03

Para saber mais: Nexus SDL Converter

Agora que já vimos como funciona a abordagem code-first, pode ficar a dúvida: e se eu quiser refatorar uma API GraphQL já existente para utilizar as funcionalidades do Nexus? Tenho que reescrever tudo, mesmo que eu já tenha meus schemas definidos em SDL (a linguagem de query do GraphQL)?

Nesse caso, você pode usar uma ferramenta do próprio Nexus, o Nexus SDL Converter (https://nexusjs.org/converter) para converter schemas em SDL para funções. Por exemplo, inserindo o schema abaixo:

```
scalar DateTime
    type User {
        id: Int
        nome: String
        email: String
        createdAt: DateTime
        posts: [Post]
    }

    type Post {
        id: Int
        titulo: String
        conteudo: String
    }

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

COPIAR CÓDIGO

O conversor gera automaticamente o código:

```
import { objectType, scalarType } from '@nexus/schema';

const Post = objectType({
    name: "Post",
    definition(t) {
        t.int("id", { nullable: true })
        t.string("titulo", { nullable: true })
        t.string("conteudo", { nullable: true })
    }
})
const Query = objectType({
    name: "Query",
    definition(t) {
        t.field("users", {
            type: User,
            list: [false],
            nullable: true,
        })
    }
})
const User = objectType({
    name: "User",
    definition(t) {
        t.int("id", { nullable: true })
        t.string("nome", { nullable: true })
        t.string("email", { nullable: true })
        t.field("createdAt", {
                type: DateTime,
                nullable: true,
        })
        t.field("posts", {
            type: Post,
            list: [false],
```

```
            nullable: true,
        })
    }
})

const DateTime = scalarType({
    name: "DateTime",
    serialize() { /* Todo */ },
    parseValue() { /* Todo */ },
     parseLiteral() { /* Todo */ }
});
```

COPIAR CÓDIGO

No código acima, vale notar que:

- Como é normal quando usamos geradores de código, o output pode ser um pouco diferente do que faríamos "na mão" e precisar de alguns ajustes;
- Durante esta e a próxima aula, vamos organizar o *nexus schema* utilizando outros métodos, porém a estrutura vai se manter similar.
- O Nexus gerou um `scalarType()` para o tipo `DateTime`, mas deixa quem estiver desenvolvendo responsável por implementar a lógica necessária. Neste curso isso não vai ser necessário, pois o Prisma já traz o tipo `DateTime`, mas vale lembrar que o Nexus é uma ferramenta que não precisa ser utilizada necessariamente em conjunto com o Prisma. Você pode relembrar como resolvemos o tipo `DateTime` no curso de introdução ao GraphQL (https://cursos.alura.com.br/course/graphql-construindo-api-apollo-server).