MODELOS COMPUTACIONALES DE GESTIÓN ADMINISTRATIVA

REST CASTELLI y MARRACO 2019

Castelli Juan Manuel, Marraco Agustin

Universidad Abierta Interamericana

Índice

[Introducción y principios básicos 2](#_Toc20598600)

[¿Qué es REST? 2](#_Toc20598601)

[¿Cuáles son los principios que lo rigen? 2](#_Toc20598602)

[Recursos 2](#_Toc20598603)

[¿Qué es un recurso? 2](#_Toc20598604)

[Nomenclatura REST 3](#_Toc20598605)

[Definiciones de recursos 3](#_Toc20598606)

[Uso de sustantivos 3](#_Toc20598607)

[Verbos HTTP usados y su significado 3](#_Toc20598608)

[GET 3](#_Toc20598609)

[POST 3](#_Toc20598610)

[PUT 3](#_Toc20598611)

[Delete 3](#_Toc20598612)

[Divisiones de nomenclatura en base al tipo de recurso 4](#_Toc20598613)

[Comparación entre REST y SOAP (Diferencias Claves) 4](#_Toc20598614)

[Protocolo Usado 4](#_Toc20598615)

[Formato de Datos 4](#_Toc20598616)

[Manejo de estado. 5](#_Toc20598617)

[Uso de Caches 5](#_Toc20598618)

# Introducción y principios básicos

## ¿Qué es REST?

REST es un conjunto de siglas que significan Representational State Transfer, que significa transferencia de estado representacional. REST es un conjunto de guias y mejores practicas para sistemas orientados a hipermedia, que es una extesion hipertexto, incluyendo sonido y video.

## ¿Cuáles son los principios que lo rigen?

|  |  |
| --- | --- |
| Cliente Servidor | Separando la interfaz de usuario de donde guardamos los datos mejoramos la portabilidad de la interfaz de usuario en muchas plataformas y mejoramos la escalabilidad de las plataformas |
| Stateless (Sin Estado) | Cada peticion tiene que tener toda la información necesaria para ser llevada a cabo y no puede guardar ningun tipo de información de sesion del lado del servidor. Solo se puede almacenar del lado del cliente. |
| Cacheable | Se tiene que especificar si la informacion contenida en una respuesta es cacheable o no. |
| Interfaz uniforme | Las reglas de REST se orientan a este punto con el fin de simplificar la arquitectura del sistema y mejorar las interacciones. |
| Sistema por capas | Cada componente no puede ver mas alla de la capa en la que esta. |
| Código bajo demanda | Mediante la implementación de applets o scripts se puede extender la funcionalidad del lado del cliente. |

# Recursos

## ¿Qué es un recurso?

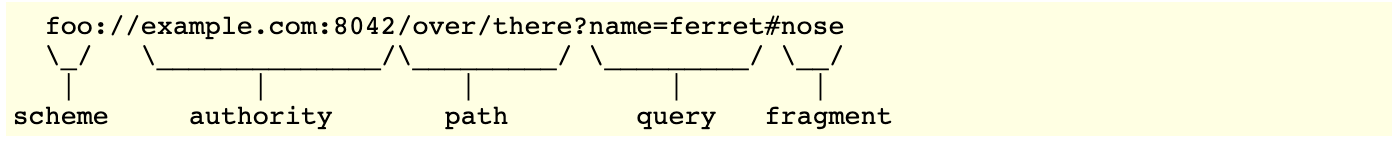
Un recurso es la abstraccion clave de REST mediante la cual se representa la información. REST usa un idenfiticador de recursos para distinguir los recursos que participan en una interacción.

El estado de un recurso en un espacio determinado de tiempo es la **representación del recurso**. Una representacion esta compuesta por metadata y datos.

El mediaType que muchas veces se ve en los headers de las llamadas, determina el tipo de representacion que va a regir a esos datos y como deberan ser procesados.

Las URIs(Uniform resource identifiers) son usadas para indicar un recurso en una interacción

Una URI esta compuesta por 5 partes



El schema se refiere a un espacio de nombres para los recursos que define como los componentes identificaran a uno dentro del mismo.

La autoridad identifica a la entidad que gobierna una parte del espacio de nombre, un server basicamente, al cual accedes por su host name.

El path y la query se usan para mandar información orientada a la identifiacion de un recurso dentro del contexto que se consulta

El fragmento nos permite referenciar una parte especifica del recurso una vez devuelta la informacion solicitada.

# Nomenclatura REST

## Definiciones de recursos

1. Un recurso puede ser singular o plural(singleton or collection) entonces tendrias algo como /clientes /cliente
2. Un recurso puede contener agrupamientos de subrecursos. Cliente/idCliente/compras

## Uso de sustantivos

Los recursos REST deben ser referenciados como sustantivos debido a que los sustantivos tienen propiedades que los verbos no.

## Verbos HTTP usados y su significado

## GET

El metodo get pide una representacion en el momento actual del recurso pedido. El payload de un recurso GET no tiene reglas semanticas definidas.

