05

Para saber mais: Aprofundando em DataSources

No curso anterior, utilizamos a lib `RESTDataSource` para conectar o GraphQL (através do ApolloServer) à base de dados - no caso, dois endpoints REST.

DataSources são classes que encapsulam métodos de **fetch** (além de outras ferramentas, como caching, manejo de erros e etc) para determinados serviços — no caso de RESTDataSource, os métodos da classe serviam para buscar dados disponíveis através de APIs REST. O Apollo utiliza estas classes para fazer todo o trabalho de conectar à base de dados, interpretar os métodos e fazer o **fetch** dos dados.

Além de REST, é possível utilizar dataSources customizadas, com nossos próprios métodos. Basta criarmos uma classe que herde de

`DataSource`, a classe "parent" que contém os métodos que o ApolloServer precisa.

Podemos criar uma classe para lidar com outras fontes de dados, mesmo que seja a array **hardcoded** que usamos como teste. Quer fazer o teste?

Instale a lib `apollo-datasource` no projeto e importe a classe-parent `{ DataSource }`. Em seguida, crie a classe `TurmasAPI` herdando de `DataSource`.

No construtor da classe, passamos somente a variável `db`, que nada mais é do que a variável da array de teste. Ou você pode passar no construtor as infos de acesso de qualquer outra fonte de dados.

```
const { DataSource } = require('apollo-

const db = [
 {
   id: 1,
```

```
    descricao: "básico",
    horario: "manhã"
  },
  {
    id: 2,
    descricao: "intermediário",
    horario: "tarde"
  }
]

class TurmasAPI extends DataSource {
 constructor() {
   super()
   this.db = db
 }


  // aqui vão os métodos

}

module.exports = TurmasAPI
```

COPIAR CÓDIGO

Como criar os métodos, por exemplo, para consultar os dados? Como estamos criando esses métodos do zero, tudo vai depender da base de dados! Como no caso é uma array em memória, podemos usar métodos nativos de array do JS.

```
class TurmasAPI extends DataSource {
 constructor() {
   super()
   this.db = db
 }

 async getTurmas() {
   return this.db
 }

 async getTurma(id){
   return this.db.find(turma => turma.i
 }
}
```

COPIAR CÓDIGO

Precisamos passar esse novo DataSources para o ApolloServer, então, antes de qualquer coisa,

vamos atualizar o ponto de entrada da aplicação (`./api/index.js`). Lembrando que `dataSources` é uma propriedade passada para os resolvers como **contexto** quando usamos o ApolloServer.

```
...

// importação dos outros métodos de Tur
const TurmasAPI = require('./turma/data

...

const server = new ApolloServer({
 typeDefs,
 resolvers,
 dataSources: () => {
   return {
     usersAPI: new UsersAPI(),
     turmasAPI: new TurmasAPI(),
   }
 }
})

...
```

COPIAR CÓDIGO

E refatorar os resolvers para chamarem os métodos corretos - veja aqui os resolvers recebendo `dataSources` e suas propriedades (no caso, `turmasAPI`) no parâmetro de contexto:

```
const turmaResolvers = {
 Query: {
   turmas: (_, __, { dataSources }) =>
   turma: (_, { id }, { dataSources })
 }
}
```

COPIAR CÓDIGO

Por último, você pode testar as queries `turma` e `turmas` no playground.

Lembrando que o parâmetro “context”, onde atualmente estamos passando somente a propriedade `{ dataSources }` , pode receber mais informações como dados de autenticação, caches e o que mais for passado para `ApolloServer` ao iniciarmos o servidor.

Estas informações podem ser acessadas através do método `initialize`, que é chamado por `ApolloServer`:

```
initialize(config) {
    this.context = config.context
}
```

COPIAR CÓDIGO

Da mesma forma que usamos RESTDataSource, uma lib desenvolvida pelo time do Apollo, podemos criar nossas próprias dataSources, sempre herdando da classe original DataSource. O exemplo acima é uma abordagem rápida, mas você pode se aprofundar na criação de DataSources customizadas! A documentação oficial do Apollo (https://www.apollographql.com/docs/tutorial/data-source/#building-a-custom-data-source) descreve o processo com mais detalhes e esse post no blog do Apollo (https://www.apollographql.com/blog/a-deep-dive-on-apollo-data-sources/) dá exemplos de outros DataSources (ambos em inglês).