**--------------APLICACIONES EN COMUNICACIONES EN RED------------**

**PRÁCTICA 2:**

Sopa de letras usando sockets de datagrama

**Alumno:**

Meza Vargas Brandon David

**Grupo:**

3CM16

**Profesor:**

Moreno Cervantes Axel Ernesto

**índice**

[**Introducción** 5](#_Toc98084629)

[**Desarrollo** 6](#_Toc98084630)

[**Sopa de letras** 6](#_Toc98084631)

[**Método setSoup** 6](#_Toc98084632)

[**Método vertical** 8](#_Toc98084633)

[**Método horizontal** 9](#_Toc98084634)

[**Método invertedVertical** 9](#_Toc98084635)

[**Método invertedHorizontal** 10](#_Toc98084636)

[**Método rightDiagonal** 10](#_Toc98084637)

[**Método leftDiagonal** 11](#_Toc98084638)

[**Método rightDownDiagonal** 11](#_Toc98084639)

[**Método leftDownDiagonal** 11](#_Toc98084640)

[**Método isWordCollapsed** 12](#_Toc98084641)

[**Método fillSoup** 14](#_Toc98084642)

[**Método showSoup** 14](#_Toc98084643)

[**Cliente** 15](#_Toc98084644)

[**Main** 15](#_Toc98084645)

[**Método connectGame** 15](#_Toc98084646)

[**Método initGame** 16](#_Toc98084647)

[**Método setDifficulty** 17](#_Toc98084648)

[**Método sendDif** 18](#_Toc98084649)

[**Método showWords** 19](#_Toc98084650)

[**Método recvSoup** 19](#_Toc98084651)

[**Método searchWords** 20](#_Toc98084652)

[**Método sendSolveTime** 24](#_Toc98084653)

[**Método recvElapsedTimes** 25](#_Toc98084654)

[**Servidor** 25](#_Toc98084655)

[**Main** 25](#_Toc98084656)

[**Método connectGame** 26](#_Toc98084657)

[**Método setDifficulty** 26](#_Toc98084658)

[**Método setConcept** 27](#_Toc98084659)

[**Método getConceptWords** 28](#_Toc98084660)

[**Método createSoup** 29](#_Toc98084661)

[**Método recvElapsedTime** 30](#_Toc98084662)

[**Método sendUserTimes** 31](#_Toc98084663)

[**Conclusiones** 37](#_Toc98084664)

[**Bibliografía** 38](#_Toc98084665)

**índice de Ilustraciones**

[Ilustración 1. Sockets de datagrama. 5](#_Toc98084666)

[Ilustración 2. Método setSoup. 8](#_Toc98084667)

[Ilustración 3. Método vertical 9](#_Toc98084668)

[Ilustración 4. Método horizontal 9](#_Toc98084669)

[Ilustración 5. Método invertedVertical 9](#_Toc98084670)

[Ilustración 6. Método invertedHorizontal 10](#_Toc98084671)

[Ilustración 7. Método rightDiagonal 10](#_Toc98084672)

[Ilustración 8. Método leftDiagonal 11](#_Toc98084673)

[Ilustración 9. Método rightDownDiagonal 11](#_Toc98084674)

[Ilustración 10. Método leftDownDiagonal 11](#_Toc98084675)

[Ilustración 11. Método isWordCollapsed 13](#_Toc98084676)

[Ilustración 12. Método fillSoup 14](#_Toc98084677)

[Ilustración 13. Método showSoup 14](#_Toc98084678)

[Ilustración 14. Cliente main 15](#_Toc98084679)

[Ilustración 15. Método connectGame 16](#_Toc98084680)

[Ilustración 16. Método initGame 17](#_Toc98084681)

[Ilustración 17. Método setDifficulty 18](#_Toc98084682)

[Ilustración 18. Método sendDif 18](#_Toc98084683)

[Ilustración 19. Método showWords 19](#_Toc98084684)

[Ilustración 20. Método recvSoup 19](#_Toc98084685)

[Ilustración 21. Método searchWords 24](#_Toc98084686)

[Ilustración 22. Método sendSolveTime 24](#_Toc98084687)