El metodo GET es seguro, porque es de lectura, y es idempotente.

## POST

El metodo POST hace una solicitud a un recurso para que procese la representacion encerrada dentro de la petición, teniendo en cuenta la semantica que define el recurso a ser consumido. Si el recurso fue creado exitosamente, se debera devolver un codigo HTTP 201.

POST no es ni seguro ni idempotente.

## PUT

El PUT se usa para crear o reemplazar la representacion de un recurso con lo contentido dentro de la petición. Se podria decir que un PUT y un POST son bastante similares, pero la diferencia radica en que intencion se tiene al usuarlos, cuando se hace un POST lo que se busca es que el recurso maneje el estado contetenido en la peticion basandonos en la semantica del recurso, mientras que, PUT se usa para reemplazar estados.

PUT es idempotente pero no seguro.

## Delete

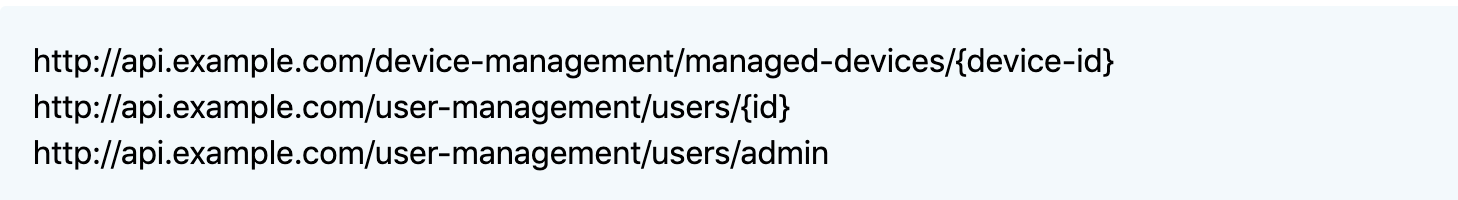
Buscar borrar una asociacion de un recurso.Este metodo no es seguro pero si idempotente.

## Divisiones de nomenclatura en base al tipo de recurso

A su vez, esta nomenclatura se puede dividir dentro de 4 categorias que encasillan la forma en la cual se deben nombrar.

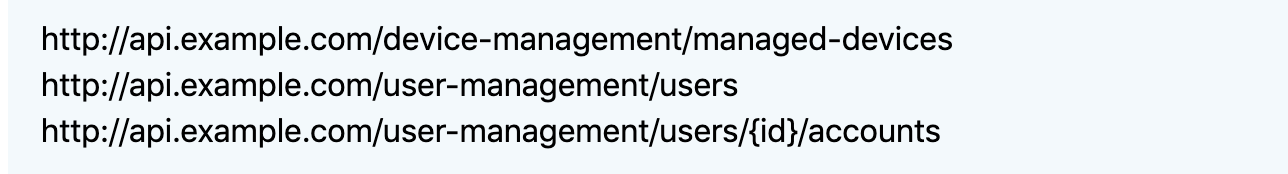
**Documento**

Un documento es una representacion singular que se asemeja por ejemplo a un registro de la base de datos. Generalemente esta dentro de una colección de objetos.



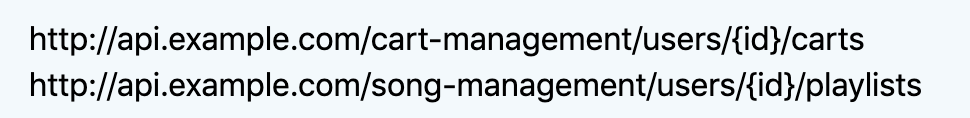
**Colección**

Es un conjunto de recursos que estan manejados del lado del servidor, los cuales pueden llegar a ser editables, permitiendo agregar, sacar y modificar las unidades individuales que integran esa colección.



**Repositorio**

Un repositorio es un conjunto de recursos que permiten guardar y manipular informacion que luego podra o no ser borrada, generalmente en un nivel jerarquico inferior a un objeto que lo contiene.



**Controlador**

Un controlador esta relacionado con funciones ejecutables que tienen parametros y retorno. En estos casos si se justifica llegar a usar un verbo en la URI.



# Comparación entre REST y SOAP (Diferencias Claves)

## Protocolo Usado

Ambos corren sobre HTTP.

## Formato de Datos

SOAP solo usa XML para sus servicios de mensajería.

REST puede usar un conjunto mas variado de estructuras, permitiendo pasar .CSV, JSON y RSS, permitiendo un mayor grado de flexibilidad.

## Manejo de estado.

REST, por definicion, no maneja estado.

SOAP, generalmente lo es, pero el estándar no lo obliga de todos modos, y se puede usar estado si así uno lo desea.

## Uso de Caches

REST facilita mucho mas el cacheo de datos cuando se obtienen a través de métodos GET, permitiendo marcar las respuestas como cacheables o no.

SOAP, al usar todos métodos POST, no permite usar el mecanismo de cacheo, debido a que los posts no garantizan la idempotencia.