



# ARQUITECTURA DEL SOFTWARE

2023-24

Jose Emilio Labra Gayo

Pablo González

Cristian Augusto Alonso

Jorge Álvarez Fidalgo



Escuela de  
Ingeniería  
Informática



Universidad de Oviedo

## Laboratorio 9

Pruebas de carga

Otras pruebas

# ¿Qué son las pruebas de carga?

- Medir el rendimiento de manera anticipada ante la carga normal o un pico de carga.
  - Ejemplo: Varios usuarios concurrentes
- Objetivo: Anticiparnos a posibles fallos.
  - Verificar la carga de trabajo de un sistema



## ¿Qué permiten probar?

- Aplicaciones web (HTTP/HTTPS).
- Servicios Web SOAP/REST.
- FTP.
- Bases de datos (JDBC).
- LDAP.
- Mail (SMTP, POP3, IMAP).
- Objetos 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 atributos de calidad pactados en el contrato.

## Herramientas

- **Gatling.**
- Apache JMeter.
- Locust.io.
- Artillery.io.
- GoReplay.
- Loader.io
- BlazeMeter
- Blitz...

Guia paso a paso:

[https://github.com/pglez82/asw2122\\_o/tree/master/webapp#load-testing-gatling](https://github.com/pglez82/asw2122_o/tree/master/webapp#load-testing-gatling)

# Gatling

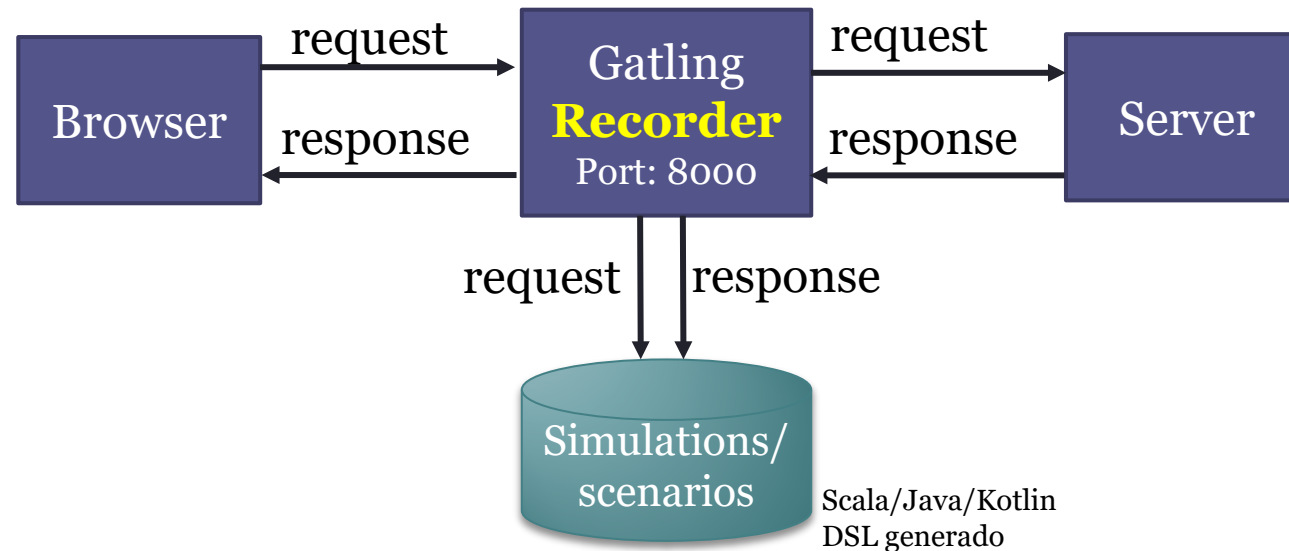
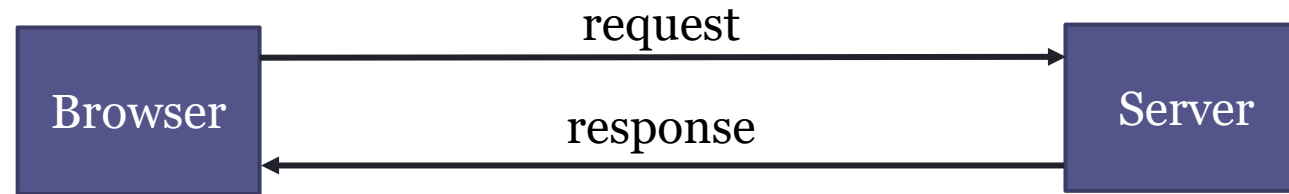
- Escrita en Scala.
- Compatible con la JVM.
- Uso de un DSL propio.
- Fácil de usar.
- Ligera.



