MEDAC
Instituto Oficial de Formación Profesional

PROGRAMACIÓN DE SERVICIOS Y PROCESOS
TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

El servicio web

12

ÍNDICE

 ISBN: 978-84-18864-98-8

OBJETIVOS

Conocer qué es un servicio web.

Conocer los estándares más usados por los servicios web.

Conocer la arquitectura SOA.

Crear proyectos que usan servicios web.

Crear clientes que usan los proyectos que usan servicios web.

/ 1. Introducción y contextualización práctica

En esta unidad estudiaremos algunos de los servicios web que vamos a poder implementar utilizando el lenguaje de programación Java.

En primer lugar, vamos a conocer qué es un servicio web, y qué estándares existen para el desarrollo de los mismos.

También vamos a ver qué tipos de arquitectura utilizan estos servicios, la arquitectura orientada a servicios, o más conocida como SOA.

Por último, implementaremos un servicio web SOAP paso a paso.

Escucha el siguiente audio en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad.

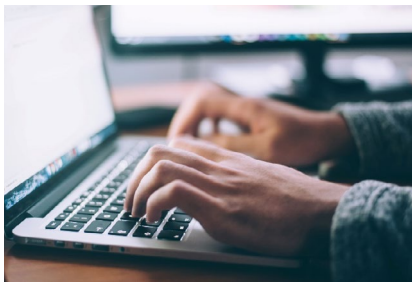

Fig. 1. Navegando por la web.

Audio Intro. "Páginas web en Java"
https://bit.ly/32YnbC1

/ 2. ¿Qué es un servicio web?

Un servicio web, o *web service*, es un "programa" que proporciona una forma de **comunicación entre aplicaciones *software* que se están ejecutando en distintas plataformas**.

Lo forman componentes de aplicaciones distribuidas que deben estar disponibles de forma externa, a los que se podrá acceder mediante una serie de protocolos web estándar, utilizando lenguajes de programación independientes de la plataforma en la que se ejecuten para el intercambio de mensajes.

Los servicios web tienen las siguientes características:

- **Se debe poder acceder a ellos a través de la web**. Para esto se debe utilizar el protocolo HTTP, y el lenguaje XML.
- Al tratarse de un estándar de representación de datos independiente de plataformas, el hecho de utilizar XML como formato de intercambio de datos entre el servicio Web y el cliente permite la **comunicación entre ambos.**
- Todos los servicios web realizan una serie de **funciones bien definidas**. Deben de contener una descripción de ellos mismos, de forma que una aplicación conozca fácilmente cuál es la función del servicio web.
- **Han de poder localizarse**. Debe existir algún mecanismo que permita encontrar un servicio web que realice una determinada función.
- Son componentes independientes que se pueden integrar, formando así **sistemas distribuidos complejos**.
- **Ofrecen interoperabilidad**. El hecho de usar HTTP y XML hace que sea posible que distintas aplicaciones puedan utilizar los servicios web.

Estas son las características más importantes que podemos destacar de los servicios web, pero te invitamos a que investigues por la red para descubrir algunas más.

Fig. 2. Arquitectura de los Servicios Web SOAP.
Fuente: https://es.wikipedia.org/wiki/Servicio_web

/ 3. Estándares y especificaciones web

A continuación, vamos a conocer cuáles son algunas de las **organizaciones que definen los estándares** que han de seguir los **servicios web**.

- **W3C**: (Consorcio para la *World Wide Web*). Esta organización fue fundada en octubre de 1994 con el objetivo de llevar a Internet a su máximo potencial. Para ello desarrollaron protocolos de uso común que han marcado la evolución y han asegurado la interoperabilidad de internet. Algunos de estos protocolos son: HTML, HTTP, XML, SOAP, WS, entre otros.

