Proyecto: “CHAT PARA ENVÍO Y RECEPCIÓN DE MENSAJES DE TEXTO” Versión 0.0

Descripción y Modelo de Análisis

Elaborado por:
Arellano Hernández Israel
Cervantes García Eduardo
Moreno Razo Laura Mildred
Vásquez Martínez Fredin Alberto

15/DICIEMBRE/2021

Índice

Introducción

El rol de la programación en los últimos años ha sido crucial, su utilidad está presente en la mayor parte de actividades que desempeñamos, conocer su funcionamiento e historia permite comprender la evolución que ha tenido la sistematización de tareas y el manejo de la información que hoy en día damos como un hecho. En efecto, la misma tiene como principal función el hecho de conseguir que innumerables trabajos que antes ejercían de forma manual sean ejecutados por un ordenador, aunque otra de sus funciones es crear innovaciones que beneficien a las sociedades y aporten al bienestar humano. Uno de los principales ejemplos de la afirmación anterior es el uso de tecnología para lograr la comunicación a distancia. Es de los principales usos que se le da al internet hoy en día. Debido a esto, es que resulta de suma importancia comprender al menos de manera básica el funcionamiento y elementos que permiten generar sistemas con este propósito.

En el presente proyecto, se explicará la estructura de un sistema desarrollado en el lenguaje de programación Java que permite la comunicación entre varios usuarios de manera remota por medio de un chat global y el envío de mensajes privados que solo se dirigen al usuario deseado.

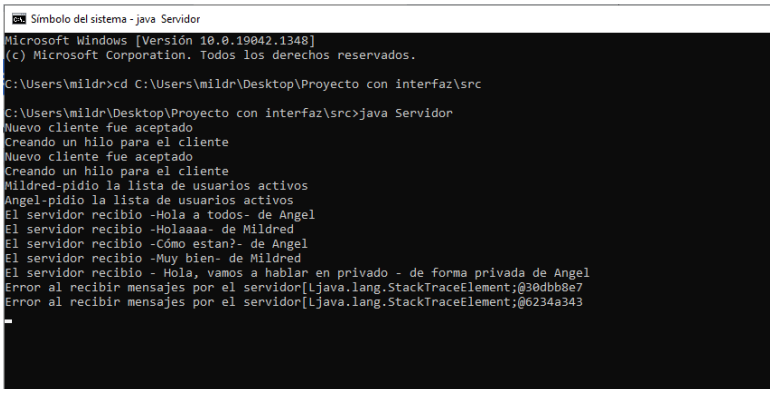
Para lograrlo, como se mencionó se utilizó Java, que implementa el paradigma orientado a objetos, que es tal vez el paradigma de programación más utilizado en el mundo del desarrollo de software y de la ingeniería de software del siglo XXI, pues trae un nuevo enfoque a los retos que se plantean. Se aplicaron todos los conceptos y aprendizajes adquiridos en la materia. Los principales aspectos son los hilos y sockets usados, que le dan la funcionalidad esperada para la comunicación. Por otro lado, el uso de swing permite tener una interfaz de usuario que resulta más amigable que la consola, pero que complicó el desarrollo del proyecto.

Como resultado final se obtuvieron dos clases principales, el servidor que permite escuchar las peticiones para realizar conexiones, razón por la cual no posee interfaz gráfica y otra clase principal llamada interfazInicioSesión a partir de la cual se llaman a todas las demás clases.

Para correr el programa primeramente se debe iniciar el servidor, para hacerlo con el archivo jar se debe escribir en la consola:

java -cp server.jar Servidor

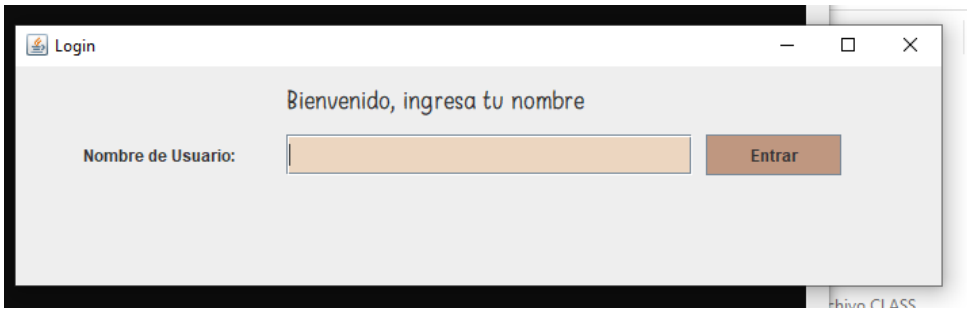
Se mantiene ahí mientras se desea que se conecten clientes. No posee interfaz, ya que no nos pareció necesario, debido a que solo notifica de las comunicaciones que están sucediendo e incluso se facilitaba su uso en consola durante las pruebas.



