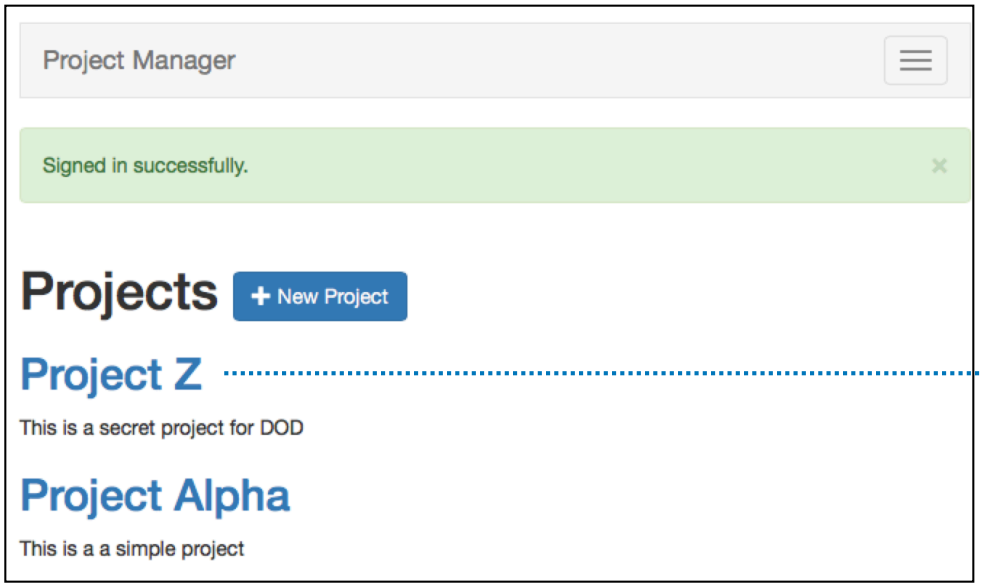
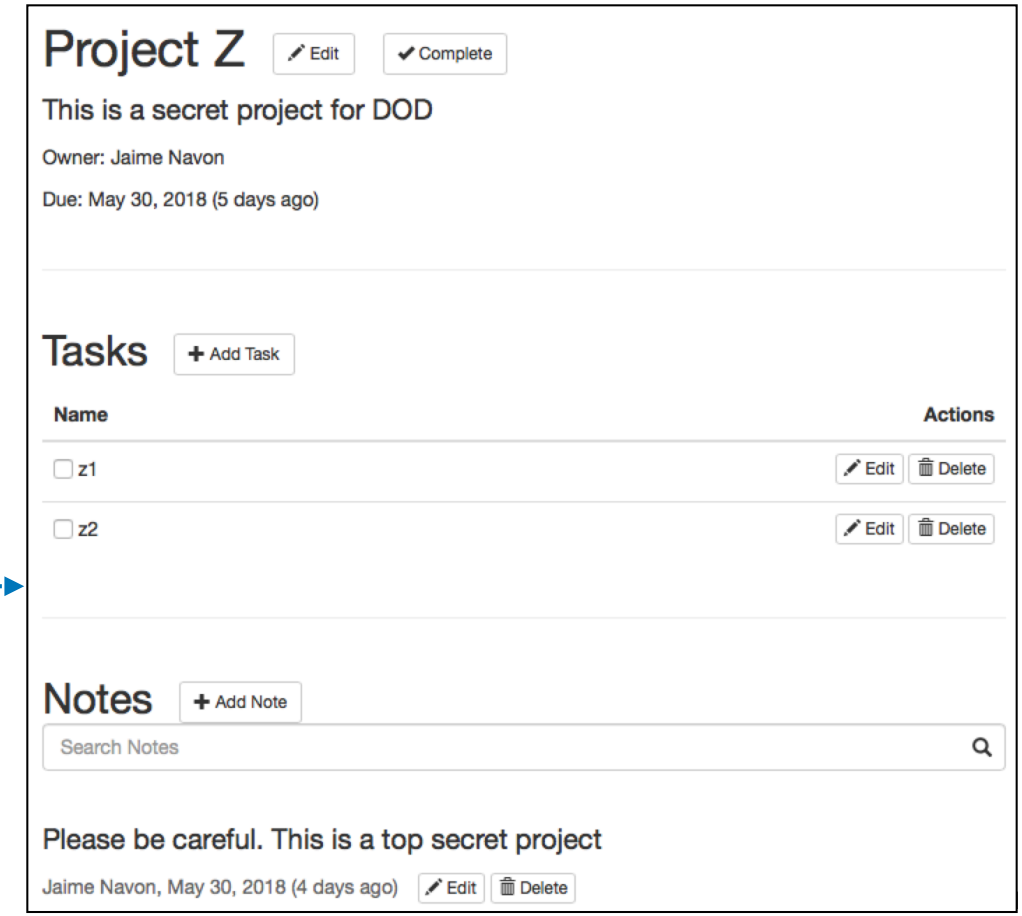
Rspec con Rails

