REST

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

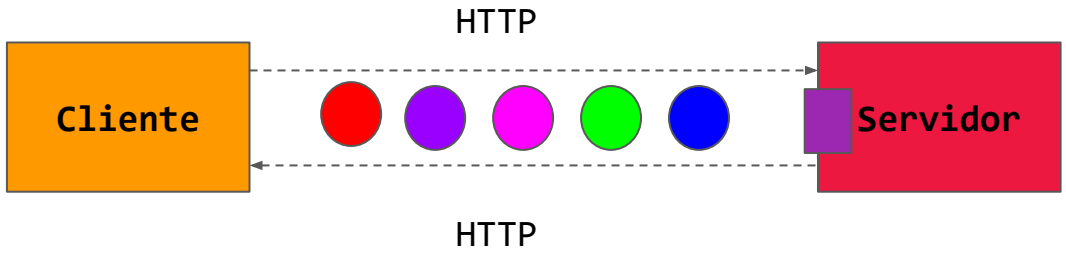
REST es un tipo de arquitectura de servicios que **proporciona estándares** entre sistemas informáticos para establecer **cómo** se van a **comunicar** entre sí.

Índice

1 Conceptos clave

Arquitectura cliente-servidor

REST es una arquitectura del tipo cliente-servidor porque debe permitir que tanto la aplicación del cliente como la aplicación del servidor se desarrollen o escalen sin interferir una con la otra. Es decir, permite integrar con cualquier otra plataforma y tecnología tanto el cliente como el servidor.



Recursos uniformes

Desde el lado del servidor, una **arquitectura REST** expone a los clientes a una interfaz uniforme.

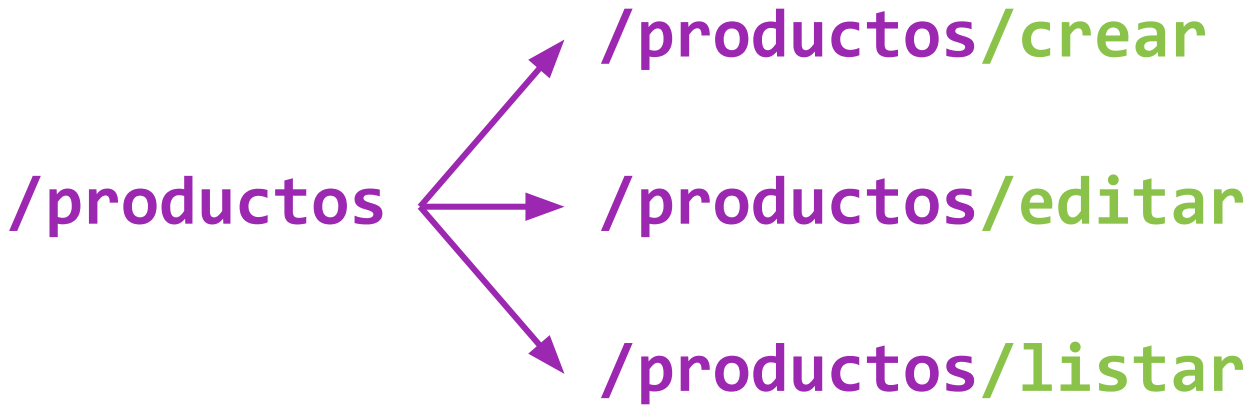
- Todos los recursos del servidor tienen un nombre en forma de URL o hipervínculo.
- Toda la información se intercambia a través del protocolo HTTP.

A esas URL las llamamos **endpoints,** es decir, el servidor expone a los clientes un conjunto de **endpoints** para que este pueda acceder. A esa interfaz uniforme, o sea, al conjunto de **endpoints**, le llamamos **API.**

Un endpoint está ligado al recurso que solicitamos, dicho recurso debe tener solamente un identificador lógico, y este proveer acceso a toda la información relacionada. Veamos un ejemplo a continuación.

Recursos uniformes

El servidor nos expone la URL `/productos/listar`. Dicho endpoint estará ligado al recurso que nos devuelva el listado de los productos solicitados.



Sin estado (*stateless*)

REST propone que todas las i**nteracciones** entre el cliente y el servidor deben ser tratadas como **nuevas** y de forma absolutamente **independiente sin guardar estado**.

Por lo tanto, si quisiéramos —por ejemplo— que el servidor distinga entre usuarios logueados o invitados, debemos mandar toda la información de autenticación necesaria **en cada petición que le hagamos a dicho servidor**.

Esto permite desarrollar aplicaciones más **confiables**, **performantes** y **escalables**.

Cacheable

En **REST**, el cacheo de datos es una herramienta muy importante, que se implementa del lado del cliente, para mejorar la performance y reducir la demanda al servidor.

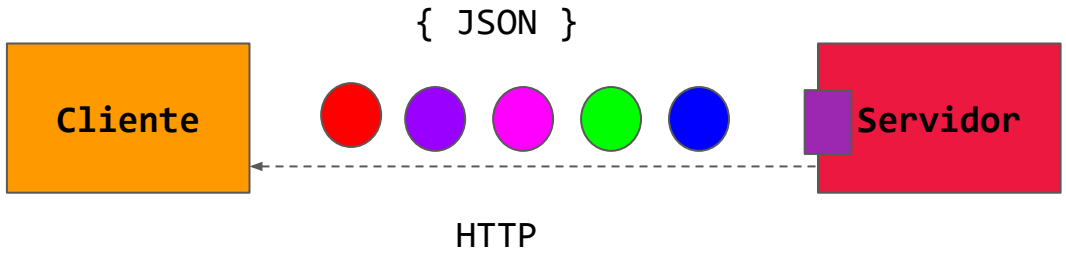
Poniendo cachés en el cliente, nos ahorramos realizar peticiones al servidor.

2 Formatos de envío de datos

Formatos

Cuando el servidor envía una solicitud, este transfiere una representación del estado del recurso requerido a quien lo haya solicitado. Dicha información se entrega por medio de HTTP en uno de estos formatos: JSON (JavaScript Object Notation), RAW, XLT o texto sin formato, URL-encoded.

JSON es el más popular.



JSON

Cuando queramos enviar datos en formato JSON debemos agregar un encabezado en los headers que diga:

```{}
"Content-Type": "application/json"
```

```JSON
{
   "id":1,
   "title": "..."
}
```

RAW

Se utiliza para mandar datos con texto sin ningún formato en particular.

TEXT

Se utiliza para enviar datos que no sean en formato JSON, como archivos HTML y CSS.

URL-encoded

Indica que se nos van a enviar datos codificados en forma de URL.

Por lo tanto, nos envía algo muy similar a un query string.

Un dato enviado mediante este método se vería de la siguiente manera:

```
{}  email%3Dcosme%40fulanito.fox%26password%3Dverysecret
```

Resumen

Una arquitectura REST se caracteriza por seguir los siguientes principios:

- Debe ser una arquitectura **cliente-servidor**.
- Tiene que ser **sin estado**, es decir, no hay necesidad de que los servicios guarden las sesiones de los usuarios (cada petición al servidor tiene que ser independiente de las demás).
- Debe soportar un sistema de **cachés**.
- Debe proveer una interfaz uniforme, para que la información se transfiera de forma estandarizada.
- Tiene que ser un sistema por capas invisible para el cliente.


DigitalHouse>
Coding School