

CAPÍTULO 12

Associações Polimórficas

"Os negócios são o dinheiro dos outros"

— Alexandre Dumas

Nesse capítulo você verá como criar uma relação *muitos-para-muitos* para mais de um tipo de modelo.

12.1 – NOSSO PROBLEMA

O cliente pede para a equipe de desenvolvedores criar uma funcionalidade que permita aos visitantes deixar comentários sobre suas visitas aos restaurantes.

Para complicar a vida do programador, o cliente pede para permitir comentários também em qualificações, permitindo aos usuários do site justificar a nota que deram.

Esse problema poderia ser resolvido de diversas maneiras sendo que trabalharemos em cima de um modelo para representar um comentário, relacionado com restaurantes e qualificações, aproveitando para mostrar como realizar tal tipo de relacionamento.

Seria simples se pudéssemos criar mais uma tabela com o comentário em si e o `id` da entidade relacionada. O problema surge no momento de diferenciar um comentário sobre qualificação de um sobre restaurante.

Para diferenciar os comentários de restaurantes e qualificações, podemos usar um atributo de nome "tipo".

Em Ruby podemos criar apelidos para um ou mais modelos, algo similar a diversas classes implementarem determinada interface (sem métodos) em java. Podemos chamar nossos modelos `Restaurante` e `Qualificacao` como comentáveis, por exemplo.

Um exemplo dessa estrutura em Java é o caso de `Serializable` – interface que não obriga a implementação de nenhum método mas serve para marcar classes como serializáveis, sendo que diversas classes da api padrão do Java implementam a primeira.

No caso do Ruby, começamos criando um modelo chamado `Comentario`.

12.2 – ALTERANDO O BANCO DE DADOS

O conteúdo do script de migração criará as colunas "comentário", "id de quem tem o comentário", e o "tipo".

Nos campos `id` e `tipo`, colocamos o nome da coluna com o apelido seguido de `_id` e `_type`, respectivamente, notificando o Ruby que ele deve buscar tais dados daquilo que é "comentavel".

Note que no português a palavra "comentavel" soa estranho e parece esquisito trabalhar com ela, mas para seguir o padrão definido no inglês em diversas linguagens, tal apelido indica o que os modelos são capazes de fazer e, no caso, eles são "comentáveis".

O script deve então criar três colunas, sem nada de novo comparado com o que vimos até agora:

```
rails generate scaffold comentario conteudo:text \
  comentavel_id:integer comentavel_type
```

Caso seja necessário, podemos ainda adicionar índices físicos nas colunas do relacionamento, deixando a migration criada como a seguir:

```
class CreateComentarios < ActiveRecord::Migration
  def change
    create_table :comentarios do |t|
      t.text :conteudo
      t.integer :comentavel_id
      t.string :comentavel_type

      t.timestamps
    end

    add_index :comentarios, :comentavel_type
    add_index :comentarios, :comentavel_id
  end
end
```

Para trabalhar nos modelos, precisamos antes gerar a nova tabela necessária:

```
$ rake db:migrate
```

O modelo Comentario (app/models/comentario.rb) deve poder ser associado a qualquer objeto do grupo de modelos comentáveis. Qualquer objeto poderá fazer o papel de comentavel, por isso dizemos que a associação é polimórfica:

```
class Comentario < ActiveRecord::Base
  belongs_to :comentavel, polymorphic: true
end
```

A instrução :polymorphic indica a não existência de um modelo com o nome :comentavel.

Falta agora comentar que uma qualificação e um restaurante terão diversos comentários, fazendo o papel de algo comentavel. Para isso usaremos o relacionamento has_many:

```
class Qualificacao < ActiveRecord::Base
  # ...

  belongs_to :cliente
  belongs_to :restaurante

  has_many :comentarios, as: comentavel

  # ...
end
```

E o Restaurante:

```
class Restaurante < ActiveRecord::Base
  # ...

  has_many :qualificacoes
  has_many :comentarios, as: comentavel

  # ...
end
```

A tradução do texto pode ser quase literal: o modelo **TEM MUITOS** comentários **COMO** comentável.

Seus livros de tecnologia parecem do século passado?

Conheça a **Casa do Código**, uma **nova** editora, com autores de destaque no mercado, foco em **ebooks** (PDF, epub, mobi), preços **imbatíveis** e assuntos **atuais**.



