

PC1 - CC3S2

Creación y versionado de una aplicación SaaS sencilla

¿Cuál es la diferencia entre el propósito y el contenido de Gemfile y Gemfile.lock?

El archivo de Gemfile, es el archivo donde podemos establecer las gemas que el proyecto necesite para ser instaladas, al usar bundle install se leerá este archivo y se instalarán las gemas y sus versiones que el proyecto necesite.

Por otro lado el Gemfile.lock es generado luego de correr el bundle install y contiene la información de las gemas instaladas y sus dependencias.

¿Qué archivo se necesita para reproducir completamente las gemas del entorno de desarrollo en el entorno de producción?

El archivo necesario es el Gemfile.lock ya que este tiene las versiones usadas de las gemas en el entorno de desarrollo, así como la información de las dependencias de estas.

Después de ejecutar el bundle, ¿por qué aparecen gemas en Gemfile.lock que no estaban en Gemfile?

Porque las gemas que necesitamos instalar establecidas en el Gemfile a su vez pueden necesitar otras gemas para funcionar correctamente, así que estas gemas también se agregan al Gemfile.lock para poder ver qué gemas se instalaron además de las solicitadas.

Crea una aplicación SaaS sencilla con Sinatra

Ejecutamos el comando para correr el servidor en el puerto 3000 y vemos que nos devuelve un Hello World



¿Qué sucede si intentas visitar una URL no raíz cómo https://localhost:3000/hello y por qué? (la raíz de tu URL variará)



Vemos que al no estar definida la ruta /hello, Sinatra nos devuelve una ruta por defecto indicando cómo podemos implementar esa ruta en el código.

Modifica la aplicación

Ahora actualizamos el código de app.rb para poner Goodbye World.

```
require 'sinatra'

class MyApp < Sinatra::Base
  get '/' do
    "<!DOCTYPE html><html><head></head><body><h1>Goodbye World</h1></body></html>"
  end
end
```

Si actualizamos la página veremos que aún nos dice Hello World, esto es debido a que debemos reiniciar el servidor para ver los cambios.

Volvemos a ejecutar el comando de Rack y vemos como ya cambia a Goodbye World

A screenshot of a web browser window. The address bar shows 'localhost:3000'. The page content displays 'Goodbye World' in a simple black font.

Actualizamos el Gemfile y agregamos rerun dentro de un grupo 'development'

```
group :development do
  gem 'rerun'
end
```

Ahora ya se actualiza en tiempo real.

Implementar en Heroku

Primero instalamos Heroku e iniciamos sesión.

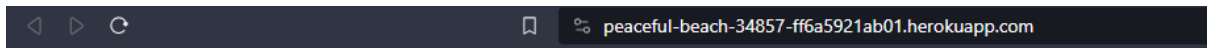
Luego hacemos un heroku create en la raíz del proyecto.

```
C:\Users\Chandler\Documents\UNI\Desarrollo-CC3S2\PC1>heroku create
» Warning: heroku update available from 7.53.0 to 8.5.0.
Creating app... done, • aqueous-ravine-01759
https://aqueous-ravine-01759-6babec68ed9c.herokuapp.com/ | https://git.heroku.com/aqueous-ravine-01759.git
```

Creamos el archivo Procfile donde estableceremos el Web Worker y subimos los cambios a Heroku

```
C:\Users\Chandler\Documents\UNI\Desarrollo-CC3S2\PC1>git push heroku master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 16 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 1.51 KiB | 1.51 MiB/s, done.
Total 13 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Updated 5 paths from 05f8bc6
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-22 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Ruby app detected
remote: -----> Installing bundler 2.3.25
remote: -----> Removing BUNDLED WITH version in the Gemfile.lock
remote: -----> Compiling Ruby/Rack
remote: -----> Using Ruby version: ruby-3.2.2
```

Y finalmente vemos que nuestra aplicación está desplegada



Go World

Parte 1: Wordguesser

Clonamos el repositorio <https://github.com/saasbook/hw-sinatra-saas-wordguesser> y corremos los test.

```
devasc@labvm:~/hw-sinatra-saas-wordguesser$ bundle exec autotest
(Not running features. To run features in autotest, set AUTOFEATURE=true.)
loading autotest/rspec
"/usr/share/rvm/rubies/ruby-2.6.6/bin/ruby" -rrubygems -S "/usr/share/rvm/gems/ruby-2.6.6/gems/rspec-core-3.3.2/exe/rspec" --tty "/home/devasc/hw-sinatra-saas-wordguesser/spec/wordguesser_game_spec.rb"
Run options: exclude {:pending=>true}

All examples were filtered out

Finished in 0.00039 seconds (files took 0.30148 seconds to load)
0 examples, 0 failures
```

