

# Práctica Calificada 1

## Contenido

<b>Introducción</b>	<b>2</b>
Creación y versionado de una aplicación SaaS sencilla	2
Correr el Bundler	2
Crea una aplicación SaaS sencilla con Sinatra	4
Modifica la aplicación	5
Implementar en Heroku	6
<b>Parte 1: Wordguesser</b>	<b>7</b>
Desarrollando Wordguesser usando TDD y Guard	7
Depuración	8
<b>Parte 2: RESTful para Wordguesser</b>	<b>10</b>
El juego como recurso RESTful	10
Asignación de rutas de recursos a solicitudes HTTP	10
<b>Parte 3: Conexión de WordGuesserGame a Sinatra</b>	<b>11</b>
La sesión	11
Ejecutando la aplicación Sinatra	11
<b>Parte 4: Cucumber</b>	<b>13</b>
Desarrollar el escenario para adivinar una letra	14
<b>Parte 5: Otros casos</b>	<b>15</b>
Envío	15

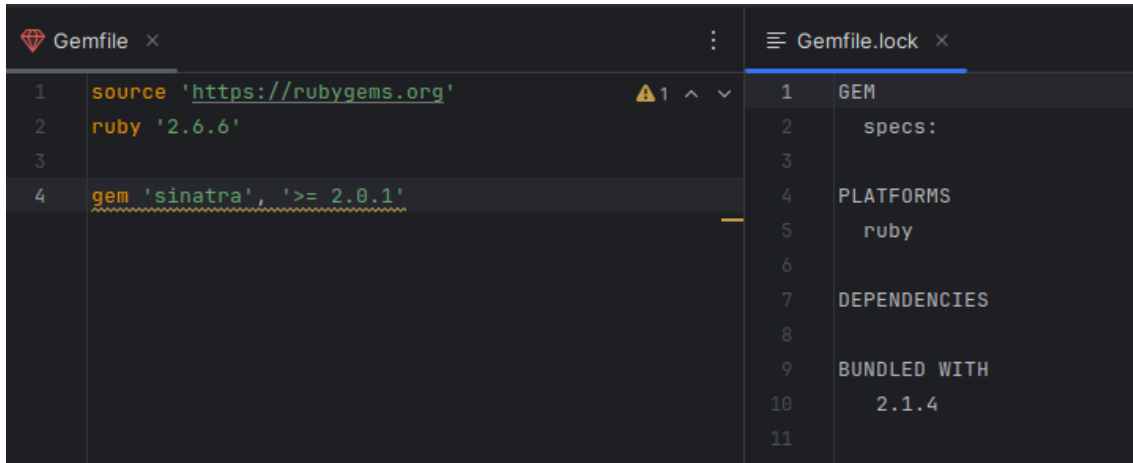
## Introducción

### Creación y versionado de una aplicación SaaS sencilla

Creamos el archivo Gemfile que aparece en el siguiente apartado y añadimos con git para su versionado.

