

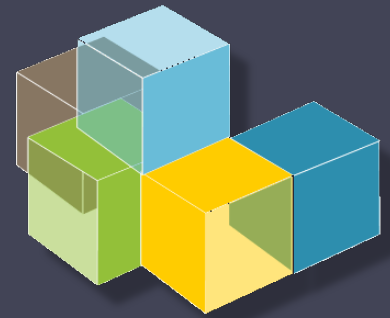


Universidad de Oviedo



Escuela de
Ingeniería
Informática

ES
Español



ARQUITECTURA
DEL SOFTWARE

Arquitectura del Software

Lab. 11

- Pruebas de carga
- Otras pruebas

2021-2022

José Emilio Labra Gayo
Pablo González
Irene Cid
Hugo Lebrede

¿Qué son?

- Son pruebas que permiten probar con cargas de usuarios concurrentes
- Permiten saber que carga de trabajo soporta una aplicación y arquitectura determinada
- Nos ayudan a dimensionar la aplicación y poder anticiparnos antes de su caída



¿Qué permiten probar?

- Aplicaciones web (HTTP/HTTPS)
- SOAP/REST Web Services
- FTP
- Databases (JDBC)
- LDAP
- Mail (SMTP, POP3, IMAP)
- Java Objects
- Etc.

¿Por qué hacer estos test?

- Permiten anticiparnos a problemas de rendimiento en la aplicación, arquitectura o infraestructura
- Permiten detectar cuellos de botella
- Permiten demostrar numéricamente los escenarios de calidad pactados en el contrato

Variables externas

- Red
- Servicios externos
- Sistemas de seguridad
- Eventos del sistema

¿Probamos en producción?

- Recomendación : Nuestras medidas deben ser para entornos de producción en vacío.

Planes de carga

- Dependiendo si se conoce o no el número de usuarios esperados.
- Variables que hay que llegar a conocer :Límite en el que nuestra aplicación deja de funcionar.
- Pruebas de picos: nos permiten ver la recuperación de nuestro sistema ante estados próximos a la saturación
- Pruebas de resistencia: Se prueba con una carga de 50-70% durante un tiempo largo.

Formas de hacer estos test

- Scripts propios
 - Mucho trabajo
 - Poco flexible
 - Analíticas pobres
 - Curva de aprendizaje corta
- Herramientas de terceros
 - Menos trabajo
 - Más flexibles
 - Mejores analíticas
 - Curva de aprendizaje elevada (según herramienta)

Herramientas

- Apache JMeter
- **Gatling**
- Loader.io
- BlazeMeter
- Blitz
- Etc.

Guia paso a paso:

https://github.com/pglez82/docker_solid_example/tree/pglez82-gatling-load-tests#load-tests-gatling

Gatling

- Escrita en Scala
- Compatible con la JVM
- Uso de un DSL propio
- Fácil de usar
- Ligera

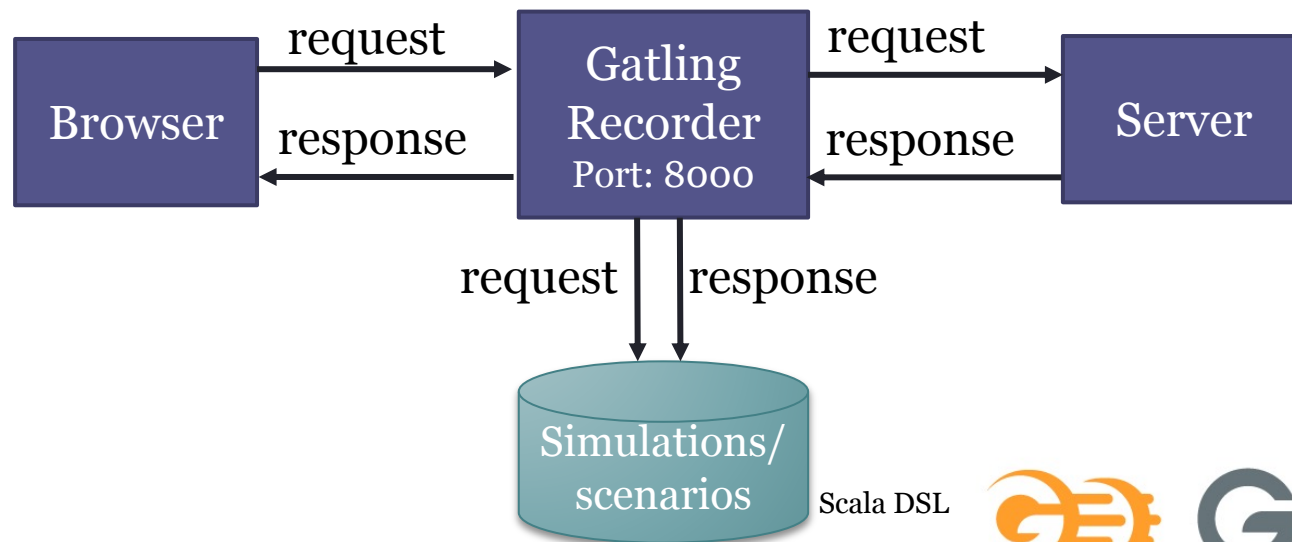
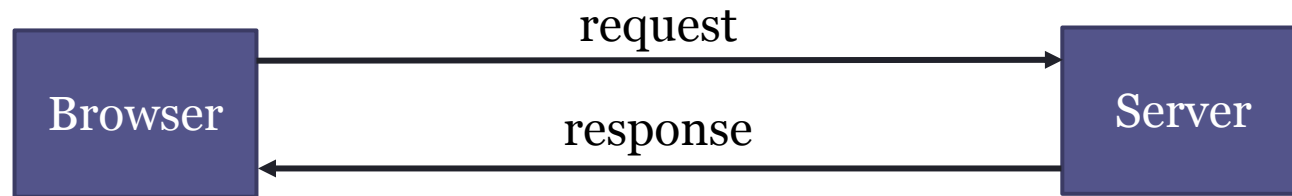


