Especialização em Desenvolvimento Web

Programação para Servidor 2 - Aula 1

Msc Édimo Sousa Silva
edimo.sousa@fapce.edu.br

Sumário

- Review
- Storage
- Serializer
- Tests
- Factory

Criando o projeto

$rails new aula2 --api --database=postgresql

$rails g model Picture name:string

$rails g controller api/v1/Pictures

$rails db:create

$rails db:migrate

$bundle install

$rails s

config/routes.rb

```
Rails.application.routes.draw do
  namespace :api do
    namespace :v1 do
      resource :pictures, only: [:index, :create]
    end
  end
end
```

app/models/picture.rb

```
class Picture < ApplicationRecord
  validates :name, presence: true, uniqueness: true
end
```

app/cotrollers/pictures_controller.rb

```ruby
class Api::V1::PicturesController < ApplicationController
  def create
    picture = Picture.new create_picture_params
    if picture.save
      render json: picture, status: :created
    else
      render json: { errors: picture.errors }, status: :unprocessable_entity
    end
  end

  def index
    render json: Picture.all
  end

  private

  def create_picture_params
    params.permit(:name)
  end
end
```

review - usem o postman

Active Storage

- Possui 2 tabelas
  - active_storage_blobs
  - active_storage_attachments

- $rails active_storage:install
  - Gera a migration para essas tabelas

- $rails db:migrate
  - executa a migration

config/storage.yml

```
test:
  service: Disk
  root: <%= Rails.root.join("tmp/storage") %>

local:
  service: Disk
  root: <%= Rails.root.join("storage") %>

amazon:
  service: S3
  access_key_id: "AKIAYAANLLSNOPK3RUEN"
  secret_access_key: "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"
  region: us-east-2
  bucket: aulapos
```

config

- Em config/environment/development.rb
  - ```
    config.active_storage.service = :amazon
    ```
- Em Gemfile
  - ```
    gem 'aws-sdk-s3'
    gem 'pry'
    ```
  - bundle install

app/models/picture.rb

```ruby
include Rails.application.routes.url_helpers
class Picture < ApplicationRecord
  validates :name, presence: true, uniqueness: true
  has_one_attached :image

  def image_url
    image.service_url.split("?").first
  end
end
```

Postman nele!

Refatorando

- app/api/v1/controllers/pictures_controller.rb

```
def index
  render json: Picture.all.map(&:image_url)
end
```

Serializer

- no arquivo Gemfile adicionar
  - gem 'active_model_serializers'
- no console
  - bundle install
- criar pasta
  - app/serializers
- criar arquivo
  - app/serializers/picture_serializer.rb

app/serializers/picture_serializer.rb

- no arquivo Gemfile adicionar
  - gem 'active_model_serializers'
- no console
  - bundle install
- criar pasta
  - app/serializers
- criar arquivo
  - app/serializers/picture_serializer.rb

app/serializers/picture_serializer.rb

```ruby
class PictureSerializer < ActiveModel::Serializer
  attributes :id, :name,   :image_url

  def image_url
    self.object.image_url
  end
end
```

app/api/v1/controllers/pictures_controller.rb
create - index

```ruby
def create
  picture = Picture.new create_picture_params
  picture.image.attach create_picture_params[:image] if create_picture_params[:image]
  if picture.save
    render json: picture, status: :created, serializer: PictureSerializer
  else
    render json: { errors: picture.errors }, status: :unprocessable_entity
  end
end

def index
  render json: Picture.all , each_serializer: PictureSerializer
end
```

app/api/v1/controllers/pictures_controller.rb update

```ruby
def update
  picture = Picture.find_by_id update_picture_params[:id]
  if picture
    picture.image.attach update_picture_params[:image] if update_picture_params[:image]
    picture.name = update_picture_params[:name]
    if picture.save
      render json: picture, status: :ok, serializer: PictureSerializer
    else
      render json: { errors: picture.errors }, status: :unprocessable_entity
    end
  else
    render json: { message: "not found"}, status: :not_found
  end
end
```

app/api/v1/controllers/pictures_controller.rb refatorando

- criar o método find
- usar os metodos a seguir

```ruby
def render_picture(picture, status)
  render json: picture, status: status, serializer: PictureSerializer
end

def render_picture_not_found
  render json: { message: "not found"}, status: :not_found
end
```

Testes com Rspec

- Em Gemfile
  - gem 'rspec'
- No console
  - bundle install
  - rspec --init
- Na raiz do projeto
  - criar a pasta specs
- criar as pastas e o arquivo
  - specs/controllers/api/v1/pictures_controller_spec.rb

spec/controllers/api/v1/pictures_controller_spec.rb

```ruby
require 'rails_helper'

RSpec.describe Api::V1::PicturesController, type: :controller do
  describe "#index" do
    before do
      get :index
    end
    it "responds :ok" do
      expect(response).to have_http_status(:ok)
    end
    it "contains all pictures" do
      expect(response.body.as_json).to include(Picture.all.as_json.to_s)
    end
  end
end
```

Usando factories

- Em Gemfile

```
group :development, :test do
  gem 'byebug', platforms: [:mri, :mingw, :x64_mingw]
  gem 'factory_bot_rails'
end
```

- Em config/storage.yml

```
test:
  service: S3
  access_key_id: "AKIAYAANLLSNOPK3RUEN"
  secret_access_key: "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"
  region: us-east-2
  bucket: aulapos
```

Usando factories - 2

- Em spec/rails_helper.rb
  - require 'support/factory_bot'
- Criar spec/support/factory_bot.rb

```
require 'factory_bot'
RSpec.configure do |config|
  config.include FactoryBot::Syntax::Methods
end
```

Pictures factory - spec/factories/pictures.rb

```ruby
include ActionDispatch::TestProcess
FactoryBot.define do
  factory :picture do
    name { "some name" }
    trait :with_image do
      after :create do |picture|
        file_path = Rails.root.join('spec', 'support', 'assets', 'naruto.jpeg')
        file = fixture_file_upload(file_path, 'image/jpeg')
        picture.image.attach(file)
      end
    end
  end
end
```

Usando a fabrica no pictures_controller_spec

```ruby
require 'rails_helper'

RSpec.describe Api::V1::PicturesController, type: :controller do
  describe "#index" do
    let!(:picture) { create :picture, :with_image }
    before do
      get :index
    end
    it "responds :ok" do
      expect(response).to have_http_status(:ok)
    end
    it "contains all pictures" do
      pictures = ActiveModel::SerializableResource.new(
        Picture.all,
        each_serializer: PictureSerializer
      ).to_json
      expect(response.body).to eq(pictures)
    end
  end
end
```

Factory

Referências

[1] - Hypertext Transfer Protocol -- HTTP/1.1, https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html, 2019

[2] - HTTP headers, https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers, 2019

[3] - Ruby guides, https://guides.rubyonrails.org/, 2019

[4] - Microsoft REST API Guidelines, https://github.com/Microsoft/api-guidelines/blob/vNext/Guidelines.md, 2019

[5] - RESTful Web Services, "O'Reilly Media, Inc.", 17 de dez de 2008.