

Primera Práctica Calificada

Integrantes:

- Acuña Napan Jaime Gonzalo
- Zuñiga Chicaña, Alejandra Aztirma

Creacion de aplicaciones SaaS

Objetivo:

El objetivo de este trabajo es comprender y documentar los pasos esenciales para crear, versionar e implementar una aplicación de Software como Servicio (SaaS). Además, se abordará la importancia de mantener la consistencia en los entornos de producción y desarrollo, asegurando que las bibliotecas y dependencias se gestionen adecuadamente.

Tareas a Realizar:

En este proyecto, se llevarán a cabo las siguientes tareas:

- Creación de una Aplicación "Hello World": Se desarrollará una aplicación de ejemplo utilizando el framework Sinatra.
- Versionamiento Correcto: Se aplicarán prácticas de versionamiento adecuadas para garantizar un control efectivo del código fuente de la aplicación. Esto incluye el uso de sistemas de control de versiones como Git.
- Implementación en Heroku: La aplicación desarrollada se implementará en la plataforma de alojamiento en la nube Heroku.

Creación y versionado de una aplicación SaaS sencilla

Para llevar a cabo esto realizamos los siguientes pasos:

- Creamos un nuevo directorio vacío destinado a alojar nuestra nueva aplicación y utilizamos el comando `git init` en ese directorio para iniciar el control de versiones mediante Git.
- Dentro de este directorio, creamos un archivo nuevo llamado `Gemfile` con el siguiente contenido:

```
source 'https://rubygems.org'
ruby '2.6.6'
gem 'sinatra', '>= 2.0.1'
```

Este archivo reconocerá las versiones de las gemas (bibliotecas) que utilizaremos en nuestra aplicación.

Luego, procedemos a la instalación de estas gemas ejecutando el comando `bundle install`. Esta acción instala automáticamente las gemas necesarias para el funcionamiento de nuestra aplicación.

```
PS C:\Users\jaime\OneDrive\Escritorio\Desarrollo de software\PracticaCalificada1_CC3S2\Production> bundle install
Fetching gem metadata from https://rubygems.org/....
Resolving dependencies...
Bundle complete! 1 Gemfile dependency, 7 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
```

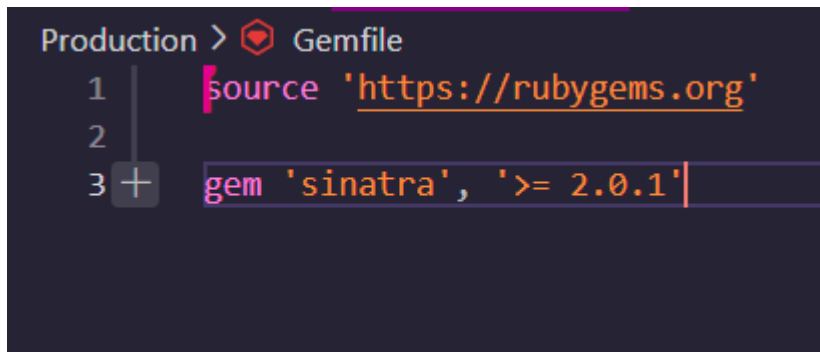
Para mantener un sistema de control de versiones para nuestra aplicación, utilizamos Git. Gracias a la configuración previa del Gemfile, podemos controlar las versiones de manera efectiva a través de Git.

```
PS C:\Users\jaime\OneDrive\Escritorio\Desarrollo de software\PracticaCalificada1_CC3S2\Production> git commit -am "Set up the Gemfile"
[main (root-commit) fc00610] Set up the Gemfile
4 files changed, 72 insertions(+)
create mode 100644 Production/Gemfile
create mode 100644 Production/Gemfile.lock
create mode 100644 README.md
create mode 100644 image.png
```

Preguntas

► Respuestas


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



```
Production > Gemfile
1 | source 'https://rubygems.org'
2 |
3 | + gem 'sinatra', '>= 2.0.1'
```

El archivo *Gemfile* que creamos alberga una lista de todas las

Por otro lado, *Gemfile.lock* registra no solo las versiones de estas gemas, sino también las versiones de otras dependencias requeridas por las gemas especificadas en *Gemfile*.

```
Production >  Gemfile.lock
1  GEM
2    remote: https://rubygems.org/
3    specs:
4      mustermann (3.0.0)
5        ruby2_keywords (~> 0.0.1)
6      rack (2.2.8)
7      rack-protection (3.1.0)
8        rack (~> 2.2, >= 2.2.4)
9      ruby2_keywords (0.0.5)
10     sinatra (3.1.0)
11       mustermann (~> 3.0)
12       rack (~> 2.2, >= 2.2.4)
13       rack-protection (= 3.1.0)
14       tilt (~> 2.0)
15     tilt (2.3.0)
16
17  PLATFORMS
18    x64-mingw-ucrt
19
20  DEPENDENCIES
21    sinatra (>= 2.0.1)
22
23  BUNDLED WITH
24    2.4.20
```

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

Para reproducir completamente las gemas del entorno de desarrollo en el entorno de producción, se necesita el archivo *Gemfile.lock*. Este archivo proporciona información detallada sobre las versiones exactas de las gemas y sus dependencias que deben instalarse para que la aplicación funcione correctamente en producción.

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

Al ejecutar el comando "bundle", Bundler examina las gemas especificadas en el archivo Gemfile. Por ejemplo, cuando se instala Sinatra, Bundler detecta que esta dependencia tiene requisitos adicionales, por lo que de manera recursiva instala todas las dependencias necesarias para satisfacer estos requisitos.

Crea una aplicación SaaS sencilla con Sinatra

En el desarrollo de aplicaciones SaaS, comenzaremos con Webrick para pruebas y luego usaremos Rack en producción. Sinatra, un marco ligero, nos permitirá definir cómo nuestra aplicación manejará solicitudes HTTP.

Paso 1: Creación del Archivo de la Aplicación

En un archivo llamado `app.rb`, escribimos el siguiente código:

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

Este código responde con "Hello World" cuando accedemos a la URL proporcionada.

Paso 2: Configuración del Archivo `config.ru`

Creamos un archivo llamado `config.ru` con el siguiente contenido:

```
require './app'
run MyApp
```

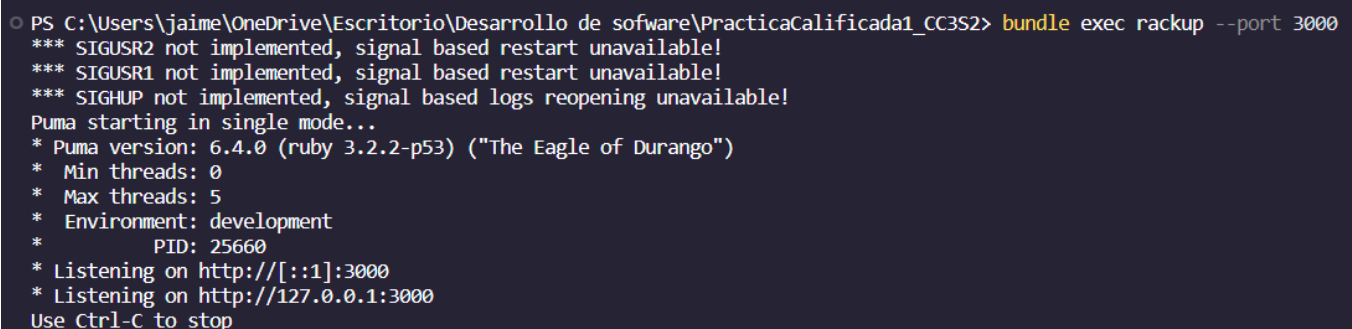
Esto le dice a Rack que nuestra aplicación se encuentra en `app.rb`.

