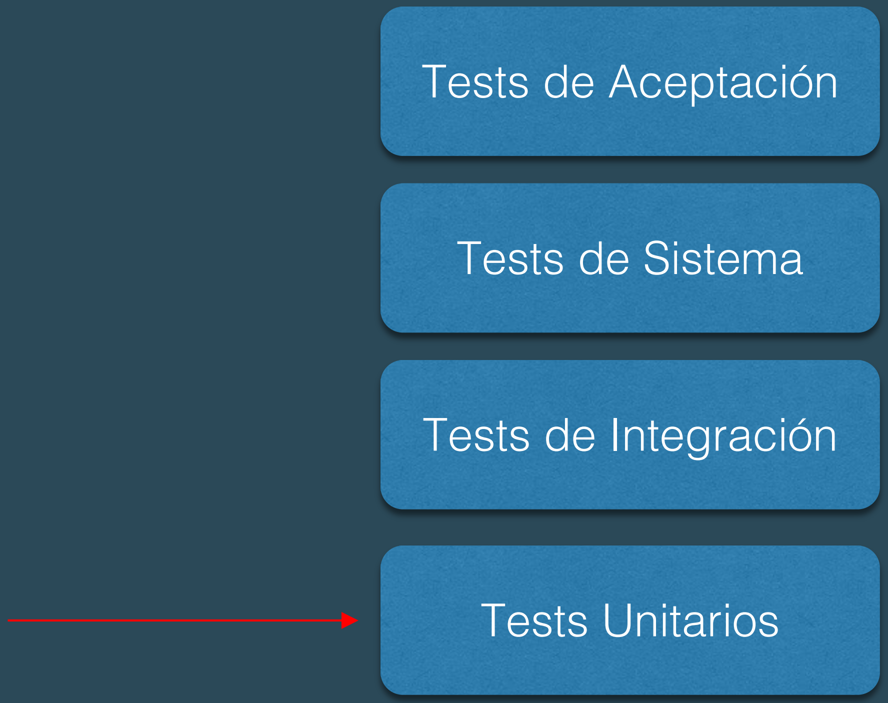
Testing

•••

Testing

- Proceso de ejecutar un programa con el objetivo de encontrar un error
- un buen caso de prueba es uno con una alta probabilidad de encontrar un error oculto
- un test exitoso es aquel que descubre un error que no se conocía

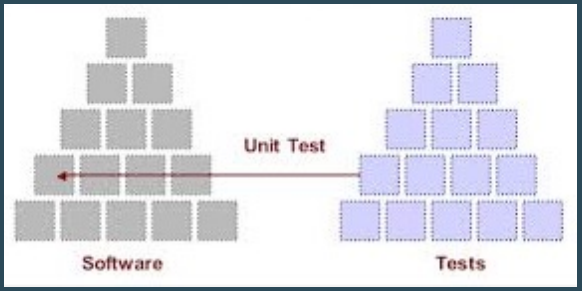
Jerarquía de Tests



Tests Unitarios

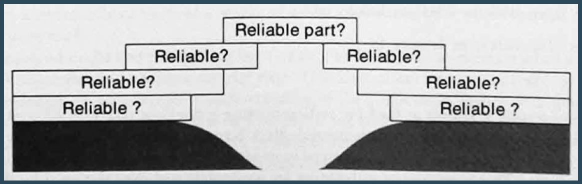
- Lo escribe el desarrollador al mismo tiempo que el código
- Pequeños
- Automatizados

- F – fast
- I – Isolated
- R – Repeatable
- S – Self-verifying
- T – Timely



I'm a programmer and #iwritebugs. All the time. Embarrassingly obvious bugs. Brain dead stupid. Fortunately, I also write tests.

Howard Lewis Ship



Tests Unitarios en Rails

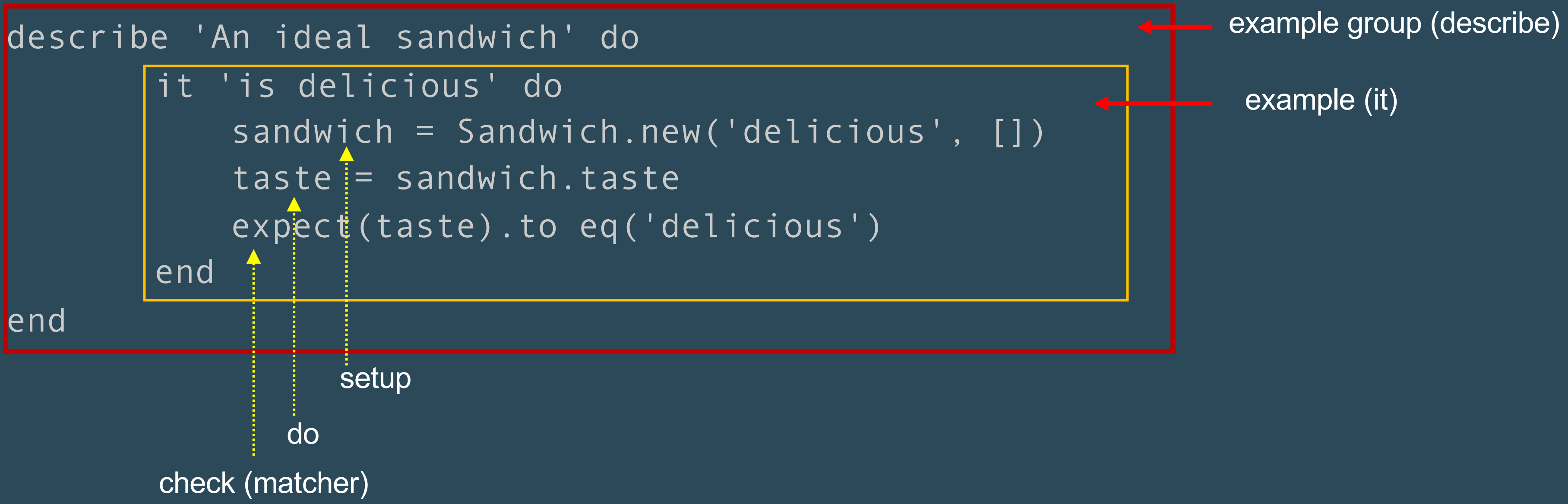
- Rails instala automáticamente una herramienta de testing llamada Minitest
- Para usar esa herramienta basta usar
  - rails new rails_minitest
- Nosotros usaremos Rspec que es la preferida por los desarrolladores profesionales
  - rails new rails_rspec
  - agregar las gemas correspondientes a RSpec

RSpec

Creada por Steven Baker en 2005

El programador debe escribir "specs" que describen el comportamiento esperado

Ejemplo (del texto Effective Testing with RSpec 3)

