Análisis del proceso de Testing

Ing. Software

Machuca Franco
Natali Valentino

Introducción

En este informe vamos a analizar la etapa de cobertura del código durante el proceso de testing de nuestro proyecto de software web. El objetivo de esta iteración es cumplir con el 100% de testeo de la aplicación, para asegurar la calidad del software y no dejar ninguna parte sin cubrir.

Para llegar al objetivo deseado se utilizó una gema recomendada por el equipo docente, llamada **SimpleCov**, que funciona para mostrar la cobertura de líneas de código al momento de ejecución de la unidad de testeo. El testing a su vez fue realizado con la gema **Rspec**. A continuación mostraremos el resultado de los test con la gema SimpleCov, seguido de una descripción detallada, organizada por cada archivo y método/endpoint, de las líneas que no fueron cubiertas por los tests. Estas líneas estarán resaltadas en color rojo.

Resultados Generales

All Files (77.88% covered at 1.42 hits/line)

15 files in total.
330 relevant lines, **257** lines covered and **73** lines missed. (77.88%)

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
controllers/questions_manager.rb	40.00 %	49	25	10	15	0.40
controllers/game_data_manager.rb	50.98 %	91	51	26	25	1.75
myapp.rb	71.30 %	176	115	82	33	1.65
models/answer.rb	100.00 %	4	3	3	0	1.00
models/exam.rb	100.00 %	4	3	3	0	1.00
models/lesson.rb	100.00 %	4	3	3	0	1.00
models/level.rb	100.00 %	14	10	10	0	1.10
models/mouse_translation.rb	100.00 %	6	3	3	0	1.00

models/multiple_choice.rb	100.00 %	4	3	3	0	1.00
models/question.rb	100.00 %	15	10	10	0	1.30
models/to_complete.rb	100.00 %	4	3	3	0	1.00
models/translation.rb	100.00 %	4	3	3	0	1.00
models/user.rb	100.00 %	5	4	4	0	1.00
models/user_level.rb	100.00 %	5	4	4	0	1.00
spec/test_spec.rb	100.00 %	140	90	90	0	1.42

Métodos/Endpoints del código sin cubrir

Questions_manager

Show

```
1.  def show(question: Question)
2.  1
3.    case question.questionable_type
4.    when "Multiple_choice"
5.      return :multiple_choice
6.    when "Translation"
7.  1
8.      return :translation
9.    when 'To_complete'
10.     return :to_complete
11.   when 'MouseTranslation'
12.     return :mouse_translation
13.   end
14. end
15.
16. # Método que analiza una respuesta a una pregunta y devuelve un
    booleano indicando si es correcta o no.
```

CorrectAnswer?

```
17. def correctAnswer?(answer: Answer, question: Question)
18.   if question.questionable_type == 'Multiple_choice'
19.     return answer.correct
20.   elsif question.questionable_type == 'Translation' ||
    question.questionable_type == 'To_complete' ||
    question.questionable_type == 'MouseTranslation'
21.     user_guess = answer.answer.gsub(/\s+/, "").downcase
```

```
      correct_phrase = question.answers.find_by(correct: true).answer.gsub(/\s+/, "").downcase
      return user_guess == correct_phrase
    else
      puts "No se reconoce el tipo"
    end
  end

  # Dada una pregunta (question), devuelve la siguiente a ella (que pertenece al mismo nivel). Devuelve nil si question es la última de su nivel.
```

NextQuestion

```
  def nextQuestion(question: Question)
    question_family = question.level.questions
    actual_question_index = question_family.find_index(question)
    next_question = question_family[actual_question_index + 1]
    return next_question
  end

  def buildUserAnswer(answer: String, question: Question)
    return Answer.new(answer: answer, correct: false, question: question)
  end
```

Game_data_manager

GetAmountOfLevelsCompleted

```
def getAmountOfLevelsCompleted(user: User)
        u = UserLevel.where(user: user)
        sum = 0
        u.each do |row|
            if row.level.playable_type == "Exam"
                exam = row.level.exam
                if exam.minScore <= row.userLevelScore
                    sum += 1
                end
            else
                if row.userLevelScore > 0
                    sum += 1
                end
            end
        end
        return sum
    end
```

CompletedLevel?

```
19.
20.     def completedLevel?(level: Level, user: User)
21. 2
22.         row = UserLevel.find_by(user: user, level: level)
23. 2
24.         if row
25. 2
26.             if row.level.playable_type == "Exam"
27.                 return row.userLevelScore >= row.level.exam.minScore
28.             else
29. 2
30.                 return row.userLevelScore > 0
31.             end
32.         else
33.             return false
34.         end
35.     end
36.
```