Una vez encendido el servidor, se pueden iniciar tantos clientes como se desee ya sea de manera remota, es decir cada uno ejecutando un archivo .jar en su equipo o de manera local, que es en la misma computadora con las instrucciones:

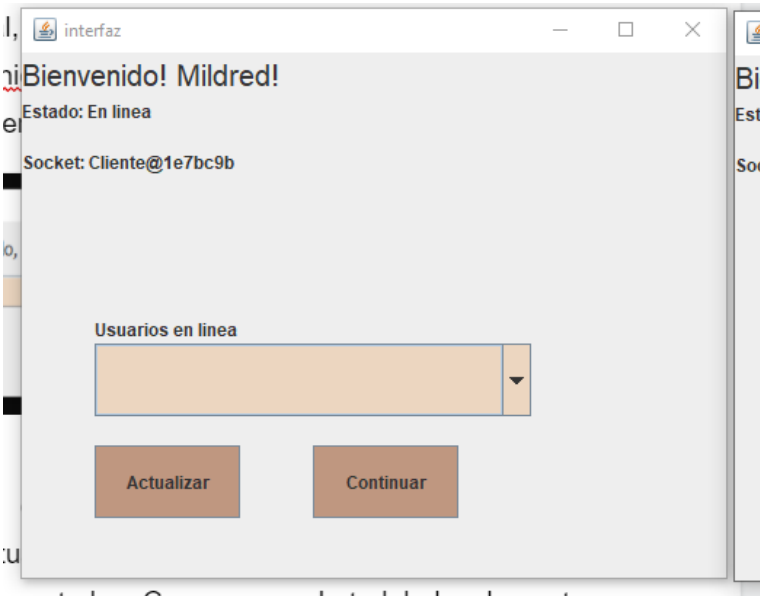
java -cp cliente.jar interfazInicioSesion

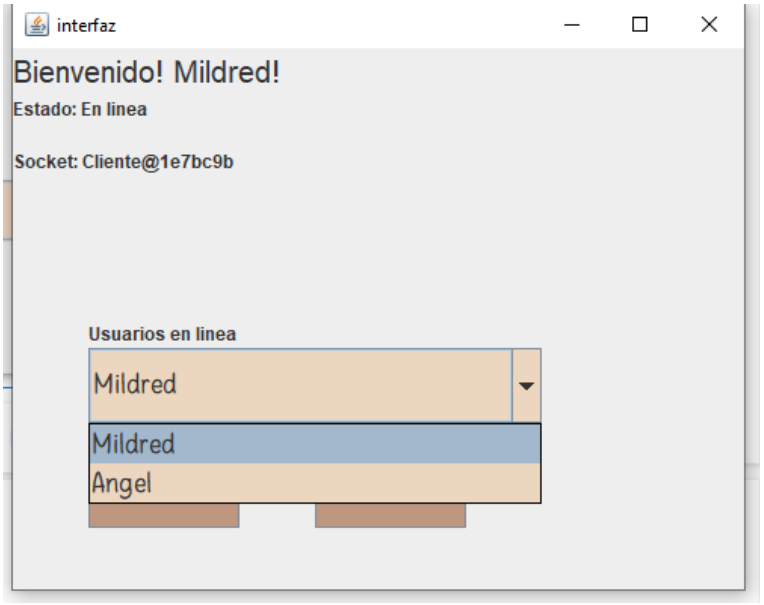
Lo primero que se observa es la interfazInicioSesion, debemos ingresar nuestro nombre para identificarnos y presionar entrar.



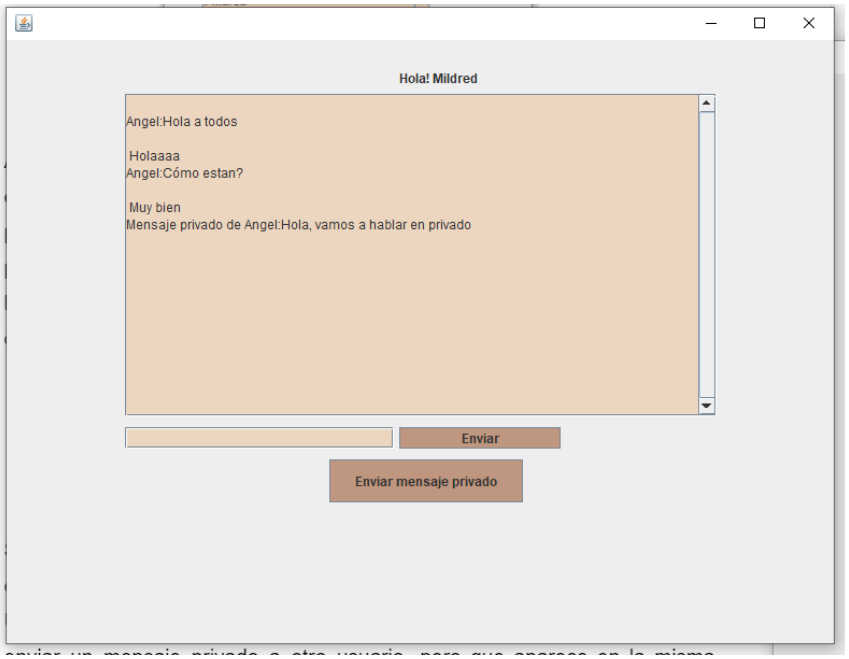
Esto lanza la interfazOnline, que nos muestra la pantalla de inicio del programa. Si presionamos actualizar podremos observar todos los nombres de los usuarios actualmente conectados.

