Programación IV
Profesora Cynthia Estrada
Fecha: 20/08

"REST,URI y GraphQL"

Integrantes:
- "Díaz Rossini Juan José",
- "Gallo Genaro"
- "Navarro Victor Leandro"

---

1. **¿Cómo se explican los servicios REST?**
2. **¿Cuándo se crearon los servicios REST?**
3. **¿Qué significa REST?**
4. **¿Cuáles son las características de estos servicios?**
5. **¿Qué es URI?**
6. **¿Qué función cumplen dichos servicios?**
7. **¿En qué lenguajes podemos utilizar REST?**
8. **Realice un cuadro comparativo entre las diferencias de REST y GraphQL**
9. **¿Qué problema solucionó GraphQL con Facebook?**
10. **¿En qué lenguajes puede utilizar GraphQL?**
11. **¿Cómo es el funcionamiento de GraphQL?**

¿Cómo se explican los servicios REST?

Los servicios REST son un estilo de arquitectura muy utilizado en el desarrollo de aplicaciones web y móviles. Su principal objetivo es permitir que distintos sistemas se comuniquen entre sí de manera clara y ordenada, sin importar en qué lenguaje estén programados. REST se apoya en el protocolo HTTP y en operaciones estándar como GET, POST, PUT y DELETE, que son fácilmente entendibles y ampliamente compatibles.

Cada recurso dentro de un servicio REST se trata como una entidad independiente (por ejemplo: usuarios, productos, pedidos) y se accede a través de un identificador único llamado URI. Gracias a esta simplicidad, REST se ha convertido en una de las formas más comunes de construir APIs que conectan el frontend (interfaz) con el backend (servidor).

¿Cuándo se crearon los servicios REST?

REST fue creado en el año 2000 por Roy Fielding, uno de los autores de la especificación HTTP. Su propuesta nació como parte de su tesis doctoral, en la que describió un nuevo estilo arquitectónico que buscaba mejorar la escalabilidad y la simplicidad de los sistemas distribuidos.

En aquel momento, existían otras formas de comunicación entre aplicaciones, como SOAP, que eran más complejas y rígidas. REST se convirtió rápidamente en una alternativa más liviana, sencilla y práctica, lo que permitió su adopción masiva en el mundo del desarrollo web.

¿Qué significa REST?

Las siglas REST significan Representational State Transfer, que en español se traduce como Transferencia de Estado Representacional. Este concepto hace referencia a que los clientes y servidores intercambian representaciones de los recursos (generalmente en formatos como JSON o XML) y que cada petición incluye toda la información necesaria para completarse, sin depender de un estado previo en el servidor.

¿Cuáles son las características de estos servicios?

Algunas de las principales características de REST son:

- **Sin estado (stateless):** cada solicitud que hace un cliente al servidor debe contener toda la información necesaria, ya que el servidor no guarda datos de la sesión.
- **Basados en recursos:** cada elemento (ejemplo: un usuario, un producto) se trata como un recurso con su propio URI.
- **Uso de métodos HTTP estándar:** se utilizan operaciones como GET (consultar), POST (crear), PUT (actualizar) y DELETE (eliminar).
- **Formato flexible:** si bien lo más común es usar JSON, también se pueden intercambiar datos en XML, HTML o texto plano.
- **Escalables y simples:** permiten que aplicaciones de distinto tamaño y complejidad se comuniquen de forma eficiente y ordenada.

¿Qué es URI?

Un URI (Uniform Resource Identifier) es una cadena de caracteres que sirve para identificar un recurso de forma única en la web. Gracias al URI, es posible acceder a un recurso específico en una aplicación o servicio.

Por ejemplo:

- **https://api.ejemplo.com/usuarios/1 → identifica al usuario con ID 1.**

- **https://tienda.com/productos/35 → apunta al producto con ID 35.**

El URI es fundamental en REST porque cada recurso del sistema necesita un identificador único para que pueda ser consultado o manipulado mediante las operaciones HTTP.

