Escuela de Ingeniería Informática

Universidad de Oviedo

Lab 9

Load testing
Other tests

2025-26

Jose Emilio Labra Gayo
Pablo González
Diego Martín
Celia Melendi

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 ()

ab - Apache Server Benchmarking tool (https://httpd.apache.org/docs/2.4/programs/ab.html)

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

Artillery.io ()

goReplay

Loader.io

BlazeMeter

Blitz …

Step by step guide:

https://github.com/pglez82/asw2324_0/blob/master/webapp/README.md#load-testing-gatling

Gatling

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

Download & installation

Download link

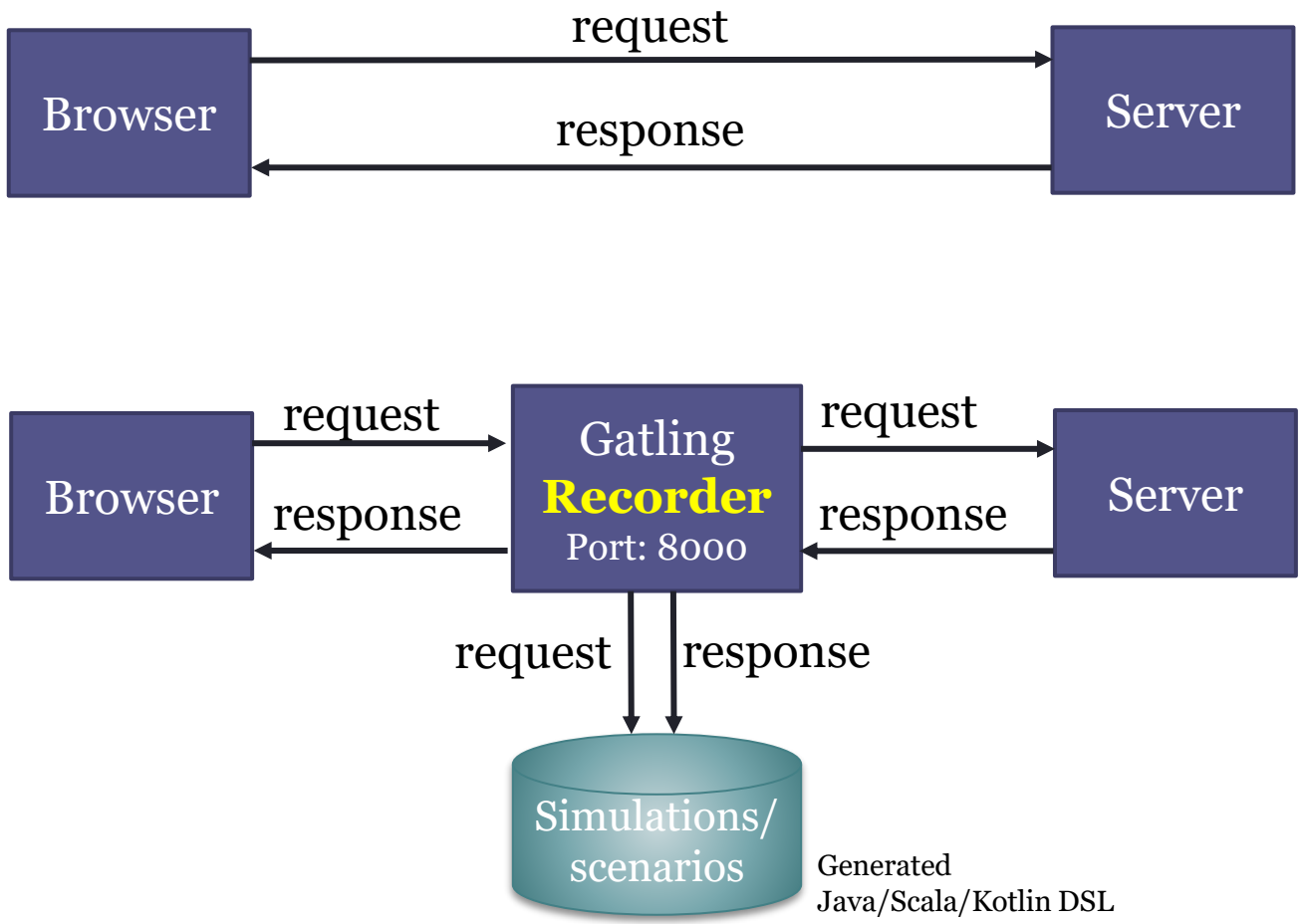
Gatling supports 64bits OpenJDK LTS (Long Term Support) versions: 11, 17 and 21

