05/09/2023

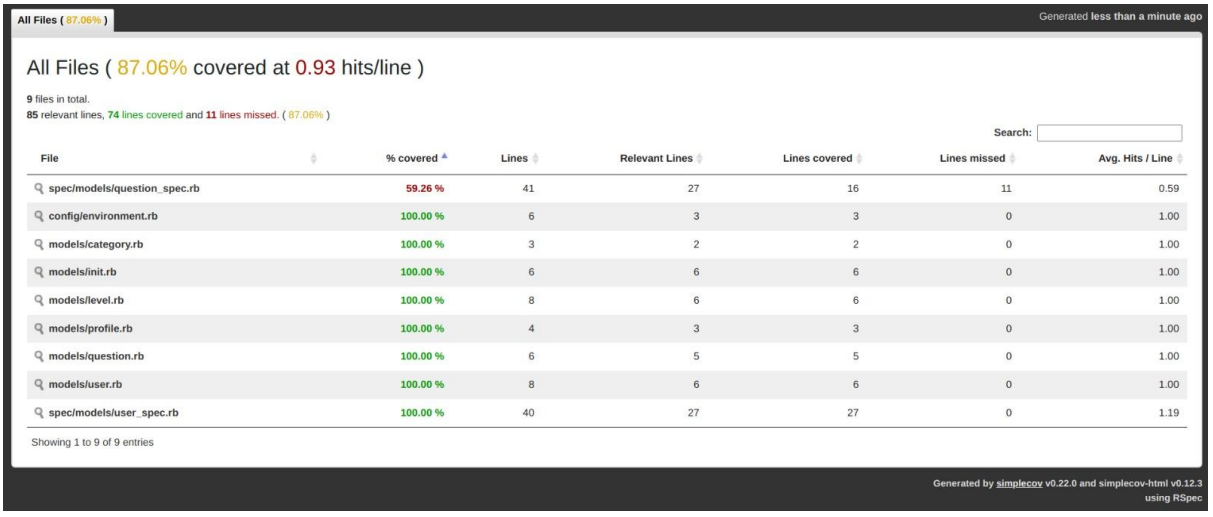
Informe
Testing Inicial (CulturAR)

Introducción:

Comenzamos los test solo teniendo con anterioridad los test de los **usuarios** al momento de usar **SimpleCov** nos daba 100% de cobertura porque no se validaban todos los tests que habíamos provisto, para ello construimos nuevos métodos para validar estos casos en el **modelo user**.

Modelo User:

Creamos test para validar que no se acepte un nombre de usuario en blanco.Que el email sea único y no lo contenga ningún otro usuario, de esta forma podemos evitar que una misma persona tenga varios usuarios.
Agregamos que el email sea de un formato válido y también que la contraseña tenga un mínimo de 5 caracteres. De esta forma tenemos estas condiciones que se deben cumplir para que un usuario se tome como válido.


All Files (87.06%)

Generated less than a minute ago

All Files (87.06% covered at 0.93 hits/line)

9 files in total.
85 relevant lines, 74 lines covered and 11 lines missed. (87.06%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
spec/models/question_spec.rb	59.26 %	41	27	16	11	0.59
config/environment.rb	100.00 %	6	3	3	0	1.00
models/category.rb	100.00 %	3	2	2	0	1.00
models/init.rb	100.00 %	6	6	6	0	1.00
models/level.rb	100.00 %	8	6	6	0	1.00
models/profile.rb	100.00 %	4	3	3	0	1.00
models/question.rb	100.00 %	6	5	5	0	1.00
models/user.rb	100.00 %	8	6	6	0	1.00
spec/models/user_spec.rb	100.00 %	40	27	27	0	1.19

