**🧩 ¿Qué es GraphQL y cómo funciona?**

**GraphQL** es un lenguaje de consultas para APIs y un entorno de ejecución para procesar esas consultas.

🔧 **¿Cómo funciona?**

* El cliente especifica exactamente qué datos necesita.
* El servidor responde con **solo esos datos**, ni más ni menos.
* Usa un solo **endpoint** (/graphql) para todas las consultas y mutaciones (modificaciones).
* Permite **consultas anidadas**, relaciones entre objetos, y tipado fuerte.

Ejemplo de consulta GraphQL:

{

usuario (id: "123") {

nombre

correo

}

}

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**📐 ¿Qué significa SDL?**

**SDL = Schema Definition Language**  
Es el lenguaje que GraphQL usa para definir la **estructura del esquema** (los tipos de datos, consultas, mutaciones, etc.).

Ejemplo:

type Usuario {

id: ID!

nombre: String!

correo: String!

}

**🧱 ¿Qué es una arquitectura de microservicios?**

Es un enfoque de diseño de software en el que una aplicación se construye como un conjunto de **servicios pequeños, independientes y desplegables por separado**.

Cada microservicio:

* Se enfoca en una única funcionalidad (usuarios, pagos, productos, etc.)
* Tiene su propia base de datos (opcional).
* Se comunica con otros microservicios mediante HTTP, mensajería, etc.

✅ Ventajas:

* Escalabilidad
* Despliegue independiente
* Tolerancia a fallos

❌ Desventajas:

* Complejidad en la gestión
* Requiere infraestructura robusta

**🏢 ¿Qué empresas usan GraphQL?**

Muchas empresas grandes lo usan en producción. Algunas son:

* **Facebook** (lo creó y lo sigue usando)
* **GitHub** (toda su API oficial es GraphQL)
* **Shopify**
* **Netflix**
* **Twitter**
* **Airbnb**
* **Pinterest**

**👨‍💻 ¿Quién creó GraphQL y cuándo?**

* **Creadores**: Lee Byron, Dan Schafer, y Nick Schrock **Facebook**
* **Año**: **Facebook desarrolló la tecnología internamente a partir de 2012 y la lanzó al público en 2015.**

Fue diseñado para mejorar el rendimiento de sus apps móviles (especialmente la app de Facebook), donde REST no era lo suficientemente eficiente.