Com a curadoria da **Caelum** e excelentes autores, é uma abordagem **diferente** para livros de tecnologia no Brasil. Conheça os títulos e a nova proposta, você vai gostar.

[Casa do Código, livros para o programador.](#)

12.3 – EXERCÍCIOS: CRIANDO MODELO DE COMENTÁRIO

1. Vamos criar o modelo do nosso comentário e fazer a migração para o banco de dados:

a. Vá ao Terminal

b. Digite:

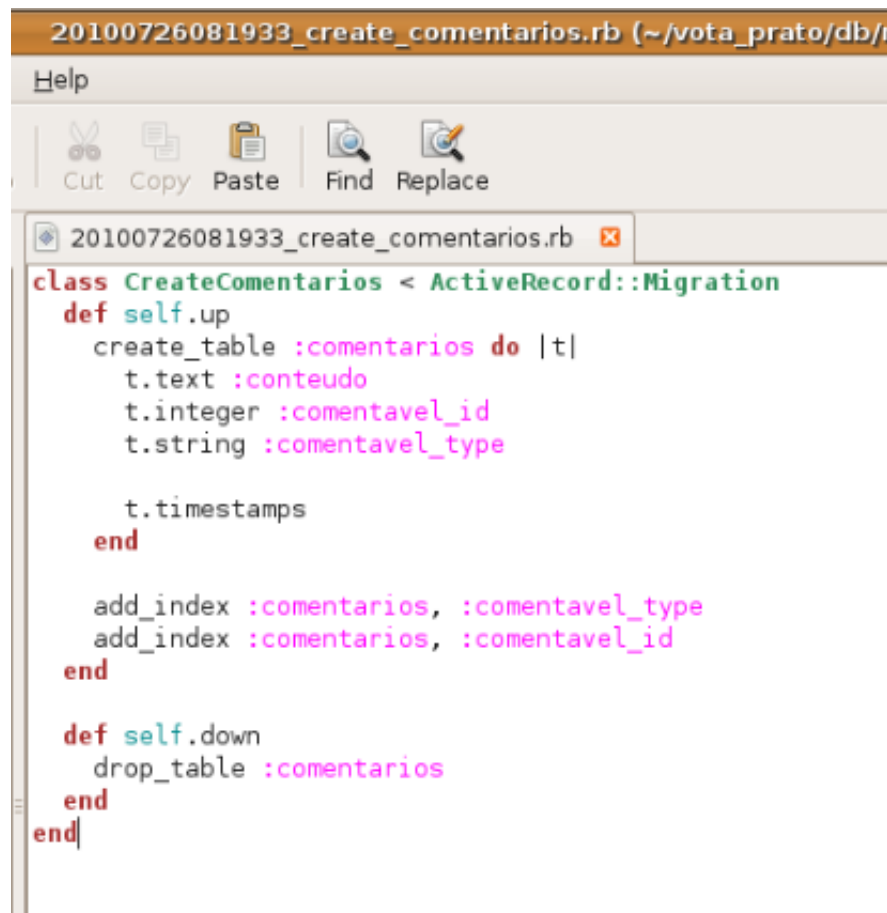
```
$ rails generate scaffold comentario conteudo:text  
comentavel_id:integer comentavel_type
```

```
rr71@caelum131-03: ~/vota_prato  
File Edit View Terminal Tabs Help  
rr71@caelum131-03:~/vota_prato$ rails generate scaffold comentario conteudo:text comentavel_id:integer comentavel_type:string  
invoke    active_record  
create    db/migrate/20100726081933_create_comentarios.rb  
create    app/models/comentario.rb  
invoke    test_unit  
create    test/unit/comentario_test.rb  
create    test/fixtures/comentarios.yml  
route     resources :comentarios  
invoke    scaffold_controller  
create    app/controllers/comentarios_controller.rb  
invoke    erb  
create    app/views/comentarios  
create    app/views/comentarios/index.html.erb  
create    app/views/comentarios/edit.html.erb  
create    app/views/comentarios/show.html.erb  
create    app/views/comentarios/new.html.erb  
create    app/views/comentarios/_form.html.erb  
invoke    test_unit  
create    test/functional/comentarios_controller_test.rb  
invoke    helper  
create    app/helpers/comentarios_helper.rb  
invoke    test_unit  
create    test/unit/helpers/comentarios_helper_test.rb  
invoke    stylesheets  
identical public/stylesheets/scaffold.css  
rr71@caelum131-03:~/vota_prato$
```

c. Vamos inserir alguns índices físicos. Abra o arquivo **db/migrate/<timestamp>_create_comentarios.rb**

d. Insira as seguintes linhas:

```
add_index :comentarios, :comentavel_type  
add_index :comentarios, :comentavel_id
```

