educación IT
CODE YOUR FUTURE

ReactJS Developer

Módulo 6

educación IT
CODE YOUR FUTURE

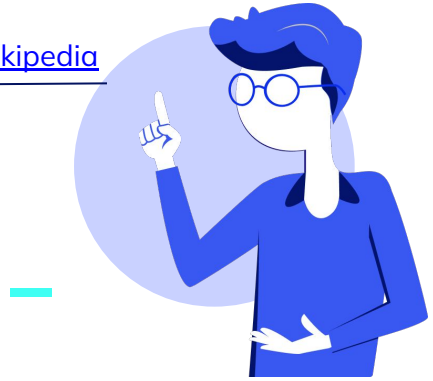
Arquitectura *REST*

Introducción a la arquitectura *REST*

“La transferencia de estado representacional (en inglés, *Representational State Transfer*) o **REST** es **un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.**

El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo”.

Fuente: Wikipedia



Características

- **Arquitectura sin estado:** en la arquitectura REST cada operación debe contar **con todo lo que necesita para funcionar en sí misma**. En otras palabras, la API **no debe almacenar estados ni datos extra**. Una arquitectura *sin estado* significa que **la petición contiene en sí misma todo lo necesario, sin necesidad de datos extra en el servidor**.

- **Uso de métodos HTTP para definir operaciones:** en vez de definir una función para cada tarea a realizar por la API, definimos **un recurso** (*sustantivo*) y **utilizamos los métodos HTTP para definir el comportamiento de la API con ese recurso**; lo que conlleva a una simplificación de la interacción con la API. (Por ejemplo: en vez de programar `getUsers`, `createUser`, `deleteUser` y `editUser`, programamos `User` que se comporte de forma diferente según se acceda por el método `GET`, `POST`, `DELETE` y `PUT`).

- **Identificador de cada estructura de datos (recurso) mediante URIs simples:** una API REST es **un conjunto de recursos accedidos mediante URIs HTTP.** Un recurso es una entidad de la API.
- **Uso de lenguajes de intercambio:** hoy en día, las API REST **intercambian datos**, fundamentalmente, en JSON.

Recursos

La arquitectura REST sirve principalmente para **compartir recursos de hipermedia.** Cada uno **puede ser compartido mediante una arquitectura REST.**

A grandes rasgos y de forma muy simple, podemos decir que un recurso de hipermedia es un recurso compartido por la red.

Por ejemplo:

- Imágenes
- Documentos PDF
- Audio
- Documentos JSON
- Archivos XML
- Hojas de estilo
- Archivos HTML

URI

La **URI** es el **identificador único** para cada recurso. La arquitectura REST define normas para elaborar las URIs.

- La URI debe ser **corta.**
- Debe **describir con precisión el recurso.**
- Debe **centrarse en el recurso** (sustantivo) y no en la acción. La acción se delega al método por el cual se accede a la URI.
- Puede contener **parámetros.**

Ejemplos de **buenas URIs** son:

- /productos
- /usuarios

Ejemplos de **malas URIs** son:

- /obtenerProductos
- /productos/obtenerTodos

Métodos HTTP

Es gracias **al uso intensivo de métodos HTTP** que la arquitectura REST logra proveer una interfaz tan simple para el acceso a recursos.

Los métodos más comunes son:

- `GET` (para leer recurso/s **ya existente/s**)
- `POST` (para crear **un nuevo recurso**)
- `PUT` (para editar **un recurso ya existente**)
- `DELETE` (para **borrar un recurso ya existente**)

JSON

Si bien es cierto que la arquitectura REST sirve para intercambiar todo tipo de recursos, también es cierto que la gran mayoría de las API REST modernas de hoy en día intercambian datos en formato **JSON**, como veremos en la próxima slide.

- Recordemos que JSON (*JavaScript Object Notation*) es **un lenguaje de intercambio de datos.** JSON se basa en una estructura definida denominada **objeto.** El objeto JSON es simplemente **un conjunto de pares "clave":** valores encerrados entre llaves **({** y **})**.
- Podemos representar varios objetos JSON agrupándolos con **corchetes [ ].**
- JSON soporta **casi todos los tipos de datos de JavaScript,** salvo las funciones.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!