Desenvolvimento de Software para a Web II - Roteiro 2

Prof. Thiago Cavalcante

1. Crie um app Rails chamado **segundo_app** com o comando **rails new segundo_app**
2. Substitua o conteúdo do **Gemfile** do segundo_app com o conteúdo do Gemfile do **primeiro_app** 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 **segundo_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
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**
7. Crie o recurso de **usuários** com o comando **scaffold**, isso vai gerar o código para implementar usuários com nome e e-mail no seu app

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

8. Faça a migração do banco de dados, ou seja, crie a tabela de usuários no banco, com o comando abaixo

```
rails db:migrate
```

9. Rode o seu app com **rails server** para explorar as páginas de usuários. Crie, edite e remova usuários. Clique na página de um usuário para ver suas informações. Observe a URL que aparece na barra de endereços quando você faz cada uma dessas atividades. O comando **scaffold** gera uma série de ações e visualizações associadas, de acordo com a tabela abaixo:

URL	Ação	Descrição
/users	**index**	Lista todos os usuários
/users/1	**show**	Mostra o usuário com id 1
/users/new	**new**	Cria um novo usuário
/users/1/edit	**edit**	Edita o usuário com id 1

10. **Exercícios**

    (a) Crie um novo usuário e inspecione o **código-fonte** da página para descobrir o **id** CSS da mensagem de confirmação. Qual é o id? O que acontece quando a página é atualizada? (responder por escrito)

    (b) O que acontece ao tentar criar um usuário sem e-mail? (responder por escrito)

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

    (d) Destrua os usuários das questões anteriores. O app mostra alguma mensagem quando o usuário é destruído? (responder por escrito)

11. 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
```

12. Abra no editor de texto o arquivo do controlador de usuário (**app/controllers/users_controller.rb**). Veja as ações (métodos) que estão definidas lá. Quais são? (responder por escrito)

13. De forma semelhante aos passos 7 e 8, crie o recurso de **microposts** com o comando **scaffold** e faça a migração do banco. Isso vai gerar o código para implementar microposts com conteúdo e id de usuário (autor do micropost)

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

14. **Exercícios**

    (a) Repita o exercício **10.(a)** para os microposts (responder por escrito)

    (b) Tente criar um micropost sem conteúdo e sem id de usuário. O que acontece? (responder por escrito)

    (c) Tente criar um micropost com mais de 140 caracteres. O que acontece? (responder por escrito)

    (d) Destrua os microposts das questões anteriores

15. Abra no editor de texto o arquivo do controlador de microposts (**app/controllers/microposts_controller.rb**). Veja as ações (métodos) que estão definidas lá. Quais são? (responder por escrito)

16. Crie uma **validação** para o tamanho do conteúdo no **modelo** dos microposts (verifique que deu certo tentando criar um micropost com mais de 140 caracteres)

app/models/micropost.rb

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

17. **Exercícios**

(a) Repita o exercício **14.(c)**; existe alguma mudança no resultado? (responder por escrito)

(b) Inspecione o **código-fonte** da página para descobrir o **id** CSS da mensagem de erro produzida na questão anterior. Qual o id? (responder por escrito)

18. Crie uma **associação** entre usuários e microposts usando **has_many** e **belongs_to**. Dessa forma, o aplicativo vai poder associar os microposts aos usuários que os criaram, usando o campo **user_id** presente na tabela de 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. Para verificar o resultado da associaçao que foi criada, abra o console do rails rodando o comando **rails console** no seu terminal. Ao abrir o terminal, atribua a uma variável **primeiro_usuario** o valor **User.first** (ou seja, ela recebe o primeiro usuário guardado no banco de dados) e rode o comando **primeiro_usuario.microposts**. Com isso, serão exibidos todos os microposts associados ao primeiro usuário. Para verificar o caminho contrário da associação (começando a partir de um micropost), execute os comandos a seguir. (obs.: crie pelo menos um usuário e vários microposts com o id desse usuário, para poder ver os resultados)

```
algum_micropost = primeiro_usuario.microposts.first
algum_micropost.user
```

20. **Exercícios**

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

(b) Baseado no passo **16** do roteiro, crie uma validação para a **presença** do **conteúdo** no **modelo de micropost** (**DICA: "presence: true"**) (verifique que deu certo)

(c) Baseado no exercício anterior, crie também validações para a **presença** do **nome** e **e-mail** no **modelo de usuário** (verifique que deu certo)

21. **Exercícios**

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

    (b) Existe um arquivo mostrando que a classe **ApplicationRecord** herda da classe **ActiveRecord::Base**? Qual? (dica: procure na pasta de modelos) (responder por escrito)

22. Faça um commit com as alterações (exemplo de mensagem: "Finalizar segundo_app") e envie para o GitHub e Heroku usando o comando git push. Quando você entra na página do Heroku do seu app, o site funciona normalmente? (responder por escrito)

23. Use **heroku logs** para checar o erro no *deployment* (procure por ActionView::Template::Error). Qual a mensagem de erro completa? (responder por escrito)

24. Faça a migração do banco de dados no **Heroku** com **heroku run rails db:migrate** e acesse a página do seu app no Heroku novamente e veja que ela está funcionando