**UNIVERSIDAD DE TARAPACÁ**



**FACULTAD DE INGENIERÍA**

Departamento de Ingeniería en Computación e Informática



**Taller 1:**

**Socket Programming**

**Autor**: Leonel Alarcón Bravo José Vásquez Gutiérrez

 Gonzalo Vega Mujica

**Curso**: Sistemas Distribuidos

**Profesor**: Diego Aracena Pizarro

ARICA, 08 de Septiembre 2019

**Historial de Cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor(es)** |
| 28/08/2019 | 1.0 | Versión preliminar del formato | Leonel AlarcónJosé VásquezGonzalo Vega |
| 01/09/2019 | 1.1 | Adición de contenidos al plan | Leonel AlarcónJosé VásquezGonzalo Vega |
| 04/09/2019 | 1.2 | Adición de contenidos al plan | Leonel AlarcónJosé VásquezGonzalo Vega |
| 08/09/2019 | 1.3 | Revisión, modificación y finalización del plan | Leonel AlarcónJosé VásquezGonzalo Vega |

**Contenidos**

SECCIÓN PÁGINA

[**I. INTRODUCCIÓN**](#_1fob9te) **5**

[**II. RESUMEN DEL PROYECTO**](#_3znysh7) **6**

[**2.1 PROPÓSITO**](#_skbejrbdh74d) **6**

[**2.2 OBJETIVO GENERAL**](#_tyjcwt) **6**

[**2.3 OBJETIVOS ESPECÍFICOS**](#_3dy6vkm) **6**

[**2.4 SUPOSICIONES**](#_elld7lu7u28h) **6**

[**2.5 RESTRICCIONES**](#_mqgcjx48p429) **6**

[**2.6 ENTREGABLES**](#_2eqz92h6fk5w) **7**

[**III. PROPÓSITO**](#_9nif68ypmpx3) **8**

[**IV. ASPECTOS ÉTICOS**](#_asyofw2lgzxx) **9**

[**V. DESARROLLO**](#_qi467ct21462) **10**

[**¿QUÉ ES SOCKETS?**](#_4i6j89v08l5p) **10**

[**5.1 ANÁLISIS Y DISEÑO**](#_aq4kbcuey3hj) **12**

[**5.1.1 REQUERIMIENTOS FUNCIONALES**](#_mjewps5jprve) **12**

[**5.1.2 REQUERIMIENTOS NO FUNCIONALES**](#_wgslviellhpi) **12**

[**5.1.3 ARQUITECTURA CLIENTE-SERVIDOR**](#_q62nu62ckkj2) **13**

[**5.1.4 INTERFAZ DE USUARIO**](#_ugy53pz4llo0) **13**

[**5.2 IMPLEMENTACIÓN**](#_jcgrhpxfu9qt) **14**

[**5.2.1 AMBIENTE Y RECURSOS**](#_e1aakvds7iux) **14**

[**5.2.2 CÓDIGO**](#_ydyl18i7pxzs) **15**

[**VI. PRUEBAS Y RESULTADOS**](#_97hncflg06t) **20**

[**VII. CONCLUSIÓN**](#_l5gj7l433sx8) **27**

[**VIII. REFERENCIAS**](#_2jxsxqh) **28**

**Figuras**

[Figura 1. Funcionamiento Socket 11](#_Toc18866002)

[Figura 2. Arquitectura cliente y servidor 13](#_Toc18866003)

[Figura 3. Interfaz del Menu.java. 13](#_Toc18866004)

[Figura 4. Interfaz del Juego.java. 14](#_Toc18866005)

[Figura 5. Archivos utilizados. 15](#_Toc18866006)

[Figura 6. Código de los botones de la clase Menu.java 15](#_Toc18866007)

[Figura 7. Código del constructor de la clase Juego.java 16](#_Toc18866008)

[Figura 8. Código del método contectarAServidor de la clase Juego.java. 16](#_Toc18866009)

[Figura 9. Código del método esperarConexion de la clase Juego.java 17](#_Toc18866010)