2 scripts:

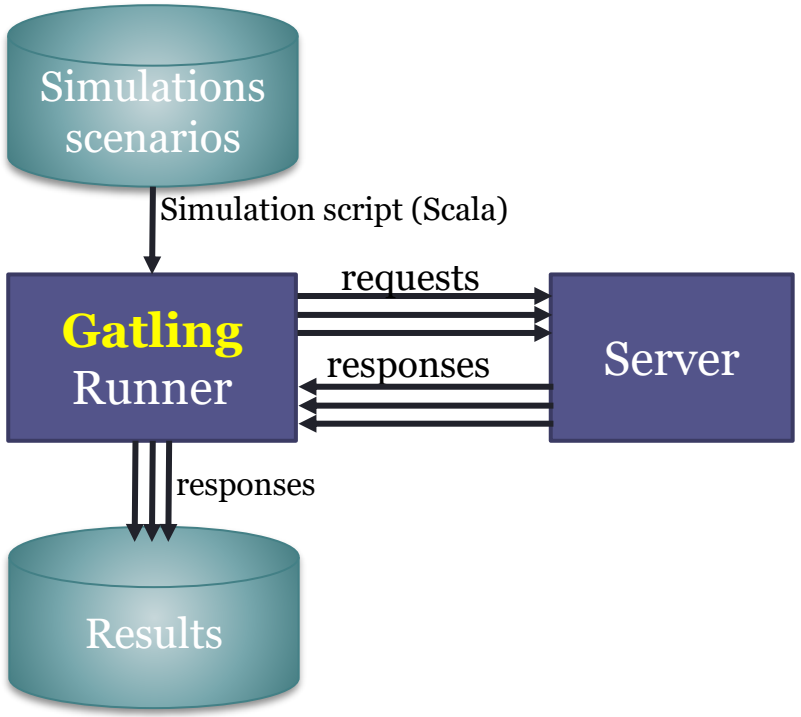
Recorder.sh/Recorder.bat

Gatling.sh/Gatling.bat

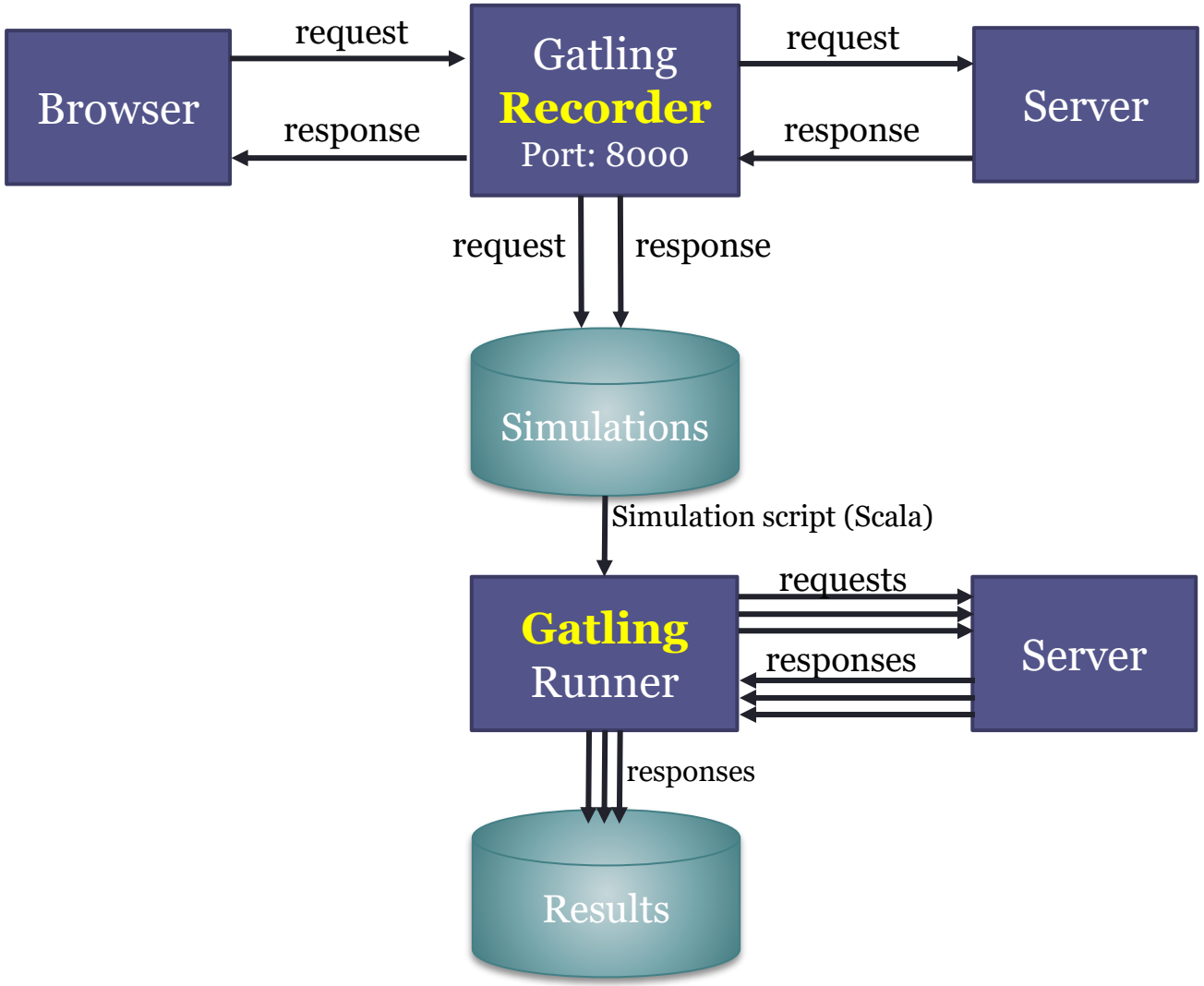
Gatling recorder



Gatling runner



Workflow

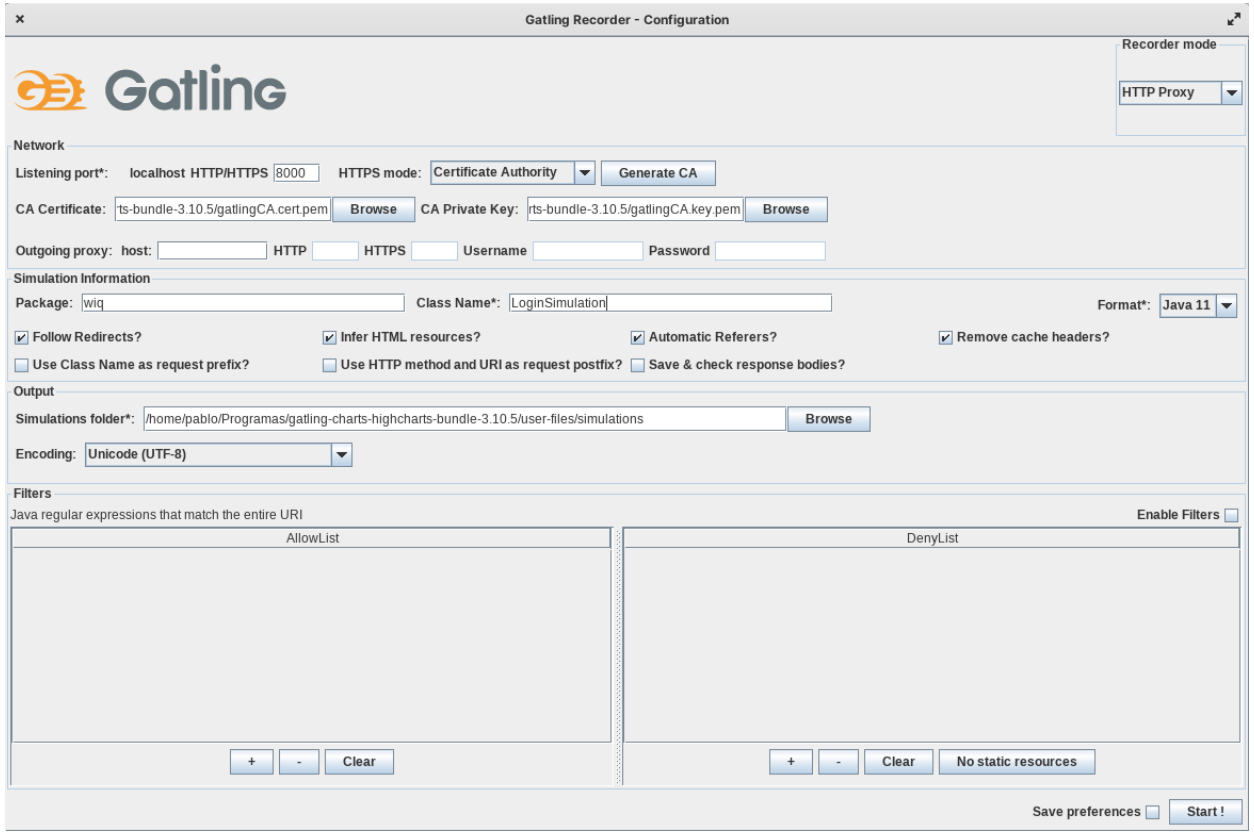

Browser
request
response
Gatling
Recorder
Port: 8000
request
response
Server
request
response
Simulations
Simulation script (Scala)
Gatling
Runner
requests
responses
Server
responses
Results

Gatling: Recorder

Test case: Wiq

Launch 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
```



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: anything we want to avoid hitting