Descarga e instalación

- <http://gatling.io>
- Necesita tener Java 8 instalado (hecho en Scala)
- Listo para funcionar
- Dos scripts:
 - Recorder.sh/Recorder.bat
 - Gatling.sh/Gatling.bat



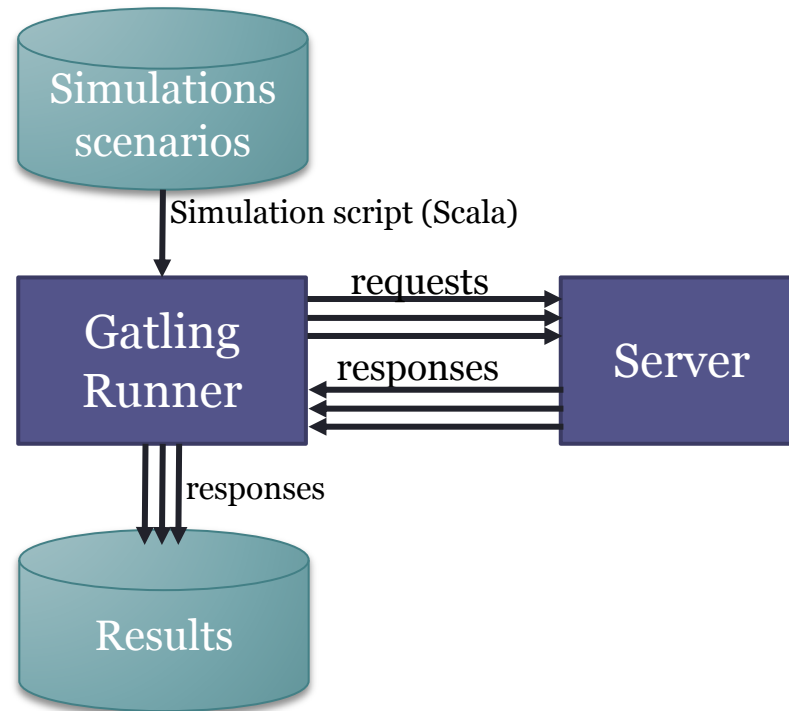
Gatling recorder



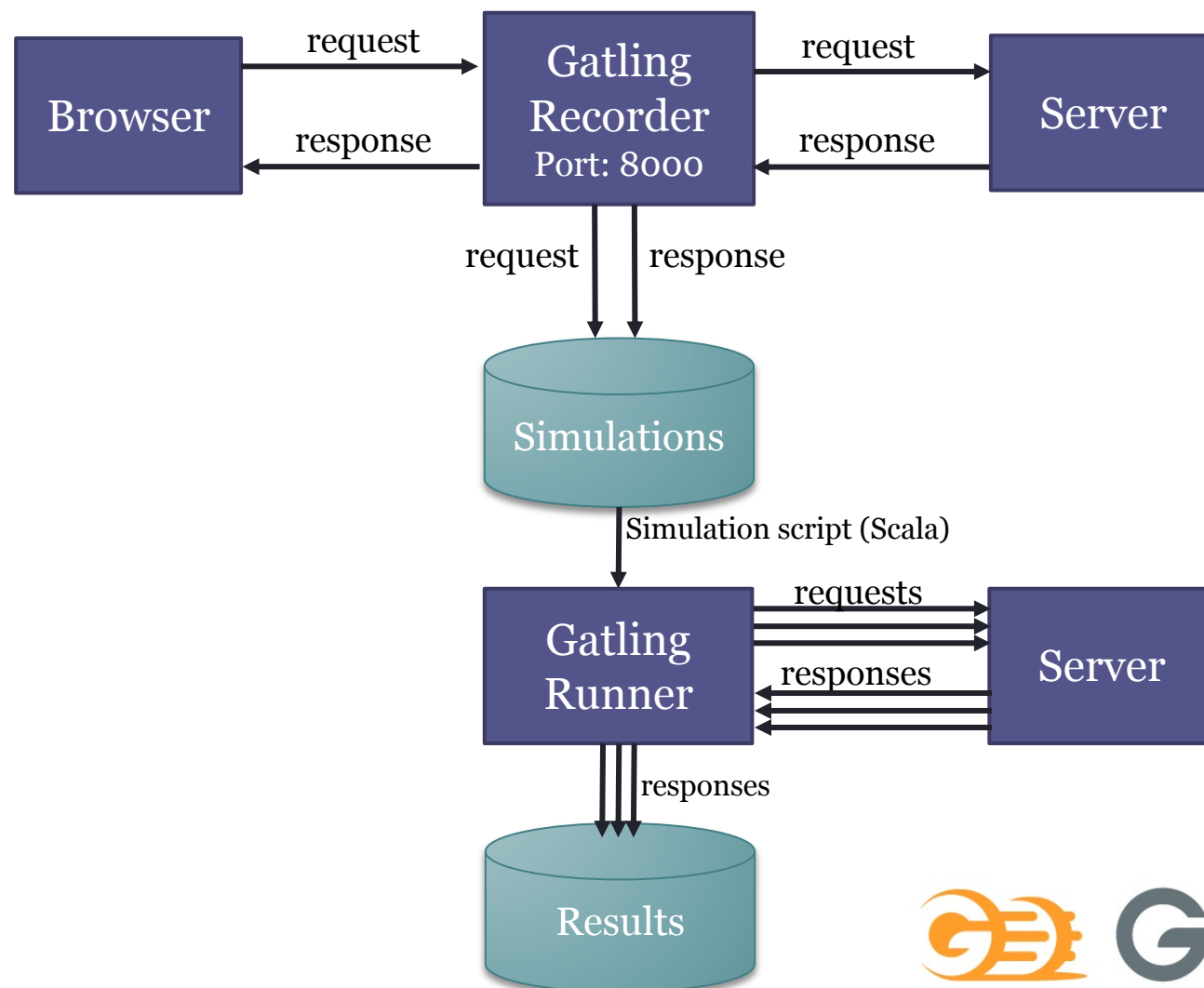
Scala DSL

**Gatling**

Gatling runner




Workflow



Recorder

Gatling Recorder - Configuration

 **Gatling**

Recorder mode: HTTP Proxy

Network

Listening port*: localhost HTTP/HTTPS 8000 HTTPS mode: Certificate Authority Generate CA

CA Certificate: /s-bundle-3.5.0/certs/gatlingCA.cert.pem Browse CA Private Key: /home/pablo/Programas/gatling-charts-t Browse

Outgoing proxy: host: HTTP HTTPS Username Password

Simulation Information

Package: Class Name*: GetUsersList

☒ Follow Redirects? ☒ Infer HTML resources? ☒ Automatic Referers? ☒ Remove cache headers?

☐ Use Class Name as request prefix? ☐ Use HTTP method and URI as request postfix? ☐ Save & check response bodies?

Output

Simulations folder*: /home/pablo/Programas/gatling-charts-highcharts-bundle-3.5.0/user-files/simulations Browse

Encoding: Unicode (UTF-8)

Filters

