EN
English

Universidad de Oviedo

School of Computer Science

SOFTWARE ARCHITECTURE

Software Architecture

Lab. 11
Load testing
Other tests…

2020-21

Jose Emilio Labra Gayo
Pablo González
Irene Cid
Paulino Álvarez

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/docker_solid_example/tree/pglez82-gattling-load-tests#load-tests-gatling

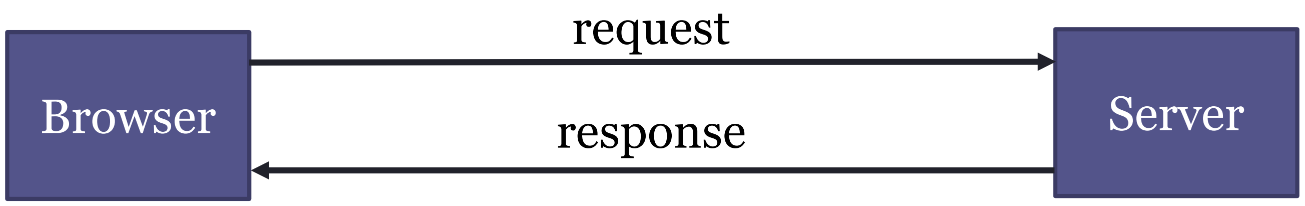
Gatling

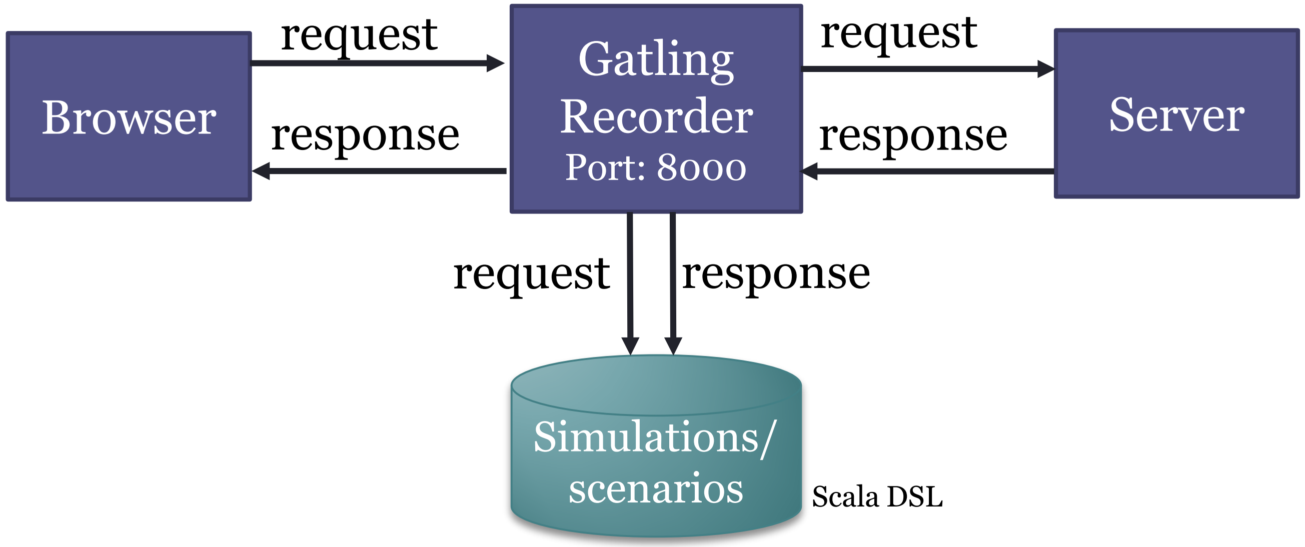
Written in Scala
JVM compatible
Embedded DSL for testing
Easy to use
Light

Download & installation

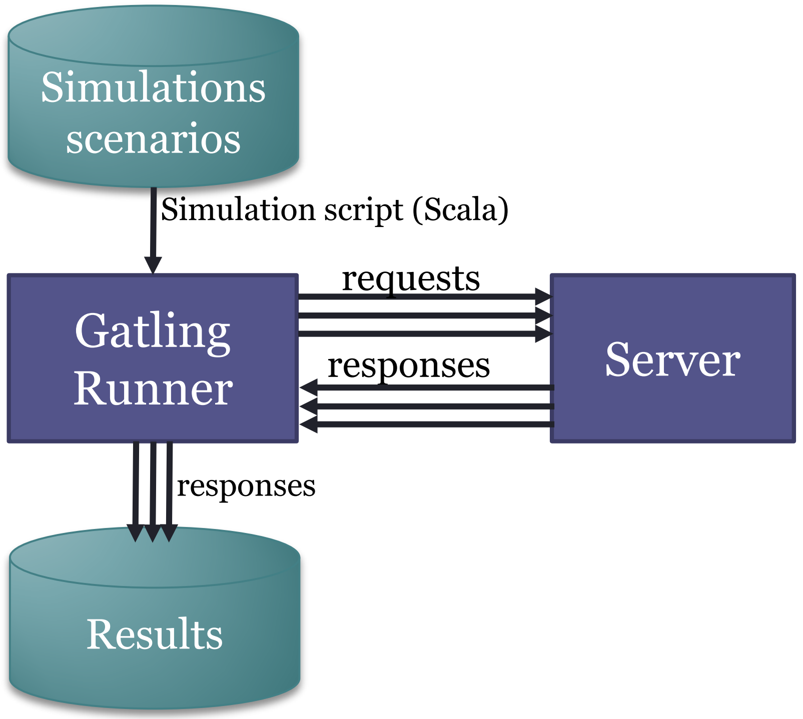
- http://gatling.io
  - It needs Java 8 installed
- 2 scripts:
  - Recorder.sh/Recorder.bat
  - Gatling.sh/Gatling.bat