Vemos que tenemos '0 examples, 0 failures', si cambiamos el pending a false, entonces veremos otro resultado

```
21
22 describe 'guessing', :pending => false do
23   context 'correctly' do
24     before :each do
25       @game = WordGuesserGame.new('garply')
26       @valid = @game.guess('a')
27     end
28     it 'changes correct guess list', :pending => true do
29       expect(@game.guesses).to eq('a')
30       expect(@game.wrong_guesses).to eq('')
31     end
  end
end

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Failures:

  1) WordGuesserGame new takes a parameter and returns a WordGuesserGame object
     Failure/Error: expect(@game.word).to eq('glorp')
     NoMethodError:
       undefined method `word' for #<WordGuesserGame:0x0000558096b56960 @word="glorp">
       # ./spec/wordguesser_game_spec.rb:16:in `block (3 levels) in <top (required)>'

Finished in 0.00992 seconds (files took 1.15 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./spec/wordguesser_game_spec.rb:13 # WordGuesserGame new takes a parameter and returns a WordGuesserGame object
```

Y así con todos los demás

```
Finished in 0.09816 seconds (files took 0.41472 seconds to load)
18 examples, 18 failures
```

Según los casos de prueba, ¿cuántos argumentos espera el constructor de la clase de juegos (identifica la clase) y, por lo tanto, cómo será la primera línea de la definición del método que debes agregar a wordguesser_game.rb?

Como vemos tiene un solo argumento, en este caso es 'glorp'

```
describe 'new' do
  it "takes a parameter and returns a WordGuesserGame object" do
    @game = WordGuesserGame.new('glorp')
    expect(@game).to be_an_instance_of(WordGuesserGame)
    expect(@game.word).to eq('glorp')
    expect(@game.guesses).to eq('')
    expect(@game.wrong_guesses).to eq('')
  end
end
```

Así que la primera línea que debemos agregar a `wordguesser_game.rb` sería el constructor con un argumento.

Según las pruebas de este bloque describe, ¿qué variables de instancia se espera que tenga `WordGuesserGame`?

Se esperan 3 variables de instancia, estas variables son `'world, guesses, world_guesses'`

Ahora con el autotest funcionando, empezamos a agregar los métodos necesarios a la clase `WordGuesserGame` y logramos superar los 18 test.

```
"/usr/share/rvm/rubies/ruby-2.6.6/bin/ruby" -rrubygems -S "/usr/share/rvm/gems/ruby-2.6.6/gems/rspec-core-3.3.2/exe/rspec" --tty "/home/devasc/hw-sinatra-saas-wordguesser/spec/wordguesser_game_spec.rb"
Run options: exclude {:pending=>true}

WordGuesserGame
  new
    takes a parameter and returns a WordGuesserGame object
  guessing
    correctly
      changes correct guess list
      returns true
    incorrectly
      changes wrong guess list
      returns true
  same letter repeatedly
    does not change correct guess list
    does not change wrong guess list
    returns false
  is case insensitive
  invalid
    throws an error when empty
    throws an error when not a letter
    throws an error when nil
  displayed word with guesses
    should be 'b-n-n-' when guesses are 'bn'
    should be '-----' when guesses are 'def'
    should be 'banana' when guesses are 'ban'
  game status
    should be win when all letters guessed
    should be lose after 7 incorrect guesses
    should continue play if neither win nor lose

Finished in 0.01281 seconds (files took 0.27756 seconds to load)
18 examples, 0 failures
```

Parte 2: RESTful para Wordguesser

Enumera el estado mínimo del juego que se debe mantener durante una partida de Wordguesser.

Los intentos que se han hecho, la palabra a adivinar, las letras correctas y las letras incorrectas.

Enumera las acciones del jugador que podrían provocar cambios en el estado del juego. Empezar una nueva partida y adivinar una letra de la palabra.

Para un buen diseño RESTful, ¿cuáles de las operaciones de recursos deberían ser manejadas por HTTP GET y cuáles deberían ser manejadas por HTTP POST?

Como Crear Juego y Adivinar letra, modifican el estado del juego, entonces deberíamos usar POST, en cambio Show solo nos muestra el estado del juego así que sería un GET.

¿Por qué es apropiado que la nueva acción utilice GET en lugar de POST?

Porque `new` nos va a devolver un formulario para crear una nueva partida, y esto al no cambiar el estado de nuestra aplicación no es necesario que sea POST.

Explica por qué la acción GET `/new` no sería necesaria si tu juego Wordguesser fuera llamado como un servicio en una verdadera arquitectura orientada a servicios.

Porque en una verdadera arquitectura orientada a servicios, la petición HTTP de tipo POST se generaría automáticamente y el usuario no tendría que crearla desde `/new`.

