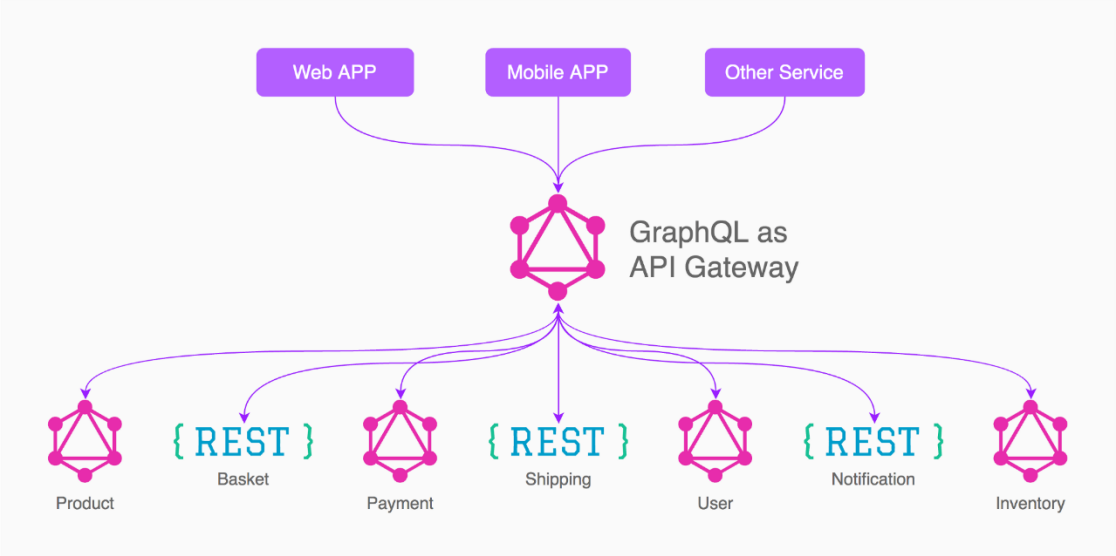
Ing. Luis Guillermo Molero Suárez



GraphQL es un lenguaje de consulta y un tiempo de ejecución del lado del servidor para interfaces de programación de aplicaciones (API) que prioriza brindar a los clientes exactamente los datos que solicitan y nada más.

GraphQL está diseñado para hacer que las API sean rápidas, flexibles y fáciles de usar para los desarrolladores. Incluso se puede implementar dentro de un entorno de desarrollo integrado (IDE) conocido como GraphiQL. Como alternativa a REST, GraphQL permite a los desarrolladores construir solicitudes que extraen datos de múltiples fuentes de datos en una sola llamada a la API.

Además, GraphQL brinda a los mantenedores de API la flexibilidad de agregar o desaprobar campos sin afectar las consultas existentes. Los desarrolladores pueden crear API con los métodos que prefieran, y la especificación GraphQL garantizará que funcionen de manera predecible para los clientes.

7 claves para una gestión de API eficaz

Esquemas, solucionadores y otros términos comunes de GraphQL
Los desarrolladores de API usan GraphQL para crear un esquema que describa todos los datos posibles que los clientes pueden consultar a través de ese servicio.

Un esquema GraphQL se compone de tipos de objetos, que definen qué tipo de objeto puede solicitar y qué campos tiene.

A medida que llegan las consultas, GraphQL valida las consultas con el esquema. GraphQL luego ejecuta las consultas validadas.

El desarrollador de API adjunta cada campo en un esquema a una función llamada resolutor. Durante la ejecución, se llama al resolutor para producir el valor.

Además de definir y validar la sintaxis para las consultas de API (descritas en el repositorio de especificaciones de graphql), GraphQL deja la mayoría de las otras decisiones al diseñador de API. GraphQL no proporciona ninguna dirección sobre cómo almacenar datos o qué lenguaje de programación usar; los desarrolladores pueden usar PHP (graphql-php), Scala (Sangria), Python (Graphene Python), Ruby (graphql-ruby), JavaScript (graphql. js) y más. GraphQL no ofrece requisitos para la red, autorización o paginación.

Desde el punto de vista del cliente, es probable que las operaciones GraphQL más comunes sean consultas y mutaciones. Si tuviéramos que pensar en ellos en términos del modelo de creación, lectura, actualización y eliminación (CRUD), una consulta equivaldría a leer. Todos los demás (crear, actualizar y eliminar) son manejados por mutaciones.

Mejores prácticas de diseño de API

Ventajas y desventajas de GraphQL en entornos corporativos

¿Está pensando en probar GraphQL en un entorno empresarial o empresarial?

Viene con pros y contras.