  Para saber más puedes visitar su página web: https://www.w3.org/standards/webofservices/description.html

- **OASIS**: Esta organización es un consorcio sin fines de lucro que se dedica a impulsar el desarrollo, la convergencia y la adopción de los estándares abiertos para la sociedad de la información global. OASIS siempre ha promovido el consenso de la industria y ha producido una serie de normas internacionales para la arquitectura orientada a servicios (SOA, *Service Orientated Architecture*), las funciones y calidad de servicios web, la seguridad de la web, el *cloud computing*, la publicación electrónica de documentos, etc. Los estándares OASIS abiertos han permitido reducir costes, fomentar la innovación y proteger el derecho de libre elección de la tecnología, entre otras muchas acciones.

- **JPC**: Esta organización trabaja con cada versión de JAVA, incluyendo un conjunto de especificaciones de diferentes tecnologías, definidas por este mismo organismo, y que son nombradas como JSR (*Java Specification Request*, petición de especificación de Java), seguidas de un número. Algunos ejemplos de ellas son:

  - **JSR 370**: define la API JAX-RS 2.1.
  - **JSR 224**: define la API JAX-WS 2. 0 para servicios web basados en XML.
  - **JSR 342**: define las especificaciones para java EE 7.
  - **JSR 366**: define las especificaciones para java EE 8.

Estos son solo algunos ejemplos de los principales estándares que se utilizan en los servicios web, pero pueden existir cientos de ellos...

Audio 1. "Servicios REST"
https://bit.ly/2IJpGl4

/ 4. Caso práctico 1: "Conversor de medidas"

Planteamiento: Pilar y José están realizando un ejercicio sobre la implementación de servicios SOAP.

Este ejercicio consiste en convertir una cierta cantidad de centímetros a pulgadas.

Para ello deberán realizar un servicio SOAP donde el cliente podrá indicar la cantidad de centímetros a convertir a pulgadas. También deberán realizar todas las comprobaciones oportunas para garantizar que el servicio SOAP no falle.

Nudo: ¿Cómo crees que debe afrontarse inicialmente el ejercicio?¿Qué comprobaciones crees que deberían realizar nuestros amigos para que el servicio SOAP no falle?

Desenlace. En este ejercicio nos centraremos en la fase de análisis del problema, que nos servirá para saber cómo enfocar posteriormente el desarrollo del servicio SOAP.

La conversión de centímetros a pulgadas puede realizarse con una fórmula muy simple, ya que basta con conocer cuántas pulgadas equivalen a un centímetro: concretamente 1 centímetro es igual a 0,393701 pulgadas.

Con respecto a las comprobaciones previas al ejercicio podemos cerciorarnos que el dato que haya enviado el usuario no sea un campo vacío, ya que esto podría ocasionar una excepción en el servidor, con el respectivo error. Otro de los aspectos que podríamos comprobar es que el dato que envíe el cliente para convertir sean números, ya que si envía una letra o palabra se producirá una excepción en el cálculo.

Por último, podríamos comprobar que la cantidad enviada no fuese negativa, que, aunque no es tan importante como las comprobaciones anteriores, no tiene mucho sentido una distancia negativa.

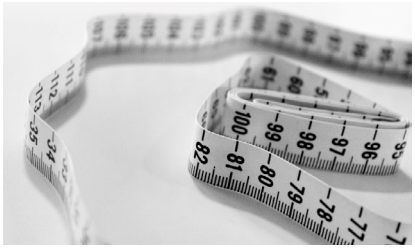

Fig. 3. Centímetros.

/ 5. Arquitectura orientada a servicios: SOA

La arquitectura SOA es una **tecnología** que nos permite el **diseño de aplicaciones basadas en peticiones a un servicio**. De esta forma podemos crear pequeños **elementos *software* reutilizables** y, además, independientes del lenguaje con el que fueron creados.

Esto ha servido para dar lugar a un nuevo tipo de programación, la llamada ***Software as a Service (software como servicio o SaaS)***. Este tipo de programación se basa en que las aplicaciones no se diseñan para ser instaladas en el ordenador del cliente (como ocurre en la programación de aplicaciones clásica), sino que se instalan en un servidor al que los clientes realizan una serie de peticiones. De este modo, tenemos un servicio que está disponible desde cualquier punto del planeta, siempre que se pueda tener acceso a Internet.