Parte 3: Conexión de WordGuesserGame a Sinatra

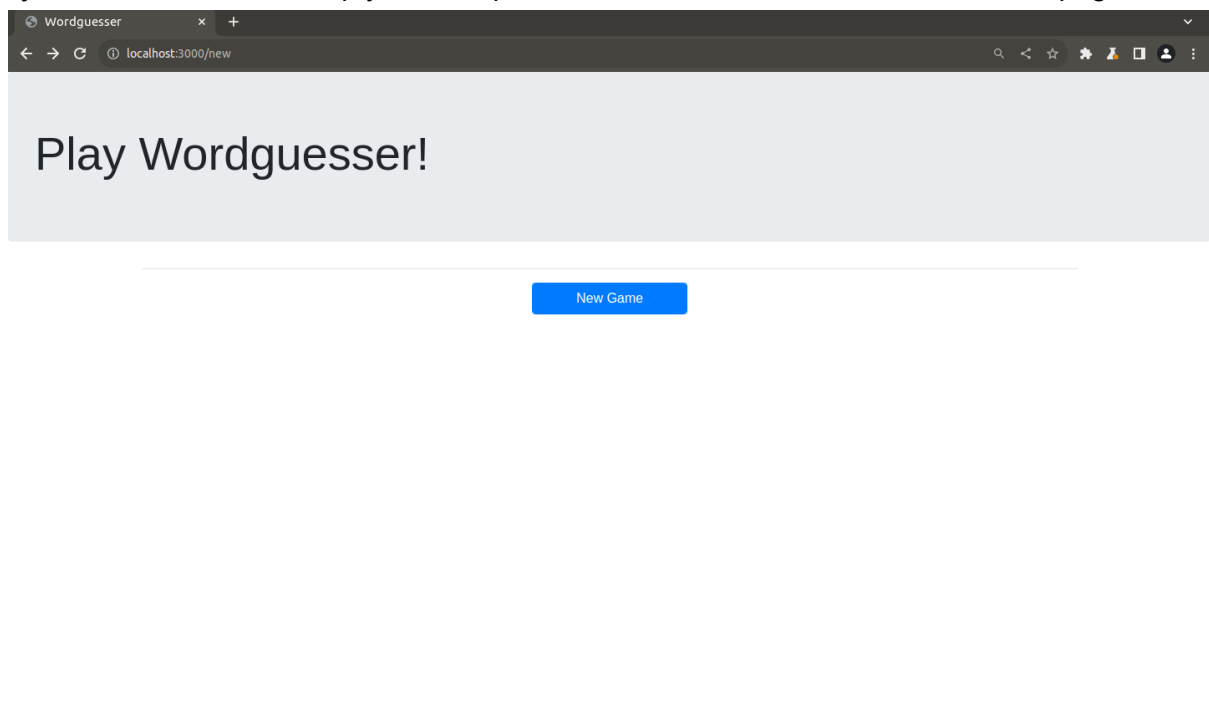
@game en este contexto es una variable de instancia de qué clase?

@game es una variable de instancia de la clase WordGuesserApp y es de tipo WordGuesserGame

¿Por qué esto ahorra trabajo en comparación con simplemente almacenar esos mensajes en el hash de sesion {}?

Los datos que almacenamos en sesion[] persisten durante toda la sesión, sin embargo habrá mensajes que queramos transmitir de forma corta sin que se queden durante toda la sesión, para esto usaremos flash[] que nos permitirá guardar información entre las redirecciones que hagamos y luego borrar esa información que ya no usaremos.

Ejecutamos usando rackup y vemos que al ir a localhost:3000 nos manda a esta pagina



Según el resultado de ejecutar este comando, ¿cuál es la URL completa que debes visitar para visitar la página New Game?

Como vemos en el código de ruby para la ruta raíz '/', nos redirige hacia la ruta /new

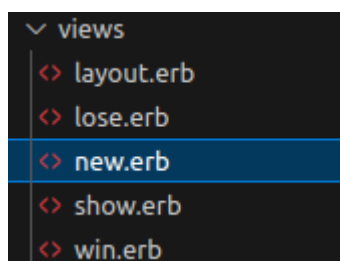
```
get '/' do
  redirect '/new'
end
```

Así que la ruta para visitar la página New Game es <http://localhost:3000/new>

¿Dónde está el código HTML de esta página?

El código html se encuentra dentro de las carpetas view, ya que al estar usando .erb este nos permite ejecutar código Ruby dentro de un HTML.

