Anatomy Easy

SIMPLECOV

Ingeniería en Software (3304)
Integrantes:

- Gutierrez Marysol
- Palacios Micaela

INTRODUCCIÓN

Este informe tiene como objetivo evaluar y documentar la cobertura de código de nuestra aplicación web utilizando la herramienta SimpleCov.

La cobertura de código es un indicador clave de la calidad de las pruebas, ya que mide el porcentaje de código que es ejecutado por los tests automatizados.

Inicialmente, se realizó un análisis de la cobertura existente para identificar áreas críticas que requerían mayor atención. A partir de esta evaluación, se realizaron modificaciones y mejoras en los tests con el fin de incrementar la cobertura y confiabilidad del software.

Este informe presenta los resultados obtenidos antes y después de aplicar dichas mejoras, destacando las áreas que han experimentado un mayor incremento en la cobertura, así como aquellas que aún presentan oportunidades para una cobertura más exhaustiva.

HERRAMIENTAS UTILIZADAS

En el desarrollo y evaluación de la cobertura de código de nuestra aplicación, se utilizaron las siguientes herramientas:

- **SimpleCov**: Es una herramienta de Ruby que permite medir la cobertura de código en las pruebas automatizadas. SimpleCov genera un informe detallado que muestra qué líneas de código han sido ejecutadas durante la ejecución de los tests y cuáles no. Este informe es clave para identificar áreas del código que requieren pruebas adicionales.

- **RSpec**: Es un framework de testing en Ruby diseñado para la escritura de pruebas de comportamiento de manera clara y expresiva. RSpec se utilizó para desarrollar y ejecutar los tests que verifican el correcto funcionamiento de las distintas rutas y funcionalidades de la aplicación.

- **Sinatra**: No es una herramienta de testing, es un framework ligero de Ruby sobre el que se construyó la aplicación. La configuración de Sinatra fue fundamental para garantizar que las rutas y los controladores de la aplicación estuvieran cubiertos por las pruebas.

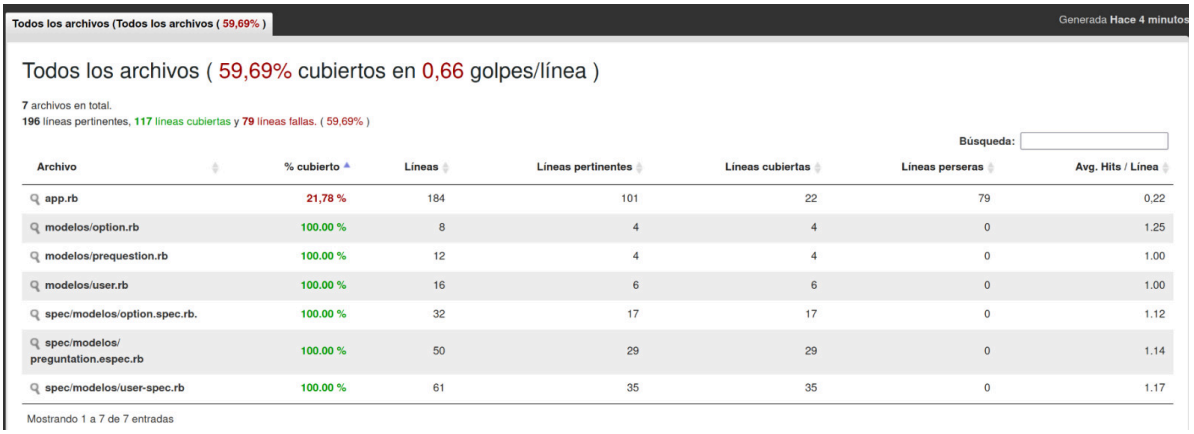
- **Database Cleaner**: Se empleó para limpiar la base de datos entre cada ejecución de prueba, asegurando que las pruebas se ejecutarán en un estado consistente y libre de interferencias entre ellas.

COBERTURA DE CÓDIGO INICIAL Y MEJORAS

Al inicio de nuestro análisis, la cobertura de código de la aplicación mostraba varias áreas que no estaban completamente cubiertas por las pruebas.

