Desenvolvimento de Software para a Web II - Roteiro 3

Tema: Páginas estáticas e testes

Prof. Thiago Cavalcante

1. Crie um app Rails chamado **terceiro_app** com o comando **rails new terceiro_app**
2. Substitua o conteúdo do **Gemfile** do terceiro_app com o conteúdo do Gemfile do link **https://git.io/JvtEb** e atualize os pacotes com o **Bundler** usando **bundle install --without production**, **bundle update** e **bundle install --without production**
3. Adicione os arquivos no **git** com **git add -A** e faça o **primeiro commit** com **git commit -m "Inicializar o repositório"**
4. Crie um novo repositório chamado **terceiro_app** no GitHub (lembre-se de enviar o repositório para mim ou me adicionar como colaborador) e envie o código do seu repositório local para ele
5. De forma semelhante ao **primeiro_app**, crie uma ação **ola** no controlador da aplicação, modifique a rota da página inicial para essa ação e **faça um commit** para salvar essas alterações (**git commit -m "Adicionar ação olá ao controlador"**)
6. Crie o app no **Heroku** com **heroku create** (lembre-se de enviar o link do Heroku para mim) e envie o código para o **GitHub** e para o **Heroku** (*deployment*) usando **git push origin master** e **git push heroku master** (verifique que o *deployment* funcionou)
7. Crie um novo branch para trabalhar com as páginas estáticas do aplicativo usando **git checkout -b paginas-estaticas**

8. Páginas estáticas são aquelas cuja informação é sempre a mesma (ao contrário de páginas dinâmicas). Serão criadas três páginas estáticas: **inicio**, **ajuda** e **sobre**. Para gerar as duas primeiras automaticamente, utilize **rails generate controller PaginasEstaticas inicio ajuda** (preste atenção ao formato da escrita -- classes em ruby tem o nome escrito com o formato **CamelCase** e o rails converte para **snake_case** na hora de criar seus arquivos)

9. Salve suas alterações no git com um commit (exemplo de mensagem: "Criar controlador para páginas estáticas", não esqueça de adicionar os arquivos antes de fazer o commit) e envie para o GitHub usando **git push -u origin paginas-estaticas**

10. Se você acessar o arquivo **routes.rb**, verá que o comando **generate** do passo 8 criou algumas linhas de código novas lá.

config/routes.rb

```
Rails.application.routes.draw do
  get 'paginas_estaticas/inicio'
  get 'paginas_estaticas/ajuda'
  root 'application#ola'
end
```

O comando **get 'paginas_estaticas/inicio'**, por exemplo, mapeia uma requisição para a URL "/paginas_estaticas/inicio" à ação **inicio** no controlador das páginas estáticas. O uso de **get**, especificamente, faz com que essa rota responda ao **método GET do protocolo HTTP**. Para verificar o resultado dessas rotas, inicie seu aplicativo com **rails server** e acesse as URLs **localhost:3000/paginas_estaticas/inicio** e **localhost:3000/paginas_estaticas/ajuda**

OBS.: O **protocolo HTTP** é a forma com a qual o seu navegador (*cliente*) troca informações com as páginas na internet (que rodam em um *servidor*). Quando você está programando e testando um aplicativo em Rails, seu computador assume os dois papéis ao mesmo tempo (cliente e servidor). O protocolo HTTP possui quatro operações: **GET** ("pegar"

uma página), **POST** (criar coisas, submeter formulários), **PATCH** (atualizar coisas) e **DELETE** (remover coisas). Esses métodos tem uma relação bem próxima com as ações de controlador geradas para os usuários e os microposts no Roteiro 2 (ao contrário das ações de controlador das páginas estáticas).

11. Para entender de onde vem as URLs que foram exibidas no passo 10, pode-se verificar o que suas ações de controlador no arquivo **app/controllers/paginas_estaticas_controller.rb** fazem. Quais ações estão lá e qual o conteúdo de cada uma? (responder por escrito)

12. Lembrando de como funciona a arquitetura MVC utilizada pelo Rails, além de o controlador executar a ação apropriada, ele também retorna para o usuário uma visualização associada à URL. As visualizações de um aplicativo Rails ficam na pasta **app/views**. O comando **generate**, além de criar um controlador com as ações especificadas, também cria uma pasta do controlador na pasta de visualizações do projeto. Cada ação possui uma visualização correspondente. Quais os nomes dos arquivos de visualização gerados? (responder por escrito)

13. As visualizações que foram geradas são apenas temporárias. O desenvolvedor deve alterá-las de acordo com as suas necessidades. Modifique os títulos e descrições das páginas de início e ajuda do seu projeto. Você pode seguir os modelos abaixo ou colocar um conteúdo à sua escolha.

