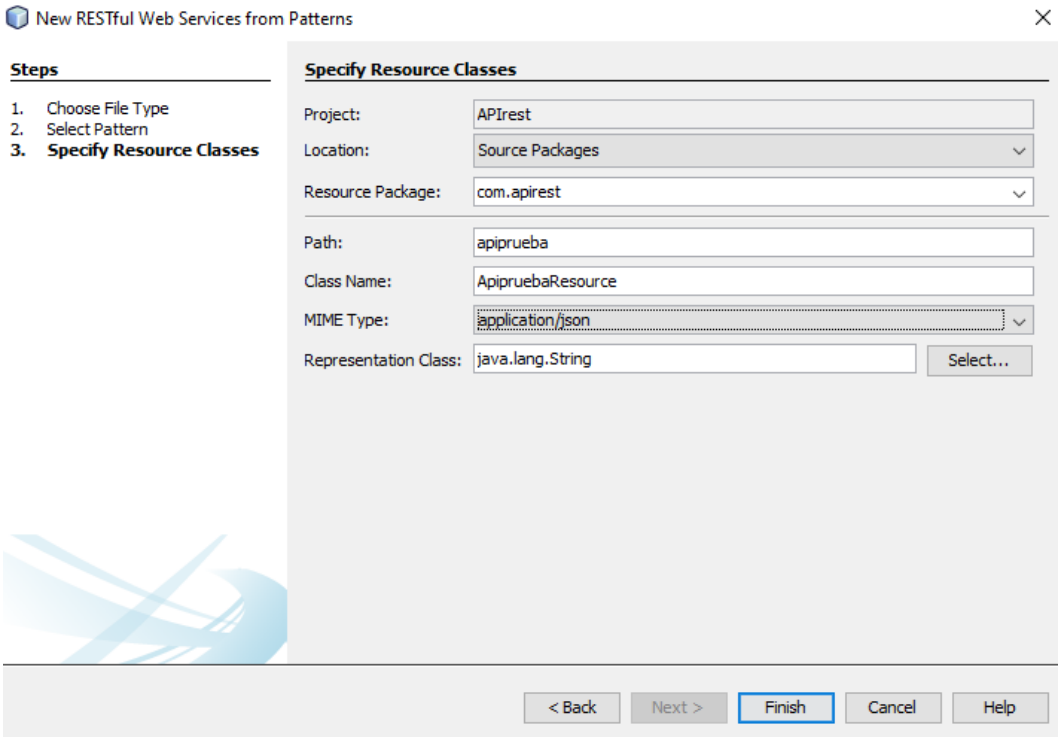
1. En el Source Packages crearemos los siguientes paquetes



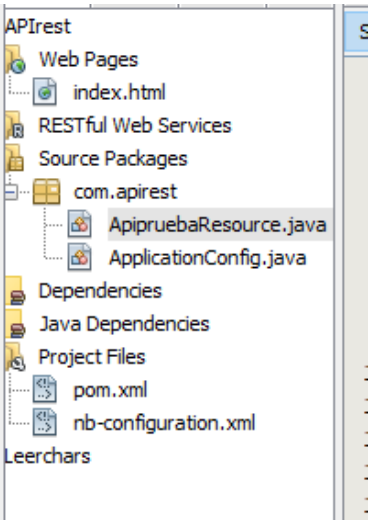
2. En com.apirest, hacemos clic derecho en el paquete, seleccionamos New y buscamos la opción de **New Restful Web Services from Patterns**:



