

**Universidad Nacional de Río Cuarto**

**Facultad de Cs. Exactas Físico Químicas y Naturales**

**Departamento de Computación**

**Ingeniería de Software (Cód. 3304) - 2023**

**Taller IS 2023 - Actividad Nro: 1**

**Integrantes:** Darío Ferreyra y Kevin Riberi

## Antes de comenzar - Retomando el trabajo de AYDS

En ese momento teníamos planteado en nuestros modelos la validación de:

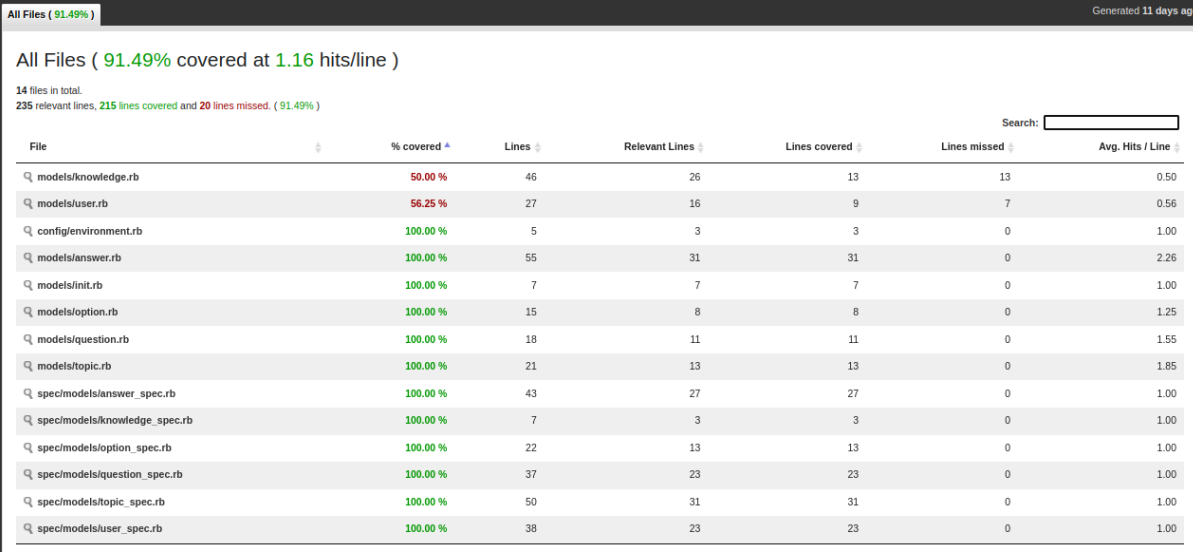
- atributos mediante el método 'validates' (proporcionado por ActiveRecord)
- atributos que eran clave foránea chequeando que dicho atributos estén presente efectivamente en la base de datos (para evitar errores de integridad referencial)
- reglas relacionadas con el negocio. Por ejemplo:

-que en cada instancia de 'Answer' la opción se corresponda a la pregunta del registro.

-que en cada instancia de 'Answer' el nivel de la pregunta que responda el usuario se corresponda con el nivel que tiene en dicho tema.

Además, en algunos de los modelos teníamos definidos métodos que utilizamos para manejar la lógica de la aplicación.

La ejecución con SimpleCov (Figura 1) nos dio lo siguiente:



All Files ( 91.49% covered at 1.16 hits/line )

14 files in total.  
235 relevant lines, 215 lines covered and 20 lines missed. ( 91.49% )