Configure Proxy

`localhost:8000`

For all addresses, included localhost

In case of HTTPS, the certificate must be configured

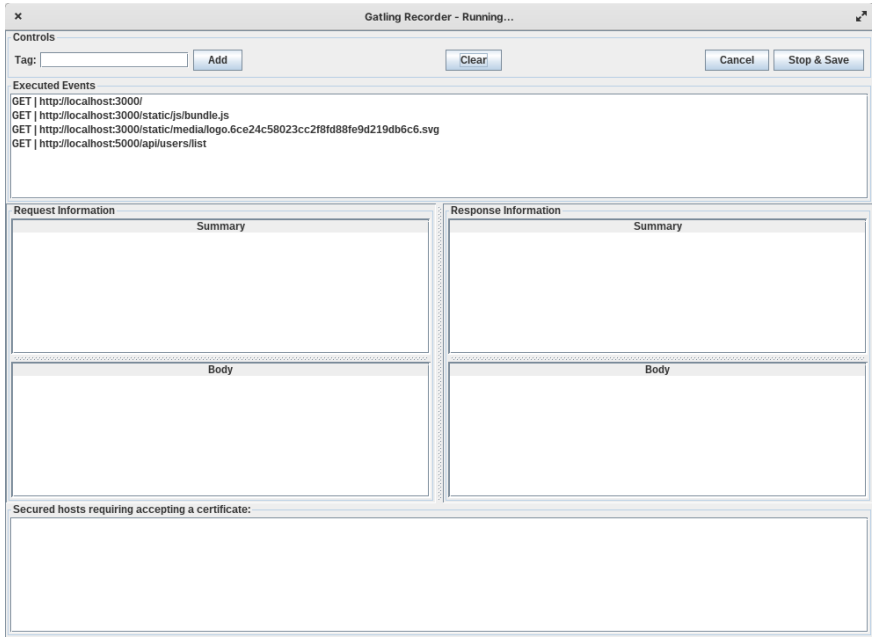
For localhost in Firefox, set:
`network.proxy.allow_hijacking_localhost` to true in `about:config`

Gatling: Recorder

Browser > Web Proxy > localhost:8000

Recorder: Start

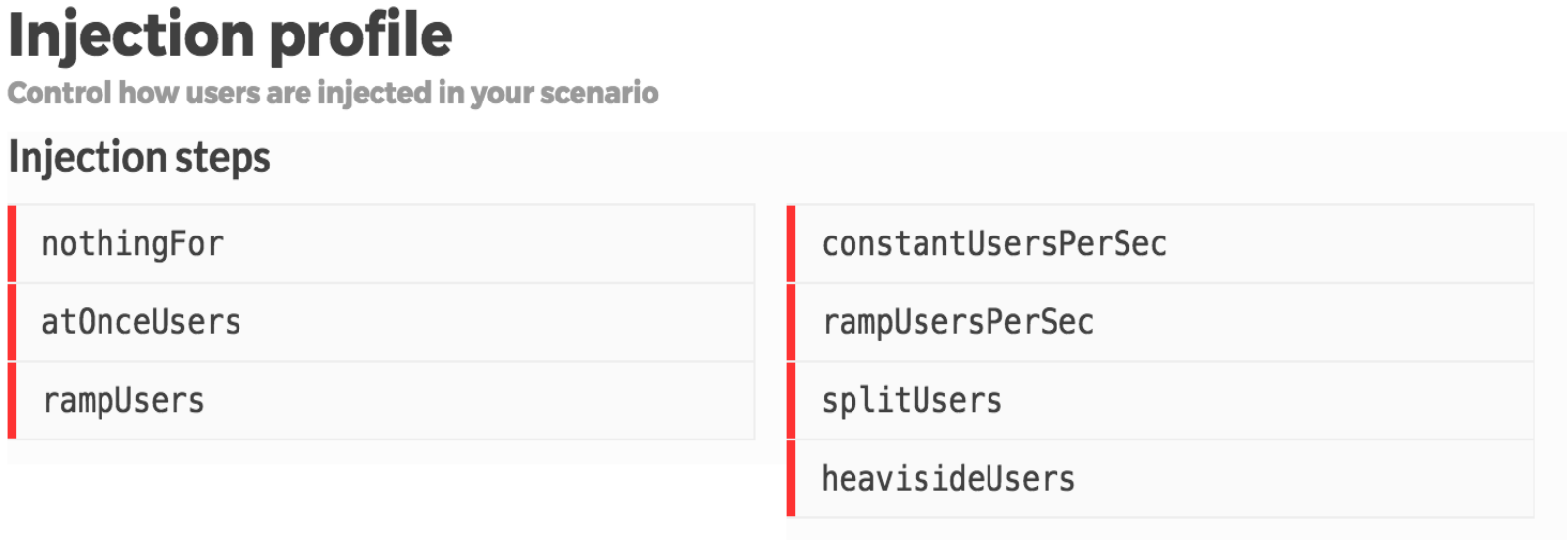
- After starting, open the website and perform the actions that you want to be part of the test
- After finishing press Stop
- Actions will be recorded in **Scala** language
- The simulation will be saved under the directory *user-files/simulations*



Simulation Example

In this case we have loaded the main page of the application and performed a login operation
Note the last line of the test, we can adjust the load here
Tests can be more complicated, performing multiple actions on the system
We can also choose to **write our own code** without the need of the recorder

How-to configure the number of users…



https://docs.gatling.io/reference/script/core/injection/

Feeders – Configurando los datos de test

- CSV