[Ilustración 23. Método recvElapsedTimes 25](#_Toc98084688)

[Ilustración 24. Main del servidor 25](#_Toc98084689)

[Ilustración 25. Método connectGame 26](#_Toc98084690)

[Ilustración 26. Método setDifficulty 27](#_Toc98084691)

[Ilustración 27. Método setConcept 28](#_Toc98084692)

[Ilustración 28. Método getConceptWords 29](#_Toc98084693)

[Ilustración 29. Método createSoup 30](#_Toc98084694)

[Ilustración 30. Método recvElapsedTime 31](#_Toc98084695)

[Ilustración 31. Método sendUserTimes 32](#_Toc98084696)

[Ilustración 32. Corriendo el servidor y el cliente. 32](#_Toc98084697)

[Ilustración 33. Usuario introduciendo su nombre 33](#_Toc98084698)

[Ilustración 34. Servidor indicando quien se ha conectado. 33](#_Toc98084699)

[Ilustración 35. Sopa de letras mostrada 34](file:///C:\Users\PC\Desktop\6to%20Semestre%20ESCOM\Aplicaciones%20para%20comunicaciones%20en%20red\Prácticas\Práctica%202%20-%20Sopa%20de%20letras\Reporte\Meza_Vargas_Practica2%20-%20sopa%20de%20letras.docx#_Toc98084700)

[Ilustración 36. Información del servidor. 34](#_Toc98084701)

[Ilustración 37. Encontrando una palabra. 35](#_Toc98084702)

[Ilustración 38. Mensaje de error 35](#_Toc98084703)

[Ilustración 39. Mensaje de error con coordenadas no válidas. 35](#_Toc98084704)

[Ilustración 40. Finalizando el juego 36](#_Toc98084705)

[Ilustración 41. Tiempo capturado en servidor. 36](#_Toc98084706)

[Ilustración 42. Cerrando conexión. 36](#_Toc98084707)

# **Introducción**

Los sockets de datagrama son un servicio de transporte sin conexión. Son más eficientes que TCP, pero no está garantizada la fiabilidad. Los datos se envían y reciben en paquetes, cuya entrega no está garantizada. Los paquetes pueden ser duplicados, perdidos o llegar en un orden diferente al que se envió.

El protocolo de comunicaciones con datagramas es un protocolo sin conexión, es decir, cada vez que se envíen datagramas es necesario enviar el descriptor del socket local y la dirección del socket que debe recibir el datagrama. Como se puede ver, hay que enviar datos adicionales cada vez que se realice una comunicación.

Para la mayoría de los programas que utilicen la red, el usar un flujo TCP en vez de un datagrama UDP es más sencillo y hay menos posibilidades de tener problemas. Sin embargo, cuando se requiere un rendimiento óptimo, y está justificado el tiempo adicional que supone realizar la verificación de los datos, los datagramas son un mecanismo realmente útil.

En el presente reporte se detallará la práctica realizada haciendo uso de sockets de datagrama para poder enviar datos de un cliente a un servidor y viceversa con el objetivo de hacer un juego de sopa de letras.

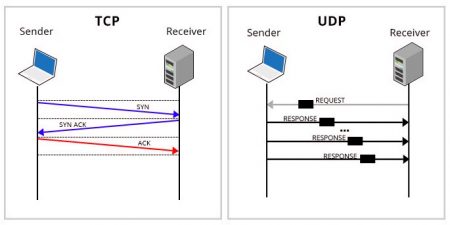


Ilustración 1. Sockets de datagrama.

# **Desarrollo**

A continuación se muestra una explicación de cada parte que conforma la sopa de letras creada en esta práctica, así como las capturas correspondientes a cada método creado.

## **Sopa de letras**

### **Método setSoup**

Este método se encarga de setear la sopa de letras, establece el límite de filas y columnas, además de que va estableciendo la orientación de las palabras de manera aleatoria.

