

# REPORTE DE PRUEBAS

Los NullPointers

*Guatemala 22 de junio 2021  
Universidad San Carlos de Guatemala*

## Pruebas Unitarias (Jest & SuperTest)

Para realizar las pruebas unitarias en el servidor se utilizó la herramienta Jest la cual permite realizar pruebas en el lenguaje JavaScript y SuperTest el cual permite realizar las pruebas a endpoints.

Lo primero que debemos realizar es instalar cada una de las herramientas como dependencias de desarrollo a nuestro proyecto. Para esto ejecutamos los siguientes comandos.



```
npm install jest --save-dev
npm install supertest --save-dev
```

Luego de instalar las herramientas de testing es necesario crear nuestros scripts para ejecutarlos. Se crearán dos scripts uno para dejar a la escucha de cambios y el otro para solo realizar una ejecución del mismo. Esto se debe realizar en el archivo package.json



```
{
  ...,
  "scripts": {
    ...,
    "test": "jest --verbose --silent",
    "test:watch": "npm run test -- --watch"
  },
  ...,
  "jest": {
    "testEnvironment": "node"
  }
}
```

Luego de configurar nuestros scripts podemos comenzar a crear nuestros archivos de prueba para esto es necesario que nuestros archivos cumplan alguna de las siguientes condiciones

- ✎ Estar dentro de un directorio llamado **test** o **\_\_test\_\_**
- ✎ Tener una extensión **\*.spect.js** o **\*.test.js**

Al cumplir con los requisitos anteriores podemos comenzar a escribir nuestros casos de prueba. Para esto debemos importar el servidor y las conexiones que puedan quedar

abiertas. Luego de esto podemos directamente escribir nuestros casos o describirlos encerrándolos en las funciones **describe** propias de jest

```
const supertest = require('supertest');

const {
  app,
  server
} = require('../app');
const api = supertest(app);
const mysql = require('../config/database');

describe('GET /report', () => {
  describe('Get all reports', () => {
    test('Should respond with a 200 status code', async () => {
      await api
        .get('/report')
        .expect(200)
        .expect('Content-Type', /application\/json/);
    });
  });
});

afterAll(() => {
  mysql.end();
  server.close();
});
```

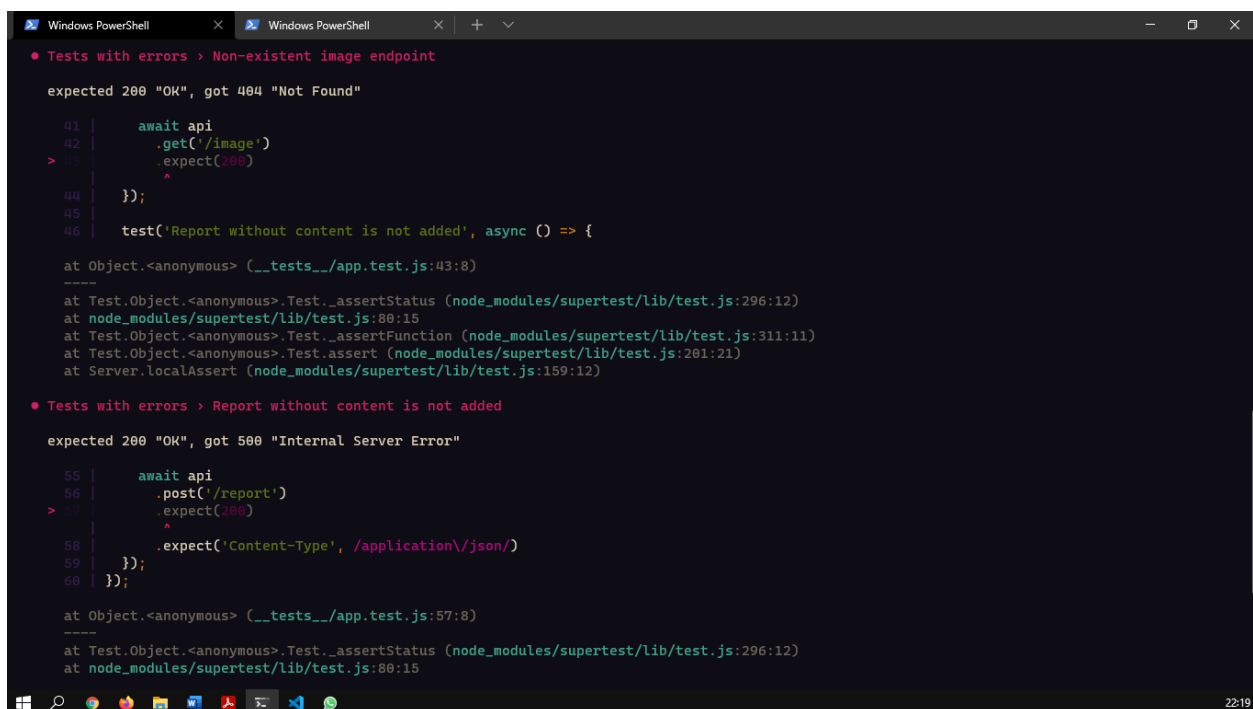
Para este proyecto se realizaron 4 las cuales se requirió que 2 tuvieran estado correcto y 2 estado fallido.