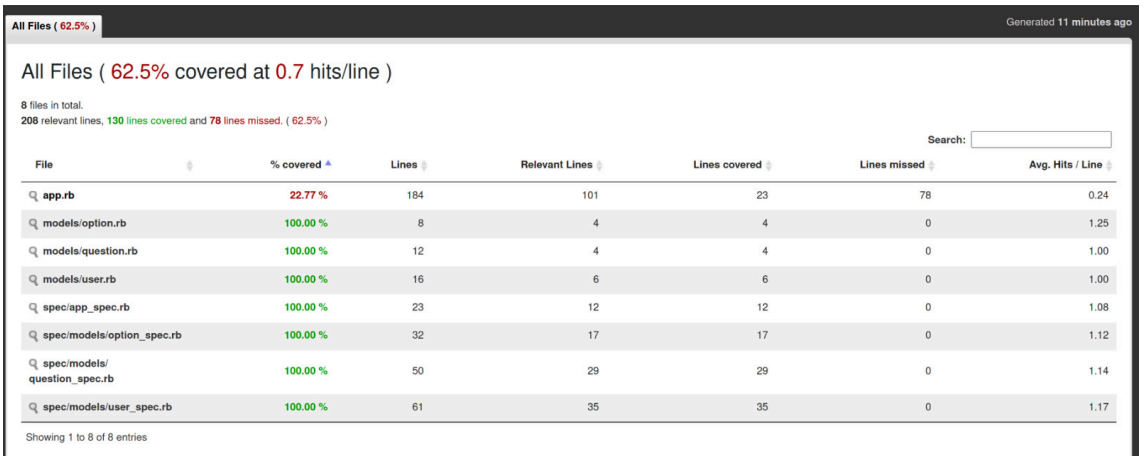
Según el informe de SimpleCov, la cobertura total era del **93.29%**, lo que indica que un 6.71% del código no estaba siendo ejecutado durante las pruebas.

Ciertas rutas y métodos de la aplicación no estaban cubiertos, lo que generaba un incorrecto funcionamiento.



Áreas con baja cobertura

Se identificaron a la **app.rb** con baja cobertura ya que no habíamos creado pruebas para ella. Luego, creamos algunas pruebas y verificando los test, para ver posibles errores.
Inicialmente, se creó una prueba para ver el comienzo de la app (**home**), lo cual, a medida que realizamos tests, corroboramos las siguientes rutas.

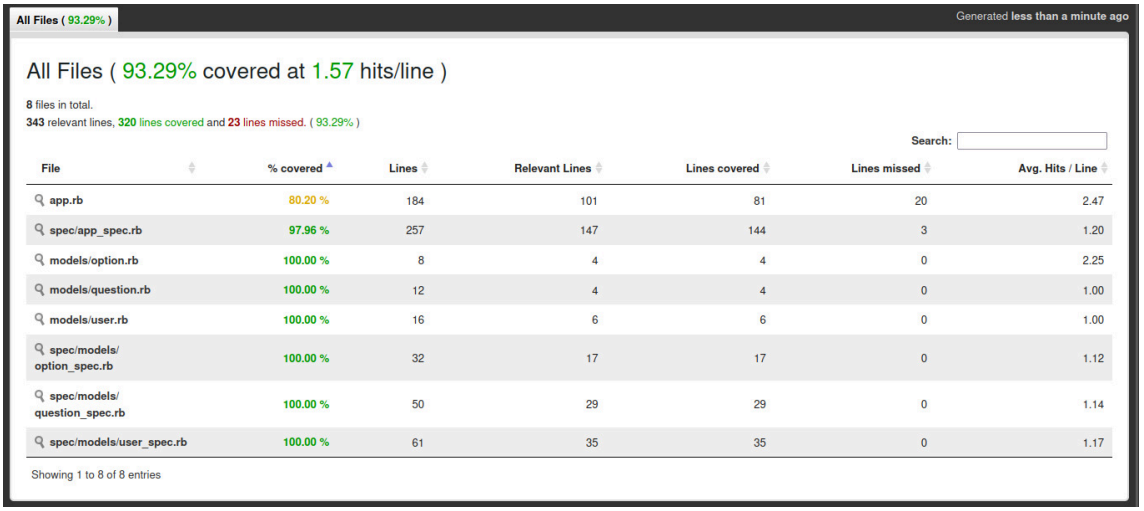


```
+ # spec/app_spec.rb
+ require 'spec_helper'
+
+ RSpec.describe 'App Routes', type: :request do
+   include Rack::Test::Methods
+
+   def app
+     Sinatra::Application
+   end
+
+   context 'GET /' do
+     it 'returns a successful response' do
+       get '/'
+       expect(last_response).to be_ok
+     end
+
+     it 'renders the home page content' do
+       get '/'
+       expect(last_response.body).to include("Bienvenido a \"AnatomyEasy\"")
+     end
+   end
+
+ end
```

