Trabajo en Grupo sobre:
API GraphQL

Integrantes del Grupo:

- Santiago Galiano Segura
- Rafael Sabater Carbonell
- Luis Cutillas Armenteros
- David Bernabeu Ferrer

Introducción:

Definición y tipos de APIs:

Una API (Application Programming Interface) es un conjunto de protocolos y herramientas que permiten a diferentes sistemas o aplicaciones comunicarse entre sí. Por lo general, una API incluye funciones o métodos que pueden ser llamados por otra aplicación y que devuelven datos o realizan acciones como respuesta a dicha llamada.

Por ejemplo, una API puede ser usada para conectarse a un servicio en línea, como una base de datos o un sitio web, o para conectarse a otro software de forma local, es decir, haciendo uso de la misma máquina. Las APIs son de gran utilidad debido a que permiten a las aplicaciones acceder a diversas funcionalidades y datos de otras aplicaciones o servicios sin tener que conocer los detalles de cómo estos funcionan internamente.

En la actualidad, podemos encontrar diferentes tipos de APIs, algunos de los cuales son:

- API REST (Representational State Transfer): utilizan el protocolo HTTP para obtener datos de un servidor o enviar nuevos datos al mismo.
- APIs de RPC (Remote Procedure Call): permiten a los clientes llamar directamente a métodos de un servidor para obtener o enviar datos.
- APIs de eventos: permiten a los servidores enviar notificaciones a los clientes cuando ocurren ciertos eventos.
- APIs de librerías: son utilizadas por programadores para acceder a funcionalidades específicas de una biblioteca o framework.
- API de GraphQL: utilizan un lenguaje específico llamado GraphQL para la recuperación y manipulación de datos de un servidor.

API GRAPHQL:

Definición:

Las APIs GraphQL utilizan un lenguaje de consulta que fue desarrollado por Facebook para permitir a los clientes obtener exactamente los datos que necesitan de un servidor, y nada más que dichos datos. En lugar de exponer datos a través de una serie de endpoints REST estáticos, una API GraphQL permite a los clientes enviar consultas específicas que describen los datos que necesitan. El servidor ejecuta la consulta y devuelve los datos solicitados.

Una de las ventajas principales, como ya se ha comentado, es que permite a los clientes obtener únicamente los datos que necesitan, en lugar de obtener un conjunto de datos predeterminado como ocurre cuando se usan APIs REST. Esto es muy útil cuando se trata de aplicaciones móviles o de otro tipo que necesitan acceder a una gran cantidad de datos desde un servidor, ya que puede ayudar a reducir el ancho de banda y mejorar la velocidad de la aplicación.

Otra ventaja fundamental es que permite a los clientes realizar consultas múltiples en una sola solicitud, lo que puede ser muy útil en aplicaciones que necesitan acceder a varios conjuntos de datos simultáneamente.

Desde una perspectiva con carácter más técnico, el uso de las APIs GraphQL presenta las siguientes ventajas:

- Un esquema de GraphQL establece una fuente única de información en una aplicación de GraphQL. Por lo tanto, ofrece una forma de unificar toda la API de una organización o empresa.
- Las llamadas a la API se gestionan en un solo recorrido de ida y vuelta. Los clientes obtienen lo que solicitan sin que se genere una sobrecarga.
- Los tipos de datos bien definidos reducen los problemas de comunicación entre el cliente y el servidor.
- Es una herramienta introspectiva. El cliente puede solicitar información sobre los tipos de datos disponibles. Además permite la generación automática de documentación sobre la API.
- GraphQL permite que la API evolucione sin afectar a las consultas que ya existían previamente.

Pero no todo son ventajas, por lo que también encontramos algunos inconvenientes:

- GraphQL presenta una curva de aprendizaje elevada para los desarrolladores que tienen experiencia con las API REST.
- Delega gran parte del trabajo de las consultas al servidor, lo que representa mayor complejidad para los desarrolladores de servidores.
- Requiere de estrategias de gestión de APIs diferentes a las API REST, las cuales pueden llegar a ser notablemente más complejas.
- El almacenamiento en caché es más complejo que con las API REST.

- Los desarrolladores y los encargados del mantenimiento de la API tienen la tarea de escribir un esquema GraphQL que sea fácil de mantener en el tiempo.

Historia:

El lenguaje GraphQL fue desarrollado inicialmente por la empresa Facebook, hoy conocida como Meta. Los primeros usos de APIs desarrolladas con este lenguaje los encontramos en el año 2012, ligados sobre todo a las aplicaciones móviles, las cuales tenían que ejecutarse en dispositivos que, en aquella época, tenían muy poca potencia y recursos si los comparamos con los que tenemos hoy en día, tras 10 años de avances.

En el año 2015 se publicó oficialmente el lenguaje como un proyecto open source, lo que permitió a otros desarrolladores utilizar y contribuir al proyecto. Desde entonces, se convirtió en una opción popular para construir APIs y ha sido adoptada por numerosas empresas y organizaciones, incluyendo GitHub, Pinterest y Shopify.

