Restfull API

Gabriel Rodríguez Flores

November 23, 2021

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

    - Teoría y puesta en práctica de una Restfull API
- Trabajo
- Restful API vs OData vs GraphQl

Contents

1 Teoría **3**
1.1 Métodos . . . 3
1.2 Buenas prácticas . . . 3
1.2.1 Uso de sustantivos . . . 3
1.2.2 Manejo y formato de los errores . . . 4
1.2.3 Filtrado, ordenación y paginado . . . 4
1.2.4 Seguridad . . . 5
1.2.5 Versionado de API . . . 5
1.3 Alternativas . . . 5
1.3.1 SOAP . . . 5
1.3.2 GraphQL . . . 5
1.3.3 OData . . . 5

2 Ejemplos **5**

3 Ejercicios **5**

4 Entregables **6**
4.1 En clase . . . 6
4.2 Tarea . . . 6
4.2.1 Trabajo . . . 6

1 Teoría

- RESTful API
- Buenas Prácticas Restfull API
- REST en Nodejs

1.1 Métodos

- Get: Es lo que hace el navegador al entrar en una página
  - Envío
  - Backend
- Post: Para envío de datos extensos o ficheros. Encripta los datos.
- Put: Sobreescribir los datos del recurso objetivo
- Patch: Actualizar los datos del recurso objetivo
- Delete: Eliminar el recurso objetivo

1.2 Buenas prácticas

- Descripción de los tipos de métodos
- API Restful buenas prácticas
- Códigos HTTP: Cuáles y cómo usarlos

1.2.1 Uso de sustantivos

Las rutas han de ser sustantivos, ya que son los métodos (verbos) los que van a decidir la acción a realizar.

- Ej. `/users`

También es interesante el uso de en capsulación, para acceder a un parámetro de un recurso concreto.

- Ej. `/users/:userId/comments`

Donde estaríamos accediendo a los comentarios del usuario con el ID que se inserte.

1.2.2 Manejo y formato de los errores

- Control de errores
- Facebook: Ejemplo de buenas prácticas
  - Objeto y control de errores

El control de los errores ha de ser centralizado, eso significa que toda respuesta ha de ser manejada en un mismo lugar (middleware).

Tambien es conveniente definir un estándar de estructura de error incluyendo los datos que se consideren necesarios. Ha de tener al menos:

```
{
  "code": 404,
  "error": "Not Found",
  "message": "Error: Path not found"
}
```

1.2.3 Filtrado, ordenación y paginado

Filtrado: Se trata de definir uno o varios parámetros en la búsqueda de recursos (BBDD o similar) y limitar la respuesta a los que coincidan con estos criterios.

Ordenación: Se trata de seleccionar un parámetro para que marque el orden de los recursos a devolver.

Se suele indicar con un número positivo (1) si es orden ascendente o negativo (-1) i es orden descendente.

Paginado: Se indica un número de elementos a mostrar, y devolverá esta cantidad junto con otros parámetros, en lugar de devolver la totalidad de los elementos (que pueden ser muchos).

Se suelen ver 2 formas de paginado:

- Offset y limite: Número del elemento por el que se empieza y cuántos se muestran.
- Paginas y cantidad: Número de página actual y elementnos por página.

Otras técnicas y estándares

- HATEOAS proporciona directamente los enlaces para navegación entre páginas
  - Paypal: caso de uso

1.2.4 Seguridad

Añadir seguridad a la API, sistema de login y token.

1.2.5 Versionado de API

Se incluye en la raiz de la ruta el path `vX`, donde `X` indica el número de la versión de la api que se está utilizando.

- Ejemplo de url con ruta *ping* en la versión 1 de la api: `localhost:3000/api/v1/ping`

1.3 Alternativas

1.3.1 SOAP

Es anterior a REST y ya no se usa porque es más pesado. Basado en XML

1.3.2 GraphQL

Completamente distinto, haciendo una analogía con el meme de mesas, camareros y cocina para una REST API, GraphQL es un Buffet libre.

1.3.3 OData

Sigue siendo una API, pero se gestiona de manera diferenta a REST API.

2 Ejemplos

3 Ejercicios

- Realizar un servidor, aplicando el uso de buenas prácticas, que permita:
  - Realizar el CRUD completo de una colección de usuarios bajo la ruta `/users`
    * Listar todos los usuarios que existan
    * Recoger un usuario concreto dado su ID
    * Crear / Actualizar / Sobreescribir / Borrar usuarios

  - Debe permitir el almacenamiento en memoria para mantener los ususarios creados y poder recogerlos
    * Para un uso práctico y no muy pesado, se puede precargar el contenido y tener usuarios iniciados al cargar (integrar el lanzamiento en los loaders).
  - Prestar atención a:
    * Endpoints REST y métodos HTTP
    * Control de errores
    * Uso correcto de códigos HTTP
  - Realización de los test unitarios

4 Entregables

4.1 En clase

Dejar tiempo de estudio y hacer un *Kahoot*

- Enlace a Kahoot

4.2 Tarea

Realizar el ejercicio detallado.

4.2.1 Trabajo

- Investigación y comparativa de REST vs OData vs GraphQL
  - Escribir documentación en Markdown (Readme.md)
  - Seleccionar uno entre OData y GraphQL y realizar una puesta en práctica con Nodejs