Gatling recorder

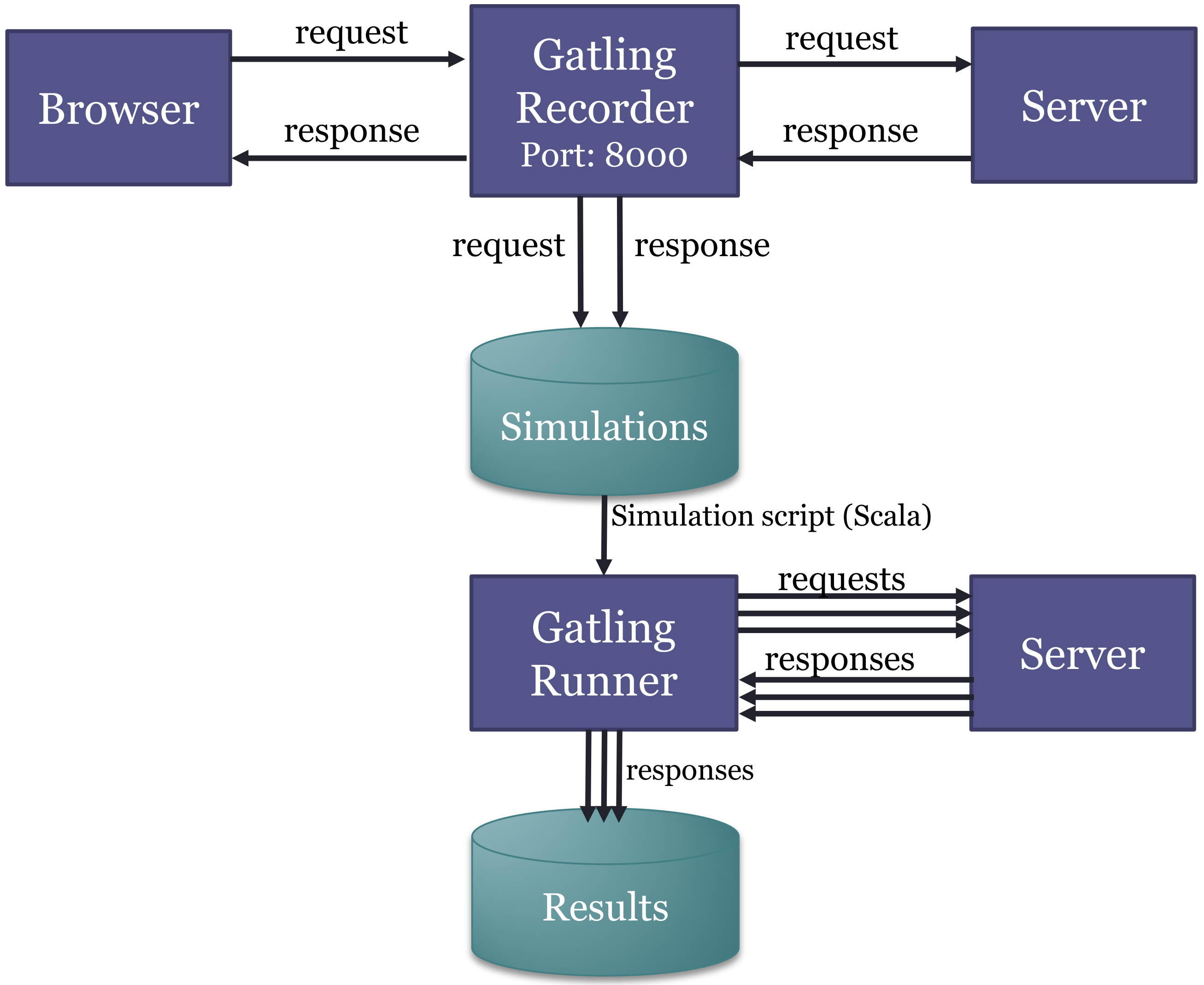




Gatling runner



Workflow


Browser
request
response
Gatling Recorder Port: 8000
request
response
Server
request
response
Simulations
Simulation script (Scala)
Gatling Runner
requests
responses
Server
responses
Results