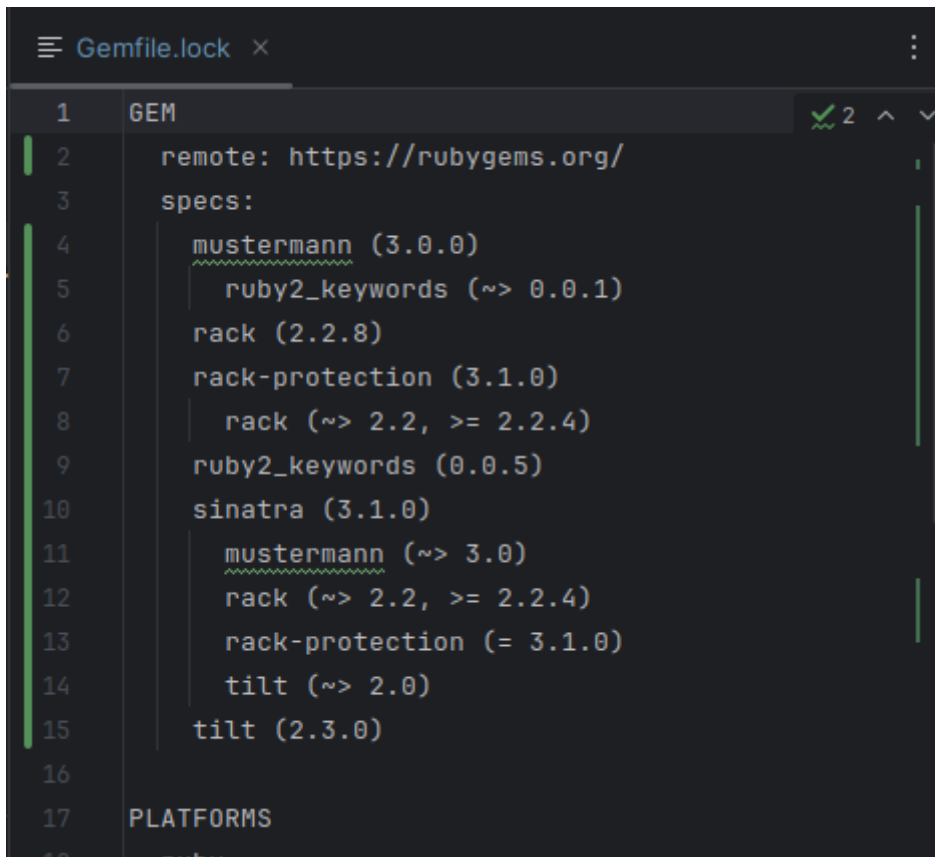
### Correr el Bundler

Aquí tenemos nuestros archivos Gemfile y Gemfile.lock antes de ejecutar el comando bundle.



```
$ bundle
# Your Ruby version is 2.7.0, but your Gemfile specified 2.6.6
$ source ~/.rvm/scripts/rvm
$ bundle
# ... (Gem::GemNotFoundException)
# To install the missing version, run `gem install bundler:2.1.4`
$ gem install bundler:2.1.4
# ... 1 gem installed
$ bundle
# ... Bundle complete! 1 Gemfile dependency, 7 gems now installed.
```

El Gemfile.lock cambió:



```
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
```

**Pregunta.** ¿Cuál es la diferencia entre la finalidad y el contenido de Gemfile y Gemfile.lock?  
¿Qué archivo se necesita para reproducir completamente las gemas del entorno de desarrollo en el entorno de producción?

**Respuesta.** El archivo Gemfile especifica las gemas que necesitas (p. e. gem 'sqlite3') y, en algunos casos, las restricciones sobre qué versión(es) son aceptables (p. e. gem 'puma', '>= 1.2', '< 2.0'). Gemfile.lock registra las versiones reales encontradas, no solo de las gemas que especificaste explícitamente, sino también de cualquier otra gema de la que dependen, por lo que es el archivo utilizado por tu entorno de producción para reproducir las gemas disponibles en tu entorno de desarrollo.

**Pregunta.** Después de ejecutar bundle, ¿por qué hay gemas listadas en Gemfile.lock que no estaban listadas en Gemfile?