En 2018, la Fundación Linux adquirió GraphQL y se convirtió en un proyecto de código abierto liderado por la comunidad. Desde entonces, ha seguido creciendo y mejorando y sigue siendo una de las opciones más populares para la construcción de APIs en la actualidad.

Evolución de la API GraphQL:

Desde la creación de GraphQL, ha ganado popularidad y ha sido adoptado por muchas empresas y organizaciones para construir aplicaciones software y APIs.

A medida que ha evolucionado, GraphQL ha recibido varias actualizaciones y mejoras. Algunos de los cambios más importantes incluyen:

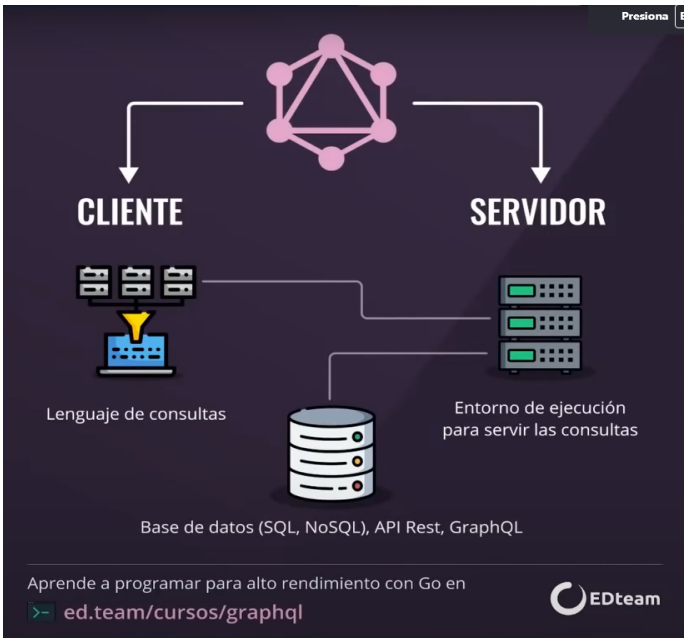
- **Mejoras en la sintaxis**: GraphQL ha mejorado su sintaxis a lo largo de los años, añadiendo características como soporte para fragmentos y operadores.
- **Nuevas características**: GraphQL ha añadido nuevas características como soporte para mutaciones (que permiten modificar datos en el servidor), subscriptions (que permiten a los clientes recibir actualizaciones en tiempo real de los datos) y directives (que permiten a los usuarios modificar el comportamiento de una consulta).
- **Ampliación del ecosistema:** GraphQL ha crecido para incluir una amplia variedad de herramientas y bibliotecas para diferentes lenguajes de programación y plataformas.

En general, GraphQL ha evolucionado para ser una herramienta más potente y versátil para construir APIs y aplicaciones de software.

Funcionamiento GraphQL:

En primer lugar sabemos que en una comunicación de API tenemos dos extremos, el cliente y el servidor, para el caso de GraphQL las tareas que realiza cada extremo son las siguientes:

- **Cliente:** Está constituido por un lenguaje de consultas para recuperar una serie de parámetros o atributos del servidor. Un ejemplo de lenguaje de consultas podría ser SQL en el que nosotros ejecutamos una sentencia **select** y podemos obtener los parámetros que a nosotros nos interesa.
- **Servidor:** compuesto por un entorno de ejecución que nos permite entregar las consultas realizadas por el cliente, conectándose a la capa de datos. Si nos abstraemos al ejemplo anterior es como en SQL el que ejecuta las consultas es el motor de la bbdd.

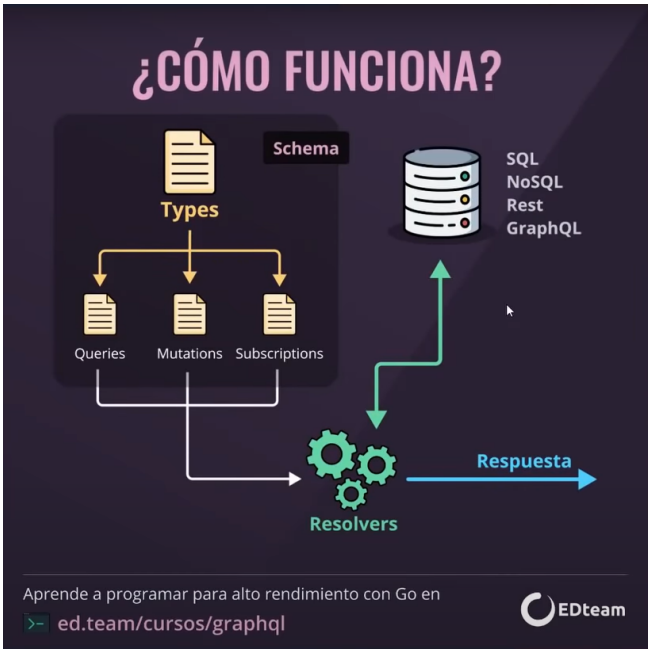


Una de las principales ventajas de GraphQL es que nos elimina cualquier dependencia tanto en lenguaje de programación utilizado como base de datos utilizada. El funcionamiento a grandes rasgos de una API GraphQL consiste en lo siguiente:

1. Definición de una estructura de datos en el backend de nuestra aplicación.
    a. Se utiliza un sistema de tipos para estructurar información. En ese sistema de tipos existen diferentes tipos:
        i. Object: representa entidad bbdd.
        ii. Queries: representan las consultas de datos.
        iii. Mutations: el equivalente en REST viene a ser las peticiones POST,DELETE,UPDATE.
        iv. Subscriptions: Eventos en tiempo real que se comunican a través de un websocket.