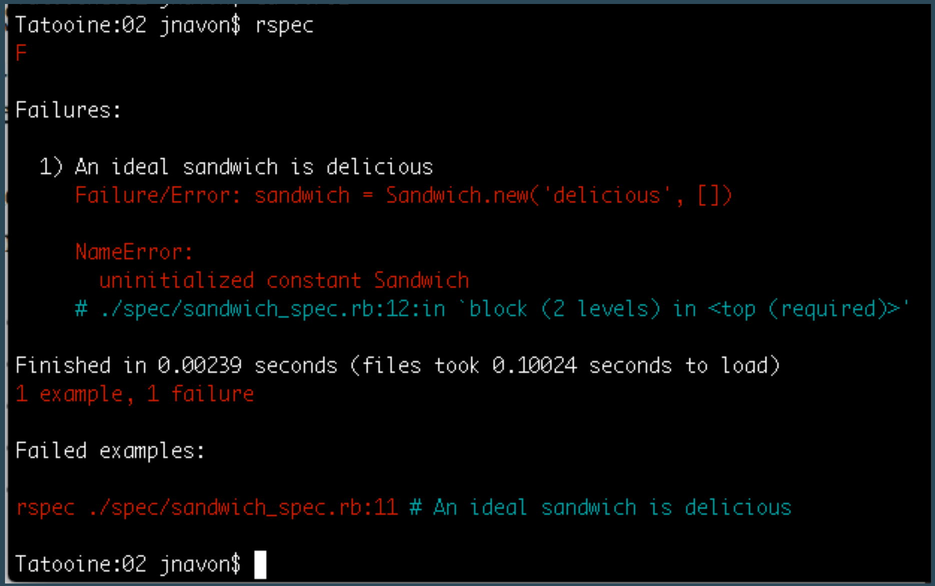


```
describe 'An ideal sandwich' do
    it 'is delicious' do
        sandwich = Sandwich.new('delicious', [])
        taste = sandwich.taste
        expect(taste).to eq('delicious')
    end
end
```

Terminología

- Un test valida que un trozo de código esté funcionando correctamente
- Un spec describe el comportamiento deseado de un trozo de código
- Un ejemplo muestra como se espera que una funcionalidad sea usada

Corriendo el test ...



```
Tatooine:02 jnavon$ rspec
F

Failures:

  1) An ideal sandwich is delicious
     Failure/Error: sandwich = Sandwich.new('delicious', [])

     NameError:
       uninitialized constant Sandwich
     # ./spec/sandwich_spec.rb:12:in `block (2 levels) in <top (required)>'

Finished in 0.00239 seconds (files took 0.10024 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./spec/sandwich_spec.rb:11 # An ideal sandwich is delicious

Tatooine:02 jnavon$ 
```

passing specs - green
failing specs - red
example descriptions and structural text - black
extra details - blue
pending specs - yellow

Testing con Rspec

Estructura de un test en Rspec

```
describe ‘something’ do

    it ‘do something’ do
       expect ‘result’ match ‘something else’
    end

end
```

Uso del Context

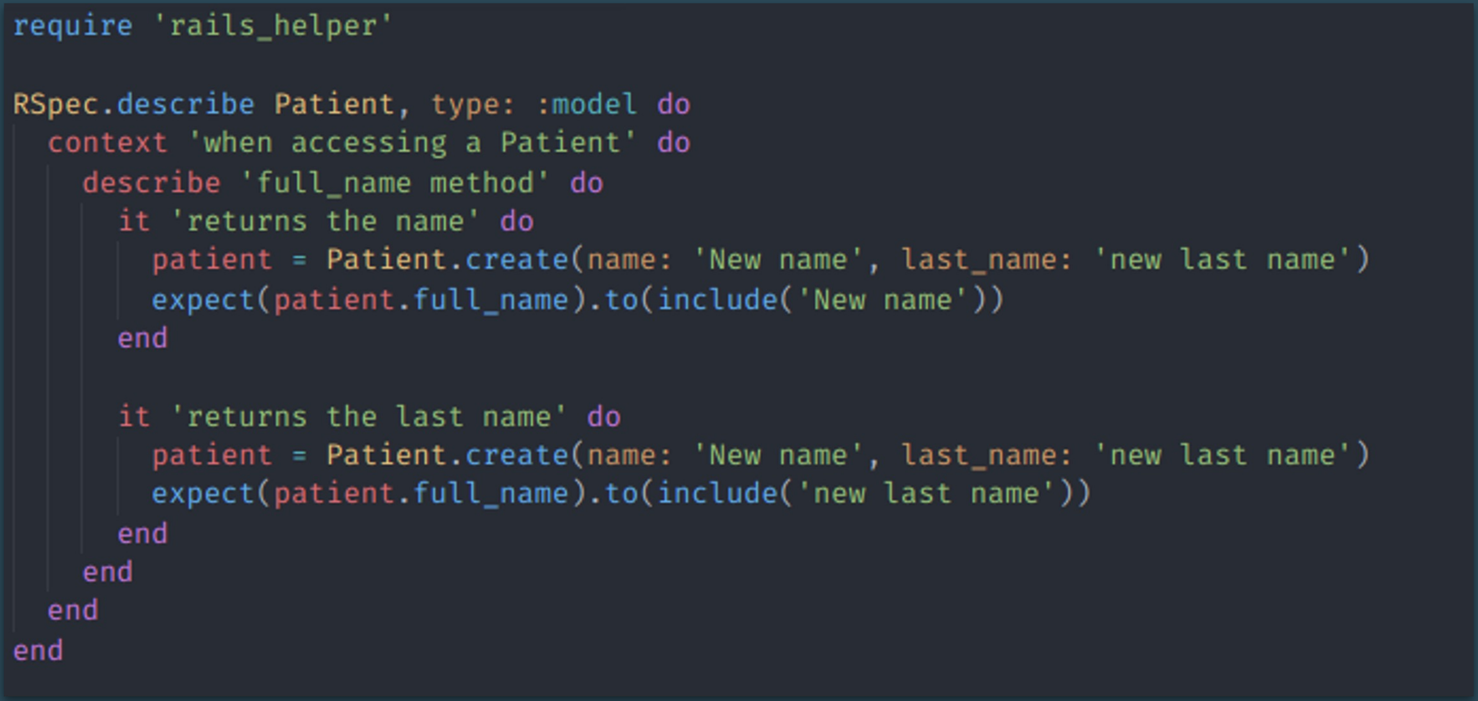
- Permite definir subgrupos de tests

```
describe Course do
  context "when user is logged in" do
    it "displays the course lessons" do
    end
    it "displays the course description" do
    end
  end

  context "when user it NOT logged in" do
    it "redirects to login page" do
    end
    it "it shows a message" do
    end
  end
end
```

