Mini Curso de Testes Ruby com Rspec
One Bit Code

https://www.youtube.com/watch?v=RhbDeNXbBc0&list=PL0SpR9mvMDV7o2PZvbGjtp6sgvBcn-519&index=3

Resumo do curso feito por Roberto Pinheiro

Aula 04 - Criando seu primeiro teste

Instalando a gem

gem install rspec

```
C:\sites\test_like_a_hero (master)
λ gem install rspec
Fetching: rspec-support-3.9.0.gem (100%)
Successfully installed rspec-support-3.9.0
Fetching: rspec-core-3.9.0.gem (100%)
Successfully installed rspec-core-3.9.0
Fetching: diff-lcs-1.3.gem (100%)
Successfully installed diff-lcs-1.3
Fetching: rspec-expectations-3.9.0.gem (100%)
Successfully installed rspec-expectations-3.9.0
Fetching: rspec-mocks-3.9.0.gem (100%)
Successfully installed rspec-mocks-3.9.0
Fetching: rspec-3.9.0.gem (100%)
Successfully installed rspec-3.9.0
Parsing documentation for rspec-support-3.9.0
Installing ri documentation for rspec-support-3.9.0
Parsing documentation for rspec-core-3.9.0
Installing ri documentation for rspec-core-3.9.0
Parsing documentation for diff-lcs-1.3
Couldn't find file to include 'Contributing.rdoc' from README.rdoc
Couldn't find file to include 'License.rdoc' from README.rdoc
Installing ri documentation for diff-lcs-1.3
Parsing documentation for rspec-expectations-3.9.0
Installing ri documentation for rspec-expectations-3.9.0
Parsing documentation for rspec-mocks-3.9.0
Installing ri documentation for rspec-mocks-3.9.0
Parsing documentation for rspec-3.9.0
Installing ri documentation for rspec-3.9.0
Done installing documentation for rspec-support, rspec-core, diff-lcs, rspec-expectations, rspec-mocks, rspec after 28 seconds
6 gems installed
```

Iniciando o Rspec

rspec --init

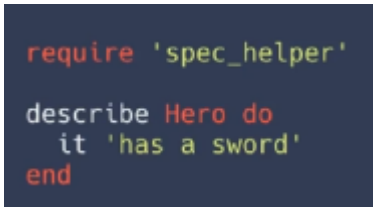
```
C:\sites\test_like_a_hero (master)
λ rspec --init
  create   .rspec
  create   spec/spec_helper.rb
```

Criando a estrutura básica

```
$ mkdir lib && mkdir spec/lib
```

Preparando o arquivo de testes

spec/lib/hero_spec.rb



require 'spec helper'

describe Hero do
    it 'has a sword'
end

Após describe vem o nome de uma classe ou de uma string.

Nesse caso, Hero é a classe que será criada.

Criando a classe

lib/hero.rb

class Hero
    attr_accessor :weapon

    def initialize
        @weapon = 'sword'
    end
end

Melhorando o teste

spec/lib/hero_spec.rb

```ruby
require 'spec_helper'
require 'hero'

describe Hero do
  it 'has a sword' do
    hero = Hero.new
    expect(hero.weapon).to eq('sword')
  end
end
```

require 'spec_helper'
require 'hero'

describe Hero do
    it 'has a sword' do
        hero = Hero.new
        expect(hero.weapon).to eql('sword')
    end
end

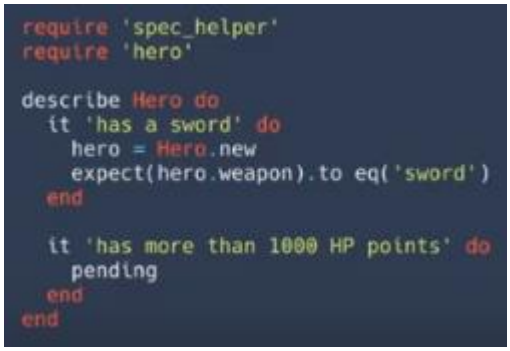
Testando

rspec spec/lib/hero_spec.rb

```
C:\sites\test_like_a_hero (master)
λ rspec spec/lib/hero_spec.rb
.

Finished in 0.03773 seconds (files took 0.76118 seconds to load)
1 example, 0 failures
```

Aula 05 - Desafio do herói #1

spec/lib/hero_spec.rb



```ruby
require 'spec_helper'
require 'hero'

describe Hero do
  it 'has a sword' do
    hero = Hero.new
    expect(hero.weapon).to eq('sword')
  end

  it 'has more than 1000 HP points' do
    pending
  end
end
```

https://www.rubydoc.info/gems/rspec-expectations/frames

Comparisons

expect(actual).to be > expected
expect(actual).to be >= expected
expect(actual).to be <= expected
expect(actual).to be < expected
expect(actual).to be_within(delta).of(expected)

Solução

lib/hero.rb

```
class Hero
    attr_accessor :weapon, :hp_points

    def initialize
        @weapon = 'sword'
        @hp_points = 1200
    end
end
```

spec/lib/hero_spec.rb

```
require 'spec_helper'
require 'hero'

describe Hero do
    it 'has a sword' do
        hero = Hero.new
        expect(hero.weapon).to eql('sword')
    end

    it 'has more than 1000 HP points' do
        hero = Hero.new
        expect(hero.hp_points).to be > 1000
    end
end
```

```
C:\sites\test_like_a_hero (master)
λ rspec spec/lib/hero_spec.rb
..

Finished in 0.0432 seconds (files took 0.74159 seconds to load)
2 examples, 0 failures
```

Aula 06 - Principais elementos do Rspec

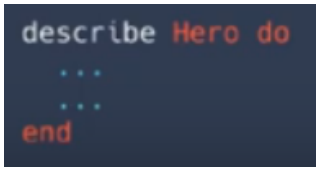
- describe
- context
- it
- expect

describe

É usado para definir um "Grupo de Exemplos" (grupo de testes).

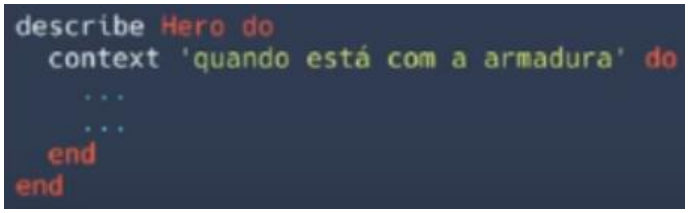
Pode receber um nome de classe ou uma string como parâmetros.

Hero
'#destroy'



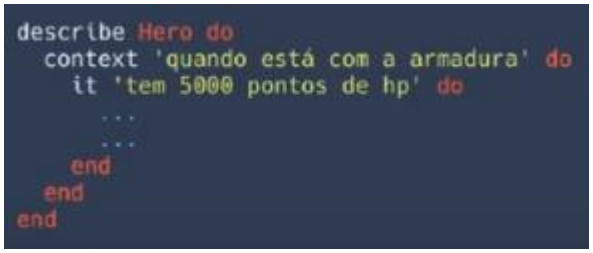
context

- Agrupa testes associados ao mesmo contexto.
- Não é obrigatório, mas ajuda a tornar os testes mais legíveis e simples.



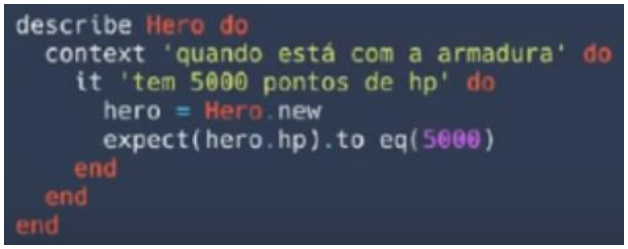
it

É usado para definir um exemplo (teste).



expected

É onde verificamos se uma determinada condição está ocorrendo para concluirmos que nosso teste passou.



Aula 08 - Instalando o Rspec no Rails

Criando o projeto

rails new test_like_a_hero --database=mysql

Incluindo o Rspec no Gemfile

gem 'rspec-rails', '~> 3.8'

```
group :development, :test do
  ...
  gem 'rspec-rails', '~> 3.8'
  ...
end
```

bundle install

Criando o banco de dados

rails db:create

```
C:\sites\test_like_a_hero (master)
λ rails db:create
Created database 'test_like_a_hero_development'
Created database 'test_like_a_hero_test'
```

Instalando o Rspec

rails generate rspec:install

```
C:\sites\test_like_a_hero (master)
λ rails generate rspec:install
      create  .rspec
      create  spec
      create  spec/spec_helper.rb
      create  spec/rails_helper.rb
```

Rodando os testes

bundle exec rspec

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec
No examples found.


