Arep-Lab05

Sergio Alejandro Nuñez Mendivelso

Febrero 2020

1 Introducción

En este documento explicaremos la funcionalidad de un servidor web que soporta múltiples solicitudes seguida 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 utilizo el lenguaje de programación Java, también se utilizo maven. Se implemento una clase (Controla) la cual es la encargada de la lógica de la aplicación y además gestionar el pool de hilos para resolver las peticione hechas de manera concurrente, luego se implemento 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 investigo 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 aprendo 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 paginas html e imagenes de tipo .jpg .

- se puede desplegar una aplicación en una maquina AWS y que sirva perfectamente desde esta maquina.

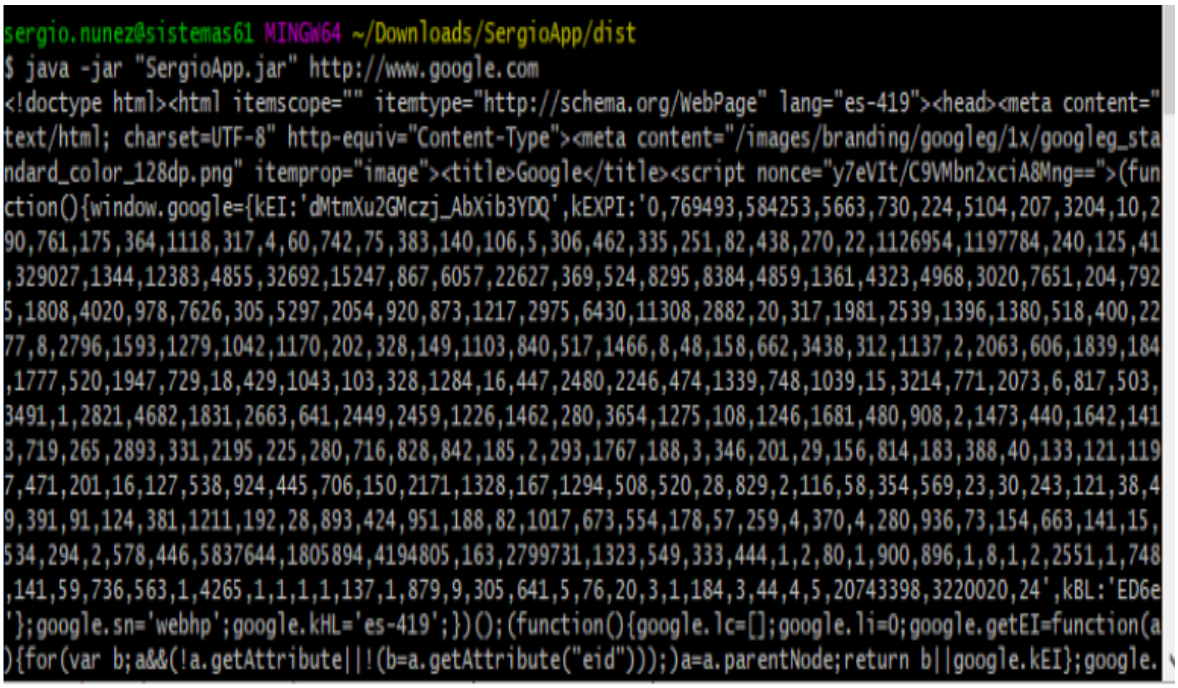

Figure 1: Vemos el resultado de la solicitud de google.