Java regular expressions that matches the entire URI Strategy BlackList First

Whitelist	Blacklist
	*.js
	*.css
	*.gif
	*.jpeg
	*.jpg
	*.ico

+ - Clear

+ - Clear No static resources

Save preferences ☒ Start !

Gatling: Recorder

Test case: dede_0

Lanzar recorder

```
pablo@pablo-ZenBook-UX431DA-UM431DA:~/Programas/gatling-charts-highcharts-bundle-3.5.0/bin$ ./recorder.sh
GATLING_HOME is set to /home/pablo/Programas/gatling-charts-highcharts-bundle-3.5.0
```

Configuración Recorder

- Generar los certificados
- Importar los certificados a firefox
- Configurar el puerto
- Otra configuración:
 1. Package: nombre paquete
 2. Name: nombre simulación
 3. Follow Redirects ☒
 4. Automatic Referers ☒
 5. Strategy: primero Black list
 6. Blacklist: *.*.css, *.*.js, etc

Gatling Recorder - Configuration

Network

Listening port: localhost HTTP/HTTPS 8000 HTTPS mode: Certificate Authority

CA Certificate: /s-bundle-3.5.0/certs/gatlingCA.cert.pem CA Private Key: /home/pablo/Programas/gatling-charts-h

Outgoing proxy: host: HTTP ☐ HTTPS ☐ Username Password

Simulation Information

Package: Class Name*: GetUsersList

☒ Follow Redirects? ☒ Infer HTML resources? ☒ Automatic Referers? ☒ Remove cache headers?