Finished in 0.00099 seconds (files took 0.97497 seconds to load)
0 examples, 0 failures
```

spec_helper.rb

- Ele é o arquivo base de configuração do Rspec

Exemplo de configuração:

```
config.order = :random
```

rails_helper.rb

Arquivo que carrega as dependências do Rails e faz as configurações necessárias para que o Rspec rode em conjunto com ele.

Exemplo de configuração:

```
config.use_transactional_fixtures = true
```

Aula 9 - Usando generators

Generators são scripts que geram arquivos de testes com a estrutura básic pronta para que você possa criar seus testes facilmente.

Gerando um model com teste

rails generate model user

```
C:\sites\test_like_a_hero (master)
λ rails generate model user
      invoke  active_record
      create    db/migrate/20191110033243_create_users.rb
      create    app/models/user.rb
      invoke    rspec
      create      spec/models/user_spec.rb
```

spec/models/user_spec.rb

```ruby
require 'rails_helper'

RSpec.describe User, type: :model do
  pending "add some examples to (or delete) #{__FILE__}"
end
```

Gerando um teste de request

rails generate rspec:request User

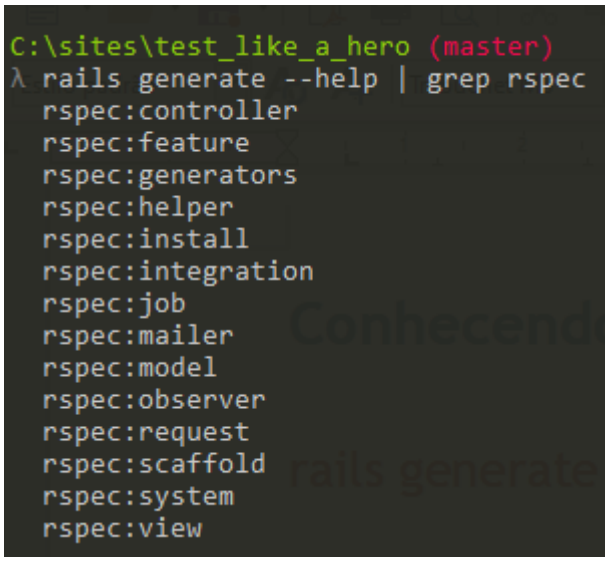
```
C:\sites\test_like_a_hero (master)
λ rails generate rspec:request User
      create  spec/requests/users_spec.rb
```

spec/requests/users_spec.rb

```ruby
require 'rails_helper'

RSpec.describe "Users", type: :request do
  describe "GET /users" do
    it "works! (now write some real specs)" do
      get users_path
      expect(response).to have_http_status(200)
    end
  end
end
```

Conhecendo todas as opções

rails generate --help | grep rspec



Aula 10 - Por que e quando testar models

O que são models

São classes Ruby que se conectam às tabelas no banco de dados para permitir uma fácil manipulação delas.

Testar os models serve para garantir que a estrutura de dados esteja correta.

Quando testar

Em geral, quando existem validações customizadas.

Aula 11 - Preparando nosso projeto para o teste

Incrementando nosso model user

rails g migration add_name_kind_level_to_user nickname:string kind:integer level:integer

```
C:\sites\test_like_a_hero (master)
λ rails g migration add_name_kind_level_to_user nickname:string kind:integer level:integer
      invoke  active_record
      create    db/migrate/20191110061011_add_name_kind_level_to_user.rb
```

db/migrate/20191110061011_add_name_kind_level_to_user.rb

```
class AddNameKindLevelToUser < ActiveRecord::Migration[5.2]
  def change
    add_column :users, :nickname, :string
    add_column :users, :kind, :integer
    add_column :users, :level, :integer
  end
end
```

Criando a tabela user

rails db:migrate RAILS_ENV=test

```
C:\sites\test_like_a_hero (master)
λ rails db:migrate RAILS_ENV=test
== 20191110060623 CreateUsers: migrating ======================================
-- create_table(:users)
   -> 0.2397s
== 20191110060623 CreateUsers: migrated (0.2406s) =============================

== 20191110061011 AddNameKindLevelToUser: migrating ===========================
-- add_column(:users, :nickname, :string)
   -> 0.4058s
-- add_column(:users, :kind, :integer)
   -> 0.4146s
-- add_column(:users, :level, :integer)
   -> 0.3637s
== 20191110061011 AddNameKindLevelToUser: migrated (1.1931s) ==================
```

Melhorando nosso model

app/models/user.rb

```
class User < ApplicationRecord
  enum kind: [:knight, :wizard]
end
```

tipos:

- knight: cavaleiro
- wizard: mago

Incluindo o método title

```
def title
  "#{self.kind} #{self.nickname} ##{self.level}"
end
```

```
class User < ApplicationRecord
    enum kind: [:knight :wizard]
    def title
        "#{self.kind} #{self.nickname} ##{self.level}"
    end
end
```

Incluindo a validação do level

```
validates :level, numericality: { greater_than: 0, less_than_or_equal_to: 99 }
```

```
class User < ApplicationRecord
    enum kind: [:knight, :wizard]
    validates :level, numericality: { greater_than: 0, less_than_or_equal_to: 99 }
    def title
        "#{self.kind} #{self.nickname} ##{self.level}"
    end
end
```

Aula 12 - Testando o model

Preparando a base do nosso teste

spec/models/user_spec.rb

```
require 'rails_helper'

RSpec.describe User, type: :model do
  it "is invalid if the level is not between 1 and 99"
  it "returns the correct hero title"
end
```

require 'rails_helper'

RSpec.describe User, type: :model do
    it "is invalid if the level is not between 1 and 99"
    it "returns the correct hero title"
end

Rodando o teste

```
$ bundle exec rspec spec/models/user_spec.rb
```

bundle exec rspec spec/models/user_spec.rb

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/models/user_spec.rb
**

Pending: (Failures listed here are expected and do not affect your suite's status)

  1) User is invalid if the level is not between 1 and 99
     # Not yet implemented
     # ./spec/models/user_spec.rb:4

  2) User returns the correct hero title
     # Not yet implemented
     # ./spec/models/user_spec.rb:5


Finished in 0.10124 seconds (files took 37.25 seconds to load)
2 examples, 0 failures, 2 pending
```

Incluindo o teste de level

```
it "is invalid if the level is not between 1 and 99" do
  expect(User.create(nickname: 'Chronos', kind: :wizard, level: 100)).to_not be_valid
end
```

```
require 'rails_helper'

RSpec.describe User, type: :model do
    it "is invalid if the level is not between 1 and 99" do
        expect(User.create(nickname: 'Chronos', kind: :wizard, level: 100)).to_not be_valid
    end
    it "returns the correct hero title"
end
```

Rodando o teste

bundle exec rspec spec/models/user_spec.rb

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/models/user_spec.rb
.*

Pending: (Failures listed here are expected and do not affect your suite's status)

  1) User returns the correct hero title
     # Not yet implemented
     # ./spec/models/user_spec.rb:7


Finished in 0.33232 seconds (files took 36.78 seconds to load)
2 examples, 0 failures, 1 pending
```

Incluindo o teste de title

```
it "returns the correct hero title" do
  user = User.create(nickname: 'Chronos', kind: :wizard, level: 1)
  expect(user.title).to eq('wizard Chronos #1')
end
```

```
require 'rails_helper'

RSpec.describe User, type: :model do
    it "is invalid if the level is not between 1 and 99" do
        expect(User.create(nickname: 'Chronos', kind: :wizard, level: 100)).to_not be_valid
    end
    it "returns the correct hero title" do
        user = User.create(nickname: 'Chronos', kind: :wizard, level: 1)
        expect(user.title).to eq('wizard Chronos #1')
    end
end
```

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/models/user_spec.rb
..

Finished in 0.29173 seconds (files took 36.95 seconds to load)
2 examples, 0 failures
```

Aula 13 - Melhorando o nosso teste com a gem FFaker

O que é a gem FFaker?

É uma gem que permite gerar valores aleatórios.

Exemplos:

```
FFaker::Name.name #=> "Green wizard"
FFaker::Internet.email #=> "green@wizard.com"
FFaker::Address.city #=> "Camelot"
```

Por que ela pode melhorar nossos testes?

Porque ela evita que criemos testes viciados, ou seja, que só funcionam com determinados valores de variáveis.