En este caso, se fueron agregando/modificando pruebas a las diferentes rutas, como **login, register, select_system, y las rutas del proyecto.**

Ruta de login

```
+   context 'GET /login' do
+     it 'returns a successful response' do
+       get '/login'
+       expect(last_response).to be_ok
+     end
+
+     it 'renders the login page content' do
+       get '/login'
+       expect(last_response.body).to include("Iniciar Sesión")
+     end
+   end
+
```



En las rutas de Login y Registro, añadimos pruebas para verificar que las rutas de login (GET /login, POST /login) y registro (GET /register, POST /register) funcionen correctamente tanto con datos válidos como inválidos. Por lo cual, los usuarios puedan autenticarse y registrarse adecuadamente. También incluimos pruebas para manejar situaciones en las que las credenciales sean incorrectas o los campos estén incompletos durante el registro.

Implementamos pruebas para verificar que las rutas protegidas, como GET /select_system, para que se redirijan a la página de login si el usuario no está autenticado. Por lo que, solo los usuarios autenticados puedan acceder a ciertas funcionalidades.

Verificamos que la página de selección del sistema (GET /select_system) se carga correctamente cuando el usuario está autenticado y que se redirige al login cuando no lo está.

Añadimos pruebas para asegurarnos de que el juego inicie correctamente con la ruta POST /start_play, garantizando que la sesión del usuario se maneje adecuadamente y que se redirija a la primera pregunta del juego.

Implementamos pruebas para verificar que la ruta POST /play/question avance a la siguiente pregunta después de responder, y que muestre el mensaje correcto o incorrecto según la respuesta dada.

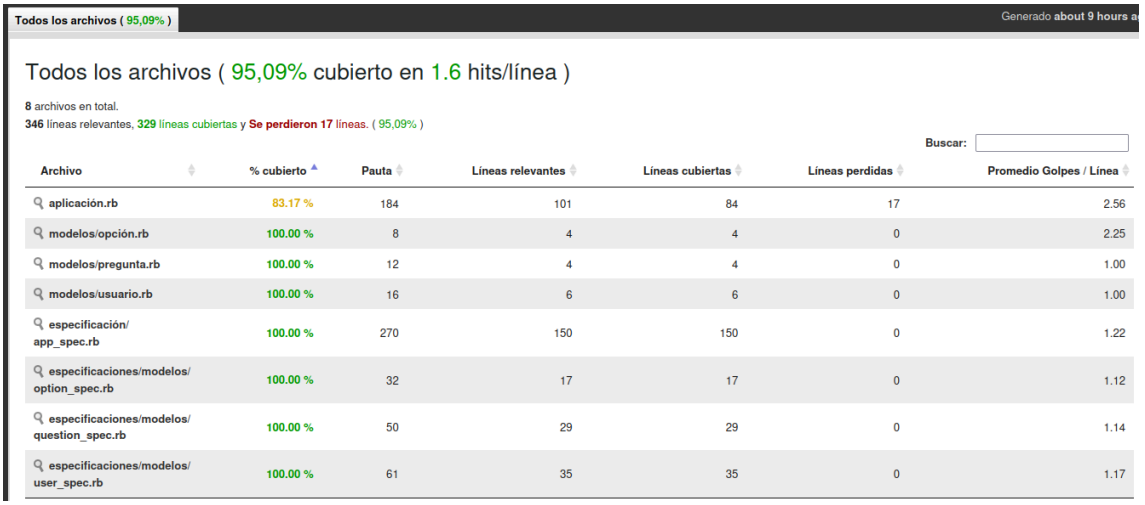
También incluimos pruebas para asegurar que, cuando no hay más preguntas disponibles, se redirige al usuario a la página de selección del sistema (/select_system), garantizando una correcta finalización de juego.