Así que el código que muestra la página está en views/new.erb



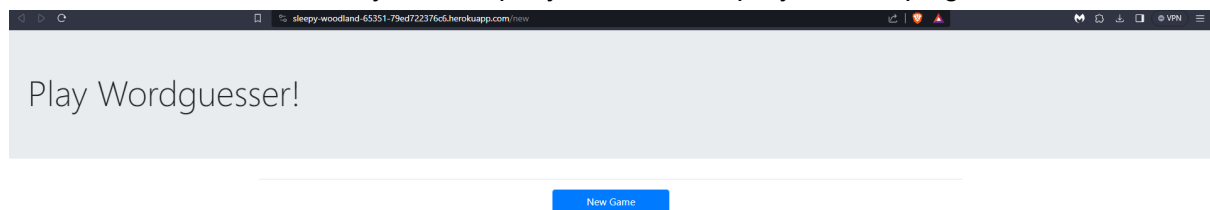
Y este es el código HTML

```
new.erb x
views > <> new.erb
1 <!-- This form is incomplete--it needs a destination URL as well as a method: -->
2 <form method="post">
3   <div class="form-row py-3 border-top">
4     <input type="submit" value="New Game" class="col-md-2 offset-md-5 btn btn-primary form-control"/>
5   </div>
6 </form>
```

Cómo estamos usando Ruby 2.6.6 debemos usar el stack de Heroku-20 para poder subir el proyecto sin problemas.

```
devasc@labvm:~/pc1-wordguesser$ heroku create --stack heroku-20
Creating app... done, 🌀sleepy-woodland-65351, stack is heroku-20
https://sleepy-woodland-65351-79ed722376c6.herokuapp.com/ | https://git.heroku.com/sleepy-woodland-65351.git
```

Ahora subimos los cambios y vemos que ya tenemos el proyecto desplegado



Parte 4: Cucumber

Lea la sección sobre " Using Capybara with Cucumber" en la página de inicio de Capybara. ¿Qué pasos utiliza Capybara para simular el servidor como lo haría un navegador? ¿Qué pasos utiliza Capybara para inspeccionar la respuesta de la aplicación al estímulo? Como vemos en la documentación nos muestra métodos como fill_in y click_button

```
When /I sign in/ do
  within("#session") do
    fill_in 'Email', with: 'user@example.com'
    fill_in 'Password', with: 'password'
  end
  click_button 'Sign in'
end
```

Estos simulan la interacción con un navegador web.

Mirando features/guess.feature, ¿cuál es la función de las tres líneas que siguen al encabezado "Feature:"?

```
Feature: guess correct letter

  As a player playing Wordguesser
  So that I can make progress toward the goal
  I want to see when my guess is correct
```

Como vemos, son definiciones de cómo debería funcionar, Cucumber no ejecuta, son de referencia.

En el mismo archivo, observando el paso del escenario Given I start a new game with word "garply" qué líneas en game_steps.rb se invocarán cuando Cucumber intente ejecutar este paso y cuál es el papel de la cadena "garply" en el paso?

```

9 When /^I start a new game with word "(.*)"/ do |word|
10   stub_request(:post, "http://randomword.saa3book.info/RandomWord").
11     to_return(:status => 200, :headers => {}, :body => word)
12   visit '/new'
13   click_button "New Game"
14 end

```

Se ejecutarán el código entre las 9 y 14, “garply” es la palabra de ejemplo que usamos, y el código lo valida mediante una expresión regular.

Ahora haremos las pruebas usando Cucumber para completar la aplicación

Cuando el "simulador de navegador" en Capybara emite la solicitud de visit '/new', Capybara realizará un HTTP GET a la URL parcial /new en la aplicación. ¿Por qué crees que visit siempre realiza un GET, en lugar de dar la opción de realizar un GET o un POST en un paso determinado?

Porque como vimos anteriormente Capybara emula un navegador web, una persona al navegar entre páginas, no puede mandar solicitudes POST sin hacer uso de un formulario, por eso ‘visit’ solo mando solicitudes GET.

Usamos el comando para probar el feature/start_new_game.feature usando Cucumber

```

devasc@labvm:~/pc1-wordguesser$ cucumber features/start_new_game.feature
Feature: start new game

  As a player
  So I can play Wordguesser
  I want to start a new game

  Scenario: I start a new game # features/start_new_game.feature:7
    Given I am on the home page # features/step_definitions/game_steps.rb:61
    And I press "New Game" # features/step_definitions/game_steps.rb:74
    Then I should see "Guess a letter" # features/step_definitions/game_steps.rb:70
      expected to find text "Guess a letter" in "Not Found" (RSpec::Expectations::ExpectationNotMetError)
      ./features/step_definitions/game_steps.rb:71:in `/(?:|I) should see "([^\"]*)"(?: within "([^\"]*)"?)?/'
      features/start_new_game.feature:11:in `Then I should see "Guess a letter"'
    And I press "New Game" # features/step_definitions/game_steps.rb:74
    Then I should see "Guess a letter" # features/step_definitions/game_steps.rb:70

Failing Scenarios:
cucumber features/start_new_game.feature:7 # Scenario: I start a new game

1 scenario (1 failed)
5 steps (1 failed, 2 skipped, 2 passed)
0m0.047s
Coverage report generated for Cucumber Features to /home/devasc/pc1-wordguesser/coverage. 30 / 65 LOC (46.15%) covered.

