# Taller IS 2023 - Actividad Nro: 1



Materia: Ingeniería de Software

**Profesores:** Marcela Daniele - Marcelo Uva - Ariel Arsaute - Daniela Solivellas **Alumnos:** Balestra Edgar Agustin - Bernardi Quiroga Matias - Olivo Lucas Gabriel

### Metodología

La metodología que utilizamos para medir la cobertura de nuestro código fue hacer uso de "test unitarios" o "test de unidad" escritos en Ruby usando el framework RSpec. Esta metodología implica aislar componentes individuales del código y comprobarlos de manera independiente para asegurarse de que funcionen correctamente.

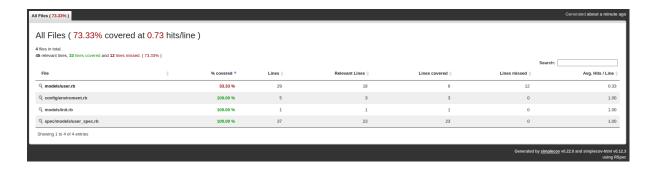
La cobertura fue medida utilizando la herramienta de cobertura simpleCov

#### **Test para Model User**

Durante el final del primer cuatrimestre, implementamos pruebas sobre el modelo User, en el que este debía contener un nombre y email únicos, o sea que no puede haber otro de usuario que utilice alguno de estos, y también debía contar con una password.

Además, en el modelo de usuario se encontraban dos funciones que son utilizadas en Schema para sumarle o restarle puntos al usuario adecuadamente cuando responde una pregunta.

Nuestros primeros tests sólo abarcaban la parte de validación del usuario, cubriendo solo el 33,33% de cobertura del modelo User, ya que no teníamos casos de prueba para cubrir las funciones de puntos del usuario. Una cobertura muy baja de acuerdo al modelo que teníamos implementado.

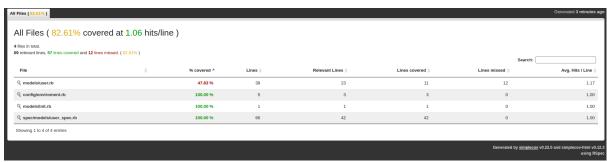


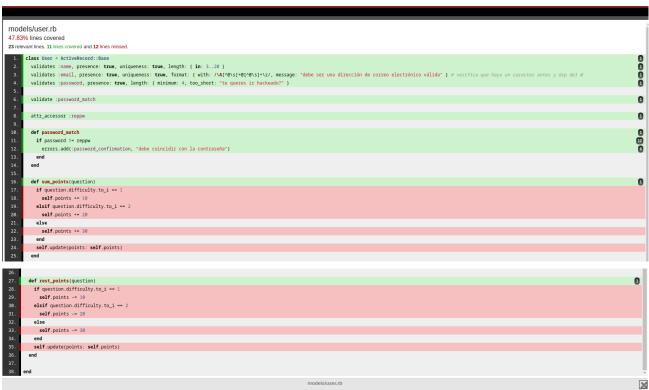
```
models/user.rb
33.33% lines covered
18 relevant lines. 6 lines covered and 12 lines missed.
 1. class User < ActiveRecord::Base
                                                                                                                                                                                                            1
      validates :name, presence: true, uniqueness: true validates :email, presence: true, uniqueness: true validates :password, presence: true
                                                                                                                                                                                                            8
      def sum_points (questio
                                                                                                                                                                                                            0
       if question.difficulty.to_i == 1
self.points += 10
elsif question.difficulty.to_i == 2
self.points += 20
else
        else
        else
   self.points += 30
end
self.update(points: self.points)
       def rest_points (question)
                                                                                                                                                                                                            0
        if question.difficulty.to_i == 1
        self.points -= 10
elsif question.difficulty.to_i == 2
self.points -= 20
          self.points -= 30
         self.update(points: self.points)
     user = User.new(name: "romanino", email: "riquelmel0torero@gmail.com", password: "avostegusta?")
user.reppw = "avostegusta?"
     expect(user).to be_valid
   it "is invalid without a username" do
     user = User.new(email: "romi10@gmail.com", password: "boca123")
user.reppw = "boca123"
     expect(user).not_to be_valid
     user = User.new(name: "marten", password: "bocaaa")
user.reppw = "bocaaa"
     expect(user).not_to be_valid
   it "is invalid without a password" do
    user = User.new(name: "liomessi10", email: "soydeboca@gmail.com")
     expect(user).not_to be_valid
     existing_user = User.create(name: "pichonadvincula", email: "peru@example.com", password: "zambradios")
user = User.new(name: "cheftravel", email: "peru@example.com", password: "latercera")
     expect(user).not_to be_valid
     existing_user.destroy
     existing_user = User.create(name: "MarcosRED", email: "bocaelmasgrande@example.com", password: "matafuegos123")
user = User.new(name: "MarcosRED", email: "tefuistealab@mail.com", password: "password")
     expect(user).not_to be_valid
      existing_user.destroy
```