Al correr nuestro script de ejecución de test nos devuelve el estado de cada prueba. Un resumen de nuestros tests, donde podemos ver cuántos fueron correctos y cuantos fallaron mostrándonos la descripción y nombre que les colocamos

```
FAIL __tests__/app.test.js
Post /login
  ✓ Given a valid username and password (481 ms)
GET /report
  Get all reports
    ✓ Should respond with a 200 status code (207 ms)
Tests with errors
  ✗ Non-existent image endpoint (25 ms)
  ✗ Report without content is not added (208 ms)
```

```
Test Suites: 1 failed, 1 total
Tests:       2 failed, 2 passed, 4 total
Snapshots:   0 total
Time:        4.352 s
```

✎ Estado Fallido: Cuando un test falla nos muestra cual fue y donde fallo



```
Windows PowerShell
• Tests with errors > Non-existent image endpoint

expected 200 "OK", got 404 "Not Found"

41 |     await api
42 |       .get('/image')
> 43 |       .expect(200)
   |       ^
44 |   });
45 |
46 |   test('Report without content is not added', async () => {

at Object.<anonymous> (__tests__/app.test.js:43:8)
-----
at Test.Object.<anonymous>.Test._assertStatus (node_modules/supertest/lib/test.js:296:12)
at node_modules/supertest/lib/test.js:80:15
at Test.Object.<anonymous>.Test._assertFunction (node_modules/supertest/lib/test.js:311:11)
at Test.Object.<anonymous>.Test.assert (node_modules/supertest/lib/test.js:201:21)
at Server.localAssert (node_modules/supertest/lib/test.js:159:12)

• Tests with errors > Report without content is not added

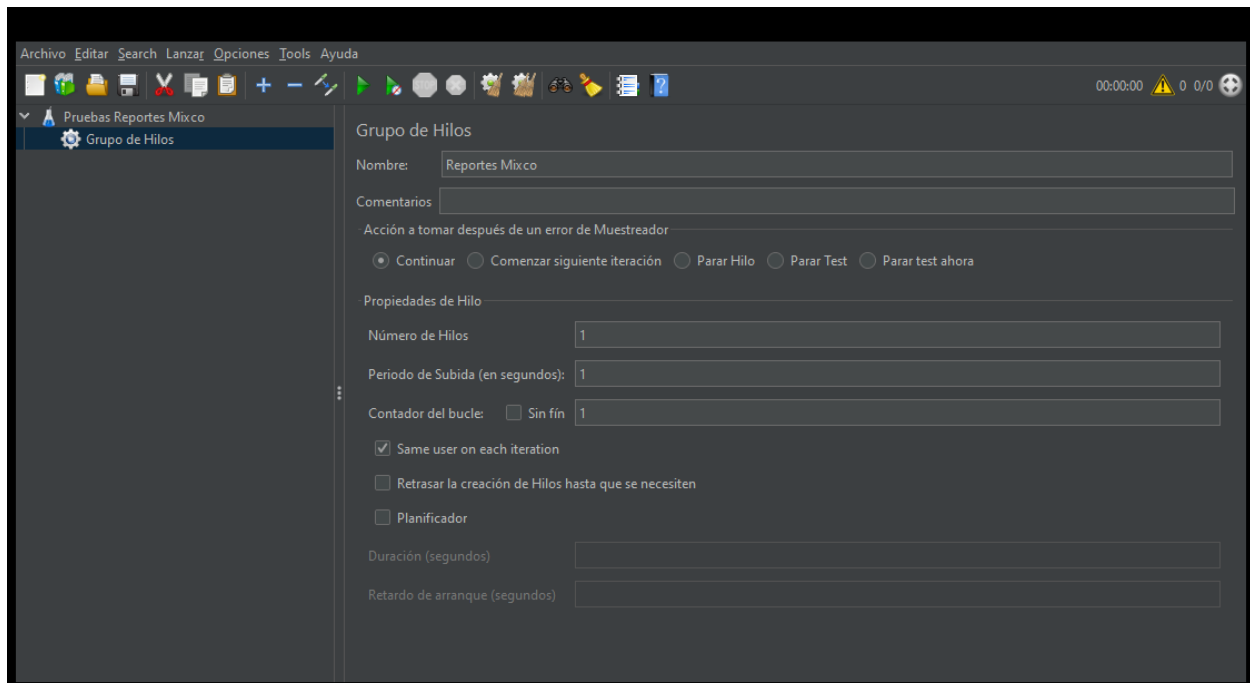
expected 200 "OK", got 500 "Internal Server Error"

55 |     await api
56 |       .post('/report')
> 57 |       .expect(200)
   |       ^
58 |       .expect('Content-Type', /application\/json/)
59 |   });
60 | });

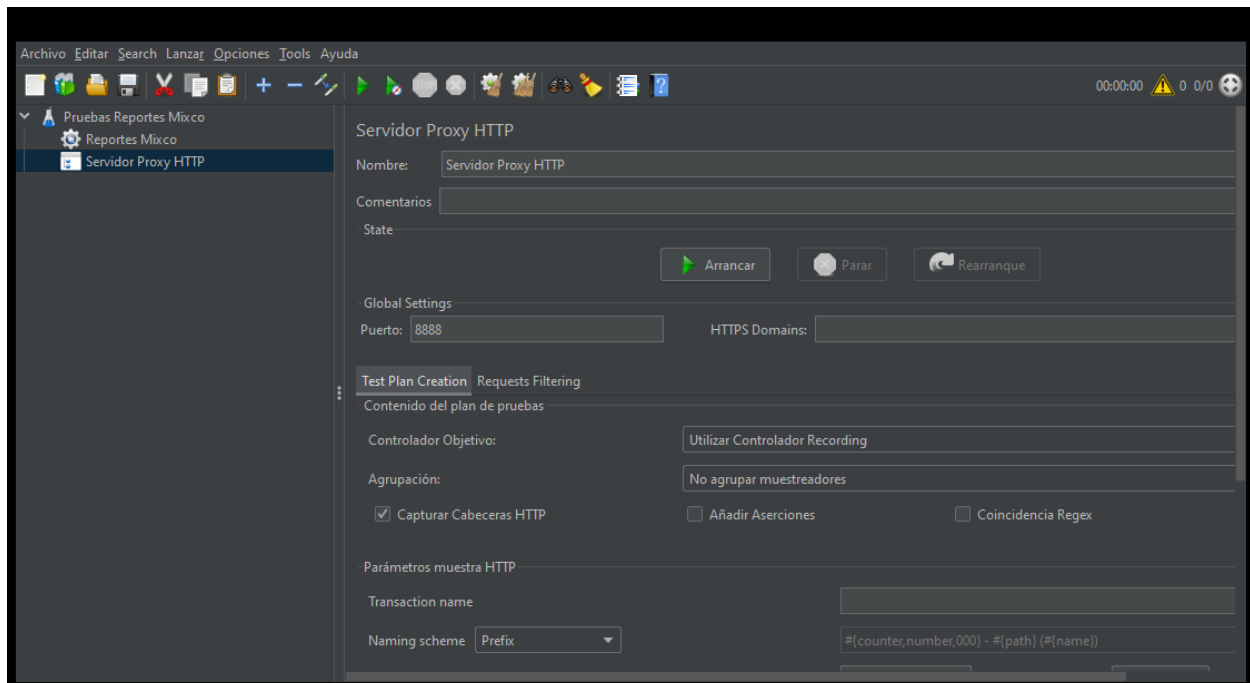
at Object.<anonymous> (__tests__/app.test.js:57:8)
-----
at Test.Object.<anonymous>.Test._assertStatus (node_modules/supertest/lib/test.js:296:12)
at node_modules/supertest/lib/test.js:80:15
```

