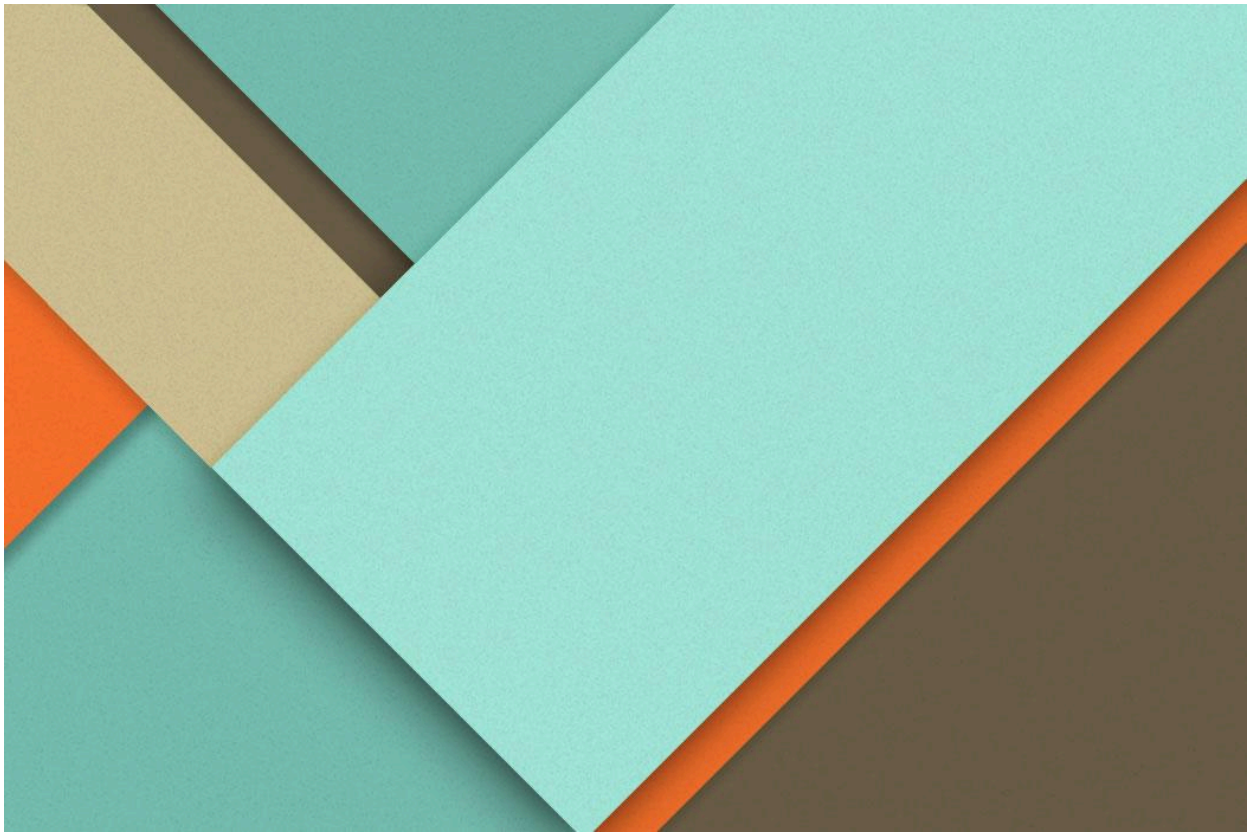

Ingeniería de Software (3304)
Taller Actividad N°1

Integrantes

Pautasso Gisel

Pautasso Stefani

Rodeghiero Tomás

1 Actividades Incompletas y Pendientes

Como equipo, hemos identificado las siguientes actividades incompletas:

- Implementación de la funcionalidad de logout.

Asimismo, las actividades pendientes que tenemos como objetivo implementar en esta materia son las siguientes:

- Cambiar la contraseña del usuario.
- Eliminar usuario del sistema.
- Agregar datos al perfil.
- Gráfico de curva de aprendizaje.
- Permitir la elección de modalidad por tiempo.
- Test general o preguntas aleatorias.
- Ranking de usuarios.
- Implementar una mascota interactiva.
- Hacer más didáctico el proceso de aprendizaje.
- Incluir imágenes y videos.
- Agregar más situaciones de emergencia.

2 Cobertura del código

2.1 Importancia de la Cobertura de código

El objetivo de esta actividad es aumentar la cobertura del código de nuestro proyecto, lo cual resulta de suma importancia ya que lograr una mayor cobertura es un objetivo crucial en el desarrollo de software debido a que mostrar un alto porcentaje de cobertura ayuda a garantizar la calidad del software. A medida que la cobertura aumenta, se incrementa también la estabilidad del código.

A raíz de ello, al tener la base de una buena cobertura del código da mayor seguridad en que las funcionalidades del sistema no presenten comportamientos inesperados y que nuevas funcionalidades se integren al proyecto adecuadamente. Esto último nos parece clave ya que representa un objetivo de la asignatura el trabajar sobre nuevas funcionalidades para nuestro proyecto.

