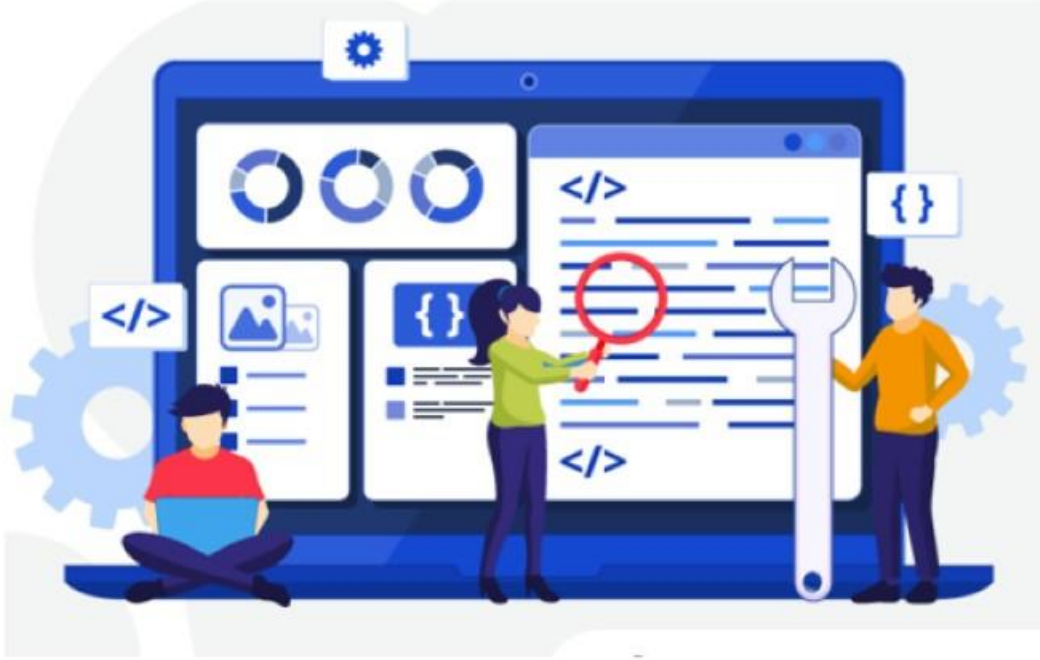
Introducción a:

GraphQL

React.js 2015

Diego Iván Oliveros Acosta



16/11/2021

Agenda

- **GraphQL**
- Arquitectura global en GraphQL
- ¿Cómo opera GraphQL?
- ¿Las mutaciones en GraphQL?
- ¿Las validaciones en GraphQL?
- Schemas en GraphQL
- Subscription
- Lenguaje de consulta para la API con GraphQL