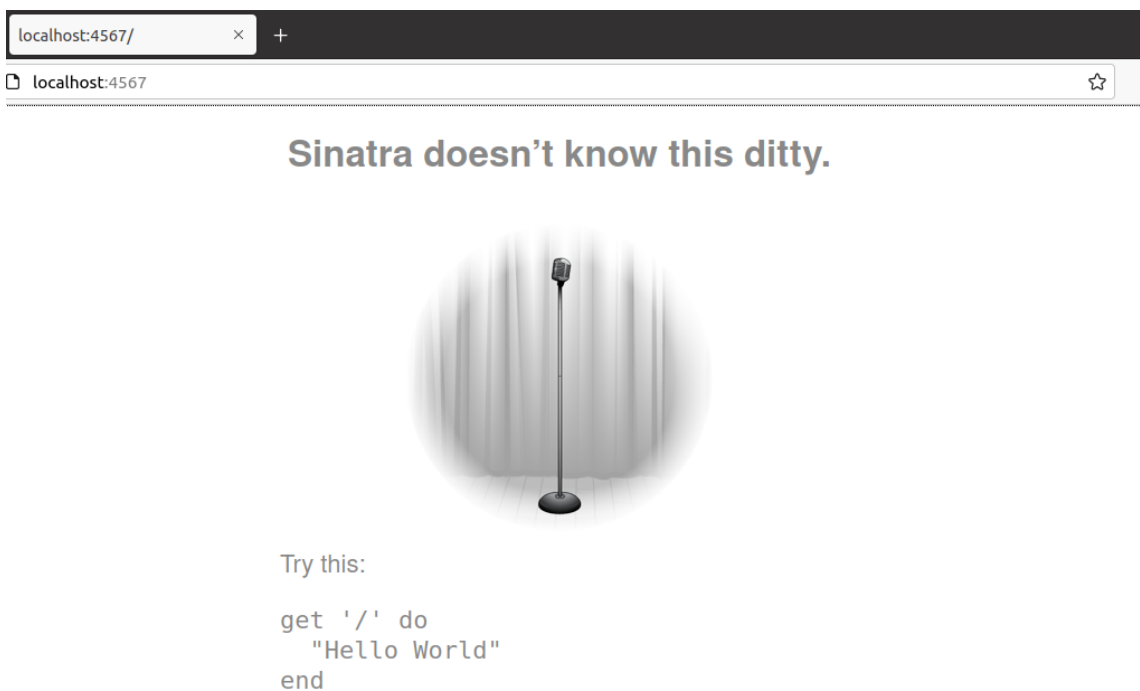
**Respuesta.** Porque esas otras gemas que no están en el Gemfile son gemas de las que dependen las gemas que sí están en el Gemfile. Son el resultado de la búsqueda recursiva que hace Bundler para que a ninguna gema le falte una gema de la que aquella depende. En el Gemfile.lock la indentación nos indica las gemas usadas por otras gemas. Por ejemplo, ni Rack ni Tilt figuran en el Gemfile, pero el Gemfile.lock indica que Sinatra usa las gemas Rack y Tilt como dependencias.

## Crea una aplicación SaaS sencilla con Sinatra

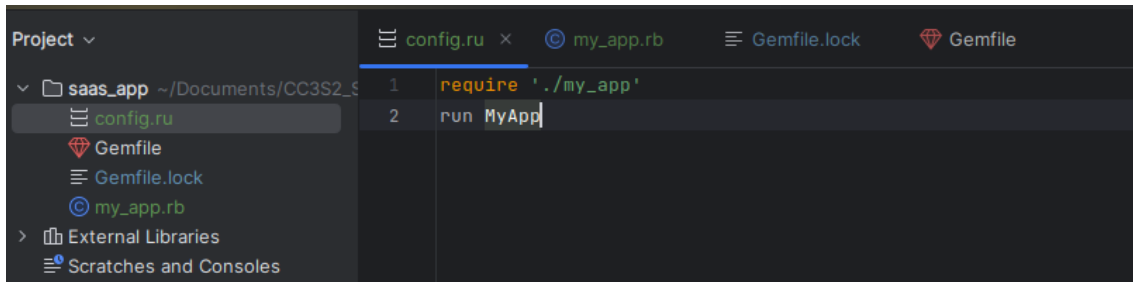
Ejecutamos nuestra aplicación, y se instancia un servidor de aplicación que escucha a través del puerto 4567:

```
my_app.rb x Gemfile.lock Gemfile
1 require 'sinatra'
2
3 new +
4 class MyApp < Sinatra::Base
5   get path '/' do
6     "<!DOCTYPE html><html><head></head><body><h1>Hello World</h1></body></html>"
7   end
8 end
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
101
```

Pero tenemos un problema con el servidor:

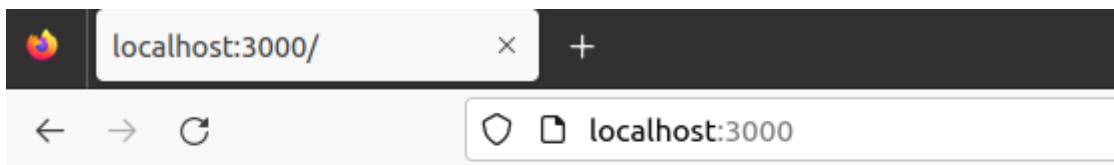


El problema es que no hemos creado el archivo config.ru para configurarlo:



Con esta configuración ahora el servidor de aplicación ya se puede conectar a nuestra aplicación y mostrar el breve contenido HTML de saludo en el método `get /` de nuestra app. Solo ejecutamos el siguiente comando:

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



# Hello World

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

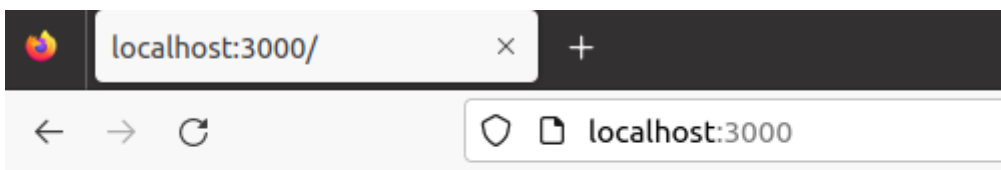
**Respuesta.** Nos sale el mismo error de antes, pero esta vez es porque no tenemos una ruta para el método `get /hola` .