```

Como vemos nos da un error, ahora vamos a solucionarlo.

```

<> new.erb M x app.rb

views > <> new.erb
1 <!-- This form is incomplete--it needs a destination URL as well as a method: -->
2 <form method="post" action="/create">
3   <div class="form-row py-3 border-top">
4     <input type="submit" value="New Game" class="col-md-2 offset-md-5 btn btn-primary form-control"/>
5   </div>
6 </form>

```

En el archivo new.erb corregimos el form y le agregamos la propiedad action="/create" para que los datos vayan a esa ruta.

```

devasc@labvm:~/pc1-wordguesser$ cucumber features/start_new_game.feature
Feature: start new game

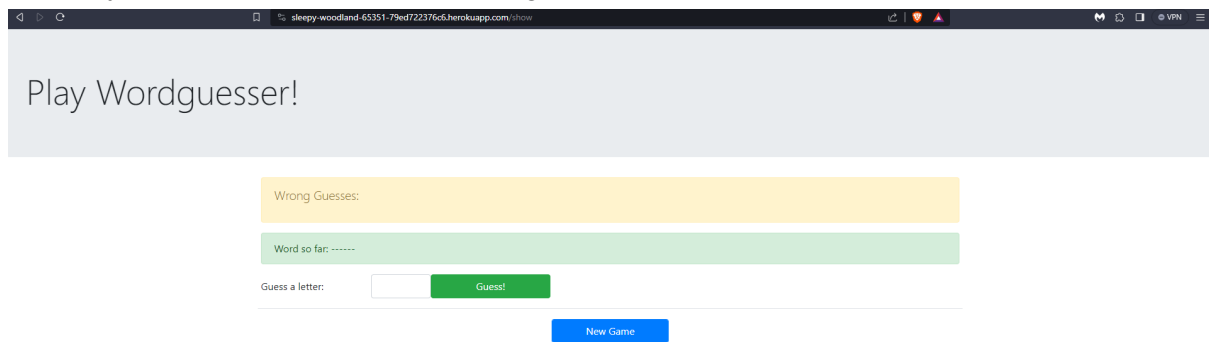
  As a player
  So I can play Wordguesser
  I want to start a new game

  Scenario: I start a new game # features/start_new_game.feature:7
    Given I am on the home page # features/step_definitions/game_steps.rb:61
    And I press "New Game" # features/step_definitions/game_steps.rb:74
    Then I should see "Guess a letter" # features/step_definitions/game_steps.rb:70
    And I press "New Game" # features/step_definitions/game_steps.rb:74
    Then I should see "Guess a letter" # features/step_definitions/game_steps.rb:70

1 scenario (1 passed)
5 steps (5 passed)
0m0.042s
Coverage report generated for Cucumber Features to /home/devasc/pc1-wordguesser/coverage. 44 / 65 LOC (67.69%) covered.

```


Como vemos ahora si pasa el escenario de Cucumber así que subimos los cambios a Heroku y probamos la aplicación desplegada.



Ahora ya podemos crear un nuevo juego.

¿Cuál es el significado de usar Given versus When versus Then en el archivo de características? ¿Qué pasa si los cambias? Realiza un experimento sencillo para averiguarlo y luego confirme los resultados utilizando Google.

'Given', 'When' y 'Then' son la misma función pero tienen distintos nombres para mejorar la legibilidad.

Ahora arreglaremos el escenario de guess.feature

```
devasc@labvm:~/pc1-wordguesser$ cucumber features/guess.feature
Feature: guess correct letter

  As a player playing Wordguesser
  So that I can make progress toward the goal
  I want to see when my guess is correct

  Scenario: guess correct letter that occurs once # features/guess.feature:7
    Given I start a new game with word "garply" # features/step_definitions/game_steps.rb:9
    When I guess "r" # features/step_definitions/game_steps.rb:16
    Then I should see "r" within "span.word" # features/step_definitions/game_steps.rb:70

  Scenario: guess correct letter that occurs multiple times # features/guess.feature:13
    Given I start a new game with word "animal" # features/step_definitions/game_steps.rb:9
    When I guess "a" # features/step_definitions/game_steps.rb:16
    Then I should see "a---a-" within "span.word" # features/step_definitions/game_steps.rb:70

  Scenario: guess incorrect letter # features/guess.feature:19
    Given I start a new game with word "xylophone" # features/step_definitions/game_steps.rb:9
    When I guess "a" # features/step_definitions/game_steps.rb:16
    Then I should see "a" within "span.guesses" # features/step_definitions/game_steps.rb:70

  Scenario: multiple correct and incorrect guesses # features/guess.feature:25
    Given I start a new game with word "foobar" # features/step_definitions/game_steps.rb:9
    When I make the following guesses: a,z,x,o # features/step_definitions/game_steps.rb:22
    Then the word should read "-oo-a-" # features/step_definitions/game_steps.rb:38
    And the wrong guesses should include: z,x # features/step_definitions/game_steps.rb:42

4 scenarios (4 passed)
13 steps (13 passed)
0m0.133s
Coverage report generated for Cucumber Features to /home/devasc/pc1-wordguesser/coverage. 55 / 66 LOC (83.33%) covered.
```