GraphQL

Describe your data

```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

GraphQL

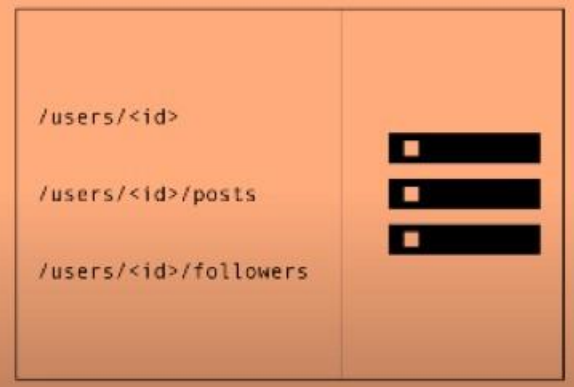
- GraphQL es un protocolo de consulta de datos, originalmente creado por Facebook para uso interno, pero fue liberado como open source desde 2015 y esto produjo grandes ventajas y beneficios prácticos.

Algunas otras características de GraphQL son:

- Es un protocolo de consultas de datos.
- Es una interfaz entre cliente (pide datos) y servidor (manipula datos).
- Es independiente de la plataforma y del almacenamiento de datos.
- Su tráfico se basa en HTTP.
- La forma de accederlo es a través de un Endpoint como /graphql.

Alternativa tradicional
vs
GraphQL

3 API endpoints



GraphQL

- Las aplicaciones en general se apoyan en protocolos de consulta de datos para obtener toda clase de reportes y el más utilizado es REST, pero GraphQL se presenta como una gran **alternativa** para las necesidades actuales.
- Los objetivos esenciales de GraphQL son:
  - Ofrecer a los clientes una forma más directa, sencilla y eficiente de obtener exactamente los datos que requiere.
  - Utilizar su **protocolo** potente y dinámico.
  - *Evitar las múltiples consultas al servidor.*