```
username,password
user1,pass1
user2,pass2 ...
```

- feed createion

```
val csvFeeder = csv("users.csv").random()
```

- Using the feed

```
val scn = scenario("Login Scenario") .feed(csvFeeder) // Inyecta los datos
.exec(http("Login Request") .post("/login") // Reemplaza con la URL de login
.formParam("username", "${username}") // Usa el valor del username del feed
.formParam("password", "${password}") //Usa el valor del password del feed
```

- Execute

```
setUp(scn.inject(atOnceUsers(10)).protocols(httpProtocol))
```

https://docs.gatling.io/reference/script/core/session/feeders/

2 users per second during 60 seconds

120 users arriving at the rate of 2 users/second

They execute a given script

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

```
(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
```

Simulation output

```
================================================================================
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: Reports

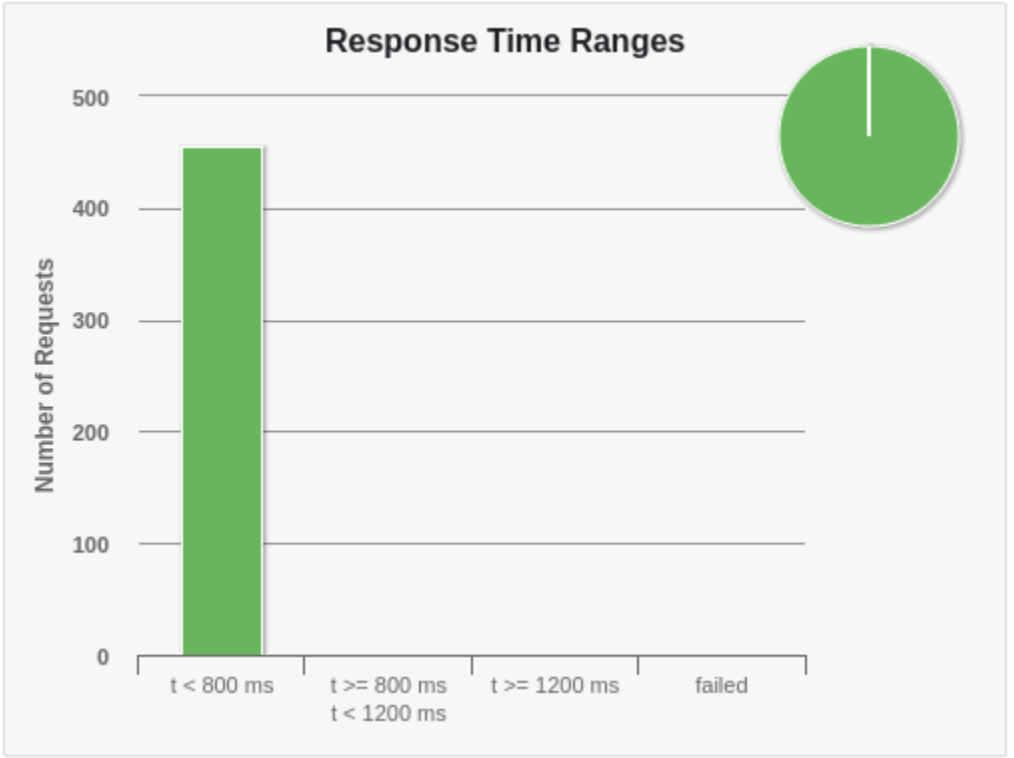
Two types of reports are generated:

