

Ingeniería de Software (3304)

Departamento de Ciencias de la Computación

Facultad de Ciencias Exactas, Físico-Químicas y Naturales

Universidad Nacional de Río Cuarto



Taller IS - Actividad N°1

Año: 2023. Darío Ferreyra y Kevin Riberi

Antes de comenzar, ¿hasta dónde habíamos llegado en 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.

Antes de comenzar, ¿hasta dónde habíamos llegado en 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.

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

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

- atributos mediante el método 'validates' (proporcionado por ActiveRecord)

```
models >  answer.rb
1  class Answer < ActiveRecord::Base
2
3      belongs_to :user
4      belongs_to :question
5      belongs_to :option
6
7
8      validates :user_id, presence: true
9      validates :question_id, presence: true
10     validates :option_id, presence: true
11
```

Antes de comenzar, ¿hasta dónde habíamos llegado en 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.

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

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

- atributos que eran clave foránea chequeando que dicho atributos estén presente efectivamente en la base de datos (para evitar errores de integridad referencial)

```
models >  answer.rb
1 class Answer < ActiveRecord::Base
2
3   belongs_to :user
4   belongs_to :question
5   belongs_to :option
6
```

```
11
12   validate :valid_user_id
13   validate :valid_question_id
14   validate :valid_option_id
15
```

```
20
21   def valid_user_id
22     return if User.exists?(user_id)
23
24     errors.add(:user_id, "does not exist")
25   end
26
27   def valid_question_id
28     return if Question.exists?(question_id)
29
30     errors.add(:question_id, "does not exist")
31   end
32
33   def valid_option_id
34     return if Option.exists?(option_id)
35
36     errors.add(:option_id, "does not exist")
37   end
38
```

Antes de comenzar, ¿hasta dónde habíamos llegado en 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.

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

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

- reglas relacionadas con el negocio.

```
models >  answer.rb
```

```
1 class Answer < ActiveRecord::Base
2
3   belongs_to :user
4   belongs_to :question
5   belongs_to :option
6
```

```
15
16   validate :valid_option_respect_question
17   validate :matching_user_level_in_topic
18
```

```
def valid_option_respect_question
  if option.present? && option.question_id != question_id
    errors.add(:option_id, "does not belong to the corresponding question")
  end
end

def matching_user_level_in_topic
  if question.present?
    topic = question.topic_id
    knowledge = Knowledge.find_by(user_id: user_id, topic_id: topic)

    if knowledge.nil? || question.level != knowledge.level
      errors.add(:question, "must have the same level as the user's level in the topic")
    end
  end
end
```


Antes de comenzar, ¿hasta dónde habíamos llegado en 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.

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

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

```
models > user.rb
1  class User < ActiveRecord::Base
2    has_secure_password
3
4    has_many :answers
5    has_many :knowledges
6    has_many :topics, :through => :knowledges
7
```

```
10
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
27
```

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

El análisis por SimpleCov nos dio el siguiente resultado:

All Files (91.49%) Generated 11 days ago

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%)

Search:

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

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

El análisis por SimpleCov nos dio el siguiente resultado:

All Files (91.49%)

Generated 11 days ago

Por mejorar, modelos de 'Knowledge' y 'User'

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%)

Search:

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

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

Para la clase 'Knowledge' no teníamos definidos ningún test:

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
```

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

Para la clase 'User' teníamos métodos que no estaban siendo testeados:

```
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
```

Antes de comenzar, ¿hasta dónde habíamos llegado en AYDS?

Además, teníamos métodos que se correspondían a 'User' por fuera del modelo:

AYDS-Taller / AppWeb / [controllers](#) / user_controller.rb

```
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
```

Corrección Parte 1 - Modelo 'User'

Agregamos dentro del modelo 'User' los métodos correspondientes:

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
```


Corrección Parte 1 - Modelo 'User'

Se escribieron los tests para los métodos recientemente agregados:

[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
```

Corrección Parte 1 - Modelo 'User'

Después de estas correcciones, el análisis por SimpleCov nos dio:

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 ▲	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

Corrección Parte 1 - Modelo 'User'

Después de estas correcciones, el análisis por SimpleCov nos dio:

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 ▲	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

Corrección Parte 2 - Modelo 'Knowledge'

Se escribieron los tests para las validaciones y métodos de 'Knowledge':

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: "cargoOtroejemplo", 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
```

Corrección Parte 2 - Modelo 'Knowledge'

Después de agregar los tests, se realizó el análisis con SimpleCov:

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 [▲]	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

Corrección Parte 2 - Modelo 'Knowledge'

Después de agregar los tests, se realizó el análisis con SimpleCov:

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 [▲]	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

Conclusiones

En base a todo el trabajo realizado podemos concluir lo siguiente:

- SimpleCov es una herramienta muy útil para identificar cuánto porcentaje de nuestro código estamos testeando. Y conocer qué líneas nos faltan testear.
- Mediante la ayuda de SimpleCov logramos tener un 100% de cobertura en las pruebas de nuestros modelos.
- Ya se encuentra que para sistemas no tan complejos si uno quiere realizar un test completo, hay que definir muchas pruebas.
- En la práctica uno debe elegir una relación de compromiso entre: cuánto porcentaje de mi sistema quiero testear vs el esfuerzo que esto conlleva.

¿Próximo paso?

```
spec > app_spec.rb
1  # spec/app_spec.rb
2  require 'rack/test'
3  require_relative '../server.rb'
4
5  RSpec.describe 'Sinatra App' do
6    include Rack::Test::Methods
7
8    def app
9      # incluir el nombre de la clase correspondiente a la Application definida en el server.rb
10     App
11   end
12
13   it 'probando rutas del server' do
14     get '/' # Accede a la ruta '/'
15     expect(last_response.status).to eq(200) # Verifica el código de respuesta HTTP
16   end
17 end
```