Paso 3: Ejecución de la Aplicación

Finalmente, ejecutamos la aplicación con el siguiente comando:

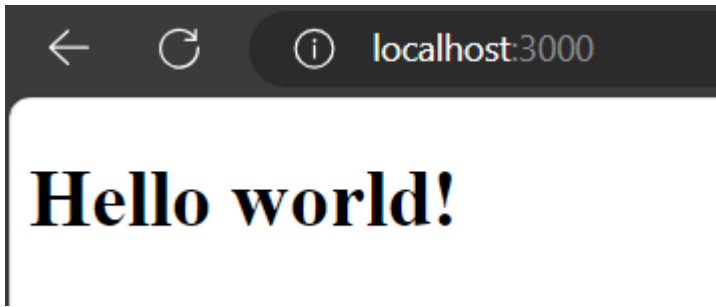
```
bundle exec rackup --port 3000
```

Después de haber completado los pasos anteriores, podremos verificar que nuestra aplicación está en funcionamiento en el puerto 3000, como se muestra en la siguiente imagen:



```
PS C:\Users\jaime\OneDrive\Escritorio\Desarrollo de software\PracticaCalificada1_CC3S2> bundle exec rackup --port 3000
*** SIGUSR2 not implemented, signal based restart unavailable!
*** SIGUSR1 not implemented, signal based restart unavailable!
*** SIGHUP not implemented, signal based logs reopening unavailable!
Puma starting in single mode...
* Puma version: 6.4.0 (ruby 3.2.2-p53) ("The Eagle of Durango")
* Min threads: 0
* Max threads: 5
* Environment: development
* PID: 25660
* Listening on http://[::1]:3000
* Listening on http://127.0.0.1:3000
Use Ctrl-C to stop
```

Como se menciona en el paso 2, al acceder a la ruta `http://localhost:3000/` veremos el mensaje Hello World, como se muestra a continuación:



Pregunta

¿Qué sucede si intentas visitar una URL no raíz cómo **https://localhost:3000/hello** y por qué?

► Respuesta



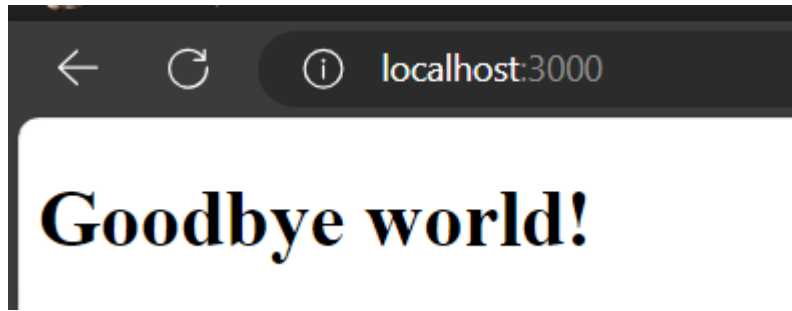
Modifica la aplicación

Para cambiar la salida de nuestra aplicación de "Hello world" a "Goodbye world", seguimos los siguientes pasos:

1. Detuvimos la aplicación actual mediante el comando **Ctrl-C**, como se muestra en la siguiente imagen:

```
- Goodbye!
Desea terminar el trabajo por lotes (S/N)? s
Desea terminar el trabajo por lotes (S/N)? s
PS C:\Users\jaime\OneDrive\Escritorio\Desarrollo de software\PracticaCalificada1_CC3S2>
```

2. Luego, modificamos el archivo `app.rb` para que la aplicación muestre el mensaje "Goodbye world".
3. Después, reiniciamos la aplicación utilizando el comando `bundle exec rackup --port 3000` para el desarrollo local.



Esto ilustra que al realizar modificaciones en nuestra aplicación mientras esta se encuentra en ejecución, debemos reiniciar Rack para que los cambios surtan efecto. Para automatizar este proceso, podemos emplear la gema `rerun`, que reinicia automáticamente Rack cuando detecta cambios en los archivos del directorio de la aplicación.

Continuando, añadimos la gema `rerun` a nuestro archivo Gemfile, como se muestra en la siguiente imagen:

```
1 source 'https://rubygems.org'
2
3 gem 'sinatra'
4 group :development do
5   gem 'rerun'
6 end
```

Con esto, ya no será necesario reiniciar manualmente el servidor cada vez que realicemos cambios en nuestra aplicación. La gema `rerun` se encargará de reiniciar automáticamente el servidor por nosotros.

Instalación de la Gema "rerun" y Ejecución de la Aplicación con ella

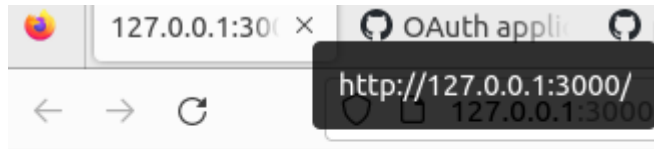
Para hacer uso de la gema `rerun` seguimos estos pasos:

1. Ejecutamos el comando `bundle install` para instalar la gema `rerun`.
2. Luego, ejecutamos la aplicación con la gema "rerun" utilizando el comando `bundle exec rerun rackup --port 3000`, como se ilustra en la imagen.

```
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ bundle exec rerun 'rackup -p 3000'
[DEPRECATED] `Bundler.with_clean_env` has been deprecated in favor of `Bundler.with_unbundled_env`.
If you instead want the environment before bundler was originally loaded, use `Bundler.with_original_env` (called at /var/lib/gems/3.0.0/gems/rerun-0.14.0/lib/rerun/notification.rb:74)

12:50:20 [rerun] Practicacalificada1_cc3s2 launched
12:50:20 [rerun] Rerun (23991) running Practicacalificada1_cc3s2 (24010)
Puma starting in single mode...
* Puma version: 6.4.0 (ruby 3.0.2-p107) ("The Eagle of Durango")
* Min threads: 0
* Max threads: 5
* Environment: development
* PID: 24010
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
```

3. Con esto, el puerto estará configurado de la siguiente manera:



Gooby wold!

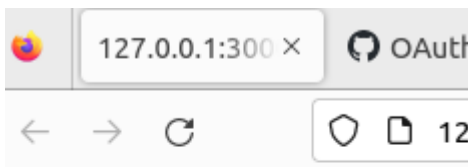
Detección Automática de Cambios y Reinicio Automático del Servidor

Ahora, si modificamos el mensaje HTML en nuestra aplicación a "Hello worl !" y guardamos los cambios, el servidor detectará automáticamente la modificación y se reiniciará por sí mismo, como se muestra aquí:

```
12:50:22 [rerun] Watching . for **/*.{rb,js,coffee,css,scss,sass,erb,html,haml,ru,yml,slim,md,featur
e,c,h} with Linux adapter
12:51:56 [rerun] Change detected: 1 modified: app.rb
- Gracefully stopping, waiting for requests to finish

12:51:56 [rerun] Practicacalificadal_cc3s2 restarted
12:51:56 [rerun] Rerun (23991) running Practicacalificadal_cc3s2 (24286)
Puma starting in single mode...
* Puma version: 6.4.0 (ruby 3.0.2-p107) ("The Eagle of Durango")
* Min threads: 0
* Max threads: 5
* Environment: development
* PID: 24286
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
```

Si observamos el navegador, veremos que el mensaje también ha cambiado:



Hello wold!

Implementar en Heroku

Heroku es una plataforma como servicio (PaaS) en la nube que nos permite implementar nuestras aplicaciones Sinatra (y más adelante, Rails). Procedemos a crear una cuenta en <http://www.heroku.com> para poder llevar a cabo esta implementación.