Incluindo no Gemfile

gem 'ffaker'

```
group :development, :test do
  ...
  gem 'ffaker'
  ...
end
```

bundle install

Melhorando o teste de level

```
it "is invalid if the level is not between 1 and 99" do
    nickname = FFaker::Name.first_name
    kind = %i[knight wizard].sample
    level = FFaker::Random.rand(100..9999)
    user = User.new(nickname: nickname, kind: kind, level: level)

    expect(user).to_not be_valid
  end
```

Melhorando o teste de title

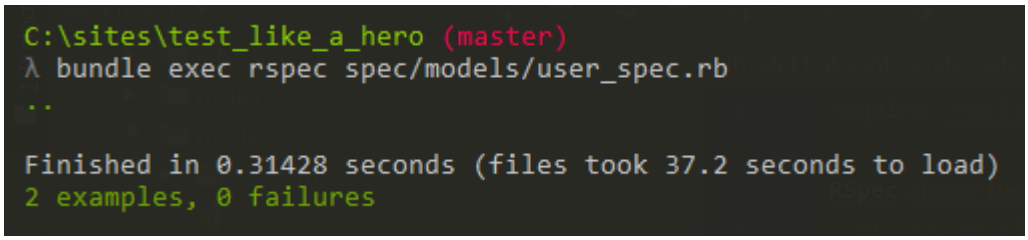
```
it "returns the correct hero title" do
  nickname = FFaker::Name.first_name
  kind = %i[knight wizard].sample
  level = FFaker::Random.rand(1..99)

  user = User.create(nickname: nickname, kind: kind, level: level)
  expect(user.title).to eq("#{kind} #{nickname} ##{level}")
end
```

spec/models/user_spec.rb

```
require 'rails_helper'

RSpec.describe User, type: :model do
  it "is invalid if the level is not between 1 and 99" do
    nickname = FFaker::Name.first_name
    kind = %i[knight wizard].sample
    level = FFaker::Random.rand(100..9999)

    user = User.new(nickname: nickname, kind: kind, level: level)
    expect(user).to_not be_valid
  end

  it "returns the correct hero title" do
    nickname = FFaker::Name.first_name
    kind = %i[knight wizard].sample
    level = FFaker::Random.rand(1..99)

    user = User.create(nickname: nickname, kind: kind, level: level)
    expect(user.title).to eq("#{kind} #{nickname} ##{level}")
  end
end
```

Rodando os testes

bundle exec rspec spec/models/user_spec.rb



Aula 14 - Melhorando o nosso teste com a gem Factory Bot

Factory Bot

Serve para manipularmos records de forma organizada nos testes.

Exemplo:

```
FactoryBot.define do
  factory :weapon do
    name { 'excalibur' }
    kind { :sword }
  end
end
```

Por que ela pode melhorar nossos testes?

Porque nós conseguimos organizar melhor a gestão dos nossos records e passamos a escrever menos códigos repetidos (DRY).

Incluindo no Gemfile

gem 'factory_bot_rails'

```
group :development, :test do
  ...
  gem 'factory_bot_rails'
  ...
end
```

bundle install

spec/rails_helper.rb

config.include FactoryBot::Syntax::Methods

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

Criando a primeira factory

- Crie, dentro da pasta ***spec***, uma subpasta chamada ***factories***. E dentro dessa subpasta crie o arquivo ***user.rb***.

```ruby
FactoryBot.define do
  factory :user do
    nickname { FFaker::Lorem.word }
    level { FFaker::Random.rand(1..99) }
    kind { %i[knight wizard].sample }
  end
end
```

spec/factories/user.rb

```ruby
FactoryBot.define do
  factory :user do
    nickname { FFaker::Lorem.word }
    level { FFaker::Random.rand(1..99) }
    kind { %i[knight wizard].sample }
  end
end
```

Atualizando o teste de level

```ruby
it "is invalid if the level is not between 1 and 99" do
  user = build(:user, level: FFaker::Random.rand(100..9999))
  expect(user).to_not be_valid
end
```

Atualizando o teste de title

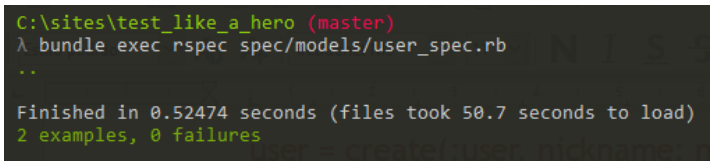
```ruby
it "returns the correct hero title" do
    nickname = FFaker::Name.first_name
    kind = %i[knight wizard].sample
    level = FFaker::Random.rand(1..99)

    user = create(:user, nickname: nickname, kind: kind, level: level)
    expect(user.title).to eq("#{kind} #{nickname} ##{level}")
  end
```

spec/models/user_spec.rb

```ruby
require 'rails_helper'

RSpec.describe User, type: :model do
  it "is invalid if the level is not between 1 and 99" do
    user = build(:user, level: FFaker::Random.rand(100..9999))
    expect(user).to_not be_valid
  end
  it "returns the correct hero title" do
    nickname = FFaker::Name.first_name
    kind = %i[knight wizard].sample
    level = FFaker::Random.rand(1..99)
    user = create(:user, nickname: nickname, kind: kind, level: level)
    expect(user.title).to eq("#{kind} #{nickname} ##{level}")
  end
end
```

Rodando os testes

bundle exec rspec spec/models/user_spec.rb



Aula 15 - Desafio do herói #2

1 - Crie um model chamado weapon com os seguintes atributos:

- name
- description
- power_base
  - pontos de poder quando a arma está no level 1, exp:3000
- power_step
  - pontos de poder que aumentam a cada level, exp:100
- level
  - level atual da arma (começando em 1)

2 - Crie os seguintes métodos no model

- current_power
  Esse método demonstra o poder atual da arma
  - ✓ (power_base + ((level - 1) * power_step))

- title
  Esse método mostra o título da arma no seguinte formato:
  - ✓ "nome_da_arma #level_da_arma"
  - ✓ ex: "excalibur #3"

3 - Realize os testes dos métodos usando as gems ffaker e factory bot.

Solução

rails generate model weapon name:string description:string power_base:integer power_step:integer level:integer

```
C:\sites\test_like_a_hero (master)
λ rails generate model weapon name:string description:string power_base:integer power_step:integer level:integer
      invoke  active_record
      create    db/migrate/20191110143528_create_weapons.rb
      create    app/models/weapon.rb
      invoke    rspec
      create      spec/models/weapon_spec.rb
      invoke      factory_bot
      create        spec/factories/weapons.rb
```

Criando a tabela

rails db:migrate RAILS_ENV=test

```
C:\sites\test_like_a_hero (master)
λ rails db:migrate RAILS_ENV=test
== 20191110143528 CreateWeapons: migrating ====================================
-- create_table(:weapons)
   -> 0.5308s
== 20191110143528 CreateWeapons: migrated (0.5323s) ===========================
```

app/models/weapon.rb

```
class Weapon < ApplicationRecord

  validates :level, numericality: { greater_than: 0, less_than_or_equal_to: 99 }

  def current_power
    (power_base + ((level - 1) * power_step))
  end

  def title
    "#{self.name} ##{self.level}"
  end

end
```

spec/factories/weapon.rb

```
FactoryBot.define do
  factory :weapon do
    name { FFaker::Name.first_name }
    description { FFaker::Lorem.sentence }
    power_base { FFaker::Random.rand(1..3000) }
    power_step { FFaker::Random.rand(1..100) }
    level { FFaker::Random.rand(1..50) }
  end
end
```

spec/models/weapon_spec.rb

```
require 'rails_helper'

RSpec.describe Weapon, type: :model do
  it "returns the weapon current power" do
    weapon = create(:weapon)
    power_weapon = (weapon.power_base + ((weapon.level - 1) * weapon.power_step))
    expect(weapon.current_power).to eq(power_weapon)
  end
  it "returns the correct weapon title" do
    weapon = create(:weapon)
    expect(weapon.title).to eq("#{weapon.name} ##{weapon.level}")
  end
end
```

spec/models/weapon_spec.rb

```
require 'rails_helper'

RSpec.describe Weapon, type: :model do
  it "returns the weapon current power" do
    weapon = create(:weapon)
    power_weapon = (weapon.power_base + ((weapon.level - 1) * weapon.power_step))
    expect(weapon.current_power).to eq(power_weapon)
  end
  it "returns the correct weapon title" do
    weapon = create(:weapon)
    expect(weapon.title).to eq("#{weapon.name} ##{weapon.level}")
  end
end
```

Rodando os testes

bundle exec rspec spec/models/weapon_spec.rb

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/models/weapon_spec.rb
..