Showing 1 to 9 of 9 entries

Generated by simplecov v0.22.0 and simplecov-html v0.12.3 using RSpec

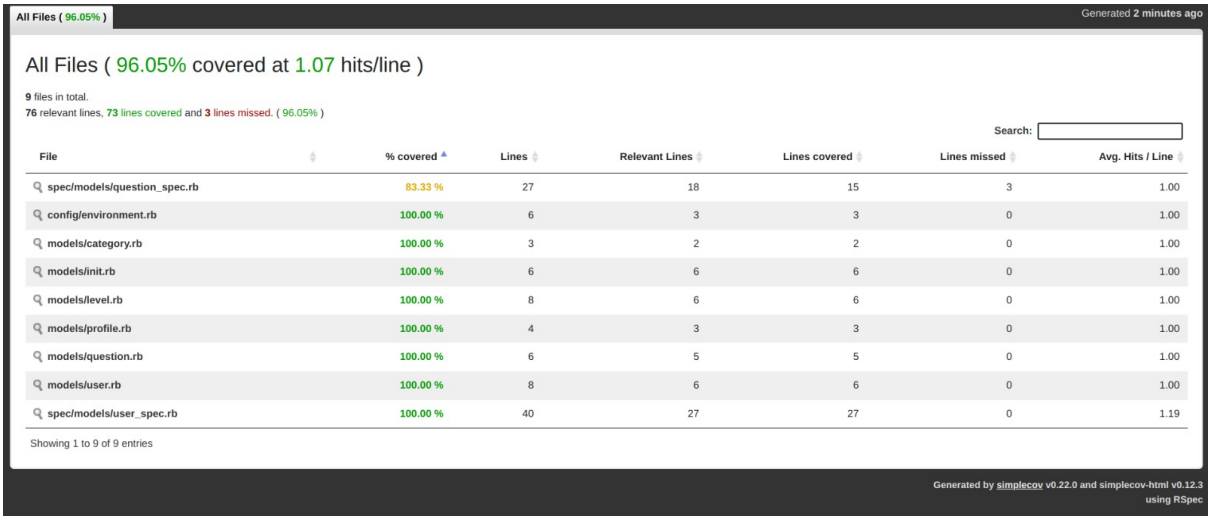
Modelo Question:

Para asegurar la validez de una **pregunta**, comenzamos por verificar que no esté en blanco, ya que debe contener información para ser considerada válida. En caso de que la pregunta esté vacía, se generará un error para notificar al usuario.

Cada pregunta tiene su propio puntaje asignado, por lo tanto, nuestro primer paso fue realizar una prueba para asegurarnos de que ninguna pregunta tenga un puntaje menor o igual a cero.

Posteriormente, implementamos una prueba diseñada para evitar preguntas con respuestas repetidas. Creamos una lista que incluye todas las respuestas, tanto las correctas como las incorrectas. Luego, convertimos todas las respuestas a minúsculas utilizando la función "downcase" y verificamos que ninguna de ellas esté duplicada en el resto de la lista, utilizando el método "include?".

En cuanto a las categorías, verificamos que cada categoría tenga al menos un nivel, lo cual es un requisito fundamental. Además, nos aseguramos de que cada categoría tenga un nombre válido, es decir, que no esté vacío.


All Files (96.05%)

Generated 2 minutes ago

All Files (96.05% covered at 1.07 hits/line)

9 files in total.
76 relevant lines, 73 lines covered and 3 lines missed. (96.05%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
spec/models/question_spec.rb	83.33 %	27	18	15	3	1.00
config/environment.rb	100.00 %	6	3	3	0	1.00
models/category.rb	100.00 %	3	2	2	0	1.00
models/init.rb	100.00 %	6	6	6	0	1.00
models/level.rb	100.00 %	8	6	6	0	1.00
models/profile.rb	100.00 %	4	3	3	0	1.00
models/question.rb	100.00 %	6	5	5	0	1.00
models/user.rb	100.00 %	8	6	6	0	1.00
spec/models/user_spec.rb	100.00 %	40	27	27	0	1.19