Posteriormente incluimos en el modelo de Users que el nombre del usuario y la contraseña tenga un rango de cuantos caracteres pueden tener, y que el email debe tener incluido un @, con un carácter antes y uno después.

Se agregó además que cuando un usuario ingrese una contraseña, esta debe coincidir con una ya existente.

Los test para cubrir los casos de prueba de las funciones de puntos aún no están implementados, teniendo en total un 47,83% de cobertura del modelo.





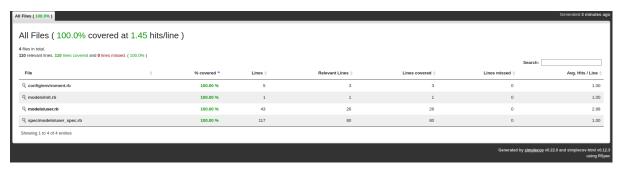
```
it "is invalid if pws not matches" do
    user = User.new(name: "MarcosRED", email: "tefuistealab@mail.com", password: "password", reppw: "paswor")
    expect(user).not_to be_valid
end

it "is valid if pws matches" do
    user = User.new(name: "marquitos", email: "tefuistealab@gmail.com", password: "paswor")
    user.reppw = "paswor"
    expect(user).to be_valid
end

it "is invalid with a email without @" do
    user = User.new(name: "koka", email: "mail.com", password: "password")
    expect(user).not_to be_valid
end

it "is invalid with a too short pw" do
    user = User.new(name: "changuito", email: "elchango7eballos@hotmail.com", password: "aaa")
    user.reppw = "aaa"
    expect(user).not_to be_valid
end
```