# Descarga e instalación

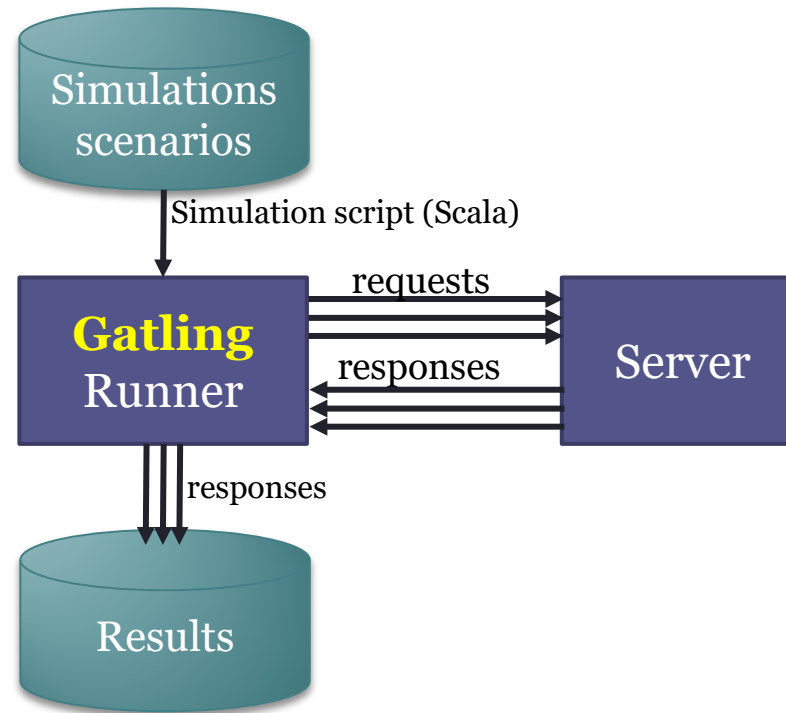
- <https://gatling.io>
  - Gatling soporta las versiones 11, 17 y 21 de 64 bits del OpenJDK LTS (Soporte a largo plazo)
- **Dos scripts:**
  - Recorder.sh/Recorder.bat
  - Gatling.sh/Gatling.bat