[Figura 10. Código del método obtenerFlujos de la clase Juego.java. 17](#_Toc18866011)

[Figura 11. Código del método procesarConexion de la clase Juego.java. 18](#_Toc18866012)

[Figura 12. Código del método cerrarConexion de la clase Juego.java 18](#_Toc18866013)

[Figura 13. Código del método enviarDatos de la clase Juego.java. 19](#_Toc18866014)

[Figura 14. Pantalla inicial del programa 20](#_Toc18866015)

[Figura 15. Pantalla del juego 21](#_Toc18866016)

[Figura 16. Ingreso de dirección IP 21](#_Toc18866017)

[Figura 17. Conexión establecida 22](#_Toc18866018)

[Figura 18. Servidor gana partida 23](#_Toc18866019)

[Figura 19. Resultados 23](#_Toc18866020)

[Figura 20. Servidor pierde 24](#_Toc18866021)

[Figura 21. Empate 24](#_Toc18866022)

[Figura 22. Resultado final 25](#_Toc18866023)

[Figura 23. Conexión finalizada 25](#_Toc18866024)

**Tablas**

[Tabla 1. Entregables del proyecto 7](#_Toc18866193)

[Tabla 2. Requerimientos funcionales. 12](#_Toc18866194)

[Tabla 3. Requerimientos no funcionales. 12](#_Toc18866195)

# **I. INTRODUCCIÓN**

En resumen, un socket es un método para la comunicación entre un programa del cliente y un programa del servidor en una red, se define, por tanto, como el punto final en una conexión. Teniendo esto en cuenta se nos solicita realizar hacer una propuesta sobre un programa que utilice socket y su respectiva implementación. Teníamos la libertad de poder hacer la elección del tema a gusto, nosotros elegimos realizar e implementar el juego “Piedra, papel o tijera” ya que a primera vista e idea se nos ocurrió que podría cumplir con las primitivas de Socket además de que sería agradable trabajar con un tema que se nos haga agradable.

Cabe destacar que en este documento se realizará primeramente un análisis teórico para así definir con más exactitud los propósitos y objetivos del proyecto, a continuación de esto se describirá todo el desarrollo e implementación acompañada de las pruebas correspondientes que se harán y sus resultados.

# **II. RESUMEN DEL PROYECTO**

## **2.1 PROPÓSITO**

El propósito del documento es presentar el trabajo realizado y los resultados obtenidos luego de realizar la implementación de un programa socket.

## **2.2 OBJETIVO GENERAL**

El objetivo general del trabajo es poder realizar un estudio inicialmente teórico sobre socket viéndolo como un método de comunicación para a continuación de esto realizar la implementación creando un programa utilizándolo y así realizar pruebas de las actividades solicitadas y si se concreta con éxito su comunicación.

## **2.3 OBJETIVOS ESPECÍFICOS**

* Investigar sobre Sockets Programming
* Revisar y comprender códigos Sockets entregados por el profesor
* Implementar el juego “Cachipún” con Sockets
* Realizar conclusiones sobre lo aprendido durante la realización del taller

## **2.4 SUPOSICIONES**

* Durante el ciclo de vida del proyecto, no se verá interrumpido el calendario académico.
* El equipo tiene los conocimientos necesarios para llevar a cabo el proyecto.
* El cliente no cambiará los requisitos del proyecto.
* Se nos entregará las herramientas necesarias para la elaboración del proyecto.

## **2.5 RESTRICCIONES**

* El presupuesto será de coste $0.
* El desarrollo del proyecto será de 4 semanas.
* Se utilizará software libre.

##

## **2.6 ENTREGABLES**

|  |  |  |
| --- | --- | --- |
| **ENTREGABLE** | **DESCRIPCIÓN** | **FORMATO** |
| Informe Sockets | * Definir el Proyecto
* Definir los Objetivos
* Implementación del software
 | Informe Sockets.doc |
| Software | * Archivo ejecutable
 | Cachipun.jar |
| Carpeta de Proyecto  | * Carpeta del proyecto completo para poder ejecutar en cualquier IDE compatible
 | Carpeta Cachipun. |

Tabla 1. Entregables del proyecto

#

# **III. PROPÓSITO**

El problema propuesto en este proyecto es de realizar un sistema Cliente-Servidor donde esté utilice la herramienta de Sockets, de tal forma que este establezca una comunicación entre sí de manera adecuada.

