# Instituto Politécnico Nacional
## Escuela Superior de Computo
Ingeniería en Sistemas Computacionales.


Unidad de Aprendizaje:
Sistemas Distribuidos.

Nombre del Profesor:
Carreto Arellano Chadwick.

Nombre del Alumno:

Cruz Cubas Ricardo

Grupo:
7CM1.

Practica 6:

Web Service.

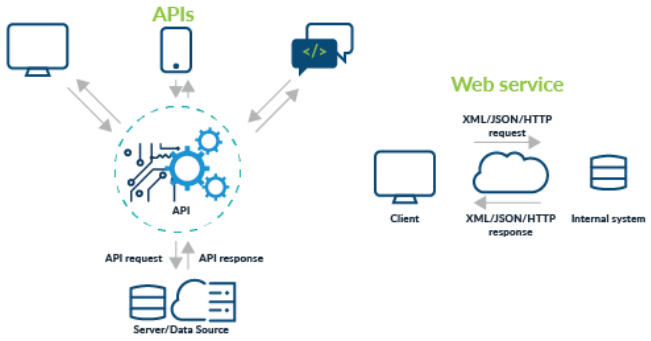
# Contenido

# Introducción

Los Web Services, o servicios web, son un componente fundamental dentro de la arquitectura de aplicaciones distribuidas modernas. Se trata de sistemas diseñados para permitir la interoperabilidad entre distintas aplicaciones que pueden estar escritas en diferentes lenguajes de programación y ejecutarse en plataformas diversas. La esencia de un Web Service radica en su capacidad de facilitar la comunicación y el intercambio de datos entre sistemas heterogéneos a través de una red, típicamente Internet o una intranet.

Un Web Service expone una funcionalidad específica de una aplicación en forma de interfaz accesible mediante protocolos estándar, principalmente HTTP. Esto permite que otros sistemas puedan descubrir, invocar y utilizar esta funcionalidad sin necesidad de conocer su implementación interna. Es decir, una aplicación cliente puede solicitar datos o enviar información a un servicio web remoto sin preocuparse por cómo se procesará esa solicitud en el servidor, siempre y cuando se sigan las reglas del protocolo y del formato de mensajes definido.

Los servicios web suelen usar formatos de datos estructurados y fácilmente intercambiables, como XML (eXtensible Markup Language) o JSON (JavaScript Object Notation), para codificar los mensajes enviados entre el cliente y el servidor. El uso de estos formatos garantiza que los datos puedan ser comprendidos por aplicaciones desarrolladas en entornos distintos, contribuyendo así a la interoperabilidad entre plataformas.




Existen principalmente dos tipos de arquitecturas para servicios web: SOAP (Simple Object Access Protocol) y REST (Representational State Transfer). SOAP es un protocolo basado en XML que define estrictamente cómo se deben construir los mensajes, incluyendo encabezados, cuerpo, errores y otros aspectos. Este enfoque, aunque más robusto en términos de seguridad y transacciones complejas, puede resultar más pesado y complejo de implementar. En contraste, REST no es un protocolo sino un estilo arquitectónico que aprovecha las capacidades del protocolo HTTP para operar con recursos a través de métodos estándar como GET, POST, PUT y

DELETE. REST se ha vuelto muy popular en el desarrollo de APIs modernas debido a su simplicidad, eficiencia y facilidad de integración con tecnologías web actuales.

Los servicios web han transformado la forma en que se desarrollan las aplicaciones distribuidas y han sido clave en la evolución hacia arquitecturas orientadas a servicios (SOA, por sus siglas en inglés) y, más recientemente, hacia microservicios. Gracias a los Web Services, es posible construir sistemas escalables, modulares y fáciles de mantener, donde distintos componentes pueden evolucionar de forma independiente y comunicarse entre sí de manera segura y estandarizada.

Además, los servicios web juegan un papel crucial en la integración de sistemas empresariales, la comunicación entre aplicaciones móviles y servidores, el desarrollo de aplicaciones web modernas, y la creación de plataformas basadas en la nube. En muchos casos, los servicios web permiten que terceros accedan a funcionalidades específicas de una aplicación mediante APIs públicas, lo cual ha dado lugar a todo un ecosistema de aplicaciones interconectadas y servicios digitales.

En conclusión, los Web Services son una piedra angular de la conectividad moderna entre aplicaciones y sistemas distribuidos. Su capacidad de abstraer la complejidad del backend, facilitar la interoperabilidad y estandarizar las comunicaciones ha hecho que sean ampliamente adoptados en casi todos los ámbitos de la informática, desde pequeñas aplicaciones móviles hasta complejas plataformas corporativas.