tests con el usuario logeado

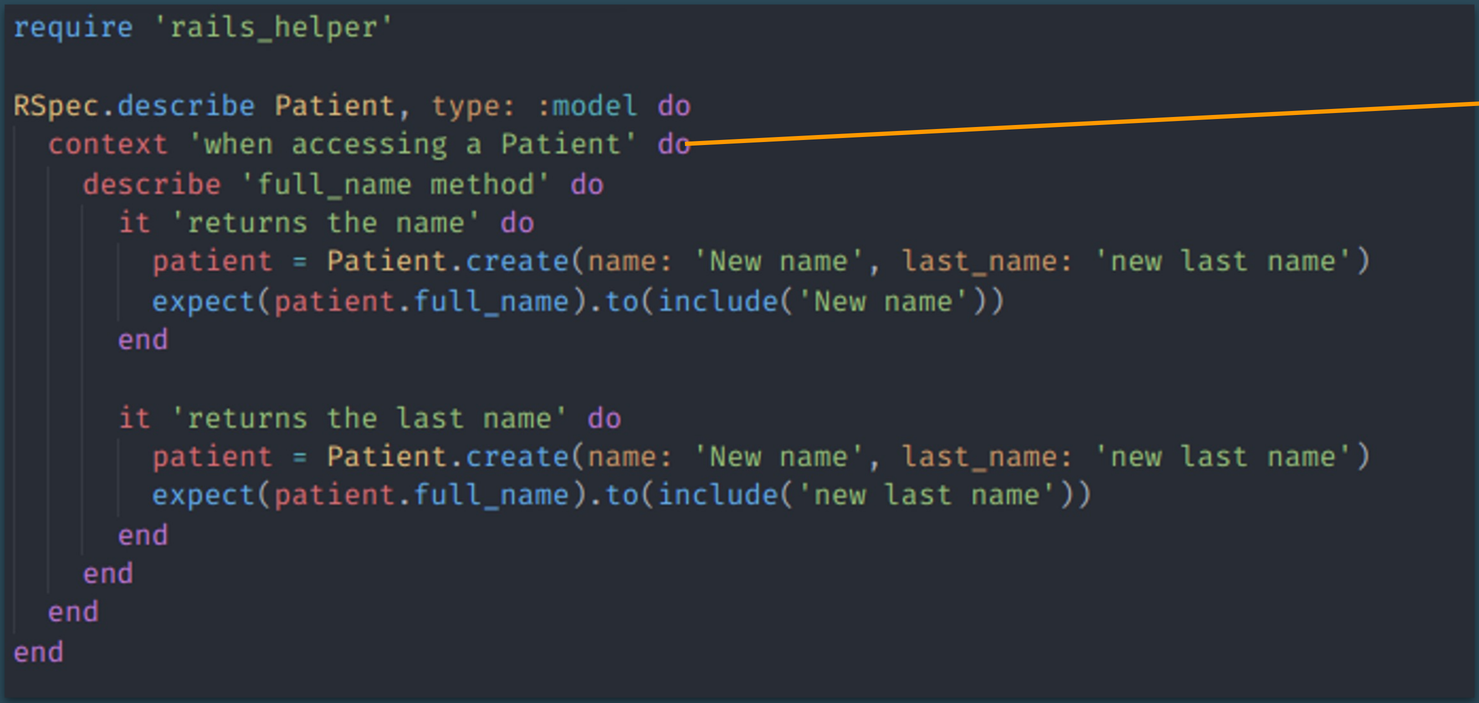
tests con el usuario no logeado

Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

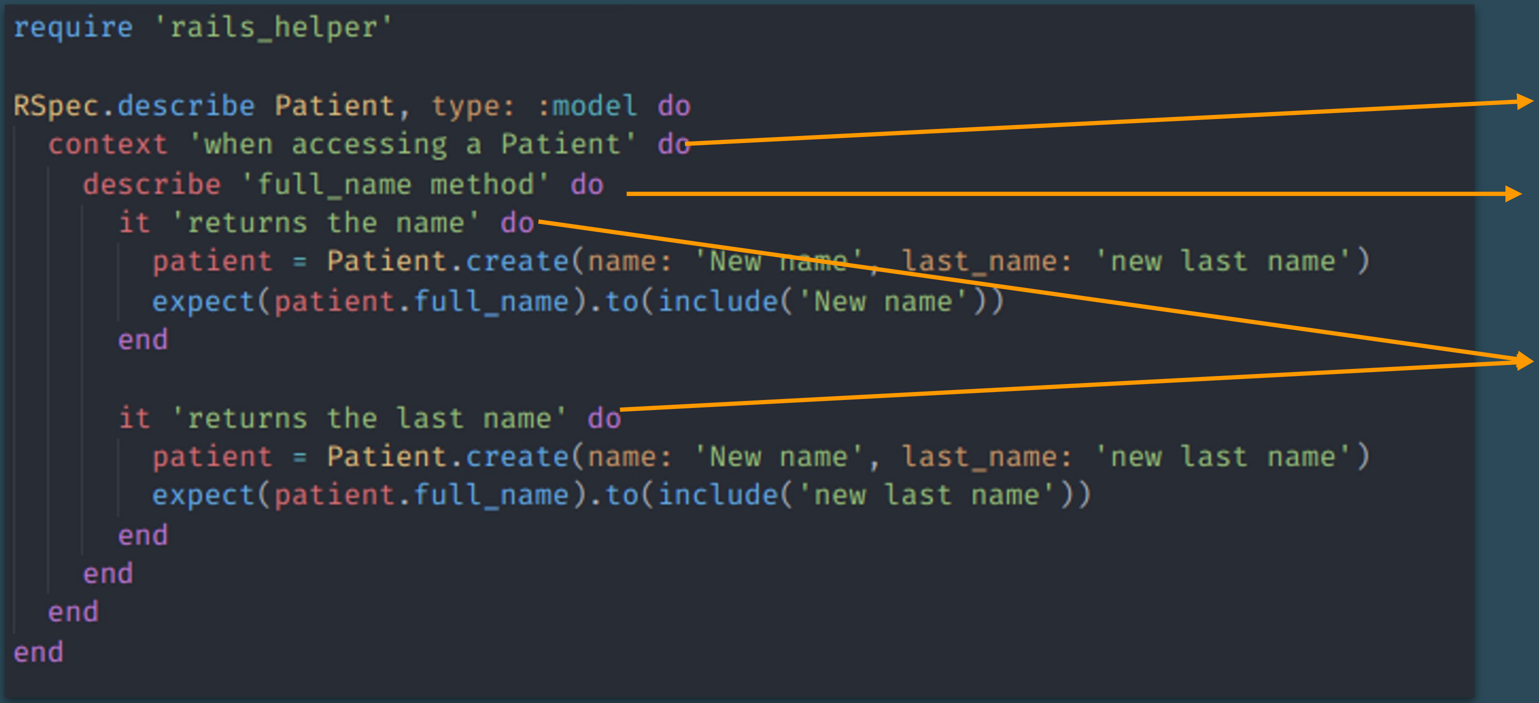
Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