Como es un chat global, solamente debemos presionar para continuar para unirnos a la conversación.





Ahora aquí podemos enviar y recibir los mensajes de todos aquellos que estén “chateando”. También está la opción de enviar mensaje privado, al presionarla solicita el nombre de la persona a la que quieres contactar por privado, después se debe escribir el mensaje y como se observa, solamente le llega al destinatario elegido. Podrás continuar chateando hasta que salgas de la ventana con el botón.



Se cumplió con el objetivo buscado respecto al funcionamiento, se logró la comunicación múltiple de clientes.

Una decisión relevante que tomamos, es realizar un chat global con opción de enviar un mensaje privado a otro usuario, pero que aparece en la misma ventana. Esto provino de las diversas dificultades y retos presentados por el uso de swing. Resultaba complicado poder entablar chats privados uno a uno, ya que al hacer la conexión podían existir muchas excepciones y situaciones que debíamos controlar, por ejemplo que el usuario destino también recibiera el mensaje o que varias personas quisieran hablar con una misma. Las ventanas necesitaban un identificador para saber a qué chat corresponden y así poder clasificar los mensajes y mostrarlos dependiendo de quién es el emisor. Es por ello que decidimos que en caso de desear enviar un mensaje privado se escriba el nombre del destinatario seguido del mensaje, de esta manera solamente se busca al hilo con ese nombre y se le pasa a su flujo, sin embargo, mostrándolo en la misma ventana.

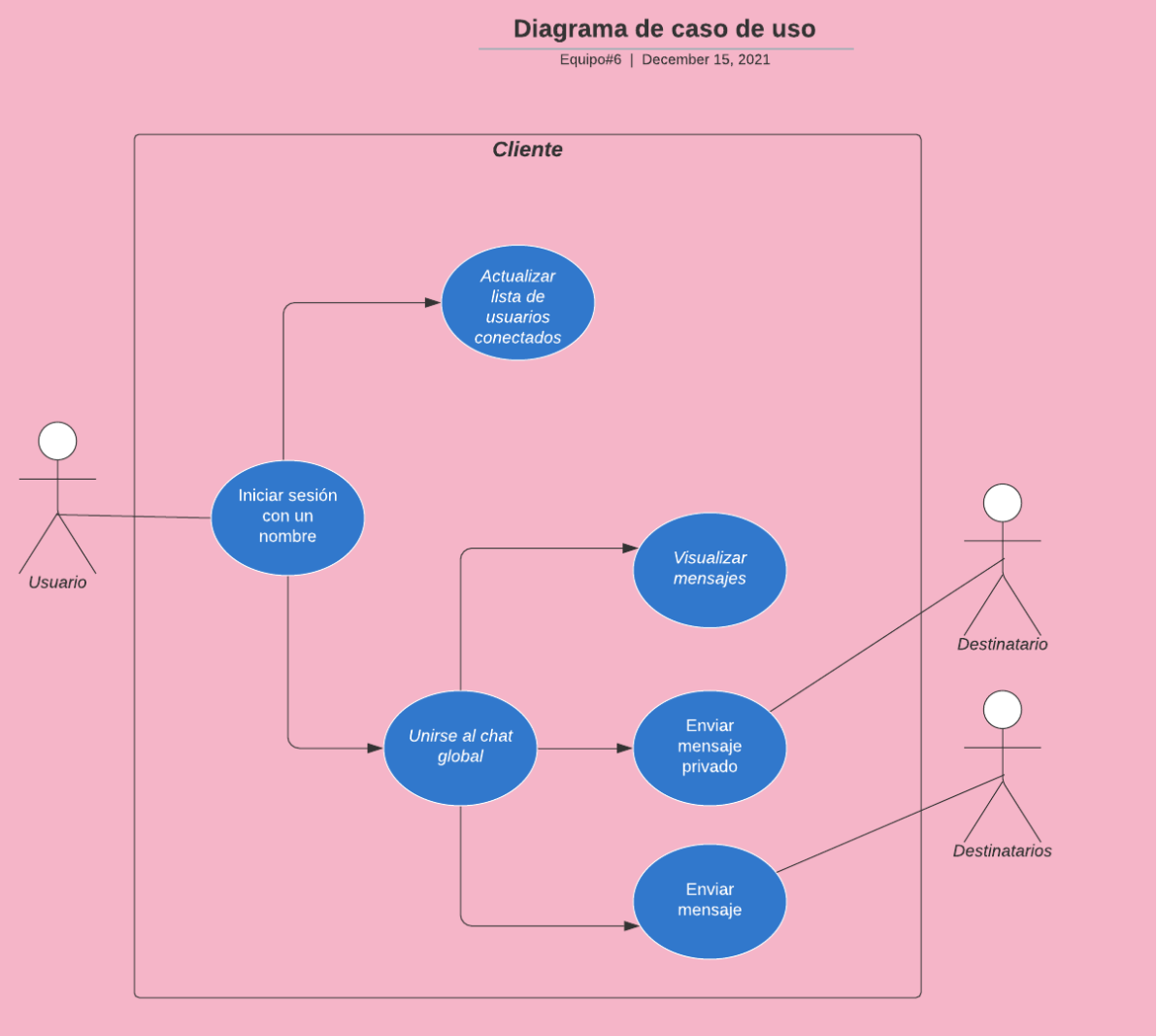
También decidimos separar las clases de las interfaces gráficas del código que da funcionamiento para solo realizar llamadas de métodos cuándo es necesario.