Además, añadimos una prueba para asegurarnos de que la página de fin del juego (GET /game_over) se muestre correctamente con el último mensaje.

Implementamos pruebas para verificar que la sesión del usuario se borre correctamente al acceder a la ruta de logout (GET /logout) y que el usuario sea redirigido a la página de inicio.

Estas mejoras se realizaron para asegurar que todas las rutas críticas de la aplicación estén cubiertas por pruebas, que se manejen adecuadamente los errores y que el juego funcione sin problemas, quedando algunas modificaciones pendientes.

A continuación mostramos el progreso de cobertura a medida que fuimos modificando/agregando pruebas.



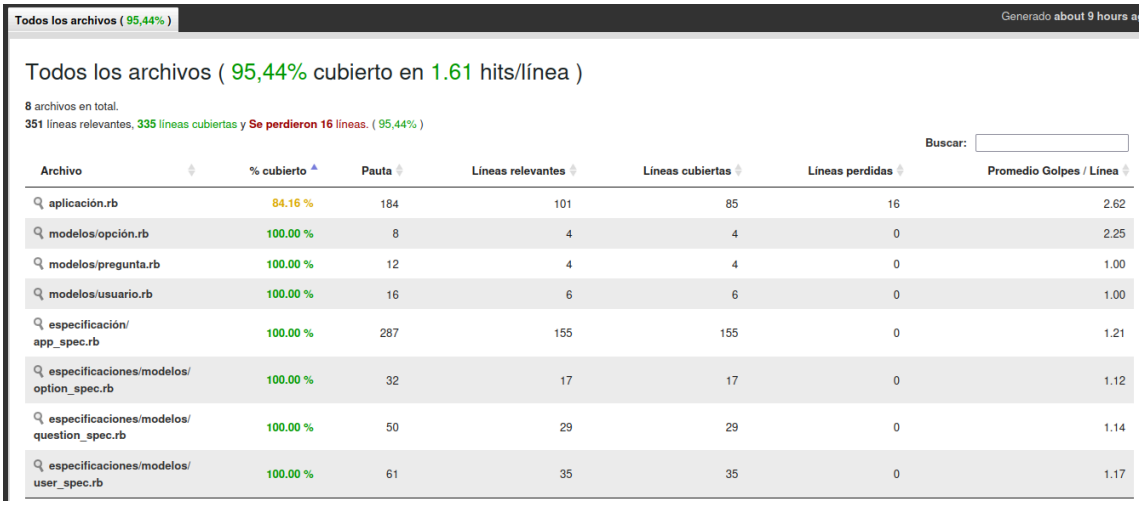
Todos los archivos (95,09%) Generado about 9 hours a

Todos los archivos (95,09% cubierto en 1.6 hits/línea)

8 archivos en total.
346 líneas relevantes, 329 líneas cubiertas y Se perdieron 17 líneas. (95,09%)

Buscar:

Archivo	% cubierto	Pauta	Líneas relevantes	Líneas cubiertas	Líneas perdidas	Promedio Golpes / Línea
aplicación.rb	83.17 %	184	101	84	17	2.56
modelos/opción.rb	100.00 %	8	4	4	0	2.25
modelos/pregunta.rb	100.00 %	12	4	4	0	1.00
modelos/usuario.rb	100.00 %	16	6	6	0	1.00
especificación/ app_spec.rb	100.00 %	270	150	150	0	1.22
especificaciones/modelos/ option_spec.rb	100.00 %	32	17	17	0	1.12
especificaciones/modelos/ question_spec.rb	100.00 %	50	29	29	0	1.14
especificaciones/modelos/ user_spec.rb	100.00 %	61	35	35	0	1.17