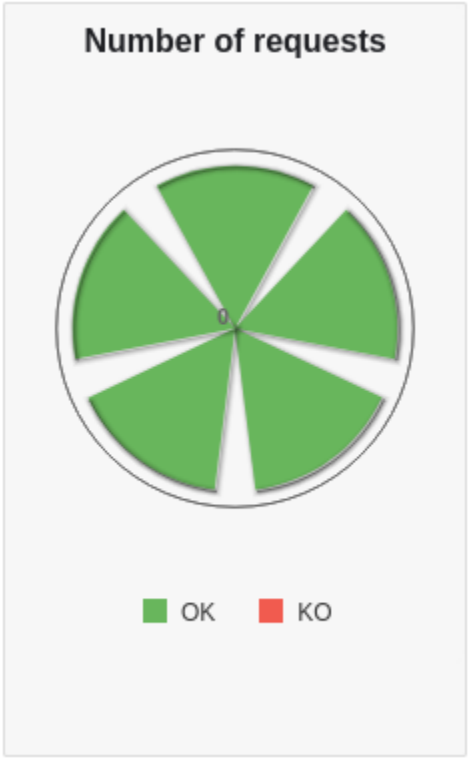
- A text report in the console

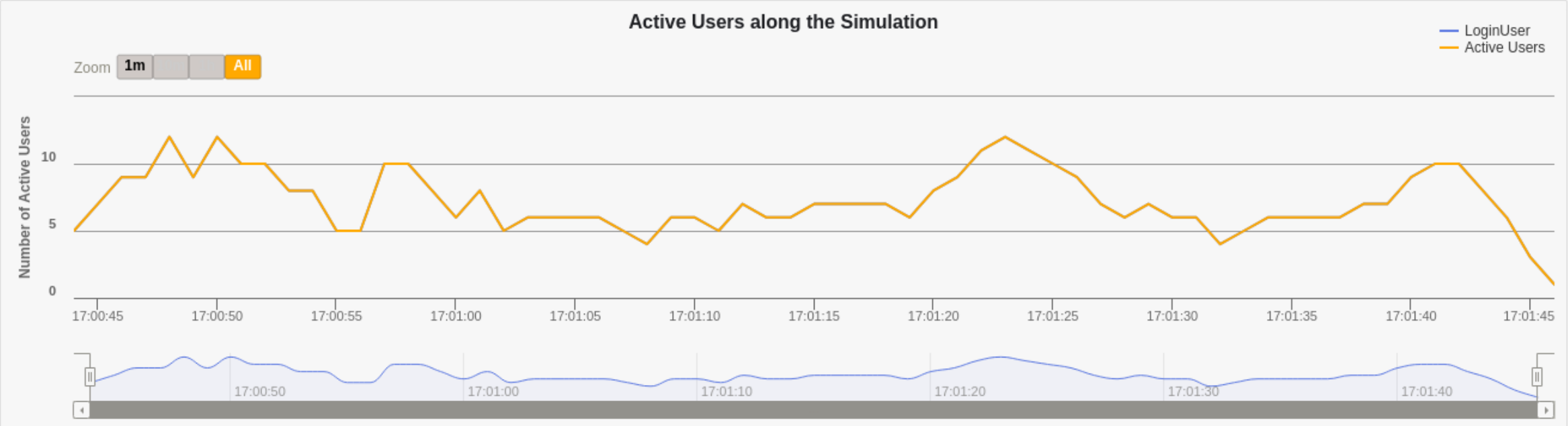
```
================================================================================
---- Global Information --------------------------------------------------------
> request count                                        455 (OK=455    KO=0     )
> min response time                                    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: Reports