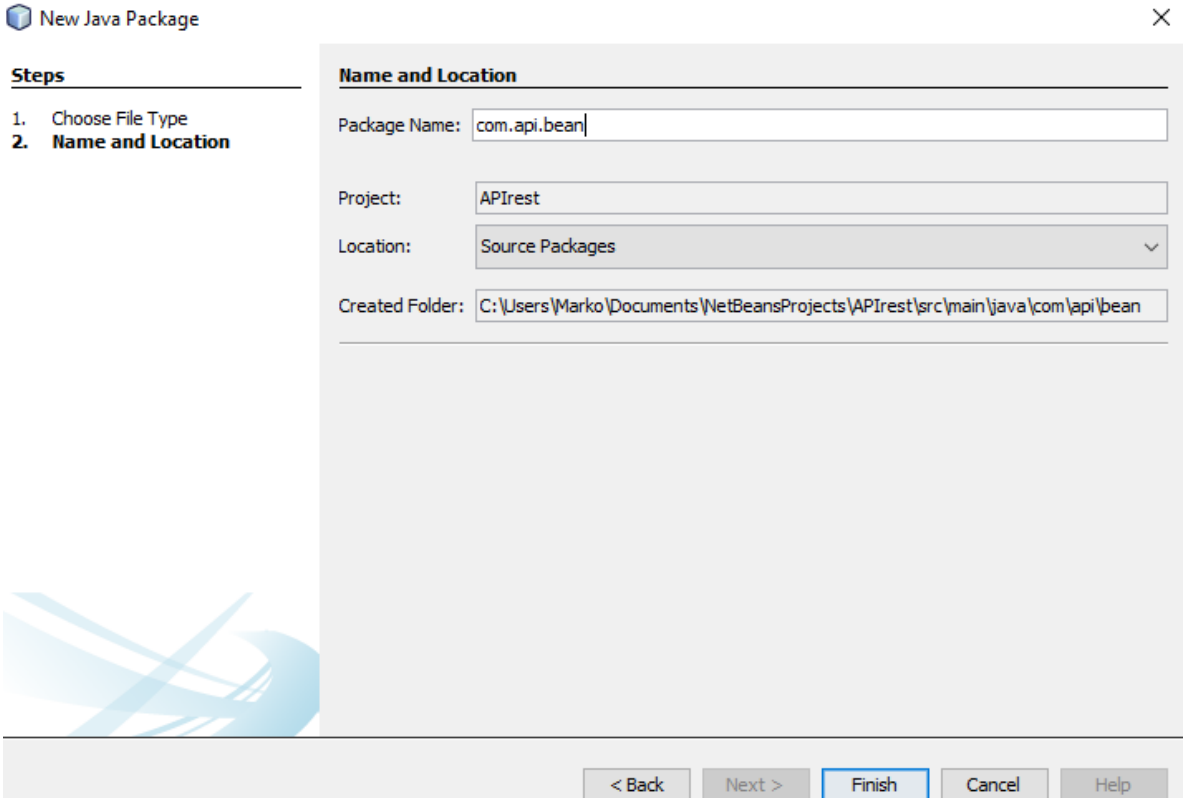
3. En la siguiente pantalla debemos llenar lo siguientes campos según corresponda:
    - Path: nombre por el cual realizaremos la consulta a nuestro API.
    - Class Name: nombre de la clase donde programaremos nuestra función GET.
    - MIME Type: Tipo de contenido que dará como respuesta nuestro API.



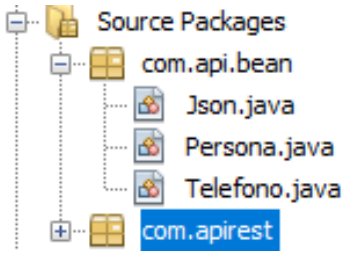
4. Se crean los siguientes archivos:



5. Agregamos un nuevo paquete llamado com.api.bean



6. Agregamos la clase persona, teléfono y json



Agregar dependencia al archivo POM

Antes de continuar, agregamos la dependencia gson en el archivo POM.xml del proyecto, esta dependencia es para convertir objetos a json.

```
<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>1.7.1</version>
</dependency>
```

Programar Clases

En el otro paquete donde creamos las clases Personas, Teléfono y Json creamos una clase normal con sus gets y sets

1. Clase persona

```
public class Persona {
    private String cedula;
    private String nombre;
    private Telefono telefono;

    public String getCedula() {
        return cedula;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Telefono getTelefono() {
        return telefono;
    }

    public void setTelefono(Telefono telefono) {
        this.telefono = telefono;
    }

}
```

2. Clase Telefono

```java
package com.api.bean;

/**
 *
 * @author Marko
 */
public class Telefono {
    private String telefono;
    private String operador;

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public String getOperador() {
        return operador;
    }

    public void setOperador(String operador) {
        this.operador = operador;
    }


}
```

3. Clase Json

```java
package com.api.bean;

/**
 *
 * @author Marko
 */
public class Json {
    private Persona persona;

    public Persona getPersona() {
        return persona;
    }

    public void setPersona(Persona persona) {
        this.persona = persona;
    }


}
```

Programar API Rest sencilla

1. En ApipruebaResource, al inicio de la clase configuramos el path

```
/**
 * REST Web Service
 *
 * @author Marko
 */
@Path("api/sps/helloworld")
public class ApipruebaResource {
```