GraphQL

- **Lenguaje tipado y validable.** Le damos una forma de lo que recibe y lo que devolvemos, además de agregarle seguridad.
- El cliente define qué recibe. Hace una consulta de la estructura, que se define como respuesta.
- Envía lo necesario. Se tiene control total de las respuestas que se esperan del servidor.
- Hace un solo request por vista. Se maneja un sola fila, prácticamente en solo request se puede mandar todo lo que se necesita.

GraphQL

- Es implementado en la principales empresas tecnológicas alrededor del mundo.

https://www.howtographql.com/basics/0-introduction/
https://landscape.graphql.org/images/landscape.png

GraphQL Landscape

2021-11-16T06:17:16Z c038c7b

See the interactive landscape at l.graphql.org

Greyed logos are not open source

Spec

Tools

Services

Databases

Projects

GraphQL

GraphQL Foundation

GraphiQL

GraphQL Foundation

Services

base

Tyk

Databases

ArangoDB

Dgraph

EDGE DB

FAUNA

HASURA

neo4j

Weaviate

yugabyteDB

GraphQL Adopter

GraphQL Adopter

Istdibs

20 minutes

99designs

aftership

airbnb

ALLOCINE

Appier

ArangoDB

ART SY

Atlassian

attendify

Audi

BetterPT

Braintree

Buildkite

bynder

cheddar