- HTML (and more detailed) report:



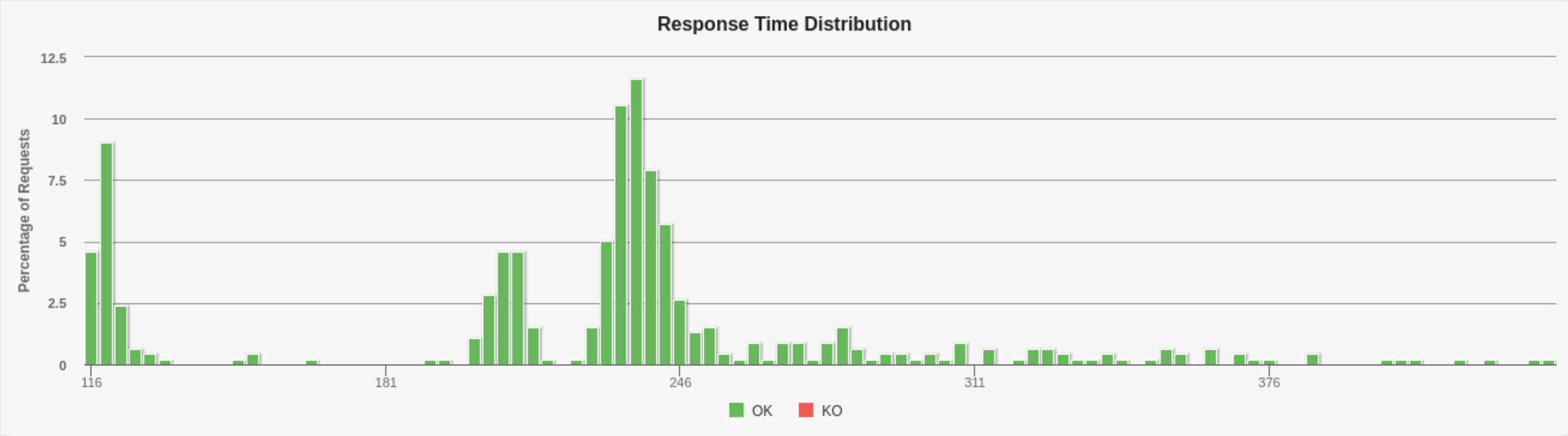




Active Users along the Simulation

Displays number of active users (sending requests and receiving responses) along simulation time.

This measure can be related to others such as response times and number of requests.



Response Time Distribution

Shows the percentage of all requests made during the test run on the Y axis

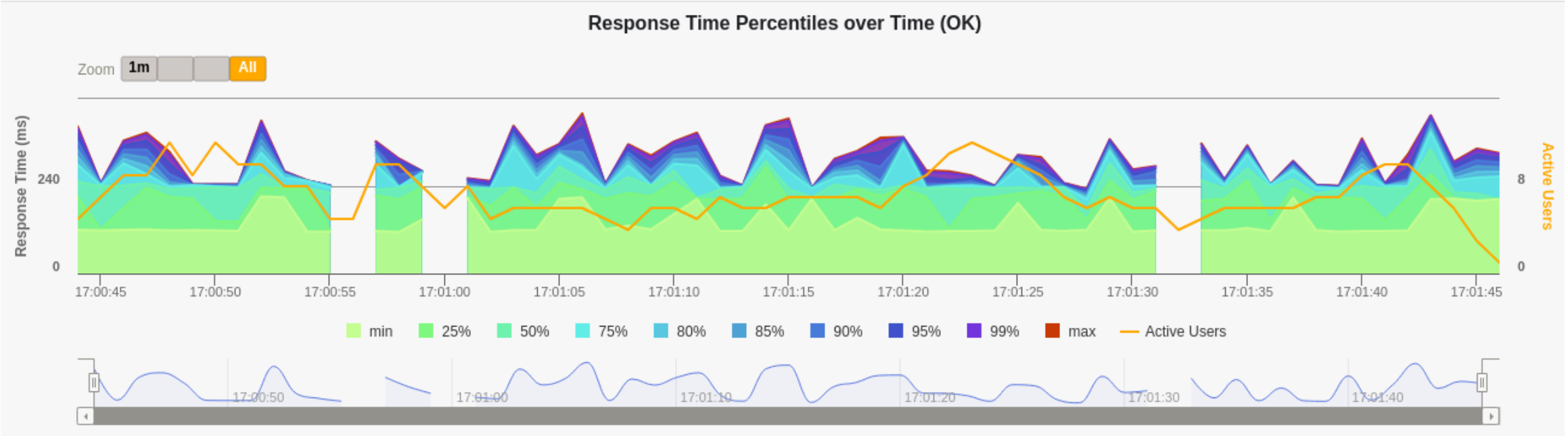
Includes both successes and failures.