Finished in 0.46032 seconds (files took 38.33 seconds to load)
2 examples, 0 failures
```

Aula 16 - O que são testes de request

São testes de integração (ou seja, que testam vários componentes ao mesmo tempo) que realiza uma request completa para um endpoint do seu projeto e verifica se ele está respondendo adequadamente.

Exemplo:

```ruby
describe "GET /home" do
  it "has the message 'Hello World'" do
    get home_path
    expect(response.body).to include("Hello World")
  end
end
```

Por que usar testes de request?

- Para garantir que o seu endpoint está devolvendo o status code e a resposta esperada para ele.
- Para garantir que a integração entre os diversos elementos (model, controller, rota, etc) necessários para a resposta do endpoint estão funcionando adequadamente juntos.

Quando utilizar?

Sempre que possível (principalmente se você estiver construindo uma API).

Aula 17 - Preparando nosso projeto para o teste

Criando nosso controller

rails g controller users index create --no-helper --no-assets --no-controller-specs --no-view-specs --skip-routes

```
C:\sites\test_like_a_hero (master)
λ rails g controller users index create --no-helper --no-assets --no-controller-specs --no-view-specs --skip-routes
      create  app/controllers/users_controller.rb
      invoke  erb
      create    app/views/users
      create    app/views/users/index.html.erb
      create    app/views/users/create.html.erb
      invoke  rspec
```

app/views/users/index.html.erb

```
<% @users.each do |user| %>
   <p><%= user.title %></p>
<% end %>
```

Preparando o método index

```
def index
   @users = User.all
end
```

Preparando o método create

```
...
def create
   @user = User.create(user_params)
   redirect_to_users_path
end

private

   def user_params
        params.require(:user).permit(:nickname, :kind, :level)
   end
...
```

app/controllers/users_controller.rb

```
class UsersController < ApplicationController
  def index
    @users = User.all
  end

  def create
    @user = User.create(user_params)
    redirect_to users_path
  end

  private

  def user_params
    params.require(:user).permit(:nickname, :kind, :level)
  end
end
```

Atualizando as rotas

config/routes.rb

```
Rails.application.routes.draw do
  resources :users, only:[:index, :create]
end
```

spec/factories/user.rb

```
FactoryBot.define do
  factory :user do
    nickname { FFaker::Lorem.word }
    level { FFaker::Random.rand(1..99) }
    kind { %w[knight wizard].sample }
  end
end
```

Aula 18 - Testando nosso controller

Preparando a base do teste

```ruby
require 'rails_helper'

RSpec.describe "Users", type: :request do
  describe "GET /users" do
    it "returns success status"
    it "the user's title is present"
  end

  describe "POST /users" do
    context "when it has valid parameters" do
      it "creates the user with correct attributes"
    end

    context "when it has no valid parameters" do
      it "does not create user"
    end
  end
end
```

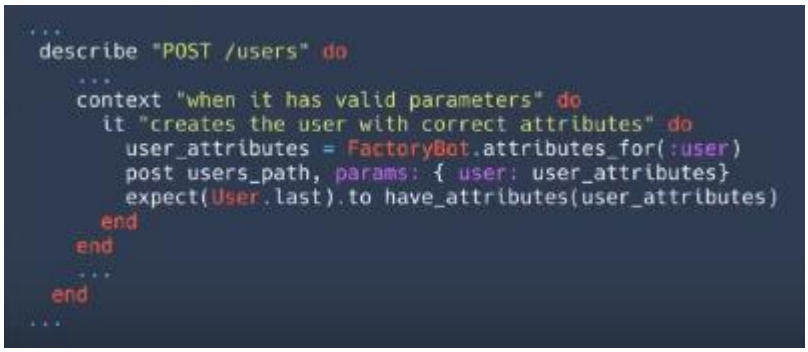
Teste de retorno de status

```ruby
...
  describe "GET /users" do
    it "returns success status" do
      get users_path
      expect(response).to have_http_status(200)
    end
  end
...
```

Teste de presença do título

```ruby
...
  describe "GET /users" do
    ....
    it "the user's title is present" do
      users = create_list(:user, 3)
      get users_path
      users.each do |user|
        expect(response.body).to include(user.title)
      end
    end
    ...
  end
...
```

Teste de criação de herói



Teste de criação de herói com parâmetros incorretos

spec/requests/users_spec.rb

```
require 'rails_helper'

RSpec.describe "Users", type: :request do
  describe "GET /users" do
    it "returns success status" do
      get users_path
      expect(response).to have_http_status(200)
    end

    it "the user's title is present" do
      users = create_list(:user, 3)
      get users_path
      users.each do |user|
        expect(response.body).to include(user.title)
      end
    end
  end
```

```
  describe "POST /users" do
    context "when it has valid parameters" do
      it "creates the user with correct attributes" do
        user_attributes = FactoryBot.attributes_for(:user)
        post users_path, params: {user: user_attributes}
        expect(User.last).to have_attributes(user_attributes)
      end
    end

    context "when it has no valid parameters" do
      it "does not create user" do
        expect {
          post users_path, params: {user: { kind: '', nickname: '', level: '' }}
        }.to_not change(User, :count)
      end
    end
  end
end
```

Rodando o teste

bundle exec rspec spec/requests/users_spec.rb

```
$ bundle exec rspec spec/requests/users_spec.rb
```

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/requests/users_spec.rb
....

Finished in 3.44 seconds (files took 38.34 seconds to load)
4 examples, 0 failures
```

Aula 19 - Desafio do herói #3

Esse desafio depende do desafio do herói #2

1 - Crie um controller para as weapons com as seguintes actions:

- index (GET /weapons)
  Devolve uma página com uma lista com os nomes, current_power's, titles e links para os detalhes das armas (página show).

- create (POST /weapons)
  Permite a criação de uma nova arma

- delete - (DELETE /weapons/:id)
  Permite a remoção de uma arma

- show (GET /weapons/:id)
  Devolve uma página com todos os detalhes de uma arma.

2 - Crie os seguintes testes:

Para o index:
- Verifique se os nomes estão sendo exibidos
- Verifique se os current_power estão sendo exibidos
- Verifique se os titles estão sendo exibidos
- Verifique se os links estão sendo exibidos corretamente

Para o create:
- Verifique se passando os parâmetros corretos a arma está sendo criada
- Verifique se passando os parâmetros incorretos a arma não está sendo criada

Para o delete:
- Verifique se passando o id correto da arma ela está sendo deletada

Para o show:
- Verifique se todos os detalhes da arma estão presentes (name, description, level, power_step, current_power e title)

Solução

rails g controller weapons index create delete show --no-helper --no-assets --no-controller-specs --no-view-specs --skip-routes

```
C:\sites\test_like_a_hero (master)
λ rails g controller weapons index create delete show --no-helper --no-assets --no-controller-specs --no-view-specs --skip-routes
      create  app/controllers/weapons_controller.rb
      invoke  erb
      create    app/views/weapons
      create    app/views/weapons/index.html.erb
      create    app/views/weapons/create.html.erb
      create    app/views/weapons/delete.html.erb
      create    app/views/weapons/show.html.erb
      invoke  rspec
```

app/controllers/weapons_controller.rb

```
class WeaponsController < ApplicationController
  before_action :set_weapon, only: [:show, :delete]

  def index
    @weapons = Weapon.all
  end

  def show
  end

  def create
    @weapon = Weapon.create(weapon_params)
    redirect_to weapons_path
  end

  def delete
    @weapon.destroy
    redirect_to weapons_path
  end

  private

  def set_weapon
    @weapon = Weapon.find(params[:id])
  end

  def weapon_params
    params.require(:weapon).permit(:name, :description, :power_base, :power_step, :level)
  end
end
```

app/views/weapons/index.html.erb

```
<% @weapons.each do |weapon| %>
  <div style="border: 1px solid red;">
    <%= weapon.name %> - <%= weapon.description %>
    <p><%= weapon.current_power %></p>
    <p><%= weapon.title %></p>
    <%= link_to 'Show', weapon %>
  </div>
<% end -%>
```

app/views/weapons/show.html.erb

```
<%= @weapon.name %> - <%= @weapon.description %>
<p><%= @weapon.current_power %></p>
<p><%= @weapon.title %></p>
```

app/models/weapon.rb

```
class Weapon < ApplicationRecord
  #### VALIDATIONS
  validates :power_base, numericality: {greater_than: 0, less_than_or_equal_to: 3000}
  validates :power_step, numericality: {greater_than: 0, less_than_or_equal_to: 100}
  validates :level, numericality: {greater_than: 0, less_than_or_equal_to: 50}

  def current_power
    (self.power_base + ((self.level-1)*self.power_step))
  end

  def title
    "#{self.name} ##{self.level}"
  end
end
```

Atualizando as rotas

config/routes.rb

```
Rails.application.routes.draw do
  resources :users, only: [:index, :create]
  resources :weapons, only: [:index, :create, :show]
  delete '/weapons/:id' => 'weapons#delete'
end
```

Factory

spec/factories/weapons.rb

```
FactoryBot.define do
  factory :weapon do
    name { FFaker::Name.first_name }
    description { FFaker::Lorem.sentence }
    power_base { FFaker::Random.rand(1..3000) }
    power_step { FFaker::Random.rand(1..100) }
    level { FFaker::Random.rand(1..50) }
  end
end
```

