Ruby On Rails: Laboratório 03

Laboratório Engenharia de Software

PUC Minas

Poços de Caldas

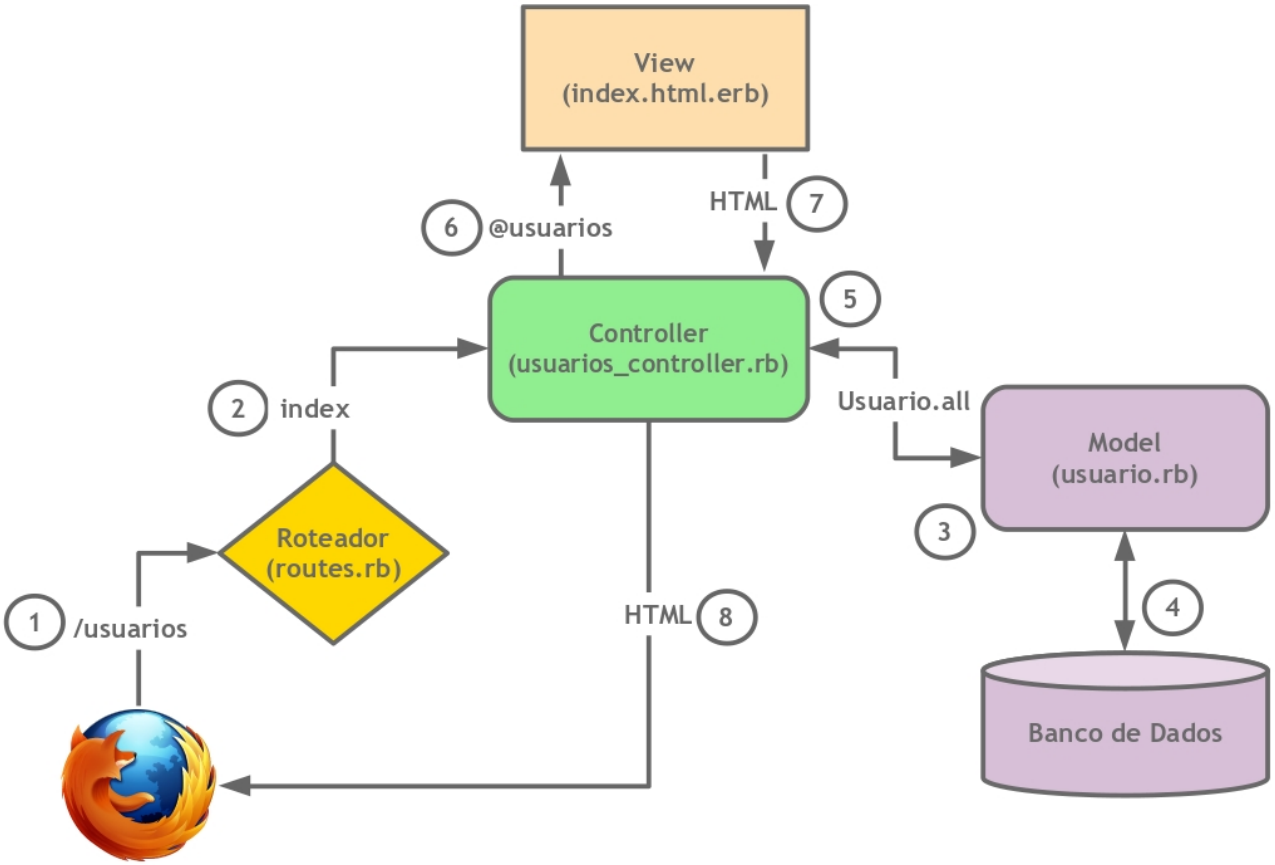
Luiz Alberto Ferreira Gomes

Curso de Ciência da Computação

16 de abril de 2020

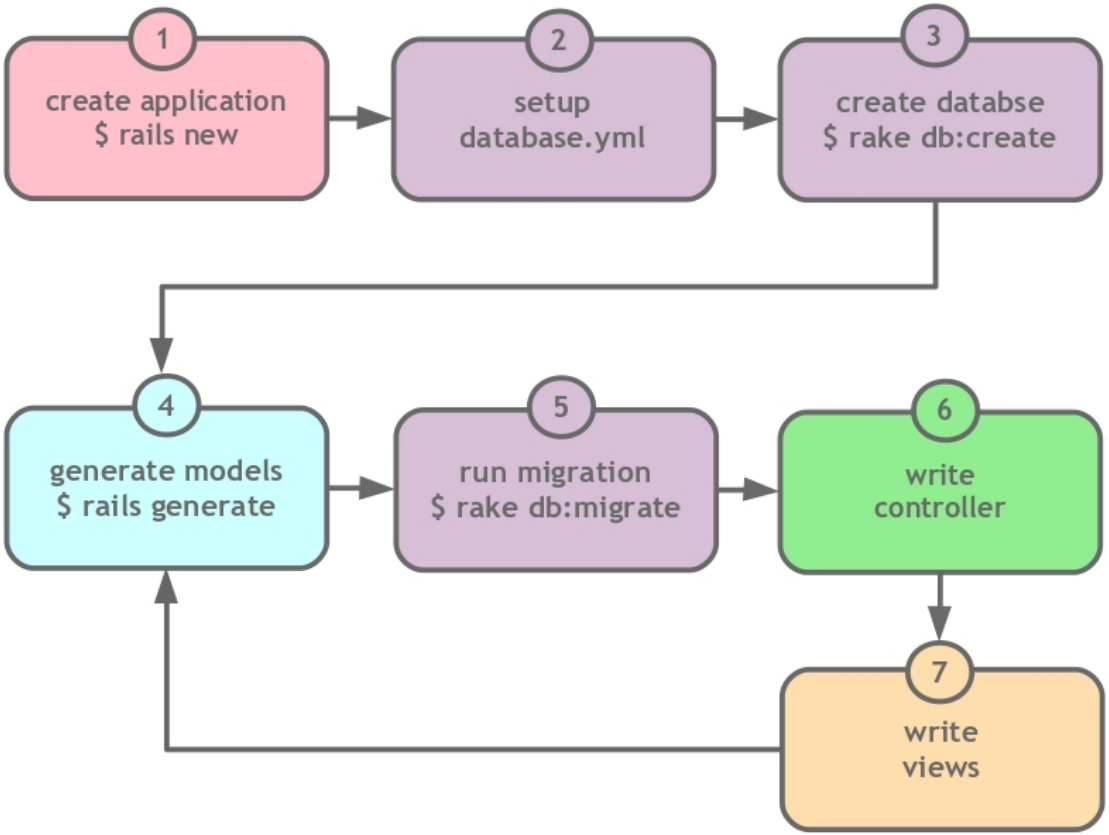
Agenda

Model-View-Controller



Agenda

Metodologia de Trabalho



Agenda

Estrutura de uma Aplicação Rails (1)

File/Folder	Purpose
app/	Contains the controllers, models, views, helpers, mailers, channels, jobs, and assets for your application. You'll focus on this folder for the remainder of this guide.
bin/	Contains the rails script that starts your app and can contain other scripts you use to setup, update, deploy, or run your application.
config/	Configure your application's routes, database, and more. This is covered in more detail in Configuring Rails Applications.
config.ru	Rack configuration for Rack based servers used to start the application. For more information about Rack, see the Rack website.
db/	Contains your current database schema, as well as the database migrations.
Gemfile Gemfile.lock	These files allow you to specify what gem dependencies are needed for your Rails application. These files are used by the Bundler gem. For more information about Bundler, see the Bundler website.