# Gatling recorder

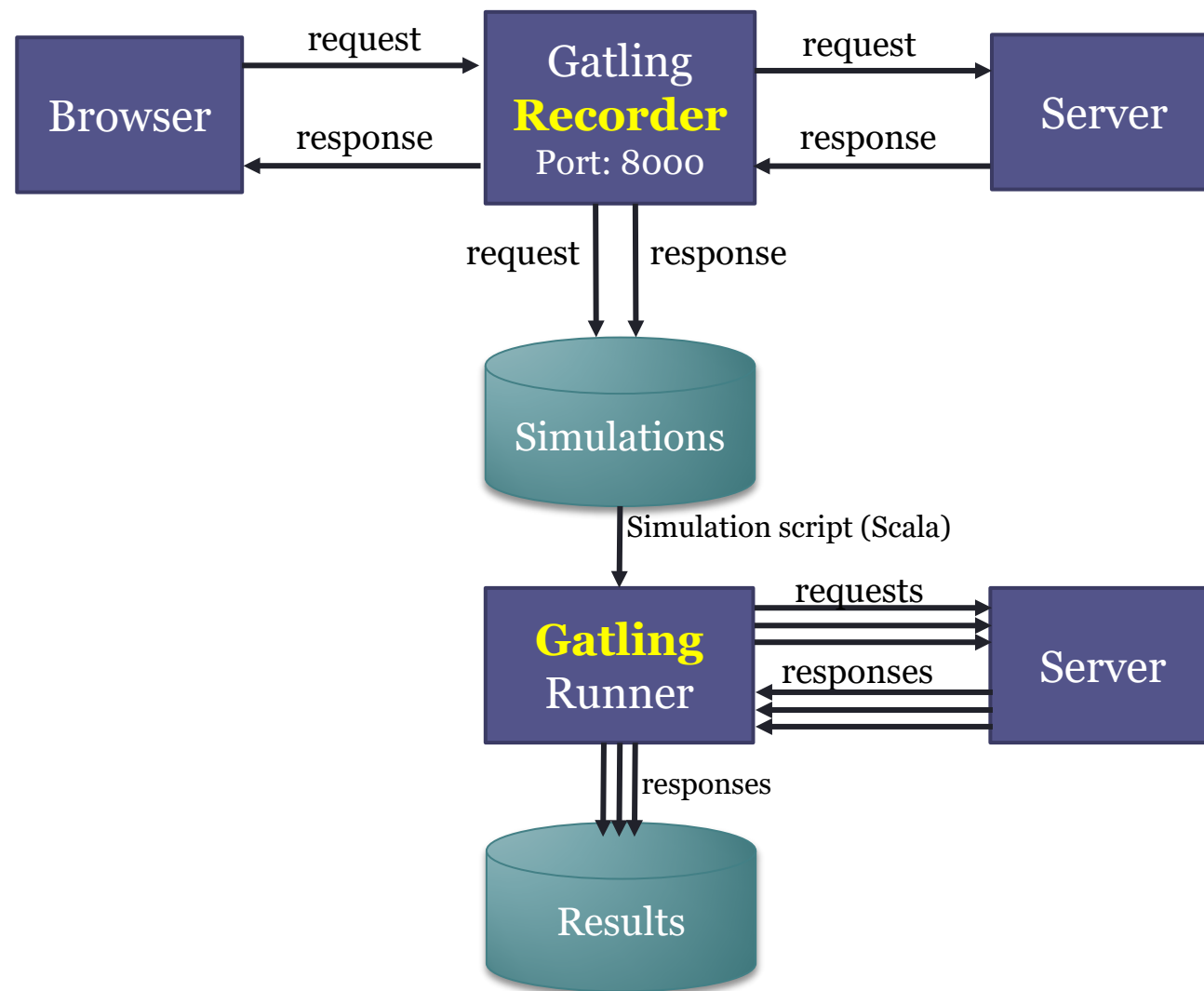




# Gatling runner



# Workflow



# Gatling: Recorder

## Test case: Wiq

Lanzar recorder:

```
(base) pablo@ZenBookUX431DA:~/Programas/gatling-charts-highcharts-bundle-3.10.5/bin$ ./recorder.sh
GATLING_HOME is set to /home/pablo/Programas/gatling-charts-highcharts-bundle-3.10.5
```

## Configuración Recorder:

- Generar los certificados.
- Importar los certificados a Firefox.
- Configurar el puerto.
- Otras configuraciones:
  1. Package: nombre del paquete.
  2. Name: nombre de la simulación.
  3. Follow Redirects
  4. Automatic Referers
  5. Strategy: primero *Blacklist*.
  6. Blacklist : `*\css, *\js, etc.`

# Configurar Proxy

localhost:8000

Para todas las direcciones, incluida localhost.

Si se usa HTTPS hay que configurar el certificado.



