Universidad de Oviedo

School of Computer Science

SOFTWARE ARCHITECTURE

Software Architecture

Lab. 11
Load testing
Other tests…

2021-22

Jose Emilio Labra Gayo
Pablo González
Irene Cid
Hugo Lebredo

What are load tests?

Measure performance under normal or anticipated peak load conditions

Example: Several concurrent users

Goal: Anticipate possible failures

verify work load of some system

What can we test

- Web applications (Http/https)
- SOAP/REST Web Services
- FTP
- Databases (JDBC)
- LDAP
- Mail (SMTP, POP3, IMAP)
- Java Objects
- Etc.

Why should we do load tests?

- Anticipate performance problems
- Detect bottlenecks
- Prove quality attributes

Load testing tools

Gatling

Apache Jmeter ()

Locust.io (http://locust.io/)

Artillery.io ()

goReplay

Loader.io

BlazeMeter

Blitz …

Step by step guide:

https://github.com/pglez82/asw2122_0/tree/master/webapp#load-testing-gatling

Gatling

Written in Scala

JVM compatible

Embedded DSL for testing

Easy to use
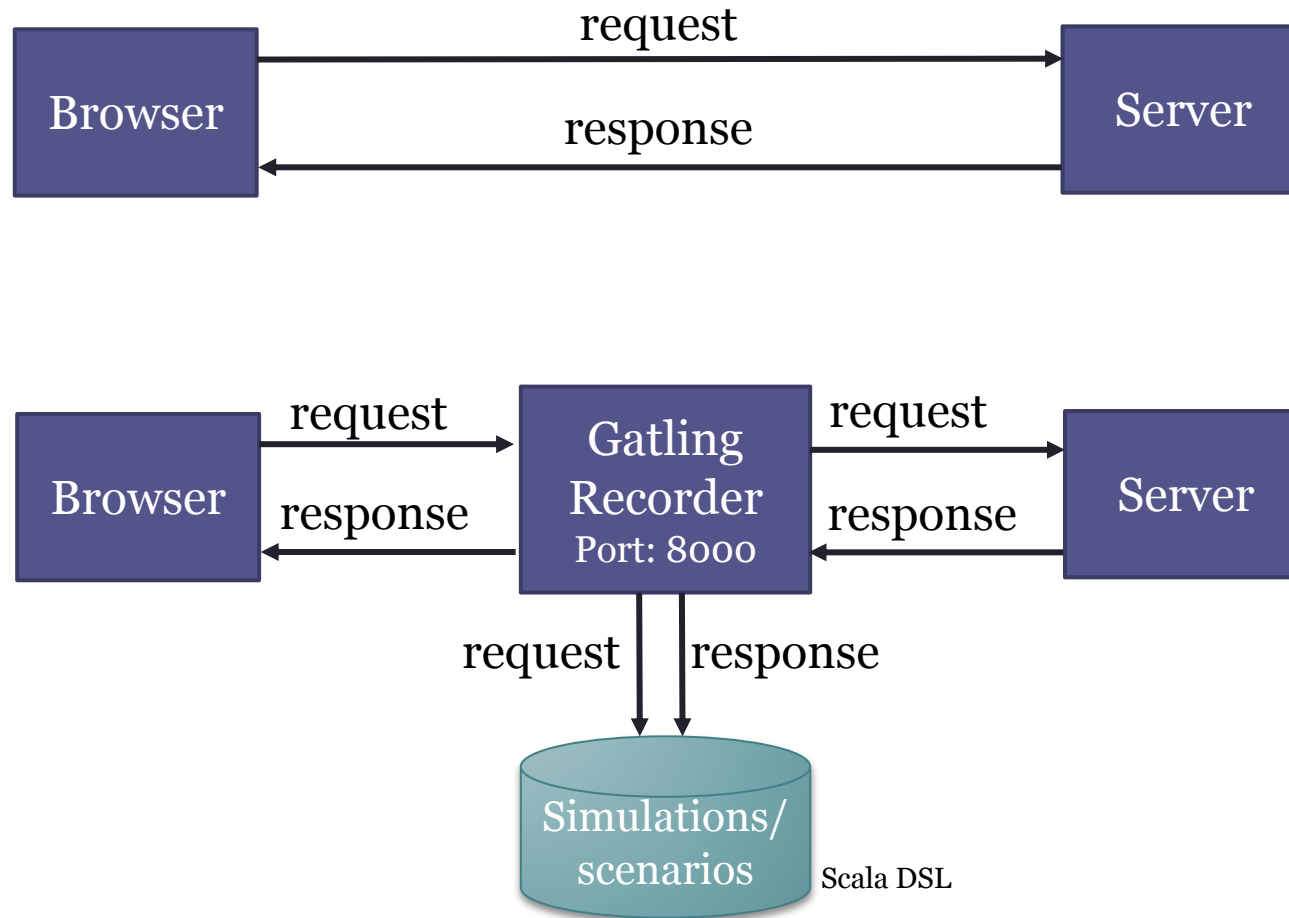
Light

Download & installation

http://gatling.io
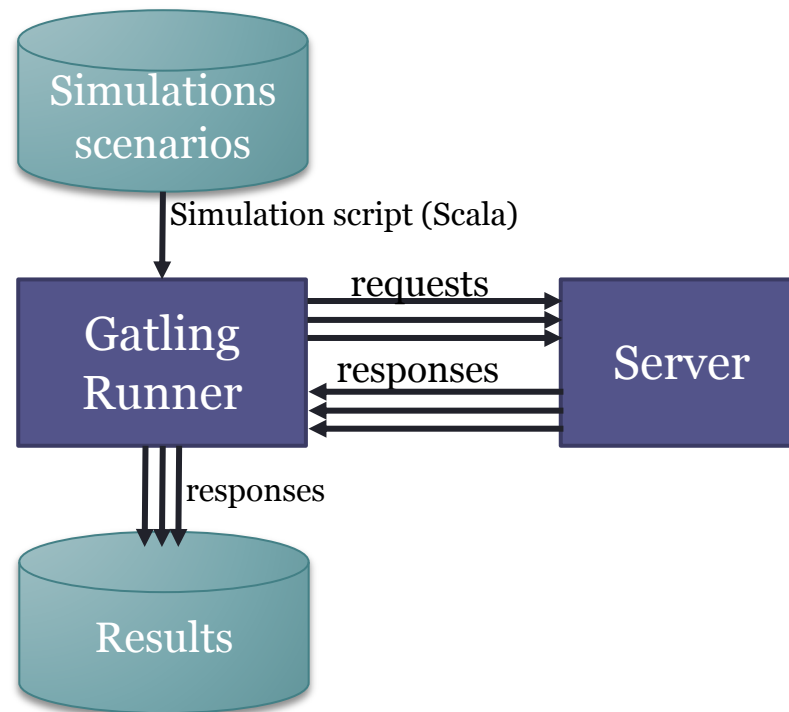
It needs Java 8 installed