Estrutura de uma Aplicação Rails (2)

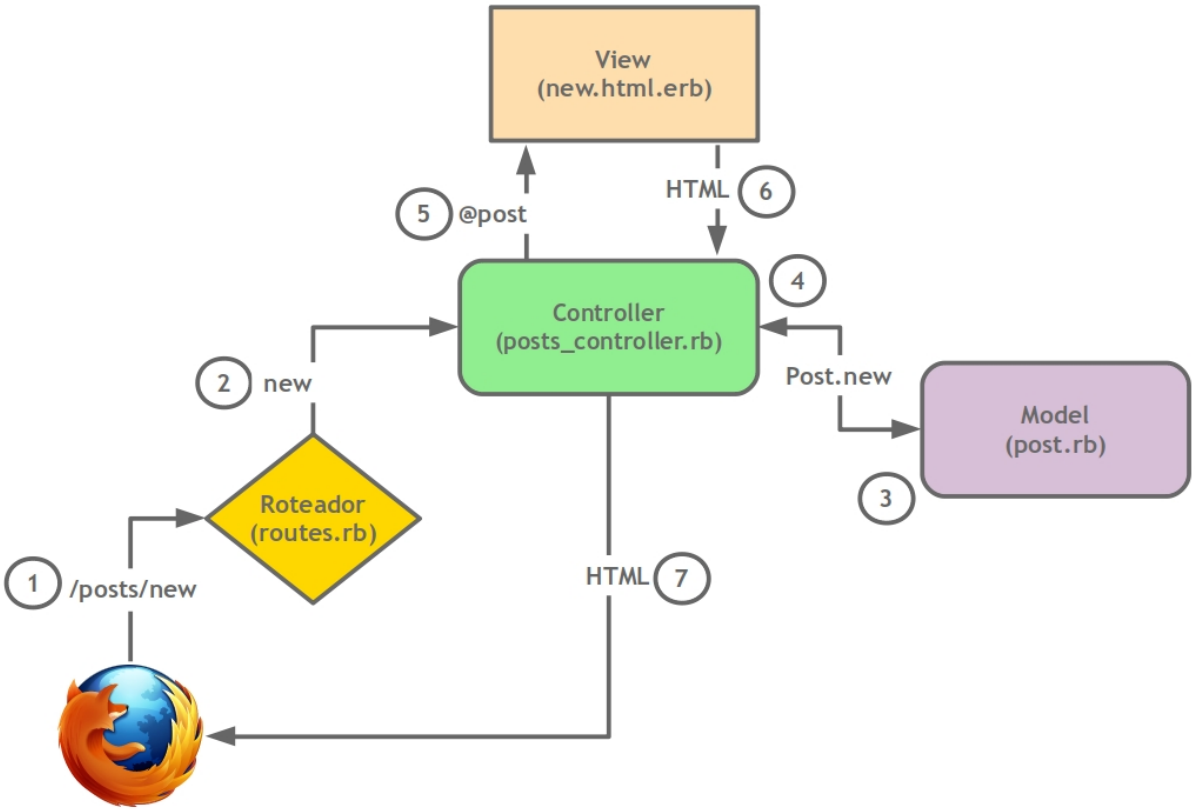
lib/	Extended modules for your application.
log/	Application log files.
package.json	This file allows you to specify what npm dependencies are needed for your Rails application. This file is used by Yarn. For more information about Yarn, see the Yarn website.
public/	The only folder seen by the world as-is. Contains static files and compiled assets.
Rakefile	This file locates and loads tasks that can be run from the command line. The task definitions are defined throughout the components of Rails. Rather than changing `Rakefile`, you should add your own tasks by adding files to the `lib/tasks` directory of your application.
README.md	This is a brief instruction manual for your application. You should edit this file to tell others what your application does, how to set it up, and so on.

Estrutura de uma Aplicação Rails (3)

storage/	Active Storage files for Disk Service. This is covered in Active Storage Overview.
test/	Unit tests, fixtures, and other test apparatus. These are covered in Testing Rails Applications.
tmp/	Temporary files (like cache and pid files).
vendor/	A place for all third-party code. In a typical Rails application this includes vendored gems.
.gitignore	This file tells git which files (or patterns) it should ignore. See GitHub - Ignoring files for more info about ignoring files.
.ruby-version	This file contains the default Ruby version.