Construindo o arquivo de testes de requests

spec/requests/weapons_spec.rb

```
require 'rails_helper'

RSpec.describe "Weapons", type: :request do
  describe "GET /weapons" do
    it "returns success status" do
      get weapons_path
      expect(response).to have_http_status(200)
    end

    it "the weapon's name is present" do
      weapons = create_list(:weapon, 2)
      get weapons_path
      weapons.each do |weapon|
        expect(response.body).to include(weapon.name)
      end
    end

    it "the weapon's current_power is present" do
      weapons = create_list(:weapon, 2)
      get weapons_path
      weapons.each do |weapon|
        expect(response.body).to include(weapon.current_power.to_s)
      end
    end

    it "the weapon's title is present" do
      weapons = create_list(:weapon, 2)
      get weapons_path
      weapons.each do |weapon|
        expect(response.body).to include(weapon.title)
      end
    end
```

```
    it "the weapon's links is present" do
      weapons = create_list(:weapon, 2)
      get weapons_path
      weapons.each do |weapon|
        expect(response.body).to include("/weapons/#{weapon.id}")
      end
    end
  end

  describe "POST /weapons" do
    context "when it has valid parameters" do
      it "creates the weapon with correct attributes" do
        weapon_attributes = FactoryBot.attributes_for(:weapon)
        post weapons_path, params: {weapon: weapon_attributes}
        expect(Weapon.last).to have_attributes(weapon_attributes)
      end
    end

    context "when it has no valid parameters" do
      it "does not create weapon" do
        expect {
          post weapons_path, params: {weapon: { name: '', description: '', level: '' }}
        }.to_not change(Weapon, :count)
      end
    end
  end

  describe "DELETE /weapon/:id" do
    context "when it has valid parameters" do
      it "destroy weapon" do
        weapon = create(:weapon)
        expect {
          delete weapon_path(weapon)
        }.to change{ Weapon.count }.by(-1)
      end
    end
  end

  describe "GET /weapon/:id" do
    context "show weapon's parameters" do
      it "all weapon's attributes" do
        weapon = create(:weapon)
        get weapon_path(weapon)
        expect(response.body).to include(weapon.name, weapon.description, weapon.level.to_s,
weapon.current_power.to_s, weapon.title)
      end
    end
  end
end
```

Rodando os testes

bundle exec rspec spec/requests/weapons_spec.rb

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/requests/weapons_spec.rb
.........

Finished in 3.83 seconds (files took 37.97 seconds to load)
9 examples, 0 failures
```

Aula 20 - Por que e quando testar APIs

O que é uma API?

API's são maneiras de conectar serviços, com elas é possível se conectar ao Google para traduzir uma frase ou ao Watson da IBM para realizar um processamento de linguagem natural.

Que tipos de testes fazemos em API's?

Testes de request e testes unitários.

O padrão principal de API's são controllers que devolvem JSON.

Como vamos testar?

Vamos incluir um novo controller com algumas actions no nosso projeto que vão funcionar como endpoints de uma API e depois vamos realizar os testes e as melhorias nos testes.

Aula 21 - Preparando nosso projeto para o teste

Criando o controller enemies

rails g controller enemies update destroy --no-helper --no-controller-specs --no-view-specs --skip-routes

```
C:\sites\test_like_a_hero (master)
λ rails g controller enemies update destroy --no-helper --no-controller-specs --no-view-specs --skip-routes
      create  app/controllers/enemies_controller.rb
      invoke  erb
      create    app/views/enemies
      create    app/views/enemies/update.html.erb
      create    app/views/enemies/destroy.html.erb
      invoke  rspec
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/enemies.coffee
      invoke    scss
      create      app/assets/stylesheets/enemies.scss
```

Criando o model enemy

rails g model enemy name:string power_base:integer power_step:integer level:integer kind: integer

```
C:\sites\test_like_a_hero (master)
λ rails g model enemy name:string power_base:integer power_step:integer level:integer kind: integer
      invoke  active_record
      create    db/migrate/20191111000444_create_enemies.rb
      create    app/models/enemy.rb
      invoke    rspec
      create      spec/models/enemy_spec.rb
      invoke      factory_bot
      create        spec/factories/enemies.rb
```

rails db:migrate RAILS_ENV=test

```
C:\sites\test_like_a_hero (master)
λ rails db:migrate RAILS_ENV=test
== 20191111000444 CreateEnemies: migrating ====================================
-- create_table(:enemies)
   -> 0.4057s
== 20191111000444 CreateEnemies: migrated (0.4072s) ===========================
```

Incluindo validações e métodos em enemy

app/models/enemy.rb

```
class Enemy < ApplicationRecord
  enum kind: [ :goblin, :orc, :demon, :dragon ]

  #### VALIDATIONS
  validates :level, numericality: {greater_than: 0, less_than_or_equal_to: 99}
  validates_presence_of :name, :power_base, :power_step, :level, :kind

  def current_power
    (self.power_base + ((self.level-1)*self.power_step))
  end
end
```

Preparando os métodos auxiliares no controller

app/controllers/enemies_controller.rb

```
class EnemiesController < ApplicationController
  before_action :set_enemy, only: [:update, :destroy]

  def update
    if @enemy.update(enemy_params)
      render json: @enemy, status: :ok
    else
      render json: { errors: @enemy.errors }, status: :unprocessable_entity
    end
  end

  def destroy
    @enemy.destroy
    head 204
  end

  private
    def set_enemy
      @enemy = Enemy.find(params[:id])
    rescue ActiveRecord::RecordNotFound => e
```

```
    render json: { message: e.message }, status: :not_found
  end

  def enemy_params
    params.permit(:name, :power_base, :power_step, :level, :kind)
  end
end
```

```
before_action :set_enemy

private

def enemy_params
  params.permit(:name, :power_base, :power_step, :level, :kind)
end

def set_enemy
  @enemy = Enemy.find(params[:id])
rescue ActiveRecord::RecordNotFound => e
  render json: { message: e.message }, status: :not_found
end
```

Atualizando as rotas

```
resources :enemies, only: [:update, :destroy]
```

config/routes.rb

```
Rails.application.routes.draw do
  resources :users, only: [:index, :create]
  resources :weapons, only: [:index, :create, :show]
  delete '/weapons/:id' => 'weapons#delete'
  resources :enemies, only: [:update, :destroy]
end
```

Criando a Factory

spec/factories/enemies.rb

```
FactoryBot.define do
  factory :enemy do
    name { FFaker::Lorem.word }
    power_base { FFaker::Random.rand(1..9999) }
    power_step { FFaker::Random.rand(1..9999) }
    level { FFaker::Random.rand(1..99) }
    kind { %w[goblin orc demon dragon].sample }
  end
end
```

Aula 22 - Testando o update enemies da API

Gerando o arquivo de testes

rails generate rspec:request Enemy

```
C:\sites\test_like_a_hero (master)
λ rails generate rspec:request Enemy
      create  spec/requests/enemies_spec.rb
```

Incluindo os testes do update

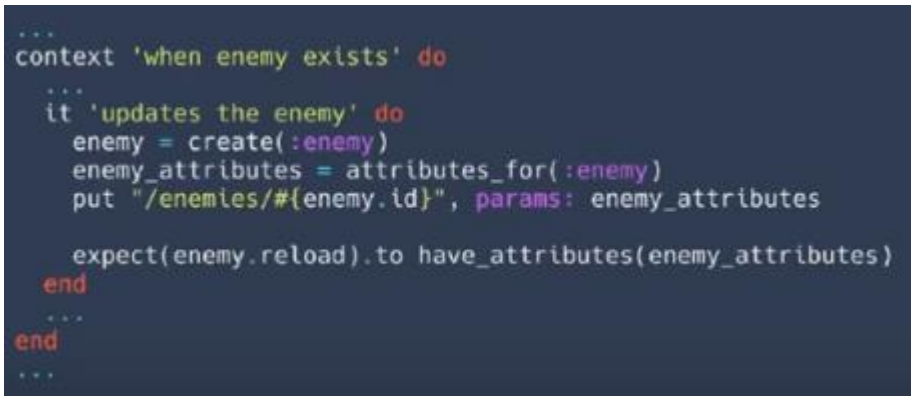
```
require 'rails_helper'

RSpec.describe "Enemies", type: :request do
  describe "PUT /enemies" do
    context 'when the enemy exists' do
      it 'returns status code 200'
      it 'updates the record'
      it 'retuns the enemy updated'
    end

    context 'when the enemy does not exist' do
      it 'returns status code 404'
      it 'returns a not found message'
    end
  end
end
```

Quando o inimigo existe, retorna o status code 200

```
...
context 'when enemy exists' do
  ...
  it 'returns status code 200' do
    enemy = create(:enemy)
    enemy_attributes = attributes_for(:enemy)
    put "/enemies/#{enemy.id}", params: enemy_attributes
    expect(response).to have_http_status(200)
  end
  ...
end
...
```

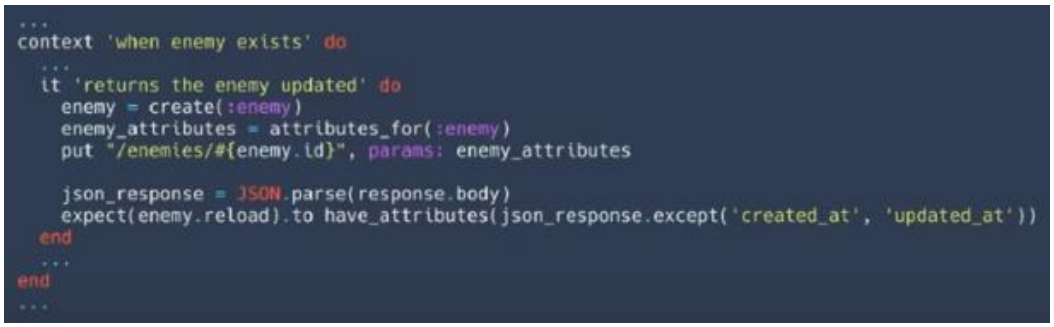
```
it "returns success status" do
    enemy = create(:enemy)
    enemy_attributes = attributes_for(:enemy)
    put "/enemies/#{enemy.id}", params: enemy_attributes
    expect(response).to have_http_status(200)
end
```

Quando o inimigo existe, atualiza o record no banco de dados



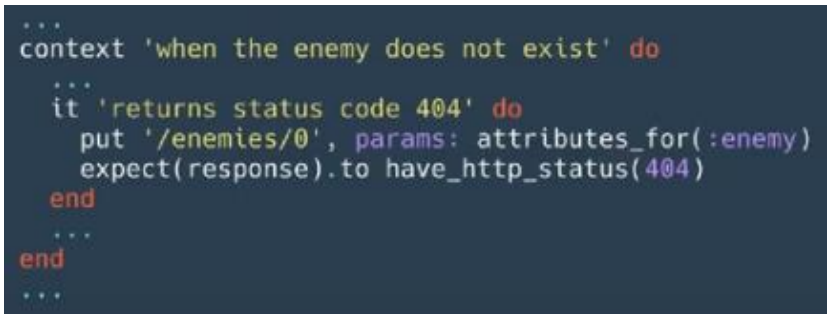

