Universidad de Sevilla
Escuela Técnica Superior de Ingeniería Informática

Feeders en Gatling

Grado en Ingeniería Informática – Ingeniería del Software
Diseño y Pruebas II

Curso 2019 – 2020

Miembros del equipo

Jorge Andrea Molina

Juan Carlos Cortés Muñoz

María Elena Molino Peña

Alejandro Muñoz Aranda

Mario Ruano Fernández

Fernando Ruíz Robles

https://github.com/dp2-g3-7/petclinic.git

Índice

Feeders en Gatling

Poniendo en valor las pruebas de rendimiento

Las pruebas de rendimiento pueden considerarse un buen primer paso para afrontar diferentes estrategias de optimización. Conocer dónde hay posibles puntos de mejora en nuestro código es primordial si queremos también optimizar recursos y tiempo empleado.

Gracias a las pruebas de rendimiento, pudimos tener una idea global de posibles problemas en la implementación de determinadas historias de usuario, por lo que nos ayudaron a repensar el código y volver a él. Todo esto puede llevarnos a perfilar la forma en la que se había ideado una funcionalidad y además sirve para adquirir experiencia sobre el desarrollo.

Sin embargo, ¿realmente qué estamos probando? Aunque se trate de una aproximación que nos deja a entrever bastantes puntos de mejora, las pruebas de rendimiento necesitan de escenarios de uso reales para ser 100% fiables. Sin un elevado ratio de éxito, no podemos asegurar que el rendimiento de nuestra aplicación es el correcto.

Aquí entran en juego funcionalidades que incorpora Gatling, de modo que podamos configurar un escenario de pruebas lo más real posible y ponerlo a funcionar, de manera concurrente, en un contexto de alta carga de trabajo y estrés para el sistema. Gracias a los "feeders", se introducen grandes lotes de datos que permiten que las peticiones puedan satisfacerse, tal como se espera que ocurra, para un determinado caso de uso de una historia de usuario.

Tras el lanzamiento de un script que se vale de esta funcionalidad, se puede observar cómo los datos que se obtienen de las pruebas de rendimiento comienzan a variar hacia un valor más fiable, capaz de reportar una información más veraz.

De este modo, tras el proceso de performance y profiling, decidimos utilizar esta técnica para repasar el rendimiento en dos historias de usuario implementadas: registrar medicamentos y registrar veterinarios.

Preparando las pruebas con feeders

Para la realización de las pruebas de rendimiento con feeders se procedió del siguiente modo:

1. Documentación sobre el uso de la herramienta y sus funcionalidades para implementar las pruebas con feeders:
    i. Feeders
    ii. Checks
2. Generar ficheros csv de datos con Mockaroo. Estos ficheros se localizan en src/performance/feeders
3. Modificación de los scripts RegistrarMedicamentos y RegistrarVeterinario.

A continuación, se indican los cambios que se realizaron en los scripts, tomando como ejemplo la historia de usuario RegistrarMedicamentos.

En primer lugar, hay que inicializar una variable que contenga los datos, que haga de feeder (alimentador). Para ello, hacemos uso de la función csv y pasamos como parámetro la ruta al archivo csv que hemos generado con Mockaroo.

```
val feeder = csv("medicines.csv")
```

Este fichero csv contiene las diferentes entradas que se utilizarán como entradas a la hora de registrar un medicamento en el sistema, del mismo modo que si se tratara de un test unitario parametrizado. Aquí se muestran las primeras líneas de dicho fichero medicines.csv.

```
name,code,expiration_date,description
TUMS,KCP-393156341,2023/02/16,et magnis dis parturient montes nascetur ridiculus mus
         etiam vel augue vestibulum rutrum rutrum neque aenean
COMETRIQ,LKM-455728038,2024/12/21,et ultrices posuere cubilia curae donec pharetra
         magna vestibulum aliquet ultrices erat tortor sollicitudin mi sit amet
         lobortis sapien sapien
DOUBLE PERFECTION LUMIERE,ZWN-135087657,2025/12/27,consequat varius integer ac leo
         pellentesque ultrices mattis odio donec vitae nisi nam ultrices libero
Hand Kleen Premium,CIO-044328535,2021/10/14,eros suspendisse accumsan tortor quis
         turpis sed ante vivamus tortor duis
```