Agenda

Ação: New (1)


View
(new.html.erb)
HTML
6
5
@post
Controller
(posts_controller.rb)
4
2
new
Post.new
Model
(post.rb)
3
Roteador
(routes.rb)
1
/posts/new
HTML
7

Ação: New (2)

- Um novo objeto @post da classe Post é instanciado
- Procura pela visão new.html.erb para renderizar a resposta

Listing 1: app/controllers/posts_controller.rb

```
1  Class PostsController < ApplicationController
2    def new
3      @post = Post.new
4    end
5  end
```

Visão: New (1)

- Reinicie o servidor web e acesse a url ⟨http:\localhost:3000/posts/new⟩. Veja o erro que ocorreu.
- Implemente a visão new.html.erb:

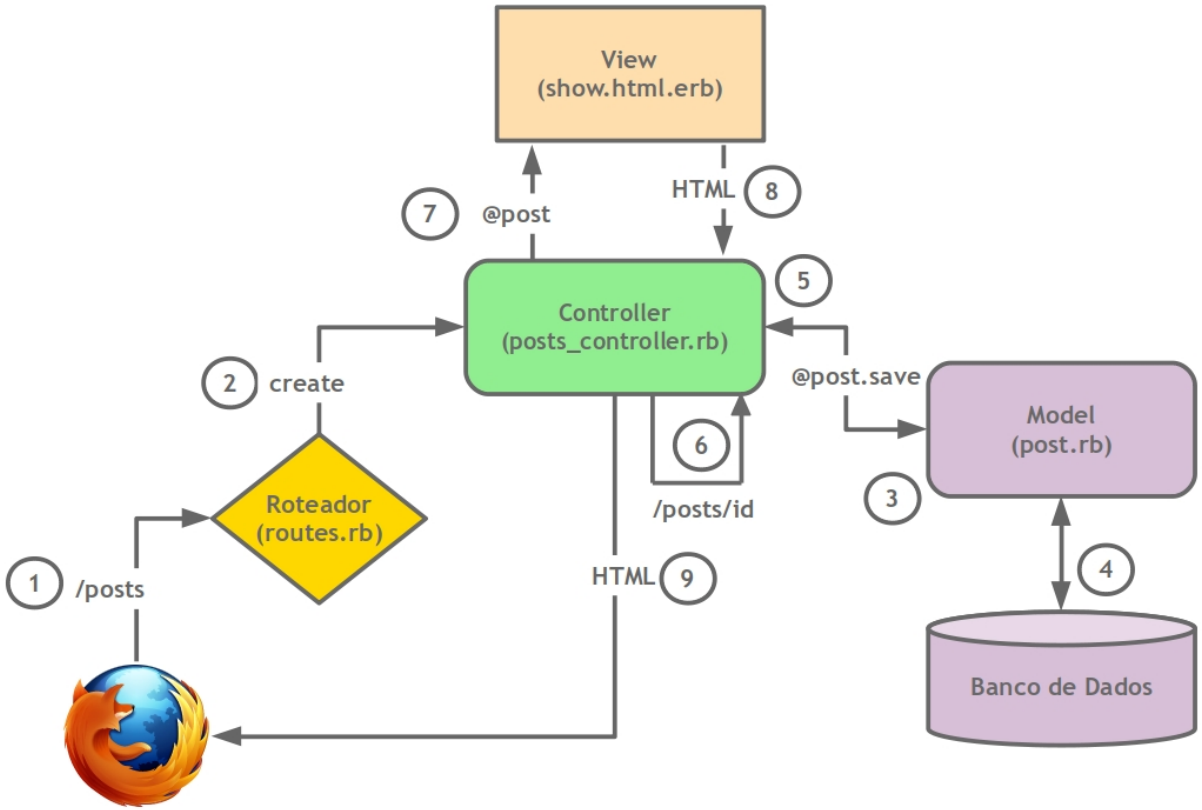
Listing 2: views/posts/new.html.erb

```
1  <h1>Novo Post</h1>
2  <%= form_with scope: :post, url: posts_path,
3     local: true do |form| %>
4     <p>
5        <%= form.label :title %><br>
6        <%= form.text_field :title %>
7     </p>
8
9     <p>
10       <%= form.label :body %><br>
```

Visão: New (2)

```
11          <%= form.text_area :body %>
12      </p>
13
14      <p>
15          <%= form.submit %>
16      </p>
17  <% end %>
```

Ação: Create (1)



Ação: Create (2)

- Um novo objeto @post da classe Post é criado com os parâmetros que foram passados pelo formulário new
- Tenta salvar o objeto @post no banco de dados