Recuerda restaurar el proxy después de cerrar la grabadora para tener acceso a internet

Para localhost en Firefox, ejecutar:  
`network.proxy.allow_hijacking_localhost to true in about:config`

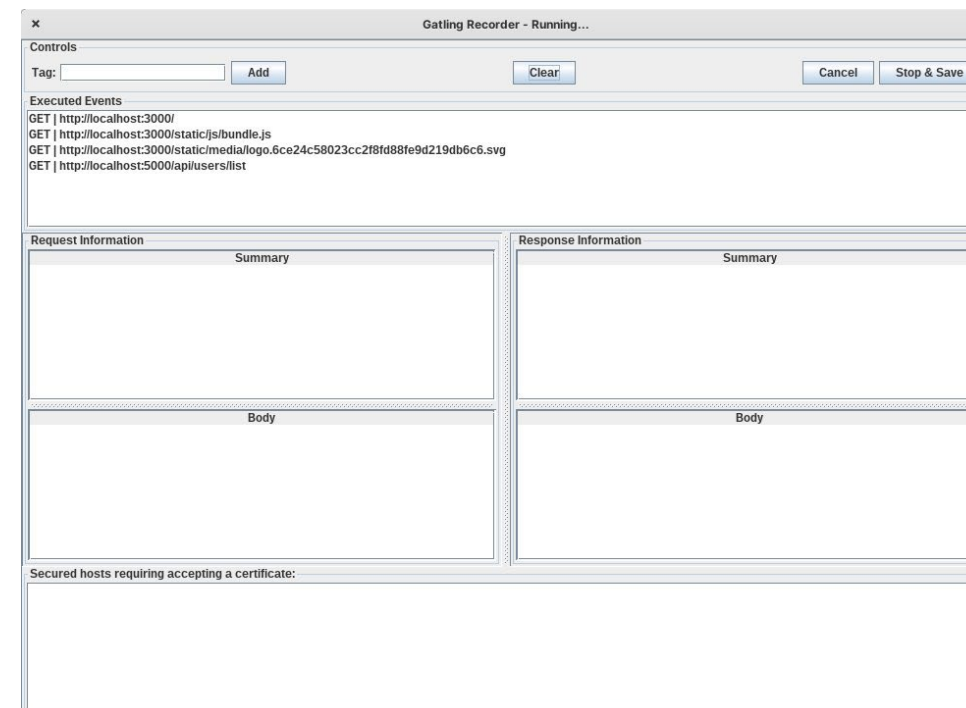
# Gatling: Recorder

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 simulación

## Consideraciones Previas:

- En este caso **sólo** hemos cargado la página principal de la aplicación.
- Tenga en cuenta la última línea de la prueba, podemos **ajustar la carga**.
- Obviamente, las pruebas pueden ser mucho más complicadas, realizando múltiples acciones en el sistema.
- También podemos **escribir nuestro propio código** sin usar la grabadora

# 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/gatling/reference/current/core/injection/>

## 2 usuarios por segundo durante 60 segundos

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

```
...
setUp(
    scn.injectOpen(constantUsersPerSec(2) during (60 seconds)
        randomized)
).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

```
(base) pablo@ZenBookUX431DA:~/Programas/gatling-charts-highcharts-bundle-3.10.5/bin$ ./gatling.sh
GATLING_HOME is set to /home/pablo/Programas/gatling-charts-highcharts-bundle-3.10.5
Do you want to run the simulation locally, on Gatling Enterprise, or just package it?
Type the number corresponding to your choice and press enter
[0] <Quit>
[1] Run the Simulation locally
[2] Package and upload the Simulation to Gatling Enterprise Cloud, and run it there
[3] Package the Simulation for Gatling Enterprise
[4] Show help and exit
1
Choose a simulation number:
  [0] computerdatabase.ComputerDatabaseSimulation
  [1] wiq.LoginUser
```

- Salida Simulación

```
=====
2024-03-31 15:01:46 GMT                                     63s elapsed
---- Requests -----
> Global (OK=455 KO=0 )
> get_index_page (OK=91 KO=0 )
> get_javascript (OK=91 KO=0 )
> get_css (OK=91 KO=0 )
> get_login_page (OK=91 KO=0 )
> login_post (OK=91 KO=0 )

