

Taller de Ingeniería de Software

Profesores: Marcelo Uva, Ariel Arsauté.

Integrantes del equipo 5: Genaro Salomone, Cristian Herrera.

Informe actividad 1



Nuestro proyecto fue entregado a la materia Análisis y Diseño de Sistemas completamente funcional y con solo dos casos de test. Un caso de test para el modelo User y otro caso de test para el modelo Difficulty, los respectivos archivos de test implementados fueron `user_spec.rb` y `difficulty_spec.rb`.

Ya cursando la materia Ingeniería de Software procedimos a instalar la herramienta de medición de cobertura *simpleCov* en nuestro proyecto y obtener información acerca del porcentaje de cobertura de los modelos y test. Además procedimos a implementar los demás test que faltaban para los modelos restantes como Question, Choice, Autocomplete, True_False, Answer, QuestionAnswer, Trivia y Ranking.



El primer test implementado fue `choice_spec.rb` para el modelo Choice,

Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/choice_spec.rb	 100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/difficulty_spec.rb	100.00 %	71	45	45	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/user_spec.rb	100.00 %	62	37	37	0

el segundo y tercer test implementados fueron `autocomplete_spec.rb` y `true_false_spec.rb` para los modelos Autocomplete y True_False,

Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/autocomplete_spec.rb	 100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/choice_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/difficulty_spec.rb	100.00 %	71	45	45	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/true_false_spec.rb	 100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/user_spec.rb	100.00 %	62	37	37	0

el cuarto y quinto test implementados fueron answer_spec.rb y question_answer_spec.rb para los modelos Answer y QuestionAnswer,

Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/answer_spec.rb	 100.00 %	58	33	33	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/autocomplete_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/choice_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/difficulty_spec.rb	100.00 %	71	45	45	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/question_answer_spec.rb	 100.00 %	37	25	25	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/true_false_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/user_spec.rb	100.00 %	62	37	37	0

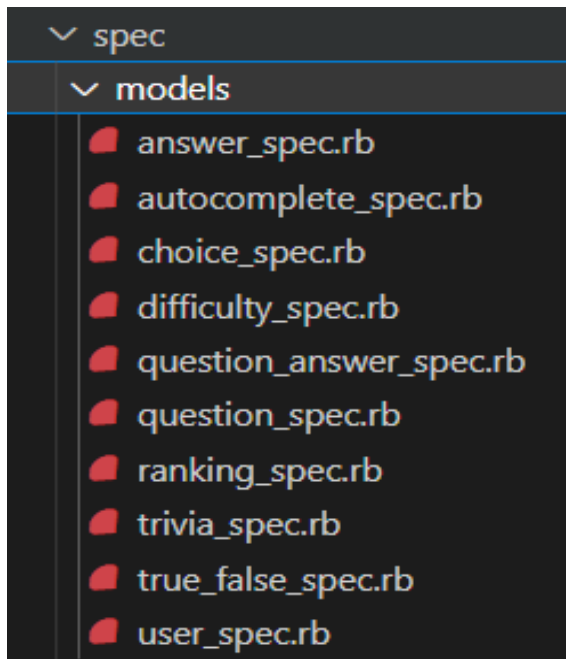
el sexto test implementado fue ranking_spec.rb para el modelo Ranking,

Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/answer_spec.rb	100.00 %	58	33	33	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/autocomplete_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/choice_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/difficulty_spec.rb	100.00 %	71	45	45	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/question_answer_spec.rb	100.00 %	38	25	25	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/question_spec.rb	100.00 %	22	15	15	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/ranking_spec.rb	100.00 %	45	23	23	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/true_false_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/user_spec.rb	100.00 %	62	37	37	0

y el último y séptimo test implementado fue trivia_spec.rb para el modelo Trivia,

Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/answer_spec.rb	100.00 %	58	33	33	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/autocomplete_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/choice_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/difficulty_spec.rb	100.00 %	71	45	45	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/question_answer_spec.rb	100.00 %	38	25	25	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/question_spec.rb	100.00 %	22	15	15	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/ranking_spec.rb	100.00 %	62	34	34	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/trivia_spec.rb	100.00 %	25	16	16	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/true_false_spec.rb	100.00 %	41	21	21	0
Q C:/Users/Camila/desktop/universidad/lenguajes/ruby/ayds_project/spec/models/user_spec.rb	100.00 %	62	37	37	0

Teniendo un total de diez test, es decir, uno para cada modelo.