```
it "updates the enemy" do
    enemy = create(:enemy)
    enemy_attributes = attributes_for(:enemy)
    put "/enemies/#{enemy.id}", params: enemy_attributes
    expect(enemy.reload).to have_attributes(enemy_attributes)
end
```

Quando o inimigo existe, retorna o inimigo atualizado



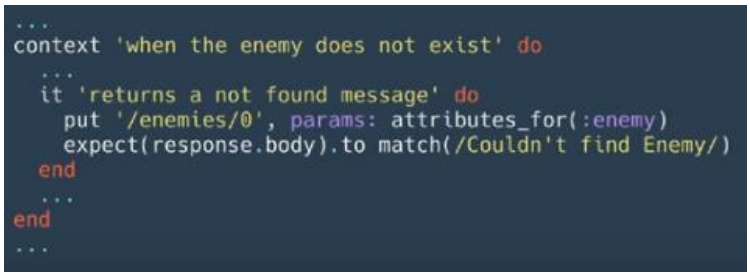

```
it "returns the enemy updated" do
    enemy = create(:enemy)
    enemy_attributes = attributes_for(:enemy)
    put "/enemies/#{enemy.id}", params: enemy_attributes
    json_response = JSON.parse(response.body)
    expect(enemy.reload).to have_attributes(json_response.except('created_at', 'updated_at'))
end
```

Quando o inimigo não existe, retorna o status code 404




```
it "returns status code 404" do
    put '/enemies/0', params: attributes_for(:enemy)
    expect(response).to have_http_status(404)
end
```

Quando o inimigo não existe, retorna uma mensagem de não encontrado




```
it "returns a not found message" do
    put "/enemies/0", params: attributes_for(:enemy)
    expect(response.body).to match(/Couldn't find Enemy/)
end
```

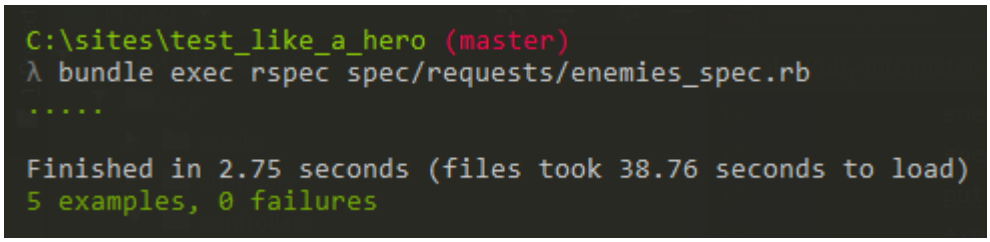
spec/requests/enemies_spec.rb

```
require 'rails_helper'

RSpec.describe "Enemies", type: :request do

  describe "PUT /enemies" do

    context "when the enemy exists" do
      it "returns success status" do
        enemy = create(:enemy)
        enemy_attributes = attributes_for(:enemy)
        put "/enemies/#{enemy.id}", params: enemy_attributes
        expect(response).to have_http_status(200)
      end
```

```
    it "updates the enemy" do
      enemy = create(:enemy)
      enemy_attributes = attributes_for(:enemy)
      put "/enemies/#{enemy.id}", params: enemy_attributes
      expect(enemy.reload).to have_attributes(enemy_attributes)
    end

    it "returns the enemy updated" do
      enemy = create(:enemy)
      enemy_attributes = attributes_for(:enemy)
      put "/enemies/#{enemy.id}", params: enemy_attributes
      json_response = JSON.parse(response.body)
      expect(enemy.reload).to have_attributes(json_response.except('created_at', 'updated_at'))
    end
  end


  context "when the enemy does not exist" do
    it "returns status code 404" do
      put '/enemies/0', params: attributes_for(:enemy)
      expect(response).to have_http_status(404)
    end

    it "returns a not found message" do
      put "/enemies/0", params: attributes_for(:enemy)
      expect(response.body).to match(/Couldn't find Enemy/)
    end
  end

 end

end
```

Rodando os testes

bundle exec rspec spec/requests/enemies_spec.rb



Aula 23 - Testando o destroy enemies da API

Incluindo os testes do destroy

```
require 'rails_helper'

RSpec.describe "Enemies", type: :request do
  ...
  describe 'DELETE /enemies' do
    context 'when the enemy exists' do
      it 'returns status code 200'
      it 'destroy the record'
    end

    context 'when the enemy does not exist' do
      it 'returns status code 404'
      it 'returns a not found message'
    end
  end
  ...
end
```

Quando o inimigo existe, retorna o status code 204

```
...
context 'when enemy exists' do
  ...
  it 'returns status code 204' do
    enemy = create(:enemy)
    delete "/enemies/#{enemy.id}"
    expect(response).to have_http_status(204)
  end
  ...
end
...
```

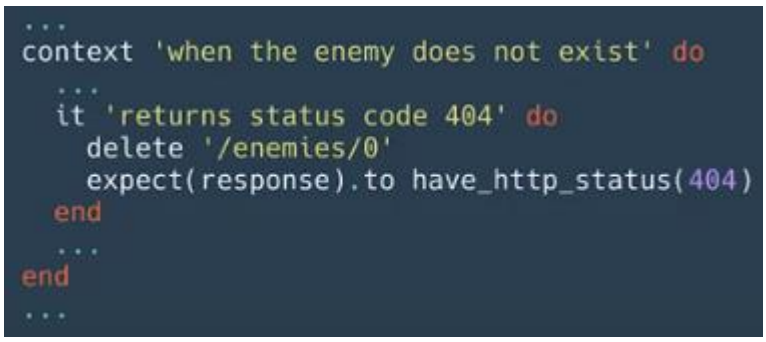
```
it "return status code 204" do
    enemy = create(:enemy)
    delete "/enemies/#{enemy.id}"
    expect(response).to have_http_status(204)
end
```

Quando o inimigo existe, destrói o record

```
...
context 'when enemy exists' do
  ...
  it 'destroy the record' do
    enemy = create(:enemy)
    delete "/enemies/#{enemy.id}"
    expect { enemy.reload }.to raise_error ActiveRecord::RecordNotFound
  end
  ...
end
...
```

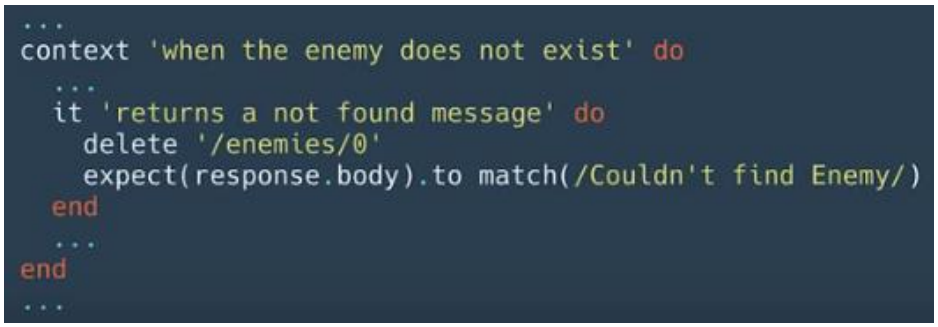
```
it "destroy the record" do
    enemy = create(:enemy)
    delete "/enemies/#{enemy.id}"
    expect { enemy.reload }.to raise_error ActiveRecord::RecordNotFound
end
```

Quando o inimigo não existe, retorna o status code 404




```
it "returns status code 404" do
    delete '/enemies/0'
    expect(response).to have_http_status(404)
end
```

Quando o inimigo não existe, retorna uma mensagem de não encontrado




