08

Faça como eu fiz: Organizando o projeto

Neste ponto do projeto o `schema.js` já está um pouco longo. Vamos usar a abordagem de "módulos-barril" para separar e organizar melhor o código, de forma similar a que fizemos no curso pré-requisito.

1) Dentro de `src` crie uma nova pasta chamada `schemas` ;

2) Nesta pasta, crie um arquivo para cada uma das partes do schema:

- `user.js`
- `post.js`
- `review.js`
- `query.js`
- `mutation.js`

3) Crie também um arquivo `index.js` dentro da pasta `src/schemas` , que servirá de "barril" para importarmos e exportarmos os módulos do schema.

4) Em cada um dos arquivos, separe as partes do schema, sem esquecer de importar e exportar os módulos necessários:

Em `src/schemas/user.js` :

```
const { objectType } = require('@nexus/schema')

const User = objectType({
    name: 'User',
    definition(t) {
```

```
            t.model.id()
            t.model.nome()
            t.model.email()
            t.model.createdAt()
            t.model.posts()
        }
    })

    module.exports = User
```

COPIAR CÓDIGO

Em `src/schemas/post.js` :

```
    const { objectType } = require('@nexus/schema')

    const Post = objectType({
        name: 'Post',
        definition(t) {
            t.model.id()
            t.model.titulo()
            t.model.conteudo()
            t.model.publicado()
            t.model.autor()
            t.model.createdAt()
        }
    })

    module.exports = Post
```

COPIAR CÓDIGO

Em `src/schemas/review.js` :

```
const { objectType } = require('@nexus/schema')

const Review = objectType({
    name: 'Review',
    definition(t) {
        t.model.id()
        t.model.post()
        t.model.reviewer()
        t.model.nota()
        t.model.createdAt()
    }
})

module.exports = Review
```

COPIAR CÓDIGO

Em `src/schemas/query.js` :

```
const { queryType, nonNull, stringArg } = require('@nexus/schema

const Query = queryType({
    name: 'Query',
    definition(t) {
        t.crud.users({
            filtering: true,
            ordering: true
        })
        t.crud.user()
        t.crud.reviews()
        t.list.field('postsAprovados', {
            type: 'Post',
            resolve: (_, __, { prisma }) => {
                return prisma.post.findMany({
                    where: { publicado: true },
```

```
                    orderBy: {
                        createdAt: 'desc'
                    }
                })
            }
        })
        t.list.field('buscaAutoresPublicados', {
            type: 'User',
            args: {
                email: nonNull(stringArg())
            },
            resolve: (_, args, { prisma }) => {
                console.log(args)
                return prisma.user.findMany({
                    where: {
                        email: { contains: args.email },
                        posts: { some: { publicado: true } }
                    }
                })
            }
        })
    }
})

module.exports = Query
```

COPIAR CÓDIGO

Em `src/schemas/mutation.js` :

```
const { mutationType } = require('@nexus/schema')

const Mutation = mutationType({
    name: 'Mutation',
    definition(t) {
```

```
            t.crud.createOnePost()
        }
    })

    module.exports = Mutation
```

COPIAR CÓDIGO

5) Importe e exporte os módulos em `src/schemas/index.js` :

```
    const User = require('./user')
    const Post = require('./post')
    const Review = require('./review')
    const Query = require('./query')
    const Mutation = require('./mutation')

    module.exports = {
        User,
        Post,
        Review,
        Query,
        Mutation
    }
```

COPIAR CÓDIGO

6) No arquivo `src/schema.js` vai permanecer apenas o código que gera o schema final - aquele que é passado para o ApolloServer. É aqui que vamos importar o conteúdo (agora reestruturado) da pasta `src/schemas` :

```
    const { makeSchema } = require('@nexus/schema')
    const { nexusPrisma } = require('nexus-plugin-prisma')
    const path = require('path')

    const { User, Post, Review, Query, Mutation } = require('./schema
```

```
const schema = makeSchema({
    types: [User, Post, Review, Query, Mutation ],
    plugins: [nexusPrisma({ experimentalCRUD: true })],
    outputs: {
        schema: path.join(__dirname, 'schema.graphql'),
        typegen: path.join(__dirname, '../prisma/generated','nexu
    }
})

module.exports = schema
```

COPIAR CÓDIGO

7) Em `src/index.js`, o ponto de entrada da API, não precisamos modificar nada; a linha `const schema = require('./schema')` continua importando o schema gerado com `makeSchema()`.

> **Importante:** Ao reestruturar os arquivos, cuidado com os caminhos das pastas. Veja que acima foi criada a pasta `src/schemas` e, dentro dessa pasta, o arquivo `src/schemas/index.js` está concentrando todas as importações e exportações de módulos necessárias. O arquivo `src/schema.js`, onde o schema final é gerado, está fora dessa pasta, e é este arquivo que importamos para `src/index.js` (e passamos como parâmetro em `ApolloServer()`).

Caso prefira utilizar `import/export`, você pode consultar o guia de boas práticas do Nexus (https://nexusjs.org/docs/guides/best-practices) que além da organização de pastas que fizemos acima tem também algumas outras dicas.

Você pode consultar os arquivos modificados neste commit (https://github.com/alura-cursos/2036-graphql-

prisma2/commit/7936d0c33de114db44eec1d390eb9982a713c751) no repositório do curso.