A screenshot of a text editor window titled "20100726081933_create_comentarios.rb (~vota_prato/db/migrations)". The editor shows the following Ruby code:

```
class CreateComentarios < ActiveRecord::Migration
  def self.up
    create_table :comentarios do |t|
      t.text :conteudo
      t.integer :comentavel_id
      t.string :comentavel_type

      t.timestamps
    end

    add_index :comentarios, :comentavel_type
    add_index :comentarios, :comentavel_id
  end

  def self.down
    drop_table :comentarios
  end
end
```

e. Volte ao Terminal e rode as migrações com o comando:

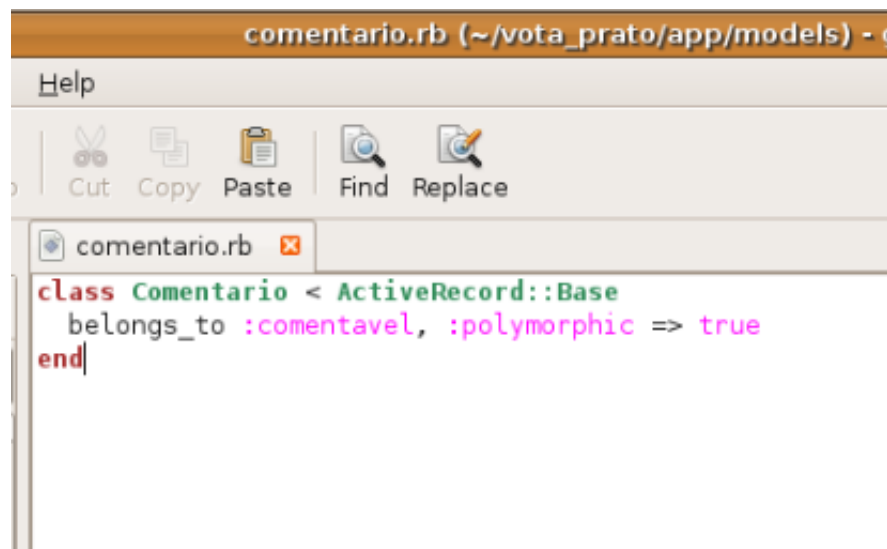
```
$ rake db:migrate
```

2. Vamos modificar nossos modelos:

a. Abra o arquivo **app/models/comentario.rb**

b. Adicione a seguinte linha:

```
belongs_to :comentavel, polymorphic: true
```

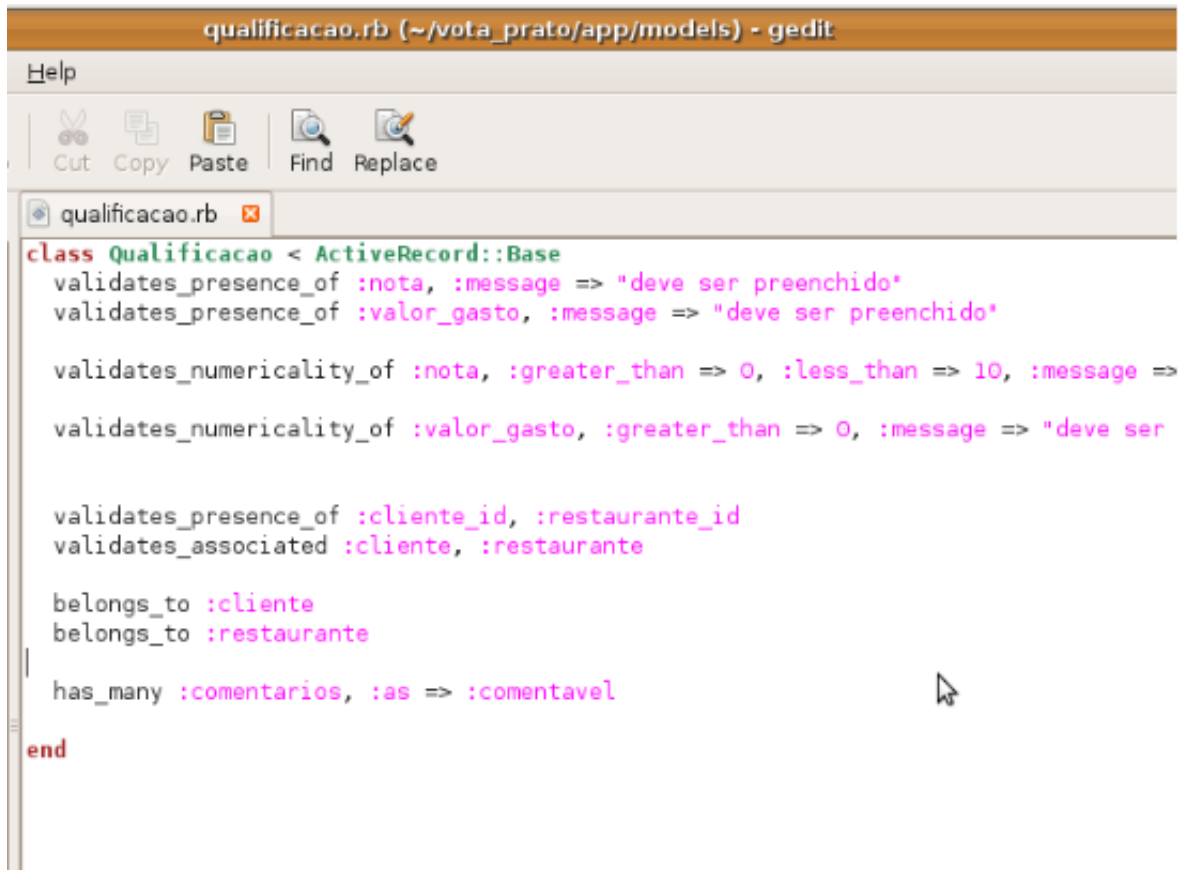
A screenshot of a text editor window titled "comentario.rb (~vota_prato/app/models)". The editor shows the following Ruby code:

```
class Comentario < ActiveRecord::Base
  belongs_to :comentavel, :polymorphic => true
end
```

