07

Faça como eu fiz: Deletar e cancelar matrícula

Agora que já temos a funcionalidade principal de `matricula` definida — matricular estudantes —, podemos adicionar mutations para deletar um registro de matrícula do banco, e também para somente cancelar uma matrícula, mantendo-se o registro.

Você pode seguir o passo-a-passo abaixo ou acessar o commit com o código finalizado (https://github.com/alura-cursos/1982-graphql/commit/d7b61c87c91e016614f3ecc4700e9a0a62fa6f3d).

Deletar um registro de matrícula

Primeiro criamos a nova mutação em `./api/matricula/schema/matricula.graphql` :

```
type Mutation {
  # ... código anterior
  deletarMatricula (matricula: ID!): R
}
```

COPIAR CÓDIGO

Aqui apenas solicitamos o `id` da matrícula.

Implementando no resolver (`./api/matricula/resolvers/matriculaResolvers.js`):

```
Mutation: {
   // ... código anterior
   deletarMatricula: (_, { matricula },
dataSources.matriculasAPI.deletarMatric
},
```

COPIAR CÓDIGO

Note que, no segundo parâmetro do resolver, passamos `{ matricula }` . Isso porque *sempre* recebemos os parâmetros que vêm da query como

um objeto, onde cada propriedade representa um dos campos que definimos na mutations.

No caso, só definimos na mutation o campo `matricula: ID!`, então os parâmetros da query chegam da seguinte forma: `{ matricula: '<id passado na query>' }`.

O método `deletarMatricula()` ainda não existe na classe `MatriculasAPI`, então vamos criá-lo:

```
async deletarMatricula(idMatricula) {
  await this.db('matriculas')
  .where({ id: Number(idMatricula) })
  .del()

  this.Resposta.mensagem = "registro d
  return this.Resposta
}
```

COPIAR CÓDIGO

Usando o Knex, acessamos o banco e deletamos o registro.

Faça o teste no playground com a query:

```
mutation {
  deletarMatricula(matricula: <id de ma
    mensagem
  }
}
```

COPIAR CÓDIGO

Cancelar uma matrícula

Quando criamos o tipo `Matricula`, incluímos o campo `status: String!`, e cada novo registro recebe, por padrão, o valor "confirmado" para este campo — supondo que não faz muito sentido incluir no sistema uma matrícula cancelada. Mas deve existir a opção de cancelar uma matrícula que já esteja no banco.

Primeiro criamos mais uma mutação:

```
type Mutation {
  # ... código anterior
  cancelarMatricula (matricula: ID!):
}
```

COPIAR CÓDIGO

Aqui também recebemos somente um id de matrícula como parâmetro.

Implementando no resolver:

```
Mutation: {
  // ... código anterior
  cancelarMatricula: (_, { matricula }
dataSources.matriculasAPI.cancelarMatri
 },
```

COPIAR CÓDIGO

O método `cancelarMatricula()` ainda não existe na classe `MatriculasAPI`, então vamos criá-lo:

```
  async cancelarMatricula(idMatricula) {
    await this.db
      .update({ status: "cancelado" })
      .where({ id: Number(idMatricula) }
      .into('matriculas')

    this.Resposta.mensagem = "matrícula
    return this.Resposta
  }
```

COPIAR CÓDIGO

Usando o Knex, passamos `{ status: "cancelado" }` como parâmetro do `insert` para que somente o valor da coluna `status` seja alterado, onde `where({ id: Number(idMatricula) }) .`

Faça o teste no playground com a query:

```
mutation {
  cancelarMatricula(matricula: <id de m
    mensagem
  }
}
```

COPIAR CÓDIGO

Ainda existem outras funcionalidades relativas a matrículas que poderiam ser implementadas. Por exemplo, alterar todas as matrículas relativas a certo `id` de estudante para `status: "cancelado"` caso o cadastro de estudante seja alterado de `ativo: true` para `ativo: false`. Isso envolve um pouco mais de prática em SQL do que estamos usando nos exemplos — será que você consegue fazer essas alterações como desafio?