Por lo tanto, la solución propuesta para dicho problema fue de realizar un sistema el cual se basa en el conocido juego “Piedra, Papel o Tijera”, donde el servidor al iniciar sus servicios debe esperar a que un cliente establezca la comunicación entre ellos mismos, así de esa forma el servidor también estaría actuando como cliente. Ya establecida la conexión, se dará inicio al juego.

Finalmente, esta esta solución lleva implementado lo que tiene relación con sockets para tener comunicación entre los participantes.

#

# **IV. ASPECTOS ÉTICOS**

Respeto: Lo principal que se necesita en cualquier tipo de relación o actividad a ejecutar es el respeto hacia nuestros pares, es por esto que el equipo se compromete a respetar tanto al profesor y compañeros de asignatura como a mismos integrantes del equipo, así los resultados obtenidos serán de mejor nivel al poder trabajar en un ambiente grato.

Compromiso de calidad: El equipo se compromete en realizar e implementar el proyecto cumpliendo con todos los requerimientos solicitados y estándares establecidos para así poder entregar y finalizar de la mejor manera posible y cumpliendo con la calidad que corresponda.

Legibilidad de nuestra propuesta: El equipo de trabajo realizará una propuesta de proyecto a implementar totalmente única, pensada y creada por nosotros, esto quiere decir que no se realizará una copia de algún proyecto anterior o de otro equipo.

Responsabilidad y compromiso: El equipo de trabajo cumplirá con los compromisos realizados tanto con la asignatura en curso como con el docente a cargo, esto quiere decir que se entregará todo lo solicitado en relación al proyecto de acuerdo a las fechas de entregas estipuladas con anterioridad para que de esta forma se cumpla el compromiso y responsabilidad con el mismo.

#

# **V. DESARROLLO**

## ¿QUÉ ES SOCKETS?

Un socket, es un método para la comunicación entre un programa del cliente y un programa del servidor en una red, se define, por tanto, como el punto final en una conexión.

Este mecanismo surge a principios de los 80 con el sistema Unix de Berkeley, para proporcionar un medio de comunicación entre procesos y presentan la misma funcionalidad que tiene la comunicación por correo o por teléfono (de un buzón se extraen mensajes completos, mientras que el teléfono permite el envío de flujos de información que no tienen una estructura claramente definida), es decir permiten que un proceso hable (emita o reciba información) con otro incluso estando estos en distintas máquinas. Esta característica de ínterconectividad hace que el concepto de socket sea de gran utilidad.

Principio de funcionamiento

Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto. Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

* Que un programa sea capaz de localizar al otro.
* Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

Para ello son necesarios los tres recursos que originan el concepto de socket:

* Un protocolo de comunicaciones, que permite el intercambio de octetos.
* Un par de direcciones del Protocolo de Red (Dirección IP, si se utiliza el Protocolo TCP/IP), que identifica la computadora de origen y la remota.
* Un par de números de puerto, que identifica a un programa dentro de cada computadora.

Los sockets permiten implementar una arquitectura cliente-servidor o peer to peer. La comunicación debe ser iniciada por uno de los programas que se denomina programa cliente. El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa servidor.

Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Cuando un cliente conecta con el servidor se crea un nuevo socket, de esta forma, el servidor puede seguir esperando conexiones en el socket principal y comunicarse con el cliente conectado, de igual manera se establece un socket en el cliente en un puerto local.

Una [aplicación servidor](https://www.ecured.cu/Servidor_de_Aplicaciones) normalmente escucha por un puerto específico esperando una petición de conexión de un cliente, una vez que se recibe, el cliente y el servidor se conectan de forma que les sea posible comunicarse entre ambos. Durante este proceso, el cliente es asignado a un número de puerto, mediante el cual envía peticiones al servidor y recibe de éste las respuestas correspondientes.

