Referencia Rápida de GraphQL

Esquemas, queries, mutations, tipos, fragments

Definición de esquema

Raíz del esquema

```
schema {
  query: Query
  mutation: Mutation
}
```

Tipo objeto

```
type User {
  id: ID!
  name: String!
  email: String
}
```

Queries

Query básica

```
query {
  user(id: "1") {
    name
    email
  }
}
```

Query nombrada con alias

```
query GetUsers {
  admin: user(role: ADMIN) { name }
  guest: user(role: GUEST) { name }
}
```

Mutations

Mutation básica

```
mutation {
  createUser(input: { name: "Alice" }) {
    id
    name
  }
}
```

Tipos de entrada

```
input CreateUserInput {
  name: String!
  email: String
}
```

Subscriptions

Subscription básica

```
subscription {
  messageAdded(channel: "general") {
    text
    sender { name }
  }
}
```

Resumen

`subscription`	Datos en tiempo real vía WebSocket
`Server push`	El servidor envía actualizaciones al cliente
`Single field`	Solo un campo raíz por subscription

Tipos

Tipos escalares

`Int`	Entero con signo de 32 bits
`Float`	Punto flotante de doble precisión
`String`	Secuencia de caracteres UTF-8
`Boolean`	true o false
`ID`	Identificador único (serializado como String)

Modificadores de tipo

`String`	Cadena nullable
`String!`	Cadena no nula
`[String]`	Lista nullable de cadenas nullables
`[String!]!`	Lista no nula de cadenas no nulas

Enum y Union

```
enum Role { ADMIN USER GUEST }
union SearchResult = User | Post
interface Node { id: ID! }
```

Argumentos y variables

Variables

```
query GetUser($id: ID!) {
  user(id: $id) {
    name
  }
}
# Variables: { "id": "123" }
```

Valores por defecto

```
query GetUsers($limit: Int = 10) {
  users(limit: $limit) { name }
}
```

Fragments

Fragment nombrado

```
fragment UserFields on User {
  id
  name
  email
}
```

Usar fragments

```
query {
  user(id: "1") { ...UserFields }
  admin: user(id: "2") { ...UserFields }
}
```

Fragment en línea

```
query {
  search(text: "a") {
    ... on User { name }
    ... on Post { title }
  }
}
```

Directivas

Directivas integradas

`@include(if: Boolean!)`	Incluir campo solo cuando la condición es verdadera
`@skip(if: Boolean!)`	Omitir campo cuando la condición es verdadera
`@deprecated(reason: String)`	Marcar campo como obsoleto

Ejemplo de uso

```
query GetUser($withEmail: Boolean!) {
  user(id: "1") {
    name
    email @include(if: $withEmail)
  }
}
```

Introspección

Introspección de tipo

```
query {
  __type(name: "User") {
    name
    fields { name type { name } }
  }
}
```

Introspección de esquema

```
query {
  __schema {
    types { name kind }
    queryType { name }
  }
}
```

Campos de introspección

`__schema`	Consultar los tipos y directivas del esquema
`__type(name:)`	Consultar un tipo específico por nombre
`__typename`	Devuelve el nombre del tipo de cualquier objeto

Patrones comunes

Paginación (estilo Relay)

```
query {
  users(first: 10, after: "cursor") {
    edges { node { name } cursor }
    pageInfo { hasNextPage }
  }
}
```

Manejo de errores

```
{
  "data": { "user": null },
  "errors": [{ "message": "Not found",
    "path": ["user"] }]
}
```

Buenas prácticas

`Nombrar operaciones`	Siempre nombrar queries y mutations
`Usar variables`	Nunca interpolar valores en cadenas de query
`Solicitar solo los campos necesarios`	Evitar over-fetching con selecciones precisas
`Usar fragments`	Compartir conjuntos de campos entre queries