## Modifica la aplicación

Modificamos la aplicación para que muestre el texto Goodbye World, pero al recargar la página que nos conecta con el servidor local, seguimos viendo lo mismo. Para verlos cambios tenemos que detener el servidor con `Ctrl+C` y volverlo a correr con `bundle exec rackup --port 3000`:

```
aldo@aldo:~/Documents/CC3S2_Software-Development_23-2/PC1/saas_app$ bundle exec rackup --port 3000
[2023-10-03 19:13:52] INFO  WEBrick 1.4.2
[2023-10-03 19:13:52] INFO  ruby 2.6.6 (2020-03-31) [x86_64-linux]
[2023-10-03 19:13:52] INFO  WEBrick::HTTPServer#start: pid=70044 port=3000
127.0.0.1 - - [03/Oct/2023:19:14:00 -0500] "GET / HTTP/1.1" 200 75 0.0324
^C[2023-10-03 19:14:09] INFO  going to shutdown ...
[2023-10-03 19:14:09] INFO  WEBrick::HTTPServer#start done.
aldo@aldo:~/Documents/CC3S2_Software-Development_23-2/PC1/saas_app$ bundle exec rackup --port 3000
[2023-10-03 19:14:31] INFO  WEBrick 1.4.2
[2023-10-03 19:14:31] INFO  ruby 2.6.6 (2020-03-31) [x86_64-linux]
[2023-10-03 19:14:31] INFO  WEBrick::HTTPServer#start: pid=70054 port=3000
127.0.0.1 - - [03/Oct/2023:19:14:35 -0500] "GET / HTTP/1.1" 200 76 0.0242
```

Ahora sí vemos los cambios:



# Goodbye World

Implementar en Heroku

## Parte 1: Wordguesser

### Desarrollando Wordguesser usando TDD y Guard

Clonamos un repositorio mediante el comando **git clone**

<https://github.com/saasbook/hw-sinatra-saas-wordguesser> y ejecutamos el comando **bundle** para asegurarnos de que las gemas estén instaladas y presentes en el entorno de desarrollo de nuestra aplicación:

```
hw-sinatra-saas-wordguesser$ bundle
Fetching gem metadata from https://rubygems.org/.....
Fetching ZenTest 4.11.2
Installing ZenTest 4.11.2
Fetching public_suffix 3.0.3
Installing public_suffix 3.0.3
Fetching addressable 2.5.2
Installing addressable 2.5.2
```

Corremos el comando **bundle exec autotest**

```
..._23-2/PCI1/part1_wordguesser/hw-sinatra-saas-wordguesser/spec/wordguesser_game_spec.rb"
Run options: exclude {:pending=>true}

All examples were filtered out

Finished in 0.00086 seconds (files took 2.15 seconds to load)
0 examples, 0 failures
```

Eliminamos el segmento `, :pending => true` de la línea 12...

```
11
12 >> describe 'new' do
13 >   it "takes a parameter and returns a WordGuesserGame object" do
14     @game = WordGuesserGame.new( word 'glorp')
15     expect(@game).to be_an_instance_of(WordGuesserGame)
16     expect(@game.word).to eq('glorp')
17     expect(@game.guesses).to eq('')
18     expect(@game.wrong_guesses).to eq('')
19   end
20 end
```

... y, como Rubymine está configurado con la opción de autosave, vemos inmediatamente que autotest detecta que hay una prueba y que falla:

```

11
12 >> describe 'new' do
13 >   it "takes a parameter and returns a WordGuesserGame object" do
14     @game = WordGuesserGame.new( word 'glorp')
15     expect(@game).to be_an_instance_of(WordGuesserGame)
16     expect(@game.word).to eq('glorp')
17     expect(@game.guesses).to eq('')
18     expect(@game.wrong_guesses).to eq('')
19   end
20 end

```

**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**?

**Respuesta.** El constructor de la clase de juegos **WordGuesserGame** espera solo un argumento (en este caso "glorp"). Cuando tratamos de pasar argumentos al método **new** para crear un nuevo objeto a partir de una clase, debemos definir un método en dicha clase llamado *initialize* al cual le pasemos los argumentos. Por eso la primera línea será **def initialize(word)**.