cherre

Compara

Conduit

coursera

curio

dailymotion

bazinga!

Drift

DueDil

ediket

eventOne

Fairfax Media

FILEJET

GetNinjas

GitHub

Gealify

Rakuten

HIJUP

hudl

INSTANA

intuit

KLM

Kyma

Loggi

lyft

Magento

Nash Tech.

NBC

New Relic

one medical

PayPal

Pinterest

SALSIFY

satispay

ScreenCloud

shopify

stackshare

SUPPLY

swapcard

THE GUILD

The New York Times

Twitter

Vazco

wayfair

Workspot

yelp

General

GraphQL Foundation Member

airbnb

aws

Apollo

Atlassian

Dgraph

Elementl

expedia group

Facebook

FAUNA

Gatsby

Goldman Sachs

GraphCDN

graphcms

HASURA

IBM

intuit

neo4j

PayPal

Prisma

saleor

SALSIFY

shopify

THE GUILD

Twitter

l.graphql.org

GraphQL Landscape

GraphQL Foundation

This landscape is intended as a map to explore the GraphQL community, and also shows the member organizations of the GraphQL Foundation.

GraphQL

- Posee gran compatibilidad
- GraphQL no es un librería, tampoco es un framework.
- Es un *"lenguaje de consulta"* que puede ser usado en diferentes códigos de programación, como:

  • JavaScript • Python • Ruby • Java • C# • Scala • Go • PHP

GraphQL

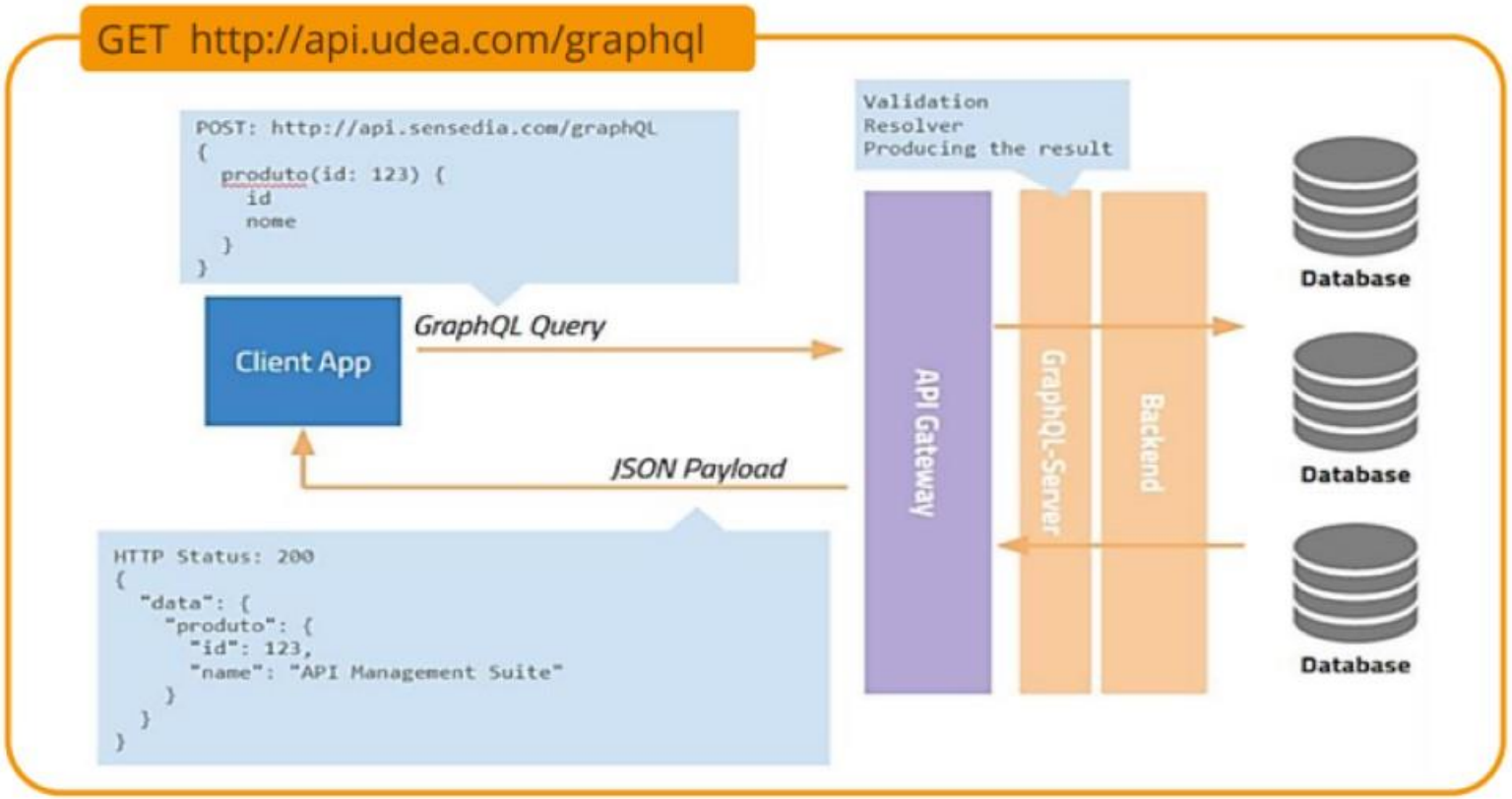