Sirve para separar escenarios

Primeros tests para nuestro modelo Patient



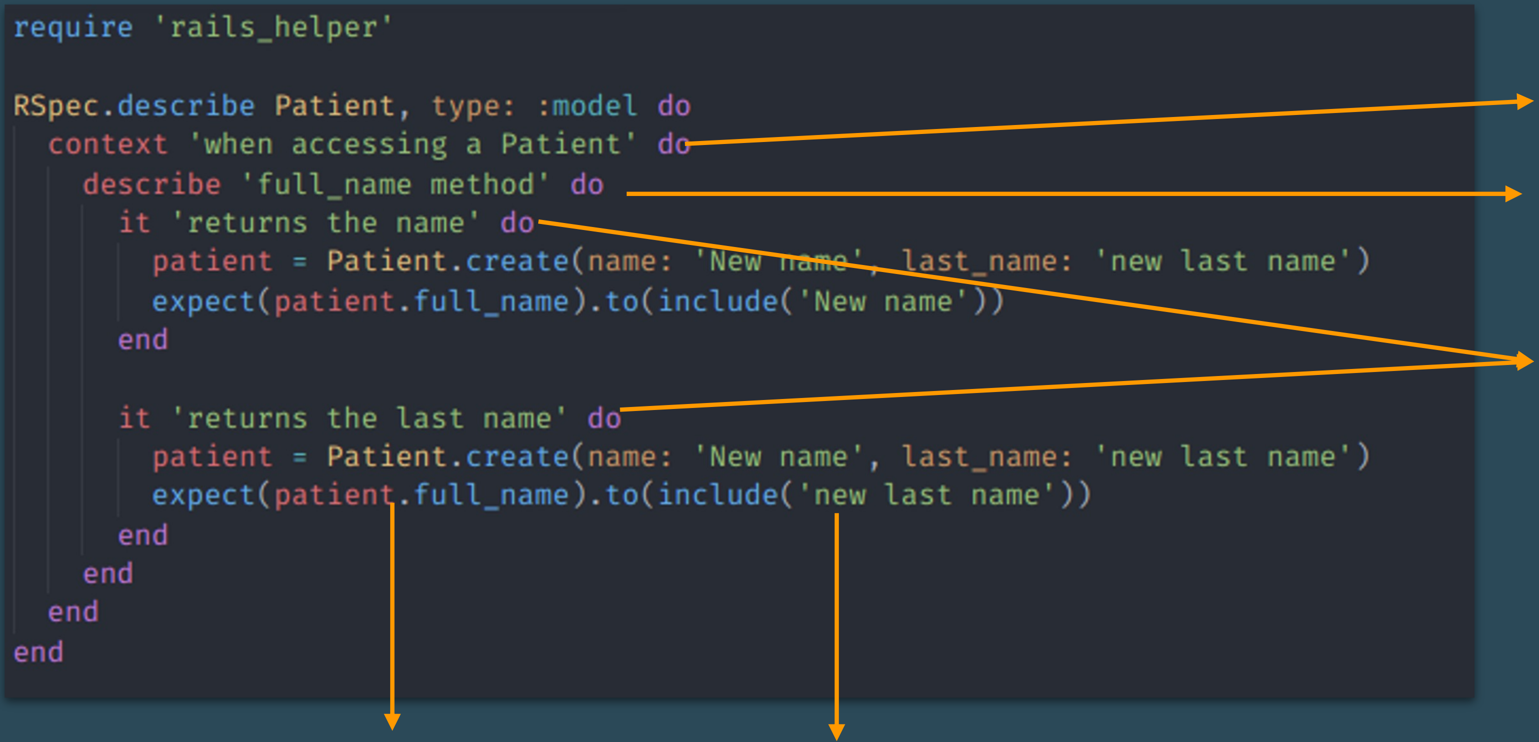
```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

Sirve para separar escenarios

Describe lo que se testea

Test

Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

Sirve para separar escenarios

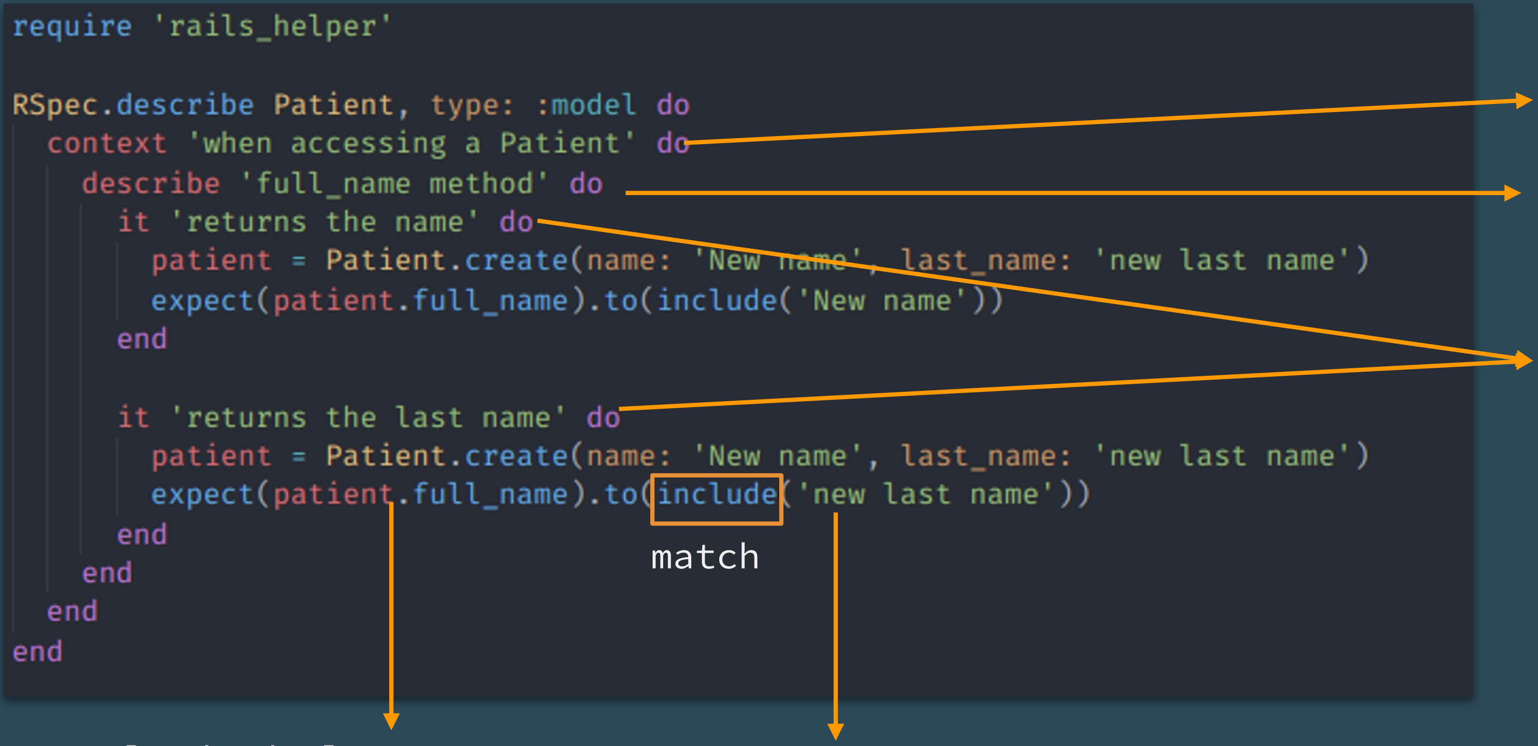
Describe lo que se testea

Test

Resultado de lo que testeo

Comparo con resultado esperado

Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

Sirve para separar escenarios

Describe lo que se testea

Test

Resultado de lo que testeo

Comparo con resultado esperado

Rspec Expectations

- Es lo que permite expresar lo que se supone debe obtenerse en un test
- Algunos de los más usados
  - expect(actual).to eq(expected)
  - 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)
  - expect(actual).to be true
  - expect(actual).to be false
- Mas detalles en

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

Matchers