UnlockNextLevelFor

```
37.
38.     def unlockNextLevelFor(user: User, possiblyCompleted: Level)
39.         if completedLevel?(level: possiblyCompleted, user: user)
40.             nextLevel = Level.where("id > ?",
    possiblyCompleted.id).first
41.             if nextLevel
42.                 UserLevel.create(user: user, level: nextLevel,
    userLevelScore: 0)
43.                 return true
44.             end
45.         end
46.         return false
47.     end
48.
```

AddUserLevelScore

```
49.
50.     def addUserLevelScore(user: User, level: Level, value: int)
51.         row = UserLevel.find_by(user: user, level: level)
52.         if row
53.             row.update(userLevelScore: value)
54.         end
55.     end
56.
```

ResetUserLevelScore

```
57.
```

```
58.     def resetUserLevelScore(user: User, level: Level)
59.         addUserLevelScore(user: user, level: level, value: 0)
60.     end
61.
```

GetLevelScore

```
62.
63.     def getLevelScore(user: User, level: Level)
64.         row = UserLevel.find_by(user: user, level: level)
65.         if row
66.             return row.userLevelScore
67.         end
68.     end
```

Myapp

Login

```
1.  post '/login' do
2.  4
3.          user = User.find_by(name: params[:name], password:
    params[:password])
4.  4
5.          if user
6.  4
7.              session[:user_id] = user.id
                                    8.  4
9.              redirect '/jugar'
10.         else
11.             @error_message = "Nombre de usuario o contraseña son
    incorrectas"
12.             erb :login
13.         end
14.     end
```

Registro

```
1.  post '/registro' do
2.  2
3.          user = User.find_by(mail: params[:mail]) || User.find_by(name:
    params[:name])
4.  2
5.          if user
6.  1
7.              @error_message = "Usted ya tenía una cuenta previa"
8.  1
9.              erb :login
```

```
        else
1
            user = User.new(name: params[:name], mail: params[:mail], password: params[:password])
1
            if user.save
1
                @gm.createGameDataFor(user: user)
1
                session[:user_id] = user.id
1
                redirect '/login'
            else
                @error_message = "Error al crear cuenta"
                erb :registro
            end
        end
    end
```

Perfil

```
    get '/perfil' do
1
        if session[:user_id]
            @user = User.find(session[:user_id])
            @total_score = @gm.getTotalScoreOf(user: @user)
            @levels_completed = @gm.getAmountOfLevelsCompleted(user: @user)
            @rank = @gm.getUserRank(user: @user)
            erb :profile
        else
1
            redirect '/login'
        end
    end
```

Comienzo de un nivel

```
1
    get '/level/:level_id' do
2
        if session[:user_id]
1
            session[:user_level_score] = 0
1
            @user = User.find(session[:user_id])
1
            @level = Level.find(params[:level_id])
1
            if @gm.completedLevel?(user: @user, level: @level)
                @gm.resetUserLevelScore(user: @user, level: @level)
```

```
            end
1
            @questions = Question.where(level_id: params[:level_id])
1
            redirect '/level/' + params[:level_id].to_s + '/' +
@questions.first.id.to_s
        else
1
            redirect '/login'
        end
    end
```

Visualización de una pregunta dentro de un nivel

```
    get '/level/:level_id/:question_id' do
2
        if session[:user_id]
1
            @level = Level.find(params[:level_id])
1
            @question = Question.find(params[:question_id])
1
            @answers = Answer.where(question_id: @question.id)
1
            if @level && @question && @answers
1
               erb @qm.show(question: @question)
            else
                redirect '/jugar'
            end
        else
1
            redirect '/login'
        end
    end
```

Verificación de la respuesta del usuario a una pregunta

```
    post '/level/:level_id/:question_id/check' do
1
        if session[:user_id]
            @question = Question.find_by(id: params[:question_id])
            @level = Level.find_by(id: params[:level_id])
            @user = User.find_by(id: session[:user_id])
            if @question && @level

                if @question.questionable_type == "Translation" ||
@question.questionable_type == "To_complete" ||
                   @question.questionable_type == "MouseTranslation"
```

```
11.                     @user_answer = @qm.buildUserAnswer(answer:
    params[:user_guess], question: @question)
12.                 else
13.                     @user_answer = Answer.find_by(id:
    params[:answer_id])
14.                 end
15.                 if @qm.correctAnswer?(answer: @user_answer, question:
    @question)
16.                     session[:user_level_score] += 100
17.                 end
18.
19.                 @next_question = @qm.nextQuestion(question: @question)
20.                 if @next_question
21.                     redirect "/level/#{params[:level_id]}/" +
    @next_question.id.to_s
22.                 else
23.                     @user = User.find_by(id: session[:user_id])
24.                     @gm.addUserLevelScore(user: @user, level: @level,
    value: session[:user_level_score])
25.                     @final_score = session[:user_level_score]
26.                     session[:user_level_score] = 0
27.                     if @gm.completedLevel?(level: @level, user: @user)
28.                         @show_success_popup = true
29.                     else
30.                         @show_failure_popup = true
31.                     end
32.                     @gm.unlockNextLevelFor(user: @user,
    possiblyCompleted: @level)
33.                     erb @qm.show(question: @question)
34.                 end
35.             else
36.                 redirect "/jugar"
37.             end
38.         else
39. 1
40.             redirect "/login"
41.         end
42.     end
```