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
models/knowledge.rb	50.00 %	46	26	13	13	0.50
models/user.rb	56.25 %	27	16	9	7	0.56
config/environment.rb	100.00 %	5	3	3	0	1.00
models/answer.rb	100.00 %	55	31	31	0	2.26
models/init.rb	100.00 %	7	7	7	0	1.00
models/option.rb	100.00 %	15	8	8	0	1.25
models/question.rb	100.00 %	18	11	11	0	1.55
models/topic.rb	100.00 %	21	13	13	0	1.85
spec/models/answer_spec.rb	100.00 %	43	27	27	0	1.00
spec/models/knowledge_spec.rb	100.00 %	7	3	3	0	1.00
spec/models/option_spec.rb	100.00 %	22	13	13	0	1.00
spec/models/question_spec.rb	100.00 %	37	23	23	0	1.00
spec/models/topic_spec.rb	100.00 %	50	31	31	0	1.00
spec/models/user_spec.rb	100.00 %	38	23	23	0	1.00

Showing 1 to 14 of 14 entries

**Figura 1** Análisis de cobertura con SimpleCov del código que nos quedó al finalizar el primer cuatrimestre.

Es decir, en ese momento teníamos cubierto en su totalidad alguno de los modelos pero nos faltaba cubrir aproximadamente la mitad del código con los tests para la clase 'Knowledge' y la clase 'User'.

De hecho, para 'Knowledge' no habíamos llegado a definir ningún test (Figura 2).

AYDS-Taller / AppWeb / spec / models / knowledge\_spec.rb 

```
1  require 'sinatra/activerecord'
2  require_relative '../models/init.rb'
3
4  describe Knowledge do
5
6
7  end
```

Figura 2. Definición de los tests para la 'Knowledge' inicialmente sin ninguna prueba.

Para la clase 'User' teníamos definidos métodos que no testeamos en nuestra suite de tests (Figura 3 y Figura 4).

```
11. def update_points (correct, level)
12.   if correct
13.     self.points += 10 * level
14.   else
15.     self.points -= 4 * level
16.   end
17. end
18.
19. def initialize_knowledges
20.   topics = Topic.all
21.
22.   topics.each do |topic|
23.     Knowledge.create(user_id: self.id, topic_id: topic.id, level: 1, correct_answers_count: 0)
24.   end
25.   self.points = 0
26. end
```

Figura 3. En el análisis de SimpleCov nos muestra que para los métodos 'update\_points' y 'initialize\_knowledges' de la clase 'User' no están siendo testeados en lo absoluto.

```
5
6 describe User do
7   it "is valid with a username an email and a password" do
8     user = User.new(username: "Martincito28", email: "martinp@example.com", password_digest: "password")
9     expect(user).to be_valid
10  end
11
12  it "is invalid without a username" do
13    user = User.new(email: "roman10@example.com", password_digest: "password")
14    expect(user).not_to be_valid
15  end
16
17  it "is invalid without an email address" do
18    user = User.new(username: "RR10", password_digest: "password")
19    expect(user).not_to be_valid
20  end
21
22  it "is invalid without an password" do
23    user = User.new(username: "RR10", email: "roman10@gmail.com")
24    expect(user).not_to be_valid
25  end
26
27  it "is invalid with a duplicate email address" do
28    existing_user = User.create(username: "Messi10", email: "messi@example.com", password_digest: "password")
29    user = User.new(username: "argento", email: "messi@example.com", password_digest: "password")
30    expect(user).not_to be_valid
31  end
32
33  it "is invalid with a duplicate username" do
34    existing_user = User.create(username: "Marcos10", email: "marcos@example.com", password_digest: "password")
35    user = User.new(username: "Marcos10", email: "otromail@example.com", password_digest: "password")
36    expect(user).not_to be_valid
37  end
38 end
```

Figura 4. Pruebas escritas para la clase 'User'. En ningún momento se testean los métodos 'update\_points' y 'initialize\_user'.

Es decir, lo que debíamos hacer es agregar los tests para los métodos que faltaban en la clase 'User' y escribir los tests para la clase 'Knowledge' en su totalidad.