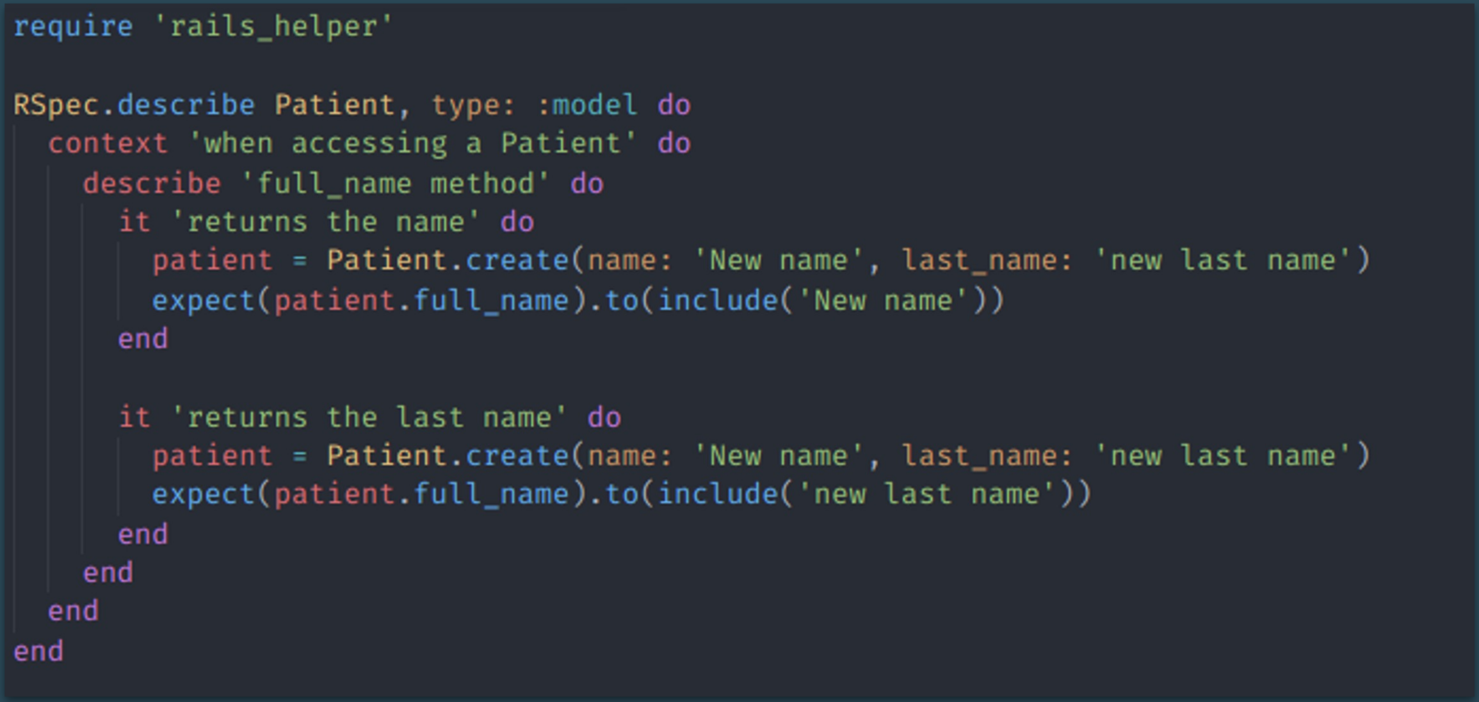
```
expect(result).to
expect(result).not_to
```

Matchers	
be eq > >= be_between be_instance_of match respond_to be true (== true) be_truthly (not false or nil) exists raise_error(ErrorClass) has_keys has_value be_empty include	be_a_new render_template redirect_to route_to have_http_status be_routable

Nos sirve para comparar el resultado de una operación con el resultado esperado

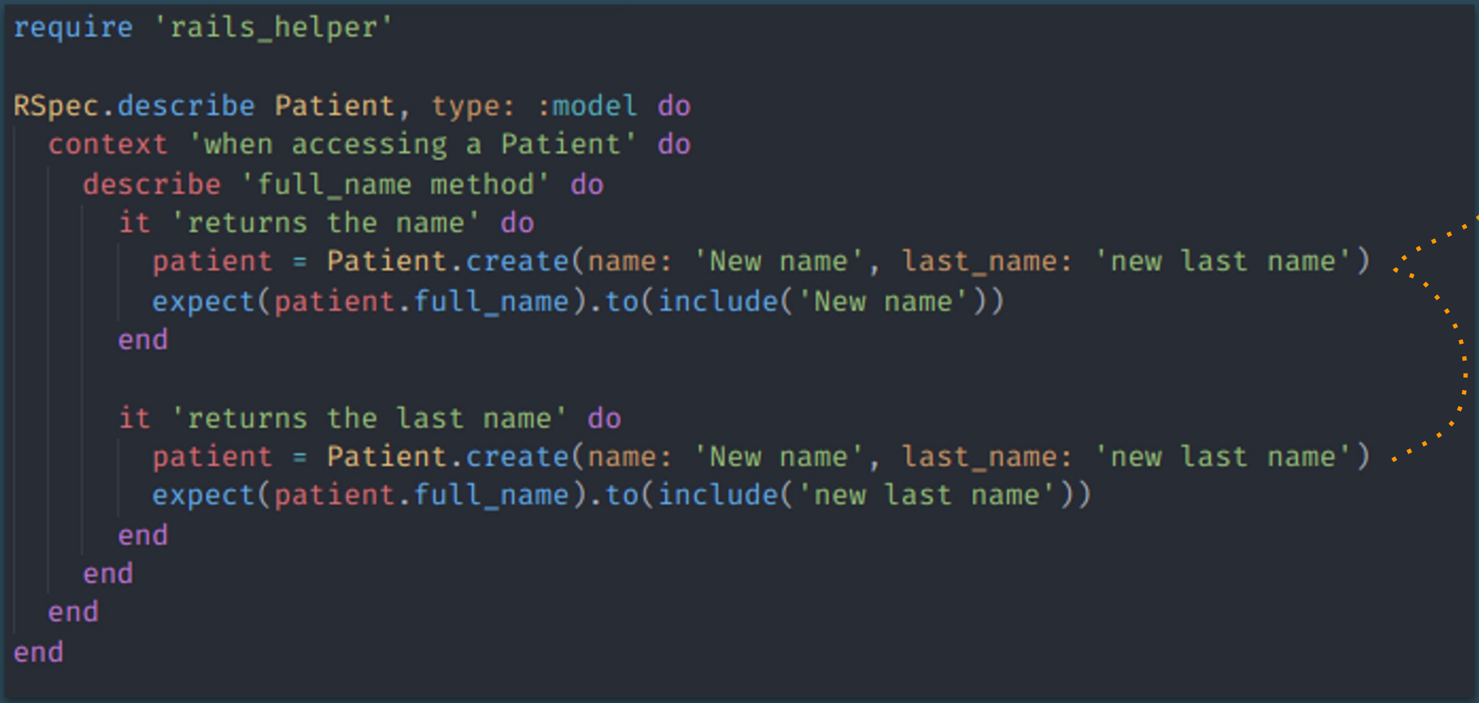
https://relishapp.com/rspec/rspec-expectations/docs/built-in-matchers

Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

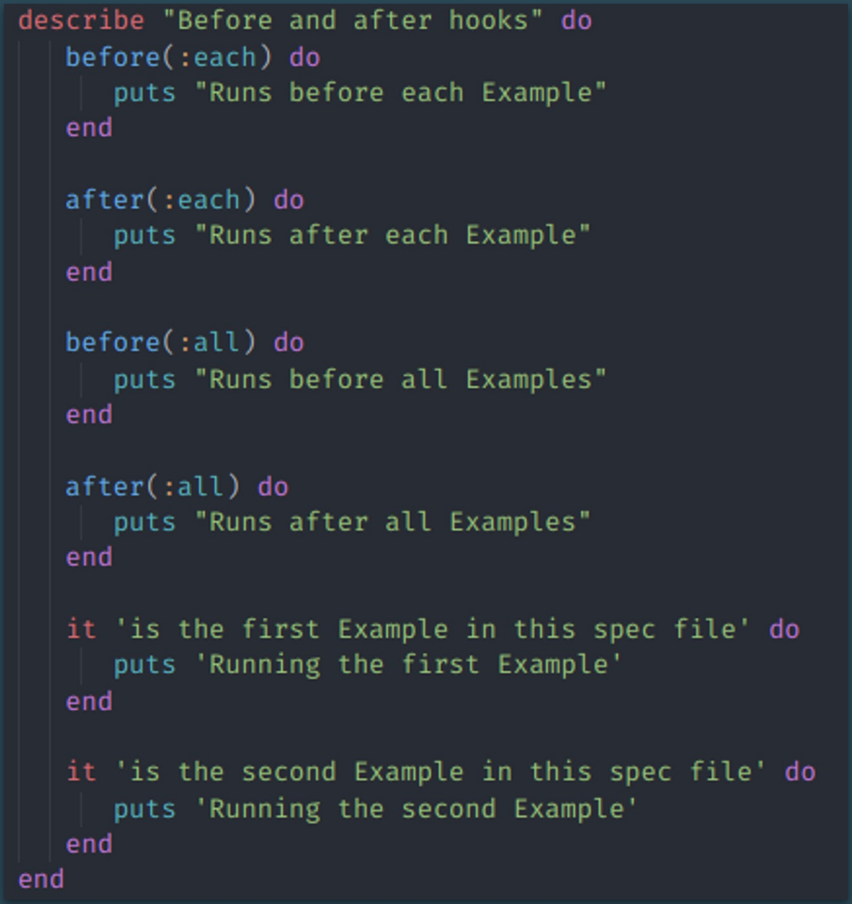
Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    describe 'full_name method' do
      it 'returns the name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('New name'))
      end

      it 'returns the last name' do
        patient = Patient.create(name: 'New name', last_name: 'new last name')
        expect(patient.full_name).to(include('new last name'))
      end
    end
  end
end
```