Como podemos observar, el 77.88% de las líneas fueron cubiertas por los tests realizados, pero aún quedan algunas partes de archivos concretos por testear. Por lo tanto, pasaremos a listar las modificaciones llevadas a cabo por el equipo.

Modificaciones

Questions_manager

```
context "questionsManager" do

    let(:qm) { QuestionsManager.new }

    let(:level) { Level.create}

    let(:question1) { Question.new(questionable_type: "Multiple_choice",
level: level) }

    let(:question2) { Question.create(description: "prueba1",
questionable_type: "Translation", level: level) }

    let(:question3) { Question.create(description: "prueba2",
questionable_type: "To_complete", level: level) }

    let(:question4) { Question.create(description: "prueba3",
questionable_type: "MouseTranslation", level: level) }


    before do

      level.questions.push(question2)

      level.questions.push(question3)

      level.questions.push(question4)

    end
```

Lo primero que realizamos para el testeo de este archivo fue setear los diferentes objetos que utilizamos en los tests de unidad. Empleamos la palabra clave "**let**" para crear variables que luego se eliminarán al finalizar la ejecución del test. Creamos una instancia de QuestionsManager para poder acceder a las funciones del módulo; las cuatro tipo de preguntas existentes en nuestro software y el nivel al que todas las preguntas pertenecen.

Show

Este método es utilizado para asignar el correspondiente view dependiendo el tipo de pregunta que se obtiene como parámetro. Realizamos los siguientes tests:

```
describe "#show" do

      it "shows multiple_choice view" do

        expect(qm.show(question: question1)).to eq(:multiple_choice)

      end


      it "shows translation view" do

        expect(qm.show(question: question2)).to eq(:translation)

      end


      it "shows to_complete view" do

        expect(qm.show(question: question3)).to eq(:to_complete)

      end


      it "shows mouse_translation view" do

        expect(qm.show(question: question4)).to eq(:mouse_translation)

      end

    end
```

El test simplemente confirma que se muestre el view correspondiente al tipo de pregunta.

CorrectAnswer?

El método “correctAnswer?” es utilizado para determinar si una respuesta a una pregunta es o no correcta.

```
describe "#correctAnswer?" do


      it "multiple_choice return true if answer is true" do

        correct_answer = Answer.new(answer: "respuesta", correct: true,
question: question1)

        expect(qm.correctAnswer?(answer: correct_answer, question:
question1)).to be true

      end


      it "multiple_choice return false if answer is false" do

        incorrect_answer = Answer.new(answer: "respuesta", correct:
false, question: question1)

        expect(qm.correctAnswer?(answer: incorrect_answer, question:
question1)).to be false

      end


      it "translation return true if user_answer equals translation
expected answer" do

        correct_answer = Answer.create(answer: "respuesta", correct:
true, question: question2)

        expect(qm.correctAnswer?(answer: correct_answer, question:
question2)).to be true

        correct_answer.destroy

      end


      it "translation return false if user_answer not equals translation
expected answer" do
```

```
        correct_answer = Answer.create(answer: "respuesta", correct:
true, question: question2)

        incorrect_answer = Answer.new(answer: "otra respuesta", correct:
true, question: question2)

        expect(qm.correctAnswer?(answer: incorrect_answer, question:
question2)).to be false

        correct_answer.destroy

      end


      it "to_complete return true if user_answer equals to_complete
expected answer" do

        correct_answer = Answer.create(answer: "respuesta", correct:
true, question: question3)

        expect(qm.correctAnswer?(answer: correct_answer, question:
question3)).to be true

        correct_answer.destroy

      end


      it "to_complete return false if user_answer not equals to_complete
expected answer" do

        correct_answer = Answer.create(answer: "respuesta", correct:
true, question: question3)

        incorrect_answer = Answer.new(answer: "otra respuesta", correct:
true, question: question3)

        expect(qm.correctAnswer?(answer: incorrect_answer, question:
question3)).to be false

        correct_answer.destroy

      end


      it "mouse_translation return true if user_answer equals
mouse_translation expected answer" do
```

```
        correct_answer = Answer.create(answer: "respuesta", correct:
true, question: question4)

        expect(qm.correctAnswer?(answer: correct_answer, question:
question4)).to be true

        correct_answer.destroy

      end


      it "mouse_translation return false if user_answer not equals
mouse_translation expected answer" do

        correct_answer = Answer.create(answer: "respuesta", correct:
true, question: question4)

        incorrect_answer = Answer.new(answer: "otra respuesta", correct:
true, question: question4)

        expect(qm.correctAnswer?(answer: incorrect_answer, question:
question4)).to be false

        correct_answer.destroy

      end


      it "Other questionable_type" do

        question = Question.create(questionable_type: "Hola", level:
level)

        correct_answer = Answer.new(answer: "respuesta", correct: true,
question: question)

        expect(qm.correctAnswer?(answer: correct_answer, question:
question)).to be nil

        correct_answer.destroy

        question.destroy

      end

    end
```