- Ejemplo extraido de Everyday Rails Testing with RSpec de Aaron Sumner (Leanpub)

- Gestor de proyectos





Model Specs

- se describe un set de expectations para diferentes ejemplos
- cada ejemplo comienza con it y espera una sola cosa
- cada ejemplo es explícito (string de it es opcional pero útil)
- cada descripción comienza con un verbo

Ejemplo (User)

```
describe User do
  it "is valid with a first name, last name, email, and password"
  it "is invalid without a first name"
  it "is invalid without a last name"
  it "is invalid without an email address"
  it "is invalid with a duplicate email address"
  it "returns a user's full name as a string"
end
```

- Leer en voz alta:
  - User is invalid without a first name
  - User is invalid without a last name
  - User returns a user´s full name as a string
  - etc

User

```
RSpec.describe User, type: :model do
  it "is valid with a first name, last name, email, and password" do

    user = User.new(
      first_name: "Aaron",
      last_name:  "Sumner",
      email:      "tester@example.com",
      password:   "dottle-nouveau-pavilion-tights-furze",
     )

     expect(user).to be_valid

  end

  it "is invalid without a first name"
  it "is invalid without a last name"
  it "is invalid without an email address"
  it "is invalid with a duplicate email address"
  it "returns a user's full name as a string"
end
```

Al correr rspec

```
User
  is valid with a first name, last name and email, and password
  is invalid without a first name (PENDING: Not yet implemented)
  is invalid without a last name (PENDING: Not yet implemented)
  is invalid without an email address (PENDING: Not yet implemented)
  is invalid with a duplicate email address (PENDING: Not yet implemented)
  returns a user's full name as a string (PENDING: Not yet implemented)
Pending: (Failures listed here are expected and do not affect your
suite's status)
  1) User is invalid without a first name
     # Not yet implemented
     # ./spec/models/user_spec.rb:14
  2) User is invalid without a last name
     # Not yet implemented
     # ./spec/models/user_spec.rb:15
  3) User is invalid without an email address
     # Not yet implemented
     # ./spec/models/user_spec.rb:16
  4) User is invalid with a duplicate email address
     # Not yet implemented
     # ./spec/models/user_spec.rb:17
  5) User returns a user's full name as a string
     # Not yet implemented
     # ./spec/models/user_spec.rb:18
Finished in 0.02839 seconds (files took 0.28886 seconds to load)
6 examples, 0 failures, 5 pending
```

Shoulda.matchers

- Gema permite simplificar las specs
- Por ejemplo validación de presencia

```
it { is_expected.to validate_presence_of :first_name }
it { is_expected.to validate_presence_of :last_name }
it { is_expected.to validate_presence_of :email }
it { is_expected.to validate_uniqueness_of(:email).case_insensitive }
```

ActiveModel matchers

- allow_value tests that an attribute is valid or invalid if set to one or more values. *(Aliased as #allow_values.)*
- have_secure_password tests usage of `has_secure_password`.
- validate_absence_of tests usage of `validates_absence_of`.
- validate_acceptance_of tests usage of `validates_acceptance_of`.
- validate_confirmation_of tests usage of `validates_confirmation_of`.
- validate_exclusion_of tests usage of `validates_exclusion_of`.
- validate_inclusion_of tests usage of `validates_inclusion_of`.
- validate_length_of tests usage of `validates_length_of`.
- validate_numericality_of tests usage of `validates_numericality_of`.
- validate_presence_of tests usage of `validates_presence_of`.

