educación IT
CODE YOUR FUTURE

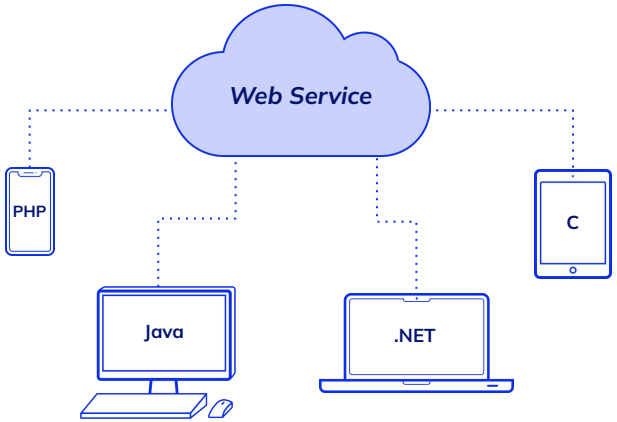
Introducción a Servicios Web

¿Qué es un servicio web?

Un *web service* es un **conjunto de protocolos y estándares** que permiten la **comunicación entre aplicaciones a través de la web (Internet).**

Los servicios web facilitan la **interoperabilidad entre diferentes sistemas y plataformas.**

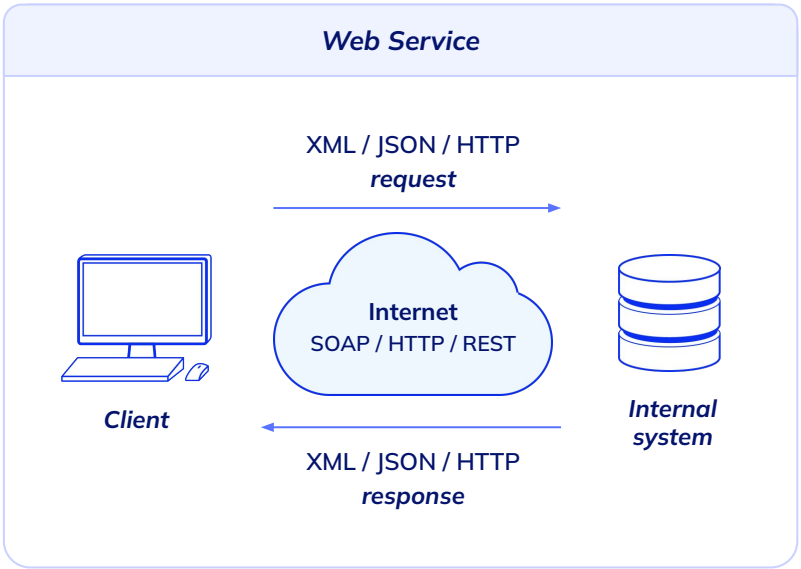
Significa que aplicaciones desarrolladas, en diferentes lenguajes de programación y en diferentes sistemas operativos, pueden **interactuar, entre sí,** sin problemas.



Características

Además de la **interoperabilidad,** los servicios web se distinguen por:

- **Uso de estándares**: Implementan estándares web como HTTP, XML, JSON, SOAP y REST.
- **Comunicación a través de la web**: Funcionan sobre protocolos de red estándar como HTTP o HTTPS.
- **Accesibilidad**: Pueden ser invocados a través de la red. Esto permite el acceso remoto a servicios y datos.



Tipos de servicios web

Simple Object Access Protocol (SOAP)

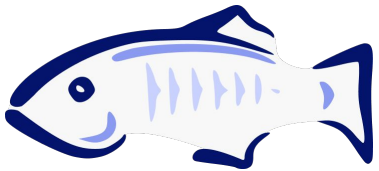
- Utiliza **XML** para el formato de sus mensajes.
- Es un protocolo **basado en estándares** que permite el intercambio de información estructurada en una **plataforma distribuida y descentralizada.**
- Incluye **reglas estrictas** sobre cómo debe estructurarse el mensaje y puede **utilizar varios protocolos de transporte** como HTTP, SMTP, y otros.