Se estructura formando lo que se denomina **schema**, esto es el contrato que comunica al cliente cómo se va a recuperar la información. A partir de eso se realiza la autodocumentación.

Los tipos se conectan a los **resolvers** que son funciones que se ejecutan cada vez que se realiza una petición**,** que a su vez se conectan a la fuente de datos, para poder devolver al cliente los datos que está solicitando.



2. El Cliente realiza una petición con los datos que desea de la estructura.
3. El Cliente obtiene los datos solicitados, no obtiene información redundante que no ha solicitado.


Presiona
¿CÓMO FUNCIONA?
1
Describe tus datos en backend usando un schema y un sistema de tipos.
2
El cliente hace la petición de con el lenguaje de consultas GraphQL
3
El cliente obtiene exactamente lo que necesita (ni más, ni menos).
Aprende a programar para alto rendimiento con Go en
ed.team/cursos/graphql
EDteam

Aplicaciones en la actualidad:

En la actualidad si comparamos las aplicaciones web del año 2012, a las de hoy en día, podemos observar que son mucho más funcionales gracias al desarrollo de nuevas tecnologías web que ha permitido optimizar e incrementar el rendimiento. Por ello,aprovechando nuestra presentación de GraphQL,expondremos algunos ejemplos de empresas que utilizan o han utilizado GraphQL para resolver algún problema.

- **Paypal**: Uno de los problemas que presentaba PayPal, es que tenía que proporcionar un SDK compatible para cada cada uno de los desarrolladores en el lenguaje de programación que ellos utilizarán, esta tarea estaba lejos de ser trivial. Por ello en el año 2018 PayPal empezó a utilizar GraphQL como plantilla para el desarrollo de la interfaz de la aplicación y se eliminaron las múltiples SDKs por un único endpoint GraphQL para clientes externos. Este cambio resultó por tanto en un incremento en la agilidad del desarrollo.
- **Shopify:** la empresa de e-commerce canadiense en el año 2018 ofertó una variante de la API REST que ofrecían a desarrolladores externos. Este cambio provocó en la compañía que se adoptará masivamente la API GraphQL. La empresa canadienese siempre se enfrentaba a problemas con el mapeo de datos ya que, siempre que se realizaba un cambio en el API REST lo que ocurría es que a los clientes de esa API les tocaba actualizar el código, esto lo que provocaría es una desincronización constante entre los desarrolladores de la API REST y los clientes que la consumían. Por ello la utilización de GraphQL ha resuelto el problema con el fuerte tipado.
- **Airbnb**: la adopción de GraphQL en la plataforma estadounidense de alquiler vacacional resultó en mejorar 10 veces la escalabilidad. Esto se observó durante el desarrollo de una aplicación de test para probar la seguridad de tipos GraphQL a la hora de recuperar datos. Lo que hicieron los ingenieros fue establecer un backend GraphQL con un frontend TypeScript. Esto resultó en una mezcla de seguridad de tipos y eficiencia que llevó a una migración entre los servicios de la compañía. Lo que significaba reemplazar las peticiones largas API REST por las equivalentes en GraphQL. Como consecuencia esto más algunos cambios más permite a Airbnb que el uso de la caché tenga mayor efecto a la hora de servir páginas, esto conlleva que la generación de contenido es más rápida y positiva desde el punto de vista del usuario.
- **GitHub**: la plataforma para alojar proyectos utilizando el sistema de control de versiones Git, comparte un hilo común con otras empresas de la industria. La principal razón por la que se realizó este cambio fue para que los usuarios pudieran solicitar únicamente los datos que necesitan, ya que habían situaciones en las que había que realizar 2 o 3 peticiones para obtener todo un recurso. Además la API que tenían estaba diseñada para ser hypermedia-driven, con lo que esto conlleva en el número de peticiones elevado en la comunicación que se establece entre el cliente y el servidor(escalabilidad).

Demo:

A continuación se muestran los enlaces correspondientes a una demostración de GraphQL utilizando Python elaborada por los miembros del grupo.
Vídeo: Demo GraphQL
Materiales: demo_GraphQL.zip

Fuentes:

Introducción y tipos de API:¿Qué es una API? Definición, tipos y ejemplos (hubspot.es)

Descripción API GraphQL: GraphQL | A query language for your API
What is GraphQL? (redhat.com)

Evolución de la API GraphQL→https://chat.openai.com/

Ejemplo de aplicaciones en la actualidad→6 Examples of GraphQL in Production at Large Companies | Nordic APIs |

Funcionamiento API GraphQL →https://www.youtube.com/watch?v=RreRD41qlpw