```
it "returns a not found message" do
    delete '/enemies/0'
    expect(response.body).to match(/Couldn't find Enemy/)
end
```

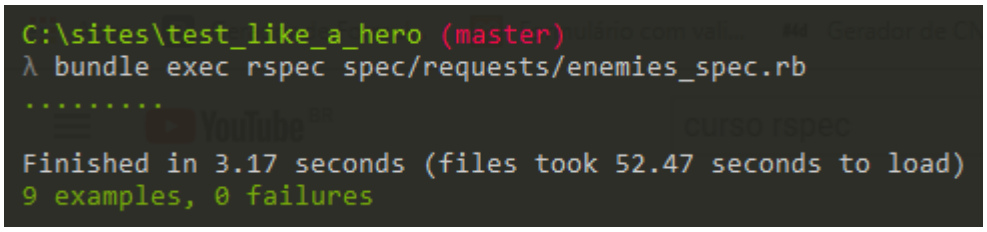
spec/requests/enemies_spec.rb

```
require 'rails_helper'

RSpec.describe "Enemies", type: :request do

  describe "PUT /enemies" do

    context "when the enemy exists" do
      it "returns success status" do
        enemy = create(:enemy)
        enemy_attributes = attributes_for(:enemy)
        put "/enemies/#{enemy.id}", params: enemy_attributes
        expect(response).to have_http_status(200)
      end

      it "updates the enemy" do
        enemy = create(:enemy)
        enemy_attributes = attributes_for(:enemy)
        put "/enemies/#{enemy.id}", params: enemy_attributes
        expect(enemy.reload).to have_attributes(enemy_attributes)
      end

      it "returns the enemy updated" do
        enemy = create(:enemy)
        enemy_attributes = attributes_for(:enemy)
        put "/enemies/#{enemy.id}", params: enemy_attributes
        json_response = JSON.parse(response.body)
        expect(enemy.reload).to have_attributes(json_response.except('created_at', 'updated_at'))
      end
    end


    context "when the enemy does not exist" do
      it "returns status code 404" do
        put '/enemies/0', params: attributes_for(:enemy)
        expect(response).to have_http_status(404)
      end

      it "returns a not found message" do
        put "/enemies/0", params: attributes_for(:enemy)
        expect(response.body).to match(/Couldn't find Enemy/)
      end
    end

  end

  describe "DELETE /enemies" do

    context "when the enemy exists" do

      it "return status code 204" do
        enemy = create(:enemy)
        delete "/enemies/#{enemy.id}"
        expect(response).to have_http_status(204)
      end
```

```
    it "destroy the record" do
      enemy = create(:enemy)
      delete "/enemies/#{enemy.id}"
      expect { enemy.reload }.to raise_error ActiveRecord::RecordNotFound
    end

  end

  context "when the enemy does not exists" do

    it "returns status code 404" do
      delete '/enemies/0'
      expect(response).to have_http_status(404)
    end

    it "returns a not found message" do
      delete '/enemies/0'
      expect(response.body).to match(/Couldn't find Enemy/)
    end
  end
end

end
```

Rodando os testes

bundle exec rspec spec/requests/enemies_spec.rb



Aula 24 - Melhorando nosso teste com um Helper Rspec

O que é um helper do Rspec?

É um método que pode ser reaproveitado ao longo dos testes.

Exemplo:

```ruby
module Helpers
  module Authentication
    def sign_in_as(user)
      # Códigos para estabelecer o sign_in
    end
  end
end
```

Por que utilizar?

Porque os helpers nos ajudam a reaproveitar nossos códigos, o que diminui a complexidade e tamanho do software.

Criando o arquivo do helper

spec/support/request_helper.rb

```
module Requests
    module JsonHelpers
        def json
            JSON.parse(response.body)
        end
    end
end
```

```ruby
module Requests
  module JsonHelpers
    def json
      JSON.parse(response.body)
    end
  end
end
```

Incluindo no Rspec

spec/rails_helper.rb

```
# This file is copied to spec/ when you run 'rails generate rspec:install'
require 'spec_helper'
ENV['RAILS_ENV'] ||= 'test'

require File.expand_path('../config/environment', __dir__)

# Prevent database truncation if the environment is production
abort("The Rails environment is running in production mode!") if Rails.env.production?
require 'rspec/rails'
# Add additional requires below this line. Rails is not loaded until this point!

# Requires supporting ruby files with custom matchers and macros, etc, in
# spec/support/ and its subdirectories. Files matching `spec/**/*_spec.rb` are
# run as spec files by default. This means that files in spec/support that end
# in _spec.rb will both be required and run as specs, causing the specs to be
# run twice. It is recommended that you do not name files matching this glob to
# end with _spec.rb. You can configure this pattern with the --pattern
# option on the command line or in ~/.rspec, .rspec or `.rspec-local`.
#
# The following line is provided for convenience purposes. It has the downside
# of increasing the boot-up time by auto-requiring all files in the support
# directory. Alternatively, in the individual `*_spec.rb` files, manually
# require only the support files necessary.
#
Dir[Rails.root.join('spec', 'support', '**', '*.rb')].each { |f| require f }

# Checks for pending migrations and applies them before tests are run.
# If you are not using ActiveRecord, you can remove these lines.
begin
  ActiveRecord::Migration.maintain_test_schema!
rescue ActiveRecord::PendingMigrationError => e
  puts e.to_s.strip
  exit 1
end
RSpec.configure do |config|
  # Remove this line if you're not using ActiveRecord or ActiveRecord fixtures
  config.fixture_path = "#{::Rails.root}/spec/fixtures"

  # If you're not using ActiveRecord, or you'd prefer not to run each of your
  # examples within a transaction, remove the following line or assign false
  # instead of true.
  config.use_transactional_fixtures = true
```

```
  # RSpec Rails can automatically mix in different behaviours to your tests
  # based on their file location, for example enabling you to call `get` and
  # `post` in specs under `spec/controllers`.
  #
  # You can disable this behaviour by removing the line below, and instead
  # explicitly tag your specs with their type, e.g.:
  #
  #     RSpec.describe UsersController, :type => :controller do
  #       # ...
  #     end
  #
  # The different available types are documented in the features, such as in
  # https://relishapp.com/rspec/rspec-rails/docs
  config.infer_spec_type_from_file_location!

  # Filter lines from Rails gems in backtraces.
  config.filter_rails_from_backtrace!
  # arbitrary gems may also be filtered via:
  # config.filter_gems_from_backtrace("gem name")

  config.include FactoryBot::Syntax::Methods

  config.include Requests::JsonHelpers, type: :request

end
```

Atualize o teste: Quando o inimigo existe retorna o inimigo atualizado

```
...
context 'when enemy exists' do
  ...
  it 'returns the enemy updated' do
    enemy = create(:enemy)
    enemy_attributes = attributes_for(:enemy)
    put "/enemies/#{enemy.id}", params: enemy_attributes
    expect(enemy.reload).to have_attributes(json.except('created_at', 'updated_at'))
  end
  ...
end
...
```

```
it "returns the enemy updated" do
    enemy = create(:enemy)
    enemy_attributes = attributes_for(:enemy)
    put "/enemies/#{enemy.id}", params: enemy_attributes
    expect(enemy.reload).to have_attributes(json.except('created_at', 'updated_at'))
end
```

Rodando os testes

```
bundle exec rspec spec/requests/enemies_spec.rb
```

```
C:\sites\test_like_a_hero (master)
λ bundle exec rspec spec/requests/enemies_spec.rb
.........

Finished in 2.98 seconds (files took 38.38 seconds to load)
9 examples, 0 failures
```

Aula 25 - Melhorando nosso teste com before e let

O que é o let?

Let é uma maneira de definir métodos/variáveis nos nossos testes que só carrega o valor quando é utilizado, e depois do primeiro usuo mantém um cache do valor durante todo o teste.

Exemplo:

```
RSpec.describe Hero do
  let(:hero) { Hero.new }

  it "has a sword" do
    expect(hero.weapon).to eq('sword')
  end
end
```

O que são hooks?

São métodos que permitem a execução de códigos antes ou depois dos testes.

Exemplo:

```
RSpec.describe Hero do
  let(:hero) { Hero.new }

  before(:each) do
    hero.update(weapon: 'axe')
  end

  it "has an axe" do
    expect(hero.weapon).to eq('axe')
  end
end
```

- Dentro do update no contexto "Quando o inimigo existe"

Retire de cada teste:

```
enemy = create(:enemy)
enemy_attributes = attributes_for(:enemy)
put "/enemies/#{enemy.id}", params: enemy_attributes
```

- Inclua no contexto:

```
context 'when enemy exists' do
  let(:enemy) { create(:enemy) }
  let(:enemy_attributes) { attributes_for(:enemy) }

  before(:each) { put "/enemies/#{enemy.id}", params: enemy_attributes }
  ...
end
```

let(:enemy) { create(:enemy) }
let(:enemy_attributes) { attributes_for(:enemy) }

before(:each) { put "/enemies/#{enemy.id}", params: enemy_attributes }

Resultado

