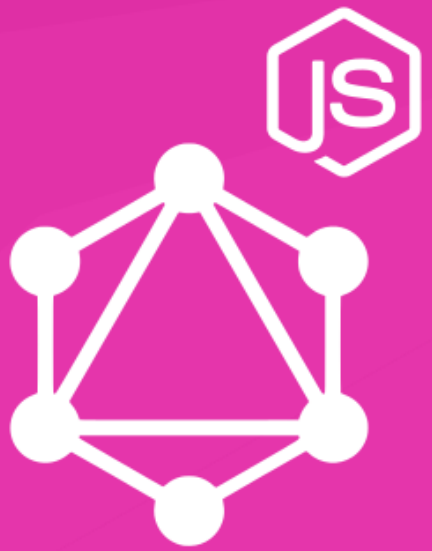


INTRODUÇÃO AO GRAPHQL

O QUE É O GRAPHQL?

É uma linguagem de consulta a dados em API's desenvolvida pelo Facebook. As consultas são interpretadas em tempo de execução no servidor usando um sistema de tipos que você define para seus dados.

Com isso facilita o processo de entregar à aplicação Client apenas o que foi requisitado pela mesma.

Não está vinculado a qualquer banco de dados ou sistema armazenamento específico.

CONCEITOS BÁSICOS

Type system: sistema de tipos que usamos para descrever nossos dados

Queries: obtém dados da nossa API (read)

Mutations: faz alterações nos dados da nossa API (write)

Schema: define o "Esquema" da nossa API, pense nele com um container para todos os tipos da nossa API (SDL)

TYPE SYSTEM

GraphQL tem seu próprio sistema de tipos para que possamos "descrever" dados para nossa API.

```
type User {
    name: String!
    email: String!
    photo: String
}
```

```
type Post {
    title: String!
    content: String!
    photo: String!
    author: User!
    comments: [ Comment! ]!
}
```

```
type Comment {
    comment: String!
    user: User!
    post: Post!
}
```

QUERIES

Queries são o que usamos para buscar dados na nossa API.
(analogia método GET do REST)
Obs: campos resolvidos paralelamente

Definição da Query

```
type Query {
    users: [ User! ]!
}
```

Requisição no Client

```
{
    query {
        users {
            name
            email
        }
    }
}
```

JSON retornado

```
{
    "data": {
        "users": [
            {
                "name": "Jon",
                "email": "jon@email.com"
            },
            ...
        ]
    }
}
```

MUTATIONS

Mutations nos permitem criar, alterar e deletar dados
(analogia ao POST, PUT e DELETE do REST)
Obs: campos resolvidos em série (um após o outro)

Definição da Mutation

```
type Mutation {
    createUser(name: String!, email: String!): User!
}
```

Requisição no Client

```
{
    mutation {
        createUser (
            name: "Dany",
            email: "dany@email.com"
        ) {
            name
        }
    }
}
```

JSON retornado

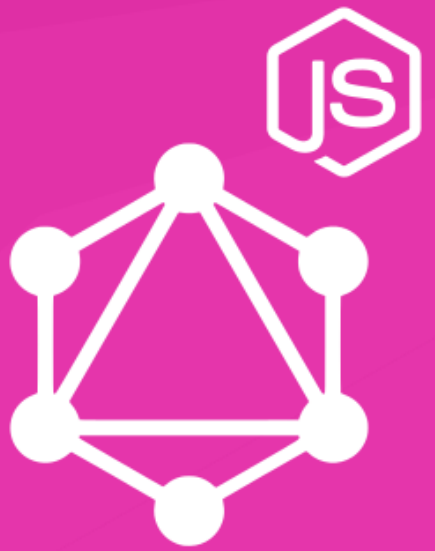
```
{
    "data": {
        "createUser": {
            "name": "Dany"
        }
    }
}
```

SCHEMA

O Schema enbloga nossas Queries, Mutations, Subscriptions, Directives, etc

Definição do Schema

```
type Schema {
    query: Query
    mutation: Mutation
}
```



EXECUÇÃO

RESOLVERS

Cada campo no GraphQL possui uma função *"Resolver"*

Query para buscar pelo id

```
type Query {
    user(id: ID!): User
}
```

Resolver assíncrono para query "user"

```
Query {
    user (parent, args, context, info) {
        return context.db.UserModel
            .findById(args.id)
    }
}
```

TRIVIAL RESOLVERS

Agora que temos o objeto User disponível, precisamos resolver seus campos também:

User

```
type User {
    name: String!
    email: String!
    photo: String
}
```

Resolvers dos campos do objeto "User"

```
User {
    name (parent, args, context, info) {
        return parent.name;
    },
    email (parent, args, context, info) {
        return parent.email;
    },
    photo (parent, args, context, info) {
        return parent.photo;
    }
}
```

SCALAR TYPES

Um objeto GraphQL possui um nome e seus campos, mas em algum momento esses campos precisam ser resolvidos com valores concretos.
É aí que entram os "Tipos Escalares": eles representam as folhas da árvore.

