

# UNIVERSIDAD NACIONAL

Facultad de Ciencias Exactas y Naturales

ESCUELA DE INFORMÁTICA



## “Primer Proyecto Programado”

**Profesor:** Armando Arce

José Gerardo Marín Fernández

1 15920602

[geramarinfer03@gmail.com](mailto:geramarinfer03@gmail.com)

Josué Valerio Ramírez

4 02270021

[josvr0511@gmail.com](mailto:josvr0511@gmail.com)

Mayo 2017  
Heredia, Costa Rica

---

## **Introducción**

En el siguiente documento se detalla la documentación del primer proyecto programado del curso Sistemas distribuidos impartido en el segundo ciclo del 2017, por el profesor Armando Arce, en la Escuela de Informática de la Universidad Nacional de Costa Rica.

En el documento se presenta: la descripción del proyecto, brindada por el profesor del curso, en esta se establecen las pautas para su realización. Además se expondrá las estructuras de datos utilizadas por cada uno de los distintos procesos (Jugador, Emisor, Difusor) así como la descripción de cada uno de estos últimos.

Se describirán los mecanismos de creación y comunicación entre los procesos. También se brindarán pruebas de ejecución de los procesos mediante capturas de pantalla.

## **Descripción del problema (este enunciado)**

El objetivo de este proyecto es poner en práctica el esquema de suscripción-publicación mediante la creación de la simulación de un juego de BINGO. Para ello será necesario crear tres tipos de procesos: el proceso jugador, el proceso difusor y el proceso emisor. La simulación podrá involucrar un proceso emisor, un proceso difusor y muchos procesos jugadores.

Toda la programación se debe realizar en Lua y utilizando la librería zmq para comunicar los diferentes procesos.

### **El proceso jugador**

El proceso jugador debe iniciar seleccionando en forma aleatoria los números de su cartón y almacenarlos posiblemente en una matriz. El cartón consiste de cinco filas y cinco columnas. Tenga en cuenta que los números de la primera columna van del 1 al 15, la segunda del 16 al 30, la tercera del 31 al 45, la cuarta del 46 al 60, y la quinta del 61 al 75. Sin embargo, la posición central del bingo queda vacía. Se debe verificar que no se repitan números en la misma columna.

Posteriormente el proceso jugador se debe suscribir al proceso difusor y esperar a que le sean enviados los números jugados. Cada vez que llegue un mensaje con un número jugado el proceso debe verificar si existe en su cartón y de ser así lo debe marcar. Posteriormente debe imprimir el cartón en la consola mostrando el estado actual del mismo. Si un proceso logra completar (marcar) todos los números de su cartón debe imprimir en la pantalla el mensaje "BINGO !!!" y enviar un mensaje al proceso difusor para informarle al resto de procesos jugadores que el juego terminó.

### **El proceso emisor (locutor)**

El proceso emisor o locutor simplemente genera números aleatorios entre 1 y 75 que va enviado periódicamente al proceso difusor. Note que los mensajes deben ser enviados con cierto intervalo de tiempo que permita que todos los procesos jugadores puedan recibirlos y procesarlos. Además, que debe tener especial cuidado de no volver a generar números repetidos, para ello debe llevar una lista de los números que ya fueron "cantados".

Cada vez que el proceso emisor genere un nuevo número debe imprimir en la consola de lista de todos los números que ha "cantado" previamente.

### **El proceso difusor**

El proceso difusor debe recibir los mensajes provenientes del proceso emisor y difundirlos entre los diferentes procesos jugadores que se estén ejecutando. Este debe ser el primer proceso que se ponga a correr, luego los procesos jugadores y por último el proceso emisor. Note que adicionalmente existe un mensaje especial que se debe procesar y es el mensaje de "BINGO". Sin embargo, usted puede codificarlo mediante cualquier otro número (p.ej. 0 ó -1), así el proceso difusor simplemente lo debe pasar y los procesos jugadores saben cómo interpretarlo.