2 scripts:
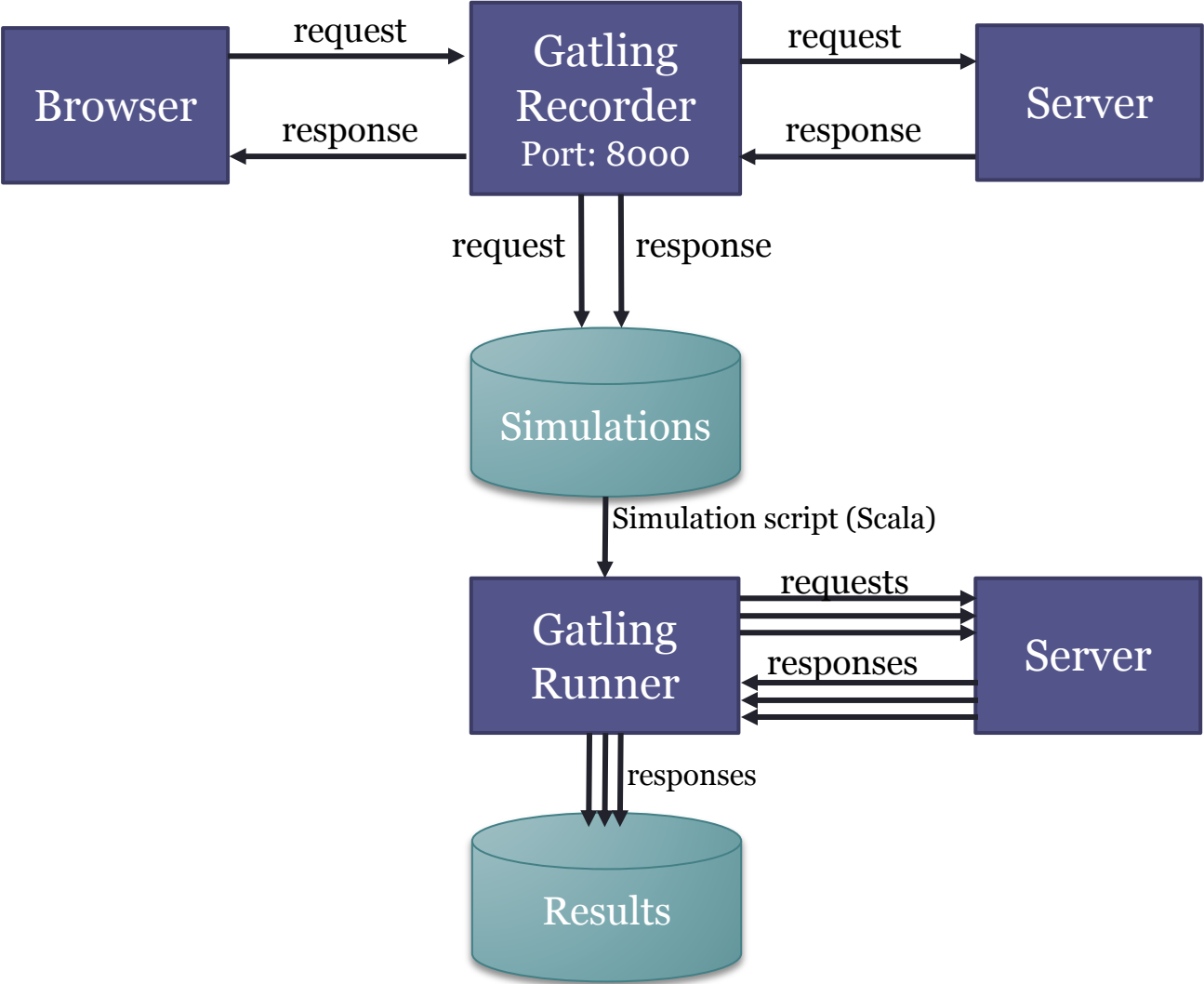
Recorder.sh/Recorder.bat

Gatling.sh/Gatling.bat

Gatling recorder

Gatling runner

Simulations scenarios

Simulation script (Scala)

Gatling Runner

requests

responses

Server

responses

Results

Workflow

Gatling: Recorder

Test case: dede

Launch recorder

```
pablo@ZenBookUX431DAUM431DAddc5ed2c:~/Programas/gatling-charts-highcharts-bundle-3.7.3/bin$ ./recorder.sh
GATLING_HOME is set to /home/pablo/Programas/gatling-charts-highcharts-bundle-3.7.3
```

Recorder setup

- Generate the certificates
- Import the certificate to Firefox
- Configure the port
- Other configuration:
    1. Package: packagename
    2. Name: SimulationName
    3. Follow Redirects ✅
    4. Automatic Referers ✅
    5. Strategy: Black list first
    6. Blacklist: .*\.css, .*\.js, etc

Configure Proxy

`localhost:8000`

For all addresses, included localhost

In case of HTTPS, the certificate must be configured

Configuración de conexión ✕