Creemos que de esta manera mantenemos un orden y aplicamos la abstracción y encapsulamiento aislando la funcionalidad de las clases.

Este chat puede tener diversas aplicaciones, comenzando por la principal que es la comunicación de varias personas con una aplicación de escritorio, pero también es una idea sumamente utilizada en páginas web. Usando el mismo concepto algunas páginas crean chats similares para juegos en línea o foros de comunicación para personas con intereses en común, es una aplicación popular en salas públicas. Un ejemplo es el famoso juego “pinturillo” que implementa un chat global en el cual te puedes comunicar con las personas activas y que estén jugando en tu sesión.

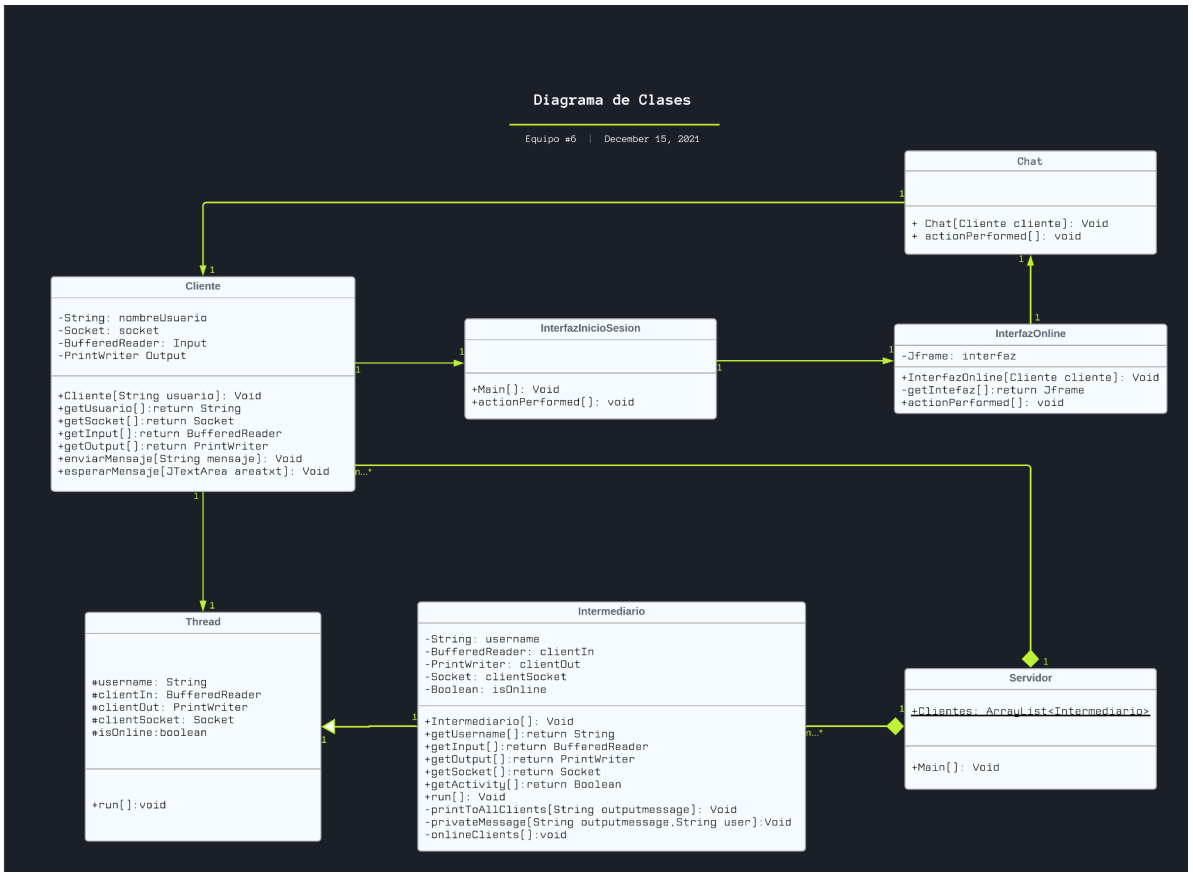
Ejemplo del chat de página web “pinturillo”