De igual forma se establece el espacio necesario para que las palabras quepan de manera diagonal. Dependiendo del tamaño de la palabra y el espacio disponible se acomodará la palabra en una determinada posición, verificando antes que la palabra no choque con alguna otra palabra ya acomodada en la sopa. En caso de que no choquen, se acomoda la palabra en una de 8 diferentes posiciones: vertical, vertical invertida, horizontal, horizontal invertido, diagonal derecha hacia arriba, diagonal derecha hacia abajo, diagonal izquierda hacia arriba y diagonal izquierda hacia abajo.

En caso de que una palabra choque se le asignará otra orientación para poder acomodarla. En la siguiente ilustración podemos ver el código de este método.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 2. Método setSoup.

### **Método vertical**

Este método acomoda la palabra de manera vertical recorriendo las filas desde la posición que se le asigno de manera automática a la palabra.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 3. Método vertical

### **Método horizontal**

Este método acomoda la palabra de manera horizontal recorriendo las columnas desde la posición que se le asigno de manera aleatoria a la palabra.

Texto

Descripción generada automáticamente

Ilustración 4. Método horizontal

### **Método invertedVertical**

Este método acomoda la palabra en la posición vertical invertida, recorriendo las filas y disminuyendo la posición de fila de la palabra.

Texto

Descripción generada automáticamente

Ilustración 5. Método invertedVertical

### **Método invertedHorizontal**

Este método acomoda la palabra de manera horizontal invertido recorriendo las columnas y disminuyendo la columna desde la posición que se le asigno a la palabra de manera aleatoria.

Texto

Descripción generada automáticamente

Ilustración 6. Método invertedHorizontal

### **Método rightDiagonal**

Este método acomoda la palabra en diagonal derecha hacia arriba, recorriendo la posición de la fila y columna

Texto

Descripción generada automáticamente

Ilustración 7. Método rightDiagonal

Los siguientes métodos son similares al anterior, solo se aumenta o disminuye la posición de la columna o fila dependiendo si es hacia arriba o abajo o izquierda o derecha, por esto solo se incluirán las capturas.

### **Método leftDiagonal**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 8. Método leftDiagonal

### **Método rightDownDiagonal**

Texto

Descripción generada automáticamente

Ilustración 9. Método rightDownDiagonal

### **Método leftDownDiagonal**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 10. Método leftDownDiagonal

### **Método isWordCollapsed**

Este método indica si una palabra esta colapasando con otra dentro de la sopa. Dependiendo de la orientación de la palabra va checando si en la posición que se esta colocando ya hay escrito algún otro carácter que haga que colapse y por lo tanto no se pueda colocar la palabra completa.

Si no choca con nada regresa false pues si se podrá colocar esta palabra.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 11. Método isWordCollapsed

### **Método fillSoup**

Este método se encarga de llenar los espacios vacíos de la sopa con palabras del alfabeto de manera aleatoria.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 12. Método fillSoup

### **Método showSoup**

Este método se encarga de imprimir en pantalla la sopa de letras.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 13. Método showSoup

## **Cliente**

### **Main**

Dentro del main se repite el juego de la sopa de letras mientras el usuario quiera seguir jugando, además de que se manda al servidor la opción del usuario de seguir jugando o no para que el servidor siga a la escucha de este.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 14. Cliente main

### **Método connectGame**

En este método se le pide al usuario que ponga su username para comenzar a jugar, este nombre se manda al servidor para que este sepa quien se está conectando.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 15. Método connectGame

### **Método initGame**

Dentro de este método se manda a llamar la función para que el usuario elija la dificultad, además de que le muestra al usuario un menú con los posibles temas que puede elegir para la sopa de letras, lee la opción del usuario y se manda al servidor que se encargará de crear la sopa de letras de acuerdo con el tema que selecciono el usuario.

De igual forma en este método se reciben las palabras que el usuario pude buscar en la sopa de letras provenientes del servidor, estas se guardan en una lista.