Ventajas

- Un esquema GraphQL establece una única fuente de verdad en una aplicación GraphQL. Ofrece a una organización una forma de federar toda su API.
- Las llamadas GraphQL se manejan en un solo viaje de ida y vuelta. Los clientes obtienen lo que solicitan sin sobrecarga.
- Los tipos de datos bien definidos reducen la falta de comunicación entre el cliente y el servidor.
- GraphQL es introspectivo. Un cliente puede solicitar una lista de tipos de datos disponibles. Esto es ideal para la generación automática de documentación.
- GraphQL permite que la API de una aplicación evolucione sin interrumpir las consultas existentes.
- Muchas extensiones GraphQL de código abierto están disponibles para ofrecer funciones que no están disponibles con las API REST.
- GraphQL no dicta una arquitectura de aplicación específica. Puede introducirse sobre una API REST existente y puede funcionar con herramientas de gestión de API existentes.
- Desventajas
- GraphQL presenta una curva de aprendizaje para desarrolladores familiarizados con las API REST.
- GraphQL traslada gran parte del trabajo de una consulta de datos al lado del servidor, lo que agrega complejidad para los desarrolladores de

servidores.

- Dependiendo de cómo se implemente, GraphQL puede requerir diferentes estrategias de administración de API que las API REST, particularmente cuando se consideran límites de tarifas y precios.
- El almacenamiento en caché es más complejo que con REST.
- Los mantenedores de API tienen la tarea adicional de escribir un esquema GraphQL que se pueda mantener.

Consejos para incorporar API en toda su empresa

Un ejemplo de consulta GraphQL

La mejor manera de apreciar GraphQL es mirar algunas consultas y respuestas de muestra. Veamos tres ejemplos adaptados del sitio web del proyecto GraphQL, graphql.org.

El primer ejemplo muestra cómo un cliente puede construir una consulta GraphQL, pidiendo a una API que devuelva campos específicos en una forma que usted haya especificado.

```
{
  me {
    name
  }
}
```

Una API GraphQL devolvería un resultado como este en formato JSON:

```
{
  "me": {
    "name": "Dorothy"
  }
}
```

Un cliente también puede pasar argumentos como parte de una consulta GraphQL, como se ve en este ejemplo:

```
{
```

```
  human(id: "1000") {
    name
    location
  }
}
```

El resultado:

```
{
  "data": {
    "human": {
      "name": "Dorothy,
      "location": "Kansas"
    }
  }
}
```

A partir de aquí, las cosas se ponen más interesantes. GraphQL brinda a los usuarios la capacidad de definir fragmentos reutilizables y asignar variables. Suponga que necesita solicitar una lista de ID y luego solicitar una serie de registros para cada ID. Con GraphQL, puede construir una consulta que extraiga todo lo que desee con una sola llamada a la API.
Entonces esta consulta:

```
query HeroComparison($first: Int = 3) {
  leftComparison: hero(location: KANSAS) {
    ...comparisonFields
  }
  rightComparison: hero(location: OZ) {
    ...comparisonFields
  }
}

fragment comparisonFields on Character {
  name
  friendsConnection(first: $first) {
    totalCount
    edges {
      node {
        name
      }
    }
  }
}
```

Podría producir este resultado:


```
{
  "data": {
    "leftComparison": {
      "name": "Dorothy",
      "friendsConnection": {
        "totalCount": 4,
        "edges": [
          {
            "node": {
              "name": "Aunt Em"
            }
          },
          {
            "node": {
              "name": "Uncle Henry"
            }
          },
          {
            "node": {
              "name": "Toto"
            }
          }
        ]
      }
    },
    "rightComparison": {
      "name": "Wizard",
      "friendsConnection": {
        "totalCount": 3,
        "edges": [
          {
            "node": {
              "name": "Scarecrow"
            }
          },
          {
            "node": {
              "name": "Tin Man"
            }
          },
          {
            "node": {
              "name": "Lion"
            }
          }
        ]
      }
    }
  }
}
```

Si eres un usuario de GitHub, una forma rápida de obtener una experiencia práctica con GraphQL es con GraphQL Explorer de GitHub.

GraphQL y código abierto

GraphQL fue desarrollado por Facebook, que comenzó a usarlo para aplicaciones móviles en 2012. La especificación GraphQL fue de código abierto en 2015. Ahora está supervisada por GraphQL Foundation.
Hay una variedad de proyectos de código abierto que involucran GraphQL. La siguiente lista no es exhaustiva, pero incluye proyectos diseñados para facilitar la adopción de GraphQL.

- Apollo, una plataforma GraphQL que incluye una biblioteca de cliente frontend (Apollo Client) y una estructura de servidor backend (Apollo Server).
- Offix, un cliente fuera de línea que permite que las mutaciones y consultas de GraphQL se ejecuten incluso cuando una aplicación es inaccesible.
- Graphback, un cliente de línea de comandos para generar servidores Node.js habilitados para GraphQL.
- OpenAPI-to-GraphQL, una interfaz de línea de comandos y biblioteca para traducir las API descritas por las especificaciones de OpenAPI o Swagger en GraphQL.