

# Arep-Lab05

Sergio Alejandro Nuñez Mendivelso

Febrero 2020

## 1 Introducción

En este documento explicaremos la funcionalidad de un servidor web que soporta múltiples solicitudes seguidas concurrentes y al ingresar una solicitud en el Browser de algún archivo .js, .jpg o .html este devuelve este archivo sin ningún problema.

## 2 Diseño

El diseño de este servidor web basado en POJOS se utilizó el lenguaje de programación Java, también se utilizó maven. Se implementó una clase (Controla) la cual es la encargada de la lógica de la aplicación y además gestionar el pool de hilos para resolver las peticiones hechas de manera concurrente, luego se implementó una anotación @Web para identificar los métodos que se van a publicar a través del framework IoC. Dichos métodos son los llamados POJOS los cuales están identificados o marcados con la anotación @Web, donde la clase Controla hace uso de la clase HttpServer que es la encargada de gestionar los recursos solicitados al servidor.

## 3 Conclusiones

- Heroku es una herramienta muy útil para este tipo de aplicaciones ya que es fácil de usarla y permite tener desplegada una aplicación de este tipo para probarla.
- Se investigó y se aprendió a conectar una base de datos por Java para que quede a la disposición de la aplicación.
- Se entendió como se usaba y se aprendió el uso de un WebSockets para utilizarlos en una aplicación Web.
- El servidor es capaz de recibir múltiples solicitudes de forma concurrente.
- El servidor es capaz de entregar páginas html e imágenes de tipo .jpg .
- se puede desplegar una aplicación en una máquina AWS y que sirva perfectamente desde esta máquina.

```

sergio.nunez@sisistemas61 MINGW64 ~/Downloads/SergioApp/dist
$ java -jar "SergioApp.jar" http://www.google.com
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="es-419"><head><meta content="
text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleg_sta
ndard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="y7eVIt/C9VMbn2xciA8Mng==">(fun
ction(){window.google={kEI:'dMtmXu2GMczj_AbXib3YDQ',kEXPI:'0,769493,584253,5663,730,224,5104,207,3204,10,2
90,761,175,364,1118,317,4,60,742,75,383,140,106,5,306,462,335,251,82,438,270,22,1126954,1197784,240,125,41
,329027,1344,12383,4855,32692,15247,867,6057,22627,369,524,8295,8384,4859,1361,4323,4968,3020,7651,204,792
5,1808,4020,978,7626,305,5297,2054,920,873,1217,2975,6430,11308,2882,20,317,1981,2539,1396,1380,518,400,22
77,8,2796,1593,1279,1042,1170,202,328,149,1103,840,517,1466,8,48,158,662,3438,312,1137,2,2063,606,1839,184
,1777,520,1947,729,18,429,1043,103,328,1284,16,447,2480,2246,474,1339,748,1039,15,3214,771,2073,6,817,503,
3491,1,2821,4682,1831,2663,641,2449,2459,1226,1462,280,3654,1275,108,1246,1681,480,908,2,1473,440,1642,141
3,719,265,2893,331,2195,225,280,716,828,842,185,2,293,1767,188,3,346,201,29,156,814,183,388,40,133,121,119
7,471,201,16,127,538,924,445,706,150,2171,1328,167,1294,508,520,28,829,2,116,58,354,569,23,30,243,121,38,4
9,391,91,124,381,1211,192,28,893,424,951,188,82,1017,673,554,178,57,259,4,370,4,280,936,73,154,663,141,15,
534,294,2,578,446,5837644,1805894,4194805,163,2799731,1323,549,333,444,1,2,80,1,900,896,1,8,1,2,2551,1,748
,141,59,736,563,1,4265,1,1,1,1,137,1,879,9,305,641,5,76,20,3,1,184,3,44,4,5,20743398,3220020,24',kBL:'ED6e
'};google.sn='webhp';google.kHL='es-419'}});(function(){google.lc=[];google.li=0;google.getEI=function(a
){for(var b;a&&(!a.getAttribute)||!(b=a.getAttribute("eid")));a=a.parentNode;return b}|google.kEI};google.

```

Figure 1: Vemos el resultado de la solicitud de google.

```
[ec2-user@ip-172-31-33-58 ~]$ java -jar SergioApp.jar https://lab-arep-05.herokuapp.com/Perdido.html
<!DOCTYPE html><html> <body> <h1>Estas Perdido</h1> </body></html>
[ec2-user@ip-172-31-33-58 ~]$ java -jar SergioApp.jar https://lab-arep-05.herokuapp.com/Perdid.html
java.io.IOException: Server returned HTTP response code: 503 for URL: https://lab-arep-05.herokuapp.com/Pe
rdid.html
[ec2-user@ip-172-31-33-58 ~]$ java -jar SergioApp.jar https://lab-arep-05.herokuapp.com/bienvenido.html
<!DOCTYPE html><html> <body> <h1>Bienvenido a la aplicacion</h1> </body></html>
[ec2-user@ip-172-31-33-58 ~]$ |
```

Figure 2: Vemos el resultado de la solicitud de nuestro servidor ya subido en heroku.

```
sergio.nunez@sistemas61 MINGW64 ~/Downloads
$ sftp -i "MyFirstKeySergio.pem" ec2-user@ec2-204-236-246-211.compute-1.amazonaws.com
Connected to ec2-user@ec2-204-236-246-211.compute-1.amazonaws.com.
sftp> ll
desktop.ini  MyFirstKeySergio.pem  SergioApp
sftp> ls
sftp> lpwd
Local working directory: /c/Users/sergio.nunez/Downloads
sftp> pwd
Remote working directory: /home/ec2-user
sftp> lcd SergioApp/
sftp> lcd dist/
sftp> ll
README.TXT  SergioApp.jar
sftp> put SergioApp.jar
Uploading SergioApp.jar to /home/ec2-user/SergioApp.jar
SergioApp.jar                                100% 2210    18.1KB/s   00:00
sftp> ls
SergioApp.jar
sftp> exit
```

Figure 3: pasamos la aplicación en un .jar de nuestra maquina local a nuestra maquina AWS.

```

public class ClienteAWS extends Thread {
    //Atributos
    private static URL url;
    private static int numHilos;
    private static ArrayList<Thread> listaHilos;

    public ClienteAWS(URL url) {
        this.url = url;
    }

    public static void main(String[] args) throws Exception {
        url = new URL(args[0]);
        numHilos = Integer.parseInt(args[1]);
        listaHilos = new ArrayList<Thread>();
        for (int i = 0; i < numHilos; i++) {
            listaHilos.add(new ClienteAWS(url));
        }
        int tem = 0;
        for (Thread h : listaHilos) {
            h.start();
            tem++;
        }
        System.out.println("Se ejecutaron " + tem + " solicitudes concurrentes.");
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(
                new InputStreamReader(url.openStream()));
            String inputLine = null;
            while ((inputLine = reader.readLine()) != null) {
                System.out.println(inputLine);
            }
        } catch (IOException x) {
            System.err.println(x);
        }
    }
}

```

Figure 4: evidencia del cliente concurrente en mi aplicación.