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración 16. Método initGame

### **Método setDifficulty**

Este método simplemente le muestra al usuario un menú con las posibles dificultades que puede elegir, dependiendo de la dificultad que seleccione se manda a llamar la función que manda la dificultad al servidor.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 17. Método setDifficulty

### **Método sendDif**

Este método se encarga de mandar al servidor la dificultad que el usuario escogió, el envío se realiza debidamente usando sockets de datagrama.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 18. Método sendDif

### **Método showWords**

Este método simplemente se encarga de imprimir las palabras que el usuario no ha encontrado en la sopa de letras.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 19. Método showWords

### **Método recvSoup**

Este método se encarga de recibir la sopa de letras creada por el servidor a través de sockets de datagrama. Para esto se creo un arreglo bidimensional donde se almacenará la sopa del servidor. Por cada iteración se manda un carácter almacenado en la sopa en la posición actual del ciclo, esto es eficiente debido a la velocidad de UDP.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 20. Método recvSoup

### **Método searchWords**

Este es el método más complejo pues aquí se encarga de buscar la palabra que el usuario quiere encontrar en la sopa.

Primero se le pide al usuario que elija la palabra a buscar y sus coordenadas de inicio y de fin. Posteriormente, dependiendo de las coordenadas se determina la orientación de la palabra y ya con la orientación determinada podemos ir avanzando posición por posición en la sopa de letras para ver si coincide con el carácter de la palabra en esa posición actual del ciclo. Si es encontrada, se le avisa al usuario y su puntaje aumenta en uno, mostrándole las palabras restantes por buscar de acuerdo con la dificultad que eligió al inicio del juego.

En caso de encontrar la palabra se le vuelve a mostrar las palabras y la sopa de letras.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

Ilustración 21. Método searchWords

### **Método sendSolveTime**

Este método manda al servidor el tiempo total que le tomo al usuario resolver la sopa de letras en minutos.

Texto

Descripción generada automáticamente

Ilustración 22. Método sendSolveTime

### **Método recvElapsedTimes**

Este método recibe desde el servidor los tiempos que han obtenido todos los jugares que han jugado a la sopa de letras, se le muestra el tiempo, a la dificultad y con el tema que eligieron para jugar.

Texto

Descripción generada automáticamente

Ilustración 23. Método recvElapsedTimes

## **Servidor**

### **Main**

En el main solo mandamos a llamar la función connectGame

Texto

Descripción generada automáticamente con confianza baja

Ilustración 24. Main del servidor

### **Método connectGame**

En este método se hace la aceptación del cliente recibiendo el nombre de usuario y un ciclo infinito para recibir datagramas, además de un ciclo que dependiendo si el jugador quiere seguir jugando cierra o no el socket.

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración 25. Método connectGame

### **Método setDifficulty**

Este método se encarga de establecer la dificultar del juego de acuerdo a la que escogió el usuario al inicio.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 26. Método setDifficulty

### **Método setConcept**

Aquí establecemos de que archivo se agarrarán las palabras para la sopa de letras de acuerdo con el tema escogió por el usuario. Se recibe la opción por medio de un paquete y con un swtich determinamos el concepto del juego.

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración 27. Método setConcept

### **Método getConceptWords**

En este método obtenemos las palabras del archivo que corresponde al concepto seleccionado por el usuario y así como se van leyendo las palabras se van enviando al cliente por medio de paquetes hasta que ya no haya más palabras.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 28. Método getConceptWords

### **Método createSoup**

Este método se encarga de crear, setar y llenar la sopa con ayuda de los métodos ya vistos de la sopa de letras, de igual forma en este método se hace el envió de la sopa de letras.

Texto

Descripción generada automáticamente

Ilustración 29. Método createSoup

### **Método recvElapsedTime**

Este método recibe el tiempo que tardo el usuario para guardarlo al final de un archivo que contiene todos los tiempos históricos de los jugadores que han jugado a la sopa de letras, además se guarda el nombre del usuario que hizo ese tiempo, en la dificultad que lo hizo y en que concepto jugó.

Texto

Descripción generada automáticamente

Ilustración 30. Método recvElapsedTime

### **Método sendUserTimes**

Este método se encarga de leer los registros de tiempos de todos los jugadores para mandarlos a través de paquetes al cliente para que este pueda ver los tiempos de los demás jugadores y compararlos con el suyo.

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración 31. Método sendUserTimes

**Pruebas de funcionamiento**

Al correr el servidor este indica que ha sido inicializado y el cliente le pregunta al usuario su nombre como se ve en la siguiente ilustración.