# Planteamiento del Problema

En el desarrollo de aplicaciones distribuidas o sistemas que requieren el intercambio de datos entre distintos componentes o plataformas, surge la necesidad de implementar mecanismos eficientes, seguros y estandarizados para la comunicación remota. Uno de los enfoques más utilizados para este fin son los servicios web (Web Services), los cuales permiten a diferentes aplicaciones interactuar entre sí a través de protocolos de red, sin importar el lenguaje de programación o el sistema operativo en el que se ejecuten.

Actualmente, muchas organizaciones enfrentan el reto de exponer funcionalidades simples o complejas a otras aplicaciones o usuarios externos mediante una interfaz accesible desde Internet o una red interna. Sin embargo, incluso tareas aparentemente simples como la suma de dos números pueden requerir una estructura robusta si deben ser utilizadas por múltiples clientes, en diferentes entornos, de manera confiable y reutilizable.

En este contexto, se plantea la necesidad de desarrollar un servicio web básico que permita realizar operaciones aritméticas, comenzando por una de las más elementales: la suma de dos números enteros. Esta funcionalidad servirá como punto de partida para demostrar cómo construir, publicar e invocar servicios web utilizando tecnologías estándar como Java y el protocolo SOAP, con el objetivo de facilitar la interoperabilidad entre diferentes sistemas.

Este problema no solo aborda la implementación técnica de un servicio, sino también la comprensión del ciclo completo de desarrollo e implementación de un Web Service, incluyendo la definición de operaciones, la estructura de los mensajes, y su publicación para ser consumidos por aplicaciones cliente. Además, permite sentar las bases para la expansión futura del servicio, integrando operaciones más complejas o servicios adicionales.

Por tanto, el desarrollo de este servicio web de suma constituye una solución inicial y didáctica a la problemática de compartir funcionalidades entre aplicaciones mediante una arquitectura orientada a servicios, destacando la importancia de la modularidad, la reutilización de código y la independencia de plataformas en el desarrollo de software moderno.

# Propuesta de Solución

Para dar respuesta a la necesidad de facilitar la comunicación e interoperabilidad entre aplicaciones a través de una funcionalidad compartida, se propone el desarrollo e implementación de un servicio web (Web Service) en Java que permita realizar la operación de suma entre dos números enteros. Este servicio será expuesto mediante el protocolo SOAP utilizando la especificación JAX-WS (Java API for XML Web Services), lo que garantiza compatibilidad con una amplia variedad de plataformas y lenguajes de programación.

La solución consistirá en construir una clase Java anotada con las etiquetas proporcionadas por JAX-WS, como @WebService, @WebMethod y @WebParam, para definir de manera clara y estructurada tanto el servicio como sus operaciones y parámetros. En este caso, la operación principal será suma(numero1, numero2), la cual tomará dos números enteros como entrada y devolverá como resultado su suma.

Este servicio será desplegado en un servidor de aplicaciones compatible con servicios web, como GlassFish o Apache TomEE, permitiendo su publicación en una URL accesible por clientes remotos. Los clientes podrán consumir el servicio utilizando herramientas de generación de código a partir del archivo WSDL (Web Services Description Language), el cual define de manera automática la estructura del servicio, facilitando así su integración con otros sistemas.

La implementación de este servicio no solo resuelve el problema técnico específico de realizar sumas remotamente, sino que también sirve como base pedagógica para entender la arquitectura de los servicios web. Asimismo, ofrece una plataforma escalable sobre la cual se pueden agregar nuevas funcionalidades (por ejemplo, resta, multiplicación, división) y extender el servicio hacia una calculadora remota más robusta.

Esta propuesta de solución prioriza la modularidad, la facilidad de acceso, la reutilización de componentes y la compatibilidad multiplataforma, características fundamentales para el desarrollo de aplicaciones distribuidas modernas que requieren interoperabilidad entre diversos sistemas tecnológicos.

# Materiales y Métodos Empleados

## Materiales

### Lenguaje de programación: Java

Se utilizó Java por su robustez, portabilidad y por contar con soporte nativo para el desarrollo de servicios web mediante la API JAX-WS.

### Entorno de desarrollo integrado (IDE): NetBeans

Se eligió NetBeans debido a su integración con herramientas para servicios web y su compatibilidad con servidores como GlassFish, lo cual facilita la creación, despliegue y prueba de servicios web SOAP.

