INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ALUMNO: Meza Vargas Brandon David.

PRÁCTICA: Práctica No. 6.

TEMA: Sockets Servidores.

OPCIÓN: Opción 1, ChatBot Básico o Nano Alexa

FECHA: 21-dic-2020

GRUPO: 2CM1

MATERIA: Programación Orientada a Objetos

INTRODUCCIÓN

La programación server-socket es casi tan simple como la programación cliente-socket. Una sola clase ServetSocket es usada para crear y administras conexiones de sockets clientes. El ServerSocket se conecta a un puerto y espera por nuevas conexiones de clientes. Cuando una nueva conexión de un cliente es recibida, una instancia de la clase Socket es creada por la instancia ServerSocket y es usada para comunicarse con el cliente remoto.

La clase ServerSocket provee de constructores y métodos útiles para conectar un socket server con una IP local y un respectivo puerto. Los constructores son usados para definir la dirección IP, el puerto local y la conexión reserva parámetros que serán usados. Los métodos restantes se usan para recibir nuevas conexiones TCP, refinar aspectos de instancias del socket recién creadas, determinar el estado de conexión y para cerrar el socket.

DESARROLLO

Para esta practica se escogió la opción numero 1 la cual consiste en un ChatBot básico, en donde un servidor puede almacenar al menos 10 preguntas y respuestas predefinidas, además de poder darnos la fecha actual y contarnos un chiste al azar de 10 disponibles.

Primero comenzaremos con el cliente, en primer lugar, definimos las variables que usaremos en nuestro cliente, ver figura 1;

```
Socket conexion;

PrintStream salida;
DataInputStream entrada;
String res;

JTextField pregun;
JLabel respuestas;
JButton enviar;
```

Figura 1. Variables a usar en el cliente.

En el constructor del cliente establecemos los elementos que tendrá nuestra interfaz gráfica, aquí mismo mandamos a llamar al método init() que nos sirve para añadir propiedades a los elementos de nuestra gui, así como agregarlos al frame. La figura 2 muestra el constructor.

```
public ChatClient()
{
    super("ChatBoT Basico");

    pregun = new JTextField("Pregunta algo...");
    respuestas = new JLabel("Hola!");
    enviar = new JButton("Preguntar");

    enviar.addActionListener(this);

    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    setSize(600,300);
    setVisible(true);
    init();
}
```

Figura 2. Contructor.

En al figura 3 podemos ver el método init();

```
void init(){
    setLayout(null);

    respuestas.setBounds(0,150,550,30);
    pregun.setBounds(0,200,450,30);
    enviar.setBounds(460,205,100,20);
    add(respuestas);add(pregun);    add(enviar);
}
```

Figura 3. Método init().

Al momento de enviar o preguntar algo a nuestro servidor por medio del botón preguntar hacemos la conexión con el servidor, todo esto dentro de actionPerformed() ya que nuestra clase implementa la interfaz ActionListener.

```
try
{
    conexion = new Socket("localhost", 5000);
    entrada = new DataInputStream(conexion.getInputStream());
    salida = new PrintStream(conexion.getOutputStream());

    salida.println(pregun.getText());
    respuestas.setText(pregun.getText());
    pregun.setText(null);
    res = entrada.readLine();
    respuestas.setText(res);
}
```

Figura 4. Try.

En la figura 4 se muestra nuestro try, en donde realizamos la conexión con el servidor, además de establecer los flujos de entrada y salida, en este caso, nuestros flujos de salida son las preguntas que se enviaran al servidor y los de entrada las respuestas que el mismo servidor nos da.

Para ver las respuestas en pantalla, primero leemos la entrada en la línea *entrada.readLine()* y, posteriormente, actualizamos el texto de nuestro Label con la línea *respuestas.setText(res);*

Este código lo ponemos en un try ya que puede lanzar una excepción. Nuestro catch correspondiente lo podemos ver en la figura 5;

```
catch (Exception e)
{
    respuestas.setText("Error: " + e.getMessage());
    System.out.print("Error: " + e.getMessage());
}
```

Figura 5. Catch

Nuestro main se muestra en la figura 6;

```
public static void main(String args[])
    {
        new ChatClient();
    }
```

Figura 6. Main del cliente

Ahora veamos nuestro servidor.