Similarmente, el servidor obtiene un nuevo número de puerto local que le servirá para poder continuar escuchando cada petición de conexión del puerto original. De igual forma une un socket a este puerto local.

##

Figura 1. Funcionamiento Socket

##

## **5.1 ANÁLISIS Y DISEÑO**

### **5.1.1 REQUERIMIENTOS FUNCIONALES**

|  |  |
| --- | --- |
| **CÓDIGO REQUERIMIENTO** | **DESCRIPCIÓN** |
| RF01 | Permite al sistema, como cliente, realizar conexión al servidor. |
| RF02 | Permite al sistema ingresar la dirección IP del servidor a conectar. |
| RF03 | Permite al sistema funcionar como servidor. |
| RF04 | Permite al sistema actualizar los datos de las rondas. |
| RF05 | Permite al sistema finalizar la conexión entre cliente y servidor. |
| RF06 | Permite al sistema seleccionar entre piedra, papel o tijera. |
| RF07 | Permite al sistema determinar al ganador o empate de la ronda. |

Tabla 2. Requerimientos funcionales.

### **5.1.2 REQUERIMIENTOS NO FUNCIONALES**

|  |  |
| --- | --- |
| **CÓDIGO REQUERIMIENTO** | **DESCRIPCIÓN** |
| RNF01 | Se tendrá un solo software que actuará como cliente o como servidor. |
| RNF02 | El sistema debe contar con una interfaz de usuario con imágenes. |
| RNF03 | El sistema debe ser capaz de esperar y establecer la conexión. |
| RNF04 | El sistema mostrará el estado del juego. |

Tabla 3. Requerimientos no funcionales.

###

### **5.1.3 ARQUITECTURA CLIENTE-SERVIDOR**

La arquitectura cliente-servidor está basada en que el servidor aceptará un solo cliente para la conexión, para esto será necesario un switch, router o router wifi.



Figura 2. Arquitectura cliente y servidor

### **5.1.4 INTERFAZ DE USUARIO**

Para el Menu.java se creó la siguiente interfaz, permitiendo elegir entre cliente y servidor, es caso de ser cliente es necesario ingresar la dirección IP del servidor que desea conectar.



Figura 3. Interfaz del Menu.java.

Para el Juego.java se creo la siguiente interfaz, la misma para cliente y servidor, permitiendo a los usuarios elegir las opciones que son piedra, papel o tijera, también visualizar la opción elegida por los usuarios, el contador de rondas, rondas ganadas, rondas empatadas y rondas ganadas por el oponente.



Figura 4. Interfaz del Juego.java.

## **5.2 IMPLEMENTACIÓN**

### **5.2.1 AMBIENTE Y RECURSOS**

Para la creación del juego, se consideró realizarlo en un solo software, es decir, colocar el cliente y servidor en un solo software, permitiendo elegir al usuario ser cliente o servidor.

La implementación se realizó en lenguaje Java y se utilizó NetBeans IDE 8.2 para facilitar el trabajo de la interfaz de usuario.

Para el proyecto de NetBeans se crearon las clases Menu.java y Juego.java, luego se utilizaron 3 imágenes para representar la piedra, papel y tijera.



Figura 5. Archivos utilizados.

### **5.2.2 CÓDIGO**

Para el Menu.java al ser un menú, solo permite elegir entre cliente y servidor, para eso al pulsar el botón “Servidor” creará una instancia de la clase Juego pasando como parámetro “-1” que significa iniciar como servidor, en caso de pulsar el botón “Cliente” con una IP válida ingresada, creará una instancia de la clase Juego pasando como parámetro la IP.



Figura 6. Código de los botones de la clase Menu.java

Para el Juego.java al ser el juego, tiene varios métodos, que los más importantes serán explicados a continuación.

El constructor de la clase recibe el parámetro entregado por la instancia de Menu.java permitiendo elegir el modo a operar, ya sea cliente o servidor, se inicializa los valores del juego, en el bloque try catch se instancia de la clase DatagramSocket para obtener la dirección ip del adaptador de red principal (Significativo para los dispositivos que tienen más de un adaptador de red) y por último se crea un hilo para la conexión.