app/views/paginas_estaticas/inicio.html.erb

```
<h1>Terceiro App</h1>
<p>
 Este é o terceiro aplicativo Rails da disciplina
 <strong>Desenvolvimento de Software para a Web II</strong>
 do curso de Sistemas de Informação em
 <a href="http://ufal.edu.br/arapiraca/unidades-de-ensino/penedo">
  Penedo</a>
</p>
```

app/views/paginas_estaticas/ajuda.html.erb

```
<h1>Ajuda</h1>
<p>
  Você pode ler mais sobre Rails na página
  <a href="https://guides.rubyonrails.org/">
    Ruby on Rails Guides</a>
  e sobre Ruby em <a href="https://ruby-doc.org/">Ruby-Doc.org</a>
</p>
```

14. Para criar a terceira página estática (**sobre**), você vai utilizar **testes** para garantir que a sua implementação está correta. Desenvolver aplicativos com o auxílio de testes requer que o programador escreva uma quantidade adicional de código. No entanto, quando bem feitos, os testes podem economizar tempo na hora de encontrar erros no *software*. Você vai começar escrevendo **testes de controlador** no roteiro atual. O comando **generate** já gerou automaticamente um arquivo de testes para as páginas estáticas. Ele está localizado em **test/controllers/paginas_estaticas_controller_test.rb**. Verifique o conteúdo desse arquivo. Quantos testes estão declarados lá? (responder por escrito)

Formato genérico para declaração de um teste

```
test "nome do teste" do
  # código do teste
  #
  # obs.: em geral, o teste termina com um comando
  # 'assert', que verifica se o resultado é o
  # esperado para o teste
end
```

15. Os testes declarados no arquivo do passo anterior simplesmente carregam as páginas de início e ajuda (com uma requisição **HTTP** do tipo **GET**) e verificam (com o método **assert_response**) se a resposta indica que o carregamento ocorreu com sucesso ou não. Para ver o resultado dos testes no seu terminal, rode os comandos **rails db:migrate** (necessário apenas na primeira vez) e então **rails test**. Qual a quantidade de: testes realizados? verificações (*assertions*) realizadas? erros obtidos? (responder por escrito)

16. Para adicionar a página estática **sobre**, será adotado o procedimento de escrita do teste primeiro. Observando como estão escritos os testes no arquivo **paginas_estaticas_controller_test.rb**, defina, nesse mesmo arquivo, um novo teste para página **sobre** e veja que o teste falha quando você roda o comando **rails test**. Qual a mensagem de erro? (responder por escrito) (dica: você pode copiar um dos testes já predefinidos e apenas alterar o nome da página)

17. A mensagem do teste anterior mostra que a URL da página **sobre** não existe. Isso é uma indicação de que o roteador da aplicação (**config/routes.rb**) precisa ser atualizado. Observando as outras linhas de código no arquivo do roteador, adicione uma nova chamada **get** para a página **sobre** (essa nova chamada cria automaticamente o método 'paginas_estaticas_sobre_url'). Rode novamente os testes com **rails test**. Existe algum erro? Se sim, qual a mensagem? (responder por escrito)

18. A mensagem do teste anterior mostra que está faltando uma ação **sobre** no controlador das páginas estáticas. Siga o modelo do arquivo **app/controllers/paginas_estaticas_controller.rb** e defina uma nova ação **sobre**. Rode novamente os testes com **rails test**. Existe algum erro? Se sim, qual a mensagem? (responder por escrito)

19. A mensagem de erro do teste anterior aponta a falta de um *template* para a página **sobre**. Dentro um aplicativo Rails, um *template* é basicamente uma **visualização**. Você deve, então, criar um novo arquivo de visualização para a página **sobre** na pasta **app/views/paginas_estaticas**. Isso pode ser feito com o comando **touch app/views/paginas_estaticas/sobre.html.erb**. Adicione algum conteúdo à essa visualização. Siga o modelo abaixo ou adicione outra coisa à sua escolha.

app/views/paginas_estaticas/sobre.html.erb

```
<h1>Sobre</h1>
<p>
 <ul>
  <li><strong>Desenvolvimento de Software para Web II</strong></li>
  <li><strong>Código: </strong>SISB031</li>
  <li><strong>Semestre: </strong>7</li>
  <li><strong>Professor: </strong>Thiago Cavalcante</li>
 </ul>
</p>
```

20. Rode novamente os testes com **rails test**. Existe algum erro? (responder por escrito)

21. Salve suas modificações no git com um commit (lembre-se de adicionar os arquivos, sugestão de mensagem: "Finalizar páginas estáticas"). Mude o seu repositório para o *branch* principal com **git checkout master**. Junte as modificações que você fez no *branch* **paginas-estaticas** ao *branch* principal com **git merge paginas-estaticas**. Envie seu código para o GitHub com **git push origin master** e faça o *deployment* para o Heroku com **git push heroku master**.