En nuestro servidor almacenamos las preguntas, las respuestas, así como los chistes que puede contar, ver figura 7;

```
String preguntas[]= {
    "Como te llamas?",
    "En que ciudad vives?",
    "Tienes sentimientos?",
    "Quien te creo?",
    "Harias mi tarea?",
    "Que haces?",
    "Cual es tu edad?",
    "Dame la fecha",
    "Tienes un pasatiempo?"
};

String respuestas[]={
    "Pedrito Sola",
    "No en una ciudad, pero vivo en el corazon de todos",
    "Los robots no tenemos sentimientos",
    "El que hizo esta practica, Brandon Meza",
    "No lo creo jaja",
    "Viendo que me preguntas",
    "Tengo una semana",
    "Claro, estamos a "+ new Date(),
    "Navegar por la red"
};

ring chistes []={
 "Habia un perro de goma que cuando se rascaba se borraba.",
 "Habia una vez un hombre tan pequeño que se subio encima de una canica y dijo: El mundo es mio!",
 "- Sabes que mi hermano anda en bicicleta desde los cuatro anios? Mmm, ya debe estar lejos.",
 "- Pedrito, que planeta va despues de Marte? Miercoles",
 " Cual es el pez que huele mucho? El Peztoso!!!",
 " Mama, en el colegio me llaman distraido. Juanito, tu vives en la casa de enfrente",
 "Con que juega un bebe vampiro? Con globulos rojos.",
 "Que hace un vampiro conduciendo un tractor? Sembrar el miedo.",
 "En que se parece una bruja y unos dias de vacaciones? En que los dos se van volando.",
 "Como se dice perro en ingles? -Dog. -Y como se dice veterinario? -Muy facil. Dog-tor."
```

Figura 7. Chistes, preguntas y respuestas.

Además de tener las variables que usaremos para iniciar el servidor y mandar las respuestas al cliente (ver figura 8).

```
Random r= new Random();
int aux = 0;
ServerSocket socketServidor;
Socket conexion;

PrintStream salida;
DataInputStream entrada;

String pre;
```

Figura 8. Variables a usar en el servidor.

De manera seguida, tenemos el método go() con el cual vamos a iniciar el servidor.

Primeramente, tenemos un try que contiene un ciclo while infinito, dentro de este ciclo esperamos a que un cliente se conecte, además de recibir las preguntas que hizo el cliente, esto lo vemos en la figura 9.

```
conexion = socketServidor.accept();
entrada = new DataInputStream(conexion.getInputStream());
salida = new PrintStream(conexion.getOutputStream());
pre = entrada.readLine();
```

Figura 9. Primera parte de nuestro ciclo while dentro del try,

Para los chistes, tenemos un if que verifica que la pregunta sea igual a “cuenta un chiste”, de esta forma escoge un chiste al azar de los 10 chistes almacenados mostrados en la figura 7, (ver figura 10).

```
if(pre.equals("Cuenta un chiste")){
    int n = r.nextInt(11);
    salida.println(chistes[n]);
}
```

Figura 10. If que selección un chiste.

Para las preguntas, tenemos un ciclo for que recorre las preguntas almacenadas hasta que coincida con la pregunta que realizo el cliente, de esta forma arroja la respuesta correcta a la pregunta del cliente, si el cliente realiza una pregunta que no esta almacenada, el servidor no responde a la pregunta. (ver figura 11).

```
for (int i=0;i<9;i++)
{

    if(pre.equals(preguntas[i]))
        salida.println(respuestas[i]);
    else
        aux = 1;

}

if(aux== 1)
    salida.println("No tengo una respuesta :(");
```

Figura 11. For para las preguntas.

Por último, tenemos el catch correspondiente, lo vemos en la figura 12.

```
catch(Exception e)
{
    System.out.println("Error: " + e.getMessage());
}
```

Figura 12. Catch del servidor.

Finalmente, nuestro main lo vemos en la figura 13;

```
public static void main(String args[]){
    new ChatServidor().go();
}
```

Figura 13. main del servidor.

Al correr nuestro programa, se ve como se muestra en la figura 14;