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

**Respuesta.** Se espera que un objeto **WordGuesserGame** tenga las variables de instancia **@word**, **@guesses** y **@wrong\_guesses**, de las cuales Rubymine solo detecta la existencia de una: **@word**.

## Depuración

```

new *
27 def guess(new_guess)
28   byebug
29   if @word.include? new_guess
30     guesses << new_guess

```

```

(byebug) s
[25, 34] in /home/aldo/Documents/CC3S2_Software-Development_23-2/PC1/part1_wordguesser/hw-sinatra-saas-wordguesser/lib/wordguesser_game.rb
25:   }
26: end
27: def guess(new_guess)
28:   byebug
29:   if @word.include? new_guess
=> 30:     guesses << new_guess
31:   else
32:     wrong_guesses.include? new_guess
33:   end
34:
(byebug) new_guess
"a"

```



```
aldo@aldo:~/Documents/part1_wordguesser/hw-sinatra-saas-wordguesser$ curl --data '' http://randomword.sasbook.info/RandomWord
tightfistedaldo@aldo:~/Documents/part1_wordguesser/hw-sinatra-saas-wordguesser$ curl --data '' http://randomword.sasbook.info/RandomWord
stripedaldo@aldo:~/Documents/part1_wordguesser/hw-sinatra-saas-wordguesser$ █
```

## Parte 2: RESTful para Wordguesser

**Pregunta.** Enumera el mínimo estado del juego que debe mantenerse durante una partida de Wordguesser.

**Respuesta.** La palabra secreta, la lista de letras que han sido adivinadas correctamente y la lista de letras que han sido adivinadas incorrectamente. Estos componentes del estado del juego ya están codificados como variables de instancia de la clase WordGuesserGame, con lo cual esta clase encapsula el estado.

### El juego como recurso RESTful

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

**Respuesta.** Hay dos acciones que pueden cambiar el estado del juego: adivinar una letra y empezar una nueva partida. Adivinar una letra o bien modifica la lista de letras adivinadas correcta o incorrectamente, o bien resulta en ganar o perder el juego. Por su parte, empezar una nueva partida desencadena la elección de una nueva palabra secreta y vacía las listas de letras adivinadas correcta e incorrectamente.

### Asignación de rutas de recursos a solicitudes HTTP

**Pregunta.** Para un buen diseño RESTful, ¿cuáles de las operaciones del recurso deben ser manejadas por HTTP GET y cuáles por HTTP POST?

**Respuesta.** Hasta el momento contemplamos tres operaciones: show, Las operaciones manejadas con GET no deberían tener efectos secundarios en el recurso, así que mostrar puede ser manejado por un GET, pero crear y adivinar (que modifican el estado del juego) deberían usar POST. (De hecho, en una verdadera arquitectura orientada a servicios también podemos optar por utilizar otros verbos HTTP como PUT y DELETE, pero no cubriremos eso en esta tarea).

**Pregunta.** ¿Por qué es conveniente que la nueva acción utilice GET en lugar de POST?

**Respuesta.** El uso de GET para la nueva acción es apropiado porque solo implica presentar un formulario al jugador, permitiéndole enviarlo, sin causar ninguna alteración inherente al estado del sistema.

**Pregunta.** Explique por qué la acción GET /nuevo no sería necesaria si su juego Wordguesser fuera llamado como un servicio en una verdadera arquitectura orientada a servicios.

**Respuesta.** En una auténtica Arquitectura Orientada a Servicios (SOA), el servicio que interactúa con Wordguesser sería capaz de generar directamente una petición HTTP POST. En una configuración de este tipo, la acción GET /new se hace innecesaria, ya que su propósito principal es facilitar a los usuarios la generación de esta solicitud. En un entorno SOA puro, este paso intermedio para la interacción humana se omite.