### Servidor de aplicaciones: GlassFish

Se utilizó GlassFish como servidor para publicar y ejecutar el servicio web. Este servidor soporta la tecnología JAX-WS y permite la administración sencilla de aplicaciones Java EE.

### API para Servicios Web: JAX-WS (Java API for XML Web Services)

Es la especificación de Java que permite crear y consumir servicios web basados en XML y SOAP. Se usó para anotar y exponer la clase del servicio.

### Navegador Web o Cliente SOAP

Para la prueba del servicio web, se utilizó un navegador web para acceder al WSDL, y herramientas como SoapUI o el propio cliente generado por NetBeans para realizar pruebas de consumo.

## Métodos

### Diseño del servicio

Se definió la operación a implementar: la suma de dos números enteros. Se determinó que el servicio tendría una única operación accesible de forma remota, llamada suma, que recibiría dos parámetros tipo int y retornaría un resultado tipo int.

### Codificación del servicio web

Se implementó una clase Java que representa el servicio. Esta clase se anotó con @WebService para identificarla como un servicio web, y su método suma fue etiquetado con @WebMethod y @WebParam para definir el nombre de la operación y sus parámetros de entrada.

**Publicación del servicio en el servidor**

La clase fue desplegada en el servidor GlassFish mediante el IDE NetBeans, lo que permitió que el servicio estuviera disponible en una URL específica. Automáticamente se generó el archivo WSDL que describe el servicio y puede ser usado por clientes para su consumo.

**Pruebas del servicio**

Se realizaron pruebas funcionales enviando peticiones al servicio con diferentes valores de entrada para comprobar que la suma se realizaba correctamente. Estas pruebas se realizaron tanto desde el entorno de desarrollo como desde herramientas externas como SoapUI.

**Validación del WSDL y del consumo remoto**

Se validó la estructura del archivo WSDL generado y se utilizaron herramientas de generación de clientes para probar el consumo del servicio desde otras aplicaciones, asegurando la interoperabilidad.

# Desarrollo

Para el desarrollo de esta practica, tenemos en cuenta que El servicio web se encuentra desarrollado en el lenguaje Java utilizando la API JAX-WS.

Importamos las anotaciones necesarias para que la clase pueda ser reconocida y publicada como un servicio web.

- @WebService: indica que la clase define un servicio web.
- @WebMethod: señala qué métodos dentro de la clase deben estar expuestos al cliente como operaciones del servicio.
- @WebParam: permite asignar nombres a los parámetros del método expuesto, lo que mejora la legibilidad en el archivo WSDL.

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
```

Se define la clase WebServiceSuma y se anota con @WebService. El atributo serviceName especifica el nombre del servicio, el cual aparecerá en el archivo WSDL generado. Esto facilita su identificación al ser consumido por un cliente.

```
@WebService(serviceName = "WebServiceSuma")
public class WebServiceSuma {
```

Definimos la única operación que ofrece el servicio web.

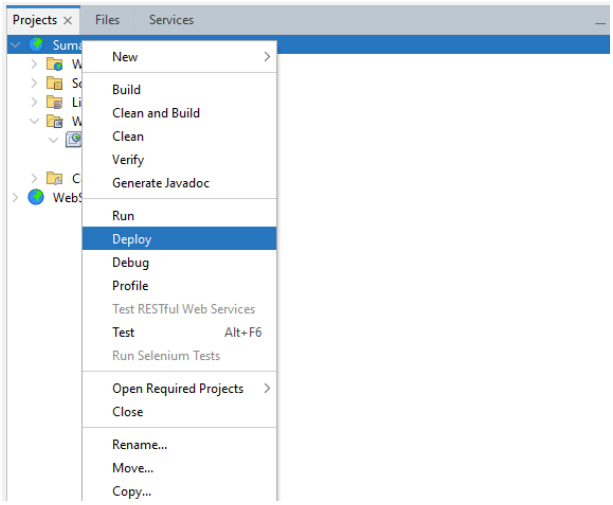
@WebMethod(operationName = "suma"): expone el método como una operación del servicio web y define su nombre accesible. Así, desde el cliente, se podrá invocar la operación suma.

El método suma recibe dos parámetros enteros: numero1 y numero2. Ambos están anotados con @WebParam para indicar los nombres de los parámetros tal como se mostrarán en el archivo WSDL y en las solicitudes SOAP.

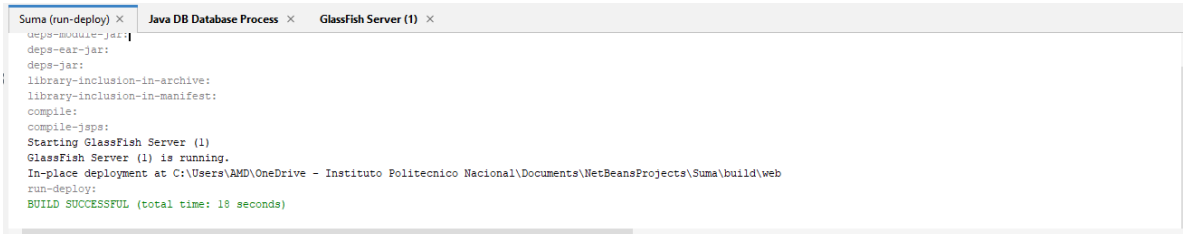
Dentro del cuerpo del método, simplemente se devuelve el resultado de sumar los dos números.