GraphQL está diseñado para ordenar la manipulación de API a través de un protocolo bien definido que es fácil de aprender.

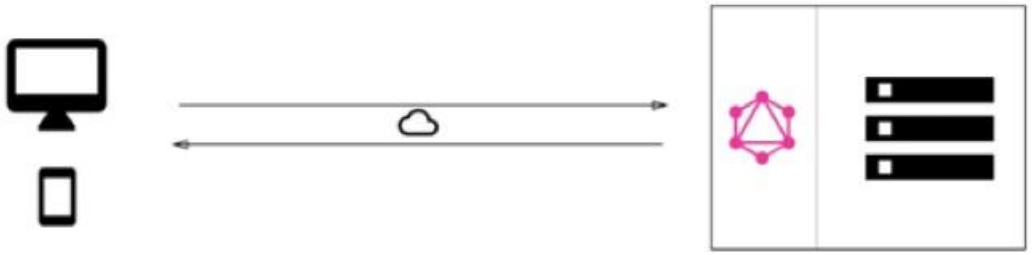
Reduce los costos de desarrollo.

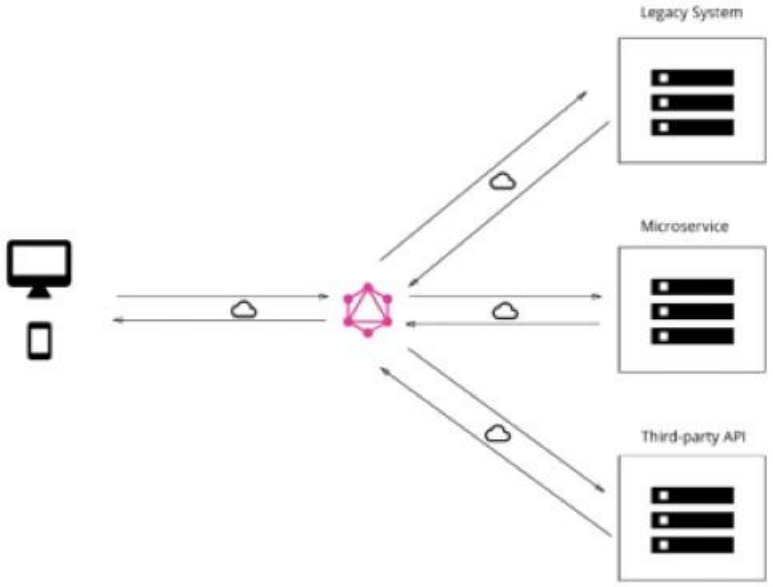
Acelera el proceso de generación de código.

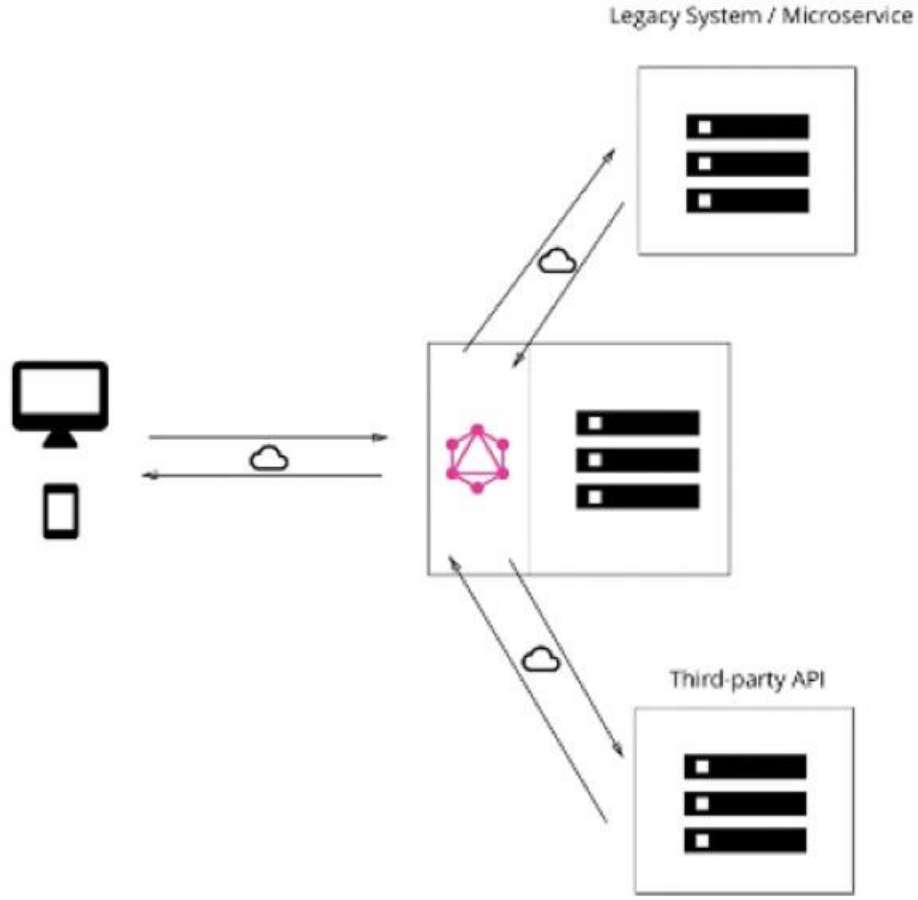
Ofrece mayor productividad.

Arquitectura global en GraphQL






Legacy System
Microservice
Third-party API


Legacy System / Microservice
Third-party API

¿Cómo opera GraphQL?

En GraphQL hay tres operaciones principales o Querys:

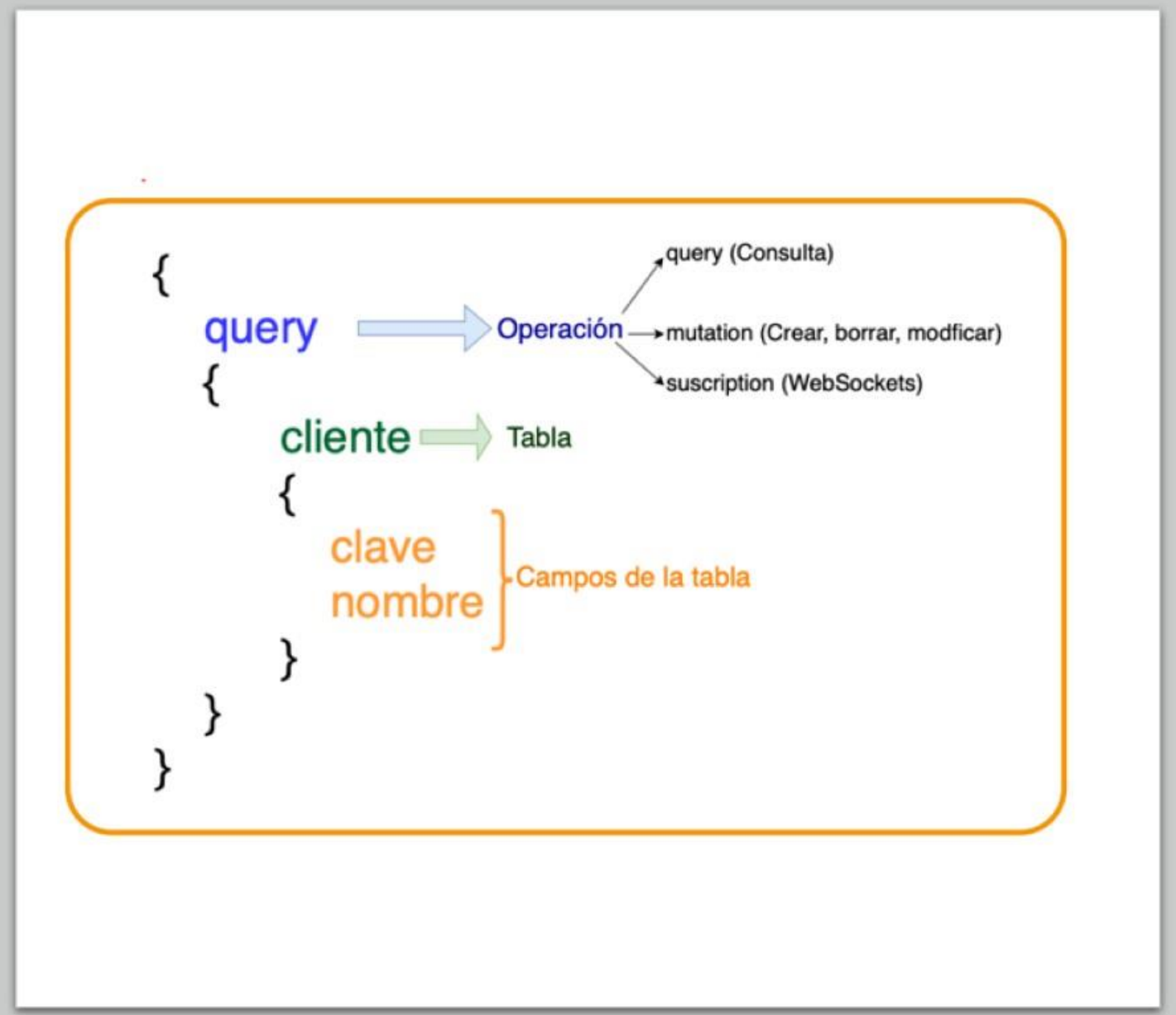
LAS CONSULTAS A LAS API

LAS MUTACIONES (FUNCIONES QUE REALIZAN MODIFICACIONES Y ENVÍAN UN RESPUESTA)

LAS SUSCRIPCIONES (PERMITE RECIBIR LOS DATOS DE LOS CAMBIOS EN EL SERVIDOR)

¿Cómo opera GraphQL?

- **GraphQL** está diseñada para generar resultados que son fácilmente interpretados por humanos.
- Simplifica las consultas porque GraphQL utiliza una solicitud (request) única para acceder a la información así esté en diferentes tablas o bases de datos.



¿Cómo opera GraphQL?

- **Queries** = leer datos del servidor (por lo general extraídos de una DB)
  - **Mutations** = crear / eliminar / modificar datos del servidor