ActiveRecord matchers

- accept_nested_attributes_for tests usage of the `accepts_nested_attributes_for` macro.
- belong_to tests your `belongs_to` associations.
- define_enum_for tests usage of the `enum` macro.
- have_and_belong_to_many tests your `has_and_belongs_to_many` associations.
- have_db_column tests that the table that backs your model has a specific column.
- have_db_index tests that the table that backs your model has an index on a specific column.
- have_many tests your `has_many` associations.
- have_one tests your `has_one` associations.
- have_readonly_attribute tests usage of the `attr_readonly` macro.
- serialize tests usage of the `serialize` macro.
- validate_uniqueness_of tests usage of `validates_uniqueness_of`.

ActionController matchers

- filter_param tests parameter filtering configuration.
- permit tests that an action places a restriction on the `params` hash.
- redirect_to tests that an action redirects to a certain location.
- render_template tests that an action renders a template.
- render_with_layout tests that an action is rendered with a certain layout.
- rescue_from tests usage of the `rescue_from` macro.
- respond_with tests that an action responds with a certain status code.
- route tests your routes.
- set_session makes assertions on the `session` hash.
- set_flash makes assertions on the `flash` hash.
- use_after_action tests that an `after_action` callback is defined in your controller.
- use_around_action tests that an `around_action` callback is defined in your controller.
- use_before_action tests that a `before_action` callback is defined in your controller.

Independent matchers

- delegate_method tests that an object forwards messages to other, internal objects by way of delegation.

Datos para los tests

- Lo más simple, usar POROs (plain old ruby objects)
- Puede ser engorroso cuando la data es mas compleja
- La popular gema Factory Girl es una de las mas utilizadas para ésto
- El uso de "factories" es una alternativa al uso de "fixtures" provista por Rails

Fixtures

- Archivos .yml que se usan para crear objetos

projects.yml

```
death_star:
  name: "Death Star"
  description: "Create the universe's ultimate battle station"
  due_on: 2016-08-29

rogue_one:
  name: "Steal Death Star plans"
  description: "Destroy the Empire's new battle station"
  due_on: 2016-08-29
```

- Entonces en un test puedo hace referencia a projects(:rogue_one) y recibo un objeto con los datos del segundo
- Problema: recordar cambios en fixtures, bypass validaciones
- Mejor usar factories

Generando Factories

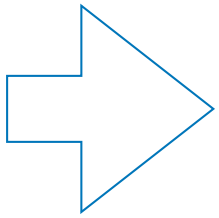
- factory_girl agrega un generador de factories
- nuevo directorio en spec de nombre factories

$ rails g factory_girl :model user

spec/factories/users.rb

```
FactoryGirl.define do
  factory :user do

  end
end
```



```
FactoryGirl.define do
  factory :user do
    first_name "Aaron"
    last_name  "Sumner"
    email "tester@example.com"
    password "dottle-nouveau-pavilion-tights-furze"
  end
end
```

- ahora podemos crear un usuario en un test mediante Factory-Girl.create(:user)

Versatilidad de las factories

spec/models/user_spec.rb

```ruby
it "is invalid without a first name" do
  user = FactoryGirl.build(:user, first_name: nil)
  user.valid?
  expect(user.errors[:first_name]).to include("can't be blank")
end

it "is invalid without a last name" do
  user = FactoryGirl.build(:user, last_name: nil)
  user.valid?
  expect(user.errors[:last_name]).to include("can't be blank")
end

it "is invalid without an email address" do
  user = FactoryGirl.build(:user, email: nil)
  user.valid?
  expect(user.errors[:email]).to include("can't be blank")
end
```

spec/factories/users.rb

```ruby
FactoryGirl.define do
  factory :user do
    first_name "Aaron"
    last_name  "Sumner"
    sequence(:email) { |n| "tester#{n}@example.com" }
    password "dottle-nouveau-pavilion-tights-furze"
  end
end
```

Mas detalles

Factory Girl documentation

https://github.com/thoughtbot/factory_girl/blob/master/GETTING_STARTED.md

Nota:
A fines del 2017 la gema FactoryGirl pasa a llamarse FactoryBot

¿Por qué?