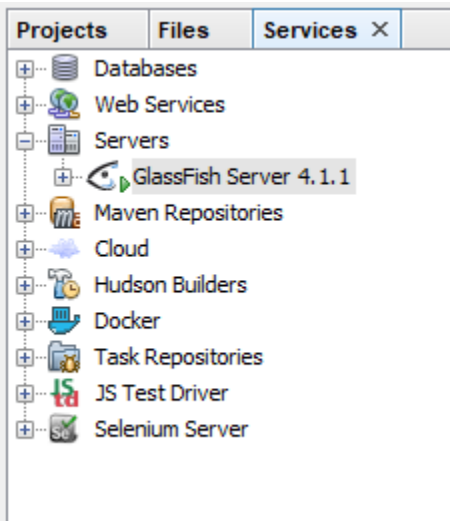
2. Modificamos la función getJson a **consultaPersona** y agregamos las variables queryparam que vamos a recibir.

```
@GET
@Path("/v1")
@Produces(MediaType.APPLICATION_JSON)
public String consultaPersona(@QueryParam("cedula") String cedula,
                              @QueryParam("nombre") String nombre,
                              @QueryParam("telefono") String telefono,
                              @QueryParam("operador") String operador) {

    Telefono tel = new Telefono();
    Persona persona = new Persona();

    tel.setTelefono(telefono);
    tel.setOperador(operador);

    persona.setCedula(cedula);
    persona.setNombre(nombre);
    persona.setTelefono(tel);

    Json obj = new Json();
    obj.setPersona(persona);

    Gson gson = new Gson();
    String jsonString = gson.toJson(obj);

    return jsonString;
    //TODO return proper representation object
   // throw new UnsupportedOperationException();
}
```

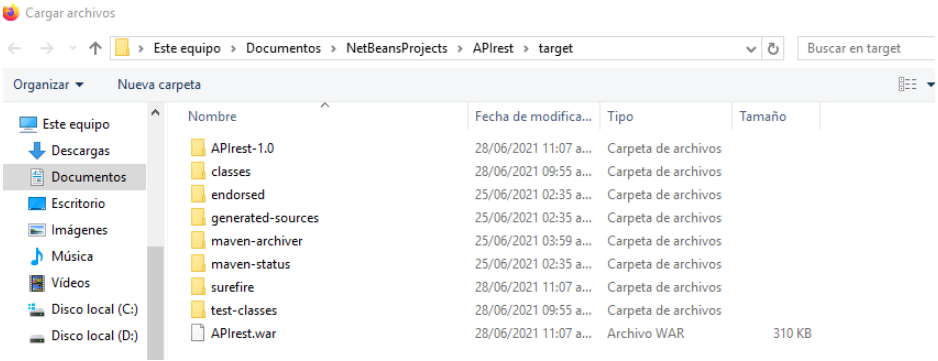
- Como apreciamos en la imagen anterior, creamos un objeto de Persona y Teléfono, le pasamos las variables que recibimos como argumento.
- Creamos un objeto de Json y metimos el objeto de persona que creamos.
- Con la dependencia Gson que agregamos al POM creamos un objeto y usamos la función toJson para convertir el objeto a Json.
- Devolvemos el resultado en jsonString.

Prueba

1. Iniciamos el servidor GlassFish
    - Seleccionamos la pestaña Services

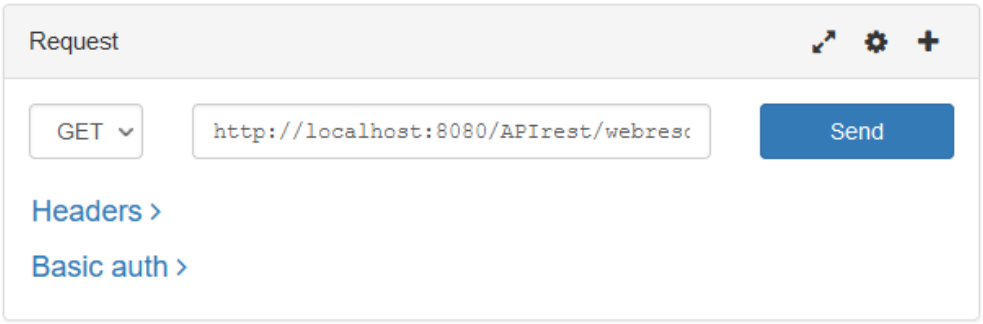


    - Clic secundario en el servidor y seleccinamos Start
    - El Admin server se levantara en la siguiente URL: http://localhost:4848/
2. Hacemos clic Build Project para construir el archivo WAR
3. Desplegamos el archivo WAR en el servidor



4. Usamos el plugin RESTED desde el navegador para probar la API

http://localhost:8080/APIrest/webresources/api/sps/helloworld/v1?cedula=9876&nombre=Marko+Alan&telefono=445566&operador=Telcel




```
{
  "persona": {
    "cedula": "9876",
    "nombre": "Marko Alan",
    "telefono": {
      "telefono": "445566",
      "operador": "Telcel"
    }
  }
}
```