Paso 1: Instalamos Heroku CLI

Paso 2: Iniciamos Sesión en nuestra Cuenta Heroku Ejecutamos el comando `heroku login -i` en nuestra terminal. Esto nos solicita ingresar el correo y la contraseña de nuestra cuenta de Heroku.

```
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ heroku login -i
> Warning: heroku update available from 7.60.1 to 8.5.0.
heroku: Enter your login credentials
Email: jaime.acuna.n@uni.pe
Password: *****
Logged in as jaime.acuna.n@uni.pe
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$
```

Paso 3: Creamos una Nueva Aplicación Heroku

Utilizamos el comando `heroku create` para crear una nueva aplicación en Heroku.

```
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ heroku create
> Warning: heroku update available from 7.60.1 to 8.5.0.
Creating app... done, ● evening-escarpment-84783
https://evening-escarpment-84783-ab75b75c26c5.herokuapp.com/ | https://git.heroku.com/evening-escarpment-84783.git
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$
```

Paso 4: Creamos un Archivo Procfile

Creamos un archivo llamado "Procfile" en nuestro proyecto con las instrucciones necesarias para ejecutar nuestra aplicación en Heroku. Este archivo define el proceso web que Heroku debe ejecutar.



```
Procfile
1 web: bundle exec rackup config.ru -p $PORT
```



Paso 5: Subimos nuestro Repositorio a Heroku Usamos el comando `git push heroku master` para cargar nuestro repositorio en Heroku.

```
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ git push heroku main:main
Enumerando objetos: 52, listo.
Contando objetos: 100% (52/52), listo.
Compresión delta usando hasta 2 hilos
Comprimiendo objetos: 100% (45/45), listo.
Escribiendo objetos: 100% (52/52), 342.37 KiB | 26.34 MiB/s, listo.
Total 52 (delta 14), reusados 22 (delta 3), pack-reusados 0
remote: Updated 21 paths from 97609de
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-3.1.4
remote: -----> Installing dependencies using bundler 2.3.25
remote: Running: BUNDLE_WITHOUT='development:test' BUNDLE_PATH=vendor/bundle BUNDLE_B
IN=vendor/bundle/bin BUNDLE_DEPLOYMENT=1 bundle install -j4
remote: Fetching gem metadata from https://rubygems.org/.....
remote: Using bundler 2.3.26
remote: Using ruby2_keywords 0.0.5
remote: Fetching nio4r 2.5.9
```

Paso 6: Verificamos la Implementación en Heroku



Podemos verificar la ejecución de nuestra aplicación desde la página de Heroku.



 Personal >  evening-escarpment-84783



GitHub  peg1163/PracticaCalificada1_CC352 



[Overview](#) [Resources](#) [Deploy](#) [Metrics](#) [Activity](#) [Access](#) [Settings](#)



Activity Feed



  **alejandra.zuniga.c@uni.pe:** Deployed 544fb256
Today at 11:50 PM · v6 · [Compare diff](#)

  **alejandra.zuniga.c@uni.pe:** Build succeeded
Today at 11:50 PM · [View build log](#)

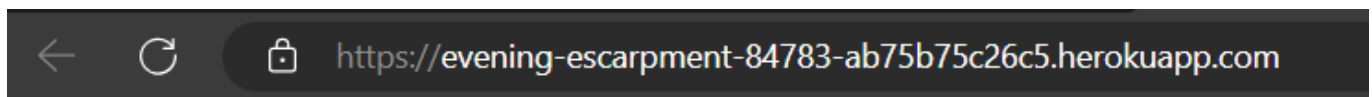
  **jaime.acuna.n@uni.pe:** Deployed 544fb256
Today at 11:33 PM · v5 · [Roll back to here](#) · [Compare diff](#)

  **jaime.acuna.n@uni.pe:** Build succeeded
Today at 11:33 PM · [View build log](#)

  **jaime.acuna.n@uni.pe:** Deployed 439c9fc4
Today at 11:17 PM · v4 · [Roll back to here](#)

  **jaime.acuna.n@uni.pe:** Set LANG, RACK_ENV config vars
Today at 11:17 PM · v3 · [Roll back to here](#)

Si todo se ejecuta correctamente, vemos nuestra aplicación en funcionamiento en la plataforma Heroku.



Hello wold!

Este proceso nos permite llevar nuestra aplicación Sinatra a la nube y hacerla accesible en línea a través de Heroku.

Parte 1: Wordguesser

Con todos estos pasos en mente, procedamos a clonar este repositorio y a trabajar en el juego de adivinanza de palabras, conocido como Wordguesser.

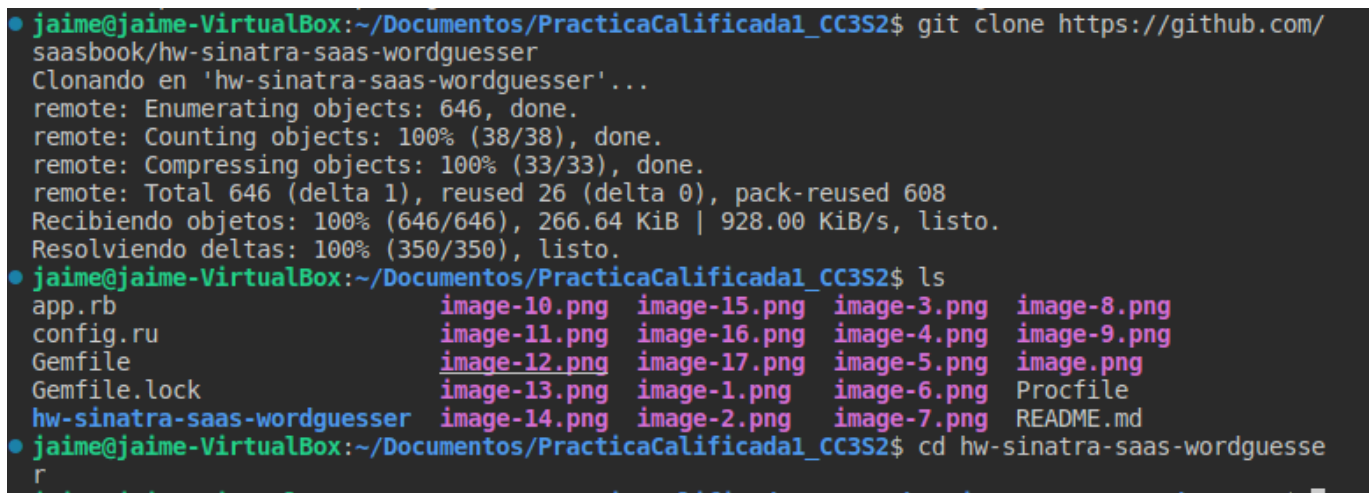
Para llevar a cabo esto, clonamos el repositorio de Wordguesser utilizando el siguiente comando:

```
git clone https://github.com/saasbook/hw-sinatra-saas-wordguesser
```

Luego, ingresamos al repositorio clonado para continuar con nuestro trabajo.

```
cd hw-sinatra-saas-wordguesser
```

A continuación, se presenta una vista de lo que realizamos:



```
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ git clone https://github.com/saasbook/hw-sinatra-saas-wordguesser
Clonando en 'hw-sinatra-saas-wordguesser'...
remote: Enumerating objects: 646, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 646 (delta 1), reused 26 (delta 0), pack-reused 608
Recibiendo objetos: 100% (646/646), 266.64 KiB | 928.00 KiB/s, listo.
Resolviendo deltas: 100% (350/350), listo.
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ ls
app.rb          image-10.png  image-15.png  image-3.png   image-8.png
config.ru       image-11.png  image-16.png  image-4.png   image-9.png
Gemfile         image-12.png  image-17.png  image-5.png   image.png
Gemfile.lock    image-13.png  image-1.png   image-6.png   Procfile
hw-sinatra-saas-wordguesser image-14.png  image-2.png   image-7.png   README.md
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2$ cd hw-sinatra-saas-wordguesser
```

Desarrollo de Wordguesser usando TDD y Guard

En esta etapa del desarrollo, nos centraremos en implementar la lógica del juego Wordguesser utilizando la metodología de Desarrollo Basado en Pruebas (TDD).