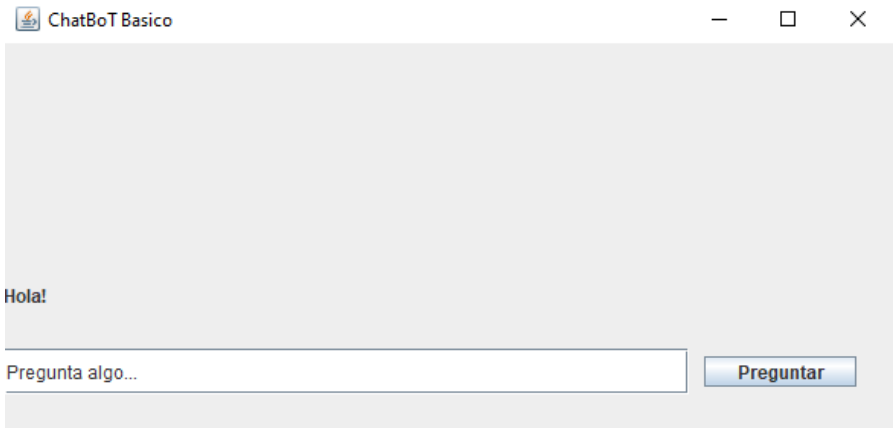

Figura 14. Nuestro programa corriendo.

A continuación, algunas capturas del funcionamiento de nuestro programa, (figuras 15-19)

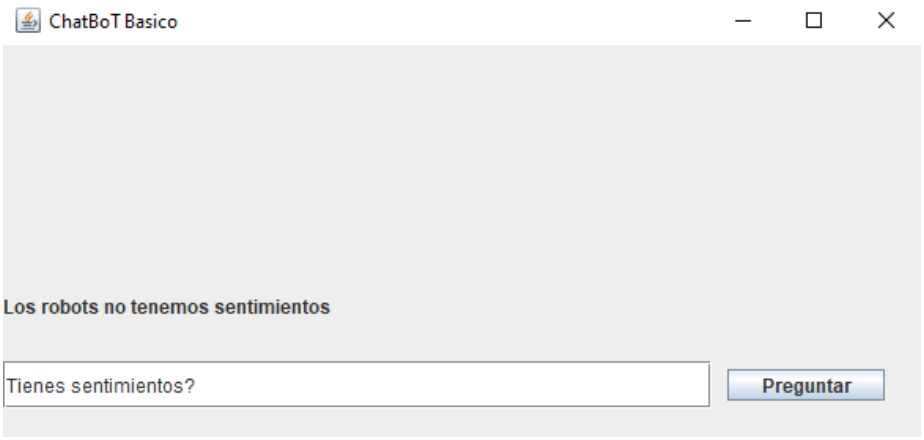

Figura 15. Ejemplo del programa.

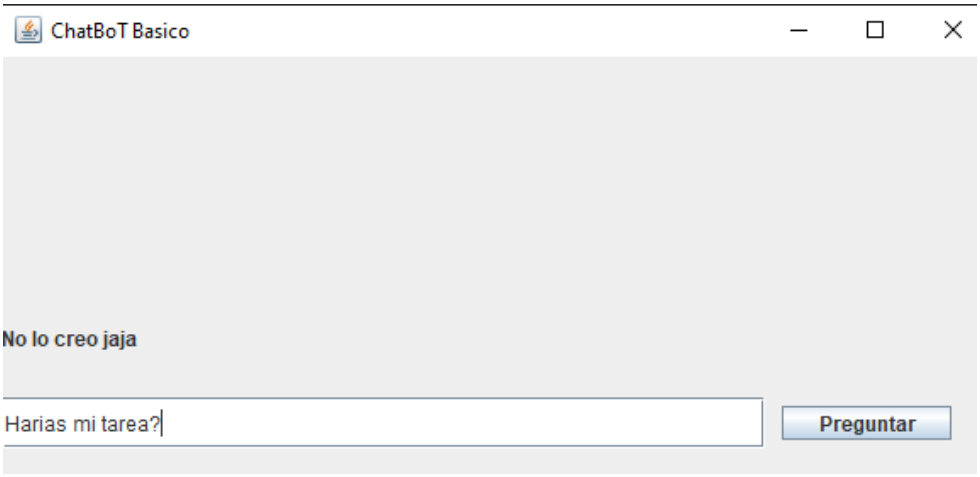

Figura 16. Ejemplo del programa.