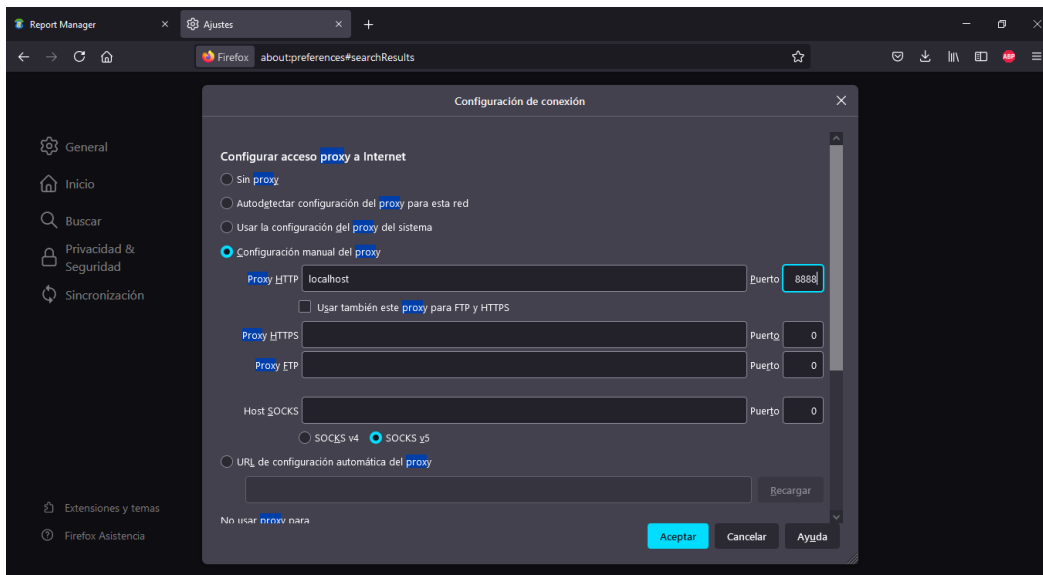
# Pruebas de Carga (JMeter)