A diferencia de lo que se muestra en la aplicación, optamos por utilizar Guard en lugar de Autotest para ejecutar las pruebas. Esta decisión se basó en ciertos problemas que enfrentamos con las gemas desactualizadas en el proyecto clonado, ya que este requería una versión específica de Ruby, que en este caso era Ruby 2.6.6. Para gestionar esta versión de Ruby, utilizamos RVM (Ruby Version Manager). Esto nos permitió asegurarnos de que estábamos trabajando con la versión exacta de Ruby que el proyecto requería. Sin embargo, dado que Autotest no funcionaba correctamente debido a las discrepancias en las gemas y las versiones de Ruby, decidimos optar por Guard para la ejecución de las pruebas. Guard nos proporcionó una solución más efectiva para administrar las pruebas y asegurarnos de que funcionaran sin problemas, incluso con nuestras gemas actualizadas y la versión específica de Ruby.

Una vez configurado nuestro entorno y ejecutado el guard, nos dimos cuenta de que había un conjunto de 18 pruebas pendientes:

```

18) WordGuesserGame game status should continue play if neither win nor lose
# No reason given
Failure/Error: game.guess(letter)

NoMethodError:
  undefined method `guess' for #<WordGuesserGame:0x0000564c0e6a0c60 @word="dog">
# ./spec/wordguesser_game_spec.rb:8:in `block in guess_several_letters'
# ./spec/wordguesser_game_spec.rb:7:in `chars'
# ./spec/wordguesser_game_spec.rb:7:in `guess_several_letters'
# ./spec/wordguesser_game_spec.rb:120:in `block (3 levels) in <top (required)>'
# /var/lib/gems/3.0.0/gems/webmock-3.19.1/lib/webmock/rspec.rb:39:in `block (2 levels) in <top (required)>'

Finished in 0.07899 seconds (files took 0.30995 seconds to load)
18 examples, 0 failures, 18 pending

[1] guard(main)> 

```

Comenzaremos eliminando pending => true (estado pendiente) de la primera prueba y guardamos el archivo. Esto hará que Guard ejecute automáticamente las pruebas relacionadas.

```

describe 'new', :pending => true 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

```

Como mencionamos eliminamos pending => true (estado "pendiente"), de este modo esta prueba dejará de estar pendiente, pero aún se observa que la prueba fallará, esto debió a que aún no hemos implementado el método al cual la prueba hace referencia.

```

Python RES:

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:0x0000563265c18f98 @word="glorp">
# ./spec/wordguesser_game_spec.rb:16:in `block (3 levels) in <top (required)>'
# /var/lib/gems/3.0.0/gems/webmock-3.19.1/lib/webmock/rspec.rb:39:in `block (2 levels) in <top (required)>'

Finished in 0.06685 seconds (files took 0.37313 seconds to load)
18 examples, 1 failure, 17 pending

```

Pregunta

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`?

```

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('')
  end
end

```

```
expect(@game.wrong_guesses).to eq('')  
end  
end
```

► Respuesta

El constructor de la clase `WordGuesserGame` espera un argumento, que es la palabra a adivinar. Entonces, la primera línea de la definición del método `initialize` en el archivo `wordguesser_game.rb` debería ser:

```
def initialize(word_to_guess)
```

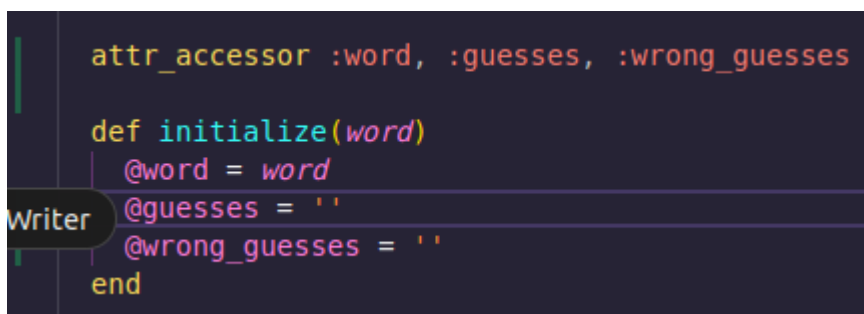
Esto significa que el constructor tomará un único argumento llamado `word_to_guess`, que será la palabra que se va a adivinar.

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

► Respuesta

Según las pruebas en el bloque `describe`, se espera que la clase `WordGuesserGame` tenga tres variables de instancia. La primera, `@game.word`, debería contener la palabra a adivinar. La segunda, `@game.guesses`, se llenará con las letras adivinadas correctamente y estará vacía al inicio. La tercera, `@game.wrong_guesses`, se utilizará para almacenar letras adivinadas incorrectamente y también comenzará vacía.

Continuando con la actividad, implementamos la clase `WordGuesserGame` con un constructor que acepte los argumentos necesarios según las pruebas que tenemos. Además nos aseguramos de que estas variables actúen como getters y setters para que podamos acceder y modificar sus valores según sea necesario para pasar las pruebas.



```
attr_accessor :word, :guesses, :wrong_guesses  
  
def initialize(word)  
  @word = word  
  @guesses = ''  
  @wrong_guesses = ''  
end
```

Anteriormente, eliminamos `pending => true` de la primera prueba, como se puede observar, ya que planeamos trabajar en esta prueba una vez que hayamos implementado completamente la clase `WordGuesserGame`.

```

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

```

Cuando guardamos los cambios, el sistema de guardado detecta la modificación y nos muestra el siguiente mensaje:

```

# /var/lib/gems/3.0.0/gems/webmock-3.19.1/lib/webmock/rspec.rb:3
Finished in 0.05982 seconds (files took 0.24036 seconds to load)
18 examples, 0 failures, 17 pending

```

Luego, continuamos eliminando `:pending => true` del resto de las pruebas: `guess`, `check_win_or_lose` y `word_with_guesses`.