Casos de Uso-Análisis



Este diagrama muestra el funcionamiento del sistema en cuanto a la interacción con los usuarios. Se observa que la primera interacción posible es iniciar sesión con tu nombre. A partir de ahí, el usuario puede aplicar cualquiera de los casos que aparecen, pues el sistema lo permite. Actualizar la lista de usuarios o a partir de unirse a la conversación grupal enviar mensajes públicos, privados y ver lo que está recibiendo. Es una buena representación, ya que permite ver los objetivos y lo que se espera que el usuario pueda hacer con el sistema.

Diagrama de Clases



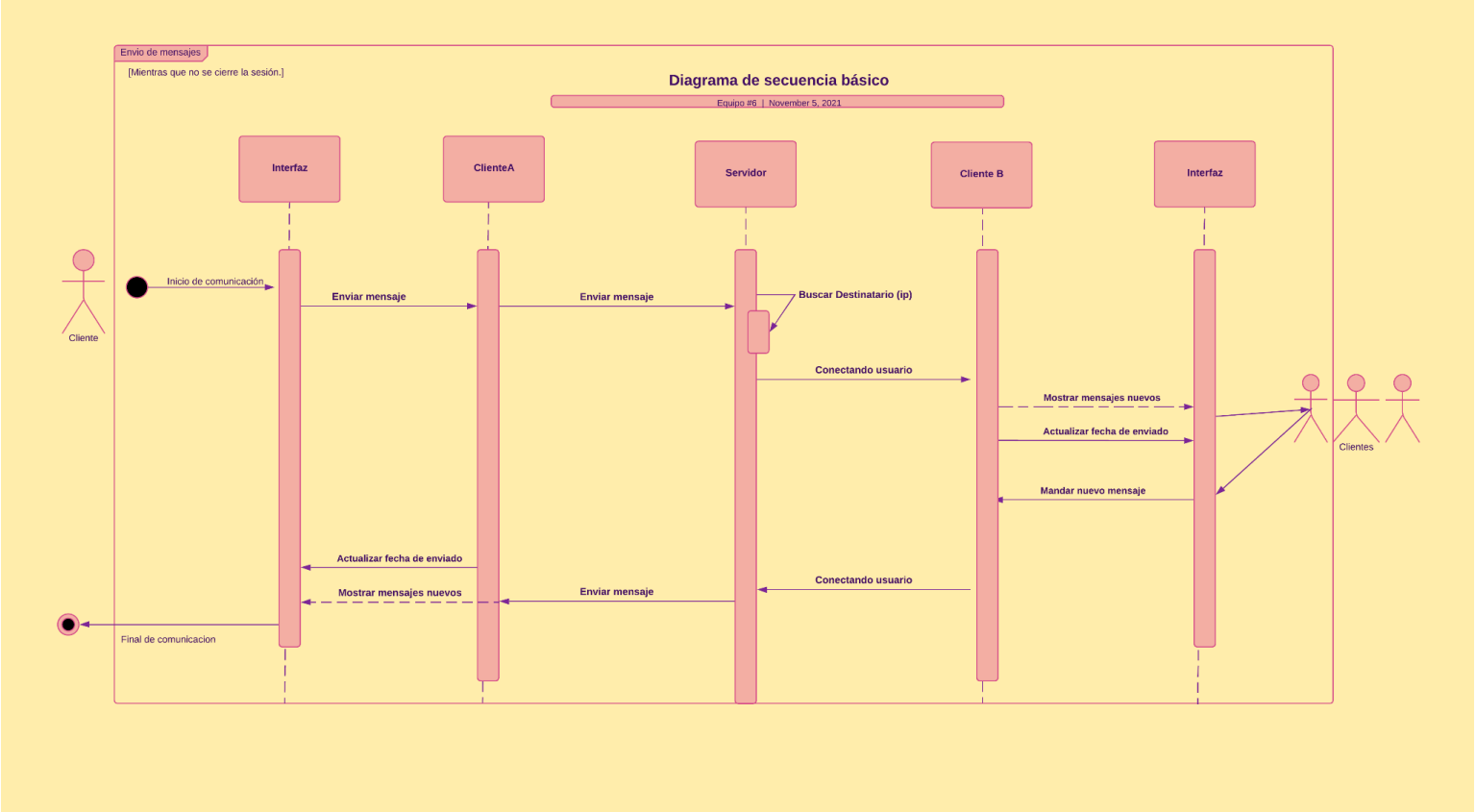
Este diagrama muestra todas las clases que tendrá el programa y que se utilizarán para establecer la conexión entre 2 o más máquinas para realizar una comunicación entre ellos. La comunicación estará siendo interpretada como un chat grupal, y se dispondrá de un botón para una comunicación privada. En el diagrama se muestra la relación entre las clases, así como la relación de herencia que tiene la clase Thread con respecto a las clases que hacen uso de ellas y la de asociación de las demás clases que solo se mandan a llamar para hacer uso de sus métodos.