##

Figura 7. Código del constructor de la clase Juego.java

Al ejecutar como cliente se utiliza el método conectarAServidor(String ip) que realiza la conexión al servidor con la ip entregada con el puerto “12345”, luego se muestra en pantalla a quien se hizo la conexión y se desbloquea los botones del juego.



Figura 8. Código del método contectarAServidor de la clase Juego.java.

Al ejecutar como servidor previamente debe ejecutarse “servidor = new ServerSocket(12345, 1);”, que permite crear un servidor socket con el puerto “12345” para un cliente, también se utiliza el método esperarConexion() que espera que un cliente se conecte, luego se muestra en pantalla con quien se hizo la conexión y se desbloquea los botones del juego.



Figura 9. Código del método esperarConexion de la clase Juego.java

Los siguientes métodos son utilizados tanto para cliente como para servidor.

 El método obtenerFlujos() establece la entrada y salida, permitiendo posteriormente enviar y recibir mensajes.



Figura 10. Código del método obtenerFlujos de la clase Juego.java.

El método procesarConexion() es donde se entra en un bucle para recibir los mensajes enviados por la otra máquina, y cada mensaje se discrimina por ser una opción de juego o una dirección ip, ya sea de otra máquina o la misma (127.0.0.1) para que se muestre en pantalla.



Figura 11. Código del método procesarConexion de la clase Juego.java.

El método cerrarConexion() finaliza la conexión entre cliente y servidor, también bloquea el juego para que no hayan problemas.



Figura 12. Código del método cerrarConexion de la clase Juego.java

El método enviarDatos(String mensaje) permite enviar un mensaje a la otra máquina.



Figura 13. Código del método enviarDatos de la clase Juego.java.

Esos son los métodos principales de la conexión, luego tenemos más métodos que van enfocado al juego, como por ejemplo actualizarDatos(), reiniciar(), resultadoRonda(), mostrarElegidos().

#

# **VI. PRUEBAS Y RESULTADOS**

 Piedra, papel o tijera más conocido en Chile como “Cachipún” es un juego de manos donde existen principales y únicos componentes, los cuales son piedra, papel y tijeras, tal como su nombre lo indica.

 La idea de este juego es que cada jugador debe escoger uno de estos elementos y ambos mostrar las manos al mismo tiempo formando la figura de su elemento seleccionado y ganarle al contrincante dependiendo de su elección. Las prioridades son las siguientes:

* Piedra le gana a Tijeras
* Tijeras le gana Papel
* Papel le gana a Piedra



Figura 14. Pantalla inicial del programa

 En la figura anterior, se ve la pantalla principal del programa donde el cual uno puede elegir ser servidor o ingresar como cliente.

 En caso de ser servidor, se abrirá una pantalla donde nos muestra un mensaje de espera y nuestra dirección IP la cual debe ingresar el cliente en el cuadro de texto de la pantalla inicial.



Figura 15. Pantalla del juego

Como se aprecia en la imagen, las figuras de piedra, papel o tijera están bloqueadas, esto se debe a que por el momento ningún jugador ha ingresado a la partida. Por lo tanto en este instante, por parte del cliente, este debe ingresar en el cuadro de texto mencionado anteriormente la dirección IP que muestra el servidor en su pantalla.



Figura 16. Ingreso de dirección IP

Ya con la dirección IP ingresada, se accede a la pantalla del juego y ya la imágenes están desbloqueadas para poder jugar, además de que se muestra un mensaje que indica la dirección con la cual se establece conexión.



Figura 17. Conexión establecida

 Cabe mencionar, que en este sistema el servidor también actúa como cliente, es decir, también pasa a ser un jugador dentro de la partida.

 Continuando, veremos los diferentes casos de los resultados posibles que se pueden obtener a partir de la opción elegida por parte de ambos jugadores. Entonces se mostrarán ambas pantallas para corroborar los resultados.