## Parte 3: Conexión de WordGuesserGame a Sinatra

**Pregunta.** ¿@game en este contexto es una variable de instancia de qué clase?

**Respuesta.** Es importante tener en cuenta que, en este contexto particular, la variable de instancia "@game" está asociada con la clase WordGuesserApp, que se encuentra en el archivo app.rb. Puede ser un poco difícil de discernir porque estamos tratando con dos clases Ruby separadas aquí. Una de ellas es WordGuesserGame, responsable de encapsular la lógica central del juego. Por otro lado, WordGuesserApp gestiona la lógica necesaria para ofrecer el juego como una aplicación Software as a Service (SaaS). Esencialmente, se puede pensar en WordGuesserApp como un controlador que abarca la lógica y la capacidad de renderizar vistas utilizando erb.

### La sesión

**Pregunta.** ¿Por qué esto ahorra trabajo en comparación con simplemente almacenar esos mensajes en el hash de `session []`?

**Respuesta:** La diferencia de funcionalidad proviene de la duración de la persistencia de los datos. Cuando almacenamos algo en el hash `session[]`, permanece allí hasta que decidimos eliminarlo manualmente. Esto es apropiado cuando los datos necesitan persistir a través de múltiples peticiones. Sin embargo, si la intención es mostrar un mensaje solo una vez, especialmente después de una redirección, el hash `flash[]` de la gema sinatra-flash proporciona una clara ventaja. No solo retiene el mensaje para la petición en curso, sino que también lo elimina automáticamente después de la siguiente petición, que suele ser una redirección. Este comportamiento se adapta a situaciones en las que se requieren mensajes de corta duración, asegurando que se muestran precisamente cuando se necesitan y se borran rápidamente de la memoria.

### Ejecutando la aplicación Sinatra

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

**Respuesta.** Para acceder a la página New Game, se necesita usar la URL `http://localhost:3000/new`, ya que la sección de código Ruby en app.rb marcada por `get '/new'` do... es responsable de renderizar esta página específica.

**Pregunta.** ¿Dónde está el código HTML de esta página?

**Respuesta.** El código HTML de la página New Game se encuentra en el archivo `views/new.erb`. Es un archivo que se procesa en HTML a través de la directiva `erb :new`, permitiendo la correcta visualización de la página.

## Parte 4: Cucumber

**Pregunta.** 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?

**Respuesta.** Las definiciones de pasos que emplean métodos como `visit`, `click_button` y `fill_in` simulan activamente el comportamiento de un navegador visitando páginas web, interactuando con formularios y pulsando botones, tal y como lo haría un usuario. Por otro lado, las definiciones de pasos que utilizan `have_content` inspeccionan la respuesta de la aplicación buscando contenido específico en las páginas HTML servidas.

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

**Respuesta:** Las tres líneas que siguen al título "Feature:" sirven como comentarios que aclaran el propósito y los actores implicados en el reportaje concreto. Proporcionan una breve descripción sobre de qué trata la función y a quién implica. Es importante tener en cuenta que Cucumber no ejecutará estas líneas; simplemente proporcionan contexto y documentación para la característica.

**Pregunta.** 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?

**Respuesta:** Cuando Cucumber intenta ejecutar el paso "Given I start a new game with word 'garply'", las líneas 13 a 16 en el archivo `game_steps.rb` serán invocadas. Este paso en particular coincide con una expresión regular, y la cadena "garply" juega el papel de un parámetro o argumento que se pasa a la definición del paso. En este caso, se utiliza para especificar la palabra con la que se debe iniciar el nuevo juego, lo que permite escenarios de prueba dinámicos y personalizables.

### 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?

**Respuesta.** El método `visit` de Capybara siempre opta por una petición HTTP GET cuando emite una petición como `visit '/new'`. La razón detrás de esta elección está arraigada en la filosofía de que Cucumber/Capybara está diseñado para emular las acciones de un usuario humano cuando interactúa con una aplicación web. Como hemos comentado antes, en el ámbito de los navegadores web, un usuario humano normalmente inicia una petición POST enviando un formulario HTML. Este acto de enviar un formulario se refleja en Capybara usando el método

click\_button. Por lo tanto, la decisión de utilizar GET con visita se alinea con la idea de que Capybara debe imitar el comportamiento natural de los usuarios humanos en escenarios de navegación web.