La relación entre las clases se fundamenta en el código escrito realizado, además de que este diagrama sirvió como guía para realizar las asociaciones descritas.

El servidor e intermediario se relacionan por realización, pues forman parte de un todo, una no puede funcionar sin la otra. Tanto Intermediario como Cliente heredan de Thread, intermediario para permitir múltiples clientes y Cliente para tener un hilo en espera de mensajes nuevos en su input stream. “interfazCliente” es la primer ventana donde se inicia sesión, debido a esto ahí se instancia Cliente e interfazOnline, así que hay asociación. Finalmente en interfazOnline al presionar

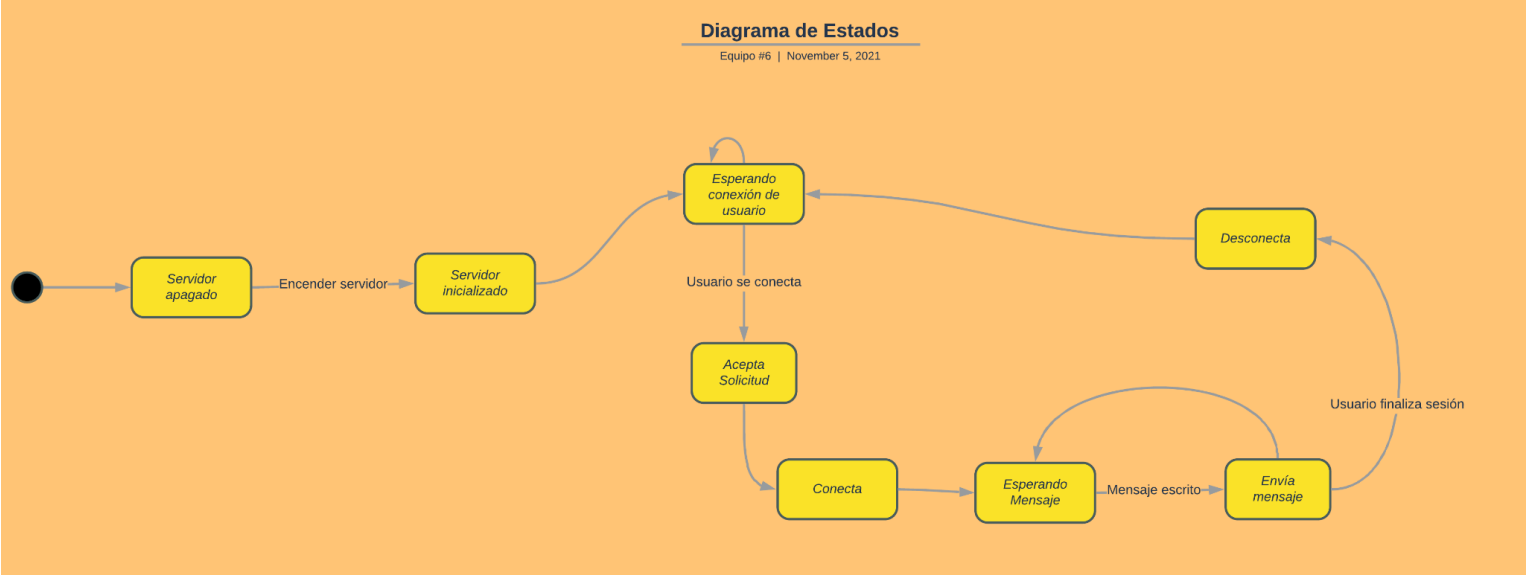
continuar es que se instancia un chat cuya clase contiene al cliente como atributo para permitir la entrada salida, así que también hay asociación.

Diagrama de Secuencia



Este diagrama describe la interacción entre clases que permite el envío, recepción y visualización de mensajes entre 2 clientes como mínimo, puesto que se permite la comunicación con todos los usuarios que se encuentran conectados. En el diagrama se narra la transición entre clases que estará haciendo el usuario para realizar el envío de un mensaje, encontrando al principio la interfaz para realizar el registro del usuario para después encontrar a los usuarios conectados, posteriormente entrar al chat el cual se realizará usando el servidor como intermediario, es por esto mismo que se observa al servidor entre el usuario que está funcionando como el destinatario.