Int: um inteiro de 32 bits (assinado)

Float: um ponto flutuante de dupla precisão (assinado)

String: uma sequência de carateres UTF-8

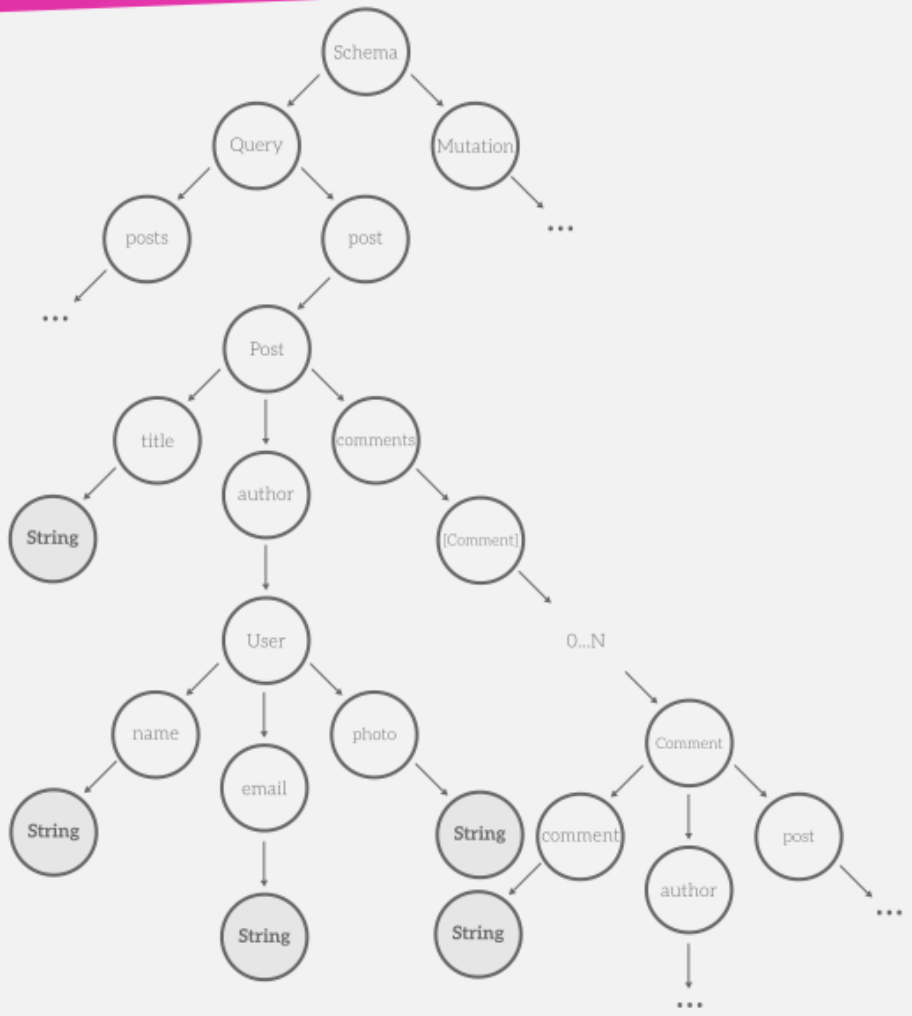
Boolean: true ou **false**

ID: Representa um identificador único, geralmente usado para rebuscar um objeto ou como chave de cache.

GRAPHQL É UMA ÁRVORE

A forma como os campos são resolvidos no GraphQL, é bem semalhante a estrutura de dados do tipo árvore:

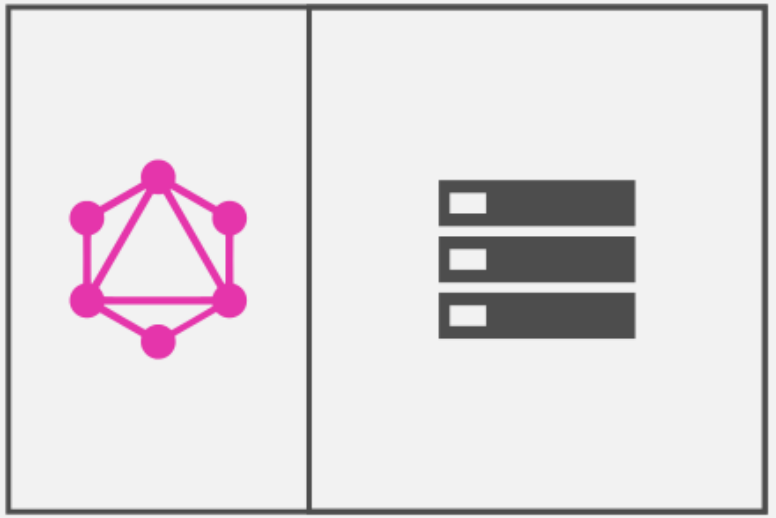
```
type Post {
    title: String!
    content: String!
    photo: String!
    author: User!
    comments: [ Comment! ]!
}
```



COMO FUNCIONA?

```
query {
    user(id: 53) {
        name
        posts {
            title
        }
        followers(last: 2) {
            name
        }
    }
}
```

HTTP POST



```
{
  "data": {
      "user": {
          "name": "Jon",
          "posts": [
              { "title": "Learn GraphQL" }
          ],
          "followers": [
              { "name": "Dany" },
              { "name": "Davos" }
          ]
      }
  }
}
```

ONDE APRENDER MAIS?

Documentação / Referências:

Documentação Oficial: **http://graphql.org**

Referência: **https://howtographql.com**


JS