Por otra parte, teníamos métodos definidos en nuestra aplicación que se correspondían a 'User' pero que no estaban definidos dentro del modelo (Figura 5). También se debían pasar dichos métodos dentro del modelo y agregar sus respectivos tests.

```

58
59   # definicion de metodos auxiliares
60   |
61   def username_taken?(username)
62     User.exists?(username: username)
63   end
64
65   def email_taken?(email)
66     User.exists?(email: email)
67   end
68
69   def passwords_match?(password, confirm_password)
70     !password.empty? && !confirm_password.empty? && password == confirm_password
71   end
72
73   ✓ def create_user(username, email, password)
74     user = User.new(username: username, email: email, password: password)
75     user.save
76     user.initialize_knowledges
77   end

```

**Figura 5.** Métodos definidos que se corresponden al modelo 'User' se encuentran definidos en el archivo 'user\_controller.rb'. De esta forma, los métodos quedan aislados para reutilizarlos en otra parte de la aplicación.

## Corrección Parte 1 - Modelo 'User'

Se escribieron dentro del modelo 'User' los métodos correspondientes que estaban definidos en otra parte. Además, se escribieron los tests correspondientes a dichos métodos (Figura 6 y 7).

AYDS-Taller / AppWeb / models / user.rb 

```
28     def self.username_taken?(username)
29       User.exists?(username: username)
30     end
31
32     def self.email_taken?(email)
33       User.exists?(email: email)
34     end
35
36     def self.passwords_match?(password, confirm_password)
37       !password.empty? && !confirm_password.empty? && password == confirm_password
38     end
```

Figura 6. Definición de los métodos faltantes dentro de la clase 'User'.

AYDS-Taller / AppWeb / spec / models / user\_spec.rb

```
56     describe "email_taken?" do
57       it 'returns true if email is taken' do
58         existing_user = User.create(username: "Messi10", email: "messi@example.com", password_digest: "password")
59         result = User.email_taken?('messi@example.com')
60         expect(result).to be true
61         existing_user.destroy
62       end
63
64       it 'returns false if username is not taken' do
65         result = User.username_taken?('non_existing_email')
66         expect(result).to be false
67       end
68     end
69
70     describe 'passwords_match?' do
71       it 'returns true if passwords match' do
72         result = User.passwords_match?('password123', 'password123')
73         expect(result).to be true
74       end
75     end
```

Figura 7. Definición de las pruebas para los métodos recientemente agregados.

Después de todas estas correcciones, el análisis realizado con SimpleCov (Figura 8) nos arrojó el siguiente resultado:

All Files ( 96.27% )

Generated 11 days ago

All Files ( 96.27% covered at 1.78 hits/line )

14 files in total.  
295 relevant lines, 284 lines covered and 11 lines missed. ( 96.27% )

Search:

File	% covered <sup>▲</sup>	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
models/knowledge.rb	57.69 %	46	26	15	11	2.50
config/environment.rb	100.00 %	5	3	3	0	1.00
models/answer.rb	100.00 %	55	31	31	0	2.26
models/init.rb	100.00 %	7	7	7	0	1.00
models/option.rb	100.00 %	15	8	8	0	1.25
models/question.rb	100.00 %	18	11	11	0	1.55
models/topic.rb	100.00 %	21	13	13	0	1.85
models/user.rb	100.00 %	40	22	22	0	2.45
spec/models/answer_spec.rb	100.00 %	44	27	27	0	1.00
spec/models/knowledge_spec.rb	100.00 %	7	3	3	0	1.00
spec/models/option_spec.rb	100.00 %	22	13	13	0	1.00
spec/models/question_spec.rb	100.00 %	37	23	23	0	1.00
spec/models/topic_spec.rb	100.00 %	50	31	31	0	1.00
spec/models/user_spec.rb	100.00 %	134	77	77	0	2.30

Showing 1 to 14 of 14 entries

**Figura 8** Análisis de cobertura con SimpleCov del código después de realizar las correcciones en el modelo 'User'.

Es decir, logramos tener un 100% de cobertura con los tests para nuestra clase 'User'.