Diagrama de Estado

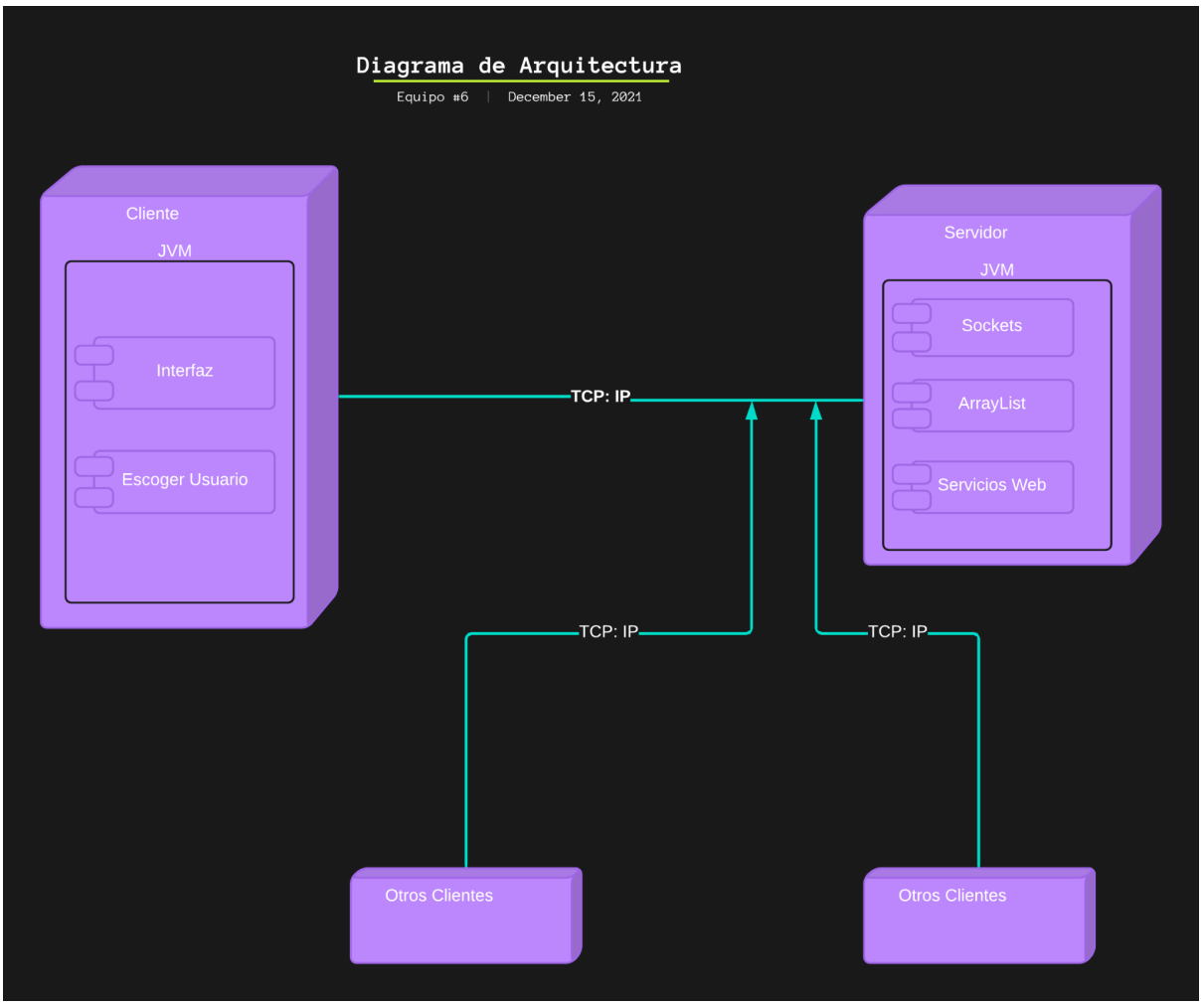


El diagrama presente describe los cambios de estado del servidor con respecto a lo que va ocurriendo conforme se ejecuta la aplicación. Es de este modo que es posible denotar lo siguiente:

1. El servidor no finalizará, estará en una constante espera de un nuevo usuario. Esto viene de la idea de que un servidor debe estar siempre encendido.
2. Será posible estar en un estado donde se denote una espera constante de nuevos mensajes, o en caso de no seguir con mensajes se regresa al estado de esperar, nuevamente, a un nuevo usuario.
3. La comunicación existirá entre 2 usuarios como mínimo, puesto que funciona como un chat grupal.

El flujo de las clases se sigue por los pasos que estará haciendo el usuario, teniendo siempre como primero la inicialización del servidor el cual será el encargado de atender a las solicitudes de los clientes e ir aceptandolos conforme van llegando para poder entrar este bucle infinito el cual estará implementando la capacidad de atender.