```
query {
  getMovies {
    name
    year
    score
    actors {
      name
      age
      country
    }
    comments {
      user
      commentary
      timestamp
    }
  }
}
```

```
mutation {
  addClient(
    integration_name: "test",
    grant_type: "client_credentials",
    permissions: ["read"],
  ){
    id
    integration_url
    integration_name
    client_secret
    redirect_uri
    user_id
    grant_type
    permissions
  }
}
```

¿Las mutaciones en GraphQL?

Request

```
mutation {
  CreatePerson(name: "Daniel Varanda") {
    id
    name
  }
}
```

Response - JSON e dados solicitados

```
{
  "data": {
    "CreatePerson": {
      "id": "123",
      "name": "Daniel Varanda"
    }
  }
}
```

Request

```
mutation {
  DeletePerson(id: "123")
}
```

Response - JSON e dados solicitados

```
{
  "data": {
    "DeletePerson": {
      "deleted": true
    }
  }
}
```

¿Las validaciones en GraphQL?

Request

```
mutation {
  UpdatePerson(id: "123")
}
```

Response - JSON e dados solicitados

```
{
  "data": {
    "UpdatePerson": null
  },
  "errors": [
    {
      "message": "Person not found",
      "path": [
        "UpdatePerson"
      ],
    }
  ]
}
```

Schemas en GraphQL

- El manejo de esquemas dentro de GraphQL facilita la forma en la que se deciden las entidades.
- Definen cómo se relacionan entre ellas y cuáles son las entidades que están disponibles para cada cliente.

```
query {
  User(id: "er3tg439frjw") {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

```
{
  "data": {
    "User": {
      "name": "Mary",
      "posts": [
        { title: "Learn GraphQL today" },
        { title: "React & GraphQL - A declarative love story" }
        { title: "Why GraphQL is better than REST" }
        { title: "Relay vs Apollo - GraphQL Clients" }
      ],
      "followers": [
        { name: "John" },
        { name: "Alice" },
        { name: "Sarah" },
      ]
    }
  }
}
```

Scalars en GraphQL

Los schemas están compuestos de types, los cuales se conocen como **scalars**:

- Int: números enteros
- Float: números con decimales
- String: cadenas de texto
- Boolean: maneja los valores True o False
- ID: identificador único (puede ser de tipo Int o String)

Schemas en GraphQL

- Un problema común para los desarrolladores es cuando se tiene más de un cliente para un mismo proyecto.
- Por ejemplo, una estructura para iOS, otra para Android y otra para la web.
- Sin embargo, gracias a la compatibilidad y capacidad multisistema de GraphQL es posible conectar información y comunicar resultados desde diferentes entornos.

Schemas en GraphQL

- La unión de esquemas, o “Schema stitching”, de GraphQL permite conectar y combinar múltiples API para formar una sola interfaz en el backend.
- Es muy útil cuando queremos conectarnos con API de terceros, como Github, y podemos delegar o combinar información y extraer datos de todas estas interfaces.

Limitaciones

Profundidad máxima de consulta

Limitación basada en la complejidad de la consulta

Limitación basada en la hora del servidor

```
query {                 # Depth: 0
  me {                  # Depth: 1
    friends {           # Depth: 2
      friends {         # Depth: 3
        friends {       # Depth: 4
          friends {     # Depth: 5
            name        # Depth: 6
          }
        }
      }
    }
  }
}
```

Schemas en GraphQL

• Describir nuestros modelos • Relacionar entre los modelos • Exponer sus campos / datos

Types

- Int
- Float
- String
- Boolean
- Object

Campo obligatorio

- Int!
- Float!
- String!
- Boolean!
- Object!

ARREGLOS

- [Int]
- [Float]
- [String]
- [Boolean]
- [Object]

- [Int]!
- [Float]!
- [String]!
- [Boolean]!
- [Object]!