Configurar proxies para el acceso a Internet

- ○ Sin proxy
- ○ Autodetectar configuración del proxy para esta red
- ○ Usar la configuración del proxy del sistema
- ● Configuración manual del proxy:

Proxy HTTP:	localhost	Puerto:
✔ Usar el mismo proxy para todo		
Proxy SSL:	localhost	Puerto: 8000
Proxy FTP:	localhost	Puerto: 8000
Servidor SOCKS:	localhost	Puerto: 8000
○ SOCKS v4		

No usar proxy para:

Remember to restore proxy after closing the recorder to have internet access

For localhost in Firefox, set:
`network.proxy.allow_hijacking_localhost` to true in `about:config`

Gatling: Recorder

Browser > Web Proxy > localhost:8000

Recorder: Start

- After starting, open the website and perform the actions that you want to be part of the test
- After finishing press Stop
- Actions will be recorded in **Scala** language
- The simulation will be saved under the directory *user-files/simulations*
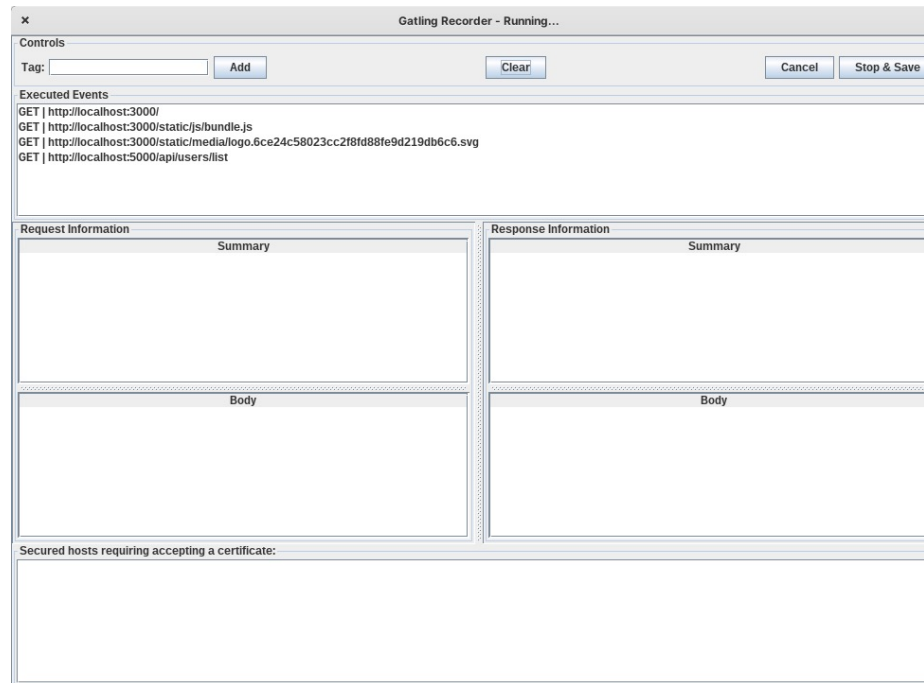
Simulation Example