c. Abra o arquivo **app/models/qualificacao.rb**

d. Adicione a seguinte linha:

```
has_many :comentarios, as: :comentavel
```



```
qualificacao.rb (~/vota_prato/app/models) - gedit

Help

Cut Copy Paste Find Replace

qualificacao.rb

class Qualificacao < ActiveRecord::Base
  validates_presence_of :nota, :message => "deve ser preenchido"
  validates_presence_of :valor_gasto, :message => "deve ser preenchido"

  validates_numericality_of :nota, :greater_than => 0, :less_than => 10, :message =>
  validates_numericality_of :valor_gasto, :greater_than => 0, :message => "deve ser

  validates_presence_of :cliente_id, :restaurante_id
  validates_associated :cliente, :restaurante

  belongs_to :cliente
  belongs_to :restaurante

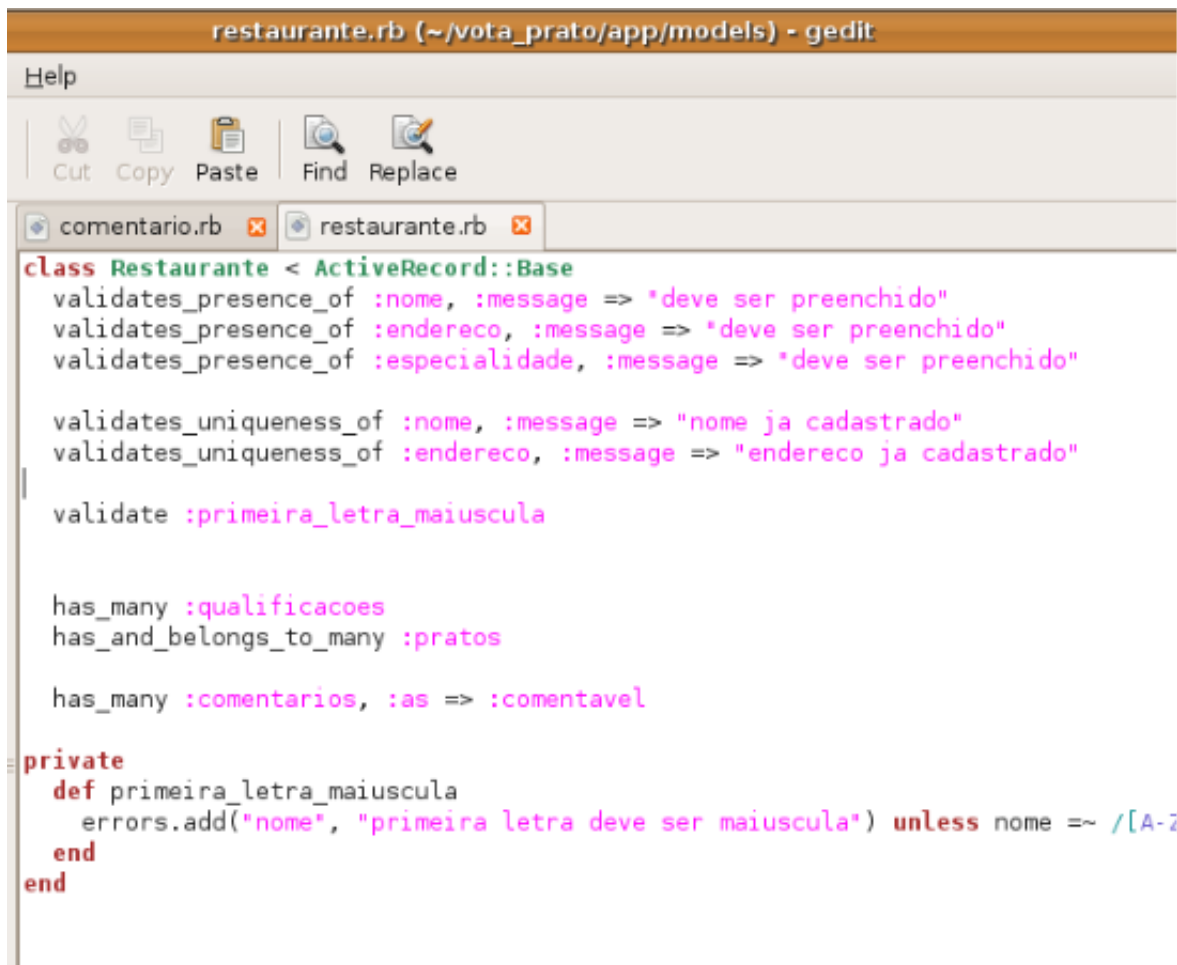
  has_many :comentarios, :as => :comentavel

end
```

e. Abra o arquivo **app/models/restaurante.rb**

f. Adicione a seguinte linha:

```
has_many :comentarios, as: :comentavel
```



```
restaurante.rb (~/vota_prato/app/models) - gedit
Help
Cut Copy Paste Find Replace
comentario.rb x restaurante.rb x
class Restaurante < ActiveRecord::Base