☐ Use Class Name as request prefix? ☐ Use HTTP method and URI as request postfix? ☐ Save & check response bodies?

Output

Simulations folder*: /home/pablo/Programas/gatling-charts-highcharts-bundle-3.5.0/user-files/simulations

Encoding: Unicode (UTF-8)

Filters

Java regular expressions that matches the entire URI

Strategy: BlackList First

Whitelist

Blacklist

*.js
*.css
*.gif
*.jpeg
*.jpg
*.ico

Save preferences ☒

Configure Proxy

localhost:8000

Para todas las direcciones, incluida localhost

Si se usa HTTPS hay que configurar el certificado.

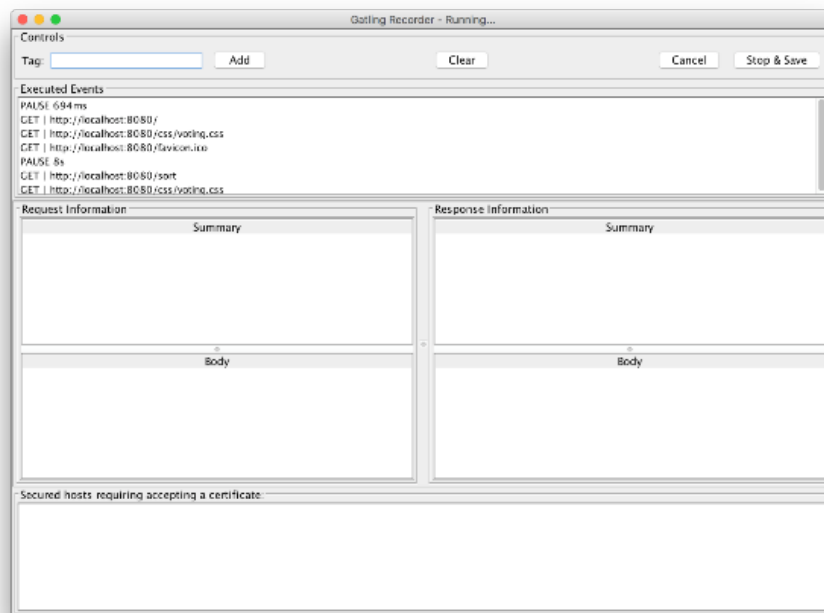


Para localhost en firefox, ejecutar:

`network.proxy.allow_hijacking_localhost` to true in `about:config`

Gatling: Recorder

- Navegador > Web Proxy > localhost:8000
- Recorder: Start
- Escenario de ejemplo:
 1. Después de comenzar, abrir el sitio web y realice las acciones que desea que formen parte de la prueba.
 2. Después de terminar presione STOP
 3. Las acciones se registrarán en lenguaje Scala.
 4. La simulación se guardará en el directorio archivos de *user-files/simulations*



Ejemplo Gatling

Consideraciones Previas:

- **Solo** en dede_o
- En este caso **solo** hemos cargado la página principal de la aplicación
- Tenga en cuenta la última línea de la prueba, podemos **ajustar la carga** aquí.
- Obviamente, las pruebas pueden ser mucho más complicadas, realizando múltiples acciones en el sistema.

https://github.com/Arquisoft/dede_o/blob/master/webapp/loadtestexample/GetUsersList.scala



Configurando el número de usuarios

Injection profile

Control how users are injected in your scenario

Injection steps

nothingFor

atOnceUsers

rampUsers

constantUsersPerSec

rampUsersPerSec

splitUsers

heavisideUsers

https://gatling.io/docs/current/general/simulation_setup



Ejemplo 1: 2 usuarios por segundo durante 60 segundos

- 120 usuarios llegando a una tasa de 2 usuarios / segundo
- Ejecutan un script dado

```
...  
setUp(  
    scn.inject(constantUsersPerSec(2) during (60 seconds) randomized)  
).protocols(httpProtocol)
```

Ejemplo 2: 50 usuarios en 60 segundos

- 50 usuarios concurrentes
- Entra un usuario nuevo cada 1,2 segundos
- Desarrollan todo el script grabado anteriormente

```
...  
setUp(scn.inject(rampUsers(50) during(60 seconds))).  
    protocols(httpProtocol)  
  
}
```

Disparando Gatling

- Ejecutar Script: `gatling.sh/.bat`
 - Escogemos la clase con el script grabado previamente (simulación)
 - Podemos configurar el ID y la descripción
- En la ejecución vamos viendo un progreso textual de la prueba
- Al finalizar genera un informe con analíticas y gráficas en un fichero HTML