"Its original name was confusing and potentially uncomfortable. Given the gender imbalance in the industry, anything named "girl" has the potential to cause alienation."

Controller Specs

- A partir de Rails 5.0 pasan a ser deprecadas
- Se sugiere pasar la mayor parte de los tests a los modelos o a tests de integración
- Todavía son ampliamente utilizados

spec/controllers/projects_controller_spec.rb

```
require 'rails_helper'

RSpec.describe ProjectsController, type: :controller do
  describe "#index" do
    context "as an authenticated user" do
      before do
        @user = FactoryGirl.create(:user)
      end

      it "responds successfully" do
        sign_in @user
        get :index
        expect(response).to be_success
      end

      it "returns a 200 response" do
        sign_in @user
        get :index
        expect(response).to have_http_status "200"
      end
    end

    context "as a guest" do
      # tests will go here
    end
  end
end
```

Feature Specs

- Cómo modelos y controladores funcionan para producir funcionalidades que le interesan al usuario
- Validación de que el software funciona como realmente se espera que lo haga desde perspectiva del usuario
- Se requiere de otra gema: Capybara
- Capybara permite simular el uso de la aplicación mediante métodos como click_link, visit, etc.

Un feature spec básico

```ruby
require 'rails_helper'

RSpec.feature "Projects", type: :feature do
  scenario "user creates a new project" do
    user = FactoryGirl.create(:user)

    visit root_path
    click_link "Sign in"
    fill_in "Email", with: user.email
    fill_in "Password", with: user.password
    click_button "Log in"

    expect {
      click_link "New Project"
      fill_in "Name", with: "Test Project"
      fill_in "Description", with: "Trying out Capybara"
      click_button "Create Project"

      expect(page).to have_content "Project was successfully created"
      expect(page).to have_content "Test Project"
      expect(page).to have_content "Owner: #{user.name}"
    }.to change(user.projects, :count).by(1)
  end
end
```

Otros métodos de Capybara

```
scenario "works with all kinds of HTML elements" do
  visit "/fake/page"
  click_on "A link or button label"
  check "A checkbox label"
  uncheck "A checkbox label"
  choose "A radio button label"
  select "An option", from: "A select menu"
  attach_file "A file upload label", "/some/file/in/my/test/suite.gif"

  expect(page).to have_css "h2#subheading"
  expect(page).to have_selector "ul li"
  expect(page).to have_current_path "/projects/new"
end
```

Testing the API provided

- van en spec/requests
- no se usa Capybara (no hay interacción con el browser)
- Uso de métodos http simples: get, post, delete

```ruby
require 'rails_helper'

describe 'Projects API', type: :request do
  it 'loads a project' do
    user = FactoryGirl.create(:user)
    FactoryGirl.create(:project,
      name: "Sample Project")
    FactoryGirl.create(:project,
      name: "Second Sample Project",
      owner: user)

    get api_projects_path, params: {
      user_email: user.email,
      user_token: user.authentication_token
    }
    expect(response).to have_http_status(:success)
    json = JSON.parse(response.body)
    expect(json.length).to eq 1
    project_id = json[0]["id"]

    get api_project_path(project_id), params: {
      user_email: user.email,
      user_token: user.authentication_token
    }

    expect(response).to have_http_status(:success)
    json = JSON.parse(response.body)
    expect(json["name"]).to eq "Second Sample Project"
    # Etc.
  end
end
```

Un Post

```ruby
require 'rails_helper'

describe 'Projects API', type: :request do

  # First example omitted ...

  it 'creates a project' do
    user = FactoryGirl.create(:user)

    project_attributes = FactoryGirl.attributes_for(:project)

    expect {
      post api_projects_path, params: {
        user_email: user.email,
        user_token: user.authentication_token,
        project: project_attributes
      }
    }.to change(user.projects, :count).by(1)

    expect(response).to have_http_status(:success)
  end
end
```

Mas recursos

Documentación Oficial de RSpec	https://www.relishapp.com/ rspec-rails
Libro	https://pragprog. com/book/rspec3/effective-testing-with-rspec-3
Better Specs	http://betterspecs.org
The right way	https://www.pluralsight.com/courses/rspec-the-right-way
Railcasts	http://railscasts.com/?tag_id=7
Google Group	http://groups.google.com/ group/rspec