¿Qué función cumplen dichos servicios?

Los servicios REST cumplen la función de intermediarios de comunicación entre aplicaciones diferentes. Permiten que un sistema (por ejemplo, un sitio web o una app móvil) pueda acceder a la información y funciones que ofrece otro sistema (como un servidor o base de datos).

Esto hace posible que existan APIs abiertas o privadas que integren aplicaciones, por ejemplo: un sistema de pagos online, un servicio de mapas o una plataforma de mensajería. En la actualidad, REST es la base de la mayoría de las integraciones modernas entre aplicaciones.

¿En qué lenguajes podemos utilizar REST?

REST no está limitado a un lenguaje específico, ya que se basa en HTTP, que es universal. Por lo tanto, puede implementarse en prácticamente

cualquier lenguaje de programación moderno. Algunos de los más comunes son:

- **Java y JavaScript (Node.js): muy usados en aplicaciones web y móviles.**
- **Python: con frameworks como Flask o Django.**
- **PHP: uno de los más utilizados en desarrollo web clásico.**
- **C# (.NET): para aplicaciones empresariales.**
- **Ruby, Go, Kotlin, Swift, entre otros.**
  Gracias a esta versatilidad, REST se ha convertido en un estándar mundial en la creación de APIs.

Característica	REST	GraphQL
Creación	Año 2000, Roy Fielding.	Año 2012 en Facebook (publicado en 2015).
Forma de trabajo	Usa varios endpoints (uno por recurso).	Usa un único endpoint para todas las consultas.
Datos recibidos	Puede devolver más o menos datos de los necesarios (overfetching o underfetching).	Devuelve exactamente lo que el cliente pide, sin información extra.
Formato de respuesta	JSON o XML.	Siempre JSON.
Flexibilidad	Menos flexible, se requieren múltiples peticiones para datos complejos.	Muy flexible, una sola consulta puede obtener todos los datos necesarios.
Uso típico	APIs públicas tradicionales (Twitter, APIs REST clásicas).	APIs modernas con alta demanda de datos personalizados (Facebook, GitHub GraphQL).

Cuadro comparativo entre REST y GraphQL

¿Qué problema solucionó GraphQL con Facebook?

En Facebook surgió la necesidad de mostrar grandes volúmenes de datos en su aplicación móvil. Con REST, cada pantalla de la app requería múltiples peticiones a diferentes endpoints, lo que generaba lentitud y consumo de datos innecesarios.

Además, a veces REST devolvía datos de más (overfetching) o datos insuficientes (underfetching). Esto afectaba directamente la experiencia de los usuarios.

➡ GraphQL resolvió este problema al permitir realizar una única consulta en la que el cliente especifica exactamente qué información necesita, reduciendo la cantidad de peticiones y optimizando el rendimiento de la aplicación.

¿En qué lenguajes puede utilizar GraphQL?

GraphQL es independiente del lenguaje de programación. Gracias a que se publicaron librerías oficiales y de la comunidad, se puede usar en casi todos los lenguajes modernos. Algunos de los más comunes son: JavaScript, TypeScript, Python, Java, PHP, C#, Ruby, Go, Kotlin y Swift. Esto lo convierte en una herramienta sumamente flexible y adoptada por grandes empresas tecnológicas.

¿Cómo es el funcionamiento de GraphQL?

El funcionamiento de GraphQL se basa en un modelo cliente-servidor.

1. **El cliente realiza una query (consulta) en la que define de forma explícita qué datos necesita (por ejemplo: `usuario { nombre, email }`).**

2. **El servidor utiliza un schema (esquema) que describe los tipos de datos disponibles y sus relaciones.**

3. **El servidor procesa la consulta y devuelve al cliente únicamente la información solicitada, en formato JSON.**

Este funcionamiento evita la sobrecarga de datos y mejora la eficiencia, ya que el cliente recibe exactamente lo que pidió en una sola petición.