Siguiendo las pruebas y los casos propuestos, a continuación, se presenta la prueba "guessing":

```

describe 'guessing' do
  context 'correctly' do
    before :each do
      @game = WordGuesserGame.new('garply')
      @valid = @game.guess('a')
    end
    it 'changes correct guess list' do
      expect(@game.guesses).to eq('a')
      expect(@game.wrong_guesses).to eq('')
    end
    it 'returns true' do
      expect(@valid).not_to be false
    end
  end
  context 'incorrectly' do
    before :each do
      @game = WordGuesserGame.new('garply')
      @valid = @game.guess('z')
    end
    it 'changes wrong guess list' do
      expect(@game.guesses).to eq('')
      expect(@game.wrong_guesses).to eq('z')
    end
    it 'returns true' do
      expect(@valid).not_to be false
    end
  end
end

```

```

context 'same letter repeatedly' do
  before :each do
    @game = WordGuesserGame.new('garply')
    guess_several_letters(@game, 'aq')
  end
  it 'does not change correct guess list' do
    @game.guess('a')
    expect(@game.guesses).to eq('a')
  end
  it 'does not change wrong guess list' do
    @game.guess('q')
    expect(@game.wrong_guesses).to eq('q')
  end
end

are_returns_false' do
  expect(@game.guess('a')).to be false
  expect(@game.guess('q')).to be false
end

it 'is case insensitive' do
  expect(@game.guess('A')).to be false
  expect(@game.guess('Q')).to be false
  expect(@game.guesses).not_to include('A')
  expect(@game.wrong_guesses).not_to include('Q')
end
end

context 'invalid' do
  before :each do
    @game = WordGuesserGame.new('foobar')
  end
  it 'throws an error when empty' do
    expect { @game.guess('') }.to raise_error(ArgumentError)
  end
  it 'throws an error when not a letter' do
    expect { @game.guess('%') }.to raise_error(ArgumentError)
  end
  it 'throws an error when nil' do
    expect { @game.guess(nil) }.to raise_error(ArgumentError)
  end
end
end

```

Para esta prueba se implementa el método guess de la clase `WordGuesserGame`

```

def guess(letter)
  #descartamos algunas posibilidades
  raise ArgumentError if letter.nil? || letter.empty? || letter =~ /^[^a-zA-Z]/
  #convertimos a minúscula
  letter.downcase!
  #comprobamos si la letra esta en la palabra
  if @word.include?(letter)
    #si esta, comprobamos si ya estaba en la lista de aciertos
    return false if @guesses.include?(letter)
    #si no estaba, la añadimos
    @guesses += letter
  #si la letra no esta, comprobamos si ya estaba en la lista de fallos
  else
    #si ya estaba, devolvemos false
    return false if @wrong_guesses.include?(letter)
    #si no estaba, la añadimos
    @wrong_guesses += letter
  end
  #devolvemos
  return true
end

```

Analogamente para el resto pruebas `check_win_or_lose` y `word_with_guesses`

```

describe 'displayed word with guesses' do
  before :each do
    @game = WordGuesserGame.new('banana')
  end
  # for a given set of guesses, what should the word look like?
  @test_cases = {
    'bn' => 'b-n-n-',
    'def' => '-----',
    'ban' => 'banana'
  }
  @test_cases.each_pair do |guesses, displayed|
    it "should be '#{displayed}' when guesses are '#{guesses}'" do
      guess_several_letters(@game, guesses)
      expect(@game.word_with_guesses).to eq(displayed)
    end
  end
end

describe 'game status' do
  before :each do
    @game = WordGuesserGame.new('dog')
  end
  it 'should be win when all letters guessed' do
    guess_several_letters(@game, 'ogd')
    expect(@game.check_win_or_lose).to eq(:win)
  end
  it 'should be lose after 7 incorrect guesses' do
    guess_several_letters(@game, 'tuvwxyz')
    expect(@game.check_win_or_lose).to eq(:lose)
  end
  it 'should continue play if neither win nor lose' do
    guess_several_letters(@game, 'do')
    expect(@game.check_win_or_lose).to eq(:play)
  end
end
end

```

Se definio para prueba, su método respectivo.

```

def word_with_guesses
  # Inicializamos una variable para almacenar la representación de la palabra
  wordguesses = ''
  # Iteramos a través de cada letra en la palabra original
  @word.each_char do |letter|
    if @guesses.include?(letter)
      # Si la letra actual está en la lista de letras adivinadas (en @guesses)
      # la agregamos a la representación de la palabra
      wordguesses += letter
    else
      # Si la letra actual no está en la lista de letras adivinadas, la reemplazamos con un guion ("-")
      wordguesses += '-'
    end
  end
  # Devolvemos la representación de la palabra con letras adivinadas y guiones
  return wordguesses
end

def check_win_or_lose
  # Comprobamos si la palabra con letras adivinadas es igual a la palabra original
  # Si es igual, el jugador ha adivinado todas las letras y ha ganado
  return :win if self.word_with_guesses == @word

  # Comprobamos si el número de fallos es mayor o igual a 7
  # Si el jugador ha realizado 7 o más intentos incorrectos, pierde el juego
  return :lose if @wrong_guesses.length >= 7

  # Si no se cumple ninguna de las condiciones anteriores, el juego todavía está en curso
  # En este caso, devolvemos :play para indicar que el juego sigue en marcha
  return :play
end

```

Al guardar y actualizar las pruebas, hemos comprobado que cumplen con todos los requisitos necesarios:


```
Finished in 0.06272 seconds (files took 0.83741 seconds to load)
18 examples, 0 failures

[1] guard(main)> █
```

Este proceso de desarrollo y pruebas asegura que nuestro código cumpla con los estándares y funcione correctamente.

Parte 2: RESTful para Wordguesser

Identificación del estado mínimo del juego

Antes de diseñar la aplicación, es importante identificar cuál es el estado mínimo del juego que debemos mantener para que los usuarios puedan jugar de manera efectiva. Este estado mínimo es esencial para preservar el progreso del juego entre las solicitudes HTTP.

Pregunta 1: Enumera el estado mínimo del juego que se debe mantener durante una partida de Wordguesser.

El estado mínimo del juego que debe mantenerse incluye:

- La palabra oculta que el jugador está tratando de adivinar.
- Las letras adivinadas correctamente.
- Las letras adivinadas incorrectamente.
- El número de intentos restantes.

Identificación de las acciones del jugador

Para diseñar una aplicación interactiva como Wordguesser, es crucial identificar las acciones que los jugadores pueden realizar y que tienen un impacto en el estado del juego. Estas acciones definirán las rutas RESTful de la aplicación.

Pregunta 2: Enumera las acciones del jugador que podrían provocar cambios en el estado del juego.

Las acciones del jugador que pueden cambiar el estado del juego incluyen:

- Adivinar una letra.
- Comenzar un nuevo juego.

Asignación de métodos HTTP en un diseño RESTful

En el diseño de una aplicación RESTful, es fundamental asignar correctamente los métodos HTTP a las operaciones de recursos. Esto garantiza que las solicitudes se manejen de manera coherente y segura.

Pregunta 3: 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?

En un diseño RESTful, las operaciones suelen manejarse de la siguiente manera:

- **GET** se utiliza para acciones que no modifican el estado del servidor y que pueden mostrarse en una página web, como mostrar el estado actual del juego o las páginas de victoria/derrota.
- **POST** se utiliza para acciones que modifican el estado del servidor, como enviar una adivinanza o iniciar un nuevo juego.

Uso de **GET** en la acción "nueva"

En el diseño RESTful, es importante asignar el método HTTP correcto para cada acción. La pregunta se centra en la elección de **GET** para la acción "nueva" y su justificación.

Pregunta 4: ¿Por qué es apropiado que la nueva acción utilice GET en lugar de POST?

La acción "nueva" utiliza el método **GET** porque no modifica el estado del servidor. Simplemente muestra un formulario para que el usuario humano inicie un nuevo juego, y esta acción no tiene un impacto directo en el servidor.

Acción **GET /new** en una arquitectura orientada a servicios

Se explora la necesidad de la acción **GET /new** en el contexto de una arquitectura orientada a servicios frente a una aplicación web tradicional.

Pregunta 5: 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.

En una arquitectura orientada a servicios, los servicios web generalmente no necesitan una acción **GET /new** porque las interacciones se realizan programáticamente a través de API y no a través de interfaces web humanas. La acción **GET /new** es específica de una interfaz web para permitir que los usuarios inicien juegos manualmente.

Parte 3 :Conexión de WordGuesserGame a Sinatra

Con una comprensión más clara de los conceptos, vamos a conectar la aplicación con Sinatra.

Pregunta

En este contexto, ¿de qué clase es la variable de instancia @game?