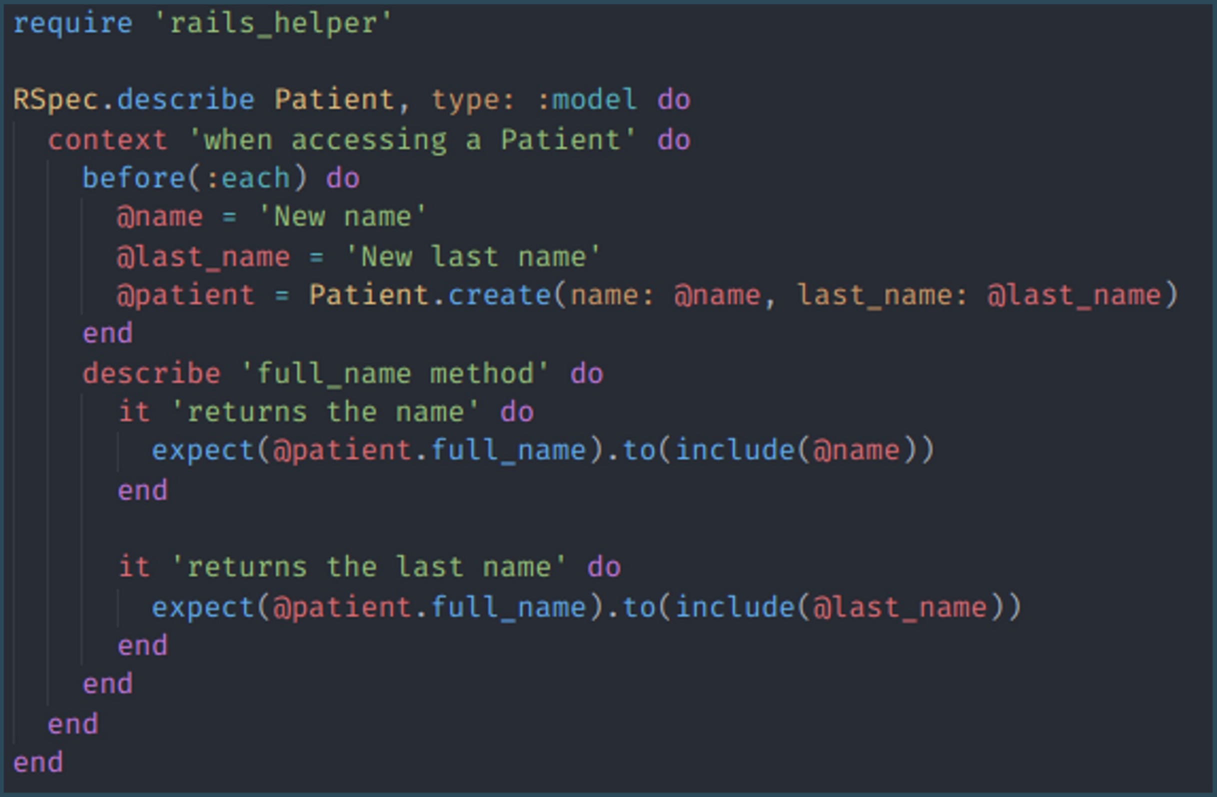
DRY!!!