Disparando Gatling

- Ejecutar Gatling (/bin/gatling.sh) y escoger el escenario

```
pablo@pablo-ZenBook-UX431DA-UM431DA:~/Programas/gatling-charts-highcharts-bundle-3.5.0/bin$ ./gatling.sh
GATLING_HOME is set to /home/pablo/Programas/gatling-charts-highcharts-bundle-3.5.0
Choose a simulation number:
  [0] GetUsersList
  [1] computerdatabase.BasicSimulation
  [2] computerdatabase.advanced.AdvancedSimulationStep01
  [3] computerdatabase.advanced.AdvancedSimulationStep02
  [4] computerdatabase.advanced.AdvancedSimulationStep03
  [5] computerdatabase.advanced.AdvancedSimulationStep04
  [6] computerdatabase.advanced.AdvancedSimulationStep05
```

- Ejecutar Simulación

```
=====
2021-04-14 19:56:46                               60s elapsed
---- Requests ----
> Global (OK=393 KO=0 )
> request_0 (OK=131 KO=0 )
> request_1 (OK=131 KO=0 )
> request_2 (OK=131 KO=0 )

class BasicSimulation extends Simulation {
---- GetUsersList ----
[#####]100%
    waiting: 0 / active: 0 / done: 131
=====

Simulation GetUsersList completed in 60 seconds
```



Gatling: Informes

Se generan dos tipos de informes:

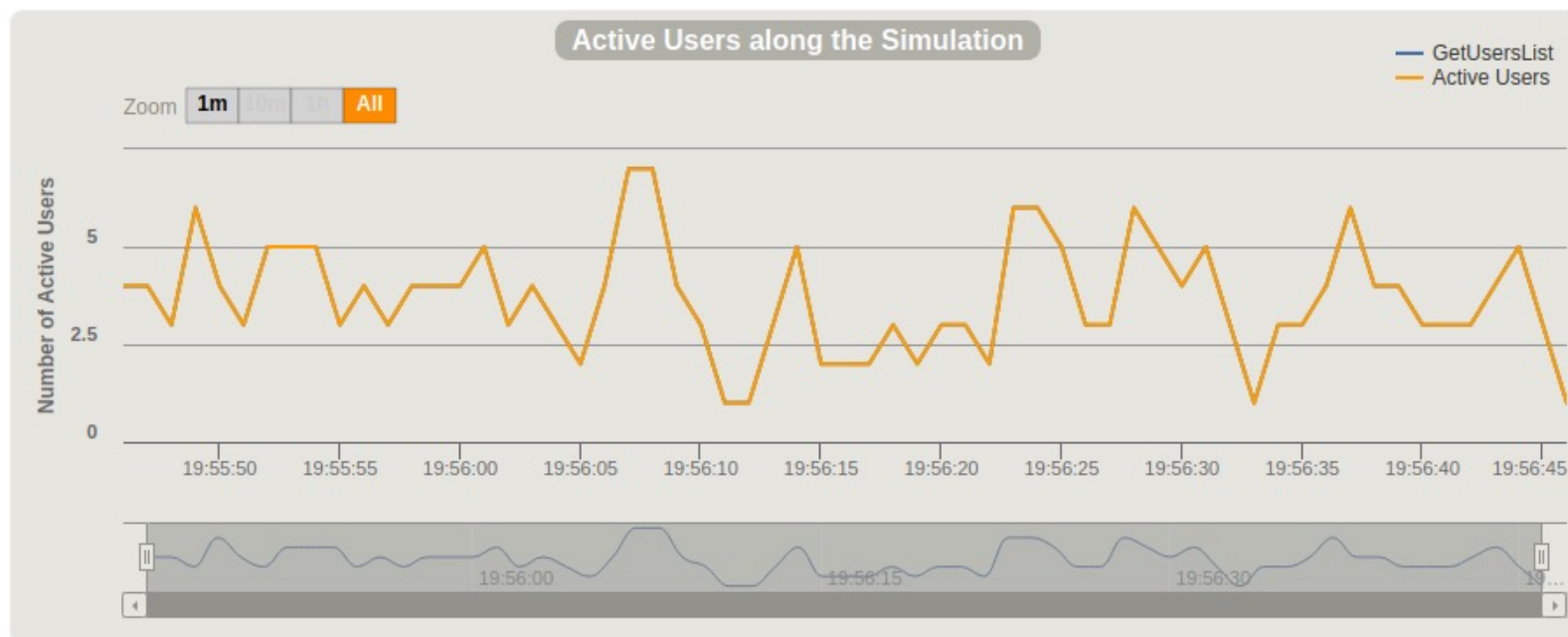
- Un informe de texto por consola

```
=====
---- Global Information ----
> request count                393 (OK=393   KO=0   )
> min response time            65 (OK=65   KO=-   )
> max response time            716 (OK=716  KO=-   )
> mean response time           256 (OK=256  KO=-   )
> std deviation                131 (OK=131  KO=-   )
> response time 50th percentile 302 (OK=302  KO=-   )
> response time 75th percentile 348 (OK=348  KO=-   )
> response time 95th percentile 433 (OK=433  KO=-   )
> response time 99th percentile 483 (OK=483  KO=-   )
> mean requests/sec            6.443 (OK=6.443 KO=-   )
---- Response Time Distribution ----
> t < 800 ms                   393 (100%)
> 800 ms < t < 1200 ms         0 ( 0%)
> t > 1200 ms                   0 ( 0%)
> failed                        0 ( 0%)
=====
```