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.

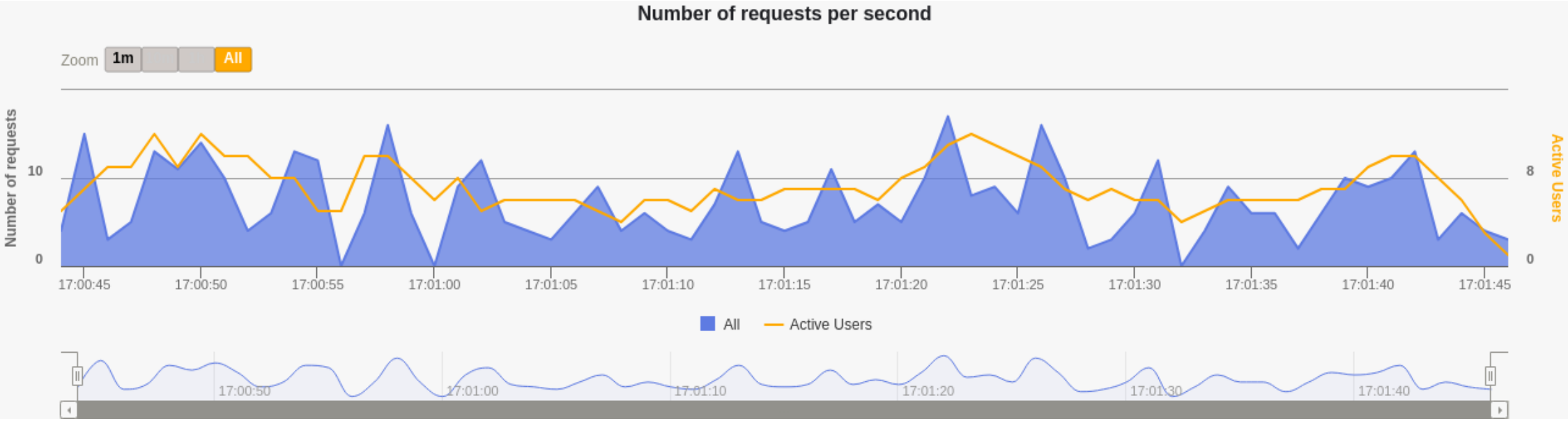
As you increase load on the server, you should see the data on this chart move farther to the right

Response times will get slower



Response Time Percentiles over Time

This is similar to Response Time Distribution, but it shows the data over a longer period of time to assess how your system behaves when under a sustained load.



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)