Schemas en GraphQL

Adding a relation

```
type Person {
  name: String!
  age: Int!
  posts: [Post!]!
}
```

```
type Post {
  title: String!
  author: Person!
}
```


```
{
  hero {
    name
    friends {
      name
      homeWorld {
        name
        climate
      }
      species {
        name
        lifespan
        origin {
          name
        }
      }
    }
  }
}
```

```
type Query {
  hero: Character
}

type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
  species: Species
}

type Planet {
  name: String
  climate: String
}

type Species {
  name: String
  lifespan: Int
  origin: Planet
}
```

GraphQL Subscriptions

Una parte importante de GraphQL son las suscripciones.

Son relativamente nuevas en las librerías para los servidores.

Simplemente agregan un manejador que notifica a quien se haya suscrito a nuestro servidor (a través de conexiones WebSockets) a mantener una interacción RealTime.

Subscription

- ¿Quién le ha dado like a nuestras fotos? (similar a Instagram)

```
subscription {
 notificationOf("type": "like"){
  id
  type
  data {
   photo {
    id
    name
   }
   user {
    id
    avatar
    name
   }
   date
  }
 }
}
```

Respuesta

```
{
 "data":{
  "id": 45,
  "type": "like",
  "data": {
   "photo": {
    "id": 76376482,
    "name": "photos_news"
   },
   "user": {
    "id": 767,
    "name": "Tristan O'connors"
   },
   "date": "2020-04-05T16:40:46"
  }
 }
}
```

Lenguaje de consulta para la API con GraphQL

- GraphQL es un **lenguaje de consulta para API** y un tiempo de ejecución para completar esas consultas con sus datos existentes.
- GraphQL proporciona una descripción completa y comprensible de los datos en su API y brinda a los clientes el poder de pedir exactamente lo que necesitan.
- Las consultas GraphQL siempre devuelven resultados predecibles.
- Las aplicaciones que usan GraphQL son rápidas y estables porque controlan los datos que obtienen, no el servidor.

Lenguaje de consulta **para** la API con GraphQL

- Facilita la evolución de las API a lo largo del tiempo y habilita poderosas herramientas para desarrolladores.
- Las API de GraphQL están organizadas en términos de **tipos y campos**, no de puntos finales.
- Accede a todas las capacidades de sus datos desde un único punto final.
- GraphQL usa tipos para garantizar que las aplicaciones solo soliciten lo que sea posible y proporcionen errores claros y útiles.
- Las aplicaciones pueden usar tipos para evitar escribir código de análisis manual.

Lenguaje de consulta para la API con GraphQL

- Un ejemplo de un Query GraphQL puede ser:

Describa sus datos

```
type Project {
  name: String
  tagline: String
  contributors: [user]
}
```