---- LoginUser -----
[#####]100%
      waiting: 0      / active: 0      / done: 91
=====
```

# Gatling: Informes

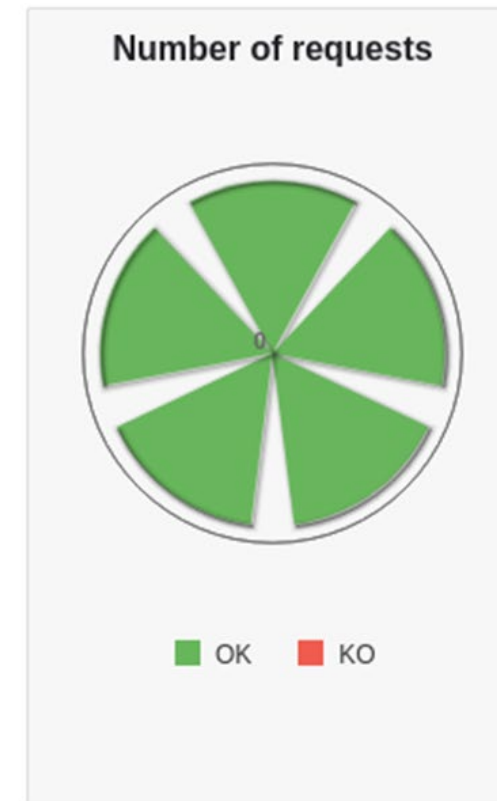