### **Definición de estructuras de datos**

**Emisor:** Utiliza dos estructuras de tipo lista.

1. Bingo: Lista de números que contiene todas las bolas del 1 al 75, las cuales son tomadas de forma aleatoria para el desarrollo del juego.
2. Cantados: Esta lista almacena las bolas que ya han sido cantadas por el emisor y enviadas a los jugadores. Se muestra en pantalla para conocer el progreso de las bolas que ya han salido en el juego.

**Difusor:** No utiliza ninguna estructura de tipo lista o matriz.

**Jugador:** Utiliza una estructura tipo lista y un tipo matriz.

1. Array: Lista de números que contiene el cartón, se utiliza para evitar colocar números repetidos en las casillas del cartón del jugador.
2. Cartón: Matriz para representar el cartón del jugador. Contiene los números del jugador, y se muestra en pantalla para mostrar el proceso del jugador.

## Descripción detallada y explicación de los componentes principales del programa:

### Emisor:

Cumple la función de tómbola, es el proceso encargado de contener bolas del juego (1 al 75), las cuales son enviadas al difusor de forma periódica.

```
function cargarbolas()
  for i =1, 75 do
    bingo[#bingo+1]=i
  end
end
```

Contiene una lista con las bolas del juego. El cual se carga a partir de una función *cargarBolas()*. La cual llena la lista de números.

```
function generarBola()
  espacio = math.random(1, random)
  bolita = bingo[espacio]
  insertarCantados(bolita)
  table.remove(bingo, espacio)
  random = random -1

  return bolita
end
```

El emisor se encarga de generar bolas de forma aleatoria, a partir de la función *generarBola()*. Esta función toma las bolas de forma aleatoria de la lista del bingo y posteriormente las remueve para evitar números repetidos.

```
function imprimirCantados()
  os.execute("cls")
  row=""
  for i=1, #cantados do
    row= row.."[" ..cantados[i].." ]"
  end

  print("Bolas cantadas: "..row)
end
```

El emisor contiene una lista de bolas que ya han sido cantadas. Las mismas son mostradas en pantalla para observar el proceso del juego. Se imprimen a partir de la función *imprimirCantados()*.

```
local jugar = true
cargarbolas()

while jugar do
  s_sleep(1000)
  local bola
  bola = generarBola()
  publisher:send(""..bola)
  imprimirCantados()

  local mensaje = subscriber:recv()

  if mensaje == "0" then
    jugar = false
  end
end
```

El proceso emisor se compone por un ciclo principal (While), el cual se encarga de repetir las acciones principales: generar bola, imprimir cantados, enviar al difusor, esto lo realiza de forma periódica para mantener el desarrollo del juego. Este ciclo finaliza cuando el difusor le informa que algún jugador ganó el juego.

## Difusor:

El difusor tiene como objetivo cantar los números del Bingo, este componente cumple con dos funciones:

1- Recolectar la cantidad de jugadores que se conectan antes de iniciar una partida de Bingo, esto lo hace utilizando el método *suscribirJugadores()*

```
function suscribirJugadores()
  while escuchar do
    local request = socket:recv()

    if request == "play" then
      subs = subs + 1
      print("Jugador conectado")
    end

    if request == "init" then
      print("Comienza el juego")
      escuchar = false
      jugar = true
    end
    socket:send("ok")
  end
end
```

Al ejecutar este proceso de primero, el método de *suscribirJugadores* inicia, por cada jugador que se conecte, este enviará un mensaje a este proceso, el cual tomará ese mensaje, si el mensaje es un "play" significa que se conecto un nuevo jugador, por lo tanto sumará 1 a la variable subs. En caso de que el mensaje sea "init" esto significa que el que envió el mensaje fue el proceso Emisor y no un Jugador, por lo tanto este método finaliza.