  validates_presence_of :nome, :message => "deve ser preenchido"
  validates_presence_of :endereco, :message => "deve ser preenchido"
  validates_presence_of :especialidade, :message => "deve ser preenchido"

  validates_uniqueness_of :nome, :message => "nome ja cadastrado"
  validates_uniqueness_of :endereco, :message => "endereco ja cadastrado"

  validate :primeira_letra_maiuscula

  has_many :qualificacoes
  has_and_belongs_to_many :pratos

  has_many :comentarios, :as => :comentavel

private
  def primeira_letra_maiuscula
    errors.add("nome", "primeira letra deve ser maiuscula") unless nome =~ /[A-Z]
  end
end
```

3. Para o próximo capítulo, vamos precisar que o nosso sistema já inclua alguns comentários. Para criá-los, você pode usar o rails console ou ir em <http://localhost:3000/comentarios> e adicionar um comentário qualquer para o "Comentavel" 1, por exemplo, e o tipo "Restaurante". Isso criará um comentário para o restaurante de ID 1.

CAPÍTULO ANTERIOR:

[Calculations](#)

PRÓXIMO CAPÍTULO:

[Mais sobre views](#)

Você encontra a Caelum também em:

[Blog Caelum](#)

[Cursos Online](#)

[Facebook](#)

[Newsletter](#)

[Casa do Código](#)

[Twitter](#)