Gatling: Recorder

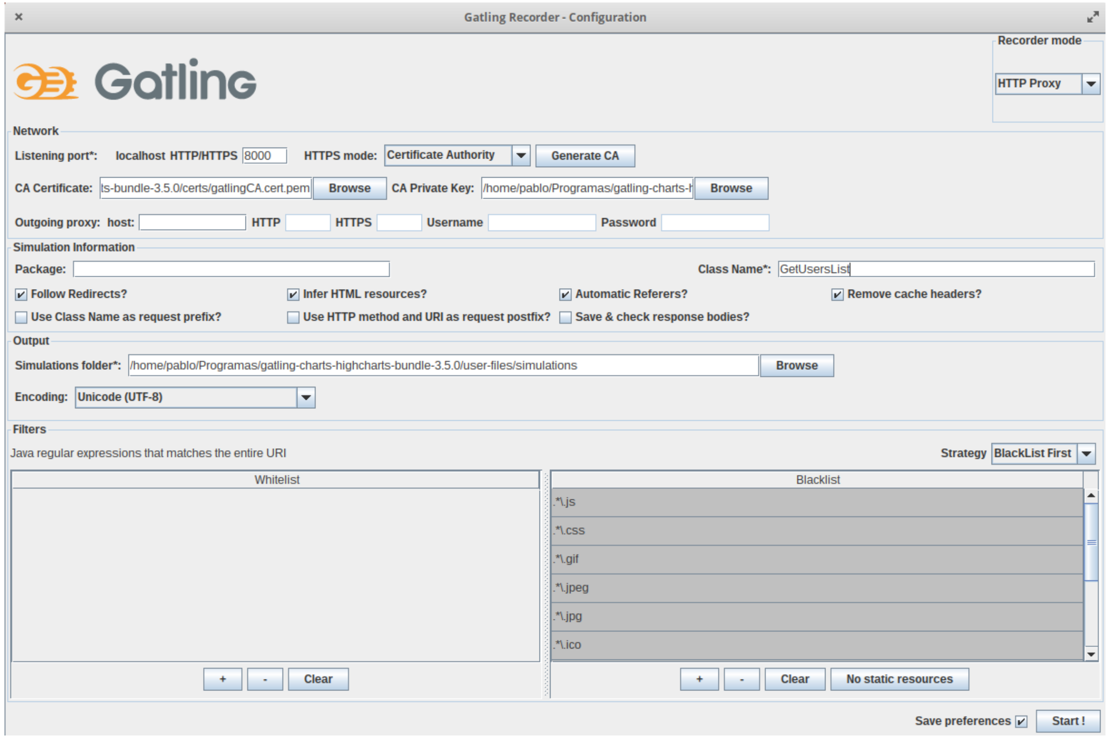
Test case: radarin_0

Launch 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
```

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



Recorder mode: HTTP Proxy

Network

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

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

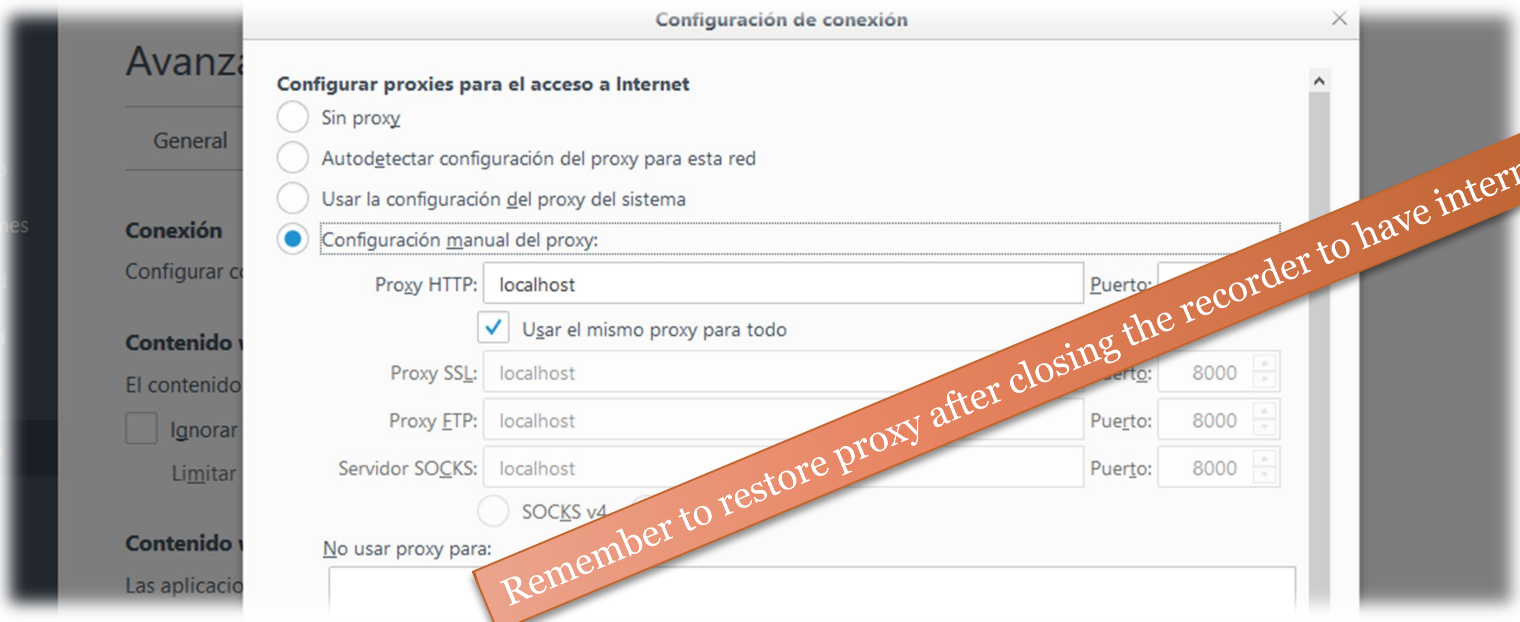
Save preferences [✓] | Start !

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

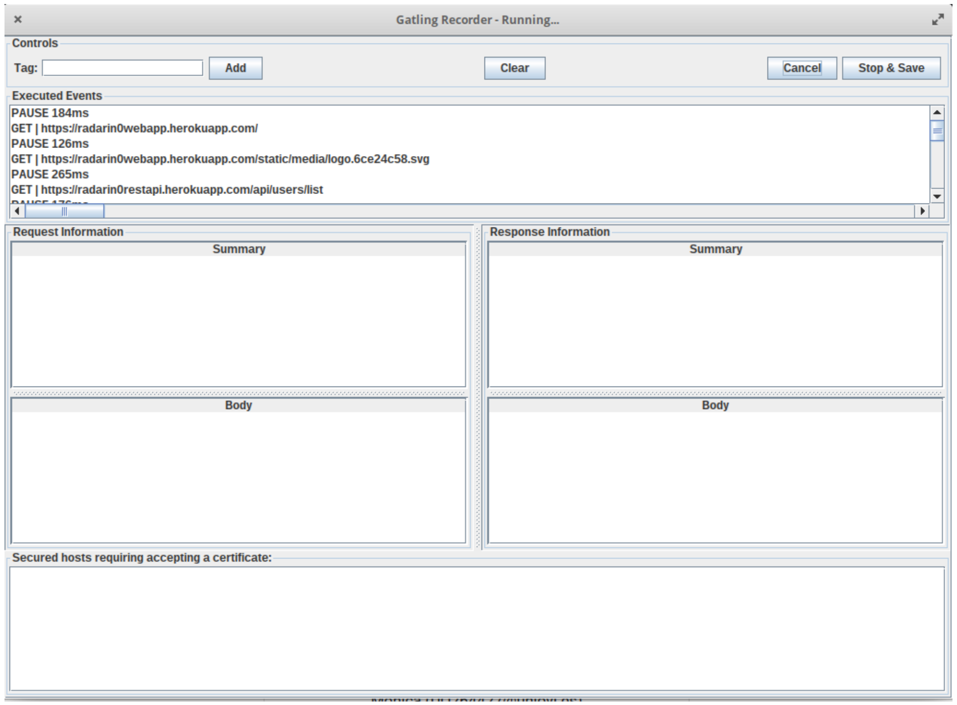
For localhost in firefox, set:
`network.proxy.allow_hijacking_localhost` to true in `about:config`

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*



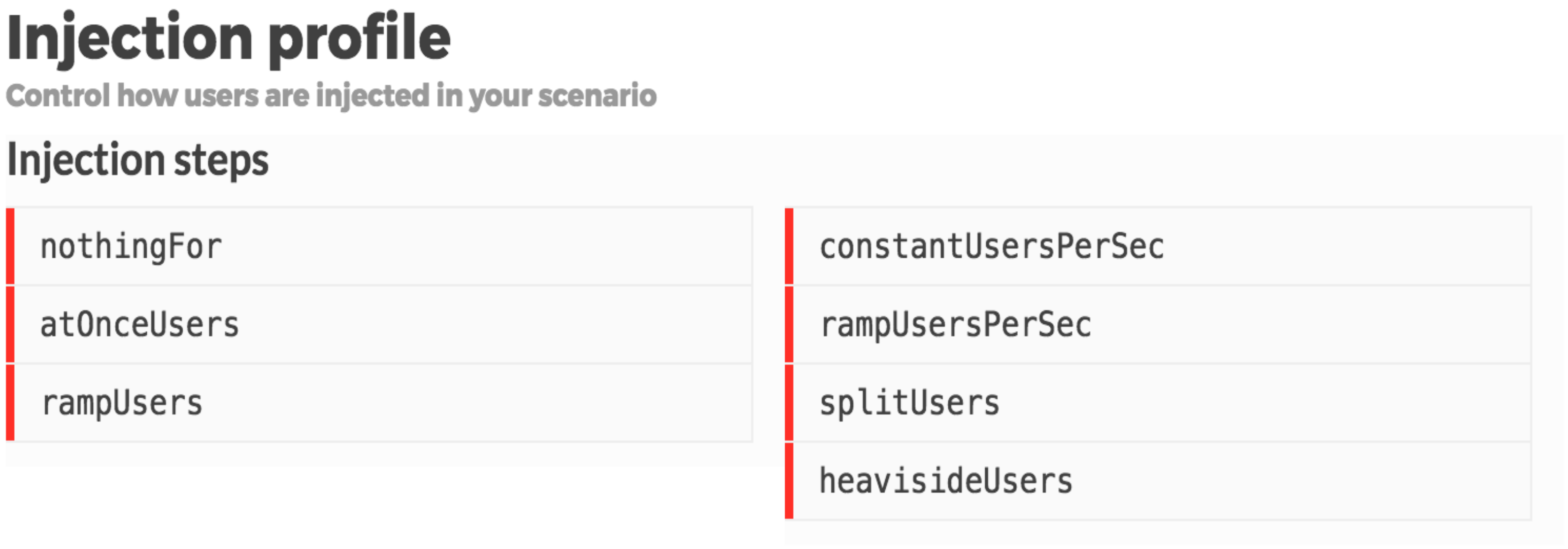
```
PAUSE 184ms
GET | https://radarin0webapp.herokuapp.com/
PAUSE 126ms
GET | https://radarin0webapp.herokuapp.com/static/media/logo.6ce24c58.svg
