RIFERIMENTO RAPIDO GRAPHQL

Schema, query, mutation, tipi, fragment

Definizione Schema

Root dello Schema

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

Tipo Oggetto

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

Query

Query Base

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

Query Nominata con Alias

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

Mutation

Mutation Base

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

Tipi Input

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

Subscription

Subscription Base

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

Panoramica

`subscription`	Dati in tempo reale via WebSocket
`Server push`	Il server invia aggiornamenti al client
`Campo singolo`	Un solo campo root per subscription

Tipi

Tipi Scalari

`Int`	Intero con segno a 32 bit
`Float`	Virgola mobile a doppia precisione
`String`	Sequenza di caratteri UTF-8
`Boolean`	true o false
`ID`	Identificatore univoco (serializzato come String)

Modificatori di Tipo

`String`	Stringa nullable
`String!`	Stringa non-null
`[String]`	Lista nullable di stringhe nullable
`[String!]!`	Lista non-null di stringhe non-null

Enum, Union e Interface

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

Argomenti e Variabili

Variabili

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

Valori Default

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

Fragment

Fragment Nominato

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

Utilizzo dei Fragment

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

Fragment Inline

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

Directive

Directive Integrate

`@include(if: Boolean!)`	Include il campo solo quando la condizione è vera
`@skip(if: Boolean!)`	Salta il campo quando la condizione è vera
`@deprecated(reason: String)`	Marca il campo come deprecato

Esempio di Utilizzo

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

Introspezione

Introspezione Tipo

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

Introspezione Schema

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

Campi di Introspezione

`__schema`	Interroga i tipi e le directive dello schema
`__type(name:)`	Interroga un tipo specifico per nome
`__typename`	Restituisce il nome del tipo di qualsiasi oggetto

Pattern Comuni

Paginazione (stile Relay)

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

Gestione Errori

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

Best Practice

`Nomina le operazioni`	Assegna sempre un nome a query e mutation
`Usa variabili`	Non interpolare mai valori nelle stringhe di query
`Richiedi solo i campi necessari`	Evita l'over-fetching con selezioni precise
`Usa i fragment`	Condividi set di campi tra le query