Listing 3: controllers/posts_controller.rb

```
1  class PostsController < ApplicationController
2    def new
3        @post = Post.new
4    end
5
6    def create
7        @post = Post.new(post_params)
8
9        @post.save
10       redirect_to @post
11   end
```

Ação: Create (3)

```
12
13  private
14    def post_params
15      params.require(:post).permit(:title, :body)
16    end
17  end
```

- a linha 15 implementa strong parameters para aumentar a segurança da aplicação

- Como a ação **Show** ainda não foi implementada, ocorrerá uma erro quando o botão **Submit** for pressionado.

Prática de 1

1. Implemente a ação Create para que um post possa ser gravado no banco de dados. blog.

Database Console

- O comando rails db fornece uma console para acesso aos bancos dados MySQL, PostgreSQL e SQLite.

```
$ rails db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> .headers on
sqlite> .mode columns
sqlite> select * from posts;
id          title             body            created_at                  updated_at
----------  ----------------  --------------  --------------------------  --------------------------
5           A Linguagem Ruby  Ruby e legal.  2016-04-30 22:45:20.636363  2016-04-30 22:45:20.636363
sqlite>
```

- Dica: utilize headers on e mode coluns

Prática 2 (1)

- Inicialize na pasta da aplicação a console do banco de dados e configure a sua exibição:

```
$ rails db
sqlite> .headers on
sqlite> .mode columns
```

- Exiba os colunas da tabela `posts`:

```
sqlite> .schema posts
```

Prática 2 (2)

- Exiba todos os `posts`:

```
sqlite> SELECT * FROM posts;
```

- Exiba todos os `posts` ordenados pelo título (title):

```
sqlite> SELECT * FROM posts ORDER BY title;
```

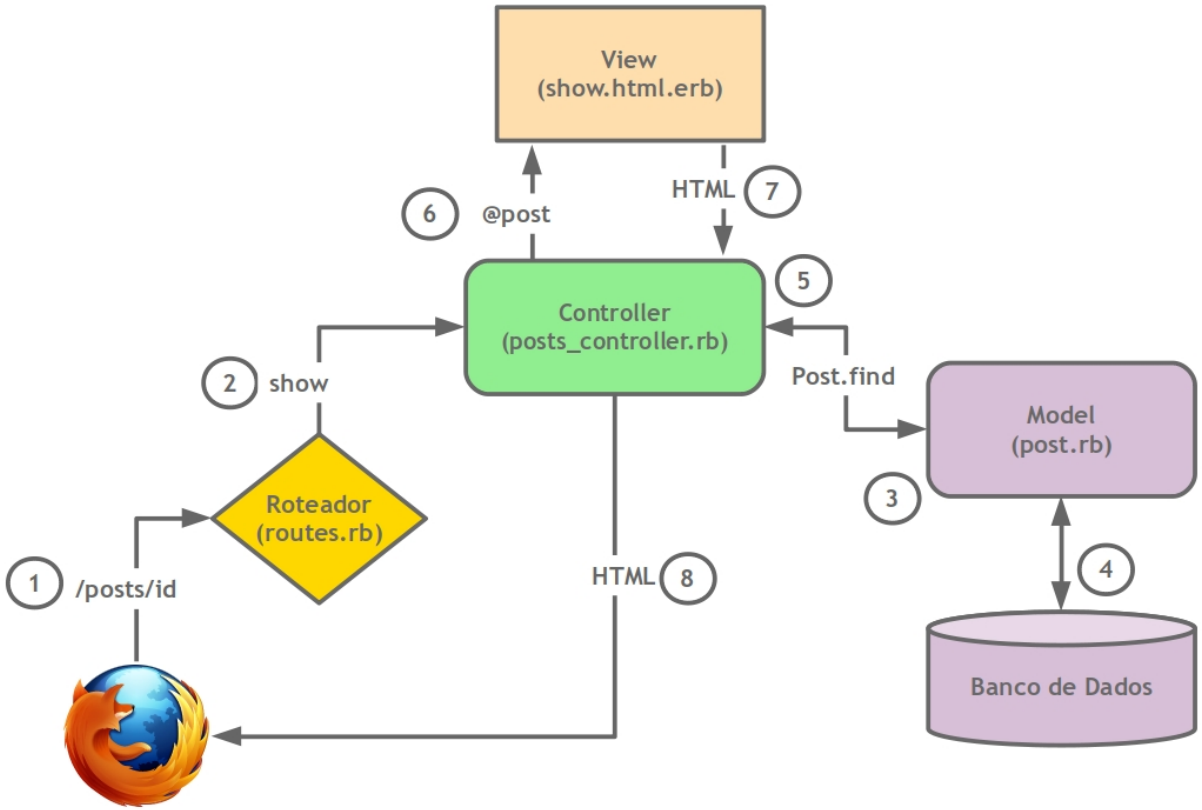
- Exiba um `post`:

```
sqlite> SELECT * FROM posts LIMIT 1
```

