Contenidos a Trabajar

1. Servicios web: interacción entre máquinas
2. Funcionamiento y Objetivo de los Servicios Web
3. CORS - Control de acceso HTTP

Servicios web: interacción entre máquinas

Hoy en día es común utilizar diferentes servicios a través de la web. Comprar en línea, leer el diario, reservar una mesa en un restaurante o ver películas son solo algunos ejemplos de las muchas interacciones diarias entre el usuario y la máquina. Pero además, aunque no lo notemos, estas interacciones también ocurren entre máquinas. El cliente y el servidor se envían constantemente solicitudes y respuestas gracias a los servicios web.

¿Qué es un servicio web?

Un servicio web es una forma de ofrecer servicios a través de Internet. Es una interfaz que permite la comunicación entre dos máquinas o aplicaciones. Esta tecnología se caracteriza por dos aspectos:

- **Multiplataforma:** no es necesario que el cliente y el servidor tengan la misma configuración para comunicarse. El servicio web se encarga de hacerlo posible.
- **Distribuido:** generalmente, un servicio web no está disponible solo para un cliente, sino que varios clientes pueden acceder a él a través de Internet.

Cuando se utiliza un servicio web, el cliente envía una solicitud al servidor, lo que desencadena una acción por parte de este. Luego, el servidor envía una respuesta al cliente.

La tecnología detrás de un servicio web: un ejemplo

Todos los servicios web tienen un Identificador Uniforme de Recursos (URI), que es la dirección del servicio web. Es similar a un Localizador Uniforme de Recursos (URL) que se utiliza para acceder a páginas web. El catálogo UDDI también debería haber sido importante, ya que permitía encontrar servicios web, pero nunca tuvo éxito y sus principales defensores se retiraron del proyecto.

El lenguaje Web Service Description Language (WSDL) es el más relevante. Un servicio web contiene un archivo WSDL que describe detalladamente el servicio. Con esta información, el cliente puede comprender qué funciones puede realizar en el servidor a través del servicio web. La comunicación se realiza mediante diferentes protocolos y arquitecturas. Entre ellos, los más populares son el protocolo de red SOAP en combinación con el estándar de Internet HTTP, o los servicios web basados en una arquitectura REST.

Estas tecnologías permiten el intercambio de solicitudes y respuestas, a menudo utilizando el lenguaje de marcado extensible (XML). Este lenguaje, relativamente sencillo, puede ser interpretado tanto por personas como por computadoras, y

es adecuado para conectar sistemas con requisitos diferentes. Sin embargo, REST también admite otros formatos, como JSON.

Veamos cómo funciona esta tecnología con un ejemplo de servicio web:

Supongamos que tenemos un programa escrito en Visual Basic que se ejecuta en una computadora con Windows. El programa necesita acceder al servicio de un servidor web Apache. Para ello, el cliente envía una solicitud SOAP en forma de mensaje HTTP al servidor. El servicio web interpreta el contenido de la solicitud y asegura que el servidor realice una acción. Finalmente, el servicio web envía una respuesta al cliente (nuevamente con SOAP y HTTP), que vuelve a interpretarla. La información se envía al programa, donde será procesada.

Ventajas y desventajas de los servicios web

La gran ventaja de los servicios web es que la comunicación no depende de una plataforma específica, por lo que el cliente y el servidor no necesitan tener muchos puntos en común para comunicarse. Esto se logra utilizando formatos estandarizados que todos los sistemas pueden interpretar.

Sin embargo, aquí es donde encontramos una desventaja. El formato XML, en particular, es bastante pesado y genera grandes paquetes de datos, lo que puede causar problemas en conexiones de red lentas. Otra opción para conectar dos sistemas a través de Internet son las **API web**. Generalmente, son más

rápidas, pero requieren que el cliente y el servidor cumplan con especificaciones más concretas, lo que limita la interoperabilidad.

