Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Unidad 4 -Tarea 9:

API RESTful

API RESTful (Representational State Transfer)

Introducción

Las **APIs RESTful** son uno de los enfoques más utilizados para el diseño de interfaces de programación en servicios web modernos. REST (Representational State Transfer) es un estilo arquitectónico definido por Roy Fielding en el año 2000, que propone un conjunto de principios para diseñar sistemas distribuidos escalables y sencillos. Una API RESTful es una API que cumple con estos principios y utiliza el protocolo HTTP para facilitar la comunicación entre clientes y servidores. Gracias a su simplicidad, eficiencia y compatibilidad con la web, REST se ha convertido en el estándar de facto para el desarrollo de aplicaciones distribuidas y sistemas conectados.

Desarrollo

Una API RESTful se basa en la idea de **recursos** que pueden ser accedidos y manipulados a través de **métodos HTTP** estándar. Cada recurso es identificado por una **URI (Uniform Resource Identifier)** y puede ser representado en distintos formatos, como JSON o XML, siendo JSON el más popular en la actualidad.

Características clave de una API RESTful:

1. **Uso del protocolo HTTP:** Cada operación se asocia con un método HTTP:
    - **GET:** Obtener un recurso.
    - **POST:** Crear un nuevo recurso.
    - **PUT:** Actualizar completamente un recurso existente.
    - **PATCH:** Actualizar parcialmente un recurso.

- **DELETE:** Eliminar un recurso.

2. **Identificación de recursos mediante URIs:** Cada recurso tiene una dirección única.
   Ejemplo: `https://api.ejemplo.com/productos/23`

3. **Sin estado (stateless):** Cada solicitud del cliente debe contener toda la información necesaria. El servidor no guarda contexto entre solicitudes.

4. **Uso de representaciones:** Un recurso puede representarse en diferentes formatos. Lo habitual es usar JSON, aunque también puede utilizarse XML, YAML, etc.

5. **Interfaz uniforme:** Todos los recursos se manipulan mediante una interfaz común, simplificando la comprensión y uso de la API.

6. **Caché y escalabilidad:** Las respuestas pueden ser almacenadas temporalmente para mejorar el rendimiento.

Ventajas:

- Simplicidad de uso y comprensión.
- Ligereza en comparación con protocolos más complejos como SOAP.
- Fácil integración con clientes HTTP estándar (navegadores, scripts, apps móviles).
- Compatible con arquitecturas modernas (microservicios, aplicaciones móviles, IoT).

Desventajas:

- No incluye estándares avanzados de seguridad o transacciones (como SOAP).
- Puede no ser adecuado para operaciones complejas o fuertemente acopladas.
- Depende del diseño correcto de URIs y convenciones.

Ejemplo:

Supongamos una API RESTful para gestionar una base de datos de libros. A continuación, se muestra cómo se usarían los métodos HTTP:

- **GET** `/libros` → Lista todos los libros.
- **GET** `/libros/10` → Devuelve los datos del libro con ID 10.
- **POST** `/libros` con datos en el cuerpo → Crea un nuevo libro.
- **PUT** `/libros/10` con datos en el cuerpo → Reemplaza completamente el libro con ID 10.
- **PATCH** `/libros/10` con nuevos datos → Modifica parcialmente el libro con ID 10.
- **DELETE** `/libros/10` → Elimina el libro con ID 10.

Los datos usualmente se intercambian en formato JSON:

```
{
  "titulo": "Cien años de soledad",
  "autor": "Gabriel García Márquez",
  "anio": 1967
}
```

Conclusión

Las APIs RESTful representan una solución elegante y práctica para la comunicación entre aplicaciones distribuidas. Su adopción masiva en el desarrollo de servicios web, microservicios e IoT se debe a su sencillez, su buen rendimiento y su alineación con los estándares del protocolo HTTP. Sin embargo, para obtener buenos resultados, es importante diseñarlas cuidadosamente, aplicar principios REST adecuados y acompañarlas con una buena documentación y medidas de seguridad.

Bibliografía

1. Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral Dissertation, University of California, Irvine.

2. Richardson, L., & Ruby, S. (2008). *RESTful Web Services*. O'Reilly Media.

3. Masse, M. (2011). *REST API Design Rulebook*. O'Reilly Media.

4. Webber, J., Parastatidis, S., & Robinson, I. (2010). *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly.

5. API Academy. (2023). *Best Practices for Designing REST APIs*. apiacademy.co