- In this case we only have loaded the main page of the application and added one user
- Note the last line of the test, we can adjust the load here
- Obviously, tests can be much more complicated, performing multiple actions in the system

  https://github.com/pglez82/asw2122_0/blob/master/webapp/loadtestexample/GetUsersList.scala

How-to configure the number of users…

Injection profile
Control how users are injected in your scenario

Injection steps

nothingFor	constantUsersPerSec
atOnceUsers	rampUsersPerSec
rampUsers	splitUsers
	heavisideUsers

2 users per second during 60 seconds

- 120 users arriving at the rate of 2 users/second
- They execute a given script

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

Triggering Gatling

Run script: `gatling.sh/.bat`

  choose the class with the previous script

  Configure ID and description

In the execution we can see the textual progress

At the end, an HTML file is generated

  It contains graphical load test analysis

Triggering Gatling

Run Gatling (/bin/gatling.sh) and choose the scenario

```
pablo@ZenBookUX431DAUM431DAddc5ed2c:~/Programas/gatling-charts-highcharts-bundle-3.7.3/bin$ ./gatling.sh
GATLING_HOME is set to /home/pablo/Programas/gatling-charts-highcharts-bundle-3.7.3
Choose a simulation number:
     [0] computerdatabase.BasicSimulation
     [1] computerdatabase.advanced.AdvancedSimulationStep01
     [2] computerdatabase.advanced.AdvancedSimulationStep02
     [3] computerdatabase.advanced.AdvancedSimulationStep03
     [4] computerdatabase.advanced.AdvancedSimulationStep04
     [5] computerdatabase.advanced.AdvancedSimulationStep05
     [6] dede.UserList1
```

Simulation output

```
================================================================================
2022-04-13 15:26:34                                            19s elapsed
---- Requests ------------------------------------------------------------------
> Global                                            (OK=315     KO=0      )
> request_0                                         (OK=45      KO=0      )
> request_3                                         (OK=45      KO=0      )
> request_1                                         (OK=45      KO=0      )
> request_2                                         (OK=45      KO=0      )
> request_4                                         (OK=45      KO=0      )
> request_5                                         (OK=45      KO=0      )
> request_6                                         (OK=45      KO=0      )

---- UserList1 -----------------------------------------------------------------
[#####################################################################]100%
          waiting: 0      / active: 0       / done: 45
================================================================================
```

Gatling: Reports

Two types of reports are generated:

- A text report in the console

```
================================================================================
---- Global Information --------------------------------------------------------
> request count                                          315 (OK=315     KO=0      )
> min response time                                        1 (OK=1       KO=-      )
> max response time                                       20 (OK=20      KO=-      )
> mean response time                                       4 (OK=4       KO=-      )
> std deviation                                            2 (OK=2       KO=-      )
> response time 50th percentile                            3 (OK=3       KO=-      )
> response time 75th percentile                            5 (OK=5       KO=-      )
> response time 95th percentile                            7 (OK=7       KO=-      )
> response time 99th percentile                           10 (OK=10      KO=-      )
> mean requests/sec                                    15.75 (OK=15.75   KO=-      )
---- Response Time Distribution ------------------------------------------------
> t < 800 ms                                             315 (100%)
> 800 ms < t < 1200 ms                                     0 (  0%)
> t > 1200 ms                                              0 (  0%)
> failed                                                   0 (  0%)
================================================================================
```
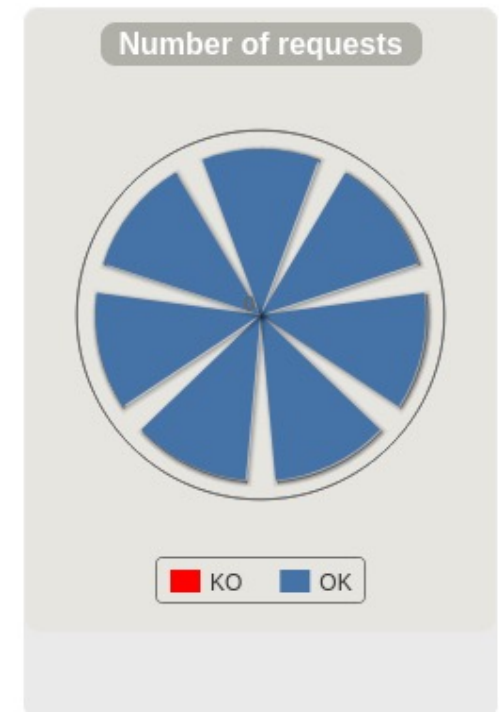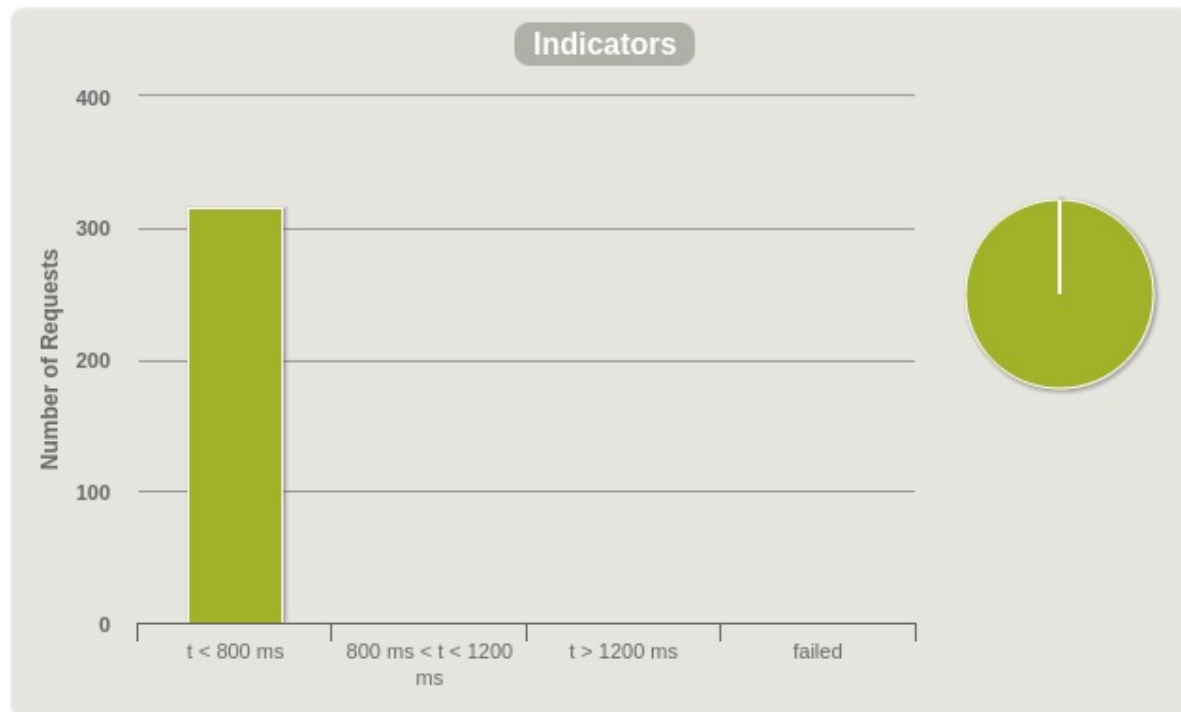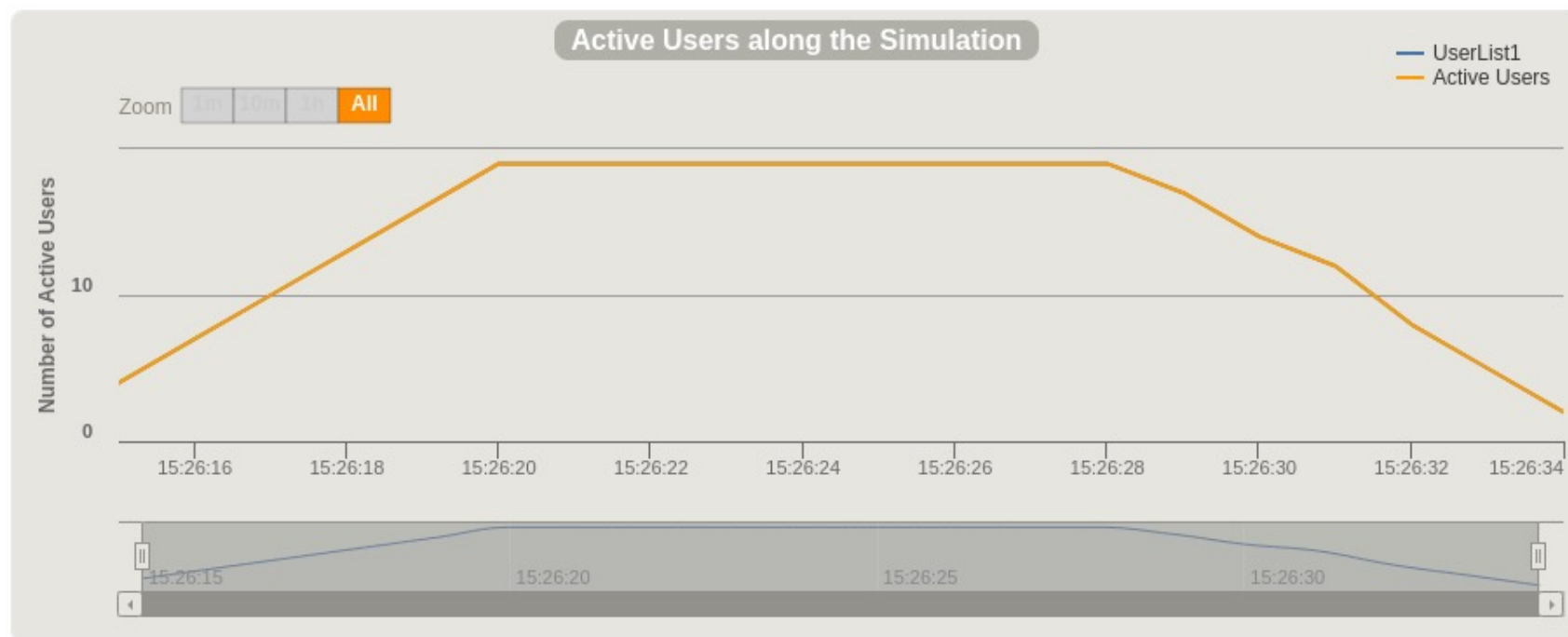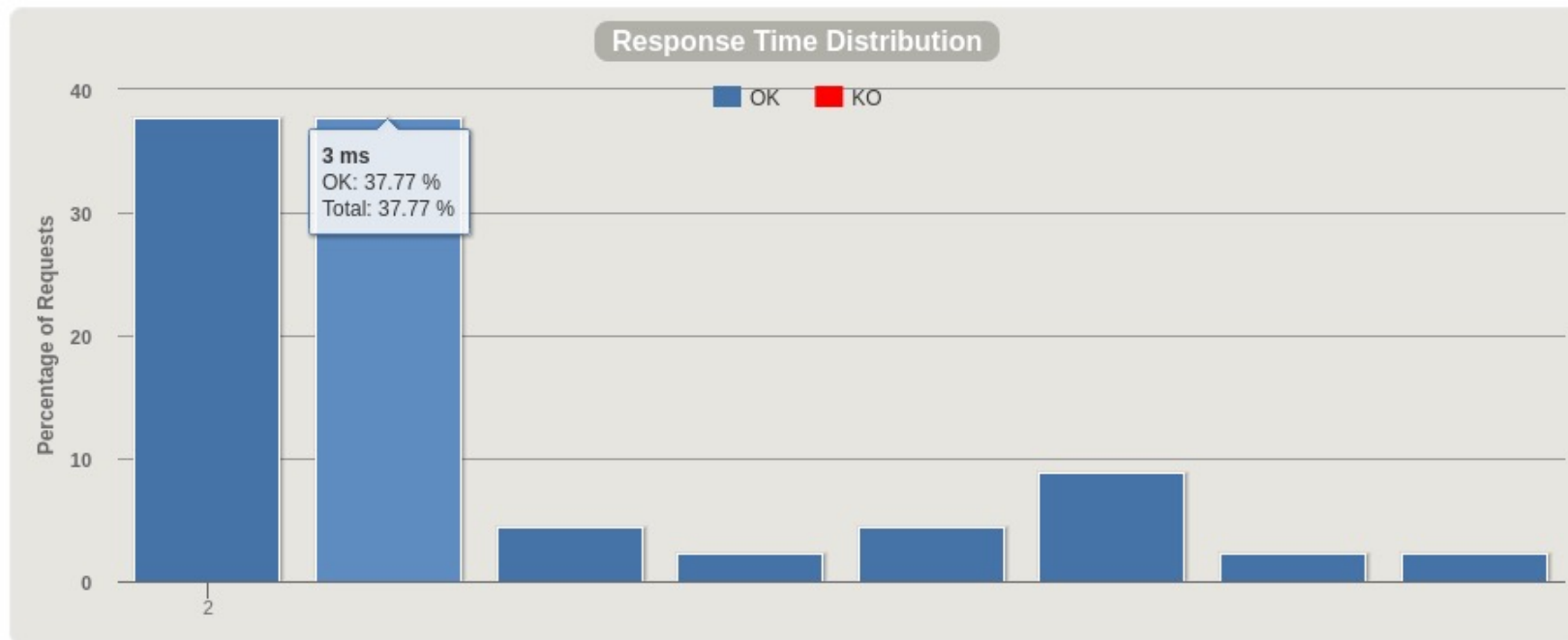
# Gatling: Reports