Para esto, en el archivo app.rb llamaremos al método guess(letter) de @game

```
post '/guess' do
  letter = params[:guess].to_s[0]
  @game.guess(letter)
  redirect '/show'
end
```

Ahora subimos los cambios a Heroku y probamos

Play Wordguesser!

Wrong Guesses: cursndhz

Word so far: -e-a-e

Guess a letter:

En `game_steps.rb`, mira el código del paso "I start a new game..." y, en particular, el comando `stub_request`. Dada la pista de que ese comando lo proporciona una gema (biblioteca) llamada `webmock`, ¿qué sucede con esa línea y por qué es necesaria? (Utiliza Google si es necesario).

Lo usamos para interceptar una petición HTTP y en lugar de llamarla pasar datos falsos que nosotros generamos como respuesta.

```
When /^I start a new game with word "(.*)"/ do |word|
  stub_request(:post, "http://randomword.saasbook.info/RandomWord").
    to_return(:status => 200, :headers => {}, :body => word)
  visit '/new'
  click_button "New Game"
end
```

Como vemos aquí, `stub_request` "manda" una solicitud HTTP de tipo POST para obtener una palabra random, pero en lugar de obtener la respuesta, retorna su propia respuesta con la palabra que le hemos dado, en este caso 'word'.

En tu código Sinatra para procesar una adivinación, ¿qué expresión usaría para extraer *solo el primer carácter* de lo que el usuario escribió en el campo de adivinación de letras del formulario en `show.erb`?

Como vemos en el código obtenemos el parámetro de nombre 'guess', lo convertimos a una string y obtenemos la posición 0 del string.

```
letter = params[:guess].to_s[0]
```

Parte 5: Otros casos

Ahora haremos la implementación de las rutas faltantes y corregiremos todas las pruebas de Cucumber.

Primero empecemos con la ruta `/show`

Aquí vamos a verificar la lógica si ya se adivinó la palabra o terminaron los intentos.

```
get '/show' do
  state = @game.check_win_or_lose
  if state == :win
    redirect '/win'
  end
  if state == :lose
    redirect '/lose'
  end
  erb :show # You may change/remove this line
end
```

Como ya tenemos el método para verificar si se perdió o ganar en la variable @game entonces la usamos para verificar el estado del juego si es de victorio o derrota.

```
devasc@labvm:~/pc1-wordguesser$ cucumber features/game_over.feature
Feature: game over

  As a player playing Wordguesser
  So I can get back to my life
  I want to know when the game is over

  Scenario: game over because I guess the word # features/game_over.feature:7
    Given I start a new game with word "foobar" # features/step_definitions/game_steps.rb:9
    When I make the following guesses: f,o,b,a,r # features/step_definitions/game_steps.rb:22
    Then I should see "You Win!" # features/step_definitions/game_steps.rb:70

  Scenario: game over because I run out of guesses # features/game_over.feature:13
    Given I start a new game with word "zebra" # features/step_definitions/game_steps.rb:9
    When I make the following guesses: t,u,v,w,x,y # features/step_definitions/game_steps.rb:22
    And I guess "d" # features/step_definitions/game_steps.rb:16
    Then I should see "Sorry, you lose!" # features/step_definitions/game_steps.rb:70

2 scenarios (2 passed)
7 steps (7 passed)
0m0.088s
Coverage report generated for Cucumber Features to /home/devasc/pc1-wordguesser/coverage. 68 / 71 LOC (95.77%) covered.
```

Como vemos pasa la prueba game_over.feature del Cucumber, así que ya está completo. Pero, ¿Qué pasa si ponemos /win o /lose en la url del juego?

Obviamente cambiará la vista al de la ruta especificada, así que para evitar esto podemos verificar el estado del juego y redireccionar a la ruta /show si aún no hay condición de derrota o victoria.

```
get '/win' do
  if @game.check_win_or_lose != :win
    redirect '/show'
  end
  erb :win # You may change/remove this line
end

get '/lose' do
  if @game.check_win_or_lose != :lose
    redirect '/show'
  end
  erb :lose # You may change/remove this line
end
```

Modificando esta parte ya nos aseguramos que aunque se vaya a otra ruta, siempre vuelva a /show si la condición no es de victoria o derrota, subimos los cambios a Heroku y probamos.