Luego ampliamos nuestros tests para abordar los casos relacionados con la puntuación del usuario, evaluando cómo se asignaban los puntos al usuario en función de la dificultad de las preguntas y si sus respuestas eran correctas o incorrectas. Esta ampliación nos permitió alcanzar una cobertura completa del 100% para nuestro modelo User.



```
class User < ActiveRecord::Base validates 'name, presence: true, uniqueness: true, length: { in: 3..20 }

validates 'name, presence: true, uniqueness: true, format: { with: /Af_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1=f_eMs_1
```

```
it "updates points tratament
it "updates points with a question of difficulty 1" do
    user = User.new(name: "colo", email: "colobarco@gmail.com", password: "aguantebocal23")
    user.reppw = "aguantebocal23"
    question = double("¿Cual es el manjar tipico de Argentina?", difficulty: 1)
    user.sum_points(question)
    cynect(user_points) to em(10)
    expect(user.points).to eq(10)
it "updates points with a question of difficulty 2" do
   user = User.new(name: "lucasjanson", email: "jansonson@mail.com", password: "bocaaaaa12")
   user.reppw = "bocaaaaa12"
    question = double("¿Qué tipo de madera es tradicionalmente utilizada para el fuego del asado?", difficulty: 2)
    user.points = 100
    user.sum_points(question)
    expect(user.points).to eq(120)
it "updates points with a question of difficulty 3" do
  user = User.new(name: "Carlos", email: "apache@mail.com", password: "bocaaaaa12")
  user.reppw = "bocaaaaa12"
    question = double("¿Qué rol cumple la vainilla en su preparación?", difficulty: 3)
    user.sum_points(question)
   expect(user.points).to eq(30)
it "updates points negatively with a question of difficulty 1" do
   user = User.new(name: "hamburguesitafabra", email: "ffabra@example.com", password: "liberenAlPanitaVilla123")
   user.reppw = "liberenAlPanitaVilla123"
   question = double("¿Cuáles son los ingredientes para preparar dulce de leche?", difficulty: 1)
   user.rest_points(question)
    expect(user.points).to eq(-10)
it "updates points negatively with a question of difficulty 2" do

user = User.new(name: "javi60Dcia", email: "javmancol@mail.com", password: "aBreyLeFaltaTodavia123")

user.reppw = "aBreyLeFaltaTodavia123"

question = double("¿Cuál es el proceso principal para lograr un buen asado argentino?", difficulty: 2)

user.rest_points(question)

expect(user_noints) to apd_20)
    expect(user.points).to eq(-20)
it "updates points negatively with a question of difficulty 3" do
    user = User.new(name: "Roncaglia", email: "troncaglia2@mail.com", password: "oPasaLaPelotaoPasaElJugador")
    user.reppw = "oPasaLaPelotaoPasaElJugador"
    user.points = 100
   question = double("¿Por qué el Dulce de Leche se debe revolver constantemente durante su preparación?", difficulty: 3) user.rest_points(question)
    expect(user.points).to eq(100-30)
end
```

## **Test para Model Question**

En nuestro modelo de Question, una pregunta debe contener texto, un nivel de dificultad y una respuesta correcta asociada; para ello realizamos una serie de casos de pruebas en los que logramos obtener un 100% de cobertura sobre el modelo.

```
models/question.rb

100.0% lines covered

7 relevant lines. 7 lines covered and 0 lines missed.

1. require 'active_record'
2. require_relative 'answer'
3.
4. class Question < ActiveRecord:Base
5. has_many :answers
6.
7. validates :text, presence: true
validates :levels_id, presence: true, numericality: { only_integer: true, greater_than: 0 }
validates :answer_id, presence: true, numericality: { only_integer: true, greater_than: 0 } # has a correct associated answer

10.
11.
12. end

models/question.rb
```

```
defired > pac > models > d question_sec.b

    require 'sinatra/activerecord'

    require_relative '../../models/init.rb'

    describe Question do

    it 'is a valid question' do
        question = Question.new(text: "Estas leyendo?", levels_id: 1, answer_id: 2)
        expect(question).to be_valid
    end

it 'is invalid with no text' do
    question = Question.new(levels_id: 1, answer_id: 2)
    expect(question).not_to be_valid
end

it 'is invalid with no correct answer associated' do
    question = Question.new(text: "Que te parece?", levels_id: 1)
expect(question).on_ew(text: "Que te parece?", levels_id: 1)
end

end

it 'is invalid with negative id of answer associated' do
    question = Question.new(text: "Seguis leyendo?", levels_id: 1, answer_id: -2)
    expect(question).not_to be_valid
end

end

end

it 'is invalid with negative id of answer associated' do
    question = Question.new(text: "Seguis leyendo?", levels_id: 1, answer_id: -2)
    expect(question).not_to be_valid
end

end
```

#### Conclusión

En términos generales, nuestra metodología se basó en la aplicación de pruebas unitarias utilizando el framework RSpec en Ruby y la herramienta simpleCov para medir la cobertura de código. El objetivo era asegurarnos de que cada componente del código se evaluará de manera independiente para confirmar su correcto funcionamiento.

Nos centramos en la aplicación de pruebas unitarias y la mejora continua de la cobertura de código para garantizar la calidad y confiabilidad de nuestra aplicación, garantizando que todas las funcionalidades estén correctamente probadas y funcionando según lo previsto.