Funcionamiento y Objetivo de los Servicios Web

Introducción

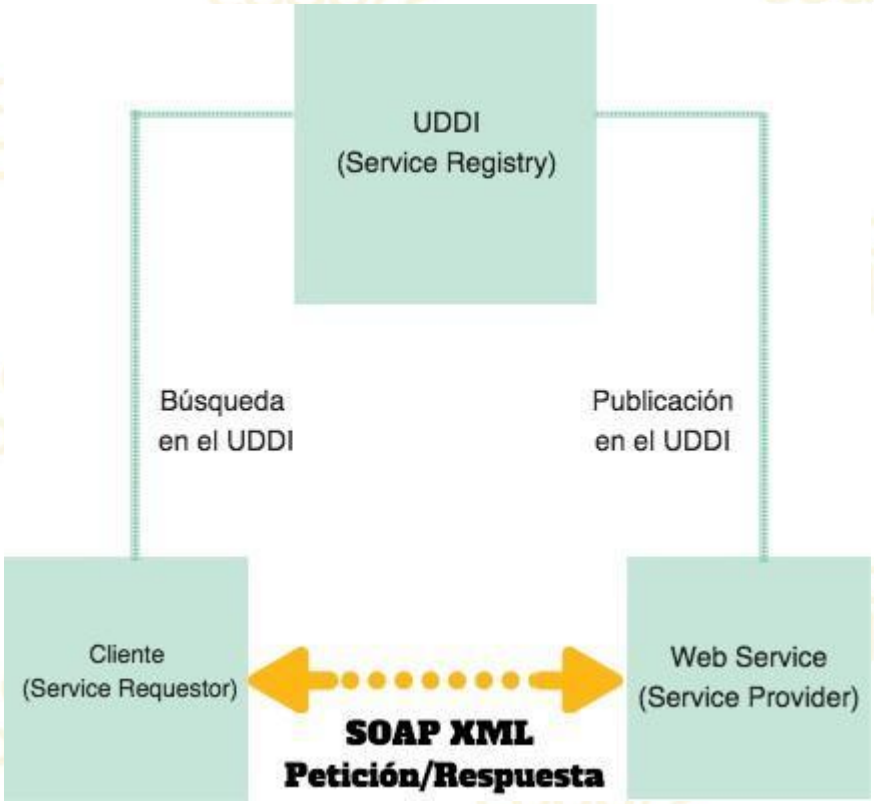
Como vimos anteriormente, un **Servicio Web**, también conocido como ***Web Service***, es una forma de comunicación entre dos dispositivos electrónicos en una red. Se basa en una colección de protocolos abiertos y estándares utilizados para intercambiar datos entre aplicaciones o sistemas.

Las aplicaciones escritas en diferentes lenguajes de programación y que se ejecutan en plataformas distintas pueden utilizar servicios web para intercambiar información a través de una red. La clave de esta interoperabilidad, por ejemplo, entre Java y Python o entre Windows y Linux, radica en el uso de estándares abiertos.

Como sistema de mensajería se utiliza XML, un lenguaje estandarizado. El protocolo más sencillo para el intercambio de información entre computadoras es XML-RPC, que emplea XML para realizar llamadas a procedimientos remotos (RPC). RPC, Remote Procedure Call, es un protocolo de red que permite que un programa ejecute código en una máquina remota. Las solicitudes XML-RPC son una combinación de contenido XML y encabezados HTTP. La simplicidad de XML-RPC llevó a la evolución del estándar SOAP, que es uno de los componentes fundamentales de los Servicios Web.

La base de la comunicación entre servicios web es, por lo tanto, el lenguaje XML y el protocolo HTTP.

Componentes de los Servicios Web



Los servicios web estandarizados funcionan utilizando los siguientes componentes:

SOAP - Simple Object Access Protocol: es un protocolo escrito en XML para el intercambio de información entre aplicaciones. Es un formato diseñado específicamente para la comunicación en Internet y puede extender los encabezados HTTP. Define cómo se envían los mensajes y qué información se

incluye, todo mediante XML. Básicamente, es un protocolo para acceder a un servicio web.

WSDL - Web Services Description Language: es un lenguaje basado en XML utilizado para describir los servicios web y cómo acceder a ellos. Es el formato estándar para describir un servicio web y fue diseñado por Microsoft e IBM. WSDL es una parte integral del estándar UDDI y es el lenguaje que este último utiliza.