Lo primero que se necesita es crear un grupo de hilos para la prueba



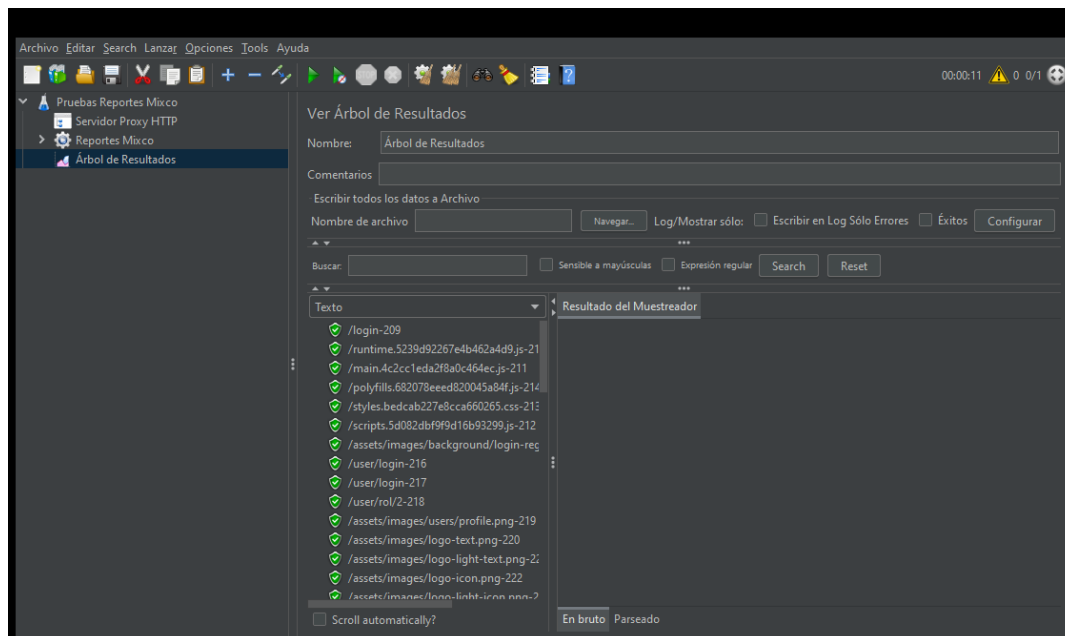
Luego se necesita añadir elementos de prueba servidor proxy HTTP y a la vez configurar nuestro navegador con el puerto en este caso es el 8888





Al terminar de realizar las configuraciones podemos arrancar la grabación de la prueba especificando a que grupo de hilos donde se almacenaran las peticiones.

