03

Para saber mais: Usando construtores

É possível que você já tenha encontrado APIs GraphQL ou trechos de código onde os tipos estão definidos de forma similar ao exemplo abaixo:

```
const graphql = require('graphql')

const tipoLivro = new graphql.GraphQLObjectType({
    name: 'Livro',
    fields: {
        id: { type: graphql.GraphQLID },
        titulo: { type: graphql.GraphQLString },
        autoria: { type: graphql.GraphQLString },
        paginas: { type: graphql.GraphQLInt },
    }
})
```

Em vez de um schema escrito em SDL (*Schema Definition Language*), o trecho de código acima usa JavaScript para criar um novo tipo `Livro`, através de uma nova instância de `GraphQLObjectType`. Podemos identificar a estrutura de campos no objeto passado como parâmetro para o construtor.

E qual a diferença entre usar a linguagem de schema (a forma como fizemos durante o projeto do curso) e criar schemas e tipos através de construtores em JavaScript (como está no exemplo acima)?

Para grande parte das APIs GraphQL, é possível definir um schema "fixo", que é carregado e não sofre alterações durante o fluxo de uso da aplicação; nesse caso utilizamos a linguagem de schema própria, que foi criada para isso.

Porém, em certas situações é útil definir "quando" o schema será construído — por exemplo, para gerar um schema GraphQL automaticamente, utilizando parâmetros vindos de outras partes de uma aplicação, como um schema de banco de dados. É possível fazer isso usando o módulo `graphql` e funções construtoras.

Você pode ver um exemplo de servidor GraphQL utilizando construtores na documentação do GraphQL (https://graphql.org/graphql-js/constructing-types/) e conferir também a lista de construtores (https://graphql.org/graphql-js/type/).