

## Actividad 4. Estándares y buenas prácticas

### Grupo A (Request)

#### Nivel de madurez de Richardson


Es una forma de medir la madurez de tu REST Api basado en 4 modelos, de 0 a 4. Habría que indicar animar a que esto no es un estándar. De hecho, no hay un estándar definido para decir si una Api REST es suficientemente buena o no. Pero la verdad es que el Modelo de Madurez de Richardson fue rápidamente adoptado por la industria de TI, por lo menos es tener "algo a seguir" para hacerle saber que lo está haciendo bien

- Nivel 0:

Si tienes una API definida sobre el protocolo Https para transferir datos, entonces ya estás en ese nivel. ¡Es gratis!

- Nivel 1:

Este nivel pretende tener un buen nivel de abstracción basado en los recursos. Seguir el plural sobre el singular, y no incluir verbosidad en ningún endpoint.

Ejemplos de buena nomenclatura 

`/empleados`

`/departamentos`

`/employees/9e0343b9-a873-4db7-813e-fdeb68305d3b/departments`


`/candidatos/buscar?name=jhon&page=1&offset=20`

El primer endpoint se puede utilizar para crear/actualizar un solo empleado, o incluso para recuperar toda la colección

El segundo endpoint es similar al primero, pero basado en departamentos

El tercero puede utilizarse para actualizar los departamentos de un empleado determinado. También para recuperar los departamentos en los que se ha añadido el empleado (si se encuentra la clave)

El último es un punto final de búsqueda siguiendo la semántica de Google, que recupera la primera página con sólo 20 candidatos que se llaman John.

Ejemplos de mala nomenclatura 

`/empleado` (no plural)

`/departments/create` (no se permite la verbosidad)

`/getEmployeeById` (por favor, no lo haga)

`/search?type=candidate&name=jhon&page=1&offset=20` (Evite usar banderas, nivel de abstracción incorrecto)

Nivel 2:

Dado que el endpoint no permite incluir verbos, tenemos que usar el Verbo Http correcto para especificar el tipo de acción que queremos ejecutar. Además, es una buena práctica elegir el código de estado Http adecuado para saber si la acción solicitada se ha realizado correctamente o no.

POST: sólo para crear nuevos recursos.

GET: recuperar datos sin cambiar el estado del servidor.

PUT: actualizar todo un recurso existente.

PATCH: actualizar parcialmente un recurso existente.

DELETE: para eliminar un recurso del servidor.

- **Versionado:** Ponga el número de versión en la url, por ejemplo:

`http://ejemplo.maes /v1/empleados`

- **Ubicación de los datos:**

@RequestParam: Los datos se extraen de la cadena de consulta de la URL (después del ?).

@PathVariable: Los datos se extraen de la ruta de la URL (entre barras diagonales).

<https://martinfowler.com/articles/richardsonMaturityModel.html>


## Grupo B (Response)

- JSON: Las respuestas deben ser un objeto JSON (no un array). Usar un array para retornar resultados limita la capacidad de incluir metadata sobre resultados, y limita la capacidad de las API's para agregar top-level keys en el futuro.

Las claves de los atributos deben ser legibles y saber su significado con solo leerlas.

Más info en [json.org](http://json.org)

No valores en claves. La metadata solamente debe contener propiedades directas a la respuesta, no propiedades relacionadas a la información de la respuesta.

Ejemplo válido 

```
"tags": [  
  {"id": "125", "name": "Ciudadano"},  
  {"id": "834", "name": "Servicios"}  
],
```

Ejemplo NO válido ✕

```
"tags": [  
  {"125": "Ciudadano"},  
  {"834": "Servicios"}  
],
```

**Json de errores.** En Spring Boot 3, se ha añadido una clase ProblemDetail que estandariza la salida de un endpoint si ha habido errores. Tiene los siguientes atributos

```
{  
  
  "type": "https://example.com/probs/out-of-stock",  
  
  "title": "The requested product is out of stock",  
  
  "status": 404,  
  
  "detail": "The product with ID 123 is currently out of stock.",  
  
  "instance": "https://example.com/products/123"  
}
```

- **HTTP Codes:** Los códigos de estado HTTP se dividen en 5 «tipos». Se trata de agrupaciones de respuestas que tienen significados similares o relacionados. Saber qué son puede ayudarte a determinar rápidamente la sustancia general de un código de estado antes de que vayas a buscar su significado específico.