Gatling: Informes

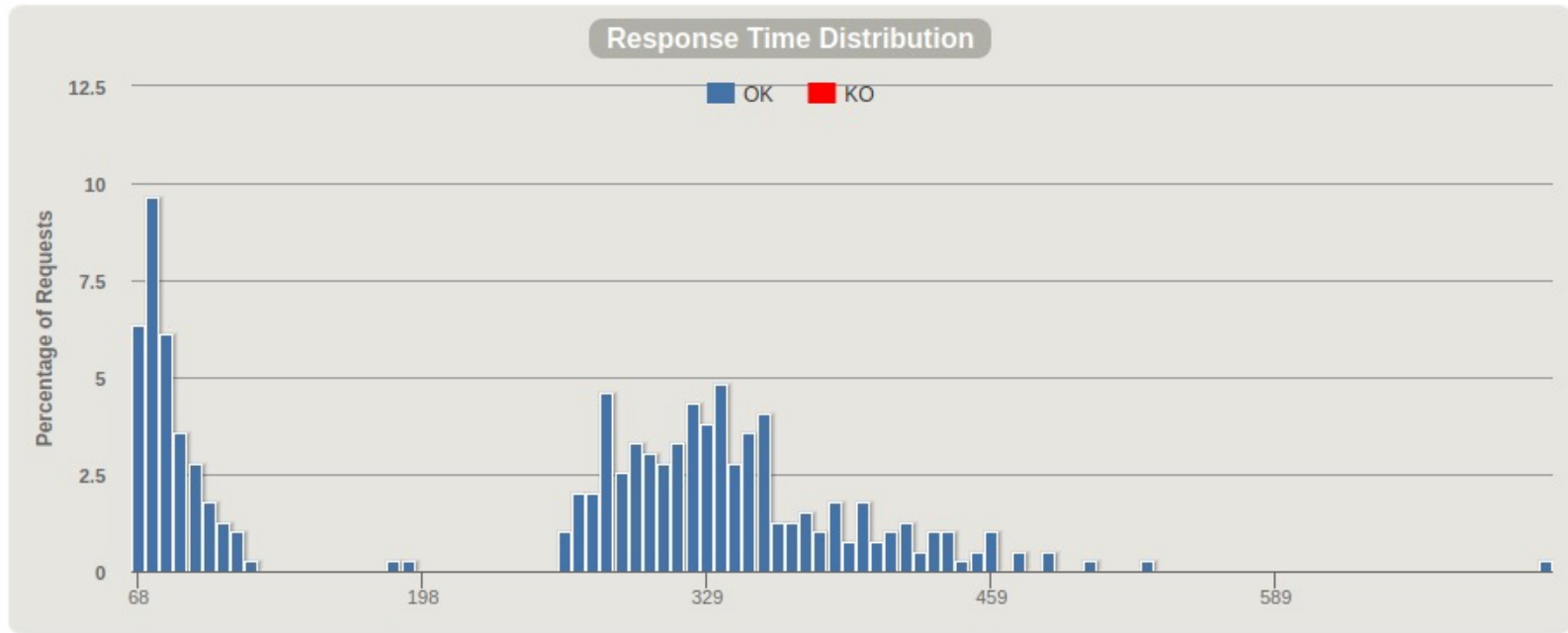
- Un informe HTML con más detalles:





