REST

Agenda

- Qué es REST? y Porqué usar REST?
- Que son las restricciones REST?
- Terminología REST.
- URIs, buenas prácticas.

Qué es REST?

REpresentational State Transfer

- REST es esencialmente una serie de restricciones.
- Es un estilo de arquitectura para construir aplicaciones basadas en redes de computadoras, particularmente Web.

Restricción Cliente-Servidor

- Fuerza la separación entre cliente y servidor.
- Separa responsabilidades
- Permite la evolución independiente de los componentes del cliente y del server.
- Excelente base para arquitecturas distribuidas.

Restricción “stateless”

- La comunicación entre cliente y servidor debe ser sin estado entre requests.
- Cada request debe contener toda la información necesaria para que el server complete el requerimiento.
- Todos los datos de sesión se deben retornar al cliente al final del request. El cliente es responsable de mantener el estado, NO el server.

Restricción “cache”

- Cada respuesta debe ser explícitamente marcada como cacheable o no cacheable.
- El objetivo es aprovechar la infraestructura existente en la Web.

Restricción Contrato Uniforme

- El servidor y todos los clientes comparten una única interfaz técnica.
- El conjunto de interfaces se denomina API.
- La API debe ser reusable por cualquier cliente y debe soportar cualquier interacción.
- Una API debe proveer servicios abstractos de alto nivel lo más genérico posibles.
- **NO** hay que pensar una API en un cliente en particular.

Restricción Sistemas Multicapa

- Los componentes cliente y servidor, deben permitir intermediarios (Middlewares).
- Los intermediarios deben ser transparentes a protocolo, arquitectura, etc. La comunicación entre cliente y servidor debe ser siempre igual, con o sin intermediarios

Terminología

- **Resource**
  - Clave de la abstracción de REST
  - Cualquier fuente de información
  - Todos los recursos deben ser identificados y accedidos por un id global. La URI

Terminología

- **Representación**
  - Un recurso puede tener varias representaciones.
  - Una representación captura el estado de un recurso en un formato particular, como puede ser JSON, XML, etc
  - Una representación está compuesta por una serie de bytes más los metadatos que los describen.

Terminología

- **Hypermedia-Types**
  - Es un formato específico para una representación más información extra que aporta semántica extra.
  - Se pueden definir nuevos tipos hypermedia.
  - JSON no es un tipo hypermedia, es solo un formato específico de representación de un recurso.

URIs

- **Modelando acciones**
  - Verbos vs Sustantivos
    - La estructura de URIs de una API debe siempre ser expresiva y significativa, no deben perder el sentido. En otras palabras, debe ser predecible.
    - La API debe ser centrada en los recursos y luego en sus operaciones
    - Los verbos HTTP representan las operaciones sobre los recursos.

No deben usarse verbos en las URIs!!!

URIs

- **Modelando acciones**
  - Verbos vs Sustantivos
    - Ejemplo:
      - Mal: `/cuenta/`**`pagar`** ← **`verbo`**
      - Bien: `/cuenta/`**`pago`** ← **`recurso`**
  - Plural vs Singular
    - Ambos están bien!
    - El plural es más aceptado.
    - Pero solo debe usarse uno, no mezclar. (La estructura de URIs debe ser predecible!)

URIs

- **Modelando acciones**
  - Ejemplo:
    - `[GET] /usuarios/` ← obtener colección
    - `[GET] /usuarios/{ID}` ← obtener usuario
    - `[POST] /usuarios/` ← crear usuario
    - `[PUT] /usuarios/` ← modificar/reemplazar usuario
    - `[DELETE] /usuarios/{ID}` ← eliminar usuario

URIs

- **Modelando acciones**
  - Representación de datos en la URI:
    - NO! `/usuarios/27.json`
    - PEOR! `/usuarios/27?format=JSON`
    - SI! `/usuarios/27` más cabecera de requerimiento `Accept` y `Accept-Language`
  - Otras buenas prácticas
    - Usar guiones medios “-“ en vez de guiones de subrayado “_”
    - Usar siempre lowercase