```
@WebMethod(operationName = "suma")
    public int suma(@WebParam(name = "numero1") int numero1,
@WebParam(name = "numero2") int numero2) {
        return numero1 + numero2;
    }
```

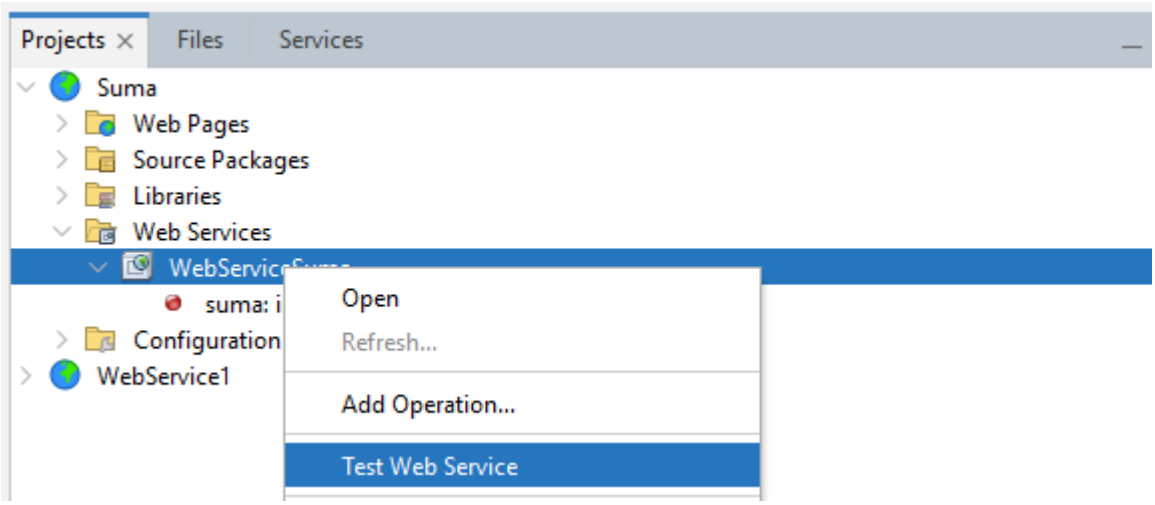
Para ejecutar publicamos nuestra web service dando clic derecho al proyecto y seleccionando deploy.




Para confirmar que se este ejecutando correctamente y que no tengamos problemas con el puerto se muestra el siguiente mensaje

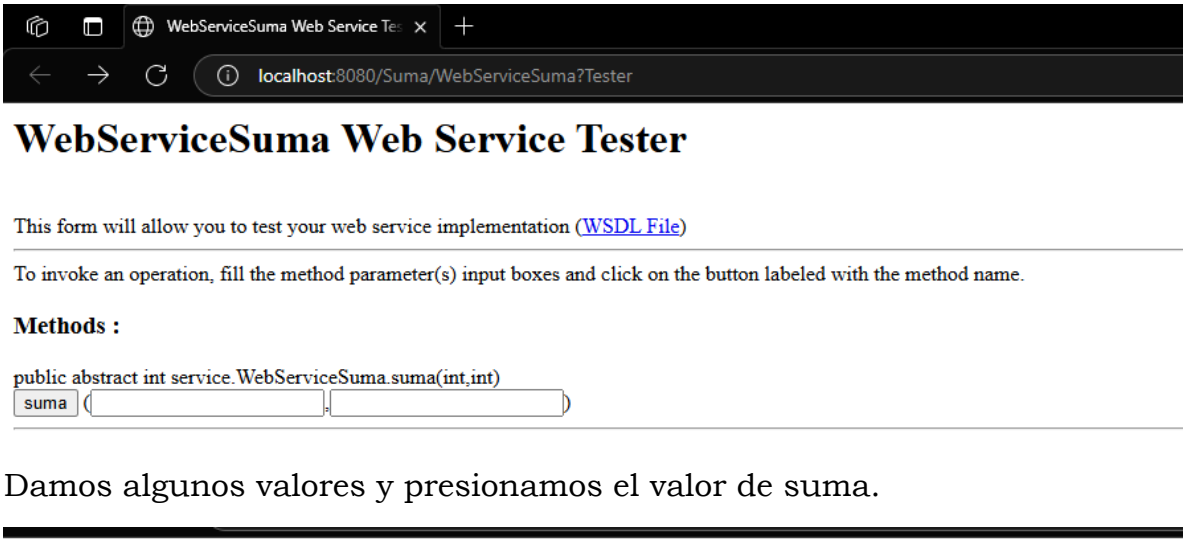



Para ejecutarlo debemos de posicionarnos en la carpeta “Web service”, la desplegamos y en la opción de “Web Service Suma” nuevamente damos clic derecho y seleccionamos la opción de “Test Web Service” la cual nos llevara al navegador para su ejecución.

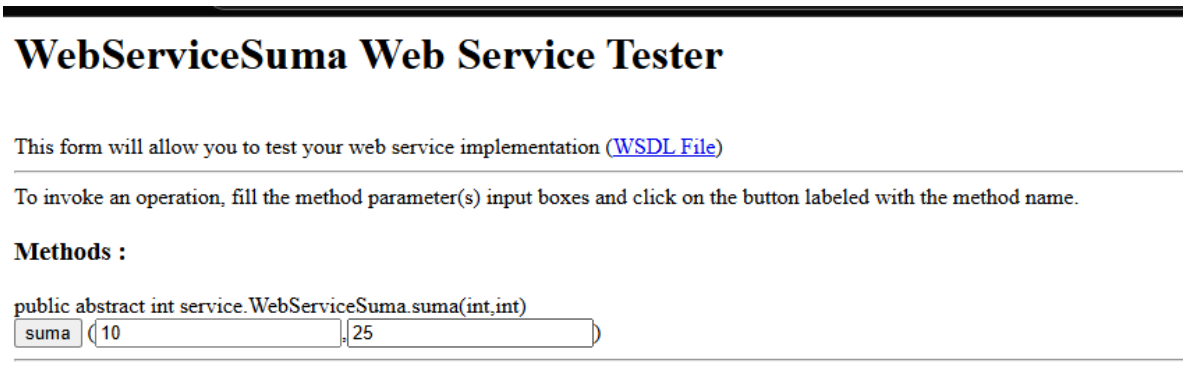


# Resultados

En la ventana del navegador nos muestra el siguiente resultado




Damos algunos valores y presionamos el valor de suma.




Resultado:


Se nos muestra en una tabla los valores que dimos y el resultado además del formato xml de los datos que enviamos con Soap.

# Conclusión

La implementación del servicio web WebServiceSuma demuestra de manera clara y práctica cómo es posible exponer funcionalidades simples, como una operación de suma, mediante tecnologías estándar y ampliamente adoptadas como Java y JAX-WS. A través de este proyecto se logró comprender el proceso completo para la creación de un Web Service, desde la definición de la lógica del servicio hasta su publicación en un servidor de aplicaciones y su posterior consumo por clientes remotos.

Este ejercicio permitió evidenciar las ventajas que ofrecen los servicios web en términos de interoperabilidad, reutilización de código y facilidad de integración entre distintas plataformas y lenguajes de programación. Al emplear el protocolo SOAP y generar automáticamente el archivo WSDL, se facilita la conexión entre sistemas heterogéneos, garantizando que cualquier cliente compatible con este estándar pueda acceder y utilizar el servicio.

Además, este desarrollo sienta las bases para futuros proyectos más complejos, donde se pueden incorporar nuevas operaciones, mejorar la seguridad, y optimizar la comunicación entre sistemas distribuidos. En definitiva, el proyecto no solo resolvió la necesidad funcional de realizar una suma remota, sino que también representó una valiosa experiencia formativa en el diseño e implementación de arquitecturas orientadas a servicios.