Pero primero probaremos los escenarios de Cucumber.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Then the word should read "b--b--b--" # features/step_definitions/game_steps.rb:38
And I should see "You have already used that letter" # features/step_definitions/game_steps.rb:70
expected to find text "You have already used that letter" in "ang=en-us">\nWordguesser\nPlay Wordguesser!\nWrong Guesses:\nWord so far: b
--b--b--\nGuess a letter:" (RSpec::Expectations::ExpectationNotMetError)
./features/step_definitions/game_steps.rb:71:in '^(?:[I ])should see "([^\"]*)"?(?: within "([^\"]*)"?)?$/
features/repeated_guess.feature:13:in 'And I should see "You have already used that letter"'

Scenario: guess incorrect letter that I have already tried # features/repeated_guess.feature:15
  Given I start a new game with word "giraffe" # features/step_definitions/game_steps.rb:9
  When I guess "z" # features/step_definitions/game_steps.rb:16
  And I guess "z" again # features/step_definitions/game_steps.rb:16
  Then the word should read "-----" # features/step_definitions/game_steps.rb:38
  And I should see "You have already used that letter" # features/step_definitions/game_steps.rb:70
  expected to find text "You have already used that letter" in "ang=en-us">\nWordguesser\nPlay Wordguesser!\nWrong Guesses: z\nWord so far:
-----\nGuess a letter:" (RSpec::Expectations::ExpectationNotMetError)
./features/step_definitions/game_steps.rb:71:in '^(?:[I ])should see "([^\"]*)"?(?: within "([^\"]*)"?)?$/
features/repeated_guess.feature:21:in 'And I should see "You have already used that letter"'

Scenario: guessing an incorrect letter does not count towards guesses # features/repeated_guess.feature:23
  Given I start a new game with word "snake" # features/step_definitions/game_steps.rb:9
  When I guess "z" 30 times in a row # features/step_definitions/game_steps.rb:49
  Then I should be on the show page # features/step_definitions/game_steps.rb:65

Feature: start new game

  As a player
  So I can play Wordguesser
  I want to start a new game

  Scenario: I start a new game # features/start_new_game.feature:7
    Given I am on the home page # features/step_definitions/game_steps.rb:61
    And I press "New Game" # features/step_definitions/game_steps.rb:74
    Then I should see "Guess a letter" # features/step_definitions/game_steps.rb:70
    And I press "New Game" # features/step_definitions/game_steps.rb:74
    Then I should see "Guess a letter" # features/step_definitions/game_steps.rb:70

Failing Scenarios:
cucumber features/invalid_guess.feature:8 # Scenario: guess an empty guess
cucumber features/invalid_guess.feature:15 # Scenario: guess a noncharacter guess
```

Vemos que hay errores así que vamos a solucionarlos.

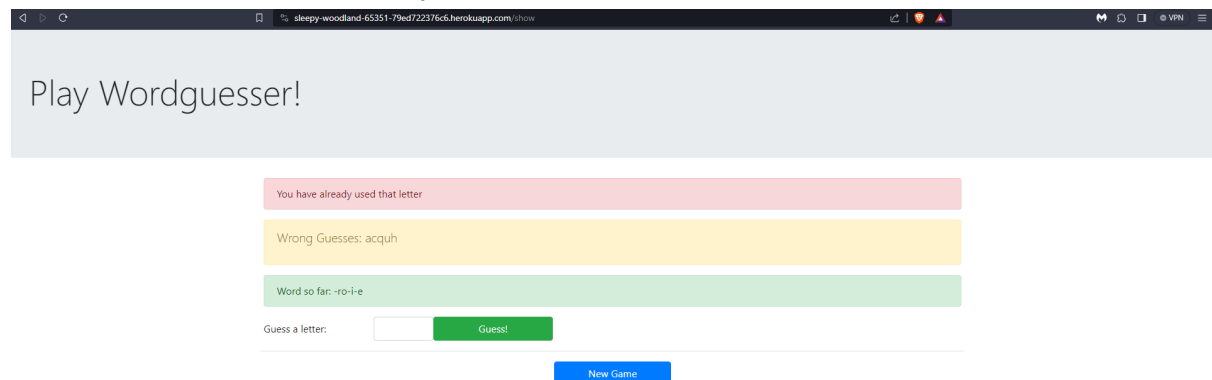
```
post '/guess' do
  letter = params[:guess].to_s[0]
  begin
    if @game.guess(letter)
      redirect '/show'
    end
    flash[:message] = "You have already used that letter"
  rescue
    flash[:message] = "Invalid guess."
  end
  redirect '/show'
end
```

Ahora estamos usando los `flash[]` para mandar los errores y usamos un `begin-rescue` para capturar los errores, en este caso el método `guess()` de `@game` nos puede dar una excepción al pasarle un argumento que no sea una letra, así que al pasar esto mandamos el mensaje “Invalid guess.”, de lo contrario si ya usamos la letra nos manda el mensaje “You have already used that letter” y finalmente si se adivina correctamente la letra entonces nos redirige a `/show`.