Los servicios web son el punto fuerte de las aplicaciones SOA, debido a que la tecnología SOA trabaja con un conjunto de características que hacen posible que se ejecuten los servicios web a la perfección. Existen dos tipos de arquitecturas orientadas a servicios:

1. **Arquitectura Orientada a Servicios Tradicional (SOA tradicional)**. Este tipo de arquitectura utiliza los principios y las tecnologías básicas de los servicios web, pudiendo utilizar SOAP como lenguaje de intercambio de datos, WSDL como lenguaje para la descripción de los servicios utilizados y UDDI para la publicación.

   Estas arquitecturas son muy utilizadas, pero no incluyen características tan básicas como pueden ser: seguridad, transaccionalidad, garantía de entrega, direccionamiento, entre otras.

2. **Arquitectura Orientada a Servicios de segunda generación**. Un servido SOA de segunda generación consigue ampliar la funcionalidad de los anteriores. Para ello, se añaden nuevas características que buscan mejorar la calidad del servicio según los estándares WS (Servicios Web) aprobados por OASIS. Algunas de estas características que se mejoran son: política de seguridad, transacción, gestión...

En la actualidad, se reciben actualizaciones frecuentes en las características de los servicios SOA, esto es debido a que constantemente están saliendo a la luz nuevos fallos de seguridad, nuevas formas de realizar transacciones de datos, etc.

Los servicios SOA, según la fuente, nos los podremos encontrar también como SOAP (*Simple Objet Access Protocol*), y no debemos confundirlos. **Podríamos decir que SOA es el modelo de la arquitectura, y SOAP, una forma de comunicación (protocolo) que se permite en SOA**.

Vídeo 1. "Aplicaciones distribuidas"
https://bit.ly/3kLSCpa

/ 6. Creación de servicios web SOAP I

Vamos a ver paso a paso cómo podemos crear un **servicio web SOAP**, que nos permita **calcular el volumen de una esfera**.

Una vez hecho esto crearemos un cliente web que haciendo uso del servicio anterior sea capaz de calcular dicho volumen.

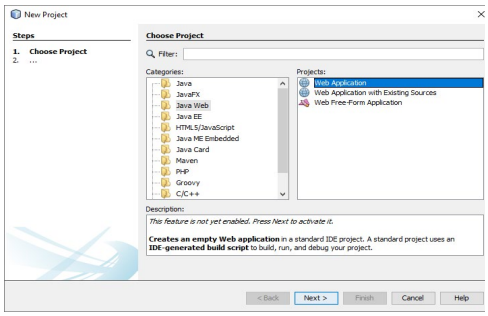

Fig. 4. Creando un servicio web en Java.

- Lo primero es crear una aplicación web en Java, para ello seleccionamos ***Java Web y Web Application***.
- En la siguiente pantalla deberemos indicar el nombre del proyecto, en este ejemplo los llamaremos ***WebServiceEsfera***.
- Aceptados y nos aparecerá la pantalla de configuración de ***GlassFish***, que será el servidor que vamos a utilizar para nuestros proyectos con NetBeans. Este viene integrado por defecto en la versión completa de NetBeans 8.2. Dejamos la configuración de *GlassFish* como está por defecto.
- Una vez creado el proyecto, vamos a crear un paquete nuevo, para ello pulsamos nuevo y paquete.

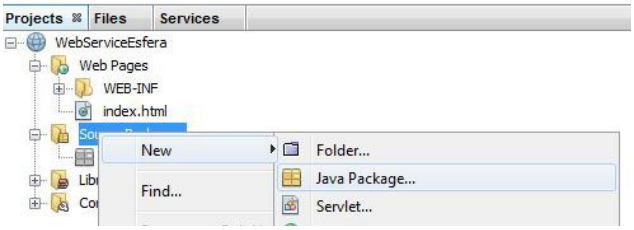

Fig. 5. Creando un paquete.

- Llamaremos al paquete **esfera.com**.
- El siguiente paso será el de agregar un nuevo *web service*, para ello pinchamos en *Nuevo -> web service*, y lo llamamos ***CalcularVolumenEsfera***.

