Referência Rápida GraphQL

Schemas, queries, mutations, tipos, fragments

Definição de Schema

Raiz do Schema

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

Tipo de Objeto

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

Queries

Query Básica

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

Query Nomeada com Alias

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

Mutations

Mutation Básica

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

Tipos Input

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

Subscriptions

Subscription Básica

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

Visão Geral

`subscription`	Dados em tempo real via WebSocket
`Server push`	Servidor envia atualizações para o cliente
`Single field`	Apenas um campo raiz por subscription

Tipos

Tipos Escalares

`Int`	Inteiro de 32 bits com sinal
`Float`	Ponto flutuante de dupla precisão
`String`	Sequência de caracteres UTF-8
`Boolean`	true ou false
`ID`	Identificador único (serializado como String)

Modificadores de Tipo

`String`	String nullable
`String!`	String não-null
`[String]`	Lista nullable de strings nullable
`[String!]!`	Lista não-null de strings não-null

Enum e Union

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

Argumentos e Variáveis

Variáveis

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

Valores Padrão

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

Fragments

Fragment Nomeado

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

Usando Fragments

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

Fragment Inline

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

Diretivas

Diretivas Embutidas

`@include(if: Boolean!)`	Incluir campo somente quando condição for verdadeira
`@skip(if: Boolean!)`	Pular campo quando condição for verdadeira
`@deprecated(reason: String)`	Marcar campo como depreciado

Exemplo de Uso

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

Introspecção

Introspecção de Tipo

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

Introspecção de Schema

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

Campos de Introspecção

`__schema`	Consultar tipos e diretivas do schema
`__type(name:)`	Consultar um tipo específico por nome
`__typename`	Retorna o nome do tipo de qualquer objeto

Padrões Comuns

Paginação (estilo Relay)

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

Tratamento de Erros

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

Boas Práticas

`Name operations`	Sempre nomear queries e mutations
`Use variables`	Nunca interpolar valores em strings de query
`Request only needed fields`	Evitar over-fetching com seleções precisas
`Use fragments`	Compartilhar conjuntos de campos entre queries