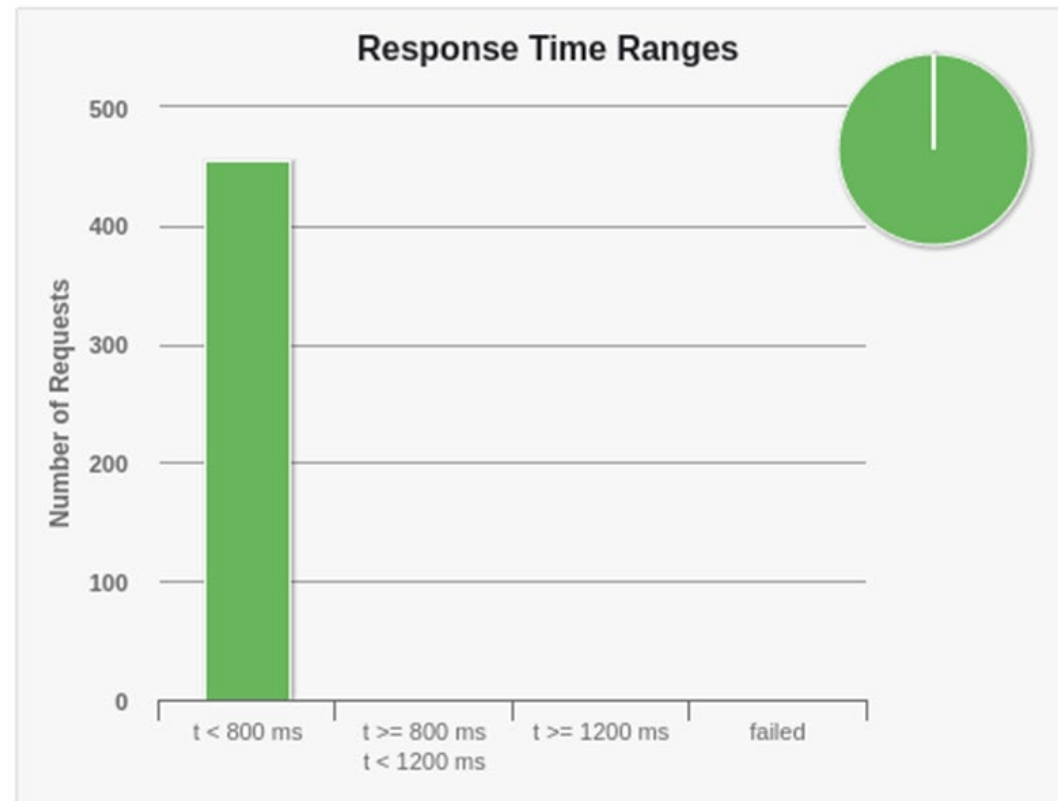
Se generan dos tipos de informes:

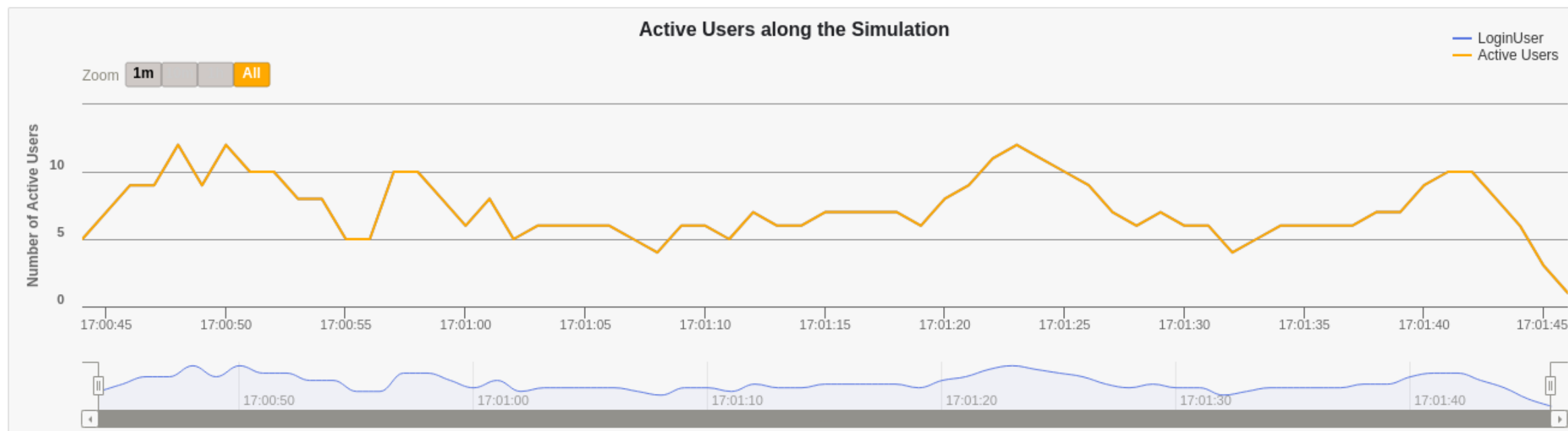
- Un informe de texto por consola.

