Desenvolvimento de Software para a Web II – Roteiro 2

Prof. Thiago Cavalcante

1. Crie um app Rails chamado **segundo_app**
2. Substitua o **Gemfile** padrão pelo Gemfile do **primeiro_app** e atualize os pacotes com o **Bundler***
3. Adicione os arquivos no **git** e faça o **primeiro commit**
4. Crie um repositório no **GitHub** e envie o código
5. De forma semelhante ao **primeiro_app**, crie uma ação **ola**, modifique a rota da página inicial, faça um **commit**, crie o app no **Heroku** e envie o código para o **GitHub** e para o **Heroku** (*deployment*)
6. Gere o recurso de **usuários** com o comando **scaffold**

```
rails generate scaffold User name:string email:string
```

7. Faça a migração do banco de dados

```
rails db:migrate
```

8. Rode o seu app com **rails server** para explorar as páginas de usuários
9. **Exercícios**
   (a) Crie um novo usuário e inspecione o **código-fonte** da página para descobrir o **id** CSS da mensagem de confirmação; o que acontece quando a página é atualizada?
   (b) O que acontece ao tentar criar um usuário sem e-mail?

*Lembrar de usar a opção **-without production**

(c) O que acontece ao tentar criar um usuário com um e-mail inválido?

(d) Destrua os usuários anteriores; o app mostra alguma mensagem quando o usuário é destruído?

10. Modifique o arquivo de rotas para que a página inicial do app leve à página inicial dos usuários

config/routes.rb

```
Rails.application.routes.draw do
  resources :users
  root 'users#index'
end
```

11. Faça um diagrama da arquitetura MVC e explique os passos que são realizados ao acessar a página **/users/1/edit**

12. Encontre no código a linha que obtém do banco de dados as informações do usuário do exercício anterior (dica: **set_user**)

13. Qual é o nome do arquivo de visualização para a página de edição do usuário?

14. Gere o recurso de **microposts** com o comando **scaffold** e faça a migração do banco

```
rails generate scaffold Micropost content:text user_id:integer
rails db:migrate
```

15. **Exercícios**

(a) Repita o exercício **9.(a)** para os microposts

(b) Tente criar um micropost sem conteúdo e sem id de usuário

(c) Tente criar um micropost com mais de 140 caracteres

(d) Destrua os microposts anteriores

16. Crie uma **validação** para o tamanho do conteúdo no **modelo** dos microposts

app/models/micropost.rb

```
class Micropost < ApplicationRecord
  validates :content, length: { maximum: 140 }
end
```

17. **Exercícios**

    (a) Repita o exercício **15.(c)**; existe alguma mudança no resultado?

    (b) Inspecione o **código-fonte** da página para descobrir o **id** CSS da mensagem de erro produzida na questão anterior

18. Crie uma **associação** entre usuários e microposts

    app/models/user.rb

```
class User < ApplicationRecord
  has_many :microposts
end
```

    app/models/micropost.rb

```
class Micropost < ApplicationRecord
  belongs_to :user
  validates :content, length: { maximum: 140 }
end
```

19. Abra o **console** do rails para verificar o resultado da associação usando **rails console**

20. **Exercícios**

    (a) Edite a página que mostra (**show**) o usuário para mostrar também o conteúdo do seu **primeiro micropost**

    (b) Crie uma validação para a **presença** do conteúdo no modelo de micropost (**presence: true**)

    (c) Baseado no exercício anterior, crie validações para a presença do nome e e-mail no modelo de usuário

21. **Exercícios**

    (a) Encontre a linha, no controlador da aplicação, que mostra que a classe **ApplicationController** herda da classe **Action-Controller::Base**

    (b) Existe um arquivo mostrando que a classe **ApplicationRecord** herda da classe **ActiveRecord::Base**?

22. Faça um commit com as alterações e envie para o GitHub e Heroku (deu tudo certo?)

23. Use **heroku logs** para checar o erro no *deployment*
24. Faça a migração do banco de dados no **Heroku** com **heroku run rails db:migrate**