**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.

**Respuesta.** Los términos "Dado", "Cuando" y "Entonces" dentro de un archivo de características de Cucumber representan alias intercambiables para la misma función. Sirven para estructurar escenarios. "Dado" establece el contexto inicial, "Cuándo" describe la acción que tiene lugar y "Entonces" especifica el resultado esperado. Aunque usted tiene la flexibilidad para experimentar con la alteración de su secuencia, el orden convencional es "Dado", "Cuando" y "Entonces" en aras de la claridad y la comprensibilidad. Cucumber en sí no impone un orden estricto para estas palabras clave, pero adherirse a la convención mejora la comprensión del escenario tanto para desarrolladores como para no desarrolladores.

## Desarrollar el escenario para adivinar una letra

**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).

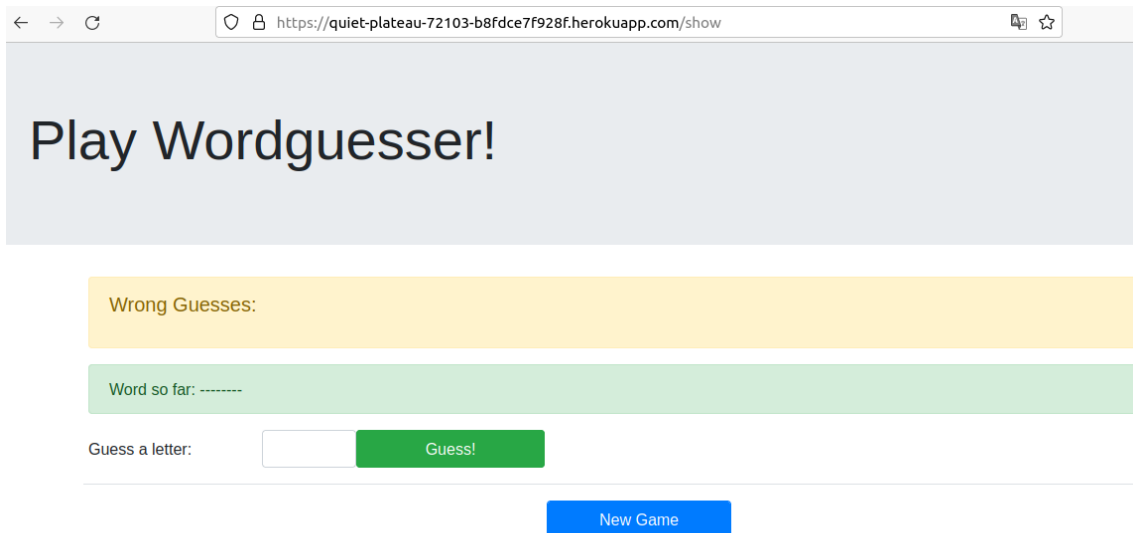
**Respuesta:** La línea que incluye `stub_request` en `game_steps.rb` hace uso de la gema `Webmock` para permitir que nuestras pruebas capturen peticiones HTTP iniciadas por nuestra aplicación y dirigidas a servicios externos. En concreto, intercepta una solicitud POST concreta, similar a la realizada manualmente mediante `curl` anteriormente en esta tarea. Al interceptar la petición, `Webmock` nos permite fabricar un valor de respuesta. Esta interceptación y manipulación de la respuesta son vitales por varias razones. En primer lugar, nos permite asegurar un comportamiento consistente y predecible en nuestras pruebas. Esta previsibilidad es crucial para la eficacia de las pruebas. En segundo lugar, evita que nuestras pruebas envíen solicitudes a servidores externos cada vez que se ejecutan, lo que puede ser lento y potencialmente problemático para el servicio externo. Por lo tanto, `Webmock` es una valiosa herramienta para controlar y mejorar el entorno de pruebas.





**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

**Pregunta.** Mientras juegas, ¿qué sucede si agregas **/win** directamente al final de la URL de tu aplicación?

No se redirige a una página donde se muestra que ha ganado, simplemente se queda en la misma pestaña. decir, <https://quiet-plateau-72103-b8fdce7f928f.herokuapp.com/show>



← → ↻   https://quiet-plateau-72103-b8fdce7f928f.herokuapp.com/show  

# Play Wordguesser!

Wrong Guesses:

Word so far: -----

Guess a letter:

Envío