La segunda función que realiza este proceso es la de "cantar" los números que el proceso Emisor envía, esto lo procesa en el método llamado *cantar()*

```
function cantar()
  print("Comienza a cantar a "..subs.." jugadores")
  while jugar do

    local bola = subscriber:recv()
    os.execute("cls")
    print("Bola #: "..bola)
    publisher:send(" "..bola)

    for i=1, subs do
      local request = socket:recv()
      if request == "1" then
        jugar = false
        os.execute("cls")
        print("Juego terminado tenemos ganador")
      end
      socket:send("ok")
    end

    if jugar == false then
      publisher:send("0")
    end
  end
end
```

Este método informa a todos los jugadores cuál fue el número cantado, además realiza la comprobación de que ningún jugador haya enviado un mensaje de que ya ganó, en caso de que un jugador informe, este método detiene se detiene e informa antes al emisor y a los demás jugadores de que el juego finaliza porque ya se tiene un ganador.

### Jugador:

Este proceso se encarga del control de juego en el cartón del jugador. Se encarga de revisar si el número jugado se encuentra o no en el cartón del jugador, además de notificar al Locutor (Difusor) en caso de ganar la partida de Bingo.

Este Proceso utiliza la librería “cartonBingo”, la cual presenta los siguientes métodos.

### llenarCarton(matriz, array)

```
function llenarCarton(m,a)
  --cambie el random, pero lo hace cada segundo
  math.randomseed(os.time())
  mitad = math.modf(#m/2) +1
  -- Mueve filas
  for i=1, #m do
    --Mueve Columnas (rangos)
    for j=1, #m[1] do
      if (i == mitad and j == mitad) then
        m[j][i] = 0
      else
        m[j][i] = generarNumero(a,rangos[j]+1,rangos[j+1])
      end
    end
  end
end
```

Este método genera un cartón de Bingo para el jugador, esto mediante números aleatorios. El centro del cartón del Bingo debe de ir vacío, para lograr esto se inserta un “0” en el centro de la matriz.

Este método recibe como parámetro un cartón vacío, con la estructura del mismo, y un array que contiene los números del cartón de Bingo.

### generarNumero(array, menor, mayor)

```
function generarNumero(array,menor,mayor)
  local valor
  band = true
  while band do
    valor = math.random(menor, mayor)
    if revisarRepetidos(array,valor) == false then
      array[#array+1] = valor
      band=false
    end
  end
  return valor
end
```

Este método se encarga de generar los números aleatorios que el método llenarCarton utiliza.

Esta librería posee un array llamado “rangos” el cual contiene los números permitidos en cada columna, según las reglas básicas del Bingo, donde la primera columna del cartón contiene los números del 1 al 15, la

segunda de 16 al 30, y así sucesivamente. Estos rangos son utilizados como valores mínimo y máximo para generar el número aleatorio.

### revisarRepetidos(array, numero)

```
function revisarRepetidos(array,num)
  for i=1,#array do
    if tonumber(array[i])==tonumber(num) then
      return true
    end
  end
  return false
end
```

Se encarga de revisar si el número generado ya se encuentra en el cartón de Bingo del jugador.

### actualizarCarton(carton, bola)

```
function actualizarCarton(c,b)
    bol = tonumber(b)
    columna = seleccionarColumna(bol)

    colum_selected = c[columna]

    for i = 1, #colum_selected do
        if colum_selected[i] == bol then
            colum_selected[i] = -1
            return true
        end
    end

    return false
end
```

La función de este método es actualizar el carton del jugador por cada número que el Locutor cante y este en el cartón.

```
local jugar = true
while jugar do

    local bola = subscriber:recv()
    --os.execute("cls")

    if bola ~= "0" then
        print("Bola #: "..bola)

        if actualizarCarton(carton,bola)==true then
            acertadas = acertadas +1
        end

        imprimirCarton(carton)

        if revisarCarton(acertadas) == true then
            print("  -----")
            print(" | BINGO0000000 |")
            print("  -----")
            socket:send("1")
        else
            socket:send("0")
        end

        socket:recv()

    else
        print("juego terminado")
        jugar = false
    end
end
```

En el archivo Jugador.lua se encuentra el método del suscriptor, el cual escucha las bolas cantadas por el Locutor (difusor) y utiliza los métodos de la librería cartonBingo.lua antes mencionados para hacer las verificaciones y procesos pertinentes para desempeñar el juego de Bingo.