PAUSE 265ms
GET | https://radarin0restapi.herokuapp.com/api/users/list
```

Simulation Example

- In this case we only have loaded the main page of the application
- 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/Arquisoft/radarin_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
- atOnceUsers
- rampUsers
- constantUsersPerSec
- rampUsersPerSec
- splitUsers
- heavisideUsers

2 users per second during 60 seconds

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

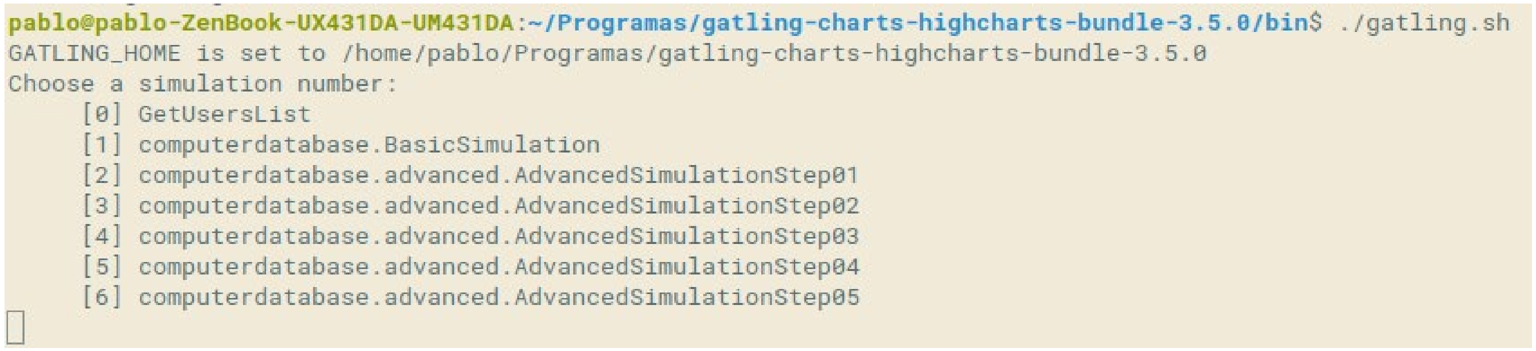
```
. . .
setUp(
   scn.inject(constantUsersPerSec(2) during (60 seconds)  randomized)
).protocols(httpProtocol)
```

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

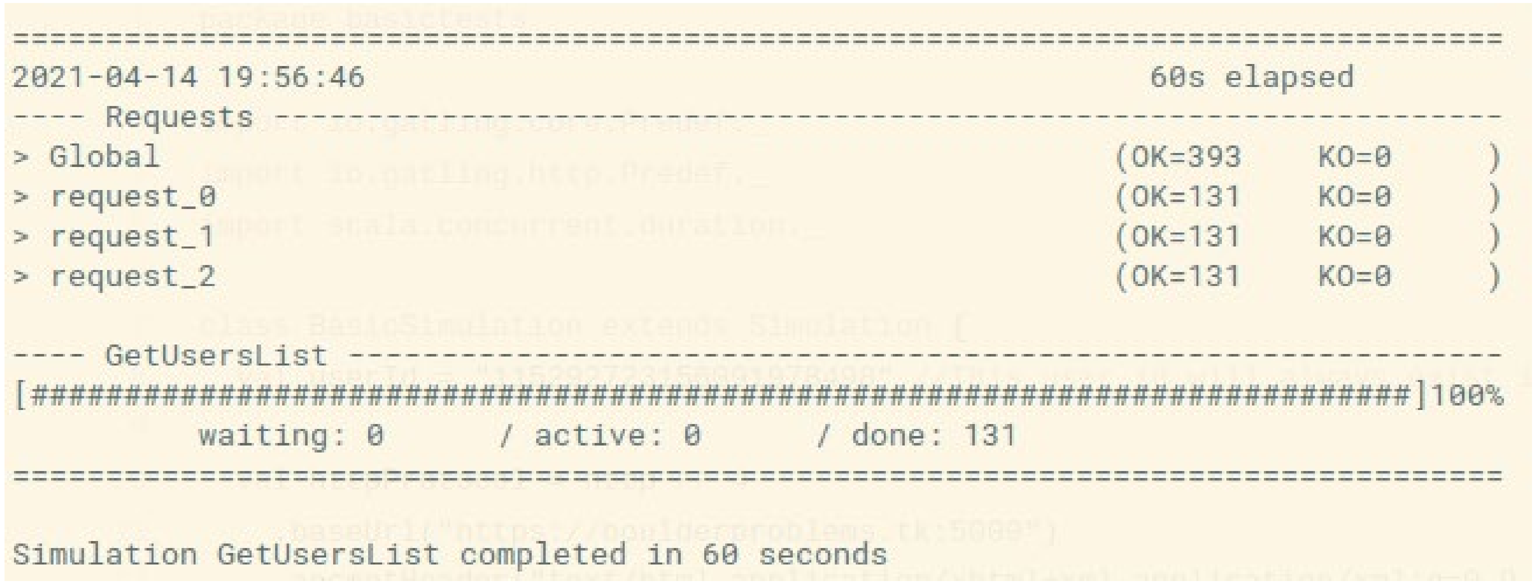
Triggering Gatling

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