Al ejecutar el comando *bundle exec rspec --require spec_helper.rb* en nuestro proyecto, los test implementados se ejecutan y pasan la prueba obteniendo un 100% de cobertura de sentencias en todos los test y modelos.

```
PS C:\Users\Camila\desktop\universidad\lenguajes\ruby\ayds_project> bundle exec rspec --require spec_helper.rb
.....
Finished in 17.5 seconds (files took 16.9 seconds to load)
62 examples, 0 failures

Coverage report generated for RSpec to C:/Users/Camila/desktop/universidad/lenguajes/ruby/AYDS_Project/coverage. 370 / 370 LOC (100.0%) covered.
```

Mejora de Robustez: Tests para nuestro endpoint post /registrarse

Se añaden 5 tests los cuales verifican escenarios específicos que manejamos en nuestro endpoint. Mejoramos la robustez de el software ya que, ahora se envían códigos de estado HTTP, los cuales se verifican en nuestros tests:

```
post '/registrarse' do
  # Verificar si las contraseñas coinciden
  if password == confirm_password
    if User.exists?(username: username)
      status 302 (Redirección) # Código HTTP 302
      redirect "/error?code=registration&reason=username_taken"
    elsif User.exists?(email: email)
      status 302 # Código HTTP 302
      redirect "/error?code=registration&reason=email_taken"
    end
  end
end
```

Ejemplo de test a el endpoint:

```
context 'when the data is valid' do
  it 'redirects the user to a success page' do
    post '/registrarse', {
      username: 'test_user',
      password: 'password',
      confirm_password: 'password',
      email: 'example@example.com'
    }

    expect(last_response.status).to eq(200)
    expect(last_response.body).to include('Vuelva a logearse por favor, vaya a inicio de sesión')
  end
end
```

Con estos nuevos tests, mejoramos la robustez de el software ya que además de ejecutar pruebas a los modelos, tenemos un endpoint el cual se verifican todas las ramas del if en escenarios específicos.

Funcionalidad agregada: Google Sign In

Vista front-end:


¡Te damos la bienvenida a la trivia!

Temática: Software y Tecnología

¿Estás listo para comenzar?

Iniciar sesión

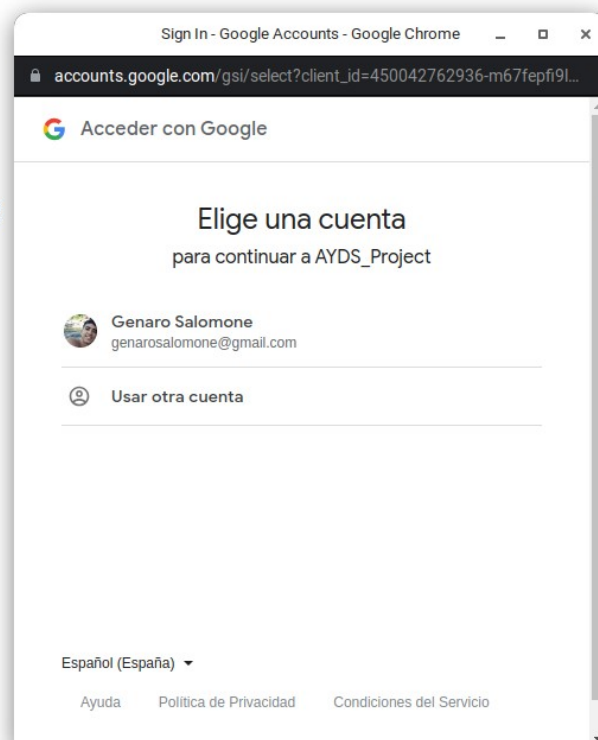
Registrarse

 Acceder con Google

Finalizar sesión Google

¡Te

via!



La funcionalidad de inicio de sesión con Google en nuestra app se logra mediante la integración de Google Sign-In (GSI) utilizando una biblioteca JavaScript proporcionada por Google. Cuando los usuarios hacen clic en el botón de inicio de sesión de Google, se llama a la función JavaScript `handleCredentialResponse`.

Esta función se encarga de enviar el token de identificación (`id_token`) al servidor de la aplicación para su verificación y procesamiento. Luego, en el servidor, se verifica el token mediante la función `google_verify`. Si el token es válido, se crea una nueva cuenta de usuario o se inicia sesión con una cuenta existente.

```
post '/google' do
  request_body = JSON.parse(request.body.read)
  id_token = request_body['id_token']

  begin
    usuario_info = google_verify(id_token)
    usuario = User.find_by(email: usuario_info[:email])
    usuario_por_username = User.find_by(username:
usuario_info[:username])

    if !usuario && !usuario_por_username
      user = User.create(username: usuario_info[:username],
email: usuario_info[:email], password: ':P')
      session[:user_id] = user.id
    else
      session[:user_id] = usuario.id if usuario
      session[:user_id] = usuario_por_username.id if
usuario_por_username
    end

    content_type :json
    {
      correo: usuario_info[:email],
      success: true,
      session: session[:user_id],
    }.to_json
  end
end
```


En este nuevo endpoint, se recibe el `id_token`, se verifica y se crea o inicia sesión en la cuenta de usuario correspondiente. El resultado se envía de vuelta al cliente, que luego redirige al usuario a una página protegida si la autenticación es exitosa.