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.

- Permite o Client especificar exatamente os dados que precisa
- Facilita buscar dados de várias fontes (vários bancos, outras API's, sistemas legados)
- Usa um sistema de tipos para definir os dados

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)

Subscriptions: permite ouvir mudanças em "tempo real" (real-time)

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  {
   id: ID!
   name: String!
   email: String!
   password: String!
}
```

```
type Chat {
   id: ID!
   messages: [Message!]!
   users: [User!]!
   title: String
   createdAt: DateTime!
   isGroup: Boolean!
}
```

```
type Message {
   id: ID!
   text: String!
   createdAt: DateTime!
   sender: User!
   chat: Chat!
}
```

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 {
    allUsers: [ User! ]!
}
```

Requisição no Client

```
{
    query {
        allUsers {
            name
            email
        }
    }
}
```

JSON retornado

```
{
    "data": {
        "allUsers": [
            {
                "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: "Tyrion",
            email: "tyrion@email.com"
        ) {
            name
        }
    }
}
```

JSON retornado

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

SUBSCRIPTIONS

Subscriptions permitem nos "inscrever" no servidor para receber alterações em tempo real (Web Sockets)

Definição da Subscription

```
type Subscription {
    Message(
        mutation_in: [
            MutationType!
        ]
    ): Message!
}
```

Requisição no Client

```
{
    subscription {
        Message (
            mutation_in: [CREATED]
        ) {
            title
            sender {
                name
            }
        }
    }
}
```

JSON retornado

```
{
    "data": {
        "Message": {
            "title": "Hello GraphQL",
            "sender": {
                "name": "Tyrion"
            }
        }
    }
}
```

SCHEMA

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

Definição do Schema

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

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)
    }
}
```

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":

Int: um inteiro de 32 bits (assinado)

Float: um ponto flutuante de dupla precisão (assinado)

String: uma sequência de

Boolean: true ou false

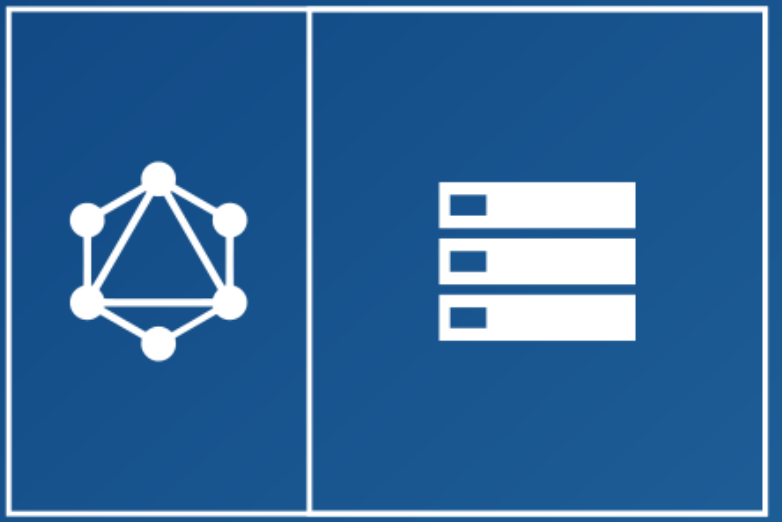
ID: Representa um identificador único, geralmente usado para rebuscar um objeto ou como chave de cache.

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, Artigos:

Documentação Oficial: http://graphql.org

Referência: https://howtographql.com

Apollo GraphQL: https://www.apollographql.com/

Blog Apollo Data: https://dev-blog.apollodata.com/