## Mecanismo de creación y comunicación de procesos

### Emisor – Difusor:

- Se utilizó una comunicación de sockets publicador y suscriptor (Sub-Pub). Donde el emisor abre la conexión por medio del puerto 5555, y el difusor se suscribe. Por medio de esta comunicación el emisor envía las bolas al difusor de forma periódica. Para este proceso solo se necesita una comunicación unidireccional.

```
local publisher = context:socket(zmq.PUB)
publisher:bind("tcp://*:5555")
```

- También se creó una comunicación de tipo Solicitud y Respuesta (Req-Rep). Donde el difusor abre la conexión por medio del puerto 5557 realizando el papel de servidor. El emisor envía una solicitud al difusor para saber si puede iniciar la partida de Bingo.

```
local socket = context:socket(zmq.REQ)
socket:connect("tcp://localhost:5557")
```

- El emisor se suscribe al proceso difusor por medio del puerto 5556, esperando que el difusor le notifique cuando el Juego termina.

```
local subscriber = context:socket(zmq.SUB)
subscriber:connect("tcp://localhost:5556")
subscriber:setopt(zmq.SUBSCRIBE, "")
```

### Difusor – Jugadores.

- Se creó una comunicación principal de tipo publicador y suscriptor (Sub-Pub) por medio de Sockets, el difusor abre dicha conexión por medio del puerto 5556 y los jugadores se suscriben para unirse a la partida. El difusor transmite la bola que proviene del emisor a todos los suscriptores (jugadores). Comunicación unidireccional. Por medio de esta comunicación el difusor también notifica a los jugadores y el emisor cuando el juego termina.

```
local publisher = context:socket(zmq.PUB)
publisher:bind("tcp://*:5556")
```

- Se creó una comunicación de tipo Solicitud y Respuesta (Req-Rep) con los jugadores, el difusor abre esta comunicación por medio del puerto 5557. Esta comunicación bidireccional permite que un jugador notifique al difusor cuando ganó el juego, permitiendo que el difusor notifique a los demás procesos y detenga el juego.

```
local socket = context:socket(zmq.REP)
socket:bind("tcp://*:5557")
```



## Significado de mensajes

### Emisor

Mensaje de conexión al difusor e iniciar partida.

```
19 socket:send("init")
```

Enviar bola al difusor.

```
bola = generarBola()  
publisher:send("".bola)
```

Mensaje recibido del difusor para saber si termino la partida.

```
local mensaje = subscriber:recv()
```

### Difusor

Mensaje recibido de jugadores y emisor para notificar conexión.

```
local request = socket:recv()
```

Mensaje para aceptar conexión a jugadores y emisor.

```
socket:send("ok")
```

Enviar bola a jugadores suscritos al difusor.

```
publisher:send("".bola)
```

Mensaje para notificar a jugadores y emisor que el juego termino.

```
publisher:send("0")
```

### jugador

Mensaje enviado al difusor para notificar conexión al juego.

```
socket:send("play")
```

Bola recibida desde el difusor, en caso de ser 0 indica que el juego ha terminado.