- Exiba o `post` cujo `id` é 2:

```
sqlite> SELECT * FROM posts WHERE id=2;
```

Ação: Show (1)


View
(show.html.erb)
HTML
7
6
@post
Controller
(posts_controller.rb)
5
2
show
Post.find
Model
(post.rb)
Roteador
(routes.rb)
3
4
1
/posts/id
HTML
8
Banco de Dados

Ação: Show (2)

- Recupera uma postagem específica no parâmetro id passado como parte da URL
- (Implicitamente) procura pelo show.html.erb para renderizar a resposta

Listing 4: controllers/posts_controller.rb

```
1  class PostsController < ApplicationController
2     def show
3        @post = Post.find(params[:id])
4     end
5
6     def new
7        @post = Post.new
8     end
9
```

Ação: Show (3)

```
10      def create
11          @post = Post.new(post_params)
12
13          @post.save
14          redirect_to @post
15      end
16  private
17      def post_params
18          params.require(:post).permit(:title, :body)
19      end
20  end
```

Visão: Show (1)

- Implemente a visão show.html.erb:

Listing 5: views/posts/show.html.erb

```
1  <p>
2     <strong>Title:</strong>
3     <%= @post.title %>
4  </p>
5  <p>
6  <strong>Body:</strong>
7     <%= @post.body %>
8  </p>
```

Rails Console (1)

- Inicialize na pasta da aplicação a console do Rails (não a do banco de dados):

```
$ rails c
```

- Exiba os atributos da classe `Post`:

```
irb(main):004:0> Post.column_names
```

- Crie um novo `post` e salve no banco de dados:

```
irb(main):005:0> p1 = Post.new
irb(main):006:0> p1.title="Rails is Cool!"
irb(main):007:0> p1.body="Rails is really Cool..."
irb(main):008:0> p1.save
```

Rails Console (2)

- Exiba todos os `posts`:

```
irb(main):007:0> Post.all
```

- Exiba todos os `posts` ordenados pelo título (title):

```
irb(main):007:0> Post.all.order(title: :asc)
```

- Exiba um `post`:

```
irb(main):007:0> Post.first
```

- Exiba o `post` cujo `id` é 2:

```
irb(main):007:0> Post.find_by(id: 2)
```

Rails Console (3)

- Atualize o título do primeiro `post`:

```
irb(main):007:0> p1=Post.first
irb(main):008:0> p1.update(title: "Rails rules!")
```

- Remova do primeiro `post`:

```
irb(main):007:0> p1=Post.first
irb(main):008:0> p1.destroy
```

Prática de 3 (1)

1. Implemente a ação Show e a visão correspondente na aplicação blog.

Agenda

Para Saber Mais

- ⟨https://www.ruby-lang.org/en/⟩
  - referência oficial da linguagem Ruby onde a toda a sua documentação está disponível para ser consultada.
- ⟨http://rubyonrails.org/⟩
  - referência oficial do framework Rails onde a toda a sua documentação está disponível para ser consultada.
- ⟨http://www.codecademy.com/pt/tracks/ruby⟩
  - curso iterativo em portugês sobre a linguagem Ruby.
- ⟨https://gorails.com/setup/ubuntu/16.04⟩
  - guia para instalação do Ruby on Rails no Ubuntu e no Mac OSX.