Hooks



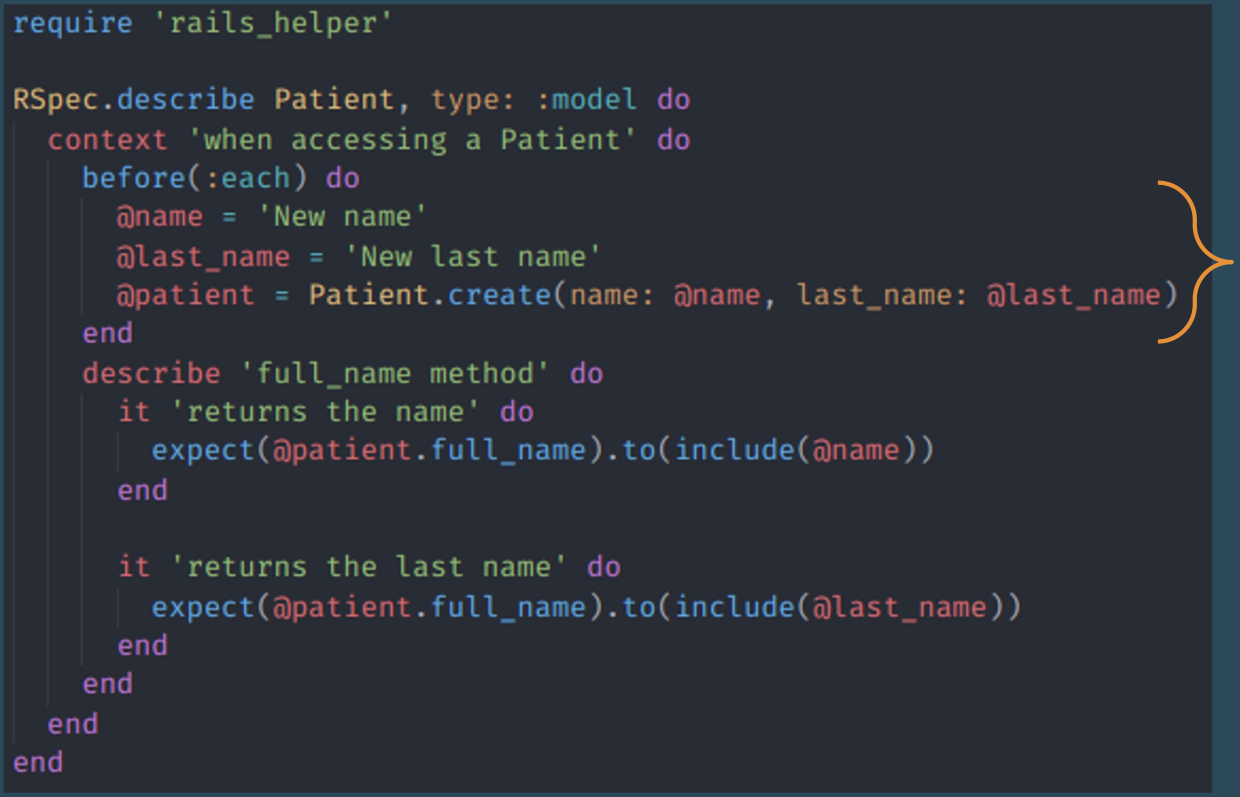
```ruby
describe "Before and after hooks" do
  before(:each) do
    puts "Runs before each Example"
  end

  after(:each) do
    puts "Runs after each Example"
  end

  before(:all) do
    puts "Runs before all Examples"
  end

  after(:all) do
    puts "Runs after all Examples"
  end

  it 'is the first Example in this spec file' do
    puts 'Running the first Example'
  end

  it 'is the second Example in this spec file' do
    puts 'Running the second Example'
  end
end
```

Primeros tests para nuestro modelo Patient



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    before(:each) do
      @name = 'New name'
      @last_name = 'New last name'
      @patient = Patient.create(name: @name, last_name: @last_name)
    end
    describe 'full_name method' do
      it 'returns the name' do
        expect(@patient.full_name).to(include(@name))
      end

      it 'returns the last name' do
        expect(@patient.full_name).to(include(@last_name))
      end
    end
  end
end
```

Primeros tests para nuestro modelo Patient

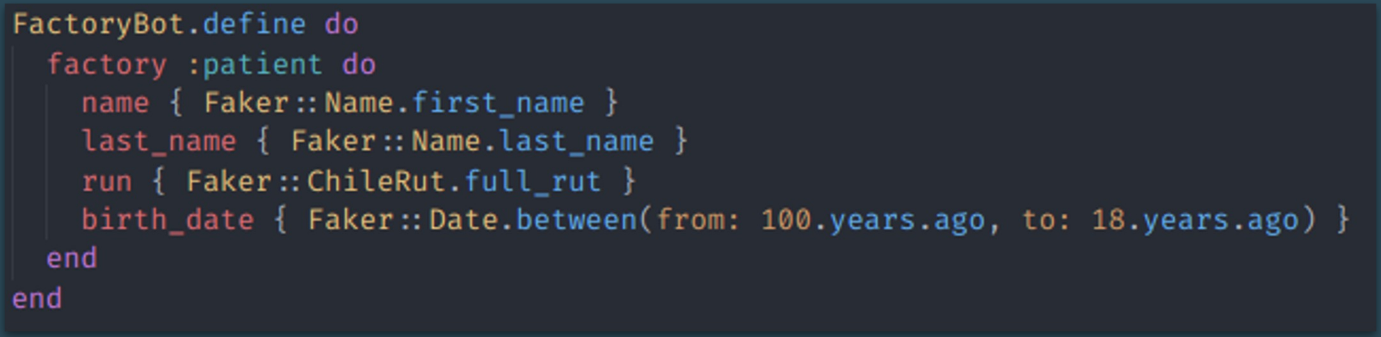