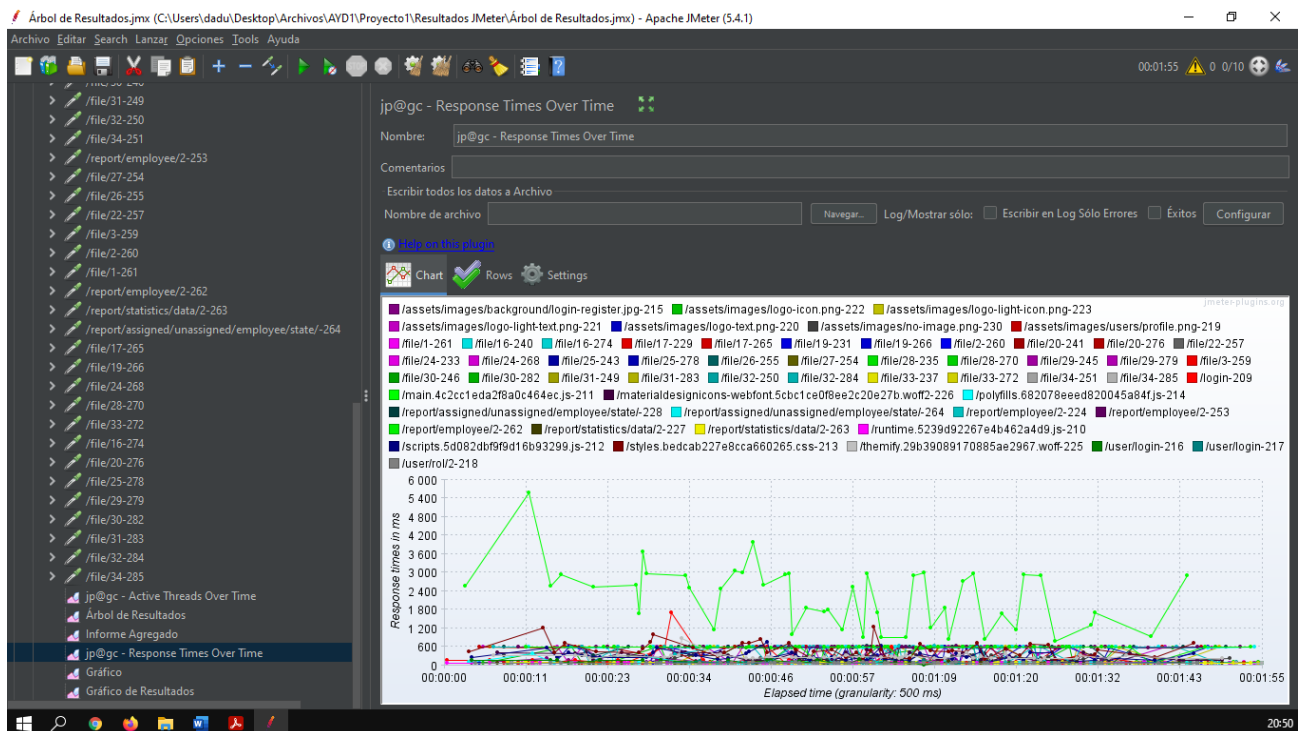
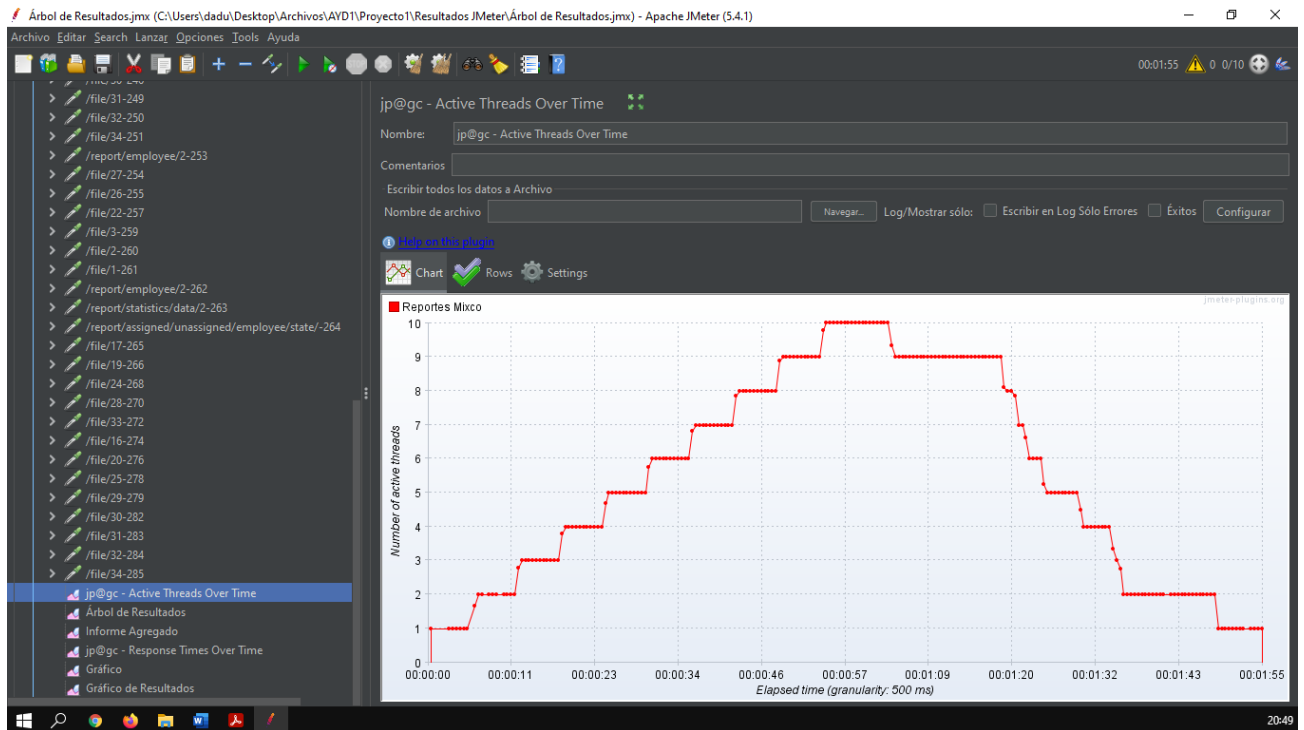
También debemos agregar un receptor de tipo árbol en el cual podremos ver el estado de cada petición realizada por los hilos y los plugins necesarios como lo es el Standard Set