Texto

Descripción generada automáticamente con confianza baja

Imagen que contiene Texto

Descripción generada automáticamente

Ilustración 32. Corriendo el servidor y el cliente.

Al introducir el nombre el cliente, el servidor lo recibe y se le pregunta al usuario la dificultad que va a seleccionar como se ve en la siguientes ilustraciónes.

Texto, Carta

Descripción generada automáticamente

Ilustración 33. Usuario introduciendo su nombre

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 34. Servidor indicando quien se ha conectado.

Al escoger el usuario la dificultad se le pregunta al usuario el concepto que quiere para la sopa de letras, al elegirlo se muestra la sopa de letras junto con las palabras que puede buscar, el servidor recibe el concepto como se ve en las siguientes ilustraciones.

Tabla

Descripción generada automáticamente

Ilustración 35. Sopa de letras mostrada

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 36. Información del servidor.

A partir de esto el cliente ya puede ingresar la palabra que quiere buscar y se le pedirán las coordenadas de inicio y fin, si la palabra fue encontrada se quitará de la lista a buscar y se el aumenta la puntuación como se puede ver en la siguiente ilustración.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 37. Encontrando una palabra.

En caso de que el usuario quiera buscar una palabra que no esta en la lista se le mostrará un error.

Texto

Descripción generada automáticamente con confianza media

Ilustración 38. Mensaje de error

De igual forma si las coordenadas que ingresa no corresponden a la palabra que quiere buscar se le mandará un error.

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Ilustración 39. Mensaje de error con coordenadas no válidas.

Cuando el usuario encuentra todas las palabras que debe buscar de acuerdo con la dificultad elegida se le muestra el tiempo total que hizo y el tiempo de los demás jugadores y se le pregunta si quiere jugar de nuevo, de igual forma el servidor recibe el tiempo que hizo este usuario como se ve en las siguientes ilustraciones.

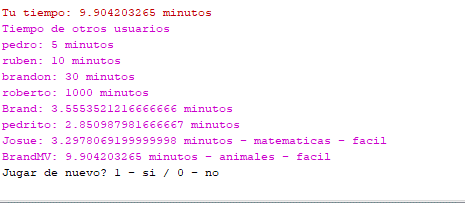


Ilustración 40. Finalizando el juego

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 41. Tiempo capturado en servidor.

Si el jugador selecciona que no quiere jugar más se cierra el socket.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 42. Cerrando conexión.

# **Conclusiones**

Saber sobre sockets de datagrama es muy importante pues son usados en aplicaciones done se requiere velocidad al enviar paquetes, un ejemplo es el juego desarrollado en esta práctica, pues queremos que el usuario reciba la mayor experiencia y que el juego sea rápido en el envío y recibimiento de datos.

Gracias a esta práctica pude comprender de una mejora manera el uso de los sockets de datagrama usando, en este caso, lenguaje Java para su aplicación en un juego de sopa de letras donde el envió y recibo de datos fue constante. Sin dudas fue una buena práctica para poner en práctica lo visto en clase relacionado a los sockets de datagrama y ver las diferencias con los sockets de flujo vistos en la práctica anterior.

Durante el desarrollo de esta práctica presente varios problemas, sobre todo al momento de mandar la sopa de letras, pues había momentos en los que nunca se enviaba, esto lo resolví mandando la sopa de letras ya creada para después recibirla en el cliente. De igual forma tuve conflicto en la lógica de buscar las palabras en la sopa de letras a partir de las coordenadas de inicio y fin, esto lo resolví de buena forma identificando la orientación de la palabra y a partir de unas condiciones con las coordenadas poder buscar la palabra de una manera eficiente en la sopa de letras.

Esta práctica represento un reto y fue divertido de solucionar, además de que me llevo un conocimiento reforzado sobre los sockets de datagrama.

# **Bibliografía**

1. Froufe, A. (2020). “Sockets de datagrama”. Obtenido de: https://mate.uprh.edu/~jse/cursos/4097/notas/java/javaEspanol/JavaTut/Cap9/socket.html