Por último, hay que hacer un par de modificaciones sencillas en los scripts que previamente se usaron para realizar las primeras pruebas de rendimiento. El objetivo es que se cumpla compruebe que el estado de la respuesta es el esperado, lo que nos garantiza que el escenario se completó con el resultado satisfactorio.

```
object AddMedicines {
  val addMedicines = exec(http("MedicineForm")
    .get("/medicines/new")
    .headers(headers_0)
    .check(css("input[name=_csrf]", "value").saveAs("stoken")))
  .pause(15)
  .feed(feeder)    // Justo aquí se insertan datos, alimentando a
                   // la petición POST (tenemos una entrada nueva en BBDD).
  .exec(http("AddMedicines")
    .post("/medicines/new")
    .headers(headers_3)
    .formParam("name", "${name}")  // Aquí se indican los nombres de las
                                   // columnas que correspondan del fichero csv.
    .formParam("code", "${code}")
    .formParam("expirationDate", "${expiration_date}")
    .formParam("description", "${description}")
    .formParam("_csrf", "${stoken}")
    .check(currentLocationRegex("http://www.dp2.com/medicines")))
        // Si la petición se resuelve como se espera, se comprueba que se
        // retorna a dicha vista.
}
```

Además, hay que hacer un cambio en el setup del escenario, indicando que que no debe haber fallo alguno en cualquiera de las peticiones que se lanzan.

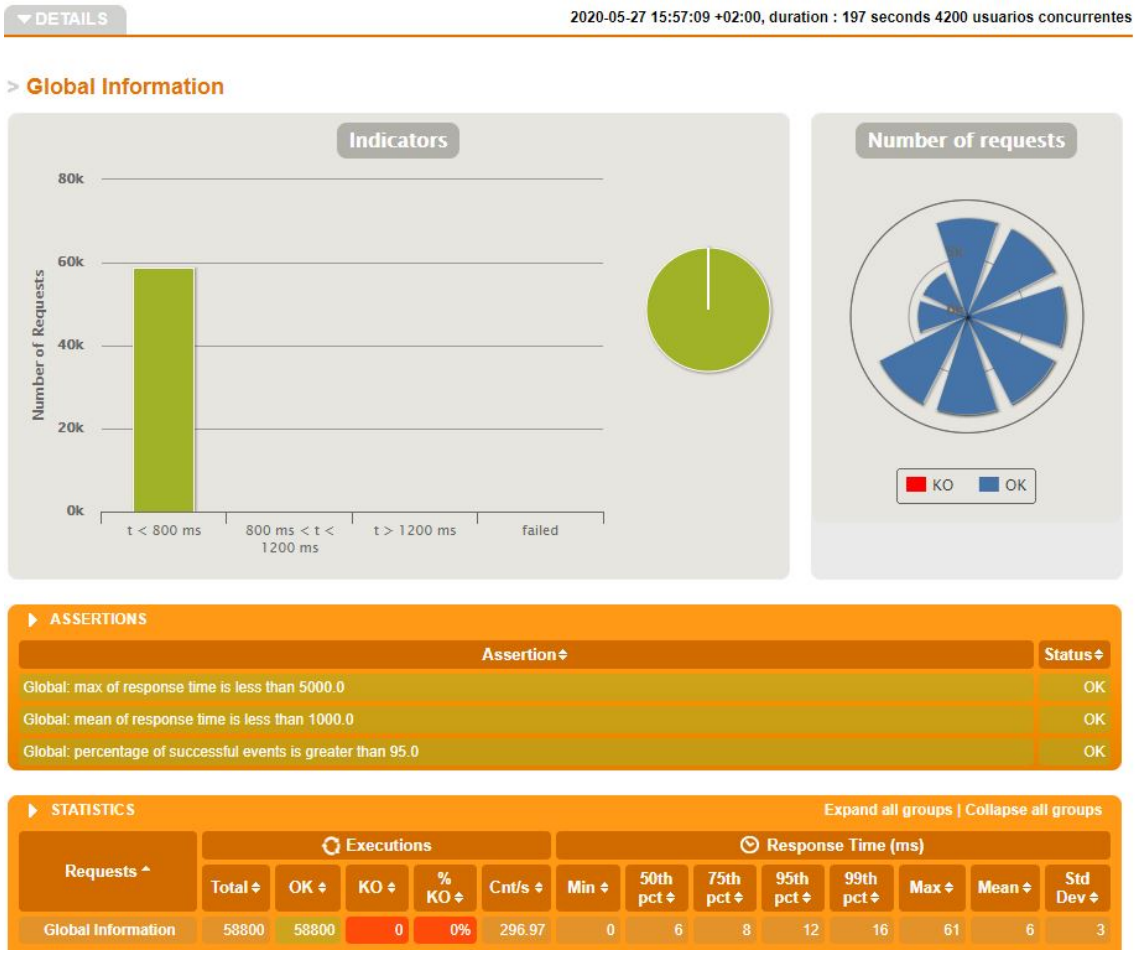
```
setUp(
  addMedicineScn.inject(rampUsers(1800) during (120 seconds)),
  dontAddMedicineScn.inject(rampUsers(2000) during (100 seconds)))
    .protocols(httpProtocol)
    .assertions(
      global.responseTime.max.lt(5000),
      global.responseTime.mean.lt(1000),
      global.successfulRequests.percent.gt(95),
      forAll.failedRequests.percent.lte(0) // Se indica que el porcentaje de
                                           // fallos en las peticiones debe
                                           // ser cero.
)
```

Con esto, nos aseguramos que, de ejecutarse con éxito las pruebas de rendimiento, se estarían cumpliendo con las reglas de negocio de los escenarios de prueba especificados.