- An html (and more detailed) report:

Active Users along the Simulation

It displays the number of active users (sending requests and receiving responses) along the simulation time. This measure can be related to others such as response times and number of requests.

Response Time Distribution

This chart shows you the percentage of all requests made during your test run on the Y axis. It will include both successes and failures. All of the Y values should add up to 100%. The response time (the time it takes to request the page and send data back to the server to acknowledge you received it) is on the x axis. As you increase load on the server, you should see the data on this chart move farther to the right (response times will get slower).
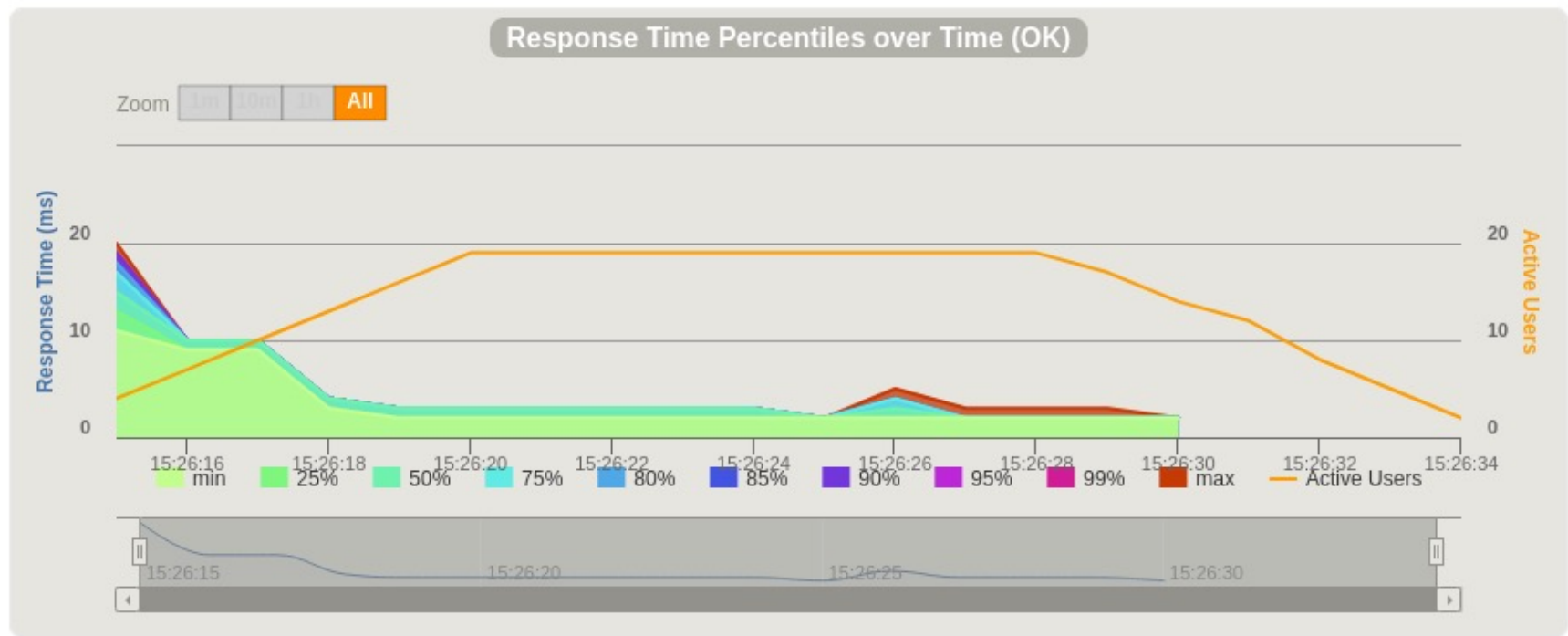
Response Time Percentiles over Time