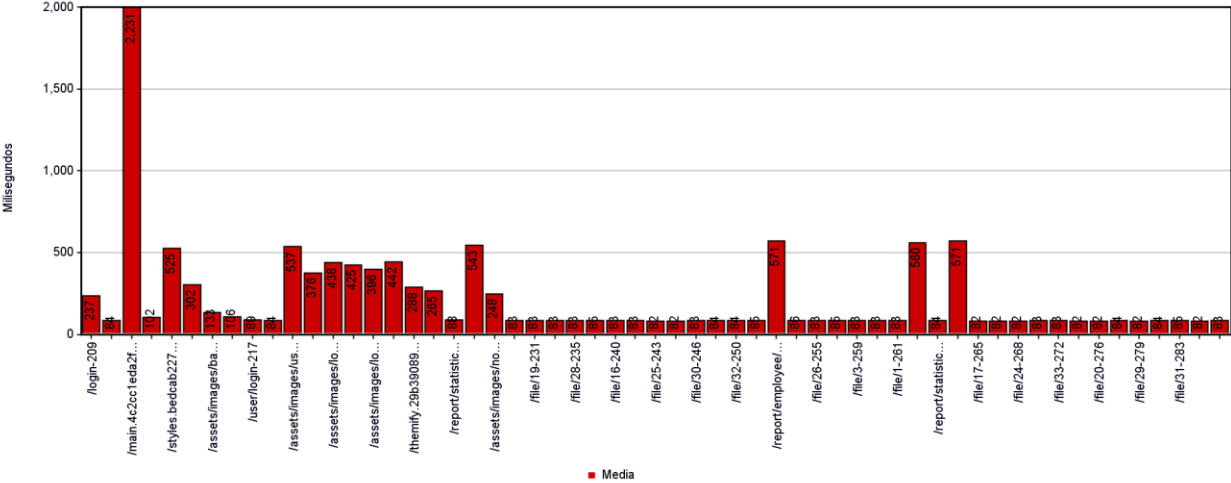
Ya configurados nuestros plugins es necesario que configuremos el periodo de la prueba y los usuarios. Para este proyecto se realizaron 5 pruebas en 60 segundos cada una con 10, 100 y 1000 usuarios. Llegando a la conclusión que el servidor donde se encuentra