En el contexto proporcionado en la aplicación Sinatra, **@game** es una variable de instancia de la clase WordGuesserGame. Se utiliza para mantener el estado del juego a lo largo de las solicitudes HTTP.

Pregunta

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

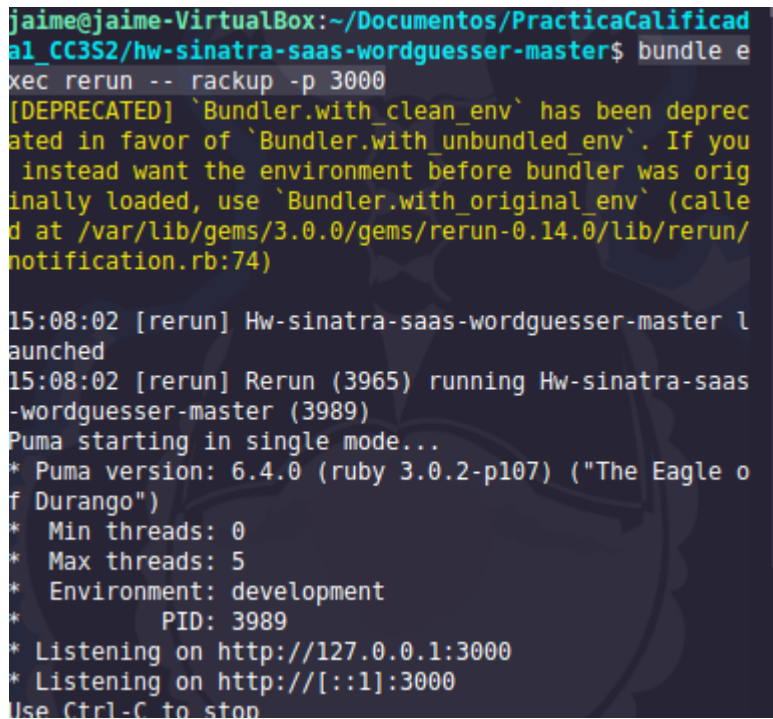
Usar `flash[]` ahorra trabajo porque elimina la necesidad de borrar manualmente los mensajes temporales de la sesión. Los mensajes flash se borran automáticamente después de su uso, lo que significa que los desarrolladores no tienen que preocuparse de mantener la sesión limpia.

Ejecutando la aplicación Sinatra

Anteriormente, pudimos lanzar una aplicación sencilla con un "Hello World" en un puerto local. Ahora, vamos a levantar la aplicación Wordguesser en un puerto local con la ayuda de Sinatra. Ejecutamos el siguiente comando:

```
bundle exec rerun -- rackup -port 3000
```

La siguiente imagen muestra el resultado de ejecutar el comando `bundle exec rerun -- rackup -port 3000` en la terminal. Este comando levanta una aplicación Sinatra en el puerto 3000.



```
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificada1_CC3S2/hw-sinatra-saas-wordguesser-master$ bundle e
xec rerun -- rackup -p 3000
[DEPRECATED] `Bundler.with_clean_env` has been deprec
ated in favor of `Bundler.with_unbundled_env`. If you
instead want the environment before bundler was orig
inally loaded, use `Bundler.with_original_env` (calle
d at /var/lib/gems/3.0.0/gems/rerun-0.14.0/lib/rerun/
notification.rb:74)

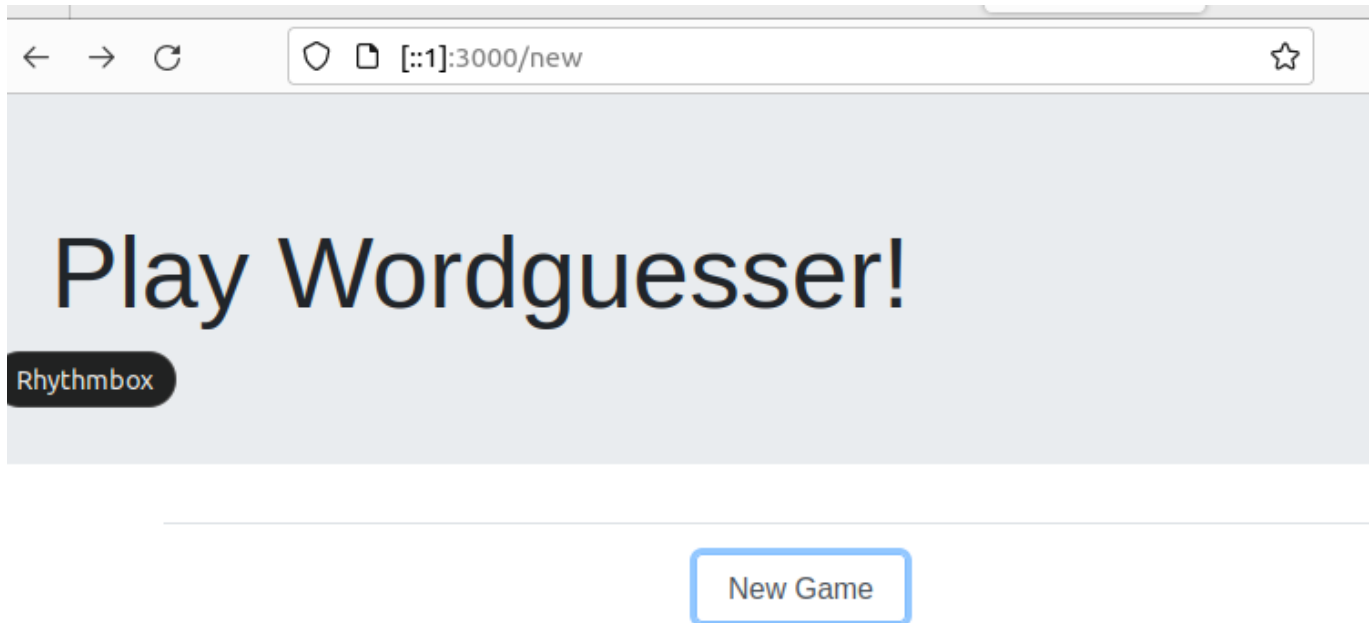
15:08:02 [rerun] Hw-sinatra-saas-wordguesser-master l
aunched
15:08:02 [rerun] Rerun (3965) running Hw-sinatra-saas
-wordguesser-master (3989)
Puma starting in single mode...
* Puma version: 6.4.0 (ruby 3.0.2-p107) ("The Eagle o
f Durango")
* Min threads: 0
* Max threads: 5
* Environment: development
* PID: 3989
* Listening on http://127.0.0.1:3000
* Listening on http://[::]:3000
Use Ctrl-C to stop
```

Pregunta

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

Al utilizar Sinatra, la aplicación se despliega en nuestro puerto local 3000. Para acceder a la página "New Game", simplemente debemos agregar `/new` a la URL del puerto local, lo que resulta en la siguiente dirección: `http://localhost:3000/new`, tal como se muestra en la imagen anterior.

Visitamos la URL proporcionada y confirmamos que la página "Iniciar New Game" se muestra correctamente:



¿Dónde está el código HTML de esta página?

El código HTML de esta página se encuentra en el archivo `new.erb`. Cuando revisamos el código en `app.rb`, notamos que al ejecutar la ruta `"/new"`, esta ruta renderiza el archivo `"new.erb"`, que contiene el código HTML correspondiente.