```
=====
---- Global Information -----
> request count                455 (OK=455   KO=0    )
> min response time           114 (OK=114   KO=-   )
> max response time           440 (OK=440   KO=-   )
> mean response time          226 (OK=226   KO=-   )
> std deviation                64 (OK=64    KO=-   )
> response time 50th percentile 234 (OK=234   KO=-   )
> response time 75th percentile 243 (OK=243   KO=-   )
> response time 95th percentile 342 (OK=342   KO=-   )
> response time 99th percentile 407 (OK=407   KO=-   )
> mean requests/sec           7.222 (OK=7.222 KO=-   )
---- Response Time Distribution -----
> t < 800 ms                   455 (100%)
> 800 ms <= t < 1200 ms       0 ( 0%)
> t >= 1200 ms                 0 ( 0%)
> failed                       0 ( 0%)
=====
```

# Gatling: Informes

- Un informe HTML con más detalles:

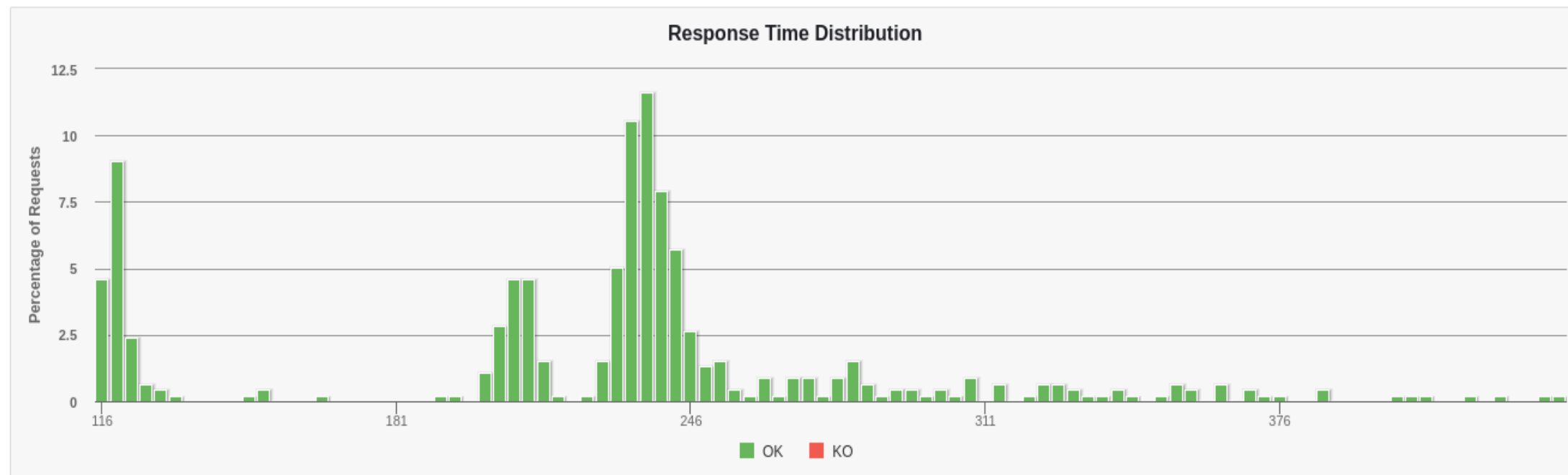




## Usuarios activos durante la simulación

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 o el nº de peticiones.**



## Distribución de tiempos de respuesta

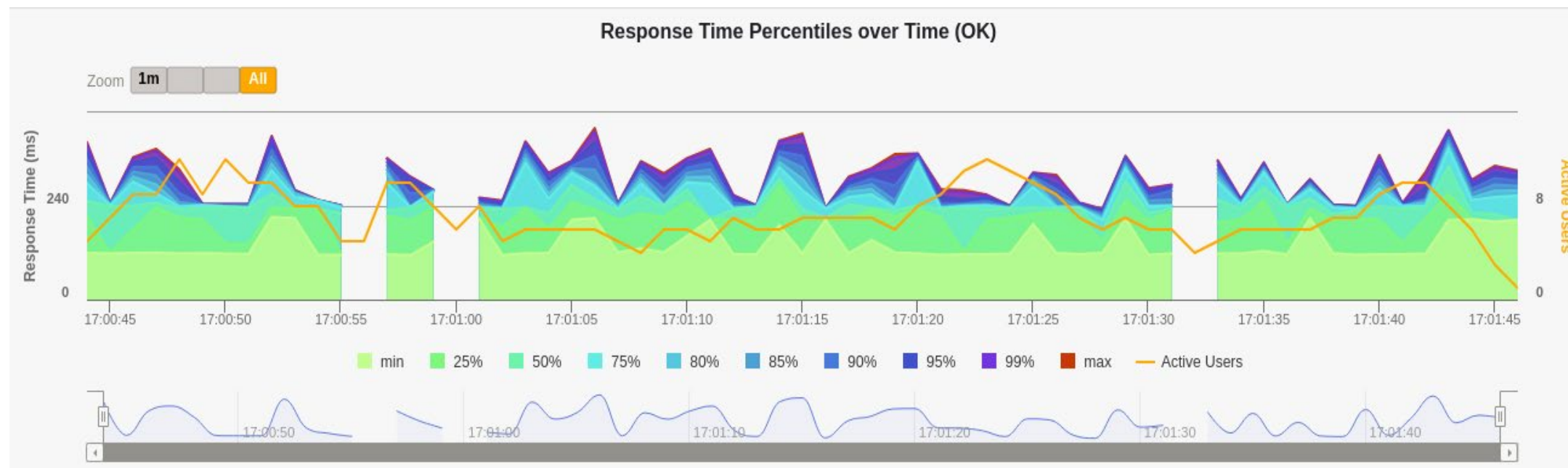
El gráfico muestra el porcentaje de peticiones aceptadas (eje Y) durante la ejecución del test.

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.

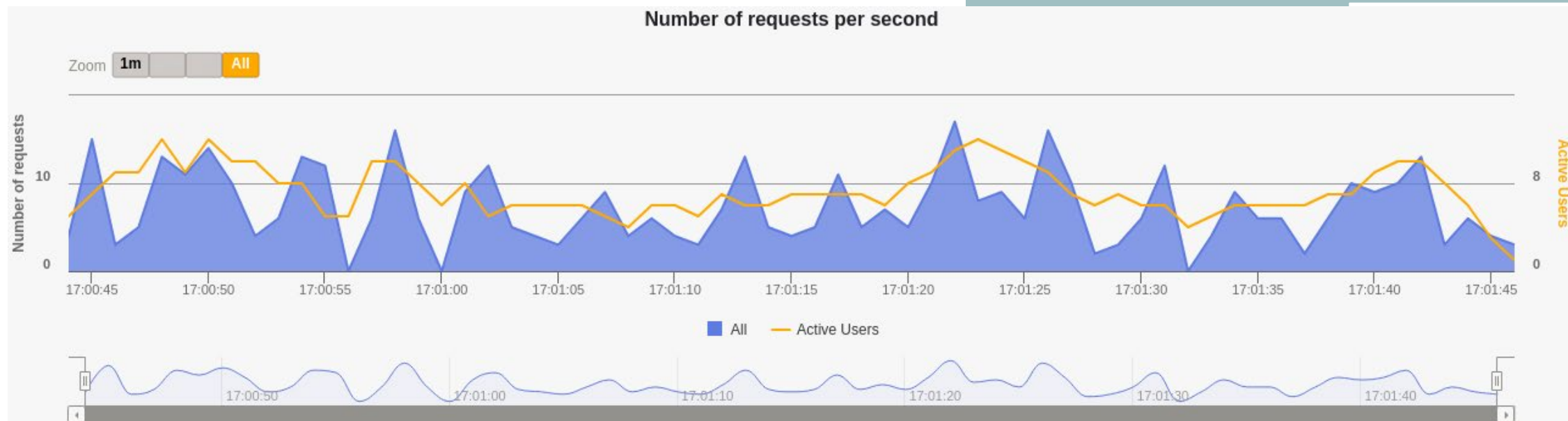
Cuanto más se incrementa la carga en el servidor las barras se desplazarán a la derecha del gráfico, indicando que los tiempos de respuesta son mayores.



## 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.



# Conceptos de Gatling & DSL

- Simulación: descripción de un test de carga.
  - Define el método setUp
  - Scenario: representa el comportamiento de los usuarios.
- Es posible inyectar usuarios en escenarios.
  - Varias posibilidades:
    - nothingFor
    - atOnceUsers
    - rampUsers
    - constantUsersPerSec
    - ...
  - Protocolos: indicar definiciones de protocolo (usualmente. http)
  - Assertions: verificar algunas estadísticas.
- Se puede usar para integración continua.

# 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/)

<https://gatling.io/2018/03/07/the-art-of-destroying-your-web-app/>

[The Scala Programming Language \(https://www.scala-lang.org/\)](https://www.scala-lang.org/)

[Refactoring \(Advanced Gatling-Scala\)](https://gatling.io/docs/2.3/advanced_tutorial#advanced-tutorial)

[https://gatling.io/docs/2.3/advanced\\_tutorial#advanced-tutorial](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/)

<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)

[https://github.com/pglez82/docker\\_solid\\_example/tree/pglez82-gatling-load-tests#load-tests-gatling](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)

<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/)

<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)

[https://www.cubettech.com/blog/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)