alojada la pagina llega a soportar un máximo de 1000 usuarios los cuales no tendrían una navegación optima debido a que tendrían tiempos de respuestas largos.

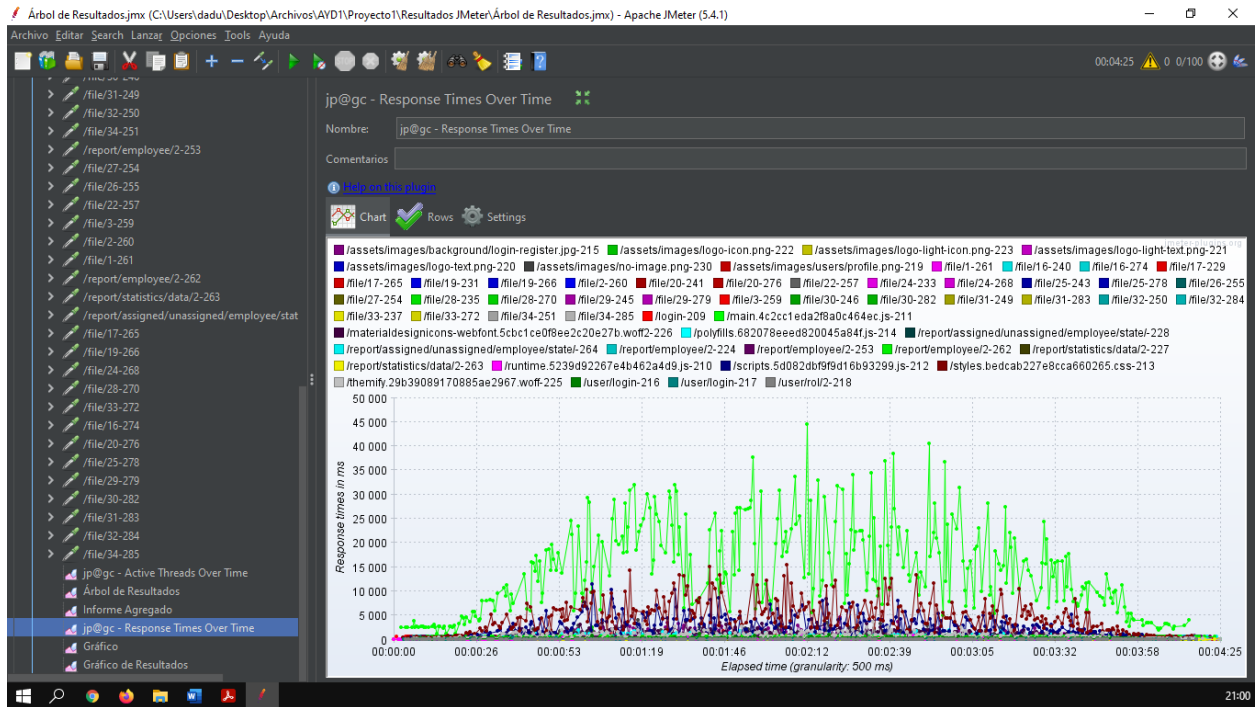
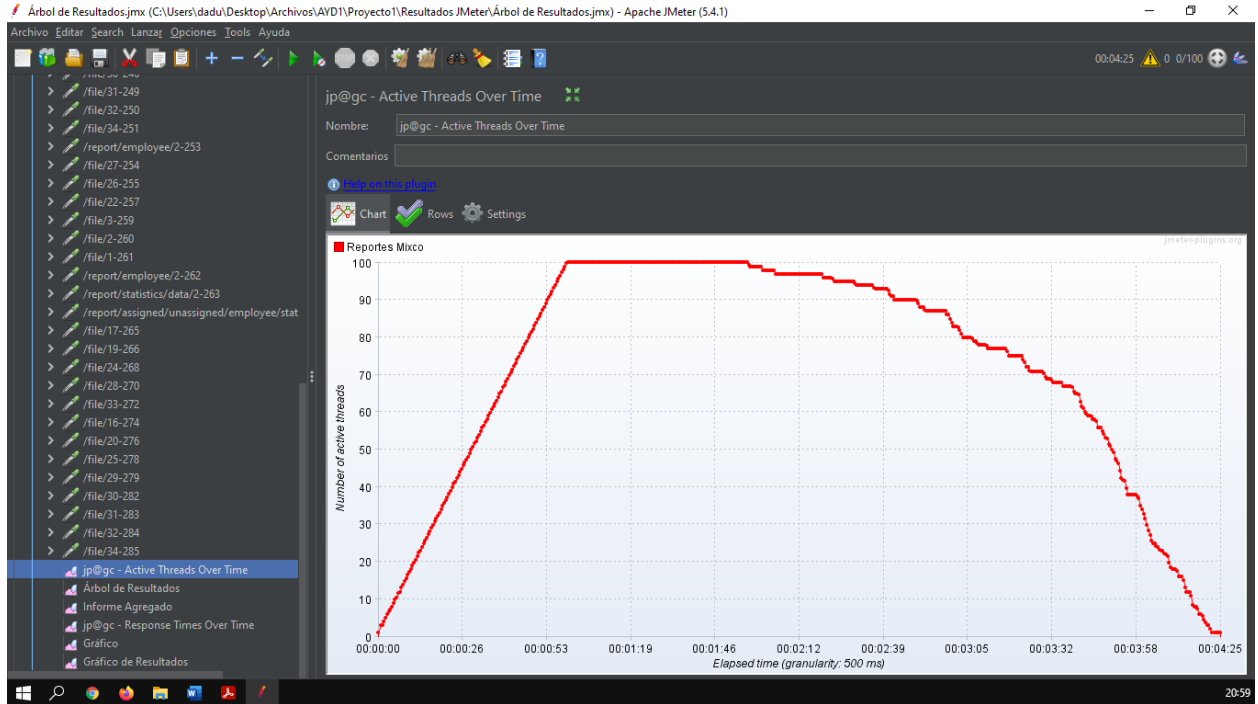
## 10 usuarios