Caso 1: Servidor gana

**

Figura 18. Servidor gana partida

Como se aprecia en la figura anterior, tanto en el cliente (derecha) como en el servidor (izquierda) da su respectivo resultado, es decir, que por parte del cliente le informa que ha perdido esa ronda y por el lado contrario, en el servidor le informa que ha ganado la ronda. También cabe mencionar que esta pantalla donde se ven los resultados de la ronda tiene un lapso de tiempo de 3 segundos antes de pasar a la siguiente ronda.

Luego, de pasar los 3 segundos los contadores de Ganadas o Empate aumenta según la vista en la que se está jugando. Obteniendo así lo siguiente.



Figura 19. Resultados

Caso 2: Servidor pierde

 

Figura 20. Servidor pierde

Caso 3: Empate



Figura 21. Empate

 En las imágenes se visualizaban como actúa el programa al ver los distintos casos, y como se había explicado en el caso 1, dependiendo en qué vista se esté jugando aumenta el contador tanto del jugador como del oponente. Entonces ya realizando estas 3 rondas, el resultado final fue el siguiente.

 

Figura 22. Resultado final

Para finalizar la conexión entre los participantes simplemente se debe cerrar la aplicación. Al realizar dicha acción, el último participante que cierra el juego le aparecerá el mensaje “Conexión finalizada” en la parte inferior de la pantalla, de esa forma las imágenes se bloquean ya que no hay jugadores con quien jugar. Así, debe cerrar la aplicación ya que no podrá realizar otra acción.



Figura 23. Conexión finalizada

Por último, en este sistema está permitido jugar hasta con 2 participantes, si un tercer participante quiere ingresar a la partida no podrá hacerlo, solamente se quedará esperando en la pantalla del juego esperando alguna conexión, esto se debe a que el servidor ya ha establecido una conexión con un previo jugador, por lo tanto, no permite que ingrese otro. A pesar de ello, la partida no sufre ningún error, es decir, el servidor y los servicios que entrega no se caen.

# **VII. CONCLUSIÓN**

Podemos concluir como equipo que los propósitos y objetivos planteados con anterioridad sobre el taller se han cumplido con éxito, es decir que pudimos identificar todos los aspectos importantes de Socket analizando la parte teórica y los proyectos y/o códigos que entregó el profesor siendo así una guía para nosotros de lo que debemos llegar a implementar. la implementación se llevó a cabo de la mejor manera posible guiándonos por los estándares de calidad que necesita un software. cabe destacar que primeramente los resultados no fueron los óptimos ya que el programa quedaba con algunos detalles y defectos, pero gracias a la realización de varias pruebas y re análisis del código de pudo llegar a la versión final que fue la entregada.

La importancia del uso de socket es muy alta ya que desde su creación y primera implementación se hizo conocida mundialmente por el éxito rotundo que tuvo este, en la actualidad podemos encontrarla en muchos software tanto en software de administración de archivos como también en páginas webs que os entregan algún tipo de servicio, es por esto que el desarrollo de este taller fue de gran importancia para nosotros los estudiantes ya que nos prepara obteniendo conocimiento no solo teórico sino también práctico sobre el tema.

Como trabajo a futuro que nosotros podemos visualizar sería hacer una nueva versión del programa la cual permita que 2 clientes puedan realizar el establecimiento de comunicación con el servidor y así este solo sea un tipo de “Administrador” dentro juego dejando así que los clientes interactúen entre sí mediante él.

# **VIII. REFERENCIAS**

1. Libro “Sistemas Distribuidos Conceptos y Diseño”

○ James F. Kurose - Keith W. Ross / Quinta edición - Capítulo 5

2. Libro “Como programar en Java”

○ Harvey M. Deitel - Paul J. Deitel / Quinta edición - Capítulo 18

3. Libro “Principios y paradigmas de sistemas distribuidos”

○ Fred Halsall / Quinta edición - Capítulo 3

4. Conceptos básicos de socket

○ https://www.ecured.cu/Socket

5. Normas APA / Marco teórico

○ http://normasapa.net/marco-teorico/