Cada instancia de test bajo este contexto prueba el resultado esperado para respuestas correctas e incorrectas. En cada unidad se crea una respuesta esperada y se le asigna a

la pregunta actual, para luego realizar lo siguiente: En el caso de que una respuesta sea correcta, solo pasamos la respuesta vinculada a la pregunta como parámetro del método; en caso de que queramos una respuesta incorrecta, simplemente pasamos cualquier otro tipo de respuesta diferente a la vinculada con la pregunta. Se testea cada tipo de pregunta por separado y también se considera el caso de que el tipo de pregunta sea inexistente.

nextQuestion

Como lo dice su nombre, esta función devuelve la siguiente pregunta.

```
describe "#nextQuestion" do

      it "return next question of a question" do

        expect(qm.nextQuestion(question: question2)).to be question3

      end

    end
```

Lo único que se prueba aquí es que, efectivamente, se obtiene la siguiente pregunta.

buildUserAnswer

Este método es auxiliar y se utiliza para crear una respuesta escrita por el usuario (También es utilizada en el tipo de pregunta donde el mouse es el atractivo principal).

```
describe '#buildUserAnswer' do

      it 'crea una nueva respuesta de usuario' do

        user_answer = qm.buildUserAnswer(answer: "respuesta del usuario",
question: question1)

        expect(user_answer.answer).to eq("respuesta del usuario")

        expect(user_answer.correct).to be false

        expect(user_answer.question).to eq(question1)

        user_answer.destroy

      end

    end
```

En este test se asegura de que la respuesta fue creada exitosamente, comparando todos sus campos a los valores esperados.

Game_data_manager y Myapp

```
context "when playing" do

    before(:each) do

      get '/logout'

      u = User.new(name: 'Homero Simpson', password: 'callefalsa123',
mail: 'hs@example.com')

      u.save

      i = 1

      while i <= 6

        UserLevel.create(user: u, level: Level.find(i), userLevelScore:
500)

        i += 1

      end

      post '/login', name: 'Homero Simpson', password: 'callefalsa123'

      follow_redirect!

    end


    after(:each) do

      User.find_by(name: 'Homero Simpson').destroy

    end
```

Se crea un usuario con todos los niveles desbloqueados y se destruye luego de cada “it”.

```
it "enters to a level correctly" do
      get '/level/1'
      follow_redirect!
      expect(last_response).to be_ok
    end

    it "unlocks exams correctly" do
      u = User.find_by(name: 'Homero Simpson')
      UserLevel.destroy_by(user: u, level: Level.find(6), userLevelScore:
500)
      gm = GameDataManager.new
      gm.unlockNextLevelFor(user: u, possiblyCompleted: Level.find(5))
      get '/level/6'
      follow_redirect!
      expect(last_request.path).to eq('/level/6/21')
    end

    it "unlocks levels correctly" do
      u = User.find_by(name: 'Homero Simpson')
      UserLevel.destroy_by(user: u, level: Level.find(6))
      UserLevel.destroy_by(user: u, level: Level.find(5))
      gm = GameDataManager.new
      gm.unlockNextLevelFor(user: u, possiblyCompleted: Level.find(4))
      get '/level/5'
      follow_redirect!
      expect(last_request.path).to eq('/level/5/17')
    end

    it "enters to an exam correctly" do
```

```
      get '/level/6'
      follow_redirect!
      expect(last_response).to be_ok
    end

    it "shows the profile page correctly" do
      get '/perfil'
      expect(last_response).to be_ok
    end

    it "doesn't unlock a level without having beaten the previous one
first" do
      u = User.find_by(name: 'Homero Simpson')
      UserLevel.destroy_by(user: u, level: Level.find(6))
      UserLevel.destroy_by(user: u, level: Level.find(5))
      gm = GameDataManager.new
      gm.unlockNextLevelFor(user: u, possiblyCompleted: Level.find(5))
      get '/level/6'
      expect(last_response).to be_redirect
      follow_redirect!
      expect(last_request.path).to eq("/jugar")
    end
```