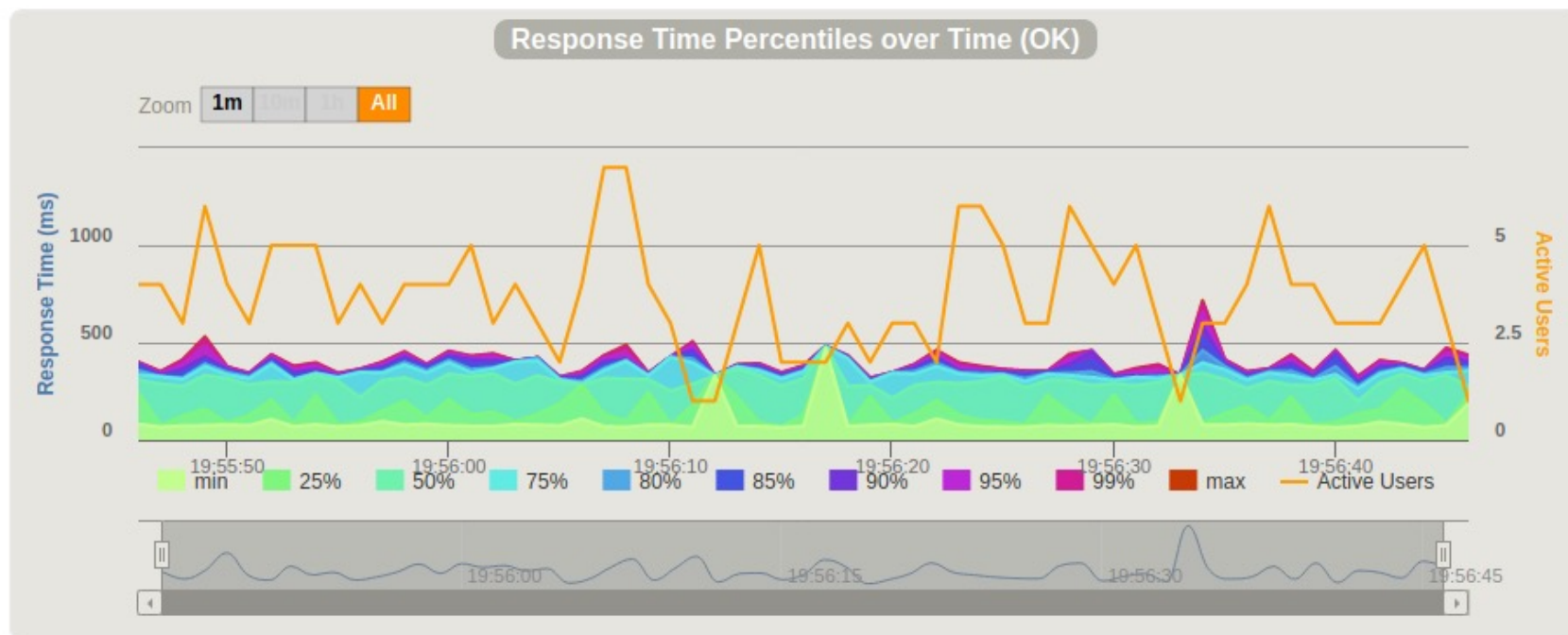
Usuarios activos durante la simulación

Muestra el número de usuarios activos (enviando peticiones y recibiendo respuestas) a lo largo del tiempo de simulación. Esta medida puede relacionarse con otras como tiempos de respuesta y número de peticiones.