```
context 'when enemy exists' do
  let(:enemy) { create(:enemy) }
  let(:enemy_attributes) { attributes_for(:enemy) }

  before(:each) { put "/enemies/#{enemy.id}", params: enemy_attributes }

  it 'returns status code 200' do
    expect(response).to have_http_status(200)
  end

  it 'updates the enemy' do
    expect(enemy.reload).to have_attributes(enemy_attributes)
  end

  it 'returns the enemy updated' do
    expect(enemy.reload).to have_attributes(json.except('created_at', 'updated_at'))
  end
end
```

- Dentro do update no contexto "Quando o inimigo não existe":

Retire de cada teste

```
put '/enemies/0', params: attributes_for(:enemy)
```

Inclua no contexto:

```
before(:each) { put '/enemies/0', params: attributes_for(:enemy) }
```

before(:each) { put '/enemies/0', params: attributes_for(:enemy) }

Resultado

```
context 'when the enemy does not exist' do
  before(:each) { put '/enemies/0', params: attributes_for(:enemy) }

  it 'returns status code 404' do
    expect(response).to have_http_status(404)
  end

  it 'returns a not found message' do
    expect(response.body).to match(/Couldn't find Enemy/)
  end
end
```

- Retire de cada teste:

```
enemy = create(:enemy)
delete "/enemies/#{enemy.id}"
```

- Inclua no contexto:

```
let(:enemy) { create(:enemy) }
before(:each) { delete "/enemies/#{enemy.id}" }
```

let(:enemy) { create(:enemy) }
before(:each) { delete "/enemies/#{enemy.id}" }

Resultado

```
context 'when the enemy exists' do
  let(:enemy) { create(:enemy) }
  before(:each) { delete "/enemies/#{enemy.id}" }

  it 'returns status code 200' do
    expect(response).to have_http_status(204)
  end

  it 'destroy the record' do
    expect { enemy.reload }.to raise_error ActiveRecord::RecordNotFound
  end
end
```

- Dentro do destroy no contexto "Quando o inimigo não existe"

Retire de cada teste:

```
delete '/enemies/0'
```

Inclua no contexto:

```
before(:each) { delete '/enemies/0' }
```

before(:each) { delete '/enemies/0' }

Resultado

delete '/enemies/0'

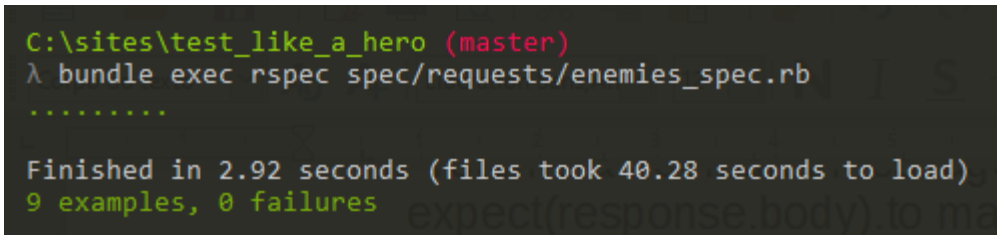
spec/requests/enemies_spec.rb

```
require 'rails_helper'

RSpec.describe "Enemies", type: :request do

  describe "PUT /enemies" do

    context "when the enemy exists" do

      let(:enemy) { create(:enemy) }
      let(:enemy_attributes) { attributes_for(:enemy) }

      before(:each) { put "/enemies/#{enemy.id}", params: enemy_attributes }

      it "returns success status" do
        expect(response).to have_http_status(200)
      end

      it "updates the enemy" do
        expect(enemy.reload).to have_attributes(enemy_attributes)
      end

      it "returns the enemy updated" do
        expect(enemy.reload).to have_attributes(json.except('created_at', 'updated_at'))
      end
    end


    context "when the enemy does not exist" do

      before(:each) { put '/enemies/0', params: attributes_for(:enemy) }

      it "returns status code 404" do
        expect(response).to have_http_status(404)
      end

      it "returns a not found message" do
        expect(response.body).to match(/Couldn't find Enemy/)
      end
    end

  end

  describe "DELETE /enemies" do

    context "when the enemy exists" do

      let(:enemy) { create(:enemy) }
      before(:each) { delete "/enemies/#{enemy.id}" }

      it "return status code 204" do
        expect(response).to have_http_status(204)
      end
```

```
    it "destroy the record" do
      expect { enemy.reload }.to raise_error ActiveRecord::RecordNotFound
    end

  end

  context "when the enemy does not exists" do

    before(:each) { delete '/enemies/0' }

    it "returns status code 404" do
      expect(response).to have_http_status(404)
    end

    it "returns a not found message" do
      expect(response.body).to match(/Couldn't find Enemy/)
    end
  end
 end

end
```

Rodando o teste

bundle exec rspec spec/requests/enemies_spec.rb



Aula 26 - Desafio do herói #4

1 - Crie no controller enemies os seguintes métodos:

- index (GET /enemies)
  Método que devolve todas as informações do inimigo via json.

- show (GET /enemies/:id)
  Método que devolve as informações de um inimigo (especificado por id) via json.

- create (POST /enemies)
  Método que permite a criação de novos inimigos e que retorna via json os dados do inimigo criado.

2 - Realize os testes nos métodos criados.

Solução

app/controllers/enemies_controller.rb

```
class EnemiesController < ApplicationController
  before_action :set_enemy, only: [:show, :update, :destroy]

  def index
    @enemies = Enemy.all
    render json: @enemies, status: :ok
  end

  def show
  end

  def create
    @enemy = Enemy.new(enemy_params)
    if @enemy.save
      render json: @enemy, status: :ok
    else
      render json: { errors: @enemy.errors }, status: :unprocessable_entity
    end
  end

  def update
    if @enemy.update(enemy_params)
      render json: @enemy, status: :ok
    else
      render json: { errors: @enemy.errors }, status: :unprocessable_entity
    end
  end

  def destroy
    @enemy.destroy
    head 204
  end

  private

    def set_enemy
      @enemy = Enemy.find(params[:id])
    rescue ActiveRecord::RecordNotFound => e
      render json: { message: e.message }, status: :not_found
    end

    def enemy_params
      params.permit(:name, :power_base, :power_step, :level, :kind)
    end
end
```

config/routes.rb

```
Rails.application.routes.draw do
  resources :users, only: [:index, :create]
  resources :weapons, only: [:index, :create, :show]
  delete '/weapons/:id' => 'weapons#delete'
  resources :enemies, only: [:index, :show, :create, :update, :destroy]
end
```

spec/requests/enemies_spec.rb

```
require 'rails_helper'

RSpec.describe "Enemies", type: :request do
  describe "GET /enemies" do
    it "returns success status" do
      get enemies_path
      expect(response).to have_http_status(200)
    end
  end

  describe "GET /enemies/:id" do
    let(:enemy) { create(:enemy) }

    it "returns success status" do
      get enemies_path(:enemy)
      expect(response).to have_http_status(200)
    end
  end

  describe "POST /enemies" do
    context "when have valid attributes" do
      it "creates the enemy with correct attributes" do
        enemy_attributes = attributes_for(:enemy)
        post enemies_path, params: enemy_attributes
        expect(Enemy.last).to have_attributes(enemy_attributes)
      end
    end

    context "when does not have valid attributes" do
      it "does not create enemy" do
        expect {
          post enemies_path, params: { name: '', level: '' }
        }.to_not change(Enemy, :count)
      end
    end
  end

  describe "PUT /enemies" do
    context "when the enemy exists" do
      let(:enemy) { create(:enemy) }
      let(:enemy_attributes) { attributes_for(:enemy) }

      before(:each) { put "/enemies/#{enemy.id}", params: enemy_attributes }
```

```
      it "return status code 200" do
        expect(response).to have_http_status(200)
      end

      it "updates the record" do
        expect(enemy.reload).to have_attributes(enemy_attributes)
      end

      it "returns the enemy updated" do
        expect(enemy.reload).to have_attributes(json.except("created_at", "updated_at"))
      end
    end

    context "when the enemy does not exists" do
      before(:each) { put "/enemies/0", params: attributes_for(:enemy) }

      it "return status code 404" do
        expect(response).to have_http_status(404)
      end

      it "returns a not found message" do
        expect(response.body).to match(/Couldn't find Enemy with 'id'=0/)
      end
    end
  end

  describe "DELETE /enemies" do
    context "when the enemy exists" do
      let(:enemy) { create(:enemy) }
      before(:each) { delete "/enemies/#{enemy.id}" }

      it "return status code 200" do
        expect(response).to have_http_status(204)
      end

      it "destroy the record" do
        expect { enemy.reload }.to raise_error ActiveRecord::RecordNotFound
      end
    end

    context "when the enemy does not exists" do
      before(:each) { delete "/enemies/0" }

      it "return status code 404" do
        expect(response).to have_http_status(404)
      end

      it "return a not found message" do
        expect(response.body).to match(/Couldn't find Enemy with 'id'=0/)
      end
    end
  end
end
```

Rodando os testes

bundle exec rspec spec/requests/enemies_spec.rb