```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do
    before(:each) do
      @name = 'New name'
      @last_name = 'New last name'
      @patient = Patient.create(name: @name, last_name: @last_name)
    end
    describe 'full_name method' do
      it 'returns the name' do
        expect(@patient.full_name).to(include(@name))
      end

      it 'returns the last name' do
        expect(@patient.full_name).to(include(@last_name))
      end
    end
  end
end
```

$\}$ Existe una mejor manera

Factories

- Factory Method es un patrón de diseño de software
- Es una clase que centraliza la creación de objetos
- El desarrollador se olvida de los detalles de cómo crear un objeto, qué valores usar, etc. Sólo se preocupa de utilizar la fábrica o factory
- Con Rspec podemos usar la gema factory_bot

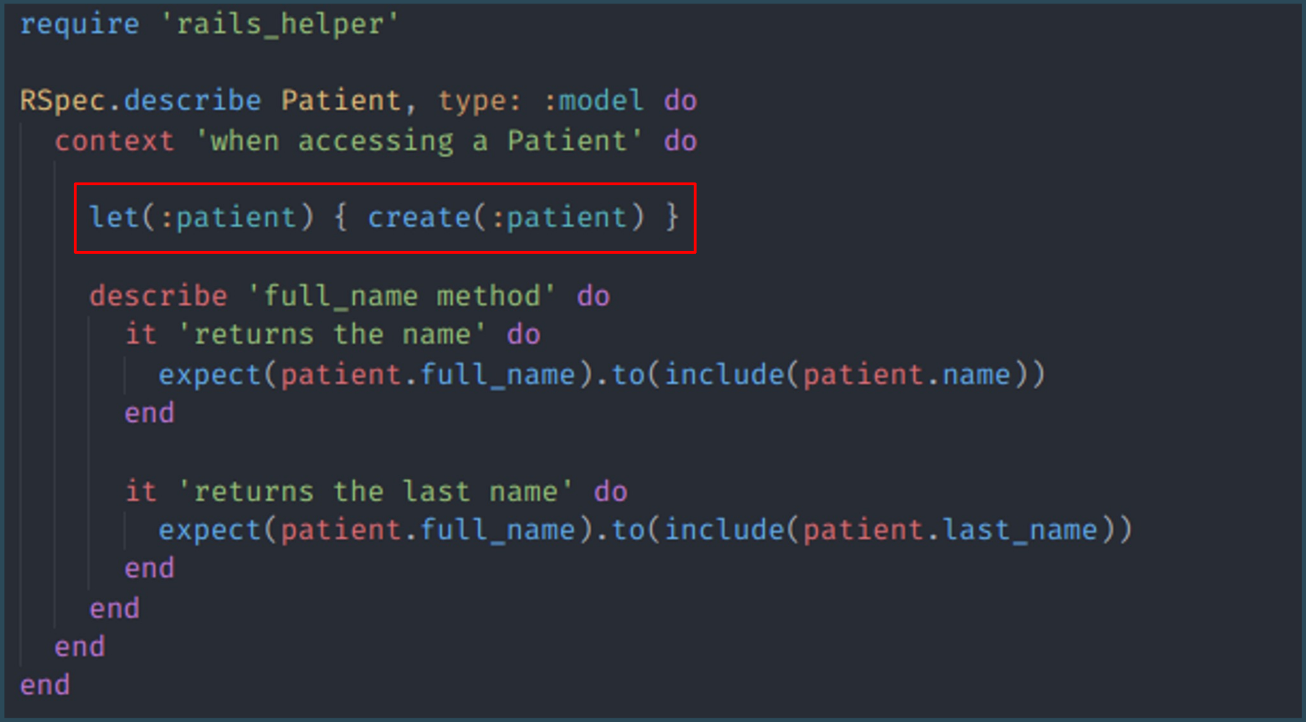


```ruby
FactoryBot.define do
  factory :patient do
    name { Faker::Name.first_name }
    last_name { Faker::Name.last_name }
    run { Faker::ChileRut.full_rut }
    birth_date { Faker::Date.between(from: 100.years.ago, to: 18.years.ago) }
  end
end
```

La gema Faker

- Permite generar data random
  - instalar con gem install faker
- Incluye muchos tipos de dato
- Entre ellos por ejemplo Name, Date, ChileRut, Color, etc.
- Detalles en https://github.com/faker-ruby/faker

Usando la factory



```ruby
require 'rails_helper'

RSpec.describe Patient, type: :model do
  context 'when accessing a Patient' do

    let(:patient) { create(:patient) }

    describe 'full_name method' do
      it 'returns the name' do
        expect(patient.full_name).to(include(patient.name))
      end

      it 'returns the last name' do
        expect(patient.full_name).to(include(patient.last_name))
      end
    end
  end
end
```

Uso del método Let

- Permite escribir varios tests usando el mismo objeto
- El objeto se crea la primera vez que se usa

```
describe Factorial do
  let(:calculator) { Factorial.new }
  it "finds the factorial of 5" do
    expect(calculator.factorial_of(5)).to eq(120)
  end
  it "finds the factorial of 0" do
    expect(calculator.factorial_of(0)).to eq(1)
  end
end
```

- subject es equivalente a let pero solo permite referirse a un objeto (implícito)

```
describe Factorial do
  it "finds the factorial of 5" do
    expect(subject.factorial_of(5)).to eq(120)
  end
end
```

Mas detalles de RSpec

https://rspec.info/
https://rspec.info/documentation/3.11/rspec-core/
https://devhints.io/rspec

Mas material ...

- Tutoriales Rspec
  - https://www.tutorialspoint.com/rspec/index.htm
  - https://rspec.info/documentation/5.0/rspec-rails/
  - https://dev.to/isalevine/intro-to-rspec-in-rails-basic-syntax-and-strategy-for-testing-3hh6
- Documentación de factory_bot y tutorial
  - https://github.com/thoughtbot/factory_bot/blob/master/GETTING_STARTED.md
  - https://semaphoreci.com/community/tutorials/working-effectively-with-data-factories-using-factorygirl
- Documentación de Faker
  - https://github.com/faker-ruby/faker