```
pablo@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
```

Simulation output

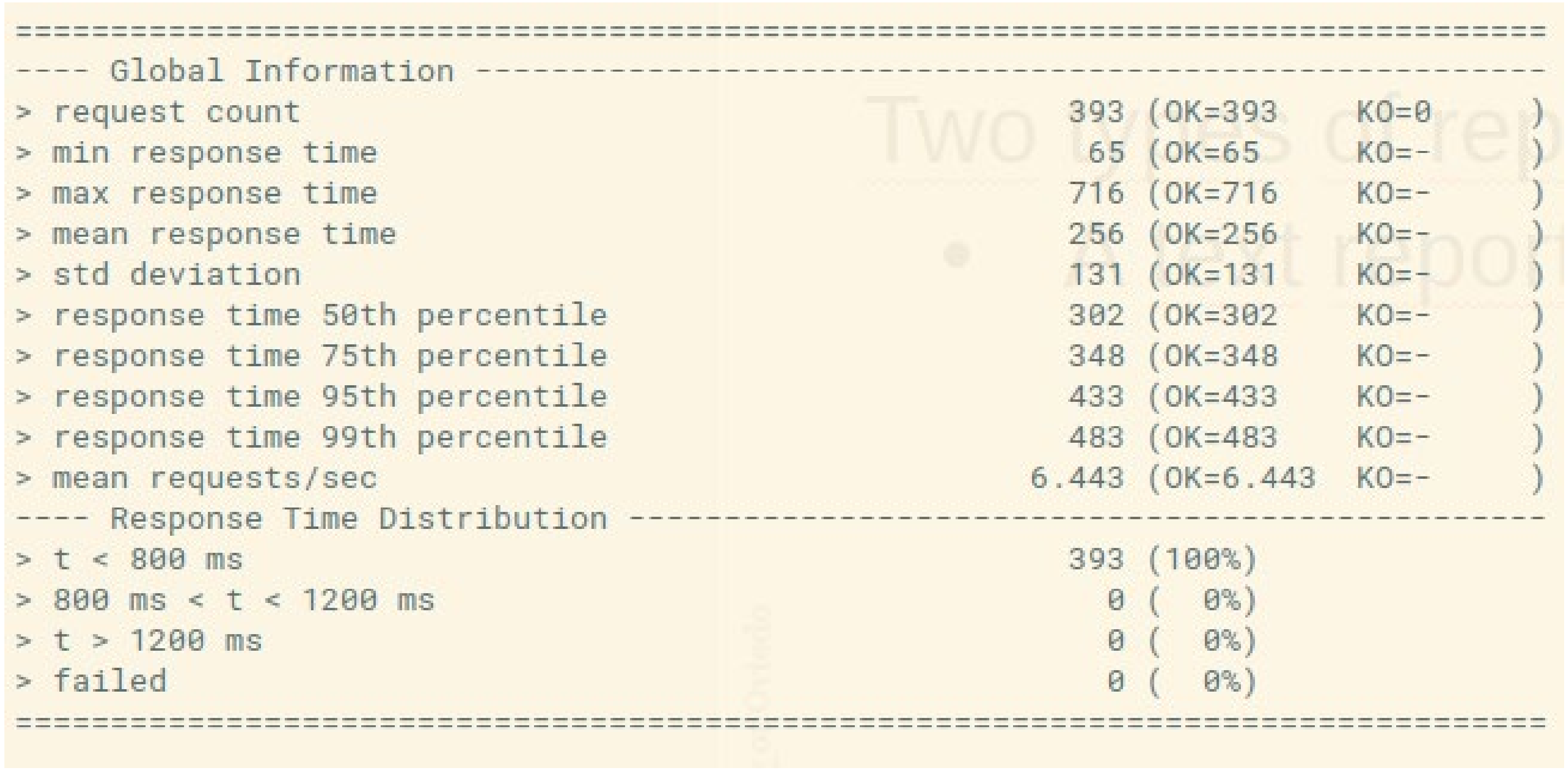