Chequear respuesta

```
it "checks an answer correctly (User-input)" do
      get '/level/1'
      follow_redirect!
      post '/level/1/1/check', user_guess: 'a'
      follow_redirect!
      expect(last_request.path).to eq('/level/1/2')
    end

    it "checks an answer correctly (Multiple-choice)" do
      get '/level/2'
      follow_redirect!
      post '/level/1/2/check', answer_id: '2'
      follow_redirect!
      expect(last_request.path).to eq('/level/1/3')
    end
```

Nivel, pregunta o respuesta inexistente.

```
it "redirects to the level selection page when trying to access an
invalid question" do
      get '/level/1/21'
      follow_redirect!
      expect(last_request.path).to eq('/jugar')
    end
```

```
    it "redirects to the level selection page when trying to access an
invalid level" do
      get '/level/100/1'
      follow_redirect!
      expect(last_request.path).to eq('/jugar')
    end


    it "redirects to the level selection page when trying to answer an
invalid question" do
      post '/level/1/45/check', user_guess: 'a'
      follow_redirect!
      expect(last_request.path).to eq('/jugar')
    end


    it "redirects to the level selection page when trying to answer a
question of an invalid level" do
      post '/level/100/1/check'
      follow_redirect!
      expect(last_request.path).to eq('/jugar')
    end
```

Popups

```
it "shows the level success popup correctly" do
      get '/level/1'
      follow_redirect!
      post '/level/1/1/check', user_guess: 'a'
      follow_redirect!
      post '/level/1/2/check', answer_id: '1'
      follow_redirect!
      post '/level/1/3/check', user_guess: 'b'
      follow_redirect!
      post '/level/1/4/check', user_guess: '.-'
      expect(last_response).to be_ok
    end


    it "shows the level failed popup correctly" do
      get '/level/1'
      follow_redirect!
      post '/level/1/1/check', user_guess: 'abc'
      follow_redirect!
      post '/level/1/2/check', answer_id: '4'
      follow_redirect!
      post '/level/1/3/check', user_guess: 'bca'
      follow_redirect!
      post '/level/1/4/check', user_guess: '.'
      expect(last_response).to be_ok
    end
```

Login Incorrecto

```
it "shows an error message if log in credentials are incorrect" do

      post '/login', name: 'Homero Simpson', password: 'callefalsa123'

      expect(last_response.body).to include("Nombre de usuario o
contraseña son incorrectas")

    end
```

Conclusión

All Files (100.0% covered at 5.48 hits/line)

15 files in total.
505 relevant lines, **505** lines covered and **0** lines missed. (100.0%)

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
controllers/game_data_manager.rb	100.00 %	89	50	50	0	20.02
controllers/questions_manager.rb	100.00 %	49	25	25	0	7.36
models/answer.rb	100.00 %	4	3	3	0	1.00
models/exam.rb	100.00 %	4	3	3	0	1.00
models/lesson.rb	100.00 %	4	3	3	0	1.00
models/level.rb	100.00 %	14	10	10	0	1.10
models/mouse_translation.rb	100.00 %	6	3	3	0	1.00
models/multiple_choice.rb	100.00 %	4	3	3	0	1.00
models/question.rb	100.00 %	15	10	10	0	1.30
models/to_complete.rb	100.00 %	4	3	3	0	1.00
models/translation.rb	100.00 %	4	3	3	0	1.00
models/user.rb	100.00 %	5	4	4	0	1.00
models/user_level.rb	100.00 %	5	4	4	0	1.00
myapp.rb	100.00 %	183	118	118	0	6.91
spec/test_spec.rb	100.00 %	392	263	263	0	2.72

Estos son los resultados finales obtenidos luego de implementar los cambios. Donde pudimos avanzar más fué en el archivo question manager, ya que era el que menor cobertura poseía entre todos los archivos. En el caso de los demás archivos, fue

suficiente con simular una sesión de juego en el programa y verificar algunas direcciones que faltaban por cubrir.