Descripción de Arquitectura



Este diagrama muestra la arquitectura del programa; cómo funcionará en términos generales y cómo se comunicarán entre sí 2 máquinas que estén chateando. Será a través del cliente el cual se podrá ver como nodo y contendrá una interfaz así como escoger un usuario que será la lista de usuarios que se encuentran conectados en ese momento, este mismo se estará entablando conectándose al servidor por medio de la red, el cual en este caso todo es remoto. Durante esa conexión también otros usuarios podrán realizar la misma conexión

Conclusión

Lo que se realizó en este proyecto fue un sistema de mensajería, el cual primeramente pide introducir el nombre de cada cliente para posteriormente, si se abren 2 o más clientes, tenemos la opción de escoger con quién realizar el envío de mensajes. Cuando otro cliente se conecta automáticamente formamos un grupo de todos los clientes conectados, no obstante existe la opción de mandar un mensaje privado al cliente que se elija.

Como análisis de resultados, se puede observar que la comunicación se hace de manera grupal al principio para después escoger un modo de mensajería privado, esto es desde la interfaz del chat, la manera de efectuar el envío de un mensaje privado es por medio de un evento asociado a un botón el cual despliega en la pantalla donde se ve reflejado el texto de los mensajes. Este mensaje dicta que se escriba el nombre del usuario (que debe de estar conectado) al que desea realizar el envío del mensaje privado.

Posteriormente a esto se entablará una conexión de ese usuario A con el usuario B, de modo que al enviar el usuario A un mensaje al usuario B, donde este último no ha declarado que desea comunicarse con A, el mensaje se verá reflejado en la pantalla de texto (entrada) del usuario B. Con esto último queda al descubierto que ambas partes no deben de estar de acuerdo para que el envío de mensaje se pueda efectuar, algo que sucede en la vida real con aplicaciones como Messenger.

Aunque hay que dejar en claro que todos estos mensajes se verán en una única pantalla donde está colocado un *JTextArea* con el modo de editar en false para que los usuarios solo puedan ver los mensajes más no alterarlos.

Por otra parte, la importancia que logra conseguir este botón para enviar mensajes privados se verá reflejada para realizar envíos de mensajes privados de forma constante, esto es debido a que al escoger esta opción de envío privado únicamente será válido para un único mensaje, cualquier otro mensaje después de este ya no

será privado sino que, a modo de grupo, sé consigue que el mensaje se refleje a todos los participantes, al menos que el siguiente mensaje también se escoja que sea privado.

Otro análisis posible de realizar es al observar el comportamiento de los usuarios conectados y cómo se logran observar en la pantalla, puesto que al realizar una nueva conexión no se verá reflejada automáticamente al menos que se active el evento del botón que funciona para actualizar, de tal modo que se podrá activar dichos eventos las veces que se quiera sin que exista alteraciones en el programa, debido a que en una circunstancia real no será posible saber cuántas veces se deberá de dar click en actualizar para mostrar a todos los usuarios.

Cuando un nuevo cliente se une a la conversación y si ya existe, los mensajes mandados de manera grupal se verán reflejados en el área de texto de este nuevo usuario. Otra cosa agregar será que se podrá salir de la conversación sin tener problema alguno.

Ahora, con respecto a la lista de usuarios conectados solo sirve para ver qué usuarios están conectados más no para escoger a quién mandarle mensaje. Y este modo de observar a los usuarios únicamente se encuentra disponible cuando el usuario ingresa por primera vez, ya que en grupo no se verá reflejado de ninguna forma que hay un nuevo usuario conectado, al menos que este último mande mensaje.

Por último, hay que analizar el servidor el cual no tendrá manifestación gráfica alguna, lo cual genera que su cierre se haga desde consola. Por aparte de esto, el servidor se hará valer por medio de la clase Intermediario el cual hará la función de comunicación realmente.

Los objetivos del proyecto fueron cumplidos, tal vez la interpretación de envío de mensajes privados no sea la adecuada así como la lista de usuarios conectados, pero se logró el cometido de realizar una comunicación entre usuarios conectados a un servidor el cual estará funcionando como intermediario, así como el uso de interfaces para una visualización un poco más cómoda para un usuario final.

No obstante, sí conocemos y empezamos a desarrollar la manera de realizar una comunicación de 1 a 1 y no empezar de manera grupal, el problema es que el factor tiempo estuvo en contra además de que en cierto punto llegaba a ser un poco complejo y la lectura de algunos flujos de entrada parecía que no estaban recibiendo nada cuando el server (funcionando como intermediario) si estaba realizando el envío del mensaje. Lo que sí podemos asegurar es que aprendimos bastante con respecto al manejo de eventos y la incorporación de hilos, nos hubiera gustado un poco más de tiempo para poder completar esto de una manera mucho mejor, pero como siempre lo que más falta es tiempo y no las ganas o creatividad en los proyectos. También este modo en línea produce un poco de complejidad para trabajar en equipo en proyectos como este, y más cuando se trata de la primera vez de realizar un proyecto así por equipos.