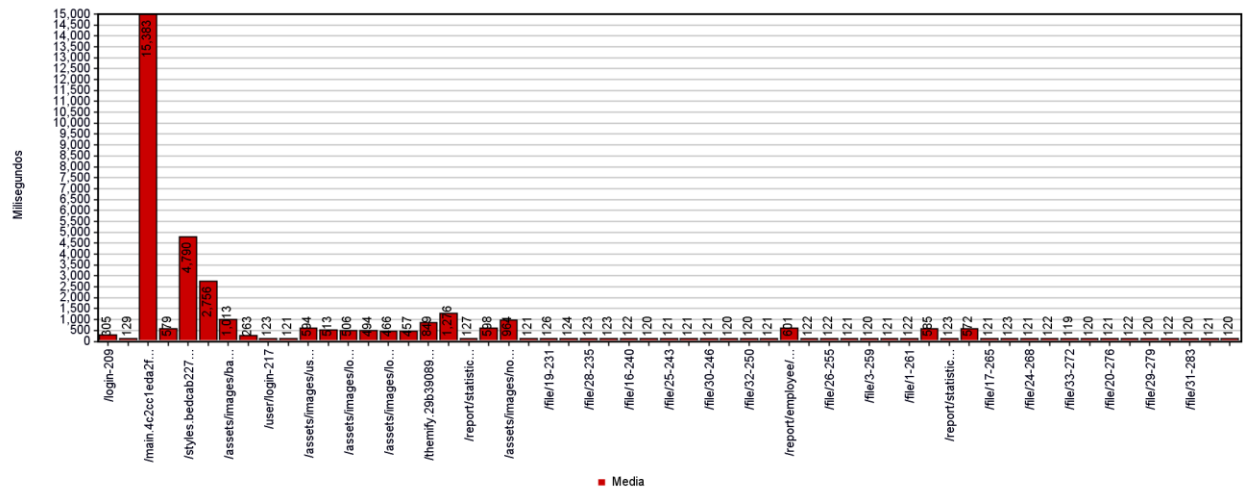
Gráfico



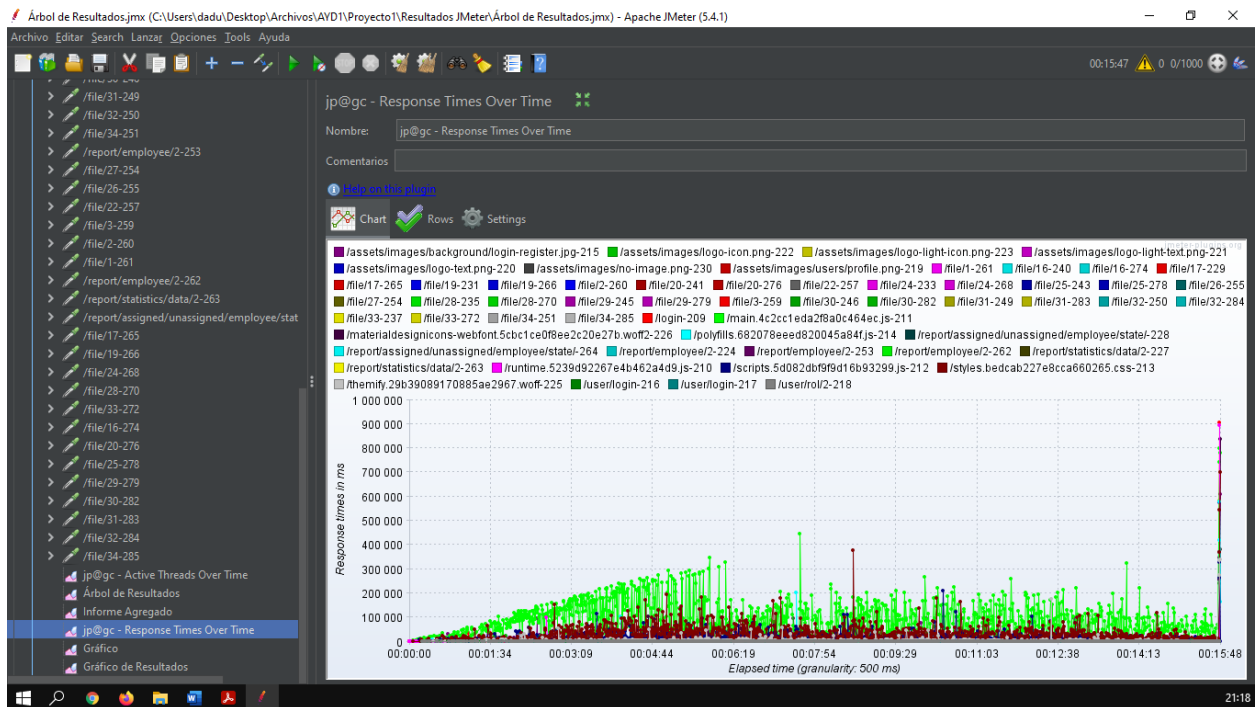
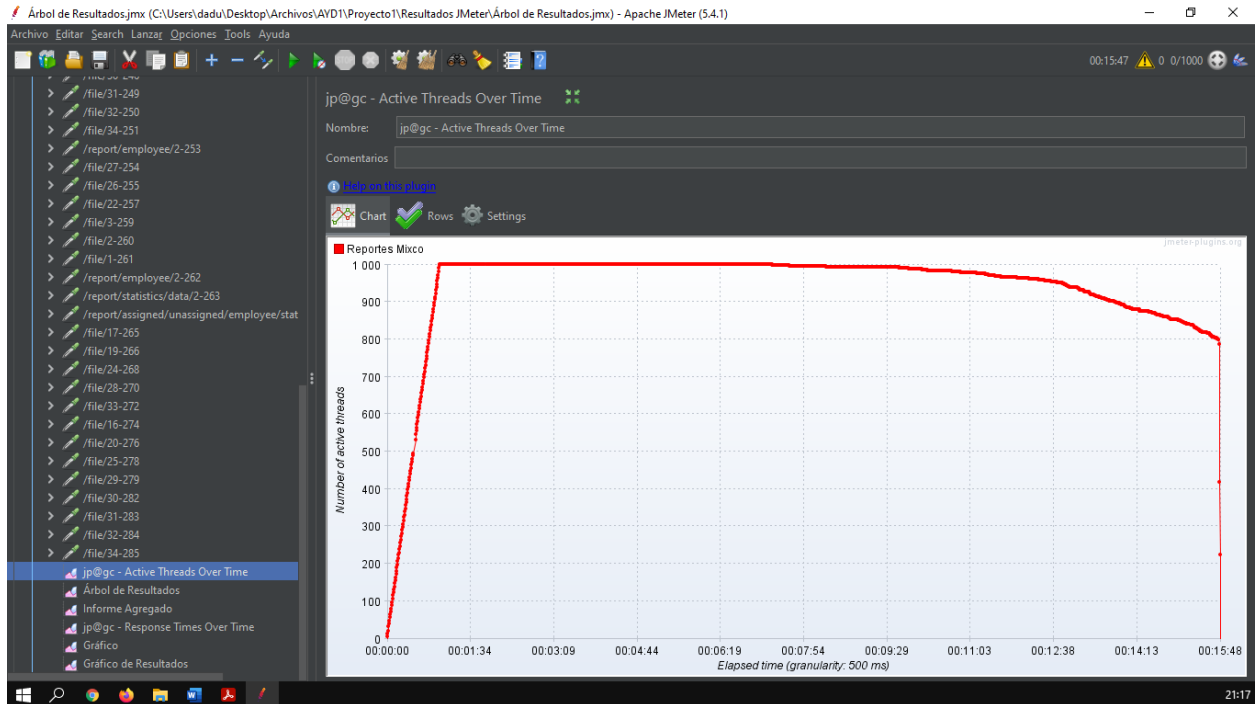




- Media



## 1000 usuarios



Milisegundos