This is similar to Response Time Distribution, but it shows you the data over a longer period of time to assess how your system behaves when under a sustained load. For example, 200 users accessing various web pages over the course of 5 minutes.
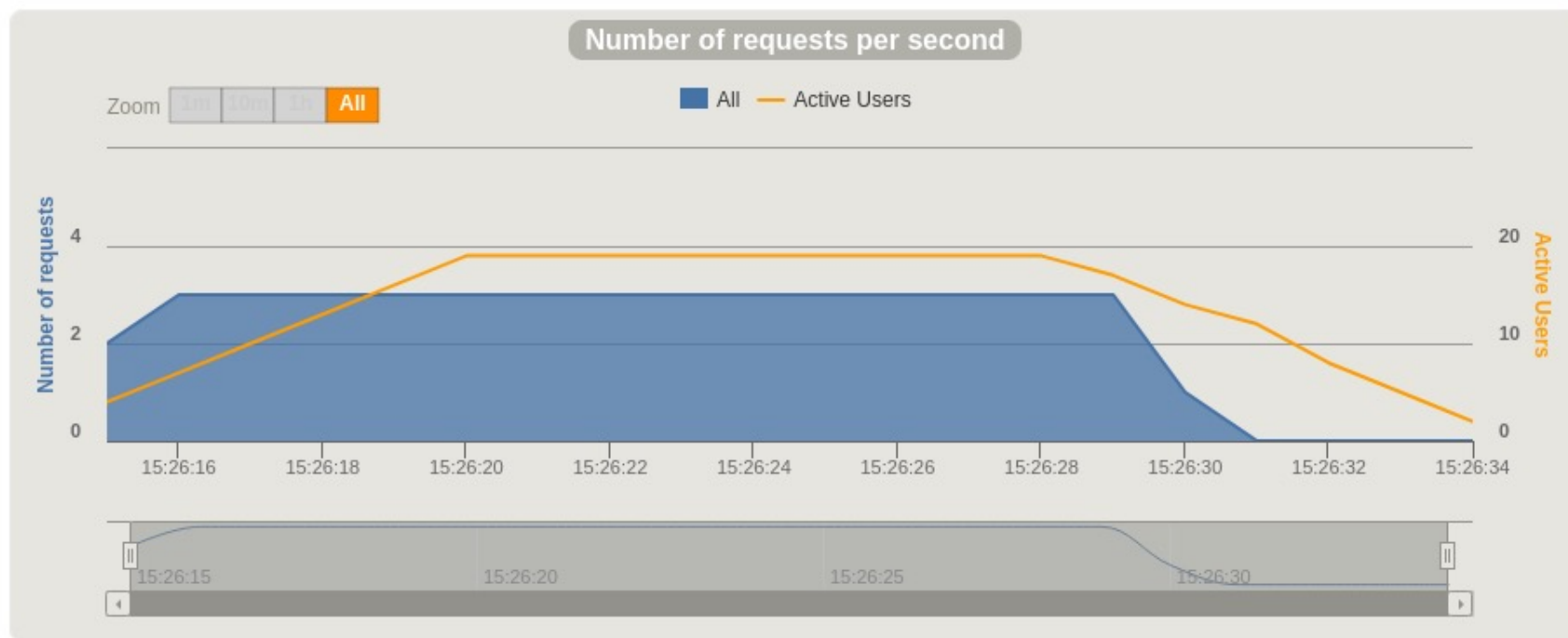
Requests/responses per second

The number of times you make a request for a resource from the server per second. For example, if you simulate 200 users accessing one file on a server all at the same time once a second, you'll have 200 requests/responses per second.

Gatling concepts & DSL

Simulation: Description of a load test

Defines method `setUp`

Scenario: Represents users' behaviours

It is possible to inject users to scenarios

Several possibilities:

nothingFor

atOnceUsers

rampUsers

constantUsersPerSec

...

Protocols: set protocol definitions (usually http)

Assertions: Verify some statistics

Can be used for continuous integration

Other tests

Usability

Allow to determine if a given application is easy to use. They assess users´ experience before (formative) and after (summative) the release of a given software.

Among the measures they can provide:

Ease of learning and memorising
Precision and completeness
Efficiency and productivity (time spent to perform a task)
Errors
Satisfaction
Accessibility

Testing techniques include observation, benchmarking, surveys, interviews, questionnaires, eye-tracking..

Other tests

Security

Allow measuring the level of security.
Ethical Hacking

Vulnerability reports and possible solutions

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

Scalability, maintainability, portability..

Links

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/

Other tests

Types of software testing
https://www.softwaretestinghelp.com/types-of-software-testing/
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, Trabajo fin de Grado (Universidad de Oviedo)