T3: REST

jueves, 15 de mayo de 2025 15:32

"Las API's deben ser independientes del protocolo, sin estado, pero cacheables con una interfaz uniforme, e idempotentes"

REST (*Representations State Transfer*) No es una implementación, ni un estándar, sino, un estilo de **Arquitectura**, como conjunto de buenas prácticas, principios o restricciones que se pueden aplicar sobre distintos protocolos, para diseñar servicios web, que normalmente se comunican a través de peticiones HTTP.

La idea central de REST es tratar a la **información como recursos** que pueden ser manipulados mediante operaciones estándar como GET, POST, PUT, DELETE…

Tiene una arquitectura basada en **cliente-servidor**, **sin estados** (el servidor no guarda sesiones entre peticiones, esto prevé errores (simplifica el balanceo de cargas, permite las cachés), pero degrada la eficiencia), **en capas** (es decir, que puede contar con intermediarios como proxies o gateways entre los extremos), cacheable (o no) (mejora, la eficiencia, y escalabilidad, pero degrada la fiabilidad), con interfaces uniformes y desacopladas (degrada la eficiencia pero, mejora la visibilidad, impulsa la evolución, y desacopla la implementación), **orientada a recursos** (identificables por su URI) y que funciona a través de usar hipermedios -**HATEOAS**- (para navegar por la API con los enlaces).

En REST, una aplicación web, se puede ver como una gran **máquina virtual de estados**, donde los **recursos** son los **nodos** de la máquina, los **enlaces** las **transiciones de estado.**

Las **ventajas**/beneficios/**propiedades** de REST son:
- Heterogeneidad: REST permite la interoperabilidad entre sistemas heterogéneos
- Escalabilidad y evolución (Gracias a la estructura modular, como la separación de su cliente y servidor, y a que es sin estado)
- Visibilidad, ya que cada petición tiene toda la información necesaria
- Fiabilidad, ya que no tiene estados, es fácil recuperarse de errores
- Eficiencia, gracias a la cache, al uso simple de recursos.
- Trabaja con tecnologías y plataformas bien documentadas y maduras HTTP

En el caso de SOAP, es documento WSDL, que describe y documento las funcionalidades del sistema, no es necesario en REST, pero puede dar apoyo como un ***contrato formal del servicio***;
La opción moderna es OpenAPI (antes Swagger) y la Clásica es WADL.

Desde el punto de vista de la seguridad, REST emplea los mecanismo que están ya disponibles en las capas de transporte -SSL/TLS- que permiten cifrado y firma.

SOA es la reacción directa de REST

En REST, cuando un cliente solicita un recurso, se envía en el cuerpo del mensaje HTTP. El formato en el que se envía el recurso se refleja en *"Content-Type"* y aunque hay varios, el más común es "*application/json*"; sin embargo, REST también le permite al cliente negociar el contenido, con un:

```
Accept: application/json
```

Uno de los fundamentos principales de REST es que toda interacción, entre cliente-servidor deben seguir una **interfaz uniforme**, esto significa que:

- Cada recurso estará identificado con un URI único
- Se utilizan los mimos métodos (se les llama *verbos*) de HTTP universalmente para todos los recursos
- No se manipula directamente el recurso, sino, su representación
- Los mensajes deben ser siempre auto descriptivos

- `GET`: lectura
- `POST`: creación
- `PUT`: actualización o creación
- `DELETE`: eliminación

HATEOAS (Hypertext As The Engine Of Application State)
Las respuestas del servidor no necesariamente son siempre datos llanos… Esta función de REST, permite devolver como respuestas, enlaces a siguientes funcionalidades/estados de la aplicación

Ejemplo: Una respuesta a `/pedidos/2` podría incluir:

json Copiar Editar

```
{
  "id": 2,
  "estado": "pendiente",
  "enlaces": [
    {"rel": "confirmar", "href": "/pedidos/2/confirmar"},
    {"rel": "cancelar", "href": "/pedidos/2/cancelar"}
  ]
}
```

Modelo de Madurez de Richardson -Servicios RESTful-

Evalúa el grado en que un servicio web aprovecha los principios REST

Nivel 0: *Swamp of Pox*	• Solo utiliza una URI • Solo utiliza un método GET/POST
Nivel 1: *Recursos*	• Se definen varios recursos con sus URI's • Solo utiliza un método GET/POST
Nivel 2: *Métodos HTTP*	• Usa varios recursos con sus URI's • Usa varios métodos HTTP • Usa todas las operaciones CRUD • Usa correctamente códigos de estado
Nivel 3: *HATEOAS*	• Las representaciones de los recursos incluyen enlaces • El cliente "descubre dinámicamente" cómo interactuar con el servidor

Buenas prácticas:

- Paginación
- Sustantivos no Verbos
- Asociaciones intuitivas
- Si hay muchos argumentos usar un POST o PUT
- Para buscar/filtrar, no se crean más recursos, se parametriza el GET →
- Mantener la compatibilidad hacia atrás

```
GET http://example.com/api/books?sort=rank_asc
GET http://example.com/api/books?category=children
...
```

Ejercicio

Tenemos un cine y queremos modernizarnos.
Queremos que la información de la cartelera y las sesiones esté accesible. Actualmente está disponible en nuestra página web, pero queremos que otras aplicaciones puedan trabajar con esa información de forma sencilla obteniéndola de nuestro servicio web para conseguir visibilidad. Aprovechando esta oportunidad queremos también utilizar este sistema para poder trabajar nosotros él.
¿Nos podrías ayudar?

Ejercicio (cont.)

- Diseña un servicio web basado en REST
- Define las diferentes rutas que necesitamos
- Piensa en las diferentes acciones que se pueden realizar
- Piensa en ejemplos de mensaje y los campos que deberían tener
- Considera los códigos de estado

Verbos	Path	Acciones	Códigos de estado		
GET	/peliculas	Lista las películas disponibles	200	400	404
POST		Agregar una película	201	400	409
PUT	/peliculas/{id}	Actualizar información sobre una película	201	400	404
DELETE		Eliminar una película del catálogo	204		404
GET	/peliculas/{id}/sesiones	Listar las sesiones disponibles para la película	200	400	404
POST		Agregar una sesión para la película X	201	400	409
PUT	/peliculas/{id}/sesiones/{idSesion}	Actualizar información sobre una sesión Z de la película X	201	400	404
DELETE		Eliminar una sesión Z de la película X	204		404