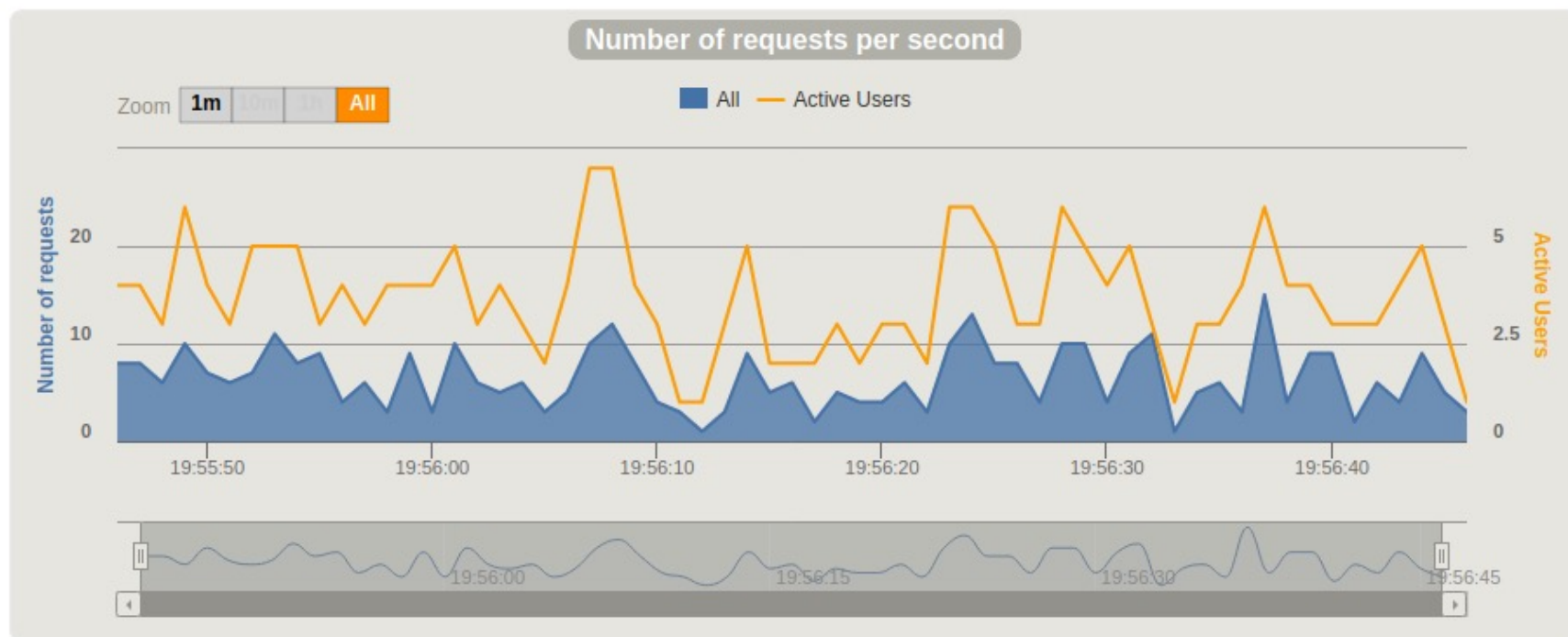
Distribución de tiempos de respuesta

El gráfico muestra el porcentaje de peticiones aceptadas (eje Y) durante la ejecución del test. Incluye tanto peticiones aceptadas como fallos. La totalidad de los valores de Y deben sumar 100%. El tiempo de respuesta (tiempo que le lleva a la página solicitar la petición, enviar los datos al servidor para confirmar que los recibió) está en el eje X. Cuanto más se incrementa la carga en el servidor las barras se desplazarán a la derecha del gráfico, indicando que los tiempos de respuesta son mayores.



Percentiles de tiempos de respuesta en el tiempo

Parecido a la distribución de tiempos de respuesta, pero muestra los datos en un mayor periodo de tiempo para que se pueda evaluar el comportamiento del sistema en un escenario de carga sostenida. Por ejemplo 200 usuarios accediendo a diferentes páginas web durante 5 minutos.



Peticiones/respuestas por segundo

Número de veces que se hace una petición a un recurso del servidor por segundo. Por ejemplo, simular 200 usuarios accediendo a un archivo del servidor al mismo tiempo, se obtendrá 200 peticiones/respuestas por segundo.

Otras pruebas

- Usabilidad

Permiten determinar si una aplicación es fácil de usar.

Evalúan la experiencia del usuario antes (formativas) y después (sumativas) de la puesta en producción.

Entre las características que se pueden medir están:

- Facilidad de aprendizaje y memorización
- Precisión y completitud de las tareas
- Eficiencia y productividad (tiempo en realizar la tarea)
- Errores
- Satisfacción
- Accesibilidad

Las técnicas de pruebas incluyen observación, benchmarking, encuestas, entrevistas, cuestionarios, eye-tracking..

Otras pruebas

Seguridad

Permiten determinar las características de seguridad del sistema.

Se realizan auditorías de seguridad y hacking 'ético'.

Informe de vulnerabilidades y posibles soluciones.

Herramientas open source: Wapiti, Zed Attack Proxy, Vega, W3af, Skipfish, Ratproxy, SQLMap, Wfuzz, Grendel-Scan, Arachni, Grabber.

Escalabilidad, mantenibilidad, portabilidad.. 😊💧

Enlaces de interés

- **Gatling** <https://gatling.io/>
 - **The Art of Destroying Your Web App With Gatling**
<https://gatling.io/2018/03/07/the-art-of-destroying-your-web-app/>
 - **The Scala Programming Language** (<https://www.scala-lang.org/>)
 - **Refactoring (Advanced Gatling-Scala)**
https://gatling.io/docs/2.3/advanced_tutorial#advanced-tutorial
<https://github.com/gatling/gatling/tree/master/gatling-bundle/src/main/scala/computerdatabase>
 - **Testing Node.Js Application with Gatling**
<https://blog.knoldus.com/testing-node-js-application-with-gatling/>
 - **Step by step guide**
https://github.com/pglez82/docker_solid_example/tree/pglez82-gatling-load-tests#load-tests-gatling
- **Otras Pruebas**
 - **Tipos de pruebas de software**
<http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html>
 - **Qué son: Pruebas de usabilidad (Andrea Cantú)**
<https://blog.acantu.com/que-son-pruebas-usabilidad/>
 - **An overview on usability testing & 6 tools to automate it**
<https://www.cubettech.com/blog/an-overview-on-usability-testing-6-tools-to-automate-it/>
 - **“Solución automatizada de pruebas de penetración y auditoría de seguridad para entornos de prestación de servicios empresariales en Cloud”**
David Lorenzo González, TFG (Universidad de Oviedo)

Presentación de la práctica

- TG para ensayar presentación
- Semana del 3 de Mayo
- 15-20 minutos
 - Presentación
 - Demo (video y en vivo)
 - Preguntas
- Escogemos quien habla
- Nota: 70% trabajo equipo + 30% trabajo personal