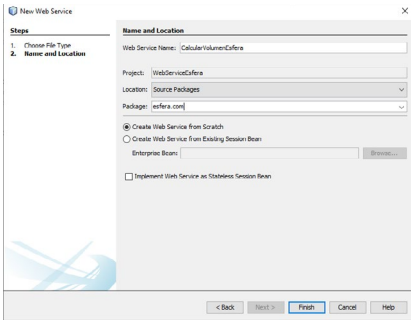

Fig. 6. Creando un nuevo web service.

- Ahora dentro de nuestro paquete tendremos un fichero llamado *CalcularVolumenEsfera.java* con un método ***hello***. Borramos el método *hello* y escribimos el método que nos permitirá calcular el volumen de la esfera.

```
1  @WebMethod(operationName = "volumen")
2  public Double calcularVolumen(@WebParam(name = "radio") Double radio)
3  {
4      return (4.0/3.0)*Math.PI * Math.pow(radio, 3);
5  }
```

Fig. 7. Método para calcular el volumen de la esfera.

- Si nos fijamos en la carpeta *Web Services*, el servicio que creamos previamente se nos ha actualizado y ahora mostrará el ***web service calcularVolumen***.

- El siguiente paso sería lanzar el proyecto, para ello, pulsamos el botón de **ejecutar** y se iniciarán todos los servicios necesarios (la primera vez suele tardar un poco y además, puede que el firewall lance algún tipo de aviso, si esto ocurre, habría que permitir el acceso de la aplicación).

  Si hemos seguido todas las indicaciones podremos acceder a nuestro servicio mediante la dirección: http://localhost:8080/WebServiceEsfera/

- Ahora podremos modificar el fichero HTML como deseemos para darle la interfaz que más nos guste, pudiendo utilizar CSS para ello.

/ 7. Caso práctico 2: "Calculadora SOAP"

Planteamiento: Pilar y José se disponen a realizar el último ejercicio de la relación de esta unidad. Este consiste en realizar un servicio web SOAP que permita ejecutar la funcionalidad de una calculadora en el servidor, pudiendo realizar las operaciones de suma, resta, multiplicación y división. El cliente podrá elegir qué operación realizar, pasándole además los datos con los que operar.

Pilar y José se ponen a la obra...

Nudo: ¿Cómo pueden realizar las operaciones del servicio SOAP? ¿Cómo crees que pueden indicar por parte del cliente qué operación realizar?

Desenlace: Siempre que tengamos que pensar cómo vamos a resolver un problema en programación, debemos ser conscientes de que existen muchas formas de resolverlo, y este no es una excepción.

Como tenemos que poder elegir la opción en la parte del cliente, es decir, en la parte donde vamos a interactuar, ya que ésta está programada mediante HTML, podremos crear, por ejemplo, una lista con las posibles opciones de la calculadora. Otra forma de resolver el problema podría ser mediante unos botones de tipo radio, en los que podamos elegir únicamente una opción.

Con respecto a la implementación SOAP tenemos que tener en cuenta que hemos de distinguir claramente qué operación queremos realizar, la cual, habría sido elegida en la parte descrita anteriormente, pudiéndolo hacer fácilmente con un *switch*. Observa en la siguiente figura cómo se ha implementado para la operación 'suma', siendo la forma de resolverlo exactamente igual para el resto de operaciones.

```
@WebMethod(operationName = "operacion")
public Double calcularOperacion(
        @WebParam(name = "operacion") String operacion,
        @WebParam(name = "dato1") Double dato1,
        @WebParam(name = "dato2") Double dato2) {
    Double resultado = 0;

    switch(peracion)
    {
        case "suma":
            resultado = dato1 + dato2;
            break;
        // Demás casos de igual forma
    }

    return resultado;
}
```

Fig. 8. Método para el servicio de la calculadora SOAP.

/ 8. Creación de servicios web SOAP II

Con el servicio ya implementado, crearemos otro proyecto Java web que nos servirá de cliente de nuestro servicio, el cual llamaremos **ClienteEsfera**.

- Para ello, creamos un ***Web Service Client***, pinchando en *Nuevo -> web service* client. Accedemos a su configuración y seleccionamos el proyecto de calcular el volumen de la esfera.