Las cinco clases incluyen:

- **100s:** Códigos informativos que indican que la solicitud iniciada por el navegador continúa.
- **200s:** Los códigos con éxito regresaron cuando la solicitud del navegador fue recibida, entendida y procesada por el servidor.
- **300s:** Códigos de redireccionamiento devueltos cuando un nuevo recurso ha sido sustituido por el recurso solicitado.
- **400s:** Códigos de error del cliente que indican que hubo un problema con la solicitud.
- **500s:** Códigos de error del servidor que indican que la solicitud fue aceptada, pero que un error en el servidor impidió que se cumpliera.

Los más usados:

- **200:** «Todo está bien». Este es el código que se entrega cuando una página web o recurso actúa exactamente como se espera.
- **201:** «Creado». El servidor ha cumplido con la petición del navegador y, como resultado, ha creado un nuevo recurso.
- **204:** «Sin contenido». Este código significa que el servidor ha procesado con éxito la solicitud, pero no va a devolver ningún contenido.
- **400:** «Mala petición». El servidor no puede devolver una respuesta debido a un error del cliente. Vea nuestra guía para resolver este error.

- **401:** «No autorizado» o «Se requiere autorización». Esto es devuelto por el servidor cuando el recurso de destino carece de credenciales de autenticación válidas.
- **403:** «El acceso a ese recurso está prohibido». Este código se devuelve cuando un usuario intenta acceder a algo a que no tiene permiso para ver. Por ejemplo, intentar acceder a un contenido protegido por contraseña sin registrarse podría producir un error 403.
- **404:** «No se encontró el recurso solicitado». Este es el mensaje de error más común de todos ellos. Este código significa que el recurso solicitado no existe, y el servidor no sabe si alguna vez existió.
- **500:** «Hubo un error en el servidor y la solicitud no pudo ser completada». Este es un código genérico que simplemente significa «error interno del servidor». Algo salió mal en el servidor y el recurso solicitado no fue entregado.

<https://www.restapitutorial.com/httpstatuscodes.html>

## Grupo C (Pruebas)

- Pruebas unitarias. Características:
  - Se prueba cada método de la API de forma individual.
  - Asegura que cada método devuelve el resultado esperado para diferentes entradas.
  - Utilizar herramientas como JUnit o Mockito para automatizar las pruebas unitarias.
- Pruebas integradas:
  - Prueba cómo interactúan diferentes partes de la API.
  - Asegura que la API funciona correctamente como un todo.
  - Utiliza herramientas como Postman o RestAssured para automatizar las pruebas de integración.

### El patrón Gherkin:

Es una forma de escribir casos de prueba que se basa en tres pasos principales:

1. Contexto (Given): Describe el estado inicial del sistema antes de que se realice la acción.
2. Acción (When): Describe la acción que se va a realizar en el sistema.
3. Resultado (Then): Describe el resultado esperado después de que se haya realizado la acción.

Ejemplo:

- Contexto:

Un usuario está en la página de inicio de una tienda web.

El usuario tiene un carrito de compras vacío.

- **Acción:**

El usuario agrega un producto al carrito de compras.

- **Resultado:**

El producto se agrega al carrito de compras.

El número de artículos en el carrito de compras aumenta en 1.

Se muestra un mensaje de confirmación al usuario.

Tener en cuenta en las pruebas unitarias de cada método:

- **Entradas y salidas:** Definir entradas de prueba que representen diferentes escenarios posibles y verificar las salidas esperadas. Tanto resultados correctos como resultado erróneos falta de parámetros obligatorios...
- **Verbos:** Definir correctamente el verbo del método de la API (GET, POST, PUT, DELETE).
- **Mensajes de error:** Verificar que se devuelven los mensajes de error correctos para las entradas no válidas.

Buenas prácticas:

- **Nombres descriptivos:** Nombrar las pruebas de forma descriptiva para que sea fácil identificar su propósito.
- **Atomicidad:** Asegurar que cada prueba sea atómica y que no dependa de otras pruebas.
- **Independencia:** Las pruebas no deben depender del orden de ejecución.