UDDI - Universal Description, Discovery and Integration: es un estándar XML utilizado para describir, publicar y encontrar servicios web. Es un directorio donde las empresas pueden registrar y buscar servicios web. Es un directorio de interfaces de servicios web descritos en WSDL y que se comunican mediante SOAP.

Arquitectura de los Servicios Web

Service Discovery: es responsable de centralizar los servicios web en un directorio de registro común y proporcionar una funcionalidad sencilla para publicar y buscar servicios. UDDI se encarga del descubrimiento de servicios.

Service Description: uno de los aspectos más distintivos de los servicios web es que se autodescriben. Esto significa que una vez que se ha localizado un servicio web, proporcionará información sobre las operaciones que admite y cómo activarlo. Esto se realiza mediante el lenguaje de descripción de servicios web (WSDL).

Service Invocation: invocar un servicio web implica enviar mensajes entre el cliente y el servidor. SOAP (Simple Object Access Protocol) especifica cómo deben formatearse los mensajes de solicitud hacia el servidor y cómo el servidor debe formatear sus mensajes de respuesta.

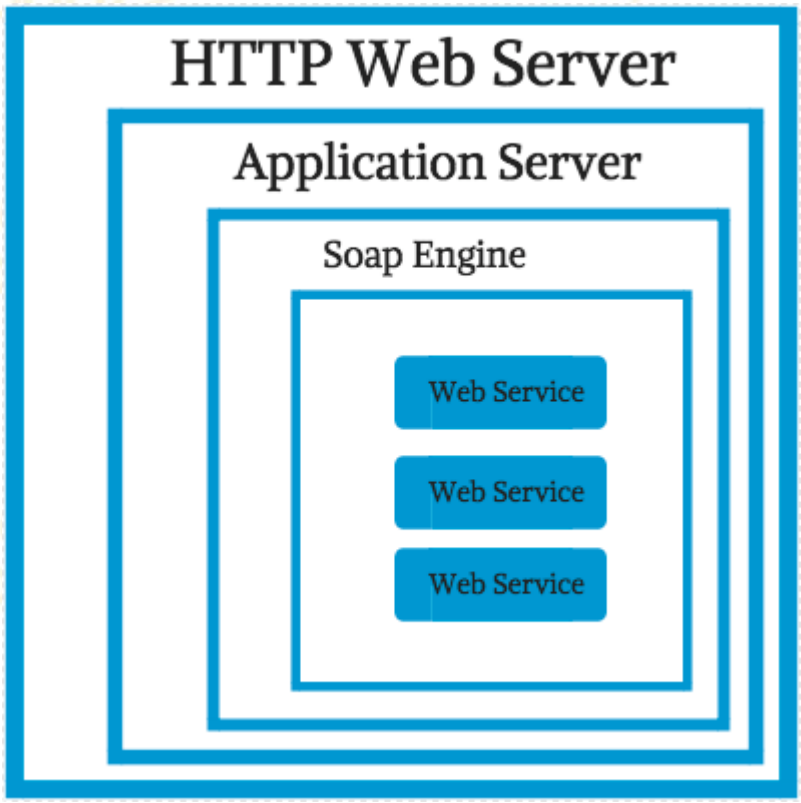
Transport: todos estos mensajes deben ser transmitidos de alguna manera entre el servidor y el cliente. El protocolo elegido para esto es HTTP (HyperText Transfer Protocol). Se pueden utilizar otros protocolos, pero HTTP es el más utilizado en la actualidad.

Cómo funciona un Servicio Web

1. El **Service Provider** genera el WSDL que describe el servicio web y lo registra en el directorio UDDI o en el **Service Registry**.
2. El **Service Requestor** o la aplicación cliente requiere un servicio web y se pone en contacto con UDDI para localizarlo.
3. Basándose en la descripción proporcionada por el WSDL, el **Cliente** envía una solicitud para un servicio específico al Listener del servicio web, que se encarga de recibir y enviar los mensajes en formato SOAP.
4. El **Web Service** analiza el mensaje SOAP de la solicitud e invoca una operación específica en la aplicación para procesar la solicitud. El

resultado se escribe nuevamente en SOAP como una respuesta y se envía al cliente.