Nuevos resultados tras ejecutar

Tal como se dijo al principio, el valor de las pruebas de rendimiento para detectar posibles puntos de mejora y de optimización, pasa por contar con una ejecución y una carga de trabajo lo más cercana a la realidad. Solo así sabremos si estamos poniendo a prueba o no a la implementación.

Estos fueron los resultados para la historia de usuario que nos detiene, en un primer momento, antes de utilizar feeders:



Con una considerable carga de trabajo, con 4200 usuarios concurrentes para cada escenario de uso y un tiempo de respuesta realmente rápido. Pero, ¿realmente se estaban realizando de manera exitosa los escenarios? La respuesta es evidente. No se cumplía con la regla de negocio que dice que un medicamento no puede tener el mismo código ("code"). Solo se registraba de forma exitosa la primera petición POST que se realizaba.

Entonces, llegados a este punto, ¿podemos confiar en los resultados del test de rendimiento?

Tras realizar de nuevo las pruebas de rendimiento con las citadas modificaciones, detectamos que el rendimiento máximo aceptable indicado en un principio era mucho más ambicioso de lo que se estaba observando con el uso de feeders. Por lo general, el rendimiento era bueno, pero se detectaban cuellos de botella en las peticiones GET que listaban todas las medicinas o todos los veterinarios.

ASSERTIONS

Assertion	Status
Global: mean of response time is less than 1000.0	OK
Global: percentage of successful events is greater than 95.0	OK
Home: percentage of failed events is less than or equal to 0.0	OK
Login: percentage of failed events is less than or equal to 0.0	OK
Logged: percentage of failed events is less than or equal to 0.0	OK
Logged Redirect 1: percentage of failed events is less than or equal to 0.0	OK
listMedicines: percentage of failed events is less than or equal to 0.0	OK
MedicineForm: percentage of failed events is less than or equal to 0.0	OK
DontAddMedicines: percentage of failed events is less than or equal to 0.0	OK
AddMedicines: percentage of failed events is less than or equal to 0.0	OK
AddMedicines Redirect 1: percentage of failed events is less than or equal to 0.0	OK

STATISTICS — Expand all groups | Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	28332	28332	0	0%	164.721	0	3	5	36	58	1599	10	49
Home	3800	3800	0	0%	22.093	1	3	4	7	512	1599	15	110
Login	3800	3800	0	0%	22.093	0	1	1	2	4	12	1	1
Logged	3800	3800	0	0%	22.093	1	2	2	3	8	194	2	8
Logged Redirect 1	3800	3800	0	0%	22.093	1	2	2	4	6	37	2	2
listMedicines	3800	3800	0	0%	22.093	3	17	31	42	188	276	22	26
MedicineForm	3800	3800	0	0%	22.093	2	3	4	6	467	604	12	64
DontAddMedicines	2000	2000	0	0%	11.628	3	5	5	8	11	55	5	2
AddMedicines	1800	1800	0	0%	10.465	2	3	4	6	11	156	4	4
AddMedic...direct 1	1732	1732	0	0%	10.07	3	33	41	57	65	80	31	15

Si bien es verdad que pensamos la aplicación para ser utilizada en pequeñas clínicas veterinarias de manera independiente, es decir, mediante una instalación independiente y sin compartir datos, es obvio que no sería un caso real tener que listar a 3000 o 4000 veterinarios. Sin embargo este cuello de botella nos da una importante pista de dónde optimizar, por ejemplo, mediante la aplicación de la técnica de paginación.

De no haber realizado este tipo de pruebas avanzadas, podríamos haber concluido que la implementación era perfectamente correcta para una carga de trabajo muy superior a la pensada. Sin embargo, se observa que esta implementación no escala bien y debería

optimizarse, ya que se tuvo que reducir la carga de trabajo en un 50% respecto a los resultados iniciales.

A pesar de ello, de que se podría obtener una importante mejora aquí, en el momento en el que decidimos realizar dichas pruebas ya habíamos concluido con la labor de profiling para otras historias de usuario, por lo que se escapaba de nuestras posibilidades invertir más tiempo en estas nuevas optimizaciones.