- Una vez realizado esto vamos a crear un **jsp** para que se ejecute nuestro cliente: pulsamos en *web pages -> nuevo -> jsp*. Lo llamaremos ***index***. Por ahora en el proyecto disponemos de dos *index*, un html y un jsp. Procederemos a crear nuestro cliente en el jsp, que nos permitirá escribir código Java, no obstante, si ejecutamos el cliente tal cual está ahora, se ejecutará el html. Para cambiarlo, accederemos a las propiedades del proyecto cliente y seleccionaremos ***run***. Una vez aquí, observamos una opción denominada *Relative URL*, donde escribiremos */index.jsp* para que, al ejecutarse, lance el fichero index.jsp.

- A continuación, haciendo uso de la paleta de elementos disponibles crearemos un formulario con un *input* para introducir el radio y un botón para realizar los cálculos.

- El siguiente paso sería escribir el código Java de nuestro cliente. Para ello, nos dirigiremos al proyecto cliente, a la carpeta *Web Services References -> CalcularVolumenEsfera -> **CalcularVolumenEsferaPort***, donde observaremos un punto rojo en el servicio de calcular el volumen, lo seleccionamos y lo arrastramos dentro de nuestro jsp, concretamente debajo del formulario y dentro del *body*. Para terminar, habría que completar el código tal y como está implementado en el fichero index.jsp que podrás encontrar en la sección de Recursos del tema.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <div>Volumen de la una esfera</div>
        <form action="index.jsp" method="POST">
            <input type="text" name="radio" value="1" />
            <input type="submit" value="Calcular" />
        </form>
    </body>
</html>
```

Fig. 9. Código HTML.

- Finalmente, compilamos nuestro proyecto servidor, y posteriormente el cliente, y ya podremos ejecutar nuestro **servicio web**.

Vídeo 2. "Ejemplo SOAP completo"
https://bit.ly/3nEFGmW

/ 9. Resumen y resolución del caso práctico de la unidad

A lo largo de esta unidad hemos visto algunos de los **servicios web** que se pueden implementar utilizando el lenguaje de programación Java. En primer lugar, hemos contextualizado qué es un servicio web y hemos expuesto qué estándares existen para el desarrollo de los mismos, además de quién los define. También hemos estudiado qué tipo de arquitectura utilizan estos servicios. Esta es la conocida como la **arquitectura orientada a servicios**, también llamada por sus siglas, SOA. Por último, hemos ejecutado paso a paso la creación de un **servicio web SOAP,** que nos ha servido de ejemplo de cómo se pueden crear estos servicios, para que, a partir de lo aprendido con éste, podamos ir creando servicios cada vez más sofisticados.

Resolución del caso práctico de la unidad

Cuando estamos acostumbrados a desarrollar programas de escritorio en Java es normal que nos sorprenda la idea de que con Java se puedan crear páginas web, aunque sería más correcto indicar que **son servicios web**, no páginas web en sí.

Hay multitud de lenguajes de programación que nos permiten crear páginas y servicios web, como pueden ser *Python* (que ya conocemos de Sistemas de Gestión Empresarial) con el *framework* Django, o incluso el ya citado y más que conocido por nuestra parte Java, que nos va a permitir crear servicios web que se implementarán en un servidor y que los clientes podrán consumir. Aunque seguro que lo sabes, no obstante, cabe destacar que no hay que confundir Java con Javascript, ya que son lenguajes diferentes, y se parecen únicamente en su nombre.

Fig. 10. Consumiendo un servicio Java.

/ 10. Bibliografía

Colaboradores de Wikipedia. (2020a, julio 1). Arquitectura orientada a servicios. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

Colaboradores de Wikipedia. (2020, 19 agosto). Servicio web. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Servicio_web

PowerData, R. (2014, 8 septiembre). Qué es la arquitectura orientada a servicios SOA. PowerData. https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/394442/qu-es-la-arquitectura-orientada-a-servicios-soa

Luis Enrique Juarez. (2013, 14 junio). Web Services usando Java y Netbeans. YouTube. https://www.youtube.com/watch?v=BPk-vup0_0M