Representational State Transfer (REST)

- Es un estilo arquitectónico para **diseñar servicios de red escalables.**
- Utiliza HTTP y **se basa en recursos,** donde cada recurso es identificado por una **URL.**
- Los **métodos HTTP *(GET, POST, PUT, DELETE)*** se utilizan para realizar operaciones sobre estos recursos.

JAX-RS: La API de Java para servicios web *RESTful*

Java API for RESTful Web Services es una **especificación para la creación de servicios web basados en el estilo arquitectónico *Representational State Transfer (REST).***



Características

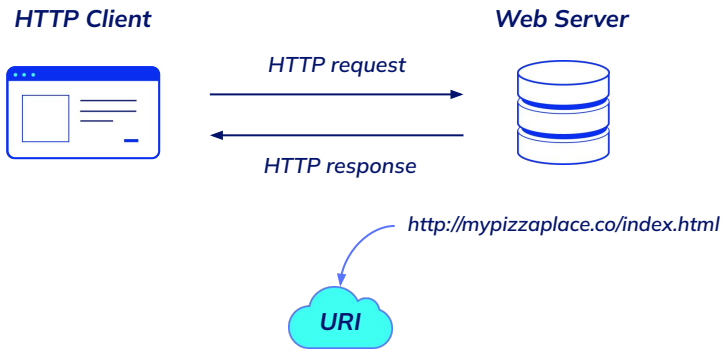
- Utiliza anotaciones, introducidas en **Java SE 5,** para simplificar el desarrollo y despliegue de los clientes y puntos finales de los servicios web. A partir de la versión 1.1 en adelante, **JAX-RS** es parte oficial de **Java EE 6.**
- Cuenta con una implementación de referencia llamada **Jersey.**

Se puede obtener la especificación completa en: *Java Community Process*.

Recursos

Un concepto clave en REST es la **existencia de recursos** (elementos de información), que pueden ser **accedidos utilizando un identificador global** (Identificador Uniforme de Recurso - **URI**).

Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una **interfaz estándar (HTTP)** e intercambian representaciones de estos recursos.



Conectores y comunicación *stateless* en REST

La petición puede ser transmitida por **cualquier número de conectores** (por ejemplo clientes, servidores, cachés, túneles, y otros) pero cada uno lo hace **sin "ver más allá" de su propia petición** (lo que se conoce como stateless o sin estado), otra **restricción de REST**, que es un principio común con muchas otras partes de la arquitectura de redes y de la información).

Así, **una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida**, no necesita conocer si existen cachés, *proxys,* cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información.

La aplicación, sin embargo, **debe comprender el formato** de la información devuelta (la representación), que es por lo general **un documento HTML o XML**, aunque también puede ser una imagen o cualquier otro contenido.

RESTful

RESTful es un término utilizado para describir **servicios web que implementan el estilo arquitectónico *REST*.**

Cuando un servicio *REST* posee ciertas características, se considera *RESTful.*

Características

- **Recursos y URIs:** En REST, todo es considerado un recurso. Cada recurso es identificado por una URI.

  Por ejemplo, en un servicio de biblioteca, un recurso podría ser un libro, identificado por `/books/{id}`.

- **Métodos HTTP:** REST utiliza los métodos estándar de HTTP para realizar operaciones sobre los recursos. Los métodos principales son:
  - **`GET`**: Recuperar una representación de un recurso.
  - **`POST`**: Crear un nuevo recurso.
  - **`PUT`**: Actualizar un recurso existente.
  - **`DELETE`**: Eliminar un recurso.
  - **`PATCH`**: Aplicar modificaciones parciales a un recurso.