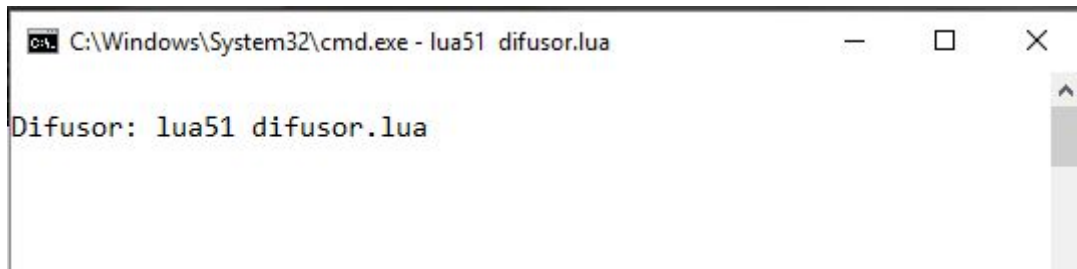
```
local bola = subscriber:recv()
```

Mensaje del jugador para notificar al difusor que ha ganado el juego.

```
socket:send("1")
```

## Pruebas de ejecución

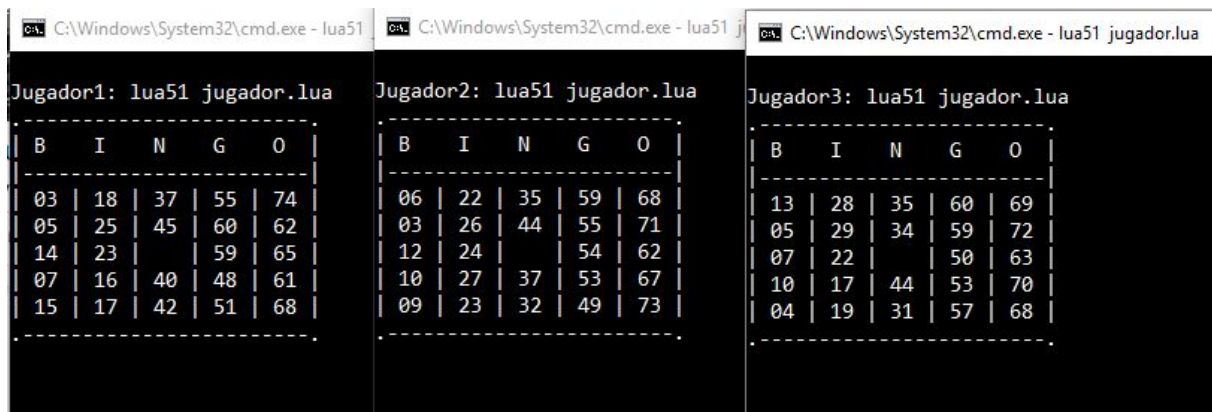
Para ejecutar el sistema de Bingo se deben de ejecutar los procesos en orden, siendo el primer proceso a ejecutar el de Difusor.lua



```
C:\Windows\System32\cmd.exe - lua51 difusor.lua

Difusor: lua51 difusor.lua
```

Seguidamente se deben de ejecutar los procesos Jugadores, para esta prueba se ejecutan tres procesos Jugador.lua en ventanas CMD distintas, al ejecutar este proceso se nos muestra el cartón de Bingo del jugador.

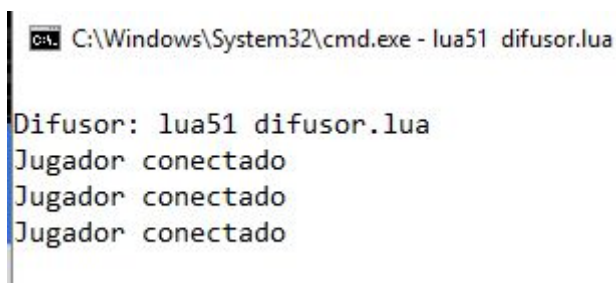


```
C:\Windows\System32\cmd.exe - lua51 jugador.lua
Jugador1: lua51 jugador.lua
+-----+
| B | I | N | G | O |
+-----+
| 03 | 18 | 37 | 55 | 74 |
| 05 | 25 | 45 | 60 | 62 |
| 14 | 23 |   | 59 | 65 |
| 07 | 16 | 40 | 48 | 61 |
| 15 | 17 | 42 | 51 | 68 |
+-----+

C:\Windows\System32\cmd.exe - lua51 jugador.lua
Jugador2: lua51 jugador.lua
+-----+
| B | I | N | G | O |
+-----+
| 06 | 22 | 35 | 59 | 68 |
| 03 | 26 | 44 | 55 | 71 |
| 12 | 24 |   | 54 | 62 |
| 10 | 27 | 37 | 53 | 67 |
| 09 | 23 | 32 | 49 | 73 |
+-----+

C:\Windows\System32\cmd.exe - lua51 jugador.lua
Jugador3: lua51 jugador.lua
+-----+
| B | I | N | G | O |
+-----+
| 13 | 28 | 35 | 60 | 69 |
| 05 | 29 | 34 | 59 | 72 |
| 07 | 22 |   | 50 | 63 |
| 10 | 17 | 44 | 53 | 70 |
| 04 | 19 | 31 | 57 | 68 |
+-----+
```