```
14 escenarios (14 passed)
56 steps (56 passed)
0m0.433s
```

Vemos que todos los escenarios han sido resueltos.

Subimos los cambios a Heroku y probamos.



Play Wordguesser!

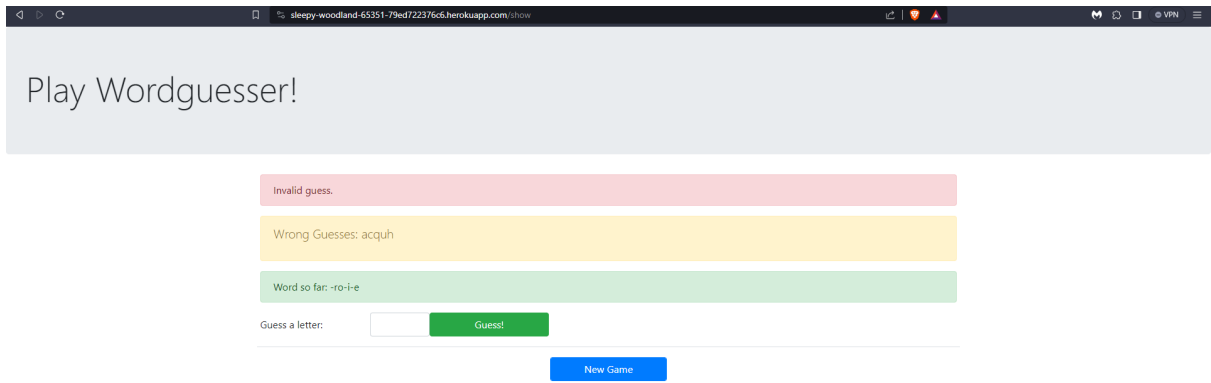
You have already used that letter

Wrong Guesses: acquh

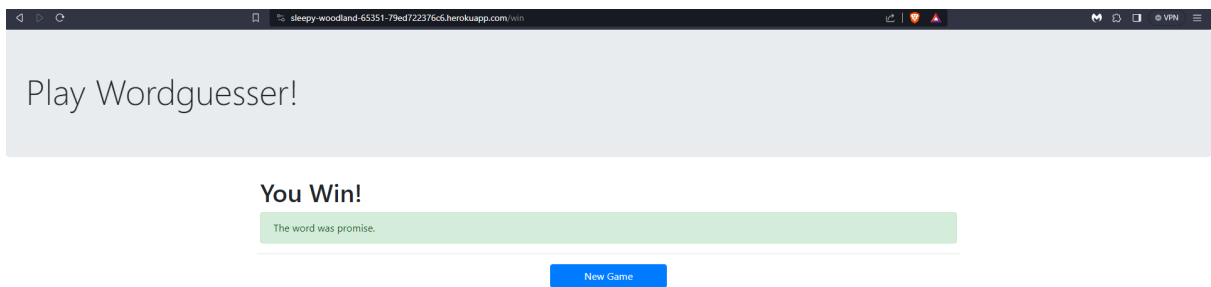
Word so far: -ro-i-e

Guess a letter:

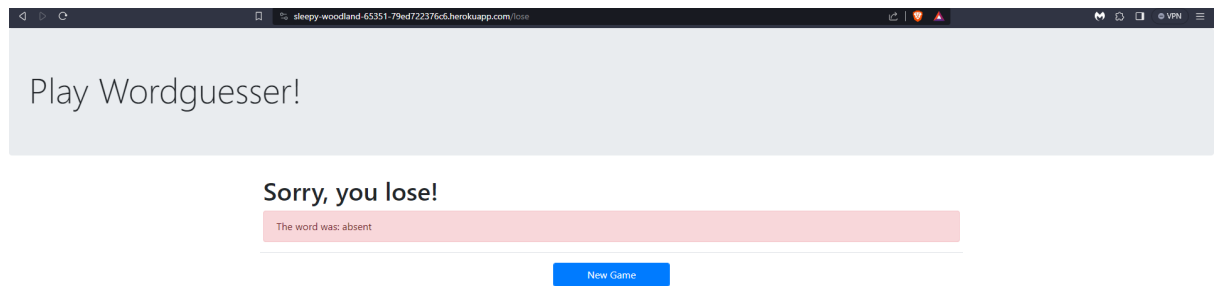
Ahora aparecen las letras que ya usamos.



También las respuestas que no son válidas.



La vista de /win



La vista de /lose

Como observamos nuestra aplicación funciona correctamente.

El URL para probarlo es <https://sleepy-woodland-65351-79ed722376c6.herokuapp.com>