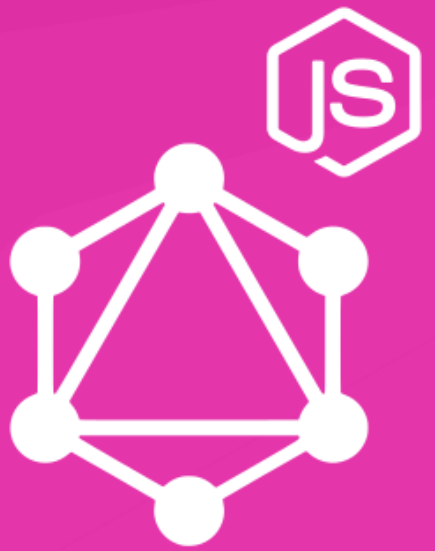

GRAPHQL É MELHOR QUE REST!

GRAPHQL WINS!

API's REST se mostraram muito inflexíveis para acompanhar as rápidas mudanças de requisitos dos Clients que as acessam.

O GraphQL foi desenvolvido para lidar com a necessidade de maior flexibilidade e eficiência! Resolve muitas das deficiências e ineficiências que os desenvolvedores experimentam ao interagir com as API's REST.

NO UNDERFETCHING

Não precisa implementar uma série de endpoints para obter os dados que precisa:

REST

```
/users/<id>
/users/<id>/posts
/users/<id>/followers
```

GraphQL

```
type User {
    name: String!
    email: String!
    posts: [ Post ]
    followers: [ User ]
}
```

NO OVERFETCHING

Adeus à sobrecarga de dados:

REST

/posts/73

```
{
    "post": {
        "title": "Learn GraphQL",
        "content": "Lorem ipsum...",
        "comments": [ ... ],
        "author": { ... }
    }
}
```

GRAPHQL

```
query {
    post(id: 73) {
        title
        author {
            name
        }
    }
}
```

```
{
    "data": {
        "post": {
            "title": "Learn GraphQL",
            "author": {
                "name": "Jon"
            }
        }
    }
}
```

RÁPIDA PROTOTIPAGEM

Um padrão comum no REST é criar endpoints de acordo com as "views" da aplicação. Porém isso torna despendioso cada mudança que precisar ser feita no front end.

Com GraphQL isso é resolvido graças a sua natureza flexível. Uma vez que o Client pode especificar exatamente os dados de que precisa, nenhum engenheiro precisa fazer ajustes no back end quando a estrutura dos dados precisa ser mudada no front end.

BENEFÍCIOS DO SCHEMA & TYPE SYSTEM

O GraphQL usa um sitema de tipagem forte para definir os recursos de uma API. Todos os tipos são expostos usando o *Schema Definition Language (SDL)*.

Isto serve como um contrato entre o *Client* e o *Server* para definir como os dados podem ser acessados.

Uma vez definido, back end e front end podem trabalhar sem comunicação, já que ambos estão conscientes da estrutura de dados existente.


JS