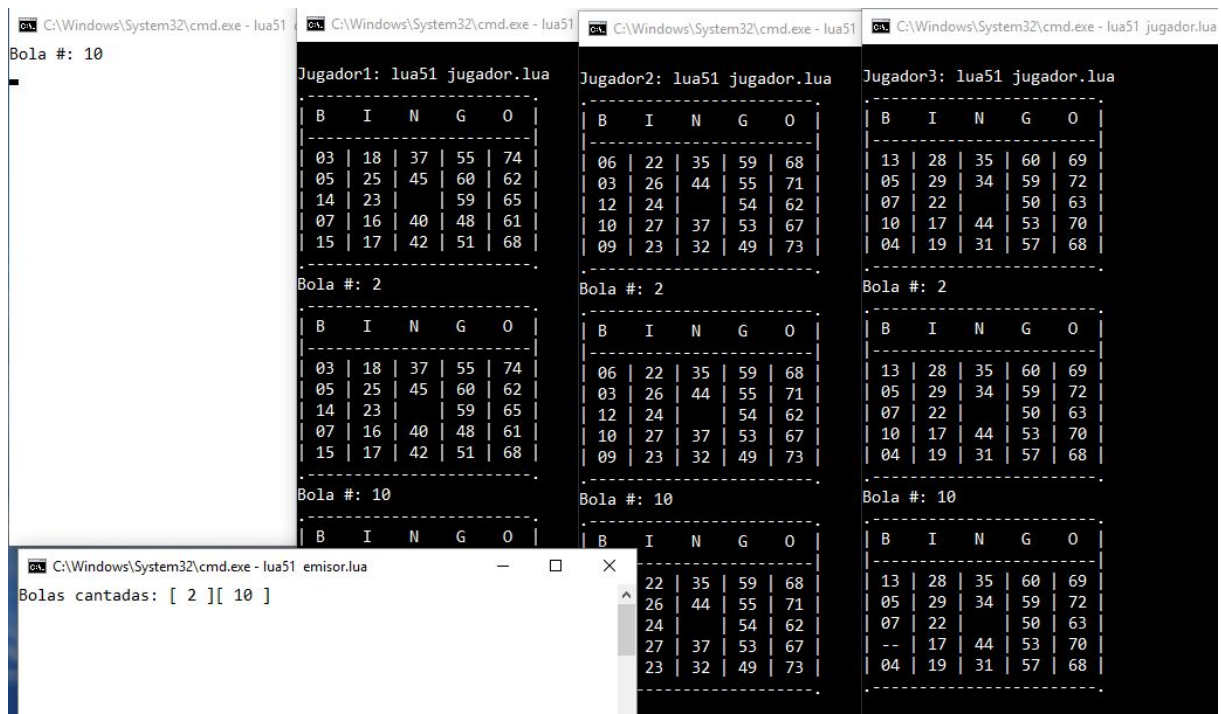
Al conectarse los jugadores el proceso Difusor nos muestra los siguientes mensajes.