Al hacer clic en el botón "New Game", se generará un error intencionado debido a que al examinar el HTML en el archivo `"new.erb"`, se puede notar que el elemento `form` está incompleto, lo que ocasiona el error al hacer clic en "New Game".

```
hw-sinatra-saas-wordguesser-master > views > new.erb
1  <!-- This form is incomplete--it needs a destination URL as well as a
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
5    </div>
6  </form>
7
```

Este error es evidente cuando se intenta iniciar un nuevo juego, como se muestra en la siguiente imagen:



[::1]:3000/new



Sinatra doesn't know this ditty.

[Ayuda](#)

Try this:

```
# in app.rb
class WordGuesserApp
  post '/new' do
    "Hello World"
  end
end
```

Despliegue en Heroku :

Durante el proceso de implementación de nuestra aplicación en Heroku, nos encontramos con una dificultad inesperada. A pesar de que la aplicación funcionaba correctamente en nuestro entorno de desarrollo local, al intentar desplegarla en Heroku, nos encontramos con un error que resultó ser confuso y desafiante. Para investigar el problema con más detalle, utilizamos el comando `heroku --tail` para revisar los registros de la aplicación en Heroku, pero lamentablemente, no proporcionaron información esclarecedora.

Después de realizar búsquedas en línea y consultar la documentación oficial de Heroku, seguimos varias soluciones recomendadas, pero ninguna de ellas logró resolver el problema que estábamos enfrentando. Ante la falta de avance y el tiempo limitado disponible, tomamos la decisión de continuar trabajando en el desarrollo de la aplicación en nuestro entorno de desarrollo local, posponiendo temporalmente el intento de desplegarla en Heroku.

Aunque esta medida nos permitió avanzar con el proyecto y cumplir con los plazos establecidos, lamentablemente no pudimos aprovechar las ventajas de la plataforma de despliegue en la nube de Heroku en esta ocasión.



A continuación se adjunto las capturas de los errores que nos mostraba tanto en el terminal como en heroku.





Application error

An error occurred in the application and your page could not be served. If you are the application owner, [check your logs for details](#). You can do this from the Heroku CLI with the command

```
heroku logs --tail
```

 Personal >  evening-escarpment-84783

GitHub  peg1163/PracticaCalificadal_CC3S2  main

Overview Resources Deploy Metrics **Activity** Access Settings

Activity Feed > Build Log

Please upgrade your Ruby version.

For a list of supported Ruby versions see:

<https://devcenter.heroku.com/articles/ruby-support#supported-runtimes>

-----> Discovering process types

Procfile declares types -> web

Default types for buildpack -> console, rake

-----> Compressing...

Done: 21M

-----> Launching...

Released v63

<https://evening-escarpment-84783-ab75b75c26c5.herokuapp.com/> deployed to Heroku

This app is using the Heroku-20 stack, however a newer stack is available.

To upgrade to Heroku-22, see:

<https://devcenter.heroku.com/articles/upgrading-to-the-latest-stack>

Build finished

```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

2023-10-06T21:09:53.941927+00:00 heroku[web.1]: State changed from crashed to starting
2023-10-06T21:09:54.000000+00:00 app[api]: Build succeeded
2023-10-06T21:09:56.180003+00:00 heroku[web.1]: Starting process with command `bundle exec rackup c
onfig.ru -p 51953`
2023-10-06T21:09:58.013014+00:00 app[web.1]: bundler: failed to load command: rackup (/app/vendor/b
undle/ruby/3.0.0/bin/rackup)
2023-10-06T21:09:58.013047+00:00 app[web.1]: /app/vendor/bundle/ruby/3.0.0/gems/rack-2.2.8/lib/rack
/handler.rb:45:in `pick': Couldn't find handler for: puma, thin, falcon, webrick. (LoadError)
2023-10-06T21:09:58.013048+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/rack-2.2.8/lib
/rack/handler.rb:60:in `default'
2023-10-06T21:09:58.013048+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/rack-2.2.8/lib
/rack/server.rb:334:in `server'
2023-10-06T21:09:58.013049+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/rack-2.2.8/lib
/rack/server.rb:327:in `start'
2023-10-06T21:09:58.013049+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/rack-2.2.8/lib
/rack/server.rb:168:in `start'
2023-10-06T21:09:58.013049+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/rack-2.2.8/bin
/rackup:5:in `'
2023-10-06T21:09:58.013049+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/bin/rackup:23:in `l
oad'
2023-10-06T21:09:58.013050+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/bin/rackup:23:in `<
top (required)>'
2023-10-06T21:09:58.013050+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/bundler-2.3.25
/lib/bundler/cli/exec.rb:58:in `load'
2023-10-06T21:09:58.013051+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/bundler-2.3.25
/lib/bundler/cli/exec.rb:58:in `kernel_load'
2023-10-06T21:09:58.013051+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/bundler-2.3.25
/lib/bundler/cli/exec.rb:23:in `run'
2023-10-06T21:09:58.013051+00:00 app[web.1]: from /app/vendor/bundle/ruby/3.0.0/gems/bundler-2.3.25
/lib/bundler/cli.rb:486:in `exec'

aztirma@aztirma-VirtualBox:~/Documentos/GitHub/PracticaCalificada1_CC3S2$ bundle exec rackup config
.ru -p 51953
Puma starting in single mode...
* Puma version: 6.4.0 (ruby 3.0.2-p107) ("The Eagle of Durango")
* Min threads: 0
* Max threads: 5
* Environment: development
* PID: 29207
* Listening on http://127.0.0.1:51953
* Listening on http://[::]:51953
Use Ctrl-C to stop
127.0.0.1 - - [06/Oct/2023:16:16:23 -0500] "GET / HTTP/1.1" 302 - 0.0334
127.0.0.1 - - [06/Oct/2023:16:16:23 -0500] "GET /new HTTP/1.1" 200 949 0.0054
127.0.0.1 - - [06/Oct/2023:16:16:24 -0500] "GET /favicon.ico HTTP/1.1" 404 513 0.0018

* Historial restaurado

aztirma@aztirma-VirtualBox:~/Documentos/GitHub/PracticaCalificada1_CC3S2$

```

Parte 4 : Cucumber

En nuestro proceso de desarrollo de software, utilizamos Cucumber para escribir pruebas de aceptación e integración de alto nivel en lenguaje natural, almacenadas en archivos .feature. Además, combinamos Cucumber con RSpec para impulsar el desarrollo de código. Para simular el comportamiento de un navegador y realizar pruebas de integración, utilizamos Capybara, que simula el servidor como lo haría un navegador y nos permite inspeccionar las respuestas de la aplicación a las acciones del usuario.

Una vez leída la sección sobre "Using Capybara with Cucumber" en la página de inicio de Capybara, procedemos a responder las siguientes preguntas:

Preguntas

¿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?

Para simular el servidor como lo haría un navegador utilizando Capybara en Cucumber, Capybara sigue estos pasos:

En primer lugar, se configura Capybara en el proyecto de Cucumber, como se muestra en el ejemplo proporcionado:

```
require 'capybara/cucumber'
Capybara.app = MyRackApp
```

Luego, escribes los escenarios de prueba utilizando el DSL de Capybara en los pasos de Cucumber. Capybara interactuará con la aplicación web como lo haría un navegador. En el ejemplo proporcionado

```
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
```

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
```

Estas tres líneas que siguen al encabezado "Feature:" en un archivo de características de Cucumber funcionan como un comentario descriptivo o título de la característica. Este comentario brinda una breve descripción de alto nivel de lo que se está evaluando o probando en esa característica específica.

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?

En el escenario "Given I start a new game with word "garply"", las líneas en `game_steps.rb` que se invocarán son las siguientes:

```
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
```

El papel de la cadena "garply" en este paso es actuar como un argumento que se pasa al paso para personalizar la solicitud HTTP simulada y el inicio del juego.

Haz que pase tu primer escenario

Pregunta 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?

La razón por la que visit siempre hace una solicitud GET en Capybara es porque Capybara simula el comportamiento de un navegador web. Cuando los usuarios navegan por un sitio web, generalmente hacen clic en enlaces o escriben URL en la barra de direcciones, lo que resulta en solicitudes GET. Las solicitudes POST están más relacionadas con el envío de formularios, una acción más específica y deliberada.

Para que se cumpla el primer escenario, debemos ejecutar el comando `cucumber` `features/start_new_game.feature`. Al hacerlo, notamos que la prueba no se aprueba debido a la necesidad de completar ciertos atributos en el formulario

```
user-master$ 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 `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)
0m0.145s
```

Para resolver esto, agregamos un atributo adicional llamado "action". Este atributo especifica la acción que el usuario desea realizar.