Pregunte lo que quiera

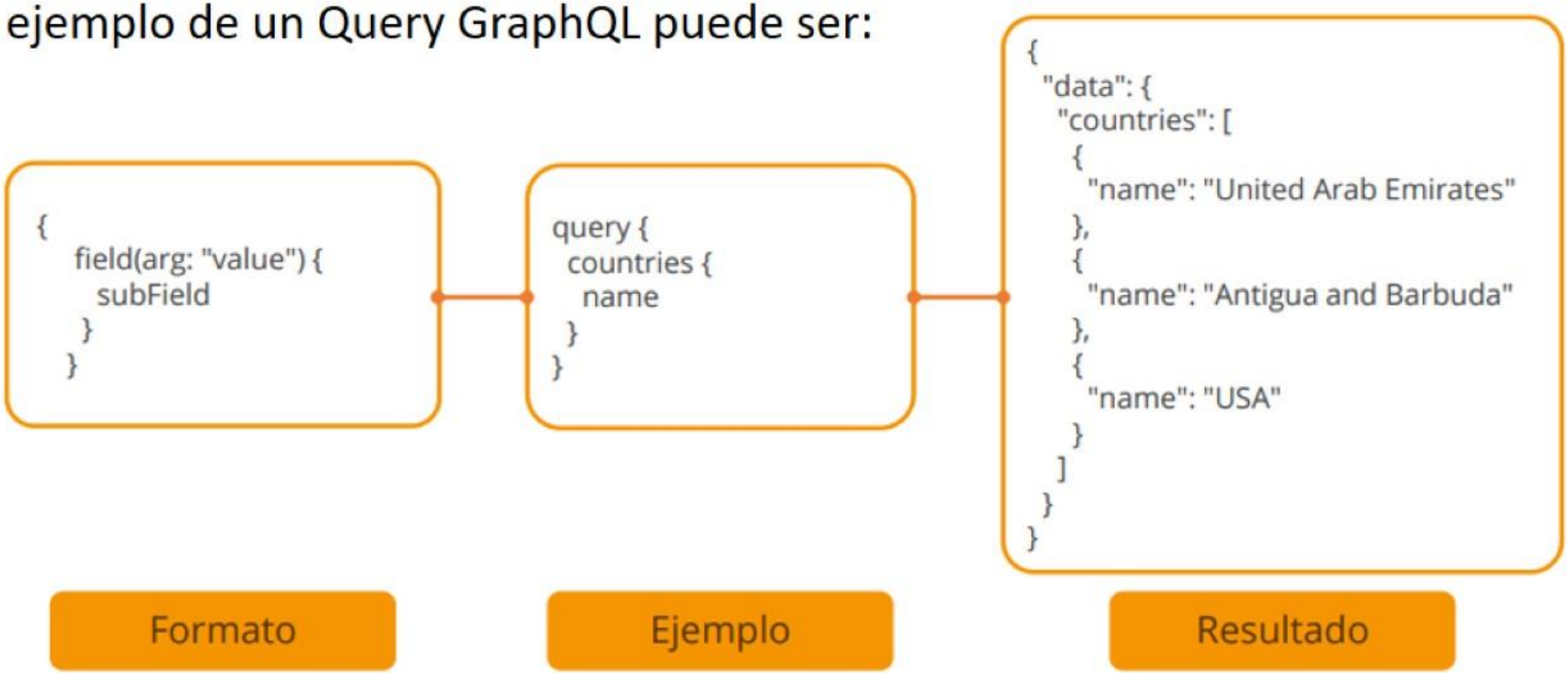
```
{
Project (name: "Graphgl) {
   tagline
 }
}
```

Obtenga el resultado

```
{
"project":  {
   "tagline": "A query languaje for APIs"
 }
}
```

Lenguaje de consulta para la API con GraphQL

- Otro ejemplo de un Query GraphQL puede ser:



```
type Query {
  allPersons(last: Int): [Person!]!
  allPosts(last: Int): [Post!]!
}

type Mutation {
  createPerson(name: String!, age: String!): Person!
  updatePerson(id: ID!, name: String!, age: String!): Person!
  deletePerson(id: ID!): Person!
  createPost(title: String!): Post!
  updatePost(id: ID!, title: String!): Post!
  deletePost(id: ID!): Post!
}

type Subscription {
  newPerson: Person!
  updatedPerson: Person!
  deletedPerson: Person!
  newPost: Post!
  updatedPost: Post!
  deletedPost: Post!
}
```

```
type Person {
  id: ID!
  name: String!
  age: Int!
  posts: [Post!]!
}

type Post {
  id: ID!
  title: String!
  author: Person!
}
```

Resumen

- **GraphQL**
- Arquitectura global en GraphQL
- ¿Cómo opera GraphQL?
- ¿Las mutaciones en GraphQL?
- ¿Las validaciones en GraphQL?
- Schemas en GraphQL
- Subscription
- Lenguaje de consulta para la API con GraphQL


Gracias
@scalapp.co

Referencias:

- Facultad de Ingeniería, Ingenia@, Formación por ciclos: Desarrollo de Aplicaciones web, ciclo 4, Introducción a GraphQl, Universidad de Antioquia,
- https://lms.misiontic2022udea.com/pluginfile.php/81122/mod_resource/content/10/2021_0017_DW_Semana3_MinTic_GraphQL%20Introduccion_V6_Andres.pdf
- https://graphql.org/
- **https://graphql.org/learn/**
- https://www.howtographql.com/

Referencias

Conferences

Community Resources

Exploring GraphQL: A Query Language for APIs

https://www.edx.org/course/exploring-graphql-a-query-language-for-apis

THE LINUX FOUNDATION

Exploring GraphQL: A Query Language for APIs

Learn about GraphQL, an alternative to REST, and practice GraphQL queries in an interactive playground.

Estimated 7 weeks
1–2 hours per week

Self-paced
Progress at your own speed

Free
Optional upgrade available