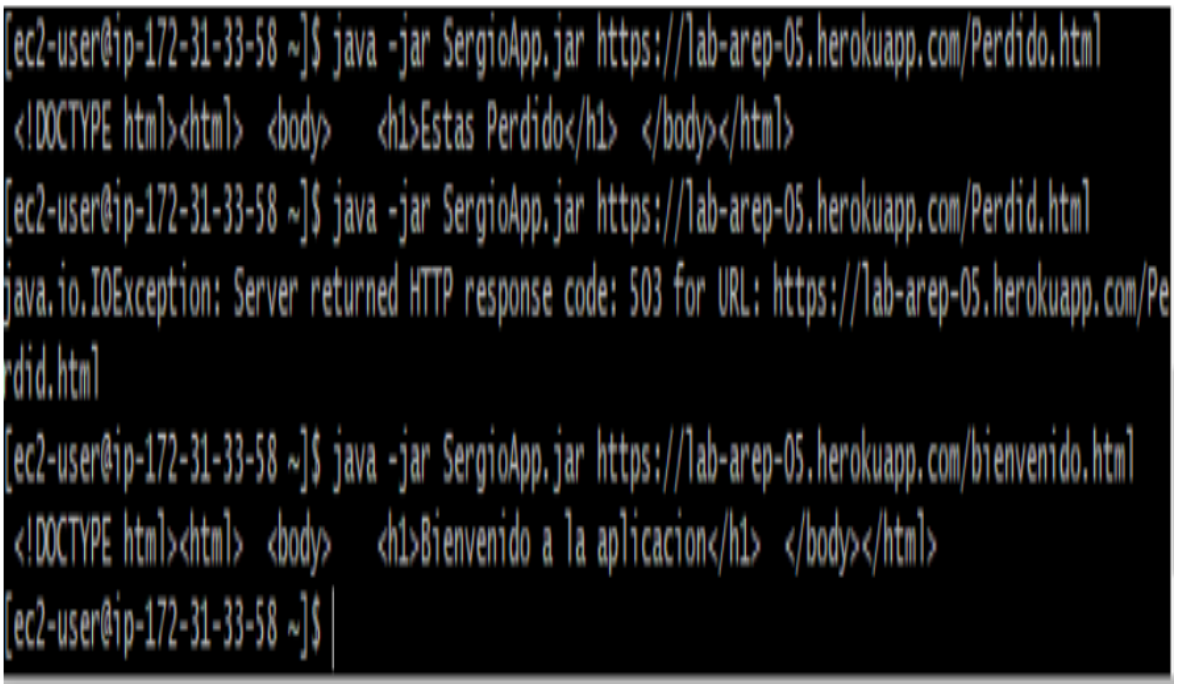

Figure 2: Vemos el resultado de la solicitud de nuestro servidor ya subido en heroku.

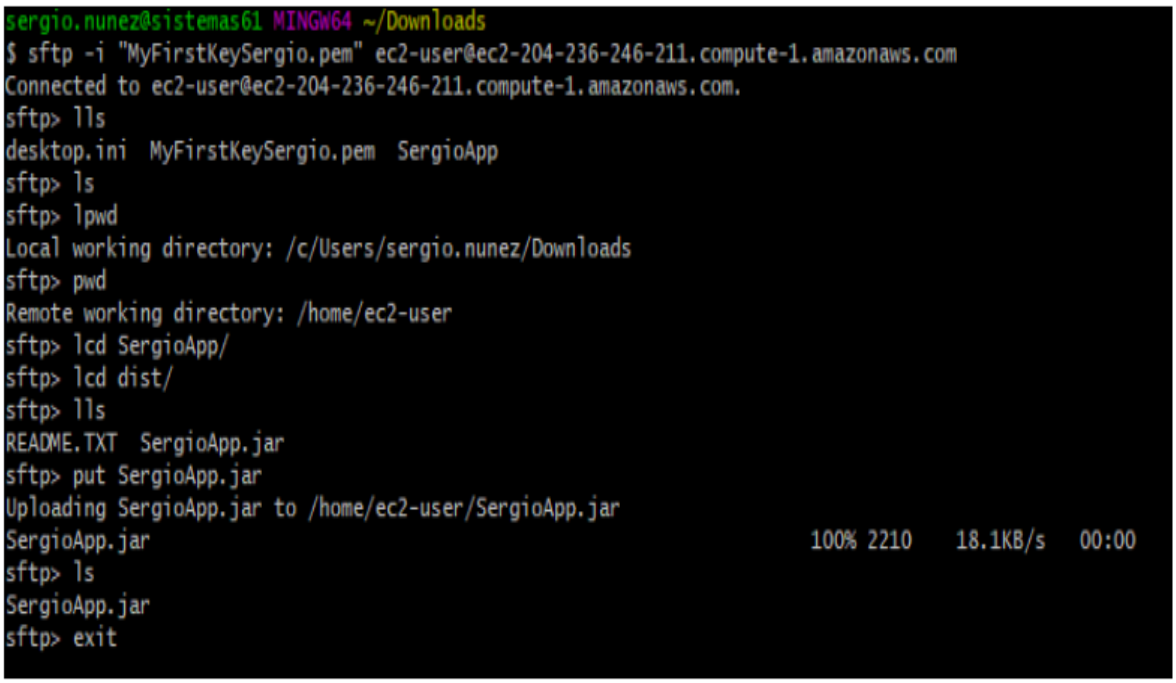

Figure 3: pasamos la aplicación en un .jar de nuestra maquina local a nuestra maquina AWS.

```java
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.