Showing 1 to 9 of 9 entries

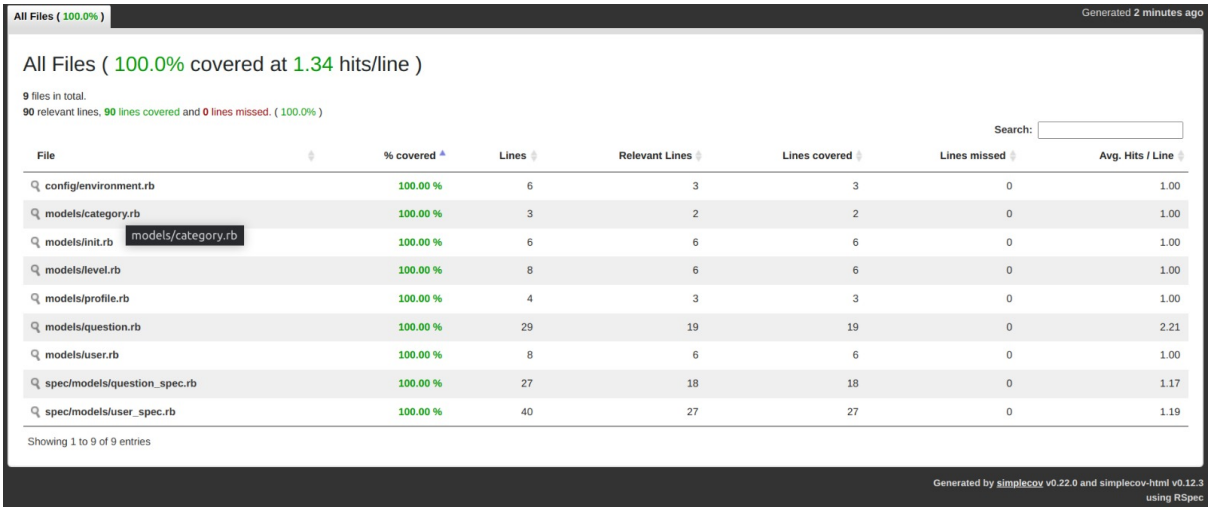
Generated by simplecov v0.22.0 and simplecov-html v0.12.3 using RSpec

Modelo Category:

En relación a las **categorías**, en primer lugar, verificamos que cada una de ellas cuente con al menos un nivel, lo cual es un requisito fundamental para su validez. La ausencia de niveles en una categoría no permite su funcionamiento adecuado.

Posteriormente, nos aseguramos de que cada categoría tenga un nombre válido, es decir, que no se encuentre en blanco. Esto es esencial para la identificación y clasificación adecuada de las categorías en nuestro sistema.

En resumen, nuestro proceso de verificación de categorías se enfoca en garantizar la presencia de niveles y nombres válidos, asegurando así que cumplan con los estándares necesarios para su correcto uso en nuestro sistema.



Modelo Profile:

Finalmente intentamos asegurarnos de que todos los **perfiles** estén ligados a una cuenta y viceversa, pero siempre nos saltaban errores.. Estos errores encontrados en el perfil se deben a que, en realidad, las pruebas que estábamos realizando en ActiveRecord se ven obstaculizadas por las restricciones impuestas por las relaciones en sí mismas. Cuando se ejecuta la prueba, se genera un error debido a que estas acciones no pueden llevarse a cabo con éxito.

Es importante destacar que no existe una manera de obtener resultados positivos en estas pruebas, ya que siempre arrojan resultados negativos. Esto se debe a que las restricciones inherentes a ActiveRecord hacen que ciertas acciones sean inviables, lo que se refleja en los resultados en rojo. Por lo tanto, se plantea la necesidad de explorar alternativas en cuanto a las pruebas, centrándonos en aspectos que no estén relacionados con las restricciones que ActiveRecord ya verifica por su cuenta.