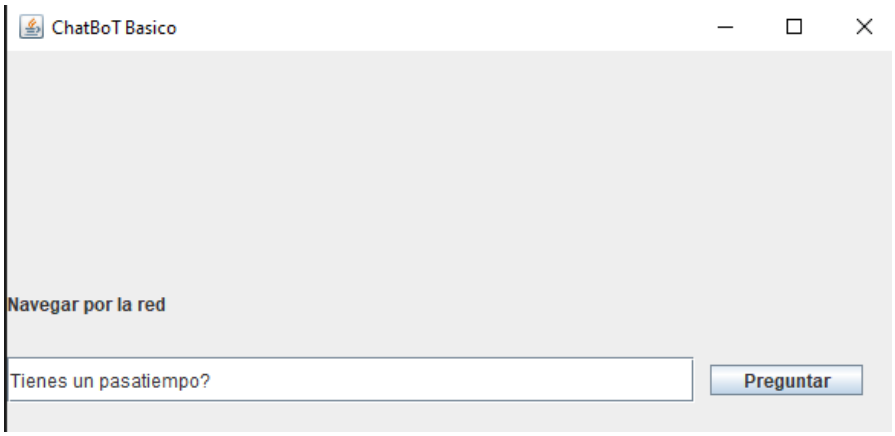

Figura 17. Ejemplo del programa.

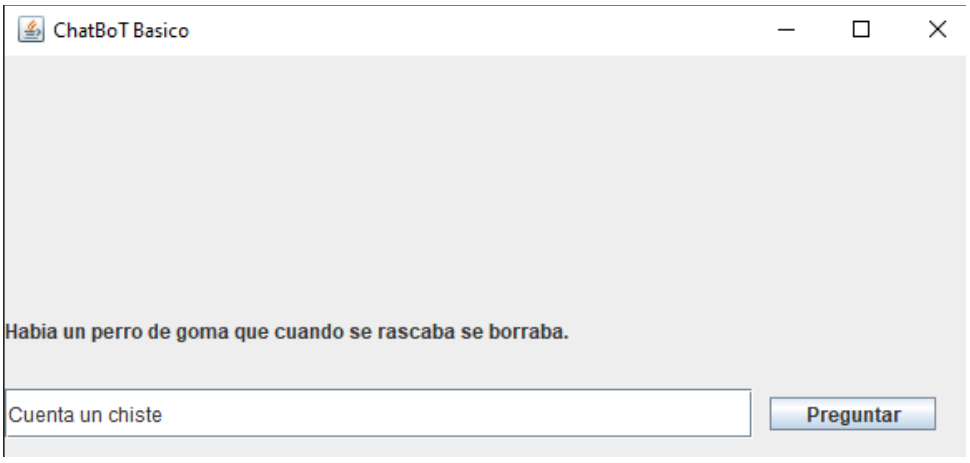

Figura 18. Ejemplo del programa.

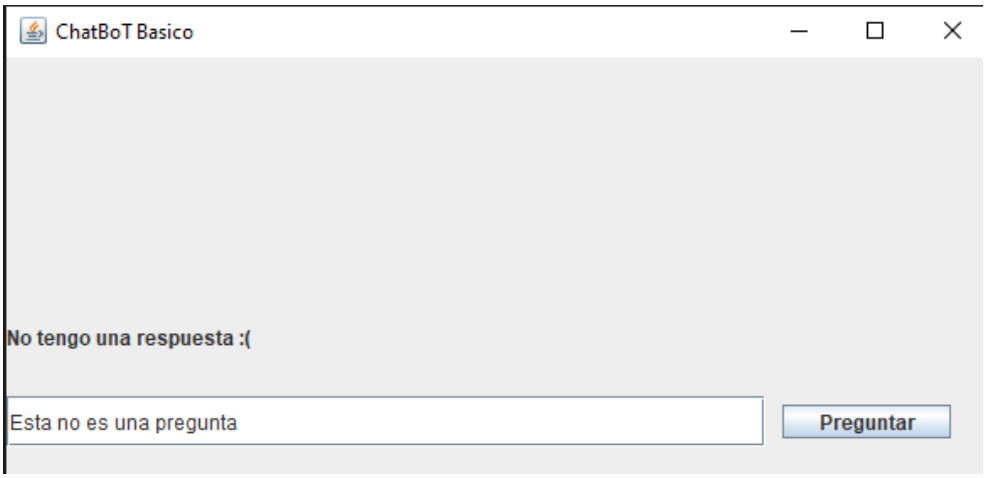

Figura 19. Ejemplo del programa.

CONCLUSIÓN

A partir de la practica realizada puedo concluir que realizar un programa en la red es muy importante a día de hoy, pues muchas veces se requerirá que un programa sea usado por varios clientes y no solo de manera local.

Además, vimos que gracias los sockets podemos establecer un enlace entre dos programas que se ejecutan independientes el uno del otro, generalmente un cliente y un servidor, como fue el caso en esta práctica.

Gracias a esta práctica reforcé un poco mas lo visto en las clases sobre server sockets, además de retomar elementos anteriores como lo fue la gui.