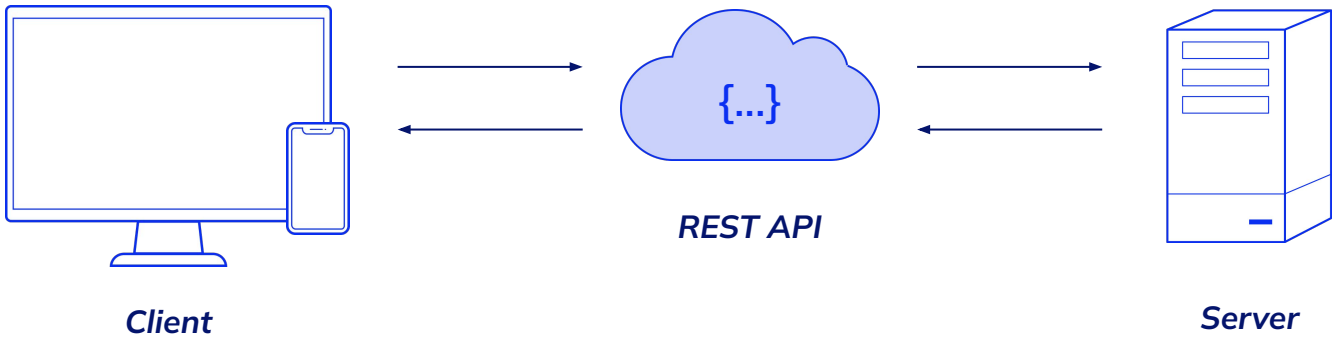
- **Representaciones:** Los recursos se representan en formatos estándar, como JSON, XML, HTML, y otros.

- ***Stateless* (sin estado):** Cada solicitud del cliente al servidor debe contener toda la información necesaria para entender y procesar la solicitud.

- ***Cacheable:*** Las respuestas deben definir si pueden ser almacenadas en caché o no. Las respuestas cacheables permiten mejorar la eficiencia de la red y la escalabilidad del servicio.

- **Interfaz uniforme:** REST se basa en una interfaz uniforme para que diferentes componentes del sistema puedan comunicarse de manera sencilla y coherente.

Representación esquemática:



Ejemplo: Gestión de recursos de libros en la API

Detalle de las **operaciones CRUD** para gestionar libros en la API. Incluye obtención, creación, actualización y eliminación de libros mediante **métodos HTTP como `GET`, `POST`, `PUT` y `DELETE`.**

Operación	Método	URI	Descripción
Obtener una lista de libros	`GET`	`/books`	Recupera una lista de todos los libros disponibles.
Obtener un libro específico	`GET`	`/books/{id}`	Recupera los detalles del libro con el ID especificado.
Añadir un nuevo libro	`POST`	`/books`	Crea un nuevo libro con los datos proporcionados.
Actualizar un libro existente	`PUT`	`/books/{id}`	Actualiza los detalles del libro con el ID especificado.
Eliminar un libro	`DELETE`	`/books/{id}`	Elimina el libro con el ID.

Errores y códigos HTTP

Los errores HTTP son **códigos de estado** que un servidor web devuelve a un cliente (como un navegador web) para **indicar el resultado** de una solicitud. Estos códigos ayudan a **diagnosticar y resolver problemas** que pueden ocurrir durante la comunicación entre el cliente y el servidor.

Los **códigos de estado HTTP** tienen una relación muy estrecha con las *API REST*, ya que son utilizados para **indicar el resultado de las operaciones** realizadas a través de las solicitudes HTTP.

Las *API RESTful* aprovechan estos códigos para comunicar de manera eficiente y estándar **el estado de una solicitud** entre el cliente y el servidor.

Por ejemplo:

Código de estado HTTP	Utilizado cuando:	Ejemplo
`200 OK`	Una solicitud `GET`, `PUT`, `PATCH` o `DELETE` se ha completado exitosamente.	Un `GET` a `/usuarios/123` devuelve los datos del usuario con ID `123`.
`400 Bad Request`	La solicitud del cliente es inválida o malformada.	Un `POST` a `/usuarios` con datos incorrectos devuelve `400`.
`401 Unauthorized`	La solicitud requiere autenticación.	Un `GET` a `/perfil` sin token de autenticación devuelve `401`.
`404 Not Found`	El recurso solicitado no se encuentra.	Un `GET` a `/usuarios/999` para un usuario que no existe devuelve `404`.
`500 Internal Server Error`	El servidor encuentra un error inesperado.	Un `GET` a `/usuarios` que falla por un problema en la base de datos devuelve `500`.