2.2 Análisis Inicial

Una vez preparado nuestro sistema para la integración con la herramienta SimpleCov, obtuvimos un primer informe sobre la cobertura del código:

- **Cobertura general:** 94.75%
- **Cobertura en modelos:** 100%
- **Cobertura en server.rb:** 72.63%

A partir de estos resultados, concluimos que nuestros modelos estaban bien testeados. Decidimos entonces realizar un análisis exhaustivo del archivo `server.rb` para mejorar su cobertura mediante la creación de los tests necesarios.

Se puede observar el informe obtenido por SimpleCov ingresando al Informe 1 en el anexo.

2.3 Funcionamiento de Logout

Durante la revisión de la cobertura, detectamos que la funcionalidad de logout presentaba errores de referencia y no cumplía con la semántica del protocolo HTTP. Corregimos esta situación en el archivo `server.rb`, dejando solo un método `POST` para el logout, que es lo correcto, dado que modifica el estado del servidor.

Luego de las correcciones, creamos un test denominado `logout_functionality` para verificar el correcto funcionamiento de esta funcionalidad. Durante esta prueba, identificamos un error en la redirección a la página de bienvenida, que fue corregido. Tras estos cambios, los resultados de la cobertura fueron:

- **Cobertura general:** 95.84%
- **Cobertura en server.rb:** 77.17%

Se puede observar el informe obtenido por SimpleCov ingresando al Informe 2 en el anexo.

2.4 Acceso a vistas

Observamos que la mayoría de las líneas de código no cubiertas en `server.rb` correspondían a la carga de vistas. Para asegurar que las rutas definidas carguen correctamente las vistas esperadas, construimos una serie de tests de integración que verifican la respuesta del servidor y la presencia de elementos esperados en las páginas. Los tests creados fueron:

- `progress_page`

- `section_page`
- `sections_page`
- `lesson_page`
- `test_page`
- `profile_page`
- `welcome_page`
- `login_page`
- `registration_page`

Tras la inclusión de estos tests, los resultados fueron:

- **Cobertura general:** 99.51%
- **Cobertura en server.rb:** 96.74%

Se puede observar el informe obtenido por SimpleCov ingresando al Informe 3 en el anexo.

2.5 Login denegado

En cuanto al post de login, detectamos que las líneas de código correspondientes al manejo de credenciales inválidas no estaban cubiertas. Diseñamos el test `login_invalid_credentials` para cubrir este caso, asegurando que se muestre un mensaje de error en la vista de login al fallar el acceso. Luego de incluir este test, los resultados fueron:

- **Cobertura general:** 99.84%
- **Cobertura en server.rb:** 98.91%

Se puede observar el informe obtenido por SimpleCov ingresando al Informe 4 en el anexo.

2.6 Rutas protegidas y rutas públicas

Por último analizamos el caso de la cláusula before en el server la cual controla que no se pueda acceder a ciertas rutas de la aplicación si no se tienen las credenciales adecuadas, es decir si no se ingresa con un usuario válido al sistema no se pueden acceder a ciertas rutas para así evitar errores.

Se dan como excepción ciertas rutas que podríamos definir como “rutas públicas”, las cuales pueden ser accedidas sin la necesidad de tener un usuario. Estas abarcan la página de bienvenida de la aplicación como también las de login y register.

Además nuestra lógica dentro del caso de que se quisiera ingresar a rutas protegidas es que se redirija automáticamente a la ruta de login. Para lograr lo anterior creamos un test `authentication_redirect` donde se busca probar que si se intenta ingresar a una ruta

protegida, como por ejemplo la del perfil de usuario, sin credenciales válidas se redirige al usuario a la ruta de login.

Pensando en los casos en que se pueda acceder a las rutas públicas sin necesitar ingresar al sistema notamos que ya los test de integración vistos en la sección 2.4 hechos sobre las páginas de login, register y la página de bienvenida cubren estos casos ya que en las pruebas no se utiliza ningún usuario para acceder a estas rutas y se probó que se podía ingresar sin problemas.

Así mismo si pensamos el caso de que se pueda acceder a las rutas protegidas sólo si se ha ingresado al sistema, pruebas vistas en la sección 2.4 cómo puede ser profile_page prueban este caso.

Luego de incluir el test `authentication_redirect`, los resultados fueron:

- **Cobertura general:** 100.0%
- **Cobertura en server.rb:** 100.00%

Se puede observar el informe obtenido por SimpleCov ingresando al Informe 5 en el anexo.

2.7 Conclusiones finales

La actividad nos permitió encontrar errores que no habíamos sido capaces de detectar anteriormente además de que logramos aumentar significativamente el porcentaje de cobertura de nuestro código, lo cual nos permitirá introducir nuevas funcionalidades al sistema en un futuro de forma más segura.

Anexo: Resultados de los Tests

Los resultados detallados de la cobertura de código obtenidos con SimpleCov para los tests de RSpec pueden verse en el siguiente enlace:

Informes de cobertura