```
================================================================================
2021-04-14 19:56:46                                           60s elapsed
---- Requests ------------------------------------------------------------------
> Global                                                   (OK=393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

---- GetUsersList --------------------------------------------------------------
[##########################################################################]100%
          waiting: 0      / active: 0      / done: 131
================================================================================

Simulation GetUsersList completed in 60 seconds
```

Gatling: Reports

Two types of reports are generated:

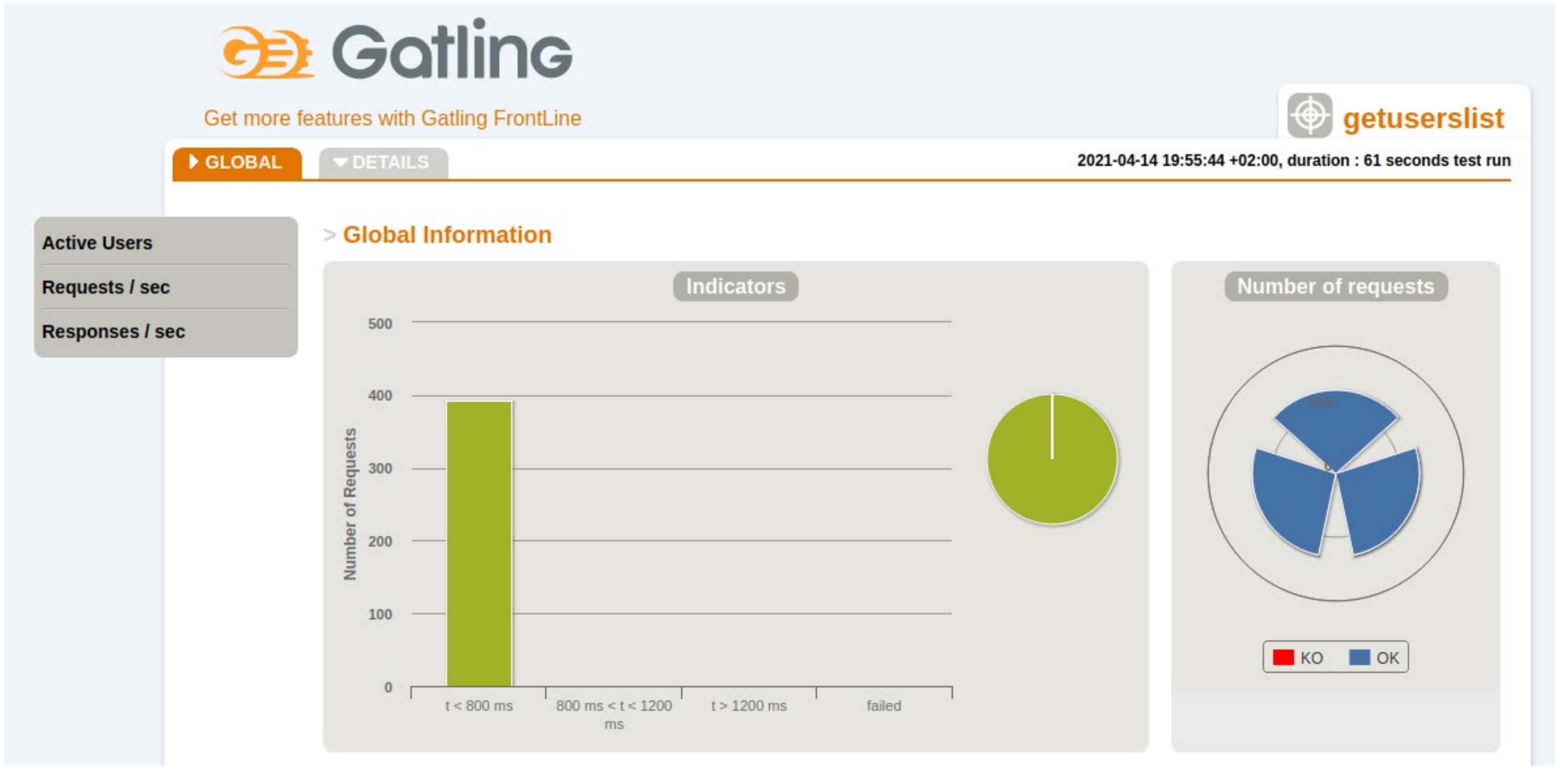
- A text report in the console

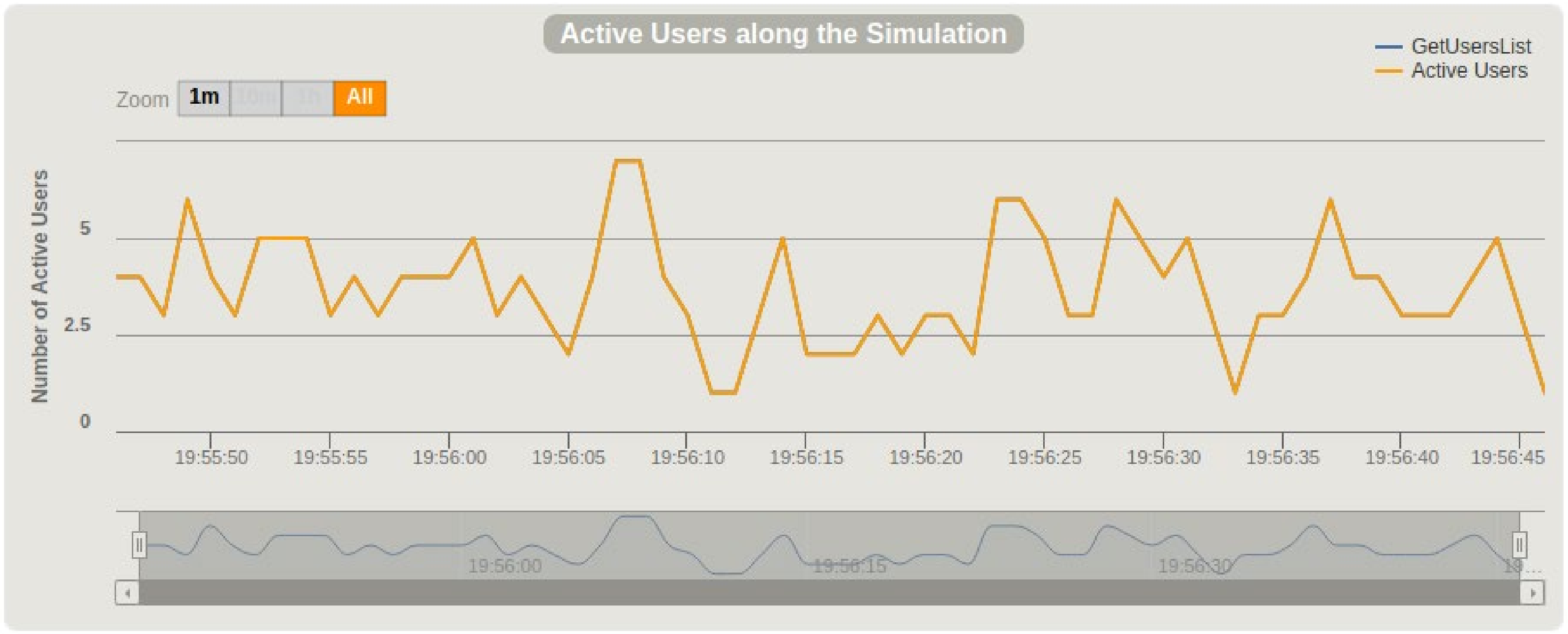