Todos los archivos (95,44%) Generado about 9 hours a

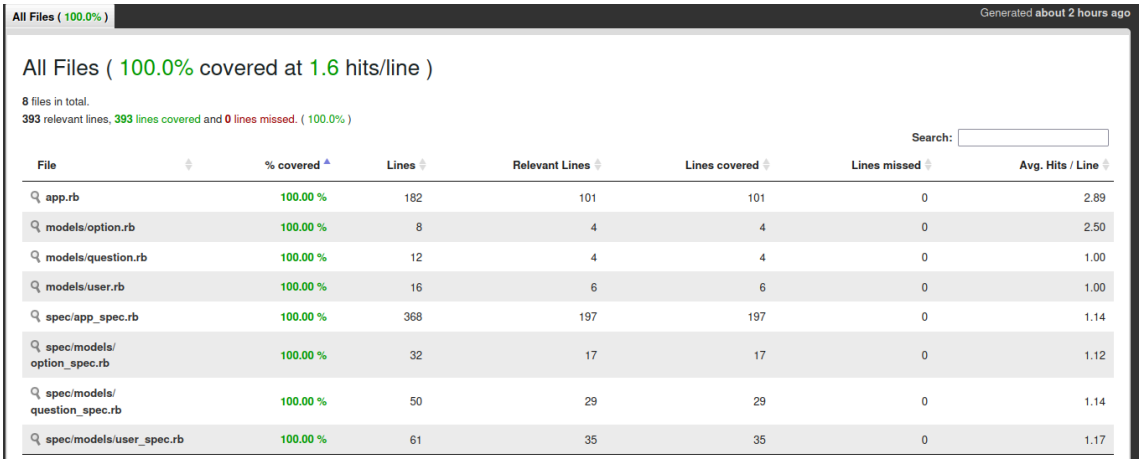
Todos los archivos (95,44% cubierto en 1.61 hits/línea)

8 archivos en total.
351 líneas relevantes, 335 líneas cubiertas y Se perdieron 16 líneas. (95,44%)

Buscar:

Archivo	% cubierto	Pauta	Líneas relevantes	Líneas cubiertas	Líneas perdidas	Promedio Golpes / Línea
aplicación.rb	84.16 %	184	101	85	16	2.62
modelos/opción.rb	100.00 %	8	4	4	0	2.25
modelos/pregunta.rb	100.00 %	12	4	4	0	1.00
modelos/usuario.rb	100.00 %	16	6	6	0	1.00
especificación/ app_spec.rb	100.00 %	287	155	155	0	1.21
especificaciones/modelos/ option_spec.rb	100.00 %	32	17	17	0	1.12
especificaciones/modelos/ question_spec.rb	100.00 %	50	29	29	0	1.14
especificaciones/modelos/ user_spec.rb	100.00 %	61	35	35	0	1.17

Finalmente, pudimos agregar/modificar pruebas para llegar a la cobertura total, implementando /**submit_evaluation** y **ready_for_evaluation** y modificando las pruebas de **play/questions y register.**


All Files (100.0%)
Generated about 2 hours ago
All Files (100.0% covered at 1.6 hits/line)
8 files in total.
393 relevant lines, 393 lines covered and 0 lines missed. (100.0%)
Search:
File
% covered
Lines
Relevant Lines
Lines covered
Lines missed
Avg. Hits / Line
app.rb 100.00 % 182 101 101 0 2.89
models/option.rb 100.00 % 8 4 4 0 2.50
models/question.rb 100.00 % 12 4 4 0 1.00
models/user.rb 100.00 % 16 6 6 0 1.00
spec/app_spec.rb 100.00 % 368 197 197 0 1.14
spec/models/option_spec.rb 100.00 % 32 17 17 0 1.12
spec/models/question_spec.rb 100.00 % 50 29 29 0 1.14
spec/models/user_spec.rb 100.00 % 61 35 35 0 1.17

CONCLUSIÓN DEL INFORME

Se mejoró la cobertura de código con la agregación de pruebas exhaustivas para todas las rutas clave de la aplicación, como inicio de sesión, registro, selección de sistema, lección, juego y evaluación, lo cual reduce el riesgo de errores en producción.

A medida que el proyecto avance, se van a seguir implementando continuas pruebas y tests para ejecutar en cada cambio de código.