5. El **Cliente** analiza el mensaje de respuesta SOAP y lo interpreta o genera un error si corresponde.



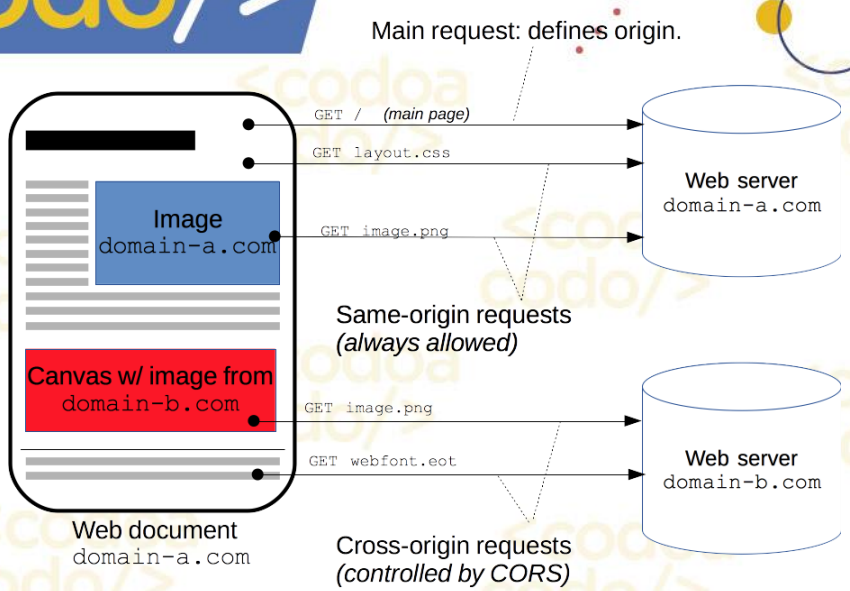
CORS - Control de acceso HTTP

El Control de Acceso HTTP (CORS) es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un navegador web obtenga permiso para acceder a recursos específicos en un servidor que se encuentra en un origen (dominio) diferente al del propio documento.

Esto ocurre cuando un **user agent** (como un navegador) realiza una solicitud a un recurso en un dominio, protocolo o puerto diferente al del documento actual.

Un ejemplo de una solicitud de origen cruzado es cuando el código JavaScript de una aplicación web alojada en **http://dominio-a.com** utiliza **XMLHttpRequest** para cargar el recurso **http://api.dominio-b.com/data.json**

Por razones de seguridad, los navegadores restringen las **solicitudes HTTP** de origen cruzado iniciadas por scripts. Por ejemplo, las APIs **XMLHttpRequest** y **Fetch** siguen la política de "mismo origen". Esto significa que una aplicación que utiliza estas APIs solo puede hacer solicitudes HTTP a su propio dominio, a menos que se utilicen cabeceras CORS.



La **W3C** recomienda el uso del mecanismo de Control de Acceso HTTP de Origen Cruzado (CORS) ya que proporciona controles de acceso a dominios cruzados para servidores web y permite la transferencia segura de datos entre navegadores y servidores web en diferentes dominios. Los navegadores modernos utilizan CORS dentro de APIs como XMLHttpRequest o Fetch para mitigar los riesgos de las solicitudes HTTP de origen cruzado.

¿Qué peticiones utiliza CORS?

El estándar de intercambio de origen cruzado (CORS) se utiliza para habilitar las siguientes solicitudes HTTP de sitios cruzados:

1. Invocaciones de las APIs XMLHttpRequest o Fetch en un contexto de sitio cruzado.
2. Fuentes web (utilizadas en dominios cruzados con la regla @font-face dentro de CSS), permitiendo que los servidores muestren fuentes TrueType que solo pueden ser cargadas por sitios cruzados y utilizadas por sitios web autorizados.
3. Texturas en WebGL.
4. Imágenes dibujadas en patrones utilizando drawImage.
5. Acceso a hojas de estilo (CSSOM).
6. Ejecución de scripts (para excepciones inmutables).