```
================================================================================
---- Global Information --------------------------------------------------------
> request count                                        393 (OK=393    KO=0     )
> min response time                                     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: Reports

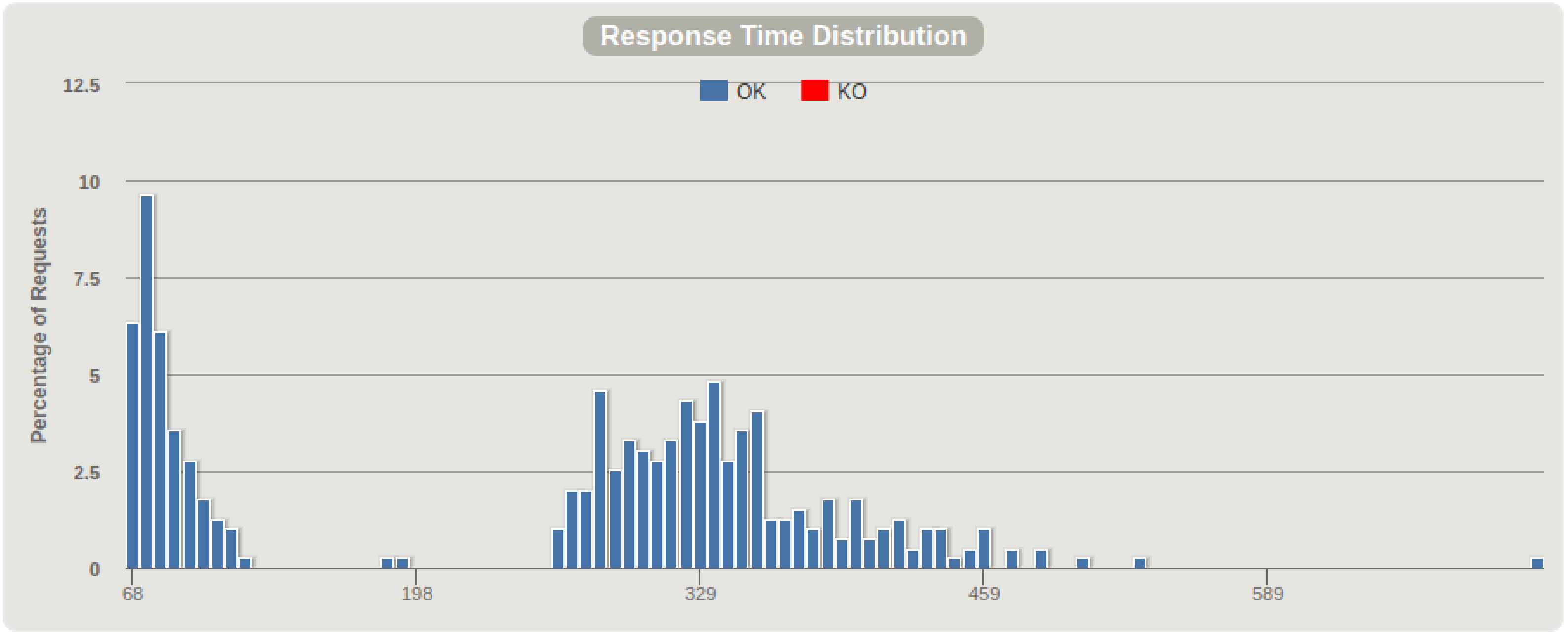
- An html (and more detailed) report:





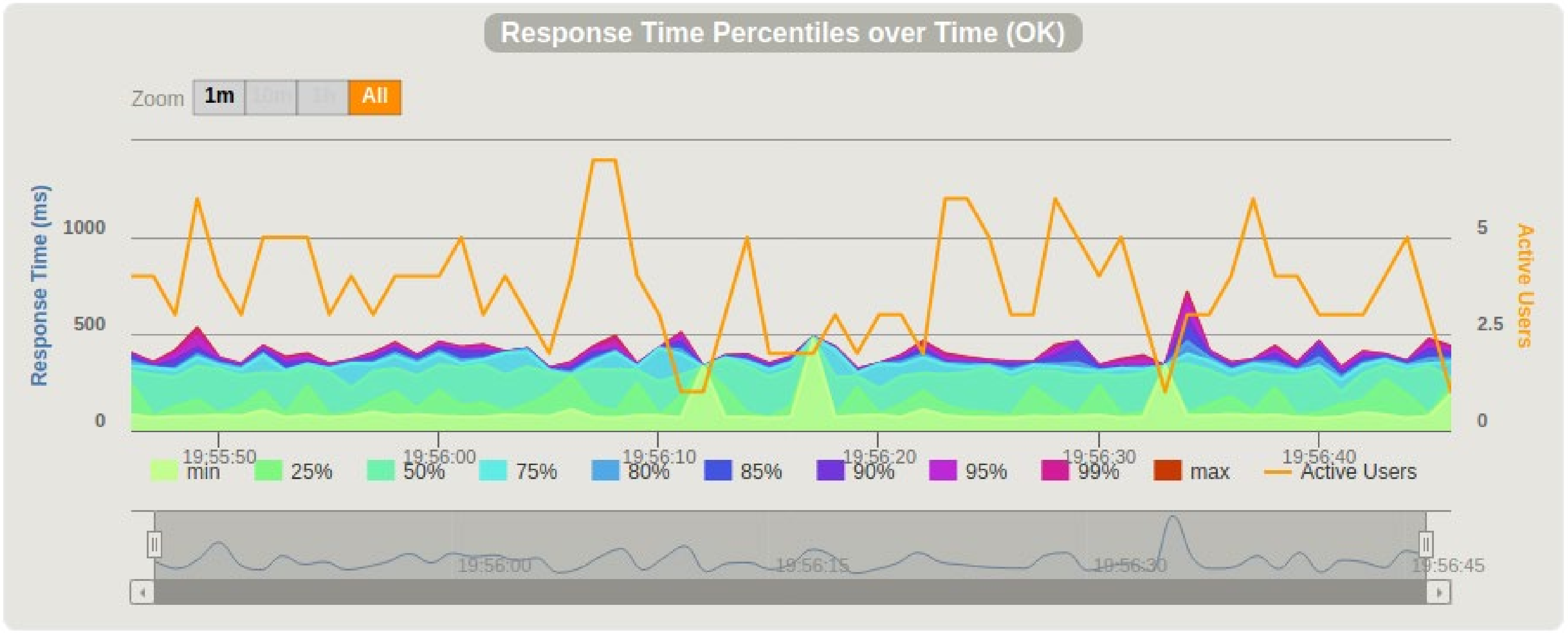
Active Users along the Simulation

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.



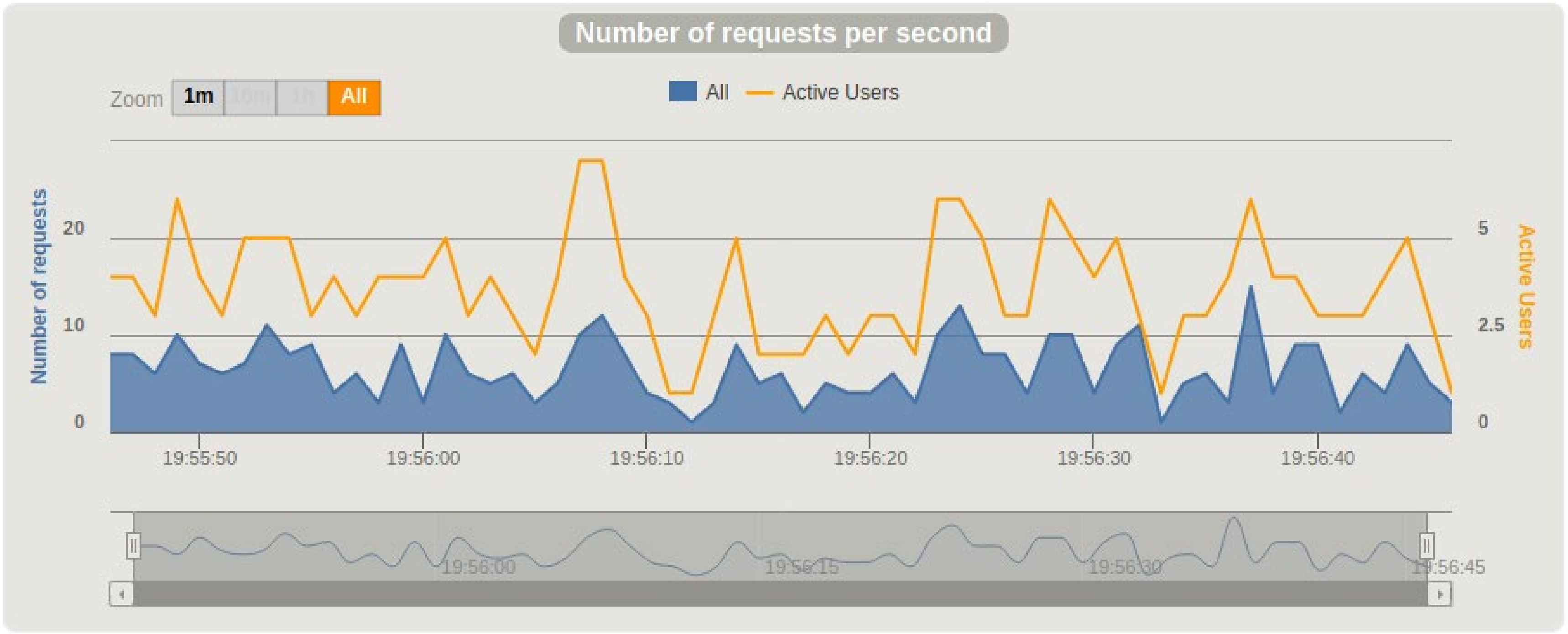
Response Time Distribution

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)