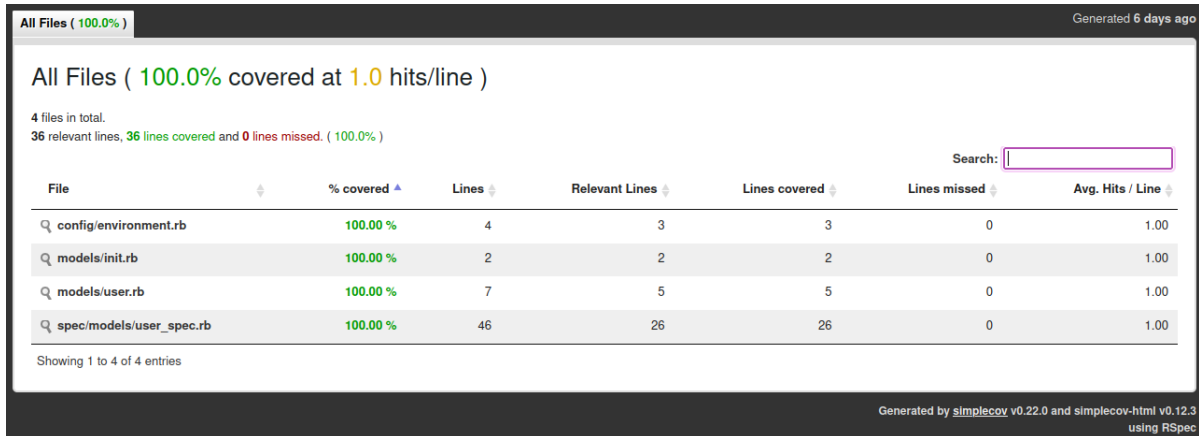


# Informe Ingeniería de software

## Actividad Nro 1

Instalamos y ejecutamos la herramienta SimpleCov en nuestro proyecto “Aprendiendo con Fitito” , donde demostramos la cobertura de nuestros test.

Apenas comenzamos con SimpleCov teníamos esta cobertura de código



The screenshot shows the SimpleCov report interface. At the top, it says "All Files ( 100.0% )" and "Generated 6 days ago". Below this, a summary states "All Files ( 100.0% covered at 1.0 hits/line )". It also mentions "4 files in total." and "36 relevant lines, 36 lines covered and 0 lines missed. ( 100.0% )". A search bar is present. The main table lists the files and their coverage details.

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
config/environment.rb	100.00 %	4	3	3	0	1.00
models/init.rb	100.00 %	2	2	2	0	1.00
models/user.rb	100.00 %	7	5	5	0	1.00
spec/models/user_spec.rb	100.00 %	46	26	26	0	1.00

Showing 1 to 4 of 4 entries

Generated by simplecov v0.22.0 and simplecov-html v0.12.3 using RSpec

El user\_spec.rb comenzamos con

```

spec > models > user_spec.rb
1  require_relative '../models/init.rb'
2  require 'sinatra/activerecord'
3
4  describe 'User' do
5    describe 'valid' do
6      describe 'when there is no email' do
7        it 'should be invalid' do
8          u = User.new
9          expect(u.valid?).to eq(false)
10         end
11       end
12     end
13   end
14
15   describe 'User' do
16     describe 'valid' do
17       describe 'when there is no username' do
18         it 'should be invalid' do
19           u = User.new(email: 'maria@example.com')
20           expect(u.valid?).to eq(false)
21         end
22       end
23
24       describe 'when the password is too short' do
25         it 'should be invalid' do
26           u = User.new(username: 'juancito', email: 'juan@example.com', password: '1234')
27           expect(u.valid?).to eq(false)
28         end
29       end
30
31       describe 'when the username is too short' do
32         it 'should be invalid' do
33           u = User.new(username: 'ab', email: 'juan@example.com')
34           expect(u.valid?).to eq(false)
35         end
36       end
37
38       describe 'when the email domain is invalid' do
39         it 'should be invalid' do
40           u = User.new(username: 'luis_cabral', email: 'luisito@example')
41           expect(u.valid?).to eq(false)
42         end
43       end
44     end
45   end

```

Luego comenzamos a hacer nuevos test en user, option y exam, agregando módulos dentro de exam.rb para pasar los test cuando una examen es válido, la suma y resta de puntos y resta de vidas según definida en nuestra aplicación. También los agregamos a nuestro server para poder así tener una mejor definición de casos y tener un código más legible.

```

models > exam.rb
1  class Exam < ActiveRecord::Base
2      has_and_belongs_to_many :users
3      has_and_belongs_to_many :questions
4
5      validates :score, numericality: { greater_than_or_equal_to: 0 }
6      def isValid
7          | return life > 0
8      end
9      def sumaPuntos
10         | return score + 10
11     end
12     def restaPuntos
13         | if score == 0
14             | return 0
15         else
16             | return score - 5
17         end
18     end
19     def restaVida
20         | return life - 1
21     end
22 end

```

En los test de option nos fijamos si tiene 0, 1, 2 o 3 opciones y identificar así su validez que es únicamente cuando hay 3 opciones. En el test de user agregamos que marque como invalido cuando falta @ en el mail del usuario, también un test de que deba tener todos los atributos para ser válido.

Logrando así la cobertura de los test y teniendo como resultado

All Files ( 100.0% ) Generated about 3 hours ago

All Files ( 100.0% covered at 1.05 hits/line )

9 files in total.  
152 relevant lines, 152 lines covered and 0 lines missed. ( 100.0% )

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
config/environment.rb	100.00 %	4	3	3	0	1.00
models/exam.rb	100.00 %	22	14	14	0	1.50
models/init.rb	100.00 %	5	5	5	0	1.00
models/option.rb	100.00 %	8	6	6	0	1.00
models/question.rb	100.00 %	7	5	5	0	1.00
models/user.rb	100.00 %	7	5	5	0	1.00
spec/models/exam_spec.rb	100.00 %	80	54	54	0	1.00
spec/models/option_spec.rb	100.00 %	39	24	24	0	1.00
spec/models/user_spec.rb	100.00 %	62	36	36	0	1.00

Showing 1 to 9 of 9 entries

Generated by simplecov v0.22.0 and simplecov-html v0.12.3  
using RSpec

Consideramos que nuestro software tenga robustez y calidad en las pruebas, logrando así una cobertura de código y reduciendo posibles vulnerabilidades.