```
hw-sinatra-saas-wordguesser-master > views > new.erb
1 <!-- This form is incomplete--it needs a destination URL as well as a
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
5   </div>
6 </form>
7
```

Luego, ejecutamos el comando de prueba:


```
cucumber features/start_new_game.feature
```

Como se muestra a continuación, el cual nos da la siguiente salida que significa que nuestra prueba al implementar el atributo, paso.

```
sser-
master$ 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)
0m0.098s
```

Pregunta

¿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.

Los pasos "Given", "When" y "Then" tienen significados específicos:

"Given" se utiliza para establecer el contexto inicial.

"When" se utiliza para describir la acción que se realiza.

"Then" se utiliza para describir las expectativas o resultados esperados después de la acción.

Cambiar estos pasos afectará el flujo de la prueba y cómo se comunica el comportamiento esperado de la aplicación.

Desarrollar el escenario para adivinar una letra

En esta etapa de desarrollo de nuestra aplicación, nos centraremos en el escenario en el que el usuario intenta adivinar una letra en un juego de ahorcado. Para llevar a cabo esta funcionalidad, utilizaremos el archivo `guess.feature`.

```
cucumber features/guess.feature:7 # Scenario: gu
orrect letter that occurs once
cucumber features/guess.feature:13 # Scenario: g
correct letter that occurs multiple times
cucumber features/guess.feature:19 # Scenario: g
incorrect letter
cucumber features/guess.feature:25 # Scenario: m
le correct and incorrect guesses

4 scenarios (4 failed)
13 steps (4 failed, 2 skipped, 7 passed)
0m0.297s
```

En la imagen podemos observar que las pruebas no pasaron debido a que aún no habíamos realizado modificaciones en la sección de "guess" en el archivo app.rb. Para solucionar esto, hemos realizado las siguientes modificaciones en el código:

```
# If a guess is invalid, set flash[:message] to "Invalid guess."
post '/guess' do
  begin
    #obtiene la primera letra de la palabra
    letter = params[:guess].to_s[0]
    #almacena las letras que se han adivinado
    initial_guesses = @game.guesses
    #revisa si la letra ha sido utilizada
    if !@game.guess(letter)

      flash[:message] = "You have already used that letter."
      #revisa si la letra es invalida
    elsif initial_guesses == @game.guesses
      flash[:message] = "Invalid guess."
    end
    #captura el error para una suposicion invalida
  rescue ArgumentError
    flash[:message] = "Invalid guess."
  end
  #redirecciona a la pagina de show
  redirect '/show'
end
```

Después de realizar estas modificaciones, procedimos a ejecutar nuevamente las pruebas:

```
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)
0m0.536s
```

Ahora, con las modificaciones realizadas en `app.rb`, observamos que las pruebas han tenido éxito y han pasado correctamente.

Pregunta

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).

```
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
```

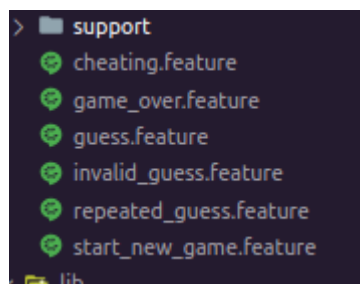
La línea de código mencionada, utiliza la gema "webmock" para simular una solicitud HTTP POST a una URL que proporciona una palabra aleatoria. La simulación establece que la respuesta de la solicitud será la palabra que especificamos en el paso de prueba, en este caso **word**. Esto nos ayuda a hacer que nuestras pruebas sean más predecibles y controladas.

Pregunta

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?

Parte 5: Otros casos

Ahora , en el archivo feature vemos que hay mas pruebas que podemos ejecutar



y así poder completar :

```
# Notice that the show.erb template expects to use  
# wrong_guesses and word_with_guesses from @game.  
get '/show' do  
  ### YOUR CODE HERE ###  
  erb :show # You may change/remove this line  
end  
  
get '/win' do  
  ### YOUR CODE HERE ###  
  erb :win # You may change/remove this line  
end  
  
get '/lose' do  
  ### YOUR CODE HERE ###  
  erb :lose # You may change/remove this line  
end  
  
end
```

Veamos el caso de show , la prueba para este es cheating.feature , ejecutamos el comando con cucumber y nos da lo siguiente :

```

features/cheating.feature:10:in `I should be on
And the word should read "-----" # feat

Scenario: navigate to win page # feat
Given I start a new game with word "snake" # feat
When I try to go to the URL "/win" # feat
Then I should be on the show page # feat

expected: "/show"
got: "/win"

(compared using eql?)
(RSpec::Expectations::ExpectationNotMetError)
./features/step_definitions/game_steps.rb:67:in
features/cheating.feature:16:in `I should be on
And the word should read "-----" # feat

Failing Scenarios:
cucumber features/cheating.feature:7 # Scenario: navi
cucumber features/cheating.feature:13 # Scenario: nav

2 scenarios (2 failed)
8 steps (2 failed, 2 skipped, 4 passed)
0m0.420s

```

vemos que tambien depende de win y lose asi

que completamos codigo para win ,lose y show

```

get '/show' do
  # Verifica el resultado del juego utilizando @game.check_win_or_lose
  result = @game.check_win_or_lose
  # Evalúa el resultado y realiza acciones en consecuencia
  case result
  when :win
    # Si el resultado es :win, redirige a la ruta '/win'
    redirect '/win'
  when :lose
    # Si el resultado es :lose, redirige a la ruta '/lose'
    redirect '/lose'
  else
    # Si el resultado no es :win ni :lose, muestra la vista 'show'
    erb :show
  end
end

get '/win' do
  # Verifica el resultado del juego utilizando @game.check_win_or_lose
  result = @game.check_win_or_lose
  # Redirige a la ruta '/show' si el resultado es :play
  redirect '/show' if result == :play
  # Si el resultado no es :play, muestra la vista 'win'
  erb :win
end

get '/lose' do
  # Verifica el resultado del juego utilizando @game.check_win_or_lose
  result = @game.check_win_or_lose
  # Redirige a la ruta '/show' si el resultado es :play
  redirect '/show' if result == :play
  # Si el resultado no es :play, muestra la vista 'lose'
  erb :lose
end
end

```

Ejecutamos nuevamente las pruebas :

Para win o lose , esta `game_over.feature` , cuando ejecutamos el test en este hay algunos que logran pasar el test , esto sucede porque no depende solo de win o lose sino tambien de guess , aqui es donde pasan las pruebas .(la imagen es antes de completar win y lose)

Ejecutando las pruebas con win y lose implementados :

```
Coverage report generated for Cucumber Features to /home/jaime/Documentos/PracticaCalificadal_CC3S2/coverage. 51 / 73 LOC (69.86%) covered.
jaime@jaime-VirtualBox:~/Documentos/PracticaCalificadal_CC3S2$ 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)
0m0.323s
```