## Correcciones Parte 2 - Modelo 'Knowledge'

Para la clase 'Knowledge' se escribieron todos los tests para las definiciones que estaban dentro del modelo (Figura 9).

AYDS-Taller / AppWeb / spec / models / knowledge\_spec.rb 

```
1  require 'sinatra/activerecord'
2  require_relative '../models/init.rb'
3
4  describe Knowledge do
5    context "validations" do
6      it "is valid with valid attributes" do
7        user = User.create(username: "cargo0troejemplo", email: "yoyfdsdsao@example.com", password: "password")
8        topic = Topic.create(name: "Biologiadfssda", amount_questions_L1: 3, amount_questions_L2: 5, amount_questions_L3: 2)
9        knowledge = Knowledge.new(user: user, topic: topic, level: 1)
10       expect(knowledge).to be_valid
11       user.destroy
12       topic.destroy
13     end
14
15     it "is not valid without a user" do
16       topic = Topic.create(name: "Science")
17       knowledge = Knowledge.new(topic: topic, level: 2)
18       expect(knowledge).not_to be_valid
19       topic.destroy
20     end
21
22     it "is not valid without a topic" do
23       user = User.create(username: "Alice", email: "alice@example.com", password: "password")
24       knowledge = Knowledge.new(user: user, level: 3)
25       expect(knowledge).not_to be_valid
26       user.destroy
27     end
28   end
29 end
```

Figura 9. Parte de la definición de las pruebas para el modelo 'Knowledge'.

Con estos tests agregados, el análisis con SimpleCov (Figura 10) nos dio el siguiente resultado:



All Files ( 100.0% )

Generated about 15 hours ago

All Files ( 100.0% covered at 1.35 hits/line )

14 files in total.  
404 relevant lines, 404 lines covered and 0 lines missed. ( 100.0% )

Search:

File	% covered <sup>▲</sup>	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
config/environment.rb	100.00 %	5	3	3	0	1.00
models/answer.rb	100.00 %	55	31	31	0	2.71
models/init.rb	100.00 %	7	7	7	0	1.00
models/knowledge.rb	100.00 %	48	28	28	0	2.14
models/option.rb	100.00 %	15	8	8	0	1.75
models/question.rb	100.00 %	18	11	11	0	2.64
models/topic.rb	100.00 %	21	13	13	0	3.00
models/user.rb	100.00 %	40	22	22	0	1.32
spec/models/answer_spec.rb	100.00 %	68	48	48	0	1.00
spec/models/knowledge_spec.rb	100.00 %	87	61	61	0	1.00
spec/models/option_spec.rb	100.00 %	32	23	23	0	1.00
spec/models/question_spec.rb	100.00 %	46	30	30	0	1.00
spec/models/topic_spec.rb	100.00 %	51	32	32	0	1.00
spec/models/user_spec.rb	100.00 %	147	87	87	0	1.00

Showing 1 to 14 of 14 entries

**Figura 10.** Análisis de cobertura con SimpleCov del código después de definir las pruebas en el modelo 'Knowledge'.

Es decir, conseguimos un 100% de cobertura en cuanto a todos los modelos de nuestra aplicación.

## Conclusiones

Basándonos en el trabajo realizado, podemos llegar a las siguientes conclusiones:

- SimpleCov es una herramienta muy útil para determinar el porcentaje de cobertura de nuestras pruebas y para identificar las líneas de código que aún no han sido probadas.
- Con la ayuda de SimpleCov, logramos alcanzar una cobertura del 100% en nuestras pruebas de modelos. Sin embargo, es importante destacar que en sistemas menos complejos, lograr una cobertura completa puede requerir definir numerosas pruebas.
- En la práctica, es esencial encontrar un equilibrio entre la cantidad de código que deseamos cubrir con pruebas y el esfuerzo necesario para lograrlo.