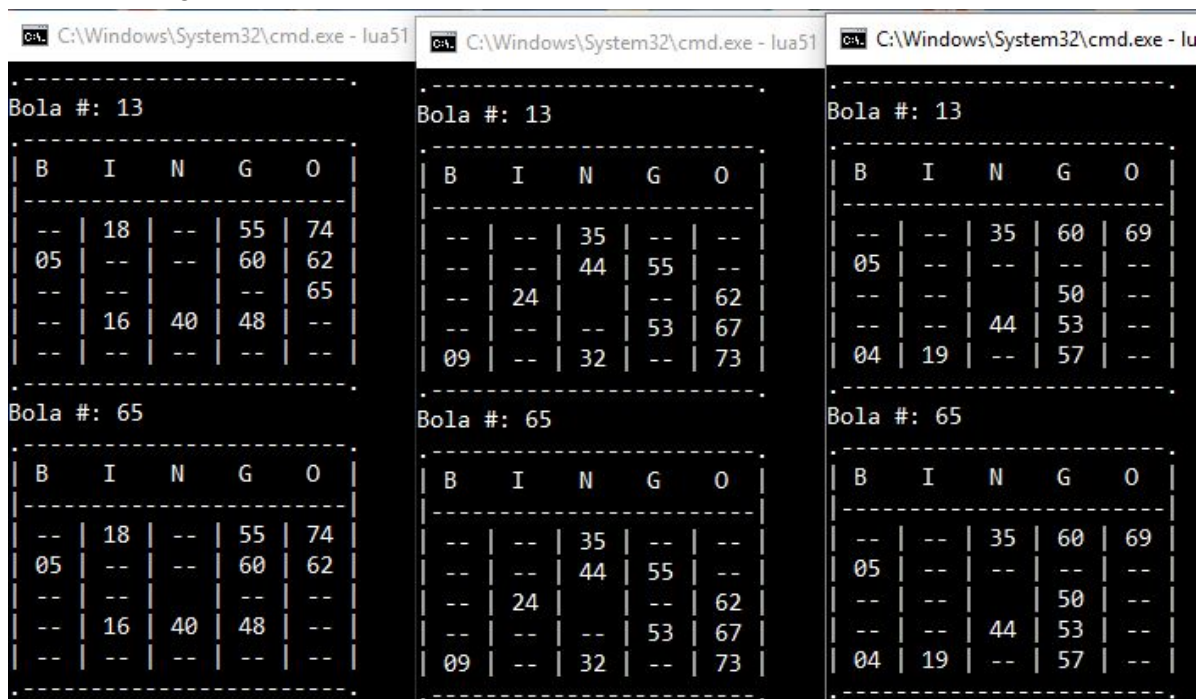
```
C:\Windows\System32\cmd.exe - lua51 difusor.lua

Difusor: lua51 difusor.lua
Jugador conectado
Jugador conectado
Jugador conectado
```

Por último se ejecuta el Emisor.lua e inicia así el juego de Bingo



Se observa cómo por cada bola cantada el jugador muestra la versión actualizada de su cartón de Bingo.



Al ganar un Jugador, este informa al proceso Difusor para parar el juego



Se puede encontrar un video del funcionamiento de este pequeño sistema. Para ver el video, este se encuentra tanto en la carpeta de archivos como en el siguiente enlace: <https://drive.google.com/file/d/0Bwe9DCGRp0m0MIAYaEJtVG1MM3c/view?usp=sharing>

## Conclusiones

El desarrollo de este juego de bingo ha sido de gran provecho, debido a que nos permitió aumentar nuestras habilidades en la programación con el lenguaje Lua.

Nos permitió practicar el uso de estructuras de control de flujo como los: IF, ELSE, FOR, WHILE. También el manejo de estructuras de datos como son las matrices y listas para el control de números.

Nos permitió aprender y mejorar nuestros conocimientos sobre los sistemas distribuidos utilizando diferentes procesos y comunicándose por medio sockets con técnicas como la publicador- suscriptor y la de solicitud-respuesta para el paso de mensajes entre los diferentes procesos.